



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.

The screenshot shows the MySQL Workbench interface with the title bar "Untitled - MySQL Workbench". The left sidebar shows the schema "world_database" selected, with "Tables" currently active. The main pane displays a query editor with the following content:

```
1 #Query 1
2 • USE sql_store;
3 • SELECT*
4   FROM customers
5
```

Below the query editor is a "Result Grid" showing the data from the "customers" table. The columns are: customer_id, first_name, last_name, birth_date, phone, address, city, state, and points. The data consists of 10 rows of customer information. A vertical scroll bar is visible on the right side of the result grid.

| customer_id | first_name | last_name | birth_date | phone | address | city | state | points |
|-------------|------------|------------|------------|--------------|------------------------|------------------|-------|--------|
| 1 | Barbara | MacCaffrey | 1986-03-28 | 781-932-9754 | 0 Sage Terrace | Waltham | MA | 2273 |
| 2 | Ines | Brushfield | 1986-04-13 | 804-427-9456 | 14187 Commercial Trail | Hampton | VA | 947 |
| 3 | Freddi | Boagey | 1985-02-07 | 719-724-7869 | 251 Springs Junction | Colorado Springs | CO | 2967 |
| 4 | Ambur | Roseburgh | 1974-04-14 | 407-231-8017 | 30 Arapahoe Terrace | Orlando | FL | 457 |
| 5 | Clemmie | Betchley | 1973-11-07 | NULL | 5 Spohn Circle | Arlington | TX | 3675 |
| 6 | Elka | Twiddell | 1991-09-04 | 312-480-8498 | 7 Manley Drive | Chicago | IL | 3073 |
| 7 | Ilene | Dowson | 1964-08-30 | 615-641-4759 | 50 Lillian Crossing | Nashville | TN | 1672 |
| 8 | Thacher | Naseby | 1993-07-17 | 941-527-3977 | 538 Mosinee Center | Sarasota | FL | 205 |
| 9 | Romola | Rumgay | 1992-05-23 | 559-181-3744 | 3520 Ohio Trail | Visalia | CA | 1486 |
| 10 | Levy | Mynett | 1969-10-13 | 404-246-3370 | 68 Lawn Avenue | Atlanta | GA | 796 |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |



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.

Untitled - MySQL Workbench

Administration Schemas MySQL Assignment Part 1 (TT)* create-databases MySQL Assignment Part 2 world db Search

SCHEMAS

new_schema sys world_database

Tables Views Stored Procedur... Functions

#Query 1

```
1 USE sql_store;
2 • SELECT*
3 •   FROM customers
4
5 -- WHERE CUSTOMER_ID=1
6 order by first_name
7 ;
8
9
```

Result Grid Filter Rows: Search Edit: Export/Import: Result Grid Form Editor Field Types

| customer_id | first_na... | last_name | birth_date | phone | address | city | state | points |
|-------------|-------------|------------|------------|--------------|------------------------|------------------|-------|--------|
| 4 | Ambur | Roseburgh | 1974-04-14 | 407-231-8017 | 30 Arapahoe Terrace | Orlando | FL | 457 |
| 1 | Barbara | MacCaffrey | 1986-03-28 | 781-932-9754 | 0 Sage Terrace | Waltham | MA | 2273 |
| 5 | Clemmie | Betchley | 1973-11-07 | NULL | 5 Spohn Circle | Arlington | TX | 3675 |
| 6 | Elka | Twidwell | 1991-09-04 | 312-480-8498 | 7 Manley Drive | Chicago | IL | 3073 |
| 3 | Freddi | Boagey | 1985-02-07 | 719-724-7869 | 251 Springs Junction | Colorado Springs | CO | 2967 |
| 7 | Ilene | Dowson | 1964-08-30 | 615-641-4759 | 50 Lillian Crossing | Nashville | TN | 1672 |
| 2 | Ines | Brushfield | 1986-04-13 | 804-427-9456 | 14187 Commercial Trail | Hampton | VA | 947 |
| 10 | Levy | Mynett | 1969-10-13 | 404-246-3370 | 68 Lawn Avenue | Atlanta | GA | 796 |
| 9 | Romola | Rumgay | 1992-05-23 | 559-181-3744 | 3520 Ohio Trail | Visalia | CA | 1486 |
| 8 | Thacher | Naseby | 1993-07-17 | 941-527-3977 | 538 Mosinee Center | Sarasota | FL | 205 |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |



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.

Screenshot of MySQL Workbench showing the execution of Query 2 and its results.

The SQL editor contains the following query:

```
9
10  #Query 2 SELECT
11 •  SELECT last_name, first_name, points, points + 10
12  FROM customers;
13
14
```

The Result Grid shows the following data:

| last_name | first_na... | points | points + 10 |
|------------|-------------|--------|-------------|
| MacCaffrey | Babara | 2273 | 2283 |
| Brushfield | Ines | 947 | 957 |
| Boagey | Freddi | 2967 | 2977 |
| Roseburgh | Ambur | 457 | 467 |
| Betchley | Clemmie | 3675 | 3685 |
| Twiddell | Elka | 3073 | 3083 |
| Dowson | Ilene | 1672 | 1682 |
| Naseby | Thacher | 205 | 215 |
| Rumgay | Romola | 1486 | 1496 |
| Mynett | Levy | 796 | 806 |



Task 1

- - - - X

Part a)

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

The screenshot shows the MySQL Workbench interface with the title bar "Untitled - MySQL Workbench". The main area displays a SQL editor with the following code:

```
14 #Task 1a
15 # Using the Query 2 you created change the points to reads times by 10 and plus 100. Record your results in your word document
16 • SELECT last_name, first_name, points, (points * 10) + 100
17 FROM customers;
18
```

The results grid shows the following data:

| last_name | first_na... | points | (points * 10) + 1... |
|------------|-------------|--------|----------------------|
| MacCaffrey | Barbara | 2273 | 22830 |
| Brushfield | Ines | 947 | 9570 |
| Boagey | Freddi | 2967 | 29770 |
| Roseburgh | Ambur | 457 | 4670 |
| Betchley | Clemmie | 3675 | 36850 |
| Twiddell | Elka | 3073 | 30830 |
| Dowson | Ilene | 1672 | 16820 |
| Naseby | Thacher | 205 | 2150 |
| Rungay | Romola | 1486 | 14960 |
| Mynett | Levy | 796 | 8060 |

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

The screenshot shows the MySQL Workbench interface with the title bar "Untitled - MySQL Workbench". The main area displays a SQL editor with the following code:

```
19 #Task 1b
20 #Change the Query 2 code to create a discount factor so the table now shows a discount header and changing the (point + 10) *100 AS discount_factor
21 • SELECT last_name, first_name, points, (points + 10) * 100 AS discount_factor
22 FROM customers;
23
```

The results grid shows the following data:

| last_name | first_na... | points | discount_fac... |
|------------|-------------|--------|-----------------|
| MacCaffrey | Barbara | 2273 | 228300 |
| Brushfield | Ines | 947 | 95700 |
| Boagey | Freddi | 2967 | 297700 |
| Roseburgh | Ambur | 457 | 46700 |
| Betchley | Clemmie | 3675 | 368500 |
| Twiddell | Elka | 3073 | 308300 |
| Dowson | Ilene | 1672 | 168200 |
| Naseby | Thacher | 205 | 21500 |
| Rungay | Romola | 1486 | 149600 |
| Mynett | Levy | 796 | 80600 |



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.

The screenshot shows the MySQL Workbench interface. The top navigation bar includes 'Migration' and 'Local instance 3306'. The main window has tabs for 'Administration' and 'Schemas'. Under 'Schemas', there are entries for 'new_schema' and 'sys', with 'sys' currently selected. The central workspace contains two queries:

```
1 #Task 2
2 • SELECT name, unit_price, unit_price * 1.1 AS new_price
3 FROM products;
```

Below the queries is a 'Result Grid' showing the output of the second query:

| name | unit_price | new_price |
|------------------------------|------------|-----------|
| Foam Dinner Plate | 1.21 | 1.331 |
| Pork - Bacon,back Peameal | 4.65 | 5.115 |
| Lettuce - Romaine, Heart | 3.35 | 3.685 |
| Broccolini - Gaylan, Chinese | 4.53 | 4.983 |
| Sauce - Ranch Dressing | 1.63 | 1.793 |
| Petit Baguette | 2.39 | 2.629 |
| Sweet Pea Sprouts | 3.29 | 3.619 |
| Island Oasis - Raspberry | 0.74 | 0.814 |
| Longan | 2.26 | 2.486 |
| Broom - Brush | 1.00 | 1.00 |

The bottom left shows 'Object Info' and 'Session' tabs, and the bottom center indicates 'Schema: sys'. On the right side, there are icons for 'Result Grid', 'Form Editor', and 'Field Types'.



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.

The screenshot shows the MySQL Workbench interface. In the top navigation bar, 'Migration' and 'Local instance 3306' are selected. The main area has tabs for 'Administration' and 'Schemas'. Under 'Schemas', there are two entries: 'new_schema' and 'sys', with 'sys' currently selected. Below this is a 'Query Editor' tab labeled 'Query 2 SELECT' containing the following SQL code:

```
1 #Task 3
2 • SELECT *
3   FROM customers
4 WHERE birth_date > '1990-01-01';
5
```

Below the code is a 'Result Grid' showing the results of the query. The columns are: customer_id, first_name, last_name, birth_date, phone, address, city, state, and points. The data includes:

| | customer_id | first_name | last_name | birth_date | phone | address | city | state | points |
|---|-------------|------------|-----------|------------|--------------|--------------------|----------|-------|--------|
| 6 | | Elka | Twiddell | 1991-09-04 | 312-480-8498 | 7 Manley Drive | Chicago | IL | 3073 |
| 8 | | Thacher | Naseby | 1993-07-17 | 941-527-3977 | 538 Mosinee Center | Sarasota | FL | 205 |
| 9 | | Romola | Rumgay | 1992-05-23 | 559-181-3744 | 3520 Ohio Trail | Visalia | CA | 1486 |
| | | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |

At the bottom left, 'Object Info' and 'Session' tabs are visible, along with a 'Schema: sys' indicator. On the right side, there are icons for 'Result Grid', 'Form Editor', and 'Field Types'.



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.

The screenshot shows the MySQL Workbench interface with the title bar 'Untitled - MySQL Workbench'. The left sidebar displays the 'SCHEMAS' tree, showing databases like 'new_schema', 'sql_hr', 'sql_inventory', 'sql_invoicing', 'sql_store', and 'sys'. The 'sql_inventory' database is expanded, showing 'Tables', 'Views', 'Stored Procedures', and 'Functions'. The main area contains the SQL editor with the following code:

```
1 #Task 4
2 • USE sql_inventory;
3 • SELECT *
4   FROM products
5   ORDER BY quantity_in_stock DESC
6   LIMIT 1
7
8
```

Below the editor is the 'Result Grid' pane, which displays the query results:

| product_id | name | quantity_in_stock | unit_price |
|------------|-------------------|-------------------|------------|
| 7 | Sweet Pea Sprouts | 98 | 3.29 |
| HULL | HULL | HULL | HULL |



Task 5

- - - - X

- 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.

The screenshot shows the MySQL Workbench interface. The SQL editor tab contains the following query:

```
1 #Task 5
2 • USE sql_inventory;
3 • SELECT*
4 FROM products
5 ORDER BY unit_price DESC
6 LIMIT 1
7
```

The Result Grid shows the output of the query:

| product_id | name | quantity_in_stock | unit_price |
|------------|---------------------------|-------------------|------------|
| 2 | Pork - Bacon,back Peameal | 49 | 4.65 |



Task 6

- - - - X

- 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.

The screenshot shows the MySQL Workbench interface with the title bar 'Untitled - MySQL Workbench'. The left sidebar displays the 'SCHEMAS' tree, showing databases like 'new_schema', 'sql_hr', 'sql_inventory', 'sql_invoicing', 'sql_store', and 'sys'. The main area has two tabs: 'MySQL Assignment Part 1 (TT)' and 'Query 2 SELECT'. The 'Query 2 SELECT' tab contains the following SQL code:

```
# Task 6
USE sql_store;
SELECT first_name, last_name, address, birth_date
FROM customers
ORDER BY birth_date ASC
LIMIT 1;
```

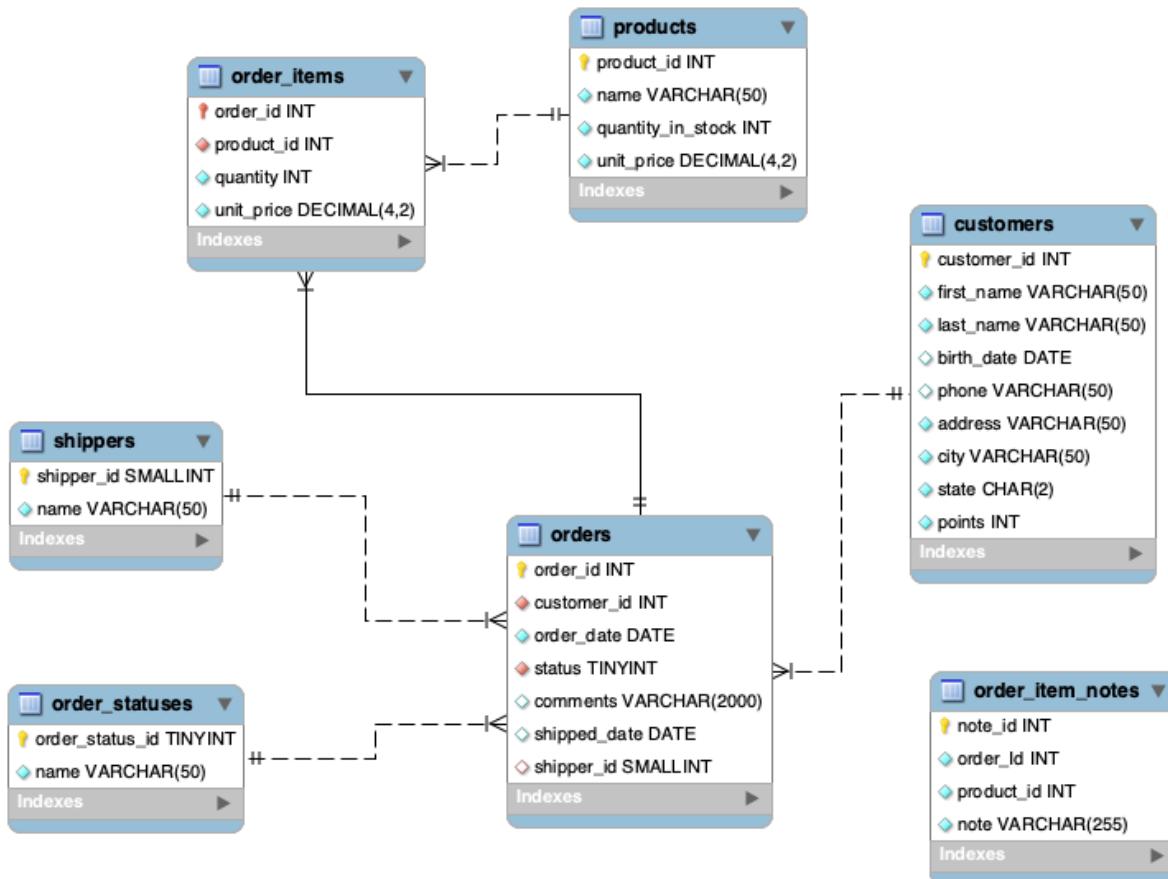
Below the code, the 'Result Grid' shows the output:

| first_name | last_name | address | birth_date |
|------------|-----------|---------------------|------------|
| Ilene | Dowson | 50 Lillian Crossing | 1964-08-30 |



EER Diagram

- - - - X





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 they provide a thorough snapshot of its 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 in 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 | | |