**Neo4j Aura – Best-Selling Books Dataset Yükleme ve Graph Modelleme Dökümantasyonu**

Bu dokümanda Kaggle üzerinden edindiğim "Best-Selling Books" veri setini Neo4j Aura veritabanına kod tabanlı olarak nasıl yüklediğimi, hangi kararları neden aldığımı ve yaptığım Cypher sorgularını detaylı olarak anlatıyor olacağım.

**Neden Bu Yöntemi Tercih Ettim?**

Kaggle'dan edindiğim CSV dosyası zipli olduğu için Neo4j Data Importer gibi ara yüzlere doğrudan yükleyemedim. Bunun yerine CSV'yi önce **Google Drive** hesabıma yükledim ve daha sonra bu dosyayı **doğrudan Cypher komutları** ile çekerek Neo4j Aura'ya import ettim.

Bu yöntemi tercih etmemin başlıca nedenleri:

* Daha fazla **esneklik** sağlaması (veri temizleme, ayrıştırma gibi)
* Tüm aşamaları **otomasyonlaştırabilmem**
* Kod bazlı olduğu için tekrar tekrar çalıştırılabilir ve versiyonlanabilir olması

metin, ekran görüntüsü, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Cypher komutlarım part1

metin, ekran görüntüsü, yazı tipi, yazılım içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Cypher komutlarım part2

**Veri Yapısına Genel Bakış**

Veri setinde her kitap için şu bilgiler vardı:

* Book: Kitap adı
* Author(s): Yazarlık yapan kişiler (virgülle ayrılı)
* Original language: Kitabın yazıldığı dil
* First published: Yayınlandığı yıl
* Approximate sales in millions: Tahmini satış rakamı (milyon cinsinden)
* Genre: Kitabın türü

Bu yapıyı bir graph veri modeline dönüştürmek için her kavramı ayrı bir **Node** olarak tasarladım ve aralarındaki bağlantıları **Relationship** ile ifade ettim.

**Constraint'ler (Benzersizlik Kuralları)**

Veride tekrar eden yazar veya kitap isimleri olmasın diye şu constraint'leri tanımladım:

CREATE CONSTRAINT book\_title IF NOT EXISTS FOR (b:Book) REQUIRE b.title IS UNIQUE;

CREATE CONSTRAINT author\_name IF NOT EXISTS FOR (a:Author) REQUIRE a.name IS UNIQUE;

Bu constraint'ler sayesinde:

* Kitaplar sadece **bir kez** graph'a ekleniyor
* Yazarlar tekrar tekrar node olarak oluşturulmuyor

**Adım Adım Yaptıklarım**

**1. Kitap Node'larını Oluşturma**

**LOAD CSV WITH HEADERS FROM 'https://drive.google.com/uc?export=download&id=17GpSwhWI6hKu3uDfjI01iXTB0Tp9sUxf' AS row**

**MERGE (b:Book {title: row.Book})**

**SET b.originalLanguage = row.`Original language`,**

**b.firstPublished = toInteger(row.`First published`),**

**b.approximateSalesMillions = toFloat(row.`Approximate sales in millions`),**

**b.genre = row.Genre;**

**2. Yazar Node'larını Oluşturup Kitaplarla Bağlantı Kurma**

**LOAD CSV WITH HEADERS FROM 'https://drive.google.com/uc?export=download&id=17GpSwhWI6hKu3uDfjI01iXTB0Tp9sUxf' AS row**

**WITH row, split(row.`Author(s)`, ',') AS authors**

**UNWIND authors AS author**

**WITH trim(author) AS authorName, row**

**MERGE (a:Author {name: authorName})**

**WITH a, row**

**MATCH (b:Book {title: row.Book})**

**MERGE (a)-[r:WROTE]->(b);**

* split() fonksiyonuyla birden fazla yazarlı kitaplar ayrıldı.
* trim() ile fazladan boşluklar temizlendi.

**3. Kitap-Tür (Genre) Node ve İlişkisi**

**LOAD CSV WITH HEADERS FROM 'https://drive.google.com/uc?export=download&id=17GpSwhWI6hKu3uDfjI01iXTB0Tp9sUxf' AS row**

**WITH row, split(row.Genre, ',') AS genres**

**UNWIND genres AS genre**

**WITH trim(genre) AS genreName, row**

**MERGE (g:Genre {name: genreName})**

**WITH g, row**

**MATCH (b:Book {title: row.Book})**

**MERGE (b)-[r:BELONGS\_TO]->(g);**

Bazı kitaplar birden fazla türe sahip olabileceği için yine split() kullanıldı.

**4. Kitap-Dil (Language) Node ve İlişkisi**

**LOAD CSV WITH HEADERS FROM 'https://drive.google.com/uc?export=download&id=17GpSwhWI6hKu3uDfjI01iXTB0Tp9sUxf' AS row**

**MERGE (l:Language {name: row.`Original language`})**

**WITH l, row**

**MATCH (b:Book {title: row.Book})**

**MERGE (b)-[r:WRITTEN\_IN]->(l);**

Diller ayrı birer node olarak tutuldu ve kitaplarla WRITTEN\_IN ilişkisi kuruldu.

**Sorgu Örnekleri**

**1. En çok kitap yazan yazarlar**

**MATCH (a:Author)-[:WROTE]->(b:Book)**

**RETURN a.name, count(b) AS totalBooks**

**ORDER BY totalBooks DESC**

**LIMIT 5;**

**metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

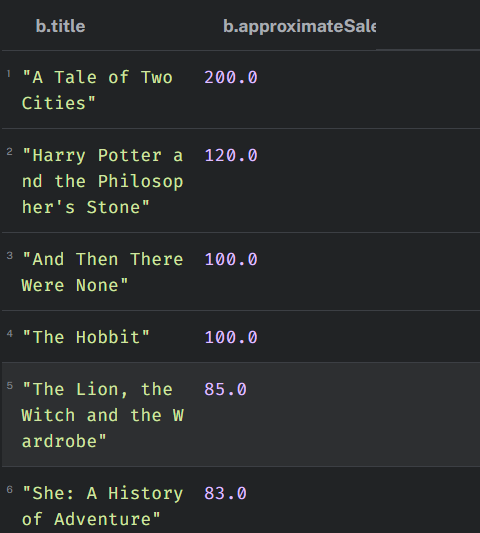
Yapay zeka tarafından oluşturulan içerik yanlış olabilir.**

**2. En çok satılan kitaplar (dile göre)**

**MATCH (b:Book)-[:WRITTEN\_IN]->(:Language {name: "English"})**

**RETURN b.title, b.approximateSalesMillions**

**ORDER BY b.approximateSalesMillions DESC;**

****

**3. Her türdeki en eski kitap**

**MATCH (b:Book)-[:BELONGS\_TO]->(g:Genre)**

**WITH g.name AS genre, b**

**ORDER BY b.firstPublished ASC**

**WITH genre, collect(b)[0] AS oldest**

**RETURN genre, oldest.title AS title, oldest.firstPublished;**

**metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.**

**4. Belirli bir yazarın tüm bağlantılarını gör**

**MATCH (a:Author {name: "J. K. Rowling"})**

**OPTIONAL MATCH (a)-[r:WROTE]->(b:Book)**

**OPTIONAL MATCH (b)-[:BELONGS\_TO]->(g:Genre)**

**OPTIONAL MATCH (b)-[:WRITTEN\_IN]->(l:Language)**

**RETURN a, r, b, g, l;**

**daire, saat, ekran görüntüsü, sanat içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.**

**5. En çok satılan ilk 3 kitabı ve yazarlarını, türlerini ve dillerini göster**

**MATCH (b:Book)**

**WITH b ORDER BY b.approximateSalesMillions DESC LIMIT 3**

**OPTIONAL MATCH (a:Author)-[w:WROTE]->(b)**

**OPTIONAL MATCH (b)-[t:BELONGS\_TO]->(g:Genre)**

**OPTIONAL MATCH (b)-[lrel:WRITTEN\_IN]->(l:Language)**

**RETURN a, w, b, t, g, lrel, l;**

**daire, renklilik, ekran görüntüsü, sanat içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.**

**Sonuç**

Bu ödevde Neo4j’in sunduğu graph veri modelinin geleneksel tablolara göre ne kadar esnek ve güçlü olduğunu deneyimledim. Verileri sadece yüklemekle kalmadım, aynı zamanda düğümler arası ilişkiler sayesinde daha önce fark edilmeyen bağları sorgularla açığa çıkarabildim. Özellikle çok yazarlı kitaplar, çok dilli yayınlar ve çok türlü yapıtlar üzerinden detaylı analizler yapmak oldukça tatmin ediciydi.

Veri modelleme, temizleme ve sorgulama adımlarını kod ile yapmak hem süreci kontrol etmemi hem de gerektiğinde kolayca güncelleme yapmamı sağladı.

**Ayşe Nur Özer**