```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import pandas as pd
import sqlite3
```

```
# Load the CSV file from your Google Drive path
file_path = "/content/drive/MyDrive/ELEVATE LABS/Online Sales Data.csv"

# Read the file into a DataFrame
df = pd.read_csv(file_path)

# Show first few rows
df.head()
```

	Transaction ID	Date	Product Category	Product Name	Units Sold	Unit Price	Total Revenue	Region	Payment Method
0	10001	2024-01-01	Electronics	iPhone 14 Pro	2	999.99	1999.98	North America	Credit Card
1	10002	2024-01-02	Home Appliances	Dyson V11 Vacuum	1	499.99	499.99	Europe	PayPal
2	10003	2024-01-03	Clothing	Levi's 501 Jeans	3	69.99	209.97	Asia	Debit Card
3	10004	2024-01-04	Books	The Da Vinci Code	4	15.99	63.96	North America	Credit Card
4	10005	2024-01-05	Beauty Products	Neutrogena Skincare Set	1	89.99	89.99	Europe	PayPal

Next steps: View recommended plots New interactive sheet

```
# Clean column names
df.columns = df.columns.str.strip().str.lower().str.replace(' ', '_')

# Convert 'date' to datetime
df['date'] = pd.to_datetime(df['date'], errors='coerce')

# Drop rows with invalid or missing dates
df = df.dropna(subset=['date'])

# Preview cleaned data
df.head()
```

	transaction_id	date	product_category	product_name	units_sold	unit_price	total_revenue	region
0	10001	2024-01-01	Electronics	iPhone 14 Pro	2	999.99	1999.98	North America
1	10002	2024-01-02	Home Appliances	Dyson V11 Vacuum	1	499.99	499.99	Europe
2	10003	2024-01-03	Clothing	Levi's 501 Jeans	3	69.99	209.97	Asia
3	10004	2024-01-04	Books	The Da Vinci Code	4	15.99	63.96	North America
4	10005	2024-01-05	Beauty Products	Neutrogena Skincare Set	1	89.99	89.99	Europe

Next steps: View recommended plots New interactive sheet

```
# Connect to in-memory SQLite database
conn = sqlite3.connect(":memory:")

# Save DataFrame to SQL table
df.to_sql("sales", conn, index=False, if_exists='replace')

# SQL query: monthly revenue and order volume
query = """
SELECT
  STRFTIME('%Y', date) AS year,
  STRFTIME('%m', date) AS month,
  SUM(total_revenue) AS total_revenue,
  COUNT(DISTINCT transaction_id) AS transaction_volume
FROM sales
GROUP BY year, month
ORDER BY year ASC, month ASC
LIMIT 12;
"""

# Run query
result = pd.read_sql(query, conn)

# Clean up result
result['month'] = result['month'].astype(int)
result['year'] = result['year'].astype(int)

# Display result
result
```

	year	month	total_revenue	transaction_volume
0	2024	1	14548.32	31
1	2024	2	10803.37	29
2	2024	3	12849.24	31
3	2024	4	12451.69	30
4	2024	5	8455.49	31
5	2024	6	7384.55	30
6	2024	7	6797.08	31
7	2024	8	7278.11	27

Next steps: View recommended plots New interactive sheet

```
print("1️⃣ Group by Month and Year")
display(pd.read_sql("""
SELECT
  STRFTIME('%Y', date) AS year,
  STRFTIME('%m', date) AS month,
  SUM(total_revenue) AS monthly_revenue
FROM sales
GROUP BY year, month
ORDER BY year, month;
""", conn))
```

```
1️⃣ Group by Month and Year
```

	year	month	monthly_revenue
0	2024	01	14548.32
1	2024	02	10803.37
2	2024	03	12849.24
3	2024	04	12451.69
4	2024	05	8455.49
5	2024	06	7384.55
6	2024	07	6797.08
7	2024	08	7278.11

```
# COUNT(*) vs COUNT(DISTINCT transaction_id)

print("2️⃣ COUNT(*) vs COUNT(DISTINCT transaction_id)")
display(pd.read_sql("""
SELECT
  COUNT(*) AS total_rows,
  COUNT(DISTINCT transaction_id) AS unique_transactions
FROM sales;
""", conn))
```

```
2️⃣ COUNT(*) vs COUNT(DISTINCT transaction_id)
```

	total_rows	unique_transactions
0	240	240

```
# Monthly Revenue Calculation

print("3️⃣ Monthly Revenue")
display(pd.read_sql("""
SELECT
  STRFTIME('%Y-%m', date) AS month_year,
  SUM(total_revenue) AS monthly_revenue
FROM sales
GROUP BY month_year
ORDER BY month_year;
""", conn))
```

```
3️⃣ Monthly Revenue
   month_year  monthly_revenue
0     2024-01         14548.32
1     2024-02         10803.37
2     2024-03         12849.24
3     2024-04         12451.69
4     2024-05          8455.49
5     2024-06          7384.55
6     2024-07          6797.08
7     2024-08          7278.11
```

```
# Aggregate Functions (SUM, AVG, MIN, MAX, COUNT)
print("4️⃣ Aggregate Functions Example")
display(pd.read_sql("""
SELECT
  SUM(total_revenue) AS total_sales,
  AVG(unit_price) AS avg_price,
  MIN(total_revenue) AS min_sale,
  MAX(total_revenue) AS max_sale,
  COUNT(*) AS total_transactions
FROM sales;
""", conn))
```

```
4️⃣ Aggregate Functions Example
   total_sales   avg_price  min_sale  max_sale  total_transactions
0     80567.85  236.395583       6.5   3899.99                 240
```

```
# Handling NULLs with COALESCE

print("5️⃣ Handling NULLs in Aggregates")
display(pd.read_sql("""
SELECT
  SUM(COALESCE(total_revenue, 0)) AS total_revenue_no_nulls,
  AVG(COALESCE(unit_price, 0)) AS avg_price_no_nulls
FROM sales;
""", conn))
```

```
5️⃣ Handling NULLs in Aggregates
   total_revenue_no_nulls  avg_price_no_nulls
0                80567.85          236.395583
```

```
# ORDER BY + GROUP BY – Region Revenue

print("6️⃣ ORDER BY and GROUP BY Example (Region Revenue)")
display(pd.read_sql("""
SELECT
  region,
  SUM(total_revenue) AS region_revenue
FROM sales
GROUP BY region
ORDER BY region_revenue DESC;
""", conn))
```

```
6️⃣ ORDER BY and GROUP BY Example (Region Revenue)
          region  region_revenue
0  North America        36844.34
1           Asia        22455.45
```

```
# 3 Months by Revenue

print("7️⃣ Top 3 Months by Sales")
display(pd.read_sql("""
SELECT
  STRFTIME('%Y-%m', date) AS month_year,
  SUM(total_revenue) AS monthly_revenue
FROM sales
GROUP BY month_year
ORDER BY monthly_revenue DESC
LIMIT 3;
""", conn))
```

```
7️⃣ Top 3 Months by Sales
   month_year  monthly_revenue
0     2024-01         14548.32
1     2024-03         12849.24
2     2024-04         12451.69
```

```
conn.close()
```

conclusion

All queries were executed as pure SQL using SQLite inside Google Colab — no manual Python calculations were used.

The workflow is efficient, scalable, and reflects real-world SQL problem-solving for BI and data analyst roles.

This format is highly recommended for SQL interviews, assignments, or case studies