Project Title	**Sports Analytics on NCAAFB Data Using Real-Time API Feeds**
Skills take away From This Project	**Python scripting, Data Collection using API integration, Data Management using SQL, Streamlit**
Domain	**Sports/Data Analytics**

Problem Statement:

The NCAAFB Data Explorer project aims to build a unified platform for storing, analyzing, and visualizing NCAA Football data extracted from the Sportradar NCAAFB API. The system will integrate weekly rankings, team rosters, player profiles, and season statistics into a relational database, enabling users to explore team performances, player progressions, and national rankings through an interactive Streamlit dashboard.

The system provides a structured and queryable foundation for sports data analysis, enabling insights into team performance, ranking trends, player profiles, and seasonal outcomes.

Business Use Cases

- **Team Analytics:** Explore detailed information about teams, including venue, conference, and coach details.
- **Ranking Insights:** Track weekly poll rankings, points, and first-place votes over time.

- **Player Performance:** Analyze player stats (rushing, receiving, etc.) and eligibility over multiple seasons.
- **Season Insights:** View aggregated team performance and player statistics across years.
- **Venue and Conference Comparison:** Compare team venues by capacity or conference success rates.

Approach:

Data Extraction

- Parse and extract data from Sportradar JSON responses.(using API)
- Transform nested JSON structures into a flat relational schema for analysis.

Data Storage:

- Create a SQL database with well-designed schema (e.g., defining appropriate data types and primary keys).   The data to be collected is provided below.

Data Collection:

- **Parse JSON data from NCAAFB API endpoints:**
    - `/rankings/weekly`
    - `/teams/{team_id}/roster`
    - `/players/{player_id}/profile`
    - `/seasons`
    - `/season/{year}/schedule`

Data Transformation

- Flatten nested structures (e.g., teams within rankings, players within rosters).
- Clean and normalize for SQL-based analysis.

Create simplified and clean relational schema for the NCAAFB API data:

---

1. Teams

Stores core team info (shared across all endpoints).

Column	Type	Description
team_id (PK)	UUID	Unique ID from API
market	VARCHAR	City/region (e.g., “Ohio State”)
name	VARCHAR	Team name (e.g., “Buckeyes”)
alias	VARCHAR	Short code (e.g., “OSU”)
founded	INT	Year founded (optional, from roster)
mascot	VARCHAR	Team mascot
fight_song	VARCHAR	Fight song title

championships_won	INT	Total championships
conference_id (FK)	UUID	Linked to Conferences table
division_id (FK)	UUID	Linked to Divisions table
venue_id (FK)	UUID	Linked to Venues table

---

2. Venues

Each team plays in one venue (stadium).

Column	Type	Description
venue_id (PK)	UUID	Unique venue ID
name	VARCHAR	Stadium name
city	VARCHAR	City
state	VARCHAR	State
country	VARCHAR	Country

zip	VARCHAR	ZIP code
address	VARCHAR	Address
capacity	INT	Seating capacity
surface	VARCHAR	Turf type
roof_type	VARCHAR	Indoor/outdoor
latitude	DECIMAL	Latitude
longitude	DECIMAL	Longitude

3. Conferences

From roster and team info.

Column	Type	Description
conference_id (PK)	UUID	Unique ID
name	VARCHAR	Conference name (e.g., Southeastern)

alias	VARCHAR	Short code (e.g., SEC)

4. Seasons

From seasons endpoint.

Column	Type	Description
season_id (PK)	UUID	Unique ID
year	INT	Season year
start_date	DATE	Start date
end_date	DATE	End date
status	VARCHAR	Status (closed/active)
type_code	VARCHAR	Type (REG, POST, etc.)

5. Players

Core player info (from `player profile` and `team roster`). Game roster

Column	Type	Description
player_id (PK)	UUID	Player ID
first_name	VARCHAR	First name
last_name	VARCHAR	Last name
abbr_name	VARCHAR	Abbreviation
birth_place	VARCHAR	Birth place
position	VARCHAR	Position (QB, RB, etc.)
height	INT	Height (inches)
weight	INT	Weight (lbs)
status	VARCHAR	Status (ACT, NWT, etc.)

eligibility	VARCHAR	Class (FR, JR, SR, etc.)
team_id (FK)	UUID	Linked to Teams

📊 6. Player_Statistics

Each player’s stats per season and team (from player profile’s nested structure).

Column	Type	Description
stat_id (PK)	SERIAL	Unique record ID
player_id (FK)	UUID	Linked to Players
team_id (FK)	UUID	Linked to Teams
season_id (FK)	UUID	Linked to Seasons
games_played	INT	Games played
games_started	INT	Games started
rushing_yards	INT	Total rushing yards

rushing_touchdowns	INT	Rushing touchdowns
receiving_yards	INT	Receiving yards
receiving_touchdowns	INT	Receiving touchdowns
kick_return_yards	INT	Kick return yards
fumbles	INT	Fumbles count

(Optional: You could further break out rushing, receiving, penalties, etc. into sub-tables — but this compact form is enough for analytics.)

🏅 7. Rankings

From `rankings_weekly` endpoint.

Column	Type	Description
ranking_id (PK)	SERIAL	Unique ID
poll_id	UUID	e.g., AP25
poll_name	VARCHAR	Poll name

season_id (FK)	**UUID**	**Linked to Seasons**
week	**INT**	**Week number**
effective_time	**TIMESTAMP**	**Poll release time**
team_id (FK)	**UUID**	**Linked to Teams**
rank	**INT**	**Rank position**
prev_rank	**INT**	**Previous rank**
points	**INT**	**Poll points**
fp_votes	**INT**	**First-place votes**
wins	**INT**	**Wins**
losses	**INT**	**Losses**
ties	**INT**	**Ties**

8. Coaches

Column	Type	Description
coach_id (PK)	UUID	Coach ID
full_name	VARCHAR	Full name
position	VARCHAR	Role (Head Coach, etc.)
team_id (FK)	UUID	Linked to Teams

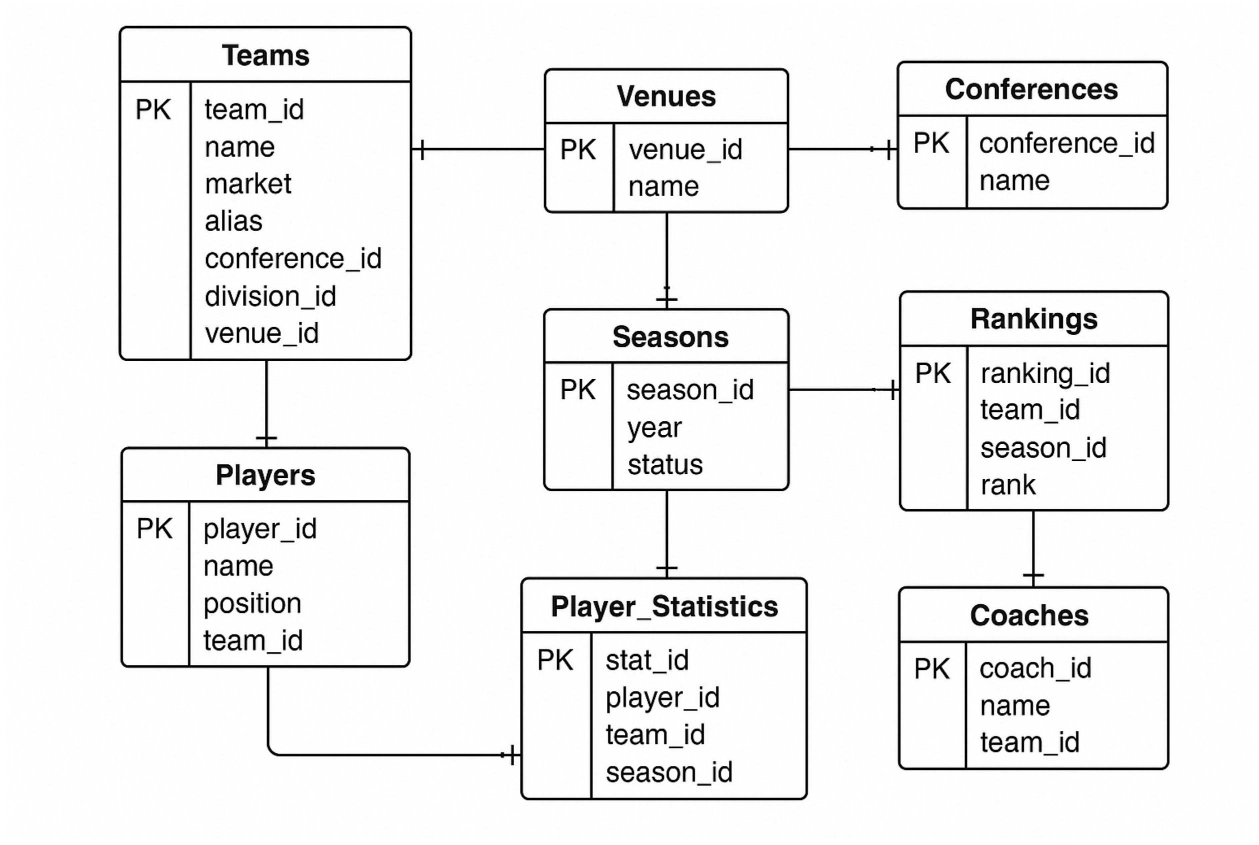
Relationships Summary

- Teams → Conference, Division, Venue
- Players → belongs to Teams
- Player_Statistics → links Players, Teams, and Seasons
- Rankings → links Teams and Seasons
- Coaches → belongs to Teams
- Seasons → independent, linked by year to rankings & stats

Relationship Summary (Quick Map)

Parent Table	Child Table	Key Link
Conferences	Teams	`Teams.conference_id` → `Conferences.conference_id`
Divisions	Teams	`Teams.division_id` → `Divisions.division_id`
Venues	Teams	`Teams.venue_id` → `Venues.venue_id`
Teams	Players	`Players.team_id` → `Teams.team_id`
Teams	Player_Statistics	`Player_Statistics.team_id` → `Teams.team_id`
Seasons	Player_Statistics	`Player_Statistics.season_id` → `Seasons.season_id`
Players	Player_Statistics	`Player_Statistics.player_id` → `Players.player_id`
Teams	Rankings	`Rankings.team_id` → `Teams.team_id`

Seasons	**Rankings**	`Rankings.season_id → Seasons.season_id`
Teams	**Coaches**	`Coaches.team_id → Teams.team_id`



Key Relationships

- `team_id` → **Links Teams to Players, Rankings_Weekly, and Season_Schedule**
- `player_id` → **Links Players to Player_Profiles**
- `season_id` → **Connects Seasons with Rankings_Weekly and Season_Schedule**
- `venue_id` → **Connects Venues with Season_Schedule**

Possible Analysis Questions

1. Which teams have maintained Top 5 rankings across multiple seasons?
2. What are the average ranking points per team by season?
3. How many first-place votes did each team receive across weeks?
4. Which players have appeared in multiple seasons for the same team?
5. What are the most common player positions and their distribution across teams?
6. How many home vs away games were played per team in a season?
7. Which venues hosted the most games across all seasons?
8. How does ranking improvement correlate with game performance (points scored)?

---

4. Streamlit Application Overview

The Streamlit app acts as an interface to execute real-time SQL queries and display results as dataframes (tables).
Core Pages / Features

1. **🏠 Home Dashboard**
   - Displays summary data tables, such as:
     - All available teams and their conferences.
     - All active players.
     - All current season years and their status.
   - Provides a quick overview of the dataset.
2. **🧩 Teams Explorer**
   - View all teams with details like team name, market, alias, conference, and venue.
   - Apply filters for conference, division, or state.
   - Search for a team by name or alias.
   - Option to view team roster by selecting a team.
3. **👥 Players Explorer**
   - Display all players with attributes like position, eligibility, height, and weight.

- Filter by position, status (Active, Injured, etc.), or eligibility year.
- Search by player name or team name.

4. 📅 **Season & Schedule Viewer**
   - List of available seasons with start and end dates.
   - Filter by year or status (open/closed).
   - Joinable with `rankings_weekly` for season-specific rankings data.
5. 🏆 **Rankings Table**
   - Display the weekly rankings from the AP Poll.
   - Columns include: week, team name, rank, points, first-place votes, wins, and losses.
   - Filters available for season, week, and rank range.
   - Supports searching for a specific team's ranking history.
6. 🏟️ **Venue Directory**
   - View list of venues and details such as city, state, capacity, and roof type.
   - Filter by state or roof type (outdoor/indoor).
7. 🧑‍🏫 **Coaches Table**
   - Display all coaches with details like name, position, and associated team.
   - Search by coach name or team.

Expected Deliverables

- A fully normalized relational schema (6–8 tables) with primary–foreign key constraints.
- SQL queries to analyze team rankings, player rosters, and match statistics.
- A Streamlit application for ranking and performance trends.

Outcome:
A centralized NCAA Football analytics system that connects rankings, teams, players, and game data into a single, relationally linked platform — empowering advanced performance and trend analysis.

Project Guidelines:

1. **Coding Standards**
    - Use meaningful names: Variables, functions, and database tables should have descriptive names.
    - Follow PEP 8 (for Python): Maintain consistent formatting with proper indentation and spacing.
    - Modularize your code: Break your code into functions or classes to enhance readability and reusability.
    - Error handling: Implement try-except blocks for handling API errors and SQL exceptions.
    - Document your code: Include docstrings and comments to explain logic and functions.
2. **SQL Database Practices**
    - Normalize tables: Avoid redundancy and ensure efficient data storage.
    - Use indexes: Optimize query performance with appropriate indexing.
    - Follow naming conventions: Use consistent and descriptive names for tables and fields.
3. **Streamlit Application Development**
    - Interactive features: Ensure the UI is responsive, with interactive widgets for filters.
    - Minimalist design: Keep the layout simple for a smooth user experience.
4. **General Best Practices**
    - Test frequently: Regularly test each component (e.g., API requests, SQL queries, Streamlit app) during development.

  - Documentation: Provide a README file with setup instructions, project objectives.

Reference:

If you don't know how to approach the project, kindly refer to the project orientation recording provided in this table.

Streamlit Doc	https://docs.streamlit.io/library/api-reference
SportRadar API key	https://console.sportradar.com/signup
SAMPLE PROJECT FOR REFERENCE(ENGLISH)	Project Excellence Series: Guided Learning …
Streamlit recording (English)	Special session for STREAMLIT(11/08/2…
Project Live Evaluation Metrics	Project Live Evaluation
Capstone Explanation Guideline	Capstone Explanation Guideline
GitHub Reference	How to Use GitHub.pptx

Timeline:

The project must be completed and submitted within **10 days from the assigned date.**

PROJECT DOUBT CLARIFICATION SESSION (PROJECT AND CLASS DOUBTS)

About Session: The Project Doubt Clarification Session is a helpful resource for resolving questions and concerns about projects and class topics. It provides support in understanding project requirements, addressing code issues, and clarifying class concepts. The session aims to enhance comprehension and provide guidance to overcome challenges effectively.
Note: Book the slot at least before 12:00 Pm on the same day

Timing: Monday to Saturday (3:30PM to 4:30PM)

Booking link :https://forms.gle/XC553oSbMJ2Gcfug9

LIVE EVALUATION SESSION (CAPSTONE AND FINAL PROJECT)

About Session: The Live Evaluation Session for Capstone and Final Projects allows participants to showcase their projects and receive real-time feedback for improvement. It assesses project quality and provides an opportunity for discussion and evaluation.
Note: This form will Open on Saturday and Sunday Only on Every Week

Timing: Monday-Saturday (5:30 PM to 6:30 PM)

Booking link : https://forms.gle/1m2Gsro41fLtZurRA