Part 1

1) Data Source

Data Source: Brazilian E-Commerce Public Dataset by Olist (Kaggle)

Link: https://www.kaggle.com/datasets/olistbr/brazilian-ecommerce

The Brazilian E-Commerce Public Dataset by Olist, available on Kaggle, provides a comprehensive collection of data from 100,000 orders made between 2016 and 2018 across various marketplaces in Brazil. It offers insights into order details, customer behavior, and market trends. This dataset is a valuable resource for analyzing the Brazilian e-commerce landscape, customer satisfaction, and supply chain management.

2) Create a scenario for the data

Scenario: Olist Store's E-commerce Performance Analysis

Background:

Olist Store is a leading e-commerce platform operating in Brazil, connecting customers with a diverse range of products through various marketplaces. With a focus on delivering a seamless shopping experience, Olist Store strives to optimize order fulfillment, customer satisfaction, and business performance.

Organizational Structure:

Olist Store follows a structured organizational setup, comprising departments responsible for different product categories, order processing, customer service, and logistics. These departments include Electronics, Home Appliances, Fashion, Beauty & Personal Care, Books & Media, and more. Each department manages its inventory, ensures timely deliveries, and fosters customer relationships.

Dataset Description:

The dataset for Olist Store's e-commerce performance analysis consists of six tables, each providing specific information about different aspects of the business. The tables and their column names are as follows:

Table: Customers

Column Name	Description
customer_id	A unique identifier for each customer.
customer_unique_id	A unique identifier that represents each customer in a unique way. It distinguishes individual customers even if they have made multiple purchases or have the same name.
customer_zip_code_prefix	The prefix of the customer's zip code. In Brazil, the zip code is composed of 8 digits, with the first 5 digits representing the city/region and the last 3 digits representing the specific area within the city/region.
customer_city	The city where the customer is located.
customer_state	The state where the customer is located. In Brazil, state names are represented by their two-letter abbreviations (e.g., SP for São Paulo, RJ for Rio de Janeiro).

Table: Order items

Column Name	Description
order_id	A unique identifier for each order. It represents a specific transaction made by a customer.
order_item_id	A unique identifier for each item within an order. Since an order can contain multiple items, this column distinguishes each item within the order.
product_id	A unique identifier for each product. It represents a specific product that is part of an order.
seller_id	A unique identifier for each seller. It represents the seller or merchant who is responsible for selling the product.
shipping_limit_	The date and time by which the order should be shipped. It represents the

date	deadline provided for the seller to ship the product.
price	The price of the product. It represents the cost of the item before any additional charges or discounts.
freight_value	The shipping cost associated with the item. It represents the additional amount charged for shipping the product to the customer.

Table: Order payments

Column Name	Description
order_id	A unique identifier for each order. It represents a specific transaction made by a customer.
payment_sequential	The sequential number assigned to each payment within an order. Since an order can have multiple payments, this column indicates the order in which the payments were made.
payment_type	The type of payment used for the order. It represents the method or channel through which the customer made the payment.
payment_installments	The number of installments or payment terms chosen by the customer. It represents the division of the total payment amount into multiple smaller amounts, typically used for credit card payments.
payment_value	The value or amount of the payment made for the order. It represents the total amount paid by the customer, including the product price and any additional charges like shipping or taxes.

Table: Sellers

Column Name	Description
seller_id	A unique identifier for each seller. It represents the seller or merchant who is responsible for selling products.
seller_zip_code_prefix	The prefix of the seller's zip code. In Brazil, the zip code is composed of 8 digits, with the first 5 digits representing the city/region and the last 3 digits representing the specific area within the city/region. This column represents the prefix part of the zip code for the seller's location.
seller_city	The city where the seller is located.
seller_state	The state where the seller is located. In Brazil, state names are represented by their two-letter abbreviation (e.g., SP for São Paulo, RJ for Rio de

	Janeiro). This column indicates the state of the seller's location.

Table: Products

Column Name	Description
product_id	A unique identifier for each product. It represents a specific product available for sale in the e-commerce platform.
product_category_name	The name of the product category to which the product belongs. It represents the broad category or classification of the product.
product_photos_qty	The number of photos available for the product. It represents the quantity of product images associated with the product listing.
product_weight_g	The weight of the product in grams. It represents the mass or heaviness of the product.
product_length_cm	The length of the product is centimeters. It represents the linear measurement of the product in the longest dimension.
product_height_cm	The height of the product is centimeters. It represents the vertical measurement of the product.
product_width_cm	The width of the product is centimeters. It represents the horizontal measurement of the product.

Table: Orders

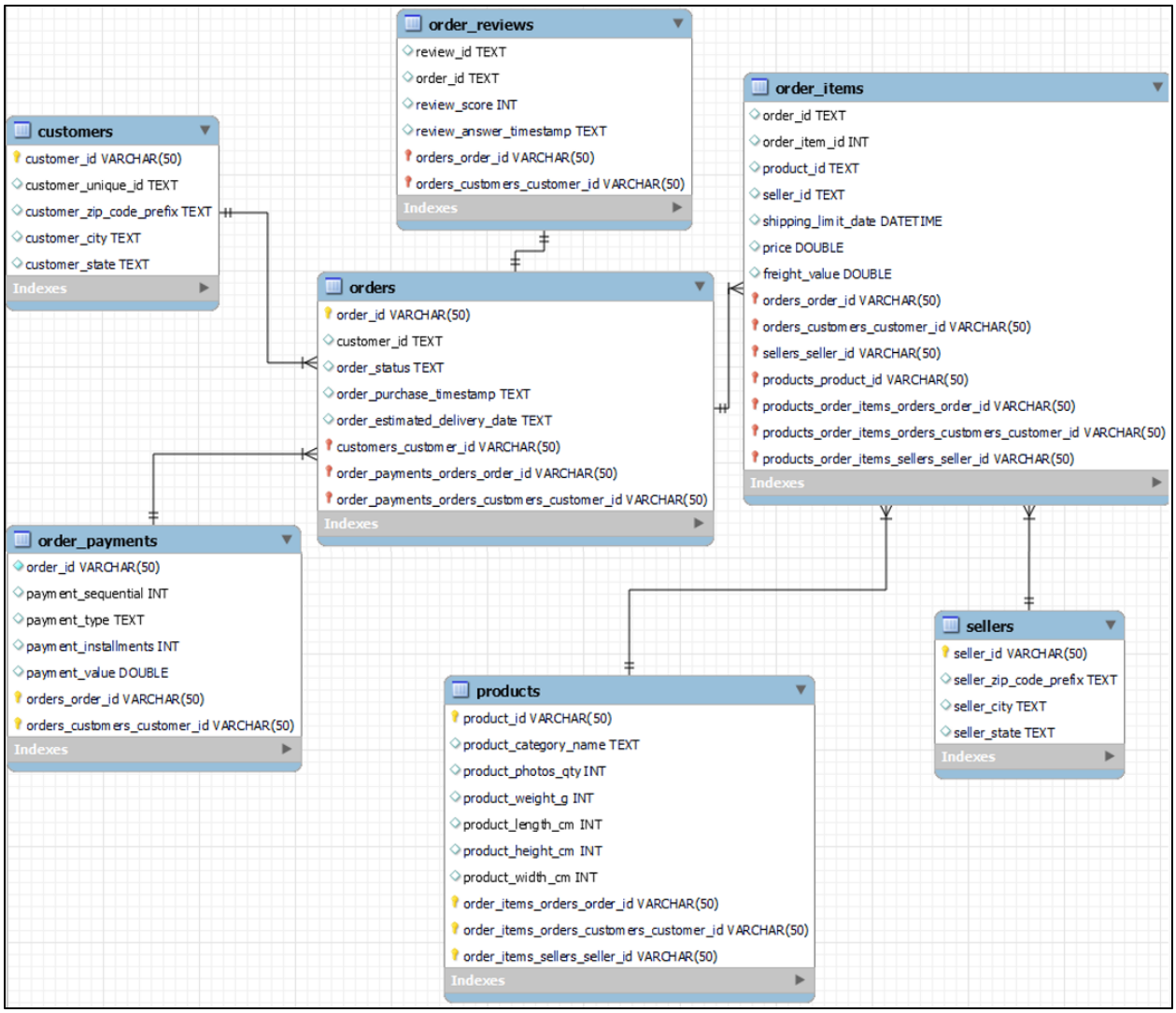
Column Name	Description
order_id	A unique identifier for each order. It represents a specific transaction made by a customer.
customer_id order_status	A unique identifier for each customer. It represents the customer who placed the order. The status of the order. It represents the current state of the order, such as "delivered," "shipped," "in progress," or "canceled."
order_purchase_timestamp	The timestamp indicates when the order was placed. It represents the date and time when the customer initiated the purchase.
order_estimated_delivery_date	The estimated delivery date for the order. It represents the expected or predicted date by which the order should be delivered to the customer.

Next, the following are performance metrics and insights that can be used to represent the performance of a business:

1) Customer Segmentation based on Purchasing Behavior:
    a) By analyzing the number of orders and total spent by each customer, distinct customer segments can be identified.
    b) These segments can provide insights into different customer behaviors and help tailor marketing strategies accordingly.
2) Top Selling Products and Revenue Generation:
    a) The top 10 products with the highest sales and revenue can be identified.
    b) This information helps in understanding the product categories that contribute significantly to the overall revenue of the business.
3) Popular Payment Types:
    a) The most commonly used payment types by customers can be determined.
    b) This knowledge helps in optimizing payment options and improving the overall customer experience.
4) Untapped Geographic Markets with High Customer Potential:
    a) Regions or cities with high customer potential but relatively low customer counts can be identified.
    b) These areas present opportunities for business expansion and targeted marketing efforts.
5) Top Sellers by Revenue:
    a) The top 10 sellers with the highest revenue can be determined.
    b) This information can help in evaluating seller performance and building strategic partnerships.
6) Geographic Market Potential:
    a) Regions or cities with high customer potential can be identified based on customer counts.
    b) This information helps target specific geographic areas for marketing campaigns or expansion opportunities.

These performance metrics and insights provide valuable information for understanding customer behavior, optimizing marketing strategies, improving revenue generation, and identifying growth opportunities for the business.

3) Identify possible relations



customers (customer_id, customer_unique_id, customer_zip_code_prefix, customer_city, customer_state) **Primary key** customer_id	**orders** (order_id, customer_id, order_status, order_purchase_timestamp, order_estimated_delivery_date) **Primary key** order_id **Foreign key** customer_id **reference**

Alternate key customer_unique_id	customer(cutomer_id)
order_items (order_id, order_item_id, product_id, seller_id, shipping_limit_date, price, freight_value) **Foreign key** order_id **reference** orders(order_id)	**products** (product_id, product_category_name, product_photos_qty, product_weight_g, product_length_cm, product_height_cm, product_width_cm) **Primary key** product_id
order_payments (order_id, payment_sequential, payment_type, payment_installments payment_value) **Foreign key** order_id **reference** orders(order_id)	**Seller** (seller_id, seller_zip_code_prefix, seller_city, seller_state) **Primary key** seller_id
order_reviews (review_id, order_id, review_score, review_answer_timestamp) **Primary key** review_id **Foreign key** order_id **reference** orders(order_id)	

4) SQL queries and Index

1) Identify distinct customer segments based on their purchasing behaviour

```
CREATE INDEX idx_customers_unique_id ON customers (customer_unique_id);

SELECT c.customer_unique_id, COUNT(DISTINCT o.order_id) AS total_orders,
SUM(op.payment_value) AS total_spent
FROM customers AS c
INNER JOIN orders AS o ON c.customer_id = o.customer_id
INNER JOIN order_payments AS op ON o.order_id = op.order_id;
```

customer_unique_id	total_orders	total_spent
0000366f3b9a7992bf8c76cfdf3221e2	1	141.9
0000b849f77a49e4a4ce2b2a4ca5be3f	1	27.19
0000f46a3911fa3c0805444483337064	1	86.22
0000f6ccb0745a6a4b88665a16c9f078	1	43.62
0004aac84e0df4da2b147fca70cf8255	1	196.89
0004bd2a26a76fe21f786e4fbd80607f	1	166.98
00050ab1314c0e55a6ca13cf7181fecf	1	35.38
00053a61a98854899e70ed204dd4bafe	1	419.18
0005e1862207bf6ccc02e4228effd9a0	1	150.12
0005ef4cd20d2893f0d9fbd94d3c0d97	1	129.76
0006fdc98a402fceb4eb0ee528f6a8d4	1	29
00082cbe03e478190aadbea78542e933	1	126.26
00090324bbad0e9342388303bb71ba0a	1	63.66

2) The top 10 products with the highest revenue?

```
SELECT p.product_category_name, SUM(op.payment_value) AS total_revenue
FROM products AS p
INNER JOIN order_items AS oi ON p.product_id = oi.product_id
INNER JOIN orders AS o ON oi.order_id = o.order_id
INNER JOIN order_payments AS op ON o.order_id = op.order_id
GROUP BY p.product_id, p.product_category_name
ORDER BY total_revenue DESC
limit 10;
```

product_category_name	total_revenue
watches_gifts	26709.65
cool_stuff	23713.26
bed_bath_table	13511.94
garden_tools	12457.16
sports_leisure	10860.9
health_beauty	8985.95
computers_accessories	7765.39
computers	7010.05
furniture_decor	6738.5
auto	6537.51

3) Which payment types are most commonly used by customers?

```
SELECT payment_type, COUNT(DISTINCT order_id) AS total_orders
FROM order_payments
GROUP BY payment_type
ORDER BY total_orders DESC;
```

payment_type	total_orders
credit_card	76505
boleto	19784
voucher	3866
debit_card	1528
not_defined	3

4) Are there untapped geographic markets or regions with high customer potential?

SELECT customer_city, customer_state, COUNT(DISTINCT customer_unique_id) AS customer_count
FROM customers
GROUP BY customer_city, customer_state
ORDER BY customer_count DESC;

customer_city	customer_state	customer_count
sao paulo	SP	14984
rio de janeiro	RJ	6620
belo horizonte	MG	2672
brasilia	DF	2069
curitiba	PR	1465
campinas	SP	1398
porto alegre	RS	1326
salvador	BA	1209
guarulhos	SP	1153
sao bernardo do campo	SP	908
niteroi	RJ	811
santo andre	SP	768
osasco	SP	717
santos	SP	692
goiania	GO	671
sao jose dos campos	SP	666
fortaleza	CE	643
sorocaba	SP	610
recife	PE	590
jundiai	SP	547
florianopolis	SC	546
ribeirao preto	SP	489
belem	PA	432
nova iguacu	RJ	432
barueri	SP	419
contagem	MG	417
juiz de fora	MG	415
sao goncalo	RJ	399

5) Top 10 sellers with the highest revenue

```
SELECT s.seller_id, SUM(op.payment_value) AS total_revenue
FROM sellers AS s
INNER JOIN order_items AS oi ON s.seller_id = oi.seller_id
INNER JOIN order_payments AS op ON oi.order_id = op.order_id
GROUP BY s.seller_id
ORDER BY total_revenue DESC
LIMIT 10;
```

seller_id	total_revenue
7e93a43ef30c4f03f38b393420bc753a	185134.21
7d13fca15225358621be4086e1eb0964	129169.98
ccc4bbb5f32a6ab2b7066a4130f114e3	84993.28
37be5a7c751166fbc5f8ccba4119e043	65183.5
7142540dd4c91e2237acb7e911c4eba2	47434.8
7299e27ed73d2ad986de7f7c77d919fa	47194.54
7178f9f4dd81dcef02f62acdf8151e01	43669.22
9f505651f4a6abe901a56cdc21508025	36831.1
a3a38f4affed601eb87a97788c949667	36152.66
a416b6a846a11724393025641d4edd5e	32997.9

6) Geographic Market Potential: Identify regions or cities with high customer potential based on customer count.

```
SELECT customer_city, customer_state, COUNT(DISTINCT customer_unique_id) AS customer_count
FROM customers
GROUP BY customer_city, customer_state
ORDER BY customer_count DESC;
```

customer_city	customer_state	customer_count
sao paulo	SP	14984
rio de janeiro	RJ	6620
belo horizonte	MG	2672
brasilia	DF	2069
curitiba	PR	1465
campinas	SP	1398
porto alegre	RS	1326
salvador	BA	1209
guarulhos	SP	1153
sao bernardo do campo	SP	908
niteroi	RJ	811
santo andre	SP	768
osasco	SP	717
santos	SP	692
goiania	GO	671
sao jose dos campos	SP	666
fortaleza	CE	643
sorocaba	SP	610
recife	PE	590
jundiai	SP	547
florianopolis	SC	546
ribeirao preto	SP	489
belem	PA	432
nova iguacu	RJ	432
barueri	SP	419
contagem	MG	417

7) The total revenue by each year (2016-2018) in descending order

CREATE INDEX idx_order_purchase_timestamp ON orders (order_purchase_timestamp);

SELECT YEAR(STR_TO_DATE(o.order_purchase_timestamp, '%d/%m/%Y')) AS purchase_year, CONCAT('$', FORMAT(SUM(op.payment_value), 2)) AS total_revenue
FROM orders o
JOIN order_payments op ON o.order_id = op.order_id
WHERE STR_TO_DATE(o.order_purchase_timestamp, '%d/%m/%Y') BETWEEN '2016-01-01' AND '2018-12-31'
GROUP BY purchase_year
ORDER BY purchase_year DESC;

purchase_year	total_revenue
2018	$2,864,928.58
2017	$2,392,873.94
2016	$19,566.35

Part 2

1) Converting tables to documents in MongoDB

To establish the relationships between the collections based on the given relations, the data can be structured as follows:

Collection: "orders"

- Embed the core order information: order_id, order_status, order_purchase_timestamp, order_estimated_delivery_date.
- Include an embedded document for order_reviews with fields: review_id, review_score, review_answer_timestamp.
- Include an embedded document for customer with fields: customer_id, customer_unique_id, customer_zip_code_prefix, customer_city, customer_state.
- Include an array of embedded documents for order_items with fields: order_item_id, product_id, seller_id, shipping_limit_date, price, freight_value.

Collection: "products"

- Embed the product information: product_id, product_category_name, product_photos_qty, product_weight_g, product_length_cm, product_height_cm, product_width_cm.
- Include an array of embedded documents for order_items with fields: order_item_id, order_id, seller_id, shipping_limit_date, price, freight_value.

Collection: "sellers"

- Embed the seller information: seller_id, seller_zip_code_prefix, seller_city, seller_state.
- Include an array of embedded documents for order_items with fields: order_item_id, order_id, product_id, shipping_limit_date, price, freight_value.

Code:

```
use project
db.orders.find().forEach(function(order) {
  var orderReviews = db.order_reviews.findOne({ order_id: order.order_id });
  var customer = db.customers.findOne({ customer_id: order.customer_id });
```

```
  var orderPayments = db.order_payments.find({ order_id: order.order_id }).toArray();
  var orderItems = db.order_items.find({ order_id: order.order_id }).toArray();

  db.orders.updateOne(
    { _id: order._id },
    {
      $set: {
        order_reviews: orderReviews,
        customer: customer,
        order_payments: orderPayments,
        order_items: orderItems
      }
    }
  );
});

db.products.find().forEach(function(product) {
  var orderItems = db.order_items.find({ product_id: product.product_id }).toArray();

  db.products.updateOne(
    { _id: product._id },
    {
      $set: {
        order_items: orderItems
      }
    }
  );
});

db.sellers.find().forEach(function(seller) {
  var orderItems = db.order_items.find({ seller_id: seller.seller_id }).toArray();

  db.sellers.updateOne(
    { _id: seller._id },
    {
      $set: {
        order_items: orderItems
      }
    }
  );
});
```

With this structure, the "orders" collection contains all the relevant information about an order, including order reviews, customer details, and order items. The "products" collection includes

product-related information along with the embedded order items, and the "sellers" collection contains seller-related information along with the embedded order items.

The chosen structure incorporates arrays and embedded documents to establish relationships between the collections. By embedding relevant information such as order reviews, customer details, and order items within the orders collection, and utilizing arrays to represent one-to-many relationships between products/sellers and order items, the data model enables efficient retrieval of related data and simplified querying.

By utilizing embedded documents and arrays, the relationships between the collections are established, enabling efficient retrieval of related data and simplified querying. This structure facilitates easy navigation through the data hierarchy and access to the required information across different entities.

2) MongoDB Queries

1) Identify distinct customer segments based on their purchasing behavior

```
{
  path: "$order_payments",
}
{
  _id: {
    customer_unique_id:
      "$customer.customer_unique_id",
  },
  total_order: {
    $addToSet: "$order_id",
  },
  total_spent: {
    $sum: "$order_payments.payment_value",
  },
}
{
  _id: 0,
  customer_unique_id: "$_id.customer_unique_id",
  total_order: {
    $size: "$total_order",
```

```
  },
  total_spent: "$total_spent",
}
{
  total_order: -1,
}
```

2. The top 10 products with the highest revenues?

```
 [
  {
    $unwind: {
      path: "$order_items",
    },
  },

  {
    $group: {
      _id: "$product_category_name",
      total_revenue: {
        $sum: {
          $subtract: [
            "$order_items.price",
            "$order_items.freight_value",
          ],
        },
      },
    },
  },
  {
    $project: {
      product_id: 1,
      totalrevenue: "$total_revenue",
    },
  },
  {
    $sort: {
      totalrevenue: -1,
    },
  },
]
```

3) Which payment types are most commonly used by customers?

```
[
 {
  $unwind: {
   path: "$order_payments",
  },
 },
 {
  $group: {
   _id: "$order_payments.payment_type",
   total_orders: {
    $addToSet: "$order_id",
   },
  },
 },
 {
  $project: {
   _id: 1,
   total_orders: {
    $size: "$total_orders",
   },
  },
 },
 {
  $sort: {
   total_orders: -1,
  },
 },
```

```
]
```

4)Are there untapped geographic markets or regions with high customer potential?

```
[
  {
    $group: {
      _id: {
        customer_city: "$customer.customer_city",
        customer_state:
          "$customer.customer_state",
      },
      customer_count: {
        $addToSet: "$customer.customer_unique_id",
      },
    },
  },
  {
    $project: {
      _id: 0,
      customer_city: "$_id.customer_city",
      customer_state: "$_id.customer_state",
      customer_count: {
        $size: "$customer_count",
      },
    },
  },
  {
    $sort: {
      customer_count: -1,
```

```
    },
  },
]
```

5) Top 10 sellers with the highest revenue

```
[
  {
    $unwind: {
      path: "$order_items",
    },
  },
  {
    $group: {
      _id: "$seller_id",
      total_revenue: {
        $sum: {
          $subtract: [
            "$order_items.price",
            "$order_items.freight_value",
          ],
        },
      },
    },
  },
  {
    $project: {
      seller_id: "$_id",
      totalrevenue: "$total_revenue",
```

```
    },
  },
  {
    $sort: {
      totalrevenue: -1,
    },
  },
]
```

6) Geographic Market Potential: Identify regions or cities with high customer potential based on customer count

```
[
  {
    $group: {
      _id: {
        customercity: "$customer.customer_city",
        customerstate: "$customer.customer_state",
      },
      customer_count: {
        $addToSet: "$customer.customer_unique_id",
      },
    },
  },
  {
    $project: {
      _id: 0,
      customer_city: "$_id.customercity",
      customer_state: "$_id.customerstate",
      customer_count: {
        $size: "$customer_count",
      },
    },
  },
  {
    $sort: {
      customer_count: -1,
    },
  },
]
```

3) Applicability of data model in case study

Based on the provided case study of the Brazilian E-Commerce dataset, a relational data model is more applicable compared to document-based data model. While a document-based data model, such as NoSQL databases, can offer flexibility and scalability advantages in certain scenarios, the structured nature of the provided dataset and the need for maintaining relationships and ensuring data integrity make a relational data model a more suitable choice for the given case study. Here's the justification for this choice:

a) Structured and Tabular Data: The dataset consists of structured and tabular data with well-defined relationships between entities such as customers, orders, products, and sellers. The relational data model is designed to handle structured data and allows for efficient querying, joining, and analyzing data based on predefined relationships.
b) Data Consistency and Integrity: The relational data model enforces referential integrity through primary key and foreign key constraints. This ensures that the relationships between entities are maintained and that data remains consistent and reliable. It helps in preventing orphaned records and maintaining data integrity across tables.
c) Querying Flexibility: Relational databases provide a standardized query language (SQL) that allows for complex and flexible querying capabilities. This is beneficial for performing various analysis tasks and generating meaningful insights from the dataset. SQL provides powerful tools for filtering, aggregating, and combining data from multiple tables, which aligns well with the analytical requirements of the case study.
d) Scalability and Performance: Relational databases are well-suited for handling large volumes of structured data efficiently. They offer indexing, query optimization, and transaction management mechanisms that optimize performance. Given the size of the dataset is about 100k orders and the potential growth over time, a relational data model can handle the scalability requirements effectively.
e) Data Integrity Constraints: Relational databases allow the enforcement of data integrity constraints, such as unique constraints and check constraints, to ensure the validity of data. This is particularly important for maintaining data quality and consistency in an e-commerce context where accurate and reliable data is crucial for business operations.