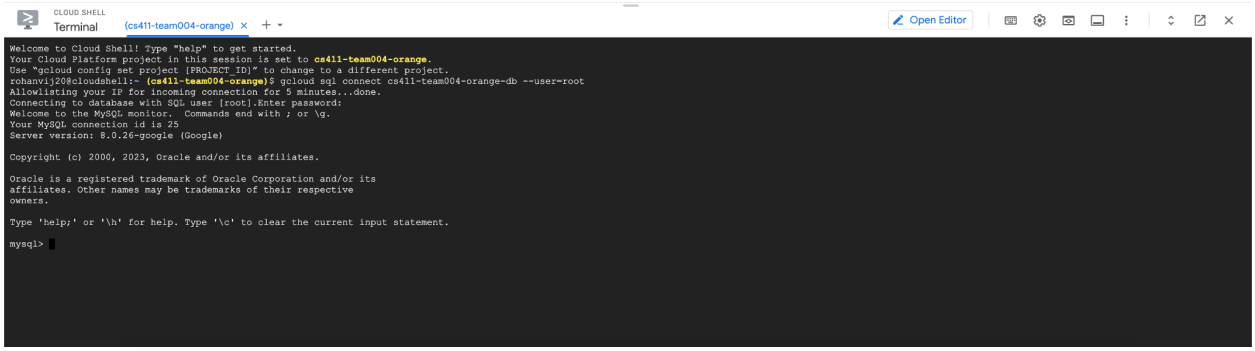
Screenshot of the connection (i.e. showing your terminal/command-line information):



Provided the DDL commands for your tables (-0.5% for each mistake):

```
CREATE TABLE Athletes
(
   Ath_Name VARCHAR(512) PRIMARY KEY,
   NOC VARCHAR(512),
   Discipline VARCHAR(512) REFERENCES Sport(Discipline),
   Age INT,
   Salary  INT
);
```

```
CREATE TABLE Coaches
(
   Coa_Name VARCHAR(512) PRIMARY KEY,
   NOC VARCHAR(512),
   Discipline  VARCHAR(512),
   Event   VARCHAR(512)
);
```

```
CREATE TABLE Sport
(
   Discipline  VARCHAR(512) PRIMARY KEY,
   Avg_Salary  INT,
   Avg_Age INT
);
```

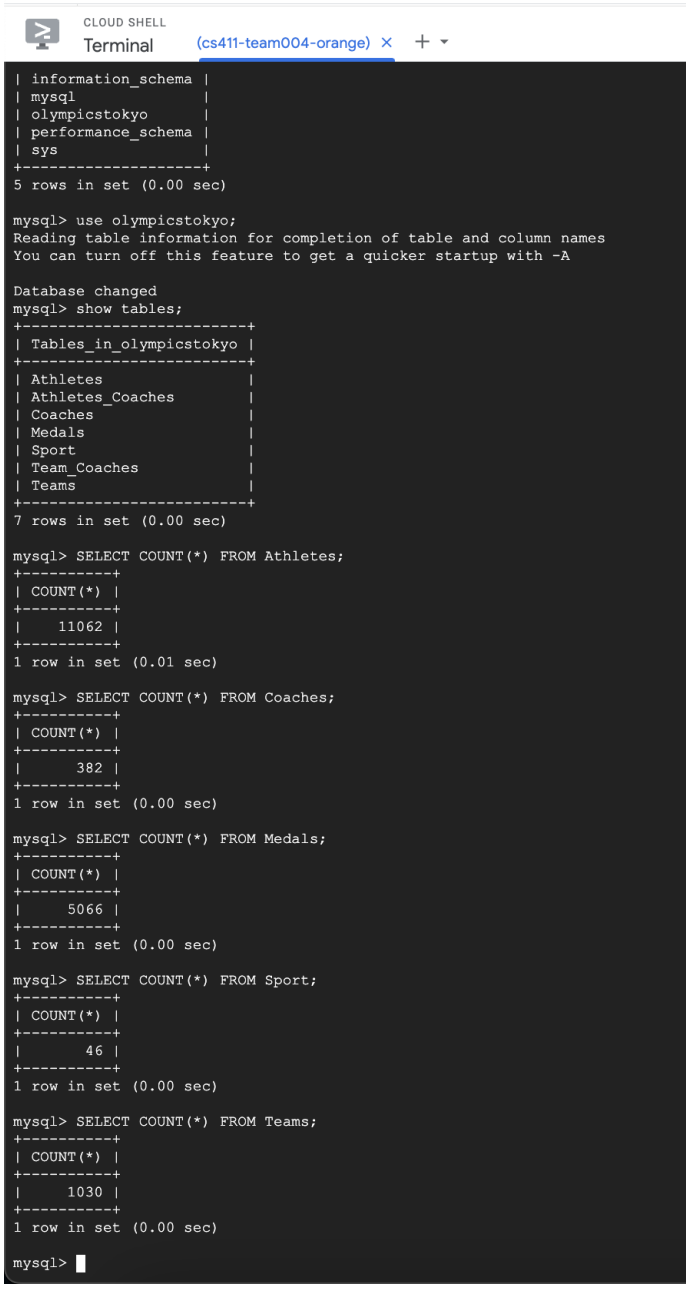
```
CREATE TABLE Teams
(
   Name CHAR(100),
   Discipline CHAR(100),
   NOC CHAR(100),
   Event VARCHAR(512),
```

```
   PRIMARY KEY (NOC, Discipline, Event),
   FOREIGN KEY (Discipline) REFERENCES Sport(Discipline) ON DELETE CASCADE
);
```

```
CREATE TABLE Medals (
 Ath_Name CHAR(100),
 NOC CHAR(100),
 Num_total INT,
 Num_Gold INT,
 Num_Silver INT,
 Num_Brozen INT,
 PRIMARY KEY (Ath_Name, NOC),
 FOREIGN KEY (Ath_Name) REFERENCES Athletes(Ath_Name)
);
```

```
CREATE TABLE Athletes_Coaches (
   Ath_Name CHAR(100),
   Coa_Name CHAR(100),

   PRIMARY KEY (Ath_Name, Coa_Name),
   FOREIGN KEY (Ath_Name) REFERENCES Athletes(Ath_Name),
   FOREIGN KEY (Coa_Name) REFERENCES Coaches(Coa_Name)
);

CREATE TABLE Team_Coaches (
   Coa_Name CHAR(100),
   NOC CHAR(100),
   Discipline CHAR(100),
   Event CHAR(100),

   PRIMARY KEY (Coa_Name, NOC, Discipline, Event),

   FOREIGN KEY (Coa_Name) REFERENCES Coaches(Coa_Name),
   FOREIGN KEY (NOC) REFERENCES Teams(NOC),
   FOREIGN KEY (Discipline) REFERENCES Teams(Discipline)
);
```

Inserted at least 1000 rows in the tables
You should upload a screenshot of a count query to show this, 1% for each table
You should insert at least 1000 rows each in three of the tables:



Two advanced queries:

-- Query 1:

SELECT DISTINCT Athletes.Ath_Name, Athletes.Discipline, Athletes.NOC, Coaches.Event, Teams.Name
FROM Athletes
JOIN Coaches ON Athletes.NOC = Coaches.NOC
JOIN Teams ON Teams.NOC = Athletes.NOC AND Teams.Discipline = Athletes.Discipline AND Teams.Event = Coaches.Event

```
WHERE Athletes.Age > 30
AND Athletes.Salary > (SELECT AVG(Avg_Salary) FROM Sport WHERE Discipline =
Athletes.Discipline);
```

-- Query 1 calculates the names, disciplines, NOCs, events, and teams of all athletes over the age of 30 who are making more than the average salary for their discipline.

```
mysql> SELECT DISTINCT Athletes.Ath_Name, Athletes.Discipline, Athletes.NOC, Coaches.Event, Teams.Name
    -> FROM Athletes
    -> JOIN Coaches ON Athletes.NOC = Coaches.NOC
    -> JOIN Teams ON Teams.NOC = Athletes.NOC AND Teams.Discipline = Athletes.Discipline AND Teams.Event = Coaches.Event
    -> WHERE Athletes.Age > 30
    -> AND Athletes.Salary > (SELECT AVG(Avg_Salary) FROM Sport WHERE Discipline = Athletes.Discipline)
    -> LIMIT 15;
+--------------------------+------------------+--------------+----------+--------------+
| Ath_Name                 | Discipline       | NOC          | Event    | Name         |
+--------------------------+------------------+--------------+----------+--------------+
| ABALDE Alberto           | Basketball       | Spain        | Women    | Spain        |
| ABALDE Alberto           | Basketball       | Spain        | Men      | Spain        |
| ABBINGH Lois             | Handball         | Netherlands  | Women    | Netherlands  |
| ABRAHAMS Rusten John     | Hockey           | South Africa | Women    | South Africa |
| ABRAHAMS Rusten John     | Hockey           | South Africa | Men      | South Africa |
| ABRINES REDONDO Alejandro | Basketball       | Spain        | Women    | Spain        |
| ABRINES REDONDO Alejandro | Basketball       | Spain        | Men      | Spain        |
| ACHIUWA Precious         | Basketball       | Nigeria      | Women    | Nigeria      |
| ACHIUWA Precious         | Basketball       | Nigeria      | Men      | Nigeria      |
| ADEL Ibrahim             | Football         | Egypt        | Men      | Egypt        |
| AEDO Yanara              | Football         | Chile        | Women    | Chile        |
| AGATSUMA Haruka          | Baseball/Softball | Japan        | Softball | Japan        |
| AGATSUMA Haruka          | Baseball/Softball | Japan        | Baseball | Japan        |
| AGUIRRE Eduardo          | Football         | Mexico       | Men      | Mexico       |
| AKOZ Simge Sebnem        | Volleyball       | Turkey       | Women    | Turkey       |
+--------------------------+------------------+--------------+----------+--------------+
15 rows in set (0.01 sec)
```

Image of the first 15 rows for query 1

-- Query 2:

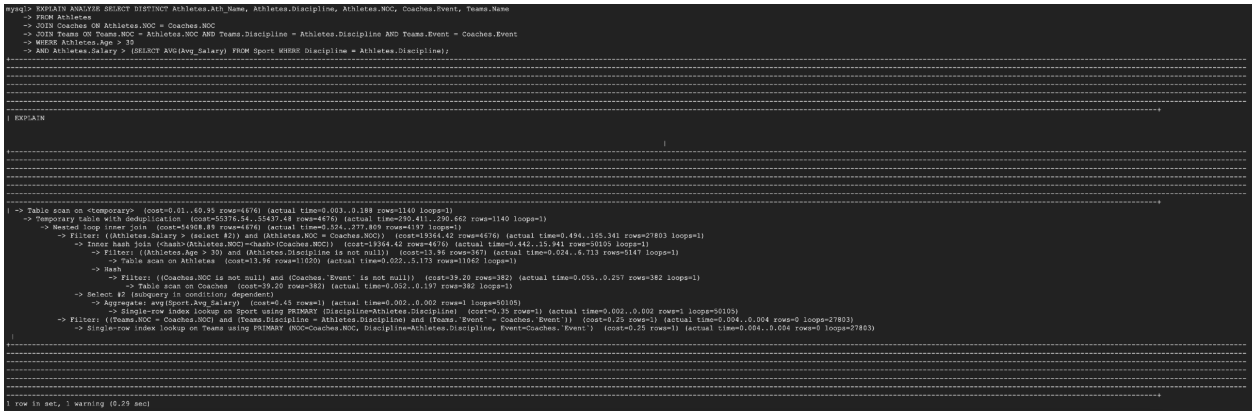
```
SELECT Coaches.Coa_Name, Coaches.Discipline, Coaches.Event, SUM(Medals.Num_Gold)
AS Total_Gold
FROM Coaches
JOIN Medals ON Coaches.NOC = Medals.NOC
JOIN Teams ON Coaches.NOC = Teams.NOC AND Coaches.Discipline = Teams.Discipline
AND Coaches.Event = Teams.Event
GROUP BY Coaches.Coa_Name, Coaches.Discipline, Coaches.Event
HAVING SUM(Medals.Num_Gold) > 5;
```

-- Query 2: This query selects the coaches' names, disciplines, events, and the total number of gold medals won by teams coached by each coach. It only returns results for coaches who have won at least 5 gold medals.

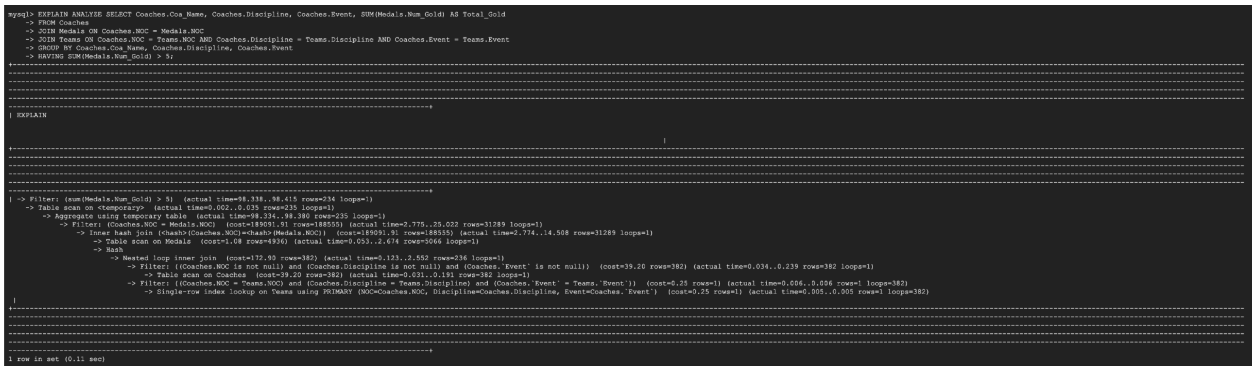
```
mysql>  SELECT Coaches.Coa_Name, Coaches.Discipline, Coaches.Event, SUM(Medals.Num_Gold) AS Total_Gold
    -> FROM Coaches
    -> JOIN Medals ON Coaches.NOC = Medals.NOC
    -> JOIN Teams ON Coaches.NOC = Teams.NOC AND Coaches.Discipline = Teams.Discipline AND Coaches.Event = Teams.Event
    -> GROUP BY Coaches.Coa_Name, Coaches.Discipline, Coaches.Event
    -> HAVING SUM(Medals.Num_Gold) > 5
    -> LIMIT 15;
+-----------------------+-------------------+----------+------------+
| Coa_Name              | Discipline        | Event    | Total_Gold |
+-----------------------+-------------------+----------+------------+
| TERZIC Zoran          | Volleyball        | Women    |         30 |
| SAVIC Dejan           | Water Polo        | Men      |         30 |
| MALJKOVIC Marina      | Basketball        | Women    |         30 |
| ZAITSEVA Olesia       | Artistic Swimming | Duet     |         72 |
| MEZHENINA Valeriia    | Artistic Swimming | Team     |         72 |
| SACCHETTI Romeo       | Basketball        | Men      |        165 |
| PIZZOLINI Federico    | Baseball/Softball | Softball |        165 |
| PALERMI Giovanna      | Baseball/Softball | Softball |        165 |
| MAZZANTI Davide       | Volleyball        | Women    |        165 |
| GIALLOMBARDO Patrizia | Artistic Swimming | Duet     |        165 |
| FARINELLI Roberta     | Artistic Swimming | Duet     |        165 |
| CAMPAGNA Alessandro   | Water Polo        | Men      |        165 |
| BLENGINI Gianlorenzo  | Volleyball        | Men      |        165 |
| THOMAIDIS Lisa        | Basketball        | Women    |        166 |
| SZAUDER Gabor         | Artistic Swimming | Duet     |        166 |
+-----------------------+-------------------+----------+------------+
15 rows in set (0.10 sec)
```

Image of the first 15 rows for query 2

EXPLAIN ANALYZE BEFORE INDEXING QUERY 1:



EXPLAIN ANALYZE BEFORE INDEXING QUERY 2:

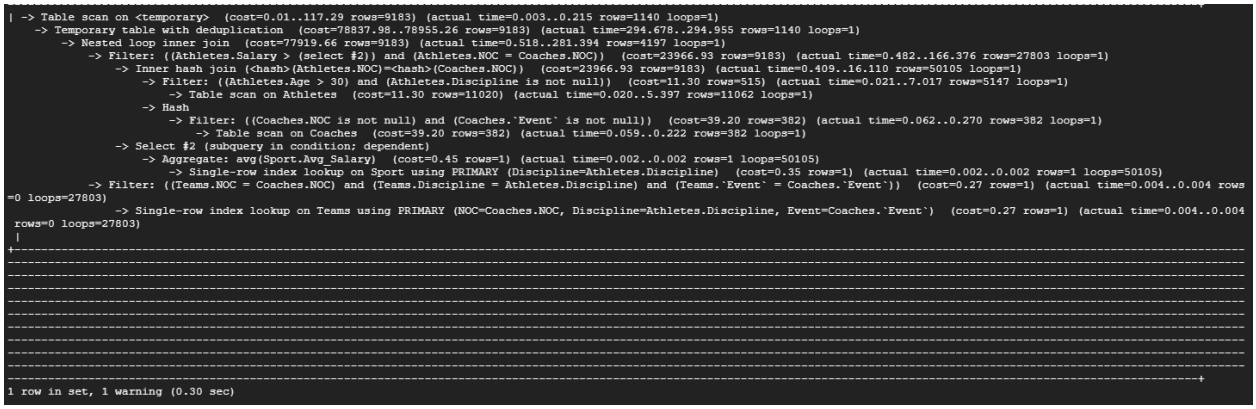


Before indexing, the running time of query 1 is 0.29 seconds and the running time for query 2 is 0.11 seconds from the images above. So the average time cost is (0.29+0.11)/2 = 0.2 second.

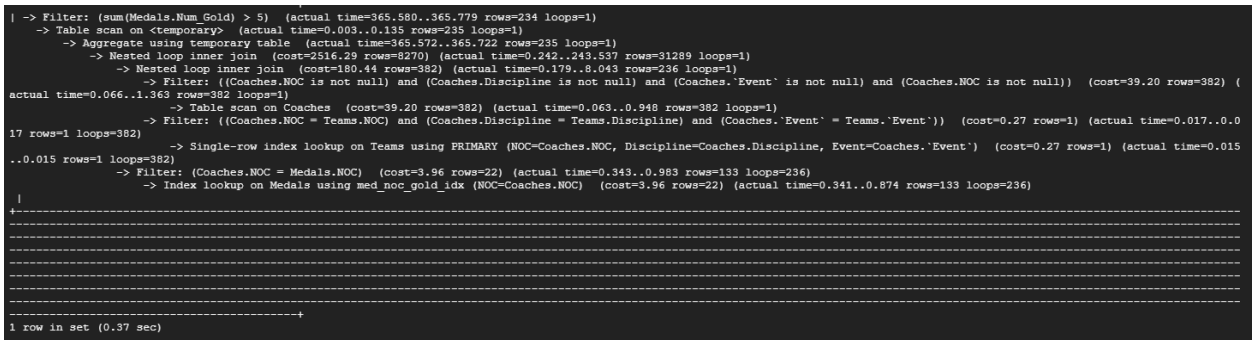
Because the first query manipulates the data from Table Teams and Athletes. The second query manipulates the data from Table Coaches, Medals and Teams. So we will focus on adding indices on those tables to improve the performance of the query.

Index design1:
CREATE INDEX ath_age_salary_idx ON Athletes (Age, Salary);
CREATE INDEX med_noc_gold_idx ON Medals (NOC, Num_Gold);
We try to add indices to the random tables first.

Query1:



Query2:



After adding those indices, the first query takes 0.3 second and the second query takes 0.37 which is even worse. So these indices are bad choice for the performance

Index design2:
CREATE INDEX team_idx ON Teams(NOC,Discipline);
We try to add indices to the table that is used by both queries, which is Teams.

```
| -> Table scan on <temporary>  (cost=0.01..60.95 rows=4676) (actual time=0.005..0.205 rows=1140 loops=1)
    -> Temporary table with deduplication  (cost=55376.54..55437.48 rows=4676) (actual time=308.044..308.311 rows=1140 loops=1)
        -> Nested loop inner join  (cost=54908.89 rows=4676) (actual time=0.509..294.176 rows=4197 loops=1)
            -> Filter: ((Athletes.Salary > (select #2)) and (Athletes.NOC = Coaches.NOC))  (cost=19364.42 rows=4676) (actual time=0.474..173.486 rows=27803 loops=1)
                -> Inner hash join (<hash>(Athletes.NOC)=<hash>(Coaches.NOC))  (cost=19364.42 rows=4676) (actual time=0.405..16.802 rows=50105 loops=1)
                    -> Filter: ((Athletes.Age > 30) and (Athletes.Discipline is not null))  (cost=13.96 rows=367) (actual time=0.052..7.132 rows=5147 loops=1)
                        -> Table scan on Athletes  (cost=13.96 rows=11020) (actual time=0.050..5.401 rows=11062 loops=1)
                    -> Hash
                        -> Filter: ((Coaches.NOC is not null) and (Coaches.`Event` is not null))  (cost=39.20 rows=382) (actual time=0.058..0.225 rows=382 loops=1)
                            -> Table scan on Coaches  (cost=39.20 rows=382) (actual time=0.055..0.186 rows=382 loops=1)
                -> Select #2 (subquery in condition; dependent)
                    -> Aggregate: avg(Sport.Avg_Salary)  (cost=0.45 rows=1) (actual time=0.002..0.002 rows=1 loops=50105)
                        -> Single-row index lookup on Sport using PRIMARY (Discipline=Athletes.Discipline)  (cost=0.35 rows=1) (actual time=0.002..0.002 rows=1 loops=50105)
            -> Filter: ((Teams.NOC = Coaches.NOC) and (Teams.Discipline = Athletes.Discipline) and (Teams.`Event` = Coaches.`Event`))  (cost=0.25 rows=1) (actual time=0.004..0.004 rows
=0 loops=27803)
                -> Single-row index lookup on Teams using PRIMARY (NOC=Coaches.NOC, Discipline=Athletes.Discipline, Event=Coaches.`Event`)  (cost=0.25 rows=1) (actual time=0.004..0.004
 rows=0 loops=27803)
 |
+--------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set, 1 warning (0.31 sec)
```

```
| -> Filter: (sum(Medals.Num_Gold) > 5)  (actual time=97.873..97.952 rows=234 loops=1)
    -> Table scan on <temporary>  (actual time=0.002..0.037 rows=235 loops=1)
        -> Aggregate using temporary table  (actual time=97.868..97.917 rows=235 loops=1)
            -> Filter: (Coaches.NOC = Medals.NOC)  (cost=189091.91 rows=188555) (actual time=2.802..24.969 rows=31289 loops=1)
                -> Inner hash join (<hash>(Coaches.NOC)=<hash>(Medals.NOC))  (cost=189091.91 rows=188555) (actual time=2.801..14.076 rows=31289 loops=1)
                    -> Table scan on Medals  (cost=1.08 rows=4936) (actual time=0.046..2.616 rows=5066 loops=1)
                    -> Hash
                        -> Nested loop inner join  (cost=172.90 rows=382) (actual time=0.120..2.563 rows=236 loops=1)
                            -> Filter: ((Coaches.NOC is not null) and (Coaches.Discipline is not null) and (Coaches.`Event` is not null))  (cost=39.20 rows=382) (actual time=0.051..0.2
57 rows=382 loops=1)
                                -> Table scan on Coaches  (cost=39.20 rows=382) (actual time=0.049..0.209 rows=382 loops=1)
                            -> Filter: ((Coaches.NOC = Teams.NOC) and (Coaches.Discipline = Teams.Discipline) and (Coaches.`Event` = Teams.`Event`))  (cost=0.25 rows=1) (actual time=0.
006..0.006 rows=1 loops=382)
                                -> Single-row index lookup on Teams using PRIMARY (NOC=Coaches.NOC, Discipline=Coaches.Discipline, Event=Coaches.`Event`)  (cost=0.25 rows=1) (actual ti
me=0.005..0.005 rows=1 loops=382)
 |
+--------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------+
1 row in set (0.10 sec)
```

At this time, the second query takes 0.1 second which is faster than 0.11 second before, but the first query takes a longer time as well which is 0.31 second. So the average time cost is (0.31+0.1)/2 = 0.205 that the performance hasn't changed a lot. So it's still not a good choice.

Index design3
CREATE INDEX salary_idx ON Athletes (Salary);
CREATE INDEX coa_idx ON Coaches(Discipline)/CREATE INDEX coa_idx ON Coaches(NOC);
We take a closer look at the two query and identify table Coaches need more indices due to the GROUP BY implementation in query2. We also add index to the Salary in athletes table due to the comparison implementation in query1.

```
| -> Table scan on <temporary>  (cost=0.01..285.38 rows=22630) (actual time=0.003..0.211 rows=1140 loops=1)
    -> Temporary table with deduplication  (cost=19230.43..19515.79 rows=22630) (actual time=200.038..200.311 rows=1140 loops=1)
        -> Nested loop inner join  (cost=16967.40 rows=22630) (actual time=0.219..187.318 rows=4197 loops=1)
            -> Nested loop inner join  (cost=9046.82 rows=22630) (actual time=0.192..80.526 rows=27803 loops=1)
                -> Filter: ((Athletes.Age > 30) and (Athletes.Salary > (select #2)) and (Athletes.NOC is not null) and (Athletes.Discipline is not null))  (cost=1126.25 rows=3673) (act
ual time=0.084..22.641 rows=2840 loops=1)
                    -> Table scan on Athletes  (cost=1126.25 rows=11020) (actual time=0.039..4.887 rows=11062 loops=1)
                    -> Select #2 (subquery in condition; dependent)
                        -> Aggregate: avg(Sport.Avg_Salary)  (cost=0.45 rows=1) (actual time=0.002..0.002 rows=1 loops=5147)
                            -> Single-row index lookup on Sport using PRIMARY (Discipline=Athletes.Discipline)  (cost=0.35 rows=1) (actual time=0.002..0.002 rows=1 loops=5147)
                -> Filter: (Coaches.`Event` is not null)  (cost=1.54 rows=6) (actual time=0.010..0.020 rows=10 loops=2840)
                    -> Index lookup on Coaches using coa_idx (NOC=Athletes.NOC)  (cost=1.54 rows=6) (actual time=0.010..0.019 rows=10 loops=2840)
            -> Filter: ((Teams.NOC = Athletes.NOC) and (Teams.Discipline = Athletes.Discipline) and (Teams.`Event` = Coaches.`Event`))  (cost=0.25 rows=1) (actual time=0.004..0.004 row
s=0 loops=27803)
                -> Single-row index lookup on Teams using PRIMARY (NOC=Athletes.NOC, Discipline=Athletes.Discipline, Event=Coaches.`Event`)  (cost=0.25 rows=1) (actual time=0.003..0.00
3 rows=0 loops=27803)
 |
+--------------------------------------------------------------------------------------------------------------------------------------------------------
---------+
1 row in set, 1 warning (0.20 sec)
```

```
| -> Filter: (sum(Medals.Num_Gold) > 5)  (actual time=120.109..120.194 rows=234 loops=1)
    -> Table scan on <temporary>  (actual time=0.003..0.044 rows=235 loops=1)
        -> Aggregate using temporary table  (actual time=120.104..120.158 rows=235 loops=1)
            -> Filter: (Coaches.NOC = Medals.NOC)  (cost=23968.96 rows=22231) (actual time=24.530..46.730 rows=31289 loops=1)
                -> Inner hash join (<hash>(Coaches.NOC)=<hash>(Medals.NOC))  (cost=23968.96 rows=22231) (actual time=24.529..36.038 rows=31289 loops=1)
                    -> Table scan on Medals  (cost=2.53 rows=4936) (actual time=0.062..2.627 rows=5066 loops=1)
                    -> Hash
                        -> Nested loop inner join  (cost=1672.97 rows=45) (actual time=3.238..24.255 rows=236 loops=1)
                            -> Index scan on Teams using Discipline  (cost=96.60 rows=731) (actual time=0.068..0.475 rows=731 loops=1)
                            -> Filter: ((Coaches.Discipline = Teams.Discipline) and (Coaches.`Event` = Teams.`Event`))  (cost=1.54 rows=0) (actual time=0.029..0.032 rows=0 loops=731)
                                -> Index lookup on Coaches using coa_idx (NOC=Teams.NOC), with index condition: (Coaches.NOC = Teams.NOC)  (cost=1.54 rows=6) (actual time=0.014..0.030
rows=11 loops=731)
 |
+---------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------
----------------------------+
1 row in set (0.12 sec)
```

```
| -> Filter: (sum(Medals.Num_Gold) > 5)  (actual time=96.655..96.732 rows=234 loops=1)
    -> Table scan on <temporary>  (actual time=0.002..0.036 rows=235 loops=1)
        -> Aggregate using temporary table  (actual time=96.651..96.697 rows=235 loops=1)
            -> Filter: (Coaches.NOC = Medals.NOC)  (cost=189091.91 rows=188555) (actual time=2.744..24.178 rows=31289 loops=1)
                -> Inner hash join (<hash>(Coaches.NOC)=<hash>(Medals.NOC))  (cost=189091.91 rows=188555) (actual time=2.744..13.685 rows=31289 loops=1)
                    -> Table scan on Medals  (cost=1.08 rows=4936) (actual time=0.037..2.514 rows=5066 loops=1)
                    -> Hash
                        -> Nested loop inner join  (cost=172.90 rows=382) (actual time=0.098..2.556 rows=236 loops=1)
                            -> Filter: ((Coaches.NOC is not null) and (Coaches.Discipline is not null) and (Coaches.`Event` is not null))  (cost=39.20 rows=382) (actual time=0.033..0.2
41 rows=382 loops=1)
                                -> Table scan on Coaches  (cost=39.20 rows=382) (actual time=0.031..0.192 rows=382 loops=1)
                            -> Filter: ((Coaches.NOC = Teams.NOC) and (Coaches.Discipline = Teams.Discipline) and (Coaches.`Event` = Teams.`Event`))  (cost=0.25 rows=1) (actual time=0.
006..0.006 rows=1 loops=382)
                                -> Single-row index lookup on Teams using PRIMARY (NOC=Coaches.NOC, Discipline=Coaches.Discipline, Event=Coaches.`Event`)  (cost=0.25 rows=1) (actual ti
me=0.005..0.005 rows=1 loops=382)
 |
+---------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------+
1 row in set (0.10 sec)
```

The time cost for the first query decreases a lot from 0.29 to 0.2 seconds. But the second query doesn't change the time cost which means adding an index to table Coaches doesn't improve the performance of query2.

In addition, We also try to add an index to the key in the Table Medals. But the time cost of the query is still about 0.1 seconds.

To conclude, we find adding indices to the Athletes table will make query1 perform better. But we can't find a better index to make query2 work better.

The reason why some of the indexing will make databases slower is

1. Indexes can cause overhead: The more indexes you have on a table, the more overhead there is in terms of storage space and processing power required to maintain those indexes. This overhead can slow down queries.

2. Index selection: The query optimizer must select the best index to use for a given query. When there are many indexes on a table, the optimizer has more options to consider, which can increase the time required to choose the best index.