

Database Systems (CSCI 4380-01)

Homework 1 — Practice Problems (Q1)

Peiyun Li

February 3, 2026

Overview

This document contains **original practice problems** and **fully worked, step-by-step solutions** for Homework 1, Questions 1 and 2. It follows the same dataset, schema, relational algebra notation, and formatting conventions as the original assignment.

Course Context. Homework 1 focuses on relational algebra and normalization using an airline database.

Database Description.

The Airline Data Analysis database contains the following relations (keys are underlined):

- Aircraft(Code, Model, Range)
- Airports(Code, Name, City, Latitude, Longitude, Timezone)
- Flights(FId, FlightNumber, ScheduledDeparture, ScheduledArrival, DepartureAirportCode, ArrivalAirportCode, AircraftCode, ActualDeparture, ActualArrival)
- Tickets(TicketNo, PassengerId)
- TicketFlights(TicketNo, FId, FareConditions, Amount)
- Passengers(PassengerId, FirstName, LastName, DOB)

Date/Time Note. Date comparisons use the homework format, e.g. `ScheduledDeparture::date = date '02/01/2026'` and `ActualDeparture <= now()`.

Practice Problem for Homework 1 Question 1

Problem Statement

Use the same airline dataset and relations as Homework 1.

- Return the **airport code and city** of all airports located in the timezone "Europe/London".
- Return the **FlightNumber** of all flights that are scheduled to **depart today** but have **not yet departed**.
- Return the **first and last names** of passengers who have flown on **at least one flight** but have **never flown in Economy class**.

Solution (a): Airports in Europe/London

Step 1. Timezone and city are stored in the `Airports` relation.

Step 2. Select airports whose timezone is "Europe/London".

Step 3. Project the requested attributes (Code and City).

Relational Algebra:

```
project_(Code, City)
  select_{Timezone = "Europe/London"}(Airports)
```

Semantics: Returns the codes and cities of airports located in the Europe/London timezone.

Solution (b): Scheduled today but not yet departed

Assume today is date '02/01/2026'.

Step 1. Select flights scheduled to depart today.

Step 2. Identify flights that have already departed (`ActualDeparture <= now()`).

Step 3. Subtract departed flights from all flights scheduled today.

Step 4. Project the `FlightNumber`.

Relational Algebra:

```
Today(FId, FlightNumber) = project_(FId, FlightNumber)
  select_{ScheduledDeparture::date = date '02/01/2026'}(Flights)

Departed(FId, FlightNumber) = project_(FId, FlightNumber)
  select_{ScheduledDeparture::date = date '02/01/2026' AND
    ActualDeparture <= now()}(Flights)

project_(FlightNumber)(Today - Departed)
```

Semantics: Returns flight numbers of flights scheduled to depart today that have not yet departed.

Solution (c): Flew at least once, but never Economy

Step 1. Passengers who have flown appear in `Tickets * TicketFlights`.

Step 2. Project `PassengerId` to obtain all passengers who have flown.

Step 3. Identify passengers who have ever flown Economy.

Step 4. Subtract Economy flyers from all flyers.

Step 5. Join with `Passengers` to retrieve names.

Relational Algebra:

```
Flew(PassengerId) = project_(PassengerId)(Tickets * TicketFlights)

Economy(PassengerId) = project_(PassengerId)
  (select_{FareConditions = "Economy"}(TicketFlights) * Tickets)

project_(FirstName, LastName)((Flew - Economy) * Passengers)
```

Semantics: Returns passengers who have taken at least one flight but never in Economy class.