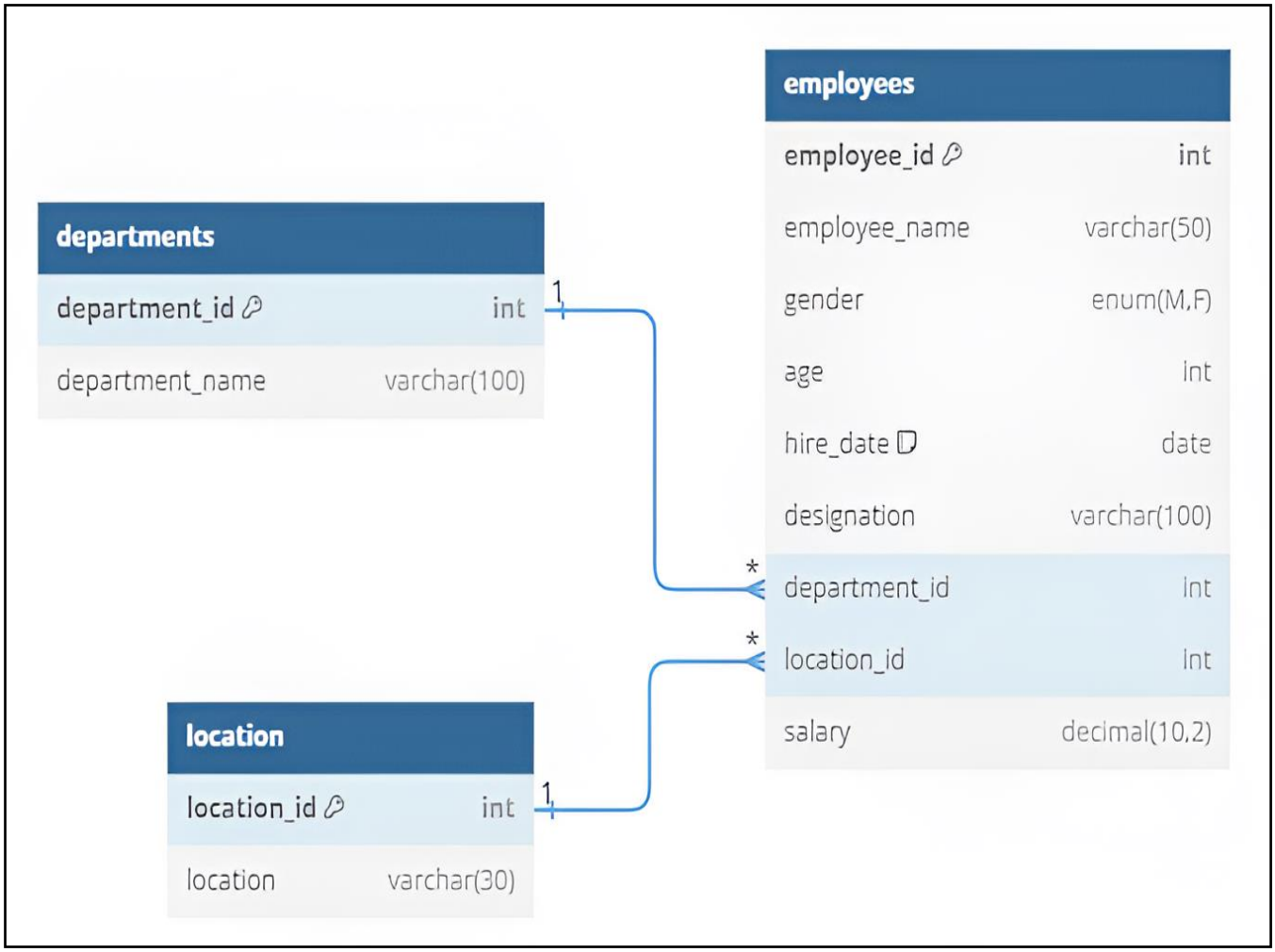
MINI CAPSTONE PROJECT - MYSQL

DDL Commands;

EMPLOYEE DATABASE SCHEMA:



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

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.

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

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

◉ Rename the “hire_date” column to “date_of_joining”.

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

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

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

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

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

DDL - CONSTRAINTS:

1. Database Recreation:

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

2. Departments Table:

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

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

3. Location Table:

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

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

4. Employees Table:

◉ Guarantee that each employee has a distinct identifier.

◉ Create a restriction to ensure that the employee's name is always provided.

◉ 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.

DML COMMANDS:

1. **Insertion (INSERT):** Insert the records into the employees database respective tables: Departments, Location, and Employees by referring to the tables provided. Ensure that each record is inserted correctly with all the required fields filled.

LOCATION TABLE:

location_id	location
1	Chennai
2	Bangalore
3	Hyderabad
4	Pune

DEPARTMENTS TABLE:

department_id	department_name
1	Software Development
2	Marketing
3	Data Science
4	Human Resources
5	Product Management
6	Content Creation
7	Finance
8	Design
9	Research and Development

10	Customer Support
11	Business Development
12	IT
13	Operations

EXPLOYEES TABLE:

INSERT INTO employees (employee_id, employee_name, gender, age, hire_date, designation, department_id, location_id, salary) VALUES

(5001, 'Vihaan Singh', 'M', 27, '2015-01-20', 'Data Analyst', 3, 4, 60000),

(5002, 'Reyansh Singh', 'M', 31, '2015-03-10', 'Network Engineer', 12, 1, 80000),

(5003, 'Aaradhya Iyer', 'F', 26, '2015-05-20', 'Customer Support Executive', 10, 2, 45000),

(5004, 'Kiara Malhotra', 'F', 29, '2015-07-05', NULL, 8, 3, 70000),

(5005, 'Anvi Chaudhary', 'F', 25, '2015-09-11', 'Business Development Executive', 11, 1, 55000),

(5006, 'Dhruv Shetty', 'M', 28, '2015-11-20', 'UI Developer', 8, 2, 65000),

(5007, 'Anushka Singh', 'F', 32, '2016-01-15', 'Marketing Manager', 2, 3, 90000),

(5008, 'Diya Jha', 'F', 27, '2016-03-05', 'Graphic Designer', 8, 4, 70000),

(5009, 'Kiaan Desai', 'M', 30, '2016-05-20', 'Sales Executive', 11, 3, 55000),

(5010, 'Atharv Yadav', 'M', 29, '2016-07-10', 'Systems Administrator', 12, 4, 80000),

(5011, 'Saanvi Patel', 'F', 28, '2016-09-20', 'Marketing Analyst', 2, 1, 60000),

(5012, 'Myra Verma', 'F', 26, '2016-11-05', 'Operations Manager', 13, 2, 95000),

(5013, 'Arnav Rao', 'M', 33, '2017-01-20', 'Customer Success Manager', 10, 3, 75000),

(5014, 'Vihaan Mohan', 'M', 30, '2017-03-10', 'Supply Chain Analyst', 10, 2, 60000),

(5015, 'Ishaan Kumar', 'M', 27, '2017-05-20', 'Financial Analyst', 7, 1, 85000),

(5016, 'Zoya Khan', 'F', 31, '2017-07-05', 'Legal Counsel', 4, 4, 100000),

(5017, 'Kabir Nair', 'M', 28, '2017-09-11', 'IT Support Specialist', 12, 2, 80000),

(5018, 'Ishan Mishra', 'M', 25, '2017-11-20', 'Research Scientist', 9, 3, 75000),

(5019, 'Ishika Patel', 'F', 29, '2018-01-15', 'Talent Acquisition Specialist', 4, 4, 55000),

(5020, 'Aarav Nair', 'M', 32, '2018-03-05', 'Software Engineer', 1, 1, 90000),

(5021, 'Advik Kapoor', 'M', 26, '2018-05-20', 'Finance Analyst', 7, 3, 85000),

(5022, 'Aadhya Iyengar', 'F', 28, '2018-07-10', 'HR Specialist', 4, 4, 60000),

(5023, 'Anika Paul', 'F', 30, '2018-09-20', 'Public Relations Specialist', 2, 2, 70000),

(5024, 'Aryan Shetty', 'M', 27, '2018-11-05', 'Product Manager', 5, 1, 95000),

(5025, 'Avni Iyengar', 'F', 31, '2019-01-20', 'Data Scientist', 3, 4, 100000),

(5026, 'Vivaan Singh', 'M', 29, '2019-03-10', 'Business Analyst', 3, 2, 75000),

(5027, 'Ananya Paul', 'F', 32, '2019-05-20', 'Content Writer', 6, 3, 60000),

(5028, 'Anaya Kapoor', 'F', 26, '2019-07-05', 'Event Coordinator', 6, 1, 60000),

(5029, 'Arjun Kumar', 'M', 33, '2019-09-11', 'Quality Assurance Analyst', 12, 2, 80000),

(5030, 'Sara Iyer', 'F', 28, '2019-11-20', 'Project Manager', 5, 1, 90000) ;

2. **Selection (SELECT):** Write SQL queries to retrieve the following information:

◉ Retrieve all data from the Departments table.

◉ Display only the location name from the Location table.

◉ Display the employee names and their respective designations.

3. **Updating (UPDATE):** Update the records in the Employees table based on the following criteria:

◉ Change the designation of employee with ID 5001 to 'Senior Data Analyst'.

◉ Increase the salary of all employees in the Finance department by 10%.

4. **Deletion (DELETE):** Write SQL statements to delete the following records:

◉ Remove all records from the Employees table where the hire date is before '2017-01-01'.

◉ Delete the record of the employee with ID 5025.

Querying Data:

1. Distinct Values:

◉ Write a query to retrieve distinct salaries from the Employees table.

2. Alias (AS):

◉ Provide aliases for the "age" and "salary" columns as "Employee_Age" and "Employee_Salary", respectively.

3. Where Clause & Operators (Arithmetic, Comparison, Logical):

◉ Calculate the net salary per employee by adding a 10% bonus.

◉ Retrieve employees with a salary greater than ₹50000 and hired before 2016-01-01.

◉ List employees who are Data Analyst or Data Scientist.

4. Other Operators:

◉ Find the employee whose designation is missing and fill it with "Data Scientist".

◉ List employees whose department_id is either 1, 3, 4, 9, or 12.

◉ Identify employees whose salary is not between 50000 and 80000.

◉ Find employees whose name starts with a vowel.

◉ Display the employee names that have 'sh' as the second and third characters.

Sorting and Grouping Data:

1. **Order by:**

◉ List employees hired after January 1, 2019, ordered by hire date.

◉ Find employees sorted by department ID in ascending order and salary in descending order.

2. **Limit:**

◉ Retrieve the top 10 highest paid employees from the Employees table.

◉ Display the first 5 employees hired in the year 2018.

3. **Aggregate Functions:**

◉ Calculate the sum of all salaries in the Finance department.

◉ Find the minimum age among all employees.

4. **Group by:**

◉ List the maximum salary for each location.

◉ Calculate the average salary for each designation containing the word 'Analyst'.

5. **Having:**

◉ Find departments with less than 3 employees.

◉ Find locations with female employees whose average age is below 30.

JOINS & FUNCTIONS

Joins:

1. **Inner Join:**

◉ List employee names, their designations, and department names where employees are assigned to a department.

2. **Left Join:**

◉ List all departments along with the total number of employees in each department, including departments with no employees.

3. **Right Join:**

◉ Display all locations along with the names of employees assigned to each location. If no employees are assigned to a location, display NULL for employee name.

Functions:

1. Built-in Numeric Functions:

◉ Find the average age of employees, rounded off to the nearest integer.

◉ Retrieve all employee records including a bonus column, calculated as the square root of their salary rounded up to the nearest larger integer.

2. Built-in String Functions:

◉ Extract the first three characters from the location names.

◉ Concatenate employee names with their designations separated by a hyphen.

3. Built-in Date Time Functions:

◉ Extract the year from the hire date of all employees.

◉ Calculate the number of days between the hire date and the current date for each employee.

◉ Format the hire date of employees to display in 'DD-MM-YYYY' format.

◉ Find the 'employment_confirmation' date of each employee which is 3 months from their hire_date.

4. User-defined Functions:

◉ Define a function to retrieve the employee count by location name. Find the employee count for the cities Bangalore and Hyderabad..

SUBQUERIES & STORED PROCEDURE

Subqueries:

1. Single-Row Subqueries:

◉ Retrieve the details of employees with salaries greater than the average salary of all employees.

◉ List the employee(s) with the highest salary.

◉ Retrieve the details of employees who are working in the same department as the employee Arjun Kumar.

2. Multi-Row Subqueries:

◉ Find the employees who work in departments with 'Development' in their name.

Stored Procedure:

1. Stored Procedure with IN Parameter:

◉ Define a stored procedure named GetEmployeesByDepartmentName that takes a department name as input and retrieves employees belonging to that department. Retrieve employee details of the departments 'IT' and 'Human Resources'.

2. Stored Procedure with OUT Parameter:

◉ Create a stored procedure named FindEmployeeBirthYearByID that takes an employee ID as input and returns the employee's birth year. Retrieve the birth year of employees with the IDs 5004, 5018 and 5029.

BEFORE TRIGGER & AFTER TRIGGER

1. Before Insert Trigger:

◉ Create a before insert trigger that automatically sets the salary of an employee to the average of all salaries if it's not provided during insertion. Test the trigger by attempting to insert a new employee without providing a salary.

2. After Delete Trigger:

◉ Create an after-delete trigger to prevent the deletion of employees who joined in the most recent year. Test the trigger by attempting to delete an employee who joined in the latest year.