Neo4j Movies Dataset - Intermediate Cypher Sorguları Raporu

Giriş

Bu rapor Neo4j Movies dataset'i üzerinde intermediate düzeyde Cypher sorguları ve analizlerini içermektedir. Dataset filmler, aktörler, yönetmenler ve aralarındaki ilişkileri (ACTED_IN, DIRECTED, PRODUCED, WROTE) modellemektedir.

1. Gelişmiş Filtreleme ve Aggregation Sorguları

1.1 En Çok Film Çeken Aktörler (Top 10)

cypher

```cypher
MATCH (a:Person)-[:ACTED_IN]->(m:Movie)
RETURN a.name AS actor_name,
       count(m) AS movie_count,
       collect(m.title)[0..3] AS sample_movies
ORDER BY movie_count DESC
LIMIT 10
```

Açıklama: Bu sorgu aktörleri film sayısına göre sıralar ve örnek film isimlerini gösterir.

1.2 Yıllara Göre Film Dağılımı

cypher

```cypher
MATCH (m:Movie)
WHERE m.released IS NOT NULL
RETURN m.released AS year,
       count(m) AS movie_count,
       avg(m.rating) AS avg_rating
ORDER BY year DESC
LIMIT 20
```

1.3 Yüksek Puanlı Filmlerde En Çok Oynayan Aktörler

cypher

```cypher
MATCH (a:Person)-[:ACTED_IN]->(m:Movie)
WHERE m.rating >= 8.0
WITH a, count(m) AS high_rated_movies,
     avg(m.rating) AS avg_movie_rating
WHERE high_rated_movies >= 2
RETURN a.name AS actor_name,
       high_rated_movies,
       round(avg_movie_rating, 2) AS avg_rating
ORDER BY high_rated_movies DESC, avg_rating DESC
```

2. Karmaşık İlişki Analizi Sorguları

2.1 Aktör-Yönetmen İşbirlikleri

cypher

```cypher
MATCH (director:Person)-[:DIRECTED]->(m:Movie)<-[:ACTED_IN]-(actor:Person)
WITH director, actor, count(m) AS collaborations,
     collect(m.title) AS movies
WHERE collaborations >= 2
RETURN director.name AS director_name,
       actor.name AS actor_name,
       collaborations,
       movies
ORDER BY collaborations DESC
```

2.2 Co-Actor Network (Birlikte Oynayan Aktörler)

cypher

```cypher
MATCH (a1:Person)-[:ACTED_IN]->(m:Movie)<-[:ACTED_IN]-(a2:Person)
WHERE id(a1) < id(a2)
WITH a1, a2, count(m) AS shared_movies,
     collect(m.title) AS movie_titles
WHERE shared_movies >= 2
RETURN a1.name AS actor1,
       a2.name AS actor2,
       shared_movies,
       movie_titles
ORDER BY shared_movies DESC
LIMIT 15
```

2.3 Multi-Role Kişiler (Birden Fazla Rolü Olan)

cypher

```cypher
MATCH (p:Person)
OPTIONAL MATCH (p)-[:ACTED_IN]->(m1:Movie)
OPTIONAL MATCH (p)-[:DIRECTED]->(m2:Movie)
OPTIONAL MATCH (p)-[:PRODUCED]->(m3:Movie)
OPTIONAL MATCH (p)-[:WROTE]->(m4:Movie)
WITH p,
     count(DISTINCT m1) AS acted_count,
     count(DISTINCT m2) AS directed_count,
     count(DISTINCT m3) AS produced_count,
     count(DISTINCT m4) AS wrote_count
WHERE (acted_count > 0 AND directed_count > 0) OR
      (acted_count > 0 AND produced_count > 0) OR
      (directed_count > 0 AND produced_count > 0)
RETURN p.name AS person_name,
       acted_count,
       directed_count,
       produced_count,
       wrote_count,
       (acted_count + directed_count + produced_count + wrote_count) AS total_roles
ORDER BY total_roles DESC
```

3. Gelişmiş Path ve Pattern Matching

3.1 Kevin Bacon Derecesi (2-3 derece)

cypher

```cypher
MATCH path = (kevin:Person {name: "Kevin Bacon"})-[:ACTED_IN*2..6]-(other:Person)
WHERE other.name <> "Kevin Bacon"
WITH other, min(length(path)) AS bacon_number
WHERE bacon_number <= 6
RETURN other.name AS actor_name,
       bacon_number/2 AS degrees_from_kevin_bacon
ORDER BY degrees_from_kevin_bacon, actor_name
LIMIT 20
```

3.2 En Uzun Aktör Zinciri

cypher

```cypher
MATCH path = (start:Person)-[:ACTED_IN*4..8]-(end:Person)
WHERE start <> end
RETURN extract(n IN nodes(path) | n.name) AS actor_chain,
       length(path) AS chain_length
ORDER BY chain_length DESC
LIMIT 5
```

3.3 Belirli Türdeki Filmlerde Uzmanlaşan Aktörler

cypher

```cypher
MATCH (a:Person)-[:ACTED_IN]->(m:Movie)
WHERE exists(m.genres)
UNWIND m.genres AS genre
WITH a, genre, count(m) AS genre_count,
     collect(m.title) AS movies_in_genre
WHERE genre_count >= 3
RETURN a.name AS actor_name,
       genre,
       genre_count,
       movies_in_genre[0..3] AS sample_movies
ORDER BY actor_name, genre_count DESC
```

4. Gelişmiş Aggregation ve Statistical Sorguları

4.1 Yönetmenlerin İstatistiksel Analizi

cypher

```cypher
MATCH (d:Person)-[:DIRECTED]->(m:Movie)
WITH d, collect(m) AS movies
WHERE size(movies) >= 2
UNWIND movies AS movie
WITH d, movies,
     avg(movie.rating) AS avg_rating,
     stdev(movie.rating) AS rating_std_dev,
     min(movie.released) AS first_movie_year,
     max(movie.released) AS last_movie_year
RETURN d.name AS director_name,
       size(movies) AS total_movies,
       round(avg_rating, 2) AS average_rating,
       round(rating_std_dev, 2) AS rating_consistency,
       first_movie_year,
       last_movie_year,
       (last_movie_year - first_movie_year) AS career_span
ORDER BY average_rating DESC, total_movies DESC
```

4.2 Dekatlara Göre Film Analizi

cypher

```cypher
MATCH (m:Movie)
WHERE m.released IS NOT NULL
WITH m, (m.released / 10) * 10 AS decade
RETURN decade AS decade_start,
       count(m) AS movie_count,
       round(avg(m.rating), 2) AS avg_rating,
       max(m.rating) AS highest_rating,
       min(m.rating) AS lowest_rating,
       collect(m.title)[0..3] AS sample_movies
ORDER BY decade_start DESC
```

4.3 Aktör Ağı Centrality Analizi

cypher

```cypher
MATCH (a:Person)-[:ACTED_IN]->(m:Movie)<-[:ACTED_IN]-(other:Person)
WHERE a <> other
WITH a, count(DISTINCT other) AS connections,
     count(DISTINCT m) AS movie_count
RETURN a.name AS actor_name,
       connections AS co_actor_connections,
       movie_count,
       round(toFloat(connections) / movie_count, 2) AS connection_ratio
ORDER BY connections DESC
LIMIT 20
```

5. Conditional Logic ve Complex Filtering

5.1 Kariyerinin Farklı Dönemlerindeki Performans

cypher

```cypher
MATCH (a:Person)-[:ACTED_IN]->(m:Movie)
WHERE m.released IS NOT NULL
WITH a, collect({movie: m.title, year: m.released, rating: m.rating}) AS movies
WHERE size(movies) >= 5
WITH a, movies,
     [movie IN movies WHERE movie.year < 1990] AS early_movies,
     [movie IN movies WHERE movie.year >= 1990 AND movie.year < 2000] AS mid_movies,
     [movie IN movies WHERE movie.year >= 2000] AS recent_movies
WHERE size(early_movies) > 0 AND size(recent_movies) > 0
RETURN a.name AS actor_name,
       size(early_movies) AS early_career_movies,
       round(avg([m IN early_movies | m.rating]), 2) AS early_avg_rating,
       size(recent_movies) AS recent_career_movies,
       round(avg([m IN recent_movies | m.rating]), 2) AS recent_avg_rating
ORDER BY actor_name
```

5.2 Kritik ve Ticari Başarı Karşılaştırması

cypher

```cypher
MATCH (m:Movie)
WHERE m.rating IS NOT NULL AND m.revenue IS NOT NULL
WITH m,
     CASE
       WHEN m.rating >= 8.0 THEN "High Rated"
       WHEN m.rating >= 6.0 THEN "Medium Rated"
       ELSE "Low Rated"
     END AS rating_category,
     CASE
       WHEN m.revenue >= 100000000 THEN "Blockbuster"
       WHEN m.revenue >= 50000000 THEN "Commercial Success"
       ELSE "Limited Success"
     END AS commercial_category
RETURN rating_category,
       commercial_category,
       count(m) AS movie_count,
       avg(m.rating) AS avg_rating,
       avg(m.revenue) AS avg_revenue
ORDER BY rating_category, commercial_category
```

6. Performans Optimizasyonu Örnekleri

6.1 Index Kullanımı ve Profiling

cypher

```cypher
// Index oluşturma önerisi
CREATE INDEX person_name_index FOR (p:Person) ON (p.name);
CREATE INDEX movie_released_index FOR (m:Movie) ON (m.released);

// Profiling ile sorgu optimizasyonu
PROFILE
MATCH (a:Person {name: "Tom Hanks"})-[:ACTED_IN]->(m:Movie)
RETURN m.title, m.released
ORDER BY m.released DESC;
```

6.2 Parameterized Query Örneği

cypher

```cypher
// Parametre kullanımı
MATCH (a:Person)-[:ACTED_IN]->(m:Movie)
WHERE m.released >= $min_year AND m.released <= $max_year
WITH a, count(m) AS movie_count
WHERE movie_count >= $min_movies
RETURN a.name, movie_count
ORDER BY movie_count DESC
LIMIT $result_limit
```

Sonuç ve Öneriler

Bu sorguların her biri farklı analiz senaryolarına hizmet etmektedir:

1. **Temel Aggregation:** Veri özetleme ve gruplandırma
2. **İlişki Analizi:** Ağ yapısını anlama
3. **Path Matching:** Bağlantı zincirlerini keşfetme
4. **İstatistiksel Analiz:** Veri dağılımlarını inceleme
5. **Conditional Logic:** Karmaşık filtreleme ve sınıflandırma
6. **Performans:** Sorgu optimizasyonu teknikleri

Bu sorgular movies dataset'inin zengin ilişki yapısını keşfetmek ve anlamlı içgörüler elde etmek için kullanılabilir. Gerçek uygulamalarda, veri boyutuna göre LIMIT ve WHERE koşullarının optimize edilmesi önerilir.