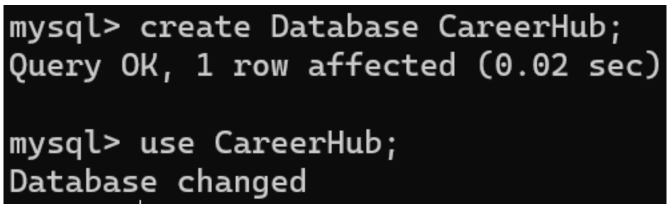
SQL Coding Challenges – CareerHub, The Job Board

Tasks:

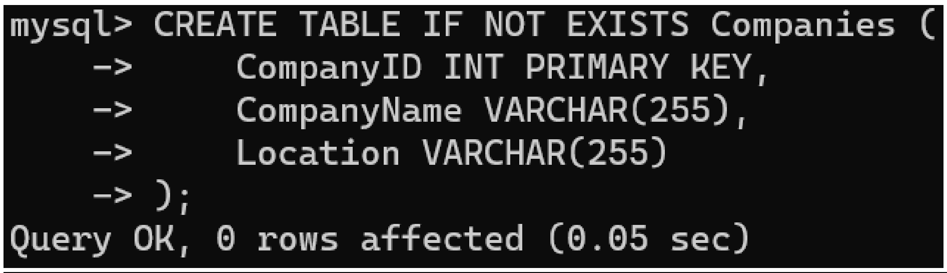
1. Provide a SQL script that initializes the database for the Job Board scenario "CareerHub".



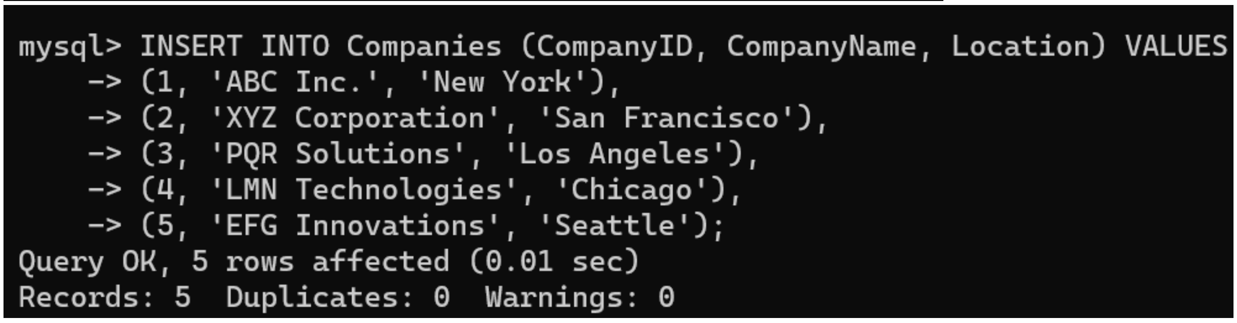
```
mysql> create Database CareerHub;
Query OK, 1 row affected (0.02 sec)

mysql> use CareerHub;
Database changed
```

2. Create tables for Companies, Jobs, Applicants and Applications. Define appropriate primary keys, foreign keys, and constraints. Ensure the script handles potential errors, such as if the database or tables already exist.
Companies:

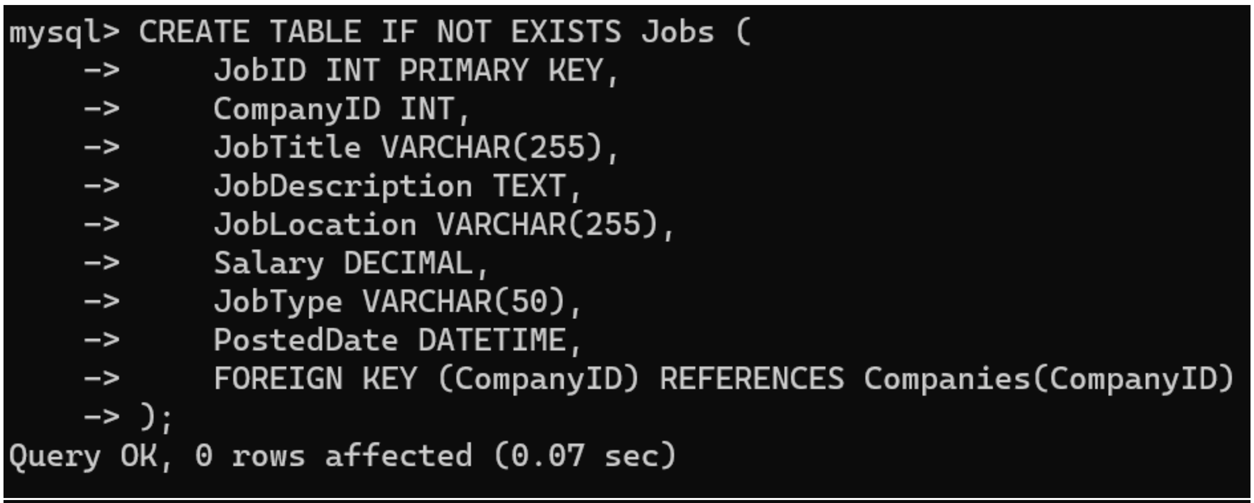


```
mysql> CREATE TABLE IF NOT EXISTS Companies (
    ->     CompanyID INT PRIMARY KEY,
    ->     CompanyName VARCHAR(255),
    ->     Location VARCHAR(255)
    -> );
Query OK, 0 rows affected (0.05 sec)
```

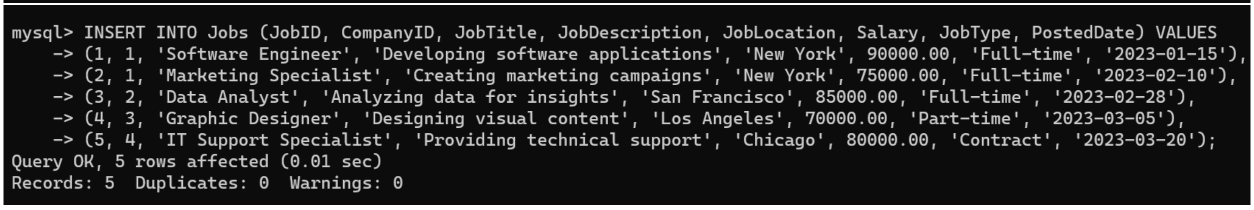


```
mysql> INSERT INTO Companies (CompanyID, CompanyName, Location) VALUES
    -> (1, 'ABC Inc.', 'New York'),
    -> (2, 'XYZ Corporation', 'San Francisco'),
    -> (3, 'PQR Solutions', 'Los Angeles'),
    -> (4, 'LMN Technologies', 'Chicago'),
    -> (5, 'EFG Innovations', 'Seattle');
Query OK, 5 rows affected (0.01 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

Jobs:

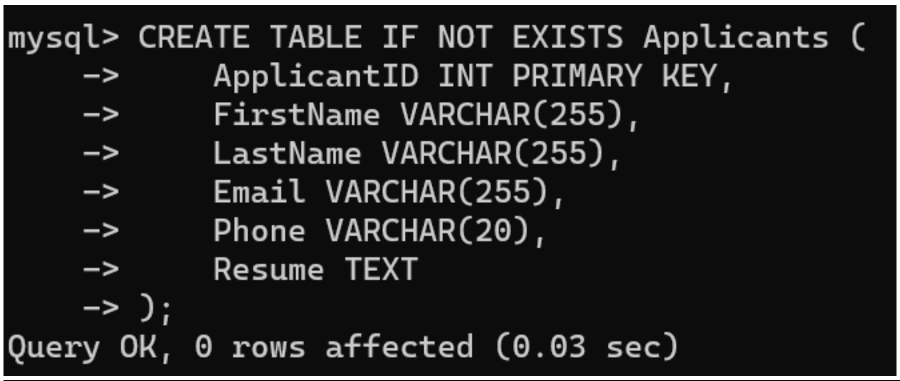


```
mysql> CREATE TABLE IF NOT EXISTS Jobs (
    ->     JobID INT PRIMARY KEY,
    ->     CompanyID INT,
    ->     JobTitle VARCHAR(255),
    ->     JobDescription TEXT,
    ->     JobLocation VARCHAR(255),
    ->     Salary DECIMAL,
    ->     JobType VARCHAR(50),
    ->     PostedDate DATETIME,
    ->     FOREIGN KEY (CompanyID) REFERENCES Companies(CompanyID)
    -> );
Query OK, 0 rows affected (0.07 sec)
```

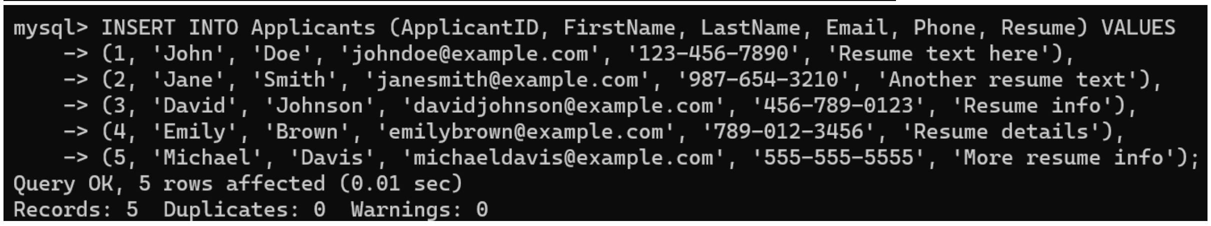


```
mysql> INSERT INTO Jobs (JobID, CompanyID, JobTitle, JobDescription, JobLocation, Salary, JobType, PostedDate) VALUES
    -> (1, 1, 'Software Engineer', 'Developing software applications', 'New York', 90000.00, 'Full-time', '2023-01-15'),
    -> (2, 1, 'Marketing Specialist', 'Creating marketing campaigns', 'New York', 75000.00, 'Full-time', '2023-02-10'),
    -> (3, 2, 'Data Analyst', 'Analyzing data for insights', 'San Francisco', 85000.00, 'Full-time', '2023-02-28'),
    -> (4, 3, 'Graphic Designer', 'Designing visual content', 'Los Angeles', 70000.00, 'Part-time', '2023-03-05'),
    -> (5, 4, 'IT Support Specialist', 'Providing technical support', 'Chicago', 80000.00, 'Contract', '2023-03-20');
Query OK, 5 rows affected (0.01 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

Applicants:

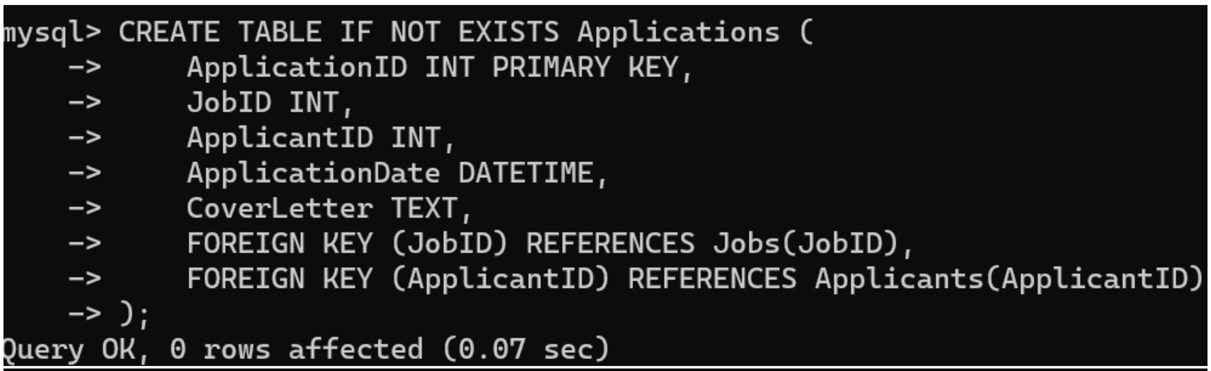


```
mysql> CREATE TABLE IF NOT EXISTS Applicants (
    ->     ApplicantID INT PRIMARY KEY,
    ->     FirstName VARCHAR(255),
    ->     LastName VARCHAR(255),
    ->     Email VARCHAR(255),
    ->     Phone VARCHAR(20),
    ->     Resume TEXT
    -> );
Query OK, 0 rows affected (0.03 sec)
```

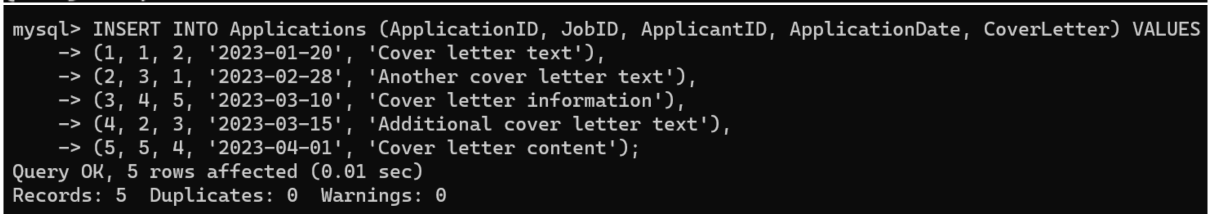


```
mysql> INSERT INTO Applicants (ApplicantID, FirstName, LastName, Email, Phone, Resume) VALUES
    -> (1, 'John', 'Doe', 'johndoe@example.com', '123-456-7890', 'Resume text here'),
    -> (2, 'Jane', 'Smith', 'janesmith@example.com', '987-654-3210', 'Another resume text'),
    -> (3, 'David', 'Johnson', 'davidjohnson@example.com', '456-789-0123', 'Resume info'),
    -> (4, 'Emily', 'Brown', 'emilybrown@example.com', '789-012-3456', 'Resume details'),
    -> (5, 'Michael', 'Davis', 'michaeldavis@example.com', '555-555-5555', 'More resume info');
Query OK, 5 rows affected (0.01 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

Applications:

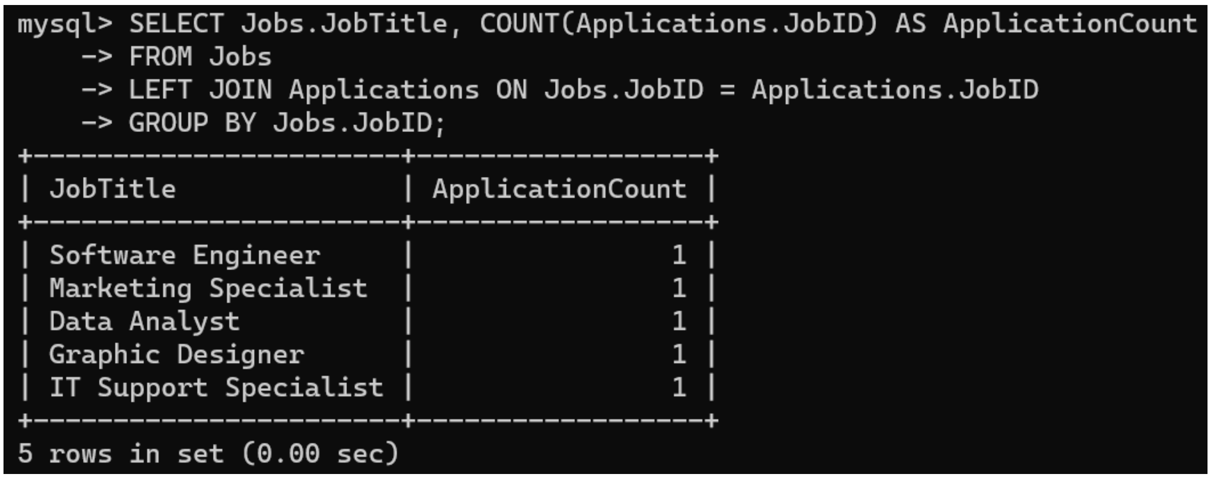


```
mysql> CREATE TABLE IF NOT EXISTS Applications (
    ->     ApplicationID INT PRIMARY KEY,
    ->     JobID INT,
    ->     ApplicantID INT,
    ->     ApplicationDate DATETIME,
    ->     CoverLetter TEXT,
    ->     FOREIGN KEY (JobID) REFERENCES Jobs(JobID),
    ->     FOREIGN KEY (ApplicantID) REFERENCES Applicants(ApplicantID)
    -> );
Query OK, 0 rows affected (0.07 sec)
```



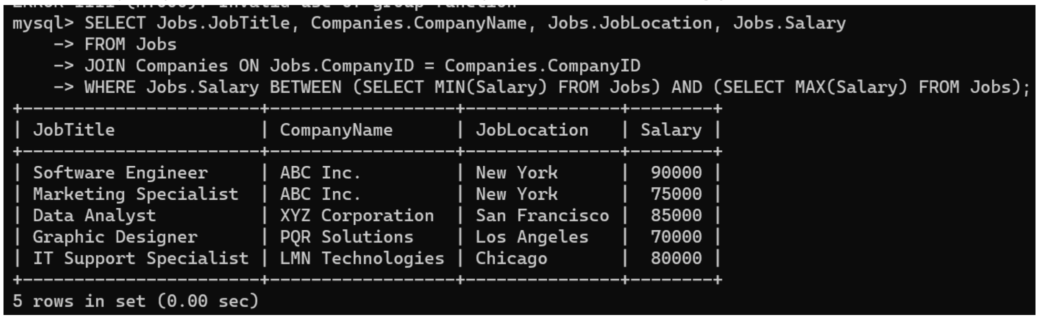
```
mysql> INSERT INTO Applications (ApplicationID, JobID, ApplicantID, ApplicationDate, CoverLetter) VALUES
    -> (1, 1, 2, '2023-01-20', 'Cover letter text'),
    -> (2, 3, 1, '2023-02-28', 'Another cover letter text'),
    -> (3, 4, 5, '2023-03-10', 'Cover letter information'),
    -> (4, 2, 3, '2023-03-15', 'Additional cover letter text'),
    -> (5, 5, 4, '2023-04-01', 'Cover letter content');
Query OK, 5 rows affected (0.01 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

3. 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.



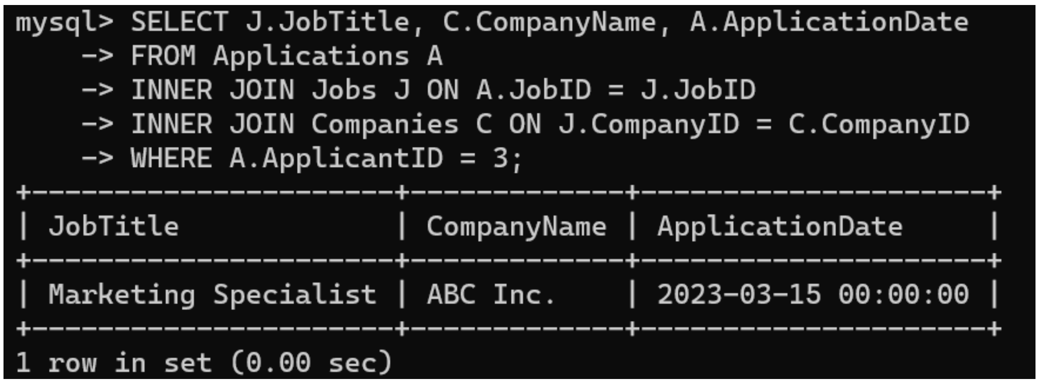
```
mysql> SELECT Jobs.JobTitle, COUNT(Applications.JobID) AS ApplicationCount
    -> FROM Jobs
    -> LEFT JOIN Applications ON Jobs.JobID = Applications.JobID
    -> GROUP BY Jobs.JobID;
+-----------------------+------------------+
| JobTitle              | ApplicationCount |
+-----------------------+------------------+
| Software Engineer     |                1 |
| Marketing Specialist  |                1 |
| Data Analyst          |                1 |
| Graphic Designer      |                1 |
| IT Support Specialist |                1 |
+-----------------------+------------------+
5 rows in set (0.00 sec)
```

4. 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.



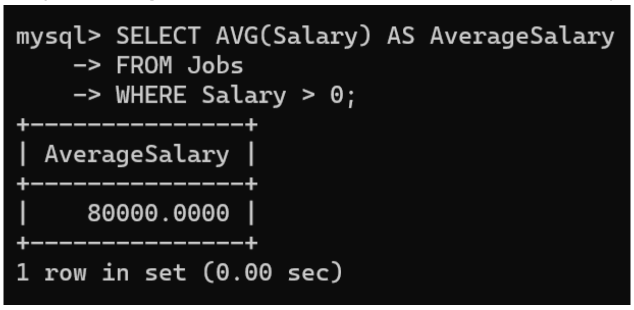
```
mysql> SELECT Jobs.JobTitle, Companies.CompanyName, Jobs.JobLocation, Jobs.Salary
    -> FROM Jobs
    -> JOIN Companies ON Jobs.CompanyID = Companies.CompanyID
    -> WHERE Jobs.Salary BETWEEN (SELECT MIN(Salary) FROM Jobs) AND (SELECT MAX(Salary) FROM Jobs);
+----------------------+-----------------+---------------+--------+
| JobTitle             | CompanyName     | JobLocation   | Salary |
+----------------------+-----------------+---------------+--------+
| Software Engineer    | ABC Inc.        | New York      |  90000 |
| Marketing Specialist | ABC Inc.        | New York      |  75000 |
| Data Analyst         | XYZ Corporation | San Francisco |  85000 |
| Graphic Designer     | PQR Solutions   | Los Angeles   |  70000 |
| IT Support Specialist | LMN Technologies | Chicago     |  80000 |
+----------------------+-----------------+---------------+--------+
5 rows in set (0.00 sec)
```

5. 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.



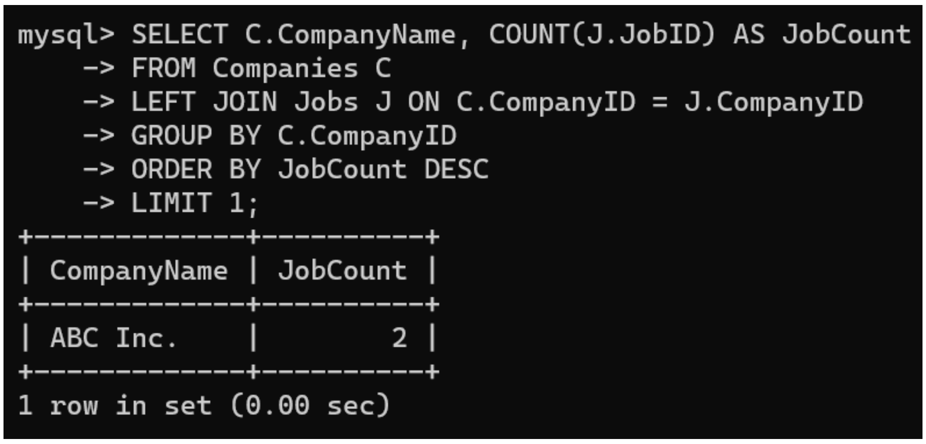
```
mysql> SELECT J.JobTitle, C.CompanyName, A.ApplicationDate
    -> FROM Applications A
    -> INNER JOIN Jobs J ON A.JobID = J.JobID
    -> INNER JOIN Companies C ON J.CompanyID = C.CompanyID
    -> WHERE A.ApplicantID = 3;
+----------------------+-------------+---------------------+
| JobTitle             | CompanyName | ApplicationDate     |
+----------------------+-------------+---------------------+
| Marketing Specialist | ABC Inc.    | 2023-03-15 00:00:00 |
+----------------------+-------------+---------------------+
1 row in set (0.00 sec)
```

6. 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.



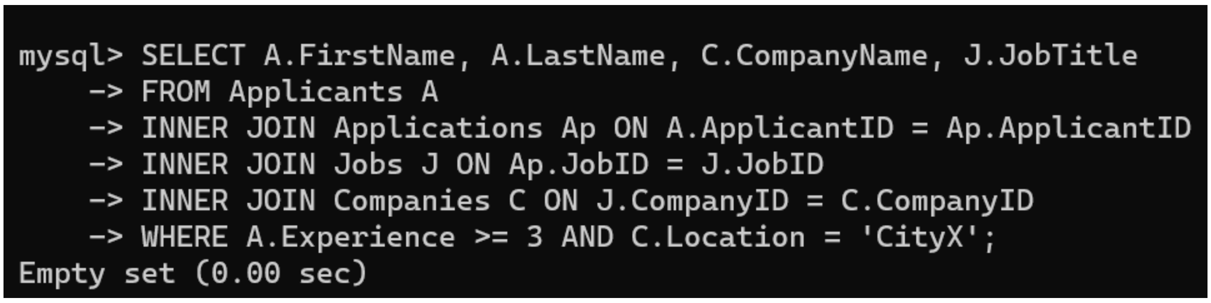
```
mysql> SELECT AVG(Salary) AS AverageSalary
    -> FROM Jobs
    -> WHERE Salary > 0;
+---------------+
| AverageSalary |
+---------------+
|    80000.0000 |
+---------------+
1 row in set (0.00 sec)
```

7. 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.



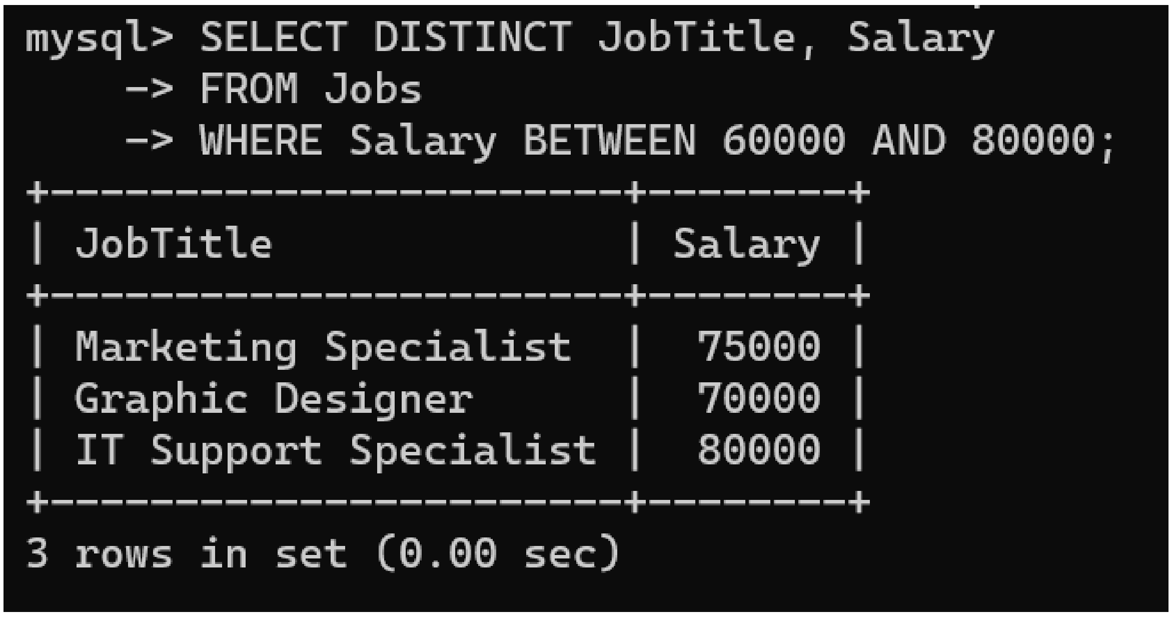
```
mysql> SELECT C.CompanyName, COUNT(J.JobID) AS JobCount
    -> FROM Companies C
    -> LEFT JOIN Jobs J ON C.CompanyID = J.CompanyID
    -> GROUP BY C.CompanyID
    -> ORDER BY JobCount DESC
    -> LIMIT 1;
+-------------+----------+
| CompanyName | JobCount |
+-------------+----------+
| ABC Inc.    |        2 |
+-------------+----------+
1 row in set (0.00 sec)
```

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



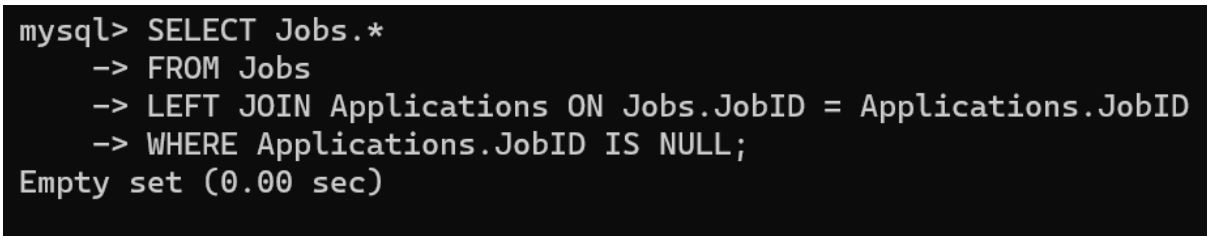
```
mysql> SELECT A.FirstName, A.LastName, C.CompanyName, J.JobTitle
    -> FROM Applicants A
    -> INNER JOIN Applications Ap ON A.ApplicantID = Ap.ApplicantID
    -> INNER JOIN Jobs J ON Ap.JobID = J.JobID
    -> INNER JOIN Companies C ON J.CompanyID = C.CompanyID
    -> WHERE A.Experience >= 3 AND C.Location = 'CityX';
Empty set (0.00 sec)
```

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



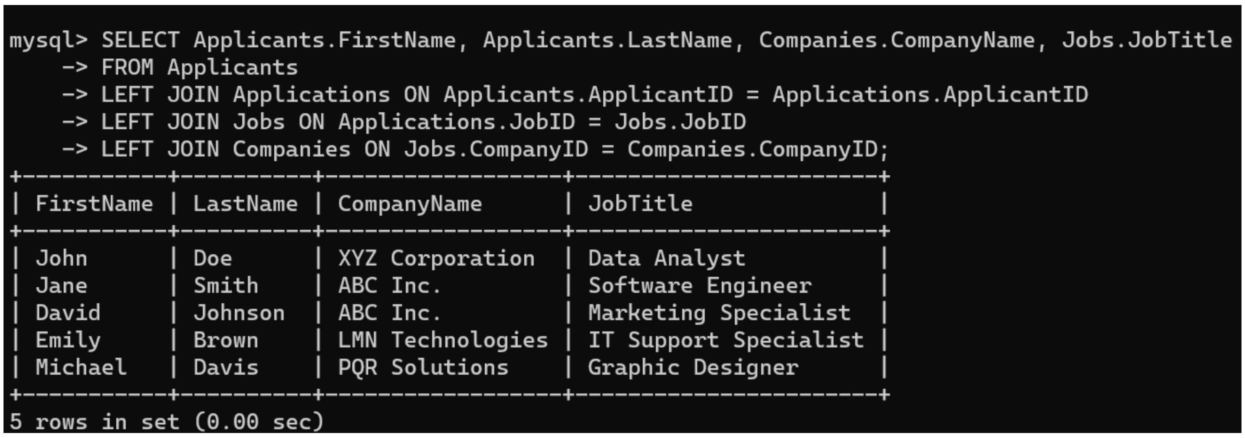
```
mysql> SELECT DISTINCT JobTitle, Salary
    -> FROM Jobs
    -> WHERE Salary BETWEEN 60000 AND 80000;
+-----------------------+--------+
| JobTitle              | Salary |
+-----------------------+--------+
| Marketing Specialist  |  75000 |
| Graphic Designer      |  70000 |
| IT Support Specialist |  80000 |
+-----------------------+--------+
3 rows in set (0.00 sec)
```

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



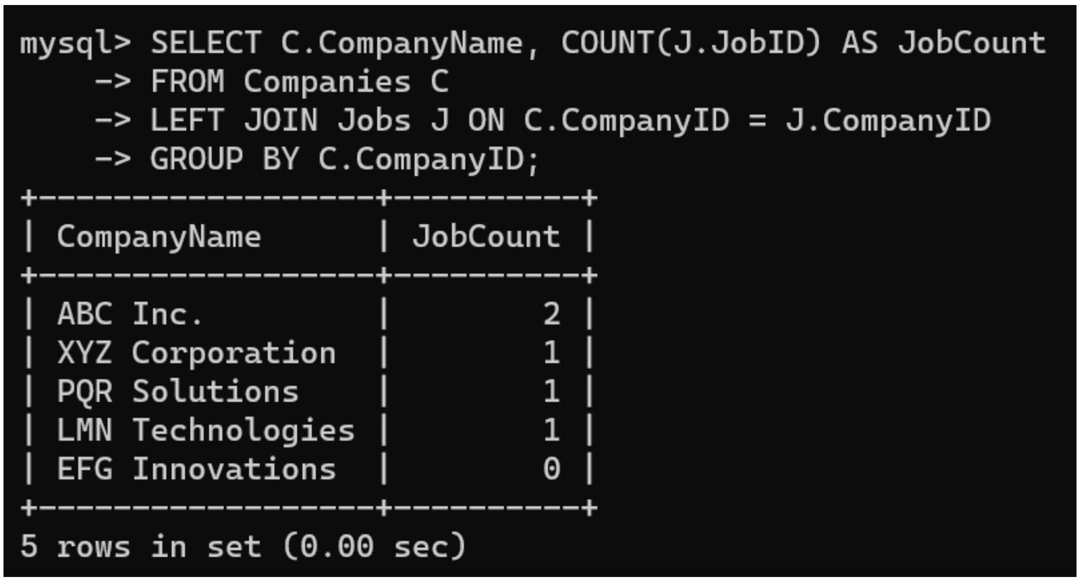
```
mysql> SELECT Jobs.*
    -> FROM Jobs
    -> LEFT JOIN Applications ON Jobs.JobID = Applications.JobID
    -> WHERE Applications.JobID IS NULL;
Empty set (0.00 sec)
```

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



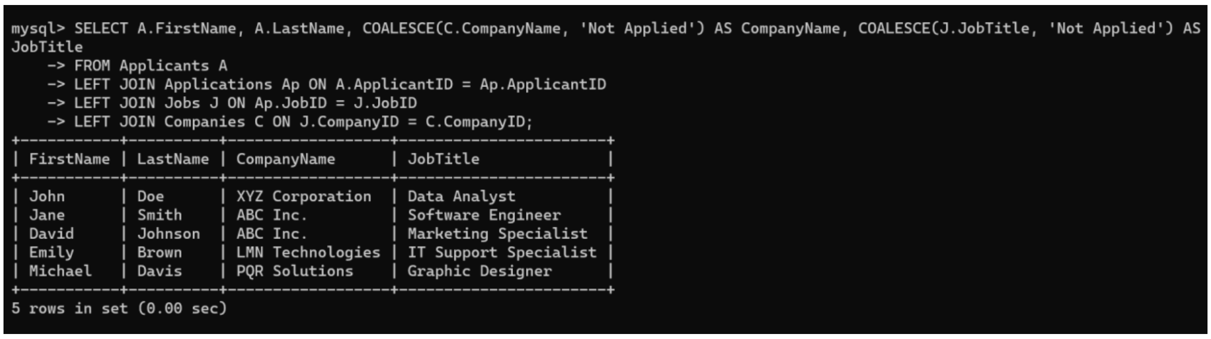
```
mysql> SELECT Applicants.FirstName, Applicants.LastName, Companies.CompanyName, Jobs.JobTitle
    -> FROM Applicants
    -> LEFT JOIN Applications ON Applicants.ApplicantID = Applications.ApplicantID
    -> LEFT JOIN Jobs ON Applications.JobID = Jobs.JobID
    -> LEFT JOIN Companies ON Jobs.CompanyID = Companies.CompanyID;
+-----------+----------+------------------+-----------------------+
| FirstName | LastName | CompanyName      | JobTitle              |
+-----------+----------+------------------+-----------------------+
| John      | Doe      | XYZ Corporation  | Data Analyst          |
| Jane      | Smith    | ABC Inc.         | Software Engineer     |
| David     | Johnson  | ABC Inc.         | Marketing Specialist  |
| Emily     | Brown    | LMN Technologies | IT Support Specialist |
| Michael   | Davis    | PQR Solutions    | Graphic Designer      |
+-----------+----------+------------------+-----------------------+
5 rows in set (0.00 sec)
```

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



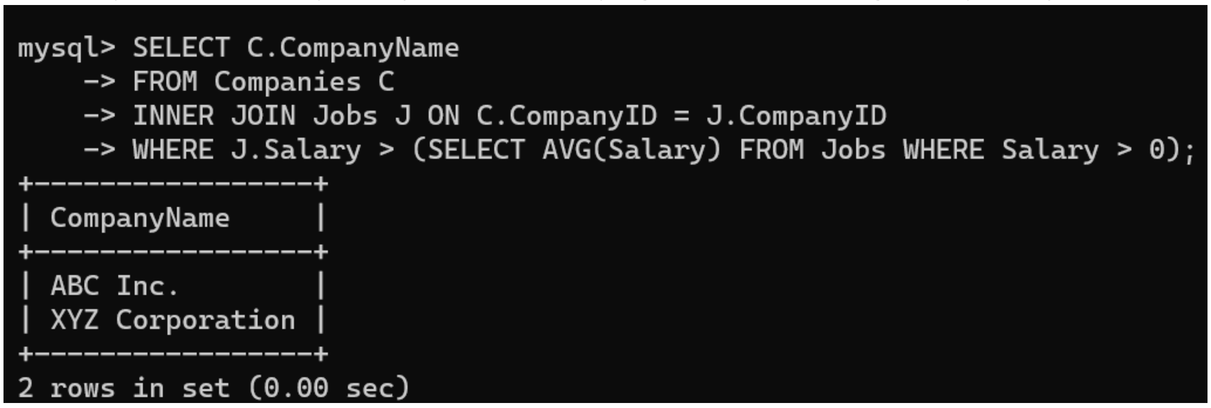
```
mysql> SELECT C.CompanyName, COUNT(J.JobID) AS JobCount
    -> FROM Companies C
    -> LEFT JOIN Jobs J ON C.CompanyID = J.CompanyID
    -> GROUP BY C.CompanyID;
+------------------+----------+
| CompanyName      | JobCount |
+------------------+----------+
| ABC Inc.         |        2 |
| XYZ Corporation  |        1 |
| PQR Solutions    |        1 |
| LMN Technologies |        1 |
| EFG Innovations  |        0 |
+------------------+----------+
5 rows in set (0.00 sec)
```

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



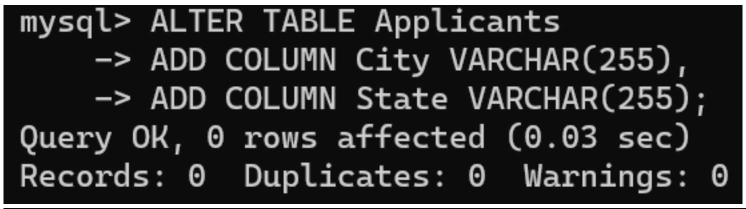
```
mysql> SELECT A.FirstName, A.LastName, COALESCE(C.CompanyName, 'Not Applied') AS CompanyName, COALESCE(J.JobTitle, 'Not Applied') AS
JobTitle
    -> FROM Applicants A
    -> LEFT JOIN Applications Ap ON A.ApplicantID = Ap.ApplicantID
    -> LEFT JOIN Jobs J ON Ap.JobID = J.JobID
    -> LEFT JOIN Companies C ON J.CompanyID = C.CompanyID;
+-----------+----------+------------------+----------------------+
| FirstName | LastName | CompanyName      | JobTitle             |
+-----------+----------+------------------+----------------------+
| John      | Doe      | XYZ Corporation  | Data Analyst         |
| Jane      | Smith    | ABC Inc.         | Software Engineer    |
| David     | Johnson  | ABC Inc.         | Marketing Specialist |
| Emily     | Brown    | LMN Technologies | IT Support Specialist |
| Michael   | Davis    | PQR Solutions    | Graphic Designer     |
+-----------+----------+------------------+----------------------+
5 rows in set (0.00 sec)
```

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

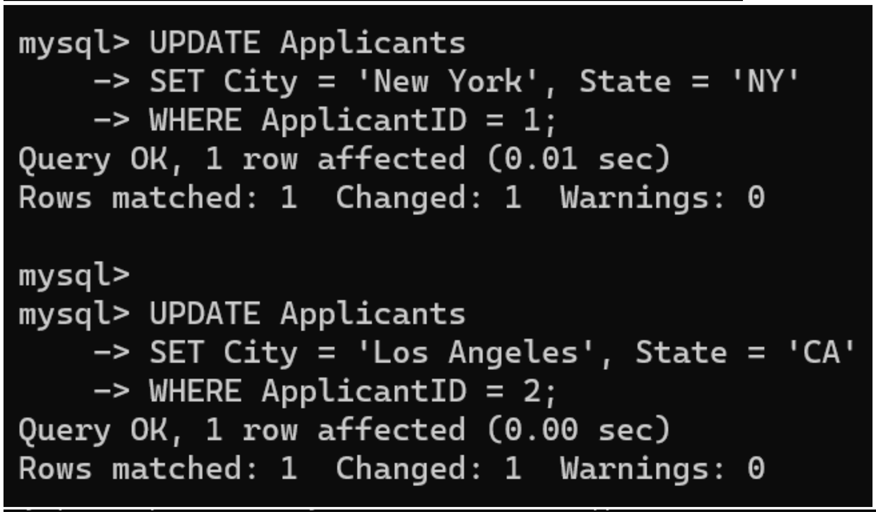


```
mysql> SELECT C.CompanyName
    -> FROM Companies C
    -> INNER JOIN Jobs J ON C.CompanyID = J.CompanyID
    -> WHERE J.Salary > (SELECT AVG(Salary) FROM Jobs WHERE Salary > 0);
+-----------------+
| CompanyName     |
+-----------------+
| ABC Inc.        |
| XYZ Corporation |
+-----------------+
2 rows in set (0.00 sec)
```

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

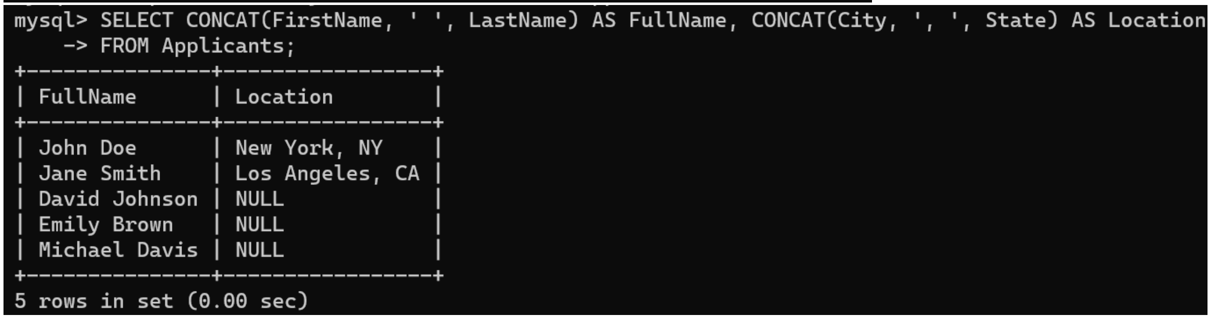


```
mysql> ALTER TABLE Applicants
    -> ADD COLUMN City VARCHAR(255),
    -> ADD COLUMN State VARCHAR(255);
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0
```



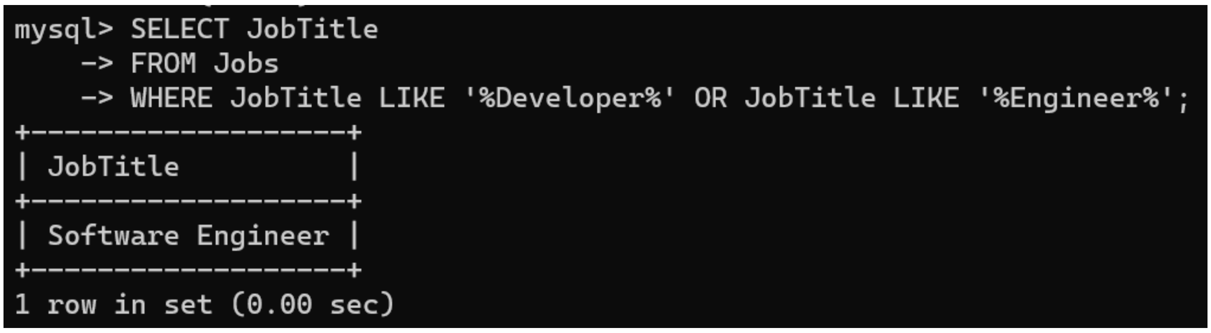
```
mysql> UPDATE Applicants
    -> SET City = 'New York', State = 'NY'
    -> WHERE ApplicantID = 1;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql>
mysql> UPDATE Applicants
    -> SET City = 'Los Angeles', State = 'CA'
    -> WHERE ApplicantID = 2;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```



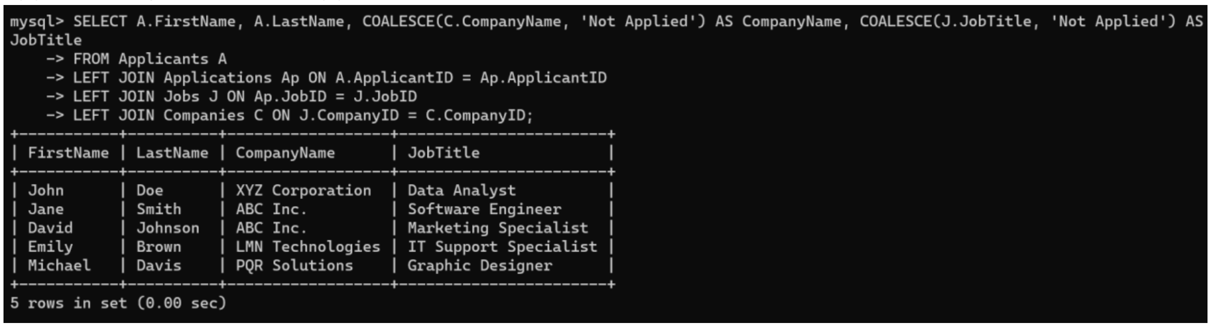
```
mysql> SELECT CONCAT(FirstName, ' ', LastName) AS FullName, CONCAT(City, ', ', State) AS Location
    -> FROM Applicants;
+---------------+-----------------+
| FullName      | Location        |
+---------------+-----------------+
| John Doe      | New York, NY    |
| Jane Smith    | Los Angeles, CA |
| David Johnson | NULL            |
| Emily Brown   | NULL            |
| Michael Davis | NULL            |
+---------------+-----------------+
5 rows in set (0.00 sec)
```

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



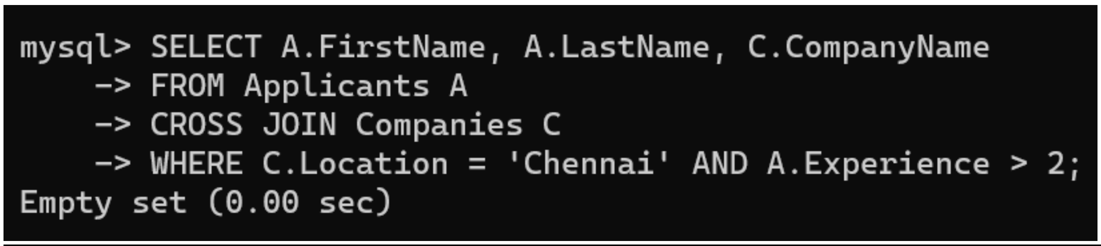
```
mysql> SELECT JobTitle
    -> FROM Jobs
    -> WHERE JobTitle LIKE '%Developer%' OR JobTitle LIKE '%Engineer%';
+-------------------+
| JobTitle          |
+-------------------+
| Software Engineer |
+-------------------+
1 row in set (0.00 sec)
```

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

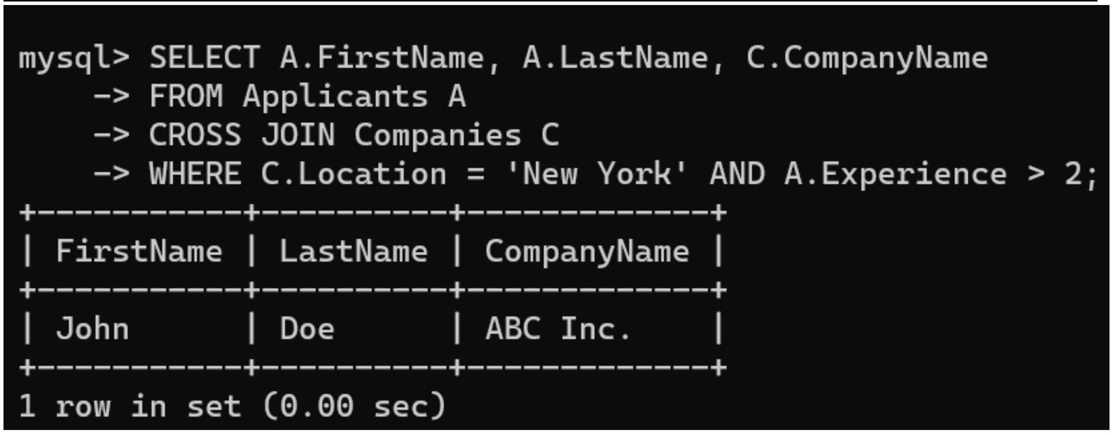


```
mysql> SELECT A.FirstName, A.LastName, COALESCE(C.CompanyName, 'Not Applied') AS CompanyName, COALESCE(J.JobTitle, 'Not Applied') AS
JobTitle
    -> FROM Applicants A
    -> LEFT JOIN Applications Ap ON A.ApplicantID = Ap.ApplicantID
    -> LEFT JOIN Jobs J ON Ap.JobID = J.JobID
    -> LEFT JOIN Companies C ON J.CompanyID = C.CompanyID;
+-----------+----------+------------------+----------------------+
| FirstName | LastName | CompanyName      | JobTitle             |
+-----------+----------+------------------+----------------------+
| John      | Doe      | XYZ Corporation  | Data Analyst         |
| Jane      | Smith    | ABC Inc.         | Software Engineer    |
| David     | Johnson  | ABC Inc.         | Marketing Specialist |
| Emily     | Brown    | LMN Technologies | IT Support Specialist |
| Michael   | Davis    | PQR Solutions    | Graphic Designer     |
+-----------+----------+------------------+----------------------+
5 rows in set (0.00 sec)
```

18. 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



```
mysql> SELECT A.FirstName, A.LastName, C.CompanyName
    -> FROM Applicants A
    -> CROSS JOIN Companies C
    -> WHERE C.Location = 'Chennai' AND A.Experience > 2;
Empty set (0.00 sec)
```



```
mysql> SELECT A.FirstName, A.LastName, C.CompanyName
    -> FROM Applicants A
    -> CROSS JOIN Companies C
    -> WHERE C.Location = 'New York' AND A.Experience > 2;
+-----------+----------+-------------+
| FirstName | LastName | CompanyName |
+-----------+----------+-------------+
| John      | Doe      | ABC Inc.    |
+-----------+----------+-------------+
1 row in set (0.00 sec)
```