PySpark is a Python Module. To perform Processing Jobs on huge data sets in a distributed system, we can use PySpark.

In Pyspark if we want to work on any Dataframe. First we need to create SparkSession and name that spark session.

To read a csv file: **df = spark.read.csv(‘filename’)**

**df** we will get what type of columns in table as o/p.

**df.show()** To see the entire data set with \_\_c0,\_\_c1 as column headings

**spark.read.option(‘header’,’true’).csv(‘filename’)** To set the dataframe first row values as column names.

**df=spark.read.option(‘header’,’true’).csv(‘filename’)**

**df.show()** To get the dataframe as first row values as column names of that dataframe.

**df.head(5)** To get first five rows data

**df.printSchema()** To get the information about the columns like Name(c1) – string, PhNo(c2) – integer in Data Frame.

**Day2:-**

PySpark Data Frame, Reading the Datasets, Checking the Data types of Column(Schema),Selecting Columns and Indexing, Check describe option similar to Pandas, Adding Columns, Dropping Columns

**from pyspark.sql import SparkSession**

**spark = SparkSession.builder.appName(‘DataFrame’).getOrCreate()**

**df = spark.read.option(‘header’,’true’).csv(‘filename’,inferSchema=True)** To get the datatypes of the columns correctly. If we don’t give inferSchema option = true we will get every column type as a string.

**df.show()**

**df.printSchema()**

**OR**

**df = spark.read.csv(‘filename’,header=True,inferSchema=True)**

**df.show()**

**df.printSchema()**

**------------------------------------------------------------------------------------------------------------------------------------------**

**df.columns** To get the column names of the DataFrame

**df.select(‘Column Name’)** returns dataframe as o/p

**df.select(‘Column Name’).show()** returns particular column as dataframe.

**df.select([‘columnname’,’columnname’]).show()** to get multiple column names in a table.

**df.dtypes** to get the column datatypes.

**df.describe().show()** to get count,max,min values of a column

**df[‘columnname’]** will get o/p as Dataframe column name and datatype

**df = df.withColumn(‘Experience after 2 year’,df[‘Experience’]+2)** To add New Column to existed Data frame

**df.show()**

**df = df.drop(‘columnname’)** To drop a column from dataframe

**df= df.drop([‘columnname’,’columnname’])** To drop multiple Columns

**df=df.withColumnRenamed(‘Existing Column Name’,’Replacing with New Column Name’)**

**Day3:-**

Handling Missing Values, Dropping Columns, Dropping Rows, Various Parameters in Dropping Functionalities, Handling Missing Values with Mean, Median or Mode Values.

**df.na.drop().show()** PySpark will drop all the rows which contains null values in any of the column.By default drop function has how=’any’ attribute value

how=’any’ It will drop row if any column value has null value

how=’all’ it will drop row if all column values has null value.

**df.na.drop(thresh=2).show()** It will see if that record has at least **2 not a null values.** And will show that records. If there are more than one column value as not a null value those records will be retrieved.

**df.na.drop(subset=’column\_name’).show()** it will drop the record if the specified column value is null.

**df.na.fill(‘Missing Values’,[‘Name’,’Age’]).show()** it will fill the null values with given value ‘Missing Values’ for Columns Name and Age. \*\*To fill missing values in Integer column we have to change **inferSchema=False** so we can fill null values in Integer Columns too bcz it will consider those columns as String Columns.

To replace null values with mean or median or mode values of the specific column we should import imputer module.

from pyspark.ml.feature import Imputer

imputer=Imputer(

inputCols=[‘age’,’Experience’,’salry’],

outputtCols=[“{}\_imputed”.format(c) for c in [‘age’,’Experience’,’salary’]]

).setStrategy(‘mean’)

imputer.fit(df).transform(df).show()

**Day 4:- Filters**

**df.filter(‘Salary<=20000’).select([‘Name’,’Age’]).show()** It will show the records with Salary <= 20000 and will get only Name and Age Columns.

**df.filter((df[‘Salary’]<=20000) & (df[‘Salary’]>=15000)).select(‘name’,’age’).show()**

**df.filter(~(‘Salary<=2000’)).show()** It will retrieve the records where Salary > 20000.

**Day 5:- Groupby and aggregate(sum(),mean(),min(),max().......)**

**df.groupBy(‘Name’).sum()**

**df.agg({‘Salary’:’sum’}).show()**