Q-team107-teamBlue (anishrj2, mehar2, oru2)
CS 411 Section Q

Database Implementation and Indexing

Database implementation was done locally via MySQL. We plan to upload the database to GCP.

Screenshot of Connection (terminal information):

Tables Implemented (at least 4):

- Organization
- Team
- Player
- Game
- BoxScore
- Sign
- PlayerPlay

DDL Commands:

```
CREATE TABLE Organization (
        OrgId INT,
        OrgAbbv VARCHAR(5),
        OrgName VARCHAR(255),
        PRIMARY KEY(OrgId)
);
CREATE TABLE Team (
        TeamId INT,
        Year INT,
        OrgId INT,
        PRIMARY KEY(TeamId),
        FOREIGN KEY (OrgId) REFERENCES Organization(OrgId) ON DELETE CASCADE
);
CREATE TABLE Player (
        PlayerId INT,
        FirstName VARCHAR(255),
        LastName VARCHAR(255),
        Height VARCHAR(255),
        Weight INT,
        PRIMARY KEY(PlayerId)
)
CREATE TABLE Game (
        GameId INT,
        Date Date,
        HomeScore INT,
        AwayScore INT,
        HomeTeamId INT,
        AwayTeamId INT,
        PRIMARY KEY(GameId),
```

```
        FOREIGN KEY(HomeTeamId) REFERENCES Team(TeamId)
        ON DELETE CASCADE,
        FOREIGN KEY(AwayTeamId) REFERENCES Team(TeamId)
        ON DELETE CASCADE
);
CREATE TABLE BoxScore (
        BoxScoreId INT PRIMARY KEY,
        Pts INT,
        Asts INT,
        Rebs INT,
        Blks INT,
        Stls INT,
        TOs INT,
        FGM INT,
        FGA INT,
        TPM INT,
        TPA INT,
        FTM INT,
        FTA INT,
        PlayerId INT,
        GameId INT,
        FOREIGN KEY(PlayerId) REFERENCES Player(PlayerId) ON DELETE CASCADE,
        FOREIGN KEY(GameId) REFERENCES Game(GameId) ON DELETE CASCADE
);
CREATE TABLE Sign (
        TeamId INT,
        PlayerId INT,
        Year INT,
        PRIMARY KEY (TeamId, PlayerId),
        FOREIGN KEY(TeamId) REFERENCES Team(TeamId) ON DELETE CASCADE,
        FOREIGN KEY(PlayerId) REFERENCES Player(PlayerId) ON DELETE CASCADE
);
CREATE TABLE PlayerPlay (
        GameId INT,
        PlayerId INT,
        PRIMARY KEY (GameId, PlayerId),
        FOREIGN KEY(GameId) REFERENCES Game(GameId) ON DELETE CASCADE,
        FOREIGN KEY(PlayerId) REFERENCES Player(PlayerId) ON DELETE CASCADE
);
```

Cardinality of Implemented Tables (at least 3 should have 1000+ entries, shown in bold):

- Organization: 30 entries
- Team: 60 entries
- Player: 773 entries
- Game: **2310 entries**
- BoxScore: **60161 entries**
- Sign: **1009 entries**
- PlayerPlay: **49093 entries**

Cardinality Proof With Count Query:

```
mysql> SELECT COUNT(*) FROM Organization;
+----------+
| COUNT(*) |
+----------+
|       30 |
+----------+
1 row in set (0.05 sec)
```

```
mysql> SELECT COUNT(*) FROM Team;
+----------+
| COUNT(*) |
+----------+
|       60 |
+----------+
1 row in set (0.06 sec)
```

```
mysql> SELECT COUNT(*) FROM Player;
+----------+
| COUNT(*) |
+----------+
|      773 |
+----------+
1 row in set (0.01 sec)
```

```
mysql> SELECT COUNT(*) FROM Game;
+----------+
| COUNT(*) |
+----------+
|     2310 |
+----------+
1 row in set (0.01 sec)
```

```
mysql> SELECT COUNT(*) FROM BoxScore;
+----------+
| COUNT(*) |
+----------+
|    60161 |
+----------+
1 row in set (0.02 sec)
```

```
mysql> SELECT COUNT(*) FROM Sign;
+----------+
| COUNT(*) |
+----------+
|     1009 |
+----------+
1 row in set (0.01 sec)
```

```
mysql> SELECT COUNT(*) FROM PlayerPlay;
+----------+
| COUNT(*) |
+----------+
|    49093 |
+----------+
1 row in set (0.01 sec)
```

Advanced Queries (at least 2):

Advanced Query #1 (uses JOIN and a subquery):

Description: Describes the game in a season in which a player scored their season high

```
SELECT g1.Date, g1.HomeScore, g1.AwayScore, g1.HomeTeamId, g1.AwayTeamId,
b1.Pts
FROM Player p1 JOIN BoxScore b1 USING(PlayerId) JOIN Game g1 USING(GameId)
JOIN Team t1 ON(t1.TeamId = g1.HomeTeamId)
WHERE t1.Year = 2021 AND p1.PlayerId = 2544 AND b1.Pts = (
    SELECT MAX(b2.Pts)
    FROM Player p2 JOIN BoxScore b2 USING(PlayerId) JOIN Game g2
USING(GameId) JOIN Team t2 ON (t2.TeamId = g2.HomeTeamId)
    WHERE p2.PlayerId = 2544 AND t2.Year = 2021
);
```

Advanced Query #2 (uses JOIN, a set operation, and two subqueries):

Description: Gets the number of team wins in a season

```
SELECT COUNT(*)
FROM (
    SELECT g1.GameId
    FROM Team t1 JOIN Game g1 ON(t1.TeamId = g1.HomeTeamId)
    WHERE (g1.HomeScore > g1.AwayScore AND g1.HomeTeamId = 1 AND
t1.Year = 2020)

    UNION

    SELECT g2.GameId
    FROM Team t2 JOIN Game g2 ON(t2.TeamId = g2.AwayTeamId)
    WHERE (g2.AwayScore > g2.HomeScore AND g2.AwayTeamId = 1 AND
t2.Year = 2020)
) AS winning_games;
```

Advanced Queries Output:

Advanced Query #1:
The game where Lebron James (PlayerId = 2544) scored his max in 2021

```
mysql> SELECT g1.Date, g1.HomeScore, g1.AwayScore, g1.HomeTeamId, g1.AwayTeamId, b1.Pts
    -> FROM Player p1 JOIN BoxScore b1 USING(PlayerId) JOIN Game g1 USING(GameId) JOIN Team t1 ON(t1.TeamId = g1.HomeTeamId)
    -> WHERE t1.Year = 2021 AND p1.PlayerId = 2544 AND b1.Pts = (
    -> SELECT MAX(b2.Pts)
    -> FROM Player p2 JOIN BoxScore b2 USING(PlayerId) JOIN Game g2 USING(GameId) JOIN Team t2 ON (t2.TeamId = g2.HomeTeamId)
    -> WHERE p2.PlayerId = 2544 AND t2.Year = 2021 ) ;
+------------+-----------+-----------+------------+------------+------+
| Date       | HomeScore | AwayScore | HomeTeamId | AwayTeamId | Pts  |
+------------+-----------+-----------+------------+------------+------+
| 2022-03-05 |       124 |       116 |         22 |         16 |   56 |
+------------+-----------+-----------+------------+------------+------+
1 row in set (0.00 sec)
```

Advanced Query #2:
The amount of wins by the Atlanta Hawks (TeamId = 1) in 2020

```
mysql> SELECT COUNT(*)
    -> FROM (
    -> SELECT g1.GameId
    -> FROM Team t1 JOIN Game g1 ON(t1.TeamId = g1.HomeTeamId)
    -> WHERE (g1.HomeScore > g1.AwayScore AND g1.HomeTeamId = 1 AND t1.Year = 2020)
    ->
    -> UNION
    ->
    -> SELECT g2.GameId
    -> FROM Team t2 JOIN Game g2 ON(t2.TeamId = g2.AwayTeamId)
    -> WHERE (g2.AwayScore > g2.HomeScore AND g2.AwayTeamId = 1 AND t2.Year = 2020)
    -> ) AS winning_games;
+----------+
| COUNT(*) |
+----------+
|       41 |
+----------+
1 row in set (0.01 sec)
```

Query Performance Before Indexing (using EXPLAIN ANALYZE)**:**

Advanced Query #1:

```
| -> Nested loop inner join  (cost=34.65 rows=1) (actual time=2.658..2.674 rows=1 loops=1)
    -> Nested loop inner join  (cost=30.98 rows=11) (actual time=2.651..2.667 rows=1 loops=1)
        -> Filter: ((b1.Pts = (select #2)) and (b1.GameId is not null))  (cost=27.30 rows=11) (actual time=2.626..2.642 rows=1 loops=1)
            -> Index lookup on b1 using PlayerId (PlayerId=2544)  (cost=27.30 rows=105) (actual time=1.131..1.186 rows=105 loops=1)
            -> Select #2 (subquery in condition; run only once)
                -> Aggregate: max(b2.Pts)  (cost=111.30 rows=1) (actual time=1.339..1.340 rows=1 loops=1)
                    -> Nested loop inner join  (cost=110.25 rows=11) (actual time=0.777..1.314 rows=60 loops=1)
                        -> Nested loop inner join  (cost=73.50 rows=105) (actual time=0.364..0.936 rows=105 loops=1)
                            -> Filter: (b2.GameId is not null)  (cost=36.75 rows=105) (actual time=0.344..0.444 rows=105 loops=1)
                                -> Index lookup on b2 using PlayerId (PlayerId=2544)  (cost=36.75 rows=105) (actual time=0.342..0.409 rows=105 loops=1)
                            -> Filter: (g2.HomeTeamId is not null)  (cost=0.25 rows=1) (actual time=0.004..0.004 rows=1 loops=105)
                                -> Single-row index lookup on g2 using PRIMARY (GameId=b2.GameId)  (cost=0.25 rows=1) (actual time=0.003..0.003 rows=1 loops=105)
                        -> Filter: (t2.`Year` = 2021)  (cost=0.25 rows=0.1) (actual time=0.003..0.003 rows=1 loops=105)
                            -> Single-row index lookup on t2 using PRIMARY (TeamId=g2.HomeTeamId)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=105)
        -> Filter: (g1.HomeTeamId is not null)  (cost=0.26 rows=1) (actual time=0.006..0.006 rows=1 loops=1)
            -> Single-row index lookup on g1 using PRIMARY (GameId=b1.GameId)  (cost=0.26 rows=1) (actual time=0.005..0.005 rows=1 loops=1)
    -> Filter: (t1.`Year` = 2021)  (cost=0.25 rows=0.1) (actual time=0.005..0.005 rows=1 loops=1)
        -> Single-row index lookup on t1 using PRIMARY (TeamId=g1.HomeTeamId)  (cost=0.25 rows=1) (actual time=0.004..0.004 rows=1 loops=1)
 |
```

Advanced Query #2:

```
mysql> EXPLAIN ANALYZE SELECT COUNT(*) FROM ( SELECT g1.GameId FROM Team t1 JOIN Game g1 ON(t1.TeamId = g1.HomeTeamId) WHERE (g1.HomeScore > g1.AwayScore AND g1.HomeTeamId = 1)  UNION  SELECT g2.GameId FR
OM Team t2 JOIN Game g2 ON(t2.TeamId = g2.AwayTeamId) WHERE (g2.AwayScore > g2.HomeScore AND g2.AwayTeamId = 1) ) AS winning_games;
+--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------
---------------+
| EXPLAIN

               |
+--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------
---------------+
| -> Aggregate: count(0)  (cost=26.49..26.49 rows=1) (actual time=0.413..0.414 rows=1 loops=1)
    -> Table scan on winning_games  (cost=21.42..24.09 rows=24) (actual time=0.386..0.402 rows=41 loops=1)
        -> Union materialize with deduplication  (cost=21.30..21.30 rows=24) (actual time=0.384..0.384 rows=41 loops=1)
            -> Filter: (g1.HomeScore > g1.AwayScore)  (cost=9.52 rows=13) (actual time=0.169..0.197 rows=24 loops=1)
                -> Index lookup on g1 using Game_Home_Team_Id (HomeTeamId=1)  (cost=9.52 rows=38) (actual time=0.166..0.183 rows=38 loops=1)
            -> Filter: (g2.AwayScore > g2.HomeScore)  (cost=9.38 rows=11) (actual time=0.090..0.112 rows=17 loops=1)
                -> Index lookup on g2 using AwayTeamId (AwayTeamId=1)  (cost=9.38 rows=34) (actual time=0.089..0.102 rows=34 loops=1)
 |
+--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------
---------------+
```

Indexing Design & Analysis (at least 3 designs):

Advanced Query #1:

Design 1: CREATE INDEX BoxScore_Pts ON BoxScore(Pts);

```
| -> Nested loop inner join  (cost=0.54 rows=0.005) (actual time=0.035..0.037 rows=1 loops=1)
    -> Nested loop inner join  (cost=0.52 rows=0.05) (actual time=0.030..0.033 rows=1 loops=1)
        -> Filter: ((b1.Pts = (select #2)) and (b1.PlayerId = 2544) and (b1.GameId is not null))  (cost=0.51 rows=0.05) (actual time=0.023..0.025 rows=1 loops=1)
            -> Index lookup on b1 using BoxScore_Pts (Pts=(select #2))  (cost=0.51 rows=2) (actual time=0.020..0.022 rows=2 loops=1)
            -> Select #2 (subquery in condition; run only once)
                -> Aggregate: max(b2.Pts)  (cost=111.30 rows=1) (actual time=1.114..1.115 rows=1 loops=1)
                    -> Nested loop inner join  (cost=110.25 rows=11) (actual time=0.494..1.096 rows=60 loops=1)
                        -> Nested loop inner join  (cost=73.50 rows=105) (actual time=0.243..0.602 rows=105 loops=1)
                            -> Filter: (b2.GameId is not null)  (cost=36.75 rows=105) (actual time=0.223..0.297 rows=105 loops=1)
                                -> Index lookup on b2 using PlayerId (PlayerId=2544)  (cost=36.75 rows=105) (actual time=0.213..0.267 rows=105 loops=1)
                            -> Filter: (g2.HomeTeamId is not null)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=105)
                                -> Single-row index lookup on g2 using PRIMARY (GameId=b2.GameId)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=105)
                        -> Filter: (t2.`Year` = 2021)  (cost=0.25 rows=0.1) (actual time=0.004..0.004 rows=1 loops=105)
                            -> Single-row index lookup on t2 using PRIMARY (TeamId=g2.HomeTeamId)  (cost=0.25 rows=1) (actual time=0.004..0.004 rows=1 loops=105)
        -> Filter: (g1.HomeTeamId is not null)  (cost=2.25 rows=1) (actual time=0.007..0.007 rows=1 loops=1)
            -> Single-row index lookup on g1 using PRIMARY (GameId=b1.GameId)  (cost=2.25 rows=1) (actual time=0.006..0.006 rows=1 loops=1)
    -> Filter: (t1.`Year` = 2021)  (cost=0.45 rows=0.1) (actual time=0.004..0.004 rows=1 loops=1)
        -> Single-row index lookup on t1 using PRIMARY (TeamId=g1.HomeTeamId)  (cost=0.45 rows=1) (actual time=0.003..0.003 rows=1 loops=1)
 |
```

- Heavy drop in cost in both `Nested loop inner join`(s)
- Increase in Cost in Filter:
- Drop in Cost

Design 2: CREATE INDEX Game_HomeScore ON Game(HomeScore);

```
| -> Nested loop inner join  (cost=34.65 rows=1) (actual time=10.090..10.110 rows=1 loops=1)
    -> Nested loop inner join  (cost=30.98 rows=11) (actual time=10.082..10.101 rows=1 loops=1)
        -> Filter: ((b1.Pts = (select #2)) and (b1.GameId is not null))  (cost=27.30 rows=11) (actual time=10.034..10.053 rows=1 loops=1)
            -> Index lookup on b1 using PlayerId (PlayerId=2544)  (cost=27.30 rows=105) (actual time=7.777..7.823 rows=105 loops=1)
            -> Select #2 (subquery in condition; run only once)
                -> Aggregate: max(b2.Pts)  (cost=111.30 rows=1) (actual time=2.058..2.059 rows=1 loops=1)
                    -> Nested loop inner join  (cost=110.25 rows=11) (actual time=1.203..2.036 rows=60 loops=1)
                        -> Nested loop inner join  (cost=73.50 rows=105) (actual time=0.586..1.643 rows=105 loops=1)
                            -> Filter: (b2.GameId is not null)  (cost=36.75 rows=105) (actual time=0.515..0.727 rows=105 loops=1)
                                -> Index lookup on b2 using PlayerId (PlayerId=2544)  (cost=36.75 rows=105) (actual time=0.513..0.704 rows=105 loops=1)
                            -> Filter: (g2.HomeTeamId is not null)  (cost=0.25 rows=1) (actual time=0.008..0.008 rows=1 loops=105)
                                -> Single-row index lookup on g2 using PRIMARY (GameId=b2.GameId)  (cost=0.25 rows=1) (actual time=0.008..0.008 rows=1 loops=105)
                        -> Filter: (t2.`Year` = 2021)  (cost=0.25 rows=0.1) (actual time=0.003..0.003 rows=1 loops=105)
                            -> Single-row index lookup on t2 using PRIMARY (TeamId=g2.HomeTeamId)  (cost=0.25 rows=1) (actual time=0.003..0.003 rows=1 loops=105)
        -> Filter: (g1.HomeTeamId is not null)  (cost=0.26 rows=1) (actual time=0.023..0.023 rows=1 loops=1)
            -> Single-row index lookup on g1 using PRIMARY (GameId=b1.GameId)  (cost=0.26 rows=1) (actual time=0.022..0.022 rows=1 loops=1)
    -> Filter: (t1.`Year` = 2021)  (cost=0.25 rows=0.1) (actual time=0.006..0.007 rows=1 loops=1)
        -> Single-row index lookup on t1 using PRIMARY (TeamId=g1.HomeTeamId)  (cost=0.25 rows=1) (actual time=0.006..0.006 rows=1 loops=1)
 |
```

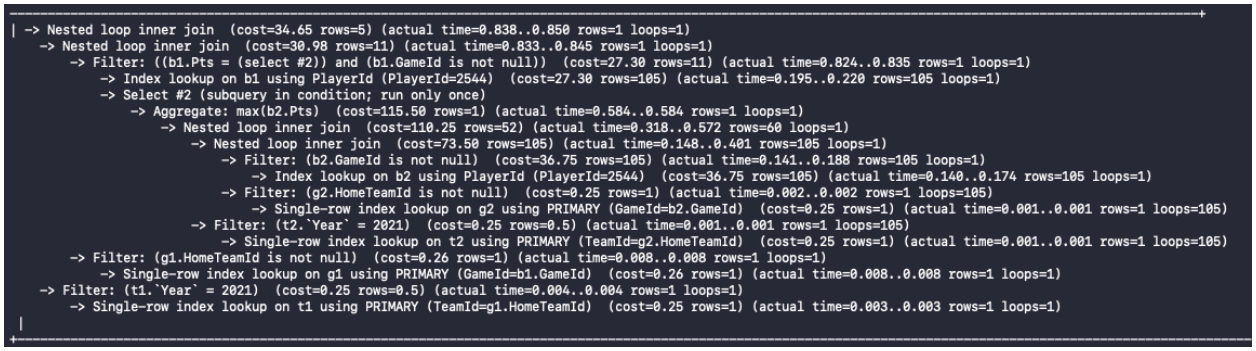
- No change

Design 3: CREATE INDEX Game_AwayScore ON Game(AwayScore);

```
| -> Nested loop inner join  (cost=34.65 rows=1) (actual time=3.738..3.752 rows=1 loops=1)
    -> Nested loop inner join  (cost=30.98 rows=11) (actual time=3.733..3.747 rows=1 loops=1)
        -> Filter: ((b1.Pts = (select #2)) and (b1.GameId is not null))  (cost=27.30 rows=11) (actual time=3.715..3.728 rows=1 loops=1)
            -> Index lookup on b1 using PlayerId (PlayerId=2544)  (cost=27.30 rows=105) (actual time=2.353..2.383 rows=105 loops=1)
            -> Select #2 (subquery in condition; run only once)
                -> Aggregate: max(b2.Pts)  (cost=111.30 rows=1) (actual time=1.287..1.287 rows=1 loops=1)
                    -> Nested loop inner join  (cost=110.25 rows=11) (actual time=0.611..1.271 rows=60 loops=1)
                        -> Nested loop inner join  (cost=73.50 rows=105) (actual time=0.188..0.926 rows=105 loops=1)
                            -> Filter: (b2.GameId is not null)  (cost=36.75 rows=105) (actual time=0.141..0.385 rows=105 loops=1)
                                -> Index lookup on b2 using PlayerId (PlayerId=2544)  (cost=36.75 rows=105) (actual time=0.139..0.368 rows=105 loops=1)
                            -> Filter: (g2.HomeTeamId is not null)  (cost=0.25 rows=1) (actual time=0.005..0.005 rows=1 loops=105)
                                -> Single-row index lookup on g2 using PRIMARY (GameId=b2.GameId)  (cost=0.25 rows=1) (actual time=0.004..0.004 rows=1 loops=105)
                        -> Filter: (t2.`Year` = 2021)  (cost=0.25 rows=0.1) (actual time=0.003..0.003 rows=1 loops=105)
                            -> Single-row index lookup on t2 using PRIMARY (TeamId=g2.HomeTeamId)  (cost=0.25 rows=1) (actual time=0.002..0.003 rows=1 loops=105)
        -> Filter: (g1.HomeTeamId is not null)  (cost=0.26 rows=1) (actual time=0.011..0.011 rows=1 loops=1)
            -> Single-row index lookup on g1 using PRIMARY (GameId=b1.GameId)  (cost=0.26 rows=1) (actual time=0.011..0.011 rows=1 loops=1)
    -> Filter: (t1.`Year` = 2021)  (cost=0.25 rows=0.1) (actual time=0.004..0.004 rows=1 loops=1)
        -> Single-row index lookup on t1 using PRIMARY (TeamId=g1.HomeTeamId)  (cost=0.25 rows=1) (actual time=0.003..0.003 rows=1 loops=1)
 |
```

- No Change

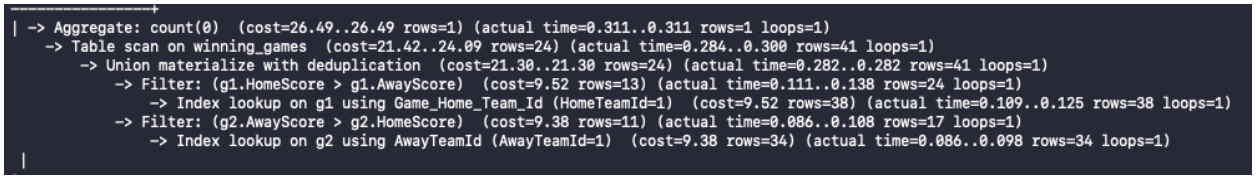
Design 4: CREATE INDEX Team_Year ON Team(Year);



- No change

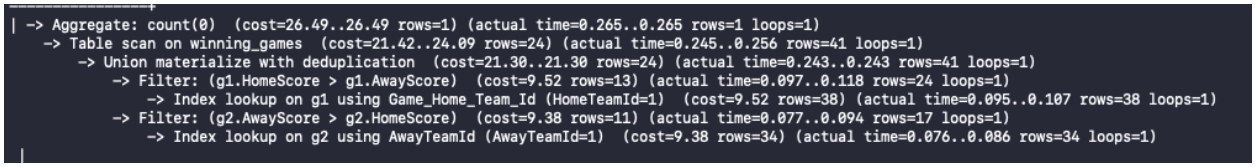
<u>Advanced Query #2</u>:

Design 1: CREATE INDEX Game_HomeScore ON Game(HomeScore);



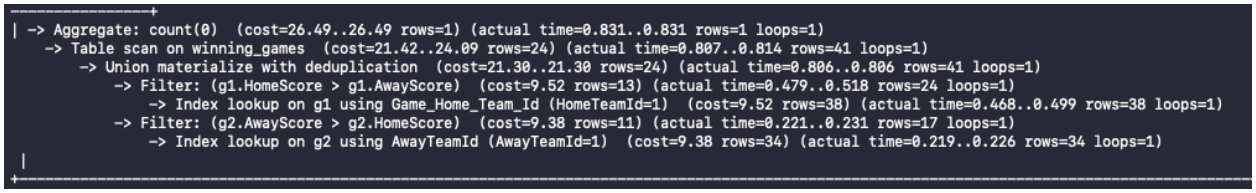
- No change

Design 2: CREATE INDEX Game_AwayScore ON Game(AwayScore);



- No change

Design 3: CREATE INDEX Team_Year ON Team(Year);



- No change

Analysis:

All these indices were made separately from each other, removing those previous before creating another. As shown through the cost and time displays, there was a significant improvement in

Query 1 when an index was created on BoxScore.Pts but otherwise no significant improvement in cost. We noticed slight fluctuations in the times, but the costs stayed the same.

The BoxScore.Pts Index improves costs in the query because we are trying to look up specific points value in BoxScore.Pts (namely, the maximum). Since we already know the number to look for, finding the data location can happen quickly with an index. The improvement is even more substantial since it performs the lookup in a table of over 60k entries.

The indices on Game.HomeScore and Game.AwayScore likely cause no benefit to efficiency since our query simply extracts those values once already found by the filter in the WHERE clause. The index on Team.Year did not prove helpful, which we hypothesize is because there are only 2 years of information in our tables. So, finding a specific year isn't very computationally expensive in the first place.

For the second query, the indices on Game.HomeScore and Game.AwayScore are ineffective because the condition in our query that uses these attributes relies on both (HomeScore > AwayScore and AwayScore > HomeScore). As such, we are not filtering based on properties of one attribute, nor are we trying to find games with a specific score. The index of Team.Year fails for the same reason as explained above for the first advanced query.

We considered creating indexes on primary keys but primary keys are default indexes and that's why there is no improvement. Additionally, we could not make any indices on Foreign Keys that referred to Primary Keys.

Indexing Design Chosen:

We will plan to keep the index on BoxScore.Pts since it definitely improves the cost. All the other indices made no changes.