***SPARK USECASE***

1. Clean and Transform data for proper processing and getting complete insights without any garbage values

import pyspark

from pyspark.sql import SparkSession

from pyspark.sql.functions import col, asc,desc,row\_number

import pandas as pd

import csv

df = pd.read\_csv("C:\\Users\\AbbleAnif\\Documents\\casestudy\\pyspark\_usecase\\complaints22.csv")

import pandas as pd

# change in date time format

date1 = pd.to\_datetime(pd.Series(df['Date received']))

df['Date received']= date1.dt.strftime('%d/%m/%Y')

date1 = pd.to\_datetime(pd.Series(df['Date sent to company']))

df['Date sent to company']= date1.dt.strftime('%d/%m/%Y')

df.to\_csv("C:\\Users\\AbbleAnif\\Documents\\casestudy\\pyspark\_usecase\\complaints21.csv", index = False)

output with cleaned data



1. Find the number of complaints for which the Company response is currently "in progress".

import pyspark

from pyspark.sql import SparkSession

from pyspark.sql.functions import col, asc,desc,row\_number

spark = SparkSession.builder.appName('SparkByExamples.com').getOrCreate()

df = spark.read.csv("C:\\Users\\AbbleAnif\\Documents\\casestudy\\pyspark\_usecase\\complaints.csv")

df.createOrReplaceTempView("complaints")

df2 = spark.sql("select count(complaints.\_c0) from complaints where complaints.\_c11='In progress' ")

df2.show()

OUTPUT



1. Which company has the maximum consumer complaints.

import findspark

findspark.init()

import pyspark

from pyspark.sql import SparkSession

from pyspark.sql.functions import col, asc,desc,row\_number

spark = SparkSession.builder.appName('SparkByExamples.com').getOrCreate()

df = spark.read.csv("C:\\Users\\AbbleAnif\\Documents\\casestudy\\pyspark\_usecase\\complaints.csv")

#df.show()

#df.printSchema()

df.createOrReplaceTempView("complaints")

df1 = spark.sql("select \_c10 as company,count(\_C0) as c from complaints group by \_c10")

#df1.show()

df2=df1.orderBy(col("c").desc()).limit(1)

df2.show()

OUTPUT



1. Which Companies is able to solve issues of customers (on the terms of Company response and timely response)

import findspark

findspark.init()

import pyspark

from pyspark.sql import SparkSession

from pyspark.sql.functions import col, asc,desc,row\_number

spark = SparkSession.builder.appName('SparkByExamples.com').getOrCreate()

df = spark.read.csv("C:\\Users\\AbbleAnif\\Documents\\casestudy\\pyspark\_usecase\\complaints.csv")

#df.show()

df.createOrReplaceTempView("complaints")

df1=spark.sql(" select distinct \_c10 as company from complaints where \_c11='Closed with explanation' and \_c12='Yes'")

df1.show()

OUTPUT



1. Which companies are having least response time for a complaint raised?

import findspark

findspark.init()

import pyspark

from pyspark.sql import SparkSession

from pyspark.sql.functions import col, asc,desc,row\_number

spark = SparkSession.builder.appName('SparkByExamples.com').getOrCreate()

path="C:\\Users\\AbbleAnif\\Documents\\casestudy\\pyspark\_usecase\\complaints.csv"

df = spark.read.csv(path)

#df.show()

df.createOrReplaceTempView("complaints")

df1=spark.sql("select distinct \_c10 from complaints where \_c12='Yes' ")

df1.show()

OUTPUT



***SCALA SPARK USECASE***

1. Find the issue that occurred mostly.

import org.apache.spark.sql.SparkSession  
import org.apache.spark.sql.functions.*desc*object usecase\_6 extends App{  
  
 val *spark*: SparkSession = SparkSession.*builder*()  
 .master("local[\*]")  
 .appName("SparkByExamples.com")  
 .getOrCreate()  
  
 //val spark = SparkSession.builder.appName(name="SparkByExamples.com").getOrCreate()  
  
 val *data*=*spark*.read.csv("src/main/Data/complaints.csv")  
 *data*.createOrReplaceTempView("PERSON\_DATA")  
  
 //data.show()  
  
 val *df* = *spark*.sql("SELECT \_c3,count(\_c3) as c from PERSON\_DATA group by \_c3 ")  
 //df.show()  
 //val df1=spark.sql("SELECT \_c2,count(\_c2) as c from PERSON\_DATA group by \_c2 order by")  
  
 //val df1=df.orderBy('c').desc()  
 val *df1*=*df*.sort(*desc*("c")).limit(1)  
  
 *df1*.show()

OUTPUT



1. Which are the Top 5 states having the highest volume of complaints coming.

import org.apache.spark.sql.{SparkSession, functions}  
import usecase\_6.*spark*import org.apache.spark.sql.functions.*desc*object usecase\_7 extends App{  
  
 val *spark*: SparkSession = SparkSession.*builder*()  
 .master("local[\*]")  
 .appName("SparkByExamples.com")  
 .getOrCreate()  
  
 //val spark = SparkSession.builder.appName(name="SparkByExamples.com").getOrCreate()  
  
 val *data*=*spark*.read.csv("src/main/Data/complaints.csv")  
 *data*.createOrReplaceTempView("PERSON\_DATA")  
  
 val *df*=*spark*.sql("select \_c5 as state,count(\_C0) as c from PERSON\_DATA group by \_c5")  
 val *df1*=*df*.sort(*desc*("c")).limit(5)  
  
 *df1*.show()  
  
}

OUTPUT



1. Which are the Top 5 companies people complaining the most.

import org.apache.spark.sql.SparkSession  
import org.apache.spark.sql.functions.*desc*import usecase\_7.*df*object usecase\_8 extends App{  
  
 val *spark*: SparkSession = SparkSession.*builder*()  
 .master("local[\*]")  
 .appName("SparkByExamples.com")  
 .getOrCreate()  
  
 //val spark = SparkSession.builder.appName(name="SparkByExamples.com").getOrCreate()  
  
 val *data*=*spark*.read.csv("src/main/Data/complaints.csv")  
 *data*.createOrReplaceTempView("PERSON\_DATA")  
 val *df*=*spark*.sql("select \_c10 as company,count(\_C0) as c from PERSON\_DATA group by \_c10")  
 val *df1*=*df*.sort(*desc*("c")).limit(5)  
 *df1*.show()  
  
}

OUTPUT



1. Which product has the most number of complaints.

import org.apache.spark.sql.SparkSession  
import org.apache.spark.sql.functions.*desc*object usecase\_9 extends App{  
  
 val *spark*: SparkSession = SparkSession.*builder*()  
 .master("local[\*]")  
 .appName("SparkByExamples.com")  
 .getOrCreate()  
  
 //val spark = SparkSession.builder.appName(name="SparkByExamples.com").getOrCreate()  
  
 val *data*=*spark*.read.csv("src/main/Data/complaints.csv")  
 *data*.createOrReplaceTempView("PERSON\_DATA")  
 val *df*=*spark*.sql("select \_c1 as product,count(\_C0) as c from PERSON\_DATA group by \_c1")  
 val *df1*=*df*.sort(*desc*("c")).limit(1)  
 *df1*.show()  
  
}

OUTPUT

