ASSIGNMENT SOLUTION

The following Common Boilerplate code to create a Spark Session has to be executed before running the queries.

```
from pyspark.sql import SparkSession

import getpass

username = getpass.getuser()

spark = SparkSession. \

builder. \

config('spark.ui.port', '0'). \

config("spark.sql.warehouse.dir", f"/user/itv005357/warehouse"). \

enableHiveSupport(). \

master('yarn'). \

getOrCreate()
```

Question 1:

1. we need to find top 10 customers who have spent the most amount (premium customers)

```
#Loading the data to RDD's

orders_rdd = spark.sparkContext.textFile("/public/trendytech/retail_db/orders/*")

order_items_rdd = spark.sparkContext.textFile("/public/trendytech/retail_db/order_items/*")

#Taking the order_id and order_item_subtotal from order_items_rdd

order_items_map = order_items_rdd.map(lambda x: ((int(x.split(',')[1]),float(x.split(',')[4]))))

#Taking the order_id and order_customer_id from orders_rdd

orders_map = orders_rdd.map(lambda x: (int(x.split(',')[0]),(int(x.split(',')[2]))))

#Joining the 2 rdd's using the common column – order_id

join_rdd = order_items_map.join(orders_map)

[

join_rdd looks like below:
```

[(4, (49.98, 8827)),

(4, (299.95, 8827)),

(4, (150.0, 8827)),

(4, (199.92, 8827)),

(8, (179.97, 2911))]

That is,

[(order_id, (order_item_subtotal,order_customer_id))]

[(x[0] , (x[1][0] , x[1][1]))]

So here we are taking order_customer_id and order_item_subtotal.

So resultant rdd should be = (x[1][1],x[1][0])

]

```
mapped_rdd = join_rdd.map(lambda x: (x[1][1],x[1][0]))
```

#Now we need to sum up the amount for each customer and then sort it in descending order.

```
reduced_rdd = mapped_rdd.reduceByKey(lambda x,y : x+y).sortBy(lambda x: x[1], ascending=False)
reduced_rdd.take(10)
```

```
from pyspark.sql import SparkSession
import getpass
username = getpass.getuser()
spark = SparkSession. \
builder. \
config('spark.ui.port', '0'). \
config("spark.sql.warehouse.dir", f"/user/itv005357/warehouse"). \
enableHiveSupport(). \
master('yarn'). \
getOrCreate()
```

```
orders_rdd = spark.sparkContext.textFile("/public/trendytech/retail_db/orders/*")
order_items_rdd = spark.sparkContext.textFile("/public/trendytech/retail_db/order_items/*")
```

```
order_items_map = order_items_rdd.map(lambda x: ((int(x.split(',')[1]),float(x.split(',')[4]))))
```

```
order_items_map.take(5)
```

```
[(1, 299.98), (2, 199.99), (2, 250.0), (2, 129.99), (4, 49.98)]
```

```
orders_map = orders_rdd.map(lambda x: (int(x.split(',')[0]),(int(x.split(',')[2]))))
```

```
orders_map.take(5)
```

```
[(1, 11599), (2, 256), (3, 12111), (4, 8827), (5, 11318)]
```

```
join_rdd = order_items_map.join(orders_map)
```

```
join_rdd.take(5)
```

```
[(4, (49.98, 8827)),
 (4, (299.95, 8827)),
 (4, (150.0, 8827)),
 (4, (199.92, 8827)),
 (8, (179.97, 2911))]
```

```
mapped_rdd = join_rdd.map(lambda x: (x[1][1],x[1][0]))
```

```
mapped_rdd.take(5)
```

```
[(8827, 49.98), (8827, 299.95), (8827, 150.0), (8827, 199.92), (2911, 179.97)]
```

```
reduced_rdd = mapped_rdd.reduceByKey(lambda x,y : x+y).sortBy(lambda x: x[1], ascending=False)
```

```
reduced_rdd.take(10)
```

```
[(791, 10524.169999999996),
 (9371, 9299.029999999999),
 (8766, 9296.14),
 (1657, 9223.71),
 (2641, 9130.92),
 (1288, 9019.11),
 (3710, 9019.099999999999),
 (4249, 8918.85),
 (5654, 8904.95),
 (5624, 8761.98)]
```

2. top 10 product id's with most quantities sold

#Loading data to RDD

order_items_rdd = spark.sparkContext.textFile("/public/trendytech/retail_db/order_items/*")

#We are taking order_item_product_id, order_item_quantity columns and then adding the quantities of each product item and sorting it.

mapped_rdd = order_items_rdd.map(lambda x: ((int(x.split(",")[2])),(int(x.split(",")[3]))))

reduced_rdd = mapped_rdd.reduceByKey(lambda x, y: x + y)

top_products = reduced_rdd.sortBy(lambda x: x[1], ascending=False)

top_products.take(10)

```
order_items_rdd = spark.sparkContext.textFile("/public/trendytech/retail_db/order_items/*")
```

```
mapped_rdd = order_items_rdd.map(lambda x: ((int(x.split(",")[2])),(int(x.split(",")[3]))))
```

```
mapped_rdd.take(5)
```

```
[(957, 1), (1073, 1), (502, 5), (403, 1), (897, 2)]
```

```
reduced_rdd = mapped_rdd.reduceByKey(lambda x, y: x + y)
```

```
top_products = reduced_rdd.sortBy(lambda x: x[1], ascending=False)
```

```
top_products.take(10)
```

```
[(365, 73698),
 (502, 62956),
 (1014, 57803),
 (191, 36680),
 (627, 31735),
 (403, 22246),
 (1004, 17325),
 (1073, 15500),
 (957, 13729),
 (977, 998)]
```

3. how many customers are from Caguas city

#Loading data to RDD

customers_rdd = spark.sparkContext.textFile("/public/trendytech/retail_db/customers/*")

#Taking customer_city and filtering based on city names = Caguas

mapped_rdd = customers_rdd.map(lambda x: x.split(",")[6])

filtered_rdd = mapped_rdd.filter(lambda x: x == 'Caguas')

#Calling count() action

filtered_rdd.count()

```
customers_rdd = spark.sparkContext.textFile("/public/trendytech/retail_db/customers/*")
```

```
mapped_rdd = customers_rdd.map(lambda x: x.split(",")[6])
```

```
mapped_rdd.take(5)
```

```
['Brownsville', 'Littleton', 'Caguas', 'San Marcos', 'Caguas']
```

```
filtered_rdd = mapped_rdd.filter(lambda x: x == 'Caguas')
```

```
filtered_rdd.count()
```

```
4584
```

4. Top 3 states with maximum customers

#Loading data to RDD

customers_rdd = spark.sparkContext.textFile("/public/trendytech/retail_db/customers/*")

#Taking customer_states from customers_rdd and adding and sorting to get top 3 states

mapped_rdd = customers_rdd.map(lambda x: (x.split(",")[7],1))

reduced_rdd = mapped_rdd.reduceByKey(lambda x, y: x + y)

sorted_rdd = reduced_rdd.sortBy(lambda x: x[1], ascending=False)

sorted_rdd.take(3)

```
customers_rdd = spark.sparkContext.textFile("/public/trendytech/retail_db/customers/*")
```

```
mapped_rdd = customers_rdd.map(lambda x: (x.split(",")[7],1))
```

```
mapped_rdd.take(5)
```

```
[('TX', 1), ('CO', 1), ('PR', 1), ('CA', 1), ('PR', 1)]
```

```
reduced_rdd = mapped_rdd.reduceByKey(lambda x, y: x + y)
```

```
sorted_rdd = reduced_rdd.sortBy(lambda x: x[1], ascending=False)
```

```
sorted_rdd.take(3)
```

```
[('PR', 4771), ('CA', 2012), ('NY', 775)]
```

5. how many customers have spent more than $1000 in total

#Loading the data to RDD's

```
orders_rdd = spark.sparkContext.textFile("/public/trendytech/retail_db/orders/*")

order_items_rdd = spark.sparkContext.textFile("/public/trendytech/retail_db/order_items/*")
```

#Taking the order_id and order_item_subtotal from order_items_rdd

```
order_items_map = order_items_rdd.map(lambda x: ((int(x.split(',')[1]),float(x.split(',')[4]))))
```

#Taking the order_id and order_customer_id from orders_rdd

```
orders_map = orders_rdd.map(lambda x: (int(x.split(',')[0]),(int(x.split(',')[2]))))
```

#Joining the 2 rdd's using the common column – order_id

```
join_rdd = order_items_map.join(orders_map)
```

[

join_rdd looks like below:

```
[(35212, (49.98, 8774)),
 (35212, (299.97, 8774)),
 (35212, (249.9, 8774)),
 (35212, (49.98, 8774)),
 (35212, (149.94, 8774))]
```

That is,

[(order_id, (order_item_subtotal,order_customer_id))]

[(x[0] , (x[1][0] , x[1][1]))]

So here we are taking order_customer_id and order_item_subtotal.

So resultant rdd should be = (x[1][1],x[1][0])

]

taking order_customer_id and order_item_subtotal.

```
mapped_rdd = join_rdd.map(lambda x: (x[1][1],x[1][0]))
```

sum up the amount for each customer

```
reduced_rdd = mapped_rdd.reduceByKey(lambda x,y : x+y)
```

filter the records having order_item_subtotal >1000 and counting it.

#Here we are caching the results for optimization

```
final_rdd = reduced_rdd.filter(lambda x: x[1] > 1000).cache()

final_rdd.count()
```

```
orders_rdd = spark.sparkContext.textFile("/public/trendytech/retail_db/orders/*")
order_items_rdd = spark.sparkContext.textFile("/public/trendytech/retail_db/order_items/*")
```

```
order_items_map = order_items_rdd.map(lambda x: ((int(x.split(',')[1]),float(x.split(',')[4]))))
orders_map = orders_rdd.map(lambda x: (int(x.split(',')[0]),(int(x.split(',')[2]))))
```

```
join_rdd = order_items_map.join(orders_map)
```

```
join_rdd.take(5)
```

```
[(35212, (49.98, 8774)),
 (35212, (299.97, 8774)),
 (35212, (249.9, 8774)),
 (35212, (49.98, 8774)),
 (35212, (149.94, 8774))]
```

```
mapped_rdd = join_rdd.map(lambda x: (x[1][1],x[1][0]))
reduced_rdd = mapped_rdd.reduceByKey(lambda x,y : x+y)
final_rdd = reduced_rdd.filter(lambda x: x[1] > 1000).cache()
final_rdd.count()
```

```
11148
```

6. which state has most number of orders in CLOSED status

#Load data to RDD

orders_rdd = spark.sparkContext.textFile("/public/trendytech/retail_db/orders/*")

customers_rdd = spark.sparkContext.textFile("/public/trendytech/retail_db/customers/*")

#Taking order_customer_id and order_status columns and extracting only CLOSED orders from orders_rdd

orders_map = orders_rdd.map(lambda x: ((int(x.split(',')[2])),(x.split(',')[3]))).filter(lambda x: x[1] == 'CLOSED')

#Taking customer_id and customer_state columns from customer_rdd

customers_map = customers_rdd.map(lambda x: (int(x.split(',')[0]),x.split(',')[7]))

#Joining 2 Rdd's by common column – customer_id

join_data = orders_map.join(customers_map)

[

Joined rdd looks like below:

[(5116, ('CLOSED', 'MA')),

(5116, ('CLOSED', 'MA')),

(10604, ('CLOSED', 'NC')),

(10604, ('CLOSED', 'NC')),

(16, ('CLOSED', 'PR'))]

[(customer_id, (order_status , customer_state))]

[(x[0] , (x[1][0] , x[1][1]))]

here we need to take customer_state alone -> x[1][1]

And we need to group each state and sum it up, for that we map to -> (x[1][1],1)

After performing reduceByKey on (x[1][1],1) each state gets grouped.

]

mapped_rdd = join_data.map(lambda x: (x[1][1],1))

final_rdd = mapped_rdd.reduceByKey(lambda x,y:x+y).sortBy(lambda x: x[1], ascending=False)

final_rdd.take(1)

```
orders_rdd = spark.sparkContext.textFile("/public/trendytech/retail_db/orders/*")
customers_rdd = spark.sparkContext.textFile("/public/trendytech/retail_db/customers/*")
```

```
orders_map = orders_rdd.map(lambda x: ((int(x.split(',')[2])),(x.split(',')[3]))).filter(lambda x: x[1] == 'CLOSED')
```

```
orders_map.take(5)
```

```
[(11599, 'CLOSED'),
 (8827, 'CLOSED'),
 (1837, 'CLOSED'),
 (1205, 'CLOSED'),
 (11441, 'CLOSED')]
```

```
customers_map = customers_rdd.map(lambda x: (int(x.split(',')[0]),x.split(',')[7]))
```

```
customers_map.take(5)
```

```
[(1, 'TX'), (2, 'CO'), (3, 'PR'), (4, 'CA'), (5, 'PR')]
```

```
join_data = orders_map.join(customers_map)
```

```
join_data.take(5)
```

```
[(5116, ('CLOSED', 'MA')),
 (5116, ('CLOSED', 'MA')),
 (10604, ('CLOSED', 'NC')),
 (10604, ('CLOSED', 'NC')),
 (16, ('CLOSED', 'PR'))]
```

```
mapped_rdd = join_data.map(lambda x: (x[1][1],1))
```

```
mapped_rdd.take(5)
```

```
[('MA', 1), ('MA', 1), ('NC', 1), ('NC', 1), ('PR', 1)]
```

```
final_rdd = mapped_rdd.reduceByKey(lambda x,y:x+y).sortBy(lambda x: x[1], ascending=False)
```

```
final_rdd.take(1)
```

```
[('PR', 2891)]
```

7. how many customers are active (active customers are the one's who placed atleast one order)

#Load data to RDD

orders_rdd= spark.sparkContext.textFile("/public/trendytech/retail_db/orders/*")

mapped_rdd = orders_rdd.map(lambda x: (int(x.split(",")[2]),1))

It then reduces the RDD by key, summing up the values

reduced_rdd = mapped_rdd.reduceByKey(lambda x,y: x + y)

#Finally, it filters the RDD to include only customers who have made at least one order

filtered_rdd = reduced_rdd.filter(lambda x: x[1] >= 1)

filtered_rdd.count()

```
orders_rdd= spark.sparkContext.textFile("/public/trendytech/retail_db/orders/*")
```

```
mapped_rdd = orders_rdd.map(lambda x: (int(x.split(",")[2]),1))
```

```
mapped_rdd.take(5)
```

```
[(11599, 1), (256, 1), (12111, 1), (8827, 1), (11318, 1)]
```

```
reduced_rdd = mapped_rdd.reduceByKey(lambda x,y: x + y)
reduced_rdd.take(5)
```

```
[(256, 10), (11318, 6), (7130, 7), (4530, 10), (5648, 13)]
```

```
filtered_rdd = reduced_rdd.filter(lambda x: x[1] >= 1)
```

```
filtered_rdd.count()
```

```
12405
```

8. What is the revenue generated by each state in sorted order.

#Load data to RDD

orders_rdd = spark.sparkContext.textFile("/public/trendytech/retail_db/orders/*")

customers_rdd = spark.sparkContext.textFile("/public/trendytech/retail_db/customers/*")

order_items_rdd = spark.sparkContext.textFile("/public/trendytech/retail_db/order_items/*")

#Taking order_customer_id and order_id from orders_rdd

```
orders_map = orders_rdd.map(lambda x: ((int(x.split(',')[2])),int(x.split(',')[0])))
#Taking customer_id and customer_state from customers_rdd
customers_map = customers_rdd.map(lambda x: ((int(x.split(',')[0])),x.split(',')[7]))
#Join the 2 rdds using the common column – customer_id
join_rdd = orders_map.join(customers_map)
```

[

join_rdd looks like below:

```
[(1868, (34574, 'NY')),
 (1868, (44606, 'NY')),
 (1868, (48859, 'NY')),
 (1868, (3571, 'NY')),
 (1868, (8201, 'NY'))]
```

[(customer_id , (order_id , customer_state))]

[(x[0] , (x[1][0] , x[1][1]))]

Now we can take only order_id and customer_state.

Here we have taken order_id inorder to join with order_items_rdd which is having a common column odrer_id.

]

```
mapped_rdd =  join_rdd.map(lambda x : (x[1][0],x[1][1]))
#Taking order_id and order_item_subtotal from order_items_rdd
order_items_map = order_items_rdd.map(lambda x: (int(x.split(',')[1]),float(x.split(',')[4])))
#Now join the 2 rdds with the common column – order_id
join_new_rdd = mapped_rdd.join(order_items_map)
```

[

join_new_rdd looks like below:

[(6756, ('TN', 399.98)),

(6756, ('TN', 129.99)),

(6756, ('TN', 299.97)),

(6756, ('TN', 399.98)),

(8430, ('TN', 29.97))]

[(order_id , (customer_state, order_item_subtotal))]

[(x[0] , (x[1][0] , x[1][1]))]

we can sum up order_item_subtotal for each state]

reduced_rdd = join_new_rdd.map(lambda x : (x[1][0],x[1][1])).reduceByKey(lambda x,y:x+y)

final_rdd = reduced_rdd.sortBy(lambda x: x[1], ascending=False)

final_rdd.collect()

```
orders_rdd = spark.sparkContext.textFile("/public/trendytech/retail_db/orders/*")
customers_rdd = spark.sparkContext.textFile("/public/trendytech/retail_db/customers/*")
order_items_rdd = spark.sparkContext.textFile("/public/trendytech/retail_db/order_items/*")
```

```
orders_map = orders_rdd.map(lambda x: ((int(x.split(',')[2])),int(x.split(',')[0])))
```

```
orders_map.take(5)
```

```
[(11599, 1), (256, 2), (12111, 3), (8827, 4), (11318, 5)]
```

```
customers_map = customers_rdd.map(lambda x: ((int(x.split(',')[0])),x.split(',')[7]))
```

```
customers_map.take(5)
```

```
[(1, 'TX'), (2, 'CO'), (3, 'PR'), (4, 'CA'), (5, 'PR')]
```

```
join_rdd = orders_map.join(customers_map)
```

```
join_rdd.take(5)
```

```
[(1868, (34574, 'NY')),
 (1868, (44606, 'NY')),
 (1868, (48859, 'NY')),
 (1868, (3571, 'NY')),
 (1868, (8201, 'NY'))]
```

```
mapped_rdd =  join_rdd.map(lambda x : (x[1][0],x[1][1]))
```

```
mapped_rdd.take(5)
```

```
[(34574, 'NY'), (44606, 'NY'), (48859, 'NY'), (3571, 'NY'), (8201, 'NY')]
```

```
order_items_map = order_items_rdd.map(lambda x: (int(x.split(',')[1]),float(x.split(',')[4])))
```

```
order_items_map.take(5)
```

```
[(1, 299.98), (2, 199.99), (2, 250.0), (2, 129.99), (4, 49.98)]
```

```
join_new_rdd = mapped_rdd.join(order_items_map)
```

```
join_new_rdd.take(5)
```

```
[(6756, ('TN', 399.98)),
 (6756, ('TN', 129.99)),
 (6756, ('TN', 299.97)),
 (6756, ('TN', 399.98)),
 (8430, ('TN', 29.97))]
```

```
reduced_rdd = join_new_rdd.map(lambda x : (x[1][0],x[1][1])).reduceByKey(lambda x,y:x+y)
```

```
reduced_rdd.take(5)
```

```
[('CA', 5542722.999999756),
 ('NY', 2152706.7399999835),
 ('VA', 344824.3500000002),
 ('CT', 211264.24000000008),
 ('NC', 378877.6400000002)]
```

```
final_rdd = reduced_rdd.sortBy(lambda x: x[1], ascending=False)
final_rdd.collect()
```

```
[('PR', 13208867.68999927),
 ('CA', 5542722.999999756),
 ('NY', 2152706.7399999835),
 ('TX', 1731407.489999999),
 ('IL', 1457225.8300000029),
 ('FL', 1048609.7700000026),
 ('OH', 773804.1100000018),
 ('MI', 730078.9700000018),
 ('PA', 724375.9300000016),
 ('NJ', 606550.990000001),
 ('AZ', 566459.290000001),
 ('GA', 467765.1800000007),
 ('MD', 456100.4200000007),
 ('NC', 378877.6400000002),
 ('CO', 358310.6000000003),
 ('VA', 344824.3500000002),
 ('OR', 315239.5100000001),
 ('MA', 306025.7300000002),
 ('TN', 297614.41000000015),
 ('NV', 276364.9700000001),
 ('MO', 260417.28000000012)
```

Question 2:

1.Find the top 10 states with the highest no. of positive cases.

#Load data to RDD

rdd = spark.sparkContext.textFile("/public/trendytech/covid19/cases/covid_dataset_cases.csv")

#Take state and positive columns from rdd

filtered_rdd = rdd.map(lambda x : (x.split(",")[1],int(x.split(",")[2])))

positive_cases = filtered_rdd.reduceByKey(lambda x,y : x + y).sortBy(lambda x: x[1], ascending=False)

positive_cases.take(10)

```
[1]: from pyspark.sql import SparkSession
     import getpass
     username = getpass.getuser()
     spark = SparkSession. \
     builder. \
     config('spark.ui.port', '0'). \
     config("spark.sql.warehouse.dir", f"/user/itv005357/warehouse"). \
     enableHiveSupport(). \
     master('yarn'). \
     getOrCreate()

[2]: rdd = spark.sparkContext.textFile("/public/trendytech/covid19/cases/covid_dataset_cases.csv")

[3]: rdd.take(3)

[3]: ['20200122,AP,2,0,48,26,15,18,2,38,10,34,B,18,19/05/2022,23,24,29,34,19,45,5,44,42,49,53,0,0,2,2,0,2,0,0,8f8db794931706272489cddd51e917a4a69c8c9b,0,0,0,0,0',
      '20200123,AP,2,0,48,41,2,20,30,40,5,50,B,1,08/11/2022,14,7,33,36,14,18,36,37,45,8,53,0,0,2,2,0,2,0,0,e16af2a6a8f060355ff5ba499a28309a262c0b1e,0,0,0,0,0',
      '20200124,HP,2,0,16,14,5,29,43,22,11,11,D,31,17/05/2022,10,37,11,25,45,25,2,32,30,41,53,0,0,2,2,0,2,0,0,094154f68e74bfc30b977cdee888f9c07be4360e,0,0,0,0,0']

[4]: filtered_rdd = rdd.map(lambda x : (x.split(",")[1],int(x.split(",")[2])))
     positive_cases = filtered_rdd.reduceByKey(lambda x,y : x + y).sortBy(lambda x: x[1], ascending=False)
     positive_cases.take(10)

[4]: [('WA', 1701),
      ('GA', 1017),
      ('MH', 730),
      ('MI', 61),
      ('CA', 53),
      ('GJ', 35),
      ('BR', 23),
      ('JH', 13),
      ('CG', 8),
      ('RI', 6)]
```

2.Find the total count of people in ICU currently

rdd_2 = spark.sparkContext.textFile("/public/trendytech/covid19/cases/covid_dataset_cases.csv")

icu_count = rdd_2.map(lambda x: int(x.split(",")[7])).sum()

print(icu_count)

```
[5]: rdd_2 = spark.sparkContext.textFile("/public/trendytech/covid19/cases/covid_dataset_cases.csv")

[6]: icu_count = rdd_2.map(lambda x: int(x.split(",")[7])).sum()

[7]: print(icu_count)

     1344
```

3.Find the top 15 States having maximum no. of recovery.

rdd_3 = spark.sparkContext.textFile("/public/trendytech/covid19/cases/covid_dataset_cases.csv")

recovered_rdd = rdd_3.map(lambda x: (x.split(",")[1],int(x.split(",")[11]))).reduceByKey(lambda x,y: x+y)

sorted_rdd = recovered_rdd.sortBy(lambda x: x[1], ascending=False)

sorted_rdd.take(15)

```
[8]: rdd_3 = spark.sparkContext.textFile("/public/trendytech/covid19/cases/covid_dataset_cases.csv")

[9]: recovered_rdd = rdd_3.map(lambda x: (x.split(",")[1],int(x.split(",")[11]))).reduceByKey(lambda x,y: x+y)

[10]: sorted_rdd = recovered_rdd.sortBy(lambda x: x[1], ascending=False)

[11]: sorted_rdd.take(15)

[11]: [('WA', 451),
       ('MH', 165),
       ('MI', 101),
       ('GA', 87),
       ('AP', 84),
       ('RI', 72),
       ('BR', 68),
       ('JH', 50),
       ('KA', 43),
       ('AZ', 38),
       ('AS', 30),
       ('GJ', 27),
       ('CA', 23),
       ('HR', 20),
       ('HP', 19)]
```

4.Find the top 3 States having least no. of deaths.

rdd_4 = spark.sparkContext.textFile("/public/trendytech/covid19/cases/covid_dataset_cases.csv")

death_rdd = rdd_4.map(lambda x: (x.split(",")[1],int(x.split(",")[23]))).reduceByKey(lambda x,y: x+y)

least_rdd = death_rdd.sortBy(lambda x: x[1])

least_rdd.take(3)

```
[12]: rdd_4 = spark.sparkContext.textFile("/public/trendytech/covid19/cases/covid_dataset_cases.csv")

[13]: death_rdd = rdd_4.map(lambda x: (x.split(",")[1],int(x.split(",")[23]))).reduceByKey(lambda x,y: x+y)

[14]: least_rdd = death_rdd.sortBy(lambda x: x[1])

[15]: least_rdd.take(3)

[15]: [('AS', 9), ('JH', 10), ('CG', 31)]
```

5.Find the total number of people hospitalized currently.

rdd_5 = spark.sparkContext.textFile("/public/trendytech/covid19/cases/covid_dataset_cases.csv")

hosp_rdd = rdd_5.map(lambda x: int(x.split(",")[5])).sum()

print(hosp_rdd)

```
[16]: rdd_5 = spark.sparkContext.textFile("/public/trendytech/covid19/cases/covid_dataset_cases.csv")

[17]: hosp_rdd = rdd_5.map(lambda x: int(x.split(",")[5])).sum()

[18]: print(hosp_rdd)

      1319
```

6.List the twitter handle and fips code for the top 15 states with the highest number of total cases.

#Load data to RDD

cases_rdd = spark.sparkContext.textFile("/public/trendytech/covid19/cases/covid_dataset_cases.csv")

states_rdd = spark.sparkContext.textFile("/public/trendytech/covid19/states/covid_dataset_states.csv")

#Taking state, twitter and fips columns from states_rdd

states_mapped =states_rdd.map(lambda x: (x.split(",")[0],(x.split(",")[5],int(x.split(",")[8]))))

#Taking state, total columns from cases_rdd

rdd1 = cases_rdd.map(lambda x: (x.split(",")[1],int(x.split(",")[28])))

#Summing up total cases for each states

total_cases = rdd1.reduceByKey(lambda x, y: x + y)

#Joining the 2 rdds based on common column -> state

joined_rdd = total_cases.join(states_mapped)

[

Joined_rdd looks like below:

[('AS', (2, ('@ASCovid', 6))),

('GJ', (35, ('@GJCovid', 44))),

('MH', (730, ('@MHCovid', 26))),

('HR', (2, ('@HRCovid', 9))),

('KA', (5, ('@KACovid', 53)))]

[(state, (total , (twitter , fips))]

[(x[0], (x[1][0] , (x[1][1][0] , x[1][1][1]))]

so we need to sort it based on total -> x[1][0]

]

final_rdd = joined_rdd.sortBy(lambda x: x[1][0], ascending=False)

final_rdd.take(15)

```
[19]: cases_rdd = spark.sparkContext.textFile("/public/trendytech/covid19/cases/covid_dataset_cases.csv")

[20]: states_rdd = spark.sparkContext.textFile("/public/trendytech/covid19/states/covid_dataset_states.csv")

[21]: states_mapped =states_rdd.map(lambda x: (x.split(",")[0],(x.split(",")[5],int(x.split(",")[8]))))

[27]: states_mapped.take(5)

[27]: [('HP', ('@HPCovid', 53)),
       ('AS', ('@ASCovid', 6)),
       ('HR', ('@HRCovid', 9)),
       ('KA', ('@KACovid', 53)),
       ('WA', ('@WACovid', 44))]
```

[22]:
```
rdd1 = cases_rdd.map(lambda x: (x.split(",")[1],int(x.split(",")[28])))
```

[23]:
```
total_cases = rdd1.reduceByKey(lambda x, y: x + y)
```

[24]:
```
joined_rdd = total_cases.join(states_mapped)
```

[28]:
```
joined_rdd.take(5)
```

[28]:
```
[('AS', (2, ('@ASCovid', 6))),
 ('GJ', (35, ('@GJCovid', 44))),
 ('MH', (730, ('@MHCovid', 26))),
 ('HR', (2, ('@HRCovid', 9))),
 ('KA', (5, ('@KACovid', 53)))]
```

[25]:
```
final_rdd = joined_rdd.sortBy(lambda x: x[1][0], ascending=False)
```

[26]:
```
final_rdd.take(15)
```

[26]:
```
[('WA', (2100, ('@WACovid', 44))),
 ('GA', (1034, ('@GACovid', 44))),
 ('MH', (730, ('@MHCovid', 26))),
 ('CA', (515, ('@CACovid', 4))),
 ('MI', (61, ('@MICovid', 53))),
 ('GJ', (35, ('@GJCovid', 44))),
 ('AZ', (34, ('@AZCovid', 53))),
 ('BR', (23, ('@BRCovid', 53))),
 ('RI', (16, ('@RICovid', 26))),
 ('JH', (13, ('@JHCovid', 53))),
 ('CG', (8, ('@CGCovid', 53))),
 ('KA', (5, ('@KACovid', 53))),
 ('HP', (4, ('@HPCovid', 53))),
 ('AS', (2, ('@ASCovid', 6))),
 ('HR', (2, ('@HRCovid', 9)))]
```

Question 3:

1.Find the top 20 words from Trendytech Students Google Reviews excluding the boring words.

First move the boring words data from edge node to HDFS.

hadoop fs -mkdir TT

hadoop fs -put /data/trendytech/boringwords.txt TT

```
[itv005357@g01 ~]$ hadoop fs -mkdir TT
[itv005357@g01 ~]$ hadoop fs -put /data/trendytech/boringwords.txt TT
[itv005357@g01 ~]$ hadoop fs -ls TT
Found 1 items
-rw-r--r--   3 itv005357 supergroup      79180 2023-05-03 14:59 TT/boringwords.txt
[itv005357@g01 ~]$
```

Now you can write the spark code:

#load data to rdd

rdd1 = spark.sparkContext.textFile("/public/trendytech/reviews/trendytech-student-reviews.csv")

converting all the words to lowercase

rdd2 = rdd1.flatMap(lambda x:x.split(" ")).map(lambda x:x.lower())

rdd3 = rdd2.map(lambda x: (x,1)).reduceByKey(lambda x,y:x+y)

#Load the boringwords

boring_words=spark.sparkContext.textFile("/user/itv005357/TT/boringwords.txt")

#broadcasting the boring words to all worker nodes

broadcast_bw = spark.sparkContext.broadcast(boring_words.collect())

#Filtering the words which are not in broadcast_bw(boringwords).

rdd4 = rdd3.filter(lambda x : x[0] not in broadcast_bw.value)

rdd5 = rdd4.reduceByKey(lambda x,y : x+y).sortBy(lambda x: x[1], ascending=False)

rdd5.take(20)

```
rdd1 = spark.sparkContext.textFile("/public/trendytech/reviews/trendytech-student-reviews.csv")
```

```
rdd2 = rdd1.flatMap(lambda x:x.split(" ")).map(lambda x:x.lower())
```

```
rdd2.take(5)
```

```
['i', 'got', 'to', 'know', 'about']
```

```
rdd3 = rdd2.map(lambda x: (x,1)).reduceByKey(lambda x,y:x+y)
```

```
rdd3.take(5)
```

```
[('i', 215), ('got', 7), ('know', 11), ('this', 135), ('of', 182)]
```

```
boring_words=spark.sparkContext.textFile("/user/itv005357/TT/boringwords.txt")
```

```
boring_words.take(5)
```

```
['shouldnt', 'worrying', 'simplify', 'tidy', 'shouldnt']
```

```
broadcast_bw = spark.sparkContext.broadcast(boring_words.collect())
```

```
rdd4 = rdd3.filter(lambda x : x[0] not in broadcast_bw.value)
```

```
rdd4.take(3)
```

```
[('sumit', 109), ('knowledge.', 5), ('aspirants', 1)]
```

```
rdd5 = rdd4.reduceByKey(lambda x,y : x+y).sortBy(lambda x: x[1], ascending=False)
```

```
rdd5.take(20)
```

```
[('data', 201),
 ('sumit', 109),
 ('trendytech', 67),
 ('', 64),
 ('data.', 34),
 ('course.', 33),
 ("sir's", 23),
 ('trendy', 14),
 ('course,', 13),
 ("master's", 13),
 ('domain.', 12),
 ("trendytech's", 12),
 ('sir.', 11),
 ('program.', 9),
 ('field.', 9),
 ('concepts.', 9),
 ('hands-on', 8),
 ('fresher', 8),
 ('amazing.', 8),
 ('career.', 7)]
```