

HEXAWARE-PYTHON FOUNDATION TRAINING
CODING CHALLENGE – MYSQL
CARRER HUB

Name: Mathew P

Batch: 04

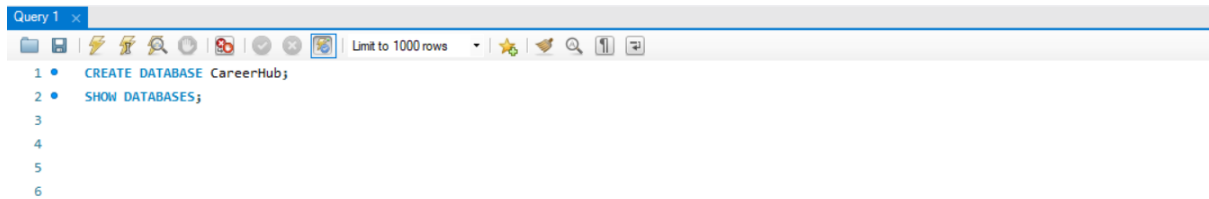
TASKS

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

Query:

```
CREATE DATABASE CareerHub;
```

```
SHOW DATABASES;
```



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

Query:

```
USE CareerHub;
```

```
CREATE TABLE Companies
```

```
(
```

```
CompanyID INT,
```

```
CompanyName VARCHAR(50)NOT NULL,
```

```
Location VARCHAR(200)
```

```
);
```

```
CREATE TABLE Jobs
```

```
(
```

```
JobID INT,
```

```
CompanyID INT,  
JobTitle varchar(100),  
JobDescription TEXT,  
JobLocation VARCHAR(200),  
Salary DECIMAL(10,2) NOT NULL,  
JobType VARCHAR(100),  
PostedDate DATETIME  
);
```

```
CREATE TABLE Applicants  
(  
ApplicantID INT,  
FirstName VARCHAR(100) NOT NULL,  
LastName VARCHAR(100) ,  
Email VARCHAR(200) UNIQUE NOT NULL,  
Phone VARCHAR(20),  
Resume TEXT  
);
```

```
CREATE TABLE Applications  
(  
ApplicationID INT,  
JobID INT,  
ApplicantID INT,  
ApplicationDate DATETIME,  
CoverLetter TEXT  
);
```

```
SHOW TABLES;
```

```
Query 1 x
Limit to 1000 rows

1 • USE CareerHub;
2 • CREATE TABLE Companies
3 • (
4 •     CompanyID INT,
5 •     CompanyName VARCHAR(50) NOT NULL,
6 •     Location VARCHAR(200)
7 • );
8
9 • CREATE TABLE Jobs
10 • (
11 •     JobID INT,
12 •     CompanyID INT,
13 •     JobTitle varchar(100),
14 •     JobDescription TEXT,
15 •     JobLocation VARCHAR(200),
16 •     Salary DECIMAL(10,2) NOT NULL,
17 •     JobType VARCHAR(100),
18 •     PostedDate DATETIME
19 • );
20
21 • CREATE TABLE Applicants
22 • (
23 •     ApplicantID INT,
24 •     FirstName VARCHAR(100) NOT NULL,
25 •     LastName VARCHAR(100) ,
26 •     Email VARCHAR(200) UNIQUE NOT NULL,
27 •     Phone VARCHAR(20),
28 •     Resume TEXT
29 • );
30
31 • CREATE TABLE Applications
32 • (
33 •     ApplicationID INT,
34 •     JobID INT,
35 •     ApplicantID INT,
36 •     ApplicationDate DATETIME,
37 •     CoverLetter TEXT
38 • );
39
40 • SHOW TABLES;
41
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

Tables_in_careerhub
applicants
applications
companies
jobs

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

Query:

Primary Key and Foreign Key:

ALTER TABLE Companies ADD CONSTRAINT PRIMARY KEY(CompanyID);

ALTER TABLE Jobs ADD CONSTRAINT PRIMARY KEY(JobID);

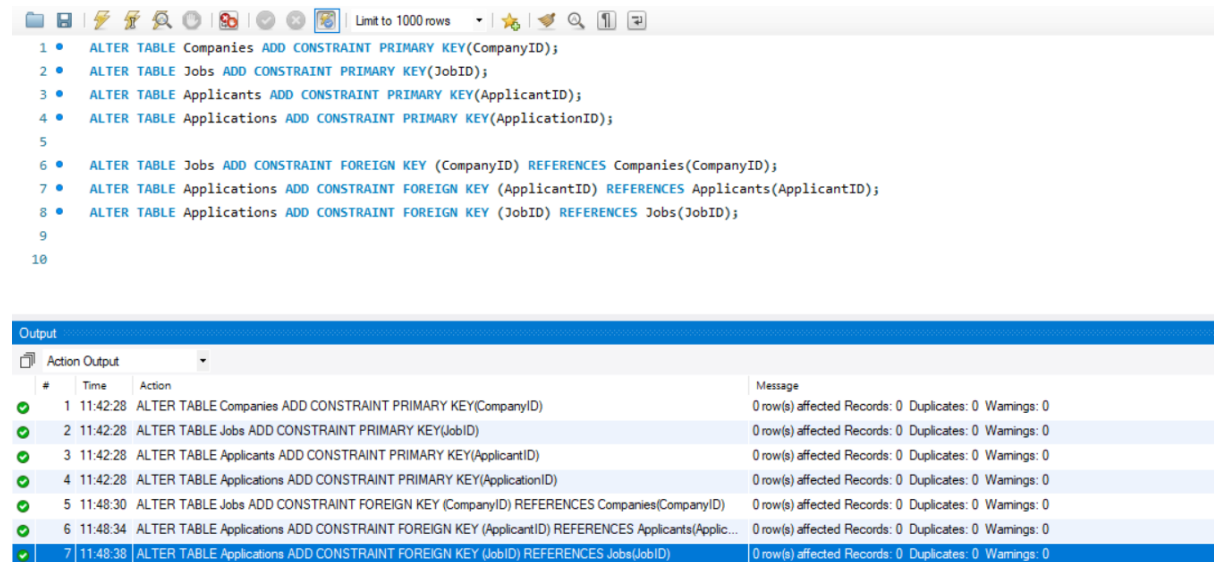
ALTER TABLE Applicants ADD CONSTRAINT PRIMARY KEY(ApplicantID);

ALTER TABLE Applications ADD CONSTRAINT PRIMARY KEY(ApplicationID);

ALTER TABLE Jobs ADD CONSTRAINT FOREIGN KEY (CompanyID) REFERENCES Companies(CompanyID);

ALTER TABLE Applications ADD CONSTRAINT FOREIGN KEY (ApplicantID) REFERENCES Applicants(ApplicantID);

ALTER TABLE Applications ADD CONSTRAINT FOREIGN KEY (JobID) REFERENCES Jobs(JobID);



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

```
1 ALTER TABLE Companies ADD CONSTRAINT PRIMARY KEY(CompanyID);
2 ALTER TABLE Jobs ADD CONSTRAINT PRIMARY KEY(JobID);
3 ALTER TABLE Applicants ADD CONSTRAINT PRIMARY KEY(ApplicantID);
4 ALTER TABLE Applications ADD CONSTRAINT PRIMARY KEY(ApplicationID);
5
6 ALTER TABLE Jobs ADD CONSTRAINT FOREIGN KEY (CompanyID) REFERENCES Companies(CompanyID);
7 ALTER TABLE Applications ADD CONSTRAINT FOREIGN KEY (ApplicantID) REFERENCES Applicants(ApplicantID);
8 ALTER TABLE Applications ADD CONSTRAINT FOREIGN KEY (JobID) REFERENCES Jobs(JobID);
9
10
```

The bottom output window, titled 'Output', shows the execution results of these statements:

#	Time	Action	Message
✓ 1	11:42:28	ALTER TABLE Companies ADD CONSTRAINT PRIMARY KEY(CompanyID)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
✓ 2	11:42:28	ALTER TABLE Jobs ADD CONSTRAINT PRIMARY KEY(JobID)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
✓ 3	11:42:28	ALTER TABLE Applicants ADD CONSTRAINT PRIMARY KEY(ApplicantID)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
✓ 4	11:42:28	ALTER TABLE Applications ADD CONSTRAINT PRIMARY KEY(ApplicationID)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
✓ 5	11:48:30	ALTER TABLE Jobs ADD CONSTRAINT FOREIGN KEY (CompanyID) REFERENCES Companies(CompanyID)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
✓ 6	11:48:34	ALTER TABLE Applications ADD CONSTRAINT FOREIGN KEY (ApplicantID) REFERENCES Applicants(ApplicantID)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
✓ 7	11:48:38	ALTER TABLE Applications ADD CONSTRAINT FOREIGN KEY (JobID) REFERENCES Jobs(JobID)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0

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

Query:

CREATE DATABASE IF NOT EXISTS CareerHub;

USE CareerHub;

DROP TABLE IF EXISTS Applications;

DROP TABLE IF EXISTS Jobs;

DROP TABLE IF EXISTS Applicants;

DROP TABLE IF EXISTS Companies;

CREATE TABLE IF NOT EXISTS Companies (

CompanyID INT PRIMARY KEY,

CompanyName VARCHAR(100) NOT NULL,

Location VARCHAR(100) NOT NULL
);

CREATE TABLE IF NOT EXISTS Applicants (
ApplicantID INT PRIMARY KEY,
FirstName VARCHAR(50) NOT NULL,
LastName VARCHAR(50) NOT NULL,
Email VARCHAR(100) UNIQUE NOT NULL,
Phone VARCHAR(15),
Resume TEXT,
City VARCHAR(50),
State VARCHAR(50)
);

CREATE TABLE IF NOT EXISTS Jobs (
JobID INT PRIMARY KEY,
CompanyID INT,
JobTitle VARCHAR(100) NOT NULL,
JobDescription TEXT,
JobLocation VARCHAR(100),
Salary DECIMAL(10, 2) CHECK (Salary >= 0),
JobType VARCHAR(50),
PostedDate DATETIME,
FOREIGN KEY (CompanyID) REFERENCES Companies(CompanyID)
ON DELETE CASCADE
ON UPDATE CASCADE
);

CREATE TABLE IF NOT EXISTS Applications (

```

ApplicationID INT PRIMARY KEY,

JobID INT,

ApplicantID INT,

ApplicationDate DATETIME,

CoverLetter TEXT,

FOREIGN KEY (JobID) REFERENCES Jobs(JobID)

    ON DELETE CASCADE

    ON UPDATE CASCADE,

FOREIGN KEY (ApplicantID) REFERENCES Applicants(ApplicantID)

    ON DELETE CASCADE

    ON UPDATE CASCADE

);

```

The screenshot shows a SQL query editor window titled "Query 1". The script contains the following SQL statements:

```

1
2 • CREATE DATABASE IF NOT EXISTS CareerHub;
3 • USE CareerHub;
4
5 • DROP TABLE IF EXISTS Applications;
6 • DROP TABLE IF EXISTS Jobs;
7 • DROP TABLE IF EXISTS Applicants;
8 • DROP TABLE IF EXISTS Companies;
9
10 • CREATE TABLE IF NOT EXISTS Companies (
11     CompanyID INT PRIMARY KEY,
12     CompanyName VARCHAR(100) NOT NULL,
13     Location VARCHAR(100) NOT NULL
14 );
15
16 • CREATE TABLE IF NOT EXISTS Applicants (
17     ApplicantID INT PRIMARY KEY,
18     FirstName VARCHAR(50) NOT NULL,
19     LastName VARCHAR(50) NOT NULL,
20     Email VARCHAR(100) UNIQUE NOT NULL,
21     Phone VARCHAR(15),
22     Resume TEXT,
23     City VARCHAR(50),
24     State VARCHAR(50)
25 );
--

```

The output window at the bottom shows the execution results:

#	Time	Action	Message
4	14:24:50	DROP TABLE IF EXISTS Applications	0 row(s) affected
5	14:24:50	DROP TABLE IF EXISTS Jobs	0 row(s) affected

```

);

CREATE TABLE IF NOT EXISTS Jobs (
    JobID INT PRIMARY KEY,
    CompanyID INT,
    JobTitle VARCHAR(100) NOT NULL,
    JobDescription TEXT,
    JobLocation VARCHAR(100),
    Salary DECIMAL(10, 2) CHECK (Salary >= 0),
    JobType VARCHAR(50),
    PostedDate DATETIME,
    FOREIGN KEY (CompanyID) REFERENCES Companies(CompanyID)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);

CREATE TABLE IF NOT EXISTS Applications (
    ApplicationID INT PRIMARY KEY,
    JobID INT,
    ApplicantID INT,
    ApplicationDate DATETIME,
    CoverLetter TEXT,
    FOREIGN KEY (JobID) REFERENCES Jobs(JobID)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    FOREIGN KEY (ApplicantID) REFERENCES Applicants(ApplicantID)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);

```

Automa disabled. manually current c toggle

Context Help

Action Output

#	Time	Action	Message
4	14:24:50	DROP TABLE IF EXISTS Applications	0 row(s) affected
5	14:24:50	DROP TABLE IF EXISTS Jobs	0 row(s) affected

Query 1

```

33 Salary DECIMAL(10, 2) CHECK (Salary >= 0),
34 JobType VARCHAR(50),
35 PostedDate DATETIME,
36 FOREIGN KEY (CompanyID) REFERENCES Companies(CompanyID)
37     ON DELETE CASCADE
38     ON UPDATE CASCADE
39 );
40
41 CREATE TABLE IF NOT EXISTS Applications (
42     ApplicationID INT PRIMARY KEY,
43     JobID INT,
44     ApplicantID INT,
45     ApplicationDate DATETIME,
46     CoverLetter TEXT,
47     FOREIGN KEY (JobID) REFERENCES Jobs(JobID)
48         ON DELETE CASCADE
49         ON UPDATE CASCADE,
50     FOREIGN KEY (ApplicantID) REFERENCES Applicants(ApplicantID)
51         ON DELETE CASCADE
52         ON UPDATE CASCADE
53 );
54
55
56
57

```

Output

Action Output

#	Time	Action	Message
4	14:24:50	DROP TABLE IF EXISTS Applications	0 row(s) affected

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.

Query:

```

SELECT j.JobID, j.JobTitle, COUNT(a.ApplicantID) AS Application_Count
FROM Jobs j

```


Join Applications a ON j.JobID = a.JobID

GROUP BY j.JobID, j.JobTitle;

The screenshot shows a SQL IDE interface. At the top, a toolbar includes icons for file operations, a 'Limit to 1000 rows' dropdown, and other utility icons. Below the toolbar, a SQL query is entered in a text area:

```
1 • SELECT j.JobID, j.JobTitle, COUNT(a.ApplicantID) AS Application_Count
2 FROM Jobs j
3 Join Applications a ON j.JobID = a.JobID
4 GROUP BY j.JobID, j.JobTitle;
5
```

Below the query editor, a 'Result Grid' tab is active, displaying the query results in a table:

JobID	JobTitle	Application_Count
101	Software Developer	1
102	UI/UX Designer	1
103	Data Analyst	1
104	Project Manager	1
105	DevOps Engineer	1
106	QA Tester	1
107	Backend Developer	1
108	Frontend Developer	1
109	AI Engineer	1
110	Content Manager	1

At the bottom, an 'Output' tab is visible, showing a log of actions:

#	Time	Action	Message
1	12:28:52	SELECT j.JobID, j.JobTitle, COUNT(a.ApplicantID) AS Application_Count FROM Jobs j Join Applications a ON j.JobID ...	10 row(s) returned

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.

Query:

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 80000;

Query 1 x

Limit to 1000 rows

```

1 • SELECT j.JobTitle, c.CompanyName, j.JobLocation, j.Salary
2 FROM Jobs j
3 JOIN Companies c
4 ON j.CompanyID = c.CompanyID
5 WHERE j.Salary BETWEEN 40000 AND 80000;

```

Result Grid

	JobTitle	CompanyName	JobLocation	Salary
▶	UI/UX Designer	GreenLeaf Solutions	Chennai	60000.00
	Data Analyst	DataForge Systems	Hyderabad	75000.00
	QA Tester	BrightWave Tech	Pune	65000.00
	Backend Developer	QuantumLeap Ltd	Chennai	80000.00
	Frontend Developer	CodeVerse Solutions	Noida	72000.00
	Content Manager	Zentrix Digital	Kolkata	55000.00

Result 7 x

Output

Action Output

#	Time	Action	Message
✓ 1	12:39:30	SELECT j.JobTitle, c.CompanyName, j.JobLocation, j.Salary FROM Jobs j JOIN Companies c ON j.CompanyID = c.Com...	6 row(s) returned

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.

Query:

```

SELECT j.JobTitle, j.CompanyID, c.CompanyName, a.ApplicationDate
FROM Jobs j
JOIN Companies c ON j.CompanyID = c.CompanyID
JOIN Applications a ON j.JobID = a.JobID
WHERE ApplicantID = '205';

```

Query 1 x

Limit to 1000 rows

```

1 • SELECT j.JobTitle, j.CompanyID, c.CompanyName, a.ApplicationDate
2 FROM Jobs j
3 JOIN Companies c ON j.CompanyID = c.CompanyID
4 JOIN Applications a ON j.JobID = a.JobID
5 WHERE ApplicantID = '205';

```

Result Grid

	JobTitle	CompanyID	CompanyName	ApplicationDate
▶	DevOps Engineer	5	CloudNest Corp.	2025-06-13 14:00:00
	Frontend Developer	8	CodeVerse Solutions	2025-06-16 11:15:00
	Content Manager	10	Zentrix Digital	2025-06-17 13:00:00

Result 14 x

Output

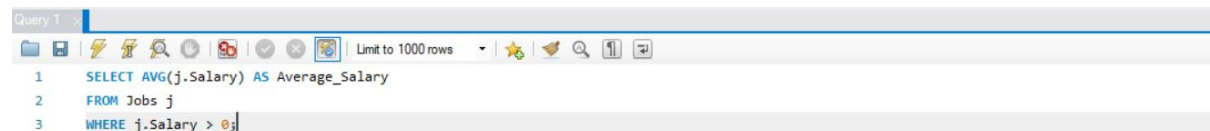
Action Output

#	Time	Action	Message
✓ 1	12:51:59	SELECT j.JobTitle, j.CompanyID, c.CompanyName, a.ApplicationDate FROM Jobs j JOIN Companies c ON j.CompanyID = c.Com...	3 row(s) returned
✓ 2	12:52:13	SELECT j.JobTitle, j.CompanyID, c.CompanyName, a.ApplicationDate FROM Jobs j JOIN Companies c ON j.CompanyID = c.Com...	3 row(s) returned

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.

Query:

```
SELECT AVG(j.Salary) AS Average_Salary  
FROM Jobs j  
WHERE j.Salary > 0;
```

A screenshot of the "Result Grid" showing the output of the query. It has a toolbar with "Filter Rows", "Export", and "Wrap Cell Content" options. The table has one column, "Average_Salary", and one row with the value "76466.666667".

Average_Salary
76466.666667

A screenshot of the "Action Output" window. It shows a table with columns "#", "Time", "Action", and "Message". The first row contains: "1", "12:58:17", "SELECT AVG(j.Salary) AS Average_Salary FROM Jobs j WHERE j.Salary > 0 LIMIT 0, 1000", and "1 row(s) returned".

#	Time	Action	Message
1	12:58:17	SELECT AVG(j.Salary) AS Average_Salary FROM Jobs j WHERE j.Salary > 0 LIMIT 0, 1000	1 row(s) returned

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.

Query:

```
SELECT c.CompanyName, COUNT(j.JobID) AS JobCount  
FROM Companies c  
JOIN Jobs j ON c.CompanyID = j.CompanyID  
GROUP BY c.CompanyID, c.CompanyName  
HAVING COUNT(j.JobID) = (  
    SELECT MAX(JobCount)  
FROM (  
    SELECT COUNT(JobID) AS JobCount  
FROM Jobs
```

GROUP BY CompanyID

) As Subquery

);

The screenshot shows a SQL IDE interface. The top pane displays a SQL query:

```
1 SELECT c.CompanyName, COUNT(j.JobID) AS JobCount
2 FROM Companies c
3 JOIN Jobs j ON c.CompanyID = j.CompanyID
4 GROUP BY c.CompanyID, c.CompanyName
5 HAVING COUNT(j.JobID) = (
6     SELECT MAX(JobCount)
7     FROM (
8         SELECT COUNT(JobID) AS JobCount
9         FROM Jobs
10        GROUP BY CompanyID
11    ) As Subquery
12 );
```

The bottom pane shows the 'Result Grid' with the following data:

CompanyName	JobCount
TechNova Inc.	2
GreenLeaf Solutions	2
InnovaSoft Pvt Ltd	2
QuantumLeap Ltd	2
NeuralLogic AI	2

Below the result grid, there is an 'Output' section showing the execution of the query:

```
Result 24 x
Output
Action Output
# Time Action Message
1 13:21:43 SELECT c.CompanyName, COUNT(j.JobID) AS JobCount FROM Companies c JOIN Jobs j ON c.CompanyID = j.Compa... 5 row(s) returned
```

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

Explanation :

Step 1: To find the year of experience of applicants we need to add another column named as Experience in Applicant Table and Update the values.

Query:

```
ALTER TABLE Applicants ADD COLUMN Experience INT;
```

```
UPDATE Applicants SET Experience = 4 WHERE ApplicantID IN(201,202,203);
```

```
UPDATE Applicants SET Experience = 2 WHERE ApplicantID IN (204, 205);
```

```
UPDATE Applicants SET Experience = 5 WHERE ApplicantID IN (206, 207);
```

```
UPDATE Applicants SET Experience = 1 WHERE ApplicantID IN (208);
```

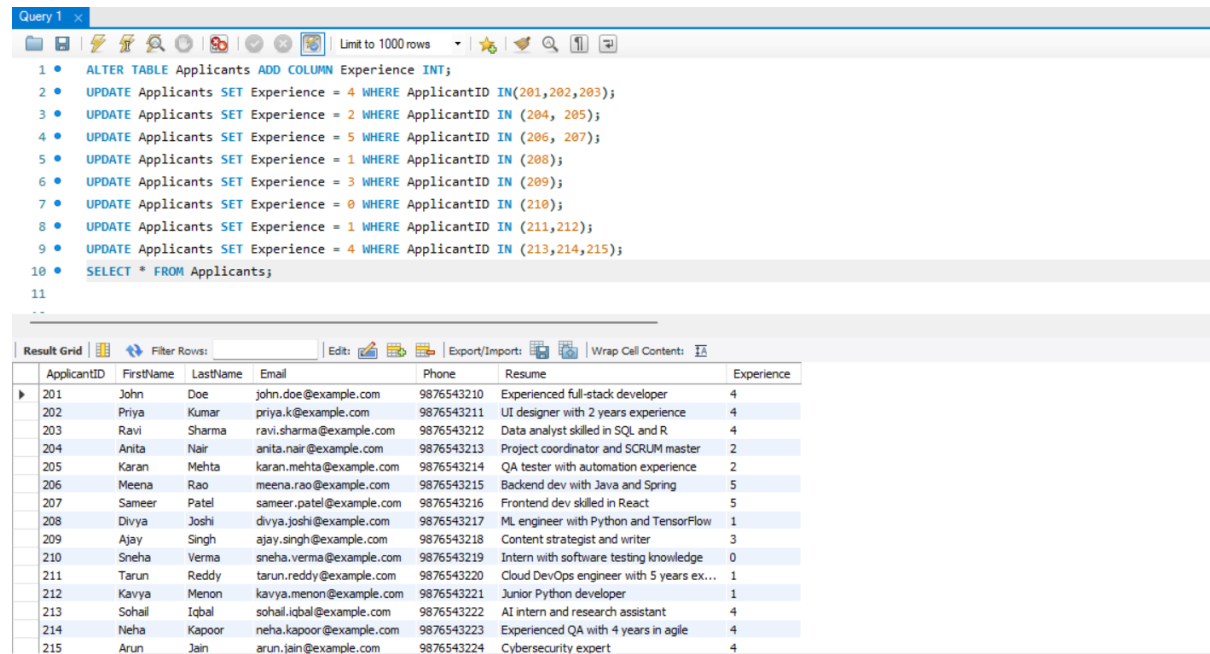
```
UPDATE Applicants SET Experience = 3 WHERE ApplicantID IN (209);
```

```
UPDATE Applicants SET Experience = 0 WHERE ApplicantID IN (210);
```

UPDATE Applicants SET Experience = 1 WHERE ApplicantID IN (211,212);

UPDATE Applicants SET Experience = 4 WHERE ApplicantID IN (213,214,215);

SELECT * FROM Applicants;



Query 1

```
1 • ALTER TABLE Applicants ADD COLUMN Experience INT;
2 • UPDATE Applicants SET Experience = 4 WHERE ApplicantID IN(201,202,203);
3 • UPDATE Applicants SET Experience = 2 WHERE ApplicantID IN (204, 205);
4 • UPDATE Applicants SET Experience = 5 WHERE ApplicantID IN (206, 207);
5 • UPDATE Applicants SET Experience = 1 WHERE ApplicantID IN (208);
6 • UPDATE Applicants SET Experience = 3 WHERE ApplicantID IN (209);
7 • UPDATE Applicants SET Experience = 0 WHERE ApplicantID IN (210);
8 • UPDATE Applicants SET Experience = 1 WHERE ApplicantID IN (211,212);
9 • UPDATE Applicants SET Experience = 4 WHERE ApplicantID IN (213,214,215);
10 • SELECT * FROM Applicants;
```

Result Grid

	ApplicantID	FirstName	LastName	Email	Phone	Resume	Experience
▶	201	John	Doe	john.doe@example.com	9876543210	Experienced full-stack developer	4
	202	Priya	Kumar	priya.k@example.com	9876543211	UI designer with 2 years experience	4
	203	Ravi	Sharma	ravi.sharma@example.com	9876543212	Data analyst skilled in SQL and R	4
	204	Anita	Nair	anita.nair@example.com	9876543213	Project coordinator and SCRUM master	2
	205	Karan	Mehta	karan.mehta@example.com	9876543214	QA tester with automation experience	2
	206	Meena	Rao	meena.rao@example.com	9876543215	Backend dev with Java and Spring	5
	207	Sameer	Patel	sameer.patel@example.com	9876543216	Frontend dev skilled in React	5
	208	Divya	Joshi	divya.joshi@example.com	9876543217	ML engineer with Python and TensorFlow	1
	209	Ajay	Singh	ajay.singh@example.com	9876543218	Content strategist and writer	3
	210	Sneha	Verma	sneha.verma@example.com	9876543219	Intern with software testing knowledge	0
	211	Tarun	Reddy	tarun.reddy@example.com	9876543220	Cloud DevOps engineer with 5 years ex...	1
	212	Kavya	Menon	kavya.menon@example.com	9876543221	Junior Python developer	1
	213	Sohail	Iqbal	sohail.iqbal@example.com	9876543222	AI intern and research assistant	4
	214	Neha	Kapoor	neha.kapoor@example.com	9876543223	Experienced QA with 4 years in agile	4
	215	Arun	Jain	arun.jain@example.com	9876543224	Cybersecurity expert	4

Step 2: Now we can analyse and find the applicant who have applied for positions in companies, locations and have at least 3 years of experiences.

SELECT a.ApplicantID, CONCAT(a.FirstName," ", a.LastName) AS Name, a.Experience, c.CompanyName, j.JobTitle,j.JobLocation

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 j.Joblocation ='Chennai'

AND a.Experience >=3;

Query 1

```

1 • SELECT a.ApplicantID, CONCAT(a.FirstName, " ", a.LastName) AS Name, a.Experience, c.CompanyName, j.JobTitle, j.JobLocation
2 FROM Applicants a
3 JOIN Applications ap ON a.ApplicantID = ap.ApplicantID
4 JOIN Jobs j ON ap.JobID = j.JobID
5 JOIN Companies c ON j.CompanyID = c.CompanyID
6 WHERE j.JobLocation = 'Chennai'
7 AND a.Experience >=3;
8
9
10
11

```

Result Grid

ApplicantID	Name	Experience	CompanyName	JobTitle	JobLocation
202	Priya Kumar	4	GreenLeaf Solutions	UI/UX Designer	Chennai
202	Priya Kumar	4	GreenLeaf Solutions	UI/UX Designer	Chennai
207	Sameer Patel	5	QuantumLeap Ltd	Backend Developer	Chennai
203	Ravi Sharma	4	QuantumLeap Ltd	Backend Developer	Chennai

Result 30

Output

Action Output

#	Time	Action	Message
✓ 1	13:32:15	SELECT * FROM Applicants LIMIT 0, 1000	15 row(s) returned
✓ 2	13:40:13	SELECT c.CompanyName, COUNT(j.JobID) AS JobCount FROM Companies c JOIN Jobs j ON c.CompanyID = j.Compa...	5 row(s) returned
✓ 3	13:44:49	SELECT a.ApplicantID, CONCAT(a.FirstName, " ", a.LastName) AS Name, a.Experience, c.CompanyName, j.JobTitle j.J...	4 row(s) returned

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

Query:

SELECT DISTINCT JobTitle

FROM Jobs

WHERE Salary BETWEEN 60000 AND 80000;

Query 1

```

1 • SELECT DISTINCT JobTitle
2 FROM Jobs
3 WHERE Salary BETWEEN 60000 AND 80000;
4
5
6
7
8

```

Result Grid

JobTitle
UI/UX Designer
Data Analyst
QA Tester
Backend Developer
Frontend Developer
Security Analyst

Jobs 31

Output

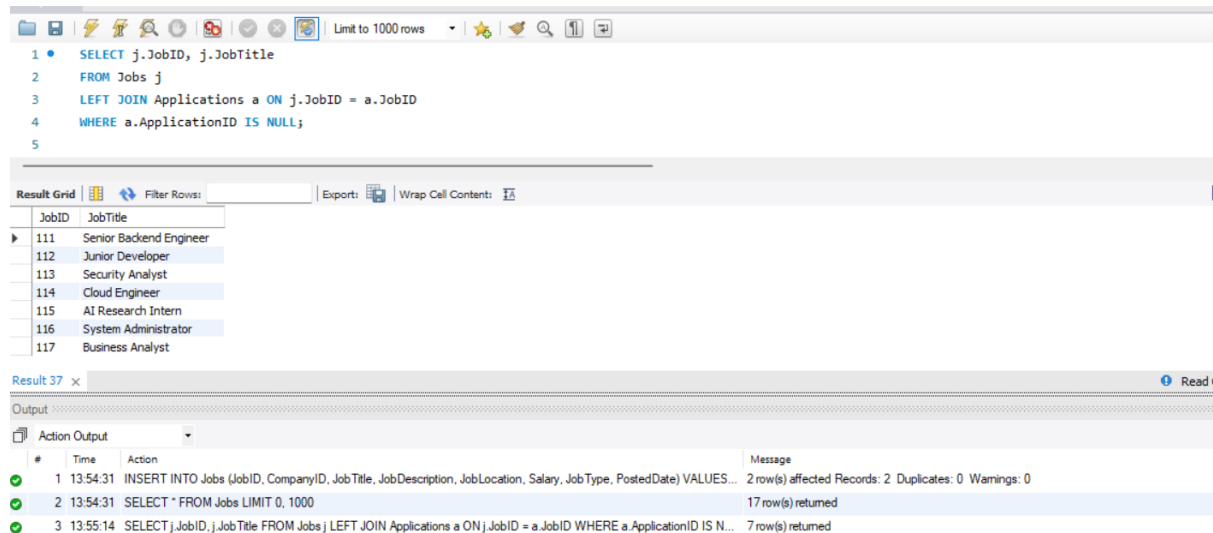
Action Output

#	Time	Action	Message
✓ 1	13:32:15	SELECT * FROM Applicants LIMIT 0, 1000	15 row(s) returned
✓ 2	13:40:13	SELECT c.CompanyName, COUNT(j.JobID) AS JobCount FROM Companies c JOIN Jobs j ON c.CompanyID = j.Compa...	5 row(s) returned
✓ 3	13:44:49	SELECT a.ApplicantID, CONCAT(a.FirstName, " ", a.LastName) AS Name, a.Experience, c.CompanyName, j.JobTitle j.J...	4 row(s) returned
✓ 4	13:47:04	SELECT DISTINCT JobTitle FROM Jobs WHERE Salary BETWEEN 60000 AND 80000 LIMIT 0, 1000	6 row(s) returned

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

Query:

```
SELECT j.JobID, j.JobTitle
FROM Jobs j
LEFT JOIN Applications a ON j.JobID = a.JobID
WHERE a.ApplicationID IS NULL;
```



The screenshot shows a database query tool interface. The top section displays the SQL query: `SELECT j.JobID, j.JobTitle FROM Jobs j LEFT JOIN Applications a ON j.JobID = a.JobID WHERE a.ApplicationID IS NULL;`. Below the query, the 'Result Grid' shows a table with two columns: 'JobID' and 'JobTitle'. The results are as follows:

JobID	JobTitle
111	Senior Backend Engineer
112	Junior Developer
113	Security Analyst
114	Cloud Engineer
115	AI Research Intern
116	System Administrator
117	Business Analyst

Below the result grid, the 'Action Output' section shows the execution log:

#	Time	Action	Message
1	13:54:31	INSERT INTO Jobs (JobID, CompanyID, JobTitle, JobDescription, JobLocation, Salary, JobType, PostedDate) VALUES...	2 row(s) affected Records: 2 Duplicates: 0 Warnings: 0
2	13:54:31	SELECT * FROM Jobs LIMIT 0, 1000	17 row(s) returned
3	13:55:14	SELECT j.JobID, j.JobTitle FROM Jobs j LEFT JOIN Applications a ON j.JobID = a.JobID WHERE a.ApplicationID IS N...	7 row(s) returned

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

Query:

```
SELECT a.ApplicantID, CONCAT( a.FirstName," ", a.LastName) AS Applicant_Name,
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;
```

Limit to 1000 rows

```

1 • SELECT a.ApplicantID, CONCAT( a.FirstName, " ", a.LastName) AS Applicant_Name, c.CompanyName, j.JobTitle
2 FROM Applicants a
3 JOIN Applications ap ON a.ApplicantID = ap.ApplicantID
4 JOIN Jobs j ON ap.JobID = j.JobID
5 JOIN Companies c ON j.CompanyID = c.CompanyID;
6

```

ApplicantID	Applicant_Name	CompanyName	JobTitle
201	John Doe	TechNova Inc.	Software Developer
202	Priya Kumar	GreenLeaf Solutions	UI/UX Designer
203	Ravi Sharma	DataForge Systems	Data Analyst
204	Anita Nair	InnovaSoft Pvt Ltd	Project Manager
205	Karan Mehta	CloudNest Corp.	DevOps Engineer
206	Meena Rao	BrightWave Tech	QA Tester
207	Sameer Patel	QuantumLeap Ltd	Backend Developer
208	Divya Joshi	CodeVerse Solutions	Frontend Developer
209	Ajay Singh	NeuralLogic AI	AI Engineer
210	Sneha Verma	Zentrix Digital	Content Manager
202	Priya Kumar	GreenLeaf Solutions	UI/UX Designer
202	Priya Kumar	InnovaSoft Pvt Ltd	Project Manager
202	Priya Kumar	BrightWave Tech	QA Tester
203	Ravi Sharma	DataForge Systems	Data Analyst
203	Ravi Sharma	QuantumLeap Ltd	Backend Developer
204	Anita Nair	TechNova Inc.	Software Developer
204	Anita Nair	NeuralLogic AI	AI Engineer
205	Karan Mehta	CodeVerse Solutions	Frontend Developer
205	Karan Mehta	Zentrix Digital	Content Manager
210	Sneha Verma	BrightWave Tech	QA Tester
210	Sneha Verma	InnovaSoft Pvt Ltd	Project Manager

Result 39

Output

Action Output

#	Time	Action	Message
1	13:59:12	SELECT a.ApplicantID, CONCAT(a.FirstName, " ", a.LastName) AS Applicant_Name, c.CompanyName, j.JobTitle FRO...	21 row(s) returned

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

Query:

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

Limit to 1000 rows

```

1 • SELECT c.CompanyName, COUNT(j.JobID) AS JobCount
2 FROM Companies c
3 LEFT JOIN Jobs j ON c.CompanyID = j.CompanyID
4 GROUP BY c.CompanyID, c.CompanyName;
5

```

CompanyName	JobCount
TechNova Inc.	3
GreenLeaf Solutions	2
DataForge Systems	1
InnovaSoft Pvt Ltd	3
CloudNest Corp.	1
BrightWave Tech	1
QuantumLeap Ltd	2
CodeVerse Solutions	1
NeuralLogic AI	2
Zentrix Digital	1

Result 41

Output

Action Output

#	Time	Action	Message
1	14:01:04	SELECT c.CompanyName, COUNT(j.JobID) AS JobCount FROM Companies c LEFT JOIN Jobs j ON c.CompanyID = j...	10 row(s) returned

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

Query:

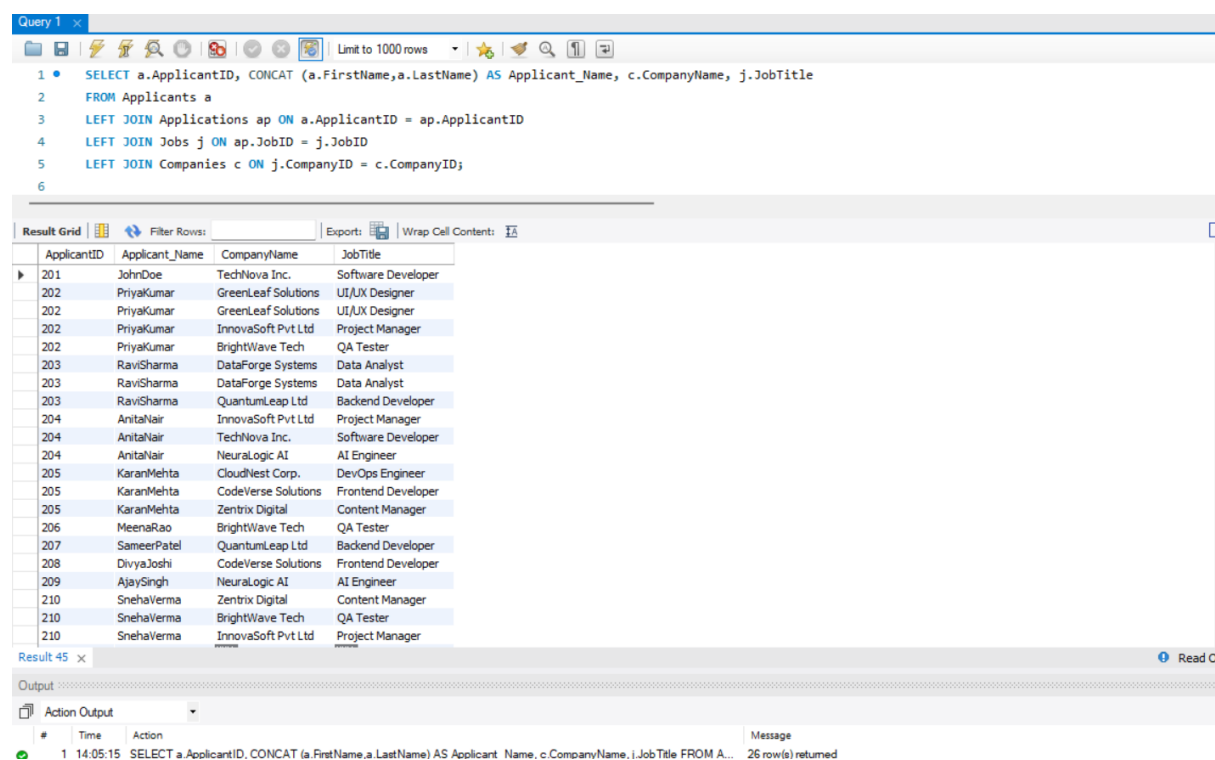
```
SELECT a.ApplicantID, CONCAT (a.FirstName,a.LastName) AS Applicant_Name,
c.CompanyName, j.JobTitle

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;
```



The screenshot shows a database query tool interface. At the top, there's a toolbar with various icons and a 'Limit to 1000 rows' dropdown. Below the toolbar, the SQL query is displayed in a text area. The query is as follows:

```
1 SELECT a.ApplicantID, CONCAT (a.FirstName,a.LastName) AS Applicant_Name, c.CompanyName, j.JobTitle
2 FROM Applicants a
3 LEFT JOIN Applications ap ON a.ApplicantID = ap.ApplicantID
4 LEFT JOIN Jobs j ON ap.JobID = j.JobID
5 LEFT JOIN Companies c ON j.CompanyID = c.CompanyID;
6
```

Below the query, there's a 'Result Grid' section. It shows a table with 4 columns: ApplicantID, Applicant_Name, CompanyName, and JobTitle. The table contains 26 rows of data. The first few rows are:

ApplicantID	Applicant_Name	CompanyName	JobTitle
201	JohnDoe	TechNova Inc.	Software Developer
202	PriyaKumar	GreenLeaf Solutions	UI/UX Designer
202	PriyaKumar	GreenLeaf Solutions	UI/UX Designer
202	PriyaKumar	InnovaSoft Pvt Ltd	Project Manager
202	PriyaKumar	BrightWave Tech	QA Tester
203	RaviSharma	DataForge Systems	Data Analyst
203	RaviSharma	DataForge Systems	Data Analyst
203	RaviSharma	QuantumLeap Ltd	Backend Developer
204	AnitaNair	InnovaSoft Pvt Ltd	Project Manager
204	AnitaNair	TechNova Inc.	Software Developer
204	AnitaNair	NeuraLogic AI	AI Engineer
205	KaranMehta	CloudNest Corp.	DevOps Engineer
205	KaranMehta	CodeVerse Solutions	Frontend Developer
205	KaranMehta	Zentrix Digital	Content Manager
206	MeenaRao	BrightWave Tech	QA Tester
207	SameerPatel	QuantumLeap Ltd	Backend Developer
208	DivyaJoshi	CodeVerse Solutions	Frontend Developer
209	AjaySingh	NeuraLogic AI	AI Engineer
210	SnehaVerma	Zentrix Digital	Content Manager
210	SnehaVerma	BrightWave Tech	QA Tester
210	SnehaVerma	InnovaSoft Pvt Ltd	Project Manager

At the bottom, there's an 'Output' section with a table showing the execution details:

#	Time	Action	Message
1	14:05:15	SELECT a.ApplicantID, CONCAT (a.FirstName,a.LastName) AS Applicant_Name, c.CompanyName, j.JobTitle FROM A...	26 row(s) returned

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

Query:

```
SELECT DISTINCT c.CompanyName

FROM Companies c

JOIN Jobs j ON c.CompanyID = j.CompanyID

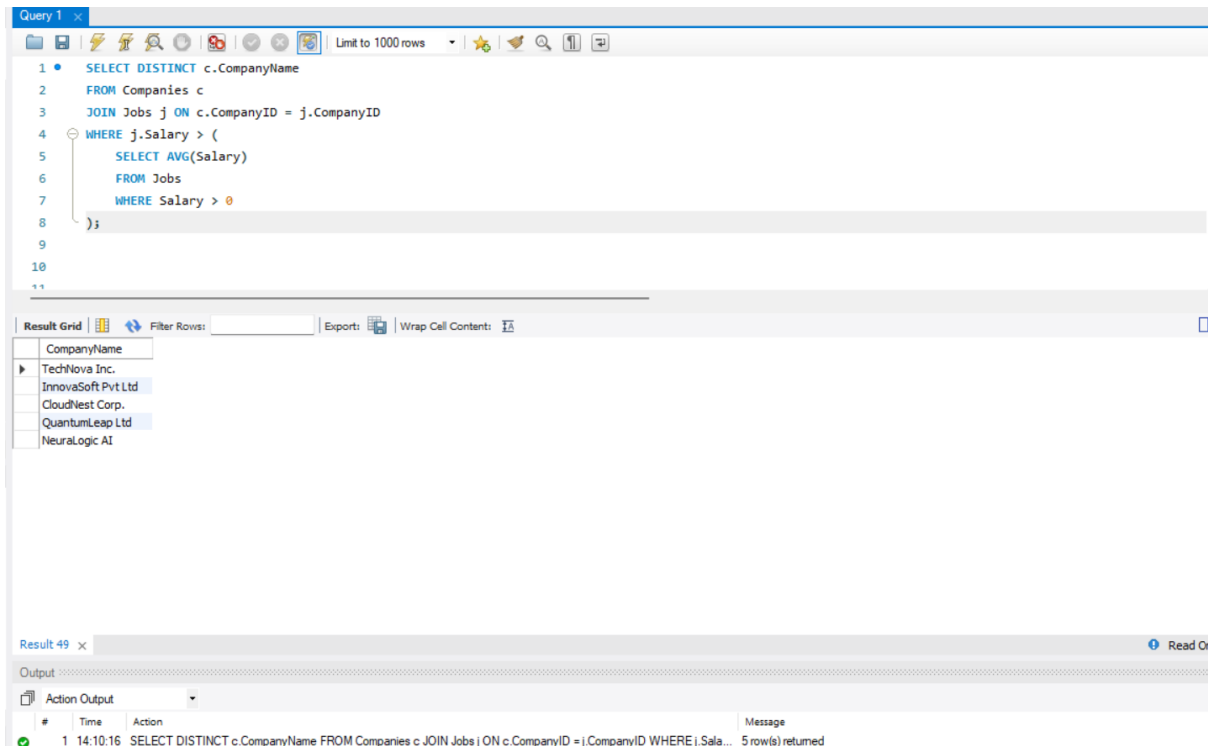
WHERE j.Salary > (

    SELECT AVG(Salary)
```

FROM Jobs

WHERE Salary > 0

);



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

Explanation: Adding extra columns to table Applications as City and State . Also display the output with Concat.

Query:

ALTER TABLE Applicants ADD City VARCHAR(50), ADD State VARCHAR(50);

UPDATE Applicants SET City='Bangalore', State='KA' WHERE ApplicantID = 201;

UPDATE Applicants SET City='Chennai', State='TN' WHERE ApplicantID = 202;

UPDATE Applicants SET City='Hyderabad', State='TS' WHERE ApplicantID = 203;

UPDATE Applicants SET City='Mumbai', State='MH' WHERE ApplicantID = 204;

UPDATE Applicants SET City='Delhi', State='DL' WHERE ApplicantID = 205;

UPDATE Applicants SET City='Pune', State='MH' WHERE ApplicantID = 206;

UPDATE Applicants SET City='Chennai', State='TN' WHERE ApplicantID = 207;

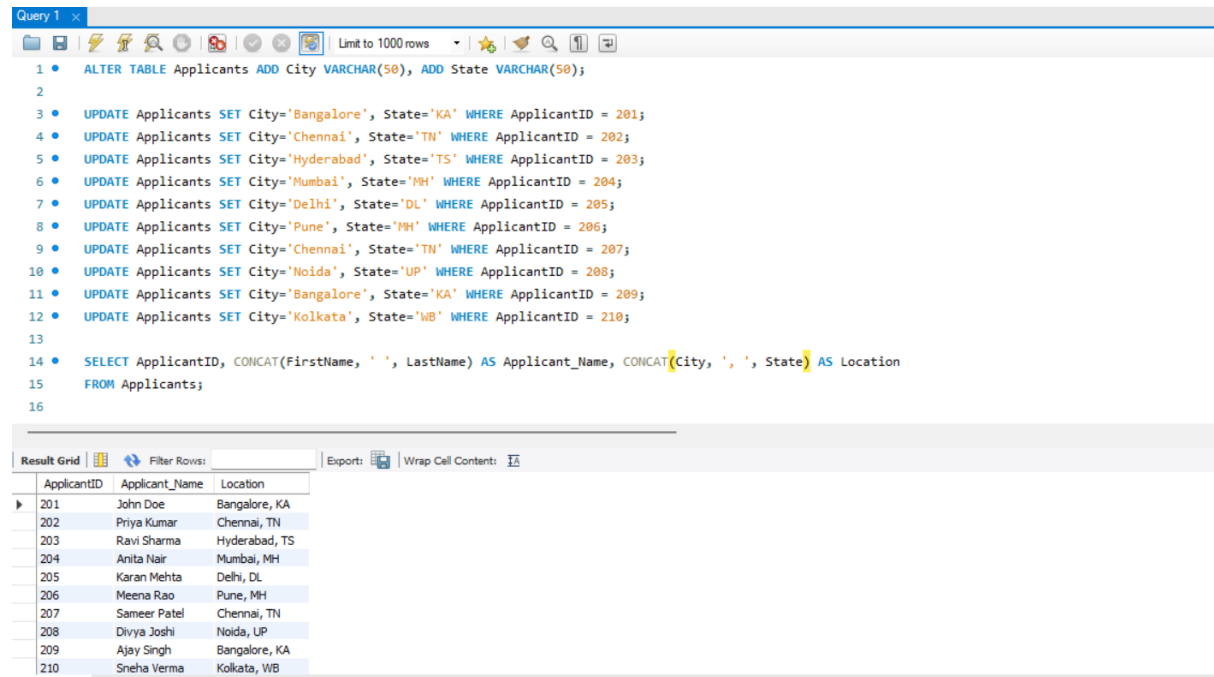
UPDATE Applicants SET City='Noida', State='UP' WHERE ApplicantID = 208;

UPDATE Applicants SET City='Bangalore', State='KA' WHERE ApplicantID = 209;

UPDATE Applicants SET City='Kolkata', State='WB' WHERE ApplicantID = 210;

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

FROM Applicants;



The screenshot shows a SQL query editor with a script that updates the City and State for 10 applicants (IDs 201-210) and then selects their IDs, names, and locations. The results grid below shows the output of the SELECT statement.

ApplicantID	Applicant_Name	Location
201	John Doe	Bangalore, KA
202	Priya Kumar	Chennai, TN
203	Ravi Sharma	Hyderabad, TS
204	Anita Nair	Mumbai, MH
205	Karan Mehta	Delhi, DL
206	Meena Rao	Pune, MH
207	Sameer Patel	Chennai, TN
208	Divya Joshi	Noida, UP
209	Ajay Singh	Bangalore, KA
210	Sneha Verma	Kolkata, WB

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

Query:

SELECT JobID, JobTitle, Salary

FROM Jobs

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

Query 1

```

1 SELECT JobID, JobTitle, Salary
2 FROM Jobs
3 WHERE JobTitle LIKE '%Developer%' OR JobTitle LIKE '%Engineer%';
4

```

Result Grid

JobID	JobTitle	Salary
101	Software Developer	85000.00
105	DevOps Engineer	90000.00
107	Backend Developer	80000.00
108	Frontend Developer	72000.00
109	AI Engineer	100000.00
111	Senior Backend Engineer	120000.00
112	Junior Developer	50000.00
114	Cloud Engineer	105000.00
NULL	NULL	NULL

Jobs 52

Output

Action Output

#	Time	Action	Message
1	14:18:44	SELECT JobID, JobTitle, Salary FROM Jobs WHERE Job Title LIKE "%Developer%" OR Job Title LIKE "%Engineer%" LIM...	8 row(s) returned

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

Query:

```

SELECT a.ApplicantID, CONCAT(a.FirstName, ' ', a.LastName) AS ApplicantName,
j.JobID, j.JobTitle
FROM Applicants a
LEFT JOIN Applications ap ON a.ApplicantID = ap.ApplicantID
LEFT JOIN Jobs j ON ap.JobID = j.JobID;

```

Query 1

```

1 SELECT a.ApplicantID, CONCAT(a.FirstName, ' ', a.LastName) AS ApplicantName, j.JobID, j.JobTitle
2 FROM Applicants a
3 LEFT JOIN Applications ap ON a.ApplicantID = ap.ApplicantID
4 LEFT JOIN Jobs j ON ap.JobID = j.JobID;
5

```

Result Grid

ApplicantID	ApplicantName	JobID	JobTitle
201	John Doe	101	Software Developer
202	Priya Kumar	102	UI/UX Designer
202	Priya Kumar	102	UI/UX Designer
202	Priya Kumar	104	Project Manager
202	Priya Kumar	106	QA Tester
203	Ravi Sharma	103	Data Analyst
203	Ravi Sharma	103	Data Analyst
203	Ravi Sharma	107	Backend Developer
204	Anita Nair	104	Project Manager
204	Anita Nair	101	Software Developer
204	Anita Nair	109	AI Engineer
205	Karan Mehta	105	DevOps Engineer
205	Karan Mehta	108	Frontend Developer
205	Karan Mehta	110	Content Manager
206	Meena Rao	106	QA Tester
207	Sameer Patel	107	Backend Developer
208	Divya Joshi	108	Frontend Developer
209	Ajay Singh	109	AI Engineer
210	Sneha Verma	110	Content Manager
210	Sneha Verma	106	QA Tester
210	Sneha Verma	104	Project Manager

Result 53

Output

Action Output

#	Time	Action	Message
1	14:20:15	SELECT a.ApplicantID, CONCAT(a.FirstName, ' ', a.LastName) AS ApplicantName, j.JobID, j.JobTitle FROM Applicants ...	26 row(s) returned

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

Query:

SELECT a.ApplicantID, CONCAT(a.FirstName, ' ', a.LastName) AS ApplicantName,
c.CompanyName, c.Location

FROM Applicants a

CROSS JOIN Companies c

WHERE c.Location = 'Chennai' AND a.Experience > 2;

Query 1

```
1 SELECT a.ApplicantID, CONCAT(a.FirstName, ' ', a.LastName) AS ApplicantName, c.CompanyName, c.Location
2 FROM Applicants a
3 CROSS JOIN Companies c
4 WHERE c.Location = 'Chennai' AND a.Experience > 2;
5
```

Result Grid

ApplicantID	ApplicantName	CompanyName	Location
201	John Doe	QuantumLeap Ltd	Chennai
201	John Doe	GreenLeaf Solutions	Chennai
202	Priya Kumar	QuantumLeap Ltd	Chennai
202	Priya Kumar	GreenLeaf Solutions	Chennai
203	Ravi Sharma	QuantumLeap Ltd	Chennai
203	Ravi Sharma	GreenLeaf Solutions	Chennai
206	Meena Rao	QuantumLeap Ltd	Chennai
206	Meena Rao	GreenLeaf Solutions	Chennai
207	Sameer Patel	QuantumLeap Ltd	Chennai
207	Sameer Patel	GreenLeaf Solutions	Chennai
209	Ajay Singh	QuantumLeap Ltd	Chennai
209	Ajay Singh	GreenLeaf Solutions	Chennai
213	Sohail Iqbal	QuantumLeap Ltd	Chennai
213	Sohail Iqbal	GreenLeaf Solutions	Chennai
214	Neha Kapoor	QuantumLeap Ltd	Chennai
214	Neha Kapoor	GreenLeaf Solutions	Chennai
215	Arun Jain	QuantumLeap Ltd	Chennai
215	Arun Jain	GreenLeaf Solutions	Chennai

Result 55

Output

Action Output

#	Time	Action	Message
1	14:23:05	SELECT a.ApplicantID, CONCAT(a.FirstName, ' ', a.LastName) AS ApplicantName, c.CompanyName, c.Location FROM...	18 row(s) returned