TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER : 1

INTRODUCTION

1.1 Objective

Holi sales analysis to improve customer experience and sales-

1. Improve customer experience by analysing sales data
2. Increase revenue.

1.2 What is Exploratory Data Analysis

Exploratory data analysis (EDA) is used by data scientists to analyze and investigate data sets and summarize their main characteristics, often employing data visualization methods. It helps determine how best to manipulate data sources to get the answers you need, making it easier for data scientists to discover patterns, spot anomalies, test a hypothesis, or check assumptions.

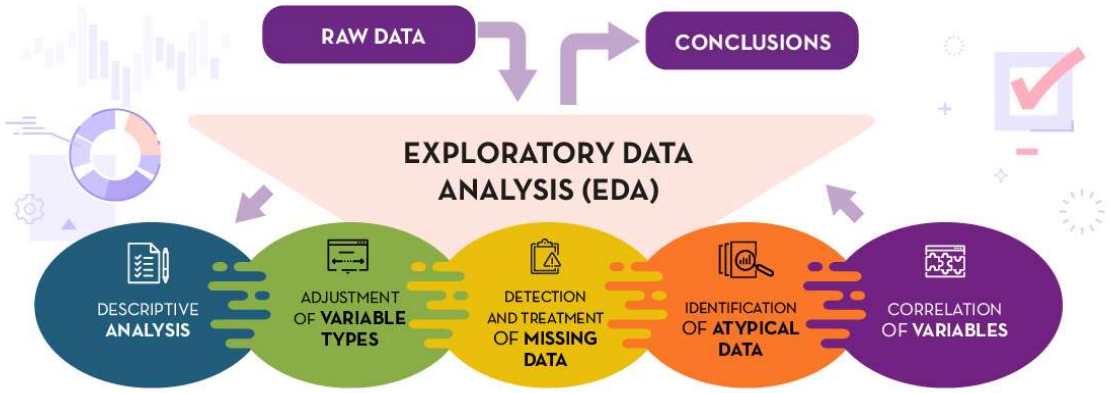


Figure 1.1 Exploratory Data Analysis

Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns,to spot anomalies,to test hypothesis and to check assumptions with the help of summary statistics and graphical representations.

EDA is primarily used to see what data can reveal beyond the formal modeling or hypothesis testing task and provides a provides a better understanding of data set variables and the relationships between them. It can also help determine if the statistical techniques you are considering for data analysis are appropriate.

Why is Exploratory data analysis important in data science?

Exploratory Data Analysis (EDA) is one of the techniques used for extracting vital features and trends used by machine learning and deep learning models in Data Science. Thus, EDA has become an important milestone for anyone working in data science. This article covers the

concept, meaning, tools, and techniques of EDA to give complete awareness to a beginner wanting to launch a career in data science. The article also enlists those fields that regularly apply EDA efficiently in promoting their business activities.

The main purpose of EDA is to help look at data before making any assumptions. It can help identify obvious errors, as well as better understand patterns within the data, detect outliers or anomalous events, find interesting relations among the variables.

The Data Science field is now very important in the business world as it provides many opportunities to make vital business decisions by analyzing hugely gathered data. Understanding the data thoroughly needs its exploration from every aspect. The impactful features enable making meaningful and beneficial decisions; therefore, EDA occupies an invaluable place in Data science.

Data scientists can use exploratory analysis to ensure the results they produce are valid and applicable to any desired business outcomes and goals. EDA also helps stakeholders by confirming they are asking the right questions. EDA can help answer questions about standard deviations, categorical variables, and confidence intervals. Once EDA is complete and insights are drawn, its features can then be used for more sophisticated data analysis or modeling, including machine learning.

Objective of Exploratory Data Analysis

The overall objective of exploratory data analysis is to obtain vital insights and hence usually includes the following sub-objectives:

1. Identifying and removing data outliers
2. Identifying trends in time and space
3. Uncover patterns related to the target
4. Creating hypotheses and testing them through experiments
5. Identifying new sources of data

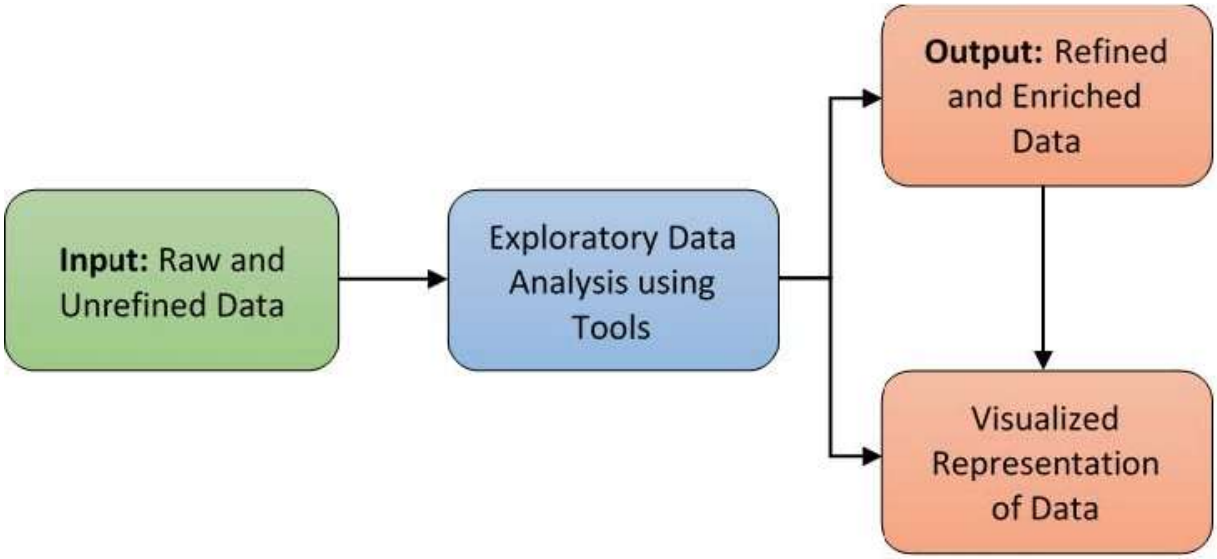


Figure 1.1 Exploratory Data Analysis steps

Steps Involved in Exploratory Data Analysis (EDA)

The key components in an EDA are the main steps undertaken to perform the EDA. These are as follows:

1. Data Collection

Nowadays, data is generated in huge volumes and various forms belonging to every sector of human life, like healthcare, sports, manufacturing, tourism, and so on. Every business knows the importance of using data beneficially by properly analyzing it. However, this depends on collecting the required data from various sources through surveys, social media, and customer reviews, to name a few. Without collecting sufficient and relevant data, further activities cannot begin.

2. Finding all Variables and Understanding Them

When the analysis process starts, the first focus is on the available data that gives a lot of information. This information contains changing values about various features or characteristics, which helps to understand and get valuable insights from them. It requires first identifying the important variables which affect the outcome and their possible impact. This step is crucial for the final result expected from any analysis.

3. Cleaning the Dataset

The next step is to clean the data set, which may contain null values and irrelevant information. These are to be removed so that data contains only those values that are relevant and important from the target point of view. This will not only reduce time but also reduces the computational power from an estimation point of view. Preprocessing takes care of all issues, such as identifying null values, outliers, anomaly detection, etc.

4. Identify Correlated Variables

Finding a correlation between variables helps to know how a particular variable is related to another. The correlation matrix method gives a clear picture of how different variables correlate, which further helps in understanding vital relationships among them.

5. Choosing the Right Statistical Methods

As will be seen in later sections, depending on the data, categorical or numerical, the size, type of variables, and the purpose of analysis, different statistical tools are employed. Statistical formulae applied for numerical outputs give fair information, but graphical visuals are more appealing and easier to interpret.

6. Visualizing and Analyzing Results

Once the analysis is over, the findings are to be observed cautiously and carefully so that proper interpretation can be made. The trends in the spread of data and correlation between variables give good insights for making suitable changes in the data parameters. The data analyst should have the requisite capability to analyze and be well-versed in all analysis techniques. The results obtained will be appropriate to data of that particular domain and are suitable for use in retail, healthcare, and agriculture.

Aspiring data science professionals must understand and practice the above EDA data science steps to master exploratory data analysis.

Exploratory data analysis tools-
Specific statistical functions and techniques you can perform with EDA tools include:

1. Clustering and dimension reduction techniques, which help create graphical displays of high-dimensional data containing many variables.
2. Univariate visualization of each field in the raw dataset, with summary statistics.
3. Bivariate visualizations and summary statistics that allow you to assess the relationship between each variable in the dataset and the target variable you're looking at.
4. Multivariate visualizations, for mapping and understanding interactions between different fields in the data.
5. K-means Clustering is a clustering method in unsupervised learning where data points are assigned into K groups, i.e. the number of clusters, based on the distance from each group's centroid. The data points closest to a particular centroid will be clustered under the same category. K-means Clustering is commonly used in market segmentation, pattern recognition, and image compression.
6. Predictive models, such as linear regression, use statistics and data to predict outcomes.

Types of exploratory data analysis-

There are four primary types of EDA:

1. Univariate non-graphical. This is simplest form of data analysis, where the data being analyzed consists of just one variable. Since it's a single variable, it doesn't deal with causes or relationships. The main purpose of univariate analysis is to describe the data and find patterns that exist within it.
2. Univariate graphical. Non-graphical methods don't provide a full picture of the data. Graphical methods are therefore required. Common types of univariate graphics include:
    a. Stem-and-leaf plots, which show all data values and the shape of the distribution.
    b. Histograms, a bar plot in which each bar represents the frequency (count) or proportion (count/total count) of cases for a range of values.
    c. Box plots, which graphically depict the five-number summary of minimum, first quartile, median, third quartile, and maximum.
3. Multivariate nongraphical: Multivariate data arises from more than one variable. Multivariate non-graphical EDA techniques generally show the relationship between two or more variables of the data through cross-tabulation or statistics.
4. Multivariate graphical: Multivariate data uses graphics to display relationships between two or more sets of data. The most used graphic is a grouped bar plot or bar chart with each group representing one level of one of the variables and each bar within a group representing the levels of the other variable.

Other common types of multivariate graphics include:

1. Scatter plot, which is used to plot data points on a horizontal and a vertical axis to show how much one variable is affected by another.
2. Multivariate chart, which is a graphical representation of the relationships between factors and a response.
3. Run chart, which is a line graph of data plotted over time.
4. Bubble chart, which is a data visualization that displays multiple circles (bubbles) in a two-dimensional plot.
5. Heat map, which is a graphical representation of data where values are depicted by color.

Exploratory Data Analysis Tools-

Some of the most common data science tools used to create an EDA include:

1. Python: An interpreted, object-oriented programming language with dynamic semantics. Its high-level, built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for rapid application development, as well as for use as a scripting or glue language to connect existing components together. Python and EDA can be used together to identify missing values in a data set, which is important so you can decide how to handle missing values for machine learning.
2. R: An open-source programming language and free software environment for statistical computing and graphics supported by the R Foundation for Statistical Computing. The R language is widely used among statisticians in data science in developing statistical observations and data analysis.

1.3 Scope

CHAPTER : 2

BACKGROUND AND LITERATURE SURVEY

2.0 Software Specification and Literature Survey

Requirement specifications is a way of summarizing the requirements that the customer and developer has on the product or program. It is often done in different types of lists and stretches from design to functions and backend. When all the demands in the requirement specification is met the product is done.

This report and project can be broken into a few different steps with the over-arching goal of building a platform where users can web scrape their Cartrade pages, upload this parsed, structured, and cleansed data a website where the data gets stored in a database. This data can then be presented in a visually pleasing way to the user. The different processes chosen for the project are presented together with the main steps done to ensure that the project is followed through.

2.1 Time planning

Due to the scope of the project the structure and planning are key elements to achieve the set goals. Three different objects need to be built with a python-based web scraper, a dynamic website that can handle active users and a database to store both log-in information regarding the user, but also the scraped data that the user uploads. To achieve this a Gant-based schedule was chosen to plan how the time was divided between the three different objects and writing the report.

2.2 Analysing process

A development process was created early in the project to go from a concept and idea to developed product. The development process for this project can be divided into 6 different steps.

2.2.1 Pre-study and data collection

The first step in the pre study is to analyze what needs to be done. How much research is needed to learn how to build a python-based web scraper. During this step questions like if code from previous projects can be re-used or not is answered. It's also the step where the baseline of the platform is outlined. Exploratory Data Analysis by the help of Python – EDA is primarily used to see what data can reveal beyond the formal modeling or hypothesis testing est

2.2.2 Requirement specification and user stories

The second step is to do a requirement specification. What must be developed to accomplish the project based on the goals and boundaries that has been previously mentioned. Creating a solid requirement specification is important so the development has an end goal. The requirement is usually what the project needs to accomplish when done.

2.2.3 Functional analysis

The third step is to do a functional analysis. Step two, requirement specification, makes it clear what the projects requirements are, what it needs to achieve, while functional specification is a continuation of this. The functional analysis is done to evaluate what functions the platform needs to have. These functions are then presented in a table that clearly outlines what the developed product needs to be able to do.

2.2.4 Traditional development

During the project iterative and agile development was used. A lot of knowledge was gained, and a lot of research was read during the project, which mean that the most important part of development was having an iterative pipeline where code and functions could be changed and iterative as the development continued. The result of the development is the finished project.

2.2.5 Python

Python is a well-developed strong language with a vast library, with the most important one for this project being beautifulsoup for web scraping. Different languages were considered for this project the choice of Python was motived through previous experience with the language and the existence of the web scraping library beautiful soup.

2.2.6 Pandas

Pandas is undoubtedly the most popular package for performing data analysis in Python, thanks to its rich-intuitive functionalities that allow us to perform countless manipulation of our data with ease. It is then no wonder that the package has become an indispensable tool for many data scientists/analysts to handle their day-to-day tasks.

2.3 Evaluation

Once everything has been developed an evaluation is done to see if the new system is better than the old system. This is a way of quality control to make sure that the goals of the project were reached. The evaluation will be done measuring two different variables. The number of clicks needed for both the new system and the old system, and the amount of time needed for the old system and the new system. This will then be presented under chapter 4. Developing a Python based web scraper – A study on the development of a web scraper for cartrade. The number of clicks is measured through counting the number of clicks from start to result. The amount of time is measured by the amount of time from start to presented result. The tests are done by the author of this report, and done back to back to ensure the fairest amount of measurements.

2.4 Presentation

When the development process is done, and the coding and development of the project is finished a technical report is written. This report includes the theory behind the project, the tools used and how the project was constructed. During the entire project an iterative design and development process was used to make sure that time was allocated as best it could be and that the project reached completion in time for the deadline.

CHAPTER : 3

DESIGN

3.1 Data Flow Diagram

It’s easy to understand the flow of data through systems with the right data flow diagram.

What is Data Flow Diagram?

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled. They can be used to analyze an existing system or model a new one. Like all the best diagrams and charts, a DFD can often visually “say” things that would be hard to explain in words, and they work for both technical and nontechnical audiences, from developer to CEO. That’s why DFDs remain so popular after all these years. While they work well for data flow software and systems, they are less applicable nowadays to visualizing interactive, real-time or database-oriented software or systems.

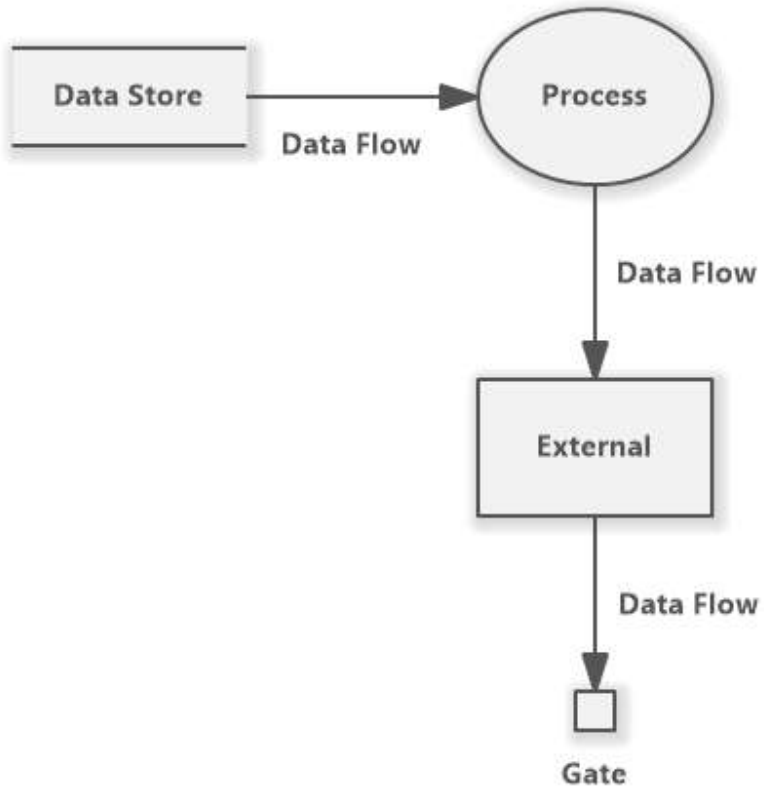


Figure 3.1 Data flow overview diagram

History of Data Flow Diagram

Data flow diagrams were popularized in the late 1970s, arising from the book structural design, by computing pioneers Ed Yourdon and Larry Constantine. They based it on the “data flow graph” computation models by David Martin and Gerald Estrin. The structured design concept took off in the software engineering field, and the DFD method took off with it. It became more popular in business circles, as it was applied to business analysis, than in academic circles.

3.2 EDA Data Flow Diagram-

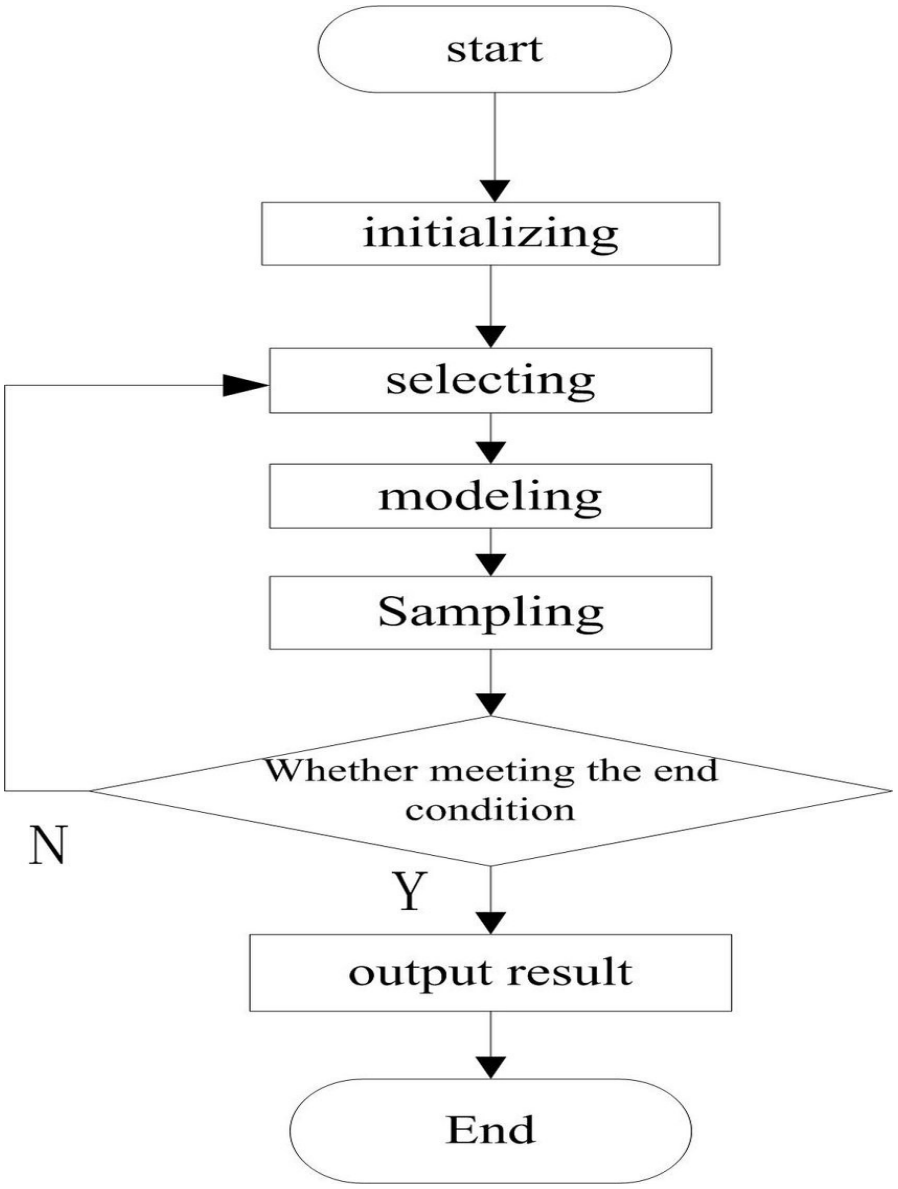


Figure 3.2 Data flow diagram

3.2 Use Case Diagram

A use case diagram is used to represent the dynamic behavior of a system. It encapsulates the system's functionality by incorporating use cases, actors, and their relationships. It models the tasks, services, and functions required by a system/subsystem of an application. It depicts the high-level functionality of a system and also tells how the user handles a system.

The main purpose of a use case diagram is to portray the dynamic aspect of a system. It accumulates the system's requirement, which includes both internal as well as external influences. It invokes persons, use cases, and several things that invoke the actors and elements accountable for the implementation of use case diagrams. It represents how an entity from the external environment can interact with a part of the system.

Following are the purposes of a use case diagram given below:

1. It gathers the system's needs.
2. It depicts the external view of the system.
3. It recognizes the internal as well as external factors that influence the system.
4. It represents the interaction between the actors.

How to draw a Use Case Diagram?

It is essential to analyze the whole system before starting with drawing a use case diagram, and then the system's functionalities are found. And once every single functionality is identified, they are then transformed into the use cases to be used in the use case diagram.

After that, we will enlist the actors that will interact with the system. The actors are the person or a thing that invokes the functionality of a system. It may be a system or a private entity, such that it requires an entity to be pertinent to the functionalities of the system to which it is going to interact.

Once both the actors and use cases are enlisted, the relation between the actor and use case/ system is inspected. It identifies the no of times an actor communicates with the system. Basically, an actor can interact multiple times with a use case or system at a particular instance of time.

Following are some rules that must be followed while drawing a use case diagram:

1. A pertinent and meaningful name should be assigned to the actor or a use case of a system.

2. The communication of an actor with a use case must be defined in an understandable way.

3. Specified notations to be used as and when required.

EDA Use Case Diagram

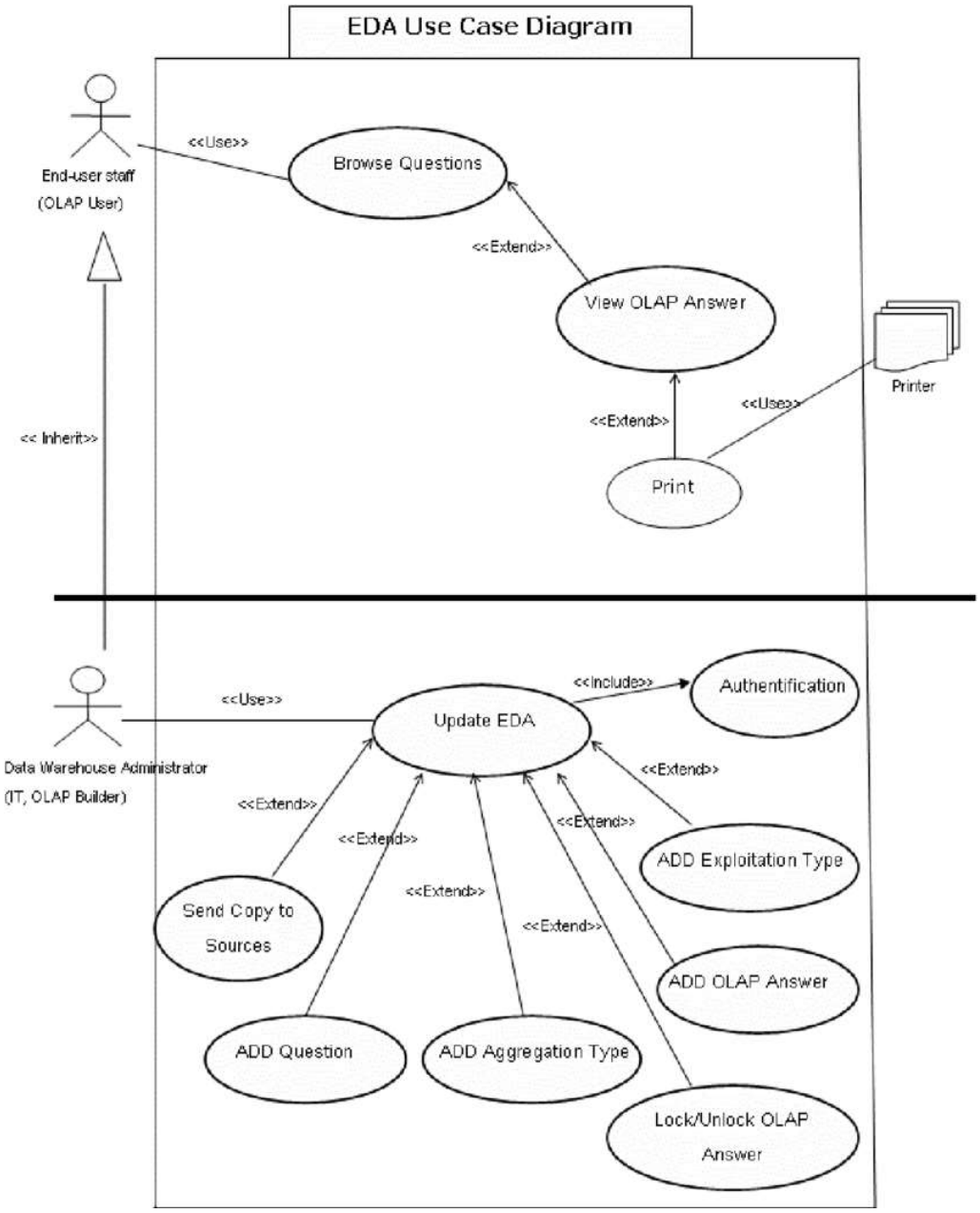


Figure 3.3 Usecase diagram

3.3 Activity Diagram

In UML, the activity diagram is used to demonstrate the flow of control within the system rather than the implementation. It models the concurrent and sequential activities.

The activity diagram helps in envisioning the workflow from one activity to another. It put emphasis on the condition of flow and the order in which it occurs. The flow can be sequential, branched, or concurrent, and to deal with such kinds of flows, the activity diagram has come up with a fork, join, etc.

It is also termed as an object-oriented flowchart. It encompasses activities composed of a set of actions or operations that are applied to model the behavioral diagram.

Why to draw an Activity diagram?

An activity diagram is a flowchart of activities, as it represents the workflow among various activities. They are identical to the flowcharts, but they themself are not exactly the flowchart. In other words, it can be said that an activity diagram is an enhancement of the flowchart, which encompasses several unique skills.Since it incorporates swimlanes, branching, parallel flows, join nodes, control nodes, and forks, it supports exception handling. A system must be explored as a whole before drawing an activity diagram to provide a clearer view of the user. All of the activities are explored after they are properly analyzed for finding out the constraints applied to the activities. Each and every activity, condition, and association must be recognized.

After gathering all the essential information, an abstract or a prototype is built, which is then transformed into the actual diagram.

Following are the rules that are to be followed for drawing an activity diagram:

1. A meaningful name should be given to each and every activity.
2. Identify all of the constraints.
3. Acknowledge the activity associations.

When to use an Activity Diagram?

An activity diagram can be used to portray business processes and workflows. Also, it used for modeling business as well as the software. An activity diagram is utilized for the followings:

1. To graphically model the workflow in an easier and understandable way.
2. To model the execution flow among several activities.
3. To model comprehensive information of a function or an algorithm employed within the system.

EDA Activity Diagram -

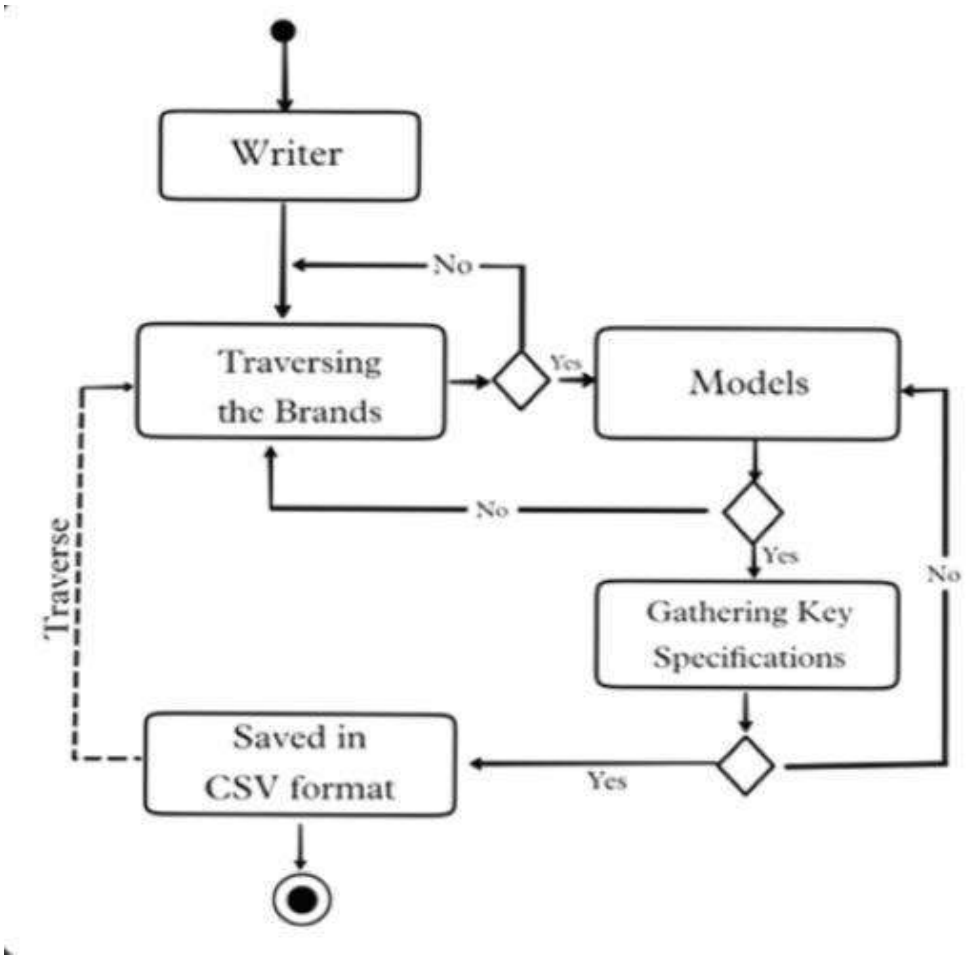


Figure 3.4 Activity diagram

CHAPTER : 4

TECHNOLOGY USED

4.1 Python

Python is a programming language. It was created by Guido van Rossum, and released in 1991.

Python is a computer programming language often used to build websites and software, automate tasks, and conduct data analysis. Python is a general-purpose language, meaning it can be used to create a variety of different programs and isn't specialized for any specific problems. This versatility, along with its beginner-friendliness, has made it one of the most-used programming languages today. A survey conducted by industry analyst firm RedMonk found that it was the second-most popular programming language among developers in 2021.

Python is commonly used for developing websites and software, task automation, data analysis, and data visualization. Since it's relatively easy to learn, Python has been adopted by many non-programmers such as accountants and scientists, for a variety of everyday tasks, like organizing finances.

"Writing programs is a very creative and rewarding activity," says University of Michigan and Coursera instructor Charles R Severance in his book *Python for Everybody.* "You can write programs for many reasons, ranging from making your living to solving a difficult data analysis problem to having fun to helping someone else solve a problem."

What can you do with python? Some things include:

- Data analysis and machine learning
- Web development
- Automation or scripting
- Software testing and prototyping

- Everyday tasks

Python is popular for a number of reasons. Here's a deeper look at what makes it so versatile and easy to use for coders.

- It has a simple syntax that mimics natural language, so it's easier to read and understand. This makes it quicker to build projects, and faster to improve on them.
- It's versatile. Python can be used for many different tasks, from web development to machine learning.
- It's beginner friendly, making it popular for entry-level coders.
- It's open source, which means it's free to use and distribute, even for commercial purposes.
- Python's archive of modules and libraries—bundles of code that third-party users have created to expand Python's capabilities—is vast and growing.
- Python has a large and active community that contributes to Python's pool of modules and libraries, and acts as a helpful resource for other programmers. The vast support community means that if coders run into a stumbling block, finding a solution is relatively easy; somebody is bound to have encountered the same problem before.

Import: Python modules can get access to code from another module by importing the file/function using import. The import statement is the most common way to invoking the import machinery, but it is not the only way.

4.1.1 Pandas.

Pandas is defined as an open-source library that provides high-performance data manipulation in Python. The name of Pandas is derived from the word Panel Data, which means an Econometrics from Multidimensional data. It is used for data analysis in Python and developed by Wes McKinney in 2008.

Data analysis requires lots of processing, such as restructuring, cleaning or merging, etc. There are different tools are available for fast data processing, such as Numpy, Scipy, Cython, and Panda. But we prefer Pandas because working with Pandas is fast, simple and more expressive than other tools.

Pandas is built on top of the Numpy package, means Numpy is required for operating the Pandas.

Before Pandas, Python was capable for data preparation, but it only provided limited support for data analysis. So, Pandas came into the picture and enhanced the capabilities of data analysis. It can perform five significant steps required for processing and analysis of data irrespective of the origin of the data, i.e., load, manipulate, prepare, model, and analyze.

The Pandas provides two data structures for processing the data, i.e., Series and DataFrame.

Pandas makes it easy to scrape a table(<table>tag) on a web page.

Key Features of Pandas

- It has a fast and efficient DataFrame object with the default and customized indexing.
- Used for reshaping and pivoting of the data sets.
- Group by data for aggregations and transformations.

- It is used for data alignment and integration of the missing data.
- Provide the functionality of Time Series.
- Process a variety of data sets in different formats like matrix data, tabular heterogeneous, time series.

4.1.2 Matplotlib.

Matplotlib is a cross-platform, data visualization and graphical plotting library (histograms, scatter plots, bar charts, etc) for Python and its numerical extension NumPy. As such, it offers a viable open source alternative to MATLAB. Developers can also use matplotlib's APIs (Application Programming Interfaces) to embed plots in GUI applications.

A Python matplotlib script is structured so that a few lines of code are all that is required in most instances to generate a visual data plot. The matplotlib scripting layer overlays two APIs:

1. The pyplot API is a hierarchy of Python code objects topped by *matplotlib.pyplot*
2. An OO (Object-Oriented) API collection of objects that can be assembled with greater flexibility than pyplot. This API provides direct access to Matplotlib's backend layers.

Matplotlib and Pyplot in Python

The pyplot API has a convenient MATLAB-style stateful interface. In fact, the matplotlib Python library was originally written as an open source alternative for MATLAB. The OO API and its interface is more customizable and powerful than pyplot, but considered more difficult to use. As a result, the pyplot interface is more commonly used, and is referred to by default in this article.

Understanding matplotlib's pyplot API is key to understanding how to work with plots:

> ***matplotlib.pyplot.figure: Figure*** is the top-level container. It includes everything visualized in a plot including one or more ***Axes***.
> ***matplotlib.pyplot.axes****: Axes* contain most of the elements in a plot**: *Axis, Tick, Line2D, Text,*** etc., and sets the coordinates. It is the area in which data is plotted. Axes include the X-Axis, Y-Axis, and possibly a Z-Axis, as well.

For more information about the pyplot API and interface, refer to ***What Is Pyplot In Matplotlib***

Installing Matplotlib

Matplotlib and its dependencies can be downloaded as a binary (pre-compiled) package from the Python Package Index (PyPI), and installed with the following command:

```
python -m pip install matplotlib
```

Matplotlib is also available as uncompiled source files from GitHub. Compiling from source will require your local system to have the appropriate compiler for your OS, all dependencies, setup scripts, configuration files, and patches available. This can result in a fairly complex installation. Alternatively, consider using the ActiveState Platform to automatically build matplotlib from source and package it for your OS.

4.1.2 Seaborn

Seaborn is one of an amazing library for visualization of the graphical statistical plotting in Python. Seaborn provides many color palettes and defaults beautiful styles to make the creation of many statistical plots in Python more attractive.

Objective of Python Seaborn library

Seaborn library aims to make a more attractive visualization of the central part of understanding and exploring data. It is built on the core of the matplotlib library and also provides dataset-oriented APIs.

Seaborn is also closely integrated with the Panda's data structures, and with this, we can easily jump between the various different visual representations for a given variable to better understand the provided dataset.

Categories of Plots in Python's seaborn library

Plots are generally used to make visualization of the relationships between the given variables. These variables can either be a category like a group, division, or class or can be completely numerical variables. There are various different categories of plots that we can create using the seaborn library.

In the seaborn library, the plot that we create is divided into the following various categories:

- Distribution plots: This type of plot is used for examining both types of distributions, i.e., univariate and bivariate distribution.
- Relational plots: This type of plot is used to understand the relation between the two given variables.
- Regression plots: Regression plots in the seaborn library are primarily intended to add an additional visual guide that will help to emphasize dataset patterns during the analysis of exploratory data.
- Categorical plots: The categorical plots are used to deals with categories of variables and how we can visualize them.
- Multi-plot grids: The multi-plot grids are also a type of plot that is a useful approach is to draw multiple instances for the same plot with different subsets of a single dataset.

- Matrix plots: The matrix plots are a type of arrays of the scatterplots.

Installation of seaborn library for Python

Here, we will learn how we can install the seaborn library for Python. After installing the seaborn library, we can import it into our Python program and use it in Python.

1. pip install seaborn

Required dependencies or prerequisites for the seaborn library:

We must have,

- Python installed with the latest version (3.6+).
- Numpy must be installed with version 1.13.3 or higher.
- SciPy must be installed with 1.0.1 or higher versions.
- Must have panda library with 0.22.0 or higher versions.
- statsmodel library must be installed with version 0.8.0 or higher.
- And should have matplotlib installed with 2.1.2 or higher versions.

Now, we will learn about some basic plots examples that we can plot in Python using the seaborn library.

Plotting Chart Using seaborn Library

1. Line plot:

The seaborn line plot is one of the most basic plots presents in the seaborn library. We use the seaborn line plot mainly to visualize the given data in some time-series form, i.e., in a continuous manner with respect to time.

2. Dist plot:

We use the seaborn dist plots to plot histograms with the given variables and data as a result. We can plot histograms with some other variations such as rugplot and kdeplot using a dist plot.

3. Lmplot:

The Lmplot is another one of the basic plots in the seaborn library. The Lmplot shows a line that represents a linear regression model with the data points on the given two-dimensional (2-D) space. In this 2-D space, we can set x and y variables as the vertical and horizontal labels, respectively.

4.2 Excel.

Excel is a spreadsheet program from Microsoft and a component of its Office product group for business applications. Microsoft Excel enables users to format, organize and calculate data in a spreadsheet.

By organizing data using software like Excel, data analysts and other users can make information easier to view as data is added or changed. Excel contains a large number of boxes called cells that are ordered in rows and columns. Data is placed in these cells.

Excel is a part of the Microsoft Office and Office 365 suites and is compatible with other applications in the Office suite. The spreadsheet software is available for Windows, macOS, Android and iOS platforms.

Features of MS Excel

Various editing and formatting can be done on an Excel spreadsheet. Discussed below are the various features of MS Excel.

The image below shows the composition of features in MS Excel:

- **Home**
- Comprises options like font size, font styles, font colour, background colour, alignment, formatting options and styles, insertion and deletion of cells and editing options

-
- **Insert**
- Comprises options like table format and style, inserting images and figures, adding graphs, charts and sparklines, header and footer option, equation and symbols
-
- **Page Layout**
- Themes, orientation and page setup options are available under the page layout option
-
- **Formulas**
- Since tables with a large amount of data can be created in MS excel, under this feature, you can add formulas to your table and get quicker solutions
-
- **Data**
- Adding external data (from the web), filtering options and data tools are available under this category
-
- **Review**
- Proofreading can be done for an excel sheet (like spell check) in the review category and a reader can add comments in this part
-
- **View**
- Different views in which we want the spreadsheet to be displayed can be edited here. Options to zoom in and out and pane arrangement are available under this category

Excel and XLS files

An XLS file is a spreadsheet file that can be created by Excel or other spreadsheet programs. The file type represents an Excel Binary File format. An XLS file stores data as binary streams -- a

compound file. Streams and substreams in the file contain information about the content and structure of an Excel workbook.

Versions of Excel after Excel 2007 use XLSX files by default, since it is a more open and structured format. Later versions of Excel still support the creation and reading of XLS files, however. Workbook data can also be exported in formats including PDF, TXT, Hypertext markup language, XPS and XLSX.

Macro-enabled Excel files use the XLSM file extension. In this case, macros are sets of instructions that automate Excel processes. XLSM files are similar to XLM files but are based on the Open XML format found in later Microsoft Office software.

Benefits of Using MS Excel

MS Excel is widely used for various purposes because the data is easy to save, and information can be added and removed without any discomfort and less hard work.

Given below are a few important benefits of using MS Excel:

- **Easy To Store Data:** Since there is no limit to the amount of information that can be saved in a spreadsheet, MS Excel is widely used to save data or to analyse data. Filtering information in Excel is easy and convenient.

- **Easy To Recover Data:** If the information is written on a piece of paper, finding it may take longer, however, this is not the case with excel spreadsheets. Finding and recovering data is easy.

- **Application of Mathematical Formulas:** Doing calculations has become easier and less time-taking with the formulas option in MS excel

- **More Secure:** These spreadsheets can be password secured in a laptop or personal computer and the probability of losing them is way lesser in comparison to data written in registers or piece of paper.

- **Data at One Place:** Earlier, data was to be kept in different files and registers when the paperwork was done. Now, this has become convenient as more than one worksheet can be added in a single MS Excel file.

- **Neater and Clearer Visibility of Information:** When the data is saved in the form of a table, analysing it becomes easier. Thus, information is a spreadsheet that is more readable and understandable.

CHAPTER- 5

CODING AND IMPLEMENTATION

Code-

```
# import python libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt # visualizing data
%matplotlib inline
import seaborn as sns


# import csv file
df = pd.read_csv('Diwali Sales Data.csv', encoding= 'unicode_escape')


 df.shape


df.head()


df.info()


#drop unrelated/blank columns
df.drop(['Status', 'unnamed1'], axis=1, inplace=True)


#check for null values
 pd.isnull(df).sum()


# drop null values
df.dropna(inplace=True)
```

```
# change data type
df['Amount'] = df['Amount'].astype('int')
```

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

```
df.columns
```

```
#rename column
df.rename(columns= {'Marital_Status':'Shaadi'})
```

```
# describe() method returns description of the data in the DataFrame (i.e. count, mean, std, etc)
df.describe()
```

```
# use describe() for specific columns
df[['Age', 'Orders', 'Amount']].describe()
```

#Exploratory Data Analysis

#Gender

```
# plotting a bar chart for Gender and it's count

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

for bars in ax.containers:
    ax.bar_label(bars)
```

```
# plotting a bar chart for gender vs total amount

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

sns.barplot(x = 'Gender',y= 'Amount' ,data = sales_gen)
```

#Age

```
ax = sns.countplot(data = df, x = 'Age Group', hue = 'Gender')

for bars in ax.containers:
    ax.bar_label(bars)
```

```
# Total Amount vs Age Group
sales_age = df.groupby(['Age Group'],
as_index=False)['Amount'].sum().sort_values(by='Amount', ascending=False)

sns.barplot(x = 'Age Group',y= 'Amount' ,data = sales_age)
```

#State

```
# total number of orders from top 10 states

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

sns.set(rc={'figure.figsize':(15,5)})
 sns.barplot(data = sales_state, x = 'State',y= 'Orders')
```

```
# total amount/sales from top 10 states

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')
```

#Marital Status

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

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


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')
```

#Occupation

```
sns.set(rc={'figure.figsize':(20,5)})
ax = sns.countplot(data = df, x = 'Occupation')

for bars in ax.containers:
    ax.bar_label(bars)
```

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

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

#Product Category

```
sns.set(rc={'figure.figsize':(20,5)})
ax = sns.countplot(data = df, x = 'Product_Category')

for bars in ax.containers:
    ax.bar_label(bars)
```

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

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

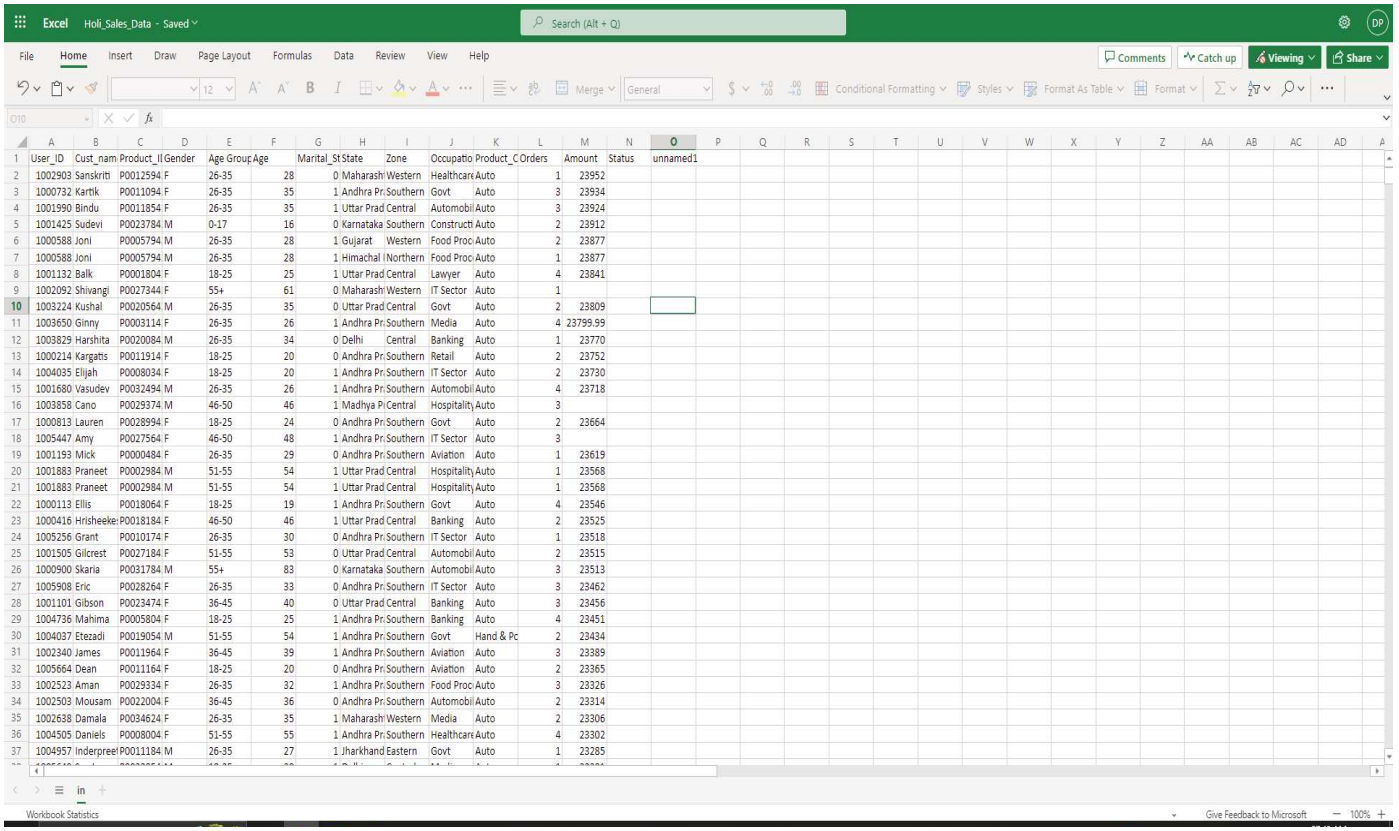
```
sales_state = df.groupby(['Product_ID'],
as_index=False)['Orders'].sum().sort_values(by='Orders', ascending=False).head(10)

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

 # top 10 most sold products (same thing as above)

```
fig1, ax1 = plt.subplots(figsize=(12,7))
```

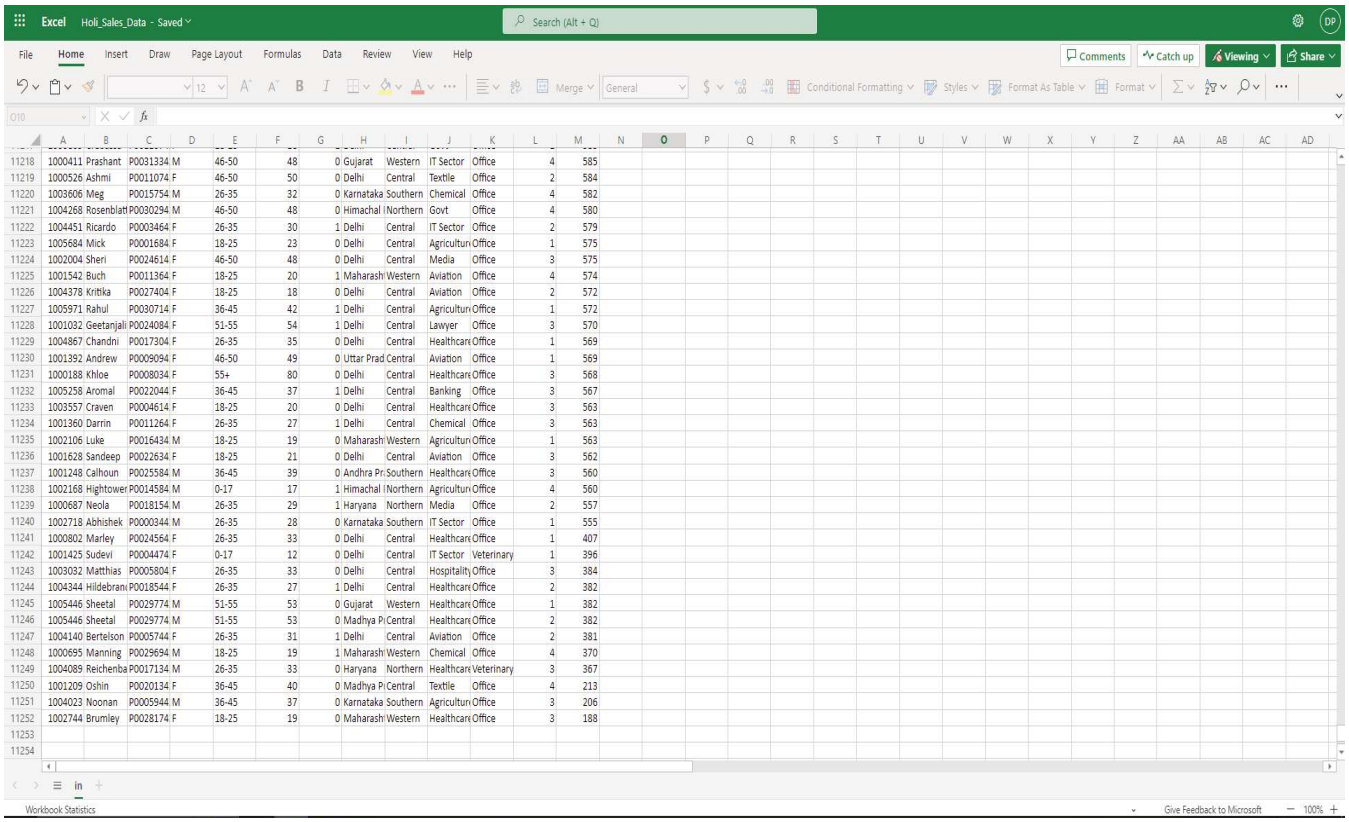
```
df.groupby('Product_ID')['Orders'].sum().nlargest(10).sort_values(ascending=False).plot(kind='bar')
```

Excel Sheets



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	User_ID	Cust_nam	Product_ID	Gender	Age Group	Age	Marital_St	State	Zone	Occupatio	Product_C	Orders	Amount	Status	unnamed1
2	1002903	Sanskriti	P0012594	F	26-35	28	0	Maharashi	Western	Healthcare	Auto	1	23952		
3	1000732	Kartik	P0011094	F	26-35	35	1	Andhra Pr	Southern	Govt	Auto	3	23934		
4	1001990	Bindu	P0011854	F	26-35	35	1	Uttar Prad	Central	Automobil	Auto	3	23924		
5	1001425	Sudevi	P0023784	M	0-17	16	0	Karnataka	Southern	Constructi	Auto	2	23912		
6	1000588	Joni	P0005794	M	26-35	28	1	Gujarat	Western	Food Proc	Auto	2	23877		
7	1000588	Joni	P0005794	M	26-35	28	1	Himachal	Northern	Food Proc	Auto	1	23877		
8	1001132	Balk	P0001804	F	18-25	25	1	Uttar Prad	Central	Lawyer	Auto	4	23841		
9	1002092	Shivangi	P0027344	F	55+	61	0	Maharashi	Western	IT Sector	Auto	1			
10	1003224	Kushal	P0020564	M	26-35	35	0	Uttar Prad	Central	Govt	Auto	2	23809		
11	1003650	Ginny	P0003114	F	26-35	26	1	Andhra Pr	Southern	Media	Auto	4	23799.99		
12	1003829	Harshita	P0020084	M	26-35	34	0	Delhi	Central	Banking	Auto	1	23770		
13	1000214	Kargatis	P0011914	F	18-25	20	0	Andhra Pr	Southern	Retail	Auto	2	23752		
14	1004035	Elijah	P0008034	F	18-25	20	1	Andhra Pr	Southern	IT Sector	Auto	2	23730		
15	1001680	Vasudev	P0032494	M	26-35	26	1	Andhra Pr	Southern	Automobil	Auto	4	23718		
16	1003858	Cano	P0029374	M	46-50	46	1	Madhya Pr	Central	Hospitality	Auto	3			
17	1000813	Lauren	P0028994	F	18-25	24	0	Andhra Pr	Southern	Govt	Auto	2	23664		
18	1005447	Amy	P0027564	F	46-50	48	1	Andhra Pr	Southern	IT Sector	Auto	3			
19	1001193	Mick	P0000484	F	26-35	29	0	Andhra Pr	Southern	Aviation	Auto	1	23619		
20	1001883	Praneet	P0002984	M	51-55	54	1	Uttar Prad	Central	Hospitality	Auto	1	23568		
21	1001883	Praneet	P0002984	M	51-55	54	1	Uttar Prad	Central	Hospitality	Auto	1	23568		
22	1000113	Ellis	P0018064	F	18-25	19	1	Andhra Pr	Southern	Govt	Auto	4	23546		
23	1000416	Hrisheeke	P0018184	F	46-50	46	1	Uttar Prad	Central	Banking	Auto	2	23525		
24	1005256	Grant	P0010174	F	26-35	30	0	Andhra Pr	Southern	IT Sector	Auto	1	23518		
25	1001505	Gilcrest	P0027184	F	51-55	53	0	Uttar Prad	Central	Automobil	Auto	2	23515		
26	1000900	Skaria	P0031784	M	55+	83	0	Karnataka	Southern	Automobil	Auto	3	23513		
27	1005908	Eric	P0028264	F	26-35	33	0	Andhra Pr	Southern	IT Sector	Auto	3	23462		
28	1001101	Gibson	P0023474	F	36-45	40	0	Uttar Prad	Central	Banking	Auto	3	23456		
29	1004736	Mahima	P0005804	F	18-25	25	1	Andhra Pr	Southern	Banking	Auto	4	23451		
30	1004037	Etezadi	P0019054	M	51-55	54	1	Andhra Pr	Southern	Govt	Hand & Pc	2	23434		
31	1002340	James	P0011964	F	36-45	39	1	Andhra Pr	Southern	Aviation	Auto	3	23389		
32	1005664	Dean	P0011164	F	18-25	20	0	Andhra Pr	Southern	Aviation	Auto	2	23365		
33	1002523	Aman	P0029334	F	26-35	32	1	Andhra Pr	Southern	Food Proc	Auto	3	23326		
34	1002503	Mousam	P0022004	F	36-45	36	0	Andhra Pr	Southern	Automobil	Auto	2	23314		
35	1002638	Damala	P0034624	F	26-35	35	1	Maharashi	Western	Media	Auto	2	23306		
36	1004505	Daniels	P0008004	F	51-55	55	1	Andhra Pr	Southern	Healthcare	Auto	4	23302		
37	1004957	Inderpreet	P0011184	M	26-35	27	1	Jharkhand	Eastern	Govt	Auto	1	23285		

Figure 5.1 Excel sheet



	A	B	C	D	E	F	G	H	I	J	K	L	M
11218	1000411	Prashant	P0031334	M	46-50	48	0	Gujarat	Western	IT Sector	Office	4	585
11219	1000526	Ashmi	P0011074	F	46-50	50	0	Delhi	Central	Textile	Office	2	584
11220	1003606	Meg	P0015754	M	26-35	32	0	Karnataka	Southern	Chemical	Office	4	582
11221	1004268	Rosenblatt	P0030294	M	46-50	48	0	Himachal I	Northern	Govt	Office	4	580
11222	1004451	Ricardo	P0003464	F	26-35	30	1	Delhi	Central	IT Sector	Office	2	579
11223	1005684	Mick	P0001684	F	18-25	23	0	Delhi	Central	Agricultur	Office	1	575
11224	1002004	Sheri	P0024614	F	46-50	48	0	Delhi	Central	Media	Office	3	575
11225	1001542	Buch	P0011364	F	18-25	20	1	Maharashi	Western	Aviation	Office	4	574
11226	1004378	Kritika	P0027404	F	18-25	18	0	Delhi	Central	Aviation	Office	2	572
11227	1005971	Rahul	P0030714	F	36-45	42	1	Delhi	Central	Agricultur	Office	1	572
11228	1001032	Geetanjali	P0024084	F	51-55	54	1	Delhi	Central	Lawyer	Office	3	570
11229	1004867	Chandni	P0017304	F	26-35	35	0	Delhi	Central	Healthcare	Office	1	569
11230	1001392	Andrew	P0009094	F	46-50	49	0	Uttar Prad	Central	Aviation	Office	1	569
11231	1000188	Khloe	P0008034	F	55+	80	0	Delhi	Central	Healthcare	Office	3	568
11232	1005258	Aromal	P0022044	F	36-45	37	1	Delhi	Central	Banking	Office	3	567
11233	1003557	Craven	P0004614	F	18-25	20	0	Delhi	Central	Healthcare	Office	3	563
11234	1001360	Darrin	P0011264	F	26-35	27	1	Delhi	Central	Chemical	Office	3	563
11235	1002106	Luke	P0016434	M	18-25	19	0	Maharashi	Western	Agricultur	Office	1	563
11236	1001628	Sandeep	P0022634	F	18-25	21	0	Delhi	Central	Aviation	Office	3	562
11237	1001248	Calhoun	P0025584	M	36-45	39	0	Andhra Pr	Southern	Healthcare	Office	3	560
11238	1002168	Hightower	P0014584	M	0-17	17	1	Himachal I	Northern	Agricultur	Office	4	560
11239	1000687	Neola	P0018154	M	26-35	29	1	Haryana	Northern	Media	Office	2	557
11240	1002718	Abhishek	P0000344	M	26-35	28	0	Karnataka	Southern	IT Sector	Office	1	555
11241	1000802	Marley	P0024564	F	26-35	33	0	Delhi	Central	Healthcare	Office	1	407
11242	1001425	Sudevi	P0004474	F	0-17	12	0	Delhi	Central	IT Sector	Veterinary	1	396
11243	1003032	Matthias	P0005804	F	26-35	33	0	Delhi	Central	Hospitality	Office	3	384
11244	1004344	Hildebran	P0018544	F	26-35	27	1	Delhi	Central	Healthcare	Office	2	382
11245	1005446	Sheetal	P0029774	M	51-55	53	0	Gujarat	Western	Healthcare	Office	1	382
11246	1005446	Sheetal	P0029774	M	51-55	53	0	Madhya Pi	Central	Healthcare	Office	2	382
11247	1004140	Bertelson	P0005744	F	26-35	31	1	Delhi	Central	Aviation	Office	2	381
11248	1000695	Manning	P0029694	M	18-25	19	1	Maharashi	Western	Chemical	Office	4	370
11249	1004089	Reichenba	P0017134	M	26-35	33	0	Haryana	Northern	Healthcare	Veterinary	3	367
11250	1001209	Oshin	P0020134	F	36-45	40	0	Madhya Pi	Central	Textile	Office	4	213
11251	1004023	Noonan	P0005944	M	36-45	37	0	Karnataka	Southern	Agricultur	Office	3	206
11252	1002744	Brumley	P0028174	F	18-25	19	0	Maharashi	Western	Healthcare	Office	3	188
11253													
11254													

Figure 5.2 Excel sheet

CHAPTER 6

FUTURE ENHANCEMENT

A lot of future work can be done in this area. Analysts are just now discovering the potential behind Exploratory data analysis and analysis for this project is very simple.
Future work can be done on providing more detailed information of the data where your stakeholder understands the data more clearly by the help of more charts and graphs. A user interface can also be developed for the Analysis so that the user can easily understand how the process work.
In the near future, Exploratory dta analysis will be one of the important technique in the lead generation process. The web scraping tool can make market research of the particular product/services and enormous benefits to offer in the marketing field.

- Increasing the number of data-sets to increase the amount and accuracy of data in the results.
- Can illustrate via graphics in future, by creating the possibilities of extracting graphics.
- Can work on the overall optimization of of the project by reducing the time.
- Can work on using more websites for a variety and the quantity of the data to be extracted.

CHAPTER : 7

CONCLUSION

Exploratory Data Analysis is quite clearly one of the important steps during the whole process of knowledge extraction.

The purpose of this minor project was to explore the data and visualize it in form of graphs and charts to make it understand better. To systematically arrange the data are provided for research purposes. This program requires good knowledge in python libraries like Pandas, Matplotlib, Seaborn, Numpy and application like Excel.

CHAPTER- 8

BIBLIOGRAPHY

www.geeksforgeeks.org