

SQL TASK

Database Creation :

Database created under the name eCommerce using the command “Create Database eCommerce ;” and the created database is used using the command “ Use eCommerce ;”.

Table creation :

Three tables are created with the required columns and sample data were inserted into them.

Customer Table :

The table ‘Customers’ is created with fields name , email, address and Id where ID is the primary key which is set to be auto incremented . Below is the Screenshots of Table Creation and Table Contents.

```
[mysql> create table Customers( Id INT Auto_INCREMENT Primary key ,
-> Name Varchar(50),
-> Email Varchar(100),
-> Address Varchar(100));
Query OK, 0 rows affected (0.03 sec)
```

```
[mysql> Select * from customers;
+----+-----+-----+-----+
| Id | Name   | Email                | Address                |
+----+-----+-----+-----+
| 1  | Arun   | arun@example.com     | Chennai, Tamil Nadu   |
| 2  | Meena  | meena@example.com    | Coimbatore, Tamil Nadu|
| 3  | Naveen | naveen@example.com   | Bangalore, Karnataka  |
| 4  | Keerthi| keerthi@example.com  | Hyderabad, Telangana  |
| 5  | Vijay  | vijay@example.com    | Kochi, Kerala         |
| 6  | Lakshmi| lakshmi@example.com  | Madurai, Tamil Nadu   |
| 7  | Sathya | sathya@example.com   | Trivandrum, Kerala    |
| 8  | Divya  | divya@example.com    | Mysore, Karnataka     |
| 9  | Karthik| karthik@example.com  | Vizag, Andhra Pradesh  |
| 10 | Revathi| revathi@example.com  | Tirupati, Andhra Pradesh|
+----+-----+-----+-----+
10 rows in set (0.00 sec)
```

Table (a) - Customers

Products Table :

The table ‘products’ is created with fields name, price, description and Id, where ID is the primary key which is set to be auto-incremented. Below is the Screenshots of Table Creation and Table Contents.

```
mysql> CREATE TABLE products (
-> id INT AUTO_INCREMENT PRIMARY KEY,
-> name VARCHAR(100),
-> price DECIMAL(10,2),
-> description TEXT
-> );
Query OK, 0 rows affected (0.01 sec)
```

```
[mysql> select * from products;
```

id	name	price	description
1	book	150.00	Notebook with ruled pages
2	pen	25.00	Blue ink ballpoint pen
3	pencil	10.00	HB graphite pencil
4	table	2500.00	Wooden study table
5	chair	1200.00	Plastic office chair
6	mobile	15000.00	Android smartphone
7	laptop	45000.00	15-inch Intel Core laptop
8	mouse	599.00	Wireless optical mouse
9	keyboard	999.00	USB mechanical keyboard
10	charger	499.00	Mobile fast charger

```
10 rows in set (0.00 sec)
```

Table(b) - Products

Orders table:

The table 'orders' is created with fields Order_date, Total_Amount , Id, where ID is the primary key which is set to be auto-incremented. The Id from customer table is set as Foreign key here under the name Customer_Id. Below is the Screenshots of Table Creation and Table Contents.

```
[mysql> Create Table Orders(
[   -> ID INT Auto_Increment Primary Key,
[   -> Customer_id Int,
[   -> Order_Date Date,
[   -> Total_amount decimal(10,2),
[   -> Foreign Key (Customer_id) REFERENCE Customers(Id));
```

```
[mysql> select * from orders
[   -> ;
```

ID	Customer_id	Order_Date	Total_amount
1	1	2025-05-28	999.99
2	2	2025-06-05	450.00
3	3	2025-06-18	1200.00
4	4	2025-06-30	199.00
5	5	2025-07-05	1499.00
6	6	2025-07-10	250.00
7	7	2025-07-13	349.75
8	8	2025-07-15	599.00
9	9	2025-07-17	699.00
10	10	2025-07-18	1300.00
11	5	2025-05-04	899.99
12	8	2025-02-04	209.99
13	5	2025-03-05	309.00
14	3	2025-03-06	234.00
15	8	2025-03-09	256.00
16	2	2025-07-09	905.00
17	1	2025-07-09	905.00
18	1	2025-07-09	904.00

```
18 rows in set (0.01 sec)
```

Table(c)-Orders

QUERIES:

1. Retrieve all customers who have placed an order in the last 30 days :

To retrieve the all the customers who have placed their order in the last 30days, the command can be given simple as “select c.* from customers c join orders o on c.id=o.customer_id where o.order_date >= '2025-06-18' “;

```
mysql> select c.* from customers c join orders o on c.id=o.customer_id where o.order_date >= '2025-06-18' ;
```

Id	Name	Email	Address
3	Naveen	naveen@example.com	Bangalore, Karnataka
4	Keerthi	keerthi@example.com	Hyderabad, Telangana
5	Vijay	vijay@example.com	Kochi, Kerala
6	Lakshmi	lakshmi@example.com	Madurai, Tamil Nadu
7	Sathya	sathya@example.com	Trivandrum, Kerala
8	Divya	divya@example.com	Mysore, Karnataka
9	Karthik	karthik@example.com	Vizag, Andhra Pradesh
10	Revathi	revathi@example.com	Tirupati, Andhra Pradesh

```
8 rows in set (0.00 sec)
```

Even though this works just fine, Instead of giving commands explicitly mentioning dates, we can write queries using DATEDIFF(gives the difference between two dates) or CURDATE(today's date) by mentioning interval as 30 which will flexible to work on.

```
mysql> SELECT DISTINCT c.*
-> FROM customers c
-> JOIN orders o ON c.id = o.customer_id
-> WHERE DATEDIFF(CURDATE(), o.order_date) <= 30;
```

Id	Name	Email	Address
3	Naveen	naveen@example.com	Bangalore, Karnataka
4	Keerthi	keerthi@example.com	Hyderabad, Telangana
5	Vijay	vijay@example.com	Kochi, Kerala
6	Lakshmi	lakshmi@example.com	Madurai, Tamil Nadu
7	Sathya	sathya@example.com	Trivandrum, Kerala
8	Divya	divya@example.com	Mysore, Karnataka
9	Karthik	karthik@example.com	Vizag, Andhra Pradesh
10	Revathi	revathi@example.com	Tirupati, Andhra Pradesh

```
8 rows in set (0.01 sec)
```


2. Get the total amount of all orders placed by each customer:

To get the sum of all orders placed by each customers , the sum() function along with GROUP BY can be used.

```
[mysql> SELECT c.name as Customer_name, SUM(o.total_amount)
-> FROM customers c
-> JOIN orders o on c.id = o.customer_id GROUP BY c.name;
+-----+-----+
| Customer_name | SUM(o.total_amount) |
+-----+-----+
| Arun          | 2808.99             |
| Meena         | 1355.00             |
| Naveen        | 1434.00             |
| Keerthi       | 199.00              |
| Vijay         | 2707.99             |
| Lakshmi       | 250.00              |
| Sathya        | 349.75              |
| Divya         | 1064.99             |
| Karthik       | 699.00              |
| Revathi       | 1300.00             |
+-----+-----+
10 rows in set (0.00 sec)
```

Here the order is grouped by customer name so each row shows one customer and their total spending.

3. Update the price of Product C to 45.00 :

The name of the products from the product table are given as book , pencil, pen etc,. So let the pencil from the third row of products can be assumed as product C. The UPDATE command is used to change the price of the product.

```
[mysql> select * from products;
+----+-----+-----+-----+
| id | name   | price | description          |
+----+-----+-----+-----+
| 1  | book   | 150.00 | Notebook with ruled pages |
| 2  | pen    | 25.00  | Blue ink ballpoint pen   |
| 3  | pencil | 10.00  | HB graphite pencil       |
| 4  | table  | 2500.00 | Wooden study table        |
| 5  | chair  | 1200.00 | Plastic office chair      |
+----+-----+-----+-----+
#product C
```

```
[mysql> Update products set price = 45.00 where name = 'pencil' ;
Query OK, 1 row affected (0.02 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

```
[mysql> select * from products;
+----+-----+-----+-----+
| id | name   | price | description          |
+----+-----+-----+-----+
| 1  | book   | 150.00 | Notebook with ruled pages |
| 2  | pen    | 25.00  | Blue ink ballpoint pen   |
| 3  | pencil | 45.00  | HB graphite pencil       |
| 4  | table  | 2500.00 | Wooden study table        |
| 5  | chair  | 1200.00 | Plastic office chair      |
| 6  | mobile | 15000.00 | Android smartphone       |
| 7  | laptop | 45000.00 | 15-inch Intel Core laptop |
| 8  | mouse  | 599.00  | Wireless optical mouse   |
| 9  | keyboard | 999.00 | USB mechanical keyboard   |
| 10 | charger | 499.00 | Mobile fast charger       |
+----+-----+-----+-----+
10 rows in set (0.00 sec)
```

4. Add a new column discount to the products table :

To add new column to the existing table ALTER table command is used.

```
[mysql> alter table products add column Discount decimal(5,2);
Query OK, 0 rows affected (0.03 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

```
[mysql> select * from products;
+-----+-----+-----+-----+-----+
| id | name      | price  | description                | Discount |
+-----+-----+-----+-----+-----+
| 1  | book      | 150.00 | Notebook with ruled pages  | NULL     |
| 2  | pen       | 25.00  | Blue ink ballpoint pen    | NULL     |
| 3  | pencil    | 45.00  | HB graphite pencil        | NULL     |
| 4  | table     | 2500.00 | Wooden study table         | NULL     |
| 5  | chair     | 1200.00 | Plastic office chair       | NULL     |
| 6  | mobile    | 15000.00 | Android smartphone        | NULL     |
| 7  | laptop    | 45000.00 | 15-inch Intel Core laptop | NULL     |
| 8  | mouse     | 599.00  | Wireless optical mouse    | NULL     |
| 9  | keyboard  | 999.00  | USB mechanical keyboard   | NULL     |
| 10 | charger   | 499.00  | Mobile fast charger        | NULL     |
+-----+-----+-----+-----+-----+
10 rows in set (0.00 sec)
```

Now the product table have a new discount column whose value can be set to default value 10% or some other value instead of NULL .

5. Retrieve the top 3 products with the highest price.

To get the top 3 highest price products, the ORDER BY Price DESC is used. To get the top lowest price value , ORDER BY ASC can be used. Here the limit is set to 3 to get the top three values.

```
[mysql> select * from products order by price desc limit 3;
+-----+-----+-----+-----+-----+
| id | name      | price  | description                | Discount |
+-----+-----+-----+-----+-----+
| 7  | laptop    | 45000.00 | 15-inch Intel Core laptop | 10.00    |
| 6  | mobile    | 15000.00 | Android smartphone        | 10.00    |
| 4  | table     | 2500.00  | Wooden study table         | 10.00    |
+-----+-----+-----+-----+-----+
3 rows in set (0.01 sec)
```

6. Join the orders and customers tables to retrieve the customer's name and order date for each order :

The Customer name and order date can be retrieved by simply joining both the tables by JOIN command.

```
mysql> SELECT c.name, o.order_date
-> FROM customers c
-> JOIN orders o ON c.id = o.customer_id;
```

name	order_date
Arun	2025-05-28
Arun	2025-07-09
Arun	2025-07-09
Meena	2025-06-05
Meena	2025-07-09
Naveen	2025-06-18
Naveen	2025-03-06
Keerthi	2025-06-30
Vijay	2025-07-05
Vijay	2025-05-04
Vijay	2025-03-05
Lakshmi	2025-07-10
Sathya	2025-07-13
Divya	2025-07-15
Divya	2025-02-04
Divya	2025-03-09
Karthik	2025-07-17
Revathi	2025-07-18

18 rows in set (0.00 sec)

7. Retrieve the orders with a total amount greater than 150.00.

All the entries in the table already have the total amount greater than 150.00 so the entire table will be displayed. So the orders with total amount greater than 1000.00 can be retrieved by using relational operator ">".

```
[mysql> SELECT * FROM orders WHERE total_amount > 1000.00;
```

ID	Customer_id	Order_Date	Total_amount
3	3	2025-06-18	1200.00
5	5	2025-07-05	1499.00
10	10	2025-07-18	1300.00

3 rows in set (0.00 sec)

All the orders having amount greater than 1000 are displayed as per the above image.

8. Normalize the database by creating a separate table for order items and updating the orders table to reference the order_items table :

Normalization is done to reduce data redundancy and maintain data integrity by dividing larger tables into smaller, related tables and using foreign keys to define the relationships.

To perform normalization , a new table called order_items have been created. This table have order id and product id as references to the respective order and product tables via foreign key.

```
mysql> CREATE TABLE order_items (  
-> id INT AUTO_INCREMENT PRIMARY KEY,  
-> order_id INT,  
-> product_id INT,  
-> quantity INT,  
-> FOREIGN KEY (order_id) REFERENCES orders(id),  
-> FOREIGN KEY (product_id) REFERENCES products(id)  
-> );  
Query OK, 0 rows affected (0.03 sec)
```

```
mysql> INSERT INTO order_items (order_id, product_id, quantity) VALUES  
-> (1, 6, 1),  
-> (1, 8, 1),  
-> (2, 2, 10),  
-> (2, 3, 4),  
-> (3, 7, 1),  
-> (4, 4, 1);  
Query OK, 6 rows affected (0.01 sec)  
Records: 6 Duplicates: 0 Warnings: 0
```

```
[mysql> select * from order_items;  
+----+-----+-----+-----+  
| id | order_id | product_id | quantity |  
+----+-----+-----+-----+  
| 1 | 1 | 6 | 1 |  
| 2 | 1 | 8 | 1 |  
| 3 | 2 | 2 | 10 |  
| 4 | 2 | 3 | 4 |  
| 5 | 3 | 7 | 1 |  
| 6 | 4 | 4 | 1 |  
+----+-----+-----+-----+  
6 rows in set (0.00 sec)
```

8.a Normalised table ORDER_ITEMS

Now the order_items have been given all the order related entries. To ensure complete normalization the total amount from the orders table can be removed as its no longer required .

9. Get the names of customers who have ordered Product A.

By assuming product A as Pen , the following query is executed.

```
mysql> SELECT DISTINCT c.name
-> FROM customers c
-> JOIN orders o ON c.id = o.customer_id
-> JOIN order_items oi ON o.id = oi.order_id
-> JOIN products p ON oi.product_id = p.id
-> WHERE p.name = 'pen';
+-----+
| name |
+-----+
| Meena |
| Vijay |
+-----+
2 rows in set (0.00 sec)

mysql> █
```

10.Retrieve the average total of all orders.

Since the order_total is not stored anymore we can calculate it by joining order and order_items table and dynamically calculate using SUM(price * quantity).

```
mysql> SELECT
-> AVG(order_total)
-> FROM (
-> SELECT
-> o.id AS order_id,
-> SUM(p.price * oi.quantity) AS order_total
-> FROM orders o
-> JOIN order_items oi ON o.id = oi.order_id
-> JOIN products p ON oi.product_id = p.id
-> GROUP BY o.id
-> ) AS order_totals;
+-----+
| AVG(order_total) |
+-----+
| 15882.250000 |
+-----+
1 row in set (0.00 sec)
```