Project Title	Analyzing Sales data
Tools	Jupyter Notebook and VS code
Technologies	Business Intelligence
Domain	E-commerce
Project Difficulties level	Advanced

Dataset : Dataset is available in the given link. You can download it at your convenience.

Click here to download data set

Analyzing Sales Data Project

This project involves analyzing Amazon sales data to gain insights into sales performance, identify trends, and make data-driven business decisions. Here's a step-by-step guide:

1. Problem Definition

Objective: Analyze Amazon sales data to understand sales trends, identify top-performing products, and optimize inventory and marketing strategies.

2. Data Collection

Datasets: Obtain sales data from Amazon. This could include:

- Order data: Order ID, product ID, order date, sales amount, etc.
- Product data: Product ID, category, price, ratings, reviews, etc.
- Customer data: Customer ID, location, demographics, etc.

3. Data Preprocessing

```
import pandas as pd

# Load datasets
orders = pd.read_csv('amazon_orders.csv')
products = pd.read_csv('amazon_products.csv')
customers = pd.read_csv('amazon_customers.csv')

# Display basic info and check for missing values
print(orders.info())
print(products.info())
print(customers.info())

# Fill missing values or drop rows/columns as necessary
orders.fillna(method='ffill', inplace=True)
products.fillna(method='ffill', inplace=True)
customers.fillna(method='ffill', inplace=True)
```

4. Exploratory Data Analysis (EDA)

```
import seaborn as sns
import matplotlib.pyplot as plt
```

```
# Basic statistics
print(orders.describe())
print(products.describe())
print(customers.describe())

# Histograms for numeric features
orders.hist(bins=30, figsize=(20, 15))
plt.show()

# Sales trend over time
orders['order_date'] = pd.to_datetime(orders['order_date'])
sales_trend = orders.groupby(orders['order_date'].dt.to_period('M')).sum()
sales_trend['sales_amount'].plot(figsize=(10, 6), title='Sales Trend Over Time')
plt.show()

# Top-selling products
top_products = orders.groupby('product_id').sum().sort_values('sales_amount',
ascending=False).head(10)
sns.barplot(x=top_products.index, y=top_products['sales_amount'])
plt.title('Top 10 Selling Products')
plt.show()
```

5. Feature Engineering

```
# Example feature engineering
orders['order_month'] = orders['order_date'].dt.month
orders['order_year'] = orders['order_date'].dt.year

# Merge datasets
```

```
data = pd.merge(orders, products, on='product_id')
data = pd.merge(data, customers, on='customer_id')
```

6. Model Selection

For predictive modeling, you might want to predict future sales, identify customer segments, or recommend products.

Predicting Future Sales

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

# Define features and target variable
X = data[['order_month', 'order_year', 'price', 'ratings']]
y = data['sales_amount']

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train the model
model = LinearRegression()
model.fit(X_train, y_train)

# Evaluate the model
y_pred = model.predict(X_test)
print(f"Mean Squared Error: {mean_squared_error(y_test, y_pred)}")
print(f"R2 Score: {r2_score(y_test, y_pred)}")
```

7. Model Interpretation

```
import matplotlib.pyplot as plt

# Coefficients of the model
coefficients = pd.DataFrame(model.coef_, X.columns, columns=['Coefficient'])
print(coefficients)
```

8. Deployment

For deployment, you could build a web application to visualize sales trends, recommend products, or provide sales forecasts.

```
from flask import Flask, request, jsonify

app = Flask(__name__)

@app.route('/predict', methods=['POST'])
def predict():
    data = request.get_json(force=True)
    input_data = np.array([data['order_month'], data['order_year'], data['price'], data['ratings']])
    prediction = model.predict([input_data])
    return jsonify({'predicted_sales_amount': prediction[0]})

if __name__ == '__main__':
    app.run(debug=True)
```

9. Monitoring and Maintenance

Set up logging and monitoring to track the performance of your deployed model, and schedule regular retraining with new data.

10. Documentation and Reporting

Maintain comprehensive documentation of the project, including data sources, preprocessing steps, model selection, and evaluation results. Create detailed reports and visualizations to communicate findings and insights to stakeholders.

Additional Considerations

- **Ethical Considerations**: Ensure ethical use of data, especially customer data.
- **Privacy and Security**: Implement measures to protect sensitive customer and business data.

Tools and Technologies

- **Programming Language**: Python
- **Libraries**: pandas, numpy, seaborn, matplotlib, scikit-learn, Flask
- **Visualization Tools**: Tableau, Power BI, or any dashboarding tool for advanced visualizations

This is a basic outline of an Amazon sales data analysis project. Depending on your specific goals and data, you may need to adjust the steps accordingly.

Sample Project Report

We will use the following libraries

1. Pandas: Data manipulation and analysis

2. Numpy: Numerical operations and calculations

3. Matplotlib: Data visualization and plotting

4. Seaborn: Enhanced data visualization and statistical graphics ¶

5. Scipy: Scientific computing and advanced mathematical operations

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy as sp
# this is for jupyter notebook to show the plot in the notebook itself instead of opening a new window
%matplotlib inline
```

Data Loading and Exploration | Cleaning

¶

Load a CSV file then creating a dataframe

In [2]:
```
df = pd.read_csv("/kaggle/input/amazon-sales-dataset/amazon.csv")
```

Set the option to show maximum columns

In [3]:
```
pd.set_option('display.max_columns', None)
```

Get a sneak peek of data

The purpose of a sneak peek is to get a quick overview of the data and identify any potential problems or areas of interest.

In [4]:
```
# Let's have a look on top 5 rows of the data
df.head(5)
```

Out[4]:

	product_id	product_name	category	discounted_price	actual_price	discount_percentage	rating	rating_count	about_product	u
0	B07JW9H4J1	Wayona Nylon Braided USB to Lightning Fast Cha...	Computers&Accessories|Accessories&Peripherals|...	₹399	₹1,099	64%	4.2	24,269	High Compatibility : Compatible With iPhone 12...	A
1	B098NS6PVG	Ambrane Unbreakable 60W / 3A Fast Charging 1.5...	Computers&Accessories|Accessories&Peripherals|...	₹199	₹349	43%	4.0	43,994	Compatible with all Type C enabled devices, be...	A
2	B096MSW6CT	Sounce Fast Phone Charging Cable & Data Sync U...	Computers&Accessories|Accessories&Peripherals|...	₹199	₹1,899	90%	3.9	7,928	【Fast Charger& Data Sync】-With built-in safet...	A
3	B08HDJ86NZ	boAt Deuce USB 300 2 in 1 Type-C & Micro USB	Computers&Accessories|Accessories&Peripherals|...	₹329	₹699	53%	4.2	94,363	The boAt Deuce USB 300 2 in 1 cable is	A

Let's see the column names

In [5]:
```
df.columns
```

Out[5]:
```
Index(['product_id', 'product_name', 'category', 'discounted_price',
       'actual_price', 'discount_percentage', 'rating', 'rating_count',
       'about_product', 'user_id', 'user_name', 'review_id', 'review_title',
       'review_content', 'img_link', 'product_link'],
      dtype='object')
```

Let's have a look on the shape of the dataset

In [6]:
```
print(f"The Number of Rows are {df.shape[0]}, and columns are {df.shape[1]}.")
```

```
The Number of Rows are 1465, and columns are 16.
```

Let's have a look on the columns and their data types using detailed info function

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1465 entries, 0 to 1464
Data columns (total 16 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   product_id           1465 non-null   object
 1   product_name         1465 non-null   object
 2   category             1465 non-null   object
 3   discounted_price     1465 non-null   object
 4   actual_price         1465 non-null   object
 5   discount_percentage  1465 non-null   object
 6   rating               1465 non-null   object
 7   rating_count         1463 non-null   object
 8   about_product        1465 non-null   object
 9   user_id              1465 non-null   object
 10  user_name            1465 non-null   object
 11  review_id            1465 non-null   object
```

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
product_id             0
product_name           0
category               0
discounted_price       0
actual_price           0
discount_percentage    0
rating                 0
rating_count           2
about_product          0
user_id                0
user_name              0
review_id              0
review_title           0
review_content         0
img_link               0
product_link           0
dtype: int64
```

Observation Set 1

- ***There are 1465 rows and 16 columns in the dataset.***
- ***The data type of all columns is object.***
- ***The columns in the datasets are:***
  - ***'product_id', 'product_name', 'category', 'discounted_price', 'actual_price', 'discount_percentage', 'rating', 'rating_count', 'about_product', 'user_id', 'user_name', 'review_id', 'review_title', 'review_content', 'img_link', 'product_link'***
- ***There are a few missing values in the dataset, which we will read in detail and deal with later on in the notebook.***

Changing Data Types of Columns from object to float

In [9]:

```
# Changing the data type of discounted price and actual price

df['discounted_price'] = df['discounted_price'].str.replace("₹",'')
df['discounted_price'] = df['discounted_price'].str.replace(",",'')
df['discounted_price'] = df['discounted_price'].astype('float64')

df['actual_price'] = df['actual_price'].str.replace("₹",'')
df['actual_price'] = df['actual_price'].str.replace(",",'')
df['actual_price'] = df['actual_price'].astype('float64')
```

In [10]:

```
# Changing Datatype and values in Discount Percentage

df['discount_percentage'] = df['discount_percentage'].str.replace('%','').astype('float64')

df['discount_percentage'] = df['discount_percentage'] / 100
```

In [11]:

```
# Finding unusual string in rating column
df['rating'].value_counts()
```

Out[11]:

```
rating
4.1    244
4.3    230
4.2    228
4.0    129
3.9    123
4.4    123
3.8     86
4.5     75
4       52
```

In [29]:

```
# Plot actual_price vs. rating
plt.scatter(df['actual_price'], df['rating'])
plt.xlabel('Actual_price')
plt.ylabel('Rating')
plt.show()
```