Hotel Booking Tourism Analysis

Carlos Cantu

DSC 650

Nasheb Ismaily

Hotel Booking Tourism Analysis

Tourism plays a significant role in the global economy, with hotels serving as vital components of the hospitality sector. Understanding the dynamics of hotel bookings is crucial for hotel managers, marketers, and policymakers alike. In this analysis, we delve into the intricacies of hotel booking data to uncover trends, preferences, and patterns that can inform decision-making processes within the hospitality industry.

The primary objective of this analysis is to process and analyze hotel booking data to identify trends, booking preferences, and cancellations. By exploring various aspects of hotel bookings, including booking patterns, customer demographics, and booking cancellations, we aim to gain insights that can help hotel managers optimize their operations, tailor their marketing strategies, and enhance customer satisfaction.

The dataset used for this analysis is the "Hotel booking demand" dataset, sourced from Kaggle. This dataset provides comprehensive information on hotel bookings, including booking dates, customer demographics, booking channels, and booking cancellations. By leveraging this dataset, we can explore a wide range of factors that influence hotel bookings and uncover valuable insights to support decision-making processes within the hospitality industry.

For this analysis, we will utilize HDFS (Hadoop Distributed File System) and PySpark, a Python API for Apache Spark. HDFS will serve as the storage system for our data, providing scalability and fault tolerance for handling large volumes of hotel booking data. PySpark, running on top of Spark, will enable us to efficiently process and analyze the data in parallel, leveraging distributed computing capabilities to handle big data analytics tasks effectively.

Processing Data Load

The analysis is conducted within the Google Cloud framework, leveraging its infrastructure for data processing. As a preparatory step and data host, a new directory named "data" is established within the Hadoop Distributed File System (HDFS). Subsequently, the hotel_bookings.csv file is uploaded to HDFS utilizing the wget technique. This file is then seamlessly transferred into the designated folder within HDFS, ensuring its accessibility for future utilization with Spark, a powerful distributed computing engine. This streamlined process ensures efficient data handling and enables seamless integration with Spark for subsequent analysis and insights extraction.

```
bash-5.0# wget "https://storage.googleapis.com/kaggle-data-sets/40850/85274/compressed/hotel_bookings.csv.zip?X
-Goog-Algorithm=GOOG4-RSA-SHA256&X-Goog-Credential=gcp-kaggle-com%40kaggle-161607.iam.gserviceaccount.com%2F202
40508%2Fauto%2Fstorage%2Fgoog4_request&X-Goog-Date=20240508T160803Z&X-Goog-Expires=259199&X-Goog-SignedHeaders=
host&X-Goog-Signature=956ad0454d798f3548627a8e3fca2da5c9b73bc04209f35a99d6e3a52b264745b7c4b1232c2c4d00806e1f579
6518f962cde11a4b5e41a7aee4c2f2c5e939c931a778b7dc25c81d32b8986878cd5e78a0a6c70f78d1d1c4405da4f25fc61b8d040f24420
ec5e81861702700a063fba23b47e383f295c5e2ab9281b5a4fc23a0c1b2541e4cccd7b9d2c045823130e3cf8ef9a44c70a816282b01202c
1c1690f800db17ac5d54701a6c4e1f14bebf5c4422007f6de6ad1f87a422a19562e7d8b148097bf2b36cb6201a3a574d899e38b3e8f19c1
e30372f1b3c87e8691e5e15cc0fd03d22ec4a1a17d7f68ebe7daa1aa7f7b665d74dfcd35215d3e63df3da7ad0e995c765e6d5ff7e6f7d"
Connecting to storage.googleapis.com (142.251.2.207:443)
wget: server returned error: HTTP/1.1 400 Bad Request
bash-5.0# wget https://raw.githubusercontent.com/cantu2555/cantu2555/main/hotel_bookings.csv
Connecting to raw.githubusercontent.com (185.199.108.133:443)
hotel_bookings.csv   100% |*******************************************************************| 16.0M  0:00:00 ETA
bash-5.0#
bash-5.0# hdfs dfs -put hotel_bookings.csv /data/
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/slf
4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/program/tez/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBin
der.class]
SLF4J: Found binding in [jar:file:/usr/program/hive/lib/log4j-slf4j-impl-2.10.0.jar!/org/slf4j/impl/StaticLogge
rBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
2024-05-14 20:14:00,963 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... u
sing builtin-java classes where applicable
bash-5.0#
```

```
Found 1 items
-rw-r--r--   1 root supergroup   16855599 2024-05-14 20:14 /data/hotel_bookings.csv
bash-5.0#
```

Once the data is confirmed to be securely stored in its designated folder, Spark is initiated for data processing. PySpark, the Python API for Spark, is chosen for its familiar and simplified coding experience compared to Scala. The data is then loaded into Spark for analysis. During this process, a warning is encountered indicating that the data representation was truncated due to its size, which prevented a simplified view. This warning underscores the vastness of the dataset and highlights the need for advanced data handling techniques to effectively manage and analyze large volumes of information.

```
bash-5.0# pyspark
Python 3.7.10 (default, Mar  2 2021, 09:06:08)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/program/spark/jars/slf4j-log4j12-1.7.30.jar!/org/slf4j/impl/StaticLogger
Binder.class]
SLF4J: Found binding in [jar:file:/usr/program/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf
4j/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]
1    [main] WARN  org.apache.hadoop.util.NativeCodeLoader  - Unable to load native-hadoop library for your plat
form... using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
884  [Thread-4] WARN  org.apache.hadoop.hive.conf.HiveConf  - HiveConf of name hive.strict.managed.tables does
not exist
884  [Thread-4] WARN  org.apache.hadoop.hive.conf.HiveConf  - HiveConf of name hive.create.as.insert.only does
not exist
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 3.0.0
      /_/

Using Python version 3.7.10 (default, Mar  2 2021 09:06:08)
SparkSession available as 'spark'.
>>> df = spark.read.format('csv').option('header', 'true').load('/data/hotel_bookings.csv')

>>> df.show()
275670 [Thread-4] WARN  org.apache.spark.sql.catalyst.util.package  - Truncated the string representation of a
plan since it was too large. This behavior can be adjusted by setting 'spark.sql.debug.maxToStringFields'.
+------------+-----------+---------+----------------+------------------+----------------------+-------------
------------+----------------------+-------------------+------+--------+------+----+-------+--------------+--
------------------+----------------+----------------------+-----------------------------+------------------+-
------------------+---------------+------------+------+-------+-------------------+-------------+------+--------
-------------------+------------------------+-----------------+----------------------+
|       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|di
stribution_channel|is_repeated_guest|previous_cancellations|previous_bookings_not_canceled|reserved_room_type|a
ssigned_room_type|booking_changes|deposit_type|agent|company|days_in_waiting_list|customer_type|   adr|required
_car_parking_spaces|total_of_special_requests|reservation_status|reservation_status_date|
+------------+-----------+---------+----------------+------------------+----------------------+-------------
------------+----------------------+-------------------+------+--------+------+----+-------+--------------+--
------------------+----------------+----------------------+-----------------------------+------------------+-
------------------+---------------+------------+------+-------+-------------------+-------------+------+--------
-------------------+------------------------+-----------------+----------------------+
|Resort Hotel|          0|      342|             2015|              July|                    27|
```

Analysis

An initial assessment was conducted to evaluate the cleanliness and utilization of the dataset. Overall, the dataset was found to be relatively clean, with minimal inconsistencies or missing values. This favorable condition allowed us to proceed with confidence in our analysis without the need for extensive data cleaning or preprocessing efforts. With the dataset deemed suitable for analysis, we could focus our efforts on extracting valuable insights and uncovering meaningful patterns.

Cancellations:

The initial focus of the analysis was to identify patterns related to hotel booking cancellations. Understanding when hotels can expect a high number of cancellations is crucial for revenue management, operational efficiency, customer service, resource allocation, and marketing strategies. By uncovering patterns in cancellations, hotels can optimize their operations, allocate resources effectively, improve customer satisfaction, and implement targeted marketing and promotional efforts. This insight-driven approach enables hotels to make informed decisions and adapt their strategies to minimize the impact of cancellations on their business performance.

To determine when cancellations peak, the dataset was initially grouped by the arrival month of bookings. This grouping allowed us to understand the distribution of hotel stays throughout the year. Subsequently, cancellations were summed within each arrival month group to calculate the total cancellations for that specific month. This step provided insight into the volume of cancellations occurring during different months of the year. After summing cancellations for each arrival month, the results were aggregated for the entire dataset. This aggregation involved summing the cancellations across all arrival months to identify overall

trends. The aggregated results were then sorted in descending order based on the total number of cancellations, revealing the months with the highest cancellation rates. This analysis allows hotel management to anticipate periods of high cancellation activity.

The analysis revealed that August, July, May, and June are the months with the highest number of cancellations, with August being the most affected, recording 5,239 cancellations. This trend suggests that summer months experience a peak in cancellations, likely due to factors such as changes in travel plans, vacation schedules, or other seasonal factors. Understanding this trend allows hotel managers to anticipate and prepare for increased cancellation activity during these months. By implementing proactive strategies such as flexible booking policies, targeted promotions, or overbooking management, hotels can mitigate the impact of cancellations and optimize their revenue and customer satisfaction levels during peak cancellation periods.

```
>>> from pyspark.sql import functions as F
>>>
>>> # Group by arrival_date_month and calculate the sum of is_canceled
>>> cancellations_by_month = df.groupBy("arrival_date_month") \
...     .agg(F.sum("is_canceled").alias("total_cancellations")) \
...     .orderBy(F.col("total_cancellations").desc())
>>>
>>> # Show the result
>>> cancellations_by_month.show()
+------------------+-------------------+
|arrival_date_month|total_cancellations|
+------------------+-------------------+
|            August|             5239.0|
|              July|             4742.0|
|               May|             4677.0|
|              June|             4535.0|
|             April|             4524.0|
|           October|             4246.0|
|         September|             4116.0|
|             March|             3149.0|
|          February|             2696.0|
|          December|             2371.0|
|          November|             2122.0|
|           January|             1807.0|
+------------------+-------------------+
```

Weekend Stays:

Understanding whether city or resort hotels attract more weekend stays is crucial for hoteliers to tailor their marketing efforts effectively. City hotels may prioritize business travelers during weekdays and shift focus to leisure travelers on weekends, while resort hotels may target families or couples seeking weekend getaways. Analyzing the distribution of weekend stays between city and resort hotels provides valuable insights for strategic planning, revenue optimization, and delivering personalized experiences to guests.

To determine this, the stays on weekend nights were grouped by hotel type – resort or city. This analysis aimed to identify which type of hotel leads in weekend stays. By comparing the weekend stay patterns between city and resort hotels, hoteliers can adjust their marketing strategies, allocate resources efficiently, and enhance guest experiences based on the unique characteristics and preferences of their target audience.

The results of our transformations reveal that city hotels experience significantly more weekend stays compared to resorts. This finding provides valuable insights for hoteliers, indicating that city hotels attract a larger proportion of guests seeking weekend accommodations. With this understanding, hotel managers should adjust their marketing strategies, allocate resources, and tailor their services to better cater to the needs and preferences of weekend guests at city hotels. By capitalizing on the high demand for weekend stays in city locations, hoteliers should optimize their revenue and enhance guest satisfaction levels.

```
>>> # Group by hotel and calculate the sum of stays_in_weekend_nights
>>> most_weekend_nights_hotel = df.groupBy("hotel") \
...     .agg(F.sum("stays_in_weekend_nights").alias("total_weekend_nights")) \
...     .orderBy(F.col("total_weekend_nights").desc())
>>>
>>> # Show the result
>>> most_weekend_nights_hotel.show()
+------------+--------------------+
|       hotel|total_weekend_nights|
+------------+--------------------+
|  City Hotel|             63082.0|
|Resort Hotel|             47664.0|
+------------+--------------------+
```

Lead Time:

The final investigation focuses on lead time, the duration between booking and stay dates, which holds critical importance for hotels across various operational facets. Lead time serves as a cornerstone for revenue management, providing a look into demand patterns and guiding pricing and inventory decisions. By understanding lead time data, hotels need to optimize revenue streams and ensure efficient allocation of resources.

Statistics were gathered for the column 'lead_time' through the describe function to obtain insights into the average duration hotels can expect from booking to stay. It was determined that the mean lead time was approximately 104 days, indicating that hotels can typically expect a period of 104 days between the booking date and the stay date. This average lead time allows hotels to anticipate guest arrivals, optimize their revenue management strategies, and efficiently allocate resources to meet guest needs during their stay.

```
>>> lead_time_stats = df.select("lead_time").describe()
>>> lead_time_stats.show()
+-------+------------------+
|summary|         lead_time|
+-------+------------------+
|  count|            119390|
|   mean|104.01141636652986|
| stddev| 106.8630970479881|
|    min|                 0|
|    max|                99|
+-------+------------------+
```

Conclusion

The analysis of hotel booking data has yielded significant insights into various facets of the hospitality industry. Trends in booking preferences, peak cancellation periods, and the distribution of weekend stays between city and resort hotels have been thoroughly explored. Additionally, the examination of lead time has underscored its critical importance across revenue management, forecasting, operational efficiency, marketing, and customer service strategies within the hotel sector.

In conclusion, the analysis of hotel booking data has unveiled valuable insights for the hospitality industry's success. Through the utilization of HDFS and Spark, not only have large volumes of data been efficiently managed, but advanced analytics have been additionally leveraged to extract actionable insights. HDFS provided a scalable and fault-tolerant storage system, ensuring data integrity and accessibility, while Spark empowered us to process and analyze the data in parallel, enabling swift and comprehensive analysis.

Looking ahead, the integration of live streaming bookings into an analytical model holds immense promise for revolutionizing resource allocation in the hospitality sector. By continuously ingesting real-time booking data streams, hotels can dynamically adjust their

resource allocation strategies in response to fluctuating demand. This proactive approach ensures optimal utilization of resources, maximizes revenue potential, and enhances the overall guest experience.

Incorporating live streaming bookings into an analytical framework not only streamlines operational efficiency, but also enables hotels to stay ahead of market trends and customer preferences in real-time. As technology continues to evolve, embracing innovations like live streaming data analytics will be paramount for hotels to remain competitive and agile in an ever-changing landscape.

Citations

Mostipak, J. (2020). Hotel Booking Demand. Retrieved from Kaggle: https://www.kaggle.com/jessemostipak/hotel-booking-demand