Integration Document for Aviation Emissions Analysis Project

# 1. Project Overview

- Objective: To analyze carbon emissions and fuel efficiency of various aircraft models (Boeing 737, Boeing 787, Airbus A380, Fokker 70, and Embraer E190) from 2019 to 2024. The goal is to identify patterns, forecast future emissions, and suggest strategies for reducing the carbon footprint of aviation in alignment with SDG 13: Climate Action.

- Components:  
 - Database: SQL database to store aircraft data, emissions, fuel efficiency, and route information.  
 - Data Analysis: Python/SQL for data analysis and predictive modeling.  
 - Dashboard: Interactive Excel dashboard for visualizing insights.

# 2. System Architecture

Overview: The system architecture includes a SQL database for data storage, a Python environment for data analysis and predictive modeling, and an Excel dashboard for displaying results.

# 3. Data Flow

- Input Data: Data is manually entered or imported into the SQL database, including aircraft models, emissions per kilometer, fuel efficiency, and flight routes.

- Data Processing: Data cleaning, transformation, and analysis are performed using SQL queries and Python scripts.

- Output Data: Processed data is exported to Excel for visualization on the dashboard.

# 4. Database Integration

- Database Design: The ERD includes entities such as Aircraft, Routes, Emissions Data, and Fuel Efficiency, with relationships defined through foreign keys.

The SQL code has been explained in the SQL documentation.

# 5. Data Analysis Integration

- Analysis Tools: Used Excel for data analysis.  
- Scripts and Queries: Excel scripts are used to run predictive models and generate insights. Example analysis might include calculating average emissions per year and forecasting future trends using time series analysis.  
- Automated Workflow: Implement automated scripts to refresh data in the database and update the Excel dashboard periodically.

# 6. Excel Dashboard Integration

- Dashboard Design: The dashboard will include interactive charts and graphs, such as:  
 - Line Charts: To show trends in emissions over time for each aircraft model.  
 - Bar Charts: Comparing fuel efficiency across different aircraft models.  
 - Data Import: Use Excel’s Data Import functionality to pull data from the SQL database or processed CSV files from Python scripts.  
- Dynamic Links: Use Excel functions and pivot tables to dynamically update visualizations based on the latest data.

# 7. Testing and Validation

- Testing Procedure\*: Validate data imports, ensure accuracy in SQL queries, and cross-check analysis results with expected outputs.  
- Validation Checks: Implement data validation rules in Excel to ensure no incorrect data is displayed on the dashboard.

# 8. Error Handling and Troubleshooting

- Common Issues: Potential issues include database connection errors, data mismatches, and incorrect script execution.  
- Error Logs: Maintain logs for database transactions to identify and troubleshoot errors quickly.

For importation of the data from MySQL Workbench to Excel, I used the following procedure:

1. In MySQL Workbench, execute the query to display results in the result grid.
2. In the result grid, click on the export button (typically represented by a disk icon or an arrow pointing down).
3. Select "Export Result Set..." or a similar option to export the displayed data.
4. Choose CSV as the export format because it’s compatible with Excel and retains data consistency.
5. Ensure that you select options to include headers if you want column names in Excel.
6. Save the CSV file in an appropriate location on your computer.
7. Import Data into Excel
8. Open Microsoft Excel:
9. Launch Excel and open a new or existing workbook where you want to import the data.
10. Go to the Data tab in Excel.
11. Click on "Get Data" > "From Text/CSV" (options might vary slightly depending on your Excel version).
12. Browse to the location where you saved the exported CSV file from MySQL.
13. Select the file and click "Import".