Barcelona School of Economics

23D021: Data Management

Lab 2: Querying

Assignment

Note: This is a hands-on lab on Database Querying. We will be using one of the most popular object relational databases: `PostgreSQL`*. We will practice how to query databases using SQL.*
In the training document, we provide instructions for setting up the environment and here we list the exercises to be solved. One group member, in the name of the group, must upload the solution. Remember to include the names of all group members in your solutions. Please check the assignment deadline and be sure to meet it. It is a strict deadline!

A Lab Statement

This lab is about running queries in the Airline database you created for the first Lab.

Note: You can use the solution (database) of any of the team members for the queries below. However, please clearly indicate which solution was used.

A.1 Querying with SQL

(Q1) Find customers who have made at least one booking in the last month and their booking details.

(Q2) List all flights with delayed or canceled status, including flight crew details and aircraft assigned.

(Q3) Get the total number of miles accumulated by a frequent flyer along with their upcoming bookings.

(Q4) Find flights departing in the next 7 days that are operated by a specific aircraft model but are not yet fully booked.

(Q5) Generate a report of flights where maintenance schedules have conflicted with the assigned aircraft.

(Q6) Calculate the revenue generated by each flight, including payment status for all bookings made.

(Q7) Find customers who have never booked a flight.

(Q8) Get flights that are fully booked (i.e., no available seats).

(Q9) Find frequent flyers who have flown the most miles but haven't made any bookings in the past year.

(Q10) Get the total Bookings and Revenue Generated per Month.

(Q11) Get the top 5 Most Popular Flight Routes.

(Q12) Show how active frequent flyers have been, summarizing their total miles, number of bookings, and total money spent.

A.2 Modeling and querying with JSON

Incorporating JSON fields into a relational database design can add flexibility to store semi-structured or dynamic data that may not require a rigid schema.

Modeling

Model (add) the following tables in your existing database:

- **CustomerPreferences**: This table stores dynamic customer preferences (e.g., special meal requests, seating preferences, communication preferences), which can vary widely between customers and evolve over time.
  - Example

```
{
  "meal": "vegetarian",
  "seating": {
    "aisle": true,
    "extra_legroom": true,
    "seat_near_exit": false
  },
  "notifications": {
    "email": true,
    "sms": false
  }
}
```

- **Aircraft Maintenance Logs**: This table stores detailed maintenance logs for each aircraft in a JSON format, which may include flexible information about specific components, inspections, and parts replaced.

– Example


```
{
  "date": "2024-10-15",
  "check_type": "Full Inspection",
  "components_checked": [
    {
      "name": "Engine",
      "status": "Operational",
      "last_replaced": "2023-12-01"
    },
    {
      "name": "Hydraulics",
      "status": "Requires Service"
    }
  ]
}
```


- **Customer Feedback and Survey**: This table captures customer feedback in a dynamic format, such as post-flight surveys or service feedback, which can include a wide range of topics and ratings.

  – Example


```
{
  "survey_date": "2024-10-20",
  "rating": 4,
  "comments": "The flight was good, but the food options were limited.",
  "topics": {
    "comfort": 5,
    "service": 4,
    "cleanliness": 3,
    "entertainment": 2
  }
}
```


Querying with JSON

Now that we have tables that store JSON data. We want you to write the following queries:

(JQ1) Find customers who prefer extra legroom and have submitted feedback with a service rating lower than 3.

(JQ2) Identify flights that used aircraft with maintenance issues flagged within the last 6 months and correlate them with customer feedback on comfort.

(JQ3) Find customers who have not provided feedback but have specific preferences (e.g., vegetarian meals).

(JQ4) Find the most common customer preferences (e.g., meal, seating) for customers who rated their flight 5 stars in overall feedback.

A.3 Triggers

A trigger is procedural code that is automatically executed in response to certain events on a particular table in a database.

We want you to implement the following triggers:

(T1) `check_maintenance_schedule`: Trigger to check aircraft maintenance schedule before assigning Aircraft to a Flight: This trigger will ensure that aircraft scheduled for maintenance are not assigned to flights during the maintenance period.

- Before inserting or updating a flight record with an aircraft assignment, check if the aircraft is undergoing maintenance.

(T2) `archive_old_feedback`: Trigger to automatically archive customer feedback after 2 years: This trigger moves customer feedback older than 2 years to an archive table, maintaining the primary table with only recent feedback.

- When inserting new feedback, if there is any feedback older than 2 years, move it to an archive table (CustomerFeedbackArchive) before inserting the new data.

Deliverables

(a) A PDF file (max two A4 pages) containing:

- The conceptual model of the solution (database) used for the exercises.
- Any assumptions and justification/explanations of the decisions made for Sections A, B, and C.

(b) A commented (including explanations) `.sql` file that implements the queries in Section A.

(c) A commented (including explanations) `.sql` file that implements both the table creations and the queries in Section B.

(d) A commented (including explanations) `.sql` file that implements the triggers in Section C.

Assessment Criteria

i) Conciseness of explanations

ii) Understandability

iii) Coherence

iv) Soundness