Week 8 - Graded Mini Project

Learning Outcome Addressed

- Data Extraction and Manipulation Using SQL and Pandas
- Deriving Business Insights through Exploratory Data Analysis and Visualisation

Objective

This mini project is designed to help you explore and analyse retail sales data using SQL, Pandas, and Python visualisations. You'll work with two datasets: one containing stock information and another recording sales transactions. Through a combination of structured queries and Python-based EDA, you'll derive business insights such as bestselling items, customer purchasing patterns, and revenue trends.

Submission Instructions

Please document your response on the following pages.

Once you have completed the activity, save the file as a PDF and upload it. Be sure to name the file as **Module 8: Graded Mini Project_[Your last name].**

Your submission will be considered complete when it meets the following criteria:

- Includes all the key elements outlined in the activity instructions and the rubric.
- Add screenshots of visual outputs wherever required.
- Adheres to the submission guidelines.
- Is submitted on time.

This is a required activity and counts towards programme completion.

Reflect on the task and respond to the following questions.

A. Basic SQL Queries (5 Tasks)- Load csv files in database. (*Note: Kindly provide the solution in the form of an SQL query.)*

Question: Retrieve all stock items that contain the word **"T-LIGHT"** in their description.

```
select * from sales_transactions where description like '%T-LIGHT%'
```

Question: Calculate the **total quantity sold per StockCode**.

```
select stockcode,sum(quantity) from sales_transactions  group by stockcode
```

Question: Find **total revenue (Quantity** × **UnitPrice)** per **CustomerID**.

```
select customerid,sum(unitprice * quantity) from sales_transactions

group by customerid
```

Question: Get a list of all **invoices** and count of **distinct stock items** per invoice.

```
select invoiceno,count(distinct description) count_of_stock_items from sales_transactions

group by invoiceno
```

Question: Perform an **inner join** between sales and stock details to display full item names along with **total revenue per item.**

```
SELECT

   p.StockCode,

   p.Description AS ItemName,

   SUM(s.Quantity * s.UnitPrice) AS TotalRevenue

FROM sales_transactions s

INNER JOIN stocks p

   ON s.StockCode = p.StockCode

GROUP BY

   p.StockCode,

   p.Description
```

```
ORDER BY

    TotalRevenue DESC;
```

B. EDA Using Pandas (5 Tasks) (*Note: Kindly provide the lines of python code.)*

Question: Check for **missing values, data types**, and **duplicates** in both datasets.

```
import pandas as pd


sales_path = base / 'sales.csv'

stock_path = base / 'StockDetails.csv'


print('Sales dataset missing values')

sales = pd.read_csv(sales_path)

print(sales.info())



print('StockDetails dataset missing values')

stockdetails = pd.read_csv(stock_path)

print(stockdetails.info())


print('Duplicates in sales dataset')

print(sales[sales.duplicated()])


print('Duplicates in stocks dataset')

print(stockdetails[stockdetails.duplicated()])
```

Question: Convert InvoiceDate into datetime, and extract:

- Invoice date
- Month
- Hour of transaction

```
import pandas as pd


sales_path =  'sales.csv'

stock_path = 'StockDetails.csv'


sales_df =  pd.read_csv(sales_path)

stock_df = pd.read_csv(stock_path)


# Convert InvoiceDate to datetime and extract Invoice date, Month, Hour

sales_df['InvoiceDate'] = pd.to_datetime(sales_df['InvoiceDate'], errors='coerce')

# Invoice date (date only)

sales_df['Invoice_date'] = sales_df['InvoiceDate'].dt.date

# Month number (1-12)

sales_df['Month'] = sales_df['InvoiceDate'].dt.month

# Hour of transaction (0-23)

sales_df['Hour'] = sales_df['InvoiceDate'].dt.hour

print('Converted InvoiceDate — NaT values:', sales_df['InvoiceDate'].isna().sum())

# Show sample of new columns

sales_df[['InvoiceDate', 'Invoice_date', 'Month', 'Hour']].head()
```

Question: Add a new column TotalPrice = Quantity × UnitPrice.

```
import pandas as pd


sales_path =  'sales.csv'

stock_path = 'StockDetails.csv'


sales_df =  pd.read_csv(sales_path)

stock_df = pd.read_csv(stock_path)


sales_df['TotalPrice']=sales_df['Quantity']*sales_df['UnitPrice']

sales_df
```

Question: Identify the **top 3 bestselling items** by quantity sold.

```
# Top 3 bestselling items by quantity sold (only positive Quantity = actual sales)

sold_qty = (sales_df[sales_df['Quantity'] > 0]

        .groupby(['StockCode', 'Description'], dropna=False)['Quantity']

        .sum()

        .reset_index(name='SoldQty')

        .sort_values('SoldQty', ascending=False))


print('Top 3 items by Sold Quantity (positive quantities only):')

display(sold_qty.head(3))
```

Question: Find out how many **unique customers** made purchases and the **average quantity per invoice**.

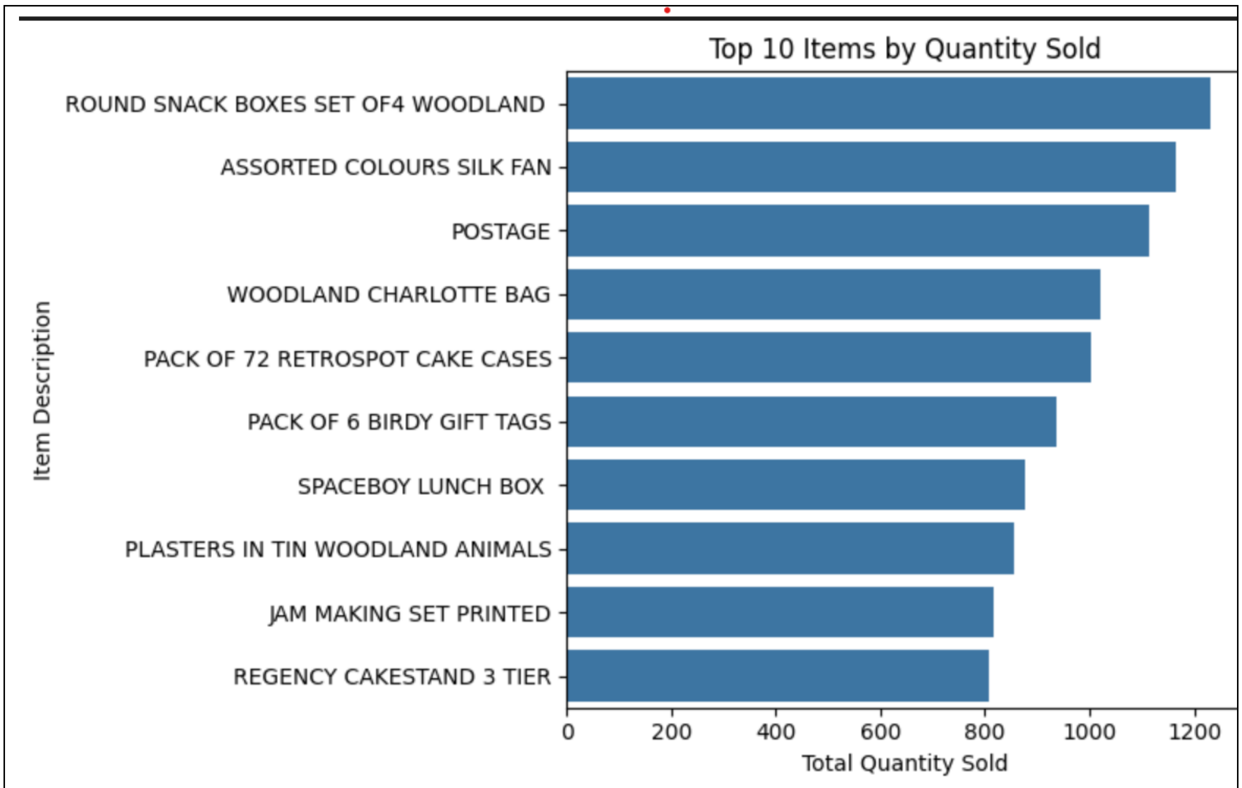
```
# ...existing code...

# Unique customers who made purchases and average quantity per invoice
```

```
n_unique_customers = sales_df.loc[sales_df['Quantity'] > 0, 'CustomerID'].dropna().nunique()

invoice_qty = sales_df.groupby('InvoiceNo')['Quantity'].sum()

positive_invoices = invoice_qty[invoice_qty > 0]

avg_qty_per_invoice = positive_invoices.mean()



print('Unique customers who made purchases:', n_unique_customers)

print(f'Average quantity per (positive) invoice: {avg_qty_per_invoice:.2f}
(n_invoices={len(positive_invoices)})')
```

C. Visualisation (5 Tasks) (*Note: Kindly provide the lines of python code and the screenshots of charts and plots.)*

Question: Plot a **bar chart** of top 10 items by **quantity sold**.

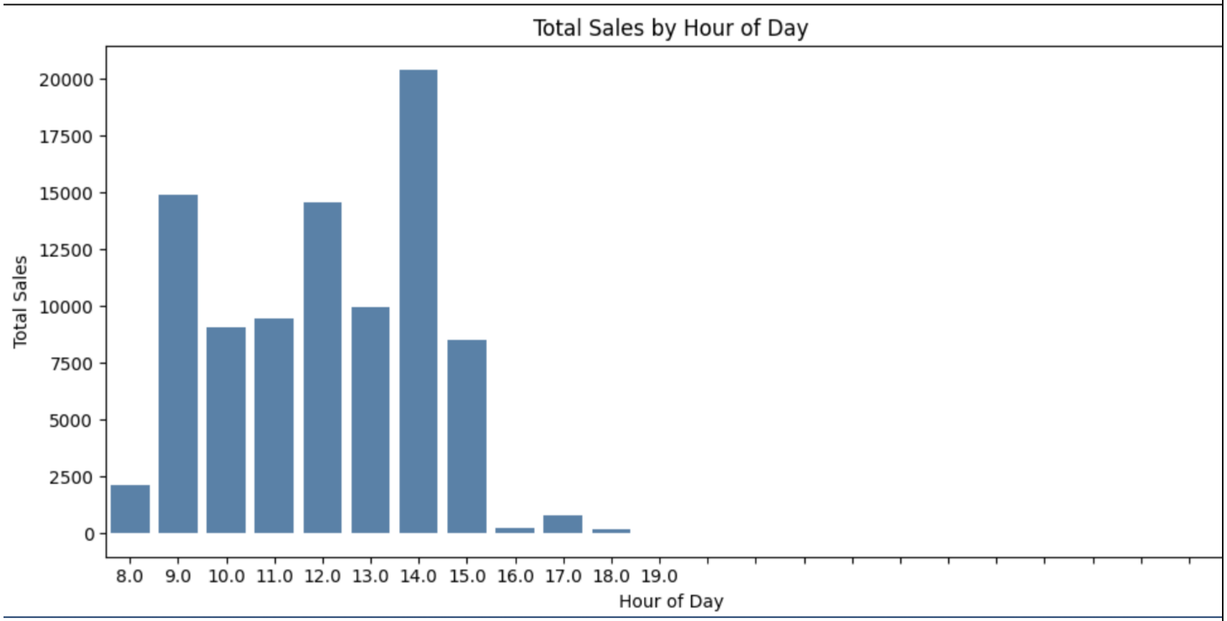
```
import matplotlib.pyplot as plt

import seaborn as sns

sold_qty = (sales_df[sales_df['Quantity'] > 0]

        .groupby(['StockCode', 'Description'], dropna=False)['Quantity']

        .sum()

        .reset_index(name='SoldQty')

        .sort_values('SoldQty', ascending=False))

top10 = sold_qty.head(10)

plt.figure(figsize=(10,6))

sns.barplot(data=top10, x='SoldQty', y='Description')

plt.xlabel('Total Quantity Sold')

plt.ylabel('Item Description')

plt.title('Top 10 Items by Quantity Sold')

plt.tight_layout()

plt.show()
```



Question: Plot **total sales per hour of the day** to understand peak shopping hours.

```
import matplotlib.pyplot as plt

import seaborn as sns



if 'TotalPrice' not in sales_df.columns:

    sales_df['TotalPrice'] = sales_df['Quantity'] * sales_df['UnitPrice']

if 'Hour' not in sales_df.columns:

    sales_df['InvoiceDate'] = pd.to_datetime(sales_df['InvoiceDate'], errors='coerce')

    sales_df['Hour'] = sales_df['InvoiceDate'].dt.hour



hourly = (sales_df.groupby('Hour', dropna=False)

          .agg(TotalSales=('TotalPrice', 'sum'),

               Transactions=('InvoiceNo', 'nunique'))

          .reset_index()
```

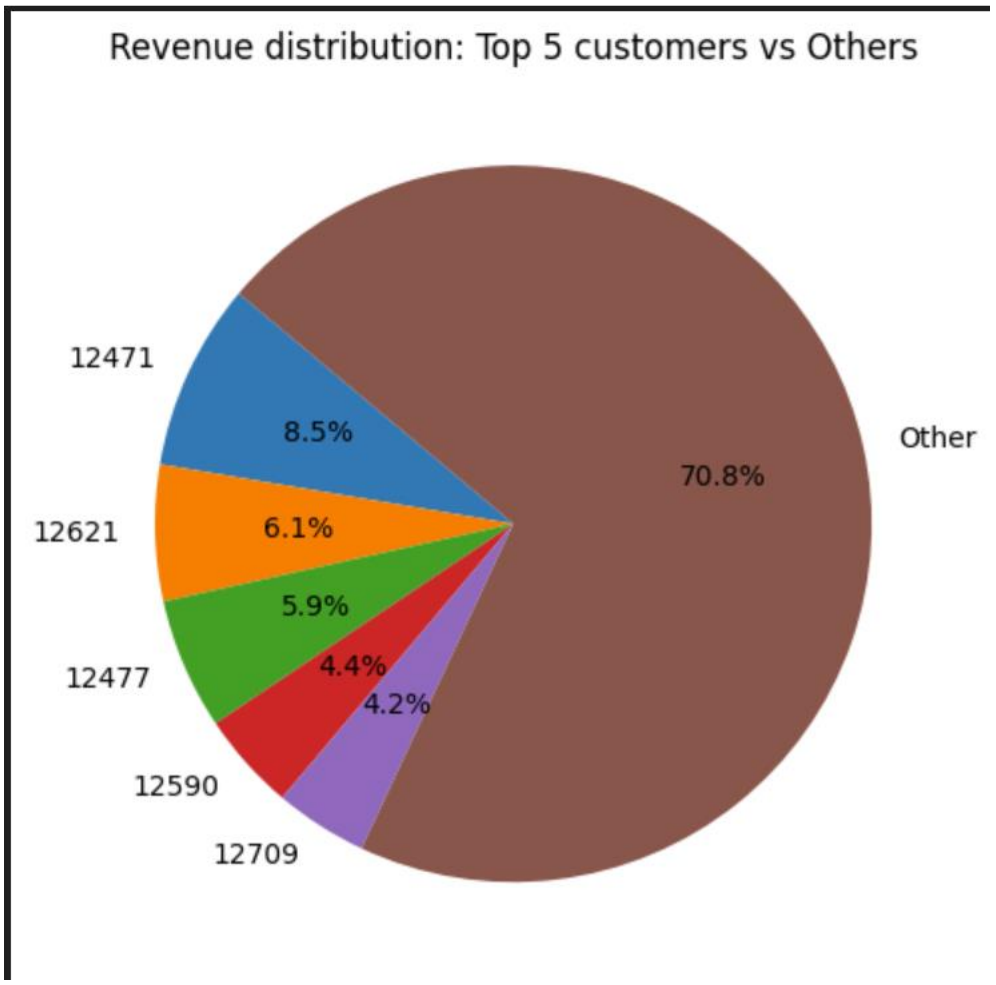
```
        .sort_values('Hour'))


plt.figure(figsize=(10,5))

sns.barplot(data=hourly, x='Hour', y='TotalSales', color='steelblue')

plt.xlabel('Hour of Day')

plt.ylabel('Total Sales')

plt.title('Total Sales by Hour of Day')

plt.xticks(range(0,24))

plt.tight_layout()

plt.show()
```



Question: Create a **pie chart** showing revenue distribution among top 5 customers.

```
import matplotlib.pyplot as plt

import pandas as pd
```

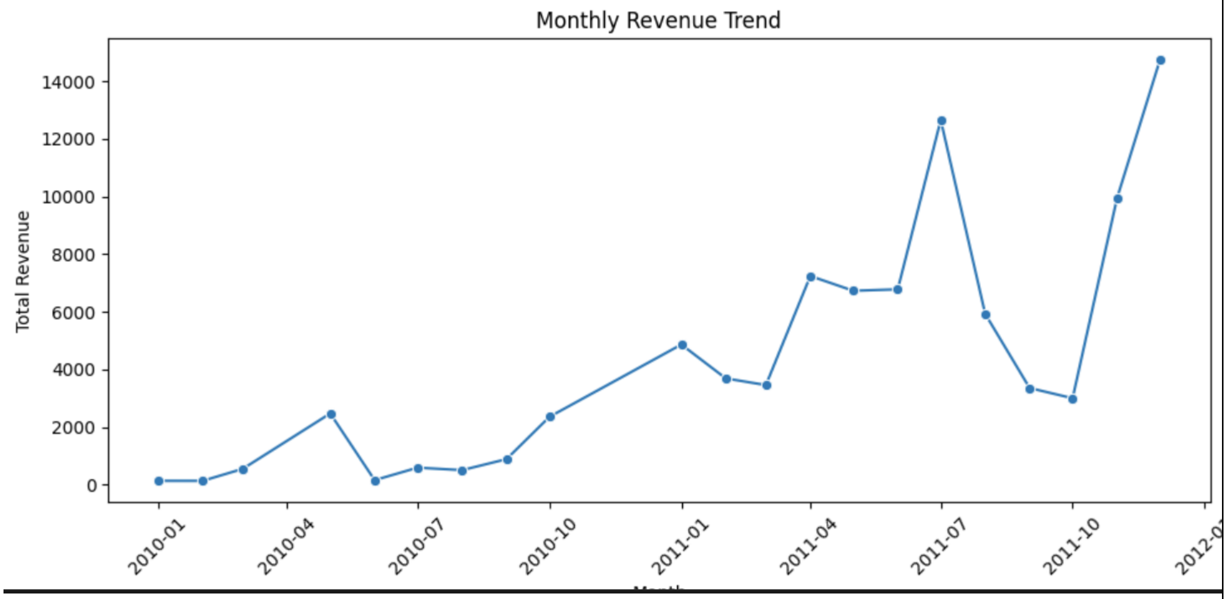
```
# Ensure TotalPrice exists

if 'TotalPrice' not in sales_df.columns:

    sales_df['TotalPrice'] = sales_df['Quantity'] * sales_df['UnitPrice']



# Revenue by customer (exclude missing CustomerID)

revenue_by_customer = (sales_df.dropna(subset=['CustomerID'])

                       .groupby('CustomerID')['TotalPrice']

                       .sum()

                       .sort_values(ascending=False))



top5 = revenue_by_customer.head(5)

others = revenue_by_customer.iloc[5:].sum()

pie_data = pd.concat([top5, pd.Series({'Other': others})])



plt.figure(figsize=(7,7))

plt.pie(pie_data, labels=pie_data.index.astype(str), autopct='%1.1f%%', startangle=140)

plt.title('Revenue distribution: Top 5 customers vs Others')

plt.tight_layout()

plt.show()
```



Question: Visualise the **monthly revenue trend** using a line chart.

```
import matplotlib.pyplot as plt

import seaborn as sns



# ensure InvoiceDate and TotalPrice exist

sales_df['InvoiceDate'] = pd.to_datetime(sales_df['InvoiceDate'], errors='coerce')

if 'TotalPrice' not in sales_df.columns:

    sales_df['TotalPrice'] = sales_df['Quantity'] * sales_df['UnitPrice']



sales_df = sales_df.dropna(subset=['InvoiceDate'])
```

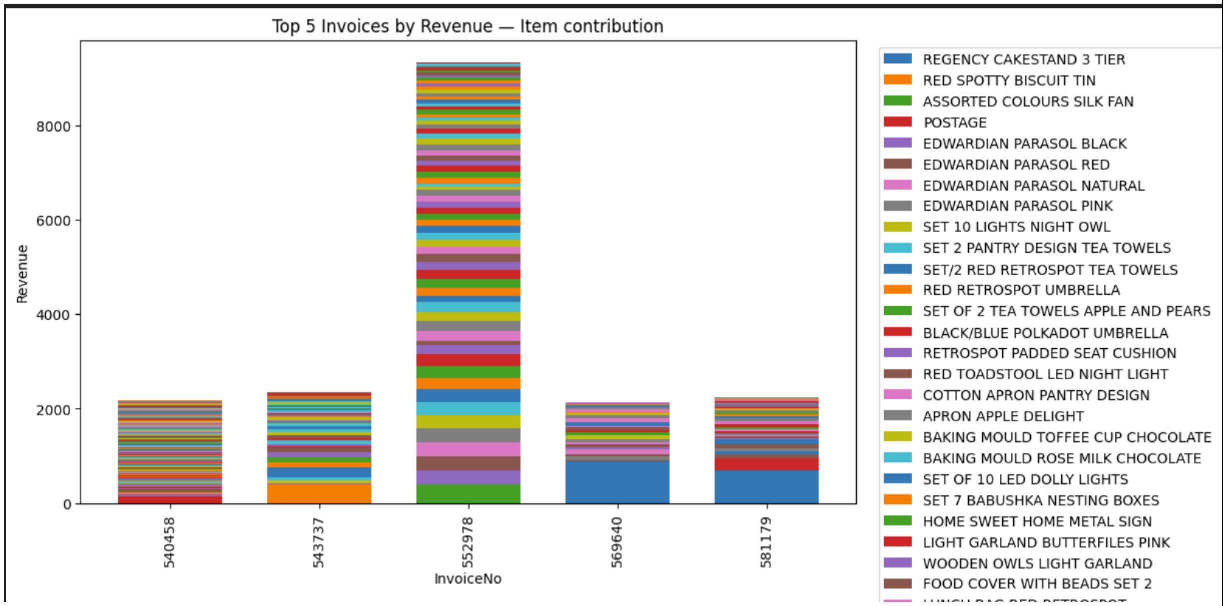
```
sales_df['YearMonth'] = sales_df['InvoiceDate'].dt.to_period('M').astype(str)


monthly = (sales_df.groupby('YearMonth')

        .agg(TotalRevenue=('TotalPrice', 'sum'),

             Transactions=('InvoiceNo', 'nunique'))

        .reset_index())


monthly['YearMonth_dt'] = pd.to_datetime(monthly['YearMonth'])


plt.figure(figsize=(10,5))

sns.lineplot(data=monthly.sort_values('YearMonth_dt'), x='YearMonth_dt', y='TotalRevenue',
marker='o')

plt.xlabel('Month')

plt.ylabel('Total Revenue')

plt.title('Monthly Revenue Trend')

plt.xticks(rotation=45)

plt.tight_layout()

plt.show()
```



Question: Create a **stacked bar chart** of top 5 invoices by revenue, showing contribution from each item.

```
import matplotlib.pyplot as plt


# Ensure TotalPrice exists

if 'TotalPrice' not in sales_df.columns:

    sales_df['TotalPrice'] = sales_df['Quantity'] * sales_df['UnitPrice']


# Use only positive revenue lines and treat InvoiceNo as string

sales_pos = sales_df[sales_df['TotalPrice'] > 0].copy()

sales_pos['InvoiceNo'] = sales_pos['InvoiceNo'].astype(str)


# Top 5 invoices by total revenue

top_invoices = (sales_pos.groupby('InvoiceNo')['TotalPrice'].sum()

                .nlargest(5).index.tolist())


# Pivot to get revenue contribution per item for each top invoice
```

```
invoice_item = (sales_pos[sales_pos['InvoiceNo'].isin(top_invoices)]

            .groupby(['InvoiceNo', 'Description'])['TotalPrice'].sum()

            .reset_index()

            .pivot(index='InvoiceNo', columns='Description', values='TotalPrice')

            .fillna(0))


# Sort item columns by total contribution for consistent stacking

invoice_item = invoice_item[invoice_item.sum().sort_values(ascending=False).index]


# Plot stacked bar chart

plt.figure(figsize=(10,6))

invoice_item.plot(kind='bar', stacked=True, width=0.7, ax=plt.gca())

plt.ylabel('Revenue')

plt.title('Top 5 Invoices by Revenue — Item contribution')

plt.legend(bbox_to_anchor=(1.02, 1), loc='upper left')

plt.tight_layout()

plt.show()
```



D. Business Insights (5 Tasks) (*Note: Kindly provide the lines of python code.)*

Question: Identify which **product generates the highest total revenue**.

```
# Identify product with highest total revenue

if 'TotalPrice' not in sales_df.columns:

   sales_df['TotalPrice'] = sales_df['Quantity'] * sales_df['UnitPrice']



product_rev = (sales_df.groupby(['StockCode', 'Description'], dropna=False)['TotalPrice']

              .sum()

              .reset_index(name='TotalRevenue')

              .sort_values('TotalRevenue', ascending=False))



top_product = product_rev.iloc[0]

display(product_rev.head(5))
```

Question: Determine **average order value per invoice**.

```
invoice_totals = sales_df.groupby('InvoiceNo', dropna=False)['TotalPrice'].sum()

pos_invoice_totals = invoice_totals[invoice_totals > 0]



avg_order_value = pos_invoice_totals.mean()

median_order_value = pos_invoice_totals.median()

n_invoices = len(pos_invoice_totals)



print(f'Average order value (per positive invoice): {avg_order_value:.2f}
(n_invoices={n_invoices})')

print(f'Median order value: {median_order_value:.2f}')
```

Question: Find the **customer with the highest number of transactions**.

```
sales_df['InvoiceNo'] = sales_df['InvoiceNo'].astype(str)

transactions = (sales_df[sales_df['TotalPrice'] > 0]

            .dropna(subset=['CustomerID'])

            .drop_duplicates(subset=['CustomerID', 'InvoiceNo'])

            .groupby('CustomerID')['InvoiceNo']

            .nunique()

            .sort_values(ascending=False))


if len(transactions) == 0:

   print('No valid transactions found.')

else:

   top_customer = transactions.index[0]

   top_count = transactions.iloc[0]

   print(f'Customer with most transactions: {top_customer} (n_transactions={top_count})')

   display(transactions.head(10))
```

Question: Check how many products in the sales data do **not have a matching entry** in the stockDetails file.

```
if 'stock_df' not in globals():

   try:

      stock_df = pd.read_csv('StockDetails.csv')

   except Exception:

      stock_df = None


sales_codes = sales_df['StockCode'].astype(str).str.strip().dropna().unique()

if stock_df is None or 'StockCode' not in stock_df.columns:

   print('StockDetails not loaded or missing StockCode column.')
```

```
else:
    stock_codes = stock_df['StockCode'].astype(str).str.strip().dropna().unique()
    missing = sorted(set(sales_codes) - set(stock_codes))
    print(f'Products in sales missing from StockDetails: {len(missing)}')
    print('Sample missing StockCodes:', missing[:20])
```