databricks (https://databricks.com)

Airline_Safety_ETL_Pipeline_Analysis.ipynb

```
!pip install -r ../requirements.txt
```

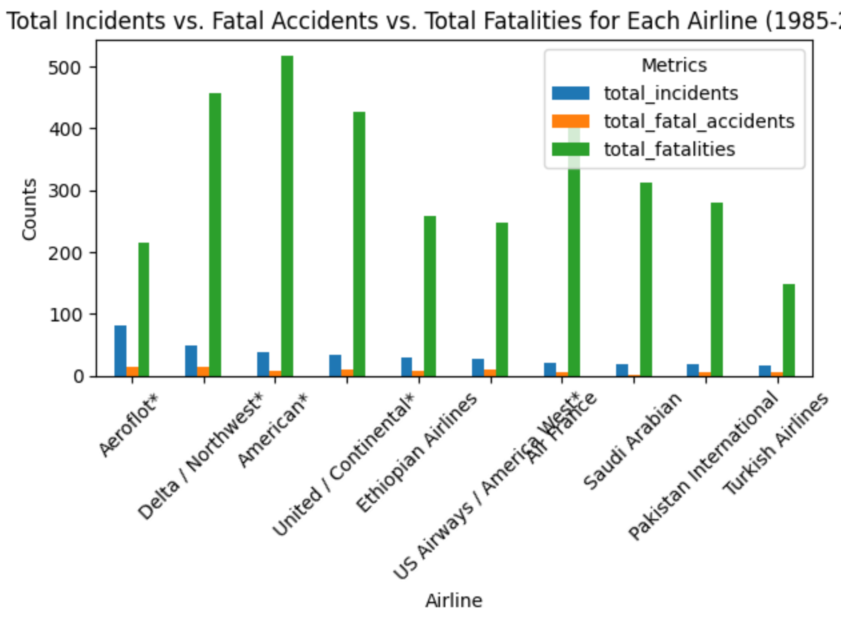
```
Collecting black==22.3.0
  Downloading black-22.3.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.5 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.5/1.5 MB 18.6 MB/s eta 0:00:0000:010:01
Collecting click==8.1.3
  Downloading click-8.1.3-py3-none-any.whl (96 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 96.6/96.6 kB 9.4 MB/s eta 0:00:00
Collecting pytest==7.1.3
  Downloading pytest-7.1.3-py3-none-any.whl (298 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 298.2/298.2 kB 21.1 MB/s eta 0:00:00
Collecting pytest-cov==4.0.0
  Downloading pytest_cov-4.0.0-py3-none-any.whl (21 kB)
Collecting ruff==0.0.284
  Downloading ruff-0.0.284-py3-none-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (5.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5.7/5.7 MB 72.4 MB/s eta 0:00:00:00:01
Collecting boto3==1.24.87
  Downloading boto3-1.24.87-py3-none-any.whl (132 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 132.5/132.5 kB 5.3 MB/s eta 0:00:00
Collecting fastapi==0.85.0
  Downloading fastapi-0.85.0-py3-none-any.whl (55 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 55.3/55.3 kB 6.9 MB/s eta 0:00:00
Collecting uvicorn==0.18.3
```

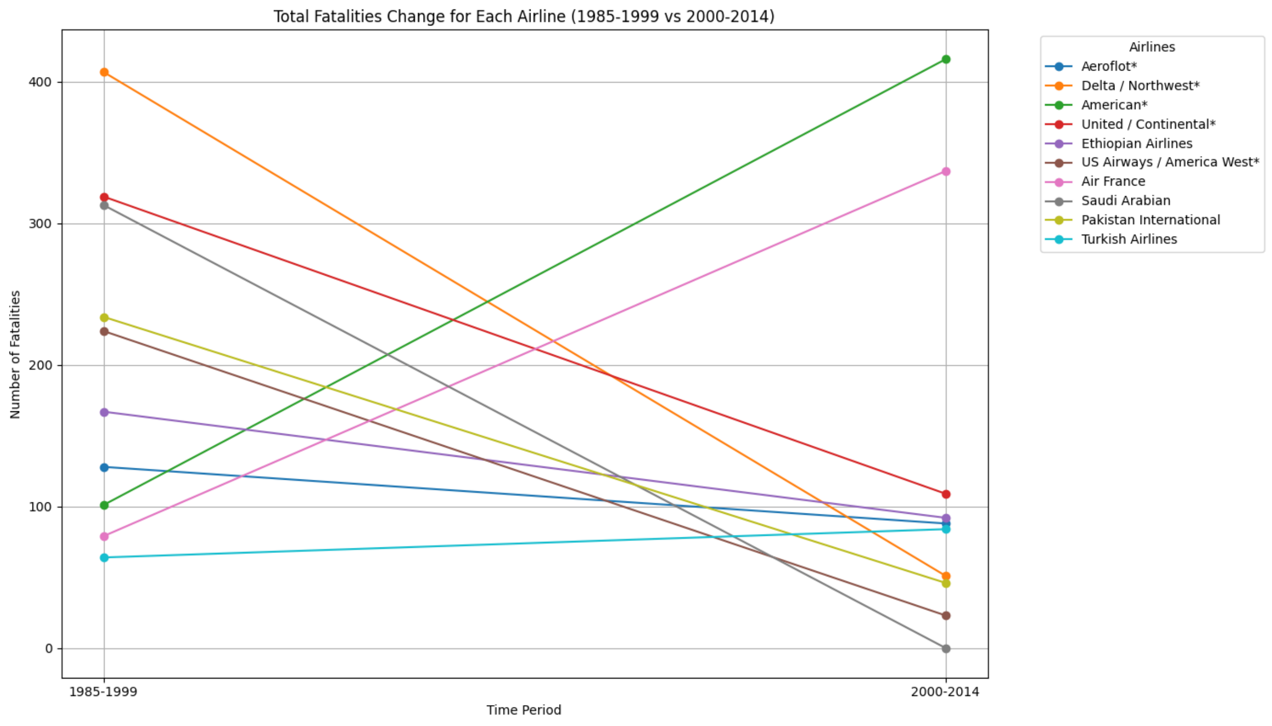
```
import requests
from dotenv import load_dotenv
import os
import json
import base64

# Load environment variables
load_dotenv()
server_h = os.getenv("SERVER_HOSTNAME")
access_token = os.getenv("ACCESS_TOKEN")
FILESTORE_PATH = "dbfs:/FileStore/tinayiluo_Databricks_ETL_Pipeline"
headers = {'Authorization': 'Bearer %s' % access_token}
url = "https://"+server_h+"/api/2.0"


def perform_query(path, headers, data={}):
    session = requests.Session()
    resp = session.request('POST', url + path,
                           data=json.dumps(data),
                           verify=True,
                           headers=headers)
    return resp.json()


def mkdirs(path, headers):
    _data = {}
    _data['path'] = path
    return perform_query('/dbfs/mkdirs', headers=headers, data=_data)


def create(path, overwrite, headers):
    _data = {}
    _data['path'] = path
    _data['overwrite'] = overwrite
    return perform_query('/dbfs/create', headers=headers, data=_data)


def add_block(handle, data, headers):
    _data = {}
    _data['handle'] = handle
    _data['data'] = data
    return perform_query('/dbfs/add-block', headers=headers, data=_data)


def close(handle, headers):
    _data = {}
    _data['handle'] = handle
    return perform_query('/dbfs/close', headers=headers, data=_data)


def put_file_from_url(url, dbfs_path, overwrite, headers):
    response = requests.get(url)
    if response.status_code == 200:
        content = response.content
        handle = create(dbfs_path, overwrite, headers=headers)['handle']
        print("Putting file: " + dbfs_path)
        for i in range(0, len(content), 2**20):
            add_block(handle,
                      base64.standard_b64encode(content[i:i+2**20]).decode(),
                      headers=headers)
        close(handle, headers=headers)
        print(f"File {dbfs_path} uploaded successfully.")
    else:
        print(f"Error downloading file from {url}. Status code: {response.status_code}")


def extract(
        url="""https://github.com/fivethirtyeight/data/blob/master/airline-safety/airline-safety.csv?raw=true""",
        file_path=FILESTORE_PATH+"/airline-safety.csv",
        directory=FILESTORE_PATH,
        overwrite=True
):
    """Extract a url to a file path"""
    # Make the directory, no need to check if it exists or not
    mkdirs(path=directory, headers=headers)
    # Add the csv files, no need to check if it exists or not
    put_file_from_url(url, file_path, overwrite, headers=headers)

    return file_path



# Databricks notebook source
"""
transform and load function
"""

from pyspark.sql import SparkSession
from pyspark.sql.functions import monotonically_increasing_id

def load(dataset="dbfs:/FileStore/tinayiluo_Databricks_ETL_Pipeline/airline-safety.csv"):
    spark = SparkSession.builder.appName("Read CSV").getOrCreate()
    # load csv and transform it by inferring schema
    airline_safety_df = spark.read.csv(dataset, header=True, inferSchema=True)

    columns = airline_safety_df.columns

    # Calculate mid index
    mid_idx = len(columns) // 2

    # Split columns into two halves
    columns1 = columns[:mid_idx]
    columns2 = columns[mid_idx:]

    # Create two new DataFrames
    airline_safety_df1 = airline_safety_df.select(*columns1)
    airline_safety_df2 = airline_safety_df.select(*columns2)

    # add unique IDs to the DataFrames
    airline_safety_df1 = airline_safety_df1.withColumn("id", monotonically_increasing_id())
    airline_safety_df2 = airline_safety_df2.withColumn("id", monotonically_increasing_id())

    # transform into a delta lakes table and store it
    airline_safety_df1.write.format("delta").mode("overwrite").saveAsTable("airline_safety1_delta")
    airline_safety_df2.write.format("delta").mode("overwrite").saveAsTable("airline_safety2_delta")

    num_rows = airline_safety_df1.count()
    print(num_rows)
    num_rows = airline_safety_df2.count()
    print(num_rows)

    return "finished transform and load"
```

```
# Databricks notebook source
from pyspark.sql import SparkSession
import matplotlib.pyplot as plt


# sample query
def query_transform():
    """
    Run a predefined SQL query on a Spark DataFrame.

    Returns:
        DataFrame: Result of the SQL query.
    """
    spark = SparkSession.builder.appName("Query").getOrCreate()
    query = (
        "SELECT "
        "a.airline, "
        "a.incidents_85_99, "
        "b.incidents_00_14, "
        "a.fatal_accidents_85_99, "
        "b.fatal_accidents_00_14, "
        "b.fatalities_85_99, "
        "b.fatalities_00_14, "
        "(a.incidents_85_99 + b.incidents_00_14) AS total_incidents, "
        "a.fatal_accidents_85_99 + b.fatal_accidents_00_14 "
        "AS total_fatal_accidents, "
        "b.fatalities_85_99 + b.fatalities_00_14 "
        "AS total_fatalities "
        "FROM "
        "airline_safety1_delta AS a "
        "JOIN "
        "airline_safety2_delta AS b "
        "ON a.id = b.id "
        "ORDER BY total_incidents DESC "
        "LIMIT 10"
    )


    query_result = spark.sql(query)
    return query_result


# sample viz for project
def viz():
    query = query_transform()
    count = query.count()
    if count > 0:
        print(f"Data validation passed. {count} rows available.")
    else:
        print("No data available. Please investigate.")

    # Convert the query_result DataFrame to Pandas for plotting
    query_result_pd = query.toPandas()

    # Bar Plot
    plt.figure(figsize=(15, 7))
    query_result_pd.plot(x='airline', y=['total_incidents', 'total_fatal_accidents',
                                         'total_fatalities'], kind='bar')
    plot_title = ('Total Incidents vs. Fatal Accidents vs. '
                  'Total Fatalities for Each Airline (1985-2014)')
    plt.title(plot_title)
    plt.ylabel('Counts')
    plt.xlabel('Airline')
    plt.xticks(rotation=45)
    plt.legend(title='Metrics')
    plt.tight_layout()
    plt.show()
    # Prepare data for plotting
    periods = ['1985-1999', '2000-2014']

    # Initialize the figure
    plt.figure(figsize=(14, 8))

    # Plot trend lines for each airline
    for index, row in query_result_pd.iterrows():
        fatalities = [row['fatalities_85_99'], row['fatalities_00_14']]
        plt.plot(periods, fatalities, marker='o', label=row['airline'])

    # Customize the plot
    plt.title('Total Fatalities Change for Each Airline '
              '(1985-1999 vs 2000-2014)')
    plt.ylabel('Number of Fatalities')
    plt.xlabel('Time Period')
    plt.legend(title='Airlines', bbox_to_anchor=(1.05, 1), loc='upper left')
    plt.grid(True)
    plt.tight_layout()

    # Show the plot
    plt.show()


query_transform()
viz()
```

```
Data validation passed. 10 rows available.
```

```
<Figure size 1500x700 with 0 Axes>
```





- Visualization 1: Total Incidents vs. Fatal Accidents vs. Total Fatalities for Each Airline (1985-2014)

Incidents vs. Fatalities Disparity: It is noteworthy that some airlines, such as Aeroflot, with a high number of total incidents do not necessarily have a proportionately high number of fatalities. This could suggest effective emergency response and safety procedures that mitigate the severity of incidents.

- Visualization 2: Total Fatalities Change for Each Airline (1985-1999 vs 2000-2014)

Overall Improvement: Most airlines have shown a decrease in the number of fatalities over the two time periods, which is a positive trend. This suggests that overall, safety may have improved in the airline industry.

Specific Airline Performance: Some airlines, such as Delta/Northwest and Saudi Arabian, have shown a significant decrease in fatalities, indicating a substantial improvement in their safety record. These airlines could be examined as case studies to understand what specific actions contributed to this improvement.

Challenges for Some: Conversely, airlines like American and Air France have shown a significant increase in fatalities. This would be a point of concern, and it's recommended that management conduct an in-depth review of safety protocols, fleet maintenance, and pilot training programs. Understanding the reasons behind the increase is critical to reversing this trend.