Project Track 1 Stage 3
Team 047 - Join_The_Party

Project Title: HandyIllinois - Connecting Homeowners with Reliable Handyperson Services

Database implementation :

1. Implementing the database tables on GCP:

```
talatitrusha88@cloudshell:~ (cs-411-team-047)$ gcloud sql connect handyillinois --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 21481
Server version: 8.0.31-google (Google)

Copyright (c) 2000, 2024, 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           |
+--------------------+
| HandyIllinois      |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.01 sec)
```

2. DDL Commands for the Tables:

    a. **Agency Table:**

```
CREATE TABLE Agency (
    AgencyId INT PRIMARY KEY AUTO_INCREMENT,
    Name VARCHAR(255) NOT NULL,
    HQLocation VARCHAR(255),
    Contact VARCHAR(255)
);
```

    b. **Customer Table:**

```
CREATE TABLE Customer (
    CustomerId INT PRIMARY KEY AUTO_INCREMENT,
    Name VARCHAR(255) NOT NULL,
    Address TEXT,
    ZIP VARCHAR(10),
    ContactNumber VARCHAR(255),
    Password VARCHAR(255),
    Email VARCHAR(255) UNIQUE,
    ProfileImage BLOB
);
```

    c. **Handyperson table:**

```
CREATE TABLE Handyperson (
    HandyId INT PRIMARY KEY AUTO_INCREMENT,
    Name VARCHAR(255) NOT NULL,
    Skills TEXT,
    Rating DECIMAL(3,1),
    Contact VARCHAR(255),
    ProfileImage BLOB,
    AgencyId INT,
    FOREIGN KEY (AgencyId) REFERENCES Agency(AgencyId) ON
DELETE CASCADE
);
```

d. **Manager table:**

```
CREATE TABLE Manager (
    ManagerId INT PRIMARY KEY AUTO_INCREMENT,
    ManagerName VARCHAR(255) NOT NULL,
    Contact VARCHAR(255),
    ProfileImage BLOB,
    AgencyId INT,
    FOREIGN KEY (AgencyId) REFERENCES Agency(AgencyId) ON
DELETE CASCADE
);
```

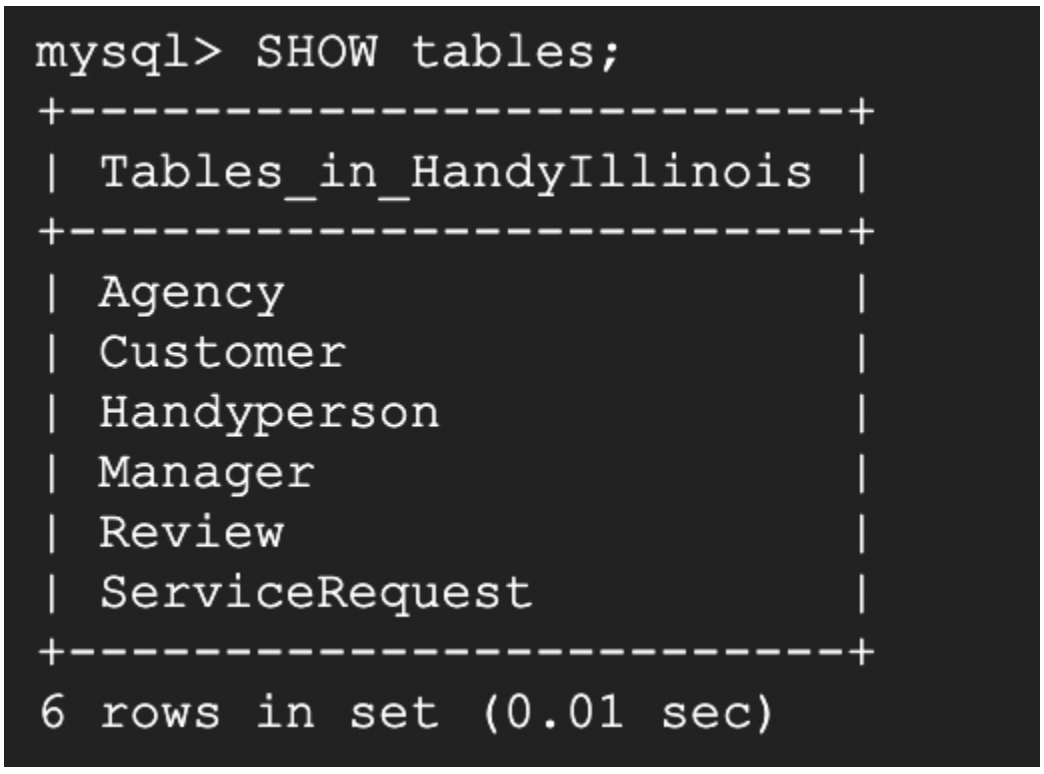
e. **Review table:**

```
CREATE TABLE Review (
    ReviewId INT PRIMARY KEY AUTO_INCREMENT,
    Comment TEXT,
    ReviewTitle VARCHAR(255),
    Rating DECIMAL(2,1),
    Date DATE,
    Time TIME,
    CustomerId INT,
    HandyId INT,
    ServiceRequestID INT,
    FOREIGN KEY (CustomerId) REFERENCES Customer(CustomerId)
ON DELETE CASCADE,
    FOREIGN KEY (HandyId) REFERENCES Handyperson(HandyId) ON
DELETE SET NULL,
    FOREIGN KEY (ServiceRequestID) REFERENCES
ServiceRequest(ServiceRequestId) ON DELETE CASCADE
);
```

f. **ServiceRequest Table:**

```
CREATE TABLE ServiceRequest (
    ServiceRequestId INT PRIMARY KEY AUTO_INCREMENT,
    Description TEXT,
    Date DATE,
```

```
    Time TIME,
    Status VARCHAR(50),
    Type VARCHAR(255),
    Charges DECIMAL(10,2),
    CustomerId INT,
    ManagerId INT,
    HandyId INT,
    FOREIGN KEY (CustomerId) REFERENCES Customer(CustomerId)
ON DELETE CASCADE,
    FOREIGN KEY (ManagerId) REFERENCES Manager(ManagerId) ON
DELETE SET NULL,
    FOREIGN KEY (HandyId) REFERENCES Handyperson(HandyId) ON
DELETE SET NULL
);
```



3. Count query:

Three tables - Customers, HandyPerson and ServiceRequest have a row count of 1000.

```
mysql> select count(*) from Customer;
+----------+
| count(*) |
+----------+
|     1000 |
+----------+
1 row in set (0.08 sec)

mysql> select count(*) from Handyperson;
+----------+
| count(*) |
+----------+
|     1000 |
+----------+
1 row in set (0.11 sec)

mysql> select count(*) from Manager;
+----------+
| count(*) |
+----------+
|       44 |
+----------+
1 row in set (0.02 sec)

mysql> select count(*) from Review;
+----------+
| count(*) |
+----------+
|      250 |
+----------+
1 row in set (0.00 sec)

mysql> select count(*) from ServiceRequest;
+----------+
| count(*) |
+----------+
|     1000 |
+----------+
1 row in set (0.01 sec)
```

Advanced Queries:

Query 1: Retrieve the Average Rating for Each Handyperson by Agency

This query finds the average rating for each Handyperson, grouped by Agency.

```
SELECT a.Name AS AgencyName, h.Name AS HandypersonName, AVG(r.Rating)
AS AvgRating
FROM Handyperson h
JOIN Agency a ON h.AgencyId = a.AgencyId
JOIN Review r ON h.HandyId = r.HandyId
GROUP BY a.Name, h.Name
ORDER BY AvgRating DESC;
```

```
mysql> SELECT a.Name AS AgencyName, h.Name AS HandypersonName, AVG(r.Rating) AS AvgRating
    -> FROM Handyperson h
    -> JOIN Agency a ON h.AgencyId = a.AgencyId
    -> JOIN Review r ON h.HandyId = r.HandyId
    -> GROUP BY a.Name, h.Name
    -> ORDER BY AvgRating DESC
    -> Limit 15;
+--------------------------+--------------------+-----------+
| AgencyName               | HandypersonName    | AvgRating |
+--------------------------+--------------------+-----------+
| Total Fix Network        | Tracy Jones        |   5.00000 |
| Complete In-Home Help    | Stephanie Fletcher |   4.00000 |
| Impact Care Solutions    | Jason Kelly        |   4.00000 |
| Pinnacle Home Help Plus  | William White      |   4.00000 |
| Halina's Handy Helpers   | Alexander Anderson |   4.00000 |
| A&N Handy Helpers        | David Smith        |   4.00000 |
| All Help Home Services   | Connor Jordan      |   4.00000 |
| Samaritan Senior Fixers  | Richard Peterson   |   4.00000 |
| Pinnacle Home Help Plus  | Sarah Cooper       |   4.00000 |
| Country Home Fixers      | Marcus Allen       |   4.00000 |
| Alpha Handyman Agency    | Glenn Newman       |   4.00000 |
| Alpha Handyman Agency    | Tyler Mckenzie     |   3.00000 |
| Your Helping Hand at Home | Stephen Allison    |   3.00000 |
| Oceangates Handy Services | Robert Scott       |   3.00000 |
| Eva's House Care         | Tracey Wheeler     |   3.00000 |
+--------------------------+--------------------+-----------+
15 rows in set (0.04 sec)
```

Result of the first 15 rows

Query 2: Retrieve Active Service Requests with Their Respective Managers and Agencies

This query lists active ServiceRequests along with the assigned Manager and their Agency, involving multiple joins.

```sql
SELECT sr.ServiceRequestId, sr.Description, sr.Date, sr.Status,
       m.ManagerName, a.Name AS AgencyName, c.Name AS CustomerName
FROM ServiceRequest sr
JOIN Manager m ON sr.ManagerId = m.ManagerId
JOIN Agency a ON m.AgencyId = a.AgencyId
JOIN Customer c ON sr.CustomerId = c.CustomerId
WHERE sr.Status = 'In Progress';
```

```
mysql> SELECT sr.ServiceRequestId, sr.Description, sr.Date, sr.Status,
    ->        m.ManagerName, a.Name AS AgencyName, c.Name AS CustomerName
    -> FROM ServiceRequest sr
    -> JOIN Manager m ON sr.ManagerId = m.ManagerId
    -> JOIN Agency a ON m.AgencyId = a.AgencyId
    -> JOIN Customer c ON sr.CustomerId = c.CustomerId
    -> WHERE sr.Status = 'In Progress'
    -> Limit 15;
+------------------+-------------------------------------------------------------------------------------------------------------------------------------------
------------+------------+--------------------------+---------------------------+------------------+
| ServiceRequestId | Description
 Date       | Status      | ManagerName              | AgencyName                | CustomerName     |
+------------------+-------------------------------------------------------------------------------------------------------------------------------------------
------------+------------+--------------------------+---------------------------+------------------+
|                1 | in apt 4-1 br#1, br#2, lv/rm repair leak at radiators. in apt # 3-1 bedroom #1: patch and plaster hole on ceiling around steam riser. in bedroom # 2
 2023-11-22 | In progress | Radnaasambuu Batdeleg    | Kind Fix Services         | Jamie Miller     |
|                2 | apt 2r 1. provide and install trhee new window guards. include all the necessary one way screws and stops. 2. make all the necessary repair to th
 2024-06-11 | In progress | Katrina Golden           | C&K House Repairs         | Samantha Mcguire  |
|                3 | (p/a) at 5th to 6th sty. stair case of intermediate landing a,b,c, & d side of 5th to 6th sty. install stop's. at 6th re-install w/g's and remove ben
 2024-02-27 | In progress | Jacqueline Lara Penarnada | Friendly Fixers          | Joshua Ellis     |
|                4 | (apt #. c9) leak affecting bdrm next to bthrm. cover bthrm floor with drop cloth. demo approx. (40 sq.ft.) ceiling. sheetrock ceiling (40 sq.ft.)
 2023-03-17 | In progress | Radnaasambuu Batdeleg    | Kind Fix Services         | Amanda Greene    |
|               11 | at apt. 56: re-install window guards in two(2) bedrooms.
 2024-07-09 | In progress | Dalia Makauskas          | Dalia's Home Solutions    | Kim Miller       |
|               17 | increase hw temp to within legal limits adjust controls provide adequate supply of hot water according to local codes of 120 degree at the time of
 2024-02-28 | In progress | Elizabeth Matysek        | Friendly Home Care        | Jacob Davis      |
|               27 | local law #1 violation : as per attached scope of work thoroughily remove all lead violations as per new york city administrative code Ã,Â§27-2056.11
 2023-09-23 | In progress | Jonathan Sison           | Almost Angels Assistance  | Kathleen Shields  |
|               28 | at apt. #2c) at bedrooms 1 and 2 provide and install 2 radiators complete. square off and repair the rotten areas at floor 6 sq. ft. each.
 2024-01-02 | In progress | Dalia Makauskas          | Dalia's Home Solutions    | Michelle Luna    |
|               31 | apt. #4a ( kitchen ) : replace defective deck faucet .
 2024-03-14 | In progress | Radnaasambuu Batdeleg    | Kind Fix Services         | Heidi George     |
|               35 | (apt. #2d) supply and install stops on (2) windows at livingroom. supply and install (2) window guards complete at bathroom (1) and bedroom (1). re
 2024-06-12 | In progress | Danuta Zajac             | Halina's Handy Helpers    | Julia Leonard    |
|               44 | apartment# 6e: bedroom# 1 reinstall 2 w/g's. bedroom# 2 reinstall 1 w/g. livingroom reinstall 1 w/g.
 2023-04-22 | In progress | Tumurmunkh Tuvshinbat    | Alpha Handyman Agency     | Amanda Johnson   |
|               50 | at apt #5a; at 2nd bathroom. reset and secure water closet bowl. (wc), Ã¢â,¬Å"remove all work related debrisÃ¢â,¬Â
```

Result of the first 12 rows

Query 3: Retrieve Service Requests Completed Within the Last Month, Grouped by Manager

This query lists the number of ServiceRequests completed in the past month, grouped by each Manager. It also displays the Manager's associated Agency.

```
SELECT m.ManagerId, m.ManagerName, a.Name AS AgencyName,
COUNT(sr.ServiceRequestId) AS CompletedRequests
FROM ServiceRequest sr
JOIN Manager m ON sr.ManagerId = m.ManagerId
JOIN Agency a ON m.AgencyId = a.AgencyId
WHERE sr.Status = 'Completed' AND sr.Date >= DATE_SUB(CURDATE(),
INTERVAL 1 MONTH)
GROUP BY m.ManagerId, m.ManagerName, a.Name;
```

```
mysql> SELECT m.ManagerId, m.ManagerName, a.Name AS AgencyName, COUNT(sr.ServiceRequestId) AS CompletedRequests
    -> FROM ServiceRequest sr
    -> JOIN Manager m ON sr.ManagerId = m.ManagerId
    -> JOIN Agency a ON m.AgencyId = a.AgencyId
    -> WHERE sr.Status = 'Completed' AND sr.Date >= DATE_SUB(CURDATE(), INTERVAL 1 MONTH)
    -> GROUP BY m.ManagerId, m.ManagerName, a.Name
    -> Limit 15;
+-----------+---------------------+---------------------------+------------------+
| ManagerId | ManagerName         | AgencyName                | CompletedRequests |
+-----------+---------------------+---------------------------+------------------+
|        15 | Dalia Makauskas     | Dalia's Home Solutions    |                2 |
|        12 | Isoken Ogbomo       | Complete In-Home Help     |                1 |
|        37 | Irene T Kul         | Polonia House Repair Agency |                1 |
|        11 | Mariola Piotrowski  | City & Suburban Fix-It Pros |                2 |
|         9 | Marie Levina        | Care & Repair Referral    |                1 |
|         8 | Katrina Golden      | C&K House Repairs         |                1 |
|        18 | Ralph Kowalski      | Eva's House Care          |                1 |
|         5 | Tumurmunkh Tuvshinbat | Alpha Handyman Agency     |                1 |
|         4 | Howard Snow         | All Help Home Services    |                1 |
+-----------+---------------------+---------------------------+------------------+
9 rows in set (0.01 sec)
```

***Result of the first 9 rows* (Due to number of managers being limited)**

Query 4: Calculate Total Revenue Generated by Each Handyperson

This query calculates the total revenue earned by each Handyperson based on the charges in their ServiceRequests.

```
SELECT h.HandyId, h.Name AS HandypersonName, SUM(sr.Charges) AS
TotalRevenue
FROM Handyperson h
JOIN ServiceRequest sr ON h.HandyId = sr.HandyId
GROUP BY h.HandyId, h.Name
ORDER BY TotalRevenue DESC;
```

```
mysql> SELECT h.HandyId, h.Name AS HandypersonName, SUM(sr.Charges) AS TotalRevenue
    -> FROM Handyperson h
    -> JOIN ServiceRequest sr ON h.HandyId = sr.HandyId
    -> GROUP BY h.HandyId, h.Name
    -> ORDER BY TotalRevenue DESC
    -> Limit 15;
+---------+------------------+--------------+
| HandyId | HandypersonName  | TotalRevenue |
+---------+------------------+--------------+
|     958 | Heather Hayes    |       637.00 |
|     394 | Diana Mclaughlin |       540.00 |
|     182 | Deborah Day      |       539.00 |
|     141 | Joseph Novak     |       538.00 |
|     918 | Donald Porter    |       499.00 |
|     547 | Evan Saunders    |       496.00 |
|     852 | Laura Carroll    |       493.00 |
|      46 | Joanne Armstrong |       468.00 |
|     436 | Matthew Davis    |       451.00 |
|     119 | Candace Duncan   |       450.00 |
|     797 | David Swanson    |       436.00 |
|     139 | Larry Jacobson   |       435.00 |
|      65 | Andrea Anderson  |       423.00 |
|     632 | April Gonzales   |       423.00 |
|     362 | Jeffrey Galloway |       412.00 |
+---------+------------------+--------------+
15 rows in set (0.02 sec)
```

Result of the first 15 rows

Indexing:

Query 1: Retrieve the Average Rating for Each Handyperson by Agency

This query finds the average rating for each Handyperson, grouped by Agency.

```
SELECT a.Name AS AgencyName, h.Name AS HandypersonName, AVG(r.Rating)
AS AvgRating
FROM Handyperson h
JOIN Agency a ON h.AgencyId = a.AgencyId
JOIN Review r ON h.HandyId = r.HandyId
GROUP BY a.Name, h.Name
```

```
ORDER BY AvgRating DESC;
```

Before Indexing:

```
| -> Sort: AvgRating DESC  (actual time=2.142..2.160 rows=221 loops=1)
    -> Table scan on <temporary>  (actual time=1.920..1.968 rows=221 loops=1)
        -> Aggregate using temporary table  (actual time=1.918..1.918 rows=221 loops=1)
            -> Nested loop inner join  (cost=207.66 rows=250) (actual time=0.128..1.450 rows=250 loops=1)
                -> Nested loop inner join  (cost=120.16 rows=250) (actual time=0.118..1.080 rows=250 loops=1)
                    -> Filter: (r.HandyId is not null)  (cost=28.75 rows=250) (actual time=0.098..0.251 rows=250 loops=1)
                        -> Table scan on r  (cost=28.75 rows=250) (actual time=0.096..0.227 rows=250 loops=1)
                    -> Filter: (h.AgencyId is not null)  (cost=0.27 rows=1) (actual time=0.003..0.003 rows=1 loops=250)
                        -> Single-row index lookup on h using PRIMARY (HandyId=r.HandyId)  (cost=0.27 rows=1) (actual time=0.003..0.003 rows=1 loops=250)
                -> Single-row index lookup on a using PRIMARY (AgencyId=h.AgencyId)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=250)
 |
```

After Indexing:

1. **Create index handy_person on Handyperson(AgencyId)**

```
| -> Sort: AvgRating DESC  (actual time=1.947..1.958 rows=221 loops=1)
    -> Table scan on <temporary>  (actual time=1.766..1.792 rows=221 loops=1)
        -> Aggregate using temporary table  (actual time=1.765..1.765 rows=221 loops=1)
            -> Nested loop inner join  (cost=203.75 rows=250) (actual time=0.111..1.215 rows=250 loops=1)
                -> Nested loop inner join  (cost=116.25 rows=250) (actual time=0.081..0.913 rows=250 loops=1)
                    -> Filter: (r.HandyId is not null)  (cost=28.75 rows=250) (actual time=0.070..0.174 rows=250 loops=1)
                        -> Table scan on r  (cost=28.75 rows=250) (actual time=0.069..0.155 rows=250 loops=1)
                    -> Filter: (h.AgencyId is not null)  (cost=0.25 rows=1) (actual time=0.003..0.003 rows=1 loops=250)
                        -> Single-row index lookup on h using PRIMARY (HandyId=r.HandyId)  (cost=0.25 rows=1) (actual time=0.003..0.003 rows=1 loops=250)
                -> Single-row index lookup on a using PRIMARY (AgencyId=h.AgencyId)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=250)
 |
```

Cost for both inner joins reduced from

207.66 -> 203.75

120.16 -> 116.25

2. **create index review_handyId on Review(HandyId);**

```
| -> Sort: AvgRating DESC  (actual time=3.679..3.692 rows=221 loops=1)
    -> Table scan on <temporary>  (actual time=3.559..3.588 rows=221 loops=1)
        -> Aggregate using temporary table  (actual time=3.557..3.557 rows=221 loops=1)
            -> Nested loop inner join  (cost=203.75 rows=250) (actual time=0.171..3.188 rows=250 loops=1)
                -> Nested loop inner join  (cost=116.25 rows=250) (actual time=0.164..2.866 rows=250 loops=1)
                    -> Filter: (r.HandyId is not null)  (cost=28.75 rows=250) (actual time=0.118..0.232 rows=250 loops=1)
                        -> Table scan on r  (cost=28.75 rows=250) (actual time=0.117..0.209 rows=250 loops=1)
                    -> Filter: (h.AgencyId is not null)  (cost=0.25 rows=1) (actual time=0.010..0.010 rows=1 loops=250)
                        -> Single-row index lookup on h using PRIMARY (HandyId=r.HandyId)  (cost=0.25 rows=1) (actual time=0.010..0.010 rows=1 loops=250)
                -> Single-row index lookup on a using PRIMARY (AgencyId=h.AgencyId)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=250)
 |
```

No change in cost

3. **create index agency_name on Agency(Name);**

```
| -> Sort: AvgRating DESC  (actual time=3.679..3.692 rows=221 loops=1)
    -> Table scan on <temporary>  (actual time=3.559..3.588 rows=221 loops=1)
        -> Aggregate using temporary table  (actual time=3.557..3.557 rows=221 loops=1)
            -> Nested loop inner join  (cost=203.75 rows=250) (actual time=0.171..3.188 rows=250 loops=1)
                -> Nested loop inner join  (cost=116.25 rows=250) (actual time=0.164..2.866 rows=250 loops=1)
                    -> Filter: (r.HandyId is not null)  (cost=28.75 rows=250) (actual time=0.118..0.232 rows=250 loops=1)
                        -> Table scan on r  (cost=28.75 rows=250) (actual time=0.117..0.209 rows=250 loops=1)
                    -> Filter: (h.AgencyId is not null)  (cost=0.25 rows=1) (actual time=0.010..0.010 rows=1 loops=250)
                        -> Single-row index lookup on h using PRIMARY (HandyId=r.HandyId)  (cost=0.25 rows=1) (actual time=0.010..0.010 rows=1 loops=250)
                -> Single-row index lookup on a using PRIMARY (AgencyId=h.AgencyId)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=250)
 |
```

No change in cost

Final index design:

1. **`Create index handy_person on Handyperson(AgencyId)`**

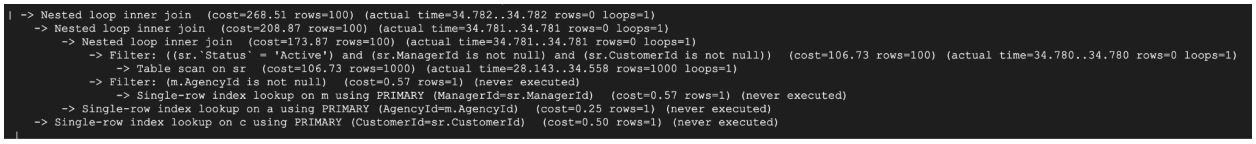
Creating a index on foreign key in handyperson table lead to better results for inner joins. But repeating this for other foreign keys did not yield any notable result. Indexing attributes in group by clause also did not yield any result.

Query 2: Retrieve Active Service Requests with Their Respective Managers and Agencies

This query lists active ServiceRequests along with the assigned Manager and their Agency, involving multiple joins.

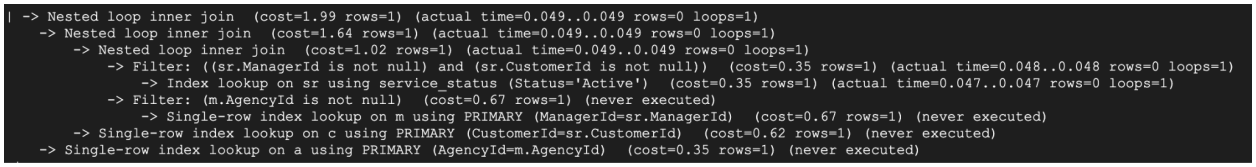
```
SELECT sr.ServiceRequestId, sr.Description, sr.Date, sr.Status,
       m.ManagerName, a.Name AS AgencyName, c.Name AS CustomerName
FROM ServiceRequest sr
JOIN Manager m ON sr.ManagerId = m.ManagerId
JOIN Agency a ON m.AgencyId = a.AgencyId
JOIN Customer c ON sr.CustomerId = c.CustomerId
WHERE sr.Status = 'In Progress';
```

Before Indexing:



After Indexing:

1. **`Create index service_status on ServiceRequest(Status);`**



Cost for all operations reduced to almost negligible values from

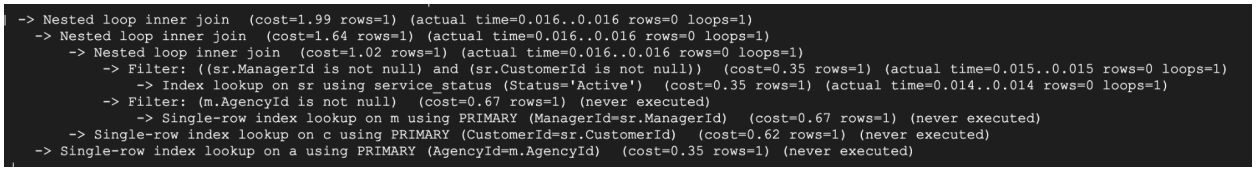
268.51 -> 1.99

208.87 -> 1.64

173.87 -> 1.02

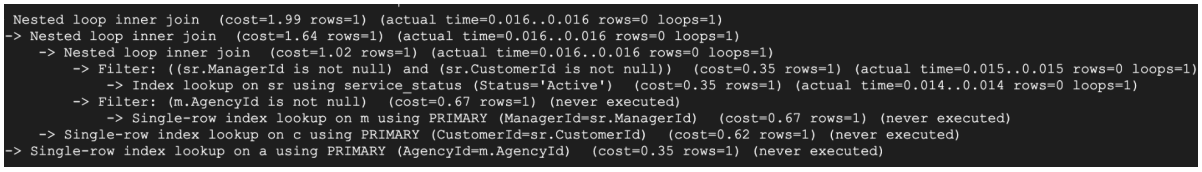
106.73 -> 0.35

2. **Create index service_managerid on ServiceRequest(ManagerId);**



No change in Cost

3. **Create index servicecustomerid on ServiceRequest(CustomerId);**



No change in cost

Final index design:

Create index service_status on ServiceRequest(Status);

Creating an index on Status attribute used in the where clause, substantially reduced cost in all operations. Further using indexes on foreign keys led to no change in performance.

Query 3: Retrieve Service Requests Completed Within the Last Month, Grouped by Manager

This query lists the number of ServiceRequests completed in the past month, grouped by each Manager. It also displays the Manager's associated Agency.

```
SELECT m.ManagerId, m.ManagerName, a.Name AS AgencyName,
COUNT(sr.ServiceRequestId) AS CompletedRequests
FROM ServiceRequest sr
JOIN Manager m ON sr.ManagerId = m.ManagerId
JOIN Agency a ON m.AgencyId = a.AgencyId
WHERE sr.Status = 'Completed' AND sr.Date >= DATE_SUB(CURDATE(),
INTERVAL 1 MONTH)
GROUP BY m.ManagerId, m.ManagerName, a.Name;
```

Before Indexing:

```
| -> Table scan on <temporary>  (actual time=0.848..0.850 rows=9 loops=1)
    -> Aggregate using temporary table  (actual time=0.848..0.848 rows=9 loops=1)
        -> Nested loop inner join  (cost=120.16 rows=33) (actual time=0.197..0.812 rows=11 loops=1)
            -> Nested loop inner join  (cost=108.50 rows=33) (actual time=0.190..0.788 rows=11 loops=1)
                -> Filter: ((sr.`Status` = 'Completed') and (sr.`Date` >= <cache>((curdate() - interval 1 month))) and (sr.ManagerId is not null))  (cost=96.83 rows=33) (actual time=0.129..0
.617 rows=11 loops=1)
                    -> Table scan on sr  (cost=96.83 rows=1000) (actual time=0.095..0.472 rows=1000 loops=1)
                -> Filter: (m.AgencyId is not null)  (cost=0.25 rows=1) (actual time=0.013..0.013 rows=1 loops=11)
                    -> Single-row index lookup on m using PRIMARY (ManagerId=sr.ManagerId)  (cost=0.25 rows=1) (actual time=0.013..0.013 rows=1 loops=11)
            -> Single-row index lookup on a using PRIMARY (AgencyId=m.AgencyId)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=11)
|
```

After Indexing:

1. **create index service_status on ServiceRequest(Status);**

```
| -> Table scan on <temporary>  (actual time=0.780..0.781 rows=9 loops=1)
    -> Aggregate using temporary table  (actual time=0.778..0.778 rows=9 loops=1)
        -> Nested loop inner join  (cost=77.16 rows=83) (actual time=0.438..0.735 rows=11 loops=1)
            -> Nested loop inner join  (cost=48.00 rows=83) (actual time=0.428..0.706 rows=11 loops=1)
                -> Filter: ((sr.`Date` >= <cache>((curdate() - interval 1 month))) and (sr.ManagerId is not null))  (cost=18.83 rows=83) (actual time=0.412..0.660 rows=11 loops=1)
                    -> Index lookup on sr using service_status (Status='Completed')  (cost=18.83 rows=250) (actual time=0.395..0.627 rows=250 loops=1)
                -> Filter: (m.AgencyId is not null)  (cost=0.25 rows=1) (actual time=0.004..0.004 rows=1 loops=11)
                    -> Single-row index lookup on m using PRIMARY (ManagerId=sr.ManagerId)  (cost=0.25 rows=1) (actual time=0.004..0.004 rows=1 loops=11)
            -> Single-row index lookup on a using PRIMARY (AgencyId=m.AgencyId)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=11)
|
```

Cost reduced from

120.16 -> **77.6**

108.50 -> **48**

96.83 -> **18.83**

2. **create index service_date on ServiceRequest(Date);**

```
| -> Table scan on <temporary>  (actual time=1.430..1.431 rows=9 loops=1)
    -> Aggregate using temporary table  (actual time=1.429..1.429 rows=9 loops=1)
        -> Nested loop inner join  (cost=26.51 rows=10) (actual time=1.199..1.390 rows=11 loops=1)
            -> Nested loop inner join  (cost=22.84 rows=10) (actual time=1.192..1.367 rows=11 loops=1)
                -> Filter: ((sr.`Status` = 'Completed') and (sr.ManagerId is not null))  (cost=19.16 rows=10) (actual time=1.177..1.327 rows=11 loops=1)
                    -> Index range scan on sr using service_date over ('2024-09-30' <= Date), with index condition: (sr.`Date` >= <cache>((curdate() - interval 1 month)))  (cost=19.16 rows=4
2) (actual time=0.046..0.189 rows=42 loops=1)
                -> Filter: (m.AgencyId is not null)  (cost=0.26 rows=1) (actual time=0.003..0.003 rows=1 loops=11)
                    -> Single-row index lookup on m using PRIMARY (ManagerId=sr.ManagerId)  (cost=0.26 rows=1) (actual time=0.003..0.003 rows=1 loops=11)
            -> Single-row index lookup on a using PRIMARY (AgencyId=m.AgencyId)  (cost=0.26 rows=1) (actual time=0.002..0.002 rows=1 loops=11)
|
```

Cost reduced from

77.6 -> **26.51**

48 -> **22.84**

3. **Create index manager_agencyId on Manager(AgencyId);**

```
| -> Table scan on <temporary>  (actual time=0.659..0.660 rows=9 loops=1)
    -> Aggregate using temporary table  (actual time=0.657..0.657 rows=9 loops=1)
        -> Nested loop inner join  (cost=26.51 rows=10) (actual time=0.132..0.626 rows=11 loops=1)
            -> Nested loop inner join  (cost=22.84 rows=10) (actual time=0.125..0.604 rows=11 loops=1)
                -> Filter: ((sr.`Status` = 'Completed') and (sr.ManagerId is not null))  (cost=19.16 rows=10) (actual time=0.113..0.570 rows=11 loops=1)
                    -> Index range scan on sr using service_date over ('2024-09-30' <= Date), with index condition: (sr.`Date` >= <cache>((curdate() - interval 1 month)))  (cost=19.16 rows=4
2) (actual time=0.082..0.533 rows=42 loops=1)
                -> Filter: (m.AgencyId is not null)  (cost=0.26 rows=1) (actual time=0.003..0.003 rows=1 loops=11)
                    -> Single-row index lookup on m using PRIMARY (ManagerId=sr.ManagerId)  (cost=0.26 rows=1) (actual time=0.003..0.003 rows=1 loops=11)
            -> Single-row index lookup on a using PRIMARY (AgencyId=m.AgencyId)  (cost=0.26 rows=1) (actual time=0.002..0.002 rows=1 loops=11)
|
```

No change in cost

Final index design:

1. **create index service_status on ServiceRequest(Status);**
2. **create index service_date on ServiceRequest(Date);**

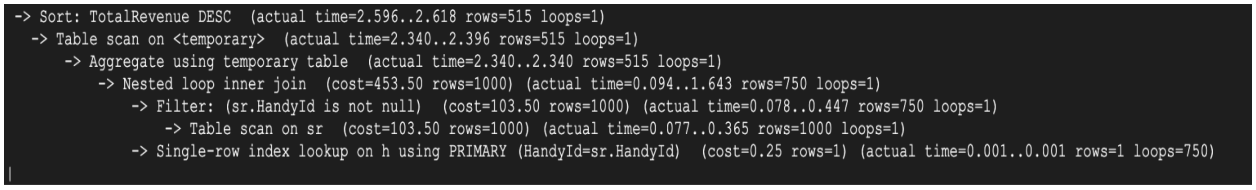
Creating an index on Status attribute used in the where clause, substantially reduced cost in all operations. Further using an index on Date attribute of ServiceRequest led to further optimization of the query leading to reduced cost as shown above.

Query 4: Calculate Total Revenue Generated by Each Handyperson

This query calculates the total revenue earned by each Handyperson based on the charges in their ServiceRequests.

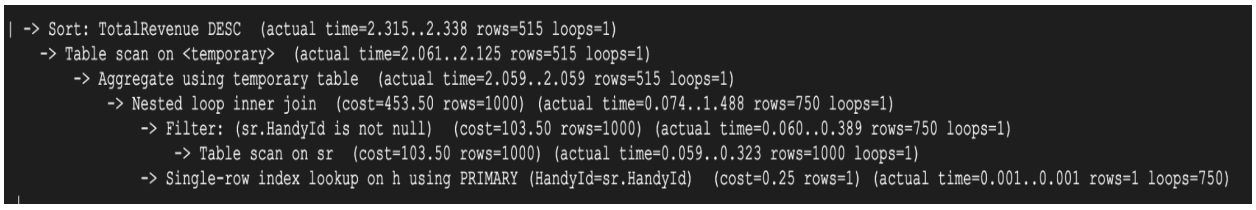
```
SELECT h.HandyId, h.Name AS HandypersonName, SUM(sr.Charges) AS
TotalRevenue
FROM Handyperson h
JOIN ServiceRequest sr ON h.HandyId = sr.HandyId
GROUP BY h.HandyId, h.Name
ORDER BY TotalRevenue DESC;
```

Before Indexing:



After Indexing:

1. **create index service_charges on ServiceRequest(Charges);**



No change in cost

2. **create index service_handyid on ServiceRequest(HandyId);**

```
-> Sort: TotalRevenue DESC  (actual time=2.626..2.649 rows=515 loops=1)
  -> Table scan on <temporary>  (actual time=2.303..2.399 rows=515 loops=1)
     -> Aggregate using temporary table  (actual time=2.302..2.302 rows=515 loops=1)
        -> Nested loop inner join  (cost=453.50 rows=1000) (actual time=0.080..1.673 rows=750 loops=1)
           -> Filter: (sr.HandyId is not null)  (cost=103.50 rows=1000) (actual time=0.063..0.420 rows=750 loops=1)
              -> Table scan on sr  (cost=103.50 rows=1000) (actual time=0.062..0.353 rows=1000 loops=1)
           -> Single-row index lookup on h using PRIMARY (HandyId=sr.HandyId)  (cost=0.25 rows=1) (actual time=0.001..0.002 rows=1 loops=750)
```

No change in cost

3. **create index handy_Names on Handyperson(Name);**

```
-> Sort: TotalRevenue DESC  (actual time=4.491..4.513 rows=515 loops=1)
  -> Table scan on <temporary>  (actual time=3.501..3.567 rows=515 loops=1)
     -> Aggregate using temporary table  (actual time=3.498..3.498 rows=515 loops=1)
        -> Nested loop inner join  (cost=453.50 rows=1000) (actual time=1.041..2.276 rows=750 loops=1)
           -> Filter: (sr.HandyId is not null)  (cost=103.50 rows=1000) (actual time=1.017..1.241 rows=750 loops=1)
              -> Covering index scan on sr using idx_sr_handyid_charges  (cost=103.50 rows=1000) (actual time=0.076..0.307 rows=1000 loops=1)
           -> Single-row index lookup on h using PRIMARY (HandyId=sr.HandyId)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=750)
```

No change in cost

Final index design:

1. **create index service_charges on ServiceRequest(Charges);**
2. **create index service_handyid on ServiceRequest(HandyId);**

Creating an index does not change the cost because of Low Cardinality and overhead from small tables. For small tables overhead of using index might make the query slower.