1) SQL queries file — task3_queries.sql

```sql
-- Task 3: SQL for Data Analysis (no interview questions)

-- 1) Total revenue by country

SELECT c.country, ROUND(SUM(o.total_amount),2) AS revenue

FROM customers c

JOIN orders o ON c.customer_id = o.customer_id

GROUP BY c.country

ORDER BY revenue DESC;


-- 2) Top selling products (units sold)

SELECT p.product_name, SUM(oi.quantity) AS units_sold

FROM products p

JOIN order_items oi ON p.product_id = oi.product_id

GROUP BY p.product_id

ORDER BY units_sold DESC

LIMIT 10;


-- 3) Monthly revenue (2023)

SELECT strftime('%m', order_date) AS month, ROUND(SUM(total_amount),2) AS
revenue

FROM orders

WHERE strftime('%Y', order_date) = '2023'

GROUP BY month

ORDER BY month;
```

```
-- 4) Average order value (AOV)

SELECT ROUND(AVG(total_amount),2) AS avg_order_value FROM orders;


-- 5) Create a view for customer spending

CREATE VIEW IF NOT EXISTS customer_spend AS

SELECT c.customer_id, c.name, c.country, ROUND(SUM(o.total_amount),2) AS total_spent

FROM customers c

LEFT JOIN orders o ON c.customer_id = o.customer_id

GROUP BY c.customer_id;


-- 6) Query the view (top customers)

SELECT * FROM customer_spend ORDER BY total_spent DESC LIMIT 10;
```

2) Python script that does everything — build_task3_bundle.py

```
build_task3_bundle.py

Creates:

 - ecommerce.db (SQLite) with synthetic data

 - task3_queries.sql (ensure in same dir if you prefer to overwrite)

 - images/ with charts

 - task3_combined_report.pdf (combined PDF)

 - README.md

 - task3-sql-analysis.zip (zip bundle ready to upload)

"""

import os, sqlite3, pandas as pd, numpy as np, matplotlib.pyplot as plt, zipfile, shutil

from datetime import datetime

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

OUTDIR = "task3-sql-analysis"
if os.path.exists(OUTDIR):
    shutil.rmtree(OUTDIR)
os.makedirs(OUTDIR, exist_ok=True)
IMGDIR = os.path.join(OUTDIR, "images"); os.makedirs(IMGDIR, exist_ok=True)

# --- 1) Build SQLite database with synthetic data
db_path = os.path.join(OUTDIR, "ecommerce.db")
conn = sqlite3.connect(db_path)
cur = conn.cursor()

cur.executescript("""
DROP TABLE IF EXISTS customers;
DROP TABLE IF EXISTS products;
DROP TABLE IF EXISTS orders;
DROP TABLE IF EXISTS order_items;
```

```
CREATE TABLE customers(
   customer_id INTEGER PRIMARY KEY,
   name TEXT,
   email TEXT,
   country TEXT
);

CREATE TABLE products(
   product_id INTEGER PRIMARY KEY,
   product_name TEXT,
   category TEXT,
   price REAL
);

CREATE TABLE orders(
   order_id INTEGER PRIMARY KEY,
   customer_id INTEGER,
   order_date TEXT,
   total_amount REAL,
   FOREIGN KEY(customer_id) REFERENCES customers(customer_id)
);

CREATE TABLE order_items(
   item_id INTEGER PRIMARY KEY,
   order_id INTEGER,
   product_id INTEGER,
   quantity INTEGER,
   FOREIGN KEY(order_id) REFERENCES orders(order_id),
  FOREIGN KEY(product_id) REFERENCES products(product_id)
);
""")

np.random.seed(123)
countries = ["USA","India","Canada","Germany","UK"]
categories = ["Electronics","Clothing","Home","Books"]
product_names = [f"Product {i}" for i in range(1,31)]

# customers
for i in range(1,51):
  cur.execute("INSERT INTO customers(customer_id,name,email,country)
VALUES(?,?,?,?)",
          (i, f"Customer {i}", f"user{i}@example.com", np.random.choice(countries)))

# products
for i, pname in enumerate(product_names, start=1):
  cur.execute("INSERT INTO products(product_id,product_name,category,price)
VALUES(?,?,?,?)",
          (i, pname, np.random.choice(categories), round(np.random.uniform(5,500),2)))

# orders and items
order_id = 1
item_id = 1
for cust in range(1,51):
  num_orders = np.random.randint(1,5)
  for _ in range(num_orders):
    month = np.random.randint(1,13)
    day = np.random.randint(1,28)
    order_date = datetime(2023, month, day).strftime("%Y-%m-%d")
    cur.execute("INSERT INTO orders(order_id,customer_id,order_date,total_amount)
VALUES(?,?,?,?)",
            (order_id, cust, order_date, 0.0))
    total = 0.0
    num_items = np.random.randint(1,4)
    for _ in range(num_items):
      pid = np.random.randint(1, len(product_names))
      qty = int(np.random.choice([1,1,2,3], p=[0.5,0.3,0.15,0.05]))
      price = cur.execute("SELECT price FROM products WHERE product_id=?",
(pid,)).fetchone()[0]
      total += price * qty
      cur.execute("INSERT INTO order_items(item_id,order_id,product_id,quantity)
VALUES(?,?,?,?)",
              (item_id, order_id, pid, qty))
      item_id += 1
    cur.execute("UPDATE orders SET total_amount=? WHERE order_id=?",
(round(total,2), order_id))
    order_id += 1

conn.commit()

# --- 2) Save queries file (task3_queries.sql)
sql_text = \"\"\"-- Task 3: SQL for Data Analysis
SELECT c.country, ROUND(SUM(o.total_amount),2) AS revenue
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
GROUP BY c.country
ORDER BY revenue DESC;
```

```
SELECT p.product_name, SUM(oi.quantity) AS units_sold
FROM products p
JOIN order_items oi ON p.product_id = oi.product_id
GROUP BY p.product_id
ORDER BY units_sold DESC
LIMIT 10;


SELECT strftime('%m', order_date) AS month, ROUND(SUM(total_amount),2) AS revenue
FROM orders
WHERE strftime('%Y', order_date) = '2023'
GROUP BY month
ORDER BY month;


SELECT ROUND(AVG(total_amount),2) AS avg_order_value FROM orders;


CREATE VIEW IF NOT EXISTS customer_spend AS
SELECT c.customer_id, c.name, c.country, ROUND(SUM(o.total_amount),2) AS total_spent
FROM customers c
LEFT JOIN orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_id;


SELECT * FROM customer_spend ORDER BY total_spent DESC LIMIT 10;
\"\"\"
with open(os.path.join(OUTDIR, "task3_queries.sql"), "w") as f:
    f.write(sql_text)
```

```
# --- 3) Read outputs into pandas

df_country = pd.read_sql_query("SELECT c.country, ROUND(SUM(o.total_amount),2) revenue FROM customers c JOIN orders o ON c.customer_id=o.customer_id GROUP BY c.country ORDER BY revenue DESC;", conn)

df_products = pd.read_sql_query("SELECT p.product_name, SUM(oi.quantity) units_sold FROM products p JOIN order_items oi ON p.product_id=oi.product_id GROUP BY p.product_id ORDER BY units_sold DESC LIMIT 10;", conn)

df_month = pd.read_sql_query("SELECT strftime('%m', order_date) month, ROUND(SUM(total_amount),2) revenue FROM orders WHERE strftime('%Y', order_date)='2023' GROUP BY month ORDER BY month;", conn)

df_aov = pd.read_sql_query("SELECT ROUND(AVG(total_amount),2) avg_order_value FROM orders;", conn)

cur.execute(\"CREATE VIEW IF NOT EXISTS customer_spend AS SELECT c.customer_id, c.name, c.country, ROUND(SUM(o.total_amount),2) total_spent FROM customers c LEFT JOIN orders o ON c.customer_id=o.customer_id GROUP BY c.customer_id;\")

conn.commit()

df_customers = pd.read_sql_query("SELECT * FROM customer_spend ORDER BY total_spent DESC LIMIT 10;", conn)


# --- 4) Create visuals

plt.figure(figsize=(6,4))

df_country.plot(x='country', y='revenue', kind='bar', legend=False)

plt.title('Revenue by Country')

plt.tight_layout(); plt.savefig(os.path.join(IMGDIR,'revenue_by_country.png')); plt.close()


plt.figure(figsize=(8,4))

df_month.plot(x='month', y='revenue', kind='line', marker='o', legend=False)

plt.title('Monthly Revenue (2023)')

plt.tight_layout(); plt.savefig(os.path.join(IMGDIR,'monthly_revenue.png')); plt.close()


plt.figure(figsize=(8,4))
```

```
df_products.plot(x='product_name', y='units_sold', kind='bar', legend=False)
plt.title('Top 10 Products by Units Sold')
plt.xticks(rotation=45, ha='right')
plt.tight_layout(); plt.savefig(os.path.join(IMGDIR,'top_products.png')); plt.close()

# dataset samples
story.append(Paragraph('<b>Dataset Samples</b>', styles['Heading2']))
for name, q in [('Customers (sample)','SELECT * FROM customers LIMIT 10;'), ('Products (sample)','SELECT * FROM products LIMIT 10;'), ('Orders (sample)','SELECT * FROM orders LIMIT 10;')]:
  df_sample = pd.read_sql_query(q, conn)
  story.append(Paragraph(f'<b>{name}</b>', styles['Heading3']))
  tbl = [df_sample.columns.tolist()] + df_sample.astype(str).values.tolist()
  t = Table(tbl, repeatRows=1)
  t.setStyle(TableStyle([('BACKGROUND',(0,0),(-1,0),colors.lightgrey),('GRID',(0,0),(-1,-1),0.25,colors.grey)]))
  story.append(t); story.append(Spacer(1,12))
story.append(PageBreak())

# Query outputs
story.append(Paragraph('<b>Query Outputs</b>', styles['Heading2']))
for title, df_out in [('Revenue by Country', df_country), ('Top Products (units sold)', df_products), ('Monthly Revenue (2023)', df_month), ('Average Order Value', df_aov), ('Top Customers (by spend)', df_customers)]:
  story.append(Paragraph(f'<b>{title}</b>', styles['Heading3']))
  tbl = [df_out.columns.tolist()] + df_out.astype(str).values.tolist()
  t = Table(tbl, repeatRows=1)
  t.setStyle(TableStyle([('BACKGROUND',(0,0),(-1,0),colors.lightgrey),('GRID',(0,0),(-1,-1),0.25,colors.grey)]))
  story.append(t); story.append(Spacer(1,12))
```

```
# visuals

story.append(PageBreak()); story.append(Paragraph('<b>Visualizations</b>',
styles['Heading2']))

for img in ['revenue_by_country.png','monthly_revenue.png','top_products.png']:

  p = os.path.join(IMGDIR, img)

  if os.path.exists(p):

    story.append(Paragraph(f'<b>{img.replace("_"," ").replace(".png","").title()}</b>',
styles['Heading3']))

    story.append(Image(p, width=450, height=260)); story.append(Spacer(1,12))


# footer list of files

story.append(PageBreak())

story.append(Paragraph('<b>Files included in the bundle</b>', styles['Heading2']))

for fdesc in ['ecommerce.db (SQLite DB)', 'task3_queries.sql',
'task3_combined_report.pdf', 'images/ (charts)', 'README.md']:

  story.append(Paragraph(f'- {fdesc}', styles['Normal']))

story.append(Spacer(1,12))

story.append(Paragraph(f'Prepared on: {datetime.now().strftime("%B %d, %Y")}',
styles['Normal']))


doc.build(story)
```