# Department of Computer Science

**CS220: Database Systems**

**Class: BESE-13AB**

**Lab 02: SQL Operators**

# Date: September 18, 2023

# Time: 10:00-1:00 & 02:00-05:00

# Instructor: Dr. Bilal & Dr. Shah Khalid

# Lab Engineer: Sundas Dawood

**Name:** Aimen Munawar

**CMS ID:** 415867

**Class:** BESE-13-A

# CLO-2: Formulate SQL queries to retrieve information from a relational database.

**Lab 02: SQL Operators**

# Introduction

Structured Query Language (SQL) was developed at IBM San Jose Research Laboratory as a part of System R project. It is a declarative query language for querying a relational database. It also includes features for defining the structure of the data, for inserting and modifying data in the database, and for specifying security constraints. It is relational complete (it supports all six core relational algebra operations). SQL commands can be classified into three groups DDL, DML & DCL.

# Objectives

After performing this lab students should be able to:

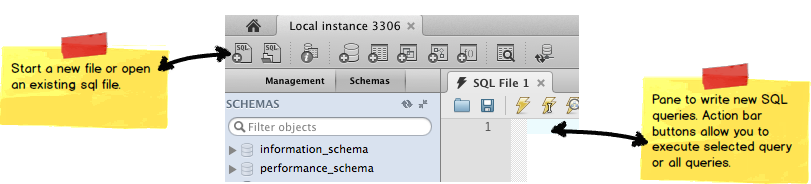
1. Design SQL queries to retrieve data using SELECT clause and various associated operators.
2. Design SQL queries with FROM & WHERE clause.
3. Execute SQL queries over MySQL using MySQL Workbench.

# Tools/Software Requirement

* MySQL Community Server
* MySQL Workbench
* Sakila Database

# Description

1. Open MySQL Workbench and open the default connection instance.
2. A new query window would open from where you can write and execute queries.



1. You can save the query file and can also add comments using # symbol.
2. On executing queries, results are displayed in the lower part of the screen.
3. Error or success messages are displayed in action output pane at the bottom.
4. Try running few SQL queries modeled during the lectures to get it going.
5. Continue playing with the Workbench and SQL queries till you are comfortable with the querying mechanism and have learnt the shortcuts to execute queries.

## SQL Basic Query Structure

### The SELECT clause

The most common use of the SQL commands is the selection of data from the tables located in a database. This can be achieved through SELECT command. We need to SELECT information FROM a table. Hence we have the most basic SQL query structure comprising of:

* SELECT
* FROM
* WHERE

The syntax for SELECT clause is as follows:

**SELECT "column\_name(s)" FROM "table\_name(s)";**

Always specify the name of the database in which a table/relation is present through which data is to be retrieved.

***e.g. Select column\_name***

***From Sakila.table\_name***

There are three ways we can retrieve data from a table:

* Retrieve one column
* Retrieve multiple columns
* Retrieve all columns (Use \*)

The select clause can contain arithmetic expressions involving the operation, +, –, \*, and /, and operating on constants or attributes of records (tuples).

### The FROM clause

The FROM clause can allow to select attributes from single table or multiple tables. When multiple tables are applied, it combines the records from the two or more tables listed and presents every possible combination of the listed attributes in SELECT clause.

This is not very useful until some filtering condition is applied. This is achieved through using the WHERE clause.

### The WHERE clause

We can use the WHERE clause to filter the result set based on certain conditions. The syntax for using WHERE in the SELECT statement is as follows:

**SELECT "column\_name(s)"  
FROM "table\_name(s)"  
WHERE "condition";**

"Condition" can include a single comparison clause (called simple condition) or multiple comparison clauses combined together using AND or OR operators (compound condition). Conditions can include other operators like IN, BETWEEN, DISTINCT etc shown in Table 1:

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| = | Equal to | Author = 'Alcott' |
| <> | Not equal to (most DBMS also accept != instead of <>) | Dept <> 'Sales' |
| > | Greater than | Hire\_Date > '2012-01-31' |
| < | Less than | Bonus < 50000.00 |
| >= | Greater than or equal | Dependants >= 2 |
| <= | Less than or equal | Rate <= 0.05 |
| BETWEEN | Between an inclusive range | Cost BETWEEN 100.00 AND 500.00 |
| LIKE | Match a character pattern | First\_Name LIKE 'Will%' |

Table 1: List of Basic Operators available in SQL

## Ordering data

The order of rows returned in a query result is undefined. The ORDER BY clause can be used to sort the rows. This clause comes last in the SELECT statement. ASC at the end of the ORDER BY clause specifies ascending order where as DESC specifies descending order. ASC is the default order. The syntax for an ORDER BY statement is as follows:

**SELECT "column\_name"  
FROM "table\_name"  
[WHERE "condition"]  
ORDER BY "column\_name" [ASC, DESC];**

## The Wildcard operators

There are times when we want to match on a string pattern. To do that, we will need to employ the concept of wildcard. In SQL, there are two wildcards:

* % (percent sign) represents zero, one, or more characters.
* \_ (underscore) represents exactly one character.

Wildcards are used with the LIKE keyword in SQL.

Below are some wildcard examples:

* 'A\_Z': All string that starts with 'A', another character, and end with 'Z'. For example, 'ABZ' and 'A2Z' would both satisfy the condition, while 'AKKZ' would not (because there are two characters between A and Z instead of one).
* 'ABC%': All strings that start with 'ABC'. For example, 'ABCD' and 'ABCABC' would both satisfy the condition.
* '%XYZ': All strings that end with 'XYZ'. For example, 'WXYZ' and 'ZZXYZ' would both satisfy the condition.
* '%AN%': All strings that contain the pattern 'AN' anywhere. For example, 'LOS ANGELES' and 'SAN FRANCISCO' would both satisfy the condition.
* '\_AN%': All strings that contain a character, then 'AN', followed by anything else. For example, 'SAN FRANCISCO' would satisfy the condition, while 'LOS ANGELES' would not satisfy the condition.

# Lab Task

## Using Sakila Database

Formulate SQL queries for the following needs and execute them on Sakila database.

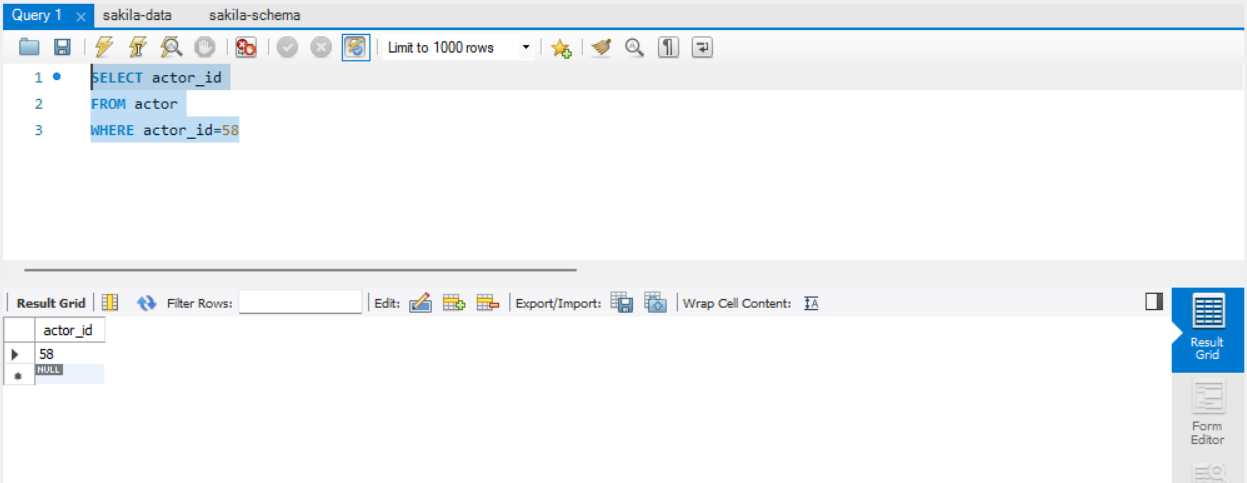
1. Write a query to Select column “actor\_id” from the table “actor” which already exists in the “Sakila” database where the “actor\_id=58”. (Observe the output when you execute the queries.)

**QUERY:**

**SELECT actor\_id**

**FROM actor**

**WHERE actor\_id=58**

****

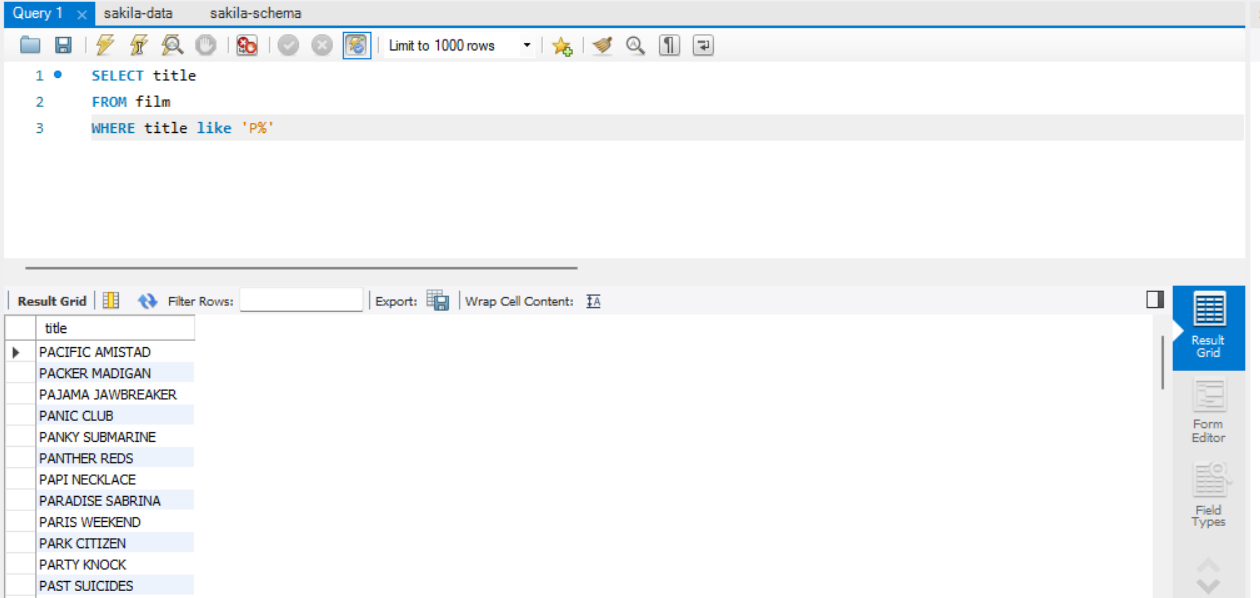
1. Write a query to retrieve names of movies starting with P.

**QUERY:**

**SELECT title**

**FROM film**

**WHERE title like 'P%'**

****

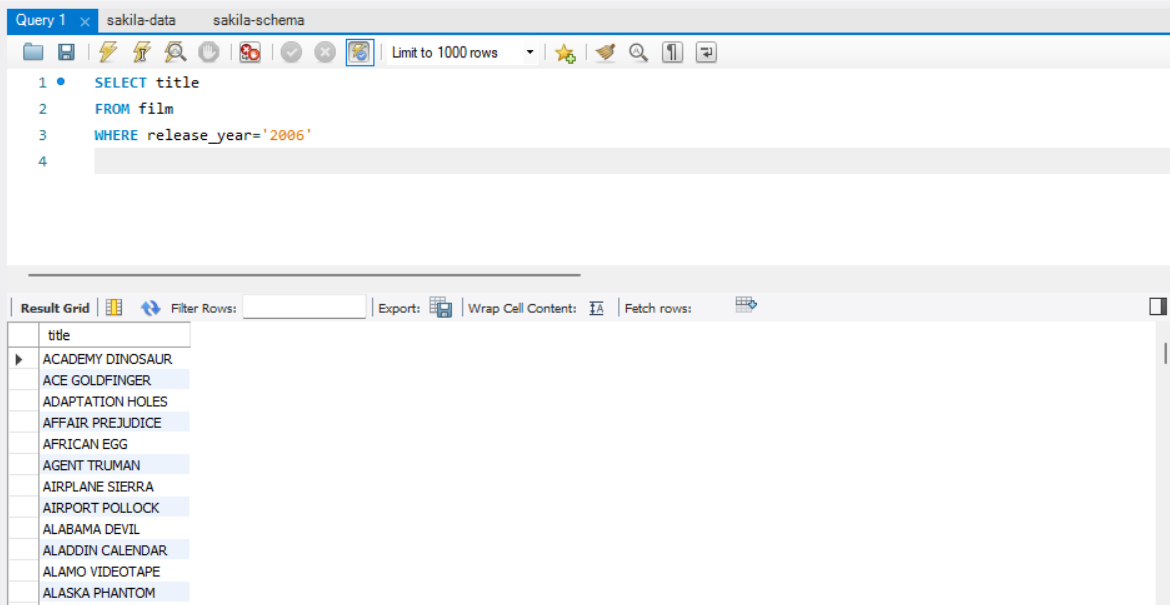
1. Write a query to retrieve movies which were released in year 2006?

**QUERY:**

**SELECT title**

**FROM film**

**WHERE release\_year='2006'**

****

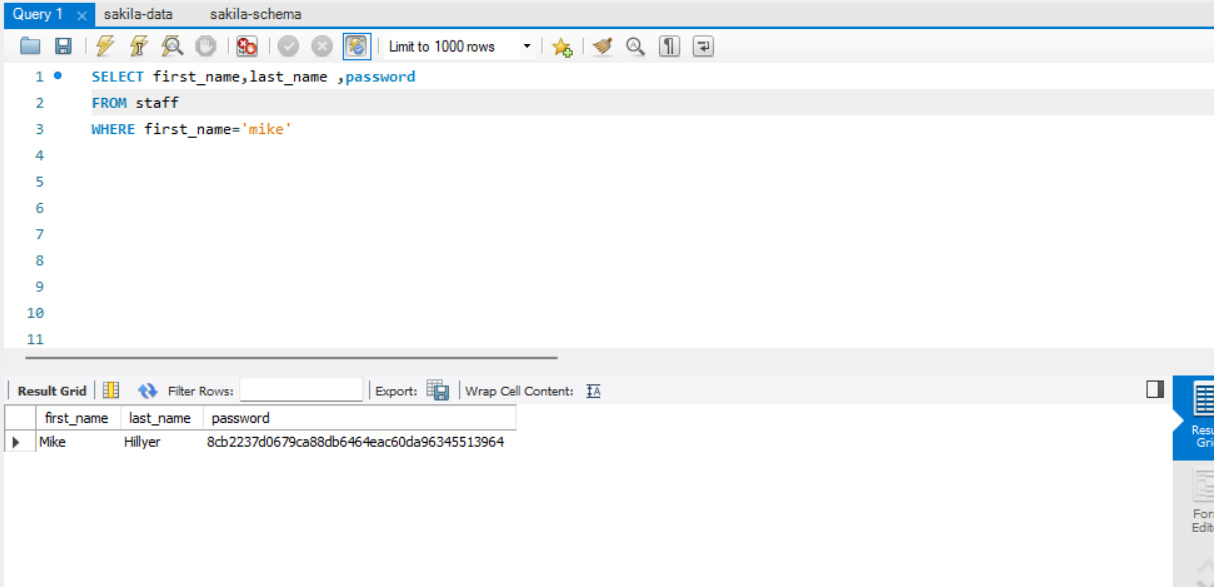
1. What password did the DBA assign to the user „MIKE‟?

**QUERY:**

**SELECT first\_name,last\_name ,password**

**FROM staff**

**WHERE first\_name='mike'**

****

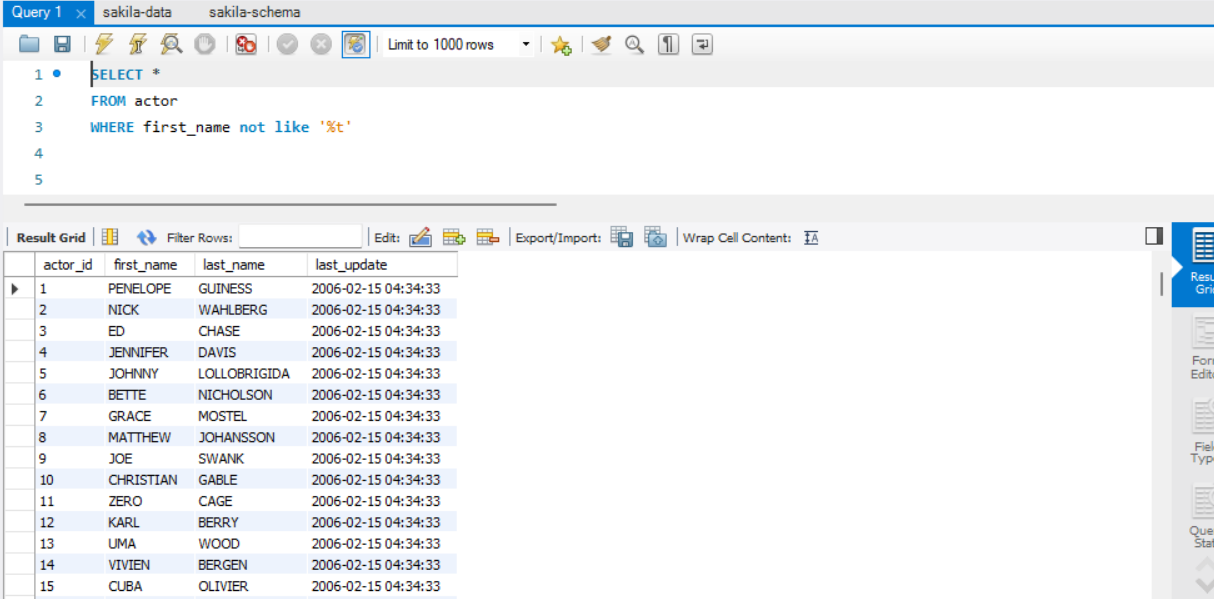
1. Write a query to retrieve data of all actors whose first names are not ending on T.

**QUERY:**

**SELECT \***

**FROM actor**

**WHERE first\_name not like '%t'**

****

1. Find and display the Income (payments) generated during August 2005. Sort them in the descending order.

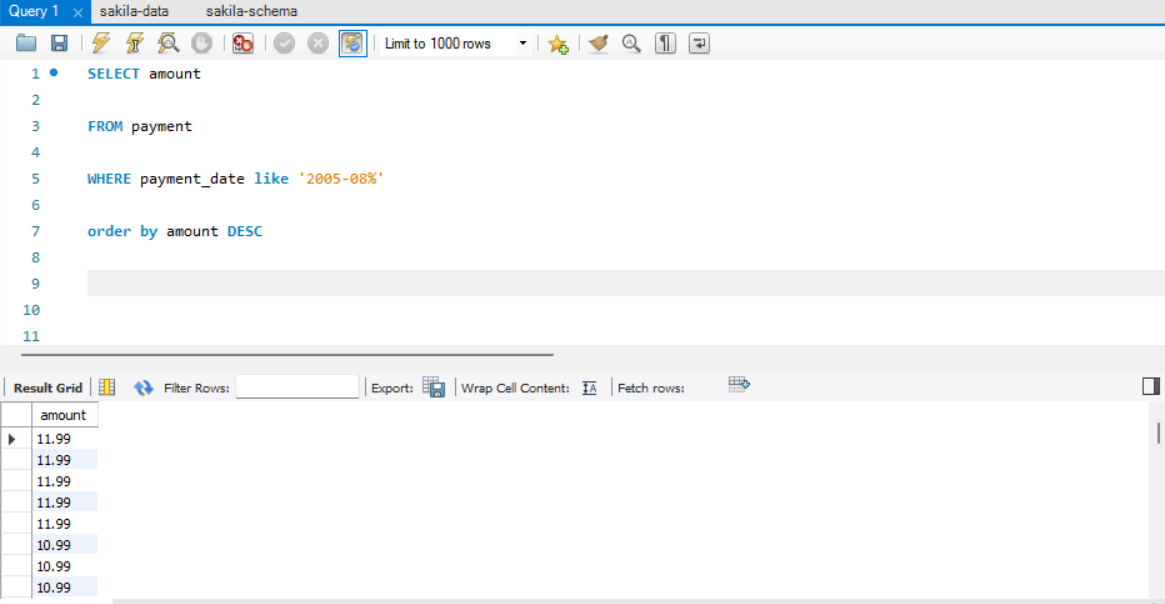
**QUERY:**

**SELECT amount**

**FROM payment**

**WHERE payment\_date like '2005-08%'**

**order by amount DESC**



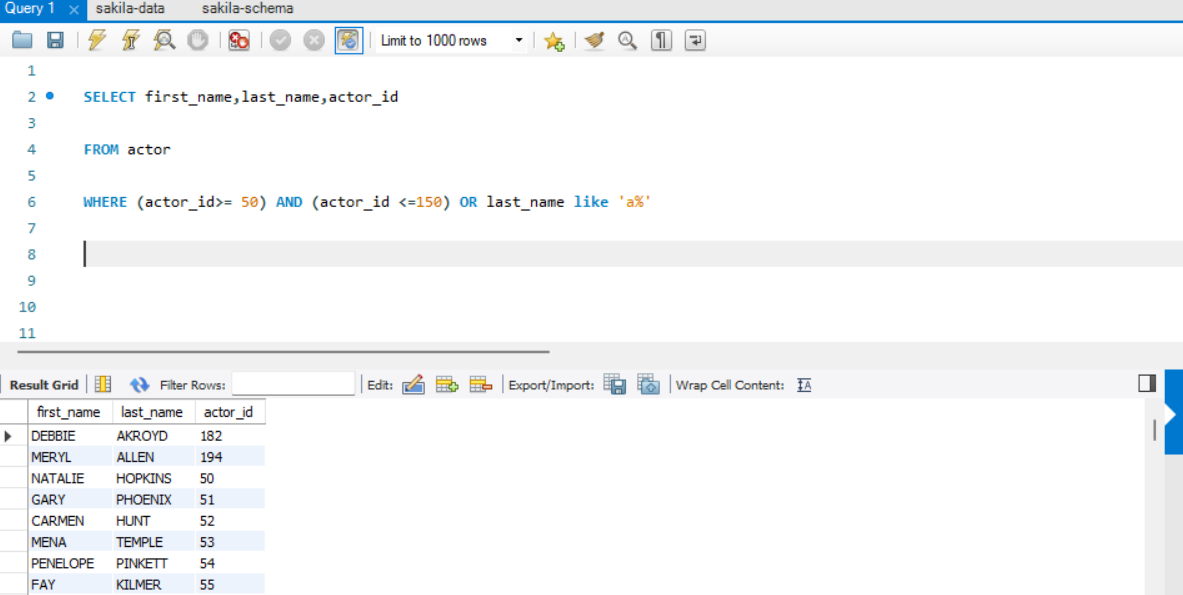
1. Select the names of actors whose IDs are between 50 and 150, or those whose last name starts with A.

**QUERY:**

**SELECT first\_name,last\_name,actor\_id**

**FROM actor**

**WHERE (actor\_id>= 50) AND (actor\_id <=150) OR last\_name like 'a%'**

****

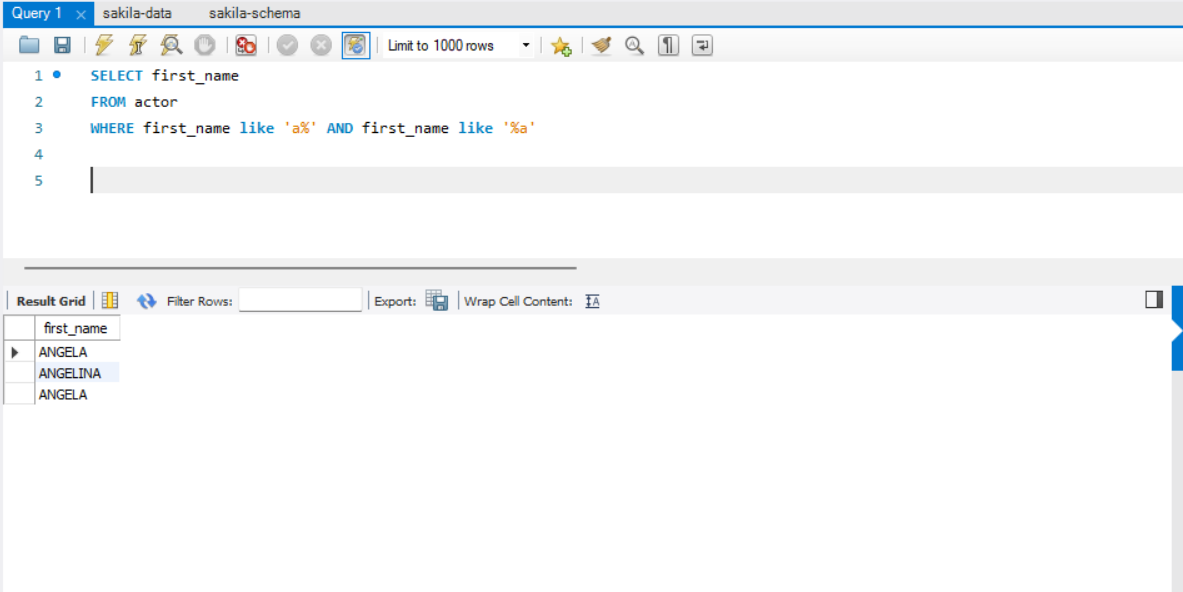
1. Display the actor first name which start with A and end on letter A.

**QUERY:**

**SELECT first\_name**

**FROM actor**

**WHERE first\_name like 'a%' AND first\_name like '%a'**



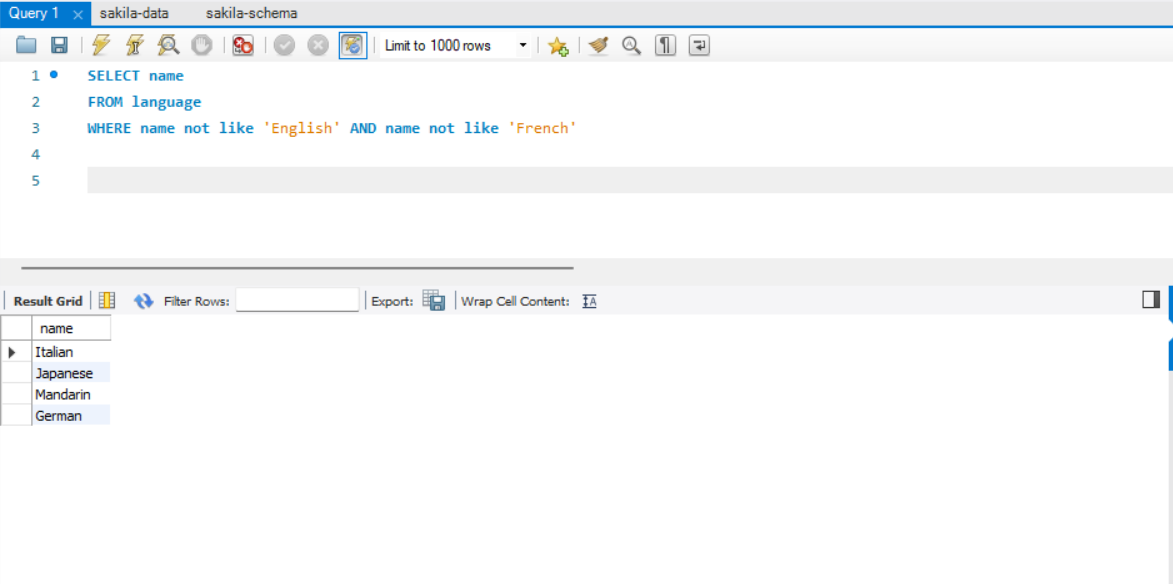
1. Write an SQL query to display list of languages which neither is English nor French.

**QUERY:**

**SELECT name**

**FROM language**

**WHERE name not like 'English' AND name not like 'French'**



# Deliverable

Submit a PDF document including the SQL queries to answer above-mentioned information needs as well as snapshot of their outcome when executed over MySQL using the Workbench.