SQL

Mastering Date & Time Functions

```
CREATE TABLE sales (
   order_id INT PRIMARY KEY AUTO_INCREMENT,
   product_id INT,
   sale_date DATE,
   amount DECIMAL(10, 2)
);

INSERT INTO sales (product_id, sale_date, amount) VALUES
(1, '2022-01-05', 100.50),
(2, '2022-01-15', 250.75),
(3, '2022-02-01', 50.00),
(1, '2022-02-10', 120.00),
(2, '2022-03-05', 300.00),
(3, '2022-03-25', 75.25),
(1, '2022-04-10', 150.00),
(2, '2022-04-20', 200.00),
(3, '2022-05-05', 60.00),
(1, '2022-06-12', 130.00),
(2, '2022-06-25', 270.00),
```

(3, '2022-07-01', 80.00),

(1, '2022-07-15', 160.00),

(2, '2022-08-01', 320.00),

(3, '2022-08-18', 90.00),

(1, '2022-09-05', 110.00),

(2, '2022-09-20', 280.00),

(3, '2022-10-01', 100.00),

(1, '2022-10-15', 170.00),

(2, '2022-11-05', 290.00),

(3, '2022-11-25', 115.00),

(1, '2022-12-01', 180.00),

(2, '2022-12-10', 350.00),

(3, '2022-12-31', 130.00),

(1, '2023-01-05', 190.00),

(2, '2023-01-18', 360.00),

(3, '2023-02-10', 140.00),

(1, '2023-02-25', 200.00),

(2, '2023-03-10', 370.00),

(3, '2023-03-30', 150.00),

(1, '2023-04-15', 210.00),

(2, '2023-04-28', 380.00),

(3, '2023-05-05', 160.00),

(1, '2023-06-01', 220.00),

(2, '2023-06-15', 390.00),

(3, '2023-07-05', 170.00),

(1, '2023-07-20', 230.00),

```
(2, '2023-08-01', 400.00),
(3, '2023-08-15', 180.00),
(1, '2023-09-01', 240.00),
(2, '2023-09-18', 410.00),
(3, '2023-10-05', 190.00),
(1, '2023-10-25', 250.00),
(2, '2023-11-10', 420.00),
(3, '2023-11-20', 200.00),
(1, '2023-12-01', 260.00),
(2, '2023-12-15', 430.00),
(3, '2023-12-29', 210.00);
```

-- 1. Calculate the total sales for each month of the year 2023.

```
SELECT
  DATE_FORMAT(sale_date, '%Y-%m') AS sale_month,
  SUM(amount) AS total_sales
FROM
  sales
WHERE
  YEAR(sale_date) = 2023
GROUP BY
  sale_month
ORDER BY
  sale_month;
```

-- 2. Find the number of days between the first and last order.

```
SELECT
  DATEDIFF(MAX(sale_date), MIN(sale_date)) AS days_between_first_last_order
FROM
  sales;
```

-- 3. What were the sales for the last 90 days from the latest order date?

```
SELECT
  SUM(amount) AS sales_in_last_90_days
FROM
  sales
WHERE
  sale_date >= DATE_SUB((SELECT MAX(sale_date) FROM sales), INTERVAL 90 DAY);
```

-- 4. Compare the total sales of the year 2023 with the year 2022.

```
SELECT
  SUM(CASE WHEN YEAR(sale_date) = 2023 THEN amount ELSE 0 END) AS
total_sales_2023,
  SUM(CASE WHEN YEAR(sale_date) = 2022 THEN amount ELSE 0 END) AS
total_sales_2022
FROM
  sales
WHERE
  YEAR(sale_date) IN (2022, 2023);
```

-- 5. Calculate the 6-month moving average of sales for each month in 2023.

```
SELECT
  DATE_FORMAT(sale_date, '%Y-%m') AS sale_month,
  AVG(SUM(amount)) OVER (ORDER BY DATE_FORMAT(sale_date, '%Y-%m') ROWS BETWEEN 5 PRECEDING AND CURRENT ROW) AS moving_average_sales
FROM
  sales
WHERE
  YEAR(sale_date) = 2023
GROUP BY
  sale_month
ORDER BY
  sale_month;
```

-- 6. Find the year-over-year sales growth percentage.

```
WITH monthly_sales AS (
  SELECT
    YEAR(sale_date) AS sale_year,
    SUM(amount) AS yearly_sales
  FROM sales
  GROUP BY sale_year
)
SELECT
  s1.sale_year,
  s1.yearly_sales,
```

```
    s2.yearly_sales AS previous_year_sales,
    ((s1.yearly_sales - s2.yearly_sales) / s2.yearly_sales) * 100 AS yoy_growth_percentage
FROM
    monthly_sales s1
JOIN
    monthly_sales s2 ON s1.sale_year = s2.sale_year + 1
ORDER BY
    s1.sale_year;
```

-- 7. List all sales that occurred on a weekend (Saturday or Sunday).

```
SELECT
    order_id,
    sale_date,
    amount
FROM
    sales
WHERE
    DAYOFWEEK(sale_date) IN (1, 7); -- 1=Sunday, 7=Saturday
```

-- 8. Calculate the cumulative sales total for each day in 2023.

```
SELECT
    sale_date,
    SUM(amount) AS daily_sales,
    SUM(SUM(amount)) OVER (ORDER BY sale_date) AS cumulative_sales
FROM
```

```
  sales
WHERE
  YEAR(sale_date) = 2023
GROUP BY
  sale_date
ORDER BY
  sale_date;
```

-- 9. Find the product that had the highest sales in each quarter of 2022.

```
WITH quarterly_product_sales AS (
  SELECT
    QUARTER(sale_date) AS quarter,
    product_id,
    SUM(amount) AS total_sales
  FROM
    sales
  WHERE
    YEAR(sale_date) = 2022
  GROUP BY
    quarter, product_id
),
ranked_sales AS (
  SELECT
    quarter,
    product_id,
```

```
    total_sales,
    ROW_NUMBER() OVER (PARTITION BY quarter ORDER BY total_sales DESC) as rn
  FROM
    quarterly_product_sales
)
SELECT
  quarter,
  product_id,
  total_sales
FROM
  ranked_sales
WHERE
  rn = 1;
```

-- 10. Determine the number of days since the last sale for each product.

```
SELECT
  product_id,
  DATEDIFF(
    (SELECT MAX(sale_date) FROM sales),
    MAX(sale_date)
  ) AS days_since_last_sale
FROM
  sales
GROUP BY
  product_id;
```