Data Analytics Internship

Day6 - Analyze monthly revenue and order volume

Dataset Used: Online Sales CSV

1. Initial Setup / Validation Queries

1. Loading csv file to database and creating table

sql:

```
Run | Reset
CREATE DATABASE IF NOT EXISTS sales_db;
Run
USE sales_db;

Run | Select
CREATE TABLE IF NOT EXISTS orders (
  order_id INT,
  order_date DATE,
  amount DECIMAL(10,2),
  product_id INT
);


Run | Select
LOAD DATA INFILE 'A:\\V5C0D3\\Internship_Tasks\\day6-task\\OnlineSales.csv'
INTO TABLE orders
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;

Run
SET GLOBAL local_infile = 1;
```

2. Monthly Revenue & Order Volume

sql:

```
▷Run | ┼Tab | JSON
SELECT
  EXTRACT(YEAR FROM order_date) AS order_year,
  EXTRACT(MONTH FROM order_date) AS order_month,
  SUM(amount) AS total_revenue,
  COUNT(DISTINCT order_id) AS total_orders
FROM orders
GROUP BY order_year, order_month
ORDER BY order_year, order_month;
```

output:

Result(RO) ×

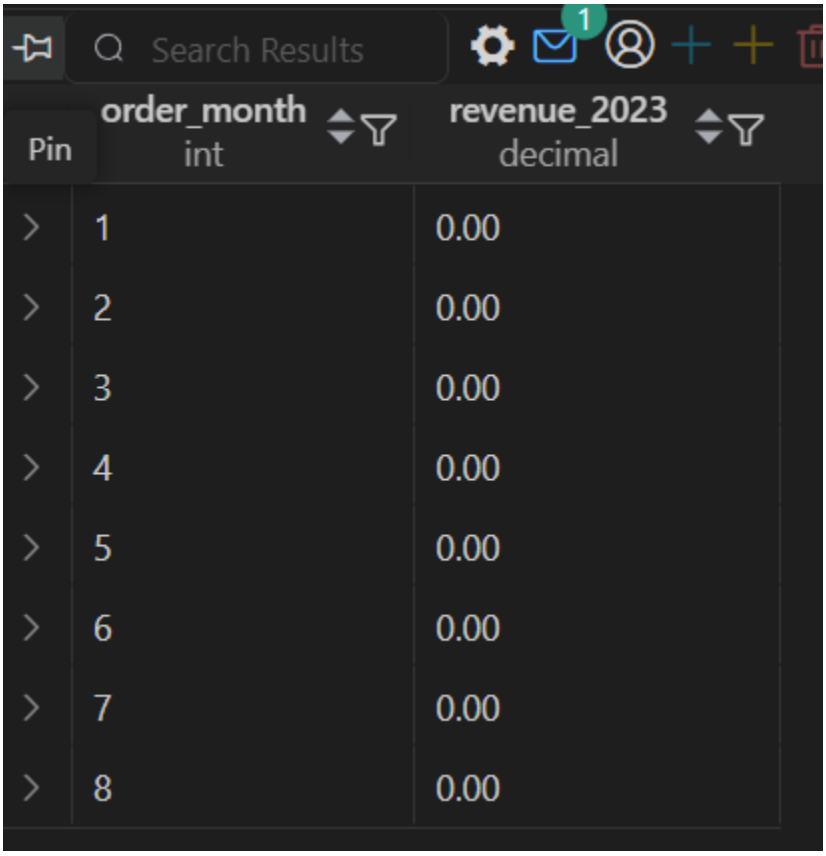
Search Results | Export | Cost: 6ms | 1 | Total 8

order_year int	order_month int	total_revenue decimal	total_orders bigint
2024	1	0.00	31
2024	2	0.00	29
2024	3	0.00	31
2024	4	0.00	30
2024	5	0.00	31
2024	6	0.00	30
2024	7	0.00	31
2024	8	0.00	27

Revenue for a Specific Year *:*

```
▷Run | +Tab | JSON
SELECT
    EXTRACT(MONTH FROM order_date) AS order_month,
    SUM(amount) AS revenue_2023
FROM orders
GROUP BY order_month
ORDER BY order_month;
```

output:



3. Limit to Last 6 Months (Assuming Today's Date = CURDATE())

sql:

```
▷Run | +Tab | JSON
40  SELECT
41      DATE_FORMAT(order_date, '%Y-%m') AS month,
42      SUM(amount) AS total_revenue,
43      COUNT(DISTINCT order_id) AS total_orders
44  FROM orders
45  WHERE order_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
46  GROUP BY month
47  ORDER BY month;
```

output:

Result(RO)

Search Results | Export | Cost:

month varchar	revenue decimal	prev_month_revenue decimal	percent_growth decimal
2024-02	0.00	0.00	(NULL)
2024-03	0.00	0.00	(NULL)
2024-04	0.00	0.00	(NULL)
2024-05	0.00	0.00	(NULL)
2024-06	0.00	0.00	(NULL)
2024-07	0.00	0.00	(NULL)
2024-08	0.00	0.00	(NULL)

4. Monthly Growth in Revenue (Month-over-Month %)

sql:

```
▷Run | +Tab | JSON
WITH monthly_revenue AS (
  SELECT
    DATE_FORMAT(order_date, '%Y-%m') AS month,
    SUM(amount) AS revenue
  FROM orders
  GROUP BY month
),
growth AS (
  SELECT
    month,
    revenue,
    LAG(revenue) OVER (ORDER BY month) AS prev_month_revenue
  FROM monthly_revenue
)
SELECT
  month,
  revenue,
  prev_month_revenue,
  ROUND(((revenue - prev_month_revenue)/prev_month_revenue) * 100, 2) AS percent_growth
FROM growth
WHERE prev_month_revenue IS NOT NULL;
```

output:

Result(RO)

Search Results | Export | Cost: 10ms | 1 | Total 7

month varchar	revenue decimal	prev_month_revenue decimal	percent_growth decimal
2024-02	0.00	0.00	(NULL)
2024-03	0.00	0.00	(NULL)
2024-04	0.00	0.00	(NULL)
2024-05	0.00	0.00	(NULL)
2024-06	0.00	0.00	(NULL)
2024-07	0.00	0.00	(NULL)
2024-08	0.00	0.00	(NULL)

5. Best Month by Revenue

sql:

```
▷ Run | + Tab | JSON
SELECT
    DATE_FORMAT(order_date, '%Y-%m') AS month,
    SUM(amount) AS total_revenue
FROM orders
GROUP BY month
ORDER BY total_revenue DESC
LIMIT 1;
```

output:

Result(RO)

Search Results | Export | Cost: 10ms | 1 | Total 7

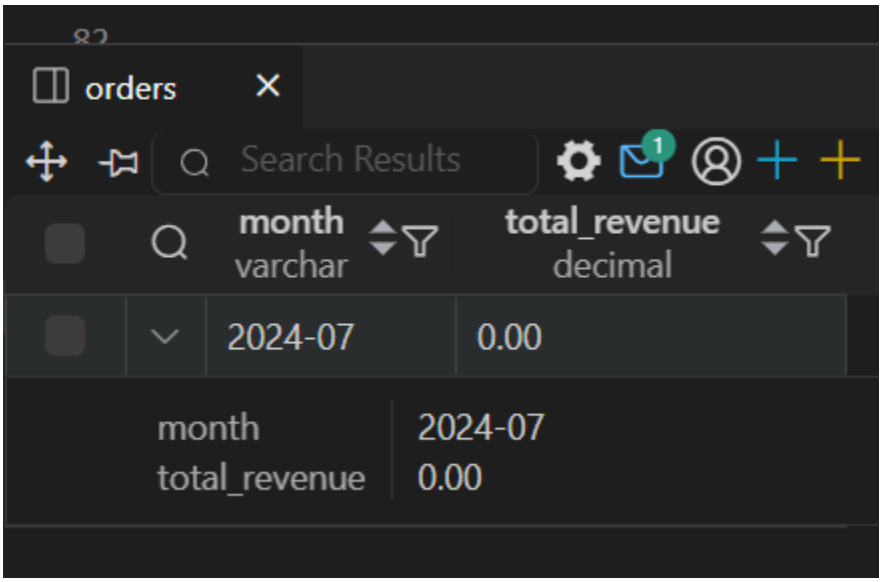
month varchar	revenue decimal	prev_month_revenue decimal	percent_growth decimal
2024-02	0.00	0.00	(NULL)
2024-03	0.00	0.00	(NULL)
2024-04	0.00	0.00	(NULL)
2024-05	0.00	0.00	(NULL)
2024-06	0.00	0.00	(NULL)
2024-07	0.00	0.00	(NULL)
2024-08	0.00	0.00	(NULL)

6. Total Revenue and Volume by Product

sql:

```
▷Run | +Tab | JSON
WITH monthly_avg AS (
  SELECT
    EXTRACT(YEAR_MONTH FROM order_date) AS ym,
    AVG(amount) AS avg_amt
  FROM orders
  GROUP BY ym
)
SELECT
  o.*
FROM orders o
JOIN monthly_avg m
  ON EXTRACT(YEAR_MONTH FROM o.order_date) = m.ym
WHERE o.amount > m.avg_amt;
```

output:

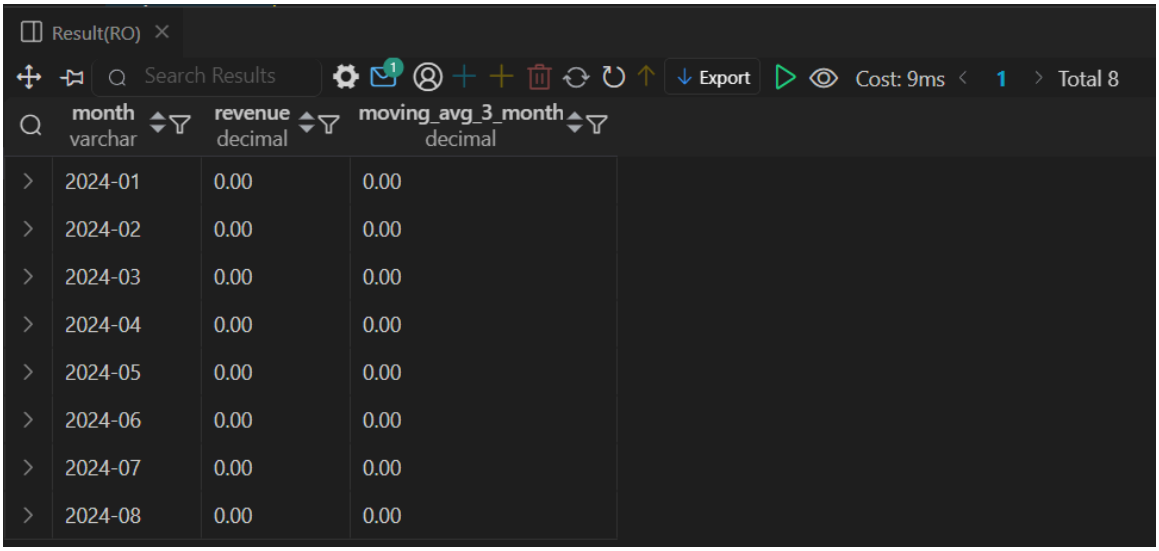


7. Month with Highest Average Order Value (AOV)

sql:

```
▷Run | +Tab | JSON
WITH monthly AS (
  SELECT
    DATE_FORMAT(order_date, '%Y-%m') AS month,
    SUM(amount) AS revenue
  FROM orders
  GROUP BY month
),
moving_avg AS (
  SELECT
    month,
    revenue,
    ROUND(AVG(revenue) OVER (ORDER BY month ROWS BETWEEN 2 PRECEDING AND CURRENT ROW), 2) AS moving_avg_3_month
  FROM monthly
)
SELECT * FROM moving_avg;
```

output:



month varchar	revenue decimal	moving_avg_3_month decimal
2024-01	0.00	0.00
2024-02	0.00	0.00
2024-03	0.00	0.00
2024-04	0.00	0.00
2024-05	0.00	0.00
2024-06	0.00	0.00
2024-07	0.00	0.00
2024-08	0.00	0.00