CIS 182 – SQL Fundamentals – Winter 2024

Team Project – Suggested Projects

(For the due date, please refer to this lab’s posting on Canvas)

This document presents a list of projects to choose from. For each project, the Tables section gives the SQL code for creating the database tables is provided.

Once you choose a project, your team will use the SQL code provided in the Tables section and execute it in MSSMS to create the tables. The team is required to complete the tasks specified in the Requirements section.

## Online Store Database:

* Design a relational database schema for an online store that sells products. Include tables for customers, orders, products, and inventory.

### Tables

* + These tables form the foundation for managing bookstore-related data.
  + Feel free to adjust the column names, data types, and constraints based on your specific requirements.
  + Feel free to expand the schema by adding more tables related to inventory, transactions, or other relevant aspects!

#### Books Table:

* Stores information about books available in the store.
* Includes columns for book ID, title, author ID, genre, and ISBN.

|  |
| --- |
| CREATE TABLE Books (  book\_id INT PRIMARY KEY,  title VARCHAR(255),  author\_id INT,  publisher\_id INT,  price DECIMAL(10, 2),  stock\_quantity INT  ); |

#### Authors Table:

* Contains details about authors.
* Columns include author ID, first name, last name, and birth year.

|  |
| --- |
| CREATE TABLE Authors (  author\_id INT PRIMARY KEY,  author\_name VARCHAR(100)  ); |

#### Publishers Table:

* Stores information about book publishers.
* Columns include publisher ID, publisher name, and location.

|  |
| --- |
| CREATE TABLE Publishers (  publisher\_id INT PRIMARY KEY,  publisher\_name VARCHAR(100),  location VARCHAR(100)  ); |

#### Customers Table:

* Records customer details.
* Includes columns for customer ID, first name, last name, email, and phone number.

|  |
| --- |
| CREATE TABLE Customers (  customer\_id INT PRIMARY KEY,  first\_name VARCHAR(50),  last\_name VARCHAR(50),  email VARCHAR(100),  phone\_number VARCHAR(20),  address VARCHAR(255)  ); |

#### Orders Table:

* Tracks customer orders.
* Columns include order ID, customer ID, order date, and total amount.

|  |
| --- |
| CREATE TABLE orders (  order\_id INT PRIMARY KEY,  customer\_id INT,  order\_date DATE,  total\_amount DECIMAL(10, 2),  FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id)  ); |

### Requirements

* + Implement CRUD (Create, Read, Update, Delete) operations for managing customer data, product details, and order history.
  + Create complex queries to retrieve information such as top-selling products, customer order summaries, and inventory levels.

## Library Management System:

* Build a database to manage library resources, including books, borrowers, and transactions.

### Tables

* + Feel free to customize the column names, data types, and constraints based on your specific requirements.
  + Feel free to adjust the schema according to your course guidelines and any additional features you’d like to include!

#### Book Table

* The Books table stores information about books, including their title, author, genre, and ISBN.

|  |
| --- |
| CREATE TABLE Books (  BookID INT PRIMARY KEY,  Title VARCHAR(255) NOT NULL,  Author VARCHAR(255),  Genre VARCHAR(50),  ISBN VARCHAR(20) UNIQUE  ); |

#### Borrowers Table

* The Borrowers table contains details about library patrons, such as their first name, last name, email, and phone number.

|  |
| --- |
| CREATE TABLE Borrowers (  BorrowerID INT PRIMARY KEY,  FirstName VARCHAR(50) NOT NULL,  LastName VARCHAR(50) NOT NULL,  Email VARCHAR(100) UNIQUE,  Phone VARCHAR(15)  ); |

#### Transactions Table

* The Transactions table records borrowing transactions, linking books to borrowers and tracking borrow and return dates.

|  |
| --- |
| CREATE TABLE Transactions (  TransactionID INT PRIMARY KEY,  BookID INT,  BorrowerID INT,  BorrowDate DATE,  ReturnDate DATE,  FOREIGN KEY (BookID) REFERENCES Books(BookID),  FOREIGN KEY (BorrowerID) REFERENCES Borrowers(BorrowerID)  ); |

### Requirements

* + Implement queries to track book availability, overdue books, and popular genres.
  + Explore JOINs to link borrower information with borrowed books.

## Employee Management System:

Develop a database for an organization’s employee records. Include tables for employees, departments, and job roles.

### Tables

* Feel free to adjust the column names, data types, and constraints based on your specific requirements. These tables provide a foundation for managing employee data and relationships within an organization.

#### Employee Table:

* Stores information about employees.
* Includes columns for employee ID, last name, first name, middle initial, gender, street address, and city.

|  |
| --- |
| CREATE TABLE Employee (  EmployeeID INT PRIMARY KEY,  LastName VARCHAR(50) NOT NULL,  FirstName VARCHAR(50) NOT NULL,  MidInitial CHAR(1),  Gender CHAR(1),  Street VARCHAR(100),  City VARCHAR(50)  ); |

#### Works Table:

* Records employee details related to their work.
* Contains columns for employee ID, last name, first name, middle initial, company name, and salary.

|  |
| --- |
| CREATE TABLE Works (  EmployeeID INT PRIMARY KEY,  LastName VARCHAR(50) NOT NULL,  FirstName VARCHAR(50) NOT NULL,  MidInitial CHAR(1),  CompanyName VARCHAR(100),  Salary NUMERIC(10, 2),  FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID),  FOREIGN KEY (CompanyName) REFERENCES Company(CompanyName)  ); |

#### Company Table:

* + Stores information about companies.
  + Includes columns for company name and city.

|  |
| --- |
| CREATE TABLE Company (  CompanyName VARCHAR(100) PRIMARY KEY,  City VARCHAR(50)  ); |

#### Manages Table:

* + Represents the management hierarchy.
  + Contains columns for employee ID, last name, first name, middle initial, manager’s last name, manager’s first name, manager’s middle initial, and start date.

|  |
| --- |
| CREATE TABLE Manages (  Manager\_EmployeeID INT,  EmployeeID INT,  StartDate DATE,  PRIMARY KEY (Manager\_EmployeeID,EmployeeID),  FOREIGN KEY (Manager\_EmployeeID) REFERENCES Employee(EmployeeID)  ); |

* Write queries to find the highest-paid employees, average salaries by department, and employee hierarchy.
* Use subqueries to identify employees with specific qualifications or experience.

## Social Media Analytics:

* Create a database to store social media posts, user profiles, and interactions (likes, comments, shares).

### Tables:

* These tables form the foundation for analyzing social media data, understanding user interactions, and deriving insights from the platform.
* Feel free to adjust the column names, data types, and constraints if necessary.
* Feel free to expand this schema by adding more tables related to hashtags, mentions, or additional user attributes as needed!

#### Users Table:

* Store information about users (profiles).
* Include columns for user ID, username, follower count, following count, and post count.

|  |
| --- |
| CREATE TABLE Users (  UserID INT PRIMARY KEY,  Username VARCHAR(50) NOT NULL,  Followers INT,  Following INT,  Posts INT  ); |

#### Posts Table:

* Record details about individual posts.
* Columns include post ID, user ID (who posted it), timestamp, caption, and media type (photo, video, etc.).

|  |
| --- |
| CREATE TABLE Posts (  PostID INT PRIMARY KEY,  UserID INT,  Timestamp DATETIME,  Caption TEXT,  MediaType VARCHAR(20),  FOREIGN KEY (UserID) REFERENCES Users(UserID)  ); |

#### Likes Table:

* Track likes given to posts.
* Columns include like ID, user ID (who liked), and post ID (which post was liked).

|  |
| --- |
| CREATE TABLE Likes (  LikeID INT PRIMARY KEY,  UserID INT,  PostID INT,  FOREIGN KEY (UserID) REFERENCES Users(UserID),  FOREIGN KEY (PostID) REFERENCES Posts(PostID)  ); |

#### Comments Table:

* Store comments made on posts.
* Columns include comment ID, user ID (who commented), post ID (which post received the comment), and the comment text.

|  |
| --- |
| CREATE TABLE Comments (  CommentID INT PRIMARY KEY,  UserID INT,  PostID INT,  CommentText TEXT,  FOREIGN KEY (UserID) REFERENCES Users(UserID),  FOREIGN KEY (PostID) REFERENCES Posts(PostID)  ); |

### Requirements

* + Design queries to analyze user engagement, trending topics, and influential users.
  + Explore aggregate functions to calculate average likes per post or most active users.

## Healthcare System:

* Model a healthcare system with tables for patients, doctors, appointments, and medical records.

### Tables

* + These tables provide a foundation for managing healthcare-related data within a hospital system.
  + Feel free to adjust the column names, data types, and constraints based on your specific requirements.
  + Feel free to expand this schema by adding more tables related to appointments, medications, or other relevant aspects of healthcare management!

#### Physicians Table:

* Stores information about physicians.
* Includes columns for physician ID, name, position, social security number (SSN), and department ID.

|  |
| --- |
| CREATE TABLE Physicians (  EmployeeID INT PRIMARY KEY,  Name VARCHAR(100) NOT NULL,  Position VARCHAR(50),  SSN VARCHAR(11) UNIQUE,  DepartmentID INT,  FOREIGN KEY (DepartmentID) REFERENCES Departments(DepartmentID)  ); |

#### Departments Table:

* Contains details about hospital departments.
* Columns include department ID and department name.

|  |
| --- |
| CREATE TABLE Departments (  DepartmentID INT PRIMARY KEY,  DepartmentName VARCHAR(100) NOT NULL  ); |

#### Medical Procedures Table:

* Records information about various medical procedures.
* Columns include procedure code, name, and cost.

|  |
| --- |
| CREATE TABLE MedicalProcedures (  ProcedureCode VARCHAR(10) PRIMARY KEY,  ProcedureName VARCHAR(100) NOT NULL,  Cost DECIMAL(10, 2)  ); |

#### Certifications Table:

* Tracks certifications for physicians.
* Columns include physician ID, procedure code, certification start date, and expiration date.

|  |
| --- |
| CREATE TABLE Certifications (  EmployeeID INT,  ProcedureCode VARCHAR(10),  CertificationStartDate DATE,  CertificationExpirationDate DATE,  PRIMARY KEY (EmployeeID, ProcedureCode),  FOREIGN KEY (EmployeeID) REFERENCES Physicians(EmployeeID),  FOREIGN KEY (ProcedureCode) REFERENCES MedicalProcedures(ProcedureCode)  ); |

#### Patients Table:

* Stores patient information.
* Columns include patient SSN, name, address, phone number, insurance ID, and primary care physician (PCP).

|  |
| --- |
| CREATE TABLE Patients (  SSN VARCHAR(11) PRIMARY KEY,  Name VARCHAR(100) NOT NULL,  Address VARCHAR(200),  Phone VARCHAR(15),  InsuranceID VARCHAR(20),  PCP INT,  FOREIGN KEY (PCP) REFERENCES Physicians(EmployeeID)  ); |

#### Nurses Table:

* Contains details about registered nurses.
* Columns include nurse ID, name, position, and registration status.

|  |
| --- |
| CREATE TABLE Nurses (  EmployeeID INT PRIMARY KEY,  Name VARCHAR(100) NOT NULL,  Position VARCHAR(50),  Registered BOOLEAN  ); |

### Requirements

* + Write queries to find patient demographics, appointment schedules, and common diagnoses.
  + Utilize stored procedures to automate tasks like appointment reminders.

## Blood Donation Management System:

* + Create a database to store information about blood donors, recipients, and blood banks.

### Tables

* + These tables provide a foundation for managing blood donations data within a blood bank system.
  + Feel free to adjust the column names, data types, and constraints if you see necessary.
  + Feel free to expand this schema by adding more tables related to appointments, medications, or other relevant aspects of healthcare management!

#### ****Donor Table****:

* The Donor table stores information about blood donors.

|  |
| --- |
| CREATE TABLE Donor (  donor\_id INT PRIMARY KEY,  donor\_complete\_name VARCHAR(50) NOT NULL,  blood\_group VARCHAR(3) NOT NULL,  contact\_number VARCHAR(15) NOT NULL,  email\_address VARCHAR(100),  registration\_date DATE  ); |

#### ****Recipient Table****:

* The Recipient table holds details about blood recipients.

|  |
| --- |
| CREATE TABLE Recipient (  recipient\_id INT PRIMARY KEY,  recipient\_name VARCHAR(50) NOT NULL,  blood\_group VARCHAR(3) NOT NULL,  contact\_number VARCHAR(15) NOT NULL,  address VARCHAR(100) NOT NULL  ); |

#### ****Blood Bank Table****:

* The BloodBank table manages blood bank information.

|  |
| --- |
| CREATE TABLE BloodBank (  bank\_id INT PRIMARY KEY,  bank\_name VARCHAR(100) NOT NULL,  location VARCHAR(100) NOT NULL,  contact\_number VARCHAR(15) NOT NULL  ); |

#### ****Donation Table****:

* The Donation table tracks blood donations.

|  |
| --- |
| CREATE TABLE Donation (  donation\_id INT PRIMARY KEY,  donor\_id INT NOT NULL,  bank\_id INT NOT NULL,  donation\_date DATE NOT NULL,  donation\_amount DECIMAL(5, 2) NOT NULL,  FOREIGN KEY (donor\_id) REFERENCES Donor(donor\_id),  FOREIGN KEY (bank\_id) REFERENCES BloodBank(bank\_id)  ); |

#### ****Inventory Table****:

* The Inventory table manages blood inventory.

|  |
| --- |
| CREATE TABLE Inventory (  inventory\_id INT PRIMARY KEY,  bank\_id INT NOT NULL,  blood\_group VARCHAR(3) NOT NULL,  quantity INT NOT NULL,  FOREIGN KEY (bank\_id) REFERENCES BloodBank(bank\_id)  ); |

### Requirements:

* + Implement queries to track blood donations, match donors with recipients, and manage inventory.

## Cooking Recipe Website:

* + Design a database for storing recipes, ingredients, and user ratings.
  + Write queries to retrieve recipes based on ingredients, cuisine, and difficulty level.

## Voice Commands Transport Enquiry System:

* + Develop a database for public transportation information.
  + Implement queries to retrieve routes, schedules, and real-time updates based on voice commands.

## Carbon-Emission Calculator:

* + Create a database to store emissions data for vehicles, factories, and households.
  + Write queries to calculate carbon footprints, compare emissions, and suggest eco-friendly alternatives.

## Railway Control System Database:

* + Design a database for managing train schedules, stations, and ticket bookings.
  + Develop queries to handle seat availability, reservations, and cancellations.

## Analyze Data in a Model Car Database:

* + Use a sample car database to practice complex queries.
  + Explore joins, subqueries, and aggregation to analyze car sales, models, and performance.