# Developing apps quickly with Azure Database for PostgreSQL

## Overview

Azure Database for PostgreSQL is a PostgreSQL database service built on Microsoft’s scalable cloud infrastructure for application developers. Leverage your existing open-source PostgreSQL skills and tools and, scale on-the-fly without downtime to efficiently deliver existing and new applications with reduced operational overhead. Built-in features maximize performance, availability, and security. Azure Database for PostgreSQL empowers developers to focus on application innovation instead of database management tasks.

### Scenario Overview

This hands-on lab (code-challenge) will step you through the following features:

1. Gitclone an application - [**https://github.com/vitorfs/bootcamp.git**](https://github.com/vitorfs/bootcamp.git) on your Ubuntu Linux VM.
2. Validate the .env using “vim .env”; which reflects the shiny new “Azure Database for PostgreSQL” (this is pre-populated)
3. **Run migration and serve the app**
   * python manage.py migrate
   * python manage.py runserver 0.0.0.0:8000
4. **Go to chrome/firefox and go to** http://<appidaddress**>**
5. Play around with the app, go to **PgAdmin to show that it works**

### About the code challenge

Wish to learn how to use the new Azure Database for PostgreSQL to build apps rapidly? This exercise will walk you through the steps to deploy a Python/Djano app and connect to PostgreSQL database service. Part of the lab, you will experience the capabilities of PostgreSQL database service using PostgreSQL tools.

Here's what we have already done for you to get started.

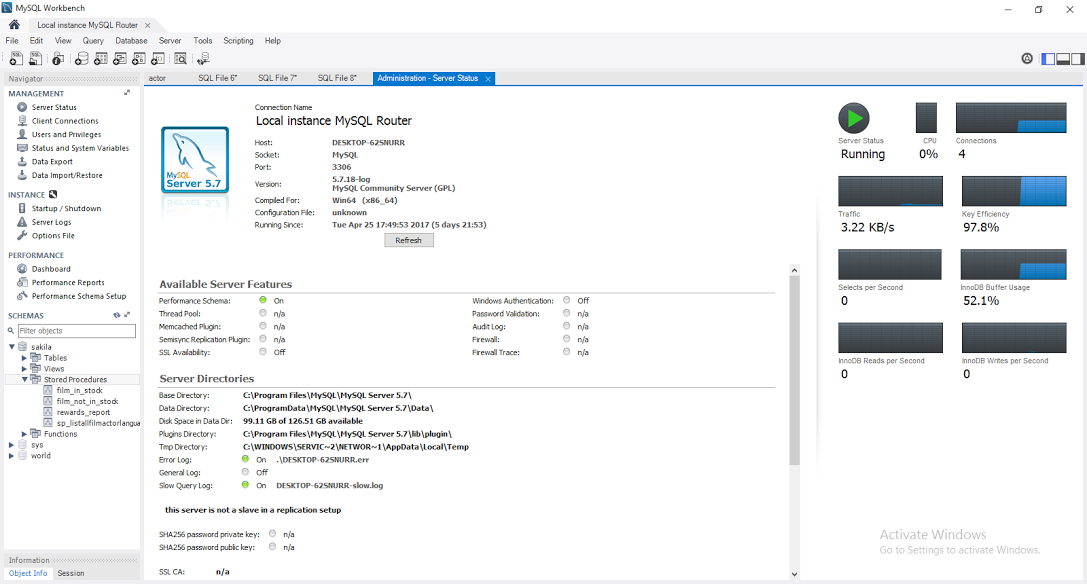
* Create server
* Enable firewall
* Create database bootcamp through pgadmin
* **Download Putty**: <http://www.putty.org/>
* **Login into to ubuntu VM**
* **Run as sudo**
  + sudo su
* **Download app**
  + git clone <https://github.com/vitorfs/bootcamp.git>
  + cd bootcamp
* **Install pre-reqs**
  + pip install -U -r requirements.txt
* **Change your connection string**
  + vim .env
  + Paste this in the .env file (change database credentials to yours)
    - DEBUG=True
    - SECRET\_KEY='mys3cr3tk3y'
    - DATABASE\_URL='postgres://pgdemouser@pgdemo:<passwd>@pgdemo.database.windows.net:5432/bootcamp'
* **Run migration and serve the app**
  + Python manage.py migrate
  + Python manage.py runserver 0.0.0.0:8000
* **Go to chrome/firefox and go to** http://<appidaddress**>**
* Play around with the app, go to **PgAdmin to show that it works**

This lab uses a simple MySQL Application as a test application. This simple application can be modified to test the various features of **Azure Database for MySQL.**

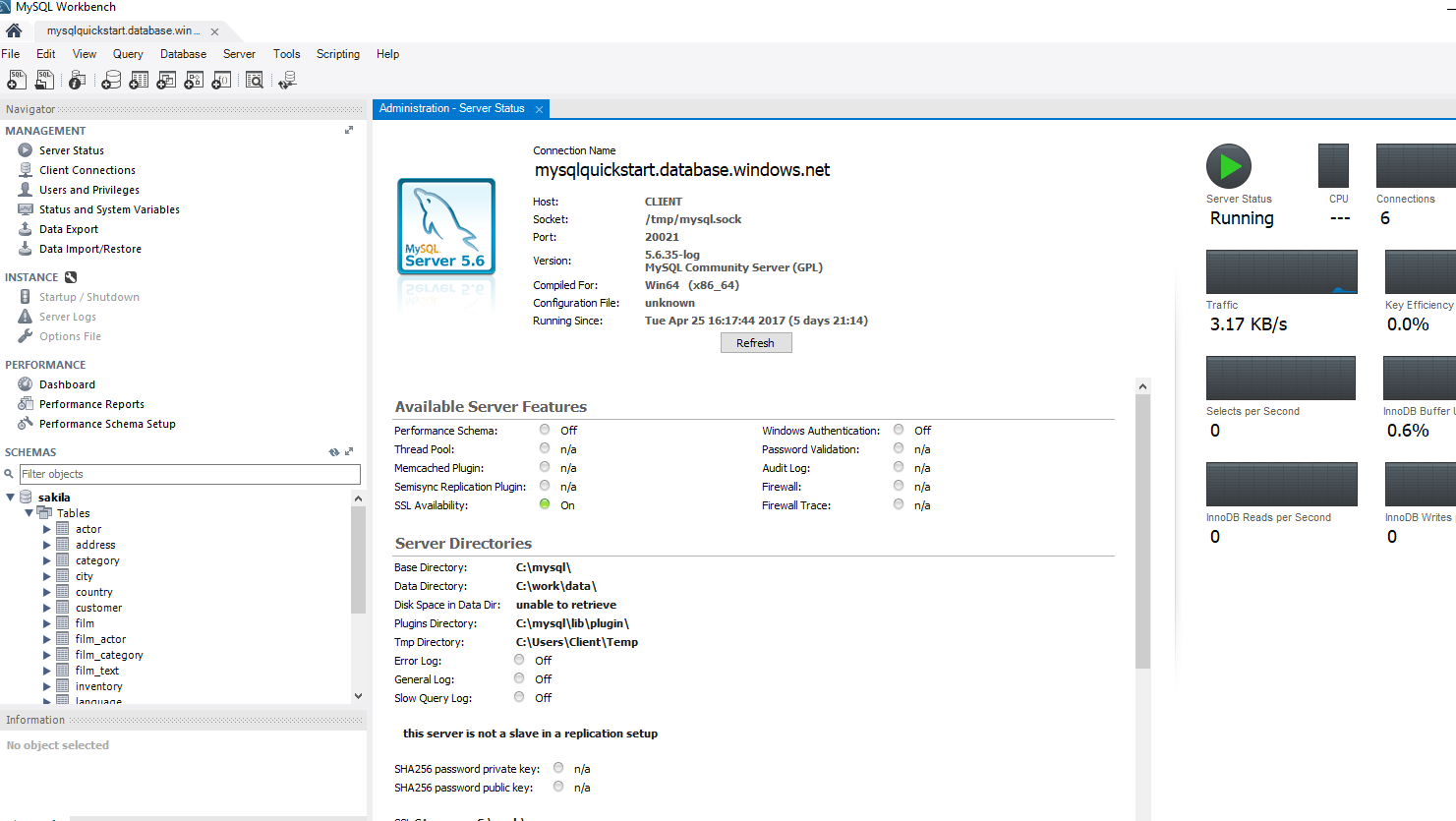
We have built an Azure Database for MySQL (sakila) as well as a Windows VM with a local MySQL Database(sakila) for the user. We also created all necessary tables with Primary, Foreign Keys, Constraints, Indexes and imported data into the respective databases.

MySQL is installed on a Windows VM including Workbench (mysql-installer-web-community-5.7.18.1). MySQL Workbench is also installed on a Local PC from where Azure Database for MySQL is connected.

**Below is the screenshot of a Windows VM with a Local MySQL Database using MySQL Workbench:**



**Below is the screenshot of an Azure Database for MySQL using MySQL Workbench:**



**Connection string details**

**Server:** **mysqlquickstart.database.windows.net**

**User:** quickstartuser@mysqlquickstart

**Password**: Quick$tart123

**Resource Group:** Build17MySQLQuickStart

**Location:** West US

**Port:**3306

**Note:** The Azure Database for MySQL that we will be querying during this lab was created via the Azure Portal. For more information on the Azure Portal refer to the **Appendix** at the bottom of the page.

Scenario 1: Creating and Running a Stored Procedure on both of above environments:

In this scenario, we will explain how to create/call or execute **a stored procedure**.

* Open MySQL WorkBench
* Call the stored procedure sakila.sp\_listallfilmactorlanguage by entering below in MySQL Workbench -> New Query -> call sakila.sp\_listallfilmactorlanguage and then execute.
* You will notice that you did not get any results since the stored procedure doesn’t exist - let's fix this by creating a new Stored Procedure on both environments.

DELIMITER //

CREATE PROCEDURE sakila.`sp\_listallfilmactorlanguage` ()

LANGUAGE SQL

DETERMINISTIC

SQL SECURITY DEFINER

COMMENT 'A procedure'

BEGIN

select FA.film\_id, FA.actor\_id, A.first\_name,A.last\_name, F.title, F.description,

F.release\_year,F.language\_id, L.name, F.rating

from

sakila.film\_actor FA

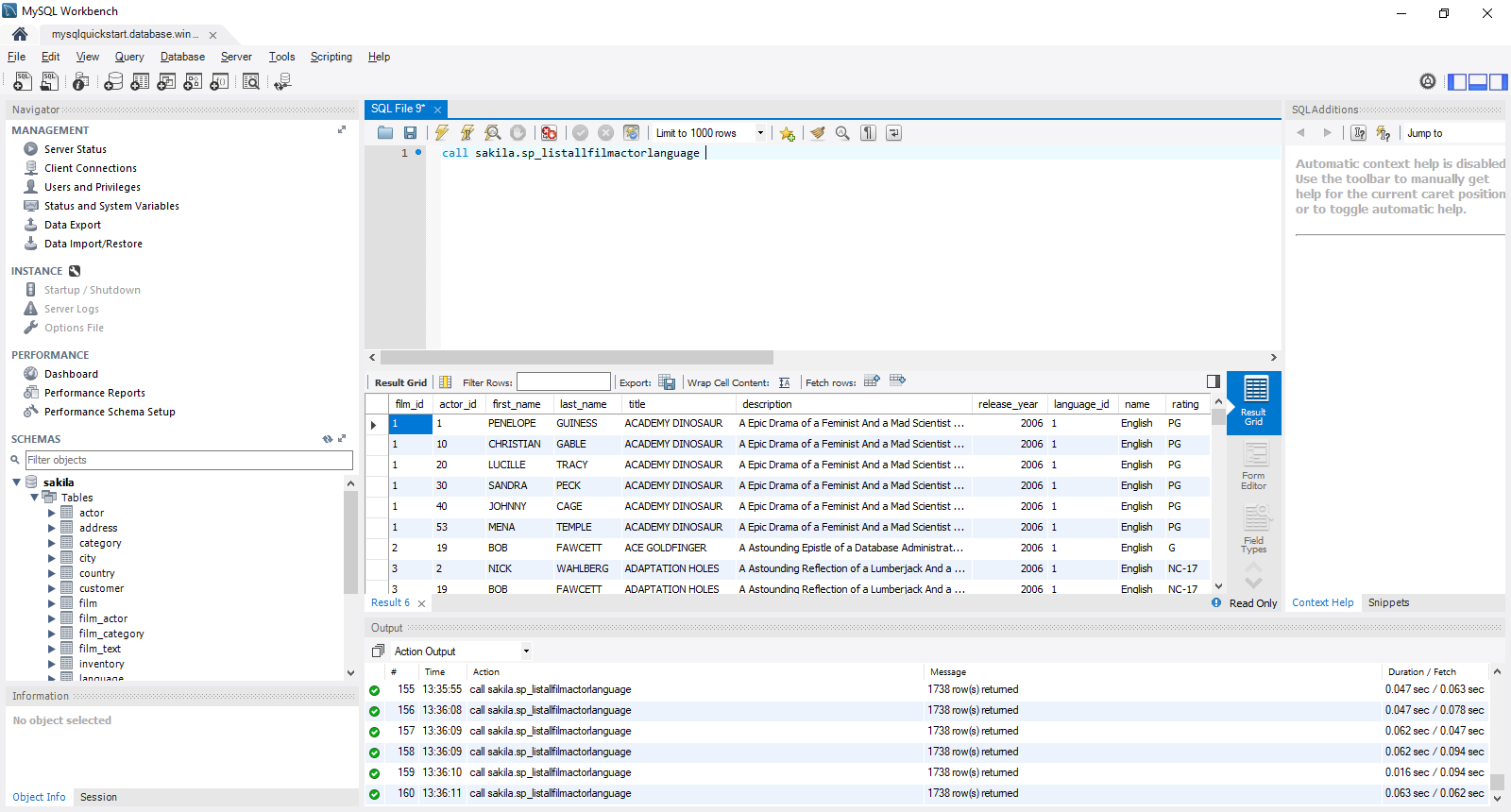
left outer join sakila.actor A on FA.actor\_id=A.actor\_id

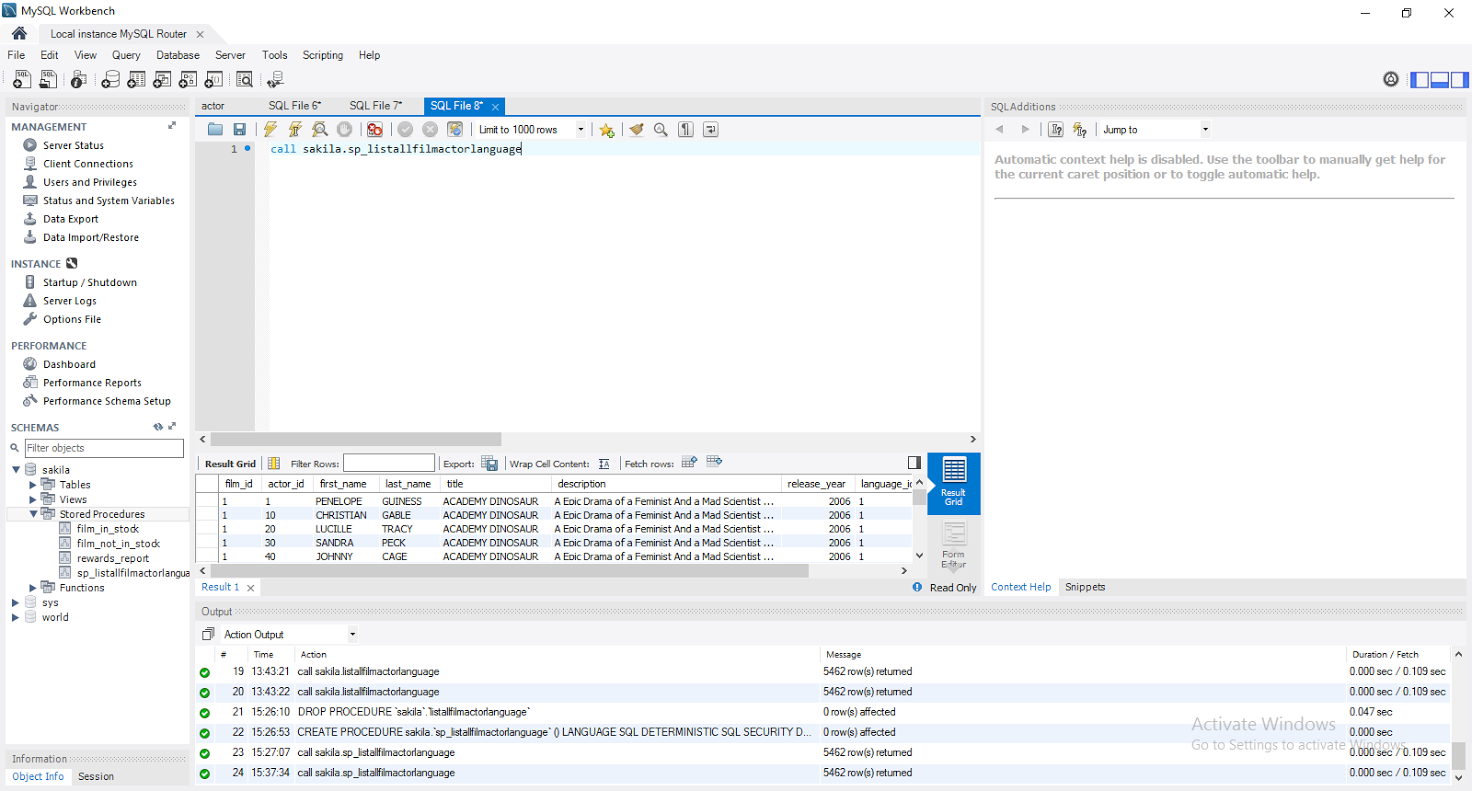
left outer join sakila.film F on FA.film\_id=F.film\_id

left outer join sakila.language L on L.language\_id=F.language\_id;

END//

It will display the result as below:





### 

## Scenario 2: Creating and retrieving data from a View on both of above environments:

In this scenario, we will explain how to create and retrieve the data from **a View**.

* Select the data from the view **sakila.v\_listallcustomerinfo** by entering below in MySQL Workbench -> **New Query -> SELECT \* FROM sakila.v\_listallcustomerinfo;** and then execute.
* You will notice that you did not get any results since the view doesn’t exist - let's fix this by creating a new view on both environments.

USE sakila;

CREATE view v\_listallcustomerinfo as

select C.customer\_id, C.first\_name, C.last\_name, C.store\_id, C.email, C.address\_id,

A.address,A.address2,A.city\_id, CT.city,

A.postal\_code, CT.country\_id,CN.country, A.phone

from

sakila.customer C

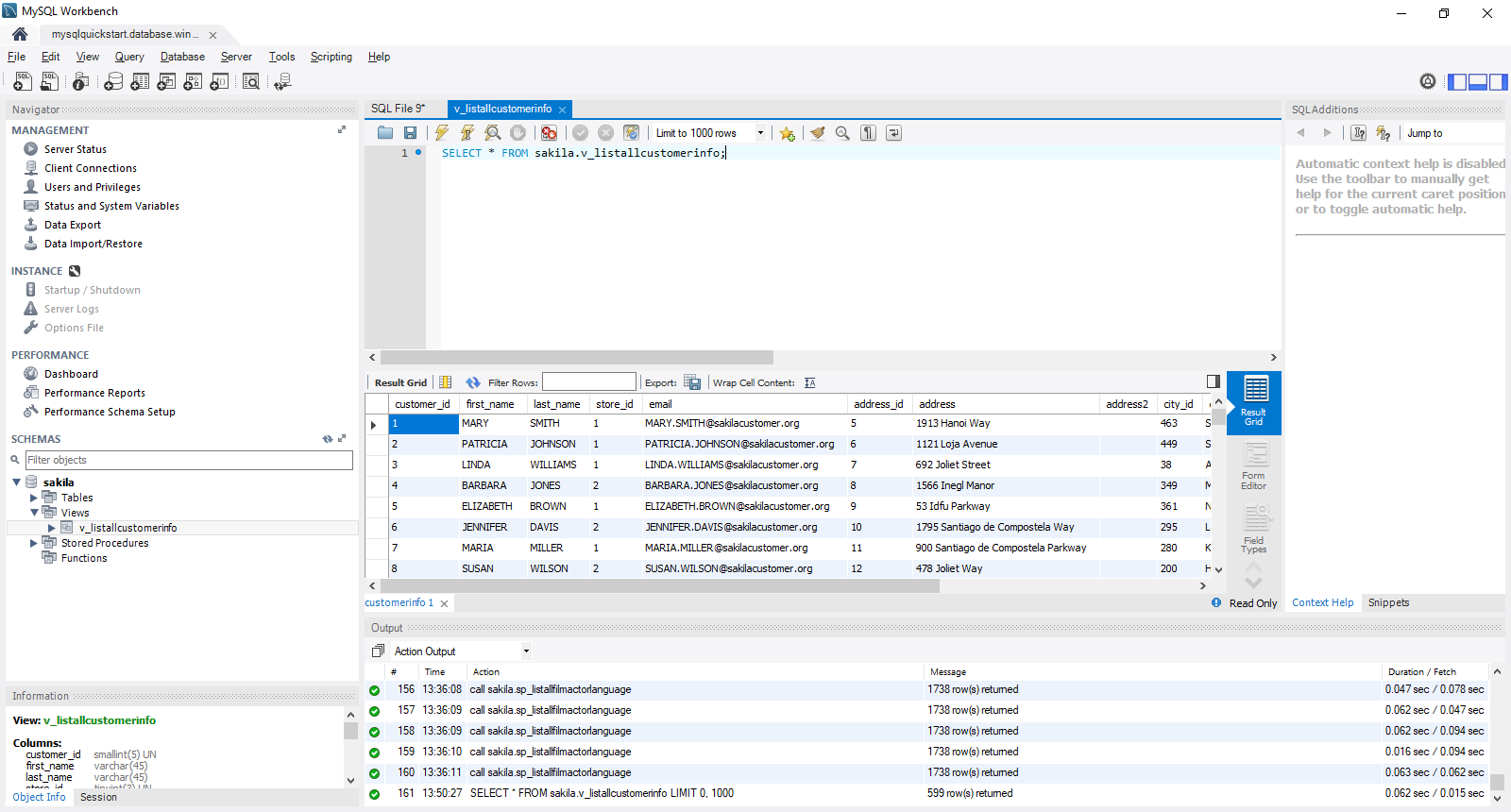
left outer join sakila.store S on C.store\_id=S.store\_id

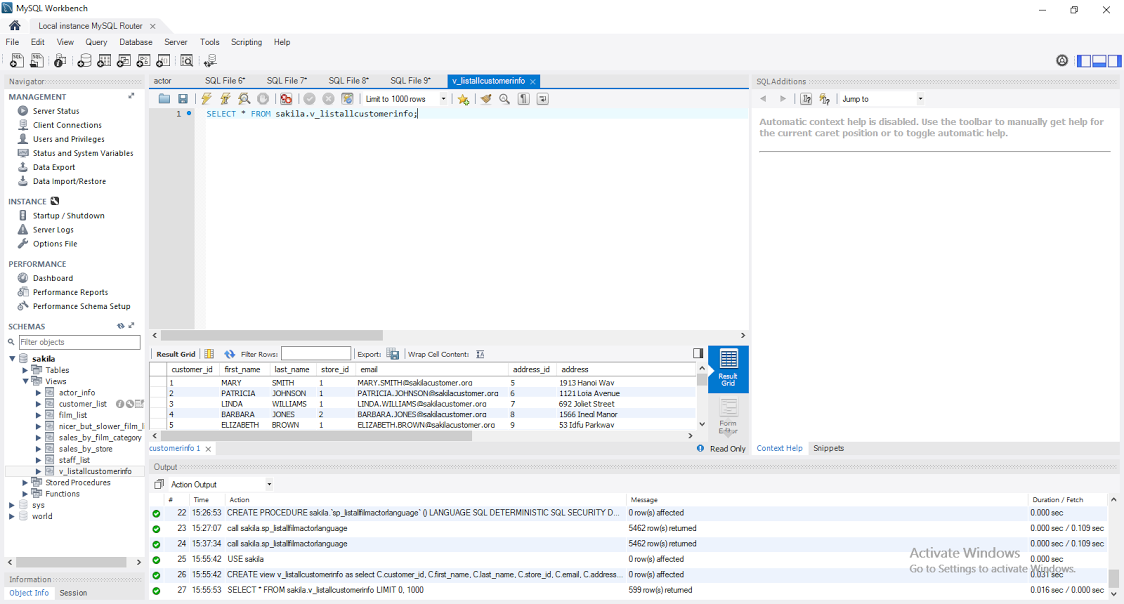
left outer join sakila.address A on C.address\_id=A.address\_id

left outer join sakila.city CT on A.city\_id=CT.city\_id

left outer join sakila.country CN on CT.country\_id=CN.country\_id;

**It will display the result as below:**





## Scenario 3: Creating Functions

In this scenario, we will explain how to create and call **a Function**.

* Select the data from the function sakila.v\_listallcustomerinfo by entering below in **MySQL Workbench -> New Query -> select fncustomercount ();** and then execute.
* You will notice that you did not get any results since the function doesn’t exist - let's fix this by creating a new function.

USE sakila;

DELIMITER \\

create DEFINER=`root`@`localhost` FUNCTION `fncustomercount`() RETURNS int(11)

BEGIN

declare vcuscount integer;

select count(customer\_id) into vcuscount from sakila.customer;

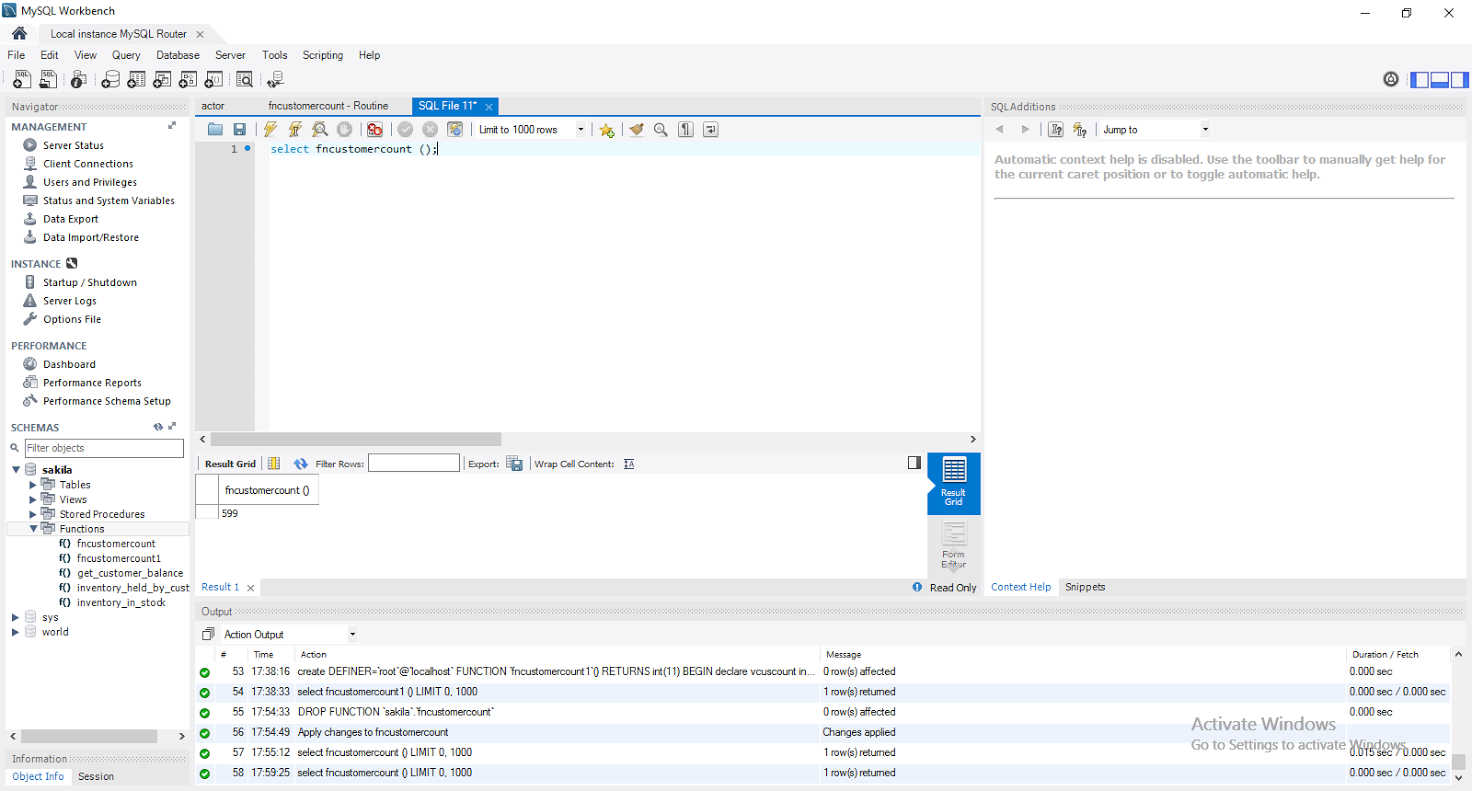
RETURN vcuscount;

END\\

DELIMITER ;

* Workbench -> New Query -> select fncustomercount ();

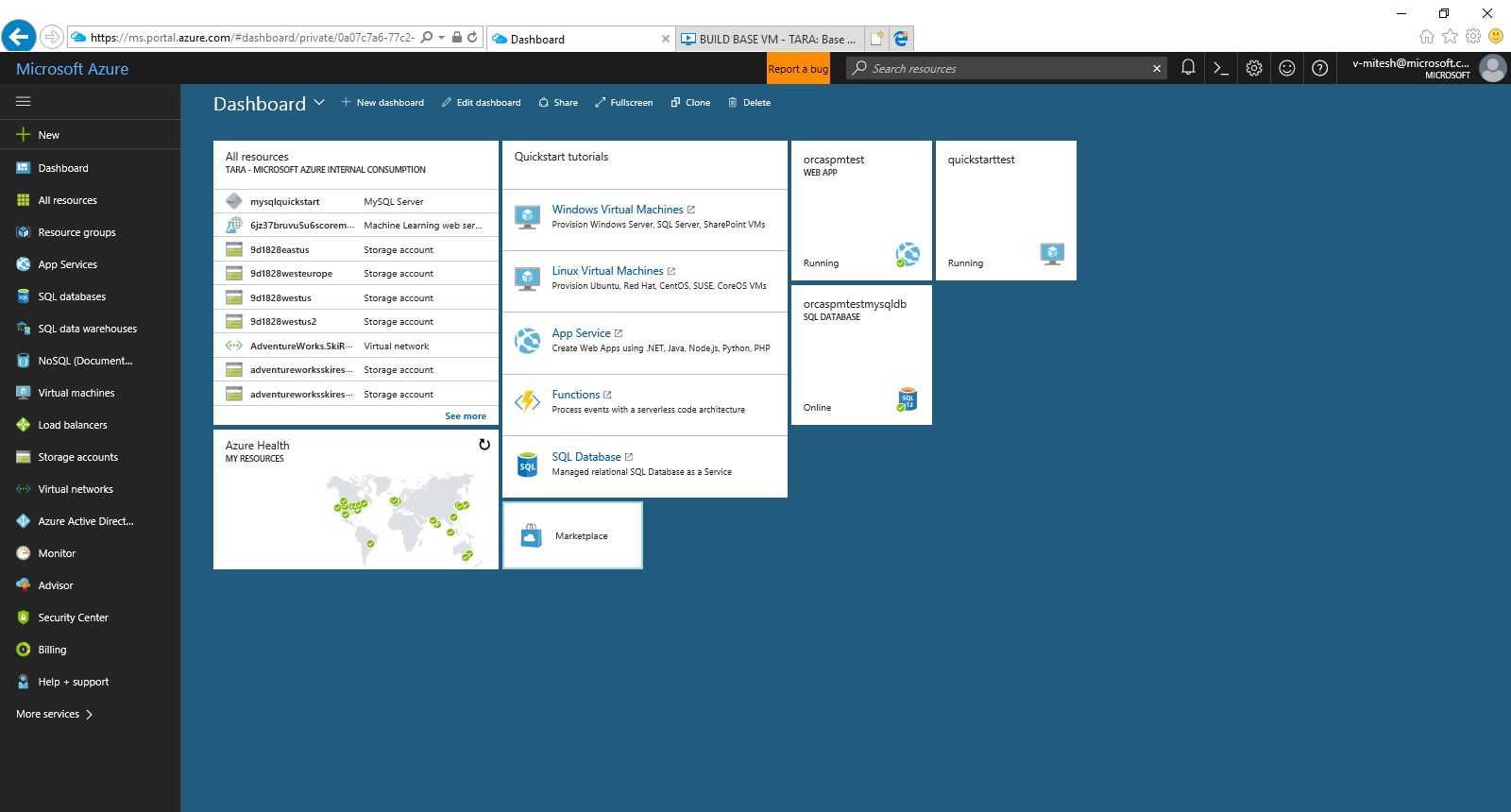
It will display the result as below:



## Appendix

The Azure Portal was used to create the Azure Search server. The Azure Portal can be found at <https://portal.azure.com/>.

**Dashboard:**



**Connection String:**

