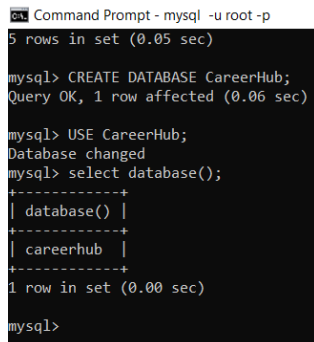


SQL Coding Challenge - 2

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

Ans. CREATE DATABASE CareerHub;

USE CareerHub;



```
Command Prompt - mysql -u root -p
mysql> CREATE DATABASE CareerHub;
Query OK, 1 row affected (0.06 sec)

mysql> USE CareerHub;
Database changed
mysql> select database();
+-----+
| database() |
+-----+
| careerhub  |
+-----+
1 row in set (0.00 sec)

mysql>
```

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

Ans.

CREATE TABLE Companies (

CompanyID INT PRIMARY KEY,

CompanyName VARCHAR(255),

Location VARCHAR(255)

);

CREATE TABLE Jobs (

JobID INT PRIMARY KEY,

CompanyID INT,

FOREIGN KEY (CompanyID) REFERENCES Companies(CompanyID),

JobTitle VARCHAR(255),

JobDescription TEXT,

JobLocation VARCHAR(255),

Salary DECIMAL,

JobType VARCHAR(255),

PostedDate DATETIME,

```
CONSTRAINT FK_JobCompany FOREIGN KEY (CompanyID) REFERENCES  
Companies(CompanyID)  
);
```

```
CREATE TABLE Applicants (  
    ApplicantID INT PRIMARY KEY,  
    FirstName VARCHAR(255),  
    LastName VARCHAR(255),  
    Email VARCHAR(255),  
    Phone VARCHAR(20),  
    Resume TEXT  
);
```

```
CREATE TABLE Applications (  
    ApplicationID INT PRIMARY KEY,  
    JobID INT,  
    ApplicantID INT,  
    ApplicationDate DATETIME,  
    CoverLetter TEXT,  
    CONSTRAINT FK_ApplicationJob FOREIGN KEY (JobID) REFERENCES Jobs(JobID),  
    CONSTRAINT FK_ApplicationApplicant FOREIGN KEY (ApplicantID) REFERENCES  
Applicants(ApplicantID)  
);
```

```

mysql> CREATE TABLE Companies (
->     CompanyID INT PRIMARY KEY,
->     CompanyName VARCHAR(255),
->     Location VARCHAR(255)
-> );
Query OK, 0 rows affected (0.06 sec)

mysql> CREATE TABLE Jobs (
->     JobID INT PRIMARY KEY,
->     CompanyID INT,
->     FOREIGN KEY (CompanyID) REFERENCES Companies(CompanyID),
->     JobTitle VARCHAR(255),
->     JobDescription TEXT,
->     JobLocation VARCHAR(255),
->     Salary DECIMAL,
->     JobType VARCHAR(255),
->     PostedDate DATETIME,
->     CONSTRAINT FK_JobCompany FOREIGN KEY (CompanyID) REFERENCES Companies(CompanyID)
-> );
Query OK, 0 rows affected (0.09 sec)

mysql> CREATE TABLE Applicants (
->     ApplicantID INT PRIMARY KEY,
->     FirstName VARCHAR(255),
->     LastName VARCHAR(255),
->     Email VARCHAR(255),
->     Phone VARCHAR(20),
->     Resume TEXT
-> );
Query OK, 0 rows affected (0.04 sec)

mysql> CREATE TABLE Applications (
->     ApplicationID INT PRIMARY KEY,
->     JobID INT,
->     ApplicantID INT,
->     ApplicationDate DATETIME,
->     CoverLetter TEXT,
->     CONSTRAINT FK_ApplicationJob FOREIGN KEY (JobID) REFERENCES Jobs(JobID),
->     CONSTRAINT FK_ApplicationApplicant FOREIGN KEY (ApplicantID) REFERENCES Applicants(ApplicantID)
-> );
Query OK, 0 rows affected (0.06 sec)

mysql>

```

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

Ans. To ensure that the script handles potential errors, such as if the database or tables already exist, you can use conditional logic and error handling mechanisms. In MySQL, you can use the IF NOT EXISTS clause to check if a database or table already exists before attempting to create it.

4. 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
    JobID,
    JobTitle,
    COUNT(ApplicationID) AS ApplicationCount
FROM
    Jobs
LEFT JOIN
    Applications ON Jobs.JobID = Applications.JobID
GROUP BY
    JobID, JobTitle
ORDER BY
    JobID;
```

```
mysql> SELECT
-> J.JobID,
-> 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;
+-----+-----+-----+
| JobID | JobTitle           | ApplicationCount |
+-----+-----+-----+
| 1     | Software Engineer  | 1               |
| 2     | Data Scientist     | 1               |
| 3     | Marketing Specialist | 1               |
| 4     | Front-end Developer | 1               |
| 5     | Financial Analyst   | 1               |
| 6     | Database Administrator | 1             |
| 7     | Environmental Scientist | 1             |
| 8     | Software QA Engineer | 1               |
| 9     | Construction Manager | 1               |
| 10    | Graphic Designer    | 1               |
+-----+-----+-----+
10 rows in set (0.00 sec)

mysql> _
```

5. 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
    JobTitle,
```

```

(SELECT CompanyName FROM Companies WHERE Companies.CompanyID =
Jobs.CompanyID) AS CompanyName,
    JobLocation,
    Salary
FROM
    Jobs
WHERE
    Salary BETWEEN @MinSalary AND @MaxSalary;

```

```

mysql> SELECT
->     J.JobTitle,
->     C.CompanyName,
->     J.JobLocation,
->     J.Salary
-> FROM
->     Jobs J
-> INNER JOIN
->     Companies C ON J.CompanyID = C.CompanyID
-> WHERE
->     J.Salary BETWEEN 70000 AND 90000;

```

JobTitle	CompanyName	JobLocation	Salary
Software Engineer	ABC Corporation	New York	80000
Data Scientist	XYZ Inc.	San Francisco	90000
Marketing Specialist	123 Enterprises	Los Angeles	70000
Front-end Developer	Tech Innovators	Seattle	85000
Database Administrator	Data Wizards	Boston	82000
Environmental Scientist	Eco Friendly Ltd.	Portland	72000
Software QA Engineer	Innovate Tech	Austin	78000
Construction Manager	City Builders	Denver	90000

```

8 rows in set (0.03 sec)

```

6. 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
    Jobs.JobTitle,
    Companies.CompanyName,
    Applications.ApplicationDate
FROM
    Applications
JOIN

```

Jobs ON Applications.JobID = Jobs.JobID

JOIN

Companies ON Jobs.CompanyID = Companies.CompanyID

WHERE

Applications.ApplicantID = 1;

```
mysql> SELECT
->   J.JobTitle,
->   C.CompanyName,
->   A.ApplicationDate
-> FROM
->   Applications A
-> INNER JOIN
->   Jobs J ON A.JobID = J.JobID
-> INNER JOIN
->   Companies C ON J.CompanyID = C.CompanyID
-> WHERE
->   A.ApplicantID = 1;
+-----+-----+-----+
| JobTitle      | CompanyName | ApplicationDate |
+-----+-----+-----+
| Software Engineer | ABC Corporation | 2023-01-18 13:15:00 |
+-----+-----+-----+
1 row in set (0.00 sec)
```

7. 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;

```
mysql> SELECT
->     AVG(Salary) AS AverageSalary
-> FROM
->     Jobs
-> WHERE
->     Salary > 0;
+-----+
| AverageSalary |
+-----+
|    80700.0000 |
+-----+
1 row in set (0.00 sec)
```

8. 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
    TOP 1 WITH TIES
    Companies.CompanyName,
    COUNT(Jobs.JobID) AS JobCount
FROM
    Companies
JOIN
    Jobs ON Companies.CompanyID = Jobs.CompanyID
GROUP BY
    Companies.CompanyName
ORDER BY
    JobCount DESC
LIMIT 1;
```

```
mysql> 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
-> ORDER BY
->     JobCount DESC
-> LIMIT 1;
+-----+-----+
| CompanyName | JobCount |
+-----+-----+
| ABC Corporation |      1 |
+-----+-----+
1 row in set (0.00 sec)

mysql> _
```

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

Ans. SELECT

A.ApplicantID,

A.FirstName,

A.LastName,

COALESCE(C.CompanyName, 'Not Applied') AS CompanyName,

COALESCE(J.JobTitle, 'Not Applied') AS JobTitle

FROM

Applicants A

INNER JOIN

Applications App ON A.ApplicantID = App.ApplicantID

INNER JOIN

Jobs J ON App.JobID = J.JobID

INNER JOIN

Companies C ON J.CompanyID = C.CompanyID

WHERE

C.Location = 'CityX'

AND DATEDIFF(CURDATE(), J.PostedDate) >= 1095; -- 3 years in days

```
mysql> SELECT
->   A.ApplicantID,
->   A.FirstName,
->   A.LastName,
->   COALESCE(C.CompanyName, 'Not Applied') AS CompanyName,
->   COALESCE(J.JobTitle, 'Not Applied') AS JobTitle
-> FROM
->   Applicants A
-> INNER JOIN
->   Applications App ON A.ApplicantID = App.ApplicantID
-> INNER JOIN
->   Jobs J ON App.JobID = J.JobID
-> INNER JOIN
->   Companies C ON J.CompanyID = C.CompanyID
-> WHERE
->   C.Location = 'CityX'
->   AND DATEDIFF(CURDATE(), J.PostedDate) >= 1095; -- 3 years in days
Empty set (0.00 sec)

mysql> _
```

10. Retrieve a list of distinct job titles with salaries veen \$60,000 and \$80,000.

Ans.

SELECT DISTINCT

JobTitle

FROM

Jobs

WHERE

Salary BETWEEN 60000 AND 80000;

```
mysql> SELECT DISTINCT
->   JobTitle
-> FROM
->   Jobs
-> WHERE
->   Salary BETWEEN 60000 AND 80000;
+-----+
| JobTitle |
+-----+
| Software Engineer |
| Marketing Specialist |
| Environmental Scientist |
| Software QA Engineer |
| Graphic Designer |
+-----+
5 rows in set (0.00 sec)

mysql>
```

11. Find the jobs that have not received any applications least 3 years of experience.

```
SELECT
    Jobs.JobID,
    Jobs.JobTitle,
    Jobs.JobLocation,
    Jobs.Salary,
    Jobs.JobType,
    Jobs.PostedDate
FROM
    Jobs
LEFT JOIN
    Applications ON Jobs.JobID = Applications.JobID
WHERE
    Applications.ApplicationID IS NULL
    AND Jobs.RequiredExperience >= 3;
```

```
mysql> SELECT
->     J.JobID,
->     J.JobTitle,
->     J.PostedDate
-> FROM
->     Jobs J
-> LEFT JOIN
->     Applications A ON J.JobID = A.JobID
-> WHERE
->     A.ApplicationID IS NULL
->     AND DATEDIFF(CURDATE(), J.PostedDate) >= 1095; -- 3 years in days
Empty set (0.01 sec)

mysql>
```

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

SELECT

Applicants.ApplicantID,
Applicants.FirstName,
Applicants.LastName,
Companies.CompanyName,
Jobs.JobTitle

FROM

Applicants

JOIN

Applications ON Applicants.ApplicantID = Applications.ApplicantID

JOIN

Jobs ON Applications.JobID = Jobs.JobID

JOIN

Companies ON Jobs.CompanyID = Companies.CompanyID;

```
mysql> SELECT
->   A.ApplicantID,
->   A.FirstName,
->   A.LastName,
->   C.CompanyName,
->   J.JobTitle
-> FROM
->   Applicants A
-> INNER JOIN
->   Applications App ON A.ApplicantID = App.ApplicantID
-> INNER JOIN
->   Jobs J ON App.JobID = J.JobID
-> INNER JOIN
->   Companies C ON J.CompanyID = C.CompanyID;
```

	ApplicantID	FirstName	LastName	CompanyName	JobTitle
1	John	Doe	ABC Corporation	Software Engineer	
2	Jane	Smith	XYZ Inc.	Data Scientist	
3	Mike	Johnson	123 Enterprises	Marketing Specialist	
4	Emily	Williams	Tech Innovators	Front-end Developer	
5	Chris	Davis	Global Solutions	Financial Analyst	
6	Amy	Brown	Data Wizards	Database Administrator	
7	Daniel	Miller	Eco Friendly Ltd.	Environmental Scientist	
8	Eva	Jones	Innovate Tech	Software QA Engineer	
9	Alex	Taylor	City Builders	Construction Manager	
10	Sophia	Moore	Creative Minds	Graphic Designer	

```
10 rows in set (0.00 sec)

mysql> _
```

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

SELECT

Companies.CompanyID,
Companies.CompanyName,
COUNT(Jobs.JobID) AS PostedJobsCount

FROM

Companies

LEFT JOIN

Jobs ON Companies.CompanyID = Jobs.CompanyID

LEFT JOIN

Applications ON Jobs.JobID = Applications.JobID

GROUP BY

Companies.CompanyID, Companies.CompanyName;

```
mysql> SELECT
->     C.CompanyID,
->     C.CompanyName,
->     COUNT(J.JobID) AS JobCount
-> FROM
->     Companies C
-> LEFT JOIN
->     Jobs J ON C.CompanyID = J.CompanyID
-> LEFT JOIN
->     Applications A ON J.JobID = A.JobID
-> GROUP BY
->     C.CompanyID, C.CompanyName;
```

CompanyID	CompanyName	JobCount
1	ABC Corporation	1
2	XYZ Inc.	1
3	123 Enterprises	1
4	Tech Innovators	1
5	Global Solutions	1
6	Data Wizards	1
7	Eco Friendly Ltd.	1
8	Innovate Tech	1
9	City Builders	1
10	Creative Minds	1

```
10 rows in set (0.00 sec)

mysql> _
```

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

SELECT

Applicants.ApplicantID,
Applicants.FirstName,
Applicants.LastName,
Companies.CompanyName,
Jobs.JobTitle

FROM

Applicants

CROSS JOIN

Companies

CROSS JOIN

Jobs

LEFT JOIN

Applications ON Applicants.ApplicantID = Applications.ApplicantID AND Jobs.JobID = Applications.JobID;

```
mysql> SELECT
-> A.ApplicantID,
-> A.FirstName,
-> A.LastName,
-> COALESCE(C.CompanyName, 'Not Applied') AS CompanyName,
-> COALESCE(J.JobTitle, 'Not Applied') AS 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;
```

	ApplicantID	FirstName	LastName	CompanyName	JobTitle
1	John	Doe	ABC Corporation	Software Engineer	
2	Jane	Smith	XYZ Inc.	Data Scientist	
3	Mike	Johnson	123 Enterprises	Marketing Specialist	
4	Emily	Williams	Tech Innovators	Front-end Developer	
5	Chris	Davis	Global Solutions	Financial Analyst	
6	Amy	Brown	Data Wizards	Database Administrator	
7	Daniel	Miller	Eco Friendly Ltd.	Environmental Scientist	
8	Eva	Jones	Innovate Tech	Software QA Engineer	
9	Alex	Taylor	City Builders	Construction Manager	
10	Sophia	Moore	Creative Minds	Graphic Designer	

```
10 rows in set (0.00 sec)

mysql>
```

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

```
SELECT DISTINCT
    Companies.CompanyID,
    Companies.CompanyName
FROM
    Companies
JOIN
    Jobs ON Companies.CompanyID = Jobs.CompanyID
WHERE
    Jobs.Salary > (SELECT AVG(Salary) FROM Jobs);
```

```
mysql> SELECT
->     DISTINCT C.CompanyID,
->     C.CompanyName
-> FROM
->     Companies C
-> INNER JOIN
->     Jobs J ON C.CompanyID = J.CompanyID
-> WHERE
->     J.Salary > (SELECT AVG(Salary) FROM Jobs);
+-----+-----+
| CompanyID | CompanyName |
+-----+-----+
| 2 | XYZ Inc. |
| 4 | Tech Innovators |
| 5 | Global Solutions |
| 6 | Data Wizards |
| 9 | City Builders |
+-----+-----+
5 rows in set (0.00 sec)

mysql> _
```

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

```
SELECT
    ApplicantID,
    FirstName,
    LastName,
    CONCAT(City, ' ', State) AS Location
FROM
```

Applicants;

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

```
SELECT
    JobID,
    JobTitle
FROM
    Jobs
WHERE
    JobTitle LIKE '%Developer%' OR JobTitle LIKE '%Engineer%';
```

```
mysql> SELECT
->     JobID,
->     JobTitle
-> FROM
->     Jobs
-> WHERE
->     JobTitle LIKE '%Developer%' OR JobTitle LIKE '%Engineer%';
+-----+-----+
| JobID | JobTitle |
+-----+-----+
| 1     | Software Engineer |
| 4     | Front-end Developer |
| 8     | Software QA Engineer |
+-----+-----+
3 rows in set (0.03 sec)

mysql>
```

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.FirstName,
    A.LastName,
    COALESCE(J.JobTitle, 'No Application') AS AppliedJobTitle
FROM
    Applicants A
```

LEFT JOIN

Applications AP ON A.ApplicantID = AP.ApplicantID

LEFT JOIN

Jobs J ON AP.JobID = J.JobID;

```
mysql> SELECT
->   A.ApplicantID,
->   A.FirstName,
->   A.LastName,
->   COALESCE(C.CompanyName, 'Not Applied') AS CompanyName,
->   COALESCE(J.JobTitle, 'Not Applied') AS 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;
```

ApplicantID	FirstName	LastName	CompanyName	JobTitle
1	John	Doe	ABC Corporation	Software Engineer
2	Jane	Smith	XYZ Inc.	Data Scientist
3	Mike	Johnson	123 Enterprises	Marketing Specialist
4	Emily	Williams	Tech Innovators	Front-end Developer
5	Chris	Davis	Global Solutions	Financial Analyst
6	Amy	Brown	Data Wizards	Database Administrator
7	Daniel	Miller	Eco Friendly Ltd.	Environmental Scientist
8	Eva	Jones	Innovate Tech	Software QA Engineer
9	Alex	Taylor	City Builders	Construction Manager
10	Sophia	Moore	Creative Minds	Graphic Designer

10 rows in set (0.00 sec)

```
mysql>
```

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.

Ans.

SELECT

A.ApplicantID,

A.FirstName,

A.LastName,

COALESCE(C.CompanyName, 'Not Applied') AS CompanyName,

COALESCE(J.JobTitle, 'Not Applied') AS 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

WHERE

J.JobLocation = 'Chennai'

AND DATEDIFF(CURDATE(), J.PostedDate) >= 730; -- 2 years in days

```
mysql> SELECT
->   A.ApplicantID,
->   A.FirstName,
->   A.LastName,
->   COALESCE(C.CompanyName, 'Not Applied') AS CompanyName,
->   COALESCE(J.JobTitle, 'Not Applied') AS 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
-> WHERE
->   J.JobLocation = 'Chennai'
->   AND DATEDIFF(CURDATE(), J.PostedDate) >= 730; -- 2 years in days
Empty set (0.03 sec)

mysql>
```