Tutorial 8 Solutions

Exercise 1:

1. **Increase the `Salary` of all players in the 'Forward' position by 5%.**

```
use SDB
UPDATE Players
SET Salary = Salary * 1.05
WHERE Position = 'Forward';
```

2. **Update the `CoachName` for the team 'Eagles' to 'Coach Taylor' if their current number of players is less than 15.**

```
UPDATE Teams
SET CoachName = 'Coach Taylor'
WHERE TeamName = 'Eagles'
  AND (
      SELECT COUNT(PlayerID)
      FROM Players
      WHERE Players.TeamID = Teams.TeamID
  ) < 15;
```

3. **Reduce the `Salary` of all players whose salary exceeds $150,000 more than the average salary of all players by 12%.**

```
UPDATE Players
SET Salary = Salary * 0.88
WHERE Salary > (
    SELECT AVG(Salary)
    FROM Players
) + 150000;
```

Exercise 2:

1. **Insert a new player into the `Players` table...**

```
INSERT INTO Players (FirstName, LastName, TeamID, Position, Salary)
VALUES ('Serena', 'Williams', 5, 'Guard', 1000000);
```

2. **Delete the game record for the game with `GameID = 50`.**

```
DELETE FROM Games
WHERE GameID = 50;
```

3. **Delete all games that were played more than 6 months ago.**

- To solve conflicting issues with the 'Stats' table, we first delete the corresponding stats (from the 'Stats' table), then we delete the games from the 'Games' table:

```
DELETE FROM Stats
WHERE GameID IN (
    SELECT GameID
    FROM Games
    WHERE GameDate < DATEADD(MONTH, -6, GETDATE())
);
```

```
    DELETE FROM Games
    WHERE GameDate < DATEADD(MONTH, -6, GETDATE());
```

3. **Insert a new team...**

```
INSERT INTO Teams (TeamName, City)
VALUES ('The Dragons', 'Atlanta');
```

4. **Insert a new game... between the team 'Bears' and the team 'Sharks'.**

```
INSERT INTO Games (HomeTeamID, AwayTeamID, GameDate, Attendance)
VALUES (
    (SELECT TeamID FROM Teams WHERE TeamName = 'Bears'),
    (SELECT TeamID FROM Teams WHERE TeamName = 'Sharks'),
    GETDATE(),
    25000
);
```

Exercise 3:

1. **Create a view named `Players_HighSalary` ...**

```
CREATE VIEW Players_HighSalary AS
SELECT PlayerID, FirstName, LastName, Salary
FROM Players
WHERE Salary > 500000;
```

2. **Create a view named `Team_Revenue` ...**

```
CREATE VIEW Team_Revenue AS
SELECT
    Teams.TeamName,
    SUM(Games.Attendance * 50) AS TotalRevenue
FROM Teams
INNER JOIN Games  ON Teams.TeamID = Games.HomeTeamID OR Teams.TeamID =
Games.AwayTeamID
GROUP BY Teams.TeamName;
```

3. **Create a view named `Star_Players` of players who have scored a total of more than 500 points...**

```
CREATE VIEW Star_Players AS
SELECT
    Players.PlayerID,
    Players.FirstName,
    Players.LastName,
    SUM(Stats.PointsScored) AS TotalCareerPoints
FROM Players
INNER JOIN Stats  ON Players.PlayerID = Stats.PlayerID
GROUP BY Players.PlayerID, Players.FirstName, Players.LastName
HAVING SUM(Stats.PointsScored) > 500;
```

Exercise 4:

1. **Return a list of all views currently defined in the database.**

```
use SDB;
SELECT TABLE_NAME
FROM INFORMATION_SCHEMA.TABLES
WHERE TABLE_TYPE = 'VIEW'
```

or:

```
    use SDB;
SELECT name AS ViewName
FROM sys.views;
```

2. **List all tables in the database that were created in the current calendar year.**

```
use SDB;
SELECT name AS TableName
FROM sys.tables
WHERE create_date >= DATEFROMPARTS(YEAR(GETDATE()), 1, 1);
```

3. **List the names of all columns across all tables whose name contains the word 'Name'.**

```
USE SDB;
SELECT COLUMN_NAME
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME like '%Name%';
```

Exercise 5:

1. **Create a view named `Player_Game_Performance` ...**

```
use sdb;
go
```

```sql
CREATE VIEW Player_Game_Performance AS
SELECT
    Players.FirstName + ' ' + Players.LastName AS PlayerFullName,
    Teams.TeamName,
    Games.GameDate,
    Stats.PointsScored,
    case
    when stats.Fouls IS NULL or stats.Fouls=0 Then NULL
    else
    Stats.PointsScored/Stats.Fouls
    end
    AS PerformanceRatio
FROM Stats
INNER JOIN Players  ON Stats.PlayerID = Players.PlayerID
INNER JOIN Teams  ON Players.TeamID = Teams.TeamID
INNER JOIN Games ON Stats.GameID = Games.GameID;
```

2. **Find the `TeamName` and `City` of teams that do not have any players currently assigned to them.**

```sql
SELECT Teams.TeamName, Teams.City
FROM Teams
LEFT OUTER JOIN Players  ON Teams.TeamID = Players.TeamID
WHERE Players.PlayerID IS NULL;
```

3. **Display the `GameID`, the `GameDate`, and the `Attendance` in a text format...**

```sql
SELECT
    GameID,
    GameDate,
    Attendance,
    CASE
        WHEN Attendance < 10000 THEN 'Low Attendance'
        WHEN Attendance BETWEEN 10000 AND 30000 THEN 'Moderate Crowd'
        WHEN Attendance > 30000 THEN 'Sellout Game'
        ELSE 'Attendance Data Missing'
    END AS CrowdStatus
FROM Games;
```

4. **Increase the `Salary` of all players using the following logic...**

```sql
UPDATE Players
SET Salary = Salary *
    CASE Position
        WHEN 'Guard' THEN 1.05
        WHEN 'Center' THEN 1.10
        ELSE 1.00
    END;
```

5. **Find the `FirstName` and `LastName` of players who have never committed a foul...**

```sql
SELECT P.FirstName, P.LastName
FROM Players P
WHERE P.PlayerID NOT IN (
SELECT S.PlayerID
FROM Stats S
WHERE S.Fouls > 0
);
```

6. **List the `TeamName` for teams whose players have an average salary higher than $800,000.**

```sql
SELECT Teams.TeamName
FROM Teams
INNER JOIN Players  ON Teams.TeamID = Players.TeamID
GROUP BY Teams.TeamName, teams.teamid
HAVING AVG(Players.Salary) > 800000;
```

7. **Find the `TeamName` for teams whose players have never scored more than 10 points in any single game.**

```sql
SELECT TeamName
FROM Teams
WHERE TeamID NOT IN (
    SELECT Players.TeamID
    FROM Players
    INNER JOIN Stats  ON Players.PlayerID = Stats.PlayerID
    WHERE Stats.PointsScored > 10
);
```