МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ

СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Практикум №1-2

з курсу «Аналіз даних в інформаційнних системах»

на тему: «Створення процедур завантаження даних»

Викладач: Виконала:

Олійник Ю.О. студентка 2 курсу

Групи ІП-21 Скрипець Ольга ФІОТ

Київ-2024

Для виконання лабораторної роботи було обрано датасет на сайті https://www.kaggle.com/, що містить декілька таблиць, пов’язаних із футболом. У подальшій роботі було взято 4 таблиці: стадіон (stadium), матчі (matches), команди (teams), тренери (managers).

1. Таблиця "stadiums" містить дані про стадіони, такі як їх назва, розташування (місто та країна), а також місткість.
2. Таблиця "matches" містить дані про всі матчі протягом певного періоду, зазначаючи сезон, дату та час, назви домашньої та гостьової команд, стадіон, кількість голів, чи була серія пенальті та відвідування матчу глядачами.
3. Таблиця "teams" містить інформацію про всі команди, що брали участь у турнірі, зокрема назву, країну походження та домашній стадіон.
4. Таблиця "managers" містить інформацію про тренерів, включаючи їх національність, дату народження та команду, яку вони керують.

Посилання на сам датасет, де були взяті дані: <https://www.kaggle.com/datasets/cbxkgl/uefa-champions-league-2016-2022-data>

**Опис джерел даних**

Ось таблиця Stadiums.сsv, яка містить назви 4 полів та їх призначення:

|  |  |
| --- | --- |
| **Назва поля** | **Призначення** |
| name | назва стадіону |
| city | місто, в якому знаходиться стадіон |
| country | країна, в якій знаходиться стадіон |
| capacity | місткість стадіону |

Ось таблиця Teams.сsv, яка містить назви 3 полів та їх призначення:

|  |  |
| --- | --- |
| **Назва поля** | **Призначення** |
| team\_name | назва команди |
| country | країна походження команди |
| home\_stadium | домашній стадіон команди |

Ось таблиця Managers.сsv, яка містить назви 5 полів та їх призначення:

|  |  |
| --- | --- |
| **Назва поля** | **Призначення** |
| first\_name | ім’я тренера |
| last\_name | прізвище тренера |
| nationality | національність тренера |
| date\_of\_birth | дата народження тренера |
| team | команда, яку він керує |

Ось таблиця Matches.сsv, яка містить назви 10 полів та їх призначення:

|  |  |
| --- | --- |
| **Назва поля** | **Призначення** |
| match\_id | ідентифікатор матчу |
| season | сезон, в якому відбувся матч |
| date\_time | дата та час проведення матчу |
| home\_team | назва домашньої команди |
| away\_team | назва гостьової команди |
| stadium | стадіон, на якому відбувся матч |
| home\_team\_score | кількість голів, забитих домашньою командою |
| away\_team\_score | кількість голів, забитих гостьовою командою |
| penalty\_shoot\_out | чи була серія пенальті після матчу |
| attendance | кількість глядачів, які відвідали матч |

**Редагування даних**

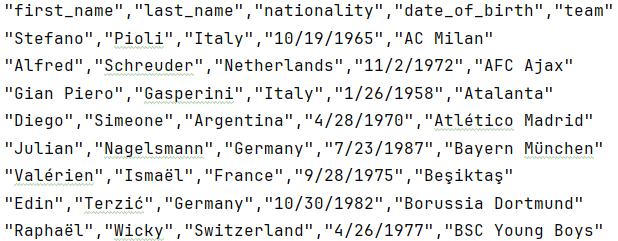
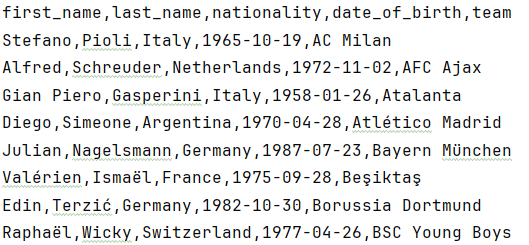
Я відредагувала csv файли для зручної подальшої обробки та аналізу даних. Стандартний формат дати (YYYY-MM-DD) є більш зручним для обробки та аналізу, тому я змінила його на такий.

Також я оновила ідентифікатори матчів, видаляючи «mt» перші два символи, для зручнішого формату ідентифікаторів матчів.

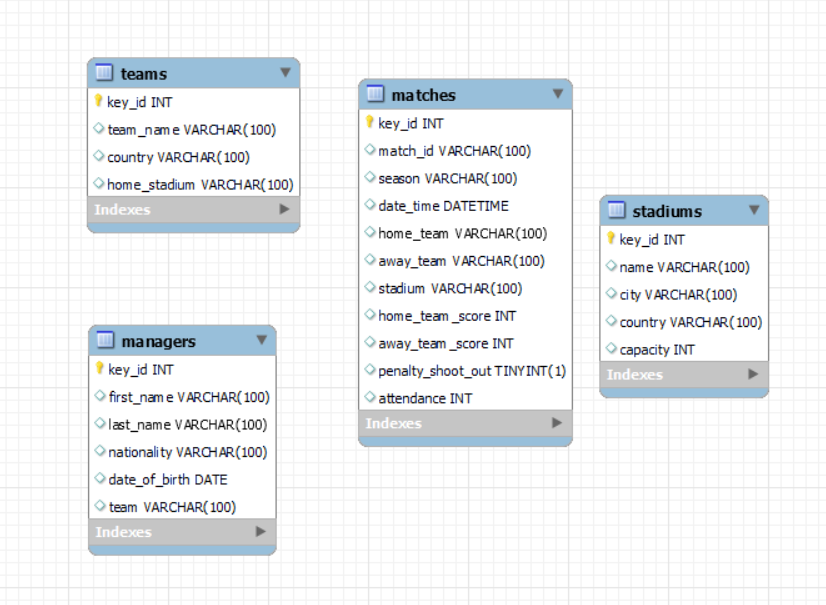
В результаті цих змін отримала оновлені CSV-файли, які тепер можна легко аналізувати. Формат дати та ідентифікаторів матчів був уніфікований. Це полегшує розуміння та аналіз даних.

Код:

import csv  
from datetime import datetime  
  
def date\_edit(row\_name, source, dest, start\_date\_format, flag):  
 with open(source, 'r') as csvfile:  
 reader = csv.DictReader(csvfile)  
 with open(dest, 'w', newline='') as outfile:  
 fieldnames = reader.fieldnames  
 writer = csv.DictWriter(outfile, fieldnames=fieldnames)  
 writer.writeheader()  
 for row in reader:  
 date\_str = row[row\_name]  
 date = datetime.strptime(date\_str, start\_date\_format)  
 updated\_date\_str = date.strftime('%Y-%m-%d')  
 row[row\_name] = updated\_date\_str  
 if flag == 1:  
 old\_value = row['match\_id']  
 new\_value = old\_value[2:]  
 row['match\_id'] = new\_value  
 writer.writerow(row)  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 date\_edit('date\_of\_birth', 'managers.csv', 'updated\_managers.csv', '%m/%d/%Y', 0)  
 date\_edit('date\_time', 'matches.csv', 'updated\_matches.csv', '%d-%b-%y %I.%M.%S.%f000000 PM', 1)



**Stage зона**

****

Таблиця Stadiums містить:

* key\_id - це автоматично збільшуване ціле число, яке використовується як первинний ключ.
* name, city і country - це рядки (VARCHAR), які можуть містити до 100 символів. Вони використовуються для зберігання назви стадіону, міста та країни відповідно.
* capacity - це ціле число (INT), яке використовується для зберігання місткості стадіону.

Ця таблиця дозволяє зберігати інформацію про стадіони, включаючи їх назву, місто та країну розташування, а також місткість. Це може бути корисним для аналізу даних про матчі, які проводилися на цих стадіонах, або для вивчення впливу різних факторів (наприклад, місткості стадіону) на результати матчів.

Таблиця Teams містить:

* key\_id - це автоматично збільшуване ціле число, яке використовується як первинний ключ.
* team\_name, country і home\_stadium - це рядки (VARCHAR), які можуть містити до 100 символів. Вони використовуються для зберігання назви команди, країни походження та домашнього стадіону відповідно.

Ця таблиця дозволяє зберігати інформацію про команди, включаючи їх назву, країну походження та домашній стадіон. Це може бути корисним для аналізу даних про матчі, які проводилися цими командами, або для вивчення впливу різних факторів (наприклад, домашнього стадіону) на результати матчів.

Таблиця Managers містить:

* key\_id - це автоматично збільшуване ціле число, яке використовується як первинний ключ.
* first\_name, last\_name, nationality і team - це рядки (VARCHAR), які можуть містити до 100 символів. Вони використовуються для зберігання імені, прізвища, національності тренера та команди, яку він керує, відповідно.
* date\_of\_birth - це дата (DATE), яка використовується для зберігання дати народження тренера.

Ця таблиця дозволяє зберігати інформацію про тренерів, включаючи їх ім’я, національність, дату народження та команду, яку вони керують. Це може бути корисним для аналізу даних про тренерів та їх вплив на успіхи команд. Завдяки первинному ключу key\_id можна легко з’єднати дані з іншими таблицями, які містять пов’язану інформацію.

Таблиця Matches містить:

* key\_id - це автоматично збільшуване ціле число, яке використовується як первинний ключ.
* match\_id, season, home\_team, away\_team і stadium - це рядки (VARCHAR), які можуть містити до 100 символів. Вони використовуються для зберігання ідентифікатора матчу, сезону, назв домашньої та гостьової команд та стадіону відповідно.
* date\_time - це дата та час (DATETIME), які використовуються для зберігання дати та часу проведення матчу.
* home\_team\_score, away\_team\_score і attendance - це цілі числа (INT), які використовуються для зберігання кількості голів, забитих домашньою та гостьовою командами, та кількості глядачів відповідно.
* penalty\_shoot\_out - це булеве значення (BOOLEAN), яке використовується для вказівки, чи була серія пенальті після матчу.

Ця таблиця дозволяє зберігати інформацію про матчі, включаючи ідентифікатор матчу, сезон, дату та час проведення, команди, стадіон, кількість голів, чи була серія пенальті та кількість глядачів. Це може бути корисним для аналізу даних про матчі та вивчення впливу різних факторів на результати матчів. Завдяки первинному ключу key\_id можна легко з’єднати дані з іншими таблицями, які містять пов’язану інформацію.

Код створення Stage зони:

DROP database if exists STAGE;

CREATE database STAGE;

USE STAGE;

CREATE TABLE Stadiums (

key\_id INT AUTO\_INCREMENT,

name VARCHAR(100),

city VARCHAR(100),

country VARCHAR(100),

capacity INT,

PRIMARY KEY (key\_id)

);

CREATE TABLE Teams (

key\_id INT AUTO\_INCREMENT,

team\_name VARCHAR(100),

country VARCHAR(100),

home\_stadium VARCHAR(100),

PRIMARY KEY (key\_id)

);

CREATE TABLE Managers (

key\_id INT AUTO\_INCREMENT,

first\_name VARCHAR(100),

last\_name VARCHAR(100),

nationality VARCHAR(100),

date\_of\_birth DATE,

team VARCHAR(100),

PRIMARY KEY (key\_id)

);

CREATE TABLE Matches (

key\_id INT AUTO\_INCREMENT,

match\_id VARCHAR(100),

season VARCHAR(100),

date\_time DATETIME,

home\_team VARCHAR(100),

away\_team VARCHAR(100),

stadium VARCHAR(100),

home\_team\_score INT,

away\_team\_score INT,

penalty\_shoot\_out BOOLEAN,

attendance INT,

PRIMARY KEY (key\_id)

);

**Заповнення Stage зони**

USE STAGE;

SHOW VARIABLES LIKE 'secure\_file\_priv';

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/stadiums.csv'

INTO TABLE Stadiums

FIELDS TERMINATED BY ','

ENCLOSED BY '"'

LINES TERMINATED BY '\n'

IGNORE 1 ROWS

(name, city, country, capacity);

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/teams.csv'

INTO TABLE Teams

FIELDS TERMINATED BY ','

ENCLOSED BY '"'

LINES TERMINATED BY '\n'

IGNORE 1 ROWS

(team\_name, country, home\_stadium);

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/managers.csv'

INTO TABLE Managers

FIELDS TERMINATED BY ','

ENCLOSED BY '"'

LINES TERMINATED BY '\n'

IGNORE 1 ROWS

(first\_name, last\_name, nationality, date\_of\_birth, team);

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/matches.csv'

INTO TABLE Matches

FIELDS TERMINATED BY ','

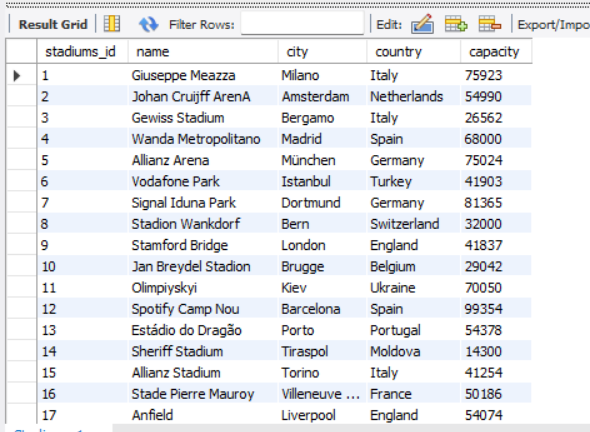
ENCLOSED BY '"'

LINES TERMINATED BY '\n'

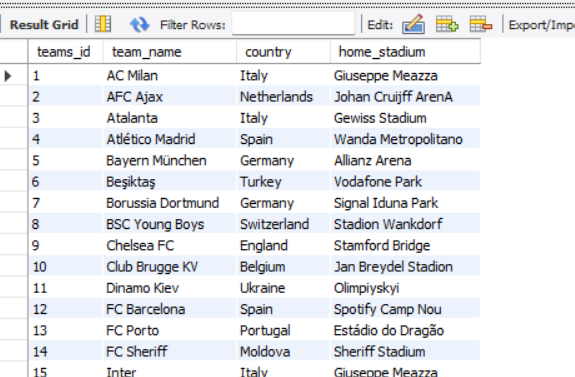
IGNORE 1 ROWS

(match\_id, season, date\_time, home\_team, away\_team, stadium, home\_team\_score, away\_team\_score, penalty\_shoot\_out, attendance);

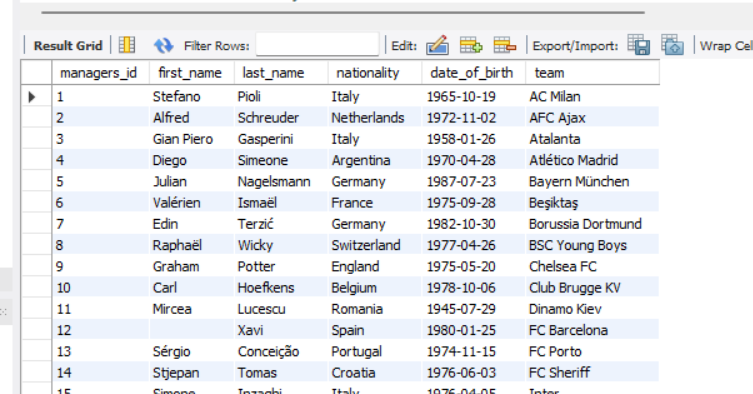
Таблиця Stadiums:



Таблиця Teams:



Таблиця Managers:

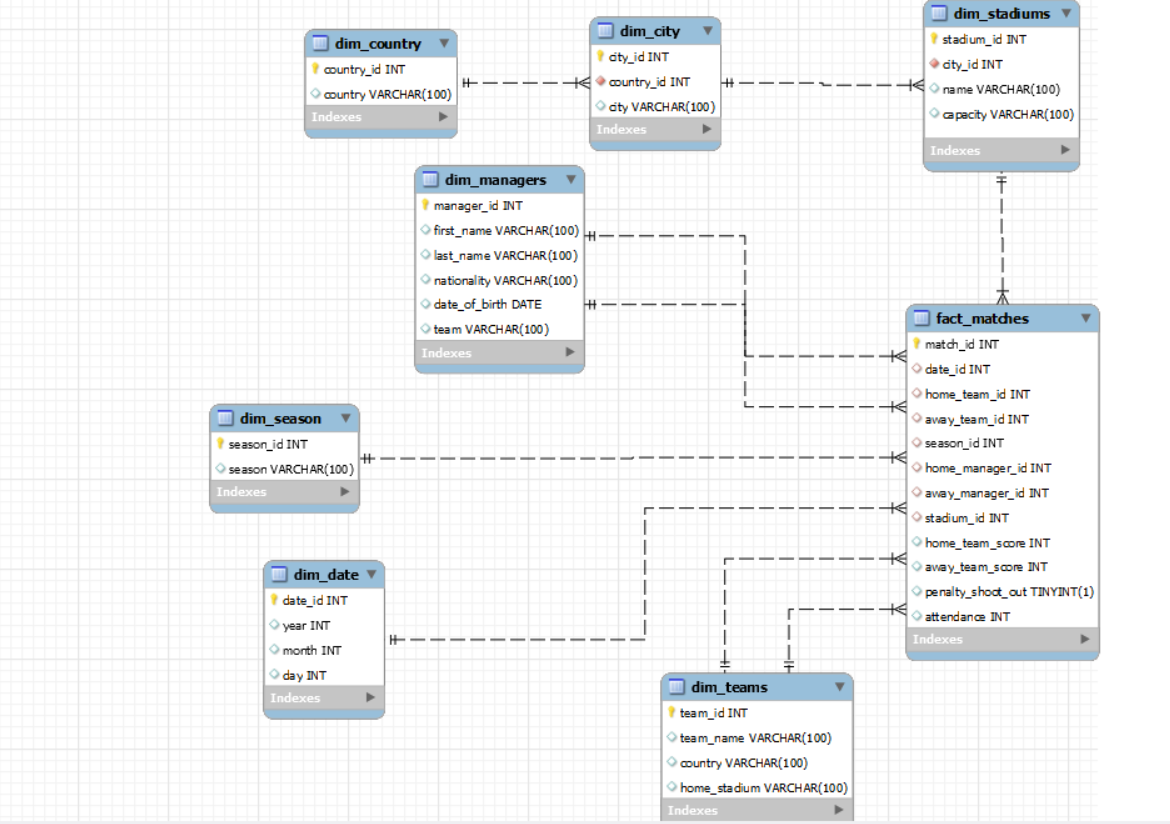


Таблиця Matches:



Отже, дані успішно завантажені до Stage зони

**Модель основного сховища**

****

У моїй базі даних я маю фактову таблицю, яка називається fact\_Matches. Ця таблиця є центральною частиною схеми зірки і містить інформацію про кожен окремий матч, який відбувся.

Кожен запис в таблиці fact\_Matches відповідає одному матчу. Він містить інформацію про дату матчу, команди, які брали участь, сезон, в якому він відбувся, тренерів обох команд, стадіон, на якому він відбувся, а також результати матчу.

Результати матчу включають кількість голів, забитих кожною командою, чи була серія пенальті, та відвідуванність матчу. Ці дані дозволяють нам аналізувати результати матчів та вивчати різні аспекти гри.

Таблиця fact\_Matches містить зовнішні ключі на таблиці виміри, які містять детальні дані про команди, стадіони, тренерів, сезони та дати. Ці таблиці виміри надають мені додаткову інформацію, яка допомагає краще розуміти контекст кожного матчу.

Наприклад, таблиця dim\_Teams містить інформацію про кожну команду, включаючи її назву, країну та домашній стадіон. Таблиця dim\_Managers містить інформацію про тренерів, включаючи їх імена, національність, дату народження та команду, яку вони тренують. Таблиці dim\_\_Season та dim\_Date містять інформацію про сезони та дати відповідно.

Використовуючи цю схему зірки, я можу легко виконувати складні запити до бази даних та отримувати цінну інформацію про матчі, команди, тренерів та інші аспекти нашого дослідження.

Код створення основного сховища:

DROP database MAIN\_STORAGE;

CREATE database MAIN\_STORAGE;

USE MAIN\_STORAGE;

CREATE TABLE dim\_Teams (

team\_id INT AUTO\_INCREMENT,

team\_name VARCHAR(100),

country VARCHAR(100),

home\_stadium VARCHAR(100),

PRIMARY KEY (team\_id)

);

CREATE TABLE dim\_Date (

date\_id INT AUTO\_INCREMENT,

year INT,

month INT,

day INT,

PRIMARY KEY (date\_id)

);

CREATE TABLE dim\_Season (

season\_id INT AUTO\_INCREMENT,

season VARCHAR(100),

PRIMARY KEY (season\_id)

);

CREATE TABLE dim\_Country (

country\_id INT AUTO\_INCREMENT,

country VARCHAR(100),

PRIMARY KEY (country\_id)

);

CREATE TABLE dim\_City (

city\_id INT AUTO\_INCREMENT,

country\_id INT NOT NULL,

city VARCHAR(100),

PRIMARY KEY (city\_id),

FOREIGN KEY (country\_id) REFERENCES dim\_Country (country\_id)

);

CREATE TABLE dim\_Stadium (

stadium\_id INT AUTO\_INCREMENT,

source\_id INT DEFAULT NULL,

city\_id INT NOT NULL,

name VARCHAR(100),

capacity VARCHAR(100),

start\_date DATE DEFAULT NULL,

end\_date DATE DEFAULT NULL,

PRIMARY KEY (stadium\_id),

FOREIGN KEY (city\_id) REFERENCES dim\_City (city\_id)

);

CREATE TABLE dim\_Managers (

manager\_id INT AUTO\_INCREMENT,

first\_name VARCHAR(100),

last\_name VARCHAR(100),

nationality VARCHAR(100),

date\_of\_birth DATE,

team VARCHAR(100),

PRIMARY KEY (manager\_id)

);

CREATE TABLE fact\_Matches (

match\_id INT AUTO\_INCREMENT,

date\_id INT,

home\_team\_id INT,

away\_team\_id INT,

season\_id INT,

home\_manager\_id INT,

away\_manager\_id INT,

stadium\_id INT,

home\_team\_score INT,

away\_team\_score INT,

penalty\_shoot\_out BOOLEAN,

attendance INT,

PRIMARY KEY (match\_id),

FOREIGN KEY (date\_id) REFERENCES dim\_Date (date\_id),

FOREIGN KEY (home\_team\_id) REFERENCES dim\_Teams (team\_id),

FOREIGN KEY (away\_team\_id) REFERENCES dim\_Teams (team\_id),

FOREIGN KEY (season\_id) REFERENCES dim\_Season (season\_id),

FOREIGN KEY (home\_manager\_id) REFERENCES dim\_Managers (manager\_id),

FOREIGN KEY (away\_manager\_id) REFERENCES dim\_Managers (manager\_id),

FOREIGN KEY (stadium\_id) REFERENCES dim\_Stadium (stadium\_id)

);

**Заповнення основного сховища**

USE STAGE;

INSERT INTO MAIN\_STORAGE.dim\_Date(year, month, day)

SELECT DISTINCT YEAR(date\_time), MONTH(date\_time), DAY(date\_time)

FROM matches;

INSERT INTO MAIN\_STORAGE.dim\_City(country\_id, city)

SELECT DISTINCT dc.country\_id, city

FROM stadiums

JOIN MAIN\_STORAGE.dim\_Country dc ON dc.country = stadiums.country;

INSERT INTO MAIN\_STORAGE.dim\_Season(season)

SELECT DISTINCT season

FROM matches;

INSERT INTO MAIN\_STORAGE.dim\_Teams(team\_name, country, home\_stadium)

SELECT DISTINCT team\_name, country, home\_stadium

FROM teams;

INSERT INTO MAIN\_STORAGE.dim\_Managers(first\_name, last\_name, nationality, date\_of\_birth, team)

SELECT DISTINCT first\_name, last\_name, nationality, date\_of\_birth, team

FROM managers;

INSERT INTO MAIN\_STORAGE.dim\_Country(country)

SELECT DISTINCT country

FROM stadiums;

INSERT INTO MAIN\_STORAGE.dim\_Stadiums(city\_id, name, capacity)

SELECT DISTINCT city\_id, name, capacity

FROM stadiums

JOIN MAIN\_STORAGE.dim\_City ds ON ds.city = stadiums.city;

-- Вставляємо date\_id

INSERT INTO MAIN\_STORAGE.fact\_Matches(date\_id)

SELECT dd.date\_id

FROM STAGE.Matches m

JOIN MAIN\_STORAGE.dim\_Date dd ON YEAR(m.date\_time) = dd.year AND MONTH(m.date\_time) = dd.month AND DAY(m.date\_time) = dd.day;

-- Вставляємо home\_team\_id

UPDATE MAIN\_STORAGE.fact\_Matches fm

JOIN STAGE.Matches m ON fm.match\_id = m.match\_id

JOIN MAIN\_STORAGE.dim\_Teams dt ON m.home\_team = dt.team\_name

SET fm.home\_team\_id = dt.team\_id;

-- Вставляємо away\_team\_id

UPDATE MAIN\_STORAGE.fact\_Matches fm

JOIN STAGE.Matches m ON fm.match\_id = m.match\_id

JOIN MAIN\_STORAGE.dim\_Teams dt ON m.away\_team = dt.team\_name

SET fm.away\_team\_id = dt.team\_id;

-- Вставляємо season\_id

UPDATE MAIN\_STORAGE.fact\_Matches fm

JOIN STAGE.Matches m ON fm.match\_id = m.match\_id

JOIN MAIN\_STORAGE.dim\_Season ds ON m.season = ds.season

SET fm.season\_id = ds.season\_id;

-- Вставляємо stadium\_id

UPDATE MAIN\_STORAGE.fact\_Matches fm

JOIN STAGE.Matches m ON fm.match\_id = m.match\_id

JOIN MAIN\_STORAGE.dim\_Stadiums dst ON m.stadium = dst.name

SET fm.stadium\_id = dst.stadium\_id;

-- Вставляємо home\_team\_score

UPDATE MAIN\_STORAGE.fact\_Matches fm

JOIN STAGE.Matches m ON fm.match\_id = m.match\_id

SET fm.home\_team\_score = m.home\_team\_score;

-- Вставляємо away\_team\_score

UPDATE MAIN\_STORAGE.fact\_Matches fm

JOIN STAGE.Matches m ON fm.match\_id = m.match\_id

SET fm.away\_team\_score = m.away\_team\_score;

SET SQL\_SAFE\_UPDATES = 0;

-- Вставляємо penalty\_shoot\_out

UPDATE MAIN\_STORAGE.fact\_Matches fm

JOIN STAGE.Matches m ON fm.match\_id = m.match\_id

SET fm.penalty\_shoot\_out = m.penalty\_shoot\_out;

-- Вставляємо attendance

UPDATE MAIN\_STORAGE.fact\_Matches fm

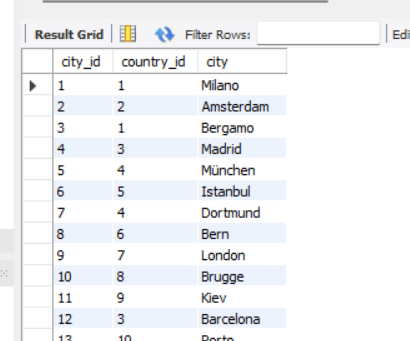
JOIN STAGE.Matches m ON fm.match\_id = m.match\_id

SET fm.attendance = m.attendance;

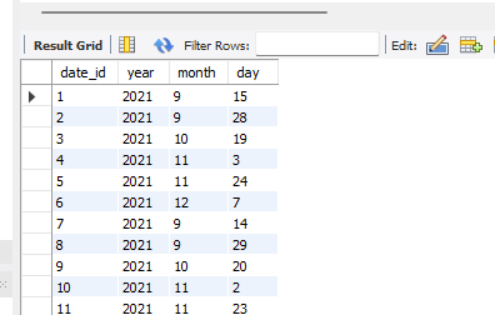
Таблиця dim\_Teams:



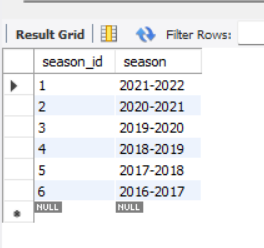
Таблиця dim\_City:



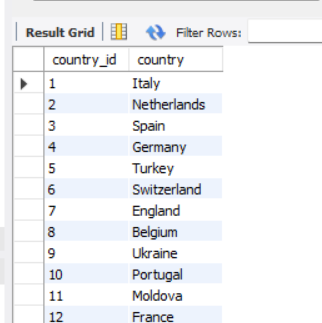
Таблиця dim\_Date:



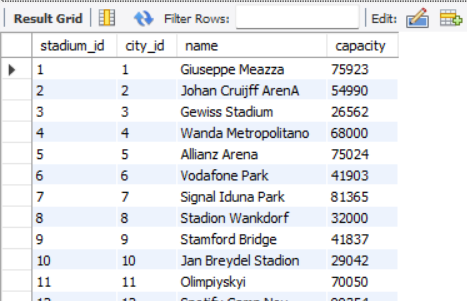
Таблиця dim\_Season:



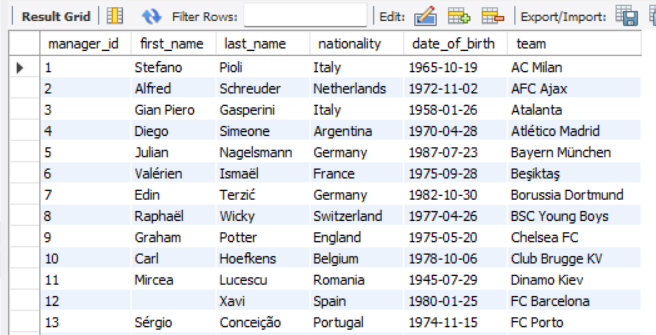
Таблиця dim\_Country:



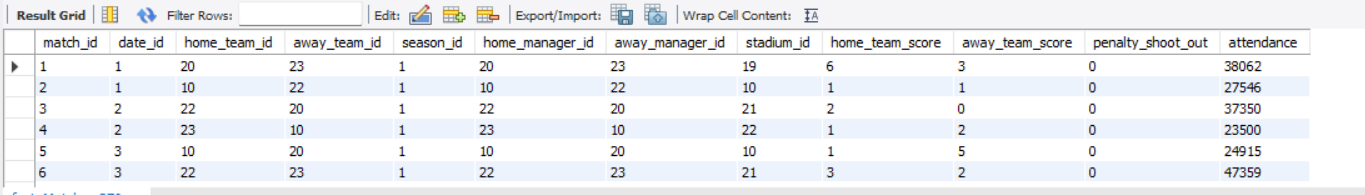
Таблиця dim\_Stadiums:



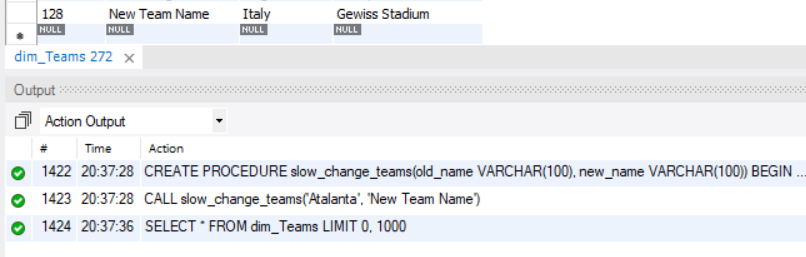
Таблиця dim\_Managers:



Таблиця fact\_Matches:



**slowly changing dimension**



Моя база даних містить інформацію про команди, менеджерів, стадіони тощо. Ці сутності можуть змінюватися з часом. Наприклад, команда може змінити свою назву, менеджер може перейти до іншої команди, а стадіон може змінити свою ємність.

SCD дозволяє відслідковувати ці зміни в часі. Тому мій код використовує SCD для відслідковування змін в назвах команд. Коли команда змінює свою назву, я створюю новий запис в таблиці dim\_Teams з новою назвою, але зберігаю також і старі значення для країни та домашнього стадіону. Також оновлюю відповідні записи в таблиці fact\_Matches, щоб вони відображали нову назву команди. Це дозволяє відслідковувати, як зміна назви команди впливає на її виступи в матчах.

Використання SCD допомагає забезпечити точність та цілісність даних. Воно також дозволяє зберігати більш детальну історію змін, що може бути корисною для глибокого аналізу та прогнозування.

Цей код створює процедуру , яка змінює назву команди в таблиці dim\_Teams та оновлює відповідні записи в таблиці fact\_Matches:

USE MAIN\_STORAGE;

DROP PROCEDURE IF EXISTS slow\_change\_teams;

DELIMITER //

CREATE PROCEDURE slow\_change\_teams(old\_name VARCHAR(100), new\_name VARCHAR(100))

BEGIN

DECLARE old\_id INT DEFAULT NULL;

DECLARE old\_country VARCHAR(100);

DECLARE old\_home\_stadium VARCHAR(100);

SELECT team\_id, country, home\_stadium

INTO old\_id, old\_country, old\_home\_stadium

FROM dim\_Teams

WHERE team\_name = old\_name;

IF old\_id IS NULL THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'The old name of the team does not exist';

ELSE

INSERT INTO dim\_Teams (team\_name, country, home\_stadium)

VALUES (new\_name, old\_country, old\_home\_stadium);

UPDATE fact\_Matches

SET home\_team\_id = (SELECT team\_id FROM dim\_Teams WHERE team\_name = new\_name)

WHERE home\_team\_id = old\_id;

UPDATE fact\_Matches

SET away\_team\_id = (SELECT team\_id FROM dim\_Teams WHERE team\_name = new\_name)

WHERE away\_team\_id = old\_id;

END IF;

END //

DELIMITER ;

CALL slow\_change\_teams('Atalanta', 'New Team Name');

**Incremental load**

Поступове завантаження відіграє важливу роль у забезпеченні актуальності вашого сховища даних, мінімізуючи при цьому використання ресурсів. Воно передбачає ідентифікацію та вибіркове завантаження лише нових записів, що оптимізує продуктивність та зменшує накладні витрати на обробку.

Моя база даних, яка включає таблиці команд, менеджерів, стадіонів та матчів, використовує стратегію поступового завантаження для ефективного оновлення даних. Це зроблено шляхом ідентифікації нових записів в проміжній області та їх вибіркового завантаження в відповідні таблиці в сховищі.

Код:

USE STAGE;

-- Завантажте нові команди в таблицю dim\_Teams

INSERT INTO MAIN\_STORAGE.dim\_Teams (team\_name, country, home\_stadium)

SELECT DISTINCT team\_name, country, home\_stadium

FROM STAGE.Teams t

WHERE NOT EXISTS (

SELECT 1

FROM MAIN\_STORAGE.dim\_Teams dt

WHERE dt.team\_name = t.team\_name

);

-- Завантажте нові стадіони в таблицю dim\_Stadiums

INSERT INTO MAIN\_STORAGE.dim\_Stadiums (name, city\_id, capacity)

SELECT DISTINCT s.name, c.city\_id, s.capacity

FROM STAGE.Stadiums s

JOIN MAIN\_STORAGE.dim\_City c ON s.city = c.city

WHERE NOT EXISTS (

SELECT 1

FROM MAIN\_STORAGE.dim\_Stadiums ds

WHERE ds.name = s.name

);

-- Завантажте нових менеджерів в таблицю dim\_Managers

INSERT INTO MAIN\_STORAGE.dim\_Managers (first\_name, last\_name, nationality, date\_of\_birth, team)

SELECT DISTINCT first\_name, last\_name, nationality, date\_of\_birth, team

FROM STAGE.Managers m

WHERE NOT EXISTS (

SELECT 1

FROM MAIN\_STORAGE.dim\_Managers dm

WHERE dm.first\_name = m.first\_name AND dm.last\_name = m.last\_name

);

-- Завантажте нові матчі в таблицю fact\_Matches

INSERT INTO MAIN\_STORAGE.fact\_Matches (date\_id, home\_team\_id, away\_team\_id, season\_id, home\_manager\_id, away\_manager\_id, stadium\_id, home\_team\_score, away\_team\_score, penalty\_shoot\_out, attendance)

SELECT d.date\_id, ht.team\_id, at.team\_id, s.season\_id, hm.manager\_id, am.manager\_id, st.stadium\_id, m.home\_team\_score, m.away\_team\_score, m.penalty\_shoot\_out, m.attendance

FROM STAGE.Matches m

JOIN MAIN\_STORAGE.dim\_Date d ON DATE(m.date\_time) = CONCAT(d.year, '-', d.month, '-', d.day)

JOIN MAIN\_STORAGE.dim\_Teams ht ON m.home\_team = ht.team\_name

JOIN MAIN\_STORAGE.dim\_Teams at ON m.away\_team = at.team\_name

JOIN MAIN\_STORAGE.dim\_Season s ON m.season = s.season

JOIN MAIN\_STORAGE.dim\_Managers hm ON ht.team\_name = hm.team

JOIN MAIN\_STORAGE.dim\_Managers am ON at.team\_name = am.team

JOIN MAIN\_STORAGE.dim\_Stadiums st ON m.stadium = st.name

WHERE NOT EXISTS (

SELECT 1

FROM MAIN\_STORAGE.fact\_Matches fm

WHERE fm.match\_id = m.match\_id

);

**Висновок**

У цьому проекті було розроблено сховище даних, спеціально призначене для зберігання інформації, пов’язаної з футболом. Основний акцент було зроблено на впровадженні стратегії поступового завантаження, що гарантує, що лише нові або змінені дані з проміжної бази даних (STAGE) вставляються в відповідні таблиці сховища (MAIN\_STORAGE). Цей підхід мінімізує використання ресурсів і забезпечує консистентність даних.

База даних, яка включає таблиці dim\_Teams, dim\_Managers, dim\_Stadiums та fact\_Matches, використовує цю стратегію для ефективного оновлення даних. Це забезпечує, що сховище завжди містить найновішу інформацію, зберігаючи при цьому цілісність і послідовність даних.

Цей підхід до поступового завантаження даних дозволяє максимально ефективно використовувати ресурси, забезпечуючи при цьому актуальність даних у сховищі.