Ai\_phase-2

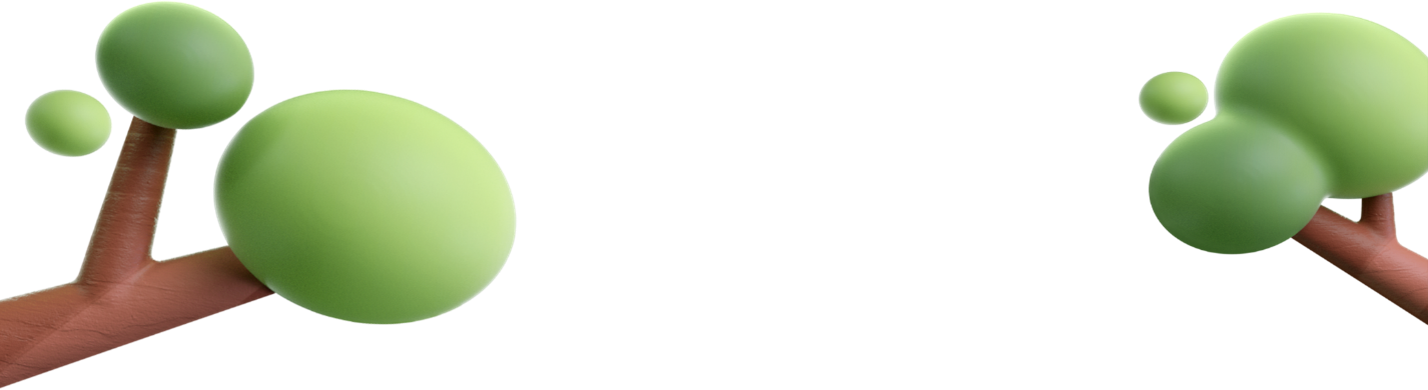
PROJECT:

MARKET BASKET ANALYSIS

SUBMITED BY:

A.ARUN PANDI,

BE-CSE,



Market Basket Analysis

Market Basket Analysis (MBA) is a data mining technique used in retail and e-commerce to uncover associations between products that are frequently purchased together by customers. It's based on the idea that if a customer buys one item, they are likely to buy related items as well. MBA helps retailers identify patterns and correlations in customer purchasing behavior, which can be valuable for several purposes:

**Product Recommendations:** MBA can be used to suggest complementary products to customers, increasing sales and improving the shopping experience. For example, if customers often buy pasta and pasta sauce together, a recommendation system might suggest both items when one is added to the cart.

**Inventory Management:** Retailers can optimize their inventory by ensuring that frequently associated products are stocked together. This reduces out-of-stock situations and minimizes storage costs.

**Pricing and Promotions:** MBA insights can guide pricing strategies and promotion planning. Retailers can offer discounts or promotions on bundles of products that are often bought together to increase sales.

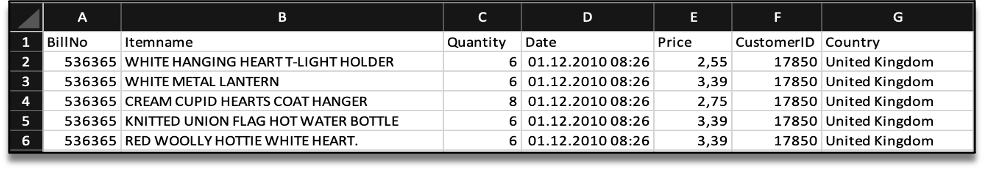
**Store Layout:** MBA can influence store layout design by placing related items closer together to encourage cross-selling. For example, placing wine near cheese and crackers

# Introduction

In the rapidly evolving landscape of retail, the strategic utilization of data-driven insights is not just advantageous; it's imperative for staying competitive and meeting the ever-shifting expectations of customers. Our design thinking process, meticulously crafted and executed, involves a comprehensive approach to retail analytics, leveraging the potent Apriori algorithm for association rule mining. This intricate framework navigates through the complexities of retail data, from its inception to the extraction of actionable business recommendations, ushering in a new era of strategic decision making.

# Dataset Description

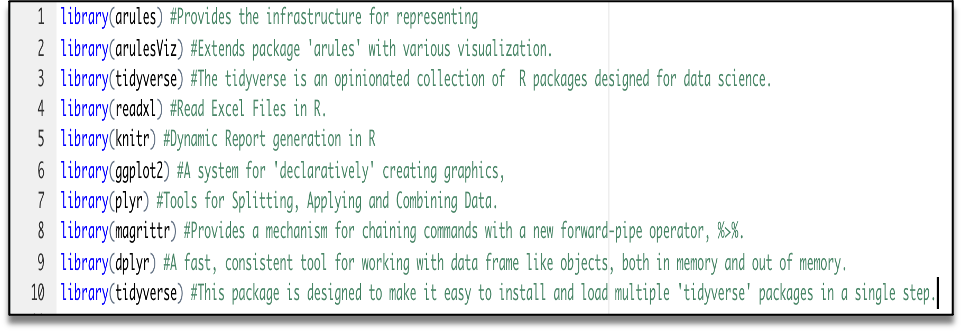
* File name: Data\_1
* List name: retaildata
* File format: . xlsx
* Number of Row: 522065
* Number of Attributes: 7
* BillNo: 6-digit number assigned to each transaction. Nominal.
* Itemname: Product name. Nominal.
* Quantity: The quantities of each product per transaction. Numeric.
* Date: The day and time when each transaction was generated. Numeric.
* Price: Product price. Numeric.
* CustomerID: 5-digit number assigned to each customer. Nominal.
* Country: Name of the country where each customer resides. Nominal.



# Libraries in R

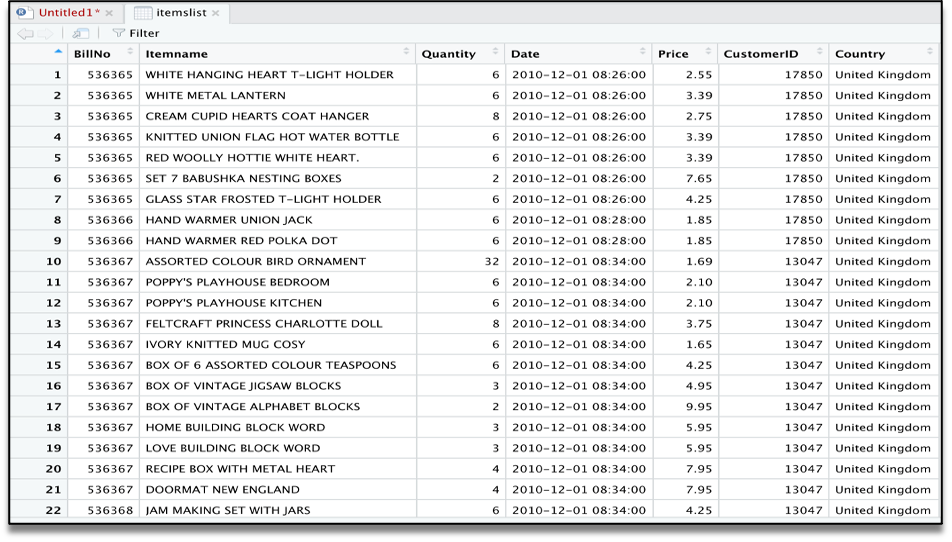
First, we need to load required libraries. Shortly I describe all libraries.

* arules - Provides the infrastructure for representing,  
  manipulating and analyzing transaction data and patterns (frequent itemsets and association rules).
* arulesViz - Extends package 'arules' with various visualization.  
  techniques for association rules and item-sets. The package also includes several interactive visualizations for rule exploration.
* tidyverse - The tidyverse is an opinionated collection of R packages designed for data science.
* readxl - Read Excel Files in R.
* plyr - Tools for Splitting, Applying and Combining Data.
* ggplot2 - A system for 'declaratively' creating graphics, based on "The Grammar of Graphics". You provide the data, tell 'ggplot2' how to map variables to aesthetics, what graphical primitives to use, and it takes care of the details.
* knitr - Dynamic Report generation in R.
* magrittr- Provides a mechanism for chaining commands with a new forward-pipe operator, %>%. This operator will forward a value, or the result of an expression, into the next function call/expression. There is flexible support for the type of right-hand side expressions.
* dplyr - A fast, consistent tool for working with data frame like objects, both in memory and out of memory.
* tidyverse - This package is designed to make it easy to install and load multiple 'tidyverse' packages in a single step.

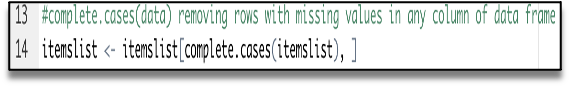


# Data Pre-processing

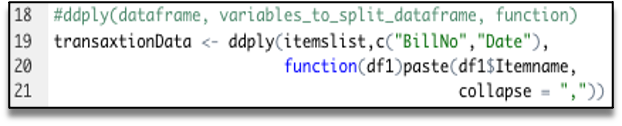
Next, we need to upload Assignment-1\_Data. xlsx to R to read the dataset.Now we can see our data in R.

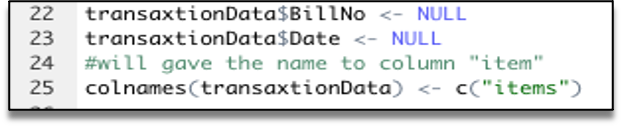
After we will clear our data frame, will remove missing values.



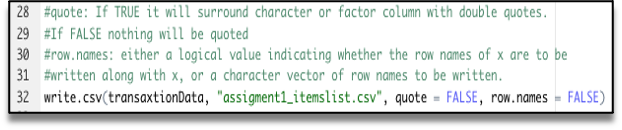
To apply Association Rule mining, we need to convert dataframe into transaction data to make all items that are bought together in one invoice will be in one row. Below lines of code will combine all products from one BillNo and Date and combine all products from that BillNo and Date as one row, with each item, separated by (,)

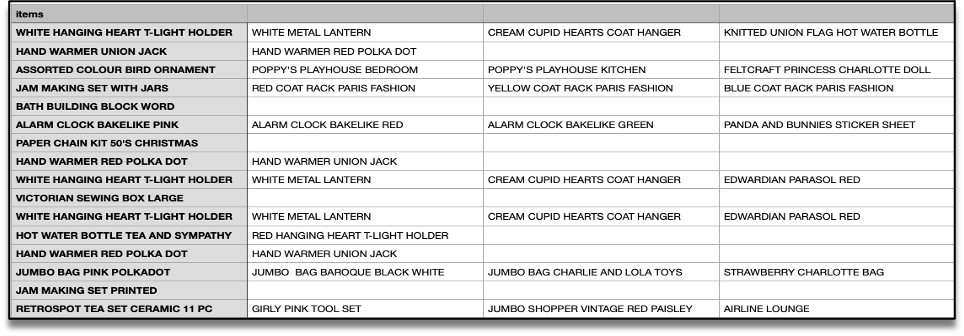


We don’t need BillNo and Date, we will make it as Null.  
Next, you have to store this transaction data into .csv



This how should look transaction data before we will go to next step.





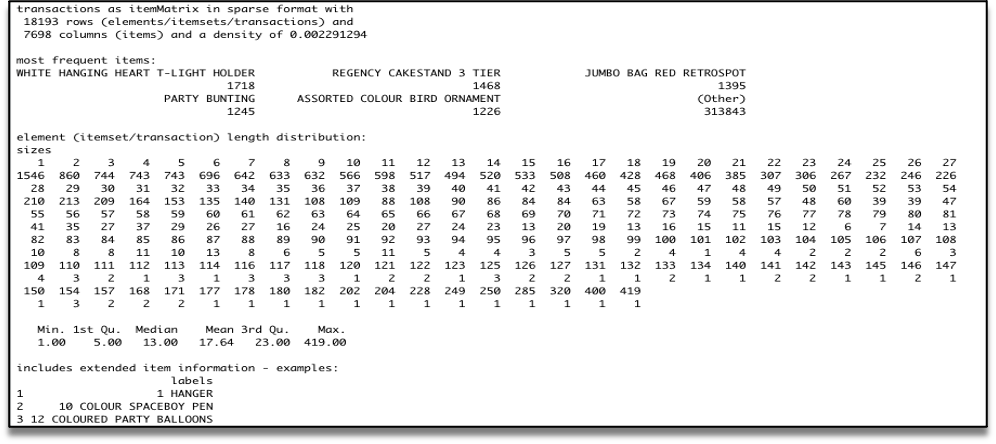
At this step we already have our transaction dataset, and it shows the matrix of items which bought together. We can’t see here any rules and how often it was purchase together. Now let’s check how many transactions we have and what they are. We will have to have to load this transaction data into an object of the transaction class. This is done by using the R function read.transactions of the arules package. Our format of Data frame is basket.

image

Let’s have a view our transaction object by summary(transaction)

image

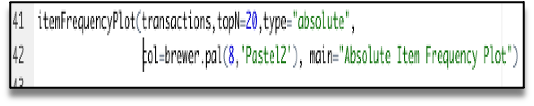
We can see 18193 transactions (rows) and 7698 items (columns). 7698 is the product descriptions and 18193 transactions are collections of these items.



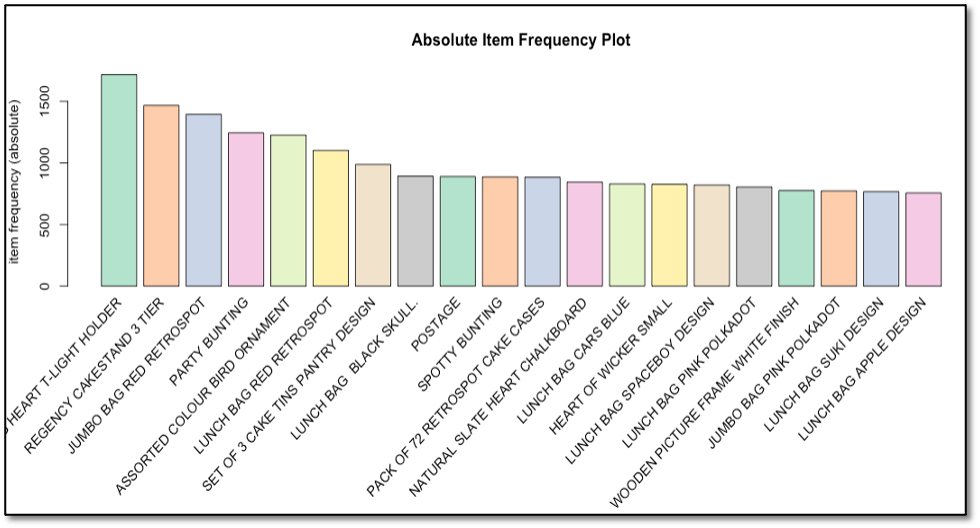
The summary gives us some useful information:

* Density tells the percentage of non-zero cells in a sparse matrix. In other words, total number of items that are purchased divided by a possible number of items in that matrix. You can calculate how many items were purchased by using density: 18193x7698x0.002291294=337445
* Summary will show us most frequent items.
* Element (itemset/transaction) length distribution: It will gave us how many transactions are there for 1-itemset, 2-itemset and so on. The first row is telling you a number of items and the second row is telling you the number of transactions.  
  For example, there is only 1546 transaction for one item, 860 transactions for 2 items, and there are 419 items in one transaction which is the longest.

Let’s check item frequency plot, we will generate an itemFrequencyPlot to create an item Frequency Bar Plot to view the distribution of objects based on itemMatrix (e.g., >transactions or items in >itemsets and >rules) which is our case.







In itemFrequencyPlot(transaction,topN=20,type="absolute") first argument - our transaction object to be plotted that is tr. topN is allows us to plot top N highest frequency items. type can be as type="absolute" or type="relative". If we will chouse absolute it will plot numeric frequencies of each item independently. If relative it will plot how many times these items have appeared as compared to others. As well I made it in colure for better visualization

**VISUALIZATION**

In today’s world, a lot of data is being generated on a daily basis. And sometimes to analyze this data for certain trends, patterns may become difficult if the data is in its raw format. To overcome this data visualization comes into play. Data visualization provides a good, organized pictorial representation of the data which makes it easier to understand, observe, analyze. In this tutorial, we will discuss how to visualize data using Python.



Python provides various libraries that come with different features for visualizing data. All these libraries come with different features and can support various types of graphs.

In this project, we will be discussing four such libraries.

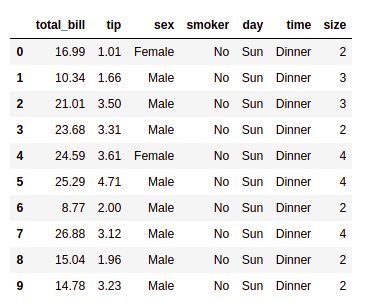
* **Matplotlib**
* **Seaborn**
* **Bokeh**
* **ScatterPlot**

## Database Used

## Database is the record of the tip given by the customers in a restaurant for two and a half months in the early 1990s. It contains 6 columns such as total\_bill, tip, sex, smoker, day, time, size.

|  |
| --- |
| **import** pandas as pd      # reading the database  data **=** pd.read\_csv("tips.csv")    # printing the top 10 rows  display(data.head(10)) |

**Output:**

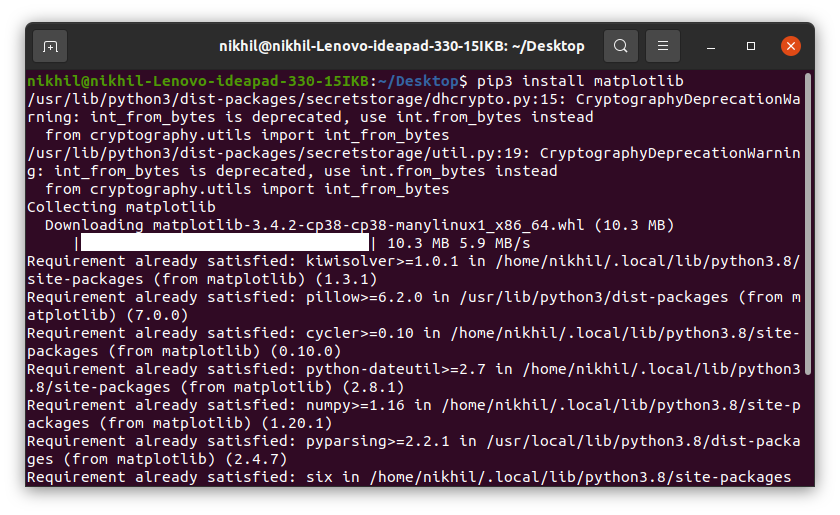


## Matplotlib

Matplotlib is an easy-to-use, low-level data visualization library that is built on NumPy arrays. It consists of various plots like scatter plot, line plot, histogram, etc. Matplotlib provides a lot of flexibility.

To install this type the below command in the terminal.

**pip install matplotlib**

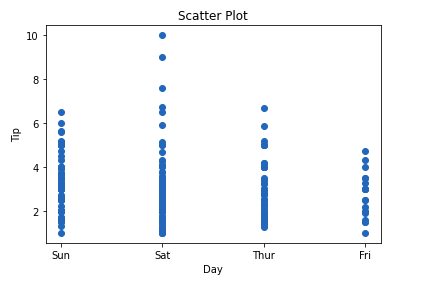


### Scatter Plot

Scatter plots are used to observe relationships between variables and uses dots to represent the relationship between them. The [**scatter()**](https://www.geeksforgeeks.org/matplotlib-pyplot-scatter-in-python/) method in the matplotlib library is used to draw a scatter plot.

|  |
| --- |
| **import** pandas as pd  **import** matplotlib.pyplot as plt      # reading the database  data **=** pd.read\_csv("tips.csv")    # Scatter plot with day against tip  plt.scatter(data['day'], data['tip'])    # Adding Title to the Plot  plt.title("Scatter Plot")    # Setting the X and Y labels  plt.xlabel('Day')  plt.ylabel('Tip')    plt.show() |

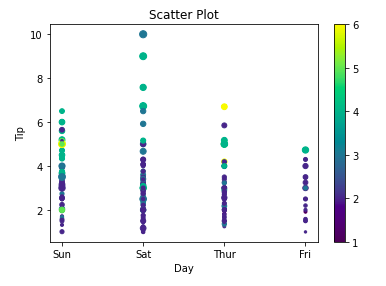
**Output:**



This graph can be more meaningful if we can add colors and also change the size of the points. We can do this by using the **c and s** parameterrespectivelyof the scatter function. We can also show the color bar using the [colorbar()](https://www.geeksforgeeks.org/matplotlib-pyplot-colorbar-function-in-python/) method.

|  |
| --- |
| **import** pandas as pd  **import** matplotlib.pyplot as plt      # reading the database  data **=** pd.read\_csv("tips.csv")    # Scatter plot with day against tip  plt.scatter(data['day'], data['tip'], c**=**data['size'],  s**=**data['total\_bill'])    # Adding Title to the Plot  plt.title("Scatter Plot")    # Setting the X and Y labels  plt.xlabel('Day')  plt.ylabel('Tip')    plt.colorbar()    plt.show() |

**Output:**

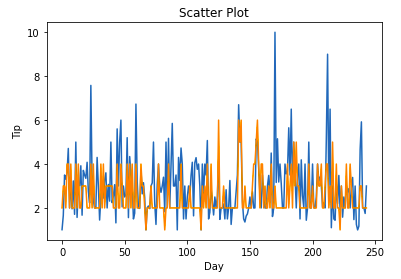


### Line Chart

[Line Chart](https://www.geeksforgeeks.org/line-chart-in-matplotlib-python/) is used to represent a relationship between two data X and Y on a different axis. It is plotted using the **plot()** function. Let’s see the below example.

|  |
| --- |
| **import** pandas as pd  **import** matplotlib.pyplot as plt      # reading the database  data **=** pd.read\_csv("tips.csv")    # Scatter plot with day against tip  plt.plot(data['tip'])  plt.plot(data['size'])    # Adding Title to the Plot  plt.title("Scatter Plot")    # Setting the X and Y labels  plt.xlabel('Day')  plt.ylabel('Tip')    plt.show() |

**Output:**

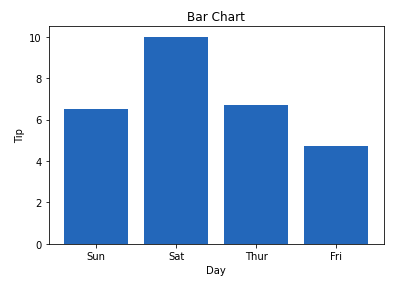


### Bar Chart

A [bar plot](https://www.geeksforgeeks.org/bar-plot-in-matplotlib/) or bar chart is a graph that represents the category of data with rectangular bars with lengths and heights that is proportional to the values which they represent. It can be created using the **bar()** method.

|  |
| --- |
| **import** pandas as pd  **import** matplotlib.pyplot as plt      # reading the database  data **=** pd.read\_csv("tips.csv")    # Bar chart with day against tip  plt.bar(data['day'], data['tip'])    plt.title("Bar Chart")    # Setting the X and Y labels  plt.xlabel('Day')  plt.ylabel('Tip')    # Adding the legends  plt.show() |

**Output:**

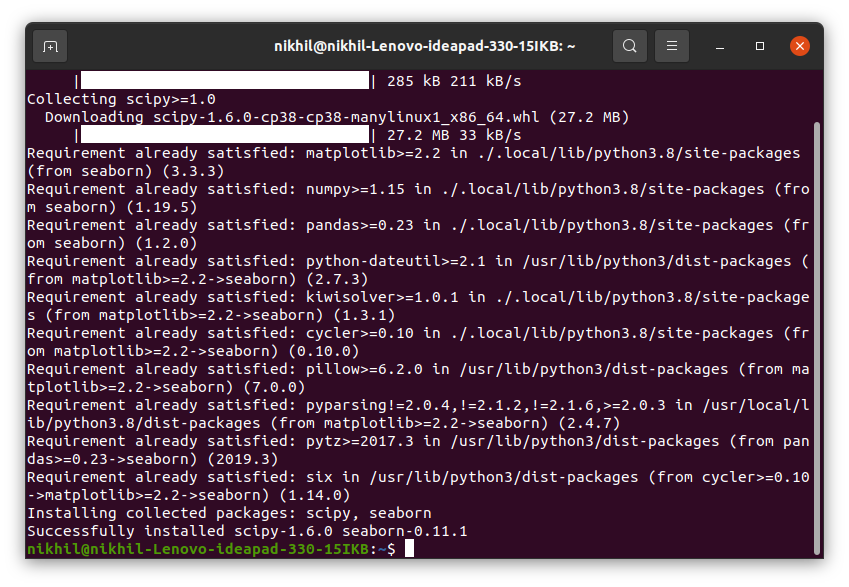


## Seaborn

**Seaborn** is a high-level interface built on top of the Matplotlib. It provides beautiful design styles and color palettes to make more attractive graphs.

To install seaborn type the below command in the terminal.d

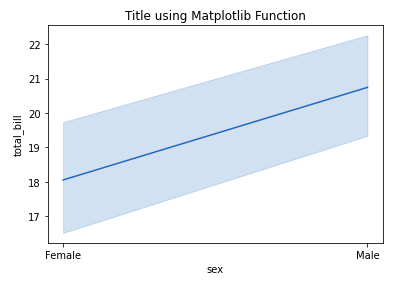
**pip install seaborn**



Seaborn is built on the top of Matplotlib, therefore it can be used with the Matplotlib as well. Using both Matplotlib and Seaborn together is a very simple process. We just have to invoke the Seaborn Plotting function as normal, and then we can use Matplotlib’s customization function.

|  |
| --- |
| # importing packages  **import** seaborn as sns  **import** matplotlib.pyplot as plt  **import** pandas as pd      # reading the database  data **=** pd.read\_csv("tips.csv")    # draw lineplot  sns.lineplot(x**=**"sex", y**=**"total\_bill", data**=**data)    # setting the title using Matplotlib  plt.title('Title using Matplotlib Function')    plt.show() |

**Output:**

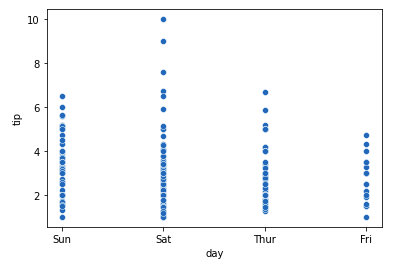


### Scatter Plot

[Scatter plot](https://www.geeksforgeeks.org/scatterplot-using-seaborn-in-python/) is plotted using the **scatterplot()** method. This is similar to Matplotlib, but additional argument data is reque

|  |
| --- |
| # importing packages  **import** seaborn as sns  **import** matplotlib.pyplot as plt  **import** pandas as pd    # reading the database  data **=** pd.read\_csv("tips.csv")    sns.scatterplot(x**=**'day', y**=**'tip', data**=**data,)  plt.show() |

**Output:**



You will find that while using Matplotlib it will a lot difficult if you want to color each point of this plot according to the sex. But in scatter plot it can be done with the help of hue argument.

|  |
| --- |
| # importing packages  **import** seaborn as sns  **import** matplotlib.pyplot as plt  **import** pandas as pd    # reading the database  data **=** pd.read\_csv("tips.csv")    sns.scatterplot(x**=**'day', y**=**'tip', data**=**data,  hue**=**'sex')  plt.show() |

**Output:**

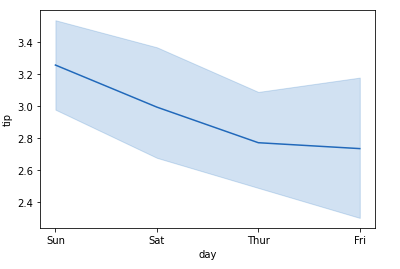
### 

### Line Plot

[Line Plot](https://www.geeksforgeeks.org/data-visualization-with-seaborn-line-plot/) in Seaborn plotted using the [**lineplot()**](https://www.geeksforgeeks.org/seaborn-lineplot-method-in-python/) method. In this, we can pass only the data argument also.

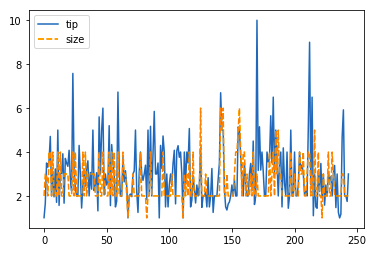
|  |
| --- |
| # importing packages  **import** seaborn as sns  **import** matplotlib.pyplot as plt  **import** pandas as pd      # reading the database  data **=** pd.read\_csv("tips.csv")    sns.lineplot(x**=**'day', y**=**'tip', data**=**data)  plt.show() |

**Output:**



|  |
| --- |
| # importing packages  **import** seaborn as sns  **import** matplotlib.pyplot as plt  **import** pandas as pd      # reading the database  data **=** pd.read\_csv("tips.csv")    # using only data attribute  sns.lineplot(data**=**data.drop(['total\_bill'], axis**=**1))  plt.show() |

**Output:**

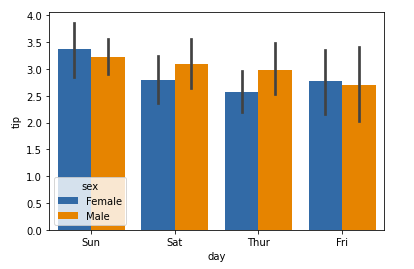


### Bar Plot

[Bar Plot](https://www.geeksforgeeks.org/barplot-using-seaborn-in-python/) in Seaborn can be created using the [**barplot()**](https://www.geeksforgeeks.org/seaborn-barplot-method-in-python/) method.

|  |
| --- |
| # importing packages  **import** seaborn as sns  **import** matplotlib.pyplot as plt  **import** pandas as pd      # reading the database  data **=** pd.read\_csv("tips.csv")    sns.barplot(x**=**'day',y**=**'tip', data**=**data,  hue**=**'sex')    plt.show() |

**Output:**

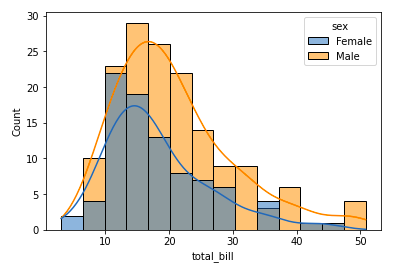


### Histogram

The histogram in Seaborn can be plotted using the **histplot()** fun

|  |
| --- |
| # importing packages  **import** seaborn as sns  **import** matplotlib.pyplot as plt  **import** pandas as pd      # reading the database  data **=** pd.read\_csv("tips.csv")    sns.histplot(x**=**'total\_bill', data**=**data, kde**=**True, hue**=**'sex')    plt.show() |

**Output:**



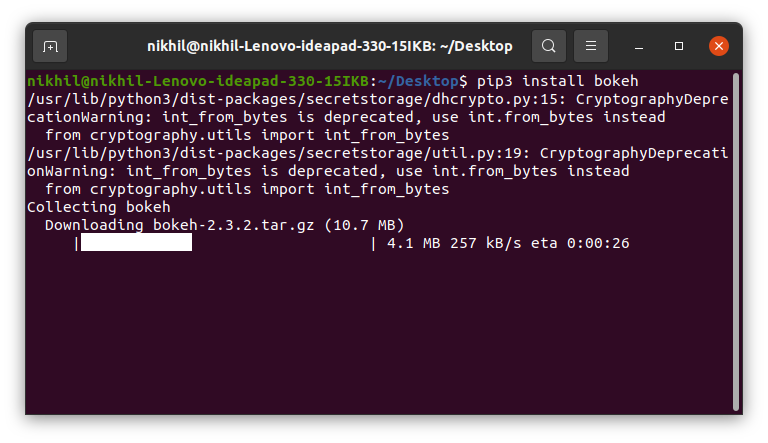
After going through all these plots you must have noticed that customizing plots using Seaborn is a lot more easier than using Matplotlib. And it is also built over matplotlib then we can also use matplotlib functions while using Seaborn.

## Bokeh

## Bokeh is mainly famous for its interactive charts visualization. Bokeh renders its plots using HTML and JavaScript that uses modern web browsers for presenting elegant, concise construction of novel graphics with high-level interactivity.

To install this type the below command in the terminal.

**pip install bokeh**

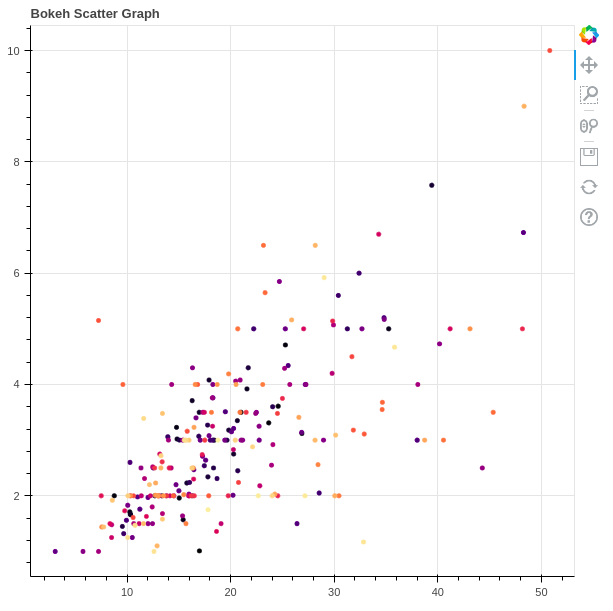


### Scatter Plot

[Scatter Plot](https://www.geeksforgeeks.org/python-bokeh-plotting-a-scatter-plot-on-a-graph/) in Bokeh can be plotted using the scatter() method of the plotting module. Here pass the x and y coordinates respectively.

|  |
| --- |
| # importing the modules  **from** bokeh.plotting **import** figure, output\_file, show  **from** bokeh.palettes **import** magma  **import** pandas as pd      # instantiating the figure object  graph **=** figure(title **=** "Bokeh Scatter Graph")    # reading the database  data **=** pd.read\_csv("tips.csv")    color **=** magma(256)    # plotting the graph  graph.scatter(data['total\_bill'], data['tip'], color**=**color)    # displaying the model  show(graph) |

**Output:**



**Business Recommendations**

**Actionable Strategies**

The culmination of our analytical journey involves extracting actionable recommendations aligned seamlessly with the strategic goals of the retail business. Our insights go beyond being mere observations; they are actionable strategies seamlessly integrated into the fabric of business operations.

**Product Bundling for Promotions**

Insights derived from Apriori, such as strong associations between products, pave the way for concrete recommendations. This may involve strategic product bundling for targeted promotions, a proven strategy in influencing customer behavior.

**Optimized Inventory Management**

Recommendations extend to optimizing inventory management based on purchasing patterns. This strategic approach ensures that stock levels align dynamically with customer demand, enhancing operational efficiency.

**Tailored Pricing and Marketing Campaigns**

Our strategies encompass tailored pricing and marketing campaigns, where insights from association analysis guide the formulation of approaches that resonate with customers.

This tailored approach ensures that marketing efforts are not only impactful but also cost-effective.

**CONCLUSION:**

Our approach goes beyond conventional data analysis; it empowers the retail business with actionable intelligence, driving strategic decision-making and enhancing customer-centric operations. Through this extensive and meticulous process, we not only uncover patterns but translate them into tangible strategies that propel the business forward in the highly competitive and dynamic retail landscape.