Assignment No.6

Name :- Shailesh Pawar

PRN :- 123B1B230

Solve following SQL Queries:

1. Retrieve all customers along with their corresponding orders (including customers who haven't ordered).

Code :-

```
SELECT *FROM Customers_230
left OUTER JOIN Orders_230
ON (Customers_230.customer_id = Orders_230.customer_id);
```

Output :-

```
sql> SELECT *FROM Customers_230
left OUTER JOIN Orders_230
ON (Customers_230.customer_id = Orders_230.customer_id) LIMIT 0, 1001;
```

customer_id	name	email	city	order_id	customer_id	order_date	total_amount
1	Alice Johnson	alice@example.com	New York	1	1	2024-02-15	150.00
1	Alice Johnson	alice@example.com	New York	4	1	2024-03-05	250.00
2	Bob Smith	bob@example.com	Los Angeles	2	2	2024-02-18	300.00
2	Bob Smith	bob@example.com	Los Angeles	6	2	2024-03-12	500.00
3	Charlie Brown	charlie@example.com	Chicago	3	3	2024-02-20	450.00
4	David Wilson	david@example.com	New York	5	4	2024-03-10	120.00
5	Eve Adams	eve@example.com	San Francisco	7	5	2024-03-14	350.00
6	Frank White	frank@example.com	Seattle	8	6	2024-03-20	275.00
7	Grace Green	grace@example.com	Houston	9	7	2024-03-22	600.00
8	Henry Baker	henry@example.com	Miami	10	8	2024-03-25	700.00
9	Isla Turner	isla@example.com	Boston	null	null	null	null
10	Jack Cooper	jack@example.com	Denver	null	null	null	null

2. List all orders along with the product names and their quantities.

Code :-

```
SELECT order_id, product_name, quantity
FROM Order_Items_230 as oi
JOIN Products_230 as p ON oi.product_id = p.product_id
ORDER BY oi.order_id;
```

Output :-

order_id	product_name	quantity
1	Laptop	1
1	Headphones	2
2	Keyboard	1
2	Mouse	2
3	Monitor	1
3	Table	1
4	Headphones	1
5	Keyboard	2
6	Laptop	2
7	Smartphone	1

3. Find the total number of orders placed by each customer.

Code :-

```
SELECT c.customer_id,name,COUNT(order_id) as "Total Order" FROM Customers_230 as c
JOIN Orders_230 as o
ON (c.customer_id = o.customer_id)
GROUP BY o.customer_id;
```

Output :-

customer_id	name	Total Order
1	Alice Johnson	2
2	Bob Smith	2
3	Charlie Brown	1
4	David Wilson	1
5	Eve Adams	1
6	Frank White	1
7	Grace Green	1
8	Henry Baker	1

4. Find the total number of products available in each category.

Code :-

```
SELECT category,COUNT(product_id) as "Available Product" FROM Products_230
GROUP BY category;
```

Output :-

category	Available Product
Electronics	6
Furniture	3
Accessories	1

5. Retrieve the order details, including customer name and total amount, for all orders placed in the last 30 days.

Code :-

```
SELECT c.customer_id, c.name, SUM(o.total_amount) AS "Total Amount"
FROM Customers_230 c
JOIN Orders_230 o
ON c.customer_id = o.customer_id
WHERE o.order_date >= DATE_SUB('2024-03-25', interval 30 day)
GROUP BY c.customer_id, c.name;
```

Output :-

customer_id	name	Total Amount
1	Alice Johnson	250.00
4	David Wilson	120.00
2	Bob Smith	500.00
5	Eve Adams	350.00
6	Frank White	275.00
7	Grace Green	600.00
8	Henry Baker	700.00

7. Show all customers who have never placed an order.

Code :-

```
SELECT *FROM Customers_230 c
LEFT OUTER JOIN Orders_230 o
on c.customer_id = o.customer_id
WHERE o.customer_id IS NULL;
```

Output :-

customer_id	name	email	city	order_id	customer_id	order_date	total_amount
9	Isla Turner	isla@example.com	Boston	null	null	null	null
10	Jack Cooper	jack@example.com	Denver	null	null	null	null

8. Retrieve details of orders where the total amount is greater than the average order total.(solve suing subquery)

Code :-

```
SELECT customer_id, SUM(total_amount) AS total_amount FROM Orders_230
GROUP BY customer_id
HAVING SUM(total_amount) > (SELECT AVG(total_amount) FROM Orders_230);
```

Output :-

customer_id	total_amount
1	400.00
2	800.00
3	450.00
7	600.00
8	700.00

9. Find customers who have placed at least two orders.

Code :-

```
SELECT c.customer_id,c.name,COUNT(o.order_id) AS order_count
FROM customers_230 AS c
```

```
INNER JOIN orders_230 AS o
ON c.customer_id = o.customer_id
GROUP BY c.customer_id, c.name
HAVING COUNT(o.order_id) >= 2;
```

Output :-

customer_id	name	order_count
1	Alice Johnson	2
2	Bob Smith	2

10. Find the top 3 most ordered products based on quantity sold.

Code :-

```
SELECT o.product_id,p.product_name,sum(o.quantity) from Order_Items_230 as o
INNER JOIN products_230 as p
ON o.product_id = p.product_id
GROUP by o.product_id
LIMIT 3;
```

Output :-

product_id	product_name	sum(o.quantity)
1	Laptop	3
2	Headphones	3
3	Keyboard	3

13. Find all products that have never been ordered.

Code :-

```
SELECT p.* FROM products_230 as p
LEFT OUTER JOIN order_items_230 as o
ON p.product_id = o.product_id
WHERE o.product_id IS NULL;
```

Output :-

product_id	product_name	category	price
7	Chair	Furniture	90.00
8	Desk Lamp	Furniture	40.00
10	Backpack	Accessories	80.00

14. Retrieve the names of customers who have placed the highest number of orders.

Code :-

```
SELECT c.customer_id,c.name,count(o.customer_id) as Order_c from Customers_230 as c
join Orders_230 as o
on c.customer_id = o.customer_id
GROUP by c.customer_id,c.name
ORDER BY Order_c DESC
LIMIT 1;
```

Output :-

customer_id	name	Order_count
1	Alice Johnson	2

15. Find all customers who have ordered m

ore than 5 different products.

(Hint: Use COUNT DISTINCT on product_id.)

Code :-

```
SELECT o.customer_id
FROM orders_230 o
JOIN order_items_230 oi ON o.order_id = oi.order_id
GROUP BY o.customer_id
HAVING COUNT(DISTINCT oi.product_id) > 5;
```

Output :

```
OK, 0 records retrieved in 1.31ms
```

16. Find products that are sold by at least two different sellers but have never been ordered.

(Hint: Use HAVING COUNT(DISTINCT seller_id) > 1 and NOT EXISTS.)

Code :-

```
SELECT p.product_id
FROM products_230 p
JOIN order_items_230 oi ON p.product_id = oi.product_id
WHERE oi.order_id IS NULL
GROUP BY p.product_id
HAVING COUNT(DISTINCT oi.product_id) > 1;
```

Output :-

```
OK, 0 records retrieved in 1.075ms
```

17. Find the customer who has spent the most money overall.

(Hint: Use SUM and ORDER BY.)

Code :-

```
SELECT o.customer_id, SUM(o.total_amount) AS total_spent
FROM orders_230 o
GROUP BY o.customer_id
ORDER BY total_spent DESC
LIMIT 1;
```

Output :-

```
+-------------+-------------+
| customer_id | total_spent |
+-------------+-------------+
|           2 |      800.00 |
+-------------+-------------+
```

18. Find all customers who have either placed an order or live in the same city as a seller.

(Hint: Use UNION to combine customers who placed orders and those in seller cities.)

Code :-

```
SELECT DISTINCT c.customer_id
FROM customers_230 c
JOIN orders_230 o ON c.customer_id = o.customer_id
UNION
SELECT DISTINCT c.customer_id
FROM customers_230 c
JOIN sellers_230 s ON c.city = s.city;
```

Output :-

customer_id
1
2
3
4
5
6
7
8
9
10

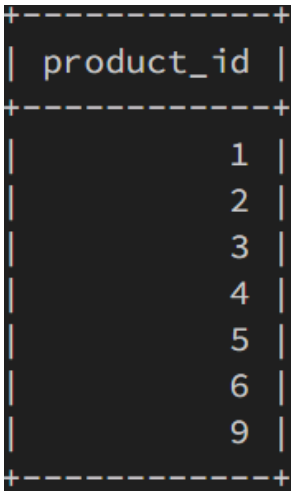
19. Retrieve all products that are either in stock with at least one seller or have been ordered at least once.

(Hint: Use UNION between Product_Sellers and Order_Items.)

Code :-

```
SELECT DISTINCT p.product_id
FROM products_230 p
WHERE p.product_id IN (SELECT product_id FROM order_items_230)
UNION
SELECT DISTINCT p.product_id
FROM products_230 p
WHERE EXISTS (SELECT 1 FROM order_items_230 oi WHERE oi.product_id = p.product_id);
```

Output :-



product_id
1
2
3
4
5
6
9

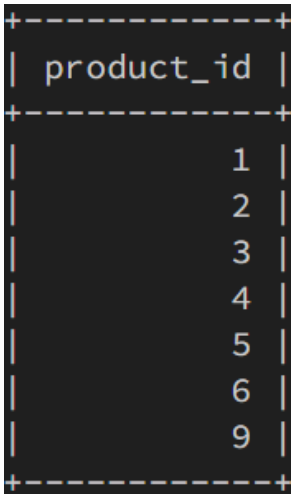
20. Retrieve products that have been both ordered and are currently in stock.

(Hint: Use INTERSECT between Order_Items and Product_Sellers.)

Code :-

```
SELECT DISTINCT oi.product_id
FROM order_items_230 oi
INTERSECT
SELECT DISTINCT p.product_id
FROM products_230 p;
```

Output :-



product_id
1
2
3
4
5
6
9

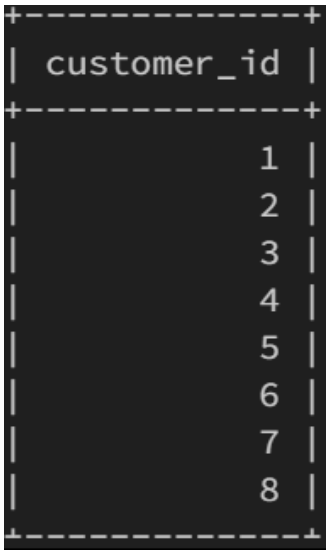
21. Find customers who have both placed an order and live in a city where a seller is located.

(Hint: Use INTERSECT between customers who have placed orders and customers who

Code :-

```
SELECT DISTINCT c.customer_id
FROM customers_230 c
JOIN orders_230 o ON c.customer_id = o.customer_id
INTERSECT
SELECT DISTINCT c.customer_id
FROM customers_230 c
JOIN sellers_230 s ON c.city = s.city;
```

Output :-



customer_id
1
2
3
4
5
6
7
8

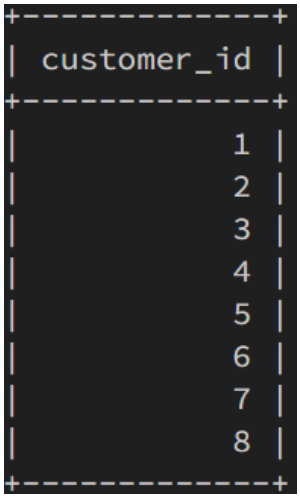
22. Retrieve all customers who have placed at least one order in each year available in the database.

(Hint: Use INTERSECT with orders grouped by year.)

Code :-

```
SELECT customer_id FROM (
    SELECT customer_id, EXTRACT(YEAR FROM order_date) AS order_year
    FROM orders_230
    GROUP BY customer_id, order_year
) AS customer_years GROUP BY customer_id
HAVING COUNT(DISTINCT order_year) = (
    SELECT COUNT(DISTINCT EXTRACT(YEAR FROM order_date)) FROM orders_230);
```

Output :-



customer_id
1
2
3
4
5
6
7
8