Faculty of Media Engineering and Technology
German University in Cairo
Dr. Nourhan Ehab

# Milestone 2: Airline Customer Holiday Booking *(10%)*

*Deadline: 2nd of December at 23:59*

In this milestone, we transition from predicting passenger satisfaction using tabular data to uncovering deep, verifiable insights into **airline booking and travel experience trends** through a structured **Neo4j Knowledge Graph (KG)**. The KG transforms raw operational and feedback data into an interconnected network that explicitly captures relationships among passengers, flights, routes, cabin classes, and satisfaction drivers. This structure enables the pre-computation of complex analytical features such as route popularity, traveler profiles, and sentiment patterns. The KG will serve as the verifiable knowledge source for a **Graph Retrieval-Augmented Generation (GraphRAG)** system in the next milestone, enabling users to ask complex, multi-criteria airline booking questions in natural language, such as “Which routes provide the highest satisfaction for frequent travelers in Economy Class?”, and receive accurate, transparent answers grounded directly in the graph’s structure.

The core challenge is transforming the tabular Airline data into a graph structure that explicitly models positional scoring rules, temporal flow, and complex relationships for efficient natural language reasoning.

You will be working with **“Airline_surverys_sample.csv”**. This .csv file contains a sample from the original detailed survey .csv file, with each row representing a passenger’s experience on a particular flight. Columns include flight identifiers (flight_number, origin_station_code, destination_station_code), passenger information (record_locator, loyalty_program_level, generation), journey characteristics (number_of_legs, class, fleet_type_description, actual_flown_miles), and feedback metrics such as arrival_delay_minutes and food_satisfaction_score. You will use this dataset to build a knowledge Graph capturing passengers’ feedback, flights, and airports, and then run queries to extract insights on overall satisfaction and travel patterns.

Faculty of Media Engineering and Technology
German University in Cairo
Dr. Nourhan Ehab

GUC
German University in Cairo

# Milestone 2 Requirements:

This milestone requires you to construct a comprehensive Knowledge Graph database using Neo4j that represents the airline booking domain according to the specified schema. The knowledge graph will model the complex relationships between flights, their journey requirements, creating a rich semantic network that enables sophisticated querying and analysis.
Your primary tasks include:

1. **Knowledge Graph Construction**

Implement a script (using Python and the Neo4j driver) that reads the provided CSV files and constructs the knowledge graph according to the exact schema specification provided below. This involves creating nodes for Flight, Passenger, Airport, and Journey, and establishing the relationships between them as defined in the schema.

2. **Schema Compliance**

It is critical that you strictly adhere to the provided schema. You are **not permitted** to modify the schema in any way, including:
- Changing the type of any node (*e.g., you cannot rename "Flight" to "Plane"*)
- Changing the type of any relationship *(e.g., you cannot rename ": ARRIVES_AT" to ": ARRIVED_IN" or ": REACHED"*)
- Removing or modifying any properties specified for nodes
- Removing any node types or relationship types from the schema

3. **Implementing The Given Scoring Rule**

You need to apply the overall satisfaction scoring rules outlined below in order to identify the number of passengers whose satisfaction levels exceed the defined threshold. These passengers are those with an **overall_satisfaction_score greater than 3**, calculated using weighted components from their journey data. Keep in mind that clamping is applied in order to ensure that the calculated values stay within specific ranges.

4. **Cypher Query Development**

After constructing the knowledge graph, you will develop Cypher queries to answer a comprehensive set of analytical questions. These queries will test your understanding of graph traversal, aggregation, filtering, and pattern matching in Cypher. The queries range from simple filtering operations to complex multi-hop traversals that require careful consideration of relationship directions and node properties.

Faculty of Media Engineering and Technology
German University in Cairo
Dr. Nourhan Ehab

## The Knowledge Graph Schema:

### 1. Nodes:

- Passenger**:** record_locator (unique identifier), loyalty_program_level, generation
- Journey**:** feedback_ID (unique identifier), food_satisfaction_score, arrival_delay_minutes , actual_flown_miles ,number_of_legs, passenger_class
- Flight**:** flight_number, fleet_type_description (uniquely identified by both properties)
- Airport**:** station_code (e.g., ORD, LAX)

### 2. Relations:

- (Passenger)-[:TOOK ]->(Journey)
- (Journey)-[:ON ]->(Flight)
- (Flight)-[:DEPARTS_FROM ]->(Airport)
- (Flight)-[:ARRIVES_AT ]->(Airport)

## Knowledge Graph Verification Queries

**NOTE ON VERIFICATION:** These are example questions & answers for initial structure confirmation. Create queries that answer the following questions. If your queries return the same answers as below, then the structure of your knowledge graph is sound. The final assessment of the graph's design and robustness will involve a separate set of private queries to ensure the model is flexible and not hardcoded to these specific examples.

1. Identify the 5 airport-to-airport routes with the highest number of flights. This simply shows which routes have the most scheduled flights. Your query should return:
   - **origin** – the departure airport code
   - **destination** – the arrival airport code
   - **flight_count** – the total number of flights on that route

   Results must be **sorted in descending order of flight_count**.

2. Identify the top 10 Flights with the Most Passenger Feedback. It is important to highlight that “most feedback” does not imply “positive feedback”, these flights simply have the largest volume of responses, regardless of whether the ratings were good or bad. Your query should return the Flight IDs as well as the passenger counts per flight, **sorted in descending order.**

3. Calculate the average food satisfaction score for all multi-leg journeys (journeys with more than one flight leg), grouped by passenger generation. Your query should return:

   - **generation**: the passenger’s generation group
   - **multi_leg_count**: how many multi-leg journeys that generation took
   - **avg_score**: the average food satisfaction score for those journeys

   Results must be **sorted in descending order of multi_leg_count**.

4. Calculate the average arrival delay for all flights and identify the 10 flights with the shortest delays. Your query should return:
   - **flight_id:** the unique identifier of the flight
   - **avg_arrival_delay:** the average delay in minutes across all journeys for that flight

   Results must be **sorted in ascending order of avg_arrival_delay**

5. Calculate the average flown miles for passengers, grouped by loyalty program level. Your query should return:
   - **loyalty_level**: the passenger’s loyalty program category
   - **avg_actual_flown_miles**: the average flown miles for that level

   Results must be **sorted in descending order of average flown miles**

Faculty of Media Engineering and Technology
German University in Cairo
Dr. Nourhan Ehab

## Overall Satisfaction Score Rule Calculation:

Identify the number of passengers with an overall satisfaction score higher than 3, where **overall_satisfaction_score** represents the passenger's overall score based on how satisfied they are. **overall_satisfaction_score** = 0.5*food + 0.35*delay_score + 0.1*legs_score + 0.05*miles_score.

| Component | Source in KG | Rule |
|---|---|---|
| **Food Score** | Journey.food_satisfaction_score | Direct value from 1–5 |
| **Delay Score** | Journey.arrival_delay_minutes | delay_score = 5 - clamp(abs(arrival_delay_minutes)/20, 0, 5)<br>where (abs(arrival_delay_minutes)/20) is expressed as a decimal value rounded to one decimal place. |
| **Legs Score** | Journey.number_of_legs | legs_score = 5 - clamp(number_of_legs * 1.5, 0, 5)<br>where (number_of_legs * 1.5) is expressed as a decimal value rounded to one decimal place. |
| **Miles Score** | Journey.actual_flown_miles | miles_score = 5 - clamp(actual_flown_miles / 3000, 0, 5)<br>where (actual_flown_miles / 3000) is expressed as a decimal value rounded to one decimal place. |

Answer: **1856**

Faculty of Media Engineering and Technology
German University in Cairo
Dr. Nourhan Ehab

## Deliverables:

You are required to submit the following by filling out the form:

1. **Create_kg.py**
   - Python script that creates the Knowledge Graph in Neo4j.
   - Should read CSV files located in the same directory
   - Should follow the exact schema specification.
   - The script should use the config.txt file with Neo4j credentials. Not doing so might result in 0.
2. **config.txt**

   URI=neo4j://localhost:7687

   USERNAME=neo4j

   PASSWORD=your_password

3. **rule.txt**

A text file containing the implemented rule for the scoring task.

4. **queries.txt**

   A text file containing the queries for the above questions with their numbers.

   //1. Query 1

   Match(...)

   Return (....)

   //2. Query 2

   Match(..)

   Return (....)

   Etc..