Middle East Technical University

Department of Computer Engineering

CENG 351

Data Management & File Structures

Fall 2022–2023

In Class Activity 3 — SQL LAB

1 Specifications

IMPORTANT NOTE
SQLite does not support ANY/ALL.
SQLite can compare two text fields. For example, if one of the times is "10:40" and the other is "11:40", you can compare them using "<" and ">" operators. The earlier time is the smaller one. Your answers should **not include** duplicate rows (You may use DISTINCT if needed, however, do not eliminate useful data by using DISTINCT).

You are given the following database schema:

Users(userID, userName, role, status)
Buses(busID, driverID, starting_point, destination, deptime)
Sold_Tickets(userID, busID, price)
Request_Tickets(userID, starting_point, destination, deptime)

Users : Users of our database, can be either a passenger or a driver.

- userID (INT): id of the user
- userName (TEXT): name of the user
- role (TEXT): Either "passenger" or "driver"
- status (TEXT): The information about a user (both driver and passenger). Can be "banned" or "regular"

Buses : All of the buses on our database.

- busID (INT): id of the bus
- driverID (INT): id of driver (REFERENCES Users)
- starting_point (TEXT): The starting terminal of the bus.
- destination (TEXT): The destination terminal of the bus.
- deptime (TEXT): The departure time of the bus.

Sold_Tickets : The tickets that are bought by the passengers.

- userID (INT): id of the passenger (REFERENCES Users)
- busID (INT): id of the bus (REFERENCES Buses)
- price (INT): ticket price

Request_Tickets : The new feature of our system where passengers enter the starting and destination points and the time that they are able to catch the bus. Note: You will use this table only in Q6.

- userID (INT): id of the passenger (REFERENCES Users)
- starting_point (TEXT): The starting terminal of the passenger.
- destination (TEXT): The destination terminal of the passenger.
- deptime (TEXT): The earliest possible time for the passenger.

2 Questions

Q1 (10pts). Our system wants to check the tickets that are sold to the banned passengers mistakenly. You need to gather userName, busID, and price of the ticket. Order by userName ASC, busID DESC

Q2 (15pts). List the users who bought a ticket from a bus that has the driver "Osman Kocoglu". List userName, busID, starting_point, destination, and deptime of the ticket. Order by userName ASC, deptime DESC

Q3 (15pts). List the users who bought a ticket from all the buses that have the driver "Osman Kocoglu". List userID, userName, status. Order by userName ASC, status DESC.

Q4 (20pts). The buses table stores starting_point, destination pairs for each bus. Our company wants to investigate how many distinct routes we have on our system. However, although both "Istanbul-Ankara" and "Ankara-Istanbul" are in our table, we only want the one with the **lexicographically** smaller starting_point ("Ankara-Istanbul"). List all distinct one-way routes on our system as starting_point, destination pairs. Order by starting_point ASC, destination ASC.
Note: If there is a trip from city A to city B on our table, it is guaranteed that there will be a trip from city B to city A.

Q5 (20pts). Some of our drivers are banned from the traffic and their travel will be canceled. We want to calculate the cost of the situation. For each canceled bus, you should calculate the total prices of the tickets sold for that bus. You should list the busID, driverName, and the total cost of the cancellation. Order by the busID ASC.

Q6 (20pts). (Check the explanation of the Request_Tickets table before starting this question). Our company wants to test the Request_Tickets functionality. For the passenger "Elif Usta", find the earliest possible bus for each request of Elif. List busId, starting_point, destination, and deptime. Order by deptime ASC, starting_point ASC.