FRAMEWORK FOR DATA AND VISUAL ANALYTICS
EXPERIMENT : 1-8
ROLL.NO : 231501186
DEPT : AIML

EXP-1 Setting up the Python environment and libraries-Juypter Notebook

PROGRAM:

```
print("Hello, Google Colab!")
```

```
**Bold Text** and *Italic Text*

- Bullet 1
- Bullet 2

`Inline code`

[Google](https://www.google.com)
```

```
import ipywidgets as widgets
from IPython.display import display

# Slider example
slider = widgets.IntSlider(value=5, min=0, max=10, step=1,
description='Slider:')
display(slider)

# Textbox and button
text = widgets.Text(value='Hello', description='Name:')
button = widgets.Button(description='Greet')

def on_button_clicked(b):
    print(f"Hello, {text.value}!")

button.on_click(on_button_clicked)

display(text, button)
```

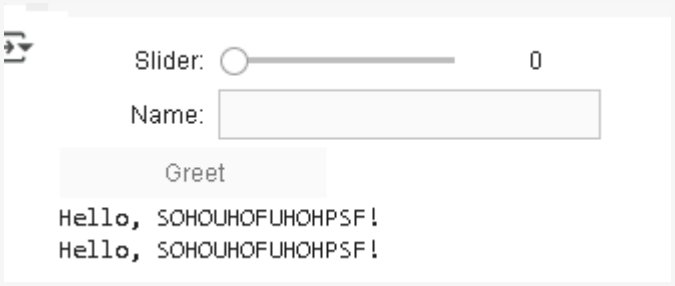
```
OUTPUT:

Hello, Google Colab!
```

Bold Text and *Italic Text*

- Bullet 1
- Bullet 2

Inline code

Google



EXP-2 Data Import and Export

PROGRAM:

```
import pandas as pd

# Replace with your CSV file URL
url =
'https://raw.githubusercontent.com/kwaldenphd/eda-pandas/main/data
/titanic.csv'
df_csv = pd.read_csv(url)

# Display the first few rows
df_csv.head()
```

```
df_excel = pd.read_excel("/content/output.xlsx")  # Replace with
uploaded file name
print("Excel Data:")
print(df_excel.head())

from google.colab import drive
drive.mount('/content/drive')
```

```
# Create sample SQLite database and table (for demo)
engine = create_engine('sqlite://', echo=False)
df_sample = pd.DataFrame({
    "Name": ["Alice", "Bob", "Charlie"],
    "Age": [25, 30, 35]
})
df_sample.to_sql("people", con=engine, index=False)

# Read from the SQL table
df_sql = pd.read_sql("SELECT * FROM people", engine)
print("SQL Data:")
print(df_sql)

# Read HTML table from a webpage
url =
"https://en.wikipedia.org/wiki/List_of_countries_by_GDP_(nominal)"
tables = pd.read_html(url)

# Display the first table
df_web = tables[0]
print("Web Table Data:")
print(df_web.head())

import pandas as pd

# Sample DataFrame
data = {'Name': ['Alice', 'Bob', 'Charlie'],
        'Age': [25, 30, 35],
        'City': ['New York', 'San Francisco', 'Los Angeles']}
df = pd.DataFrame(data)

# Export to Excel
df.to_excel('output1.xlsx', index=False)

OUTPUT:
```

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings,	female	38.0	1	0	PC 17599	71.2833	C85	C

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
			Mrs. John Bradley (Florence Briggs Th...								
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

Excel Data:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

```
3    0       113803  53.1000  C123       S
4    0       373450   8.0500   NaN       S
```

addCode
addText

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

addCode
addText

```
SQL Data:
      Name  Age
0    Alice   25
1      Bob   30
2  Charlie   35


Web Table Data:
                                                    0
0  Largest economies in the world by GDP (nominal...
```

EXP-3 Data Cleaning

PROGRAM:

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler, MinMaxScaler

# Sample dataset creation (you can replace this with your own
dataset)
data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Edward',
'Alice'],
    'Age': [25, np.nan, 30, 22, 35, 25],
    'Salary': [50000, 60000, np.nan, 52000, 58000, 50000],
    'Department': ['HR', 'IT', 'IT', np.nan, 'Finance', 'HR'],
    'JoinDate': ['2010-01-10', '2012-05-15', '2011-08-20',
'2013-07-30', '2010-11-25', '2010-01-10']
}

df = pd.DataFrame(data)
print("Original DataFrame:")
print(df)

print("\nMissing values in each column:")
print(df.isnull().sum())

print("\nMissing values in each column:")
print(df.isnull().sum())
```

```
df.dropna(subset=['Salary'], inplace=True)

df.drop_duplicates(inplace=True)

df.drop(columns=['JoinDate'], inplace=True)

df['Age'] = df['Age'].astype(int)
df['Salary'] = df['Salary'].astype(int)

df['Department'] = df['Department'].astype('category')

scaler = StandardScaler()
df[['Age', 'Salary']] = scaler.fit_transform(df[['Age',
'Salary']])
print("\nAfter Standardization:")
print(df[['Age', 'Salary']])

minmax_scaler = MinMaxScaler()
df[['Age', 'Salary']] = minmax_scaler.fit_transform(df[['Age',
'Salary']])
print("\nAfter Min-Max Scaling:")
print(df[['Age', 'Salary']])
```

OUTPUT:

```
Original DataFrame:
      Name    Age    Salary Department     JoinDate
0    Alice   25.0   50000.0         HR   2010-01-10
1      Bob    NaN   60000.0         IT   2012-05-15
2  Charlie   30.0       NaN         IT   2011-08-20
3    David   22.0   52000.0        NaN   2013-07-30
4   Edward   35.0   58000.0    Finance   2010-11-25
5    Alice   25.0   50000.0         HR   2010-01-10

Missing values in each column:
Name          0
Age           1
Salary        1
Department    1
JoinDate      0
dtype: int64

/tmp/ipython-input-4-2707674413.py:1: FutureWarning: A value is
trying to be set on a copy of a DataFrame or Series through
chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
values always behaves as a copy.
```

```
For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on
the original object.


  df['Age'].fillna(df['Age'].mean(),  inplace=True)
/tmp/ipython-input-4-2707674413.py:2: FutureWarning: A value is
trying to be set on a copy of a DataFrame or Series through
chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on
the original object.


  df['Department'].fillna(df['Department'].mode()[0],
inplace=True)
```

After Standardization:

	Age	Salary
0	-0.467257	-1.212678
1	-0.051917	1.212678
3	-1.090266	-0.727607
4	1.609440	0.727607

After Min-Max Scaling:

	Age	Salary
0	0.230769	0.0
1	0.384615	1.0
3	0.000000	0.2
4	1.000000	0.8

EXP-4 -Data Inspection and Analysis

PROGRAM:

```
import pandas as pd

import numpy as np

from sklearn.datasets import load_iris


# Load the Iris dataset from sklearn

iris = load_iris()

df = pd.DataFrame(data=iris.data, columns=iris.feature_names)

# Add the species column

df['species'] = pd.Categorical.from_codes(iris.target, iris.target_names)

df.head()       # View first 5 rows

df.tail()       # View last 5 rows

df.info()       # Summary: data types, nulls

df.describe()    # Quick stats for numerical columns

df.columns     #columnmn names

df.shape        # Rows and columns count


df[df['species'] == 'setosa']


df[(df['species'] == 'setosa') & (df['sepal length (cm)'] > 5.0)]

df[['sepal length (cm)', 'sepal width (cm)']]
```

```
df['sepal length (cm)'].mean()    # Mean
df['sepal length (cm)'].median()  # Median
df['sepal length (cm)'].mode()    # Mode (returns a Series)
df['sepal length (cm)'].min(), df['sepal length (cm)'].max()  # Range
df['sepal length (cm)'].var()      # Variance
df['sepal length (cm)'].std()      # Standard Deviation
df.corr(numeric_only=True)
```

OUTPUT:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 4 columns):
 #   Column             Non-Null Count  Dtype
---  -------            ----------------  ------
 0   sepal length (cm)  150 non-null    float64
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
dtypes: float64(4)
memory usage: 4.8 KB
(150, 4)
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
5	5.4	3.9	1.7	0.4	setosa
6	4.6	3.4	1.4	0.3	setosa
7	5.0	3.4	1.5	0.2	setosa
8	4.4	2.9	1.4	0.2	setosa
9	4.9	3.1	1.5	0.1	setosa
10	5.4	3.7	1.5	0.2	setosa
11	4.8	3.4	1.6	0.2	setosa
12	4.8	3.0	1.4	0.1	setosa
13	4.3	3.0	1.1	0.1	setosa
14	5.8	4.0	1.2	0.2	setosa

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
15	5.7	4.4	1.5	0.4	setosa
16	5.4	3.9	1.3	0.4	setosa
17	5.1	3.5	1.4	0.3	setosa
18	5.7	3.8	1.7	0.3	setosa
19	5.1	3.8	1.5	0.3	setosa
20	5.4	3.4	1.7	0.2	setosa
21	5.1	3.7	1.5	0.4	setosa
22	4.6	3.6	1.0	0.2	setosa
23	5.1	3.3	1.7	0.5	setosa
24	4.8	3.4	1.9	0.2	setosa
25	5.0	3.0	1.6	0.2	setosa
26	5.0	3.4	1.6	0.4	setosa
27	5.2	3.5	1.5	0.2	setosa
28	5.2	3.4	1.4	0.2	setosa
29	4.7	3.2	1.6	0.2	setosa

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
30	4.8	3.1	1.6	0.2	setosa
31	5.4	3.4	1.5	0.4	setosa
32	5.2	4.1	1.5	0.1	setosa
33	5.5	4.2	1.4	0.2	setosa
34	4.9	3.1	1.5	0.2	setosa
35	5.0	3.2	1.2	0.2	setosa
36	5.5	3.5	1.3	0.2	setosa
37	4.9	3.6	1.4	0.1	setosa
38	4.4	3.0	1.3	0.2	setosa
39	5.1	3.4	1.5	0.2	setosa
40	5.0	3.5	1.3	0.3	setosa
41	4.5	2.3	1.3	0.3	setosa
42	4.4	3.2	1.3	0.2	setosa
43	5.0	3.5	1.6	0.6	setosa
44	5.1	3.8	1.9	0.4	setosa

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
45	4.8	3.0	1.4	0.3	setosa
46	5.1	3.8	1.6	0.2	setosa
47	4.6	3.2	1.4	0.2	setosa
48	5.3	3.7	1.5	0.2	setosa
49	5.0	3.3	1.4	0.2	setosa

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
0	5.1	3.5	1.4	0.2	setosa
5	5.4	3.9	1.7	0.4	setosa
10	5.4	3.7	1.5	0.2	setosa
14	5.8	4.0	1.2	0.2	setosa
15	5.7	4.4	1.5	0.4	setosa
16	5.4	3.9	1.3	0.4	setosa
17	5.1	3.5	1.4	0.3	setosa
18	5.7	3.8	1.7	0.3	setosa
19	5.1	3.8	1.5	0.3	setosa
20	5.4	3.4	1.7	0.2	setosa

21	5.1	3.7	1.5	0.4	setosa
23	5.1	3.3	1.7	0.5	setosa
27	5.2	3.5	1.5	0.2	setosa
28	5.2	3.4	1.4	0.2	setosa
31	5.4	3.4	1.5	0.4	setosa
32	5.2	4.1	1.5	0.1	setosa
33	5.5	4.2	1.4	0.2	setosa
36	5.5	3.5	1.3	0.2	setosa
39	5.1	3.4	1.5	0.2	setosa
44	5.1	3.8	1.9	0.4	setosa
46	5.1	3.8	1.6	0.2	setosa
48	5.3	3.7	1.5	0.2	setosa

	sepal length (cm)	sepal width (cm)
0	5.1	3.5
1	4.9	3.0
2	4.7	3.2
3	4.6	3.1
4	5.0	3.6

…	…	…
145	6.7	3.0

	sepal length (cm)	sepal width (cm)
146	6.3	2.5
147	6.5	3.0
148	6.2	3.4
149	5.9	3.0

150 rows × 2 columns

	sepal length (cm)
0	5.0

dtype: float64

0.8280661279778629

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
sepal length (cm)	1.000000	-0.117570	0.871754	0.817941
sepal width (cm)	-0.117570	1.000000	-0.428440	-0.366126
petal length (cm)	0.871754	-0.428440	1.000000	0.962865
petal width (cm)	0.817941	-0.366126	0.962865	1.000000

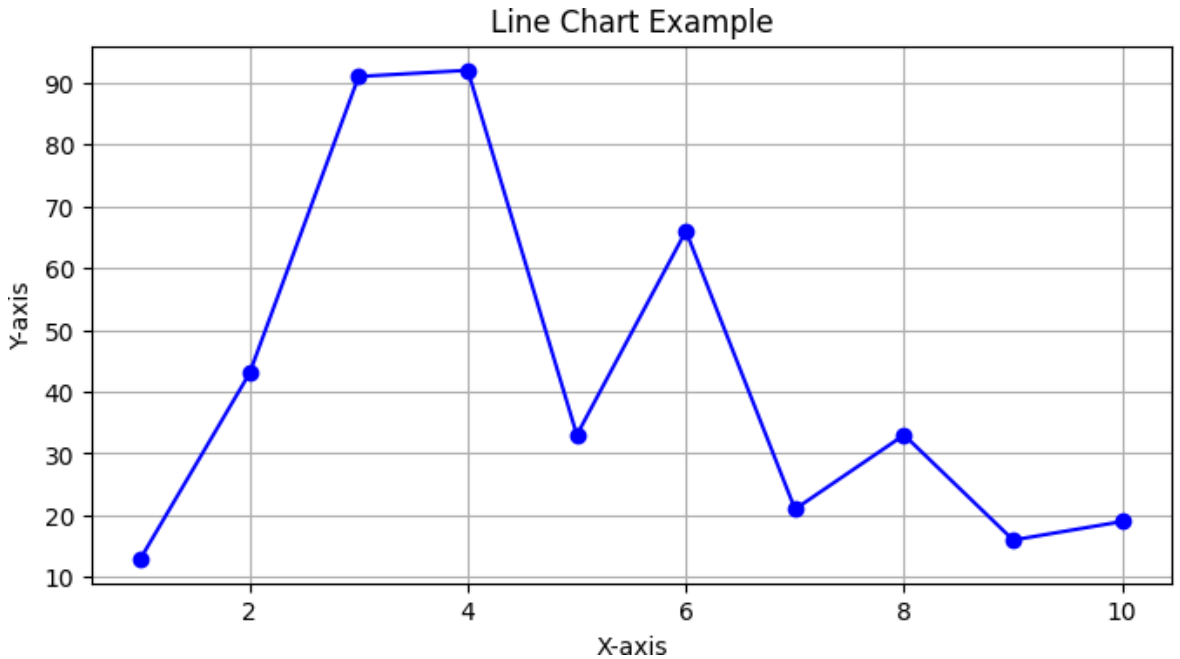
EXP-5 Data Visualization with matplotlib

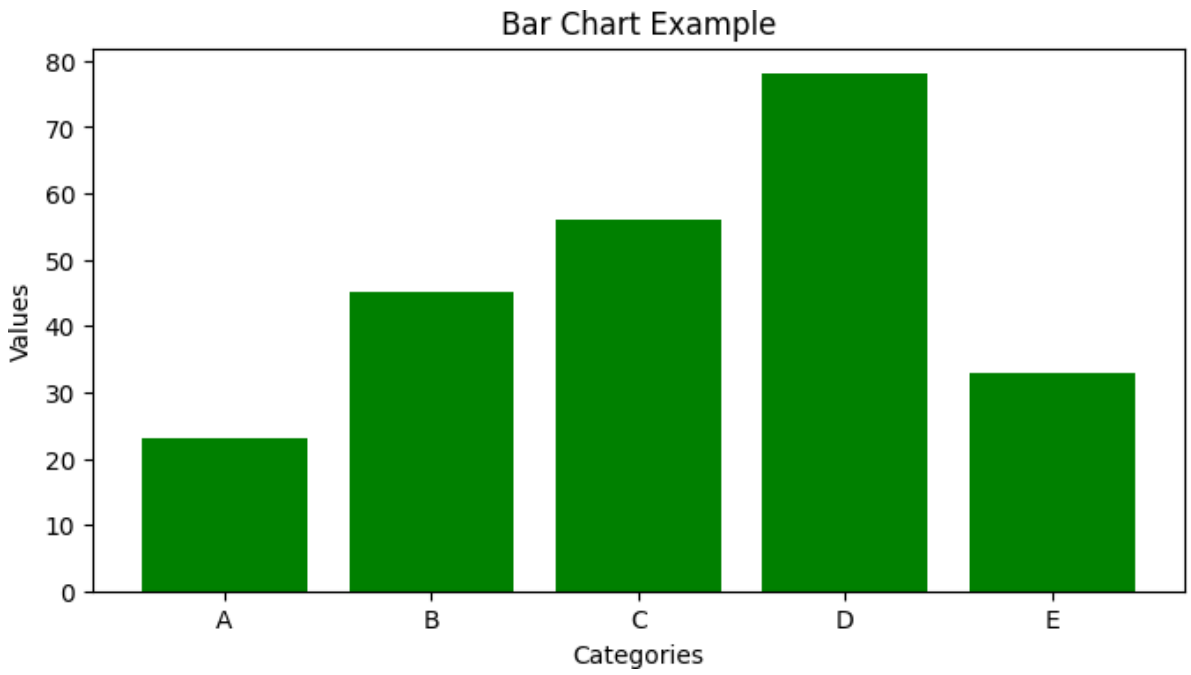
PROGRAM:

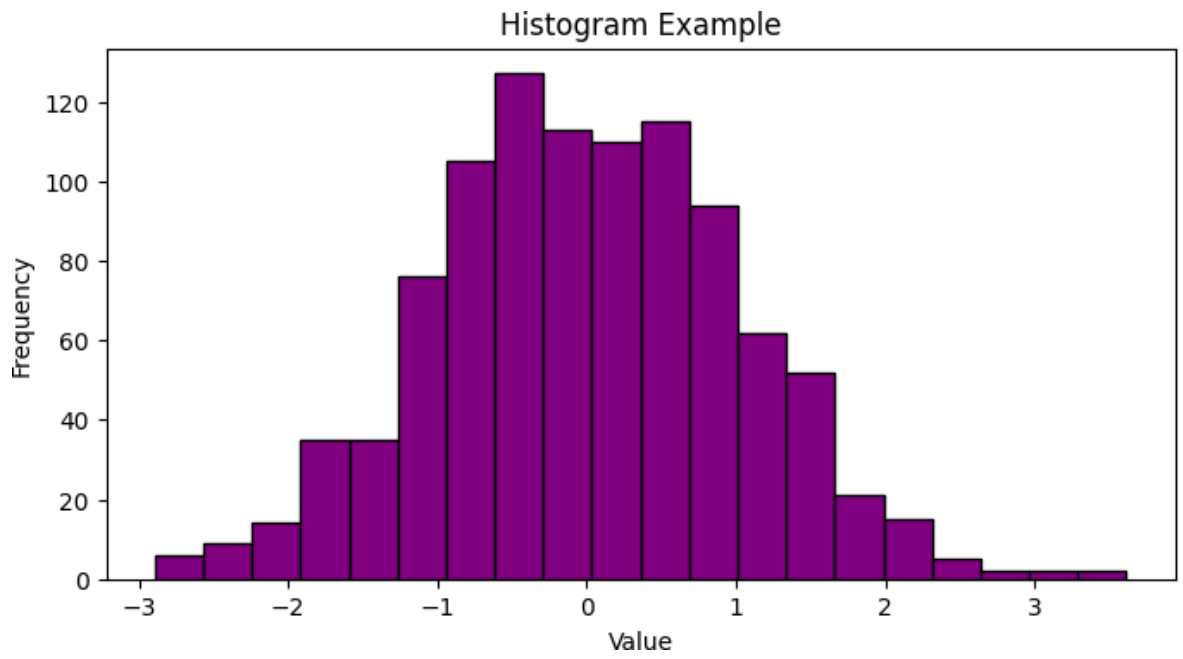
```
# EDA - Data Visualization with Matplotlib


# Install matplotlib if not already (usually preinstalled in Colab)

# !pip install matplotlib


import matplotlib.pyplot as plt

import numpy as np


# Sample data

x = np.arange(1, 11)

y = np.random.randint(10, 100, size=10)

categories = ['A', 'B', 'C', 'D', 'E']

values = [23, 45, 56, 78, 33]

hist_data = np.random.randn(1000)  # Normal distribution

# 1. Line Chart

plt.figure(figsize=(8, 4))

plt.plot(x, y, marker='o', linestyle='-', color='blue')

plt.title('Line Chart Example')

plt.xlabel('X-axis')

plt.ylabel('Y-axis')

plt.grid(True)

plt.show()
```

```
# 2. Bar Chart

plt.figure(figsize=(8, 4))

plt.bar(categories, values, color='green')

plt.title('Bar Chart Example')

plt.xlabel('Categories')

plt.ylabel('Values')

plt.show()


# 3. Histogram

plt.figure(figsize=(8, 4))

plt.hist(hist_data, bins=20, color='purple', edgecolor='black')

plt.title('Histogram Example')

plt.xlabel('Value')

plt.ylabel('Frequency')

plt.show()
```

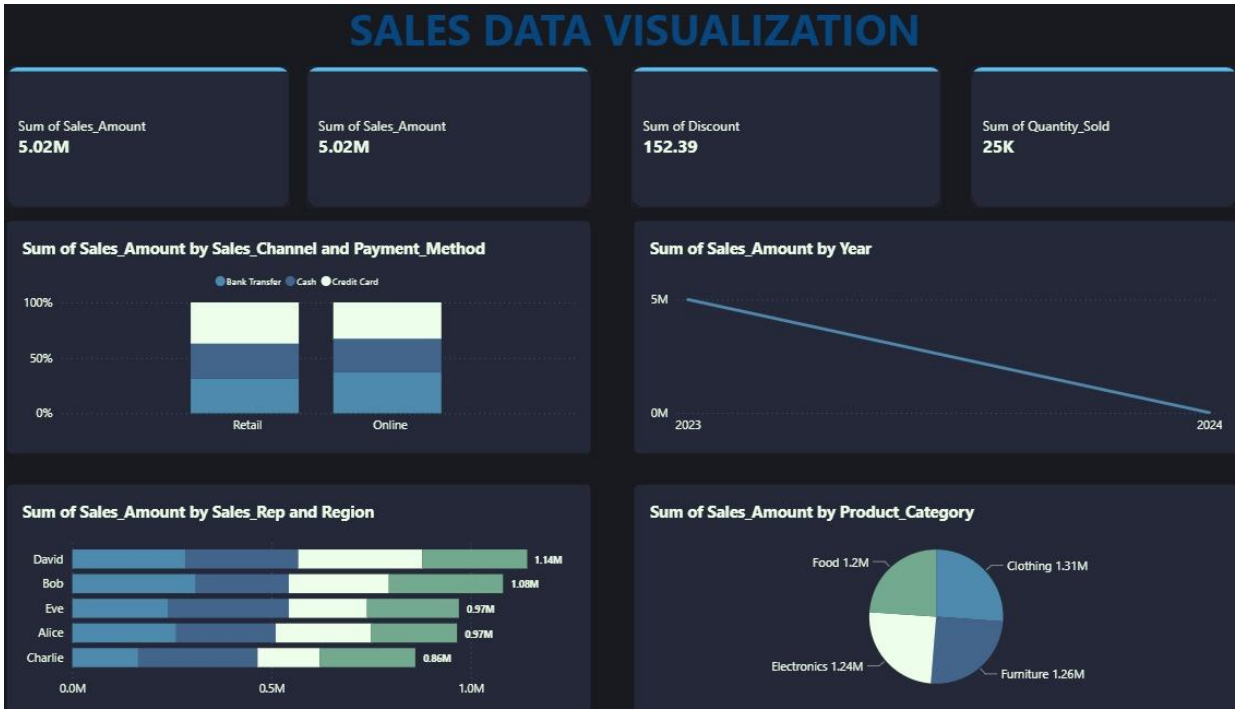
OUTPUT



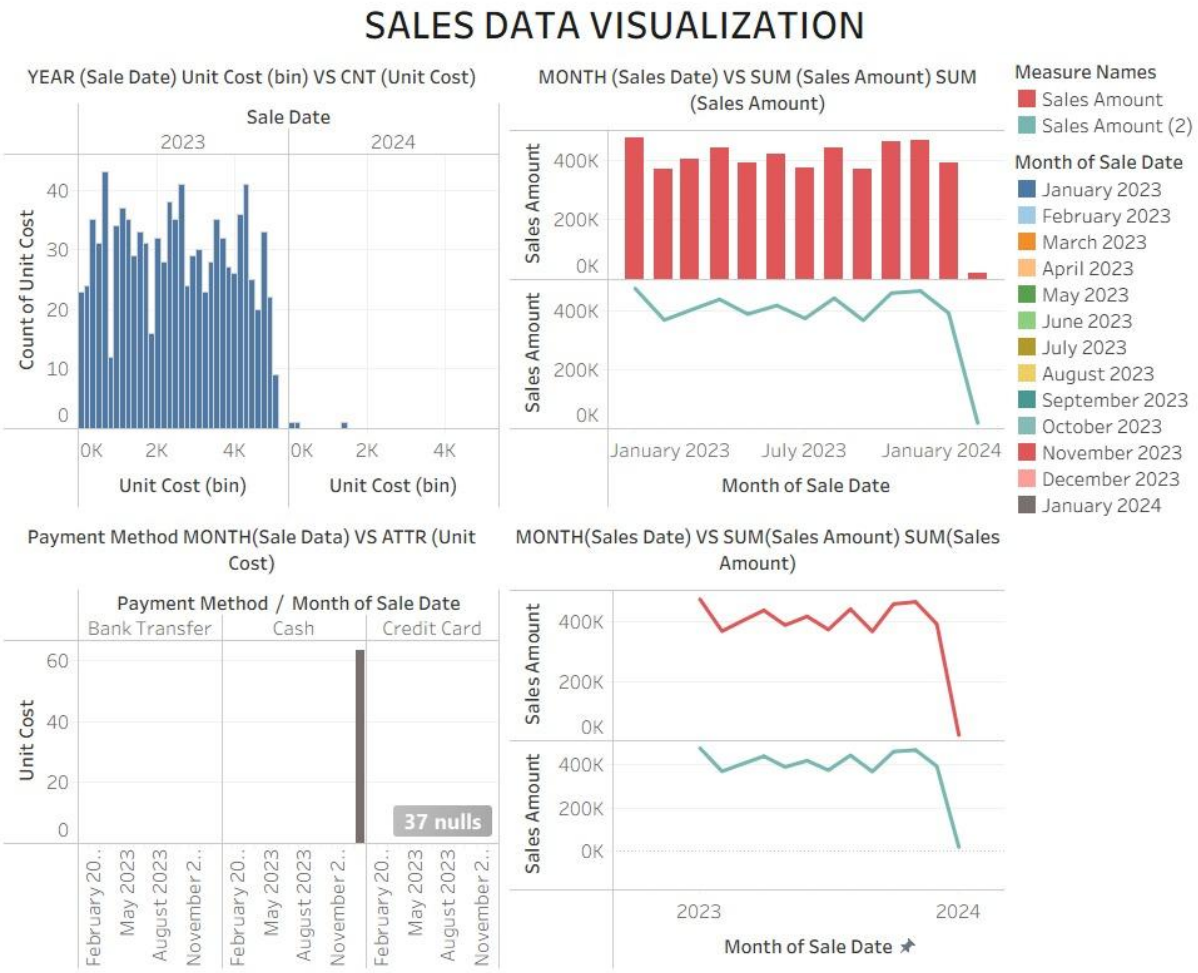

Bar Chart Example
Values
Categories
A
B
C
D
E
0
10
20
30
40
50
60
70
80


Histogram Example
Frequency
Value
0
20
40
60
80
100
120
−3
−2
−1
0
1
2
3

EXP-6 Data visualization Using PowerBi



EXP-7 Data Visualization Using Tableau



EXP-8 MINI PROJECT

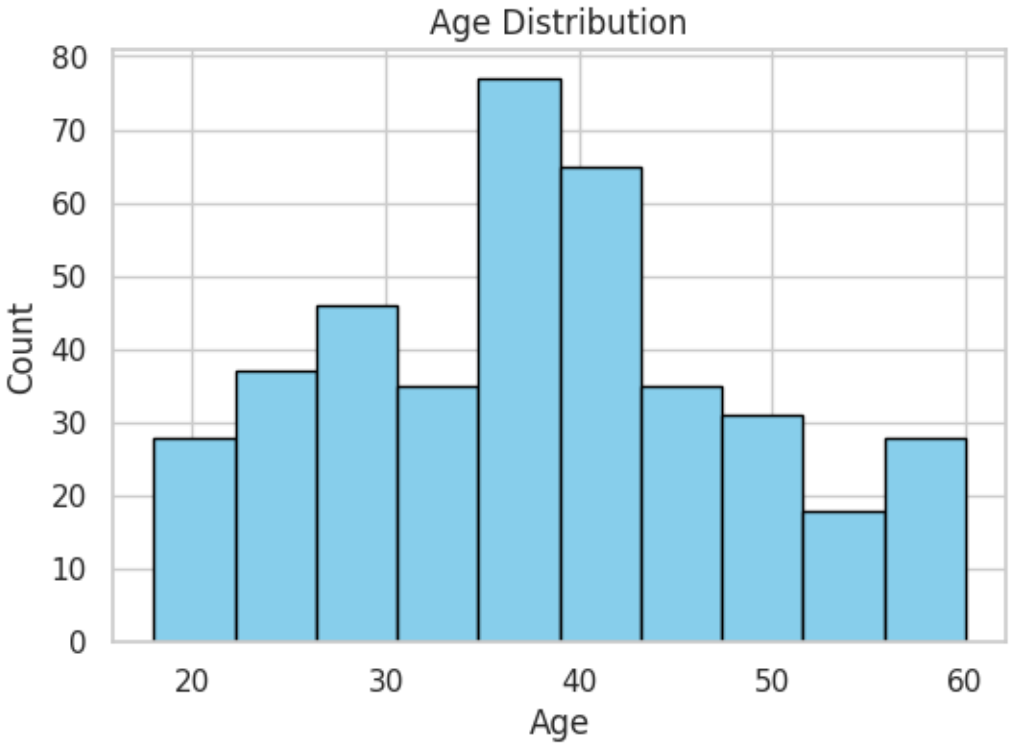
SOCIAL MEDIA PURCHASE ANALYSIS

PYTHON:

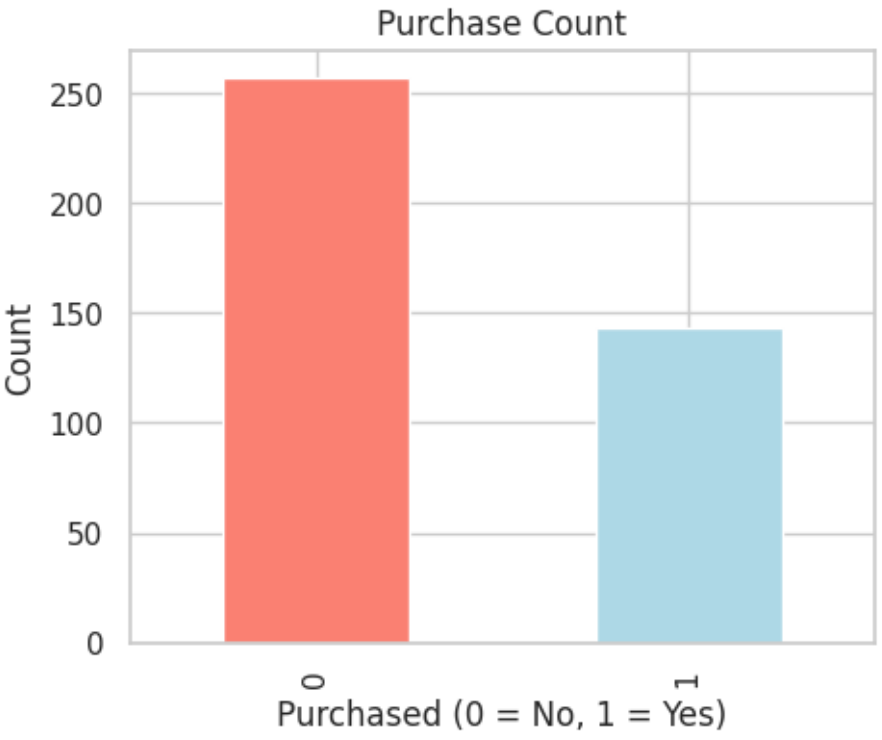
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
sns.set(style="whitegrid")
data = pd.read_csv('/content/drive/MyDrive/Colab
Notebooks/FDVA_CSV/social_ads.csv')
print(data.head())
print(data.info())
plt.figure(figsize=(6,4))
plt.hist(data['Age'], bins=10, color='skyblue', edgecolor='black')
plt.title('Age Distribution')
plt.xlabel('Age')
plt.ylabel('Count')
plt.show()
plt.figure(figsize=(6,4))
plt.hist(data['EstimatedSalary'], bins=10, color='lightgreen',
edgecolor='black')
plt.title('Estimated Salary Distribution')
plt.xlabel('Estimated Salary')
plt.ylabel('Count')
plt.show()
plt.figure(figsize=(5,4))
data['Purchased'].value_counts().plot(kind='bar',
color=['salmon','lightblue'])
plt.title('Purchase Count')
plt.xlabel('Purchased (0 = No, 1 = Yes)')
plt.ylabel('Count')
plt.show()
plt.figure(figsize=(5,5))
data['Purchased'].value_counts().plot.pie(
    autopct='%1.1f%%', startangle=90, colors=['lightgreen',
'lightgray']
)
plt.title('Purchase Proportion')
plt.ylabel('')
plt.show()
plt.figure(figsize=(6,4))
sns.histplot(data=data, x='Age', hue='Purchased', kde=True, bins=10,
palette='coolwarm')
plt.title('Age Distribution by Purchase Decision')
plt.show()
plt.figure(figsize=(6,4))
sns.histplot(data=data, x='EstimatedSalary', hue='Purchased',
kde=True, bins=10, palette='coolwarm')
plt.title('Salary Distribution by Purchase Decision')
plt.show()
plt.figure(figsize=(6,4))
sns.boxplot(x='Purchased', y='Age', data=data, palette='pastel')
```

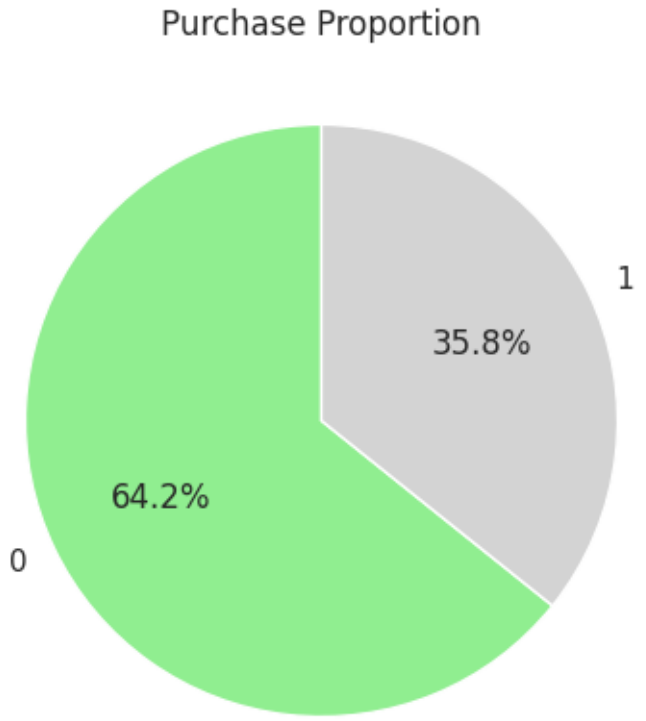
```
plt.title('Age by Purchase Decision')
plt.show()
plt.figure(figsize=(6,4))
sns.boxplot(x='Purchased', y='EstimatedSalary', data=data,
palette='pastel')
plt.title('Estimated Salary by Purchase Decision')
plt.show()
plt.figure(figsize=(6,4))
sns.violinplot(x='Purchased', y='EstimatedSalary', data=data,
palette='muted')
plt.title('Salary Distribution by Purchase Decision (Violin)')
plt.show()
plt.figure(figsize=(6,4))
plt.scatter(
    data['Age'], data['EstimatedSalary'],
    c=data['Purchased'], cmap='coolwarm', alpha=0.6
)
plt.title('Age vs Estimated Salary (Colored by Purchase)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.show()
plt.figure(figsize=(6,4))
sns.regplot(x='Age', y='EstimatedSalary', data=data,
scatter_kws={'alpha':0.6})
plt.title('Age vs Estimated Salary with Regression Line')
plt.show()
plt.figure(figsize=(6,4))
sns.kdeplot(data=data, x='EstimatedSalary', hue='Purchased',
fill=True)
plt.title('Salary Density by Purchase Decision')
plt.show()
plt.figure(figsize=(5,3))
sns.heatmap(data.corr(numeric_only=True), annot=True,
cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
sns.pairplot(data, hue='Purchased', diag_kind='kde', palette='husl')
plt.suptitle('Pairwise Relationships', y=1.02)
plt.show()

means = data.groupby('Purchased')[['Age', 'EstimatedSalary']].mean()
means.plot(kind='bar', figsize=(6,4), colormap='Set2')
plt.title('Average Age and Salary by Purchase Decision')
plt.ylabel('Average Value')
plt.show()
plt.figure(figsize=(6,5))
plt.scatter(
    data['Age'], data['EstimatedSalary'],
    s=data['Age']*2, alpha=0.4, c=data['Purchased'], cmap='viridis'
)
plt.title('Bubble Plot: Age vs Salary (Bubble Size = Age)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.show()
plt.figure(figsize=(5,3))
data['Purchased'].value_counts().plot(kind='barh',
```

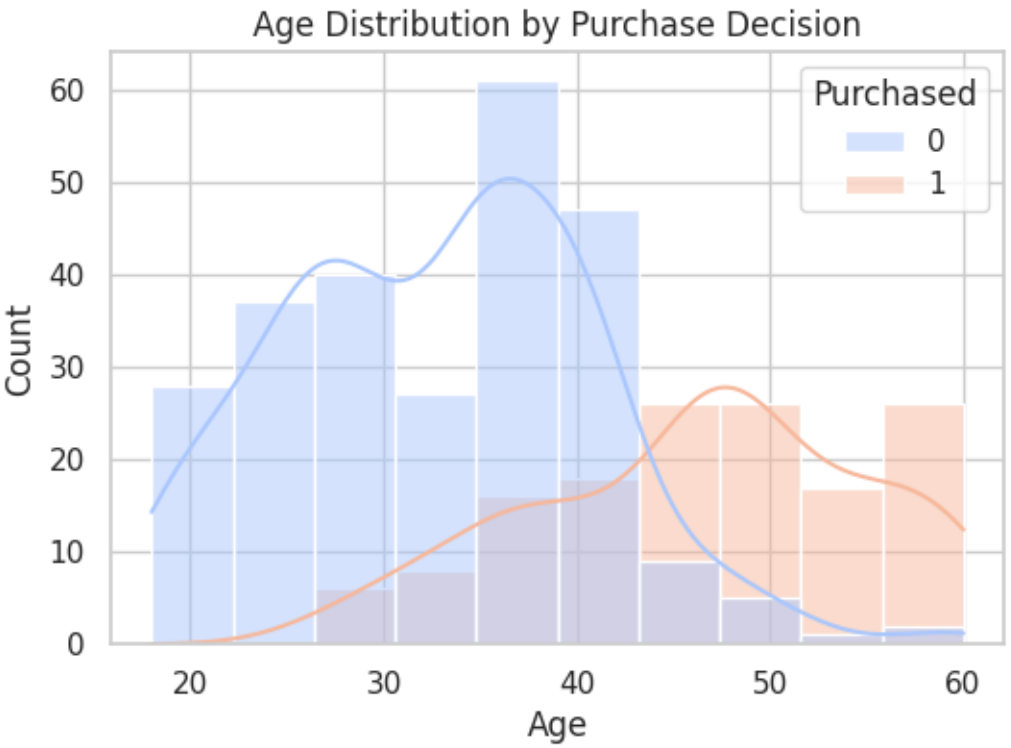
```
color=['lightcoral','skyblue'])
plt.title('Purchased Distribution (Horizontal)')
plt.xlabel('Count')
plt.show()
plt.figure(figsize=(6,4))
sns.kdeplot(data=data, x='Age', hue='Purchased', fill=True)
plt.title('Age Density by Purchase Decision')
plt.show()

print("✅ All visualizations generated successfully!")

OUTPUT:
   Age  EstimatedSalary  Purchased
0   19            19000          0
1   35            20000          0
2   26            43000          0
3   27            57000          0
4   19            76000          0
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Age              400 non-null    int64
 1   EstimatedSalary  400 non-null    int64
 2   Purchased        400 non-null    int64
dtypes: int64(3)
memory usage: 9.5 KB
None
```

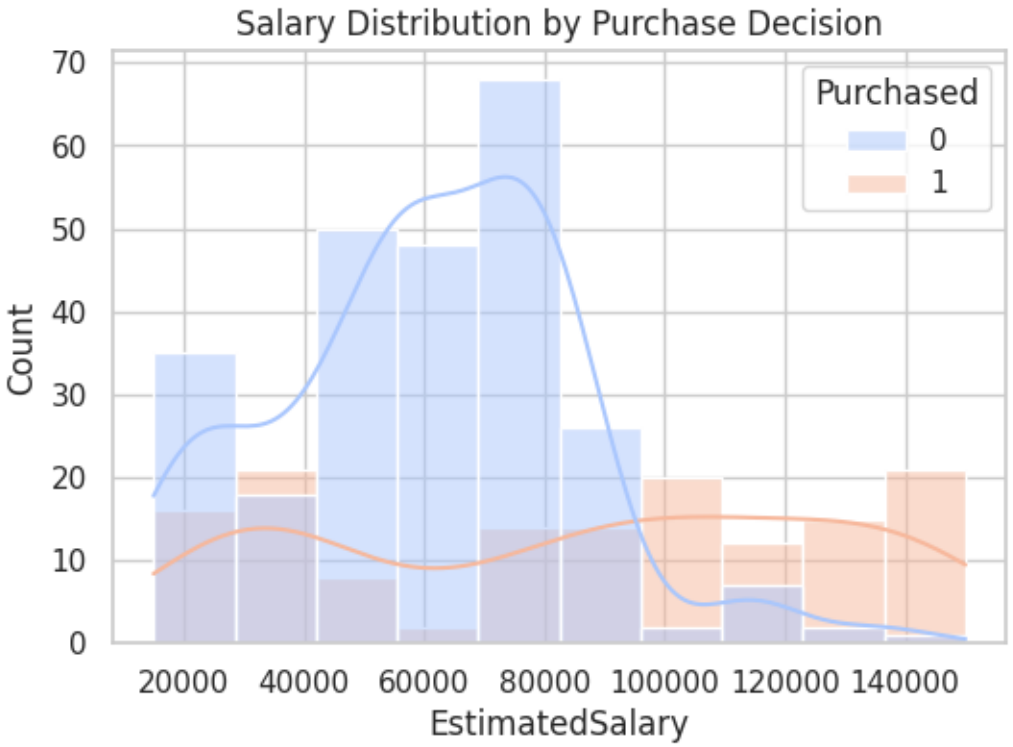



Estimated Salary Distribution
Count
0
10
20
30
40
50
60
70
80
20000
40000
60000
80000
100000
120000
140000
Estimated Salary

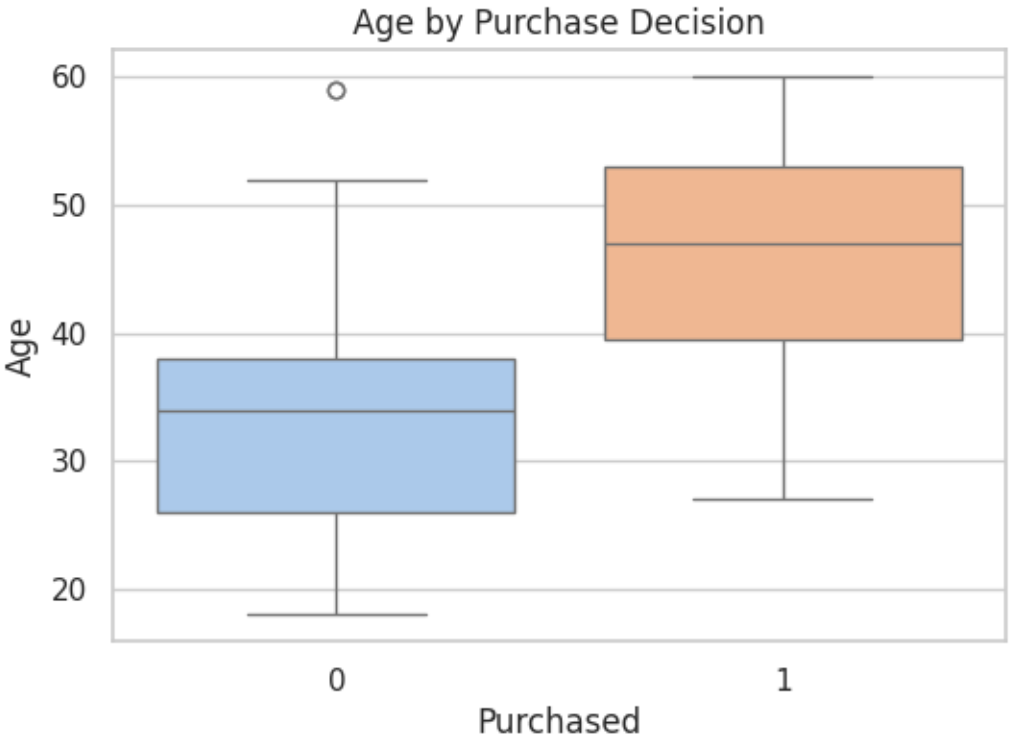

Purchase Count
Count
0
50
100
150
200
250
0
1
Purchased (0 = No, 1 = Yes)


Purchase Proportion
1
35.8%
64.2%
0


Age Distribution by Purchase Decision
Purchased
0
1
Count
60
50
40
30
20
10
0
20
30
40
50
60
Age



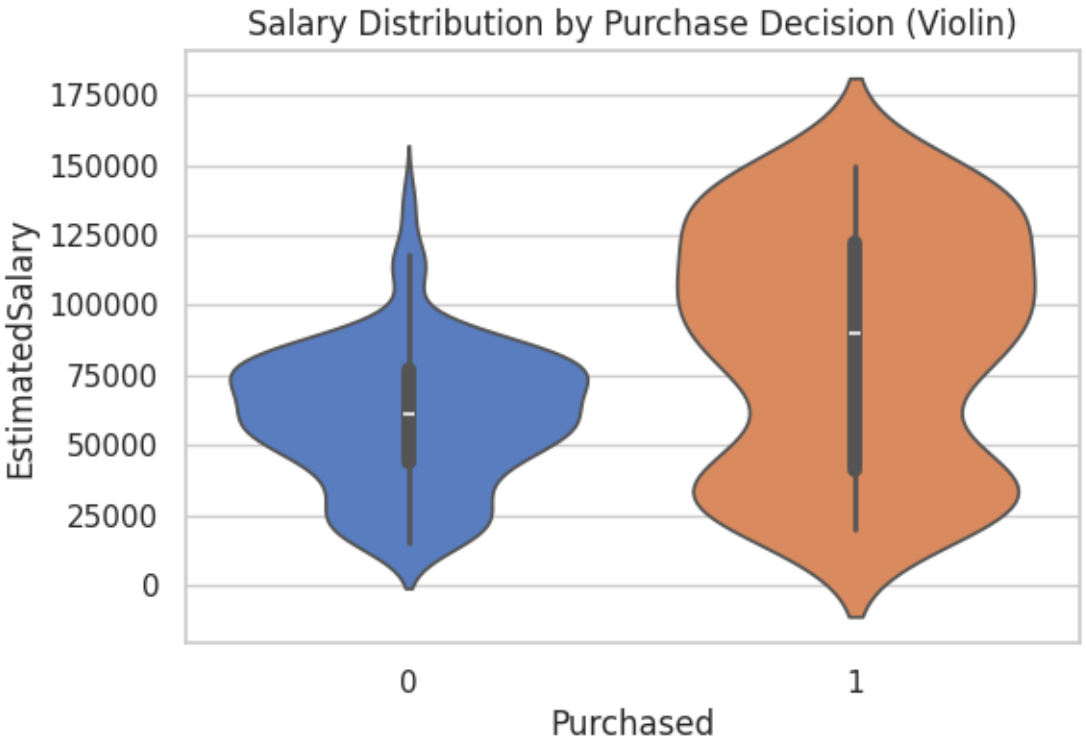

```
/tmp/ipython-input-1506193356.py:78: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.boxplot(x='Purchased', y='Age', data=data, palette='pastel')
```

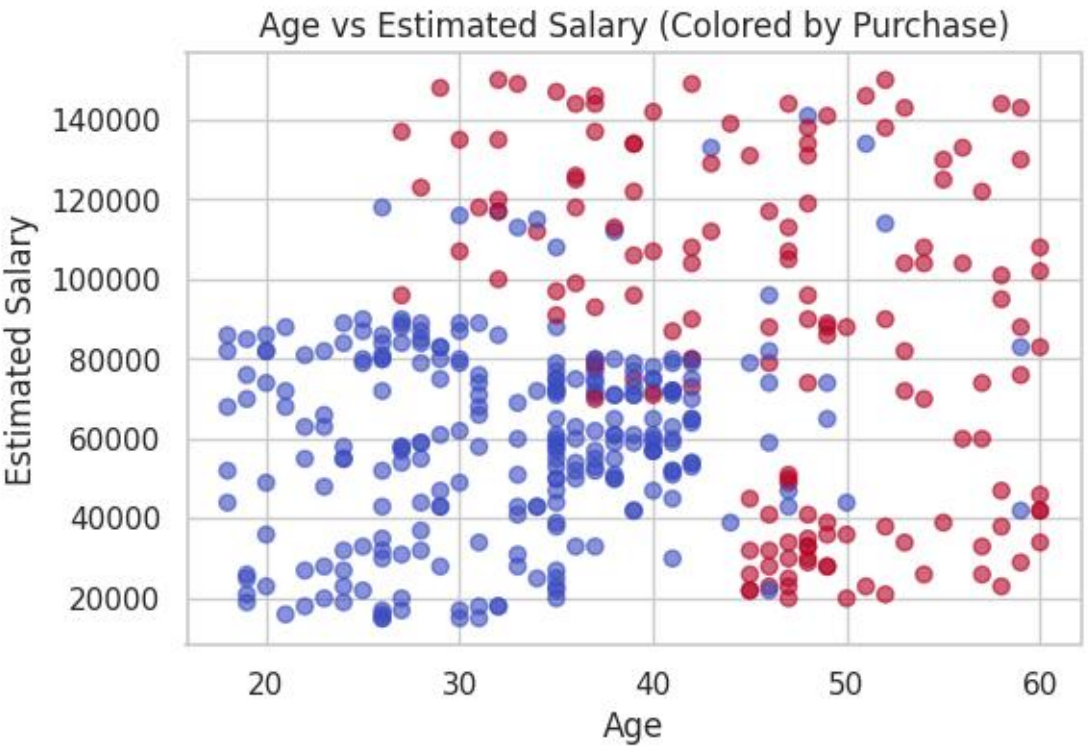



```
/tmp/ipython-input-1506193356.py:84: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
```

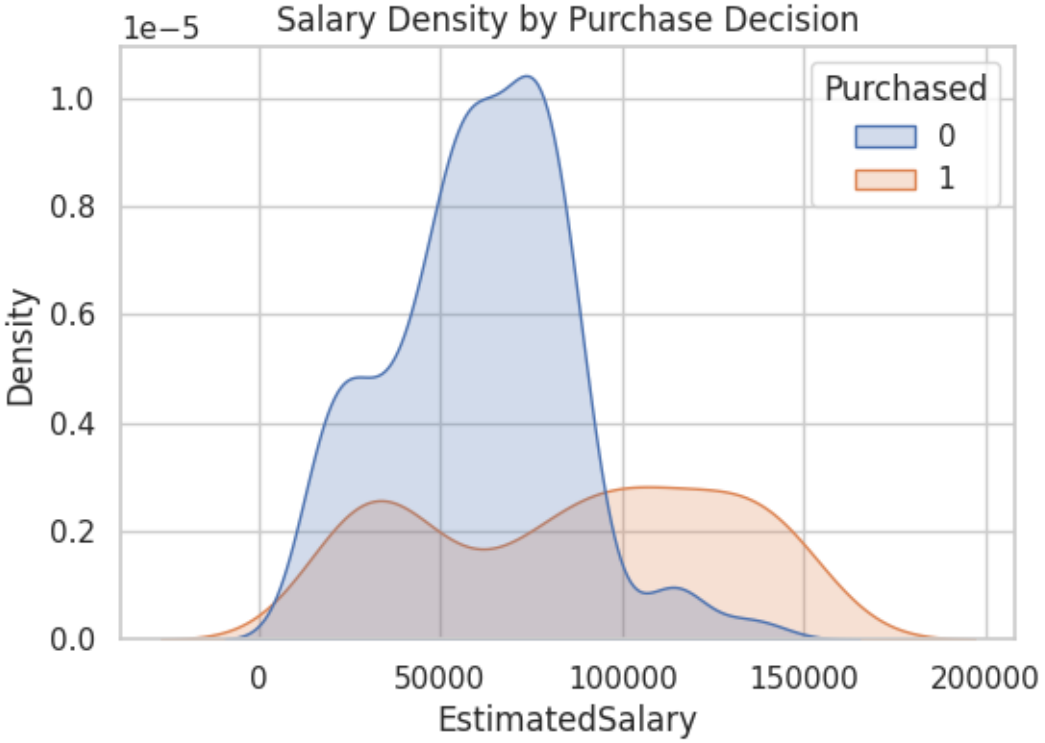
```
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.boxplot(x='Purchased', y='EstimatedSalary', data=data,
palette='pastel')
```

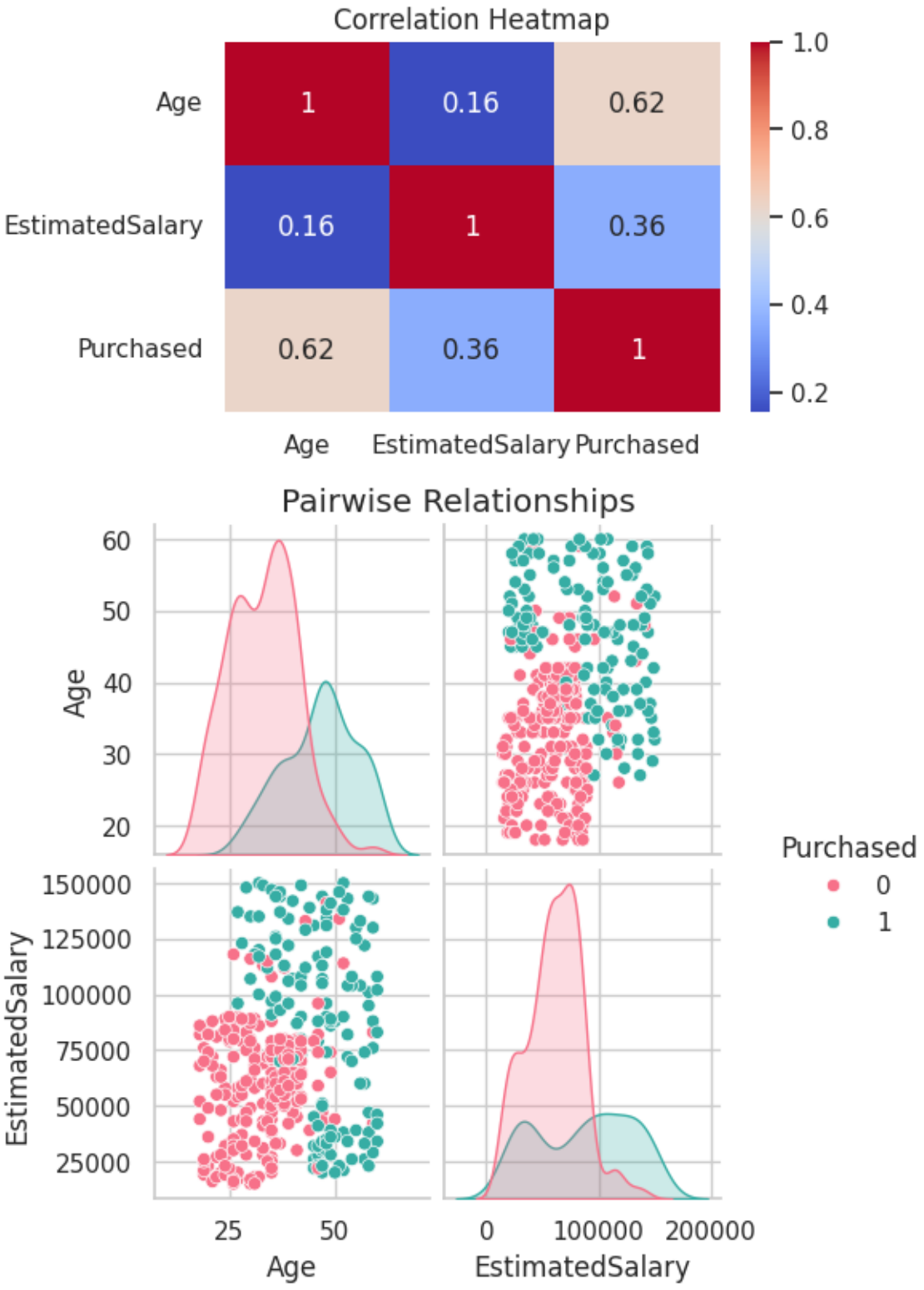

```
/tmp/ipython-input-1506193356.py:90: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.violinplot(x='Purchased', y='EstimatedSalary', data=data,
palette='muted')
```


Salary Distribution by Purchase Decision (Violin)
EstimatedSalary
175000
150000
125000
100000
75000
50000
25000
0
0
1
Purchased

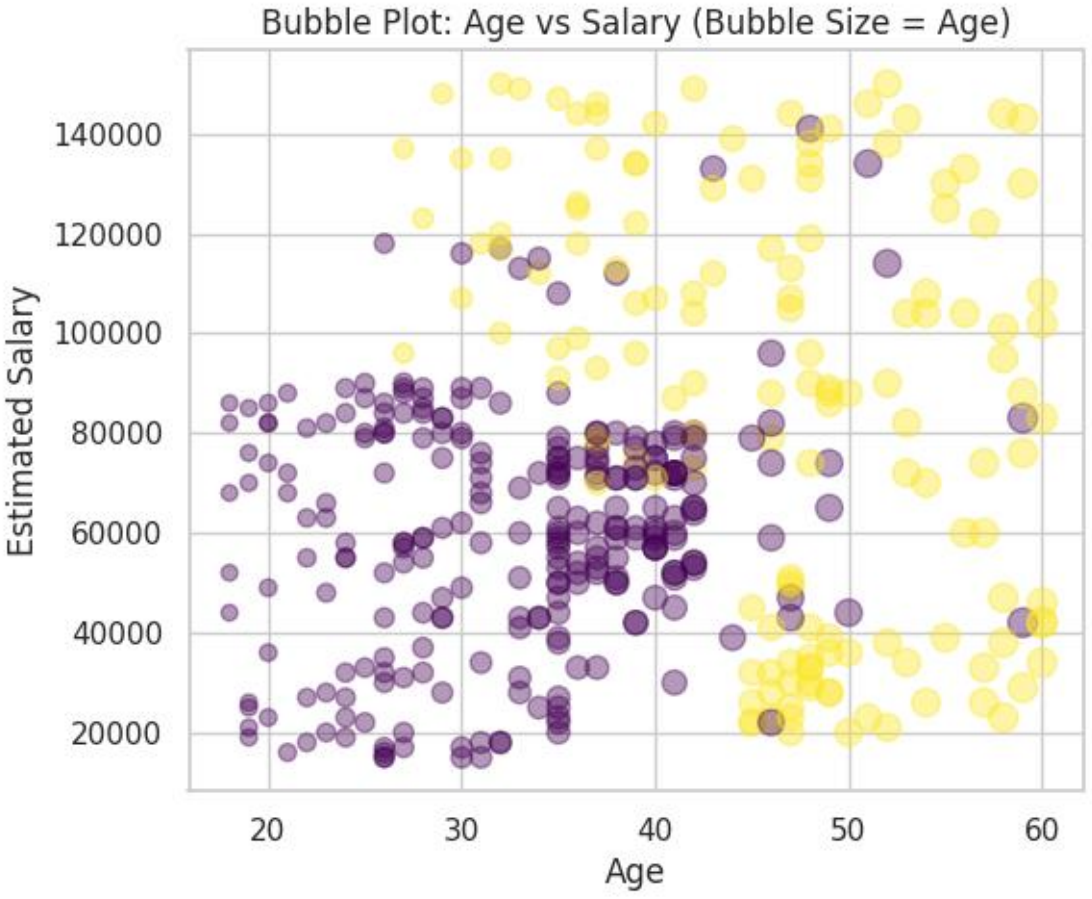

Age vs Estimated Salary (Colored by Purchase)
Estimated Salary
140000
120000
100000
80000
60000
40000
20000
20
30
40
50
60
Age

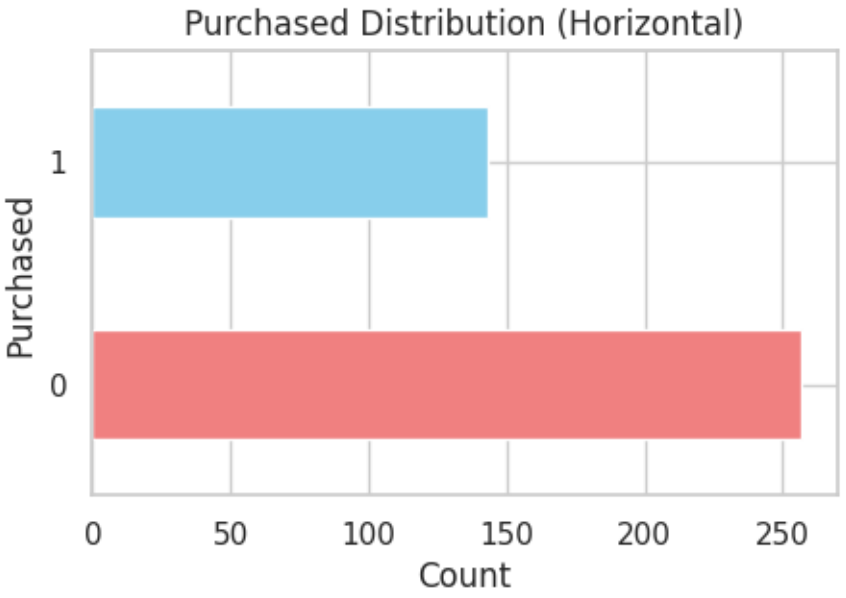

Age vs Estimated Salary with Regression Line
EstimatedSalary
140000
120000
100000
80000
60000
40000
20000
20
30
40
50
60
Age

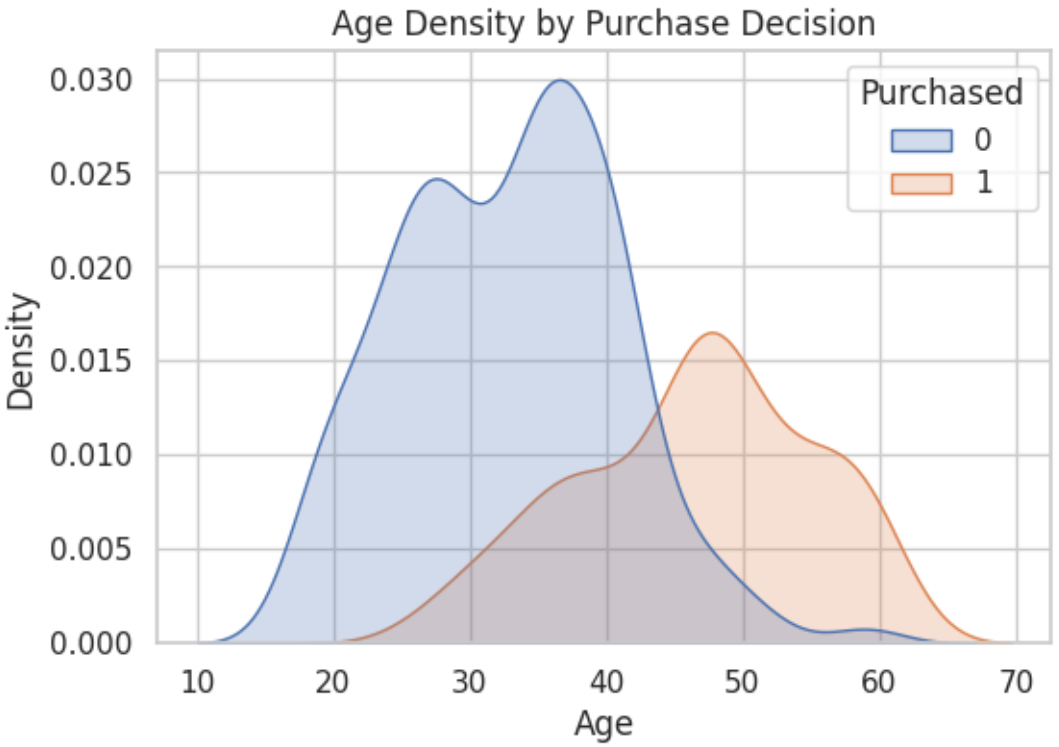

Salary Density by Purchase Decision
1e−5
Purchased
0
1
Density
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000
EstimatedSalary


Correlation Heatmap
Age
EstimatedSalary
Purchased
1
0.16
0.62
0.16
1
0.36
0.62
0.36
1
1.0
0.8
0.6
0.4
0.2
Age
EstimatedSalary
Purchased
Pairwise Relationships
60
50
40
30
20
Age
150000
125000
100000
75000
50000
25000
EstimatedSalary
25
50
Age
0
100000
200000
EstimatedSalary
Purchased
0
1


Average Age and Salary by Purchase Decision
Age
EstimatedSalary
Average Value
80000
60000
40000
20000
0
0
1
Purchased


Bubble Plot: Age vs Salary (Bubble Size = Age)
Estimated Salary
140000
120000
100000
80000
60000
40000
20000
20
30
40
50
60
Age





✅ All visualizations generated successfully!

POWER BI:

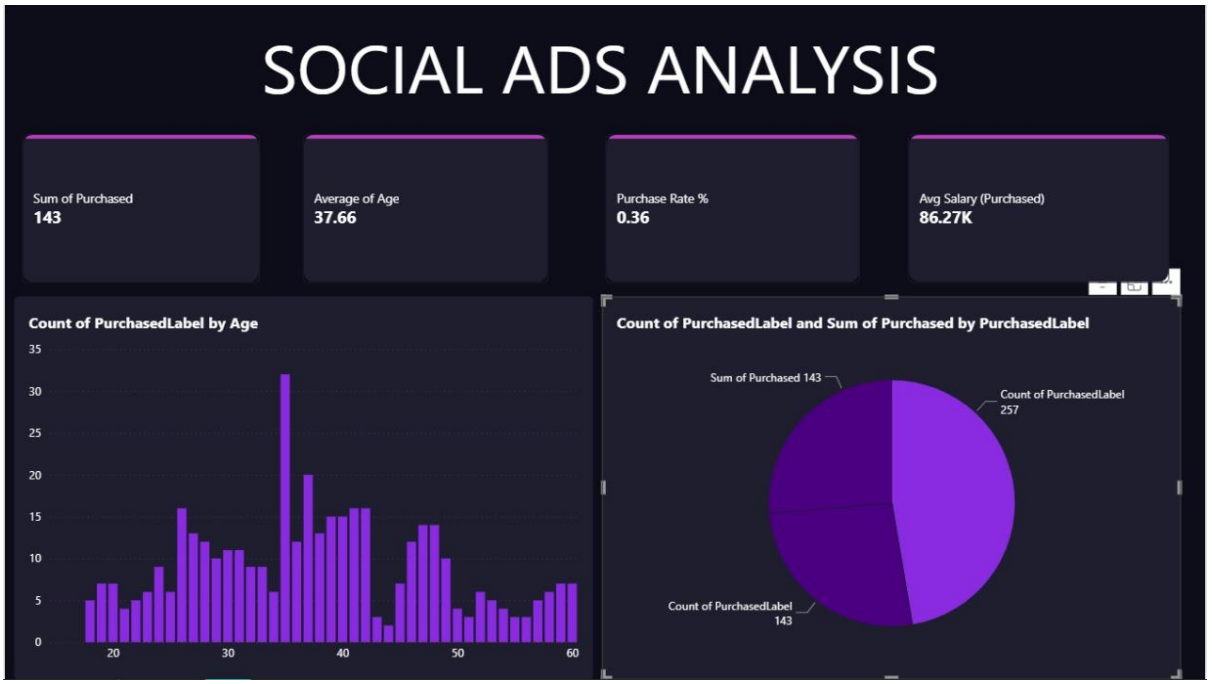
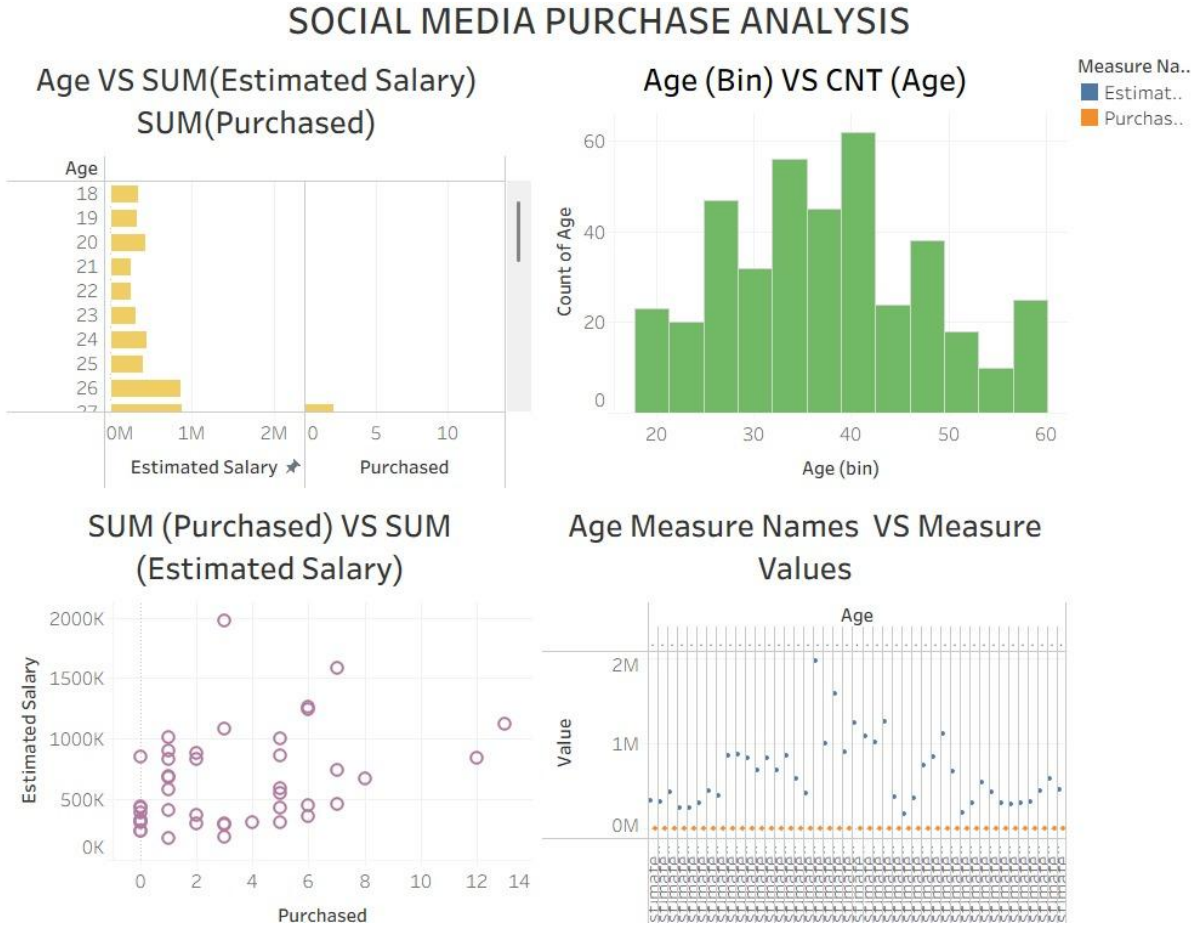

SOCIAL ADS ANALYSIS
Sum of Purchased
143
Average of Age
37.66
Purchase Rate %
0.36
Avg Salary (Purchased)
86.27K
Count of PurchasedLabel by Age
Count of PurchasedLabel and Sum of Purchased by PurchasedLabel
Sum of Purchased 143
Count of PurchasedLabel
257
Count of PurchasedLabel
143

TABLEAU:


SOCIAL MEDIA PURCHASE ANALYSIS
Age VS SUM(Estimated Salary)
SUM(Purchased)
Age
Estimated Salary
Purchased
Age (Bin) VS CNT (Age)
Measure Na..
Estimat..
Purchas..
Count of Age
Age (bin)
SUM (Purchased) VS SUM
(Estimated Salary)
Estimated Salary
Purchased
Age Measure Names VS Measure
Values
Age
Value