DATA ANALYST (Phase 2: Intermediate Intern Tasks)

- ## THEORETICAL TASKS

1. End-to-End Data Analysis Lifecycle (Real Business Example)

The end-to-end data analysis lifecycle is a structured process used by data analysts to convert raw data into meaningful business insights.
Consider an e-commerce company that wants to reduce product return rates.

First, the company performs business understanding, where the main issue—high return rate—is clearly identified.
Next, in problem definition, a measurable goal is set, such as reducing returns by 15% within three months.

After defining the goal, the analyst performs data collection from different sources like:

- Sales transaction records
- Return history logs
- Customer feedback and reviews
- Product catalog information

The collected data is then cleaned by:

- Removing duplicate entries
- Handling missing values
- Correcting incorrect formats

Once the data is clean, data integration combines multiple datasets into a single structured dataset.

The analyst then performs Exploratory Data Analysis (EDA) to find:

- Frequently returned products
- Regions with higher returns
- Customer behavior patterns

Next comes feature selection and statistical analysis, where important variables influencing returns are identified.

The results are presented through visualization and reporting, such as dashboards and charts, so business managers can easily understand them.

Finally, insights are generated, decisions are taken (like improving product quality or descriptions), and results are continuously monitored for improvement.

This complete workflow represents the real-world data analysis lifecycle.

---

2. Descriptive vs Diagnostic Analytics; Correlation vs Causation

Descriptive Analytics

Descriptive analytics explains **what has happened in the past** by summarizing historical data.

Characteristics:

- Uses totals, averages, and summaries
- Presented through reports and dashboards
- Helps track past performance

Example:
Total sales achieved in the previous month.

Diagnostic Analytics

Diagnostic analytics explains **why something happened**.

Characteristics:

- Identifies root causes of problems
- Uses comparisons, drill-downs, and correlations
- Helps in problem solving and decision making

Example:
Finding the reason for a sudden drop in sales.

Correlation

Correlation shows a **relationship between two variables**, meaning they move together.

- Does **not prove cause**
- Can be positive or negative
- Used mainly in exploratory analysis

Example:
Ice cream sales increase when temperature rises.

Causation

Causation means **one variable directly causes another**.

- Shows true cause-and-effect
- Requires logical proof or experimentation
- More reliable for business decisions

Example:
Higher temperature **causes** increased ice cream demand.

Understanding the difference between correlation and causation is important to **avoid wrong conclusions**.

---

3. Short Notes

Data Bias

Data bias occurs when a dataset **does not properly represent reality**, leading to incorrect insights.

Causes:

- Poor sampling
- Limited demographic coverage
- Human assumptions

Effects:

- Misleading results
- Unfair or inaccurate decisions
- Reduced reliability of analysis

Bias can be reduced by **using diverse and balanced data sources**.

Missing Data Strategies

Missing data is common in real datasets. Analysts handle it using:

- Removing incomplete rows or columns
- Filling values with **mean, median, or mode**
- Forward or backward filling
- Interpolation or prediction methods
- Treating missing values as a separate category

The correct method depends on **data importance and context**.

KPIs vs Metrics

KPIs (Key Performance Indicators):

- Measure **critical business goals**
- Directly linked to success
- Few in number and strategic

Metrics:

- Measure **general performance**
- Support KPIs but are not always strategic
- Larger in number

Example:

- KPI → Revenue growth rate
- Metric → Daily sales count

All KPIs are metrics, but **not all metrics are KPIs**.

---

4. Case Study: Why Dashboards Fail Even with Correct Data

Even when data is accurate, dashboards may fail due to **design and usability issues**.

Common reasons:

- Not aligned with business goals
- Too many charts causing confusion
- Wrong choice of visualization
- KPIs not clearly defined
- Data not updated regularly
- Poor layout and complex design
- Users not trained to interpret data
- Insights not explained in simple language
- No clear action points or recommendations
- Dashboard not interactive or role-specific

A successful dashboard should be:

- Simple and clear
- Focused on important KPIs
- Easy for non-technical users to understand
- Able to provide actionable business insights

PRACTICAL TASKS :

Github Link :Github link

Task 1: Business Dataset Analysis

Downloaded dataset from Kaggle.

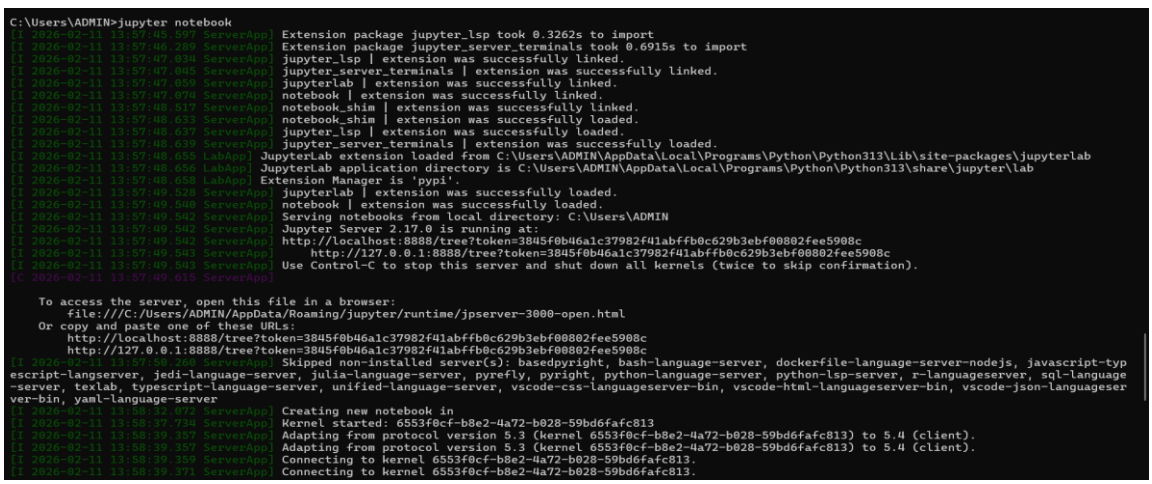
product info

customer info

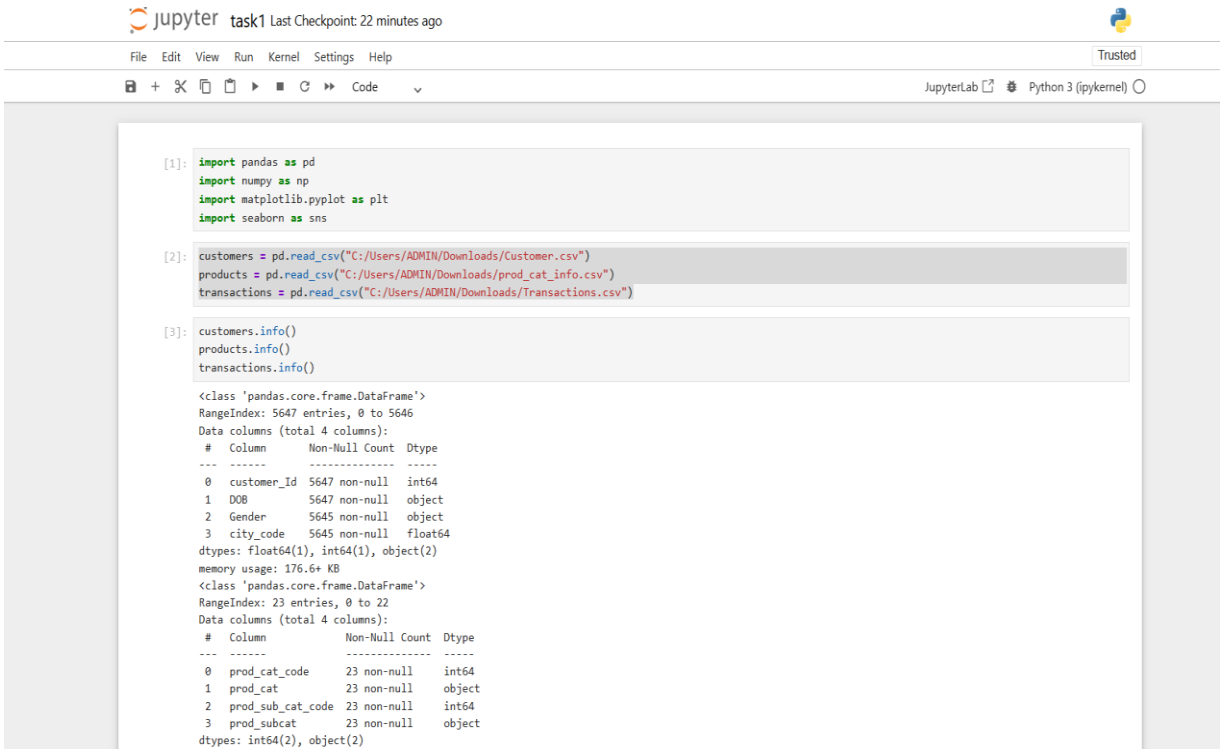
Transactions info

The above is the dataset of a **online retail store.**

Now on command prompt type this:



Now jupyter will be opended:




```
dtypes: int64(2), object(2)
memory usage: 868.0+ bytes
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23053 entries, 0 to 23052
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   transaction_id     23053 non-null  int64
 1   cust_id            23053 non-null  int64
 2   tran_date          23053 non-null  object
 3   prod_subcat_code   23053 non-null  int64
 4   prod_cat_code      23053 non-null  int64
 5   Qty                23053 non-null  int64
 6   Rate               23053 non-null  int64
 7   Tax                23053 non-null  float64
 8   total_amt          23053 non-null  float64
 9   Store_type         23053 non-null  object
dtypes: float64(2), int64(6), object(2)
memory usage: 1.8+ MB
```

```
[5]:
customers.isnull().sum()
products.isnull().sum()
transactions.isnull().sum()
```

```
[5]: transaction_id      0
     cust_id             0
     tran_date           0
     prod_subcat_code    0
     prod_cat_code       0
     Qty                 0
     Rate                0
     Tax                 0
     total_amt           0
     Store_type          0
     dtype: int64
```

[6]:
```
customers.duplicated().sum()
transactions.duplicated().sum()
```

[6]:
```
np.int64(13)
```

[8]:
```
customers = customers.drop_duplicates()
transactions = transactions.drop_duplicates()
transactions = transactions.dropna()
```

[16]:
```
print(transactions.columns)
print(customers.columns)
```

```
Index(['transaction_id', 'cust_id', 'tran_date', 'prod_subcat_code',
       'prod_cat_code', 'Qty', 'Rate', 'Tax', 'total_amt', 'Store_type'],
      dtype='object')
Index(['customer_Id', 'DOB', 'Gender', 'city_code'], dtype='object')
```

[20]:
```
print("Customers:", customers.columns.tolist())
print("Products:", products.columns.tolist())
print("Transactions:", transactions.columns.tolist())
```

```
Customers: ['customer_Id', 'DOB', 'Gender', 'city_code']
Products: ['prod_cat_code', 'prod_cat', 'prod_sub_cat_code', 'prod_subcat']
Transactions: ['transaction_id', 'cust_id', 'tran_date', 'prod_subcat_code', 'prod_cat_code', 'Qty', 'Rate', 'Tax', 'total_amt', 'Store_type']
```

[21]:
```
customers.columns = customers.columns.str.lower().str.strip()
products.columns = products.columns.str.lower().str.strip()
transactions.columns = transactions.columns.str.lower().str.strip()
```

[23]:
```
print(customers.head())
print(products.head())
print(transactions.head())
```

```
   customer_id         dob gender  city_code
0       268408  02/01/1970      M        4.0
1       269696  07/01/1970      F        8.0
2       268159  08/01/1970      F        8.0
3       270181  10/01/1970      F        2.0
4       268073  11/01/1970      M        1.0
   prod_cat_code  prod_cat  prod_sub_cat_code prod_subcat
0              1  Clothing                  4        Mens
1              1  Clothing                  1       Women
2              1  Clothing                  3        Kids
3              2  Footwear                  1        Mens
4              2  Footwear                  3       Women
   transaction_id  cust_id   tran_date  prod_subcat_code  prod_cat_code  qty  \
0     80712190438   270351  28-02-2014                 1              1   -5
1     29258453508   270384  27-02-2014                 5              3   -5
2     51750724947   273420  24-02-2014                 6              5   -2
3     93274880719   271509  24-02-2014                11              6   -3
4     51750724947   273420  23-02-2014                 6              5   -2

   rate      tax  total_amt store_type
0  -772  405.300  -4265.300     e-Shop
1 -1497  785.925  -8270.925     e-Shop
2  -791  166.110  -1748.110   TeleShop
3 -1363  429.345  -4518.345     e-Shop
4  -791  166.110  -1748.110   TeleShop
```

[24]:
```
print("CUSTOMERS COLUMNS:", customers.columns.tolist())
print("TRANSACTIONS COLUMNS:", transactions.columns.tolist())
print("PRODUCTS COLUMNS:", products.columns.tolist())
```

```
CUSTOMERS COLUMNS: ['customer_id', 'dob', 'gender', 'city_code']
TRANSACTIONS COLUMNS: ['transaction_id', 'cust_id', 'tran_date', 'prod_subcat_code', 'prod_cat_code', 'qty', 'rate', 'tax', 'total_amt', 'store_type']
PRODUCTS COLUMNS: ['prod_cat_code', 'prod_cat', 'prod_sub_cat_code', 'prod_subcat']
```

[26]:
```
print(products.columns.tolist())
```

```
['prod_cat_code', 'prod_cat', 'prod_sub_cat_code', 'prod_subcat']
```

[27]:
```
df = transactions.merge(customers, left_on="cust_id", right_on="customer_id", how="left")

df = df.merge(
    products,
    left_on=["prod_cat_code", "prod_subcat_code"],
    right_on=["prod_cat_code", "prod_sub_cat_code"],
    how="left"
)

df.head()
```

[27]:

	transaction_id	cust_id	tran_date	prod_subcat_code	prod_cat_code	qty	rate	tax	total_amt	store_type	customer_id	dob	gender	city_code	prod_
0	80712190438	270351	28-02-2014	1	1	-5	-772	405.300	-4265.300	e-Shop	270351	26/09/1981	M	5.0	Cloth
1	29258453508	270384	27-02-2014	5	3	-5	-1497	785.925	-8270.925	e-Shop	270384	11/05/1973	F	8.0	Electror
2	51750724947	273420	24-02-2014	6	5	-2	-791	166.110	-1748.110	TeleShop	273420	27/07/1992	M	8.0	Bo
3	93274880719	271509	24-02-2014	11	6	-3	-1363	429.345	-4518.345	e-Shop	271509	08/06/1981	M	3.0	Home a kitcl
4	51750724947	273420	23-02-2014	6	5	-2	-791	166.110	-1748.110	TeleShop	273420	27/07/1992	M	8.0	Bo

[27]:
```
df = transactions.merge(customers, left_on="cust_id", right_on="customer_id", how="left")

df = df.merge(
    products,
    left_on=["prod_cat_code", "prod_subcat_code"],
    right_on=["prod_cat_code", "prod_sub_cat_code"],
    how="left"
)

df.head()
```

[27]:

prod_subcat_code	prod_cat_code	qty	rate	tax	total_amt	store_type	customer_id	dob	gender	city_code	prod_cat	prod_sub_cat_code	prod_subcat
1	1	-5	-772	405.300	-4265.300	e-Shop	270351	26/09/1981	M	5.0	Clothing	1	Women
5	3	-5	-1497	785.925	-8270.925	e-Shop	270384	11/05/1973	F	8.0	Electronics	5	Computers
6	5	-2	-791	166.110	-1748.110	TeleShop	273420	27/07/1992	M	8.0	Books	6	DIY
11	6	-3	-1363	429.345	-4518.345	e-Shop	271509	08/06/1981	M	3.0	Home and kitchen	11	Bath
6	5	-2	-791	166.110	-1748.110	TeleShop	273420	27/07/1992	M	8.0	Books	6	DIY

[30]:
```
df["tran_date"] = pd.to_datetime(
    df["tran_date"],
    format="mixed",
    dayfirst=True,
    errors="coerce"
)
```

[31]:
```
df["tran_date"].isna().sum()
```

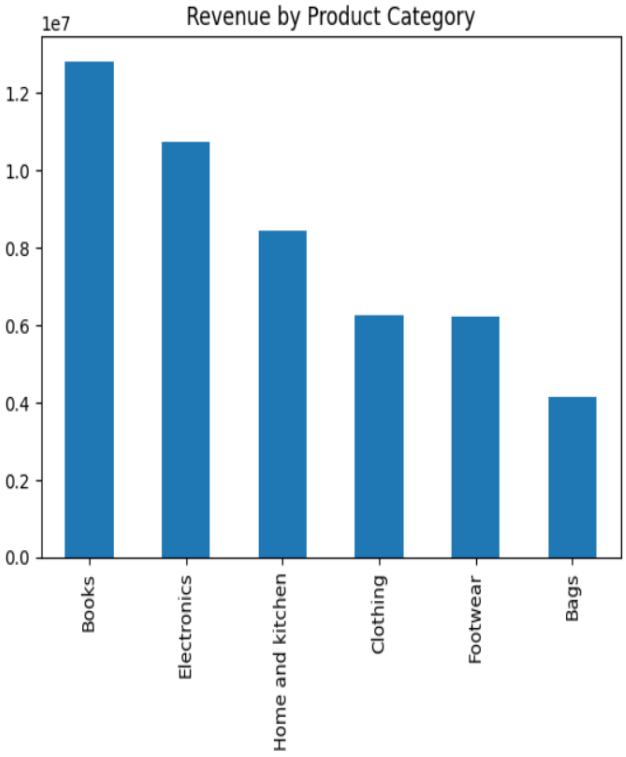
[31]:
```
np.int64(0)
```

[32]:
```
df["month"] = df["tran_date"].dt.to_period("M")
df.groupby("month")["total_amt"].sum()
```

[32]:
```
month
2011-01     337307.880
2011-02    1087283.535
2011-03    1376537.175
2011-04    1379838.915
2011-05    1174157.530
2011-06    1226771.000
2011-07    1310933.325
2011-08    1230637.395
2011-09    1437136.480
2011-10    1399860.410
2011-11    1450866.105
2011-12    1332084.130
2012-01    1207435.710
2012-02    1306702.280
2012-03    1440357.555
2012-04    1214127.590
2012-05    1351648.155
2012-06    1295998.145
2012-07    1246654.370
2012-08    1325005.500
2012-09    1320331.350
2012-10    1413123.725
2012-11    1379593.605
2012-12    1417278.525
2013-01    1378639.990
2013-02    1194731.525
2013-03    1465626.695
2013-04    1276018.640
2013-05    1286049.830
2013-06    1254308.705
2013-07    1226628.455
2013-08    1397714.500
2013-09    1339146.185
2013-10    1368587.805

2013-11    1269408.530
2013-12    1257120.930
2014-01    1500564.585
2014-02     735078.045
Freq: M, Name: total_amt, dtype: float64
```

Q1. Which product category generates the highest total revenue for the business?

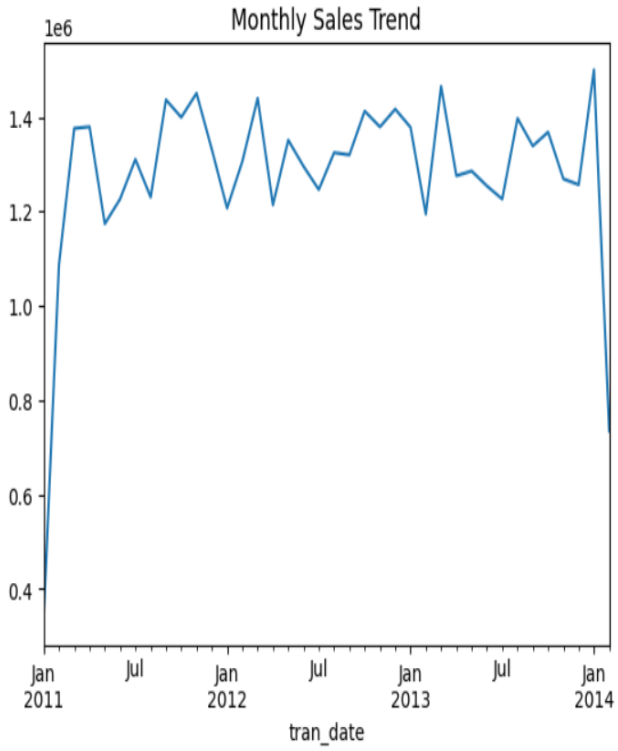
[33]:
```
category_sales = df.groupby("prod_cat")["total_amt"].sum().sort_values(ascending=False)
category_sales.plot(kind="bar", title="Revenue by Product Category")
plt.show()
```



Prod_cat

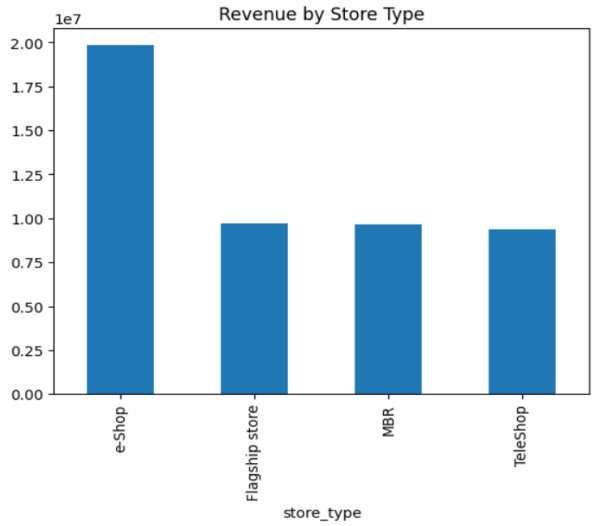
Q2. How do total sales change month-by-month across the year? Are there seasonal peaks or drops?

[34]:
```
monthly_sales = df.groupby(df["tran_date"].dt.to_period("M"))["total_amt"].sum()
monthly_sales.plot(kind="line", title="Monthly Sales Trend")
plt.show()
```



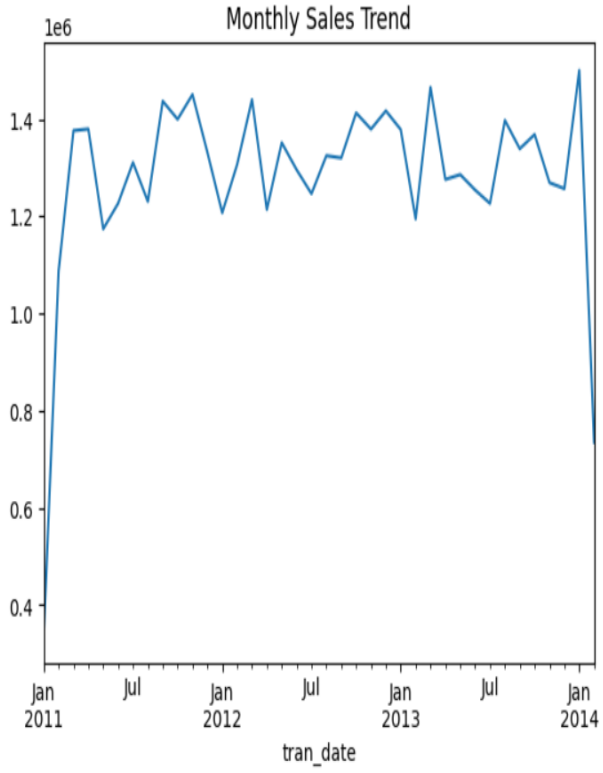
Q3. Which store type contributes the most revenue, and which performs the least?

[41]:
```
import matplotlib.pyplot as plt

store_sales = (
    df.groupby("store_type")["total_amt"]
    .sum()
    .sort_values(ascending=False)
)

store_sales.plot(kind="bar", title="Revenue by Store Type")
plt.show()
```



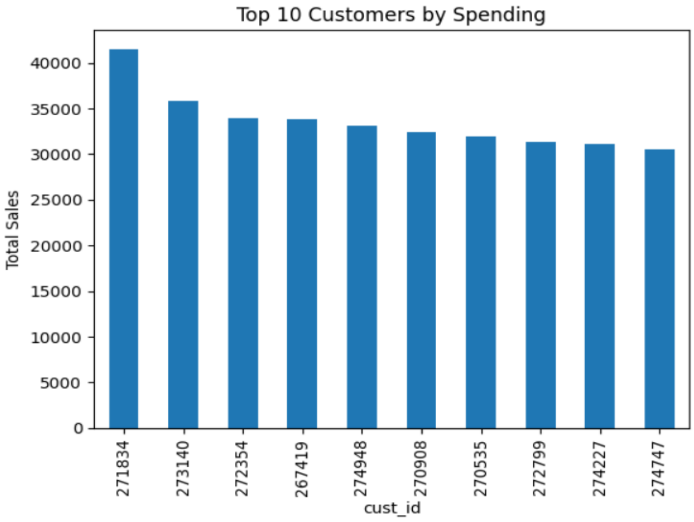
Q4. Which month recorded the highest overall sales?

[42]:
```
monthly_sales = df.groupby(df["tran_date"].dt.to_period("M"))["total_amt"].sum()
monthly_sales.plot(kind="line", title="Monthly Sales Trend")
plt.show()
```



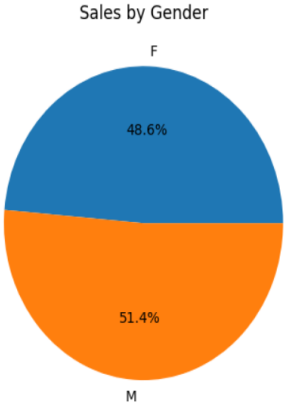
Q5. *Who are the top 10 customers contributing the highest total sales revenue?*

[44]:
```
import matplotlib.pyplot as plt

top_customers = (
    df.groupby("cust_id")["total_amt"]
    .sum()
    .nlargest(10)
)

top_customers.plot(kind="bar", title="Top 10 Customers by Spending")
plt.ylabel("Total Sales")
plt.show()
```



Q6. How does total sales contribution differ between male and female customers?

[47]:
```
import matplotlib.pyplot as plt

gender_sales = df.groupby("gender")["total_amt"].sum()

gender_sales.plot(kind="pie", autopct="%1.1f%%", title="Sales by Gender")
plt.ylabel("")   # remove y-label for clean pie chart
plt.show()
```



Retail Sales Data Analysis – Insight Report of Task 1

Objective :

The purpose of this analysis is to understand customer purchasing behavior, product performance, and revenue trends using retail transaction data. The insights help businesses make better decisions in marketing, inventory planning, and customer targeting.

Key Findings :

1. Product Category Performance
The analysis shows that a small number of product categories generate the majority of total revenue. This indicates that focusing promotions, stock availability, and marketing campaigns on high-performing categories can significantly improve overall sales.

2. Monthly Sales Trend
Sales vary across months, with noticeable peaks during certain periods that may correspond to festive seasons, discounts, or increased customer demand. Understanding these seasonal trends helps businesses plan inventory, staffing, and promotional strategies more effectively.

3. Store Type Contribution
Different store formats contribute unevenly to total revenue. One or two store types clearly outperform others, suggesting that the business should invest more in successful formats while improving or rethinking weaker ones.

4. High-Value Customers
A small group of customers contributes a large share of total revenue. Identifying and retaining these high-value customers through loyalty programs, personalized offers, and targeted communication can increase long-term profitability.

5. Gender-Based Sales Distribution
Sales contribution differs slightly between male and female customers. This insight can support gender-specific marketing strategies, product recommendations, and promotional campaigns.

Business Impact

Overall, the analysis highlights the importance of category focus, seasonal planning, store optimization, and customer retention. By acting on these insights, businesses can improve revenue growth, enhance customer satisfaction, and make data-driven strategic decisions.

Task 2: SQL Challenge