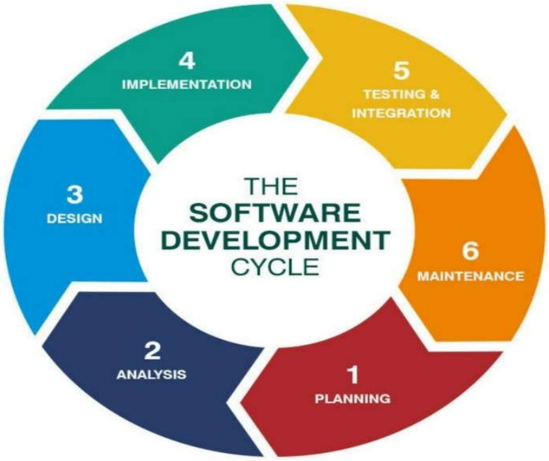
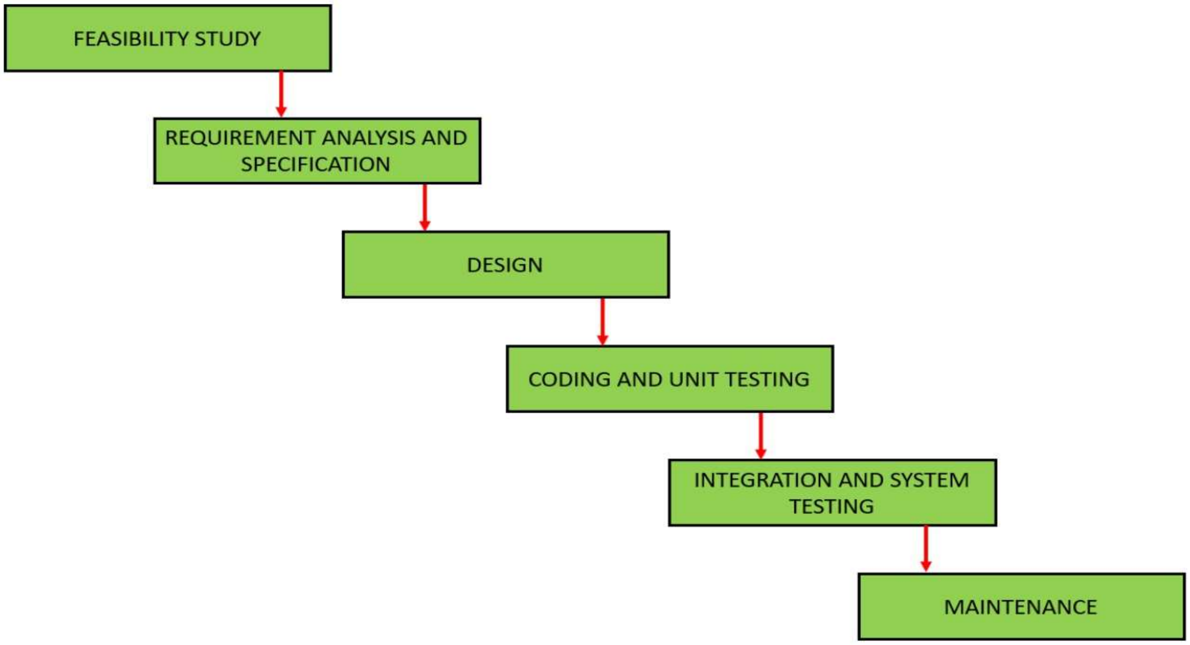


Here is a general overview of the typical stages in the SDLC:

1. **Planning:**
   - Define the project scope, objectives, and requirements.
   - Identify constraints, risks, and resources.
   - Develop a project plan outlining timelines, milestones, and deliverables.
2. **Feasibility Study:**
   - Evaluate the technical, economic, and operational feasibility of the project.
   - Assess potential risks and challenges.
   - Decide whether to proceed with the project or not.
3. **System Design:**
   - Create a high-level design of the system architecture.
   - Specify system components and their relationships.
   - Define data structures, interfaces, and algorithms.
4. **Implementation (Coding):**
   - Write code based on the detailed design specifications.
   - Follow coding standards and best practices.
   - Conduct code reviews to ensure quality and consistency.
5. **Testing:**
   - Develop and execute test cases to ensure the software meets requirements.
   - Identify and fix bugs and issues.
   - Perform various testing types, such as unit testing, integration testing, system testing, and user acceptance testing.
6. **Deployment:**
   - Release the software to the production environment.
   - Ensure a smooth transition from development to production.
   - Provide user training and support.
7. **Maintenance and Support:**
   - Address issues and bugs discovered post-deployment.
   - Make updates and enhancements based on user feedback.
   - Provide ongoing support and maintenance.

It's important to note that these stages can be executed in a sequential manner (as in the Waterfall model) or iteratively and incrementally (as in Agile methodologies). Some common SDLC models include:

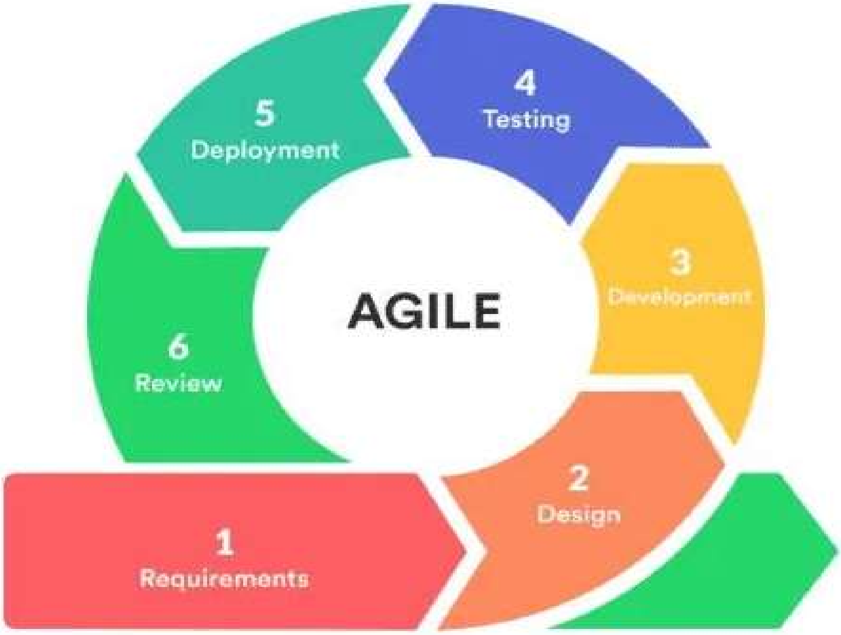
1. **Waterfall Model :**
   The Waterfall Model is a classical software development methodology. It was first introduced by Winston W. Royce in 1970. It is a linear and sequential approach to software development that consists of several phases.


FEASIBILITY STUDY
REQUIREMENT ANALYSIS AND SPECIFICATION
DESIGN
CODING AND UNIT TESTING
INTEGRATION AND SYSTEM TESTING
MAINTENANCE

- In the waterfall model, once a phase seems to be completed, it cannot be changed, and due to this less flexible nature, the waterfall model is not in practice anymore.

2. Agile Model :

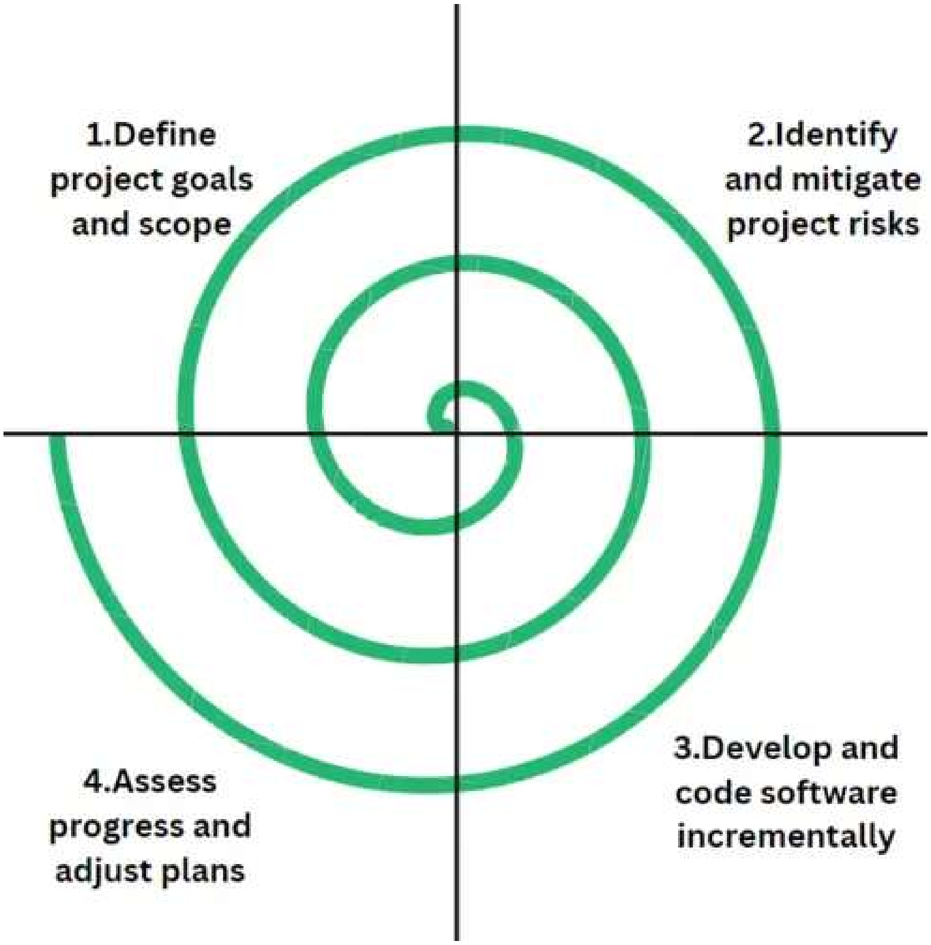
In earlier days, the **Iterative Waterfall Model** was very popular for completing a project. But nowadays, developers face various problems while using it to develop software.



The Agile Model was primarily designed to help a project adapt quickly to change requests. So, the main aim of the Agile model is to facilitate quick project completion. To accomplish this task, agility is required. Agility is achieved by fitting the process to the project and removing activities that may not be essential for a specific project.

3. Spiral Model :

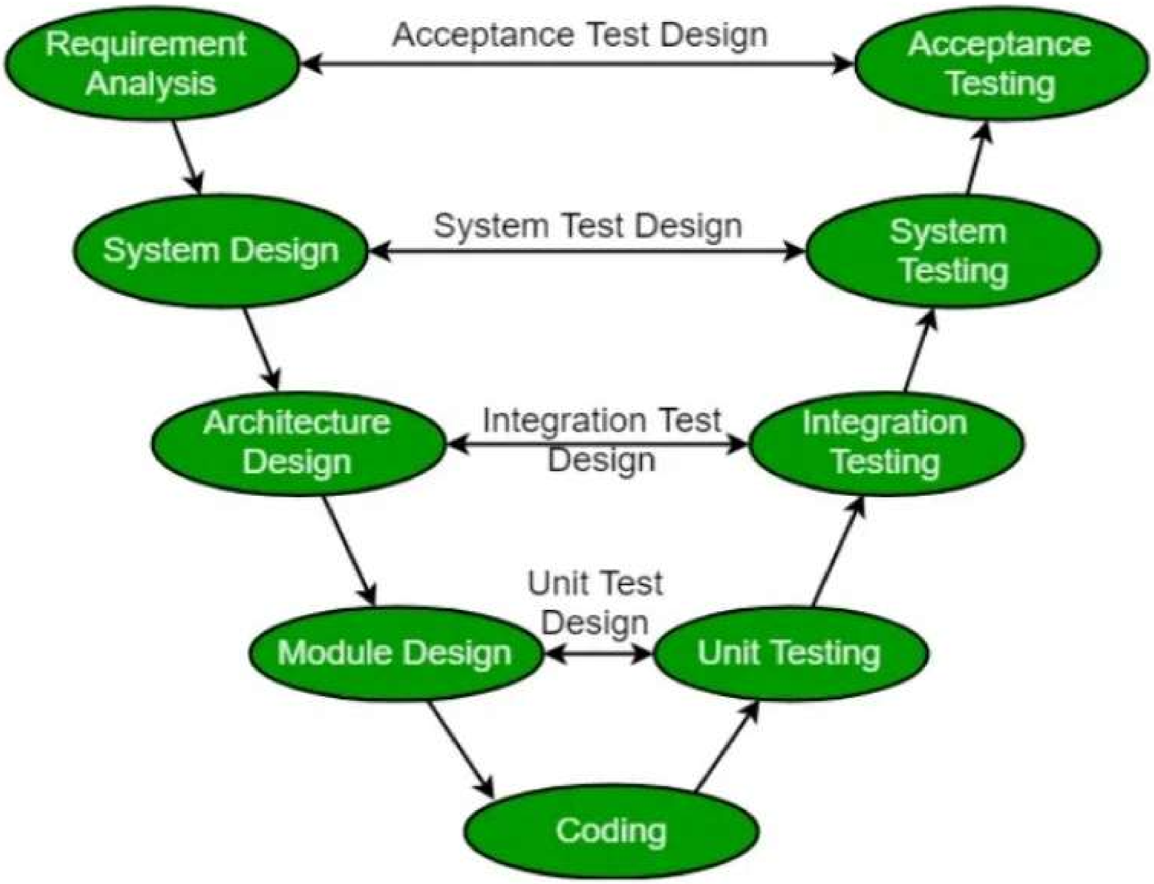
The Spiral Model is one of the most important Software Development Life Cycle models. The Spiral Model is a combination of the waterfall model and the iterative model. It provides support for Risk Handling. The Spiral Model was first proposed by Barry Boehm. This article focuses on discussing the Spiral Model in detail


1.Define project goals and scope
2.Identify and mitigate project risks
3.Develop and code software incrementally
4.Assess progress and adjust plans

- The exact number of phases needed to develop the product can be varied by the project manager depending upon the project risks.
- As the project manager dynamically determines the number of phases, the project manager has an important role in developing a product using the spiral model.

5. V-Shaped Model

- The V-shaped model is executed in a sequential manner in V-shape. Each stage or phase of this model is integrated with a testing phase.



- After every development phase, a testing phase is associated with it, and the next phase will start once the previous phase is completed, i.e., development & testing. It is also known as the verification or validation model.

Scenario :

Book Store Sales Analysis and Visualizations Using Python

Libraries used [Pandas, Numpy, Matplotlib, Seaborn]

Introduction

The **Bookstore Sales Analysis** project aims to explore and analyze the sales data of a bookstore to gain insights into sales performance, trends, and customer behavior. By leveraging data analysis techniques, this project will help identify key factors affecting sales, recognize patterns in purchasing habits, and provide recommendations for optimizing bookstore operations.

Using Python libraries such as pandas, NumPy, and visualization tools like Matplotlib and Seaborn, we will clean, process, and visualize the data to uncover trends, evaluate performance, and ultimately make data-driven decisions for improving store profitability.

The dataset used for this analysis was downloaded from ChatGPT. The dataset is a CSV file and it contains comprehensive information about book store sales on below columns User_ID, Cust_Name, Product_ID, Age group, Age, Marital status, State, Zone, Occupation, Books, Orders, Amount.

How to perform Book Store analysis?

It will provide future impact on a books store to analyze a data about readers and to come at a decisions which books are make our profit and who are like to read which genre's book

We also Does a Exploratary Data Analysis to analyze a data on various basis stratergies to analyze a data. For Ex. which states people has a mainly prefers a which genre's book. It provide a competitive advantage that you can't afford to miss out.

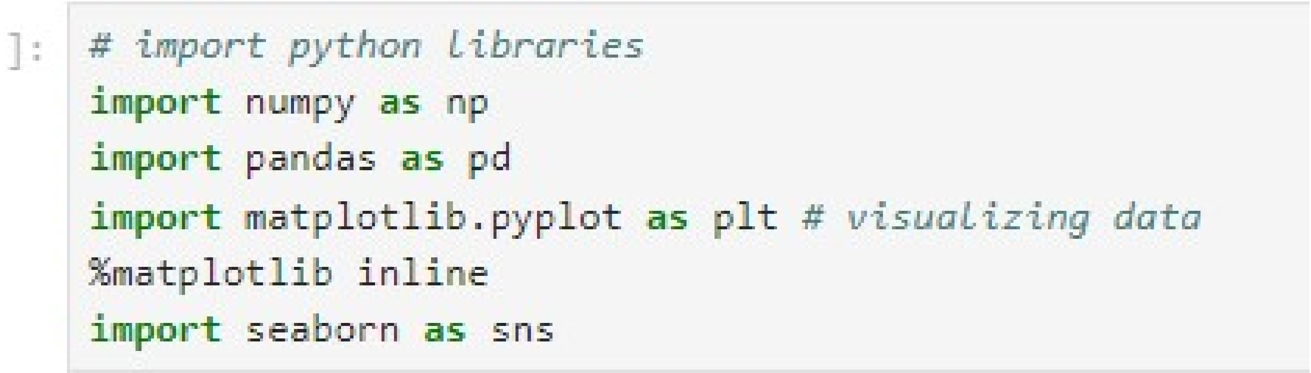
- **Overview**

Book store sales analysis refers to the systematic evaluation of sales data from a bookstore to understand patterns, trends, and insights that can inform decision-making. By using Python, a popular language for data analysis and visualization, we can extract meaningful insights from raw sales data, such as which genres perform best, seasonal trends, top-selling books, customer behavior, etc.

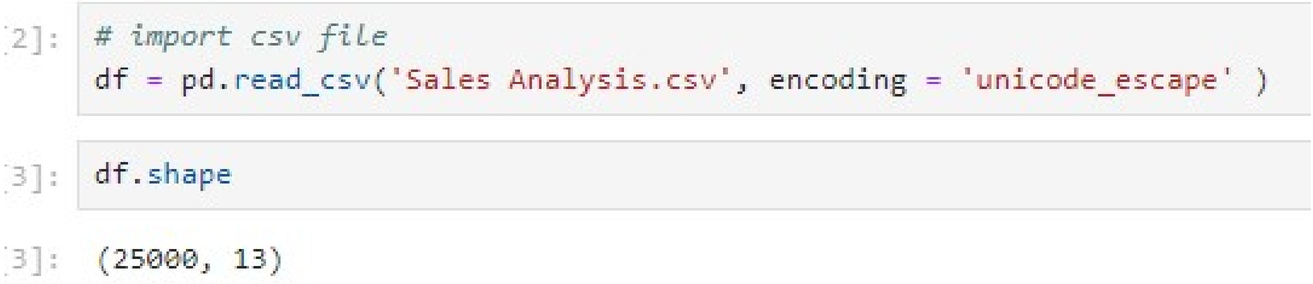
EDA involves using summary statistics and visualizations to explore the data.

Visualization is key to understanding patterns in the data and conveying insights effectively. Python libraries such as **Matplotlib**, **Seaborn**, and **Plotly** can be used to create compelling visuals.

Execution:



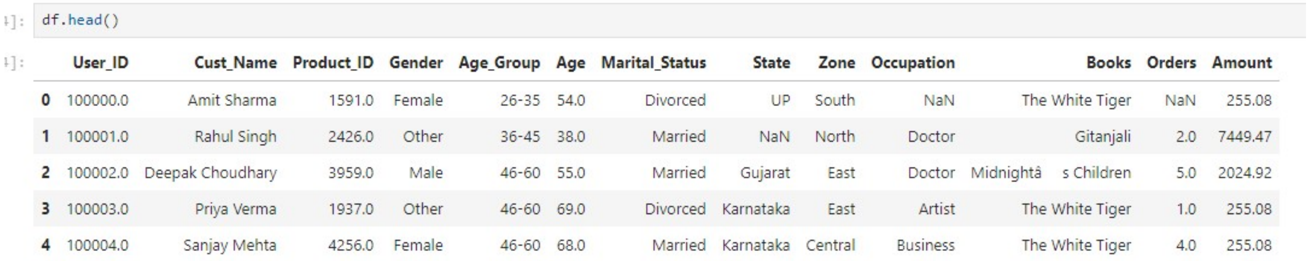
```
]: # import python libraries
   import numpy as np
   import pandas as pd
   import matplotlib.pyplot as plt # visualizing data
   %matplotlib inline
   import seaborn as sns
```



```
2]:  # import csv file
     df = pd.read_csv('Sales Analysis.csv', encoding = 'unicode_escape' )
```

```
3]:  df.shape
```

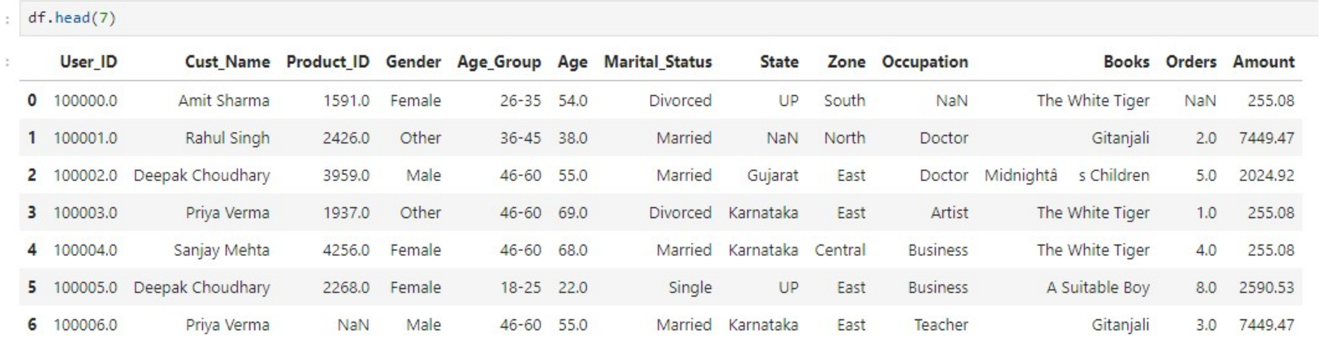
```
3]:  (25000, 13)
```



```
df.head()
```

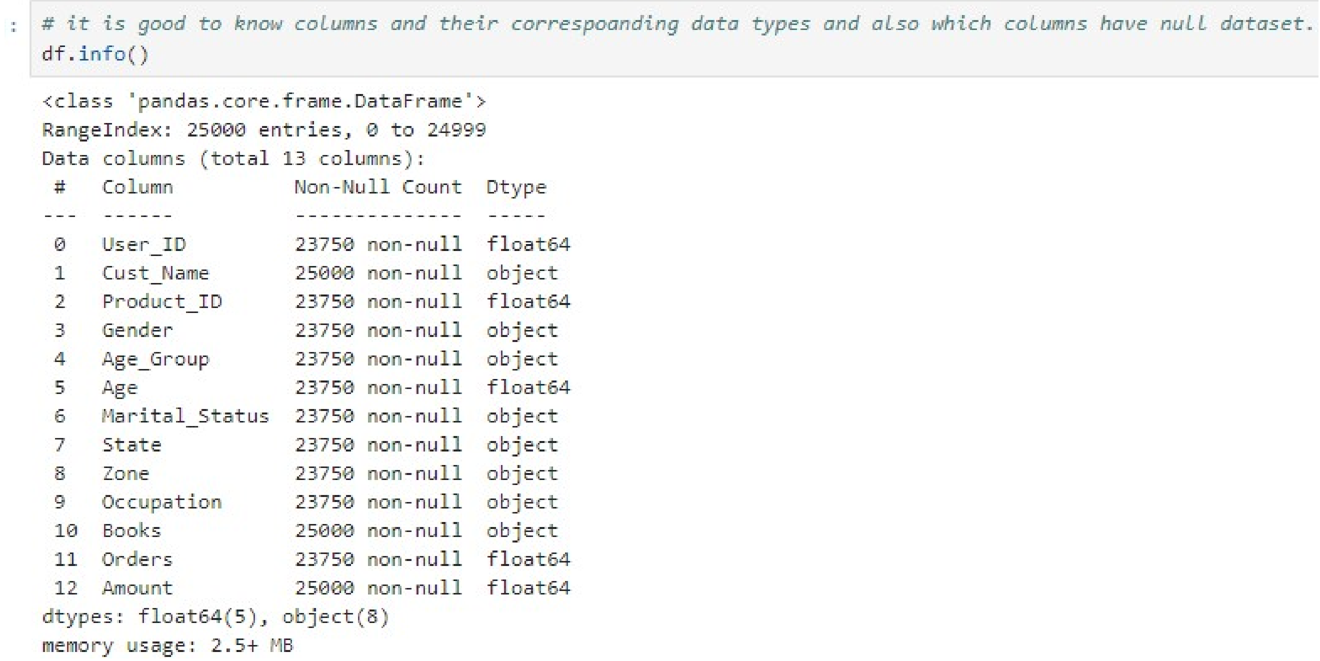
	User_ID	Cust_Name	Product_ID	Gender	Age_Group	Age	Marital_Status	State	Zone	Occupation	Books	Orders	Amount
0	100000.0	Amit Sharma	1591.0	Female	26-35	54.0	Divorced	UP	South	NaN	The White Tiger	NaN	255.08
1	100001.0	Rahul Singh	2426.0	Other	36-45	38.0	Married	NaN	North	Doctor	Gitanjali	2.0	7449.47
2	100002.0	Deepak Choudhary	3959.0	Male	46-60	55.0	Married	Gujarat	East	Doctor	Midnightâ s Children	5.0	2024.92
3	100003.0	Priya Verma	1937.0	Other	46-60	69.0	Divorced	Karnataka	East	Artist	The White Tiger	1.0	255.08
4	100004.0	Sanjay Mehta	4256.0	Female	46-60	68.0	Married	Karnataka	Central	Business	The White Tiger	4.0	255.08

#We Also Have a df.tail() to describe a data's last 5 rows.
#We also define the rows in brackets to gate a more data.



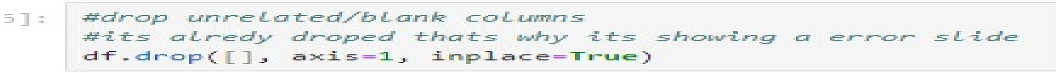
```
df.head(7)
```

	User_ID	Cust_Name	Product_ID	Gender	Age_Group	Age	Marital_Status	State	Zone	Occupation	Books	Orders	Amount
0	100000.0	Amit Sharma	1591.0	Female	26-35	54.0	Divorced	UP	South	NaN	The White Tiger	NaN	255.08
1	100001.0	Rahul Singh	2426.0	Other	36-45	38.0	Married	NaN	North	Doctor	Gitanjali	2.0	7449.47
2	100002.0	Deepak Choudhary	3959.0	Male	46-60	55.0	Married	Gujarat	East	Doctor	Midnightâ s Children	5.0	2024.92
3	100003.0	Priya Verma	1937.0	Other	46-60	69.0	Divorced	Karnataka	East	Artist	The White Tiger	1.0	255.08
4	100004.0	Sanjay Mehta	4256.0	Female	46-60	68.0	Married	Karnataka	Central	Business	The White Tiger	4.0	255.08
5	100005.0	Deepak Choudhary	2268.0	Female	18-25	22.0	Single	UP	East	Business	A Suitable Boy	8.0	2590.53
6	100006.0	Priya Verma	NaN	Male	46-60	55.0	Married	Karnataka	East	Teacher	Gitanjali	3.0	7449.47

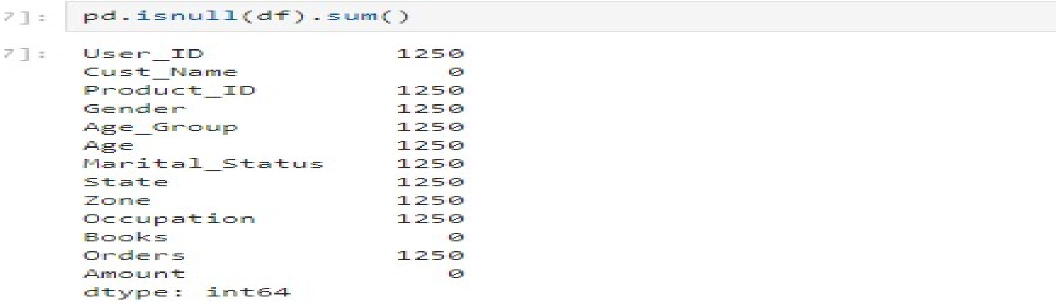


```
# it is good to know columns and their correspoanding data types and also which columns have null dataset.
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25000 entries, 0 to 24999
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   User_ID         23750 non-null  float64
 1   Cust_Name       25000 non-null  object 
 2   Product_ID      23750 non-null  float64
 3   Gender          23750 non-null  object 
 4   Age_Group       23750 non-null  object 
 5   Age             23750 non-null  float64
 6   Marital_Status  23750 non-null  object 
 7   State           23750 non-null  object 
 8   Zone            23750 non-null  object 
 9   Occupation      23750 non-null  object 
 10  Books           25000 non-null  object 
 11  Orders          23750 non-null  float64
 12  Amount          25000 non-null  float64
dtypes: float64(5), object(8)
memory usage: 2.5+ MB
```

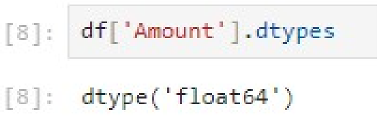


```
#drop unrelated/blank columns
#its alredy droped thats why its showing a error slide
df.drop([], axis=1, inplace=True)
```



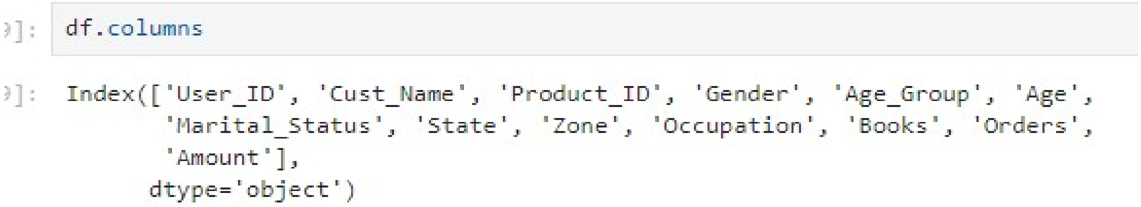
```
pd.isnull(df).sum()
```

```
User_ID           1250
Cust_Name            0
Product_ID        1250
Gender            1250
Age_Group         1250
Age               1250
Marital_Status    1250
State             1250
Zone              1250
Occupation        1250
Books                0
Orders            1250
Amount               0
dtype: int64
```



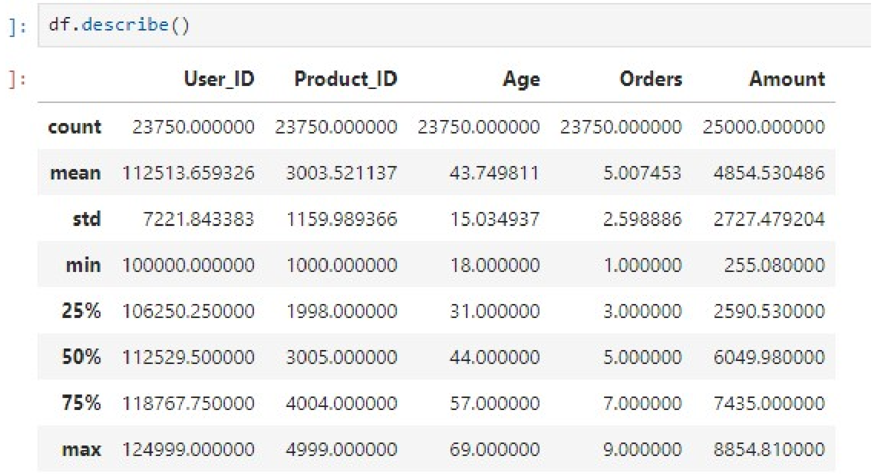
```
[8]:  df['Amount'].dtypes

[8]:  dtype('float64')
```

#To know the columns without going to data base we also use following technique



```
]:  df.columns

]:  Index(['User_ID', 'Cust_Name', 'Product_ID', 'Gender', 'Age_Group', 'Age',
           'Marital_Status', 'State', 'Zone', 'Occupation', 'Books', 'Orders',
           'Amount'],
          dtype='object')
```

# to get a better understanding of the dataset,
# we can also see the statistical summary of the dataset.

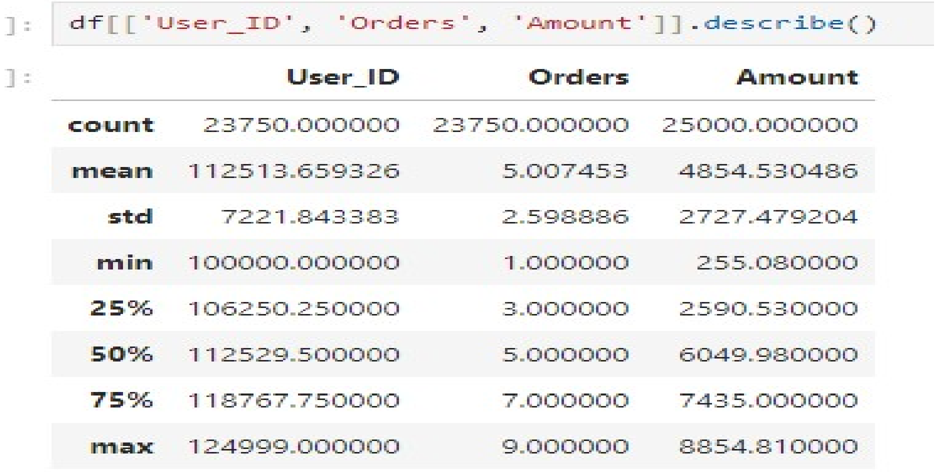


```
]:  df.describe()
```

]:

	User_ID	Product_ID	Age	Orders	Amount
count	23750.000000	23750.000000	23750.000000	23750.000000	25000.000000
mean	112513.659326	3003.521137	43.749811	5.007453	4854.530486
std	7221.843383	1159.989366	15.034937	2.598886	2727.479204
min	100000.000000	1000.000000	18.000000	1.000000	255.080000
25%	106250.250000	1998.000000	31.000000	3.000000	2590.530000
50%	112529.500000	3005.000000	44.000000	5.000000	6049.980000
75%	118767.750000	4004.000000	57.000000	7.000000	7435.000000
max	124999.000000	4999.000000	69.000000	9.000000	8854.810000

#If we want to use describe for specified columns then also use like following.



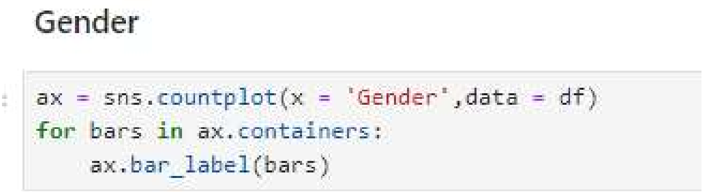
```
]:  df[['User_ID', 'Orders', 'Amount']].describe()
```

]:

	User_ID	Orders	Amount
count	23750.000000	23750.000000	25000.000000
mean	112513.659326	5.007453	4854.530486
std	7221.843383	2.598886	2727.479204
min	100000.000000	1.000000	255.080000
25%	106250.250000	3.000000	2590.530000
50%	112529.500000	5.000000	6049.980000
75%	118767.750000	7.000000	7435.000000
max	124999.000000	9.000000	8854.810000

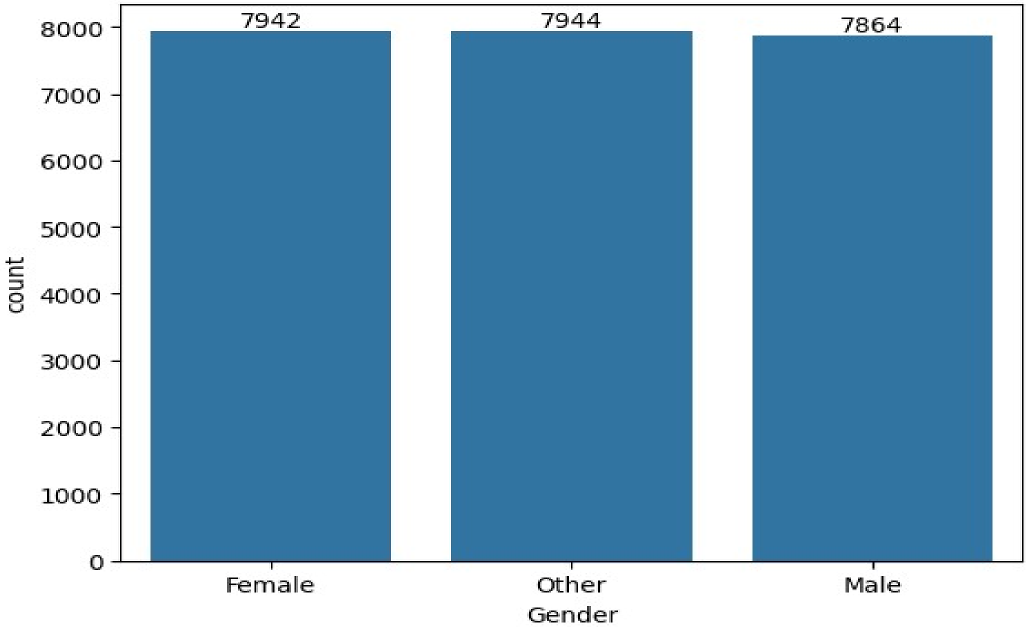
Finding Answers with the Data Using Visualizations.

i. Gender count?

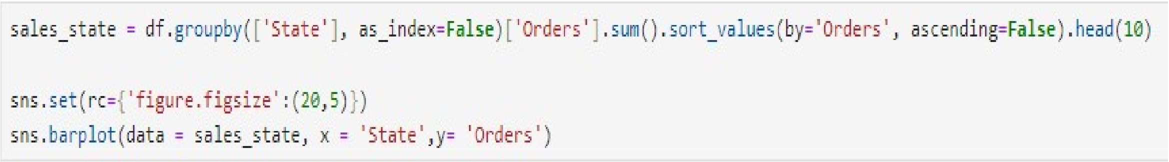


Gender

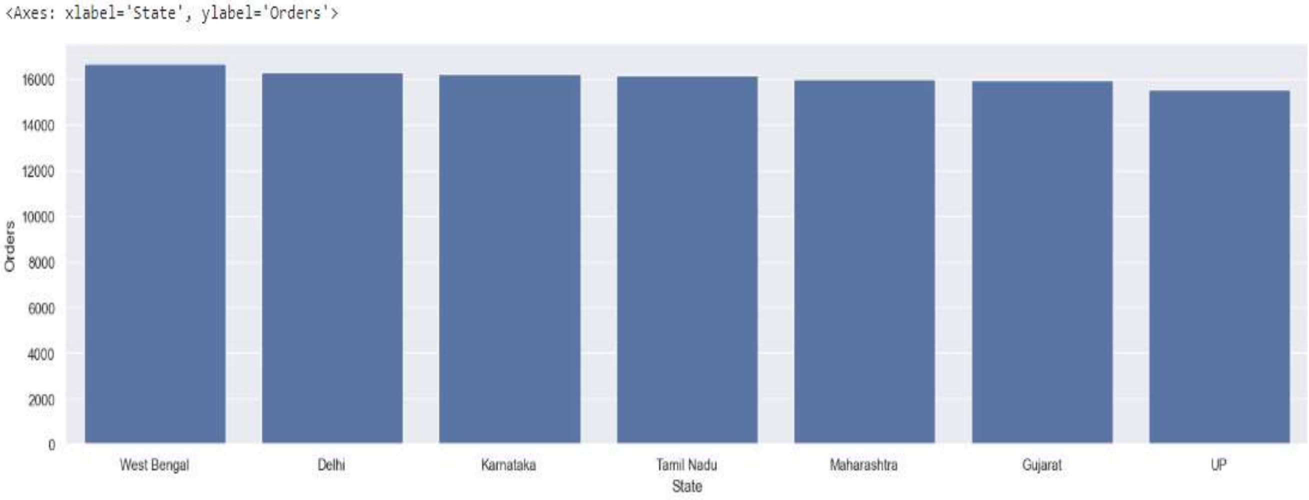
```
ax = sns.countplot(x = 'Gender',data = df)
for bars in ax.containers:
    ax.bar_label(bars)
```



ii. How many orders we have based on states for Books?

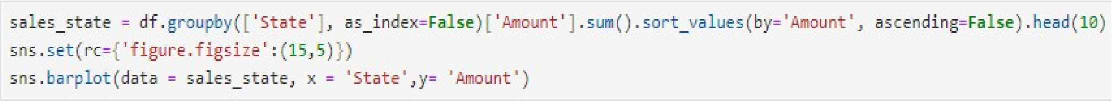


```
sales_state = df.groupby(['State'], as_index=False)['Orders'].sum().sort_values(by='Orders', ascending=False).head(10)

sns.set(rc={'figure.figsize':(20,5)})
sns.barplot(data = sales_state, x = 'State',y= 'Orders')
```



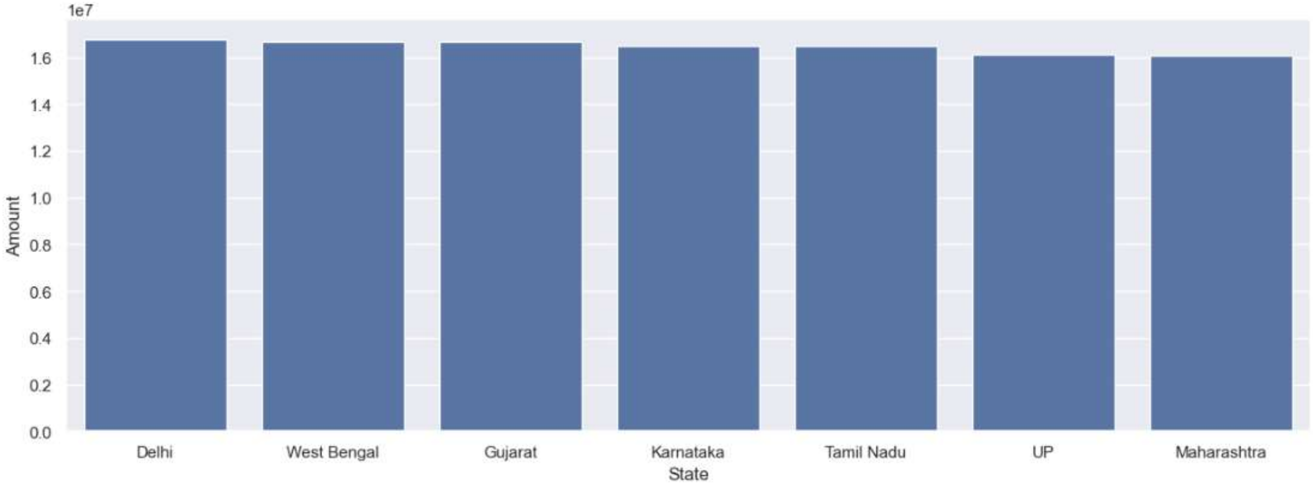
```
<Axes: xlabel='State', ylabel='Orders'>
```

iii. How many states gives us a more revenue (Amount)?

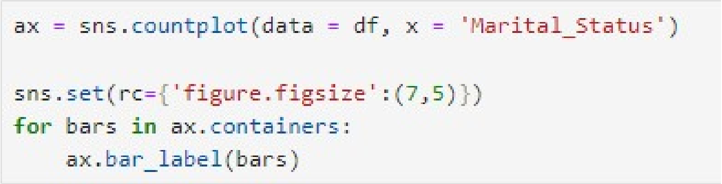


```
sales_state = df.groupby(['State'], as_index=False)['Amount'].sum().sort_values(by='Amount', ascending=False).head(10)
sns.set(rc={'figure.figsize':(15,5)})
sns.barplot(data = sales_state, x = 'State',y= 'Amount')
```

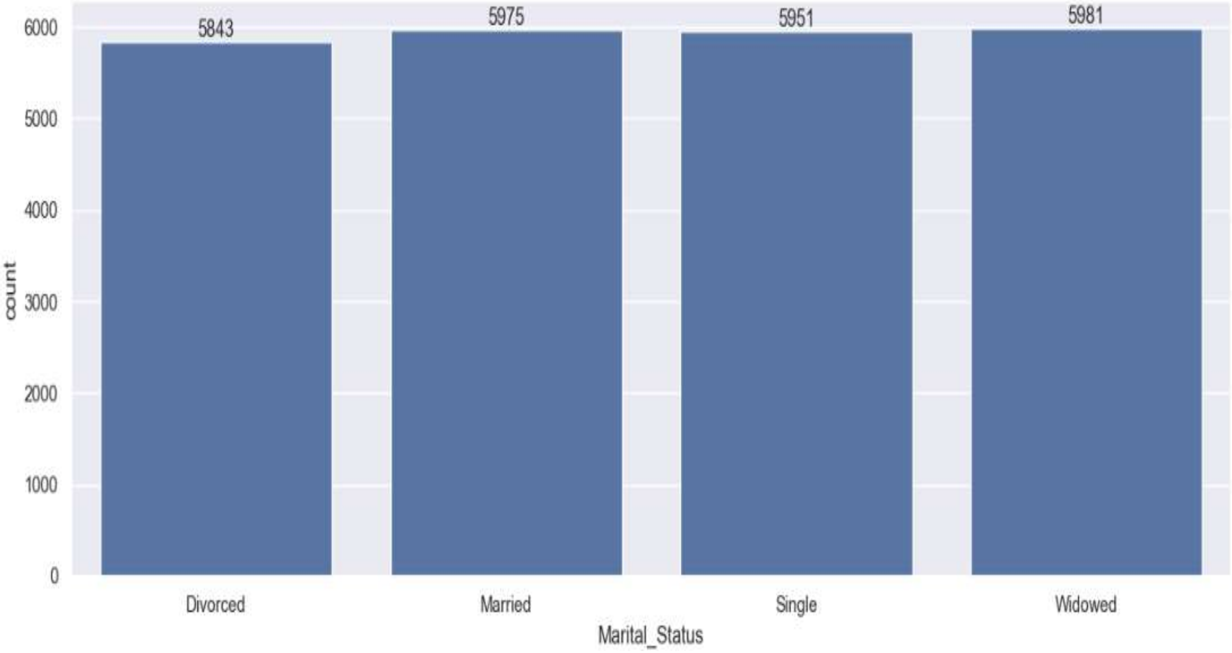
```
<Axes: xlabel='State', ylabel='Amount'>
```



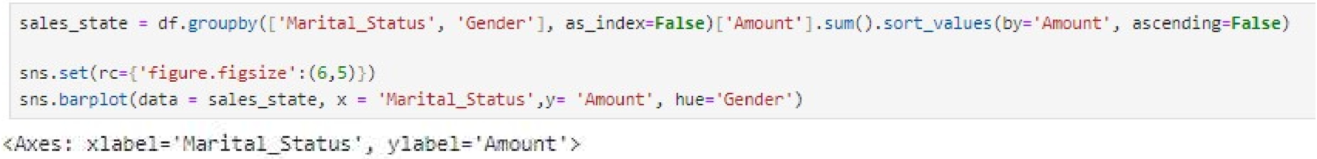
iv. Marital Status of a readers?



```
ax = sns.countplot(data = df, x = 'Marital_Status')

sns.set(rc={'figure.figsize':(7,5)})
for bars in ax.containers:
    ax.bar_label(bars)
```

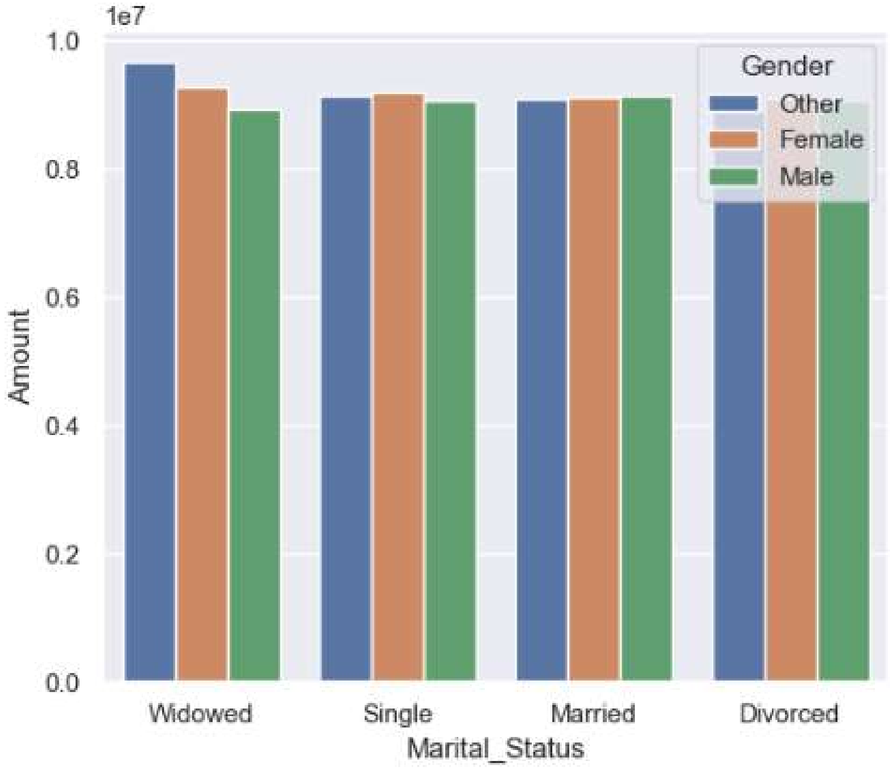


v. Marital status based on gender.

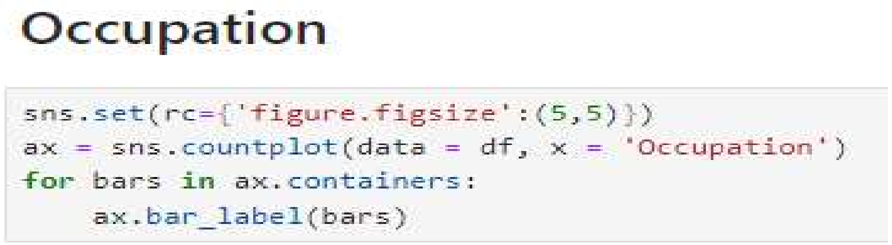


```
sales_state = df.groupby(['Marital_Status', 'Gender'], as_index=False)['Amount'].sum().sort_values(by='Amount', ascending=False)

sns.set(rc={'figure.figsize':(6,5)})
sns.barplot(data = sales_state, x = 'Marital_Status',y= 'Amount', hue='Gender')
```

```
<Axes: xlabel='Marital_Status', ylabel='Amount'>
```

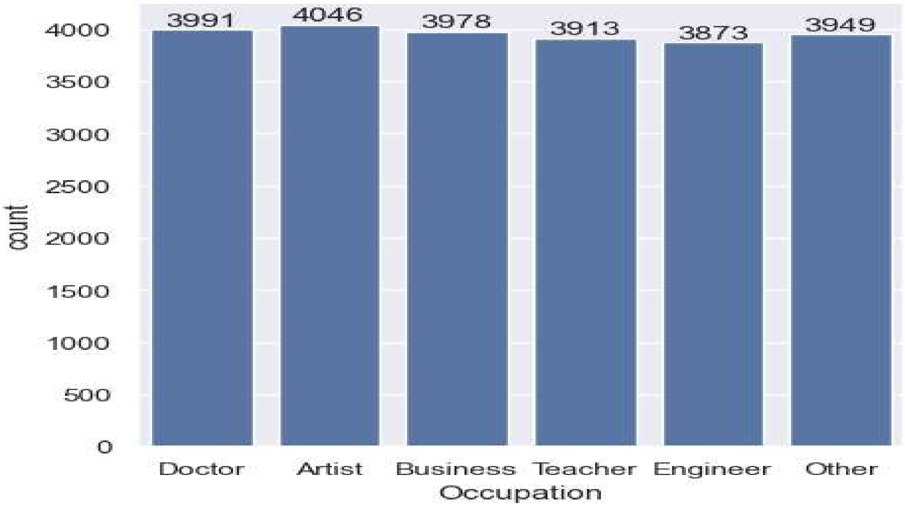


vi. Occupation of a readers.

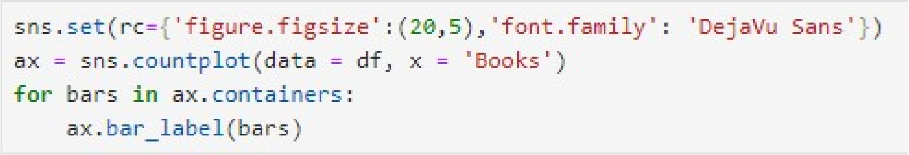


Occupation

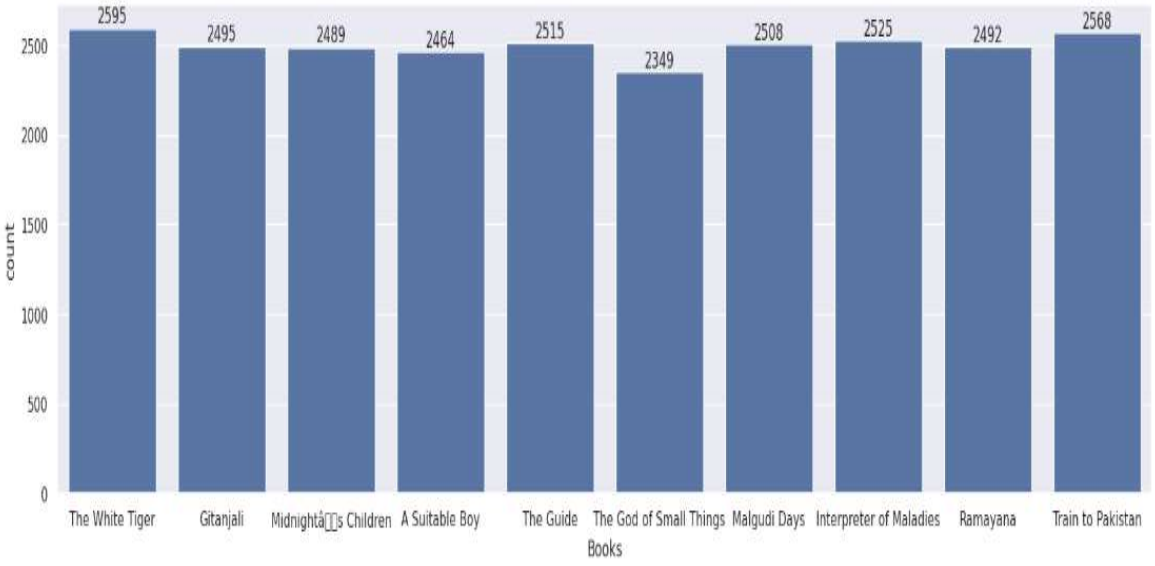
```
sns.set(rc={'figure.figsize':(5,5)})
ax = sns.countplot(data = df, x = 'Occupation')
for bars in ax.containers:
    ax.bar_label(bars)
```



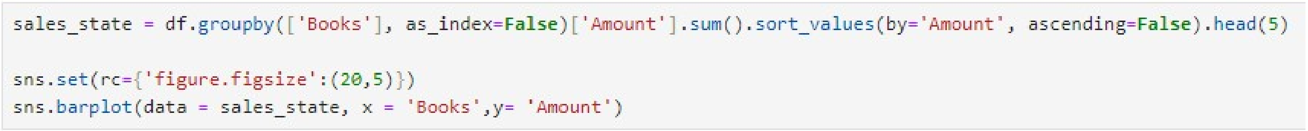
vii. How many books we have in store?



```
sns.set(rc={'figure.figsize':(20,5),'font.family': 'DejaVu Sans'})
ax = sns.countplot(data = df, x = 'Books')
for bars in ax.containers:
    ax.bar_label(bars)
```

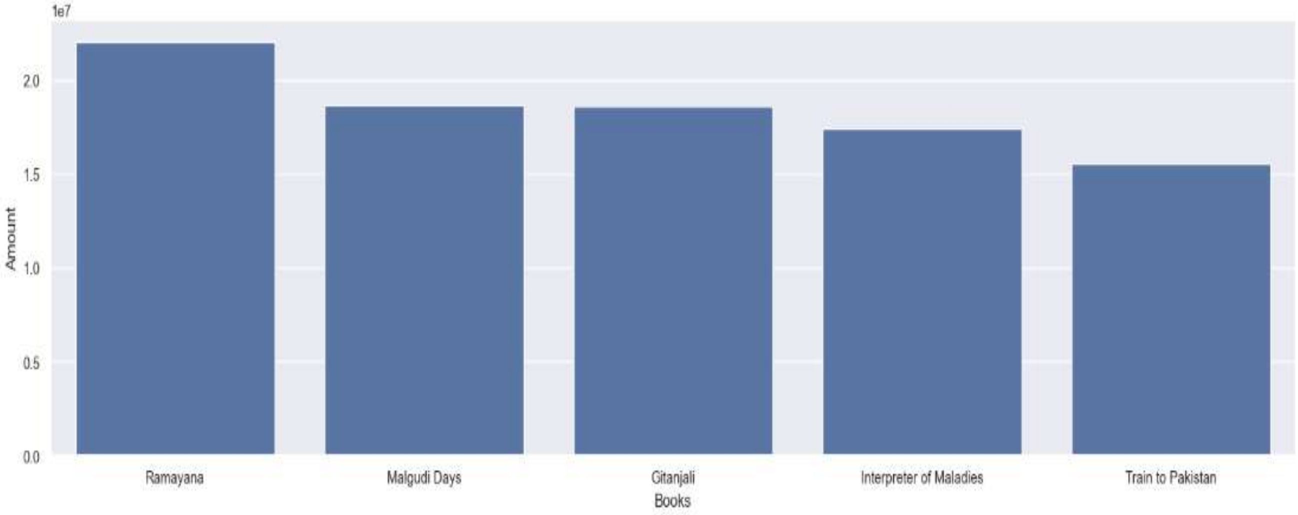


viii. Which books gives us a more revenue?

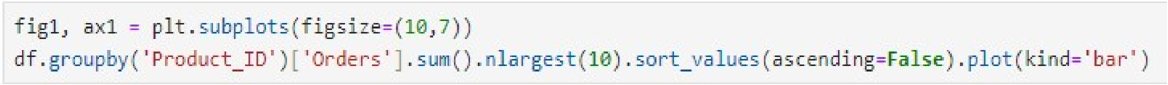


```
sales_state = df.groupby(['Books'], as_index=False)['Amount'].sum().sort_values(by='Amount', ascending=False).head(5)

sns.set(rc={'figure.figsize':(20,5)})
sns.barplot(data = sales_state, x = 'Books',y= 'Amount')
```

```
<Axes: xlabel='Books', ylabel='Amount'>
```

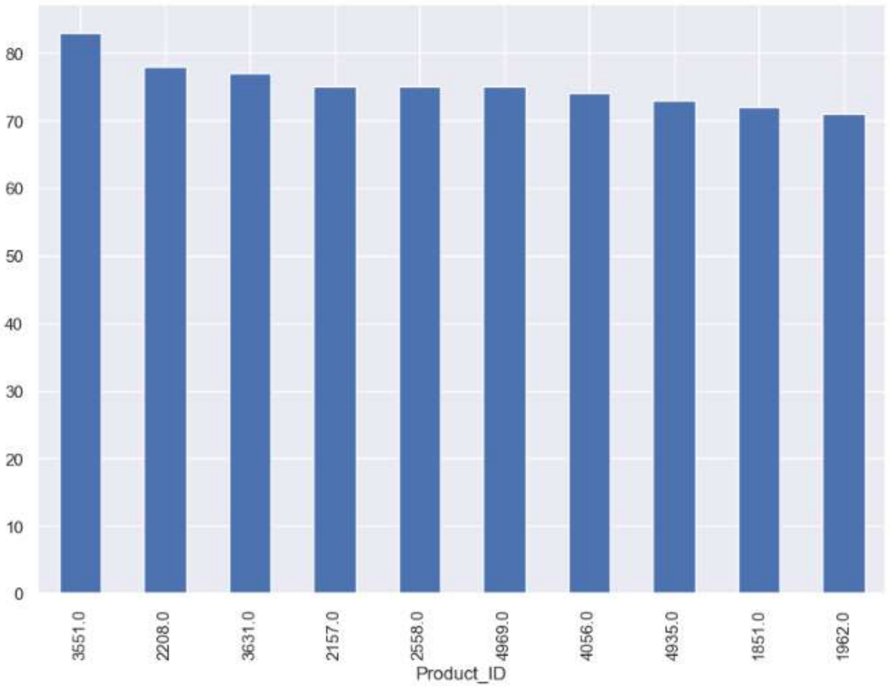


ix. Count of orders based on Product_ID.

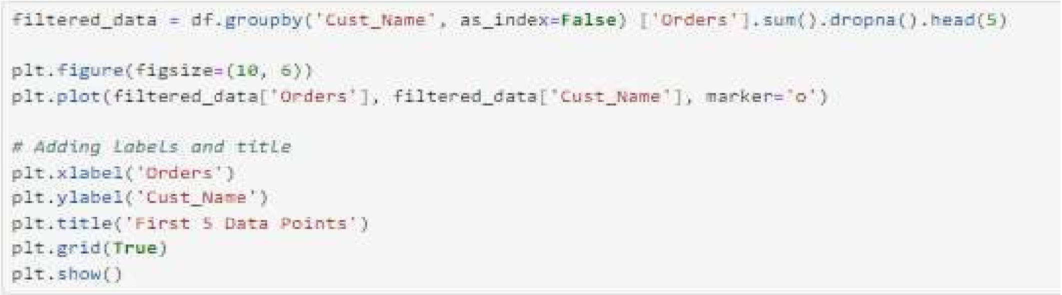


```
fig1, ax1 = plt.subplots(figsize=(10,7))
df.groupby('Product_ID')['Orders'].sum().nlargest(10).sort_values(ascending=False).plot(kind='bar')
```

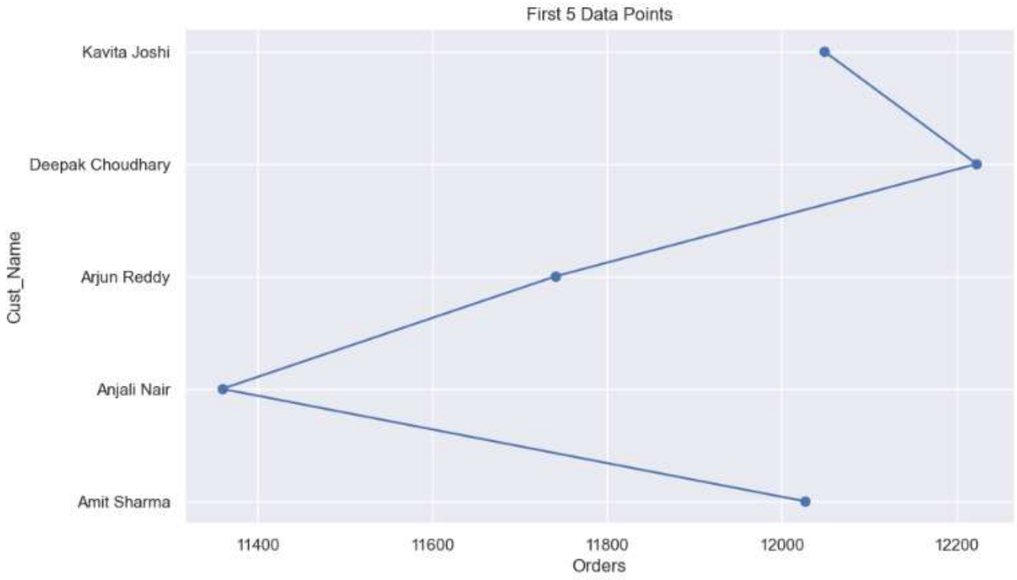
```
<Axes: xlabel='Product_ID'>
```



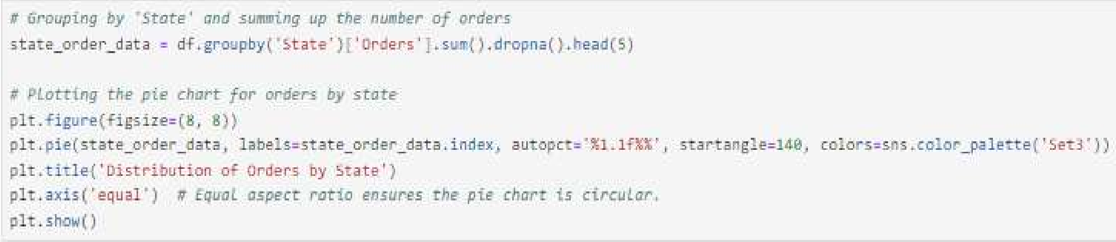
x. Line chart based on customer name and Orders?



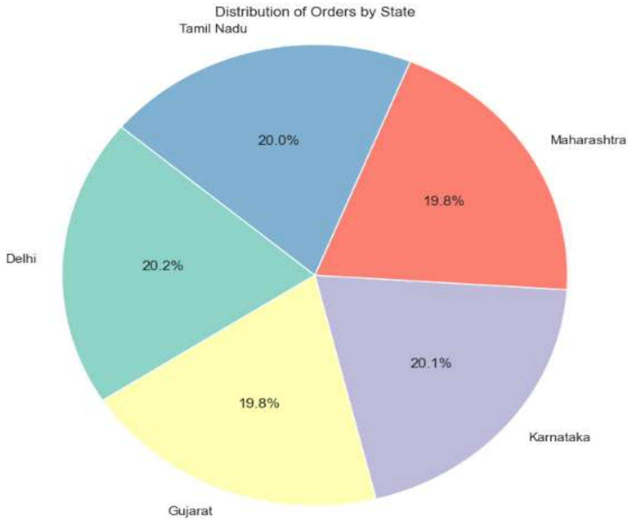
```
filtered_data = df.groupby('Cust_Name', as_index=False) ['Orders'].sum().dropna().head(5)

plt.figure(figsize=(10, 6))
plt.plot(filtered_data['Orders'], filtered_data['Cust_Name'], marker='o')

# Adding labels and title
plt.xlabel('Orders')
plt.ylabel('Cust_Name')
plt.title('First 5 Data Points')
plt.grid(True)
plt.show()
```



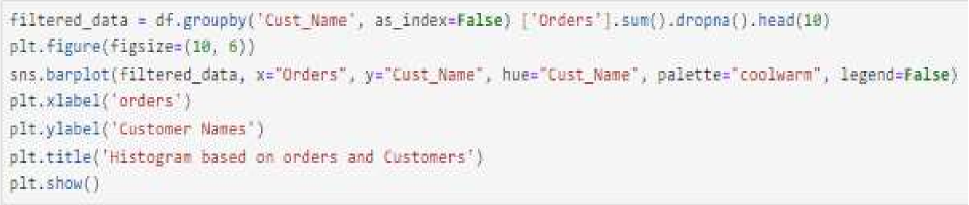
xi. Pie chart which shows the percent of orders in top states.



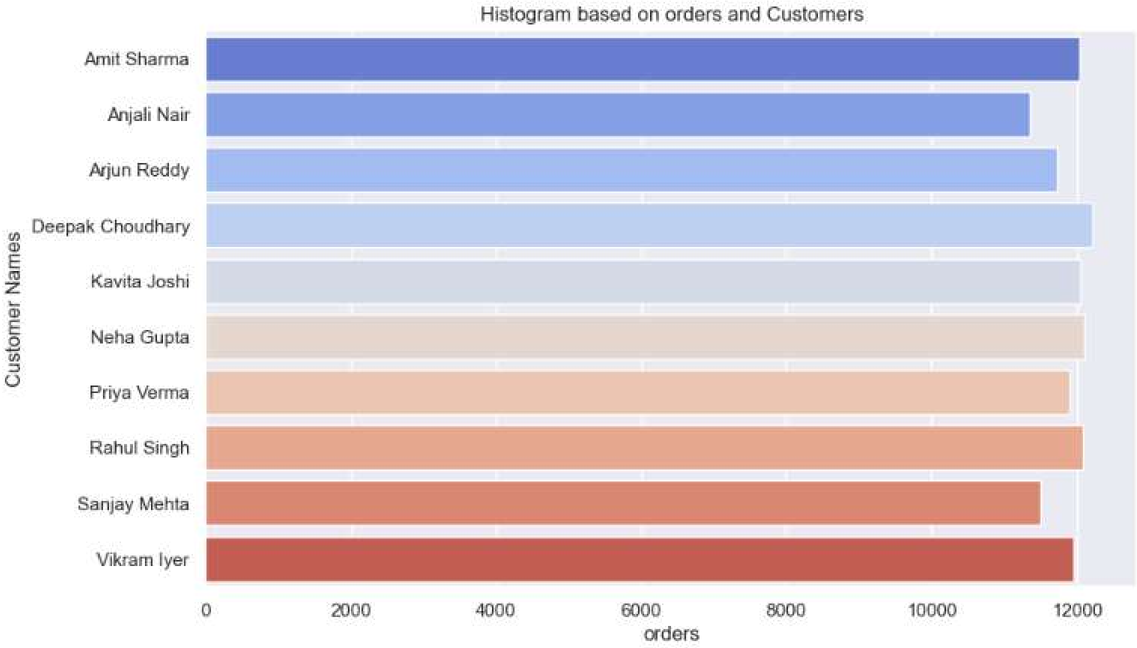
```
# Grouping by 'State' and summing up the number of orders
state_order_data = df.groupby('State')['Orders'].sum().dropna().head(5)

# Plotting the pie chart for orders by state
plt.figure(figsize=(8, 8))
plt.pie(state_order_data, labels=state_order_data.index, autopct='%1.1f%%', startangle=140, colors=sns.color_palette('Set3'))
plt.title('Distribution of Orders by State')
plt.axis('equal')  # Equal aspect ratio ensures the pie chart is circular.
plt.show()
```



xii. Barplot based on orders and customers



```
filtered_data = df.groupby('Cust_Name', as_index=False) ['Orders'].sum().dropna().head(10)
plt.figure(figsize=(10, 6))
sns.barplot(filtered_data, x="Orders", y="Cust_Name", hue="Cust_Name", palette="coolwarm", legend=False)
plt.xlabel('orders')
plt.ylabel('Customer Names')
plt.title('Histogram based on orders and Customers')
plt.show()
```


Histogram based on orders and Customers
Customer Names
Amit Sharma
Anjali Nair
Arjun Reddy
Deepak Choudhary
Kavita Joshi
Neha Gupta
Priya Verma
Rahul Singh
Sanjay Mehta
Vikram Iyer
orders