**Final Project Report**

**1. Description of the business operation task for the database design. Develop the requirements for the database design.**

**Business Operation Task Description**

We are creating a database to monitor and track detailed player statistics in a soccer league. This database will be valuable for sports analysts, team managers, and scouts by providing comprehensive performance data for each player. It will include player details, match information, and individual performance metrics, enabling effective data storage, retrieval, and analysis for performance evaluation, strategic planning, and reporting.

**Why We Chose This Dataset**

We selected this dataset because soccer is a globally popular sport, and analyzing player statistics helps teams make informed decisions about player selection, training, and strategies. The dataset includes essential metrics like player identities, match details, goals, assists, and distance covered, making it ideal for our needs.

**How We Acquired the Dataset**

The dataset was sourced from Kaggle, a well-known platform for high-quality data. It was chosen for its detailed records of matches and player performances. We ensured the data was clean and well-formatted to fit our database design, ensuring consistency and accuracy.

**Requirements for Database Design**

**Entities and Attributes:**

1. **Players:**
   * **PlayerID** (Primary Key)
   * **PlayerName**
   * **ClubID** (Foreign Key)
   * **Position**
2. **Clubs:**
   * **ClubID** (Primary Key)
   * **ClubName**
3. **Matches:**
   * **MatchID** (Primary Key)
   * **Date**
   * **Location**
   * **HomeClubId**
   * **OpponentClubID** (Foreign Key to Clubs)
4. **PlayerStats:**
   * **PlayerStatsID** (Primary Key)
   * **PlayerID** (Foreign Key to Players)
   * **MatchID** (Foreign Key to Matches)
   * **MinutesPlayed**
   * **MatchPlayed**
   * **Goals**
   * **Assists**
   * **DistanceCovered**

**Relationships:**

* One-to-Many:
  + One club can have multiple players.
  + One match can involve multiple player statistics (one entry per player per match).
  + One club can participate in multiple matches, either as the main club or as an opponent.

**Constraints and Validations:**

* Ensure unique **PlayerName**.
* Unique match identifier and date.
* Total minutes played should not exceed the match duration.
* Referential integrity with foreign keys.

**Indexing:**

* Index **PlayerID**, **ClubID**, and **MatchID** in the **PlayerStats** table.
* Composite indexes for (**PlayerID**, **MatchID**).

**Security and Access Control:**

* Role-based access control for sensitive data.
* Encryption of sensitive data both at rest and in transit.

**Scalability and Performance:**

* Design for handling large data volumes.
* Use partitioning for large tables.
* Regularly monitor and optimize query performance and indexing.

**2. ER DIAGRAM:**

**PLAYER**

**PlayerID**

**ClubID**

**Player Name**

**Position**

**CLUBS**

**ClubId**

**ClubName**

**PLAYERSTATS**

**PlayerStatsID**

**PlayerID**

**MatchID**

**MinutesPlayed**

**MatchPlayed**

**Goals**

**Assists**

**DistanceCovered**

**MATCHES**

**MatchID**

**HomeClubID**

**ClubName**

**Date**

**Location**

**Opponent Club Id**

**3. RELATION SCHEMA:**

**PLAYERS**

**PlayerID ClubID Players Name Position**

**CLUBS**

**ClubId**  **ClubName**

**MATCHES**

**MatchID OpponentClubID Date Location**

**HomeClubID**

**PLAYERSTATS**

**PlayerStatsID PlayerID MatchID MinutesPlayed MatchPlayed Goals Assists DistanceCovered**

**4. Describe the process for creating the database, including the specification of tables, fields in tables, primary keys, foreign keys, indexes?**

**Process for Creating the Database**

**1. Identifying Entities and Relationships**

The first step was to identify the main entities required to monitor player statistics in a soccer league. These entities were: Players, Clubs, Matches, and PlayerStats. Each entity has specific attributes that define its characteristics and how it relates to other entities.

**2. Defining the Entity-Relationship (ER) Diagram**

We created an ER diagram to visually represent the entities, their attributes, and the relationships between them:

**Players:** Attributes include PlayerID (Primary Key), PlayerName, ClubID (Foreign Key), and Position.

**Clubs:** Attributes include ClubID (Primary Key) and ClubName.

**Matches:** Attributes include MatchID (Primary Key), MatchDate, Location, and OpponentClubID (Foreign Key).

**PlayerStats:** Attributes include PlayerStatsID (Primary Key), PlayerID (Foreign Key), MatchID (Foreign Key), MinutesPlayed, MatchPlayed, Goals, Assists, and DistanceCovered.

**3. Designing the Relational Schema**

The relational schema translates the ER diagram into a format suitable for implementation in a relational database. This schema includes the tables, their fields, primary keys to uniquely identify records, and foreign keys to establish relationships between tables.

**4. Creating Tables in Oracle APEX**

Using Oracle APEX, we created the following tables:

**Clubs:** Includes fields ClubID (Primary Key) and ClubName.

**Players:** Includes fields PlayerID (Primary Key), PlayerName, ClubID (Foreign Key referencing Clubs), and Position.

**Matches:** Includes fields MatchID (Primary Key), MatchDate, Location, and OpponentClubID (Foreign Key Referencing Clubs).

**PlayerStats:** Includes fields PlayerStatsID (Primary Key), PlayerID (Foreign Key Referencing Players), MatchID (Foreign Key Referencing Matches), MinutesPlayed, MatchPlayed, Goals, Assists, and DistanceCovered.

**5. Populating the Tables with Data**

We inserted sample data into each table to ensure that the database could be tested and used for analysis. This involved adding records for clubs, players, matches, and player statistics.

**6. Defining Indexes**

Indexes were created to improve the performance of database queries. Key columns, such as PlayerID and MatchID in the PlayerStats table, and ClubID in the Players table, were indexed to speed up data retrieval.

**Business Question 1: Which player scored the most goals?**

**SELECT** Players.PlayerName, **SUM**(PlayerStats.Goals) **AS** TotalGoals

**FROM** Players

**JOIN** PlayerStats ON Players.PlayerID = PlayerStats.PlayerID

**GROUP BY** Players.PlayerName

**ORDER BY** TotalGoals DESC

FETCH FIRST 1 ROWS ONLY;

This query identifies the player who has scored the most goals by joining the Players and PlayerStats tables and using the GROUP BY clause to aggregate the goals for each player.

A black and white image of a person

Description automatically generated with medium confidence

**Business Question 2: What are the details of matches played by Real Madrid?**

**SELECT** Matches.MatchID, Matches.MatchDate, Matches.Location, Matches.OpponentClubID, Clubs.ClubName **AS** OpponentClubName

**FROM** Matches

JOIN Clubs **ON** Matches.OpponentClubID = Clubs.ClubID

**WHERE** Matches.MatchID IN (

**SELECT** MatchID

**FROM** PlayerStats

**WHERE** PlayerID IN (

**SELECT** PlayerID

**FROM** Players

**WHERE** ClubID = 101

)

**);**

This query retrieves the details of all matches played by Real Madrid. It uses data from the Matches, Clubs, and Players tables and utilizes a subquery to find the relevant matches.

A screenshot of a computer

Description automatically generated

**How many goals did each club score in total?**

**SELECT** Clubs.ClubName, **SUM**(PlayerStats.Goals) **AS** TotalGoals

**FROM** Players

**JOIN** PlayerStats **ON** Players.PlayerID = PlayerStats.PlayerID

**JOIN** Clubs **ON** Players.ClubID = Clubs.ClubID

**GROUP BY** Clubs.ClubName;

This query calculates the total number of goals scored by each club. It joins the Players, PlayerStats, and Clubs tables and uses the GROUP BY clause to aggregate the goals for each club.

A screenshot of a computer

Description automatically generated

**Business Question 4: Which player covered the most distance?**

**SELECT** Players.PlayerName, **SUM**(PlayerStats.DistanceCovered) **AS** TotalDistanceCovered

**FROM** Players

**JOIN** PlayerStats **ON** Players.PlayerID = PlayerStats.PlayerID

**GROUP BY** Players.PlayerName

**ORDER BY** TotalDistanceCovered **DESC**

**FETCH** FIRST 1 ROWS ONLY;

This query identifies the player who covered the most distance across all matches by joining the Players and PlayerStats tables, aggregating the total distance covered for each player using the GROUP BY clause, and ordering the results in descending order

A black stripe on a black surface

Description automatically generated with medium confidence