ASSIGNMENT: 2 - CUSTOMER AND ORDER

Part 1 :

```
CREATE TABLE customers (
customer_id INT PRIMARY KEY,
name VARCHAR(100),
email VARCHAR(100) UNIQUE,
city VARCHAR(50),
signup_date DATE
);

CREATE TABLE orders (
order_id INT PRIMARY KEY,
customer_id INT ,
order_date DATE,
total_amount DECIMAL(10,2),
FOREIGN KEY(customer_id) REFERENCES customers(customer_id)
);

CREATE TABLE products (
product_id INT PRIMARY KEY,
product_name VARCHAR(100) ,
category VARCHAR(50),
price DECIMAL(10,2)
);

CREATE TABLE order_details (
orderdetail_id INT PRIMARY KEY,
order_id INT ,
product_id INT,
quantity INT,
price DECIMAL(10,2),
FOREIGN KEY(order_id) REFERENCES orders(order_id) ,
FOREIGN KEY(product_id) REFERENCES products(product_id)
);

INSERT INTO customers VALUES
(1,'Tom Tills','tillstom@gmail.com','Paris','2024-02-15'),
(2,'Jerry Milver','jerry@gmail.com','New York','2023-06-25'),
(3,'Kevin Kurias','kevinkurias21@gmail.com','Kochi','2024-01-05'),
(4,'Aiwy Ann','aiwy@gmail.com','London','2023-04-30'),
(5,'Lynn Li','lilynn@gmail.com','Taiwan','2025-03-10'),
(6,'Akshay Sharma','akshaysharma@gmail.com','Delhi','2024-12-19');
```

```
INSERT INTO products VALUES
(1, 'Laptop', 'Electronics', 1200.00),
(2, 'Headphones', 'Electronics', 150.00),
(3, 'Bag', 'Accessories', 80.00),
(4, 'Water Bottle', 'Accessories', 20.00),
(5, 'Desk Chair', 'Furniture', 300.00);

INSERT INTO orders VALUES
(101, 1, '2024-04-01', 1350.00),
(102, 2, '2024-02-02', 150.00),
(103, 1, '2024-04-28', 300.00),
(104, 3, '2024-11-10', 160.00),
(105, 4, '2024-08-15', 20.00);

INSERT INTO order_details VALUES
(1001, 101, 1, 1, 1200.00),
(1002, 101, 2, 1, 150.00),
(1003, 103, 5, 1, 300.00),
(1004, 104, 3, 2, 160.00),
(1005, 105, 4, 1, 20.00);
```

Part 2:

Basic Queries

1. Get the list of all customers.
   ```
   SELECT name FROM customers;
   ```

2. Find all orders placed in the last 30 days.
   ```
   SELECT * FROM orders
   WHERE order_date >= DATE_SUB(CURRENT_DATE(),INTERVAL 30 DAY);
   ```

3. Show product names and their prices.
   ```
   SELECT product_name,price FROM products;
   ```

4. Find the total number of products in each category.
   ```
   SELECT COUNT(product_name),category FROM products
   GROUP BY category;
   ```

Filtering and Conditions

5. Get all customers from the city 'Mumbai'.
   ```
   SELECT name FROM customers WHERE city="Mumbai";
   ```

6. Find orders with a total amount greater than ₹5000
```
SELECT * FROM orders WHERE total_amount>5000;
```

7. List customers who signed up after '2024-01-01'.
```
SELECT name FROM customers WHERE signup_date >'2024-01-01';
```

Joins

8. Show all orders along with the customer's name.
```
SELECT orders.order_id, orders.order_date, orders.total_amount,
customers.name
FROM orders JOIN customers ON
customers.customer_id=orders.customer_id;
```

9. List products purchased in each order.
```
SELECT order_details.order_id, products.product_name FROM order_details
JOIN products ON order_details.product_id = products.product_id;
```

10. Find customers who have never placed an order.

```
SELECT customers.name FROM customers
LEFT JOIN orders ON customers.customer_id=orders.customer_id
WHERE orders.customer_id is NULL;
```

Aggregation and Grouping

11. Find the total amount spent by each customer.
```
SELECT c.customer_id,c.name, SUM(o.total_amount)
FROM customers c JOIN orders o ON c.customer_id=o.customer_id
GROUP BY c.customer_id;
```

12. Which product has been sold the most (by quantity)?
```
SELECT p.product_id, p.product_name, SUM(od.quantity)
FROM products p JOIN order_details od ON p.product_id=od.product_id
GROUP BY p.product_id
ORDER BY SUM(od.quantity) DESC LIMIT 1;
```

13. Find the average order value for each customer.
```
SELECT c.customer_id , c.name , AVG(o.total_amount)
FROM customers c JOIN orders o ON c.customer_id=o.customer_id
GROUP BY c.customer_id;
```

14. Total sales amount per product category.

```
SELECT p.category, SUM(od.price) AS total_sales
FROM products p JOIN order_details od ON p.product_id=od.product_id
GROUP BY p.category;
```

Subqueries

15. Find customers who spent more than the average spending.

```
SELECT c.customer_id, c.name, SUM(o.total_amount) as total_spending
FROM customers c JOIN orders o ON c.customer_id=o.customer_id
GROUP BY o.customer_id having total_spending >
(SELECT AVG(total_amount ) FROM orders);
```

16. List products that have never been ordered.

```
SELECT products.product_id,products.product_name FROM products
WHERE products.product_id NOT IN
(SELECT order_details.product_id from order_details);
```

17. Find the most recent order for each customer

```
SELECT c.customer_id, c.name, o.order_id, o.total_amount, o.order_date
FROM customers c JOIN orders o ON c.customer_id=o.customer_id
WHERE o.order_date IN
(SELECT MAX(ord.order_date) FROM orders ord
WHERE ord.customer_id=c.customer_id)
ORDER BY o.order_date DESC;
```

Advanced Queries

18. Rank customers by total spending (highest first).

```
SELECT c.customer_id, c.name, SUM(o.total_amount) as total_spending
FROM customers c JOIN orders o ON c.customer_id = o.customer_id
GROUP BY o.customer_id
ORDER BY total_spending DESC;
```

19. Get the top 3 customers based on the number of orders placed.

```
SELECT c.customer_id, c.name, COUNT(o.order_id) as total_order
FROM customers c JOIN orders o ON c.customer_id=o.customer_id
GROUP BY o.customer_id
ORDER BY total_order DESC LIMIT 3;
```

20. For each product, find how many unique customers have purchased it.

```
SELECT p.product_id, p.product_name, COUNT(DISTINCT o.customer_id) AS
unique_customers FROM products p
JOIN order_details od ON p.product_id = od.product_id
JOIN orders o ON od.order_id = o.order_id
GROUP BY p.product_id, p.product_name;
```