

**val** df = spark.read.option("header","true").csv("file:///C:/data/df.csv")

**val** df1 = spark.read.option("header","true").csv("file:///C:/data/df1.csv")

**val** cust = spark.read.option("header","true").csv("file:///C:/data/cust.csv")

**val** prod = spark.read.option("header","true").csv("file:///C:/data/prod.csv")

df.show()

df1.show()

cust.show()

prod.show()

df.createOrReplaceTempView("df")

df1.createOrReplaceTempView("df1")

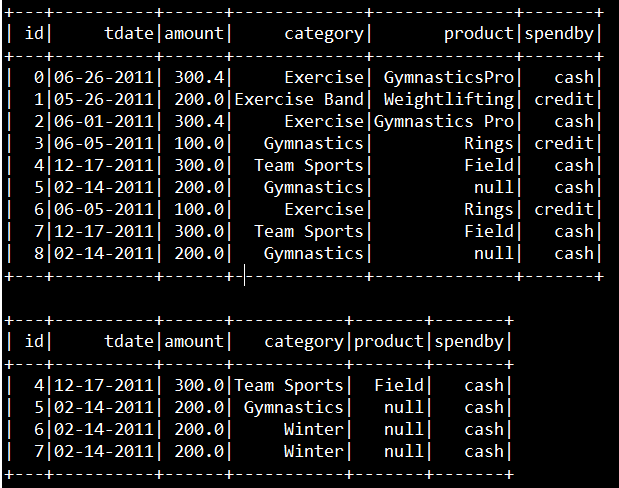
cust.createOrReplaceTempView("cust")

prod.createOrReplaceTempView("prod")

sc.setLogLevel("ERROR")

spark.sql("select \* from df order by id").show()

spark.sql("select \* from df1 order by id").show()



**====================================**

**Validate data**

**====================================**

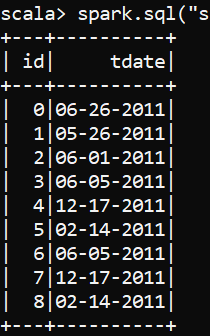
spark.sql("select \* from df ").show()

**====================================**

**Select two columns**

**====================================**

spark.sql("select id,tdate from df order by id").show()

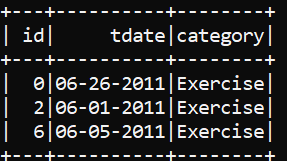


**====================================**

**Select column with category filter = Exercise**

**====================================**

spark.sql("select id,tdate,category from df where category='Exercise' order by id").show()

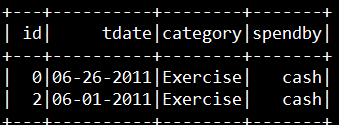


**====================================**

**Multi Column filter**

**====================================**

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

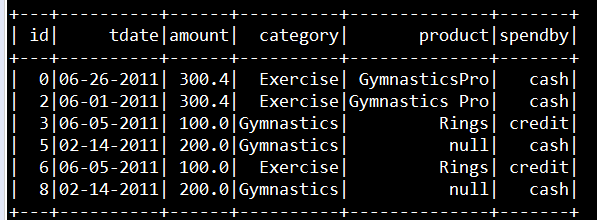


**====================================**

**Multi Value Filter**

**====================================**

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

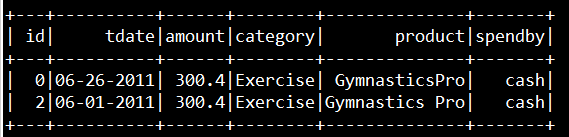


**====================================**

**Like Filter**

**====================================**

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

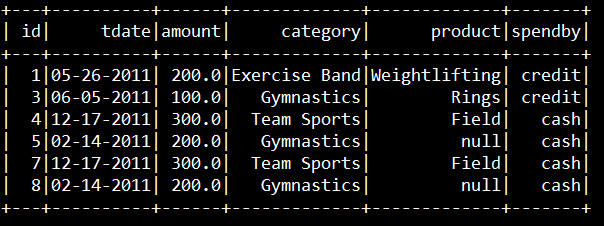


**====================================**

**Not Filters**

**====================================**

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

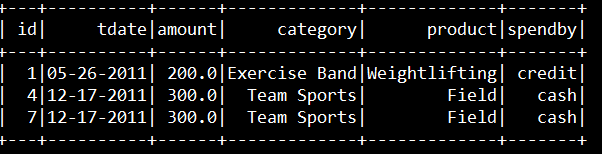


**====================================**

**Not In Filters**

**====================================**

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

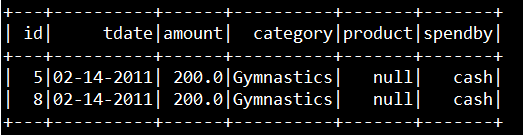


**====================================**

**Null Filters**

**====================================**

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

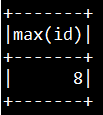


**====================================**

**Max Function**

**====================================**

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

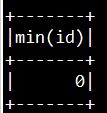


**====================================**

**Min Funtion**

**====================================**

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

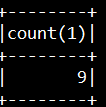


**====================================**

**Count**

**====================================**

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

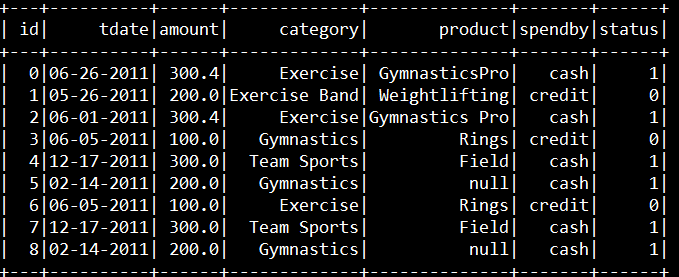


**====================================**

**Condition statement**

**====================================**

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

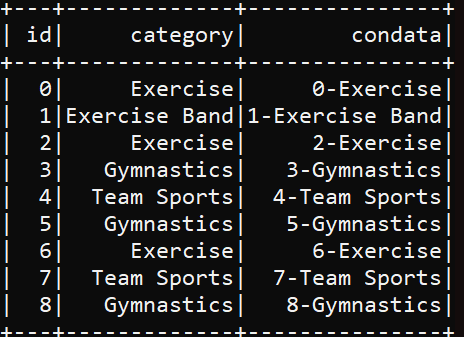


**====================================**

**Concat data**

**====================================**

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

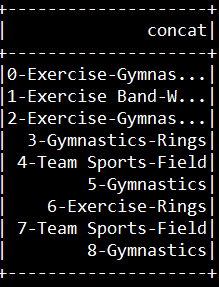


**====================================**

**Concat\_ws data**

**====================================**

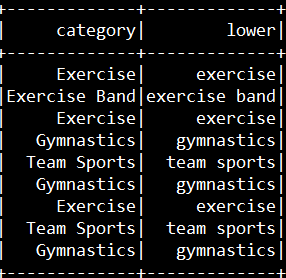
spark.sql("select concat\_ws('-',id,category,product) as concat from df ").show()



**====================================**

**Lower Case data**

**====================================**



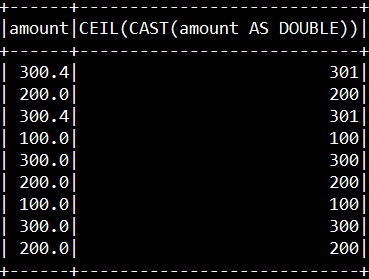
spark.sql("select category,lower(category) as lower from df ").show()

**====================================**

**Ceil data**

**====================================**

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

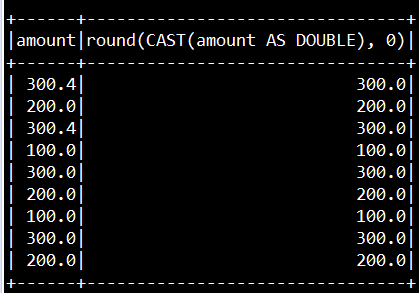


**====================================**

**Round the data**

**====================================**

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

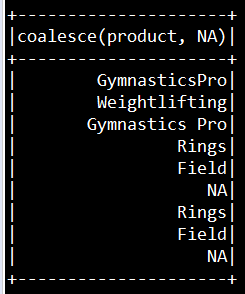


**====================================**

**Replace Nulls**

**====================================**

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

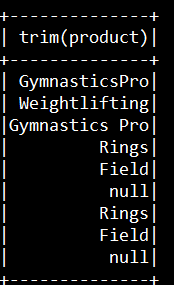


**====================================**

**Trim the space**

**====================================**

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

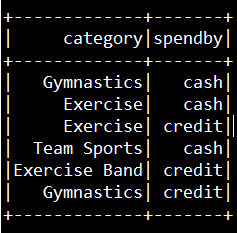


**====================================**

**Distinct the columns**

**====================================**

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

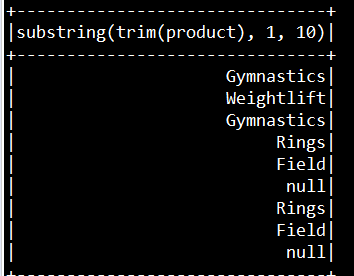


**====================================**

**Substring with Trim**

**====================================**

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

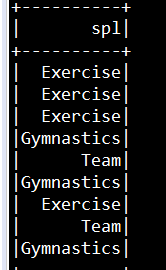


**====================================**

**Substring/Split operation**

**====================================**

spark.sql("select SUBSTRING\_INDEX(category,' ',1) as spl from df").show()

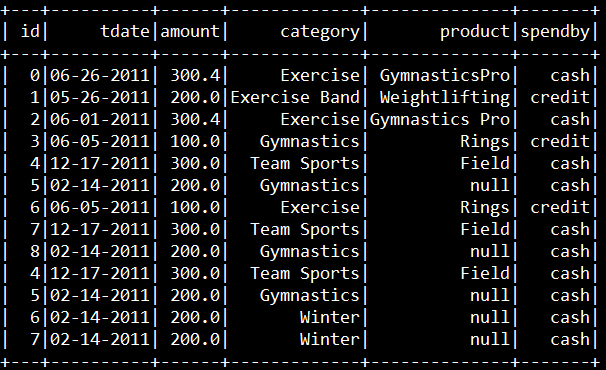


**====================================**

**Union all**

**====================================**

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

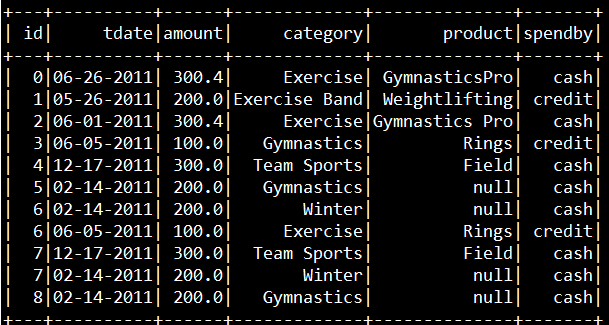


**====================================**

**Union**

**====================================**

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

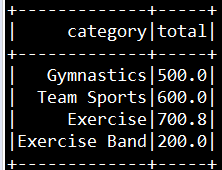


**====================================**

**Aggregate Sum**

**====================================**

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

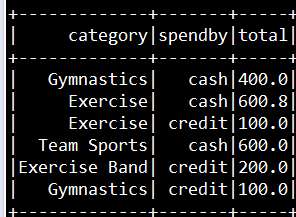


**====================================**

**Aggregate sum with two columns**

**====================================**

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

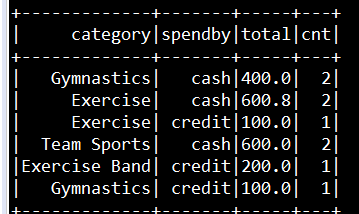


**====================================**

**Aggregate Count**

**====================================**

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

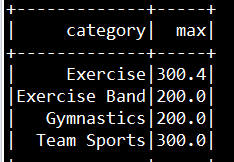


**====================================**

**Aggregate Max**

**====================================**

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

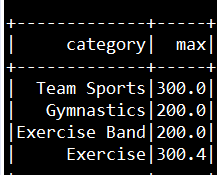


**====================================**

**Aggregate with Order Descending**

**====================================**

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

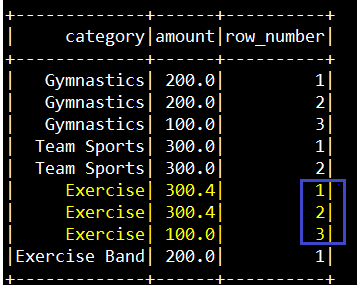


**====================================**

**Window Row Number**

**====================================**

spark.sql("SELECT category,amount, row\_number() OVER ( partition by category order by amount desc ) AS row\_number FROM df").show()

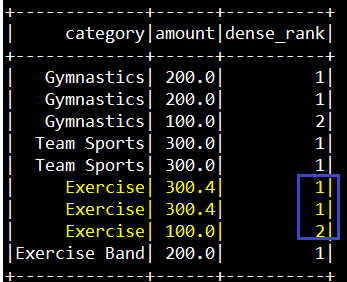
****

**====================================**

**Window Dense\_rank Number**

**====================================**

spark.sql("SELECT category,amount, dense\_rank() OVER ( partition by category order by amount desc ) AS dense\_rank FROM df").show()

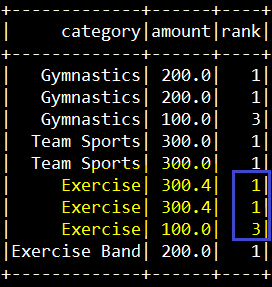


**====================================**

**Window rank Number**

**====================================**

spark.sql("SELECT category,amount, rank() OVER ( partition by category order by amount desc ) AS rank FROM df").show()

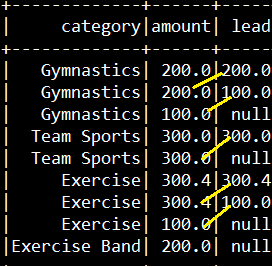


**====================================**

**Window Lead function**

**====================================**

spark.sql("SELECT category,amount, lead(amount) OVER ( partition by category order by amount desc ) AS lead FROM df").show()

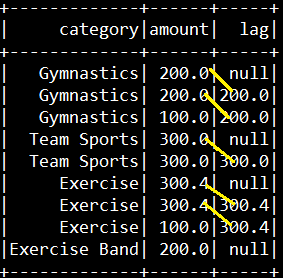


**====================================**

**Window lag function**

**====================================**

spark.sql("SELECT category,amount, lag(amount) OVER ( partition by category order by amount desc ) AS lag FROM df").show()

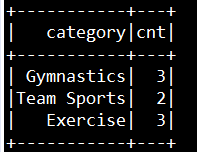


**====================================**

**Having function**

**====================================**

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

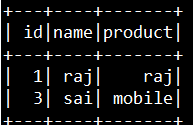


**====================================**

**Inner Join**

**====================================**

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

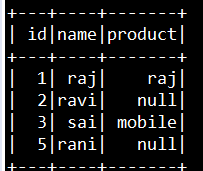


**====================================**

**Left Join**

**====================================**

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

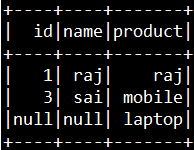


**====================================**

**Right Join**

**====================================**

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

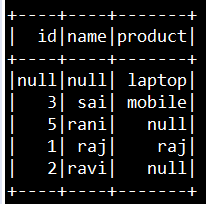


**====================================**

**Full Join**

**====================================**

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

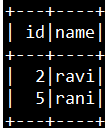


**====================================**

**left anti Join**

**====================================**

spark.sql("select a.id,a.name from cust a LEFT ANTI JOIN prod b on a.id=b.id").show()

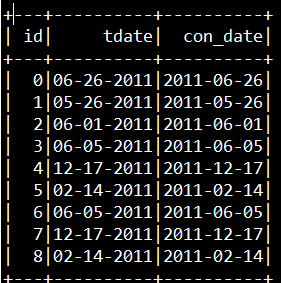


**====================================**

**Date format**

**====================================**

spark.sql("select id,tdate,from\_unixtime(unix\_timestamp(tdate,'MM-dd-yyyy'),'yyyy-MM-dd') as con\_date from df").show()



**====================================**

**Sub query**

**====================================**

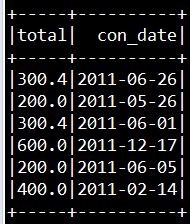
spark.sql("""

select sum(amount) as total , con\_date from(

select id,tdate,from\_unixtime(unix\_timestamp(tdate,'MM-dd-yyyy'),'yyyy-MM-dd') as con\_date,amount,category,product,spendby from df)

group by con\_date

""").show()

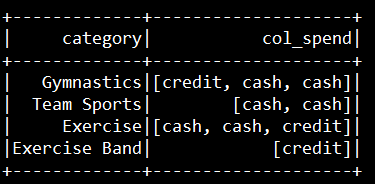


**====================================**

**collect\_list**

**====================================**

spark.sql("select category,collect\_list(spendby) as col\_spend from df group by category").show()

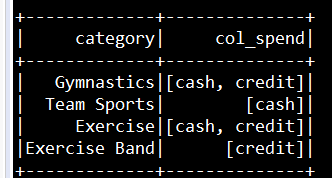


**====================================**

**collect\_set**

**====================================**

spark.sql("select category,collect\_set(spendby) as col\_spend from df group by category ").show()

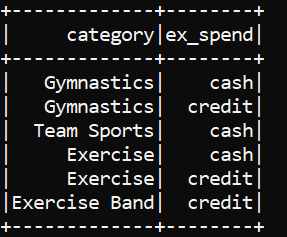


**====================================**

**Explode**

**====================================**

spark.sql("select category,explode(col\_spend) as ex\_spend from (select category,collect\_set(spendby) as col\_spend from df group by category)").show()

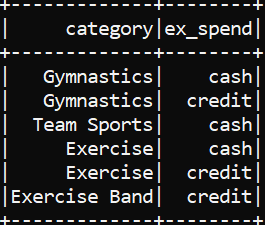


**====================================**

**explode\_outer**

**====================================**

spark.sql("select category,explode\_outer(col\_spend) as ex\_spend from (select category,collect\_set(spendby) as col\_spend from df group by category)").show()



**====================================**

**Total Eclipse Code**

**====================================**

package pack

import org.apache.spark.SparkContext // rdd

import org.apache.spark.sql.SparkSession // dataframe

import org.apache.spark.SparkConf

import org.apache.spark.sql.\_

import org.apache.spark.sql.types.\_

import org.apache.spark.sql.types.IntegerType

import org.apache.spark.sql.functions.upper

import org.apache.spark.sql.catalyst.expressions.Upper

import org.apache.spark.sql.functions.\_

import org.apache.spark.sql.expressions.Window

import scala.io.Source

object obj {

def main(args:Array[String]):Unit={

val conf = new SparkConf().setAppName("Revision").setMaster("local[\*]")

val sc = new SparkContext(conf)

sc.setLogLevel("ERROR")

val spark = SparkSession.builder().enableHiveSupport()

.config("spark.sql.warehouse.dir", "file:///C:/hivewarehou/")

.config("spark.sql.catalogImplementation","hive").getOrCreate()

import spark.implicits.\_

// ------- URL read

val df = spark.read.option("header","true").csv("file:///C:/data/df.csv")

val df1 = spark.read.option("header","true").csv("file:///C:/data/df1.csv")

val cust = spark.read.option("header","true").csv("file:///C:/data/cust.csv")

val prod = spark.read.option("header","true").csv("file:///C:/data/prod.csv")

df.show()

df1.show()

cust.show()

prod.show()

df.createOrReplaceTempView("df")

df1.createOrReplaceTempView("df1")

cust.createOrReplaceTempView("cust")

prod.createOrReplaceTempView("prod")

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

spark.sql("select id,tdate from df order by id").show()

spark.sql("select id,tdate,category from df where category='Exercise' order by id").show()

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

spark.sql("select SUBSTRING\_INDEX(category,' ',1) as spl from df").show()

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

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

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

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

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

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

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

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

spark.sql("SELECT category,amount, row\_number() OVER ( partition by category order by amount desc ) AS row\_number FROM df").show()

spark.sql("SELECT category,amount, dense\_rank() OVER ( partition by category order by amount desc ) AS dense\_rank FROM df").show()

spark.sql("SELECT category,amount, rank() OVER ( partition by category order by amount desc ) AS rank FROM df").show()

spark.sql("SELECT category,amount, lead(amount) OVER ( partition by category order by amount desc ) AS lead FROM df").show()

spark.sql("SELECT category,amount, lag(amount) OVER ( partition by category order by amount desc ) AS lag FROM df").show()

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

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

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

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

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

spark.sql("select a.id,a.name from cust a LEFT ANTI JOIN prod b on a.id=b.id").show()

spark.sql("select a.id,a.name from cust a LEFT SEMI JOIN prod b on a.id=b.id").show()

spark.sql("select id,tdate,from\_unixtime(unix\_timestamp(tdate,'MM-dd-yyyy'),'yyyy-MM-dd') as con\_date from df").show()

spark.sql("""

select sum(amount) as total , con\_date from(

select id,tdate,from\_unixtime(unix\_timestamp(tdate,'MM-dd-yyyy'),'yyyy-MM-dd') as con\_date,amount,category,product,spendby from df)

group by con\_date

""").show()

spark.sql("select category,collect\_list(spendby) as col\_spend from df group by category").show()

spark.sql("select category,collect\_set(spendby) as col\_spend from df group by category").show()

}

}