Table of Contents

[QUERIES EXECUTION Part-1 1](#_Toc165648657)

[QUERY 1 1](#_Toc165648658)

[QUERY 2 1](#_Toc165648659)

[TASK 1 2](#_Toc165648660)

[TASK 2 3](#_Toc165648661)

[TASK 3 4](#_Toc165648662)

[Task4 4](#_Toc165648663)

[Task5 4](#_Toc165648664)

[Task6 5](#_Toc165648665)

[EER DIAGRAM 5](#_Toc165648666)

[QUERIES EXECUTION Part-2 8](#_Toc165648667)

[TASK 1 9](#_Toc165648668)

[TASK 2 9](#_Toc165648669)

[TASK 3 10](#_Toc165648670)

[Task4 10](#_Toc165648671)

[Task5 11](#_Toc165648672)

[Task6 12](#_Toc165648673)

[Task7 12](#_Toc165648674)

[Task8 13](#_Toc165648675)

[Task9 14](#_Toc165648676)

[EER DIAGRAM 14](#_Toc165648677)

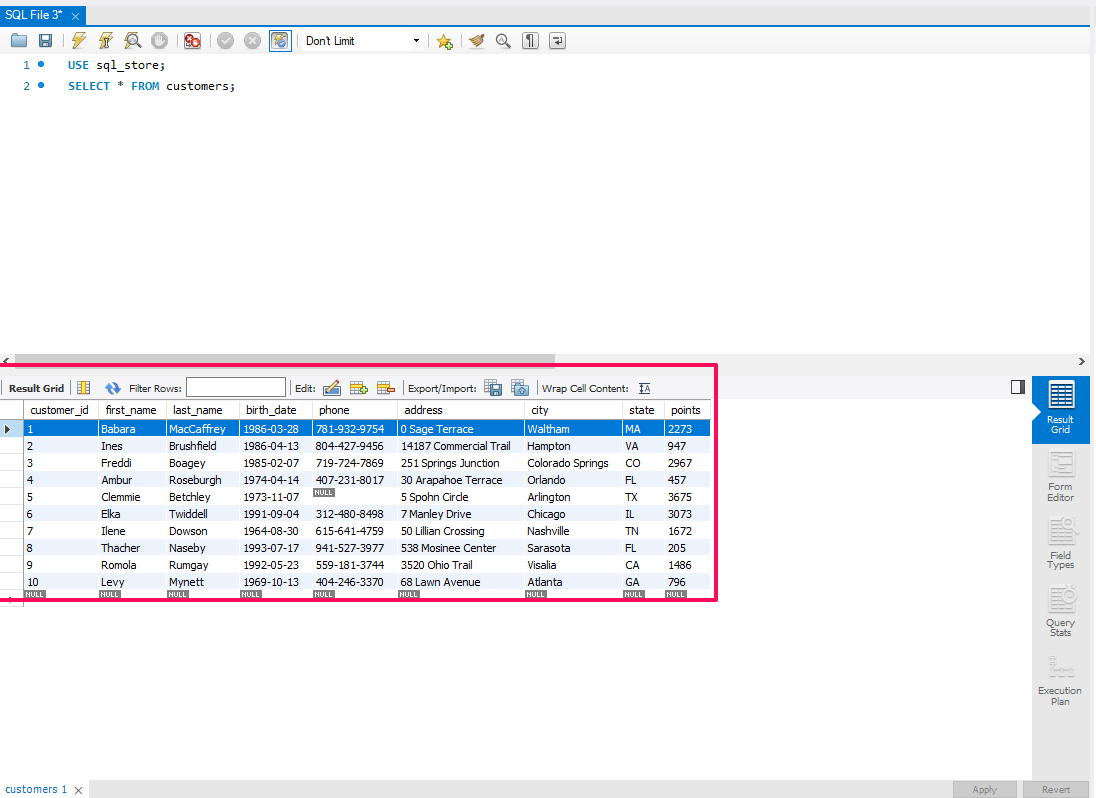
[Task 10 17](#_Toc165648678)

# QUERIES EXECUTION Part-1

## QUERY 1

1. Select all the data from the customers table contained in the store database:  
   USE sql\_store;

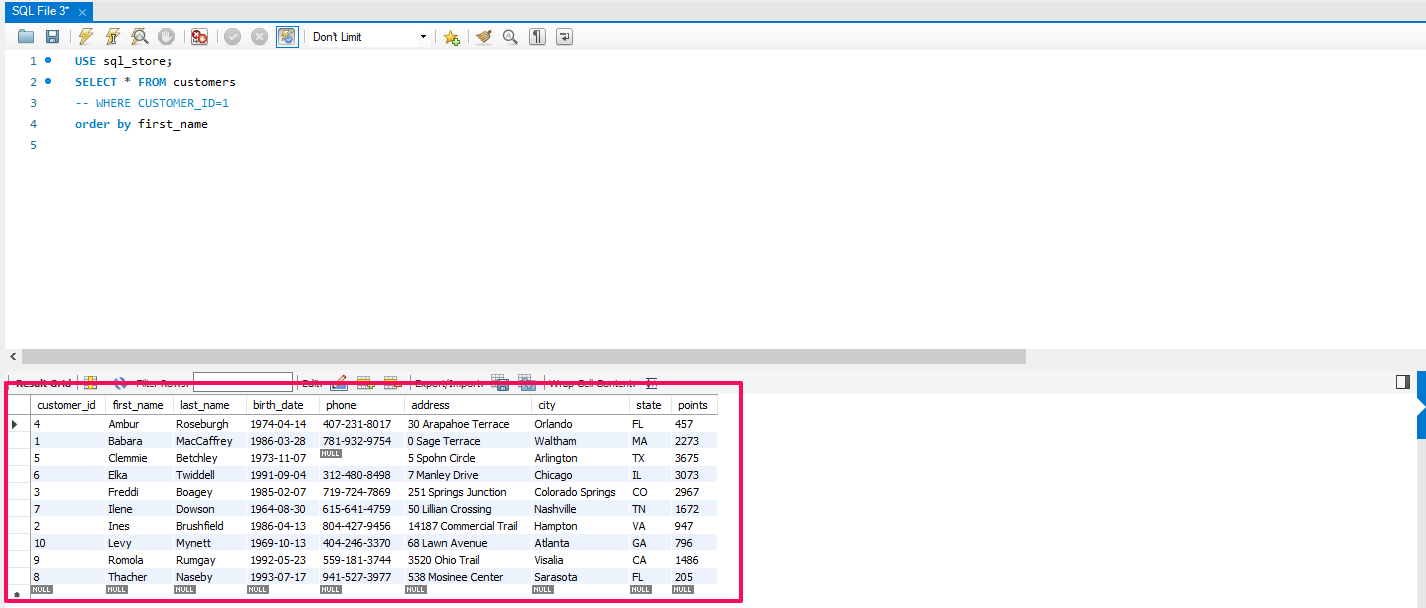
SELECT \* FROM customers;



1. Add the following into the file Query 1

Order customers alphabetically ORDER BY first\_name

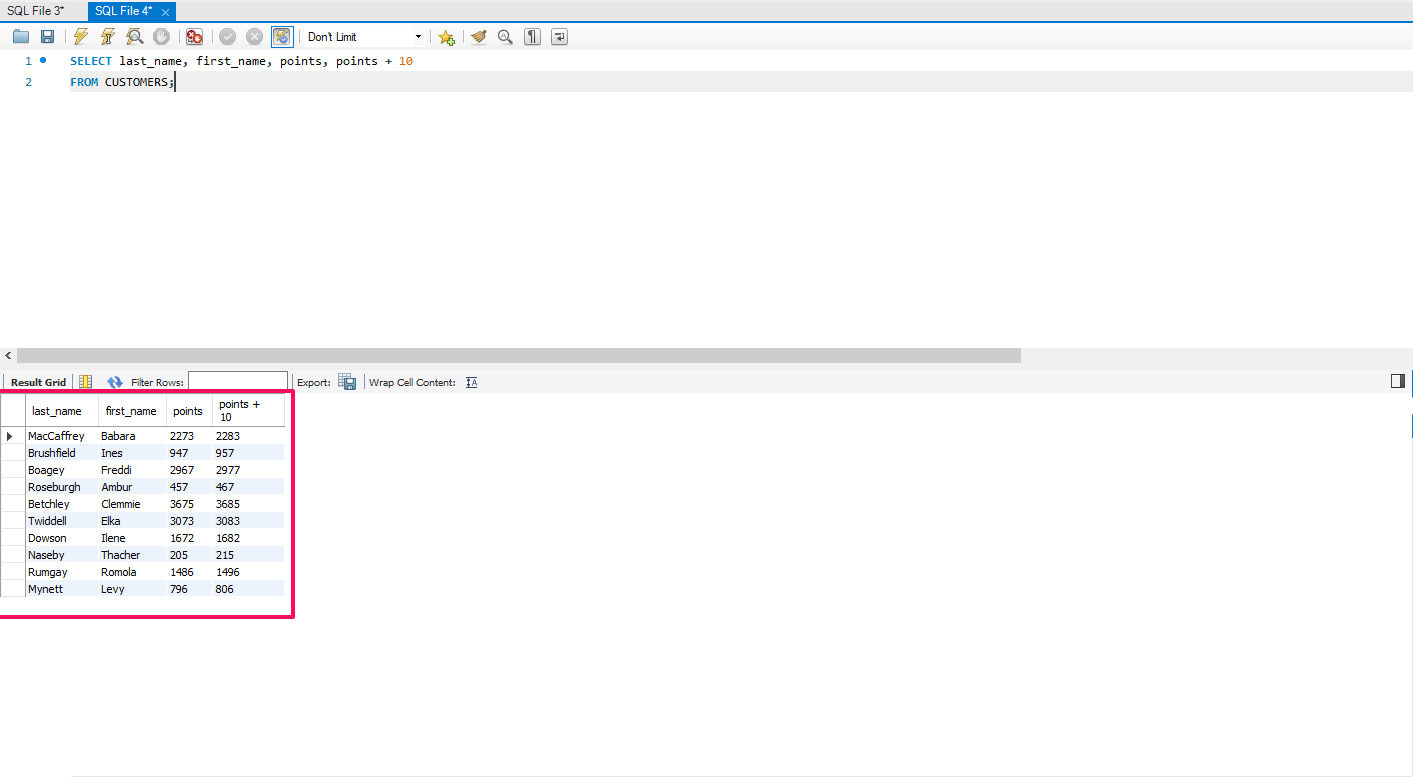
SELECT \* FROM customers ORDER BY first\_name;



## QUERY 2

1. Extract last\_name, first\_name, points and compute points +10 column

SELECT last\_name, first\_name, points, points + 10 FROM customers;



# TASK 1

Using the Query 2 we created change the points to read times by 10 and plus 100 points.

calculates the adjusted points by first multiplying the points by 10 and then adding 100 to the result.

SELECT last\_name, first\_name, points, points \* 10 + 100 FROM customers;

A screenshot of a computer

Description automatically generated

**Task:** Change the Query 2 code to create a discount factor so the table now shows a discount header and changing the (point + 10) \*100.

The result is aliased as 'discount\_factor' in the query result.

SELECTlast\_name, first\_name, points, (points + 10) \* 100 as discount\_factor. It calculates a new value based on the points column by first adding 10 and then multiplying the result by 100.

FROMcustomers;

A screenshot of a computer

Description automatically generated

# TASK 2

Write a SQL query to return all the products in our database in the result set. I want to make three new columns, name, unit price, and new column called new price, which is based on this expression, (unit price \* 1.1). what you are doing is increasing the product price of each by 10%.

So, with the query we want to retrieve all the products showing the original unit price and increased price. In this Query, the new\_price column is calculated by multiplying the unit\_price by 1.1, which increases the price.

SELECTname, unit\_price,

unit\_price \* 1.1 AS new\_price

FROMproducts;

A screenshot of a computer

Description automatically generated

# TASK 3

In this task create a new query to find all the customers with a birth date of > '1990-01-01'

Created a query which extracts data of customers which were born after 1st of Jan 1990

SELECT \* FROMcustomers

WHEREbirth\_date> '1990-01-01';

A screenshot of a computer

Description automatically generated

# Task4

Write a query to find out the name of the product with most amount in stock.

This query selects the name of the product with the highest quantity in stock from the Products table in sql\_inventory schema.

SELECT name FROMproducts

ORDER BY

quantity\_in\_stockDesc Limit 1;

# Task5

Write a query to find out the name of the most expensive product.

SELECT name, unit\_price : This selects the column name of the product and unit\_price from the table.

FROM products : This specifies the table products from which we are selecting data.

SELECT name, unit\_price

FROM products

Order BY unit\_priceDesc Limit 1;

# Task6

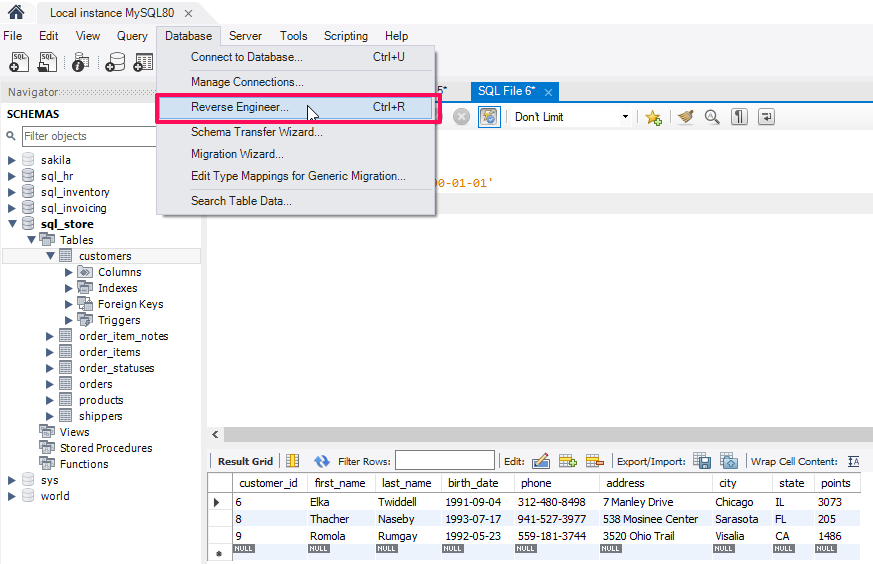
Select sql\_store. Write a query to find out the first name, last name, address, and the birthdate of the oldest customer.

This query retrieves the first name, last name, address, and birthdate of the oldest customer from the sql\_store table.

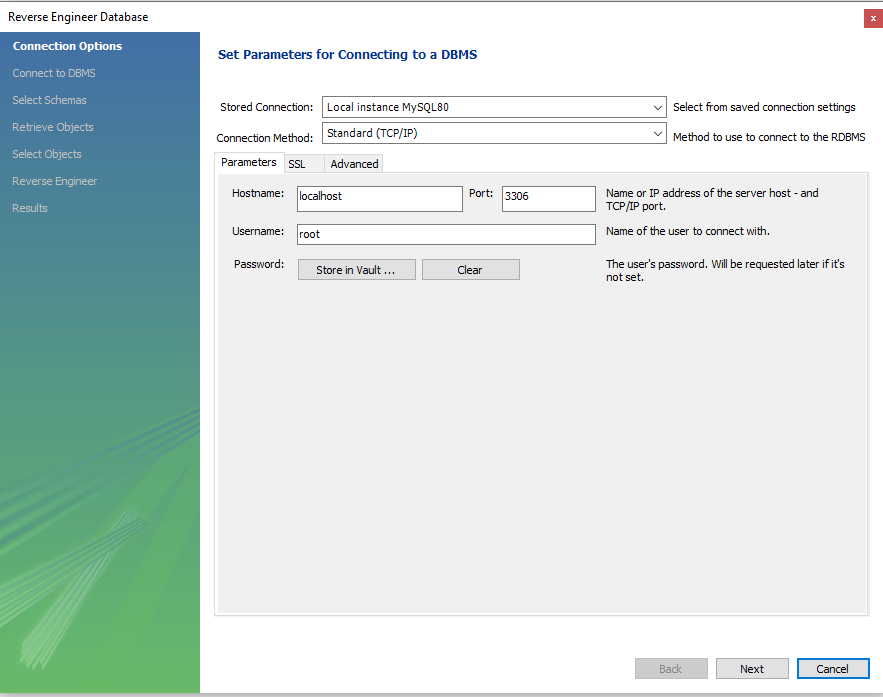
SELECTfirst\_name, last\_name, address, birth\_date FROM customers Order by birth\_dateAsc Limit 1;

# EER DIAGRAM

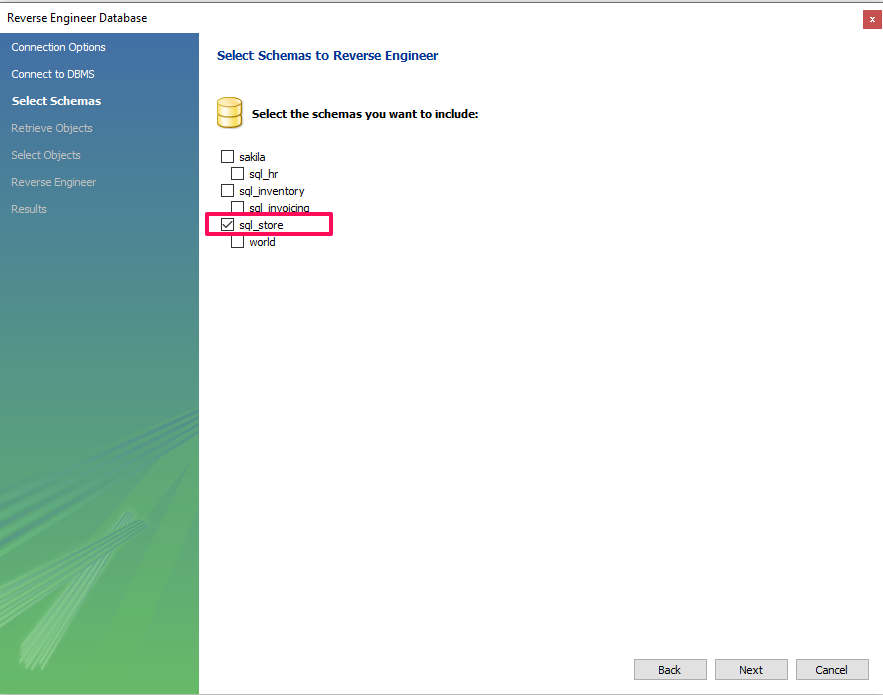
1. Create EER Diagram – navigate to Database tab, choose Reverse Engineer option.



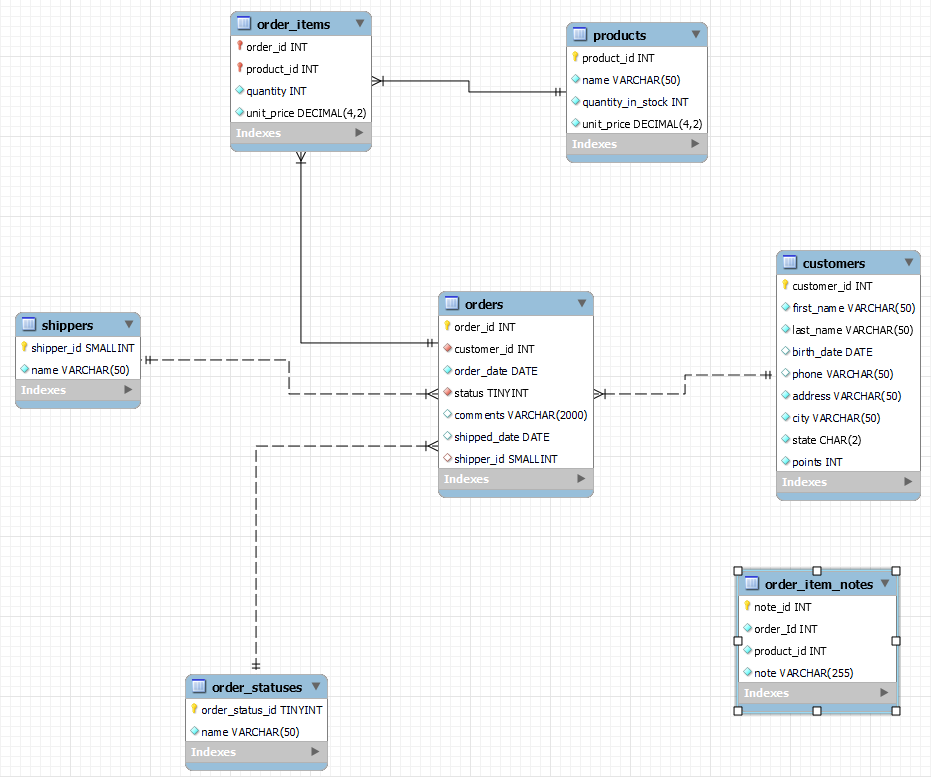
1. Navigate through setup wizard



1. We want to include the store schema



1. EER Diagram Result:



# QUERIES EXECUTION Part-2

# TASK 1

Using count, get the number of cities in the USA.

This query will count the number of rows in the cities table where the country column is equal to 'USA', giving you the number of cities in the USA.

SELECT COUNT(Countrycode)

FROM city

WHERECountrycode = 'USA';

Result:

A screenshot of a computer

Description automatically generated

# TASK 2

Find out what the population and average life expectancy for people in Argentina (ARG) is.

This query will calculate the average life expectancy and total population for people in Argentina by selecting data from the countries table and filtering for the country code 'ARG'.

SELECT code, AVG(LifeExpectancy), SUM (population)

FROM country

WHERE code = 'ARG';

A screenshot of a computer

Description automatically generated

# TASK 3

Using ORDER BY, LIMIT, what country has the highest life expectancy?

This query will order the countries by life expectancy in descending order and then limit the result to only one row, which corresponds to the country with the highest life expectancy.

SELECT name, LifeExpectancy

FROM country

ORDER BY LifeExpectancy DESC

LIMIT 1;

A screenshot of a computer

Description automatically generated

# Task4

Select 25 cities around the world that start with the letter 'F' in a single SQL query.

This query selects all columns from the city table where the Name column starts with the letter 'F’ and limits the result to 25 rows.

SELECT \* FROM city

WHERE Name LIKE 'F%'

LIMIT 25;

A screenshot of a computer

Description automatically generated

# Task5

Create a SQL statement to display columns Id, Name, Population from the city table and limit results to first 10 rows only.

This query selects specific columns (Id, Name, Population) from the city table and restricts the result to the first 10 rows.

SELECT Id, Name, Population, FROM city LIMIT 10;

A screenshot of a computer

Description automatically generated

# Task6

Create a SQL statement to find only those cities from city table whose population is larger than 2000000.

This query selects all columns from the city table and filters the result to only include cities where the Population column is larger than 2,000,000.

SELECT \* FROM city WHERE Population >2000000;

A screenshot of a computer

Description automatically generated

# Task7

Create a SQL statement to find all city names from city table whose name begins with “Be” prefix.

This query selects the Name column from the city table and filters the result to only include city names that begin with the prefix "Be" using the LIKE operator with the % wildcard character, which matches any sequence of characters.

SELECT Name FROMcity

WHERE Name LIKE 'Be%';

A screenshot of a computer

Description automatically generated

# Task8

Create a SQL statement to find only those cities from city table whose population is between 500000-1000000.

This query selects all columns (\*) from the city table and filters the result to only include cities where the Population column falls within the range of 500,000 to 1,000,000 using the BETWEEN operator. The BETWEEN operator is inclusive of both endpoints, so it will include cities with a population of exactly 500,000 or 1,000,000 in the result set.

SELECT \* FROM city WHERE Population BETWEEN 500000 AND 1000000;

A screenshot of a computer

Description automatically generated

# Task9

Create a SQL statement to find a city with the lowest population in the city table.

This query selects all columns (\*) from the city table, orders the result by population in ascending order (ASC), which will put the cities with the lowest population at the top, and limits the result to only one row using LIMIT 1. Therefore, it will return the city with the lowest population.

SELECT \* FROM city ORDER BY Population ASC LIMIT1;

A screenshot of a computer

Description automatically generated

# EER DIAGRAM

1. Create EER Diagram – navigate to Database tab, choose Reverse Engineer option.

A screenshot of a computer

Description automatically generated

1. Navigate through setup wizard.

A screenshot of a computer

Description automatically generated

1. We want to include the store schema.

A screenshot of a computer

Description automatically generated

1. EER Diagram Result:

A screenshot of a computer

Description automatically generated

# Task 10

1. The Primary Key in the Country Table is ‘**Code**’
2. The Primary Key in the City Table is ‘**ID’**
3. The Primary Key in countrylanguage table is ‘**CountryCode&Language’**
4. The Foreign Key in the City Table is ‘**CountryCode’**
5. The Foreign Key in the Countrylanguage Table is ‘ **CountryCode’**