**ERD Paragraph:**

The Team Table has attributes Team\_Name, Team\_ID, Team\_Abbreviation, Manager\_ID, and Year\_Founded, where **Team\_ID** is the **primary key**, and **Manager\_ID** is a **foreign key** pointing to the **Manager Table**. **Team\_ID** can be found as a **foreign key** in the **Player Table**.  
The Manager Table has attributes First\_Name, Last\_Name, Age, and Manager\_ID, where **Manager\_ID** is **a primary key**, and can be found as a **foreign key** in the **Team Table**.  
The Player Table has attributes First\_Name, Last\_Name, Age, Player\_ID, Team\_ID, and Shirt\_Number, where **Player\_ID** is a primary key, and **Team\_ID** is a foreign key, pointing to the **Team Table**.

**Table Normalisation**

These tables are all in BCNF, as all non-key attributes are functionally dependent on the primary keys of each table, and there are no transitive dependencies, and no non-prime attributes are dependent on any part of the table except the candidate keys.

**DDL Statements**

CREATE TABLE Team (

Team\_ID INTEGER,

Team\_Name VARCHAR(100),

Team\_Abbreviation VARCHAR(5),

Year\_Founded INTEGER,

Manager\_ID INTEGER NOT NULL,

PRIMARY KEY (Team\_ID),

FOREIGN KEY (Manager\_ID) REFERENCES Manager(Manager\_ID)

);

CREATE TABLE Manager (

Manager\_ID INTEGER

First\_Name VARCHAR(50),

Last\_Name VARCHAR(50),

Age INTEGER,

PRIMARY KEY (Manager\_ID)

);

CREATE TABLE Player (

Player\_ID INTEGER,

First\_Name VARCHAR(50),

Last\_Name VARCHAR(50),

Age INTEGER,

Team\_ID INTEGER,

Shirt\_Number INTEGER,

PRIMARY KEY (Player\_ID)

FOREIGN KEY (Team\_ID) REFERENCES Team(Team\_ID)

);

**Relational Algebra Statements**

1. Finding the average age in each team:

SELECT Team.Team\_ID, AVG(Player.Age) AS Avg\_Age, Team.Team\_Name  
FROM Player  
JOIN Team ON Player.Team\_ID = Team.Team\_ID  
GROUP BY Team.Team\_ID, Team.Team\_Name;

1. Finding the manager for each team, sorting by age:

SELECT Manager.Age, Team.Manager\_ID, Manager.First\_Name, Manager.Last\_Name

FROM Manager

JOIN Team ON Team.Manager\_ID = Manager.Manager\_ID

GROUP BY Manager.Age, Team.Manager\_ID

ORDER BY Manager.Age;

1. Deleting the manager from Arsenal

DELETE Manager.ID  
FROM Team  
WHERE Team.Name = “Arsenal FC”;

1. Finding the youngest 20 players, and the teams they play for:  
     
   SELECT Team.Team\_Name, AVG(Youngest\_Players.Age) AS Avg\_Age\_Youngest\_Players

FROM (

SELECT \*

FROM Player

ORDER BY Age ASC

LIMIT 20

) AS Youngest\_Players

JOIN Team ON Youngest\_Players.Team\_ID = Team.Team\_ID

GROUP BY Team.Team\_Name;

**DML Statements**

INSERT INTO Team(Team\_ID, Team\_Name, Team\_Abbreviation, Year\_Founded, Manager\_ID) VALUES (?,?,?,?,?);

INSERT INTO Manager(Manager\_ID, First\_Name, Last\_Name, Age)  
VALUES (?,?,?,?);

INSERT INTO Player(Player\_ID, First\_Name, Last\_Name, Age, Team\_ID, Shirt\_Number)  
VALUES (?,?,?,?,?,?);

?’s will be replaced with data in the CSV when it is all extracted, via a preparedStatement object.