Page No. :

Date :

Lab-7 (Part -1)

★ Apply Apriori Algorithm on given dataset calculate support and comfidence then find the association

1.)

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

Ans →

C_1

Item	set
1	2
2	3
3	3
4	1 → Exclude
5	3

L_2

Item	min support
1	2
2	3
3	3
5	3

C_2

Itemset
{1,2}
{1,3}
{1,5}
{2,3}
{3,5}
{2,5}

C2

Itemset	min support	
{1,2}	1	X
{1,3}	2	
{1,5}	1	X
{2,3}	2	
{2,5}	3	
{3,5}	2	

L2

Itemset	min support
{1,3}	2
{2,3}	2
{2,5}	3
{3,5}	2

C3

itemset	min support	
{1,2,3}	1	X
{1,3,5}	1	X
{2,3,5}	2	

Page No. :

Date :

→ Confidence:

$A \rightarrow B$

$2 \wedge 3 \rightarrow 5$

$$\frac{\text{Support count } (2 \wedge 3 \wedge 5)}{\text{Support count } (2 \wedge 3)} = \frac{2}{2} = 1$$

→ Association Rules:

Association Rule	Support	Confidence	in (%)
$2 \rightarrow 3 \wedge 5$	2	2/3 = 0.66	66 %
$3 \rightarrow 2 \wedge 5$	2	2/3 = 0.66	66 %
$5 \rightarrow 2 \wedge 3$	2	2/3 = 0.66	66 %
$2 \wedge 3 \rightarrow 5$	2	2/2 = 1	100 %
$2 \wedge 5 \rightarrow 3$	2	2/3 = 0.66	66 %
$3 \wedge 5 \rightarrow 2$	2	2/2 = 1	100 %

Page No. :

Date :

2.

TID	Items
1	Bread, milk,
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Cola
4	Milk, Diaper, Beer, Cola
5	Bread, Milk, Diaper, Cola

Ans ⇒

	item	min support	
1	Bread	3	
2	Milk	4	
3	Diaper	4	
4	Beer	3	
5	Cola	3	
6	Eggs	1	×

itemset	min support
1	3
2	4
3	4
4	3
5	3

itemset	min support	
{1,2}	2	
{1,3}	2	
{1,4}	1	×
{1,5}	1	×
{2,3}	3	
{2,4}	2	
{2,5}	3	
{3,4}	3	
{3,5}	3	
{4,5}	2	

itemset	min. support
{1,2}	2
{1,3}	2
{2,3}	3
{2,4}	2
{2,5}	3
{3,4}	3
{3,5}	3
{4,5}	2

Page No. :

Date :

Item	min support	
{1,2,3}	1	×
{1,2,4}	0	×
{1,2,5}	1	×
{1,3,4}	1	×
{1,3,5}	1	×
{1,4,5}	1	×
{2,3,4}	2	
{2,3,5}	3	
{2,4,5}	2	
{3,4,5}	2	

Itemset	minsupport
{2,3,4}	2
{2,3,5}	3
{2,4,5}	2
{3,4,5}	2

Itemset	minsupport
{2,3,4,5}	2

Association Rule	support	Confidence	in(%)
2 → 3^4^5	2	2/4 = 0.50	50%
3 → 2^4^5	2	2/4 = 0.50	50%
4 → 2^3^5	2	2/3 = 0.66	66%
5 → 2^3^4	2	2/3 = 0.66	66%
2^3 → 4^5	2	2/3 = 0.66	66%
2^4 → 3^5	2	2/2 = 1.0	100%
2^5 → 3^4	2	2/3 = 0.66	66%
3^4 → 2^5	2	2/3 = 0.66	66%
3^5 → 2^4	2	2/3 = 0.66	66%
4^5 → 2^3	2	2/2 = 1	100%
2^3^4 → 5	2	2/2 = 1	100%
3^4^5 → 2	2	2/2 = 1	100%
2^4^5 → 3	2	2/2 = 1	100%
2^3^5 → 4	2	2/3 = 0.66	66%

Darshan UNIVERSITY
योग: कर्मसु कौशलम्

Data Mining

Lab - 7 (Part 2)

Name: Smit Maru

Enrollment No: 23010101161

Step 1: Load the Dataset

Load the `Tdata.csv` file and display the first few rows.

In [1]:
```
import pandas as pd

df = pd.read_csv("Tdata.csv")
print(df.head())
```

```
  Transaction  bread  butter  coffee  eggs  jam  milk
0          T1      1       1       0     0    0     1
1          T2      1       1       0     0    1     0
2          T3      1       0       0     1    0     1
3          T4      1       1       0     0    0     1
4          T5      1       0       1     0    0     0
```

Step 2: Drop the 'Transaction' Column

We're only interested in the items (not the transaction IDs).

In [2]:
```
df = df.drop(columns=['Transaction'])
```

Step 3: Count Single Items

See how many transactions include each item.

In [3]:
```
item_counts = df.sum()
print(item_counts)
```

```
bread     5
butter    3
coffee    2
eggs      2
jam       2
milk      3
dtype: int64
```

Step 4: Define Apriori Function

This function finds frequent itemsets of size 1, 2, and 3 with minimum support.

In [4]:
```
from itertools import combinations
import pandas as pd

def apriori(df, min_support=0.6):
    num_txns = len(df)
    items = df.columns
    freq_itemsets = []

    L1 = []
    for item in items:
        support = df[item].sum() / num_txns
        if support >= min_support:
            L1.append((frozenset([item]), support))
    freq_itemsets.extend(L1)

    L2 = []
    for itemset in combinations([i[0] for i in L1], 2):
        union_items = itemset[0].union(itemset[1])
        support = (df[list(union_items)].sum(axis=1) == len(union_items)).mean()
        if support >= min_support:
            L2.append((union_items, support))
    freq_itemsets.extend(L2)

    L3 = []
    for itemset in combinations([i[0] for i in L1], 3):
        union_items = itemset[0].union(itemset[1]).union(itemset[2])
        support = (df[list(union_items)].sum(axis=1) == len(union_items)).mean()
        if support >= min_support:
            L3.append((union_items, support))
    freq_itemsets.extend(L3)

    return freq_itemsets

frequent_itemsets = apriori(df, min_support=0.6)

result_df = pd.DataFrame(frequent_itemsets, columns=["Itemset", "Support"])
print(result_df)
```

```
   Itemset   Support
0  (bread)  0.833333
```

Step 5: Run Apriori

Set `min_support = 0.6` and display the frequent itemsets.

In [5]:
```
frequent_itemsets = apriori(df, min_support=0.6)
print(frequent_itemsets)
```

```
[(frozenset({'bread'}), 0.8333333333333334)]
```

Step 6 Display as a DataFrame

In [6]:
```
import pandas as pd

frequent_itemsets = apriori(df, min_support=0.6)

result_df = pd.DataFrame(frequent_itemsets, columns=["Itemset", "Support"])
print(result_df)
```

```
   Itemset   Support
0  (bread)  0.833333
```

Orange Tool : - >Generate Same Frequent Patterns in Orange tools

Extra : - > Define Apriori Function without itertools

Page No. :

Date :

Lab - 8

A. Generate the frequent pattern Tree for given dataset min-support ≥ 3

1.)

TID	Items
1	E K M N O Y
2	D E K N O Y
3	A E K M
4	C K M O Y
5	C E I K O

Step-1

Item	frequency
A	1
C	2
D	1
E	4
I	1
K	5
M	3
N	2
O	3
U	1
Y	3

{K:5, E:4, O:3, M:3, Y:3}

Step-2

TID	Sorted items
1	K E M O Y
2	K E O Y
3	K E M
4	K M Y
5	K E O

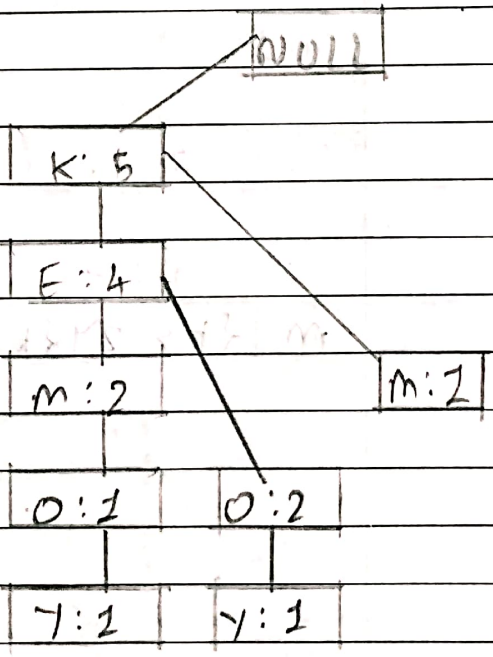
→ K E M O Y

K E O Y

K E M

K M Y

K E O



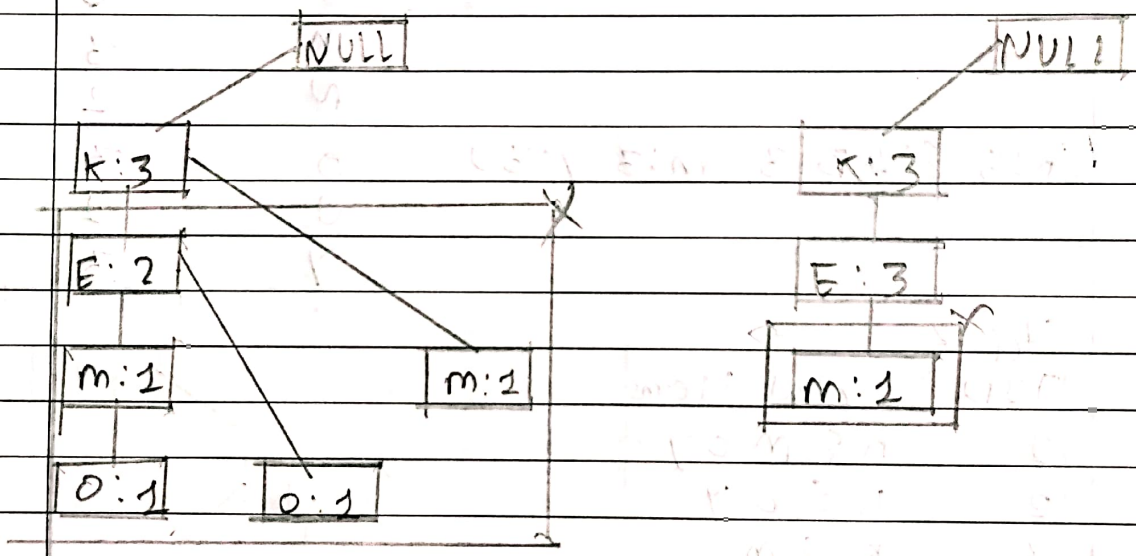
Page No. :

Date :

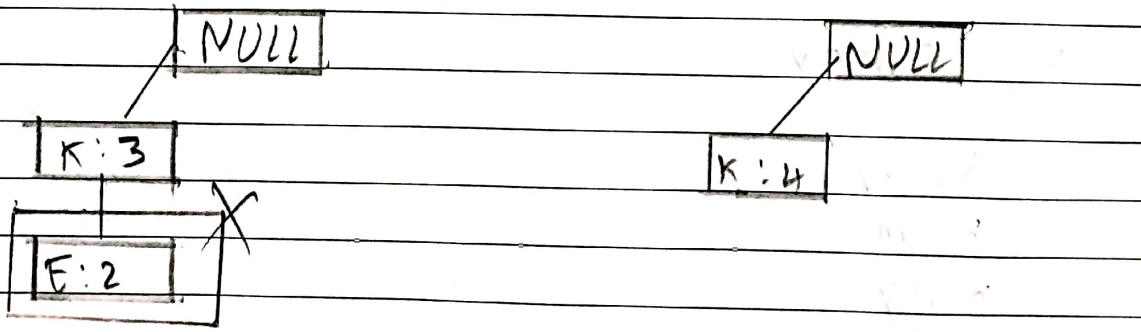
→

Item	Conditional Pattern Base
Y	{K E M O : 1} {K E O : 1} {K M : 1}
O	{K E M : 1} {K E : 2}
M	{K E : 2} {K : 1}
E	{K : 4}
K	-

Y	{K E M O : 1} {K E O : 1} {K M : 1}	{K : 3}
O	{K E M : 1} {K E : 2} {K E : 2}	{K : 3} {E : 3}



M	{K E : 2} {K : 1}	{K : 3}
E	{K : 4}	{K : 4}



Page No. :

Date :

Conditional FP-Tree and frequent pattern generated

item	Conditional Base Pattern	Conditional FP	Frequent Patterns generated
Y	{K,E,M,O:1}{K,E,O:1}{K,M:1}	{K:3}	{K,Y:3}
O	{K,E,M:1}{K,E:2}	{K:3, E:3}	{K,O:3}{E,O:3},{K,E,O:3}
M	{K,E:2}{K:1}	{K:3}	{K,M:3}
E	{K:4}	{K:4}	{K,E:4}
K	-	-	

2.

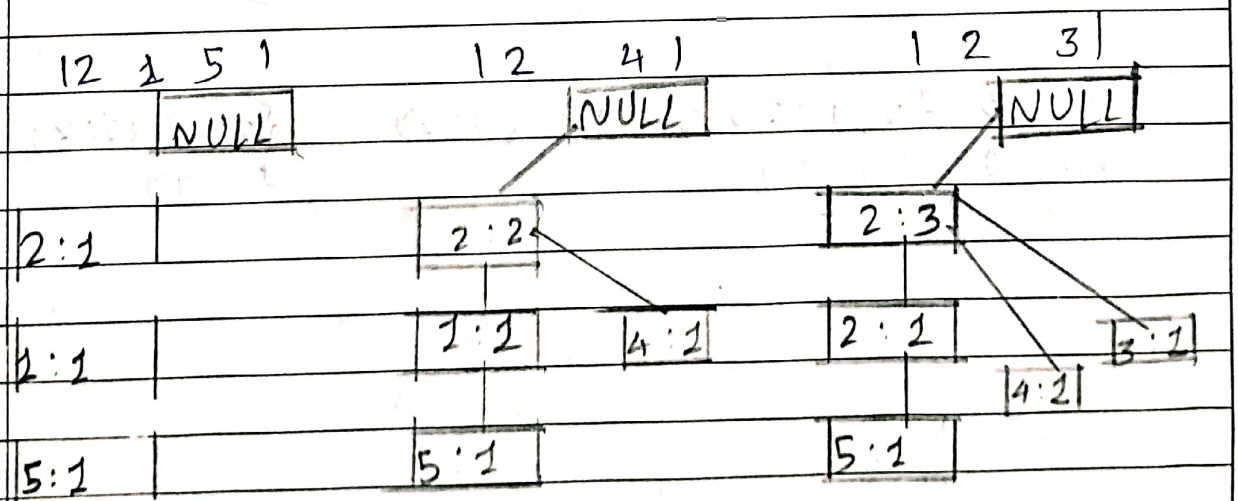
TID	Items
1	1 2 5
2	2 4
3	2 3
4	1 2 4
5	1 3
6	2 3
7	1 3
8	1 2 3 5
9	1 2 3

Step-1

1	6
2	7
3	6
4	2
5	2

Step-2

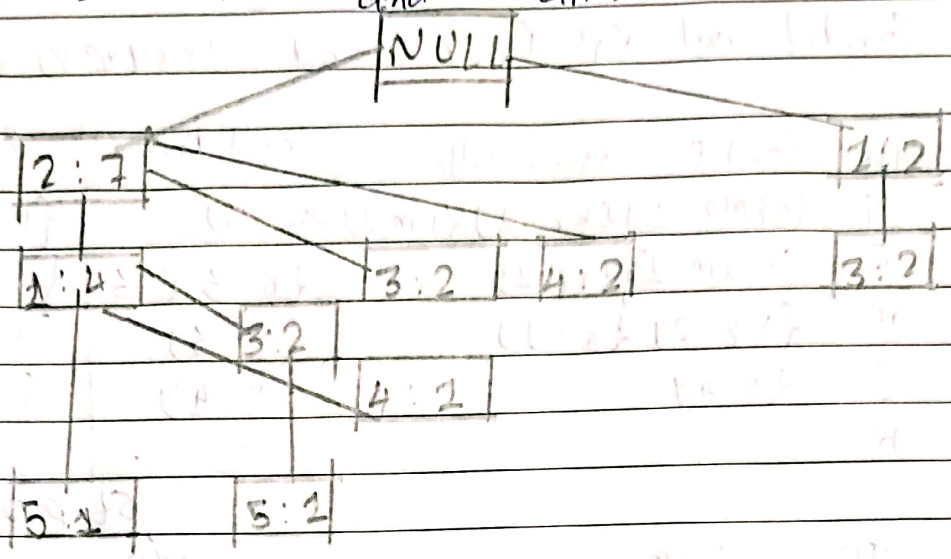
TID	Items
1	2 1 5
2	2 4
3	2 3
4	2 1 4
5	1 3
6	2 3
7	1 3
8	2 1 3 5
9	2 1 3



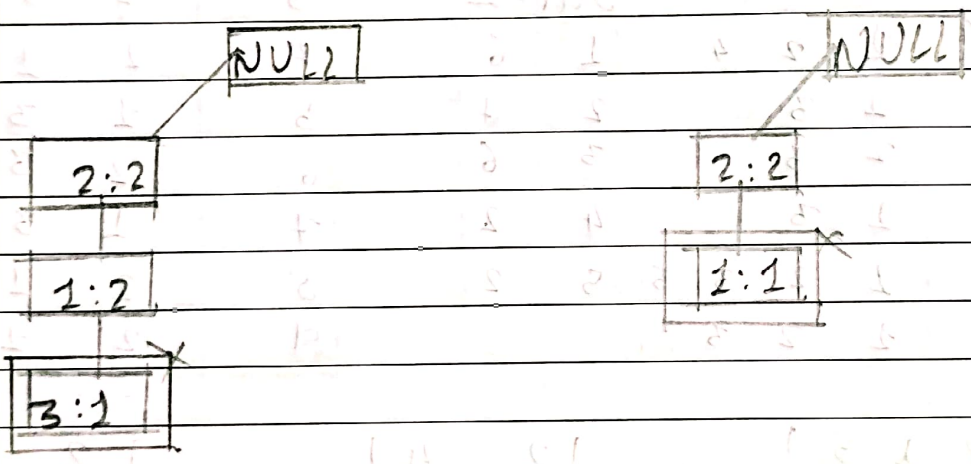
Page No. :

Date :

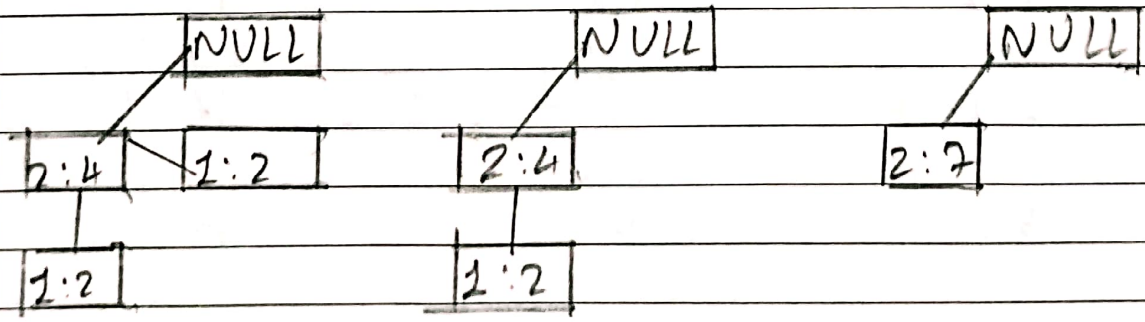
2 1 4 and all...



1.)	5	{2, 1 : 1} {2, 3 : 1}	{2 : 2} {1 : 2}
2.)	4	{2, 1 : 1} {2 : 1}	{2 : 2}



3.)	3	{2, 1 : 2} {2 : 2} {1 : 2}	{2 : 4} {1 : 3}
4.)	1	{2 : 7}	{2 : 7}



Page No. :

Date :

Item	Conditional pattern Base	Conditional FP-Tree	Frequent patterns generated
1	{2:4}	{2:4}	{1,2:4}
2	-	-	-
3	{2:2}, {1:2} {2,1:2}	{2:4} {1:4}	{2,3:2} {1,3:2} {2,1,3:2}
4	{2:1} {2,1:1}	{2:2}	{2,4:2}
5.	{2,1:1} {2,1,3:1}	{2:2}, {1:2}	{2,5:2} {1,5:2}

Darshan UNIVERSITY
योगः कर्मसु कौशलम्

Data Mining

Lab - 1

Name: Smit Maru

Enrollment No: 23010101161

Introduction to Pandas Library Function:

Step-1 Import the pandas Libraries

In [4]:
```
import pandas as pd
```

Step-2 Import the dataset from this:....

In [5]:
```
df = pd.read_csv('titanic.csv')
```

Step-3 Read csv or excel File

In [6]:
```
df = pd.read_csv('titanic.csv')
```

Step-4 Print Data from csv or excel File

In [7]:
```
df
```

Out[7]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	F
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0
...	...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7

891 rows × 12 columns

Step-5 See the First 10 Rows

In [8]:
```
df.head(10)
```

Out[8]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708

Step-6 See the Last 10 Rows

In [9]:
```
df.tail(10)
```

Out[9]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	
881	882	0	3	Markun, Mr. Johann	male	33.0	0	0	349257	7.
882	883	0	3	Dahlberg, Miss. Gerda Ulrika	female	22.0	0	0	7552	10.
883	884	0	2	Banfield, Mr. Frederick James	male	28.0	0	0	C.A./SOTON 34068	10.
884	885	0	3	Sutehall, Mr. Henry Jr	male	25.0	0	0	SOTON/OQ 392076	7.
885	886	0	3	Rice, Mrs. William (Margaret Norton)	female	39.0	0	5	382652	29.
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.

Step-7 Data type of each columns

In [10]:
```
df.dtypes
```

Out[10]:
```
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Cabin           object
Embarked        object
dtype: object
```

Step-8 Display Summary Information

In [11]:
```
df.info
```

Out[11]:
```
<bound method DataFrame.info of      PassengerId  Survived  Pclass  \
0              1         0       3
1              2         1       1
2              3         1       3
3              4         1       1
4              5         0       3
..           ...       ...     ...
886          887         0       2
887          888         1       1
888          889         0       3
889          890         1       1
890          891         0       3

                                                  Name     Sex   Age  SibSp  \
0                              Braund, Mr. Owen Harris    male  22.0      1
1    Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
2                               Heikkinen, Miss. Laina  female  26.0      0
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                             Allen, Mr. William Henry    male  35.0      0
..                                                 ...     ...   ...    ...
886                             Montvila, Rev. Juozas    male  27.0      0
887                      Graham, Miss. Margaret Edith  female  19.0      0
888          Johnston, Miss. Catherine Helen "Carrie"  female   NaN      1
889                             Behr, Mr. Karl Howell    male  26.0      0
890                               Dooley, Mr. Patrick    male  32.0      0

     Parch            Ticket     Fare Cabin Embarked
0        0         A/5 21171   7.2500   NaN        S
1        0          PC 17599  71.2833   C85        C
2        0  STON/O2. 3101282   7.9250   NaN        S
3        0            113803  53.1000  C123        S
4        0            373450   8.0500   NaN        S
..     ...               ...      ...   ...      ...
886      0            211536  13.0000   NaN        S
887      0            112053  30.0000   B42        S
888      2        W./C. 6607  23.4500   NaN        S
889      0            111369  30.0000  C148        C
890      0            370376   7.7500   NaN        Q

[891 rows x 12 columns]>
```

Step-9 Access a specific column

In [12]:
```
df['Name']
```

Out[12]:
```
0                                Braund, Mr. Owen Harris
1      Cumings, Mrs. John Bradley (Florence Briggs Th...
2                                 Heikkinen, Miss. Laina
3           Futrelle, Mrs. Jacques Heath (Lily May Peel)
4                               Allen, Mr. William Henry
                              ...
886                                Montvila, Rev. Juozas
887                         Graham, Miss. Margaret Edith
888             Johnston, Miss. Catherine Helen "Carrie"
889                                Behr, Mr. Karl Howell
890                                  Dooley, Mr. Patrick
Name: Name, Length: 891, dtype: object
```

Step-10 Access rows by their integer location

In [13]:
```
print(df.iloc[1])
```

```
PassengerId                                                     2
Survived                                                        1
Pclass                                                          1
Name           Cumings, Mrs. John Bradley (Florence Briggs Th...
Sex                                                       female
Age                                                         38.0
SibSp                                                          1
Parch                                                          0
Ticket                                                  PC 17599
Fare                                                     71.2833
Cabin                                                        C85
Embarked                                                       C
Name: 1, dtype: object
```

Step-11 Delete a specific Column

In [14]:
```
df = df.drop('Cabin', axis=1)
```

Step-12 Create a new Column

In [15]:
```
df['Status'] = 'Active'
df
```

Out[15]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	F
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0
...	...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7

891 rows × 12 columns

Step-13 Perform Condition Selection on DataFrame

In [16]:
```
male = df[df['Sex'] == 'male']
print(male)
```

```
     PassengerId  Survived  Pclass                              Name   Sex  \
0              1         0       3           Braund, Mr. Owen Harris  male
4              5         0       3         Allen, Mr. William Henry  male
5              6         0       3                 Moran, Mr. James  male
6              7         0       1          McCarthy, Mr. Timothy J  male
7              8         0       3   Palsson, Master. Gosta Leonard  male
..           ...       ...     ...                               ...   ...
883          884         0       2    Banfield, Mr. Frederick James  male
884          885         0       3           Sutehall, Mr. Henry Jr  male
886          887         0       2            Montvila, Rev. Juozas  male
889          890         1       1             Behr, Mr. Karl Howell  male
890          891         0       3               Dooley, Mr. Patrick  male

      Age  SibSp  Parch            Ticket     Fare Embarked  Status
0    22.0      1      0         A/5 21171   7.2500        S  Active
4    35.0      0      0            373450   8.0500        S  Active
5     NaN      0      0            330877   8.4583        Q  Active
6    54.0      0      0             17463  51.8625        S  Active
7     2.0      3      1            349909  21.0750        S  Active
..    ...    ...    ...               ...      ...      ...     ...
883  28.0      0      0  C.A./SOTON 34068  10.5000        S  Active
884  25.0      0      0   SOTON/OQ 392076   7.0500        S  Active
886  27.0      0      0            211536  13.0000        S  Active
889  26.0      0      0            111369  30.0000        C  Active
890  32.0      0      0            370376   7.7500        Q  Active

[577 rows x 12 columns]
```

Step-14 Compute the sum of value

In [17]:
```
total_fare = df['Fare'].sum()
print("Total Fare:", total_fare)
```

```
Total Fare: 28693.9493
```

Step-15 Compute the mean of value

In [18]:
```
mean_age = df['Age'].mean()
print("Mean Age:", mean_age)
```

```
Mean Age: 29.69911764705882
```

Step-16 Count non-null value (column)

In [19]:
```
print(df['Age'].count())
```

```
714
```

Step-17 Find Minimun or Maximum values

In [20]:
```
print("Min Age:", df['Age'].min())
print("Max Age:", df['Age'].max())
```

```
Min Age: 0.42
Max Age: 80.0
```

Darshan UNIVERSITY
योगः कर्मसु कौशलम्

Data Mining

Lab - 2

Name: Smit Maru

Enrollment No: 23010101161

Numpy & Perform Data Exploration with Pandas

Numpy

1. NumPy (Numerical Python) is a powerful open-source library in Python used for numerical and scientific computing.
2. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on them efficiently.
3. NumPy is highly optimized and written in C, making it much faster than using regular Python lists for numerical operations.
4. It serves as the foundation for many other Python libraries in data science and machine learning, like pandas, TensorFlow, and scikit-learn.
5. With features like broadcasting, vectorization, and integration with C/C++ code, NumPy allows for cleaner and faster code in numerical computations.

Step 1. Import the Numpy library

In [1]:
```
import numpy as np
```

Step 2. Create a 1D array of numbers

In [2]:
```
arr_1 = np.array([1, 2, 3, 4, 5])
print(arr_1)
```

```
[1 2 3 4 5]
```

In [3]:
```
arr_2 = np.arange(2, 11)
print(arr_2)
```

```
[ 2  3  4  5  6  7  8  9 10]
```

Step 3. Reshape 1D to 2D Array

In [4]:
```
arr_2d = np.arange(12).reshape(3, 4)
print(arr_2d)
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
```

In [5]:
```
print(arr_2d.size)
print(arr_2d.dtype)
print(arr_2d.ndim) #dimention
```

```
12
int32
2
```

Step 4. Create a Linspace array

In [6]:
```
arr_ls = np.linspace(0, 5, 20)
print(arr_ls)
```

```
[0.         0.26315789 0.52631579 0.78947368 1.05263158 1.31578947
 1.57894737 1.84210526 2.10526316 2.36842105 2.63157895 2.89473684
 3.15789474 3.42105263 3.68421053 3.94736842 4.21052632 4.47368421
 4.73684211 5.        ]
```

Step 5. Create a Random Numbered Array

In [7]:
```
arr_r = np.random.rand(5)
print(arr_r)
```

```
[0.85199473 0.40958552 0.59154038 0.84407658 0.75496985]
```

In [8]:
```
arr_2_r = np.random.rand(2, 5)
print(arr_2_r)
```

```
[[0.93932318 0.76517792 0.3279605  0.04224219 0.58414288]
 [0.08086957 0.47539208 0.56380551 0.12650827 0.98973303]]
```

Step 6. Create a Random Integer Array

In [9]:
```
arr_r_int = np.random.randint(1, 50, size = 4)
print(arr_r_int)
```

```
[16 35 10  6]
```

In [ ]:

Step 7. Create a 1D Array and get Max,Min,ArgMax,ArgMin

In [10]:
```
arr_a = np.random.randint(1, 50, size = 15)
print(arr_a)
```

```
[25 36 23 26 29 42 26  2 34 31 38 44 46 23 39]
```

In [11]:
```
print(arr_a.max())
```

```
46
```

In [12]:
```
print(arr_a.min())
```

```
2
```

In [13]:
```
print(arr_a.argmax())
```

```
12
```

In [14]:
```
print(arr_a.argmin())
```

```
7
```

Step 8. Indexing in 1D Array

In [15]:
```
a = np.arange(10)
```

In [16]:
```
print(a[0])
```

```
0
```

Step 9. Indexing in 2D Array

In [17]:
```
a2d = np.arange(12).reshape(3, 4)
```

In [18]:
```
print(a2d[0])
```

```
[0 1 2 3]
```

In [19]:
```
print(a2d[0, 2])
```

```
2
```

In [20]:
```
print(a2d[0][2])
```

```
2
```

Step 10. Conditional Selection

In [21]:
```
print(a2d[a2d > 4])
```

```
[ 5  6  7  8  9 10 11]
```

In [22]:
```
print(a2d[a2d % 2 == 0])
```

```
[ 0  2  4  6  8 10]
```

🔥 You did it! 10 exercises down — you're on fire! 🔥

Pandas

Step 1. Import the necessary libraries

In [23]:
```
#pip install pandas-->(already installed, biji vaar karvi nahi, karnaar ne sajaa th
```

In [24]:
```
import pandas as pd
```

Step 2. Import the dataset from this address.

In [25]:
```
users = pd.read_csv('https://raw.githubusercontent.com/justmarkham/DAT8/master/data
```

Step 3. Assign it to a variable called users and use the 'user_id' as index

In [26]:
```
users = pd.read_csv('https://raw.githubusercontent.com/justmarkham/DAT8/master/data
```

In [27]:
```
print(users, sep='|')
```

```
     user_id  age gender     occupation zip_code
0          1   24      M     technician    85711
1          2   53      F          other    94043
2          3   23      M         writer    32067
3          4   24      M     technician    43537
4          5   33      F          other    15213
..       ...  ...    ...            ...      ...
938      939   26      F        student    33319
939      940   32      M  administrator    02215
940      941   20      M        student    97229
941      942   48      F      librarian    78209
942      943   22      M        student    77841

[943 rows x 5 columns]
```

Step 4. See the first 25 entries

In [28]:
```
users.head(25)
```

Out[28]:

	user_id	age	gender	occupation	zip_code
0	1	24	M	technician	85711
1	2	53	F	other	94043
2	3	23	M	writer	32067
3	4	24	M	technician	43537
4	5	33	F	other	15213
5	6	42	M	executive	98101
6	7	57	M	administrator	91344
7	8	36	M	administrator	05201
8	9	29	M	student	01002
9	10	53	M	lawyer	90703
10	11	39	F	other	30329
11	12	28	F	other	06405
12	13	47	M	educator	29206
13	14	45	M	scientist	55106
14	15	49	F	educator	97301
15	16	21	M	entertainment	10309
16	17	30	M	programmer	06355
17	18	35	F	other	37212
18	19	40	M	librarian	02138
19	20	42	F	homemaker	95660
20	21	26	M	writer	30068
21	22	25	M	writer	40206
22	23	30	F	artist	48197
23	24	21	F	artist	94533
24	25	39	M	engineer	55107

Step 5. See the last 10 entries

In [29]:
```
users.tail(10)
```

Out[29]:

	user_id	age	gender	occupation	zip_code
933	934	61	M	engineer	22902
934	935	42	M	doctor	66221
935	936	24	M	other	32789
936	937	48	M	educator	98072
937	938	38	F	technician	55038
938	939	26	F	student	33319
939	940	32	M	administrator	02215
940	941	20	M	student	97229
941	942	48	F	librarian	78209
942	943	22	M	student	77841

Step 6. What is the number of observations in the dataset?

In [30]:
```
print(users.shape[0])
```

```
943
```

Step 7. What is the number of columns in the dataset?

In [31]:
```
print(users.shape[1])
```

```
5
```

Step 8. Print the name of all the columns.

In [32]:
```
print(users.columns.tolist())
```

```
['user_id', 'age', 'gender', 'occupation', 'zip_code']
```

Step 9. How is the dataset indexed?

In [33]:
```
print(users.index)
```

```
RangeIndex(start=0, stop=943, step=1)
```

Step 10. What is the data type of each column?

In [34]:
```
print(users.dtypes)
```

```
user_id        int64
age            int64
gender        object
occupation    object
zip_code      object
dtype: object
```

Step 11. Print only the occupation column

In [35]:
```
print(users['occupation'])
```

```
0         technician
1              other
2             writer
3         technician
4              other
           ...
938          student
939    administrator
940          student
941        librarian
942          student
Name: occupation, Length: 943, dtype: object
```

Step 12. How many different occupations are in this dataset?

In [36]:
```
print(users.columns)
```

```
Index(['user_id', 'age', 'gender', 'occupation', 'zip_code'], dtype='object')
```

Step 13. What is the most frequent occupation?

In [37]:
```
print(users['occupation'].value_counts().idxmax())
```

```
student
```

Step 14. Summarize the DataFrame.

In [38]:
```
print(users.describe())
```

```
          user_id         age
count  943.000000  943.000000
mean   472.000000   34.051962
std    272.364951   12.192740
min      1.000000    7.000000
25%    236.500000   25.000000
50%    472.000000   31.000000
75%    707.500000   43.000000
max    943.000000   73.000000
```

Step 15. Summarize all the columns

In [39]:
```
print(users.describe(include='all'))
```

	user_id	age	gender	occupation	zip_code
count	943.000000	943.000000	943	943	943
unique	NaN	NaN	2	21	795
top	NaN	NaN	M	student	55414
freq	NaN	NaN	670	196	9
mean	472.000000	34.051962	NaN	NaN	NaN
std	272.364951	12.192740	NaN	NaN	NaN
min	1.000000	7.000000	NaN	NaN	NaN
25%	236.500000	25.000000	NaN	NaN	NaN
50%	472.000000	31.000000	NaN	NaN	NaN
75%	707.500000	43.000000	NaN	NaN	NaN
max	943.000000	73.000000	NaN	NaN	NaN

Step 16. Summarize only the occupation column

In [40]:
```
print(users['occupation'].describe())
```

```
count         943
unique         21
top       student
freq          196
Name: occupation, dtype: object
```

Step 17. What is the mean age of users?

In [41]:
```
print(users['age'].mean())
```

```
34.05196182396607
```

Step 18. What is the age with least occurrence?

In [42]:
```
print(users['age'].value_counts().idxmin())
```

```
7
```

You're not just learning, you're mastering it. Keep aiming higher! 🚀

In [43]:
```
print("Yes!....")
```

```
Yes!....
```

Darshan UNIVERSITY
योगः कर्मसु कौशलम्

Data Mining

Lab - 3

Name: Smit Maru

Enrollment No: 23010101161

1) First, you need to read the titanic dataset from local disk and display first five records

In [1]:
```
import pandas as pd
```

In [2]:
```
import numpy as np
```

In [3]:
```
titanic = pd.read_csv('titanic.csv')
titanic.head(5)
```

Out[3]:

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

2) Identify Nominal, Ordinal, Binary and Numeric attributes from data sets and display all values.

In [4]:
```
Nominal = ['Name','Sex','Embarked']
Binary = ['Survived','SibSp']
Ordinal = ['Pclass']
Numeric = ['Age','Parch','Fare']
```

3) Identify symmetric and asymmetric binary attributes from data sets and display all values.

In [6]:
```
print("Count : ")
print(titanic['Servived'].value_count())
```

```
Count :
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File ~\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3805, in Index.get_loc(self, key)
   3804 try:
-> 3805     return self._engine.get_loc(casted_key)
   3806 except KeyError as err:

File index.pyx:167, in pandas._libs.index.IndexEngine.get_loc()

File index.pyx:196, in pandas._libs.index.IndexEngine.get_loc()

File pandas\\_libs\\hashtable_class_helper.pxi:7081, in pandas._libs.hashtable.PyObjectHashTable.get_item()

File pandas\\_libs\\hashtable_class_helper.pxi:7089, in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'Servived'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
Cell In[6], line 2
      1 print("Count : ")
----> 2 print(titanic['Servived'])

File ~\anaconda3\Lib\site-packages\pandas\core\frame.py:4102, in DataFrame.__getitem__(self, key)
   4100 if self.columns.nlevels > 1:
   4101     return self._getitem_multilevel(key)
-> 4102 indexer = self.columns.get_loc(key)
   4103 if is_integer(indexer):
   4104     indexer = [indexer]

File ~\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3812, in Index.get_loc(self, key)
   3807     if isinstance(casted_key, slice) or (
   3808         isinstance(casted_key, abc.Iterable)
   3809         and any(isinstance(x, slice) for x in casted_key)
   3810     ):
   3811         raise InvalidIndexError(key)
-> 3812     raise KeyError(key) from err
   3813 except TypeError:
   3814     # If we have a listlike key, _check_indexing_error will raise
   3815     #  InvalidIndexError. Otherwise we fall through and re-raise
   3816     #  the TypeError.
   3817     self._check_indexing_error(key)

KeyError: 'Servived'
```

4) For each quantitative attribute, calculate its average, standard deviation, minimum, mode, range and maximum values.

In [7]:
```
Quantative = ['PassengerId','Survived','Pclass','Age','SibSp','Parch','Fare']

for col in Quantative:
    print(':::',col,':::')
    print('Mean:',titanic[col].mean())
    print('SD:',titanic[col].std())
    print('Min:',titanic[col].min())
    print('Max:',titanic[col].max())
    print('Mod:',titanic[col].mode()[0])
    print('Range:',titanic[col].max() - titanic[col].min())
```

```
::: PassengerId :::
Mean: 446.0
SD: 257.3538420152301
Min: 1
Max: 891
Mod: 1
Range: 890
::: Survived :::
Mean: 0.3838383838383838
SD: 0.4865924542648585
Min: 0
Max: 1
Mod: 0
Range: 1
::: Pclass :::
Mean: 2.308641975308642
SD: 0.8360712409770513
Min: 1
Max: 3
Mod: 3
Range: 2
::: Age :::
Mean: 29.69911764705882
SD: 14.526497332334044
Min: 0.42
Max: 80.0
Mod: 24.0
Range: 79.58
::: SibSp :::
Mean: 0.5230078563411896
SD: 1.1027434322934275
Min: 0
Max: 8
Mod: 0
Range: 8
::: Parch :::
Mean: 0.38159371492704824
SD: 0.8060572211299559
Min: 0
Max: 6
Mod: 0
Range: 6
::: Fare :::
Mean: 32.204207968574636
SD: 49.693428597180905
Min: 0.0
Max: 512.3292
Mod: 8.05
Range: 512.3292
```

6) For the qualitative attribute (class), count the frequency for each of its distinct values.

In [9]:
```
print("Passenger Class frequency : ")
print(titanic['Pclass'].value_counts())
```

```
Passenger Class frequency :
Pclass
3    491
1    216
2    184
Name: count, dtype: int64
```

7) It is also possible to display the summary for all the attributes simultaneously in a table using the describe() function. If an attribute is quantitative, it will display its mean, standard deviation and various quantiles (including minimum, median, and maximum) values. If an attribute is qualitative, it will display its number of unique values and the top (most frequent) values.

In [10]:
```
print("Numeric Summary")
print(titanic.describe())
print("Catagorial Summary")
print(titanic.describe(include=['object']))
```

```
Numeric Summary
       PassengerId    Survived      Pclass         Age       SibSp  \
count   891.000000  891.000000  891.000000  714.000000  891.000000
mean    446.000000    0.383838    2.308642   29.699118    0.523008
std     257.353842    0.486592    0.836071   14.526497    1.102743
min       1.000000    0.000000    1.000000    0.420000    0.000000
25%     223.500000    0.000000    2.000000   20.125000    0.000000
50%     446.000000    0.000000    3.000000   28.000000    0.000000
75%     668.500000    1.000000    3.000000   38.000000    1.000000
max     891.000000    1.000000    3.000000   80.000000    8.000000

            Parch        Fare
count  891.000000  891.000000
mean     0.381594   32.204208
std      0.806057   49.693429
min      0.000000    0.000000
25%      0.000000    7.910400
50%      0.000000   14.454200
75%      0.000000   31.000000
max      6.000000  512.329200
Catagorial Summary
                           Name   Sex  Ticket    Cabin Embarked
count                       891   891     891      204      889
unique                      891     2     681      147        3
top     Braund, Mr. Owen Harris  male  347082  B96 B98        S
freq                          1   577       7        4      644
```

8) For multivariate statistics, you can compute the covariance and correlation between pairs of attributes.

In [11]:
```
print(titanic.cov(numeric_only=True))
```

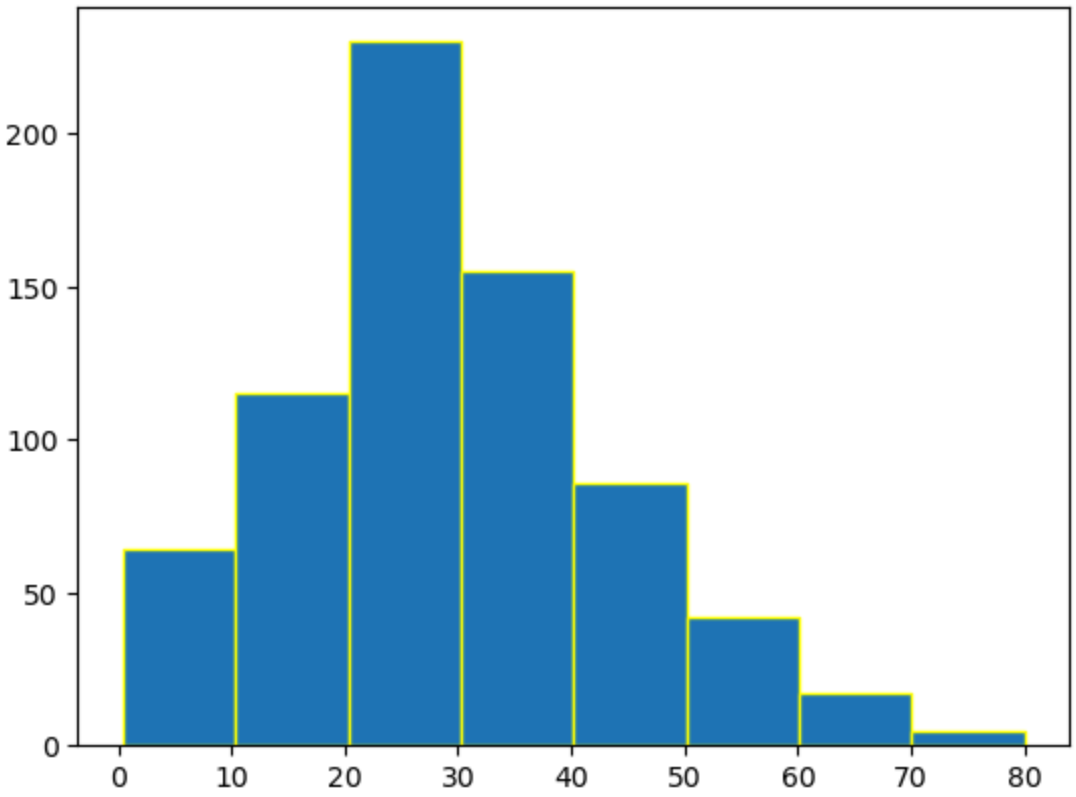
```
             PassengerId  Survived     Pclass         Age      SibSp  \
PassengerId  66231.000000 -0.626966  -7.561798  138.696504 -16.325843
Survived        -0.626966  0.236772  -0.137703   -0.551296  -0.018954
Pclass          -7.561798 -0.137703   0.699015   -4.496004   0.076599
Age            138.696504 -0.551296  -4.496004  211.019125  -4.163334
SibSp          -16.325843 -0.018954   0.076599   -4.163334   1.216043
Parch           -0.342697  0.032017   0.012429   -2.344191   0.368739
Fare           161.883369  6.221787 -22.830196   73.849030   8.748734

                Parch         Fare
PassengerId -0.342697   161.883369
Survived     0.032017     6.221787
Pclass       0.012429   -22.830196
Age         -2.344191    73.849030
SibSp        0.368739     8.748734
Parch        0.649728     8.661052
Fare         8.661052  2469.436846
```

In [12]:
```
print(titanic.corr(numeric_only=True))
```

```
             PassengerId  Survived    Pclass       Age     SibSp     Parch  \
PassengerId     1.000000 -0.005007 -0.035144  0.036847 -0.057527 -0.001652
Survived       -0.005007  1.000000 -0.338481 -0.077221 -0.035322  0.081629
Pclass         -0.035144 -0.338481  1.000000 -0.369226  0.083081  0.018443
Age             0.036847 -0.077221 -0.369226  1.000000 -0.308247 -0.189119
SibSp          -0.057527 -0.035322  0.083081 -0.308247  1.000000  0.414838
Parch          -0.001652  0.081629  0.018443 -0.189119  0.414838  1.000000
Fare            0.012658  0.257307 -0.549500  0.096067  0.159651  0.216225

                 Fare
PassengerId  0.012658
Survived     0.257307
Pclass      -0.549500
Age          0.096067
SibSp        0.159651
Parch        0.216225
Fare         1.000000
```

9) Display the histogram for Age attribute by discretizing it into 8 separate bins and counting the frequency for each bin.

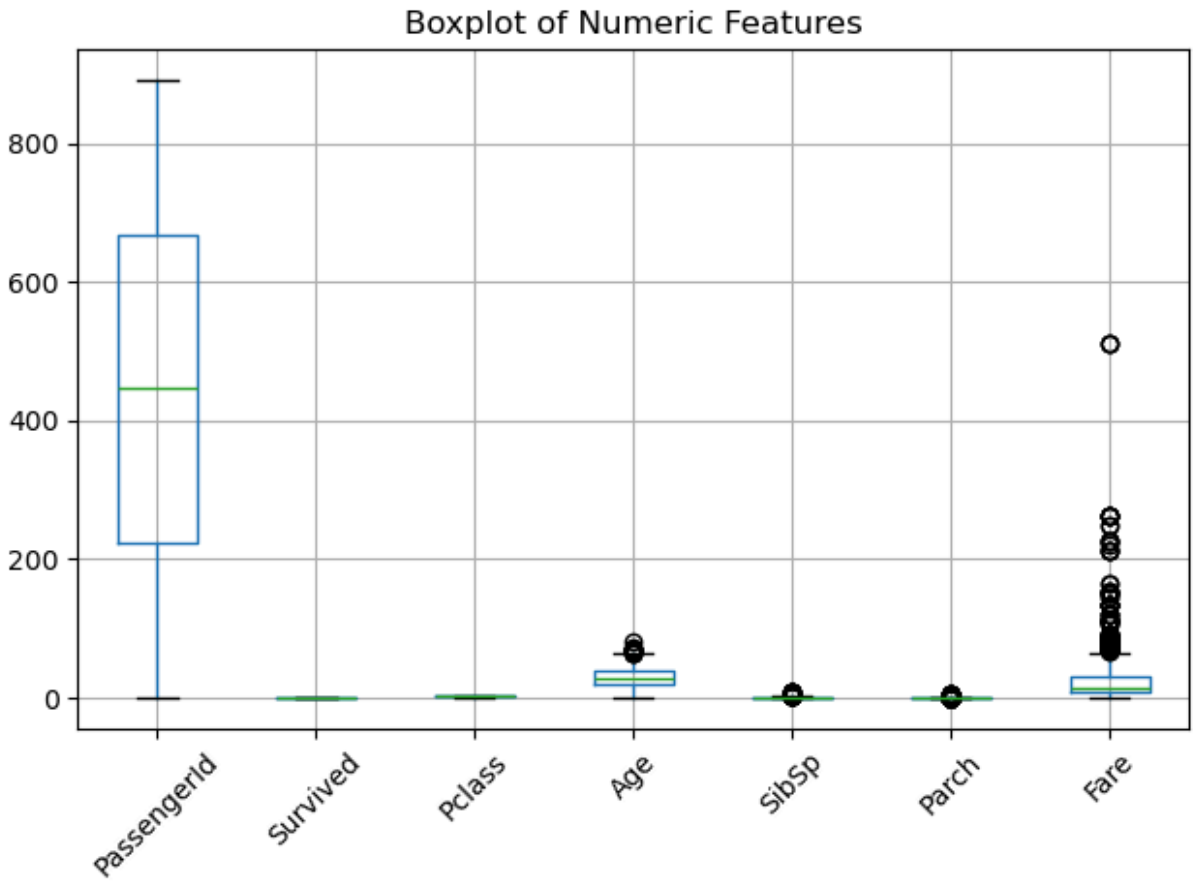
In [13]:
```
import matplotlib.pyplot as plt
plt.hist(titanic['Age'],bins=8,edgecolor='yellow')
plt.show()
```



In [ ]:

10) A boxplot can also be used to show the distribution of values for each attribute.

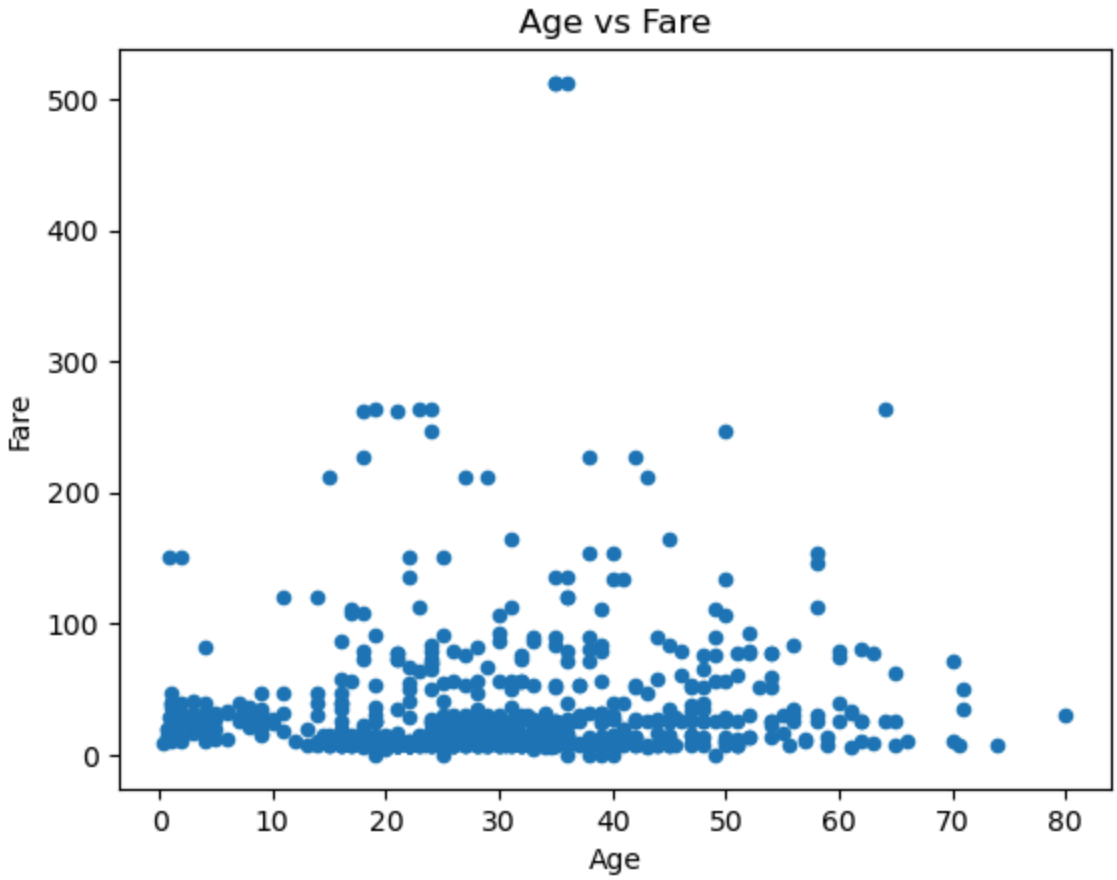
In [16]:
```
titanic.boxplot()
plt.title("Boxplot of Numeric Features")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

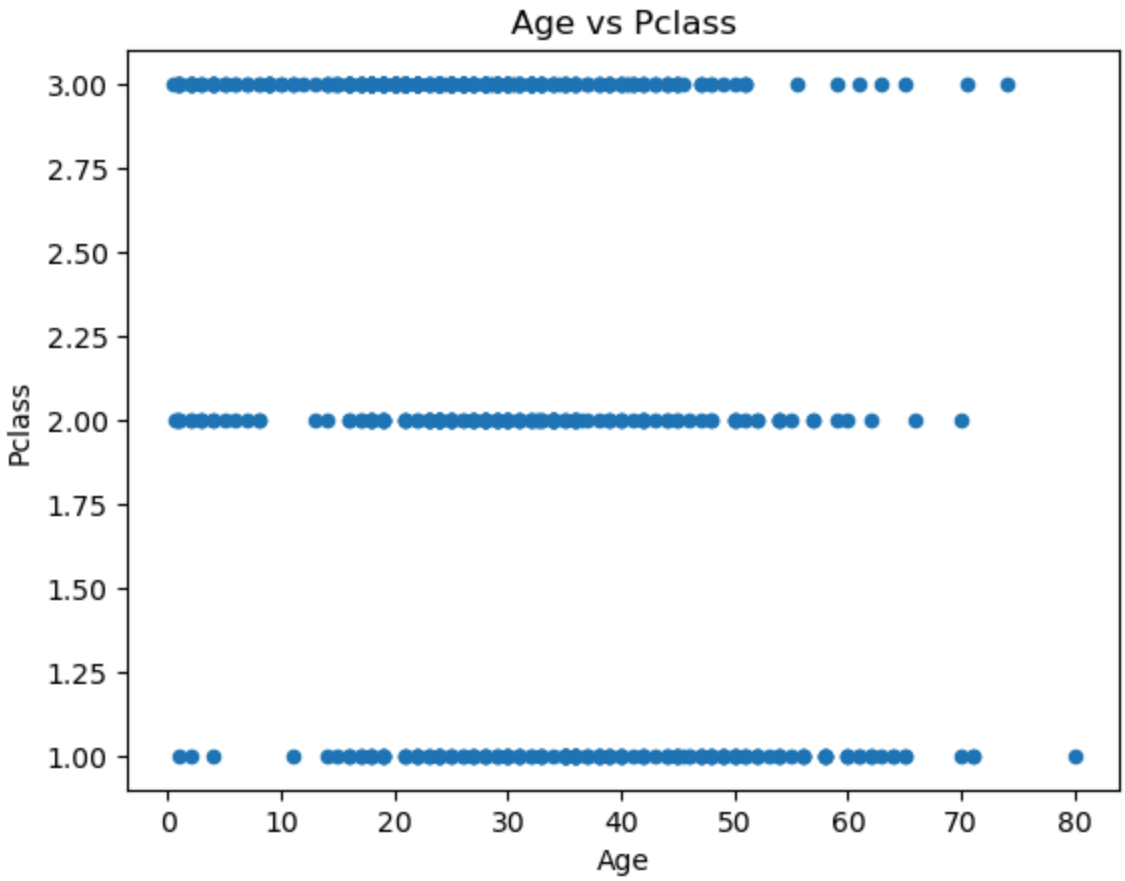


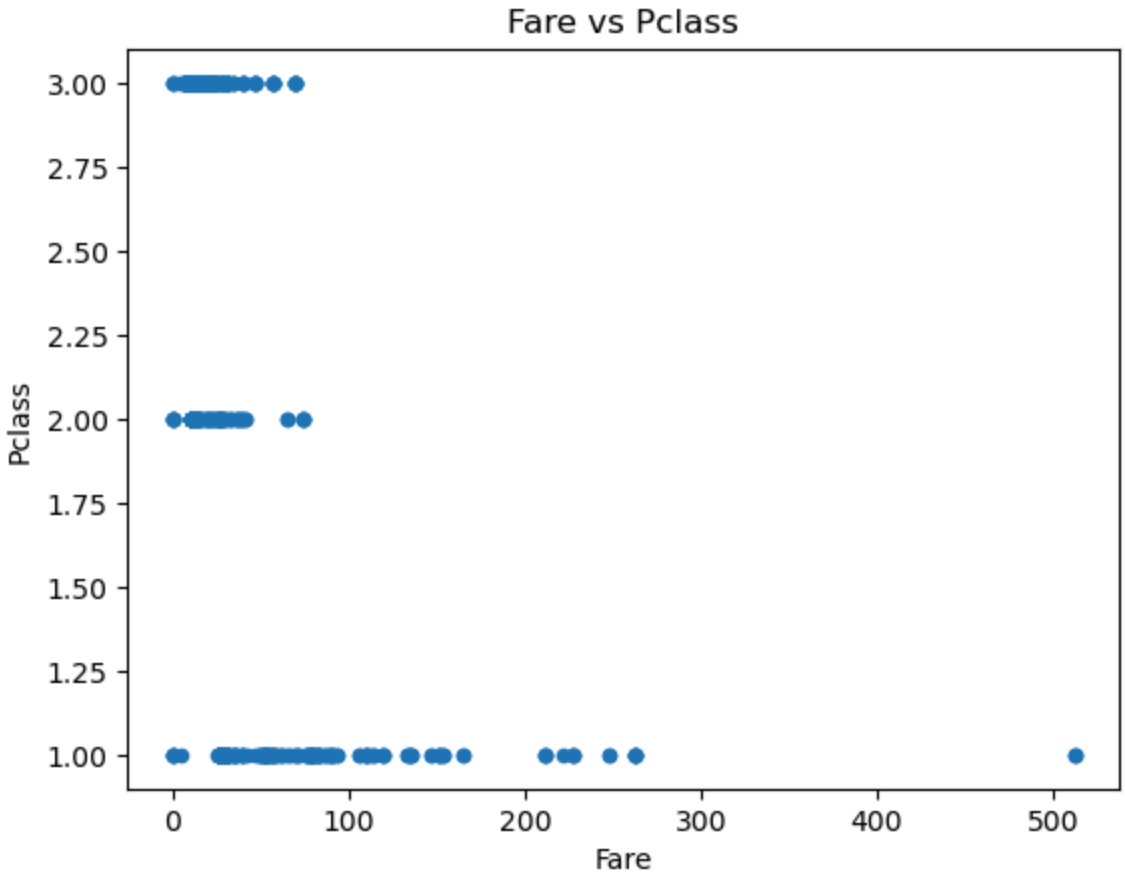
11) Display scatter plot for any 5 pair of attributes , we can use a scatter plot to visualize their joint distribution.

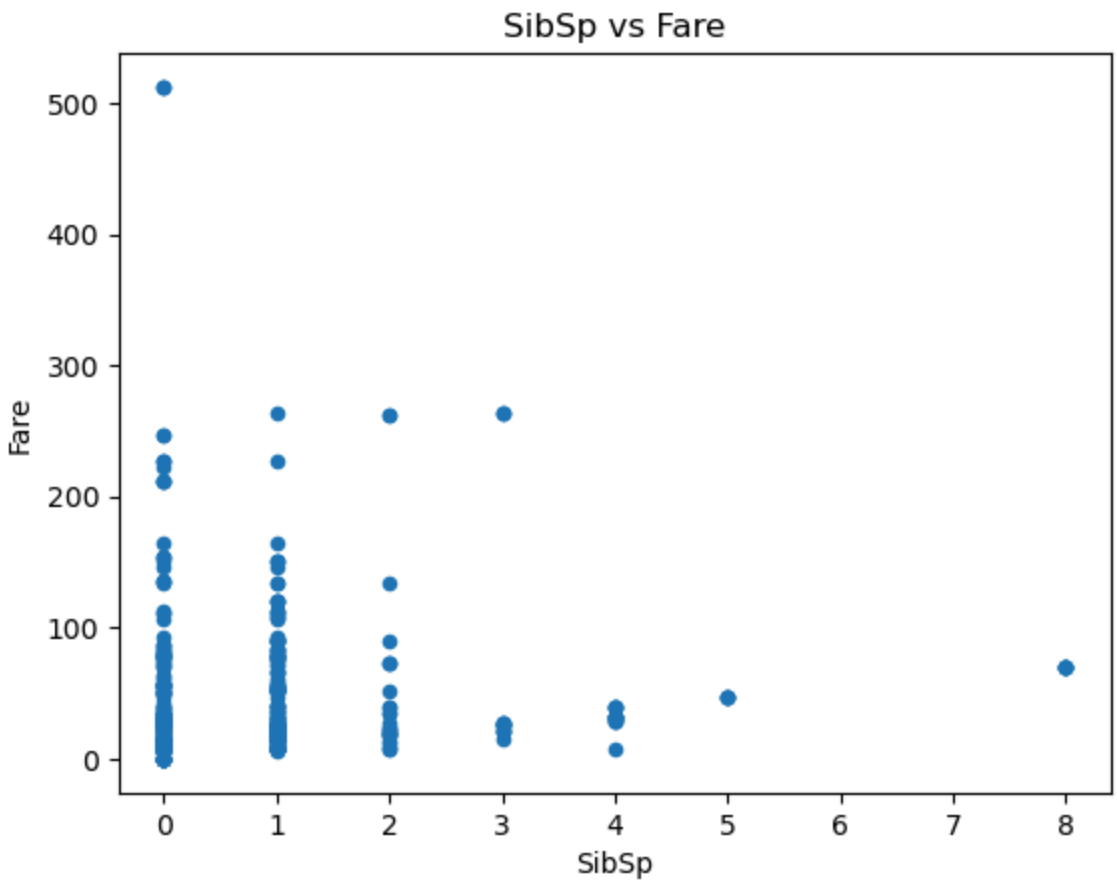
In [17]:
```
titanic.plot.scatter(x='Age', y='Fare', title='Age vs Fare')

titanic.plot.scatter(x='Age', y='Pclass', title='Age vs Pclass')

titanic.plot.scatter(x='Fare', y='Pclass', title='Fare vs Pclass')

titanic.plot.scatter(x='SibSp', y='Fare', title='SibSp vs Fare')

titanic.plot.scatter(x='Parch', y='Age', title='Parch vs Age')
```

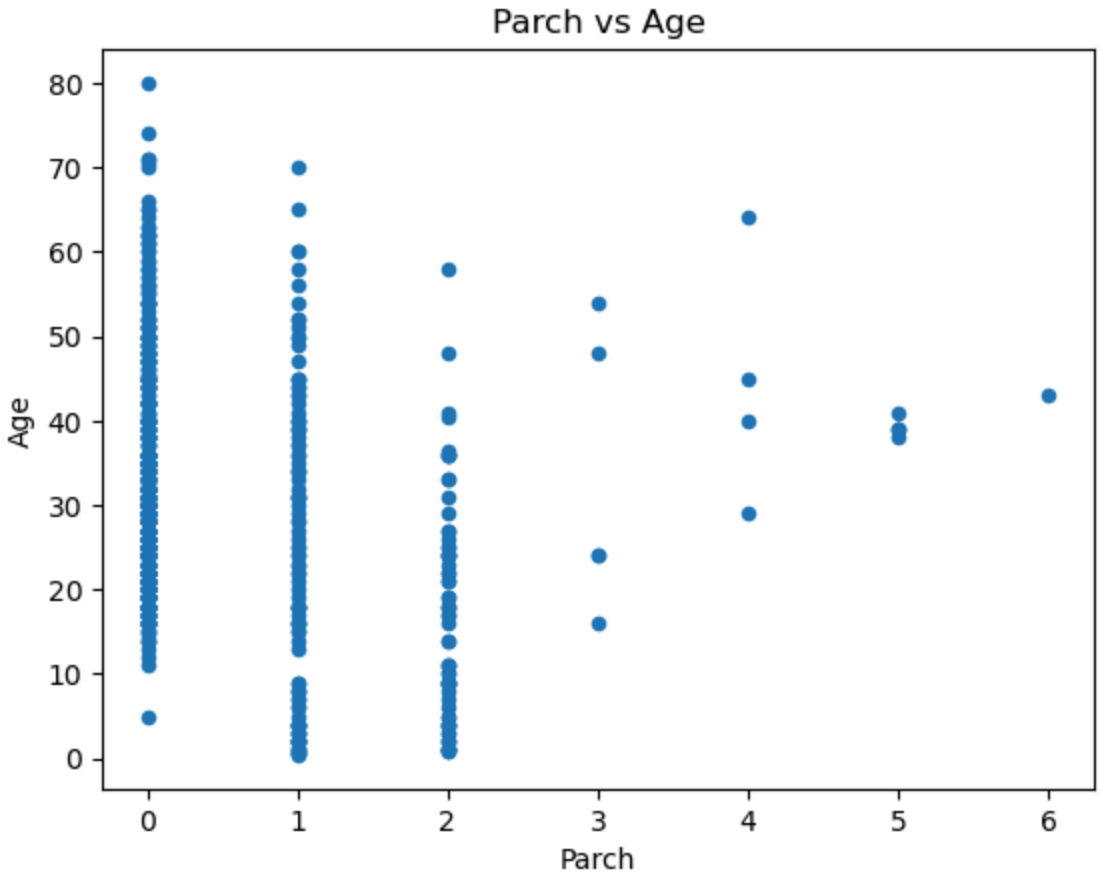
Out[17]:
```
<Axes: title={'center': 'Parch vs Age'}, xlabel='Parch', ylabel='Age'>
```


Age vs Fare
Fare
Age
0
100
200
300
400
500
0
10
20
30
40
50
60
70
80


Age vs Pclass
Pclass
Age
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
0
10
20
30
40
50
60
70
80


Fare vs Pclass
Pclass
Fare
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
0
100
200
300
400
500


SibSp vs Fare
Fare
SibSp
500
400
300
200
100
0
0
1
2
3
4
5
6
7
8


Parch vs Age
Age
Parch
0
10
20
30
40
50
60
70
80
0
1
2
3
4
5
6

Data Mining

Lab - 4

Name: Smit Maru

Enrollment No: 23010101161

Step 1. Import the necessary libraries

In [2]:
```
import pandas as pd
```

Step 2. Import the dataset from this address.

In [3]:
```
url = "https://raw.githubusercontent.com/justmarkham/DAT8/master/data/chipotle.tsv"
data = pd.read_csv(url , sep="\t")
print(data)
```

```
      order_id  quantity                              item_name  \
0            1         1           Chips and Fresh Tomato Salsa
1            1         1                                   Izze
2            1         1                       Nantucket Nectar
3            1         1  Chips and Tomatillo-Green Chili Salsa
4            2         2                           Chicken Bowl
...        ...       ...                                    ...
4617      1833         1                       Steak Burrito
4618      1833         1                       Steak Burrito
4619      1834         1                  Chicken Salad Bowl
4620      1834         1                  Chicken Salad Bowl
4621      1834         1                  Chicken Salad Bowl

                                choice_description item_price
0                                              NaN     $2.39
1                                    [Clementine]     $3.39
2                                         [Apple]     $3.39
3                                              NaN     $2.39
4     [Tomatillo-Red Chili Salsa (Hot), [Black Beans...    $16.98
...                                              ...        ...
4617  [Fresh Tomato Salsa, [Rice, Black Beans, Sour ...    $11.75
4618  [Fresh Tomato Salsa, [Rice, Sour Cream, Cheese...    $11.75
4619  [Fresh Tomato Salsa, [Fajita Vegetables, Pinto...    $11.25
4620  [Fresh Tomato Salsa, [Fajita Vegetables, Lettu...     $8.75
4621  [Fresh Tomato Salsa, [Fajita Vegetables, Pinto...     $8.75

[4622 rows x 5 columns]
```

Step 3. Assign it to a variable called chipo.

In [4]:
```
chipo = data
```

Step 4. See the first 10 entries

In [5]:
```
print(chipo.head(10))
```

```
   order_id  quantity                              item_name  \
0         1         1           Chips and Fresh Tomato Salsa
1         1         1                                   Izze
2         1         1                       Nantucket Nectar
3         1         1  Chips and Tomatillo-Green Chili Salsa
4         2         2                           Chicken Bowl
5         3         1                           Chicken Bowl
6         3         1                      Side of Chips
7         4         1                      Steak Burrito
8         4         1                    Steak Soft Tacos
9         5         1                      Steak Burrito

                               choice_description item_price
0                                             NaN     $2.39
1                                   [Clementine]     $3.39
2                                        [Apple]     $3.39
3                                             NaN     $2.39
4  [Tomatillo-Red Chili Salsa (Hot), [Black Beans...    $16.98
5  [Fresh Tomato Salsa (Mild), [Rice, Cheese, Sou...    $10.98
6                                             NaN     $1.69
7  [Tomatillo Red Chili Salsa, [Fajita Vegetables...    $11.75
8  [Tomatillo Green Chili Salsa, [Pinto Beans, Ch...     $9.25
9  [Fresh Tomato Salsa, [Rice, Black Beans, Pinto...     $9.25
```

Step 5. What is the number of observations in the dataset?

In [6]:
```
# Solution 1
temp = chipo.shape
print(temp[0])
```

```
4622
```

In [7]:
```
# Solution 2
chipo.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4622 entries, 0 to 4621
Data columns (total 5 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   order_id            4622 non-null   int64 
 1   quantity            4622 non-null   int64 
 2   item_name           4622 non-null   object
 3   choice_description  3376 non-null   object
 4   item_price          4622 non-null   object
dtypes: int64(2), object(3)
memory usage: 180.7+ KB
```

Step 6. What is the number of columns in the dataset?

In [8]:
```
temp = chipo.shape
print(temp[1])
```

```
5
```

Step 7. Print the name of all the columns.

In [9]:
```
chipo.columns
```

Out[9]:
```
Index(['order_id', 'quantity', 'item_name', 'choice_description',
       'item_price'],
      dtype='object')
```

Step 8. How is the dataset indexed?

In [10]:
```
chipo.index
```

Out[10]:
```
RangeIndex(start=0, stop=4622, step=1)
```

Step 9. Number of Unique Items ?

In [11]:
```
data["item_name"].nunique()
```

Out[11]:
```
50
```

Step 10. Which was the most-ordered item?

In [12]:
```
data.groupby('item_name').sum(numeric_only=True).sort_values('quantity', ascending
```

Out[12]:

item_name	order_id	quantity
Chicken Bowl	713926	761

Step 11. How many items were orderd in total?

In [13]:
```
data['quantity'].sum()
```

Out[13]:
```
4972
```

Step 12. Turn the item price into a float

12.a Check the item price type

In [14]:
```
data['item_price'].dtype
```

Out[14]:
```
dtype('O')
```

Step 12.b. Create a lambda function and change the type of item price

In [15]:
```
data['item_price'] = data['item_price'].apply(lambda x : float(x.replace("$","")))
```

Step 12.c. Check the item price type

In [16]:
```
data['item_price'].dtype
```

Out[16]:
```
dtype('float64')
```

Step 14. How much was the revenue for the period in the dataset?

In [17]:
```
revenue = (data['item_price']*data['quantity']).sum()
print(revenue)
print(revenue.dtype)
```

```
39237.02
float64
```

Step 15. How many orders were made ?

In [18]:
```
data['order_id'].nunique()
```

Out[18]:
```
1834
```

Step 17. How many different choice descriptions are there?

In [19]:
```
data['choice_description'].nunique()
```

Out[19]:
```
1043
```

Step 18. What items have been ordered more than 100 times?

In [20]:
```
most_orderd = data.groupby('item_name')['quantity'].sum()
print(most_orderd.count())
most_orderd = most_orderd[most_orderd > 100]
print(most_orderd)
```

```
50
item_name
Bottled Water                  211
Canned Soda                    126
Canned Soft Drink              351
Chicken Bowl                   761
Chicken Burrito                591
Chicken Salad Bowl             123
Chicken Soft Tacos             120
Chips                          230
Chips and Fresh Tomato Salsa   130
Chips and Guacamole            506
Side of Chips                  110
Steak Bowl                     221
Steak Burrito                  386
Name: quantity, dtype: int64
```

Step 19. What is the average revenue amount per order?

In [21]:
```
# Solution 1
temp = data['order_id'].nunique()
print(revenue/temp)
```

```
21.39423118865867
```

Darshan UNIVERSITY

योगः कर्मसु कौशलम्

Data Mining

Lab - 5

Data Preprocessing

Name: Smit Maru

Enrollment No: 23010101161

1) First, you need to read the titanic dataset from local disk and display Last five records

In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
df = pd.read_csv(r'titanic.csv', encoding='ISO-8859-1')
print(df)
```

```
     PassengerId  Survived  Pclass  \
0              1         0       3
1              2         1       1
2              3         1       3
3              4         1       1
4              5         0       3
..           ...       ...     ...
886          887         0       2
887          888         1       1
888          889         0       3
889          890         1       1
890          891         0       3

                                                 Name     Sex   Age  SibSp  \
0                             Braund, Mr. Owen Harris    male  22.0      1
1    Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
2                              Heikkinen, Miss. Laina  female  26.0      0
3        Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                            Allen, Mr. William Henry    male  35.0      0
..                                                ...     ...   ...    ...
886                            Montvila, Rev. Juozas    male  27.0      0
887                     Graham, Miss. Margaret Edith  female  19.0      0
888         Johnston, Miss. Catherine Helen "Carrie"  female   NaN      1
889                            Behr, Mr. Karl Howell    male  26.0      0
890                              Dooley, Mr. Patrick    male  32.0      0

     Parch            Ticket     Fare Cabin Embarked
0        0         A/5 21171   7.2500   NaN        S
1        0          PC 17599  71.2833   C85        C
2        0  STON/O2. 3101282   7.9250   NaN        S
3        0            113803  53.1000  C123        S
4        0            373450   8.0500   NaN        S
..     ...               ...      ...   ...      ...
886      0            211536  13.0000   NaN        S
887      0            112053  30.0000   B42        S
888      2        W./C. 6607  23.4500   NaN        S
889      0            111369  30.0000  C148        C
890      0            370376   7.7500   NaN        Q

[891 rows x 12 columns]
```

2) Handle Missing Values in data set [use dropna(), fillna(), and interpolate]

In [3]:
```
data_withfillna = df.copy()
mean_age = data_withfillna['Age'].mean()
data_withfillna['Age'] = data_withfillna['Age'].fillna(mean_age)
data_withfillna.head(5)
```

Out[3]:

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

In [4]:

```
data_withfillna = df.copy()
median_age = data_withfillna['Age'].median()
data_withfillna['Age'] = data_withfillna['Age'].fillna(median_age)
data_withfillna.head(5)
```

Out[4]:

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

In [5]:

```
data_withfillna = df.copy()
mode_age = data_withfillna['Age'].mode()
data_withfillna['Age'] = data_withfillna['Age'].fillna(mode_age)
data_withfillna.head(5)
```

Out[5]:

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

3) Apply Scaling to AGE attribute with min max, decimal scaling and z score.

In [6]:
```
min_age = data_withfillna['Age'].min()
max_age = data_withfillna['Age'].max()
data_withfillna['Age_MinMax'] = (data_withfillna['Age'] - min_age) / (max_age - min
data_withfillna['Age_MinMax'].head(5)
```

Out[6]:
```
0    0.271174
1    0.472229
2    0.321438
3    0.434531
4    0.434531
Name: Age_MinMax, dtype: float64
```

In [7]:
```
max_abs_age = data_withfillna['Age'].abs().max()
j = len(str(int(max_abs_age)))
data_withfillna['Age_DecimalScaling'] = data_withfillna['Age'] / (10 ** j)
data_withfillna['Age_DecimalScaling'].head(5)
```

Out[7]:
```
0    0.22
1    0.38
2    0.26
3    0.35
4    0.35
Name: Age_DecimalScaling, dtype: float64
```

In [8]:
```
mean = data_withfillna['Age'].mean()
sd = data_withfillna['Age'].std()
data_withfillna['Age_ZScore'] = (data_withfillna['Age'] - mean) / sd
data_withfillna['Age_ZScore'].head(5)
```

Out[8]:
```
0   -0.530005
1    0.571430
2   -0.254646
3    0.364911
4    0.364911
Name: Age_ZScore, dtype: float64
```

Darshan UNIVERSITY
योगः कर्मसु कौशलम्

Data Mining

Lab - 6

```
# Dimensionality Reduction using NumPy
```

Name: Smit Maru

Enrollment No: 23010101161

🔍 What is Data Reduction?

Data reduction refers to the process of reducing the amount of data that needs to be processed and stored, while preserving the essential patterns in the data.

Why do we reduce data?

- To reduce computational cost.
- To remove noise and redundant features.
- To improve model performance and training time.
- To visualize high-dimensional data in 2D or 3D.

Common data reduction techniques include:

- Principal Component Analysis (PCA)
- Feature selection
- Sampling

📉 What is Principal Component Analysis (PCA)?

PCA is a **dimensionality reduction technique** that transforms a dataset into a new coordinate system. It identifies the **directions (principal components)** where the variance

of the data is maximized.

Key Concepts:

- **Principal Components**: New features (linear combinations of original features) capturing most variance.
- **Eigenvectors & Eigenvalues**: Used to compute these principal directions.
- **Covariance Matrix**: Measures how features vary with each other.

PCA helps in **visualizing high-dimensional data**, **noise reduction**, and **speeding up algorithms**.

🧠 NumPy Functions Summary for PCA

Function	Purpose
`np.mean(X, axis=0)`	Compute mean of each column (feature-wise mean).
`X - np.mean(X, axis=0)`	Centering the data (zero mean).
`np.cov(X, rowvar=False)`	Compute covariance matrix for features.
`np.linalg.eigh(cov_mat)`	Get eigenvalues and eigenvectors (for symmetric matrices).
`np.argsort(values)[::-1]`	Sort values in descending order.
`np.dot(X, eigenvectors)`	Project original data onto new axes.

Step 1: Load the Iris Dataset

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

df = pd.read_csv('iris.csv')
```

Step 2: Standardize the data (zero mean)

In [2]:
```
X = df.iloc[:, 0:4].values
X_meaned = X - np.mean(X, axis=0)
```

Step 3: Compute the Covariance Matrix

In [3]:
```
cov_mat = np.cov(X_meaned, rowvar=False)
print(cov_mat)
```

```
[[ 0.68569351 -0.042434    1.27431544  0.51627069]
 [-0.042434    0.18997942 -0.32965638 -0.12163937]
 [ 1.27431544 -0.32965638  3.11627785  1.2956094 ]
 [ 0.51627069 -0.12163937  1.2956094   0.58100626]]
```

Step 4: Compute eigenvalues and eigenvectors

In [4]:
```
eigen_vals, eigen_vecs = np.linalg.eigh(cov_mat)
print(eigen_vals)
print(eigen_vecs)
```

```
[0.02383509 0.0782095  0.24267075 4.22824171]
[[ 0.31548719  0.58202985  0.65658877 -0.36138659]
 [-0.3197231  -0.59791083  0.73016143  0.08452251]
 [-0.47983899 -0.07623608 -0.17337266 -0.85667061]
 [ 0.75365743 -0.54583143 -0.07548102 -0.3582892 ]]
```

Step 5: Compute eigenvalues and eigenvectors

In [5]:
```
sorted_indices = np.argsort(eigen_vals)[::-1]
eigen_vals = eigen_vals[sorted_indices]
eigen_vecs = eigen_vecs[:, sorted_indices]
print(eigen_vals)
print(eigen_vecs)
```

```
[4.22824171 0.24267075 0.0782095  0.02383509]
[[-0.36138659  0.65658877  0.58202985  0.31548719]
 [ 0.08452251  0.73016143 -0.59791083 -0.3197231 ]
 [-0.85667061 -0.17337266 -0.07623608 -0.47983899]
 [-0.3582892  -0.07548102 -0.54583143  0.75365743]]
```

Step 6: Select the top k eigenvectors (top 2)

In [6]:
```
top_2_eigen_vecs = eigen_vecs[:, :2]
top_2_eigen_vecs
```

Out[6]:
```
array([[-0.36138659,  0.65658877],
       [ 0.08452251,  0.73016143],
       [-0.85667061, -0.17337266],
       [-0.3582892 , -0.07548102]])
```

Step 7: Project the data onto the top k eigenvectors

In [7]:
```
X_reduced = np.dot(X_meaned, top_2_eigen_vecs)
print("Reduced data shape:", X_reduced.shape)
```

```
Reduced data shape: (150, 2)
```

Step 8: Plot the PCA-Reduced Data

In [8]:
```
species = df['species'].values
unique_species = np.unique(species)
colors = {'setosa': 'red', 'versicolor': 'green', 'virginica': 'blue'}

for sp in unique_species:
    idx = np.where(species == sp)
    plt.scatter(X_reduced[idx, 0], X_reduced[idx, 1], label=sp, color=colors[sp])

plt.title("PCA - Iris Dataset")
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.legend()
plt.grid(True)
plt.show()
```

Extra - Bining Method

5,10,11,13,15,35,50,55,72,92,204,215.

Partition them into three bins by each of the following methods: (a) equal-frequency (equal-depth) partitioning (b) equal-width partitioning

In [9]:
```
data = pd.Series([5, 10, 11, 13, 15, 35, 50, 55, 72, 92, 204, 215])
equal_freq = pd.qcut(data, q=3)
print("Equal-Frequency Bins:")
print(equal_freq)
```

```
Equal-Frequency Bins:
0      (4.999, 14.333]
1      (4.999, 14.333]
2      (4.999, 14.333]
3      (4.999, 14.333]
4     (14.333, 60.667]
5     (14.333, 60.667]
6     (14.333, 60.667]
7     (14.333, 60.667]
8      (60.667, 215.0]
9      (60.667, 215.0]
10     (60.667, 215.0]
11     (60.667, 215.0]
dtype: category
Categories (3, interval[float64, right]): [(4.999, 14.333] < (14.333, 60.667] < (60.
667, 215.0]]
```

In [10]:
```
equal_width = pd.cut(data, bins=3)
print("Equal-Width Bins:")
print(equal_width)
```

```
Equal-Width Bins:
0       (4.79, 75.0]
1       (4.79, 75.0]
2       (4.79, 75.0]
3       (4.79, 75.0]
4       (4.79, 75.0]
5       (4.79, 75.0]
6       (4.79, 75.0]
7       (4.79, 75.0]
8       (4.79, 75.0]
9      (75.0, 145.0]
10    (145.0, 215.0]
11    (145.0, 215.0]
dtype: category
Categories (3, interval[float64, right]): [(4.79, 75.0] < (75.0, 145.0] < (145.0, 21
5.0]]
```