Project Source Code

Airline Dataset

Group 2

Ritika Aggarwal

Aiswarya Baby

Sevim Shafizadegan

Dulanjani Rajapakshe

Insight 1. Which nationalities contribute the most to travel demand.

```
spark.read.option("header", "true").option("inferschema", "true").csv("/user/root/project/AirlineDatasetUpdatedv2.csv").createOrReplaceTempView("AirlineDataset")

result = spark.sql("""SELECT Nationality,
        COUNT(*) AS total_travelers,
        ROUND((COUNT(*) * 100.0) / SUM(COUNT(*)) OVER (), 2) AS percentage
    FROM AirlineDataset
    GROUP BY Nationality
    ORDER BY total_travelers DESC
""")

result.show(truncate=False)

# Specify the output path for the CSV file

output_path = "/user/root/project/Nationality_percent.csv"

result.coalesce(1).write.csv("/user/root/project/ Nationality_percent.csv", header=True, mode='overwrite')
```

Nationalities with Status

```
result = spark.sql(""" SELECT Nationality, COUNT(CASE WHEN `Flight Status` = 'Cancelled' THEN 1 END) AS Cancelled, COUNT(CASE WHEN `Flight Status` = 'On Time' THEN 1 END) AS On_Time, COUNT(CASE WHEN `Flight Status` = 'Delayed' THEN 1 END) AS Delayed_Flights, COUNT(*) AS Total_Travellers FROM AirlineDataset GROUP BY Nationality ORDER BY Total_Travellers DESC""")

result.show(10, truncate=False)

# Specify the output path for the CSV file
```

```
output_path = "/user/root/project/Nationality_with_status.csv"

result.coalesce(1).write.csv("/user/root/project/Nationality_with_status.csv", header=True, mode='overwrite')
```

Insight 2. What are the favorite or most visited destinations for frequent travelers

```
spark.read.option("header", "true").option("inferSchema", "true").csv("/user/root/project/AirlineDatasetUpdatedv2.csv").createOrReplaceTempView("AirlineDataset")

# Find the top 5 nationalities by traveler count

top_nationalities = spark.sql("""SELECT Nationality, COUNT(*) AS total_travelers FROM AirlineDataset GROUP BY Nationality ORDER BY total_travelers DESC""")

top_nationalities_list = [row['Nationality'] for row in top_nationalities.collect()]

# Convert the list of nationalities into a string for use in the SQL query

nationalities_str = ", ".join(["'{}'".format(nat) for nat in top_nationalities_list])

# Find the top 5 countries visited by the top nationalities

query = """SELECT `Country Name`, COUNT(*) AS total_travelers, ROUND((COUNT(*) * 100.0 / SUM(COUNT(*)) OVER()), 2) AS percentage_of_travelers FROM AirlineDataset WHERE Nationality IN ({}) GROUP BY `Country Name` ORDER BY total_travelers DESC """.format(nationalities_str)

result = spark.sql(query)

top_countries_list = [

  {

    "Country Name": row['Country Name'],

    "Total Travelers": row['total_travelers'],

    "Percentage": row['percentage_of_travelers']
```

```
    }
    for row in result.collect()
]
columns = ["Country Name", "Total Travelers", "Percentage"]
top_countries_df = spark.createDataFrame(
    [(row["Country Name"], row["Total Travelers"], row["Percentage"]) for row in top_countries_list],
    columns
)
top_countries_df.show(truncate=False)
# Specify the output path for the CSV file
output_path = "/user/root/project/insight4_top5_new.csv"
top_countries_df.coalesce(1).write.csv("/user/root/project/insight4_top5_new.csv", header=True, mode='overwrite')
```

Insight 3. Passenger demographics based on Age

```
result = spark.sql("""
SELECT Age_Group, Cancelled, Delayed, On_Time, Grand_Total
FROM (
    SELECT
        CASE
            WHEN Age BETWEEN 1 AND 10 THEN '1-10'
            WHEN Age BETWEEN 11 AND 20 THEN '11-20'
```

```
        WHEN Age BETWEEN 21 AND 30 THEN '21-30'
        WHEN Age BETWEEN 31 AND 40 THEN '31-40'
        WHEN Age BETWEEN 41 AND 50 THEN '41-50'
        WHEN Age BETWEEN 51 AND 60 THEN '51-60'
        WHEN Age BETWEEN 61 AND 70 THEN '61-70'
        WHEN Age BETWEEN 71 AND 80 THEN '71-80'
        WHEN Age BETWEEN 81 AND 90 THEN '81-90'
        ELSE 'Unknown'
    END AS Age_Group,
    SUM(CASE WHEN Flight_Status = 'Cancelled' THEN 1 ELSE 0 END) AS Cancelled,
    SUM(CASE WHEN Flight_Status = 'Delayed' THEN 1 ELSE 0 END) AS Delayed,
    SUM(CASE WHEN Flight_Status = 'On Time' THEN 1 ELSE 0 END) AS On_Time,
    COUNT(*) AS Grand_Total
FROM airline_data
GROUP BY
    CASE
        WHEN Age BETWEEN 1 AND 10 THEN '1-10'
        WHEN Age BETWEEN 11 AND 20 THEN '11-20'
        WHEN Age BETWEEN 21 AND 30 THEN '21-30'
        WHEN Age BETWEEN 31 AND 40 THEN '31-40'
        WHEN Age BETWEEN 41 AND 50 THEN '41-50'
        WHEN Age BETWEEN 51 AND 60 THEN '51-60'
```

```
        WHEN Age BETWEEN 61 AND 70 THEN '61-70'
        WHEN Age BETWEEN 71 AND 80 THEN '71-80'
        WHEN Age BETWEEN 81 AND 90 THEN '81-90'
        ELSE 'Unknown'
    END
) t
ORDER BY Age_Group
""")
result.show(truncate=False)
result.coalesce(1).write.csv("/user/root/project/demo2.csv", header=True, mode='overwrite')
```

Passenger Demographics based on Gender

```
result = spark.sql("""
SELECT Gender,
    SUM(CASE WHEN Flight_Status = 'Cancelled' THEN 1 ELSE 0 END) AS Cancelled,
    SUM(CASE WHEN Flight_Status = 'Delayed' THEN 1 ELSE 0 END) AS Delayed,
    SUM(CASE WHEN Flight_Status = 'On Time' THEN 1 ELSE 0 END) AS On_Time,
    COUNT(*) AS Grand_Total
FROM airline_data
WHERE Gender IN ('Female', 'Male')
GROUP BY Gender
ORDER BY Gender
```

```
""")
result.show(truncate=False)
result.coalesce(1).write.csv("/user/root/project/demo__1.csv", header=True)
```

Insight 4. Regional Flight Cancellation

```
result = spark.sql("""
SELECT Gender,
    SUM(CASE WHEN Flight_Status = 'Cancelled' THEN 1 ELSE 0 END) AS Cancelled,
    SUM(CASE WHEN Flight_Status = 'Delayed' THEN 1 ELSE 0 END) AS Delayed,
    SUM(CASE WHEN Flight_Status = 'On Time' THEN 1 ELSE 0 END) AS On_Time,
    COUNT(*) AS Grand_Total
FROM airline_data
WHERE Gender IN ('Female', 'Male')
GROUP BY Gender
ORDER BY Gender
""")
result.show(truncate=False)
result.coalesce(1).write.csv("/user/root/project/demo__1.csv", header=True)
```

Insight 5. Monthly Trend

```
from pyspark.sql.functions import col, date_format, to_date, when
```

```
df = spark.read.option("header", "true").option("inferSchema",
"true").csv("/user/root/project/AirlineDatasetUpdatedv2.csv")

# Parse 'Departure Date' as a date type using the MM/DD/YYYY format

df = df.withColumn("Departure Date", to_date(col("Departure Date"), "MM/dd/yyyy"))

df = df.filter(df["Departure Date"].isNotNull())

df = df.withColumn("Month", date_format(col("Departure Date"), "MMM"))

result = df.groupBy("Month").count().withColumnRenamed("count", "Count of Arrival Airport")

result = result.withColumn(

    "Month_Order",

    when(col("Month") == "Jan", 1).when(col("Month") == "Feb", 2)

    .when(col("Month") == "Mar", 3).when(col("Month") == "Apr", 4)

    .when(col("Month") == "May", 5).when(col("Month") == "Jun", 6)

    .when(col("Month") == "Jul", 7).when(col("Month") == "Aug", 8)

    .when(col("Month") == "Sep", 9).when(col("Month") == "Oct", 10)

    .when(col("Month") == "Nov", 11).when(col("Month") == "Dec", 12)

).orderBy("Month_Order").drop("Month_Order")

# Show Final Output

result.show(truncate=False)

# Specify the output path for the CSV file

output_path = "/user/root/project/Monthly_Trend.csv"

result.coalesce(1).write.csv("/user/root/project/Monthly_Trend.csv", header=True,
mode='overwrite')
```

Monthly report with flight status

```
from pyspark.sql.functions import col, date_format, to_date, when, count

df = spark.read.option("header", "true").option("inferSchema",
"true").csv("/user/root/project/AirlineDatasetUpdatedv2.csv")

df = df.withColumn("Departure Date", to_date(col("Departure Date"), "MM/dd/yyyy"))

df = df.filter(df["Departure Date"].isNotNull())

df = df.withColumn("Month", date_format(col("Departure Date"), "MMM"))

result = df.groupBy("Month").pivot("Flight Status", ["Cancelled", "Delayed", "On Time"]).count()

result = result.withColumn(
    "Month_Order",
    when(col("Month") == "Jan", 1).when(col("Month") == "Feb", 2)
    .when(col("Month") == "Mar", 3).when(col("Month") == "Apr", 4)
    .when(col("Month") == "May", 5).when(col("Month") == "Jun", 6)
    .when(col("Month") == "Jul", 7).when(col("Month") == "Aug", 8)
    .when(col("Month") == "Sep", 9).when(col("Month") == "Oct", 10)
    .when(col("Month") == "Nov", 11).when(col("Month") == "Dec", 12)
).orderBy("Month_Order").drop("Month_Order")

result.show(truncate=False)

# Specify the output path for the CSV file

output_path = "/user/root/project/Monthly_Trend_with_Status.csv"

result.coalesce(1).write.csv("/user/root/project/Monthly_Trend_with_Status.csv", header=True,
mode='overwrite')
```

Insight 6. Seasonal Report

```
final_query = """

SELECT

   swc.Season,

   COALESCE(SUM(CASE WHEN swc.`Flight Status` = 'Delayed' THEN swc.FlightCount ELSE 0
END), 0) AS DelayedCount,

   ROUND((COALESCE(SUM(CASE WHEN swc.`Flight Status` = 'Delayed' THEN swc.FlightCount
ELSE 0 END), 0) * 100.0) / st.TotalFlights, 2) AS DelayedPercentage,

   COALESCE(SUM(CASE WHEN swc.`Flight Status` = 'On Time' THEN swc.FlightCount ELSE 0
END), 0) AS OnTimeCount,

   ROUND((COALESCE(SUM(CASE WHEN swc.`Flight Status` = 'On Time' THEN swc.FlightCount
ELSE 0 END), 0) * 100.0) / st.TotalFlights, 2) AS OnTimePercentage,

   COALESCE(SUM(CASE WHEN swc.`Flight Status` = 'Cancelled' THEN swc.FlightCount ELSE 0
END), 0) AS CanceledCount,

   ROUND((COALESCE(SUM(CASE WHEN swc.`Flight Status` = 'Cancelled' THEN swc.FlightCount
ELSE 0 END), 0) * 100.0) / st.TotalFlights, 2) AS CanceledPercentage

FROM (

   SELECT

      CASE

         WHEN MONTH(TO_DATE(`Departure Date`, 'MM/dd/yyyy')) IN (12, 1, 2) THEN 'Winter'

         WHEN MONTH(TO_DATE(`Departure Date`, 'MM/dd/yyyy')) IN (3, 4, 5) THEN 'Spring'

         WHEN MONTH(TO_DATE(`Departure Date`, 'MM/dd/yyyy')) IN (6, 7, 8) THEN 'Summer'

         WHEN MONTH(TO_DATE(`Departure Date`, 'MM/dd/yyyy')) IN (9, 10, 11) THEN 'Fall'
```

```
        END AS Season,
        `Flight Status`,
        COUNT(*) AS FlightCount
    FROM airlinedata
    GROUP BY
        CASE
            WHEN MONTH(TO_DATE(`Departure Date`, 'MM/dd/yyyy')) IN (12, 1, 2) THEN 'Winter'
            WHEN MONTH(TO_DATE(`Departure Date`, 'MM/dd/yyyy')) IN (3, 4, 5) THEN 'Spring'
            WHEN MONTH(TO_DATE(`Departure Date`, 'MM/dd/yyyy')) IN (6, 7, 8) THEN 'Summer'
            WHEN MONTH(TO_DATE(`Departure Date`, 'MM/dd/yyyy')) IN (9, 10, 11) THEN 'Fall'
        END,
        `Flight Status`
) swc
JOIN (
    SELECT
        CASE
            WHEN MONTH(TO_DATE(`Departure Date`, 'MM/dd/yyyy')) IN (12, 1, 2) THEN 'Winter'
            WHEN MONTH(TO_DATE(`Departure Date`, 'MM/dd/yyyy')) IN (3, 4, 5) THEN 'Spring'
            WHEN MONTH(TO_DATE(`Departure Date`, 'MM/dd/yyyy')) IN (6, 7, 8) THEN 'Summer'
            WHEN MONTH(TO_DATE(`Departure Date`, 'MM/dd/yyyy')) IN (9, 10, 11) THEN 'Fall'
        END AS Season,
        COUNT(*) AS TotalFlights
```

```
    FROM airlinedata
    GROUP BY
        CASE
            WHEN MONTH(TO_DATE(`Departure Date`, 'MM/dd/yyyy')) IN (12, 1, 2) THEN 'Winter'
            WHEN MONTH(TO_DATE(`Departure Date`, 'MM/dd/yyyy')) IN (3, 4, 5) THEN 'Spring'
            WHEN MONTH(TO_DATE(`Departure Date`, 'MM/dd/yyyy')) IN (6, 7, 8) THEN 'Summer'
            WHEN MONTH(TO_DATE(`Departure Date`, 'MM/dd/yyyy')) IN (9, 10, 11) THEN 'Fall'
        END
) st
ON swc.Season = st.Season
GROUP BY swc.Season, st.TotalFlights
ORDER BY swc.Season
"""
Display the results
final_result = spark.sql(final_query)
final_result.show()
```