**Department of Data Science and Business Systems**

**School of Computing**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**DBMS MINI PROJECT**

# JOB MANAGEMENT SYSTEM

**SUBMITTED TO**

***Dr .D.Hemavathi***

**Team Members**

VIJAY KUMAR S (RA2111027010001)

C.DINESH (RA2111027010002)

K.AKASH (RA2111027010015)

**APRIL 2024**

**SRM INSTITUTE OF SCIENCE &TECHNOLOGY**

**COLLEGE OF ENGINEERING &TECHNOLOGY**

**S.R.M. NAGAR, KATTANKULATHUR – 603203**

## BONAFIDE CERTIFICATE

Certified that this project report **Job Management system** is the bonafide work of VIJAY KUMAR S(RA2111027010001,C.DINESH (RA2111027010002**,** K.AKASH (RA2111027010015)of III Year/VI Sem B.Tech (BDA) who carried out the mini project work under my supervision for the course 18CSC303J- Database Management systems in Data Science and Business systems department, school of Computing, SRM Institute of Science and Technology during the academic year 2023-2024(Even sem).

| | |
|---|---|
| Signature of Head of the Department | Signature of Faculty In charge |
| Dr. Lakshmi M | Dr D Hemavathi |
| Head of the Department | Associate Professor |
| Data Science and Business Systems | Data Science and Business Systems |
| School of Computing | School of Computing |

# Abstract

The Job Management System (JMS) is a comprehensive database management project designed to streamline and automate the process of job allocation, monitoring, and reporting within an organization. Leveraging the power of database management systems (DBMS), this system offers a user-friendly interface for administrators to efficiently manage job assignments and track their progress.

The system allows administrators to create, modify, and delete job entries, each containing relevant details such as job title, description, priority level, deadline, and assigned personnel. Through a series of structured queries, users can perform various tasks including job assignment, status updates, and generating reports based on specific criteria.

Key features of the Job Management System include:

Job Creation: Administrators can create new job entries, specifying essential information such as job title, description, and deadline.
Job Assignment: Users can assign jobs to specific personnel or teams, ensuring clear accountability and efficient task distribution.
Status Tracking: The system enables real-time monitoring of job statuses, providing visibility into pending, ongoing, and completed tasks.
Priority Management: Jobs can be categorized based on priority levels, allowing administrators to prioritize tasks and allocate resources accordingly.
Reporting: Comprehensive reporting functionalities allow users to generate customized reports based on parameters such as job status, personnel performance, and overall workload.
By centralizing job management processes and leveraging the capabilities of a robust DBMS, the Job Management System offers organizations a scalable and efficient solution for optimizing workforce productivity and enhancing project management capabilities.

# Introduction

In today's dynamic business landscape, efficient management of human resources and job assignments is paramount for organizational success. The advent of database management systems (DBMS) has revolutionized how businesses handle their data and streamline operations. A Job Management System (JMS) built on a robust DBMS platform offers a comprehensive solution to address the complexities of job allocation, task scheduling, and resource optimization.

Our project aims to design and implement a Job Management System utilizing the power of queries within a DBMS framework. By leveraging the structured querying capabilities of the database, we intend to create a system that enables seamless management of tasks, assignments, and personnel, ultimately enhancing productivity and minimizing operational bottlenecks.

Through this project, we endeavor to address the following key objectives:

1. Efficient Task Allocation: Utilizing SQL queries to allocate tasks to appropriate personnel based on skillsets, availability, and priority, ensuring optimal resource utilization.

2. Real-time Monitoring and Tracking: Implementing query-based mechanisms to enable real-time monitoring of task progress, resource allocation, and project milestones, providing stakeholders with timely insights into project status.

3. Flexible Reporting and Analytics: Leveraging SQL queries to generate customizable reports and perform in-depth analytics on job performance, resource utilization, and operational efficiency, facilitating data-driven decision-making.

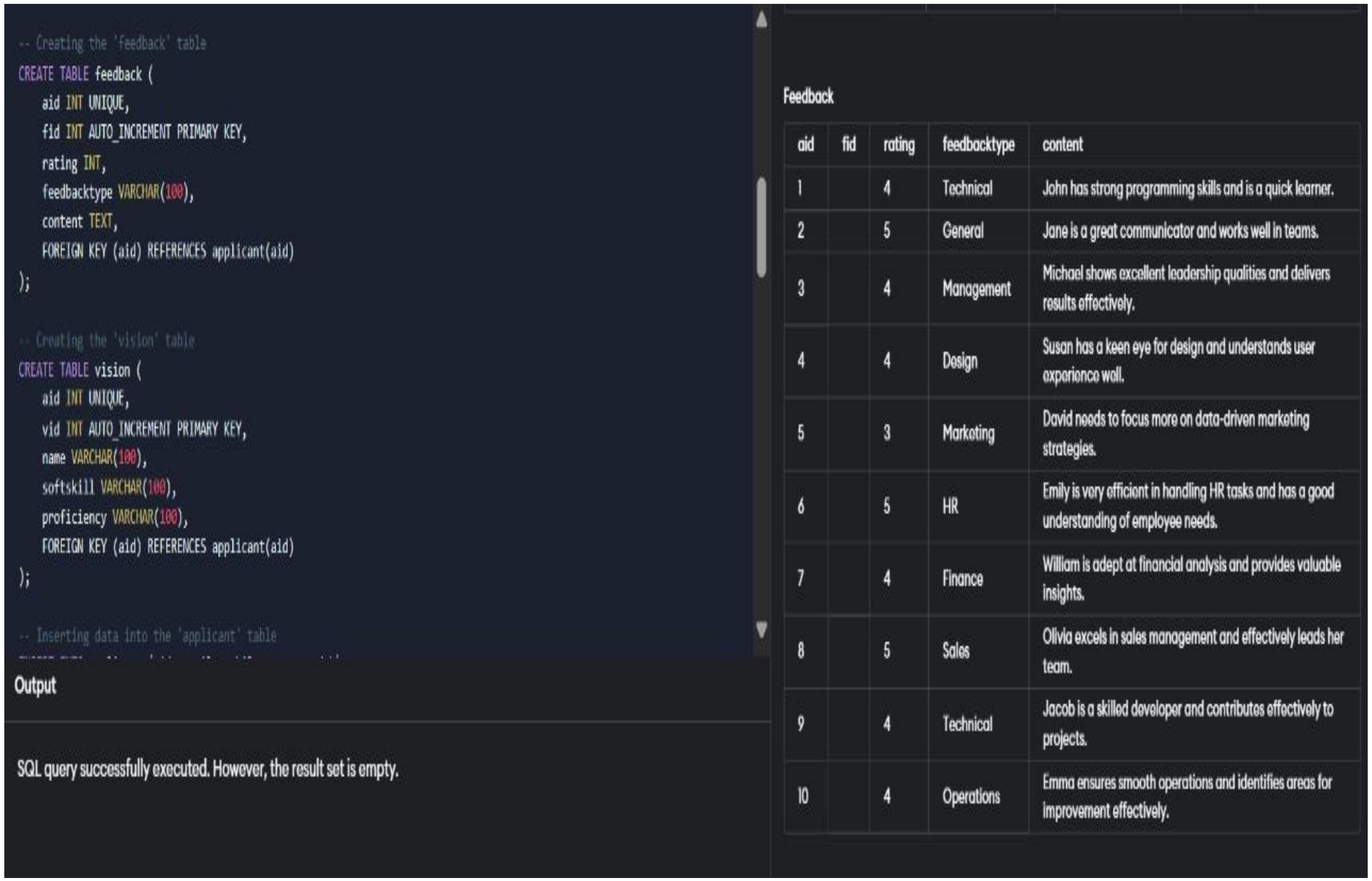
4. Scalability and Maintainability: Designing a scalable and maintainable database schema that can accommodate future growth and evolving business requirements, ensuring the longevity and adaptability of the Job Management System.

5. User-friendly Interface: Integrating intuitive user interfaces with query-driven functionalities to facilitate ease of use for administrators, managers, and employees, enhancing overall user experience and adoption.

By harnessing the capabilities of DBMS and leveraging queries as the backbone of our Job Management System, we aim to deliver a robust, efficient, and scalable solution that empowers organizations to effectively manage their workforce, streamline operations, and drive business success.

## Proposed work details

## TABLES:



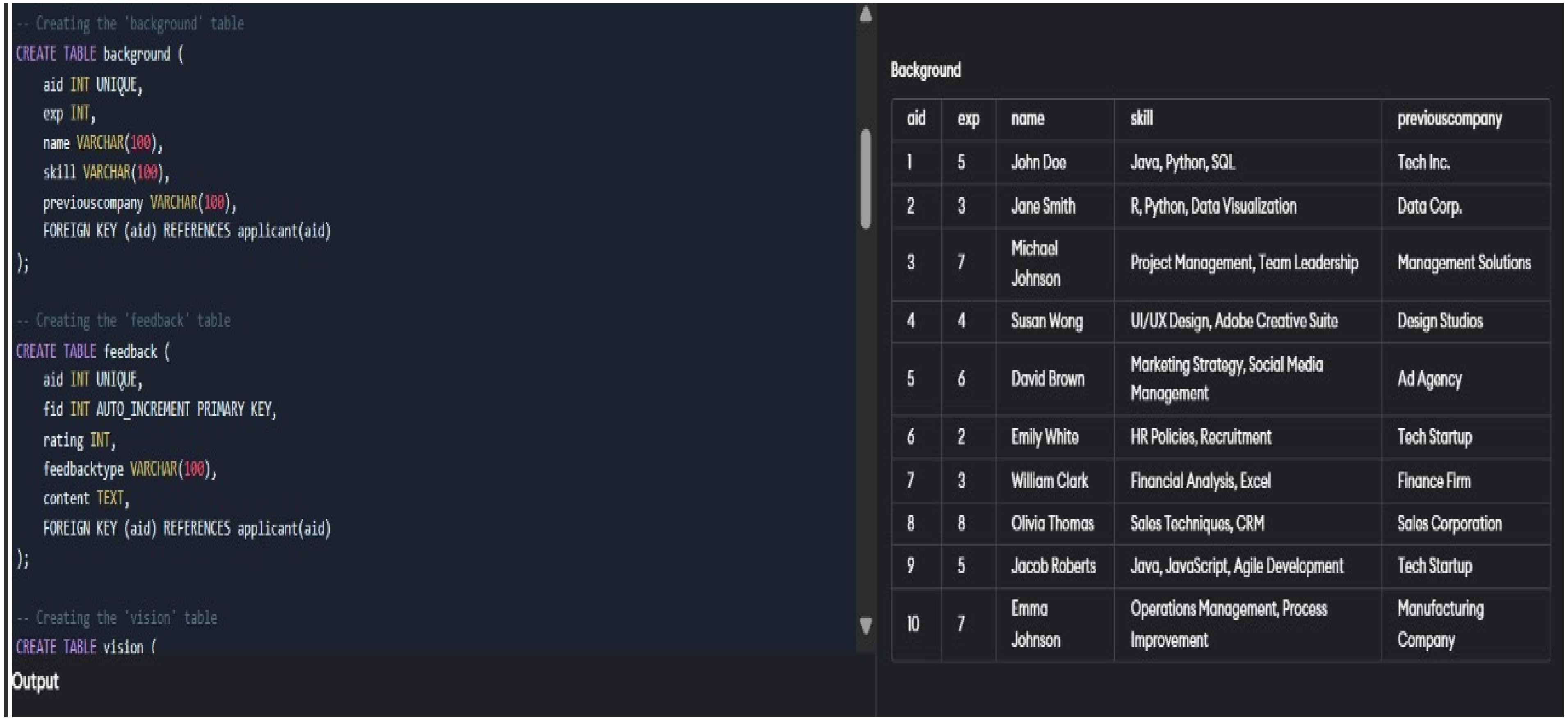
```
-- Creating the 'feedback' table
CREATE TABLE feedback (
    aid INT UNIQUE,
    fid INT AUTO_INCREMENT PRIMARY KEY,
    rating INT,
    feedbacktype VARCHAR(100),
    content TEXT,
    FOREIGN KEY (aid) REFERENCES applicant(aid)
);

-- Creating the 'vision' table
CREATE TABLE vision (
    aid INT UNIQUE,
    vid INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100),
    softskill VARCHAR(100),
    proficiency VARCHAR(100),
    FOREIGN KEY (aid) REFERENCES applicant(aid)
);

-- Inserting data into the 'applicant' table
```

Output

```
SQL query successfully executed. However, the result set is empty.
```

Feedback

| aid | fid | rating | feedbacktype | content |
|---|---|---|---|---|
| 1 | | 4 | Technical | John has strong programming skills and is a quick learner. |
| 2 | | 5 | General | Jane is a great communicator and works well in teams. |
| 3 | | 4 | Management | Michael shows excellent leadership qualities and delivers results effectively. |
| 4 | | 4 | Design | Susan has a keen eye for design and understands user experience well. |
| 5 | | 3 | Marketing | David needs to focus more on data-driven marketing strategies. |
| 6 | | 5 | HR | Emily is very efficient in handling HR tasks and has a good understanding of employee needs. |
| 7 | | 4 | Finance | William is adept at financial analysis and provides valuable insights. |
| 8 | | 5 | Sales | Olivia excels in sales management and effectively leads her team. |
| 9 | | 4 | Technical | Jacob is a skilled developer and contributes effectively to projects. |
| 10 | | 4 | Operations | Emma ensures smooth operations and identifies areas for improvement effectively. |

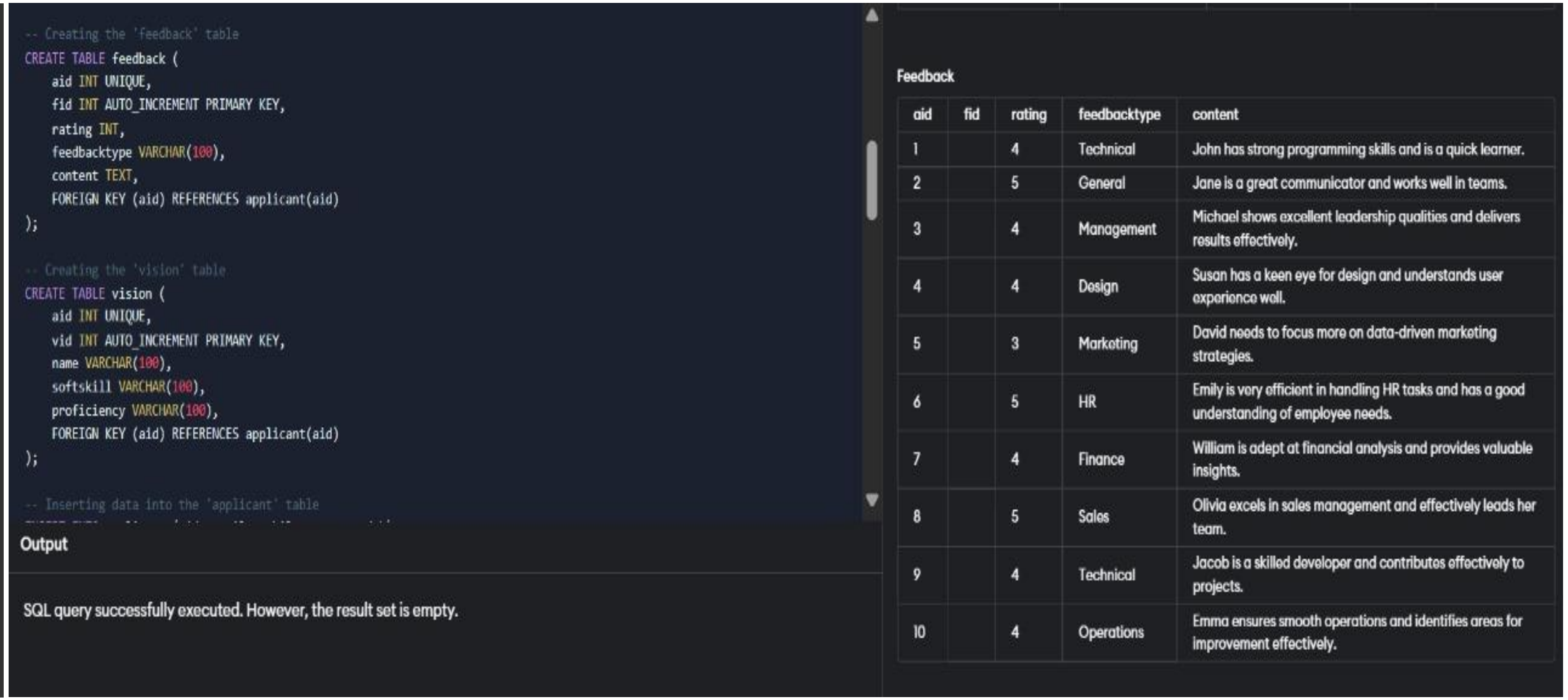


```
-- Creating the 'background' table
CREATE TABLE background (
    aid INT UNIQUE,
    exp INT,
    name VARCHAR(100),
    skill VARCHAR(100),
    previouscompany VARCHAR(100),
    FOREIGN KEY (aid) REFERENCES applicant(aid)
);

-- Creating the 'feedback' table
CREATE TABLE feedback (
    aid INT UNIQUE,
    fid INT AUTO_INCREMENT PRIMARY KEY,
    rating INT,
    feedbacktype VARCHAR(100),
    content TEXT,
    FOREIGN KEY (aid) REFERENCES applicant(aid)
);

-- Creating the 'vision' table
CREATE TABLE vision (
```

Output

Background

| aid | exp | name | skill | previouscompany |
|---|---|---|---|---|
| 1 | 5 | John Doe | Java, Python, SQL | Tech Inc. |
| 2 | 3 | Jane Smith | R, Python, Data Visualization | Data Corp. |
| 3 | 7 | Michael Johnson | Project Management, Team Leadership | Management Solutions |
| 4 | 4 | Susan Wong | UI/UX Design, Adobe Creative Suite | Design Studios |
| 5 | 6 | David Brown | Marketing Strategy, Social Media Management | Ad Agency |
| 6 | 2 | Emily White | HR Policies, Recruitment | Tech Startup |
| 7 | 3 | William Clark | Financial Analysis, Excel | Finance Firm |
| 8 | 8 | Olivia Thomas | Sales Techniques, CRM | Sales Corporation |
| 9 | 5 | Jacob Roberts | Java, JavaScript, Agile Development | Tech Startup |
| 10 | 7 | Emma Johnson | Operations Management, Process Improvement | Manufacturing Company |



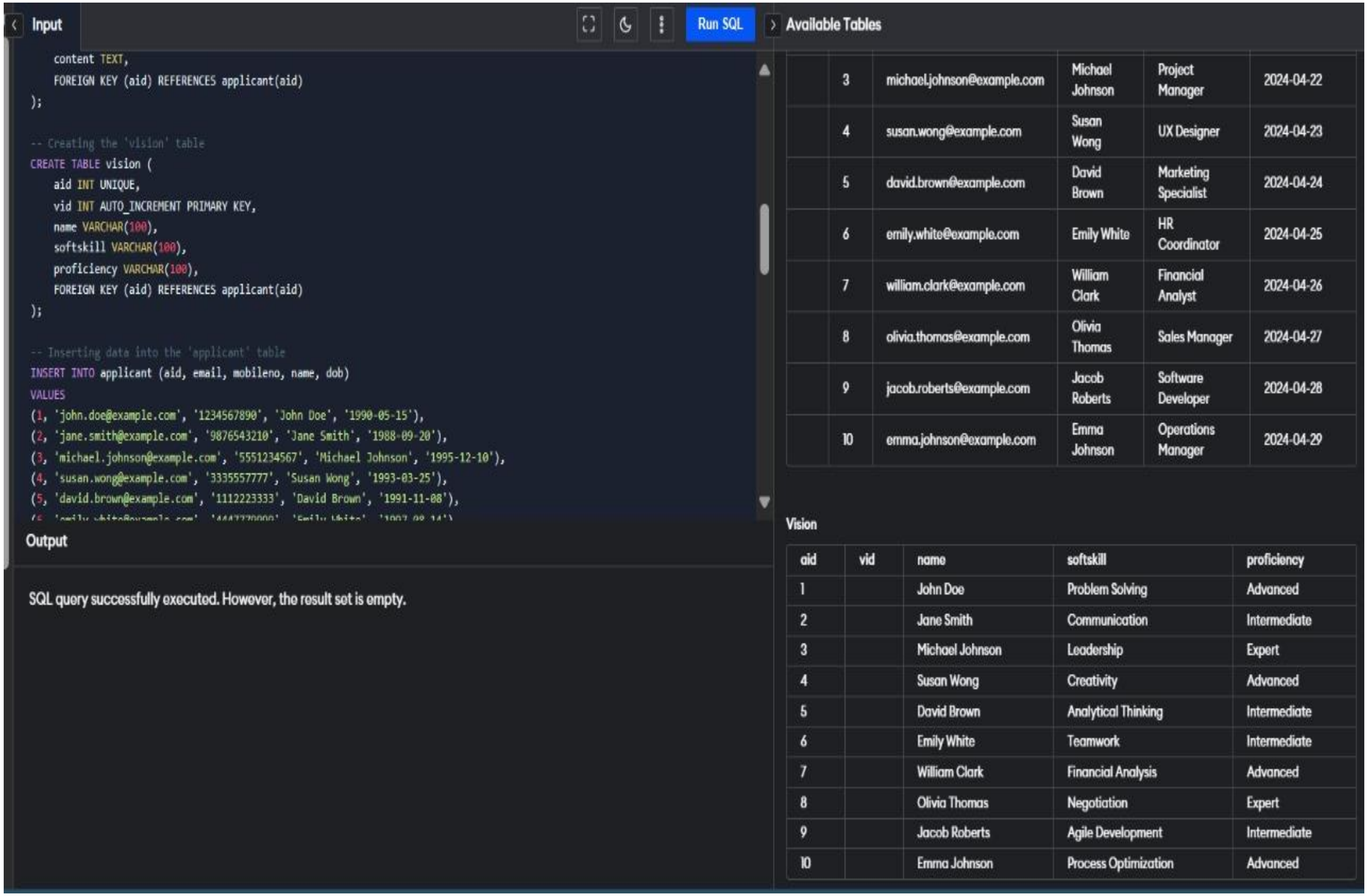
```sql
-- Creating the 'feedback' table
CREATE TABLE feedback (
    aid INT UNIQUE,
    fid INT AUTO_INCREMENT PRIMARY KEY,
    rating INT,
    feedbacktype VARCHAR(100),
    content TEXT,
    FOREIGN KEY (aid) REFERENCES applicant(aid)
);

-- Creating the 'vision' table
CREATE TABLE vision (
    aid INT UNIQUE,
    vid INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100),
    softskill VARCHAR(100),
    proficiency VARCHAR(100),
    FOREIGN KEY (aid) REFERENCES applicant(aid)
);

-- Inserting data into the 'applicant' table
```

Output

```
SQL query successfully executed. However, the result set is empty.
```

Feedback

| aid | fid | rating | feedbacktype | content |
|---|---|---|---|---|
| 1 | | 4 | Technical | John has strong programming skills and is a quick learner. |
| 2 | | 5 | General | Jane is a great communicator and works well in teams. |
| 3 | | 4 | Management | Michael shows excellent leadership qualities and delivers results effectively. |
| 4 | | 4 | Design | Susan has a keen eye for design and understands user experience well. |
| 5 | | 3 | Marketing | David needs to focus more on data-driven marketing strategies. |
| 6 | | 5 | HR | Emily is very efficient in handling HR tasks and has a good understanding of employee needs. |
| 7 | | 4 | Finance | William is adept at financial analysis and provides valuable insights. |
| 8 | | 5 | Sales | Olivia excels in sales management and effectively leads her team. |
| 9 | | 4 | Technical | Jacob is a skilled developer and contributes effectively to projects. |
| 10 | | 4 | Operations | Emma ensures smooth operations and identifies areas for improvement effectively. |



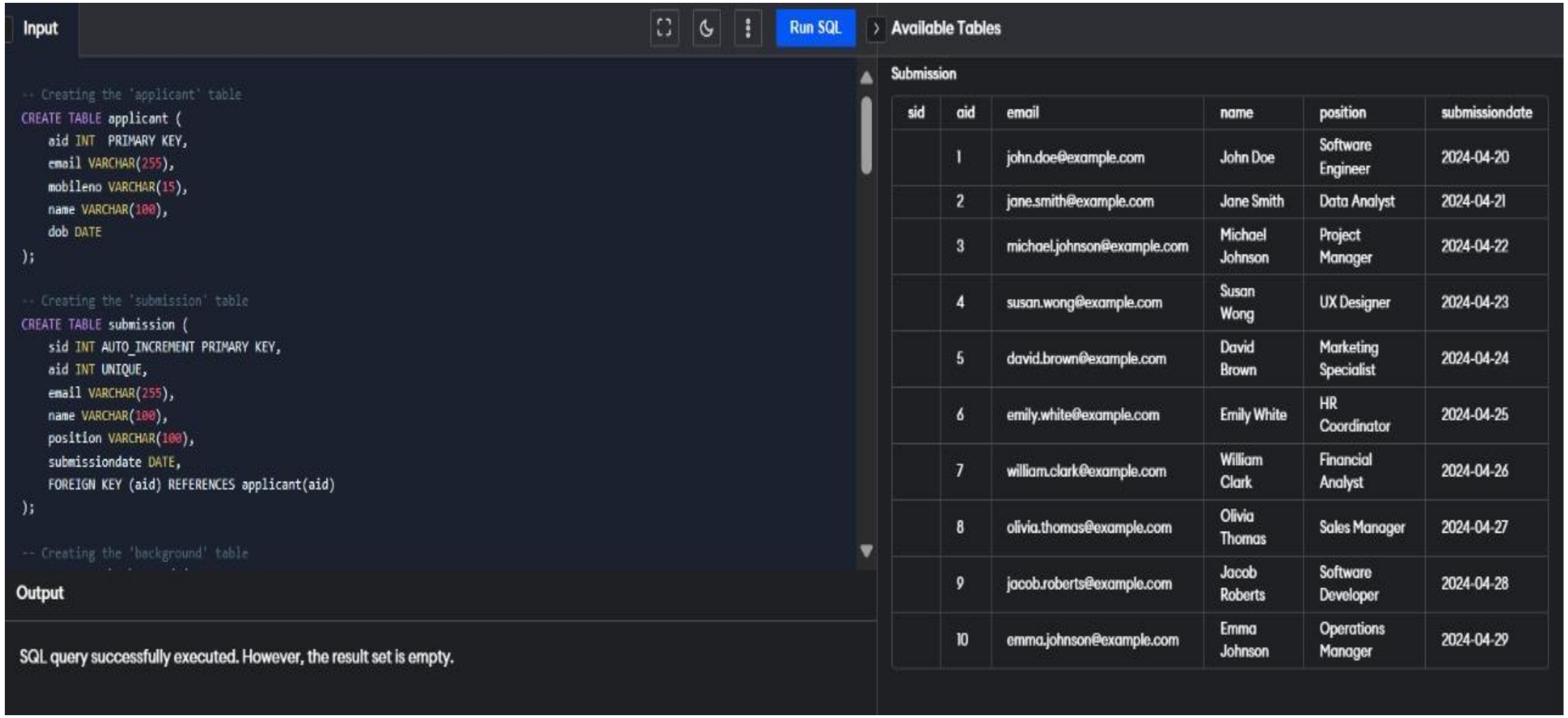
```
    content TEXT,
    FOREIGN KEY (aid) REFERENCES applicant(aid)
);

-- Creating the 'vision' table
CREATE TABLE vision (
    aid INT UNIQUE,
    vid INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100),
    softskill VARCHAR(100),
    proficiency VARCHAR(100),
    FOREIGN KEY (aid) REFERENCES applicant(aid)
);

-- Inserting data into the 'applicant' table
INSERT INTO applicant (aid, email, mobileno, name, dob)
VALUES
(1, 'john.doe@example.com', '1234567890', 'John Doe', '1990-05-15'),
(2, 'jane.smith@example.com', '9876543210', 'Jane Smith', '1988-09-20'),
(3, 'michael.johnson@example.com', '5551234567', 'Michael Johnson', '1995-12-10'),
(4, 'susan.wong@example.com', '3335557777', 'Susan Wong', '1993-03-25'),
(5, 'david.brown@example.com', '1112223333', 'David Brown', '1991-11-08'),
```

SQL query successfully executed. However, the result set is empty.

| | | | | | |
|---|---|---|---|---|---|
| | 3 | michael.johnson@example.com | Michael Johnson | Project Manager | 2024-04-22 |
| | 4 | susan.wong@example.com | Susan Wong | UX Designer | 2024-04-23 |
| | 5 | david.brown@example.com | David Brown | Marketing Specialist | 2024-04-24 |
| | 6 | emily.white@example.com | Emily White | HR Coordinator | 2024-04-25 |
| | 7 | william.clark@example.com | William Clark | Financial Analyst | 2024-04-26 |
| | 8 | olivia.thomas@example.com | Olivia Thomas | Sales Manager | 2024-04-27 |
| | 9 | jacob.roberts@example.com | Jacob Roberts | Software Developer | 2024-04-28 |
| | 10 | emma.johnson@example.com | Emma Johnson | Operations Manager | 2024-04-29 |

Vision

| aid | vid | name | softskill | proficiency |
|---|---|---|---|---|
| 1 | | John Doe | Problem Solving | Advanced |
| 2 | | Jane Smith | Communication | Intermediate |
| 3 | | Michael Johnson | Leadership | Expert |
| 4 | | Susan Wong | Creativity | Advanced |
| 5 | | David Brown | Analytical Thinking | Intermediate |
| 6 | | Emily White | Teamwork | Intermediate |
| 7 | | William Clark | Financial Analysis | Advanced |
| 8 | | Olivia Thomas | Negotiation | Expert |
| 9 | | Jacob Roberts | Agile Development | Intermediate |
| 10 | | Emma Johnson | Process Optimization | Advanced |



```
-- Creating the 'applicant' table
CREATE TABLE applicant (
    aid INT  PRIMARY KEY,
    email VARCHAR(255),
    mobileno VARCHAR(15),
    name VARCHAR(100),
    dob DATE
);

-- Creating the 'submission' table
CREATE TABLE submission (
    sid INT AUTO_INCREMENT PRIMARY KEY,
    aid INT UNIQUE,
    email VARCHAR(255),
    name VARCHAR(100),
    position VARCHAR(100),
    submissiondate DATE,
    FOREIGN KEY (aid) REFERENCES applicant(aid)
);

-- Creating the 'background' table
```

```
SQL query successfully executed. However, the result set is empty.
```

Submission

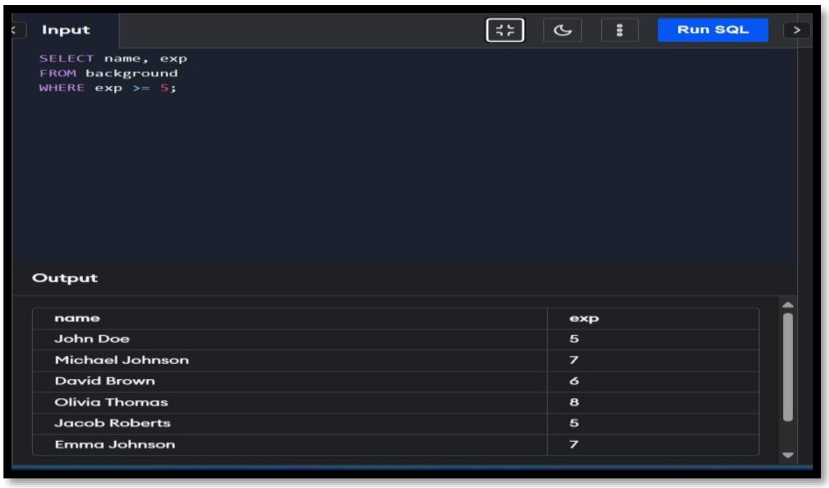
| sid | aid | email | name | position | submissiondate |
|---|---|---|---|---|---|
| | 1 | john.doe@example.com | John Doe | Software Engineer | 2024-04-20 |
| | 2 | jane.smith@example.com | Jane Smith | Data Analyst | 2024-04-21 |
| | 3 | michael.johnson@example.com | Michael Johnson | Project Manager | 2024-04-22 |
| | 4 | susan.wong@example.com | Susan Wong | UX Designer | 2024-04-23 |
| | 5 | david.brown@example.com | David Brown | Marketing Specialist | 2024-04-24 |
| | 6 | emily.white@example.com | Emily White | HR Coordinator | 2024-04-25 |
| | 7 | william.clark@example.com | William Clark | Financial Analyst | 2024-04-26 |
| | 8 | olivia.thomas@example.com | Olivia Thomas | Sales Manager | 2024-04-27 |
| | 9 | jacob.roberts@example.com | Jacob Roberts | Software Developer | 2024-04-28 |
| | 10 | emma.johnson@example.com | Emma Johnson | Operations Manager | 2024-04-29 |

--relational operator

SELECT name, exp

FROM background

WHERE exp >= 5;



```
SELECT name, exp
FROM background
WHERE exp >= 5;
```

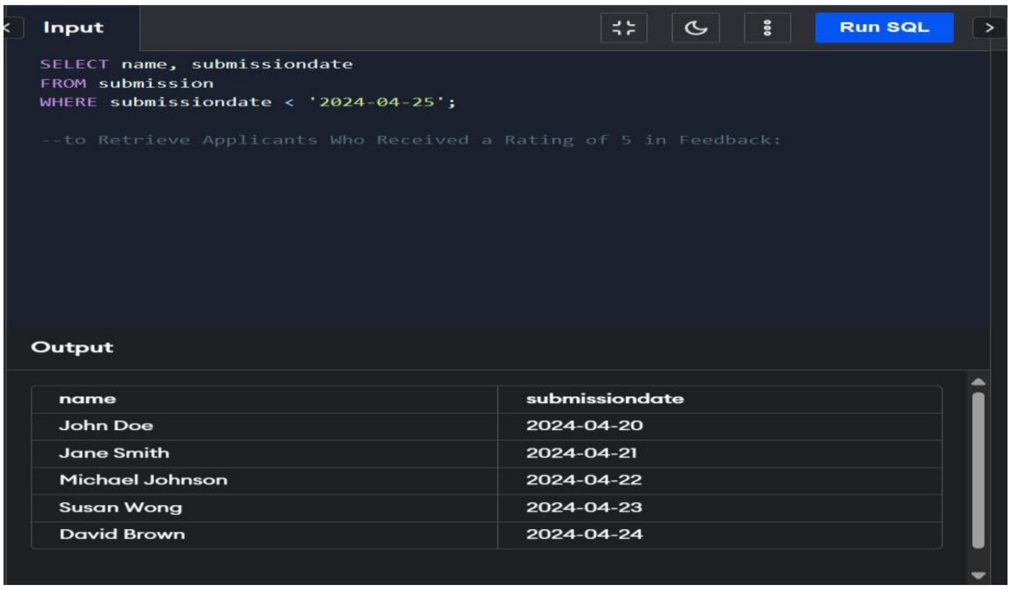
| name | exp |
|---|---|
| John Doe | 5 |
| Michael Johnson | 7 |
| David Brown | 6 |
| Olivia Thomas | 8 |
| Jacob Roberts | 5 |
| Emma Johnson | 7 |

--Query to Retrieve Applicants Who Submitted Before April 25, 2024:

SELECT name, submissiondate

FROM submission

WHERE submissiondate < '2024-04-25';



```
SELECT name, submissiondate
FROM submission
WHERE submissiondate < '2024-04-25';

--to Retrieve Applicants Who Received a Rating of 5 in Feedback:
```

| name | submissiondate |
|---|---|
| John Doe | 2024-04-20 |
| Jane Smith | 2024-04-21 |
| Michael Johnson | 2024-04-22 |
| Susan Wong | 2024-04-23 |
| David Brown | 2024-04-24 |

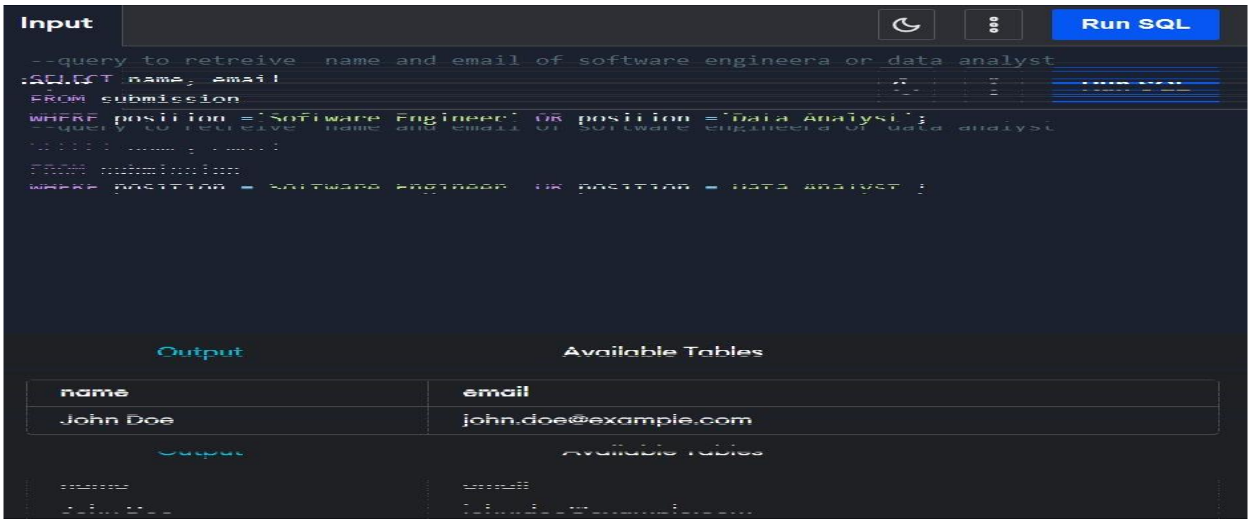
to Retrieve Applicants Who Received a Rating of 5 in Feedback:

--query to retreive  name and email of software engineera or data analyst

SELECT name, email

FROM submission

WHERE position = 'Software Engineer' OR position = 'Data Analyst';



```
--query to retreive  name and email of software engineera or data analyst
SELECT name, email
FROM submission
WHERE position ='Software Engineer' OR position ='Data Analyst';
```

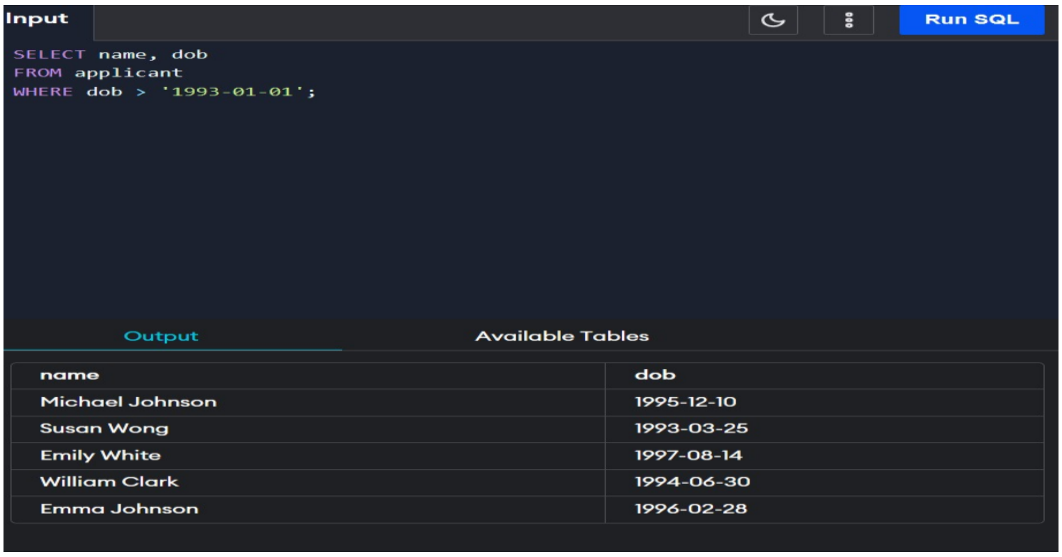
```
name        email
John Doe    john.doe@example.com
```

Query to Retrieve Applicants Born After January 1, 1993:

SELECT name, dob

FROM applicant

WHERE dob > '1993-01-01';



```
SELECT name, dob
FROM applicant
WHERE dob > '1993-01-01';
```

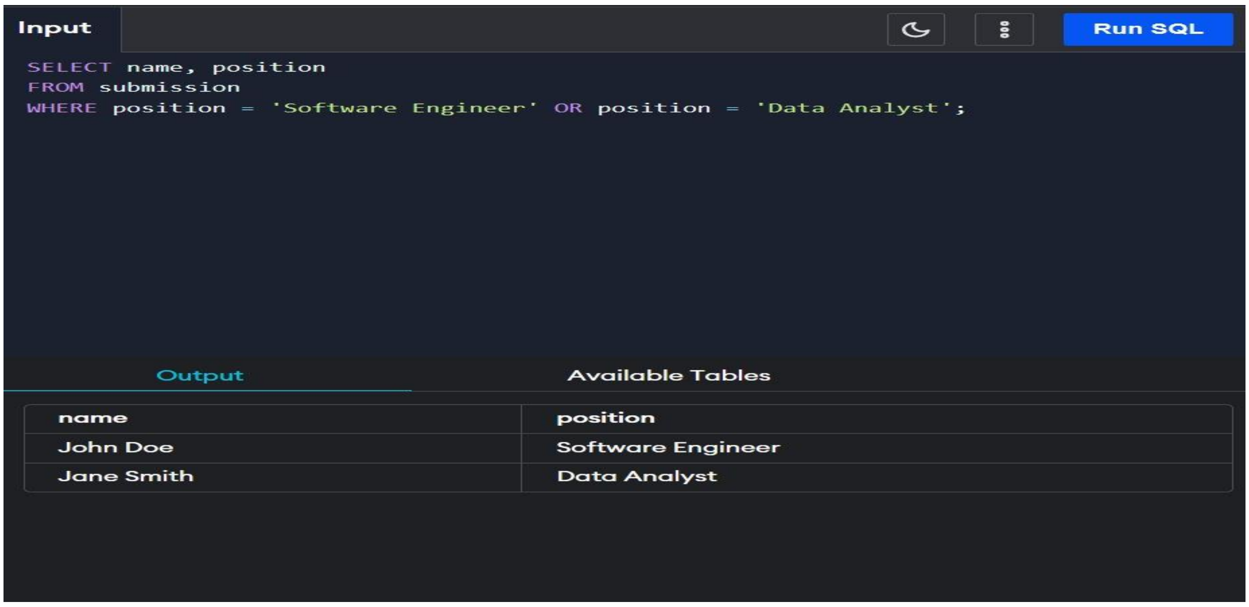
```
name              dob
Michael Johnson   1995-12-10
Susan Wong        1993-03-25
Emily White       1997-08-14
William Clark     1994-06-30
Emma Johnson      1996-02-28
```

-- logical operators

SELECT name, position

FROM submission

WHERE position = 'Software Engineer' OR position = 'Data Analyst';



```
SELECT name, position
FROM submission
WHERE position = 'Software Engineer' OR position = 'Data Analyst';
```

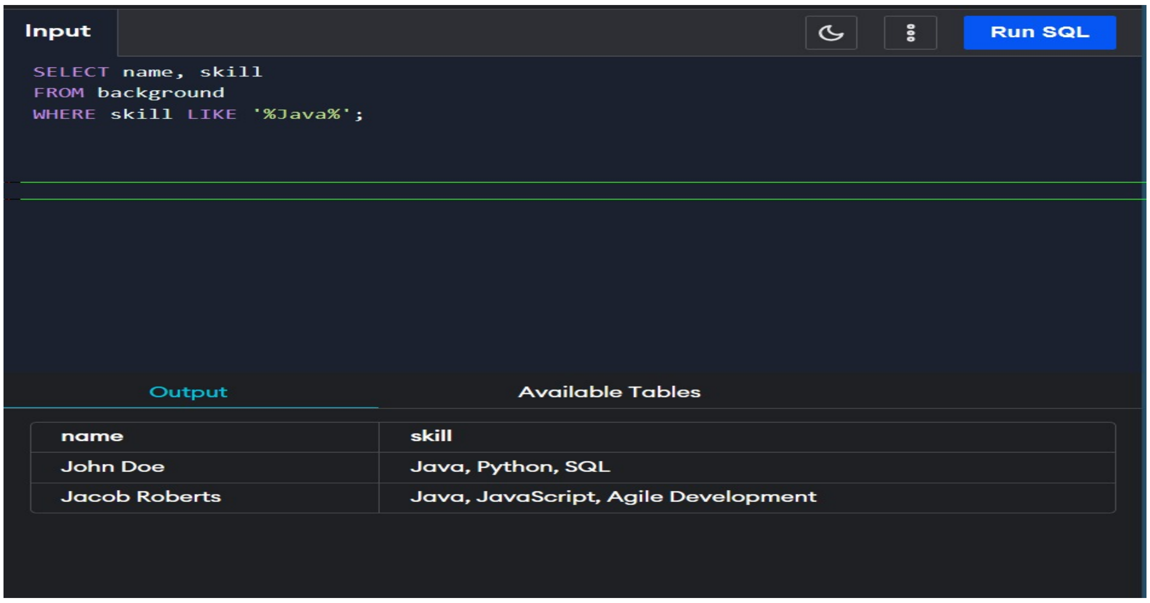
| name | position |
|---|---|
| John Doe | Software Engineer |
| Jane Smith | Data Analyst |

Query to Retrieve Applicants Who Applied for Technical Positions

SELECT name, position

FROM submission

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



```
SELECT name, skill
FROM background
WHERE skill LIKE '%Java%';
```

| name | skill |
|---|---|
| John Doe | Java, Python, SQL |
| Jacob Roberts | Java, JavaScript, Agile Development |

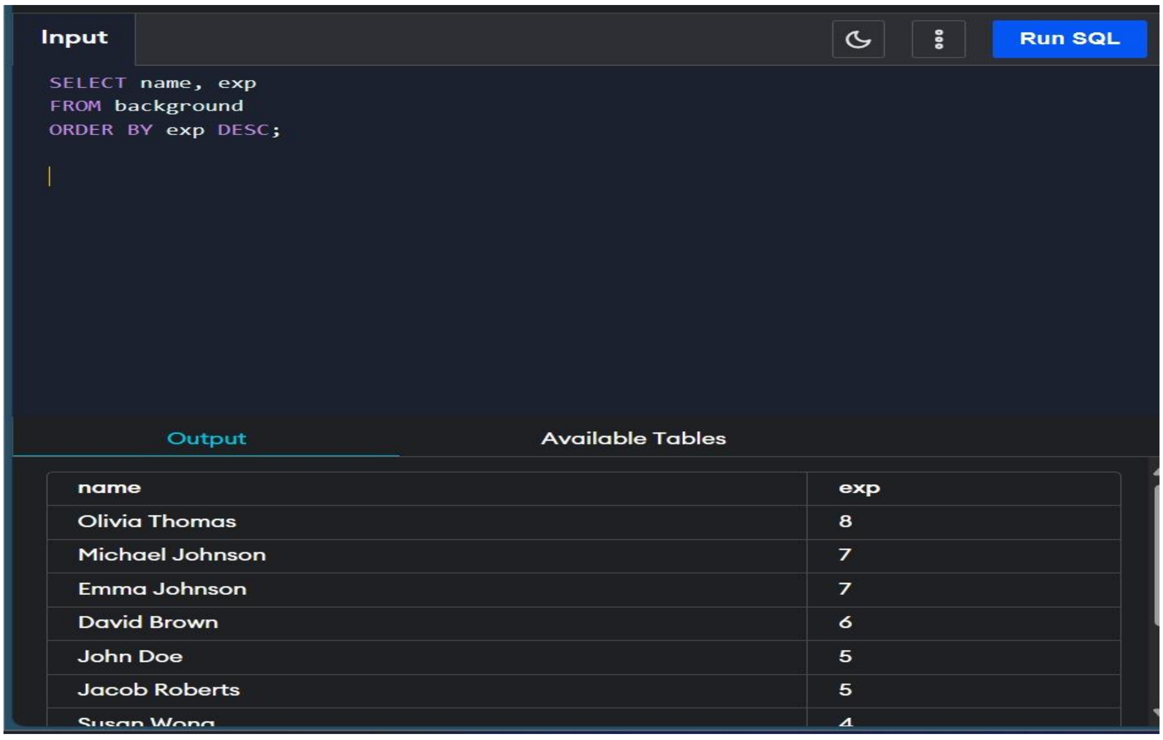
-- order by

query to retrieve name and experience of applicant from background table:

SELECT name, exp

FROM background

ORDER BY exp DESC;



```
SELECT name, exp
FROM background
ORDER BY exp DESC;
```

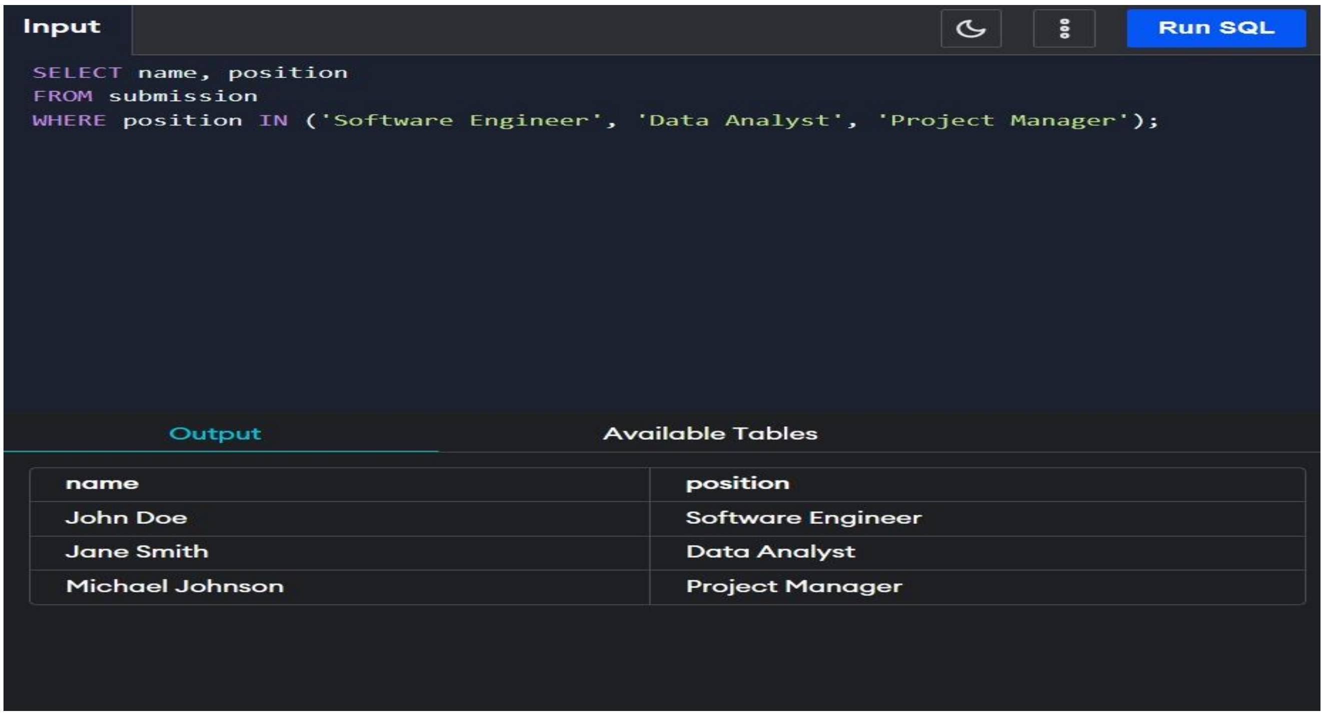
| name | exp |
|---|---|
| Olivia Thomas | 8 |
| Michael Johnson | 7 |
| Emma Johnson | 7 |
| David Brown | 6 |
| John Doe | 5 |
| Jacob Roberts | 5 |
| Susan Wong | 4 |

--IN

SELECT name, position

FROM submission

WHERE position IN ('Software Engineer', 'Data Analyst', 'Project Manager');



```
SELECT name, position
FROM submission
WHERE position IN ('Software Engineer', 'Data Analyst', 'Project Manager');
```

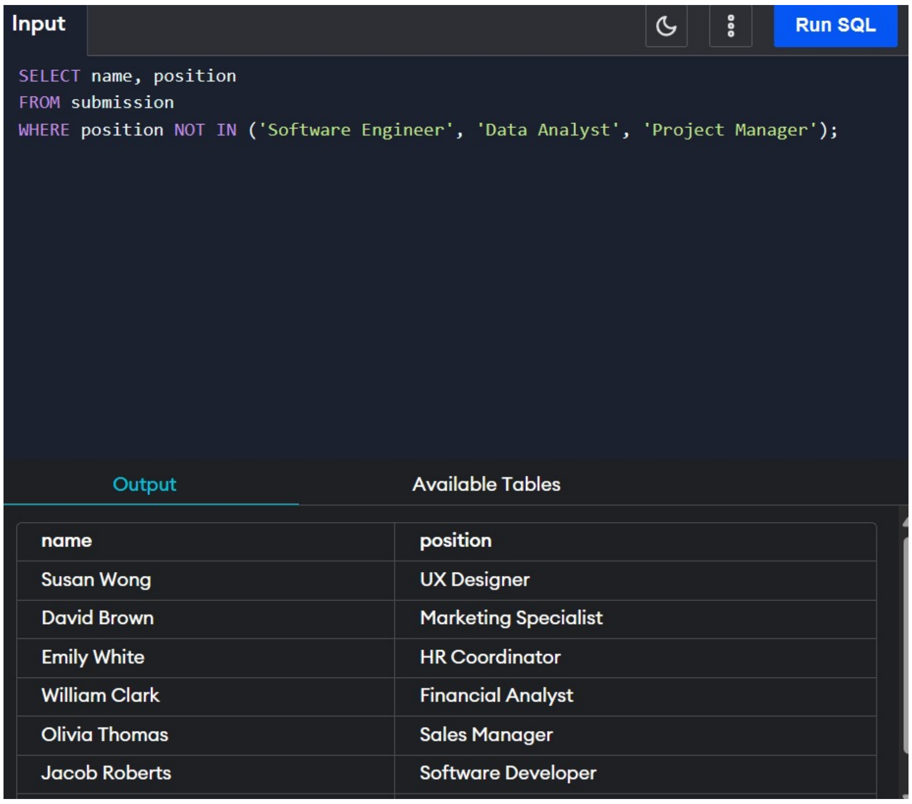
| name | position |
|---|---|
| John Doe | Software Engineer |
| Jane Smith | Data Analyst |
| Michael Johnson | Project Manager |

--not in

SELECT name, position

FROM submission

WHERE position NOT IN ('Software Engineer', 'Data Analyst', 'Project Manager');



```
SELECT name, position
FROM submission
WHERE position NOT IN ('Software Engineer', 'Data Analyst', 'Project Manager');
```

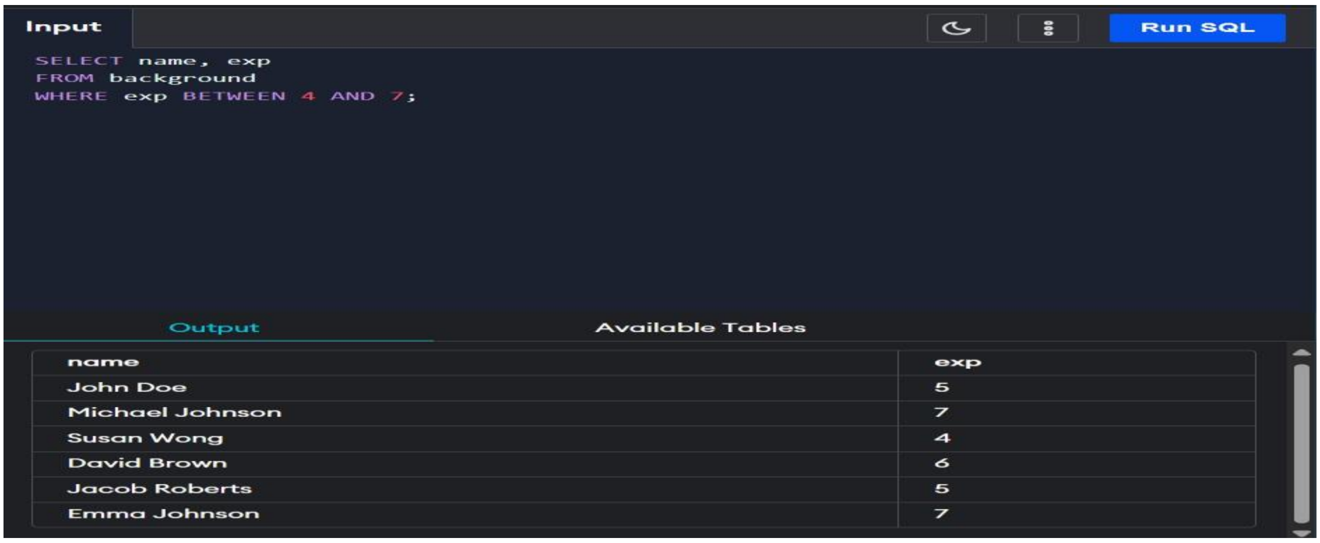
| name | position |
|---|---|
| Susan Wong | UX Designer |
| David Brown | Marketing Specialist |
| Emily White | HR Coordinator |
| William Clark | Financial Analyst |
| Olivia Thomas | Sales Manager |
| Jacob Roberts | Software Developer |

--BETWEEN AND

SELECT name, exp

FROM background

WHERE exp BETWEEN 4 AND 7;



```
SELECT name, exp
FROM background
WHERE exp BETWEEN 4 AND 7;
```

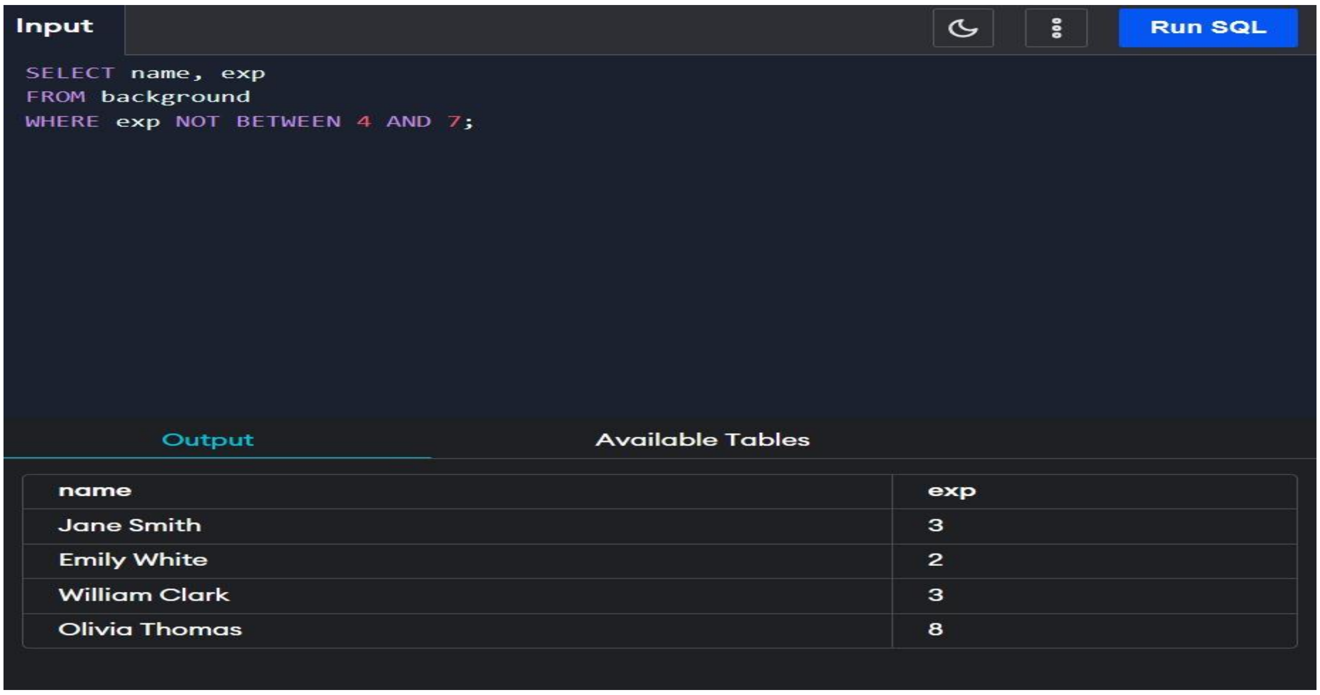
```
name              exp
John Doe          5
Michael Johnson   7
Susan Wong        4
David Brown       6
Jacob Roberts     5
Emma Johnson      7
```

--NOT BETWEEN

SELECT name, exp

FROM background

WHERE exp NOT BETWEEN 4 AND 7;



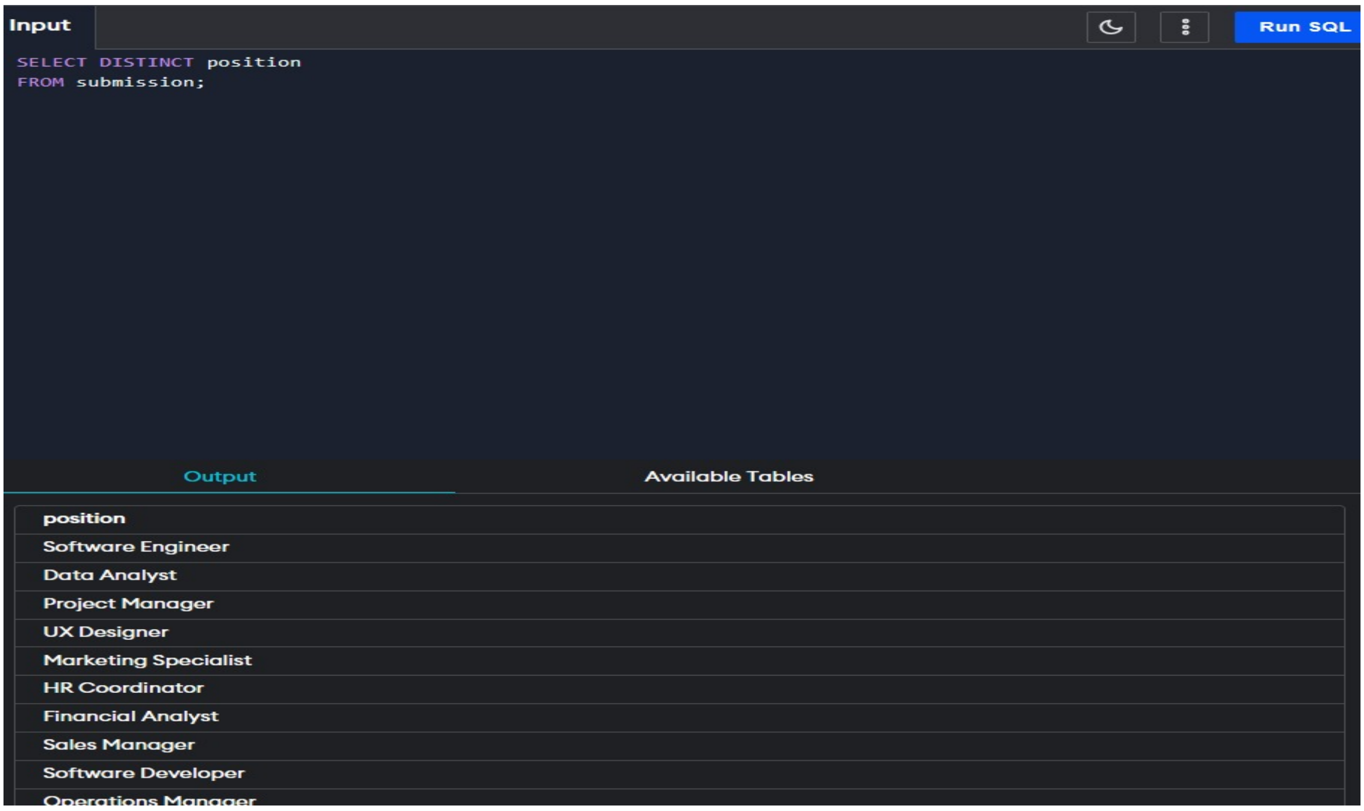
```
SELECT name, exp
FROM background
WHERE exp NOT BETWEEN 4 AND 7;
```

```
name              exp
Jane Smith        3
Emily White       2
William Clark     3
Olivia Thomas     8
```

--DISTINCT

SELECT DISTINCT position

FROM submission;



```
SELECT DISTINCT position
FROM submission;
```

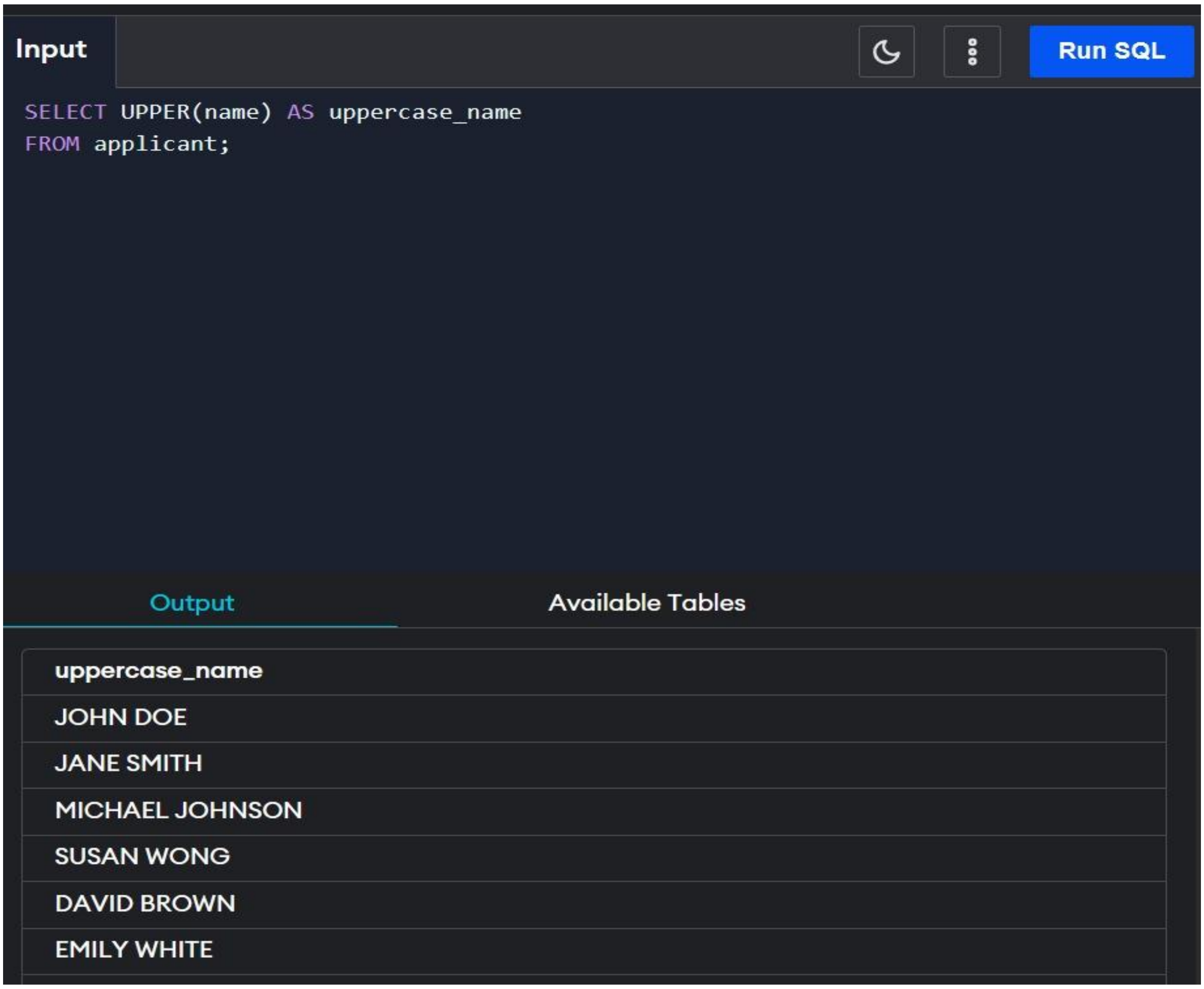
```
position
Software Engineer
Data Analyst
Project Manager
UX Designer
Marketing Specialist
HR Coordinator
Financial Analyst
Sales Manager
Software Developer
Operations Manager
```

--- case manipulation

Query to Convert Names to Uppercase:

SELECT UPPER(name) AS uppercase_name

FROM applicant;



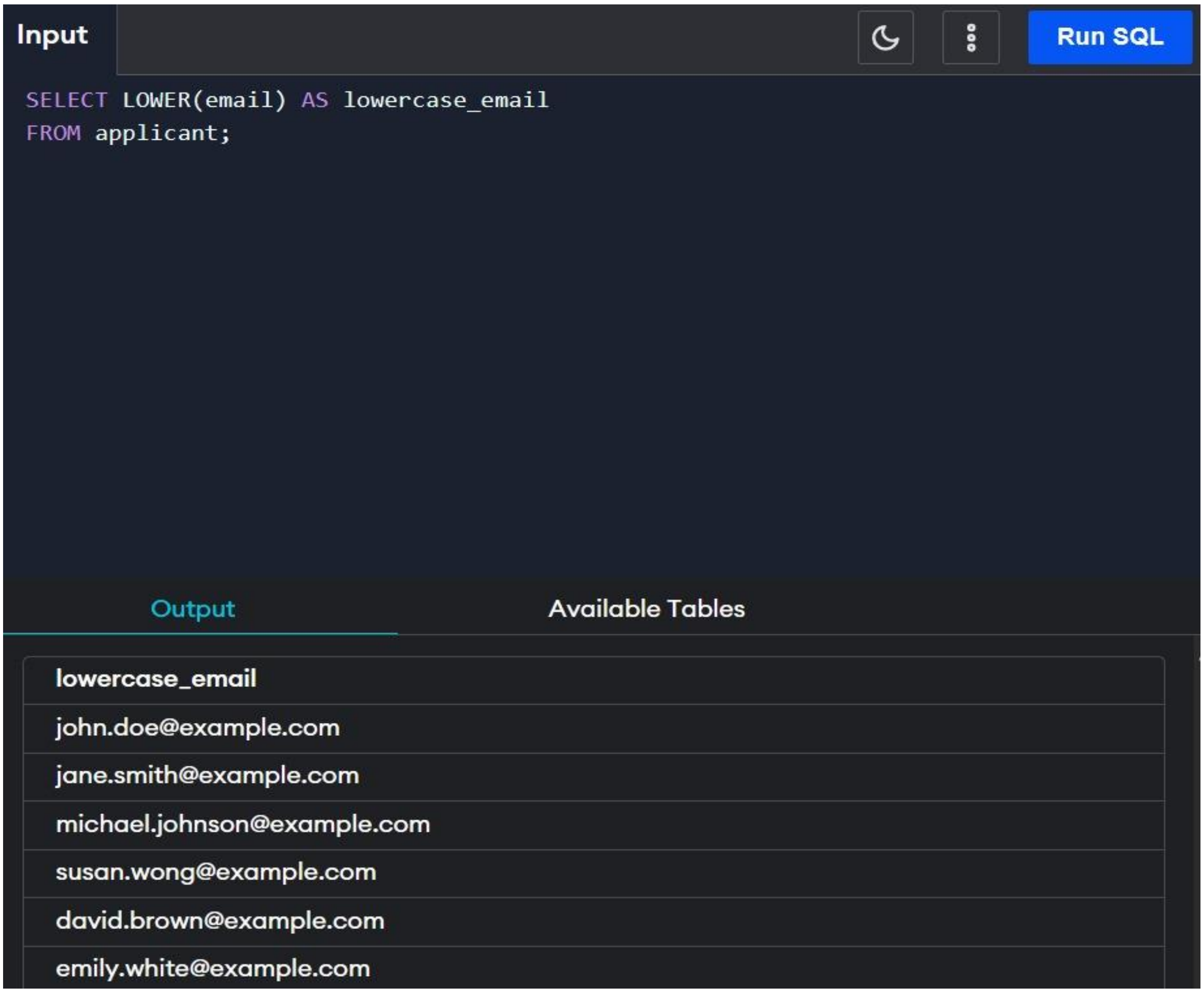
```
SELECT UPPER(name) AS uppercase_name
FROM applicant;
```

```
uppercase_name
JOHN DOE
JANE SMITH
MICHAEL JOHNSON
SUSAN WONG
DAVID BROWN
EMILY WHITE
```

Query to Convert Emails to Lowercase:

SELECT LOWER(email) AS lowercase_email

FROM applicant;



```
SELECT LOWER(email) AS lowercase_email
FROM applicant;
```

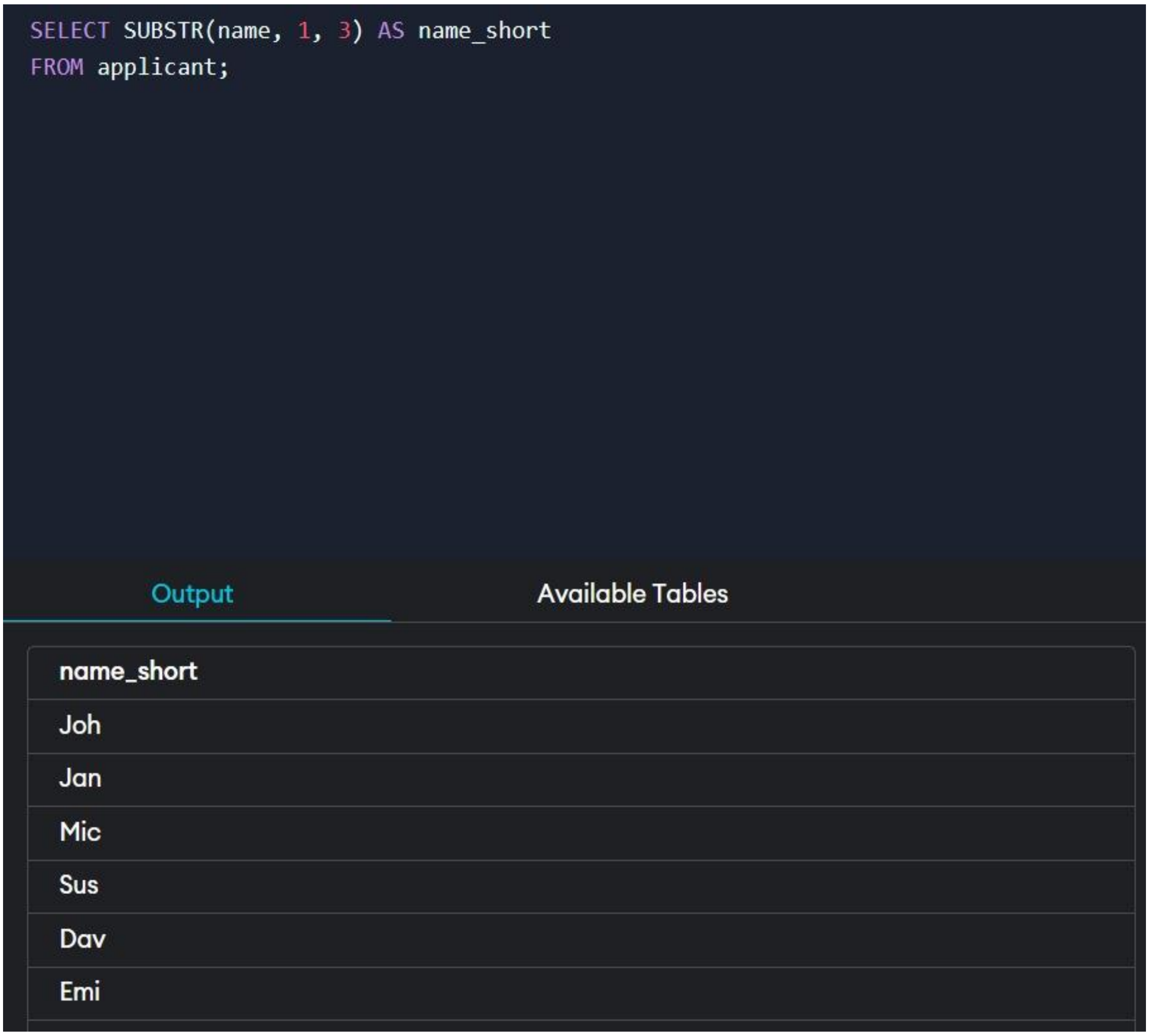
```
lowercase_email
john.doe@example.com
jane.smith@example.com
michael.johnson@example.com
susan.wong@example.com
david.brown@example.com
emily.white@example.com
```

Query to Capitalize Applicant Names:

--extract the first three characters of the names of applicants

SELECT SUBSTR(name, 1, 3) AS name_short

FROM applicant;



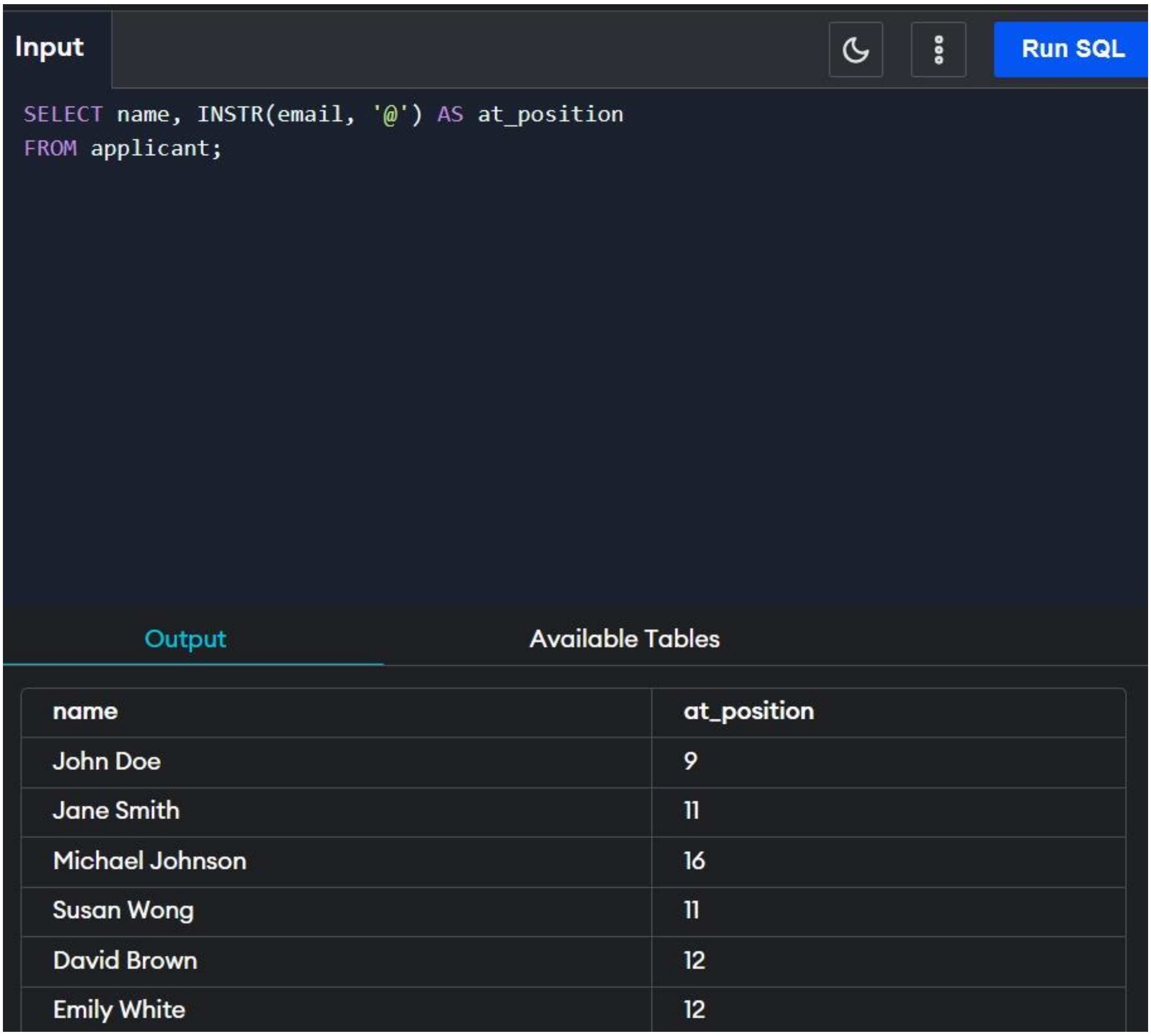
```
SELECT SUBSTR(name, 1, 3) AS name_short
FROM applicant;
```

```
name_short
Joh
Jan
Mic
Sus
Dav
Emi
```

--query selects the names of applicants from the applicant table and finds the position of the '@' symbol in their email addresses using the INSTR function

SELECT name, INSTR(email, '@') AS at_position

FROM applicant;



```
SELECT name, INSTR(email, '@') AS at_position
FROM applicant;
```

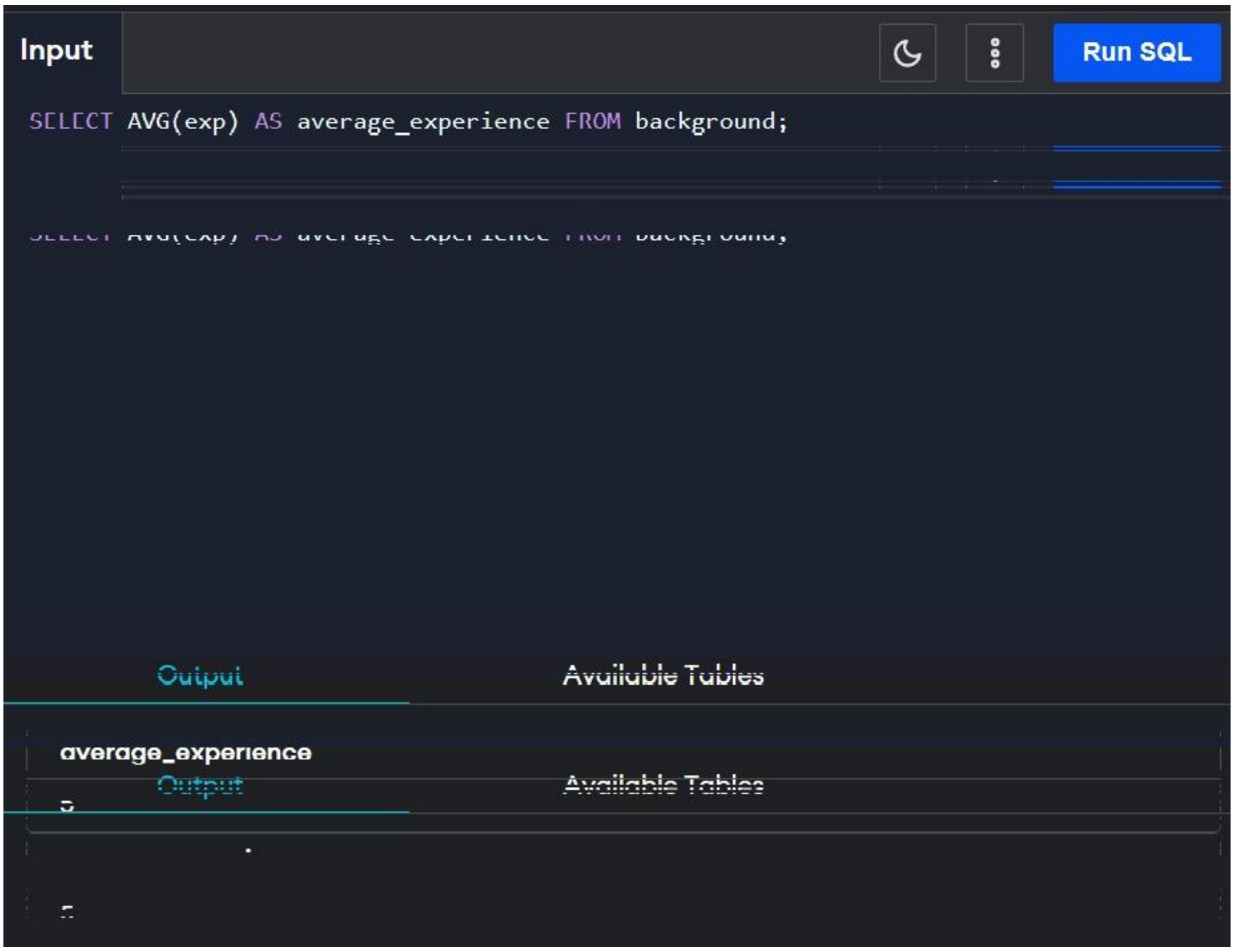
| name | at_position |
|---|---|
| John Doe | 9 |
| Jane Smith | 11 |
| Michael Johnson | 16 |
| Susan Wong | 11 |
| David Brown | 12 |
| Emily White | 12 |

--exp 5

--average

calculates the average years of experience across all applicants:

SELECT AVG(exp) AS average_experience FROM background;

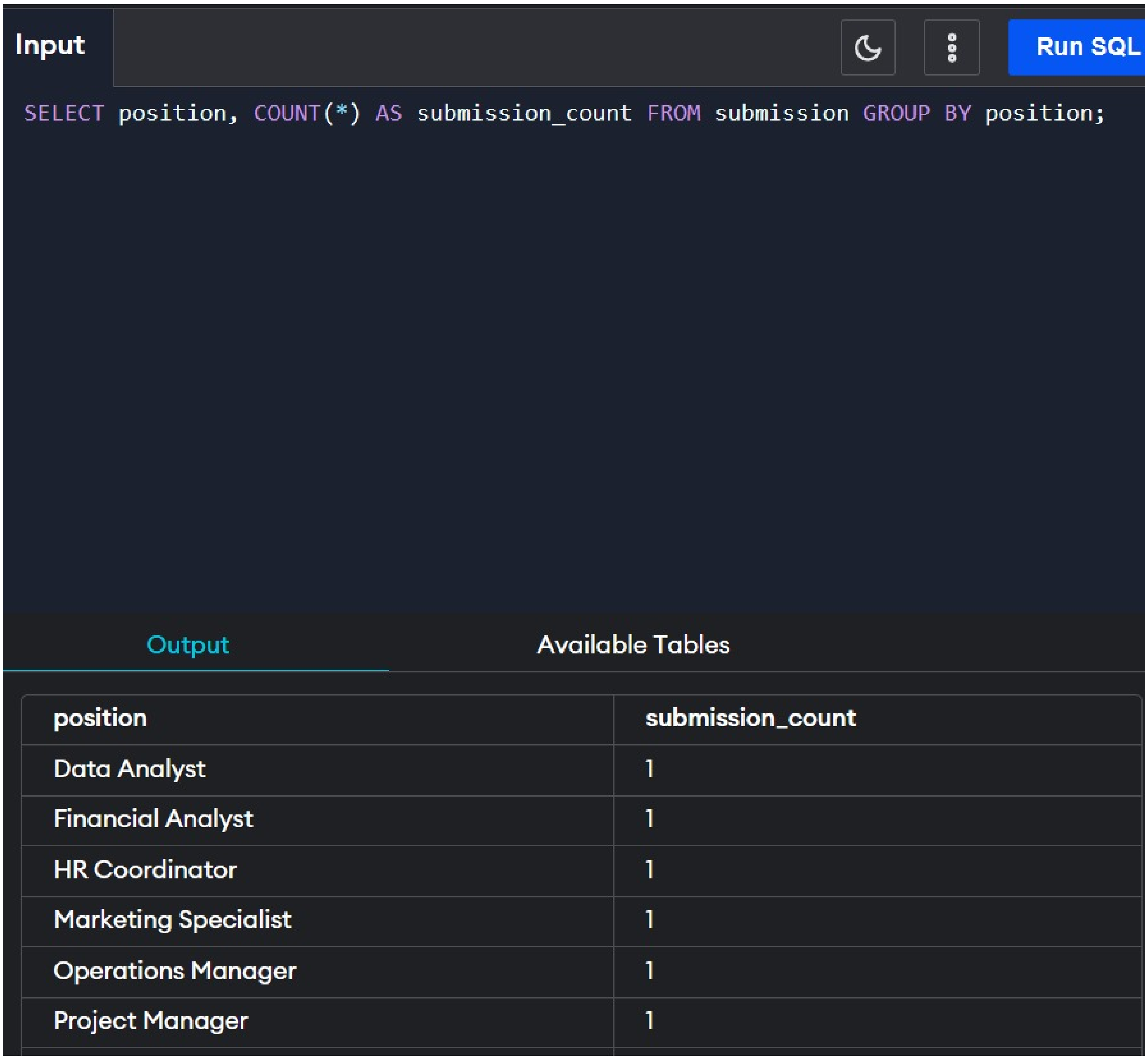


```
SELECT AVG(exp) AS average_experience FROM background;
```

```
average_experience
5
```

--count of submissions grouped by the position

SELECT position, COUNT(*) AS submission_count FROM submission GROUP BY position;

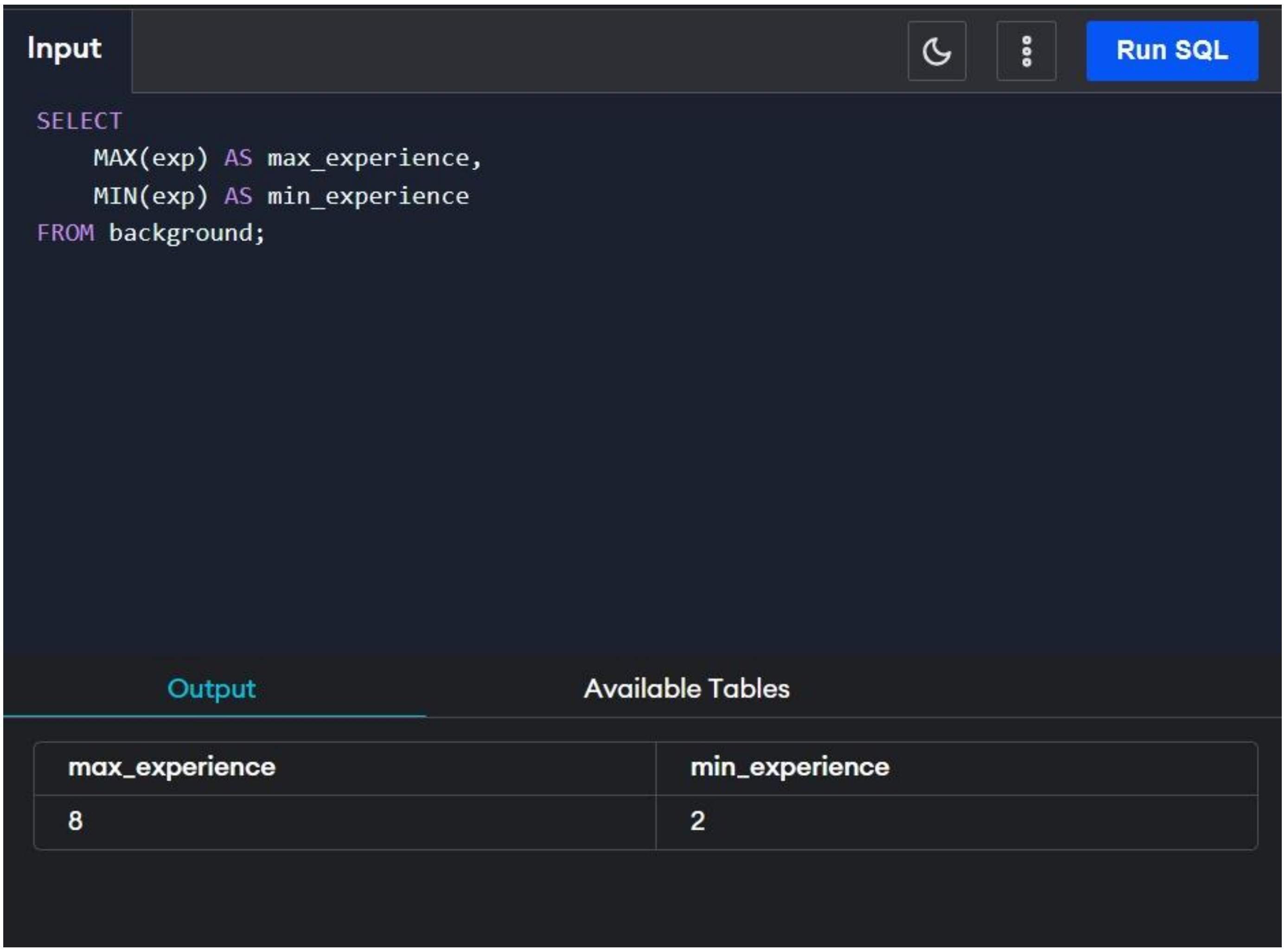


```
SELECT position, COUNT(*) AS submission_count FROM submission GROUP BY position;
```

| position | submission_count |
|---|---|
| Data Analyst | 1 |
| Financial Analyst | 1 |
| HR Coordinator | 1 |
| Marketing Specialist | 1 |
| Operations Manager | 1 |
| Project Manager | 1 |

--max and min

```
SELECT
    MAX(exp) AS max_experience,
    MIN(exp) AS min_experience
FROM background;
```



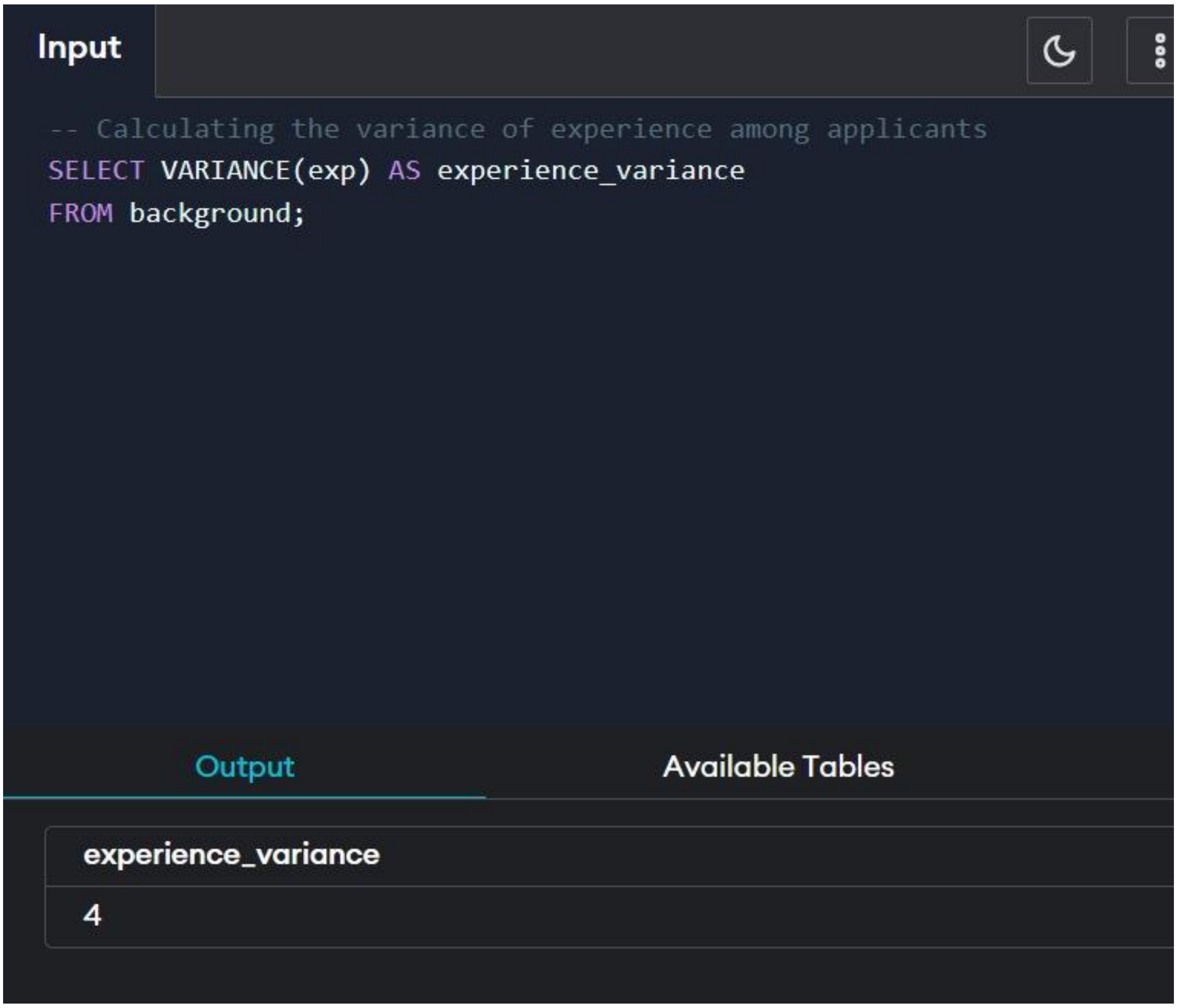
```
SELECT
    MAX(exp) AS max_experience,
    MIN(exp) AS min_experience
FROM background;
```

| max_experience | min_experience |
|---|---|
| 8 | 2 |

-- Calculating the variance of experience among applicants

SELECT VARIANCE(exp) AS experience_variance

FROM background;



```
-- Calculating the variance of experience among applicants
SELECT VARIANCE(exp) AS experience_variance
FROM background;
```

```
experience_variance
4
```

```
--Violating foreign key constraintViolating foreign key constraint

INSERT INTO submission (aid, email, name, position, submissiondate)
VALUES
(11, 'new.applicant@example.com', 'New Applicant', 'Intern', '2024-05-01');

--Violating foreign key constraint in the 'background' table

INSERT INTO background (aid, exp, name, skill, previouscompany)
VALUES
(11, 2, 'New Applicant', 'C++, Java', 'Tech Inc.');

--Violating Primary Key Constraint:

INSERT INTO applicant (aid, email, mobileno, name, dob)
VALUES
(1, 'new.email@example.com', '9998887777', 'New Applicant', '2000-01-01');

--Violating Check Constraint:

INSERT INTO applicant (aid, email, mobileno, name, dob)
VALUES
(11, 'new.applicant@example.com', '1112223333', 'New Applicant', '2025-01-01');
```

--union

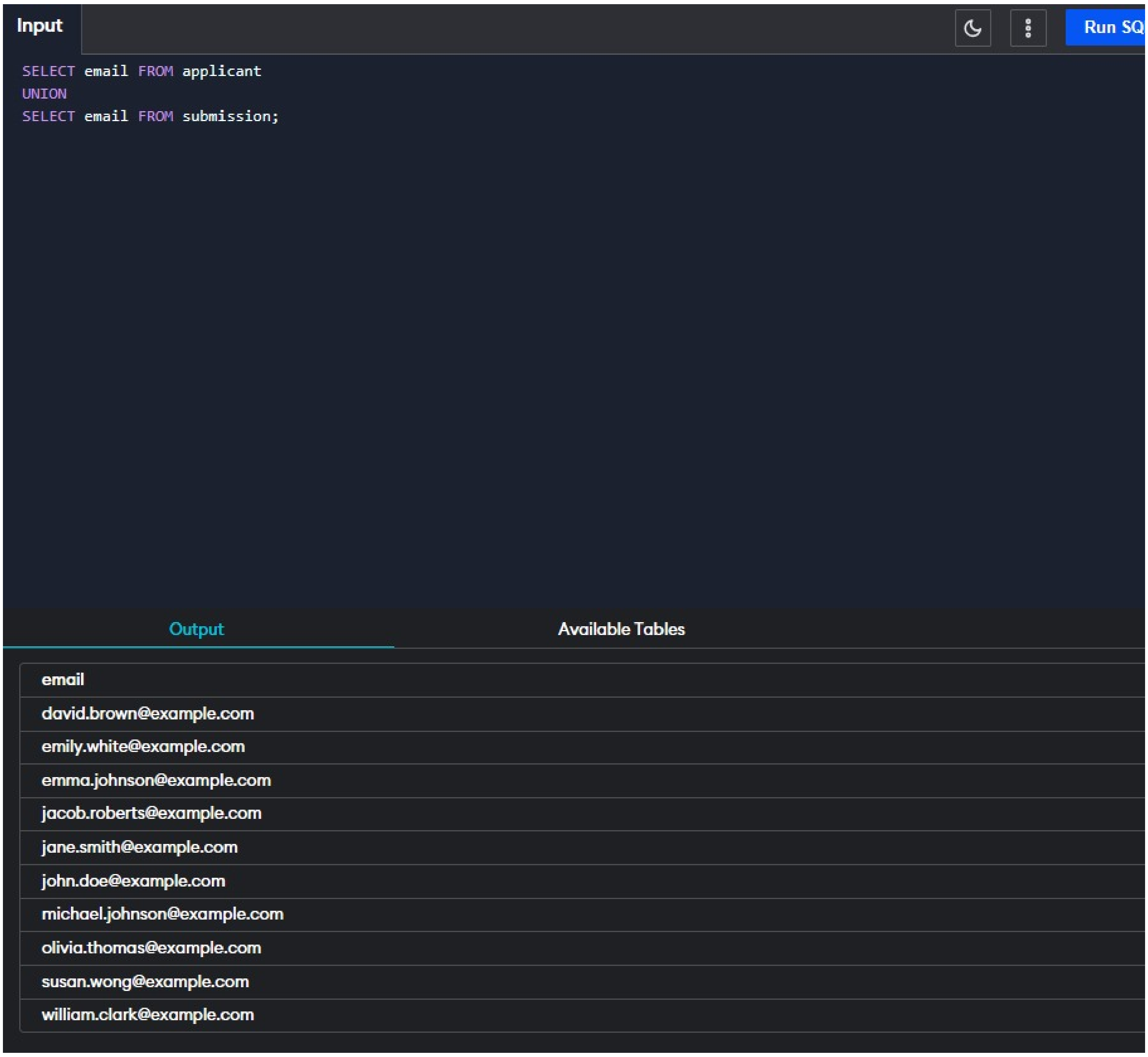
--Finding all unique emails

-- Finding all unique emails from both 'applicant' and 'submission' tables

(SELECT email FROM applicant)

UNION

(SELECT email FROM submission);



```
SELECT email FROM applicant
UNION
SELECT email FROM submission;
```

```
email
david.brown@example.com
emily.white@example.com
emma.johnson@example.com
jacob.roberts@example.com
jane.smith@example.com
john.doe@example.com
michael.johnson@example.com
olivia.thomas@example.com
susan.wong@example.com
william.clark@example.com
```

-- Finding applicants who have submitted an application but have not received any feedback

SELECT name

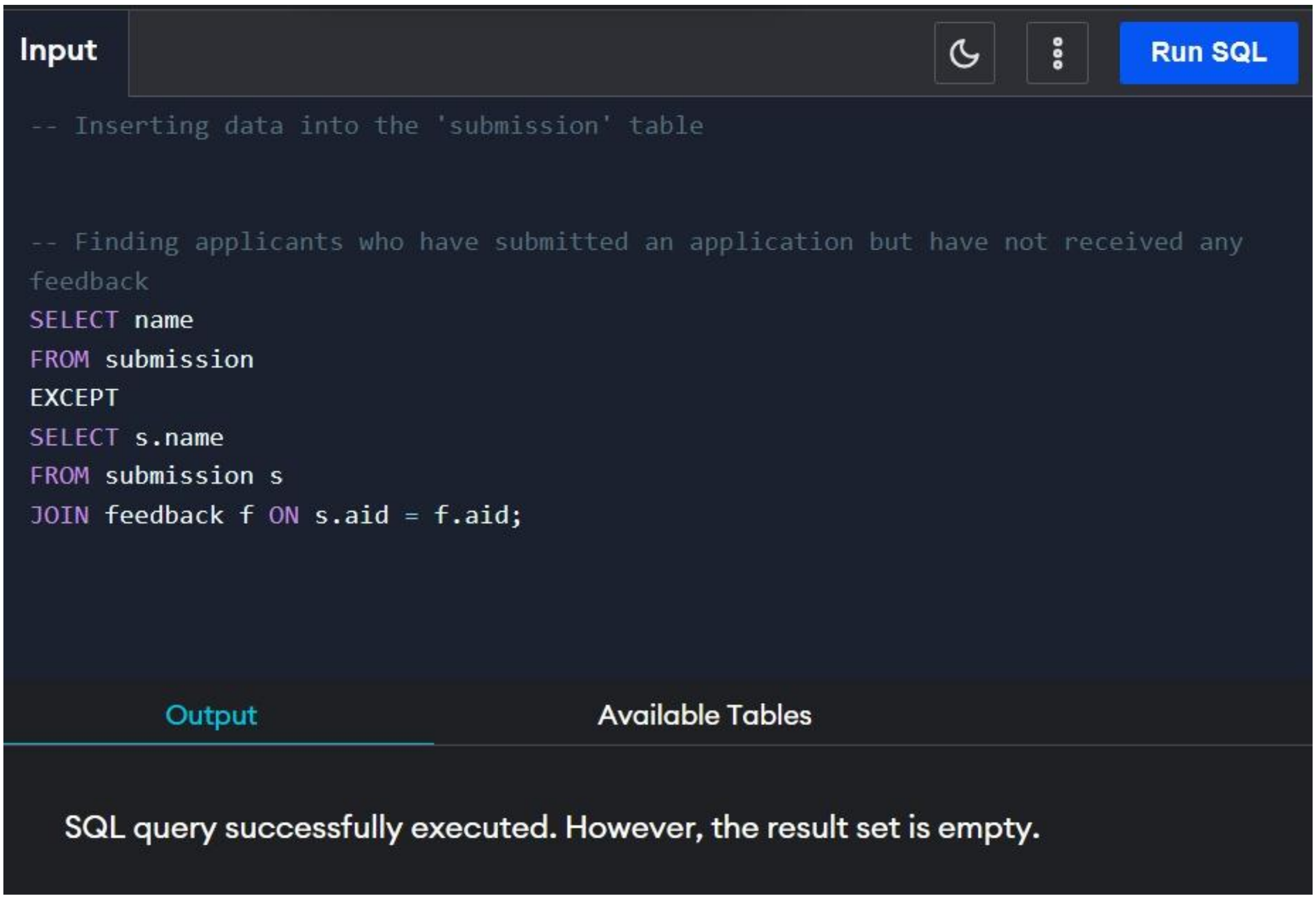
FROM submission

EXCEPT

SELECT s.name

FROM submission s

JOIN feedback f ON s.aid = f.aid;



```
-- Inserting data into the 'submission' table


-- Finding applicants who have submitted an application but have not received any
feedback
SELECT name
FROM submission
EXCEPT
SELECT s.name
FROM submission s
JOIN feedback f ON s.aid = f.aid;
```

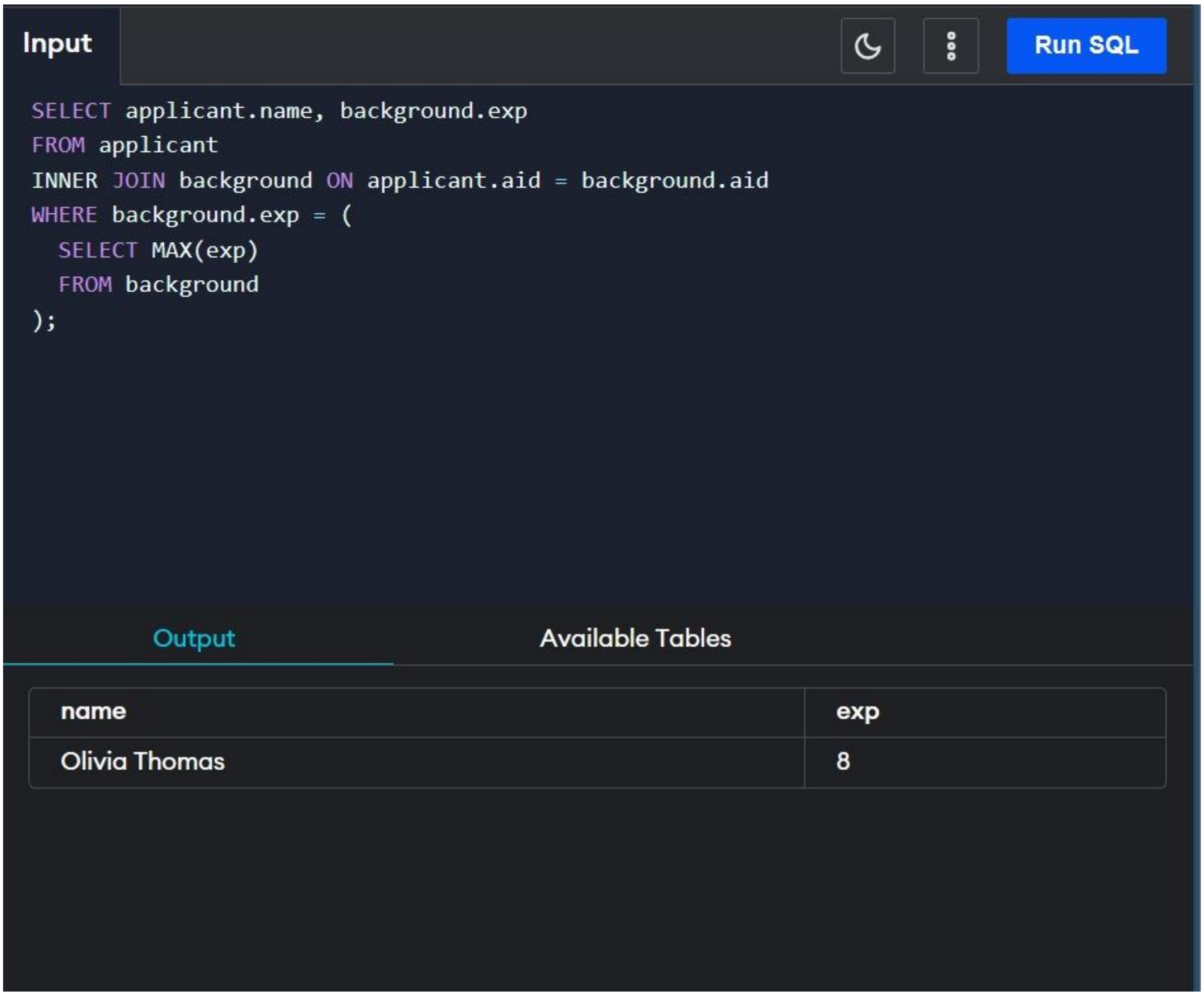
```
SQL query successfully executed. However, the result set is empty.
```

exp 9

-- single row subqueries

-- This query finds the applicant with the highest work experience

```
SELECT applicant.name, background.exp

FROM applicant

INNER JOIN background ON applicant.aid = background.aid

WHERE background.exp = (

  SELECT MAX(exp)

  FROM background

);
```



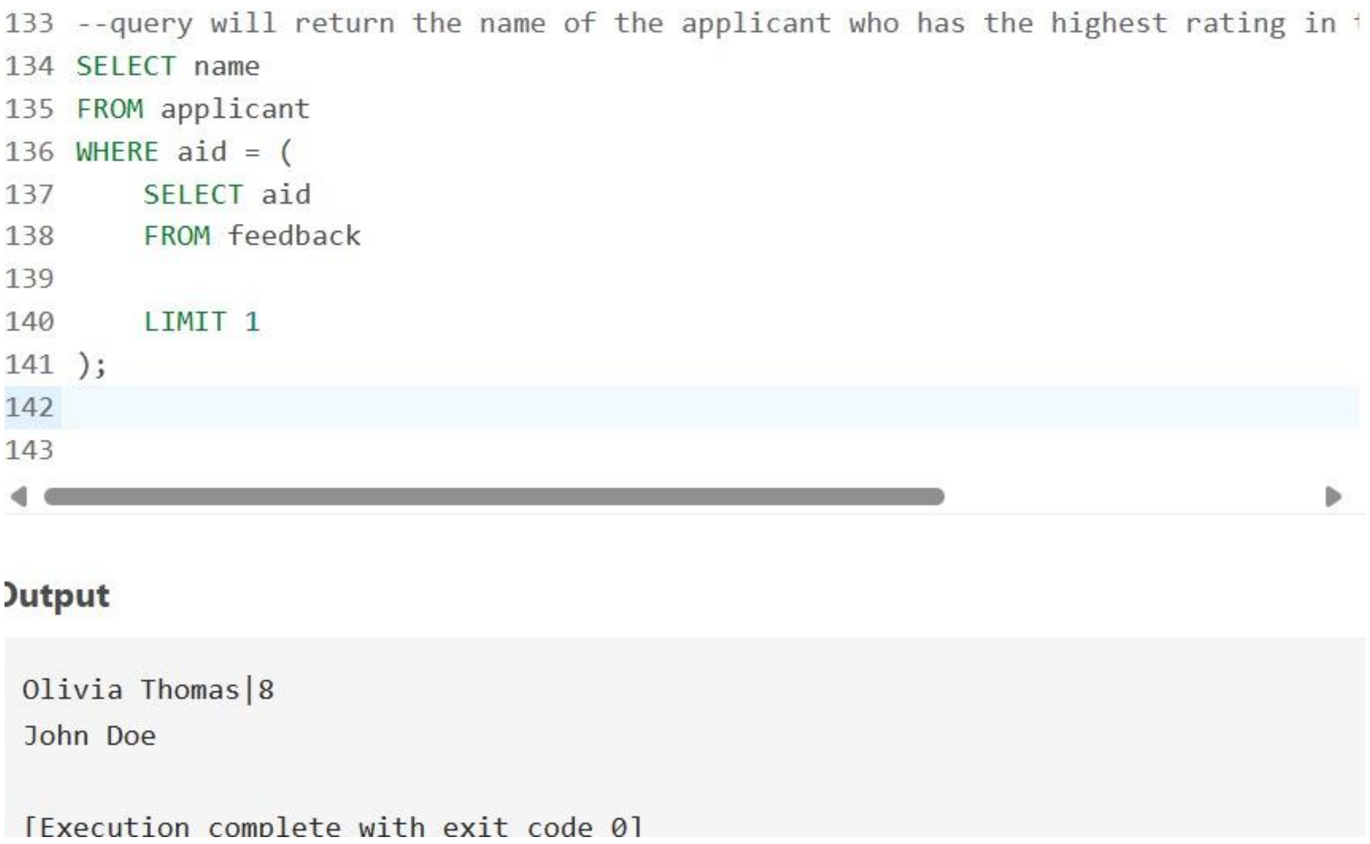
```
SELECT applicant.name, background.exp
FROM applicant
INNER JOIN background ON applicant.aid = background.aid
WHERE background.exp = (
  SELECT MAX(exp)
  FROM background
);
```

| name | exp |
|---|---|
| Olivia Thomas | 8 |

-

-query will return the name of the applicant who has the highest rating in the feedback table:

```
SELECT name

FROM applicant

WHERE aid = (

    SELECT aid

    FROM feedback

    ORDER BY rating DESC

     LIMIT 1

);
```



```
--query will return the name of the applicant who has the highest rating in 
SELECT name
FROM applicant
WHERE aid = (
    SELECT aid
    FROM feedback

    LIMIT 1
);
```

utput

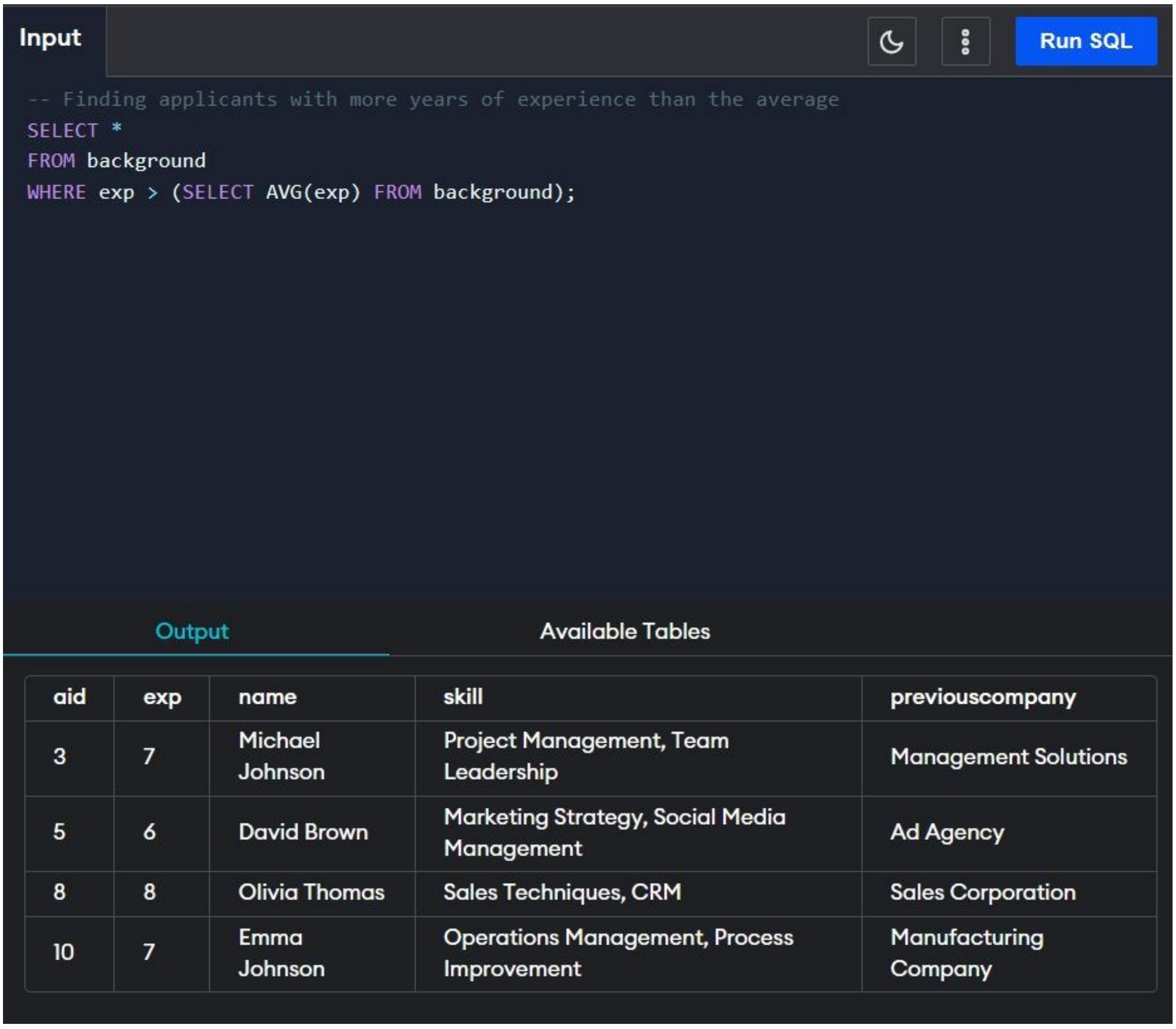
```
Olivia Thomas|8
John Doe

[Execution complete with exit code 0]
```

-- Finding applicants with more years of experience than the average

SELECT *

FROM background

WHERE exp > (SELECT AVG(exp) FROM background);



```
-- Finding applicants with more years of experience than the average
SELECT *
FROM background
WHERE exp > (SELECT AVG(exp) FROM background);
```

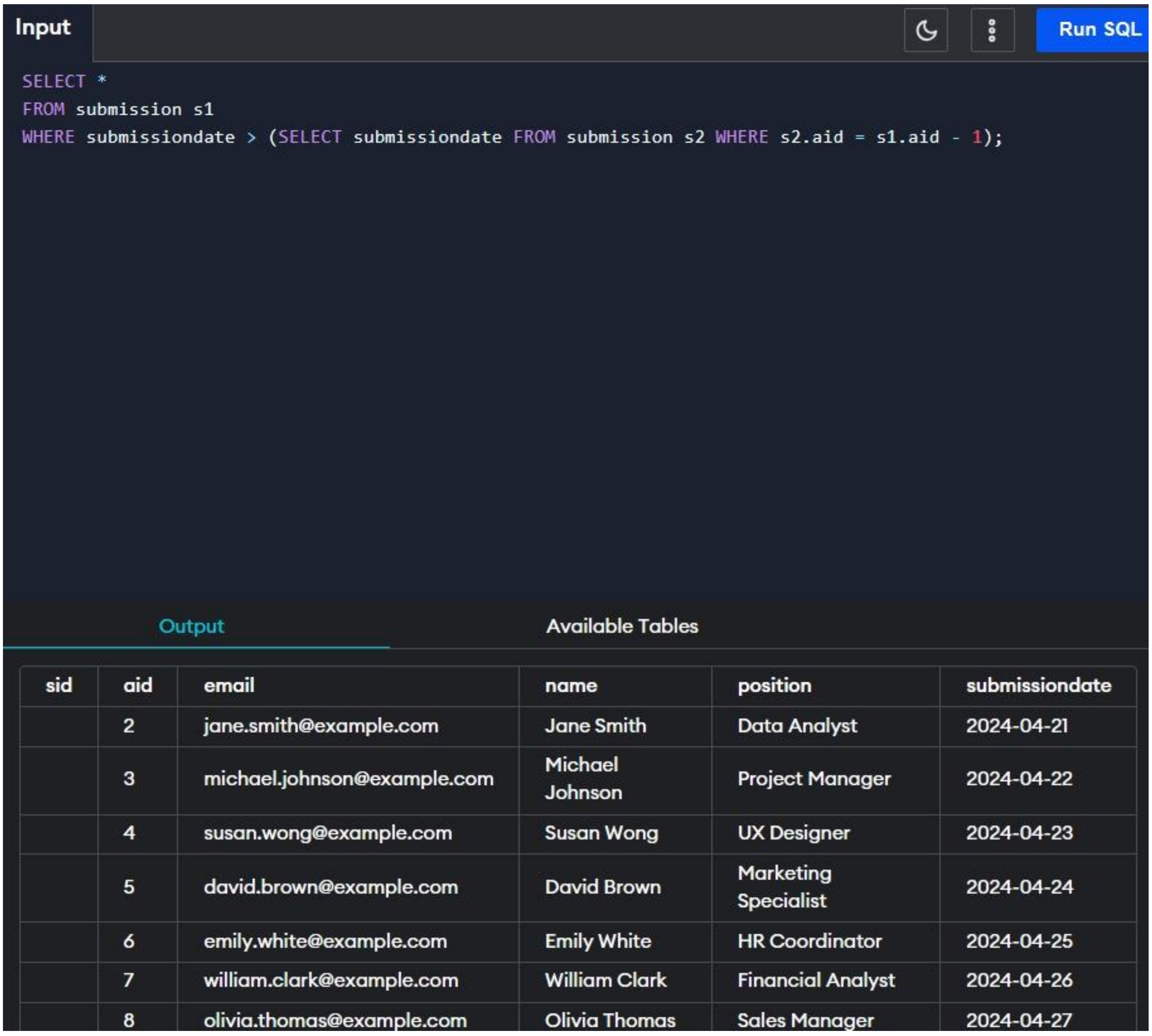
| aid | exp | name | skill | previouscompany |
|---|---|---|---|---|
| 3 | 7 | Michael Johnson | Project Management, Team Leadership | Management Solutions |
| 5 | 6 | David Brown | Marketing Strategy, Social Media Management | Ad Agency |
| 8 | 8 | Olivia Thomas | Sales Techniques, CRM | Sales Corporation |
| 10 | 7 | Emma Johnson | Operations Management, Process Improvement | Manufacturing Company |

-- Finding applicants who submitted for positions after another applicant

SELECT *

FROM submission s1

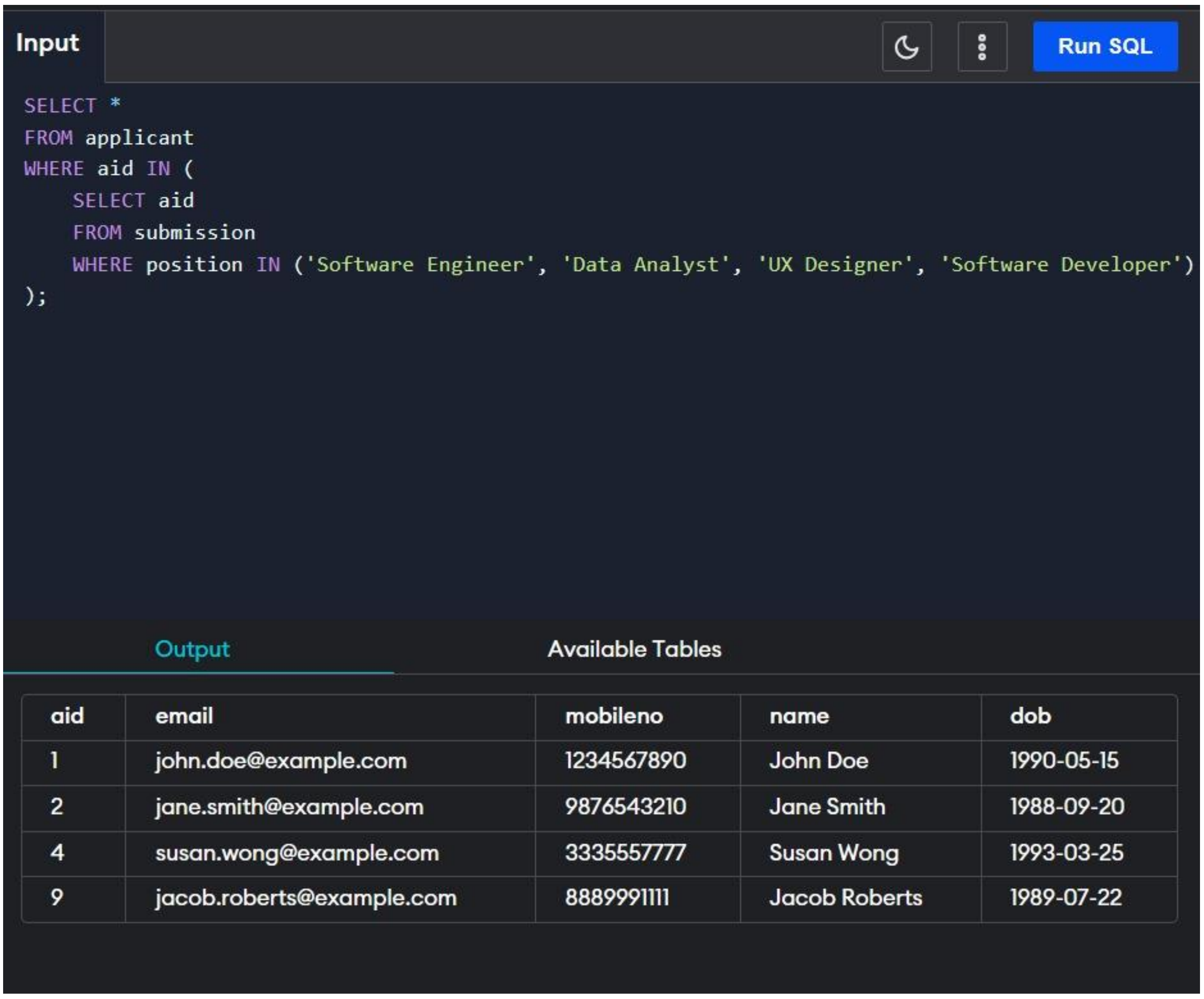
WHERE submissiondate > (SELECT submissiondate FROM submission s2 WHERE s2.aid = s1.aid - 1);



```
SELECT *
FROM submission s1
WHERE submissiondate > (SELECT submissiondate FROM submission s2 WHERE s2.aid = s1.aid - 1);
```

| sid | aid | email | name | position | submissiondate |
|---|---|---|---|---|---|
|  | 2 | jane.smith@example.com | Jane Smith | Data Analyst | 2024-04-21 |
|  | 3 | michael.johnson@example.com | Michael Johnson | Project Manager | 2024-04-22 |
|  | 4 | susan.wong@example.com | Susan Wong | UX Designer | 2024-04-23 |
|  | 5 | david.brown@example.com | David Brown | Marketing Specialist | 2024-04-24 |
|  | 6 | emily.white@example.com | Emily White | HR Coordinator | 2024-04-25 |
|  | 7 | william.clark@example.com | William Clark | Financial Analyst | 2024-04-26 |
|  | 8 | olivia.thomas@example.com | Olivia Thomas | Sales Manager | 2024-04-27 |

```
-- Finding applicants who have submitted for technical positions

SELECT *

FROM applicant

WHERE aid IN (

    SELECT aid

    FROM submission

    WHERE position IN ('Software Engineer', 'Data Analyst', 'UX Designer', 'Software Developer')

);
```



```
SELECT *
FROM applicant
WHERE aid IN (
    SELECT aid
    FROM submission
    WHERE position IN ('Software Engineer', 'Data Analyst', 'UX Designer', 'Software Developer')
);
```

| aid | email | mobileno | name | dob |
|---|---|---|---|---|
| 1 | john.doe@example.com | 1234567890 | John Doe | 1990-05-15 |
| 2 | jane.smith@example.com | 9876543210 | Jane Smith | 1988-09-20 |
| 4 | susan.wong@example.com | 3335557777 | Susan Wong | 1993-03-25 |
| 9 | jacob.roberts@example.com | 8889991111 | Jacob Roberts | 1989-07-22 |

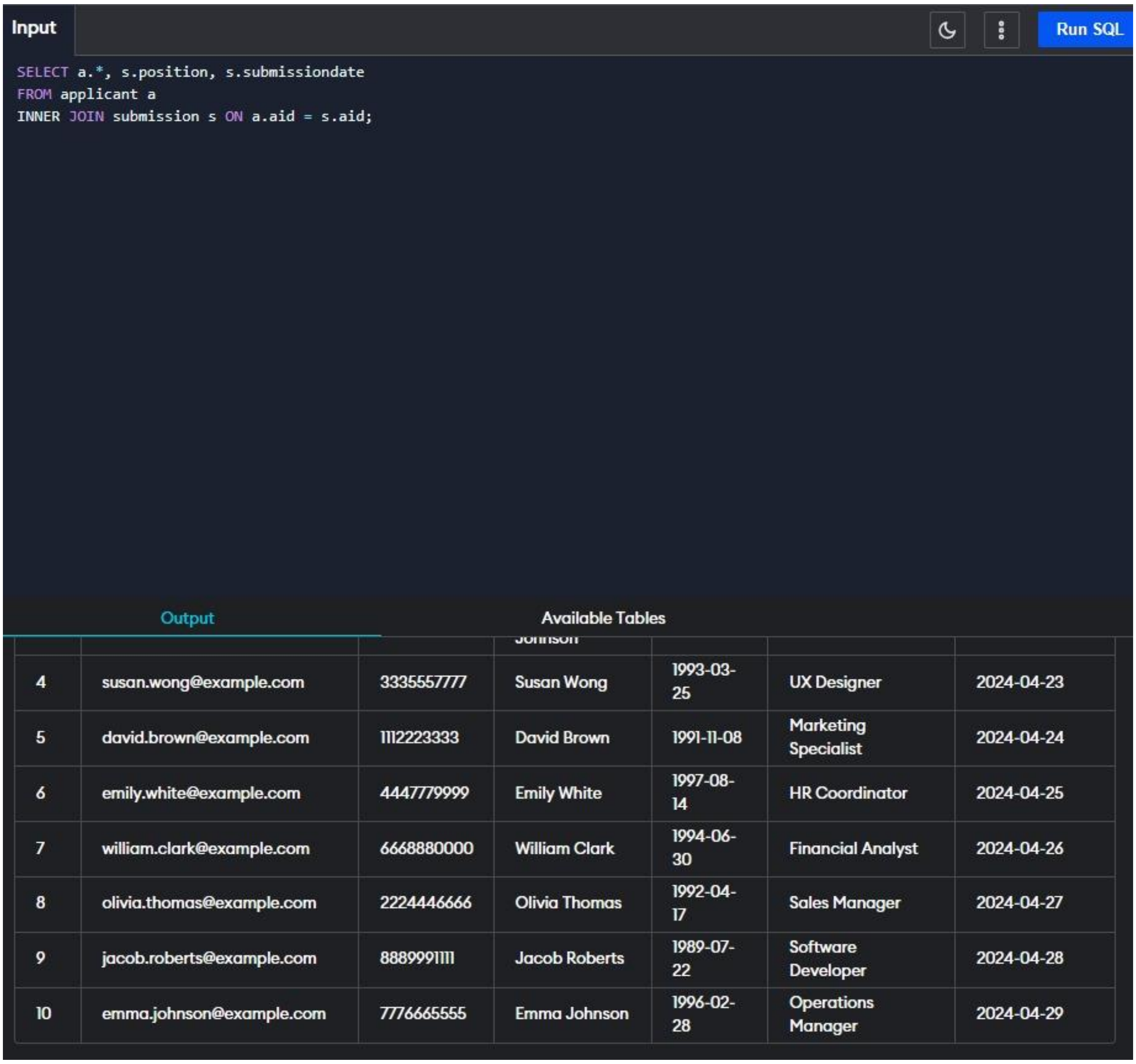
--joins

--Inner Join to Retrieve Applicant Information with Submission Details:

SELECT a.*, s.position, s.submissiondate

FROM applicant a

INNER JOIN submission s ON a.aid = s.aid;



```
SELECT a.*, s.position, s.submissiondate
FROM applicant a
INNER JOIN submission s ON a.aid = s.aid;
```

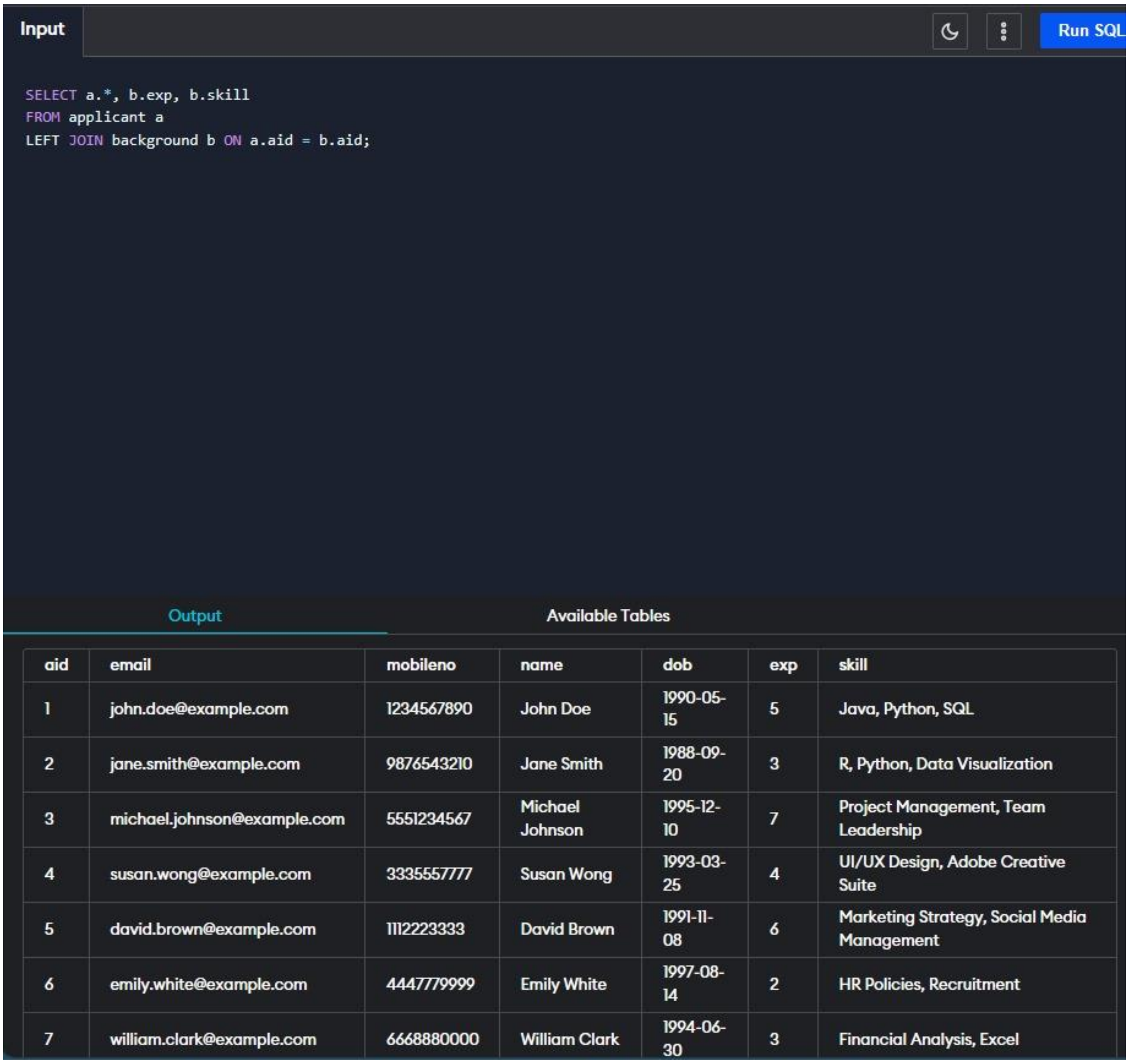
| | | | | | | |
|---|---|---|---|---|---|---|
| 4 | susan.wong@example.com | 3335557777 | Susan Wong | 1993-03-25 | UX Designer | 2024-04-23 |
| 5 | david.brown@example.com | 1112223333 | David Brown | 1991-11-08 | Marketing Specialist | 2024-04-24 |
| 6 | emily.white@example.com | 4447779999 | Emily White | 1997-08-14 | HR Coordinator | 2024-04-25 |
| 7 | william.clark@example.com | 6668880000 | William Clark | 1994-06-30 | Financial Analyst | 2024-04-26 |
| 8 | olivia.thomas@example.com | 2224446666 | Olivia Thomas | 1992-04-17 | Sales Manager | 2024-04-27 |
| 9 | jacob.roberts@example.com | 8889991111 | Jacob Roberts | 1989-07-22 | Software Developer | 2024-04-28 |
| 10 | emma.johnson@example.com | 7776665555 | Emma Johnson | 1996-02-28 | Operations Manager | 2024-04-29 |

--Left Join to Retrieve Applicant Information with Background Details:

SELECT a.*, b.exp, b.skill

FROM applicant a

LEFT JOIN background b ON a.aid = b.aid;



```
SELECT a.*, b.exp, b.skill
FROM applicant a
LEFT JOIN background b ON a.aid = b.aid;
```

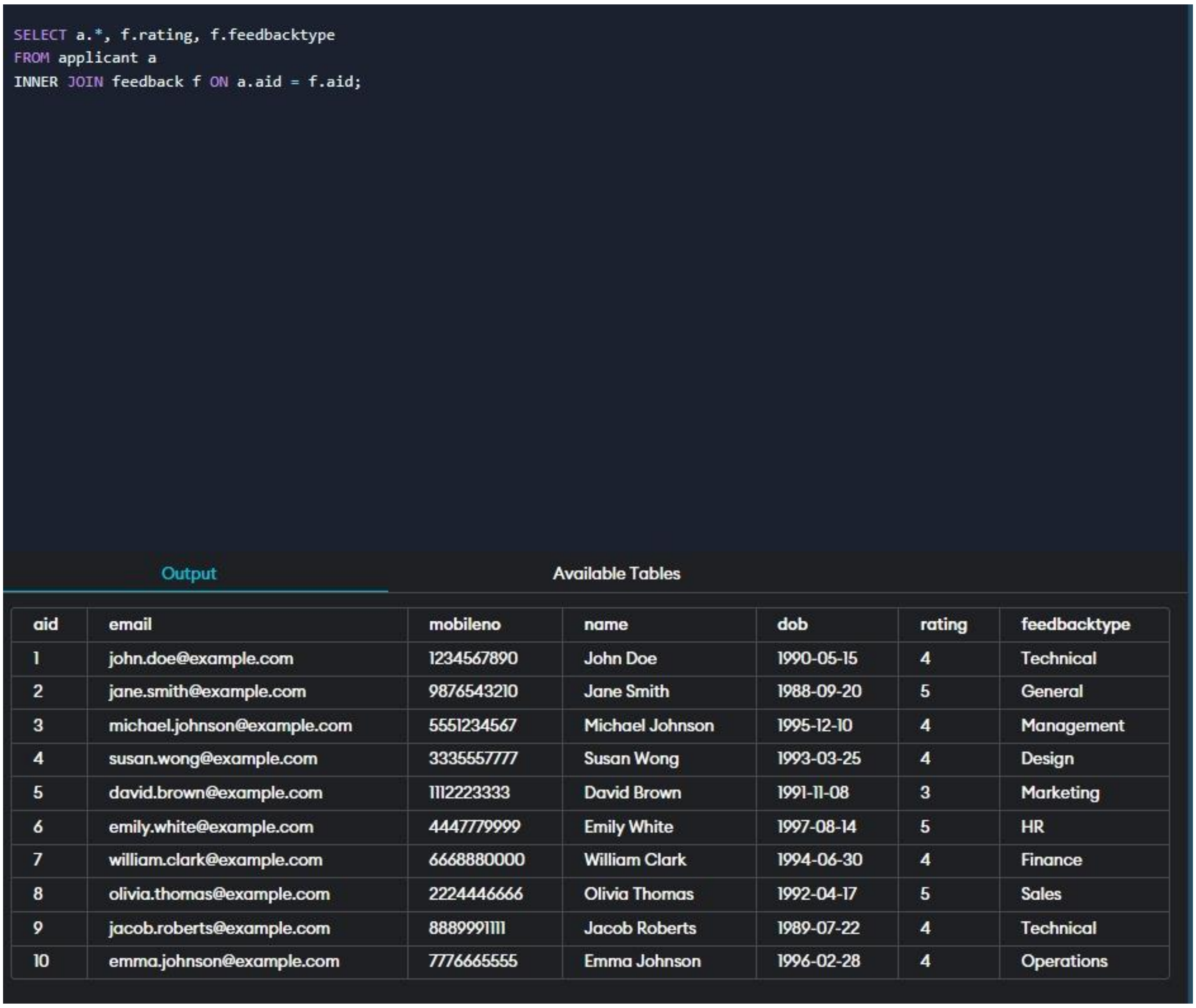
| aid | email | mobileno | name | dob | exp | skill |
|---|---|---|---|---|---|---|
| 1 | john.doe@example.com | 1234567890 | John Doe | 1990-05-15 | 5 | Java, Python, SQL |
| 2 | jane.smith@example.com | 9876543210 | Jane Smith | 1988-09-20 | 3 | R, Python, Data Visualization |
| 3 | michael.johnson@example.com | 5551234567 | Michael Johnson | 1995-12-10 | 7 | Project Management, Team Leadership |
| 4 | susan.wong@example.com | 3335557777 | Susan Wong | 1993-03-25 | 4 | UI/UX Design, Adobe Creative Suite |
| 5 | david.brown@example.com | 1112223333 | David Brown | 1991-11-08 | 6 | Marketing Strategy, Social Media Management |
| 6 | emily.white@example.com | 4447779999 | Emily White | 1997-08-14 | 2 | HR Policies, Recruitment |
| 7 | william.clark@example.com | 6668880000 | William Clark | 1994-06-30 | 3 | Financial Analysis, Excel |

--Inner Join to Retrieve Applicant Information with Feedback Details:

SELECT a.*, f.rating, f.feedbacktype

FROM applicant a

INNER JOIN feedback f ON a.aid = f.aid;



```
SELECT a.*, f.rating, f.feedbacktype
FROM applicant a
INNER JOIN feedback f ON a.aid = f.aid;
```

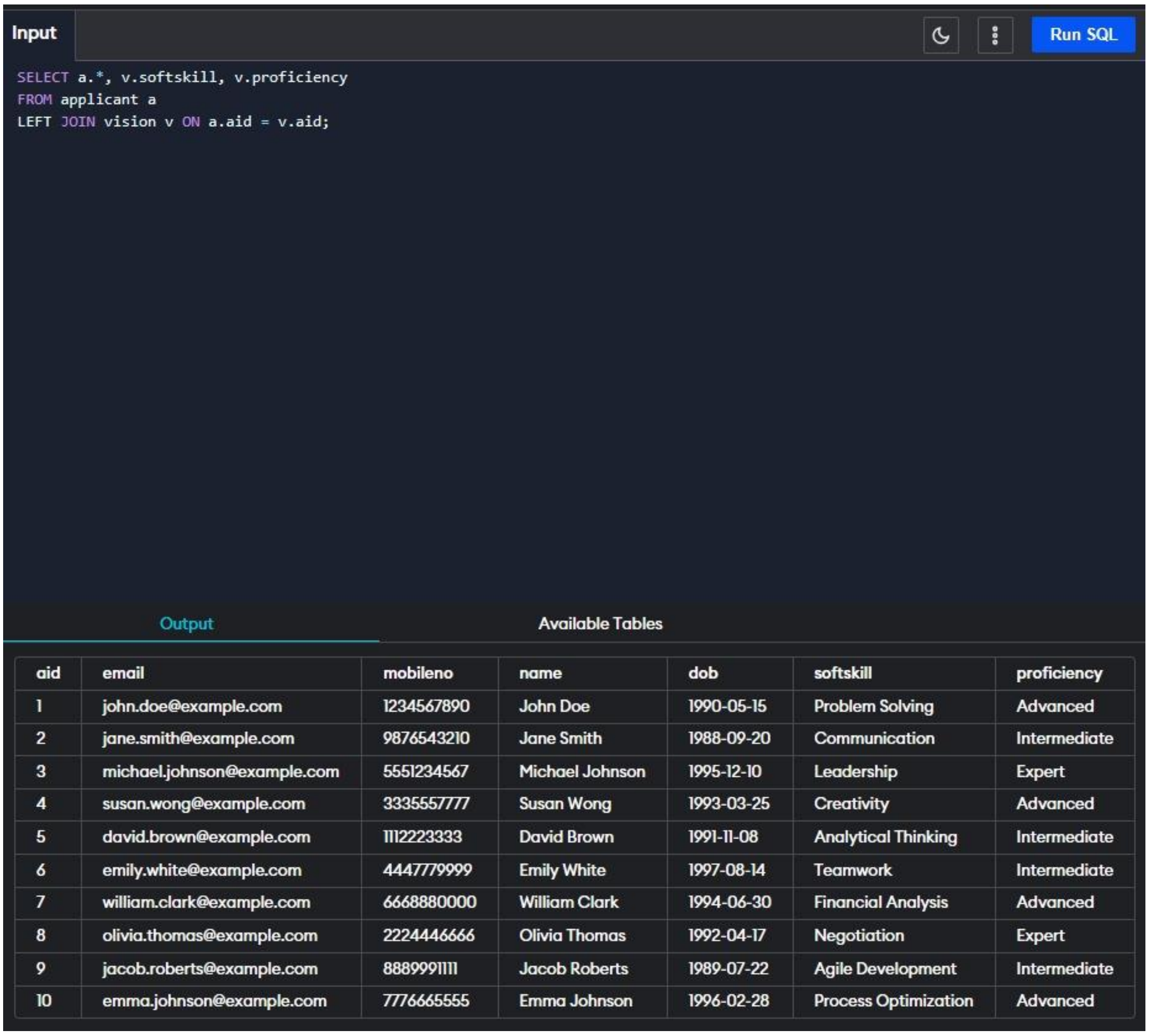
| aid | email | mobileno | name | dob | rating | feedbacktype |
|---|---|---|---|---|---|---|
| 1 | john.doe@example.com | 1234567890 | John Doe | 1990-05-15 | 4 | Technical |
| 2 | jane.smith@example.com | 9876543210 | Jane Smith | 1988-09-20 | 5 | General |
| 3 | michael.johnson@example.com | 5551234567 | Michael Johnson | 1995-12-10 | 4 | Management |
| 4 | susan.wong@example.com | 3335557777 | Susan Wong | 1993-03-25 | 4 | Design |
| 5 | david.brown@example.com | 1112223333 | David Brown | 1991-11-08 | 3 | Marketing |
| 6 | emily.white@example.com | 4447779999 | Emily White | 1997-08-14 | 5 | HR |
| 7 | william.clark@example.com | 6668880000 | William Clark | 1994-06-30 | 4 | Finance |
| 8 | olivia.thomas@example.com | 2224446666 | Olivia Thomas | 1992-04-17 | 5 | Sales |
| 9 | jacob.roberts@example.com | 8889991111 | Jacob Roberts | 1989-07-22 | 4 | Technical |
| 10 | emma.johnson@example.com | 7776665555 | Emma Johnson | 1996-02-28 | 4 | Operations |

--Left Join to Retrieve Applicant Information with Vision Details:

SELECT a.*, v.softskill, v.proficiency

FROM applicant a

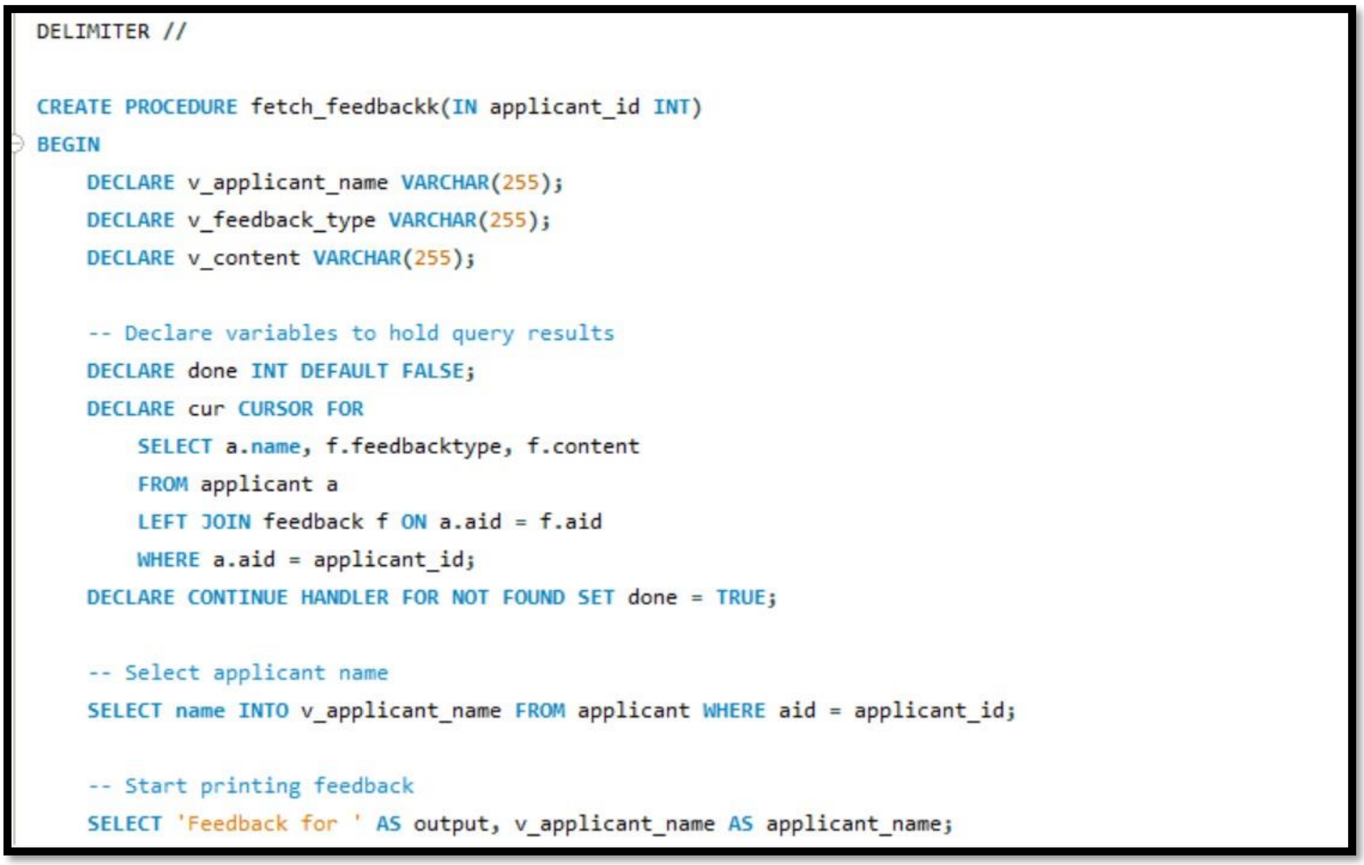
LEFT JOIN vision v ON a.aid = v.aid;



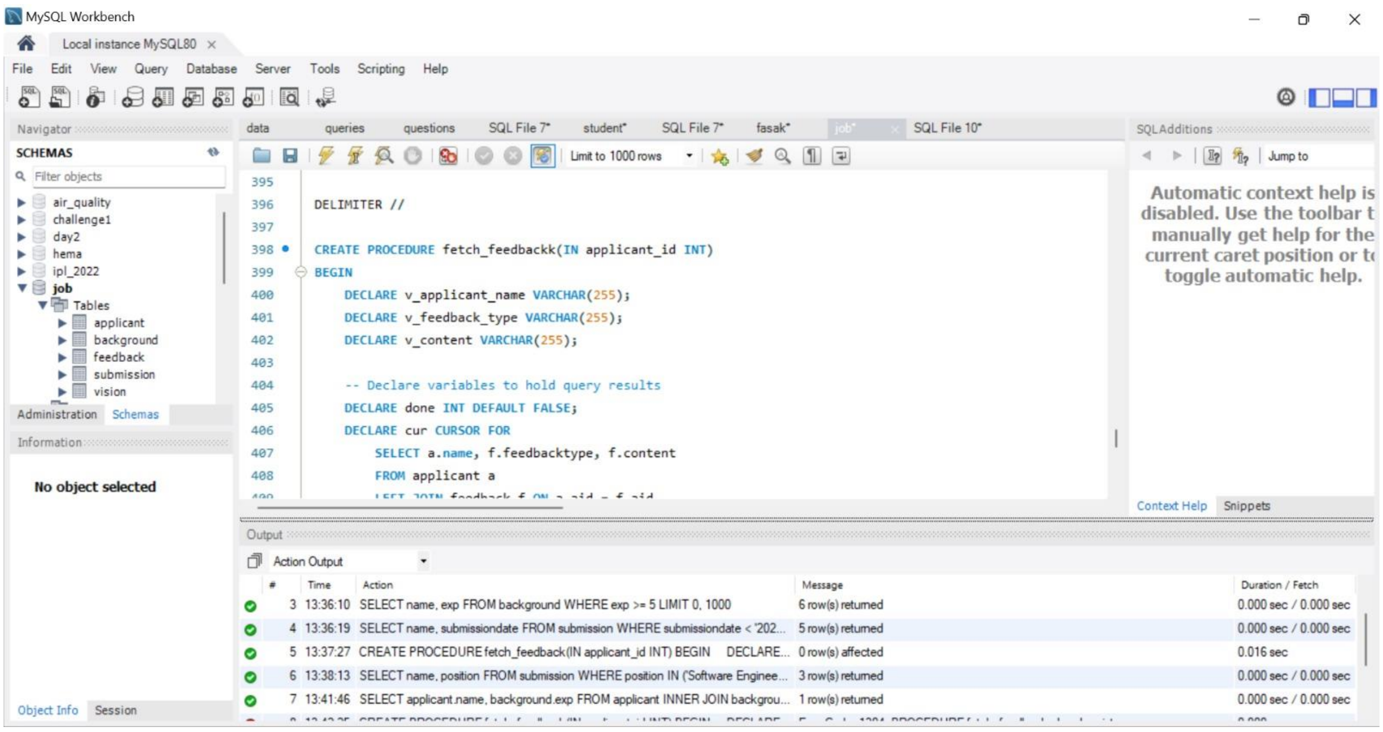
```
SELECT a.*, v.softskill, v.proficiency
FROM applicant a
LEFT JOIN vision v ON a.aid = v.aid;
```

| aid | email | mobileno | name | dob | softskill | proficiency |
|---|---|---|---|---|---|---|
| 1 | john.doe@example.com | 1234567890 | John Doe | 1990-05-15 | Problem Solving | Advanced |
| 2 | jane.smith@example.com | 9876543210 | Jane Smith | 1988-09-20 | Communication | Intermediate |
| 3 | michael.johnson@example.com | 5551234567 | Michael Johnson | 1995-12-10 | Leadership | Expert |
| 4 | susan.wong@example.com | 3335557777 | Susan Wong | 1993-03-25 | Creativity | Advanced |
| 5 | david.brown@example.com | 1112223333 | David Brown | 1991-11-08 | Analytical Thinking | Intermediate |
| 6 | emily.white@example.com | 4447779999 | Emily White | 1997-08-14 | Teamwork | Intermediate |
| 7 | william.clark@example.com | 6668880000 | William Clark | 1994-06-30 | Financial Analysis | Advanced |
| 8 | olivia.thomas@example.com | 2224446666 | Olivia Thomas | 1992-04-17 | Negotiation | Expert |
| 9 | jacob.roberts@example.com | 8889991111 | Jacob Roberts | 1989-07-22 | Agile Development | Intermediate |
| 10 | emma.johnson@example.com | 7776665555 | Emma Johnson | 1996-02-28 | Process Optimization | Advanced |

## PL/SQL:



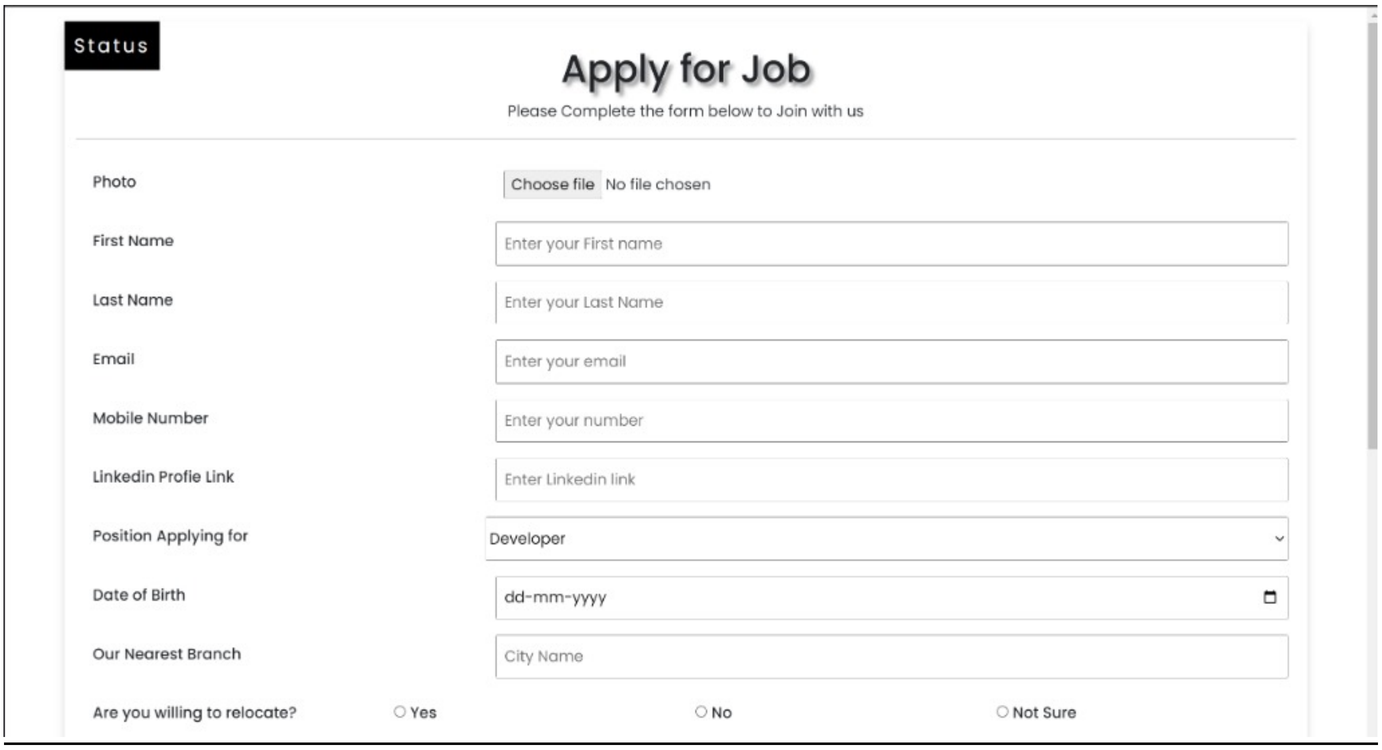
```sql
DELIMITER //

CREATE PROCEDURE fetch_feedbackk(IN applicant_id INT)
BEGIN
    DECLARE v_applicant_name VARCHAR(255);
    DECLARE v_feedback_type VARCHAR(255);
    DECLARE v_content VARCHAR(255);

    -- Declare variables to hold query results
    DECLARE done INT DEFAULT FALSE;
    DECLARE cur CURSOR FOR
        SELECT a.name, f.feedbacktype, f.content
        FROM applicant a
        LEFT JOIN feedback f ON a.aid = f.aid
        WHERE a.aid = applicant_id;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    -- Select applicant name
    SELECT name INTO v_applicant_name FROM applicant WHERE aid = applicant_id;

    -- Start printing feedback
    SELECT 'Feedback for ' AS output, v_applicant_name AS applicant_name;
```



```sql
DELIMITER //

CREATE PROCEDURE fetch_feedbackk(IN applicant_id INT)
BEGIN
    DECLARE v_applicant_name VARCHAR(255);
    DECLARE v_feedback_type VARCHAR(255);
    DECLARE v_content VARCHAR(255);

    -- Declare variables to hold query results
    DECLARE done INT DEFAULT FALSE;
    DECLARE cur CURSOR FOR
        SELECT a.name, f.feedbacktype, f.content
        FROM applicant a
```

```
#  Time      Action                                                                  Message            Duration / Fetch
3  13:36:10  SELECT name, exp FROM background WHERE exp >= 5 LIMIT 0, 1000           6 row(s) returned  0.000 sec / 0.000 sec
4  13:36:19  SELECT name, submissiondate FROM submission WHERE submissiondate < '202... 5 row(s) returned  0.000 sec / 0.000 sec
5  13:37:27  CREATE PROCEDURE fetch_feedback(IN applicant_id INT) BEGIN  DECLARE...   0 row(s) affected  0.016 sec
6  13:38:13  SELECT name, position FROM submission WHERE position IN ('Software Enginee... 3 row(s) returned  0.000 sec / 0.000 sec
7  13:41:46  SELECT applicant.name, background.exp FROM applicant INNER JOIN backgrou... 1 row(s) returned  0.000 sec / 0.000 sec
```

## FRONTEND:

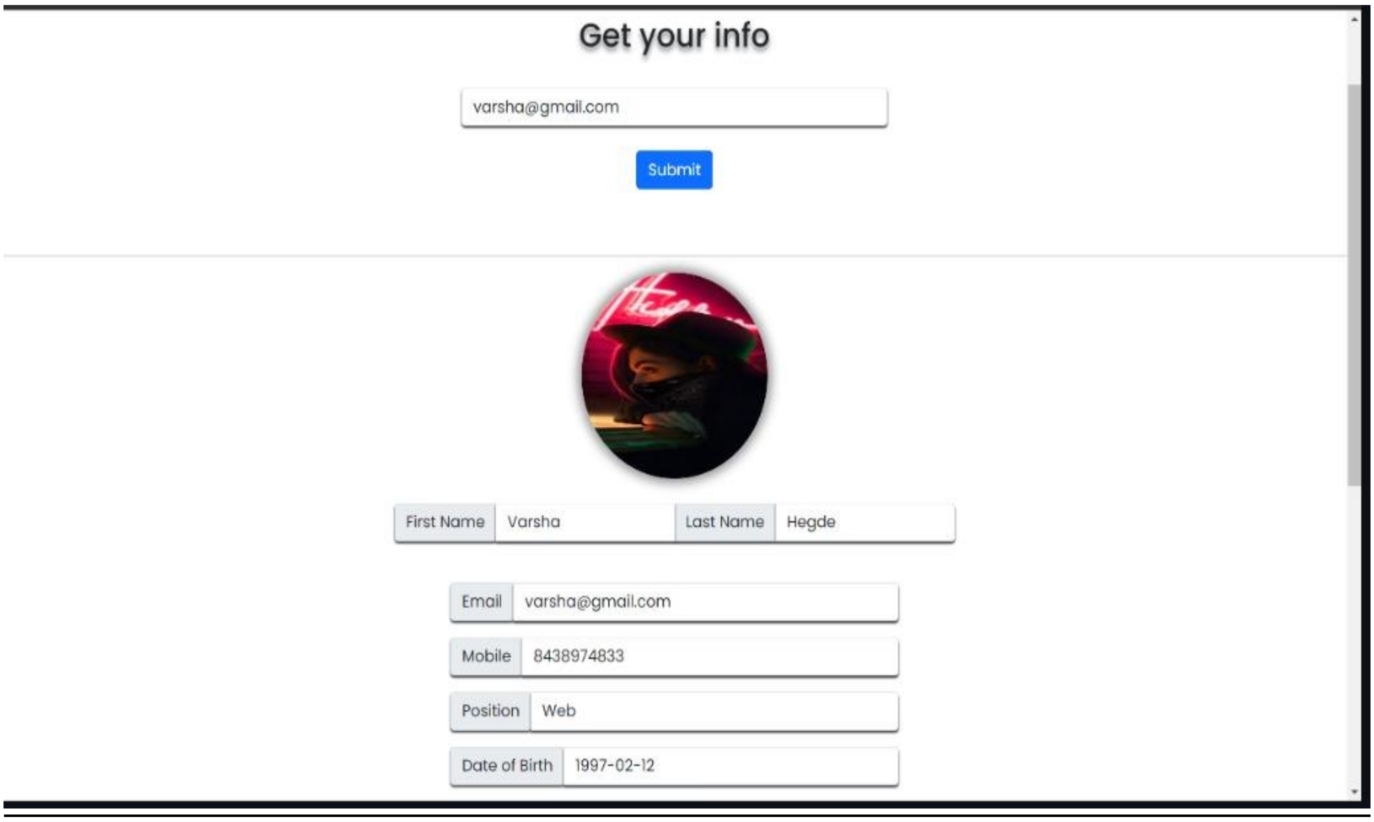


Status

Apply for Job

Please Complete the form below to Join with us

| | |
|---|---|
| Photo | Choose file No file chosen |
| First Name | Enter your First name |
| Last Name | Enter your Last Name |
| Email | Enter your email |
| Mobile Number | Enter your number |
| Linkedin Profie Link | Enter Linkedin link |
| Position Applying for | Developer |
| Date of Birth | dd-mm-yyyy |
| Our Nearest Branch | City Name |
| Are you willing to relocate? | ○ Yes ○ No ○ Not Sure |



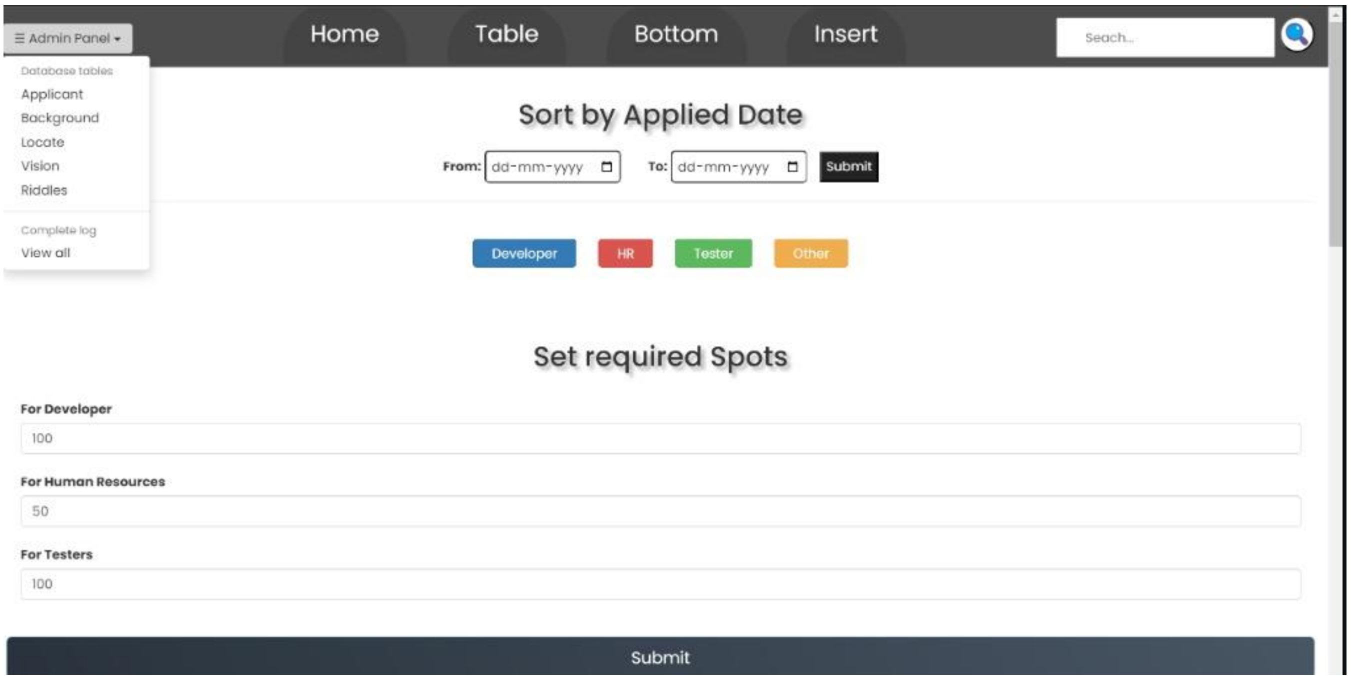
Get your info

varsha@gmail.com

Submit

| | | | |
|---|---|---|---|
| First Name | Varsha | Last Name | Hegde |

| | |
|---|---|
| Email | varsha@gmail.com |
| Mobile | 8438974833 |
| Position | Web |
| Date of Birth | 1997-02-12 |



Sort by Applied Date

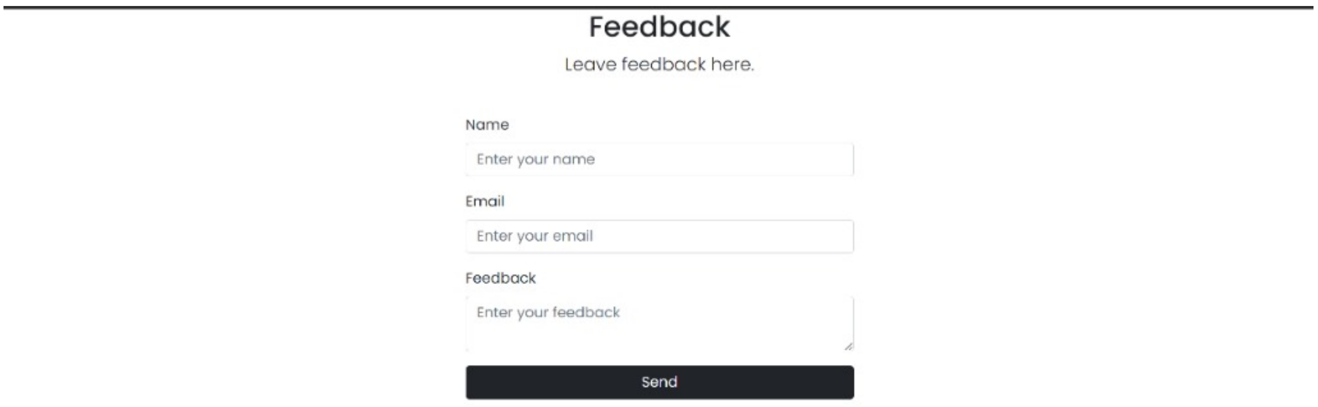
From: dd-mm-yyyy To: dd-mm-yyyy Submit

Set required Spots

For Developer
100

For Human Resources
50

For Testers
100


Feedback

Leave feedback here.

Name
Enter your name

Email
Enter your email

Feedback
Enter your feedback

## **CONCLUSION:**

In conclusion, the implementation of a Job Management System (JMS) using queries within a Database Management System (DBMS) offers a powerful solution for organizations to efficiently manage their resources, tasks, and projects. Through the utilization of structured query language (SQL) and the inherent capabilities of the DBMS, our project has demonstrated the potential to streamline job allocation, enhance productivity, and optimize resource utilization.

By effectively leveraging queries, we have designed a system that enables dynamic task allocation, real-time monitoring, and flexible reporting, empowering stakeholders with timely insights and actionable data. The scalability and maintainability of our database schema ensure that the Job Management System can adapt to evolving business needs and accommodate future growth.

Furthermore, the user-friendly interfaces integrated with query-driven functionalities enhance usability and promote adoption across all levels of the organization. With a focus on efficiency, transparency, and data-driven decision-making, our Job Management System facilitates improved organizational performance and fosters a culture of productivity and collaboration.

In essence, the integration of DBMS with queries to develop a Job Management System represents a significant step towards achieving operational excellence and driving business success in today's competitive landscape. As organizations continue to evolve, the importance of leveraging technology to streamline processes and optimize resource utilization cannot be overstated, and our project serves as a testament to the transformative potential of DBMS-powered job management solutions.

## REFERENCES:

https://www.w3schools.com/sql/sql_unique.asp

https://www.w3schools.com/sql/sql_join.asp

https://www.tutorialspoint.com/plsql/index.htm

https://www.tutorialspoint.com/plsql/plsql_exceptions.htm

https://www.geeksforgeeks.org/sql-ddl-dql-dml-dcl-tcl-commands/

https://www.geeksforgeeks.org/sql-trigger-student-database/

https://www.w3schools.com/sql/sql_view.asp

https://docs.microsoft.com/en-us/sql/sql-server/?view=sql-server-ver15