School of Computing
Department of Computer Science & Engineering
(Artificial Intelligence and Machine Learning)

ACADEMIC YEAR 2025-26 (SUMMER SEMESTER)

LAB RECORD NOTEBOOK

10212CA214 - DATA VISUALIZATION

NAME: K. Rohith

VTU.NO: 26001

REG.NO: 23UECL0024

BRANCH: CSE [AI&ML]

YEAR/SEM: III rd year / VI th Sem

SLOT: S12L1

Vel Tech
Rangarajan Dr. Sagunthala
R&D Institute of Science and Technology

School of Computing
Department of Computer Science & Engineering
(Artificial Intelligence and Machine Learning)

ACADEMIC YEAR 2025-26 (SUMMER SEMESTER)

BONAFIDE CERTIFICATE

NAME	: K. Rohith	**BRANCH**	: CSE [AI & ML]
VTU NO.	: 26001	**REG.NO.**	: 23UECL 002U
YEAR/SEM	: III rd year / V th Sem	**SLOT NO.**	: S12L1

Certified that this is a bonafide record of work done by above student in the **"10212CA214 - DATA VISUALIZATION LABORATORY"** during the year 2025-2026 (Summer Semester).

R.T. 29/10/25

SIGNATURE OF LAB HANDLING FACULTY

SIGNATURE OF HOD

Submitted for the Semester Practical Examination held on **03.11.25** at Vel Tech Rangarajan Dr.Sagunthala R&D Institute of Science and Technology.

INTERNAL EXAMINER

EXTERNAL EXAMINER

INDEX

No	Date	Title	Page No.	Marks	Faculty Signature
1.	21/07/25	Exploration of Data Visualization Tools like Tableau, Python libraries, D3.js	1-6	17	21/7
2.	28/07/25	To visualize and perform Univariate analysis using continuous and categorical data	7-11	17	28/7
3.	04/08/25	To visualize and perform Bivariate analysis using continuous and categorical data	12-16	16	4/8
4.	11/08/25	To visualize and perform Multivariate analysis using Multiple variables involving Multiple measures	17-19	18	11/8
5.	18/08/25	To design and perform visualization for Trees	20-23	20	18/8
6.	25/08/25	To design and perform visualization for Graphs and Networks	24-26	18	25/8
7.	08/09/25	To generate insight using Text Network Analysis and Visualization	27-30	18	8/9
8.	15/09/25	To analyze and visualize Spatial and Geospatial data	31-32	20	17/9
9.	29/09/25	To analyze and visualize Time Oriented Data	33-36	19	29/9
10.	13/10/25	Use Case: Call - Tome - Analyses	37-39	18	13/10
11.	27/10/25	Use Case: Health-care Analysis	40-42	18	27/10

Completed

Total Marks: 199 /220

Signature of Faculty

Output:

Program Explanation

1. Pandas is a powerfull data manipulation and analysis library
2. pd.read_csv reads the csv file
3. df.head() shows the first 5 rows of the dataset
4. df.info() gives a summary of the dataset
5. len(df.columns) returns the total no. of columns.
6. df.types displays the datatype of each column
7. sort.values sorts the dataframe
8. df.unique shows the no. of unique values in each column.
9. df.describe() returns sum mary stats or numerical column
10. df.shape gives a tuple (rows, columns) representing.

Task-1A Retreiving The Data 21/7/25

Aim:- The Primary aim of this task is to Perform a Preliminary Exploratory data Analysis in the ELT Processed Diabetic Dataset

Algorithm:-

1. Load necessary Python libraries (Pandas, NumPy)
2. Read the diabetic dataset using Pandas (csv/Excel)
3. Display the first few rows to understand the data struct.
4. check each coloumn name and its datatype
5. Sort and show rows based on the first coloumn in both ascending and descending order.
6. Compute the number of unique values a cross the dataset.
7. Use Pandas 'describe' to compute statistics like count, mean, std, min, etc. for numerical features.
8. Get the shape of the dataset to know number of rows and coloumns.

Output:

```
=== First 5 Rows ===
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin
0	6	148	72	35	0
1	1	85	66	29	0
2	8	183	64	0	0
3	1	89	66	23	94
4	0	137	40	35	168

	BMI	DiabetesPedigreeFunction	Age	Outcome
0	33.6	0.627	50	1
1	26.6	0.357	31	0
2	23.3	0.672	32	1
3	28.1	0.167	21	0
4	43.1	2.288	33	1

```
=== DataFrame Info ===
```

#	Column	Non-Null Count	Dtype
0	Pregnancies	768 non-null	int64
1	Glucose	768 non-null	int64
2	BloodPressure	768 non-null	int64
3	SkinThickness	768 non-null	int64
4	Insulin	768 non-null	int64
5	BMI	768 non-null	float64
6	DiabetesPedigreeFunction	768 non-null	float64
7	Age	768 non-null	int64
8	Outcome	768 non-null	int64

```
=== Number of Columns ===
9
=== Column Datatypes ===
```

Pregnancies	int64
Glucose	int64
BloodPressure	int64
SkinThickness	int64
Insulin	int64
BMI	float64
DiabetesPedigreeFunction	float64
Age	int64
Outcome	int64

```
dtype: object
```


Program:

```
import pandas as pd
df = pd.read_csv("diabetic_data.csv")
Print("=== First 5 Rows ===")
Print(df.head(), "\n")
Print("=== DataFrame.Info ===")
df.info()
Print()
Print("=== Number of columns ===")
Print(len(df.columns))
Print("=== Column Data types ===")
Print(df.dtypes, "\n")
Print("=== Top 5 Rows Sorted by 'age' Ascending ===")
Print(df.sort_values by="age", ascending=True).head(), "\n")
Print("=== Top 5 Rows Sorted by 'age', ascending=False. head(), "\n")
Print("=== Unique values Per column ===")
Print(df.unique(), "\n")
total_unique = df.stack().nunique()
Print(f"=== Total unique values in Entire DataFrame: {total_unique} ===", "\n")
Print("=== Descriptive statistics ===")
Print(df.describe().T, "\n")
rows, cols = df.shape
Print(f"=== Dataset shape ===\n Rows: {rows}: {rows} \n columns: {cols}")
```

=== Top 5 Rows sorted by 'age' Ascending ===

	Pregnancies	Glucose	Blood Pressure	Skin Thickness	Insulin
736	0	126	86	27	120
738	2	99	60	17	160
721	1	114	66	36	200
60	2	84	0	0	0
290	0	78	88	29	40

	BMI	Diabetis Pedigree Function	Age	outcome
736	27.4	0.515	21	0
738	36.6	0.453	21	0
721	38.1	0.289	21	0
60	0.0	0.304	21	0
290	36.9	0.434	21	0

=== Top 5 Rows sorted by 'age' descending ===

	Pregnancies	Glucose	Blood Pressure	Skin Thickness	Insulin
459	9	134	74	33	60
453	2	119	0	0	0
666	4	145	82	18	0
684	5	136	82	0	0
123	5	132	80	0	0

	Diabetis Pedigree Function	Age	BMI
459	0.460	81	25.9
453	0.832	72	19.6
666	0.235	70	32.5
684	0.638	69	0.0
123	0.176	69	26.8

=== Unique values

Pregnancies	17
Glucose	136
BloodPressure	47
Skin Thickness	51
Insulin	186
BMI	248
Diabetis Pedigree Function	517
Age	52
outcome	2

dtype: int64

=== Total unique values in Entire DataFrame: 1005 ===

=== Descriptive statistics ===

	count	mean	std
Pregnancies	768.0	3.845052	3.369578
Glucose	768.0	120.894531	31.972618
BloodPressure	768.0	69.105469	19.355807
SkinThickness	768.0	20.536458	15.952218
Insulin	768.0	79.799479	115.244002
BMI	768.0	31.992578	7.884160
DiabetesPedigreeFunction	768.0	0.471876	0.331329
Age	768.0	33.240885	11.760232
Outcome			0.476951

min	25%	50%	75%	max
0.000	1.0000	3.0000	6.00000	17.00
0.000	99.00000	117.000	140.2500	199.00
0.000	62.00000	72.0000	80.0000	122.00
0.000	0.0000	23.0000	32.0000	99.00
0.000	0.0000	30.5000	127.25000	846.000
0.000	27.3000	32.0000	36.60000	67.10
0.078	0.24375	0.3725	0.62025	2.42
21.000	24.0000	29.0000	41.0000	81.00
0.000	0.0000	0.0000	1.0000	1.00

=== Dataset shape ===

Rows: 768

Columns: 9

Result: In this program, performed exploratory data analysis on the CLI-processed diabetic dataset by loading it into a dataframe and displaying its structure and contents.

output!

Program Explanation!

1. Pandas is a powerfull library for data analysis and manipulation.
2. pd.read_csv is used to read a csv file
3. df.head() displays the first 5 rows of dataset
4. df.isnull() creates a boolean dataframe with True for missing values.
5. dropna() removes any rows that contain null value.
6. df_cleaned.duplicated() returns a boolean series with True for duplicates
7. drop_duplicates() removes any repeated duplicated rows.

Task-1B

Aim!-

To perform data preprocessing on a dataset by identifying and handling null values and duplicate entries.

Algorithm:

1. Import necessary python libraries
2. load the dataset using Pandas.
3. Display the first 5 rows of the dataset
4. check for null values in each column.
5. Remove null values if any
6. Identify Duplicate rows
7. Remove Duplicate rows.

output:

First 5 rows of the dataset:

	Speal_length	Speal_width	Petal_length	Petal_width	Species
0	5.1	3.5	1.4	0.2	setose
1	4.9	3.0	1.4	0.2	setose
2	4.7	3.2	1.3	0.2	setose
3	4.6	3.1	1.5	0.2	setose
4	5.0	3.6	1.4	0.2	setose

Count of null values in each column:

Speal_length 0

Speal_width 0

Petal_length 0

Petal_width 0

species

dtype: int64

Data Frame after removing null values(if any):

	Speal_length	Speal_width	Petal_length	Petal_width	species
0	5.1	3.5	1.4	0.2	setose
1	4.9	3.0	1.4	0.2	setose
2	4.7	3.2	1.3	0.2	setose
3	4.6	3.1	1.5	0.2	setose
4	5.0	3.6	1.4	0.2	setose

Duplicate rows in the dataset:

	Speal_length	Speal_width	Petal_length	Petal_width	species
34	4.9	3.1	1.5	0.1	setose
37	4.9	3.1	1.5	0.1	setose

DataFrame after removing duplicates

	Speal_length	Speal_width	Petal_length	Petal_width	species
0	5.1	3.5	1.4	0.2	setose
1	4.9	3.0	1.4	0.2	setose
2	4.7	3.2	1.3	0.2	setose
3	4.6	3.1	1.5	0.2	setose
4	5.0	3.6	1.4	0.2	setose

Program:

```
import pandas as pd
df = pd.read_csv("https://raw.githubusercontent.com/luiuc_rsel
                  data-Raleigh_Page/data/iris.csv")
print("First 5 rows of the dataset")
print(df.head())
print("\n count of null values in each column:")
print(df.isnull().sum())
df_cleaned = df.dropna()
print("\n DataFrame after removing null values:")
print(df_cleaned.head())
duplicates = df_cleaned[df_cleaned.duplicated()]
print("\n Duplicate rows in the dataset:")
print(duplicates)
df_no_duplicates = df_cleaned.drop_duplicates()
print("\n Data Frame after removing duplicates:")
print(df_no_duplicates.head())
```

VELTECH	
EX No.	1
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	5
TOTAL (20)	25
SIGN WITH DATE	21/8/25

Result:- Thus to perform data preprocessing on a dataset by identifying and handling null values and duplicates are done and executed successfully.

04/8/25

Task 2:- To visualize and perform univariate analysis using continuous and categorical data

categorical Data: Bar chart, Pie chart

continuous Data: Scatterplot, line plot, Strip Plot, Swarm plot, Histogram, Density Plot, Rug plot

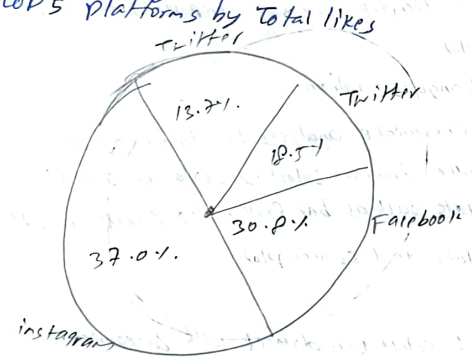
Tools: Table IV, Language Python

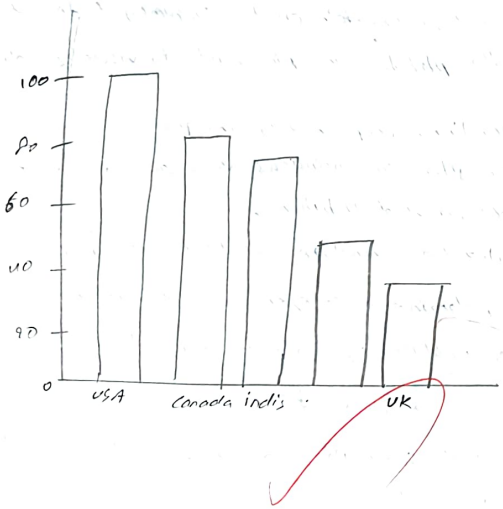
Aim:- To perform univariate analysis by identifying categorical and continuous attributes from a selected dataset and visualizing the insights using plots such as bar char, Pie charts, scattered plot, line plot, strip plot, and swarm plot

Algorithm:

1. Choose Dataset:- select a dataset with categorical and continuous attributes.
2. Identify Data types: Distinguish blw categorical (non-numeric) and continuous (numeric) attributes.
3. Construct Bar and pie charts:- create frequency tables for categorical data, then plot bar and pie charts to visualize distribution.
4. Construct Scatter, line, strip and swarm plots:- Utilize appropriate plots for continuous data and analysis considering relationships, trends, and distributions.
5. Interpret visualizations:- Analyze insights gained from visualization identify patterns, dominate categories and correlations.
6. Consider Further Analysis: Reflect on implications for further exploration or decision-making based on the analysis
7. Document Findings: summarize findings and observations for reporting or presentation purposes

output:

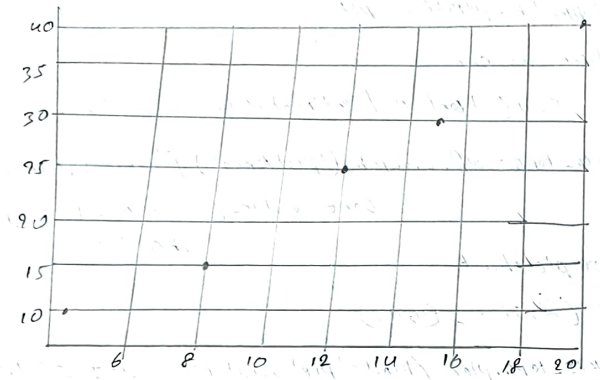




Program:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv("/content/sentiment dataset.csv")
platform_likes_tops = df.groupby('Platform')['Likes'].sum().
                      sort_values(ascending=False)
                      .head(5)
#create a pie chart
plt.figure(figsize=(10,10))
platform_likes_tops.plot(kind='pie', autopct='%1.1f%%',
                         startangle=140, colors=['skyblue','lightcoral',
'lightgreen','lightsalmon','lightsteelblue'])
plt.title("Top5 platforms by Total likes")
plt.ylabel("")
plt.show()
#top5_countries = df.nlargest(5,'likes')
#create a bar chart
plt.figure(figsize=(10,6))
plt.bar(top5_countries['Country'], top5_countries['likes'],
        color=['lightsalmon','lightsteelblue'])
plt.xlabel('Country')
plt.ylabel("Total likes")
plt.title("Top5 countries by Total likes")
plt.show()
data = {
'Text': ['Enjoying a beautiful day at the park!', 'Traffic was
Terrible this morning', 'Just finished an amazing workout',
'Excited about upcoming weekend getaway', 'Trying out a
```

Scatter plot of Retweet vs likes



o/p

new recipe for dinner]

'Retweets': [15, 5, 20, 8, 12],

'Likes': [30, 10, 40, 15, 25]

}

df = pd.DataFrame(data)

#set the seaborn style

plt.figure(figsize=(10,6))

sns.scatterplot x = 'Retweets', y = 'Likes', data=sns.scatter

plot, data = df, color = 'blue', 'alpha'=0.7)

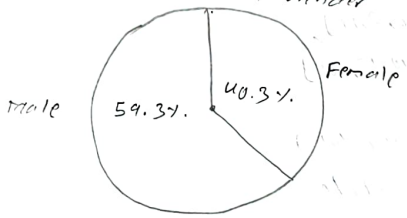
plt.title("Scatter plot of Retweets vs Likes")

plt.show()

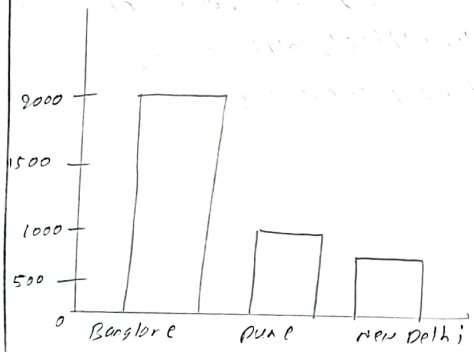
Result: Thus to perform univariate calculas analysis by identifying categorical and continuous attributes and visualizing the insights are done and executed successfully.

output:

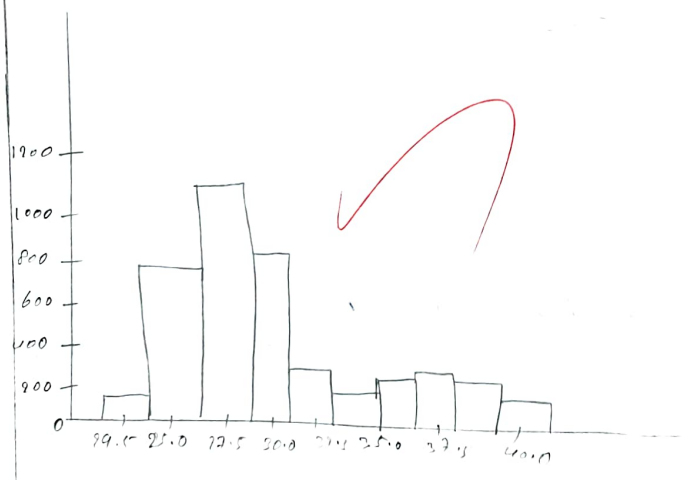
Pie chart of Gender



Bar chart of City



Age Distribution



Task-2b

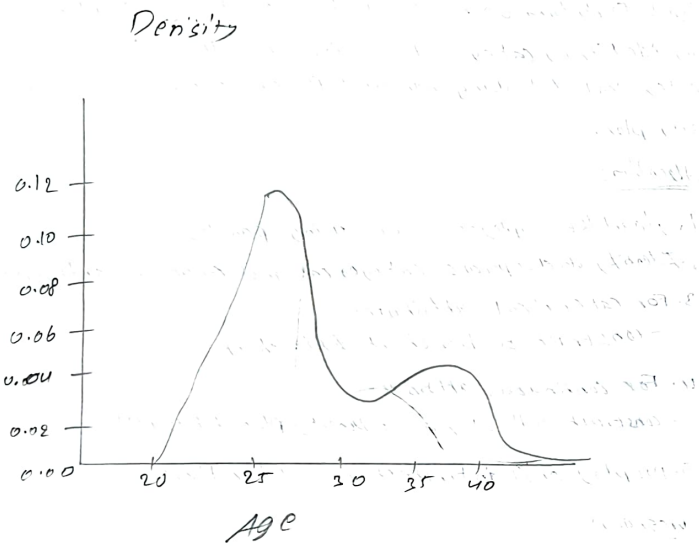
Aim: To perform univariate analysis on a given employee dataset by identifying categorical and continuous attributes and visualizing categorical data using bar and pie charts continuous data and rug plot.

Algorithm:

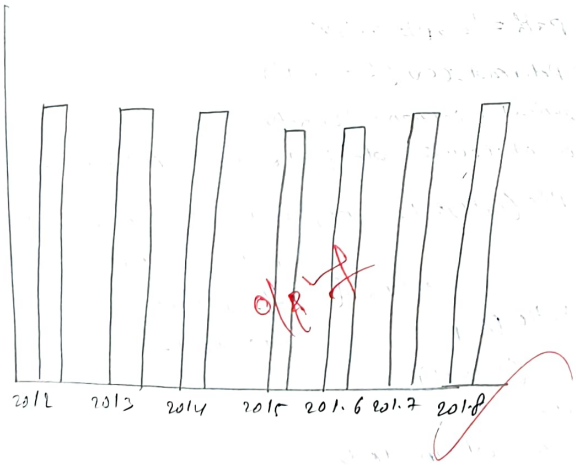
1. Load the employee dataset using pandas
2. Identify and seperate categorical and continuous attributes.
3. For categorical attributes:
   - construct a bar chart & pie chart
4. For continuous attributes
   - construct a histogram & Density plot & Rug plot
5. Display and interpret each visualization

Program:

```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
file_path = 'Employee.csv'
df = pd.read_csv(file_path)
categorical_column = 'Gender'
category_counts = df[categorical_column].value_counts
plt.pie(category_counts, labels = category_counts.index
                                autopct = '%0 1.1f%%.', startangle=90,
                                colors=['skyblue', 'light coral'])
plt.title('Piechart of {categorical_column}')
plt.axis('equal')
plt.show()
categorical_column = 'city'
categorical_counts = df[categorical_column].value_counts()
plt.bar(category_counts.index, category_counts,
color=['skyblue', 'light coral', 'lightgreen', 'orange'])
```



Rug plot of Joining year




```
plt.title(f'Bar Chart of {categorical_column}')
plt.xlabel(categorical_column)
plt.ylabel('Count')
plt.show()
continuous_column = 'Age'
plt.hist(df[continuous_column], bins=10, edgecolor='black')
needed. bins = blue bars
plt.title(f'{continuous_column} Distribution')
plt.xlabel(continuous_column)
plt.ylabel('Frequency')
plt.show()
continuous_column = 'Age'
sns.kdeplot(df[continuous_column], fill=True)
plt.title(f'Density plot of {continuous_column}')
plt.xlabel(continuous_column)
plt.ylabel('Density')
plt.show()
continuous_column = 'Joining year'
sns.rugplot(df[continuous_column], height=0.5)
plt.title(f'Rug plot of {continuous_column}')
plt.xlabel(continuous_column)
plt.yticks([])
plt.show()
```

VEL TECH	
EX No.	12
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	4
RECORD (5)	4
TOTAL (20)	13+4=17
SIGN WITH DATE	

Result: Thus to perform univariate analysis on a given employee dataset by identifying categorical and continuous attributes and visualizing are done and executed successfully.

4/8/25

Task-3: To visualize and perform Bivariate analysis using continuous and categorical data

categorical vs categorical: stacked Barchart, Grouped Barchart, Segmented Barchart, Mosaic plots.

continuous vs continuous: scatterplot Fit-lines

categorical vs continuous: Bar chart (Summary statistics), Grouped Kernel density plots, Box plots, violin plots, Ridgeline plots, Beeswarm plots.

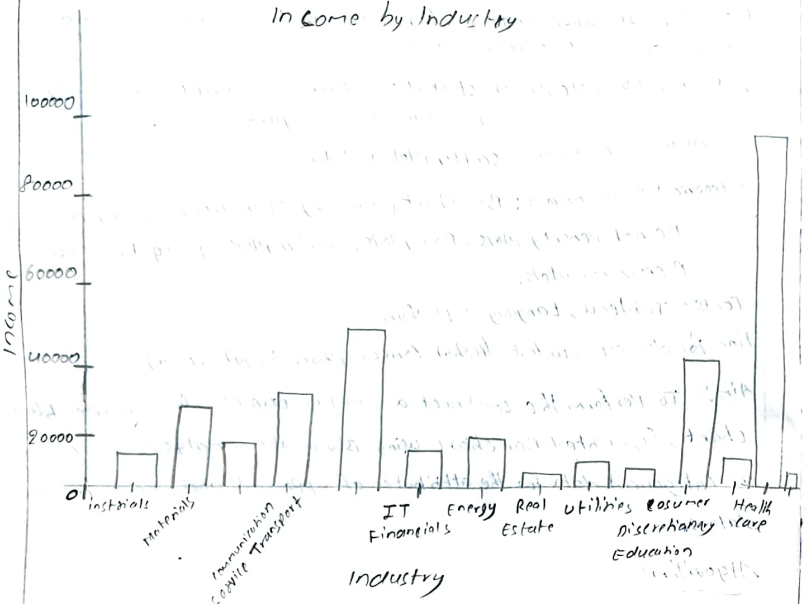
Tools: Tableau, Language: Python

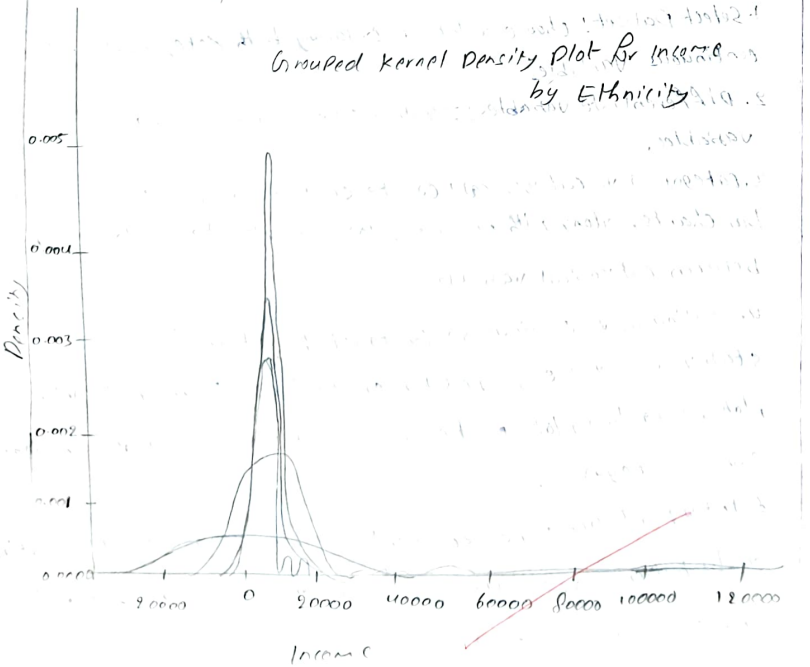
link for dataset: Student Alcohol Consumption (kaggle.com)

Aim:- To Perform the construct a stacked Bar chart, Grouped Bar chart, Segmented Bar chart using Bivariate analysis of categorical vs categorical data for the attributes of approved and gender in above dataset.

Algorithm:-

1. Select Dataset: choose a dataset containing both categorical and continuous variables.
2. Differentiate variables: Identify categorical and continuous variables.
3. categorical vs categorical: create stacked, grouped or segmented bar charts, along with mosaic plots, to visualize relationship between categorical variables.
4. continuous vs continuous: construct bar charts for summary statistics and use grouped kernel density plots, box plots, violin plots, ridgeline plots, or beeswarm plots to visualize distributions across categories.
6. Interpretation: Analyze visualizations for insights into relationships, distributions, and patterns, drawing conclusions for further analysis or decision-making

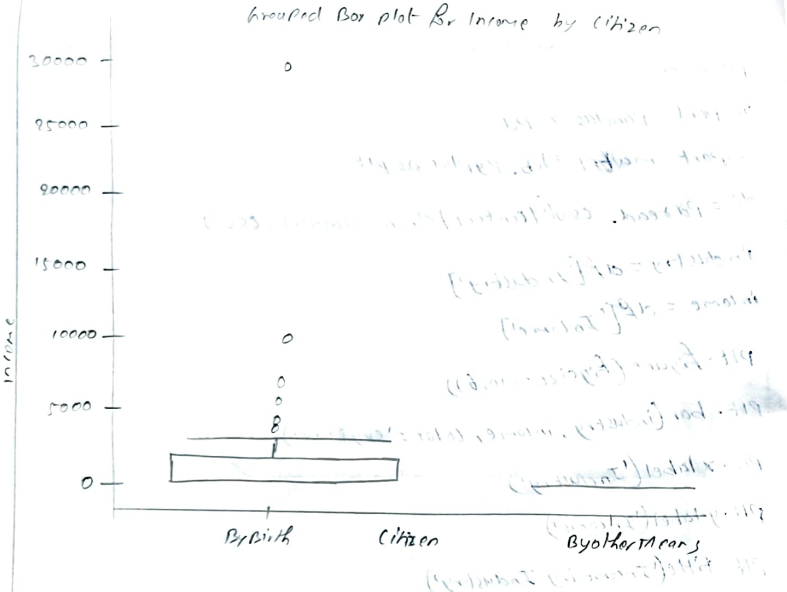


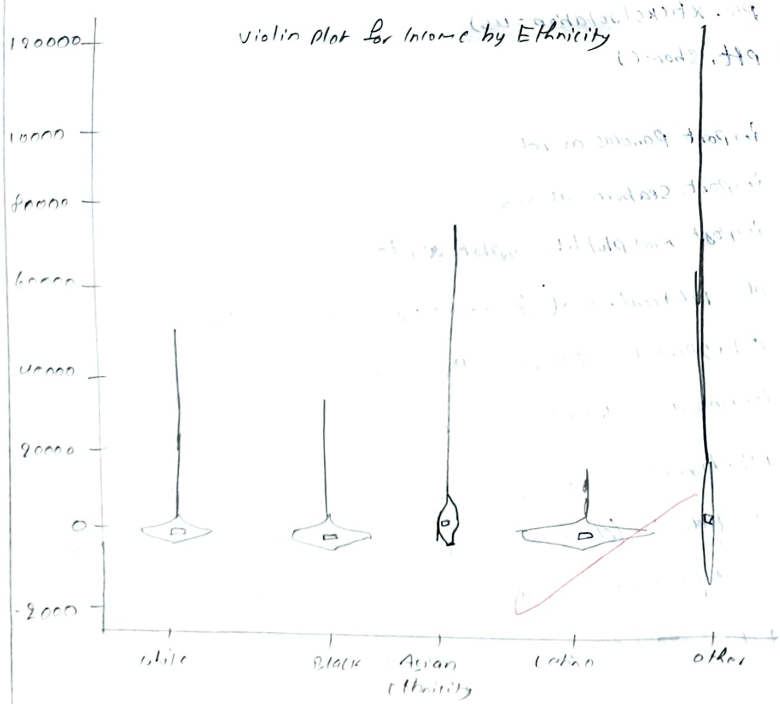


Program:-

```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('/content/clean_dataset.csv')
industry = df['Industry']
income = df['Income']
plt.figure(figsize=(10,6))
plt.bar(industry, income, color='sky blue')
plt.xlabel('Industry')
plt.ylabel('Income')
plt.title('Income by Industry')
plt.xticks(rotation=45)
plt.show()
```

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
df = pd.read_csv('/content/clean_dataset.csv')
categorical_column = 'Ethnicity'
continuous_column = 'Income'
plt.figure(figsize=(12,8))
sns.kdeplot(data=df, x=continuous_column, hue=categorical_
    column, fill=True, common_norm=False)
plt.xlabel(continuous_column)
```



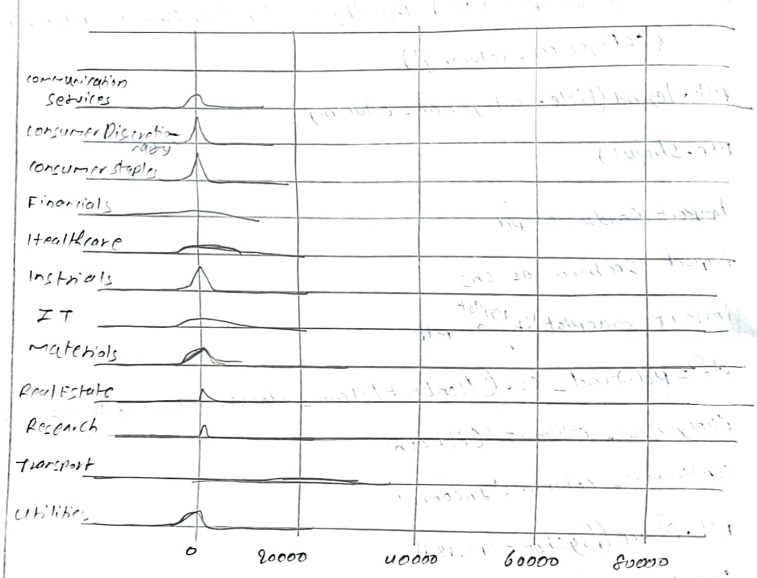


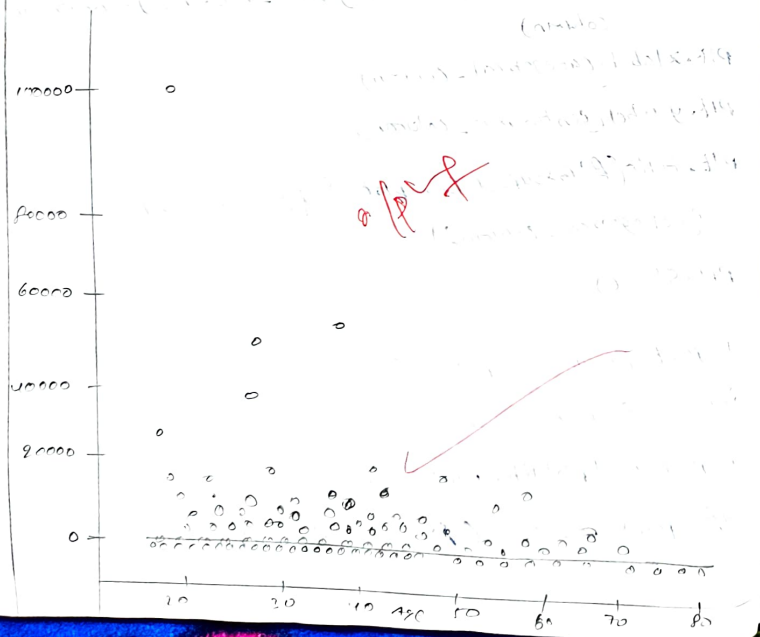

```
plt.title(f'Grouped Kernel Density plot for {continuous_column} by
    {categorical_column}')
plt.legend(title=categorical_column)
plt.show()

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
df = pd.read_csv('/content/clean_dataset.csv', nrows=50)
categorical_column = 'Citizen'
continuous_column = 'Income'
plt.figure(figsize=(16,12))
sns.boxplot(data=df, x=categorical_column, y=continuous_
    column)
plt.xlabel(categorical_column)
plt.ylabel(continuous_column)
plt.title(f'Grouped Box plot for {continuous_column} by
  {categorical_column}')
plt.show()

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
df = pd.read_csv('/content/clean_dataset.csv')
```

Ridgeline Plot for Income by Industry






```
categorical_column = 'Ethnicity'
continuous_column = 'Income'
plt.figure(figsize=(12,8))
sns.violinplot(data=df, x=categorical_column, y=continuous_column)
plt.xlabel(categorical_column)
plt.ylabel(continuous_column)
plt.title(f'Violin Plot for {continuous_column} by {categorical_column}')
plt.show()

import pandas as pd
from joypy import joyplot
import matplotlib.pyplot as plt
df = pd.read_csv('/content/clean_dataset.csv')
categorical_column = 'Industry'
continuous_column = 'Income'
plt.figure(figsize=(12,8))
joyplot(
    data=df, by=categorical_column, kind='kde', fill=True,
    linecolor="black", grid=True, linewidth=1, legend=True,)
plt.xlabel(continuous_column)
plt.title(f'Ridgeline Plot for {continuous_column} by {categorical_column}')
plt.show()
```

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
df = pd.read_csv ('/content/DV task-4.csv')
sns.lmplot(data = df, x = 'Age', y = 'Income', height = 6, line_
                                        kws = {'color': 'red'})
plt.title('Scatter Plot with Fit-line')
plt.xlabel('Age')
plt.ylabel('Income')
plt.show()
```

VEL TECH	
EX No.	3.
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	3
RECORD (5)	+3
TOTAL (20)	13+3=16
SIGN WITH DATE	8/2

14/8/21

Result: Thus the Perform visualize and Perform Bivariate analysis using continuous and categorical data attributes are done and executed successfully.

Task-4:- To visualize and perform multivariate analysis using Multiple variables involving multiple measures
Scatterplot Matrix, Parallel coordinates, Line Graph, Stacked Bar chart

Tools: Tableau, Language: Python

Aim:- To perform this dataset contains a cleaned a cleaned version of this dataset from UCI machine learning repository on credit card approvals. Missing values have been filled and feature names and categorical names have been inferred, resulting in more context and it being easier to use.

Algorithm:-

1. Import necessary libraries including numpy, Pandas, seaborn, matplotlib.pyplot, and plotly.express.
2. Load the dataset from a zip file using Pandas 'read_csv' function.
3. Print the loaded dataset to inspect its structure and contents.
4. Select numeric columns (Age, Debt, YearsEmployed), create a pairplot using seaborn to visualize pairwise relationships, and display the plot.
5. Use plotly.express to create a parallel coordinates plot, color-coded by the 'Approved' column, with dimensions as Age, Debt and Credit Score, and show the plot.
6. Extract the first 20 entries of the dataset and plot the Age against Debt and CreditScore using a line plot with different colors for Debt and CreditScore, then display the plot.

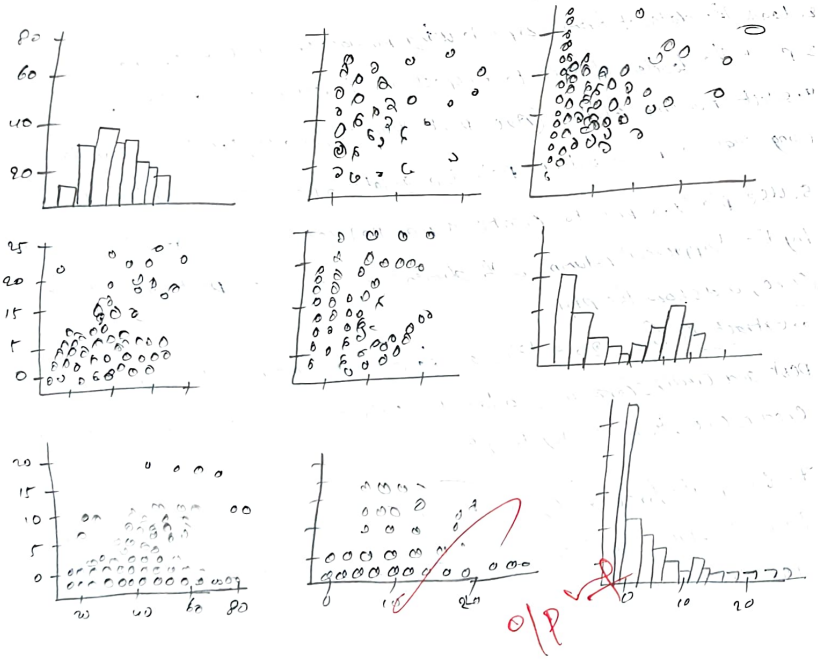
7. Group the first 20 entries by Age, summing up Debt and CreditScore for each Age group, create a stacked bar chart representing Debt and CreditScore for each Age group, and display the chart.

Output:-

	Gender	Age	Debt	Married	BankCustomer	Industry	Ethnicity
0	1	30.83	0.000	1	1	Industrials	White
1	0	58.67	4.460	1	1	Materials	Black
2	0	24.50	0.500	1	1	Materials	Black

YearsEmployed	PriorDefault	Employed	CreditScore	DriverLicense	Citizen
1.25	1	1	1	0	By Birth
3.04	1	1	6	0	By Birth
1.50	1	0	0	0	By Birth

ZipCode	Income	Approved
202	0	1
43	560	1
280	824	1



O/P

Program:-

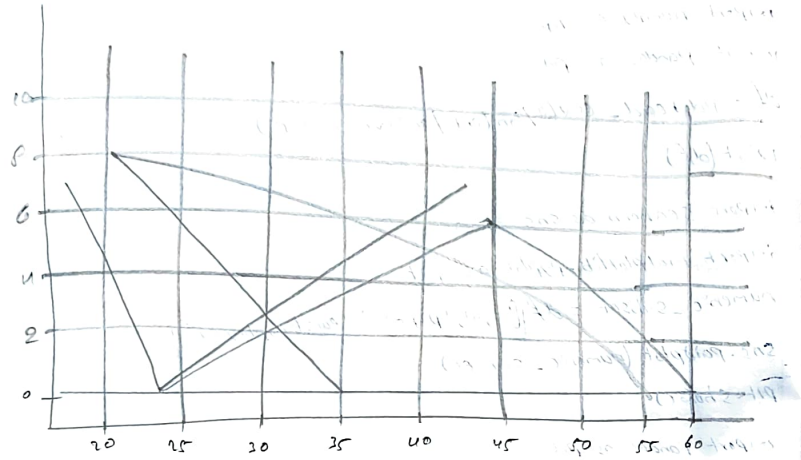
```
import numpy as np
import pandas as pd
df = pd.read_csv("/content/archive(1).zip")
print(df)

import seaborn as sns
import matplotlib.pyplot as plt
numeric_subset = df[['Age', 'Debt', 'YearsEmployed']]
sns.pairplot(numeric_subset)
plt.show()

import pandas as pd
import plotly.express as px
attributes = ["Age", "Debt", "CreditScore"]
fig = px.parallel_coordinates(df, color="Approved", dimensional=
attributes, color_continuous_scale=px.colors.diverging.Tealrose, color_
continuous_midpoint=0.5)

fig.show()

first_20_entries = df.head(20)
age = first_20_entries['Age']
debt = first_20_entries['Debt']
credit_score = first_20_entries['CreditScore']
plt.figure(figsize=(10,6))
plt.plot(age, debt, label='Debt', color='blue')
plt.plot age, credit_score, label='CreditScore', color='green')
plt.xlabel('Age')
plt.ylabel('Value')
plt.title('Line Graph: Age, Debt, and CreditScore')
plt.legend()
plt.grid(True)
```




```
plt.show()
grouped_data = first_20_entries.groupby('Age').sum()[['Debt',
'Credit Score']]
grouped_data.plot(kind='bar', stacked=True, figsize=(12,8))
plt.xlabel('Age')
plt.ylabel('Value')
plt.title('Stacked Bar chart: Debt and Credit Score by Age')
plt.legend(title='Attribute')
plt.show()
```

VEL TECH	
ER No.	4
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	3
RECORD (5)	5
TOTAL (20)	3+5=18
...N WITH DATE	

4/8

Result: Thus the Perform of the visualize and Perform multivariate analysis using multiple variables involving multiple measures are done and executed successfully.

Task-5:- To design and perform visualization for Trees

* Tree Map, Sun Burst Display

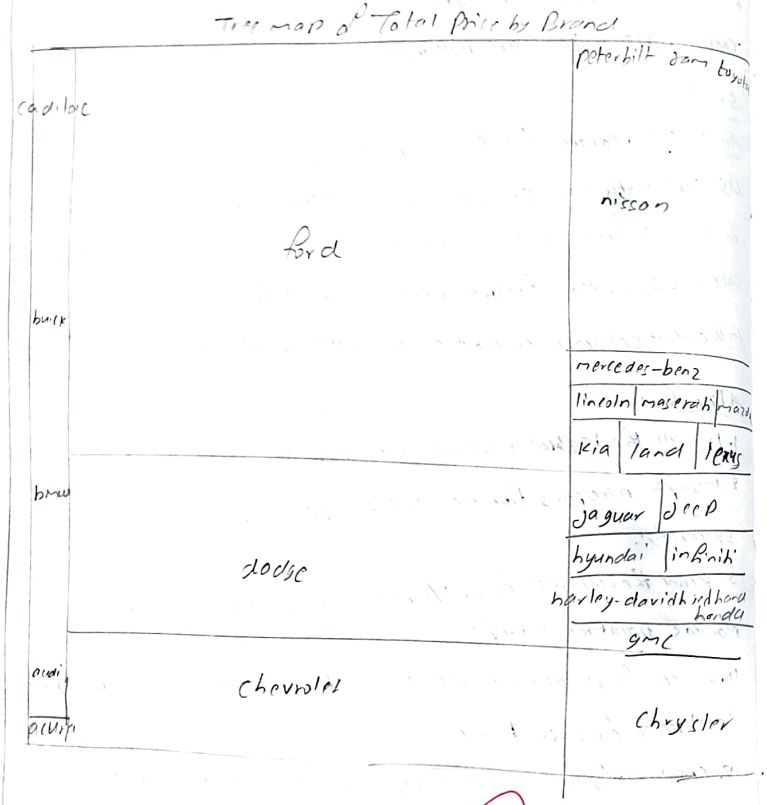
Tools: Tableau, language: Python

5a):-

Aim:- To perform the treemap display on a real-word dataset. US car's data was scraped from AUCTION EXPORT.com. This dataset included information about 28 brands of clean and used vehicles for sale in US. Twelve features were assembled for each car in the dataset. In the data set you can take any attributes, values and visualize it.

Algorithm:

1. Install the 'squarify' package using pip.squarify
2. Import necessary libraries such as 'pandas', 'matplotlib.pyplot' and squarify'.
3. Read the dataset form '/content/USA_cars_datasets.csv' into a Pandas DataFrame 'df'.
4. Group the DataFrame 'df' by the 'brand' column and calculate the total price for each brand.
5. Create a figure of size 18X12 inches for the tree map visualization.
6. Use 'squarify.plot()' to create a treemap plot.
7. Pass the sizes of the rectangles as the total price of each brand (brand_total_price['price']) and labels as the brand names (brand_total_price['brand'])
8. Set transparency with alpha=0.4 for better visualization.
9. Turn off the axis.
10. Set the title of the plot as 'Treemap of Total Price by Brand'.
11. Display the plot using plt.show()



Program:

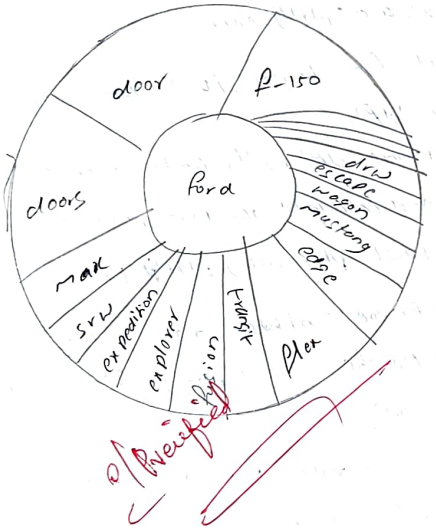
```
pip install squarify
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import squarify
df = pd.read_csv('/content/USA_cars_datasets.csv')
print(df)
brand_total_price = df.groupby('brand')['price'].sum().reset_index()
plt.figure(figsize=(18,12))
squarify.plot(size=brand_total_price['price'], label=brand_
  total_price['brand'], alpha=0.4)
plt.axis('off')
plt.title('Tree Map of Total Price by Brand')
plt.show()
```

5b :-

Aim :- To perform the build a sun burst display using above program dataset.

Algorithm :-

1. Install the required packages 'squarify' and 'plotly' using pip.
2. Import necessary libraries: 'pandas', 'numpy', 'matplotlib.pyplot', 'seaborn', 'squarify', and 'plotly.express'.
3. Read the dataset from '/content/USA_cars_dataset.csv' into a Pandas DataFrame 'df'.
4. Group the DataFrame 'df' by the combination of 'brand' and 'model' columns and calculate the total price for each combination.
5. Create a sunburst plot using plotly Express ('px.sunburst()'):
   - Specify the DataFrame 'sunburst_data' as the data source.
   - Set the path of the sunburst plot to follow the hierarchy of 'brand' and 'model'.
   - Define 'price' as the values to be represented.
   - Set the title of the plot as 'Sunburst Display of Total Price by Brand and Model'.
6. Display the plot using 'fig.show()'.



Program:

```
pip install squarify
pip install plotly
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import squarify
import plotly.express as px
df = pd.read_csv('/content/USA_cars_datasets.csv')
print(df)
sunburst_data = df.groupby(['brand', 'model']).sum().reset_index()
fig = px.sunburst(sunburst_data, path=['brand', 'model'], values='price', title='Sunburst Display of Total Price by Brand and Model')
fig.show()
```

VEL TECH	
EX No.	5
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	5
TOTAL (20)	5+5+5+5=20
SIGN WITH DATE	

Result: Thus the Perform of the to design and perform visualization for Trees are done and executed successfully.

25/8/25

Task!-6 : To design and Perform visualization for Graphs and network

-> Force based layout

Tools : Tableau, Language : Python

Aim!- To perform utilize a given dataset link it contains network links, source and target technical tags, and the link value blw each pair. It also contains the nodes of the network, the name of each node, the group to which this node belongs and a node size based on the frequency of use of this technological becon.

Algorithm!-

Import necessary libraries:

Import Network X as nx for Creating and manipulating network graphs.

Import Matplotlib.pyplot as plt for data visualization.

Import Pandas as pd for data handling.

Create an empty graph object G and set an attribute 'day' to "Stack over flow"

Read node and edge data from csv files ('stack_network_nodes.csv' and 'stack_network_links.csv') into DataFrames df nodes and df links, respectively.

Add nodes to the graph G:

Iterate through the rows in df nodes.

For each row, add a node to the graph with a 'name', 'group' and 'nodesize' attributes.

Add weighted edges to the graph G:

Iterate through the rows in df links.

For each row, add a weighted edge color, width, node labels, and font weight.

set visualization parameters:

Create a figure with a specified size.

Define visualizations options like edge color, width, node labels and font weight.

Determine colors for nodes based on their 'group' attribute.

Adjust node size based on the 'nodesize' attribute.

Draw the network graph:

use nx.draw to visualize the graph.

Customize the node colors and sizes based on the calculated values.

Define the layout of the nodes using spring layout with specified parameters.

Set the edge color to a specific color.

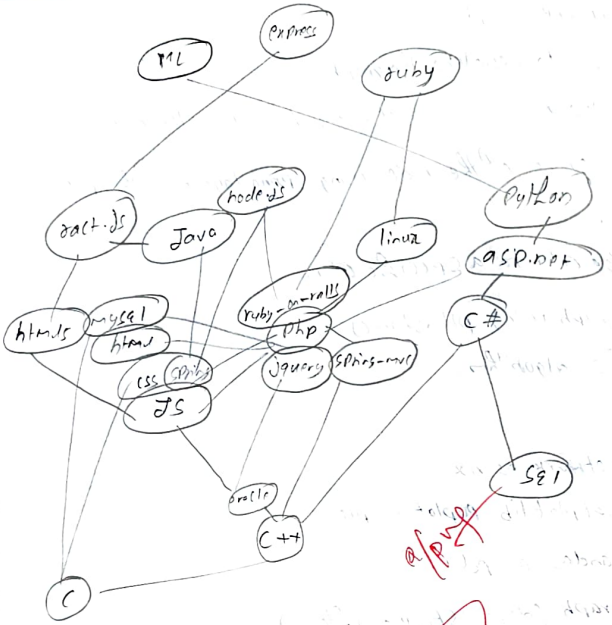
Show the graph using plt.show()

End of the algorithm.

Code:

```
import network as nx
import matplotlib.pyplot as plt
import pandas as pd
G = nx.Graph (day = "stackoverflow")
df_nodes = pd.read_csv('stack_network_nodes.csv')
df_links = pd.read_csv('stack_network_links.csv')
for index, row in df_nodes.iterrows():
G.add node (row['name'], group = row['group'], nodesize = row
                                                ['nodesize'])
for index, row in df_links.iterrows():
G.add weighted edges from([(row['source'], row['target'], row
                                                ['value'])])
plt.figure(fig size = (15, 15))
options = {
  'edge color': '#FFDEA2',
  'width': 1.5,
  'with_labels': True,
  'font_weight': 'regular',
  }
colors = [color_map[G.nodes[node]['group']] for node in G]
```

Output:-




```
sizes =[G.nodes [node] ['nodesize'] * 25 for node in G]

nx.draw(G, node_color = colors, node_size = sizes, pos = nx.spring
        layout(G, K=1.5, iterations = 15), ** options)

ax = plt.gca()

ax. collections[0]. set_edgecolor("#555555")

plt.show()
```

PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	4
RECORD (5)	4
TOTAL (20)	4+4=(8
SIGN WITH DATE	

Result:-

Thus, the design and Perform visualizations for graphs and networks are executed successfully.

1/9/25

Task-7:- To generate insight using Text Network Analysis and visualization.

Tools: Wordle, Tag cloud, Word Tree, Infra Nodus

7a) Tag Crowd tool using Build a word cloud it contains your details.

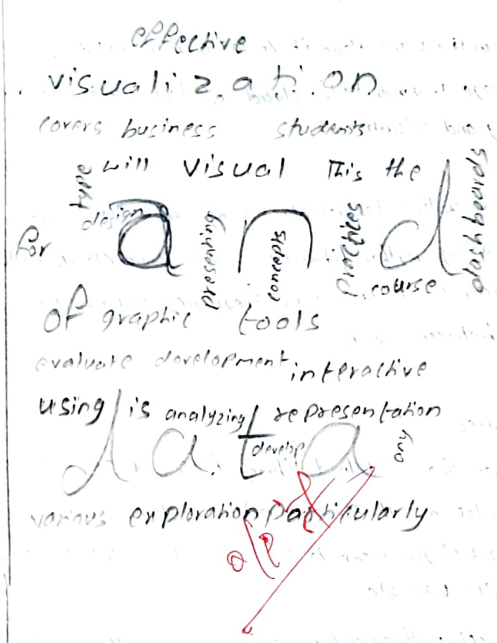
7b) Utilize wordtree package to generate the cloud of text and to plot graph using matplotlib library and visualize it.

Aim:- To Perform the to visualize textual data using word cloud and word Tree technique by leveraging Python libraries such as word cloud, networkx, and matplotlib, in order to analyze word frequency and explore contextual relationships between words.

Algorithm:-

1. Import necessary libraries:
2. Import word cloud from the wordcloud library for word cloud creation.
3. Import matplotlib.pyplot as plt for data visualization.
4. Define the text from which you want to create a word cloud. In this case, it's stored in the details variable.
5. Create a word cloud object with specific parameters: Set the width and height of the word cloud image.
5. Specify the background color (white in this case).
6. Provide a list of stopwords to execute exclude common words (an empty list in this case). set the minimum font size for displayed words. Create a matplotlib figure for displaying the word cloud:
7. Define the figure size (5x5 inches in this case) and specify the face-color (None, which is transparent). Generate the word cloud image using the generate method of the word cloud object, passing in the text from step 2.
8. Display the word cloud image:
9. Use plt.imshow to display the word cloud. Turn off the axis labels. Adjust the layout to minimize padding. Show the word cloud using plt.show().
10. End of the algorithm.

output!



code!

```
from wordcloud import WordCloud
import matplotlib.pyplot as plt
details = "data visualization is the visual and interactive exploration
and graphic representation of data of any type. This course covers data
visualization concepts, practices, and tools particularly for analyzin.
g and presenting business data. Students will evaluate, design, and
develop effective visualizations and dashboards, using various
development tools"
wordcloud = WordCloud(width =800, height =800,
background_color = 'white',
stopwords = [],
min_font_size =10).generate(details)
plt.figure(figsize =(5,5), facecolor =None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad =0)
plt.show()
```

Result!. Thus the generate insight using Text network Analysis and visualization of the word cloud was executed successfully.

7b)

Aim:- To generate a word cloud using the tag cloud/word cloud approach that visually represents the frequency of words from a given text containing personal details.

Algorithm:-

1) Start
2. Take a text document (or) Paragraph as input.
3. Convert the text to lowercase for uniformity.
4. Count the frequency for each remaining word
5. Generate the word cloud where word size = frequency
6. Display the visualization.
7. Stop

Program:-

```
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
text = """
My name is Rohith,
I am Pursuing B.Tech in CSE (AI & ML),
I am Skilled in Python, Machine Learning, Data visualization, and
web development,
I am passionate about AI, Data Science, & Competitive Programmin
g,
"""
stopwords = set(STOPWORDS)
wordcloud = WordCloud width = 800, height = 400,
                background_color = "white",
                colormap = 'plasma',
                collection = False).generate(text)
plt.figure(figsize=(10,5))
plt.imshow(wordcloud, interpolation = 'bilinear')
```

output:

computer
Python
Data machine
Science Learning
web passionate
development competitive

plt.axis('off')
plt.title("Word cloud - Personal Details", fontsize=16)
plt.show()

VEL TECH	
EX No.	7
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	3
RECORD (5)	5
TOTAL (20)	18
SIGN WITH DATE	

Result: Thus, the Generation of insight text network plot graph using matplotlib and visualization is executed successfully.

Task 8 :- To analyze and visualize spatial and geospatial data

8/9/25

Aim :- To analyze the population statistics of a region using spatial and geospatial techniques and to visualize them on maps.

Algorithm :-

1. Import the dataset containing population statistic across census years.
2. Process data
   -> Standardize column names.
   -> Handle missing values.
   -> Extract census year population years.
3. Calculate population growth rates blw census years.
4. Integrate Geospatial data
   -> load regional boundary shapefile.
   -> merge population data with spatial boundaries.
5. Visualization
   -> Generate chropleth maps to represent population density.
   -> Create point-based visualization for location specific values.
   -> Develop interactive geospatial maps for dynamic exploration.
6. Analysis
   -> Study spatial / variations
   -> Identify regions of high and low growth
   -> observe clustering and distribution patterns.

Program :-

```
import pandas as pd
import geopandas as gpd
import matplotlib.pyplot as plt.
import folium.
df = pd.read_csv("population_data.csv")
df.columns = ["Region", "Status", "Zone", "1991", "2001", "2011"]
df = df.dropna()
```

```
df["Growth_1991_2011"] = ((df["2011"] - df["1991"]) / df["1991"]) * 100
map_data = gpd.read_file("region_boundaries.shp")
region_stats = df.groupby("zone").sum().reset_index()
merged = map_data.merge(regional_stats, on="zone")
fig, ax = plt.subplots(figsize=(10,8))
merged.plot(column="2011", cmap="OrRd", linewidth=0.8, ax=ax,
edgecolor="0.8", legend=True)
plt.title("Spatial distribution of population (2011 census)")
plt.show()
m = folium.Map(location=[11.0, 78.0], zoom_start=6)
for _, row in df.iterrows():
    folium.CircleMarker(location=[row["latitude"],
      row["longitude"]],
        radius=row["2011"]/100000,
      popup=f"population: {row['2011']}",
        color="blue",
        fill=True,
        fill_opacity=0.6).add_to(m)
m.save("population_spatial_map.html")
```

Output:

Chennai
District: Chennai
Population 1991: 4010000
Population 2011: 4640000
Growth 15.7%.
Madopi
District: madopai
Population 1991: 1200000
Population 2011: 1370000
Growth: 14.17%.

O/P

VEL TECH	
EX No.	8
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	5
TOTAL (20)	20
SIGN WITH DATE	15/9/25

15/9/25

Result: Thus to analyze the population statistics of a program using spatial and geospatial techniques and to visualize them maps is done and executed successfully.

Task 9:- To analyze and visualize Time oriented Data 29/9/25

Analysis to identify systemic patterns in the data that help of form trends, cycles or seasonal variances to forecast the data, - line Graph, Trend Lines, Area chart

Tools: Tableau, Language : Python

9a) The dataset contains various features such as temperature, humidity, rain, precipitation, etc.

Aim:- To analyze and visualize Time oriented Data using Linegraph, trend lines, area chart.

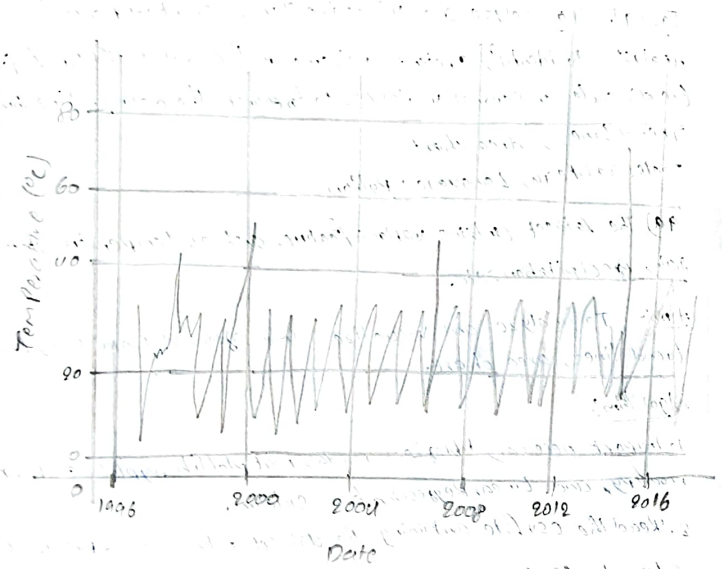
Algorithm:-

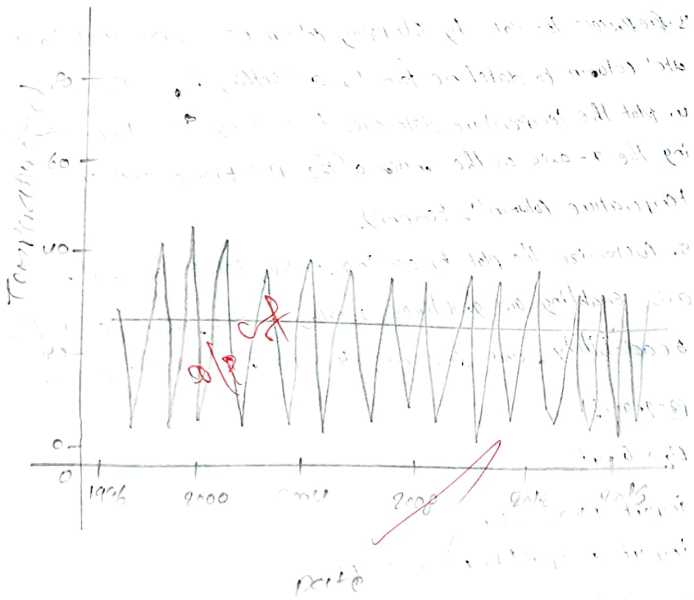
1. Import necessary libraries: Pandas, matplotlib.Pyplot, seaborn, numpy, and LinearRegression from sklearn.
2. Read the csv file containing the dataset into a Data-Frame using Pd.read_csv().
3. Preprocess the data by stripping column names, converting 'datetime-utc' column to datetime format, and setting it as the index.
4. plot the temperature data over time using sns.lineplot(), specifying the x-axis as the index of the DataFrame and y-axis as the temperature column('_tempm').
5. Customize the plot by adding a title, labels for the x and y axis, enabling and grid lines, rotating x-axis labels for better readability, and displaying the plot using plt.show().

Program:-

Line Graph

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.linear_model import LinearRegression
data = pd.read_csv('/content/best set.csv')
```

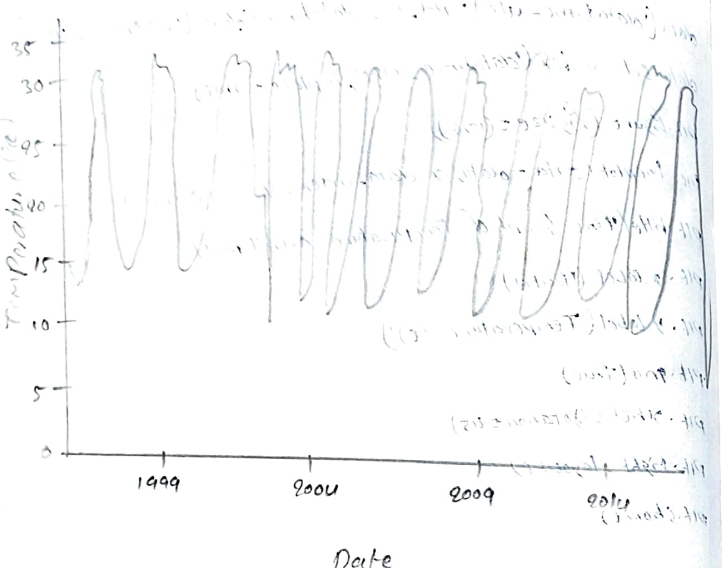




```
data.columns = data.columns.str.strip()
data['datetime_utc'] = pd.to_datetime(data['datetime_utc'])
data.set_index('datetime_utc', inplace=True)
plt.figure(figsize=(10,6))
sns.lineplot(data=data, x=data.index, y='_tempm')
plt.title('Line Graph of Temperature over Time')
plt.xlabel('Date')
plt.ylabel('Temperature (°C)')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

Trend lines:-

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
data = pd.read_csv('/content/testset.csv')
data.columns = data.columns.str.strip()
data['datetime_utc'] = pd.to_datetime(data['datetime_utc'])
data.set_index('datetime_utc', inplace=True)
data['_tempm'].fillna(data['_tempm'].mean(), inplace=True)
x = data.index.astype(int).values.reshape(-1,1)
y = data['_tempm'].values
model = LinearRegression()
model.fit(x,y)
slop = model.coef_[0]
intercept = model.intercept_
```




```
plt.figure(figsize=(14,6))
plt.scatter(data.index, data['_temp_1'], color='blue', label='Temperature data')
plt.plot(data.index, slope*x + intercept, color='red', linestyle='--', linewidth=2, label='Trend Line')
plt.title('Temperature Trend with Trend Line')
plt.xlabel('Date')
plt.ylabel('Temperature (°C)')
plt.grid(True)
plt.legend()
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

Area chart:-

```
import pandas as pd
import matplotlib.pyplot as plt
data = pd.read_csv('/content/testset.csv')
data.columns = data.columns.str.strip()
data['datetime_utc'] = pd.to_datetime(data['datetime_utc'])
data.set_index('datetime_utc', inplace=True)
monthly_mean_temp = data['_temp_1'].resample('M').mean()
plt.figure(figsize=(10,6))
monthly_mean_temp.plot(kind='area', color='sky blue', alpha=0.7)
plt.title('Monthly Mean Temperature')
plt.xlabel('Date')
plt.ylabel('Temperature (°C)')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

VEL TECH	
No.	9
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	4
RECORD (5)	5
TOTAL (20)	19
SIGN WITH DATE	

29/9

Result:- Thus, we successfully implemented the line graph, Trend lines, Area chart by Time Oriented Data and output is verified.

USE CASE

Batchmates

Y. Sasidhar Reddy	VTU25663
P. Rang Susmitha	VTU25834
Y. Rushikesh Reddy	VTU25844
P. Sravan Kumar Reddy	VTU25879
C. Naga Hemanth	VTU25928
K. Rohith	VTU26001
P. Sushma	VTU26063
P. Manjunath Reddy	VTU26193
V. Jaya Sree	VTU26140
~~K~~ Sangay	

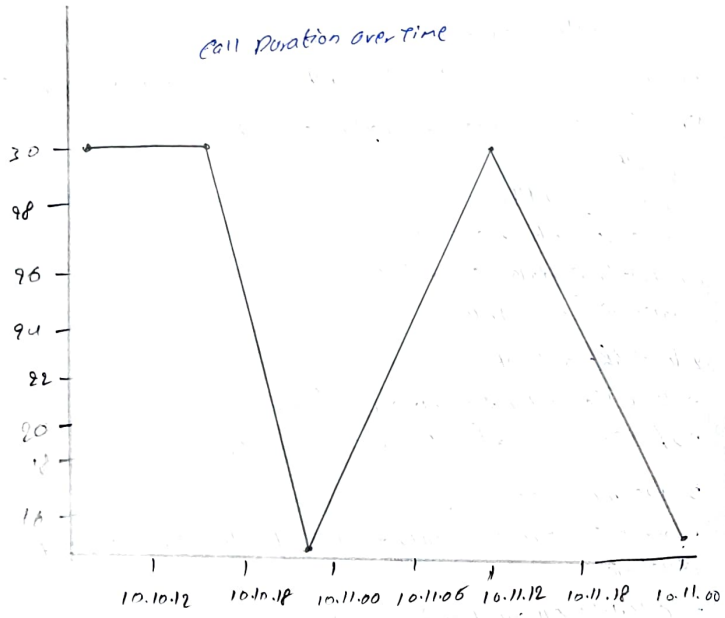
Title: CALL-TIME-ANALYSIS

13/10/25

Aim:- To analyze call duration and patterns using data visualization techniques and gain insights into call behaviour overtime, by using Python libraries such as Pandas, matplotlib and seaborn.

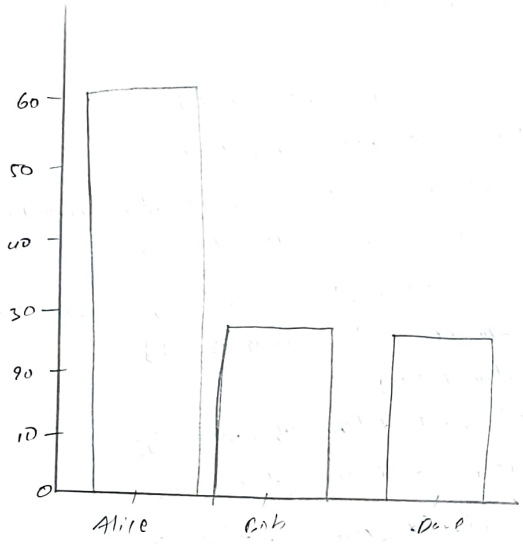
Algorithm:

1. Import necessary libraries
   -> Pandas for data manipulation
   -> matplotlib for data manipulation
   -> datetime for time processing
2. Load the dataset:
   -> Read the call log data into a DataFrame
   -> If Reading from a file, use pd.read_csv(file name.csv)
3. Preprocess the data:
   -> Convert call start and call End columns to datetime format using pd.to_datetime
   -> Calculate call duration in minutes using:
      Duration = (callEnd - callstart).total_seconds()/60
   -> Extract useful time-based features.
      • Hour of the call
      • Date of the call
4. Visualize the data:
   -> line chart: Plot call duration overtime to observe trends.
   -> Heatmaps: Show no. of calls per hour and per day.
   -> Bar chart: Display total call durations per caller
5. Display all plots using plt.show()



Program 1

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime
data = {
    'caller': ['Alice', 'Bob', 'Alice', 'Dave', 'Alice'],
    'receiver': ['Bob', 'Alice', 'Dave', 'Bob', 'Charlie'],
    'callstart': ['2023-10-10 09:00', '2023-10-10 10:15', '2023-10-11 10:30',
                  '2023-10-11 16:00', '2025-10-12 13:00'],
    'callEnd': ['2023-10-10 09:30', '2023-10-10 10:45', '2023-10-11 16:30',
                '2023-10-12 13:15']}
df = pd.DataFrame(data)
df['callstart'] = pd.to_datetime(df['callstart'])
df['callEnd'] = pd.to_datetime(df['callEnd'])
df['Duration(min)'] = (df['callEnd'] - df['callstart']).
                      dt.total_seconds()/60
df['Hour'] = df['callstart'].dt.hour
df['Day'] = df['callstart'].dt.date
plt.figure(figsize=(10,6))
sns.lineplot(data=df, x='callstart', y='Duration(min)',
             marker='o')
plt.title('Call Duration over Time')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
pivot_table = df.pivot_table(index='Hour', columns='Day', values=
              'Duration(min)', aggfunc='count')
```




```
plt.figure(figsize=(8,6))
sns.heatmap(pivot_table, annot=True, cmap='YlGnBu', fmt="d")
plt.title('Incoming calls by Hours and Day')
plt.show()
plt.figure(figsize=(8,6))
caller_duration = df.groupby('Caller')['Duration (min)'].sum().sort_
                  values(ascending=False)
caller_duration.plot(kind='bar', color='orange')
plt.title('Total call duration Per Caller')
plt.ylabel('Total Duration (min)')
plt.xticks(rotation=45)
plt.show()
```

VEL TECH	
EX.No	10
PERFORMANCE (5)	5
RE...	5
VIVA VOCE (5)	3
RECORD (5)	5
TOTAL (25)	18
SIGN WITH DATE	

13/10

Result:- Thus, the Program for call time analysis is completed successfully.

Title: Health care analyses for Patient care and reducing costs.

27/10/25

Aim: To analyze healthcare data to identify patterns and trends that improve patient care and reduce healthcare costs using data visualization and Predictive analytics.

Algorithm:

1. Data collection
-> Gather healthcare-related data

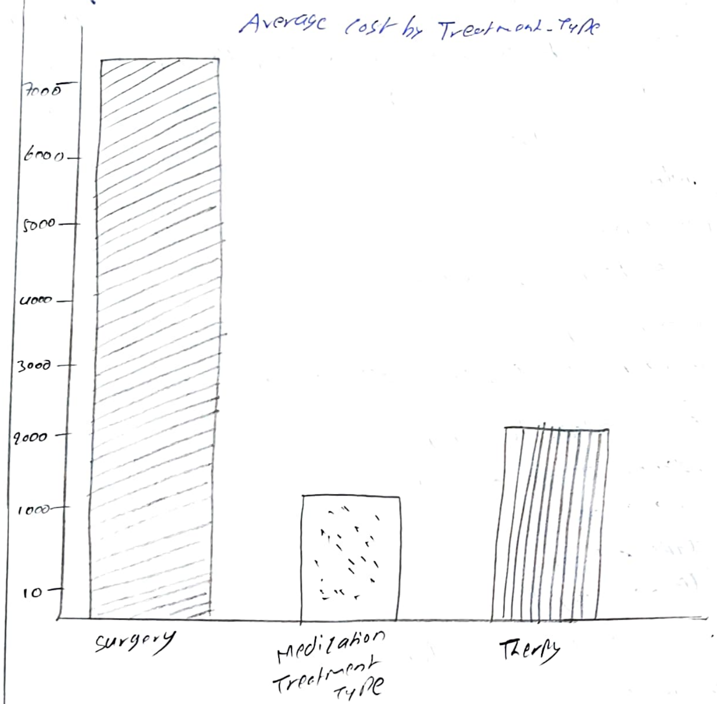
2. Data cleaning
-> Handle missing values, convert data types, remove outliers.

3. Feature selection/Engineering
->Identify relevant features like:
Age & gender
Type of treatment
length of stay

4. Exploratory Data Analysis (EDA)
Visualize:
cost distribution
Relationship b/w treatment and outcomes.

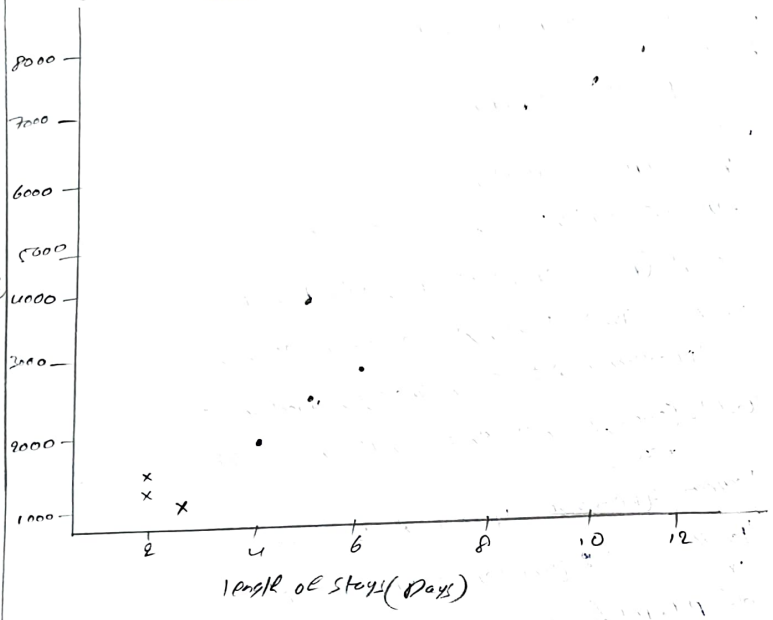
5. Model Building
-> use regression/classification to predict cost.

6. Insights and recommendations.
-> Identify costly treatments with poor outcomes.
-> Suggest preventive measures or optimized care paths.



Program

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style='whitegrid')
data={
 'Patient ID': range (1,11),
 'Age':(25, 40, 60, 35, 70, 50, 45, 35, 65, 35),
'Gender':['F','M','F','M','F','M','F','F','M','M'],
'Treatment_Type' = 'Surgery','Medication','Surgery','Therapy',
     'Medication','Surgery','Therapy','Medication','Therapy','surgery'],
'Cost':[5000, 1200, 7000, 2000, 1500, 5000, 2200, 1400, 2500,
     7500],
'outcome':['Recovered','Recovered','Recovered','Recovered','Recovered'],
'length_of_stay': [5,2,10,4,3,12,5,2,6,11]
}
df=pd.DataFrame(data)
plt.figure(figsize=(8,5))
sns.barplot (x='Treatment_type', y='cost', data=df, ci=None,
        palette='Pastel')
plt.title('Average cost by Treatment Type')
plt.ylabel (Cost ($))
plt.xlabel ('Treatment Type')
plt.tight_layout()
plt.show()
plt.figure(figsize (8,5))
sns.scatterplot (x='length_of_stay', y='cost', hue='outcome',
     style='Treatment_type', data=df, s=100)
```



outcome:
o Recovered
• not-Recovered
| surgery
X medication
□ Therapy

```
plt.title('cost vs length of stay')
plt.xlabel('length of stay (days)')
plt.ylabel('cost($)')
plt.legend(bbox_to_anchor=(1.05,1)  loc='upper left')
plt.tight_layout()
plt.show()
```

Result: Thus the Program for Healthcare analysis for patient care and reducing costs was completed successfully.