SQL Project with Premier League Dataset from Kaggle

Sports have always been an interest and passion of mine and being a former collegiate soccer player, I naturally have an interest in the English Premier League which is arguably the world's top football (soccer) league. It's always fun to look a bit further into the statistics of the league and see what you can find. See the summary below for how I conducted analysis on a dataset from the 2023-24 season in the English Premier League. This dataset was sourced from Kaggle.

Here are the hypothetical scenarios that I created to analyze the data, along with my solutions and results.

1. Find the Top 10 Scorers on the Season

```sql
-- Top 10 Scorers for 23-24 season
-- Shows the top 10 players with the most goals scored on the season

select
      s.player_name
    , s.team
    , s.goals
from
    premier_league_player_stats s
order by
    s.goals desc
limit
    10
;
```

A simple query to start to retrieve the top 10 players with the most goals scored during the 2023-24 season.

	player_name character varying (255)	team character varying (255)	goals integer
1	Erling Haaland	Manchester City	27
2	Cole Palmer	Chelsea	22
3	Alexander Isak	Newcastle United	21
4	Dominic Solanke	Bournemouth	19
5	Ollie Watkins	Aston Villa	19
6	Phil Foden	Manchester City	19
7	Mohamed Salah	Liverpool	18
8	Son Heung-min	Tottenham Hotspur	17
9	Jarrod Bowen	West Ham United	16
10	Bukayo Saka	Arsenal	16

I selected the player's name, team, and number of goals they scored. I ordered the data by goals in descending order to rank the highest scorers at the top which shows us that Erling Haaland of Manchester City led the way with 27 goals on the year.

2. Top 5 Teams by Non-Penalty Goals

```sql
-- Top 5 Teams by Non-Penalty Goals
-- Shows the top 5 teams with the highest total number of non-penalty goals scored by players

select
      s.team
    , sum(s.goals_minus_penalty_goals) as team_non_penalty_goals
from
    premier_league_player_stats s
group by
    s.team
order by
    team_non_penalty_goals desc
limit
    5
;
```

Identifying the top five teams based on total non-penalty goals scored by their players. The sum function calculates the total non-penalty goals per team, while the group by clause groups these results by team to aggregate non-penalty goals for each team.

	team character varying (255)	team_non_penalty_goals bigint
1	Manchester City	85
2	Arsenal	76
3	Newcastle United	75
4	Liverpool	74
5	Aston Villa	68

We can see that Manchester City led the way with 85 total non-penalty goals as a team followed by Arsenal (76) and Newcastle United (75). No surprise as Manchester City had two players among the top 10 scorers from the previous result set.

3. Most Efficient Goal Scorers

```sql
--Most Efficient Goal Scorers (Highest Goals per 90mins vs. Expected Goals per 90mins)
--Identifies the players who outperformed their expected goals the most, i.e, players with the highest difference between goals
--per 90 minutes and expected goals per 90 minutes

select
      pn.player_name
    , pn.team
    , pn.minutes
    , pn.goals_per_90
    , pn.exp_goals_per_90
    , (pn.goals_per_90 - pn.exp_goals_per_90) as goal_production_performance
from
    premier_league_stats_per_90 pn
where
    1=1
    and (pn.goals_per_90 - pn.exp_goals_per_90) > 0 -- only includes players who outperformed
    and minutes > 1000 --only includes players who played 1000 minutes or more on the season
order by
    goal_production_performance desc
limit
    10
;
```

I wanted to highlight the most efficient goal scorers in the league, showing players who scored more goals per 90 minutes than their expected goals per 90 minutes. Expected goals (xG) per 90 minutes is a metric that measures the quality of a team or player's chances to score during a match. (xG) is calculated by assigning a value between 0 and 1 to each shot, based on the likelihood of it resulting in a goal. This value is based on many factors, including the distance and angle to the goal, the type of shot, and the position of the defenders. I calculated the difference between goals_per_90 and exp_goals_per_90 to assess how much a player exceeded expectations. I also added a where clause to filter out players who played fewer than 1000 minutes on the season to ensure meaningful data.

	player_name character varying (255)	team character varying (255)	minutes integer	goals_per_90 numeric (4,2)	exp_goals_per_90 numeric (4,2)	goal_production_performance numeric
1	Diogo Jota	Liverpool	1145	0.79	0.42	0.37
2	Michael Olise	Crystal Palace	1275	0.71	0.39	0.32
3	Phil Foden	Manchester City	2857	0.60	0.33	0.27
4	Callum Hudson-Odoi	Nottingham Forest	1854	0.39	0.13	0.26
5	Elijah Adebayo	Luton Town	1419	0.63	0.37	0.26
6	Noni Madueke	Chelsea	1053	0.43	0.21	0.22
7	Leandro Trossard	Arsenal	1649	0.65	0.43	0.22
8	Jean-Philippe Mateta	Crystal Palace	2282	0.63	0.43	0.20
9	Hwang Hee-chan	Wolverhampton	2119	0.51	0.33	0.18
10	Leon Bailey	Aston Villa	2068	0.44	0.27	0.17

We can see some big-name players in the top 3 here, Liverpool superstar Diogo Jota exceeded expectations the most of any player with this criteria, while Michael Olise was second on our list, earning him a move to German club Bayern Munich and Phil Foden who was named Player of the Season.

4. Age vs. Offensive Performance

```
--Age vs. Offensive Performance
--Analyze whether older players offensively performed better or worse than younger players
--Offensive performance based on goals per 90 minutes and assists per 90 minutes

select
    case
        when
            p.age > 27 then 'Players Over 27'
        else
            'Players Under 27'
        end as
            age_group
    , round(avg(pn.goals_per_90), 4) as avg_goals_per_90
    , round(avg(pn.assists_per_90), 4) as avg_assists_per_90
    , round(avg(pn.goals_per_90) + avg(pn.assists_per_90), 4) as combined_avg_per_90
from
    premier_league_stats_per_90 pn
join
    premier_league_players p
        on pn.player_name = p.player_name
group by
    age_group
;
```

This query compares offensive performance (goals and assists per 90 minutes) between players over and under the age of 27. The case statement groups players into either the “Over 27” or “Under 27” category. I then calculated the averages for goals_per_90, assists_per_90, and their combined total for each of these age groups. The join connected the player stats to the player data for age information.

	age_group text	avg_goals_per_90 numeric	avg_assists_per_90 numeric	combined_avg_per_90 numeric
1	Players Under 27	0.1319	0.1003	0.2322
2	Players Over 27	0.1047	0.0745	0.1792

From the result set we can conclude that players under 27 generally are performing better offensively, averaging a higher number of goals and assists per 90 minutes compared to players over 27. Younger players showed a combined offensive contribution of 0.2322 per 90 minutes, while older players contributed 0.1792, indicating a slight decline in offensive output with age.

5. Filtering Players with High Discipline

```sql
--Filtering players with high discipline
--Using a temp table to only show players with no more than 3 yellow cards or 1 red card and finding top 5 scorers from this group

create temp table disciplined_players as
select
      s.player_name
    , s.team
    , s.pos
    , s.goals
    , s.yellow_cards
    , s.red_cards
from
    premier_league_player_stats s
where
    yellow_cards <= 3 and red_cards = 1;

-- Querying the temp table to find the top 5 scorers
select
      player_name
    , team
    , pos
    , goals
from
    disciplined_players
order by
    goals desc
limit
    5
;
```

Continuing to explore different factors in the statistics, I wanted to find the top five goal scorers from a group of high-disciplined players, those that had no more than three yellow cards and no more than one red card. A temp table was created to store players who met the yellow/red card criteria, and the second query pulled the data from the temp table to find the top five scorers from this group.

	player_name character varying (255)	team character varying (255)	pos character varying (255)	goals integer
1	Erling Haaland	Manchester City	FW	27
2	Alexander Isak	Newcastle United	FW	21
3	Dominic Solanke	Bournemouth	FW	19
4	Phil Foden	Manchester City	FW,MF	19
5	Mohamed Salah	Liverpool	FW	18

6. Best Goal-to-Minutes Ratio

```sql
--Calculating Best Goal-to-Minutes Ratio
--Identifies the player with the best goal-to-minutes ratio (goals per minute played), only considering players who played at least 1000 minutes

select
      s.player_name
    , s.goals
    , s.minutes
    , round(cast(s.goals as numeric) / cast(s.minutes as numeric), 5) as goals_per_minute
from
    premier_league_player_stats s
where
    minutes >= 1000
order by
    goals_per_minute desc
limit
    10
;
```

Here I am finding the players with the best goals-to-minutes ratio, indicating goal scoring efficiency. The where clause filters to only players with at least 1000 minutes played similar to an earlier query.

	player_name character varying (255)	goals integer	minutes integer	goals_per_minute numeric
1	Erling Haaland	27	2552	0.01058
2	Alexander Isak	21	2255	0.00931
3	Diogo Jota	10	1145	0.00873
4	Cole Palmer	22	2607	0.00844
5	Michael Olise	10	1275	0.00784
6	Chris Wood	14	1812	0.00773
7	Richarlison	11	1491	0.00738
8	Leandro Trossard	12	1649	0.00728
9	Mohamed Salah	18	2534	0.00710
10	Elijah Adebayo	10	1419	0.00705

From the result set we see that Haaland remained in the top position but the rest of the scorers were shuffled around when compared to the top 10 scorers list that I outputted in the beginning.

7. Creating a View for Player Performance

```sql
--Creating View to store data for later visualizations

create view player_performance_view as
    with goals_per_minute_cte as (
        select
              s.player_name
            , s.team
            , s.pos
            , s.goals
            , s.minutes
            , (s.goals::float / s.minutes::float) as goals_per_minute
    from
        premier_league_player_stats s
    where
        minutes >= 500
)
    select
         p.player_name
        , p.team
        , p.pos
        , p.age
        , case
            when p.age > 27 then 'Older Players'
            else 'Younger Players'
    end as
          age_group
        , ps90.goals_per_90
        , ps90.assists_per_90
        , (ps90.goals_per_90 + ps90.assists_per_90) as total_contributions_per_90
        , s.yellow_cards
        , s.red_cards
        , gpm.goals_per_minute
    from
        premier_league_player_stats s
join
    premier_league_players p on s.player_name = p.player_name
join
    premier_league_stats_per_90 ps90 on s.player_name = ps90.player_name
join
    goals_per_minute_cte gpm on s.player_name = gpm.player_name
;
```

Lastly, I created a view to consolidate key player statistics (goals per minute, goals per 90 minutes, assists per 90 minutes) to be used for later visualizations. A CTE was used to calculate the goals_per_minute for players with more than 500 minutes played and the main query joins multiple tables to pull age, yellow/red cards, and offensive stats.

That's it. Thanks for taking the time to explore!