**SQL Scenarios**

**Common Code:**

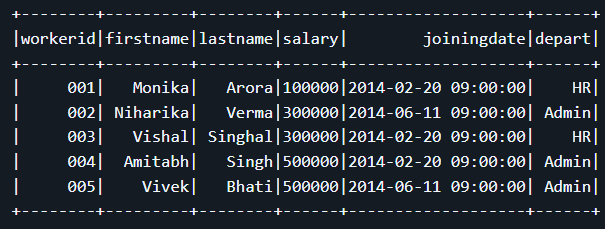
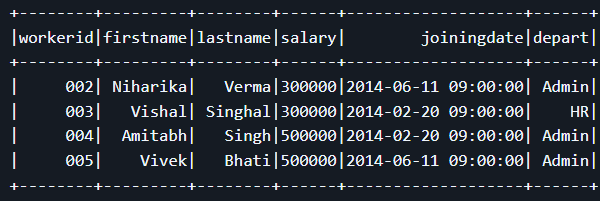
**================ ⬇️ Main Code ⬇️ =======================**

**from pyspark import SparkConf, SparkContext  
from pyspark.sql import SparkSession  
from pyspark.sql.functions import \*  
import sys  
import os  
  
python\_path = sys.executable  
os.environ['PYSPARK\_PYTHON'] = python\_path  
os.environ['JAVA\_HOME'] = r'C:\Users\HP\.jdks\corretto-1.8.0\_462' <----- 🔴JAVA PATH🔴  
  
conf = SparkConf().setAppName("pyspark").setMaster("local[\*]").set("spark.driver.host","localhost").set("spark.default.parallelism", "1")  
sc = SparkContext(conf=conf)  
spark = SparkSession.builder.getOrCreate()**

**================ ⬆️ Main Code ⬆️ =======================**

**Scenario 1:** **Write Query to get who are getting equal salary.**

**Input Output**

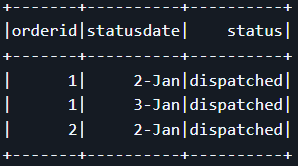
****

**Solution**:

**data** = [  
 ("001", "Monika", "Arora", 100000, "2014-02-20", "09:00:00", "HR"),  
 ("002", "Niharika", "Verma", 300000, "2014-06-11", "09:00:00", "Admin"),  
 ("003", "Vishal", "Singhal", 300000, "2014-02-20", "09:00:00", "HR"),  
 ("004", "Amitabh", "Singh", 500000, "2014-02-20", "09:00:00", "Admin"),  
 ("005", "Vivek", "Bhati", 500000, "2014-06-11", "09:00:00", "Admin")  
]  
  
  
**df** = spark.createDataFrame(data, ["Id","FirstName","LastName","Salary","JoiningDate","Time","Department"])  
**df.show()**  
  
**countdf** = df.groupBy("Salary").count().filter("count>1")  
  
**finaldf** = df.join(countdf,"Salary","inner").select("Id", "Firstname", "Lastname", "Salary", "joiningDate", "Department")  
**finaldf.show()**

**Scenario 2: Need the dates when the status gets changed like ordered to dispatched.**

**Input Output**

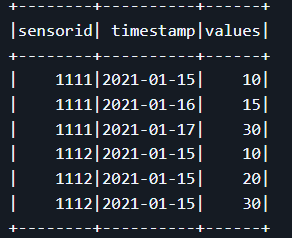
****

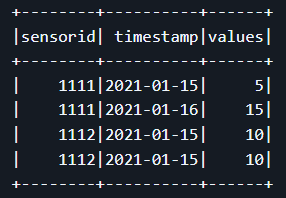
**Solution:**

**data** = [  
 ("1","1-Jan","Ordered"),  
 ("1","2-Jan","dispatched"),  
 ("1","3-Jan","dispatched"),  
 ("1","4-Jan","Shipped"),  
 ("1","5-Jan","Shipped"),  
 ("1","6-Jan","Delivered"),  
 ("2","1-Jan","Ordered"),  
 ("2","2-Jan","dispatched"),  
 ("2","3-Jan","Shipped")  
]

**df** = spark.createDataFrame(data,["orderid","statusdate","status"])  
**df.show()**  
  
**dispatcheddf** = df.filter("status = 'dispatched'")  
**dispatcheddf.show()**

**Scenario 3: Write a query to calculate the difference between consecutive sensor values per sensorid.**

** Input Output**

****

**Solution:**

**from pyspark.sql.functions import \*  
from pyspark.sql.window import \***

**data** = [  
 ("1111","2021-01-15","10"),  
 ("1111","2021-01-16","15"),  
 ("1111","2021-01-17","30"),  
 ("1112","2021-01-15","10"),  
 ("1112","2021-01-15","20"),  
 ("1112","2021-01-15","30")  
]  
  
**df** = spark.createDataFrame(data,["sensorid","timestamp","values"])  
**df.show()**  
  
**windowdf** = Window.partitionBy("sensorid").orderBy("values")

# partitioned the sensorid column for gouping  
  
**nextdf** = df.withColumn("nextvalues", lead("values", 1).over(windowdf))

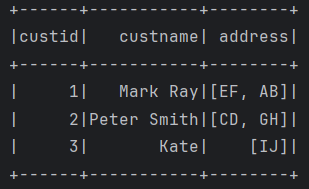
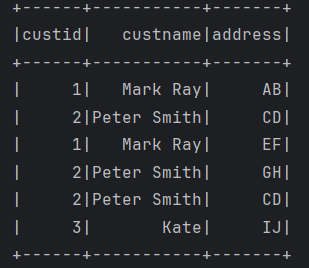
# adding new column  
  
**filterdf** = nextdf.filter(col("nextvalues").isNotNull())

# filter out the null column  
  
**finaldf** = filterdf.withColumn("values", expr("nextvalues-values" ).cast("int")).drop("nextvalues"). orderBy(col("sensorid"))

**finaldf.show()**

**Q: Write a query to list the unique customer names in the custtab table, along with the number of addresses associated with each customer.**

**Input Output**

****

**Solution:**

**data** = [  
 ("1","Mark Ray","AB"),  
 ("2","Peter Smith","CD"),  
 ("1","Mark Ray","EF"),  
 ("2","Peter Smith","GH"),  
 ("2","Peter Smith","CD"),  
 ("3","Kate","IJ")  
]  
  
**df** = spark.createDataFrame(data, ["custid","custname","address"])  
**df.show()**  
  
**finaldf** = df.groupBy("custid","custname").agg(collect\_set("address").alias("address")). orderBy("custid")

**finaldf.show()**