Pocket Fantasy Football Analytics

Stage 3 - Database Implementation and Analytics

Team # 032 - MangoDB

Adish Patil, Sid Karnam, Rohit Chalamala, Neehar Sawant

TA: Lu, Yicheng

1. Database Table DDL Commands

Creating Tables -

The structure of these tables comes from our Database Design document, where we created an ER diagram and Relational Schema.

```
/* NFL Player Information Table */
CREATE TABLE NFL_Players(
   player_id   VARCHAR(255) NOT NULL,
   player_name VARCHAR(255) NOT NULL,
   team        VARCHAR(5) NOT NULL,
   position    VARCHAR(5) NOT NULL
   PRIMARY KEY(player_id),
   FOREIGN KEY(team) REFERENCES NFL_Teams(team) ON DELETE CASCADE
);

/* NFL Wide Reciever Table */
CREATE TABLE NFL_Players_WR(
   player_id     VARCHAR(255) NOT NULL,
   player_name   VARCHAR(255) NOT NULL,
   team          VARCHAR(5) NOT NULL,
   rushing_yds   INTEGER  NOT NULL,
   receptions    INTEGER  NOT NULL,
   receiving_yds INTEGER  NOT NULL,
   fumbles       INTEGER  NOT NULL,
   touchdowns    INTEGER  NOT NULL,
   FOREIGN KEY(player_id) REFERENCES NFL_Players(player_id) ON DELETE CASCADE
);

/* NFL Running Back Table */
CREATE TABLE NFL_Players_RB(
   player_id     VARCHAR(255) NOT NULL,
   player_name   VARCHAR(255) NOT NULL,
   team          VARCHAR(5) NOT NULL,
   rushing_yds   INTEGER  NOT NULL,
   receptions    INTEGER  NOT NULL,
   receiving_yds INTEGER  NOT NULL,
   fumbles       INTEGER  NOT NULL,
   touchdowns    INTEGER  NOT NULL,
   FOREIGN KEY(player_id) REFERENCES NFL_Players(player_id) ON DELETE CASCADE
```

```
);

/* NFL Quarterback Table */
CREATE TABLE NFL_Players_QB(
   player_id       VARCHAR(255) NOT NULL,
   player_name     VARCHAR(255) NOT NULL,
   team            VARCHAR(5) NOT NULL,
   passing_yds     INTEGER  NOT NULL,
   rushing_yds     INTEGER  NOT NULL,
   turnovers       INTEGER  NOT NULL,
   pass_touchdowns INTEGER  NOT NULL,
   rush_touchdowns INTEGER  NOT NULL,
   FOREIGN KEY(player_id) REFERENCES NFL_Players(player_id) ON DELETE CASCADE
);

/* NFL Tight-End Table */
CREATE TABLE NFL_Players_TE(
   player_id     VARCHAR(255) NOT NULL,
   player_name   VARCHAR(255) NOT NULL,
   team          VARCHAR(5) NOT NULL,
   receiving_yds INTEGER  NOT NULL,
   receptions    INTEGER  NOT NULL,
   fumbles       INTEGER  NOT NULL,
   touchdowns    INTEGER  NOT NULL,
   FOREIGN KEY(player_id) REFERENCES NFL_Players(player_id) ON DELETE CASCADE
);

/* NFL Kickers Table */
CREATE TABLE NFL_Players_K(
   player_id      VARCHAR(255) NOT NULL,
   player_name    VARCHAR(255) NOT NULL,
   team           VARCHAR(5) NOT NULL,
   extrapt_made   INTEGER  NOT NULL,
   extrapt_missed INTEGER  NOT NULL,
   fg_made        INTEGER  NOT NULL,
   fg_missed      INTEGER  NOT NULL,
   FOREIGN KEY(player_id) REFERENCES NFL_Players(player_id) ON DELETE CASCADE
);

/* NFL Defense & Special Teams Table */
CREATE TABLE NFL_Players_D_ST(
   team_id       VARCHAR(255) NOT NULL,
```

```
   team          VARCHAR(5) NOT NULL,
   interceptions INTEGER  NOT NULL,
   sacks         NUMERIC(4,2) NOT NULL,
   fumbles       INTEGER  NOT NULL,
   safety        INTEGER  NOT NULL,
   touchdowns    INTEGER  NOT NULL,
   PRIMARY KEY(team_id),
   FOREIGN KEY(team) REFERENCES NFL_Teams(team) ON DELETE CASCADE
);

/* NFL Team Information Table */
CREATE TABLE NFL_Teams(
   team_id              VARCHAR(255) NOT NULL,
   team                 VARCHAR(5) NOT NULL,
   wins                 INTEGER  NOT NULL,
   losses               INTEGER  NOT NULL,
   ties                 BIT  NOT NULL,
   points_for           INTEGER  NOT NULL,
   points_against       INTEGER  NOT NULL,
   strength_of_schedule NUMERIC(5,3) NOT NULL,
   strength_of_victory  NUMERIC(5,3) NOT NULL,
   PRIMARY KEY(team)
);

/* NFL Injuries */
CREATE TABLE Injuries(
   player_id    VARCHAR(255) NOT NULL,
   player       VARCHAR(255) NOT NULL,
   team         VARCHAR(5) NOT NULL,
   position     VARCHAR(5) NOT NULL,
   injury       VARCHAR(255),
   week_injured INTEGER  NOT NULL,
   FOREIGN KEY(player_id) REFERENCES NFL_Players(player_id) ON DELETE CASCADE
);

/* NFL Prospects */
CREATE TABLE NFL_Prospects(
   player_id    VARCHAR(255) NOT NULL PRIMARY KEY,
   name         VARCHAR(255) NOT NULL,
   position     VARCHAR(3) NOT NULL,
   team         VARCHAR(5),
   conference   VARCHAR(45),
```

```
   top_prospect VARCHAR(3) NOT NULL,
   year         INTEGER  NOT NULL
);
```

Inserting Data -

As described in the Project Proposal, we used the Sportradar NFL v7 API to get all the relevant data needed for our Database Schema. We used a python script to call the API and collected all the data for each table in CSV files. Below is the link to a folder on our GitHub repository with all the CSV files…
https://github.com/cs411-alawini/fa22-cs411-A-team032-MangoDB/tree/main/csv_data

Then, using an online CSV to SQL tool, we could insert all the data into the tables we had created. You can view the entire schema and insert tables in a singular SQL file that can be viewable here…

https://github.com/cs411-alawini/fa22-cs411-A-team032-MangoDB/blob/main/Pocket_Fantasy_Football_Analytics.sql

Lastly, we started an SQL server and connected to it using Azure Data Studio to run our queries and commands properly.

2. Advanced SQL Queries

Advanced Query 1: Finding the best FLEX offensive players (WR, RB, TE) in the NFL based on the most important stats for each position (touchdowns, recieving_yds, rushing_yds).

SQL Code -

```
SELECT DISTINCT player_id, player_name, team, touchdowns, receiving_yds as yards
FROM master.dbo.NFL_Players_WR
WHERE    (touchdowns >= (SELECT MAX(touchdowns)
                     FROM master.dbo.NFL_Players_WR
                    ))
       OR
       (receiving_yds >= (SELECT MAX(receiving_yds)
                     FROM master.dbo.NFL_Players_WR
                    ))
```

```
UNION

SELECT DISTINCT player_id, player_name, team, touchdowns, receiving_yds as yards
FROM master.dbo.NFL_Players_TE
WHERE    (touchdowns >= (SELECT MAX(touchdowns)
                    FROM master.dbo.NFL_Players_TE
                   ) )
       OR
       (receiving_yds >= (SELECT MAX(receiving_yds)
                    FROM master.dbo.NFL_Players_TE
                   ))

UNION

SELECT DISTINCT player_id, player_name, team, touchdowns, rushing_yds as yards
FROM master.dbo.NFL_Players_RB
WHERE    (touchdowns >= (SELECT MAX(touchdowns)
                    FROM master.dbo.NFL_Players_RB
                   ))
       OR
       (rushing_yds >= (SELECT MAX(rushing_yds)
                    FROM master.dbo.NFL_Players_RB
                   ))
```

Result -

	player_id	player_name	team	touchdowns	yards
1	01d8aee3-e1c4-4988-970a-8…	Tyreek Hill	MIA	2	701
2	0618f387-9b72-4270-8b8f-d…	Mark Andrews	BAL	5	455
3	4bd60b33-9fbf-4156-ba2b-8…	Nick Chubb	CLE	7	649
4	a1c40664-b265-4083-aad2-5…	Stefon Diggs	BUF	6	656
5	c3859e06-5f23-4302-a71b-0…	Travis Kelce	KC	7	455

Query 2: Find the NFL WRs on the Buffalo Bills team that have receiving yards greater than or equal to the average WR receiving yards across the NFL.

SQL Code -

```
SELECT DISTINCT b.player_id, b.player_name, b.team, receiving_yds, touchdowns
FROM NFL_Players a JOIN NFL_Players_WR b ON (a.player_id = b.player_id)
WHERE b.team = 'BUF' AND b.receiving_yds >= (SELECT AVG(receiving_yds)
```

```
                                    FROM NFL_Players_WR
                                    WHERE receiving_yds > 0)
```

Result -

	player_id	player_name	team	receiving_yds	touchdowns
1	a1c40664-b265-4083-aad2-5…	Stefon Diggs	BUF	656	6
2	dc397432-7157-4ce4-976d-b…	Gabe Davis	BUF	383	4

3. Indexing

**Instead of using EXPLAIN ANALYZE, since we made our database locally, we added "set statistics time on" before the query. Finally, after the query, we added: "set statistics time off." This is because we are using Azure Data Studio at the moment.*

Advanced Query 1:

(a) Query Time Performance Without Indexing -

```
8:24:40 PM      Started executing query at Line 1
                (5 rows affected)

                SQL Server Execution Times:
                  CPU time = 83 ms,  elapsed time = 84 ms.
                Total execution time: 00:00:00.097
```

(b) Different Indexes with Query Time Performance -

- CREATE INDEX player_id_idx ON NFL_Players_WR(player_id(7));

```
8:37:23 PM      Started executing query at Line 1
                (5 rows affected)

                SQL Server Execution Times:
                  CPU time = 39 ms,  elapsed time = 39 ms.
                Total execution time: 00:00:00.043
```

  -

- CREATE INDEX player_id_idx ON NFL_Players_RB(player_id(7));

```
8:39:50 PM      Started executing query at Line 1
                (5 rows affected)

                SQL Server Execution Times:
                  CPU time = 38 ms,  elapsed time = 39 ms.
                Total execution time: 00:00:00.042
```

  -

- CREATE INDEX player_id_idx ON NFL_Players_TE(player_id(7));

```
8:45:29 PM      Started executing query at Line 1
                (5 rows affected)

                SQL Server Execution Times:
                  CPU time = 45 ms,  elapsed time = 45 ms.
                Total execution time: 00:00:00.050
```

  -

(c) Report on the index design you all select and explain why you chose it, referencing the analysis you performed in (b) -

The indexes we tried out were implemented on the player_id attribute. This attribute is a VARCHAR consisting of 36 characters. We used Professor Alawini's approach in the lecture when he indexed the comments to 5 characters. Like that, we thought we could implement it on the player_id for 7 characters. We tried doing that on three tables: NFL_Players_WR, NFL_Players_RB, and NFL_Players_TE. In the end, the index on the NFL_Players_RB gave us the best query performance time.

CREATE INDEX player_id_idx ON NFL_Players_RB(player_id(7));

Advanced Query 2:

(d) Query Time Performance Without Indexing -

8:49:28 PM Started executing query at Line 1
(2 rows affected)

SQL Server Execution Times:
CPU time = 1 ms, elapsed time = 0 ms.
Total execution time: 00:00:00.013

(e) Different Indexes with Query Time Performance -

- CREATE INDEX player_id_idx ON NFL_Players_WR(player_id(7));

9:12:03 PM Started executing query at Line 1
(2 rows affected)

SQL Server Execution Times:
CPU time = 0 ms, elapsed time = 0 ms.
Total execution time: 00:00:00.005

  -

(f) Report on the index design you all select and explain why you chose it, referencing the analysis you performed in (b) -

With this Advanced Query, the performance time originally was quite fast.

The index we tried out was implemented on the player_id attribute. This attribute is a VARCHAR consisting of 36 characters. We used Professor Alawini's approach in the lecture when he indexed the comments to 5 characters. Like that, we thought we could implement it on the player_id for 7 characters. We tried doing that on the NFL_Players_WR, which resulted in a very fast speed. However, there wasn't much difference compared to the original query time without the index.

Overall, when we used indexing, the results were slightly better. I think this can be attributed to the fact that our databases aren't super large; we believe if we had even longer queries with larger databases, the indexing would give us an even greater advantage.