

Name:

Bagato, Carl Erick
Cancela, Mizpa Mae
Condat, Mary Grace
Granadil, Heaven

Lab No. Lab#.py

Git Repo/ Colab Link:

<https://colab.research.google.com/drive/1fL6T6BWVz0kxrJW4SltViYbAtLQCoaLr?usp=sharing>

Date: 02/06/25

Objective

The objective of this lab is to explore partitioning strategies and transformation techniques in Apache Spark. The focus is on optimizing data distribution using partitioning methods and performing key transformations such as summarization, sorting, and filtering to extract insights from dataset collected.

Introduction

Apache Spark is a powerful distributed computing framework designed for processing large-scale datasets efficiently. This lab focuses on fundamental data transformation techniques using PySpark, specifically partitioning strategies, data transformations, and aggregations. Partitioning strategies include understanding how data is distributed across partitions using `repartition()`, `repartitionByRange()`, and partitioning by column. The key data transformations include summarizing data by counting bookings per hotel type, sorting data based on arrival dates, and filtering records based on country. Additionally, aggregations and analysis are performed by grouping data to analyze trends in booking volume and pricing (ADR - Average Daily Rate) across different years.

The dataset used in this lab is sourced from Kaggle and contains hotel booking records. It includes key fields such as `hotel`, `lead_time`, `arrival_date_year`, `arrival_date_month`, `stays_in_weekend_nights`, `stays_in_week_nights`, `adults`, `children`, `meal`, `country`, `market_segment`, `reserved_room_type`, `assigned_room_type`, `booking_changes`, `deposit_type`, `days_in_waiting_list`, `customer_type`, `adr` (average daily rate), `total_of_special_requests`, `reservation_status`, and `reservation_status_date`, among others. These fields provide critical information for analyzing booking trends, customer behavior, and pricing patterns.

Methodology

1. Data Loading
 - 1.1. Read the dataset `hotel_bookings.csv` into a PySpark `DataFrame`.
 - 1.2. Print the schema to understand column data types.
2. Partitioning Strategies
 - 2.1. Partition by Column: Partition the data by hotel type.
 - 2.2. Repartitioning: Adjust the number of partitions to 4 using `repartition(4)`.

- 2.3. Range Partitioning: Apply `repartitionByRange()` based on `adr` and `arrival_date_year` to optimize data distribution.
3. Data Transformations
 - 3.1. Summarization: Count bookings per hotel type.
 - 3.2. Sorting: Order data by `arrival_date_year` and `arrival_date_month` in descending order.
 - 3.3. Filtering: Extract bookings where country is "USA".
 - 3.4. Aggregation:
 - 3.4.1. Categorize ADR values into "Low", "Medium", and "High" ranges.
 - 3.4.2. Compute the average ADR per year (2015, 2016, 2017).

Results and Analysis

The results demonstrate that city hotels had a significantly higher number of bookings compared to resort hotels. Sorting the data by arrival date revealed a steady increase in hotel reservations over the years, with notable peaks in certain months. Filtering by country showed that the USA had a substantial number of bookings, contributing significantly to the overall dataset.

Further analysis of the ADR values indicated that most bookings fell within the "Medium" category (ADR between 50 and 150), while "Low" ADR bookings (ADR < 50) were relatively rare. The average ADR increased consistently from 2015 to 2017, highlighting a trend of rising room rates over time. This suggests potential factors such as inflation, higher demand, or seasonal price adjustments influencing the ADR values. Additionally, range partitioning by ADR and arrival year optimized the data distribution, improving query performance and analysis efficiency.

Challenges and Solutions

One of the main challenges encountered in this lab was handling the large dataset efficiently, as operations such as sorting and aggregations were slow. To address this, we applied partitioning strategies, including repartitioning and range partitioning, to optimize data distribution and improve query performance.

Conclusion

This lab demonstrated effective partitioning and transformation techniques using PySpark. By leveraging different partitioning strategies, we optimized data distribution, leading to improved performance. The transformation pipeline provided key insights into hotel booking trends, ADR variations, and country-wise booking distributions. The increasing trend in ADR suggests a rise in pricing over the years, and range partitioning further enhanced efficiency in analyzing pricing trends. The lab reinforced the importance of data partitioning, filtering, sorting, and aggregation in large-scale data processing.

Documentation

```
hotel_booking_colab.ipynb
File Edit View Insert Runtime Tools Help
+ Code + Text
Connect Gemini
!pip install pyspark

from pyspark.sql import SparkSession
from pyspark.sql.functions import col, count, desc
from pyspark.sql.functions import when
from pyspark.sql.functions import avg

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

# Load the dataset (Ensure correct file path)
file_path = "/content/hotel_bookings.csv"
df = spark.read.csv(file_path, header=True, inferSchema=True)

# Show schema to understand the data
df.printSchema()

# Partitioning Strategy 1: Partition by 'hotel' column
df_partitioned = df.repartition("hotel")

# Partitioning Strategy 2: Repartition into a fixed number of partitions (e.g., 4)
df_repartitioned = df.repartition(4)

# Transformation Pipeline
# 1. Summarize data - Count bookings per hotel type
summary_df = df.groupBy("hotel").agg(count("*").alias("booking_count"))

# 2. Sorting - Sort by arrival_date_year and arrival_date_month
sorted_df = df.orderBy(desc("arrival_date_year"), desc("arrival_date_month"))

# 3. Filtering - Filter bookings for a specific country (e.g., 'USA')
filtered_df = df.filter(col("country") == "USA")

# Show results
print("Summary of Bookings per Hotel Type:")
summary_df.show()

print("Sorted Bookings by Arrival Date:")
sorted_df.show(5) # Show top 5 sorted results

print("Filtered Bookings for USA:")
filtered_df.show(5) # Show top 5 results

# Filter the dataset for the years 2015, 2016, and 2017
df_filtered_years = df.filter((col("arrival_date_year") == 2015) |
                               (col("arrival_date_year") == 2016) |
                               (col("arrival_date_year") == 2017))

# Partitioning Strategy 4: Range Partitioning by 'adr' and 'arrival_date_year'
df_range_partitioned = df_filtered_years.repartitionByRange("adr", "arrival_date_year")

print("Range Partitioning by 'adr' and 'arrival_date_year' done!")

# Transformation 3: Summarize bookings by 'adr' range and year (Low, Medium, High)
adr_summary_df = df_filtered_years.withColumn("adr_range",
                                              when(col("adr") < 50, "Low")
                                              .when((col("adr") >= 50) & (col("adr") < 150), "Medium")
                                              .otherwise("High"))
adr_summary_df.groupBy("arrival_date_year", "adr_range").agg(count("*").alias("booking_count"))

# Show the summarized results
print("Summary of Bookings by 'adr' Range and Year (2015, 2016, 2017):")
adr_summary_df.orderBy("arrival_date_year", "adr_range").show()

# Transformation 4: Calculate the average 'adr' per year (2015, 2016, 2017)
adr_avg_df = df_filtered_years.groupBy("arrival_date_year").agg(avg("adr").alias("average_adr"))

# Show the average adr per year
print("Average ADR per Year (2015, 2016, 2017):")
adr_avg_df.orderBy("arrival_date_year").show()
```

Requirement already satisfied: pyspark in /usr/local/lib/python3.11/dist-packages (3.5.4)
Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python3.11/dist-packages (from pyspark) (0.10.9.7)
root

```
-- hotel: string (nullable = true)
-- is_canceled: integer (nullable = true)
-- lead_time: integer (nullable = true)
-- arrival_date_year: integer (nullable = true)
-- arrival_date_month: string (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)
-- adults: integer (nullable = true)
-- children: string (nullable = true)
-- babies: integer (nullable = true)
-- meal: string (nullable = true)
-- country: string (nullable = true)
-- market_segment: string (nullable = true)
-- distribution_channel: string (nullable = true)
-- is_repeated_guest: integer (nullable = true)
-- previous_cancellations: integer (nullable = true)
-- previous_bookings_not_canceled: integer (nullable = true)
-- reserved_room_type: string (nullable = true)
-- assigned_room_type: string (nullable = true)
-- booking_changes: integer (nullable = true)
-- deposit_type: string (nullable = true)
-- agent: string (nullable = true)
-- company: string (nullable = true)
-- days_in_waiting_list: integer (nullable = true)
-- customer_type: string (nullable = true)
-- adr: double (nullable = true)
-- required_car_parking_spaces: integer (nullable = true)
-- total_of_special_requests: integer (nullable = true)
-- reservation_status: string (nullable = true)
-- reservation_status_date: string (nullable = true)
```

Summary of Bookings per Hotel Type:

```
+-----+-----+
| hotel|booking_count|
+-----+-----+
| City Hotel| 79330|
| Resort Hotel| 40060|
+-----+-----+
```

Sorted Bookings by Arrival Date:

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
City Hotel	0	1	2017	May	18	3	0	1	1	0	0	BB	PR
Resort Hotel	1	72	2017	May	18	1	1	2	2	2	0	BB	ES
City Hotel	0	6	2017	May	19	10	0	2	1	0	0	BB	PR
Resort Hotel	1	208	2017	May	18	1	1	4	2	0	0	BB	GB
City Hotel	0	1	2017	May	20	15	1	1	1	0	0	BB	PR

only showing top 5 rows

Filtered Bookings for USA:

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
Resort Hotel	0	68	2015	July	27	1	0	4	2	0	0	BB	US
Resort Hotel	0	14	2015	July	27	2	0	2	2	0	0	BB	US
Resort Hotel	0	10	2015	July	27	3	0	2	2	2	0	BB	US
Resort Hotel	0	9	2015	July	27	3	0	1	2	0	0	BB	US
Resort Hotel	0	51	2015	July	28	6	1	3	2	0	0	BB	US

only showing top 5 rows

Range Partitioning by 'adr' and 'arrival_date_year' done!

Summary of Bookings by 'adr' Range and Year (2015, 2016, 2017):

arrival_date_year	adr_range	booking_count
2015	High	2085
2015	Low	3001
2015	Medium	16910
2016	High	6285
2016	Low	5528
2016	Medium	44894
2017	High	8517
2017	Low	3060
2017	Medium	29110