Group Project - Neo4j

of Systems and Methods for Big and Unstructured Data Course (SMBUD)

held by
Brambilla Marco
Tocchetti Andrea

Group 78

Pisante Giuseppe	Raffaelli Martina
10696936	10709893

Academic year 2024/2025

POLITECNICO
MILANO 1863

Contents

1 Introduction

The project aims to design and implement a database system to manage and analyze data related to an e-commerce platform. The database will include entities such as Customer, Product, Order, Category, and Payment. The goal is to create an efficient system that can store and query information about customer purchases, product details, and transactions, providing valuable insights into user behavior, sales trends, and inventory management. The project will be developed using a relational database management system (RDBMS), which will facilitate structured querying of data to support key functions like product recommendations, customer segmentation, sales analysis, and inventory forecasting. The system will allow for tracking of customer purchase history, order details, product categories, payment methods, and discounts, helping businesses optimize their operations and enhance the customer experience.

2 Assumptions

The project is based on the following assumptions:

- Each customer has a unique ID, name, surname, email address, and date of birth.
- Each product has a unique identifier, name, description, price, and category.
- Each product can be associated with multiple categories.
- Each customer can place multiple orders.
- Each order can contain multiple products.
- Each order has a unique ID, order date, total amount, and payment method.
- Each product can have multiple variants (e.g., size, color).
- Each product can have multiple reviews, where each review includes a rating and comments from a customer.
- Each payment is associated with a unique transaction ID, payment method, and amount.
- Each customer can leave a review for a product if and only if they have purchased it.

3 ER diagram

Figure 1: E-R Diagram

3.1 Entities

Starting from the considerations previously exposed regarding the implementation hypotheses, we have drawn an ER diagram (**Figure 1**) which includes 5 different entities and 7 many-to-many relationships described below in the logical model:

- **Customer**(customer_id, Name, Surname, Email, BirthDate, Address, PhoneNumber)
- **Product**(product_id, ProductName, Description, Price, Stock, Category)
- **Order**(order_id, CustomerID, OrderDate, TotalAmount, PaymentMethod, ShippingAddress)

- **Order_Item**(order_id, product_id, Quantity, UnitPrice)
- **Review**(review_id, CustomerID, ProductID, Rating, Comment, ReviewDate)
- **Category**(category_id, CategoryName)
- **Payment**(payment_id, OrderID, PaymentDate, PaymentAmount, PaymentMethod)

The **Customer** entity represents the individuals who make purchases on the platform, storing essential information such as their personal details and contact information. The **Product** entity represents the items available for sale, including product details such as name, description, price, stock level, and category. The **Order** entity captures information related to a customer's order, such as the order date, total amount, payment method, and shipping address. The **Order_Item** entity links products to specific orders, detailing the quantity of each product ordered and its unit price. The **Review** entity stores customer feedback for purchased products, including ratings and comments, along with the review date. The **Category** entity defines the product categories, allowing for classification of products into different types. Finally, the **Payment** entity records the payment details for each order, including the payment method and the amount paid.

3.2 Relationships

ACTED_IN $(Person) - [: ACTED\_IN]- > (Title)$
Relationship between a Person, whose primary profession is actor, and a Title.

DIRECTED $(Person) - [: DIRECTED]- > (Title)$
Relationship between a Person, whose primary profession is director, and a Title.

WROTE $(Person) - [: WROTE]- > (Title)$
Relationship between a Person, whose primary profession is writer, and a Title.

PART_OF $(Episode) - [: PART\_OF]- > (Title)$
Relationship between an Episode and its parent Title, whose TitleType is TV series.

HAS_GENRE $(Title) - [: HAS\_GENRE]- > (Genre)$
Relationship between a Title and a Genre.

HAS_RATING $(Title) - [: HAS\_RATING]- > (Rating)$
Relationship between a Title and its Rating.

3.3 Constraints:

Ciao

4 Cypher Queries

4.1 Market Analysis for Marketing Department

This section of the Cypher Queries aims at providing the Marketing department with insights on the share within the market by providing the number of orders in the Countries in which the Company operates. In particular, this is done by providing an overview on the total orders per country, the revenue per country, a demographic overview per country and an analysis of the most present brands within a specific category, which in this study was chosen as the category with the most orders for further relevance.

4.1.1 Total orders by Country

```
MATCH (u:User)-[:PLACES]->(o:Order)
RETURN u.country AS country, COUNT(o) AS total_orders
ORDER BY total_orders DESC;
```

4.1.2 Revenue per Country

```
MATCH (u:User)-[:PLACES]->(o:Order)-[:CONTAINS]->(oi:OrderItem)
MATCH (oi)-[:REFERS_TO]->(p:Product)
RETURN u.country AS country, SUM(p.cost) AS total_revenue
ORDER BY total_revenue DESC;
```

4.1.3 Demographic Overview

```
MATCH (u:User)
RETURN u.country AS country,
     COUNT(u) AS total_users,
     AVG(u.age) AS average_age
ORDER BY total_users DESC;
```

Table 1: Total Orders by Country

Country	Total Orders
China	42,986
United States	28,099
Brasil	18,262
South Korea	6,620
France	5,968
United Kingdom	5,673
Germany	5,286
Spain	4,965
Japan	2,945
Australia	2,630
Belgium	1,441
Poland	325
Colombia	19
España	4
Austria	2
Deutschland	1

4.1.4 Categories with the most orders

```
MATCH (oi:OrderItem)-[:REFERS_TO]->(p:Product)
RETURN p.product_category AS category, COUNT(oi) AS total_orders
ORDER BY total_orders DESC
LIMIT 3;
```

4.1.5 Most present brands within the intimates category

```
MATCH (oi:OrderItem)-[:REFERS_TO]->(p:Product)
WHERE p.category = "Intimates"

RETURN p.brand AS brand,
      COUNT(oi) AS total_sales
ORDER BY total_sales DESC
LIMIT 5;
```

Table 2: Total Revenue by Country

Country	Total Revenue
China	1,800,865.85
United States	1,162,263.11
Brasil	747,519.90
South Korea	278,958.29
France	243,130.02
United Kingdom	242,263.07
Germany	217,875.61
Spain	210,278.08
Japan	124,807.02
Australia	106,009.74
Belgium	60,226.06
Poland	13,386.90
Colombia	572.51
España	91.87
Deutschland	65.70
Austria	41.51

4.1.6 Most present brands within the intimates category per country

```
MATCH (u:User)-[:ORDERED]->(oi:OrderItem)-[:REFERS_TO]->(p:Product)
WHERE p.category = "Intimates"
WITH u.country AS country,
    p.brand AS brand,
    COUNT(oi) AS total_sales
ORDER BY country, total_sales DESC
WITH country, COLLECT({brand: brand, total_sales: total_sales}) AS brand_sal
RETURN country,
      brand_sales[0].brand AS top_brand,
      brand_sales[0].total_sales AS top_sales
ORDER BY country;
```

4.1.7 Top traffic source per Country

```
MATCH (u:User)
WHERE u.traffic_source IS NOT NULL
```

Table 3: Total Users and Average Age by Country

Country	**Total Orders**	**Average Value**
China	34,150	40.89
United States	22,522	41.21
Brasil	14,507	41.19
South Korea	5,316	41.25
France	4,700	41.57
United Kingdom	4,561	41.05
Germany	4,155	40.86
Spain	4,062	41.01
Japan	2,438	40.89
Australia	2,146	40.98
Belgium	1,185	39.54
Poland	235	42.43
Colombia	17	34.88
Deutschland	2	40.50
España	2	38.50
Austria	2	50.00

Table 4: Categories with the most orders

Category	**Total Orders**
Intimates	13,474
Jeans	12,698
Tops & Tees	11,925

Table 5: Most present brands within the intimates categor

Brand	**Total Sales**
Bali	405
Maidenform	383
Hanes	364
Laura	342
Vanity Fair	306

```
WITH u.country AS country, u.traffic_source AS traffic_source, COUNT(*) AS a
WITH country, traffic_source, amount
ORDER BY country, amount DESC
```

Table 6: Total Orders and Average Value by Country

Country	Total Orders	Average Value
China	34,150	40.89
United States	22,522	41.21
Brasil	14,507	41.19
South Korea	5,316	41.25
France	4,700	41.57
United Kingdom	4,561	41.05
Germany	4,155	40.86
Spain	4,062	41.01
Japan	2,438	40.89
Australia	2,146	40.98
Belgium	1,185	39.54
Poland	235	42.43
Colombia	17	34.88
Deutschland	2	40.50
España	2	38.50
Austria	2	50.00

```
WITH country, COLLECT({traffic_source: traffic_source, amount: amount}) AS t
RETURN country, traffic_data[0].traffic_source AS top_traffic_source, traffi
ORDER BY country;
```

4.1.8 Segmentation of User base

```
MATCH (u:User)-[:PLACES]->(o:Order)-[:CONTAINS]->(oi:OrderItem)-[:REFERS_TO]
WITH u, o, SUM(p.cost) AS total_price, COUNT(oi) AS order_count
WITH u, total_price, order_count,
    CASE
        WHEN total_price > 30 AND order_count > 3 THEN 'High Frequency, High
        WHEN total_price <= 30 AND order_count > 3 THEN 'High Frequency, Low
        WHEN total_price > 30 AND order_count <= 3 THEN 'Low Frequency, High
        ELSE 'Low Frequency, Low Spending'
    END AS segment
RETURN segment, COUNT(u) AS user_count
ORDER BY user_count DESC;
```

Table 7: Search Data by Country

Country	Platform	Count
Australia	Search	1518
Austria	Facebook	1
Belgium	Search	856
Brasil	Search	10147
China	Search	23876
Colombia	Search	9
Deutschland	Search	1
España	Search	2
France	Search	3310
Germany	Search	2921
Japan	Search	1746
Poland	Search	177
South Korea	Search	3701
Spain	Search	2845
United Kingdom	Search	3197
United States	Search	15768

Table 8: Customer Segments by Frequency and Spending

Segment	Count
Low Frequency, Low Spending	67152
Low Frequency, High Spending	51702
High Frequency, High Spending	6340
High Frequency, Low Spending	32

4.1.9 Best and Worst Selling Product Categories by Season

This query analyzes the purchasing patterns of product categories by season. For each season (Winter, Spring, Summer, and Fall), it identifies the most frequently purchased category (best-selling) and the least frequently purchased category (worst-selling). The results are based on the frequency of orders containing products from each category, with the categories being ranked by the total number of purchases in each season.

```
MATCH (u:User)-[:PLACES]->(o:Order)-[:CONTAINS]->(oi:OrderItem)-[:REFERS_TO]
WITH p.category AS product_category,
  CASE
```

```
     WHEN toInteger(substring(o.created_at, 5, 2)) IN [12, 1, 2] THEN 'Wint
     WHEN toInteger(substring(o.created_at, 5, 2)) IN [3, 4, 5] THEN 'Sprin
     WHEN toInteger(substring(o.created_at, 5, 2)) IN [6, 7, 8] THEN 'Summe
     WHEN toInteger(substring(o.created_at, 5, 2)) IN [9, 10, 11] THEN 'Fal
   END AS season,
   COUNT(*) AS frequency
  WITH season, product_category, frequency
  ORDER BY season, frequency DESC
  WITH season, collect({category: product_category, freq: frequency}) AS categ
  RETURN season, categories[0] AS top_category, categories[-1] AS worst_catego
```

Table 9: Customer Segments by Frequency and Spending

Season	TopCategory	WorstCategory
”Fall”	”category”: ”Intimates”,”freq”: 3854	”category”: ”Clothing Sets”,”freq”: 63
”Spring”	”category”: ”Intimates”, ”freq”: 2543	”category”: ”Clothing Sets”, ”freq”: 36
”Summer”	”category”: ”Intimates”, ”freq”: 3078	”category”: ”Clothing Sets”, ”freq”: 52
”Winter”	”category”: ”Intimates”, ”freq”: 3999	”category”: ”Clothing Sets”, ”freq”: 62

The result we obtain may seem strange since in all the seasons the top and worst categories are the same, so we checked the correctness of the result with the following query, which computes the frequency of every category in each season:

```
MATCH (u:User)-[:PLACES]->(o:Order)-[:CONTAINS]->(oi:OrderItem)-[:REFERS_TO]->(p
WITH p.category AS product_category,
     CASE
         WHEN toInteger(substring(o.created_at, 5, 2)) IN [12, 1, 2] THEN 'Winte
         WHEN toInteger(substring(o.created_at, 5, 2)) IN [3, 4, 5] THEN 'Spring
         WHEN toInteger(substring(o.created_at, 5, 2)) IN [6, 7, 8] THEN 'Summer
         WHEN toInteger(substring(o.created_at, 5, 2)) IN [9, 10, 11] THEN 'Fall
     END AS season,
     COUNT(*) AS frequency
RETURN product_category, season, frequency
ORDER BY frequency DESC;
```

4.2 Logistic Analysis for Logistics Department

This section of the Cypher Queries aims at providing the Logistics department with insights on the real-time tracking of the orders and all the possible information related to tuser in order to make the delivery as smooth as possible. In particular, this is done by evaluating the closest distribution center for a specific user, the status of the order, the update of the history of the orders of user and to check if the user has some order pending, in which case the items can be sent together.

4.2.1 Closest Distribution Center

```
MATCH (u:User)-[:PERFORM_EVENT]->(h:History)
WHERE u.id = "100"
MATCH (d:DistributionCenter)
WHERE d.latitude IS NOT NULL AND d.longitude IS NOT NULL
WITH u, d,
    point({latitude: u.latitude, longitude: u.longitude}) AS user_location,
    point({latitude: d.latitude, longitude: d.longitude}) AS distribution_cent
WITH u, d, user_location, distribution_center_location,
    point.distance(user_location, distribution_center_location) AS distance
ORDER BY distance
LIMIT 1
RETURN d.name, distance;
```

Table 10: Data for Los Angeles CA

Location	Value
Los Angeles CA	10539304.46257035

4.2.2 Order History

In order to provide our logistic department with the most updated information, we have to provide them with the history of the orders of a specific user. In particular, we have to fill the history node with the latest information. In order to do that, we first add the new order to order, and then we add the new order to the history of the user.

```
// Add the new order to the order node
  MATCH (u:User)
  WHERE u.id="100"
  WITH u
  MATCH (o:Order)
  WITH u, MAX(o.order_id) AS max_order_id
  CREATE (newOrder:Order {
      order_id: toString(200000),
      status: "Processing",
      gender: u.gender,
      created_at: date(),
      returned_at: null,
      shipped_at: null,
      delivered_at: null,
      num_of_item: 3
  })
  MERGE (u)-[:PLACES]->(newOrder);

// Add the new order to the history of the user
  MATCH (u:User)-[:PERFORM_EVENT]->(h:History)
  WHERE u.id = "100"
  MATCH (o:Order)
  WHERE o.order_id ="300000"
  SET h.orders_list = coalesce(h.orders_list, []) + "300000",
      h.number_of_orders = h.number_of_orders +1
  MERGE (u)-[:USER_HAS_ORDER]->(o)
```

4.2.3 Number of orders pending

```
MATCH (u:User {id: <user_id>})-[:PLACES]->(o:Order)
WITH u, count(o) AS total_orders
WHERE total_orders = 10
RETURN u.id AS user_id, total_orders
```

4.2.4 Management of multiple orders

```
MATCH (u:User)-[:PLACES]->(o:Order)
WHERE u.id="100" AND o.status = 'Processing'
RETURN o.order_id, o.status AS order_status, o.created_at;
```

Table 11: Data for Processing Task

ID	Status	Date
300000	Processing	2024-12-11

4.2.5 set shipping date or delivered date

```
MATCH (o:Order )
WHERE o.order_id = "300000"
RETURN o.shipped_at;
```

This can be implemented for the delivery date by switching the attribute **shipped_at** with **delivered_at**.

4.2.6 Order Status

```
MATCH (u:User)-[:PLACES]->(o:Order)
WHERE u.id = "100"
WITH o ORDER BY o.created_at DESC LIMIT 1
RETURN o.order_id, o.status AS order_status, o.created_at;
```

Table 12: Data for Processing Task

ID	Status	Timestamp
561	Processing	2023-12-15 17:28:00+00:00

4.2.7 Cross-Sell Opportunity

```
MATCH (oi1:OrderItem)-[:REFERS_TO]->(p1:Product),
    (oi2:OrderItem)-[:REFERS_TO]->(p2:Product)
WHERE oi1.order_id = oi2.order_id AND p1.id < p2.id
WITH p1.id AS product1_id, p2.id AS product2_id, oi1.order_id AS order_id
WITH product1_id, product2_id, COUNT(DISTINCT order_id) AS frequency
RETURN product1_id, product2_id, frequency
ORDER BY frequency DESC
LIMIT 10;
```

!!!! TODO: ADD TABLE WITH RESULTS

4.2.8 Average Shipping Time by Country

This query computes the average shipping time for each country. In particular it checks if the order has been delivered and created in the same month, and then calculates the difference in days between the two dates.

```
MATCH (u:User)-[:PLACES]->(o:Order)
WHERE o.delivered_at IS NOT NULL AND o.created_at IS NOT NULL
  AND toInteger(substring(o.delivered_at, 5, 2)) = toInteger(substring(o.creat
WITH u.country AS country, AVG(toInteger(substring(o.delivered_at, 8, 2)) - to
    avg_shipping_time
RETURN country, avg_shipping_time
ORDER BY avg_shipping_time DESC;
```

Table 13: Data for Processing Task

Country	AvgShippingTime
”Austria”	6.0
”Colombia”	4.857142857142857
”Japan”	3.9200000000000004
”France”	3.845439650464229
”Spain”	3.8244325767690226
”Brasil”	3.8161648177496104
”Germany”	3.8119062697910056
”Belgium”	3.805104408352669
”United States”	3.797328420082921
”China”	3.7945538906214358
”South Korea”	3.794032723772863
”United Kingdom”	3.7925840092699894
”Australia”	3.779141104294479
”Poland”	3.670103092783506
”España”	2.0

4.3 Customer Segmentation Analysis

This section of the Cypher Queries aims at providing the Marketing department with more deep insights on the habits of a specific segment of users, in particular we are going to perform an age-based segmentation of the users, and then we are going to focus on the biggest segment.

4.3.1 Age-based Segmentation

This query segments users based on their age into the following groups: 18-24, 25-34, 35-44, 45-54, 55-64, and 65+. It then calculates the total number of users, total amount spent, and average amount spent per user for each age group and returns the results in ascending order of age group. Analysing the results we can see that the values of the average amount spent per user are quite similar across all age groups, so we should rely on the total amount spent to identify the most valuable segments, which is going to be the 65+ group.

```
MATCH (u:User)-[:PLACES]->(o:Order)-[:CONTAINS]->(oi:OrderItem)-[:REFERS_TO]->
WITH u,
    CASE
        WHEN u.age >= 18 AND u.age <= 24 THEN ’18-24’
        WHEN u.age >= 25 AND u.age <= 34 THEN ’25-34’
        WHEN u.age >= 35 AND u.age <= 44 THEN ’35-44’
        WHEN u.age >= 45 AND u.age <= 54 THEN ’45-54’
        WHEN u.age >= 55 AND u.age <= 64 THEN ’55-64’
        ELSE ’65+’
    END AS age_group,
    p.cost AS product_price
WITH age_group, COUNT(DISTINCT u) AS user_count, SUM(product_price) AS total_s
WITH age_group, user_count, total_spent,
    CASE
        WHEN user_count > 0 THEN total_spent / user_count
        ELSE 0
    END AS avg_spent_per_user
RETURN age_group, user_count, total_spent, avg_spent_per_user
ORDER BY age_group;
```

Table 14: Customer Segmentation

AgeGroup	UserCount	TotalSpent	AvgSpentPerUser
”18-24”	9505	616204.0050523246	64.82945871144919
”45-54”	13507	876656.9810595925	64.903900278344
”35-44”	13654	879153.417123852	64.38797547413593
”25-34”	13486	882255.4374447308	65.42009768980652
”55-64”	13630	896658.8652415214	65.78568343664867
”65+”	16262	1057426.51384831	65.02438284640942

4.3.2 Top Product Categories Purchased by Gender in the Selected Segment

This query identifies the top 5 product categories with the highest purchase frequency for each gender (male and female) for the users belonging to the 65+ age group. This analysis can provide insights into the preferences and behaviors of the older demographic, helping tailor marketing strategies to better target this segment.

```
MATCH (u:User)-[:PLACES]->(o:Order)-[:CONTAINS]->(oi:OrderItem)-[:REFERS_TO]->
WHERE u.age >= 65
WITH u.gender AS gender, p.category AS product_category, COUNT(*) AS frequency
// Raggruppa per genere e categoria di prodotto
WITH gender, product_category, frequency
ORDER BY frequency DESC
// Limita il numero di risultati per ciascun gruppo
WITH gender, COLLECT({category: product_category, frequency: frequency}) AS ca
WITH gender, categories[0..5] AS top_categories
UNWIND top_categories AS top_category
RETURN gender, top_category.category AS product_category, top_category.frequen
ORDER BY gender, frequency DESC;
```

Table 15: Top 5 Product Categories per Gender

Gender	ProductCategory	Frequency
”F”	”Intimates”	1349
”F”	”Swim”	544
”F”	”Dresses”	539
”F”	”Fashion Hoodies & Sweatshirts”	536
”F”	”Maternity”	524
”M”	”Underwear”	786
”M”	”Tops & Tees”	773
”M”	”Jeans”	771
”M”	”Pants”	726
”M”	”Fashion Hoodies & Sweatshirts”	692

5 References & Sources

[1] Course Slides

[2] https://pysimplegui.readthedocs.io/en/latest/call

[3] https://py2neo.org/

[4] https://neo4j.com/docs/cypher-manual/current/

[5] https://neo4j.com/developer/python/

[6] http://iniball.altervista.org/Software/ProgER

[7] https://neo4j.com/developer/cypher/

[8] https://pandas.pydata.org/docs/