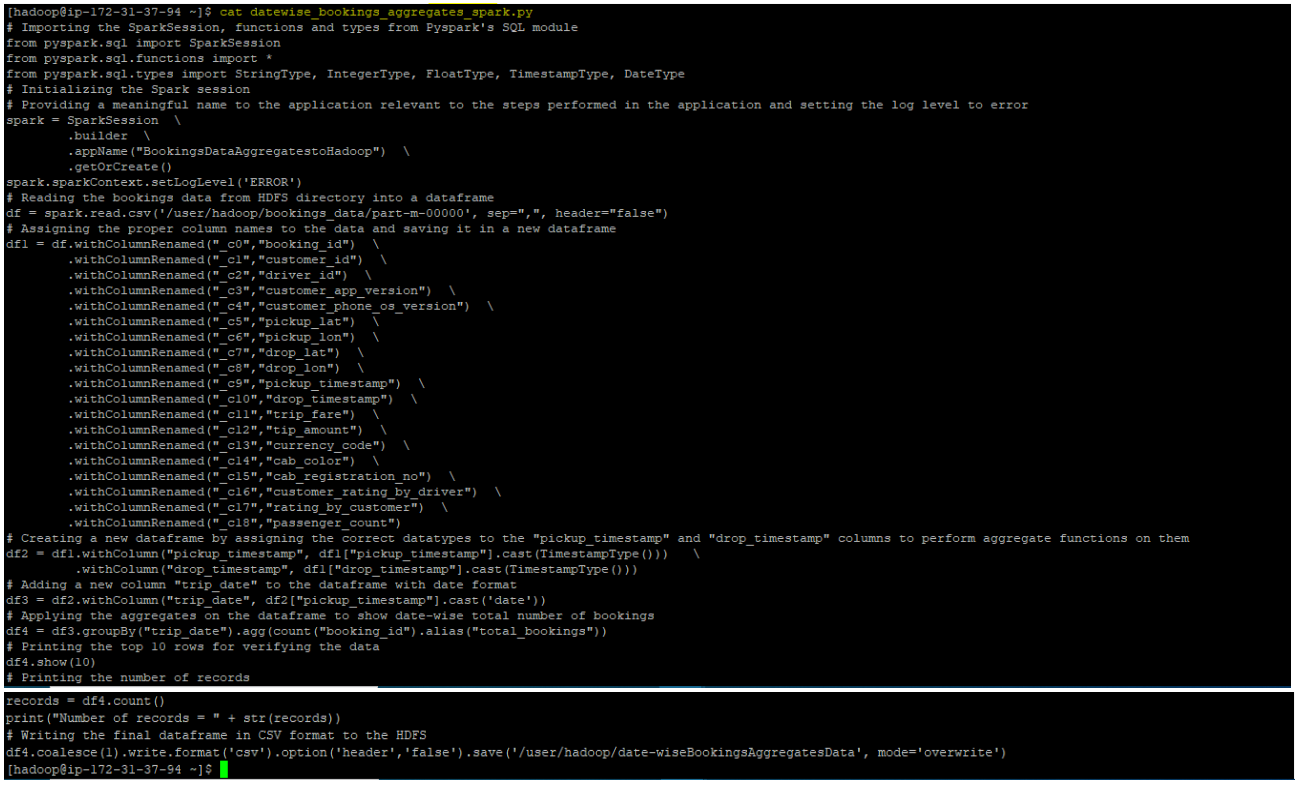
Load data from AWS RDS to Hadoop

Command to run the python file

- Below is the screenshot of the python file (**datewise_bookings_aggregates_spark.py**) containing the code to create aggregates for finding date-wise total bookings, store the data in CSV format and move the data to HDFS.



```
[hadoop@ip-172-31-37-94 ~]$ cat datewise_bookings_aggregates_spark.py
# Importing the SparkSession, functions and types from Pyspark's SQL module
from pyspark.sql import SparkSession
from pyspark.sql.functions import *
from pyspark.sql.types import StringType, IntegerType, FloatType, TimestampType, DateType
# Initializing the Spark session
# Providing a meaningful name to the application relevant to the steps performed in the application and setting the log level to error
spark = SparkSession  \
        .builder  \
        .appName("BookingsDataAggregatestoHadoop")  \
        .getOrCreate()
spark.sparkContext.setLogLevel('ERROR')
# Reading the bookings data from HDFS directory into a dataframe
df = spark.read.csv('/user/hadoop/bookings_data/part-m-00000', sep=",", header="false")
# Assigning the proper column names to the data and saving it in a new dataframe
df1 = df.withColumnRenamed("_c0","booking_id")  \
        .withColumnRenamed("_c1","customer_id")  \
        .withColumnRenamed("_c2","driver_id")  \
        .withColumnRenamed("_c3","customer_app_version")  \
        .withColumnRenamed("_c4","customer_phone_os_version")  \
        .withColumnRenamed("_c5","pickup_lat")  \
        .withColumnRenamed("_c6","pickup_lon")  \
        .withColumnRenamed("_c7","drop_lat")  \
        .withColumnRenamed("_c8","drop_lon")  \
        .withColumnRenamed("_c9","pickup_timestamp")  \
        .withColumnRenamed("_c10","drop_timestamp")  \
        .withColumnRenamed("_c11","trip_fare")  \
        .withColumnRenamed("_c12","tip_amount")  \
        .withColumnRenamed("_c13","currency_code")  \
        .withColumnRenamed("_c14","cab_color")  \
        .withColumnRenamed("_c15","cab_registration_no")  \
        .withColumnRenamed("_c16","customer_rating_by_driver")  \
        .withColumnRenamed("_c17","rating_by_customer")  \
        .withColumnRenamed("_c18","passenger_count")
# Creating a new dataframe by assigning the correct datatypes to the "pickup_timestamp" and "drop_timestamp" columns to perform aggregate functions on them
df2 = df1.withColumn("pickup_timestamp", df1["pickup_timestamp"].cast(TimestampType()))   \
         .withColumn("drop_timestamp", df1["drop_timestamp"].cast(TimestampType()))
# Adding a new column "trip_date" to the dataframe with date format
df3 = df2.withColumn("trip_date", df2["pickup_timestamp"].cast('date'))
# Applying the aggregates on the dataframe to show date-wise total number of bookings
df4 = df3.groupBy("trip_date").agg(count("booking_id").alias("total_bookings"))
# Printing the top 10 rows for verifying the data
df4.show(10)
# Printing the number of records
records = df4.count()
print("Number of records = " + str(records))
# Writing the final dataframe in CSV format to the HDFS
df4.coalesce(1).write.format('csv').option('header','false').save('/user/hadoop/date-wiseBookingsAggregatesData', mode='overwrite')
[hadoop@ip-172-31-37-94 ~]$
```

- Below is the spark-submit command to run **datewise_bookings_aggregates_spark.py file.**

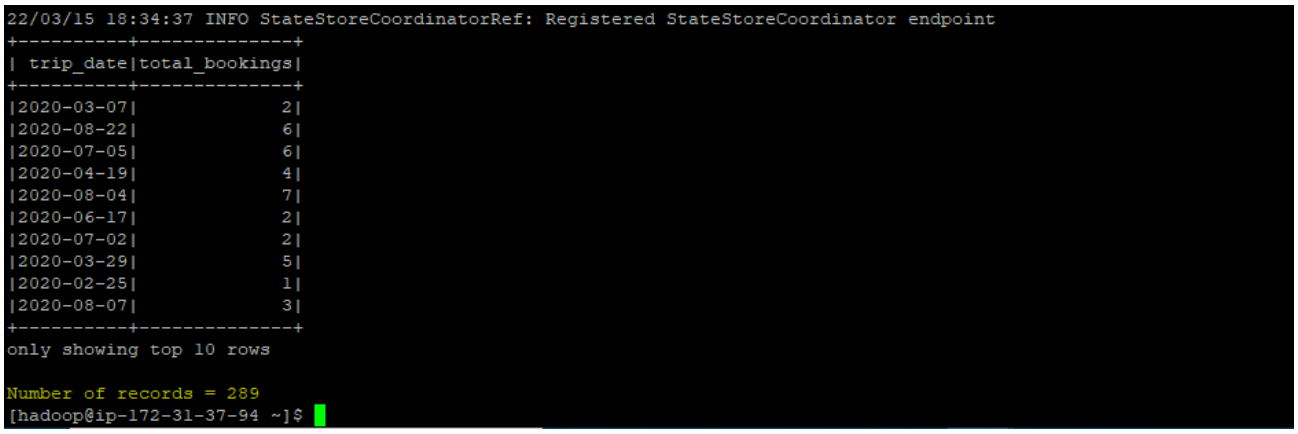
Command: spark-submit datewise_bookings_aggregates_spark.py

Command to move the CSV file to HDFS

- Below is the command used to move the CSV file to the HDFS. (Below line of code is included in the **datewise_bookings_aggregates_spark.py** file).

Command:
df4.coalesce(1).write.format('csv').option('header','false').save('/user/hadoop/date-wiseBookingsAggregatesData', mode='overwrite')

- Below is the screenshot of the output after running the python file (**datewise_bookings_aggregates_spark.py**).



Screenshot of the file in HDFS

- Below is the screenshot of the data in CSV format moved to HDFS.