# Homework #6

CIS 4301 - Spring 2025

## Submission Format

Submit a soft copy of your solution via e-Learning (`http://elearning.ufl.edu`) by the end of the day (23:59 / 11:59 PM) on April 23th. Save your solution as a PDF file and name it `hw6.pdf`. Include your name, assignment number, and due date at the top of the file.

## Problem Statement

This assignment uses the following Travel Agency database schema, with the specified primary key and foreign key constraints:

- **Booking(agent: String, traveler_ssn: integer, trip_id: integer)**
  *Foreign keys:* traveler_ssn references Traveler(ssn), trip_id references Trip(id), agent references TravelAgent(name)
- **GoesOn(ssn: integer, id: integer)**
  *Foreign keys:* ssn references Traveler(ssn), id references Trip(id)
- **Leg(Trip id: integer, startLocation: String, endLocation: String, startDate: Date, endDate: Date)**
  *Foreign keys:* trip_id references Trip(id)
- **Owns(ssn: integer, passport_number: integer, country: String)**
  *Foreign keys:* ssn references Traveler(ssn)

- **Passport(passport_number: integer, country: String, expirationDate: Date, holderName: String)**
- **TravelAgent(name: String, years_experience: integer, phone: String)**
- **Traveler(name: String, ssn: integer, dob: Date)**
- **Trip(id: integer, start_location: String, end_location: String, start_date: Date, end_date: Date)**

## Instructions

### Part 0: Start Your MariaDB

### Part 1: Create a New Database

Before beginning this assignment, create a new database named `HW_6_7`. This database will be used exclusively for **Homework 6 and Homework 7**. The previous database you created for Homework 3 and Homework 4 will **not** be used for this assignment.
Use the following command to create the new database in MariaDB:

```
CREATE DATABASE HW_6_7;
```

After creating the database, make sure to switch to it before proceeding:

```
USE HW_6_7;
```

## Part 2: Create Tables with Constraints (40 points)

Write SQL statements to create each of the tables listed above, including primary keys and the specified foreign key constraints. Ensure the following:

1. **Primary Keys and Foreign Keys:** Include primary keys for each table and establish foreign key constraints as specified in the schema.

2. **Attribute-Based Constraints:** Define the following two attribute-based constraints:

   - `years_experience` in `TravelAgent` must be greater than or equal to 1.
   - `expirationDate` in `Passport` must be after January 1, 2020.

3. **Tuple-Based Constraints:** Define the following two tuple-based constraints:

   - In `Leg`, ensure `endDate` is not earlier than `startDate`.
   - In `Trip`, ensure `end_date` is not earlier than `start_date`.

4. **Cascade on Update and Delete:** Set up a foreign key constraint such that updating and deleting a `Traveler` will automatically update and delete all related records in `Booking`, `Owns` and `GoesOn` tables.

5. **Cascade on Delete:** Implement a foreign key constraint in `Booking` such that if an agent in `TravelAgent` is deleted, the `agent` attribute in any related `Booking` records is deleted.

6. **NOT NULL Constraint:** Add a `NOT NULL` constraint to the `name` column in `Traveler`.

## Part 3: Insert Data

Use the following SQL commands to insert data into each relation. Follow the format shown to populate each relation with the tuples provided below. Make sure to insert the tables in the specified sequence. This order ensures compliance with foreign key constraints:

1. **Insert into ‘Traveler‘, ‘TravelAgent‘, ‘Trip‘, and ‘Passport‘**:

```sql
-- Traveler
INSERT INTO Traveler (name, ssn, dob) VALUES
('John Doe', 101, '1985-06-12'),
('Alice Brown', 102, '1992-03-05'),
('Mike Johnson', 103, '1998-09-17'),
('Lisa Turner', 104, '2000-12-22'),
('Sarah Connor', 105, '2003-11-01');

-- TravelAgent
INSERT INTO TravelAgent (name, years_experience, phone) VALUES
('Emily Clark', 12, '123-456-7890'),
('Robert Smith', 8, '234-567-8901'),
('Anna Wilson', 15, '345-678-9012'),
('Michael Davis', 10, '456-789-0123'),
('Mary Johnson', 3, '567-890-1234');

-- Trip
INSERT INTO Trip (id, start_location, end_location, start_date, end_date) VALUES
(201, 'New York', 'Paris', '2023-07-10', '2023-07-20'),
(202, 'Tokyo', 'Sydney', '2023-08-01', '2023-08-15'),
(203, 'London', 'Rome', '2023-09-05', '2023-09-15'),
(204, 'Berlin', 'Tokyo', '2023-10-02', '2023-10-12'),
(205, 'Miami', 'New York', '2023-10-22', '2023-10-29');
```

```
-- Passport
INSERT INTO Passport (passport_number, country, expirationDate, holderName) VALUES
(3001, 'USA', '2025-11-30', 'John Doe'),
(3002, 'Canada', '2026-08-20', 'Alice Brown'),
(3003, 'UK', '2024-09-15', 'Mike Johnson'),
(3004, 'Australia', '2027-02-10', 'Lisa Turner'),
(3005, 'France', '2023-12-05', 'Sarah Connor');
```

2. **Insert into \`Owns\`**:

```
INSERT INTO Owns (ssn, passport_number, country) VALUES
(101, 3001, 'USA'),
(102, 3002, 'Canada'),
(103, 3003, 'UK'),
(104, 3004, 'Australia'),
(105, 3005, 'France');
```

3. **Insert into \`Booking\`**:

```
INSERT INTO Booking (agent, traveler_ssn, trip_id) VALUES
('Emily Clark', 101, 201),
('Robert Smith', 102, 202),
('Anna Wilson', 103, 203),
('Michael Davis', 104, 204),
('Emily Clark', 105, 205);
```

4. **Insert into \`GoesOn\`**:

```
INSERT INTO GoesOn (ssn, id) VALUES
(101, 201),
(102, 202),
(103, 203),
(104, 204),
(105, 205);
```

```
5. **Insert into `Leg`**:
    INSERT INTO Leg (trip_id, startLocation, endLocation, startDate, endDate) VALUES
    (201, 'New York', 'Paris', '2023-07-10', '2023-07-20'),
    (202, 'Tokyo', 'Sydney', '2023-08-01', '2023-08-15'),
    (203, 'London', 'Rome', '2023-09-05', '2023-09-15'),
    (204, 'Berlin', 'Tokyo', '2023-10-02', '2023-10-12'),
    (205, 'Miami', 'New York', '2023-10-22', '2023-10-29');
```

## Part 4: Triggers (30 points)

Define the necessary tables and triggers to enforce additional data integrity:

- **Trigger 1 (15 points):**
  - First, create the `DeletedTravelerLog` table to store logs of deleted travelers. This table should include columns for `name` and `ssn` to record the details of each deleted traveler.
  - Next, create a trigger on the `Traveler` table. Before a record is deleted, this trigger should log the `name` and `ssn` of the traveler to the `DeletedTravelerLog` table.
- **Trigger 2 (15 points):**
  - First, create the `TravelerStats` table. This table should include a column for the traveler's `ssn` and a column for the total `trip_count` to store the number of trips each traveler has taken.
  - Next, create a trigger on the `GoesOn` table. After a new record is inserted into `GoesOn`, this trigger should update the `trip_count` in the `TravelerStats` table for the traveler's `ssn` to reflect the total number of trips.

## Part 5: Constraint Scenarios (30 points)

Test the functionality of the constraints and triggers implemented in Parts 2 and 4. For each scenario below, write and run the necessary SQL commands to test the specified behavior. Describe any violations and explain how the constraints or triggers enforce data integrity.

1. **Delete a Trip ID in Trip (3 points):** Attempt to delete a `Trip` entry where the `id` is 201. Observe and describe the violation that occurs due to the foreign key constraints.

2. **Update an SSN in Owns (3 points):** Attempt to update the `ssn` in `Owns` that has the value 101 to 999. Describe the violation and explain how the foreign key constraint prevents this update.

3. **Insert a Travel Agent with 0 Years of Experience (3 points):** Try to insert a new `TravelAgent` ('Jake Taylor', 0, '678-901-2345'). Describe the result and explain how the attribute-based constraint enforces data validity.

4. **Update Passport Expiration Date to an Invalid Date (3 points):** Attempt to update the `Passport` with the number 3001 to have an expiration date '2018-12-01'. Describe the result and explain the purpose of the attribute-based constraint on expiration dates.

5. **Insert a Trip with an Invalid Date Range (3 points):** Try to insert a new `Trip` with the following values (206, 'Paris', 'Berlin', '2023-12-10', '2023-12-01'). Describe the violation and explain how the tuple-based constraint ensures logical date ranges.

6. **Update a Traveler to Cascade Changes (3 points):** Update the `ssn` of `Traveler` 102 to have the value 106 and observe how the update cascades to the `Booking` , `Owns` and `GoesOn` tables. Describe the effect and explain how the cascade on update policy maintains referential integrity.

7. **Delete a Travel Agent to Cascade changes in Booking (3 points):** Delete the `TravelAgent` 'Emily Clark' and observe how the delete cascades to the `Booking`. Describe the effect and explain how the cascade on update policy maintains referential integrity.

8. **Insert a Traveler with a NULL Name (3 points):** Attempt to insert a `Traveler` with the following values (NULL, 107, '2001-05-10') to test the `NOT NULL` constraint. Describe the result and explain why a NOT NULL constraint is important for certain fields.

9. **Trigger 1: Deletion of Traveler (3 points):** Delete a `Traveler` with ssn 103 and verify that the trigger logs the deleted traveler's `name` and `ssn` to the `DeletedTravelerLog` table. Describe the effect of the trigger.

10. **Trigger 2: Insert into GoesOn to Update TravelerStats (3 points):** Insert a new record with the values (104, 205) into `GoesOn` and check the `TravelerStats` table to confirm the trip count updates as expected. Describe the result and explain how the trigger maintains the `TravelerStats` table.

## What to Submit

For this assignment, please include the following in your submission:

- **Part 2: SQL Commands for Creating Tables**
  Submit a textual Format of SQL commands you used to create each table with the specified constraints. Ensure that all primary keys, foreign keys, attribute-based, and tuple-based constraints are included.

- **Part 4: SQL Commands for Creating Triggers**
  Submit a textual Format SQL commands used to create the `DeletedTravelerLog` and `TravelerStats` tables, as well as the SQL commands for each trigger.

- **Part 5: SQL Commands and Screenshots for Testing Constraints and Triggers**

  For each scenario in Part 5, submit the following:

  - The SQL command(s) used to test the constraint or trigger.
  - The violation occurred and how the constraints or triggers enforce data integrity.
  - Screenshots of the output, showing the violation message and the modified tables to demonstrate the effect of the constraint or trigger.

Make sure that all SQL commands are formatted and clearly labeled, and that all screenshots are legible and correspond to the correct scenario. This ensures clarity in demonstrating how constraints and triggers enforce data integrity within the Travel Agency database schema.