Database Implementation (GCP):
Connection:

```
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to credible-skill-440221-n5.
Use “gcloud config set project [PROJECT_ID]” to change to a different project.
apulkit0622@cloudshell:~ (credible-skill-440221-n5)$ gcloud sql connect orange-team --user=root --quiet
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 2091
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> use uniranker;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SHOW TABLES
    -> ;
+---------------------+
| Tables_in_uniranker |
+---------------------+
| Choices             |
| Favorites           |
| Major               |
| University          |
| User                |
+---------------------+
5 rows in set (0.00 sec)
```

Tables (DDL Commands):

```
-- University Table
CREATE TABLE University (
  InstitutionName VARCHAR(100) PRIMARY KEY,
  Quality INT,
  Research INT,
  Alumni INT
);


-- Major Table
CREATE TABLE Major (
  MajorID INT PRIMARY KEY,
  Major VARCHAR(100),
  Employment INT,
  Median INT
);


-- User Table
```

```
CREATE TABLE User (
  UserID INT PRIMARY KEY,
  Password VARCHAR(50),
  Name VARCHAR(100),
  MajorID INT,
  FOREIGN KEY (MajorID) REFERENCES Major(MajorID)
);


-- Choices Table
CREATE TABLE Choices (
  ChoiceID INT PRIMARY KEY,
  Quality INT,
  Research INT,
  Alumni INT,
  FirstChoice VARCHAR(100),
  SecondChoice VARCHAR(100),
  ThirdChoice VARCHAR(100)
);


-- Favorites Table
CREATE TABLE Favorites (
  FavoriteID INT PRIMARY KEY,
  ChoiceID INT,
  UserID INT,
  Date DATE,
  FOREIGN KEY (ChoiceID) REFERENCES Choices(ChoiceID),
  FOREIGN KEY (UserID) REFERENCES User(UserID)
);
```

Inserting 1000 Rows (Screenshot of Counts of 1000+ record tables):

```
mysql> SELECT COUNT(*) FROM User;
+----------+
| COUNT(*) |
+----------+
|     1001 |
+----------+
1 row in set (0.02 sec)

mysql> SELECT COUNT(*) FROM University
    -> ;
+----------+
| COUNT(*) |
+----------+
|     1001 |
+----------+
1 row in set (0.00 sec)

mysql> SELECT COUNT(*) FROM Favorites
    -> ;
+----------+
| COUNT(*) |
+----------+
|     2010 |
+----------+
1 row in set (0.00 sec)

mysql> SELECT COUNT(*) FROM Major
    -> ;
+----------+
| COUNT(*) |
+----------+
|       11 |
+----------+
1 row in set (0.00 sec)

mysql> SELECT COUNT(*) FROM Choices;
+----------+
| COUNT(*) |
+----------+
|        8 |
+----------+
1 row in set (0.01 sec)
```

Advanced Queries:
Advanced Query 1:
Query Explanation: This query finds institutions from the University table who have higher than average research, quality, and alumni scores.
Advanced Aspects: Subqueries, Set Operations

```
(SELECT InstitutionName
FROM University
WHERE Research > (SELECT AVG(Research) FROM University))
INTERSECT
(SELECT InstitutionName
FROM University
WHERE Quality > (SELECT AVG(Quality) FROM University))
INTERSECT
(SELECT InstitutionName
FROM University
WHERE Alumni > (SELECT AVG(Alumni) FROM University));
```

```
mysql> (SELECT InstitutionName
    -> FROM University
    -> WHERE Research > (SELECT AVG(Research) FROM University))
    -> INTERSECT
    -> (SELECT InstitutionName
    -> FROM University
    -> WHERE Quality > (SELECT AVG(Quality) FROM University))
    -> INTERSECT
    -> (SELECT InstitutionName
    -> FROM University
    -> WHERE Alumni > (SELECT AVG(Alumni) FROM University))
    -> LIMIT 15;
+---------------------------------+
| InstitutionName                 |
+---------------------------------+
| Aaron Rivas University          |
| Aimee Lawrence University       |
| Alec Wheeler University         |
| Alexander Cherry University     |
| Alexander Oconnor University    |
| Alicia Jackson University       |
| Alicia Watkins University       |
| Amanda Davenport University     |
| Amanda Fritz University         |
| Amber Clark University          |
| Amber Short University          |
| Amy Soto University             |
| Andre White University          |
| Andrew Robertson University     |
| Angelica Rodriguez MD University |
+---------------------------------+
15 rows in set (0.01 sec)
```

Advanced Query 2:
Query Explanation: This query finds the number of users enrolled in each major and the average median salary associated with those majors.
Advanced Aspects: Joins multiple relations, Group By
There are 10 total majors to display, so less than 15.

```
SELECT m.Major, COUNT(u.UserID) AS StudentCount, AVG(m.Median) AS AvgMedianSalary
FROM User u
JOIN Major m ON u.MajorID = m.MajorID
WHERE m.MajorID != 0
GROUP BY m.Major
ORDER BY AvgMedianSalary DESC;
```

```
mysql> SELECT m.Major, COUNT(u.UserID) AS StudentCount, AVG(m.Median) AS AvgMedianSalary
    -> FROM User u
    -> JOIN Major m ON u.MajorID = m.MajorID
    -> WHERE m.MajorID != 0
    -> GROUP BY m.Major
    -> ORDER BY AvgMedianSalary DESC;
+------------------------+--------------+-----------------+
| Major                  | StudentCount | AvgMedianSalary |
+------------------------+--------------+-----------------+
| Mathematics            |           96 |     234675.0000 |
| Economics              |           94 |     212540.0000 |
| Electrical Engineering |          101 |     137232.0000 |
| Civil Engineering      |           91 |     133933.0000 |
| Computer Science       |           95 |     122666.0000 |
| Psychology             |          111 |      90640.0000 |
| Physics                |           93 |      74050.0000 |
| Biology                |          105 |      59622.0000 |
| Chemistry              |          107 |      56535.0000 |
| Mechanical Engineering |          107 |      46037.0000 |
+------------------------+--------------+-----------------+
10 rows in set (0.01 sec)
```

Advanced Query 3:
Query Explanation: This query finds the most recent first choice university for each user in the User table and returns the respective major, number of students that have this university as a first choice and the median salary, grouped by major.

Advanced Aspects: Joins multiple relations, uses subqueries

```
SELECT U.UserID, U.Name, C.FirstChoice, F.Date
FROM User U
JOIN Favorites F ON U.UserID = F.UserID
JOIN Choices C ON F.ChoiceID = C.ChoiceID
WHERE F.Date = (
  SELECT MAX(Date)
  FROM Favorites F2
  WHERE F2.UserID = U.UserID
);
```

```
mysql> SELECT U.UserID, U.Name, C.FirstChoice, F.Date
    -> FROM User U
    -> JOIN Favorites F ON U.UserID = F.UserID
    -> JOIN Choices C ON F.ChoiceID = C.ChoiceID
    -> WHERE F.Date = (
    ->   SELECT MAX(Date)
    ->   FROM Favorites F2
    ->   WHERE F2.UserID = U.UserID
    -> )
    -> LIMIT 15;
+--------+-----------------------+-----------------------+------------+
| UserID | Name                  | FirstChoice           | Date       |
+--------+-----------------------+-----------------------+------------+
|      5 | Amanda Jones          | Teresa Choi University | 2024-06-15 |
|     17 | Alyssa Barnes         | Teresa Choi University | 2024-08-19 |
|     22 | Carmen Thompson MD    | Teresa Choi University | 2024-03-19 |
|     35 | Stacie Young          | Teresa Choi University | 2024-08-20 |
|     53 | Michael Anderson      | Teresa Choi University | 2024-10-11 |
|     57 | Kerry Lopez           | Teresa Choi University | 2024-02-18 |
|     61 | Bryan Bernard         | Teresa Choi University | 2024-05-19 |
|     64 | Patrick Wise          | Teresa Choi University | 2024-03-30 |
|     94 | Ryan Williams         | Teresa Choi University | 2024-09-02 |
|    100 | Jackie Neal           | Teresa Choi University | 2024-10-22 |
|    102 | Carmen Owens          | Teresa Choi University | 2024-06-30 |
|    114 | Jonathan Mcfarland Jr. | Teresa Choi University | 2024-06-23 |
|    115 | Daniel Lee            | Teresa Choi University | 2024-08-18 |
|    129 | Mark Scott            | Teresa Choi University | 2024-10-15 |
|    130 | Shelly Mccarty        | Teresa Choi University | 2024-09-23 |
+--------+-----------------------+-----------------------+------------+
15 rows in set, 1 warning (0.01 sec)
```

Advanced Query 4:
Query Explanation: This query orders majors based on the amount of favorites users of that major have created, with majors that have created more favorites being at the top.
Advanced Aspects: Joins multiple relations, Group By
There are 10 total majors to display, so less than 15.

```
SELECT M.Major, COUNT(*) AS FavoriteCount
FROM Major M
JOIN User U ON M.MajorID = U.MajorID
JOIN Favorites F ON U.UserID = F.UserID
WHERE M.MajorID != 0
GROUP BY M.MajorID
ORDER BY FavoriteCount DESC;
```

```
mysql> SELECT M.Major, COUNT(*) AS FavoriteCount
    -> FROM Major M
    -> JOIN User U ON M.MajorID = U.MajorID
    -> JOIN Favorites F ON U.UserID = F.UserID
    -> WHERE M.MajorID != 0
    -> GROUP BY M.MajorID
    -> ORDER BY FavoriteCount DESC;
+------------------------+---------------+
| Major                  | FavoriteCount |
+------------------------+---------------+
| Chemistry              |           229 |
| Biology                |           227 |
| Psychology             |           214 |
| Mechanical Engineering |           208 |
| Electrical Engineering |           202 |
| Computer Science       |           197 |
| Economics              |           187 |
| Mathematics            |           183 |
| Civil Engineering      |           182 |
| Physics                |           180 |
+------------------------+---------------+
10 rows in set (0.01 sec)
```

Indexing Analysis:
Link to our outputs from explain analyze: LINK

Query 1 Indexing Analysis:
For this query, we decided to use Research, Quality, and Alumni as potential columns for indexing, since they are involved in the WHERE clauses. Our original query without any indexing had a cost of 137.22 for the table scan on the intersection temporary table. With indexing on Research, the table scan cost increased to 223.78. Using indexing on Quality, the table scan cost was 223.08, and using indexing on Alumni, the table scan cost was 222.37. After comparing these indexing options to the original query, we can see that without indexing, the original query performed a lot more efficiently with lower costs. This could have been because of the increased overhead from using indexing which degraded our queries performance. Therefore, for this query, we have decided against using indexing.

Query 2 Indexing Analysis:
For this query we decided to use Major, MajorID, and (Major, Median) as potential indexing options as they are involved in the JOIN and GROUP BY clauses. For our original query, our inner loop join had a cost of 6.56, and our filter had a cost of 2.71. For our Major indexed query, our inner loop join had a cost of 105.83, and our filter had a cost of 2.71. For our MajorID indexed query, our inner loop join had a cost of 105.83, and our filter had a cost of 2.71. For our (Major, Median) indexed query, our inner loop join had a cost of 104.47, and our filter had a cost of 1.35. Based on these numbers, it is clear that our original query without any further indexing had a lower cost than the indexing options we tried. This could have been because of the increased overhead from using indexing which degraded our queries performance. As a result, we have decided not to use further indexing in this case.

Query 3 Indexing Analysis:
For this query we used User.Name, Favorites.Date, Choices.FirstChoice as potential indexing columns, since they are attributes we use for join and where operations in the advanced queries. Explain analysis on the original query produced an inner loop join cost of 947.55, a table scan cost of 1.05, and a filter cost of 8.39. For each of the indexing options, we found no changes to the costs of any of these query aspects, thus indexing had no effect on the query. This is expected as the query includes smaller datasets, and the majority of the cost of this advanced query is the subquery structure involved. Indexing was ultimately not an effective choice to improve cost output.

Query 4 Indexing Analysis:
For this query we decided to use (MajorID, UserID), (MajorID, Major), and UserID as potential columns for indexing, since they are involved in the JOINS and SELECT clauses. The original query had an inner loop join cost of 1609.75. After performing the specified indexing, the overall cost remained the same as 1609.75. Therefore, since there was no change in costs, the indexing had no effects on the query performance, showing how indexing wasn’t very effective for this query’s purpose.