10 Commonly Asked SQL for Data Analytics Interview Questions with Answers and Explanations:

1. Find the total number of customers and the average order value:

SQL
```
SELECT COUNT(*) AS total_customers, AVG(amount) AS avg_order_value
FROM orders;
```

Explanation:

- COUNT(*) counts all rows in the orders table, giving the total number of customers.
- AVG(amount) calculates the average of the amount column, representing the average order value.

2. Identify the top 5 products with the highest sales in the last quarter:

SQL
```
SELECT product_id, product_name, SUM(amount) AS total_sales
FROM orders
WHERE order_date >= DATE_SUB(CURDATE(), INTERVAL 3 MONTH)
GROUP BY product_id, product_name
ORDER BY total_sales DESC
LIMIT 5;
```

Explanation:

- DATE_SUB(CURDATE(), INTERVAL 3 MONTH) filters for orders placed in the last three months.
- GROUP BY product_id, product_name aggregates sales for each product.
- SUM(amount) calculates the total sales for each product.
- ORDER BY total_sales DESC sorts products by descending sales.
- LIMIT 5 shows the top 5 products.

3. Calculate the year-to-date revenue for each region and compare it to the previous year.

SQL
```
WITH yearly_revenue AS (
  SELECT region, YEAR(order_date) AS order_year, SUM(amount) AS
revenue
   FROM orders
   GROUP BY region, YEAR(order_year)
)
SELECT region,
       r.revenue AS current_year_revenue,
       p.revenue AS previous_year_revenue,
       ((r.revenue - p.revenue) / p.revenue) * 100 AS
growth_percentage
FROM yearly_revenue AS r
LEFT JOIN (
  SELECT region, YEAR(order_date) AS order_year, SUM(amount) AS
revenue
  FROM orders
  WHERE YEAR(order_date) = YEAR(CURDATE()) - 1
  GROUP BY region, YEAR(order_year)
) AS p ON r.region = p.region AND r.order_year = p.order_year
ORDER BY region;
```

Explanation:

- A Common Table Expression (CTE) yearly_revenue calculates revenue for each region and year.
- The main query joins yearly_revenue with a subquery filtering for the previous year.
- growth_percentage calculates the year-over-year change in revenue.
- The final result shows revenue and growth for each region.

4. Find customers who placed an order but haven't completed it yet.

```
SQL
SELECT c.customer_id, c.name, o.order_date
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
WHERE o.order_status = 'PENDING';
```

Explanation:

- Joins customers and orders tables on customer_id.
- Filters for orders with order_status equal to 'PENDING'.
- Shows customer details and order date for pending orders.

5. Identify the most active day of the week for online orders.

```
SQL
SELECT DAYNAME(order_date) AS day_of_week, COUNT(*) AS order_count
FROM orders
WHERE order_channel = 'ONLINE'
GROUP BY DAYNAME(order_date)
ORDER BY order_count DESC
LIMIT 1;
```

Explanation:

- DAYNAME(order_date) extracts the day of the week from the order_date column.

- Filters for orders from the 'ONLINE' channel.
- COUNT(*) counts the number of orders for each day.
- ORDER BY order_count DESC sorts days by order count in descending order.
- LIMIT 1 shows the day with the most online orders.

6. Calculate the customer lifetime value (CLTV) for each customer.

```sql
  SQL
WITH customer orders AS (
  SELECT customer_id, order_date, amount
  FROM orders
)
SELECT c.customer_id, c.name, SUM(o.amount) AS total_spend,
       SUM(o.amount) / DATEDIFF(CURDATE(), MIN(o.order_date)) AS cltv
FROM customers c
JOIN customer_orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_id, c.name;
```

Explanation:

- A CTE customer_orders selects relevant data from the orders table.
- The main query joins customers with customer_orders on customer_id.
- SUM(o.amount) calculates the total spend for each customer.
- DATEDIFF(CURDATE(), MIN(o.order_date)) calculates the difference between the current date and the first order date for each customer, representing their customer lifetime.
- cltv divides total spend by customer lifetime to derive the average spend per day, essentially estimating CLTV.

7. Find the top 10 employees with the highest number of sales in the current year.

```sql
  SQL
SELECT e.employee_id, e.name, COUNT(*) AS sales_count
FROM employees e
JOIN orders o ON e.employee_id = o.employee_id
WHERE YEAR(order_date) = YEAR(CURDATE())
GROUP BY e.employee_id, e.name
ORDER BY sales_count DESC
LIMIT 10;
```

Explanation:

- Joins employees and orders tables on employee_id.
- Filters for orders placed in the current year using YEAR(order_date).
- COUNT(*) counts the number of orders for each employee, representing their sales count.
- ORDER BY sales_count DESC sorts by sales count in descending order.
- LIMIT 10 shows the top 10 employees.

8. Identify products with a positive sales growth between Q1 and Q2 of the current year.

SQL
```
SELECT product id, product name,
       SUM(CASE WHEN quarter(order_date) = 1 THEN amount ELSE 0 END)
AS q1_sales,
       SUM(CASE WHEN quarter(order_date) = 2 THEN amount ELSE 0 END)
AS q2_sales,
       ((q2_sales - q1_sales) / q1_sales) * 100 AS growth_percentage
FROM orders
WHERE YEAR(order date) = YEAR(CURDATE())
GROUP BY product_id, product_name
HAVING growth_percentage > 0;
```

Explanation:

- Uses CASE statement to separate sales into Q1 and Q2 based on the quarter function.
- q1_sales and q2_sales sum sales for each quarter for each product.
- growth_percentage calculates the percentage change in sales between Q1 and Q2.
- HAVING growth_percentage > 0 filters for products with positive growth.

9. Find customers who made purchases in both online and offline channels (separate tables) and calculate their total spending.

SQL
```
SELECT o.customer_id, c.name, SUM(o.amount) AS online_spending,
SUM(of.amount) AS offline_spending,
       SUM(o.amount + of.amount) AS total_spending
FROM online_orders o
JOIN customers c ON c.customer_id = o.customer_id
LEFT JOIN offline_orders of ON of.customer_id = o.customer_id
GROUP BY o.customer_id, c.name;
```

Explanation:

- Joins online_orders and customers tables on customer_id.
- Left joins offline_orders to include potential missing offline purchases.
- SUM functions calculate spending in each channel and combined total spending.
- This identifies customers who have used both channels.

10. Use a CTE to calculate the rolling weekly average sales for each product category for the past year.

SQL
```
WITH weekly sales AS (
  SELECT product_category, date_trunc('week', order_date) AS
order_week, SUM(amount) AS weekly_sales
   FROM orders
   WHERE order_date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
   GROUP BY product_category, order_week
)
```