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) ON DELETE CASCADE
);
```

```
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, Team
   WHERE Player.Team_ID = Team.Team_ID
   GROUP BY Player.Team_ID, Team.Team_Name;
   ```

   π Team_ID, Team_Name, ρ Avg_Age ← AVG(Age)(Player ⋈ Team) (σ Player.Team_ID = Team.Team_ID (Player x Team))

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

   ```
   SELECT Manager.Age, Team.Manager_ID, Manager.First_Name, Manager.Last_Name
   FROM Manager, Team
   WHERE Team.Manager_ID = Manager.Manager_ID
   GROUP BY Manager.Age, Team.Manager_ID
   ORDER BY Manager.Age;
   ```

   π Age, Manager_ID, First_Name, Last_Name(σ Team.Manager_ID = Manager.Manager_ID(Manager ⋈ Team))

3. Deleting Arsenal FC

```
DELETE *
FROM Team
WHERE Team_ID = 1;
```

Team ← Team – σ Team_ID = 1 (Team)

4. Deleting Arsenal's Manager:

```
DELETE *
FROM Manager
WHERE Manager_ID = 1;
```

Manager ← Manager – σ Manager_ID = 1 (Manager)

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.