(https://databricks.com)

Data Prepration

Includes : 1. Data Selection (Subset) 2. Data Cleaning & Tranformation

-Dataset: dl_carpark_merged

-Libraries: Spark

Summary : Below conditions was considered for Data Prepration:

-carpark_type_name = "Prebook & Roll up"

-booking_detail_booking_status = "C" (only "Confirmed" Booking status) and

-booking_detail_source_gross_amount >0 and

-booking_detail_staydays >= 0

Summary Output : Data Prepration

Variables:

-Total columns/variables: 1060

-Variables selected: 24 (Final)

Records:

-Total records : ~40M

-ST Long Data records (2012-2023): ~28M

-ST Long Data 2021 records : ~500K

-ST Long Data 2021 with above conditions records : ~470K (Final)

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col
from pyspark.sql.types import DateType
from pyspark.sql.functions import to_date
from pyspark.sql.functions import col, to_date
from pyspark.sql.utils import AnalysisException

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


database_name = "airline_db"
table_name = "dl_carpark_merged"


# Read data from the table
data = spark.read.table(f"{database_name}.{table_name}")


data.count()
```

```
Out[91]: 39287704
```

```
# Get the number of rows
num_rows = data.count()

# Get the number of columns
num_columns = len(data.columns)

# Print the dimensions
print("Number of Rows:", num_rows)
print("Number of Columns:", num_columns)
```

```
Number of Rows: 39287704
Number of Columns: 1060
```

1. Data Selection (Subset)

```
# Select 27 variables
selected_variables = ["car_park_prebook_description",
"carpark_type_name",
"booking_summary_source_booking_num",
"booking_date_date_value",
"booked_entry_date_date_value",
"booked_exit_date_date_value",
"actual_entry_date_date_value",
"actual_exit_date_date_value",
"booking_date_calendar_year_week",
"booked_entry_date_calendar_year_week",
"booked_exit_date_calendar_year_week",
"booking_date_calendar_by_day_month",
"booked_entry_date_calendar_by_day_month",
"booked_exit_date_calendar_by_day_month",
"booking_date_phase",
"booked_entry_date_phase",
"booked_exit_date_phase",
"booking_detail_staydays",
"channel_long_description",
"actual_exit_date_week_day_name",
"actual_entry_date_week_day_name",
"car_park_carpark_capacity",
"booking_detail_source_gross_amount",
"booking_detail_booking_status",
"booking_detail_lead_time",
"booking_summary_inbound_flight",
"booking_summary_outbound_flight"
]


# Subset the data
cols_subset_data = data.select(*selected_variables)


# Subset Data on 24 Columns
num_rows = cols_subset_data.count()

# Get the number of columns
num_columns = len(cols_subset_data.columns)

# Print the dimensions
print("Number of Rows:", num_rows)
print("Number of Columns:", num_columns)

Number of Rows: 39287704
Number of Columns: 27
```

```
# Filter the data for rows where "car_park_prebook_description" is "ST Long Stay"
filtered_data = cols_subset_data.filter(cols_subset_data["car_park_prebook_description"] == "ST Long Stay")


# Filtered Data on ST Long Stay
num_rows = filtered_data.count()

# Get the number of columns
num_columns = len(filtered_data.columns)

# Print the dimensions
print("Number of Rows:", num_rows)
print("Number of Columns:", num_columns)
```

```
Number of Rows: 5018799
Number of Columns: 27
```

```
filtered_data.dtypes
```

```
Out[98]: [('car_park_prebook_description', 'string'),
 ('carpark_type_name', 'string'),
 ('booking_summary_source_booking_num', 'string'),
 ('booking_date_date_value', 'date'),
 ('booked_entry_date_date_value', 'date'),
 ('booked_exit_date_date_value', 'date'),
 ('actual_entry_date_date_value', 'date'),
 ('actual_exit_date_date_value', 'date'),
 ('booking_date_calendar_year_week', 'int'),
 ('booked_entry_date_calendar_year_week', 'int'),
 ('booked_exit_date_calendar_year_week', 'int'),
 ('booking_date_calendar_by_day_month', 'int'),
 ('booked_entry_date_calendar_by_day_month', 'int'),
 ('booked_exit_date_calendar_by_day_month', 'int'),
 ('booking_date_phase', 'string'),
 ('booked_entry_date_phase', 'string'),
 ('booked_exit_date_phase', 'string'),
 ('booking_detail_staydays', 'bigint'),
 ('channel_long_description', 'string'),
 ('actual_exit_date_week_day_name', 'string'),
 ('actual_entry_date_week_day_name', 'string'),
```

```
# Filter the data where Car Park is ST Long Stay and Booking Id is not NULL
filtered_data_NEW = cols_subset_data.filter(
    (cols_subset_data["car_park_prebook_description"] == "ST Long Stay") &
    (col("booking_summary_source_booking_num").isNotNull())
)
```

```
# Filtered Data on ST Long Stay
num_rows = filtered_data_NEW.count()

# Get the number of columns
num_columns = len(filtered_data_NEW.columns)

# Print the dimensions
print("Number of Rows:", num_rows)
print("Number of Columns:", num_columns)
```

```
Number of Rows: 2881359
Number of Columns: 27
```

```
df = filtered_data_NEW.withColumn("booking_date_date_value", to_date(col("booking_date_date_value"), "yyyy-MM-dd"))
```

Changed the Year from 2023 to 2021 but the Dataframe name (df_2023) remains the same

```
df_2023= df.filter(col("booking_date_date_value")>="2021-01-01")
```

```
# Filtered Data on 2023
num_rows = df_2023.count()

# Get the number of columns
num_columns = len(df_2023.columns)

# Print the dimensions
print("Number of Rows:", num_rows)
print("Number of Columns:", num_columns)
```

```
Number of Rows: 507084
Number of Columns: 27
```

```
display(df_2023)
```

Table

	car_park_prebook_description	carpark_type_name	booking_summary_source_booking_num	booking_date_date_value	booked_entry_date_date_value	booked_exit_date_date_value	actual_e
1	ST Long Stay	Prebook	DW05161706OLI	2023-09-03	2023-09-29	2023-10-06	null
2	ST Long Stay	Prebook	DWC03787042PHO	2023-03-31	2023-03-31	2023-04-07	null
3	ST Long Stay	Prebook	DWC03787042PHO	2023-03-31	2023-03-31	2023-04-07	null
4	ST Long Stay	Prebook	DW03906821GAR	2023-04-14	2023-10-10	2023-10-14	null
5	ST Long Stay	Prebook	DW03472388BEV	2023-02-22	2023-02-24	2023-03-07	null
6	ST Long Stay	Prebook	DW02618240ABI	2022-10-11	2022-12-22	2022-12-30	null

7	ST Long Stay	Prebook	DW02646710OLA	2022-10-14	2022-10-23	2022-10-28	null

7,948 rows | Truncated data

```
# Filter the df_2023 data further : "ST Long Stay" Data from Jan 2023 >> Prebook with Confirmed Booking Status, Gross Amount >0 and Stay Days >= 0
df_2023_filter = df_2023.filter(
    (df_2023["booking_summary_source_booking_num"].isNotNull()) &
    (col("booking_detail_booking_status") == "C") &
    (col("booking_detail_source_gross_amount") > 0) &
    (col("booking_detail_staydays") >= 0)
)


# Filtered Data on 2023
num_rows = df_2023_filter.count()

# Get the number of columns
num_columns = len(df_2023_filter.columns)

# Print the dimensions
print("Number of Rows:", num_rows)
print("Number of Columns:", num_columns)

Number of Rows: 467660
Number of Columns: 27


display(df_2023_filter)
```

Table

	car_park_prebook_description	carpark_type_name	booking_summary_source_booking_num	booking_date_date_value	booked_entry_date_date_value	booked_exit_date_date_value	actual_e
1	ST Long Stay	Prebook	DW05161706OLI	2023-09-03	2023-09-29	2023-10-06	null
2	ST Long Stay	Prebook	DWC03787042PHO	2023-03-31	2023-03-31	2023-04-07	null
3	ST Long Stay	Prebook	DWC03787042PHO	2023-03-31	2023-03-31	2023-04-07	null
4	ST Long Stay	Prebook	DW03906821GAR	2023-04-14	2023-10-10	2023-10-14	null
5	ST Long Stay	Prebook	DW03472388BEV	2023-02-22	2023-02-24	2023-03-07	null
6	ST Long Stay	Prebook	DW02646710OLA	2022-10-14	2022-10-23	2022-10-28	null
7	ST Long Stay	Prebook	PC03252141OJZ	2023-01-25	2023-02-19	2023-02-25	null

7,926 rows | Truncated data

2. Data Cleaning & Transformation

Summary : below actitivites were performed for Data Cleaning & Transformations:

-Replace values in "booking_summary_outbound_flight" with flight no.

-Replace NULL values in Entry and Exit Dates

-Derive weekday name

```
#Replace values in "booking_summary_outbound_flight" with flight no. Eg. "<font style=""vertical-align: inherit;""><font style=""vertical-align: inherit;"">EJU8097</font></font>" replace with EJU8097

#Additional Cleaning: with new dataset from 2021 - 4 new similar records found (date: 02112023)
```

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

df_2023_filter = df_2023_filter.withColumn("booking_summary_outbound_flight",
                                           when(df_2023_filter["booking_summary_outbound_flight"].like('%<font style=""vertical-align: inherit;""><font style=""vertical-align:
inherit;"">VY7821</font></font>%'), 'VY7821')
                                           .otherwise(df_2023_filter["booking_summary_outbound_flight"]))

df_2023_filter = df_2023_filter.withColumn("booking_summary_outbound_flight",
                                           when(df_2023_filter["booking_summary_outbound_flight"].like('%<font style=""vertical-align: inherit;""><font style=""vertical-align:
inherit;"">W95747</font></font>%'), 'W95747')
                                           .otherwise(df_2023_filter["booking_summary_outbound_flight"]))

df_2023_filter = df_2023_filter.withColumn("booking_summary_outbound_flight",
                                           when(df_2023_filter["booking_summary_outbound_flight"].like('%<font style=""vertical-align: inherit;""><font style=""vertical-align:
inherit;"">VY8945</font></font>%'), 'VY8945')
                                           .otherwise(df_2023_filter["booking_summary_outbound_flight"]))

df_2023_filter = df_2023_filter.withColumn("booking_summary_outbound_flight",
                                           when(df_2023_filter["booking_summary_outbound_flight"].like('%<font style=""vertical-align: inherit;""><font style=""vertical-align:
inherit;"">EJU8097</font></font>%'), 'EJU8097')
                                           .otherwise(df_2023_filter["booking_summary_outbound_flight"]))

df_2023_filter = df_2023_filter.withColumn("booking_summary_outbound_flight",
                                           when(df_2023_filter["booking_summary_outbound_flight"].like('%<font style=""vertical-align: inherit;""><font style=""vertical-align:
inherit;"">W45703</font></font>%'), 'W45703')
                                           .otherwise(df_2023_filter["booking_summary_outbound_flight"]))

df_2023_filter = df_2023_filter.withColumn("booking_summary_outbound_flight",
                                           when(df_2023_filter["booking_summary_outbound_flight"].like('%<font style=""vertical-align: inherit;""><font style=""vertical-align:
inherit;"">VY6227</font></font>%'), 'VY6227')
                                           .otherwise(df_2023_filter["booking_summary_outbound_flight"]))

df_2023_filter = df_2023_filter.withColumn("booking_summary_outbound_flight",
                                           when(df_2023_filter["booking_summary_outbound_flight"].like('% XQ591%'), 'XQ591')
                                           .otherwise(df_2023_filter["booking_summary_outbound_flight"]))

df_2023_filter = df_2023_filter.withColumn("booking_summary_outbound_flight",
                                           when(df_2023_filter["booking_summary_outbound_flight"].like('%<font style=""vertical-align: inherit;""><font style=""vertical-align:
inherit;"">EZY8961</font></font>%'), 'EZY8961')
                                           .otherwise(df_2023_filter["booking_summary_outbound_flight"]))

df_2023_filter = df_2023_filter.withColumn("booking_summary_outbound_flight",
                                           when(df_2023_filter["booking_summary_outbound_flight"].like('%<font style=""vertical-align: inherit;""><font style=""vertical-align:
inherit;"">W95713</font></font>%'), 'W95713')
                                           .otherwise(df_2023_filter["booking_summary_outbound_flight"]))

df_2023_filter = df_2023_filter.withColumn("booking_summary_outbound_flight",
                                           when(df_2023_filter["booking_summary_outbound_flight"].like('%<font style=""vertical-align: inherit;""><font style=""vertical-align:
inherit;"">EZY8913</font></font>%'), 'EZY8913')
```

```
                                        .otherwise(df_2023_filter["booking_summary_outbound_flight"]))

df_2023_filter = df_2023_filter.withColumn("booking_summary_outbound_flight",
                                        when(df_2023_filter["booking_summary_outbound_flight"].like('%"<recite class=""recite-ele recite-ele-style"" style=""display:
inline;"">EI911</recite>"%'), 'EI911')
                                        .otherwise(df_2023_filter["booking_summary_outbound_flight"]))


#Replace the actual_entry_date_date_value if NULL to the date given in "booked_entry_date_date_value"
#Replace the actual_exit_date_date_value if NULL to the date given in "booked_exit_date_date_value"


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

# Replace NULL values in "actual_entry_date_date_value" with values from "booked_entry_date_date_value"
df_2023_filter = df_2023_filter.withColumn("actual_entry_date_date_value",
                                        when(df_2023_filter["actual_entry_date_date_value"].isNull(), df_2023_filter["booked_entry_date_date_value"])
                                        .otherwise(df_2023_filter["actual_entry_date_date_value"]))

# Replace NULL values in "actual_exit_date_date_value" with values from "booked_exit_date_date_value"
df_2023_filter = df_2023_filter.withColumn("actual_exit_date_date_value",
                                        when(df_2023_filter["actual_exit_date_date_value"].isNull(), df_2023_filter["booked_exit_date_date_value"])
                                        .otherwise(df_2023_filter["actual_exit_date_date_value"]))


#Derive weekday name


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

# Derive "actual_entry_date_week_day_name" based on "actual_entry_date_date_value" where value is null
df_2023_filter = df_2023_filter.withColumn("actual_entry_date_week_day_name",
                                        when(col("actual_entry_date_week_day_name").isNull(),
                                             date_format(col("actual_entry_date_date_value"), "EEEE"))
                                        .otherwise(col("actual_entry_date_week_day_name")))

# Derive "actual_exit_date_week_day_name" based on "actual_exit_date_date_value" where value is null
df_2023_filter = df_2023_filter.withColumn("actual_exit_date_week_day_name",
                                        when(col("actual_exit_date_week_day_name").isNull(),
                                             date_format(col("actual_exit_date_date_value"), "EEEE"))
                                        .otherwise(col("actual_exit_date_week_day_name")))

# Show the DataFrame with the derived weekday names
display(df_2023_filter)
```

Table

	car_park_prebook_description	carpark_type_name	booking_summary_source_booking_num	booking_date_date_value	booked_entry_date_date_value	booked_exit_date_date_value	actual_e
1	ST Long Stay	Prebook	DW05161706OLI	2023-09-03	2023-09-29	2023-10-06	2023-09
2	ST Long Stay	Prebook	DWC03787042PHO	2023-03-31	2023-03-31	2023-04-07	2023-03

3	ST Long Stay	Prebook	DWC03787042PHO	2023-03-31	2023-03-31	2023-04-07	2023-03
4	ST Long Stay	Prebook	DW03906821GAR	2023-04-14	2023-10-10	2023-10-14	2023-10
5	ST Long Stay	Prebook	DW03472388BEV	2023-02-22	2023-02-24	2023-03-07	2023-02
6	ST Long Stay	Prebook	DW02646710OLA	2022-10-14	2022-10-23	2022-10-28	2022-10
7	ST Long Stay	Prebook	PC03252141OJZ	2023-01-25	2023-02-19	2023-02-25	2023-02

7,687 rows | Truncated data

Save the Data Outputs in Parquet & CSV files in FileStore.

Note: the Dataframe output name "df_2023_filter" is changed to "df_2021_filter"

```
df_2021_filter = df_2023_filter


# Define the output path in the DBFS file store
output_path = "dbfs:/FileStore/df_2021_filter.parquet"

# Write the DataFrame to the DBFS in Parquet format
df_2021_filter.write.parquet(output_path, mode="overwrite")

# Provide a confirmation message
print(f"DataFrame 'df_2021_filter' has been written to '{output_path}'.")
```

```
DataFrame 'df_2021_filter' has been written to 'dbfs:/FileStore/df_2021_filter.parquet'.

DataFrame 'df_2021_filter' has been written to 'dbfs:/FileStore/df_2021_filter.csv'.
```