Task 7

```
#!/usr/bin/env python3

"""

build_task7_package.py

Creates a full, GitHub-ready deliverable for Task 7:

- SQLite DB (sales_data.db) with a 'sales' table (synthetic demo data)

- SQL query file (task7_queries.sql) — no interview questions

- Aggregated CSV exports

- README.md


Run:

  pip install pandas matplotlib reportlab

  python build_task7_package.py

"""


import os, sqlite3, zipfile, shutil, textwrap

from pathlib import Path

from datetime import datetime, timedelta

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from reportlab.lib.pagesizes import A4

from reportlab.platypus import SimpleDocTemplate, Paragraph, Spacer, Image, Table, TableStyle, PageBreak

from reportlab.lib.styles import getSampleStyleSheet, ParagraphStyle

from reportlab.pdfbase.cidfonts import UnicodeCIDFont

from reportlab.pdfbase import pdfmetrics

from reportlab.lib import colors
```

```
# 0. Setup
OUT = Path("task7-sales-summary")
if OUT.exists():
    shutil.rmtree(OUT)
OUT.mkdir(parents=True)
IMGDIR = OUT / "images"
IMGDIR.mkdir()

# 1. Create a tiny SQLite DB (sales_data.db) with a sample 'sales' table
DB_PATH = OUT / "sales_data.db"
conn = sqlite3.connect(str(DB_PATH))
cur = conn.cursor()

cur.executescript("""
DROP TABLE IF EXISTS sales;
CREATE TABLE sales(
    sale_id INTEGER PRIMARY KEY,
    sale_date TEXT,
    product TEXT,
    quantity INTEGER,
    price REAL,
    region TEXT
);
""")
conn.commit()

# Populate sample data (you can replace this by loading your own CSV)
```

```
np.random.seed(42)
products = ["Widget A","Widget B","Widget C","Gadget X","Gadget Y"]
regions = ["North","South","East","West"]
start = datetime(2023,1,1)
rows = []
sale_id = 1
for day_offset in range(0, 180):  # six months of synthetic data
  date = (start + timedelta(days=day_offset)).strftime("%Y-%m-%d")
  for _ in range(np.random.randint(5, 12)):  # 5-11 sales per day
    prod = np.random.choice(products, p=[0.2,0.2,0.2,0.2,0.2])
    qty = int(np.random.choice([1,1,1,2,3], p=[0.45,0.25,0.15,0.1,0.05]))
    price = round(float(np.random.uniform(50, 500)), 2)
    region = np.random.choice(regions)
    rows.append((sale_id, date, prod, qty, price, region))
    sale_id += 1

# Insert into DB in one shot via pandas
df_sales = pd.DataFrame(rows,
columns=["sale_id","sale_date","product","quantity","price","region"])
df_sales.to_sql("sales", conn, if_exists="append", index=False)

# 2. SQL queries file (no interview questions)
sql_text = textwrap.dedent("""
-- Task 7: Basic Sales Summary (deliverables only)

-- 1) Total quantity and revenue by product
SELECT product,
    SUM(quantity) AS total_quantity,
    ROUND(SUM(quantity * price),2) AS total_revenue
FROM sales
GROUP BY product
ORDER BY total_revenue DESC;

-- 2) Total quantity and revenue by region
SELECT region,
    SUM(quantity) AS total_quantity,
    ROUND(SUM(quantity * price),2) AS total_revenue
FROM sales
GROUP BY region
ORDER BY total_revenue DESC;

-- 3) Daily revenue (for time-series chart)
SELECT sale_date,
    ROUND(SUM(quantity * price),2) AS daily_revenue
FROM sales
GROUP BY sale_date
ORDER BY sale_date;
""")
with open(OUT/"task7_queries.sql", "w") as f:
  f.write(sql_text.strip())

# 3. Run the queries from Python and save outputs
# Query 1: by product
q1 = "SELECT product, SUM(quantity) AS total_quantity, ROUND(SUM(quantity * price),2) AS total_revenue FROM sales GROUP BY product ORDER BY total_revenue DESC;"
df_by_product = pd.read_sql_query(q1, conn)
```

```
df_by_product.to_csv(OUT/"sales_by_product.csv", index=False)


# Query 2: by region
q2 = "SELECT region, SUM(quantity) AS total_quantity, ROUND(SUM(quantity * price),2) AS total_revenue FROM sales GROUP BY region ORDER BY total_revenue DESC;"
df_by_region = pd.read_sql_query(q2, conn)
df_by_region.to_csv(OUT/"sales_by_region.csv", index=False)


# Query 3: daily revenue
q3 = "SELECT sale_date, ROUND(SUM(quantity * price),2) AS daily_revenue FROM sales GROUP BY sale_date ORDER BY sale_date;"
df_daily = pd.read_sql_query(q3, conn)
df_daily.to_csv(OUT/"daily_revenue.csv", index=False)


# 4. Create visuals (PNG)
plt.style.use("seaborn-darkgrid")


# Bar: revenue by product
plt.figure(figsize=(8,5))
plt.bar(df_by_product['product'], df_by_product['total_revenue'])
plt.title("Total Revenue by Product")
plt.xlabel("Product")
plt.ylabel("Revenue")
plt.tight_layout()
prod_img = IMGDIR / "revenue_by_product.png"
plt.savefig(prod_img, dpi=200); plt.close()


# Bar: revenue by region
plt.figure(figsize=(6,4))
plt.bar(df_by_region['region'], df_by_region['total_revenue'], color='tab:orange')

plt.title("Total Revenue by Region")

plt.xlabel("Region")

plt.ylabel("Revenue")

plt.tight_layout()

reg_img = IMGDIR / "revenue_by_region.png"

plt.savefig(reg_img, dpi=200); plt.close()


# Line: daily revenue

plt.figure(figsize=(12,4))

plt.plot(pd.to_datetime(df_daily['sale_date']), df_daily['daily_revenue'], marker='o',
linewidth=1)

plt.title("Daily Revenue (time series)")

plt.xlabel("Date")

plt.ylabel("Revenue")

plt.tight_layout()

daily_img = IMGDIR / "daily_revenue_timeseries.png"

plt.savefig(daily_img, dpi=200); plt.close()


# KPI banner (simple)

total_qty = int(df_sales['quantity'].sum())

total_rev = round((df_sales['quantity'] * df_sales['price']).sum(), 2)

avg_order_value = round(total_rev / (df_sales.shape[0]), 2)

plt.figure(figsize=(9,1.6)); plt.axis("off")

kpi_text = f"Total quantity sold: {total_qty:,}   |   Total revenue: ${total_rev:,.2f}   |   Avg
order value: ${avg_order_value:,.2f}"

plt.text(0.01, 0.5, kpi_text, fontsize=11)

kpi_img = IMGDIR / "kpi_banner.png"

plt.savefig(kpi_img, bbox_inches='tight', dpi=200); plt.close()
```