Order Tracking Incremental Load

1. this project was about creating two different spark jobs in databricks. Once spark jobs were ready, I put them into a workflow job and scheduled that workflow. Basically on the execution of this scheduled job, we were moving data from staging table and doing upsert operation in target table.

2. We received daily csv file in gcs bucket input dir & from there we read data.

	A	B	C	D	E	F	G
1	order_num	tracking_num	pck_recieved_date	package_deliver_date	status	address	last_update_timestamp
2	1001	TRK1001	03-01-2023	06-01-2023	In Transit	123 Elm St	03-01-2023 16:09
3	1002	TRK1002	03-01-2023	06-01-2023	Delayed	789 Pine St	03-01-2023 07:36
4	1003	TRK1003	03-01-2023	05-01-2023	Delivered	123 Elm St	03-01-2023 10:09
5	1004	TRK1004	03-01-2023	05-01-2023	Returned	202 Birch Rd	03-01-2023 07:55
6	1005	TRK1005	03-01-2023	05-01-2023	Delivered	101 Maple Ave	03-01-2023 09:20
7	1006	TRK1006	03-01-2023	06-01-2023	Out for Delivery	202 Birch Rd	03-01-2023 17:01
8							

3. For 'stage_logistic_tracking_job' spark script we had gcs bucket input storage and an archieve storage. Also we had a dbfs staging table path. Next step was to writing data in delta format at staging path if not exists or else just overwriting it.

```
# Path to the service account JSON key in DBFS
service_account_path = "/dbfs/FileStore/tables/databricks_projects_419308_eb59c24c45fe.json"

# Configure Spark to use the service account JSON key for GCS authentication
spark.conf.set("fs.gs.auth.service.account.json.keyfile", service_account_path)


# GCS bucket details
bucket_name = "order_tracking_sb"
data_directory = f"gs://{bucket_name}/input/"
archive_directory = f"gs://{bucket_name}/archive/"

# Define the path for the staging Delta table
staging_table_path = "dbfs:/tmp/staging_order_tracking"

# Read all CSV files from the specified GCS directory
df = spark.read.csv(data_directory, inferSchema=True, header=True)

df.show()

print("Create Stage Table")
# Check if the Delta table exists
if DeltaTable.isDeltaTable(spark, staging_table_path):
    # If table exists, overwrite it
    df.write.format("delta").mode("overwrite").save(staging_table_path)
else:
    # If not, create the table
    df.write.format("delta").mode("append").save(staging_table_path)

print("Data Inserted In Stage Table")
```

4. This way data were inserted at staging path. Further, created delta table on the top of that delta location. At last, we were moving input source file to the gcs archieve folder once the staging process is done.

```
# Create a Delta Lake table using the staging table path
spark.sql(f"CREATE DATABASE IF NOT EXISTS hive_metastore.staging")
spark.sql(f"CREATE TABLE IF NOT EXISTS hive_metastore.staging.staging_order_tracking USING DELTA LOCATION '{staging_table_path}'")

# List and move files individually
file_list = dbutils.fs.ls(data_directory)
for file in file_list:
    if file.name.endswith(".csv"):
        print(f"{file} Moved in archive folder")
        dbutils.fs.mv(file.path, os.path.join(archive_directory, file.name))
```

```
+---------+------------+----------------+-------------------+---------------+------------+--------------------+
|order_num|tracking_num|pck_recieved_date|package_deliver_date|         status|     address|last_update_timestamp|
+---------+------------+----------------+-------------------+---------------+------------+--------------------+
|     1000|     TRK1000|      2023-01-01|         2023-01-06|       Returned|  456 Oak St|  2023-01-01 05:41:55|
|     1001|     TRK1001|      2023-01-01|         2023-01-06|     In Transit|  789 Pine St|  2023-01-01 08:43:50|
|     1002|     TRK1002|      2023-01-01|         2023-01-05|Out for Delivery|  123 Elm St|  2023-01-01 17:44:29|
|     1003|     TRK1003|      2023-01-01|         2023-01-04|     In Transit|  789 Pine St|  2023-01-01 05:29:16|
|     1004|     TRK1004|      2023-01-01|         2023-01-02|        Delayed|202 Birch Rd|  2023-01-01 13:12:49|
|     1005|     TRK1005|      2023-01-01|         2023-01-03|      Delivered|101 Maple Ave|  2023-01-01 23:42:57|
|     1006|     TRK1006|      2023-01-01|         2023-01-02|       Returned|101 Maple Ave|  2023-01-01 09:37:34|
|     1007|     TRK1007|      2023-01-01|         2023-01-02|      Delivered|  456 Oak St|  2023-01-01 22:46:09|
|     1008|     TRK1008|      2023-01-01|         2023-01-03|     In Transit|  123 Elm St|  2023-01-01 16:14:44|
|     1009|     TRK1009|      2023-01-01|         2023-01-05|        Delayed|202 Birch Rd|  2023-01-01 00:42:25|
|     1010|     TRK1010|      2023-01-01|         2023-01-04|     In Transit|202 Birch Rd|  2023-01-01 05:07:32|
```

5. For 'target_logistic_tracking_upsert_job' spark script, we had paths for staging and target delta tables in dbfs. We were next reading data from staging delta file location. Then we were writing staging dataframe at target table path if that not existed and created delta table object from their target path. Further performed upsert from staging to target table using tracking_num as key. At last, created delta table on the top of that target location.

```
# Initialize Spark Session
spark = SparkSession.builder.appName("Upsert into Target Delta Table").getOrCreate()

# Paths for the staging and target Delta tables
staging_table_path = "dbfs:/tmp/staging_order_tracking"
target_table_path = "dbfs:/tmp/target_order_tracking"

# Read from the staging Delta table
staging_df = spark.read.format("delta").load(staging_table_path)
staging_df.show()
print("Data read from staging table completed")

# Check if the target Delta table exists, create it if not
if not DeltaTable.isDeltaTable(spark, target_table_path):
    staging_df.write.format("delta").save(target_table_path)

# Create DeltaTable object for the target table
target_delta_table = DeltaTable.forPath(spark, target_table_path)

# Perform upsert from staging to target table using tracking_num as key
target_delta_table.alias("target").merge(
    staging_df.alias("staging"),
    "target.tracking_num = staging.tracking_num"
).whenMatchedUpdateAll().whenNotMatchedInsertAll().execute()

print("Data upserted in target table")

# Register the target table in the Hive Metastore (Optional)
spark.sql(f"CREATE DATABASE IF NOT EXISTS hive_metastore.target")
spark.sql(f"CREATE TABLE IF NOT EXISTS hive_metastore.target.target_order_tracking USING DELTA LOCATION '{target_table_path}'")
```

```
+---------+------------+----------------+-------------------+---------------+------------+--------------------+
|order_num|tracking_num|pck_recieved_date|package_deliver_date|         status|     address|last_update_timestamp|
+---------+------------+----------------+-------------------+---------------+------------+--------------------+
|     1000|     TRK1000|      2023-01-01|         2023-01-06|       Returned|  456 Oak St|  2023-01-01 05:41:55|
|     1001|     TRK1001|      2023-01-01|         2023-01-06|     In Transit|  789 Pine St|  2023-01-01 08:43:50|
|     1002|     TRK1002|      2023-01-01|         2023-01-05|Out for Delivery|  123 Elm St|  2023-01-01 17:44:29|
|     1003|     TRK1003|      2023-01-01|         2023-01-04|     In Transit|  789 Pine St|  2023-01-01 05:29:16|
|     1004|     TRK1004|      2023-01-01|         2023-01-02|        Delayed|202 Birch Rd|  2023-01-01 13:12:49|
|     1005|     TRK1005|      2023-01-01|         2023-01-03|      Delivered|101 Maple Ave|  2023-01-01 23:42:57|
```