//Start SQL and connect to JDBC

service mysql start

cd /root/spark/jars

wget https://storage.googleapis.com/infs3208/dockerenv/mysql-connector-java-5.1.48-bin.jar

chmod 777 mysql-connector-java-5.1.48-bin.jar

//Import related packages

import org.apache.spark.sql.SQLContext

import java.util.Properties

import java.sql.DriverManager

import java.sql.Connection

import org.apache.spark.sql.{Row, SparkSession}

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

import scala.collection.mutable

import java.text.SimpleDateFormatcd /root/spark/sbin

/ Configure MySQL connection

val spark = new SparkSession.Builder().appName("studentmangement").getOrCreate()

val sc = spark.sparkContext

var url = "jdbc:mysql://localhost:3306/hive"

val sqlContext = new SQLContext(sc)

val properties = new Properties()

properties.setProperty("user","hive")

properties.setProperty("password","hive")

properties.setProperty("useSSL","false")

//test connection

import spark.sqlContext.implicits.\_

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

import org.apache.spark.mllib.linalg.Vectors

import org.apache.spark.mllib.stat.Statistics

var tableName = "ROLES"

val dataFrame = sqlContext.read.jdbc(url, tableName,properties)

dataFrame.select("ROLE\_ID","ROLE\_NAME").show()

//create project database

mysql -u root

create user 'project'@'%' identified by 'project';

create schema project default character set utf8 collate utf8\_general\_ci;

GRANT ALL PRIVILEGES ON project.\* TO 'project'@'%';

flush privileges ;

//connect to the project database

properties.setProperty("user","project")

properties.setProperty("password","project")

url = "jdbc:mysql://localhost:3306/project"

//Import the library json file into database

val library\_df = spark.read.option("multiline", true).json("hdfs://sparkmaster:9000/test/librarydata.hdfs.json")

library\_df.show(12, false)

properties.setProperty("user","project")

properties.setProperty("password","project")

url = "jdbc:mysql://localhost:3306/project"

tableName = "library"

library\_df.write.mode("append").jdbc(url, tableName, properties)

//Chooes the library name, address,locality and email

url = "jdbc:mysql://localhost:3306/project"

tableName = "library"

val library\_df\_load = sqlContext.read.jdbc(url, tableName,properties)

library\_df\_load.select("library\_Name","address","locality", "email").show(6)

//Choose the libraries which has the prefer index bigger than 40

library\_df.createOrReplaceTempView("library")

val results = spark.sql("SELECT library\_name, address, wifi,locality, postcode,phone, prefer\_index FROM library WHERE prefer\_index>40")

results.show(12)

//Choose the libraries that cloes on Monday

val results = spark.sql("SELECT opening\_hours\_monday,library\_name, address, wifi,locality, postcode,phone

FROM library WHERE opening\_hours\_monday = 'Closed'")

results.show(5)

//Choose the libraries that not have the wifi

val results = spark.sql("SELECT library\_name, address, wifi,locality, postcode,phone, prefer\_index FROM library WHERE wifi = 'No'")

results.show(5)

//Choose the libraries has wifi

val results = spark.sql("SELECT library\_name, address, wifi,locality, postcode,phone, prefer\_index FROM library WHERE wifi = 'Yes'")

results.show(12)

library\_df.createOrReplaceTempView("library")

//Choose the library that has the best prefer index

val results = spark.sql("SELECT library\_name,address,email,phone,locality,postcode, wifi, prefer\_index FROMlibrary WHERE prefer\_index = (SELECT MAX(prefer\_index) FROM library)")

results.show(12)

//The rank list of whole libraries

val results = spark.sql("SELECT library\_name, address, locality, postcode,phone, prefer\_index FROM library ORDER BY prefer\_index DESC")

results.show(11)

//The top three libraries that have the highest prefer index

val results = spark.sql("SELECT library\_name, address, locality, postcode, phone

FROM (SELECT library\_name, address, locality, postcode,phone, prefer\_index FROM library ORDER BY prefer\_index DESC)")

results.show(3)

//The libraries that open on Saturday

val results = spark.sql("SELECT opening\_hours\_saturday, library\_name, wifi, address, locality, postcode, phone, email FROM library WHERE opening\_hours\_saturday <> 'Closed'")

results.show(12)

//The libraries that open on Sunday

val results = spark.sql("SELECT opening\_hours\_sunday, library\_name, wifi, address, locality, postcode, phone, email FROM library WHERE opening\_hours\_sunday <> 'Closed'")

results.show(12)

// ML: try to find which library users prefer if they want to go library in Toowong.

//Thus, these two libraries could organize some activity in common.

import org.apache.spark.ml.fpm.FPGrowth

val dataset = spark.createDataset(Seq(

"toowong indroopilly",

"indroopilly city sunnybank stlucia",

"toowong city sunnybank southbank",

"indroopilly toowong city sunnybank",

"indroopilly toowong city southbank",

"toowong indroopilly city")

).map(t => t.split(" ")).toDF("items")

val testDataset = spark.createDataset(Seq("toowong")).map(t => t.split(" ")).toDF("items")

val fpgrowth = new FPGrowth().setItemsCol("items").setMinSupport(0.6).setMinConfidence(0.6)

val model = fpgrowth.fit(dataset)

// Display frequent itemsets.

model.freqItemsets.show()

// Display generated association rules.

model.associationRules.show()

// transform examines the input items against all the association rules and summarize the

// consequents as prediction

model.transform(testDataset).show()