SQL Outputs

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

IF NOT EXISTS (

SELECT name

FROM sys.databases

WHERE name = N'Career\_Hub2'

)

BEGIN

CREATE DATABASE Career\_Hub2;

PRINT 'Database Career\_Hub created successfully';

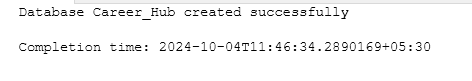
END

ELSE

BEGIN

PRINT 'Database Career\_Hub already exists';

END;



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

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

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

IF NOT EXISTS (

SELECT \* FROM sys.objects WHERE object\_id = OBJECT\_ID(N'companies') AND type = N'U'

)

BEGIN

CREATE TABLE companies(

companyId INT PRIMARY KEY,

companyName VARCHAR(100) NOT NULL,

location VARCHAR(100) NOT NULL

);

PRINT 'Table companies created successfully';

END

ELSE

BEGIN

PRINT 'Table companies already exists';

END;

--jobs

IF NOT EXISTS (

SELECT \* FROM sys.objects WHERE object\_id = OBJECT\_ID(N'jobs') AND type = N'U'

)

BEGIN

CREATE TABLE jobs(

jobId INT PRIMARY KEY,

companyId INT,

jobTitle VARCHAR(100) NOT NULL,

jobDescription VARCHAR(MAX) NOT NULL,

jobLocation VARCHAR(100) NOT NULL,

salary DECIMAL(15,2) NOT NULL,

jobType VARCHAR(30) NOT NULL,

postedDate DATETIME NOT NULL,

CONSTRAINT FK\_companies FOREIGN KEY (companyId) REFERENCES companies(companyId),

CONSTRAINT check\_jobType CHECK (jobType IN ('Full-time', 'Part-time', 'contract'))

);

PRINT 'Table jobs created successfully';

END

ELSE

BEGIN

PRINT 'Table jobs already exists';

END;

--applicants

IF NOT EXISTS (

SELECT \* FROM sys.objects WHERE object\_id = OBJECT\_ID(N'applicants') AND type = N'U'

)

BEGIN

CREATE TABLE applicants(

applicantId INT PRIMARY KEY,

first\_name VARCHAR(100) NOT NULL,

last\_name VARCHAR(100) NOT NULL,

email VARCHAR(100) NOT NULL,

phone VARCHAR(10) NOT NULL,

resume VARCHAR(MAX) NOT NULL

);

PRINT 'Table applicants created successfully';

END

ELSE

BEGIN

PRINT 'Table applicants already exists';

END;

--applicantions

IF NOT EXISTS (

SELECT \* FROM sys.objects WHERE object\_id = OBJECT\_ID(N'applications') AND type = N'U'

)

BEGIN

CREATE TABLE applications(

applicationId INT PRIMARY KEY,

jobID INT,

applicantId INT,

applicationtime DATETIME NOT NULL,

coverLetter VARCHAR(MAX) NOT NULL,

CONSTRAINT FK\_jobs FOREIGN KEY (jobID) REFERENCES jobs(jobID),

CONSTRAINT FK\_applicants FOREIGN KEY (applicantId) REFERENCES applicants(applicantId)

);

PRINT 'Table applications created successfully';

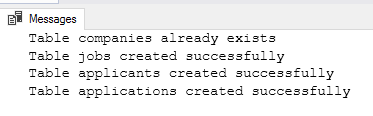
END

ELSE

BEGIN

PRINT 'Table applications already exists';

END;

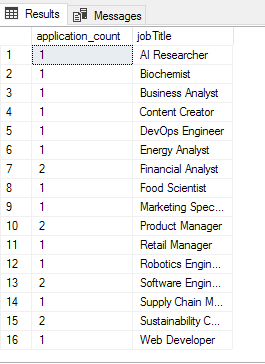


5. Write an SQL query to count the number of applications received for each job listing in the "Jobs" table. Display the job title and the corresponding application count. Ensure that it lists all jobs, even if they have no applications.

select count(jobTitle) as application\_count,jobTitle

from jobs

group by jobTitle;



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

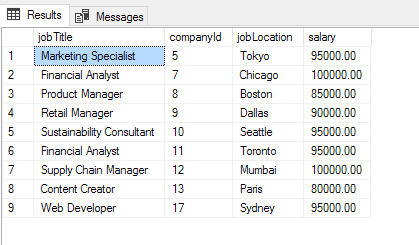
select j.jobTitle,c.companyId,j.jobLocation,j.salary from jobs as j

JOIN

companies as c

on j.companyId = c.companyId

where j.salary between 80000 and 100000;



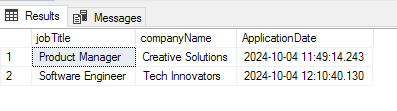
7. Write an SQL query that retrieves the job application history for a specific applicant. Allow a parameter for the ApplicantID, and return a result set with the job titles, company names, and application dates for all the jobs the applicant has applied to.

select j.jobTitle,c.companyName,a.applicationtime as ApplicationDate from applications as a

JOIN jobs as j on a.jobID = j.jobId

JOIN companies as c on j.companyId = c.companyId

where a.applicantId = 2;



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 c.companyName,avg(j.salary) as avg\_salary\_offered from jobs as j

JOIN

companies as c

on c.companyId = j.companyId

where j.salary > 0

group by j.companyId,c.companyName;



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.

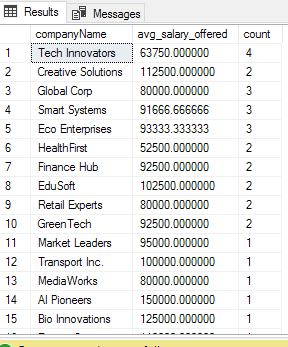
select c.companyName,avg(salary) as avg\_salary\_offered, count(j.jobTitle) count from jobs as j

JOIN

companies as c

on c.companyId = j.companyId

group by j.companyId,c.companyName;



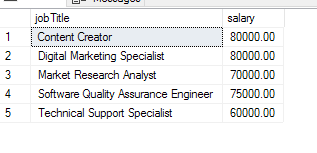
10. Find the applicants who have applied for positions in companies located in 'CityX' and have at least 3 years of experience.

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

SELECT DISTINCT jobTitle,salary

FROM jobs

WHERE salary BETWEEN 60000 AND 80000;



12. Find the jobs that have not received any applications

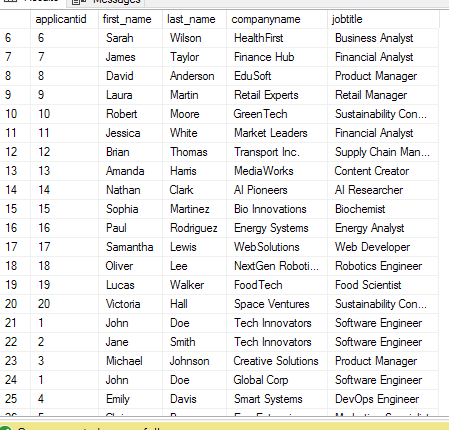
13. Retrieve a list of job applicants along with the companies they have applied to and the positions they have applied for

select a.applicantid, a.first\_name, a.last\_name, c.companyname, j.jobtitle from applicants as a

join applications as app on a.applicantid = app.applicantid

join jobs as j on app.jobid = j.jobid

join companies as c on j.companyid = c.companyid;



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

SELECT DISTINCT c.companyId, c.companyName

FROM companies AS c

JOIN jobs AS j ON c.companyId = j.companyId

WHERE j.salary > (SELECT AVG(salary) FROM jobs);



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

ALTER TABLE applicants

ADD city VARCHAR(100),

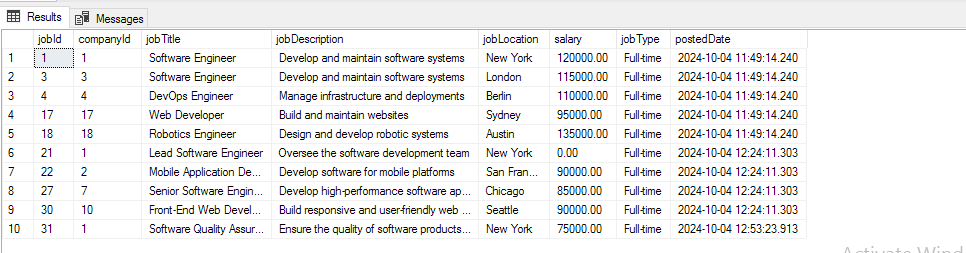
state VARCHAR(100);

SELECT a.first\_name, a.last\_name, CONCAT(a.city, ', ', a.state) AS city\_state FROM applicants AS a;

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

SELECT \*FROM jobs

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

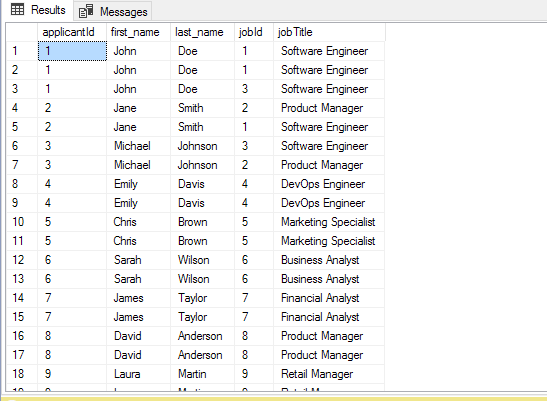


19. Retrieve a list of applicants and the jobs they have applied for, including those who have not applied and jobs without applicants

SELECT a.applicantId,a.first\_name,a.last\_name,j.jobId,j.jobTitle FROM applicants AS a

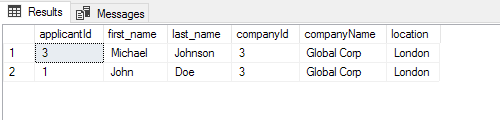
FULL OUTER JOIN applications AS ap ON a.applicantId = ap.applicantId

FULL OUTER JOIN jobs AS j ON ap.jobID = j.jobId;



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.applicantId,a.first\_name,a.last\_name,c.companyId,c.companyName,c.location FROM applicants AS a

JOIN applications AS ap ON a.applicantId = ap.applicantId

JOIN jobs AS j ON ap.jobID = j.jobId

JOIN companies AS c ON j.companyId = c.companyId

WHERE c.location = 'london' AND a.experience > 2;