SQL for Data Analysis - Outputs

SQL Queries:

```
-- 1. SELECT basic data
SELECT * FROM customers;

-- 2. SELECT with WHERE, ORDER BY
SELECT * FROM products
WHERE price > 50
ORDER BY price DESC;

-- 3. GROUP BY and aggregate function (SUM)
SELECT customer_id, SUM(quantity) AS total_items_purchased
FROM orders
GROUP BY customer_id
ORDER BY total_items_purchased DESC;

-- 4. INNER JOIN customers and orders
SELECT c.name, o.order_id, o.order_date
FROM customers c
INNER JOIN orders o ON c.customer_id = o.customer_id;

-- 5. LEFT JOIN to find customers who haven't placed orders
SELECT c.name, o.order_id
FROM customers c
LEFT JOIN orders o ON c.customer_id = o.customer_id
WHERE o.order_id IS NULL;

-- 6. Subquery: Find products with price higher than the average price
SELECT * FROM products
WHERE price > (SELECT AVG(price) FROM products);

-- 7. Aggregate function (AVG)
SELECT AVG(price) AS average_product_price FROM products;

-- 8. Create a VIEW to show customer order summary
CREATE VIEW customer_order_summary AS
SELECT c.customer_id, c.name, COUNT(o.order_id) AS number_of_orders, SUM(o.quantity) AS total_items
FROM customers c
LEFT JOIN orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_id, c.name;
```

SQL for Data Analysis - Outputs

-- 9. Query the view
SELECT * FROM customer_order_summary;

-- 10. Create INDEX to optimize query
CREATE INDEX idx_customer_id ON orders(customer_id);

Sample Outputs:

Table: Customers

1 | Alice | alice@email.com
2 | Bob | bob@email.com
3 | Charlie | charlie@email.com

Table: Products

101 | Laptop | $999
102 | Phone | $699
103 | Headphones | $199

Table: Orders

1001 | Customer 1 | Product 101 | Qty 1 | 2024-05-01
1002 | Customer 2 | Product 103 | Qty 2 | 2024-05-03
1003 | Customer 1 | Product 102 | Qty 1 | 2024-05-04