sqoop import-all-tables --connect jdbc:mysql://quickstart.cloudera:3306/retail\_db --username retail\_dba --password cloudera --hive-import --hive-database problem61 -m 1

val hiveContext = new org.apache.spark.sql.hive.HiveContext(sc)

hiveContext.sql("use problem61")

hiveContext.sql("select department\_id,product\_id,product\_price,rank() OVER (partition by department\_id order by product\_price) rnk, dense\_rank() OVER (partition by department\_id order by product\_price) dense\_rnk from products p join categories c on p.product\_category\_id = c.category\_id join departments d on c.category\_department\_id = d.department\_id order by department\_id,dense\_rnk desc,rnk desc")

hiveContext.sql("select department\_id,product\_id,product\_price,rank() OVER (partition by department\_id order by product\_price) rnk from products p join categories c on p.product\_category\_id = c.category\_id join departments d on c.category\_department\_id = d.department\_id order by department\_id,rnk desc")

hiveContext.sql("select customer\_id,COUNT(distinct(product\_id)) unique\_products from products p join order\_items oi on p.product\_id = oi.order\_item\_product\_id join orders o on oi.order\_item\_order\_id = o.order\_id join customers c on o.order\_customer\_id = c.customer\_id group by customer\_id order by unique\_products desc,customer\_id limit 10").show()

+-----------+---------------+

|customer\_id|unique\_products|

+-----------+---------------+

| 1288| 17|

| 1657| 17|

| 1810| 17|

| 12226| 17|

| 2292| 16|

| 2403| 16|

| 3161| 16|

| 5691| 16|

| 5715| 16|

| 8766| 16|

+-----------+---------------+

hiveContext.sql("select c.customer\_id, c.customer\_fname, count(distinct(oi.order\_item\_product\_id)) unique\_products from customers c inner join orders o on o.order\_customer\_id = c.customer\_id inner join order\_items oi on o.order\_id = oi.order\_item\_order\_id group by c.customer\_id, c.customer\_fname order by unique\_products desc, c.customer\_id limit 10").show()

case class Department(department\_id: Int, department\_name: String)

case class Category(category\_id: Int,category\_department\_id: Int, category\_name: String)

case class Customer(customer\_id :Int, customer\_fname: String,customer\_lname : String, customer\_email : String,customer\_password: String, customer\_street: String, customer\_city : String, customer\_state: String, customer\_zipcode: String)

case class Product(product\_id: Int,product\_category\_id: Int,product\_name: String,product\_description: String, product\_price: Float,product\_image : String)

case class Order\_Item(order\_item\_id: Int,order\_item\_order\_id: Int,order\_item\_product\_id: Int,order\_item\_quantity: Int,order\_item\_subtotal: Float, order\_item\_product\_price: Float)

case class Order(order\_id: Int,order\_date: String,order\_customer\_id: Int, order\_status: String)

val ordersRDD = sc.textFile("/user/cloudera1/problem6/retail\_db/orders")

val orderItemsRDD = sc.textFile("/user/cloudera1/problem6/retail\_db/order\_items")

val productsRDD = sc.textFile("/user/cloudera1/problem6/retail\_db/products")

val categoriesRDD = sc.textFile("/user/cloudera1/problem6/retail\_db/categories")

val customersRDD = sc.textFile("/user/cloudera1/problem6/retail\_db/customers")

val departmentsRDD = sc.textFile("/user/cloudera1/problem6/retail\_db/departments")

sqlContext.sql("select d.department\_id,p.product\_id,p.product\_price, RANK() OVER (partition by d.department\_id order by p.product\_price) rnk, DENSE\_RANK() OVER (PARTITION by d.department\_id order by p.product\_price) dense\_rnk from products p join categories c on p.product\_category\_id = c.category\_id join departments d on d.department\_id = c.category\_department\_id order by department\_id,rnk desc, dense\_rnk desc").show()

sqlContext.sql("select d.department\_id,p.product\_id,p.product\_price,rank() over(partition by d.department\_id order by p.product\_price) from products p join categories c on p.product\_category\_id = c.category\_id join departments d on d.department\_id = c.category\_department\_id order by department\_id").show()

sqlContext.sql("select rank() over(partition by product\_category\_id order by product\_price) from products p order by product\_category\_id").show()

val resultSQL = sqlContext.sql("select customer\_id,COUNT(distinct(product\_id)) unique\_products from products p join orderItems oi on p.product\_id = oi.order\_item\_product\_id join orders o on oi.order\_item\_order\_id = o.order\_id join customers c on o.order\_customer\_id = c.customer\_id group by customer\_id order by unique\_products desc,customer\_id desc limit 10")

resultSQL.registerTempTable("Unique\_Pro")

sqlContext.sql("select distinct p.\* from products p join orderItems oi on p.product\_id = oi.order\_item\_product\_id join orders o on oi.order\_item\_order\_id = o.order\_id join customers c on o.order\_customer\_id = c.customer\_id join Unique\_Pro up on c.customer\_id = up.customer\_id where p.product\_price < 100").show()

val hiveSQL = hiveContext.sql("select customer\_id,COUNT(distinct(product\_id)) unique\_products from retail\_db.products p join retail\_db.order\_items oi on p.product\_id = oi.order\_item\_product\_id join retail\_db.orders o on oi.order\_item\_order\_id = o.order\_id join retail\_db.customers c on o.order\_customer\_id = c.customer\_id group by customer\_id order by unique\_products desc,customer\_id desc limit 10")

hiveSQL.registerTempTable("uni")

hiveSQL.registerTempTable("create table retail\_db.Unique\_Pro as select \* from uni")

hiveContext.sql("select distinct p.\* from retail\_db.products p join retail\_db.order\_items oi on p.product\_id = oi.order\_item\_product\_id join retail\_db.orders o on oi.order\_item\_order\_id = o.order\_id join retail\_db.customers c on o.order\_customer\_id = c.customer\_id join retail\_db.Unique\_Pro up on c.customer\_id = up.customer\_id where p.product\_price < 100").show()