Data Mining

Lab - 3

23010101407

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

In [5]:
```
import pandas as pd
import numpy as np
```

In [7]:
```
titanic = pd.read_csv("titanic.csv")
```

In [9]:
```
titanic.head(5)
```

Out[9]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	
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

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

In [12]:
```
Nominal = ['Name','Sex','Cabin','Embarked','Ticket']
Ordinal = ['Pclass']
binary = ['Sex','Survived']
Numeric = ['Age','Fare','SibSp','Parch']

print(Nominal)
print(Ordinal)
print(binary)
print(Numeric)
```

```
['Name', 'Sex', 'Cabin', 'Embarked', 'Ticket']
['Pclass']
['Sex', 'Survived']
['Age', 'Fare', 'SibSp', 'Parch']
```

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

In [15]:
```
print("survied assymrtric count ----------")
print(titanic['Survived'].value_counts())
print('================================================================================
print("gender count symmetric count ----------")
print(titanic['Sex'].value_counts())
```

```
survied assymrtric count ----------
Survived
0    549
1    342
Name: count, dtype: int64
================================================================================
=
gender count symmetric count ----------
Sex
male      577
female    314
Name: count, dtype: int64
```

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

In [18]:
```
quantitative = ['PassengerId','Survived','Pclass','Age','SibSp','Parch','Fare']
for col in quantitative:
    print(col,":")
    print("mean =",titanic[col].mean())
    print("Standard deviation =",titanic[col].std())
    print("minimum =",titanic[col].min())
    print("mode =",titanic[col].mode()[0])
    print("Max =",titanic[col].max())
    print("range =",titanic[col].max() - titanic[col].min())
    print("-----------------------------------------------")
```

```
PassengerId :
mean = 446.0
Standard deviation = 257.3538420152301
minimum = 1
mode = 1
Max = 891
range = 890
------------------------------------------------
Survived :
mean = 0.3838383838383838
Standard deviation = 0.4865924542648585
minimum = 0
mode = 0
Max = 1
range = 1
------------------------------------------------
Pclass :
mean = 2.308641975308642
Standard deviation = 0.8360712409770513
minimum = 1
mode = 3
Max = 3
range = 2
------------------------------------------------
Age :
mean = 29.69911764705882
Standard deviation = 14.526497332334044
minimum = 0.42
mode = 24.0
Max = 80.0
range = 79.58
------------------------------------------------
SibSp :
mean = 0.5230078563411896
Standard deviation = 1.1027434322934275
minimum = 0
mode = 0
Max = 8
range = 8
------------------------------------------------
Parch :
mean = 0.38159371492704824
Standard deviation = 0.8060572211299559
minimum = 0
mode = 0
Max = 6
range = 6
------------------------------------------------
Fare :
mean = 32.204207968574636
Standard deviation = 49.693428597180905
minimum = 0.0
mode = 8.05
Max = 512.3292
range = 512.3292
------------------------------------------------
```

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

In [21]:
```
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 [24]:
```
print("describe numeric attribute::")
print(titanic.describe())
print('------------------')
print("describe ALL numeric attribute::")
print(titanic.describe(include = 'all'))
print('------------------')
print("describe Object numeric attribute::")
print(titanic.describe(include = 'object'))
print('------------------')
```

```
describe numeric attribute::
       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
------------------
describe ALL numeric attribute::
       PassengerId    Survived      Pclass                     Name   Sex  \
count    891.000000  891.000000  891.000000                      891   891
unique          NaN         NaN         NaN                      891     2
top             NaN         NaN         NaN  Braund, Mr. Owen Harris  male
freq            NaN         NaN         NaN                        1   577
mean     446.000000    0.383838    2.308642                      NaN   NaN
std      257.353842    0.486592    0.836071                      NaN   NaN
min        1.000000    0.000000    1.000000                      NaN   NaN
25%      223.500000    0.000000    2.000000                      NaN   NaN
50%      446.000000    0.000000    3.000000                      NaN   NaN
75%      668.500000    1.000000    3.000000                      NaN   NaN
max      891.000000    1.000000    3.000000                      NaN   NaN

               Age       SibSp       Parch  Ticket        Fare    Cabin  \
count   714.000000  891.000000  891.000000     891  891.000000      204
unique         NaN         NaN         NaN     681         NaN      147
top            NaN         NaN         NaN  347082         NaN  B96 B98
freq           NaN         NaN         NaN       7         NaN        4
mean     29.699118    0.523008    0.381594     NaN   32.204208      NaN
std      14.526497    1.102743    0.806057     NaN   49.693429      NaN
min       0.420000    0.000000    0.000000     NaN    0.000000      NaN
25%      20.125000    0.000000    0.000000     NaN    7.910400      NaN
50%      28.000000    0.000000    0.000000     NaN   14.454200      NaN
75%      38.000000    1.000000    0.000000     NaN   31.000000      NaN
max      80.000000    8.000000    6.000000     NaN  512.329200      NaN

       Embarked
count       889
unique        3
top           S
freq        644
mean        NaN
std         NaN
min         NaN
25%         NaN
50%         NaN
75%         NaN
max         NaN
```

```
------------------
describe Object numeric attribute::
                           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 [27]:
```
print("corelation numeric col:")
print(titanic.corr(numeric_only=True))
```

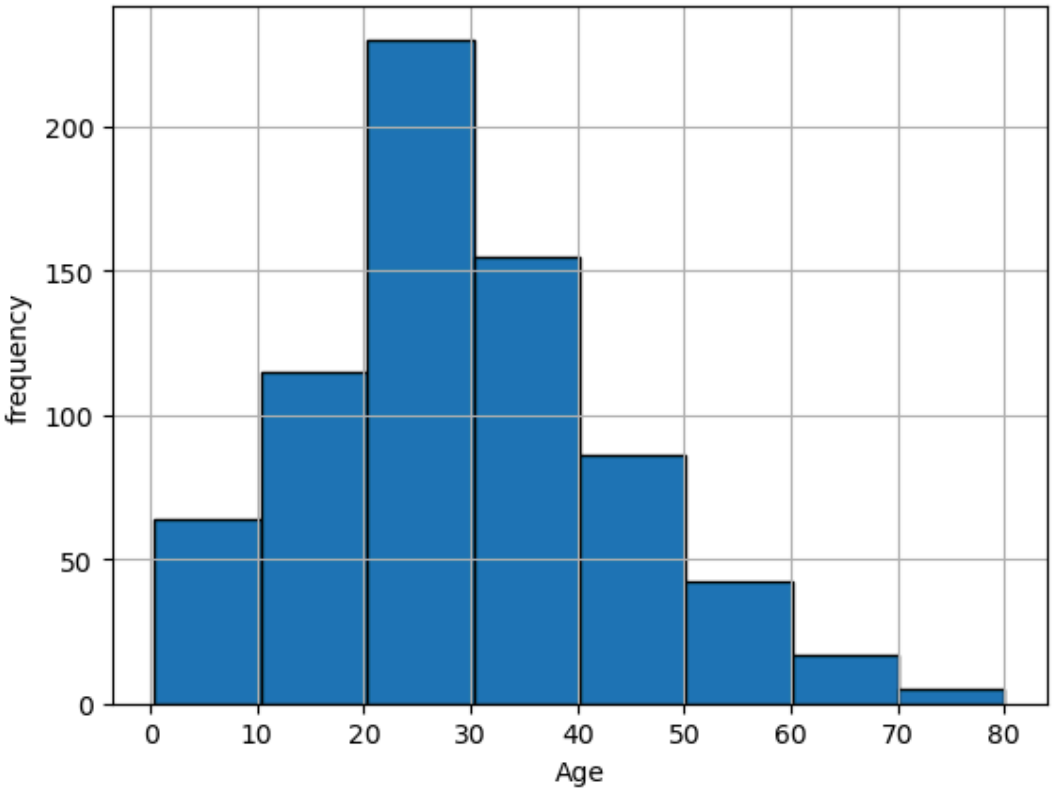
```
corelation numeric col:
             PassengerId  Survived    Pclass       Age     SibSp     Parch  \
PassengerId     1.000000 -0.005007 -0.035144  0.036847 -0.057527 -0.001652
Survived       -0.005007  1.000000 -0.338481 -0.077221 -0.035322  0.081629
Pclass         -0.035144 -0.338481  1.000000 -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
```

In [29]:
```
print("corelation numeric col:")
print(titanic.corr(numeric_only=True))
```

```
corelation numeric col:
             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 [32]:
```
import matplotlib.pyplot as plt
titanic['Age'].dropna().hist(bins=8,edgecolor='black')
plt.xlabel('Age')
plt.ylabel('frequency')
plt.show()
```



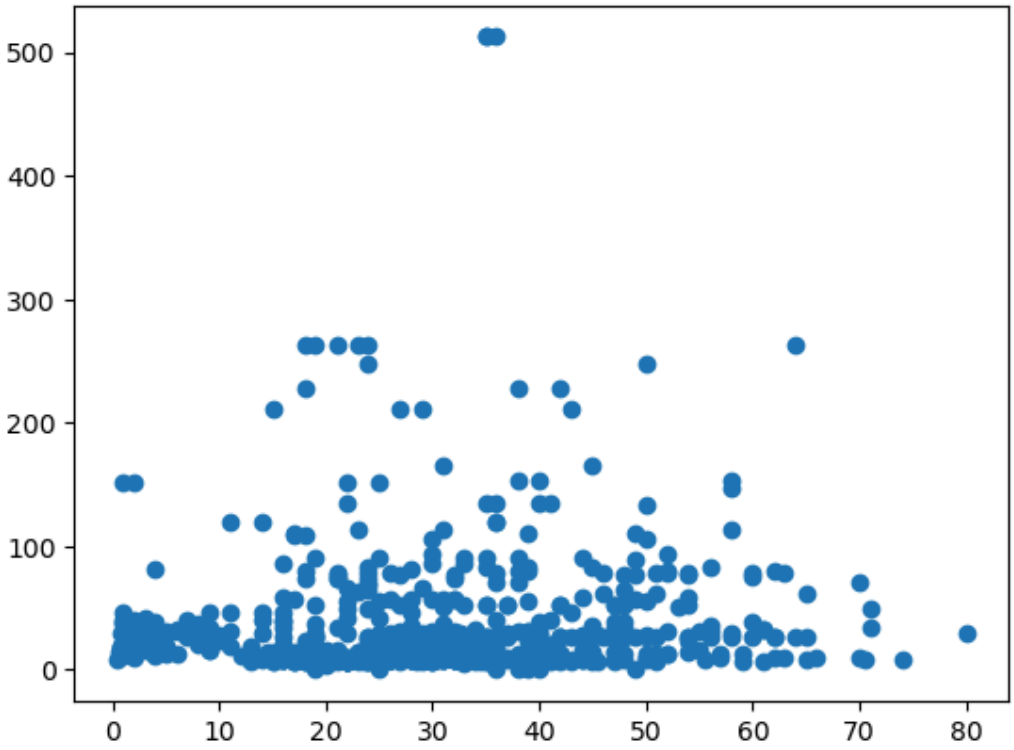
In [ ]:
```

```

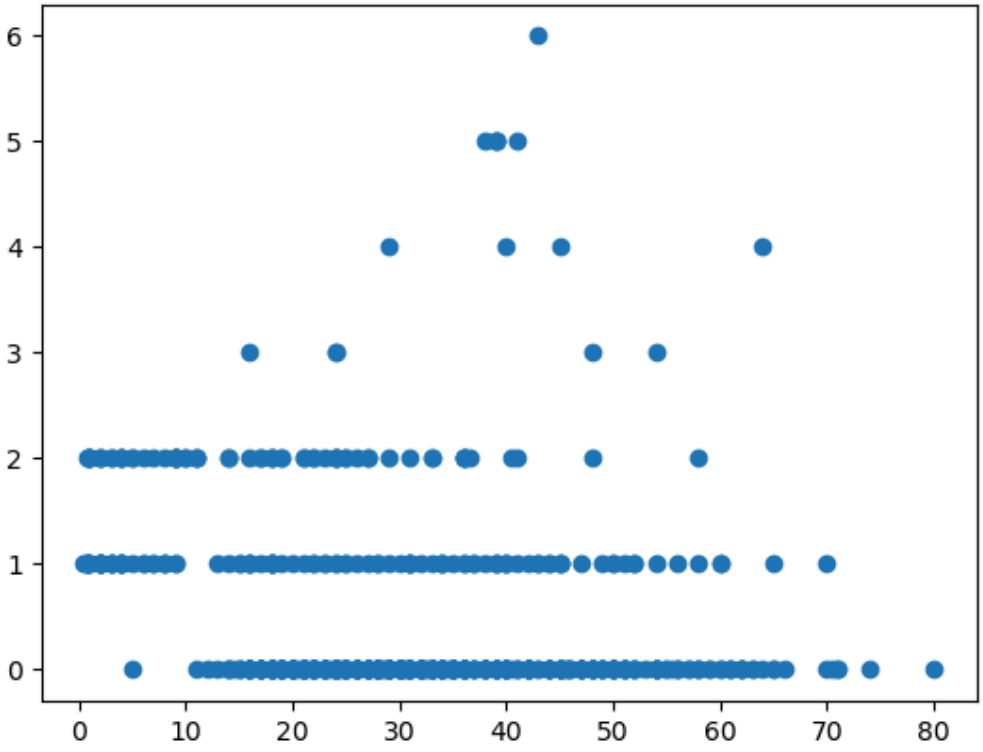
10) A boxplot can also be used to show the distribution of values for each attribute.

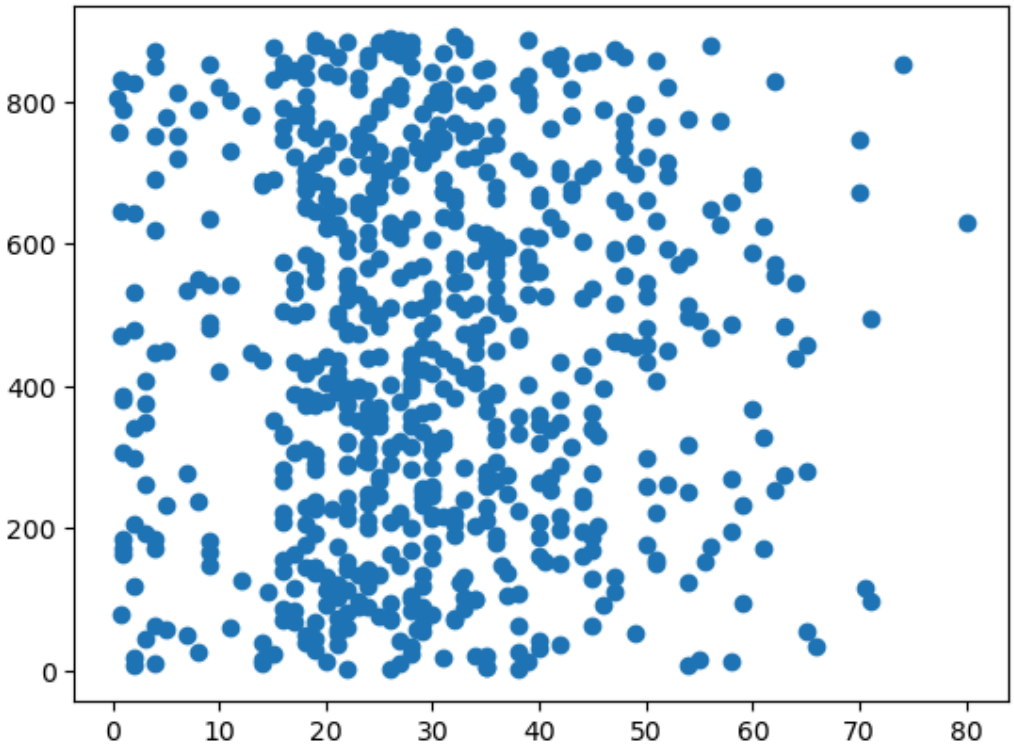
In [ ]:
```

```

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

In [38]:
```
plt.scatter(titanic['Age'],titanic['Fare'])
plt.show()

plt.scatter(titanic['Age'],titanic['Parch'])
plt.show()
plt.scatter(titanic['Age'],titanic['PassengerId'])
plt.show()
```

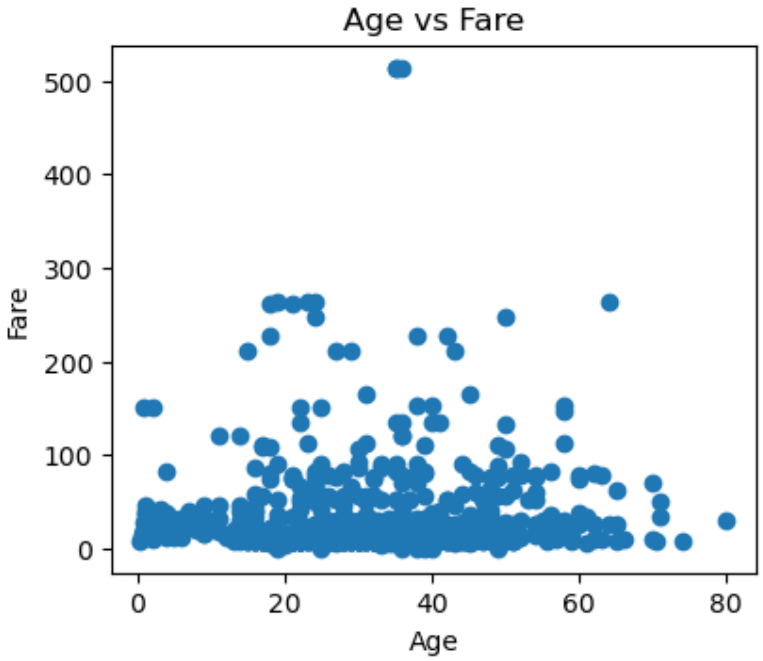

500
400
300
200
100
0
0
10
20
30
40
50
60
70
80


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



In [46]:
```
pairs = [
    {'Age','Fare'},
    {'Age','sibsp'},
    {'Age','Parch'},
    {'Fare','sibsp'},
    {'Fare','Parch'},
]
plt.figure(figsize=(15,8))

for i, (x,y) in enumerate(pairs):
        plt.subplot(2,3,i+1)
        plt.scatter(titanic[x],titanic[y])
        plt.xlabel(x)
        plt.ylabel(y)
        plt.title(f'{x} vs {y}')
        plt.grid(true)


plt.show()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[46], line 16
     14         plt.ylabel(y)
     15         plt.title(f'{x} vs {y}')
---> 16         plt.grid(true)
     19 plt.show()

NameError: name 'true' is not defined
```



In [ ]:

Data Mining

Lab - 4

23010101407

Step 1. Import the necessary libraries

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

Step 2. Import the dataset from this address.

Step 3. Assign it to a variable called chipo.

In [5]:
```
url = 'https://raw.githubusercontent.com/justmarkham/DAT8/master/data/chipotle.t
chipo = pd.read_csv(url, sep='\t')
```

Step 4. See the first 10 entries

In [7]:
```
chipo.head(10)
```

Out[7]:

	order_id	quantity	item_name	choice_description	item_price
0	1	1	Chips and Fresh Tomato Salsa	NaN	$2.39
1	1	1	Izze	[Clementine]	$3.39
2	1	1	Nantucket Nectar	[Apple]	$3.39
3	1	1	Chips and Tomatillo-Green Chili Salsa	NaN	$2.39
4	2	2	Chicken Bowl	[Tomatillo-Red Chili Salsa (Hot), [Black Beans...	$16.98
5	3	1	Chicken Bowl	[Fresh Tomato Salsa (Mild), [Rice, Cheese, Sou...	$10.98
6	3	1	Side of Chips	NaN	$1.69
7	4	1	Steak Burrito	[Tomatillo Red Chili Salsa, [Fajita Vegetables...	$11.75
8	4	1	Steak Soft Tacos	[Tomatillo Green Chili Salsa, [Pinto Beans, Ch...	$9.25
9	5	1	Steak Burrito	[Fresh Tomato Salsa, [Rice, Black Beans, Pinto...	$9.25

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

In [13]:
```
# Solution 1
chipo.shape[0]
```

Out[13]: 4622

In [15]:
```
# 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 [17]:
```
chipo.shape[1]
```

Out[17]: 5

Step 7. Print the name of all the columns.

In [19]:
```
chipo.columns
```

Out[19]:
```
Index(['order_id', 'quantity', 'item_name', 'choice_description',
       'item_price'],
      dtype='object')
```

Step 8. How is the dataset indexed?

In [21]:
```
chipo.index
```

Out[21]:
```
RangeIndex(start=0, stop=4622, step=1)
```

Step 9. Number of Unique Items ?

In [29]:
```
print(chipo.nunique())
print(chipo['item_name'].nunique())
```

```
order_id              1834
quantity                 9
item_name               50
choice_description    1043
item_price              78
dtype: int64
50
```

Step 10. Which was the most-ordered item?

In [60]:
```
c = chipo.groupby('item_name')
c = c.sum()
c = c.sort_values(['quantity'], ascending = False)
c.head(1)
```

Out[60]:

item_name	order_id	quantity	choice_description	item_price
Chicken Bowl	713926	761	[Tomatillo-Red Chili Salsa (Hot), [Black Beans...	16.9810.98 11.258.75 8.4911.25 $8.75 ...

Step 11. How many items were orderd in total?

In [48]:
```
chipo['quantity'].sum()
```

Out[48]:
```
4972
```

Step 12. Turn the item price into a float

Step 12.a. Check the item price type

In [56]:
```
chipo['item_price'].dtype
```

Out[56]:
```
dtype('O')
```

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

In [70]:
```
dollarization = lambda x: float(x[1:-1])
chipo.item_price = chipo.item_price.apply(dollarization)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[70], line 2
      1 dollarization = lambda x: float(x[1:-1])
----> 2 chipo.item_price = chipo.item_price.apply(dollarization)

File ~\anaconda3\Lib\site-packages\pandas\core\series.py:4924, in Series.apply(se
lf, func, convert_dtype, args, by_row, **kwargs)
   4789 def apply(
   4790     self,
   4791     func: AggFuncType,
   (...)
   4796     **kwargs,
   4797 ) -> DataFrame | Series:
   4798     """
   4799     Invoke function on values of Series.
   4800
   (...)
   4915     dtype: float64
   4916     """
   4917     return SeriesApply(
   4918         self,
   4919         func,
   4920         convert_dtype=convert_dtype,
   4921         by_row=by_row,
   4922         args=args,
   4923         kwargs=kwargs,
-> 4924     ).apply()

File ~\anaconda3\Lib\site-packages\pandas\core\apply.py:1427, in SeriesApply.appl
y(self)
   1424     return self.apply_compat()
   1426 # self.func is Callable
-> 1427 return self.apply_standard()

File ~\anaconda3\Lib\site-packages\pandas\core\apply.py:1507, in SeriesApply.appl
y_standard(self)
   1501 # row-wise access
   1502 # apply doesn't have a `na_action` keyword and for backward compat reason
s
   1503 # we need to give `na_action="ignore"` for categorical data.
   1504 # TODO: remove the `na_action="ignore"` when that default has been change
d in
   1505 #  Categorical (GH51645).
   1506 action = "ignore" if isinstance(obj.dtype, CategoricalDtype) else None
-> 1507 mapped = obj._map_values(
   1508     mapper=curried, na_action=action, convert=self.convert_dtype
   1509 )
   1511 if len(mapped) and isinstance(mapped[0], ABCSeries):
   1512     # GH#43986 Need to do list(mapped) in order to get treated as nested
   1513     #  See also GH#25959 regarding EA support
   1514     return obj._constructor_expanddim(list(mapped), index=obj.index)

File ~\anaconda3\Lib\site-packages\pandas\core\base.py:921, in IndexOpsMixin._map
_values(self, mapper, na_action, convert)
    918 if isinstance(arr, ExtensionArray):
    919     return arr.map(mapper, na_action=na_action)
--> 921 return algorithms.map_array(arr, mapper, na_action=na_action, convert=con
vert)
```

```
File ~\anaconda3\Lib\site-packages\pandas\core\algorithms.py:1743, in map_array(a
rr, mapper, na_action, convert)
   1741 values = arr.astype(object, copy=False)
   1742 if na_action is None:
-> 1743     return lib.map_infer(values, mapper, convert=convert)
   1744 else:
   1745     return lib.map_infer_mask(
   1746         values, mapper, mask=isna(values).view(np.uint8), convert=convert
   1747     )

File lib.pyx:2972, in pandas._libs.lib.map_infer()

Cell In[70], line 1, in <lambda>(x)
----> 1 dollarization = lambda x: float(x[1:-1])
      2 chipo.item_price = chipo.item_price.apply(dollarization)

TypeError: 'float' object is not subscriptable
```

Step 12.c. Check the item price type

In [72]:
```
chipo['item_price'].dtype
```

Out[72]:
```
dtype('float64')
```

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

In [78]:
```
final_revenue = (chipo['item_price'] * chipo['quantity']).sum()
print(final_revenue)
```

```
39237.02
```

Step 15. How many orders were made ?

In [84]:
```
chipo['order_id'].nunique()
```

Out[84]:
```
1834
```

Step 17. How many different choice descriptions are there?

In [86]:
```
chipo['choice_description'].nunique()
```

Out[86]:
```
1043
```

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

In [92]:
```
c = chipo.groupby('item_name')['quantity'].sum()
c[c>100]
```

Out[92]:
```
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 [103…
```
# Solution 1
chipo['revenue'] = (chipo['item_price'] * chipo['quantity'])
order_grouped = chipo.groupby('order_id').sum()
order_grouped['revenue'].mean()
```

Out[103… 21.39423118865867

In [ ]:
```
# Solution 2
```

Out[ ]: 21.394231188658654

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Lab - 5 - Data Preprocessing

23010101407

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

In [3]:
```
import pandas as pd
import numpy as np
```

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

Out[5]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	7
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	5
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	
...	...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	1
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	3
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	2
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	3
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	

891 rows × 12 columns

In [7]:
```
data2 = data.copy()
data2.head()
```

Out[7]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	
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

In [9]:
```
data2.tail()
```

Out[9]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Far
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

In [11]:
```
data2.nunique()
```

Out[11]:
```
PassengerId    891
Survived         2
Pclass           3
Name           891
Sex              2
Age             88
SibSp            7
Parch            7
Ticket         681
Fare           248
Cabin          147
Embarked         3
dtype: int64
```

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

In [14]:
```
data_dropnaAll = data2.dropna(how='all')
data_dropna = data2.dropna()
data_dropna.count()
# axis:
# Specifies whether to drop rows or columns.
# 0 or 'index' (default): Drops rows containing missing values.
# 1 or 'columns': Drops columns containing missing values.
```

Out[14]:
```
PassengerId    183
Survived       183
Pclass         183
Name           183
Sex            183
Age            183
SibSp          183
Parch          183
Ticket         183
Fare           183
Cabin          183
Embarked       183
dtype: int64
```

In [16]:
```
data_fillna = data2.fillna({'Age' : 35,'Cabin' : 'Not Available'})
# data_fillna = data2.fillna(35)
data_fillna
```

Out[16]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	7
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	5
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	
...	...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	1
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	3
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	35.0	1	2	W./C. 6607	2
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	3
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	

891 rows × 12 columns

In [ ]:

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

In [20]:
```
age_mean = data2.Age.mean()
age_mean
fill_data = data2.fillna({'Age' : age_mean})
fill_data
```

Out[20]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Tick
0	1	0	3	Braund, Mr. Owen Harris	male	22.000000	1	0	A/ 211
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.000000	1	0	PC 175
2	3	1	3	Heikkinen, Miss. Laina	female	26.000000	0	0	STON/O 31012
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.000000	1	0	1138
4	5	0	3	Allen, Mr. William Henry	male	35.000000	0	0	3734
...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.000000	0	0	2115
887	888	1	1	Graham, Miss. Margaret Edith	female	19.000000	0	0	1120
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	29.699118	1	2	W./ 66
889	890	1	1	Behr, Mr. Karl Howell	male	26.000000	0	0	1113
890	891	0	3	Dooley, Mr. Patrick	male	32.000000	0	0	3703

891 rows × 12 columns

In [22]:
```
data2['Age'].min()
```

Out[22]:
```
0.42
```

In [24]:
```
data['Age'].max()
```

Out[24]:
```
80.0
```

In [26]:
```
data_interpolation = data2.interpolate()
data_interpolation
```

```
C:\Users\dholn\AppData\Local\Temp\ipykernel_18036\3603189312.py:1: FutureWarning:
DataFrame.interpolate with object dtype is deprecated and will raise in a future
version. Call obj.infer_objects(copy=False) before interpolating instead.
  data_interpolation = data2.interpolate()
```

Out[26]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	7
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	5
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	
...	...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	1
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	3
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	22.5	1	2	W./C. 6607	2
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	3
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	

891 rows × 12 columns

In [28]:
```
data2.fillna(data2.Age.mean(),inplace=True)
```

In [30]:
```
minAge = data2.Age.min()
maxAge = data2.Age.max()
```

```
data2['MinMaxAge'] = (data2['Age']-minAge)/(maxAge-minAge)
data2
```

Out[30]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Tick
0	1	0	3	Braund, Mr. Owen Harris	male	22.000000	1	0	A 211
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.000000	1	0	PC 175
2	3	1	3	Heikkinen, Miss. Laina	female	26.000000	0	0	STON/O 31012
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.000000	1	0	1138
4	5	0	3	Allen, Mr. William Henry	male	35.000000	0	0	3734
...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.000000	0	0	2115
887	888	1	1	Graham, Miss. Margaret Edith	female	19.000000	0	0	1120
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	29.699118	1	2	W./ 66
889	890	1	1	Behr, Mr. Karl Howell	male	26.000000	0	0	1113
890	891	0	3	Dooley, Mr. Patrick	male	32.000000	0	0	3703

891 rows × 13 columns

In [32]:
```
data2.describe(include='all')
```

Out[32]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	
count	891.000000	891.000000	891.000000	891	891	891.000000	891.000000	89
unique	NaN	NaN	NaN	891	2	NaN	NaN	
top	NaN	NaN	NaN	Braund, Mr. Owen Harris	male	NaN	NaN	
freq	NaN	NaN	NaN	1	577	NaN	NaN	
mean	446.000000	0.383838	2.308642	NaN	NaN	29.699118	0.523008	
std	257.353842	0.486592	0.836071	NaN	NaN	13.002015	1.102743	
min	1.000000	0.000000	1.000000	NaN	NaN	0.420000	0.000000	
25%	223.500000	0.000000	2.000000	NaN	NaN	22.000000	0.000000	
50%	446.000000	0.000000	3.000000	NaN	NaN	29.699118	0.000000	
75%	668.500000	1.000000	3.000000	NaN	NaN	35.000000	1.000000	
max	891.000000	1.000000	3.000000	NaN	NaN	80.000000	8.000000	

In [34]:
```
maxAge = data2.Age.max()
noOfDigits = len(str(int(maxAge)))
noOfDigits
```

Out[34]: 2

In [36]:
```
data2['AgeDS'] = data2['Age']/(10**noOfDigits)
```

In [38]:
```
data2
```

Out[38]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Tick
0	1	0	3	Braund, Mr. Owen Harris	male	22.000000	1	0	A 211
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.000000	1	0	PC 175
2	3	1	3	Heikkinen, Miss. Laina	female	26.000000	0	0	STON/O 31012
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.000000	1	0	1138
4	5	0	3	Allen, Mr. William Henry	male	35.000000	0	0	3734
...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.000000	0	0	2115
887	888	1	1	Graham, Miss. Margaret Edith	female	19.000000	0	0	1120
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	29.699118	1	2	W./ 66
889	890	1	1	Behr, Mr. Karl Howell	male	26.000000	0	0	1113
890	891	0	3	Dooley, Mr. Patrick	male	32.000000	0	0	3703

891 rows × 14 columns

In [40]:

```
mins = data2.Age.min()
stds = data2.Age.std()
data2['ageScore'] = (data2['Age'] - mins)/stds
data2
```

Out[40]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Tick
0	1	0	3	Braund, Mr. Owen Harris	male	22.000000	1	0	A 211
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.000000	1	0	PC 175
2	3	1	3	Heikkinen, Miss. Laina	female	26.000000	0	0	STON/O 31012
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.000000	1	0	1138
4	5	0	3	Allen, Mr. William Henry	male	35.000000	0	0	3734
...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.000000	0	0	2115
887	888	1	1	Graham, Miss. Margaret Edith	female	19.000000	0	0	1120
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	29.699118	1	2	W./ 66
889	890	1	1	Behr, Mr. Karl Howell	male	26.000000	0	0	1113
890	891	0	3	Dooley, Mr. Patrick	male	32.000000	0	0	3703

891 rows × 15 columns

In [ ]:

In [ ]:

Darshan UNIVERSITY
योगः कर्मसु कौशलम्

Data Mining

Lab - 6

23010101407

```
# Dimensionality Reduction using NumPy
```

🔍 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 [5]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [6]:
```
iris = pd.read_csv('iris.csv')
demo = iris.copy()
demo
```

Out[6]:

	sepal_length	sepal_width	petal_length	petal_width	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
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

In [8]:
```
x = demo.drop(columns="species")
y = demo['species'].map({
    'setosa' : 0,
    'versicolor' : 1,
    'virginica' : 2
})

print("original shape :: ", x.shape)
```

```
original shape ::  (150, 4)
```

Step 2: Standardize the data (zero mean)

In [12]:
```
x_meaned = x - np.mean(x,axis=0)
print(x_meaned[:5])
```

```
   sepal_length  sepal_width  petal_length  petal_width
0     -0.743333     0.442667        -2.358    -0.999333
1     -0.943333    -0.057333        -2.358    -0.999333
2     -1.143333     0.142667        -2.458    -0.999333
3     -1.243333     0.042667        -2.258    -0.999333
4     -0.843333     0.542667        -2.358    -0.999333
```

Step 3: Compute the Covariance Matrix

In [26]:
```
cov_mat = np.cov(x_meaned,rowvar=False)
print(cov_mat)
print(" covarience of matrix shape : ", cov_mat.shape)
```

```
[[ 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]]
 covarience of matrix shape :  (4, 4)
```

Step 4: Compute eigenvalues and eigenvectors

In [24]:
```
eigen_values,eigen_vector = np.linalg.eigh(cov_mat)
print("eigen_values",eigen_values)
print('\neigen_vector\n',eigen_vector[:,:2])
```

```
eigen_values [0.02383509 0.0782095  0.24267075 4.22824171]

eigen_vector
 [[ 0.31548719  0.58202985]
 [-0.3197231  -0.59791083]
 [-0.47983899 -0.07623608]
 [ 0.75365743 -0.54583143]]
```

Step 5: Compute eigenvalues and eigenvectors in descending order

In [35]:
```
sorted_index = np.argsort(eigen_values[::-1])
sorted_eigenvalues = eigen_values[sorted_index]
sorted_eigenvecotores = eigen_vector[:,sorted_index]
print(sorted_index)
print(sorted_eigenvalues)
print(sorted_eigenvecotores)
```

```
[3 2 1 0]
[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 [45]:
```
k = 2
eigenvector_subset = sorted_eigenvecotores[:,0:k]
print(eigenvector_subset)
```

```
[[-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

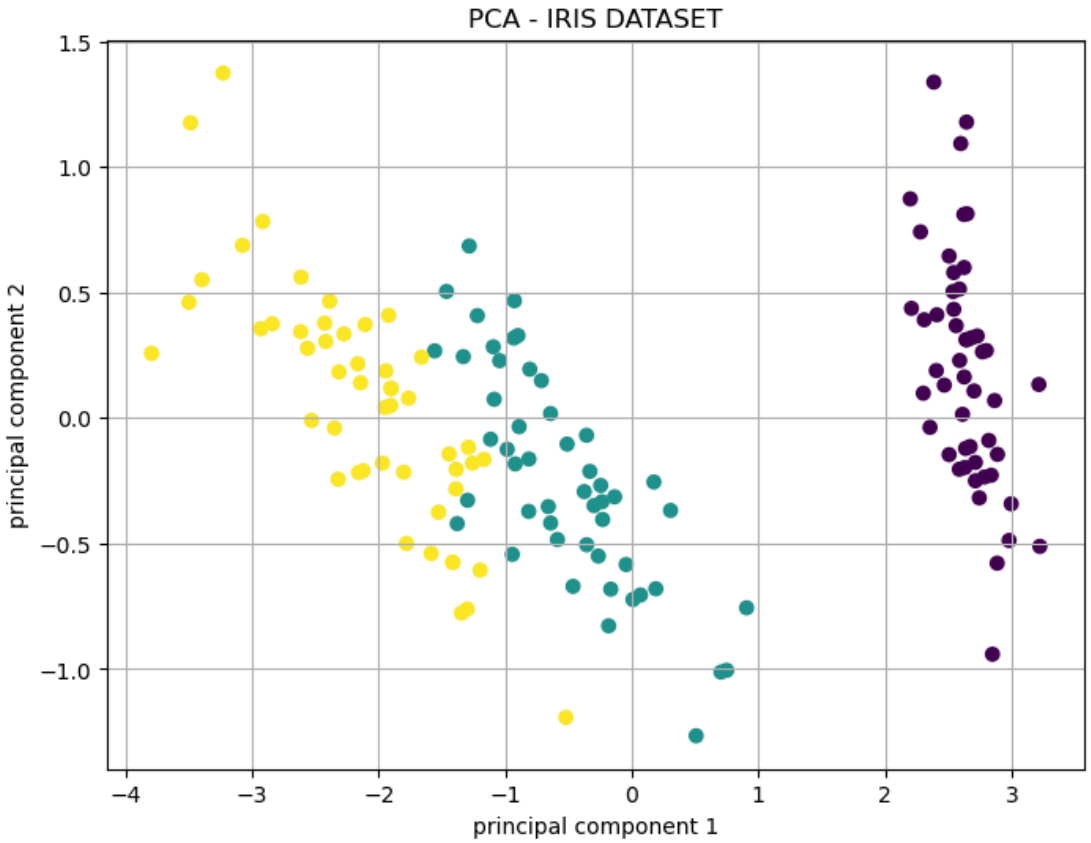
In [49]:
```
x_reduced = np.dot(x_meaned, eigenvector_subset)
print("Reduced data shape :: ",x_reduced.shape)
```

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

Step 8: Plot the PCA-Reduced Data

In [57]:
```
plt.figure(figsize=(8,6))
plt.scatter(x_reduced[:,0],x_reduced[:,1],c=y)
plt.xlabel('principal component 1')
plt.ylabel('principal component 2')
plt.title('PCA - IRIS DATASET')
plt.grid()
plt.plot()
```

Out[57]: []



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 [23]:
```
equal_fre
```

```
Sorted Data: [5, 10, 11, 13, 15, 35, 50, 55, 72, 92, 204, 215]

(a) Equal-Frequency Bins:
Bin 1: [5, 10, 11, 13]
Bin 2: [15, 35, 50, 55]
Bin 3: [72, 92, 204, 215]

(b) Equal-Width Bins:
Bin 1: [5, 10, 11, 13, 15, 35, 50, 55, 72]
Bin 2: [92]
Bin 3: [204, 215]
```

In [71]:
```
210/3
```

Out[71]: 70.0

In [77]:
```
5+70
```

Out[77]: 75

In [79]:
```
90+70
```

Out[79]: 160

In [ ]:
```

```

Darshan UNIVERSITY
योग: कर्मसु कौशलम्

Data Mining

Lab - 7 (Part 2)

23010101407

Step 1: Load the Dataset

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

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

data = pd.read_csv('Tdata.csv')
data.head()
```

Out[1]:

	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 [4]:
```
data.drop("Transaction",axis=1,inplace=True)
```

Step 3: Count Single Items

See how many transactions include each item.

In [7]:
```
data.sum()
```

Out[7]:
```
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 [10]:
```
from itertools import combinations

def find_frequent_itemsets(data, min_support):
    n = len(data)
    result = []

    for k in [1,2,3]: # for 1-item , 2-item , 3-item
        for items in combinations(data.columns,k):
            mask = data[list(items)].all(axis=1)
            support = mask.sum() / n
            if support >= min_support:
                result.append((frozenset(items), round(support, 2)))

    return result
```

Step 5: Run Apriori

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

In [13]:
```
frequent_itemsets=find_frequent_itemsets(data,min_support=0.5)
for itemset,support in frequent_itemsets:
    print(f"{set(itemset)}-> support: {support}")
```

```
{'bread'}-> support: 0.83
{'butter'}-> support: 0.5
{'milk'}-> support: 0.5
{'bread', 'butter'}-> support: 0.5
{'bread', 'milk'}-> support: 0.5
```

Step 6 Display as a DataFrame

In [15]:
```
result_data = pd.DataFrame(frequent_itemsets,columns=['itemset','support'])
result_data
```

Out[15]:

	itemset	support
0	(bread)	0.83
1	(butter)	0.50
2	(milk)	0.50
3	(bread, butter)	0.50
4	(bread, milk)	0.50

In [ ]:

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

In [ ]:

Extra : - > Define Apriori Function without itertools

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Lab - 7

Date
Page

* Apply Apriori Algoritham

(1)

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

Scan D

Ans :- C_1

itemset	minsup
{1}	2
{2}	3
{3}	3
{4}	1 X
{5}	3

L_1

items	support
{1}	2
{2}	3
{3}	3
{5}	3

Frequent - 2

C_2

items	support
{1,2}	1 X
{1,3}	2
{1,5}	1 X
{2,3}	2
{2,5}	3
{3,5}	2

L_2

items	support
{1,3}	2
{2,3}	2
{2,5}	3
{3,5}	2

Frequent - 3

C_3

item	support
{1,2,3}	1 X
{1,3,5}	1 X
{2,3,5}	2

L_3

item	support
{2,3,5}	2

=> Rule Generation

Association Rule	support	confidence	%
2^3 → 5	2	2/2 = 1	100%
3^5 → 2	2	2/2 = 1	100%
2^5 → 3	2	2/3 = 0.66	66%
2 → 3^5	2	2/3 = 0.66	66%
3 → 2^5	2	2/3 = 0.66	66%
5 → 2^3	2	2/3 = 0.66	66%

Strong association → 2^3 → 5
3^5 → 2

(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

Min-sup=3

frequent 1

C1

Items	Support
{Bread}	3
{diaper}	4
{beer}	3
{eggs}	1 ✗
{cola}	3
{milk}	4

L1

items	Support
{bread}	3
{diaper}	4
{beer}	3
{cola}	3
{milk}	4

frequent - 2

C_2 Items	support
{bread, diaper}	2 ✗
{bread, beer}	1 ✗
{bread, cola}	1 ✗
{bread, milk}	2 ✗
{diaper, beer}	3
{diaper, cola}	3
{diaper, milk}	3
{beer, cola}	2 ✗
{beer, milk}	2 ✗
{cola, milk}	3

L_2 Items	support
{diaper, beer}	3
{diaper, cola}	3
{diaper, milk}	3
{cola, milk}	3

C_3 Items	support
{diaper, beer, cola}	2 ✗
{diaper, beer, milk}	2 ✗
{diaper, cola, milk}	3

L_3 Items	support
{diaper, cola, milk}	3

← Rule Generation

Association Rule	Support	confi.	%
diaper → cola ^ milk	2	3/4 = 0.75	75%
cola → diaper ^ milk	3	3/3 = 1	100%
milk → diaper ^ cola	3	3/4 = 0.75	75%
diaper ^ cola → milk	3	3/3 = 1	100%
diaper ^ milk → cola	3	3/3 = 1	100%
cola ^ milk → diaper	3	3/3 = 1	100%

Lab-8

Date
Page

FP - Growth Exemple

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 U Y
5	C E I K O

→ step 1 : freq Itemset min_support = 3

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

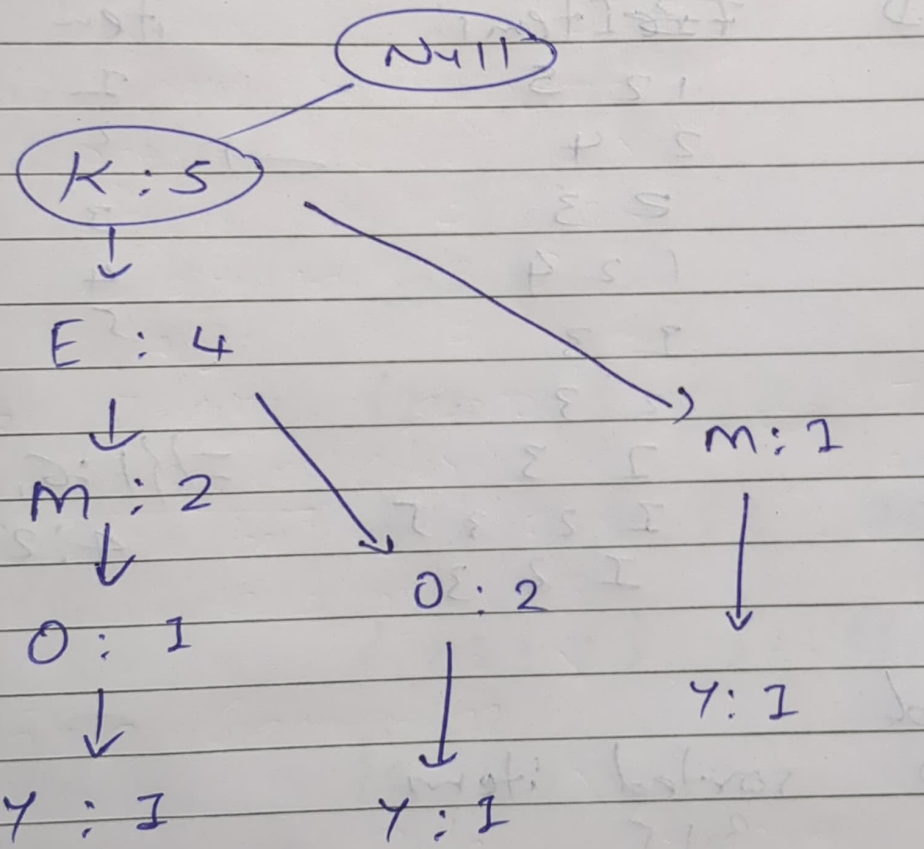
→ step 2 : transaction with item sorted based on frequency, and ignoring the infrequent items

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

~~I I~~

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

~) Conditional pattern Base



item	Conditional pattern Base	Fptree
Y	{K E M O : 1} {K E O : 1} {K M : 1}	{K : 3}
O	{K E M : 1} {K E : 2}	{K : 3, E : 3}
M	{K E : 2} {K : 1}	{K : 3}
E	{K : 4}	{K : 4}
K	—	

frequent pattern Gen

{K, Y:3}

{K, O:3} {E:O:3} {K, E, O:3}

{K, M:3}

{K, E:4}

Ex:2

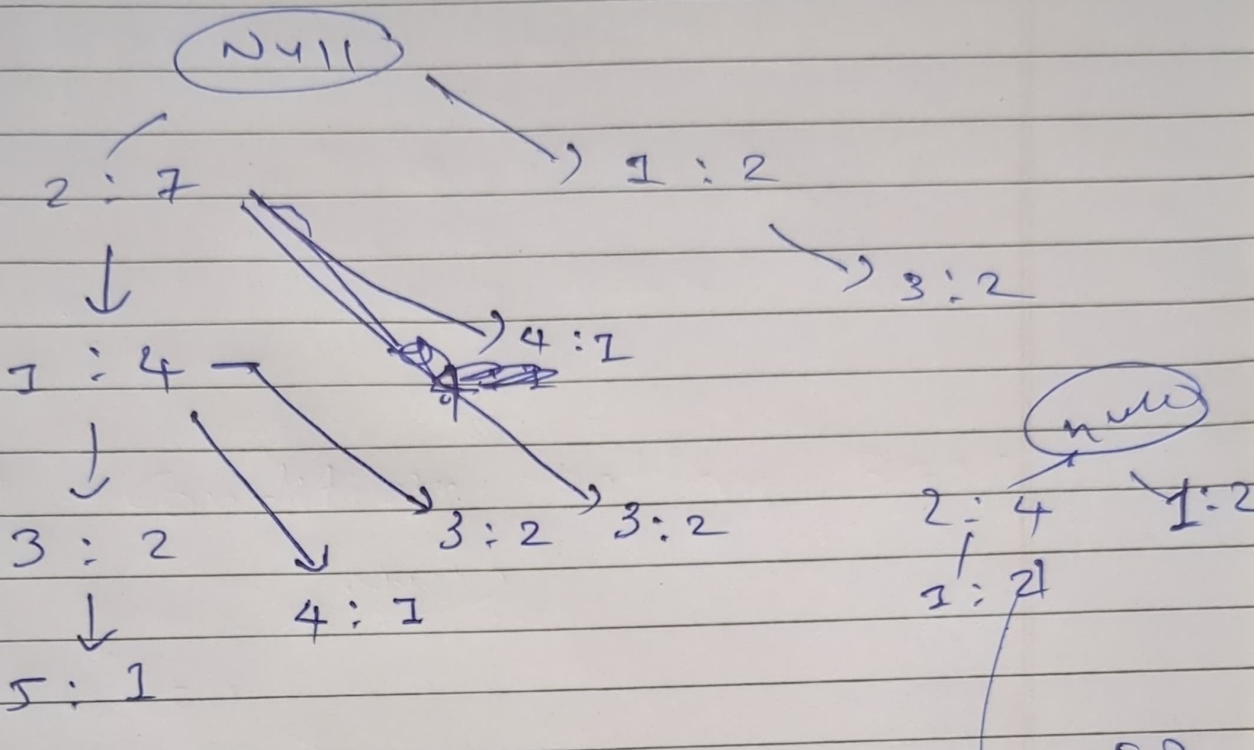
IID	Items
1	1 2 5
2	2 4
3	2 3
4	1 2 4
5	1 3
6	2 3
7	1 3
8	1 2 3 5
9	1 2 3

item	freq
1	6
2	7
3	6
4	2
5	2

=> {1:6, 2:7, 3:6, 4:2, 5:2}

=> sorted

IID	sorted item
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



item	condition	RP
5	{2 1 : 1}, {2 1 3 : 1}	{12 : 2}
4	{2 1 : 1}, {2 : 1}	{2 : 2}
3	{2 1 : 2}, {1 : 2}, {2 : 2}	{2 : 4} {1 : 2}
1	{2 : 4}	{2 : 4}

free.

{2, 5 : 2}, {1, 5 : 2}, {2, 1, 5 : 2}

{2, 4 : 2}

{2, 3 : 4}, {1, 3 : 2}

{2, 1 : 4}