

Documentation du projet Data Viz France

Introduction au projet

L'objectif de ce projet est de poursuivre le travail effectué lors de mon mémoire de M1 : Analyse comparative de 3 facteurs de performance dans le football : l'impact du 1er but, la distribution temporelle des buts et l'influence de l'avantage du terrain sur le match (domicile/extérieur) entre les équipes de jeunes (U17N et U19N) et le monde professionnel (Ligue 1).

Afin d'étendre cette analyse, on comparera cette fois ci les compétition suivantes, et ceux des saisons 2021/2022 à 2024/2025 (lorsque cela est possible) :

- **Ligue 1**
- **Ligue 2**
- **National 1**
- **National 2**
- **Championnat U19N**
- **D1 Féminine**
- **D2 Féminine**

Pour rappel, on s'occupera des facteurs suivants :

- **l'influence du 1er but sur le match**
- **la distribution temporelles des buts**
- **l'influence du paramètre domicile/extérieur**
- **l'avantage du terrain x 1er but**
- **le nombre de buts par match**

Ce travail s'articulera est plusieurs étapes avec :

- **la récupération des données via l'utilisation du web scrapping auprès du site Sofa Score**
- **le stockage de ces données dans des tables via l'utilisation de Supabase**
- **l'analyse des données collectées**
- **la mise en page de l'application web**
- **le déploiement de cette application.**

Web scraping / Stockage des données sur Supabase

Import des librairies

On va ensuite importer des librairies qui seront utile pour plus tard tels que pandas, numpy ou encore BeautifulSoup. À noter qu'il faudra appliquer la ligne suivante au sein de votre terminal afin d'installer toutes les dépendances nécessaires au bon fonctionnement du projet :

```
pip install -r requirements.txt
```

Liaison avec la base de données Supabase

On se servira de variables d'environnement afin de stocker nos données personnels (url du projet et la clé api associé) pour accéder au projet sur Supabase.

Création des requêtes permettant la création des tables sur Supabase (à faire sur Supabase)

Veuillez créer également ces tables au sein de votre projet directement sur Supabase. On ajoutera les données plus tard dans le projet.

- **Création de la table Compétition**

```
CREATE TABLE Competition (id_competition SERIAL PRIMARY KEY,competition_name VARCHAR(255) NOT NULL,country_name VARCHAR(255) NOT NULL,link_url TEXT NOT NULL);
```

- **Création de la table Saison**

```
CREATE TABLE Season (id_season SERIAL PRIMARY KEY,season_name VARCHAR(255) NOT NULL,id_competition INT NOT NULL,link_url TEXT NOT NULL,CONSTRAINT fk_competition FOREIGN KEY (id_competition) REFERENCES Competition(id_competition) ON DELETE CASCADE);
```

- **Création de la table Équipe**

```
CREATE TABLE team (id_team INT PRIMARY KEY,team_name TEXT NOT NULL);
```

- **Création de la table des informations des matchs**

```
CREATE TABLE info_match (id_match INT PRIMARY KEY,id_season INT NOT NULL,id_home_team INT NOT NULL,id_away_team INT NOT NULL,match_date DATE NOT NULL,link_url TEXT,FOREIGN KEY (id_home_team) REFERENCES team(id_team),FOREIGN KEY (id_away_team) REFERENCES team(id_team));
```

- **Création de la table des informations des buts sur les matchs**

```
CREATE TABLE Info_goal (id_match INT NOT NULL,score_home INT,score_away INT,result INT,home_0_15 INT,away_0_15 INT,home_16_30 INT,away_16_30 INT,home_31_45 INT,away_31_45 INT,home_46_60 INT,away_46_60 INT,home_61_75 INT,away_61_75 INT,home_76_90 INT,away_76_90 INT,PRIMARY KEY (id_match),FOREIGN KEY (id_match) REFERENCES info_match(id_match));
```

Stocker les informations des compétitions française

Ensuite, on va créer la classe Compétition, et sa fonction permettant l'insertion de ces données dans la table sur Supabase associé (et créé précédemment)

Pour récupérer les données de compétitions, on va effectuer du web scrapping auprès du site Sofa Score et le lien ci dessous :

- <https://www.sofascore.com/fr/football/france>.

On va s'infiltrer dans la page source afin de récupérer les informations de chaque compétition via la librairie BeautifulSoup. Après avoir ciblé la classe contenant ces informations, on stockera les données suivantes :

- l'identifiant de la compétition
- le nom de la compétition
- le nom du pays de la compétition
- le lien url de la compétition.

Par la suite, ces dernières seront stocker dans un dataframe, puis insérer dans la table associé, via la fonction insert_competition créé précédemment.

Stocker les informations des saisons française

Par la suite, avec l'aide de la librairie psycopg2, on va accéder à la table précédemment créé, dans le but de stocker les informations des saisons disponible via la table des compétitions. On utilisera la librairie conn pour effectuer la requête suivante : SELECT id_competition, link_url FROM competition. Cette requête nous ressort ainsi les identifiants et les liens urls de chacune des compétitions françaises disponibles sur le site Sofascore.

Dans la même logique que la section précédente, on va créer la classe Season et sa fonction associé. Cette classe contiendra les informations suivantes :

- l'identifiant de la saison
- le nom de la saison
- l'identifiant de la compétition
- le lien url de la saison.

On va ensuite initialiser un driver pour effectuer notre web scraping à partir de la requête, précédemment créé. À noter que l'on écartera les compétitions suivantes : Coupe de France, Trophée des Champions, Coupe de France Féminine, afin de se focaliser sur les compétitions n'étant pas à élimination directe, et s'établissant plus sur le long-terme. Dans une volonté d'analyser la tendance récente, on ne s'occupera uniquement des 4 dernières saisons française (de 2021/2022 à 2024/2025).

Pour résumer les principales étapes du stockage de ces informations, on va logiquement dans un 1er temps se connecter au lien de la compétition présent dans la table associé. On va ensuite fermé la page de cookie, pouvant bloquer la collecte de données. Pour information, sur ce site existe un menu déroulant afin de sélectionner la saison de notre choix. On s'en servira (dans la mesure du possible) pour cliquer sur les saisons suivantes :

- 2021/2022
- 2022/2023
- 2023/2024
- 2024/2025 (ou 2024/25).

Les informations essentielles figurant en 1ere page de ces saisons, on les collectera dans un objet, avant de passer à la saison suivante en effectuant la même opération. Les informations seront ensuite mis sous le format dataframe, et stocker dans la table season via la fonction insert_seasons. On fermara le driver une fois la tâche effectué.

Recherche des informations de chaque match et des équipes associés

Pour cette étape, on va récolter les informations de chaque match, et des équipes impliquées à partir des saisons stockées précédemment. On va ré-utiliser la même logique qu'auparavant :

- **Accession à la table season stockée sur Supabase**
- **Récupération des données à partir de la requête suivante : SELECT id_season, link_url FROM season, donnant l'identifiant de la saison et son lien url**
- **Création d'une classe Team, contenant l'identifiant de l'équipe, ainsi que son nom**

- Création d'une fonction insert_team
- Création d'une classe Match, contenant l'identifiant du match, saison, équipe à domicile et extérieur, la date du match, et enfin son lien url
- Création d'une fonction insert_matches

Étant donné la longueur plus conséquente de cette étape dans le but final de collecter les données sur les équipes et le match associé, on divisera chaque tâches par fonction. À noter que la logique reste semblable aux sections précédentes :

- Ouvrir un driver pour notre web scraping
- Fermeture de la page de cookies lorsque cela est nécessaire
- Récupération de la page d'une saison à partir de la requête créé précédemment
- Stockage des informations suivantes pour chaque match de la journée en cours dans un objet (en ne prenant pas en compte les matchs reportés, abandonné, ou donnant lieu à un tapis vert
- Appuyer sur la touche permettant d'accéder à la journée précédente lorsque tous les matchs de la journée courante ont été collecté
- Non prise en compte des saisons passées déjà collecté dans la recherche des données à insérer (permet d'accélérer la collecte des données)
- Stockage de nouveau des informations de ces matchs, et équipes, jusqu'à la 1ère journée, pour enfin passer à la saison suivante
- Mise des données au format dataframe
- Utilisation de la fonction insert_teams et insert_matches pour stocker tout cela sur notre projet Supabase
- Fermeture du driver une fois toutes les tâches effectuées.

Récupération des informations sur les buts de chaque match

Enfin, pour cette étape, on va récolter les informations de but de chaque match stockées précédemment. On va ré-utiliser la même logique qu'auparavant :

- Accession à la table info_match stockée sur Supabase
- Récupération des données à partir de la requête suivante : `SELECT id_match, link_url, id_season FROM info_match`, donnant l'identifiant du match, de sa saison et son lien url
- Récupération de l'information sur tous les identifiants des saisons et leur nom à partir de la requête suivante : `SELECT DISTINCT s.id_season, s.season_name FROM Season s JOIN info_match im ON s.id_season = im.id_season`; Cela permettra de demander à l'utilisateur les saisons qu'il souhaite stocker
- Récupération des identifiants de matchs déjà présent dans la base de données de buts, cela permettant de ne pas récupérer la même information une nouvelle fois lorsque qu'elle a déjà été stocké précédemment
- Création d'une classe Goal, contenant l'identifiant du match, le score de l'équipe à domicile et à l'extérieur, le résultat du match, le nombre de buts inscrit par chaque équipe par tranche de 15 minutes et l'influence du 1er but sur le match
- Création d'une fonction insert_goals

Encore une fois, étant donné la longueur plus conséquente de cette étape dans le but final de collecter les données sur les buts pour chacun des matchs, on divisera chaque tâches par fonction. Il est important de souligner que l'on demande à l'utilisateur les saison qu'il souhaite collecter dans une volonté réduire les chances que cela plante (dans le cas où trop de données sont à stocker). À noter que la logique reste semblable aux sections précédentes :

- Initialisation d'un driver pour notre web scraping. À noter que les visualisations superflues seront enlevées afin d'accélérer la collecte des informations
- Fermeture de la page de cookies lorsque cela est nécessaire
- Récupération de la page d'une match à partir de la requête créé précédemment (sans prendre en compte les matchs déjà collectés)
- Accession à la section incidents contenant les faits marquants du matchs, dont sur chaque but du match et le score final
- Extraction du score du match via les colonnes homeScore et awayScore
- Déduction du résultat du match en fonction de la fonction précédente (Victoire à domicile, Nul ou Victoire à l'extérieur)
- Extraction des informations de buts par intervalles grâce aux données incidents. Toutes les colonnes seront égales à 0 si le score est nul et vierge
- Déduction de l'influence du 1er but sur le match à partir de la fonction de result, et de l'information sur l'influence du 1er but
- Mise de toutes ces données au format dataframe
 - Non prise en compte des saisons passées déjà collecté, et des matchs déjà collectés dans la recherche des données à insérer (permet d'accélérer la collecte des données)
- Réinitialisation du driver tous les 10 matchs, afin de réduire les chances que le code plante
- Utilisation de la fonction insert_goals pour stocker tout cela sur notre projet Supabase
- Fermeture du driver une fois toutes les tâches effectuées.

Liste des pages et sections

On aura 4 pages distinctes avec ses requêtes associées :

- l'analyse d'une équipe
- l'analyse des confrontations entre deux équipes
- l'analyse d'une saison

- l'analyse d'une compétition

De plus, chaque page contiendra ces 5 sections :

- les statistiques générales de buts (nombre de buts inscrits et concédés, fréquence de score ect..)
- l'influence du 1er but sur le match
- la distribution temporelles des buts inscrits et concédés(par intervalle de 15 min ou 45min)
- l'influence du paramètre domicile/extérieur
- comparaison avec les autres saisons/compétitions/confrontations

Par ailleurs, des requêtes seront réalisés afin de constater quels sont les meilleurs équipes à travers ces indicateurs sur les années récentes.

Arborescence du projet pour la mise en place de l'application

Application |—— pages (Analyse_saison.py, Analyse_confrontations_équipe.py, Analyse_saison.py, Analyse_competition.py) |—— Image |—— CV |—— Mémoire |——
Documentation |—— Accueil.py : Main |—— README.md |—— requirements.txt

Mise en place de l'application

Accueil

Cette page sert à afficher les objectifs de ce projet, accompagné de divers ressources afin d'explorer en profondeur ce dernier (Mémoire, Code, Documentation), et en savoir plus sur moi-même (CV en Anglais et en Français)

Section analyse

En tête du projet

Logiquement, on commence chacun des fichiers réservés à l'analyse des données par l'import des librairies (streamlit, pandas, matplotlib, plotly, seaborn, supabase et python-dotenv). Ensuite, le titre de la page se doit d'être nécessairement placé au début du code, elle aura cette structure pour notre projet : `st.set_page_config(page_title="Data Viz 🌐 🇫🇷", page_icon="🇫🇷", layout="wide")`.

Par la suite, on accédera aux variables d'environnement du nom du projet et de l'anon key. Il s'agit des informations indispensables pour se connecter à la base de données sur Supabase en tant qu'utilisateur anonyme.

Initialisation des fonctions et création des pages

Via la fonction `supabase.rpc`, on va créer des fonctions pour chaque requête dans le but de récupérer les informations provenant de chaque procédure (préalablement créé sur Supabase). Pour cela, il faudra renseigner le nom de la procédure (exemple : `get_first_goal_season`) et la liste des paramètres à renseigner (exemple : `season_name`).

On créera également une fonction pour arrondir à deux chiffres après la virgule les valeurs si besoin, ou en entier selon cette dernières. Enfin des fonctions serviront à surligner les équipes/saisons/compétition choisi par l'utilisateur dans une couleur distinctive au sein d'un dataframe.

Affichage de l'application

Une barre latérale sera créé avec les pages disponibles dans le dossier pages, et une demande de sélection de l'équipe/saison/compétition du choix de l'utilisateur selon le type de page. À partir des choix effectuées par l'utilisateur, ce dernier disposera des 5 sections détaillées précédemment, dans lequel figure tous les graphiques ou tableaux associées. L'utilisateur cliquera sur la section de son choix pour accéder aux informations associées.

Chaque partie de code liée à l'affichage d'un graphique/tableaux/jauges dispose de la même logique.

- Dans un 1er temps, on récupérera les données via la fonction de la requête correspondant,
- Mise des données dans un dataframe en associant les données avec un nom de colonne associée,
- Affichage des graphiques/tableaux/jauges via les fonctions issues de plotly, dataframe. pie, ou encore pivot,
- Centrage et choix du titre, et traitement des valeurs numériques pour les afficher correctement (coloration de la colonne choisie si besoin).

Une boucle for sera créé afin de récupérer toutes les informations sur plusieurs saisons dans la section Comparaison entre les équipes/saisons/compétition. Par ailleurs, les données de maximum sur une saison ou une compétition seront stockées dans le cas d'un affichage des jauges. Enfin si l'on ne dispose de pas assez de données sur un aspect concernant une équipe, les graphiques associées ne seront pas affichés.

Annexe : Liste des requêtes/procédures SQL

- Information sur l'analyse d'une équipe -- Recherche des équipes create or replace function `get_teams()` returns setof text language sql as

```
select distinct team_name from team;
```

; grant execute on function `get_teams()` to anon; -- Recherche des saisons disponibles pour une équipe donnée create or replace function `get_seasons(team_name_input text)` returns setof text

language sql as

```
select season, ame from ( -- Sélection des saisons où l'équipe était à domicile
select season, ame from info_match
join season on info_match.id_season = s.id_season
join team on info_match.id_team = t.id_team
where t.ame = 'D' )
```

```
; grant execute on function get_seasons(text) to anon; -- Recherche des statistiques de buts pour une équipe donnée
create or replace function get_avg_goals_stats(season_name_input text)
returns table ( season_name text, team_name text, avg_goals_per_match numeric(10,2), avg_team_goals_per_match numeric(10,2), avg_team_goals_conceded_per_match numeric(10,2), avg_team_home_goals numeric(10,2), avg_team_away_goals numeric(10,2), avg_conceded_home_goals numeric(10,2), avg_conceded_away_goals numeric(10,2) )
language sql as
```

```
select -- Nom de la saison, season_ame, -- Nom de l'équipe, team_ame, -- Nombre de buts moyen par match
ROUND(avg((ig.score_home + ig.score_away) / 2), 2) as avg_goals_per_match, -- Nombre de
```

```
; grant execute on function get_avg_goals_stats(text) to anon; -- Recherche les buts inscrits d'une équipe donnée
create or replace function get_goals_scored(season_name_input text)
returns table ( team_name text, total_goals_scored numeric(10,2), avg_goals_scored numeric(10,2), goals_scored_home numeric(10,2), avg_goals_scored_home numeric(10,2), goals_scored_away numeric(10,2), avg_goals_scored_away numeric(10,2) )
language sql as
```

```
select -- Nom de l'équipe, team_ame, -- Total de buts marqués
coalesce(sum(case when is_home = 1 then score_away else score_home end), 0) as total_goals_scored, -- Moyenne de buts marqués
round(coal
```

```
; grant execute on function get_goals_scored(text) to anon; -- Recherche les buts concédés d'une équipe donnée
create or replace function get_goals_conceded(season_name_input text)
returns table ( team_name text, total_goals_conceded numeric(10,2), avg_goals_conceded numeric(10,2), goals_conceded_home numeric(10,2), avg_goals_conceded_home numeric(10,2), goals_conceded_away numeric(10,2), avg_goals_conceded_away numeric(10,2) )
language sql as
```

```
SELECT -- Nom de l'équipe, team_ame, -- Total des buts encaissés
SUM(CASE WHEN is_home = 1 THEN score_away ELSE 0 END) + SUM(CASE WHEN is_home = 0 THEN score_home ELSE 0 END) as total
```

```
; grant execute on function get_goals_conceded(text) to anon; -- Recherche de la fréquence des scores d'une équipe donnée
create or replace function get_frequent_score(team_name_input text, season_name_input text)
returns table ( score_home int, score_away int, percentage numeric(5,2) )
language sql as
```

```
WITH score_counts AS (SELECT -- Nom de l'équipe, team_ame, -- Nom de la saison, season_ame, -- Nombre de buts à domicile
ig.score_home, -- Nombre de buts à l'extérieur
ig.score_away, -- F
```

```
; grant execute on function get_frequent_score(text, text) to anon; -- Recherche des informations de 1er but inscrit ou encaissé d'une équipe donnée
create or replace function get_first_goal_season(season_name_input text)
returns table ( season_name text, team_name text, proportion_1st_goal_for numeric(10,2), proportion_no_goal numeric(10,2), proportion_1st_goal_against numeric(10,2), proportion_1st_goal_home_for numeric(10,2), proportion_no_goal_home numeric(10,2), proportion_1st_goal_home_against numeric(10,2), proportion_1st_goal_away_for numeric(10,2), proportion_no_goal_away numeric(10,2), proportion_1st_goal_away_against numeric(10,2), first_goal_win numeric(10,2), first_goal_draw numeric(10,2), first_goal_lose numeric(10,2), proportion_1st_goal_home_win numeric(10,2), proportion_1st_goal_home_draw numeric(10,2), proportion_1st_goal_home_lose numeric(10,2), proportion_1st_goal_away_win numeric(10,2), proportion_1st_goal_away_draw numeric(10,2), proportion_1st_goal_away_lose numeric(10,2), first_goal_conceded_win numeric(10,2), first_goal_conceded_draw numeric(10,2), first_goal_conceded_lose numeric(10,2), proportion_1st_goal_conceded_home_win numeric(10,2), proportion_1st_goal_conceded_home_draw numeric(10,2), proportion_1st_goal_conceded_home_lose numeric(10,2), proportion_1st_goal_conceded_away_win numeric(10,2), proportion_1st_goal_conceded_away_draw numeric(10,2), proportion_1st_goal_conceded_away_lose numeric(10,2) )
language sql as
```

```
SELECT -- Nom de la saison, season_ame, -- Nom de l'équipe, team_ame, -- Proportion que l'équipe a marqué le 1er but
ROUND((COUNT(CASE WHEN squad_1st_goal = 1 AND is_home = 1 THEN 1 END) /
```

```
; grant execute on function get_first_goal_season(text) to anon; -- Recherche des informations de la distribution des buts d'une équipe donnée
create or replace function get_distribution_goals_season(season_name_input text)
returns table ( season_name text, team_name text, proportion_buts_inscrit_1ere_periode numeric(10,2), proportion_buts_inscrit_2nde_periode numeric(10,2), proportion_buts_0_15 numeric(10,2), proportion_buts_16_30 numeric(10,2), proportion_buts_31_45 numeric(10,2), proportion_buts_46_60 numeric(10,2), proportion_buts_61_75 numeric(10,2), proportion_buts_76_90 numeric(10,2), proportion_buts_encaissés_1ere_periode numeric(10,2), proportion_buts_encaissés_2nde_periode numeric(10,2), proportion_buts_encaissés_0_15 numeric(10,2), proportion_buts_encaissés_16_30 numeric(10,2), proportion_buts_encaissés_31_45 numeric(10,2), proportion_buts_encaissés_46_60 numeric(10,2), proportion_buts_encaissés_61_75 numeric(10,2), proportion_buts_encaissés_76_90 numeric(10,2), buts_inscrit_1ere_periode numeric(10,2), buts_inscrit_2nde_periode numeric(10,2), nbr_buts_0_15 numeric(10,2), nbr_buts_16_30 numeric(10,2), nbr_buts_31_45 numeric(10,2), nbr_buts_46_60 numeric(10,2), nbr_buts_61_75 numeric(10,2), nbr_buts_76_90 numeric(10,2), buts_encaissés_1ere_periode numeric(10,2), buts_encaissés_2nde_periode numeric(10,2), buts_encaissés_0_15 numeric(10,2), buts_encaissés_16_30 numeric(10,2), buts_encaissés_31_45 numeric(10,2), buts_encaissés_46_60 numeric(10,2), buts_encaissés_61_75 numeric(10,2), buts_encaissés_76_90 numeric(10,2) )
```

LANGUAGE SQL AS

```
NULLIF(SU

NULLIF(SU

ROUND(SUM(CASE WHEN is_home = 1 '
SUM(C

ROUND(SUM(CASE WHEN is_home = 1 T
SUM(C

ROUND(SUM(CASE WHEN is_home = 1 T
SUM(C

ROUND(SUM(CASE WHEN is_home = 1 T
SUM(C

ROUND(SUM(CASE WHEN is_home = 1 T
SUM(C

ROUND(SUM(CASE WHEN is_home = 1 T
SUM(C

ROUND(SUM(CASE WHEN is_home = 1 THEN away_0_15 + away_16_30 + away_31_45

ROUND(SUM(CASE WHEN is_home = 1 THEN away_46_60 + away_61_75 + away_76_90

ROUND(SUM(CASE WHEN is_home = 1 '
SUM(CASE

ROUND(SUM(CASE WHEN is_home = 1 THEN away_16_30 ELSE home_16_30 END) * 100.0 / NULLIF(SUM(CASE WHEN is_home = 1 THEN away_0_15 ELSE home_0_15 END) + 5

ROUND(SUM(CASE WHEN is_home = 1 T
SUM(CASE

ROUND(SUM(CASE WHEN is_home = 1 T
SUM(CASE

ROUND(SUM(CASE WHEN is_home = 1 T
SUM(CASE

ROUND(SUM(CASE WHEN is_home = 1 T
SUM(CASE
```

```
; grant execute on function get_distribution_goals_season(text) to anon; -- Recherche des informations sur le facteur Domicile/Extérieur d'une équipe donnée CREATE OR REPLACE
FUNCTION get_rank_season( season_name_input TEXT ) RETURNS TABLE ( type text, season_name text, team_name text, matches numeric(10,2), wins numeric(10,2), draws
numeric(10,2), losses numeric(10,2), points numeric(10,2), avg_points numeric(10,2), home_advantage numeric(10,2) ) LANGUAGE SQL AS
```

```
WITH home_stats AS (SELECT -- Nom de la saison. season_name, -- Nom de l'équipe. team_name, -- Nombre de matchs joués à domicile COUNT(im.id_match) AS home_matches, -- Nombre de victoires
```

```
; grant execute on function get_rank_season(text) to anon; - Information sur l'analyse entre deux équipes -- Recherche des matchs entre deux équipes create or replace function
get_teams_in_season(season_name_input text) RETURNS TABLE ( team_name TEXT ) LANGUAGE SQL AS
```

```
SELECT t.team_name AS team_name FROM info_match JOIN seasons ON Nim.id_season = s.id_season JOIN team ON Nim.id_team = t.id_team OR im.id_away_team = t.id_team WHERE s.season_name = seas
```

```
; grant execute on function get_teams_in_season(text) to anon; - Recherche des matchs entre deux équipes create or replace function get_matches_between_teams(selected_team_home_input
text, selected_team_away_input text) RETURNS TABLE ( season_name TEXT, home_team_name TEXT, away_team_name TEXT, score_home numeric(10,2), score_away
numeric(10,2), match_date DATE ) LANGUAGE SQL AS
```

```
WITH match_details AS (SELECT -- Nom de la saison. season_name AS season_name, -- Nom de l'équipe évoluant à domicile l.team_name AS home_team_name, -- Nom de l'équipe évoluant à l'extérieur t
```

```
; grant execute on function get_matches_between_teams(text, text) to anon; -- Recherche des informations de buts entre deux équipes create or replace function
get_avg_goals_stats_between_teams(selected_team_home_input text, selected_team_away_input text) RETURNS TABLE ( avg_goals_selected_home numeric(10,2),
avg_goals_selected_away numeric(10,2), avg_goals_home_at_home numeric(10,2), avg_goals_away_at_away numeric(10,2) ) LANGUAGE SQL AS
```

```
SELECT -- Moyenne des buts marqués par sélectionné team_name AS team_name, -- Proportion de 1er but inscrit par l'équipe sélectionnée (domicile + extérieur) ROUND(AVG(CASE WHEN t.team_name = selected_team_home_input AND t.team_name = se
```

```
; grant execute on function get_avg_goals_stats_between_teams(text, text) to anon; -- Recherche des informations de 1er but entre deux équipes create or replace function
get_1st_goal_stats_between_teams(selected_team_home_input text, selected_team_away_input text) RETURNS TABLE ( team TEXT, proportion_1st_goal_for numeric(10,2),
proportion_no_goal numeric(10,2), proportion_1st_goal_against numeric(10,2), proportion_1st_goal_win numeric(10,2), proportion_1st_goal_draw numeric(10,2),
proportion_1st_goal_lose numeric(10,2), proportion_1st_goal_conceded_win numeric(10,2), proportion_1st_goal_conceded_draw numeric(10,2), proportion_1st_goal_conceded_lose
numeric(10,2) ) LANGUAGE SQL AS
```

```
SELECT -- Nom de l'équipe t.team_name AS team_name, -- Proportion de 1er but inscrit par l'équipe sélectionnée (domicile + extérieur) ROUND(COUNT(*) FILTER(WHERE (ig.squad_1st_goal = 1 AND in
```

```
; grant execute on function get_1st_goal_stats_between_teams(text, text) to anon; -- Recherche des informations de distribution de buts entre deux équipes create or replace function
get_distrib_goal_between_teams(selected_team_home_input text, selected_team_away_input text) RETURNS TABLE ( team TEXT, proportion_0_45 numeric(10,2), proportion_46_90
numeric(10,2), proportion_0_15 numeric(10,2), proportion_16_30 numeric(10,2), proportion_31_45 numeric(10,2), proportion_46_60 numeric(10,2), proportion_61_75
```


NUMERIC(10,2), proportion_76_90 NUMERIC(10,2)) LANGUAGE SQL AS

```

        WITH data_team AS (
            SELECT
                -- Nom de l'équipe
                th.team_name AS team,
                -- Type de match (Domicile/Extérieur)
                'home' AS match_type,
                -- Nombre de buts inscrits à domicile durant les 15 premières minutes
                SUM(ig.home_0_15) AS goals_0_15,
                -- Nombre de buts inscrits à domicile entre la 15ème et 30ème minutes
                SUM(ig.home_16_30) AS goals_16_30,
                -- Nombre de buts inscrits à domicile entre la 31ème et 45ème minutes
                SUM(ig.home_31_45) AS goals_31_45,
                -- Nombre de buts inscrits à domicile entre la 46ème et 60ème minutes
                SUM(ig.home_46_60) AS goals_46_60,
                -- Nombre de buts inscrits à domicile entre la 61ème et 75ème minutes
                SUM(ig.home_61_75) AS goals_61_75,
                -- Nombre de buts inscrits à domicile entre la 76ème et 90ème minutes
                SUM(ig.home_76_90) AS goals_76_90,
                -- Nombre de buts inscrits à domicile durant la 1ère période
                SUM(ig.home_0_15 + ig.home_16_30 + ig.home_31_45) AS first_period_goals,
                -- Nombre de buts inscrits à domicile durant la 2nd période
                SUM(ig.home_46_60 + ig.home_61_75 + ig.home_76_90) AS second_period_goals,
                -- Nombre de buts inscrits à domicile au total
                SUM(ig.home_0_15 + ig.home_16_30 + ig.home_31_45 + ig.home_46_60 + ig.home_61_75 + ig.home_76_90) AS total_goals
            FROM info_match im
            JOIN team th ON im.id_home_team = th.id_team
            JOIN team ta ON im.id_away_team = ta.id_team
            JOIN info_goal ig ON im.id_match = ig.id_match
        WHERE (th.team_name = selected_team_home_input AND ta.team_name = selected_team_away_input) OR (th.team_name = selected_team_away_input AND ta.team_name = selected_team_home_input)
        GROUP BY th.team_name

        UNION ALL

        SELECT
            -- Nom de l'équipe
            ta.team_name AS team,
            -- Type de match (Domicile/Extérieur)
            'away' AS match_type,
            -- Nombre de buts inscrits à l'extérieur durant les 15 premières minutes
            SUM(ig.away_0_15) AS goals_0_15,
            -- Nombre de buts inscrits à l'extérieur entre la 15ème et 30ème minutes
            SUM(ig.away_16_30) AS goals_16_30,
            -- Nombre de buts inscrits à l'extérieur entre la 31ème et 45ème minutes
            SUM(ig.away_31_45) AS goals_31_45,
            -- Nombre de buts inscrits à l'extérieur entre la 46ème et 60ème minutes
            SUM(ig.away_46_60) AS goals_46_60,
            -- Nombre de buts inscrits à l'extérieur entre la 61ème et 75ème minutes
            SUM(ig.away_61_75) AS goals_61_75,
            -- Nombre de buts inscrits à l'extérieur entre la 76ème et 90ème minutes
            SUM(ig.away_76_90) AS goals_76_90,
            -- Nombre de buts inscrits à l'extérieur durant la 1ère période
            SUM(ig.away_0_15 + ig.away_16_30 + ig.away_31_45) AS first_period_goals,
            -- Nombre de buts inscrits à l'extérieur durant la 2nd période
            SUM(ig.away_46_60 + ig.away_61_75 + ig.away_76_90) AS second_period_goals,
            -- Nombre de buts inscrits à l'extérieur au total
            SUM(ig.away_0_15 + ig.away_16_30 + ig.away_31_45 + ig.away_46_60 + ig.away_61_75 + ig.away_76_90) AS total_goals
        FROM info_match im
        JOIN team th ON im.id_home_team = th.id_team
        JOIN team ta ON im.id_away_team = ta.id_team
        JOIN info_goal ig ON im.id_match = ig.id_match
        WHERE (th.team_name = selected_team_home_input AND ta.team_name = selected_team_away_input) OR (th.team_name = selected_team_away_input AND ta.team_name = selected_team_home_input)
        GROUP BY ta.team_name

    )
    SELECT
        -- Nom de l'équipe
        team,
        -- Nombre de buts inscrits en 1ère période
        ROUND(SUM(first_period_goals) * 100.0 / NULLIF(SUM(total_goals), 0), 2) AS proportion_0_45,
        -- Nombre de buts inscrits en 2nd période
        ROUND(SUM(second_period_goals) * 100.0 / NULLIF(SUM(total_goals), 0), 2) AS proportion_46_90,
        -- Nombre de buts inscrits durant les 15 premières minutes
        ROUND(SUM(goals_0_15) * 100.0 / NULLIF(SUM(total_goals), 0), 2) AS proportion_0_15,
        -- Nombre de buts inscrits entre la 16ème et la 30ème minute
        ROUND(SUM(goals_16_30) * 100.0 / NULLIF(SUM(total_goals), 0), 2) AS proportion_16_30,
        -- Nombre de buts inscrits entre la 31ème et la 45ème minute
        ROUND(SUM(goals_31_45) * 100.0 / NULLIF(SUM(total_goals), 0), 2) AS proportion_31_45,
        -- Nombre de buts inscrits entre la 46ème minute et la 60ème minute
        ROUND(SUM(goals_46_60) * 100.0 / NULLIF(SUM(total_goals), 0), 2) AS proportion_46_60,
        -- Nombre de buts inscrits entre la 61ème et la 75ème minute
        ROUND(SUM(goals_61_75) * 100.0 / NULLIF(SUM(total_goals), 0), 2) AS proportion_61_75,
        -- Nombre de buts inscrits entre la 76ème et la 90ème minute
        ROUND(SUM(goals_76_90) * 100.0 / NULLIF(SUM(total_goals), 0), 2) AS proportion_76_90
    FROM data_team
    GROUP BY team;

```

; grant execute on function get_distrib_goal_between_teams(text, text) to anon; -- Recherche de l'influence du facteur Domicile/Extérieur entre les deux équipes create or replace function get_home_away_selected_teams(selected_team_home_input text, selected_team_away_input text) RETURNS TABLE (team_name TEXT, home_win NUMERIC(10,2), home_draws NUMERIC(10,2), home_losses NUMERIC(10,2), home_advantage NUMERIC(10,2), total_wins NUMERIC(10,2), total_draws NUMERIC(10,2), total_losses NUMERIC(10,2)) LANGUAGE SQL AS

WITH home_stats AS (SELECT -- Nom de l'équipe th.team_name, -- Nombre de matchs à domicile COUNT(im.id_match) AS home_matches, -- Nombre de victoires à domicile COUNT(CASE WHEN ig.res = 'home' THEN 1 ELSE 0 END) AS home_wins, -- Nombre de matchs à l'extérieur COUNT(im.id_match) AS away_matches, -- Nombre de victoires à l'extérieur COUNT(CASE WHEN ig.res = 'away' THEN 1 ELSE 0 END) AS away_wins, -- Nombre de matchs nuls à domicile COUNT(CASE WHEN ig.res = 'draw' THEN 1 ELSE 0 END) AS home_draws, -- Nombre de matchs nuls à l'extérieur COUNT(CASE WHEN ig.res = 'draw' THEN 1 ELSE 0 END) AS away_draws, -- Nombre de buts marqués à domicile SUM(ig.home_goals) AS home_goals, -- Nombre de buts marqués à l'extérieur SUM(ig.away_goals) AS away_goals, -- Nombre de buts encaissés à domicile SUM(ig.home_conceded) AS home_conceded, -- Nombre de buts encaissés à l'extérieur SUM(ig.away_conceded) AS away_conceded FROM info_match im JOIN info_goal ig ON im.id_match = ig.id_match WHERE (th.team_name = selected_team_home_input AND ta.team_name = selected_team_away_input) OR (th.team_name = selected_team_away_input AND ta.team_name = selected_team_home_input) GROUP BY th.team_name)

; grant execute on function get_home_away_selected_teams(text, text) to anon; - Information sur l'analyse d'une saison - Recherche des compétitions disponibles create or replace function get_competitions() returns setof text language sql as

SELECT DISTINCT competition, competition_name FROM competition JOIN season ON competition.id_competition = season.id_competition;

; grant execute on function get_competitions() to anon; -- Recherche des saisons disponibles pour une équipe donnée create or replace function

get_seasons_by_competition(competition_name_input text) returns set of text language sql as

```
SELECT DISTINCT season, season_name FROM season JOIN competition ON season.id_competition = competition.id_competition WHERE competition.competition_name = competition_name_input;
```

; grant execute on function get_seasons_by_competition(text) to anon; -- Recherche des informations donnée sur les buts marqués sur une saison en prenant en compte la compétition create or replace function get_avg_goals_stats_by_competition() returns table (competition_name text, season_name text, avg_goals_per_match numeric(10,2), avg_home_goals numeric(10,2), avg_away_goals numeric(10,2)) language sql as

```
SELECT -- Nom de la compétition, competition_name, -- Nom de la saison, season_name, -- Nombre de buts par match (SUM(ig.score_home) + SUM(ig.score_away)) * 1.0 / COUNT(im.id_match) AS
```

; grant execute on function get_avg_goals_stats_by_competition() to anon; -- Recherche de la fréquence des scores d'une équipe donnée CREATE OR REPLACE FUNCTION get_frequent_score_by_season(season_name_input TEXT) RETURNS TABLE (score_home INT, score_away INT, percentage NUMERIC(5,2)) LANGUAGE SQL AS

```
WITH score_counts AS (SELECT -- Nom de la compétition, competition_name, -- Nom de la saison, season_name, -- Score de l'équipe à domicile, score_home, -- Score de l'équipe à l'extérieur, ig
```

; grant execute on function get_frequent_score_by_season(text) to anon; -- Recherche des meilleures équipes en termes de buts inscrits create or replace function get_top5_goals_scored(competition_name_input text) returns table (team_name text, season_name text, total_goals_scored numeric(10,2), avg_goals_scored numeric(10,2), goals_scored_home numeric(10,2), avg_goals_scored_home numeric(10,2), goals_scored_away numeric(10,2), avg_goals_scored_away numeric(10,2)) language sql as

```
WITH team_avg_goals AS (SELECT -- Nom de l'équipe, team_name, -- Nom de la saison, season_name, -- Nombre de buts inscrits sur la saison, SUM(CASE WHEN is_home = 1 THEN score_home ELSE 0 END)
```

; grant execute on function get_top5_goals_scored(text) to anon; -- Recherche des équipes ayant encaissé le moins de but CREATE OR REPLACE FUNCTION get_top5_goals_conceded(competition_name_input TEXT) RETURNS TABLE (team_name TEXT, season_name text, total_goals_conceded NUMERIC(10,2), avg_goals_conceded NUMERIC(10,2), goals_conceded_home NUMERIC(10,2), avg_goals_conceded_home NUMERIC(10,2), goals_conceded_away NUMERIC(10,2), avg_goals_conceded_away NUMERIC(10,2)) LANGUAGE SQL AS

```
WITH team_avg_goals AS (SELECT -- Nom de l'équipe, team_name, -- Nom de la saison, season_name, -- Nombre de buts concédés, SUM(CASE WHEN is_home = 1 THEN score_away ELSE 0 END) + SUM(C
```

; grant execute on function get_top5_goals_conceded(text) to anon; -- Recherche des informations de 1er but inscrit à l'échelle d'une saison CREATE OR REPLACE FUNCTION get_first_goal_stats(season_name_input TEXT) RETURNS TABLE (season_name text, proportion_no_goal NUMERIC(10,2), proportion_1st_goal_home NUMERIC(10,2), proportion_1st_goal_away NUMERIC(10,2), first_goal_win NUMERIC(10,2), first_goal_draw NUMERIC(10,2), first_goal_lose NUMERIC(10,2), first_goal_home_win NUMERIC(10,2), first_goal_home_draw NUMERIC(10,2), first_goal_home_lose NUMERIC(10,2), first_goal_away_win NUMERIC(10,2), first_goal_away_draw NUMERIC(10,2), first_goal_away_lose NUMERIC(10,2)) LANGUAGE SQL AS

```
SELECT -- Nom de la saison, season_name, -- Proportion des matchs où aucun but n'est inscrit, ROUND(COUNT(CASE WHEN squad_1st_goal = 0 THEN 1 END) * 100 / COUNT(squad_1st_goal), 2) AS prop
```

; grant execute on function get_first_goal_stats(text) to anon; -- Recherche des informations des meilleures équipes au niveau du 1er but inscrit CREATE OR REPLACE FUNCTION get_top_teams_first_goal(competition_name_input TEXT) RETURNS TABLE (season_name text, team_name text, proportion_1st_goal_for NUMERIC(10,2)) LANGUAGE SQL AS

```
SELECT -- Nom de la saison, season_name, -- Nom de l'équipe, team_name, -- Proportion que l'équipe marque le 1er but, ROUND((COUNT(CASE WHEN squad_1st_goal = 1 AND is_home = 1 THEN 1 EN
```

; grant execute on function get_top_teams_first_goal(text) to anon; -- Recherche des informations des meilleures équipes au niveau du 1er but inscrit vainqueur CREATE OR REPLACE FUNCTION get_top_teams_first_goal_win(competition_name_input TEXT) RETURNS TABLE (season_name text, team_name text, first_goal_win NUMERIC(10,2)) LANGUAGE SQL AS

```
SELECT -- Nom de la saison, season_name, -- Nom de l'équipe, team_name, -- Proportion des victoires lorsquel'équipe marque le 1er but, ROUND((COUNT(CASE WHEN squad_1st_goal = 1 AND result =
```

; grant execute on function get_top_teams_first_goal_win(text) to anon; -- Recherche des informations des meilleures équipes au niveau du 1er but concédés mais tout de même vainqueur CREATE OR REPLACE FUNCTION get_top_teams_first_goal_conceded_win(competition_name_input TEXT) RETURNS TABLE (season_name text, team_name text, first_goal_conceded_win NUMERIC(10,2)) LANGUAGE SQL AS

```
SELECT -- Nom de la saison, season_name, -- Nom de l'équipe, team_name, -- Proportion des victoires lorsquel'équipe encaisse le 1er but (engénéral, à domicile ou à l'extérieur), ROUND((COUNT(CAS
```

; grant execute on function get_top_teams_first_goal_conceded_win(text) to anon; -- Recherche des informations sur la distribution des buts à l'échelle d'une saison CREATE OR REPLACE FUNCTION get_distribution_goals(season_name_input TEXT) RETURNS TABLE (season_name text, proportion_buts_1ere_periode NUMERIC(10,2), proportion_buts_2nde_periode NUMERIC(10,2), proportion_buts_0_15 NUMERIC(10,2), proportion_buts_16_30 NUMERIC(10,2), proportion_buts_31_45 NUMERIC(10,2), proportion_buts_46_60 NUMERIC(10,2), proportion_buts_61_75 NUMERIC(10,2), proportion_buts_76_90 NUMERIC(10,2)) LANGUAGE SQL AS

```
ROUND((SUM(home_0_15) + SUM(away_0_15) + SUM(home_16_30) + SUM(away_16_30) + SUM(home_31_45) + SUM(away_31_45)) * 100.0 / NULLIF(SUM(home_0_15) + SUM(aw
```

```
ROUND((SUM(home_46_60) + SUM(away_46_60) + SUM(home_61_75) + SUM(away_61_75) + SUM(home_76_90) + SUM(away_76_90)) * 100.0 / NULLIF(SUM(home_0_15) + SUM(aw
```

```
ROUND((SUM(home_0_15) + SUM(away_0_15)) * 100.0 / NULLIF(SUM(home_0_15) + SUM(away_0_15) + SUM(home_16_30) + SUM
```

```
ROUND((SUM(home_16_30) + SUM(away_16_30)) * 100.0 / NULLIF(SUM(home_0_15) + SUM(away_0_15) + SUM(home_16_30) + SUM
```

```
ROUND((SUM(home_31_45) + SUM(away_31_45)) * 100.0 / NULLIF(SUM(home_0_15) + SUM(away_0_15) + SUM(home_16_30) + SUM
```

```
ROUND((SUM(home_46_60) + SUM(away_46_60)) * 100.0 / NULLIF(SUM(home_0_15) + SUM(away_0_15) + SUM(home_16_30) + SUM
```

```
ROUND((SUM(home_61_75) + SUM(away_61_75)) * 100.0 / NULLIF(SUM(home_0_15) + SUM(away_0_15) + SUM(home_16_30) + SUM
```

```
ROUND((SUM(home_76_90) + SUM(away_76_90)) * 100.0 / NULLIF(SUM(home_0_15) + SUM(away_0_15) + SUM(home_16_30) + SUM
```

; grant execute on function get_distribution_goals(text) to anon; -- Recherche des meilleures équipes en 1ère période au regard d'une compétition donnée CREATE OR REPLACE FUNCTION get_top_teams_1st_period(competition_name_input TEXT) RETURNS TABLE (season_name text, team_name text, proportion_buts_1ere_periode NUMERIC(10,2), nbr_buts_inscrit_1ere_periode NUMERIC(10,2), proportion_buts_0_15 NUMERIC(10,2), nbr_buts_0_15 NUMERIC(10,2), proportion_buts_16_30 NUMERIC(10,2), nbr_buts_16_30

```
NUMERIC(10,2), proportion_buts_31_45 NUMERIC(10,2), nbr_buts_31_45 NUMERIC(10,2) ) LANGUAGE SQL AS
```

```
SUM(CASE WHEN is_home = 1 THEN home_0_15 + home_16_30 + home_31_45 ELSE away_0
```

```
SUM(CASE WHEN is_home = 1 THEN home_0_15 ELSE away_0_15 END) * 100.0 / NULLIF(SUM(CASE WHEN is_home = 1 THEN home_0_15 ELSE away_0_15 END) + SUM(CASE
```

```
SUM(CASE WHEN is_home = 1 THEN home_16_30  
SUM(CASE WHE!
```

```
SUM(CASE WHEN is_home = 1 THEN home_31_45  
SUM(CASE WHE!
```

```
; grant execute on function get_top_teams_1st_period(text) to anon; -- Recherche des meilleurs équipes en 2nd période au regard d'une compétition donnée CREATE OR REPLACE  
FUNCTION get_top_teams_2nd_period(competition_name_input TEXT) RETURNS TABLE ( season_name text, team_name text, proportion_buts_inscrit_2nde_periode NUMERIC(10,2),  
nbr_buts_inscrit_2nde_periode NUMERIC(10,2), proportion_buts_46_60 NUMERIC(10,2), nbr_buts_46_60 NUMERIC(10,2), proportion_buts_61_75 NUMERIC(10,2), nbr_buts_61_75  
NUMERIC(10,2), proportion_buts_76_90 NUMERIC(10,2), nbr_buts_76_90 NUMERIC(10,2) ) LANGUAGE SQL AS
```

```
SUM(CASE WHEN is_home = 1 THEN home_46_60 + home_61_75 + home_76_90 ELSE away_46_60 + away_61_75 + away_76_90 END) * 100.0 / NULLIF(SUM(CASE WHEN is_home
```

```
SUM(CASE WHEN is_home = 1 THEN home_46_60 + hor
```

```
--  
SUM(CASE WHEN is_home = 1 THEN home_46_60 ELSE away_46_60 END) * 100.0 / NULLIF(SUM(CASE WHEN is_home = 1 THEN hor  
SUM(CASE WHEN is_home = 1 THEN home_46_60 ELSE away_46_60 END) + SUM(CASE WHEN is_hoi
```

```
--  
SUM(CASE WHEN i
```

```
--  
SUM(CASE WHEN is_home = 1 THEN home_61_75 ELSE away_61_75 END) * 100.0 / NULLIF(SUM(CASE WHEN is_home = 1 THEN hor  
SUM(CASE WHEN is_home = 1 THEN home_46_60 ELSE away_46_60 END) + SUM(CASE WHEN is_hoi
```

```
--  
SUM(CASE WHEN i
```

```
--  
SUM(CASE WHEN is_home = 1 THEN home_76_90 ELSE away_76_90 END) * 100.0 / NULLIF(SUM(CASE WHEN is_home = 1 THEN hor  
SUM(CASE WHEN is_home = 1 THEN home_46_60 ELSE away_46_60 END) + SUM(CASE WHEN is_hoi
```

```
--  
SUM(CASE WHEN is
```

```
SELECT t.tea
```

```
SELECT t.tea
```

```
; grant execute on function get_top_teams_2nd_period(text) to anon; -- Recherche des meilleurs équipes dans les 15 dernières minutes au regard d'une compétition donnée CREATE OR
REPLACE FUNCTION get_top_teams_last_minutes(competition_name_input TEXT) RETURNS TABLE ( season_name text, team_name text, proportion_buts_76_90 NUMERIC(10,2),
nbr_buts_76_90 NUMERIC(10,2) ) LANGUAGE SQL AS
```

```

SELECT
-- Nom de la saison
season_name,
-- Nom de l'équipe
team_name,
-- Proportion de buts inscrits entre la 76ème et la
SUM(CASE WHEN is_home = 1 THEN home_76_90 ELSE away_76_90 END) * 100.0 / NULLIF(SUM(CASE WHEN is_home = 1 THEN home_0_15 ELSE away_0_15 END) + SUM(CASE
SUM(CASE WHEN is_home = 1 THEN home_46_60 ELSE away_46_60 END) + SUM(CASE WHEN is_home = 1 THEN home_61_75 ELSE away_61_75
-- Nombre de buts inscrits entre la 76ème et la 9
SUM(CASE WHEN is_home = 1 THEN home_76_90 ELSE away_
FROM (
-- Données pour les équipes jouant à don
SELECT t.team_name, ig.*, 1 AS is_home, s.season_name, n
FROM info_match im
JOIN season s USING(id_season)
JOIN team t ON im.id_home_team = t.id
JOIN info_goal ig USING(id_match)
JOIN competition c USING(id_competiti
UNION ALL
-- Données pour les équipes jouant à l'ext
SELECT t.team_name, ig.*, 0 AS is_home, s.season_name, n
FROM info_match im
JOIN season s USING(id_season)
JOIN team t ON im.id_away_team = t.id
JOIN info_goal ig USING(id_match)
JOIN competition c USING(id_competi
) AS all_matches
WHERE competition_name = competition_nar
GROUP BY season_name, team_name
HAVING COUNT(*) >= 5
ORDER BY proportion_buts_76_90 DE
LIMIT 5;
```

```
; grant execute on function get_top_teams_last_minutes(text) to anon; -- Recherche des informations sur l'avantage du terrain à l'échelle d'une saison CREATE OR REPLACE FUNCTION
get_home_away_advantage() RETURNS TABLE ( season_name text, proportion_home_win NUMERIC(10,2), proportion_draw NUMERIC(10,2), proportion_away_win NUMERIC(10,2),
home_advantage NUMERIC(10,2) ) LANGUAGE SQL AS
```

```
SELECT -- Nom de la saison season_name, -- Proportion des victoires lorsquel'equipe évolue à domicile ROUND(COUNT(CASE WHEN is_home = 1 AND result = 1 THEN 1 END) * 100 / COUNT(CASE
```

```
; grant execute on function get_home_away_advantage() to anon; -- Recherche des informations sur le classement à domicile à l'échelle d'une saison CREATE OR REPLACE FUNCTION
get_rank_home_season(season_name_input TEXT) RETURNS TABLE ( team_name text, all_matches NUMERIC(10,2), number_home_win NUMERIC(10,2), number_home_draw
NUMERIC(10,2), number_home_lose NUMERIC(10,2), home_points NUMERIC(10,2), avg_home_points NUMERIC(10,2) ) LANGUAGE SQL AS
```

```
SELECT -- Nom de l'équipe team_name, -- Nombre de matchs COUNT(im.id_match) AS all_matches, -- Nombre de victoires à domicile COUNT(CASE WHEN ig.result = 1 THEN 1 END) AS number_won
```

```
; grant execute on function get_rank_home_season(text) to anon; -- Recherche des informations sur le classement à l'extérieur à l'échelle d'une saison CREATE OR REPLACE FUNCTION
get_rank_away_season(season_name_input TEXT) RETURNS TABLE ( team_name text, all_matches NUMERIC(10,2), number_away_win NUMERIC(10,2), number_away_draw
NUMERIC(10,2), number_away_lose NUMERIC(10,2), away_points NUMERIC(10,2), avg_away_points NUMERIC(10,2) ) LANGUAGE SQL AS
```

```
SELECT -- Nom de l'équipe team_name, -- Nombre de matchs COUNT(im.id_match) AS all_matches, -- Nombre de victoires à l'extérieur COUNT(CASE WHEN ig.result = 2 THEN 1 END) AS number_won
```

```
; grant execute on function get_rank_away_season(text) to anon; -- Recherche des 5 meilleurs équipes à domicile d'une compétition donné CREATE OR REPLACE FUNCTION
get_top5_home_rank_competition(competition_name_input TEXT) RETURNS TABLE ( season_name text, team_name text, all_matches NUMERIC(10,2), number_home_win
NUMERIC(10,2), number_home_draw NUMERIC(10,2), number_home_lose NUMERIC(10,2), home_points NUMERIC(10,2), avg_home_points NUMERIC(10,2) ) LANGUAGE SQL AS
```

```
SELECT -- Nom de la saison season_name, -- Nom de l'équipe team_name, -- Nombre de matchs COUNT(im.id_match) AS all_matches, -- Nombre de victoires à domicile COUNT(CASE WHEN ig
```

```
; grant execute on function get_top5_home_rank_competition(text) to anon; -- Recherche des meilleurs équipes à l'extérieur à l'échelle d'une saison CREATE OR REPLACE FUNCTION
get_top5_away_rank_competition(competition_name_input TEXT) RETURNS TABLE ( season_name text, team_name text, all_matches NUMERIC(10,2), number_away_win
NUMERIC(10,2), number_away_draw NUMERIC(10,2), number_away_lose NUMERIC(10,2), away_points NUMERIC(10,2), avg_away_points NUMERIC(10,2) ) LANGUAGE SQL AS
```

```
SELECT -- Nom de la saison season_name, -- Nom de l'équipe team_name, -- Nombre de matchs COUNT(im.id_match) AS all_matches, -- Nombre de victoires à l'extérieur COUNT(CASE WHEN i
```

```
; grant execute on function get_top5_away_rank_competition(text) to anon; - Information sur l'analyse d'une compétition -- Recherche des informations donnée sur les buts marqués sur une
saison en prenant en compte la compétition de manière générale create or replace function get_avg_goals_stats_by_competition_2() returns table ( competition_name text, avg_goals_per_match
numeric(10,2), avg_home_goals numeric(10,2), avg_away_goals numeric(10,2) ) language sql as
```

```
SELECT -- Nom de la compétition c.competition_name, -- Moyenne de buts par match ROUND((SUM(ig.score_home) + SUM(ig.score_away)) * 1.0 / COUNT(im.id_match), 2) AS avg_goals_per_match, --
```

```
; grant execute on function get_avg_goals_stats_by_competition_2() to anon; -- Recherche de la fréquence des scores d'une compétition donnée CREATE OR REPLACE FUNCTION
get_frequent_score_by_competition(competition_name_input TEXT) RETURNS TABLE ( score_home INT, score_away INT, percentage NUMERIC(5,2) ) LANGUAGE SQL AS
```

```
WITH score_counts AS (SELECT -- Nom de la compétition c.competition_name, -- Nombre de buts inscrits par l'équipe évoluant à domicile sur le match ig.score_home, -- Nombre de buts inscrits par l'équipe é
```

```
; grant execute on function get_frequent_score_by_competition(text) to anon; -- Recherche des informations de 1er but inscrit à l'échelle d'une saison CREATE OR REPLACE FUNCTION
get_first_goal_stats_by_competition(competition_name_input TEXT) RETURNS TABLE ( competition_name text, proportion_no_goal NUMERIC(10,2), proportion_1st_goal_home
NUMERIC(10,2), proportion_1st_goal_away NUMERIC(10,2), first_goal_win NUMERIC(10,2), first_goal_draw NUMERIC(10,2), first_goal_lose NUMERIC(10,2), first_goal_home_win
NUMERIC(10,2), first_goal_home_draw NUMERIC(10,2), first_goal_home_lose NUMERIC(10,2), first_goal_away_win NUMERIC(10,2), first_goal_away_draw NUMERIC(10,2),
first_goal_away_lose NUMERIC(10,2) ) LANGUAGE SQL AS
```

```
SELECT -- Nom de la compétition competition_name, -- Proportion des matchs où aucun but n'est inscrit ROUND(COUNT(CASE WHEN squad_1st_goal = 0 THEN 1 END) * 100 / COUNT(squad_1st_goal),
```

```
; grant execute on function get_first_goal_stats_by_competition(text) to anon; -- Recherche des informations sur la distribution des buts à l'échelle d'une compétition CREATE OR REPLACE
FUNCTION get_distribution_goals_by_competition(competition_name_input TEXT) RETURNS TABLE ( competition_name text, proportion_buts_1ere_période NUMERIC(10,2),
proportion_buts_2nde_période NUMERIC(10,2), proportion_buts_0_15 NUMERIC(10,2), proportion_buts_16_30 NUMERIC(10,2), proportion_buts_31_45 NUMERIC(10,2),
```

```
proportion_buts_46_60 NUMERIC(10,2), proportion_buts_61_75 NUMERIC(10,2), proportion_buts_76_90 NUMERIC(10,2) ) LANGUAGE SQL AS
```

```
ROUND((SUM(home_0_15) + SUM(away_0_15) + SUM(home_16_30) + SUM(away_16_30) + SUM(home_31_45) + SUM(away_31_45)) * 100.0 / NULLIF(SUM(home_0_15) + SUM(away_0_15) + SUM(home_16_30) + SUM(away_16_30) + SUM(home_31_45) + SUM(away_31_45)), 2) AS home_away_advantage,
ROUND((SUM(home_46_60) + SUM(away_46_60) + SUM(home_61_75) + SUM(away_61_75) + SUM(home_76_90) + SUM(away_76_90)) * 100.0 / NULLIF(SUM(home_0_15) + SUM(away_0_15) + SUM(home_16_30) + SUM(away_16_30) + SUM(home_31_45) + SUM(away_31_45)), 2) AS home_away_advantage,
ROUND((SUM(home_0_15) + SUM(away_0_15)) * 100.0 / NULLIF(SUM(home_0_15) + SUM(away_0_15) + SUM(home_16_30) + SUM(away_16_30) + SUM(home_31_45) + SUM(away_31_45)), 2) AS home_away_advantage,
ROUND((SUM(home_16_30) + SUM(away_16_30)) * 100.0 / NULLIF(SUM(home_0_15) + SUM(away_0_15) + SUM(home_16_30) + SUM(away_16_30) + SUM(home_31_45) + SUM(away_31_45)), 2) AS home_away_advantage,
ROUND((SUM(home_31_45) + SUM(away_31_45)) * 100.0 / NULLIF(SUM(home_0_15) + SUM(away_0_15) + SUM(home_16_30) + SUM(away_16_30) + SUM(home_31_45) + SUM(away_31_45)), 2) AS home_away_advantage,
ROUND((SUM(home_46_60) + SUM(away_46_60)) * 100.0 / NULLIF(SUM(home_0_15) + SUM(away_0_15) + SUM(home_16_30) + SUM(away_16_30) + SUM(home_31_45) + SUM(away_31_45)), 2) AS home_away_advantage,
ROUND((SUM(home_61_75) + SUM(away_61_75)) * 100.0 / NULLIF(SUM(home_0_15) + SUM(away_0_15) + SUM(home_16_30) + SUM(away_16_30) + SUM(home_31_45) + SUM(away_31_45)), 2) AS home_away_advantage,
ROUND((SUM(home_76_90) + SUM(away_76_90)) * 100.0 / NULLIF(SUM(home_0_15) + SUM(away_0_15) + SUM(home_16_30) + SUM(away_16_30) + SUM(home_31_45) + SUM(away_31_45)), 2) AS home_away_advantage
```

```
; grant execute on function get_distribution_goals_by_competition(text) to anon; -- Recherche des informations sur l'avantage du terrain à l'échelle d'une saison CREATE OR REPLACE
FUNCTION get_home_away_advantage_by_competition() RETURNS TABLE ( competition_name text, proportion_home_win NUMERIC(10,2), proportion_draw NUMERIC(10,2),
proportion_away_win NUMERIC(10,2), home_advantage NUMERIC(10,2) ) LANGUAGE SQL AS
```

```
SELECT -- Nom de la compétition competition_name, -- Proportion des victoires lorsque l'équipe évolue à domicile ROUND(COUNT(CASE WHEN is_home = 1 AND result = 1 THEN 1 END) * 100 / COUNT(is_home = 1)),
-- Proportion des matchs nuls lorsque l'équipe évolue à domicile ROUND(COUNT(CASE WHEN is_home = 1 AND result = 0 THEN 1 END) * 100 / COUNT(is_home = 1)),
-- Proportion des défaites lorsque l'équipe évolue à domicile ROUND(COUNT(CASE WHEN is_home = 1 AND result = -1 THEN 1 END) * 100 / COUNT(is_home = 1)),
-- Proportion des victoires lorsque l'équipe évolue à l'extérieur ROUND(COUNT(CASE WHEN is_home = -1 AND result = 1 THEN 1 END) * 100 / COUNT(is_home = -1)),
-- Proportion des matchs nuls lorsque l'équipe évolue à l'extérieur ROUND(COUNT(CASE WHEN is_home = -1 AND result = 0 THEN 1 END) * 100 / COUNT(is_home = -1)),
-- Proportion des défaites lorsque l'équipe évolue à l'extérieur ROUND(COUNT(CASE WHEN is_home = -1 AND result = -1 THEN 1 END) * 100 / COUNT(is_home = -1))
```

```
; grant execute on function get_home_away_advantage_by_competition() to anon;
```