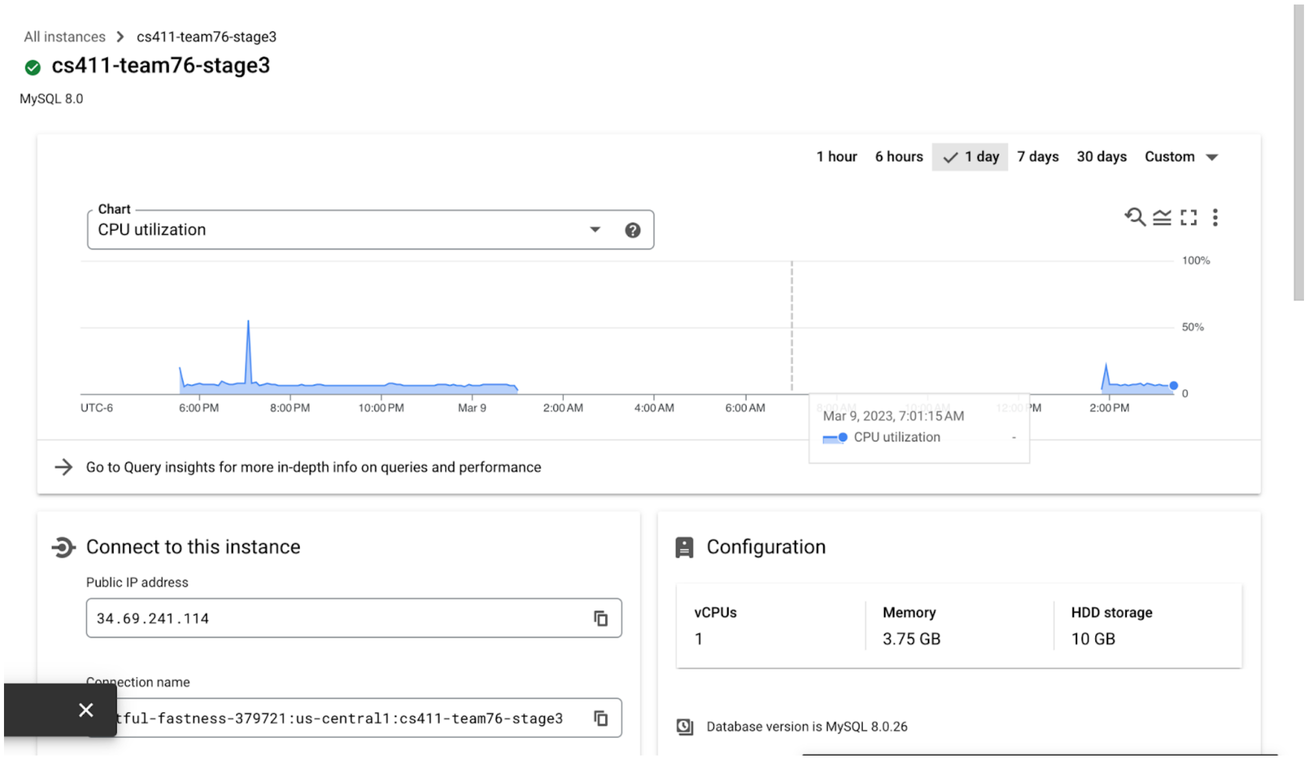
Step 1. Implement at least four main tables (i.e., tables that include core application information, not auxiliary information, such as user profiles and login information).

The database is hosted on GCP MySQL v8

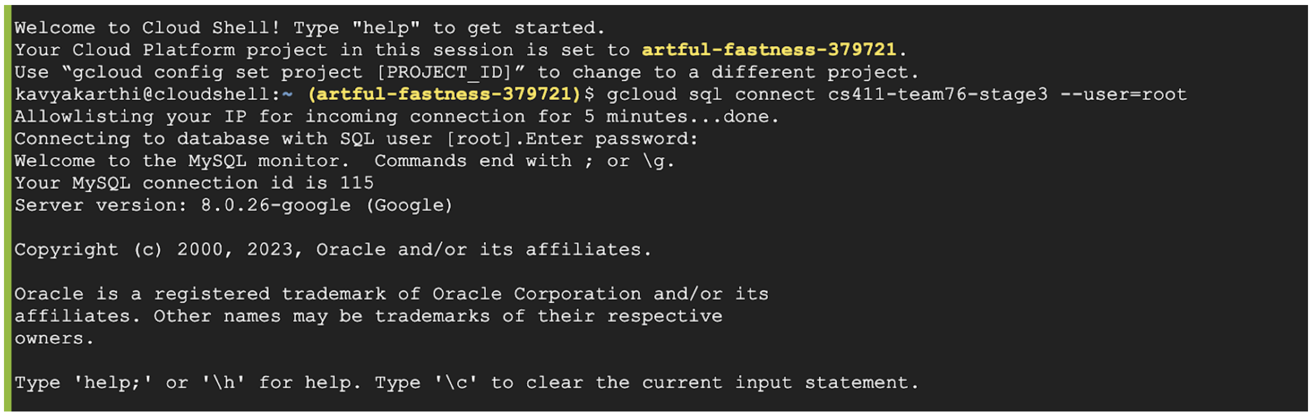


Public IP address: 34.69.241.114

Connection name: tful-fastness-379721:us-central1:cs411-team76-stage3

vCPUs	Memory	HDD storage
1	3.75 GB	10 GB

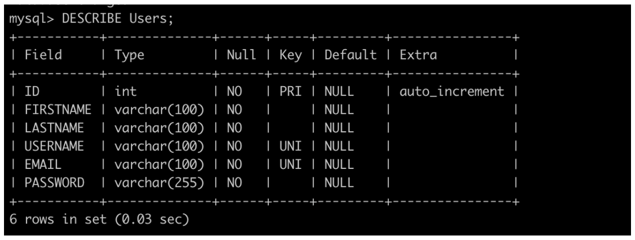
Database version is MySQL 8.0.26



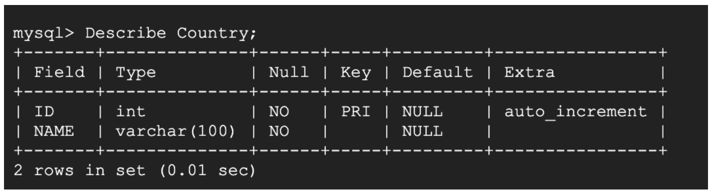
```
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to artful-fastness-379721.
Use “gcloud config set project [PROJECT_ID]” to change to a different project.
kavyakarthi@cloudshell:~ (artful-fastness-379721)$ gcloud sql connect cs411-team76-stage3 --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 115
Server version: 8.0.26-google (Google)

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

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

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

A total of 8 tables were implemented, with one of 8 being the auxiliary DB for login - Users.

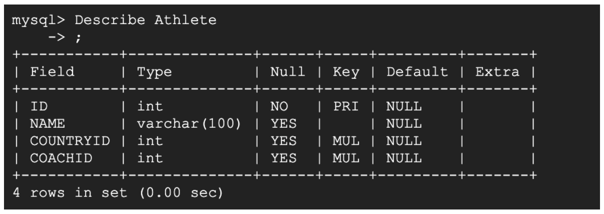
```
+----------------+
| Tables_in_CS411 |
+----------------+
| Athlete        |
| Coach          |
| Country        |
| Medals         |
| Plays          |
| Sport          |
| Users          |
| schedule       |
+----------------+
```



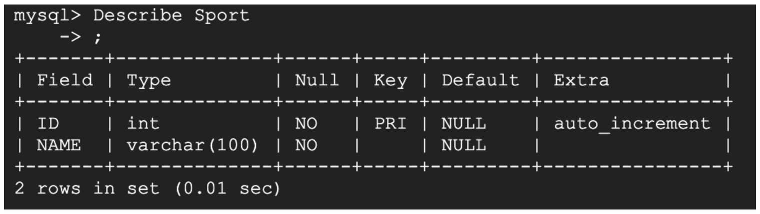
```
mysql> DESCRIBE Users;
+-----------+--------------+------+-----+---------+----------------+
| Field     | Type         | Null | Key | Default | Extra          |
+-----------+--------------+------+-----+---------+----------------+
| ID        | int          | NO   | PRI | NULL    | auto_increment |
| FIRSTNAME | varchar(100) | NO   |     | NULL    |                |
| LASTNAME  | varchar(100) | NO   |     | NULL    |                |
| USERNAME  | varchar(100) | NO   | UNI | NULL    |                |
| EMAIL     | varchar(100) | NO   | UNI | NULL    |                |
| PASSWORD  | varchar(255) | NO   |     | NULL    |                |
+-----------+--------------+------+-----+---------+----------------+
6 rows in set (0.03 sec)
```



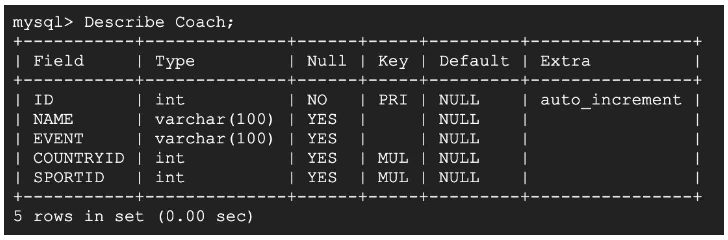
```
mysql> Describe Country;
+-------+--------------+------+-----+---------+----------------+
| Field | Type         | Null | Key | Default | Extra          |
+-------+--------------+------+-----+---------+----------------+
| ID    | int          | NO   | PRI | NULL    | auto_increment |
| NAME  | varchar(100) | NO   |     | NULL    |                |
+-------+--------------+------+-----+---------+----------------+
2 rows in set (0.01 sec)
```



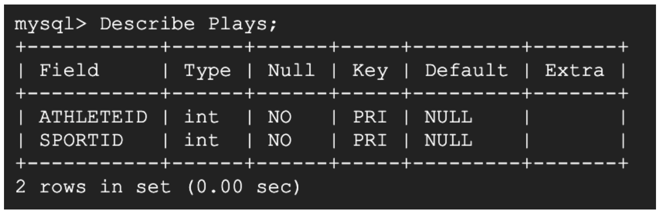
```
mysql> Describe Athlete
    -> ;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| ID        | int          | NO   | PRI | NULL    |       |
| NAME      | varchar(100) | YES  |     | NULL    |       |
| COUNTRYID | int          | YES  | MUL | NULL    |       |
| COACHID   | int          | YES  | MUL | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```



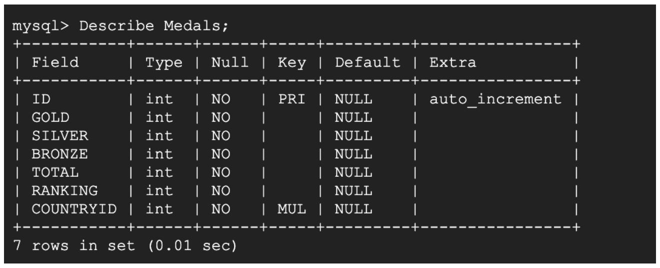
```
mysql> Describe Sport
    -> ;
+-------+--------------+------+-----+---------+----------------+
| Field | Type         | Null | Key | Default | Extra          |
+-------+--------------+------+-----+---------+----------------+
| ID    | int          | NO   | PRI | NULL    | auto_increment |
| NAME  | varchar(100) | NO   |     | NULL    |                |
+-------+--------------+------+-----+---------+----------------+
2 rows in set (0.01 sec)
```



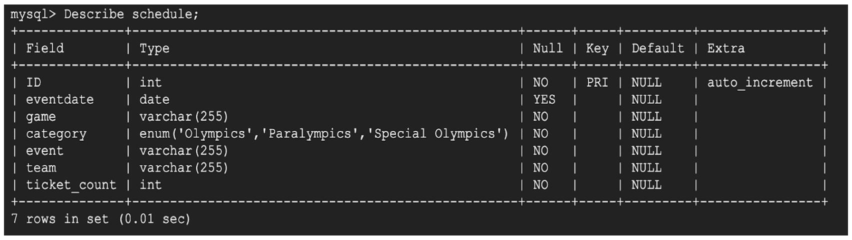
```
mysql> Describe Coach;
+-----------+--------------+------+-----+---------+----------------+
| Field     | Type         | Null | Key | Default | Extra          |
+-----------+--------------+------+-----+---------+----------------+
| ID        | int          | NO   | PRI | NULL    | auto_increment |
| NAME      | varchar(100) | YES  |     | NULL    |                |
| EVENT     | varchar(100) | YES  |     | NULL    |                |
| COUNTRYID | int          | YES  | MUL | NULL    |                |
| SPORTID   | int          | YES  | MUL | NULL    |                |
+-----------+--------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)
```



```
mysql> Describe Plays;
+-----------+------+------+-----+---------+-------+
| Field     | Type | Null | Key | Default | Extra |
+-----------+------+------+-----+---------+-------+
| ATHLETEID | int  | NO   | PRI | NULL    |       |
| SPORTID   | int  | NO   | PRI | NULL    |       |
+-----------+------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```



```
mysql> Describe Medals;
+-----------+------+------+-----+---------+----------------+
| Field     | Type | Null | Key | Default | Extra          |
+-----------+------+------+-----+---------+----------------+
| ID        | int  | NO   | PRI | NULL    | auto_increment |
| GOLD      | int  | NO   |     | NULL    |                |
| SILVER    | int  | NO   |     | NULL    |                |
| BRONZE    | int  | NO   |     | NULL    |                |
| TOTAL     | int  | NO   |     | NULL    |                |
| RANKING   | int  | NO   |     | NULL    |                |
| COUNTRYID | int  | NO   | MUL | NULL    |                |
+-----------+------+------+-----+---------+----------------+
7 rows in set (0.01 sec)
```



```
mysql> Describe schedule;
+--------------+-----------------------------------------------------+------+-----+---------+----------------+
| Field        | Type                                                | Null | Key | Default | Extra          |
+--------------+-----------------------------------------------------+------+-----+---------+----------------+
| ID           | int                                                 | NO   | PRI | NULL    | auto_increment |
| eventdate    | date                                                | YES  |     | NULL    |                |
| game         | varchar(255)                                        | NO   |     | NULL    |                |
| category     | enum('Olympics','Paralympics','Special Olympics')   | NO   |     | NULL    |                |
| event        | varchar(255)                                        | NO   |     | NULL    |                |
| team         | varchar(255)                                        | NO   |     | NULL    |                |
| ticket_count | int                                                 | NO   |     | NULL    |                |
+--------------+-----------------------------------------------------+------+-----+---------+----------------+
7 rows in set (0.01 sec)
```

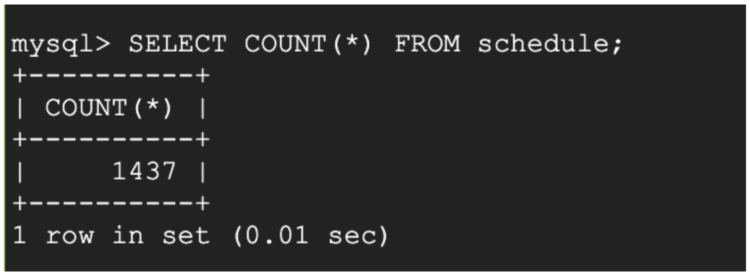
Step 2. In the Database Design markdown or pdf, provide the Data Definition Language (DDL) commands you all used to create each of these tables in the database. Here's the syntax of the CREATE TABLE DDL command:CREATE TABLE table_name (column1 datatype, column2 datatype, column3 datatype,...);

```
CREATE TABLE Sport (
   ID INT  AUTO_INCREMENT PRIMARY KEY,
   NAME VARCHAR(100)
);

CREATE TABLE Country (
   ID INT  AUTO_INCREMENT PRIMARY KEY,
   NAME VARCHAR(100)
);

CREATE TABLE Coach (
   ID INT  AUTO_INCREMENT PRIMARY KEY,
   NAME VARCHAR(100),
   EVENT VARCHAR(100),
   COUNTRYID INT,
   SPORTID INT,
   FOREIGN KEY (COUNTRYID) REFERENCES Country(ID),
   FOREIGN KEY (SPORTID) REFERENCES Sport(ID)
);

CREATE TABLE Athlete (
   ID INT PRIMARY KEY,
   NAME VARCHAR(100) ,
   COUNTRYID INT,
   COACHID INT,
   FOREIGN KEY (COUNTRYID) REFERENCES Country(ID),
   FOREIGN KEY (COACHID) REFERENCES Coach(ID)
);

CREATE TABLE Medals (
   ID INT  AUTO_INCREMENT PRIMARY KEY,
   GOLD INT  CHECK (GOLD >= 0),
   SILVER INT  CHECK (SILVER >= 0),
   BRONZE INT  CHECK (BRONZE >= 0),
   TOTAL INT  CHECK (TOTAL >= 0),
   RANKING INT ,
   COUNTRYID INT ,
   FOREIGN KEY (COUNTRYID) REFERENCES Country(ID)
);

CREATE TABLE Users (
   ID INT  AUTO_INCREMENT PRIMARY KEY,
   FIRSTNAME VARCHAR(100) ,
   LASTNAME VARCHAR(100) ,
   USERNAME VARCHAR(100) ,
   EMAIL VARCHAR(100) ,
   PASSWORD VARCHAR(255)  CHECK (LENGTH(PASSWORD) >= 6),
   UNIQUE (USERNAME),
   UNIQUE (EMAIL)
);

CREATE TABLE schedule (
   ID INT  AUTO_INCREMENT,
   eventdate DATE,
   game VARCHAR(255) ,
   category ENUM('Olympics', 'Paralympics', 'Special Olympics') ,
   event VARCHAR(255) ,
   team VARCHAR(255) ,
   ticket_count INT  CHECK (ticket_count BETWEEN 0 AND 1000),
   PRIMARY KEY (ID)
);

CREATE TABLE Plays (
  ATHLETEID INT ,
  SPORTID INT ,
  PRIMARY KEY (ATHLETEID, SPORTID),
  FOREIGN KEY (ATHLETEID) REFERENCES Athlete(ID) ON DELETE CASCADE,
  FOREIGN KEY (SPORTID) REFERENCES Sport(ID) ON DELETE CASCADE
);
```

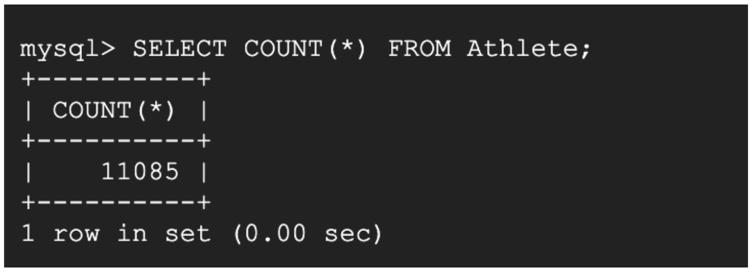
Step 3. Insert data into these tables. You should insert at least 1000 rows each in three of the tables. Try to use real data, but if you cannot find a good dataset for a particular table, you may use auto-generated data.

4 Tables have the following rows of data

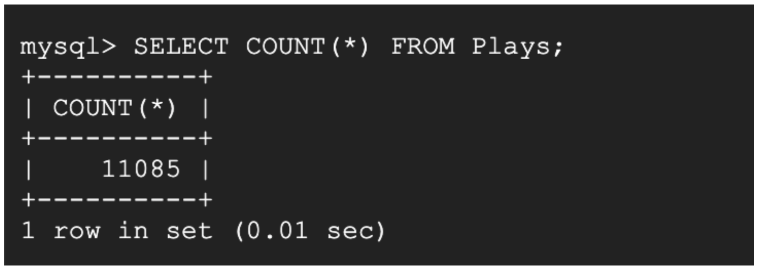
a. Schedule - 1437
b. Athlete - 11,085
c. Plays - 11,085
d. Users - 1027



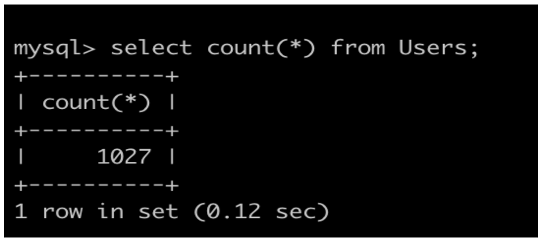
```
mysql> SELECT COUNT(*) FROM schedule;
+----------+
| COUNT(*) |
+----------+
|     1437 |
+----------+
1 row in set (0.01 sec)
```



```
mysql> SELECT COUNT(*) FROM Athlete;
+----------+
| COUNT(*) |
+----------+
|    11085 |
+----------+
1 row in set (0.00 sec)
```



```
mysql> SELECT COUNT(*) FROM Plays;
+----------+
| COUNT(*) |
+----------+
|    11085 |
+----------+
1 row in set (0.01 sec)
```



```
mysql> select count(*) from Users;
+----------+
| count(*) |
+----------+
|     1027 |
+----------+
1 row in set (0.12 sec)
```

Step 4.As a group, develop two advanced SQL queries related to the project that are different from one another. The two advance queries are expected to be part of your final application. The queries should each involve at least two of the following SQL concepts:Join of multiple relations,Set operations,Aggregation via GROUP BY,Subqueries

---

Query 1:

A Query to list out countries and their medal tallies in a columnar format.

Reason : We will be leveraging this to display Rankings of Country in a structured format so it is easy to visualize all the Country Rankings.

Code Block for Query 1

```
SELECT Country.NAME, sum(Medals.GOLD) as Gold, sum(Medals.SILVER) as Silver,
sum(Medals.BRONZE) as Bronze
FROM Country inner join Medals on Country.ID = Medals.COUNTRYID inner join Plays on
Medals.COUNTRYID = Plays.ATHLETEID inner join Sport on Plays.SPORTID = Sport.ID
GROUP BY Country.NAME
ORDER BY sum(Medals.TOTAL) DESC;
```

This query performs the following operations :

- Join of multiple relations: It joins four relations (Country, Medals, Plays, and Sport)
- Aggregation via GROUP BY: Aggregates by country name using GROUP BY function.

Query 2:

A Query to list out sports and also show athletes along with their countries

Reason : We will be leveraging this to display Sports so that we can easily show participants of a sport or game directly

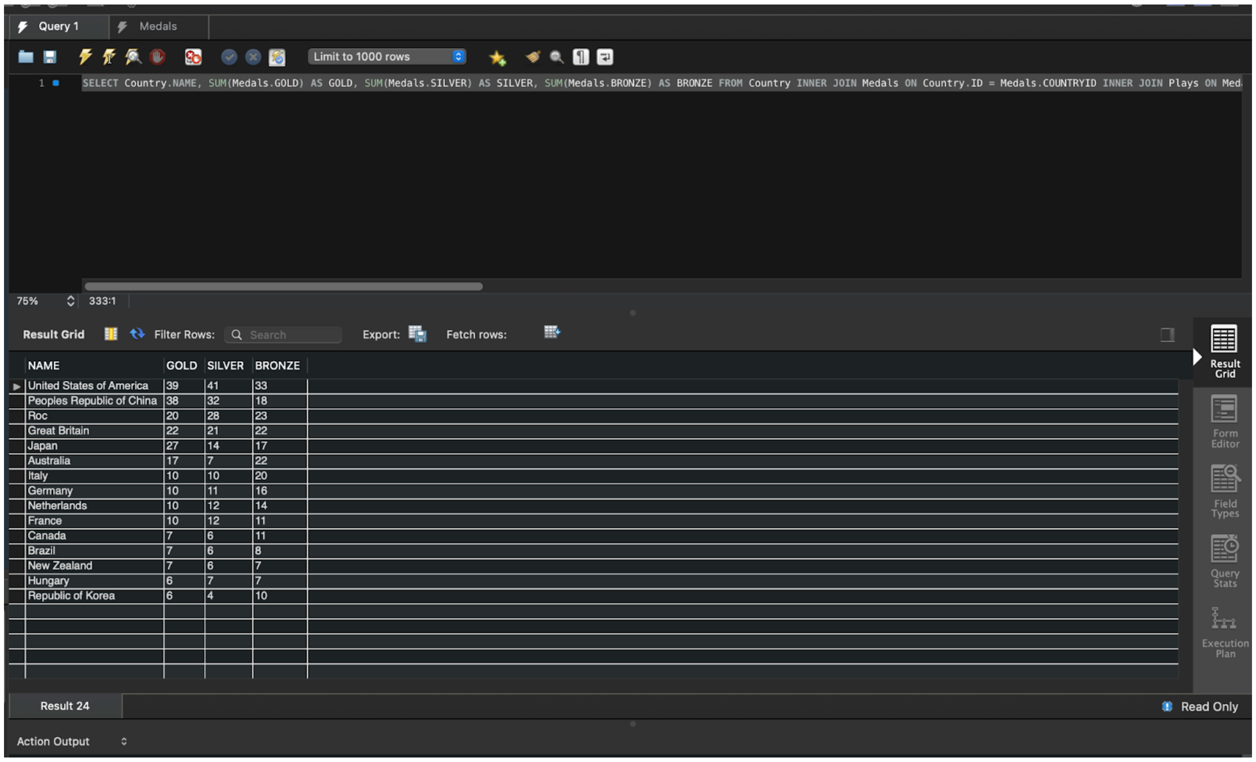
Code Block for Query 2

```sql
SELECT Sport.NAME as Sport, Country.NAME as Country, Athlete.NAME as Athlete
FROM Sport JOIN Plays on Sport.ID = Plays.SPORTID
INNER JOIN Athlete ON Plays.ATHLETEID = Athlete.ID
INNER JOIN Country ON Athlete.COUNTRYID = Country.ID
GROUP BY Sport.NAME, Country.NAME, Athlete.ID
ORDER BY Sport.NAME;
```

This query performs the following operations :
Join of multiple relations: It joins four relations (Sport, Plays, Athlete, and Country)
Aggregation via GROUP BY: Aggregates by sportname,countryname,athleteid using GROUP BY function.

Step 5. Execute your advanced SQL queries and provide a screenshot of the top 15 rows of each query result (you can use the LIMIT clause to select the top 15 rows). If your output is less then 15 rows, say that in your output

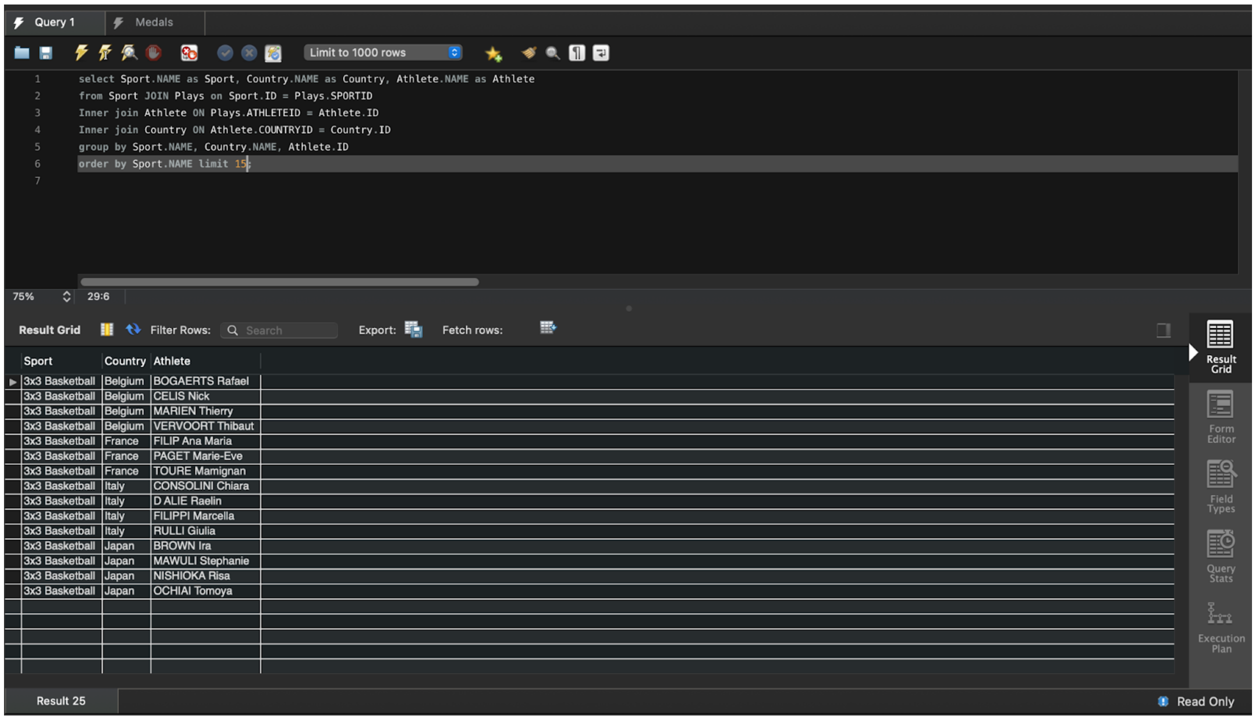
Query1



```sql
SELECT Country.NAME, SUM(Medals.GOLD) AS GOLD, SUM(Medals.SILVER) AS SILVER, SUM(Medals.BRONZE) AS BRONZE FROM Country INNER JOIN Medals ON Country.ID = Medals.COUNTRYID INNER JOIN Plays ON Med
```

NAME	GOLD	SILVER	BRONZE
United States of America	39	41	33
Peoples Republic of China	38	32	18
Roc	20	28	23
Great Britain	22	21	22
Japan	27	14	17
Australia	17	7	22
Italy	10	10	20
Germany	10	11	16
Netherlands	10	12	14
France	10	12	11
Canada	7	6	11
Brazil	7	6	8
New Zealand	7	6	7
Hungary	6	7	7
Republic of Korea	6	4	10

Query2



```
select Sport.NAME as Sport, Country.NAME as Country, Athlete.NAME as Athlete
from Sport JOIN Plays on Sport.ID = Plays.SPORTID
Inner join Athlete ON Plays.ATHLETEID = Athlete.ID
Inner join Country ON Athlete.COUNTRYID = Country.ID
group by Sport.NAME, Country.NAME, Athlete.ID
order by Sport.NAME limit 15;
```

Sport	Country	Athlete
3x3 Basketball	Belgium	BOGAERTS Rafael
3x3 Basketball	Belgium	CELIS Nick
3x3 Basketball	Belgium	MARIEN Thierry
3x3 Basketball	Belgium	VERVOORT Thibaut
3x3 Basketball	France	FILIP Ana Maria
3x3 Basketball	France	PAGET Marie-Eve
3x3 Basketball	France	TOURE Mamignan
3x3 Basketball	Italy	CONSOLINI Chiara
3x3 Basketball	Italy	D ALIE Raelin
3x3 Basketball	Italy	FILIPPI Marcella
3x3 Basketball	Italy	RULLI Giulia
3x3 Basketball	Japan	BROWN Ira
3x3 Basketball	Japan	MAWULI Stephanie
3x3 Basketball	Japan	NISHIOKA Risa
3x3 Basketball	Japan	OCHIAI Tomoya

Step 6.Indexing

Query 1.

```
SELECT Country.NAME, sum(Medals.GOLD) as Gold, sum(Medals.SILVER) as Silver,
sum(Medals.BRONZE) as Bronze
FROM Country inner join Medals on Country.ID = Medals.COUNTRYID inner join Plays on
Medals.COUNTRYID = Plays.ATHLETEID inner join Sport on Plays.SPORTID = Sport.ID
GROUP BY Country.NAME ORDER BY sum(Medals.TOTAL) DESC;
```

Indexing Stage 1: First we ran the query without any indexing set upon, The explain analyze command on the query provided with an actual time of 0.81. Kindly note that here we restricted the output since we would be displaying the top N countries on the banner from the webpage. This should be representative of indexing when actually leveraging the table or the query in our web application upon completion.
Indexing Stage 2: First we add the following index

```
CREATE INDEX testingindexhereonplays ON Plays (SPORTID);
```

After this we saw that the time dropped down to 0.781
Post this we add another index

```sql
CREATE INDEX testingindexhereonmedals ON Medals (COUNTRYID);
```

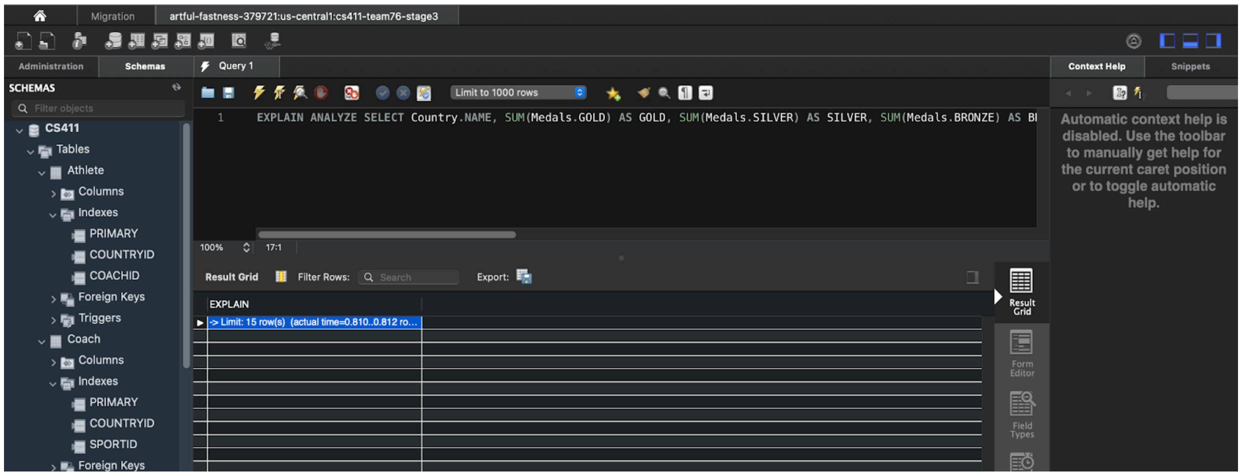
This changed the performance to 0.776

Post this we add another index

```sql
CREATE INDEX testingindexhereonplaysatheletes ON Plays (ATHLETEID);
```

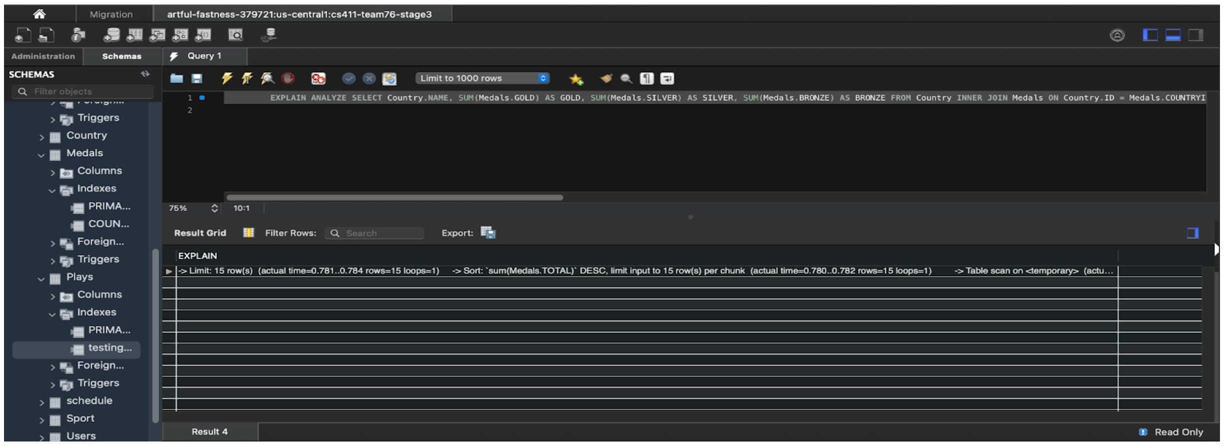
This changed the performance to 0.805

Attached are all the images for the same below



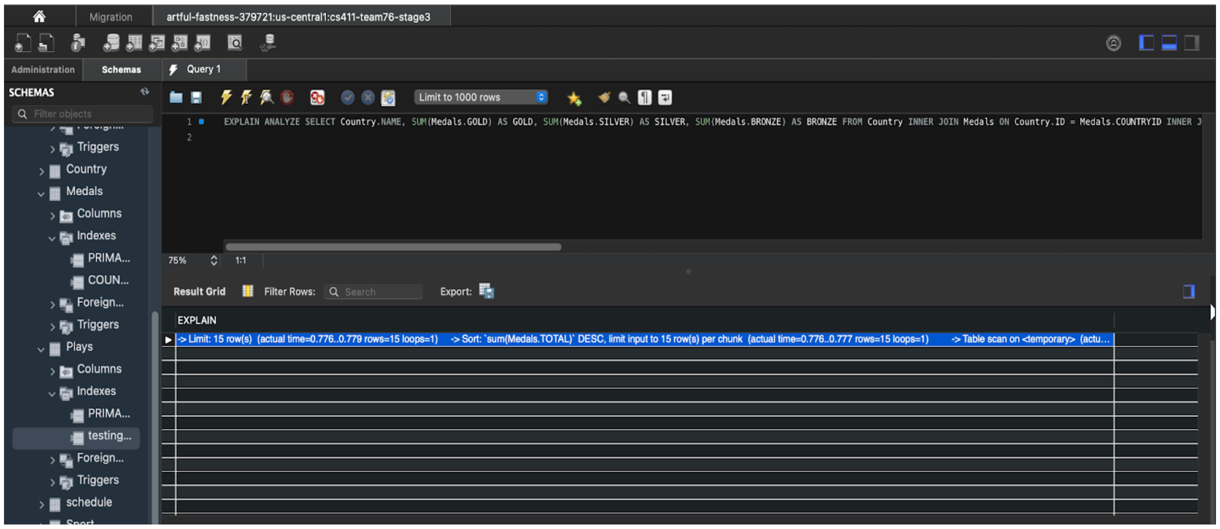
```
EXPLAIN ANALYZE SELECT Country.NAME, SUM(Medals.GOLD) AS GOLD, SUM(Medals.SILVER) AS SILVER, SUM(Medals.BRONZE) AS BI
```

```
EXPLAIN
-> Limit: 15 row(s)  (actual time=0.810..0.812 ro...
```



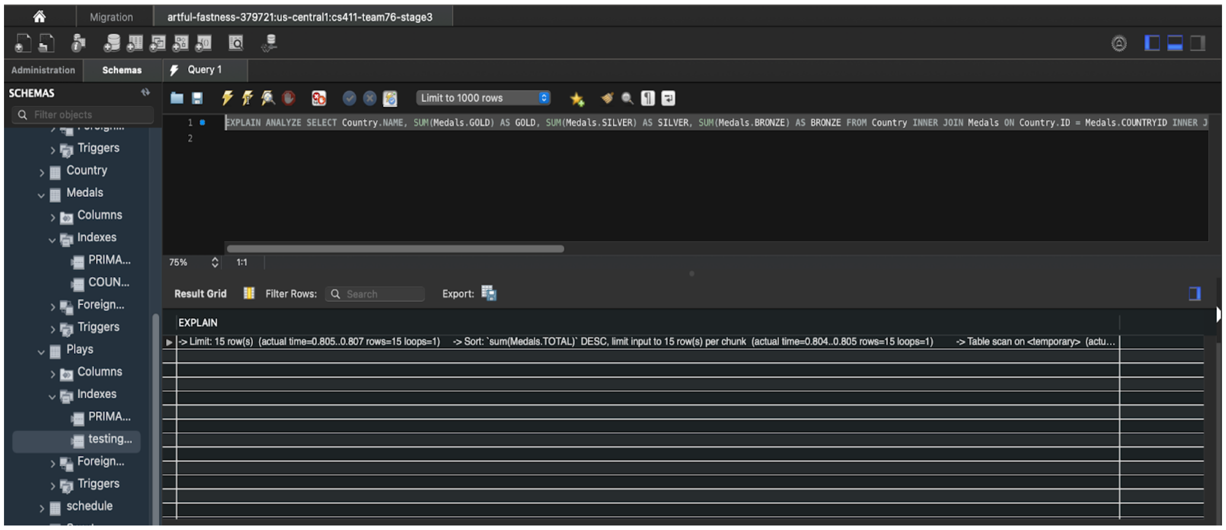
```
EXPLAIN ANALYZE SELECT Country.NAME, SUM(Medals.GOLD) AS GOLD, SUM(Medals.SILVER) AS SILVER, SUM(Medals.BRONZE) AS BRONZE FROM Country INNER JOIN Medals ON Country.ID = Medals.COUNTRYI
```

```
EXPLAIN
-> Limit: 15 row(s)  (actual time=0.781..0.784 rows=15 loops=1)    -> Sort: `sum(Medals.TOTAL)` DESC, limit input to 15 row(s) per chunk  (actual time=0.780..0.782 rows=15 loops=1)        -> Table scan on <temporary>  (actu...
```



```
EXPLAIN ANALYZE SELECT Country.NAME, SUM(Medals.GOLD) AS GOLD, SUM(Medals.SILVER) AS SILVER, SUM(Medals.BRONZE) AS BRONZE FROM Country INNER JOIN Medals ON Country.ID = Medals.COUNTRYID INNER J
```

```
EXPLAIN
-> Limit: 15 row(s)  (actual time=0.776..0.779 rows=15 loops=1)    -> Sort: `sum(Medals.TOTAL)` DESC, limit input to 15 row(s) per chunk  (actual time=0.776..0.777 rows=15 loops=1)    -> Table scan on <temporary>  (actu...
```



```
EXPLAIN ANALYZE SELECT Country.NAME, SUM(Medals.GOLD) AS GOLD, SUM(Medals.SILVER) AS SILVER, SUM(Medals.BRONZE) AS BRONZE FROM Country INNER JOIN Medals ON Country.ID = Medals.COUNTRYID INNER J
```

```
EXPLAIN
-> Limit: 15 row(s)  (actual time=0.805..0.807 rows=15 loops=1)    -> Sort: `sum(Medals.TOTAL)` DESC, limit input to 15 row(s) per chunk  (actual time=0.804..0.805 rows=15 loops=1)    -> Table scan on <temporary>  (actu...
```

Indexing 3: We saw that in the query we perform joins on the following tables (Country, Medals, Plays, and Sport). We saw that the performance changed from 0.81 to 0.78 to 0.776 to 0.80.

For the first index : We will be using this index. Performance Positive (0.81 to 0.78)

```
CREATE INDEX testingindexhereonplays ON Plays (SPORTID);
```

It would make sense why the first index was on the sport id on the Plays table (the weak entity table) on sport id to country id of the medal table. The key here was the medals table primarily needs country id and medal id to put up the result that we would need for the so the index on this slightly improved the performance [0.81 to 0.78]. The index on country id would have fast tracked the search and enabled. So we should keep this index as it would help find using country id upon leveraging the join operations.

For the second index: We will be using this index. Performance Positive (0.78 to 0.776)

```
CREATE INDEX testingindexhereonmedals ON Medals (COUNTRYID);
```

It again would make sense why the second index helps. Here we see that we created an index on CountryID directly on the Medals Table. Again the key here was the medals table primarily needs country id and medal id to put up the result that we would need for the so the index on this slightly improved the performance [0.78 to 0.776] . The index on country id now this time directly on the medals table would have fast tracked the search and enabled. So we should keep this index as it would help find using country id upon leveraging the join operations.

For the third index: We will not be using this index. Performance Negative (0.776 to 0.80)

```
CREATE INDEX testingindexhereonplaysatheletes ON Plays (ATHLETEID);
```

It makes sense why the performance dropped here. On the weak entity now we try to make athlete id along with the sport id this time around in addition to the previous index.We saw that the performance took a hit here now. This would make sense now as this time around we put the index on AtheleteId of the weak entity. AthleteID makes no sense to the overall search index as atheleteid should have no bearing on the overall performance as it has no significance and adding an unnecessary index could have performance impacts. Which we do see in this case so we can tell that this index is not right for us so we are dropping this index.

Indexing 4: Note that if you did not find any difference in your results, report that as well. Explain why you think this change in indexing did not bring a better effect to your query.

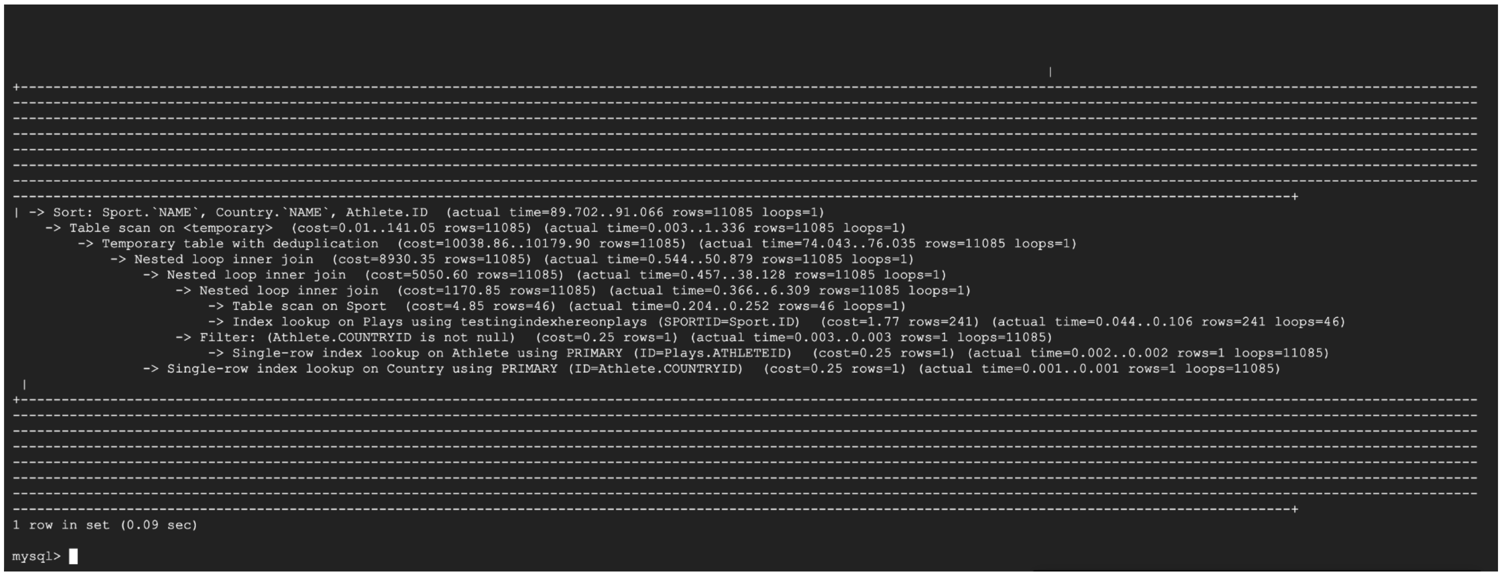
We did see performance changes upon using the indexes so this section does not apply to us in this case.

Query 2.

```
SELECT Sport.NAME as Sport, Country.NAME as Country, Athlete.NAME as Athlete
FROM Sport JOIN Plays on Sport.ID = Plays.SPORTID
INNER JOIN Athlete ON Plays.ATHLETEID = Athlete.ID
INNER JOIN Country ON Athlete.COUNTRYID = Country.ID
GROUP BY Sport.NAME, Country.NAME, Athlete.ID
ORDER BY Sport.NAME;
```

Indexing Stage 1: First we ran the query without any indexing set upon, The explain analyze command on the query provided an actual time of 89.702. We restricted the output since we would be displaying the top N countries on the banner from the webpage.

Run Explain Analyze:



```
                                                                                                                                 |
+--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------+
| -> Sort: Sport.`NAME`, Country.`NAME`, Athlete.ID  (actual time=89.702..91.066 rows=11085 loops=1)
    -> Table scan on <temporary>  (cost=0.01..141.05 rows=11085) (actual time=0.003..1.336 rows=11085 loops=1)
        -> Temporary table with deduplication  (cost=10038.86..10179.90 rows=11085) (actual time=74.043..76.035 rows=11085 loops=1)
            -> Nested loop inner join  (cost=8930.35 rows=11085) (actual time=0.544..50.879 rows=11085 loops=1)
                -> Nested loop inner join  (cost=5050.60 rows=11085) (actual time=0.457..38.128 rows=11085 loops=1)
                    -> Nested loop inner join  (cost=1170.85 rows=11085) (actual time=0.366..6.309 rows=11085 loops=1)
                        -> Table scan on Sport  (cost=4.85 rows=46) (actual time=0.204..0.252 rows=46 loops=1)
                        -> Index lookup on Plays using testingindexhereonplays (SPORTID=Sport.ID)  (cost=1.77 rows=241) (actual time=0.044..0.106 rows=241 loops=46)
                    -> Filter: (Athlete.COUNTRYID is not null)  (cost=0.25 rows=1) (actual time=0.003..0.003 rows=1 loops=11085)
                        -> Single-row index lookup on Athlete using PRIMARY (ID=Plays.ATHLETEID)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=11085)
                -> Single-row index lookup on Country using PRIMARY (ID=Athlete.COUNTRYID)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=11085)
 |
+--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set (0.09 sec)

mysql> 
```

It takes a total of 89.702

Indexing Stage 2: First we add the following index

```
CREATE INDEX isportathlete ON Plays(SPORTID, ATHLETEID);;
```

Time decreases to 57.192

Then we create the next index

```
CREATE INDEX icountry ON Athlete (ID, COUNTRYID);
```

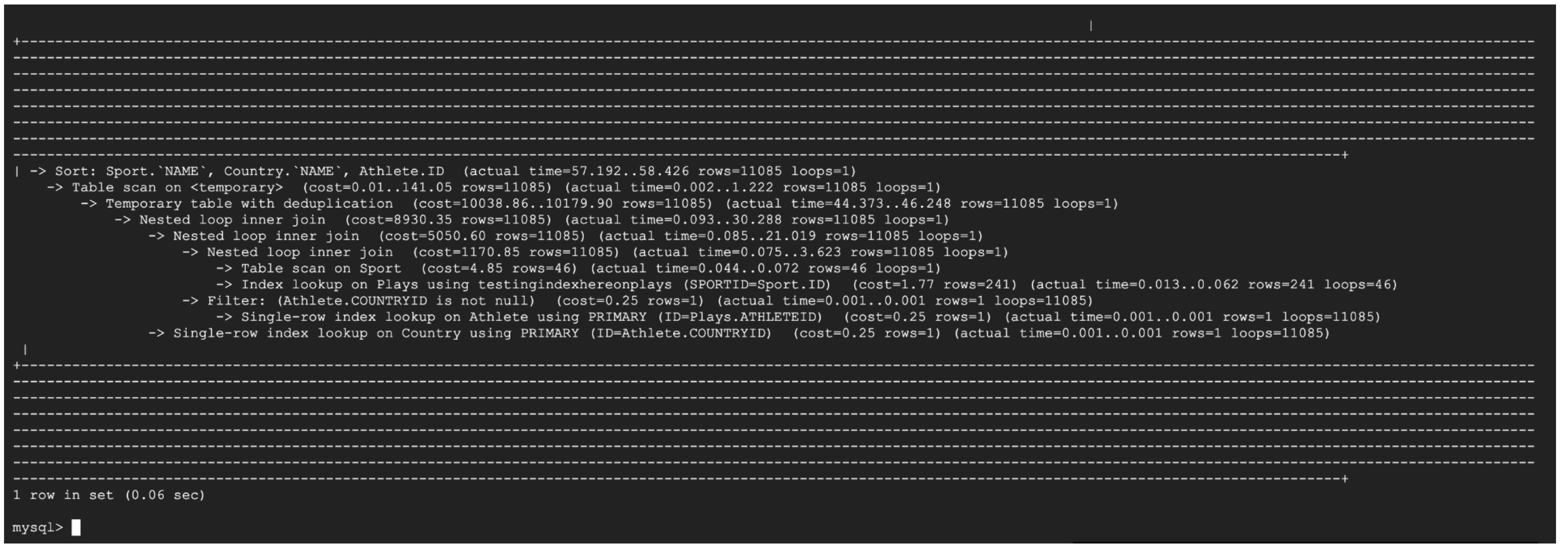
Time decreases to 54.656 seconds.

We then create our final index.

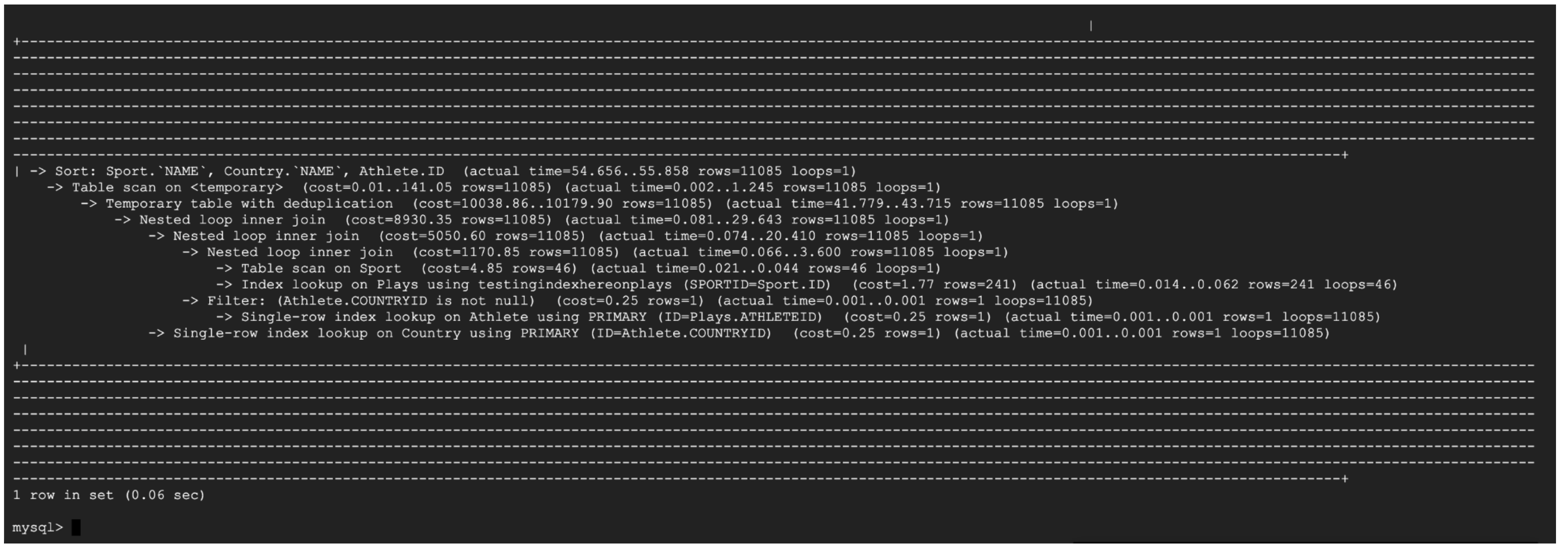
```
CREATE INDEX iplays ON Plays(ATHLETEID);
```

Time now increases to 70.871 seconds.

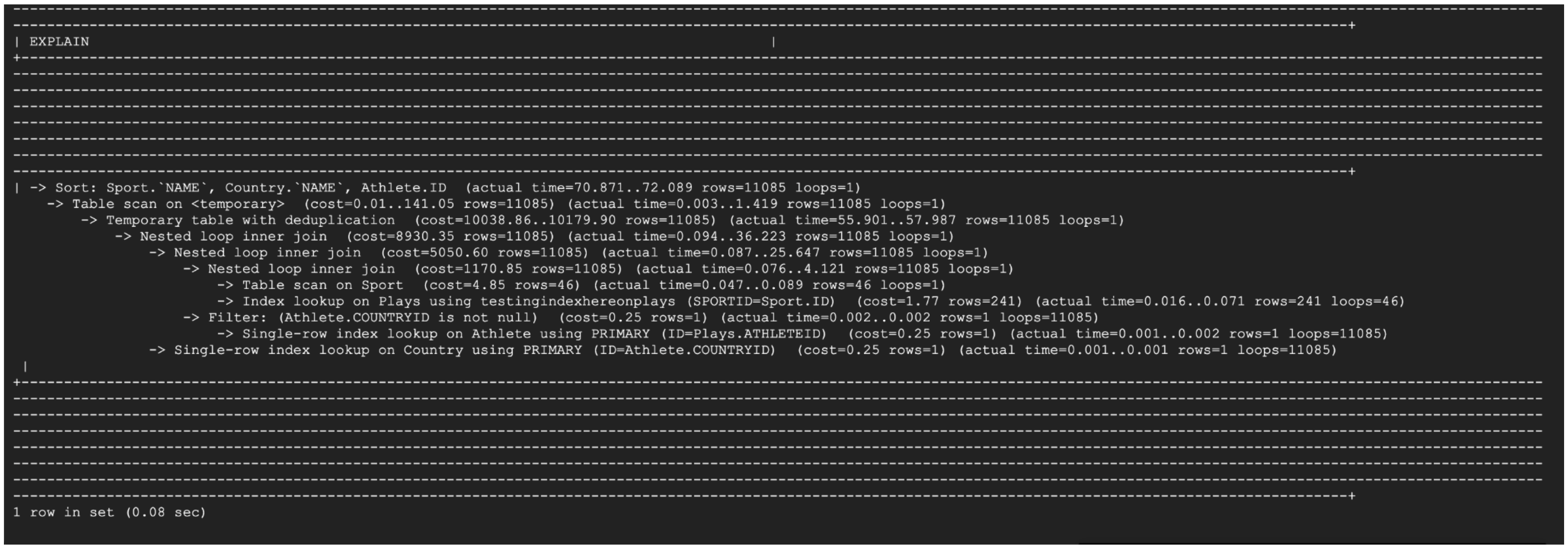
Attached are the images that prove these times:



```
                                                                                                                                          |
+--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| -> Sort: Sport.`NAME`, Country.`NAME`, Athlete.ID  (actual time=57.192..58.426 rows=11085 loops=1)
    -> Table scan on <temporary>  (cost=0.01..141.05 rows=11085) (actual time=0.002..1.222 rows=11085 loops=1)
        -> Temporary table with deduplication  (cost=10038.86..10179.90 rows=11085) (actual time=44.373..46.248 rows=11085 loops=1)
            -> Nested loop inner join  (cost=8930.35 rows=11085) (actual time=0.093..30.288 rows=11085 loops=1)
                -> Nested loop inner join  (cost=5050.60 rows=11085) (actual time=0.085..21.019 rows=11085 loops=1)
                    -> Nested loop inner join  (cost=1170.85 rows=11085) (actual time=0.075..3.623 rows=11085 loops=1)
                        -> Table scan on Sport  (cost=4.85 rows=46) (actual time=0.044..0.072 rows=46 loops=1)
                        -> Index lookup on Plays using testingindexhereonplays (SPORTID=Sport.ID)  (cost=1.77 rows=241) (actual time=0.013..0.062 rows=241 loops=46)
                    -> Filter: (Athlete.COUNTRYID is not null)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=11085)
                        -> Single-row index lookup on Athlete using PRIMARY (ID=Plays.ATHLETEID)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=11085)
                -> Single-row index lookup on Country using PRIMARY (ID=Athlete.COUNTRYID)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=11085)
 |
+--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set (0.06 sec)

mysql> 
```



```
                                                                                                                                          |
+--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| -> Sort: Sport.`NAME`, Country.`NAME`, Athlete.ID  (actual time=54.656..55.858 rows=11085 loops=1)
    -> Table scan on <temporary>  (cost=0.01..141.05 rows=11085) (actual time=0.002..1.245 rows=11085 loops=1)
        -> Temporary table with deduplication  (cost=10038.86..10179.90 rows=11085) (actual time=41.779..43.715 rows=11085 loops=1)
            -> Nested loop inner join  (cost=8930.35 rows=11085) (actual time=0.081..29.643 rows=11085 loops=1)
                -> Nested loop inner join  (cost=5050.60 rows=11085) (actual time=0.074..20.410 rows=11085 loops=1)
                    -> Nested loop inner join  (cost=1170.85 rows=11085) (actual time=0.066..3.600 rows=11085 loops=1)
                        -> Table scan on Sport  (cost=4.85 rows=46) (actual time=0.021..0.044 rows=46 loops=1)
                        -> Index lookup on Plays using testingindexhereonplays (SPORTID=Sport.ID)  (cost=1.77 rows=241) (actual time=0.014..0.062 rows=241 loops=46)
                    -> Filter: (Athlete.COUNTRYID is not null)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=11085)
                        -> Single-row index lookup on Athlete using PRIMARY (ID=Plays.ATHLETEID)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=11085)
                -> Single-row index lookup on Country using PRIMARY (ID=Athlete.COUNTRYID)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=11085)
 |
+--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set (0.06 sec)

mysql> 
```



```
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| EXPLAIN                                                                                              |
+--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| -> Sort: Sport.`NAME`, Country.`NAME`, Athlete.ID  (actual time=70.871..72.089 rows=11085 loops=1)
    -> Table scan on <temporary>  (cost=0.01..141.05 rows=11085) (actual time=0.003..1.419 rows=11085 loops=1)
        -> Temporary table with deduplication  (cost=10038.86..10179.90 rows=11085) (actual time=55.901..57.987 rows=11085 loops=1)
            -> Nested loop inner join  (cost=8930.35 rows=11085) (actual time=0.094..36.223 rows=11085 loops=1)
                -> Nested loop inner join  (cost=5050.60 rows=11085) (actual time=0.087..25.647 rows=11085 loops=1)
                    -> Nested loop inner join  (cost=1170.85 rows=11085) (actual time=0.076..4.121 rows=11085 loops=1)
                        -> Table scan on Sport  (cost=4.85 rows=46) (actual time=0.047..0.089 rows=46 loops=1)
                        -> Index lookup on Plays using testingindexhereonplays (SPORTID=Sport.ID)  (cost=1.77 rows=241) (actual time=0.016..0.071 rows=241 loops=46)
                    -> Filter: (Athlete.COUNTRYID is not null)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=11085)
                        -> Single-row index lookup on Athlete using PRIMARY (ID=Plays.ATHLETEID)  (cost=0.25 rows=1) (actual time=0.001..0.002 rows=1 loops=11085)
                -> Single-row index lookup on Country using PRIMARY (ID=Athlete.COUNTRYID)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=11085)
 |
+--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set (0.08 sec)
```

Indexing 3:
For the first index: We will be using this index. Performance Positive (89.702 to 57.192)

```
CREATE INDEX country ON Athlete (ID, COUNTRYID);
```

We will be utilizing this first index because it improves the performance of the query. This is because allowing COUNTRYID to be directly connected to the Athlete entity reduces overhead and simplifies the overall search. The index on country id now this time directly on the athlete table would have fast tracked the search and enabled. So we should keep this index as it would help find using country id upon leveraging the join operations.

For the second index: We will be using this index. Performance Positive (57.192 to 54.656)

```
CREATE INDEX testingindexhereonplays ON Plays (SPORTID);
```

Along with the first index, we will be using this second index because it improves the overall performance of the query. We created an index on the plays table through the sport's ID. The index on sport id now this time directly on the plays table fast tracks the search. So we should keep this index as it would help find using country id upon leveraging the join operations.

For the third index: We will not be using this index. Performance Negative (54.656 to 70.871)

```
CREATE INDEX iplays ON Plays(ATHLETEID);
```

For the first time the indexing we are adding has reduced performance. This is because performance downfalls are exacerbated by the extraneous information that this indexing provides. Overall adding this index would not make sense, not only because of the obvious performance issues, but the necessity of this index would not be worth the tradeoff.

Indexing 4: Note that if you did not find any difference in your results, report that as well. Explain why you think this change in indexing did not bring a better effect to your query.

We did see performance changes upon using the indexes so this section does not apply to us in this case.