Analyzing Hotel Booking Cancellations with HDFS, Hive, and Spark

J. Ryan Weaver

Bellevue University

DSC-650: Big Data

Nasheb Ismaily

In this comprehensive report, we journeyed through the realms of Hadoop Distributed File System (HDFS), Apache Hive, and Apache Spark to dissect, analyze, and predict hotel booking cancellations using the Hotel Bookings dataset. Each component is pivotal in handling big data, enabling efficient querying, and empowering machine learning capabilities.

1. Data Loading:

Our journey commences with acquiring the Hotel Bookings dataset from GitHub, a task accomplished by utilizing `wget` to fetch the data and `hdfs dfs -put` to transfer it to HDFS. The orchestration of our Hadoop ecosystem is simplified with Docker, facilitated by `docker-compose up -d` and accessing the master node using `docker-compose exec master bash`.

2. Hive – A Data Warehousing Solution:

As we navigate through vast datasets, Hive emerges as our first ally. With its SQL-like interface, Hive facilitates the creation of structured data tables. We initiate our journey by crafting a table through the `CREATE TABLE` statement, dictating the schema of our dataset.

The decision to `CLUSTER BY` and `BUCKET` the table by country is strategic. Clustering boosts data locality, optimizing query performance. Binning data into eight buckets enhances parallelism and further accelerates query processing. Our choice of `ROW FORMAT DELIMITED` and `FIELDS TERMINATED BY ','` ensures seamless parsing of the CSV data, with `STORED AS TEXTFILE` aligning it conveniently for Hadoop's file handling. To ensure data quality, `tblproperties("skip.header.line.count"="1")` skips the header line during ingestion.

Subsequently, we import our data into the Hive table, employing `LOAD DATA INPATH` to seamlessly transfer the contents of 'hotel_bookings.csv' into our structured Hive table.

3. In-Depth Analysis with Hive:

Harnessing Hive's analytical capabilities, we delve into the intricacies of room allocation dynamics. By investigating the correlation between reserved and assigned room types, we unearth patterns that go beyond mere booking statistics. Filtering the results to consider only room types below the average reservation frequency provides a nuanced understanding of room allocation trends.

Taking a step further, we construct a view to calculate the average duration of stay based on matching reserved and assigned room types. This materialized view optimizes performance and provides a persistent snapshot, which is crucial for time-sensitive analyses.

4. Spark – Unleashing the Power of Distributed Computing:

The second leg of our journey introduces Spark, a robust distributed computing engine. As our dataset grows, Spark accelerates data processing, fosters iterative algorithms, and facilitates machine learning operations.

Creating a Spark session bridges the gap between Spark and Hive, enabling seamless integration. Our choice to enable Hive support ensures fluid data interchange between the two components.

5. Machine Learning with Spark:

Our analytical journey takes a machine learning detour as we harness the predictive capabilities of Spark. Transferring data from Hive to Spark for model training, we prepare the dataset using the Vector Assembler. This transformation aggregates features into a single vector, readying the data for machine learning algorithms.

A pivotal point is partitioning our data into training and testing sets. With an 80/20 split, we ensure adequate training while preserving a substantial dataset for robust model evaluation.

6. Logistic Regression and Decision Trees:

The logistic regression model steps onto the stage first. Trained on our prepared dataset, the Binary Classification Evaluator evaluates the model. Calculating the area under the ROC curve provides insights into the model's predictive accuracy. With a resulting score of 0.538, we understand the model's discrimination capabilities.

Next in line is the Decision Tree model, following a similar evaluation metric. The resultant area under the ROC curve, pegged at 0.486, paints a distinct picture. A comparative analysis between logistic regression and decision trees unfolds, shedding light on their relative strengths and weaknesses.

7. Conclusion:

Our journey through HDFS, Hive, and Spark encapsulates the entire data lifecycle, from acquisition and storage to analysis and prediction. The strategic use of each component highlights its unique capabilities, catering to specific needs within the data analytics spectrum.

In conclusion, the harmonious collaboration of HDFS, Hive, and Spark provides a robust infrastructure for handling big data analytics and machine learning. The intricate dance between structured storage, SQL-like querying, and distributed computing has unveiled patterns, trends, and predictive insights in the realm of hotel booking cancellations, empowering decision-makers with actionable intelligence.

Hotel Booking Dataset Explanation:

Summary: This data set contains booking information for a city hotel and a resort hotel, and includes information such as when the booking was made, length of stay, the number of adults, children, and/or babies, and the number of available parking spaces, among other things.

- **Hotel:** Hotel (H1 = Resort Hotel or H2 = City Hotel)
- **Is_canceled:** Value indicating if the booking was canceled (1) or not (0)
- **Lead_time:** Number of days that elapsed between the entering date of the booking into the PMS and the arrival date
- **Arrival_date_year:** Year of arrival date
- **Arrival_date_month:** Month of arrival date
- **arrival_date_week_number:** Week number of year for arrival date
- **arrival_date_day_of_month:** Day of arrival date
- **stays_in_weekend_nights:** Number of weekend nights (Saturday or Sunday) the guest stayed or booked to stay at the hotel

- **stays_in_week_nights:** Number of week nights (Monday to Friday) the guest stayed or booked to stay at the hotel
- **adults:** Number of adults
- **children:** Number of children
- **babies:** Number of babies
- **meal:** Type of meal booked. Categories are presented in standard hospitality meal packages: Undefined/SC – no meal package; BB – Bed & Breakfast; HB – Half board (breakfast and one other meal – usually dinner); FB – Full board (breakfast, lunch and dinner)
- **country:** Country of origin. Categories are represented in the ISO 3155–3:2013 format
- **market_segment:** Market segment designation. In categories, the term "TA" means "Travel Agents" and "TO" means "Tour Operators"
- **distribution_channel:** Booking distribution channel. The term "TA" means "Travel Agents" and "TO" means "Tour Operators"
- **is_repeated_guest:** Value indicating if the booking name was from a repeated guest (1) or not (0)
- **previous_cancellations:** Number of previous bookings that were cancelled by the customer prior to the current booking
- **previous_bookings_not_canceled:** Number of previous bookings not cancelled by the customer prior to the current booking

- **reserved_room_type:** Code of room type reserved. Code is presented instead of designation for anonymity reasons.
- **assigned_room_type:** Code for the type of room assigned to the booking. Sometimes the assigned room type differs from the reserved room type due to hotel operation reasons (e.g. overbooking) or by customer request. Code is presented instead of designation for anonymity reasons.
- **booking_changes:** Number of changes/amendments made to the booking from the moment the booking was entered on the PMS until the moment of check-in or cancellation
- **deposit_type:** Indication on if the customer made a deposit to guarantee the booking. This variable can assume three categories: No Deposit – no deposit was made; Non Refund – a deposit was made in the value of the total stay cost; Refundable – a deposit was made with a value under the total cost of stay.
- **Agent:** ID of the travel agency that made the booking
- **Company:** ID of the company/entity that made the booking or responsible for paying the booking. ID is presented instead of designation for anonymity reasons
- **days_in_waiting_list:** Number of days the booking was in the waiting list before it was confirmed to the customer
- **customer_type:** Type of booking, assuming one of four categories: Contract - when the booking has an allotment or other type of contract associated to it; Group – when the booking is associated to a group; Transient – when the booking is not part of a group or contract, and is not

associated to other transient booking; Transient-party – when the booking is transient, but is associated to at least other transient booking

- **adr:** Average Daily Rate as defined by dividing the sum of all lodging transactions by the total number of staying nights

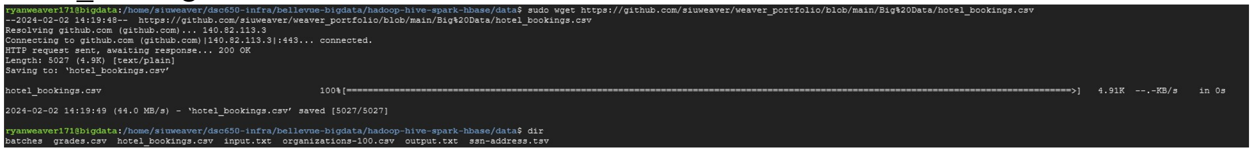
- **required_car_parking_spaces:** Number of car parking spaces required by the customer

- **total_of_special_requests:** Number of special requests made by the customer (e.g. twin bed or high floor)

- **reservation_status:** Reservation last status, assuming one of three categories: Canceled – booking was canceled by the customer; Check-Out – customer has checked in but already departed; No-Show – customer did not check-in and did inform the hotel of the reason why

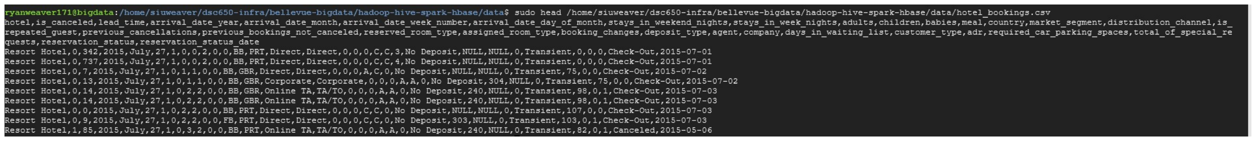
- **reservation_status_date:** Date at which the last status was set. This variable can be used in conjunction with the ReservationStatus to understand when was the booking canceled or when did the customer checked-out of the hotel

Walkthrough With Screenshots:

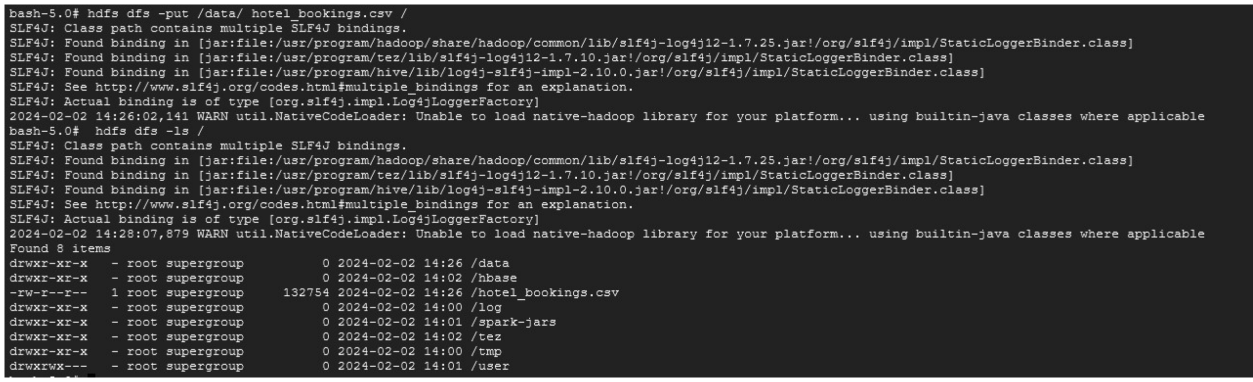
1. Load Data
    a. cd /home/siuweaver/dsc650-infra/bellevue-bigdata/hadoop-hive-spark-hbase/data
    b. sudo wget https://raw.githubusercontent.com/siuweaver/weaver_portfolio/main/Big%20Data/hotel_bookings.csv
    c. 

```
ryanweaver171@bigdata:/home/siuweaver/dsc650-infra/bellevue-bigdata/hadoop-hive-spark-hbase/data$ sudo wget https://github.com/siuweaver/weaver_portfolio/blob/main/Big%20Data/hotel_bookings.csv
--2024-02-02 14:19:48--  https://github.com/siuweaver/weaver_portfolio/blob/main/Big%20Data/hotel_bookings.csv
Resolving github.com (github.com)... 140.82.113.3
Connecting to github.com (github.com)|140.82.113.3|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 5027 (4.9K) [text/plain]
Saving to: 'hotel_bookings.csv'

hotel_bookings.csv                              100%[===================================================================================================================>]   4.91K  --.-KB/s    in 0s

2024-02-02 14:19:49 (44.0 MB/s) - 'hotel_bookings.csv' saved [5027/5027]

ryanweaver171@bigdata:/home/siuweaver/dsc650-infra/bellevue-bigdata/hadoop-hive-spark-hbase/data$ dir
batches  grades.csv  hotel_bookings.csv  input.txt  organizations-100.csv  output.txt  ssn-address.tsv
```

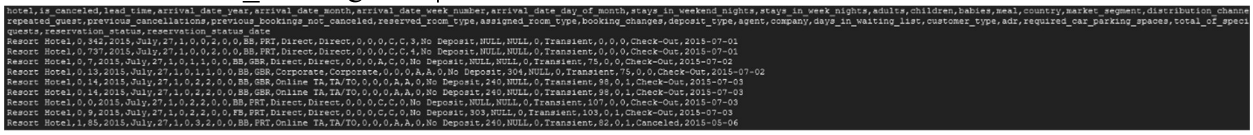
    d. sudo head /home/siuweaver/dsc650-infra/bellevue-bigdata/hadoop-hive-spark-hbase/data/hotel_bookings.csv

e. 

```
ryanweaver171@bigdata:/home/siuweaver/dsc650-infra/bellevue-bigdata/hadoop-hive-spark-hbase/data$ sudo head /home/siuweaver/dsc650-infra/bellevue-bigdata/hadoop-hive-spark-hbase/data/hotel_bookings.csv
hotel,is_canceled,lead_time,arrival_date_year,arrival_date_month,arrival_date_week_number,arrival_date_day_of_month,stays_in_weekend_nights,stays_in_week_nights,adults,children,babies,meal,country,market_segment,distribution_channel,is_
repeated_guest,previous_cancellations,previous_bookings_not_canceled,reserved_room_type,assigned_room_type,booking_changes,deposit_type,agent,company,days_in_waiting_list,customer_type,adr,required_car_parking_spaces,total_of_special_re
quests,reservation_status,reservation_status_date
Resort Hotel,0,342,2015,July,27,1,0,0,2,0,0,BB,PRT,Direct,Direct,0,0,0,C,C,3,No Deposit,NULL,NULL,0,Transient,0,0,0,Check-Out,2015-07-01
Resort Hotel,0,737,2015,July,27,1,0,0,2,0,0,BB,PRT,Direct,Direct,0,0,0,C,C,4,No Deposit,NULL,NULL,0,Transient,0,0,0,Check-Out,2015-07-01
Resort Hotel,0,7,2015,July,27,1,0,1,1,0,0,BB,GBR,Direct,Direct,0,0,0,A,C,0,No Deposit,NULL,NULL,0,Transient,75,0,0,Check-Out,2015-07-02
Resort Hotel,0,13,2015,July,27,1,0,1,1,0,0,BB,GBR,Corporate,Corporate,0,0,0,A,A,0,No Deposit,304,NULL,0,Transient,75,0,0,Check-Out,2015-07-02
Resort Hotel,0,14,2015,July,27,1,0,2,2,0,0,BB,GBR,Online TA,TA/TO,0,0,0,A,A,0,No Deposit,240,NULL,0,Transient,98,0,1,Check-Out,2015-07-03
Resort Hotel,0,14,2015,July,27,1,0,2,2,0,0,BB,GBR,Online TA,TA/TO,0,0,0,A,A,0,No Deposit,240,NULL,0,Transient,98,0,1,Check-Out,2015-07-03
Resort Hotel,0,0,2015,July,27,1,0,2,2,0,0,BB,PRT,Direct,Direct,0,0,0,C,C,0,No Deposit,NULL,NULL,0,Transient,107,0,0,Check-Out,2015-07-03
Resort Hotel,0,9,2015,July,27,1,0,2,2,0,0,FB,PRT,Direct,Direct,0,0,0,C,C,0,No Deposit,303,NULL,0,Transient,103,0,1,Check-Out,2015-07-03
Resort Hotel,1,85,2015,July,27,1,0,3,2,0,0,BB,PRT,Online TA,TA/TO,0,0,0,A,A,0,No Deposit,240,NULL,0,Transient,82,0,1,Canceled,2015-05-06
```

2. Environment Initialization
   a. cd ..
      i. Return to hadoop-hive-spark-hbase
   b. sudo docker-compose up -d
   c. sudo docker-compose exec master bash
3. Load
   a. hdfs dfs -put /data/hotel_bookings.csv /
4. Confirm data is loaded
   a. hdfs dfs -ls /
   b. 

```
bash-5.0# hdfs dfs -put /data/ hotel_bookings.csv /
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/program/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/program/tez/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/program/hive/lib/log4j-slf4j-impl-2.10.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
2024-02-02 14:26:02,141 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
bash-5.0#  hdfs dfs -ls /
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/program/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/program/tez/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/program/hive/lib/log4j-slf4j-impl-2.10.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
2024-02-02 14:28:07,879 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 8 items
drwxr-xr-x   - root supergroup          0 2024-02-02 14:26 /data
drwxr-xr-x   - root supergroup          0 2024-02-02 14:02 /hbase
-rw-r--r--   1 root supergroup     132754 2024-02-02 14:26 /hotel_bookings.csv
drwxr-xr-x   - root supergroup          0 2024-02-02 14:00 /log
drwxr-xr-x   - root supergroup          0 2024-02-02 14:01 /spark-jars
drwxr-xr-x   - root supergroup          0 2024-02-02 14:02 /tez
drwxr-xr-x   - root supergroup          0 2024-02-02 14:00 /tmp
drwxrwx---   - root supergroup          0 2024-02-02 14:01 /user
```

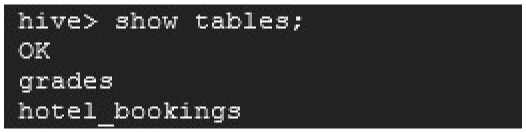
   c. hdfs dfs -cat /hotel_bookings.csv | head
   d. 

```
hotel,is_canceled,lead_time,arrival_date_year,arrival_date_month,arrival_date_week_number,arrival_date_day_of_month,stays_in_weekend_nights,stays_in_week_nights,adults,children,babies,meal,country,market_segment,distribution_channe
repeated_guest,previous_cancellations,previous_bookings_not_canceled,reserved_room_type,assigned_room_type,booking_changes,deposit_type,agent,company,days_in_waiting_list,customer_type,adr,required_car_parking_spaces,total_of_speci
quests,reservation_status,reservation_status_date
Resort Hotel,0,342,2015,July,27,1,0,0,2,0,0,BB,PRT,Direct,Direct,0,0,0,C,C,3,No Deposit,NULL,NULL,0,Transient,0,0,0,Check-Out,2015-07-01
Resort Hotel,0,737,2015,July,27,1,0,0,2,0,0,BB,PRT,Direct,Direct,0,0,0,C,C,4,No Deposit,NULL,NULL,0,Transient,0,0,0,Check-Out,2015-07-01
Resort Hotel,0,7,2015,July,27,1,0,1,1,0,0,BB,GBR,Direct,Direct,0,0,0,A,C,0,No Deposit,NULL,NULL,0,Transient,75,0,0,Check-Out,2015-07-02
Resort Hotel,0,13,2015,July,27,1,0,1,1,0,0,BB,GBR,Corporate,Corporate,0,0,0,A,A,0,No Deposit,304,NULL,0,Transient,75,0,0,Check-Out,2015-07-02
Resort Hotel,0,14,2015,July,27,1,0,2,2,0,0,BB,GBR,Online TA,TA/TO,0,0,0,A,A,0,No Deposit,240,NULL,0,Transient,98,0,1,Check-Out,2015-07-03
Resort Hotel,0,14,2015,July,27,1,0,2,2,0,0,BB,GBR,Online TA,TA/TO,0,0,0,A,A,0,No Deposit,240,NULL,0,Transient,98,0,1,Check-Out,2015-07-03
Resort Hotel,0,0,2015,July,27,1,0,2,2,0,0,BB,PRT,Direct,Direct,0,0,0,C,C,0,No Deposit,NULL,NULL,0,Transient,107,0,0,Check-Out,2015-07-03
Resort Hotel,0,9,2015,July,27,1,0,2,2,0,0,FB,PRT,Direct,Direct,0,0,0,C,C,0,No Deposit,303,NULL,0,Transient,103,0,1,Check-Out,2015-07-03
Resort Hotel,1,85,2015,July,27,1,0,3,2,0,0,BB,PRT,Online TA,TA/TO,0,0,0,A,A,0,No Deposit,240,NULL,0,Transient,82,0,1,Canceled,2015-05-06
```

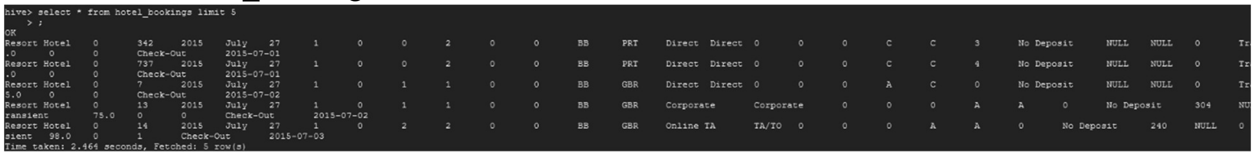
5. Start Hive
   a. hive
6. Create a Hive Table

```
CREATE TABLE hotel_bookings (

  hotel STRING,

  is_canceled INT,

  lead_time INT,

  arrival_date_year INT,

  arrival_date_month STRING,

  arrival_date_week_number INT,

  arrival_date_day_of_month INT,

  stays_in_weekend_nights INT,
  stays_in_week_nights INT,
  adults INT,
  children INT,
  babies INT,
  meal STRING,
  country STRING,
  market_segment STRING,
  distribution_channel STRING,
  is_repeated_guest INT,
  previous_cancellations INT,
  previous_bookings_not_canceled INT,
  reserved_room_type STRING,
  assigned_room_type STRING,
  booking_changes INT,
  deposit_type STRING,
  agent INT,
  company INT,
  days_in_waiting_list INT,
  customer_type STRING,
  adr DOUBLE,
  required_car_parking_spaces INT,
  total_of_special_requests INT,
  reservation_status STRING,
  reservation_status_date STRING
)
CLUSTERED BY (country) INTO 8 BUCKETS
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
```

```
STORED AS TEXTFILE

tblproperties("skip.header.line.count"="1");
```

7. Confirm table
    a. show tables;
    b. 

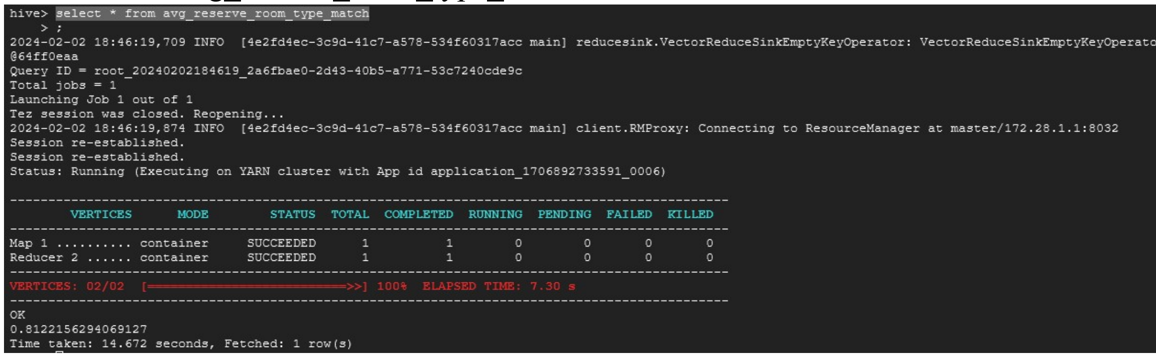
```
hive> show tables;
OK
grades
hotel_bookings
```

8. Load Data into table
    a. LOAD DATA INPATH '/hotel_bookings.csv' INTO TABLE hotel_bookings;
9. Confirm Loading
    a. select * from hotel_bookings limit 5;
    b. 

```
hive> select * from hotel_bookings limit 5
    > ;
OK
Resort Hotel    0       342     2015    July    27      1       0       0       2       0       0       BB      PRT     Direct  Direct  0       0       0       C       C       3       No Deposit      NULL    NULL    0       Tr
.0      0       0       Check-Out       2015-07-01
Resort Hotel    0       737     2015    July    27      1       0       0       2       0       0       BB      PRT     Direct  Direct  0       0       0       C       C       4       No Deposit      NULL    NULL    0       Tr
.0      0       0       Check-Out       2015-07-01
Resort Hotel    0       7       2015    July    27      1       0       1       1       0       0       BB      GBR     Direct  Direct  0       0       0       A       C       0       No Deposit      NULL    NULL    0       Tr
5.0     0       0       Check-Out       2015-07-02
Resort Hotel    0       13      2015    July    27      1       0       1       1       0       0       BB      GBR     Corporate       Corporate       0       0       0       A       A       0       No Deposit      304     NU
ransient        75.0    0       0       Check-Out       2015-07-02
Resort Hotel    0       14      2015    July    27      1       0       2       2       0       0       BB      GBR     Online TA       TA/TO   0       0       0       A       A       0       No Deposit      240     NULL    0
sient   98.0    0       1       Check-Out       2015-07-03
Time taken: 2.464 seconds, Fetched: 5 row(s)
```

10. Create View for Average Room Type Match
    a. View stores the percent of the time the reserved room type matches the assigned room type for all reservations that weren't cancelled. This will allow us to compare the overall average to the value for each individual room type

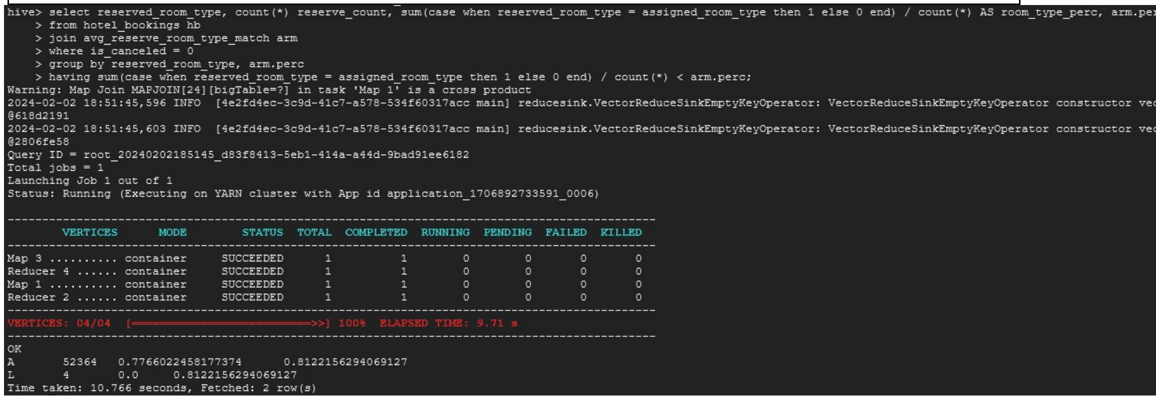
```
CREATE VIEW IF NOT EXISTS avg_reserve_room_type_match AS

SELECT sum(case when reserved_room_type = assigned_room_type then 1 else 0 end) / count(
AS perc

FROM hotel_bookings

WHERE is_canceled = 0;
```

    b. Validate view
        i. select * from avg_reserve_room_type_match;
        ii. 

```
hive> select * from avg_reserve_room_type_match
    > ;
2024-02-02 18:46:19,709 INFO  [4e2fd4ec-3c9d-41c7-a578-534f60317acc main] reducesink.VectorReduceSinkEmptyKeyOperator: VectorReduceSinkEmptyKeyOperato
@64ff0eaa
Query ID = root_20240202184619_2a6fbae0-2d43-40b5-a771-53c7240cde9c
Total jobs = 1
Launching Job 1 out of 1
Tez session was closed. Reopening...
2024-02-02 18:46:19,874 INFO  [4e2fd4ec-3c9d-41c7-a578-534f60317acc main] client.RMProxy: Connecting to ResourceManager at master/172.28.1.1:8032
Session re-established.
Session re-established.
Status: Running (Executing on YARN cluster with App id application_1706892733591_0006)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      1          1        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      1          1        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 7.30 s
----------------------------------------------------------------------------------------------
OK
0.8122156294069127
Time taken: 14.672 seconds, Fetched: 1 row(s)
```

    c. Return reserved room types below average

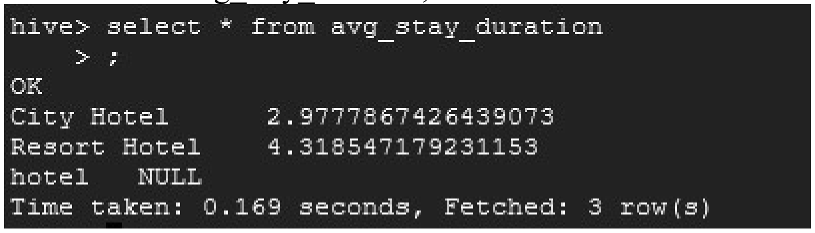
```
select reserved_room_type, count(*) reserve_count, sum(case when reserved_room_type = assigned_room_type then 1 else 0 end) / count(*) AS room_type_perc, arm.perc

from hotel_bookings hb

join avg_reserve_room_type_match arm

where is_canceled = 0

group by reserved_room_type, arm.perc

having sum(case when reserved_room_type = assigned_room_type then 1 else 0 end) / count(*) < arm.perc;
```

i. 

```
hive> select reserved_room_type, count(*) reserve_count, sum(case when reserved_room_type = assigned_room_type then 1 else 0 end) / count(*) AS room_type_perc, arm.per
    > from hotel_bookings hb
    > join avg_reserve_room_type_match arm
    > where is_canceled = 0
    > group by reserved_room_type, arm.perc
    > having sum(case when reserved_room_type = assigned_room_type then 1 else 0 end) / count(*) < arm.perc;
Warning: Map Join MAPJOIN[24][bigTable=?] in task 'Map 1' is a cross product
2024-02-02 18:51:45,596 INFO  [4e2fd4ec-3c9d-41c7-a578-534f60317acc main] reducesink.VectorReduceSinkEmptyKeyOperator: VectorReduceSinkEmptyKeyOperator constructor vec
@618d2191
2024-02-02 18:51:45,603 INFO  [4e2fd4ec-3c9d-41c7-a578-534f60317acc main] reducesink.VectorReduceSinkEmptyKeyOperator: VectorReduceSinkEmptyKeyOperator constructor vec
@2806fe58
Query ID = root_20240202185145_d83f8413-5eb1-414a-a44d-9bad91ee6182
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1706892733591_0006)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 3 .......... container     SUCCEEDED      1          1        0        0       0       0
Reducer 4 ...... container     SUCCEEDED      1          1        0        0       0       0
Map 1 .......... container     SUCCEEDED      1          1        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      1          1        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 04/04  [==========================>>] 100%  ELAPSED TIME: 9.71 s
----------------------------------------------------------------------------------------------
OK
A       52364   0.7766022458177374      0.8122156294069127
L       4       0.0     0.8122156294069127
Time taken: 10.766 seconds, Fetched: 2 row(s)
```

d. Create Materialized View

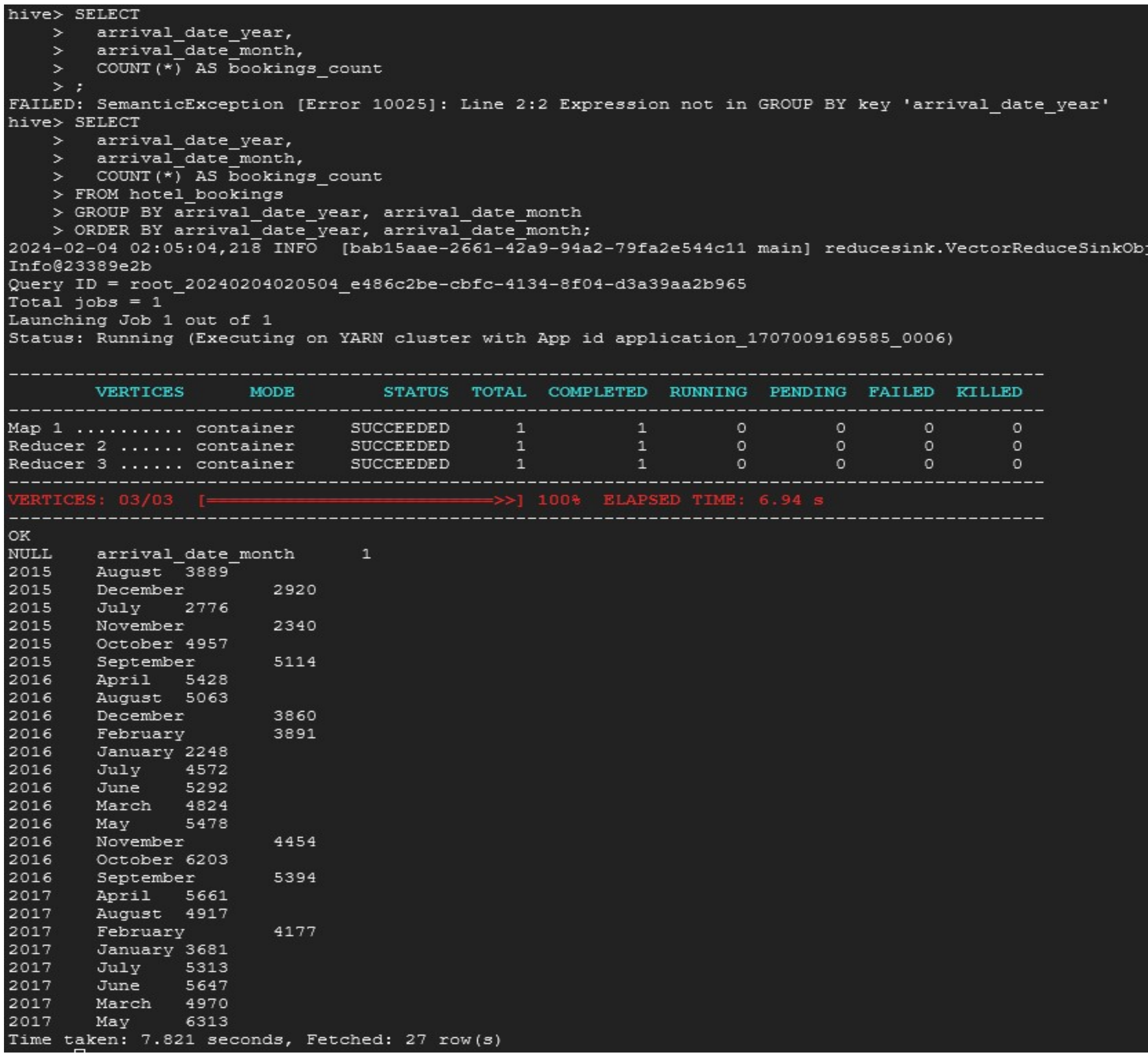
```
CREATE MATERIALIZED VIEW avg_stay_duration

AS

SELECT hotel, AVG(stays_in_weekend_nights + stays_in_week_nights) AS avg_duration

FROM hotel_bookings

GROUP BY hotel;
```

1. Validate results
   a. Select * from avg_stay_duration;
   b. 

```
hive> select * from avg_stay_duration
    > ;
OK
City Hotel      2.9777867426439073
Resort Hotel    4.318547179231153
hotel   NULL
Time taken: 0.169 seconds, Fetched: 3 row(s)
```

ii. View Trending over Time

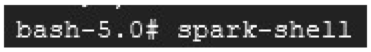
```
SELECT

  arrival_date_year,

  arrival_date_month,

  COUNT(*) AS bookings_count

FROM hotel_bookings

GROUP BY arrival_date_year, arrival_date_month

ORDER BY arrival_date_year, arrival_date_month;
```

iii. 

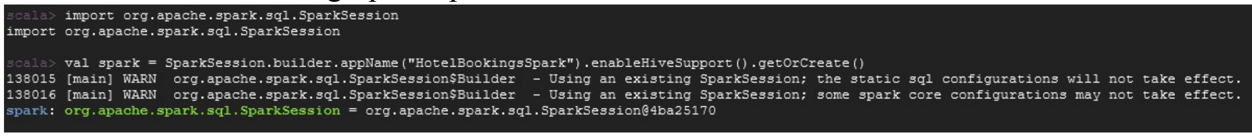
```
hive> SELECT
    >   arrival_date_year,
    >   arrival_date_month,
    >   COUNT(*) AS bookings_count
    > ;
FAILED: SemanticException [Error 10025]: Line 2:2 Expression not in GROUP BY key 'arrival_date_year'
hive> SELECT
    >   arrival_date_year,
    >   arrival_date_month,
    >   COUNT(*) AS bookings_count
    > FROM hotel_bookings
    > GROUP BY arrival_date_year, arrival_date_month
    > ORDER BY arrival_date_year, arrival_date_month;
2024-02-04 02:05:04,218 INFO  [bab15aae-2661-42a9-94a2-79fa2e544c11 main] reducesink.VectorReduceSinkOb
Info@23389e2b
Query ID = root_20240204020504_e486c2be-cbfc-4134-8f04-d3a39aa2b965
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1707009169585_0006)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      1          1        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      1          1        0        0       0       0
Reducer 3 ...... container     SUCCEEDED      1          1        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 6.94 s
----------------------------------------------------------------------------------------------
OK
NULL    arrival_date_month      1
2015    August  3889
2015    December        2920
2015    July    2776
2015    November        2340
2015    October 4957
2015    September       5114
2016    April   5428
2016    August  5063
2016    December        3860
2016    February        3891
2016    January 2248
2016    July    4572
2016    June    5292
2016    March   4824
2016    May     5478
2016    November        4454
2016    October 6203
2016    September       5394
2017    April   5661
2017    August  4917
2017    February        4177
2017    January 3681
2017    July    5313
2017    June    5647
2017    March   4970
2017    May     6313
Time taken: 7.821 seconds, Fetched: 27 row(s)
```

Spark-Shell

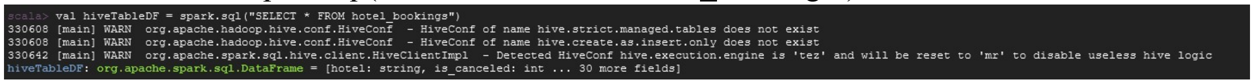
1. spark_shell

   a. 

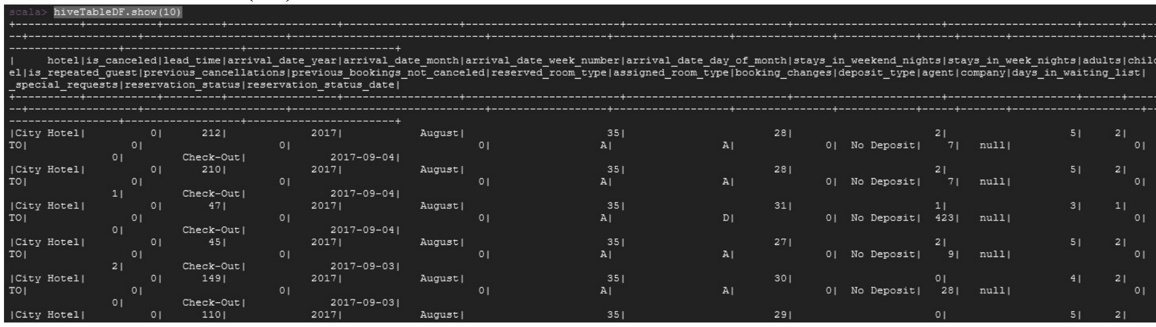
```
bash-5.0# spark-shell
```

2. import org.apache.spark.sql.SparkSession
   a. This line of code imports the SparkSession class from the Apache Spark SQL library, allowing the creation of a unified entry point to access data and execute Spark functionalities in a Spark application.
3. val spark = SparkSession.builder.appName("HotelBookingsSpark").enableHiveSupport().getOrCreate()
   a. This code initializes a SparkSession named "HotelBookingsSpark" with Hive support enabled, allowing seamless integration with Hive and creating a unified environment for executing Spark operations.
   b. 

```
scala> import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.SparkSession

scala> val spark = SparkSession.builder.appName("HotelBookingsSpark").enableHiveSupport().getOrCreate()
138015 [main] WARN  org.apache.spark.sql.SparkSession$Builder  - Using an existing SparkSession; the static sql configurations will not take effect.
138016 [main] WARN  org.apache.spark.sql.SparkSession$Builder  - Using an existing SparkSession; some spark core configurations may not take effect.
spark: org.apache.spark.sql.SparkSession = org.apache.spark.sql.SparkSession@4ba25170
```

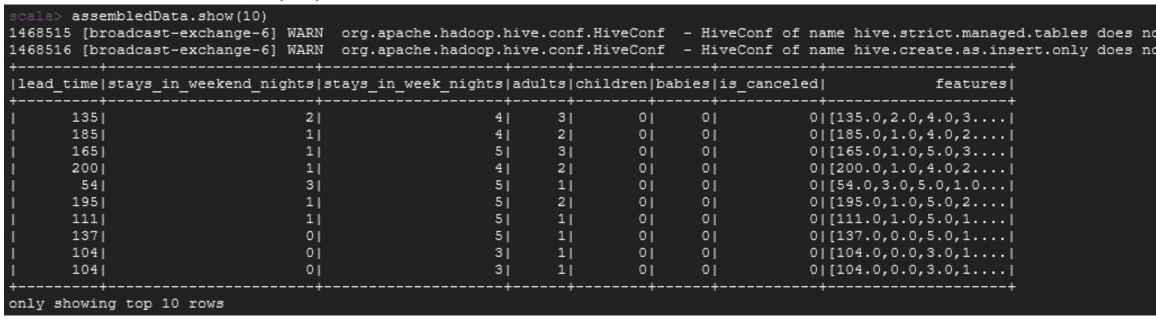
4. Read Hive Table into Spark DataFrame:
   a. val hiveTableDF = spark.sql("SELECT * FROM hotel_bookings")
   b. 

```
scala> val hiveTableDF = spark.sql("SELECT * FROM hotel_bookings")
330608 [main] WARN  org.apache.hadoop.hive.conf.HiveConf  - HiveConf of name hive.strict.managed.tables does not exist
330608 [main] WARN  org.apache.hadoop.hive.conf.HiveConf  - HiveConf of name hive.create.as.insert.only does not exist
330642 [main] WARN  org.apache.spark.sql.hive.client.HiveClientImpl  - Detected HiveConf hive.execution.engine is 'tez' and will be reset to 'mr' to disable useless hive logic
hiveTableDF: org.apache.spark.sql.DataFrame = [hotel: string, is_canceled: int ... 30 more fields]
```

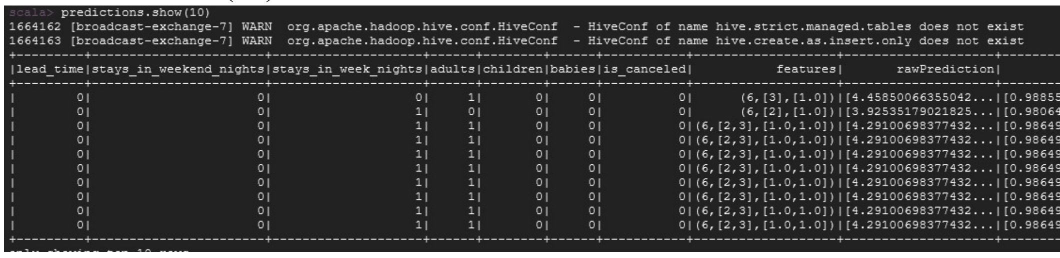
   c. Confirm data loaded
      i. hiveTableDF.show(10)
      ii. 

```
scala> hiveTableDF.show(10)
+----------+-----------+---------+-----------------+------------------+-----------------------+-------------------------+-----------------------+--------------------+------+----
--+------------------+----------------------+------------------------------+------------------+------------------+---------------+------------+-----+-------+--------------------+--
-----------------+------------------+----------------------+
|     hotel|is_canceled|lead_time|arrival_date_year|arrival_date_month|arrival_date_week_number|arrival_date_day_of_month|stays_in_weekend_nights|stays_in_week_nights|adults|chil
el|is_repeated_guest|previous_cancellations|previous_bookings_not_canceled|reserved_room_type|assigned_room_type|booking_changes|deposit_type|agent|company|days_in_waiting_list|
_special_requests|reservation_status|reservation_status_date|
+----------+-----------+---------+-----------------+------------------+-----------------------+-------------------------+-----------------------+--------------------+------+----
--+------------------+----------------------+------------------------------+------------------+------------------+---------------+------------+-----+-------+--------------------+--
-----------------+------------------+----------------------+
|City Hotel|          0|      212|             2017|            August|                     35|                       28|                      2|                   5|     2|
TO|                0|                     0|                             0|                 A|                 A|              0|  No Deposit|    7|   null|                   0|
                0|         Check-Out|              2017-09-04|
|City Hotel|          0|      210|             2017|            August|                     35|                       28|                      2|                   5|     2|
TO|                0|                     0|                             0|                 A|                 A|              0|  No Deposit|    7|   null|                   0|
                1|         Check-Out|              2017-09-04|
|City Hotel|          0|       47|             2017|            August|                     35|                       31|                      1|                   3|     1|
TO|                0|                     0|                             0|                 A|                 D|              0|  No Deposit|  423|   null|                   0|
                0|         Check-Out|              2017-09-04|
|City Hotel|          0|       45|             2017|            August|                     35|                       27|                      2|                   5|     2|
TO|                0|                     0|                             0|                 A|                 A|              0|  No Deposit|    9|   null|                   0|
                2|         Check-Out|              2017-09-03|
|City Hotel|          0|      149|             2017|            August|                     35|                       30|                      0|                   4|     2|
TO|                0|                     0|                             0|                 A|                 A|              0|  No Deposit|   28|   null|                   0|
                0|         Check-Out|              2017-09-03|
|City Hotel|          0|      110|             2017|            August|                     35|                       29|                      0|                   5|     2|
```

5. Prepare data for model
   a. Import Vector Assembler
      i. import org.apache.spark.ml.feature.VectorAssembler
   b. Identify Feature Columns into an array
      i. val featureCols = Array("lead_time", "stays_in_weekend_nights", "stays_in_week_nights", "adults", "children", "babies")
   c. Vector Assembler
      i. val assembler = new VectorAssembler().setInputCols(featureCols).setOutputCol("features").setHandleInvalid("keep")
         1. This code defines a VectorAssembler named "assembler" that combines specified input columns into a single feature column named "features," preserving invalid values during the assembly process.
      ii. val assembledData = assembler.transform(data)

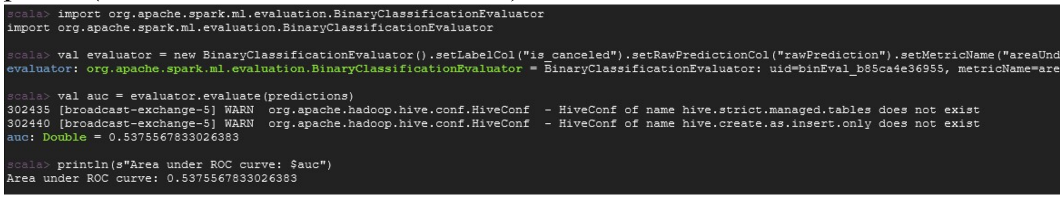
            1. This code applies the previously defined VectorAssembler ("assembler") to transform the input DataFrame "data," creating a new DataFrame named "assembledData" with the assembled features.
    d. validate assembled data
        i. assembledData.show(10)
        ii. 

```
scala> assembledData.show(10)
1468515 [broadcast-exchange-6] WARN  org.apache.hadoop.hive.conf.HiveConf  - HiveConf of name hive.strict.managed.tables does no
1468516 [broadcast-exchange-6] WARN  org.apache.hadoop.hive.conf.HiveConf  - HiveConf of name hive.create.as.insert.only does no
+---------+----------------------+--------------------+------+--------+------+-----------+--------------------+
|lead_time|stays_in_weekend_nights|stays_in_week_nights|adults|children|babies|is_canceled|            features|
+---------+----------------------+--------------------+------+--------+------+-----------+--------------------+
|      135|                     2|                   4|     3|       0|     0|          0|[135.0,2.0,4.0,3....|
|      185|                     1|                   4|     2|       0|     0|          0|[185.0,1.0,4.0,2....|
|      165|                     1|                   5|     3|       0|     0|          0|[165.0,1.0,5.0,3....|
|      200|                     1|                   4|     2|       0|     0|          0|[200.0,1.0,4.0,2....|
|       54|                     3|                   5|     1|       0|     0|          0|[54.0,3.0,5.0,1.0...|
|      195|                     1|                   5|     2|       0|     0|          0|[195.0,1.0,5.0,2....|
|      111|                     1|                   5|     1|       0|     0|          0|[111.0,1.0,5.0,1....|
|      137|                     0|                   5|     1|       0|     0|          0|[137.0,0.0,5.0,1....|
|      104|                     0|                   3|     1|       0|     0|          0|[104.0,0.0,3.0,1....|
|      104|                     0|                   3|     1|       0|     0|          0|[104.0,0.0,3.0,1....|
+---------+----------------------+--------------------+------+--------+------+-----------+--------------------+
only showing top 10 rows
```

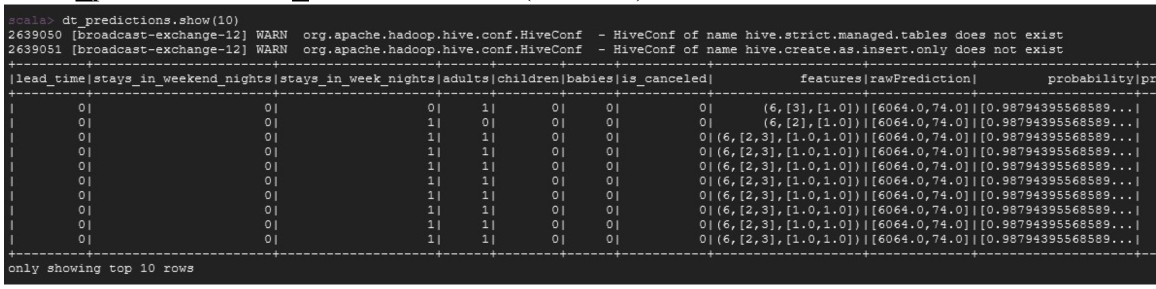
    e. Train/Test Split
        i. val Array(trainData, testData) = assembledData.randomSplit(Array(0.8, 0.2), seed = 123)
6. Train Logistic Regression
    a. Import and Train
        i. import org.apache.spark.ml.classification.LogisticRegression
        ii. val lr = new LogisticRegression().setLabelCol("is_canceled").setFeaturesCol("features")
            1. This code initializes a Logistic Regression model ("lr") and specifies the label and features columns for training, with the label column set to "is_canceled" and the features column set to "features."
        iii. val model = lr.fit(trainData)
    b. Predict on Test Data
        i. val predictions = model.transform(testData)
        ii. Validate Output
            1. Predictions.show(10)
            2. 

```
scala> predictions.show(10)
1664162 [broadcast-exchange-7] WARN  org.apache.hadoop.hive.conf.HiveConf  - HiveConf of name hive.strict.managed.tables does not exist
1664163 [broadcast-exchange-7] WARN  org.apache.hadoop.hive.conf.HiveConf  - HiveConf of name hive.create.as.insert.only does not exist
+---------+----------------------+--------------------+------+--------+------+-----------+--------------------+--------------------+------
|lead_time|stays_in_weekend_nights|stays_in_week_nights|adults|children|babies|is_canceled|            features|       rawPrediction|
+---------+----------------------+--------------------+------+--------+------+-----------+--------------------+--------------------+------
|        0|                     0|                   0|     1|       0|     0|          0|      (6,[3],[1.0])|[4.45850066355042...|[0.98855
|        0|                     0|                   1|     0|       0|     0|          0|      (6,[2],[1.0])|[3.92535179021825...|[0.98064
|        0|                     0|                   1|     1|       0|     0|          0|(6,[2,3],[1.0,1.0])|[4.29100698377432...|[0.98649
|        0|                     0|                   1|     1|       0|     0|          0|(6,[2,3],[1.0,1.0])|[4.29100698377432...|[0.98649
|        0|                     0|                   1|     1|       0|     0|          0|(6,[2,3],[1.0,1.0])|[4.29100698377432...|[0.98649
|        0|                     0|                   1|     1|       0|     0|          0|(6,[2,3],[1.0,1.0])|[4.29100698377432...|[0.98649
|        0|                     0|                   1|     1|       0|     0|          0|(6,[2,3],[1.0,1.0])|[4.29100698377432...|[0.98649
|        0|                     0|                   1|     1|       0|     0|          0|(6,[2,3],[1.0,1.0])|[4.29100698377432...|[0.98649
|        0|                     0|                   1|     1|       0|     0|          0|(6,[2,3],[1.0,1.0])|[4.29100698377432...|[0.98649
|        0|                     0|                   1|     1|       0|     0|          0|(6,[2,3],[1.0,1.0])|[4.29100698377432...|[0.98649
+---------+----------------------+--------------------+------+--------+------+-----------+--------------------+--------------------+------
```

    c. Evaluate Model
        i. import org.apache.spark.ml.evaluation.BinaryClassificationEvaluator
        ii. val evaluator = new BinaryClassificationEvaluator().setLabelCol("is_canceled").setRawPredictionCol("rawPrediction").setMetricName("areaUnderROC")

            1. This code creates a Binary Classification Evaluator ("evaluator") for evaluating the Logistic Regression model, specifying the label column as "is_canceled," raw prediction column as "rawPrediction," and metric name as "areaUnderROC."
        iii. val auc = evaluator.evaluate(predictions)
        iv. show evaluation
            1. println(s"Area under ROC curve: $auc")
            2. 

```
scala> import org.apache.spark.ml.evaluation.BinaryClassificationEvaluator
import org.apache.spark.ml.evaluation.BinaryClassificationEvaluator

scala> val evaluator = new BinaryClassificationEvaluator().setLabelCol("is_canceled").setRawPredictionCol("rawPrediction").setMetricName("areaUnd
evaluator: org.apache.spark.ml.evaluation.BinaryClassificationEvaluator = BinaryClassificationEvaluator: uid=binEval_b85ca4e36955, metricName=are

scala> val auc = evaluator.evaluate(predictions)
302435 [broadcast-exchange-5] WARN  org.apache.hadoop.hive.conf.HiveConf  - HiveConf of name hive.strict.managed.tables does not exist
302440 [broadcast-exchange-5] WARN  org.apache.hadoop.hive.conf.HiveConf  - HiveConf of name hive.create.as.insert.only does not exist
auc: Double = 0.5375567833026383

scala> println(s"Area under ROC curve: $auc")
Area under ROC curve: 0.5375567833026383
```

7. Train Decision Tree
    a. Import and Train
        i. import org.apache.spark.ml.classification.DecisionTreeClassifier
        ii. val dt = new DecisionTreeClassifier().setLabelCol("is_canceled").setFeaturesCol("features")
        iii. val dt_model = dt.fit(trainData)
    b. Predict
        i. val dt_predictions = dt_model.transform(testData)
        ii. 

```
scala> dt_predictions.show(10)
2639050 [broadcast-exchange-12] WARN  org.apache.hadoop.hive.conf.HiveConf  - HiveConf of name hive.strict.managed.tables does not exist
2639051 [broadcast-exchange-12] WARN  org.apache.hadoop.hive.conf.HiveConf  - HiveConf of name hive.create.as.insert.only does not exist
+---------+----------------------+--------------------+------+--------+------+-----------+--------------------+-------------+--------------------+--
|lead_time|stays_in_weekend_nights|stays_in_week_nights|adults|children|babies|is_canceled|            features|rawPrediction|         probability|pr
+---------+----------------------+--------------------+------+--------+------+-----------+--------------------+-------------+--------------------+--
|        0|                     0|                   0|     1|       0|     0|          0|      (6,[3],[1.0])|[6064.0,74.0]|[0.98794395568589...|
|        0|                     0|                   1|     0|       0|     0|          0|      (6,[2],[1.0])|[6064.0,74.0]|[0.98794395568589...|
|        0|                     0|                   1|     1|       0|     0|          0|(6,[2,3],[1.0,1.0])|[6064.0,74.0]|[0.98794395568589...|
|        0|                     0|                   1|     1|       0|     0|          0|(6,[2,3],[1.0,1.0])|[6064.0,74.0]|[0.98794395568589...|
|        0|                     0|                   1|     1|       0|     0|          0|(6,[2,3],[1.0,1.0])|[6064.0,74.0]|[0.98794395568589...|
|        0|                     0|                   1|     1|       0|     0|          0|(6,[2,3],[1.0,1.0])|[6064.0,74.0]|[0.98794395568589...|
|        0|                     0|                   1|     1|       0|     0|          0|(6,[2,3],[1.0,1.0])|[6064.0,74.0]|[0.98794395568589...|
|        0|                     0|                   1|     1|       0|     0|          0|(6,[2,3],[1.0,1.0])|[6064.0,74.0]|[0.98794395568589...|
|        0|                     0|                   1|     1|       0|     0|          0|(6,[2,3],[1.0,1.0])|[6064.0,74.0]|[0.98794395568589...|
|        0|                     0|                   1|     1|       0|     0|          0|(6,[2,3],[1.0,1.0])|[6064.0,74.0]|[0.98794395568589...|
+---------+----------------------+--------------------+------+--------+------+-----------+--------------------+-------------+--------------------+--
only showing top 10 rows
```

    c. Evaluate
        i. val dt_evaluator = new BinaryClassificationEvaluator().setLabelCol("is_canceled").setRawPredictionCol("rawPrediction").setMetricName("areaUnderROC")
        ii. val dt_auc = dt_evaluator.evaluate(dt_predictions)
        iii. println(s"Area under ROC curve: $dt_auc")

References:

Antonio, N. (2019, February). Hotel Booking Demand, Version 1. Retrieved February 4, 2024 from

https://www.kaggle.com/datasets/jessemostipak/hotel-booking-demand/data.