How to backup and restore MySQL databases using the mysqldump command

In this article, I am going to explain different ways to generate the backup in the MySQL database server. As we know, data is a valuable asset to the organization. As database administrators, it is our primary and crucial job to keep the data available and safe. If the system or data center fails, database corruption, and data loss, we must be able to recover it within the defined SLA.

Different database platforms provide various methods to generate the backup and restore the database. Many vendors provide state-of-the-art software and hardware solutions that can help to back up the database, and it can restore the database within the defined RTO and RPO.

Here, we are not going to discuss any third-party vendor's backup solutions. I am going to explain the native methods that are used to generate the backup of the database. We can generate the backup of the MySQL database using any of the following methods:

1. Generate the backup using ***mysqldump*** utility
2. Generate Incremental backups using Binary Log
3. Generate backups using the Replication of Slaves

In this article, I am going to explain how we can use ***mysqldump*** to generate the backup of the MySQL database.

Generate backup using mysqldump utility

Mysqldump is a command-line utility that is used to generate the logical backup of the MySQL database. It produces the SQL Statements that can be used to recreate the database objects and data. The command can also be used to generate the output in the XML, delimited text, or CSV format.

This command is easy to use, but the only problem that occurs while restoring the database. As I mentioned, when we generate a backup of the MySQL database, it creates a backup file that contains SQL commands that are necessary to rebuild or restore the database. Now, when we restore the database, the command executes all the SQL Statements to create tables and insert the data. If you have a large database, then the restoration process takes a long time to complete.

Note: By default, mysqldump command does not dump the **information_schema** database, **performance_schema,** and MySQL Cluster **ndbinfo** database.

If you want to include the **information_schema** tables, you must explicitly specify the name of the database in the ***mysqldump*** command, also include the ***—skip-lock-tables*** option.

There are lots of options and features that can be used with ***mysqldump***. You can view the complete list of options here. I am going to some of the basic features. Following is the syntax of the ***mysqldump*** utility.

```
mysqldump -u [user name] -p [password] [options] [database_name] [tablename] >
[dumpfilename.sql]
```

The parameters are as following:

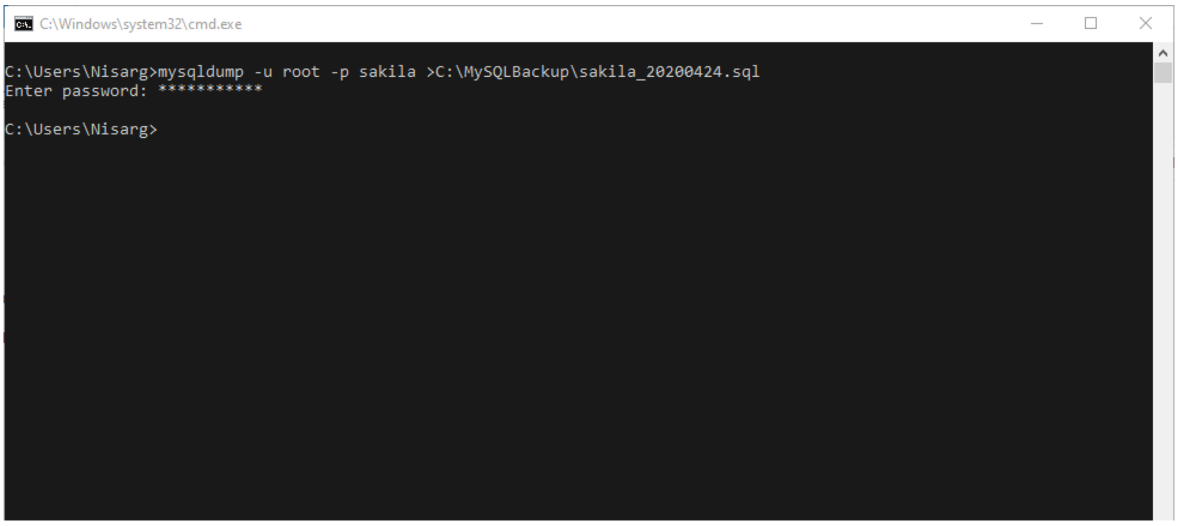
1. **-u [user_name]:** It is a username to connect to the MySQL server. To generate the backup using ***mysqldump***, '**Select**' to dump the tables, '**Show View**' for views, '**Trigger**' for the triggers. If you are not using **—*single-transaction*** option, then '**Lock Tables**' privileges must be granted to the user
2. **-p [password]:** The valid password of the MySQL user
3. **[option]:** The configuration option to customize the backup
4. **[database name]:** Name of the database that you want to take backup
5. **[table name]:** This is an optional parameter. If you want to take the backup specific tables, then you can specify the names in the command
6. **"<" OR ">":** This character indicates whether we are generating the backup of the database or restoring the database. You can use "**>**" to generate the backup and "**<**" to restore the backup
7. **[dumpfilename.sql]:** Path and name of the backup file. As I mentioned, we can generate the backup in XML, delimited text, or SQL file so we can provide the extension of the file accordingly

Generate the backup of a single database

For example, you want to generate the backup of the single database, run the following command. The command will generate the backup of the "**sakila**" database with structure and data in the **sakila_20200424.sql** file.

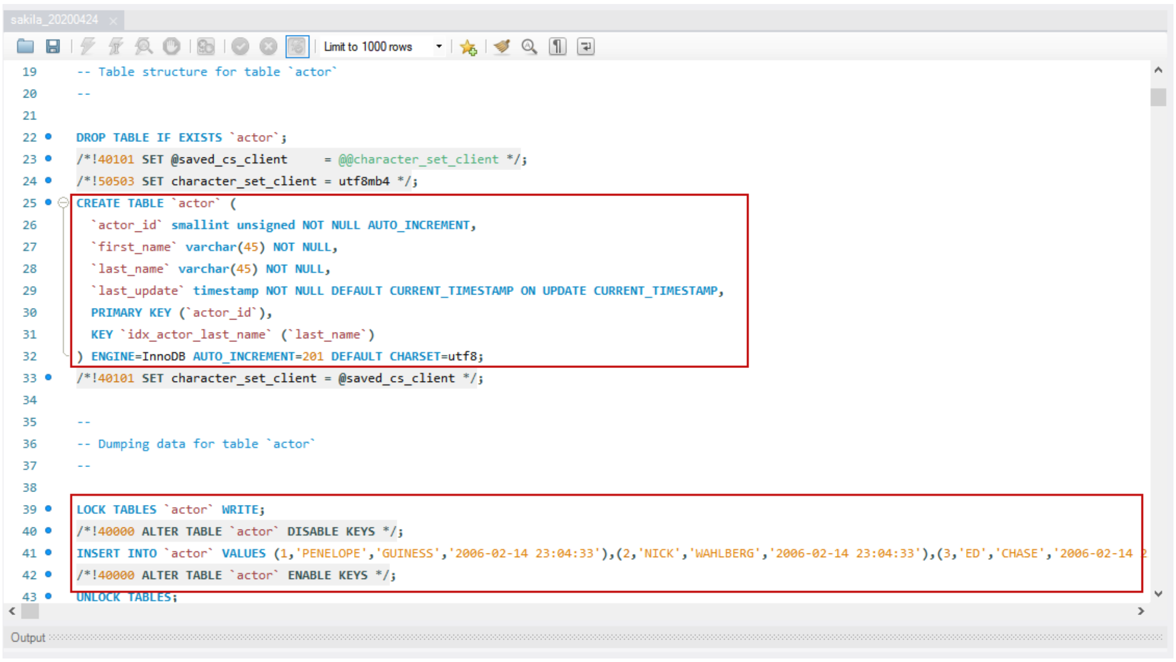
```
mysqldump -u root -p sakila > C:\MySQLBackup\sakila_20200424.sql
```

When you run this command, it prompts for the password. Provide the appropriate password. See the following image:



```
C:\Users\Nisarg>mysqldump -u root -p sakila >C:\MySQLBackup\sakila_20200424.sql
Enter password: ***********

C:\Users\Nisarg>
```

Once backup generated successfully, let us open the backup file to view the content of the backup file. Open the backup location and double-click on the "**sakila_20200424.sql**" file.



```
-- Table structure for table `actor`
--

DROP TABLE IF EXISTS `actor`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `actor` (
  `actor_id` smallint unsigned NOT NULL AUTO_INCREMENT,
  `first_name` varchar(45) NOT NULL,
  `last_name` varchar(45) NOT NULL,
  `last_update` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (`actor_id`),
  KEY `idx_actor_last_name` (`last_name`)
) ENGINE=InnoDB AUTO_INCREMENT=201 DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `actor`
--

LOCK TABLES `actor` WRITE;
/*!40000 ALTER TABLE `actor` DISABLE KEYS */;
INSERT INTO `actor` VALUES (1,'PENELOPE','GUINESS','2006-02-14 23:04:33'),(2,'NICK','WAHLBERG','2006-02-14 23:04:33'),(3,'ED','CHASE','2006-02-14 
/*!40000 ALTER TABLE `actor` ENABLE KEYS */;
UNLOCK TABLES;
```

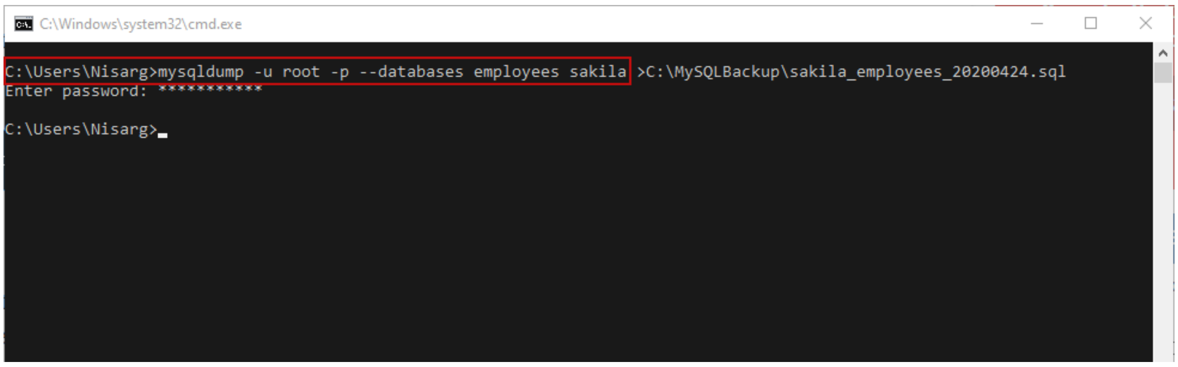
As you can see in the above image, the backup file contains the various T-SQL statements that can be used to re-create the objects.

Generate the backup of multiple databases or all the databases

For example, you want to generate a backup of more than one database. You must add the **—*databases*** option in the ***mysqldump*** command. The following command will generate the back-up of "**sakila**" and "**employees**" database with structure and data.

```
mysqldump -u root -p --databases sakila employees > C:\MySQLBackup\sakila_em-
ployees_20200424.sql
```

See the following image:

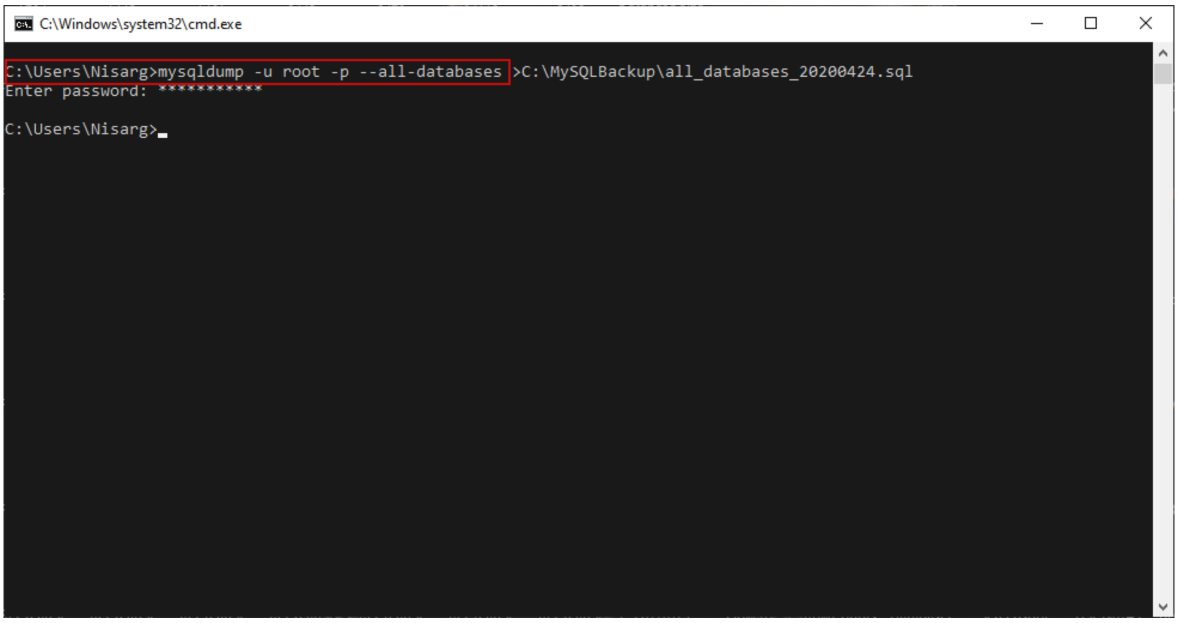


```
C:\Users\Nisarg>mysqldump -u root -p --databases employees sakila >C:\MySQLBackup\sakila_employees_20200424.sql
Enter password: ***********

C:\Users\Nisarg>_
```

Similarly, if you want to generate the backup of all the databases, you must use ***–all-databases*** option in the ***mysqldump*** command. The following command will generate the backup of all databases within MySQL Server.

```
mysqldump -u root -p --all-databases > C:\MySQLBackup\all_databases_20200424.sq
l
```

See the following image:



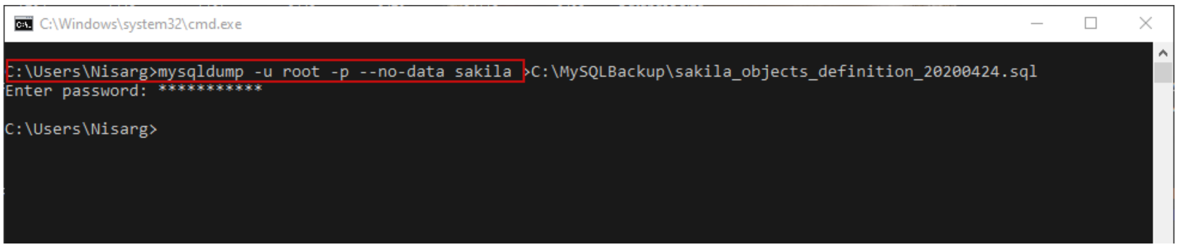
```
C:\Users\Nisarg>mysqldump -u root -p --all-databases >C:\MySQLBackup\all_databases_20200424.sql
Enter password: ***********

C:\Users\Nisarg>_
```

Generate the backup of database structure

If you want to generate the backup of the database structure, then you must use the ***–no-data*** option in the ***mysqldump*** command. The following command generates the backup of the database structure of the **sakila** database.

```
mysqldump -u root -p --no-data sakila > C:\MySQLBackup\sakila_objects_defini-
tion_20200424.sql
```

See the following image:



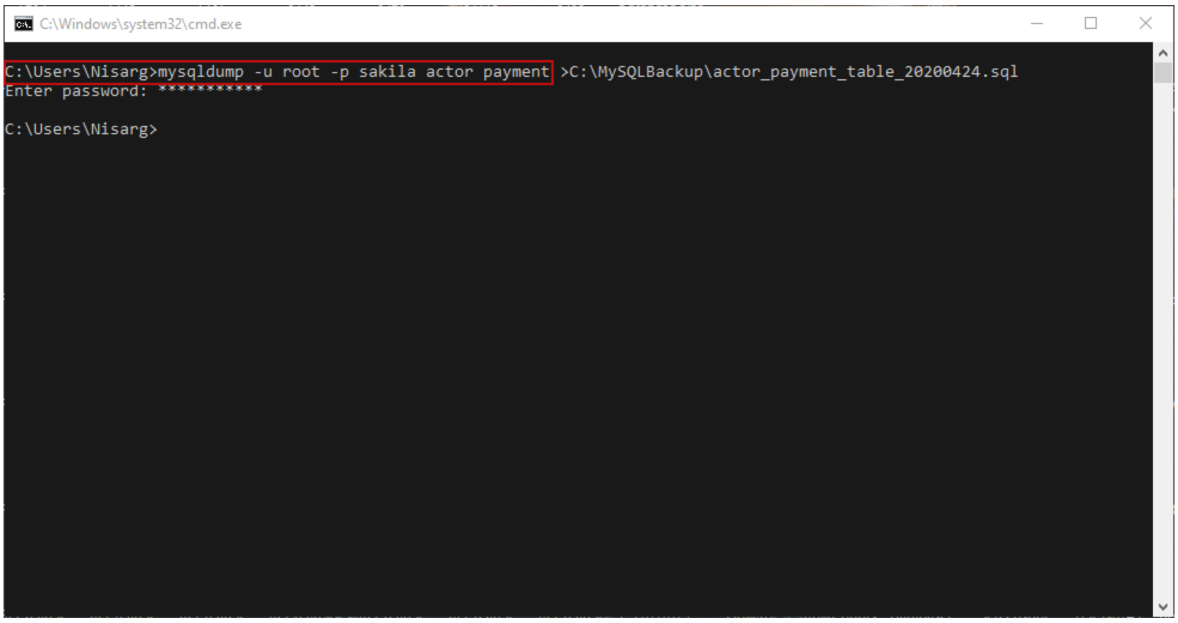
```
C:\Users\Nisarg>mysqldump -u root -p --no-data sakila >C:\MySQLBackup\sakila_objects_definition_20200424.sql
Enter password: ***********

C:\Users\Nisarg>
```

Generate the backup of a specific table

If you want to generate the backup of a specific table, then you must specify the name of the tables after the name of the database. The following command generates the backup of the **actor** table of the **sakila** database.

```
mysqldump -u root -p sakila actor payment > C:\MySQLBackup\actor_paymen-
t_table_20200424.sql
```

If you want to generate the backup of more than one tables, than you must separate the names of the tables with space, the following command generates the backup of **actor** and **payment** table of **sakila** database.



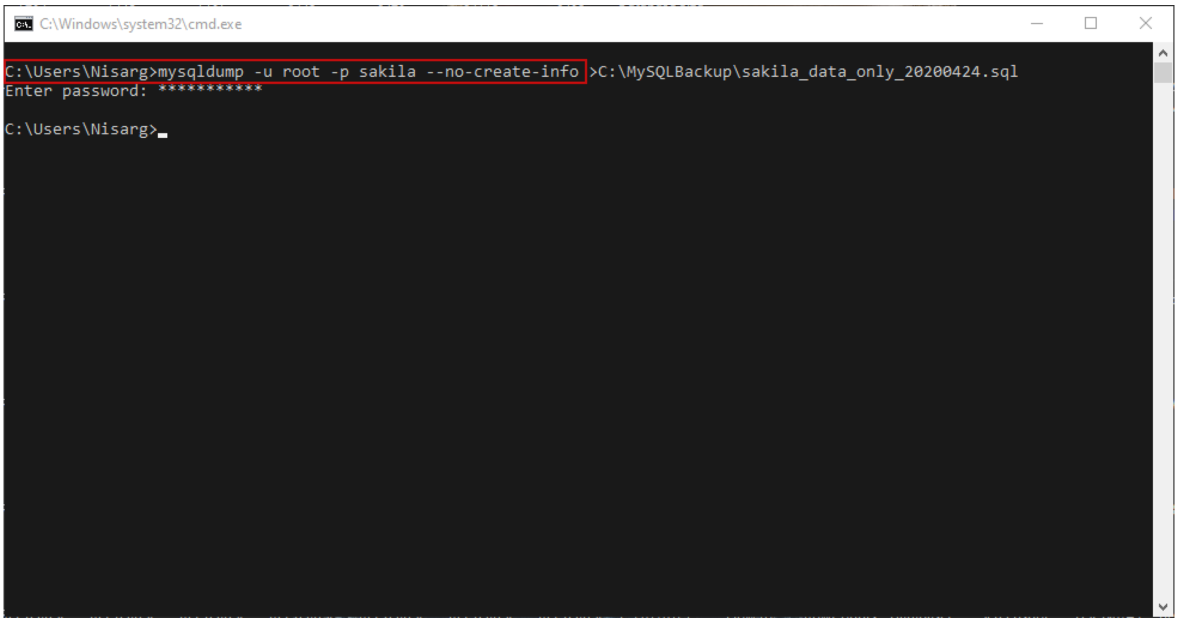
```
C:\Users\Nisarg>mysqldump -u root -p sakila actor payment >C:\MySQLBackup\actor_payment_table_20200424.sql
Enter password: ************

C:\Users\Nisarg>
```

Generate the backup of database data

If you want to generate the backup of the data without the database structure, then you must use the ***–no-create-info*** option in the ***mysqldump*** command. The following command generates the backup of data of the **sakila** database.

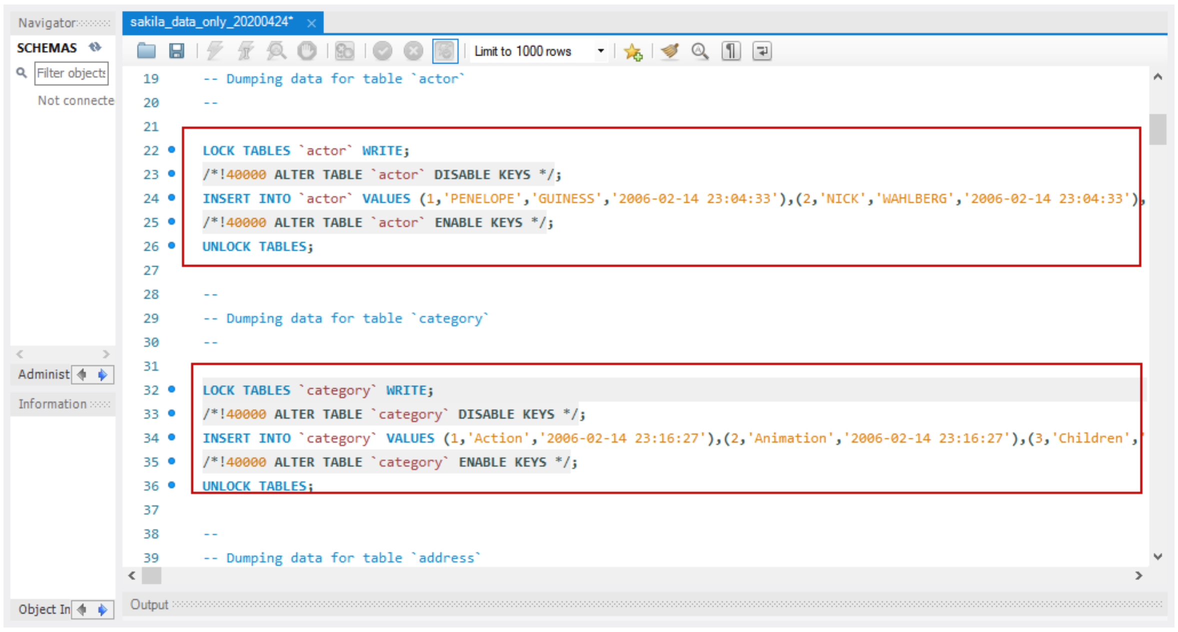
```
mysqldump -u root -p sakila --no-create-info > C:\MySQLBackup\sakila_data_on-
ly_20200424.sql
```

See the following image:



```
C:\Users\Nisarg>mysqldump -u root -p sakila --no-create-info >C:\MySQLBackup\sakila_data_only_20200424.sql
Enter password: ***********

C:\Users\Nisarg>_
```

Let us view the content of the backup file.



```
-- Dumping data for table `actor`
--

LOCK TABLES `actor` WRITE;
/*!40000 ALTER TABLE `actor` DISABLE KEYS */;
INSERT INTO `actor` VALUES (1,'PENELOPE','GUINESS','2006-02-14 23:04:33'),(2,'NICK','WAHLBERG','2006-02-14 23:04:33'),
/*!40000 ALTER TABLE `actor` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Dumping data for table `category`
--

LOCK TABLES `category` WRITE;
/*!40000 ALTER TABLE `category` DISABLE KEYS */;
INSERT INTO `category` VALUES (1,'Action','2006-02-14 23:16:27'),(2,'Animation','2006-02-14 23:16:27'),(3,'Children',
/*!40000 ALTER TABLE `category` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Dumping data for table `address`
```

As you can see in the above screenshot, the backup file contains the various T-SQL statements that can be used to insert data in the tables.

Restore the MySQL Database

Restoring a MySQL database using ***mysqldump*** is simple. To restore the database, you must create an empty database. First, let us drop and recreate the sakila database by executing the following command.

```
mysql> drop database sakila;
Query OK, 24 rows affected (0.35 sec)
mysql> create database sakila;
mysql> create database sakila;
Query OK, 1 row affected (0.01 sec)
MySQL>
```

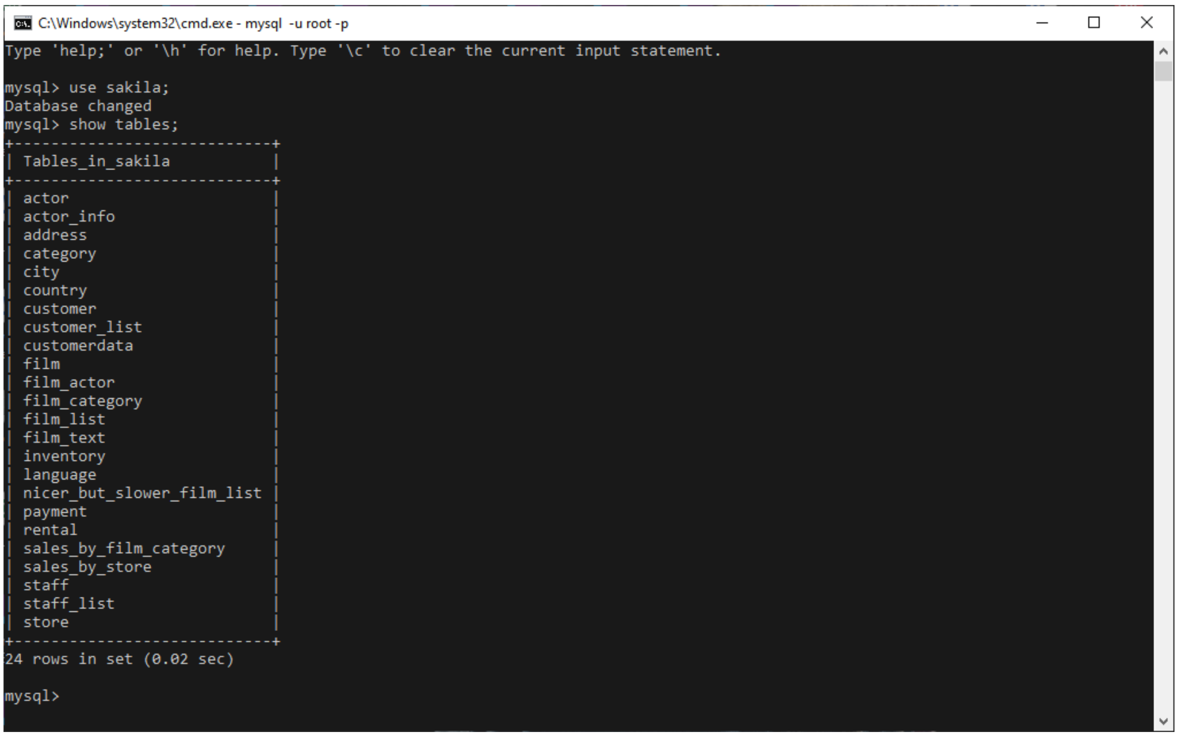
When you restore the database, instead of using ***mysqldump***, you must use ***mysql;*** otherwise, the ***mysqldump*** will not generate the schema and the data. Execute the following command to restore the sakila database:

```
mysql -u root -p sakila < C:\MySQLBackup\sakila_20200424.sql
```

Once command executes successfully, execute the following command to verify that all objects have been created on the **sakila** database.

```
mysql> use sakila;
Database changed
mysql> show tables;
```

See the following image:



```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use sakila;
Database changed
mysql> show tables;
+----------------------------+
| Tables_in_sakila           |
+----------------------------+
| actor                      |
| actor_info                 |
| address                    |
| category                   |
| city                       |
| country                    |
| customer                   |
| customer_list              |
| customerdata               |
| film                       |
| film_actor                 |
| film_category              |
| film_list                  |
| film_text                  |
| inventory                  |
| language                   |
| nicer_but_slower_film_list |
| payment                    |
| rental                     |
| sales_by_film_category     |
| sales_by_store             |
| staff                      |
| staff_list                 |
| store                      |
+----------------------------+
24 rows in set (0.02 sec)

mysql>
```

Restore a specific table in the database

For instance, someone dropped a table from the database. Instead of restoring the entire data-base, we can restore the dropped table from the available backup. To demonstrate, drop the **actor** table from the sakila database by executing the following command on the MySQL command-line tool.

```
mysql> use sakila;
Database changed
mysql> drop table actor;
```

To restore the actor table, perform the following step by step process.

Step 1 :

Create a dummy database named ***sakila_dummy*** and restore the backup of the ***sakila*** database on it. Following is the command.

```
mysql> create database sakila_dummy;
mysql> use sakila_dummy;
mysql> source C:\MySQLBackup\sakila_20200424.sql
```

Step 2:

Backup the ***actor*** table to ***sakila_dummy_actor_20200424.sql*** file. Following is the command

```
C:\Users\Nisarg> mysqldump -u root -p sakila_dummy actor > C:\MySQLBackup\saki-
la_dummy_actor_20200424.sql
```

Step 3:

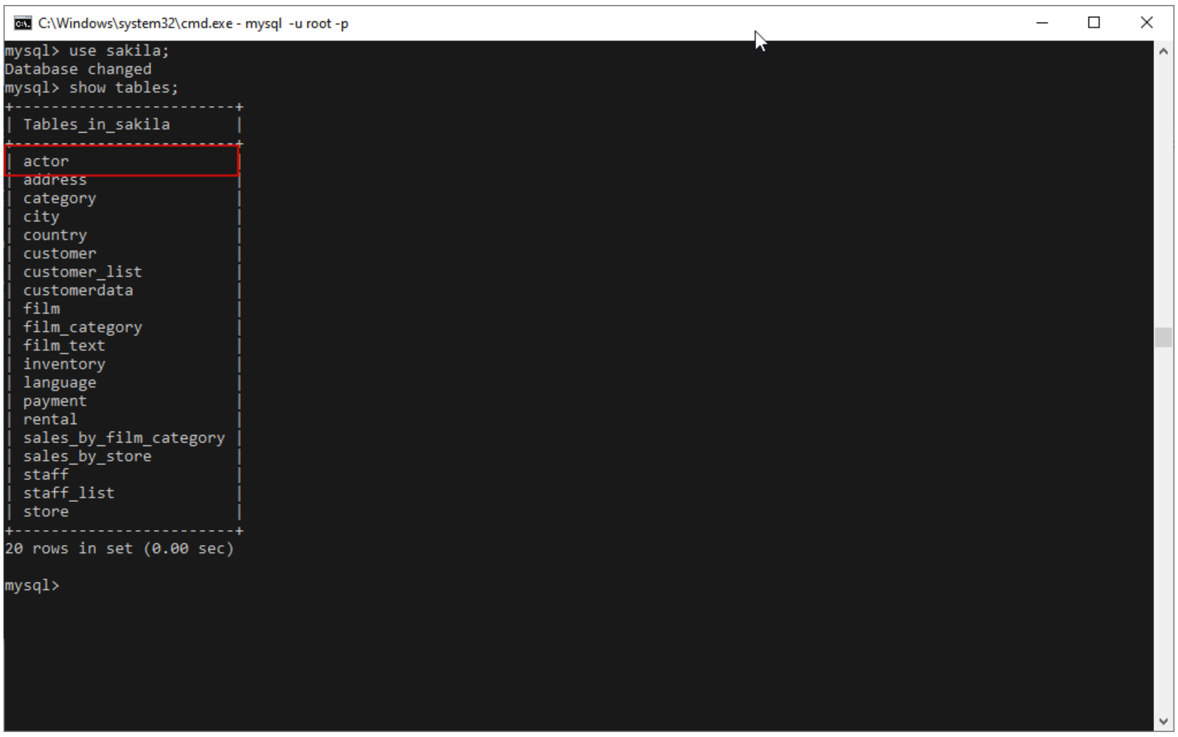
Restore the actor table from the "***sakila_dummy_actor_20200424.sql***" file. Following is the command on the MySQL command-line tool.

```
mysql> source C:\MySQLBackup\sakila_dummy_actor_20200424.sql
```

Execute the following command to verify the table has been restored successfully.

```
mysql> use sakila;
Database changed
mysql> show tables;
```

See the following image:



```
mysql> use sakila;
Database changed
mysql> show tables;
+------------------------+
| Tables_in_sakila       |
+------------------------+
| actor                  |
| address                |
| category               |
| city                   |
| country                |
| customer               |
| customer_list          |
| customerdata           |
| film                   |
| film_category          |
| film_text              |
| inventory              |
| language               |
| payment                |
| rental                 |
| sales_by_film_category |
| sales_by_store         |
| staff                  |
| staff_list             |
| store                  |
+------------------------+
20 rows in set (0.00 sec)

mysql>
```

Summary

Summary

In this article, I have explained how we can use the ***mysqldump*** command-line utility to generate the following:

1. The backup of MySQL database, table, or the structure of the database
2. Restore the MySQL database or table from the backup