IMPORTING SQL

```
=================================================================
from pyspark import SQLContext
sqlContext = SQLContext(sc)
departmentsJson = sqlContext.jsonFile("/user/cloudera/pyspark/departments.json")
departmentsJson.registerTempTable("departmentsTable")
departmentsData = sqlContext.sql("select * from departmentsTable")
for rec in departmentsData.collect():
  print(rec)
=================================================================
```

SQL FROM PYSPARK

```
from pyspark.sql import HiveContext
sqlContext = HiveContext(sc)
depts = sqlContext.sql("select * from departments")
for rec in depts.collect():
  print(rec)

sqlContext.sql("create table departmentsSpark as select * from departments")
depts = sqlContext.sql("select * from departmentsSpark")
for rec in depts.collect():
  print(rec)
```

# Using Hive

```
===========================================================
from pyspark.sql import HiveContext
sqlContext = HiveContext(sc)
sqlContext.sql("set spark.sql.shuffle.partitions=10");

joinAggData = sqlContext.sql("select o.order_date,
round(sum(oi.order_item_subtotal), 2), \
count(distinct o.order_id) from orders o join order_items oi \
on o.order_id = oi.order_item_order_id \
group by o.order_date order by o.order_date")

for data in joinAggData.collect():
  print(data)
===========================================================
```

# Using Pyspark native sql

```
from pyspark.sql import SQLContext, Row
sqlContext = SQLContext(sc)
sqlContext.sql("set spark.sql.shuffle.partitions=10");

ordersRDD = sc.textFile("/user/cloudera/sqoop_import/orders")
ordersMap = ordersRDD.map(lambda o: o.split(","))
orders = ordersMap.map(lambda o: Row(order_id=int(o[0]), order_date=o[1], \
order_customer_id=int(o[2]), order_status=o[3]))
ordersSchema = sqlContext.inferSchema(orders)
```

```
ordersSchema.registerTempTable("orders")

orderItemsRDD = sc.textFile("/user/cloudera/sqoop_import/order_items")
orderItemsMap = orderItemsRDD.map(lambda oi: oi.split(","))
orderItems = orderItemsMap.map(lambda oi: Row(order_item_id=int(oi[0]),
order_item_order_id=int(oi[1]), \
order_item_product_id=int(oi[2]), order_item_quantity=int(oi[3]),
order_item_subtotal=float(oi[4]), \
order_item_product_price=float(oi[5])))
orderItemsSchema = sqlContext.inferSchema(orderItems)
orderItemsSchema.registerTempTable("order_items")

joinAggData = sqlContext.sql("select o.order_date, sum(oi.order_item_subtotal), \
count(distinct o.order_id) from orders o join order_items oi \
on o.order_id = oi.order_item_order_id \
group by o.order_date order by o.order_date")

for data in joinAggData.collect():
  print(data)
```
==

SORTING IN SQL

#Global sorting and ranking

```
select * from products order by product_price desc;
select * from products order by product_price desc limit 10;
```

#By key sorting

#Using order by is not efficient, it serializes

select * from products order by product_category_id, product_price desc;

#Using distribute by sort by (to distribute sorting and scale it up)

select * from products distribute by product_category_id sort by product_price desc;

#By key ranking (in Hive we can use windowing/analytic functions)

select * from (select p.*,

dense_rank() over (partition by product_category_id order by product_price desc) dr

from products p

distribute by product_category_id) q

===

GET NUMBER OF CANCLED ORDERS IN MOTH OF JANUVARY IN 2014

select order_status, count(1) from orders

where from_unixtime(cast(substr(order_date, 1, 10) as int)) like '2014-01%' group by order_status;

where dr <= 2 order by product_category_id, dr;