MYSQL Assignment 1 – DDL commands & Constrains

DDL Commands

1. **Table Creation (CREATE):** Write the SQL statements to create a database named "employee" and the following tables based on the provided schema:

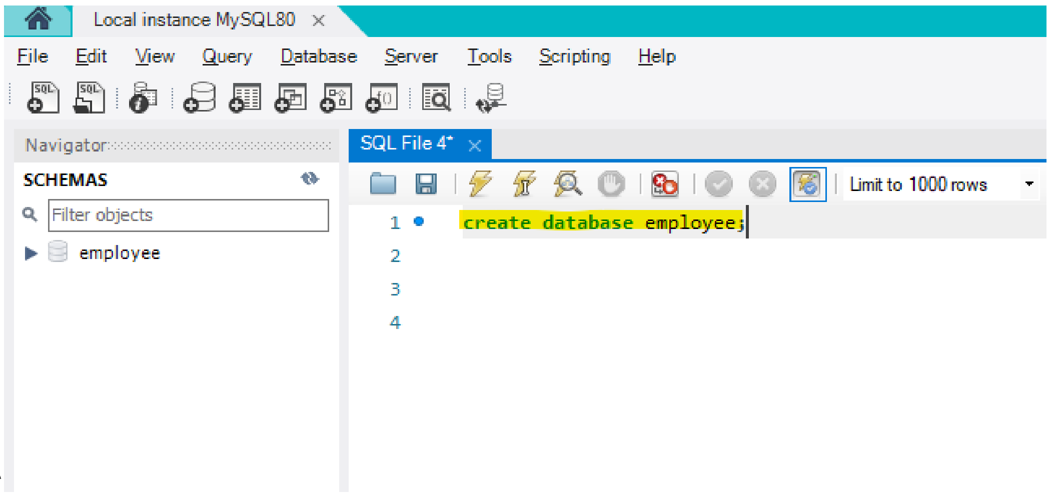
   Departments

   Location

   Employees

- **Database**

create database employee;



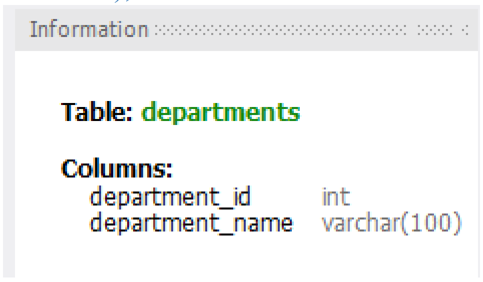
```
create database employee;
```

`

- **Tables**

  Departments

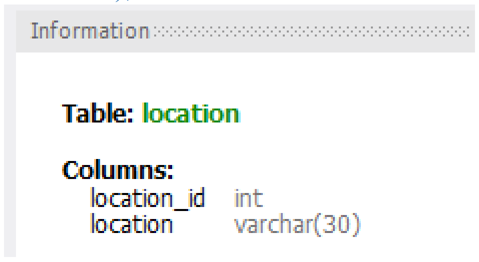
```
create table Departments (
        department_id INT,
        department_name VARCHAR(100)
  );
```



```
Information

Table: departments

Columns:
  department_id      int
  department_name    varchar(100)
```

- **Location**

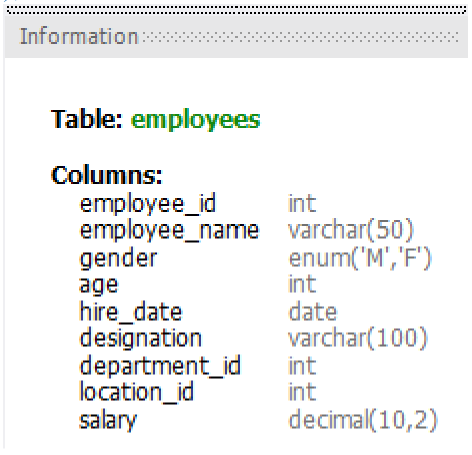
```
create table Location (
            location_id INT,
             location VARCHAR(30)
              );
```



```
Information

Table: location

Columns:
    location_id   int
    location      varchar(30)
```

- **Employees**

```
create table Employees (
            employee_id INT,
            employee_name VARCHAR(50),
            gender ENUM('M','F'),
            age INT,
            hire_date DATE,
            designation VARCHAR(100),
            department_id INT,
            location_id INT,
            salary DECIMAL(10,2)
);
```

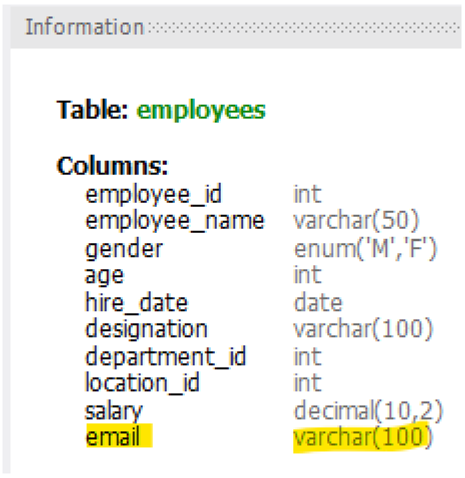


```
Information

Table: employees

Columns:
    employee_id      int
    employee_name    varchar(50)
    gender           enum('M','F')
    age              int
    hire_date        date
    designation      varchar(100)
    department_id    int
    location_id      int
    salary           decimal(10,2)
```

2. **Table Alteration (ALTER):** Consider the following scenarios and write the SQL statements to alter the structure of the tables accordingly:

   - Add a new column named "email" to the Employees table to store employee email addresses.

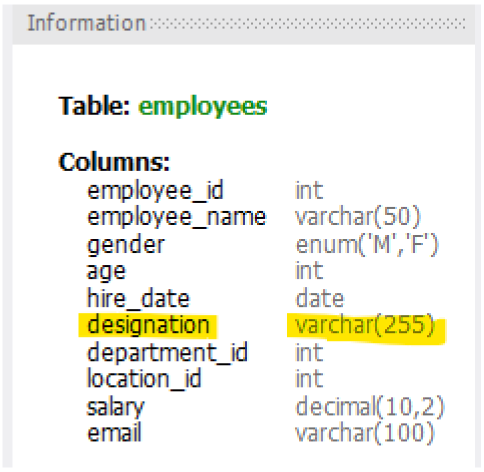
     ```
     Alter table Employees
     ADD email VARCHAR(100);
     ```



```
Information

Table: employees

Columns:
   employee_id      int
   employee_name    varchar(50)
   gender           enum('M','F')
   age              int
   hire_date        date
   designation      varchar(100)
   department_id    int
   location_id      int
   salary           decimal(10,2)
   email            varchar(100)
```

   - Modify the data type of the "designation" column in the Employees table to support a wider range of values.

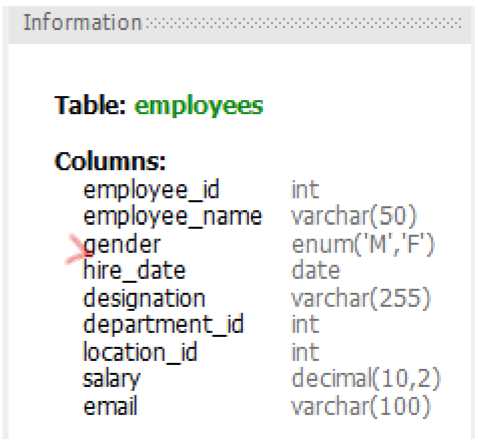
     ```
     alter table employees
     modify designation VARCHAR(255);
     ```



```
Information

Table: employees

Columns:
   employee_id      int
   employee_name    varchar(50)
   gender           enum('M','F')
   age              int
   hire_date        date
   designation      varchar(255)
   department_id    int
   location_id      int
   salary           decimal(10,2)
   email            varchar(100)
```

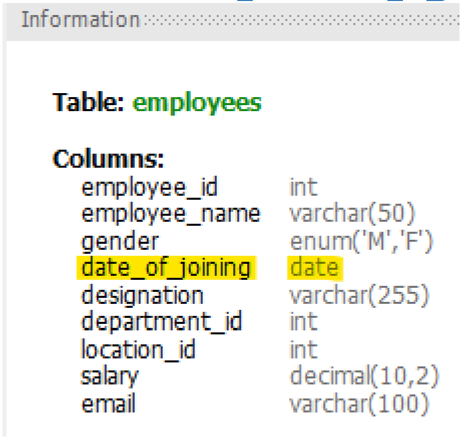
- Drop the “age” column from the Employees table.

  alter table employees
  drop age;



```
Information

Table: employees

Columns:
    employee_id        int
    employee_name      varchar(50)
    gender             enum('M','F')
    hire_date          date
    designation        varchar(255)
    department_id      int
    location_id        int
    salary             decimal(10,2)
    email              varchar(100)
```

- Rename the “hire_date” column to “date_of_joining”.
  alter table employees
  rename column hire_date to date_of_joining;

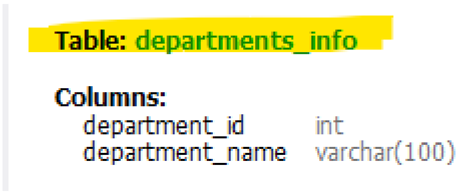


```
Information

Table: employees

Columns:
    employee_id        int
    employee_name      varchar(50)
    gender             enum('M','F')
    date_of_joining    date
    designation        varchar(255)
    department_id      int
    location_id        int
    salary             decimal(10,2)
    email              varchar(100)
```

3. **Table Renaming (RENAME):** Rewrite the SQL statements to rename the following tables:

   - Rename the "Departments" table to "Departments_Info".

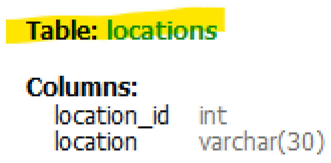
     rename table departments to departments_Info;



```
Table: departments_info

Columns:
  department_id      int
  department_name    varchar(100)
```

   - Rename the "Location" table to "Locations".

     rename table location to locations;



```
Table: locations

Columns:
  location_id   int
  location      varchar(30)
```

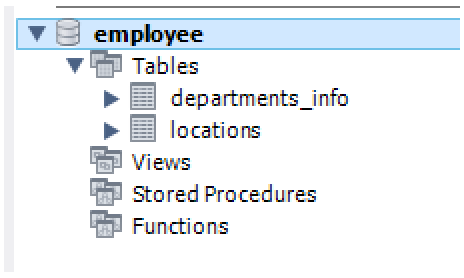
4. **Table Truncation (TRUNCATE):** Write an SQL statement to truncate the Employees table.

   truncate employees;

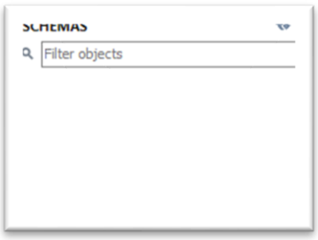
   (It will erase the data of the table only not the table structure)

5. **Database & Table Dropping (DROP):** Write the SQL statements to drop the Employees table and then the “employee” database.

drop table employees;



drop database employee;



Constraints

1. Database Recreation

   Drop the 'employee' database if it exists and recreate it using the provided schema, ensuring that all tables are created with the appropriate constraints as instructed.

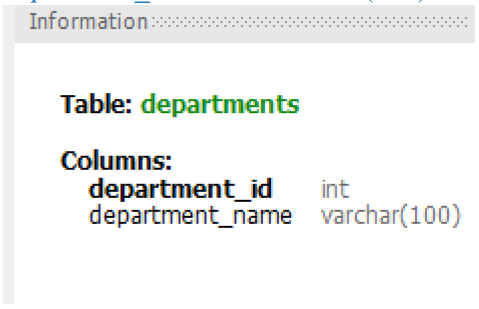
2. Departments Table

   - Ensure that the "department_id" uniquely identifies each department.

     department_id INT Unique,

   - Set up constraints on the "department_name" to avoid duplicate and null entries.

     department_name VARCHAR(100) Unique NotNull



```
Information

Table: departments

Columns:
   department_id      int
   department_name    varchar(100)
```

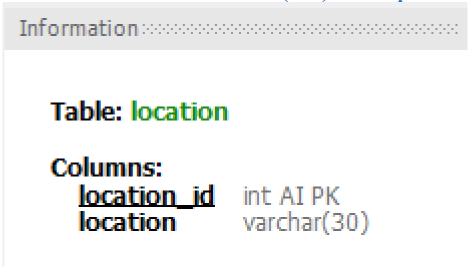
3. Location Table

   - Establish a mechanism to automatically generate unique identifiers for each location, ensuring that they are incremented sequentially.

```
CREATE TABLE Locations (
   location_id INT AUTO_INCREMENT PRIMARY KEY,
);
```

   - Implement constraints to prevent the insertion of null and duplicate locations.

     location VARCHAR(30) Unique Not Null ;



```
Information

Table: location

Columns:
   location_id   int AI PK
   location      varchar(30)
```

4. Employees Table

- Guarantee that each employee has a distinct identifier.
- Create a restriction to ensure that the employee's name is always provided.
- Limit the acceptable values for the "gender" field to only 'M' or 'F'.
- Enforce a condition to ensure that the employee's age is 18 or above.
- Automatically assign the current date to the "hire_date" field if not specified.
- Establish links between the "department_id" and "location_id" fields in the "employees" table and their respective tables.

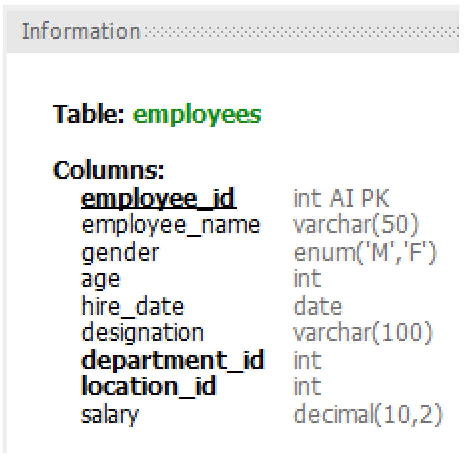


Table: employees

Columns:

employee_id	int AI PK
employee_name	varchar(50)
gender	enum('M','F')
age	int
hire_date	date
designation	varchar(100)
department_id	int
location_id	int
salary	decimal(10,2)

```
CREATE TABLE Employees (
   employee_id INT AUTO_INCREMENT PRIMARY KEY,
   employee_name VARCHAR(50) NOT NULL,
   gender ENUM('M','F') NOT NULL,
   age INT CHECK (age >= 18),
   hire_date DATE DEFAULT (CURRENT_DATE),
   designation VARCHAR(100),
   department_id INT,
   location_id INT,
   salary DECIMAL(10,2),

   FOREIGN KEY (department_id) REFERENCES
   Departments(department_id),
   FOREIGN KEY (location_id) REFERENCES Locations(location_id)
);
```