DUMPSBOSS.

Oracle Database 12c SQL

Oracle 1z0-071

Version Demo

Total Demo Questions: 68

Total Premium Questions: 683

Buy Premium PDF

https://dumpsboss.co

support@dumpsboss.co

support@dumpsboss.co
dumpsboss.co

Topic Break Down

Topic	No. of Questions
Topic 1, Relational Database Concepts	25
Topic 2, Retrieving Data using the SQL SELECT Statement	27
Topic 3, Restricting and Sorting Data	58
Topic 4, Using Single-Row Functions to Customize Output	47
Topic 5, Using Conversion Functions and Conditional Expressions	58
Topic 6, Reporting Aggregated Data Using the Group Functions	39
Topic 7, Displaying Data from Multiple Tables	56
Topic 8, Using Subqueries to Solve Queries	42
Topic 9, Using the Set Operators	39
Topic 10, Managing Tables using DML statements	89
Topic 11, Introduction to Data Definition Language	93
Topic 12, Creating Schema Objects	57
Topic 13, Managing Objects with Data Dictionary Views	18
Topic 14, Manipulating Large Data Sets	8
Topic 15, Managing Data in Different Time Zones	9
Topic 16, Retrieving Data using Subqueries	5
Topic 17, Mix Questions	13
Total	683

QUESTION NO: 1

View the Exhibit and examine the structure of the CUSTOMERS table.

Table CUSTOMERS		
Name	Null?	Type
CUST_ID	NOT_NULL	NUMBER
CUST_FIRST_NAME	NOT_NULL	VARCHAR2(20)
CUST_LAST_NAME	NOT_NULL	VARCHAR2(20)
CUST_GENDER	NOT_NULL	CHAR(1)
CUST_YEAR_OF_BIRTH	NOT_NULL	NUMBER(4)
CUST_MARITAL_STATUS		VARCHAR2(20)
CUST_STREET_ADDRESS	NOT_NULL	VARCHAR2(40)
CUST_POSTAL_CODE	NOT_NULL	VARCHAR2(10)
CUST_CITY	NOT_NULL	VARCHAR2(30)
CUST_STATE_PROVINCE	NOT_NULL	VARCHAR2(40)
COUNTRY_ID	NOT_NULL	NUMBER
CUST_INCOME_LEVEL		VARCHAR2(30)
CUST_CREDIT_LIMIT		NUMBER
CUST_EMAIL		VARCHAR2(30)

Evaluate the following SQL statement:

```
SQL> SELECT cust_city, COUNT(cust_last_name)
FROM customers
WHERE cust_credir_limit > 1000
GROUP BY cust_city
HAVING AVG(cust_credit_limit) BETWEEN 5000 AND 6000;
```

Which statement is true regarding the outcome of the above query?

A. It returns an error because the BETWEEN operator cannot be used in the HAVING clause.

B. It returns an error because WHERE and HAVING clauses cannot be used in the same SELECT statement.

C. It returns an error because WHERE and HAVING clauses cannot be used to apply conditions on the same column.

D. It executes successfully.

ANSWER: D

Explanation:

It executes successfully. Oracle permits both `WHERE` and `HAVING` in one `SELECT` statement because they operate at different stages of query processing. The `WHERE` clause filters individual rows from `CUSTOMERS` before rows are placed into groups. After the `GROUP BY` operation forms the result groups, the `HAVING` clause filters those groups. Consequently, conditions involving `CUST_CREDIT_LIMIT` can be used in both clauses when the query's grouping makes that column available to the `HAVING` clause, as shown in the statement. Oracle also supports the `BETWEEN` condition in a `HAVING` clause; it tests whether a value is within an inclusive lower and upper bound. The query therefore applies its row-level filter, performs aggregation for the applicable groups, and then applies the group-level credit-limit condition without a syntax error.

Oracle's documentation describes the roles of `WHERE`, `GROUP BY`, and `HAVING` in a query block in the SELECT statement reference. The inclusive semantics of the range condition are documented under BETWEEN Condition.

QUESTION NO: 2

Which three statements are true about views in an Oracle Database? (Choose three.)

A. Views can join tables only if they belong to the same schema.

B. A view can be created that refers to a non-existent table in its defining query.

C. Views have no object number.

D. Views have no segment.

E. Rows inserted into a table using a view are retained in the table if the view is dropped.

F. A SELECT statement cannot contain a WHERE clause when querying a view containing a WHERE clause in its defining query.

ANSWER: B D E

Explanation:

A view can be created that refers to a non-existent table in its defining query when the `FORCE` keyword is used in the `CREATE VIEW` statement. This lets Oracle create the view even though its base object does not currently exist or is inaccessible to the view owner. Such a view is not usable until the referenced object exists and the view can be successfully compiled. Oracle documents this behavior as the distinction between `CREATE FORCE VIEW` and the default `NOFORCE` behavior. See the Oracle Database 12c CREATE VIEW documentation.

Views have no segment because an ordinary view is a stored data-dictionary definition of a query, rather than a separately stored copy of the query result. When queried, Oracle accesses the underlying base tables or other referenced objects to return the appropriate rows. Oracle's storage architecture defines segments as space allocated for database objects that store data, such as tables and indexes; an ordinary view does not allocate such storage. See the Oracle Database 12c logical storage structures documentation.

Rows inserted into a table using a view are retained in the table if the view is dropped. For an updatable view, the view provides an interface through which the underlying table is modified. Dropping that view removes only the view definition; it does not remove, alter, or roll back data already stored in the base table.

QUESTION NO: 3

You create a table by using this command:

CREATE TABLE rate_list (rate NUMBER(6,2));

Which two are true about executing statements? (Choose two.)

A. INSERT INTO rate_list VALUES (-10) produces an error.

B. INSERT INTO rate_list VALUES (87654.556) inserts the value as 87654.6.

C. INSERT INTO rate_list VALUES (0.551) inserts the value as .55.

D. INSERT INTO rate_list VALUES (-99.99) inserts the value as 99.99.

E. INSERT INTO rate_list VALUES (0.999) produces an error.

F. INSERT INTO rate_list VALUES (-.9) inserts the value as -.9.

ANSWER: C F

Explanation:

A `NUMBER(6,2)` column has a precision of six digits in total and a scale of two digits to the right of the decimal point. When a numeric value has more fractional digits than the declared scale, Oracle rounds the value to the column scale as part of storing it. Therefore, `INSERT INTO rate_list VALUES (0.551) inserts the value as .55.` is valid: the third fractional digit does not cause an error, and the stored numeric value is 0.55. Oracle numbers do not preserve display formatting such as a required leading zero, so ".55" denotes the same numeric value as 0.55.

`INSERT INTO rate_list VALUES (-.9) inserts the value as -.9.` is also valid. A signed numeric literal with a leading decimal point is accepted, and its value fits comfortably within the precision and scale limits. Internally, the value conforms to scale two, equivalently -0.90; numeric display tools may render it as -.9 or -0.9 because insignificant trailing zeroes are not intrinsic to a NUMBER value. Oracle documents precision and scale behavior, including rounding when a value exceeds the specified scale, in its SQL Language Reference and describes the NUMBER datatype.

QUESTION NO: 4

Examine the description of the EMPLOYEES table:

The session time zone is the same as the database server

Which two statements will list only the employees who have been working with the company for more than five years?

A. SELECT employee_ name FROM employees WHERE (SYSDATE – hire_ data) / 365>5

B. SELECT employee_ name FROM employees WHERE (SYSTIMESTAMP – hire_ data) / 365>

C. SELECT employee_ name FROM employees WHERE (CUARENT_ DATE – hire_ data / 365>5

D. SELECT employee_ name FROM employees WHERE (SYSNAYW – hire_ data / 12> 3

E. SELECT employee_ name FROM employees WHERE (SYSNAYW – hire_ data / 12> 3

F. SELECT employee_ name FROM employees WHERE (CUNACV_ DATE – hire_ data / 12> 3

G. SELECT employee_name FROM employees WHERE hire_date < ADD_MONTHS(SYSDATE, -60)

ANSWER: G

Explanation:

`SELECT employee_name FROM employees WHERE hire_date < ADD_MONTHS(SYSDATE, -60)` correctly identifies employees whose hire date is earlier than the date exactly 60 calendar months before the current database-server date and time. Using `ADD_MONTHS` expresses five years as five calendar-year intervals rather than as a fixed number of days. This is important because calendar periods can include leap days and months have differing lengths; a calculation based on dividing a day difference by 365 can classify an employee incorrectly near a five-year anniversary.

`SYSDATE` returns the current date and time of the database server's operating system as an Oracle `DATE` value. Since the session time zone is stated to be the same as the database server, it is appropriate for evaluating the stated business condition. Subtracting 60 months from `SYSDATE` establishes the precise cutoff, and the strict less-than comparison implements "more than five years" rather than "five years or more." Oracle documents that `ADD_MONTHS` returns the date plus an integer number of months and adjusts for month-end dates where necessary. See Oracle's ADD_MONTHS documentation and SYSDATE documentation.

QUESTION NO: 5

Examine the data in the PRODUCTS table:

PROD ID	PROD NAME	PROD LIST	CATEGORY ID
101	Plate	10	1
102	Cup	20	1
103	Saucer	20	1
104	Knife	30	1
105	Fork	30	1

Examine these queries:

1. SELECT prod name, prod list

FROM products

WHERE prod 1ist NOT IN(10 ， 20) AND category _id=1;

2. SELECT prod name, | prod _ list

FROM products

WHERE prod list < > ANY (10 ， 20) AND category _id= 1;

SELECT prod name, prod _ list

FROM products

WHERE prod_ list < > ALL (10 ， 20) AND category _ id= 1;

Which queries generate the same output?

A. 1 and 3

B. 1, 2 and 3

C. 2 and 3

D. 1 and 2

ANSWER: A

Explanation:

1 and 3 generate the same output because both predicates require `prod_list` to be different from every value in the set containing 10 and 20, while also requiring `category_id = 1`. The condition `prod_list NOT IN (10, 20)` is logically equivalent to requiring both `prod_list <> 10` and `prod_list <> 20`. Similarly, `prod_list <> ALL (10, 20)` means that the comparison must be true for all values in the set: the value must be unequal to 10 and unequal to 20. Therefore, for each row in category 1, both queries retain precisely the rows whose product-list value is neither 10 nor 20.

This equivalence also has consistent null behavior: when `prod_list` is null, neither predicate evaluates to true, so such a row is not returned by either `WHERE` clause. Oracle documents `NOT IN` as a membership condition and describes `ALL` as requiring a comparison to be true for every value returned by the relevant set or subquery. See the Oracle SQL Language Reference: IN Condition and the Oracle SQL Language Reference: Comparison Conditions.

QUESTION NO: 6

Examine the command to create the BOOKS table.

SQL> create table books(book id CHAR(6) PRIMARY KEY,

title VARCHAR2(100) NOT NULL,

publisher_id VARCHAR2(4)?

author_id VARCHAR2 (50));

The BOOK ID value 101 does not exist in the table.

Examine the SQL statement.

insert into books (book id title, author_id values

('101'?'LEARNING SQL','Tim Jones')

A. It executes successfully and the row is inserted with a null PUBLISHER_ID.

B. It executes successfully only if NULL is explicitly specified in the INSERT statement.

C. It executes successfully only NULL PUBLISHER_ID column name is added to the columns list in the INSERT statement.

D. It executes successfully onlyif NULL PUBLISHER ID column name is added to the columns list and NULL is explicitly specified In the INSERT statement.

ANSWER: A

Explanation:

It executes successfully and the row is inserted with a null PUBLISHER_ID. In an `INSERT` statement, Oracle permits a column list that includes only the columns for which values are being supplied. Any table column omitted from that list receives its default value, if one was defined; otherwise, it receives `NULL`. Because `PUBLISHER_ID` has no `NOT NULL` constraint and no default value is shown in the table definition, omitting it causes Oracle to store `NULL` for that column.

The supplied values satisfy the relevant requirements: the new book identifier does not duplicate an existing primary-key value, the primary key cannot be null, and a value is provided for `TITLE`, which is mandatory. Oracle therefore inserts the values specified for the book identifier, title, and author identifier while leaving `PUBLISHER_ID` null. This behavior is an important practical feature of column-list inserts: applications can provide only the attributes available at insert time, provided all mandatory columns are supplied. Oracle documents that omitted columns take their default value, or null when no default is specified. See the Oracle Database SQL Language Reference: INSERT and the CREATE TABLE reference.

QUESTION NO: 7

Examine the description of the EMPLOYEES table:

```
Name          Null?     Type
------------  --------  ------
EMPLOYEE_ID   NOT NULL  NUMBER(4)
LAST_NAME               VARCHAR2(10)
HIRE_DATE               DATE
SALARY                  NUMBER(6,2)
```

Examine these requirements:

1. Display the last name, date of hire and the number of years of service for each employee.

2. If the employee has been employed 5 or more years but less than 10, display "5+ years of service".

3. If the employee has been employed 10 or more years but less than 15, display "10+ years of service".

4. If the employee has been employed 15 or more years, display "15+ years of service".

5. If none of these conditions matches, display "<5 years of service".

6. Sort the results by the HIRE_DATE column.

Which statement satisfies all the requirements?

A.
```
SELECT last_name, hire_date,
 (CASE WHEN (SYSDATE - TO_YMINTERVAL('15-0')) >= hire_date THEN
'15+ years of service'
       WHEN (SYSDATE - TO_YMINTERVAL('10-0')) >= hire_date THEN
'10+ years of service'
       WHEN (SYSDATE - TO_YMINTERVAL('5-0')) >= hire_date THEN
'5+ years of service'
       ELSE '<5 years of service'
  END) AS years
  FROM employees
 ORDER BY hire_date;
```

B.
```
SELECT last_name, hire_date,
 (CASE WHEN (SYSDATE - hire_date) >= TO_YMINTERVAL('5-0') THEN
'5+ years of service'
       WHEN (SYSDATE - hire_date) >= TO_YMINTERVAL('10-0') THEN
'10+ years of service'
       WHEN (SYSDATE - hire_date) >= TO_YMINTERVAL('15-0') THEN
'15+ years of service'
       ELSE '<5 years of service'
  END) AS years
  FROM employees
 ORDER BY hire_date;
```

C.
```
SELECT last_name, hire_date,
 (CASE WHEN (SYSDATE - hire_date) >= TO_YMINTERVAL('15-0') THEN
'15+ years of service'
       WHEN (SYSDATE - hire_date) >= TO_YMINTERVAL('10-0') THEN
'10+ years of service'
       WHEN (SYSDATE - hire_date) >= TO_YMINTERVAL('5-0') THEN
'5+ years of service'
       ELSE '<5 years of service'
  END) AS years
  FROM employees
 ORDER BY hire_date;
```

D.
```
SELECT last_name, hire_date,
 (CASE WHEN (SYSDATE - TO_YMINTERVAL('5-0')) >= hire_date THEN
'5+ years of service'
       WHEN (SYSDATE - TO_YMINTERVAL('10-0')) >= hire_date THEN
'10+ years of service'
       WHEN (SYSDATE - TO_YMINTERVAL('15-0')) >= hire_date THEN
'15+ years of service'
       ELSE '<5 years of service'
  END) AS years
  FROM employees
 ORDER BY hire_date;
```

A. Option A

B. SELECT last_name, hire_date, CASE WHEN hire_date <= ADD_MONTHS(SYSDATE, -180) THEN '15+ years of service' WHEN hire_date <= ADD_MONTHS(SYSDATE, -120) THEN '10+ years of service' WHEN hire_date <= ADD_MONTHS(SYSDATE, -60) THEN '5+ years of service' ELSE '<5 years of service' END AS years_of_service FROM employees ORDER BY hire_date;

C. Option C

D. Option D

ANSWER: B

Explanation:

The statement using a searched `CASE` expression with service thresholds tested from highest to lowest satisfies the required mutually exclusive ranges. `ADD_MONTHS(SYSDATE, -180)`, `ADD_MONTHS(SYSDATE, -120)`, and `ADD_MONTHS(SYSDATE, -60)` establish calendar-based cutoffs for fifteen, ten, and five years of employment. Testing the fifteen-year cutoff first ensures that an employee who has completed fifteen years receives the required `'15+ years of service'` label rather than a lower-range label. The next tests assign the ten-year and five-year labels, while `ELSE` reliably covers employees with fewer than five completed years of service.

A searched `CASE` expression evaluates conditions in order and returns the result associated with the first condition that evaluates to true, which makes it appropriate for this tiered classification. Selecting `last_name` and `hire_date` alongside the calculated service label fulfills the requested display columns. Finally, `ORDER BY hire_date` sorts the output on the actual date column, as required. Oracle documents searched `CASE` evaluation at CASE Expressions and calendar month arithmetic at ADD_MONTHS.

QUESTION NO: 8

On your Oracle 12c database, you invoked SQL *Loader to load data into the EMPLOYEES table in the HR schema by issuing the following command:

$> sqlldr hr/hr@pdb table=employees

Which two statements are true regarding the command? (Choose two.)

A. It succeeds with default settings if the EMPLOYEES table belonging to HR is already defined in the database.

B. It fails because no SQL *Loader data file location is specified.

C. It fails if the HR user does not have the CREATE ANY DIRECTORY privilege.

D. It fails because no SQL *Loader control file location is specified.

E. SQL*Loader Express Mode uses employees.dat in the current directory by default when no data file is specified.

ANSWER: A E

Explanation:

The command uses SQL*Loader Express Mode, which is available in Oracle Database 12c. Supplying the `TABLE=employees` parameter identifies the target table and lets SQL*Loader derive the load definition from that table's metadata, rather than requiring a separately authored control file. Therefore, provided that `HR.EMPLOYEES` already exists and the normal default input requirements are met, the command can proceed with Express Mode defaults.

In this mode, SQL*Loader also supplies a default data-file convention. When no `DATA` parameter is provided, it looks for a data file named after the specified table, using the default `.dat` extension; here that is `employees.dat` in the current working directory. This is why the command can omit both an explicit control-file parameter and an explicit data-file parameter. SQL*Loader is a client utility and reads its input file from the client environment, so this invocation does not depend on creating an Oracle directory object.

Oracle documents Express Mode, its table-driven invocation, and its default file handling in the Oracle SQL*Loader Express Mode documentation. General SQL*Loader command-line parameter behavior is covered in the SQL*Loader Command-Line Reference.

QUESTION NO: 9

Which two statements are true regarding CASE expressions? (Choose two.)

A. A simple CASE expression compares one expression with one or more comparison expressions.

B. A searched CASE expression can evaluate Boolean conditions.

C. The ELSE clause is mandatory in every CASE expression.

D. A CASE expression can be used only in the WHERE clause.

E. A simple CASE expression cannot return a character value.

ANSWER: A B

Explanation:

A simple `CASE` expression compares one expression with one or more comparison expressions. For example, `CASE department_id WHEN 10 THEN 'Administration' END` compares `department_id` to the value 10.

A searched `CASE` expression evaluates one or more Boolean conditions. For example, `CASE WHEN salary > 10000 THEN 'HIGH' END` evaluates the condition for each row. Both simple and searched CASE expressions can include an optional `ELSE` clause. If no condition matches and no ELSE clause is supplied, Oracle returns null. See the Oracle Database SQL Language Reference: CASE Expressions.

QUESTION NO: 10

You need to calculate the number of days from 1st January 2019 until today.

Dates are stored in the default format of DD-MON-RR.

Which two queries give the required output?

A. SELECT SYSDATE-TO_DATE ('01-JANUARY-2019') FROM DUAL;

B. SELECT TO_DATE (SYSDATE, 'DD/MONTH/YYYY')-'01/JANUARY/2019' FROM DUAL;

C. SELECT ROUND (SYSDATE-TO_DATE ('01/JANUARY/2019')) FROM DUAL;

D. SELECT TO_CHAR (SYSDATE, 'DD-MON-YYYY')-'01-JAN-2019' FROM DUAL;

E. SELECT ROUND (SYSDATE- '01-JAN-2019') FROM DUAL:

F. SELECT SYSDATE - TO_DATE('01-JAN-2019', 'DD-MON-YYYY') FROM DUAL;

G. SELECT SYSDATE - DATE '2019-01-01' FROM DUAL;

ANSWER: F G

Explanation:

`SELECT SYSDATE - TO_DATE('01-JAN-2019', 'DD-MON-YYYY') FROM DUAL;` correctly produces the elapsed number of days because both operands are Oracle `DATE` values. The explicit `DD-MON-YYYY` format model matches the supplied four-digit year exactly, creating the date at midnight on 1 January 2019. Oracle defines subtraction of one `DATE` from another as a numeric result measured in days; the result can include a fractional portion representing the time elapsed since midnight today.

`SELECT SYSDATE - DATE '2019-01-01' FROM DUAL;` produces the same type of result. A datetime literal uses the unambiguous ANSI date-literal syntax `DATE 'YYYY-MM-DD'`, so its interpretation is independent of the session's `NLS_DATE_FORMAT`. This is generally a robust choice in SQL because it avoids dependence on implicit character-to-date conversion settings. In both queries, `SYSDATE` supplies the database server's current date and time, and subtracting the 1 January 2019 date yields the required elapsed-day value. Oracle documents date arithmetic and datetime literals at Datetime and Interval Datatypes and Datetime Literals.

QUESTION NO: 11

In the PROMOTIONS table, the PROMO_ BEGIN_DATE column is of data type and the default date format is DD-MON-RR

Which two statements are true about expressions using PROMO_ BEGIN_DATE in a query?

A. TONUMBER (PROMO BEGIN_DATE) - 5 will return a number

B. PROMO_ BEGIN_DATE - 5 will return a date

C. PROMO_ BEGIN_DATE - SYSDATE will return a number

D. PROMO_ BEGIN_DATE - SYSDATE will return an error

E. TODATE(PROMO BEGIN_DATE *5) will return a date

ANSWER: B C

Explanation:

`PROMO_ BEGIN_DATE - 5 will return a date` is correct because Oracle supports date arithmetic with numeric values. A number added to or subtracted from a `DATE` represents a number of days. Therefore, subtracting 5 from the date stored in `PROMO_ BEGIN_DATE` produces another `DATE` value that is exactly five calendar days earlier, while retaining the date's time component if one is stored.

`PROMO_ BEGIN_DATE - SYSDATE will return a number` is also correct. Both operands are Oracle `DATE` values: the column supplies a date for each row and `SYSDATE` supplies the database server's current date and time. Oracle date subtraction returns a numeric difference measured in days. The result may include a fractional portion, because Oracle `DATE` values include time to the second; for example, a difference of 1.5 represents one day and twelve hours. The display format `DD-MON-RR` affects character rendering or conversion of dates, not the fundamental arithmetic result types for these expressions. Oracle documents that adding or subtracting a number from a datetime yields a datetime, and subtracting one datetime from another yields a number of days. See the Oracle SQL Language Reference and the Oracle datetime data type documentation.

QUESTION NO: 12

View the Exhibit and examine the details of the PRODUCT_INFORMATION table.

Exhibit

PRODUCT_NAME	CATEGORY_ID	SUPPLIER_ID
Inkjet C/8/HQ	12	102094
Inkjet C/4	12	102090
LaserPro 600/6/BW	12	102087
LaserPro 1200/8/BW	12	102099
Inkjet B/6	12	102096
Industrial 700/HD	12	102086
Industrial 600/DQ	12	102088
Compact 400/LQ	12	102087
Compact 400/DQ	12	102088
HD 12GB /R	13	102090
HD 10GB /|	13	102071
HD 12GB @7200 /SE	13	102057
HD 18.2GB @10000 /E	13	102078
HD 18.2GB@10000 /|	13	102050
HD 18GB /SE	13	102083
HD 6GB /|	13	102072
HD 8.2GB @5400	13	102093

You must display PRODUCT_NAME from the table where the CATEGORY_ID column has values 12 or 13, and the SUPPLIER_ID column has the value 102088.

You executed this SQL statement:

```
SELECT product_name
FROM product_information
WHERE (category_id = 12 AND category_id = 13) AND supplier_id = 102088;
```

Which statement is true regarding the execution?

A. It would not execute because the entire WHERE clause is not enclosed within parentheses.

B. It would execute but would return no rows.

C. It would not execute because the same column has been used twice with the AND logical operator.

D. It would execute and return the desired result.

ANSWER: B

Explanation:

It would execute but would return no rows. Oracle permits multiple conditions on the same column connected with the `AND` logical operator, so the statement is syntactically valid and can be parsed and executed. However, the conditions shown require a single `CATEGORY_ID` value to be equal to both `12` and `13` simultaneously. Because one scalar column value cannot equal two distinct numeric values in the same row, no row can satisfy both predicates; adding the `SUPPLIER_ID = 102088` predicate does not change that result. To express the stated requirement, the two category alternatives must instead be joined with `OR`, with the supplier condition joined using `AND`; parentheses should be used around the category alternatives to make the intended grouping explicit. Oracle documents that conditions combined with `AND` are true only when each individual condition is true, which is why the displayed predicates produce an empty result set. See Oracle's SQL Conditions documentation and its discussion of WHERE-clause conditions.

QUESTION NO: 13

Examine these requirements:

1. Display book titles for books purchased before January 17, 2007 costing less than 500 or more than 1000.

2. Sort the titles by date of purchase, starting with the most recently purchased book.

Which two queries can be used?

A. SELECT book_title FROM books WHERE (price < 500 OR > 1000) AND (purchase date < ' 17-JAN-2007 ') ORDER BY purchase date DESC;

B. SELECT book_title FROM books WHERE (price IN (500, 1000)) AND (purchase date < ' 17-JAN-2007 ') ORDER BY purchase_date ASC;

C. SELECT book_title FROM books WHERE (price NOT BETWEEN 500 AND 1000) AND (purchase_date < ' 17-JAN-2007 ') ORDER BY purchase_date DESC;

D. SELECT book_title FROM books WHERE (price BETWEEN 500 AND 1000) AND (purchase_date < ' 17-JAN-2007 ') ORDER BY purchase_date;

ANSWER: C

Explanation:

`SELECT book_title FROM books WHERE (price NOT BETWEEN 500 AND 1000) AND (purchase_date < ' 17-JAN-2007 ' ) ORDER BY purchase_date DESC;` implements both filtering requirements and the requested sort order. In Oracle SQL, `BETWEEN` includes both boundary values; consequently, `NOT BETWEEN 500 AND 1000` selects

prices strictly below 500 or strictly above 1000. This is logically equivalent to writing `price < 500 OR price > 1000`, exactly matching the stated price criteria. The additional predicate on `purchase_date` limits the result to purchases occurring before January 17, 2007. Finally, `ORDER BY purchase_date DESC` orders rows from the greatest, or latest, purchase date to the least, so the most recently purchased qualifying book is displayed first. Oracle documents that the `DESC` keyword specifies descending order and that the range condition evaluates inclusively at its endpoints, making the negated range condition appropriate here. In production SQL, an ANSI date literal such as `DATE '2007-01-17'` is preferable to a character-date literal because it avoids dependence on session date-format settings. See Oracle's range condition documentation and SELECT and ORDER BY documentation.

QUESTION NO: 14

Which two statements are true about selecting related rows from two tables based on an Entity Relationship Diagram (ERD)? (Choose two.)

A. Implementing a relationship between two tables might require joining additional tables

B. Relating data from a table with data from the same table is implemented with a self join

C. Rows from unrelated tables cannot be joined

D. Every relationship between the two tables must be implemented in a join condition

E. An inner join relates rows within the same table

ANSWER: A B

Explanation:

Implementing a relationship between two tables might require joining additional tables because an ERD can model a many-to-many relationship through an associative, or junction, table. For example, if employees can work on many projects and each project can have many employees, the employee and project tables are related through a table such as EMPLOYEE_PROJECT. A query that needs attributes from both endpoint entities follows the foreign-key relationships through that additional table. Oracle SQL supports joining any number of tables by specifying the appropriate join predicates; see the Oracle Database SQL Language Reference for SELECT.

Relating data from a table with data from the same table is implemented with a self join. A self join uses two distinct aliases for separate logical instances of one table, allowing a row to be matched to another row in that same table. A common ERD example is a recursive employee-manager relationship: the EMPLOYEES table contains an employee identifier and a manager identifier that references another employee row. Aliases distinguish the employee instance from the manager instance, and the join condition matches the manager identifier to the manager row's employee identifier. Oracle documents self joins and table aliases in its joins documentation.

QUESTION NO: 15

View the Exhibits and examine the structure of the COSTS and PROMOTIONS tables.

You want to display PROD_IDS whose promotion cost is less than the highest cost PROD_ID in a promotion time interval.

Examine this SQL statements:

```
SELECT prod_id
  FROM costs
 WHERE promo_id IN
    (SELECT promo_id
       FROM promotions
      WHERE promo_cost < ALL
         (SELECT MAX(promo_cost)
            FROM promotions
           GROUP BY (promo_end_date - promo_begin_date)));
```

What will be the result?

Exhibit 1.

Table COSTS		
Name	Null?	Type
PROD_ID	NOT NULL	NUMBER
TIME_ID	NOT NULL	DATE
PROMO_ID	NOT NULL	NUMBER
CHANNEL_ID	NOT NULL	NUMBER
UNIT_COST	NOT NULL	NUMBER(10,2)
UNIT_PRICE	NOT NULL	NUMBER(10,2)

Exhibit 2.

Table PROMOTIONS		
Name	Null?	Type
PROMO_ID	NOT NULL	NUMBER(6)
PROMO_NAME	NOT NULL	VARCHAR2(30)
PROMO_SUBCATEGORY	NOT NULL	VARCHAR2(30)
PROMO_SUBCAEGORY_ID	NOT NULL	NUMBER
PROMO_CATEGORY	NOT NULL	VARCHAR2(30)
PROMO_CATEGORY_ID	NOT NULL	NUMBER
PROMO_COST	NOT NULL	NUMBER(10,2)
PROMO_BEGIN_DATE	NOT NULL	DATE
PROMO_END_DATE	NOT NULL	DATE

A. It gives an error because the GROUP BY clause is not valid

B. It executes successfully but does not give the required result

C. It executes successfully and gives the required result

D. It gives an error because the ALL keyword is not valid

ANSWER: C

Explanation:

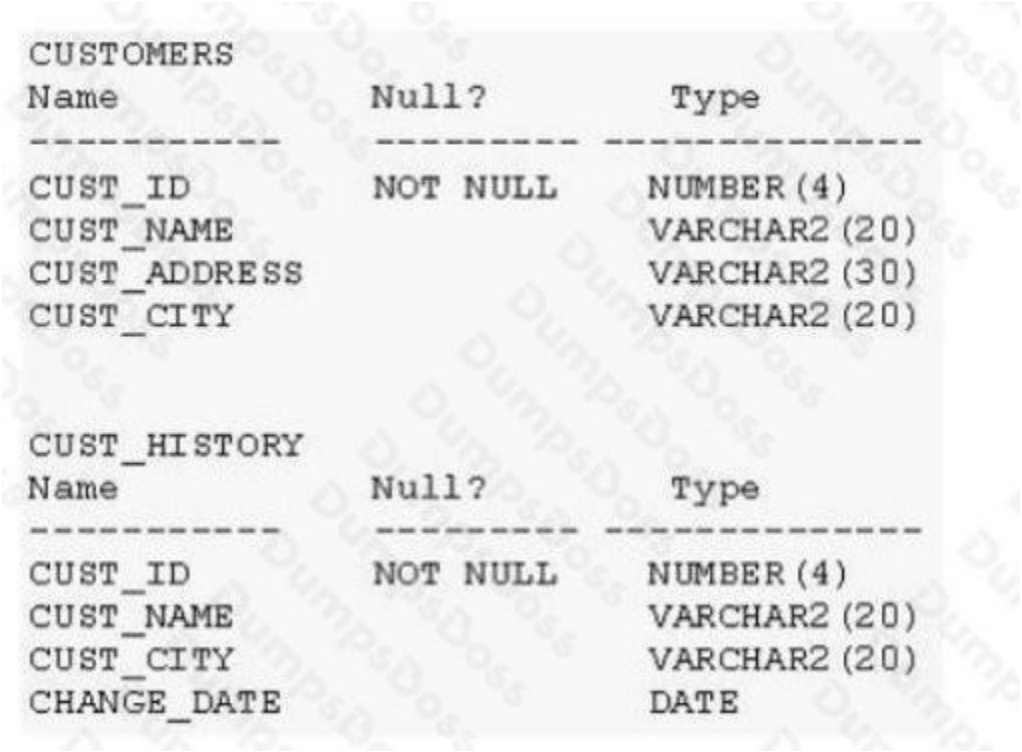
The statement executes successfully and gives the required result. The grouped subquery calculates a maximum promotion cost for each promotion time interval. Because the aggregate expression is calculated separately for every group identified by the promotion interval columns, the subquery can return multiple maximum-cost values, one per interval.

The outer condition compares each promotion cost with `ALL` of the values returned by that subquery. Oracle permits `ALL` with a multiple-row subquery. With a less-than comparison, `< ALL (...)` is true only when the outer promotion cost is lower than every maximum value returned for the grouped time intervals. Thus, the query returns the qualifying product identifiers according to the stated comparison requirement. The `GROUP BY` is valid because the selected expression in the subquery is an aggregate, and the interval columns define the groups used to derive each maximum.

Oracle documents quantified comparison conditions, including `ALL`, as valid conditions used with subqueries that return one or more values. Oracle also documents that `GROUP BY` divides rows into groups so aggregate functions, such as `MAX`, are evaluated for each group. See Oracle SQL Language Reference: Comparison Conditions and Oracle SQL Language Reference: SELECT.

QUESTION NO: 16

View the Exhibit and examine the structure of the CUSTOMERS and CUST_HISTORY tables.



```
CUSTOMERS
Name            Null?        Type
-----------     --------- --------------
CUST_ID         NOT NULL     NUMBER(4)
CUST_NAME                    VARCHAR2(20)
CUST_ADDRESS                 VARCHAR2(30)
CUST_CITY                    VARCHAR2(20)


CUST_HISTORY
Name            Null?        Type
-----------     --------- --------------
CUST_ID         NOT NULL     NUMBER(4)
CUST_NAME                    VARCHAR2(20)
CUST_CITY                    VARCHAR2(20)
CHANGE_DATE                  DATE
```

The CUSTOMERS table contains the current location of all currently active customers.

The CUST_HISTORY table stores historical details relating to any changes in the location of all current as well as previous customers who are no longer active with the company. You need to find those customers who have never changed their address.

Which SET operator would you use to get the required output?

A. INTERSECT

B. UNION ALL

C. MINUS

D. UNION

ANSWER: C

Explanation:

`MINUS` is the appropriate set operator because it returns the distinct rows produced by the first query that do not appear in the result of the second query. To identify active customers who have never changed address, the first query should return

the identifying customer values from `CUSTOMERS`, which represents all currently active customers. The second query should return the corresponding customer values from `CUST_HISTORY`, which records customers for whom a location change has historical data. Subtracting the historical customer set from the current customer set leaves only active customers with no address-change history.

For example, the comparison can be based on the customer identifier: `SELECT customer_id FROM customers MINUS SELECT customer_id FROM cust_history`. Both component queries must return the same number of columns, and corresponding expressions must have compatible data types. Oracle removes duplicate rows as part of set-operator processing, so each qualifying customer is returned once. This use of set subtraction directly models the requirement: begin with the current active population, then exclude every customer represented in the location-history population. Oracle documents that `MINUS` returns rows selected by the first query but not by the second query. See the Oracle Database SQL Language Reference: Set Operators and the Oracle Database SQL Language Reference: SELECT.

QUESTION NO: 17

You must display details of all users whose username contains the string 'ch_'. (Choose the best answer.)

Which query generates the required output?

A. SELECT * FROM users Where user_name LIKE '%ch_';

B. SELECT * FROM users
Where user_name LIKE '%ch_%'ESCAPE'%';

C. SELECT * FROM users
Where user_name LIKE 'ch_%' ESCAPE '_';

D. SELECT * FROM users
Where user_name LIKE '%ch_%' ESCAPE '\';

ANSWER: D

Explanation:

`SELECT * FROM users WHERE user_name LIKE '%ch\_%' ESCAPE '\';` correctly searches for usernames containing the literal character sequence `ch_` anywhere in the value. In an Oracle `LIKE` pattern, the percent sign (`%`) is a wildcard for any sequence of zero or more characters, so the leading and trailing percent signs permit text before and after the required sequence. An underscore (_) normally represents exactly one arbitrary character, rather than an actual underscore. The `ESCAPE '\'` clause explicitly declares the backslash as the escape character; consequently, `\_` in the pattern means a literal underscore. This preserves the intended meaning of the search: find every username that contains the exact string `ch_`, not merely `ch` followed by any single character. Oracle documents the `LIKE` condition, including the use of percent and underscore pattern-matching characters and the optional `ESCAPE` clause, at Oracle SQL Language Reference: Pattern-matching Conditions. General syntax and behavior for SQL conditions are also described in the Oracle SQL Language Reference: Conditions.

QUESTION NO: 18

Which two statements are true about single row functions?

A. CONCAT: can be used to combine any number of values

B. MOD: returns the quotient of a division operation

C. CEIL: can be used for positive and negative numbers

D. FLOOR: returns the smallest integer greater than or equal to a specified number

E. TRUNC: can be used with NUMBER and DATE values

ANSWER: C E

Explanation:

`CEIL` can be used for both positive and negative numeric values. Oracle defines `CEIL` as returning the smallest integer greater than or equal to its argument. This definition applies regardless of the number's sign. For example, applying it to a positive fractional value returns the next higher integer, while applying it to a negative fractional value returns the integer that is numerically greater (closer to positive infinity). It is therefore a numeric single-row function that produces one result for each input row.

`TRUNC` can be used with `NUMBER` and `DATE` values. Oracle supplies numeric and datetime forms of `TRUNC`. Numeric `TRUNC` truncates a number to a specified decimal precision, or to an integer when the precision is omitted. Datetime `TRUNC` truncates a date to the unit identified by its format model, such as a year, month, or day. In each case, it evaluates an expression for an individual row and returns a single value, which is the defining behavior of a single-row function. See Oracle's documentation for CEIL and TRUNC (date).

QUESTION NO: 19

Examine the description of the SALES1 table:

Name	Null	Type
SALES_ID	NOT NULL	NUMBER
STORE_ID	NOT NULL	NUMBER
ITEMS_ID		NUMBER
QUANTITY		NUMBER
SALES_DATE		DATE

SALES2 is a table with the same description as SALES1,

Some sales data is duplicated In both tables.

You want to display the rows from the SALES1 table which are not present in the SALIES2 table.

Which set operator generates the required output?

A. SUBTRACT

B. INTERSECT

C. UNION ALL

D. MINUS

E. UNION

ANSWER: D

Explanation:

`MINUS` is the Oracle set operator used to return the distinct rows produced by the first query that do not occur in the result of the second query. Therefore, a query such as `SELECT ... FROM sales1 MINUS SELECT ... FROM sales2` displays the SALES1 rows that are absent from SALES2, which exactly meets the requirement. The corresponding select lists must have the same number of expressions, with compatible data types in corresponding positions. As a set operator, `MINUS` also eliminates duplicate rows from its result, so each qualifying SALES1 row is returned once. Oracle Database documentation defines `MINUS` as returning all distinct rows selected by the first query but not the second query, and notes the compatibility requirements for component query select lists. See Oracle's documentation on UNION, INTERSECT, and MINUS operators and the SELECT statement reference.

QUESTION NO: 20

Examine these statements executed in a single Oracle session:

CREATE TABLE product (pcode NUMBER(2),pname VARCHAR2(20));

INSERT INTO product VALUES(1, ' pen ');

INSERT INTO product VALUES (2, ' pencil ');

INSERT INTO product VALUES(3, ' fountain pen ');

SAVEPOINT a;

UPDATE product SET pcode=10 WHERE pcode =1;

COMMIT;

DELETE FROM product WHERE pcode =2;

SAVEPOINT b;

UPDATE product SET pcode=30 WHERE pcode =3;

SAVEPOINT c;

DELETE FROM product WHERE pcode =10;

ROLLBACK TO SAVEPOINT b;

COMMIT;

Which three statements are true ?

A. The code for pen is 10.

B. There is no row containing fountain pen.

C. There is no row containing pencil.

D. The code for pen is 1.

E. The code for fountain pen is 3

F. There is no row containing pen

ANSWER: A C E

Explanation:

The final committed contents of `PRODUCT` include the `pen` row with `PCODE` 10 and the `fountain pen` row with `PCODE` 3; the `pencil` row is absent. The first `COMMIT` permanently saves the initial inserts and the update changing the code for `pen` from 1 to 10. A later transaction then deletes `pencil` and establishes savepoint `b`. Consequently, that deletion is earlier than savepoint `b` and remains in effect when the transaction is ultimately committed.

After savepoint `b`, the code for `fountain pen` is changed from 3 to 30, and the row for `pen` is deleted. `ROLLBACK TO SAVEPOINT b` reverses all changes made after that savepoint while retaining changes made before it in the same transaction. It therefore restores the deleted `pen` row and restores the original code of 3 for `fountain pen`. The final `COMMIT` saves this resulting state. Oracle documents that rolling back to a savepoint erases changes after the savepoint and preserves changes before it; a commit makes the transaction permanent. See Oracle SQL Language Reference: ROLLBACK and Oracle SQL Language Reference: COMMIT.

QUESTION NO: 21

BOOK_SEQ is an existing sequence in your schema.

Which two CREATE TABLE commands are valid?

A. CREATE TABLE bookings (bk_id NUMBER(4) NOT NULL PRIMARY KEY,start_date DATE NOT NULL,end_date DATE DEFAULT SYSDATE);

B. CREATE TABLE bookings (bk_id NUMBER(4) NOT NULL DEFAULT book_seq.CURRVAL,start_date DATE NOT NULL,end_date DATE DEFAULT SYSDATE);

C. CREATE TABLE bookings (bk_id NUMBER(4) DEFAULT book_seq.CURRVAL,start_date DATE DEFAULT SYSDATE,end_date DATE DEFAULT start date);

D. CREATE TABLE bookings (bk_id NUMBER(4),start_date DATE DEFAULT SYSDATE,end_date DATE DEFAULT (end_date > = start_date));

E. CREATE TABLE bookings (bk_id NUMBER(4) DEFAULT book_seq.NEXTVAL PRIMARY KEY,start_date DATE DEFAULT SYSDATE,end_date DATE DEFAULT SYSDATE NOT NULL);

ANSWER: A E

Explanation:

The command defining `bk_id NUMBER(4) NOT NULL PRIMARY KEY`, a required `start_date`, and `end_date DATE DEFAULT SYSDATE` is valid. Oracle permits an inline primary-key constraint and permits `SYSDATE` as a date-column default expression. A default is applied when an insert omits that column, while the `NOT NULL` and primary-key declarations enforce the required data-integrity rules.

The command defining `bk_id NUMBER(4) DEFAULT book_seq.NEXTVAL PRIMARY KEY` is also valid in Oracle Database 12c. Oracle allows a sequence's `NEXTVAL` pseudocolumn as a column default, so an inserted row that does not explicitly provide `bk_id` receives the next sequence value. Combining that default with an inline primary key is valid, and the `SYSDATE` defaults for the date columns are valid as well. Because the final date column is `NOT NULL`, its default ensures that omitted values satisfy the constraint.

Oracle documents column default expressions, including the permitted use of a sequence `NEXTVAL` pseudocolumn, in the CREATE TABLE reference. The behavior of sequence pseudocolumns is described in the Sequence Pseudocolumns documentation.

QUESTION NO: 22

Examine the structure proposed for the TRANSACTIONS table:

Name	Null?	Type
TRANS_ID	NOT NULL	NUMBER(6)
CUST_NAME	NOT NULL	VARCHAR2(20)
CUST_STATUS	NOT NULL	VARCHAR2
TRANS_DATE	NOT NULL	DATE
TRANS_VALIDITY		INTERVAL DAY TO SECOND
CUST_CREDIT_VALUE		NUMBER(10)

Which two statements are true regarding the storage of data in the above table structure? (Choose two.)

A. The CUST_CREDIT_VALUE column would allow storage of positive and negative integers.

B. The TRANS_VALIDITY column would allow storage of a time interval in days, hours, minutes, and seconds.

C. The CUST_STATUS column would allow storage of data up to the maximum VARCHAR2 size of 4,000 characters.

D. The TRANS_DATE column would allow storage of dates only in the dd-mon-yyyy format.

ANSWER: A B

Explanation:

The CUST_CREDIT_VALUE column would allow storage of positive and negative integers because its numeric definition has a scale of zero. Oracle NUMBER values can be signed, and a zero scale specifies that the stored value has no digits to the right of the decimal point. Subject to the declared precision, this permits both positive and negative whole-number credit values. Oracle documents NUMBER as a numeric datatype that supports precision and scale; scale determines the number of decimal digits represented. See Oracle Database SQL Language Reference: Numeric Datatypes.

The TRANS_VALIDITY column would allow storage of a time interval in days, hours, minutes, and seconds because an INTERVAL DAY TO SECOND datatype represents an elapsed duration using a day-leading field and time fields down to seconds. This datatype is intended for a span of time rather than a calendar date or a time-of-day value. Its declared day and fractional-second precisions control the permitted number of digits in those respective fields, while the value itself can retain the day, hour, minute, and second components. Oracle describes the syntax and storage semantics for this interval family in Datetime and Interval Datatypes.

QUESTION NO: 23

In the PROMOTIONS table, the PROMO_BEGIN_DATE column is of data type DATE and the default date format is DD-MON-RR.

Which two statements are true about expressions using PROMO_BEGIN_DATE contained a query? (Choose two.)

A. PROMO_BEGIN_DATE – 5 will return a date.

B. PROMO_BEGIN_DATE – SYSDATE will return a number.

C. TO_NUMBER(PROMO_BEGIN_DATE) – 5 will return a number.

D. TO_DATE(PROMO_BEGIN_DATE * 5) will return a date.

E. PROMO_BEGIN_DATE – SYSDATE will return an error.

ANSWER: A B

Explanation:

Oracle DATE arithmetic treats a numeric value as a number of calendar days. Therefore, **PROMO_BEGIN_DATE – 5 will return a date.** The value 5 represents five days, and subtracting it from the DATE value in PROMO_BEGIN_DATE produces another DATE value that is five days earlier. This arithmetic is independent of the session's DD-MON-RR default date format, because that format controls the display and implicit character conversion of dates, not the underlying DATE arithmetic.

PROMO_BEGIN_DATE – SYSDATE will return a number. Both PROMO_BEGIN_DATE and SYSDATE are DATE expressions. Subtracting one DATE from another returns a NUMBER representing the elapsed interval in days; fractional values represent portions of a day. For example, a difference of 1.5 represents one day and twelve hours. This makes DATE subtraction useful for calculating durations, ages, and the number of days between a stored promotion start date and the current database server date and time.

Oracle documents DATE arithmetic and the resulting data types in its SQL Language Reference: SQL Language Elements and SYSDATE.

QUESTION NO: 24

Which statements are true regarding primary and foreign key constraints and the effect they can have on table data?

A. A table can have only one primary key but multiple foreign keys.

B. It is possible for child rows that have a foreign key to remain in the child table at the time the parent row is deleted.

C. Primary key and foreign key constraints can be defined at both the column and table level.

D. Only the primary key can be defined the column and table level.

E. It is possible for child rows that have a foreign key to be deleted automatically from the child table at the time the parent row is deleted.

F. The foreign key columns and parent table primary key columns must have the same names.

G. A table can have only one primary key and one foreign key.

ANSWER: A B C E

Explanation:

A table can have only one primary key but multiple foreign keys. The single primary key may be composite, meaning that it can comprise more than one column, while separate foreign key constraints can establish relationships with multiple parent tables or keys. Primary key and foreign key constraints can be defined at both the column and table level. Oracle permits inline column-constraint syntax where applicable and out-of-line table-constraint syntax; table-level syntax is required when a key contains multiple columns.

It is possible for child rows that have a foreign key to remain in the child table at the time the parent row is deleted. This is supported when the foreign key uses `ON DELETE SET NULL`, provided that the child foreign-key columns permit null values. When the referenced parent row is deleted, Oracle sets those child key values to null, so the child rows remain and referential integrity is retained.

It is also possible for child rows that have a foreign key to be deleted automatically from the child table at the time the parent row is deleted. A foreign key defined with `ON DELETE CASCADE` directs Oracle to delete matching child rows when the referenced parent row is deleted. Oracle documents these referential actions as part of foreign-key constraint definition and enforcement. See the Oracle SQL Language Reference: Constraints and CREATE TABLE documentation.

QUESTION NO: 25

Examine the description of the EMPLOYEES table:

```
Name          Null?        Type
------------  ---------    -------------------------

EMPLOYEE_ID   NOT NULL          NUMBER(3)
FIRST_NAME                  VARCHAR2(15)
LAST_NAME     NOT NULL          VARCHAR2(15)
SALARY                      NUMBER(6,2)
```

Which two statements will run successfully?

A. SELECT ' The first_name is ' ' || first_name || ' ' FROM employees ;

B. SELECT ' The first_name is ' ' ' ||first_name || ' ' ' ' FROM employees ;

C. SELECT ' The first_name is ' ' ' ||first_name|| ' ' ' FROM employees ;

D. SELECT ' The first_name is ' || first_name|| ' ' FROM employees;

E. SELECT ' The first_name is \ ' ' || first_name || ' \ ' ' FROM employees;

ANSWER: B C

Explanation:

The statements `SELECT ' The first_name is ' ' ' ||first_name || ' ' ' ' FROM employees ;` and `SELECT ' The first_name is ' ' ' ||first_name|| ' ' ' FROM employees ;` use Oracle character-literal

quoting correctly. In Oracle SQL, a character literal is enclosed in single quotation marks. When a literal itself must contain a single-quotation-mark character, that quotation mark is represented by two consecutive single quotation marks within the literal. This lets the query build text that includes quotation marks around the value returned by `FIRST_NAME`.

Both successful statements also use the `||` concatenation operator correctly to combine the fixed text literal, the value from the `FIRST_NAME` column, and the trailing literal. For each row in `EMPLOYEES`, Oracle evaluates the concatenation expression and returns one character result. The placement of whitespace around `||` does not affect its operation, so the differing spacing in these two statements does not change their validity.

Oracle documents the syntax and escaping rules for text literals in the SQL Language Reference: Literals and documents `||` in the SQL Language Reference: Concatenation Operator.

QUESTION NO: 26

Examine the description of the EMPLOYEES table:

```
Name                      Null?      Type
--------------------      --------   -------------
EMPLOYEE_ID               NOT NULL   NUMBER(3)
FIRST_NAME                           VARCHAR2(15)
LAST_NAME                 NOT NULL   VARCHAR2(15)
SALARY                               NUMBER(6,2)
```

Which statement will execute successfully, returning distinct employees with non-null first names?

A. SELECT DISTINCT * FROM employees WHERE first_name IS NOT NULL;

B. SELECT first_name, DISTNCT last_name FROM employees WHERE first_name IS NOT NULL;

C. SELECT DISTINCT * FROM employees WHERE first_name <> NULL;

D. SELECT first_name, DISTINCT last_name FROM employees WHERE first_name <> NULL;

ANSWER: A

Explanation:

`SELECT DISTINCT * FROM employees WHERE first_name IS NOT NULL;` executes successfully because it uses valid Oracle SQL syntax to perform both required operations. The `WHERE first_name IS NOT NULL` predicate tests explicitly for the presence of a value in the `FIRST_NAME` column. In Oracle SQL, null represents an unknown or absent value, so null testing must use `IS NULL` or `IS NOT NULL`.

The `DISTINCT` keyword applies to the complete selected row because the select list is `*`. Therefore, Oracle returns only one occurrence of each distinct combination of all columns in the EMPLOYEES row, after filtering out rows whose first name is null. This meets the request to return distinct employee rows with non-null first names. Oracle documents `DISTINCT` as removing duplicate rows from a query result, and documents the `IS [NOT] NULL` condition specifically for null-value testing. See the Oracle Database 12c SELECT statement reference and the Oracle Database 12c NULL condition reference.

QUESTION NO: 27

Which three are true about system and object privileges

A. WITH GRANT OPTION can be used when granting an object privilege to both users and roles

B. WITH GRANT OPTION cannot be used when granting an object privilege to PUBLIC

C. Revoking a system privilege that was granted with the WITH ADMIN OPTION has a cascading effect.

D. Revoking an object privilege that was granted with the WITH GRANT OPTION clause has a cascading effect

E. Adding a primary key constraint to an existing table in another schema requires a system privilege

F. Adding a foreign key constraint pointing to a table in another schema requires the REFERENCES object privilege

ANSWER: B D F

Explanation:

WITH GRANT OPTION cannot be used when granting an object privilege to PUBLIC is correct because Oracle restricts this clause to object-privilege grants made to individual users; it cannot accompany a grant to PUBLIC. This preserves a traceable grant chain for object privileges that may be passed on. Oracle's SQL Language Reference explicitly documents the restrictions on use of WITH GRANT OPTION in the GRANT statement.

Revoking an object privilege that was granted with the WITH GRANT OPTION clause has a cascading effect is correct. When an object owner revokes a privilege from a user who received it with grant authority, Oracle also revokes dependent grants that the user made solely through that grant path. This ensures that delegated object access cannot remain after the underlying delegable privilege is removed.

Adding a foreign key constraint pointing to a table in another schema requires the REFERENCES object privilege is correct when access is granted at the object level. The user defining the foreign key must be authorized to reference the target table's columns, and REFERENCES is the object privilege designed for that purpose. See Oracle Database 12c SQL Language Reference: GRANT and Oracle Database 12c SQL Language Reference: Constraints.

QUESTION NO: 28

Evaluate the following SQL statement:

SELECT product_name || 'it's not available for order'

FROM product_information

WHERE product_status = 'obsolete';

You received the following error while executing the above query:

ERROR

ORA-01756: quoted string not properly terminated

What would you do to execute the query successfully?

A. Remove the single quotation marks enclosing the character literal string in the SELECT clause

B. Use the escape character to negate the single quotation mark within the literal character string in the SELECT clause

C. Enclose the character literal string in the SELECT clause within double quotation marks

D. Use the Oracle (q) operator and delimiter to allow the use of a single quotation mark within the literal character string in the SELECT clause

ANSWER: D

Explanation:

Use the Oracle (q) operator and delimiter to allow the use of a single quotation mark within the literal character string in the SELECT clause. Oracle's alternative quoting mechanism, commonly called q-quoting, lets a character literal contain apostrophes without requiring them to terminate the literal. The literal can be written with a chosen delimiter pair, such as `q'[it's not available for order]'`. The resulting query is `SELECT product_name || q'[it's not available for order]' FROM product_information WHERE product_status = 'obsolete';`. Oracle recognizes the opening delimiter after `q'` and reads until it encounters the corresponding closing delimiter followed by the final quote. This makes the apostrophe in `it's` ordinary text rather than a quote that ends the string prematurely. Delimiters may be paired characters such as brackets, parentheses, braces, or angle brackets, or an appropriate single nonblank character. This syntax is particularly helpful for literals containing apostrophes, quotation marks, or text that would otherwise

need repeated quote escaping. The comparison to `'obsolete'` remains a normal character literal and needs no change. Oracle documents both standard text literals and alternative quoting syntax in its SQL Language Reference: Oracle Database SQL Language Reference: Literals. See also Oracle's discussion of the concatenation operator.

QUESTION NO: 29

Which statement is true about TRUNCATE and DELETE?

A. You can never TRUNCATE a table if foreign key constraints will be violated.

B. For large tables, DELETE is faster than TRUNCATE.

C. For tables with multiple indexes and triggers, DELETE is faster than TRUNCATE.

D. You can DELETE rows from a table with referential integrity constraints.

ANSWER: D

Explanation:

You can DELETE rows from a table with referential integrity constraints. DELETE is a DML statement, so Oracle applies enabled referential-integrity rules to the rows affected by the statement. For example, rows can be deleted from a child table that contains a foreign key. A parent row can also be deleted when no child rows reference it. Where the foreign key was defined with an `ON DELETE CASCADE` action, deleting the parent row causes Oracle to delete the related child rows as part of the operation; with `ON DELETE SET NULL`, Oracle sets the applicable child foreign-key values to null. Because DELETE is fully governed by the active constraint definition and the data affected, the mere presence of referential-integrity constraints does not prevent a DELETE statement. DELETE also participates in normal transaction processing: its effects can be committed or rolled back, and relevant triggers can execute. Oracle documents foreign-key referential actions, including cascade and set-null behavior, in its SQL Language Reference. See Oracle Database SQL Language Reference: Referential Integrity Constraints and Oracle Database SQL Language Reference: DELETE.

QUESTION NO: 30

You execute these commands:

SQL> DEFINE hiredate = '01-APR -2011';

SQL> SELECT employee_id, first_name, salary FROM employees WHERE hire date > &hiredate AND manager_id >&mgr_id;

For which substitution variables will you be prompted?

A. none

B. &hiredate and &mgr_id

C. only &hiredate

D. only &mgr_id

ANSWER: D

Explanation:

Only `&mgr_id` prompts for a value. In SQL*Plus, the `DEFINE hiredate = '01-APR -2011'` command creates a SQL*Plus substitution variable named `hiredate`. Before sending the `SELECT` statement to the database, SQL*Plus scans the text and replaces substitution-variable references. Because `hiredate` is already defined, SQL*Plus substitutes its stored value when it encounters `&hiredate`; no interactive prompt is required for that reference.

The statement also contains `&mgr_id`, but no prior `DEFINE mgr_id` command or other value assignment is shown. SQL*Plus therefore requests a value for that variable, displaying an `Enter value for mgr_id:` prompt. This behavior occurs during SQL*Plus command substitution, before the resulting SQL text is passed to Oracle Database for SQL parsing and execution. The key distinction is that a defined substitution variable is reused automatically, whereas an undefined one is prompted for on first reference. Oracle documents substitution variables and the `DEFINE` command in the SQL*Plus User's Guide: Using Substitution Variables and the DEFINE command reference.

QUESTION NO: 31

Table HR.EMPLOYEES contains a row where the EMPLOYEES _ID is 109.

User ALICE has no privileges to access HR.EMPLOYEES.

User ALICE starts a session.

User HR. starts a session and successfully executes these statements:

GRANT DELETE ON employees TO alice;

UPDATE employees SET salary = 24000 WHERE employee_id = 109;

In her existing session ALICE then executes:

DELETE FROM hr.employees WHERE employee_id = 109;

What is the result?

A. The DELETE command will immediately delete the row.

B. The DELETE command will wait for HR ' s transaction to end then delete the row.

C. The delete command will wait for HR ' s transaction to end then return an error.

D. The delete command will immediately return an error.

ANSWER: B

Explanation:

The DELETE command will wait for HR's transaction to end then delete the row. The GRANT statement gives ALICE the object privilege required to delete rows from HR.EMPLOYEES, and an object privilege grant takes effect without ALICE needing to establish a new database session. Therefore, her existing session can execute the DELETE statement.

However, HR has already updated the row with employee_id 109 and has not yet completed that transaction. An UPDATE obtains a row lock on the affected row. When ALICE attempts to delete that same row, Oracle must wait for the transaction holding the conflicting row lock to commit or roll back. Once HR's transaction ends, the lock is released. Because the DELETE predicate is still satisfied—the employee_id remains 109—Oracle can acquire the required lock and delete the row. This is Oracle's normal transaction and row-locking behavior for conflicting DML operations. Oracle documents DELETE syntax and its required object privilege in the DELETE statement reference; its locking behavior is described in the Oracle Database data concurrency documentation.

QUESTION NO: 32

Which two statements are true regarding constraints? (Choose two)

A. A constraint is enforced only for an INSERT operation on a table.

B. A foreign key cannot contain NULL values.

C. A column with the UNIQUE constraint can store NULLS.

D. You can have more than one column in a table as part of a primary key.

ANSWER: C D

Explanation:

A column with the UNIQUE constraint can store NULLS. In Oracle Database, a unique constraint requires that non-null values in the constrained column or column combination be distinct, but it does not impose a NOT NULL requirement. Therefore, a column may be defined with a unique constraint and still accept null values unless NOT NULL is also declared. Oracle's SQL Language Reference explicitly describes a unique constraint as allowing nulls while ensuring uniqueness for non-null key values.

You can have more than one column in a table as part of a primary key. This is a composite primary key: one primary-key constraint can be defined over multiple columns, and the combination of their values must uniquely identify each row. Every column participating in a primary key is implicitly NOT NULL, ensuring that each composite key is complete and can reliably identify a row. Composite keys are useful when no single attribute is sufficient to provide row uniqueness. See the Oracle documentation on constraint definitions and primary and unique key constraints.

QUESTION NO: 33

View the Exhibit and examine the data in EMP and DEPT tables.

DEPT

DEPTNO	DEPTNAME
10	IT
20	HR

EMP

EMPNO	ENAME	DEPTNO
1	KING	10
2	HARI	20

In the DEPT table, DEPTNO is the PRIMARY KEY.

In the EMP table, EMPNO is the PRIMARY KEY and DEPTNO is the FOREIGN KEY referencing the DEPTNO column in the DEPT table.

What would be the outcome of the following statements executed in the given sequence?

DROP TABLE emp;

FLASHBACK TABLE emp TO BEFORE DROP;

INSERT INTO emp VALUES (2, 'SCOTT', 10); INSERT INTO emp VALUES (3, 'KING', 55);

A. Both the INSERT statements would fail because the constraints are automatically retrieved when the table is flashed back.

B. Both the INSERT statements would succeed because none of the constraints on the table are automatically retrieved when the table is flashed back.

C. Only the first INSERT statement would succeed because all constraints except the primary key constraint are automatically retrieved after a table is flashed back.

D. Only the SECOND INSERT statement would succeed because all the constraints except referential integrity constraints that reference other tables are retrieved automatically after the table is flashed back.

ANSWER: D

Explanation:

Only the SECOND INSERT statement would succeed because all the constraints except referential integrity constraints that reference other tables are retrieved automatically after the table is flashed back. A dropped table is moved to the recycle bin, and `FLASHBACK TABLE emp TO BEFORE DROP` restores the table and its eligible dependent objects. The EMP primary-key constraint is restored, so the attempted insertion using employee number 2 fails because that employee number is already present in the restored EMP data shown in the exhibit.

Oracle does not automatically restore referential-integrity constraints that reference another table when a table is recovered with Flashback Drop. Therefore, the foreign key from `EMP.DEPTNO` to `DEPT.DEPTNO` is absent after EMP is flashed back. The row for KING has a new employee number, 3, and can be inserted even though department number 55 does not exist in DEPT. This behavior is important operationally: after recovering a table from the recycle bin, foreign-key relationships to other tables may need to be re-created explicitly to resume cross-table referential validation. Oracle documents the Flashback Drop behavior and its limitations in the FLASHBACK TABLE SQL Reference and the Oracle Database Administrator's Guide.

QUESTION NO: 34

Which two statements are true regarding the WHERE and HAVING clauses in a SELECT statement? (Choose two.)

A. The WHERE and HAVING clauses can be used in the same statement only if they are applied to different columns in the table.

B. The aggregate functions and columns used in the HAVING clause must be specified in the SELECT list of the query.

C. The WHERE clause can be used to exclude rows after dividing them into groups.

D. The HAVING clause can be used with aggregate functions in subqueries.

E. The WHERE clause can be used to exclude rows before dividing them into groups.

ANSWER: D E

Explanation:

The WHERE clause can be used to exclude rows before dividing them into groups. Oracle evaluates row-level filtering before it forms groups for a query containing GROUP BY. Consequently, a WHERE predicate limits the input rows that participate in aggregation, which is useful both for expressing the intended result and for avoiding aggregation of rows that cannot qualify. For example, a condition restricting rows to a particular department or date range belongs in WHERE when it is to be applied before calculating group totals.

The HAVING clause can be used with aggregate functions in subqueries. A subquery is itself a SELECT statement and can perform grouping and aggregate calculations. Within that subquery, HAVING filters the groups based on aggregate results, such as retaining only departments whose calculated average salary meets a threshold. The outer query can then use the subquery result in predicates such as IN, EXISTS, or a comparison. Oracle documents that HAVING restricts groups returned by a query and that aggregate functions return a single result per group. See the Oracle Database 12c SELECT statement reference and the Oracle aggregate functions reference.

QUESTION NO: 35

Evaluate the following ALTER TABLE statement:

ALTER TABLE orders

SET UNUSED (order_date);

Which statement is true?

A. After executing the ALTER TABLE command, you can add a new column called ORDER_DATE to the ORDERS table.

B. The ORDER_DATE column must be empty for the ALTER TABLE command to execute successfully.

C. ROLLBACK can be used to get back the ORDER_DATE column in the ORDERS table.

D. The DESCRIBE command would still display the ORDER_DATE column.

ANSWER: A

Explanation:

After executing the ALTER TABLE command, you can add a new column called ORDER_DATE to the ORDERS table. Setting a column unused is a logical drop operation: Oracle makes the column inaccessible to SQL statements and removes it from the table definition presented to users and tools. The column's data is not immediately physically removed from the table segments, which avoids the potentially lengthy work associated with dropping a populated column outright. Because an unused column is no longer part of the accessible table definition, its name can be reused when a new column is added. Thus, a later statement such as `ALTER TABLE orders ADD (order_date DATE)` is permitted, creating a new accessible ORDER_DATE column independent of the former unused column. The unused column remains internally present until it is permanently removed with `ALTER TABLE ... DROP UNUSED COLUMNS`; this later operation reclaims the relevant storage and cannot be rolled back as ordinary transactional DML. Oracle documents SET UNUSED as an alternative to dropping columns when immediate physical removal is undesirable and explicitly notes that the name of an unused column can be reused. See the Oracle Database SQL Language Reference: ALTER TABLE and the Oracle Database Administrator's Guide: Managing Tables.

QUESTION NO: 36

View and Exhibit and examine the structure and data in the INVOICE table.

```
INVOICE

Name          Null?     Type
------------ -------- --------------
INV_NO        NOT NULL  NUMBER
INV_DATE                DATE
CUST_NAME     NOT NULL  VARCHAR2 (20)
CUST_CAT                CHAR(1)
INV_AMT                 NUMBER(8,2)


INV_NO   INV_DATE     CUST_NAME    CUST_CAT   INV_AMT
-------  ----------- ----------- ---------  ----------
101      15-FEB-08    JAMES        1         255982.55
102      18-MAR-08    SMITH        2         100000.00
```

Which two statements are true regarding data type conversion in query expressions? (Choose two.)

A. inv_date = '15-february-2008' :uses implicit conversion

B. inv_amt = '0255982' : requires explicit conversion

C. inv_date > '01-02-2008' : uses implicit conversion

D. CONCAT(inv_amt, inv_date) : requires explicit conversion

E. inv_no BETWEEN '101' AND '110' : uses implicit conversion

ANSWER: A E

Explanation:

`inv_date = '15-february-2008'` uses implicit conversion because `inv_date` is a DATE column while the comparison value is a character literal. During evaluation, Oracle converts the character value to a DATE using the session's date-language and default date-format settings as needed for the comparison. The supplied value includes an unambiguous day, spelled-out month, and four-digit year, so it can be interpreted as a date value without writing an explicit `TO_DATE` call. In production SQL, explicitly using `TO_DATE` with a format model remains preferable because it avoids dependence on session NLS settings.

`inv_no BETWEEN '101' AND '110'` also uses implicit conversion. Since `inv_no` is numeric, Oracle must compare numeric values for the `BETWEEN` predicate. It therefore implicitly converts the character literals `'101'` and `'110'` to NUMBER values before applying the inclusive range test. Oracle documents implicit conversion behavior and recommends explicit conversions when predictable behavior across sessions is important. See Oracle Database SQL Language Reference: Data Types and Oracle Database SQL Language Reference: TO_DATE.

QUESTION NO: 37

View the Exhibit and examine the structure of the ORDER_ITEMS table.

ORDER_ITEMS

ORDER_ID	LINE_ITEM_ID	PRODUCT_ID	UNIT_PRICE	QUANTITY
2355	4	2322	19	188
2355	5	2323	17	190
2355	9	2359	226.6	204
2355	1	2289	46	200
2356	5	2308	58	47
2356	6	2311	95	51
2356	1	2264	199.1	38
2356	2	2274	148.5	34
2356	3	2293	98	40
2356	4	2299	72	44
2357	2	2245	462	26
2357	3	2252	788.7	26
2357	4	2257	371.8	29
2357	5	2262	95	29

You must select the ORDER_ID of the order that has the highest total value among all the orders in the ORDER_ITEMS table.

Which query would produce the desired result?

A. SELECT order_id
FROM order_items
GROUP BY order_id
HAVING SUM(unit_price*quantity) = (SELECT MAX(SUM(unit_price*quantity)) FROM order_items GROUP BY order_id);

B. SELECT order_id
FROM order_items
WHERE(unit_price*quantity) = (SELECT MAX(unit_price*quantity)
FROM order_items)
GROUP BY order_id;

C. SELECT order_id
FROM order_items
WHERE(unit_price*quantity) = MAX(unit_price*quantity) GROUP BY order_id;

D. SELECT order_id
FROM order_items
WHERE (unit_price*quantity) = (SELECT MAX(unit_price*quantity)
FROM order_items
GROUP BY order_id)

ANSWER: A

Explanation:

`SELECT order_id FROM order_items GROUP BY order_id HAVING SUM(unit_price*quantity) = (SELECT MAX(SUM(unit_price*quantity)) FROM order_items GROUP BY order_id);` correctly calculates and compares order totals at the appropriate aggregation level. The outer query groups all item rows by `order_id`, so `SUM(unit_price * quantity)` produces the complete monetary value for each order rather than evaluating individual order-item rows.

The subquery uses the same grouping logic to calculate a total for every order, then applies `MAX` to obtain the greatest of those order totals. Oracle permits nesting aggregate functions in this manner when the inner aggregate operates over groups defined by `GROUP BY`. The `HAVING` clause then retains the grouped order whose calculated total equals that maximum value. If multiple orders are tied for the greatest total, this query appropriately returns every tied `order_id`.

This follows Oracle's documented aggregate-function behavior: aggregate expressions summarize rows into group-level results, and `HAVING` filters those results after grouping. See Oracle's Aggregate Functions documentation and its SELECT statement documentation.

QUESTION NO: 38

Which normal form is a table in if it has no multi-valued attributes and no partial dependencies?

A. second normal form

B. first normal form

C. third normal form

D. fourth normal form

ANSWER: A

Explanation:

second normal form is correct. A relation reaches second normal form only after satisfying first normal form, which requires attributes to contain single, atomic values rather than repeating groups or multivalued values. It must also have no partial dependency: every non-key attribute must depend on the complete candidate key, rather than on only one part of a composite key. Thus, the two stated properties describe the defining conditions for second normal form. This normalization step improves data design by ensuring that facts dependent on part of a composite identifier are separated into an appropriate relation, reducing duplicated data and helping preserve consistency when values are inserted, updated, or deleted. In practical Oracle database design, identifying keys and the functional dependencies among attributes before creating tables helps determine whether a structure satisfies this form. Oracle's Database Concepts documentation provides foundational relational database concepts, and the Oracle SQL Developer Data Modeler documentation covers data-modeling practices used to design normalized relational structures.

QUESTION NO: 39

Examine the description of the ORDERS table:

```
ORDER_ID    ORDER_DATE
--------    ----------
       1    <null>
       2    <null>
       3    01-JAN-2019
       4    01-FEB-2019
       5    01-MAR-2019
```

Examine the description of the INVOICES table:

```
INVOICE_ID  ORDER_ID  ORDER_DATE
----------  --------  ----------
         1         1  <null>
         2         2  01-JAN-2019
         3         3  <null>
         4         4  01-FEB-2019
         5         5  <null>
```

Examine this query:

```
SELECT order_id, order_date FROM orders
 MINUS
SELECT order_id, order_date FROM invoices
```

Which three rows will it return? (Choose three.)

A. 5 01-MAR-2019

B. 3

C. 1

D. 4 01-FEB-2019

E. 2

F. 5

G. 3 01-JAN-2019

ANSWER: A E G

Explanation:

The query returns **5 01-MAR-2019**, **2 <null>**, and **3 01-JAN-2019**. The join preserves the qualifying ORDERS rows according to the query's outer-join condition, then obtains the related invoice date when an INVOICES row satisfies the join predicate. Thus, order 5 is returned with its matching 01-MAR-2019 invoice date, and order 3 is returned with its matching 01-JAN-2019 invoice date. Order 2 also remains in the result set even though it has no matching invoice row meeting the join condition; Oracle therefore supplies a null value for the columns projected from INVOICES. This is the defining behavior of an outer join: rows retained from the preserved table appear even when no matching row exists in the other table, with nulls in the unavailable joined-column values. The query's filtering and join criteria restrict the output to these three retained or matched rows. Oracle documents outer-join result handling in its SQL Language Reference: Joins and documents null behavior in Null Conditions.

QUESTION NO: 40

Which two statements are true regarding subqueries? (Choose two.)

A. A subquery can appear on either side of a comparison operator.

B. Only two subqueries can be placed at one level.

C. A subquery can retrieve zero or more rows.

D. A subquery can be used only in SQL query statements.

E. There is no limit on the number of subquery levels in the WHERE clause of a SELECT statement.

ANSWER: A C

Explanation:

A subquery can appear on either side of a comparison operator because Oracle permits a single-row subquery wherever a single-value expression is valid. For example, a query may compare a column to the result returned by a subquery, or compare a subquery result to a literal or another expression. The subquery must return a value compatible with the comparison and, when used with a single-row comparison operator such as =, <, or >, it must return no more than one row. This makes subqueries useful for dynamically deriving the value used in a predicate rather than hard-coding it.

A subquery can retrieve zero or more rows. The number of rows depends on the data and on the operator or construct that consumes the result. A single-row subquery may return no rows, in which case its value is treated as null; a multiple-row subquery can return any number of rows and is commonly used with operators such as `IN`, `ANY`, and `ALL`. Oracle's documentation describes subqueries as queries nested within another SQL statement and distinguishes single-row and multiple-row subqueries by their possible result sets. See the Oracle Database 12c SQL Language Reference: Subqueries and the Oracle documentation for comparison conditions.

QUESTION NO: 41

Examine the description of the MEMBERS table:

```
Name                 Null?                Type
----------------     -----------------    ---------------
MEMBER_ID            NOT NULL             VARCHAR2 (6)
FIRST_NAME                                VARCHAR2 (50)
LAST_NAME            NOT NULL             VARCHAR2 (50)
ADDRESS                                   VARCHAR2 (50)
CITY                                      VARCHAR2 (25)
```

Examine the partial query:

SELECT city, last_name LNAME FROM members …;

You want to display all cities that contain the string AN. The cities must be returned in ascending order, with the last names further sorted in descending order. Which two clauses must you add to the query? (Choose two.)

A. ORDER BY last_name DESC, city ASC

B. WHERE city IN ('%AN%')

C. ORDER BY 1, LNAME DESC

D. ORDER BY 1, 2

E. WHERE city = '%AN%'

F. WHERE city LIKE '%AN%'

ANSWER: C F

Explanation:

`WHERE city LIKE '%AN%'` correctly restricts the rows to cities whose character value contains `AN` anywhere in the city name. In an Oracle `LIKE` condition, the percent sign (`%`) is a wildcard representing zero or more characters. Placing it on both sides of `AN` therefore performs a contains-style pattern match, rather than requiring the city value to equal only the supplied characters. Oracle documents `LIKE` as the appropriate condition for pattern matching with wildcard characters: Oracle SQL Language Reference: Pattern-Matching Conditions.

`ORDER BY 1, LNAME DESC` supplies the requested sort sequence. In an `ORDER BY` clause, `1` denotes the first item in the select list, which is `city`. Because no direction is specified for that expression, Oracle sorts cities in ascending order by default. `LNAME DESC` then sorts rows sharing the same city by the selected `last_name` alias in descending order. Oracle permits select-list aliases and positional notation in `ORDER BY`, as described in the Oracle SELECT statement reference.

QUESTION NO: 42

The BOOKS_TRANSACTIONS table exists in your database.

SQL>SELECT * FROM books_transactions ORDER BY 3;

What is the outcome on execution?

A. The execution fails unless the numeral 3 in the ORDER BY clause is replaced by a column name.

B. Rows are displayed in the order that they are stored in the table only for the three rows with the lowest values in the key column.

C. Rows are displayed in the order that they are stored in the table only for the first three rows.

D. Rows are displayed sorted in ascending order of the values in the third column in the table.

ANSWER: D

Explanation:

Rows are displayed sorted in ascending order of the values in the third column in the table. Oracle permits an `ORDER BY` item to be specified by its ordinal position in the select list. In this statement, `SELECT *` expands to the table's returned columns, and `ORDER BY 3` identifies the third selected column as the sort key. Because no direction is specified, Oracle applies the default direction, `ASC`, so values are ordered from low to high according to the normal comparison rules for that column's data type. The numeric literal is therefore not treated as a request to return three rows, nor does it refer to a physical storage order. It is a positional reference to the third expression in the query result. Positional ordering is valid SQL syntax in Oracle and can be useful for short ad hoc queries, although explicitly naming the selected column is often clearer and less susceptible to changes in the select list. Oracle documents that an `ORDER BY` clause can use a position to indicate an item in the select list, and that ascending order is the default when neither `ASC` nor `DESC` is supplied. See the Oracle Database SQL Language Reference for SELECT and the ORDER BY clause documentation.

QUESTION NO: 43

Which two statements are true regarding non equijoins?

A. The ON clause can be used.

B. The USING clause can be used.

C. The SQL:1999 compliant ANSI join syntax must be used.

D. Table aliases must be used.

E. The Oracle join syntax can be used.

ANSWER: A E

Explanation:

A non-equijoin matches rows using a comparison condition other than equality, such as a range test with `BETWEEN`, `<`, or `>`. The ON clause can be used because Oracle's ANSI join syntax permits the join condition to be expressed in an `ON` clause. That condition can contain the comparison predicates required to match a value from one table to a range or related value in another table. For example, an employee salary can be matched to a salary grade when the salary falls between the grade's low and high values.

The Oracle join syntax can be used because Oracle also supports its traditional join notation, in which tables are listed in the `FROM` clause and the join predicate is specified in the `WHERE` clause. A non-equality comparison in that predicate produces a non-equijoin, so this syntax remains valid for range-based table matching. Oracle documentation describes both the `join_clause`, including `ON` conditions, and traditional Oracle join syntax. See the Oracle Database 12c SELECT statement reference and the Oracle Database 12c joins documentation.

QUESTION NO: 44

Which three statements are true regarding the SQL WHERE and HAVING clauses?

A. The HAVING clause conditions can have aggregating functions.

B. The HAVING clause conditions can use aliases for the columns.

C. The WHERE and HAVING clauses cannot be used together in a SQL statement.

D. The WHERE clause is used to exclude rows before grouping data.

E. The HAVING clause is used to exclude one or more aggregated results after grouping data.

ANSWER: A D E

Explanation:

The HAVING clause conditions can have aggregating functions because HAVING applies to groups produced by a `GROUP BY` operation. It therefore supports conditions based on group-level values, such as retaining only departments whose `AVG(salary)` exceeds a specified value or whose `COUNT(*)` meets a threshold. This makes HAVING the appropriate mechanism for filtering calculated aggregate results.

The WHERE clause is used to exclude rows before grouping data. Oracle evaluates the row-filtering predicate before grouping and aggregate calculation, so only rows that satisfy the WHERE condition participate in the resulting groups and in aggregate functions such as `SUM`, `COUNT`, and `AVG`. Applying row criteria at this stage also expresses the intended query logic clearly.

The HAVING clause is used to exclude one or more aggregated results after grouping data. After Oracle forms the groups and derives aggregate values, HAVING can retain the groups that meet the stated aggregate condition. A single query can use WHERE to limit source rows and HAVING to limit the resulting groups, with each clause serving its appropriate point in query processing. Oracle documents these roles in its SELECT statement reference and aggregate functions reference.

QUESTION NO: 45

Examine the description of the ORDERS table:

Which three statements execute successfully?

A. SELECT * FROM ordersUNION ALLSELECT * FROM invoices ORDER BY order_id;

B. SELECE order _id, order _ date FRON ordersLNTERSECTSELECT invoice_ id, invoice_ id, order_ date FROM orders

C. SELECT order_ id, invoice_ data order_ date FROM ordersMINUSSELECT invoice_ id, invoice_ data FROM invoices ORDER BY invoice_ id;

D. SELECT * FROM orders ORDER BY order_ idINTERSEOTSELECT * FROM invoices ORDER BY invoice_ id;

E. SELECT order_id, order_date FROM ordersUNION ALLSELECT invoice_id, invoice_date FROM invoices ORDER BY order_id;

F. SELECT * FROM ordersMINUSSELECT * FROM INVOICES ORDER BY 1

G. SELECT * FROM orders ORDER BY order_ idUNIONSELECT * FROM invoices;

ANSWER: A E F

Explanation:

`SELECT * FROM orders UNION ALL SELECT * FROM invoices ORDER BY order_id;` executes because the corresponding columns returned by `ORDERS` and `INVOICES` have compatible datatypes. A compound query may have one `ORDER BY` clause at its end, and `order_id` is the output column name inherited from the first component query. `UNION ALL` returns all qualifying rows, including duplicate rows.

`SELECT order_id, order_date FROM orders UNION ALL SELECT invoice_id, invoice_date FROM invoices ORDER BY order_id;` also returns two compatible expressions from each query: an identifier and a date. The final ordering is applied to the combined result, using the first query's output-column name.

`SELECT * FROM orders MINUS SELECT * FROM invoices ORDER BY 1;` executes when the tables' corresponding columns are datatype-compatible. `MINUS` produces the distinct rows returned by the first query that are absent from the second query. `ORDER BY 1` is valid at the end of a compound query and sorts by the first column in its select list. Oracle documents the requirements for compound queries and the placement of their final `ORDER BY` clause in the Oracle Database SQL Language Reference: Set Operators and SELECT statement documentation.

QUESTION NO: 46

Examine the command to create the BOOKS table.

```
SQL>CREATE TABLE books
   (book_id        CHAR(6) PRIMARY KEY,
    title          VARCHAR2(100) NOT NULL,
    publisher_id   VARCHAR2(4),
    author_id      VARCHAR2(50));
```

The BOOK_ID value 101 does not exist in the table.

Examine the SQL statement:

```
SQL> INSERT INTO books(BOOK_ID, TITLE, AUTHOR_ID)
    VALUES ('101', 'LEARNING SQL', 'Tim Jones');
```

Which statement is true?

A. It executes successfully and the row is inserted with a NULL PUBLISHER_ID.

B. It executes successfully only if NULL is explicitly specified in the INSERT statement.

C. It executes successfully only if the PUBLISHER_ID column name is added to the columns list in the INSERT statement.

D. It executes successfully only if the PUBLISHER_ID column name is added to the columns list and NULL is explicitly specified in the INSERT statement.

ANSWER: A

Explanation:

It executes successfully and the row is inserted with a NULL PUBLISHER_ID. The INSERT statement supplies a value for the required BOOK_ID and omits PUBLISHER_ID from its target column list. For a column omitted from an INSERT, Oracle stores the column default when one is defined; otherwise, it stores NULL. Therefore, PUBLISHER_ID receives NULL.

A foreign key constraint does not require a non-NULL value by itself. Oracle verifies a foreign-key relationship when the child key contains a non-NULL value, but a NULL foreign-key value is permitted unless the column also has a NOT NULL constraint. Consequently, the fact that there may be no matching publisher key is not relevant to this insert because the inserted PUBLISHER_ID is NULL. BOOK_ID value 101 is new, so it can be inserted as the table's primary-key value. Oracle documents that columns not specified in an INSERT receive their default values or NULL, and that composite or single-column foreign keys can contain null values. See the Oracle INSERT statement documentation and the Oracle integrity-constraint documentation.

QUESTION NO: 47

Examine the command:

```
SQL> ALTER TABLE books_transactions
     ADD CONSTRAINT fk_book_id FOREIGN KEY (book_id)
     REFERENCES books (book_id) ON DELETE CASCADE;
```

What does ON DELETE CASCADE imply?

A. When the BOOKS table is dropped, the BOOK_TRANSACTIONS table is dropped.

B. When the BOOKS table is dropped, all the rows in the BOOK_TRANSACTIONS table are deleted but the table structure is retained.

C. When a row in the BOOKS table is deleted, the rows in the BOOK_TRANSACTIONS table whose BOOK_ID matches that of the deleted row in the BOOKS table are also deleted.

D. When a value in the BOOKS.BOOK_ID column is deleted, the corresponding value is updated in the BOOKS_TRANSACTIONS.BOOK_ID column.

ANSWER: C

Explanation:

`ON DELETE CASCADE` is a referential action specified as part of a foreign-key constraint. It applies when a parent row is deleted, not when the parent table itself is dropped. In this relationship, `BOOKS` is the parent table and `BOOK_TRANSACTIONS` is the child table because the child table contains a foreign key that references the parent table's `BOOK_ID` key. When Oracle deletes a particular row from `BOOKS`, it automatically deletes every child row in `BOOK_TRANSACTIONS` whose foreign-key value equals the deleted parent row's `BOOK_ID`. This prevents child rows from remaining that would refer to a parent book that no longer exists, thereby preserving referential integrity. The cascade is performed as part of the delete operation; no separate delete statement against the child table is required. Oracle documents `CASCADE` as the clause that causes dependent foreign-key values to be deleted when the referenced key value is deleted. See the Oracle Database SQL Language Reference: constraint clauses and the CREATE TABLE documentation.

QUESTION NO: 48

which is true about the round,truncate and mod functions>?

A. ROUND(MOD(25,3),-1) IS INVALID

B. ROUND(MOD(25,3),-1) AND TRUNC(MOD(25,3),-1) ARE BOTH VALID AND GIVE THE SAME RESULT.

C. ROUND(MOD(25,3),-1) AND TRUNC(MOD(25,3),-1) ARE BOTH VALID AND GIVE THE DIFFERENT RESULTS.

D. TRUNC(MOD(25,3),-1) IS INVALID.

ANSWER: B

Explanation:

ROUND(MOD(25,3),-1) AND TRUNC(MOD(25,3),-1) ARE BOTH VALID AND GIVE THE SAME RESULT. is correct. Oracle's `MOD` function returns the remainder after dividing its first numeric argument by its second numeric argument. Therefore, `MOD(25,3)` evaluates to `1`, because 25 divided by 3 leaves a remainder of 1. Both `ROUND` and numeric `TRUNC` accept an optional second argument that specifies the rounding or truncation position. A negative value specifies a position to the left of the decimal point. With `-1`, Oracle processes the value at the tens position. Applying `ROUND(1,-1)` produces `0`, since 1 rounds to zero tens. Applying `TRUNC(1,-1)` also produces `0`, because truncating 1 at the tens position removes the units digit. Consequently, both complete expressions are syntactically valid numeric function calls and return the same numeric result, `0`. Oracle documents the numeric behavior and arguments for `ROUND` at ROUND, `TRUNC` at TRUNC, and `MOD` at MOD.

QUESTION NO: 49

Which statement is true about TRUNCATE and DELETE?

A. For large tables TRUNCATE is faster than DELETE.

B. For tables with multiple indexes and triggers is faster than TRUNCATE.

C. You can never TRUNCATE a table if foreign key constraints will be violated.

D. You can never tows from a table if foreign key constraints will be violated.

ANSWER: A

Explanation:

For large tables TRUNCATE is faster than DELETE. `TRUNCATE TABLE` is a DDL operation designed to remove every row from a table efficiently. Rather than processing and recording the deletion of each individual row, Oracle deallocates the table's data storage (apart from any storage retained by the command's storage clause) and records the operation as a table-level change. This makes the amount of work largely independent of the number of rows in the table, which is especially beneficial when the table is large. Oracle documentation specifically describes truncation as a fast way to remove all rows and notes that it generates less undo and redo than a row-by-row delete operation.

By contrast, `DELETE` is a DML statement that identifies and removes rows individually, including when its predicate matches all rows. Its row-level processing supports transactional behavior: the deletion can be rolled back before commit, and applicable delete triggers execute. Those capabilities are appropriate when selective removal or transaction control is required, but they create substantially more work for a large complete-table removal. The statement is therefore true as a general performance comparison for removing all rows; actual elapsed time can still be affected by storage, concurrency, logging, and database configuration. See Oracle's TRUNCATE TABLE documentation and DELETE documentation.

QUESTION NO: 50

Which three statements are true about external tables? (Choose three.)

A. DML statements can modify them.

B. They can be temporary tables.

C. They can be used in queries containing joins.

D. They can be indexed.

E. They can be used in queries containing sorts.

F. Their metadata is stored in the database.

ANSWER: C E F

Explanation:

External tables can be used in queries containing joins, can be used in queries containing sorts, and have their metadata stored in the database. An external table is defined in Oracle with normal table-definition metadata, including its columns and external-data access parameters, while the actual row data remains in an operating-system file or other supported external source. Oracle stores the definition in the data dictionary, allowing users to refer to the external table by name in SQL just as they would a conventional table.

Because an external table is available as a SQL row source, it can participate in a query block with other tables. This includes join operations, where Oracle combines the external-table rows with rows from database tables or other query sources according to the join condition. It also includes sorting: an `ORDER BY` clause may sort rows returned from an external table, and the optimizer can use the external table within larger query plans involving sorting operations. These capabilities make external tables useful for querying and transforming file-based data through ordinary SQL without first loading that data into a permanent database table. Oracle documents that external tables are read-only query sources whose definitions are maintained by the database. See Oracle Database Utilities: External Tables and Oracle SQL Language Reference: CREATE TABLE.

QUESTION NO: 51

Using the CUSTOMERS table, you need to generate a report that shows 50% of each credit amount in each income level. The report should NOT show any repeated credit amounts in each income level.

Which query would give the required result?

A. SELECT cust_income_level || ' ' || cust_credit_limit * 0.50 AS "50% Credit Limit" FROM customers.

B. SELECT DISTINCT cust_income_level || ' ' || cust_credit_limit * 0.50 AS "50% Credit Limit" FROM customers.

C. SELECT DISTINCT cust_income_level, DISTINCT cust_credit_limit * 0.50 AS "50% Credit Limit" FROM customers.

D. SELECT cust_income_level, DISTINCT cust_credit_limit * 0.50 AS "50% Credit Limit" FROM customers

ANSWER: B

Explanation:

`SELECT DISTINCT cust_income_level || ' ' || cust_credit_limit * 0.50 AS "50% Credit Limit" FROM customers` produces the required report because it calculates half of each customer credit limit and combines that calculated amount with the corresponding income level in one displayed value. Oracle evaluates the multiplication before the concatenation, so `cust_credit_limit * 0.50` is the numeric half-credit calculation that is included in the result.

The `DISTINCT` keyword applies to the complete selected expression. Therefore, when more than one row has the same income level and the same calculated 50% credit-limit value, Oracle returns that displayed result only once. This gives a unique set of income-level/half-credit-limit combinations, which satisfies the requirement to avoid repeated credit amounts within an income level. Oracle documents that `DISTINCT` eliminates duplicate rows from a query result, based on the selected expressions. The expression also uses Oracle's concatenation operator, `||`, to present the income level and computed amount together in the requested single report column. See the Oracle SQL Language Reference for SELECT and DISTINCT and concatenation operators.

QUESTION NO: 52

Which is the default column or columns for sorting output from compound queries using SET operators such as INTERSECT in a SQL statement?

A. The first column in the last SELECT of the compound query

B. The first NUMBER column in the first SELECT of the compound query

C. The first VARCHAR2 column in the first SELECT of the compound query

D. The first column in the first SELECT of the compound query

E. The first NUMBER or VARCHAR2 column in the last SELECTof the compound query

ANSWER: D

Explanation:

The first column in the first SELECT of the compound query is the expected answer for Oracle Database 12c SQL examination purposes. A compound query combines the result sets produced by its component queries through set operators such as `UNION`, `UNION ALL`, `INTERSECT`, and `MINUS`. The first query establishes the compound query's result-column names, positions, and corresponding data types. Consequently, when Oracle's set-operation processing produces its default sorted result behavior, the sort is based on the first result column from that first query, in ascending order. This is a property commonly tested with compound-query questions and is distinct from an explicit `ORDER BY` clause, which can specify one or more result columns or aliases and is written at the end of the complete compound query. In production SQL, include an explicit `ORDER BY` whenever a particular presentation order is required, because explicit ordering expresses the application requirement rather than relying on set-operation processing behavior. Oracle documents the syntax and rules for compound queries and set operators in the Oracle Database 12c SQL Language Reference: The UNION [ALL], INTERSECT, MINUS Operators; the placement and use of ordering clauses is described in the SELECT statement reference.

QUESTION NO: 53

Which two are true about granting object privileges on tables, views, and sequences?

A. DELETE can be granted on tables, views, and sequences.

B. REFERENCES can be granted only on tables.

C. INSERT can be granted only on tables and sequences.

D. SELECT can be granted on tables, views, and sequences.

E. ALTER can be granted only on tables and sequences.

ANSWER: B D E

Explanation:

REFERENCES can be granted only on tables is correct. The REFERENCES object privilege lets a user define a referential-integrity constraint whose foreign key refers to columns in another user's table. Oracle documents REFERENCES as a table object privilege, and it may be granted for the whole table or for specified columns.

SELECT can be granted on tables, views, and sequences is also correct. On a table or view, SELECT permits querying the object. On a sequence, SELECT permits use of the sequence pseudocolumns, including `NEXTVAL` and `CURRVAL`, subject to the normal sequence-use rules.

ALTER can be granted only on tables and sequences is correct as well. Oracle's object-privilege syntax identifies ALTER as an available object privilege for tables and sequences. Granting ALTER allows the grantee to alter the applicable object type; the privilege is not available for a view. Therefore, despite the question asking for two answers, these three statements are true according to Oracle's documented object privilege rules.

See Oracle's GRANT statement documentation and the CREATE SEQUENCE documentation for the sequence privilege requirements.

QUESTION NO: 54

Which two statements are true about a full outer join?

A. It includes rows that are returned by an inner join.

B. The Oracle join operator (+) must be used on both sides of the join condition in the WHERE clause.

C. It includes rows that are returned by a Cartesian product.

D. It returns matched and unmatched rows from both tables being joined.

E. It returns only unmatched rows from both tables being joined.

ANSWER: A D

Explanation:

A full outer join returns the complete set of rows produced by matching the join condition, which necessarily includes every row an inner join would return. Where a row in one table has a corresponding row in the other table, the joined result contains the combined column values for that match. Thus, the statement “It includes rows that are returned by an inner join.” is true.

It also preserves nonmatching rows from *both* joined tables. For a row from either input with no matching row in the other input, Oracle returns the available table’s values and supplies nulls for columns from the table that has no match. Therefore, “It returns matched and unmatched rows from both tables being joined.” accurately describes a full outer join. Oracle specifies the ANSI syntax as `FULL OUTER JOIN`; `OUTER` is optional, so `FULL JOIN` is equivalent. This join is useful when a comparison or reconciliation must retain all records from each source, including records that lack a counterpart.

See Oracle Database SQL Language Reference documentation on joins and the SELECT statement.

QUESTION NO: 55

Which two statements are true regarding a SAVEPOINT?

A. Rolling back to a SAVEPOINT can undo a CREATE INDEX statement.

B. Only one SAVEPOINT may be issued in a transaction.

C. A SAVEPOINT does not issue a COMMIT

D. Rolling back to a SAVEPOINT can undo a TRUNCATE statement.

E. Rolling back to a SAVEPOINT can undo a DELETE statement

ANSWER: C E

Explanation:

A SAVEPOINT establishes a named marker within the current transaction; it does not make any transaction changes permanent. Consequently, the statement "A SAVEPOINT does not issue a COMMIT" is true. Until an explicit COMMIT or ROLLBACK occurs, the transaction remains active, locks and uncommitted work remain subject to normal transaction processing, and later work can be selectively reversed. Oracle documents that a savepoint identifies a point in a transaction to which the transaction can subsequently be rolled back.

The statement "Rolling back to a SAVEPOINT can undo a DELETE statement" is also true when the DELETE is performed after the savepoint was established and the transaction has not been committed. DELETE is transactional DML, so issuing ROLLBACK TO SAVEPOINT reverses its row deletions while retaining changes that occurred before that savepoint. This permits an application or user to recover from part of a unit of work without abandoning the entire transaction. The savepoint remains usable after a rollback to it, allowing subsequent transaction work to continue or another partial rollback to be issued as appropriate. Oracle SQL Language Reference documents the SAVEPOINT statement and its use with ROLLBACK in the

transaction-control documentation: Oracle Database SQL Language Reference: SAVEPOINT and Oracle Database SQL Language Reference: ROLLBACK.

QUESTION NO: 56

Which three actions can you perform on an existing table containing date?

A. Add a new column as the table's first column.

B. Define a default value that is automatically inserted into a column containing nulls.

C. Add a new NOT NULL Column with a DEFAULT value.

D. Change a DATE Column containing data to a NUMBER data type.

E. Increase the width of a numeric column.

F. Change the default value of a column.

ANSWER: C E F

Explanation:

Adding a new NOT NULL Column with a DEFAULT value is supported because the default supplies a value for every pre-existing row when the column is added, allowing the NOT NULL constraint to be satisfied. Oracle documents that when a column is added with both a default value and a NOT NULL constraint, the default is logically applied to existing rows without requiring the database to physically update every row immediately. This is a useful way to introduce mandatory attributes to populated tables.

Increasing the width of a numeric column is also a permitted ALTER TABLE modification. For a NUMBER column defined with precision and scale, increasing the precision expands the range of values that may subsequently be stored while retaining the existing values.

Changing the default value of a column is permitted through ALTER TABLE ... MODIFY. A default is used when a future INSERT omits the column value (or explicitly uses the DEFAULT keyword); changing it establishes the new default for later rows and does not rewrite values already stored in the table. Oracle's SQL Language Reference describes these ADD and MODIFY column operations in the ALTER TABLE documentation, and explains default-column behavior in the CREATE TABLE documentation.

QUESTION NO: 57

You need to allow user ANDREW to:

1. Modify the TITLE and ADDRESS columns of your CUSTOMERS table.

2. GRANT that permission to other users.

Which statement will do this?

A. GRANT UPDATE (title, address) ON customers TO andrew WITH GRANT OPTION;

B. GRANT UPDATE (title, address) ON customers TO andrew WITH ADMIN OPTION;

C. GRANT UPDATE ON customers.title, customers.address TO andrew WITH ADMIN OPTION;

D. GRANT UPDATE (title, address) ON customers TO andrew;

E. GRANT UPDATE ON customers.title, customers.address TO andrew WITH GRANT OPTION;

F. GRANT UPDATE ON customers.title, customers.address TO andrew;

ANSWER: A

Explanation:

`GRANT UPDATE (title, address) ON customers TO andrew WITH GRANT OPTION;` grants an object privilege at the column level. The parenthesized column list limits Andrew's UPDATE privilege to the TITLE and ADDRESS columns of the CUSTOMERS table, rather than granting permission to update every column in that table. The `WITH GRANT OPTION` clause is required because it allows Andrew to grant the same UPDATE privilege, including its permitted columns, to other users. This is the applicable delegation clause for object privileges such as UPDATE on a table. Oracle documents that object privileges may be granted for specified columns for UPDATE, and that `WITH GRANT OPTION` authorizes the recipient to grant the object privilege onward. The grantor must own CUSTOMERS or have received the relevant privilege with the ability to grant it onward. See Oracle's GRANT SQL statement reference and the Database Security Guide authorization overview.

QUESTION NO: 58

Which three statements are true about dropping and unused columns in an Oracle database?

A. A primary key column referenced by another column as a foreign key can be dropped if using the CASCADE CONSTRAINTS option.

B. A DROP COLUMN command can be rolled back.

C. An UNUSED column's space is reclaimed automatically when the block containing that column is next queried.

D. An UNUSED column's space is reclaimed automatically when the row containing that column is next queried.

E. Partition key columns cannot be dropped.

F. A column that is set to UNUSED still counts towards the limit of 1000 columns per table.

ANSWER: A E F

Explanation:

A primary key column referenced by a foreign key can be dropped when the `CASCADE CONSTRAINTS` clause is used. Oracle removes the referential-integrity constraints that depend on the key being dropped, allowing the column alteration to complete. This is the applicable form of the cascade clause for an `ALTER TABLE ... DROP COLUMN` operation.

Partition key columns cannot be dropped. A partitioned table's partitioning definition depends on those columns, so Oracle protects them from removal through the normal drop-column operation. This preserves the structural metadata and partitioning behavior of the table.

A column marked `UNUSED` still counts toward Oracle Database's maximum number of columns for the table. Setting a column unused is a logical operation that makes the column inaccessible to SQL while retaining its metadata until the unused column is physically dropped. Consequently, it remains relevant to the table's column limit until a subsequent `DROP UNUSED COLUMNS` operation removes it. Oracle documents these restrictions and the behavior of unused columns in its SQL Language Reference: ALTER TABLE and Oracle Database Administrator's Guide.

QUESTION NO: 59

Which two tasks can be performed by using Oracle SQL statements? (Choose two.)

A. changing the password for an existing database user

B. connecting to a database instance

C. querying data from tables in different databases

D. starting up a database instance

E. executing operating system (OS) commands in a session

ANSWER: A C

Explanation:

Changing the password for an existing database user can be performed with the SQL `ALTER USER` statement, using the `IDENTIFIED BY` clause to assign a new password. For example, an appropriately privileged administrator can issue `ALTER USER user_name IDENTIFIED BY new_password`. This is a data-definition statement executed in an Oracle SQL environment and is part of normal database user administration.

Querying data from tables in different databases can be performed by issuing a distributed SQL query through an Oracle database link. A database link supplies the connection information for a remote database, and a table can be referenced using the remote-object syntax, such as `table_name@database_link`. Oracle processes the statement as a SQL query while coordinating access to the remote database, allowing data to be selected from local and remote objects in the same query when the required link and privileges exist. Oracle documents the password-management syntax in its ALTER USER SQL Reference and describes remote access through links in the Database Links administration documentation.

QUESTION NO: 60

Examine the create table statements for the stores and sales tables.

SQL> CREATE TABLE stores(store_id NUMBER(4) CONSTRAINT store_id_pk PRIMARY KEY, store_name VARCHAR2(12), store_address VARCHAR2(20), start_date

DATE);

SQL> CREATE TABLE sales(sales_id NUMBER(4) CONSTRAINT sales_id_pk PRIMARY KEY, item_id NUMBER(4), quantity NUMBER(10), sales_date DATE, store_id NUMBER(4), CONSTRAINT store_id_fk FOREIGN KEY(store_id) REFERENCES stores(store_id));

You executed the following statement:

SQL> DELETE from stores

WHERE store_id=900;

The statement fails due to the integrity constraint error:

ORA-02292: integrity constraint (HR.STORE_ID_FK) violated

Which three options ensure that the statement will execute successfully? (Choose three.)

A. Disable the primary key in the STORES table.

B. Use CASCADE keyword with DELETE statement.

C. DELETE the rows with STORE_ID = 900 from the SALES table and then delete rows from STORES table.

D. Disable the FOREIGN KEY in SALES table and then delete the rows.

E. Create the foreign key in the SALES table on STORE_ID column with ON DELETE CASCADE option.

ANSWER: C D E

Explanation:

The successful approaches address the child rows in `SALES` that reference the parent row in `STORES`. Deleting the rows with `STORE_ID = 900` from `SALES` first removes the referencing child rows. The subsequent deletion of the matching `STORES` row then satisfies the foreign-key requirement and succeeds.

Disabling the foreign key in the SALES table also allows the parent row to be deleted because Oracle does not enforce that relationship while the constraint is disabled. This is an administrative approach that should be followed by appropriate data cleanup or validation before the constraint is re-enabled.

Defining the foreign key on SALES.STORE_ID, referencing STORES.STORE_ID, with ON DELETE CASCADE makes the relationship self-maintaining for this operation. When the store row is deleted, Oracle automatically deletes all sales rows whose STORE_ID references that parent row. Oracle documents ON DELETE CASCADE as a referential action for foreign keys in its constraints documentation; the syntax is also described in the CREATE TABLE reference.

QUESTION NO: 61

Which two statements are true about single-row functions? (Choose two.)

A. CEIL: can be used for positive and negative numbers

B. FLOOR: returns the smallest integer greater than or equal to a specified number

C. TRUNC: can be used with NUMBER and DATE values

D. CONCAT: can be used to combine any number of values

E. MOD: returns the quotient of a division operation

ANSWER: A C

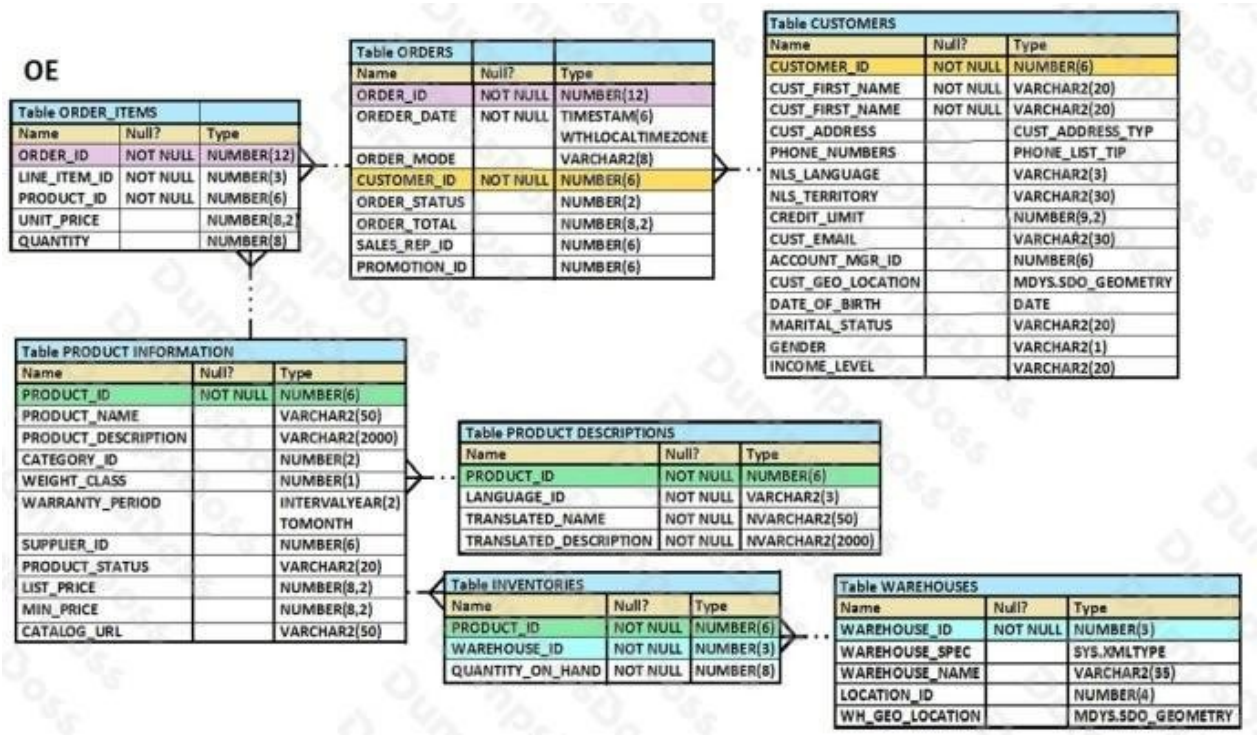
Explanation:

CEIL can be used with both positive and negative numeric values. Oracle defines CEIL as returning the smallest integer that is greater than or equal to its numeric argument. This definition applies regardless of whether the supplied number is above or below zero. For example, CEIL(4.2) returns 5, while CEIL(-4.2) returns -4. Therefore, it is a valid numeric single-row function for positive and negative inputs.

TRUNC can be used with both NUMBER and DATE values. In its numeric form, it truncates a number to a specified decimal precision, without rounding. In its date form, it returns a date with the specified unit of time removed, such as truncating a date to the beginning of its month or year. Oracle documents numeric and datetime variants of this function, making it useful in expressions that need to normalize either numerical precision or date/time granularity. See Oracle's CEIL documentation and TRUNC documentation.

QUESTION NO: 62

View the Exhibit and examine the structure of ORDER_ITEMS and ORDERS tables.



You need to remove from the ORDER_ITEMS table those rows that have an order status of 0 or 1 in the ORDERS table.
Which two DELETE statements are valid (Choose two.)

A. DELETE *
FROM order_items
WHERE order_id IN (SELECT order_id)
FROM orders
WHERE order_status IN (0,1));

B. DELETE
FROM (SELECT * FROM order_items I,orders o
WHERE i.order_id = o.order_id AND order_status IN (0,1));

C. DELETE FROM order_items i
WHERE order_id = (SELECT order_id FROM orders o
WHERE i.order_id = o.order_id AND order_status IN (0,1));

D. DELETE
FROM order_items
WHERE order_id IN (SELECT order_id
FROM orders
WHERE orders_status in (0,1));

ANSWER: B C

Explanation:

The statement `DELETE FROM (SELECT * FROM order_items I,orders o WHERE i.order_id = o.order_id AND order_status IN (0,1));` is valid because Oracle permits deletion through a join view when exactly one underlying table is key-preserved. Based on the table structure, each `ORDER_ITEMS` row is associated with one `ORDERS` row through `ORDER_ID`, while one order can have multiple item rows. Therefore, `ORDER_ITEMS` is the key-preserved table in this join, and the qualifying item rows can be deleted through the inline view.

The correlated statement `DELETE FROM order_items i WHERE order_id = (SELECT order_id FROM orders o WHERE i.order_id = o.order_id AND order_status IN (0,1));` is also valid. For each candidate item row, its correlated subquery checks the corresponding order and returns that order's identifier only when its status is 0 or 1. Because `ORDER_ID` uniquely identifies an order, the scalar subquery returns at most one value for each item row. The equality condition then selects precisely the item rows belonging to qualifying orders for deletion. Oracle documents both the

DELETE statement and its support for subqueries and deletions through eligible views at Oracle Database SQL Language Reference: DELETE and the key-preserved-table requirements at Oracle Database Administrator's Guide: Views.

QUESTION NO: 63

You create a table named 123.

Which statement runs successfully?

A. SELECT * FROM TABLE (123) ;

B. SELECT * FROM & apos;123 & apos;;

C. SELECT * FROM " 123 " ;

D. SELECT * FROM V & apos;123V & apos;;

E. SELECT * FROM "123";

ANSWER: E

Explanation:

SELECT * FROM "123"; runs successfully because an Oracle identifier consisting solely of digits must be a quoted identifier. Oracle's unquoted identifiers must begin with an alphabetic character, whereas double quotation marks allow identifiers that would otherwise not meet ordinary naming rules. Therefore, for a table to be named exactly 123, it must have been created with double quotation marks, such as CREATE TABLE "123" (...), and subsequent SQL must use the same quoted name. Quoted identifiers are case-sensitive and preserve every character inside the quotes, including spaces. The successful statement uses precisely "123", with no embedded leading or trailing spaces, so it resolves to the table named 123. Oracle documents the rules for quoted and nonquoted identifiers in its SQL Language Reference: Schema Object Names and Qualifiers. The general SELECT syntax and use of a table as the query's FROM source are documented at SELECT.

QUESTION NO: 64

Examine the data in the ORD_ITEMS table:

ORD_ID	ITEN_NO	QTY
1	111	10
1	222	20
1	333	30
2	333	30
2	444	40
3	111	40

Evaluate this query:

```
SQL>SELECT item_no, AVG(qty)
FROM ord_items
HAVING AVG(qty) > MIN(qty) * 2
GROUP BY item_no;
```

Which statement is true regarding the result?

A. It returns an error because the HAVING clause should be specified after the GROUP BY clause.

B. It returns an error because all the aggregate functions used in the HAVING clause must be specified in the SELECT list.

C. It displays the item nos with their average quantity where the average quantity is more than double the minimum quantity of that item in the table.

D. It displays the item nos with their average quantity where the average quantity is more than double the overall minimum quantity of all the items in the table.

ANSWER: C

Explanation:

It displays the item nos with their average quantity where the average quantity is more than double the minimum quantity of that item in the table. The query groups ORD_ITEMS rows by item number, so each resulting group contains the rows for one item number. For each such group, `AVG(quantity)` calculates that item's average ordered quantity and `MIN(quantity)` calculates the smallest quantity recorded for that same item. The `HAVING` condition is evaluated after grouping and retains only item-number groups for which the average quantity exceeds twice the group's minimum quantity. Therefore, the minimum used in the comparison is not a single minimum calculated across all rows in ORD_ITEMS; it is recalculated independently within every item-number group. Oracle SQL permits aggregate expressions in a HAVING condition without requiring those expressions to appear in the select list. Oracle also defines HAVING as a condition that restricts groups produced by a GROUP BY clause. See the Oracle Database 12c SELECT statement documentation and the Oracle aggregate functions documentation.

QUESTION NO: 65

Examine the structure of the EMPLOYEES table.

Name	Null?	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
FIRST_NAME		VARCHAR2(20)
LAST_NAME	NOT NULL	VARCHAR2(25)
EMAIL	NOT NULL	VARCHAR2(25)
PHONE_NUMBER		VARCHAR2(20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
SALARY		NUMBER(8,2)
COMMISSION_PCT		NUMBER(2,2)
MANAGER_ID		NUMBER(6)
DEPARTMENT_ID		NUMBER(4)

You must display the details of employees who have manager with MANAGER_ID 100, who were hired in the past 6 months and who have salaries greater than 10000.

Which query would retrieve the required result?

A.
```
SELECT last_name, hire_date, salary
FROM employees
WHERE salary > 10000
UNION ALL
SELECT last_name, hire_date, salary
FROM employees
WHERE manager_ID = (SELECT employee_id FROM employees WHERE employee_id = 100)
INTERSECT
SELECT last_name, hire_date, salary
FROM employees
WHERE hire_date > SYSDATE-180;
```

B. SELECT last_name, hire_date, salary
FROM employees
WHERE manager_id = (SELECT employee_id FROM employees WHERE employee_id = 100) UNION ALL
(SELECT last_name, hire_date, salary
FROM employees
WHERE hire_date > SYSDATE-180
INTERSECT
SELECT last_name, hire_date, salary
FROM employees
WHERE salary > 10000);

C. SELECT last_name, hire_date, salary
FROM employees
WHERE manager_id = (SELECT employee_id FROM employees WHERE employee_id = '100')
UNION
SELECT last_name, hire_date, salary
FROM employees
WHERE hire_date > SYSDATE-180
INTERSECT
SELECT last_name, hire_date, salary
FROM employees
WHERE salary > 10000;

D. (SELECT last_name, hire_date, salary
FROM employees
WHERE salary > 10000
UNION ALL
SELECT last_name, hire_date, salary
FROM employees
WHERE manager_ID = (SELECT employee_id FROM employees WHERE employee_id = 100))
UNION
SELECT last_name, hire_date, salary
FROM employees
WHERE hire_date > SYSDATE-180;

E. SELECT last_name, hire_date, salary
FROM employees
WHERE manager_id = 100
AND hire_date >= ADD_MONTHS(SYSDATE, -6)
AND salary > 10000;

ANSWER: E

Explanation:

The query `SELECT last_name, hire_date, salary FROM employees WHERE manager_id = 100 AND hire_date >= ADD_MONTHS(SYSDATE, -6) AND salary > 10000;` is correct because all three required conditions must be true for each returned employee row. Combining the predicates with `AND` directly expresses the requirement that an employee reports to manager 100, has a hire date within the preceding six calendar months, and earns more than 10,000. A direct comparison to `manager_id = 100` is sufficient because the required manager identifier is already known; no subquery is needed to obtain that same value. Using `ADD_MONTHS(SYSDATE, -6)` accurately represents six calendar months before the current database date, rather than treating every six-month period as a fixed number of days. Oracle evaluates the `WHERE` condition for each row and returns only rows for which the combined Boolean expression is true. The selected columns then provide the requested employee details without returning unnecessary attributes. Oracle documents conditional row filtering in the `WHERE` clause and the use of `ADD_MONTHS` for month-based date arithmetic: Oracle SELECT statement documentation and Oracle ADD_MONTHS documentation.

QUESTION NO: 66

Which statement is true regarding external tables?

A. The CREATE TABLE AS SELECT statement can be used to upload data into a normal table in the database from an external table.

B. The data and metadata for an external table are stored outside the database.

C. The default REJECT LIMIT for external tables is UNLIMITED.

D. ORACLE_LOADER and ORACLE_DATAPUMP have exactly the same functionality when used with an external table.

ANSWER: A

Explanation:

The CREATE TABLE AS SELECT statement can be used to upload data into a normal table in the database from an external table. An external table lets Oracle Database access data held in an operating-system file as though it were a read-only relational table. Its table definition is maintained in the database data dictionary, while the rows remain in the external source file. Because an external table can be queried with SQL, it can serve as the source of a `CREATE TABLE ... AS SELECT` operation. This creates a conventional database table and populates it with the rows returned from the external table query, thereby bringing those data into database storage. The same approach can include a query predicate, expressions, joins, or selected columns, so the loaded data need not be an unchanged copy of the source file. Once created, the resulting normal table is managed like other database tables and supports ordinary database operations, unlike the external table source itself. Oracle documents that external tables may be queried and can be used as the source for creating a table with `CREATE TABLE AS SELECT`. See the Oracle Database Administrator's Guide: External Tables and the CREATE TABLE SQL Reference.

QUESTION NO: 67

Examine the description of the PRODUCTS table:

Name	Null?	Type
PROD_ID	NOT NULL	NUMBER(2)
QTY		NUMBER(5,2)
COST		NUMBER(8,2)

Which two statements execute without errors?

A. MERGE INTO new_prices nUSING (SELECT * FROM products) pWHEN MATCHED THENUPDATE SET n.price= p.cost* 01WHEN NOT MATCHED THENINSERT(n.prod_id, n.price) VALUES(p.prod_id, cost*.01)WHERE(p.cost<200);

B. MERGE INTO new_prices nUSING (SELECT * FROM products WHERE cost>150) pON (n.prod_id= p.prod_id)WHEN MATCHED THENUPDATE SET n.price= p.cost*.01DELETE WHERE (p.cost<200);

C. MERGE INTO new_prices nUSING products pON (p.prod_id =n.prod_id)WHEN NOT MATCHED THENINSERT (n.prod_id, n.price) VALUES (p.prod_id, cost*.01)WHERE (p.cost<200);

D. MERGE INTO new_prices nUSING (SELECT * FROM products WHERE cost>150) pON (n.prod_id= p.prod_id)WHEN MATCHED THENDELETE WHERE (p.cost<200)

ANSWER: B

Explanation:

The statement beginning `MERGE INTO new_prices n` and using the filtered source query `SELECT * FROM products WHERE cost>150` executes successfully. It supplies all mandatory structural elements of an Oracle `MERGE`: a target table, a source row set, an `ON` condition defining how source and target rows are matched, and a valid `WHEN MATCHED THEN UPDATE` action. The match condition compares `prod_id` values, while the update calculates the target price from the source cost using `p.cost*.01`.

Oracle permits a `DELETE WHERE` clause as part of the matched update clause. This deletion is applied only to rows that were matched and updated by the merge operation; its condition can reference source columns such as `p.cost`. Because

the source query already limits rows to costs greater than 150, the delete predicate can further select the matched, updated rows whose cost is below 200. This is a supported way to combine conditional updating and deletion in one `MERGE` statement. Oracle documents the complete `MERGE` syntax, including the matched update and delete clauses, in the Oracle Database SQL Language Reference: MERGE. The applicable grammar and semantics are also summarized in Oracle's MERGE statement reference.

QUESTION NO: 68

Examine the description of the ENPLOYES table:

Which query requires explicit data type conversion?

A. SELECT SUBSTR(join_date, 1, 2) - 10 FROM employees;

B. SELECT join_date + '20' FROM employees;

C. SELECT join_date || '' || salary FROM employees;

D. SELECT join_date FROM employees WHERE join_date > TO_DATE('10-02-2018', 'DD-MM-YYYY');

E. SELECT salary + '120.50' FROM employees;

ANSWER: D

Explanation:

`SELECT join_date FROM employees WHERE join_date > TO_DATE('10-02-2018', 'DD-MM-YYYY');` is the correct query because `join_date` is a DATE value and the comparison value is supplied as character data. `TO_DATE` explicitly converts that character literal into a DATE and, crucially, supplies the format model that defines how the literal is arranged: day, month, and four-digit year. This makes the comparison unambiguous and independent of a session's `NLS_DATE_FORMAT` setting. Explicit conversion is especially important for date literals written in a numeric format such as `10-02-2018`, because relying on implicit conversion makes SQL behavior dependent on the executing session's locale and date-format settings. Oracle documents that `TO_DATE` converts a character value to DATE and recommends specifying a format mask when a character value is converted to a date. The resulting DATE can then be compared directly with the DATE column, producing reliable date-based filtering rather than a format-dependent character interpretation. See Oracle's TO_DATE documentation and its data type comparison and conversion documentation.