Querying database

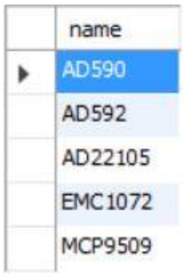
1) Get the name of all the available items

Query:

```
SELECT name

FROM fis_assignment2.items;
```

Result:



name
AD590
AD592
AD22105
EMC1072
MCP9509

2) Get the name, phone number and city of the customer with Customer ID number 2

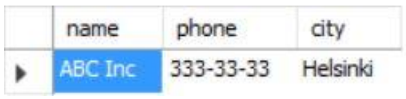
Query:

```
SELECT name, phone, city

FROM fis_assignment2.customers

WHERE customers.id = 2;
```

Result:



name	phone	city
ABC Inc	333-33-33	Helsinki

3) Retrieve the item IDs and item names purchased in the Order ID number 4

Query:

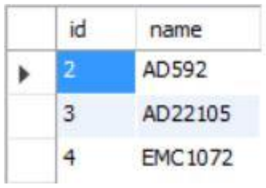
```
SELECT id, name

FROM items INNER JOIN order_details

ON items.id = order_details.item_id

WHERE order_details.order_id = 4;
```

Result:



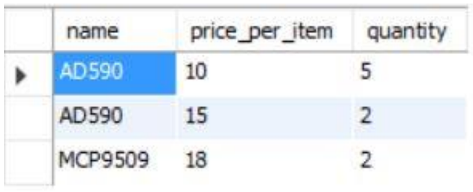
id	name
2	AD592
3	AD22105
4	EMC1072

4) Retrieve the names of the items, price per item, and quantity per item purchased in the Order ID number 3

Query:

```
SELECT name, price_per_item, quantity
FROM items INNER JOIN order_details
ON items.id = order_details.item_id
WHERE order_details.order_id = 3;
```

Result:



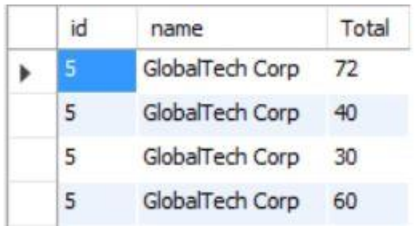
name	price_per_item	quantity
AD590	10	5
AD590	15	2
MCP9509	18	2

5) Retrieve order ID, customer name and total sum of order for order ID = 5

Query:

```
SELECT orders.id, customers.name, (price_per_item * quantity) AS Total
FROM orders, order_details, customers
WHERE orders.customer_id = customers.id and orders.id = order_details.order_id and order_id = 5;
```

Result:



id	name	Total
5	GlobalTech Corp	72
5	GlobalTech Corp	40
5	GlobalTech Corp	30
5	GlobalTech Corp	60

Explanation:

This query is used to get the total sum of single order by a customer using its order id. The "(price_per_item * quantity) AS Total" part of the query is used to get the product of the values of 'price_per_item' and 'quantity' in an attribute of 'total'. The FROM command defines the tables from which we are to get the values and the WHERE clause defines the conditions. Which in this case is used to specify the order_id which is to be queried against and the relations of equality between the tables.

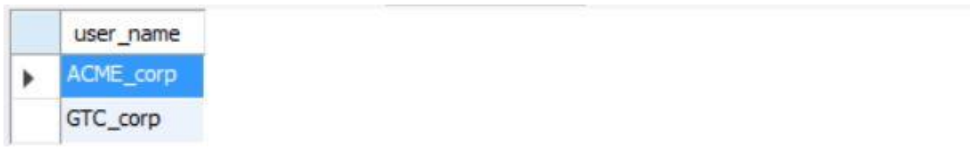
6) Retrieve user names of customer accounts for customers, who have orders with item names "MCP9509" or "AD590"

Query:

```
SELECT DISTINCT customer_accounts.user_name
```

```
FROM orders INNER JOIN customer_accounts
ON orders.customer_id = customer_accounts.customer_id
INNER JOIN order_details
ON orders.id = order_details.order_id
INNER JOIN items
ON order_details.item_id = items.id
WHERE items.name = 'MCP9509' or items.name = 'AD590';
```

Result:



user_name
ACME_corp
GTC_corp

Explanation:

DISTINCT command is used to eliminate repetition in this case. INNER JOIN command is used to create subsets of related value between two different tables. Two item name are provided in the requirement against which we have to fetch the Customer username. The item name exist in the items table and the user_name exist in the customer_accounts table. The tables from customer_accounts to items are hierarchal which is accessed by using INNER JOIN command by referencing to their relationship through primary keys. In the end fetching the required result.

Problems Encountered

The provided example for the task had a different way of defining the menuHandler function which is function description method. I had some problem implementing it in that manner with my cases so I have changed its definition. The queries were simple enough in both the first and second tasks. I also had some issues regarding the process.stdin.on() command as my menuHandler function is defined differently so I have changed its type from ‘readable’ to ‘data’ so it takes input value only in that instance of the module changing the command to process.stdin.once().