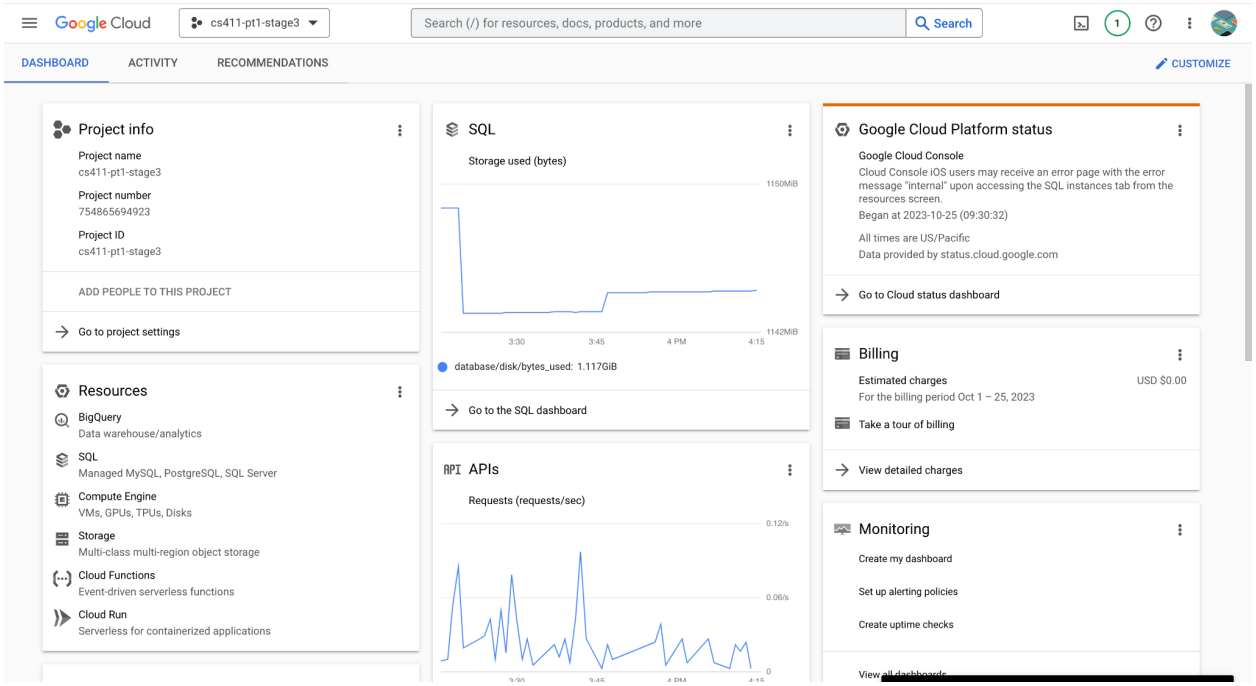
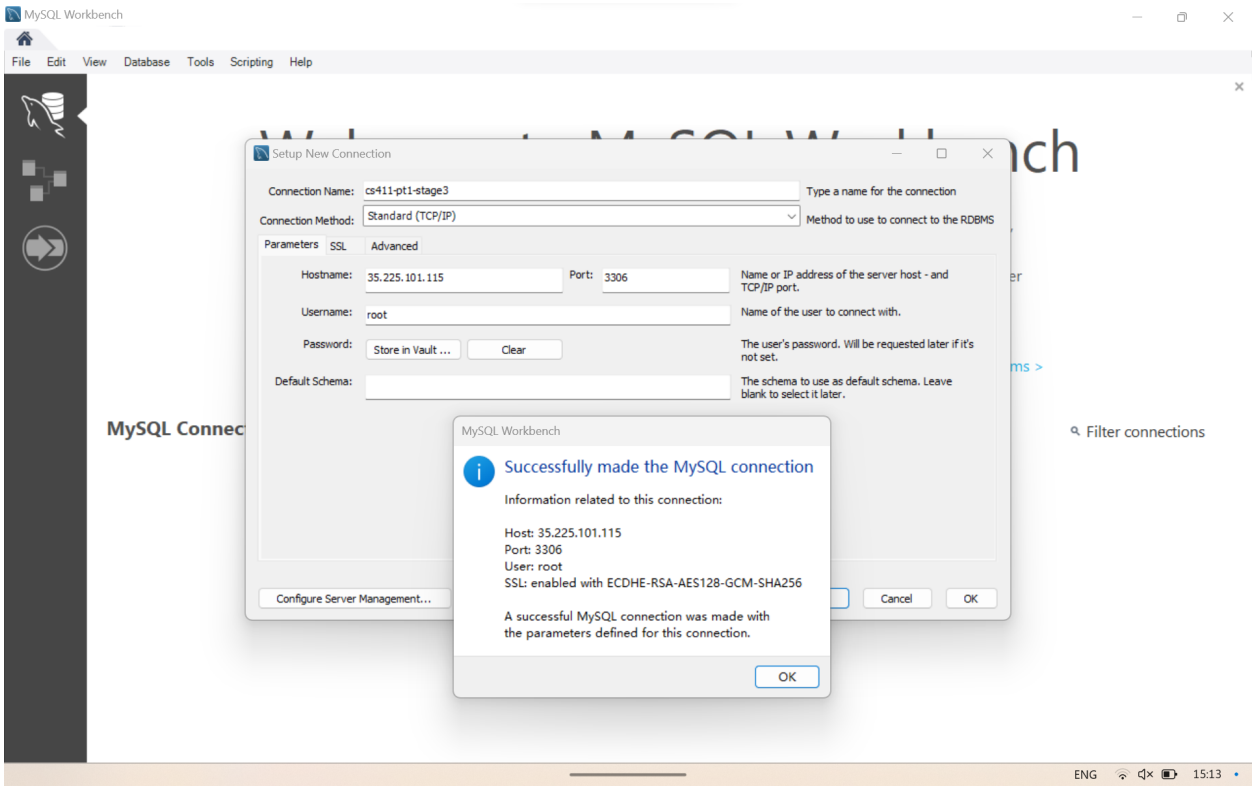
Implementing Database on GCP:



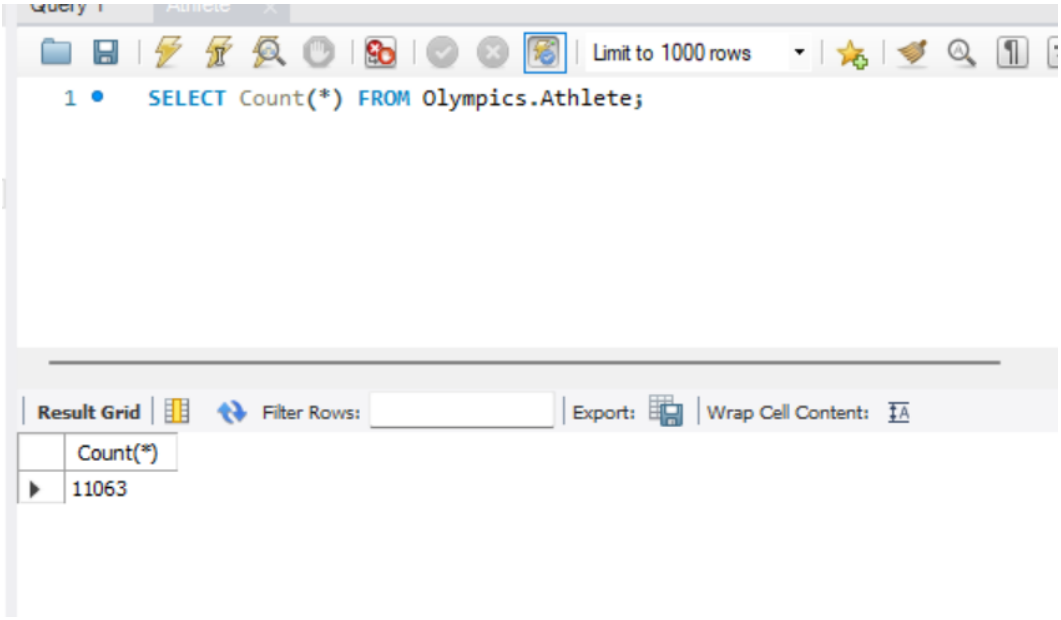


Implementation of four main tables:

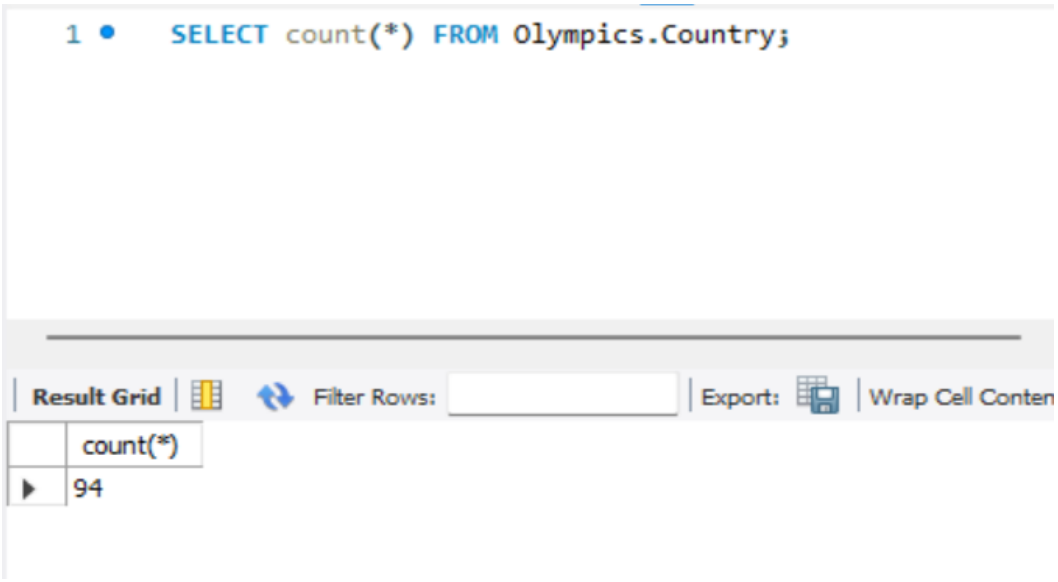
```
CREATE TABLE Country (
        CountryName VARCHAR(255) PRIMARY KEY,
        Gold INT,
        Silver INT,
        Bronze INT,
        Total INT
);


CREATE TABLE Athlete (
        Name VARCHAR(45) PRIMARY KEY,
        CountryName VARCHAR(45) NOT NULL,
        Discipline VARCHAR(45) NOT NULL,
        FOREIGN KEY (CountryName) REFERENCES Country(CountryName)
                                ON DELETE CASCADE
);


CREATE TABLE Coach (
        Name VARCHAR(45),
        CountryName VARCHAR(45) NOT NULL,
        Discipline VARCHAR(45) NOT NULL,
        Event VARCHAR(45) DEFAULT 'N',
        PRIMARY KEY (Name, Event),
        FOREIGN KEY (CountryName) REFERENCES Country(CountryName)
                                        ON DELETE CASCADE
);


CREATE TABLE Team (
        TeamId INT PRIMARY KEY,
        CountryName VARCHAR(45) NOT NULL,
        Discipline VARCHAR(45) NOT NULL,
        Event VARCHAR(45) NOT NULL,
        FOREIGN KEY (CountryName) REFERENCES Country(CountryName)
                                        ON DELETE CASCADE
);
```

Number of Entries per table:

Athlete:



Country:



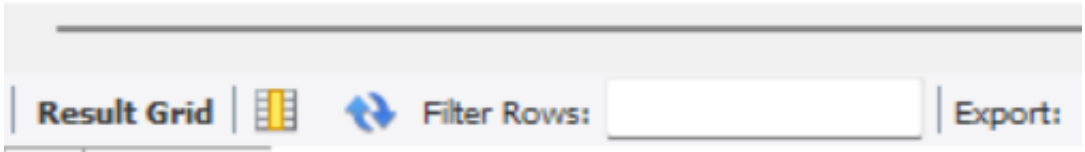
Coach:

```
1 •  SELECT COUNT(*) FROM Olympics.Coach;
```

	COUNT(*)
▶	1194

Team:

```
1 •  SELECT count(*) FROM Olympics.Team;
```



	count(*)
▶	1037

Advanced Query1:

SELECT c.name as CoachName, COUNT(a.name) as AthleteCount
FROM Olympics.Athlete as a JOIN Olympics.Coach as c
WHERE a.CountryName = c.CountryName AND a.Discipline = c.Discipline
GROUP BY c.name
ORDER BY c.name;

```
1  SELECT c.name as CoachName, COUNT(a.name) as AthleteCount
2  FROM Olympics.Athlete as a JOIN Olympics.Coach as c
3  WHERE a.CountryName = c.CountryName AND a.Discipline = c.Discipline
4  GROUP BY c.name
5  ORDER BY c.name
6  LIMIT 15;
7
```

100% 10:6

Result Grid Filter Rows: Search Export: Fetch rows:

CoachName	AthleteCount
ABDELMAGID Wael	20
ABE Junya	22
ABE Katsuhiko	22
Adam Burgess DDS	1
ADAMA Cherif	20
AGEBA Yuya	22
AIKMAN Siegfried Gottlieb	32
AL SAADI Kais	33
ALAMEDA Lonni	14
Albert Brown DVM	1
ALEKNO Vladimir	12
ALEKSEEV Alexey	14
Alexander Hancock	13
Alexis Wells	4
Alice Evans	29

Result Grid
Form Editor
Field Types
Query Stats

Justification:

This query will return the number of athletes each coach has. In this way, when users are searching for information about coaches and athletes, we can provide more insights to them with advanced query like this. We would use similar queries to generate the number of coaches an athlete has, and we can also calculate the average based on the results. Thus, a comprehensive analysis of the number of athletes and coaches for each country and discipline can be displayed with a set of queries in our system.

Analysis for advanced query 1:

Indexes before of table Athlete:

```
10 •  SHOW INDEX FROM Olympics.Athlete;
11
12 •  SHOW INDEX FROM Olympics.Coach;
13
```

100% 34:10

Result Grid | Filter Rows: Search | Export:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Cor
Athlete	0	PRIMARY	1	Name	A	11252	NULL	NULL		BTREE	

Indexes before of table Coach:

```
12 •  SHOW INDEX FROM Olympics.Coach;
13
14 •  EXPLAIN ANALYZE SELECT c.name as CoachName, COUNT(a.name) as AthleteCount
15    FROM Olympics.Athlete as a JOIN Olympics.Coach as c
16    WHERE a.CountryName = c.CountryName AND a.Discipline = c.Discipline
```

100% 32:12

Result Grid | Filter Rows: Search | Export:

T	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
C	0	PRIMARY	1	Name	A	1177	NULL	NULL		BTREE			YES	NULL
C	0	PRIMARY	2	Event	A	1194	NULL	NULL		BTREE			YES	NULL

Initial runtime, runtime without adding any extra indices: 13.802

```
14 •  EXPLAIN ANALYZE SELECT c.name as CoachName, COUNT(a.name) as AthleteCount
15    FROM Olympics.Athlete as a JOIN Olympics.Coach as c
16    WHERE a.CountryName = c.CountryName AND a.Discipline = c.Discipline
17    GROUP BY c.name
18    ORDER BY c.name
19    LIMIT 15;
20
```

100% 1:20

Form Editor | Navigate: 1 / 1

EXPLAIN:

```
-> Limit: 15 row(s)  (actual time=13.802..13.804 rows=15 loops=1)
    -> Sort: c.`Name`, limit input to 15 row(s) per chunk  (actual time=13.801..13.802 rows=15 loops=1)
        -> Table scan on <temporary>  (actual time=13.594..13.696 rows=719 loops=1)
            -> Aggregate using temporary table  (actual time=13.593..13.593 rows=719 loops=1)
```

New Indexes added to Athlete, we only add b-tree index on Athlete.CountryName:

```
25
26 •  CREATE INDEX CountryName_idx ON Athlete(CountryName);
27 •  SHOW INDEX FROM Olympics.Athlete;
28
29
30
31
32
```

100% 34:27

Result Grid Filter Rows: Search Export:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Cor
Athlete	0	PRIMARY	1	Name	A	11252	NULL	NULL		BTREE	
Athlete	1	CountryName_idx	1	CountryName	A	207	NULL	NULL		BTREE	

New Runtime: 0.679, much faster than before

```
14 •  EXPLAIN ANALYZE SELECT c.name as CoachName, COUNT(a.name) as AthleteCount
15    FROM Olympics.Athlete as a JOIN Olympics.Coach as c
16    WHERE a.CountryName = c.CountryName AND a.Discipline = c.Discipline
17    GROUP BY c.name
18    ORDER BY c.name
19    LIMIT 15;
20
```

100% 10:19

Form Editor Navigate: 1 / 1

EXPLAIN:

```
-> Limit: 15 row(s)  (cost=16237.67 rows=11) (actual time=0.679..7.462 rows=15 loops=1)
  -> Group aggregate: count(a.`Name`)  (cost=16237.67 rows=11) (actual time=0.667..7.448 rows=15 loops=1)
    -> Nested loop inner join  (cost=16236.58 rows=11) (actual time=0.411..7.392 rows=260 loops=1)
      -> Index scan on c using PRIMARY  (cost=0.00 rows=2) (actual time=0.035..0.063 rows=25 loops=1)
```

New Indexes added to Athlete, we only add b-tree index on Athlete.Discipline:

```
29 •  CREATE INDEX Discipline_idx ON Athlete(Discipline);
30 •  SHOW INDEX FROM Olympics.Athlete;
31
32
33
34
35
```

100% 34:30

Result Grid Filter Rows: Search Export:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Cor
Athlete	0	PRIMARY	1	Name	A	11252	NULL	NULL		BTREE	
Athlete	1	Discipline_idx	1	Discipline	A	47	NULL	NULL		BTREE	

New Runtime: 1.759, faster than initial 13.802 but slower than only add index for CountryName

```
14 •  EXPLAIN ANALYZE SELECT c.name as CoachName, COUNT(a.name) as AthleteCount
15    FROM Olympics.Athlete as a JOIN Olympics.Coach as c
16    WHERE a.CountryName = c.CountryName AND a.Discipline = c.Discipline
17    GROUP BY c.name
18    ORDER BY c.name
19    LIMIT 15;
```

100% 1:20

Form Editor Navigate: 1 / 1

EXPLAIN:

```
-> Limit: 15 row(s)  (cost=71488.51 rows=15) (actual time=1.759..11.650 rows=15 loops=1)
    -> Group aggregate: count(a.`Name`)  (cost=71488.51 rows=24) (actual time=1.759..11.648 rows=15 loops=1)
        -> Nested loop inner join  (cost=71486.11 rows=24) (actual time=0.615..11.597 rows=260 loops=1)
            -> Index scan on c using PRIMARY  (cost=0.00 rows=1) (actual time=0.029..0.058 rows=25 loops=1)
```

New Indexes added to Coach, we only add b-tree index on Coach.CountryName:

```
33 •  CREATE INDEX CountryName_idx ON Coach(CountryName);
34 •  SHOW INDEX FROM Olympics.Coach;
35
36
37
38
39
```

100% 32:34

Result Grid Filter Rows: Search Export:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Cor
Coach	0	PRIMARY	1	Name	A	1177	NULL	NULL		BTREE	
Coach	0	PRIMARY	2	Event	A	1194	NULL	NULL		BTREE	
Coach	1	CountryName_idx	1	CountryName	A	62	NULL	NULL		BTREE	

New Runtime: 295.502 which is slower than the initial one.

```
14 •  EXPLAIN ANALYZE SELECT c.name as CoachName, COUNT(a.name) as AthleteCount
15    FROM Olympics.Athlete as a JOIN Olympics.Coach as c
16    WHERE a.CountryName = c.CountryName AND a.Discipline = c.Discipline
17    GROUP BY c.name
18    ORDER BY c.name
19    LIMIT 15;
```

100% 10:19

Form Editor Navigate: 1 / 1

EXPLAIN:

```
-> Limit: 15 row(s)  (actual time=295.502..295.504 rows=15 loops=1)
    -> Sort: c.`Name`, limit input to 15 row(s) per chunk  (actual time=295.501..295.502 rows=15 loops=1)
        -> Table scan on <temporary>  (actual time=295.252..295.393 rows=719 loops=1)
            -> Aggregate using temporary table  (actual time=295.249..295.249 rows=719 loops=1)
```

New Indexes added to Coach, we only add b-tree index on Coach.Discipline:

```
37 • CREATE INDEX Discipline_idx ON Coach(Discipline);
38 • SHOW INDEX FROM Olympics.Coach;
39
40
41
42
43
```

100% 32:38

Result Grid Filter Rows: Search Export:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Cor
Coach	0	PRIMARY	1	Name	A	1177	NULL	NULL		BTREE	
Coach	0	PRIMARY	2	Event	A	1194	NULL	NULL		BTREE	
Coach	1	Discipline_idx	1	Discipline	A	21	NULL	NULL		BTREE	

New Runtime: 445.446 which is much slower than the initial one.

```
14 • EXPLAIN ANALYZE SELECT c.name as CoachName, COUNT(a.name) as AthleteCount
15 FROM Olympics.Athlete as a JOIN Olympics.Coach as c
16 WHERE a.CountryName = c.CountryName AND a.Discipline = c.Discipline
17 GROUP BY c.name
18 ORDER BY c.name
19 LIMIT 15;
20
```

100% 10:19

Form Editor Navigate: 1 / 1

EXPLAIN:

```
-> Limit: 15 row(s)  (actual time=445.446..445.448 rows=15 loops=1)
    -> Sort: c.`Name`, limit input to 15 row(s) per chunk  (actual time=445.445..445.446 rows=15 loops=1)
        -> Table scan on <temporary>  (actual time=445.203..445.340 rows=719 loops=1)
            -> Aggregate using temporary table  (actual time=445.201..445.201 rows=719 loops=1)
```

Finally we try to add b-tree index on Athlete.CountryName and b-tree index on Athlete.Discipline together.

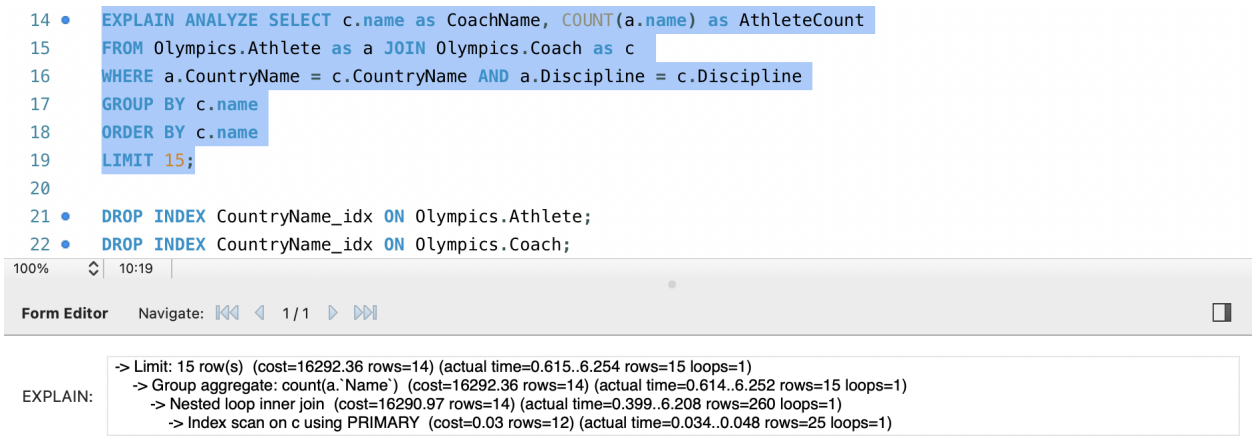
```
29 • CREATE INDEX CountryName_idx ON Athlete(CountryName);
30 • SHOW INDEX FROM Olympics.Athlete;
31
32 • CREATE INDEX Discipline_idx ON Athlete(Discipline);
33 • SHOW INDEX FROM Olympics.Athlete;
```

100% 34:33

Result Grid Filter Rows: Search Export:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Co
Athlete	0	PRIMARY	1	Name	A	11252	NULL	NULL		BTREE	
Athlete	1	CountryName_idx	1	CountryName	A	207	NULL	NULL		BTREE	
Athlete	1	Discipline_idx	1	Discipline	A	47	NULL	NULL		BTREE	

New Runtime: 0.615 which is the best performance that we achieve so far.



Results and explanation:
By using Athlete.CountryName as an index, we were able to speed up the run time of the query from 13 to 0.67 units of time By using Athlete. Discipline as an index, we were able to speed up the run time of the query from 13 to 1.8 units of time. By combining these two indexes, we can speed up the performance of the query from 13 to 0.61. However, by adding Coach.Discipline and Coach.CountryName as an index, the performance of the query dropped from 13 to 445, 295 respectively. So in the final design, we might build the Athlete.CountryName as an index.

We chose to add an index on Athlete.CountryName and Athlete.Discipline because there are the attributes that we will compare with another table. Since “Athlete” is a table with more than 10,000 rows, a b-tree index can significantly improve the query performance.

We chose to add an index on Coach.CountryName and Coach.Discipline because there are the attributes that we will compare with the table “Athlete”. However, since we need to group by the Coach. name and the table “Coach” only has 1000+ rows, the new index may not be helpful and even make the performance worse.

Finally, we combine the index of Athletes.CountryName and Athlete.Discipline, and get the best performance so far. However, it only improved from 0.67 to 0.61 with the Athlete. The Discipline index is added above the Athlete.CountryName index. So we consider only adding Athlete.CountryName as the index is enough.

Advanced Query2:

SELECT COUNT(a.Name) as totoalAthlete, a.CountryName, c.Total
FROM Olympics.Athlete a JOIN Olympics.Country c ON c.CountryName = a.CountryName
WHERE c.Total >= 20
GROUP BY a.CountryName
ORDER BY c.Total;

```
1 •  SELECT COUNT(a.Name) as totoalAlthete,a.CountryName, c.Total
2    FROM Olympics.Athlete a JOIN Olympics.Country c ON c.CountryName = a.CountryName
3    WHERE c.Total > 10
4    GROUP BY a.CountryName
5    ORDER BY c.Total
6    LIMIT 15;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

totoalAlthete	CountryName	Total
117	Czech Republic	11
103	Denmark	11
67	Chinese Taipei	12
115	Switzerland	13
102	Turkey	13
194	Poland	14
69	Cuba	15
322	Spain	17
152	Ukraine	19
202	New Zealand	20
222	Republic of Ko...	20
154	Hungary	20
291	Brazil	21
368	Canada	24
377	France	33

Justification:

We use the group by and join operation in this query.
This query will return Country and number of total athletes from that country sorted by the total medal counts. It could provide insight for user to find the connection between total medals and total athletes, such as the question “Will the country with more athletes get more medals?”. In our application, we also will include the pie chart to show the proportion of total athletes/medals from each country.

Analysis for advanced query 2:

Indexes before of table Athlete:

```
1    SHOW INDEX FROM Olympics.Athlete
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
Athlete	0	PRIMARY	1	Name	A	10953	NULL	NULL		BTREE

Indexes before of table Country:

```
1 •  SHOW INDEX FROM Olympics.Country;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
Country	0	PRIMARY	1	CountryName	A	94	NULL	NULL		BTREE

Initial runtime, runtime without adding any extra indices:

```
1 •  EXPLAIN ANALYZE SELECT COUNT(a.Name) as totoalAlthete,a.CountryName, c.Total
2    FROM Olympics.Athlete a JOIN Olympics.Country c ON c.CountryName = a.CountryName
3    WHERE c.Total >= 20
4    GROUP BY a.CountryName
5    ORDER BY c.Total
6    LIMIT 15
```

Form Editor | Navigate: 1 / 1

EXPLAIN:

```
-> Limit: 15 row(s)  (actual time=21.272..21.274 rows=15 loops=1)
  -> Sort: Olympics.c.Total, limit input to 15 row(s) per chunk  (actual time=21.271..21.272 rows=15 loops=1)
    -> Table scan on <temporary>  (actual time=21.253..21.255 rows=15 loops=1)
      -> Aggregate using temporary table  (actual time=21.251..21.251 rows=15 loops=1)
```

New Indexes added to Athlete, we add b-tree index on Athlete.CountryName:

```
1   SHOW INDEX FROM Olympics.Athlete
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
Athlete	0	PRIMARY	1	Name	A	10953	NULL	NULL		BTREE
Athlete	1	btreeathlete	1	CountryName	A	207	NULL	NULL		BTREE

New Runtime: 5.038, much faster than before

```
1   EXPLAIN ANALYZE SELECT COUNT(a.Name) as totoalAlthete,a.CountryName, c.Total
2   FROM Olympics.Athlete a JOIN Olympics.Country c ON c.CountryName = a.CountryName
3   WHERE c.Total >= 20
4   GROUP BY a.CountryName
5   ORDER BY c.Total
6   LIMIT 15
```

Form Editor | Navigate:

EXPLAIN:

```
-> Limit: 15 row(s)  (actual time=5.038..5.039 rows=15 loops=1)
  -> Sort: Olympics.c.Total, limit input to 15 row(s) per chunk  (actual time=5.037..5.038 rows=15 loops=1)
    -> Table scan on <temporary>  (actual time=5.019..5.023 rows=15 loops=1)
      -> Aggregate using temporary table  (actual time=5.018..5.018 rows=15 loops=1)
```

Added new index to Country: add b-tree index to Country.Total

```
1    SHOW INDEX FROM Olympics.Country
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
Country	0	PRIMARY	1	CountryName	A	94	NULL	NULL		BTREE
Country	1	btree	1	Total	A	31	NULL	NULL	YES	BTREE

New Runtime: 5.055, did not improve runtime significantly.

```
1 •  EXPLAIN ANALYZE SELECT COUNT(a.Name) as totoalAlthete,a.CountryName, c.Total
2    FROM Olympics.Athlete a JOIN Olympics.Country c ON c.CountryName = a.CountryName
3    WHERE c.Total >= 20
4    GROUP BY a.CountryName
5    ORDER BY c.Total
6    LIMIT 15
```

Form Editor | Navigate:

EXPLAIN:

```
-> Limit: 15 row(s)  (actual time=5.055..5.057 rows=15 loops=1)
  -> Sort: Olympics.c.Total, limit input to 15 row(s) per chunk  (actual time=5.054..5.055 rows=15 loops=1)
    -> Table scan on <temporary>  (actual time=5.033..5.038 rows=15 loops=1)
      -> Aggregate using temporary table  (actual time=5.031..5.031 rows=15 loops=1)
```

Remove the index on Athlete.CountryName and check runtime again with only additional index on Country.Total, the query becomes as slow as the initial state:

```
1 •  EXPLAIN ANALYZE SELECT COUNT(a.Name) as totoalAlthete,a.CountryName, c.Total
2    FROM Olympics.Athlete a JOIN Olympics.Country c ON c.CountryName = a.CountryName
3    WHERE c.Total >= 20
4    GROUP BY a.CountryName
5    ORDER BY c.Total
6    LIMIT 15
```

Form Editor | Navigate:

EXPLAIN:

```
-> Limit: 15 row(s)  (actual time=21.106..21.108 rows=15 loops=1)
  -> Sort: Olympics.c.Total, limit input to 15 row(s) per chunk  (actual time=21.105..21.106 rows=15 loops=1)
    -> Table scan on <temporary>  (actual time=21.088..21.091 rows=15 loops=1)
      -> Aggregate using temporary table  (actual time=21.087..21.087 rows=15 loops=1)
```

Results and explanation: By using Athlete.CountryName as an index, we were able to speed up the run time of the query from 21 to 5 units of time.
We choose to add an index on Athlete.CountryName because this is the only attribute that is not a primary key besides Country.total used in this advanced query, and “Athlete” is a table with more than 10000 rows, thus a b-tree index can significantly improve the query performance.

By using Total as an index, there is not much difference between the run time of the query.
We think this change in indexing of Country.Total did not bring a better effect to the query because “Country” is a table with only 94 rows which is too small for index to bring benefits.