Ds lab-4

RAM
Disk

0s

```
import pandas as pd
import numpy as np

# Create a DataFrame with miss
data = {
    'Age': [25, 30, np.nan, 35
    'Salary': [50000, 60000, n
    'Department': ['HR', 'Fina
    'Experience': [2, 5, 3, 8
}

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

# Drop rows with any missing v
df_dropped_rows = df.dropna()
print("\nDropped Rows with Any
print(df_dropped_rows)

# Drop rows where any missing
df_dropped_specific_columns =
print("\nDropped Rows with Mis
print(df_dropped_specific_colu
```

0s

```
# Fill missing values in numer
df_filled_mean = df.copy()
df_filled_mean['Age'].fillna(
df_filled_mean['Salary'].fill
df_filled_mean['Experience'].
print("\nFilled Missing Numer
print(df_filled_mean)

# Fill missing categorical va
df_filled_mode = df.copy()
df_filled_mode['Department'].
print("\nFilled Missing Categ
print(df_filled_mode)

# Forward fill missing values
df_filled_ffill = df.copy()
df_filled_ffill.fillna(method=
print("\nForward Fill Missing
print(df_filled_ffill)

# Backward fill missing value
df_filled_bfill = df.copy()
df_filled_bfill.fillna(method=
print("\nBackward Fill Missing
```

0s

```
3   35.0   70000.0         NaN
4   40.0   80000.0   Marketing

Filled Missing Categorical V
    Age    Salary Department
0   25.0   50000.0         HR
1   30.0   60000.0    Finance
2    NaN       NaN         IT
3   35.0   70000.0    Finance
4   40.0   80000.0   Marketing

Forward Fill Missing Values:
    Age    Salary Department
0   25.0   50000.0         HR
1   30.0   60000.0    Finance
2   30.0   60000.0         IT
3   35.0   70000.0         IT
4   40.0   80000.0   Marketing

Backward Fill Missing Values
    Age    Salary Department
0   25.0   50000.0         HR
1   30.0   60000.0    Finance
```

RAM
Disk

```
0  25.0  50000.0          HR
1  30.0  60000.0     Finance
2   NaN      NaN          IT
3  35.0  70000.0     Finance
4  40.0  80000.0   Marketing

Forward Fill Missing Values:
    Age   Salary Department
0  25.0  50000.0          HR
1  30.0  60000.0     Finance
2  30.0  60000.0          IT
3  35.0  70000.0          IT
4  40.0  80000.0   Marketing

Backward Fill Missing Values
    Age   Salary Department
0  25.0  50000.0          HR
1  30.0  60000.0     Finance
2  35.0  70000.0          IT
3  35.0  70000.0   Marketing
4  40.0  80000.0   Marketing
<ipython-input-14-a3035a2677
The behavior will change in
```

3s

```
import pandas as pd
import numpy as np
from sklearn.preprocessing imp
from sklearn.compose import Co
from sklearn.pipeline import P

# Sample DataFrame with numer
data = {
    'Age': [25, 30, 35, 40, 45
    'Salary': [50000, 60000, 7
    'Department': ['HR', 'Fina
    'Experience': [2, 5, 3, 8
}

df = pd.DataFrame(data)

print("Original DataFrame:")
print(df)

# --- 1. Scaling (Normalizatio
def scale_data(df):
    # Normalization (Min-Max S
    scaler = MinMaxScaler()
    df[['Age', 'Salary', 'Expe
```

3s

```
# --- 1. Scaling (Normalizatio
def scale_data(df):
    # Normalization (Min-Max S
    scaler = MinMaxScaler()
    df[['Age', 'Salary', 'Expe
    print("\nData after Normal
    print(df)

    # Standardization (Z-score
    scaler = StandardScaler()
    df[['Age', 'Salary', 'Expe
    print("\nData after Standa
    print(df)

# Call the scale_data function
scale_data(df)

# --- 2. Encoding Categorical
def encode_data(df):
    # One-Hot Encoding for cat
    df_encoded = pd.get_dummie
    print("\nData after One-Ho
    print(df_encoded)

    # Label Encoding for categ
```

Ds lab-4

RAM
Disk

3s

```
    # Label Encoding for categ
    label_encoder = LabelEncod
    df['Department_Label'] = l
    print("\nData after Label
    print(df)

# Call the encode_data functio
encode_data(df)
```

```
Original DataFrame:
   Age  Salary Department  E
0   25   50000         HR
1   30   60000    Finance
2   35   70000         IT
3   40   80000    Finance
4   45   90000  Marketing

Data after Normalization (Mi
    Age  Salary Department
0  0.00    0.00         HR
1  0.25    0.25    Finance
2  0.50    0.50         IT
3  0.75    0.75    Finance
4  1.00    1.00  Marketing
```

RAM
Disk

```
Data after Normalization (Mi
     Age  Salary Department
0   0.00    0.00         HR
1   0.25    0.25    Finance
2   0.50    0.50         IT
3   0.75    0.75    Finance
4   1.00    1.00  Marketing

Data after Standardization (
        Age    Salary Depart
0 -1.414214 -1.414214
1 -0.707107 -0.707107    Fir
2  0.000000  0.000000
3  0.707107  0.707107    Fir
4  1.414214  1.414214  Marke

Data after One-Hot Encoding:
        Age    Salary  Exper
0 -1.414214 -1.414214   -1.1
1 -0.707107 -0.707107   -0.1
2  0.000000  0.000000   -0.8
3  0.707107  0.707107    0.7
4  1.414214  1.414214    1.4
```

RAM
Disk

```
2  0.000000  0.000000    -0.8
3  0.707107  0.707107     0.7
4  1.414214  1.414214     1.4

   Department_Marketing
0                 False
1                 False
2                 False
3                 False
4                  True

Data after Label Encoding:
        Age    Salary Depart
0 -1.414214 -1.414214
1 -0.707107 -0.707107    Fir
2  0.000000  0.000000
3  0.707107  0.707107    Fir
4  1.414214  1.414214  Marke
```

Ds lab-4

RAM
Disk

4 1.414214 1.414214 Marke

0s

```
import pandas as pd
import numpy as np
from sklearn.preprocessing imp

# Load dataset
from sklearn.datasets import
data = load_iris()
df = pd.DataFrame(data.data,
df['target'] = data.target

# 1) Creating a New Feature: P
df['petal_area'] = df['petal

# 2) Binning (Converting Cont
bins = [0, 5.0, 10.0]
labels = ['Small', 'Large']
df['petal_area_bin'] = pd.cut

# 3) Feature Scaling (Standar
scaler = StandardScaler()
df[df.columns[:-2]] = scaler.

# Output Results
print("Data after feature engi
```

RAM
Disk

0s

```
# Output Results
print("Data after feature engi
print(df.head())
```

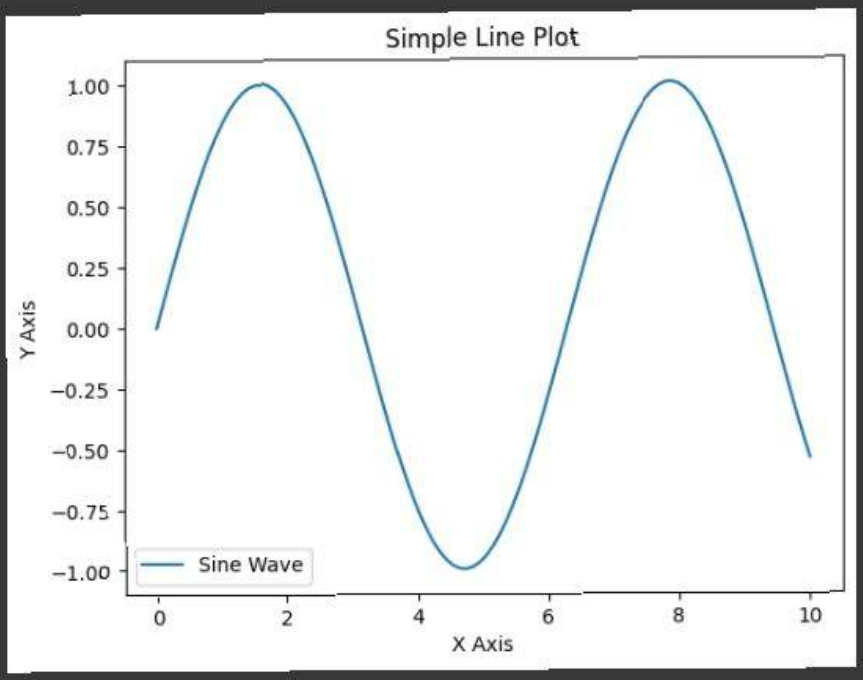
```
Data after feature engineeri
   sepal length (cm)  sepal
0          -0.900681
1          -1.143017
2          -1.385353
3          -1.506521
4          -1.021849

      target  petal_area peta
0 -1.224745        0.28
1 -1.224745        0.28
2 -1.224745        0.26
3 -1.224745        0.30
4 -1.224745        0.28
```

DS lab 5

RAM
Disk

1s

```
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(0, 10, 100)
y = np.sin(x)
plt.plot(x, y, label="Sine Wave")
plt.xlabel("X Axis")
plt.ylabel("Y Axis")
plt.title("Simple Line Plot")
plt.legend()
plt.show()
```



0s

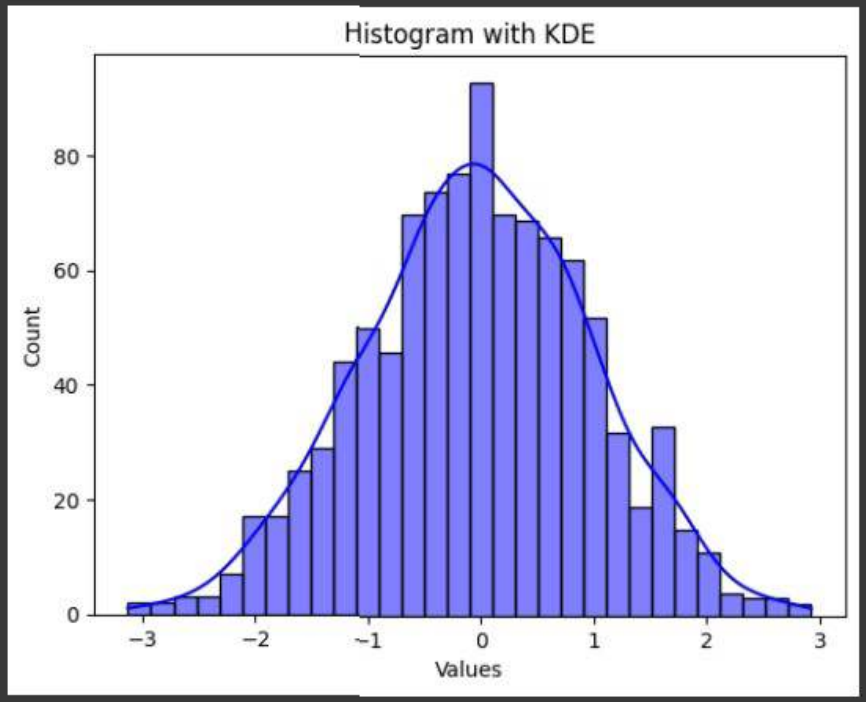
```
categories = ['A', 'B', 'C', 'D']
values = [10, 25, 15, 30]
plt.figure(figsize=(6, 4))
plt.bar(categories, values, color='
plt.xlabel("Categories")
plt.ylabel("Values")
plt.title("Bar Chart Example")
plt.show()
```

DS lab 5

RAM
Disk

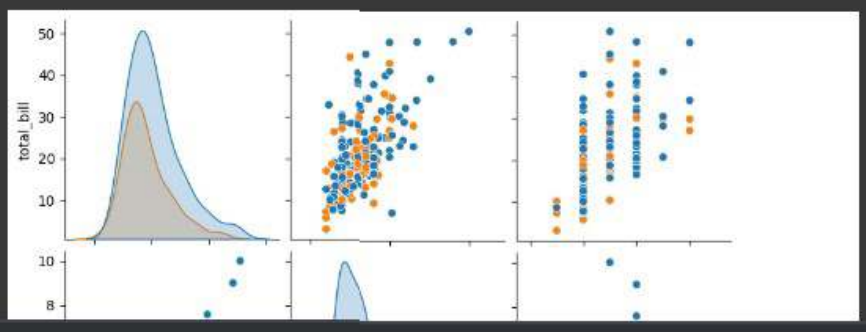
0s

```
import seaborn as sns
import pandas as pd
# Creating sample data
data = np.random.randn(1000)
df = pd.DataFrame(data, columns=['\
# Plot
sns.histplot(df['Values'], bins=30,
plt.title("Histogram with KDE")
plt.show()
```



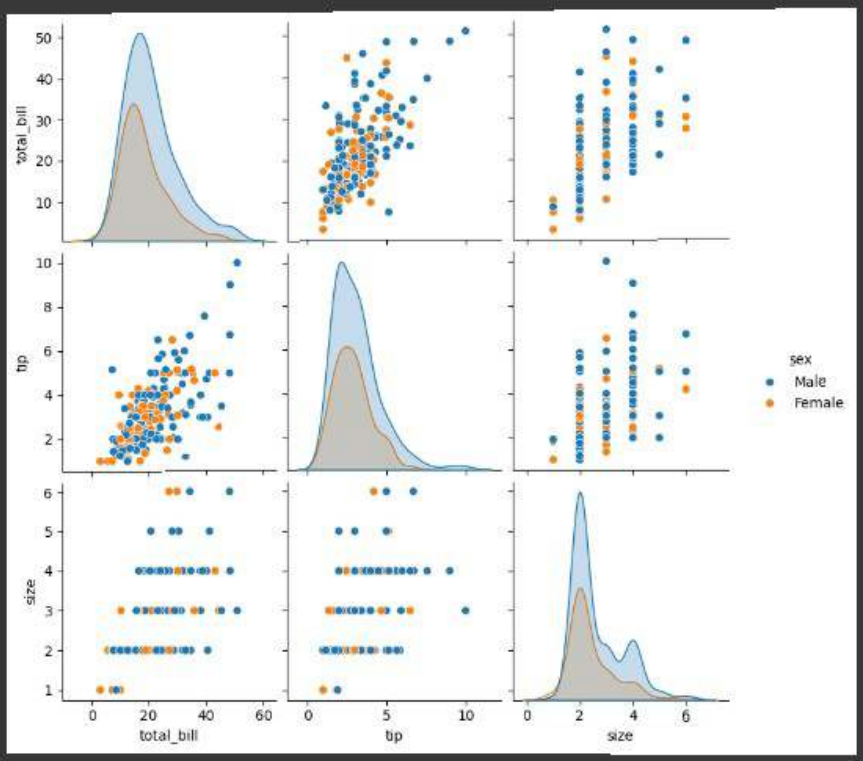
3s

```
sns.pairplot(tips, hue='sex')
plt.show()
```



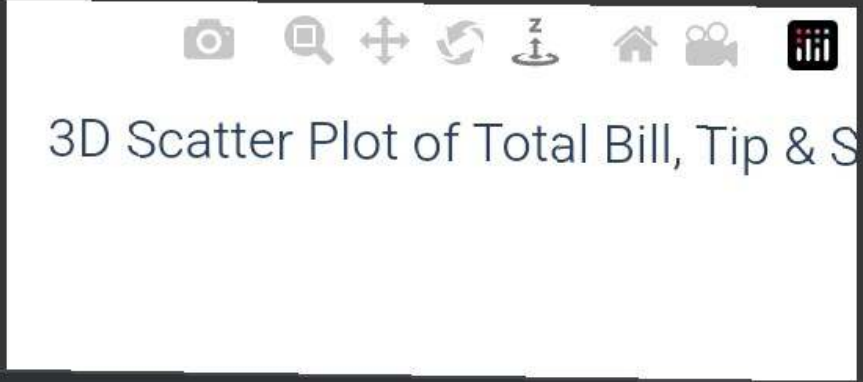
DS lab 5

RAM
Disk



0s

```
import plotly.graph_objects as go
fig = go.Figure(data=[go.Scatter3d(
 x=tips['total_bill'],
 y=tips['tip'],
 z=tips['size'],
 mode='markers',
 marker=dict(size=5, color=tips['to
)])
fig.update_layout(title="3D Scatter
fig.show()
```



DS lab 5 - ...

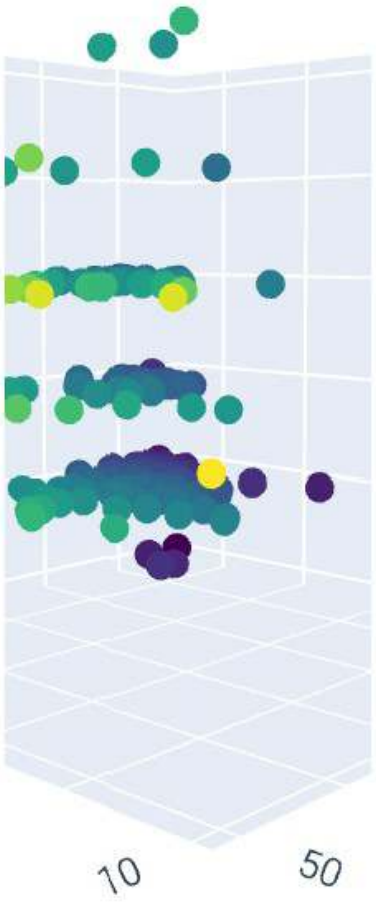
esearch.google.com

DS lab 5

RAM

Disk

3D Scatter Plot of Total Bill, Tip & S



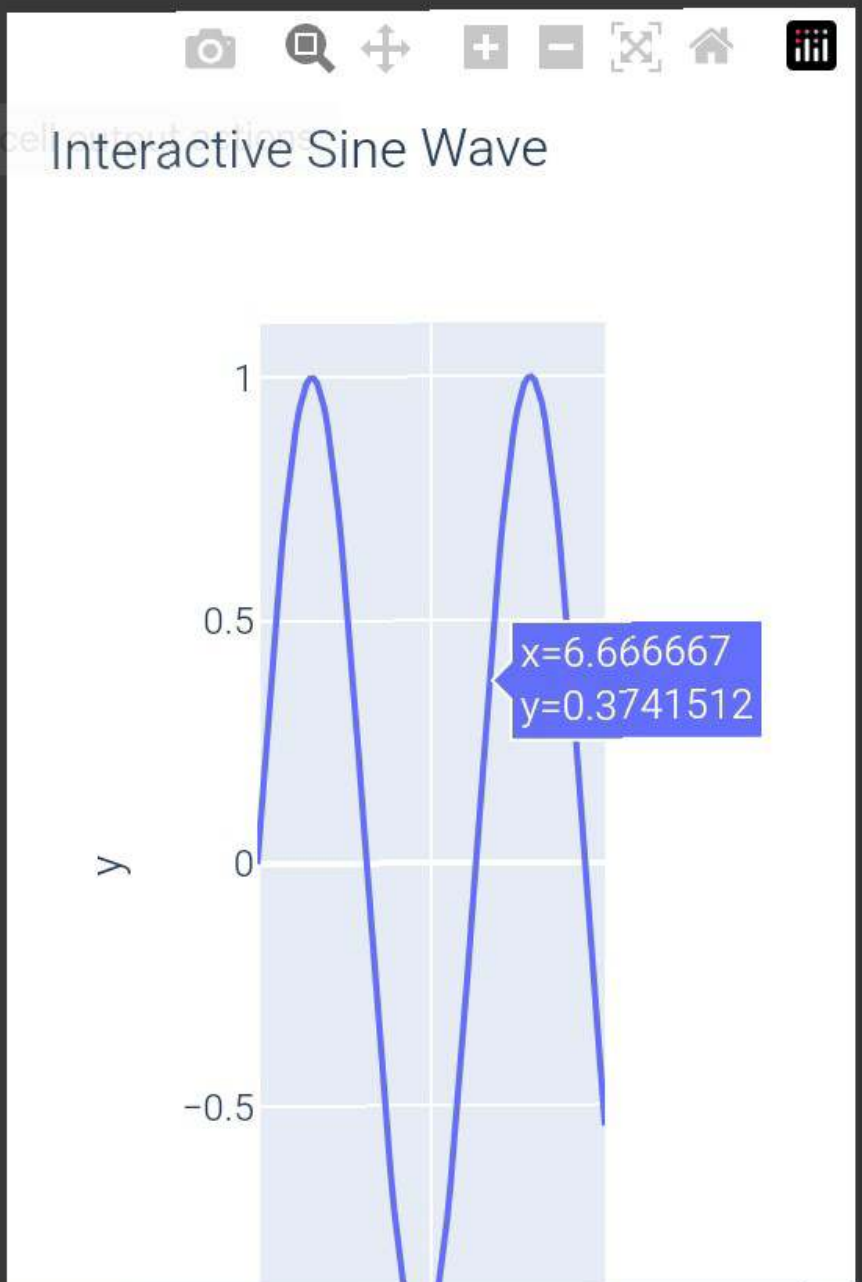
DS lab 5 - ...
esearch.google.com

DS lab 5

RAM
Disk

0s

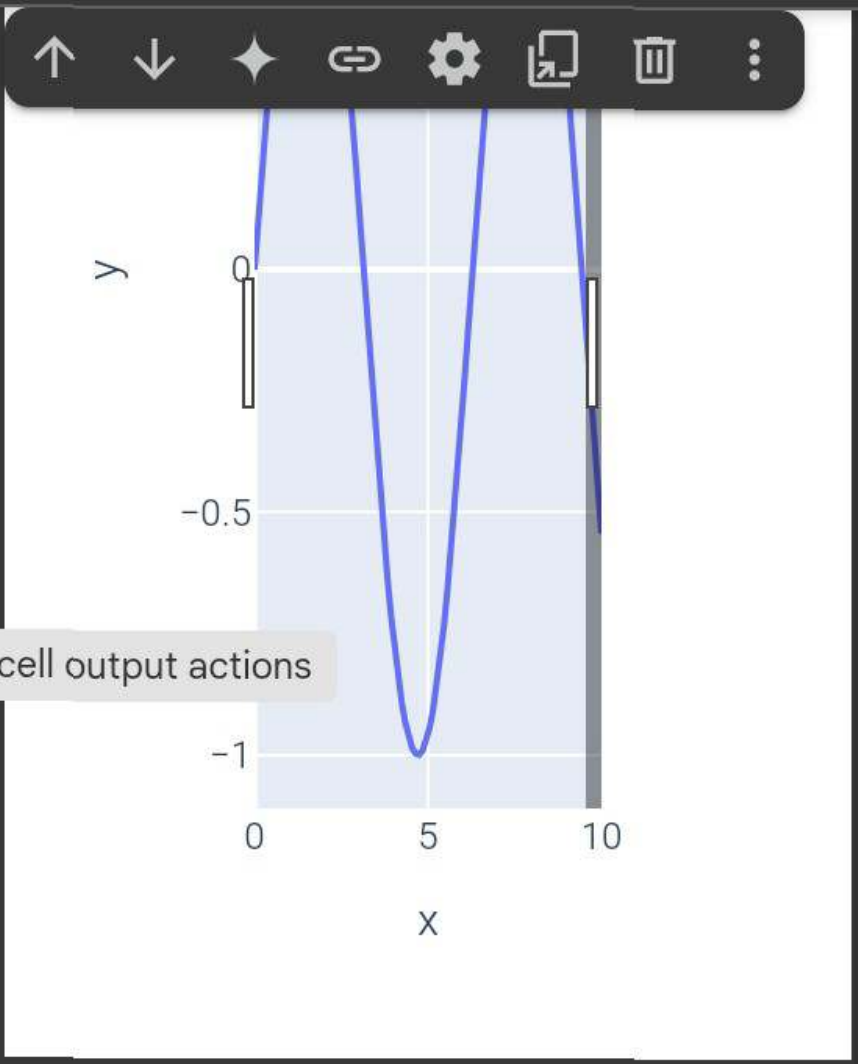
```
import plotly.express as px
df = pd.DataFrame({
"x": np.linspace(0, 10, 100),
"y": np.sin(np.linspace(0, 10, 100)
})
fig = px.line(df, x='x', y='y', tit
fig.show()
```



DS lab 5 - ...
esearch.google.com

DS lab 5

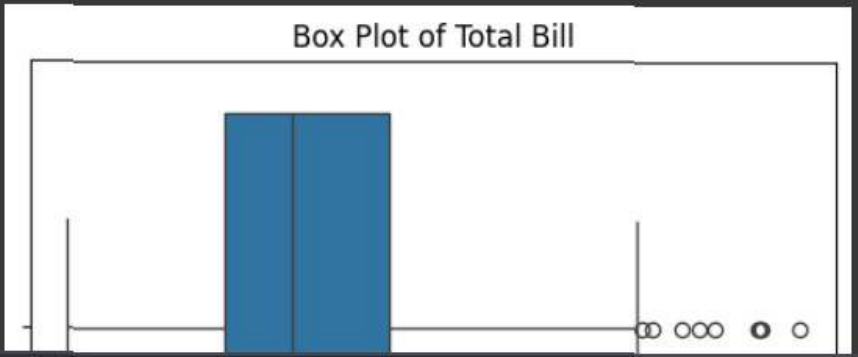
RAM
Disk



Code cell output actions

0s

```
tips = sns.load_dataset('tips')
plt.figure(figsize=(6, 4))
sns.boxplot(x=tips['total_bill'])
plt.title("Box Plot of Total Bill")
plt.show()
```



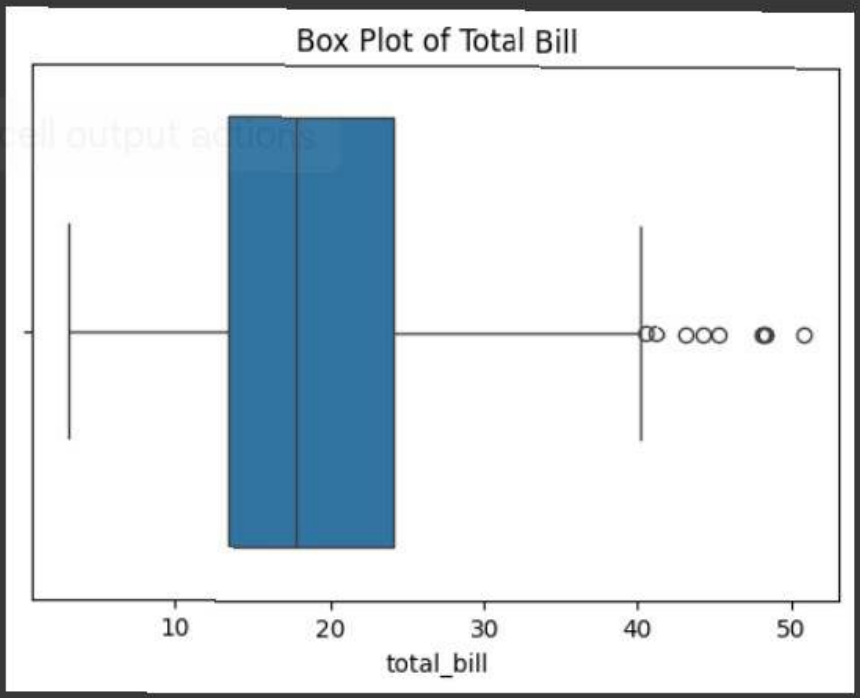
DS lab 5 - ...
esearch.google.com

DS lab 5

RAM
Disk

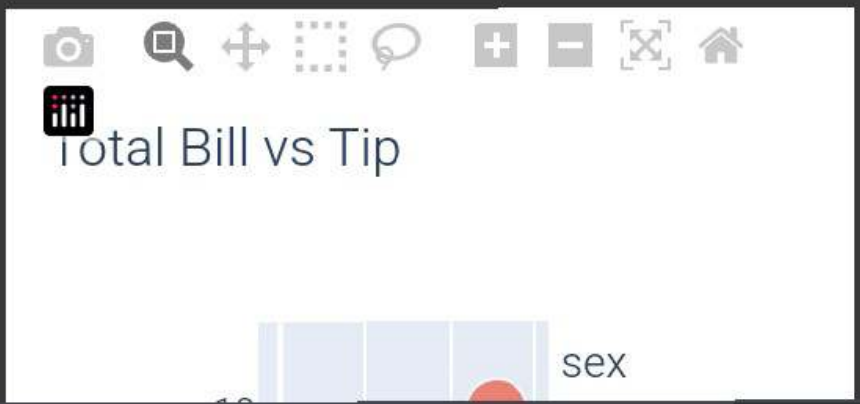
0s

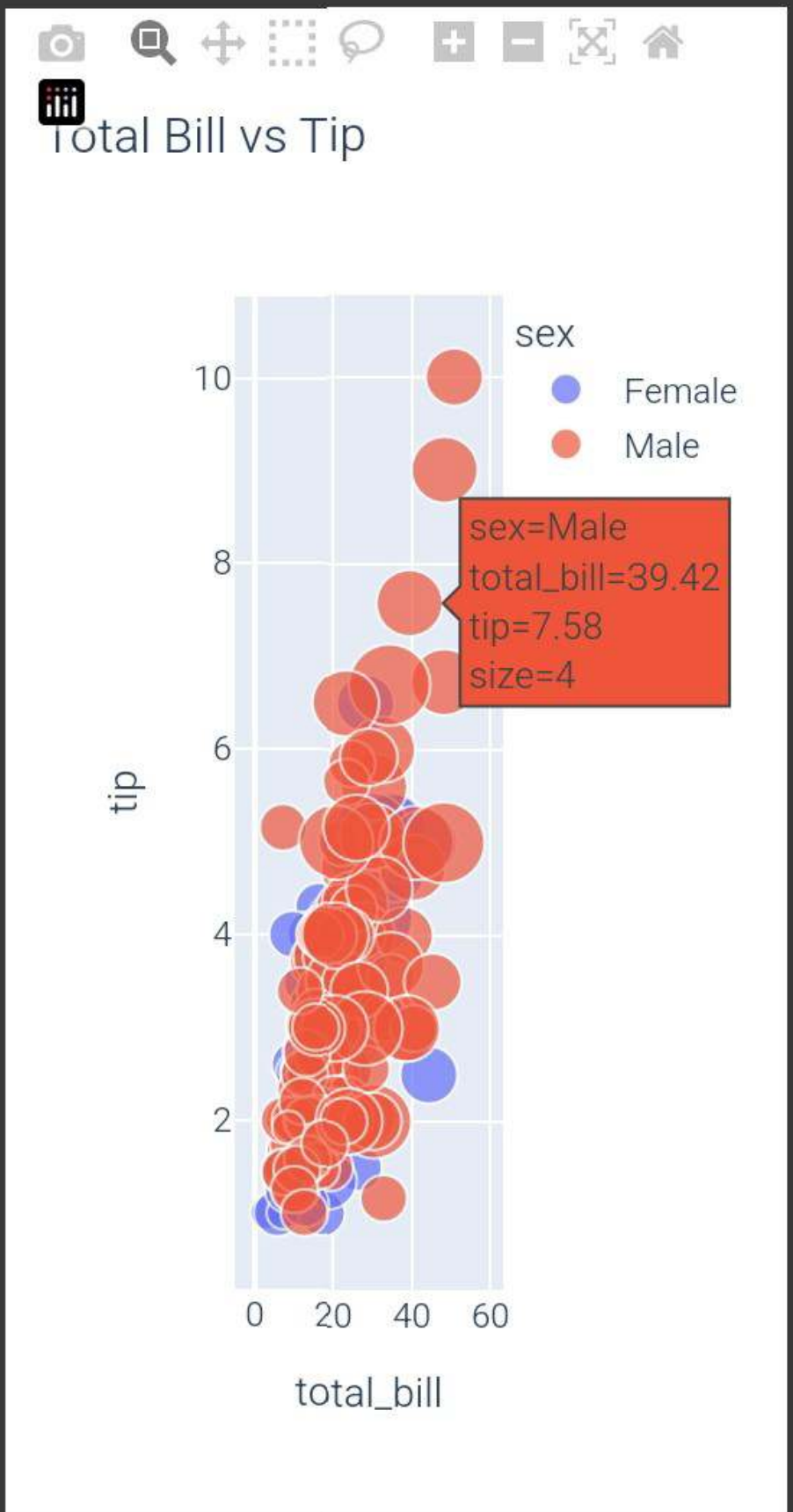
```
tips = sns.load_dataset('tips')
plt.figure(figsize=(6, 4))
sns.boxplot(x=tips['total_bill'])
plt.title("Box Plot of Total Bill")
plt.show()
```



0s

```
fig = px.scatter(tips, x='total_bil
fig.show()
```

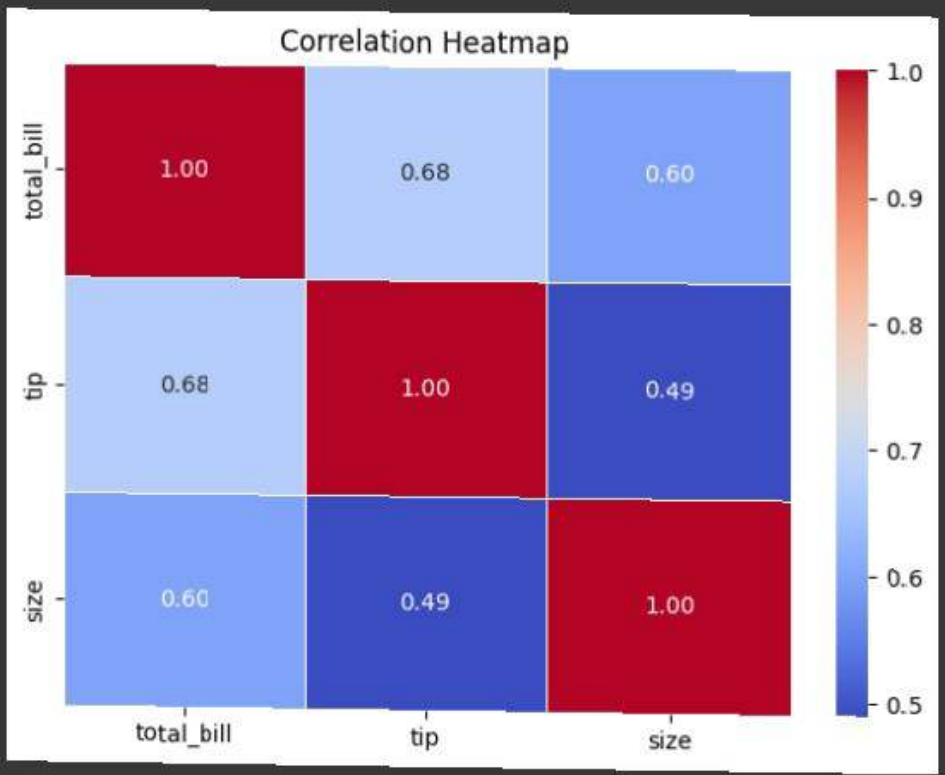




DS lab 5

0s

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Load the dataset
tips = sns.load_dataset("tips")

# Compute the correlation matrix (c
corr_matrix = tips.corr(numeric_on]

# Create the heatmap
plt.figure(figsize=(7, 5))
sns.heatmap(corr_matrix, annot=True
plt.title("Correlation Heatmap")
plt.show()
```



DS lab 1

RAM
Disk

[1] 0s
```
print("Hello, Excited to start Data
```
```
Hello, Excited to start Data Sci
```

[2] 0s
```
customer_name = "John Doe"
customer_age = 28
customer_balance = 120.75
print("Customer:", customer_name)
print("Age:", customer_age)
print("Balance:", customer_balance)
```
```
Customer: John Doe
Age: 28
Balance: 120.75
```

[3] 0s
```
product_price = 200
discount = product_price * 0.10 # 1
final_price = product_price - disc
print("Final Price after Discount:"
```
```
Final Price after Discount: 180.0
```

[4]
```
import numpy as np
sales = np.array([150, 200, 250, 30
print("Sales Data:", sales)
```
```
Sales Data: [150 200 250 300 400
```

[5]
```
print("Average Sales:", np.mean(sal
```

DS lab 1

RAM
Disk

```
[5]  print("Average Sales:", np.mean(sa
     print("Highest Sale:", np.max(sales
     print("Lowest Sale:", np.min(sales)
```

```
Average Sales: 307.1428571428571
Highest Sale: 500
Lowest Sale: 150
```

```
[6]  import pandas as pd
     data = {
     "Customer": ["Alice", "Bob", "Charl
     "Age": [25, 30, 35],
     "Amount Spent": [120, 200, 150]
     }
     df = pd.DataFrame(data)
     print(df)
```

```
  Customer  Age  Amount Spent
0    Alice   25           120
1      Bob   30           200
2  Charlie   35           150
```

```
[9]  high_spenders = df[df["Amount Spent
     print(high_spenders)
```

```
  Customer  Age  Amount Spent
1      Bob   30           200
```

```
[10] df_sorted = df.sort_values(by="Amou
     print(df_sorted)
```

20:08

research.google.com

DS lab 1

RAM
Disk

```
[9]  high_spenders = df[df["Amount Spent
     print(high_spenders)
```

```
  Customer  Age  Amount Spent
1      Bob   30           200
```

```
[10] df_sorted = df.sort_values(by="Amou
     print(df_sorted)
```

```
  Customer  Age  Amount Spent
1      Bob   30           200
2  Charlie   35           150
0    Alice   25           120
```

```
[11] df["Loyalty Points"] = df["Amount S
     print(df)
```

```
  Customer  Age  Amount Spent  Lo
0    Alice   25           120
1      Bob   30           200
2  Charlie   35           150
```

RAM
Disk

0s

```
import pandas as pd

# Creating patient data
data = {
    "Patient_ID": [101, 102, 103, 1
    "Name": ["John Doe", "Jane Smit
    "Age": [45, 38, 50, 29, 41, 60]
    "Heart_Rate (bpm)": [72, 110, 7
    "Systolic_BP (mmHg)": [120, 135
    "Diastolic_BP (mmHg)": [80, 90,
    "Temperature (°C)": [36.5, 38.2
}

# Creating DataFrame
df = pd.DataFrame(data)

# Define risk score based on conditi
df["Risk_Score"] = (
    (df["Heart_Rate (bpm)"] > 100).
    (df["Systolic_BP (mmHg)"] > 130
    (df["Diastolic_BP (mmHg)"] > 90
    (df["Temperature (°C)"] > 38).a
)

# Assigning Risk Level based on Risk
def calculate_risk_level(score):
    if score == 0:
        return "Low"
    elif score == 1:
        return "Moderate"
    elif score == 2:
        return "High"
```

```
        (df["Heart_Rate (bpm)"] > 100).
        (df["Systolic_BP (mmHg)"] > 13(
        (df["Diastolic_BP (mmHg)"] > 9(
        (df["Temperature (°C)"] > 38).a
    )

    # Sorting by risk score in descendi
    df_sorted = df.sort_values(by="Risk

    # Display sorted patients
    print("Patients Sorted by Health Ri
    print(df_sorted[["Patient_ID", "Nam

    # Saving to CSV (optional)
    df_sorted.to_csv("sorted_health_ris
    print("\nSorted patient data saved
```

```
Patients Sorted by Health Risk:

   Patient_ID             Name  Ri:
5         106    Michael Lee
1         102     Jane Smith
2         103  Alice Johnson
0         101       John Doe
3         104   Robert Brown
4         105    Emily Davis

Sorted patient data saved to 'so
```

```
import pandas as pd

# Creating patient data
data = {
    "Patient_ID": [101, 102, 103,
```

```
   Systolic_BP (mmHg)  Diastolic_
0                 120
1                 125
2                 118
3                 130
4                 122

Basic Statistics:

       Patient_ID        Age  He
count    5.000000   5.000000
mean   103.000000  40.600000
std      1.581139   7.893035
min    101.000000  29.000000
25%    102.000000  38.000000
50%    103.000000  41.000000
75%    104.000000  45.000000
max    105.000000  50.000000

       Diastolic_BP (mmHg)  Temp
count             5.000000
mean             83.000000
std               4.690416
min              78.000000
25%              80.000000
50%              82.000000
75%              85.000000
max              90.000000
```

0s

```
import pandas as pd

# Creating patient data
```

```
High-Risk Patients:

   Patient_ID             Name  Ag
1         102     Jane Smith   3
2         103  Alice Johnson   5
5         106    Michael Lee   6

   Systolic_BP (mmHg)  Diastolic
1                 135
2                 140
5                 150

High-risk patient data saved to
```

0s

```python
import pandas as pd

# Creating patient data
data = {
    "Patient_ID": [101, 102, 103,
    "Name": ["John Doe", "Jane Smi
    "Age": [45, 38, 50, 29, 41, 60
    "Heart_Rate (bpm)": [72, 110,
    "Systolic_BP (mmHg)": [120, 13
    "Diastolic_BP (mmHg)": [80, 90,
    "Temperature (°C)": [36.5, 38.
}

# Creating DataFrame
df = pd.DataFrame(data)

# Define risk score based on condi
df["Risk_Score"] = (
    (df["Heart_Rate (bpm)"] > 100).
```

```
[ ] import pandas as pd

    # Creating patient data
    data = {
        "Patient_ID": [101, 102, 103, 1
        "Name": ["John Doe", "Jane Smit
        "Age": [45, 38, 50, 29, 41],
        "Gender": ["Male", "Female", "F
        "Blood_Type": ["O+", "A-", "B+"
        "Heart_Rate (bpm)": [72, 80, 76
        "Systolic_BP (mmHg)": [120, 125
        "Diastolic_BP (mmHg)": [80, 85,
        "Temperature (°C)": [36.5, 37.0
    }

    # Creating DataFrame
    df = pd.DataFrame(data)

    # Display the DataFrame
    print("Patient Records:\n")
    print(df)

    # Display basic statistics
    print("\nBasic Statistics:\n")
    print(df.describe())
```

```
Patient Records:

   Patient_ID           Name  Ag
0         101       John Doe   4
1         102     Jane Smith   3
2         103  Alice Johnson   5
3         104   Robert Brown   2
4         105    Emily Davis   4
```

```
   Patient_ID            Name  Ag
1         102     Jane Smith   3
2         103  Alice Johnson   5
5         106    Michael Lee   6

   Systolic_BP (mmHg)  Diastolic_

1                 135
2                 140
5                 150

High-risk patient data saved to
```

```
[ ] import pandas as pd

    # Creating patient data
    data = {
        "Patient_ID": [101, 102, 103, 1
        "Name": ["John Doe", "Jane Smit
        "Age": [45, 38, 50, 29, 41, 60]
        "Heart_Rate (bpm)": [72, 110, 7
        "Systolic_BP (mmHg)": [120, 135
        "Diastolic_BP (mmHg)": [80, 90,
        "Temperature (°C)": [36.5, 38.2
    }

    # Creating DataFrame
    df = pd.DataFrame(data)

    # Define risk score based on condit
    df["Risk_Score"] = (
        (df["Heart_Rate (bpm)"] > 100).
        (df["Systolic_BP (mmHg)"] > 130
        (df["Diastolic_BP (mmHg)"] > 90
        (df["Temperature (°C)"] > 38).a
```

```
BMI: 24.49
Category: Normal weight
```

0s

```python
import numpy as np

# Simulated patient vital signs
heart_rates = np.array([72, 75,
systolic_bp = np.array([120, 122
diastolic_bp = np.array([80, 82,
body_temperatures = np.array([36

# Compute basic statistics
heart_rate_mean = np.mean(heart_
heart_rate_std = np.std(heart_ra

systolic_mean = np.mean(systolic
diastolic_mean = np.mean(diastol

temperature_mean = np.mean(body_
temperature_max = np.max(body_te
temperature_min = np.min(body_te

# Display the analysis results
print(f"Heart Rate - Mean: {hear
print(f"Blood Pressure - Mean: {
print(f"Body Temperature - Mean:
```

```
Heart Rate - Mean: 76.50 bpm,
Blood Pressure - Mean: 121.80/
Body Temperature - Mean: 36.76
```

0s

```python
import pandas as pd

# Creating patient data
```

DS lab

RAM
Disk

0s

```
import pandas as pd

# Creating patient data
data = {
    "Patient_ID": [101, 102, 103, 1
    "Name": ["John Doe", "Jane Smit
    "Age": [45, 38, 50, 29, 41, 60]
    "Gender": ["Male", "Female", "F
    "Heart_Rate (bpm)": [72, 110, 7
    "Systolic_BP (mmHg)": [120, 135
    "Diastolic_BP (mmHg)": [80, 90,
    "Temperature (°C)": [36.5, 38.2
}

# Creating DataFrame
df = pd.DataFrame(data)

# Filtering high-risk patients
high_risk_patients = df[
    (df["Heart_Rate (bpm)"] > 100)
    (df["Systolic_BP (mmHg)"] > 130
    (df["Diastolic_BP (mmHg)"] > 90
    (df["Temperature (°C)"] > 38)
]

# Display high-risk patients
print("High-Risk Patients:\n")
print(high_risk_patients)

# Saving to CSV (optional)
high_risk_patients.to_csv("high_ris
print("\nHigh-risk patient data sav
```

0s

```python
import pandas as pd

# Creating patient data
data = {
    "Patient_ID": [101, 102, 103, 1
    "Name": ["John Doe", "Jane Smit
    "Age": [45, 38, 50, 29, 41, 60]
    "Gender": ["Male", "Female", "M
    "Heart_Rate (bpm)": [72, 110, 7
    "Systolic_BP (mmHg)": [120, 135
    "Diastolic_BP (mmHg)": [80, 90,
    "Temperature (°C)": [36.5, 38.2
}

# Creating DataFrame
df = pd.DataFrame(data)

# Filtering high-risk patients
high_risk_patients = df[
    (df["Heart_Rate (bpm)"] > 100)
    (df["Systolic_BP (mmHg)"] > 130
    (df["Diastolic_BP (mmHg)"] > 90
    (df["Temperature (°C)"] > 38)
]

# Display high-risk patients
print("High-Risk Patients:\n")
print(high_risk_patients)

# Saving to CSV (optional)
high_risk_patients.to_csv("high_ris
print("\nHigh-risk patient data sav
```

RAM
Disk

0s

```python
import pandas as pd

# Creating patient data
data = {
    "Patient_ID": [101, 102, 103, 1
    "Name": ["John Doe", "Jane Smit
    "Age": [45, 38, 50, 29, 41],
    "Gender": ["Male", "Female", "F
    "Blood_Type": ["O+", "A-", "B+"
    "Heart_Rate (bpm)": [72, 80, 76
    "Systolic_BP (mmHg)": [120, 125
    "Diastolic_BP (mmHg)": [80, 85,
    "Temperature (°C)": [36.5, 37.(
}

# Creating DataFrame
df = pd.DataFrame(data)

# Display the DataFrame
print("Patient Records:\n")
print(df)

# Display basic statistics
print("\nBasic Statistics:\n")
print(df.describe())
```

```
Patient Records:

   Patient_ID           Name  Ag
0         101     John Doe    4
1         102   Jane Smith    3
2         103  Alice Johnson  5
3         104   Robert Brown  2
4         105    Emily Davis  4
```