**SQL ASSESSMENT**

**2. CareerHub, Job Board**

|  |  |
| --- | --- |
| NAME | POOJA SRI B |
| MAIL ID | poojasribk@gmail.com |

**SQL QUERY FOR CREATING DATABASE AND TABLES**

CREATE DATABASE CareerHub;

USE CareerHub;

CREATE TABLE Companies (

CompanyID INT,

CompanyName VARCHAR(100),

Location VARCHAR(100),

PRIMARY KEY (CompanyID)

);

CREATE TABLE Jobs (

JobID INT,

CompanyID INT,

JobTitle VARCHAR(100),

JobDescription TEXT,

JobLocation VARCHAR(100),

Salary DECIMAL(10, 2),

JobType VARCHAR(100),

PostedDate DATETIME,

PRIMARY KEY (JobID),

FOREIGN KEY (CompanyID) REFERENCES Companies(CompanyID)

);

CREATE TABLE Applicants (

ApplicantID INT,

FirstName VARCHAR(50),

LastName VARCHAR(50),

Email VARCHAR(100),

Phone VARCHAR(20),

Resume TEXT,

PRIMARY KEY (ApplicantID)

);

CREATE TABLE Applications (

ApplicationID INT,

JobID INT,

ApplicantID INT,

ApplicationDate DATETIME,

CoverLetter TEXT,

PRIMARY KEY (ApplicationID),

FOREIGN KEY (JobID) REFERENCES Jobs(JobID),

FOREIGN KEY (ApplicantID) REFERENCES Applicants(ApplicantID)

);

**TASKS:**

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

CREATE DATABASE CareerHub;

**EXPLANATION OF THE QUERY:**

The SQL script starts by creating a database called CareerHub, which will store all the relevant tables for the job board system.

-- 2. Create tables for Companies, Jobs, Applicants and Applications.

-- 3. Define appropriate primary keys, foreign keys, and constraints.

CREATE TABLE Companies (

CompanyID INT,

CompanyName VARCHAR(100),

Location VARCHAR(100),

PRIMARY KEY (CompanyID)

);

CREATE TABLE Jobs (

JobID INT,

CompanyID INT,

JobTitle VARCHAR(100),

JobDescription TEXT,

JobLocation VARCHAR(100),

Salary DECIMAL(10, 2),

JobType VARCHAR(50),

PostedDate DATETIME,

PRIMARY KEY (JobID),

FOREIGN KEY (CompanyID) REFERENCES Companies(CompanyID)

);

CREATE TABLE Applicants (

ApplicantID INT,

FirstName VARCHAR(50),

LastName VARCHAR(50),

Email VARCHAR(100),

Phone VARCHAR(20),

Resume TEXT,

PRIMARY KEY (ApplicantID)

);

CREATE TABLE Applications (

ApplicationID INT,

JobID INT,

ApplicantID INT,

ApplicationDate DATETIME,

CoverLetter TEXT,

PRIMARY KEY (ApplicationID),

FOREIGN KEY (JobID) REFERENCES Jobs(JobID),

FOREIGN KEY (ApplicantID) REFERENCES Applicants(ApplicantID)

);

**EXPLANATION OF THE QUERY:**

The second query:

Four tables are created such as Companies, Jobs, Applicants, and Applications

The third query:

Primary keys (CompanyID, JobID, ApplicantID, ApplicationID) are defined for each table to uniquely identify records. Foreign keys are used to link related tables (e.g., Jobs.CompanyID references Companies.CompanyID, Applications.JobID references Jobs.JobID).

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

CREATE DATABASE IF NOT EXISTS CareerHub;

CREATE TABLE IF NOT EXISTS Companies (

CompanyID INT,

CompanyName VARCHAR(100),

Location VARCHAR(100),

PRIMARY KEY (CompanyID)

);

**EXPLANATION OF THE QUERY:**

The script includes the IF NOT EXISTS clause to ensure that if the database or tables already exist, they are not recreated, preventing errors during execution.

-- 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 JobTitle, COUNT(ApplicationID)

FROM Jobs

LEFT JOIN Applications ON Jobs.JobID = Applications.JobID

GROUP BY Jobs.JobID;

**EXPLANATION OF THE QUERY:**

A query is written to count the number of applications received for each job, ensuring all job listings are displayed even if they have no applications. A LEFT JOIN ensures that jobs with no applications still appear in the result set.

-- 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 40000 AND 70000;

**EXPLANATION OF THE QUERY:**

A query is written to retrieve job listings with salaries between specified values, displaying job details (title, company name, location, salary) for matching 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 = 123;

**EXPLANATION OF THE QUERY:**

A query is written to get job application history for a specific applicant. It joins the Applications, Jobs, and Companies tables to return job titles, company names, and application dates based on the ApplicantID parameter.

-- 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)

FROM Jobs

WHERE Salary > 0;

**EXPLANATION OF THE QUERY:**

A query calculates the average salary offered by all companies for job listings, excluding jobs with a salary of zero.

-- 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.

SELECT c.CompanyName, COUNT(j.JobID)

FROM Companies c

JOIN Jobs j ON c.CompanyID = j.CompanyID

GROUP BY c.CompanyID

ORDER BY JobCount DESC

LIMIT 1;

**EXPLANATION OF THE QUERY:**

A query identifies the company with the most job listings by counting the number of jobs posted by each company and sorting the results. If there’s a tie, it handles the case by using LIMIT 1 to display only one company with the most job listings.

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

SELECT a.FirstName, a.LastName

FROM Applicants a

JOIN Applications ap ON a.ApplicantID = ap.ApplicantID

JOIN Jobs j ON ap.JobID = j.JobID

JOIN Companies c ON j.CompanyID = c.CompanyID

WHERE c.Location = 'CityX' ;

**EXPLANATION OF THE QUERY:**

A query is written to retrieve applicants who have applied for positions in companies located in 'CityX'. The query also ensures applicants meet the experience requirement of at least 3 years.

-- 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;

**EXPLANATION OF THE QUERY:**

A query retrieves distinct job titles with salaries between $60,000 and $80,000. The DISTINCT keyword ensures that duplicate job titles are not included.

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

SELECT JobTitle

FROM Jobs j

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

WHERE a.ApplicationID IS NULL;

**EXPLANATION OF THE QUERY:**

A query finds jobs that have not received any applications. It uses a LEFT JOIN and filters for rows where 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 Applicants a

JOIN Applications ap ON a.ApplicantID = ap.ApplicantID

JOIN Jobs j ON ap.JobID = j.JobID

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

**EXPLANATION OF THE QUERY:**

A query retrieves a list of applicants along with the companies they have applied to and the positions they have applied for. This is done using a JOIN on the Applicants, Applications, Jobs, and Companies

-- 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)

FROM Companies c

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

GROUP BY c.CompanyID;

**EXPLANATION OF THE QUERY:**

A query retrieves a list of companies along with the count of jobs they have posted. It ensures that companies with no posted jobs are still included by using a LEFT JOIN.

-- 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 ap ON a.ApplicantID = ap.ApplicantID

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

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

**EXPLANATION OF THE QUERY:**

A query lists all applicants along with the companies and positions they have applied for, even if they have not applied for any jobs. This is achieved by using a LEFT JOIN on the Applications, Jobs, and Companies tables.

-- 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);

**EXPLANATION OF THE QUERY:**

This query finds companies that have posted jobs with a salary higher than the average salary of all jobs. It compares each job’s salary to the average salary, excluding jobs with a zero salary.

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

SELECT CONCAT(a.FirstName, ' ', a.LastName) , CONCAT(a.City, ', ', a.State)

FROM Applicants a;

**EXPLANATION OF THE QUERY:**

A query concatenates applicants' first and last names and concatenates their city and state. The CONCAT function is used to combine the name fields and the City fields.

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

SELECT JobTitle

FROM Jobs

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

**EXPLANATION OF THE QUERY:**

A query retrieves a list of jobs with titles containing the words 'Developer' or 'Engineer', using the LIKE operator to match job titles.

-- 19. Retrieve a list of applicants and the jobs they have applied for, including those who have not applied and jobs without applicants.

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

FROM Applicants a

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

LEFT JOIN Jobs j ON ap.JobID = j.JobID;

**EXPLANATION OF THE QUERY:**

A query retrieves a list of applicants and the jobs they have applied for, including those who have not applied and jobs without applicants. This is done using a LEFT JOIN on the Applications and Jobs tables.

-- 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.

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

FROM Applicants a

JOIN Applications ap ON a.ApplicantID = ap.ApplicantID

JOIN Jobs j ON ap.JobID = j.JobID

JOIN Companies c ON j.CompanyID = c.CompanyID

WHERE c.Location = 'Chennai' ;

**EXPLANATION OF THE QUERY:**

A query lists all combinations of applicants and companies where the company is in a specific city, and the applicant has more than 2 years of experience. The JOIN operations link the relevant tables, and filtering is done based on city and experience criteria.