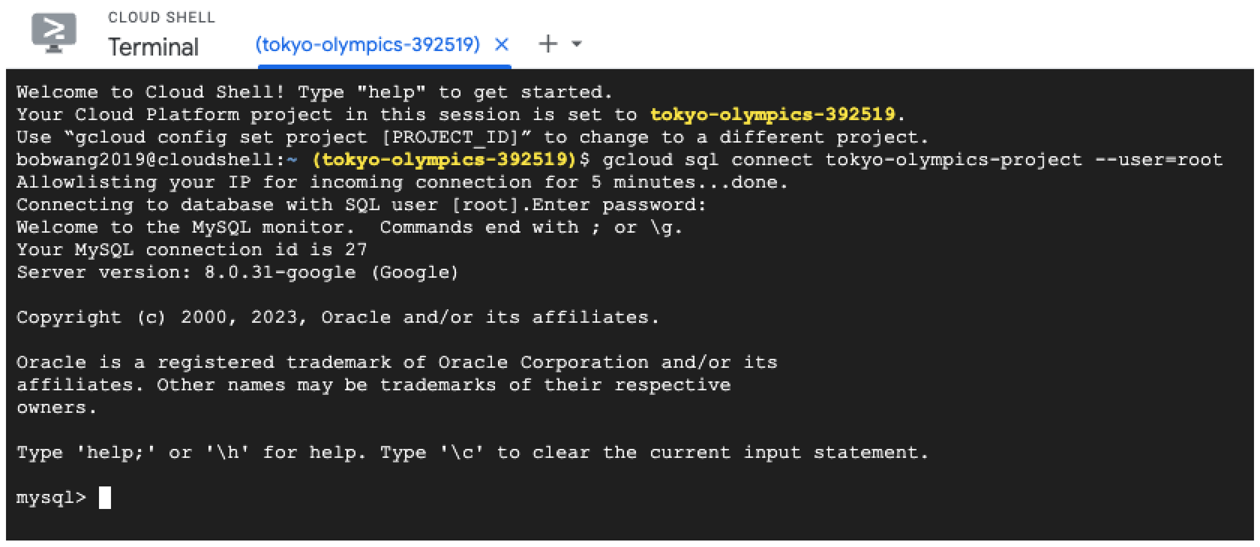
GCP connection:

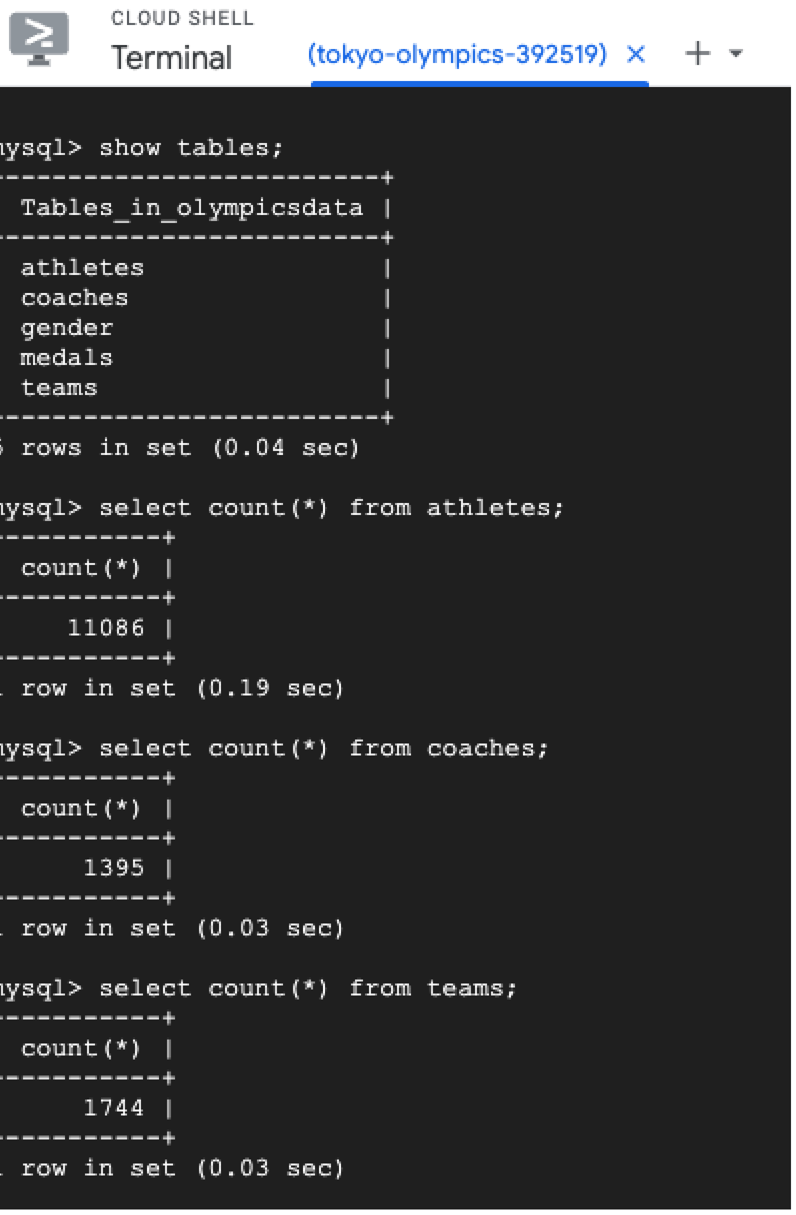


```
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to tokyo-olympics-392519.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
bobwang2019@cloudshell:~ (tokyo-olympics-392519)$ gcloud sql connect tokyo-olympics-project --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 27
Server version: 8.0.31-google (Google)

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

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

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

mysql>
```

create table DDL commands:

1) create table athletes(Name varchar(255), NOC varchar(255), Discipline varchar(255), PRIMARY KEY(Name));
2) create table coaches(Name varchar(255), NOC varchar(255), Discipline varchar(255), PRIMARY KEY(Name));
3) create table teams(Name varchar(255), Discipline varchar(255),NOC varchar(255), Event varchar(255), PRIMARY KEY(Name));
4) create table medals(NOC varchar(255), Gold int, Silver int, Bronze int, Total int, PRIMARY KEY(NOC));
5) create table gender(Discipline varchar(255), Female int, Male int,Total int, PRIMARY KEY(Discipline));

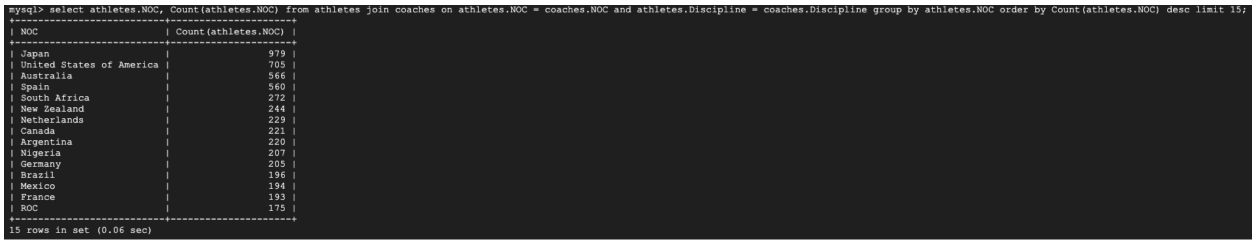
3 tables over 1000 rows:



```
mysql> show tables;
+-------------------------+
| Tables_in_olympicsdata |
+-------------------------+
| athletes                |
| coaches                 |
| gender                  |
| medals                  |
| teams                   |
+-------------------------+
5 rows in set (0.04 sec)

mysql> select count(*) from athletes;
+----------+
| count(*) |
+----------+
|    11086 |
+----------+
1 row in set (0.19 sec)

mysql> select count(*) from coaches;
+----------+
| count(*) |
+----------+
|     1395 |
+----------+
1 row in set (0.03 sec)

mysql> select count(*) from teams;
+----------+
| count(*) |
+----------+
|     1744 |
+----------+
1 row in set (0.03 sec)
```

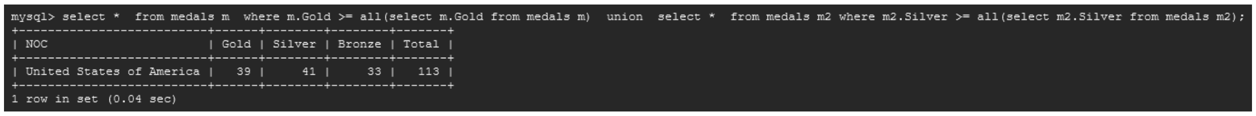
Advanced SQL queries:

select athletes.NOC, Count(athletes.NOC)
from athletes join coaches on athletes.NOC = coaches.NOC and athletes.Discipline = coaches.Discipline
group by athletes.NOC
order by Count(athletes.NOC)
desc limit 15;



```
mysql> select athletes.NOC, Count(athletes.NOC) from athletes join coaches on athletes.NOC = coaches.NOC and athletes.Discipline = coaches.Discipline group by athletes.NOC order by Count(athletes.NOC) desc limit 15;
+--------------------------+---------------------+
| NOC                      | Count(athletes.NOC) |
+--------------------------+---------------------+
| Japan                    |                 979 |
| United States of America |                 705 |
| Australia                |                 566 |
| Spain                    |                 560 |
| South Africa             |                 272 |
| New Zealand              |                 244 |
| Netherlands              |                 229 |
| Canada                   |                 221 |
| Argentina                |                 220 |
| Nigeria                  |                 207 |
| Germany                  |                 205 |
| Brazil                   |                 196 |
| Mexico                   |                 194 |
| France                   |                 193 |
| ROC                      |                 175 |
+--------------------------+---------------------+
15 rows in set (0.06 sec)
```

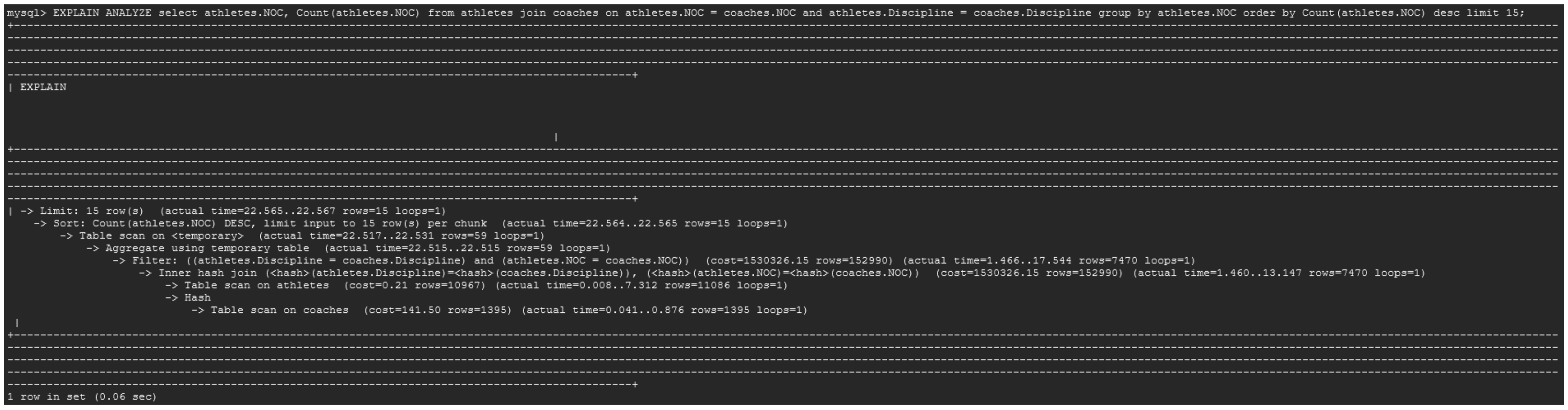
select *
from medals m
where m.Gold >= all(select m.Gold from medals m)
union
select *
from medals m2
where m2.Silver >= all(select m2.Silver from medals m2);



```
mysql> select *  from medals m  where m.Gold >= all(select m.Gold from medals m)  union  select *  from medals m2 where m2.Silver >= all(select m2.Silver from medals m2);
+--------------------------+------+--------+--------+-------+
| NOC                      | Gold | Silver | Bronze | Total |
+--------------------------+------+--------+--------+-------+
| United States of America |   39 |     41 |     33 |   113 |
+--------------------------+------+--------+--------+-------+
1 row in set (0.04 sec)
```

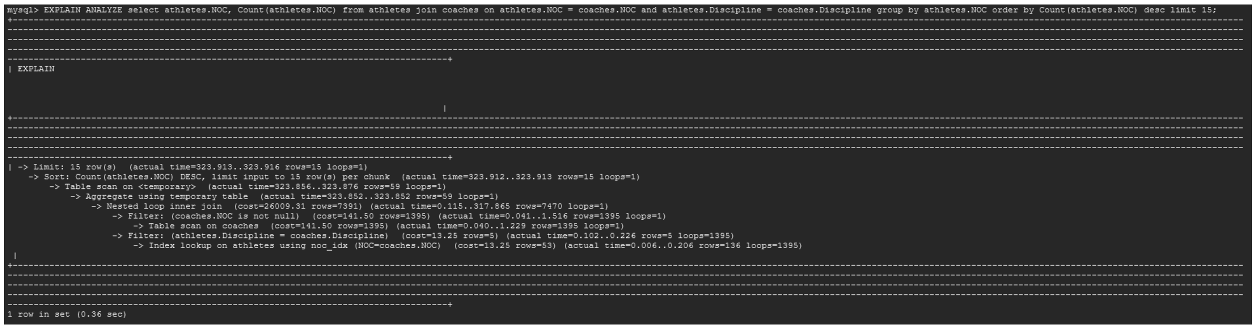
**Indexing Analysis:**

Query 1 Before:



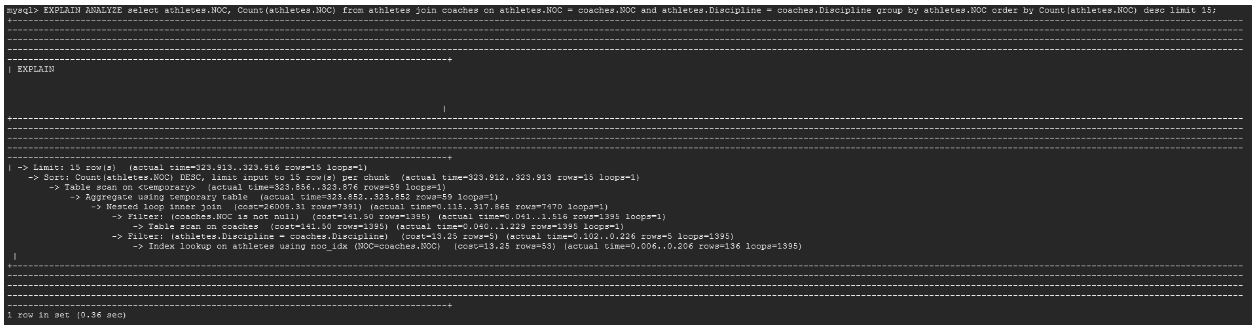
```
mysql> EXPLAIN ANALYZE select athletes.NOC, Count(athletes.NOC) from athletes join coaches on athletes.NOC = coaches.NOC and athletes.Discipline = coaches.Discipline group by athletes.NOC order by Count(athletes.NOC) desc limit 15;
+--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------+
| EXPLAIN



                                                                                    |
+--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------+
| -> Limit: 15 row(s)  (actual time=22.565..22.567 rows=15 loops=1)
    -> Sort: Count(athletes.NOC) DESC, limit input to 15 row(s) per chunk  (actual time=22.564..22.565 rows=15 loops=1)
        -> Table scan on <temporary>  (actual time=22.517..22.531 rows=59 loops=1)
            -> Aggregate using temporary table  (actual time=22.515..22.515 rows=59 loops=1)
                -> Filter: ((athletes.Discipline = coaches.Discipline) and (athletes.NOC = coaches.NOC))  (cost=1530326.15 rows=152990) (actual time=1.466..17.544 rows=7470 loops=1)
                    -> Inner hash join (<hash>(athletes.Discipline)=<hash>(coaches.Discipline)), (<hash>(athletes.NOC)=<hash>(coaches.NOC))  (cost=1530326.15 rows=152990) (actual time=1.460..13.147 rows=7470 loops=1)
                        -> Table scan on athletes  (cost=0.21 rows=10967) (actual time=0.008..7.312 rows=11086 loops=1)
                        -> Hash
                            -> Table scan on coaches  (cost=141.50 rows=1395) (actual time=0.041..0.876 rows=1395 loops=1)
 |
+--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------+
1 row in set (0.06 sec)
```

Using the index of Athlete NOC:



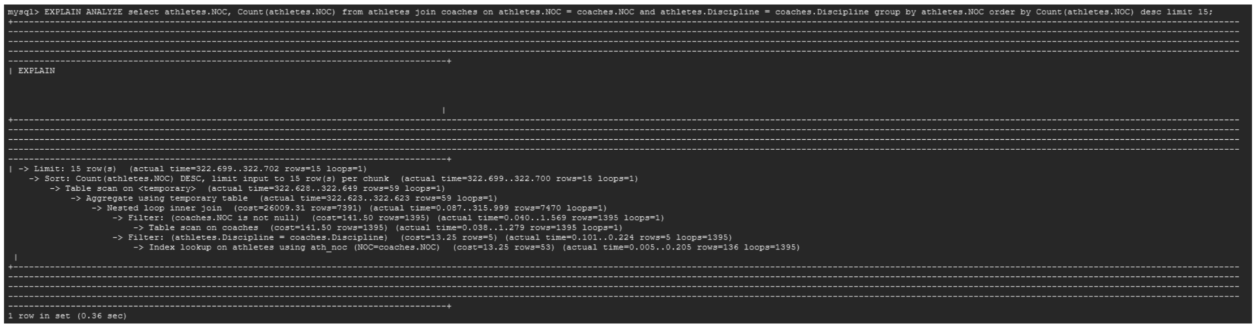
```
mysql> EXPLAIN ANALYZE select athletes.NOC, Count(athletes.NOC) from athletes join coaches on athletes.NOC = coaches.NOC and athletes.Discipline = coaches.Discipline group by athletes.NOC order by Count(athletes.NOC) desc limit 15;
+--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------+
| EXPLAIN



                                                                                 |
+--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------+
| -> Limit: 15 row(s)  (actual time=323.913..323.916 rows=15 loops=1)
    -> Sort: Count(athletes.NOC) DESC, limit input to 15 row(s) per chunk  (actual time=323.912..323.913 rows=15 loops=1)
        -> Table scan on <temporary>  (actual time=323.856..323.876 rows=59 loops=1)
            -> Aggregate using temporary table  (actual time=323.852..323.852 rows=59 loops=1)
                -> Nested loop inner join  (cost=26009.31 rows=7391) (actual time=0.115..317.865 rows=7470 loops=1)
                    -> Filter: (coaches.NOC is not null)  (cost=141.50 rows=1395) (actual time=0.041..1.516 rows=1395 loops=1)
                        -> Table scan on coaches  (cost=141.50 rows=1395) (actual time=0.040..1.229 rows=1395 loops=1)
                    -> Filter: (athletes.Discipline = coaches.Discipline)  (cost=13.25 rows=5) (actual time=0.102..0.226 rows=5 loops=1395)
                        -> Index lookup on athletes using noc_idx (NOC=coaches.NOC)  (cost=13.25 rows=53) (actual time=0.006..0.206 rows=136 loops=1395)
 |
+--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------+
1 row in set (0.36 sec)
```

Using the index of Coach NOC:



```
mysql> EXPLAIN ANALYZE select athletes.NOC, Count(athletes.NOC) from athletes join coaches on athletes.NOC = coaches.NOC and athletes.Discipline = coaches.Discipline group by athletes.NOC order by Count(athletes.NOC) desc limit 15;
+--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------+
| EXPLAIN



                                                                                 |
+--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------+
| -> Limit: 15 row(s)  (actual time=323.913..323.916 rows=15 loops=1)
    -> Sort: Count(athletes.NOC) DESC, limit input to 15 row(s) per chunk  (actual time=323.912..323.913 rows=15 loops=1)
        -> Table scan on <temporary>  (actual time=323.856..323.876 rows=59 loops=1)
            -> Aggregate using temporary table  (actual time=323.852..323.852 rows=59 loops=1)
                -> Nested loop inner join  (cost=26009.31 rows=7391) (actual time=0.115..317.865 rows=7470 loops=1)
                    -> Filter: (coaches.NOC is not null)  (cost=141.50 rows=1395) (actual time=0.041..1.516 rows=1395 loops=1)
                        -> Table scan on coaches  (cost=141.50 rows=1395) (actual time=0.040..1.229 rows=1395 loops=1)
                    -> Filter: (athletes.Discipline = coaches.Discipline)  (cost=13.25 rows=5) (actual time=0.102..0.226 rows=5 loops=1395)
                        -> Index lookup on athletes using noc_idx (NOC=coaches.NOC)  (cost=13.25 rows=53) (actual time=0.006..0.206 rows=136 loops=1395)
 |
+--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------+
1 row in set (0.36 sec)
```

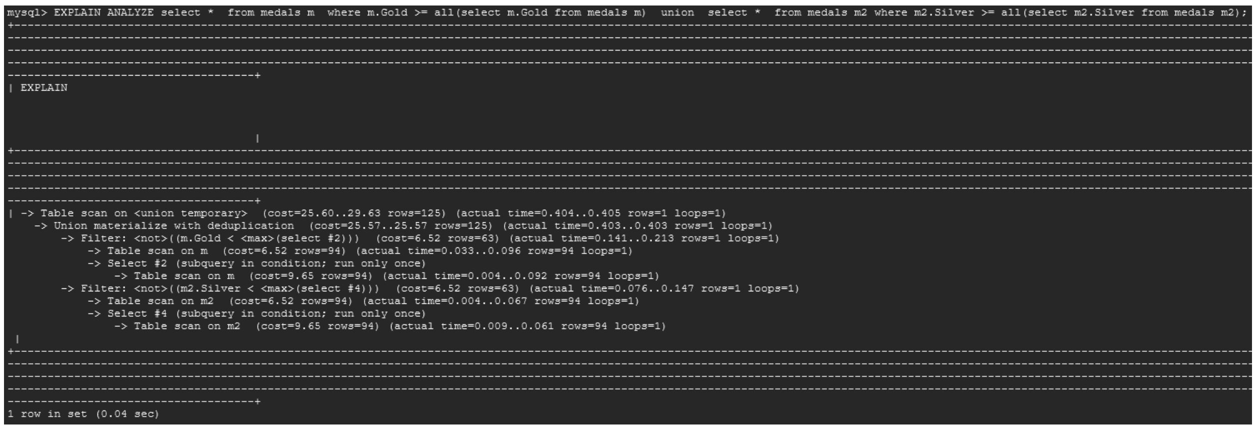
Using a combination of coaches_noc and athletes_noc idx:



```
mysql> EXPLAIN ANALYZE select athletes.NOC, Count(athletes.NOC) from athletes join coaches on athletes.NOC = coaches.NOC and athletes.Discipline = coaches.Discipline group by athletes.NOC order by Count(athletes.NOC) desc limit 15;
+--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------+
| EXPLAIN



                                                                                 |
+--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------+
| -> Limit: 15 row(s)  (actual time=322.699..322.702 rows=15 loops=1)
    -> Sort: Count(athletes.NOC) DESC, limit input to 15 row(s) per chunk  (actual time=322.699..322.700 rows=15 loops=1)
        -> Table scan on <temporary>  (actual time=322.628..322.649 rows=59 loops=1)
            -> Aggregate using temporary table  (actual time=322.623..322.623 rows=59 loops=1)
                -> Nested loop inner join  (cost=26009.31 rows=7391) (actual time=0.087..315.999 rows=7470 loops=1)
                    -> Filter: (coaches.NOC is not null)  (cost=141.50 rows=1395) (actual time=0.040..1.569 rows=1395 loops=1)
                        -> Table scan on coaches  (cost=141.50 rows=1395) (actual time=0.038..1.279 rows=1395 loops=1)
                    -> Filter: (athletes.Discipline = coaches.Discipline)  (cost=13.25 rows=5) (actual time=0.101..0.224 rows=5 loops=1395)
                        -> Index lookup on athletes using ath_noc (NOC=coaches.NOC)  (cost=13.25 rows=53) (actual time=0.005..0.205 rows=136 loops=1395)
 |
+--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------+
1 row in set (0.36 sec)
```

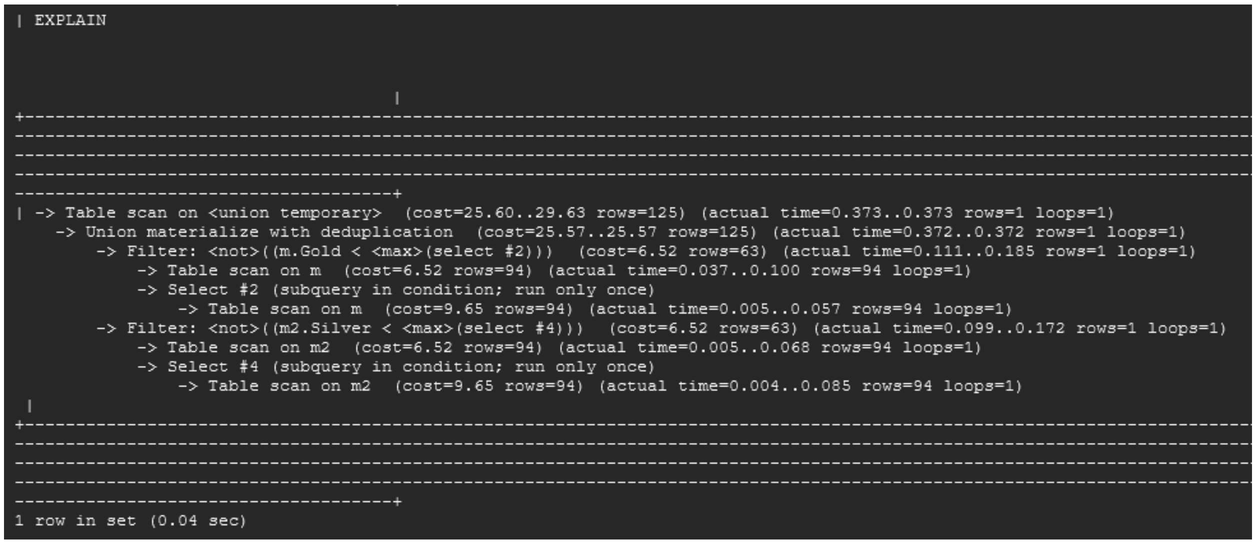
When comparing the performance of each of the created indices we noticed that adding indices only worsen the performance of the query. We believe this occurs as the query aims to get specific information that doesn't need the help of specific indices in the table. Since the tables of athletes and coaches same very similar information, adding indices will only add time to the processing and reorganization of this information. What we noticed is that additional cost was added since new filters were then added into the query in order to achieve the end result.

Query 2 Before:



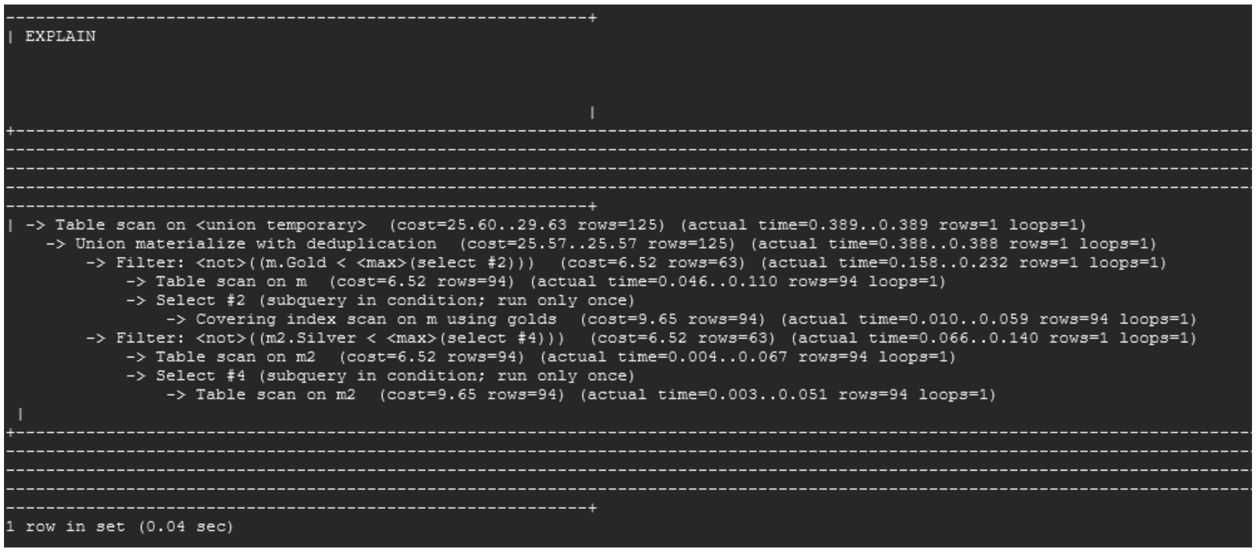
```
mysql> EXPLAIN ANALYZE select *  from medals m  where m.Gold >= all(select m.Gold from medals m)  union  select *  from medals m2 where m2.Silver >= all(select m2.Silver from medals m2);
+------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------+
| EXPLAIN



                                     |
+------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------+
| -> Table scan on <union temporary>  (cost=25.60..29.63 rows=125) (actual time=0.404..0.405 rows=1 loops=1)
    -> Union materialize with deduplication  (cost=25.57..25.57 rows=125) (actual time=0.403..0.403 rows=1 loops=1)
        -> Filter: <not>((m.Gold < <max>(select #2)))  (cost=6.52 rows=63) (actual time=0.141..0.213 rows=1 loops=1)
            -> Table scan on m  (cost=6.52 rows=94) (actual time=0.033..0.096 rows=94 loops=1)
            -> Select #2 (subquery in condition; run only once)
                -> Table scan on m  (cost=9.65 rows=94) (actual time=0.004..0.092 rows=94 loops=1)
        -> Filter: <not>((m2.Silver < <max>(select #4)))  (cost=6.52 rows=63) (actual time=0.076..0.147 rows=1 loops=1)
            -> Table scan on m2  (cost=6.52 rows=94) (actual time=0.004..0.067 rows=94 loops=1)
            -> Select #4 (subquery in condition; run only once)
                -> Table scan on m2  (cost=9.65 rows=94) (actual time=0.009..0.061 rows=94 loops=1)
 |
+------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------+
1 row in set (0.04 sec)
```

INDEX ON GOLD



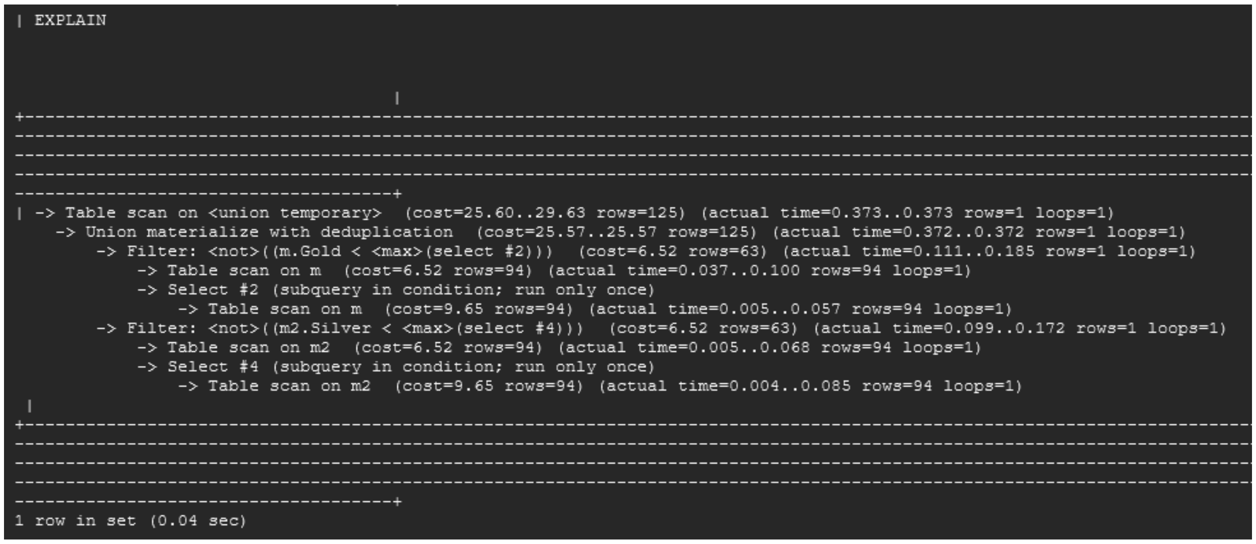
```
| EXPLAIN



                                     |
+------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------
--------------------------------------+
| -> Table scan on <union temporary>  (cost=25.60..29.63 rows=125) (actual time=0.373..0.373 rows=1 loops=1)
    -> Union materialize with deduplication  (cost=25.57..25.57 rows=125) (actual time=0.372..0.372 rows=1 loops=1)
        -> Filter: <not>((m.Gold < <max>(select #2)))  (cost=6.52 rows=63) (actual time=0.111..0.185 rows=1 loops=1)
            -> Table scan on m  (cost=6.52 rows=94) (actual time=0.037..0.100 rows=94 loops=1)
            -> Select #2 (subquery in condition; run only once)
                -> Table scan on m  (cost=9.65 rows=94) (actual time=0.005..0.057 rows=94 loops=1)
        -> Filter: <not>((m2.Silver < <max>(select #4)))  (cost=6.52 rows=63) (actual time=0.099..0.172 rows=1 loops=1)
            -> Table scan on m2  (cost=6.52 rows=94) (actual time=0.005..0.068 rows=94 loops=1)
            -> Select #4 (subquery in condition; run only once)
                -> Table scan on m2  (cost=9.65 rows=94) (actual time=0.004..0.085 rows=94 loops=1)
 |
+------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------
--------------------------------------+
1 row in set (0.04 sec)
```

INDEX ON SILVER



```
----------------------------------------------------------+
| EXPLAIN



                                                          |
+------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------+
| -> Table scan on <union temporary>  (cost=25.60..29.63 rows=125) (actual time=0.389..0.389 rows=1 loops=1)
    -> Union materialize with deduplication  (cost=25.57..25.57 rows=125) (actual time=0.388..0.388 rows=1 loops=1)
        -> Filter: <not>((m.Gold < <max>(select #2)))  (cost=6.52 rows=63) (actual time=0.158..0.232 rows=1 loops=1)
            -> Table scan on m  (cost=6.52 rows=94) (actual time=0.046..0.110 rows=94 loops=1)
            -> Select #2 (subquery in condition; run only once)
                -> Covering index scan on m using golds  (cost=9.65 rows=94) (actual time=0.010..0.059 rows=94 loops=1)
        -> Filter: <not>((m2.Silver < <max>(select #4)))  (cost=6.52 rows=63) (actual time=0.066..0.140 rows=1 loops=1)
            -> Table scan on m2  (cost=6.52 rows=94) (actual time=0.004..0.067 rows=94 loops=1)
            -> Select #4 (subquery in condition; run only once)
                -> Table scan on m2  (cost=9.65 rows=94) (actual time=0.003..0.051 rows=94 loops=1)
 |
+------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------+
1 row in set (0.04 sec)
```

INDEX ON NOC



```
| EXPLAIN



                                      |
+-----------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
--------------------------------------+
| -> Table scan on <union temporary>  (cost=25.60..29.63 rows=125) (actual time=0.373..0.373 rows=1 loops=1)
    -> Union materialize with deduplication  (cost=25.57..25.57 rows=125) (actual time=0.372..0.372 rows=1 loops=1)
        -> Filter: <not>((m.Gold < <max>(select #2)))  (cost=6.52 rows=63) (actual time=0.111..0.185 rows=1 loops=1)
            -> Table scan on m  (cost=6.52 rows=94) (actual time=0.037..0.100 rows=94 loops=1)
            -> Select #2 (subquery in condition; run only once)
                -> Table scan on m  (cost=9.65 rows=94) (actual time=0.005..0.057 rows=94 loops=1)
        -> Filter: <not>((m2.Silver < <max>(select #4)))  (cost=6.52 rows=63) (actual time=0.099..0.172 rows=1 loops=1)
            -> Table scan on m2  (cost=6.52 rows=94) (actual time=0.005..0.068 rows=94 loops=1)
            -> Select #4 (subquery in condition; run only once)
                -> Table scan on m2  (cost=9.65 rows=94) (actual time=0.004..0.085 rows=94 loops=1)
 |
+-----------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
--------------------------------------+
1 row in set (0.04 sec)
```

No matter the index that is created, or the combination of indices the performance of the second query does not change. The query aims to retrieve the nation with the user inputted constraints (ie. most gold AND silver medals) it will only need to find counts of each and perform an operation on the resulting table. Additionally, since the values in the table are all numerical lookup times for information is already quick and doesn't require any sort of processing other than looking at the specific value.