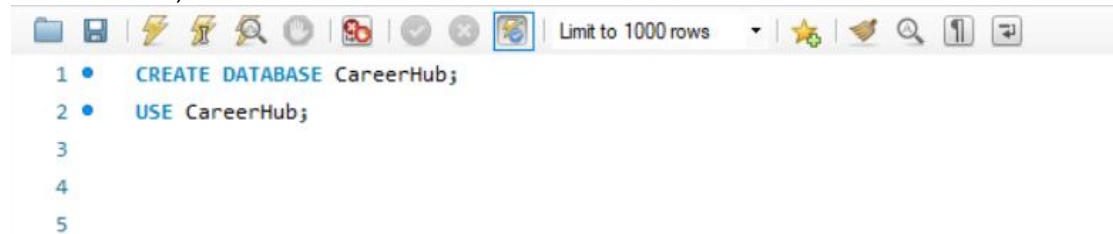


## Coding Challenges: CareerHub, The Job Board

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

```
CREATE DATABASE CareerHub;  
USE CareerHub;
```



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

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

```
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)  
);
```

```

1 CREATE TABLE Companies (
2     CompanyID INT PRIMARY KEY,
3     CompanyName VARCHAR(255),
4     Location VARCHAR(255)
5 );
6 CREATE TABLE Jobs (
7     JobID INT PRIMARY KEY,
8     CompanyID INT,
9     FOREIGN KEY (CompanyID) REFERENCES Companies(CompanyID),
10    JobTitle VARCHAR(255),
11    JobDescription TEXT,
12    JobLocation VARCHAR(255),
13    Salary DECIMAL,
14    JobType VARCHAR(255),
15    PostedDate DATETIME,
16    CONSTRAINT FK_JobCompany FOREIGN KEY (CompanyID) REFERENCES
17    Companies(CompanyID)
18 );

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [IA](#)

Tables_in_careerhub	
▶	applicants
	applications
	companies
	jobs

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

```

```

1 SELECT
2     J.JobID,
3     J.JobTitle,
4     COUNT(A.ApplicationID) AS ApplicationCount
5 FROM
6     Jobs J
7 LEFT JOIN
8     Applications A ON J.JobID = A.JobID
9 GROUP BY
10    J.JobID, J.JobTitle;

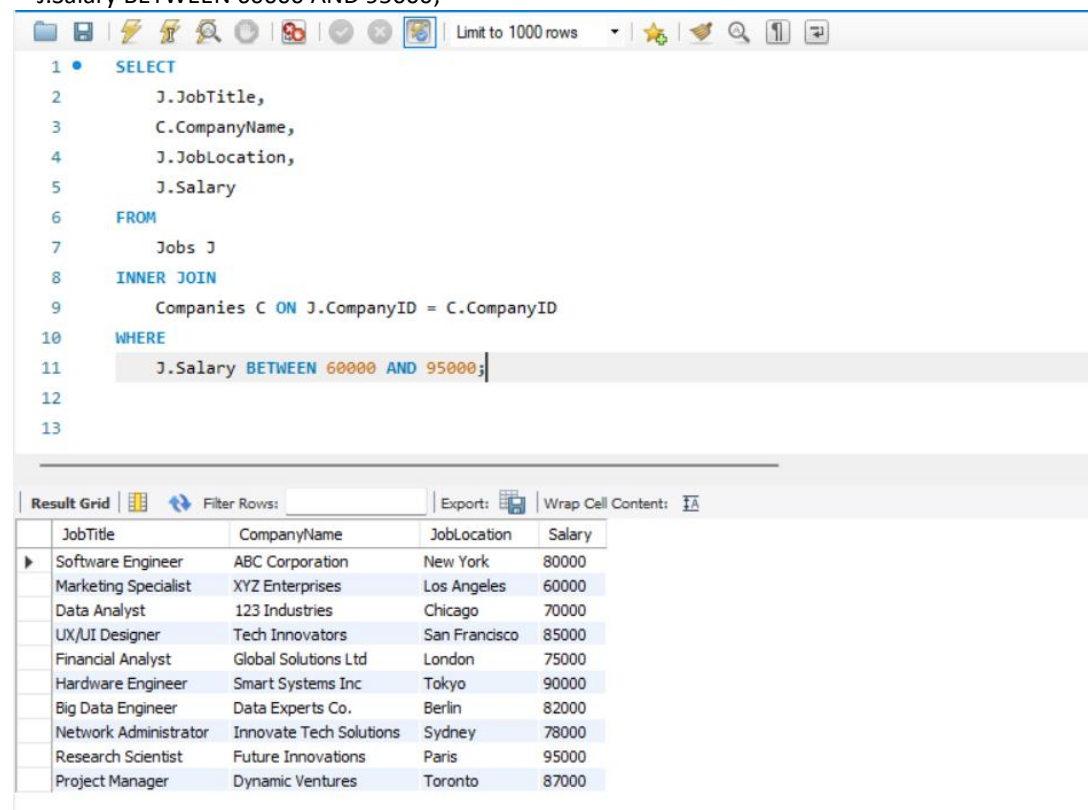
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [IA](#)

	JobID	JobTitle	ApplicationCount
▶	1	Software Engineer	1
	2	Marketing Specialist	1
	3	Data Analyst	1
	4	UX/UI Designer	1
	5	Financial Analyst	1
	6	Hardware Engineer	1
	7	Big Data Engineer	1
	8	Network Administrator	1
	9	Research Scientist	1
	10	Project Manager	1

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
INNER JOIN
    Companies C ON J.CompanyID = C.CompanyID
WHERE
    J.Salary BETWEEN 60000 AND 95000;
```



The screenshot shows a SQL query editor with the following query:

```
1 • SELECT
2     J.JobTitle,
3     C.CompanyName,
4     J.JobLocation,
5     J.Salary
6 FROM
7     Jobs J
8 INNER JOIN
9     Companies C ON J.CompanyID = C.CompanyID
10 WHERE
11     J.Salary BETWEEN 60000 AND 95000;
```

Below the query editor is a "Result Grid" showing the results of the query. The grid has 5 columns: JobTitle, CompanyName, JobLocation, and Salary. The results are as follows:

JobTitle	CompanyName	JobLocation	Salary
Software Engineer	ABC Corporation	New York	80000
Marketing Specialist	XYZ Enterprises	Los Angeles	60000
Data Analyst	123 Industries	Chicago	70000
UX/UI Designer	Tech Innovators	San Francisco	85000
Financial Analyst	Global Solutions Ltd	London	75000
Hardware Engineer	Smart Systems Inc	Tokyo	90000
Big Data Engineer	Data Experts Co.	Berlin	82000
Network Administrator	Innovate Tech Solutions	Sydney	78000
Research Scientist	Future Innovations	Paris	95000
Project Manager	Dynamic Ventures	Toronto	87000

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
INNER JOIN
    Jobs J ON A.JobID = J.JobID
INNER JOIN
    Companies C ON J.CompanyID = C.CompanyID
WHERE
    A.ApplicantID = 1;
```

```

1 • SELECT
2     J.JobTitle,
3     C.CompanyName,
4     A.ApplicationDate
5 FROM
6     Applications A
7 INNER JOIN
8     Jobs J ON A.JobID = J.JobID
9 INNER JOIN
10    Companies C ON J.CompanyID = C.CompanyID
11 WHERE
12     A.ApplicantID = 1;
13

```

Result Grid

JobTitle	CompanyName	ApplicationDate
Software Engineer	ABC Corporation	2024-01-24 09:30:00

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) AS AverageSalary
FROM
    Jobs
WHERE
    Salary > 0;

```

```

1 • SELECT
2     AVG(Salary) AS AverageSalary
3 FROM
4     Jobs
5 WHERE
6     Salary > 0;
7

```

Result Grid

AverageSalary
80200.0000

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.

```

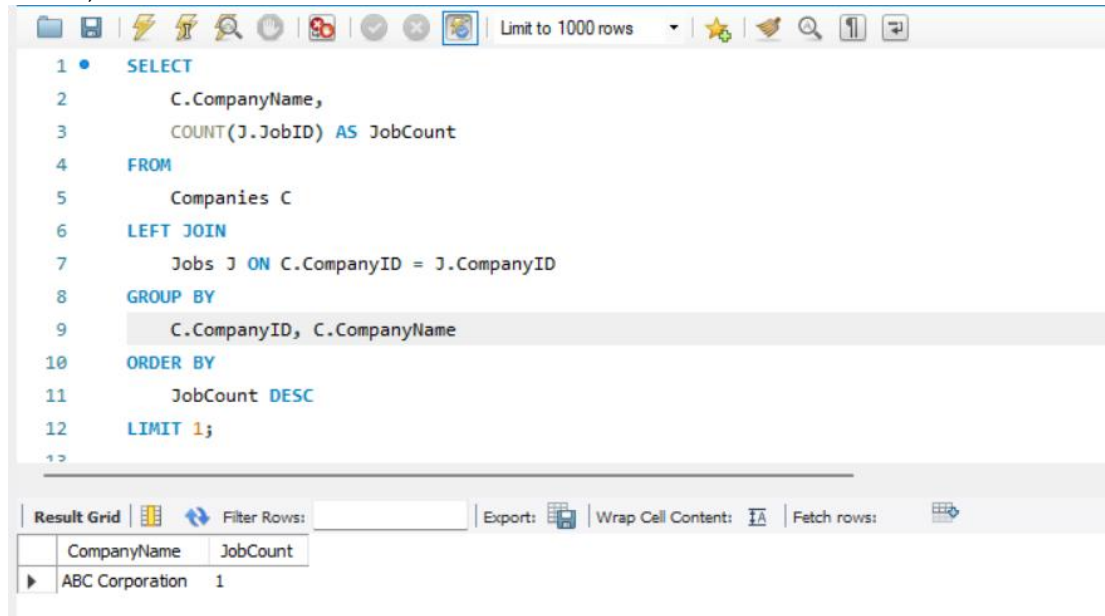
SELECT
    C.CompanyName,
    COUNT(J.JobID) AS JobCount
FROM
    Companies C

```

```

LEFT JOIN
    Jobs J ON C.CompanyID = J.CompanyID
GROUP BY
    C.CompanyID, C.CompanyName
ORDER BY
    JobCount DESC
LIMIT 1;

```



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

```

SELECT
    A.ApplicantID,
    A.FirstName,
    A.LastName,
    A.Email,
    A.Phone
FROM
    Applicants A
WHERE
    A.ApplicantID IN (
        Select
            Ap.ApplicantID
        FROM
            Applications Ap
        INNER JOIN
            Jobs J ON Ap.JobID = J.JobID
        INNER JOIN
            Companies C ON J.CompanyID = C.CompanyID
        WHERE
            J.JobLocation = 'Tokyo'
            AND DATEDIFF(CURDATE(), J.PostedDate) >= 1095
    );

```

The screenshot shows a SQL query in the Enterprise Manager interface. The query is as follows:

```

4      A.LastName,
5      A.Email,
6      A.Phone
7  FROM
8      Applicants A
9  WHERE
10     A.ApplicantID IN (
11         Select
12             Ap.ApplicantID
13         FROM
14             Applications Ap
15         INNER JOIN
16             Jobs J ON Ap.JobID = J.JobID
17         INNER JOIN
18             Companies C ON J.CompanyID = C.CompanyID
19         WHERE
20             J.JobLocation = 'Tokyo'
21             AND DATEDIFF(CURDATE(), J.PostedDate) >= 1095
22     );

```

Below the query, the 'Result Grid' is displayed with the following columns: ApplicantID, FirstName, LastName, Email, and Phone. The first row shows all NULL values.

ApplicantID	FirstName	LastName	Email	Phone
NULL	NULL	NULL	NULL	NULL

**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;

```

The screenshot shows the same SQL query in the Enterprise Manager interface. The query is as follows:

```

1  SELECT DISTINCT
2      JobTitle
3  FROM
4      Jobs
5  WHERE
6      Salary BETWEEN 60000 AND 80000;

```

Below the query, the 'Result Grid' is displayed with the following columns: JobTitle. The results are as follows:

JobTitle
Software Engineer
Marketing Specialist
Data Analyst
Financial Analyst
Network Administrator

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

```

SELECT
    J.JobID,
    J.JobTitle,
    J.JobLocation
FROM
    Jobs J

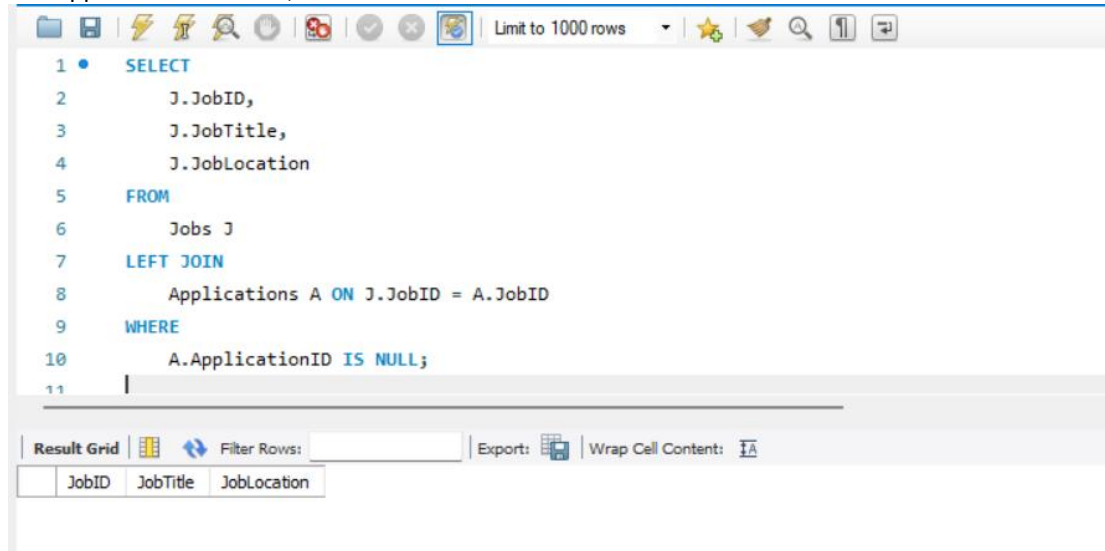
```

LEFT JOIN

Applications A ON J.JobID = A.JobID

WHERE

A.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.ApplicantID,

A.FirstName,

A.LastName,

A.Email,

A.Phone,

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;

<

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.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;

```



1	•	SELECT
2		C.CompanyID,
3		C.CompanyName,
4		COUNT(J.JobID) AS JobCount
5		FROM
6		Companies C
7		LEFT JOIN
8		Jobs J ON C.CompanyID = J.CompanyID
9		LEFT JOIN
10		Applications A ON J.JobID = A.JobID
11		GROUP BY
12		C.CompanyID, C.CompanyName;

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
CompanyID	CompanyName	JobCount	
1	ABC Corporation	1	
2	XYZ Enterprises	1	
3	123 Industries	1	
4	Tech Innovators	1	
5	Global Solutions Ltd	1	
6	Smart Systems Inc	1	
7	Data Experts Co.	1	
8	Innovate Tech Solutions	1	
9	Future Innovations	1	
10	Dynamic Ventures	1	

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

```

SELECT
    A.ApplicantID,
    A.FirstName,
    A.LastName,
    A.Email,
    A.Phone,
    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;

```

1	•	SELECT
2		A.ApplicantID,
3		A.FirstName,
4		A.LastName,
5		A.Email,
6		A.Phone,
7		COALESCE(C.CompanyName, 'Not Applied') As CompanyName,
8		COALESCE(J.JobTitle, 'Not Applied') As JobTitle
9		FROM
10		Applicants A
11		LEFT JOIN
12		Applications Ap ON A.ApplicantID = Ap.ApplicantID
13		LEFT JOIN
14		Jobs J ON Ap.JobID = J.JobID
15		LEFT JOIN
16		Companies C ON J.CompanyID = C.CompanyID;

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
ApplicantID	FirstName	LastName	Email
1	John	Doe	john.doe@email.com
2	Jane	Smith	jane.smith@email.com
3	Mike	Johnson	mike.johnson@email.com
4	Emily	Williams	emily.williams@email.com
5	Chris	Taylor	chris.taylor@email.com
6	Amanda	Clark	amanda.clark@email.com
7	Daniel	Brown	daniel.brown@email.com
8	Olivia	White	olivia.white@email.com
9	Ethan	Miller	ethan.miller@email.com
10	Sophia	Davis	sophia.davis@email.com

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

```

SELECT
    C.CompanyID,
    C.CompanyName
FROM
    Companies C
JOIN
    Jobs J ON C.CompanyID = J.CompanyID
GROUP BY
    C.CompanyID, C.CompanyName
HAVING
    MAX(J.Salary) > (SELECT AVG(Salary) FROM Jobs);

```

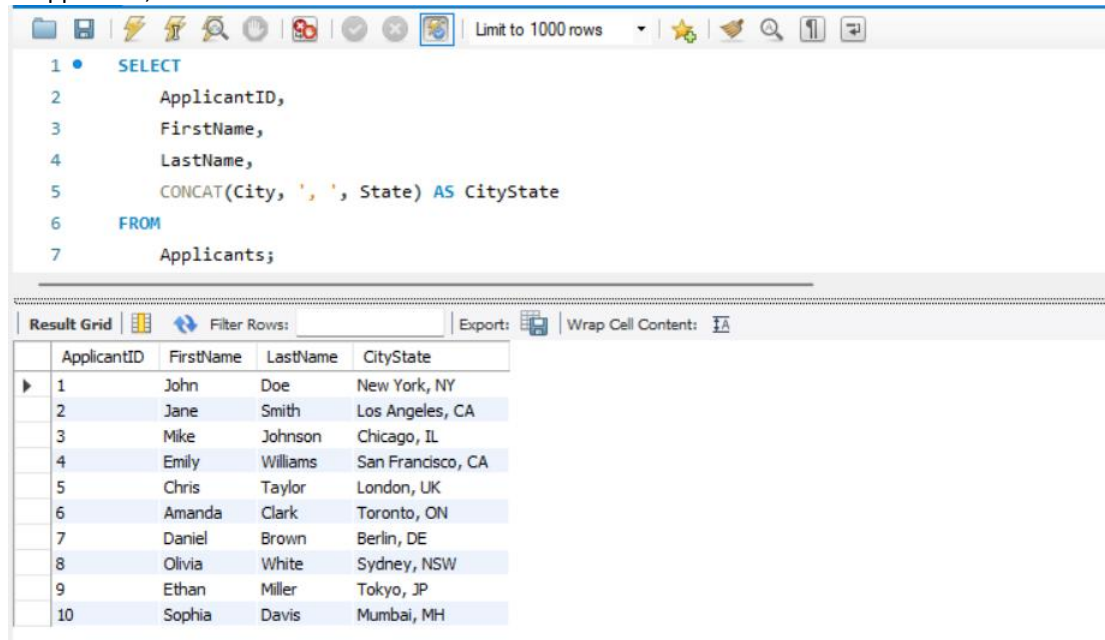
1	•	SELECT
2		C.CompanyID,
3		C.CompanyName
4		FROM
5		Companies C
6		JOIN
7		Jobs J ON C.CompanyID = J.CompanyID
8		GROUP BY
9		C.CompanyID, C.CompanyName
10		HAVING
11		MAX(J.Salary) > (SELECT AVG(Salary) FROM Jobs);
12		

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
CompanyID	CompanyName		
4	Tech Innovators		
6	Smart Systems Inc		
7	Data Experts Co.		
9	Future Innovations		
10	Dynamic Ventures		

17. 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 CityState
FROM
  Applicants;
```



The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and a 'Limit to 1000 rows' dropdown. The SQL editor contains the following query:

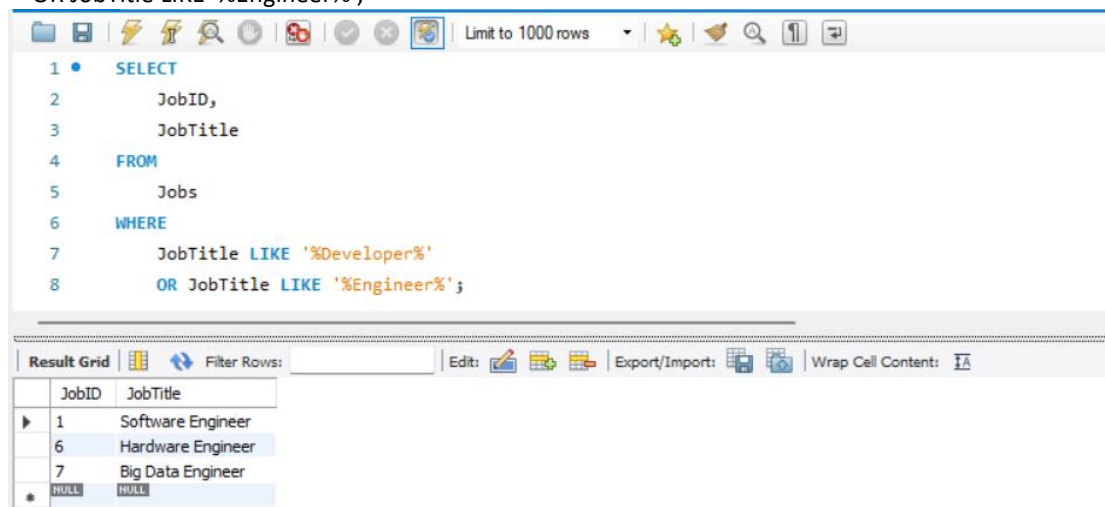
```
1 • SELECT
2     ApplicantID,
3     FirstName,
4     LastName,
5     CONCAT(City, ', ', State) AS CityState
6 FROM
7     Applicants;
```

Below the editor is the 'Result Grid' tab, which displays the query results in a table with 5 columns: ApplicantID, FirstName, LastName, and CityState. The results are as follows:

	ApplicantID	FirstName	LastName	CityState
▶	1	John	Doe	New York, NY
	2	Jane	Smith	Los Angeles, CA
	3	Mike	Johnson	Chicago, IL
	4	Emily	Williams	San Francisco, CA
	5	Chris	Taylor	London, UK
	6	Amanda	Clark	Toronto, ON
	7	Daniel	Brown	Berlin, DE
	8	Olivia	White	Sydney, NSW
	9	Ethan	Miller	Tokyo, JP
	10	Sophia	Davis	Mumbai, MH

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%';
```



The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and a 'Limit to 1000 rows' dropdown. The SQL editor contains the following query:

```
1 • SELECT
2     JobID,
3     JobTitle
4 FROM
5     Jobs
6 WHERE
7     JobTitle LIKE '%Developer%'
8     OR JobTitle LIKE '%Engineer%';
```

Below the editor is the 'Result Grid' tab, which displays the query results in a table with 2 columns: JobID and JobTitle. The results are as follows:

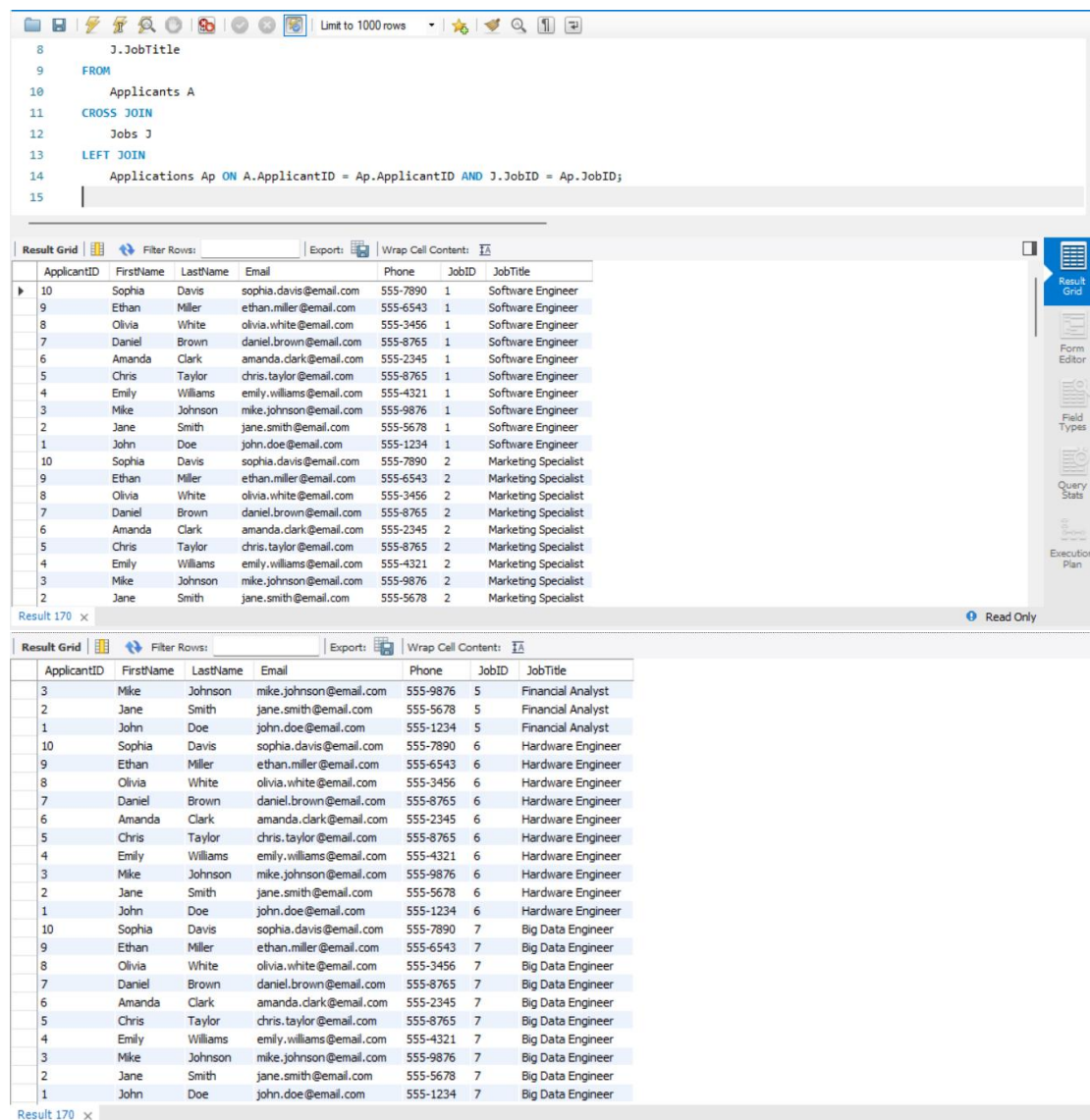
	JobID	JobTitle
▶	1	Software Engineer
	6	Hardware Engineer
	7	Big Data Engineer
*	NULL	NULL

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,
  A.Email,
  A.Phone,
  J.JobID,
  J.JobTitle
FROM
  Applicants A
CROSS JOIN
  Jobs J
LEFT JOIN
  Applications Ap ON A.ApplicantID = Ap.ApplicantID AND J.JobID = Ap.JobID;

```



The screenshot shows a database query editor with the following SQL query:

```

8      J.JobTitle
9      FROM
10     Applicants A
11     CROSS JOIN
12     Jobs J
13     LEFT JOIN
14     Applications Ap ON A.ApplicantID = Ap.ApplicantID AND J.JobID = Ap.JobID;
15

```

The results are displayed in a grid format, showing 170 rows. The grid is divided into two sections, each with 170 rows. The first section shows the results of the query, and the second section shows the results of the query with the 'Read Only' flag set.

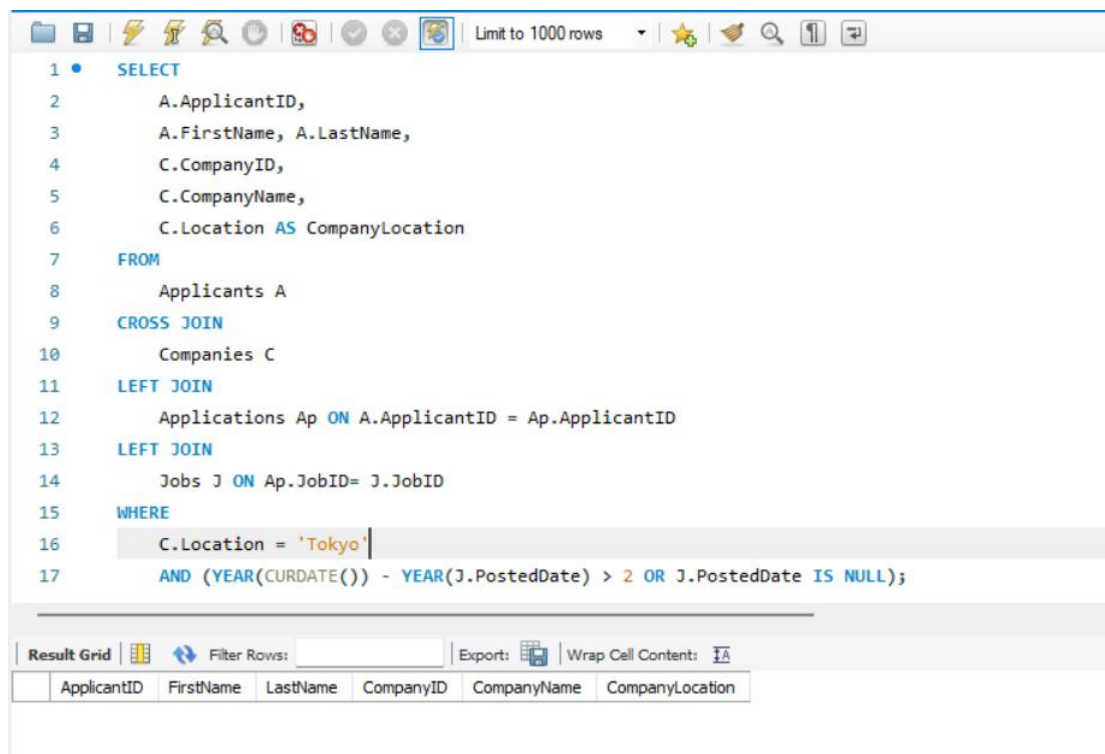
ApplicantID	FirstName	LastName	Email	Phone	JobID	JobTitle
10	Sophia	Davis	sophia.davis@email.com	555-7890	1	Software Engineer
9	Ethan	Miller	ethan.miller@email.com	555-6543	1	Software Engineer
8	Olivia	White	olivia.white@email.com	555-3456	1	Software Engineer
7	Daniel	Brown	daniel.brown@email.com	555-8765	1	Software Engineer
6	Amanda	Clark	amanda.clark@email.com	555-2345	1	Software Engineer
5	Chris	Taylor	chris.taylor@email.com	555-8765	1	Software Engineer
4	Emily	Williams	emily.williams@email.com	555-4321	1	Software Engineer
3	Mike	Johnson	mike.johnson@email.com	555-9876	1	Software Engineer
2	Jane	Smith	jane.smith@email.com	555-5678	1	Software Engineer
1	John	Doe	john.doe@email.com	555-1234	1	Software Engineer
10	Sophia	Davis	sophia.davis@email.com	555-7890	2	Marketing Specialist
9	Ethan	Miller	ethan.miller@email.com	555-6543	2	Marketing Specialist
8	Olivia	White	olivia.white@email.com	555-3456	2	Marketing Specialist
7	Daniel	Brown	daniel.brown@email.com	555-8765	2	Marketing Specialist
6	Amanda	Clark	amanda.clark@email.com	555-2345	2	Marketing Specialist
5	Chris	Taylor	chris.taylor@email.com	555-8765	2	Marketing Specialist
4	Emily	Williams	emily.williams@email.com	555-4321	2	Marketing Specialist
3	Mike	Johnson	mike.johnson@email.com	555-9876	2	Marketing Specialist
2	Jane	Smith	jane.smith@email.com	555-5678	2	Marketing Specialist

ApplicantID	FirstName	LastName	Email	Phone	JobID	JobTitle
3	Mike	Johnson	mike.johnson@email.com	555-9876	5	Financial Analyst
2	Jane	Smith	jane.smith@email.com	555-5678	5	Financial Analyst
1	John	Doe	john.doe@email.com	555-1234	5	Financial Analyst
10	Sophia	Davis	sophia.davis@email.com	555-7890	6	Hardware Engineer
9	Ethan	Miller	ethan.miller@email.com	555-6543	6	Hardware Engineer
8	Olivia	White	olivia.white@email.com	555-3456	6	Hardware Engineer
7	Daniel	Brown	daniel.brown@email.com	555-8765	6	Hardware Engineer
6	Amanda	Clark	amanda.clark@email.com	555-2345	6	Hardware Engineer
5	Chris	Taylor	chris.taylor@email.com	555-8765	6	Hardware Engineer
4	Emily	Williams	emily.williams@email.com	555-4321	6	Hardware Engineer
3	Mike	Johnson	mike.johnson@email.com	555-9876	6	Hardware Engineer
2	Jane	Smith	jane.smith@email.com	555-5678	6	Hardware Engineer
1	John	Doe	john.doe@email.com	555-1234	6	Hardware Engineer
10	Sophia	Davis	sophia.davis@email.com	555-7890	7	Big Data Engineer
9	Ethan	Miller	ethan.miller@email.com	555-6543	7	Big Data Engineer
8	Olivia	White	olivia.white@email.com	555-3456	7	Big Data Engineer
7	Daniel	Brown	daniel.brown@email.com	555-8765	7	Big Data Engineer
6	Amanda	Clark	amanda.clark@email.com	555-2345	7	Big Data Engineer
5	Chris	Taylor	chris.taylor@email.com	555-8765	7	Big Data Engineer
4	Emily	Williams	emily.williams@email.com	555-4321	7	Big Data Engineer
3	Mike	Johnson	mike.johnson@email.com	555-9876	7	Big Data Engineer
2	Jane	Smith	jane.smith@email.com	555-5678	7	Big Data Engineer
1	John	Doe	john.doe@email.com	555-1234	7	Big Data Engineer

**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**

```
SELECT
    A.ApplicantID,
    A.FirstName, A.LastName,
    C.CompanyID,
    C.CompanyName,
    C.Location AS CompanyLocation
FROM
    Applicants A
CROSS JOIN
    Companies C
LEFT JOIN
    Applications Ap ON A.ApplicantID = Ap.ApplicantID
LEFT JOIN
    Jobs J ON Ap.JobID= J.JobID
WHERE
    C.Location = 'Tokyo'
    AND (YEAR(CURDATE()) - YEAR(J.PostedDate) > 2 OR J.PostedDate IS NULL);
```



The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and a 'Limit to 1000 rows' dropdown. The SQL editor contains the following query:

```
1 • SELECT
2     A.ApplicantID,
3     A.FirstName, A.LastName,
4     C.CompanyID,
5     C.CompanyName,
6     C.Location AS CompanyLocation
7 FROM
8     Applicants A
9 CROSS JOIN
10    Companies C
11 LEFT JOIN
12    Applications Ap ON A.ApplicantID = Ap.ApplicantID
13 LEFT JOIN
14    Jobs J ON Ap.JobID= J.JobID
15 WHERE
16     C.Location = 'Tokyo'
17     AND (YEAR(CURDATE()) - YEAR(J.PostedDate) > 2 OR J.PostedDate IS NULL);
```

Below the editor is the 'Result Grid' section. It includes a 'Filter Rows' input field, an 'Export' button, and a 'Wrap Cell Content' checkbox. The result grid table has the following columns:

ApplicantID	FirstName	LastName	CompanyID	CompanyName	CompanyLocation
-------------	-----------	----------	-----------	-------------	-----------------