Lab4-Task3: Detect 30% Delay Change by Carrier.

Description

The code reads flight data and calculates the total delay (sum of departure and arrival delays) for each carrier within a sliding window of 10 days. It then compares the total delay between adjacent windows for each carrier to identify if there's a 30% change in delay times.

Techniques Covered:

- PySpark SQL
- Window Functions
- Aggregation
- Conditional Filtering

Overview

1. Initialize a SparkSession.
2. Define window specifications.
3. Read flight data.
4. Perform GroupBy and Aggregation operations.
5. Apply window functions to calculate the change in total delays between adjacent windows.
6. Filter the results based on 30% change condition.

Spark Environment Setup: Import Libraries:

```
from pyspark.sql import SparkSession
from pyspark.sql import functions as F
from pyspark.sql import Window
```

Initialize Spark Session: Establish a local Spark session utilizing all available cores with 4GB memory allocation.

```
spark = SparkSession \
    .builder \
    .master("local") \
    .config("spark.driver.memory", "4g") \
    .appName('ex4_anomalies_detection') \
    .getOrCreate()
```

Define Window Specification: A sliding window is defined that partitions data by **Carrier** and orders by the **start_range** of the windowed flight data.

```
sliding_range_window = Window.partitionBy(F.col('Carrier')).orderBy(F.col('start_range'))
```

Load Data: Fetch the flight data from the specified S3 path and cache it for enhanced performance during subsequent operations.

```
flights_df = spark.read.parquet('s3a://spark/data/transformed/flights/')

flights_df.cache()
```

Group Data with Sliding Window: The data is grouped by Carrier and a sliding window of 10 days with a 1-day step. The resulting total delay (sum of departure and arrival delays) for each window is calculated.

```
grouped_df = flights_df \
    .groupBy(F.col('Carrier'), F.window(F.col('flight_date'), '10 days', '1 day').alias('date_window')) \
    .agg(F.sum(F.col('dep_delay') + F.col('arr_delay')).alias('total_delay'))
```

Select Relevant Columns: The results are structured to contain the Carrier, the start and end of the window, and the computed total_delay.

```
structured_df = grouped_df \
    .select(F.col('Carrier'),
            F.col('date_window.start').alias('start_range'),
            F.col('date_window.end').alias('end_range'),
            F.col('total_delay'))
```

Calculate Percentage Change: For each row, find the delay from the previous window and calculate the percentage change.

```
change_df = structured_df \
    .withColumn('last_window_delay', F.lag(F.col('total_delay')).over(sliding_range_window)) \
    .withColumn('change_percent', F.abs(F.lit(1.0) - (F.col('total_delay') /
F.col('last_window_delay'))))
```

Filter Significant Changes: Retain only the records where there's more than a 30% change between consecutive windows.

```
significant_changes_df = change_df.where(F.col('change_percent') > F.lit(0.3))
significant_changes_df.show(100)
```

The data representing carriers and the date ranges with a pronounced change in delays is showcased.

Clean Up: Cached data is released, and the Spark session is terminated.

```
flights_df.unpersist()
spark.stop()
```