CS-236 Group Project Report, Phase II

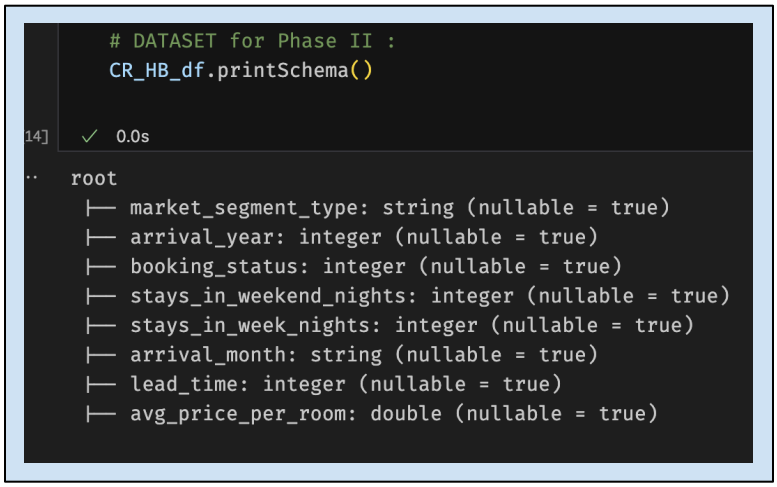
Group number : 15
Fall 2025

Introduction:

This report deals with the phase 2 of the class group project. We made a combined dataset **"CR_HB_df.csv"** by the end of phase 1, and the analysis is done in this report by using that combined database.
The report has following section:

(A) Spark Analysis on the unified dataframe.
(B) Populate the Database via PostgreSQL

Schema of the unified dataframe:



We have used numerics for denoting the calendar month :
{ 1: January , 2: February, 3: March,..., 11: November, 12: December}

A. Spark Analysis on the Unified Dataframe:

We used Pyspark to do the following analysis on the unified dataframe

1. Cancellation Rates for Each Month:

(a) We first created a sub-dataframe with columns **"arrival_month" and "booking_status"**. Next, we calculated the count of booking_status (0 and 1) grouped by the arrival month.

(b) Next we calculated the cancellation rate per month via creating a new column `"total_booking"` and then cancellation rate = sum("booking_status"==0) / total_booking, over a WindowSpec("arrival_month").

Observation:
The cancellation rate is highest in the month of July and the lowest in the month of December.

Table 1 : Cancellation Rate per Calendar Month

Calendar Month Number	Total booking per month	Total Cancellation per month (booking_status=0)	Cancellation Rate (%) Per month
1	3262	1715	52.58
2	5595	2984	53.33
3	7182	4047	56.35
4	8164	4362	53.43
5	8076	4511	55.86
6	8495	4487	52.82
7	10268	5904	**57.50**
8	12765	7017	54.97
9	15119	7930	52.45
10	16477	8794	53.37
11	9774	5547	56.75
12	9801	4811	**49.09**

2. Average Price and Average Number of Nights each month:

(a) We created a sub-dataframe with columns :
`"avg_price_per_room","stays_in_weekend_nights","stays_in_week_nights", "arrival_month"`

(b) We next created a new column :
`"night_stays"=count("stays_in_week_nights")+count("stays_in_weekend_nights")`, which gives the total number of nights of stays.

(c) At the end we used `.agg()` and `groupby("arrival_month")` to find the average stays and average price per room each month.

Observation:

1. Total number of nights stayed in the hotels = 114978 nights
2. Least number of night stayed are in January and Highest in October
3. The average price per room per night was minimum in January ($ 67.87/ night), whereas it was maximum in August ($ 122.76/ night).

Table 2: Average number of Nights Stayed and Average Price/Room for each month

Calendar Month Number	Average number of Nights_Stayed in the month	Average Room_Price per night in the calendar month ($)
1	8932	67.87
2	16097	73.34
3	23060	83.28
4	25454	93.3
5	25305	102.0
6	26820	108.86
7	38589	114.07
8	46541	122.76
9	49716	108.36
10	50154	93.25
11	30633	79.13
12	31876	83.5

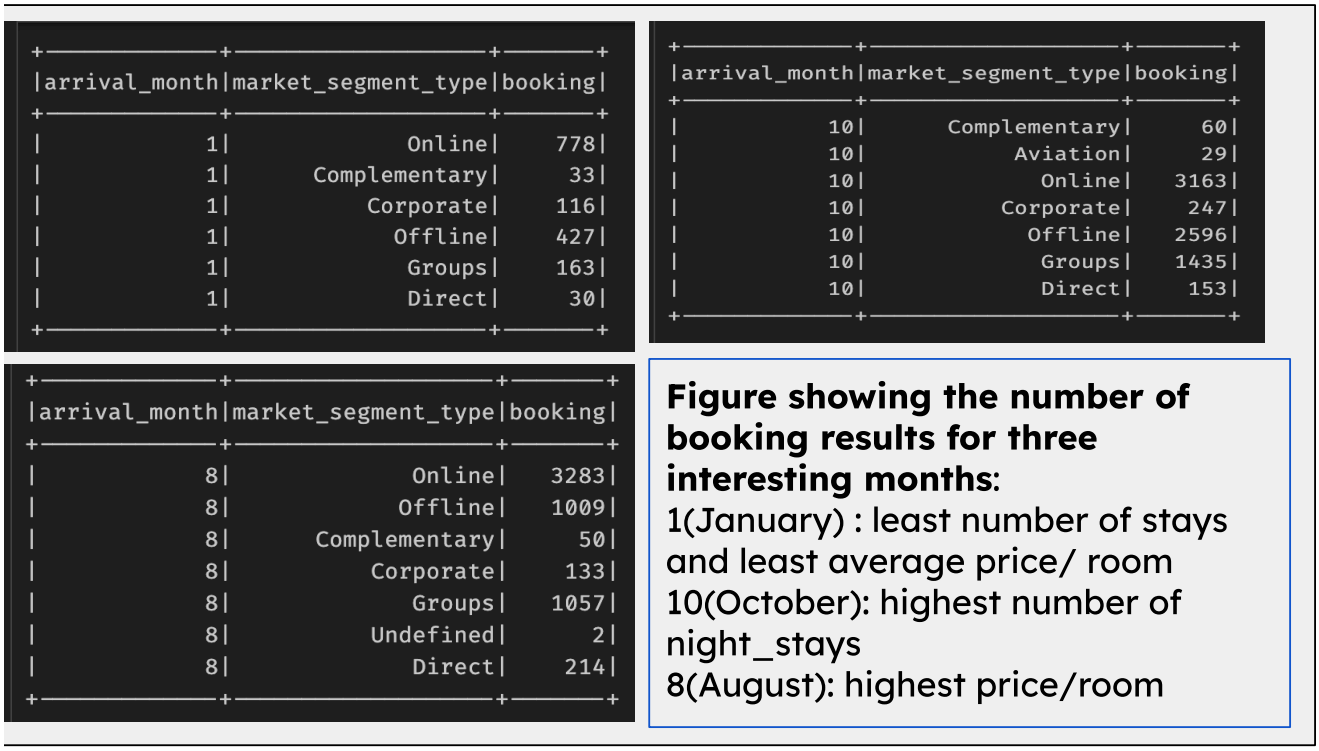
3. Monthly Bookings per Market Segment type:

(a) In the combined data-frame we had 8 market segment types as follows:
`{Complementary, Aviation ,Corporate, Online, Offline, Direct, Undefined, Groups}`

(b) Like the previous two cases we again created a new sub-data-frame with
`"arrival_month","booking_status","market_segment_type"`

(c) Then we calculated the number of bookings(booking_status=1) per market segment type grouped by arrival month

Figure 2: Monthly bookings per market segment type:



arrival_month	market_segment_type	booking
1	Online	778
1	Complementary	33
1	Corporate	116
1	Offline	427
1	Groups	163
1	Direct	30

arrival_month	market_segment_type	booking
10	Complementary	60
10	Aviation	29
10	Online	3163
10	Corporate	247
10	Offline	2596
10	Groups	1435
10	Direct	153

arrival_month	market_segment_type	booking
8	Online	3283
8	Offline	1009
8	Complementary	50
8	Corporate	133
8	Groups	1057
8	Undefined	2
8	Direct	214

Figure showing the number of booking results for three interesting months:
1(January) : least number of stays and least average price/ room
10(October): highest number of night_stays
8(August): highest price/room

Observation:

1. Except for month of August, none of the other months has the “undefined” market_segment_type
2. Most bookings are done “Online” (number=57040 overall) per month, whereas the next category is “Offline” booking.

4. Most popular month for booking with respect to Revenue:

(a) We used the formula for the Revenue = Avg_Price_Per_Room_Per_night **x** Total_night _stays
(b) We made a sub-data-frame with columns
{"avg_price_per_room","stays_in_weekend_nights","stays_in_week_nights", "arrival_month"}
(c) Next we added new columns “**night_stays**”, “**revenue_per_month**” grouped by calendar months.

Observation:
We observe that, based on the revenue per month 8:August was the most popular month/profitable month

Table 3: Total Revenue per month

Calendar Month	Revenue on the Calendar Month ($)
1	606214.84
2	1180553.98
3	1920436.8
4	2374858.2
5	2581110.0
6	2919625.2
7	4401847.23
8	5713373.16
9	5387225.76
10	4676860.5
11	2423989.29
12	2661646.0

B. Database Population using PostgreSQL:

---

1. Installing PostgreSQL with Docker:

(a) Installed the PostgreSQL in the docker
(b) Used the following codes in the terminal

```
> cd ~/Documents/code/python/pyspark
> docker ps
> docker exec -it pyspark_devcontainer-postgres-1 psql -U
postgres (enter)
> \l #to check all available databases
> CREATE DATABASE hotel_database;
> docker exec -it pyspark_devcontainer-postgres-1 psql -U
postgres -d hotel_database (# to access the database)
> \dt #to check all the tables
```

```
postgres=# CREATE DATABASE hotel_database;
CREATE DATABASE
postgres=# \c hotel_database
You are now connected to database "hotel_database" as user "postgres".
hotel_database=# \dt
                List of relations
 Schema |        Name         | Type  |  Owner
--------+---------------------+-------+----------
 public | connection_test     | table | postgres
 public | customer_resrvation | table | postgres
 public | hotel_booking       | table | postgres
(3 rows)

hotel_database=# SELECT COUNT(*) FROM hotel_booking;
 count
-------
 78703
(1 row)
```

OUTPUT 1:

Snippets of the Terminal Outputs For docker execution

```
hotel_database=# \q
sayantikanag@Sayantikas-MacBook-Pro ~/Documents/code/python/pyspark docker exec -it pyspark_devcontainer-postgres-1 psql -U postgres -d hotel_database

psql (14.19 (Debian 14.19-1.pgdg13+1))
Type "help" for help.

hotel_database=#
```

2. Creating database tables in PostgreSQL using Pyspark:

(a) Create a sparksession with PostgreSQL JDBC driver
(b) Created suitable JDBC url and Connection Props
(c) Uploaded the **"hotel_booking_new.csv"** and **"customer_reservation_new.csv"** into spark
(d) Wrote them into Postgresql with **.write.jdbc()**

Code Snippet 1:

```
from pyspark.sql import SparkSession

#1.Create Spark session with PostgreSQL JDBC driver
spark = SparkSession.builder \
    .appName("PostgresIntegration") \
    .config("spark.jars.packages", "org.postgresql:postgresql:42.6.0") \
    .getOrCreate()
#2. PostgreSQL connection info
jdbc_url = "jdbc:postgresql://pyspark_devcontainer-postgres-1:5432/hotel_database"
connection_props = {
    "user": "postgres",
    "password": "changethis",
    "driver": "org.postgresql.Driver"
}
```

[20] ✓ 0.0s

hotel_booking_new.csv== HB.csv

customer_reservation_new.csv==CR.csv

```
# 3.b Load CSVs into Spark
hotel_df = spark.read.csv("/workspace/CS236_Project_Fall2025_Datasets/HB.csv", header=True, inferSchema=True)
customer_df = spark.read.csv("/workspace/CS236_Project_Fall2025_Datasets/CR.csv", header=True, inferSchema=True)
cr_hb_df= spark.read.csv("/workspace/CS236_Project_Fall2025_Datasets/CR_HB.csv",header=True, inferSchema=True)
```

[28] ✓ 0.4s

Code snippet 2:

```
# 4. Write them into PostgreSQL
hotel_df.write.jdbc(url=jdbc_url, table="hotel_booking", mode="overwrite", properties=connection_props)
customer_df.write.jdbc(url=jdbc_url, table="customer_reservation", mode="overwrite", properties=connection_props)
cr_hb_df.write.jdbc(url=jdbc_url, table="cr_hb", mode="overwrite", properties=connection_props)

print("Data successfully loaded into PostgreSQL!")
```

✓ 0.8s

```
Data successfully loaded into PostgreSQL!
```

Check that the Schemas match:

```
hotel_df.printSchema()
```

✓ 0.0s

```
root
 ├── hotel: string (nullable = true)
 ├── booking_status: integer (nullable = true)
 ├── lead_time: integer (nullable = true)
 ├── arrival_year: integer (nullable = true)
 ├── arrival_month: integer (nullable = true)
 ├── arrival_date_week_number: integer (nullable = true)
 ├── arrival_date_day_of_month: integer (nullable = true)
 ├── stays_in_weekend_nights: integer (nullable = true)
 ├── stays_in_week_nights: integer (nullable = true)
 ├── market_segment_type: string (nullable = true)
 ├── country: string (nullable = true)
 ├── avg_price_per_room: double (nullable = true)
 ├── email: string (nullable = true)
```

```
hotel_booking_new.columns
```

✓ 0.0s

```
['hotel',
 'booking_status',
 'lead_time',
 'arrival_year',
 'arrival_month',
 'arrival_date_week_number',
 'arrival_date_day_of_month',
 'stays_in_weekend_nights',
 'stays_in_week_nights',
 'market_segment_type',
 'country',
 'avg_price_per_room',
 'email']
```

```
customer_df.printSchema()
```

✓ 0.0s

```
root
 ├── Booking_ID: string (nullable = true)
 ├── stays_in_weekend_nights: integer (nullable = true)
 ├── stays_in_week_nights: integer (nullable = true)
 ├── lead_time: integer (nullable = true)
 ├── arrival_year: integer (nullable = true)
 ├── arrival_month: integer (nullable = true)
 ├── arrival_date: integer (nullable = true)
 ├── market_segment_type: string (nullable = true)
 ├── avg_price_per_room: double (nullable = true)
 ├── booking_status: integer (nullable = true)
```

```
customer_reservation_new.columns
```

✓ 0.0s

```
['Booking_ID',
 'stays_in_weekend_nights',
 'stays_in_week_nights',
 'lead_time',
 'arrival_year',
 'arrival_month',
 'arrival_date',
 'market_segment_type',
 'avg_price_per_room',
 'booking_status']
```

```
cr_hb_df.printSchema()
[72] ✓ 0.0s
root
 ├── arrival_year: integer (nullable = true)
 ├── avg_price_per_room: double (nullable = true)
 ├── arrival_month: integer (nullable = true)
 ├── lead_time: integer (nullable = true)
 ├── booking_status: integer (nullable = true)
 ├── stays_in_weekend_nights: integer (nullable = true)
 ├── stays_in_week_nights: integer (nullable = true)
 ├── market_segment_type: string (nullable = true)
```

```
CR_HB_df.columns
✓ 0.0s
['avg_price_per_room',
 'arrival_month',
 'market_segment_type',
 'arrival_year',
 'lead_time',
 'stays_in_week_nights',
 'booking_status',
 'stays_in_weekend_nights']
```

3. Some more SQL queries in PostgreSQL:

Schema of hotel_booking in PostgreSQL:

```
hotel_database=# \d hotel_booking
                    Table "public.hotel_booking"
          Column           |       Type       | Collation | Nullable | Default
---------------------------+------------------+-----------+----------+---------
 hotel                     | text             |           |          |
 booking_status            | integer          |           |          |
 lead_time                 | integer          |           |          |
 arrival_year              | integer          |           |          |
 arrival_month             | integer          |           |          |
 arrival_date_week_number  | integer          |           |          |
 arrival_date_day_of_month | integer          |           |          |
 stays_in_weekend_nights   | integer          |           |          |
 stays_in_week_nights      | integer          |           |          |
 market_segment_type       | text             |           |          |
 country                   | text             |           |          |
 avg_price_per_room        | double precision |           |          |
 email                     | text             |           |          |
```

It is consistent with the previous schemas. So were the schemas of the other two dataframes.

We cross checked that the common columns are in fact the columns of the combined dataframe in PostgreSQL. Following are the code snippets:

```
hotel_database=# SELECT column_name
FROM information_schema.columns
WHERE table_name = 'hotel_booking'
INTERSECT
SELECT column_name
FROM information_schema.columns
WHERE table_name = 'customer_resrvation'
INTERSECT
SELECT column_name
FROM information_schema.columns
WHERE table_name = 'cr_hb';
       column_name
-------------------------
 booking_status
 arrival_month
 stays_in_week_nights
 avg_price_per_room
 lead_time
 arrival_year
 market_segment_type
 stays_in_weekend_nights
(8 rows)
```

```
hotel_database=# SELECT column_name
FROM information_schema.columns
WHERE table_name = 'hotel_booking'
INTERSECT
SELECT column_name
FROM information_schema.columns
WHERE table_name = 'customer_resrvation'
;
       column_name
-------------------------
 arrival_month
 booking_status
 stays_in_week_nights
 avg_price_per_room
 lead_time
 arrival_year
 market_segment_type
 stays_in_weekend_nights
(8 rows)
```

Project Contribution :

Phase 1:

Bufan: did the initial coding in .py file, did and wrote the “Installation of Pyspark : Bufan” in the report, generated new datasets in .xlsx format.

Sayantika: made the .ipynb notebook, wrote the report, generated new datasets in .csv format.

Phase 2:

Bufan: Made the video, did the .py file and checked Sayantika’s codes.

Sayantika: did the initial coding(Spark Analysis) and docker setup, made the .ipynb notebook, wrote the report.