**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()

}

}

package pack

import org.apache.spark.SparkContext

import org.apache.spark.SparkConf

object obj {

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

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

val sc = new SparkContext(conf)

sc.setLogLevel("ERROR")

val data = sc.textFile("file:///C:/data/scdata.txt")

println("====Raw Rdd====")

println

data.foreach(println)

println

val flatdata = data.flatMap(x => x.split("~"))

println("====flatdata Rdd====")

println

flatdata.foreach(println)

println

val statedata = flatdata.filter( x=> x.toLowerCase().contains("state"))

println("====statedata Rdd====")

println

statedata.foreach(println)

println

val citydata = flatdata.filter( x=>x.contains("City"))

println("====citydata Rdd====")

println

citydata.foreach(println)

println

val finalstate = statedata.map( x=>x.replace("State->",""))

println("=====finalstate Rdd====")

println

finalstate.foreach(println)

println

val finalcity = citydata.map( x=>x.replace("City->",""))

println("====finalcity Rdd====")

println

finalcity.foreach(println)

finalstate.coalesce(1).saveAsTextFile("file:///D:/data/statedata1")

}

}

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.\_

object obj {

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

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

val sc = new SparkContext(conf)

sc.setLogLevel("ERROR")

val spark = SparkSession.builder.getOrCreate()

import spark.implicits.\_

val df = spark

.read

.format("xml")

.option("rowtag","book")

.load("file:///C:/data/book.xml")

df.show()

}

}