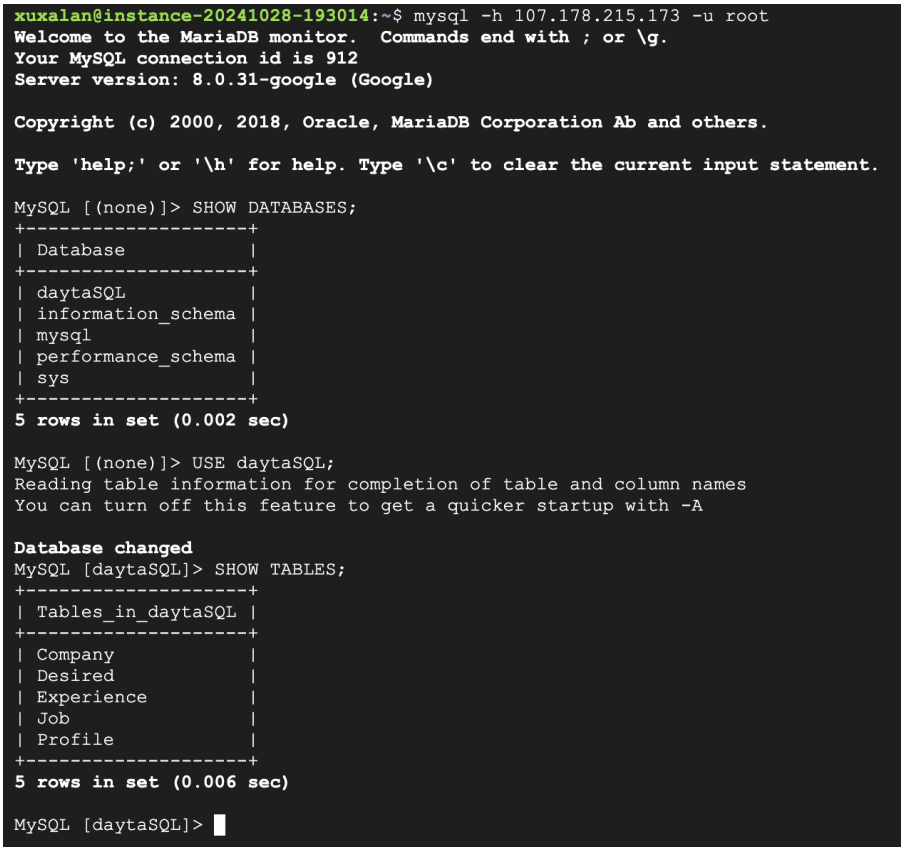
Project Track 1 Stage 3 Report

Team-026 Dayta

1.1 Implement the database tables locally or on GCP.



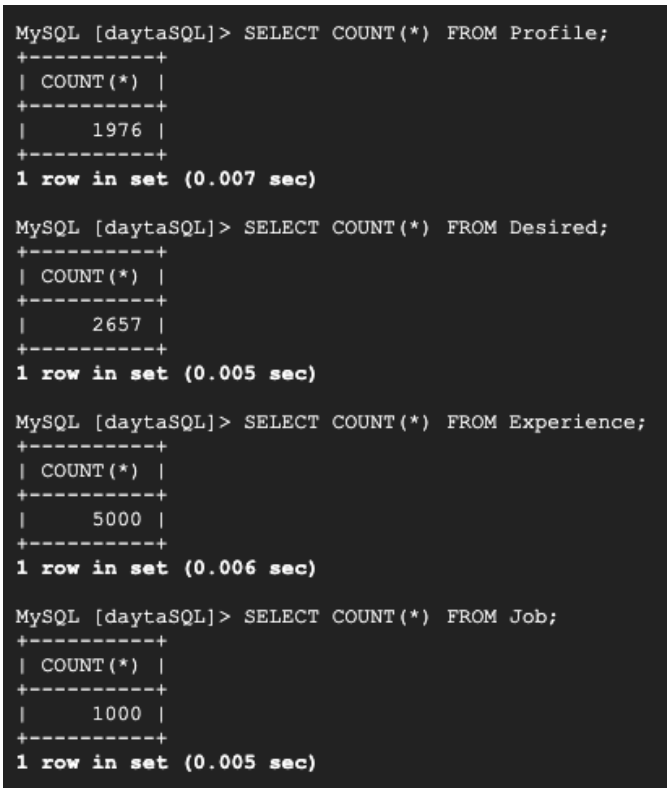
1.2 Provide the Data Definition Language (DDL) commands you used to create each table in the database.

```
CREATE TABLE Company (
        CompanyId INT NOT NULL,
        CompanyName VARCHAR(255) NOT NULL,
        EmployeeCount INT,
        PRIMARY KEY (CompanyId)
);
```

```
CREATE TABLE Profile (
        FirstName VARCHAR(255) NOT NULL,
        LastName VARCHAR(255) NOT NULL,
        UserName VARCHAR(255) NOT NULL,
        Password VARCHAR(255) NOT NULL,
        Email VARCHAR(255),
        PRIMARY KEY (UserName)
);

CREATE TABLE Desired (
        UserName VARCHAR(255) NOT NULL,
        JobCategory VARCHAR(255) NOT NULL,
        SalaryRange INT,
        Skills VARCHAR(255),
        Location VARCHAR(255),
        Type VARCHAR(255),
        PRIMARY KEY (UserName, JobCategory),
        FOREIGN KEY (UserName) REFERENCES Profile(UserName)
                ON DELETE CASCADE
);

CREATE TABLE Experience (
        ExperienceId INT AUTO_INCREMENT,
        UserName VARCHAR(255) NOT NULL,
        CompanyId INT,
        Title VARCHAR(255),
        StartDate DATE,
        EndDate DATE,
        Description VARCHAR(2047),
        PRIMARY KEY (ExperienceId),
        FOREIGN KEY (UserName) REFERENCES Profile(UserName)
                ON DELETE CASCADE,
        FOREIGN KEY (CompanyId) REFERENCES Company(CompanyId)
                ON DELETE SET NULL ON UPDATE SET NULL
);

CREATE TABLE Job (
        JobId INT AUTO_INCREMENT,
```

```
    CompanyId INT NOT NULL,
    Title VARCHAR(255) NOT NULL,
    Category VARCHAR(255) NOT NULL,
    Location VARCHAR(255),
    Duration INT,
    Type VARCHAR(255),
    SkillsKeyWord VARCHAR(1023),
    PRIMARY KEY (JobId),
    FOREIGN KEY (CompanyId) REFERENCES Company(CompanyId)
        ON DELETE CASCADE
);
```

1.3 Insert at least 1000 rows in the tables.



```
MySQL [daytaSQL]> SELECT COUNT(*) FROM Profile;
+----------+
| COUNT(*) |
+----------+
|     1976 |
+----------+
1 row in set (0.007 sec)

MySQL [daytaSQL]> SELECT COUNT(*) FROM Desired;
+----------+
| COUNT(*) |
+----------+
|     2657 |
+----------+
1 row in set (0.005 sec)

MySQL [daytaSQL]> SELECT COUNT(*) FROM Experience;
+----------+
| COUNT(*) |
+----------+
|     5000 |
+----------+
1 row in set (0.006 sec)

MySQL [daytaSQL]> SELECT COUNT(*) FROM Job;
+----------+
| COUNT(*) |
+----------+
|     1000 |
+----------+
1 row in set (0.005 sec)
```

1.4 Four advanced queries with top 15 rows.

1) Find Matched Job Positions Based on Skills and Category Only.

```
SELECT
        d.UserName, j.JobId, j.Title, j.CompanyId, j.Category,
        COUNT(CASE WHEN j.SkillsKeyWord LIKE CONCAT('%', d.Skills, '%') THEN 1 END)
                AS SkillMatchCount,
        COUNT(CASE WHEN j.Category = d.JobCategory THEN 1 END)
                AS CategoryMatchCount,
        (COUNT(CASE WHEN j.SkillsKeyWord LIKE CONCAT('%', d.Skills, '%') THEN 1 END) +
                COUNT(CASE WHEN j.Category = d.JobCategory THEN 1 END))
                AS TotalMatchScore
FROM
        Desired d JOIN Job j ON
                (d.JobCategory = j.Category OR j.SkillsKeyWord LIKE CONCAT('%', d.Skills, '%'))
GROUP BY
        d.UserName, j.JobId
HAVING
        TotalMatchScore > 0
ORDER BY
        d.UserName, TotalMatchScore DESC
LIMIT 15;
```

```
    -> LIMIT 15;
+-----------+-------+----------------------+-----------+-----------+----------------+--------------------+-----------------+
| UserName  | JobId | Title                | CompanyId | Category  | SkillMatchCount | CategoryMatchCount | TotalMatchScore |
+-----------+-------+----------------------+-----------+-----------+----------------+--------------------+-----------------+
| aanderson |     6 | Marketing Specialist |        41 | Marketing |              0 |                  1 |               1 |
| aanderson |    11 | Data Engineer        |        93 | IT        |              1 |                  0 |               1 |
| aanderson |    12 | Data Analyst         |       129 | IT        |              1 |                  0 |               1 |
| aanderson |    14 | Data Analyst         |        44 | IT        |              1 |                  0 |               1 |
| aanderson |    16 | Marketing Specialist |        24 | Marketing |              0 |                  1 |               1 |
| aanderson |    18 | Data Analyst         |         3 | IT        |              1 |                  0 |               1 |
| aanderson |    19 | Marketing Specialist |        35 | Marketing |              0 |                  1 |               1 |
| aanderson |    22 | Software Engineer    |        11 | IT        |              1 |                  0 |               1 |
| aanderson |    25 | Software Engineer    |       125 | IT        |              1 |                  0 |               1 |
| aanderson |    30 | Marketing Specialist |        51 | Marketing |              0 |                  1 |               1 |
| aanderson |    31 | Data Engineer        |       148 | IT        |              1 |                  0 |               1 |
| aanderson |    37 | Data Analyst         |        73 | IT        |              1 |                  0 |               1 |
| aanderson |    40 | Marketing Specialist |        46 | Marketing |              0 |                  1 |               1 |
| aanderson |    46 | Marketing Coordinator |       27 | Marketing |              0 |                  1 |               1 |
| aanderson |    53 | Data Engineer        |        33 | IT        |              1 |                  0 |               1 |
+-----------+-------+----------------------+-----------+-----------+----------------+--------------------+-----------------+
15 rows in set (6.221 sec)
```

2) Retrieve Company Info for 'In Person' Job Providers Using Subquery.

```
SELECT
	c.CompanyId, c.CompanyName,
	(
		SELECT COUNT(j.JobId)
		FROM Job j
		WHERE j.CompanyId = c.CompanyId AND j.Type = 'in-person'
	) AS InPersonJobCount
FROM
	Company c
WHERE
	c.CompanyId IN (
		SELECT DISTINCT j.CompanyId
		FROM Job j
		WHERE j.Type = 'in-person'
	)
GROUP BY
	c.CompanyId, c.CompanyName
HAVING
	InPersonJobCount > 0
ORDER BY
	InPersonJobCount DESC
LIMIT 15;
```

```
+-----------+---------------------------------+------------------+
| CompanyId | CompanyName                     | InPersonJobCount |
+-----------+---------------------------------+------------------+
|        28 | Watson-Lee                      |                4 |
|        18 | Bass-Bolton                     |                4 |
|        51 | Thomas-Wu                       |                4 |
|         3 | Villa, Carpenter and Henry      |                4 |
|        83 | Anderson, Nguyen and Graves     |                3 |
|       125 | Richards-Jackson                |                3 |
|        87 | Rush, Brown and Ortiz           |                3 |
|       111 | Estrada, Armstrong and Robinson |                3 |
|        91 | Stephenson-Williams             |                3 |
|        53 | Giles-Thomas                    |                3 |
|       105 | Foster, Garcia and King         |                3 |
|        80 | Calhoun-Rivers                  |                3 |
|       100 | Smith, Miller and Jones         |                3 |
|       138 | Ross-Zamora                     |                3 |
|       148 | Clarke-Jones                    |                3 |
+-----------+---------------------------------+------------------+
15 rows in set (0.007 sec)
```

3) List companies with both job openings and users who have desired positions in the same location.

```
SELECT
        j.CompanyId, j.Location,
        COUNT(DISTINCT j.JobId) AS JobOpeningsCount,
        COUNT(DISTINCT d.UserName) AS InterestedUsersCount
FROM
        Job j JOIN Desired d ON j.Location = d.Location
GROUP BY
        j.CompanyId, j.Location
HAVING
        JobOpeningsCount > 0 AND InterestedUsersCount > 0
ORDER BY
        j.CompanyId, JobOpeningsCount DESC
LIMIT 15;
```

```
+-----------+--------------------+------------------+----------------------+
| CompanyId | Location           | JobOpeningsCount | InterestedUsersCount |
+-----------+--------------------+------------------+----------------------+
|         0 | Bowmanstad         |                1 |                   37 |
|         0 | Maldonadoshire     |                1 |                   29 |
|         0 | Wadetown           |                1 |                   19 |
|         0 | West Frankshire    |                1 |                   25 |
|         0 | West Kimberlyton   |                1 |                   23 |
|         1 | New Laurenside     |                1 |                   30 |
|         2 | Davismouth         |                1 |                   26 |
|         2 | Lake Brittany      |                1 |                   29 |
|         2 | Ramoshaven         |                1 |                   29 |
|         2 | South Allison      |                1 |                   30 |
|         3 | Lauramouth         |                1 |                   32 |
|         3 | Masseyhaven        |                1 |                   30 |
|         3 | New Laurenside     |                1 |                   30 |
|         3 | North Davidborough |                1 |                   28 |
|         3 | Robinville         |                1 |                   36 |
+-----------+--------------------+------------------+----------------------+
15 rows in set (0.031 sec)
```

4) Identify Highly Demanded Job Categories by Company with Detailed User Interest.

```
SELECT
	j.CompanyId, c.CompanyName, j.Category,
	COUNT(DISTINCT j.JobId) AS JobOpeningsCount,
	(
		SELECT COUNT(DISTINCT d.UserName)
		FROM Desired d
		WHERE d.JobCategory = j.Category
	) AS InterestedUsersCount
FROM
	Job j JOIN Company c ON j.CompanyId = c.CompanyId
GROUP BY
	j.CompanyId, c.CompanyName, j.Category
HAVING
	JobOpeningsCount > 0 AND InterestedUsersCount > 0
ORDER BY
	InterestedUsersCount DESC, JobOpeningsCount DESC
LIMIT 15;
```

```
+-----------+---------------------------+-----------------+-----------------+----------------------+
| CompanyId | CompanyName               | Category        | JobOpeningsCount | InterestedUsersCount |
+-----------+---------------------------+-----------------+-----------------+----------------------+
|        84 | Baxter PLC                | Customer Service |               2 |                  453 |
|       140 | Cox Ltd                   | Customer Service |               2 |                  453 |
|         1 | Wright Group              | Customer Service |               1 |                  453 |
|         3 | Villa, Carpenter and Henry | Customer Service |               1 |                  453 |
|         6 | Wilson-Irwin              | Customer Service |               1 |                  453 |
|         8 | Frank Ltd                 | Customer Service |               1 |                  453 |
|         9 | Clark-Williams            | Customer Service |               1 |                  453 |
|        10 | Barton-Cruz               | Customer Service |               1 |                  453 |
|        30 | Evans-Butler              | Customer Service |               1 |                  453 |
|        32 | Johnson-Schultz           | Customer Service |               1 |                  453 |
|        35 | Hill PLC                  | Customer Service |               1 |                  453 |
|        39 | Thomas PLC                | Customer Service |               1 |                  453 |
|        45 | Phillips Ltd              | Customer Service |               1 |                  453 |
|        47 | Taylor-Barnett            | Customer Service |               1 |                  453 |
|        48 | Cooper-Haley              | Customer Service |               1 |                  453 |
+-----------+---------------------------+-----------------+-----------------+----------------------+
15 rows in set (1.189 sec)
```

2.1 Use the EXPLAIN ANALYZE command to measure your advanced query performance before adding indexes.

1) Explain Analyze screenshots (BEFORE ADDING INDEX)

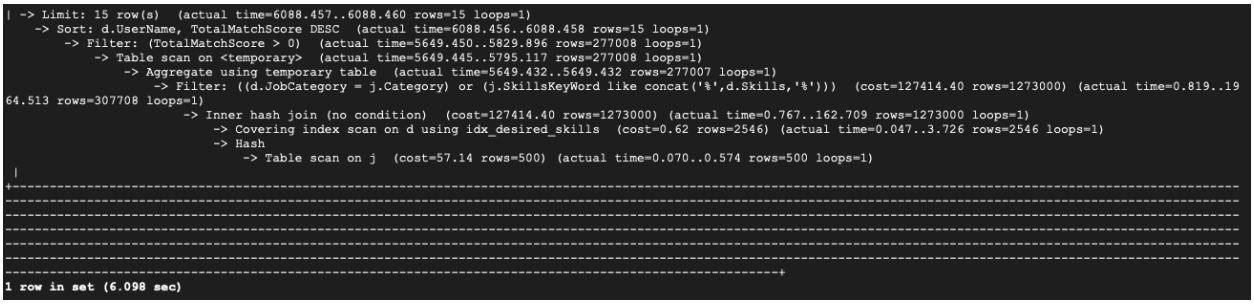
```
MySQL [daytaSQL]> EXPLAIN ANALYZE
    -> SELECT
    ->     d.UserName,
    ->     j.JobId,
    ->     j.Title,
    ->     j.CompanyId,
    ->     j.Category,
    ->     COUNT(CASE WHEN j.SkillsKeyWord LIKE CONCAT('%', d.Skills, '%') THEN 1 END) AS SkillMatchCount,
    ->     COUNT(CASE WHEN j.Category = d.JobCategory THEN 1 END) AS CategoryMatchCount,
    ->     (COUNT(CASE WHEN j.SkillsKeyWord LIKE CONCAT('%', d.Skills, '%') THEN 1 END) +
    ->      COUNT(CASE WHEN j.Category = d.JobCategory THEN 1 END)) AS TotalMatchScore
    -> FROM
    ->     Desired d
    -> JOIN
    ->     Job j ON (d.JobCategory = j.Category OR j.SkillsKeyWord LIKE CONCAT('%', d.Skills, '%'))
    -> GROUP BY
    ->     d.UserName, j.JobId
    -> HAVING
    ->     TotalMatchScore > 0
    -> ORDER BY
    ->     d.UserName, TotalMatchScore DESC
    -> LIMIT 15;
+------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------+
| EXPLAIN
                                                                                                     |
+------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------+
| -> Limit: 15 row(s)  (actual time=6304.591..6304.594 rows=15 loops=1)
    -> Sort: d.UserName, TotalMatchScore DESC  (actual time=6304.590..6304.591 rows=15 loops=1)
        -> Filter: (TotalMatchScore > 0)  (actual time=5856.552..6036.765 rows=277008 loops=1)
            -> Table scan on <temporary>  (actual time=5856.547..6003.009 rows=277008 loops=1)
                -> Aggregate using temporary table  (actual time=5856.538..5856.538 rows=277007 loops=1)
                    -> Filter: ((d.JobCategory = j.Category) or (j.SkillsKeyWord like concat('%',d.Skills,'%')))  (cost=127409.26 rows=1273000) (actual time=0.419..1963.507 rows=307708 loops=1)
                        -> Inner hash join (no condition)  (cost=127409.26 rows=1273000) (actual time=0.408..160.107 rows=1273000 loops=1)
                            -> Table scan on d  (cost=0.62 rows=2546) (actual time=0.025..4.283 rows=2546 loops=1)
                            -> Hash
                                -> Table scan on j  (cost=52.00 rows=500) (actual time=0.070..0.307 rows=500 loops=1)
 |
+------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------+
1 row in set (6.314 sec)
```

1. Index on Job(Category)
CREATE INDEX idx_job_category ON Job(Category);

```
| -> Limit: 15 row(s)  (actual time=6071.066..6071.069 rows=15 loops=1)
    -> Sort: d.UserName, TotalMatchScore DESC  (actual time=6071.066..6071.067 rows=15 loops=1)
        -> Filter: (TotalMatchScore > 0)  (actual time=5642.276..5820.449 rows=277008 loops=1)
            -> Table scan on <temporary>  (actual time=5642.264..5786.133 rows=277008 loops=1)
                -> Aggregate using temporary table  (actual time=5642.254..5642.254 rows=277007 loops=1)
                    -> Filter: ((d.JobCategory = j.Category) or (j.SkillsKeyWord like concat('%',d.Skills,'%')))  (cost=127414.40 rows=1273000) (actual time=0.588..1962.046 rows=307708 loops=1)
                        -> Inner hash join (no condition)  (cost=127414.40 rows=1273000) (actual time=0.561..158.131 rows=1273000 loops=1)
                            -> Covering index scan on d using idx_desired_skills  (cost=0.62 rows=2546) (actual time=0.073..3.632 rows=2546 loops=1)
                            -> Hash
                                -> Table scan on j  (cost=57.14 rows=500) (actual time=0.039..0.340 rows=500 loops=1)
 |
+------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------+
1 row in set (6.086 sec)
```

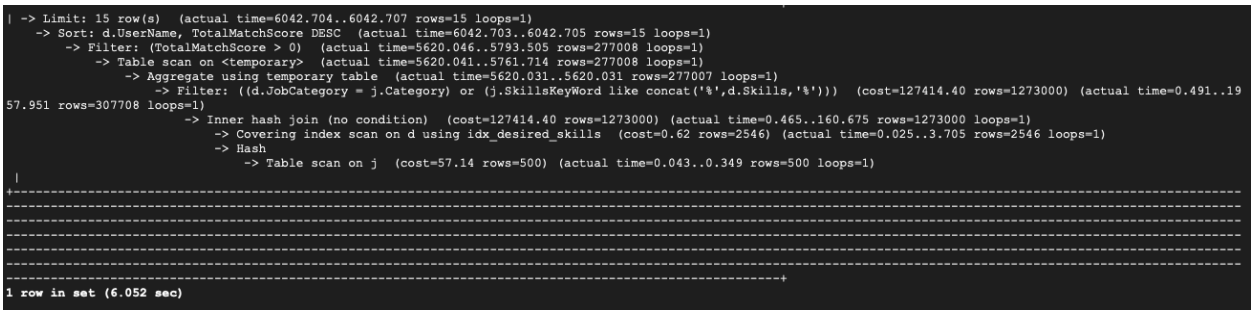
2.**Index on** Desired(JobCategory)

CREATE INDEX idx_desired_jobcategory ON Desired(JobCategory);



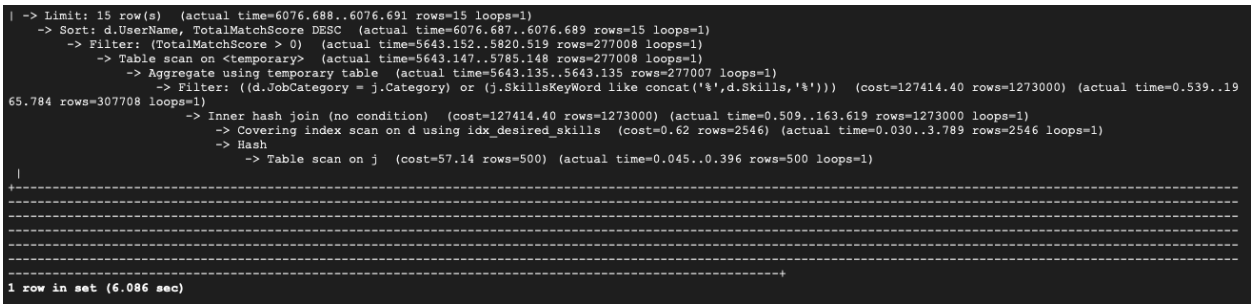
3. **Composite Index on** Job(Category, SkillsKeyWord)

CREATE INDEX idx_job_category_skills ON Job(Category, SkillsKeyWord);



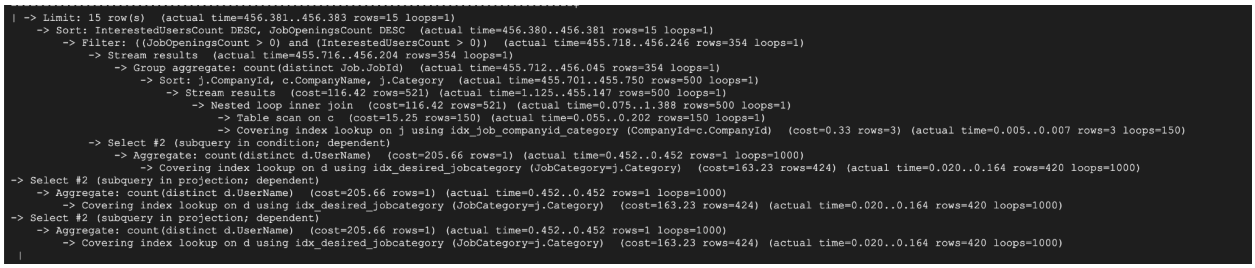
4. **Index on** Job(SkillsKeyWord)

CREATE INDEX idx_job_skillskeyword ON Job(SkillsKeyWord);



5. **Composite Index on** Desired(JobCategory, Skills)

CREATE INDEX idx_desired_jobcategory_skills ON Desired(JobCategory, Skills);



Index configuration	Total cost	
No index	127409.25	Initial query without indexes

Index on Job(Category)	127414.40	No observed improvement. The index on Category might not significantly optimize the join or aggregation due to the OR condition in the join and pattern matching in the aggregation.
Index on Desired(JobCategory)	127414.40	No improvement observed. No improvement in join
Composite Index on Job(Category, SkillsKeyWord)	127414.40	No improvement observed. This composite index does not provide benefit, likely because the pattern matching on SkillsKeyWord does not effectively utilize the index.
Index on Job(SkillsKeyWord) using partial (100 chars)	127414.40	No improvement observed. The partial index on SkillsKeyWord might be less effective due to the wildcard pattern used (LIKE '%...%'), which generally limits index use.
Composite index on Desired(JobCategory, Skills)	127414.40	No improvement observed. Although indexing both columns, the OR condition in the join and complex filtering likely prevent effective use of the index.

Complex queries with multiple joins, subqueries, or aggregated computations might not benefit linearly from indexing. Sometimes, the optimizer's path that includes using the index might appear costlier due to these complexities.

2) Explain Analyze screenshoot (BEFORE ADDING INDEX)

```
MySQL [daytaSQL]> explain analyze SELECT
    -> c.CompanyId, c.CompanyName,
    -> (
    -> SELECT COUNT(j.JobId)
    -> FROM Job j
    -> WHERE j.CompanyId = c.CompanyId AND j.Type = 'in-person'
    -> ) AS InPersonJobCount
    -> FROM
    -> Company c
    -> WHERE
    -> c.CompanyId IN (
    -> SELECT DISTINCT j.CompanyId
    -> FROM Job j
    -> WHERE j.Type = 'in-person'
    -> )
    -> GROUP BY
    -> c.CompanyId, c.CompanyName
    -> HAVING
    -> InPersonJobCount > 0
    -> ORDER BY
    -> InPersonJobCount DESC
    -> LIMIT 15;
+--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------
-------------------------+
| EXPLAIN


+--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------
-------------------------+
| -> Limit: 15 row(s)  (actual time=1.590..1.592 rows=15 loops=1)
    -> Sort: InPersonJobCount DESC  (actual time=1.590..1.590 rows=15 loops=1)
        -> Filter: (InPersonJobCount > 0)  (actual time=1.544..1.564 rows=91 loops=1)
            -> Table scan on <temporary>  (cost=82.69..85.75 rows=50) (actual time=1.543..1.555 rows=91 loops=1)
                -> Temporary table with deduplication  (cost=82.63..82.63 rows=50) (actual time=1.541..1.541 rows=91 loops=1)
                    -> Nested loop inner join  (cost=77.63 rows=50) (actual time=0.433..0.573 rows=91 loops=1)
                        -> Table scan on <subquery3>  (cost=57.06..60.13 rows=50) (actual time=0.418..0.431 rows=91 loops=1)
                            -> Materialize with deduplication  (cost=57.00..57.00 rows=50) (actual time=0.417..0.417 rows=91 loops=1)
                                -> Filter: (j.`Type` = 'in-person')  (cost=52.00 rows=50) (actual time=0.044..0.386 rows=152 loops=1)
                                    -> Table scan on j  (cost=52.00 rows=500) (actual time=0.040..0.308 rows=500 loops=1)
                        -> Single-row index lookup on c using PRIMARY (CompanyId=`<subquery3>`.CompanyId)  (cost=12.60 rows=1) (actual time=0.001..0.001 rows=1 loops=91)
            -> Select #2 (subquery in condition; dependent)
                -> Aggregate: count(j.JobId)  (cost=0.94 rows=1) (actual time=0.009..0.009 rows=1 loops=91)
                    -> Filter: (j.`Type` = 'in-person')  (cost=0.90 rows=0.3) (actual time=0.007..0.009 rows=2 loops=91)
                        -> Index lookup on j using CompanyId (CompanyId=c.CompanyId)  (cost=0.90 rows=3) (actual time=0.007..0.008 rows=4 loops=91)
-> Select #2 (subquery in projection; dependent)
    -> Aggregate: count(j.JobId)  (cost=0.94 rows=1) (actual time=0.009..0.009 rows=1 loops=91)
        -> Filter: (j.`Type` = 'in-person')  (cost=0.90 rows=0.3) (actual time=0.007..0.009 rows=2 loops=91)
            -> Index lookup on j using CompanyId (CompanyId=c.CompanyId)  (cost=0.90 rows=3) (actual time=0.007..0.008 rows=4 loops=91)
 |
```

1. **Index on CompanyId in the** Company **table:**

CREATE INDEX idx_company_companyid ON Company(CompanyId);

```
| -> Limit: 15 row(s)  (actual time=2.055..2.057 rows=15 loops=1)
    -> Sort: InPersonJobCount DESC  (actual time=2.054..2.055 rows=15 loops=1)
        -> Filter: (InPersonJobCount > 0)  (actual time=2.004..2.026 rows=91 loops=1)
            -> Table scan on <temporary>  (cost=87.83..90.89 rows=50) (actual time=2.002..2.015 rows=91 loops=1)
                -> Temporary table with deduplication  (cost=87.77..87.77 rows=50) (actual time=2.000..2.000 rows=91 loops=1)
                    -> Nested loop inner join  (cost=82.77 rows=50) (actual time=0.365..0.522 rows=91 loops=1)
                        -> Table scan on <subquery3>  (cost=62.21..65.27 rows=50) (actual time=0.355..0.370 rows=91 loops=1)
                            -> Materialize with deduplication  (cost=62.14..62.14 rows=50) (actual time=0.354..0.354 rows=91 loops=1)
                                -> Filter: (j.`Type` = 'in-person')  (cost=57.14 rows=50) (actual time=0.038..0.325 rows=152 loops=1)
                                    -> Table scan on j  (cost=57.14 rows=500) (actual time=0.034..0.253 rows=500 loops=1)
                        -> Single-row index lookup on c using PRIMARY (CompanyId=`<subquery3>`.CompanyId)  (cost=12.60 rows=1) (actual time=0.001..0.001 rows=1 loops=
91)
            -> Select #2 (subquery in condition; dependent)
                -> Aggregate: count(j.JobId)  (cost=3.17 rows=1) (actual time=0.014..0.014 rows=1 loops=91)
                    -> Filter: (j.`Type` = 'in-person')  (cost=3.13 rows=0.3) (actual time=0.012..0.014 rows=2 loops=91)
                        -> Index lookup on j using idx_job_companyid (CompanyId=c.CompanyId)  (cost=3.13 rows=3) (actual time=0.011..0.013 rows=4 loops=91)
-> Select #2 (subquery in projection; dependent)
    -> Aggregate: count(j.JobId)  (cost=3.17 rows=1) (actual time=0.014..0.014 rows=1 loops=91)
        -> Filter: (j.`Type` = 'in-person')  (cost=3.13 rows=0.3) (actual time=0.012..0.014 rows=2 loops=91)
            -> Index lookup on j using idx_job_companyid (CompanyId=c.CompanyId)  (cost=3.13 rows=3) (actual time=0.011..0.013 rows=4 loops=91)
 |
```

2. **Index on CompanyId and Type in the** Job **table:**

CREATE INDEX idx_job_companyid_type ON Job(CompanyId, Type);

```
| -> Limit: 15 row(s)  (actual time=0.984..0.985 rows=15 loops=1)
    -> Sort: InPersonJobCount DESC  (actual time=0.983..0.984 rows=15 loops=1)
        -> Filter: (InPersonJobCount > 0)  (actual time=0.937..0.957 rows=91 loops=1)
            -> Table scan on <temporary>  (cost=32.85..35.34 rows=14) (actual time=0.935..0.947 rows=91 loops=1)
                -> Temporary table with deduplication  (cost=32.66..32.66 rows=14) (actual time=0.933..0.933 rows=91 loops=1)
                    -> Nested loop inner join  (cost=31.22 rows=14) (actual time=0.056..0.427 rows=91 loops=1)
                        -> Remove duplicates from input sorted on idx_job_companyid_type  (cost=17.28 rows=14) (actual time=0.044..0.287 rows=91 loops=1)
                            -> Filter: (j.`Type` = 'in-person')  (cost=17.28 rows=14) (actual time=0.042..0.264 rows=152 loops=1)
                                -> Covering index scan on j using idx_job_companyid_type  (cost=17.28 rows=500) (actual time=0.035..0.183 rows=500 loops=1)
                        -> Single-row index lookup on c using PRIMARY (CompanyId=j.CompanyId)  (cost=12.60 rows=1) (actual time=0.001..0.001 rows=1 loops=91)
            -> Select #2 (subquery in condition; dependent)
                -> Aggregate: count(j.JobId)  (cost=0.61 rows=1) (actual time=0.004..0.004 rows=1 loops=91)
                    -> Covering index lookup on j using idx_job_companyid_type (CompanyId=c.CompanyId, Type='in-person')  (cost=0.44 rows=2) (actual time=0.003..0.004
 rows=2 loops=91)
-> Select #2 (subquery in projection; dependent)
    -> Aggregate: count(j.JobId)  (cost=0.61 rows=1) (actual time=0.004..0.004 rows=1 loops=91)
        -> Covering index lookup on j using idx_job_companyid_type (CompanyId=c.CompanyId, Type='in-person')  (cost=0.44 rows=2) (actual time=0.003..0.004 rows=2 loop
s=91)
 |
```

3. **Composite Index on Type and CompanyId in the** Job **table for the DISTINCT operation:**

CREATE INDEX idx_job_type_companyid ON Job(Type, CompanyId);

```
| -> Limit: 15 row(s)  (actual time=1.203..1.205 rows=15 loops=1)
    -> Sort: InPersonJobCount DESC  (actual time=1.202..1.203 rows=15 loops=1)
        -> Filter: (InPersonJobCount > 0)  (actual time=1.153..1.174 rows=91 loops=1)
            -> Table scan on <temporary>  (cost=4590.26..4877.75 rows=22800) (actual time=1.150..1.163 rows=91 loops=1)
                -> Temporary table with deduplication  (cost=4590.25..4590.25 rows=22800) (actual time=1.148..1.148 rows=91 loops=1)
                    -> Nested loop inner join  (cost=2310.25 rows=22800) (actual time=0.195..0.329 rows=91 loops=1)
                        -> Table scan on c  (cost=15.25 rows=150) (actual time=0.059..0.100 rows=150 loops=1)
                        -> Single-row index lookup on <subquery3> using <auto_distinct_key> (CompanyId=c.CompanyId)  (actual time=0.001..0.001 rows=1 loops=150)
                            -> Materialize with deduplication  (cost=35.37..35.37 rows=152) (actual time=0.133..0.133 rows=91 loops=1)
                                -> Covering index lookup on j using idx_job_type_companyid (Type='in-person')  (cost=20.17 rows=152) (actual time=0.025..0.085 rows=15
2 loops=1)
            -> Select #2 (subquery in condition; dependent)
                -> Aggregate: count(j.JobId)  (cost=0.61 rows=1) (actual time=0.007..0.007 rows=1 loops=91)
                    -> Covering index lookup on j using idx_job_companyid_type (CompanyId=c.CompanyId, Type='in-person')  (cost=0.44 rows=2) (actual time=0.006..0.007
 rows=2 loops=91)
-> Select #2 (subquery in projection; dependent)
    -> Aggregate: count(j.JobId)  (cost=0.61 rows=1) (actual time=0.007..0.007 rows=1 loops=91)
        -> Covering index lookup on j using idx_job_companyid_type (CompanyId=c.CompanyId, Type='in-person')  (cost=0.44 rows=2) (actual time=0.006..0.007 rows=2 loop
s=91)
```

4. **Index on Type in the** Job **table:**

CREATE INDEX idx_job_type ON Job(Type);

```
| -> Limit: 15 row(s)  (actual time=0.802..0.803 rows=15 loops=1)
    -> Sort: InPersonJobCount DESC  (actual time=0.801..0.802 rows=15 loops=1)
        -> Filter: (InPersonJobCount > 0)  (actual time=0.729..0.775 rows=91 loops=1)
            -> Table scan on <temporary>  (cost=4590.26..4877.75 rows=22800) (actual time=0.727..0.744 rows=91 loops=1)
                -> Temporary table with deduplication  (cost=4590.25..4590.25 rows=22800) (actual time=0.726..0.726 rows=91 loops=1)
                    -> Nested loop inner join  (cost=2310.25 rows=22800) (actual time=0.161..0.286 rows=91 loops=1)
                        -> Table scan on c  (cost=15.25 rows=150) (actual time=0.042..0.079 rows=150 loops=1)
                        -> Single-row index lookup on <subquery3> using <auto_distinct_key> (CompanyId=c.CompanyId)  (actual time=0.001..0.001 rows=1 loops=150)
                            -> Materialize with deduplication  (cost=35.37..35.37 rows=152) (actual time=0.115..0.115 rows=91 loops=1)
                                -> Covering index lookup on j using idx_job_type_companyid (Type='in-person')  (cost=20.17 rows=152) (actual time=0.025..0.088 rows=15
2 loops=1)
            -> Select #2 (subquery in condition; dependent)
                -> Aggregate: count(j.JobId)  (cost=0.61 rows=1) (actual time=0.004..0.004 rows=1 loops=91)
                    -> Covering index lookup on j using idx_job_companyid_type (CompanyId=c.CompanyId, Type='in-person')  (cost=0.44 rows=2) (actual time=0.003..0.003
 rows=2 loops=91)
-> Select #2 (subquery in projection; dependent)
    -> Aggregate: count(j.JobId)  (cost=0.61 rows=1) (actual time=0.004..0.004 rows=1 loops=91)
        -> Covering index lookup on j using idx_job_companyid_type (CompanyId=c.CompanyId, Type='in-person')  (cost=0.44 rows=2) (actual time=0.003..0.003 rows=2 loop
s=91)
```

Index configuration	Total cost	
No index	82.69	Initial query without indexes
Index on Company(CompanyId)	87.83	Higher cost indicates limited benefit in reducing operational overhead.
Index on Job(CompanyId, Type)	**32.85**	Most effective, significantly reducing cost by optimizing key query conditions.
Index on Job(Type, CompanyId)	4590.26	High cost reflects inefficiency, likely due to suboptimal column order for query needs.
Index on Job(Type)	4590.26	High cost suggests limited utility without coupling with CompanyId.

The **Index on** Job(CompanyId, Type) provides the most substantial performance improvement by effectively reducing the optimizer's cost estimate. This index best matches the query's use of CompanyId and Type in both the main filtering and subquery counting, making it the optimal choice for this SQL query.

- Other indexes either did not improve or significantly worsened performance due to misalignment with the query's operational needs or inefficient column ordering.

3) Explain Analyze screenshot (BEFORE ADDING INDEX)

```
MySQL [daytaSQL]> explain analyze SELECT
    -> j.CompanyId, j.Location,
    -> COUNT(DISTINCT j.JobId) AS JobOpeningsCount,
    -> COUNT(DISTINCT d.UserName) AS InterestedUsersCount
    -> FROM
    -> Job j JOIN Desired d ON j.Location = d.Location
    -> GROUP BY
    -> j.CompanyId, j.Location
    -> HAVING
    -> JobOpeningsCount > 0 AND InterestedUsersCount > 0
    -> ORDER BY
    -> j.CompanyId, JobOpeningsCount DESC
    -> LIMIT 15;
+------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------+
| EXPLAIN



+------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------+
| -> Limit: 15 row(s)  (actual time=28.311..28.313 rows=15 loops=1)
    -> Sort: j.CompanyId, JobOpeningsCount DESC  (actual time=28.311..28.312 rows=15 loops=1)
        -> Filter: ((JobOpeningsCount > 0) and (InterestedUsersCount > 0))  (actual time=19.000..28.210 rows=492 loops=1)
            -> Stream results  (actual time=18.995..28.130 rows=492 loops=1)
                -> Group aggregate: count(distinct Job.JobId), count(distinct Desired.UserName)  (actual time=18.992..27.984 rows=492 loops=1)
                    -> Sort: j.CompanyId, j.Location  (actual time=18.946..20.065 rows=12749 loops=1)
                        -> Stream results  (cost=127365.35 rows=127300) (actual time=0.361..10.362 rows=12749 loops=1)
                            -> Filter: (d.Location = j.Location)  (cost=127365.35 rows=127300) (actual time=0.357..6.603 rows=12749 loops=1)
                                -> Inner hash join (<hash>(d.Location)=<hash>(j.Location))  (cost=127365.35 rows=127300) (actual time=0.355..2.776 rows=12749 loops=1)
                                    -> Table scan on d  (cost=0.08 rows=2546) (actual time=0.026..1.008 rows=2546 loops=1)
                                    -> Hash
                                        -> Table scan on j  (cost=52.00 rows=500) (actual time=0.048..0.213 rows=500 loops=1)
 |
```

1. **Index on** Job(Location):
   CREATE INDEX idx_job_location ON Job(Location);

```
| -> Limit: 15 row(s)  (actual time=50.703..50.705 rows=15 loops=1)
    -> Sort: j.CompanyId, JobOpeningsCount DESC  (actual time=50.702..50.703 rows=15 loops=1)
        -> Filter: ((JobOpeningsCount > 0) and (InterestedUsersCount > 0))  (actual time=41.107..50.585 rows=492 loops=1)
            -> Stream results  (actual time=41.102..50.492 rows=492 loops=1)
                -> Group aggregate: count(distinct Job.JobId), count(distinct Desired.UserName)  (actual time=41.099..50.337 rows=492 loops=1)
                    -> Sort: j.CompanyId, j.Location  (actual time=41.047..42.293 rows=12749 loops=1)
                        -> Stream results  (cost=13025.84 rows=12859) (actual time=0.081..31.986 rows=12749 loops=1)
                            -> Nested loop inner join  (cost=13025.84 rows=12859) (actual time=0.079..27.835 rows=12749 loops=1)
                                -> Filter: (d.Location is not null)  (cost=259.10 rows=2546) (actual time=0.040..1.387 rows=2546 loops=1)
                                    -> Table scan on d  (cost=259.10 rows=2546) (actual time=0.038..1.161 rows=2546 loops=1)
                                -> Index lookup on j using idx_job_location (Location=d.Location)  (cost=4.51 rows=5) (actual time=0.008..0.010 rows=5 loops=2546)
 |
```

2. **Index on** Desired(Location):
   CREATE INDEX idx_desired_location ON Desired(Location);

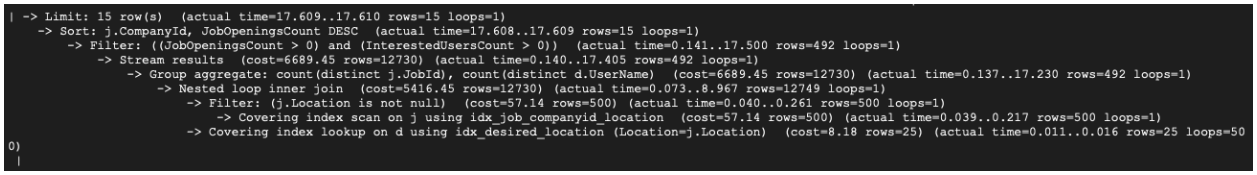
```
| -> Limit: 15 row(s)  (actual time=18.961..18.963 rows=15 loops=1)
    -> Sort: j.CompanyId, JobOpeningsCount DESC  (actual time=18.961..18.961 rows=15 loops=1)
        -> Filter: ((JobOpeningsCount > 0) and (InterestedUsersCount > 0))  (actual time=0.638..18.851 rows=492 loops=1)
            -> Stream results  (cost=6689.45 rows=12730) (actual time=0.635..18.748 rows=492 loops=1)
                -> Group aggregate: count(distinct j.JobId), count(distinct d.UserName)  (cost=6689.45 rows=12730) (actual time=0.632..18.573 rows=492 loops=1)
                    -> Nested loop inner join  (cost=5416.45 rows=12730) (actual time=0.559..10.084 rows=12749 loops=1)
                        -> Sort: j.CompanyId, j.Location  (cost=57.14 rows=500) (actual time=0.501..0.573 rows=500 loops=1)
                            -> Filter: (j.Location is not null)  (cost=57.14 rows=500) (actual time=0.048..0.324 rows=500 loops=1)
                                -> Table scan on j  (cost=57.14 rows=500) (actual time=0.047..0.282 rows=500 loops=1)
                        -> Covering index lookup on d using idx_desired_location (Location=j.Location)  (cost=8.18 rows=25) (actual time=0.012..0.017 rows=25 loops=50
0)
 |
```

3. **Composite Index on** Job(CompanyId, Location):
   CREATE INDEX idx_job_companyid_location ON Job(CompanyId, Location);

```
| -> Limit: 15 row(s)  (actual time=17.609..17.610 rows=15 loops=1)
    -> Sort: j.CompanyId, JobOpeningsCount DESC  (actual time=17.608..17.609 rows=15 loops=1)
        -> Filter: ((JobOpeningsCount > 0) and (InterestedUsersCount > 0))  (actual time=0.141..17.500 rows=492 loops=1)
            -> Stream results  (cost=6689.45 rows=12730) (actual time=0.140..17.405 rows=492 loops=1)
                -> Group aggregate: count(distinct j.JobId), count(distinct d.UserName)  (cost=6689.45 rows=12730) (actual time=0.137..17.230 rows=492 loops=1)
                    -> Nested loop inner join  (cost=5416.45 rows=12730) (actual time=0.073..8.967 rows=12749 loops=1)
                        -> Filter: (j.Location is not null)  (cost=57.14 rows=500) (actual time=0.040..0.261 rows=500 loops=1)
                            -> Covering index scan on j using idx_job_companyid_location  (cost=57.14 rows=500) (actual time=0.039..0.217 rows=500 loops=1)
                        -> Covering index lookup on d using idx_desired_location (Location=j.Location)  (cost=8.18 rows=25) (actual time=0.011..0.016 rows=25 loops=50
0)
 |
```

4. **Composite Index on** Desired(Location, UserName):

CREATE INDEX idx_desired_location_username ON Desired(Location, UserName);



Index Configuration	Cost	**Description**
Initial (No Index)	127370.49	
Index on Job(Location):	13025.84	Drastic reduction in cost; optimizes join efficiency by enabling quicker location-based matches between tables.
Index on Desired(Location)	6689.45	Significant reduction in cost due to optimized join operation on Location.
Index on Job(CompanyId, Location)	6689.45	cost reduction by enhancing data sorting and grouping efficiency.
Index on Desired(Location, UserName)	6689.45	No change

The **Index on** Job(Location) is crucial for optimizing the join condition in your query and should be prioritized for creation due to its significant impact on reducing the query's cost.

4) Explain Analyze screenshot (BEFORE ADDING INDEX)

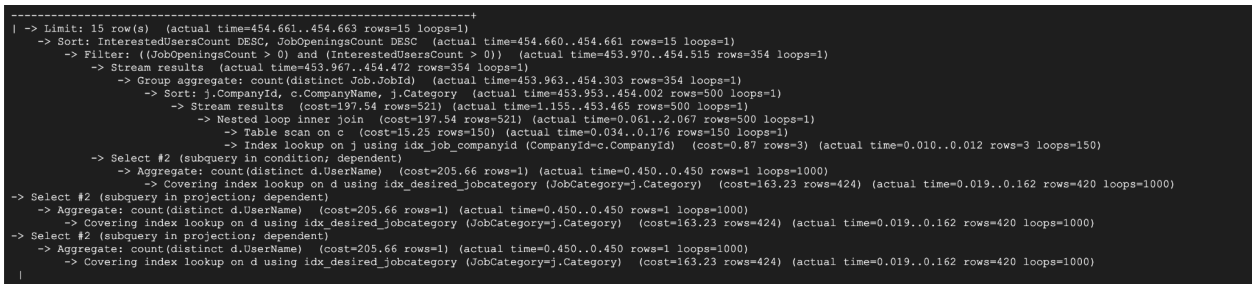
```
MySQL [daytaSQL]> explain analyze SELECT
    -> j.CompanyId, c.CompanyName, j.Category,
    -> COUNT(DISTINCT j.JobId) AS JobOpeningsCount,
    -> (
    -> SELECT COUNT(DISTINCT d.UserName)
    -> FROM Desired d
    -> WHERE d.JobCategory = j.Category
    -> ) AS InterestedUsersCount
    -> FROM
    -> Job j JOIN Company c ON j.CompanyId = c.CompanyId
    -> GROUP BY
    -> j.CompanyId, c.CompanyName, j.Category
    -> HAVING
    -> JobOpeningsCount > 0 AND InterestedUsersCount > 0
    -> ORDER BY
    -> InterestedUsersCount DESC, JobOpeningsCount DESC
    -> LIMIT 15;
+------------------------------------------------------------+
| EXPLAIN

+------------------------------------------------------------+
| -> Limit: 15 row(s)  (actual time=455.667..455.670 rows=15 loops=1)
    -> Sort: InterestedUsersCount DESC, JobOpeningsCount DESC  (actual time=455.667..455.668 rows=15 loops=1)
        -> Filter: ((JobOpeningsCount > 0) and (InterestedUsersCount > 0))  (actual time=454.977..455.537 rows=354 loops=1)
            -> Stream results  (actual time=454.974..455.494 rows=354 loops=1)
                -> Group aggregate: count(distinct Job.JobId)  (actual time=454.971..455.328 rows=354 loops=1)
                    -> Sort: j.CompanyId, c.CompanyName, j.Category  (actual time=454.958..455.014 rows=500 loops=1)
                        -> Stream results  (cost=197.54 rows=521) (actual time=1.139..454.420 rows=500 loops=1)
                            -> Nested loop inner join  (cost=197.54 rows=521) (actual time=0.075..2.420 rows=500 loops=1)
                                -> Table scan on c  (cost=15.25 rows=150) (actual time=0.045..0.199 rows=150 loops=1)
                                -> Index lookup on j using CompanyId (CompanyId=c.CompanyId)  (cost=0.87 rows=3) (actual time=0.012..0.014 rows=3 loops=150)
            -> Select #2 (subquery in condition; dependent)
                -> Aggregate: count(distinct d.UserName)  (cost=205.66 rows=1) (actual time=0.450..0.450 rows=1 loops=1000)
                    -> Covering index lookup on d using idx_desired_jobcategory (JobCategory=j.Category)  (cost=163.23 rows=424) (actual time=0.020..0.164 rows=420 loops=1000)
-> Select #2 (subquery in projection; dependent)
    -> Aggregate: count(distinct d.UserName)  (cost=205.66 rows=1) (actual time=0.450..0.450 rows=1 loops=1000)
        -> Covering index lookup on d using idx_desired_jobcategory (JobCategory=j.Category)  (cost=163.23 rows=424) (actual time=0.020..0.164 rows=420 loops=1000)
-> Select #2 (subquery in projection; dependent)
    -> Aggregate: count(distinct d.UserName)  (cost=205.66 rows=1) (actual time=0.450..0.450 rows=1 loops=1000)
        -> Covering index lookup on d using idx_desired_jobcategory (JobCategory=j.Category)  (cost=163.23 rows=424) (actual time=0.020..0.164 rows=420 loops=1000)
 |
```

1) **Index on Job.CompanyId**
   CREATE INDEX idx_job_companyid ON Job(CompanyId);

```
------------------------------------------------------------+
| -> Limit: 15 row(s)  (actual time=459.261..459.264 rows=15 loops=1)
    -> Sort: InterestedUsersCount DESC, JobOpeningsCount DESC  (actual time=459.261..459.262 rows=15 loops=1)
        -> Filter: ((JobOpeningsCount > 0) and (InterestedUsersCount > 0))  (actual time=458.535..459.098 rows=354 loops=1)
            -> Stream results  (actual time=458.531..459.054 rows=354 loops=1)
                -> Group aggregate: count(distinct Job.JobId)  (actual time=458.528..458.889 rows=354 loops=1)
                    -> Sort: j.CompanyId, c.CompanyName, j.Category  (actual time=458.516..458.574 rows=500 loops=1)
                        -> Stream results  (cost=197.54 rows=521) (actual time=1.335..457.983 rows=500 loops=1)
                            -> Nested loop inner join  (cost=197.54 rows=521) (actual time=0.095..2.358 rows=500 loops=1)
                                -> Table scan on c  (cost=15.25 rows=150) (actual time=0.057..0.233 rows=150 loops=1)
                                -> Index lookup on j using idx_job_companyid (CompanyId=c.CompanyId)  (cost=0.87 rows=3) (actual time=0.010..0.013 rows=3 loops=150)
            -> Select #2 (subquery in condition; dependent)
                -> Aggregate: count(distinct d.UserName)  (cost=205.66 rows=1) (actual time=0.454..0.454 rows=1 loops=1000)
                    -> Covering index lookup on d using idx_desired_jobcategory (JobCategory=j.Category)  (cost=163.23 rows=424) (actual time=0.020..0.164 rows=420 loops=1000)
-> Select #2 (subquery in projection; dependent)
    -> Aggregate: count(distinct d.UserName)  (cost=205.66 rows=1) (actual time=0.454..0.454 rows=1 loops=1000)
        -> Covering index lookup on d using idx_desired_jobcategory (JobCategory=j.Category)  (cost=163.23 rows=424) (actual time=0.020..0.164 rows=420 loops=1000)
-> Select #2 (subquery in projection; dependent)
    -> Aggregate: count(distinct d.UserName)  (cost=205.66 rows=1) (actual time=0.454..0.454 rows=1 loops=1000)
        -> Covering index lookup on d using idx_desired_jobcategory (JobCategory=j.Category)  (cost=163.23 rows=424) (actual time=0.020..0.164 rows=420 loops=1000)
 |
```

2) Index on Company.CompanyId

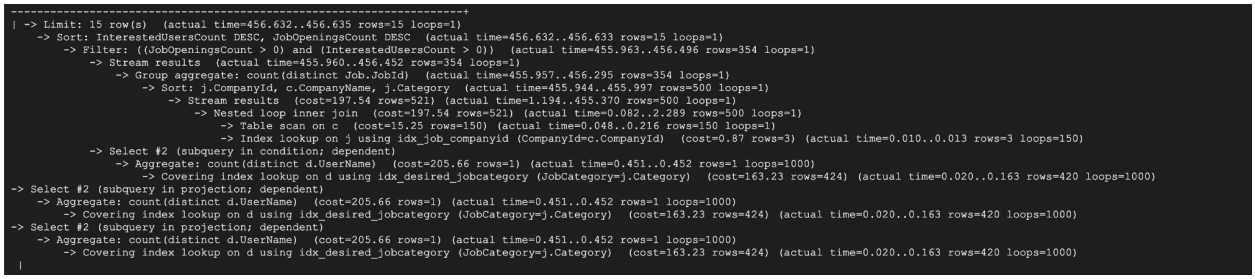
CREATE INDEX idx_company_companyid ON Company(CompanyId);



```
-> Limit: 15 row(s)  (actual time=454.661..454.663 rows=15 loops=1)
    -> Sort: InterestedUsersCount DESC, JobOpeningsCount DESC  (actual time=454.660..454.661 rows=15 loops=1)
        -> Filter: ((JobOpeningsCount > 0) and (InterestedUsersCount > 0))  (actual time=453.970..454.515 rows=354 loops=1)
            -> Stream results  (actual time=453.967..454.472 rows=354 loops=1)
                -> Group aggregate: count(distinct Job.JobId)  (actual time=453.963..454.303 rows=354 loops=1)
                    -> Sort: j.CompanyId, c.CompanyName, j.Category  (actual time=453.953..454.002 rows=500 loops=1)
                        -> Stream results  (cost=197.54 rows=521) (actual time=1.155..453.465 rows=500 loops=1)
                            -> Nested loop inner join  (cost=197.54 rows=521) (actual time=0.061..2.067 rows=500 loops=1)
                                -> Table scan on c  (cost=15.25 rows=150) (actual time=0.034..0.176 rows=150 loops=1)
                                -> Index lookup on j using idx_job_companyid (CompanyId=c.CompanyId)  (cost=0.87 rows=3) (actual time=0.010..0.012 rows=3 loops=150)
            -> Select #2 (subquery in condition; dependent)
                -> Aggregate: count(distinct d.UserName)  (cost=205.66 rows=1) (actual time=0.450..0.450 rows=1 loops=1000)
                    -> Covering index lookup on d using idx_desired_jobcategory (JobCategory=j.Category)  (cost=163.23 rows=424) (actual time=0.019..0.162 rows=420 loops=1000)
-> Select #2 (subquery in projection; dependent)
    -> Aggregate: count(distinct d.UserName)  (cost=205.66 rows=1) (actual time=0.450..0.450 rows=1 loops=1000)
        -> Covering index lookup on d using idx_desired_jobcategory (JobCategory=j.Category)  (cost=163.23 rows=424) (actual time=0.019..0.162 rows=420 loops=1000)
-> Select #2 (subquery in projection; dependent)
    -> Aggregate: count(distinct d.UserName)  (cost=205.66 rows=1) (actual time=0.450..0.450 rows=1 loops=1000)
        -> Covering index lookup on d using idx_desired_jobcategory (JobCategory=j.Category)  (cost=163.23 rows=424) (actual time=0.019..0.162 rows=420 loops=1000)
```

3) Index on Job.Category

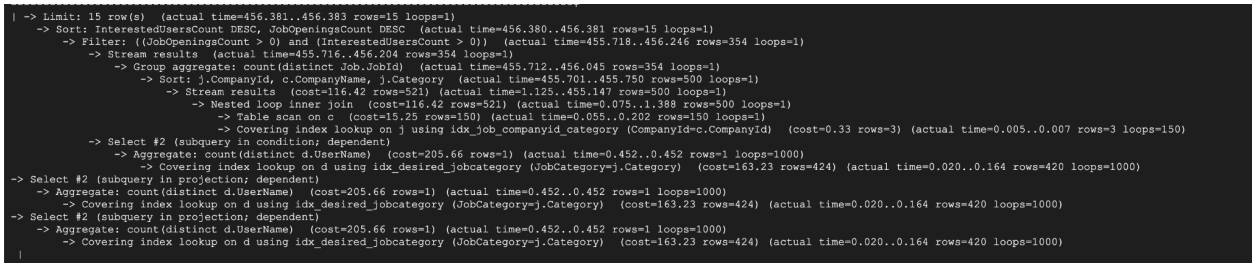
CREATE INDEX idx_job_category ON Job(Category);



```
-> Limit: 15 row(s)  (actual time=456.632..456.635 rows=15 loops=1)
    -> Sort: InterestedUsersCount DESC, JobOpeningsCount DESC  (actual time=456.632..456.633 rows=15 loops=1)
        -> Filter: ((JobOpeningsCount > 0) and (InterestedUsersCount > 0))  (actual time=455.963..456.496 rows=354 loops=1)
            -> Stream results  (actual time=455.960..456.452 rows=354 loops=1)
                -> Group aggregate: count(distinct Job.JobId)  (actual time=455.957..456.295 rows=354 loops=1)
                    -> Sort: j.CompanyId, c.CompanyName, j.Category  (actual time=455.944..455.997 rows=500 loops=1)
                        -> Stream results  (cost=197.54 rows=521) (actual time=1.194..455.370 rows=500 loops=1)
                            -> Nested loop inner join  (cost=197.54 rows=521) (actual time=0.082..2.289 rows=500 loops=1)
                                -> Table scan on c  (cost=15.25 rows=150) (actual time=0.048..0.216 rows=150 loops=1)
                                -> Index lookup on j using idx_job_companyid (CompanyId=c.CompanyId)  (cost=0.87 rows=3) (actual time=0.010..0.013 rows=3 loops=150)
            -> Select #2 (subquery in condition; dependent)
                -> Aggregate: count(distinct d.UserName)  (cost=205.66 rows=1) (actual time=0.451..0.452 rows=1 loops=1000)
                    -> Covering index lookup on d using idx_desired_jobcategory (JobCategory=j.Category)  (cost=163.23 rows=424) (actual time=0.020..0.163 rows=420 loops=1000)
-> Select #2 (subquery in projection; dependent)
    -> Aggregate: count(distinct d.UserName)  (cost=205.66 rows=1) (actual time=0.451..0.452 rows=1 loops=1000)
        -> Covering index lookup on d using idx_desired_jobcategory (JobCategory=j.Category)  (cost=163.23 rows=424) (actual time=0.020..0.163 rows=420 loops=1000)
-> Select #2 (subquery in projection; dependent)
    -> Aggregate: count(distinct d.UserName)  (cost=205.66 rows=1) (actual time=0.451..0.452 rows=1 loops=1000)
        -> Covering index lookup on d using idx_desired_jobcategory (JobCategory=j.Category)  (cost=163.23 rows=424) (actual time=0.020..0.163 rows=420 loops=1000)
```

4) Composite Index on Job.CompanyId, Job.Category

CREATE INDEX idx_job_companyid_category ON Job(CompanyId, Category);



```
-> Limit: 15 row(s)  (actual time=456.381..456.383 rows=15 loops=1)
    -> Sort: InterestedUsersCount DESC, JobOpeningsCount DESC  (actual time=456.380..456.381 rows=15 loops=1)
        -> Filter: ((JobOpeningsCount > 0) and (InterestedUsersCount > 0))  (actual time=455.718..456.246 rows=354 loops=1)
            -> Stream results  (actual time=455.716..456.204 rows=354 loops=1)
                -> Group aggregate: count(distinct Job.JobId)  (actual time=455.712..456.045 rows=354 loops=1)
                    -> Sort: j.CompanyId, c.CompanyName, j.Category  (actual time=455.701..455.750 rows=500 loops=1)
                        -> Stream results  (cost=116.42 rows=521) (actual time=1.125..455.147 rows=500 loops=1)
                            -> Nested loop inner join  (cost=116.42 rows=521) (actual time=0.075..1.388 rows=500 loops=1)
                                -> Table scan on c  (cost=15.25 rows=150) (actual time=0.055..0.202 rows=150 loops=1)
                                -> Covering index lookup on j using idx_job_companyid_category (CompanyId=c.CompanyId)  (cost=0.33 rows=3) (actual time=0.005..0.007 rows=3 loops=150)
            -> Select #2 (subquery in condition; dependent)
                -> Aggregate: count(distinct d.UserName)  (cost=205.66 rows=1) (actual time=0.452..0.452 rows=1 loops=1000)
                    -> Covering index lookup on d using idx_desired_jobcategory (JobCategory=j.Category)  (cost=163.23 rows=424) (actual time=0.020..0.164 rows=420 loops=1000)
-> Select #2 (subquery in projection; dependent)
    -> Aggregate: count(distinct d.UserName)  (cost=205.66 rows=1) (actual time=0.452..0.452 rows=1 loops=1000)
        -> Covering index lookup on d using idx_desired_jobcategory (JobCategory=j.Category)  (cost=163.23 rows=424) (actual time=0.020..0.164 rows=420 loops=1000)
-> Select #2 (subquery in projection; dependent)
    -> Aggregate: count(distinct d.UserName)  (cost=205.66 rows=1) (actual time=0.452..0.452 rows=1 loops=1000)
        -> Covering index lookup on d using idx_desired_jobcategory (JobCategory=j.Category)  (cost=163.23 rows=424) (actual time=0.020..0.164 rows=420 loops=1000)
```

Index configuration	Total cost	
No index	197.54	Initial query without indexes
Index on Job.CompanyId	197.54	Slight improvement in join
Index on Company.CompanyId	197.54	Slight improvement in join
Index on Job.Category	197.54	Helped optimize subquery

Composite index on Job.CompanyId, Job.Category	116.42	Best improvement seen here

Based on the cost reduction observed, the final index design selected is the composite index on Job.CompanyId and Job.Category. It provides the best performance improvement by reducing the cost significantly from 197.54 to 116.42. This composite index optimizes the join operation and the GROUP BY clause, leading to the lowest execution cost.