**ECommerce Database Task**

This project sets up a simple eCommerce database using MySQL with three main tables: `customers`, `orders`, and `products`. The queries include table creation, schema alteration, and common operations like retrieving customer info, updating products, and normalizing the database.

"Using freesqldatabase.com, a SQL server was created at sql12.freesqldatabase.com. phpMyAdmin was used to log in with the provided credentials to create the database, manage tables, and execute SQL queries."

Ecommerce database created using **CREATE DATABASE ecommerce**

Three tables are added inside the database by the following queries.

**Tabel : 1**

CREATE TABLE customers (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100) NOT NULL,

email VARCHAR(100) NOT NULL UNIQUE,

address TEXT

);

**Tabel : 2**

CREATE TABLE orders (

id INT AUTO\_INCREMENT PRIMARY KEY,

customer\_id INT ,

order\_date DATE,

total\_amount INT

)

Alter the table with foreign key constraint for customer\_id

ALTER TABLE `orders` ADD FOREIGN KEY (`customer\_id`) REFERENCES `customers`(`id`) ON DELETE RESTRICT ON UPDATE RESTRICT

**Tabel : 3**

CREATE TABLE products (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(30) ,

Price INT,

description VARCHAR(100)

)

**Answers for Queries**

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

**Ans: SELECT customers.name FROM `customers` LEFT JOIN orders on customers.id = orders.customer\_id WHERE DATE\_SUB(CURDATE(),INTERVAL 30 DAY) <= orders.order\_date**

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

**Ans: SELECT SUM(orders.total\_amount) FROM `orders`**

3.Update the price of Product C to 45.00.

**Ans: UPDATE products SET price = 45 WHERE name ="C"**

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

**Ans: ALTER TABLE products ADD discount INT**

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

**Ans: SELECT \* FROM `products` ORDER BY price DESC limit 3**

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

**Ans:** Step:1 - ALTER TABLE orders ADD COLUMN product\_id INT UNSIGNED

Step:2 - ALTER TABLE orders ADD CONSTRAINT fk\_product FOREIGN KEY (product\_id) REFERENCES products(id)

Step:3- **SELECT customers.name FROM customers LEFT JOIN orders ON orders.customer\_id = customers.id JOIN products ON orders.product\_id = products.id WHERE products.name = "A**"

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

**Ans: SELECT customers.name,orders.order\_date FROM `orders` LEFT JOIN customers on orders.customer\_id = customers.id**

(or)

**SELECT customers.name,orders.order\_date FROM `customers` JOIN orders ON customers.id = orders.customer\_id**

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

**Ans: SELECT \* FROM `orders` WHERE orders.total\_amount >= 150**

9.Normalize the database by creating a separate table for order items and updating the orders table to reference the order\_items table.

**Ans: Step:1 - CREATE TABLE `ecommerce`.`order\_items` ( `id` INT UNSIGNED NOT NULL AUTO\_INCREMENT , `order\_id` INT UNSIGNED NOT NULL , `product\_id` INT UNSIGNED NOT NULL , `quantity` INT UNSIGNED NOT NULL , `price` DECIMAL NOT NULL , PRIMARY KEY (`id`))**

**Step:2 - ALTER TABLE `order\_items` ADD FOREIGN KEY (`product\_id`) REFERENCES `products`(`id`) ON DELETE RESTRICT ON UPDATE RESTRICT;**

**ALTER TABLE `order\_items` ADD FOREIGN KEY (`order\_id`) REFERENCES `orders`(`id`) ON DELETE RESTRICT ON UPDATE RESTRICT;**

Example : retrieve data for customer who purchased all the products in order id 1

SELECT \* FROM `order\_items` JOIN products on order\_items.product\_id =products.id WHERE order\_items.order\_id= 1

10.Retrieve the average total of all orders.

**Ans: SELECT ROUND(AVG(total\_amount)) AS Average FROM `orders`**