```
Task 6 – Sales Trend Analysis Using Aggregations
Ready-to-run package that generates:

- Synthetic orders dataset (2022–2024)
- SQLite database (online_sales.db)
- SQL script
- Aggregation outputs (monthly revenue, 2023 revenue, top 3 months)
- High-resolution charts
"""

import os, shutil, sqlite3, zipfile, textwrap
from datetime import datetime
from pathlib import Path
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt


# 0. SETUP OUTPUT FOLDER
OUT = Path("task6-analysis")
if OUT.exists():
    shutil.rmtree(OUT)
OUT.mkdir(parents=True)
IMG = OUT / "images"
IMG.mkdir()
```

```
# 1. GENERATE SYNTHETIC ORDERS DATASET
np.random.seed(2025)


def random_dates(start, end):
    return pd.date_range(start, end, freq="D")


start_date = datetime(2022,1,1)
end_date   = datetime(2024,12,31)


dates = random_dates(start_date, end_date)


orders = []
order_id = 100000


for d in dates:
    daily_count = np.random.poisson(60)


    for _ in range(daily_count):
        amount = round(np.random.gamma(5, 20) + np.random.normal(0, 10), 2)
        if amount < 5:
            amount = round(np.random.uniform(5, 40), 2)


        product_id = np.random.randint(1,101)
        ts = d.strftime("%Y-%m-%d") + f"
{np.random.randint(0,23)}:{np.random.randint(0,59)}:00"


        orders.append((order_id, ts, amount, product_id))
        order_id += 1
```

```
df_orders = pd.DataFrame(orders,
columns=["order_id","order_date","amount","product_id"])

df_orders.to_csv(OUT/"online_sales_orders.csv", index=False)



# 2. CREATE SQLITE DATABASE
db = OUT/"online_sales.db"
conn = sqlite3.connect(db)
cur = conn.cursor()

cur.execute("""
CREATE TABLE orders (
    order_id INTEGER PRIMARY KEY,
    order_date TEXT,
    amount REAL,
    product_id INTEGER
)
""")

df_orders.to_sql("orders", conn, if_exists="append", index=False)



# 3. SQL SCRIPT (SAVED AS .sql)
sql_script = """
-- Task 6 – Sales Trend Analysis Using Aggregations

-- 1. Monthly revenue + monthly order count
SELECT
    STRFTIME('%Y', order_date) AS year,
    STRFTIME('%m', order_date) AS month,
    ROUND(SUM(amount),2) AS monthly_revenue,
    COUNT(order_id) AS monthly_orders
FROM orders
GROUP BY year, month
ORDER BY year, month;

-- 2. Breakdown for 2023 only
SELECT
    STRFTIME('%m', order_date) AS month,
    ROUND(SUM(amount),2) AS monthly_revenue,
    COUNT(order_id) AS order_count
FROM orders
WHERE STRFTIME('%Y', order_date) = '2023'
GROUP BY month
ORDER BY month;

-- 3. Top 3 months by revenue
SELECT
    STRFTIME('%Y-%m', order_date) AS year_month,
    ROUND(SUM(amount),2) AS revenue
FROM orders
GROUP BY year_month
ORDER BY revenue DESC
LIMIT 3;
"""

with open(OUT/"task6_queries.sql", "w") as f:
    f.write(sql_script)
```

```
# 4. RUN QUERIES AND EXPORT CSVs
# 4.1 monthly revenue
monthly = pd.read_sql_query("""
SELECT
    STRFTIME('%Y', order_date) AS year,
    STRFTIME('%m', order_date) AS month,
    ROUND(SUM(amount),2) AS revenue,
    COUNT(order_id) AS order_count
FROM orders
GROUP BY year, month
ORDER BY year, month
""", conn)

monthly.to_csv(OUT/"monthly_revenue_orders.csv", index=False)

# 4.2 2023 revenue
monthly23 = pd.read_sql_query("""
SELECT
    STRFTIME('%m', order_date) AS month,
    ROUND(SUM(amount),2) AS revenue,
    COUNT(order_id) AS order_count
FROM orders
WHERE STRFTIME('%Y', order_date)='2023'
GROUP BY month
ORDER BY month
""", conn)
```

```
monthly23.to_csv(OUT/"monthly_2023.csv", index=False)


# 4.3 top 3 months
top3 = pd.read_sql_query("""
SELECT
   STRFTIME('%Y-%m', order_date) AS month,
   ROUND(SUM(amount),2) AS revenue
FROM orders
GROUP BY month
ORDER BY revenue DESC
LIMIT 3
""", conn)


top3.to_csv(OUT/"top3_months.csv", index=False)



# 5. CHARTS
plt.style.use("seaborn-v0_8-darkgrid")


# build x-axis
x = pd.to_datetime(monthly["year"] + "-" + monthly["month"] + "-01")


# revenue time series
plt.figure(figsize=(12,5))
plt.plot(x, monthly["revenue"], marker="o")
plt.title("Monthly Revenue (2022–2024)")
plt.xlabel("Month"); plt.ylabel("Revenue")
plt.tight_layout()
plt.savefig(IMG/"monthly_revenue.png", dpi=200)
plt.close()

# orders time series
plt.figure(figsize=(12,5))
plt.plot(x, monthly["order_count"], marker="o", color="orange")
plt.title("Monthly Order Volume")
plt.xlabel("Month"); plt.ylabel("Orders")
plt.tight_layout()
plt.savefig(IMG/"monthly_orders.png", dpi=200)
plt.close()

# bar chart for 2023
plt.figure(figsize=(10,5))
plt.bar(monthly23["month"].astype(int), monthly23["revenue"], color="green")
plt.title("Monthly Revenue – 2023")
plt.xlabel("Month"); plt.ylabel("Revenue")
plt.tight_layout()
plt.savefig(IMG/"monthly_revenue_2023.png", dpi=200)
plt.close()

# top 3 pie chart
plt.figure(figsize=(6,6))
plt.pie(top3["revenue"], labels=top3["month"], autopct="%1.1f%%")
plt.title("Top 3 Months by Revenue")
plt.tight_layout()
plt.savefig(IMG/"top3_pie.png", dpi=200)
plt.close()
```