TASK 6: Sales Trend Analysis Using Aggregations

Objective:

Analyze monthly revenue and order volume to identify sales trends, top-performing months, products, and month-on-month growth.

📂 Dataset Description:

Table Name: online_sales

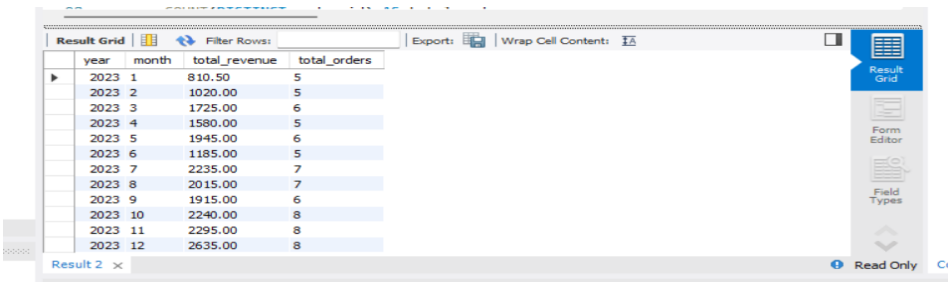
Columns:

- order_id (INT) → Unique identifier for each order
- order_date (DATE) → Date when the order was placed
- amount (DECIMAL) → Order amount in currency
- product_id (INT) → Unique product identifier

The dataset contains 76 rows of sample sales transactions for the year 2023, covering 12 months with multiple products.

1. Monthly Sales Trend (Revenue + Order Count)

```
SELECT
    YEAR(order_date) AS year,
    MONTH(order_date) AS month,
    SUM(amount) AS total_revenue,
    COUNT(DISTINCT order_id) AS total_orders
FROM online_sales
GROUP BY YEAR(order_date), MONTH(order_date)
ORDER BY year, month;
```



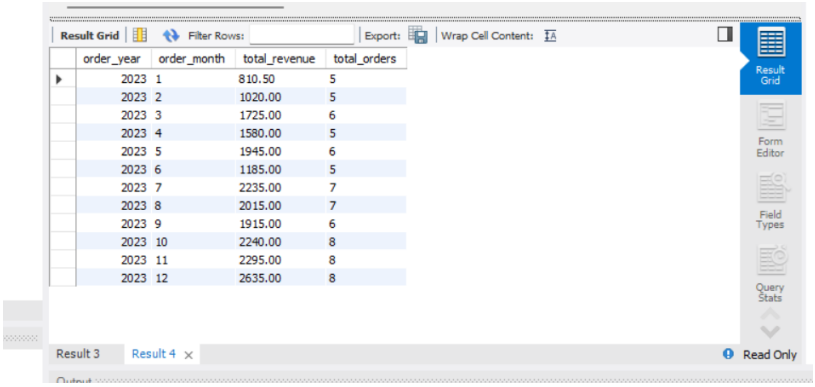
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

year	month	total_revenue	total_orders
2023	1	810.50	5
2023	2	1020.00	5
2023	3	1725.00	6
2023	4	1580.00	5
2023	5	1945.00	6
2023	6	1185.00	5
2023	7	2235.00	7
2023	8	2015.00	7
2023	9	1915.00	6
2023	10	2240.00	8
2023	11	2295.00	8
2023	12	2635.00	8

Result Grid | Form Editor | Field Types | Result 2 | Read Only

2. Monthly Sales Trend (Alternate Column Aliases)

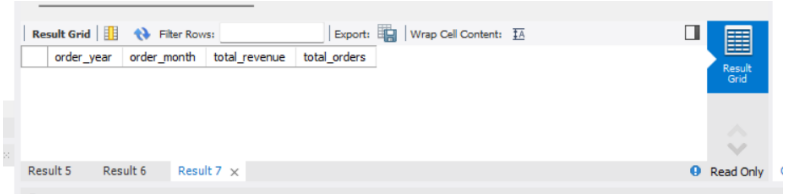
```
SELECT
    YEAR(order_date) AS order_year,
    MONTH(order_date) AS order_month,
    SUM(amount) AS total_revenue,
    COUNT(DISTINCT order_id) AS total_orders
FROM online_sales
GROUP BY order_year, order_month
ORDER BY order_year, order_month;
```



order_year	order_month	total_revenue	total_orders
2023	1	810.50	5
2023	2	1020.00	5
2023	3	1725.00	6
2023	4	1580.00	5
2023	5	1945.00	6
2023	6	1185.00	5
2023	7	2235.00	7
2023	8	2015.00	7
2023	9	1915.00	6
2023	10	2240.00	8
2023	11	2295.00	8
2023	12	2635.00	8

3. Monthly Sales Trend for Year 2024 Only

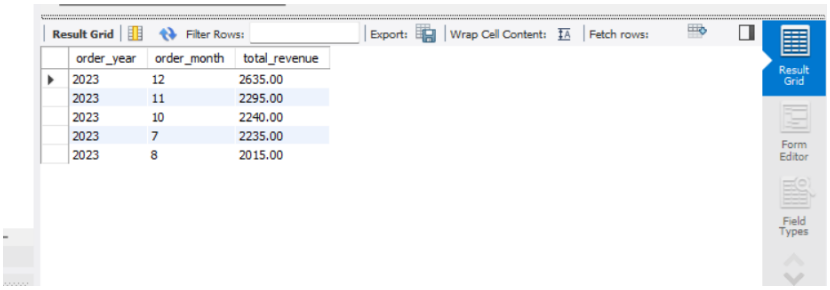
```
SELECT
    YEAR(order_date) AS order_year,
    MONTH(order_date) AS order_month,
    SUM(amount) AS total_revenue,
    COUNT(DISTINCT order_id) AS total_orders
FROM online_sales
WHERE YEAR(order_date) = 2024
GROUP BY order_year, order_month
ORDER BY order_month;
```



order_year	order_month	total_revenue	total_orders

4. **Top 5 Months by Total Revenue (All Years)**

```
SELECT
   YEAR(order_date) AS order_year,
   MONTH(order_date) AS order_month,
   SUM(amount) AS total_revenue
FROM online_sales
GROUP BY order_year, order_month
ORDER BY total_revenue DESC
LIMIT 5;
```

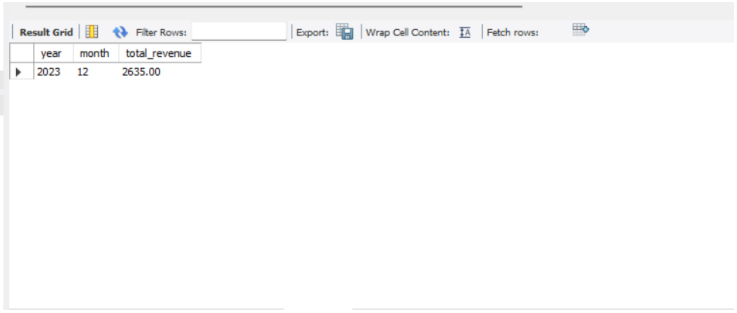


Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

order_year	order_month	total_revenue
2023	12	2635.00
2023	11	2295.00
2023	10	2240.00
2023	7	2235.00
2023	8	2015.00

5. **Highest Revenue Month (All Years)**

```
SELECT
   YEAR(order_date) AS year,
   MONTH(order_date) AS month,
   SUM(amount) AS total_revenue
FROM online_sales
GROUP BY year, month
ORDER BY total_revenue DESC
LIMIT 1;
```

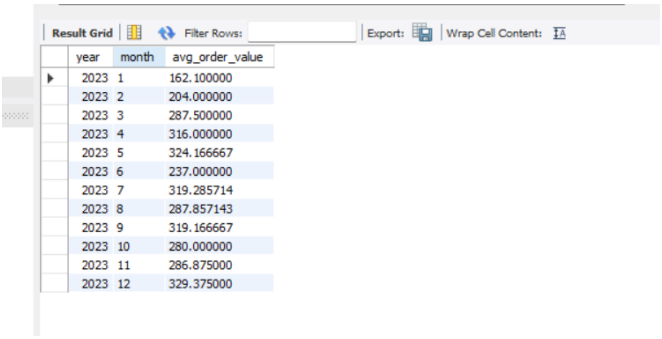


Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

year	month	total_revenue
2023	12	2635.00

6. Average Order Value Per Month

```
SELECT
   YEAR(order_date) AS year,
   MONTH(order_date) AS month,
   SUM(amount) / COUNT(DISTINCT order_id) AS avg_order_value
FROM online_sales
GROUP BY year, month
ORDER BY year, month;
```

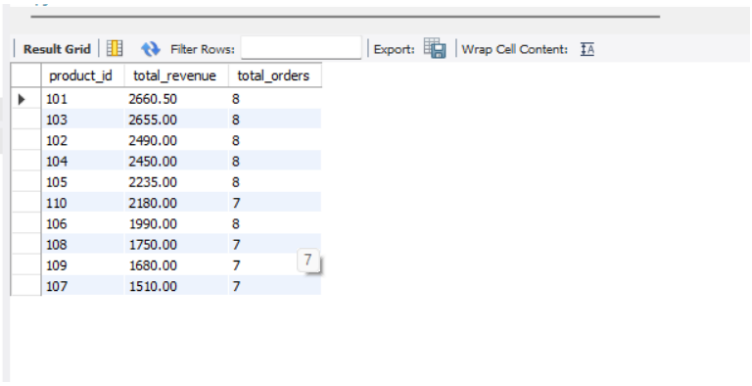


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

year	month	avg_order_value
2023	1	162.100000
2023	2	204.000000
2023	3	287.500000
2023	4	316.000000
2023	5	324.166667
2023	6	237.000000
2023	7	319.285714
2023	8	287.857143
2023	9	319.166667
2023	10	280.000000
2023	11	286.875000
2023	12	329.375000

7. Total Revenue and Orders by Product

```
SELECT
   product_id,
   SUM(amount) AS total_revenue,
   COUNT(order_id) AS total_orders
FROM online_sales
GROUP BY product_id
ORDER BY total_revenue DESC;
```



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

product_id	total_revenue	total_orders
101	2660.50	8
103	2655.00	8
102	2490.00	8
104	2450.00	8
105	2235.00	8
110	2180.00	7
106	1990.00	8
108	1750.00	7
109	1680.00	7
107	1510.00	7

8. Month-on-Month Revenue Growth Percentage

```
SELECT
   year,
   month,
   total_revenue,
   LAG(total_revenue) OVER (ORDER BY year, month) AS prev_month_revenue,
   ROUND(((total_revenue - LAG(total_revenue) OVER (ORDER BY year, month))
        / LAG(total_revenue) OVER (ORDER BY year, month)) * 100, 2) AS growth_percentage
FROM (
   SELECT
      YEAR(order_date) AS year,
      MONTH(order_date) AS month,
      SUM(amount) AS total_revenue
   FROM online_sales
   GROUP BY year, month
) AS monthly_data;
```

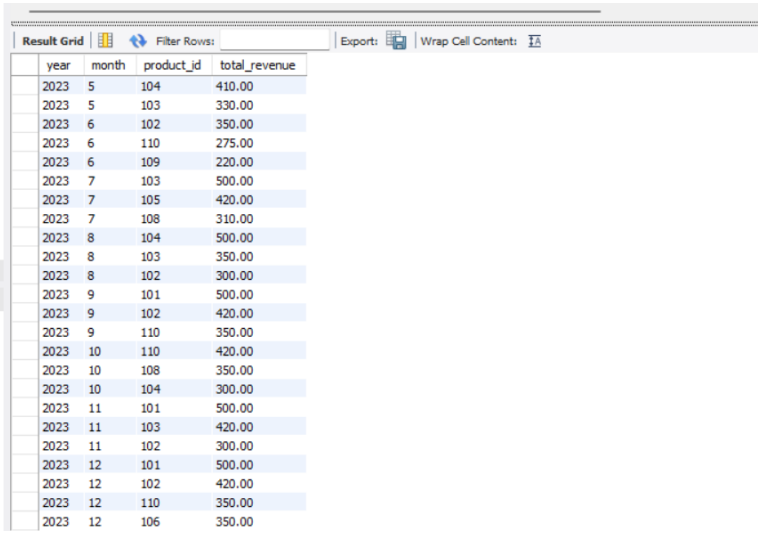
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

year	month	total_revenue	prev_month_revenue	growth_percentage
2023	1	810.50	NULL	NULL
2023	2	1020.00	810.50	25.85
2023	3	1725.00	1020.00	69.12
2023	4	1580.00	1725.00	-8.41
2023	5	1945.00	1580.00	23.10
2023	6	1185.00	1945.00	-39.07
2023	7	2235.00	1185.00	88.61
2023	8	2015.00	2235.00	-9.84
2023	9	1915.00	2015.00	-4.96
2023	10	2240.00	1915.00	16.97
2023	11	2295.00	2240.00	2.46
2023	12	2635.00	2295.00	14.81

9. Top 3 Products by Revenue Each Month

```
SELECT year, month, product_id, total_revenue
FROM (
   SELECT
      YEAR(order_date) AS year,
      MONTH(order_date) AS month,
```

```
    product_id,
    SUM(amount) AS total_revenue,
    RANK() OVER (PARTITION BY YEAR(order_date), MONTH(order_date) ORDER BY SUM(amount) DESC) AS rank_in_month
  FROM online_sales
  GROUP BY year, month, product_id
) ranked
WHERE rank_in_month <= 3
ORDER BY year, month, rank_in_month;
```



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

year	month	product_id	total_revenue
2023	5	104	410.00
2023	5	103	330.00
2023	6	102	350.00
2023	6	110	275.00
2023	6	109	220.00
2023	7	103	500.00
2023	7	105	420.00
2023	7	108	310.00
2023	8	104	500.00
2023	8	103	350.00
2023	8	102	300.00
2023	9	101	500.00
2023	9	102	420.00
2023	9	110	350.00
2023	10	110	420.00
2023	10	108	350.00
2023	10	104	300.00
2023	11	101	500.00
2023	11	103	420.00
2023	11	102	300.00
2023	12	101	500.00
2023	12	102	420.00
2023	12	110	350.00
2023	12	106	350.00