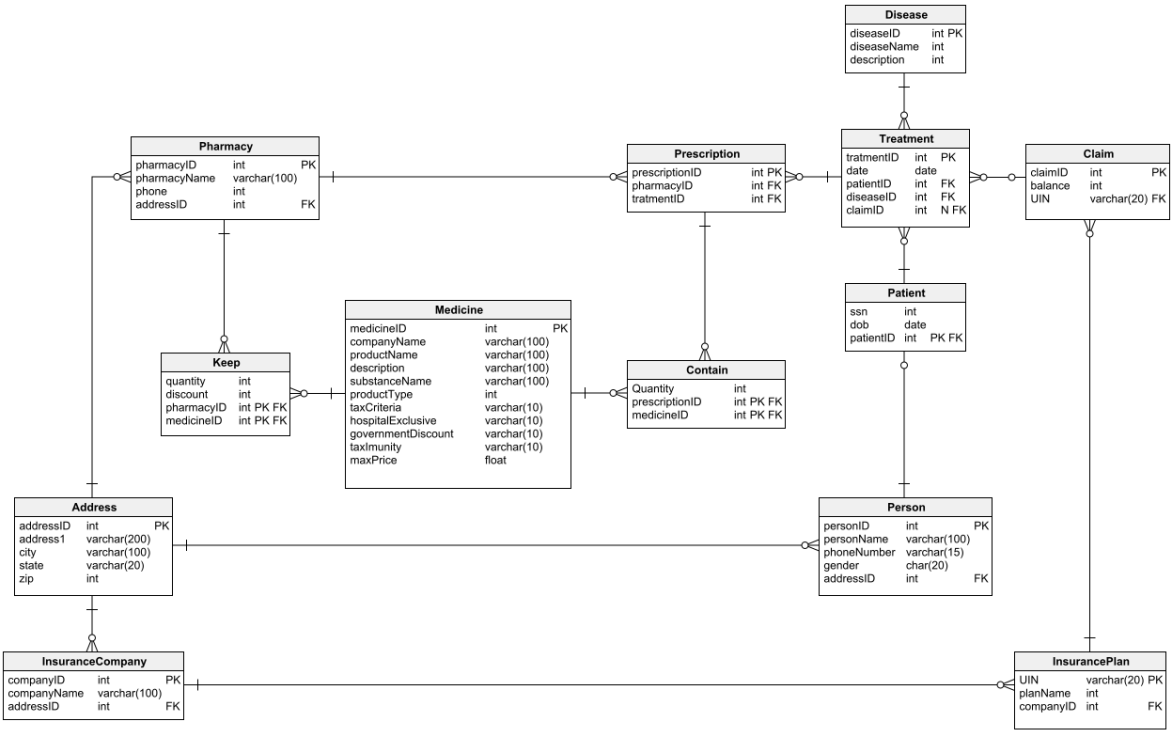
PROJECT

SQL Stored Routines

Database schema:



Problem Statement 1:

The healthcare department has requested a system to analyze the performance of insurance companies and their plan.

For this purpose, create a stored procedure that returns the performance of different insurance plans of an insurance company. When passed the insurance company ID the procedure should generate and return all the insurance plan names the provided company issues, the number of treatments the plan was claimed for, and the name of the disease the plan was claimed for the most. The plans which are claimed more are expected to appear above the plans that are claimed less.

Problem Statement 2:

It was reported by some unverified sources that some pharmacies are more popular for certain diseases. The healthcare department wants to check the validity of this report.

Create a stored procedure that takes a disease name as a parameter and would return the top 3 pharmacies the patients are preferring for the treatment of that disease in 2021 as well as for 2022.

Check if there are common pharmacies in the top 3 list for a disease, in the years 2021 and the year 2022.

Call the stored procedure by passing the values “Asthma” and “Psoriasis” as disease names and draw a conclusion from the result.

Problem Statement 3:

Jacob, as a business strategist, wants to figure out if a state is appropriate for setting up an insurance company or not.

Write a stored procedure that finds the num_patients, num_insurance_companies, and insurance_patient_ratio, the stored procedure should also find the avg_insurance_patient_ratio and if the insurance_patient_ratio of the given state is less than the avg_insurance_patient_ratio then it Recommendation section can have the value “Recommended” otherwise the value can be “Not Recommended”.

Description of the terms used:

num_patients: number of registered patients in the given state

num_insurance_companies: The number of registered insurance companies in the given state

insurance_patient_ratio: The ratio of registered patients and the number of insurance companies in the given state

avg_insurance_patient_ratio: The average of the ratio of registered patients and the number of insurance for all the states.

Problem Statement 4:

Currently, the data from every state is not in the database, The management has decided to add the data from other states and cities as well. It is felt by the management that it would be helpful if the date and time were to be stored whenever new city or state data is inserted.

The management has sent a requirement to create a PlacesAdded table if it doesn’t already exist, that has four attributes. placeID, placeName, placeType, and timeAdded.

Description

placeID: This is the primary key, it should be auto-incremented starting from 1

placeName: This is the name of the place which is added for the first time

placeType: This is the type of place that is added for the first time. The value can either be 'city' or 'state'

timeAdded: This is the date and time when the new place is added

You have been given the responsibility to create a system that satisfies the requirements of the management. Whenever some data is inserted in the Address table that has a new city or state name, the PlacesAdded table should be updated with relevant data.

Problem Statement 5:

Some pharmacies suspect there is some discrepancy in their inventory management. The quantity in the 'Keep' is updated regularly and there is no record of it. They have requested to create a system that keeps track of all the transactions whenever the quantity of the inventory is updated.

You have been given the responsibility to create a system that automatically updates a Keep_Log table which has the following fields:

id: It is a unique field that starts with 1 and increments by 1 for each new entry

medicineID: It is the medicineID of the medicine for which the quantity is updated.

quantity: The quantity of medicine which is to be added. If the quantity is reduced then the number can be negative.

For example: If in Keep the old quantity was 700 and the new quantity to be updated is 1000, then in Keep_Log the quantity should be 300.

Example 2: If in Keep the old quantity was 700 and the new quantity to be updated is 100, then in Keep_Log the quantity should be -600.