

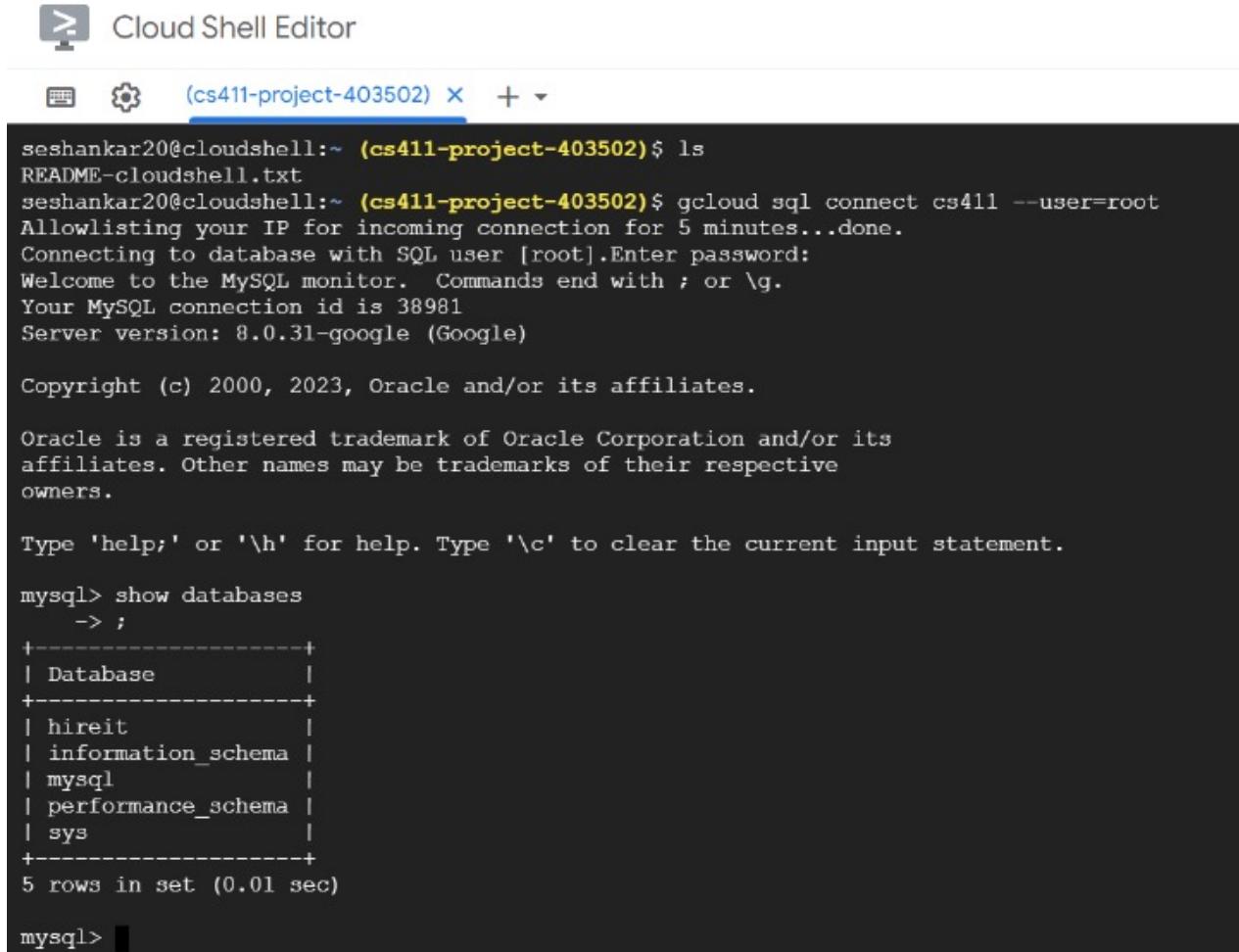
HireIt

Stage 3: Database Implementation and Indexing

Team No: 081

Team Name: ACID

GCP Connection Details:



The screenshot shows a Cloud Shell Editor window. At the top, there's a toolbar with icons for file operations and settings. Below the toolbar, the title bar displays '(cs411-project-403502)'. The main area is a terminal window showing the following MySQL session:

```
seshankar20@cloudshell:~ (cs411-project-403502)$ ls
README-cloudshell.txt

seshankar20@cloudshell:~ (cs411-project-403502)$ gcloud sql connect cs411 --user=root
Allowlisting your IP for incoming connection for 5 minutes...done.
Connecting to database with SQL user [root].Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 38981
Server version: 8.0.31-google (Google)

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases
    -> ;
+-----+
| Database      |
+-----+
| hireit        |
| information_schema |
| mysql          |
| performance_schema |
| sys            |
+-----+
5 rows in set (0.01 sec)

mysql>
```

DDL Commands:

1. For Student table

```
CREATE TABLE IF NOT EXISTS Student (
    student_id INT NOT NULL,
    first_name VARCHAR(255) NOT NULL,
    last_name VARCHAR(255) NOT NULL,
    email VARCHAR(255) NOT NULL,
    gender VARCHAR(1) NOT NULL,
    dob DATE NOT NULL,
    age INT NOT NULL,
    university_name VARCHAR(255) NOT NULL,
    degree VARCHAR(255) NOT NULL,
    gpa DECIMAL(5, 2) NOT NULL,
    grad_date INT NOT NULL,
    pwd VARCHAR(20) NOT NULL,
    PRIMARY KEY (student_id)
);
```

The screenshot shows a MySQL command-line interface. The command entered is `SELECT COUNT(*) FROM Student;`. The result grid shows one row with the value `1000` under the `COUNT(*)` column.

COUNT(*)
1000

2. For Company table

```
CREATE TABLE IF NOT EXISTS Company (
    company_id INT NOT NULL,
    company_name VARCHAR(255) NOT NULL,
    hq_location VARCHAR(255) NOT NULL,
    sector VARCHAR(255) NOT NULL,
    PRIMARY KEY (company_id)
);
```

```
23 •   SELECT COUNT(*) FROM Company;
24
100%   29:23 |
```

Result Grid Filter Rows: Search

COUNT(*)
5807

3. For Recruiter table

```
CREATE TABLE IF NOT EXISTS Recruiter(
    recruiter_id INT NOT NULL,
    first_name VARCHAR(255) NOT NULL,
    last_name VARCHAR(255) NOT NULL,
    email VARCHAR(255) NOT NULL,
    pwd VARCHAR(20) NOT NULL,
    company_id INT NOT NULL,
        FOREIGN KEY (company_id) REFERENCES Company(company_id) ON DELETE
CASCADE ON UPDATE CASCADE,
    PRIMARY KEY (recruiter_id)
);
```

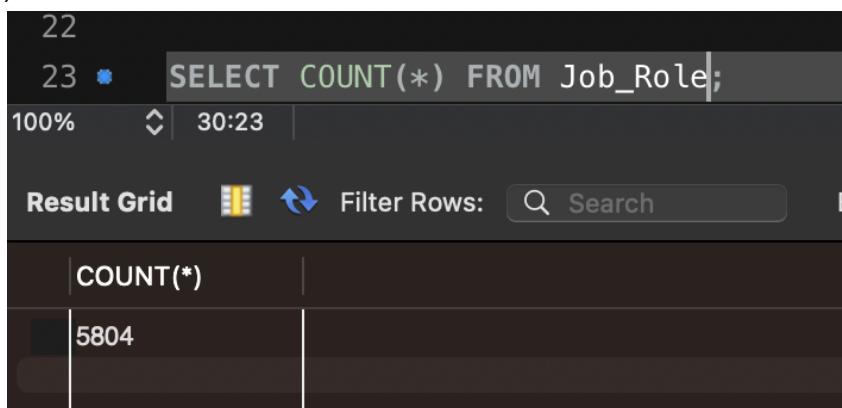
```
23 •   SELECT COUNT(*) FROM Recruiter;
24
100%   7:23 |
```

Result Grid Filter Rows: Search

COUNT(*)
5807

4. For Job_Role table

```
CREATE TABLE IF NOT EXISTS Job_Role(
    job_id INT NOT NULL,
    job_title VARCHAR(255) NOT NULL,
    salary INT NOT NULL,
    location VARCHAR(255) NOT NULL,
    job_type VARCHAR(255) NOT NULL,
    company_id INT NOT NULL,
        FOREIGN KEY (company_id) REFERENCES Company(company_id) ON DELETE CASCADE ON UPDATE CASCADE,
    PRIMARY KEY (job_id)
);
```



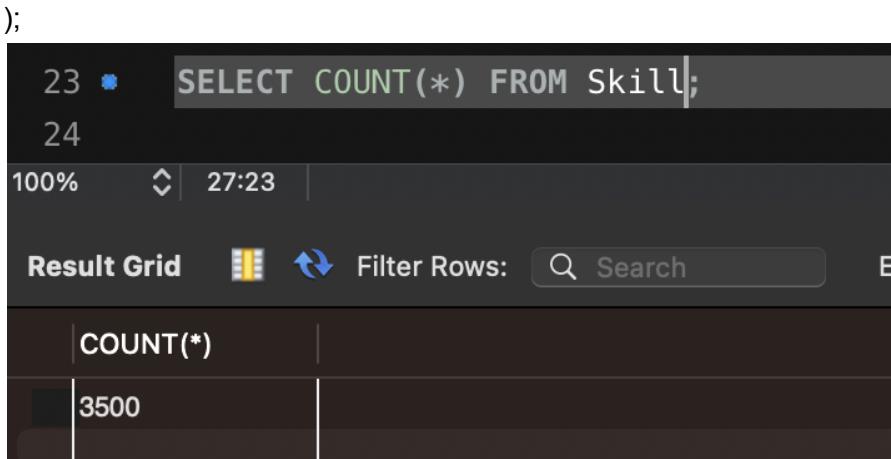
22
23 • **SELECT COUNT(*) FROM Job_Role;**
100% 30:23 |

Result Grid Filter Rows: Search E

| COUNT(*) |
| 5804 |

5. For Skill table

```
CREATE TABLE IF NOT EXISTS Skill(
    skill_id INT NOT NULL,
    skill_name VARCHAR(255) NOT NULL,
    PRIMARY KEY (skill_id)
);
```



23 • **SELECT COUNT(*) FROM Skill;**
24
100% 27:23 |

Result Grid Filter Rows: Search E

| COUNT(*) |
| 3500 |

6. For Applies table

```
CREATE TABLE IF NOT EXISTS Applies (
    status VARCHAR(255) NOT NULL,
    student_id INT NOT NULL,
    job_id INT NOT NULL,
    FOREIGN KEY (student_id) REFERENCES Student(student_id) ON DELETE CASCADE ON
UPDATE CASCADE,
    FOREIGN KEY (job_id) REFERENCES Job_Role(job_id) ON DELETE CASCADE ON
UPDATE CASCADE,
    PRIMARY KEY (student_id, job_id)
);
```

The screenshot shows a MySQL Workbench interface. At the top, there is a command line with the number 23 followed by a blue dot, indicating the current query. The query itself is `SELECT COUNT(*) FROM Applies;`. Below the command line, the status bar shows "100%" completion and a time of "29:23". Underneath the command line, there is a toolbar with "Result Grid" and other icons. The main area displays the results of the query in a grid format:

COUNT(*)
3005

7. For Worked table

```
CREATE TABLE IF NOT EXISTS Worked (
    company_id INT NOT NULL,
    student_id INT NOT NULL,
    work_exp_id INT NOT NULL,
    role VARCHAR(255) NOT NULL,
    years_of_exp INT NOT NULL,
    job_type VARCHAR(255) NOT NULL,
    FOREIGN KEY (company_id) REFERENCES Company(company_id) ON DELETE
CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (student_id) REFERENCES Student(student_id) ON DELETE CASCADE ON
UPDATE CASCADE,
    PRIMARY KEY (work_exp_id)
);
```

```
22
23 *   SELECT COUNT(*) FROM Worked;
100%   28:23 |
```

Result Grid Filter Rows: Search

COUNT(*)
998

8. For Requires table

```
CREATE TABLE IF NOT EXISTS Requires(
    skill_id INT NOT NULL,
    job_id INT NOT NULL,
    FOREIGN KEY (skill_id) REFERENCES Skill(skill_id) ON DELETE CASCADE ON UPDATE
CASCADE,
    FOREIGN KEY (job_id) REFERENCES Job_Role(job_id) ON DELETE CASCADE ON
UPDATE CASCADE,
    PRIMARY KEY(skill_id, job_id)
);
```

```
22
23 *   SELECT COUNT(*) FROM Requires;
24
100%   30:23 |
```

Result Grid Filter Rows: Search

COUNT(*)
2991

9. For Owns table

```
CREATE TABLE IF NOT EXISTS Owns (
    student_id INT NOT NULL,
    skill_id INT NOT NULL,
    FOREIGN KEY (student_id) REFERENCES Student(student_id) ON DELETE CASCADE ON UPDATE
CASCADE,
    FOREIGN KEY (skill_id) REFERENCES Skill(skill_id) ON DELETE CASCADE ON UPDATE
CASCADE,
    PRIMARY KEY(student_id, skill_id)
);
```

The screenshot shows a MySQL Workbench interface. The query window contains the following code:

```
23 • SELECT COUNT(*) FROM Owns;
```

The status bar indicates 100% completion and a duration of 26:23. Below the query window is a "Result Grid" tab, which displays the following data:

COUNT(*)
3000

Advanced Queries:

Query 1:

This query fetches the number of **Full Time** job roles posted by **Microsoft** with **salary>50000**. This query uses **Join** and **Aggregation**.

```
SELECT C.company_id, C.company_name, COUNT(*)
FROM Company C
JOIN Job_Role J ON J.company_id = C.company_id
WHERE C.company_id=9 and J.salary>50000 and J.job_type="Full Time"
GROUP BY C.company_id;
```

Output: This query returns only 1 row

The screenshot shows a MySQL Workbench result grid with the following data:

company_id	company_name	COUNT(*)
9	microsoft	6

Query 2:

This query uses **JOIN** of multiple relations and Aggregation via **GROUP BY**

```
SELECT J.job_title, St.university_name, AVG(J.salary) as avg_salary
FROM Job_Role J
JOIN Applies A ON A.job_id = J.job_id
JOIN Student St ON A.student_id = St.student_id
```

```

WHERE A.status = 'Accepted'
GROUP BY J.job_title, St.university_name
ORDER BY avg_salary DESC
LIMIT 15;

```

Output (with limit 15):

Query 1			
Result Grid			
<input type="checkbox"/> Filter Rows: <input type="text" value="Search"/> Export: <input type="button" value="CSV"/> Fetch rows: <input type="button" value="All"/>			
job_title	university_name	avg_salary	
Finance and Administration Director	Hawaii Pacific University	99939.0000	
IT Specialist	Mt San Antonio College	99917.0000	
Administrative Assistant	Flair Beauty College	99892.0000	
Chief of Party	Orleans Technical College	99825.0000	
Monitoring and Evaluation Officer	Pima Medical Institution-San Antonio	99703.0000	
Customs Clearance Officer	National American University-Albuquerque West	99646.0000	
Manager	Tri-County Technical College	99583.0000	
Logistics Driver	Saint Mary'S University Of Minnesota	99456.0000	
HR Specialist	Sierra College Of Beauty	99275.0000	
Software Developer	University Of Puerto Rico-Ponce	99258.0000	
Administrative Officer	Community College Of Baltimore County	99231.0000	
JSP/Java Developer for Lycos Bill...	Eastwick College-Ramsey	99214.0000	
Health and Safety Manager	Seton Hall University	99187.0000	
Instructor of Principles of Accounting	Altierus Career College-Tampa	99150.0000	
Programmer/ Developer	Fulton-Montgomery Community College	99064.0000	

Indexing Analysis:

Query 1:

```

SELECT C.company_id, C.company_name, COUNT(*)
FROM Company C
JOIN Job_Role J ON J.company_id = C.company_id
WHERE C.company_id=9 and J.salary>50000 and J.job_type="Full Time"
GROUP BY C.company_id;

```

Default index:

SHOW INDEX FROM Job_Role;

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
Job_Role	0	PRIMARY	1	job_id	A	6116	NULL	NULL		BTREE			YES	NULL
Job_Role	1	company_id	1	company_id	A	952	NULL	NULL		BTREE			YES	NULL

EXPLAIN ANALYZE execution time: 0.026 sec

```

18:53:31      EXPLAIN ANALYZE SELECT C.company_id, C.company_name, COUNT(*)
FROM Company C JOIN Job_Role J ON J.company_id = C.company_id WHERE
C.company_id=9 and J.salary>50000 and J.job_type="Full Time" GROUP BY C.company_id
1 row(s) returned    0.026 sec / 0.0000091 sec

```

EXPLAIN ANALYZE output:

```
'-> Group aggregate: count(0) (cost=1.54 rows=0.2) (actual time=0.051..0.051 rows=1 loops=1)
   -> Filter: ((J.job_type = 'Full Time') and (J.salary > 50000)) (cost=1.52 rows=0.2)
   (actual time=0.042..0.048 rows=6 loops=1)
      -> Index lookup on J using company_id
      (company_id=9) (cost=1.52 rows=6) (actual time=0.036..0.040 rows=6 loops=1)'
```

The screenshot shows a database interface with a timeline of actions from 18:47:15 to 18:53:31. The EXPLAIN ANALYZE output is displayed at the top, detailing the execution plan with cost, rows, and actual time. Below it, a table lists each action with its timestamp, SQL query, response, and duration/fetch time.

Action	Time	SQL Query	Response	Duration / Fetch Time
use hireit	18:47:15	SELECT ...	0 row(s) affected	0.020 sec
SELECT C.company_id, C.company_name, COUNT(*) from Company C JOIN...	18:47:20		1000 row(s) returned	0.034 sec / 0.020 sec
SELECT C.company_id, C.company_name, COUNT(*) FROM Company C JOIN Job_Role...	18:50:39		325 row(s) returned	0.035 sec / 0.00004...
SELECT C.company_id, C.company_name, COUNT(*) FROM Company C JOIN Job_Role...	18:52:18		Error Code: 1052. Column 'company_id' in where cla...	0.029 sec
SELECT C.company_id, C.company_name, COUNT(*) FROM Company C JOIN Job_Role...	18:52:32		1 row(s) returned	0.031 sec / 0.00008...
show index from Job_Role	18:53:12		2 row(s) returned	0.033 sec / 0.00000...
EXPLAIN ANALYZE SELECT C.company_id, C.company_name, COUNT(*) FROM Compan...	18:53:31		1 row(s) returned	0.026 sec / 0.000009...

Index 1: salary from Job_Role table

Table: Job_Role

Indexed column: salary

Index name: idx_salary

SHOW INDEX FROM Job_Role;

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
Job_Role	0	PRIMARY	1	job_id	A	6116	NULL	NULL		BTREE			YES	NULL
Job_Role	1	company_id	1	company_id	A	952	NULL	NULL		BTREE			YES	NULL
Job_Role	1	idx_salary	1	salary	A	5540	NULL	NULL		BTREE			YES	NULL

EXPLAIN ANALYZE execution time: 0.020 sec

```
18:55:30 EXPLAIN ANALYZE SELECT C.company_id, C.company_name, COUNT(*)
FROM Company C JOIN Job_Role J ON J.company_id = C.company_id WHERE
C.company_id=9 and J.salary>50000 and J.job_type="Full Time" GROUP BY C.company_id
1 row(s) returned 0.020 sec / 0.0000038 sec
```

EXPLAIN ANALYZE output:

```
'-> Group aggregate: count(0) (cost=1.59 rows=0.5) (actual time=0.046..0.046 rows=1
   -> Filter: ((J.job_type = 'Full Time') and (J.salary > 50000)) (cost=1.55 rows=0.5)
   (actual time=0.040..0.043 rows=6 loops=1)
      -> Index lookup on J using company_id
      (company_id=9) (cost=1.55 rows=6) (actual time=0.033..0.036 rows=6 loops=1)'
```

```

EXPLAIN:
-> Group aggregate: count(0) (cost=1.54 rows=0.2) (actual time=0.051..0.051 rows=1 loops=1)
-> Filter: ((J.job_type = 'Full Time') and (J.salary > 50000)) (cost=1.52 rows=0.2) (actual time=0.042..0.048 rows=6 loops=1)
-> Index lookup on J using company_id (company_id=9) (cost=1.52 rows=6) (actual time=0.036..0.040 rows=6 loops=1)

Result 5
Action Output | Time | Action | Response | Duration / Fetch Time
---|---|---|---|---
77 17:48:20 | SELECT J.job_id, J.job_title, J.job_type, J.salary, J.location, J.job_type,... | 9 row(s) returned | 0.015 sec / 0.000010...
78 18:47:15 | use hireit | 0 row(s) affected | 0.020 sec
79 18:47:20 | SELECT C.company_id, C.company_name, COUNT(*) from Company C JOIN... | 1000 row(s) returned | 0.034 sec / 0.020 sec
80 18:50:39 | SELECT C.company_id, C.company_name, COUNT(*) FROM Company C JOIN Job_Role... | 325 row(s) returned | 0.035 sec / 0.00004...
81 18:52:18 | SELECT C.company_id, C.company_name, COUNT(*) FROM Company C JOIN Job_Role... | Error Code: 1052. Column 'company_id' in where clause must have a non-zero length. | 0.029 sec
82 18:52:32 | SELECT C.company_id, C.company_name, COUNT(*) FROM Company C JOIN Job_Role... | 1 row(s) returned | 0.031 sec / 0.000008...
83 18:53:12 | show index from Job_Role | 2 row(s) returned | 0.033 sec / 0.00000...
84 18:53:31 | EXPLAIN ANALYZE SELECT C.company_id, C.company_name, COUNT(*) FROM Compan... | 1 row(s) returned | 0.026 sec / 0.000009...

```

Index 2: job_type from Job_Role table

Table: Job_Role

Indexed column: job_type

Index name: idx_type

SHOW INDEX FROM Job_Role;

Table	Non_unique	Key_name	Seq_in_Index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
Job_Role	0	PRIMARY	1	job_id	A	6116	NULL	NULL		BTREE			YES	HULL
Job_Role	1	company_id	1	company_id	A	952	NULL	NULL		BTREE			YES	HULL
Job_Role	1	idx_type	1	job_type	A	3	NULL	NULL		BTREE			YES	HULL

EXPLAIN ANALYZE execution time: 0.024 sec

18:58:11 EXPLAIN ANALYZE SELECT C.company_id, C.company_name, COUNT(*)
FROM Company C JOIN Job_Role J ON J.company_id = C.company_id WHERE
C.company_id=9 and J.salary>50000 and J.job_type="Full Time" GROUP BY C.company_id
1 row(s) returned **0.024 sec / 0.0000031 sec**

EXPLAIN ANALYZE output:

```

-> Group aggregate: count(0) (cost=1.62 rows=1) (actual time=0.040..0.040 rows=1 loops=1)
-> Filter: ((J.job_type = 'Full Time') and (J.salary > 50000)) (cost=1.56 rows=1) (actual time=0.033..0.037 rows=6 loops=1)
-> Index lookup on J using company_id (company_id=9) (cost=1.56 rows=6) (actual time=0.030..0.033 rows=6 loops=1)

```

```

EXPLAIN:
-> Group aggregate: count(0) (cost=1.59 rows=0.5) (actual time=0.046..0.046 rows=1 loops=1)
  -> Filter: ((J.job_type = 'Full Time') and (J.salary > 50000)) (cost=1.55 rows=0.5) (actual time=0.040..0.043 rows=6 loops=1)
    -> Index lookup on J using company_id (company_id=9) (cost=1.55 rows=6) (actual time=0.033..0.036 rows=6 loops=1)

Result 7
Action Output | Action | Response | Duration / Fetch Time |
-----|-----|-----|-----|
  60 | 18:50:39 | SELECT C.company_id, C.company_name, COUNT(*) FROM Company C JOIN Job_Role... | 323 row(s) returned | 0.003 sec / 0.000004... |
  81 | 18:52:18 | SELECT C.company_id, C.company_name, COUNT(*) FROM Company C JOIN Job_Role... | Error Code: 1052. Column 'company_id' in where clause is not in GROUP BY clause | 0.029 sec |
  82 | 18:52:32 | SELECT C.company_id, C.company_name, COUNT(*) FROM Company C JOIN Job_Role... | 1 row(s) returned | 0.031 sec / 0.000008... |
  83 | 18:53:12 | show index from Job_Role | 2 row(s) returned | 0.033 sec / 0.00000... |
  84 | 18:53:31 | EXPLAIN ANALYZE SELECT C.company_id, C.company_name, COUNT(*) FROM Compan... | 1 row(s) returned | 0.026 sec / 0.000009... |
  85 | 18:55:22 | CREATE INDEX idx_salary on Job_Role(salary) | 0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0 | 0.122 sec |
  86 | 18:55:25 | show index from Job_Role | 3 row(s) returned | 0.022 sec / 0.000013... |
  87 | 18:55:30 | EXPLAIN ANALYZE SELECT C.company_id, C.company_name, COUNT(*) FROM Compan... | 1 row(s) returned | 0.020 sec / 0.000003... |

```

Index 3: salary and job_type from Job_Role table

Table: Job_Role

Indexed columns: salary, job_type

Indices names: idx_salary, idx_type

For the 3rd type of indexing, we have combined the first two indices idx_type and idx_salary.

SHOW INDEX FROM Job_Role;

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
Job_Role	0	PRIMARY	1	job_id	A	6116	NULL	NULL		BTREE			YES	NULL
Job_Role	1	company_id	1	company_id	A	952	NULL	NULL		BTREE			YES	NULL
Job_Role	1	idx_type	1	job_type	A	3	NULL	NULL		BTREE			YES	NULL
Job_Role	1	idx_salary	1	salary	A	5540	NULL	NULL		BTREE			YES	NULL

EXPLAIN ANALYZE execution time: 0.019 sec

```

19:00:06      EXPLAIN ANALYZE SELECT C.company_id, C.company_name, COUNT(*)
FROM Company C JOIN Job_Role J ON J.company_id = C.company_id WHERE
C.company_id=9 and J.salary>50000 and J.job_type="Full Time" GROUP BY C.company_id
1 row(s) returned   0.019 sec / 0.0000038 sec

```

EXPLAIN ANALYZE output:

```

'-> Group aggregate: count(0) (cost=1.79 rows=1) (actual time=0.040..0.040 rows=1 loops=1)
-> Filter: ((J.job_type = 'Full Time') and (J.salary > 50000)) (cost=1.65 rows=1) (actual
time=0.031..0.037 rows=6 loops=1)
-> Index lookup on J using company_id (company_id=9) (cost=1.65 rows=6) (actual time=0.029..0.032 rows=6 loops=1)

```

The screenshot shows the MySQL Workbench interface. At the top, the Explain plan for a query is displayed, indicating a group aggregate count(0) with a cost of 1.62 rows=1, and a filter for job_type = 'Full Time' and salary > 50000 with a cost of 1.56 rows=6. Below this, the Action Output history shows various database operations from log entries 88 to 95, including dropping and creating indexes, running EXPLAIN ANALYZE, and showing index details.

Query 2:

```

SELECT J.job_title, St.university_name, AVG(J.salary) as avg_salary
FROM Job_Role J
JOIN Applies A ON A.job_id = J.job_id
JOIN Student St ON A.student_id = St.student_id
WHERE A.status = 'Accepted'
GROUP BY J.job_title, St.university_name
ORDER BY avg_salary DESC
LIMIT 15;
    
```

Default index:

SHOW INDEX FROM Job_Role;

The screenshot shows the results of the SHOW INDEX query for the Job_Role table. It displays two indexes: one primary key index (Job_Role 0) and one non-unique index (Job_Role 1) on the company_id column. The results grid shows the table name, index type, key name, sequence in index, column name, collation, cardinality, and sub-partition information. Below the results grid, the Action Output history shows the execution of the SHOW INDEX query.

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part
Job_Role	0	PRIMARY	1	job_id	A	6116	NULL
Job_Role	1	company_id	1	company_id	A	952	NULL

EXPLAIN ANALYZE execution time: 0.043 sec

The screenshot shows the MySQL Workbench interface with the 'PERFORMANCE' tab selected. In the central pane, the query window displays the following code:

```
10 • EXPLAIN ANALYZE SELECT J.job_title, St.university_name, AVG(J.salary) as avg_salary
11   FROM Job_Role J
12   JOIN Applies A ON A.job_id = J.job_id
13   JOIN Student St ON A.student_id = St.student_id
14 WHERE A.status = 'Accepted'
```

Below the code, the 'Result Grid' shows the EXPLAIN output:

EXPLAIN
>- Limit: 15 row(s) (actual time=15.558..15.561 rows=15 loops=1) -> Sort: avg_salary DESC, limit input to 15 row(s) per chunk (actual time=15.557..15.559 rows=15 loops=1)
>- Table scan on <temporary> (actual time=14.683..14.887 rows=1029 loops=1) -> Aggregate using temporary table (actual time=14.677..14.677 rows=1029 loops=1)
>- Nested loop inner join (cost=473.07 rows=277) (actual time=0.124..11.833 rows=1030 loops=1) -> Nested loop inner join (cost=376.09 rows=277) (actual time=0.114..9.478 rows=1030 loops=1) -> Filter: (A.`status` = 'Accepted') (cost=279.10 rows=277) (actual time=0.096..2.512 rows=1030 loops=1) -> Table scan on A (cost=279.10 rows=2771) (actual time=0.089..1.735 rows=3005 loops=1)
>- Single-row index lookup on St using PRIMARY (student_id=A.student_id) (cost=0.25 rows=1) (actual time=0.006..0.007 rows=1 loops=1030) -> Single-row index lookup on J using PRIMARY (job_id=A.job_id) (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=1030)

The 'Action Output' section shows the following table:

Action	Time	Response	Duration / Fetch Time
EXPLAIN ANALYZE SELECT J.job_title, St.university_name, AVG(J.salary) as avg_salary FROM Job_Role J JOIN Applies A ON A.job_id = J.job_id JOIN Student St ON A.student_id = St.student_id WHERE A.status = 'Accepted'	20:35:41	1 row(s) returned	0.043 sec / 0.00000...

At the bottom left, it says 'Query Completed'.

EXPLAIN ANALYZE output:

'-> Limit: 15 row(s) (actual time=15.558..15.561 rows=15 loops=1)
-> Sort: avg_salary DESC, limit input to 15 row(s) per chunk (actual time=15.557..15.559 rows=15 loops=1)
>- Table scan on <temporary> (actual time=14.683..14.887 rows=1029 loops=1) ->
Aggregate using temporary table (actual time=14.677..14.677 rows=1029 loops=1)
>- Nested loop inner join (cost=473.07 rows=277) (actual time=0.124..11.833 rows=1030 loops=1)
-> Nested loop inner join (cost=376.09 rows=277) (actual time=0.114..9.478 rows=1030 loops=1) ->
Filter: (A.`status` = 'Accepted') (cost=279.10 rows=277) (actual time=0.096..2.512 rows=1030 loops=1) ->
Table scan on A (cost=279.10 rows=2771) (actual time=0.089..1.735 rows=3005 loops=1)
>- Single-row index lookup on St using PRIMARY (student_id=A.student_id) (cost=0.25 rows=1) (actual time=0.006..0.007 rows=1 loops=1030) ->
Single-row index lookup on J using PRIMARY (job_id=A.job_id) (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=1030)

Index 1: salary from Job_Role table

Table: Job_Role

Indexed columns: salary

Indices names: idx_salary

SHOW INDEX FROM Job_Role;

The screenshot shows the MySQL Workbench interface with the 'Result Grid' tab selected. The results of the 'SHOW INDEX' query are displayed in a table with columns: Name, Non_unique, Key_name, Seq_in_Index, Column_name, Collation, Cardinality, Sub_part, Packed, Null, Index_type, Comment, Index_comment, Visible, and Expression. There are three indexes listed: a primary key index (job_id) and two BTREE indexes (company_id and salary). Below the results, the 'Action Output' section shows the history of the session, including the execution of the EXPLAIN ANALYZE query.

Name	Non_unique	Key_name	Seq_in_Index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
J...	0	PRIMARY	1	job_id	A	6116	HULL	HULL		BTREE			YES	HULL
J...	1	company_id	1	company_id	A	952	HULL	HULL		BTREE			YES	HULL
J...	1	idx_salary	1	salary	A	5540	HULL	HULL		BTREE			YES	HULL

Action Output:

Time	Action	Response	Duration / Fetch Time
9 14:34:46	explain analyze SELECT J.job_title, St.university_name, AVG(J.salary)...	1 row(s) returned	0.027 sec / 0.000006...
10 14:35:49	show index from Applies	3 row(s) returned	0.019 sec / 0.000015...
11 14:35:58	create index idx_salary on Job_Role(salary)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0.105 sec	
12 14:36:07	show index from Job_Role	3 row(s) returned	0.153 sec / 0.000038...

EXPLAIN ANALYZE execution time: 0.028 sec

The screenshot shows the MySQL Workbench interface with the 'Result Grid' tab selected. The results of the EXPLAIN ANALYZE query are displayed in a table with columns: Time, Action, Response, and Duration / Fetch Time. The output shows the execution plan for the SELECT query, including various stages such as explain, select, show index, drop index, and create index, along with their respective times and resource usage.

Time	Action	Response	Duration / Fetch Time
1 14:30:41	explain analyze SELECT J.job_title, St.university_name, AVG(J.salary)...	1 row(s) returned	0.033 sec / 0.00000...
2 14:32:02	SELECT J.job_title, St.university_name, AVG(J.salary) as avg_salary fr...	15 row(s) returned	0.027 sec / 0.000035...
3 14:32:39	show index from Applies	4 row(s) returned	0.020 sec / 0.00001...
4 14:32:45	drop index idx_status on Applies	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0.044 sec	
5 14:32:54	show index from Applies	3 row(s) returned	0.023 sec / 0.000013...
6 14:33:05	show index from Job_Role	3 row(s) returned	0.022 sec / 0.000043...
7 14:34:08	drop index idx_job_salary on Job_Role	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0.045 sec	
8 14:34:11	show index from Job_Role	2 row(s) returned	0.022 sec / 0.000016...
9 14:34:46	explain analyze SELECT J.job_title, St.university_name, AVG(J.salary)...	1 row(s) returned	0.027 sec / 0.000006...
10 14:35:49	show index from Applies	3 row(s) returned	0.019 sec / 0.000015...
11 14:35:58	create index idx_salary on Job_Role(salary)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0.105 sec	
12 14:36:07	show index from Job_Role	3 row(s) returned	0.153 sec / 0.000038...
13 14:36:25	explain analyze SELECT J.job_title, St.university_name, AVG(J.salary)...	1 row(s) returned	0.028 sec / 0.000010...

EXPLAIN ANALYZE output:

```
'-> Limit: 15 row(s) (actual time=6.427..6.429 rows=15 loops=1)
-> Sort: avg_salary DESC, limit input to 15 row(s) per chunk (actual time=6.426..6.427 rows=15 loops=1)
-> Table scan on <temporary> (actual time=5.923..6.083 rows=1029 loops=1)
-> Aggregate using temporary table (actual time=5.920..5.920 rows=1029 loops=1)
-> Nested loop inner join (cost=473.07 rows=277) (actual time=0.141..4.155 rows=1030 loops=1)
-> Nested loop inner join (cost=376.09 rows=277) (actual time=0.120..2.656 rows=1030 loops=1)
-> Filter: (A.`status` = 'Accepted') (cost=279.10 rows=277) (actual time=0.099..1.402 rows=1030 loops=1)
-> Table scan on A (cost=279.10 rows=2771) (actual time=0.092..0.948 rows=2997 loops=1)
-> Single-row index lookup on St using PRIMARY (student_id=A.student_id) (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=1030)
-> Single-row index lookup on J using PRIMARY (job_id=A.job_id) (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=1030)'
```

Index 2: job_title from Job_Role table

Table: Job_Role

Indexed columns: job_title

Indices names: idx_job_title

SHOW INDEX FROM Job_Role;

The screenshot shows the MySQL Workbench interface with the 'PERFORMANCE' tab selected. In the central pane, the results of the 'SHOW INDEX FROM Job_Role;' query are displayed in a grid. The grid includes columns for Table, Non_unique, Key_name, Seq_in_Index, Column_name, Collation, Cardinality, Sub_part, Packed, Null, Index_type, Comment, Index_comment, Visible, and Expression. The results show three indexes: PRIMARY (seq_in_index 1, column job_id), company_id (seq_in_index 1, column company_id), and idx_job_title (seq_in_index 1, column job_title). Below the grid, the 'Action Output' section shows the execution history of the three statements.

Table	Non_unique	Key_name	Seq_in_Index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
Job_Role	0	PRIMARY	1	job_id	A	6118	HOL	HOL		BTREE			YES	HOL
Job_Role	1	company_id	1	company_id	A	952	HOL	HOL		BTREE			YES	HOL
Job_Role	1	idx_job_title	1	job_title	A	3109	HOL	HOL		BTREE			YES	HOL

Action Output:

Time	Action	Response	Duration / Fetch Time
20:35:41	EXPLAIN ANALYZE SELECT J.job_title, St.university_name, AVG(J.salary) as avg_salary FROM Job_Role J JOIN Status St ON J.job_title = St.job_title WHERE St.status = 'Accepted' ORDER BY avg_salary DESC LIMIT 15;	1 row(s) returned	0.043 sec / 0.00000...
20:41:21	CREATE INDEX idx_job_title ON Job_Role(job_title)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.184 sec
20:41:26	SHOW INDEX FROM Job_Role	3 row(s) returned	0.023 sec / 0.000008...

EXPLAIN ANALYZE execution time: 0.027 sec

The screenshot shows the MySQL Workbench interface with the 'PERFORMANCE' tab selected. The central pane displays the detailed EXPLAIN ANALYZE output for the query. The output is a table with columns for Action, Response, and Duration / Fetch Time. The table lists 19 rows, each representing a step in the execution plan. The steps include explain analyze for the select statement, various table scans, index drops, and index creation. The final row shows the explain analyze for the original query, which took 0.027 seconds.

Action	Response	Duration / Fetch Time
explain analyze SELECT J.job_title, St.university_name, AVG(J.salary) as avg_salary FROM Job_Role J JOIN Status St ON J.job_title = St.job_title WHERE St.status = 'Accepted' ORDER BY avg_salary DESC LIMIT 15;	1 row(s) returned	0.033 sec / 0.00000...
SELECT J.job_title, St.university_name, AVG(J.salary) as avg_salary fr...	15 row(s) returned	0.027 sec / 0.000005...
show index from Applies	4 row(s) returned	0.020 sec / 0.00001...
drop index idx_status on Applies	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.044 sec
show index from Applies	3 row(s) returned	0.023 sec / 0.000013...
show index from Job_Role	3 row(s) returned	0.022 sec / 0.000043...
drop index idx_job_salary on Job_Role	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.045 sec
show index from Job_Role	2 row(s) returned	0.022 sec / 0.000016...
explain analyze SELECT J.job_title, St.university_name, AVG(J.salary) as avg_salary FROM Job_Role J JOIN Status St ON J.job_title = St.job_title WHERE St.status = 'Accepted' ORDER BY avg_salary DESC LIMIT 15;	1 row(s) returned	0.027 sec / 0.000008...
show index from Applies	3 row(s) returned	0.019 sec / 0.000015...
create index idx_salary on Job_Role(salary)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.105 sec
show index from Job_Role	3 row(s) returned	0.153 sec / 0.000038...
explain analyze SELECT J.job_title, St.university_name, AVG(J.salary) as avg_salary FROM Job_Role J JOIN Status St ON J.job_title = St.job_title WHERE St.status = 'Accepted' ORDER BY avg_salary DESC LIMIT 15;	1 row(s) returned	0.028 sec / 0.000010...
drop index idx_job_title on Job_Role	Error Code: 1091. Can't DROP 'idx_job_title'; check th...	0.028 sec
drop index idx_salary on Job_Role	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.045 sec
show index from Job_Role	2 row(s) returned	0.021 sec / 0.000018...
create index idx_job_title on Job_Role(job_title)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.195 sec
show index from Job_Role	3 row(s) returned	0.021 sec / 0.000010...
explain analyze SELECT J.job_title, St.university_name, AVG(J.salary) as avg_salary FROM Job_Role J JOIN Status St ON J.job_title = St.job_title WHERE St.status = 'Accepted' ORDER BY avg_salary DESC LIMIT 15;	1 row(s) returned	0.027 sec / 0.000007...

EXPLAIN ANALYZE output:

```
'-> Limit: 15 row(s) (actual time=7.536..7.539 rows=15 loops=1)
   -> Sort: avg_salary DESC,
limit input to 15 row(s) per chunk (actual time=7.536..7.537 rows=15 loops=1)
   -> Table scan on <temporary> (actual time=7.012..7.168 rows=1029 loops=1)
      -> Aggregate using temporary table (actual time=7.009..7.009 rows=1029 loops=1)
      -> Nested loop inner join (cost=473.07 rows=277) (actual time=0.155..5.007 rows=1030 loops=1)
-> Nested loop inner join (cost=376.09 rows=277) (actual time=0.144..3.192 rows=1030 loops=1)
   -> Filter: (A.`status` = 'Accepted') (cost=279.10 rows=277) (actual time=0.117..1.588 rows=1030 loops=1)
      -> Table scan on A (cost=279.10 rows=2771) (actual time=0.107..1.038 rows=2997 loops=1)
   -> Single-row index
```

lookup on St using PRIMARY (student_id=A.student_id) (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=1030)
-> Single-row index lookup on J using PRIMARY (job_id=A.job_id) (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=1030)

Index 3: status from Applies table

Table: Applies

Indexed columns: status

Indices names: idx_status

SHOW INDEX FROM Applies;

Result Grid														
Table	Non_unique	Key_name	Seq_in_Index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
A...	0	PRIMARY	1	student_id	A	925		HULL	HULL	BTREE			YES	HULL
A...	0	PRIMARY	2	job_id	A	2771		HULL	HULL	BTREE			YES	HULL
A...	1	job_id	1	job_id	A	2771		HULL	HULL	BTREE			YES	HULL
A...	1	idx_status	1	status	A	3		HULL	HULL	BTREE			YES	HULL

Result 64

Read Only

EXPLAIN ANALYZE execution time: 0.027 sec

Result Grid					
EXPLAIN					
> Limit: 15 row(s) (actual time=5.598..5.600 rows=15 loops=1) -> Sort: avg_salary DESC, limit input to 15 row(s) per chunk (actual time=5.597..5.598 rows=15 loops=1)...					
Object Info	Session	Time	Action	Response	Duration / Fetch Time
No object selected		14:40:17	show index from Job_Role	3 row(s) returned	0.020 sec / 0.00001...
		14:40:51	explain analyze SELECT J.job_title, St.university_name, AVG(J.salary)...	1 row(s) returned	0.031 sec / 0.000014...
		14:42:52	show index from Job_Role	4 row(s) returned	0.022 sec / 0.00047...
		17:30:48	drop index idx_job_salary on Job_Role	0 row(s) affected Records: 0 Duplicates: 0 Warnings:...	0.056 sec
		17:30:55	show index from Job_Role	3 row(s) returned	0.022 sec / 0.000013...
		17:31:09	drop index idx_salary on Job_Role	0 row(s) affected Records: 0 Duplicates: 0 Warnings:...	0.045 sec
		17:31:14	show index from Job_Role	2 row(s) returned	0.023 sec / 0.000011...
		17:31:21	show index from Applies	3 row(s) returned	0.024 sec / 0.000024...
		17:31:30	create index idx_status on Applies(status)	0 row(s) affected Records: 0 Duplicates: 0 Warnings:...	0.115 sec
		17:31:41	show index from Applies	4 row(s) returned	0.021 sec / 0.000013...
		17:31:48	explain analyze SELECT J.job_title, St.university_name, AVG(J.salary)...	1 row(s) returned	0.027 sec / 0.000019...
		17:32:42	show index from Applies	4 row(s) returned	0.022 sec / 0.000019...

Query Completed

EXPLAIN ANALYZE output:

'-> Limit: 15 row(s) (actual time=5.598..5.600 rows=15 loops=1)
-> Sort: avg_salary DESC, limit input to 15 row(s) per chunk (actual time=5.597..5.598 rows=15 loops=1)
-> Table scan on <temporary> (actual time=5.071..5.235 rows=1029 loops=1)
-> Aggregate using temporary table (actual time=5.067..5.067 rows=1029 loops=1)
-> Nested loop inner join (cost=934.18 rows=1030) (actual time=0.081..3.300 rows=1030 loops=1)

```
-> Nested loop inner join  (cost=573.68 rows=1030) (actual time=0.073..1.777 rows=1030 loops=1)
   > Covering index lookup on A using idx_status (status='Accepted')
(cost=213.18 rows=1030) (actual time=0.060..0.496 rows=1030 loops=1)
   ->
Single-row index lookup on St using PRIMARY (student_id=A.student_id) (cost=0.25 rows=1)
(actual time=0.001..0.001 rows=1 loops=1030)
   -> Single-row index lookup on J
using PRIMARY (job_id=A.job_id) (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1
loops=1030)
```

Indexing Observation:

Query 1:

1. Index on column "salary":

The "salary" column is used in our queries' WHERE clauses. The query performance with an index on "salary" improved to 0.020 seconds from the default 0.026 seconds. This improvement in performance could be due to the fact that data is efficiently filtered and retrieved using salary index.

2. Index on column "job_type":

The query performance with an index on "job_type" improved to 0.024 seconds, which is still better than the default. But, it is slightly slower than just indexing "salary." This could be due to the data distribution and query patterns; "salary" may have more distinct values, making it more efficient to filter on.

3. Index using both "salary" and "job_type":

The query performance further improved to 0.019 seconds, which is slightly better than indexing just "salary" alone. This suggests that for queries involving both "salary" and "job_type," the composite index is beneficial. This composite index helps the database to locate rows that match both conditions efficiently.

Index chosen:

Index using both "salary" and "job_type":

We chose to implement the composite indexing design using both salary and job_type as this method gave the best results.

Query 2:

1. Index on column "salary":

The query finds the average wage using the "salary" field. By indexing it, the database can easily find and compile wage numbers, which is essential for figuring out the average.

From the default 0.043 seconds, the query performance with an index on "salary" was increased to 0.028 seconds. This suggests that indexing the "salary" will accelerate the aggregate process and query execution.

2. Index on column "job_title":

The query's GROUP BY clause makes use of the "job_title" column. Data is effectively grouped by job title with the use of indexing. Compared to the default, the query speed improved to 0.027 seconds using an index on "job_title". More effective categorization is made possible by this index, which is important for figuring out the average pay for each job title.

3. Index on column "status":

The WHERE clause filters for "Accepted" applications using the "status" column. The performance of the query with an index on "status" was much better, reaching 0.0027 seconds. By indexing it, the speed increases the execution of queries by reducing the time required to choose the data.

Index chosen:

Index using “status”

We have chosen to implement indexing on "status" as there was a significant increase in the performance when compared to indexing "salary" and "job_title"