

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
2  CREATE DATABASE IF NOT EXISTS sales_db;
  ▷ Run
3  USE sales_db;
4
  ▷ Run | 📄 Select
5  CREATE TABLE IF NOT EXISTS orders (
6    order_id INT,
7    order_date DATE,
8    amount DECIMAL(10,2),
9    product_id INT
10 );
11
12
  ▷ Run | 📄 Select
13 LOAD DATA INFILE 'A:\\V5C0D3\\Internship_Tasks\\day6-task\\OnlineSales.csv'
14 INTO TABLE orders
15 FIELDS TERMINATED BY ','
16 ENCLOSED BY '"'
17 LINES TERMINATED BY '\n'
18 IGNORE 1 ROWS;
19
  ▷ Run
20 SET GLOBAL local_infile = 1;
```

2. Monthly Revenue & Order Volume

sql:

```

  Run | +Tab | JSON
22  SELECT
23      EXTRACT(YEAR FROM order_date) AS order_year,
24      EXTRACT(MONTH FROM order_date) AS order_month,
25      SUM(amount) AS total_revenue,
26      COUNT(DISTINCT order_id) AS total_orders
27  FROM orders
28  GROUP BY order_year, order_month
29  ORDER BY order_year, order_month;
30

```

output:

Result(RO) x

Search Results

Export

Cost: 6ms < 1 > Total 8

order_year	order_month	total_revenue	total_orders
int	int	decimal	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
32 SELECT
33     EXTRACT(MONTH FROM order_date) AS order_month,
34     SUM(amount) AS revenue_2023
35 FROM orders
36 GROUP BY order_month
37 ORDER BY order_month;
```

output:

Search Results		
Pin	order_month int	revenue_2023 decimal
>	1	0.00
>	2	0.00
>	3	0.00
>	4	0.00
>	5	0.00
>	6	0.00
>	7	0.00
>	8	0.00

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;
48
```

output:

Result(RO) X				
Search Results				
	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
50 WITH monthly_revenue AS (
51     SELECT
52         DATE_FORMAT(order_date, '%Y-%m') AS month,
53         SUM(amount) AS revenue
54     FROM orders
55     GROUP BY month
56 ),
57 growth AS (
58     SELECT
59         month,
60         revenue,
61         LAG(revenue) OVER (ORDER BY month) AS prev_month_revenue
62     FROM monthly_revenue
63 )
64 SELECT
65     month,
66     revenue,
67     prev_month_revenue,
68     ROUND(((revenue - prev_month_revenue)/prev_month_revenue) * 100, 2) AS percent_growth
69 FROM growth
70 WHERE prev_month_revenue IS NOT NULL;
```

output:

Result(RO) X				
Search Results				
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:

```
71  > Run | + Tab | JSON
72  SELECT
73  |   DATE_FORMAT(order_date, '%Y-%m') AS month,
74  |   SUM(amount) AS total_revenue
75  FROM orders
76  GROUP BY month
77  ORDER BY total_revenue DESC
78  LIMIT 1;
79
```

output:

Result(RO) X				
Search Results				
Cost: 10ms < 1 > Total 7				
month	revenue	prev_month_revenue	percent_growth	
varchar	decimal	decimal	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:

```

80
81  ▸ Run | +Tab | JSON
81  ▾ WITH monthly_avg AS (
82  ▾   SELECT
83      EXTRACT(YEAR_MONTH FROM order_date) AS ym,
84      AVG(amount) AS avg_amt
85  FROM orders
86  GROUP BY ym
87  )
88  ▾ SELECT
89      o.*
90  FROM orders o
91  ▾ JOIN monthly_avg m
92      ON EXTRACT(YEAR_MONTH FROM o.order_date) = m.ym
93  WHERE o.amount > m.avg_amt;
94

```

output:

82			
orders			
<div> <div> <div></div> <div></div> </div> <div> <div>Q</div> <div>Search Results</div> </div> <div> <div>⚙️</div> <div>📧¹</div> <div>👤</div> <div>+</div> <div>+</div> </div> </div>			
	Q	month varchar	total_revenue decimal
	▼	2024-07	0.00
month		2024-07	
total_revenue		0.00	

7. Month with Highest Average Order Value (AOV)

sql:

```

82 WITH monthly AS (
83     SELECT
84         DATE_FORMAT(order_date, '%Y-%m') AS month,
85         SUM(amount) AS revenue
86     FROM orders
87     GROUP BY month
88 ),
89 moving_avg AS (
90     SELECT
91         month,
92         revenue,
93         ROUND(AVG(revenue) OVER (ORDER BY month ROWS BETWEEN 2 PRECEDING AND CURRENT ROW), 2) AS moving_avg_3_month
94     FROM monthly
95 )
96 SELECT * FROM moving_avg;

```

output:

Result(RO) X

Search Results

Cost: 9ms < 1 > Total 8

	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