To create spark session:

```
from pyspark.sql import SparkSession

spark = SparkSession.builder \
        .appName("Test") \
        .getOrCreate()
```

OR

```
spark= SparkSession.builder.appName("Test").getOrCreate()
```

To read dataset in .csv format:

```
Spark.read.csv("filename.csv", inferSchema = True, header = True)
```

Note : default header and inferSchema value are false. We need to pass those parameters with True value.

Header is used to display the column name along with data set if we keep false as it is, then column name from dataset is not reflected.

InferSchema is used to get the number columns with int/float datatype. If we keep false as it is, then all the columns datatype selected as a string type only.

To convert spark dataframe to pandas dataframe:

```
dataframe_object.toPandas()
```

dataframe_object.limit(5).toPandas() → To display only 5 rows. We can pass any number under the limit.

To select particular column from dataset:

```
Dataframe_object.select("col_1","col_2").show()
```

Validate Data:

df = dataframe_object

dataframe_object.printSchema():

This function used to check the columns name with datatype. This function similar to df.info() function from pandas only the change is here we didn't get the missing value and row/col count.

df.columns

To check the column name from the dataset.

df.describe().show(): This function is used to check the statistical value for each column like count, mean, median.

df.summary().show(): This function similar to describe function only the one change, here we get the value for 25%,50%,75%

Search and Filter Dataframe :

Search DF:

df.select(“Name”,”Marks”).show(5)

df.select([“Name”,”Marks”]).show(5)

df.select(*).show(5)

df.select(df.Name,df.Marks).show(5)

df.select(df[“Name”],df[“Marks”]).show(5)

Column indexing:

df.select(df.columns[1:4]).show(4)

Filter DF:

Group By and Where Statement

df.select(“Name”,”Marks”).orderBy(df[“Marks”].asc()).show(5) →Ascending Order (Default Selected)

df.select(“Name”,”Marks”).orderBy(df[“Marks”].desc()).show(5) → Descending Order

df.select(“Name”,”Marks”).where(df.Name.like(“shu%”)).show(5) → Starting with Shu

df.select(["Name","Marks"]).where(df["Marks"]>=80).show(3)

Normal filter function:

1. df.filter(df.CoapplicantIncome==0).show(3) →Where CoapplicantIncome is equal to Zeros those observation/row will be display as a result.

   df.filter(~(df.CoapplicantIncome!=0)).show(3) → ~ worked for inverse match with condition
2. df.filter(df["Gender"]=="Male").show(3)
3. df.filter("Education <> 'Graduate'").show(3) → Not Equal to graduate
4. df.filter("Education == 'Graduate'").show(3)

col function in filter statement:

from pyspark.sql.functions import col

5. df.filter(col("Gender")=="Male").show(3)

Access Substring:

df.select(“Name”,df.Name.substr(-4,2)).show()

Name→shubham Result → bh