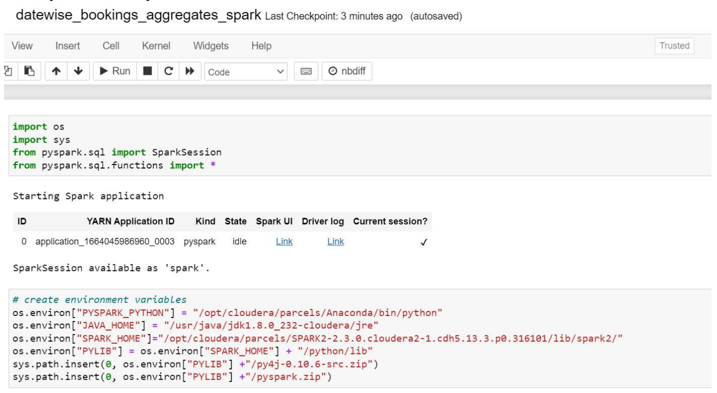
Load data from AWS RDS to Hadoop

<Command to run the python file>
Create a jupyter script and use pyspark kernel to execute it. Following screenshots show steps followed in the script to get date-wise aggregate data

Step1 – import modules and set environment variables


datewise_bookings_aggregates_spark Last Checkpoint: 3 minutes ago (autosaved)

```
import os
import sys
from pyspark.sql import SparkSession
from pyspark.sql.functions import *
```

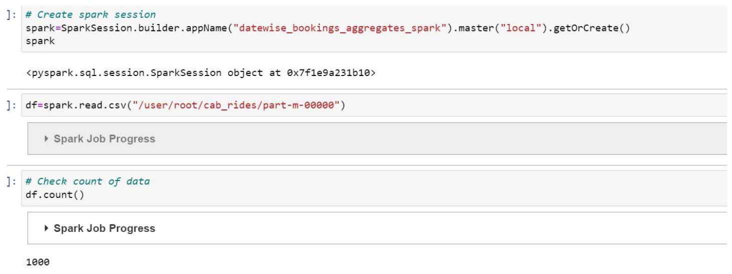
```
Starting Spark application
```

ID	YARN Application ID	Kind	State	Spark UI	Driver log	Current session?
0	application_1664045986960_0003	pyspark	idle	Link	Link	✓

```
SparkSession available as 'spark'.
```

```
# create environment variables
os.environ["PYSPARK_PYTHON"] = "/opt/cloudera/parcels/Anaconda/bin/python"
os.environ["JAVA_HOME"] = "/usr/java/jdk1.8.0_232-cloudera/jre"
os.environ["SPARK_HOME"]="/opt/cloudera/parcels/SPARK2-2.3.0.cloudera2-1.cdh5.13.3.p0.316101/lib/spark2/"
os.environ["PYLIB"] = os.environ["SPARK_HOME"] + "/python/lib"
sys.path.insert(0, os.environ["PYLIB"] +"/py4j-0.10.6-src.zip")
sys.path.insert(0, os.environ["PYLIB"] +"/pyspark.zip")
```

Step 2 Create spark session and read raw cab rides data



```
# Create spark session
spark=SparkSession.builder.appName("datewise_bookings_aggregates_spark").master("local").getOrCreate()
spark
```

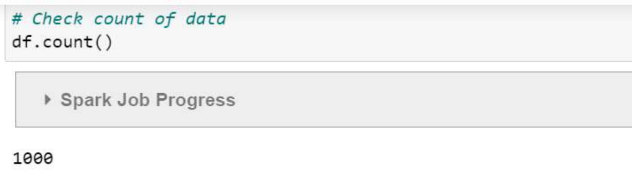
```
<pyspark.sql.session.SparkSession object at 0x7f1e9a231b10>
```

```
df=spark.read.csv("/user/root/cab_rides/part-m-00000")
```

```
# Check count of data
df.count()
```

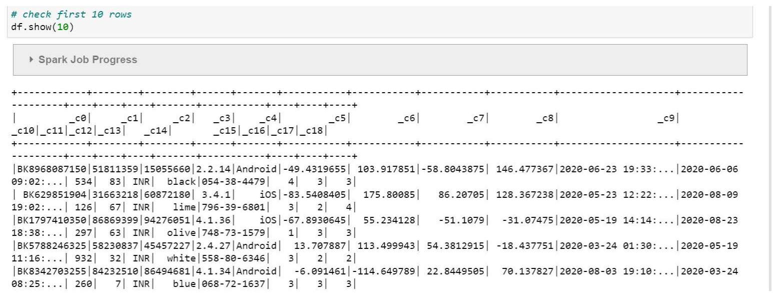
```
1000
```

Step 3 Check count of rows, schema and first 10 rows



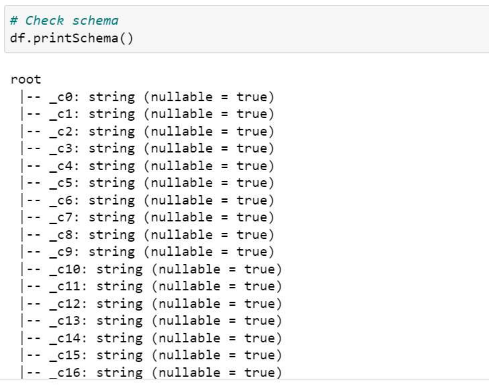
```
# Check count of data
df.count()
```

```
1000
```



```
# check first 10 rows
df.show(10)
```

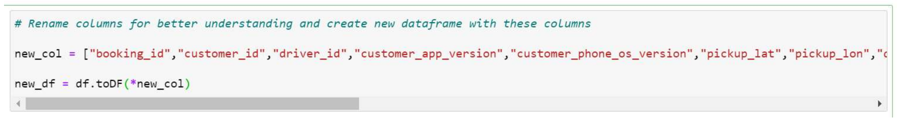
```
+------------+--------+--------+------+-------+-----------+-----------+-----------+-----------+--------------------+----------
---------+----+----+----+-------+-----------+----+----+----+
|          _c0|      _c1|      _c2|   _c3|     _c4|          _c5|          _c6|          _c7|          _c8|                   _c9|
_c10|_c11|_c12|_c13|    _c14|         _c15|_c16|_c17|_c18|
+------------+--------+--------+------+-------+-----------+-----------+-----------+-----------+--------------------+----------
---------+----+----+----+-------+-----------+----+----+----+
|BK8968087150|51811359|15055660|2.2.14|Android|-49.4319655| 103.917851|-58.8043875| 146.477367|2020-06-23 19:33:...|2020-06-06
09:02:...| 534|  83| INR|  black|054-38-4479|   4|   3|   3|
| BK629851904|31663218|60872180| 3.4.1|    iOS|-83.5408405|  175.80085|   86.20705| 128.367238|2020-05-23 12:22:...|2020-08-09
19:02:...| 126|  67| INR|   lime|796-39-6801|   3|   2|   4|
|BK1797410350|86869399|94276051|4.1.36|    iOS|-67.8930645|  55.234128|   -51.1079|  -31.07475|2020-05-19 14:14:...|2020-08-23
18:38:...| 297|  63| INR|  olive|748-73-1579|   1|   3|   3|
|BK5788246325|58230837|45457227|2.4.27|Android|  13.707887| 113.499943| 54.3812915| -18.437751|2020-03-24 01:30:...|2020-05-19
11:16:...| 932|  32| INR|  white|558-80-6346|   3|   2|   2|
|BK8342703255|84232510|86494681|4.1.34|Android|  -6.091461|-114.649789| 22.8449505|  70.137827|2020-08-03 19:10:...|2020-03-24
08:25:...| 260|   7| INR|   blue|068-72-1637|   3|   3|   3|
```



```
# Check schema
df.printSchema()
```

```
root
 |-- _c0: string (nullable = true)
 |-- _c1: string (nullable = true)
 |-- _c2: string (nullable = true)
 |-- _c3: string (nullable = true)
 |-- _c4: string (nullable = true)
 |-- _c5: string (nullable = true)
 |-- _c6: string (nullable = true)
 |-- _c7: string (nullable = true)
 |-- _c8: string (nullable = true)
 |-- _c9: string (nullable = true)
 |-- _c10: string (nullable = true)
 |-- _c11: string (nullable = true)
 |-- _c12: string (nullable = true)
 |-- _c13: string (nullable = true)
 |-- _c14: string (nullable = true)
 |-- _c15: string (nullable = true)
 |-- _c16: string (nullable = true)
```

Step 4 rename columns and create new data frame

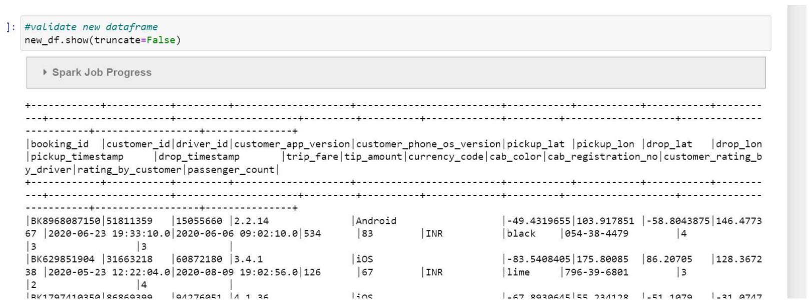


```
# Rename columns for better understanding and create new dataframe with these columns

new_col = ["booking_id","customer_id","driver_id","customer_app_version","customer_phone_os_version","pickup_lat","pickup_lon","c

new_df = df.toDF(*new_col)
```

Step 5 check new dataframe value

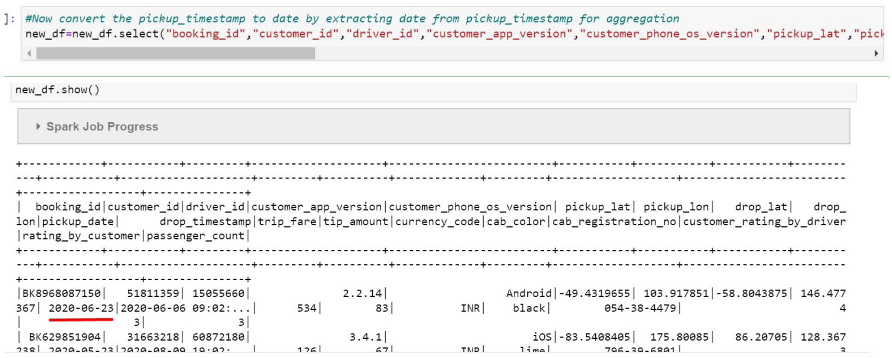


```
#validate new dataframe
new_df.show(truncate=False)
```

Spark Job Progress

```
+------------+-----------+---------+--------------------+-------------------------+-----------+-----------+-----------+--------
---+--------------------+--------------------+---------+----------+-------------+---------+-------------------+--------------
----------+------------------+---------------+
|booking_id  |customer_id|driver_id|customer_app_version|customer_phone_os_version|pickup_lat |pickup_lon |drop_lat   |drop_lon
|pickup_timestamp      |drop_timestamp        |trip_fare|tip_amount|currency_code|cab_color|cab_registration_no|customer_rating_b
y_driver|rating_by_customer|passenger_count|
+------------+-----------+---------+--------------------+-------------------------+-----------+-----------+-----------+--------
---+--------------------+--------------------+---------+----------+-------------+---------+-------------------+--------------
----------+------------------+---------------+
|BK8968087150|51811359   |15055660 |2.2.14              |Android                  |-49.4319655|103.917851 |-58.8043875|146.4773
67 |2020-06-23 19:33:10.0|2020-06-06 09:02:10.0|534      |83        |INR          |black    |054-38-4479        |4
|3                 |3              |
|BK629851904 |31663218   |60872180 |3.4.1               |iOS                      |-83.5408405|175.80085  |86.20705   |128.3672
38 |2020-05-23 12:22:04.0|2020-08-09 19:02:56.0|126      |67        |INR          |lime     |796-39-6801        |3
|2                 |4              |
```

Step 6 get only the dates from the pickup_date field which have timestamps as well.



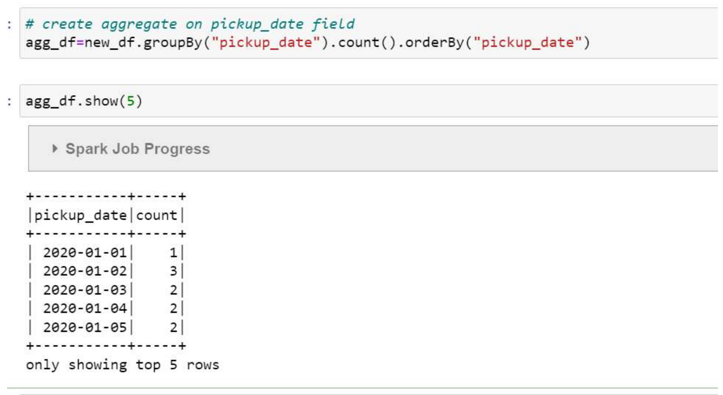
```
#Now convert the pickup_timestamp to date by extracting date from pickup_timestamp for aggregation
new_df=new_df.select("booking_id","customer_id","driver_id","customer_app_version","customer_phone_os_version","pickup_lat","pick
```

```
new_df.show()
```

Spark Job Progress

```
+------------+-----------+---------+--------------------+-------------------------+-----------+-----------+-----------+--------
---+-----------+--------------------+---------+----------+-------------+---------+-------------------+------------------------
+------------------+---------------+
|  booking_id|customer_id|driver_id|customer_app_version|customer_phone_os_version| pickup_lat| pickup_lon|   drop_lat|   drop_
lon|pickup_date|      drop_timestamp|trip_fare|tip_amount|currency_code|cab_color|cab_registration_no|customer_rating_by_driver
|rating_by_customer|passenger_count|
+------------+-----------+---------+--------------------+-------------------------+-----------+-----------+-----------+--------
---+-----------+--------------------+---------+----------+-------------+---------+-------------------+------------------------
+------------------+---------------+
|BK8968087150|   51811359| 15055660|              2.2.14|                  Android|-49.4319655| 103.917851|-58.8043875| 146.477
367| 2020-06-23|2020-06-06 09:02:...|      534|        83|          INR|    black|        054-38-4479|                       4
|                 3|              3|
| BK629851904|   31663218| 60872180|               3.4.1|                      iOS|-83.5408405|  175.80085|   86.20705| 128.367
```

Step7 create aggregate data frame by aggregating on pickup_date field



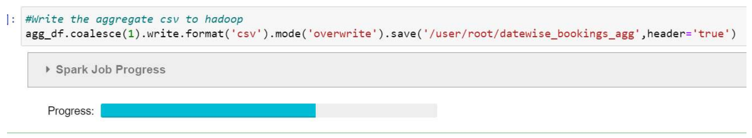
```
# create aggregate on pickup_date field
agg_df=new_df.groupBy("pickup_date").count().orderBy("pickup_date")
```

```
agg_df.show(5)
```

▸ Spark Job Progress

```
+-----------+-----+
|pickup_date|count|
+-----------+-----+
| 2020-01-01|    1|
| 2020-01-02|    3|
| 2020-01-03|    2|
| 2020-01-04|    2|
| 2020-01-05|    2|
+-----------+-----+
only showing top 5 rows
```

Step 8 Write this aggregate dataframe as CSV to hadoop



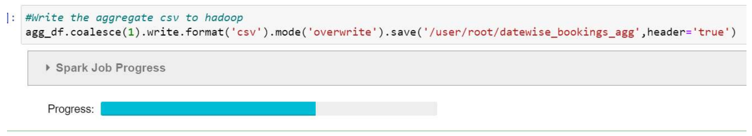
```
#Write the aggregate csv to hadoop
agg_df.coalesce(1).write.format('csv').mode('overwrite').save('/user/root/datewise_bookings_agg',header='true')
```

▸ Spark Job Progress

Progress:

<Command to move the csv file to HDFS>

agg_df.coalesce(1).write.format('csv').mode('overwrite').save('/user/root/datewise_bookings_ag g',header='true')

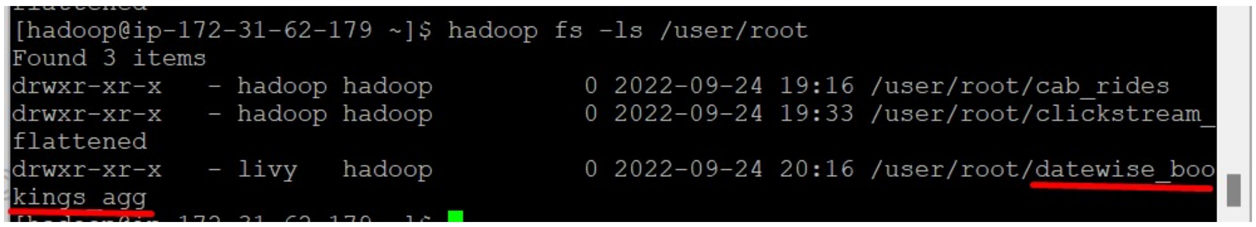


```
#Write the aggregate csv to hadoop
agg_df.coalesce(1).write.format('csv').mode('overwrite').save('/user/root/datewise_bookings_agg',header='true')
```

▸ Spark Job Progress

Progress:

<Screenshot of the file in HDFS>



```
[hadoop@ip-172-31-62-179 ~]$ hadoop fs -ls /user/root
Found 3 items
drwxr-xr-x   - hadoop hadoop          0 2022-09-24 19:16 /user/root/cab_rides
drwxr-xr-x   - hadoop hadoop          0 2022-09-24 19:33 /user/root/clickstream_
flattened
drwxr-xr-x   - livy   hadoop          0 2022-09-24 20:16 /user/root/datewise_boo
kings_agg
[hadoop@ip-172-31-62-179 ~]$
```