DOLUNAY | Yazılım

Week 1 & 2 | Database

Introduction

İlk 2 haftada database dünyasına giriş yapıyoruz.

Mini proje olarak 6 hafta sonunda çıkaracağımız “event webpage” imiz için bu iki haftada database inşa edeceğiz.

Bir API aracılığıyla event(etkinlik) verileri çekip database’imize yükleyeceğiz. Böylece içi bir miktar event bilgisiyle dolu bir database üzerinde çalışmamız için hazır olacak.

Why we have it?

Imagine you have a lot of information or data, like a collection of phone numbers, names, and addresses for all your friends. Now, think about how you'd manage and organize this information if you didn't have a phone book or a digital address book. It would be challenging to find a specific friend's contact details quickly.

That's where databases come in.

Databases are like organized, digital filing cabinets for storing and managing large amounts of information. They exist to:

Store Information: Databases store data in a structured way, making it easy to find, update, and manage. It's like having a well-organized address book for your data.

Retrieve Data Quickly: With a database, you can quickly retrieve specific pieces of information without having to search through every single item. It's like finding a friend's phone number in your address book without flipping through all the pages.

Ensure Data Integrity: Databases help maintain the accuracy and consistency of data. If a friend changes their phone number, you update it in one place (the database), and it's automatically reflected everywhere. No need to update multiple lists.

Allow Multiple Users: Databases can be accessed by multiple users simultaneously. Just like how you and your friends can share and update information in a shared online document, databases enable multiple users to interact with the data at the same time.

In a nutshell, databases make it easier to manage, organize, and retrieve information. They are like digital assistants that keep your data in order, ensuring that you can find what you

need when you need it. So, the next time you think about all the information you have, imagine how much more challenging it would be without a reliable system like a database to keep everything organized.

Tech stack

#python , #postgresql , #API

Task definition

Objective: In this task, we will first use a third party API to fetch data. Then, we will create our database schema according to the defined models in it. At the end of these 2 weeks, you will have a functional infrastructure to successfully do CRUD operations on it.

What will you learn? :

- Introduction to PostgreSQL and database design principles.
- Designing tables for users, events, categories, and feedback.
- Implementing relationships (e.g., one-to-many for users and events).
- Basic SQL commands for CRUD operations.
- Setting up a local PostgreSQL database for development.

Recommended Tools & Languages:

- PostgreSQL (installation: https://www.postgresql.org/download/)
- Python (installation: https://www.python.org/downloads/)
- Database Visual Tool (we recommend pgAdmin, choose your operating system (Windows or Mac) and install from here : https://www.pgadmin.org/download/)
- IDE (your fav 🙂, biz karışmıyoruz ona, istiyorsan word 🙂)

Steps to-do:

1. **Installation:**
    a. Install required tools & packages on your machine i.e postgresql and listed ones above.
2. **API Usage & Data Fetching:**
    a. **Use Ticketmaster API to fetch data**:

        Bu aşamada Ticketmaster'ın API'ını kullanarak onların sistemlerinden veri çekeceğiz (data fetching). API yazılımın her noktasında kullanılan önemli bir kavram, yarışmalarda da şirketler genelde bir API vererek onların data'larını bu API aracılığıyla kullanmamızı istiyor. Bu sebeple ilk olarak burada data fetching adımı ile başlıyoruz.

        What is an API?

        API stands for Application Programming Interface.

In simple terms, it's a set of rules and tools that allows one piece of software (or application) to interact with another.

It's like a bridge that enables different software systems to communicate with each other.

An API request URL is a specific web address that is used to interact with an API. It serves as a unique identifier for a particular API endpoint, and it includes information about the operation or data retrieval being requested.

Bir de Türkçesine bakalım:

API bizim ana veri kaynağına ulaşmamızı sağlayan bir aracıdır. Dataların asıl tutulduğu server'a bir istek (request) atarak bu dataları kendi tarafımıza çekeriz. API request genelde bir URL olur ve özel bir URL gönderip istediğimiz bilgileri karşı taraftan alabiliriz.

Ticketmaster çok büyük bir etkinlik sağlayıcısı. Bu link üzerinden Ticketmaster'ın open-source olarak yazılımcılara sağladığı API kaynağına üye olalım: https://developer.ticketmaster.com/

b. **API usage via Python:**
Python üzerinden API kullanımı nasıl yapılır buradan öğrenelim: https://www.dataquest.io/blog/python-api-tutorial/ .

Not: Eğer bu yazıyı okumazsanız ya da herhangi bir yerden API kullanımını bilmiyorsanız bir sonraki adım size Çince gibi gelebilir 🙂.

c. **Create the *fetch_event* Function:**
https://developer.ticketmaster.com/products-and-docs/apis/discovery-api/v2/#search-events-v2 linkindeki Ticketmaster Discovery API'ını kullanarak bir fonksiyon yazacağız.
   - Function name: fetch_events
   - Input: Name of a city from Türkiye (string)
   - Output: Json of the listed events and their filtered information (list of json)

Bu API etkinliklerin BÜTÜN bilgilerini dönüyor ama biz sadece etkinliklerin aşağıdaki verilerini çekmeyi ve yazacağınız fonksiyon sonucunda sadece bu bilgileri içeren JSON'ları dönmeyi istiyoruz. Her bir event bilgisi bir json olarak Ticketmaster'dan dönüyor. Biz bir json listesi (list of jsons) halinde event bilgilerini fonksiyonumuz sonucunda döneceğiz. İstediğimiz event bilgileri:

- name
- genre.name
- segment.name
- address
- city
- localdate
- localtime
- url

Özetle, fonksiyonumuz bir şehir adı alıyor ve bir list of JSONs dönüyor. Bu listemiz verilen şehirdeki her bir etkinliğin yukarıdaki listelediğimiz bilgilerini içeren JSON'lar içeriyor olacak.

Not: Toplamda 20 etkinlik bilgisi dönmemiz (output listemizin length'inin 20 olması 🙂) yeterli.

d. **Create *convert_to_csv* Function:**

Başka bir fonksiyon daha yazıyoruz:

- function name: convert_to_csv
- input: list of jsons, path of the csv file (string)
- output: none (our function converts given json to csv and saves it in your local machine into given path)
- *convert_to_csv* fonksiyonuna input olarak *fetch_events*'den gelen list of jsons outputunu vereceğiz. *convert_to_csv* fonksiyonu bu listeyi bir csv ye dönüştürüp csv'yi de input olarak aldığımız path'e kaydedecek.

Not: CSV dosyası data dünyası için önemli bir dosya tipidir. Buradan CSV dosyasına dair daha fazla şey öğrenebilirsiniz:
https://www.youtube.com/watch?v=UofTplCVkYI

Not 2: name kısmının header'ını event_name olarak kaydetmek SQL kullanırken bir sıkıntı yaşamamak adına iyi olabilir.

3. **Create PostgreSQL Database:**

Database dünyasına kısa bir giriş için bu videoyu izleyebiliriz:
https://www.youtube.com/watch?v=27axs9dO7AE

a. **Creating a PostgreSQL database:**

Kullandığımız Database Visual Tool (mesela pgAdmin) üzerinden ya da istediğimiz herhangi bir yolla database oluşturuyoruz.

pgAdmin üzerinden oluşturmak için buradaki *"Create Database using pgAdmin"* başlığına bakabiliriz :
https://www.tutorialsteacher.com/postgresql/create-database

Database'imizi açtıktan sonra artık o database üzerinde query çalıştırabiliriz. pgadmin'de query tool'unu açmanın yolunu öğrenmek için inceleyebiliriz: https://www.w3schools.com/postgresql/postgresql_pgadmin4.php

b. **Creating a Table:**

To import CSV, we need to create a table.
Check here to learn SQL table creation command:
https://www.w3schools.com/sql/sql_create_table.asp
According to the models fetched from the API, the necessary table will be created with their respective columns on your local.

Bir de Türkçe ifade etmek gerekirse:

CSV dosyamızı yüklemek için uygun table'ı oluşturmamız gerekiyor. Table oluşturma sql query'si için yukarıdaki linki inceleyebiliriz. CSV dosyamızdaki column adlarını kullanarak uygun veri tipleri ile table'ımızı oluşturuyoruz.

c. **Import the CSV File:**
   - Kullandığımız database visual tool üzerinden ya da istediğimiz başka bir methodla artık çektiğimiz dataları database'imize yüklüyoruz.

     pgadmin'e CSV import etmek için inceleyebiliriz:
     https://blog.n8n.io/import-csv-into-postgresql/

4. **Try CRUD Operations on Your Data:**
   a. Try the basic CRUD operations on your data, write necessary SQL scripts.
      - SQL üzerinde 4 ana işlem yapılabilmektedir: **C**reate, **R**ead, **U**pdate, **D**elete.
      - Öncelikle bir **Read** query'si yazalım ve data'mız import edilebilmiş mi bunu görelim.
      - Daha sonrasında 5 farklı **Insert (Create)** query'si yazalım, istediğiniz dataları sallayabilirsiniz mesela Dolunay'da görmek istediğiniz etkinlikler 🙂
      - Sonra bu rowlardan 1 tanesini **Update** query'si ile değiştirelim.
      - Son olarak 1 tane row'u da kurban edelim ve **Delete** query'si ile silelim.
      - Bütün CRUD operationlarını öğrenmek için inceleyebiliriz: https://www.commandprompt.com/education/how-to-create-a-table-in-postgresql-perform-crud-operations/

🎉 İlk bölümün sonuna geldikkk.
⭐ Soruların için her zaman discord'dayız, kocaman bir komünite seni bekliyor.