DUMPSBOSS.

MySQL 5.7 Database Administrator

Oracle 1z0-888

Version Demo

Total Demo Questions: 20

Total Premium Questions: 205

Buy Premium PDF

https://dumpsboss.co

support@dumpsboss.co

support@dumpsboss.co
dumpsboss.co

Topic Break Down

Topic	No. of Questions
Topic 1, MySQL Architecture	26
Topic 2, MySQL Installation	6
Topic 3, MySQL Configuration	18
Topic 4, MySQL Security	46
Topic 5, MySQL Backup and Recovery	31
Topic 6, MySQL Replication	40
Topic 7, MySQL Cluster	1
Topic 8, MySQL Performance Tuning	37
Total	205

QUESTION NO: 1

Force Majeure is a catastrophic failure on a major level of the database operation. Regular backups are key to helping avoid data loss in such situations.

Which two other steps can help avoid data loss in a major catastrophe? (Choose two.)

A. Implement a failover strategy to another geographic location.

B. Create a master-master pair for each service.

C. Have a second data centre in a different region or country.

D. Keep software updated to the latest version.

E. Use RAID 10 storage for data.

F. Use on-site network-attached storage to separate service from data.

ANSWER: A C

Explanation:

Implement a failover strategy to another geographic location and have a second data centre in a different region or country provide geographic redundancy, which is essential when a disaster affects an entire site, local utility infrastructure, or regional network access. A disaster-recovery design needs a production-capable environment outside the affected failure domain, along with a defined process for redirecting application traffic and restoring service there. Geographic separation also reduces the risk that the primary database, its local replicas, and its backup infrastructure are all lost or inaccessible during the same event.

For MySQL, replication can maintain a copy of data at a remote site, and high-availability tooling can support promotion or failover when the original site is unavailable. The key is that the alternate environment is sufficiently independent geographically and operationally to survive the catastrophe affecting the primary location. This strategy complements regular backups: backups provide point-in-time recovery capability, while remote failover infrastructure helps preserve service continuity and reduces recovery time after a site-level loss. MySQL's replication documentation describes maintaining replicas of a source server, including replicas used for backup and recovery purposes; Oracle's high-availability guidance also emphasizes redundancy and failover planning. See MySQL 5.7 Replication and Oracle High Availability Overview.

QUESTION NO: 2

Multiversion Concurrency Control (MVCC) enables what type of backup?

A. Hot

B. Binary

C. Logical

D. Incremental

ANSWER: A

Explanation:

Hot is correct. InnoDB implements multiversion concurrency control by retaining older row versions as needed, allowing a transaction to read a consistent snapshot of data without blocking concurrent updates. This capability is fundamental to taking a backup while the database remains online and continues serving application activity—a hot backup. The backup process can obtain a transactionally consistent view of InnoDB data while other sessions insert, update, or delete rows. For logical backup operations, MySQL tools can use a consistent snapshot through options such as `--single-transaction` with InnoDB tables; physical online backup tools likewise rely on InnoDB's transactional consistency mechanisms together with required log information. MVCC therefore enables backup activity without requiring the normal workload to be stopped, which is the defining characteristic of a hot backup. The consistency guarantee applies to transactional storage engines such

as InnoDB, and operational planning must still account for nontransactional tables and binary log position requirements where applicable. See the MySQL 5.7 documentation on InnoDB multi-versioning and mysqldump --single-transaction.

QUESTION NO: 3

You have just created a replication slave from a backup of the master made with mysqldump:

```
mysqldump -u backup -p --all-databases > /backups/mysql.sql
```

You try to log in to the slave with the application user, but fail as follows:

```
mysql -u application -p
ERROR 1045 (28000): Access denied for user 'application'@'localhost' (using password:
YES)
```

The login works on the master.

Which two changes to the process can fix the issue?

A. After the restore, log in to the database and execute FLUSH PRIVILEGES.

B. Use the –-flush-privileges with mysqldump.

C. Add a second dump for the 'mysql' database; a dump that specifies only application databases does not include it.

D. Use the –-grants option to include GRANT statements in the dump.

E. Include the mysql system database in the dump and execute FLUSH PRIVILEGES after restoring it.

ANSWER: E

Explanation:

Include the `mysql` system database in the dump and execute `FLUSH PRIVILEGES` after restoring it. MySQL stores account definitions, authentication credentials, host matching information, and privilege assignments in the grant tables in the `mysql` system database. A dump containing only application databases therefore does not recreate the application account on the replica, even when all application tables and data have been restored successfully. The backup process must include those grant tables, either by dumping `mysql` separately or by using a dump method that includes all system and user databases as appropriate for the recovery procedure.

After the dump is loaded, `FLUSH PRIVILEGES` instructs the running server to reload the grant tables into memory. This makes the newly restored account and its privileges available for authentication without requiring a server restart. Consequently, the application user can authenticate to the replica using the same account data that existed on the source server. Oracle documents the grant-table reload behavior in the MySQL 5.7 privilege system documentation and describes database-selection behavior for backups in the mysqldump reference.

QUESTION NO: 4

Which two statements are true regarding MySQL security? (Choose two.)

A. The mysql user needs to have sudo privileges.

B. The mysqld process owner should own all files and directories to which the server writes.

C. The root or administrator users should own all files and directories to which the server writes.

D. The mysqld process should be run as root or administrator.

E. The mysqld process should not be run as root or administrator.

ANSWER: B E

Explanation:

MySQL security best practice is to run the server under a dedicated, unprivileged operating-system account, commonly named `mysql`, rather than as the system root or Administrator account. This limits the scope of damage if the server process is compromised: the process has access only to resources explicitly granted to that account, instead of unrestricted system-level access. Accordingly, “The mysqld process should not be run as root or administrator.” is correct.

The operating-system account that runs `mysqld` must also be able to create, modify, and remove the files that MySQL manages, including database files, log files, temporary files, and other server-maintained data. Therefore, “The mysqld process owner should own all files and directories to which the server writes.” correctly expresses the required ownership principle. During installation, MySQL documentation directs administrators to set ownership of the data directory to the account used to run the server. Proper ownership, combined with restrictive file permissions, ensures that the MySQL service can operate normally while minimizing unnecessary access by other local users. See the MySQL 5.7 Security Guidelines and MySQL default privilege guidance.

QUESTION NO: 5

An admin attempts to enforce stronger security by using these commands:

The admin then leaves the system running with the specified changes. What are two remaining security concerns? (Choose two.)

A. validate_password_policy cannot be set without restarting the MySQL instance.

B. The name of the dictionary file is too obvious.

C. The dictionary file word list is too short.

D. validate_password_dictionary_file cannot be set without restarting the MySQL instance.

E. The validate_password plug-in has not been loaded.

F. The dictionary file is in an insecure location.

ANSWER: B F

Explanation:

The name of the dictionary file is too obvious remains a concern because the password-validation dictionary can reveal words and patterns that the administrator specifically intends to prohibit. A predictable filename makes it easier for an unauthorized local user who can browse the relevant directory to identify and examine that security-related file. Knowing its contents can help an attacker tailor candidate passwords to avoid the dictionary checks while still satisfying the configured validation policy.

The dictionary file is in an insecure location is also a concern because the MySQL server must be able to read the file, but that does not require it to be exposed to untrusted users. The dictionary should be stored in a protected directory and protected with restrictive operating-system ownership and permissions. This preserves the integrity of the word list: unauthorized modification could remove prohibited words or substitute a weakened dictionary, reducing the effectiveness of strong-password enforcement. MySQL documents the dictionary-file setting as part of the validate_password plug-in configuration and notes that the server reads the specified file. See the MySQL 5.7 validate_password system variables documentation and the MySQL 5.7 validate_password plug-in documentation.

QUESTION NO: 6

You have a config file for a running DB with this excerpt:

[mysqld] tmp_table_size=16M sort_buffer_size=256k

```
shell> mysqldump -u root -p applicationdb > /backup/applicationdb.sql
shell> /etc/init.d/mysql stop
shell> cd /var/lib/mysql/
shell> rm ibdata1 ib_logfile0 ib_logfile1
shell> /etc/init.d/mysql start
shell> mysql -u root -p applicationdb < /backup/applicationdb.sql
```

To address a query performance problem of connecting to the DB from an application on another host, you log in and make these changes to the DB:

mysql> SET GLOBAL tmp_table_size=32000000; mysql> SET sort_buffer_size=2000000;

This solves the problem with your queries. However, later the DB instance is restarted and the performance problem returns.

Which three best describe this scenario? (Choose three.)

A. Global variables are not persistent across server restarts.

B. Session variables are not persistent across server restarts.

C. The query benefited from sort_buffer_size increase.

D. sort_buffer_size should match tmp_table_size to be optimal.

E. The query benefited from tmp_table_size increase.

F. The query benefited from sort_buffer_size and tmp_table_size increases.

ANSWER: A B E

Explanation:

Global variables are not persistent across server restarts. The `SET GLOBAL tmp_table_size=32000000` statement changes the running server's global value, but it does not rewrite the option file. On restart, MySQL again reads `tmp_table_size=16M` from the `[mysqld]` configuration, so the runtime increase is lost. To retain the change, the relevant option-file value must be updated and the server restarted, or an appropriate persistent configuration mechanism must be used where available.

Session variables are not persistent across server restarts. The unqualified `SET sort_buffer_size=2000000` statement sets the value for the current session. Session system-variable values exist only for that connection and disappear when it ends; a server restart necessarily terminates all sessions.

The query benefited from tmp_table_size increase. `tmp_table_size` establishes the maximum size for internal in-memory temporary tables; when a query needs an internal temporary table, a larger effective limit can avoid conversion to an on-disk temporary table and thereby improve performance. The global setting supplies the default for newly created application sessions, which explains why the improvement can be observed before the restart. See the MySQL 5.7 documentation for tmp_table_size and the rules for using system variables.

QUESTION NO: 7

What two statements are true regarding FLUSH TABLES FOR EXPORT? (Choose two.)

A. It can be used to export TEMPORARY tables.

B. Table only exports when the table has its own tablespace.

C. The InnoDB Storage engine must be used for the table being exported.

D. It is the safest way to extract tables from the shared tablespace.

E. Partitioned tables are not supported.

ANSWER: B C

Explanation:

`FLUSH TABLES ... FOR EXPORT` is an InnoDB transportable-tablespace operation. Therefore, “The InnoDB Storage engine must be used for the table being exported” is correct: the command places the specified InnoDB table into a consistent state for copying, writes the metadata file needed for import, and keeps the table read-only until it is unlocked. This enables the table data and associated metadata to be copied safely for later import into another InnoDB instance.

“Table only exports when the table has its own tablespace” is also correct. The transportable-tablespace workflow operates on a table’s individual `.ibd` file, which requires an InnoDB file-per-table tablespace. A table stored in the shared InnoDB system tablespace does not have a self-contained table tablespace file that can be transported through this process. Before copying the files, the exporting session must keep the export lock in place; after the copy completes, `UNLOCK TABLES` ends the export state. Oracle’s MySQL documentation describes both the required InnoDB engine and the file-per-table prerequisite for table export and import: MySQL 5.7 Reference Manual: Importing InnoDB Tables and FLUSH TABLES ... FOR EXPORT syntax.

QUESTION NO: 8

You inherited a busy InnoDB OLTP Instance with 100 schemas and 100 active users per schema.

Total dataset size is 200G with an average schema size of 2G.

The data is transient and is not backed up and can be repopulated easily.

Performance and responsiveness of the DB is paramount.

The query pattern for the DB instance is split 90/10 read/write.

DB host is dedicated server with 256G RAM and 64 cores.

One of your colleagues made some recent changes to the system and users are now complaining of performance impacts.

Which four configuration file edits might your colleague have performed to cause the negative DB performance? (Choose four.)

A. table_open_cache = 64

B. innodb_buffer_pool_size=200G

C. innodb_flush_log_at_trx_commit = 1

D. sync_binlog=10

E. innodb_flush_method=O_DIRECT

F. tmp_table_size=2G

G. query_cache_type = 1

ANSWER: A C F G

Explanation:

`table_open_cache = 64` is far too small for a busy server hosting many schemas and active users. It forces frequent table-cache misses and repeated opening of table files, adding avoidable metadata and file-operation overhead. MySQL documents the table cache and its impact on table-open activity at MySQL 5.7 Table Cache.

`innodb_flush_log_at_trx_commit = 1` requests a redo-log flush to durable storage for every committed transaction. This maximizes transactional durability, but the associated synchronous I/O can materially increase commit latency. Since this workload’s data is disposable and responsiveness is paramount, that durability setting can impose needless write-path cost.

`tmp_table_size = 2G` permits very large internal temporary tables to remain in memory. With substantial concurrent activity, memory used by temporary-table operations can become excessive, creating memory pressure and potentially causing swapping or other resource contention.

`query_cache_type = 1` enables the MySQL query cache. In a concurrent OLTP workload, query-cache maintenance and invalidation caused by writes can serialize activity and become a contention bottleneck. The MySQL manual specifically describes the query cache's mutex contention and scalability limitations: MySQL 5.7 Query Cache Operation.

QUESTION NO: 9

You are creating a strategy for backing up MySQL using a cold binary backup. The MySQL instance is a replication master with global transaction identifiers (GTIDs) enabled and it uses Transparent Data Encryption (TDE). Other than the configuration required to make the instance a replication master and enabled GTIDs and TDE, the instance is using all default settings.

The requirements for the backup are:

It must be possible to rebuild the instance using the backup.

It must be verified.

It must allow for a catastrophic hardware failure.

Which four steps must be included in the backup strategy? (Choose four.)

A. Include the keyring data and/or configuration in the backup.

B. Restore the backup to a clean MySQL instance.

C. Copy the backup to a remote host.

D. Include the MySQL socket file in the backup.

E. Include the ibtmp1 file in the backup.

F. Include the relay logs in the backup.

G. Include the operating system disk encryption key in the backup.

H. Shut down MySQL cleanly before copying the database files.

ANSWER: A B C H

Explanation:

A cold binary backup must begin with a clean shutdown of the MySQL server before its database files are copied. This ensures that InnoDB files are internally consistent and can be used to reconstruct the server without requiring crash recovery against files that may have been copied while they were changing. MySQL documents that a cold backup is performed after the server has been stopped; see MySQL 5.7 Backup Types.

Because Transparent Data Encryption is enabled, the backup must include the keyring material or the keyring configuration needed to access it. Encrypted tablespace data is not usable unless the corresponding encryption keys remain available. The keyring must therefore be retained securely and independently from the protected database files, as described in MySQL 5.7 Keyring.

Verification requires an actual restore to a clean MySQL instance and validation that the restored server starts and can access its data. Merely confirming that backup files exist does not prove recoverability. Finally, a copy on a remote host provides geographic and hardware separation from the source system, allowing recovery after loss of the original server or its local storage.

QUESTION NO: 10

You have set innodb_stats_auto_recalc to OFF.

For tables using persistent statistics, what is the outcome of this change?

A. InnoDB no longer automatically updates index statistics after a CREATE TABLE statement.

B. InnoDB no longer automatically updates index statistics after the table structure is altered.

C. InnoDB no longer automatically updates index statistics after 10% of the rows in a table change.

D. InnoDB no longer automatically updates index statistics after an ANALYZE TABLE statement.

ANSWER: C

Explanation:

InnoDB no longer automatically updates index statistics after 10% of the rows in a table change. The `innodb_stats_auto_recalc` system variable controls whether InnoDB automatically recalculates persistent optimizer statistics after substantial modifications to a table. When it is enabled, InnoDB monitors changes and schedules an automatic recalculation when the number of modified rows exceeds 10% of the table's row count since the preceding statistics calculation. Setting the variable to `OFF` disables that automatic threshold-based recalculation for tables that use persistent statistics.

This setting is useful when an administrator needs to control when statistics are refreshed, such as during bulk data loads or periods of heavy write activity. Existing persistent statistics remain stored and available to the optimizer; the change specifically prevents the automatic refresh that would otherwise be triggered by the modification threshold. Statistics can still be refreshed deliberately by an administrator when appropriate. Oracle documents this behavior for persistent InnoDB statistics and the `innodb_stats_auto_recalc` variable in the MySQL 5.7 Reference Manual: Persistent Optimizer Statistics and the innodb_stats_auto_recalc system-variable reference.

QUESTION NO: 11

You are asked to examine user accounts and find:

```
mysql> select user,host,plugin,authentication_string from user;
+--------+-----------+-----------------------+-------------------------------------------
---+
| user   | host      | plugin                |
authentication_string                      |
+--------+-----------+-----------------------+-------------------------------------------
---+
| root   | localhost | mysql_native_password |
*2470C0C06DEE42FD1618BB99005ADCA2EC9D1E19 |
| .....
| bob    | %         | mysql_native_password |
  |
|        | localhost | mysql_native_password |
*2DCA2EC9D470CE42FD1618BB99005ADCA218BB99 |
+--------+-----------+-----------------------+-------------------------------------------
---+
10 rows in set (0.01 sec)
```

Which two statements will best secure this environment? (Choose two.)

A. DROP USER ''@'localhost';

B. ALTER USER ‘’@’localhost’ PASSWORD = ‘secret_password’;

C. ALTER USER 'bob'@'%' IDENTIFIED BY 'secret_password';

D. ALTER USER ‘root’@’localhost’ ACCOUNT LOCK;

E. ALTER USER bob PASSWORD = ‘secret_password’;

F. REVOKE ALL PRIVILEGES FROM ‘’@’localhost’;

ANSWER: A C

Explanation:

`DROP USER ''@'localhost';` is appropriate because the account list shows an anonymous account associated with the local host. Anonymous accounts permit connections without a user name and can cause MySQL to authenticate a connection as the anonymous account rather than as an intended named account. Removing the account entirely is the most direct way to eliminate that unnecessary authentication entry and its potential access path. The MySQL reference manual documents that `DROP USER` removes the specified account and its associated privileges.

`ALTER USER 'bob'@'%' IDENTIFIED BY 'secret_password';` is appropriate because the displayed `bob` account accepts connections from any host, as denoted by the `%` host pattern. Setting a password with `ALTER USER ... IDENTIFIED BY` ensures this broadly reachable account has explicit password authentication rather than a blank or otherwise insecure credential. This syntax changes authentication information for the exact `'bob'@'%'` account, which is important because MySQL accounts are identified by both user name and host. See the MySQL 5.7 documentation for DROP USER and ALTER USER.

QUESTION NO: 12

Which three statements correctly describe MySQL InnoDB Cluster? (Choose three.)

A. The cluster can be operated in multimaster mode with conflict detection for DML statements.

B. All MySQL client programs and connectors can be used for executing queries.

C. It provides fully synchronous replication between the nodes.

D. There is support for automatic failover when one node fails.

E. The data is automatically shared between the nodes.

F. Each query will be executed in parallel across the nodes.

ANSWER: A B D

Explanation:

MySQL InnoDB Cluster can run MySQL Group Replication in either single-primary or multi-primary mode. In multi-primary mode, more than one member can accept updates. Group Replication uses transaction certification to detect conflicts among concurrently submitted transactions, ensuring that conflicting writes are identified before they are applied consistently by the group. This supports the statement that the cluster can be operated in multimaster mode with conflict detection for DML statements.

InnoDB Cluster is built from standard MySQL Server instances and remains compatible with the regular MySQL client protocol. Applications can therefore use normal MySQL client programs and supported connectors to execute queries. MySQL Router is commonly placed between applications and the cluster to provide transparent routing to appropriate members, without requiring an application to use a special proprietary query interface.

Automatic failover is a core availability capability of an InnoDB Cluster deployment. Group Replication detects member failures and maintains membership and quorum, while MySQL Router obtains cluster metadata and redirects new connections when the primary role changes. This allows applications configured through Router to continue reaching an available writer after a primary-node failure. See the MySQL InnoDB Cluster documentation and the MySQL Group Replication documentation.

QUESTION NO: 13

These details are shown when logged in to an account:

```
mysql> SELECT USER(), CURRENT_USER();
+-------------------------------+-------------------------------+
| USER ()                       | CURRENT_USER()                |
+-------------------------------+-------------------------------+
| robert@localhost              | employee@localhost            |
+-------------------------------+-------------------------------+
mysql> SHOW GLOBAL VARIABLES LIKE 'check_proxy_user';
+-------------------------------+-------------+
| Variable_name                 |  Value      |
+-------------------------------+-------------+
| check_proxy_users             | OFF         |
+-------------------------------+-------------+
1 row in set (0.00 sec)
```

Which set of statements would match the accounts shown?

A. mysql> CREATE USER 'employee'@'localhost' IDENTIFIED BY 'more_secrets';
mysql> CREATE USER ''@'' IDENTIFIED BY 'valid_password' WITH PROXY 'employee'@'localhost';

B. mysql> CREATE USER 'employee'@'localhost' IDENTIFIED BY 'more_secrets';
mysql> GRANT PROXY ON 'employee'@'localhost' TO 'robert'@'localhost';

C. mysql> CREATE USER 'robert'@'localhost' IDENTIFIED BY 'secret_password';
mysql>CREATE USER 'employee'@'localhost' IDENTIFIED BY 'more_secrets';

D. mysql> CREATE USER ''@'' IDENTIFIED WITH authentication_pam ACCOUNT LOCK;
mysql> CREATE USER 'employee'@'localhost' IDENTIFIED BY 'more_secrets'; mysql> GRANT PROXY ON 'employee'@'localhost' TO ''@'';

ANSWER: D

Explanation:

The statements beginning `CREATE USER ''@'' IDENTIFIED WITH authentication_pam ACCOUNT LOCK` create the anonymous proxy-authentication account shown by the account details, using the PAM authentication plugin. This lets MySQL authenticate a client through the configured external PAM service rather than through a native MySQL password stored for a named MySQL account. The separate `'employee'@'localhost'` account is the account whose privileges are ultimately used, and it is configured with the password authentication information shown.

The essential relationship is established by `GRANT PROXY ON 'employee'@'localhost' TO ''@''`. A PROXY grant permits the authenticated proxy account to assume the identity and privileges of the employee account. Consequently, MySQL can retain the original authenticated client identity while using `'employee'@'localhost'` as the effective account for privilege checking, which is the behavior represented by the displayed account information. The account-lock attribute is part of the anonymous account definition shown and does not change the fact that the required proxy relationship is defined by the PROXY grant.

Oracle documents proxy-user identity handling and PROXY grants in the MySQL 5.7 Proxy Users documentation. The account-creation syntax, authentication-plugin selection, and account-locking clauses are documented in the CREATE USER statement reference.

QUESTION NO: 14

Consider the two partial outputs of the SHOW GLOBAL VARIABLES command from a master and slave server:

Master:

Slave:

There is a problem with the slave replicating from the master. Which statement describes the cause of the problem?

A. The log_bin variable is set to OFF on the slave.

B. server_id is not unique.

C. The max_connections variable on the slave needs to be increased.

D. The shared_memory_base_name variable must match the master.

E. The version of the slave is newer that the version of the master.

ANSWER: B

Explanation:

`server_id is not unique.` is the cause because every MySQL server participating in replication must have a distinct nonzero server ID. The source records its own server ID in each binary-log event, and a replica uses server IDs to identify event origin and prevent replication loops. If the source and replica have the same `server_id`, the replica can treat events originating on the source as its own events and skip them, preventing normal replication of the source's changes. A unique value must therefore be configured for each server, typically in the server's option file using `server-id=N`, followed by a restart if the configuration is changed there. For a standard source-to-replica topology, binary logging need not be enabled on the replica; it is required on the source and is only needed on a replica when that replica must write received transactions to its own binary log, such as for chained replication or point-in-time recovery purposes. Oracle's replication documentation explicitly states that every replication server must be assigned a unique server ID and explains its use in identifying replicated events. See MySQL 5.7 server_id replication option and MySQL 5.7 replication setup.

QUESTION NO: 15

The MySQL instance is a default RPM installation on a Linux server.

Where are the errors written?

A. in the /var/lib/mysql/hostname.log file

B. no logging enabled by default

C. in the /var/log/mysqld.err file

D. in the syslog daemon on the server

E. in the /var/log/mysqld.log file

ANSWER: E

Explanation:

For a default MySQL 5.7 installation using Oracle-provided RPM packages on Linux, the server error log is written to `/var/log/mysqld.log`. This log records significant server events, including startup and shutdown information, initialization messages, diagnostic output, and errors encountered while processing connections or executing server operations. It is therefore the first log an administrator should inspect when the MySQL service does not start normally or reports an operational problem.

The default RPM configuration is important here because error-log locations can be changed with the `log_error` server system variable or an option-file setting, and other installation methods may use different defaults. In the stated default RPM scenario, however, `/var/log/mysqld.log` is the documented location. Administrators typically need appropriate operating-system privileges to read this file, and log rotation may be managed by the installed package configuration. Oracle documents the RPM installation defaults in the MySQL 5.7 RPM installation guide; general error-log behavior and configuration are covered in the MySQL error log documentation.

QUESTION NO: 16

The /myfolder/my.cnf file has option set:

[mysqld] skip-log-bin

/myfolder2/my.cnf has this option set:

[mysqld]

log-bin = /valid/path/to/mysqlbinlog

All mentioned paths are accessible to the account that you are currently using. Assume that any other options mentioned in either file are valid and legal option definitions.

You start an instance by using this command line:

mysqld --defaults-file=/myfolder/my.cnf --defaults-extra-file=/ myfolder2/my.cnf

What is the outcome?

A. MySQL starts and Binary Logging is enabled.

B. MySQL fails to start due to the conflicting options in the configuration files.

C. MySQL fails to start due to conflicting options on the command line.

D. MySQL starts but Binary Logging is disabled.

E. MySQL fails to start because `--defaults-extra-file` is parsed with `/` as its file name.

ANSWER: E

Explanation:

MySQL fails to start because the space after `--defaults-extra-file=/` changes how the command line is parsed. The option's argument is the single character `/`, not `/myfolder2/my.cnf`. Thus, while processing option files before normal server startup, MySQL attempts to use the root directory as the required extra defaults file. A directory is not a readable option file, so defaults processing terminates with a fatal error. The following token, `myfolder2/my.cnf`, is not joined to the preceding option value; shell whitespace separates command-line arguments. The intended command would require no whitespace between the equals sign and the file name: `--defaults-extra-file=/myfolder2/my.cnf`. MySQL documents that `--defaults-extra-file` specifies an additional option file and that an explicitly named option file which cannot be opened causes an error. See the MySQL 5.7 option-file documentation at https://dev.mysql.com/doc/refman/5.7/en/option-files.html and the server option reference at https://dev.mysql.com/doc/refman/5.7/en/server-options.html.

QUESTION NO: 17

You want to dump only data from the userdata table.

Which mysqldump command argument is required to accomplish this?

A. - -no-create-info to skip writing CREATE TABLE statements

B. - -single-transaction as this obtains a consistent view of data only

C. - -data-only as this specifies that only data is to be dumped

D. - -table=userdata in order to dump only the data from the userdata table

ANSWER: A

Explanation:

`--no-create-info` to skip writing CREATE TABLE statements is required because it directs `mysqldump` not to include table-creation statements in its output. Consequently, the dump contains the table's data statements, such as `INSERT` statements, rather than schema-definition SQL. To limit the dump to the named table, the table name is supplied as a

positional argument after the database name; for example: `mysqldump --no-create-info mydatabase userdata`. The option is therefore the required mysqldump argument for the “data only” aspect of the request, while `userdata` identifies the table to dump. Oracle’s MySQL 5.7 manual documents `--no-create-info` as suppressing all CREATE TABLE statements, and documents the normal command syntax in which one or more table names follow the database name. See the MySQL 5.7 mysqldump reference and the mysqldump output-definition documentation.

QUESTION NO: 18

Consider these global status variables:

```
mysql> SELECT *
   FROM performace_schema.global_status
WHERE VARIABLE_NAME LIKE '%connection%'
        OR VARIABLE_NAME LIKE '%thread%';
+-----------------------------------------------------------------+-----------------------------------+
| VARIABLE_NAME                                                   |        VARIABLE_VALUE             |
+-----------------------------------------------------------------+-----------------------------------+
| Connection_errors_accept                                        | 0                                 |
| Connection_errors_internal                                      | 6                                 |
| Connection_errors_max_connections                               | 0                                 |
| Connections_errors_peer_address                                 | 0                                 |
| Connection_errors_select                                        | 0                                 |
| Connection_errors_tcpwrap                                       | 0                                 |
| Connections                                                     | 510                               |
| Delayed_insert_threads                                          | 0                                 |
| Max_used_connections                                            | 145                               |
| Max_used_connections_time                                       | 2018-03-22  14:54:06              |
| Performance_schema_thread_classes_lost                          | 0                                 |
| Performance_schema_thread_instances_lost                        | 0                                 |
| Slow_launch_threads                                             | 0                                 |
| Threads_cached                                                  | 6                                 |
| Threads_connected                                               | 140                               |
| Threads_created                                                 | 155                               |
| Threads_running                                                 | 14                                |
+-----------------------------------------------------------------+-----------------------------------+
17 rows in set (0.00 sec)
```

Which two conclusions can be made from the output? (Choose two.)

A. There are 140 Performance Schema threads at the time of the output.

B. There are 510 connections to MySQL at the time of the output.

C. The thread cache has been configured with thread_cache_size set to at least 6.

D. There are more connections being idle than executing queries.

E. All max_connections were in use at 2018-03-22 14:54:06

ANSWER: C D

Explanation:

The output supports the conclusion that the thread cache has been configured with `thread_cache_size` set to at least 6. The `Threads_cached` status variable is the number of threads currently held in the thread cache for reuse. Since six threads are presently cached, the configured cache can accommodate at least that many threads. MySQL uses this cache to avoid the overhead of repeatedly creating and destroying a thread as clients connect and disconnect.

It also supports the conclusion that there are more connections being idle than executing queries. `Threads_connected` is the current number of open client connections, whereas `Threads_running` is the number of threads that are not sleeping—normally those actively executing work. The running threads are included among connected threads; therefore, when `Threads_connected` exceeds `Threads_running`, the difference represents connected sessions that are idle or otherwise sleeping at that instant. These are point-in-time status counters, so this conclusion describes the server state when the status output was obtained. See the MySQL 5.7 documentation for server status variables and the thread_cache_size system variable.

QUESTION NO: 19

Consider a replication setup where a snapshot of the replication status shows:

```
mysql> SELECT CHANNEL_NAME, SERVICE_STATE, LAST_HEARTBEAT_TIMESTAMP,
RECEIVED_TRANSACTION_SET
         FROM performance_schema.replication_connection_status;
+--------------+---------------+----------------------------+--------------------------
------------------------+
| CHANNEL_NAME | SERVICE_STATE | LAST_HEARTBEAT_TIMESTAMP |
RECEIVED_TRANSACTION_SET                          |
+--------------+---------------+----------------------------+--------------------------
------------------------+
| melbourne    | ON            | 2018-05-08 14:39:50        | aa23540f-5272-11e8-8816-
08002715584a:61054-148465 |
| sydney       | ON            | 2018-05-08 14:55:01        | 4ab8feff-5272-11e8-9320-
08002715584a:93604-201574 |
+--------------+---------------+----------------------------+--------------------------
------------------------+
2 rows in set (0.00 sec)

mysql> SELECT CHANNEL_NAME, WORKER_ID, THREAD_ID, SERVICE_STATE, LAST_SEEN_TRANSACTION
         FROM performance_schema.replication_applier_status_by_worker;
+--------------+-----------+-----------+---------------+----------------------------
---------------+
| CHANNEL_NAME | WORKER_ID | THREAD_ID | SERVICE_STATE |
LAST_SEEN_TRANSACTION                     |
+--------------+-----------+-----------+---------------+----------------------------
---------------+
| melbourne    |         1 |        29 | ON            | aa23540f-5272-11e8-8816-
08002715584a:128088 |
| melbourne    |         2 |        33 | ON            | aa23540f-5272-11e8-8816-
08002715584a:128089 |
| melbourne    |         3 |        35 | ON            | aa23540f-5272-11e8-8816-
08002715584a:128090 |
| melbourne    |         4 |        37 | ON            | aa23540f-5272-11e8-8816-
08002715584a:128087 |
| sydney       |         1 |        30 | ON            | 4ab8feff-5272-11e8-9320-
08002715584a:201573 |
| sydney       |         2 |        32 | ON            | 4ab8feff-5272-11e8-9320-
08002715584a:201574 |
| sydney       |         3 |        34 | ON            | 4ab8feff-5272-11e8-9320-
08002715584a:201570 |
| sydney       |         4 |        36 | ON            | 4ab8feff-5272-11e8-9320-
08002715584a:201513 |
+--------------+-----------+-----------+---------------+----------------------------
---------------+
8 rows in set (0.00 sec)
```

Which statement is true?

A. The melbourne channel is resolving a GTID conflict.

B. The sydney channel is keeping up with the received events.

C. The melbourne channel is keeping up with the received events.

D. The last event received from the sydney channel was at 2018-05-08 14:55:01.

E. The replication slave uses multisource replication with eight sources.

ANSWER: E

Explanation:

The replication slave uses multisource replication with eight sources. In MySQL multi-source replication, a single replica can receive and process transactions from multiple independent source servers. MySQL identifies each source connection through a replication channel name, and replication status is reported separately for each channel. A status snapshot that contains eight channel-specific replication-status entries therefore represents eight configured source-to-replica replication channels. Each channel has its own receiver and applier state, relay-log position, source log coordinates, and error information, allowing the replica to monitor and administer every source independently.

This is the purpose of MySQL's multi-source replication architecture: consolidating data from several sources on one replica while retaining distinct channel configuration and status. The number of sources is determined by the number of channels shown in the status output, rather than by the number of events or timestamps displayed for any individual channel. MySQL documents that a replica in a multi-source topology creates a separate channel for each source and that channel-specific status can be examined with replication-status statements. See the MySQL 5.7 Multi-Source Replication documentation and the SHOW SLAVE STATUS reference.

QUESTION NO: 20

An existing master-slave setup is currently using a delayed replication of one hour. The master has crashed and the slave must be "rolled forward" to provide all the latest data.

The SHOW SLAVE STATUS indicates these values:

RELAY_LOG_FILE=hostname-relay-bin.00004
RELAY_LOG_POS=1383

Which command set would make the slave current?

A. STOP SLAVE; SET GLOBAL master_delay=0; START SLAVE;

B. STOP SLAVE; CHANGE MASTER TO RELAY_LOG_FILE = 'hostname-relay-bin.00004',RELAY_LOG_POS = 1383;

C. STOP SLAVE; CHANGE MASTER TO MASTER_DELAY=0; START SLAVE;

D. STOP SLAVE; CHANGE MASTER TO MASTER_DELAY=0; RELAY_LOG_FILE = 'hostname-relay-bin.00004', RELAY_LOG_POS = 1383;

ANSWER: C

Explanation:

`STOP SLAVE; CHANGE MASTER TO MASTER_DELAY=0; START SLAVE;` is correct because delayed replication is controlled by the replica's `MASTER_DELAY` setting. A value of 3600 causes the SQL/applier thread to intentionally wait one hour before executing events that have already been received into the relay logs. Setting `MASTER_DELAY=0` removes that intentional delay. After replication is started again, the SQL thread can apply the pending relay-log events immediately, allowing the replica to catch up as far as the events it had received before the source failure permit.

The relay-log filename and position shown by `SHOW SLAVE STATUS` identify the replica's current relay-log execution state; they do not need to be manually reset merely to remove a configured delay. Preserving that state is important because it lets replication continue applying the already queued transactions in normal order. Stopping the replica before issuing `CHANGE MASTER TO` follows the required administrative procedure for changing replication connection settings, and starting it afterward resumes processing. Oracle documents `MASTER_DELAY` as the option used to configure delayed replication and

specifies zero as no delay. See the MySQL 5.7 delayed replication documentation and the CHANGE MASTER TO reference.