Queries translated into SQL statements

First Query

Note:
? - is used to indicate the spot where user input would be used

- Return a user defined amount of teams and some simple stats (avg ppg, apg, etc) according to Wins

First define a view to get team wins

```sql
CREATE VIEW TeamWins AS
SELECT
    t.TeamID,
    t.TeamName,
    RGCS.Reg_Current_W AS Wins
FROM RegularGameCoachStats RGCS

JOIN Coach c ON RGCS.CoachID = c.CoachID
JOIN Team t ON c.TeamID = t.Team_ID;
```

Then find a view for player averages

```sql
CREATE VIEW PlayerStats AS
SELECT
    p.PlayerID,
    p.TeamID,
    p.FirstName,
    p.LastName,
    --basic stats
    COUNT(pIN.GameID) AS games_played,
    ROUND(AVG(pIN.PTS),1) AS avgPoints,
    ROUND(AVG(pIN.AST),1) AS avgAssists,
    ROUND(AVG(pIN.STL),1) AS avgSteals,
    -- more advanced stats
    ROUND(AVG(pIN.MP),1) AS avgMinutesPlayed,
    ROUND(AVG(pIN.FG), 1) AS avgFieldGoalsMade,
    ROUND(AVG(pIN.FGA), 1) AS avgFieldGoalsAttempted,
    ROUND(AVG(pIN.'3P'), 1) AS avgThreePointersMade,
    ROUND(AVG(pIN.'3PA'), 1) AS avgThreePointersAttempted,
    ROUND(AVG(pIN.FT), 1) AS avgFreeThrowsMade,
    ROUND(AVG(pIN.FTA), 1) AS avgFreeThrowsAttempted,
    -- now derived stats
    ROUND(CAST(SUM(pIN.FG) AS FLOAT) / NULLIF(SUM(pIN.FGA), 0), 3) AS fieldGoalPct,
    ROUND(CAST(SUM(pIN.'3P') AS FLOAT) / NULLIF(SUM(pIN.'3PA'), 0), 3) AS threePointPct,
    ROUND(CAST(SUM(pIN.FT) AS FLOAT)/ NULLIF(SUM(pIN.FTA), 0), 3) AS freeThrowPct
From Player p
JOIN PlayInGame pIN ON p.PlayerID = pIN.PlayerID
GROUP BY p.PlayerID, p.FirstName,p.LastName, p.TeamID;
```

Finally we can find return the top teams and their corresponding stats made it a view for ease of use

```sql
CREATE VIEW TeamStats AS
Select
    TW.Wins,
    TW.TeamName,
    TW.TeamID,
    ROUND(AVG(pS.avgPoints),1) AS avgTeamPoints,
    ROUND(AVG(pS.avgAssists),1) AS avgTeamAssists,
    ROUND(AVG(pS.avgSteals),1) AS avgTeamSteals,
    ROUND(AVG(pS.fieldGoalPct),3) AS TeamFGPct,
    ROUND(AVG(pS.threePointPct),3) AS Team3PointPct,
    ROUND(AVG(pS.freeThrowPct),3) AS TeamFreeThrowPct

From TeamWins TW
JOIN PlayerStats pS ON pS.TeamID = TW.TeamID
GROUP BY TW.TeamID, TW.TeamName,TW.Wins
```

final query (not a view so we can use LIMIT)

```sql
SELECT * FROM TeamStats ORDER BY Wins DESC LIMIT ?;
```

Second Query

- Return the coaches that have coached more than 5 seasons with a positive win rate in the playoffs
  - order by winrate

```sql
WITH vetCoach AS (
SELECT
    c.CoachName,
    c.CoachID
FROM Coach c
WHERE c.SeasonsOverall > 5)

SELECT
    vC.CoachName,
    ROUND(CAST(PGCS.Playoffs_Overall_W AS FLOAT)/PGCS.Playoffs_Overall_G,3) AS PlayoffWinRate

FROM vetCoach vC
JOIN PlayoffGameCoachStats PGCS ON PGCS.CoachID = vC.CoachID
WHERE PGCS.Playoffs_Overall_W > PGCS.Playoffs_Overall_L
ORDER BY PlayoffWinRate DESC;
```

Third Query

- Return the average ppg for a specific team, with the option to return the avg ppg for the league

  I will use the TeamStats view I defined earlier to easily retrieve the ppg for a specific team

```sql
SELECT TeamName, avgTeamPoints AS teamAvgPPG
FROM TeamStats
WHERE TeamName = ?;
```

Option for the league average

```
SELECT
    ts.TeamName,
    ts.avgTeamPoints,
    ts.TeamID,
  ( SELECT ROUND(AVG(ts.avgTeamPoints),1) FROM TeamStats )AS avgLeaguePoints

FROM TeamStats ts
WHERE ts.TeamName = ?;
```

forth query

- return a user defined amount players and their ppg,apg and spg for the season
  - Group by position and order by height within that position

changed this from the original query as it was trying to do to much while also requiring stats that don't exist

```
SELECT
    ps.avgPoints,
    ps.avgAssists,
    ps.avgSteals,
    ps.FirstName,
    ps.LastName,
    pi.Height,
    pi.Position
FROM PlayerStats ps
JOIN Player_information pi ON ps.PlayerID = pi.PlayerID
ORDER BY pi.Position, pi.Height
LIMIT ? OFFSET ?;
```

Since the query can return a ton of results we want to let the user page through them in python: Where limit is the number of results per page (page size) and offset = (page - 1) * page size, and offset skips a certain amount of results

fifth Query

- return the team and corresponding coach that won the championship that year

```
SELECT
    c.CoachName,
    t.TeamName,
    PGCS.Playoffs_Current_W as playoffWins
FROM PlayoffGameCoachStats PGCS
JOIN Coach c ON PGCS.CoachID = c.CoachID
JOIN Team t ON c.TeamID = t.TeamID
-- 16 wins means they won 4 games in each of the 4 rounds, meaning they won
-- the championship
WHERE PGCS.Playoffs_Current_W = 16;
```

sixth query

- list the roster of a team ordered by ppg, should also show apg and spg
  - in case of a tie the tie breaker should be apg, then spg

```
SELECT
    t.TeamName,
    p.FirstName,
    p.LastName,
    ps.avgPoints,
    ps.avgAssists,
    ps.avgSteals
FROM Team t
JOIN Player p ON t.TeamID = p.TeamID
JOIN PlayerStats ps ON p.PlayerID = ps.PlayerID
WHERE t.TeamName = ?
ORDER BY ps.avgPoints DESC,ps.avgAssists DESC,ps.avgSteals DESC;
```

7th query

- return a user defined amount players ordered by a major category inputted by the User: ppg, apg, spg

  Use player stat view from earlier

Points:

```
SELECT
    ps.FirstName,
    ps.LastName,
    ps.avgPoints,
    ps.avgAssists,
    ps.avgSteals
FROM PlayerStats ps
ORDER BY ? DESC
-- options should be avgPoints,avgAssists, or avgSteals
Limit ?;
```

8th Query

- for a team return the average age of their players
  - maybe extend this to connect with other Queries
    - ex. Maybe we want to see the average age of the top 10 teams by wins
- Note this is the age of the player per the end of the season

  made it a view so it would be easy to connect with other queries in the future if we wanted

```sql
CREATE VIEW GetTeamAvgAge AS
WITH PlayersAge AS(
SELECT
    t.TeamID,
    t.TeamName,
    p.PlayerID,
    p.FirstName,
    p.LastName,

    YEAR('2024-6-16') - YEAR(pi.Birthdate) AS playerAge
    -- year function in MYSQL not SQLite though
    -- assuming playoffs end on june 16
FROM Player_information pi
JOIN Player p ON pi.PlayerID = p.PlayerID
JOIN Team t ON p.TeamID = t.TeamID
)
SELECT
    pa.TeamName,
    ROUND(AVG(pa.playerAge),1) as avgTeamAge
FROM PlayersAge pa
GROUP BY pa.TeamName;
```

Actual query that would take user input

```sql
SELECT * FROM GetTeamAvgAge WHERE TeamName = ?;
```

9th query

- return both teams stats for a specific game

```sql
WITH playersGame AS (
  SELECT
    Team.TeamID,
    HomePTS,
    VisitorPTS,
    STL,
    FG,
    FGA,
    "3P",
    "3PA",
    AST,
    FTA,
    FT
  FROM Game
  JOIN PlayInGame ON Game.GameID = PlayInGame.GameID
  JOIN Player ON PlayInGame.PlayerID = Player.PlayerID
  JOIN Team ON Player.TeamID = Team.TeamID
  WHERE Game.GameID = ?
)

SELECT
```

```
  TeamID,
  MAX(HomePTS) AS HomePTS,
  MAX(VisitorPTS) AS VisitorPTS,
  SUM(STL) AS STL,
  SUM(FG) AS FG,
  SUM(FGA) AS FGA,
  SUM(FG)  * 1.0 / NULLIF(SUM(FGA), 0) AS FGP,
  SUM("3P") AS "3P",
  SUM("3PA") AS "3PA",
  SUM("3P") * 1.0 / NULLIF(SUM("3PA"), 0) AS "3PP",
  SUM(FT) AS FT,
  SUM(FTA) AS FTA,
  SUM(FT)  * 1.0 / NULLIF(SUM(FTA), 0) AS FTP,

  SUM(AST) AS AST
FROM playersGame
GROUP BY TeamID;
```

10th query

- list a user defined amount of players in the league for 3 point percentage
  - player must have over a user defined amount of attempts to be included

```
SELECT
  Player.PlayerID,
  FirstName,
  LastName,
  SUM("3P") * 1.0 / NULLIF(SUM("3PA"), 0) AS "3PP"
FROM Player

JOIN PlayInGame ON Player.PlayerID = PlayInGame.PlayerID
GROUP BY Player.PlayerID, FirstName, LastName
HAVING SUM("3PA") > ?
ORDER BY "3PP" DESC
LIMIT ?;
```

11th query

List the players name and PTS in a game (can be chosen) that have a height over 6ft 5in and play the position of center

```
--Field Goals = 2pts and 3 pointers = 3pts and Free Throws = 1pt
SELECT first name, last name, ( (FG-3P)*2 + (3P * 3) + (FT) ) as PTS
FROM commonPlayerInfo CPI JOIN Player
ON CPI.PlayerID = Player.PlayerID
JOIN playsIn
ON Player.PlayerID = playsIn.PlayerID
WHERE gameID = ? AND
CPI.height > '6"5'
AND CPI.position = 'Center';
```

12th query

Return the win % from one home team to a visiting team

```
--first filter the games only played by those 2 teams maybe not by team id
WITH FirstTeamID AS
(SELECT Team_id
FROM team
WHERE Team_name = ?),
SecondTeamID AS
(SELECT Team_id
FROM team
WHERE Team_name = ?),
-- get a count of how many games they played
CountAllMatchups AS (SELECT count(gameID)
FROM
game WHERE
homeTeamID = firstTeamID
AND visitorTeamID = SecondTeamID)
SELECT (count(gameID)/CountAllMatchups) * 100 AS winPCT
FROM game
WHERE HomePTS > VisitorPTS
AND homeTeamID = firstTeamID
AND visitorTeamID = SecondTeamID; - gets the win %
```

13th query

List the coach names who have made it to the 2024-2025 season playoffs and have a overall playoff win % at least as high as the user input, order coach's from playoff win % from best to worst.

```
--List all coachIDs that have made it in the playoffs you have to have Playoffs_overall_W > 0
WITH CoachesInRecentPlayoff as
(SELECT CoachID
FROM PlayoffGameCoachStats PGCS
WHERE Playoffs_Overall_G > 0)
-- Select the coaches that made the playoffs and have a sufficent overall win rate as the user requeste
SELECT Coach_name, (PGCS.Playoffs_Overall_W / PGCS.Playoffs_Overall_G) AS overallWinRate
FROM Coach JOIN PlayoffGameCoachStats AS PGCS
ON Coach.CoachID = PGCS.CoachID
WHERE  overallWinRate >= ?
AND PGCS.coachID IN (CoachesInRecentPlayoff)
ORDER BY overallWinRate DESC;
```

List all arenas that every team has won in.

```
WITH winnerOfGame AS (
  SELECT
    Team.TeamID,
    ArenaName,
    HomePTS,
    VisitorPTS
  FROM Team, Arena
```

```
  JOIN Game ON Arena.ArenaName = Game.ArenaName
  WHERE
    (Team.TeamID = Game.HomeTeamID AND HomePTS > VisitorPTS)
    OR
    (Team.TeamID = Game.VisitorTeamID AND VisitorPTS > HomePTS)
),

all_pairs AS (
  SELECT
    TeamID,
    ArenaName
  FROM Team, Arena
)

SELECT ArenaName
FROM all_pairs as ap1
WHERE NOT EXISTS (
  SELECT *
  FROM all_pairs as ap2
  WHERE ap1.ArenaName = ap2.ArenaName
  AND NOT EXISTS (
    SELECT *
    FROM winnerOfGame as wog
    WHERE (
      wog.TeamID = ap2.TeamID
      AND wog.ArenaName = ap2.ArenaName
    )
  )
)
```

Simple Queries to retrieve tables

1. List every player draft combine stats that has attended in any draft combine

```
SELECT *
FROM DraftCombine
```

2. List all drafts in every season from the drafts table

```
SELECT *
FROM Drafts
```

3. List every coach with their regular game stats and their Playoff game stats

```
SELECT *
FROM Coach AS C
JOIN RegularGameCoachStats RGCS
ON C.CoachID = RGCS.CoachID
JOIN PlayoffGameCoachStats AS PGCS
ON C.CoachID = PGCS.CoachID
```

4. List all Arenas in the dataset

```
SELECT *
FROM Arena
```

5. List all Teams in the dataset

```
SELECT *
FROM Team
```

6. List all Games in the dataset

```
SELECT *
FROM Game
```

7. List all Players with their Player Information in the Dataset

```
SELECT *
FROM Player
LEFT JOIN PlayerInformation
ON Player.PlayerID = PlayerInformation.PlayerID
```