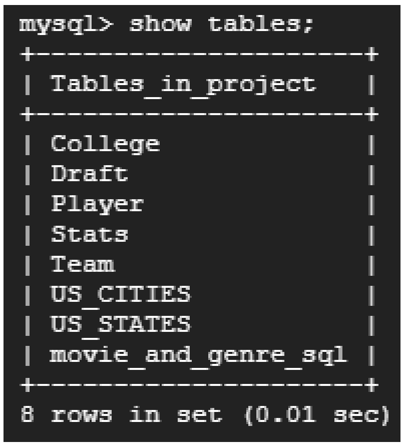
Create Table:

```
CREATE TABLE US_STATES (

      ID int(11) NOT NULL AUTO_INCREMENT,

      STATE_CODE char(2) NOT NULL,

      STATE_NAME varchar(50) NOT NULL,

      PRIMARY KEY (ID)

);



CREATE TABLE US_CITIES (

      ID int(11) NOT NULL AUTO_INCREMENT,

      ID_STATE int(11) NOT NULL,

      CITY varchar(50) NOT NULL,

      COUNTY varchar(50) NOT NULL,

      LATITUDE double NOT NULL,

      LONGITUDE double NOT NULL,

      PRIMARY KEY (ID),

   FOREIGN KEY (ID_STATE) REFERENCES
US_STATES(ID)
);



CREATE TABLE College (

   Name varchar(255)

   Address varchar(255),

   Population int,

   City varchar(255),

   State varchar(255),

   Country varchar(255),

   Zip int

);
```

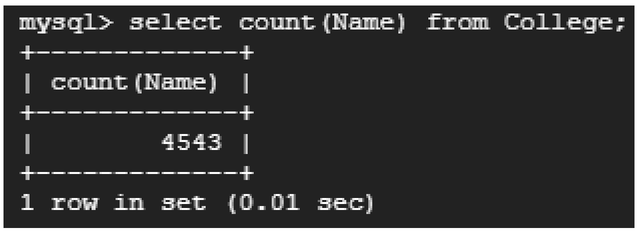
```
create table Stats(

 StatsID int primary key,

 MIN real,

 PTS real,

 FG real,

 ThreePT real,

 REB real,

 AST real,

 STL real,

 BLK real

);


create table Team(

 TeamName varchar(255) primary key,

 Stadium varchar(255),

 CityID int,

 Seat_Cap int,

 Open_year int,

 FOREIGN KEY (CityID) REFERENCES
US_CITIES(ID)
);


create table movie_and_genre_sql(

 id int primary key,

 movie varchar(255) NOT NULL,

 genre varchar(255) NOT NULL

);


create table Draft(

 Draft_ID varchar(255) primary key,
 Draft_year varchar(255),

 Draft_round varchar(255),

 Draft_number varchar(255),

);


create table Player(

 Player_ID int primary key,

 Name varchar(255),

 Height varchar(10),

 Weight real,

 Age int,

 Country varchar(255),

 TeamName varchar(255),

 CollegeName varchar(255),

 Movie_ID int

);
```

Tables:



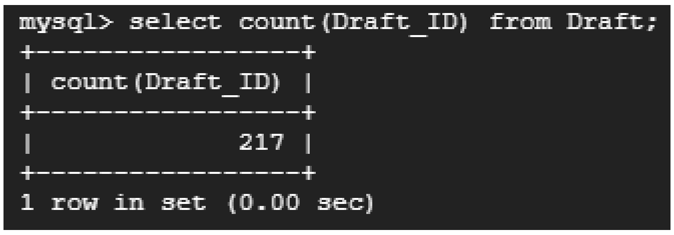
```
mysql> show tables;
+----------------------+
| Tables_in_project    |
+----------------------+
| College              |
| Draft                |
| Player               |
| Stats                |
| Team                 |
| US_CITIES            |
| US_STATES            |
| movie_and_genre_sql  |
+----------------------+
8 rows in set (0.01 sec)
```

College:



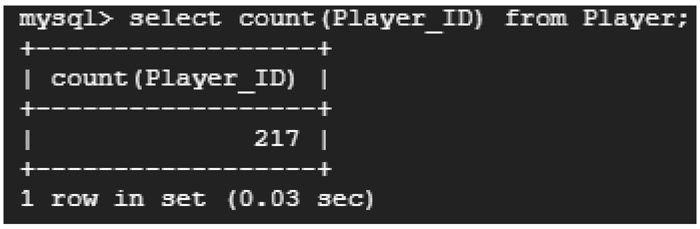
```
mysql> select count(Name) from College;
+-------------+
| count(Name) |
+-------------+
|        4543 |
+-------------+
1 row in set (0.01 sec)
```

Draft:



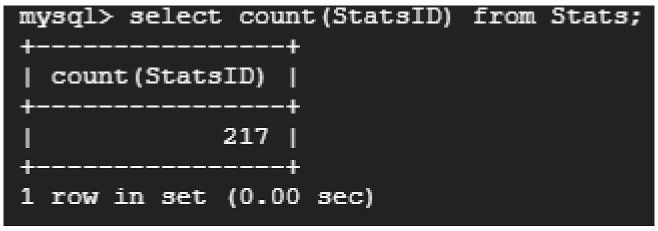
```
mysql> select count(Draft_ID) from Draft;
+-----------------+
| count(Draft_ID) |
+-----------------+
|             217 |
+-----------------+
1 row in set (0.00 sec)
```

Player:



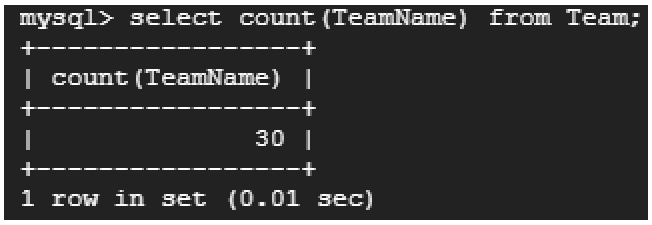
```
mysql> select count(Player_ID) from Player;
+------------------+
| count(Player_ID) |
+------------------+
|              217 |
+------------------+
1 row in set (0.03 sec)
```

Stats:



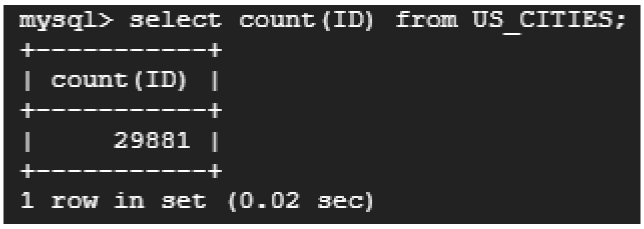
```
mysql> select count(StatsID) from Stats;
+----------------+
| count(StatsID) |
+----------------+
|            217 |
+----------------+
1 row in set (0.00 sec)
```

Team:



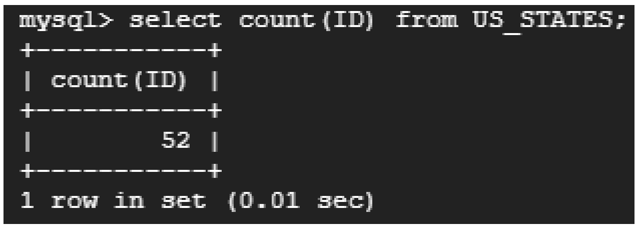
```
mysql> select count(TeamName) from Team;
+-----------------+
| count(TeamName) |
+-----------------+
|              30 |
+-----------------+
1 row in set (0.01 sec)
```

US_CITIES:



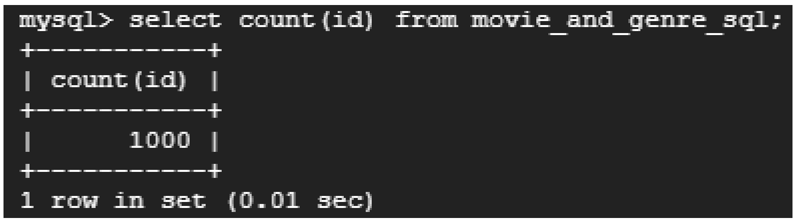
```
mysql> select count(ID) from US_CITIES;
+-----------+
| count(ID) |
+-----------+
|     29881 |
+-----------+
1 row in set (0.02 sec)
```

US_STATES:



```
mysql> select count(ID) from US_STATES;
+-----------+
| count(ID) |
+-----------+
|        52 |
+-----------+
1 row in set (0.01 sec)
```

movie_and_genre_sql

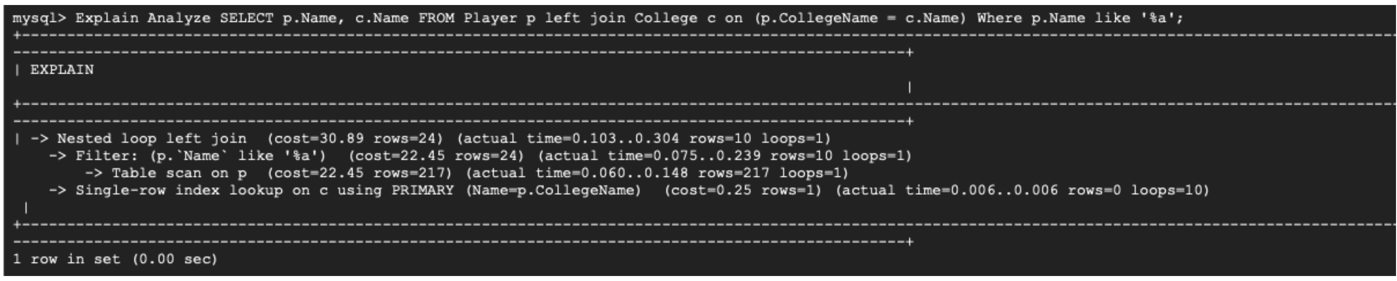


```
mysql> select count(id) from movie_and_genre_sql;
+-----------+
| count(id) |
+-----------+
|      1000 |
+-----------+
1 row in set (0.01 sec)
```

Query1:

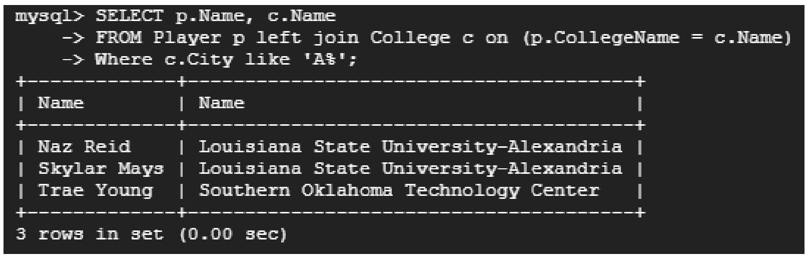
```
SELECT p.Name, c.Name
FROM Player p left join College c on (p.CollegeName = c.Name)
Where c.City like 'A%';
create unique index index1 on Player(Name);
```

Before Indexing:



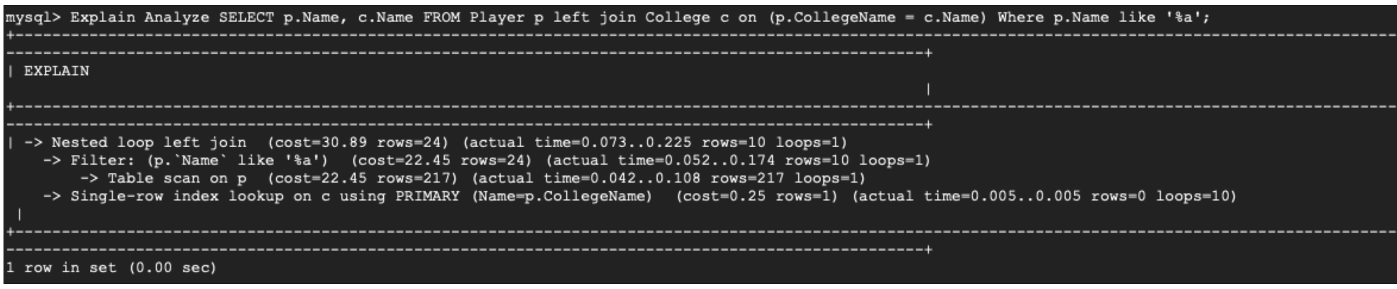
```
mysql> Explain Analyze SELECT p.Name, c.Name FROM Player p left join College c on (p.CollegeName = c.Name) Where p.Name like '%a';
+------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------+
| EXPLAIN
                                                                                         |
+------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------+
| -> Nested loop left join  (cost=30.89 rows=24) (actual time=0.103..0.304 rows=10 loops=1)
    -> Filter: (p.`Name` like '%a')  (cost=22.45 rows=24) (actual time=0.075..0.239 rows=10 loops=1)
        -> Table scan on p  (cost=22.45 rows=217) (actual time=0.060..0.148 rows=217 loops=1)
    -> Single-row index lookup on c using PRIMARY (Name=p.CollegeName)  (cost=0.25 rows=1) (actual time=0.006..0.006 rows=0 loops=10)
 |
+------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

Query result:



```
mysql> SELECT p.Name, c.Name
    -> FROM Player p left join College c on (p.CollegeName = c.Name)
    -> Where c.City like 'A%';
+-------------+----------------------------------------+
| Name        | Name                                   |
+-------------+----------------------------------------+
| Naz Reid    | Louisiana State University-Alexandria |
| Skylar Mays | Louisiana State University-Alexandria |
| Trae Young  | Southern Oklahoma Technology Center   |
+-------------+----------------------------------------+
3 rows in set (0.00 sec)
```

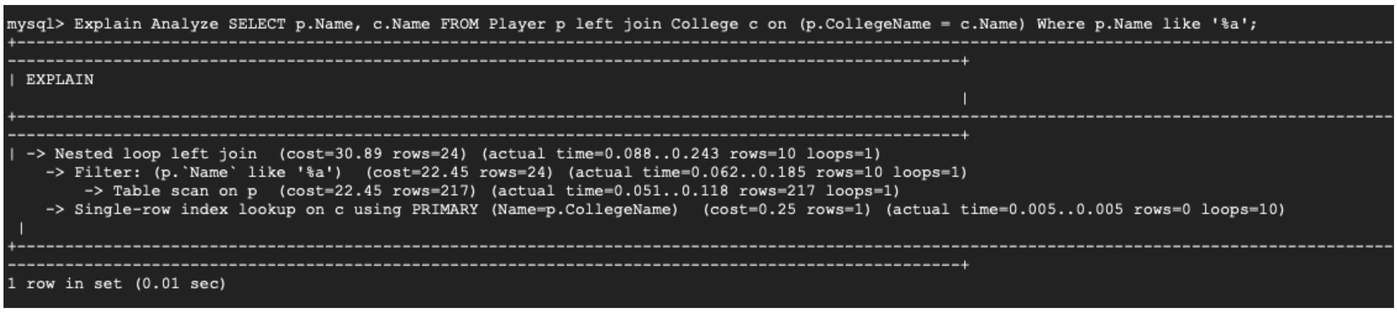
After indexing1:



```
mysql> Explain Analyze SELECT p.Name, c.Name FROM Player p left join College c on (p.CollegeName = c.Name) Where p.Name like '%a';
+------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------+
| EXPLAIN
                                                                                         |
+------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------+
| -> Nested loop left join  (cost=30.89 rows=24) (actual time=0.073..0.225 rows=10 loops=1)
    -> Filter: (p.`Name` like '%a')  (cost=22.45 rows=24) (actual time=0.052..0.174 rows=10 loops=1)
        -> Table scan on p  (cost=22.45 rows=217) (actual time=0.042..0.108 rows=217 loops=1)
    -> Single-row index lookup on c using PRIMARY (Name=p.CollegeName)  (cost=0.25 rows=1) (actual time=0.005..0.005 rows=0 loops=10)
 |
+------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

Analysis: The unique index has functionality that verifies whether the primary keys are unique. In our case, we want to make sure that the keys are unique, so we basically have a “belt and braces” here. Before adding the index, the database needs 0.239 seconds to finish the search. After adding the index, the database needs 0.174 seconds to finish the search, which makes the runtime get a lot lower.

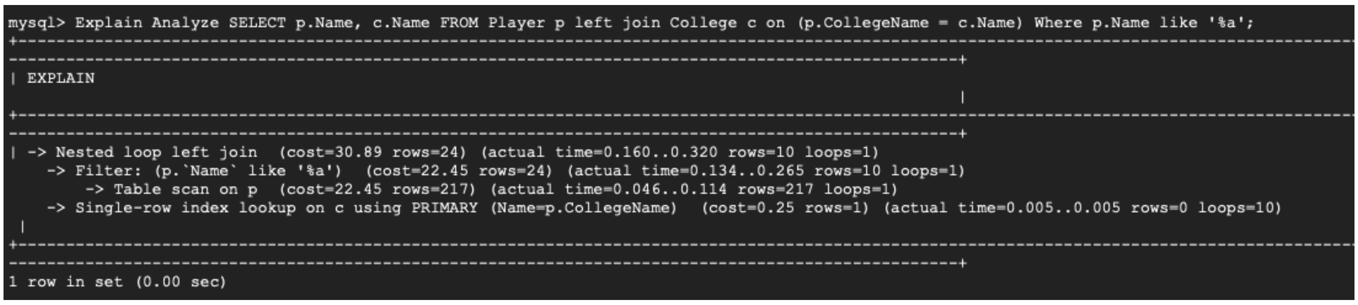
create index index2 on College(City):



```
mysql> Explain Analyze SELECT p.Name, c.Name FROM Player p left join College c on (p.CollegeName = c.Name) Where p.Name like '%a';
+------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------+
| EXPLAIN
                                                                                         |
+------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------+
| -> Nested loop left join  (cost=30.89 rows=24) (actual time=0.088..0.243 rows=10 loops=1)
    -> Filter: (p.`Name` like '%a')  (cost=22.45 rows=24) (actual time=0.062..0.185 rows=10 loops=1)
        -> Table scan on p  (cost=22.45 rows=217) (actual time=0.051..0.118 rows=217 loops=1)
    -> Single-row index lookup on c using PRIMARY (Name=p.CollegeName)  (cost=0.25 rows=1) (actual time=0.005..0.005 rows=0 loops=10)
 |
+------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------+
1 row in set (0.01 sec)
```

When we add an index for the City column in the College table, the runtime doesn’t change a lot. We want to make the database have better runtime on cities, but it doesn’t have a significant improvement and it is probably because the City Column is not sorted.

create index index3 on College(State);

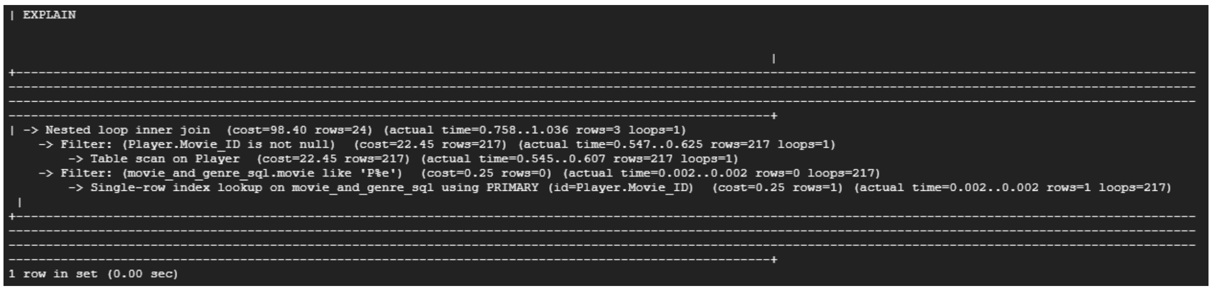


```
mysql> Explain Analyze SELECT p.Name, c.Name FROM Player p left join College c on (p.CollegeName = c.Name) Where p.Name like '%a';
+------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------+
| EXPLAIN
                                                                                                      |
+------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------+
| -> Nested loop left join  (cost=30.89 rows=24) (actual time=0.160..0.320 rows=10 loops=1)
    -> Filter: (p.`Name` like '%a')  (cost=22.45 rows=24) (actual time=0.134..0.265 rows=10 loops=1)
        -> Table scan on p  (cost=22.45 rows=217) (actual time=0.046..0.114 rows=217 loops=1)
    -> Single-row index lookup on c using PRIMARY (Name=p.CollegeName)  (cost=0.25 rows=1) (actual time=0.005..0.005 rows=0 loops=10)
 |
+------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

After adding an index on the State column, the runtime doesn't change (even gets slower). This is probably because the State column has nothing to do with our query because we never used the State in our query. Also, redundant indexes will make the database have worse functionality and it will make commands like UPDATE, INSERT, and DELETE slower. So we should avoid it.

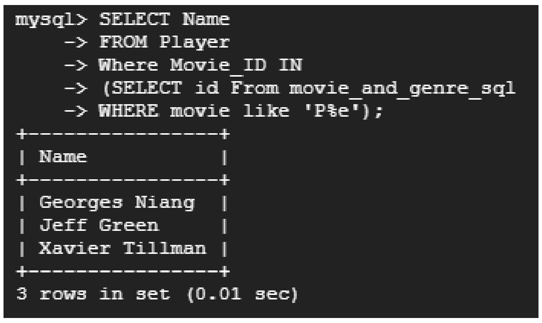
Query2:
SELECT Name
FROM Player
Where Movie_ID IN
(SELECT id From movie_and_genre_sql
WHERE movie like 'P%e');

Before Indexing:



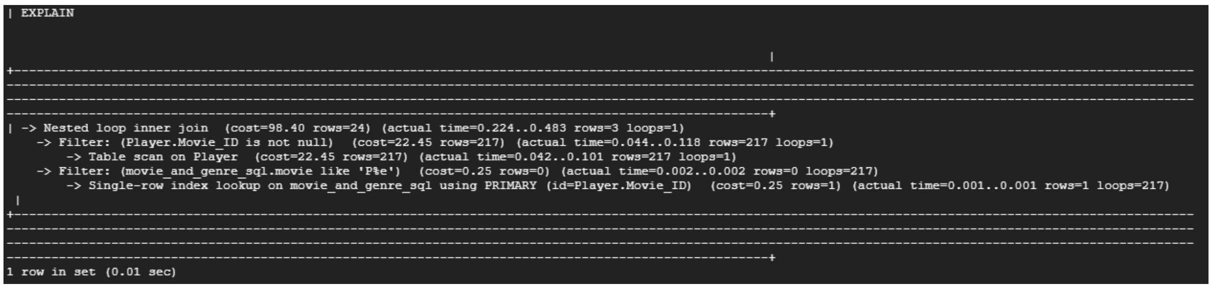
```
| EXPLAIN


                                                                                                      |
+----------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------+
| -> Nested loop inner join  (cost=98.40 rows=24) (actual time=0.758..1.036 rows=3 loops=1)
    -> Filter: (Player.Movie_ID is not null)  (cost=22.45 rows=217) (actual time=0.547..0.625 rows=217 loops=1)
        -> Table scan on Player  (cost=22.45 rows=217) (actual time=0.545..0.607 rows=217 loops=1)
    -> Filter: (movie_and_genre_sql.movie like 'P%e')  (cost=0.25 rows=0) (actual time=0.002..0.002 rows=0 loops=217)
        -> Single-row index lookup on movie_and_genre_sql using PRIMARY (id=Player.Movie_ID)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=217)
 |
+----------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

Query result:



```
mysql> SELECT Name
    -> FROM Player
    -> Where Movie_ID IN
    -> (SELECT id From movie_and_genre_sql
    -> WHERE movie like 'P%e');
+----------------+
| Name           |
+----------------+
| Georges Niang  |
| Jeff Green     |
| Xavier Tillman |
+----------------+
3 rows in set (0.01 sec)
```

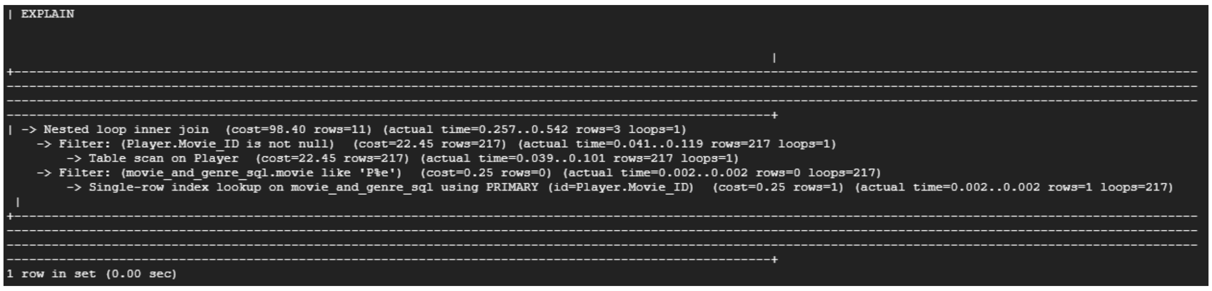
create index idx_name on Player(Name)



```
| EXPLAIN
                                                                                                     |
+--------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------+
| -> Nested loop inner join  (cost=98.40 rows=24) (actual time=0.224..0.483 rows=3 loops=1)
    -> Filter: (Player.Movie_ID is not null)  (cost=22.45 rows=217) (actual time=0.044..0.118 rows=217 loops=1)
        -> Table scan on Player  (cost=22.45 rows=217) (actual time=0.042..0.101 rows=217 loops=1)
    -> Filter: (movie_and_genre_sql.movie like 'P%e')  (cost=0.25 rows=0) (actual time=0.002..0.002 rows=0 loops=217)
        -> Single-row index lookup on movie_and_genre_sql using PRIMARY (id=Player.Movie_ID)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=217)
 |
+--------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------+
1 row in set (0.01 sec)
```

Analysis: The unique index has functionality that verifies whether the primary keys are unique. In our case, we want to make sure that the keys are unique, so we basically have a "belt and braces" here. Before adding the index, the database needs 0.625 seconds to finish the search. After adding the index, the database needs 0.118 seconds to finish the search, which makes the runtime get a lot lower.

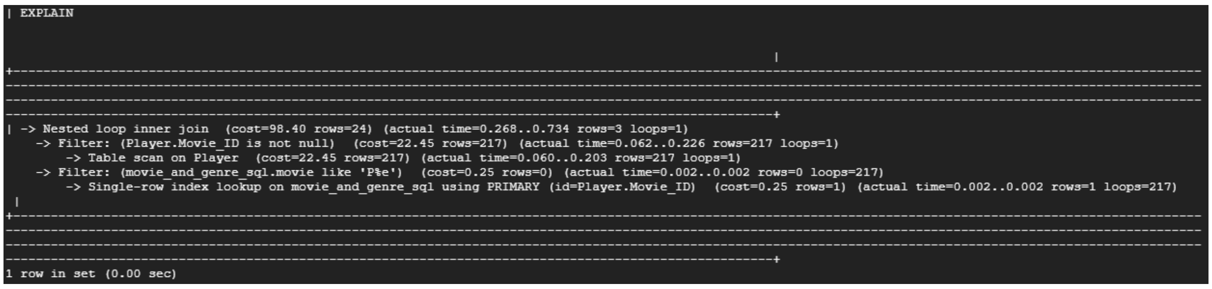
create index idx_movie on movie_and_genre_sql(movie)



```
| EXPLAIN
                                                                                                     |
+--------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------+
| -> Nested loop inner join  (cost=98.40 rows=11) (actual time=0.257..0.542 rows=3 loops=1)
    -> Filter: (Player.Movie_ID is not null)  (cost=22.45 rows=217) (actual time=0.041..0.119 rows=217 loops=1)
        -> Table scan on Player  (cost=22.45 rows=217) (actual time=0.039..0.101 rows=217 loops=1)
    -> Filter: (movie_and_genre_sql.movie like 'P%e')  (cost=0.25 rows=0) (actual time=0.002..0.002 rows=0 loops=217)
        -> Single-row index lookup on movie_and_genre_sql using PRIMARY (id=Player.Movie_ID)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=217)
 |
+--------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

When we add an index for the movie column in the movie_and_genre_sql table, the runtime doesn't change a lot. We want to make the database have better runtime on movies, but it doesn't have a significant improvement and it is probably because the movie Column is not sorted.

create index idx_movie_id on movie_and_genre_sql(id)



```
| EXPLAIN
                                                                                                     |
+--------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------+
| -> Nested loop inner join  (cost=98.40 rows=24) (actual time=0.268..0.734 rows=3 loops=1)
    -> Filter: (Player.Movie_ID is not null)  (cost=22.45 rows=217) (actual time=0.062..0.226 rows=217 loops=1)
        -> Table scan on Player  (cost=22.45 rows=217) (actual time=0.060..0.203 rows=217 loops=1)
    -> Filter: (movie_and_genre_sql.movie like 'P%e')  (cost=0.25 rows=0) (actual time=0.002..0.002 rows=0 loops=217)
        -> Single-row index lookup on movie_and_genre_sql using PRIMARY (id=Player.Movie_ID)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=217)
 |
+--------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

After adding an index on the id column, the runtime gets slower. This is probably because the id column has nothing to do with our query because we never used the id of movie in our query. Also, redundant indexes will make the database have worse functionality and it will make commands like UPDATE, INSERT, and DELETE slower. So we should avoid it.