Project 1

Game Analytics: Unlocking Tennis Data with SportRadar API

Workflow:

Planning:

Define Goals

- **Extract Data**: Pull sports competition data from the Sportradar API.
- **Store Data**: Save the data in a structured **MySQL** database.
- **Visualize Data**: Create an interactive **Streamlit** app to explore the data and gain insights.

Define Scope

- Fetching data from the Sportradar API.
- Storing data in a **MySQL** database.
- Building a **Streamlit** app for data exploration and visualizations.

Define Requirements

Functional:

- **API**: Retrieve competition data.
- **Database**: Store data about competitions, complexes, and competitors.
- **App**: Filter events and visualize in Streamlit.

Schema Design

Schema design is crucial for structuring data and ensuring efficiency. It includes:

📌 **Data Modeling** – Designing tables, relationships, and entities.

- In this project, I have carefully **designed the tables** to reflect the relationships between competitions, events, and players.

- I also ensured that each table had the appropriate **primary keys** and **foreign keys** to maintain data integrity.

Additionally, I've included all the **queries for designing the tables** along with the necessary **constraints** provided below for your reference.

Table 1:

```
CREATE TABLE `categories` (
`category_id` varchar(50) NOT NULL,
  `category_name` varchar(100) NOT NULL
);
```

Table 2:

```
CREATE TABLE `competitions` (
  `competition_id` varchar(50) NOT NULL,
  `competition_name` varchar(100) NOT NULL,
  `parent_id` varchar(50) DEFAULT NULL,
  `type` varchar(20) NOT NULL,
  `gender` varchar(10) NOT NULL,
  `category_id` varchar(50) DEFAULT NULL,
  PRIMARY KEY (`competition_id`),
  KEY `category_id` (`category_id`)
);
```

Table 3:

```
CREATE TABLE `competitor_rankings` (
  `rank_id` int NOT NULL AUTO_INCREMENT,
  `Comp_rank` int NOT NULL,
  `movement` int NOT NULL,
  `points` int NOT NULL,
  `competitions_played` int NOT NULL,
  `competitor_id` varchar(50) DEFAULT NULL,
  PRIMARY KEY (`rank_id`),
  KEY `competitor_id` (`competitor_id`),
  CONSTRAINT `competitor_rankings_ibfk_1` FOREIGN KEY (`competitor_id`) REFERENCES
`competitors` (`competitor_id`)
);
```

Table 4:

```
CREATE TABLE `competitors` (
  `competitor_id` varchar(50) NOT NULL,
  `name` varchar(100) NOT NULL,
  `country` varchar(100) NOT NULL,
  `country_code` char(3) NOT NULL,
```

```
  `abbreviation` varchar(10) NOT NULL,
  PRIMARY KEY (`competitor_id`)
);
```

Table 5:

```
CREATE TABLE `venues` (
  `venue_id` varchar(50) NOT NULL,
  `venue_name` varchar(100) NOT NULL,
  `city_name` varchar(100) NOT NULL,
  `country_name` varchar(100) NOT NULL,
  `country_code` char(3) NOT NULL,
  `timezone` varchar(100) NOT NULL,
  `complex_id` varchar(50) DEFAULT NULL,
  PRIMARY KEY (`venue_id`),
  KEY `complex_id` (`complex_id`),
  CONSTRAINT `venues_ibfk_1` FOREIGN KEY (`complex_id`) REFERENCES `complexes`
(`complex_id`)
);
```

Table 6:

```
CREATE TABLE `complexes` (
  `complex_id` varchar(50) NOT NULL,
  `complex_name` varchar(100) NOT NULL,
  PRIMARY KEY (`complex_id`)
);
```

Challenges Faced in the Project

1. **Learning New Technologies**
    - This being my first project, I spent significant time learning **Streamlit**, **MySQL**, and understanding how to work with **APIs**.
    - Structuring data, integrating an API, and creating an interactive frontend were completely new skills for me, and I had to build these from the ground up.
2. **Frontend: Implementing Multiple Filters**
    - Building an interactive **Streamlit** interface with **multiple filters** (such as SelectBox, RangeSlider) posed challenges.
    - Each filter required me to write separate queries and ensure the data updated dynamically with each user interaction.
3. **Database Queries**
    - Writing efficient **SQL queries** for dynamic data retrieval, particularly when applying multiple filters, was quite complex.

- Ensuring that the queries were **optimized** for performance, especially with large datasets, and managing relationships between tables (e.g., competitions, complexes, and competitors) proved to be a challenging task.

Insights and Lessons Learned from the Project

1. **API Data Retrieval**
   - I learned how to **retrieve data from APIs**, which included making requests and handling responses effectively.
   - I also learned to manage **JSON formatting issues** and handle scenarios where the data might not be structured as expected.
2. **Writing Effective SQL Queries**
   - I understood the importance of **query optimization** and how to structure queries to ensure data retrieval is both accurate and efficient.
   - Through trial and error, I became skilled at writing **dynamic queries** that adapt to filters and user inputs, helping improve the app's functionality.
3. **Building Streamlit from Scratch**
   - I learned how to build an interactive **Streamlit interface**, focusing on providing a smooth user experience.
   - I understood the power of **real-time updates** and how important it is to integrate filters and visualizations in a user-friendly way.
4. **Connecting the Different Components**
   - Integrating the **API**, **database**, and **frontend** taught me the importance of designing systems where all components work seamlessly together.
   - I also learned how to debug issues that arise in a complex system and ensure that data flows efficiently between the backend and frontend.