Data Engineering Project Scripts

Transformation Scripts

—Team_Data.csv

 # 1. Filter teams with possession greater than 50% and avg_age less than 25
possession = team_data.filter((col("possession") > 50) & (col("avg_age") < 25))

 # 2. Add a new column for goal contribution per 90 minutes (goals + assists)
goal_contrib_per90 = team_data.withColumn("goal_contrib_per90", col("goals_per90") + col("assists_per90"))

 # 3. Calculate the average goals per team
avg_goals = team_data.groupBy("team").agg(avg("goals").alias("avg_goals"))

 # 4. Create a column to classify players as "High", "Medium", or "Low" scorers based on goals_per90
```
scorer_category = team_data.withColumn(
    "scorer_category",
    when(col("goals_per90") > 0.5, "High")
    .when((col("goals_per90") <= 0.5) & (col("goals_per90") > 0.2), "Medium")
    .otherwise("Low")
)
```

 # 5. Calculate total minutes played per team
total_minutes = team_data.groupBy("team").agg(sum("minutes").alias("total_minutes"))

 # 6. Filter goalkeepers based on goals_against_per90 (less than 1.0)
goalkeepers = team_data.filter((col("gk_goals_against_per90") < 1.0) & (col("gk_games").isNotNull()))

 # 7. Calculate average passes completed per game per team

avg_passes_per_game = team_data.groupBy("team").agg((sum("passes_completed") / sum("games")).alias("avg_passes_per_game"))

 # 8. Add a new column for shot accuracy (shots_on_target / shots)

```
shot_accuracy = team_data.withColumn("shot_accuracy", (col("shots_on_target") / col("shots"))
* 100)


# 9. Filter teams with an average xG (expected goals) greater than 1.5
high_xg = team_data.groupBy("team").agg(avg("xg").alias("avg_xg")).filter(col("avg_xg") > 1.5)

# 10. Add a column for progressive passes ratio (progressive_passes / passes)
progressive_pass_ratio = team_data.withColumn(
    "progressive_pass_ratio", when(col("passes") > 0, col("progressive_passes") /
col("passes")).otherwise(0)
)

# Show the results (cropping to the first 10 rows for brevity and truncating output)

possession.select("team", "possession", "avg_age").show(10, truncate=False)

goal_contrib_per90.select("team", "goal_contrib_per90").show(10, truncate=False)

avg_goals.show(10, truncate=False)

scorer_category.select("team", "scorer_category").show(10, truncate=False)

total_minutes.show(10, truncate=False)

goalkeepers.select("team", "gk_goals_against_per90", "gk_games").show(10, truncate=False)

avg_passes_per_game.show(10, truncate=False)

shot_accuracy.select("team", "shot_accuracy").show(10, truncate=False)

high_xg.show(10, truncate=False)

progressive_pass_ratio.select("team", "progressive_pass_ratio").show(10, truncate=False)
```

— Group_Stats.csv

```
# 1. Calculate win percentage
win_percentage = group_stats_spark.withColumn("win_percentage", (col("wins") /
col("matches_played")) * 100)
```

```
# 2. Calculate average goals scored per match
avg_goals_scored = win_percentage.withColumn("avg_goals_scored", col("goals_scored") /
col("matches_played"))

# 3. Calculate average goals conceded per match
avg_goals_conceded = avg_goals_scored.withColumn("avg_goals_conceded",
col("goals_against") / col("matches_played"))

# 4. Add a column for point efficiency (points per match)
points_per_match = avg_goals_conceded.withColumn("points_per_match", col("points") /
col("matches_played"))

# 5. Rank teams by points and goal difference (requires Window function)
from pyspark.sql.window import Window
from pyspark.sql.functions import rank

window_spec = Window.orderBy(col("points").desc(), col("goal_difference").desc())
ranked = points_per_match.withColumn("rank_by_points_and_goals",
rank().over(window_spec))

# 6. Calculate the difference between expected and actual goals scored
goal_diff_vs_expected = ranked.withColumn("goal_diff_vs_expected", col("goals_scored") -
col("expected_goal_scored"))

# 7. Add a column for defensive efficiency
defensive_efficiency = goal_diff_vs_expected.withColumn(
    "defensive_efficiency", col("exp_goal_conceded") / col("goals_against")
)

# 8. Calculate goal difference per 90 minutes
goal_diff_per_90 = defensive_efficiency.withColumn(
    "goal_diff_per_90", col("goal_difference") / col("matches_played")
)

# 9. Identify overachievers
is_overachiever = goal_diff_per_90.withColumn(
    "is_overachiever", col("points") > col("exp_goal_difference")
)

# 10. Add a performance index
performance_index = is_overachiever.withColumn(
    "performance_index",
    (col("points") + col("goal_difference") + col("win_percentage")) / 3
```

```
)

win_percentage.select("team", "win_percentage").show(10, truncate=False)

avg_goals_scored.select("team", "avg_goals_scored").show(10, truncate=False)

avg_goals_conceded.select("team", "avg_goals_conceded").show(10, truncate=False)

points_per_match.select("team", "points_per_match").show(10, truncate=False)

ranked.select("team", "rank_by_points_and_goals").show(10, truncate=False)

goal_diff_vs_expected.select("team", "goal_diff_vs_expected").show(10, truncate=False)

defensive_efficiency.select("team", "defensive_efficiency").show(10, truncate=False)

goal_diff_per_90.select("team", "goal_diff_per_90").show(10, truncate=False)

is_overachiever.select("team", "is_overachiever").show(10, truncate=False)

performance_index.select("team", "performance_index").show(10, truncate=False)
```

SQL Scripts

— team_data.csv

1. Top Scorers per 90 Minutes

```
SELECT TOP 10
    team,
    ROUND(AVG(goals_per90 + assists_per90), 2) AS
goal_contribution_per90
FROM
    team_data
GROUP BY
    team
ORDER BY
    goal_contribution_per90 DESC;
```

2. Defensive Performance: Best Goalkeepers

```
SELECT TOP 10
    team,
    ROUND(AVG(gk_goals_against_per90), 2) AS avg_goals_against_per90,
    SUM(gk_clean_sheets) AS total_clean_sheets
FROM
    team_data
WHERE
    gk_games > 0
GROUP BY
    team
ORDER BY
    avg_goals_against_per90 ASC, total_clean_sheets DESC;
```

3. Shooting Accuracy

```
SELECT TOP 10
    team,
    ROUND(SUM(shots_on_target) * 100.0 / SUM(shots), 2) AS
shot_accuracy_percentage
FROM
    team_data
WHERE
    shots > 0
GROUP BY
    team
ORDER BY
    shot_accuracy_percentage DESC;
```

4. Passing Efficiency

```
SELECT TOP 10
    team,
    ROUND(SUM(passes_completed_short) * 100.0 / SUM(passes_short), 2)
AS short_pass_pct,
    ROUND(SUM(passes_completed_medium) * 100.0 / SUM(passes_medium),
2) AS medium_pass_pct,
    ROUND(SUM(passes_completed_long) * 100.0 / SUM(passes_long), 2) AS
long_pass_pct
FROM
    team_data
WHERE
    passes_short > 0 AND passes_medium > 0 AND passes_long > 0
GROUP BY
    team
ORDER BY
    medium_pass_pct DESC;
```

5. High-Pressure Tackles

```
SELECT TOP 10
```

```
    team,
    SUM(tackles_att_3rd) AS attacking_third_tackles
FROM
    team_data
GROUP BY
    team
ORDER BY
    attacking_third_tackles DESC;
```

6. Fouls Won by Players

```
SELECT TOP 10
    team,
    SUM(fouled) AS total_fouls_won
FROM
    team_data
GROUP BY
    team
ORDER BY
    total_fouls_won DESC;
```

7. Expected Goals (xG) Leaders

```
SELECT TOP 10
    team,
    ROUND(AVG(xg), 2) AS avg_xg
FROM
    team_data
GROUP BY
    team
ORDER BY
    avg_xg DESC;
```

8. Aerial Duel Success

```
SELECT TOP 10
```

```
    team,
    ROUND(SUM(aerials_won) * 100.0 / (SUM(aerials_won) +
SUM(aerials_lost)), 2) AS aerial_win_percentage
FROM
    team_data
WHERE
    aerials_won + aerials_lost > 0
GROUP BY
    team
ORDER BY
    aerial_win_percentage DESC;
```

9. Progressive Play

```
SELECT TOP 10
    team,
    ROUND(SUM(progressive_passes) * 100.0 / SUM(passes), 2) AS
progressive_pass_ratio
FROM
    team_data
WHERE
    passes > 0
GROUP BY
    team
ORDER BY
    progressive_pass_ratio DESC;
```

10. Players with Most Minutes

```
SELECT TOP 10
    team,
    ROUND(AVG(minutes_per_game), 2) AS avg_minutes_per_game
FROM
    team_data
WHERE
    games > 0
GROUP BY
```

```
    team
ORDER BY
    avg_minutes_per_game DESC;
```

— Group_stats.csv

1. Top 5 Teams by Points

sql
CopyEdit

```
SELECT TOP 5 team, points, wins, draws, losses
FROM group_stats
ORDER BY points DESC;
```

2. Top 5 Teams by Goal Difference

sql
CopyEdit

```
SELECT TOP 5 team, goal_difference, goals_scored, goals_against
FROM group_stats
ORDER BY goal_difference DESC;
```

3. Teams with Win Percentage Above 70%

sql
CopyEdit

```
SELECT team, wins, matches_played,
       CAST(wins * 100.0 / matches_played AS DECIMAL(5, 2)) AS
win_percentage
FROM group_stats
WHERE (wins * 100.0 / matches_played) > 70
ORDER BY win_percentage DESC;
```

4. Teams with Highest Goal Scored per Match

sql
CopyEdit

```
SELECT TOP 5 team, goals_scored, matches_played,
       CAST(goals_scored * 1.0 / matches_played AS DECIMAL(5, 2)) AS
avg_goals_per_match
FROM group_stats
```

```
ORDER BY avg_goals_per_match DESC;
```

5. Overachieving Teams (Points Exceeding Expected Goals Difference)

sql
CopyEdit

```
SELECT team, points, exp_goal_difference,
       (points - exp_goal_difference) AS overachievement
FROM group_stats
WHERE points > exp_goal_difference
ORDER BY overachievement DESC;
```

6. Defensive Efficiency (Teams Conceding Fewer Goals Than Expected)

sql
CopyEdit

```
SELECT team, goals_against, exp_goal_conceded,
       CAST(exp_goal_conceded * 1.0 / goals_against AS DECIMAL(5, 2))
AS defensive_efficiency
FROM group_stats
WHERE goals_against > 0
ORDER BY defensive_efficiency DESC;
```

7. Teams with Best Performance Index

sql
CopyEdit

```
SELECT TOP 5 team, points, goal_difference,
       CAST((points + goal_difference + (wins * 100.0 /
matches_played)) / 3 AS DECIMAL(5, 2)) AS performance_index
FROM group_stats
ORDER BY performance_index DESC;
```

8. Teams with Negative Goal Difference

sql
CopyEdit

```
SELECT team, goal_difference, goals_scored, goals_against
FROM group_stats
WHERE goal_difference < 0
```

```
ORDER BY goal_difference ASC;
```

9. Teams with Most Draws

sql
CopyEdit

```
SELECT TOP 5 team, draws, matches_played
FROM group_stats
ORDER BY draws DESC;
```

10. Teams with Lowest Expected Goal Conceded

sql
CopyEdit

```
SELECT TOP 5 team, exp_goal_conceded
FROM group_stats
ORDER BY exp_goal_conceded ASC;
```