**SQL CODE CHALLENGE CAREER HUB**

1. **Provide a SQL script that initializes the database for the Job Board scenario “CareerHub”.**

4**. Ensure the script handles potential errors, such as if the database or tables already exist**

DROP DATABASE IF EXISTS CareerHub;

create database careerhub;

use careerhub;

1. **Create tables for Companies, Jobs, Applicants and Applications.**
2. **Define appropriate primary keys, foreign keys, and constraints.**

CREATE TABLE Companies (

CompanyID INT PRIMARY KEY AUTO\_INCREMENT,

CompanyName VARCHAR(255) NOT NULL,

Location VARCHAR(255) NOT NULL

);

CREATE TABLE Jobs (

JobID INT PRIMARY KEY AUTO\_INCREMENT,

CompanyID INT,

JobTitle VARCHAR(255) NOT NULL,

JobDescription TEXT,

JobLocation VARCHAR(255) NOT NULL,

Salary DECIMAL(10,2) CHECK (Salary >= 0),

JobType VARCHAR(50) NOT NULL,

PostedDate DATETIME DEFAULT CURRENT\_TIMESTAMP,

CONSTRAINT fk\_customer FOREIGN KEY (CompanyID)

REFERENCES Companies(CompanyID) ON DELETE CASCADE

);

CREATE TABLE Applicants (

ApplicantID INT PRIMARY KEY AUTO\_INCREMENT,

FirstName VARCHAR(100) NOT NULL,

LastName VARCHAR(100) NOT NULL,

Email VARCHAR(255) UNIQUE NOT NULL,

Phone VARCHAR(20) UNIQUE NOT NULL,

Resume\_CV TEXT

);

CREATE TABLE Applications (

ApplicationID INT PRIMARY KEY AUTO\_INCREMENT,

JobID INT,

ApplicantID INT,

ApplicationDate DATETIME DEFAULT CURRENT\_TIMESTAMP,

CoverLetter TEXT,

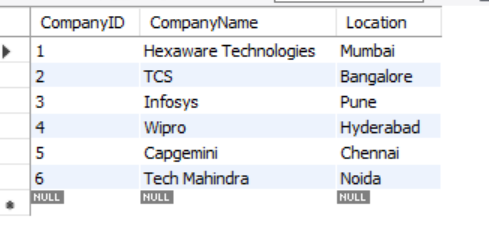
CONSTRAINT fk\_Apps FOREIGN KEY (JobID)

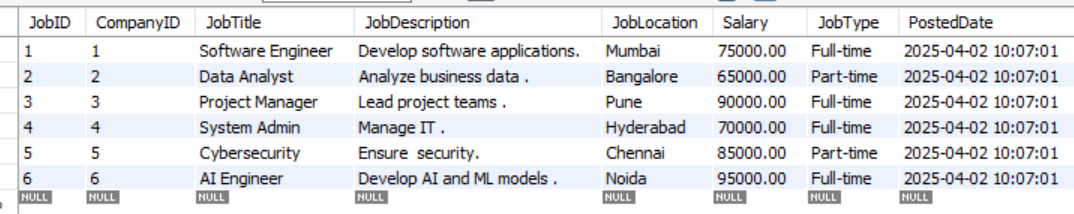
REFERENCES Jobs(JobID) ON DELETE CASCADE,

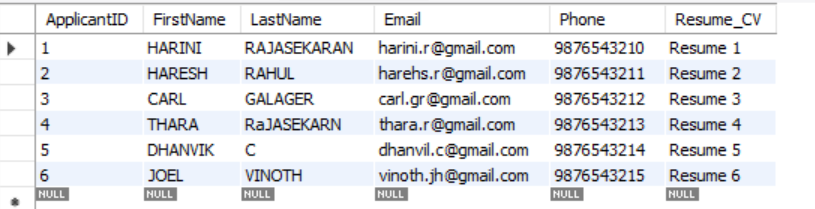
CONSTRAINT fk\_applications FOREIGN KEY (ApplicantID)

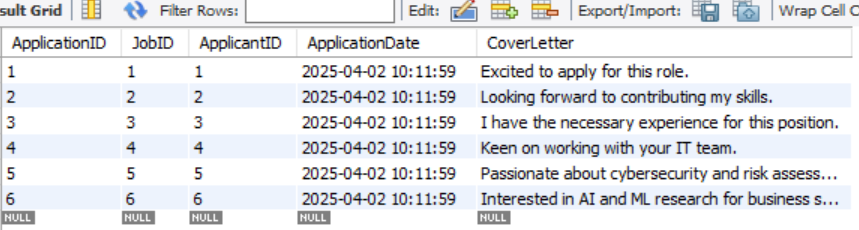
REFERENCES Applicants(ApplicantID) ON DELETE CASCADE

);









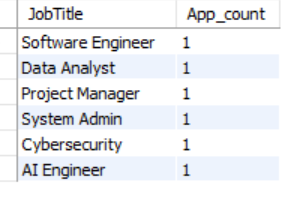
**5.Write an SQL query to count the number of applications received for each job listing in the "Jobs" table. Display the job title and the corresponding application count. Ensure that it lists all jobs, even if they have no applications.**

SELECT j.JobTitle, COUNT(a.ApplicationID) AS App\_count

FROM Jobs j

LEFT JOIN Applications a ON j.JobID = a.JobID

GROUP BY j.JobID, j.JobTitle;



**6. Develop an SQL query that retrieves job listings from the "Jobs" table within a specified salary range. Allow parameters for the minimum and maximum salary values. Display the job title, company name, location, and salary for each matching job.**

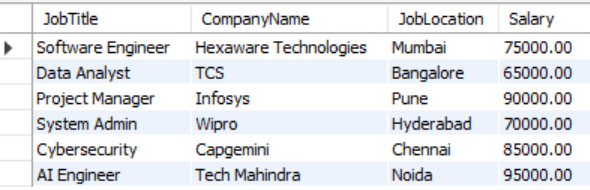
SELECT j.JobTitle, c.CompanyName, j.JobLocation, j.Salary

FROM Jobs j

JOIN Companies c ON j.CompanyID = c.CompanyID

WHERE j.Salary BETWEEN

(SELECT MIN(Salary) FROM Jobs) AND (SELECT MAX(Salary) FROM Jobs);



**7. Write an SQL query that retrieves the job application history for a specific applicant. Allow a parameter for the ApplicantID, and return a result set with the job titles, company names, and application dates for all the jobs the applicant has applied to.**

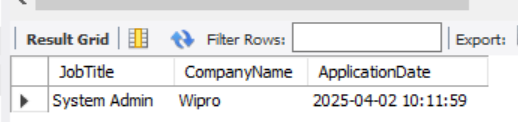
SELECT j.JobTitle, c.CompanyName, a.ApplicationDate

FROM Applications a

JOIN Jobs j ON a.JobID = j.JobID

JOIN Companies c ON j.CompanyID = c.CompanyID

WHERE a.ApplicantID = 4;

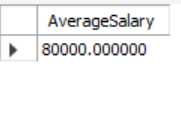


**8. Create an SQL query that calculates and displays the average salary offered by all companies for job listings in the "Jobs" table. Ensure that the query filters out jobs with a salary of zero.**

SELECT AVG(Salary) AS AverageSalary

FROM Jobs

WHERE Salary > 0;

****

**10. Find the applicants who have applied for positions in companies located in 'CityX' and have at least 3 years of experience.**

ALTER TABLE Applicants ADD Experience INT CHECK (Experience >= 0);

SET SQL\_SAFE\_UPDATES = 0;

UPDATE Applicants

SET Experience = CASE

WHEN ApplicantID = 1 THEN 2

WHEN ApplicantID = 2 THEN 5

WHEN ApplicantID = 3 THEN 3

WHEN ApplicantID = 4 THEN 1

WHEN ApplicantID = 5 THEN 4

WHEN ApplicantID = 6 THEN 6

END;

SET SQL\_SAFE\_UPDATES = 1;

SELECT a.\*

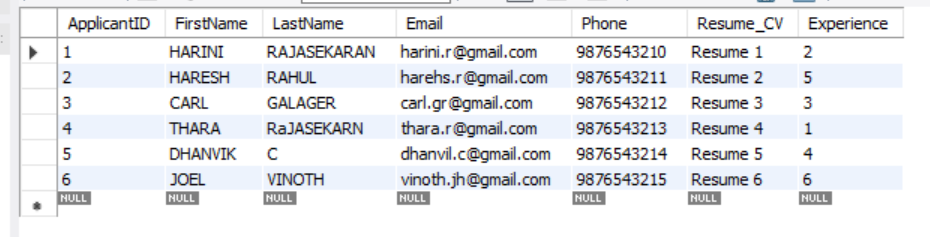
FROM Applicants a

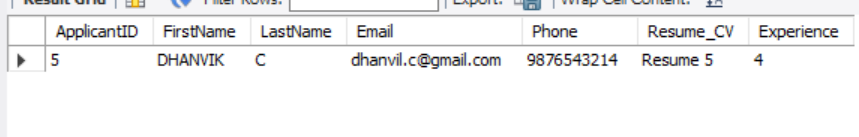
JOIN Applications app ON a.ApplicantID = app.ApplicantID

JOIN Jobs j ON app.JobID = j.JobID

JOIN Companies c ON j.CompanyID = c.CompanyID

WHERE c.Location = 'Chennai' AND a.Experience >= 3;



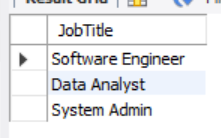


**11. Retrieve a list of distinct job titles with salaries between $60,000 and $80,000.**

SELECT DISTINCT JobTitle

FROM Jobs

WHERE Salary BETWEEN 60000 AND 80000;



**12. Find the jobs that have not received any applications.**

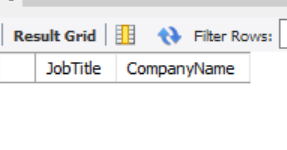
SELECT j.JobTitle, c.CompanyName

FROM Jobs j

JOIN Companies c ON j.CompanyID = c.CompanyID

LEFT JOIN Applications a ON j.JobID = a.JobID

WHERE a.ApplicationID IS NULL;



**13. Retrieve a list of job applicants along with the companies they have applied to and the positions they have applied for.**

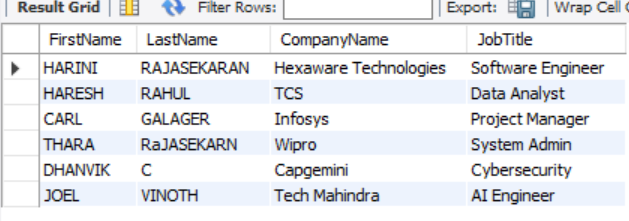
SELECT a.FirstName, a.LastName, c.CompanyName, j.JobTitle

FROM Applications app

JOIN Applicants a ON app.ApplicantID = a.ApplicantID

JOIN Jobs j ON app.JobID = j.JobID

JOIN Companies c ON j.CompanyID = c.CompanyID;



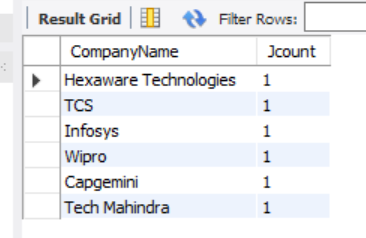
**14. Retrieve a list of companies along with the count of jobs they have posted, even if they have not received any applications.**

SELECT c.CompanyName, COUNT(j.JobID) AS Jcount

FROM Companies c

LEFT JOIN Jobs j ON c.CompanyID = j.CompanyID

GROUP BY c.CompanyID, c.CompanyName;



**15. List all applicants along with the companies and positions they have applied for, including those who have not applied.**

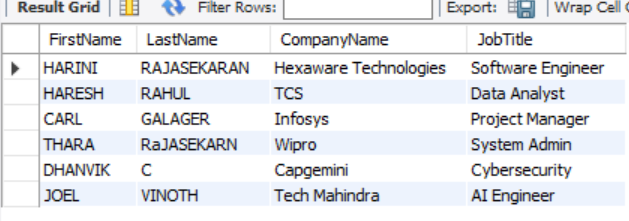
SELECT a.FirstName, a.LastName, c.CompanyName, j.JobTitle

FROM Applicants a

LEFT JOIN Applications app ON a.ApplicantID = app.ApplicantID

LEFT JOIN Jobs j ON app.JobID = j.JobID

LEFT JOIN Companies c ON j.CompanyID = c.CompanyID;



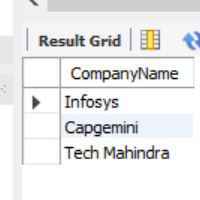
**16. Find companies that have posted jobs with a salary higher than the average salary of all jobs.**

SELECT DISTINCT c.CompanyName

FROM Companies c

JOIN Jobs j ON c.CompanyID = j.CompanyID

WHERE j.Salary > (SELECT AVG(Salary) FROM Jobs WHERE Salary > 0);



**17. Display a list of applicants with their names and a concatenated string of their city and state.**

ALTER TABLE Applicants

ADD COLUMN City VARCHAR(100) NOT NULL,

ADD COLUMN State VARCHAR(100) NOT NULL;

SET SQL\_SAFE\_UPDATES = 0;

UPDATE Applicants

SET City = CASE

WHEN ApplicantID = 1 THEN 'Mumbai'

WHEN ApplicantID = 2 THEN 'Bangalore'

WHEN ApplicantID = 3 THEN 'Pune'

WHEN ApplicantID = 4 THEN 'Hyderabad'

WHEN ApplicantID = 5 THEN 'Chennai'

WHEN ApplicantID = 6 THEN 'Noida'

END,

State = CASE

WHEN ApplicantID = 1 THEN 'Maharashtra'

WHEN ApplicantID = 2 THEN 'Karnataka'

WHEN ApplicantID = 3 THEN 'Maharashtra'

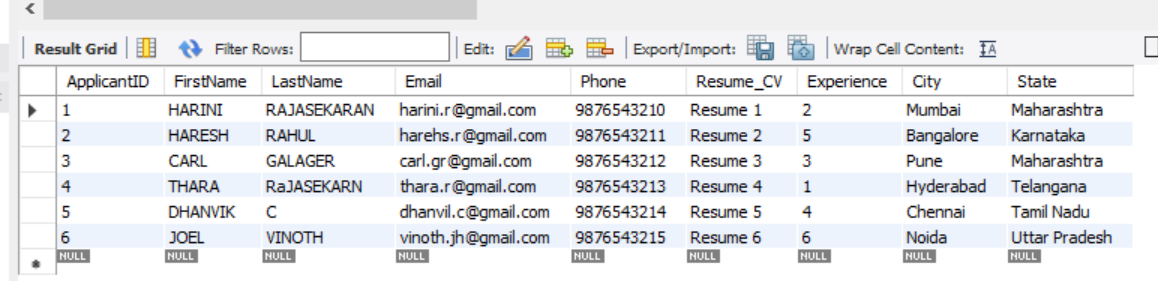
WHEN ApplicantID = 4 THEN 'Telangana'

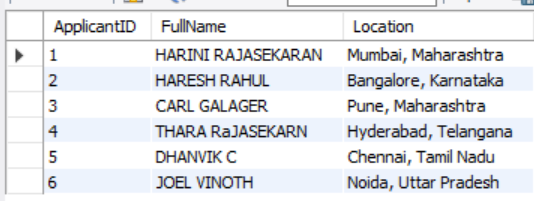
WHEN ApplicantID = 5 THEN 'Tamil Nadu'

WHEN ApplicantID = 6 THEN 'Uttar Pradesh'

END

SET SQL\_SAFE\_UPDATES = 1;



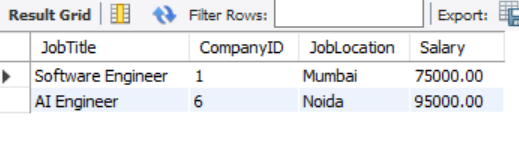


**18. Retrieve a list of jobs with titles containing either 'Developer' or 'Engineer'.**

SELECT JobTitle, CompanyID, JobLocation, Salary

FROM Jobs

WHERE JobTitle LIKE '%Developer%' OR JobTitle LIKE '%Engineer%';



**20. List all combinations of applicants and companies where the company is in a specific city and the applicant has more than 2 years of experience. For example: city=Chennai**

SELECT a.FirstName, a.LastName, c.CompanyName, c.Location

FROM Applicants a

CROSS JOIN Companies c

WHERE c.Location = 'Chennai' AND a.Experience > 2;

