PANDAS IN PYTHON

Data Science: is a branch of computer science where we study how to store, use and analyze data for deriving information from it.

Pandas is a Python library. Pandas is used to analyze data. Pandas is a Python library used for working with data sets. In computer programming, pandas is a software library written for python programming language for data manipulation and anlysis.in particular, it offers data structures and operations for manipulating numerical tables and time series.

Pandas main Data-structures

1. **Dataframes**
2. **Series**

It has functions for analyzing, cleaning, exploring, and manipulating data.

The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008. Why Use Pandas? Pandas allows us to analyze big data and make conclusions based on statistical theories.

Pandas can clean messy data sets, and make them readable and relevant.

Relevant data is very important in data science.

Pandas are also able to delete rows that are not relevant, or contains wrong values, like empty or NULL values. This is called cleaning the data.

Key Features of Pandas : 1) It has a fast and efficient DataFrame object with the default and customized indexing. 2) Used for reshaping and pivoting of the data sets. 3) Group by data for aggregations and transformations. It is used for data alignment and integration of the missing data. Provide the functionality of Time Series. Process a variety of data sets in different formats like matrix data, tabular heterogeneous, time series. Handle multiple operations of the data sets such as subsetting, slicing, filtering, groupBy, re-ordering, and re-shaping. It integrates with the other libraries such as SciPy, and scikit-learn. Provides fast performance, and If you want to speed it, even more, you can use the Cython.

In [1]:

```
# Library
import pandas as pd
```

Load the Dataset

In [2]:

```
df = pd.read_csv("services.csv")
```

In [3]:

```
df.head()
```

Out[3]:

	id	location_id	program_id	accepted_payments	alternate_name	application_process	audience	description	eligibility
0	1	1	NaN	NaN	NaN	Walk in or apply by phone.	Older adults age 55 or over, ethnic minorities...	A walk-in center for older adults that provide...	Age 55 or over for most programs, age 60 or ov...
									Age 55 or

	id	location_id	program_id	accepted_payments	alternate_name	application_process	audience	description	eligibility
1	2	2	NaN	NaN	NaN	Apply by phone for an appointment.	Residents of San Mateo County age 55 or over	Provides training and job placement to eligibl...	over, county resident and willing an...
2	3	3	NaN	NaN	NaN	Phone for information (403-4300 Ext. 4322).	Older adults age 55 or over who can benefit fr...	Offers supportive counseling services to San M...	Resident of San Mateo County age 55 or over
3	4	4	NaN	NaN	NaN	Apply by phone.	Parents, children, families with problems of c...	Provides supervised visitation services and a ...	None
4	5	5	NaN	NaN	NaN	Phone for information.	Low-income working families with children tran...	Provides fixed 8% short term loans to eligible...	Eligibility: Low-income family with legal cust...

5 rows × 22 columns

In [4]:

```
df.head(3)
```

Out[4]:

	id	location_id	program_id	accepted_payments	alternate_name	application_process	audience	description	eligibility
0	1	1	NaN	NaN	NaN	Walk in or apply by phone.	Older adults age 55 or over, ethnic minorities...	A walk-in center for older adults that provide...	Age 55 or over for most programs, age 60 or ov...
1	2	2	NaN	NaN	NaN	Apply by phone for an appointment.	Residents of San Mateo County age 55 or over	Provides training and job placement to eligibl...	Age 55 or over, county resident and willing an...
2	3	3	NaN	NaN	NaN	Phone for information (403-4300 Ext. 4322).	Older adults age 55 or over who can benefit fr...	Offers supportive counseling services to San M...	Resident of San Mateo County age 55 or over

3 rows × 22 columns

In [5]:

```
df.tail(3)
```

Out[5]:

	id	location_id	program_id	accepted_payments	alternate_name	application_process	audience	description	eligi
20	21	21	NaN	NaN	NaN	By phone during	NaN	just a test	

	id	location_id	program_id	accepted_payments	alternate_name	application_process	audience	description	eligi
20	21	21	NaN	NaN	NaN	business hours,	NaN	service	
21	22	22	NaN	Cash, Check, Credit Card	Fotos para pasaportes	Walk in or apply by phone or mail	Profit and nonprofit businesses, the public, m...	[NOTE THIS IS NOT A REAL SERVICE--THIS IS FOR ...	N
22	23	22	NaN	NaN	NaN	Walk in or apply by phone or mail	Second service and nonprofit businesses, the p...	[NOTE THIS IS NOT A REAL ORGANIZATION--THIS IS...	N

3 rows × 22 columns

In [6]:

```
type(df)
```

Out[6]:

```
pandas.core.frame.DataFrame
```

In [7]:

```
list(df.columns)
```

Out[7]:

```
['id',
 'location_id',
 'program_id',
 'accepted_payments',
 'alternate_name',
 'application_process',
 'audience',
 'description',
 'eligibility',
 'email',
 'fees',
 'funding_sources',
 'interpretation_services',
 'keywords',
 'languages',
 'name',
 'required_documents',
 'service_areas',
 'status',
 'wait_time',
 'website',
 'taxonomy_ids']
```

In [8]:

```
df['status']
```

Out[8]:

```
0       active
1       active
2       active
3       active
4       active
5       active
6       active
7       active
8       active
9       active
10      active
11      active
12      active
```

```
13       active
14       active
15       active
16       active
17       active
18       active
19      defunct
20     inactive
21       active
22       active
Name: status, dtype: object
```

In [9]:

```
type(df['status'])
```

Out[9]:

```
pandas.core.series.Series
```

In [10]:

```
list(df['status'])
```

Out[10]:

```
['active',
 'active',
 'active',
 'active',
 'active',
 'active',
 'active',
 'active',
 'active',
 'active',
 'active',
 'active',
 'active',
 'active',
 'active',
 'active',
 'active',
 'active',
 'active',
 'defunct',
 'inactive',
 'active',
 'active']
```

In [11]:

```
df[['status']]
```

Out[11]:

	status
0	active
1	active
2	active
3	active
4	active
5	active
6	active
7	active
8	active

9	status
10	active
11	active
12	active
13	active
14	active
15	active
16	active
17	active
18	active
19	defunct
20	inactive
21	active
22	active

In [12]:

```
type(df[['status']])
```

Out[12]:

```
pandas.core.frame.DataFrame
```

In [13]:

```
df.columns
```

Out[13]:

```
Index(['id', 'location_id', 'program_id', 'accepted_payments',
       'alternate_name', 'application_process', 'audience', 'description',
       'eligibility', 'email', 'fees', 'funding_sources',
       'interpretation_services', 'keywords', 'languages', 'name',
       'required_documents', 'service_areas', 'status', 'wait_time', 'website',
       'taxonomy_ids'],
      dtype='object')
```

In [14]:

```
df[['email','keywords','name']]
```

Out[14]:

	email	keywords	name
0	NaN	ADULT PROTECTION AND CARE SERVICES, Meal Sites...	Fair Oaks Adult Activity Center
1	NaN	EMPLOYMENT/TRAINING SERVICES, Job Development,...	Second Career Employment Program
2	NaN	Geriatric Counseling, Older Adults, Gay, Lesbi...	Senior Peer Counseling
3	NaN	INDIVIDUAL AND FAMILY DEVELOPMENT SERVICES, Gr...	Family Visitation Center
4	NaN	COMMUNITY SERVICES, Speakers, Automobile Loans	Economic Self-Sufficiency Program
5	NaN	ADULT PROTECTION AND CARE SERVICES, In-Home Su...	Little House Recreational Activities
6	NaN	ADULT PROTECTION AND CARE SERVICES, Adult Day ...	Rosener House Adult Day Services
7	NaN	ADULT PROTECTION AND CARE SERVICES, Meal Sites...	Meals on Wheels - South County
8	NaN	EDUCATION SERVICES, Library, Libraries, Public...	Fair Oaks Branch
9	NaN	EDUCATION SERVICES, Library, Libraries, Public...	Main Library
10	NaN	EDUCATION SERVICES, Library, Libraries, Public...	Schaberg Branch
11	NaN	EDUCATION SERVICES, Adult, Alternative, Litera...	Project Read

	email	keywords	name
12	NaN	EDUCATION SERVICES, Library, Libraries, Public	Redwood Shores Branch
13	NaN	COMMUNITY SERVICES, Interpretation/Translation...	Redwood City Corps
14	NaN	ALCOHOLISM SERVICES, Residential Care, DRUG AB...	Adult Rehabilitation Center
15	NaN	COMMODITY SERVICES, Clothing/Personal Items, C...	Sunnyvale Corps
16	NaN	COMMODITY SERVICES, Clothing/Personal Items, C...	South San Francisco Citadel Corps
17	NaN	HEALTH SERVICES, Outpatient Care, Community Cl...	Project Smile
18	NaN	HEALTH SERVICES, Outpatient Care, Community Cl...	San Mateo Free Medical Clinic
19	NaN	NaN	Service with blank fields
20	NaN	NaN	Service for Admin Test Location
21	passports@example.org	Salud, Medicina	Passport Photos
22	NaN	Ruby on Rails/Postgres/Redis, testing, wic	Example Service Name

In [15]:

```
df.dtypes
```

Out[15]:

```
id                           int64
location_id                  int64
program_id                 float64
accepted_payments           object
alternate_name              object
application_process         object
audience                    object
description                 object
eligibility                 object
email                       object
fees                        object
funding_sources             object
interpretation_services     object
keywords                    object
languages                   object
name                        object
required_documents          object
service_areas               object
status                      object
wait_time                   object
website                     object
taxonomy_ids                object
dtype: object
```

In [16]:

```
df1 = pd.read_excel("LUSID Excel - Setting up your market data.xlsx")
```

In [17]:

```
type(df1)
```

Out[17]:

```
pandas.core.frame.DataFrame
```

In [18]:

```
df1.dtypes
```

Out[18]:

```
Unnamed: 0    float64
Unnamed: 1    float64
Unnamed: 2    float64
Unnamed: 3     object
Unnamed: 4     object
Unnamed: 5     object
Unnamed: 6    float64
```

```
Unnamed: 6     float64
Unnamed: 7      object
Unnamed: 8      object
Unnamed: 9      object
dtype: object
```

In [19]:

```
df1.columns
```

Out[19]:

```
Index(['Unnamed: 0', 'Unnamed: 1', 'Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4',
       'Unnamed: 5', 'Unnamed: 6', 'Unnamed: 7', 'Unnamed: 8', 'Unnamed: 9'],
      dtype='object')
```

In [20]:

```
df1[['Unnamed: 6','Unnamed: 8']]
```

Out[20]:

	Unnamed: 6	Unnamed: 8
0	NaN	NaN
1	NaN	NaN
2	NaN	NaN
3	NaN	NaN
4	NaN	NaN
5	NaN	NaN
6	NaN	NaN
7	NaN	NaN
8	NaN	NaN
9	NaN	NaN
10	NaN	NaN
11	NaN	NaN
12	NaN	NaN
13	NaN	NaN
14	NaN	NaN
15	NaN	"YYYY-MM-DDTHH:MM:SS.00Z"
16	NaN	NaN
17	NaN	2019-04-10T 13:30:45.55Z
18	NaN	NaN
19	NaN	2019-04-10T13:30:45+04:00
20	NaN	NaN
21	NaN	2019-04-10NSingaporeClose
22	NaN	NaN
23	NaN	NaN
24	NaN	NaN
25	NaN	NaN
26	NaN	NaN
27	NaN	NaN

In [21]:

```
df2 = pd.read_csv("https://raw.githubusercontent.com/datasciencedojo/datasets/master/tita
nic.csv")
```

In [22]:

```
df2.head(3)
```

Out[22]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S

In [23]:

```
df2.columns
```

Out[23]:

```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

In [24]:

```
type(df2)
```

Out[24]:

```
pandas.core.frame.DataFrame
```

In [25]:

```
df2['Survived']
```

Out[25]:

```
0      0
1      1
2      1
3      1
4      0
      ..
886    0
887    1
888    0
889    1
890    0
Name: Survived, Length: 891, dtype: int64
```

In [26]:

```
type(df2['Survived'])
```

Out[26]:

```
pandas.core.series.Series
```

In [27]:

```
df2[['Survived', 'Pclass', 'Name']]
```

Out[27]:

	Survived	Pclass	Name
0	0	3	Braund, Mr. Owen Harris

	Survived	Pclass	Name
1	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...
2	1	3	Heikkinen, Miss. Laina
3	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)
4	0	3	Allen, Mr. William Henry
...	...	...	...
886	0	2	Montvila, Rev. Juozas
887	1	1	Graham, Miss. Margaret Edith
888	0	3	Johnston, Miss. Catherine Helen "Carrie"
889	1	1	Behr, Mr. Karl Howell
890	0	3	Dooley, Mr. Patrick

891 rows × 3 columns

In [28]:

```
df2.tail(3)
```

Out[28]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.45	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.00	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.75	NaN	Q

In [29]:

```
import lxml
import pandas as pd
url_data = pd.read_html("https://www.basketball-reference.com/leagues/NBA_2015_totals.htm
l")
```

In [30]:

```
pip install lxml
```

```
Requirement already satisfied: lxml in c:\users\dipmani\anaconda3\lib\site-packages (4.9.
1)
Note: you may need to restart the kernel to use updated packages.
```

In [31]:

```
type(url_data)
```

Out[31]:

```
list
```

In [32]:

```
len(url_data)
```

Out[32]:

```
1
```

In [33]:

```
df3 =url_data[0]
```

In [34]:

```
df3
```

Out[34]:

	Rk	Player	Pos	Age	Tm	G	GS	MP	FG	FGA	...	FT%	ORB	DRB	TRB	AST	STL	BLK	TOV	PF	PTS
0	1	Quincy Acy	PF	24	NYK	68	22	1287	152	331	...	.784	79	222	301	68	27	22	60	147	398
1	2	Jordan Adams	SG	20	MEM	30	0	248	35	86	...	.609	9	19	28	16	16	7	14	24	94
2	3	Steven Adams	C	21	OKC	70	67	1771	217	399	...	.502	199	324	523	66	38	86	99	222	537
3	4	Jeff Adrien	PF	28	MIN	17	0	215	19	44	...	.579	23	54	77	15	4	9	9	30	60
4	5	Arron Afflalo	SG	29	TOT	78	72	2502	375	884	...	.843	27	220	247	129	41	7	116	167	1035
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
670	490	Thaddeus Young	PF	26	TOT	76	68	2434	451	968	...	.655	127	284	411	173	124	25	117	171	1071
671	490	Thaddeus Young	PF	26	MIN	48	48	1605	289	641	...	.682	75	170	245	135	86	17	75	115	685
672	490	Thaddeus Young	PF	26	BRK	28	20	829	162	327	...	.606	52	114	166	38	38	8	42	56	386
673	491	Cody Zeller	C	22	CHO	62	45	1487	172	373	...	.774	97	265	362	100	34	49	62	156	472
674	492	Tyler Zeller	C	25	BOS	82	59	1731	340	619	...	.823	146	319	465	113	18	52	76	205	833

675 rows × 30 columns

In [35]:

```
df3.head()
```

Out[35]:

	Rk	Player	Pos	Age	Tm	G	GS	MP	FG	FGA	...	FT%	ORB	DRB	TRB	AST	STL	BLK	TOV	PF	PTS
0	1	Quincy Acy	PF	24	NYK	68	22	1287	152	331	...	.784	79	222	301	68	27	22	60	147	398
1	2	Jordan Adams	SG	20	MEM	30	0	248	35	86	...	.609	9	19	28	16	16	7	14	24	94
2	3	Steven Adams	C	21	OKC	70	67	1771	217	399	...	.502	199	324	523	66	38	86	99	222	537
3	4	Jeff Adrien	PF	28	MIN	17	0	215	19	44	...	.579	23	54	77	15	4	9	9	30	60
4	5	Arron Afflalo	SG	29	TOT	78	72	2502	375	884	...	.843	27	220	247	129	41	7	116	167	1035

5 rows × 30 columns

In [36]:

```
df3.tail(3)
```

Out[36]:

	Rk	Player	Pos	Age	Tm	G	GS	MP	FG	FGA	...	FT%	ORB	DRB	TRB	AST	STL	BLK	TOV	PF	PTS
672	490	Thaddeus Young	PF	26	BRK	28	20	829	162	327	...	.606	52	114	166	38	38	8	42	56	386
673	491	Cody Zeller	C	22	CHO	62	45	1487	172	373	...	.774	97	265	362	100	34	49	62	156	472
674	492	Tyler Zeller	C	25	BOS	82	59	1731	340	619	...	.823	146	319	465	113	18	52	76	205	833

3 rows × 30 columns

In [37]:

```
df3.columns
```

Out[37]:

```
Index(['Rk', 'Player', 'Pos', 'Age', 'Tm', 'G', 'GS', 'MP', 'FG', 'FGA', 'FG%',
       '3P', '3PA', '3P%', '2P', '2PA', '2P%', 'eFG%', 'FT', 'FTA', 'FT%',
       'ORB', 'DRB', 'TRB', 'AST', 'STL', 'BLK', 'TOV', 'PF', 'PTS'],
      dtype='object')
```

In [38]:

```
df3.dtypes
```

Out[38]:

```
Rk        object
Player    object
Pos       object
Age       object
Tm        object
G         object
GS        object
MP        object
FG        object
FGA       object
FG%       object
3P        object
3PA       object
3P%       object
2P        object
2PA       object
2P%       object
eFG%      object
FT        object
FTA       object
FT%       object
ORB       object
DRB       object
TRB       object
AST       object
STL       object
BLK       object
TOV       object
PF        object
PTS       object
dtype: object
```

In [39]:

```
df3[['Pos', 'Age', 'Tm']]
```

Out[39]:

	Pos	Age	Tm
0	PF	24	NYK
1	SG	20	MEM
2	C	21	OKC
3	PF	28	MIN
4	SG	29	TOT
...	...	...	...
670	PF	26	TOT
671	PF	26	MIN
672	PF	26	BRK

	Pos	Age	Tm
673	C	22	CHO
674	C	25	BOS

675 rows × 3 columns

In [40]:

```
df3.to_csv("players_data.csv",index=False)
```

In [41]:

```
df = pd.read_csv("https://raw.githubusercontent.com/datasciencedojo/datasets/master/titan
ic.csv")
```

In [42]:

```
df.head()
```

Out[42]:

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

In [43]:

```
type(df)
```

Out[43]:

```
pandas.core.frame.DataFrame
```

In [44]:

```
df.dtypes
```

Out[44]:

```
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
```

In [45]:

```
df.describe()
```

Out[45]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [46]:

```
df[['Name','Sex','Ticket','Cabin','Embarked']]
```

Out[46]:

	Name	Sex	Ticket	Cabin	Embarked
0	Braund, Mr. Owen Harris	male	A/5 21171	NaN	S
1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	PC 17599	C85	C
2	Heikkinen, Miss. Laina	female	STON/O2. 3101282	NaN	S
3	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	113803	C123	S
4	Allen, Mr. William Henry	male	373450	NaN	S
...	...	...	...	...	...
886	Montvila, Rev. Juozas	male	211536	NaN	S
887	Graham, Miss. Margaret Edith	female	112053	B42	S
888	Johnston, Miss. Catherine Helen "Carrie"	female	W./C. 6607	NaN	S
889	Behr, Mr. Karl Howell	male	111369	C148	C
890	Dooley, Mr. Patrick	male	370376	NaN	Q

891 rows × 5 columns

In [47]:

```
df.dtypes == 'object'
```

Out[47]:

```
PassengerId    False
Survived       False
Pclass         False
Name            True
Sex             True
Age            False
SibSp          False
Parch          False
Ticket          True
Fare           False
Cabin           True
Embarked        True
dtype: bool
```

In [48]:

```
df.dtypes[df.dtypes == 'object'].index
```

Out[48]:

```
Index(['Name', 'Sex', 'Ticket', 'Cabin', 'Embarked'], dtype='object')
```

In [49]:

```
type(df.dtypes[df.dtypes == 'object'])
```

Out[49]:

```
pandas.core.series.Series
```

In [50]:

```
df[df.dtypes[df.dtypes == 'object'].index].describe()
```

Out[50]:

	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

In [51]:

```
df.dtypes
```

Out[51]:

```
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
```

In [52]:

```
df[df.dtypes[df.dtypes == 'float64'].index]
```

Out[52]:

	Age	Fare
0	22.0	7.2500
1	38.0	71.2833
2	26.0	7.9250
3	35.0	53.1000
4	35.0	8.0500
...	...	...
886	27.0	13.0000
887	19.0	30.0000
888	NaN	23.4500
889	26.0	30.0000
890	32.0	7.7500

891 rows × 2 columns

In [53]:

```
df[df.dtypes[df.dtypes == 'int64'].index]
```

Out[53]:

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

891 rows × 5 columns

In [54]:

```
df.columns
```

Out[54]:

```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

In [55]:

```
df[['Ticket','Cabin']][4:11:2]
```

Out[55]:

	Ticket	Cabin
4	373450	NaN
6	17463	E46
8	347742	NaN
10	PP 9549	G6

In [56]:

```
df['new_col'] = 0
```

In [57]:

```
df.head(3)
```

Out[57]:

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

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	new_col
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	0

In [58]:

```
pd.Categorical(df['Pclass'])
```

Out[58]:

```
[3, 1, 3, 1, 3, ..., 2, 1, 3, 1, 3]
Length: 891
Categories (3, int64): [1, 2, 3]
```

In [59]:

```
pd.Categorical(df['Survived'])
```

Out[59]:

```
[0, 1, 1, 1, 0, ..., 0, 1, 0, 1, 0]
Length: 891
Categories (2, int64): [0, 1]
```

In [60]:

```
df['Cabin'].unique()
```

Out[60]:

```
array([nan, 'C85', 'C123', 'E46', 'G6', 'C103', 'D56', 'A6',
       'C23 C25 C27', 'B78', 'D33', 'B30', 'C52', 'B28', 'C83', 'F33',
       'F G73', 'E31', 'A5', 'D10 D12', 'D26', 'C110', 'B58 B60', 'E101',
       'F E69', 'D47', 'B86', 'F2', 'C2', 'E33', 'B19', 'A7', 'C49', 'F4',
       'A32', 'B4', 'B80', 'A31', 'D36', 'D15', 'C93', 'C78', 'D35',
       'C87', 'B77', 'E67', 'B94', 'C125', 'C99', 'C118', 'D7', 'A19',
       'B49', 'D', 'C22 C26', 'C106', 'C65', 'E36', 'C54',
       'B57 B59 B63 B66', 'C7', 'E34', 'C32', 'B18', 'C124', 'C91', 'E40',
       'T', 'C128', 'D37', 'B35', 'E50', 'C82', 'B96 B98', 'E10', 'E44',
       'A34', 'C104', 'C111', 'C92', 'E38', 'D21', 'E12', 'E63', 'A14',
       'B37', 'C30', 'D20', 'B79', 'E25', 'D46', 'B73', 'C95', 'B38',
       'B39', 'B22', 'C86', 'C70', 'A16', 'C101', 'C68', 'A10', 'E68',
       'B41', 'A20', 'D19', 'D50', 'D9', 'A23', 'B50', 'A26', 'D48',
       'E58', 'C126', 'B71', 'B51 B53 B55', 'D49', 'B5', 'B20', 'F G63',
       'C62 C64', 'E24', 'C90', 'C45', 'E8', 'B101', 'D45', 'C46', 'D30',
       'E121', 'D11', 'E77', 'F38', 'B3', 'D6', 'B82 B84', 'D17', 'A36',
       'B102', 'B69', 'E49', 'C47', 'D28', 'E17', 'A24', 'C50', 'B42',
       'C148'], dtype=object)
```

In [61]:

```
df.head(2)
```

Out[61]:

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

In [62]:

```
len(df[df['Age'] >18])-891
```

Out[62]:

```
-316
```

In [63]:

```
df[df['Age'] >18].head()
```

Out[63]:

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

In [64]:

```
df[df['Fare'] > 32.204208].head()
```

Out[64]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	new_col
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	0
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S	0
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S	0
23	24	1	1	Sloper, Mr. William Thompson	male	28.0	0	0	113788	35.5000	A6	S	0
27	28	0	1	Fortune, Mr. Charles Alexander	male	19.0	3	2	19950	263.0000	C23 C25 C27	S	0

In [65]:

```
df[df['Fare'] ==0].head()
```

Out[65]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	new_col
179	180	0	3	Leonard, Mr. Lionel	male	36.0	0	0	LINE	0.0	NaN	S	0
263	264	0	1	Harrison, Mr. William	male	40.0	0	0	112059	0.0	B94	S	0
271	272	1	3	Tornquist, Mr. William Henry	male	25.0	0	0	LINE	0.0	NaN	S	0
277	278	0	2	Parkes, Mr. Francis "Frank"	male	NaN	0	0	239853	0.0	NaN	S	0

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	new_col
302	303	0	3	Johnson, Mr. William Cahoone Jr	male	19.0	0	0	LINE	0.0	NaN	S	0

In [66]:

```
df[df['Sex'] == 'male'].head(3)
```

Out[66]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	new_col
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S	0
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S	0
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q	0

In [67]:

```
df[df['Sex'] == 'female'].head(3)
```

Out[67]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	new_col
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	0
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	0
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S	0

In [68]:

```
df[df["Pclass"] == 1].head(3)
```

Out[68]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	new_col
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	0
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S	0
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S	0

In [69]:

```
df['Sex'] == 'felmale'
```

Out[69]:

```
0       False
1       False
2       False
3       False
4       False
```

```
        ...
886     False
887     False
888     False
889     False
890     False
Name: Sex, Length: 891, dtype: bool
```

In [70]:

```
df['Fare'] > 32
```

Out[70]:

```
0       False
1        True
2       False
3        True
4       False
        ...
886     False
887     False
888     False
889     False
890     False
Name: Fare, Length: 891, dtype: bool
```

In [71]:

```
df[(df['Sex'] == 'felmale')  & (df['Fare'] > 32 )]
```

Out[71]:

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	new_col

In [72]:

```
df[(df['Sex'] == 'felmale')  | (df['Fare'] > 32 )].head()
```

Out[72]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	new_col
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	0
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S	0
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S	0
23	24	1	1	Sloper, Mr. William Thompson	male	28.0	0	0	113788	35.5000	A6	S	0
27	28	0	1	Fortune, Mr. Charles Alexander	male	19.0	3	2	19950	263.0000	C23 C25 C27	S	0

In [73]:

```
df[(df['Sex'] == 'male')  & (df['Fare'] > 32 )].head()
```

Out[73]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	new_col

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	new_col
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S	0
23	24	1	1	Sloper, Mr. William Thompson	male	28.0	0	0	113788	35.5000	A6	S	0
27	28	0	1	Fortune, Mr. Charles Alexander	male	19.0	3	2	19950	263.0000	C23 C25 C27	S	0
34	35	0	1	Meyer, Mr. Edgar Joseph	male	28.0	1	0	PC 17604	82.1708	NaN	C	0
35	36	0	1	Holverson, Mr. Alexander Oskar	male	42.0	1	0	113789	52.0000	NaN	S	0

In [74]:

```
df[df['Fare'] == max(df['Fare'])]['Name']
```

Out[74]:

```
258                      Ward, Miss. Anna
679    Cardeza, Mr. Thomas Drake Martinez
737                Lesurer, Mr. Gustave J
Name: Name, dtype: object
```

In [75]:

```
df.head(3)
```

Out[75]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	new_col
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S	0
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	0
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	0

In [76]:

```
df[['PassengerId','Survived','Pclass']][0:2]
```

Out[76]:

	PassengerId	Survived	Pclass
0	1	0	3
1	2	1	1

In [77]:

```
df.iloc[0:2,[0,1,2]]
```

Out[77]:

	PassengerId	Survived	Pclass
0	1	0	3
1	2	1	1

In [78]:

```
df.loc[0:2,['PassengerId','Survived','Pclass']]
```

Out[78]:

	PassengerId	Survived	Pclass
0	1	0	3
1	2	1	1
2	3	1	3

In [79]:

```
df= pd.read_csv("https://raw.githubusercontent.com/datasciencedojo/datasets/master/titani
c.csv")
```

In [80]:

```
df.columns
```

Out[80]:

```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

In [81]:

```
s = df['Name'][0:10]
```

In [82]:

```
s
```

Out[82]:

```
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
5                                     Moran, Mr. James
6                              McCarthy, Mr. Timothy J
7                       Palsson, Master. Gosta Leonard
8    Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)
9                  Nasser, Mrs. Nicholas (Adele Achem)
Name: Name, dtype: object
```

In [83]:

```
len(s)
```

Out[83]:

```
10
```

In [84]:

```
type(s)
```

Out[84]:

```
pandas.core.series.Series
```

In [85]:

```
s
```

Out[85]:

```
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
5                                     Moran, Mr. James
6                              McCarthy, Mr. Timothy J
7                       Palsson, Master. Gosta Leonard
8    Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)
9                  Nasser, Mrs. Nicholas (Adele Achem)
Name: Name, dtype: object
```

In [86]:

```
l = ['dipak','b','c','d','e','f','g','h','i','j']
```

In [87]:

```
s1 =pd.Series(list(s),index=l)
```

In [88]:

```
s
```

Out[88]:

```
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
5                                     Moran, Mr. James
6                              McCarthy, Mr. Timothy J
7                       Palsson, Master. Gosta Leonard
8    Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)
9                  Nasser, Mrs. Nicholas (Adele Achem)
Name: Name, dtype: object
```

In [89]:

```
s1
```

Out[89]:

```
dipak                              Braund, Mr. Owen Harris
b        Cumings, Mrs. John Bradley (Florence Briggs Th...
c                                   Heikkinen, Miss. Laina
d             Futrelle, Mrs. Jacques Heath (Lily May Peel)
e                                 Allen, Mr. William Henry
f                                         Moran, Mr. James
g                                  McCarthy, Mr. Timothy J
h                           Palsson, Master. Gosta Leonard
i        Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)
j                      Nasser, Mrs. Nicholas (Adele Achem)
dtype: object
```

In [90]:

```
s[0]
```

Out[90]:

```
'Braund, Mr. Owen Harris'
```

In [91]:

```
s1[0]
```

Out[91]:

```
'Braund, Mr. Owen Harris'
```

In [92]:

```
s1["dipak"]
```

Out[92]:

```
'Braund, Mr. Owen Harris'
```

In [93]:

```
s2 = s1.append(s)
```

```
C:\Users\DIPMANI\AppData\Local\Temp\ipykernel_20328\2451741888.py:1: FutureWarning: The s
eries.append method is deprecated and will be removed from pandas in a future version. Us
e pandas.concat instead.
  s2 = s1.append(s)
```

In [94]:

```
s2
```

Out[94]:

```
dipak                                 Braund, Mr. Owen Harris
b        Cumings, Mrs. John Bradley (Florence Briggs Th...
c                                      Heikkinen, Miss. Laina
d             Futrelle, Mrs. Jacques Heath (Lily May Peel)
e                                    Allen, Mr. William Henry
f                                            Moran, Mr. James
g                                     McCarthy, Mr. Timothy J
h                              Palsson, Master. Gosta Leonard
i        Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)
j                      Nasser, Mrs. Nicholas (Adele Achem)
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
5                                            Moran, Mr. James
6                                     McCarthy, Mr. Timothy J
7                              Palsson, Master. Gosta Leonard
8        Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)
9                      Nasser, Mrs. Nicholas (Adele Achem)
dtype: object
```

In [95]:

```
s2[4]
```

Out[95]:

```
'Allen, Mr. William Henry'
```

In [96]:

```
s4 = pd.Series([3,4,5,6,6] , index=[2,4,5,6,1])
```

In [97]:

```
s5 = pd.Series([34,345,45,45,454] , index=[9,4,5,6,7])
```

In [98]:

```
s6 = s4.append(s5)
```

```
C:\Users\DIPMANI\AppData\Local\Temp\ipykernel_20328\937876903.py:1: FutureWarning: The se
ries.append method is deprecated and will be removed from pandas in a future version. Use
pandas.concat instead.
  s6 = s4.append(s5)
```

In [99]:

```
s6
```

Out[99]:

```
2        3
```

```
4       4
5       5
6       6
1       6
9      34
4     345
5      45
6      45
7     454
dtype: int64
```

In [100]:

```
s6[4]
```

Out[100]:

```
4       4
4     345
dtype: int64
```

In [101]:

```
s6[0:5]
```

Out[101]:

```
2     3
4     4
5     5
6     6
1     6
dtype: int64
```

In [102]:

```
s4
```

Out[102]:

```
2     3
4     4
5     5
6     6
1     6
dtype: int64
```

In [103]:

```
s5
```

Out[103]:

```
9      34
4     345
5      45
6      45
7     454
dtype: int64
```

In [104]:

```
s4*s5
```

Out[104]:

```
1        NaN
2        NaN
4     1380.0
5      225.0
6      270.0
7        NaN
9        NaN
```

```
dtype: float64
```

In [105]:

```
s4+s5
```

Out[105]:

```
1      NaN
2      NaN
4    349.0
5     50.0
6     51.0
7      NaN
9      NaN
dtype: float64
```

PART-4

In [106]:

```
df = pd.read_csv("https://raw.githubusercontent.com/datasciencedojo/datasets/master/titan
ic.csv")
```

In [107]:

```
df.head()
```

Out[107]:

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

In [108]:

```
df.drop('PassengerId',axis=1,inplace=True)
```

In [109]:

```
df.drop(3,inplace=True)
```

In [110]:

```
df.head(5)
```

Out[110]:

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

	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
5	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q

In [111]:

```
df.set_index("Name",inplace=True)
```

In [112]:

```
df.head()
```

Out[112]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
Name										
Braund, Mr. Owen Harris	0	3	male	22.0	1	0	A/5 21171	7.2500	NaN	S
Cumings, Mrs. John Bradley (Florence Briggs Thayer)	1	1	female	38.0	1	0	PC 17599	71.2833	C85	C
Heikkinen, Miss. Laina	1	3	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
Allen, Mr. William Henry	0	3	male	35.0	0	0	373450	8.0500	NaN	S
Moran, Mr. James	0	3	male	NaN	0	0	330877	8.4583	NaN	Q

In [113]:

```
df.reset_index().head()
```

Out[113]:

	Name	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	Braund, Mr. Owen Harris	0	3	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	1	female	38.0	1	0	PC 17599	71.2833	C85	C
2	Heikkinen, Miss. Laina	1	3	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	Allen, Mr. William Henry	0	3	male	35.0	0	0	373450	8.0500	NaN	S
4	Moran, Mr. James	0	3	male	NaN	0	0	330877	8.4583	NaN	Q

In [114]:

```
d = {'key1' :[3,4,5,6,7],
     'key2':[5,6,7,8,6],
     'key3':[4,5,6,7,8]
}
```

In [115]:

```
pd.DataFrame(d)
```

Out[115]:

	key1	key2	key3
0	3	5	4
1	4	6	5
2	5	7	6
3	6	8	7
4	7	6	8

In [116]:

```
df1 = pd.read_csv('taxonomy.csv')
```

In [117]:

```
df1.head()
```

Out[117]:

	taxonomy_id	name	parent_id	parent_name
0	101	Emergency	NaN	NaN
1	101-01	Disaster Response	101	Emergency
2	101-02	Emergency Cash	101	Emergency
3	101-02-01	Help Pay for Food	101-02	Emergency Cash
4	101-02-02	Help Pay for Healthcare	101-02	Emergency Cash

In [118]:

```
df1.dropna().head()
```

Out[118]:

	taxonomy_id	name	parent_id	parent_name
1	101-01	Disaster Response	101	Emergency
2	101-02	Emergency Cash	101	Emergency
3	101-02-01	Help Pay for Food	101-02	Emergency Cash
4	101-02-02	Help Pay for Healthcare	101-02	Emergency Cash
5	101-02-03	Help Pay for Housing	101-02	Emergency Cash

In [119]:

```
df1.dropna(inplace=True)
```

In [120]:

```
df1.head()
```

Out[120]:

	taxonomy_id	name	parent_id	parent_name
1	101-01	Disaster Response	101	Emergency
2	101-02	Emergency Cash	101	Emergency
3	101-02-01	Help Pay for Food	101-02	Emergency Cash
4	101-02-02	Help Pay for Healthcare	101-02	Emergency Cash
5	101-02-03	Help Pay for Housing	101-02	Emergency Cash

In [121]:

```
df1.dropna(axis=1)
```

Out[121]:

	taxonomy_id	name	parent_id	parent_name
1	101-01	Disaster Response	101	Emergency
2	101-02	Emergency Cash	101	Emergency
3	101-02-01	Help Pay for Food	101-02	Emergency Cash
4	101-02-02	Help Pay for Healthcare	101-02	Emergency Cash
5	101-02-03	Help Pay for Housing	101-02	Emergency Cash

	taxonomy_id	name	parent_id	parent_name
3	101-02-03	Help Pay for Housing	101-02	Emergency Cash
...	...	...	...	...
285	111-01-07	Workplace Rights	111-01	Advocacy & Legal Aid
286	111-02	Mediation	111	Legal
287	111-03	Notary	111	Legal
288	111-04	Representation	111	Legal
289	111-05	Translation & Interpretation	111	Legal

279 rows × 4 columns

In [122]:

```
df2 = pd.read_csv('taxonomy.csv')
```

In [123]:

```
df2.head(2)
```

Out[123]:

	taxonomy_id	name	parent_id	parent_name
0	101	Emergency	NaN	NaN
1	101-01	Disaster Response	101	Emergency

In [124]:

```
df2.dropna(axis=1).head(4)
```

Out[124]:

	taxonomy_id	name
0	101	Emergency
1	101-01	Disaster Response
2	101-02	Emergency Cash
3	101-02-01	Help Pay for Food

In [125]:

```
df2.fillna("dipak").head(2)
```

Out[125]:

	taxonomy_id	name	parent_id	parent_name
0	101	Emergency	dipak	dipak
1	101-01	Disaster Response	101	Emergency

In [126]:

```
df.reset_index(inplace=True)
```

In [127]:

```
df.head()
```

Out[127]:

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

(Florence Briggs Th...

	Name	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
2	Heikkinen, Miss. Laina	1	3	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	Allen, Mr. William Henry	0	3	male	35.0	0	0	373450	8.0500	NaN	S
4	Moran, Mr. James	0	3	male	NaN	0	0	330877	8.4583	NaN	Q

In [128]:

```
g = df.groupby('Survived')
```

In [129]:

```
g
```

Out[129]:

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x000001727817E2E0>
```

In [130]:

```
g.sum()
```

Out[130]:

	Pclass	Age	SibSp	Parch	Fare
Survived					
0	1390	12985.50	304	181	12142.7199
1	666	8184.67	161	159	16498.1294

In [131]:

```
g.mean()
```

Out[131]:

	Pclass	Age	SibSp	Parch	Fare
Survived					
0	2.531876	30.626179	0.553734	0.329690	22.117887
1	1.953079	28.320657	0.472141	0.466276	48.381611

In [132]:

```
g1 = df.groupby('Pclass')
```

In [133]:

```
g1.sum()
```

Out[133]:

	Survived	Age	SibSp	Parch	Fare
Pclass					
1	135	7076.42	89	77	18124.3125
2	87	5168.83	74	70	3801.8417
3	119	8924.92	302	193	6714.6951

In [134]:

```
g1.mean()
```

Out[134]:

	Survived	Age	SibSp	Parch	Fare
Pclass					
1	0.627907	38.250919	0.413953	0.358140	84.299128
2	0.472826	29.877630	0.402174	0.380435	20.662183
3	0.242363	25.140620	0.615071	0.393075	13.675550

In [135]:

```
g1.max().T
```

```
C:\Users\DIPMANI\AppData\Local\Temp\ipykernel_20328\2755232466.py:1: FutureWarning: Dropp
ing invalid columns in DataFrameGroupBy.max is deprecated. In a future version, a TypeErr
or will be raised. Before calling .max, select only columns which should be valid for the
function.
  g1.max().T
```

Out[135]:

Pclass	1	2	3
Name	Young, Miss. Marie Grice	del Carlo, Mr. Sebastiano	van Melkebeke, Mr. Philemon
Survived	1	1	1
Sex	male	male	male
Age	80.0	70.0	74.0
SibSp	3	3	8
Parch	4	3	6
Ticket	WE/P 5735	W/C 14208	W./C. 6609
Fare	512.3292	73.5	69.55

In [136]:

```
df.head(3)
```

Out[136]:

	Name	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	Braund, Mr. Owen Harris	0	3	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	1	female	38.0	1	0	PC 17599	71.2833	C85	C
2	Heikkinen, Miss. Laina	1	3	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S

In [137]:

```
df5 = df[['Name', 'Survived', 'Pclass']][0:5]
```

In [138]:

```
df6 = df[['Name', 'Survived', 'Pclass']][5:10]
```

In [139]:

```
df5
```

Out[139]:

	Name	Survived	Pclass
0	Braund, Mr. Owen Harris	0	3
1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	1

	Name	Survived	Pclass
2	Heikkinen, Miss. Laina	1	3
3	Allen, Mr. William Henry	0	3
4	Moran, Mr. James	0	3

In [140]:

```
df6
```

Out[140]:

	Name	Survived	Pclass
5	McCarthy, Mr. Timothy J	0	1
6	Palsson, Master. Gosta Leonard	0	3
7	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	1	3
8	Nasser, Mrs. Nicholas (Adele Achem)	1	2
9	Sandstrom, Miss. Marguerite Rut	1	3

In [141]:

```
pd.concat([df5,df6])
```

Out[141]:

	Name	Survived	Pclass
0	Braund, Mr. Owen Harris	0	3
1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	1
2	Heikkinen, Miss. Laina	1	3
3	Allen, Mr. William Henry	0	3
4	Moran, Mr. James	0	3
5	McCarthy, Mr. Timothy J	0	1
6	Palsson, Master. Gosta Leonard	0	3
7	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	1	3
8	Nasser, Mrs. Nicholas (Adele Achem)	1	2
9	Sandstrom, Miss. Marguerite Rut	1	3

In [142]:

```
df7 = pd.concat([df5,df6],axis=1)
```

In [143]:

```
df7.fillna('dipak').head()
```

Out[143]:

	Name	Survived	Pclass	Name	Survived	Pclass
0	Braund, Mr. Owen Harris	0.0	3.0	dipak	dipak	dipak
1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1.0	1.0	dipak	dipak	dipak
2	Heikkinen, Miss. Laina	1.0	3.0	dipak	dipak	dipak
3	Allen, Mr. William Henry	0.0	3.0	dipak	dipak	dipak
4	Moran, Mr. James	0.0	3.0	dipak	dipak	dipak

In [144]:

```
data1 = pd.DataFrame({'key1':[1,2,4,5,6],
                            'key2':[4,5,6,7,8],
                            'key3':[3,4,5,6,6]
}
)
```

In [145]:

```
data1
```

Out[145]:

	key1	key2	key3
0	1	4	3
1	2	5	4
2	4	6	5
3	5	7	6
4	6	8	6

In [146]:

```
data2 = pd.DataFrame({'key1':[1,2,45,6,67],
                            'key4':[56,5,6,7,8],
                            'key5':[3,56,5,6,6]
}
)
```

In [147]:

```
data2
```

Out[147]:

	key1	key4	key5
0	1	56	3
1	2	5	56
2	45	6	5
3	6	7	6
4	67	8	6

In [148]:

```
pd.merge(data1,data2)
```

Out[148]:

	key1	key2	key3	key4	key5
0	1	4	3	56	3
1	2	5	4	5	56
2	6	8	6	7	6

In [149]:

```
pd.merge(data1,data2,how = 'left')
```

Out[149]:

	key1	key2	key3	key4	key5
0	1	4	3	56.0	3.0
1	2	5	4	5.0	56.0

	key1	key2	key3	key4	key5
2	4	6	5	NaN	NaN
3	5	7	6	NaN	NaN
4	6	8	6	7.0	6.0

In [150]:

```
pd.merge(data1,data2,how = 'right')
```

Out[150]:

	key1	key2	key3	key4	key5
0	1	4.0	3.0	56	3
1	2	5.0	4.0	5	56
2	45	NaN	NaN	6	5
3	6	8.0	6.0	7	6
4	67	NaN	NaN	8	6

In [151]:

```
pd.merge(data1,data2,how = 'outer',on = 'key1')
```

Out[151]:

	key1	key2	key3	key4	key5
0	1	4.0	3.0	56.0	3.0
1	2	5.0	4.0	5.0	56.0
2	4	6.0	5.0	NaN	NaN
3	5	7.0	6.0	NaN	NaN
4	6	8.0	6.0	7.0	6.0
5	45	NaN	NaN	6.0	5.0
6	67	NaN	NaN	8.0	6.0

In [152]:

```
pd.merge(data1,data2,how = 'cross')
```

Out[152]:

	key1_x	key2	key3	key1_y	key4	key5
0	1	4	3	1	56	3
1	1	4	3	2	5	56
2	1	4	3	45	6	5
3	1	4	3	6	7	6
4	1	4	3	67	8	6
5	2	5	4	1	56	3
6	2	5	4	2	5	56
7	2	5	4	45	6	5
8	2	5	4	6	7	6
9	2	5	4	67	8	6
10	4	6	5	1	56	3
11	4	6	5	2	5	56
12	4	6	5	45	6	5
13	4	6	5	6	7	6

	key1_x	key2	key3	key1_y	key4	key5
14	4	6	5	67	8	6
15	5	7	6	1	56	3
16	5	7	6	2	5	56
17	5	7	6	45	6	5
18	5	7	6	6	7	6
19	5	7	6	67	8	6
20	6	8	6	1	56	3
21	6	8	6	2	5	56
22	6	8	6	45	6	5
23	6	8	6	6	7	6
24	6	8	6	67	8	6

In [153]:

```
data1 = pd.DataFrame({'key1':[1,2,4,5,6],
                      'key2':[4,5,6,7,8],
                      'key3':[3,4,5,6,6]},
                     index = ['a','b','c','d','e']
)
```

In [154]:

```
data2 = pd.DataFrame({'key11':[1,2,4,5,6],
                      'key22':[4,5,6,7,8],
                      'key33':[3,4,5,6,6]
},index=['a','b','h','i','j']
)
```

In [155]:

```
data1
```

Out[155]:

	key1	key2	key3
a	1	4	3
b	2	5	4
c	4	6	5
d	5	7	6
e	6	8	6

In [156]:

```
data2
```

Out[156]:

	key11	key22	key33
a	1	4	3
b	2	5	4
h	4	6	5
i	5	7	6
j	6	8	6

In [157]:

```
data1.join(data2)
```

```
data1.join(data2)
```

Out[157]:

	key1	key2	key3	key11	key22	key33
a	1	4	3	1.0	4.0	3.0
b	2	5	4	2.0	5.0	4.0
c	4	6	5	NaN	NaN	NaN
d	5	7	6	NaN	NaN	NaN
e	6	8	6	NaN	NaN	NaN

In [158]:

```
data1.join(data2,how='right')
```

Out[158]:

	key1	key2	key3	key11	key22	key33
a	1.0	4.0	3.0	1	4	3
b	2.0	5.0	4.0	2	5	4
h	NaN	NaN	NaN	4	6	5
i	NaN	NaN	NaN	5	7	6
j	NaN	NaN	NaN	6	8	6

In [159]:

```
data1.join(data2,how='inner')
```

Out[159]:

	key1	key2	key3	key11	key22	key33
a	1	4	3	1	4	3
b	2	5	4	2	5	4

In [160]:

```
data1.join(data2,how='outer')
```

Out[160]:

	key1	key2	key3	key11	key22	key33
a	1.0	4.0	3.0	1.0	4.0	3.0
b	2.0	5.0	4.0	2.0	5.0	4.0
c	4.0	6.0	5.0	NaN	NaN	NaN
d	5.0	7.0	6.0	NaN	NaN	NaN
e	6.0	8.0	6.0	NaN	NaN	NaN
h	NaN	NaN	NaN	4.0	6.0	5.0
i	NaN	NaN	NaN	5.0	7.0	6.0
j	NaN	NaN	NaN	6.0	8.0	6.0

In [161]:

```
data1.join(data2,how='cross').head(10)
```

Out[161]:

key1	key2	key3	key11	key22	key33

	key1	key2	key3	key11	key22	key33
0	1	4	3	1	4	3
1	1	4	3	2	5	4
2	1	4	3	4	6	5
3	1	4	3	5	7	6
4	1	4	3	6	8	6
5	2	5	4	1	4	3
6	2	5	4	2	5	4
7	2	5	4	4	6	5
8	2	5	4	5	7	6
9	2	5	4	6	8	6

In [162]:

```
df.head(2)
```

Out[162]:

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

In [163]:

```
df['Fare_INR'] = df['Fare'].apply(lambda x : x*80)
```

In [164]:

```
df.head(3)
```

Out[164]:

	Name	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Fare_INR
0	Braund, Mr. Owen Harris	0	3	male	22.0	1	0	A/5 21171	7.2500	NaN	S	580.000
1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	1	female	38.0	1	0	PC 17599	71.2833	C85	C	5702.664
2	Heikkinen, Miss. Laina	1	3	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	634.000

In [165]:

```
def euro_inr(x):
    return x*80

df['Fare_INR'] = df['Fare'].apply(euro_inr)
```

In [166]:

```
df.head()
```

Out[166]:

	Name	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Fare_INR
0	Braund, Mr. Owen Harris	0	3	male	22.0	1	0	A/5 21171	7.2500	NaN	S	580.000
1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	1	female	38.0	1	0	PC 17599	71.2833	C85	C	5702.664

	Name	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Fare_INR
2	Heikkinen, Miss. Laina	1	3	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	634.000
3	Allen, Mr. William Henry	0	3	male	35.0	0	0	373450	8.0500	NaN	S	644.000
4	Moran, Mr. James	0	3	male	NaN	0	0	330877	8.4583	NaN	Q	676.664

In [167]:

```
df['name_len'] = df['Name'].apply(len)
```

In [168]:

```
df.head(3)
```

Out[168]:

	Name	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Fare_INR	name_len
0	Braund, Mr. Owen Harris	0	3	male	22.0	1	0	A/5 21171	7.2500	NaN	S	580.000	23
1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	1	female	38.0	1	0	PC 17599	71.2833	C85	C	5702.664	51
2	Heikkinen, Miss. Laina	1	3	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	634.000	22

In [169]:

```
def cat_fare(x):
    if x<10 :
        return "cheap"
    elif x>=10 and x<20:
        return 'mid'
    else :
        return 'high'
```

In [170]:

```
df['car_fare'] = df['Fare'].apply(cat_fare)
```

In [171]:

```
df.head(5)
```

Out[171]:

	Name	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Fare_INR	name_len	car_
0	Braund, Mr. Owen Harris	0	3	male	22.0	1	0	A/5 21171	7.2500	NaN	S	580.000	23	ch
1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	1	female	38.0	1	0	PC 17599	71.2833	C85	C	5702.664	51	l
2	Heikkinen, Miss. Laina	1	3	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	634.000	22	ch
3	Allen, Mr. William Henry	0	3	male	35.0	0	0	373450	8.0500	NaN	S	644.000	24	ch
4	Moran, Mr. James	0	3	male	NaN	0	0	330877	8.4583	NaN	Q	676.664	16	ch

PART-5

In [172]:

```
import pandas as pd
```

In [173]:

```
data = {"a":[1,2,3,4],
       "b":[4,5,6,7],
       "c":["akash" , "vinay","hitesh","sanket"]}
```

In [174]:

```
df= pd.DataFrame(data)
```

In [175]:

```
df
```

Out[175]:

	a	b	c
0	1	4	akash
1	2	5	vinay
2	3	6	hitesh
3	4	7	sanket

In [176]:

```
df.set_index('a',inplace=True)
```

In [177]:

```
df
```

Out[177]:

	b	c
a		
1	4	akash
2	5	vinay
3	6	hitesh
4	7	sanket

In [178]:

```
df = df.reset_index()
```

In [179]:

```
df
```

Out[179]:

	a	b	c
0	1	4	akash
1	2	5	vinay
2	3	6	hitesh

2	3	6	hitesh
	a	b	c
~~3~~	~~4~~	~~7~~	~~sanket~~

In [180]:

```
data = {"a":[1,2,3,4],
        "b":[4,5,6,7],
        "c":["akash" , "vinay","hitesh","sanket"]}
df1 = pd.DataFrame(data,index = ['a','b','c','d'])
```

In [181]:

```
df1
```

Out[181]:

	a	b	c
a	1	4	akash
b	2	5	vinay
c	3	6	hitesh
d	4	7	sanket

In [182]:

```
for i,j in df1.iterrows():
    print( j)
```

```
a         1
b         4
c     akash
Name: a, dtype: object
a         2
b         5
c     vinay
Name: b, dtype: object
a          3
b          6
c     hitesh
Name: c, dtype: object
a          4
b          7
c     sanket
Name: d, dtype: object
```

In [183]:

```
df1
```

Out[183]:

	a	b	c
a	1	4	akash
b	2	5	vinay
c	3	6	hitesh
d	4	7	sanket

In [184]:

```
for col_name , column in df1.iteritems():
    print( col_name , column)
```

```
a a    1
b    2
c    3
d    4
```

```
Name: a, dtype: int64
b a    4
b    5
c    6
d    7
Name: b, dtype: int64
c a     akash
b     vinay
c    hitesh
d    sanket
Name: c, dtype: object
```

In [185]:

```
df1
```

Out[185]:

	a	b	c
a	1	4	akash
b	2	5	vinay
c	3	6	hitesh
d	4	7	sanket

In [186]:

```
list(df['a'])
```

Out[186]:

```
[1, 2, 3, 4]
```

In [187]:

```
[i for i in df['a']]
```

Out[187]:

```
[1, 2, 3, 4]
```

In [188]:

```
df1
```

Out[188]:

	a	b	c
a	1	4	akash
b	2	5	vinay
c	3	6	hitesh
d	4	7	sanket

In [189]:

```
def test(x):
    return x.sum()
df1.apply(test,axis=0)
```

Out[189]:

```
a                          10
b                          22
c    akashvinayhiteshsanket
dtype: object
```

In [190]:

```
df2 = df1[['a','b']]
```

In [191]:

```
df2
```

Out[191]:

	a	b
a	1	4
b	2	5
c	3	6
d	4	7

In [192]:

```
df2.applymap(lambda x : x**2)
```

Out[192]:

	a	b
a	1	16
b	4	25
c	9	36
d	16	49

In [193]:

```
df
```

Out[193]:

	a	b	c
0	1	4	akash
1	2	5	vinay
2	3	6	hitesh
3	4	7	sanket

In [194]:

```
df.sort_values('c')
```

Out[194]:

	a	b	c
0	1	4	akash
2	3	6	hitesh
3	4	7	sanket
1	2	5	vinay

In [195]:

```
df
```

Out[195]:

	a	b	c

	a	b	c
0	1	4	akash
1	2	5	vinay
2	3	6	hitesh
3	4	7	sanket

In [196]:

```
df.sort_index(ascending = False)
```

Out[196]:

	a	b	c
3	4	7	sanket
2	3	6	hitesh
1	2	5	vinay
0	1	4	akash

In [197]:

```
pd.set_option("display.max_colwidth" ,1000)
df3 = pd.DataFrame({"Desc" :["Data Science Masters course is highly curated and uniquely
designed according to the latest industry standards. This program instills students the s
kills essential to knowledge discovery efforts to identify standard, novel, and truly dif
ferentiated solutions and decision-making, including skills in managing, querying, analyz
ing, visualizing, and extracting meaning from extremely large data sets. This trending pr
ogram provides students with the statistical, mathematical and computational skills neede
d to meet the large-scale data science challenges of today's professional world. You will
learn all the stack required to work in data science industry including cloud infrastruct
ure and real-time industry projects. This course will be taught in Hindi language."] })
```

In [198]:

```
df3
```

Out[198]:

	Desc
0	Data Science Masters course is highly curated and uniquely designed according to the latest industry standards. This program instills students the skills essential to knowledge discovery efforts to identify standard, novel, and truly differentiated solutions and decision-making, including skills in managing, querying, analyzing, visualizing, and extracting meaning from extremely large data sets. This trending program provides students with the statistical, mathematical and computational skills needed to meet the large-scale data science challenges of today's professional world. You will learn all the stack required to work in data science industry including cloud infrastructure and real-time industry projects. This course will be taught in Hindi language.

In [199]:

```
pd.set_option("display.max_colwidth" ,1000)
df3 = pd.DataFrame({"Desc" :["Data Science Masters course is highly curated and uniquely
designed according to the latest industry standards. This program instills students the s
kills essential to knowledge discovery efforts to identify standard, novel, and truly dif
ferentiated solutions and decision-making, including skills in managing, querying, analyz
ing, visualizing, and extracting meaning from extremely large data sets. This trending pr
ogram provides students with the statistical, mathematical and computational skills neede
d to meet the large-scale data science challenges of today's professional world. You will
learn all the stack required to work in data science industry including cloud infrastruct
ure and real-time industry projects. This course will be taught in Hindi language." , "my
name is sudh" ,"i use to teach data science "] })
```

In [200]:

```
df3
```

Out[200]:

Desc

	Desc
0	Data Science Masters course is highly curated and uniquely designed according to the latest industry standards. This program instills students the skills essential to knowledge discovery efforts to identify standard, novel, and truly differentiated solutions and decision-making, including skills in managing, querying, analyzing, visualizing, and extracting meaning from extremely large data sets. This trending program provides students with the statistical, mathematical and computational skills needed to meet the large-scale data science challenges of today's professional world. You will learn all the stack required to work in data science industry including cloud infrastructure and real-time industry projects. This course will be taught in Hindi language.
1	my name is sudh
2	i use to teach data science

In [201]:

```
df3['len'] = df3['Desc'].apply(len)
```

In [202]:

```
df3
```

Out[202]:

	Desc	len
0	Data Science Masters course is highly curated and uniquely designed according to the latest industry standards. This program instills students the skills essential to knowledge discovery efforts to identify standard, novel, and truly differentiated solutions and decision-making, including skills in managing, querying, analyzing, visualizing, and extracting meaning from extremely large data sets. This trending program provides students with the statistical, mathematical and computational skills needed to meet the large-scale data science challenges of today's professional world. You will learn all the stack required to work in data science industry including cloud infrastructure and real-time industry projects. This course will be taught in Hindi language.	765
1	my name is sudh	15
2	i use to teach data science	28

In [203]:

```
t ="i use to teach data science "
len(t.split())
```

Out[203]:

```
6
```

In [204]:

```
df3['word_count'] = df3['Desc'].apply(lambda x :len(x.split()))
```

In [205]:

```
df3
```

Out[205]:

	Desc	len	word_count
0	Data Science Masters course is highly curated and uniquely designed according to the latest industry standards. This program instills students the skills essential to knowledge discovery efforts to identify standard, novel, and truly differentiated solutions and decision-making, including skills in managing, querying, analyzing, visualizing, and extracting meaning from extremely large data sets. This trending program provides students with the statistical, mathematical and computational skills needed to meet the large-scale data science challenges of today's professional world. You will learn all the stack required to work in data science industry including cloud infrastructure and real-time industry projects. This course will be taught in Hindi language.	765	104
1	my name is sudh	15	4
2	i use to teach data science	28	6

In [206]:

```
df
```

Out[206]:

	a	b	c
0	1	4	akash
1	2	5	vinay
2	3	6	hitesh
3	4	7	sanket

In [207]:

```
df['a'][0]
```

Out[207]:

```
1
```

In [208]:

```
df['a'].mean()
```

Out[208]:

```
2.5
```

In [209]:

```
df['a'].median()
```

Out[209]:

```
2.5
```

In [210]:

```
df['a'].mode()
```

Out[210]:

```
0    1
1    2
2    3
3    4
Name: a, dtype: int64
```

In [211]:

```
df['a'].std()
```

Out[211]:

```
1.2909944487358056
```

In [212]:

```
df['a'].sum()
```

Out[212]:

```
10
```

In [213]:

```
df['a'].min()
```

Out[213]:

```
1
```

In [214]:

```
df['a'].max()
```

Out[214]:

```
4
```

In [215]:

```
df['a'].var()
```

Out[215]:

```
1.6666666666666667
```

In [216]:

```
#Python Pandas - Window Functions
df4 = pd.DataFrame({'a' : [3,4,5,2,1,3,4,5,6]})
```

In [217]:

```
df4
```

Out[217]:

	a
0	**3**
1	**4**
2	**5**
3	**2**
4	**1**
5	**3**
6	**4**
7	**5**
8	**6**

In [218]:

```
df4['a'].rolling(window=1).mean()
```

Out[218]:

```
0    3.0
1    4.0
2    5.0
3    2.0
4    1.0
5    3.0
6    4.0
7    5.0
8    6.0
Name: a, dtype: float64
```

In [219]:

```
df4['a'].rolling(window=2).mean()
```

Out[219]:

```
0    NaN
1    3.5
2    4.5
3    3.5
4    1.5
5    2.0
6    3.5
7    4.5
8    5.5
```

```
Name: a, dtype: float64
```

In [220]:

```
df4['a'].rolling(window=3).mean()
```

Out[220]:

```
0         NaN
1         NaN
2    4.000000
3    3.666667
4    2.666667
5    2.000000
6    2.666667
7    4.000000
8    5.000000
Name: a, dtype: float64
```

In [221]:

```
df4
```

Out[221]:

	a
0	3
1	4
2	5
3	2
4	1
5	3
6	4
7	5
8	6

In [222]:

```
df4['a'].rolling(window=3).sum()
```

Out[222]:

```
0     NaN
1     NaN
2    12.0
3    11.0
4     8.0
5     6.0
6     8.0
7    12.0
8    15.0
Name: a, dtype: float64
```

In [223]:

```
df4
```

Out[223]:

	a
0	3
1	4
2	5
3	2

	a
3	2
4	1
5	3
6	4
7	5
8	6

In [224]:

```
df4['a'].rolling(window=3).min()
```

Out[224]:

```
0     NaN
1     NaN
2     3.0
3     2.0
4     1.0
5     1.0
6     1.0
7     3.0
8     4.0
Name: a, dtype: float64
```

In [225]:

```
df4['a'].rolling(window=3).max()
```

Out[225]:

```
0     NaN
1     NaN
2     5.0
3     5.0
4     5.0
5     3.0
6     4.0
7     5.0
8     6.0
Name: a, dtype: float64
```

In [226]:

```
df4['a'].cumsum()
```

Out[226]:

```
0      3
1      7
2     12
3     14
4     15
5     18
6     22
7     27
8     33
Name: a, dtype: int64
```

In [227]:

```
df4
```

Out[227]:

	a
0	3
1	4
2	5

	a
3	2
4	1
5	3
6	4
7	5
8	6

In [228]:

```
#Python Pandas - Date Functionality

date = pd.date_range(start='2023-04-23' , end = '2023-06-23')
```

In [229]:

```
date
```

Out[229]:

```
DatetimeIndex(['2023-04-23', '2023-04-24', '2023-04-25', '2023-04-26',
               '2023-04-27', '2023-04-28', '2023-04-29', '2023-04-30',
               '2023-05-01', '2023-05-02', '2023-05-03', '2023-05-04',
               '2023-05-05', '2023-05-06', '2023-05-07', '2023-05-08',
               '2023-05-09', '2023-05-10', '2023-05-11', '2023-05-12',
               '2023-05-13', '2023-05-14', '2023-05-15', '2023-05-16',
               '2023-05-17', '2023-05-18', '2023-05-19', '2023-05-20',
               '2023-05-21', '2023-05-22', '2023-05-23', '2023-05-24',
               '2023-05-25', '2023-05-26', '2023-05-27', '2023-05-28',
               '2023-05-29', '2023-05-30', '2023-05-31', '2023-06-01',
               '2023-06-02', '2023-06-03', '2023-06-04', '2023-06-05',
               '2023-06-06', '2023-06-07', '2023-06-08', '2023-06-09',
               '2023-06-10', '2023-06-11', '2023-06-12', '2023-06-13',
               '2023-06-14', '2023-06-15', '2023-06-16', '2023-06-17',
               '2023-06-18', '2023-06-19', '2023-06-20', '2023-06-21',
               '2023-06-22', '2023-06-23'],
              dtype='datetime64[ns]', freq='D')
```

In [230]:

```
df_date = pd.DataFrame({'date':date})
```

In [231]:

```
df_date.dtypes
```

Out[231]:

```
date    datetime64[ns]
dtype: object
```

In [232]:

```
df_date
```

Out[232]:

	date
0	2023-04-23
1	2023-04-24
2	2023-04-25
3	2023-04-26
4	2023-04-27
...	...
57	2023-06-19

	date
57	2023-06-19
~~58~~	~~2023-06-20~~
59	2023-06-21
60	2023-06-22
61	2023-06-23

62 rows × 1 columns

In [233]:

```
df7 = pd.DataFrame({"date" : ['2023-06-23' , '2023-06-22','2023-06-20']})
```

In [234]:

```
df7
```

Out[234]:

	date
0	2023-06-23
1	2023-06-22
2	2023-06-20

In [235]:

```
df7.dtypes
```

Out[235]:

```
date     object
dtype: object
```

In [236]:

```
df7['updated_date'] = pd.to_datetime(df7['date'])
```

In [237]:

```
df7
```

Out[237]:

	date	updated_date
0	2023-06-23	2023-06-23
1	2023-06-22	2023-06-22
2	2023-06-20	2023-06-20

In [238]:

```
df7
```

Out[238]:

	date	updated_date
0	2023-06-23	2023-06-23
1	2023-06-22	2023-06-22
2	2023-06-20	2023-06-20

In [239]:

```
df7.dtypes
```

Out[239]:

```
date                      object
updated_date      datetime64[ns]
dtype: object
```

In [240]:

```
df7['year'] = df7['updated_date'].dt.year
```

In [241]:

```
df7
```

Out[241]:

	date	updated_date	year
0	2023-06-23	2023-06-23	2023
1	2023-06-22	2023-06-22	2023
2	2023-06-20	2023-06-20	2023

In [242]:

```
df7['day'] = df7['updated_date'].dt.day
```

In [243]:

```
df7
```

Out[243]:

	date	updated_date	year	day
0	2023-06-23	2023-06-23	2023	23
1	2023-06-22	2023-06-22	2023	22
2	2023-06-20	2023-06-20	2023	20

In [244]:

```
df7['month'] = df7['updated_date'].dt.month
```

In [245]:

```
df7
```

Out[245]:

	date	updated_date	year	day	month
0	2023-06-23	2023-06-23	2023	23	6
1	2023-06-22	2023-06-22	2023	22	6
2	2023-06-20	2023-06-20	2023	20	6

In [246]:

```
#Python Pandas -Time Delta
```

In [247]:

```
pd.Timedelta(days= 1,hours = 5 ,minutes = 45)
```

Out[247]:

```
Timedelta('1 days 05:45:00')
```

In [248]:

```
dt = pd.to_datetime('2023-06-20')
```

In [249]:

```
td = pd.Timedelta(days = 1 )
```

In [250]:

```
dt+td
```

Out[250]:

```
Timestamp('2023-06-21 00:00:00')
```

In [251]:

```
#Python Pandas - Categorical Data

data = ["akash" , "vinay" , "hitesh" , "navin","prakash" ,"sanket" ]
```

In [252]:

```
cat = pd.Categorical(data)
```

In [253]:

```
cat
```

Out[253]:

```
['akash', 'vinay', 'hitesh', 'navin', 'prakash', 'sanket']
Categories (6, object): ['akash', 'hitesh', 'navin', 'prakash', 'sanket', 'vinay']
```

In [254]:

```
cat.value_counts()
```

Out[254]:

```
akash      1
hitesh     1
navin      1
prakash    1
sanket     1
vinay      1
dtype: int64
```

In [255]:

```
#Python Pandas - Visualization
```

In [256]:

```
d = pd.Series([1,2,3,3,5,6,6,8])
d
```

Out[256]:

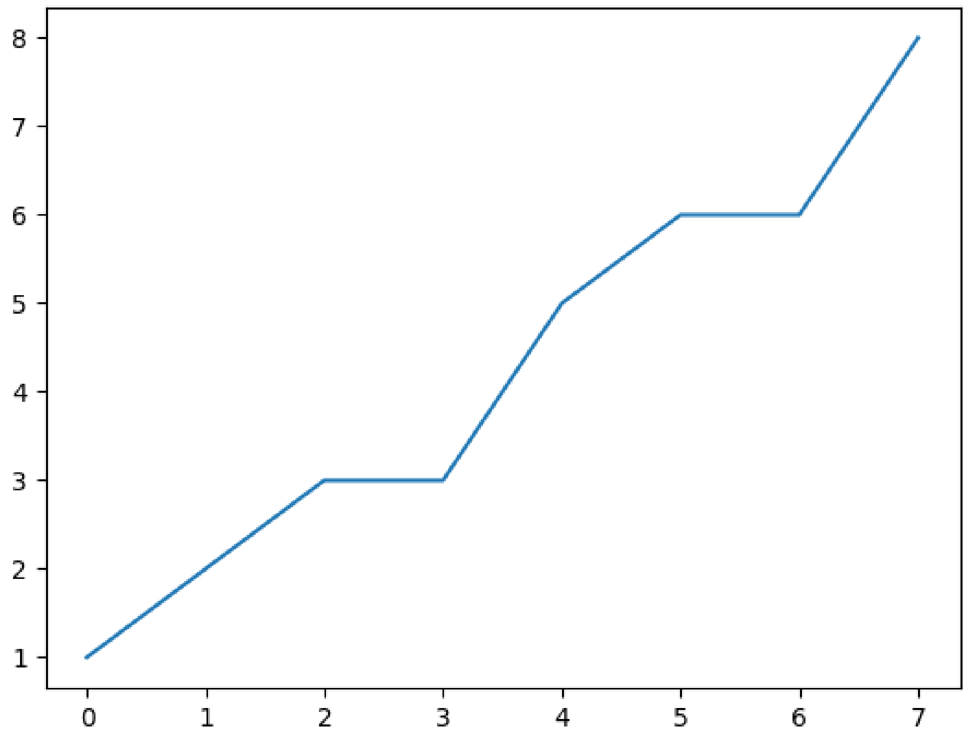
```
0    1
1    2
2    3
3    3
4    5
5    6
6    6
7    8
dtype: int64
```

In [257]:

```
d.plot()
```

Out[257]:

```
<AxesSubplot:>
```



In [258]:

```
df = pd.DataFrame({'a':[3,4,5,6,7],
                   'b':[4,5,6,7,8]})
```

In [259]:

```
df
```

Out[259]:

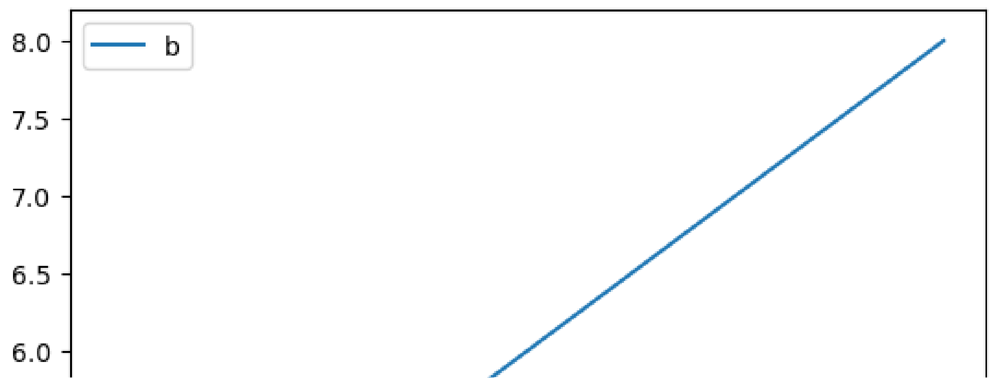
	a	b
0	3	4
1	4	5
2	5	6
3	6	7
4	7	8

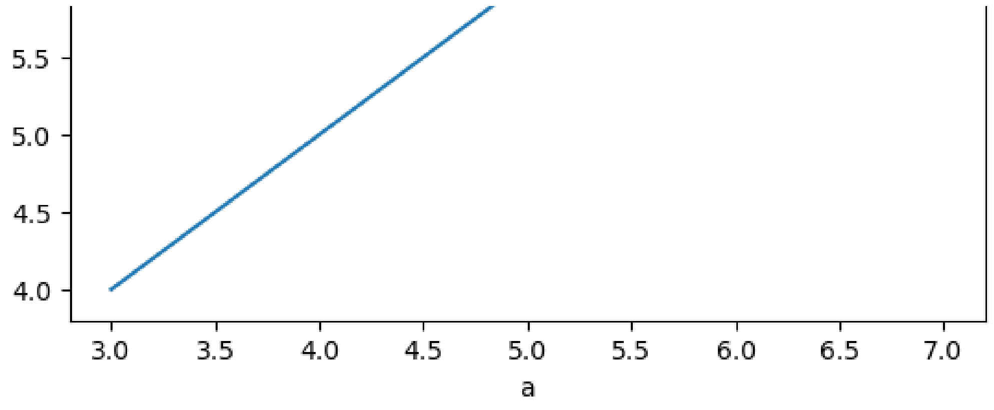
In [260]:

```
df.plot(x= 'a',y='b')
```

Out[260]:

```
<AxesSubplot:xlabel='a'>
```



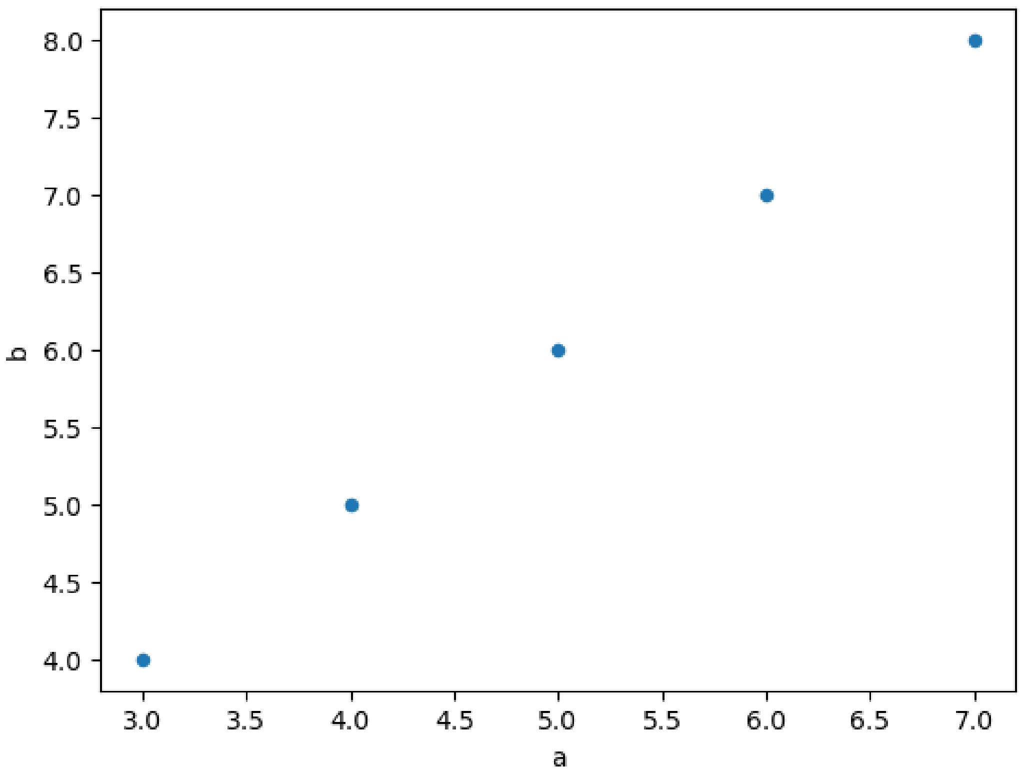


In [261]:

```
df.plot.scatter(x= 'a',y='b')
```

Out[261]:

```
<AxesSubplot:xlabel='a', ylabel='b'>
```



In [262]:

```
d = pd.Series([1,2,3,3,5,6,6,8])
d
```

Out[262]:

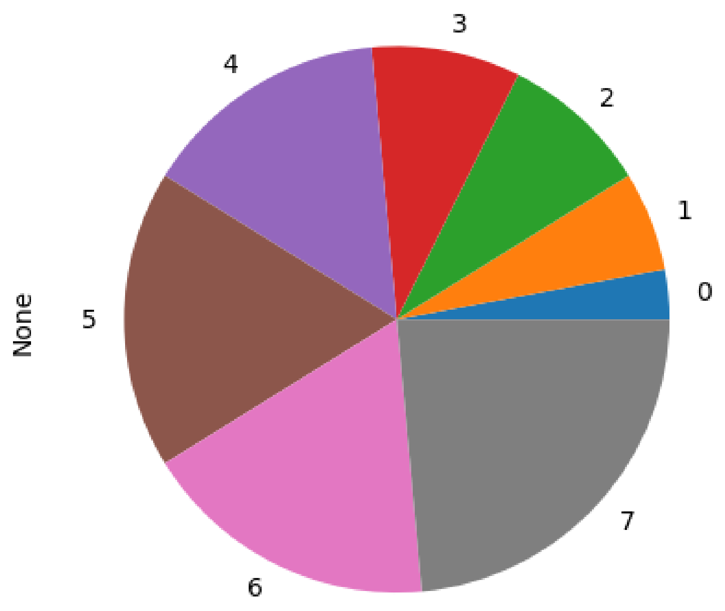
```
0    1
1    2
2    3
3    3
4    5
5    6
6    6
7    8
dtype: int64
```

In [263]:

```
d.plot.pie()
```

Out[263]:

```
<AxesSubplot:ylabel='None'>
```



```
In [ ]:
```