Practical 1

```
import pandas as pd
df1 = pd.read_csv("/content/pract1.csv")
```

```
df1.head()
```

	id	name	Duration
0	1	Ram	02:30:00
1	2	Sham	01:15:30
2	3	Krishna	00:45:15

```
df1['hours']=pd.to_timedelta(df1['Duration']).dt.total_seconds()/3600
```

```
print(df1)
```

```
   id     name  Duration     hours
0   1      Ram  02:30:00  2.500000
1   2     Sham  01:15:30  1.258333
2   3  Krishna  00:45:15  0.754167
```

```
df=pd.read_csv("/content/pract3.csv")
df.head()
```

	id	name	Duration
0	1	Ram	14:45:30
1	2	Sham	09:15:00
2	3	Krishna	23:05:45

```
df["hours"]=pd.to_timedelta(df["Duration"]).dt.total_seconds()/3600
#df_sorted=df.sort_values(by='hours',ascending=False)
```

```
print(df)
```

```
   id     name  Duration      hours
0   1      Ram  14:45:30  14.758333
1   2     Sham  09:15:00   9.250000
2   3  Krishna  23:05:45  23.095833
```

.

.

.

.

.

.

Practical **2**

```
#Sample dataset with ages
import pandas as pd

data = {'Name': ['Alice','Bob','Charlie','David','Emma'],'Age':[5,17,34,70,25]}
df = pd.DataFrame(data)


bins = [0,12,19,59,100]
labels = ['Child','Teenager','Adult','Senior']
df['Age Group'] = pd.cut(df['Age'],bins=bins,labels=labels)


print(df)
```

```
      Name  Age Age Group
0    Alice    5     Child
1      Bob   17  Teenager
2  Charlie   34     Adult
3    David   70    Senior
4     Emma   25     Adult
```

Binning Data into Equal-Width bins

```
df['Age Bin'] = pd.cut(df['Age'],bins=3)
print(df)
```

```
      Name  Age Age Group           Age Bin
0    Alice    5     Child   (4.935, 26.667]
1      Bob   17  Teenager   (4.935, 26.667]
2  Charlie   34     Adult  (26.667, 48.333]
3    David   70    Senior    (48.333, 70.0]
4     Emma   25     Adult   (4.935, 26.667]
```

Binning data in equal frequency

```
df['Quantile Bin'] = pd.qcut(df['Age'],q=3,labels=['Low','Medium','High'])
print(df)
```

```
      Name  Age Age Group           Age Bin Quantile Bin
0    Alice    5     Child   (4.935, 26.667]          Low
1      Bob   17  Teenager   (4.935, 26.667]          Low
2  Charlie   34     Adult  (26.667, 48.333]         High
3    David   70    Senior    (48.333, 70.0]         High
4     Emma   25     Adult   (4.935, 26.667]       Medium
```

.

.

.

.

.

.

Practical 3

```
import pandas as pd
import numpy as np

data =[10,20,30,40,50]
average = sum(data)/ len(data)
print("Average:",average)
```

```
Average: 30.0
```

```
#Method 2: Using Statistical mean()
import statistics
data =[10,20,30,40,50]
average = statistics.mean(data)
print("Average:",average)
```

```
Average: 30
```

```
#Method 3: Using Numpy (For Large Dataset)
import numpy as np

data =np.array([10,20,30,40,50])
average = np.mean(data)
print("Average:",average)
```

```
Average: 30.0
```

```
import numpy as np
import pandas as pd

data =[10,20,45,35,87]
Total=np.sum(data)

print("Total:",Total)
```

```
Total: 197
```

```
mean = np.mean(data)
print("Mean =",mean)
median = np.median(data)
print("Median = ",median)
std_dev = np.std(data)
print("Standard Deviation =",std_dev)
mini = np.min(data)
print("Minimum Value =",mini)
max = np.max(data)
print("Maximum Value =",max)
q1 = np.percentile(data,25)
print("First Quartile :",q1)
```

```
Mean = 39.4
Median =  35.0
Standard Deviation = 26.672832620477337
Minimum Value = 10
Maximum Value = 87
First Quartile : 20.0
```

```
import pandas as pd
data1=[20,30,50,70,100]
print("List of Element",data1)
df = pd.DataFrame(data1)
print(df.head)
df.head()
```

```
List of Element [20, 30, 50, 70, 100]
<bound method NDFrame.head of      0
0   20
1   30
2   50
3   70
4  100>
```

	0
0	20
1	30
2	50
3	70
4	100

```
df.describe()
```

	0
count	5.000000
mean	54.000000
std	32.093613
min	20.000000
25%	30.000000
50%	50.000000
75%	70.000000
max	100.000000

```
mn = df.mean()
mdn=df.median()
std_dev=df.std()
mini=df.min()
max=df.max()
print("Mean=",mn)
print("Median=",mdn)
print("Standard Deviation=",std_dev)
print("Minimum=",mini)
print("Maximum=",max)
```

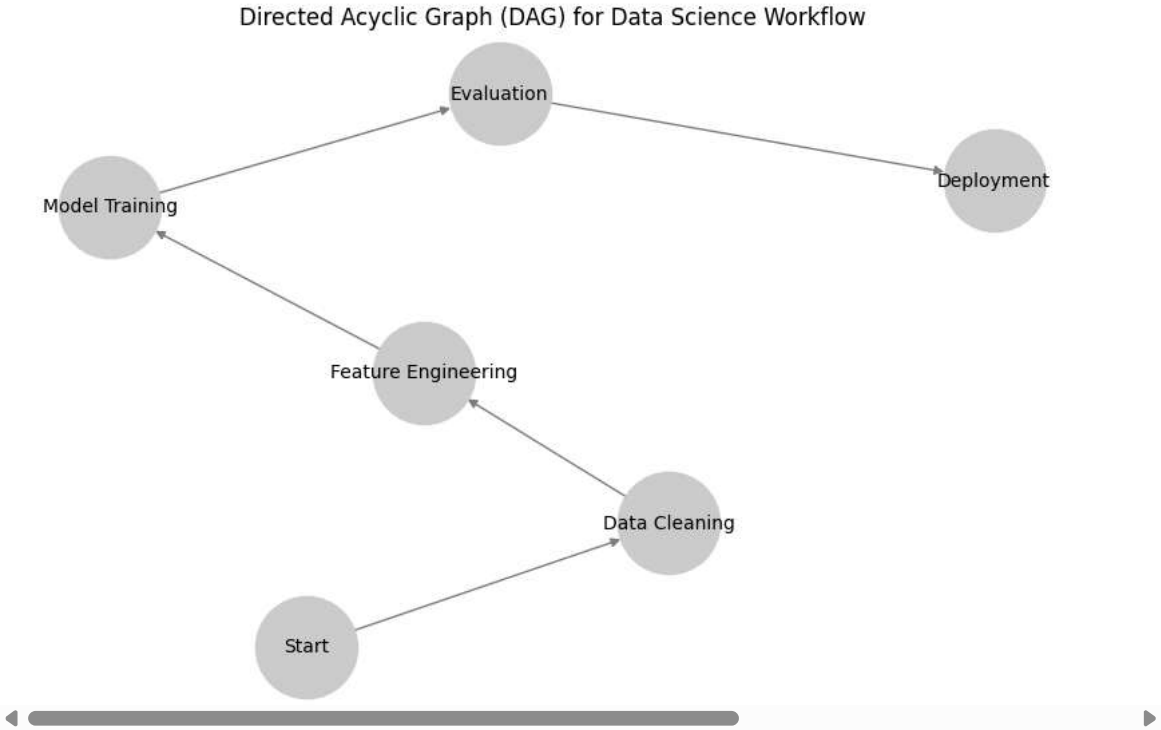
```
Mean= 0    54.0
dtype: float64
Median= 0    50.0
dtype: float64
Standard Deviation= 0    32.093613
dtype: float64
Minimum= 0    20
dtype: int64
Maximum= 0    100
dtype: int64
```

.

Practical 4

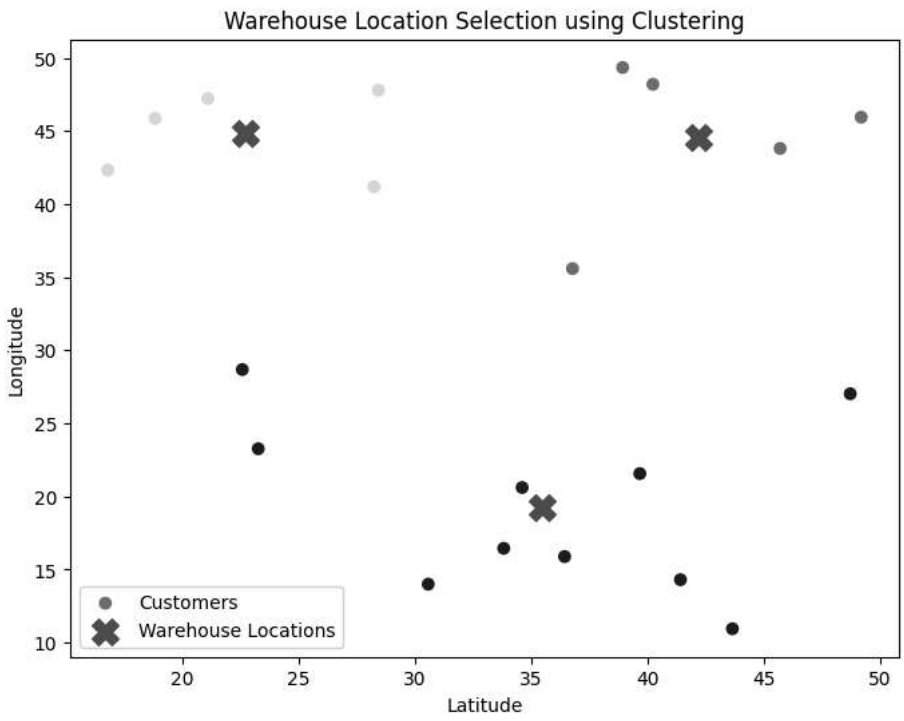
```
import networkx as nx
import matplotlib.pyplot as plt

#Create a Directed Graph (DAG)
DAG=nx.DiGraph()

#Add edges (No cycles allowed)
edges=[
    ('Start','Data Cleaning'),
    ('Data Cleaning','Feature Engineering'),
    ('Feature Engineering','Model Training'),
    ('Model Training', 'Evaluation'),
    ('Evaluation','Deployment')
    ]

DAG.add_edges_from(edges)

#Check if the graph is acyclic
if nx.is_directed_acyclic_graph(DAG):
  print("The graph is a valid DAG.")
else:
  print("The graph contains cycles.")

#Draw the DAG
plt.figure(figsize=(8,5))
pos=nx.spring_layout(DAG) #Layout for better visualization
nx.draw(DAG,pos,with_labels=True, node_size=3000, node_color="lightblue", edge_color="gray", font_size=10)
plt.title("Directed Acyclic Graph (DAG) for Data Science Workflow")
plt.show()
```

```
The graph is a valid DAG.
```



Practical 5

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from google.colab import files

#Step 1: Load or Simulate Customer Location Data
#Simulated data (Customer locations: latitude, longitude)
data={
    'Customer_ID': range(1,21),
    'Latitude': np.random.uniform(10,50,20), #Random latitudes
    'Longitude': np.random.uniform(10,50,20) #Random longitudes
}
df=pd.DataFrame(data)

#Step 2: Apply K-Means Clustering (Choose 3 clusters for warehouse locations)
k= 3
kmeans= KMeans (n_clusters=k, random_state=42)
df['Cluster']=kmeans.fit_predict(df[['Latitude', 'Longitude']])

#Step 3: Get Cluster Centers (Warehouse Locations)
warehouse_locations=kmeans.cluster_centers_

#Step 4: Visualization
plt.figure(figsize=(8,6))
plt.scatter(df['Latitude'], df['Longitude'], c=df['Cluster'], cmap='viridis', label='Customers')
plt.scatter(warehouse_locations[:,0], warehouse_locations[:,1], color='red', marker='X', s=200, label='Warehouse Locati

plt.xlabel('Latitude')
plt.ylabel('Longitude')
plt.title('Warehouse Location Selection using Clustering')
plt.legend()
plt.show()

#Step 5: Print Suggested Warehouse Locations
for i, loc in enumerate(warehouse_locations):
  print(f"Warehouse{i+1} Location: Latitude {loc[0]:.2f}, Longitude {loc[1]:.2f}")
```




```
Warehouse1 Location: Latitude 35.47, Longitude 19.27
Warehouse2 Location: Latitude 42.17, Longitude 44.58
```

Practical 6

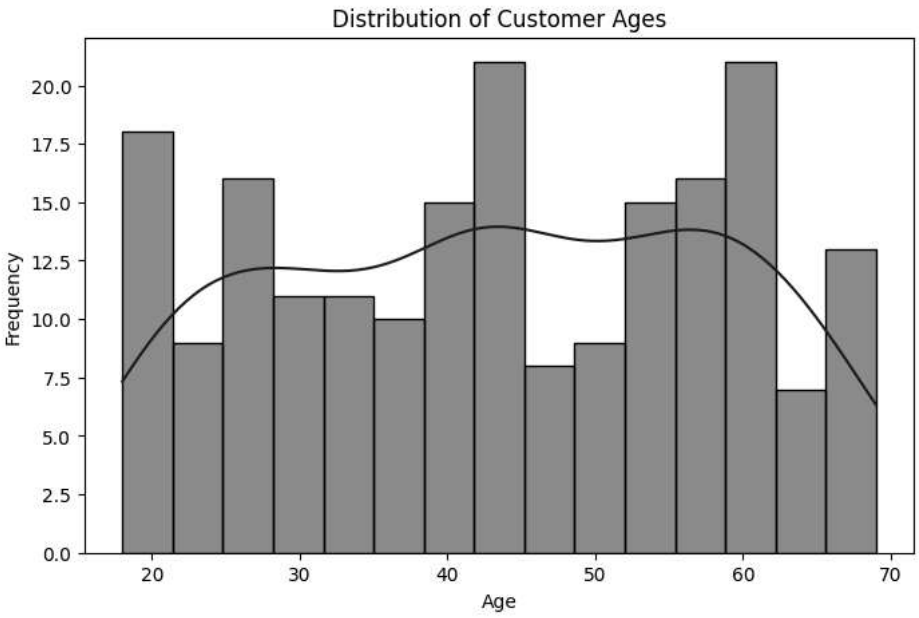
```
import pandas as pd
import datetime

#Step 1: Create the Time Hub (Unique Events)
hub_time=pd.DataFrame({
    'Event_ID': [101,102,103,104], #Unique keys
    'Event_Name': ['Order Placed','Order Shipped','Payment Processed','Order Delivered'],
    'Event_Timestamp': [
        datetime.datetime(2024,2,1,10,30),
        datetime.datetime(2024,2,1,11,0),
        datetime.datetime(2024,2,1,11,15),
        datetime.datetime(2024,2,1,12,0)]
})

#Step 2: Create the Link Table (Relationships)
link_event_location=pd.DataFrame({
    'Event_ID': [101,102,103,104], #Foreign key from hub
    'Location_ID': ['L001','L002','L003','L004'], #Location where event happened
    'Link_Hash': ['H1','H2','H3','H4'] #Unique relationship hash
})

#Step 3: Create the Satelite Table (Descriptive attributes)
sat_event_details=pd.DataFrame({
    'Event_ID': [101,102,103,104], #Foreign key from hub
    'Description': ['Order received from user', 'Package shipped via DHL', 'Payment confirmed', 'Package delivered succ
    'Recorded_At': [
        datetime.datetime(2024,2,1,10,35),
        datetime.datetime(2024,2,1,11,5),
        datetime.datetime(2024,2,1,11,20),
        datetime.datetime(2024,2,1,12,5)]
})

#Step 4: Print DataFrames
print("◆ Time Hub (Event Information)")
print(hub_time)

print("\n◆ Link Table (Event-Location Relationship)")
print(link_event_location)

print("\n◆ Satelite Table (Event Details)")
print(sat_event_details)
```

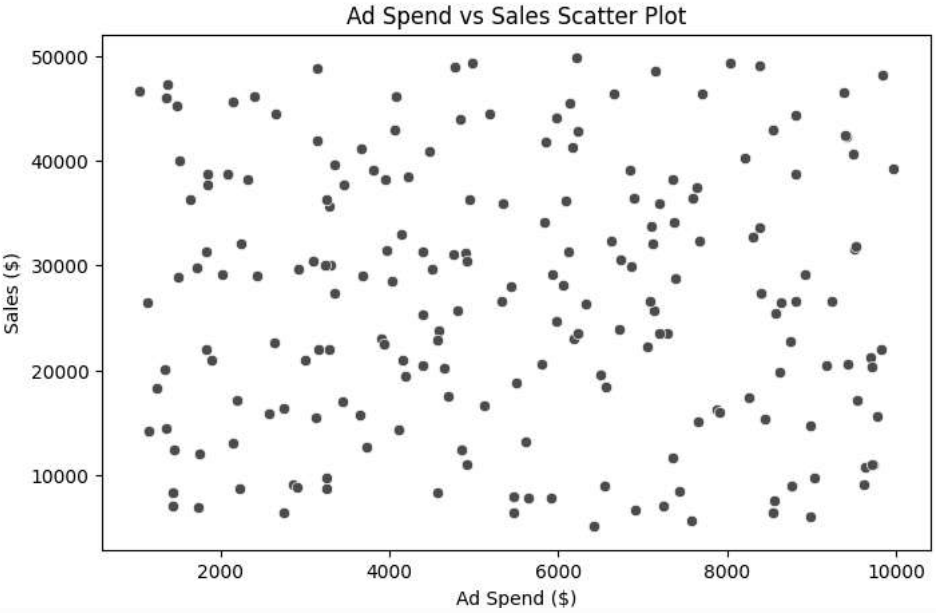
```
◆ Time Hub (Event Information)
   Event_ID         Event_Name     Event_Timestamp
0       101       Order Placed 2024-02-01 10:30:00
1       102      Order Shipped 2024-02-01 11:00:00
2       103  Payment Processed 2024-02-01 11:15:00
3       104    Order Delivered 2024-02-01 12:00:00

◆ Link Table (Event-Location Relationship)
   Event_ID Location_ID Link_Hash
0       101        L001        H1
1       102        L002        H2
2       103        L003        H3
3       104        L004        H4

◆ Satelite Table (Event Details)
   Event_ID                    Description         Recorded_At
0       101        Order received from user 2024-02-01 10:35:00
1       102         Package shipped via DHL 2024-02-01 11:05:00
2       103               Payment confirmed 2024-02-01 11:20:00
3       104  Package delivered successfully 2024-02-01 12:05:00
```

.

Practical 7

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

#Step 1: Create Sample Data
np.random.seed(42)
data=pd.DataFrame({
    'Age': np.random.randint(18,70,200), #Random ages
    'Income': np.random.randint(20000,120000,200), #Random Income
    'Ad_Spend': np.random.uniform(1000,10000,200), #Advertising Spend
    'Sales': np.random.uniform(5000,50000,200) #Sales Generated
})

#Step 2: Create Histogram (Distribution of Age)
plt.figure(figsize=(8,5))
sns.histplot(data['Age'], bins=15, kde=True, color='blue')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.title('Distribution of Customer Ages')
plt.show()

#Step 3: Create Scatter Plot (Ad Spend vs Sales)
plt.figure(figsize=(8,5))
sns.scatterplot(x=data['Ad_Spend'], y=data['Sales'], color='red')
plt.xlabel("Ad Spend ($)")
plt.ylabel("Sales ($)")
plt.title("Ad Spend vs Sales Scatter Plot")
plt.show()
```


Distribution of Customer Ages
Frequency
Age
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
20
30
40
50
60
70


Ad Spend vs Sales Scatter Plot
Sales ($)
Ad Spend ($)
10000
20000
30000
40000
50000
2000
4000
6000
8000
10000

Practical 8

```
import pandas as pd
import numpy as np

data={
    'ID': [102,101,104,103,105,102], #Duplicate ID 102
    'Name': ['Alice','Bob','Charlie','David','Eve', 'Alice'],
    'Age': [25, np.nan, 35, 40, 29, 25], #Missing Age
    'Salary': [50000, 60000, np.nan, 70000, 55000, 50000], #Missing Salary
    'Department': ['HR','IT','Finance','IT','HR','HR']
}

df3=pd.DataFrame(data)

print(df3)
```

```
     ID     Name   Age   Salary Department
0  102    Alice  25.0  50000.0         HR
1  101      Bob   NaN  60000.0         IT
2  104  Charlie  35.0      NaN    Finance
3  103    David  40.0  70000.0         IT
4  105      Eve  29.0  55000.0         HR
5  102    Alice  25.0  50000.0         HR
```

```
#Step 2: Remove Duplicates
df3=df3.drop_duplicates()

print(df3)
```

```
     ID     Name   Age   Salary Department
0  102    Alice  25.0  50000.0         HR
1  101      Bob   NaN  60000.0         IT
```