

### ➤ CSVFile

- A CSV is a comma-separated values file. This type of file can be view as an excel file and separated by commas. CSVfile is nothing more than a simple text file. However, it is the most common, simple and easiest method to store tabular data. This particular format arranges tables by a specific structure divided into rows and columns.
- Once we have the DataFrame, we can persist it in CSV on the local disk. Let's first create CSV file using data that is currently present in the DataFrame, we can store the data of this DataFrame in CSV format using API called `to_csv(...)` of Pandas

### Save data in csv

```
import pandas as pd

dic={'Name':['Ram','Shyam','Mohan','Sohan'],
     'Phy':[30,45,60,78],
     'Chemistry':[60,45,85,96],
     'Math':[80,85,87,90]
    }
df =pd.DataFrame(dic,index=['A','B','C','D'])
df.to_csv('D:/studet.csv',index=False)
print(df)
```

### Read data from csv

```
import pandas as pd
df=pd.read_csv('D:/studet.csv')
print(df)
```

### ➤ Iterating in Panda

#### ➤ sDataFrame

Iteration is a general term for taking each item of something. We can iterate an element in two ways:

(i) **Iterating over rows:** There are two functions to iterate over rows as follows:

- `iterrows()` : It returns the iterator yielding each index value along with a series containing the data in each row.
- `iteritems()`: It iterates over each column as key, value pair with label as key and column value as series object.

### Using iterrows function

```
import pandas as pd

dic={'Name':['Ram','Shyam','Mohan','Sohan'],
     'Phy':[30,45,60,78],
     'Chemistry':[60,45,85,96]
    }
df =pd.DataFrame(dic,index=['A','B','C','D'])
print(df)
```

```
for i,row in df.iterrows():
    if row['Phy']>50:
        print(row)
```

## Loc and iLoc Functions in Pandas

These are used in slicing data from the Pandas DataFrame. They are used in filtering the data according to some conditions.

loc in Pandas	iLoc in Pandas
Location (label-based-location) Label-based data selector	index-based-location Index-based data selector
loc gets rows (and/or columns) with particular <b>labels</b> .	iLoc gets rows (and/or columns) at integer <b>locations</b> .
Indices can be numerical or categorical	Indices should be numerical
The end index is included during slicing	The end index is excluded during slicing

### Use of Loc Functions

```
import pandas as pd

dic={'Name':['Ram','Shyam','Mohan','Sohan'],
     'Phy':[30,45,60,78],
     'Chemistry':[60,45,85,96],
     'Math':[80,85,87,90]
    }
df =pd.DataFrame(dic,index=['A','B','C','D'])

print(df.loc['B':'D'])
print(df.loc['B':'D','Name':'Chemistry'])

print(df.loc[['B','D']])
print(df.loc[['B','D'],['Name','Chemistry']])
```

### Use of iLoc Functions

```
import pandas as pd

dic={'Name':['Ram','Shyam','Mohan','Sohan'],
     'Phy':[30,45,60,78],
     'Chemistry':[60,45,85,96],
     'Math':[80,85,87,90]
    }
df =pd.DataFrame(dic,index=['A','B','C','D'])

print(df.iloc[1:4])
print(df.iloc[1:4,0:3])

print(df.iloc[[1,3]])
print(df.iloc[[1,3],[0,2]])
```

## Data Filtering using Condition

```
import pandas as pd

dic={'Name':['Ram','Shyam','Mohan','Sohan'],
     'Phy':[30,45,60,78],
     'Chemistry':[60,45,85,96],
     'Math':[80,85,87,90]
    }
df =pd.DataFrame(dic,index=['A','B','C','D'])

print(df[df['Name']=='Shyam'])
print(df[df['Phy']>= 70])
print(df[(df['Chemistry'] >= 70) & (df['Chemistry'] <= 90)])
```

## Student Data Filtering and Grading

```
df['Grade']="
df.loc[(df.Phy> 70), ['Grade']] = 'A'
print(df)
```

## Data Visualization

### Line Graph

```
import matplotlib.pyplot as plt
Name=['Diksha','Priya','Riya','Rohan']
English=[90,78,63,52]
plt.plot(Name,English,'r',marker='d',markersize=15,linestyle='dotted')
plt.xlabel('Name of students ')
plt.ylabel('English Marks ')
plt.title('Student Result ')
plt.show()
```

### Line Graph

```
import matplotlib.pyplot as plt
Name=['Priya','Riya','Rohan','Diksha']
Phy=[80,90,60,98]
che=[50,78,36,96]
plt.plot(Name,Phy,marker='d',markersize=15,linestyle='dotted',label='Physics')
plt.plot(Name,che,marker='D',markersize=10,linestyle='solid',label='Chemistry')

plt.yticks([0,20,40,60])
plt.grid(True)
plt.xlabel('Name of students ')
plt.ylabel('English Marks ')
plt.title('Student Result ')
plt.legend(loc='lower right')
plt.show()
```

## Bar Graph

```
import matplotlib.pyplot as plt

Name=['Priya','Riya' ,'Rohan','Diksha']
Phy=[80,90,60,98]
c=['red','green','blue','orange']
plt.bar(Name,Phy,color=c)
plt.xlabel('Name of students ')
plt.ylabel('Marks ')
plt.title('Student Result ')
plt.show()
```

## Bar Graph

```
import matplotlib.pyplot as plt

Roll=[1,2,3,4]
English=[40,50,40,36]

plt.barh(Roll,English,label='English')
plt.xlabel('Name of students ')
plt.ylabel('Marks ')
plt.title('Student Result ')
plt.legend(loc='upper left')
plt.savefig('D:/Result.png')
plt.show()
```

## Histogram

```
import matplotlib.pyplot as plt
marks=[50,45,85,96,85,75,52,55,27,25,26,33,80,90,60,98]
plt.hist(marks,bins=5)
plt.xlabel('marks of students')
plt.ylabel('Number of students')
plt.title('Result analysis')
plt.show()
```

## MYSQL

### Database:

A database is an organized collection of structured information, or data, typically stored electronically in a computer system.

**Table:** A table is a matrix with data. A table in a database looks like a simple spreadsheet.

**Column:**Field – Attribute :One column contains data of one and the same kind, for example the column Name, Roll.

**Row:** A row –tuple or record is a group of related data. For example, the data of one person, Thing.

**Degree** – Number of columns are known as Degree of the table.

**Cardinality:** Number of rows in a table is known as cardinality.

**Redundancy:** Storing data twice, redundantly to make the system faster.

Constraints:

Constraints are the condition which are defined when creating table and applied by the sql automatically when data is insert or update in a table to ensure validity of the data, called Constraints.

- Primary Key
- Not null
- Unique
- Default
- Foreign key

```
Create table student(  
  Rollno int Primary key,  
  Name varchar(25) Not Null,  
  Email varchar(25) unique,  
  Hobby varchar(25) default 'Music');
```

```
Foreign key: Create table exam (  
  Rollno integer,  
  phyint, che int,  
  math int,  
  Foreign key(Rollno) references student(Rollno));
```

**Primary Key:** In a primary key we cannot store null and duplicate value.

**Candidate Key:** Set of columns which uniquely define the values.

**Alternate Key:** One of candidate is selected as primary key remaining are known as alternate.

**Foreign Key:** in a foreign key column we can store only those value which are already stored in primary key this primary key and foreign key are define when creating table.

**Referential Integrity:** Referential Integrity makes sure that a foreign key value always points to an existing row.

## SQL Commands

### DDL-

- ❖ Create – Database , Table
- ❖ Alter – Column – Add, Drop, Modify
- ❖ Drop – Database , Table

### DML

- ❖ Insert into –
  - Value for all column
  - Value for limited column
  - Multiple Record at a time
- ❖ Delete
  - Delete All Records
  - Delete on specific record
- ❖ Update
  - Update specific column value for one record
  - Update a column value
  - Update Multiple columns for one or more record
- ❖ Select - distinct , order by
  - Where Clause
  - And, Or, in, between, like, not like, is null, is not null, Group by Having

### Create database JVP;

Query OK, 1 row affected

### use JVP;

Database changed

### Create table student(

- > Rollno Integer Primary Key,
- > Aadhar Varchar(12) Unique,
- > Name varchar(30) Not Null,
- > Percentage Decimal(5,2),
- > DOB date,
- > School\_Name varchar(30) Default 'Jamna Vidyapeeth');

Query OK, 0 rows affected

### Desc student;

Field	Type	Null	Key	Default	Extra
Rollno	int(11)	NO	PRI	NULL	
Aadhar	varchar(12)	YES	UNI	NULL	
Name	varchar(30)	NO		NULL	
Percentage	decimal(5,2)	YES		NULL	
DOB	date	YES		NULL	
School_Name	varchar(30)	YES		Jamna Vidyapeeth	

6 rows in set (0.01 sec)