Migrating Amazon RDS for MySQL to Azure Database for MySQL using the MySQL Workbench Migration Wizard

You can use a variety of utilities, such as MySQL Workbench Export/Import, Azure Database Migration Service (DMS), and MySQL dump and restore, to migrate Amazon RDS for MySQL to Azure Database for MySQL. However, using the MySQL Workbench Migration Wizard provides an easy and convenient way to move your Amazon RDS for MySQL databases to Azure Database for MySQL.

With the Migration Wizard, you can conveniently select which schemas and objects to migrate. It also provides the ability to view server logs to identify errors and bottlenecks in real time. As a result, you can edit and modify tables or database structures and objects during the migration process when an error is detected, and then resume migration without having to restart from scratch.

Note: You can also use the Migration Wizard to migrate other sources, such as Microsoft SQL Server, Oracle, PostgreSQL, MariaDB, etc., but we'll cover those scenarios in separate blog posts.

Prerequisites

Before you start the migration process, it is recommended that you ensure that several parameters and features are configured and set up properly, as described below.

- Make sure the Character set of the source and target databases are the same.
- Set the wait timeout to a reasonable time depending on the amount data or workload you want to import or migrate.
- Set the `max_allowed_packet` parameter to a reasonable amount depending on the size of the database you want to import or migrate.
- Verify that all of your tables use InnoDB, as Azure Database for MySQL Server supports only InnoDB Storage engine.
- Remove, replace, or modify all triggers, stored procedures, and other functions containing root user or super user definers (Azure Database for MySQL doesn't support the Super user privilege). To replace the definers with the name of the admin user that is running the import process, run the following command:

  ```
  DELIMITER; ;/*!50003 CREATE*/ /*!50017 DEFINER=`root`@`127.0.0.1`*/ /*!50003
  DELIMITER;
  /* Modified to */
  DELIMITER;
  /*!50003 CREATE*//*!50017 DEFINER=`AdminUserName`@`ServerName`*/ /*!50003
  DELIMITER;
  ```
- If User Defined Functions (UDFs) are running on your database server, you need to delete the privilege for the mysql database. To determine if any UDFs are running on your server, use the following query:

```
SELECT * FROM mysql.func;
```

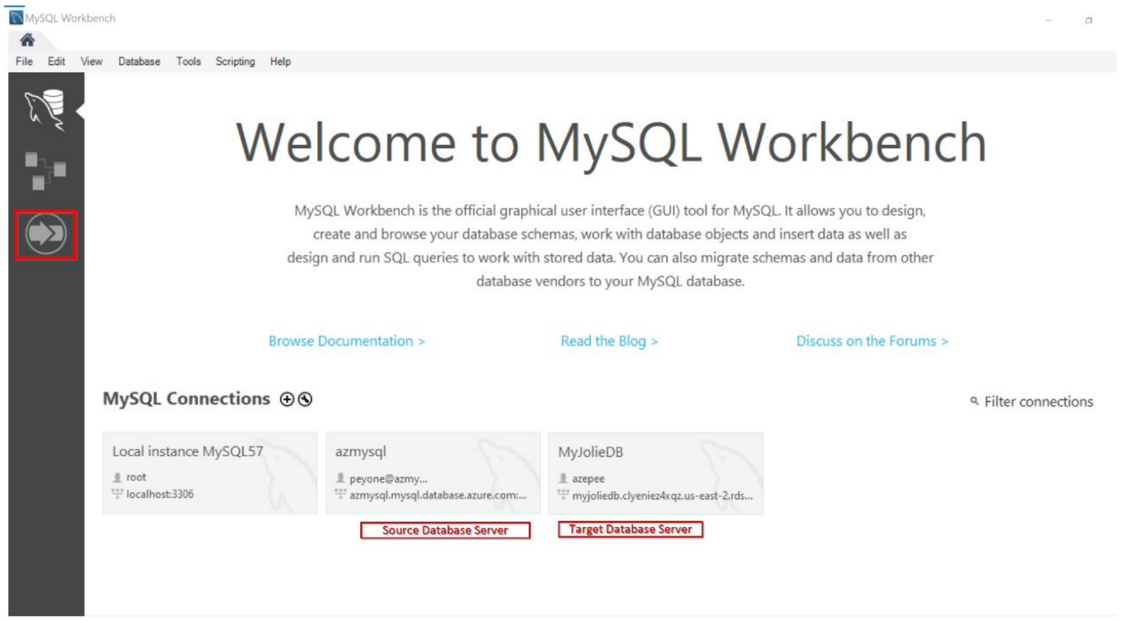
If you discover that UDFs are running, you can drop the UDFs by using the following query:

```
DROP FUNCTION your_UDFunction;
```

- Make sure that the server on which the tool is running, and ultimately the export location, has ample disk space and compute power (Vcores, CPU, and Memory) to perform the export operation, especially when exporting a very large database.
- Create a path between the on-premises or AWS instance and Azure Database for MySQL if the workload is behind firewalls or other network security layers.

Begin the migration process.

1. To start the migration process, log onto MySQL Workbench, and then select the home icon.
2. In the left-hand navigation bar, select the Migration Wizard icon, as shown in the screenshot below.



Welcome to MySQL Workbench

MySQL Workbench is the official graphical user interface (GUI) tool for MySQL. It allows you to design, create and browse your database schemas, work with database objects and insert data as well as design and run SQL queries to work with stored data. You can also migrate schemas and data from other database vendors to your MySQL database.

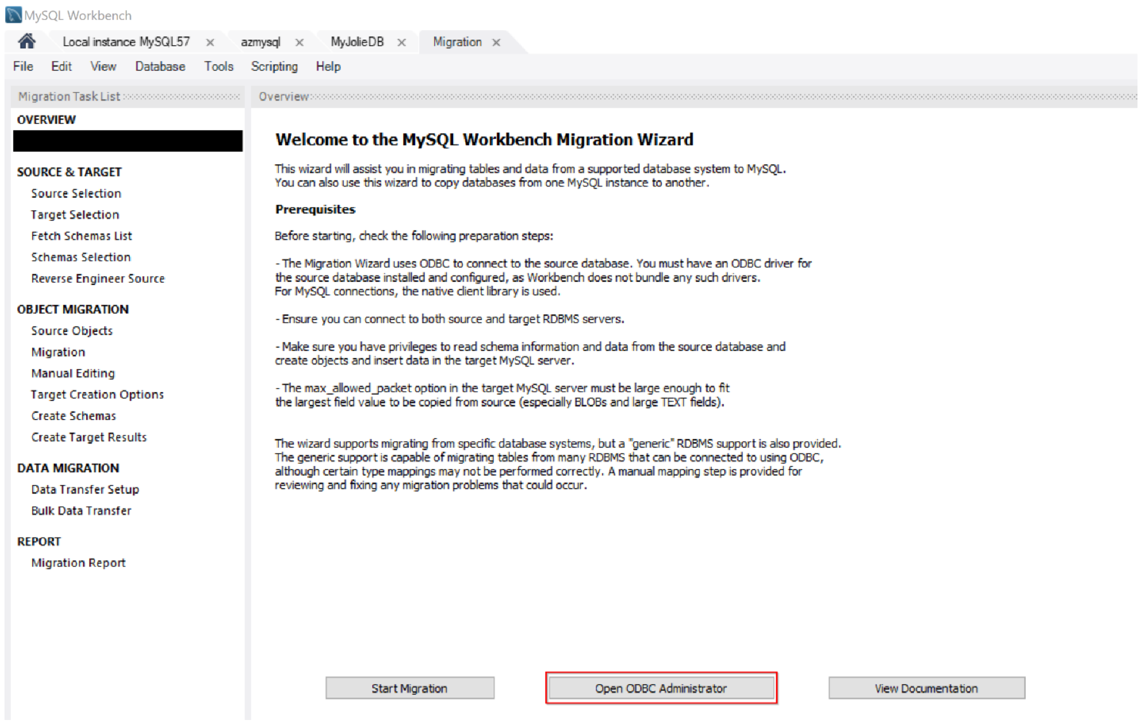
MySQL Connections

Local instance MySQL57
root
localhost:3306

azmysql
peyone@azmy...
azmysql.mysql.database.azure.com:...

MyJolieDB
azepee
myjoliedb.clyeniez4xqz.us-east-2.rds...

The **Overview** page of the Migration Wizard is displayed, as shown below.



Welcome to the MySQL Workbench Migration Wizard

This wizard will assist you in migrating tables and data from a supported database system to MySQL.
You can also use this wizard to copy databases from one MySQL instance to another.

Prerequisites

Before starting, check the following preparation steps:

- The Migration Wizard uses ODBC to connect to the source database. You must have an ODBC driver for the source database installed and configured, as Workbench does not bundle any such drivers.
For MySQL connections, the native client library is used.

- Ensure you can connect to both source and target RDBMS servers.

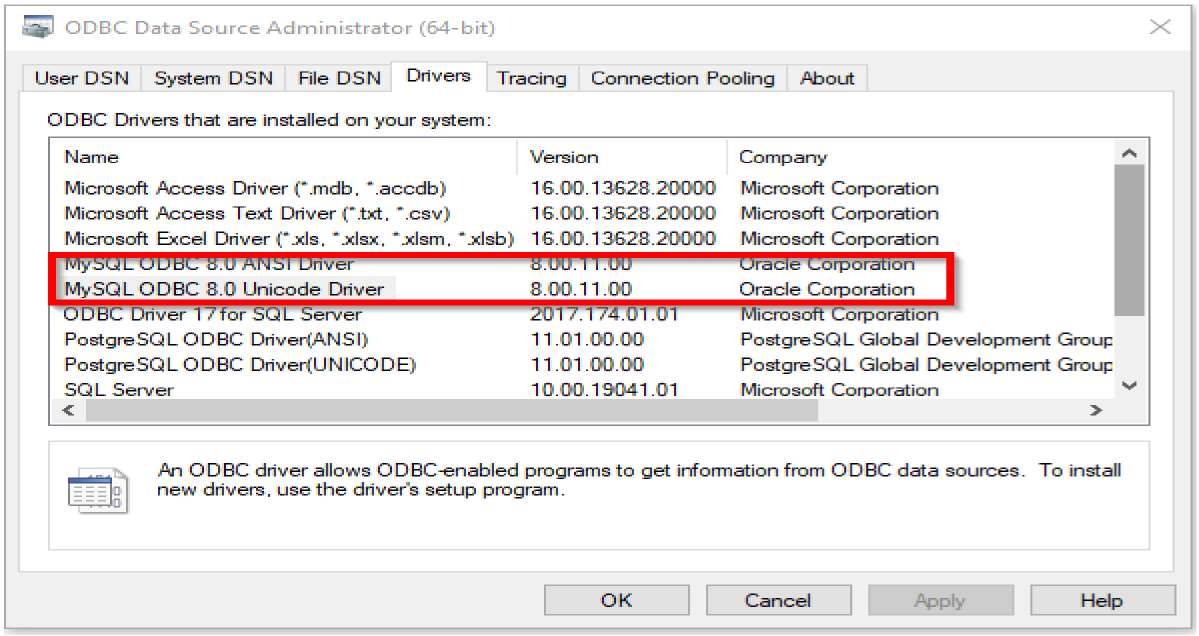
- Make sure you have privileges to read schema information and data from the source database and create objects and insert data in the target MySQL server.

- The max_allowed_packet option in the target MySQL server must be large enough to fit the largest field value to be copied from source (especially BLOBs and large TEXT fields).

The wizard supports migrating from specific database systems, but a "generic" RDBMS support is also provided.
The generic support is capable of migrating tables from many RDBMS that can be connected to using ODBC, although certain type mappings may not be performed correctly. A manual mapping step is provided for reviewing and fixing any migration problems that could occur.

3. Determine if you have an ODBC driver for MySQL Server installed by selecting **Open ODBC Administrator**.

   In our case, on the **Drivers** tab, you'll notice that there are already two MySQL Server ODBC drivers installed.



ODBC Drivers that are installed on your system:

Name	Version	Company
Microsoft Access Driver (*.mdb, *.accdb)	16.00.13628.20000	Microsoft Corporation
Microsoft Access Text Driver (*.txt, *.csv)	16.00.13628.20000	Microsoft Corporation
Microsoft Excel Driver (*.xls, *.xlsx, *.xlsm, *.xlsb)	16.00.13628.20000	Microsoft Corporation
MySQL ODBC 8.0 ANSI Driver	8.00.11.00	Oracle Corporation
MySQL ODBC 8.0 Unicode Driver	8.00.11.00	Oracle Corporation
ODBC Driver 17 for SQL Server	2017.174.01.01	Microsoft Corporation
PostgreSQL ODBC Driver(ANSI)	11.01.00.00	PostgreSQL Global Development Group
PostgreSQL ODBC Driver(UNICODE)	11.01.00.00	PostgreSQL Global Development Group
SQL Server	10.00.19041.01	Microsoft Corporation

An ODBC driver allows ODBC-enabled programs to get information from ODBC data sources. To install new drivers, use the driver's setup program.

If a MySQL ODBC driver is not installed, use the MySQL Installer you used to install MySQL Workbench to install the driver. For more information about MySQL ODBC driver installation, see the following resources:

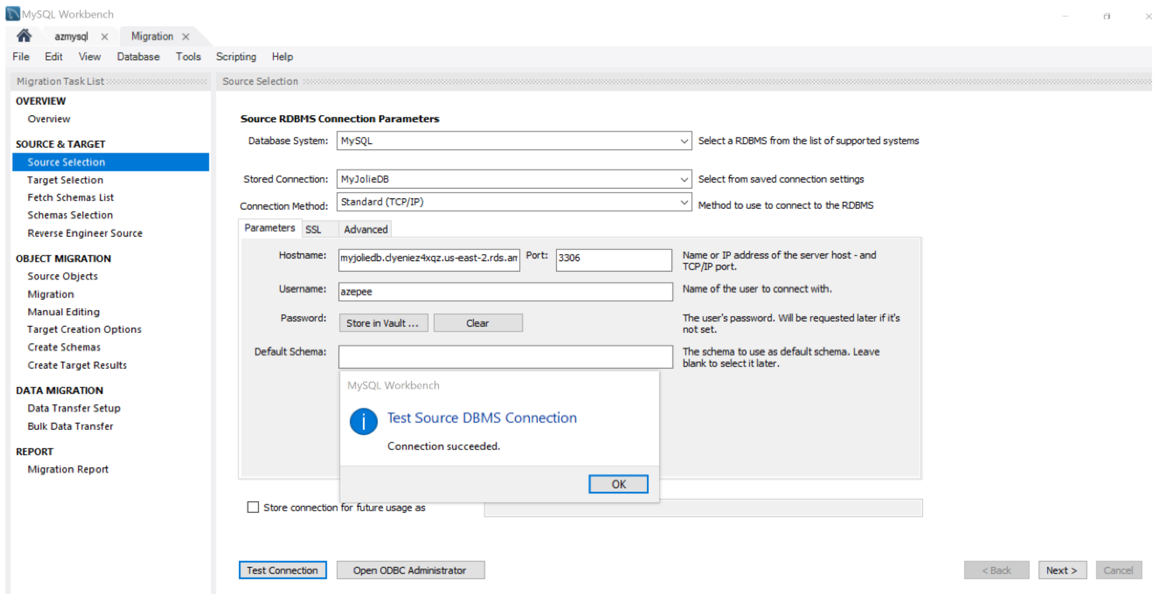
    - [MySQL :: MySQL Connector/ODBC Developer Guide :: 4.1 Installing Connector/ODBC on Windows]()
    - [ODBC Driver for MySQL: How to Install and Set up Connection (Step-by-step) – {coding}Sight (codingsight.com)]()
4. Close the **ODBC Data Source Administrator** dialog box, and then continue with the migration process.

Configuring connection parameters for the source database server

1. On the **Overview** page, select **Start Migration**.

   The **Source Selection** page appears. Use this page to provide information about the RDBMS you are migrating from and the parameters for the connection.
2. In the **Database System** field, select **MySQL**.
3. In the **Stored Connection** field, select one of the saved connection settings for that RDBMS.

   You can save connections by marking the checkbox at the bottom of the page and providing a name of your preference.
4. In the **Connection Method** field, select **Standard TCP/IP**.
5. In the **Hostname** text field, type the name of your source database server.
6. In the Port field, specify **3306**, and then enter the username and password to use to connect to the server.
7. In the **Database** field, enter the name of the database you want to migrate if you know it; otherwise leave this field blank.



Source RDBMS Connection Parameters

Database System: MySQL — Select a RDBMS from the list of supported systems

Stored Connection: MyJolieDB — Select from saved connection settings

Connection Method: Standard (TCP/IP) — Method to use to connect to the RDBMS

Hostname: myjoliedb.dyeniez4xqz.us-east-2.rds.an Port: 3306 — Name or IP address of the server host - and TCP/IP port.

Username: azepee — Name of the user to connect with.

Password: — The user's password. Will be requested later if it's not set.

Default Schema: — The schema to use as default schema. Leave blank to select it later.

MySQL Workbench

Test Source DBMS Connection

Connection succeeded.

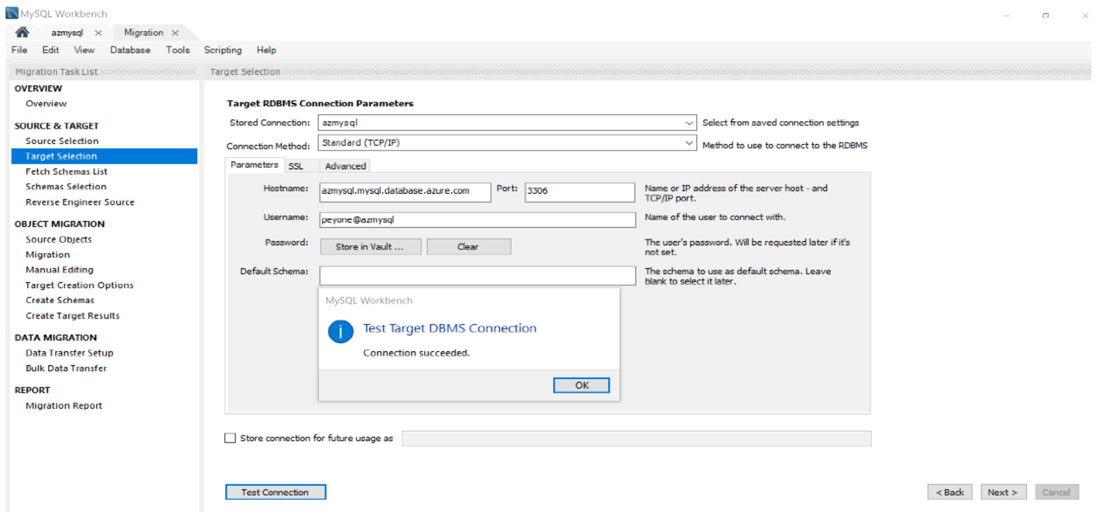
☐ Store connection for future usage as

8. Select **Test Connection** to check the connection to your MySQL Server instance.

   If you've entered the correct parameters, a message appears indicating a successful connection attempt.
9. Select **Next**.

Configuring connection parameters for the target database server

1. On the **Target Selection** page, set the parameters to connect to your Destination MySQL Server instance using a process similar to that for setting up the connection to the source server.
2. When you are done, select **Test Connection** to verify a successful connection.



Target RDBMS Connection Parameters

Stored Connection: azmysql — Select from saved connection settings

Connection Method: Standard (TCP/IP) — Method to use to connect to the RDBMS

Hostname: azmysql.mysql.database.azure.com Port: 3306 — Name or IP address of the server host - and TCP/IP port.

Username: peyone@azmysql — Name of the user to connect with.

Password: — The user's password. Will be requested later if it's not set.

Default Schema: — The schema to use as default schema. Leave blank to select it later.

MySQL Workbench

Test Target DBMS Connection

Connection succeeded.

Store connection for future usage as

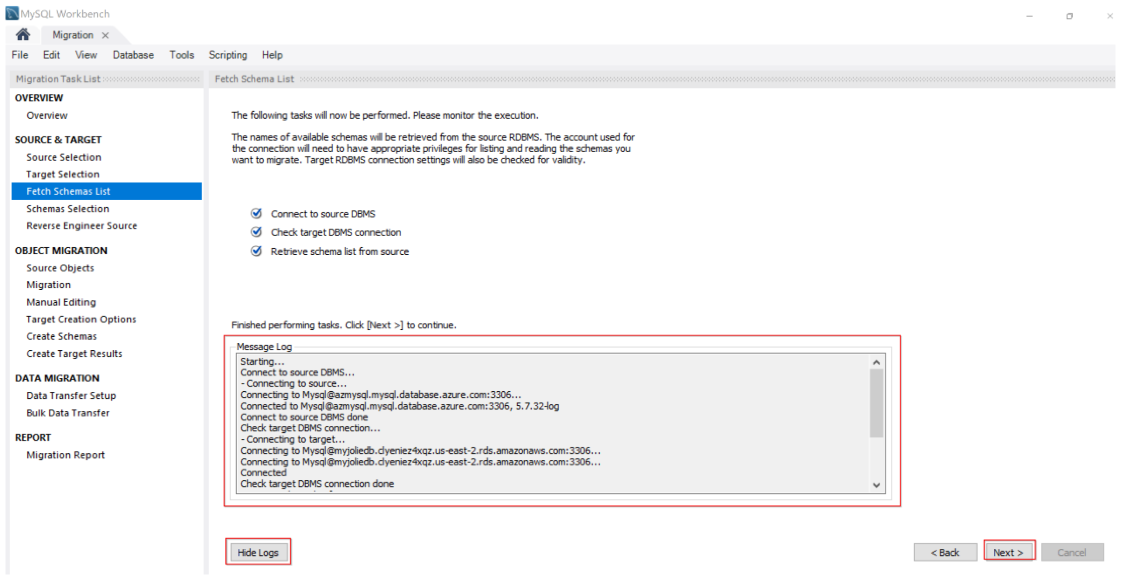
3. Select **Next**.

Selecting the schemata to migrate

The Migration Wizard will communicate to your MySQL Server instance and fetch a list of schemata from the source server.

1. Select **show logs** to view this operation.

   The screenshot below shows how the schemata are being retrieved from the source database server.



The following tasks will now be performed. Please monitor the execution.

The names of available schemas will be retrieved from the source RDBMS. The account used for the connection will need to have appropriate privileges for listing and reading the schemas you want to migrate. Target RDBMS connection settings will also be checked for validity.

- ✓ Connect to source DBMS
- ✓ Check target DBMS connection
- ✓ Retrieve schema list from source

Finished performing tasks. Click [Next >] to continue.

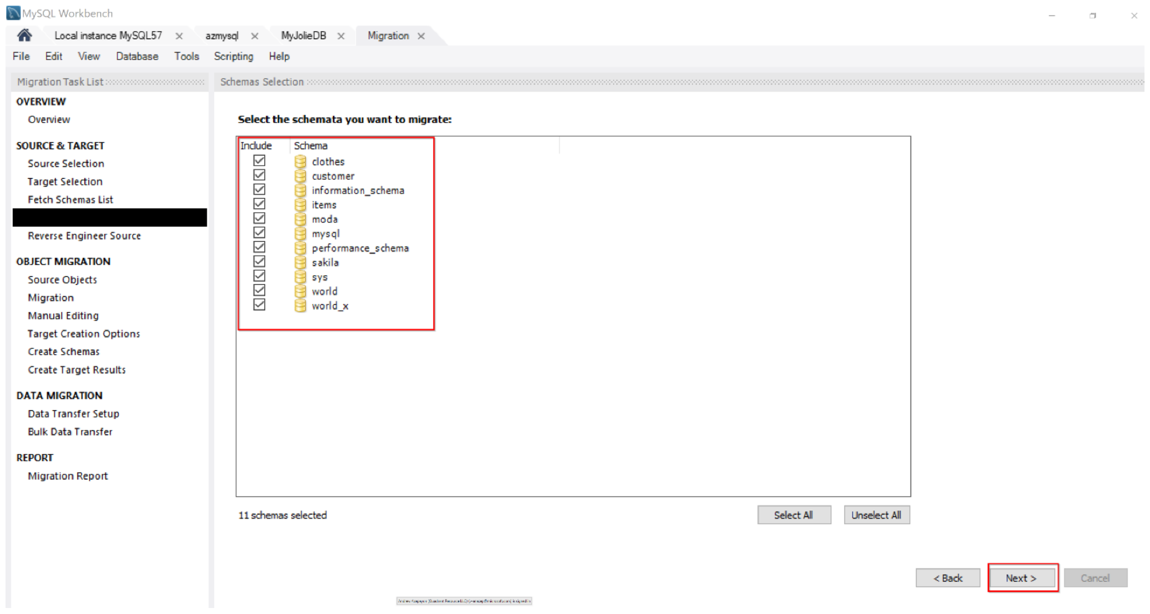
Message Log

```
Starting...
Connect to source DBMS...
- Connecting to source...
Connecting to Mysql@azmysql.mysql.database.azure.com:3306...
Connected to Mysql@azmysql.mysql.database.azure.com:3306, 5.7.32-log
Connect to source DBMS done
Check target DBMS connection...
- Connecting to target...
Connecting to Mysql@myjoliedb.clyeniez4xqz.us-east-2.rds.amazonaws.com:3306...
Connecting to Mysql@myjoliedb.clyeniez4xqz.us-east-2.rds.amazonaws.com:3306...
Connected
Check target DBMS connection done
```

2. Select **Next** to verify that all the schemata were successfully fetched.

The screenshot below shows the list of fetched schemata.

3. Select the schemata that you want to migrate.



Select the schemata you want to migrate:

Include	Schema
☑	clothes
☑	customer
☑	information_schema
☑	items
☑	moda
☑	mysql
☑	performance_schema
☑	sakila
☑	sys
☑	world
☑	world_x

11 schemas selected

Keep in mind that you can only migrate schemata that appear in the list of schemata.

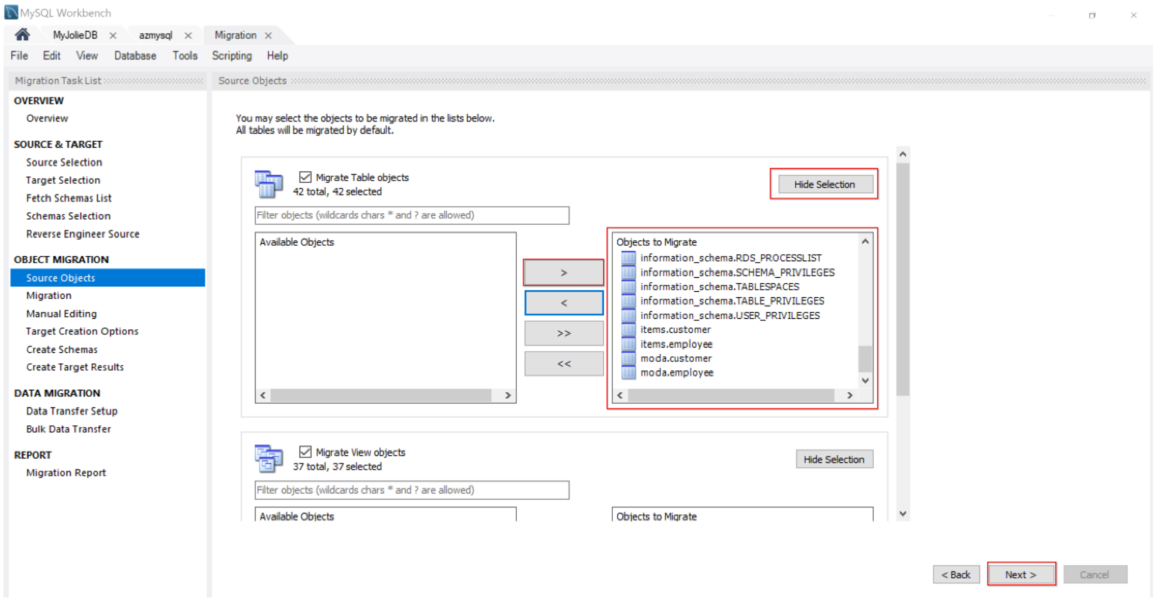
4. Select **Next**.

Object migration

Next, specify the object(s) that you want to migrate.

1. Select **Show Selection**, and then, under **Available Objects**, select and add the objects that you want to migrate.

   When you have added the objects, they will appear under **Objects to Migrate**, as shown in the screenshot below.



You may select the objects to be migrated in the lists below.
All tables will be migrated by default.

[✓] Migrate Table objects
42 total, 42 selected

Filter objects (wildcards chars * and ? are allowed)

Objects to Migrate:
information_schema.RDS_PROCESSLIST
information_schema.SCHEMA_PRIVILEGES
information_schema.TABLESPACES
information_schema.TABLE_PRIVILEGES
information_schema.USER_PRIVILEGES
items.customer
items.employee
moda.customer
moda.employee

[✓] Migrate View objects
37 total, 37 selected

Filter objects (wildcards chars * and ? are allowed)

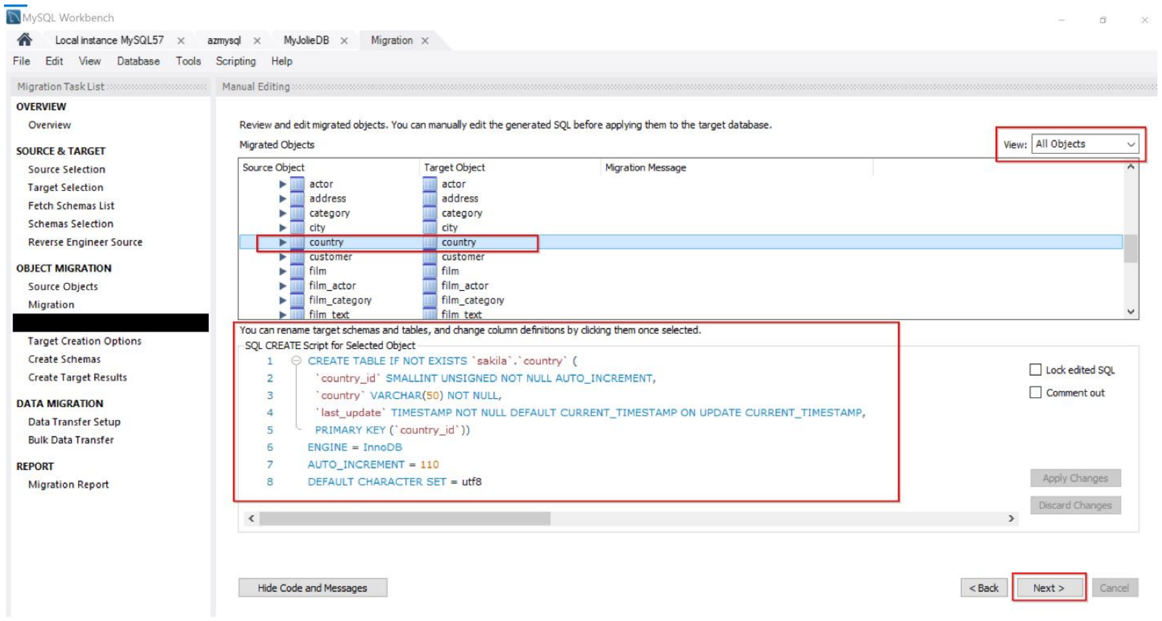
In this scenario, we've selected all table objects.

2. Select **Next**.

Editing data

In this section you have the option of editing the objects that you want to migrate.

1. On the **Manual Editing** page, notice the **View** drop-down menu in the top-right corner.



Review and edit migrated objects. You can manually edit the generated SQL before applying them to the target database.

Migrated Objects

View: All Objects

Source Object	Target Object	Migration Message
actor	actor	
address	address	
category	category	
city	city	
country	country	
customer	customer	
film	film	
film_actor	film_actor	
film_category	film_category	
film_text	film_text	

You can rename target schemas and tables, and change column definitions by clicking them once selected.

SQL CREATE Script for Selected Object

```
CREATE TABLE IF NOT EXISTS `sakila`.`country` (
  `country_id` SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
  `country` VARCHAR(50) NOT NULL,
  `last_update` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (`country_id`))
ENGINE = InnoDB
AUTO_INCREMENT = 110
DEFAULT CHARACTER SET = utf8
```

The **View** drop-down box includes three items:

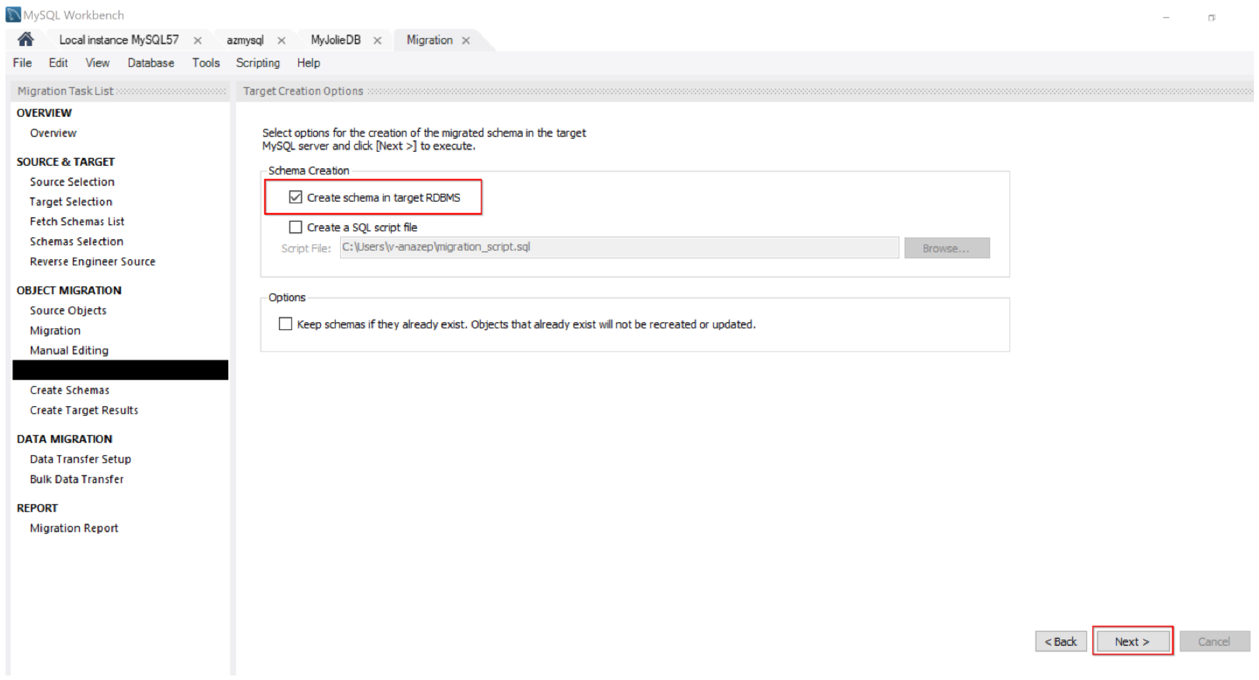
- **All Objects** – Displays all objects. With this option you can manually edit the generated SQL before applying them to the target database server. To do this, select the object and

select **Show Code and Messages**. You can see (and edit!) the generated MySQL code that corresponds to the selected object.
   - **Migration problems** – Displays any problems that occurred during the migration, which you can review and verify.
   - **Column Mapping** – Displays column mapping information. You can use this view to edit the name and change column of the target object.
2. Select **Next**.

Creating the target database

1. Select the **Create schema in target RDBMS** check box.

   You can also choose to keep already existing schemata, so they will not be modified or updated.



Select options for the creation of the migrated schema in the target MySQL server and click [Next >] to execute.

Schema Creation

[✓] Create schema in target RDBMS

[ ] Create a SQL script file

Script File: C:\Users\v-anazep\migration_script.sql

Options

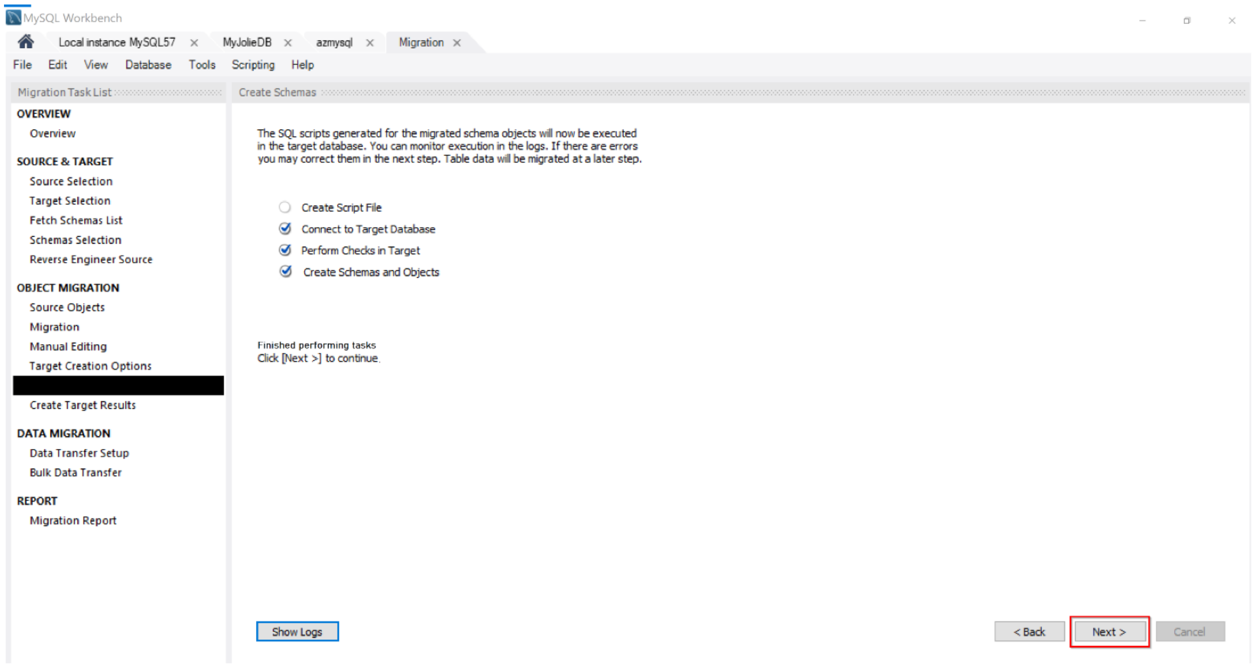
[ ] Keep schemas if they already exist. Objects that already exist will not be recreated or updated.

   We'll choose to create schema in target RDBMS option for this blog, but you can still select both options, the other option being to create a SQL script file, so that you can save the script to a file on your local machine for other purposes.

2. Select **Next**.

Running the MySQL script to create the database objects.

Since we have chosen the create schema in Target RDBMS option, the migrated SQL script will be executed in the target MySQL server. You can view its progress as shown in the screenshot below:



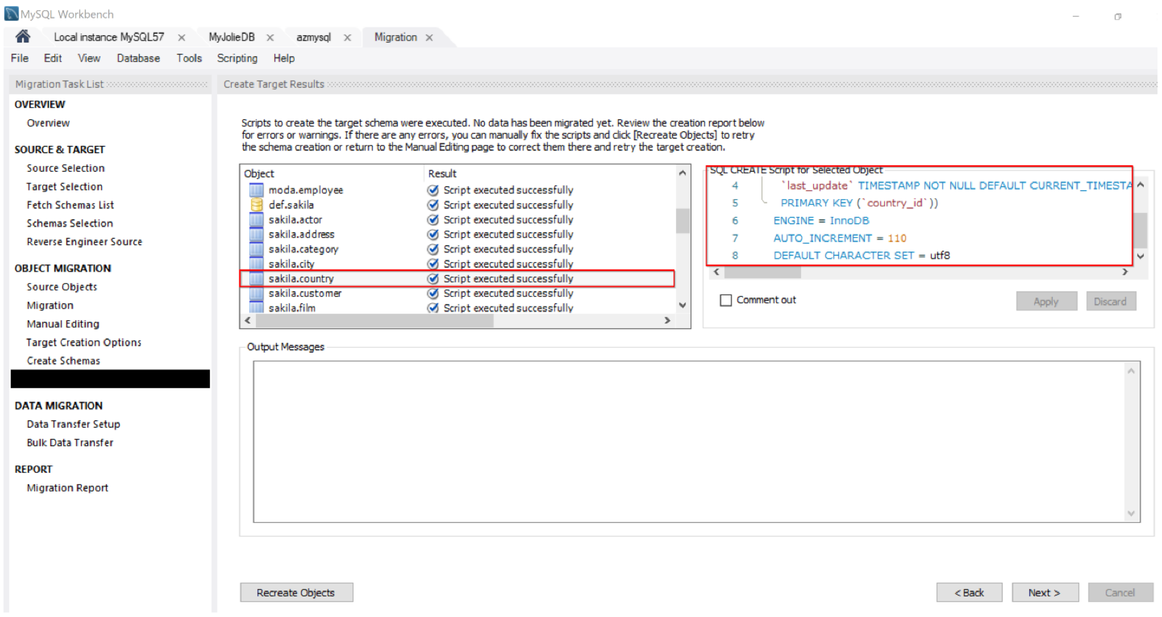
The SQL scripts generated for the migrated schema objects will now be executed in the target database. You can monitor execution in the logs. If there are errors you may correct them in the next step. Table data will be migrated at a later step.

- Create Script File
- [✓] Connect to Target Database
- [✓] Perform Checks in Target
- [✓] Create Schemas and Objects

Finished performing tasks
Click [Next >] to continue.

1. After creation of the schemata and their objects completes, select **Next**.

   On the **Create Target Results** page, you're presented with a list of the objects created and notification of any errors that were encountered while creating them, as shown in the following screenshot.



Scripts to create the target schema were executed. No data has been migrated yet. Review the creation report below for errors or warnings. If there are any errors, you can manually fix the scripts and click [Recreate Objects] to retry the schema creation or return to the Manual Editing page to correct them there and retry the target creation.

Object	Result
moda.employee	✓ Script executed successfully
def.sakila	✓ Script executed successfully
sakila.actor	✓ Script executed successfully
sakila.address	✓ Script executed successfully
sakila.category	✓ Script executed successfully
sakila.city	✓ Script executed successfully
sakila.country	✓ Script executed successfully
sakila.customer	✓ Script executed successfully
sakila.film	✓ Script executed successfully

```
    `last_update` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTA
    PRIMARY KEY (`country_id`))
  ENGINE = InnoDB
  AUTO_INCREMENT = 110
  DEFAULT CHARACTER SET = utf8
```

2. Review the detail on this page to verify that everything completed as intended.

   In this scenario we don't have any errors. If you do need to address any error messages, you can edit the migration script.

3. In the **Object** box, select the object that you want to edit.

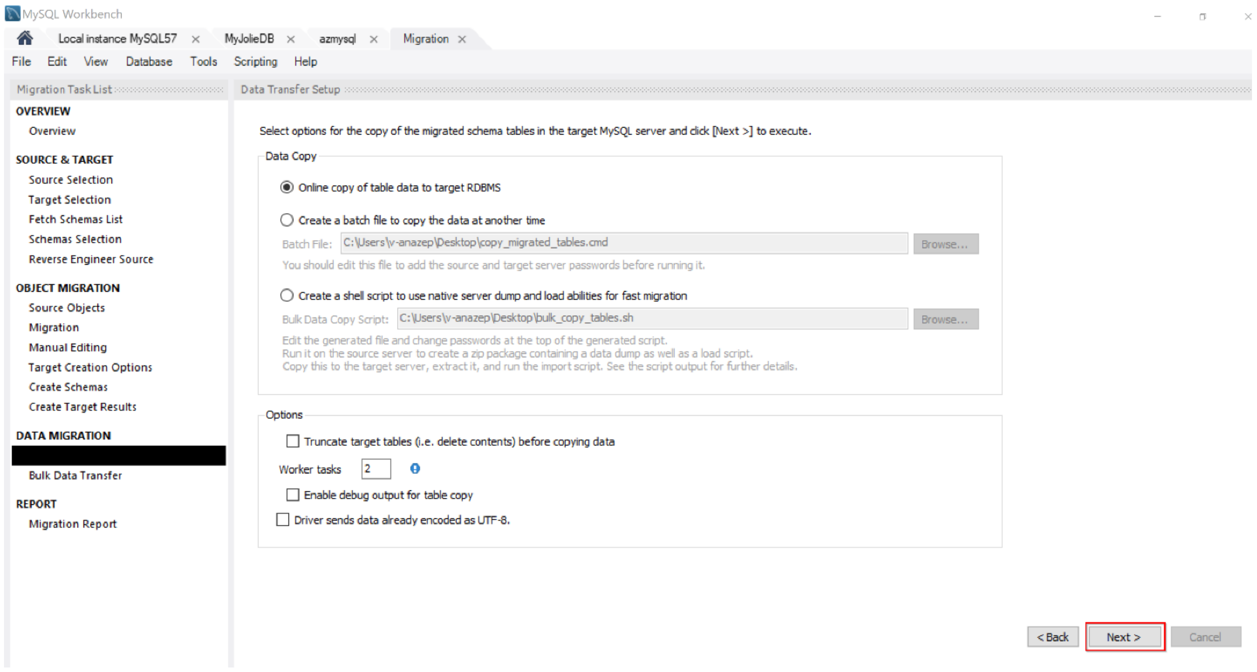
4. Under **SQL CREATE script** for selected object, modify your SQL script, and then select **Apply** to save the changes.
5. Select **Recreate Objects** to run the script including your changes.

   If the script fails, you may need to edit the generated script. You can then manually fix the SQL script and run everything again. In this blog, we're not changing anything, so we'll leave the script as it is.

6. Select **Next**.

Transferring Data

This part of the process moves data from the source MySQL Server database instance into your newly created target MySQL database instance. Use the **Data Transfer Setup** page to configure this process.



Select options for the copy of the migrated schema tables in the target MySQL server and click [Next >] to execute.

Data Copy

(●) Online copy of table data to target RDBMS

( ) Create a batch file to copy the data at another time

Batch File: C:\Users\v-anazep\Desktop\copy_migrated_tables.cmd Browse...

You should edit this file to add the source and target server passwords before running it.

( ) Create a shell script to use native server dump and load abilities for fast migration

Bulk Data Copy Script: C:\Users\v-anazep\Desktop\bulk_copy_tables.sh Browse...

Edit the generated file and change passwords at the top of the generated script.
Run it on the source server to create a zip package containing a data dump as well as a load script.
Copy this to the target server, extract it, and run the import script. See the script output for further details.

Options

☐ Truncate target tables (i.e. delete contents) before copying data

Worker tasks 2

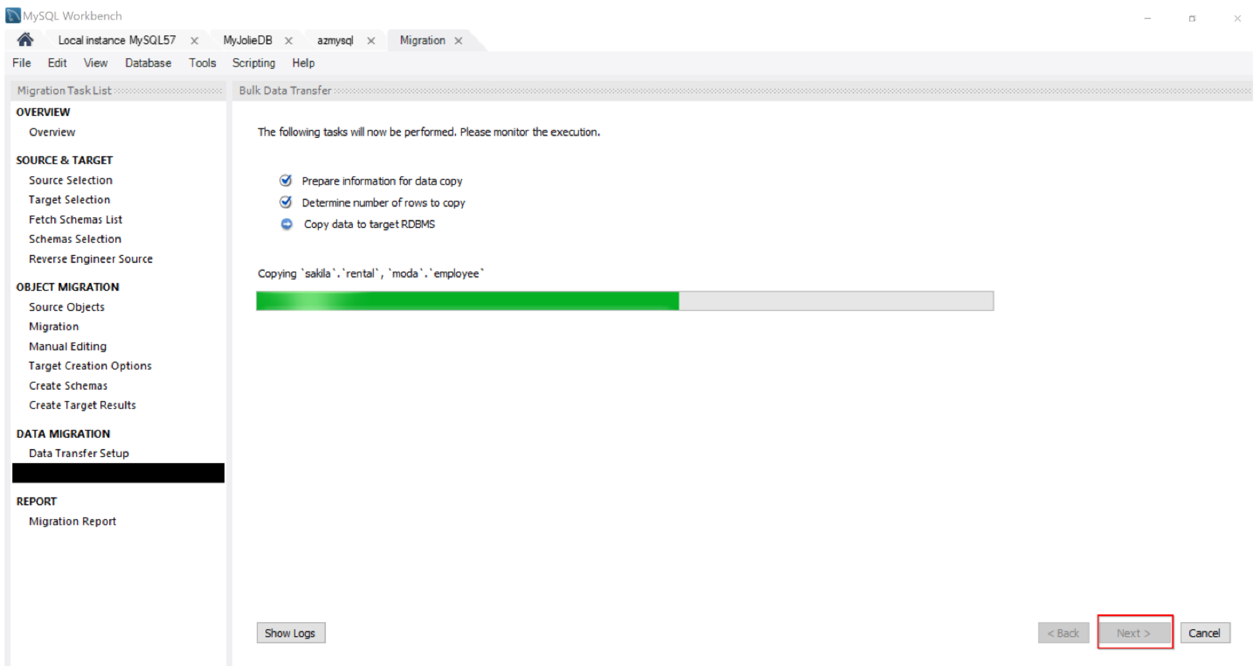
☐ Enable debug output for table copy

☐ Driver sends data already encoded as UTF-8.

This page provides two options. One option allows you to perform a live/online transfer or to dump the data into a batch or a shell file that you can run later. The other option provides a way to tune this process. For the purposes of this post, we'll leave the default values for the options.

1. To begin the actual process of transferring data, select **Next**.

   It will take a little while to copy the data. The progress of the data transfer process appears as shown in the following screenshot.



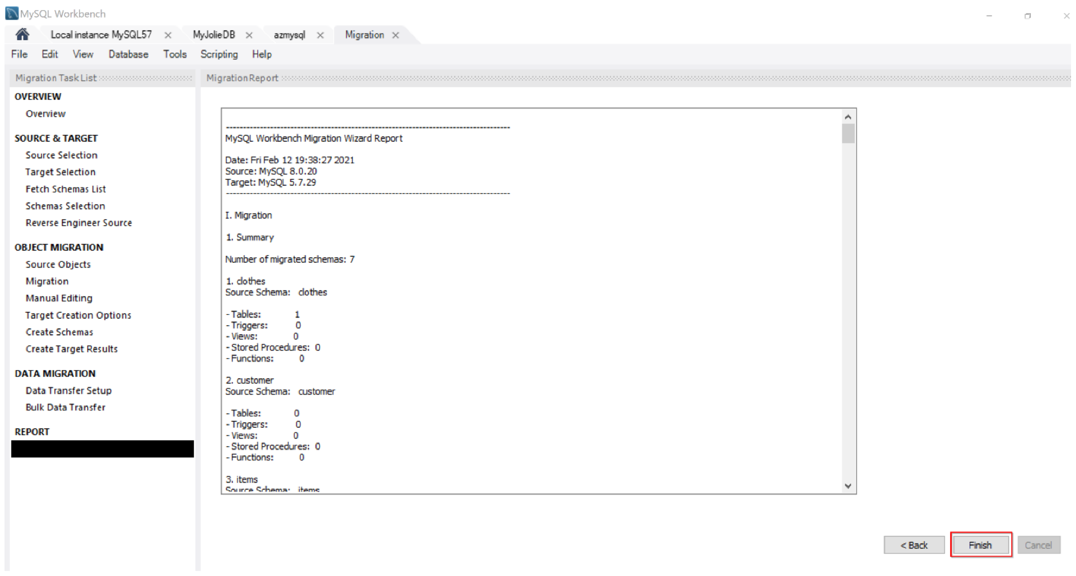
The following tasks will now be performed. Please monitor the execution.

- ✓ Prepare information for data copy
- ✓ Determine number of rows to copy
- Copy data to target RDBMS

Copying `sakila`.`rental`, `moda`.`employee`

2. After the transfer completes, select **Next**.

   The **Migration Report** page appears, providing a report summarizing the whole process, as shown on the screenshot below:



```
--------------------------------------------------------------------------------
MySQL Workbench Migration Wizard Report

Date: Fri Feb 12 19:38:27 2021
Source: MySQL 8.0.20
Target: MySQL 5.7.29
--------------------------------------------------------------------------------

I. Migration

1. Summary

Number of migrated schemas: 7

1. clothes
Source Schema:   clothes

- Tables:             1
- Triggers:           0
- Views:              0
- Stored Procedures:  0
- Functions:          0

2. customer
Source Schema:   customer

- Tables:             0
- Triggers:           0
- Views:              0
- Stored Procedures:  0
- Functions:          0

3. items
Source Schema:   items
```

3. Select **Finish** to close the Migration Wizard.

   The migration is now successfully completed.

Verifying consistency of the migrated schemata and tables

1. Next, log into your MySQL target database instance to verify that the migrated schemata and tables are consistent with your MySQL source database.

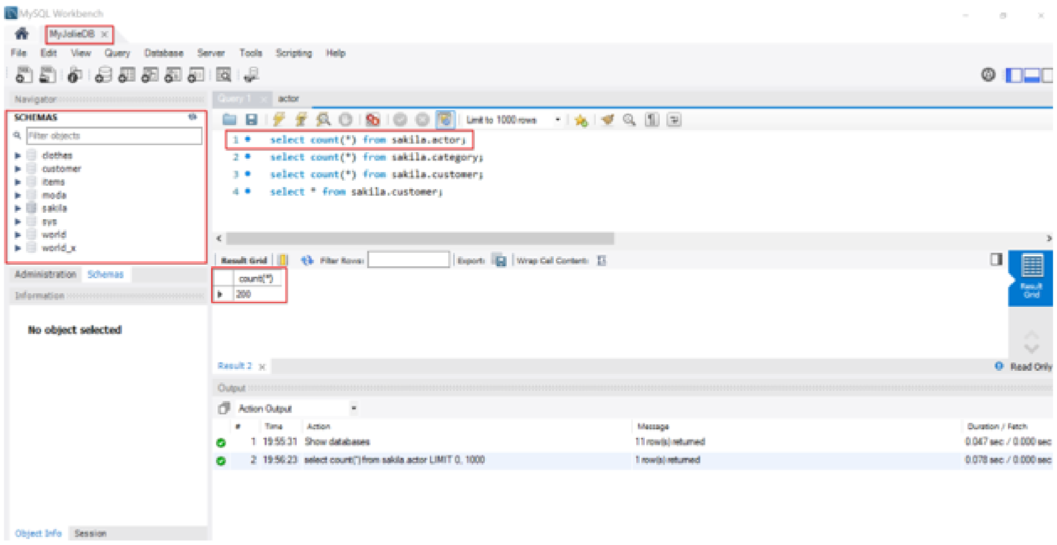
In our case, we can see that all schemata (sakila, moda, items, customer, clothes, world, and world_x) from the Amazon RDS for MySQL: **MyjolieDB** database have been successfully migrated to the Azure Database for MySQL: **azmysql** instance.

2. To verify the table and rows counts, run the following query on both instances:

```
SELECT COUNT (*) FROM sakila.actor;
```

You can see from the screenshot below that the row count for Amazon RDS MySQL is 200, which matches the Azure Database for MySQL instance.

AWS_RDS MySQL Instance: myjoliedb.clyniez4xqz.us-east-2.rds.amazonaws.com



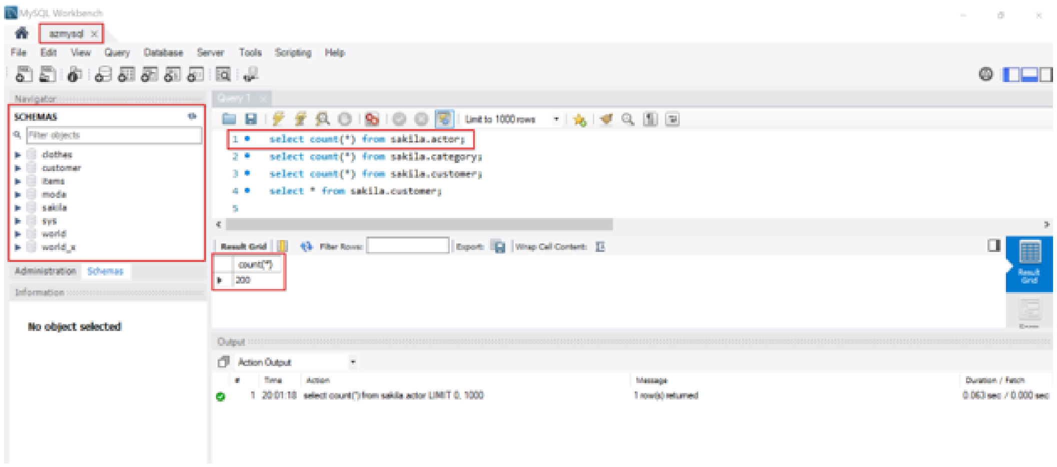
SCHEMAS: clothes, customer, items, moda, sakila, sys, world, world_x

```
select count(*) from sakila.actor;
select count(*) from sakila.category;
select count(*) from sakila.customer;
select * from sakila.customer;
```

count(*)
200

```
#   Time      Action                                          Message             Duration / Fetch
1   19:55:31  Show databases                                  11 row(s) returned  0.047 sec / 0.000 sec
2   19:56:23  select count(*) from sakila.actor LIMIT 0, 1000 1 row(s) returned   0.078 sec / 0.000 sec
```

Azure Database for MySQL Instance: azmysql.mysql.database.azure.com



SCHEMAS: clothes, customer, items, moda, sakila, sys, world, world_x

```
select count(*) from sakila.actor;
select count(*) from sakila.category;
select count(*) from sakila.customer;
select * from sakila.customer;
```

count(*)
200

```
#   Time      Action                                          Message             Duration / Fetch
1   20:01:18  select count(*) from sakila.actor LIMIT 0, 1000 1 row(s) returned   0.063 sec / 0.000 sec
```

While you can run the above query on every single schema and table, that will be quite a bit of work if you're dealing with hundreds of thousands or even millions of tables. You can use the queries below to verify the schema (database) and table size instead.

3. To check the database size, run the following query:

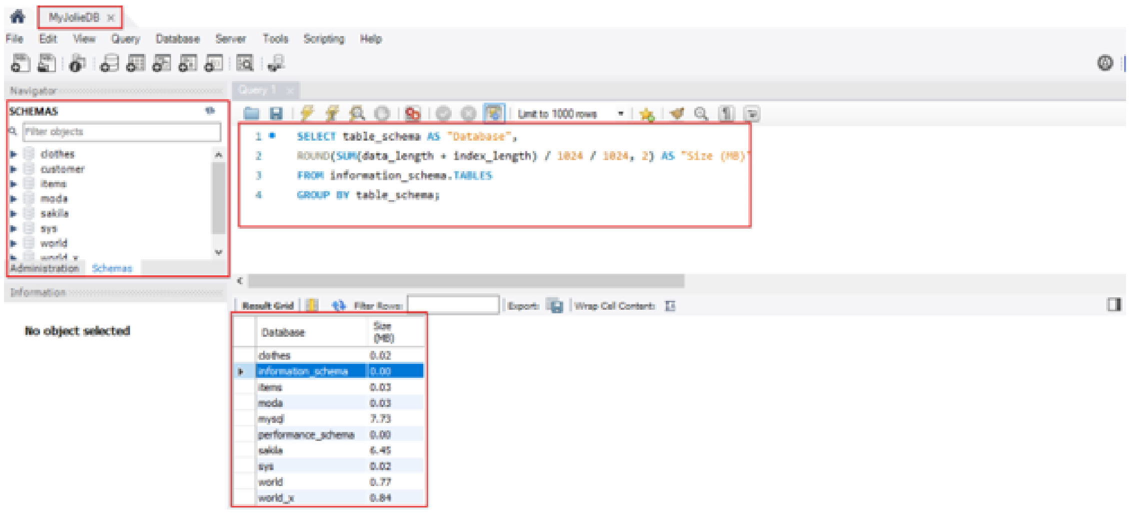
```
SELECT table_schema AS "Database",
ROUND(SUM(data_length + index_length) / 1024 / 1024, 2) AS "Size (MB)"
FROM information_schema.TABLES
GROUP BY table_schema;
```

4. To check the table size, run the following query:

```
SELECT table_name AS "Table",
ROUND(((data_length + index_length) / 1024 / 1024), 2) AS "Size (MB)"
FROM information_schema.TABLES
WHERE table_schema = "database_name"
ORDER BY (data_length + index_length) DESC;
```

You see from the screenshot below that schema (database) size from the Source Amazon RDS MySQL instance is the same as that of the target Azure Database for MySQL Instance.

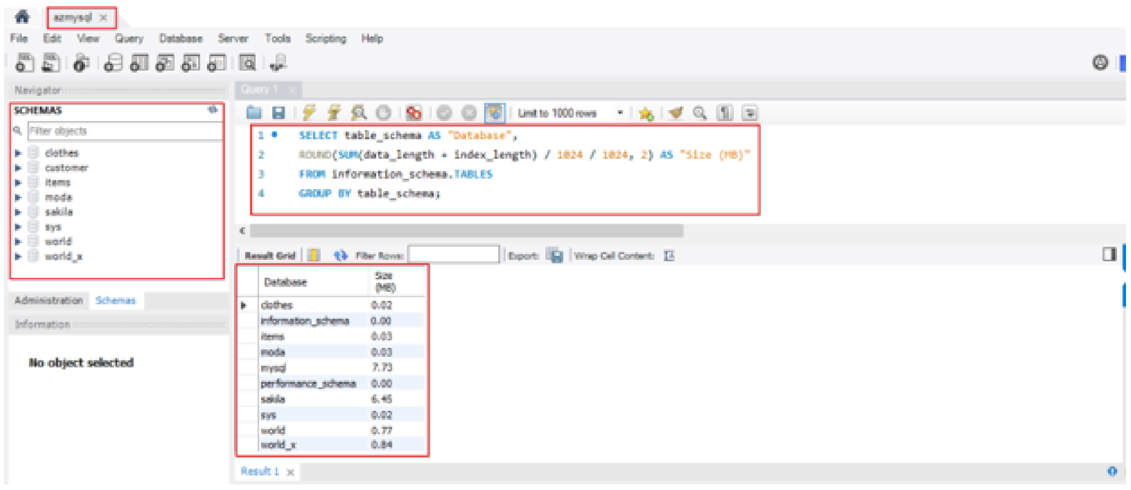
AWS_RDS MySQL Instance: myjoliedb.clyniez4xqz.us-east-2.rds.amazonaws.com



```
SELECT table_schema AS "Database",
ROUND(SUM(data_length + index_length) / 1024 / 1024, 2) AS "Size (MB)"
FROM information_schema.TABLES
GROUP BY table_schema;
```

Database	Size (MB)
clothes	0.02
information_schema	0.00
items	0.03
moda	0.03
mysql	7.73
performance_schema	0.00
sakila	6.45
sys	0.02
world	0.77
world_x	0.84

Azure Database for MySQL Instance: azmysql.mysql.database.azure.com



```
SELECT table_schema AS "Database",
ROUND(SUM(data_length + index_length) / 1024 / 1024, 2) AS "Size (MB)"
FROM information_schema.TABLES
GROUP BY table_schema;
```

Database	Size (MB)
clothes	0.02
information_schema	0.00
items	0.03
moda	0.03
mysql	7.73
performance_schema	0.00
sakila	6.45
sys	0.02
world	0.77
world_x	0.84

Since the schema (database) sizes are the same in both instances, you don’t really need to check individual table sizes. Nevertheless, you can always use the above query to check your table sizes, as necessary.

You’ve now confirmed that your migration completed successfully.

Conclusion

This concludes my post on migrating Amazon RDS for MySQL to Azure Database for MySQL using the MySQL Workbench Migration Wizard. I hope it’s provided you with the information you need to perform this important task. If you have questions, please feel free to reach out to me or the Ask Azure DB for MySQL alias. Thank you!