

Establishing MySQL Connection:

```
Bye
[shouriaddepally@Shouris-MacBook-Pro ~ % mysql -u root -p
[Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 8.0.31 MySQL Community Server - GPL

Copyright (c) 2000, 2022, 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> 
```

Creating Tables Using DDL Commands:

```
CREATE TABLE Teams (
PlayerID INT,
DefenseID INT,
Division VARCHAR(50),
Record VARCHAR(50),
Schedule VARCHAR(50),
Team VARCHAR(10)
);
```

```
CREATE TABLE Defenses (
Team VARCHAR(10),
Sacks INT,
Interceptions INT,
Fumbles_Recovered INT,
Touchdowns INT,
Points_Allowed INT,
Yards_Allowed INT,
Avg_Fantasy_Points INT
);
```

```
CREATE TABLE Games (
GameID INT,
Date VARCHAR(50),
Week INT,
Home_Team VARCHAR(5),
Home_Score INT,
Away_Score INT,
Away_Team VARCHAR(5)
);
```

```

CREATE TABLE Watchlist (
    WatchlistID INT,
    UserID INT,
    Name VARCHAR(100),
    Notes VARCHAR(200)
);

CREATE TABLE Players (
    PlayerID INT,
    Player_Name VARCHAR(50),
    Team VARCHAR(5),
    Position VARCHAR(50),
    Age INT,
    Avg_Fantasy_Points REAL
);

```

Inserting Data Into Tables:

For our database we have 1000 data-entries in the Players, Watchlist, and Games tables. For these tables, we filled in our column values based on the data we gathered on fantasy players and their fantasy outputs using a simple python script. We also followed a similar approach for the Games table which was filled out using previous years' data using home and away team scores and using a similar python script as the Players table. The Watchlist also contains players and notes that correspond to fantasy players which have generated or have potential to generate a high amount of fantasy points from week to week. Below is an image of all the data counts for the three tables which each contain at least 1000 rows.

```

mysql> show tables;
+-----+
| Tables_in_fantasy_football |
+-----+
| Defenses
| Games
| Players
| Teams
| Watchlist
+-----+
5 rows in set (0.01 sec)

mysql> select COUNT(*) FROM Players;
+-----+
| COUNT(*) |
+-----+
| 1070 |
+-----+
1 row in set (0.00 sec)

mysql> select COUNT(*) FROM Watchlist;
+-----+
| COUNT(*) |
+-----+
| 1041 |
+-----+
1 row in set (0.00 sec)

mysql> select COUNT(*) FROM Games;
+-----+
| COUNT(*) |
+-----+
| 1040 |
+-----+
1 row in set (0.01 sec)

```

Advanced SQL Queries:

```
SELECT ROUND(Total_Fantasy_Points / Total_Players, 2) AS Average_Fantasy_Points, P.Team
FROM
(SELECT SUM(Avg_Fantasy_Points) AS Total_Fantasy_Points, Team FROM Players GROUP BY Team)
AS T
JOIN
(SELECT COUNT(PlayerID) AS Total_Players, Team
FROM Players GROUP BY Team) AS P
ON
(P.Team = T.Team)
ORDER BY Average_Fantasy_Points DESC
LIMIT 15;
```

```
mysql> SELECT ROUND(Total_Fantasy_Points / Total_Players, 2) AS Average_Fantasy_Points, P.Team
-> FROM
-> (SELECT SUM(Avg_Fantasy_Points) AS Total_Fantasy_Points, Team FROM Players GROUP BY Team) AS T
-> JOIN
-> (SELECT COUNT(PlayerID) AS Total_Players, Team
-> FROM Players GROUP BY Team) AS P
-> ON
-> (P.Team = T.Team)
-> ORDER BY Average_Fantasy_Points DESC
-> LIMIT 15;
+-----+-----+
| Average_Fantasy_Points | Team |
+-----+-----+
|      7.86 | LAR  |
|      7.67 | BAL  |
|      7.45 | TAM  |
|      7.19 | BUF  |
|      7.16 | PHI  |
|      7.1  | SEA  |
|      7.06 | DEN  |
|      6.97 | ARI  |
|      6.96 | DAL  |
|      6.93 | LAC  |
|      6.82 | JAX  |
|      6.79 | PIT  |
|      6.79 | GNB  |
|      6.75 | LVR  |
|      6.74 | MIN  |
+-----+-----+
15 rows in set (0.02 sec)
```

```
(SELECT p.Player_Name, t.Team, t.Division, p.Avg_Fantasy_Points
FROM Players p JOIN Teams t ON(p.Team = t.Team)
WHERE t.Division Like('NFC West') AND p.Avg_Fantasy_Points > 13
LIMIT 7)
UNION
(SELECT p.Player_Name, t.Team, t.Division, p.Avg_Fantasy_Points
FROM Players p JOIN Teams t ON(p.Team = t.Team)
WHERE t.Division Like ('AFC West') and p.Avg_Fantasy_Points > 13
LIMIT 8);
```

```
mysql> (SELECT p.Player_Name, t.Team, t.Division, p.Avg_Fantasy_Points
->   FROM Players p JOIN Teams t ON(p.Team = t.Team)
-> WHERE t.Division Like('NFC West') AND p.Avg_Fantasy_Points > 13
-> LIMIT 7)
-> UNION
-> (SELECT p.Player_Name, t.Team, t.Division, p.Avg_Fantasy_Points
->   FROM Players p JOIN Teams t ON(p.Team = t.Team)
-> WHERE t.Division Like ('AFC West') and p.Avg_Fantasy_Points > 13
-> LIMIT 8);
+-----+-----+-----+-----+
| Player_Name | Team | Division | Avg_Fantasy_Points |
+-----+-----+-----+-----+
| Cooper Kupp | LAR | NFC West | 17.35 |
| Deebo Samuel | SFO | NFC West | 16.38 |
| James Conner | ARI | NFC West | 14.73 |
| Matthew Stafford | LAR | NFC West | 19.41 |
| Kyler Murray | ARI | NFC West | 21.43 |
| Elijah Mitchell | SFO | NFC West | 13.27 |
| Russell Wilson | SEA | NFC West | 17.36 |
| Austin Ekeler | LAC | AFC West | 17.12 |
| Justin Herbert | LAC | AFC West | 22.41 |
| Patrick Mahomes | KAN | AFC West | 21.29 |
| Derek Carr | LVR | AFC West | 15.12 |
| Teddy Bridgewater | DEN | AFC West | 14.5 |
| Marcus Kemp | KAN | AFC West | 16 |
| Raheem Gallman | LAC | AFC West | 13.41 |
| Ben Dalton | DEN | AFC West | 13.67 |
+-----+-----+-----+-----+
15 rows in set (0.01 sec)
```

Indexing Analysis:

Query 1:

```
SELECT ROUND(Total_Fantasy_Points / Total_Players, 2) AS Average_Fantasy_Points, P.Team
FROM
(SELECT SUM(Avg_Fantasy_Points) AS Total_Fantasy_Points, Team FROM Players GROUP BY Team)
AS T
JOIN
(SELECT COUNT(PlayerID) AS Total_Players, Team
FROM Players GROUP BY Team) AS P
ON
(P.Team = T.Team)
ORDER BY Average_Fantasy_Points DESC
LIMIT 15;
```

Indexing Design #0 - Default Index:

The screenshot shows a terminal window titled "shouria@deppaly ~ mysql -u root -p" with a width of 237x52. The window displays the results of a query and its EXPLAIN output.

Query Results:

		Team	
1837	184	James Proche	N/A
1838	184	Russell Wilson	N/A
1839	184	DAL	N/A

1841 rows in set (0.00 sec)

EXPLAIN Output:

```
+-----+
| EXPLAIN
+-----+
|-----+
|  Sort: Average.Fantasy.Points (actual time=26.946..24.948 rows=34 loops=1)
|    Stream results (cost=2797.88 rows=8) (actual time=22.368..22.422 rows=34 loops=1)
|      -> Nested loop inner join (cost=2797.88 rows=8) (actual time=22.313..22.362 rows=8 loops=1)
|        -> Full outer scan on <temporary> (cost=2797.88 rows=1070 loops=1)
|          -> Table scan on T (cost=2.58..2.59 rows=8) (actual time=17.998..18.004 rows=34 loops=1)
|            -> Materialize (cost=0.00..0.00 rows=0) (actual time=17.997..17.997 rows=34 loops=1)
|              -> Table scan on <temporary> (actual time=17.997..17.997 rows=34 loops=1)
|                -> Aggregate using temporary table (actual time=15.991..15.991 rows=34 loops=1)
|                  -> Table scan on Players (cost=108.50 rows=1070) (actual time=7.929..11.973 rows=1070 loops=1)
|                    -> Index lookup on P using <auto_keyb> (Team=Team) (actual time=0.126..0.127 rows=1 loops=34)
|                      -> Materialize (cost=0.00..0.00 rows=0) (actual time=0.126..0.127 rows=1 loops=1)
|                        -> Table scan on <temporary> (actual time=0.577..0.582 rows=34 loops=1)
|                          -> Aggregate using temporary table (actual time=3.576..3.576 rows=34 loops=1)
|                            -> Table scan on Players (cost=108.50 rows=1070) (actual time=0.552..0.598 rows=1070 loops=1)
|-----+
|-----+
1 row in set (0.11 sec)
```

mysql> |

Indexing Design #1 - Index on the players table:

```
CREATE INDEX index1 ON Players(PlayerID, Player_Name, Team, Position, Age, Avg_Fantasy_Points);
```

```
shouria@deppaly: ~ mysql -u root -p - 237x52
Q: tTeam
5 rows in set (0.01 sec)

mysql> CREATE INDEX index1 ON Players(PlayerID, Player_Name, Team, Position, Age, Avg_Fantasy_Points);
Query OK, 0 rows affected (0.33 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> explain analyze SELECT ROUND(Total_Fantasy_Points / Total_Players, 3) AS Average_Fantasy_Points, P.Team FROM (SELECT SUM(Avg_Fantasy_Points) AS Total_Fantasy_Points, Team FROM Players GROUP BY Team) AS T JOIN (SELECT COUNT(PlayerID) AS Total_Players, Team FROM Players GROUP BY Team) AS P ON (P.Team = T.Team) ORDER BY Average_Fantasy_Points ASC;
+----+-----+
| EXPLAIN
+----+-----+
|   +-- Sort: Average_Fantasy_Points (actual time=3.763..3.765 rows=34 loops=1)
|   |  -> Stream results (cost=2797.88 rows=0) (actual time=3.646..3.759 rows=34 loops=1)
|   |  |  -> Nested loop inner join (cost=2797.88 rows=0) (actual time=3.763..3.765 rows=34 loops=1)
|   |  |  |  -> Filter: (t.Team is not null) (cost=0.11..22.88 rows=1070) (actual time=2.042..2.054 rows=34 loops=1)
|   |  |  |  |  -> Table scan on T (cost=2.50..2.50 rows=8) (actual time=2.041..2.048 rows=34 loops=1)
|   |  |  |  |  |  -> Filter: (t.Team is not null) (cost=0.11..22.88 rows=1070) (actual time=2.041..2.054 rows=34 loops=1)
|   |  |  |  |  |  |  -> Materialize (cost=0.00..0.00 rows=0) (actual time=2.048..2.048 rows=34 loops=1)
|   |  |  |  |  |  |  |  -> Table scan on T (cost=0.00..0.00 rows=0) (actual time=2.048..2.048 rows=34 loops=1)
|   |  |  |  |  |  |  |  |  -> Aggregate using temporary table (actual time=1.982..1.982 rows=34 loops=1)
|   |  |  |  |  |  |  |  |  |  -> Covering index scan on Players using index (cost=108.50 rows=1070) (actual time=0.581..1.246 rows=1070 loops=1)
|   |  |  |  |  |  |  |  |  |  |  -> Index lookup on P using <auto_key> (Team,Team) (actual time=0.948..0.948 rows=1 loops=1)
|   |  |  |  |  |  |  |  |  |  |  |  -> Table scan on <temporary> (cost=1.531..1.538 rows=34 loops=1)
|   |  |  |  |  |  |  |  |  |  |  |  |  -> Aggregate using temporary table (actual time=1.528..1.528 rows=34 loops=1)
|   |  |  |  |  |  |  |  |  |  |  |  |  |  -> Covering index scan on Players using index1 (cost=108.50 rows=1070) (actual time=0.020..0.724 rows=1070 loops=1)
|   |
+----+-----+
1 row in set (0.01 sec)

mysql>
```

Indexing Design #2 - Index on Player_Name column in Players Table:

```
CREATE INDEX index1 ON Players(Player_Name);
```

```
shouria@deppaly: ~ mysql -u root -p - 237x52
Q: tTeam
Records: 0 Duplicates: 0 Warnings: 0

mysql> CREATE INDEX index1 ON Players(Player_Name);
Query OK, 0 rows affected (0.03 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> explain analyze SELECT ROUND(Total_Fantasy_Points / Total_Players, 3) AS Average_Fantasy_Points, P.Team FROM (SELECT SUM(Avg_Fantasy_Points) AS Total_Fantasy_Points, Team FROM Players GROUP BY Team) AS T JOIN (SELECT COUNT(PlayerID) AS Total_Players, Team FROM Players GROUP BY Team) AS P ON (P.Team = T.Team) ORDER BY Average_Fantasy_Points ASC;
+----+-----+
| EXPLAIN
+----+-----+
|   +-- Sort: Average_Fantasy_Points (actual time=3.033..3.036 rows=34 loops=1)
|   |  -> Stream results (cost=2797.88 rows=0) (actual time=2.932..3.014 rows=34 loops=1)
|   |  |  -> Nested loop inner join (cost=2797.88 rows=0) (actual time=2.929..2.997 rows=34 loops=1)
|   |  |  |  -> Filter: (t.Team is not null) (cost=0.00..2797.88 rows=0) (actual time=2.929..2.997 rows=34 loops=1)
|   |  |  |  |  -> Table scan on T (cost=2.50..2.50 rows=8) (actual time=2.929..2.931 rows=34 loops=1)
|   |  |  |  |  |  -> Filter: (t.Team is not null) (cost=0.00..2797.88 rows=0) (actual time=2.929..2.931 rows=34 loops=1)
|   |  |  |  |  |  |  -> Materialize (cost=0.00..0.00 rows=0) (actual time=2.931..2.931 rows=34 loops=1)
|   |  |  |  |  |  |  |  -> Table scan on <temporary> (actual time=1.446..1.454 rows=34 loops=1)
|   |  |  |  |  |  |  |  |  -> Table scan on <temporary> (actual time=1.446..1.454 rows=34 loops=1)
|   |  |  |  |  |  |  |  |  |  -> Aggregate using temporary table (actual time=1.446..1.454 rows=34 loops=1)
|   |  |  |  |  |  |  |  |  |  |  -> Table scan on Players (cost=108.50 rows=1070) (actual time=0.044..0.828..0.721 rows=1070 loops=1)
|   |  |  |  |  |  |  |  |  |  |  |  -> Index lookup on P using <auto_key> (Team,Team) (actual time=0.044..0.044 rows=1 loops=34)
|   |  |  |  |  |  |  |  |  |  |  |  |  -> Materialize (cost=0.00..0.00 rows=0) (actual time=1.452..1.452 rows=34 loops=1)
|   |  |  |  |  |  |  |  |  |  |  |  |  |  -> Table scan on <temporary> (actual time=1.452..1.452 rows=34 loops=1)
|   |  |  |  |  |  |  |  |  |  |  |  |  |  |  -> Aggregate using temporary table (actual time=1.419..1.419 rows=34 loops=1)
|   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  -> Table scan on Players (cost=108.50 rows=1070) (actual time=0.007..0.728 rows=1070 loops=1)
|   |
+----+-----+
1 row in set (0.00 sec)

mysql>
```

Indexing Design #3 - Unique Index on PlayerID in Players Table:

```
CREATE UNIQUE INDEX index1 ON Players(PlayerID);
```

Final Indexing Analysis Report - Query 1:

For our first query, we selected our third indexing design as the best index design. In this indexing design, we create an index on the PlayerID on the Players table compared to previous designs where we indexed on the entire Players table, as well as the Player_Name column in the Players table. The reason why we chose this indexing design as our best design was because its start up time at the first leaf level is faster than either of the indexing designs mentioned previously - measuring a range of with a minimal range of 0.0017 - 0.627 time units. In addition, this third design is also faster than the previous designs when performing an index lookup on the second subquery statement, taking roughly 0.003-0.620 time units as its start up time at the second leaf level.

Query 2:

```
(SELECT p.Player_Name, t.Team, t.Division, p.Avg_Fantasy_Points
 FROM Players p JOIN Teams t ON(p.Team = t.Team)
 WHERE t.Division Like('NFC West') AND p.Avg_Fantasy_Points > 13
 LIMIT 7)
UNION
(SELECT p.Player_Name, t.Team, t.Division, p.Avg_Fantasy_Points
 FROM Players p JOIN Teams t ON(p.Team = t.Team)
 WHERE t.Division Like ('AFC West') and p.Avg_Fantasy_Points > 13
 LIMIT 8);
```

Indexing Design #0 - Default Index:

```
mysql> explain analyze (SELECT p.Player_Name, t.Team, t.Division, p.Avg_Fantasy_Points
   FROM Players p JOIN Teams t ON(p.Team = t.Team) WHERE t.Division Like('NFC West') AND p.Avg_Fantasy_Points > 13 LIMIT 7) UNION (SELECT p.Player_Name, t.Team, t.Division, p.Avg_Fantasy_Points
   FROM Players p JOIN Teams t ON(p.Team = t.Team) WHERE t.Division Like ('AFC West') and p.Avg_Fantasy_Points > 13 LIMIT 8);
+-----+
| EXPLAIN
+-----+
|> Table scan on <union temporary> (cost=579.56..191.86 rows=3) (actual time=2.086..2.648 rows=15 loops=1)
  -> Union materialize with deduplication (cost=179.26..179.26 rows=8) (actual time=2.494..2.664 rows=15 loops=1)
    -> Limit: 7 row(s) (cost=89.21 rows=4) (actual time=1.413..1.473 rows=7 loops=1)
      -> Inner hash join (p.Team = t.Team) (cost=89.21 rows=4) (actual time=1.412..1.471 rows=7 loops=1)
        -> Filter: (p.Avg_Fantasy_Points > 13) (cost=21.56 rows=30) (actual time=0.023..0.678 rows=31 loops=1)
          -> Table scan on p (cost=21.56 rows=1978) (actual time=0.021..0.466 rows=79 loops=1)
        -> Hash
          -> Filter: (t.Division like 'NFC West') (cost=3.45 rows=4) (actual time=0.661..0.678 rows=4 loops=1)
            -> Table scan on t (cost=3.45 rows=2) (actual time=0.007..0.011 rows=2 loops=1)
      -> Hash
        -> Filter: (t.Division like ('AFC West')) (cost=89.21 rows=4) (actual time=0.077..0.756 rows=4 loops=1)
          -> Inner hash join (p.Team = t.Team) (cost=89.21 rows=4) (actual time=0.076..0.756 rows=4 loops=1)
            -> Filter: (p.Avg_Fantasy_Points > 13) (cost=21.56 rows=30) (actual time=0.008..0.665 rows=8 loops=1)
              -> Table scan on p (cost=21.56 rows=1978) (actual time=0.007..0.574 rows=968 loops=1)
            -> Hash
              -> Filter: (t.Division like 'AFC West') (cost=3.45 rows=4) (actual time=0.023..0.044 rows=4 loops=1)
                -> Table scan on t (cost=3.45 rows=32) (actual time=0.013..0.032 rows=32 loops=1)
  -> Hash
    -> Filter: (t.Division like ('AFC West')) (cost=3.45 rows=4) (actual time=0.023..0.032 rows=4 loops=1)
      -> Table scan on t (cost=3.45 rows=32) (actual time=0.013..0.032 rows=32 loops=1)
+-----+
1 row in set (0.01 sec)
```

Indexing Design #1 - Index on the players table:

```
mysql> Q: tTeam
+-----+
| EXPLAIN
+-----+
|> Table scan on <union temporary> (cost=72.40..774.91 rows=15) (actual time=1.316..1.338 rows=15 loops=1)
  -> Union materialize with deduplication (cost=772.22..772.22 rows=15) (actual time=1.315..1.315 rows=15 loops=1)
    -> Limit: 7 row(s) (cost=385.36 rows=7) (actual time=1.185..1.208 rows=7 loops=1)
      -> Inner hash join (p.Team = t.Team) (cost=385.36 rows=38) (actual time=1.184..1.207 rows=7 loops=1)
        -> Covering index scan on p using index1 (cost=0.45 rows=1978) (actual time=0.010..0.028 rows=32 loops=1)
        -> Hash
          -> Filter: (t.Division like 'NFC West') (cost=3.45 rows=4) (actual time=0.114..0.128 rows=4 loops=1)
            -> Table scan on t (cost=3.45 rows=32) (actual time=0.008..0.021 rows=32 loops=1)
      -> Hash
        -> Filter: (t.Division like ('AFC West')) (cost=3.45 rows=4) (actual time=0.048..0.098 rows=4 loops=1)
          -> Inner hash join (p.Team = t.Team) (cost=385.36 rows=38) (actual time=0.048..0.098 rows=43 loops=1)
            -> Covering index scan on p using index1 (cost=0.43 rows=1978) (actual time=0.005..0.032 rows=43 loops=1)
            -> Hash
              -> Filter: (t.Division like 'AFC West') (cost=3.45 rows=4) (actual time=0.013..0.024 rows=4 loops=1)
                -> Table scan on t (cost=3.45 rows=32) (actual time=0.008..0.022 rows=32 loops=1)
  -> Hash
    -> Filter: (t.Division like ('AFC West')) (cost=3.45 rows=4) (actual time=0.023..0.032 rows=4 loops=1)
      -> Table scan on t (cost=3.45 rows=32) (actual time=0.013..0.032 rows=32 loops=1)
+-----+
1 row in set (0.01 sec)

mysql>
```

```
CREATE INDEX index1 ON Players(PlayerID, Player_Name, Team, Position, Age, Avg_Fantasy_Points);
```

Indexing Design #2 - Index on Player_Name column in Players Table:

```
CREATE INDEX index1 ON Players(Player_Name);
```

```
mysql> DROP INDEX index1 ON Players;
Query OK, 0 rows affected (0.05 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> CREATE INDEX index1 ON Players(Player_Name);
Query OK, 0 rows affected (0.03 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> explain analyze (SELECT p.Player_Name, t.Team, t.Division FROM Players p JOIN Teams t ON(p.Team = t.Team) WHERE t.Division Like('NFC West') LIMIT 7) UNION (SELECT p.Player_Name, t.Team, t.Division FROM Players p JOIN Teams t ON(p.Team = t.Team) WHERE t.Division Like('AFC West') LIMIT 8);
+-----+
| EXPLAIN
+-----+
|   |
+-----+
| -> Table scan on union temporary> (cost=>72.40..774.91 rows=15) (actual time=2.926..2.928 rows=15 loops=1)
    -> Union materialize with deduplication (cost=772.22..772.22 rows=15) (actual time=2.924..2.924 rows=15 loops=1)
        -> Limit: 7 rows (cost=385.36 rows=7) (actual time=1.413..1.469 rows=7 loops=1)
            -> Inner hash join (p.Team = t.Team) (cost=385.36 rows=388) (actual time=1.413..1.467 rows=7 loops=1)
                -> Table scan on p (cost=3.43 rows=1878) (actual time=0.122..0.166 rows=37 loops=1)
                    -> Hash
                        -> Filter: (t.Division like 'NFC West') (cost=3.45 rows=4) (actual time=0.183 rows=4 loops=1)
                            -> Table scan on t (cost=3.45 rows=32) (actual time=0.084..0.084 rows=32 loops=1)
                -> Inner hash join (p.Team = t.Team) (cost=385.36 rows=388) (actual time=1.379..1.489 rows=8 loops=1)
                    -> Table scan on p (cost=3.43 rows=1878) (actual time=0.068..0.092 rows=43 loops=1)
                    -> Hash
                        -> Filter: (t.Division like 'AFC West') (cost=3.45 rows=4) (actual time=0.204..0.321 rows=4 loops=1)
                            -> Table scan on t (cost=3.45 rows=32) (actual time=0.288..0.310 rows=32 loops=1)
            -> Limit: 8 rows (cost=385.36 rows=8) (actual time=1.413..1.469 rows=8 loops=1)
                -> Inner hash join (p.Team = t.Team) (cost=385.36 rows=388) (actual time=1.379..1.489 rows=8 loops=1)
                    -> Table scan on p (cost=3.43 rows=1878) (actual time=0.068..0.092 rows=43 loops=1)
                    -> Hash
                        -> Filter: (t.Division like 'AFC West') (cost=3.45 rows=4) (actual time=0.204..0.321 rows=4 loops=1)
                            -> Table scan on t (cost=3.45 rows=32) (actual time=0.288..0.310 rows=32 loops=1)
+-----+
1 row in set (0.03 sec)

mysql> |
```

Indexing Design #3 - Unique Index on PlayerID in Players Table:

```
CREATE UNIQUE INDEX index1 ON Players(PlayerID);
```

```
mysql> DROP INDEX index1 ON Players;
Query OK, 0 rows affected (0.03 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> CREATE UNIQUE INDEX index1 ON Players(PlayerID);
Query OK, 0 rows affected (0.03 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> explain analyze (SELECT p.Player_Name, t.Team, t.Division FROM Players p JOIN Teams t ON(p.Team = t.Team) WHERE t.Division Like('NFC West') LIMIT 7) UNION (SELECT p.Player_Name, t.Team, t.Division FROM Players p JOIN Teams t ON(p.Team = t.Team) WHERE t.Division Like('AFC West') LIMIT 8);
+-----+
| EXPLAIN
+-----+
|   |
+-----+
| -> Table scan on union temporary> (cost=>72.40..774.91 rows=15) (actual time=0.262..0.265 rows=15 loops=1)
    -> Union materialize with deduplication (cost=772.22..772.22 rows=15) (actual time=0.261..0.261 rows=15 loops=1)
        -> Limit: 7 rows (cost=385.36 rows=7) (actual time=0.127..0.168 rows=7 loops=1)
            -> Inner hash join (p.Team = t.Team) (cost=385.36 rows=388) (actual time=0.126..0.158 rows=7 loops=1)
                -> Table scan on p (cost=3.43 rows=1878) (actual time=0.056..0.068 rows=37 loops=1)
                -> Hash
                    -> Filter: (t.Division like 'NFC West') (cost=3.45 rows=4) (actual time=0.087..0.0851 rows=4 loops=1)
                        -> Table scan on t (cost=3.45 rows=32) (actual time=0.018..0.043 rows=32 loops=1)
        -> Limit: 8 rows (cost=385.36 rows=8) (actual time=0.126..0.168 rows=8 loops=1)
            -> Inner hash join (p.Team = t.Team) (cost=385.36 rows=388) (actual time=0.126..0.168 rows=8 loops=1)
                -> Table scan on p (cost=3.43 rows=1878) (actual time=0.064..0.093 rows=43 loops=1)
                -> Hash
                    -> Filter: (t.Division like 'AFC West') (cost=3.45 rows=4) (actual time=0.011..0.031 rows=4 loops=1)
                        -> Table scan on t (cost=3.45 rows=32) (actual time=0.003..0.025 rows=32 loops=1)
            -> Limit: 8 rows (cost=385.36 rows=8) (actual time=0.126..0.168 rows=8 loops=1)
                -> Inner hash join (p.Team = t.Team) (cost=385.36 rows=388) (actual time=0.126..0.168 rows=8 loops=1)
                    -> Table scan on p (cost=3.43 rows=1878) (actual time=0.064..0.093 rows=43 loops=1)
                    -> Hash
                        -> Filter: (t.Division like 'AFC West') (cost=3.45 rows=4) (actual time=0.011..0.031 rows=4 loops=1)
                            -> Table scan on t (cost=3.45 rows=32) (actual time=0.003..0.025 rows=32 loops=1)
+-----+
1 row in set (0.00 sec)

mysql> |
```

Final Indexing Analysis Report - Query 2:

For our second query, we selected our third indexing design as the best index design. In this indexing design, we create an index on the PlayerID on the Players table compared to previous designs where we indexed on the entire Players table, as well as the Player_Name column in the Players table. In our query, we use the UNION set operation to join the players who are on teams that belong either in the AFC West or in the NFC West and had an average fantasy points score greater than 13 points per week. The first portion of our UNION operation selects attributes from the Players table as well as the Teams table, which are then joined based on

equivalent Team values across the Players and Teams tables. The filtering table scan on this first leaf level is faster than any of the previous approaches, measuring with the minimal time range of 0.018-0.043 time units. In addition, the second table scan also measures the fastest second leaf level startup time range at 0.003 - 0.025 time units. These two minimal measurements for leaf level scans is why we chose this Indexing Design #3 as the best indexing approach for this query.