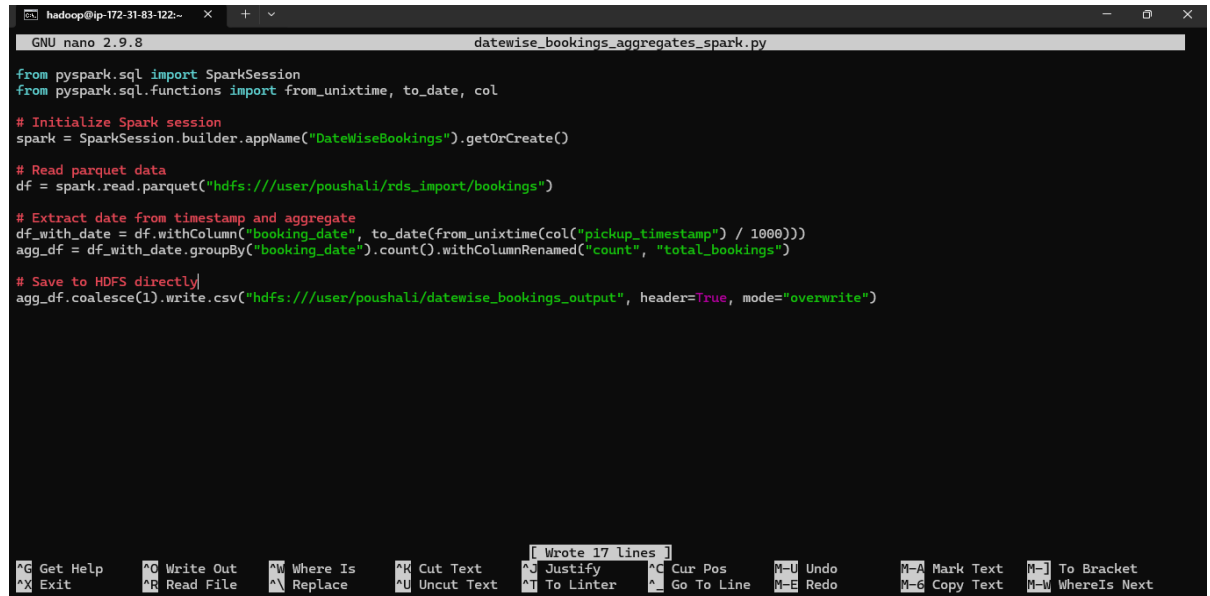


Hadoop Aggregated Booking Data

1. Running `datewise_bookings_aggregates_spark.py`



```

GNU nano 2.9.8                                datewise_bookings_aggregates_spark.py

from pyspark.sql import SparkSession
from pyspark.sql.functions import from_unixtime, to_date, col

# Initialize Spark session
spark = SparkSession.builder.appName("DateWiseBookings").getOrCreate()

# Read parquet data
df = spark.read.parquet("hdfs:///user/poushali/rds_import/bookings")

# Extract date from timestamp and aggregate
df_with_date = df.withColumn("booking_date", to_date(from_unixtime(col("pickup_timestamp") / 1000)))
agg_df = df_with_date.groupBy("booking_date").count().withColumnRenamed("count", "total_bookings")

# Save to HDFS directly
agg_df.coalesce(1).write.csv("hdfs:///user/poushali/datewise_bookings_output", header=True, mode="overwrite")
  
```

- To begin, the script creates a Spark session called "DateWiseBookings" and reads Parquet data from HDFS, which is accessible at `hdfs:///user/poushali/rds_import/bookings`.
- After that, it uses the `from_unixtime` and `to_date` methods to transform the `pickup_timestamp` field, which is in Unix time (milliseconds), into a date that can be read by humans.
- These changed dates are stored in a new column called `booking_date`. The `groupBy` and `count` methods are then used to aggregate the data and get the total number of bookings for each date.
- In the directory `hdfs:///user/poushali/datewise_bookings_output`, the generated DataFrame—which includes booking dates and the total number of bookings associated with them—is saved back to HDFS as a CSV file.
- The `header=True` option adds column headings to the CSV file, and the `coalesce(1)` function guarantees that the output is recorded to a single CSV file.
- Any existing files in the output directory will be replaced if the `mode="overwrite"` option is used.

2. Command to move the csv file to HDFS

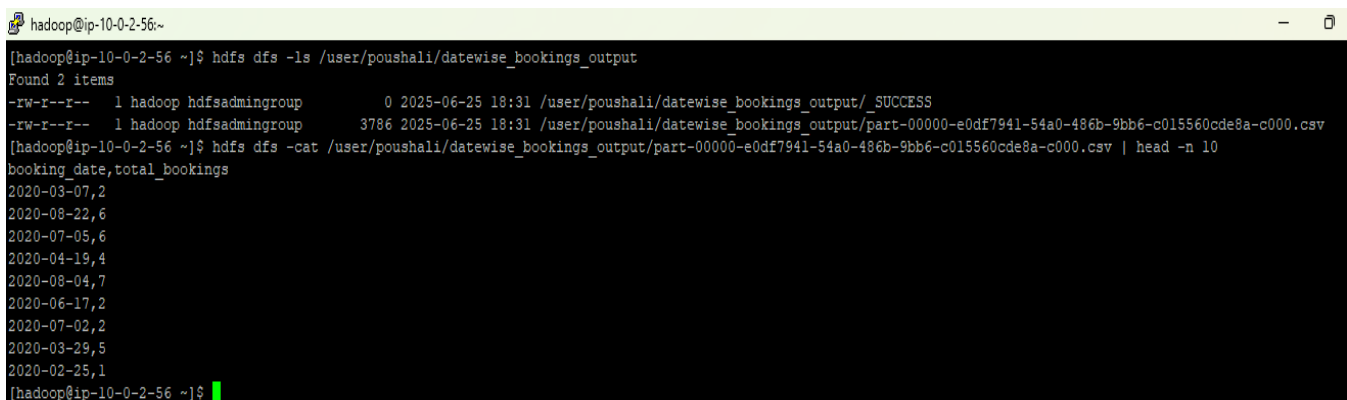
```
agg_df.coalesce(1).write.csv("hdfs:///user/poushali/datewise_bookings_output", header=True, mode="overwrite")
```

- i. `agg_df`: This is your starting point—a collection of data that already summarizes bookings by date.
- ii. `.coalesce(1)`: This is a crucial step for managing output. In a distributed system like Spark, data is often split into many parts (partitions). Without `coalesce(1)`, Spark would typically create a separate CSV file for each of these partitions. By using `.coalesce(1)`, you're telling Spark to combine all those partitions into a single one, ensuring that your output is just one consolidated CSV file rather than many smaller ones.
- iii. `.write.csv(...)`: This command initiates the process of saving your data in the popular CSV (Comma Separated Values) format.
- iv. `"hdfs:///user/poushali/datewise_bookings_output"`: This specifies the exact location where your CSV file will be stored. It's an HDFS path, meaning it's going onto a Hadoop Distributed File System.
- v. `header=True`: This option is for readability. It ensures that when you open your CSV file, the first row will clearly display the names of your columns, making it easy to understand what each column represents.
- vi. `mode="overwrite"`: This is an important safety and convenience feature. If there are already files in the specified HDFS location, this option tells Spark to delete them and replace them with your new output. If you didn't include this (or set it to "error"), Spark would stop and give you an error if it found existing files, preventing accidental overwrites.

3. Generated file in HDFS

Checking the generated file using command:

```
hdfs dfs -cat user/poushali/datewise_bookings_output/part-00000-e0df7941-54a0-486b-9bb6-c015560cde8a-c000.csv | head -n 5
```



```
hadoop@ip-10-0-2-56:~$ hdfs dfs -ls /user/poushali/datewise_bookings_output
Found 2 items
-rw-r--r-- 1 hadoop hdfsadmingroup 0 2025-06-25 18:31 /user/poushali/datewise_bookings_output/ SUCCESS
-rw-r--r-- 1 hadoop hdfsadmingroup 3786 2025-06-25 18:31 /user/poushali/datewise_bookings_output/part-00000-e0df7941-54a0-486b-9bb6-c015560cde8a-c000.csv
[hadoop@ip-10-0-2-56 ~]$ hdfs dfs -cat /user/poushali/datewise_bookings_output/part-00000-e0df7941-54a0-486b-9bb6-c015560cde8a-c000.csv | head -n 10
booking_date,total_bookings
2020-03-07,2
2020-08-22,6
2020-07-05,6
2020-04-19,4
2020-08-04,7
2020-06-17,2
2020-07-02,2
2020-03-29,5
2020-02-25,1
[hadoop@ip-10-0-2-56 ~]$
```