SQL Assignment Part 1

using the 'create database' script

Tamima Tarofdar

Data Technician Skills Bootcamp - Just IT (April 2024)

Query 1

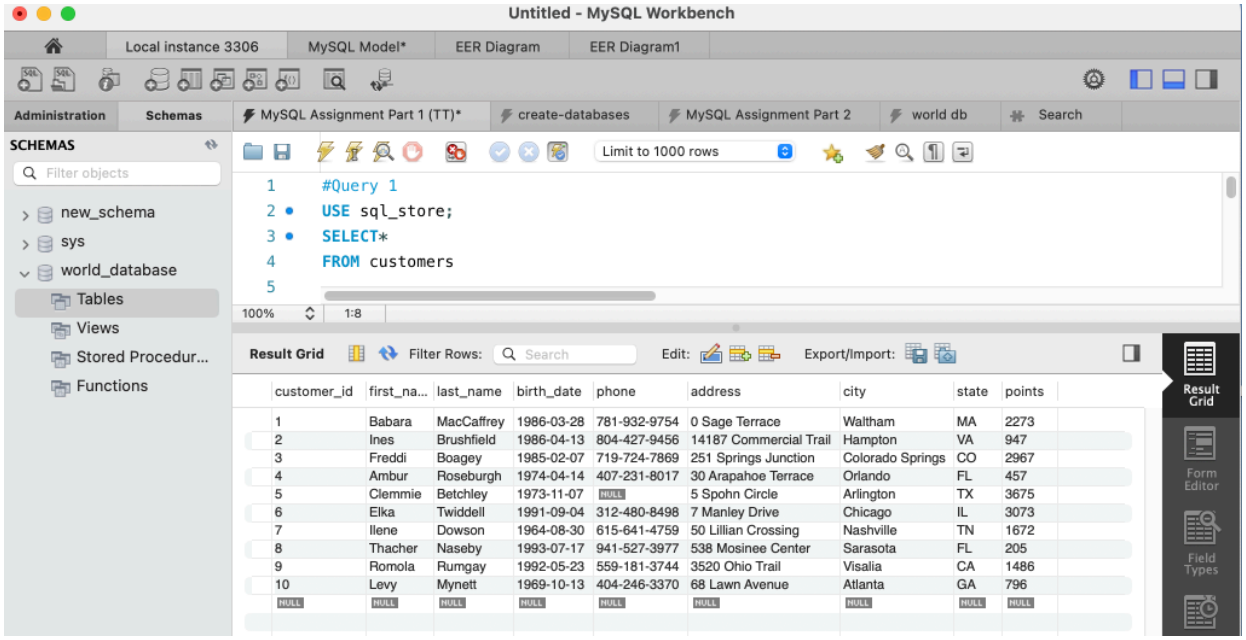
- - - - X

Before anything else, I downloaded and imported the ‘create-databases’ SQL script into MySQL Workbench. I then ran the script to create the databases.

Part a)

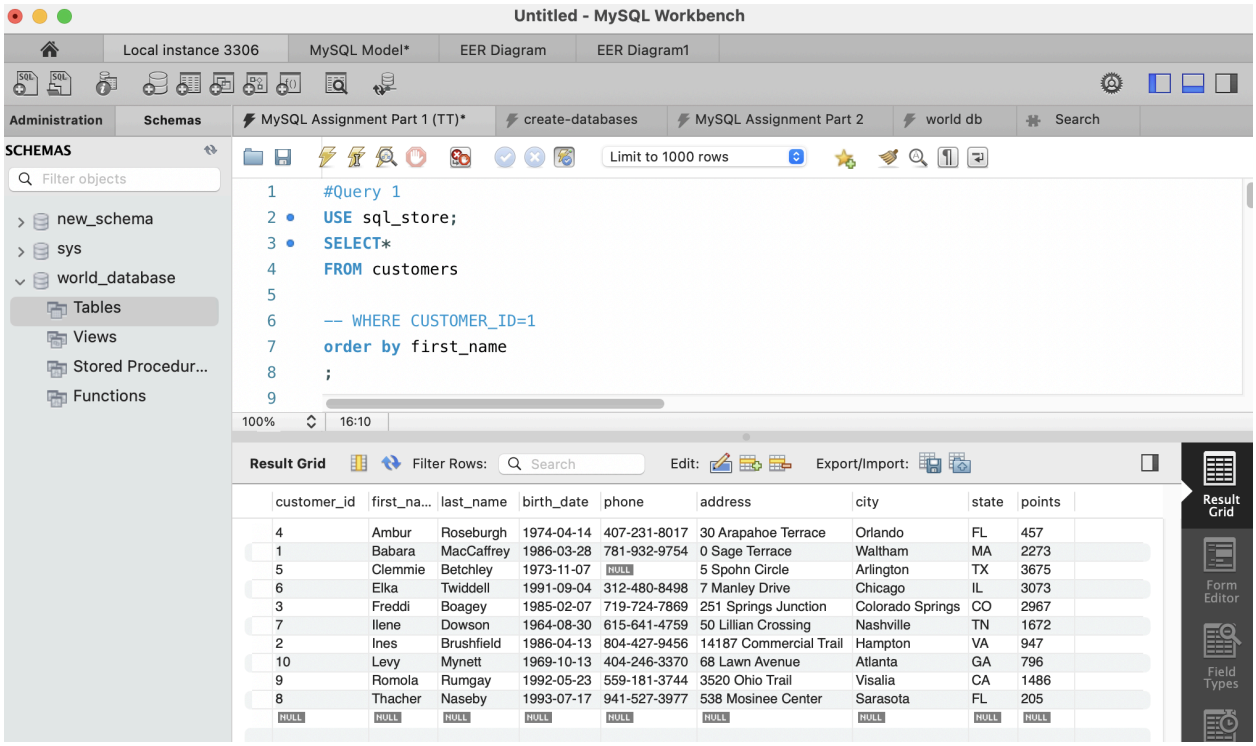
In query 1 the USE sql_store; line specifies that I would like to use the ‘sql_store’ database. This means that all queries from now on will be executed within this database.

The following select clause: SELECT * FROM customers; instructs the database to select all columns from the customers table. As shown below, the result of the query contains all rows and all columns from the customers table.



Part b)

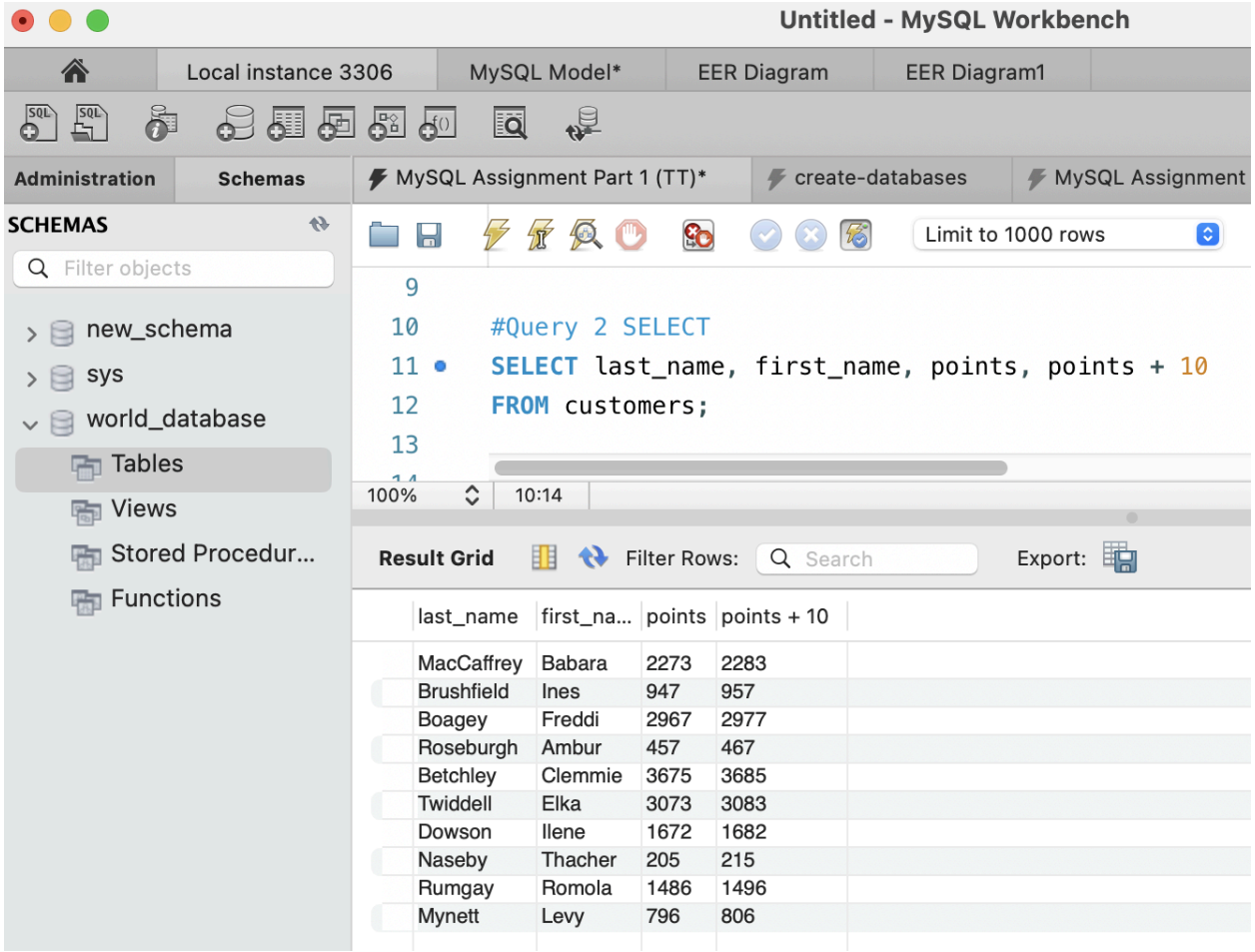
In the next section of Query 1, an order by clause was added so the ‘first_name’ column is sorted in ascending order alphabetically.



Query 2

- - - - X

Using the select clause, the 'last_name', 'first_name', original 'points' and a new points column called 'points + 10' from the customers table should be shown in the results grid.

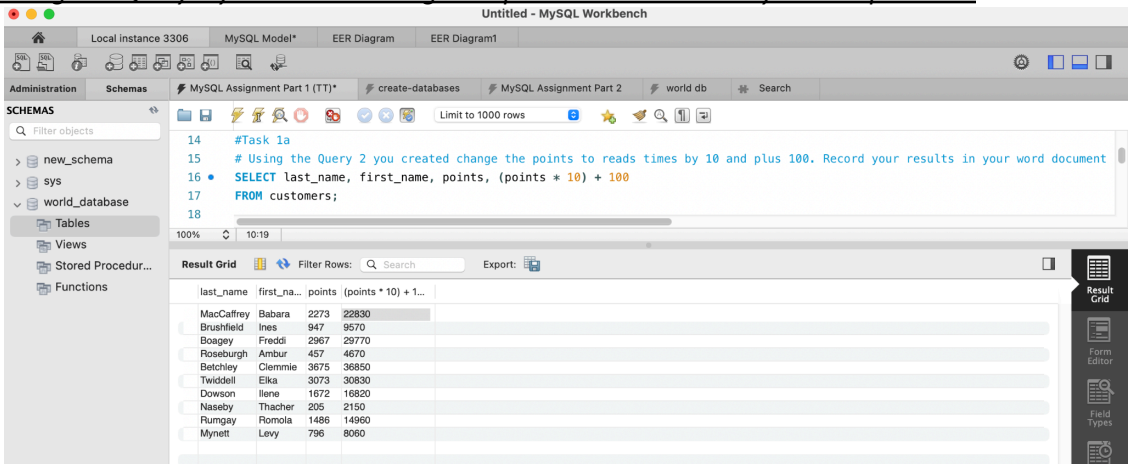


Task 1

- - - - X

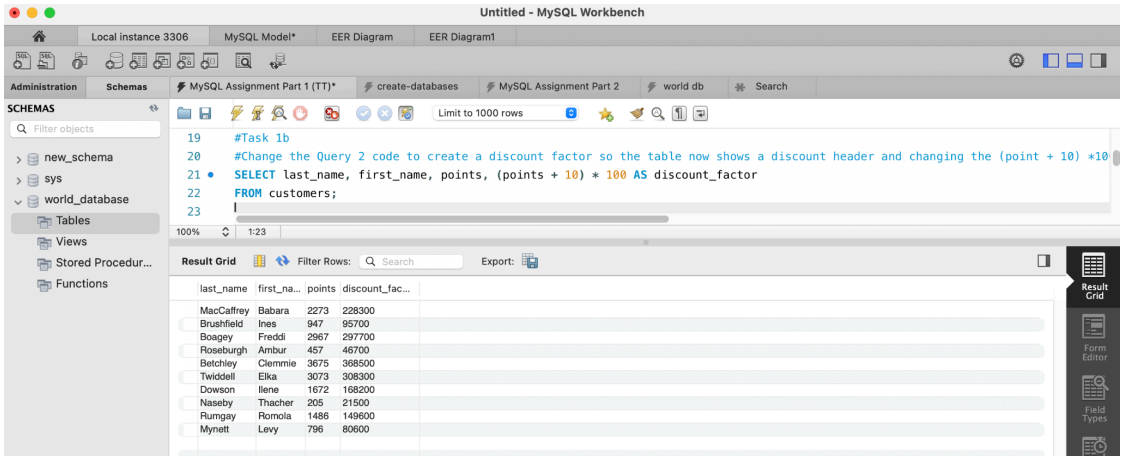
Part a)

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



Part b)

*Change the Query 2 code to create a discount factor so the table now shows a discount header and change the (point + 10) *100*



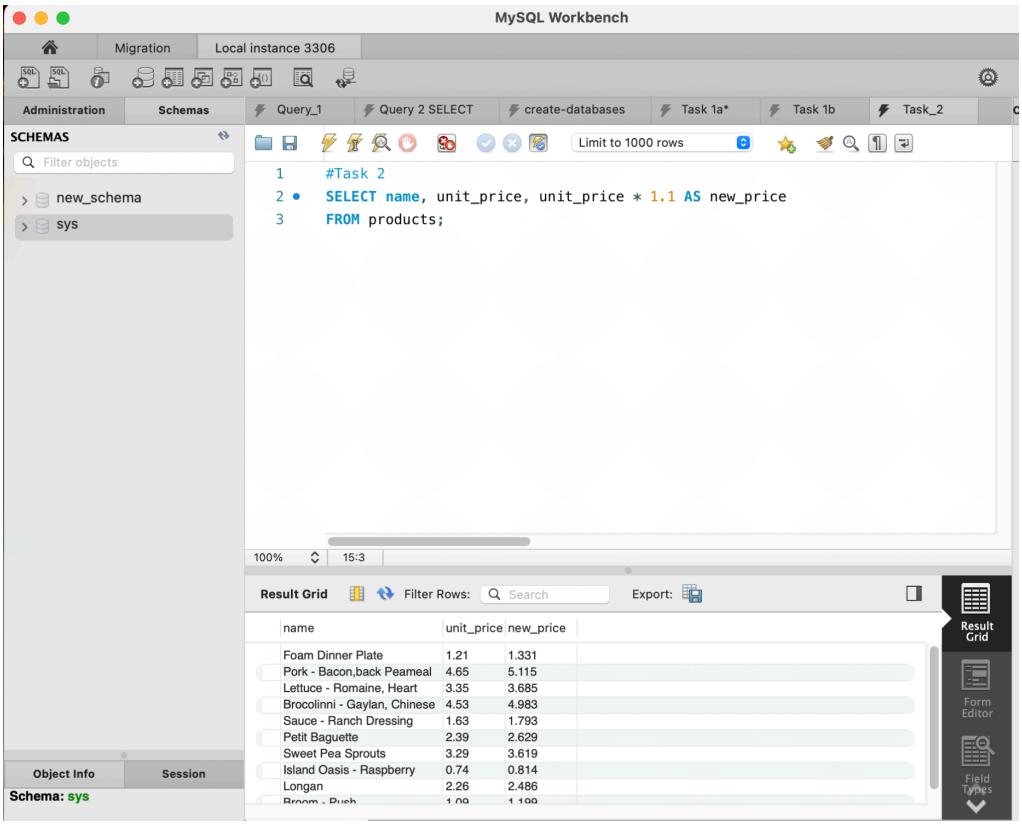
Task 2

- - - - X

*Write a SQL query to return all the products in our database in the result set. I want you to show columns; name, unit price, and a new column called new price which is based on this expression, (unit price * 1.1).*

So what you are doing is increasing the product price of each by 10%.

So with the query, we want all the products with the original price and the new price listed.



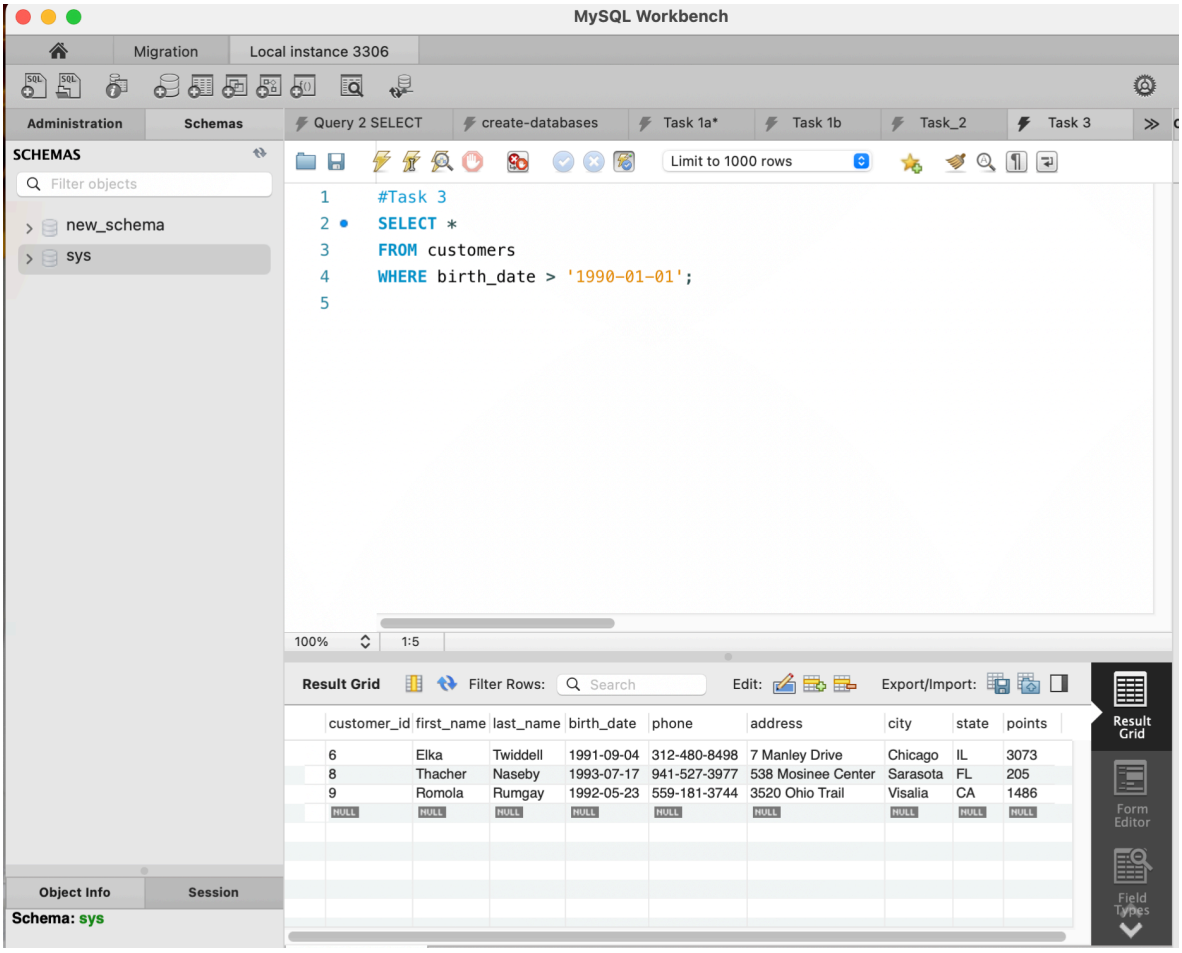
Task 3

- - - - X

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

For this task, I used the asterisk (*) symbol with the 'SELECT' statement to select all the columns from the 'Customers' table.

I then used the 'WHERE birth_date > 1990-01-01' operation to apply a condition so the results are filtered to only include rows where customers with a birth date after 1990-01-01 are shown in the results.



Task 4

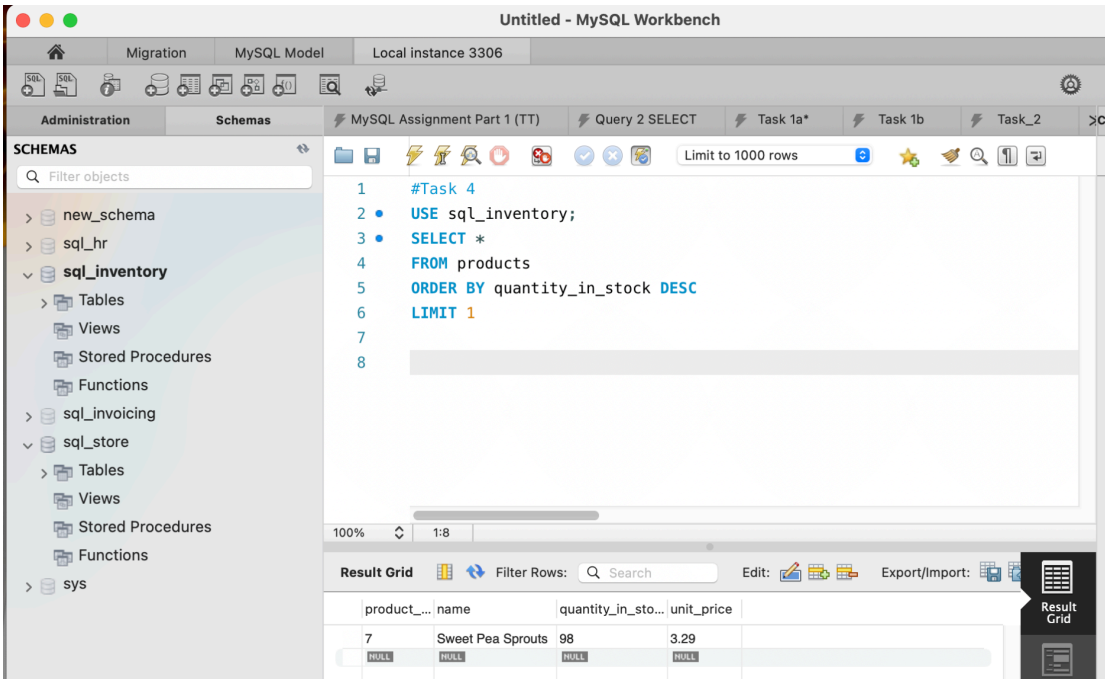
- - - - X

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

Within this code, the 'USE sql_inventory' line specifies that I would like to use the 'sql_inventory' database. This means that all queries from now on will be executed within this database. I then used the asterisk (*) symbol with the 'SELECT' statement to select all the columns from the 'products' table.

I used the 'ORDER BY' statement to organise results based on the 'quantity_in_stock' column. The 'DESC' statement was added for the column to be sorted in descending order so the product with the highest amount in stock will be listed first in the results.

I then used the 'LIMIT 1' function for the query to select only the first row, which now represents the product with most amount in stock.



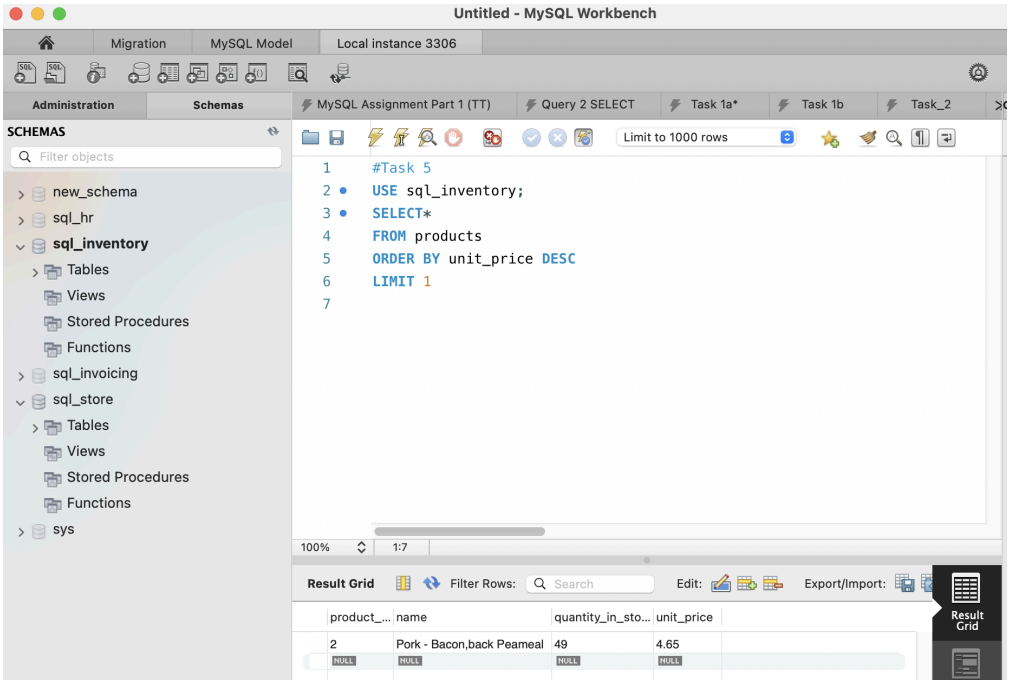
Task 5

- *Select sql_inventory.*
- *Write a query to find out the name of the most expensive product*.

Again, the 'USE sql_inventory' line was used to specify that I would like to use the 'sql_inventory' database. I then used the asterisk (*) symbol with the 'SELECT' statement to select all the columns from the 'products' table.

I used the same code from task 4 but instead used the 'ORDER BY' statement to organise results based on the 'unit_price' column. The 'DESC' statement enables the column to be sorted in descending order so the product with the highest price will be listed first in the results.

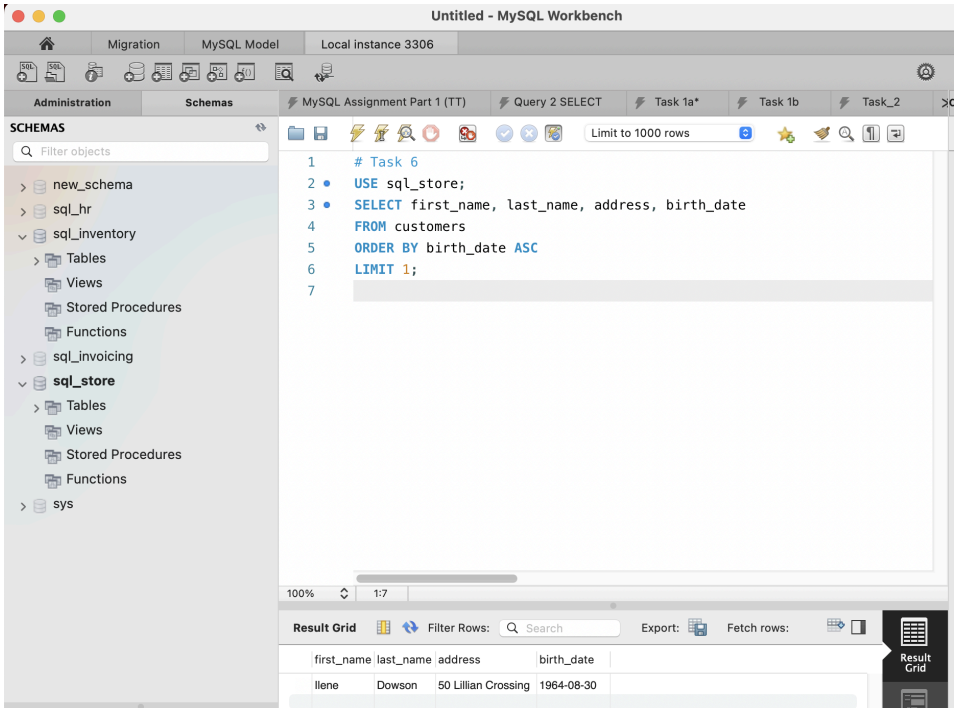
I then used the 'LIMIT 1' function for the query to select only the first row, which now represents the product with most expensive product.



Task 6

- *Select sql_store.*
- *Write a query to find out the first name, last name, address and birthdate of the oldest customer.*

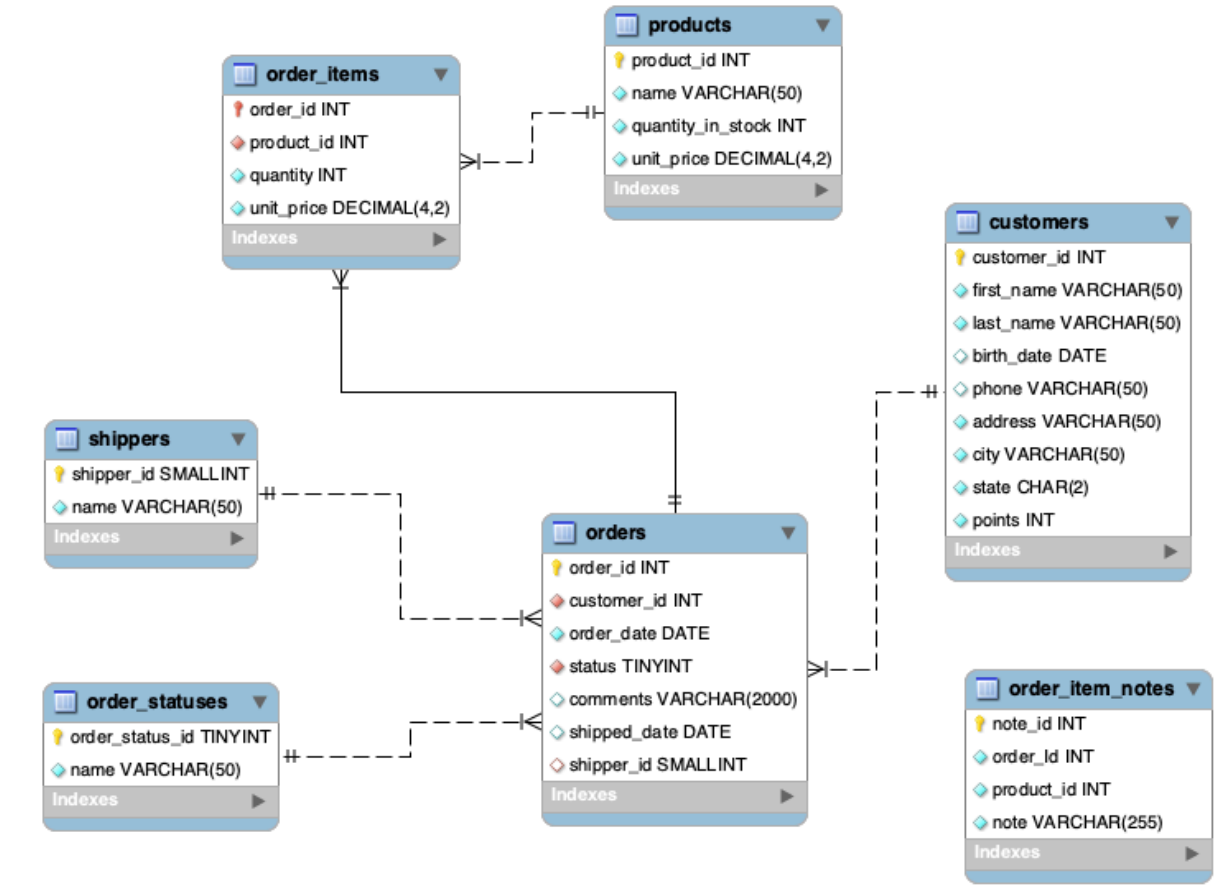
Within this code, the 'USE sql_store' line specifies that I would like to use the 'sql_store' database. I used the 'SELECT' statement to select the 'first_name', 'last_name', 'address' and 'birth_date' columns and the 'FROM' statement to specify this data is to be retrieved from the 'customers' table.



I used the 'ORDER BY' statement to organise the results based on the 'birth_date' column. The 'ASC' statement was added for the column to be sorted in ascending order so the oldest birth date will be listed first in the results.

I then used the 'LIMIT 1' function for the query to select only the first row, which now represents the oldest customer.

EER Diagram


products
product_id INT
name VARCHAR(50)
quantity_in_stock INT
unit_price DECIMAL(4,2)
Indexes
order_items
order_id INT
product_id INT
quantity INT
unit_price DECIMAL(4,2)
Indexes
customers
customer_id INT
first_name VARCHAR(50)
last_name VARCHAR(50)
birth_date DATE
phone VARCHAR(50)
address VARCHAR(50)
city VARCHAR(50)
state CHAR(2)
points INT
Indexes
shippers
shipper_id SMALLINT
name VARCHAR(50)
Indexes
orders
order_id INT
customer_id INT
order_date DATE
status TINYINT
comments VARCHAR(2000)
shipped_date DATE
shipper_id SMALLINT
Indexes
order_statuses
order_status_id TINYINT
name VARCHAR(50)
Indexes
order_item_notes
note_id INT
order_Id INT
product_id INT
note VARCHAR(255)
Indexes

An Enhanced Entity-Relationship (EER) diagram provides a visual representation of the relationships among the tables in a database schema.

The EER diagram shown above was curated from the ‘sql_store’ schema in which most of the codes in this assignment was executed in.

It shows tables as boxes, with connecting lines representing the relationships between them. It's a great way to design a new relational database or understand and modify an existing one.

These diagrams are useful when designing databases as the provide a thorough snapshot of it’s tables, relationships, properties and constraints.

The key components typically found in an EER diagram include the following:

- Yellow key: represents the primary key of the table
- Red diamond: represents the foreign keys that are laid out to other tables
- Solid blue diamonds: represent non-nullible fields
- Open/unfilled diamonds: represent fields that can contain null values
- Lines and cardinality symbols: signify a relationship and the type of relationship between the tables. Depending on the lines and shape at the end of the line a relationship between tables can be one-to-one, or many-to-one, or one-to-many.
- Data Types of each column are labelled within each table. For example INT, VARCHAR(x), DATE, etc.

For this EER diagram, I've summarised the primary and foreign keys and the relation between each table below:

Table Name	Primary Key	Foreign Keys	Relationship/ cardinality with other tables
order_items	order_id INT	product_id INT	One to many with the product_id column in the products table and order_id column in the orders table
products	product_id INT		One to many with the product_id in the order_items table
customers	customer_id INT		One to many with the customer_id column in orders table
orders	order_id INT	customer_id INT status TINYINT	One to many with the order_id column in the order_item table, the customer_id column in the customers table, the order_status_id column the order_statuses table and the shippers_id in the shippers table
shippers	shipper_id SMALL INT		One to many with the shipper_id column in the orders table
order_statuses	order_status_id		One to many with the status column in the orders table
order_item_notes	note_id INT		