**Alias():** Provides alias name to the column

from pyspark.sql import \*  
from pyspark.sql.functions import \*  
  
spark = SparkSession.builder.appName('alias').getOrCreate()  
  
myData = [(1,'Harsha',2000),  
 (2,'Mokshit',4000),  
 (3,'Harika',3000)]  
  
mySchema = ['id','name','salary']  
  
df = spark.createDataFrame(myData,mySchema)  
  
df1 = df.select(df.id.alias('emp\_id'),df.name.alias('emp\_name'),df.salary.alias('emp\_salary'))  
  
df1.show()

**Output:**

+------+--------+----------+

|emp\_id|emp\_name|emp\_salary|

+------+--------+----------+

| 1| Harsha| 2000|

| 2| Mokshit| 4000|

| 3| Harika| 3000|

+------+--------+----------+

**Asc() & desc():** Sort columns in ascending or descending order.

from pyspark.sql import \*  
from pyspark.sql.functions import \*  
  
spark = SparkSession.builder.appName('alias').getOrCreate()  
  
myData = [(1,'Harsha',2000),  
 (2,'Mokshit',4000),  
 (3,'Harika',3000)]  
  
mySchema = ['id','name','salary']  
  
df = spark.createDataFrame(myData,mySchema)  
  
df.sort(df.name.asc()).show()  
  
df.sort(df.salary.desc()).show()

**Output:**

+---+-------+------+

| id| name|salary|

+---+-------+------+

| 3| Harika| 3000|

| 1| Harsha| 2000|

| 2|Mokshit| 4000|

+---+-------+------+

+---+-------+------+

| id| name|salary|

+---+-------+------+

| 2|Mokshit| 4000|

| 3| Harika| 3000|

| 1| Harsha| 2000|

+---+-------+------+

**Cast():** convert’s the datatype

from pyspark.sql import \*  
from pyspark.sql.functions import \*  
  
spark = SparkSession.builder.appName('alias').getOrCreate()  
  
myData = [(1,'Harsha',2000),  
 (2,'Mokshit',4000),  
 (3,'Harika',3000)]  
  
mySchema = ['id','name','salary']  
  
df = spark.createDataFrame(myData,mySchema)  
  
df1 = df.select(df.id,df.name,df.salary.cast('int'))  
  
df1.show()  
df1.printSchema()

**Output:**

+---+-------+------+

| id| name|salary|

+---+-------+------+

| 1| Harsha| 2000|

| 2|Mokshit| 4000|

| 3| Harika| 3000|

+---+-------+------+

root

|-- id: long (nullable = true)

|-- name: string (nullable = true)

|-- salary: integer (nullable = true)

**Like():** Similar to SQL LIKE expression.

from pyspark.sql import \*  
from pyspark.sql.functions import \*  
  
spark = SparkSession.builder.appName('alias').getOrCreate()  
  
myData = [(1,'Harsha',2000),  
 (2,'Mokshit',4000),  
 (3,'Harika',3000)]  
  
mySchema = ['id','name','salary']  
  
df = spark.createDataFrame(myData,mySchema)  
  
df1 = df.filter(df.name.like('M%'))  
  
df1.show()

**Output:**

+---+-------+------+

| id| name|salary|

+---+-------+------+

| 2|Mokshit| 4000|

+---+-------+------+