**Coding Challenges: CareerHub, The Job Board**

**(ANSWERS)**

**-BY GETSY JACINTH**

**INPUTS GIVEN:**

INSERT INTO Companies (CompanyName, Location) VALUES

('Hexaware', 'Mumbai'),

('Infosys', 'Bangalore'),

('Wipro', 'Hyderabad'),

('Cognizant', 'Chennai'),

('Accenture', 'Pune'),

('Google', 'Hyderabad');

INSERT INTO Jobs (CompanyID, JobTitle, JobDescription, JobLocation, Salary, JobType, PostedDate) VALUES

(1, 'Software Developer', 'Develop enterprise apps', 'Mumbai', 65000.00, 'Full-time', NOW()),

(1, 'Backend Developer', 'Work on APIs and databases', 'Mumbai', 72000.00, 'Full-time', NOW()),

(2, 'Data Analyst', 'Business data analysis', 'Bangalore', 70000.00, 'Contract', NOW()),

(2, 'ML Engineer', 'Build AI models', 'Bangalore', 120000.00, 'Full-time', NOW()),

(3, 'QA Engineer', 'Testing software', 'Hyderabad', 58000.00, 'Part-time', NOW()),

(3, 'Support Engineer', 'Customer support role', 'Hyderabad', 0.00, 'Full-time', NOW()),

(4, 'DevOps Engineer', 'Deploy pipelines', 'Chennai', 95000.00, 'Full-time', NOW()),

(4, 'Frontend Developer', 'React UI Development', 'Chennai', 80000.00, 'Contract', NOW()),

(5, 'Business Analyst', 'Stakeholder interaction', 'Pune', 73000.00, 'Full-time', NOW()),

(5, 'Network Engineer', 'Infrastructure support', 'Pune', 60000.00, 'Part-time', NOW()),

(6, 'Cloud Engineer', 'Deploy cloud infra', 'Hyderabad', 110000.00, 'Full-time', NOW()),

(6, 'Software Engineer', 'Work on core systems', 'Hyderabad', 130000.00, 'Full-time', NOW());

INSERT INTO Applicants (FirstName, LastName, Email, Phone, Resume) VALUES

('Ravi', 'Kumar', 'ravi@example.com', '9999999999', 'Resume of Ravi'),

('Sneha', 'Sharma', 'sneha@example.com', '8888888888', 'Resume of Sneha'),

('Amit', 'Verma', 'amit@example.com', '7777777777', 'Resume of Amit'),

('Priya', 'Nair', 'priya@example.com', '6666666666', 'Resume of Priya'),

('Manoj', 'Desai', 'manoj@example.com', '5555555555', 'Resume of Manoj'),

('Pooja', 'Rathod', 'pooja@example.com', '4444444444', 'Resume of Pooja'),

('Karan', 'Joshi', 'karan@example.com', '3333333333', 'Resume of Karan'),

('Nikita', 'Mehra', 'nikita@example.com', '2222222222', 'Resume of Nikita');

-- Alter Applicants to add Experience, City, State

ALTER TABLE Applicants ADD Experience INT;

ALTER TABLE Applicants ADD City VARCHAR(100);

ALTER TABLE Applicants ADD State VARCHAR(100);

-- Update Applicants with Experience

UPDATE Applicants SET Experience = 4 WHERE ApplicantID = 1;

UPDATE Applicants SET Experience = 2 WHERE ApplicantID = 2;

UPDATE Applicants SET Experience = 3 WHERE ApplicantID = 3;

UPDATE Applicants SET Experience = 1 WHERE ApplicantID = 4;

UPDATE Applicants SET Experience = 5 WHERE ApplicantID = 5;

UPDATE Applicants SET Experience = 3 WHERE ApplicantID = 6;

UPDATE Applicants SET Experience = 2 WHERE ApplicantID = 7;

UPDATE Applicants SET Experience = 6 WHERE ApplicantID = 8;

-- Update Applicants with City and State

UPDATE Applicants SET City = 'Mumbai', State = 'MH' WHERE ApplicantID = 1;

UPDATE Applicants SET City = 'Bangalore', State = 'KA' WHERE ApplicantID = 2;

UPDATE Applicants SET City = 'Chennai', State = 'TN' WHERE ApplicantID = 3;

UPDATE Applicants SET City = 'Hyderabad', State = 'TS' WHERE ApplicantID = 4;

UPDATE Applicants SET City = 'Pune', State = 'MH' WHERE ApplicantID = 5;

UPDATE Applicants SET City = 'Chennai', State = 'TN' WHERE ApplicantID = 6;

UPDATE Applicants SET City = 'Hyderabad', State = 'TS' WHERE ApplicantID = 7;

UPDATE Applicants SET City = 'Mumbai', State = 'MH' WHERE ApplicantID = 8;

-- Insert into Applications

INSERT INTO Applications (JobID, ApplicantID, ApplicationDate, CoverLetter) VALUES

(1, 1, NOW(), 'I love building scalable apps.'),

(2, 1, NOW(), 'Experienced in APIs and databases.'),

(3, 2, NOW(), 'I am great at data wrangling.'),

(4, 3, NOW(), 'Passionate about AI.'),

(5, 4, NOW(), 'Good eye for detail and QA.'),

(6, 5, NOW(), 'Excellent communication skills.'),

(7, 6, NOW(), 'DevOps pipelines expert.'),

(8, 7, NOW(), 'Front-end enthusiast with ReactJS.'),

(9, 3, NOW(), 'I’m a people person.'),

(10, 2, NOW(), 'Hands-on experience with networking.');

-- Insert additional application

INSERT INTO Applications (JobID, ApplicantID, ApplicationDate, CoverLetter)

VALUES (8, 8, NOW(), 'Excited to join as Frontend Dev.');

**Tasks:**

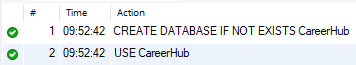
1. Provide a SQL script that initializes the database for the Job Board scenario “CareerHub”.

**QUERY:**

**CREATE DATABASE IF NOT EXISTS CareerHub;**

**USE CareerHub;**

**OUTPUT SCREENSHOT:**

****

1. Create tables for Companies, Jobs, Applicants and Applications.

**QUERY:**

**-- Companies table**

**CREATE TABLE IF NOT EXISTS Companies (**

**CompanyID INT PRIMARY KEY AUTO\_INCREMENT,**

**CompanyName VARCHAR(255) NOT NULL,**

**Location VARCHAR(255)**

**);**

**-- Jobs table**

**CREATE TABLE IF NOT EXISTS Jobs (**

**JobID INT PRIMARY KEY AUTO\_INCREMENT,**

**CompanyID INT,**

**JobTitle VARCHAR(255),**

**JobDescription TEXT,**

**JobLocation VARCHAR(255),**

**Salary DECIMAL(10, 2),**

**JobType VARCHAR(100),**

**PostedDate DATETIME DEFAULT CURRENT\_TIMESTAMP,**

**FOREIGN KEY (CompanyID) REFERENCES Companies(CompanyID)**

**);**

**-- Applicants table**

**CREATE TABLE IF NOT EXISTS Applicants (**

**ApplicantID INT PRIMARY KEY AUTO\_INCREMENT,**

**FirstName VARCHAR(100),**

**LastName VARCHAR(100),**

**Email VARCHAR(255) UNIQUE,**

**Phone VARCHAR(20),**

**Resume TEXT**

**);**

**-- Applications table**

**CREATE TABLE IF NOT EXISTS Applications (**

**ApplicationID INT PRIMARY KEY AUTO\_INCREMENT,**

**JobID INT,**

**ApplicantID INT,**

**ApplicationDate DATETIME DEFAULT CURRENT\_TIMESTAMP,**

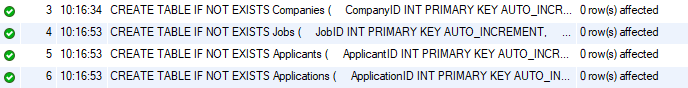
**CoverLetter TEXT,**

**FOREIGN KEY (JobID) REFERENCES Jobs(JobID),**

**FOREIGN KEY (ApplicantID) REFERENCES Applicants(ApplicantID)**

**);**

**OUTPUT SCREENSHOT:**

****

1. Define appropriate primary keys, foreign keys, and constraints.

**QUERY:**

**CREATE TABLE IF NOT EXISTS Jobs (**

**JobID INT PRIMARY KEY AUTO\_INCREMENT,**

**CompanyID INT,**

**JobTitle VARCHAR(255),**

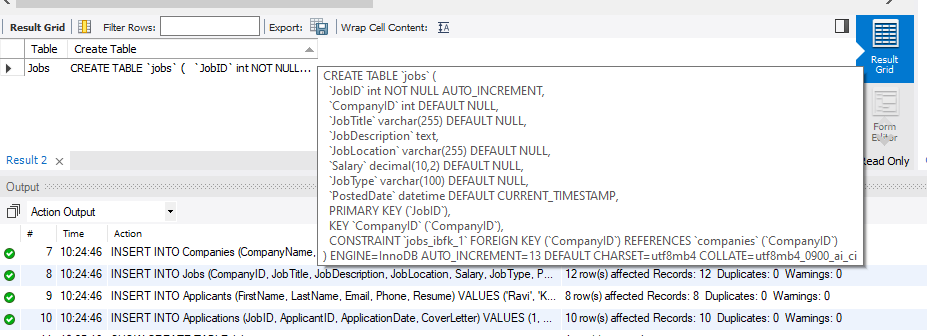
**Salary DECIMAL(10,2),**

**FOREIGN KEY (CompanyID) REFERENCES Companies(CompanyID)**

**);**

**SHOW CREATE TABLE Jobs;**

**OUTPUT SCREENSHOT:**

****

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

**QUERY:**

**CREATE DATABASE IF NOT EXISTS CareerHub;**

**USE CareerHub;**

**CREATE TABLE IF NOT EXISTS Companies (**

**CompanyID INT PRIMARY KEY AUTO\_INCREMENT,**

**CompanyName VARCHAR(255) NOT NULL,**

**Location VARCHAR(255)**

**);**

**CREATE TABLE IF NOT EXISTS Jobs (**

**JobID INT PRIMARY KEY AUTO\_INCREMENT,**

**CompanyID INT,**

**JobTitle VARCHAR(255),**

**JobDescription TEXT,**

**JobLocation VARCHAR(255),**

**Salary DECIMAL(10, 2),**

**JobType VARCHAR(100),**

**PostedDate DATETIME DEFAULT CURRENT\_TIMESTAMP,**

**FOREIGN KEY (CompanyID) REFERENCES Companies(CompanyID)**

**);**

**CREATE TABLE IF NOT EXISTS Applicants (**

**ApplicantID INT PRIMARY KEY AUTO\_INCREMENT,**

**FirstName VARCHAR(100),**

**LastName VARCHAR(100),**

**Email VARCHAR(255) UNIQUE,**

**Phone VARCHAR(20),**

**Resume TEXT**

**);**

**CREATE TABLE IF NOT EXISTS Applications (**

**ApplicationID INT PRIMARY KEY AUTO\_INCREMENT,**

**JobID INT,**

**ApplicantID INT,**

**ApplicationDate DATETIME DEFAULT CURRENT\_TIMESTAMP,**

**CoverLetter TEXT,**

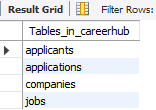
**FOREIGN KEY (JobID) REFERENCES Jobs(JobID),**

**FOREIGN KEY (ApplicantID) REFERENCES Applicants(ApplicantID)**

**);**

**SHOW TABLES;**

**OUTPUT SCREENSHOT:**

****

1. 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.

**QUERY:**

**SELECT**

**J.JobTitle,**

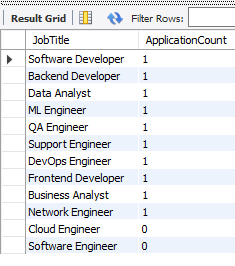
**COUNT(A.ApplicationID) AS ApplicationCount**

**FROM Jobs J**

**LEFT JOIN Applications A ON J.JobID = A.JobID**

**GROUP BY J.JobID, J.JobTitle;**

**OUPUT SCREENSHOT:**

****

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.

**QUERY:**

**SELECT**

**J.JobTitle,**

**C.CompanyName,**

**J.JobLocation,**

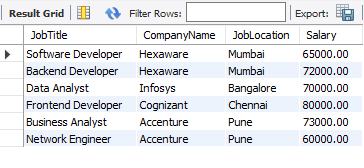
**J.Salary**

**FROM Jobs J**

**JOIN Companies C ON J.CompanyID = C.CompanyID**

**WHERE J.Salary BETWEEN 60000 AND 80000;**

**OUTPUT SCREENSHOT:**

****

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.

**QUERY:**

**SELECT**

**J.JobTitle,**

**C.CompanyName,**

**A.ApplicationDate**

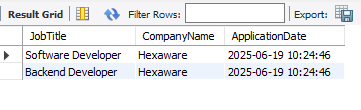
**FROM Applications A**

**JOIN Jobs J ON A.JobID = J.JobID**

**JOIN Companies C ON J.CompanyID = C.CompanyID**

**WHERE A.ApplicantID = 1;**

**OUTPUT SCREENSHOT:**

****

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.

**QUERY:**

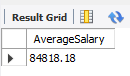
**SELECT**

**ROUND(AVG(Salary), 2) AS AverageSalary**

**FROM Jobs**

**WHERE Salary > 0;**

**OUTPUT SCREENSHOT:**

****

9. Write an SQL query to identify the company that has posted the most job listings. Display the

company name along with the count of job listings they have posted. Handle ties if multiple

companies have the same maximum count.

**QUERY:**

**SELECT**

**C.CompanyName,**

**COUNT(J.JobID) AS JobCount**

**FROM Companies C**

**JOIN Jobs J ON C.CompanyID = J.CompanyID**

**GROUP BY C.CompanyID, C.CompanyName**

**HAVING COUNT(J.JobID) = (**

**SELECT MAX(JobCount)**

**FROM (**

**SELECT COUNT(JobID) AS JobCount**

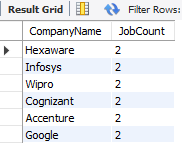
**FROM Jobs**

**GROUP BY CompanyID**

**) AS Sub**

**);**

**OUTPUT SCREENSHOT:**

****

10. Find the applicants who have applied for positions in companies located in 'CityX' and have at

least 3 years of experience.

**QUERY:**

**SELECT DISTINCT Ap.FirstName, Ap.LastName, Ap.Experience, C.CompanyName, C.Location**

**FROM Applicants Ap**

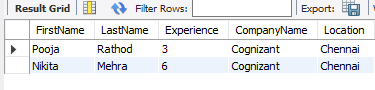
**JOIN Applications A ON Ap.ApplicantID = A.ApplicantID**

**JOIN Jobs J ON A.JobID = J.JobID**

**JOIN Companies C ON J.CompanyID = C.CompanyID**

**WHERE C.Location = 'Chennai' AND Ap.Experience >= 3;**

**OUPUT SCREENSHOT:**

****

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

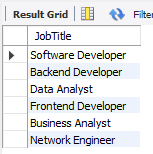
**QUERY:**

**SELECT DISTINCT JobTitle**

**FROM Jobs**

**WHERE Salary BETWEEN 60000 AND 80000;**

**OUTPUT SCREENSHOT:**

****

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

**QUERY:**

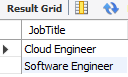
**SELECT J.JobTitle**

**FROM Jobs J**

**LEFT JOIN Applications A ON J.JobID = A.JobID**

**WHERE A.ApplicationID IS NULL;**

**OUTPUT SCREENSHOT:**

****

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

they have applied for.

**QUERY:**

**SELECT**

**Ap.FirstName,**

**Ap.LastName,**

**C.CompanyName,**

**J.JobTitle**

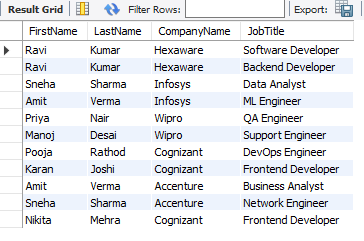
**FROM Applications A**

**JOIN Applicants Ap ON A.ApplicantID = Ap.ApplicantID**

**JOIN Jobs J ON A.JobID = J.JobID**

**JOIN Companies C ON J.CompanyID = C.CompanyID;**

**OUTPUT SCREENSHOT:**

****

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

received any applications.

**QUERY:**

**SELECT**

**C.CompanyName,**

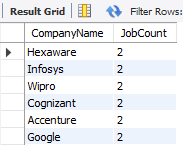
**COUNT(J.JobID) AS JobCount**

**FROM Companies C**

**LEFT JOIN Jobs J ON C.CompanyID = J.CompanyID**

**GROUP BY C.CompanyID, C.CompanyName;**

**OUTPUT SCREENSHOT:**

****

15. List all applicants along with the companies and positions they have applied for, including those

who have not applied.

**QUERY:**

**SELECT**

**Ap.FirstName,**

**Ap.LastName,**

**C.CompanyName,**

**J.JobTitle**

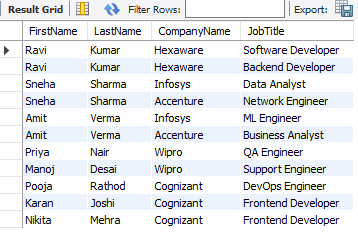
**FROM Applicants Ap**

**LEFT JOIN Applications A ON Ap.ApplicantID = A.ApplicantID**

**LEFT JOIN Jobs J ON A.JobID = J.JobID**

**LEFT JOIN Companies C ON J.CompanyID = C.CompanyID;**

**OUTPUT SCREENSHOT:**

****

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

**QUERY:**

**SELECT DISTINCT C.CompanyName**

**FROM Jobs J**

**JOIN Companies C ON J.CompanyID = C.CompanyID**

**WHERE J.Salary > (**

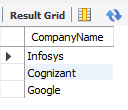
**SELECT AVG(Salary)**

**FROM Jobs**

**WHERE Salary > 0**

**);**

**OUTPUT SCREENSHOT:**

****

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

**QUERY:**

**SELECT**

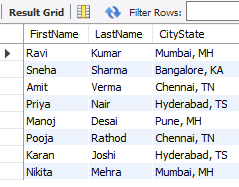
**FirstName,**

**LastName,**

**CONCAT(City, ', ', State) AS CityState**

**FROM Applicants;**

**OUTPUT SCREENSHOT:**

****

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

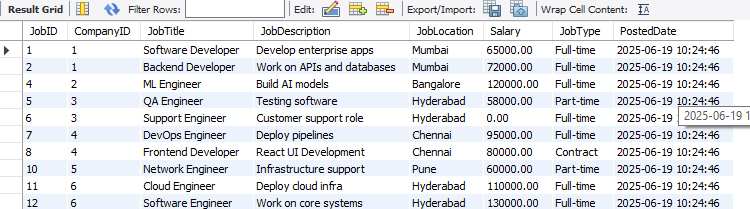
**QUERY:**

**SELECT \***

**FROM Jobs**

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

**OUTPUT SCREENSHOT:**

****

19. Retrieve a list of applicants and the jobs they have applied for, including those who have not

applied and jobs without applicants.

**QUERY:**

**-- Applicants with jobs (LEFT JOIN)**

**SELECT**

**Ap.FirstName,**

**Ap.LastName,**

**J.JobTitle**

**FROM Applicants Ap**

**LEFT JOIN Applications A ON Ap.ApplicantID = A.ApplicantID**

**LEFT JOIN Jobs J ON A.JobID = J.JobID**

**UNION**

**-- Jobs without applicants (RIGHT JOIN)**

**SELECT**

**Ap.FirstName,**

**Ap.LastName,**

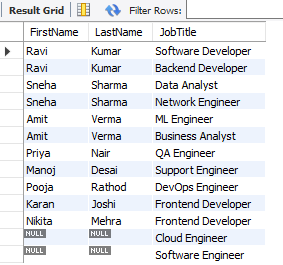
**J.JobTitle**

**FROM Applicants Ap**

**RIGHT JOIN Applications A ON Ap.ApplicantID = A.ApplicantID**

**RIGHT JOIN Jobs J ON A.JobID = J.JobID;**

**OUTPUT SCREENSHOT:**

****

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

**QUERY:**

**SELECT**

**A.FirstName,**

**A.LastName,**

**A.Experience,**

**C.CompanyName,**

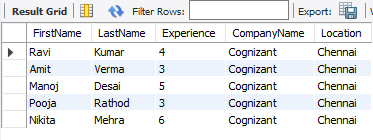
**C.Location**

**FROM Applicants A**

**CROSS JOIN Companies C**

**WHERE A.Experience > 2 AND C.Location = 'Chennai';**

**OUTPUT SCREENSHOT:**

****