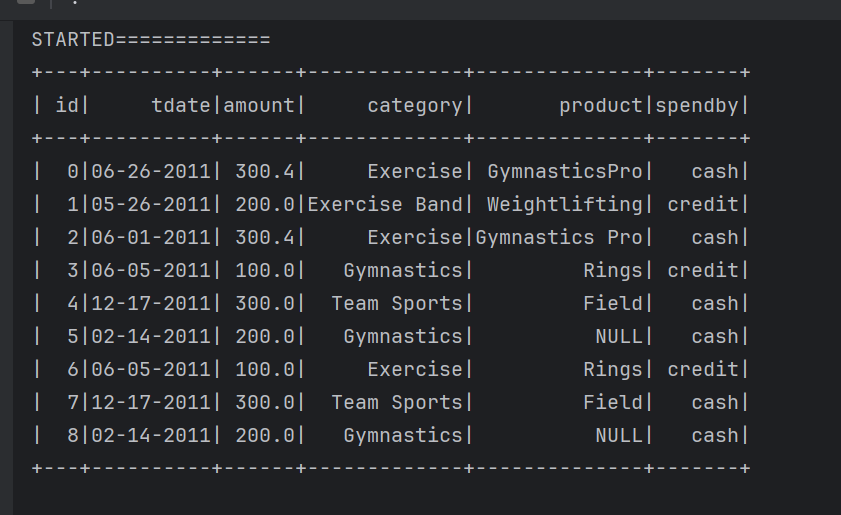
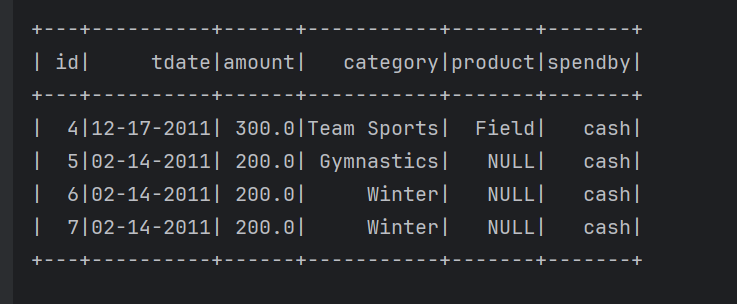
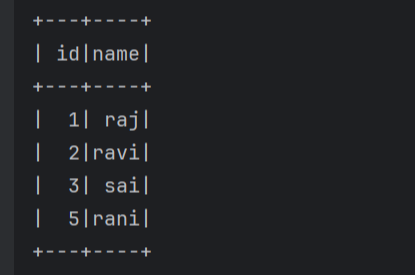
**SQL Task**

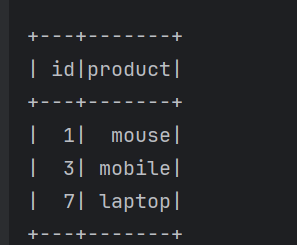
#🔴 SQL PRE REQUISITE CODE  
  
data = [  
 (0, "06-26-2011", 300.4, "Exercise", "GymnasticsPro", "cash"),  
 (1, "05-26-2011", 200.0, "Exercise Band", "Weightlifting", "credit"),  
 (2, "06-01-2011", 300.4, "Exercise", "Gymnastics Pro", "cash"),  
 (3, "06-05-2011", 100.0, "Gymnastics", "Rings", "credit"),  
 (4, "12-17-2011", 300.0, "Team Sports", "Field", "cash"),  
 (5, "02-14-2011", 200.0, "Gymnastics", None, "cash"),  
 (6, "06-05-2011", 100.0, "Exercise", "Rings", "credit"),  
 (7, "12-17-2011", 300.0, "Team Sports", "Field", "cash"),  
 (8, "02-14-2011", 200.0, "Gymnastics", None, "cash")  
]  
  
df = spark.createDataFrame(data, ["id", "tdate", "amount", "category", "product", "spendby"])  
df.show()  
  
  
  
  
  
data2 = [  
 (4, "12-17-2011", 300.0, "Team Sports", "Field", "cash"),  
 (5, "02-14-2011", 200.0, "Gymnastics", None, "cash"),  
 (6, "02-14-2011", 200.0, "Winter", None, "cash"),  
 (7, "02-14-2011", 200.0, "Winter", None, "cash")  
]  
  
df1 = spark.createDataFrame(data2, ["id", "tdate", "amount", "category", "product", "spendby"])  
df1.show()  
  
  
  
  
data4 = [  
 (1, "raj"),  
 (2, "ravi"),  
 (3, "sai"),  
 (5, "rani")  
]  
  
  
  
cust = spark.createDataFrame(data4, ["id", "name"])  
cust.show()  
  
data3 = [  
 (1, "mouse"),  
 (3, "mobile"),  
 (7, "laptop")  
]  
  
prod = spark.createDataFrame(data3, ["id", "product"])  
prod.show()  
  
csvdf = spark.read.format("csv").option("header","true").load("usdata.csv")  
print()  
print("======== CSV DF==============")  
print()  
csvdf.show()  
  
  
  
parquetdf = spark.read.format("parquet").load("file5.parquet")  
print()  
print("======== parquetdf ==============")  
print()  
parquetdf.show()  
  
  
  
orcdf = spark.read.format("orc").load("data.orc")  
print()  
print("======== orcdf ==============")  
print()  
orcdf.show()  
  
  
  
  
jsondf = spark.read.format("json").load("file4.json")  
print()  
print("======== jsondf ==============")  
print()  
jsondf.show()  
  
  
  
df.createOrReplaceTempView("df")

  
  
df1.createOrReplaceTempView("df1")

  
  
cust.createOrReplaceTempView("cust")

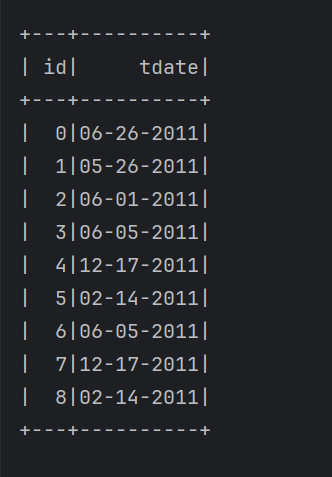


prod.createOrReplaceTempView("prod")



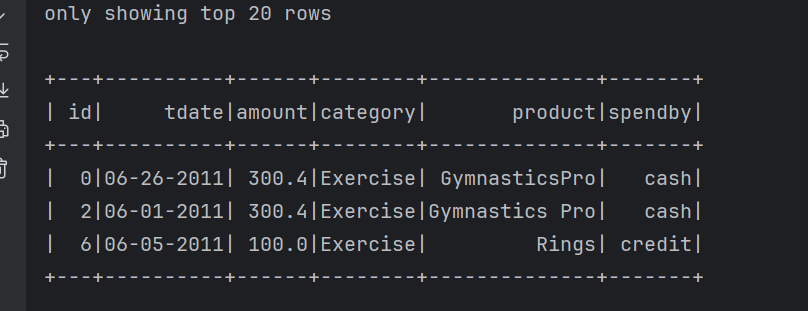
# Task start from here

**1- id and tdate from df**spark.sql("select id,tdate from df").show()



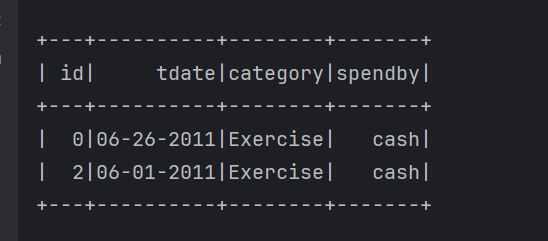
2- category= exerscise

spark.sql("select \* from df where category=’Exercise’").show()



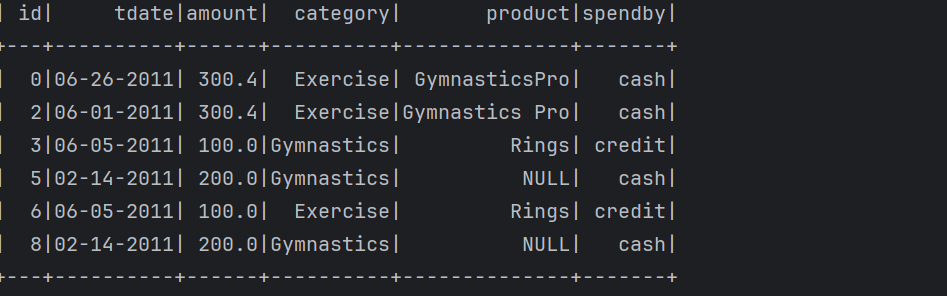
3- id,tdate,category and standby where category=excercise and standby = cash from df

spark.sql("select id,tdate,category,spendby from df where category='Exercise' and spendby ='cash' ").show()



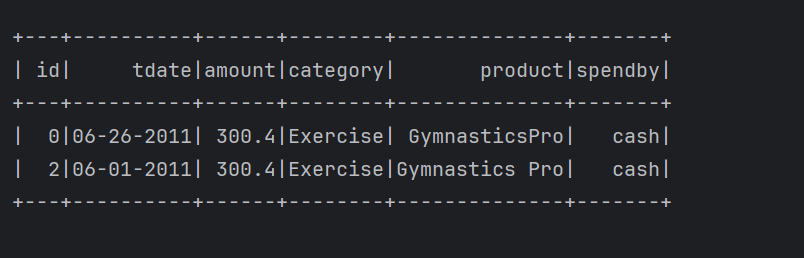
**4-** exercise and gymnastics from category column

spark.sql("select \* from df where category in ('Exercise','Gymnastics')").show()



5- product contain gymnastics

spark.sql("select \* from df where product like '%Gymnastics%'").show()



6-category not equals execise

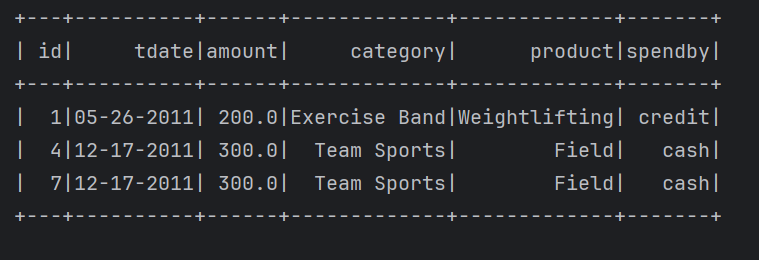
spark.sql("select \* from df where category != 'Exercise'").show()



7- category not equals to exercise and gymanastics

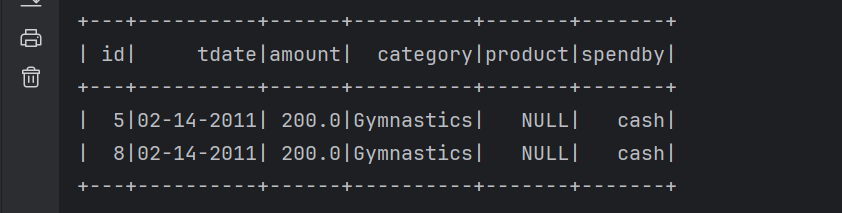
spark.sql("select \* from df where category not in ('Exercise','Gymnastics')").show()

**(not in represents multi values)**



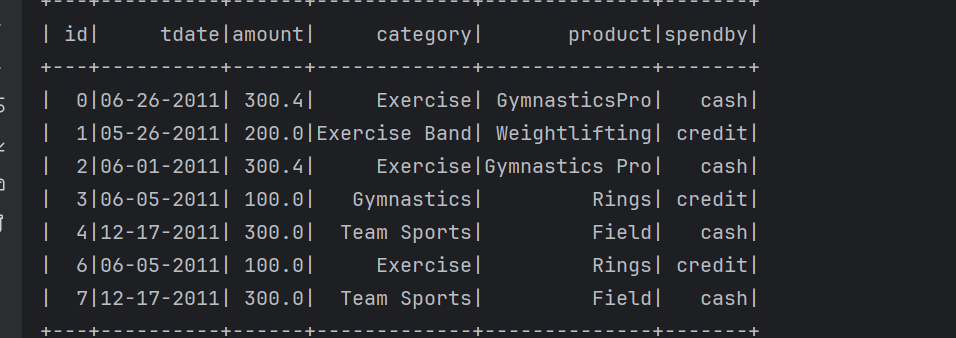
8-product is null

spark.sql("select \* from df where product is null").show()



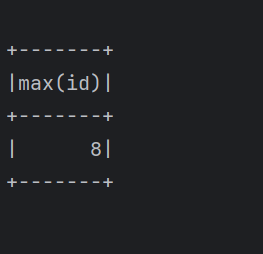
9- product is not null

spark.sql("select \* from df where product is not null").show()



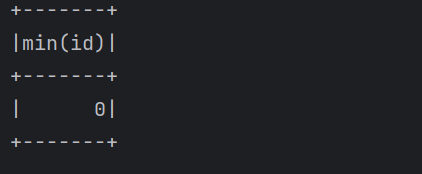
10- max of id

spark.sql("select max(id) from df ").show()



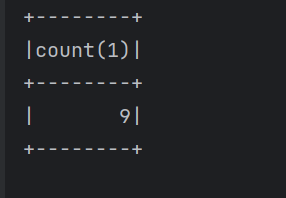
11- min of id

spark.sql("select min(id) from df ").show()



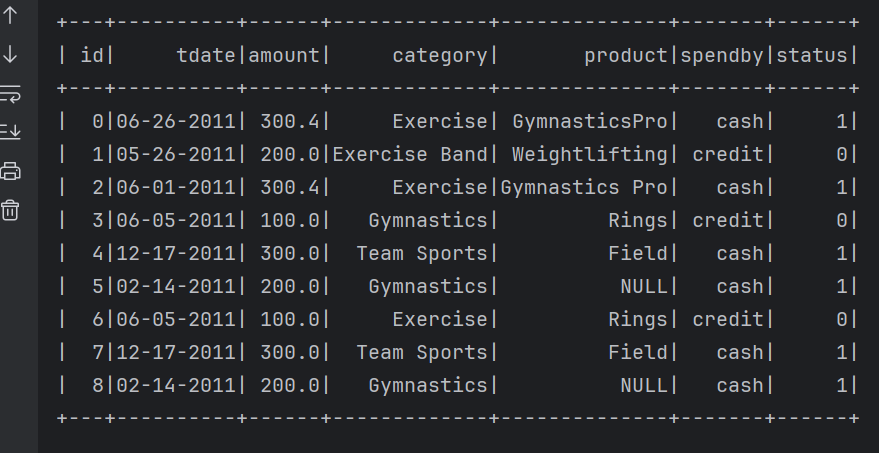
12-count

spark.sql("select count(1) from df").show()



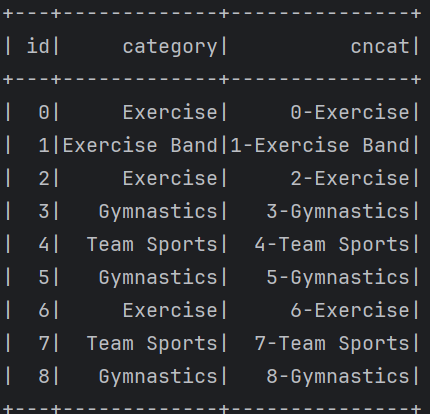
13-Add new column and add some values here(conditional statement)

spark.sql("select \*,case when spendby='cash' then 1 else 0 end as status from df").show()



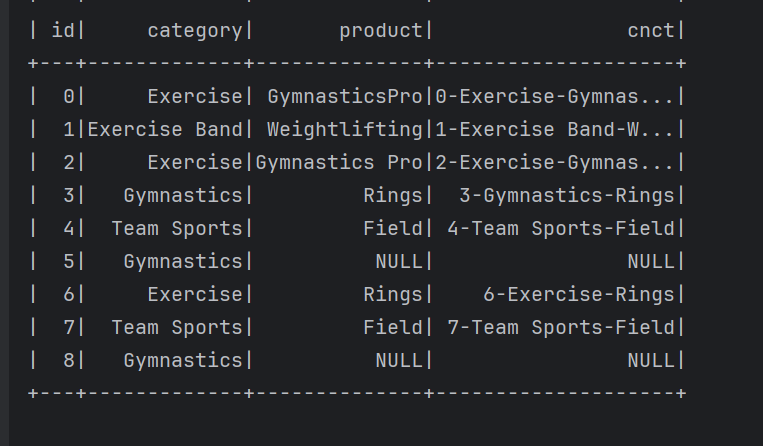
14-concat two columns

spark.sql("select id,category,concat(id,'-',category) as cncat from df").show()



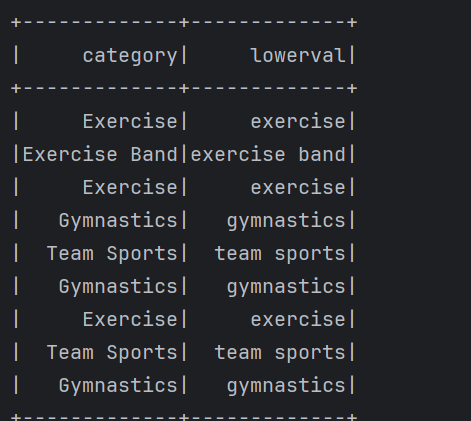
15- concat multi columns

spark.sql("select id,category,product,concat(id,'-',category,'-'product) as cncat from df").show()



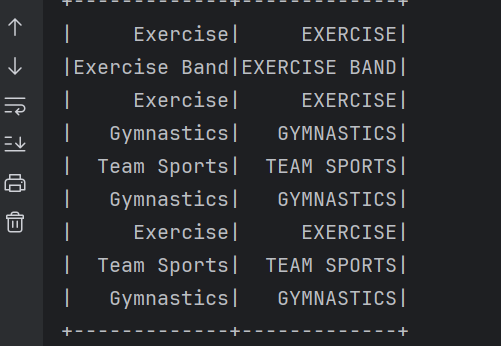
16-select lower

spark.sql("select category,lower(category) as lowerval from df").show()



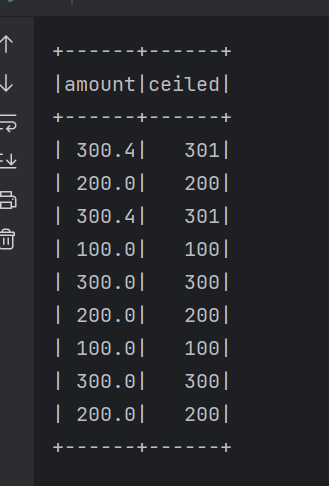
17-16-select upper

spark.sql("select category,upper(category) as lowerval from df").show()



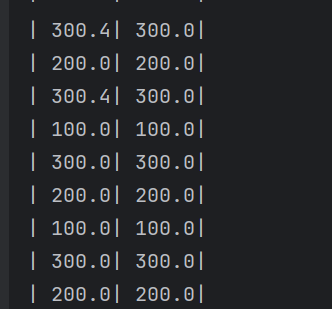
18-ceil operation

spark.sql("select amount,ceil(amount) as ceiled from df").show()



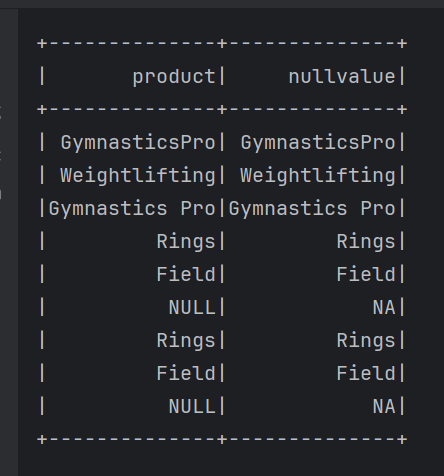
19- round operation

spark.sql("select amount,round(amount) as ceiled from df").show()



20-replace NULL

spark.sql("select product,coalesce(product,'NA') as nullvalue from df").show()



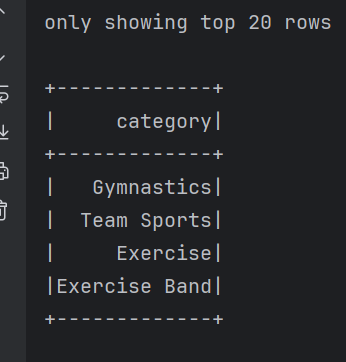
21-fill the spaces(trim- it will remove the spaces from back and front of the word)

spark.sql("select product,trim(product) as trimed from df").show()



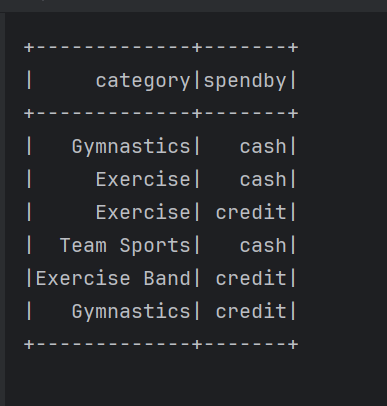
22-distinct

spark.sql("select distinct category from df").show()



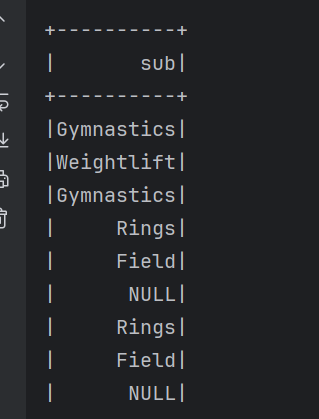
23- distinct on multimle column

spark.sql("select distinct category,standby from df").show()



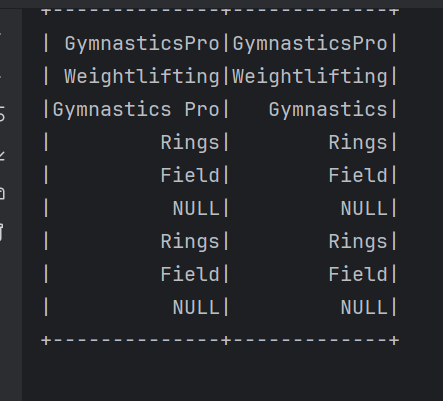
24-substring on product

spark.sql("select substring(product,1,10) as sub from df").show()



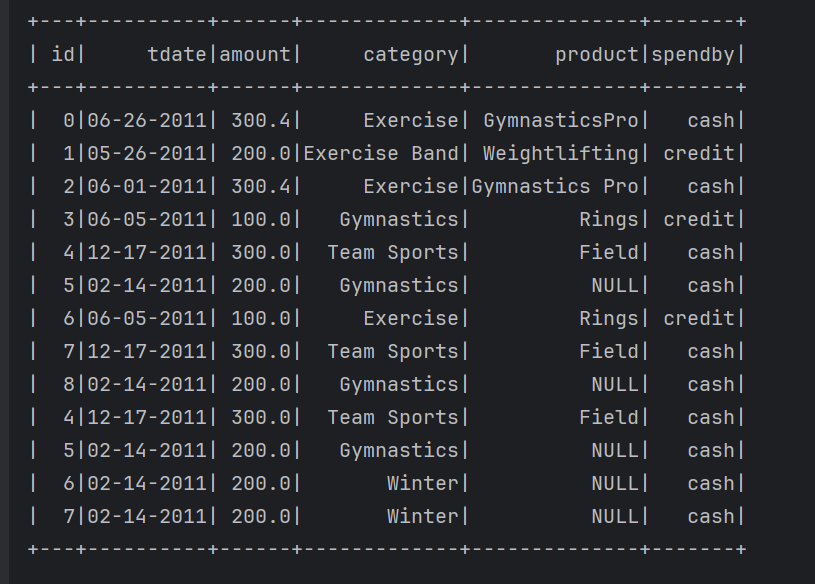
25-split

spark.sql("select product,split(product,' ')[0] as split from df").show()



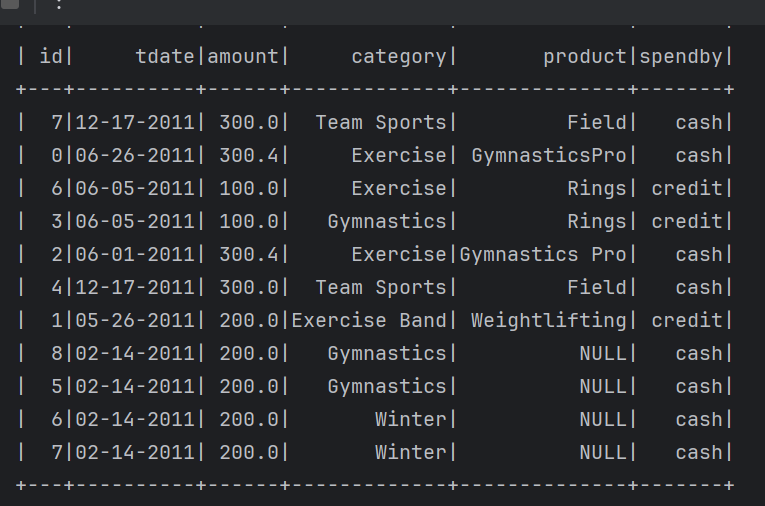
26- union all- combine two dataframe

spark.sql("select \* from df union all select \* from df1").show()



27- union - combine two dataframe and remove dublicates rows.

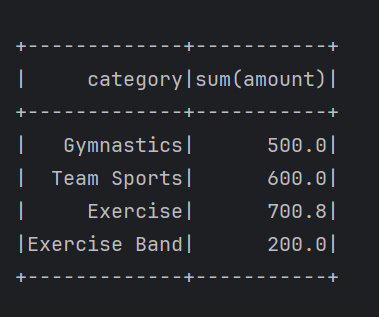
spark.sql("select \* from df union select \* from df1").show()



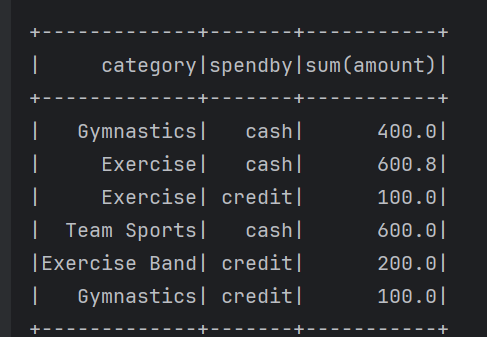
28- aggregate-grouping as column and finding the sum of other column.

-total amount column from each category.

spark.sql("select category, sum(amount) from df group by category").show()

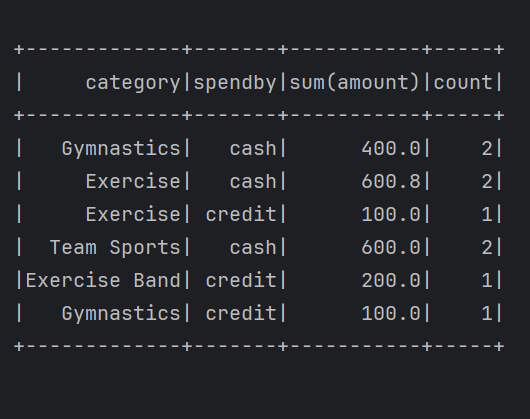


29-aggregate two columns



30- aggregate two columns and add count

spark.sql("select category,spendby, sum(amount),count(amount) as count from df group by category,spendby").show()

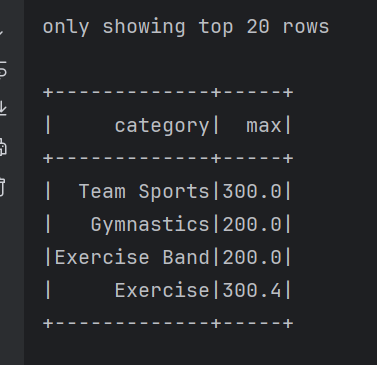


31-how to get max of category



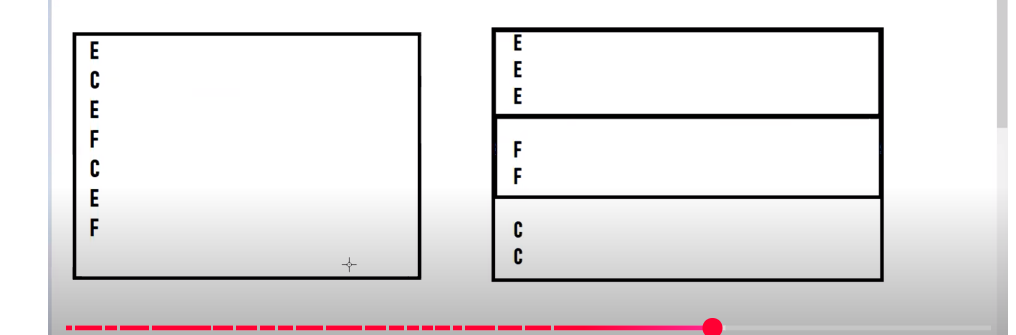
32-use orderby here

spark.sql("select category,max(amount) as max from df group by category order by category desc").show()



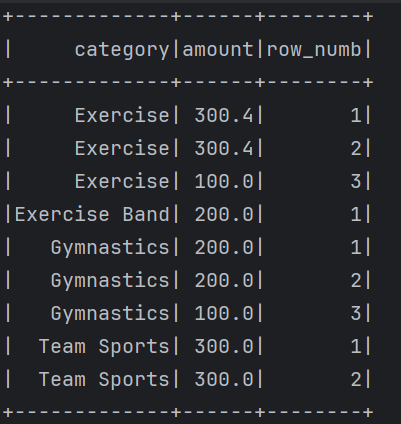
**Window row number**

Perform operations to separate each category is called window row. It is nothing but categoring the column



33-row\_num

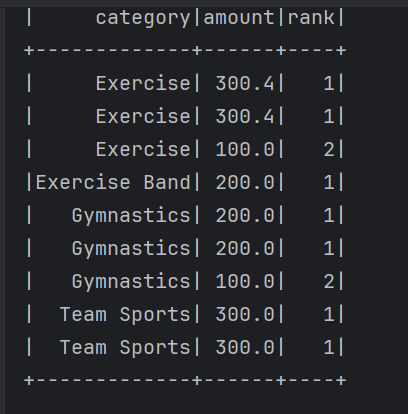
spark.sql("select category,amount,row\_number()OVER(partition by category order by amount desc) as row\_numb from df ").show()



34- you can rank instead of row num using dense\_rank

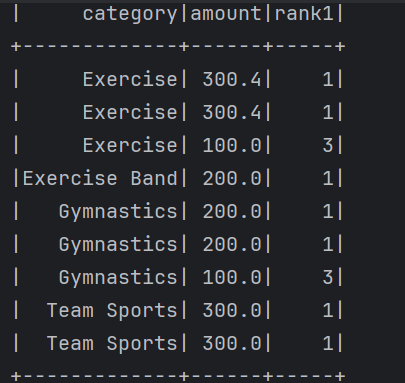
It is used when the similar data is present it will rank same to the similar data

spark.sql("select category,amount,dense\_rank()OVER(partition by category order by amount desc) as rank from df ").show()

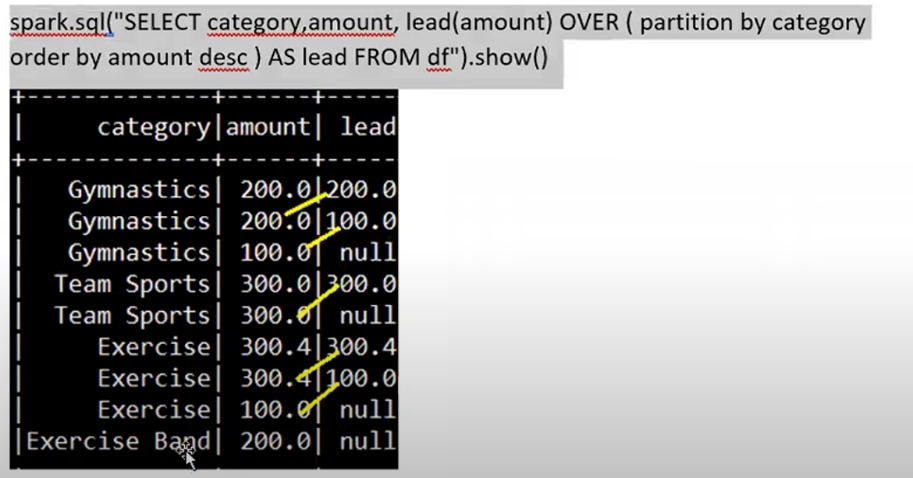


35-rank

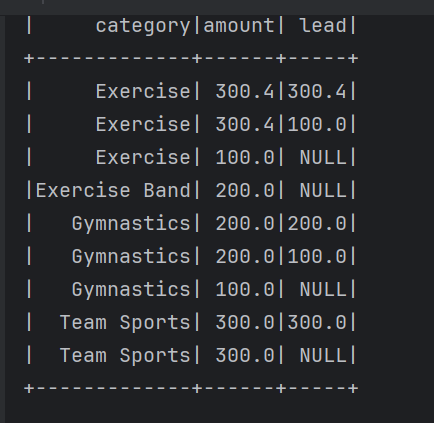
Little different from dense\_rank, it will give same rank to the similar data but, it two category got same rank like 1 in both, then third category will get rank three because already two got rank 1.



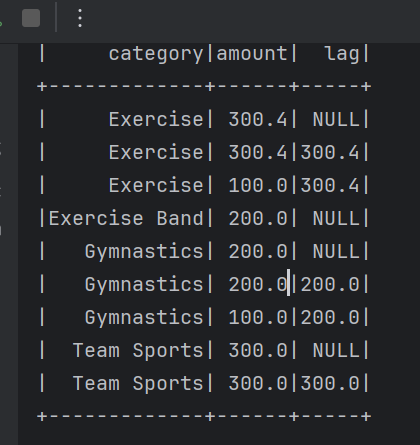
**Window lead function**



spark.sql("select category,amount,lead(amount) OVER(partition by category order by amount desc) as lead from df ").show()



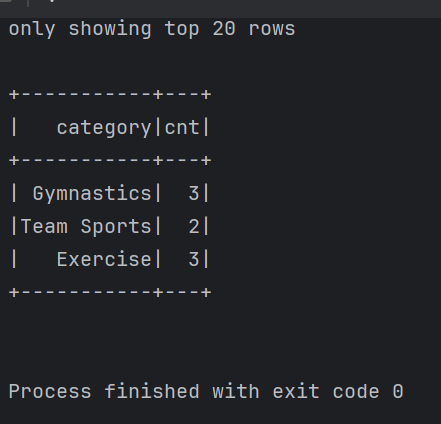
**Window lag- it will put down one step the data**



**Having function**

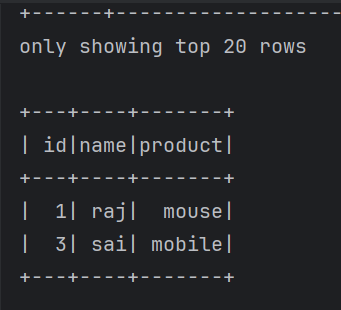
Helps you to find how many time as rows got duplicated. It will perform on group by

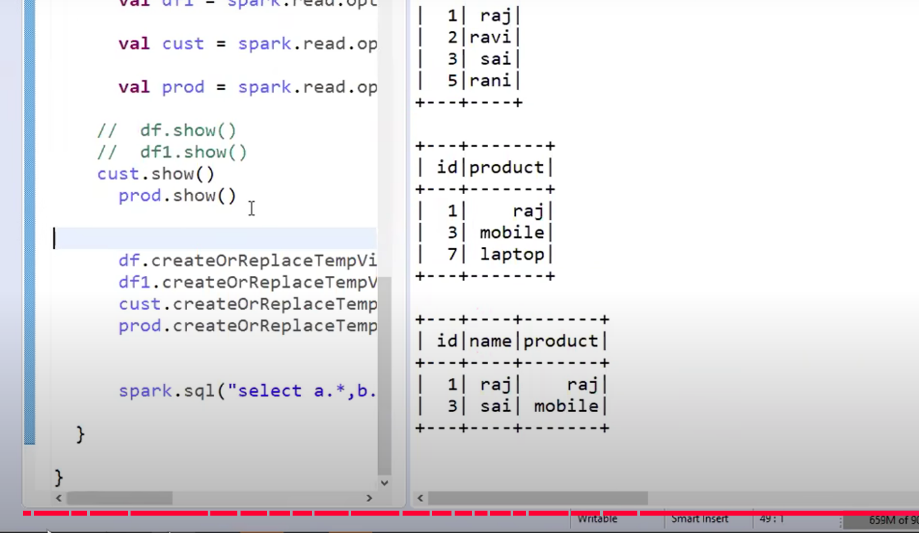
spark.sql("select category,count(category) as cnt from df group by category having cnt >1 ").show()



**Inner join**

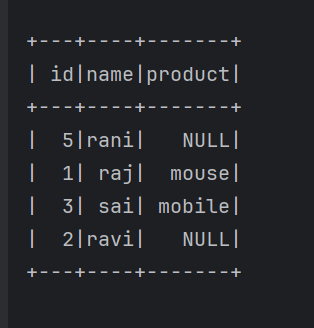
spark.sql("select a.\*,b.product from cust a join prod b on a.id=b.id ").show()





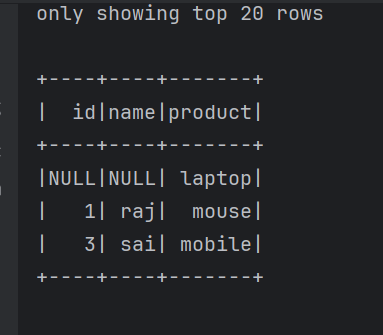
**left join**

spark.sql("select a.\*,b.product from cust a left join prod b on a.id=b.id ").show()



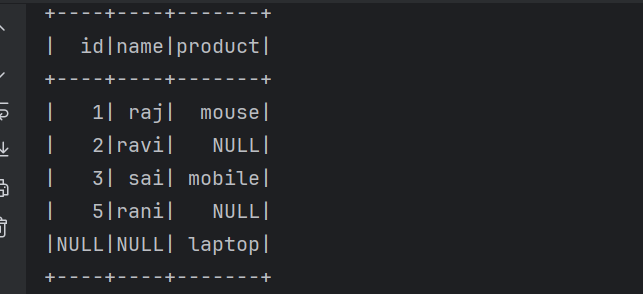
**right join**

spark.sql("select a.\*,b.product from cust a right join prod b on a.id=b.id ").show()



**full join**

spark.sql("select a.\*,b.product from cust a right join prod b on a.id=b.id ").show()



**Anti join**

Take ids from left which do not have in right

