DM Analysis

1. Introduction

E-Commerce websites have become a popular strategy for many businesses to seamlessly market and sell their products to customers. It is crucial that these websites operate smoothly to captivate more customers and attain competitive advantage and the platforms are heavily reliant on proper data management. This report discusses in detail the data management aspects of "We-Buy" a simulated E-Commerce environment from database design to analysis and reporting.

2. Database Design and Implementation

The Mini world of E-Commerce is a digitized platform where customers with an online profile can view and buy products online. The products will be sold by designated sellers and dispatched and distributed to the customers when they make a purchase and complete payment.

2.1. Requirement Collection and Analysis

2.1.1. Assumptions for ER Diagram

Customer:

- One customer must have at least one address and multiple orders can be delivered to one address.
- The email of customers must be unique.

Orders:

- One order can have multiple products.
- The 'sub_quantity' in the order details shows the number of each product separately.

- If an order is delivered, then it has only 1 transaction.

Transaction:

- One transaction match with one delivery.
- Meanwhile, if the order status is 'Pending',' Processing' or 'Cancelled', there is no transaction matched with it. If the order status is 'Succeed', it matches with one transaction record.
- Every transaction can have multiple products.

Delivery:

- One delivery matches one customer address at one time.
- One delivery can only be linked with one transaction at a time. Also, If the transaction status is 'Pending',' Processing' or 'Cancelled', there is no delivery. If the transaction status is 'Succeed', it matches with one delivery record.

Product:

- One product belongs to one category.
- One product can be supplied by multiple suppliers.
- One product can have 0 or 1 or many advertisements.
- One product can be in multiple transactions and multiple orders.
- The rate value is between 0 to 5.

Category:

- One category can have many products.
- One parent category can have multiple sub-categories.

Supply:

- This is a relationship between suppliers and products.
- Stock cannot be empty.

Supplier:

- One supplier can supply multiple products.
- The email of each supplier should be unique.

Ads:

- One ad can only match one product.
- If the ad is 'active' which means that the last day to show the ad is after today. If the ad is 'ended', it means that the last day was before today. All advertisements have a start date.

2.1.2. Logic Design

- customer (customer_id(PK), first_name, last_name, email, password_hash, gender, membership, date_of_birth, customer_address (zip_code, country, state, city, street))
- order(order_id (PK), customer_id (FK), product_id(FK), order_date, order_status, order_details(sub_quantity))
- transactions (transaction_id(PK), order_id(FK), customer_id(FK), transaction_time, transaction_status, payment_method)
- delivery (delivery_id(PK), transaction_id(FK), customer_address_id(FK), delivery_start_sate, delivery_end_date, delivery_status)
- product (product_id (PK), category_id(FK), product_price, product_name, discount_percentage, rate_value)
- supply (supply_id(PK), supplier_id(FK), product_id(FK), stock)
- supplier (supplier_id(PK), supplier_email, supplier_name, supplier_street, supplier_city, supplier_state, supplier_country, supplier_zip_code)
- ads (ad_id(PK), product_id(FK), ad_start_date, ad_end_date, ad_status))
- category (category_id(PK), parent_category_id(FK), category_type)

2.2. Data Relationships

One to One relationship

- Entity: Customer and Transaction
- Relationship: A Customer can have only one transaction at a time, and one transaction will be entered in to by one customer.

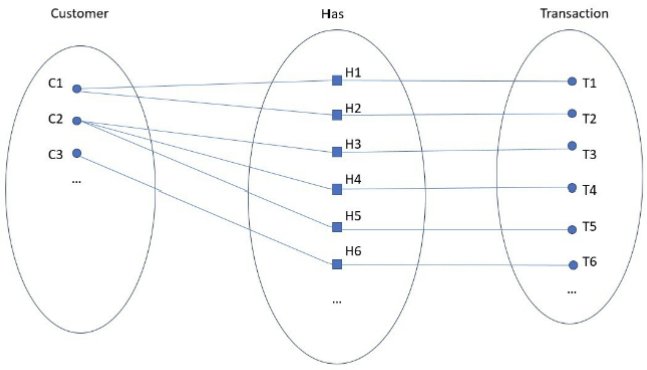


Figure 1: Customer - Transaction relationship

- Entity: Transaction and Delivery
- Relationship: A transaction will have one delivery, and each delivery will begin with one transaction

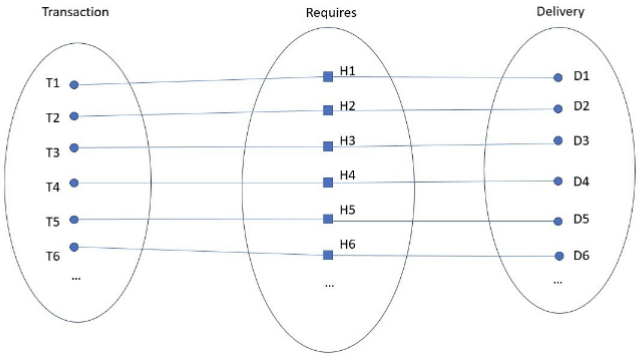


Figure 2: Transaction - Delivery relationship

One to Many relationship

- Entity: Transaction and Product
- Relationship: Each transaction can have many products, but each product will only belong to one transaction

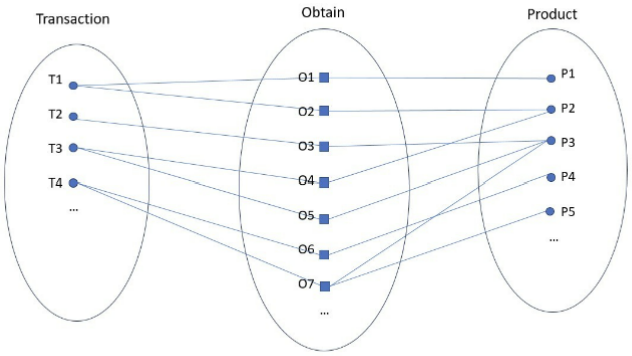


Figure 3: Transaction - Product relationship

- Entity: Product and Ads
- Relationship: Each product can have many Ads, but each Ad will only belong to one product

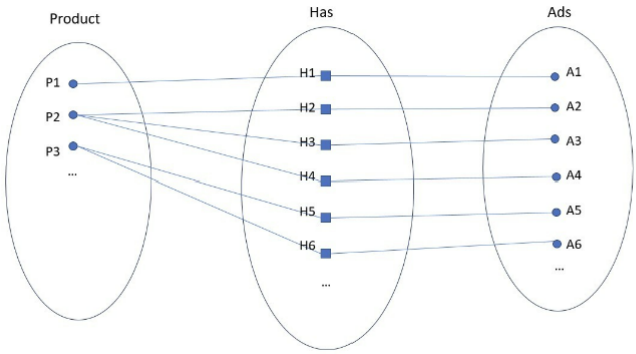


Figure 4: Product - Ad relationship

- Entity: Category and Product
- Relationship: Each category will have many products, but each product will only belong to one category

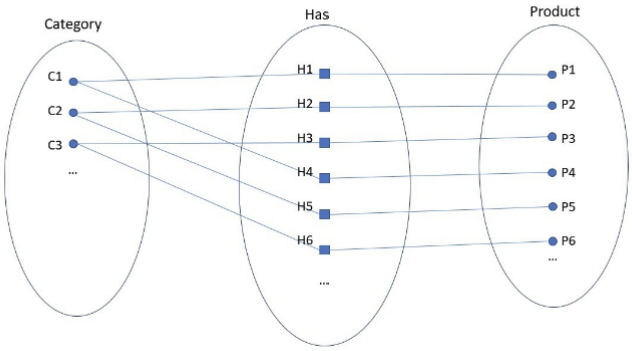


Figure 5: Category - Product relationship

- Entity: Category and Category
- Relationship: Each parent category will have many subcategories, but each subcategory will only belong to one parent category

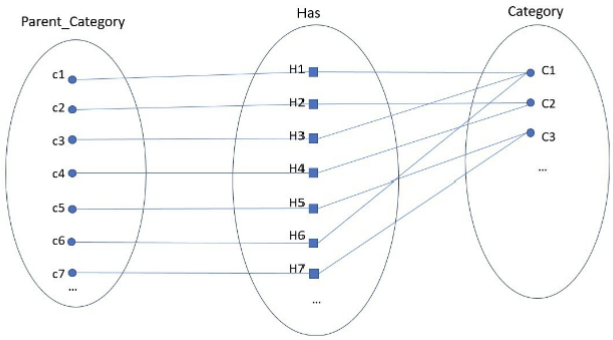


Figure 6: Category - Category relationship

Many to Many relationship

- Entity: Customer and Product
- Relationship: A customer can have multiple products, and each product can be bought by multiple customers

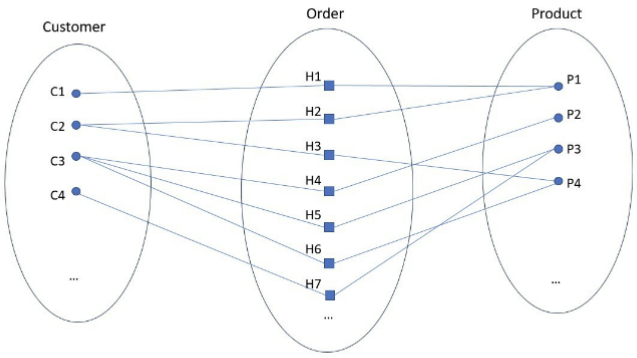


Figure 7: Customer - Product relationship

- Entity: Product and Supplier
- Relationship: A product can have many suppliers, and each supplier can have many products.

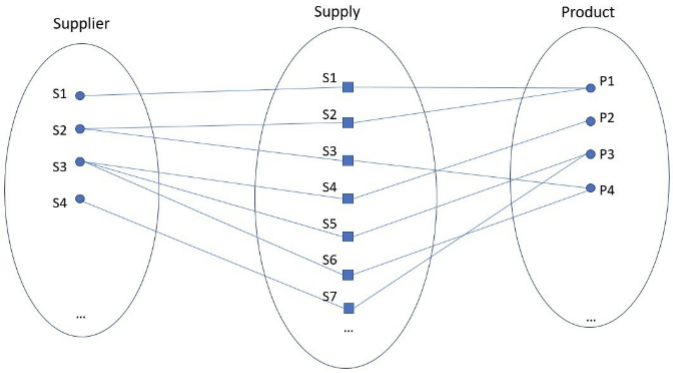


Figure 8: Supplier - Product relationship

2.3. ER Diagram Design

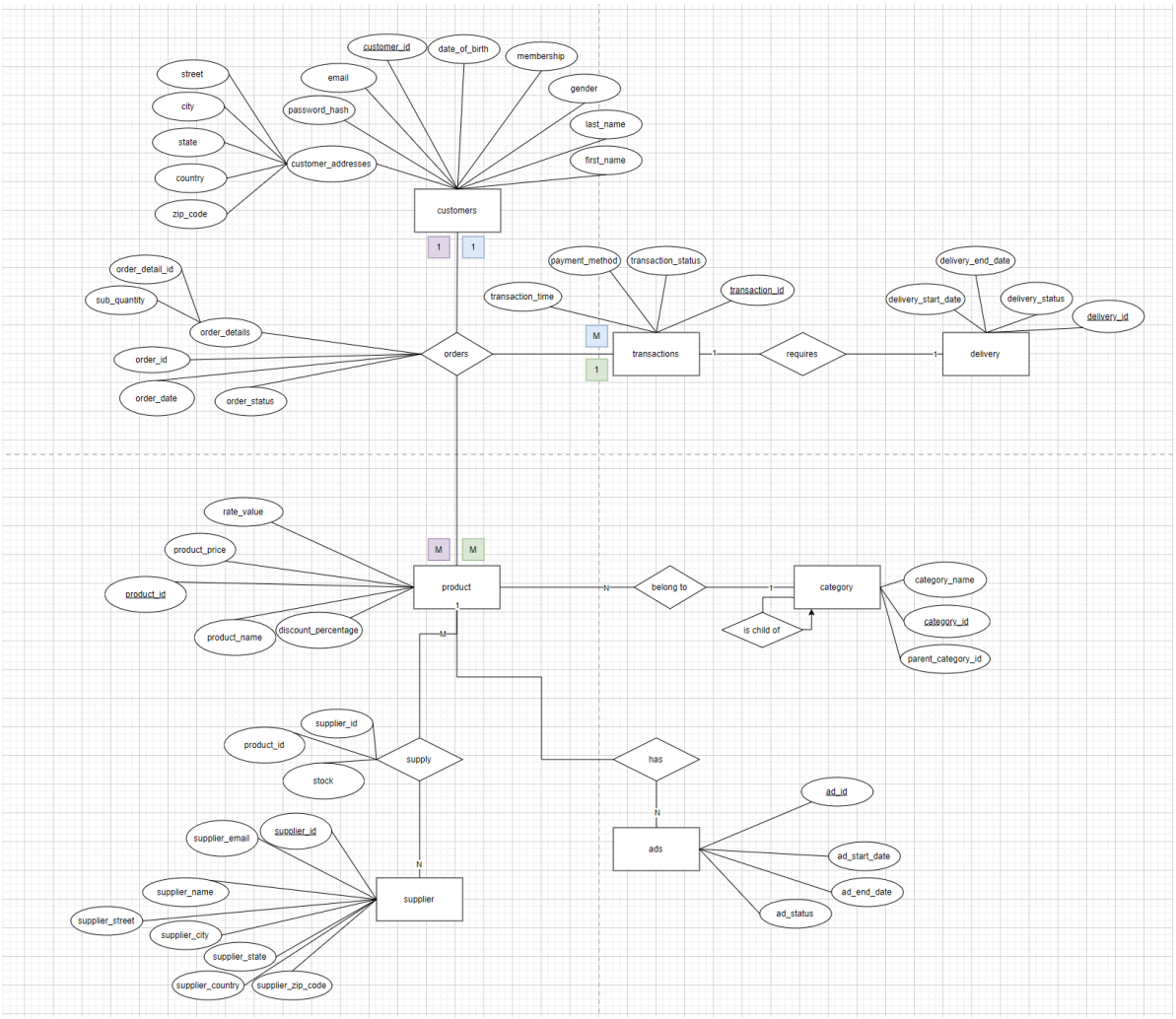


Figure 9: ER Diagram

2.4. SQL Database Design

The Figure 10 depicts theentity relationship diagramfor the database that includes various tables: Customer, Orders, Transactions, Delivery, Product, Ads, Supply, Supplier, and Category. These tables store information based on the relationship indicating how the customer details, orders, products, suppliers, and advertisements are linked to managing inventory, order processing, and customer interactions. This is done to ensure seamless flow from customer order to delivery.

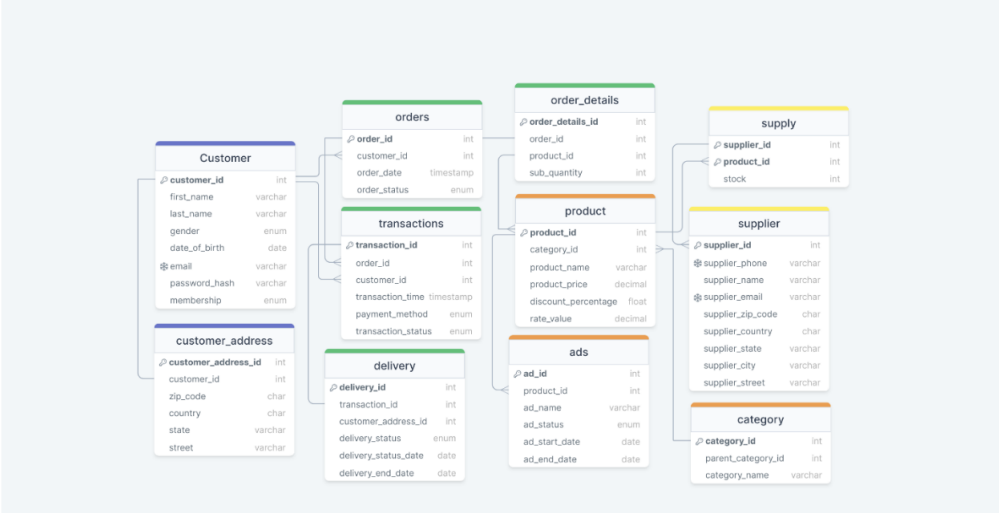


Figure 10: SQL design

3. Data generation and Management

3.1. Synthetic Data generation

Data generation methods were tested through Mockaroo, ChatGPT and Python. Mockaroo was rejected as the platform allows very less customization and the generated data were inconsistent and irrelevant. ChatGPT was rejected as the system did not respond and could not handle the specifications mentioned with the amount of data required. Python proved to be a suitable platform to generate consistent and accurate data and could handle the specifications and validations given. Faker, pandas, datetime, OS and random libraries were imported to the environment for data generation.

Several assumptions were made when generating the data.

Customer:

Customer is an important part of the data. With the help of customer data, we will be able to do necessary analysis, but there are certain conditions that should be met while creating the data for the customer.

A customer must have at least one address and multiple orders can be delivered to one.

One feature that is inspired by real world e-commerce companies is membership. Based on the membership, customers get a variety of rewards. In the case of We-Buy, the company offers discounted prices.

Order:

As an e-commerce company, it is important to understand how people are utilizing the service and what all products are being bought. This helps the company to understand what all the products that generate maximum revenue for the company.

This data should contain specific information about the order submitted by the customer. Information such as order status and order date should be considered. Order status has different values such as 'Pending', 'Processing', 'Cancelled', 'Submitted', and 'Succeed'. We must connect order data with customer information. So, customer id is also mentioned in this table so that while doing analysis, it is easy to retrieve information from customer table.

Supplier:

Suppliers play a vital role. The orders placed by customers are gone to suppliers. With this, suppliers can send the product to the respective customers. Hence suppliers are handling all the goods, it is important to store supplier information. So, geographical information such as address, and other contact information are collected and saved.

One assumption that is required for supplier is that one supplier can supply products to multiple customers.

To make sure that the customer gets the item, each supplier is required to note down the items available in stock. For this, product id is used to access the product table to get necessary information.

Product:

It is very important to capture information regarding the products. At the same time, there are several assumptions that are made to make sure that the data collected will help the company to deduce required information.

One product should be matched with one category, but at the same time, one category can have many products under it. So, it is important to make sure that the products are mapped accordingly.

It is also to be noted that ads can help with selling more products, but it is not mandatory that every product has ads. So, it is assumed that each product can have wide variety of ads or one or no ads. This information can help us understand whether ads will help to sell more products or not.

Delivery:

Since the products will be delivered to customers, it is sensible to capture the information regarding the delivery. This should include information such as delivery status, transaction id, and customer id.

There are few important assumptions that should be made while considering this information.

One major assumption is that if the difference between the delivery start date and delivery end date is 30 days, then the delivery is considered as failed. Both the delivery start date and delivery end date are spread across the last quarter of 2023 and first quarter of 2024.

Some of the other key points to note is regarding how the dates are allocated based on the delivery status.

Status	**Start Date**	**End Date**
Not Delivered	Empty	Empty
In Delivery	Not Empty	Empty
Complete	Not Empty	Not Empty
Failed	Not Empty	Not Empty

3.2. Data import and Quality assurance

It is crucial to check the quality of the data after it is imported. Especially, in this case, the data generated is synthetic data. Hence, it is even more reason to make sure if the data is correct or not. We must make sure of some of the basic information when it comes to data quality. For example, if there are two tables and both are connected by a foreign key, we must make sure that the data for foreign keys are taken correctly from the corresponding table. Also, every data will have missing values and other factors that can affect the quality. Hence, it is required to check and rectify the data if there are any discrepancies.

In R, we use str() and summary() to identify if the structure and datatypes of the data provided are not having any issues.

3.3. Data Validation Rules

For data validation, it is necessary to follow a structure. As every table holds key information, we must make sure that the data is correctly present. Below are the steps that should be followed while doing data validation.

- We need to check the primary keys are unique for every table and make sure that the foreign keys are mapped correctly.
- Next, we must check if there are NA values for the attributes mentioned in the schema.
- There are several data types involved in the tables. So, verifying that the data generated matches the data type mentioned in the schema is a crucial step.
- In the schema, for certain attributes, there are character limits mentioned. So, it is important to carry out checks to make sure that the character limits are not violated.

- Now, there can be information in specific format, for example, email. Email is used for both customers and suppliers. So, we must check that the emails are in the correct format.
- One of the important data when it comes to e-commerce business is date. Several entities have dates associated with it. So, it becomes a mandatory step to check all the date formats and make changes if there are any.
- In some cases, we need to check for conditions, and populate data based on those conditions, for example, yes or no. So, based on the schema, we must check and populate corresponding data with minimal to no issues.
- Email Validation: The email provided should have a valid email format (example@gmail.com)
- Entry Validation: Quantities entered must be a whole number
- Date Validation: Start date must always be before the end date
- Key Validation: All tables should have a Primary Key, and foreign key as required. Data types and data length should be as specified in the database

Business rules:

- Transactions with delivery start date and delivery end date more than 30 days will be failed
- Only successful orders can match with transactions
- Only successful transactions can match with delivery .

3.4 Data Storage and Retrieval

- Database structure: A relational database structure with tables, columns, primary and foreign key relationships.
- Data access methods: CRUD operations to enable efficient data retrieval and manipulation
- Indexing and Query performance: Frequently queried fields optimized to run efficiently

4. Data pipeline generation

4.1 GitHub repository and workflow setup

A repository was created through GitHub to centrally locate all information required for the data management of We-Buy. The repository was connected with R and push and pull functions synchronizing data flows to and from the repository were tested manually. All other information related such as the csv files containing the tables of the e-commerce data set, database file, quarto file, E-R diagram was uploaded to the repository for ease of use where all the members have access to the repository. Here,the required information that was necessary for the analysis was consolidated.

4.2 GitHub actions for continuous integration

We use GitHub Actions as a fundamental tool to automate processes in our analysis. By creating a general workflow in our repository, we automate the process of validating and analyzing the information with which our analysis is built. Likewise, we program the workflow so that the processes are repeated every hour, which allows us to constantly monitor the quality of the information being handled. However, the generated code is automatically activated by any push or pull. Through GitHub actions we guarantee to have control over the changes that are made to the information and processes that are carried out. Attached to this report are the access details of our repository and the code used to run the workflow.

5.Data analysis and Reporting

Initially, the database is created and connected. All tables are dropped to ensure there is no overlapping to maintain data integrity and organization. After the tables are dropped, new tables can be loaded and created, and the quality of the data is checked. The checked data is later written into the database and the database is checked.

Create and Connect Database

```
# Define the tables to drop
tables_to_drop <- c("ads", "category", "customer_address", "customer","delivery", "orders",
                    "supplier", "supply", "`transaction`")

# Connect to the database
my_connection <- RSQLite::dbConnect(RSQLite::SQLite(),"/cloud/project/database/an_e_commerce

# Drop each table
for (table_name in tables_to_drop) {
  dbExecute(my_connection, paste("DROP TABLE IF EXISTS", table_name))
```

```
}

# After dropping tables, you can proceed to create and load new tables
sql_commands <- readLines("Schema.sql", warn = FALSE)
for (sql_cmd in sql_commands) {
  tryCatch({
    dbExecute(my_connection, sql_cmd)
  }, error = function(e) {
    # Print error message without stopping the execution
    #cat("Error occurred: ", conditionMessage(e), "\n")
  })
}
```

Import Data

```
ads = readr::read_csv('/cloud/project/ecommerce_data/ads_data.csv', col_types = cols(ad_start
category = readr::read_csv('/cloud/project/ecommerce_data/category_data.csv')
```

```
Rows: 25 Columns: 3
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (3): category_id, category_name, parent_category_id

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
customer_address = readr::read_csv('/cloud/project/ecommerce_data/customer_addresses_data.csv
```

```
Rows: 205 Columns: 7
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (7): customer_address_id, customer_id, zip_code, country, state, city, s...

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
customer = readr::read_csv('/cloud/project/ecommerce_data/customers_data.csv', col_types = co
delivery = readr::read_csv('/cloud/project/ecommerce_data/delivery_data.csv', col_types = col
orders_details = readr::read_csv('/cloud/project/ecommerce_data/order_details_data.csv')
```

```
Rows: 1111 Columns: 4
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (3): order_detail_id, order_id, product_id
dbl (1): sub_quantity

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
orders = readr::read_csv('/cloud/project/ecommerce_data/orders_data.csv', col_types = cols(or
product = readr::read_csv('/cloud/project/ecommerce_data/product_data.csv')
```

```
Rows: 500 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (3): product_id, category_id, product_name
dbl (3): product_price, discount_percentage, rate_value

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
supplier = readr::read_csv('/cloud/project/ecommerce_data/supplier_data.csv')
```

```
Rows: 80 Columns: 8
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (8): supplier_id, supplier_name, supplier_email, supplier_zip_code, supp...

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
supply = readr::read_csv('/cloud/project/ecommerce_data/supply_data.csv')
```

```
Rows: 100 Columns: 4
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (3): supply_id, supplier_id, product_id
dbl (1): stock

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
transaction = readr::read_csv('/cloud/project/ecommerce_data/transactions_data.csv', col_type
```

Data Quality Check before writing into database

```
# Ads
str(ads)
```

```
spc_tbl_ [100 x 5] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ ad_id        : chr [1:100] "AD3411" "AD5591" "AD2023" "AD2393" ...
 $ product_id   : chr [1:100] "PR5955" "PR2980" "PR4455" "PR9243" ...
 $ ad_status    : chr [1:100] "active" "active" "active" "ended" ...
 $ ad_start_date: chr [1:100] "2023-09-04" "2023-12-04" "2023-06-18" "2023-08-04" ...
 $ ad_end_date  : chr [1:100] "2024-04-26" "2025-01-07" "2024-04-13" "2023-09-09" ...
 - attr(*, "spec")=
  .. cols(
  ..   ad_id = col_character(),
  ..   product_id = col_character(),
  ..   ad_status = col_character(),
  ..   ad_start_date = col_character(),
  ..   ad_end_date = col_character()
  .. )
 - attr(*, "problems")=<externalptr>
```

```
ads$ad_status = as.factor(ads$ad_status)
str(ads)
```

```
spc_tbl_ [100 x 5] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ ad_id        : chr [1:100] "AD3411" "AD5591" "AD2023" "AD2393" ...
 $ product_id   : chr [1:100] "PR5955" "PR2980" "PR4455" "PR9243" ...
 $ ad_status    : Factor w/ 2 levels "active","ended": 1 1 1 2 1 2 1 1 1 2 ...
 $ ad_start_date: chr [1:100] "2023-09-04" "2023-12-04" "2023-06-18" "2023-08-04" ...
 $ ad_end_date  : chr [1:100] "2024-04-26" "2025-01-07" "2024-04-13" "2023-09-09" ...
 - attr(*, "spec")=
  .. cols(
  ..   ad_id = col_character(),
  ..   product_id = col_character(),
  ..   ad_status = col_character(),
  ..   ad_start_date = col_character(),
  ..   ad_end_date = col_character()
```

```
  .. )
 - attr(*, "problems")=<externalptr>
```

```
summary(ads)
```

```
    ad_id            product_id         ad_status  ad_start_date
 Length:100         Length:100         active:45   Length:100
 Class :character   Class :character   ended :55   Class :character
 Mode  :character   Mode  :character               Mode  :character
 ad_end_date
 Length:100
 Class :character
 Mode  :character
```

```
# Category
# Missing value in parent_category_id is reasonable if category_id is parent_category_id its
str(category)
```

```
spc_tbl_ [25 x 3] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ category_id       : chr [1:25] "CT1001" "CT1001-208" "CT1001-546" "CT1001-865" ...
 $ category_name     : chr [1:25] "Clothing" "Shirts" "Pants" "Dresses" ...
 $ parent_category_id: chr [1:25] NA "CT1001" "CT1001" "CT1001" ...
 - attr(*, "spec")=
  .. cols(
  ..   category_id = col_character(),
  ..   category_name = col_character(),
  ..   parent_category_id = col_character()
  .. )
 - attr(*, "problems")=<externalptr>
```

```
summary(category)
```

```
 category_id        category_name      parent_category_id
 Length:25          Length:25          Length:25
 Class :character   Class :character   Class :character
 Mode  :character   Mode  :character   Mode  :character
```

```
# Customer_address
str(customer_address)
```

```
spc_tbl_ [205 x 7] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ customer_address_id: chr [1:205] "CA8785" "CA6746" "CA4295" "CA2484" ...
 $ customer_id        : chr [1:205] "CU3320" "CU3320" "CU3320" "CU8307" ...
 $ zip_code           : chr [1:205] "21639" "86282" "57281" "30961" ...
 $ country            : chr [1:205] "Greece" "Bosnia and Herzegovina" "Jersey" "Luxembourg"
 $ state              : chr [1:205] "Arizona" "Nebraska" "Washington" "West Virginia" ...
 $ city               : chr [1:205] "East Blakeborough" "East Emily" "New Douglas" "North Ke
 $ street             : chr [1:205] "76218 Catherine Freeway" "2956 Christina Inlet" "7829 E
 - attr(*, "spec")=
  .. cols(
  ..   customer_address_id = col_character(),
  ..   customer_id = col_character(),
  ..   zip_code = col_character(),
  ..   country = col_character(),
  ..   state = col_character(),
  ..   city = col_character(),
  ..   street = col_character()
  .. )
 - attr(*, "problems")=<externalptr>
```

```
customer_address$country = as.factor(customer_address$country)
customer_address$state = as.factor(customer_address$state)
customer_address$city = as.factor(customer_address$city)
str(customer_address)
```

```
spc_tbl_ [205 x 7] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ customer_address_id: chr [1:205] "CA8785" "CA6746" "CA4295" "CA2484" ...
 $ customer_id        : chr [1:205] "CU3320" "CU3320" "CU3320" "CU8307" ...
 $ zip_code           : chr [1:205] "21639" "86282" "57281" "30961" ...
 $ country            : Factor w/ 145 levels "Afghanistan",..: 53 16 71 82 125 28 57 26 62 1
 $ state              : Factor w/ 49 levels "Alabama","Alaska",..: 3 27 46 47 18 30 24 16 11
 $ city               : Factor w/ 203 levels "Aaronburgh","Adamburgh",..: 28 30 104 125 6 20
 $ street             : chr [1:205] "76218 Catherine Freeway" "2956 Christina Inlet" "7829 E
 - attr(*, "spec")=
  .. cols(
  ..   customer_address_id = col_character(),
  ..   customer_id = col_character(),
  ..   zip_code = col_character(),
  ..   country = col_character(),
  ..   state = col_character(),
  ..   city = col_character(),
  ..   street = col_character()
```

```
  .. )
 - attr(*, "problems")=<externalptr>
```

```
summary(customer_address)
```

```
 customer_address_id customer_id          zip_code
 Length:205          Length:205         Length:205
 Class :character    Class :character   Class :character
 Mode  :character    Mode  :character   Mode  :character
```

```
            country           state                city
 Barbados        :  4   New Jersey: 10   West Elizabeth:  2
 French Polynesia:  4   Hawaii    :  8   West Vincent  :  2
 Austria         :  3   Florida   :  7   Aaronburgh    :  1
 Guernsey        :  3   Missouri  :  7   Adamburgh     :  1
 Hungary         :  3   Tennessee :  7   Allenmouth    :  1
 India           :  3   Utah      :  7   Andersonberg  :  1
 (Other)         :185   (Other)   :159   (Other)       :197
    street
 Length:205
 Class :character
 Mode  :character
```

```
# Customer
str(customer)
```

```
spc_tbl_ [100 x 8] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ customer_id  : chr [1:100] "CU3320" "CU8307" "CU0170" "CU2873" ...
 $ first_name   : chr [1:100] "Brian" "Martin" "Kim" "Douglas" ...
 $ last_name    : chr [1:100] "Miller" "Christian" "Johns" "Tyler" ...
 $ gender       : chr [1:100] "female" "male" "female" "female" ...
 $ date_of_birth: chr [1:100] "1962-12-09" "1944-11-12" "1960-10-09" "1934-10-05" ...
 $ email        : chr [1:100] "robersonnancy@example.com" "ann32@example.org" "stewartshane@e
 $ password_hash: chr [1:100] "429b5dfe28a2710cee6b6ce1ffc456b8296bb1fe49df6dcb069737f6694c70
```

```
 $ membership   : chr [1:100] "yes" "yes" "no" "yes" ...
 - attr(*, "spec")=
  .. cols(
  ..   customer_id = col_character(),
  ..   first_name = col_character(),
  ..   last_name = col_character(),
  ..   gender = col_character(),
  ..   date_of_birth = col_character(),
  ..   email = col_character(),
  ..   password_hash = col_character(),
  ..   membership = col_character()
  .. )
 - attr(*, "problems")=<externalptr>
```

```
customer$gender = as.factor(customer$gender)
str(customer)
```

```
spc_tbl_ [100 x 8] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ customer_id  : chr [1:100] "CU3320" "CU8307" "CU0170" "CU2873" ...
 $ first_name   : chr [1:100] "Brian" "Martin" "Kim" "Douglas" ...
 $ last_name    : chr [1:100] "Miller" "Christian" "Johns" "Tyler" ...
 $ gender       : Factor w/ 3 levels "female","male",..: 1 2 1 1 2 3 3 1 1 1 ...
 $ date_of_birth: chr [1:100] "1962-12-09" "1944-11-12" "1960-10-09" "1934-10-05" ...
 $ email        : chr [1:100] "robersonnancy@example.com" "ann32@example.org" "stewartshane@e
 $ password_hash: chr [1:100] "429b5dfe28a2710cee6b6ce1ffc456b8296bb1fe49df6dcb069737f6694c70
 $ membership   : chr [1:100] "yes" "yes" "no" "yes" ...
 - attr(*, "spec")=
  .. cols(
  ..   customer_id = col_character(),
  ..   first_name = col_character(),
  ..   last_name = col_character(),
  ..   gender = col_character(),
  ..   date_of_birth = col_character(),
  ..   email = col_character(),
  ..   password_hash = col_character(),
  ..   membership = col_character()
  .. )
 - attr(*, "problems")=<externalptr>
```

```
summary(customer)
```

```
 customer_id         first_name         last_name            gender
```

```
 Length:100         Length:100         Length:100         female:40
 Class :character   Class :character   Class :character   male  :36
 Mode  :character   Mode  :character   Mode  :character   other :24
 date_of_birth         email           password_hash       membership
 Length:100         Length:100         Length:100         Length:100
 Class :character   Class :character   Class :character   Class :character
 Mode  :character   Mode  :character   Mode  :character   Mode  :character
```

```
# Delivery
# It's reasonable to have missing values for delivery_start_date and delivery_end_date in "Fa
str(delivery)
```

```
spc_tbl_ [10 x 6] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ delivery_id        : chr [1:10] "DE9143" "DE9770" "DE1870" "DE4792" ...
 $ transaction_id     : chr [1:10] "TR3416" "TR5378" "TR4534" "TR8649" ...
 $ customer_address_id: chr [1:10] "CA7294" "CA9915" "CA2167" "CA9655" ...
 $ delivery_status    : chr [1:10] "In_Delivery" "Completed" "Failed" "Completed" ...
 $ delivery_start_date: chr [1:10] "2023-09-22" "2024-02-25" "2023-11-22" "2023-09-02" ...
 $ delivery_end_date  : chr [1:10] NA "2024-03-12" "2023-12-22" "2023-09-04" ...
 - attr(*, "spec")=
  .. cols(
  ..   delivery_id = col_character(),
  ..   transaction_id = col_character(),
  ..   customer_address_id = col_character(),
  ..   delivery_status = col_character(),
  ..   delivery_start_date = col_character(),
  ..   delivery_end_date = col_character()
  .. )
 - attr(*, "problems")=<externalptr>
```

```
delivery$delivery_status = as.factor(delivery$delivery_status)
str(delivery)
```

```
spc_tbl_ [10 x 6] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ delivery_id        : chr [1:10] "DE9143" "DE9770" "DE1870" "DE4792" ...
 $ transaction_id     : chr [1:10] "TR3416" "TR5378" "TR4534" "TR8649" ...
 $ customer_address_id: chr [1:10] "CA7294" "CA9915" "CA2167" "CA9655" ...
 $ delivery_status    : Factor w/ 4 levels "Completed","Failed",..: 3 1 2 1 3 4 4 1 2 1
 $ delivery_start_date: chr [1:10] "2023-09-22" "2024-02-25" "2023-11-22" "2023-09-02" ...
 $ delivery_end_date  : chr [1:10] NA "2024-03-12" "2023-12-22" "2023-09-04" ...
 - attr(*, "spec")=
```

```
  .. cols(
  ..   delivery_id = col_character(),
  ..   transaction_id = col_character(),
  ..   customer_address_id = col_character(),
  ..   delivery_status = col_character(),
  ..   delivery_start_date = col_character(),
  ..   delivery_end_date = col_character()
  .. )
 - attr(*, "problems")=<externalptr>
```

```
summary(delivery)
```

```
 delivery_id        transaction_id     customer_address_id      delivery_status
 Length:10          Length:10          Length:10           Completed    :4
 Class :character   Class :character   Class :character    Failed       :2
 Mode  :character   Mode  :character   Mode  :character    In_Delivery  :2
                                                           Not_Delivered:2
 delivery_start_date delivery_end_date
 Length:10           Length:10
 Class :character    Class :character
 Mode  :character    Mode  :character
```

```
# orders
str(orders)
```

```
spc_tbl_ [200 x 4] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ order_id    : chr [1:200] "OR8136" "OR4490" "OR5487" "OR5142" ...
 $ customer_id : chr [1:200] "CU3705" "CU5023" "CU8051" "CU6553" ...
 $ order_date  : chr [1:200] "2023-12-06" "2023-09-06" "2023-09-24" "2023-09-12" ...
 $ order_status: chr [1:200] "Pending" "Pending" "Processing" "Succeed" ...
 - attr(*, "spec")=
  .. cols(
  ..   order_id = col_character(),
  ..   customer_id = col_character(),
  ..   order_date = col_character(),
  ..   order_status = col_character()
  .. )
 - attr(*, "problems")=<externalptr>
```

```r
orders$order_status = as.factor(orders$order_status)
str(orders)
```

```
spc_tbl_ [200 x 4] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ order_id    : chr [1:200] "OR8136" "OR4490" "OR5487" "OR5142" ...
 $ customer_id : chr [1:200] "CU3705" "CU5023" "CU8051" "CU6553" ...
 $ order_date  : chr [1:200] "2023-12-06" "2023-09-06" "2023-09-24" "2023-09-12" ...
 $ order_status: Factor w/ 4 levels "Cancelled","Pending",..: 2 2 3 4 3 3 3 4 2 2 ...
 - attr(*, "spec")=
  .. cols(
  ..   order_id = col_character(),
  ..   customer_id = col_character(),
  ..   order_date = col_character(),
  ..   order_status = col_character()
  .. )
 - attr(*, "problems")=<externalptr>
```

```r
summary(orders)
```

```
   order_id         customer_id         order_date          order_status
 Length:200         Length:200         Length:200         Cancelled :52
 Class :character   Class :character   Class :character   Pending   :50
 Mode  :character   Mode  :character   Mode  :character   Processing:52
                                                          Succeed   :46
```

```r
# orders_Details
str(orders_details)
```

```
spc_tbl_ [1,111 x 4] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ order_detail_id: chr [1:1111] "OD8508" "OD7044" "OD7465" "OD8159" ...
 $ order_id       : chr [1:1111] "OR8136" "OR4490" "OR5487" "OR5487" ...
 $ product_id     : chr [1:1111] "PR7365" "PR8045" "PR2068" "PR4512" ...
 $ sub_quantity   : num [1:1111] 1 4 4 3 1 2 9 1 10 1 ...
 - attr(*, "spec")=
  .. cols(
  ..   order_detail_id = col_character(),
  ..   order_id = col_character(),
  ..   product_id = col_character(),
  ..   sub_quantity = col_double()
  .. )
 - attr(*, "problems")=<externalptr>
```

```r
summary(orders_details)
```

```
 order_detail_id      order_id          product_id         sub_quantity
 Length:1111        Length:1111        Length:1111        Min.   : 1.000
 Class :character   Class :character   Class :character   1st Qu.: 3.000
 Mode  :character   Mode  :character   Mode  :character   Median : 6.000
                                                          Mean   : 5.465
                                                          3rd Qu.: 8.000
                                                          Max.   :10.000
```

```r
# Product
str(product)
```

```
spc_tbl_ [500 x 6] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ product_id         : chr [1:500] "PR8894" "PR3108" "PR0267" "PR0122" ...
 $ category_id        : chr [1:500] "CT1002-623" "CT1001-103" "CT1004-771" "CT1001-865" ...
 $ product_name       : chr [1:500] "Gaming Laptop" "Trench Coat" "Dining Set" "Summer Dress"
 $ product_price      : num [1:500] 396 1582 3071 3187 2979 ...
 $ discount_percentage: num [1:500] 0.03 0.98 0.2 0.37 0.76 0.87 0.04 0.77 0.66 0.72 ...
 $ rate_value         : num [1:500] 4.6 2.2 5 1.2 3.3 2.5 2.6 1.9 4.3 2.7 ...
 - attr(*, "spec")=
  .. cols(
  ..   product_id = col_character(),
  ..   category_id = col_character(),
  ..   product_name = col_character(),
  ..   product_price = col_double(),
  ..   discount_percentage = col_double(),
  ..   rate_value = col_double()
  .. )
 - attr(*, "problems")=<externalptr>
```

```r
summary(product)
```

```
 product_id         category_id        product_name       product_price
 Length:500         Length:500         Length:500         Min.   :  30.25
 Class :character   Class :character   Class :character   1st Qu.:1305.07
 Mode  :character   Mode  :character   Mode  :character   Median :2420.86
                                                          Mean   :2479.27
                                                          3rd Qu.:3697.51
                                                          Max.   :4997.68
```

```
 discount_percentage   rate_value
 Min.   :0.0000      Min.   :1.000
 1st Qu.:0.2200      1st Qu.:2.100
 Median :0.5000      Median :3.100
 Mean   :0.5013      Mean   :3.058
 3rd Qu.:0.7600      3rd Qu.:4.100
 Max.   :0.9900      Max.   :5.000
```

```
# Supplier
str(supplier)
```

```
spc_tbl_ [80 x 8] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ supplier_id      : chr [1:80] "SU0454" "SU3047" "SU2545" "SU3336" ...
 $ supplier_name    : chr [1:80] "Keller Ltd" "Green-Duke" "Carey-Wilson" "Wilson and Sons"
 $ supplier_email   : chr [1:80] "briannatorres@example.org" "beasleydanielle@example.org" "
 $ supplier_zip_code: chr [1:80] "51142" "61509" "76466" "93041" ...
 $ supplier_country : chr [1:80] "Montenegro" "Mexico" "Cameroon" "North Macedonia" ...
 $ supplier_state   : chr [1:80] "New Mexico" "Illinois" "California" "Maryland" ...
 $ supplier_city    : chr [1:80] "Lake Amandachester" "West Troyburgh" "Nicoleborough" "Fergu
 $ supplier_street  : chr [1:80] "77637 Wolf Valleys" "19501 Pamela Grove" "7521 Kirk Summit
 - attr(*, "spec")=
  .. cols(
  ..   supplier_id = col_character(),
  ..   supplier_name = col_character(),
  ..   supplier_email = col_character(),
  ..   supplier_zip_code = col_character(),
  ..   supplier_country = col_character(),
  ..   supplier_state = col_character(),
  ..   supplier_city = col_character(),
  ..   supplier_street = col_character()
  .. )
 - attr(*, "problems")=<externalptr>
```

```
#supplier$supplier_phone = as.character(supplier$supplier_phone)
supplier$supplier_country = as.factor(supplier$supplier_country)
supplier$supplier_state = as.factor(supplier$supplier_state)
supplier$supplier_city = as.factor(supplier$supplier_city)
str(supplier)
```

```
spc_tbl_ [80 x 8] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ supplier_id      : chr [1:80] "SU0454" "SU3047" "SU2545" "SU3336" ...
```

```
$ supplier_name    : chr [1:80] "Keller Ltd" "Green-Duke" "Carey-Wilson" "Wilson and Sons"
$ supplier_email   : chr [1:80] "briannatorres@example.org" "beasleydanielle@example.org" "
$ supplier_zip_code: chr [1:80] "51142" "61509" "76466" "93041" ...
$ supplier_country : Factor w/ 74 levels "Albania","American Samoa",..: 42 41 12 50 73 69 6
$ supplier_state   : Factor w/ 42 levels "Alabama","Alaska",..: 25 11 4 15 29 25 23 22 25 3
$ supplier_city    : Factor w/ 79 levels "Adammouth","Adamsshire",..: 28 79 44 14 7 61 52 7
$ supplier_street  : chr [1:80] "77637 Wolf Valleys" "19501 Pamela Grove" "7521 Kirk Summit
- attr(*, "spec")=
 .. cols(
 ..   supplier_id = col_character(),
 ..   supplier_name = col_character(),
 ..   supplier_email = col_character(),
 ..   supplier_zip_code = col_character(),
 ..   supplier_country = col_character(),
 ..   supplier_state = col_character(),
 ..   supplier_city = col_character(),
 ..   supplier_street = col_character()
 .. )
- attr(*, "problems")=<externalptr>
```

```r
summary(supplier)
```

```
 supplier_id        supplier_name      supplier_email     supplier_zip_code
 Length:80          Length:80          Length:80          Length:80
 Class :character   Class :character   Class :character   Class :character
 Mode  :character   Mode  :character   Mode  :character   Mode  :character
```

```
  supplier_country    supplier_state       supplier_city supplier_street
 Bahrain   : 2      Nevada    : 5     Riveraland   : 2    Length:80
 Bermuda   : 2      Ohio      : 5     Adammouth    : 1    Class :character
 India     : 2      California: 4     Adamsshire   : 1    Mode  :character
 Montserrat: 2      New Mexico: 4     Andrewchester: 1
 Peru      : 2      Indiana   : 3     Andrewmouth  : 1
 Romania   : 2      Kentucky  : 3     Andrewsmouth : 1
 (Other)   :68      (Other)   :56     (Other)      :73
```

```r
# Supply
str(supply)
```

```
spc_tbl_ [100 x 4] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ supply_id  : chr [1:100] "SP1667" "SP1197" "SP3947" "SP7219" ...
 $ supplier_id: chr [1:100] "SU2314" "SU2615" "SU6448" "SU6356" ...
 $ product_id : chr [1:100] "PR6800" "PR5180" "PR5183" "PR2265" ...
 $ stock      : num [1:100] 178 783 86 197 535 182 662 83 320 800 ...
 - attr(*, "spec")=
  .. cols(
  ..   supply_id = col_character(),
  ..   supplier_id = col_character(),
  ..   product_id = col_character(),
  ..   stock = col_double()
  .. )
 - attr(*, "problems")=<externalptr>
```

```
summary(supply)
```

```
  supply_id         supplier_id         product_id            stock       
 Length:100         Length:100         Length:100         Min.   :  4.0  
 Class :character   Class :character   Class :character   1st Qu.:231.5  
 Mode  :character   Mode  :character   Mode  :character   Median :522.0  
                                                          Mean   :509.1  
                                                          3rd Qu.:805.8  
                                                          Max.   :998.0  
```

```
# Transaction
str(transaction)
```

```
spc_tbl_ [46 x 6] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ transaction_id    : chr [1:46] "TR3416" "TR7104" "TR2501" "TR5378" ...
 $ order_id          : chr [1:46] "OR5142" "OR0737" "OR2802" "OR5005" ...
 $ customer_id       : chr [1:46] "CU6553" "CU0214" "CU5515" "CU1936" ...
 $ transaction_time  : chr [1:46] "2023-09-12 05:21:09" "2024-02-13 21:08:31" "2024-02-18 03
 $ payment_method    : chr [1:46] "Debit Card" "PayPal" "Debit Card" "Credit Card" ...
 $ transaction_status: chr [1:46] "Succeed" "Pending" "Processing" "Succeed" ...
 - attr(*, "spec")=
  .. cols(
  ..   transaction_id = col_character(),
  ..   order_id = col_character(),
  ..   customer_id = col_character(),
  ..   transaction_time = col_character(),
  ..   payment_method = col_character(),
```

```
  ..   transaction_status = col_character()
  .. )
 - attr(*, "problems")=<externalptr>
```

```r
# Transform transaction_time without the float
transaction$transaction_time = sub("\\..*", "", transaction$transaction_time)
transaction$payment_method = as_factor(transaction$payment_method)
transaction$transaction_status = as_factor(transaction$transaction_status)
str(transaction)
```

```
spc_tbl_ [46 x 6] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ transaction_id    : chr [1:46] "TR3416" "TR7104" "TR2501" "TR5378" ...
 $ order_id          : chr [1:46] "OR5142" "OR0737" "OR2802" "OR5005" ...
 $ customer_id       : chr [1:46] "CU6553" "CU0214" "CU5515" "CU1936" ...
 $ transaction_time  : chr [1:46] "2023-09-12 05:21:09" "2024-02-13 21:08:31" "2024-02-18 03
 $ payment_method    : Factor w/ 3 levels "Debit Card","PayPal",..: 1 2 1 3 2 3 2 3 1 3 ...
 $ transaction_status: Factor w/ 4 levels "Succeed","Pending",..: 1 2 3 1 3 4 2 3 2 1 ...
 - attr(*, "spec")=
  .. cols(
  ..   transaction_id = col_character(),
  ..   order_id = col_character(),
  ..   customer_id = col_character(),
  ..   transaction_time = col_character(),
  ..   payment_method = col_character(),
  ..   transaction_status = col_character()
  .. )
 - attr(*, "problems")=<externalptr>
```

```r
summary(transaction)
```

```
 transaction_id       order_id         customer_id        transaction_time
 Length:46          Length:46          Length:46          Length:46
 Class :character   Class :character   Class :character   Class :character
 Mode  :character   Mode  :character   Mode  :character   Mode  :character

    payment_method  transaction_status
 Debit Card :12     Succeed   :10
 PayPal     :15     Pending   :11
 Credit Card:19     Processing:13
                    Failed    :12
```

Write them to the database

```
RSQLite::dbWriteTable(my_connection,"ads",ads,append = TRUE,overwrite=FALSE)
RSQLite::dbWriteTable(my_connection,"category",category,append = TRUE,overwrite=FALSE)
RSQLite::dbWriteTable(my_connection,"customer_address",customer_address,append = TRUE,overwr
RSQLite::dbWriteTable(my_connection,"customer",customer,append = TRUE,overwrite=FALSE)
RSQLite::dbWriteTable(my_connection,"delivery",delivery,append = TRUE,overwrite=FALSE)
RSQLite::dbWriteTable(my_connection,"orders",orders,append = TRUE,overwrite=FALSE)
RSQLite::dbWriteTable(my_connection,"orders_details",orders_details,append = TRUE,overwrite=
RSQLite::dbWriteTable(my_connection,"product",product,append = TRUE,overwrite=FALSE)
RSQLite::dbWriteTable(my_connection,"supplier",supplier,append = TRUE,overwrite=FALSE)
RSQLite::dbWriteTable(my_connection,"supply",supply,append = TRUE,overwrite=FALSE)
RSQLite::dbWriteTable(my_connection,"transaction",transaction,append = TRUE,overwrite=FALSE)
```

Database check

```
# Tables check to see if there's any error in the data import stage
ads_check = tbl(my_connection, "ads")
ads_check
```

```
# Source:   table<ads> [?? x 5]
# Database: sqlite 3.45.0 [/cloud/project/database/an_e_commerce.db]
   ad_id  product_id ad_status ad_start_date ad_end_date
   <chr>  <chr>      <chr>     <chr>         <chr>
 1 AD3411 PR5955     active    2023-09-04    2024-04-26
 2 AD5591 PR2980     active    2023-12-04    2025-01-07
 3 AD2023 PR4455     active    2023-06-18    2024-04-13
 4 AD2393 PR9243     ended     2023-08-04    2023-09-09
 5 AD4876 PR9370     active    2023-11-08    2025-12-09
 6 AD7559 PR2975     ended     2023-07-21    2023-11-01
 7 AD4058 PR0573     active    2023-11-07    2024-05-10
 8 AD3208 PR6085     active    2023-07-20    2025-07-15
 9 AD8869 PR3912     active    2022-10-21    2025-02-22
10 AD4780 PR2619     ended     2023-06-08    2023-10-19
# i more rows
```

```
category_check = tbl(my_connection, "category")
category_check
```

```
# Source:   table<category> [?? x 3]
# Database: sqlite 3.45.0 [/cloud/project/database/an_e_commerce.db]
   category_id category_name parent_category_id
   <chr>       <chr>         <chr>
 1 CT1001      Clothing      <NA>
 2 CT1001-208  Shirts        CT1001
 3 CT1001-546  Pants         CT1001
 4 CT1001-865  Dresses       CT1001
 5 CT1001-103  Outerwear     CT1001
 6 CT1002      Electronics   <NA>
 7 CT1002-150  Smartphones   CT1002
 8 CT1002-623  Laptops       CT1002
 9 CT1002-942  Cameras       CT1002
10 CT1002-517  Accessories   CT1002
# i more rows
```

```
customer_address_check = tbl(my_connection, "customer_address")
customer_address_check
```

```
# Source:   table<customer_address> [?? x 7]
# Database: sqlite 3.45.0 [/cloud/project/database/an_e_commerce.db]
   customer_address_id customer_id zip_code country           state city  street
   <chr>               <chr>       <chr>    <chr>             <chr> <chr> <chr>
 1 CA8785              CU3320      21639    Greece            Ariz~ East~ 76218~
 2 CA6746              CU3320      86282    Bosnia and Herze~ Nebr~ East~ 2956 ~
 3 CA4295              CU3320      57281    Jersey            Wash~ New ~ 7829 ~
 4 CA2484              CU8307      30961    Luxembourg        West~ Nort~ 6383 ~
 5 CA3686              CU8307      10724    Somalia           Loui~ Anna~ 93301~
 6 CA4009              CU0170      09807    Colombia          New ~ Will~ 70245~
 7 CA3795              CU0170      63393    Guernsey          Miss~ East~ 811 D~
 8 CA9475              CU2873      20237    Christmas Island  Kans~ Lake~ 728 J~
 9 CA3216              CU2873      49246    Honduras          Hawa~ Vict~ 603 P~
10 CA2646              CU2873      58802    Sri Lanka         Virg~ Tara~ 7030 ~
# i more rows
```

```
customer_check = tbl(my_connection, "customer")
customer_check
```

```
# Source:   table<customer> [?? x 8]
# Database: sqlite 3.45.0 [/cloud/project/database/an_e_commerce.db]
   customer_id first_name last_name gender date_of_birth email     password_hash
```

```
   <chr>       <chr>      <chr>     <chr>  <chr>          <chr>     <chr>
 1 CU3320      Brian      Miller    female 1962-12-09     roberson~ 429b5dfe28a2~
 2 CU8307      Martin     Christian male   1944-11-12     ann32@ex~ da7e994ae66d~
 3 CU0170      Kim        Johns     female 1960-10-09     stewarts~ 18c5bd20707f~
 4 CU2873      Douglas    Tyler     female 1934-10-05     qberry@e~ 013f9b5037a0~
 5 CU8659      Kayla      Cooper    male   1991-09-20     lorifowl~ dbfc8b180869~
 6 CU8645      Leslie     Sanders   other  1989-05-14     ocohen@e~ 57de529ffd54~
 7 CU5458      Martha     Meyer     other  1963-11-01     boonedeb~ 8a2bea3e41bd~
 8 CU7042      Robin      Barrett   female 1945-08-18     angela17~ 6c89b3eb29e3~
 9 CU5873      Matthew    Lopez     female 1939-08-24     alexande~ 1e88c47a5ff5~
10 CU1060      Jennifer   Stout     female 1979-11-04     omaynard~ e9f3bb2ded40~
# i more rows
# i 1 more variable: membership <chr>
```

```
delivery_check = tbl(my_connection, "delivery")
delivery_check
```

```
# Source:   table<delivery> [10 x 6]
# Database: sqlite 3.45.0 [/cloud/project/database/an_e_commerce.db]
   delivery_id transaction_id customer_address_id delivery_status
   <chr>       <chr>          <chr>               <chr>
 1 DE9143      TR3416         CA7294              In_Delivery
 2 DE9770      TR5378         CA9915              Completed
 3 DE1870      TR4534         CA2167              Failed
 4 DE4792      TR8649         CA9655              Completed
 5 DE0758      TR1303         CA7441              In_Delivery
 6 DE1130      TR5157         CA5519              Not_Delivered
 7 DE3289      TR0380         CA5726              Not_Delivered
 8 DE1336      TR6304         CA9080              Completed
 9 DE1135      TR8050         CA6293              Failed
10 DE6760      TR4632         CA0338              Completed
# i 2 more variables: delivery_start_date <chr>, delivery_end_date <chr>
```

```
orders_check = tbl(my_connection, "orders")
orders_check
```

```
# Source:   table<orders> [?? x 4]
# Database: sqlite 3.45.0 [/cloud/project/database/an_e_commerce.db]
   order_id customer_id order_date order_status
   <chr>    <chr>       <chr>      <chr>
 1 OR8136   CU3705      2023-12-06 Pending
```

```
 2 OR4490   CU5023      2023-09-06 Pending
 3 OR5487   CU8051      2023-09-24 Processing
 4 OR5142   CU6553      2023-09-12 Succeed
 5 OR9845   CU3737      2023-11-05 Processing
 6 OR0940   CU5873      2023-12-30 Processing
 7 OR3870   CU0352      2024-02-13 Processing
 8 OR0737   CU0214      2024-02-13 Succeed
 9 OR9861   CU7908      2024-02-08 Pending
10 OR8203   CU3205      2024-02-15 Pending
# i more rows
```

```
orders_details_check = tbl(my_connection, "orders_details")
orders_details_check
```

```
# Source:   table<orders_details> [?? x 4]
# Database: sqlite 3.45.0 [/cloud/project/database/an_e_commerce.db]
   order_detail_id order_id product_id sub_quantity
   <chr>           <chr>    <chr>             <dbl>
 1 OD8508          OR8136   PR7365                1
 2 OD7044          OR4490   PR8045                4
 3 OD7465          OR5487   PR2068                4
 4 OD8159          OR5487   PR4512                3
 5 OD3780          OR5487   PR2654                1
 6 OD7705          OR5487   PR9602                2
 7 OD2209          OR5487   PR4947                9
 8 OD8482          OR5487   PR4104                1
 9 OD6555          OR5487   PR4994               10
10 OD4201          OR5142   PR5470                1
# i more rows
```

```
product_check = tbl(my_connection, "product")
product_check
```

```
# Source:   table<product> [?? x 6]
# Database: sqlite 3.45.0 [/cloud/project/database/an_e_commerce.db]
   product_id category_id product_name         product_price discount_percentage
   <chr>      <chr>       <chr>                        <dbl>               <dbl>
 1 PR8894     CT1002-623  Gaming Laptop                 396.                0.03
 2 PR3108     CT1001-103  Trench Coat                  1582.                0.98
 3 PR0267     CT1004-771  Dining Set                   3071.                0.2
 4 PR0122     CT1001-865  Summer Dress                 3187.                0.37
```

```
 5 PR7884     CT1003-935  Historical Biography         2979.                0.76
 6 PR1733     CT1003-799  Mathematics Textbook         1969.                0.87
 7 PR2275     CT1002-623  Ultrabook                     603.                0.04
 8 PR5132     CT1003-799  Language Workbook            2493.                0.77
 9 PR8387     CT1001-103  Windbreaker                  3093.                0.66
10 PR8525     CT1002-623  Ultrabook                    1165.                0.72
# i more rows
# i 1 more variable: rate_value <dbl>
```

```
supplier_check = tbl(my_connection, "supplier")
supplier_check
```

```
# Source:   table<supplier> [?? x 8]
# Database: sqlite 3.45.0 [/cloud/project/database/an_e_commerce.db]
   supplier_id supplier_name   supplier_email supplier_zip_code supplier_country
   <chr>       <chr>           <chr>          <chr>             <chr>
 1 SU0454      Keller Ltd      briannatorres~ 51142             Montenegro
 2 SU3047      Green-Duke      beasleydaniel~ 61509             Mexico
 3 SU2545      Carey-Wilson    frederickdani~ 76466             Cameroon
 4 SU3336      Wilson and Sons qlowe@example~ 93041             North Macedonia
 5 SU7830      Greer, Willis ~ melissa83@exa~ 31787             Wallis and Futu~
 6 SU5534      Wheeler, David~ andersonheath~ 14634             United States o~
 7 SU5030      Rodriguez, Cra~ stokessarah@e~ 96994             Tonga
 8 SU4734      Carey Group     dgardner@exam~ 20836             Romania
 9 SU2738      Ray PLC         john61@exampl~ 46284             Central African~
10 SU4246      Ayers and Sons  andrewfrankli~ 39617             Djibouti
# i more rows
# i 3 more variables: supplier_state <chr>, supplier_city <chr>,
#   supplier_street <chr>
```

```
supply_check = tbl(my_connection, "supply")
supply_check
```

```
# Source:   table<supply> [?? x 4]
# Database: sqlite 3.45.0 [/cloud/project/database/an_e_commerce.db]
   supply_id supplier_id product_id stock
   <chr>     <chr>       <chr>      <dbl>
 1 SP1667    SU2314      PR6800       178
 2 SP1197    SU2615      PR5180       783
 3 SP3947    SU6448      PR5183        86
 4 SP7219    SU6356      PR2265       197
```

```
 5 SP1187    SU8172      PR9807       535
 6 SP7378    SU4069      PR5026       182
 7 SP9899    SU7168      PR2927       662
 8 SP3207    SU2961      PR5026        83
 9 SP6177    SU0300      PR9839       320
10 SP4112    SU3311      PR2068       800
# i more rows
```

```
transaction_check = tbl(my_connection, "transaction")
transaction_check
```

```
# Source:   table<transaction> [?? x 6]
# Database: sqlite 3.45.0 [/cloud/project/database/an_e_commerce.db]
   transaction_id order_id customer_id transaction_time    payment_method
   <chr>          <chr>    <chr>       <chr>               <chr>
 1 TR3416         OR5142   CU6553      2023-09-12 05:21:09 Debit Card
 2 TR7104         OR0737   CU0214      2024-02-13 21:08:31 PayPal
 3 TR2501         OR2802   CU5515      2024-02-18 03:50:27 Debit Card
 4 TR5378         OR5005   CU1936      2024-02-21 01:15:03 Credit Card
 5 TR0480         OR4723   CU3812      2023-12-01 07:04:50 PayPal
 6 TR8780         OR6113   CU1607      2023-12-07 09:40:14 Credit Card
 7 TR3405         OR4918   CU4580      2024-01-21 09:04:35 PayPal
 8 TR0583         OR5176   CU1936      2024-02-17 23:53:51 Credit Card
 9 TR5010         OR8263   CU0298      2024-01-06 04:53:56 Debit Card
10 TR4534         OR6259   CU3737      2023-11-18 23:02:47 Credit Card
# i more rows
# i 1 more variable: transaction_status <chr>
```

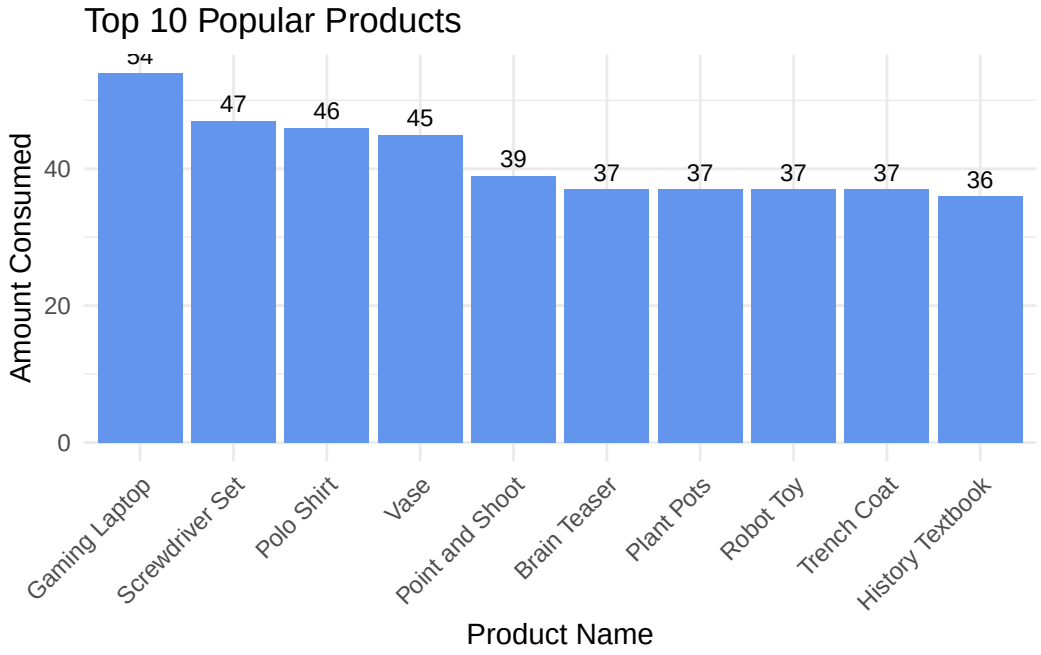
It seems no data structure errors in this stage.

Analysis

1 Product Popularity

```r
# Execute SQL query to count the frequency of each product_id and retrieve top 10 popular pr
top_10_popular_products <- dbGetQuery(my_connection, "
  SELECT p.product_name,
         SUM(od.sub_quantity) AS Frequency
  FROM orders_details od
  JOIN product p ON od.product_id = p.product_id
  GROUP BY od.product_id, p.product_name
  ORDER BY Frequency DESC
  LIMIT 10
")

# Plot the bar chart of the top 10 popular products with product names on the x-axis
ggplot(top_10_popular_products, aes(x = reorder(product_name, -Frequency), y = Frequency)) +
  geom_bar(stat = "identity", fill = "#6495ED") +
  geom_text(aes(label = Frequency), vjust = -0.5, size = 3, color = "black") +  # Add numeri
  labs(title = "Top 10 Popular Products",
       x = "Product Name",
       y = "Amount Consumed") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

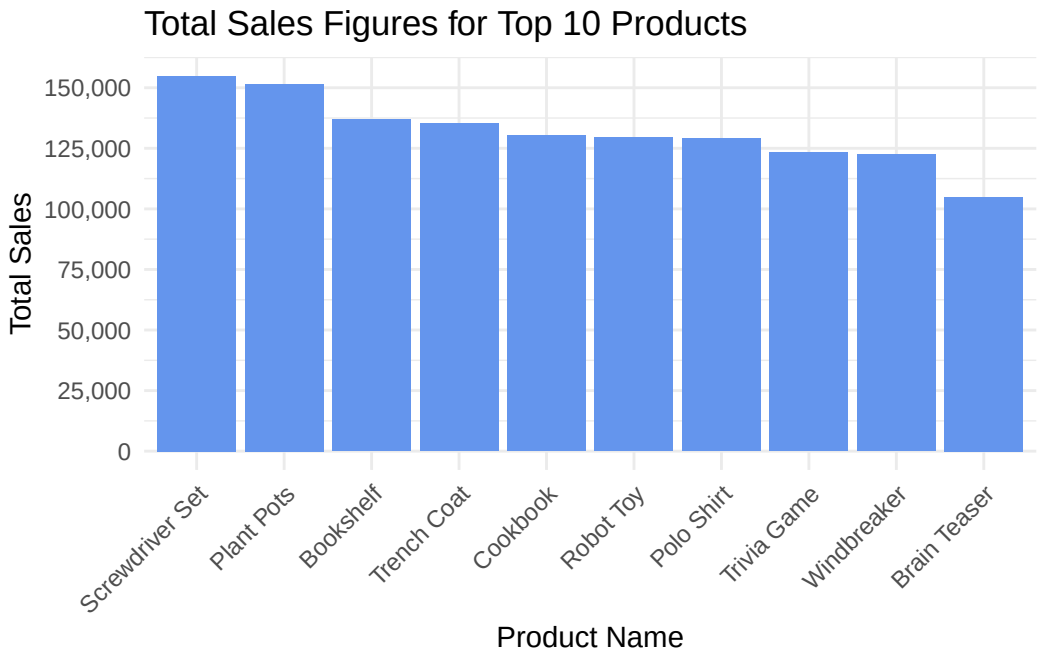


Gaming laptops, screwdriver sets and polo shirts are the top 3 popular products with more than 40 units per product being sold.

2 Product Sales Analysis

```
# Execute SQL query to calculate total revenue for each product and retrieve top 10 products
product_total_revenue <- dbGetQuery(my_connection, "
  SELECT product.product_id,
         product.product_name,
         SUM(CASE
               WHEN customer.membership = 'yes' THEN orders_details.sub_quantity * product.pr
               ELSE orders_details.sub_quantity * product.product_price
             END) AS total_revenue
  FROM orders_details
  JOIN product ON orders_details.product_id = product.product_id
  JOIN orders ON orders_details.order_id = orders.order_id
  JOIN customer ON orders.customer_id = customer.customer_id
  GROUP BY product.product_id
  ORDER BY total_revenue DESC
  LIMIT 10
")

# Create the ggplot for total revenue with product names on the x-axis
ggplot(product_total_revenue, aes(x = reorder(product_name, desc(total_revenue)), y = total_r
  geom_bar(stat = "identity", fill = "#6495ED") +
  labs(title = "Total Sales Figures for Top 10 Products",
       x = "Product Name",
       y = "Total Sales") +
  scale_y_continuous(labels = scales::comma, breaks = seq(0, max(product_total_revenue$total_
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



screw driver sets, Plant pots and bookshelves are the top 3 products that have the most sales, and all top 10 products have more than 100,000 sales which is considerably a great amount of sales. top 10 products are from all categories except the Electronics category. Which makes sense as Electronics will not have a high amount of sales.

3 Category Sales & Popularity Analysis

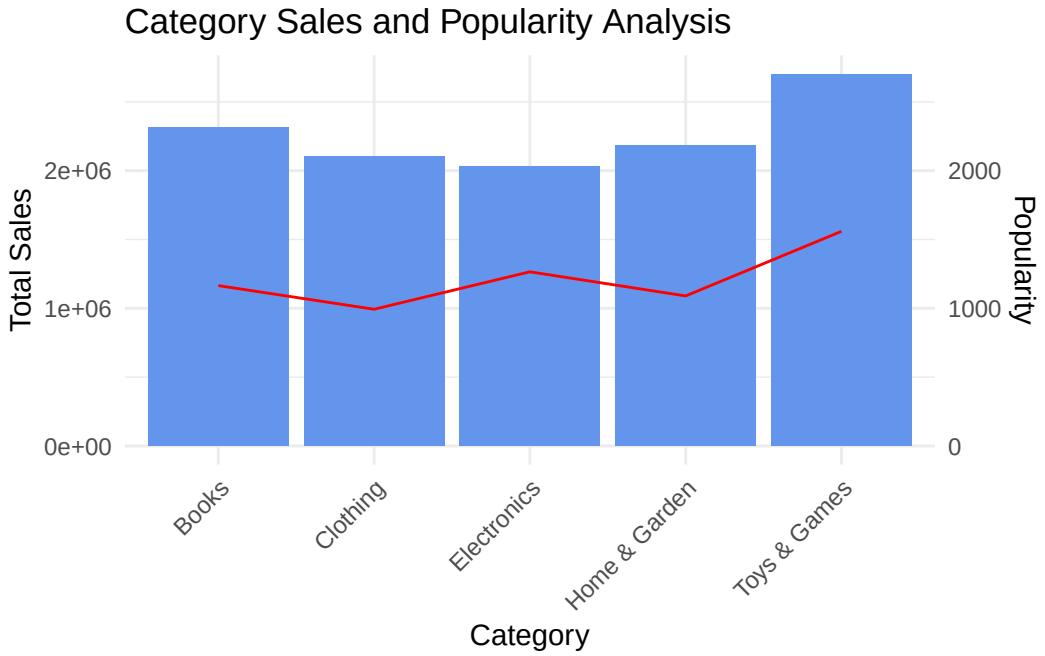
```
# Category Popularity
category_popularity <- dbGetQuery(my_connection, "
SELECT SUBSTRING(product.category_id, 1, 6) AS category,
       SUM(orders_details.sub_quantity) AS total_quantity,
       category.category_name
FROM product
JOIN orders_details ON product.product_id = orders_details.product_id
JOIN category ON SUBSTRING(product.category_id, 1, 6) = category.category_id
GROUP BY SUBSTRING(product.category_id, 1, 6), category.category_name
")

# Category Sales
category_sales <- dbGetQuery(my_connection, "
SELECT SUBSTRING(product.category_id, 1, 6) AS category,
       SUM(CASE
```

```
               WHEN customer.membership = 'yes' THEN orders_details.sub_quantity * product.pr
               ELSE orders_details.sub_quantity * product.product_price
           END) AS total_sales,
       category.category_name
FROM product
JOIN orders_details ON product.product_id = orders_details.product_id
JOIN orders ON orders_details.order_id = orders.order_id
JOIN customer ON orders.customer_id = customer.customer_id
JOIN category ON SUBSTRING(product.category_id, 1, 6) = category.category_id
GROUP BY SUBSTRING(product.category_id, 1, 6), category.category_name
")

# Combine category sales and category popularity data, removing duplicate category_name colu
combined_data <- inner_join(category_sales, category_popularity %>% select(-category_name), 

# Create the plot
ggplot(combined_data, aes(x = category_name)) +
  geom_bar(aes(y = total_sales), stat = "identity", fill = "#6495ED") +  # Bar plot represent
  geom_line(aes(y = total_quantity * 1000), color = "red", group = 1) +  # Line plot represe
  scale_y_continuous(name = "Total Sales", sec.axis = sec_axis(~./1000, name = "Popularity"))
  labs(title = "Category Sales and Popularity Analysis",
       x = "Category", y = "Total Sales",) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



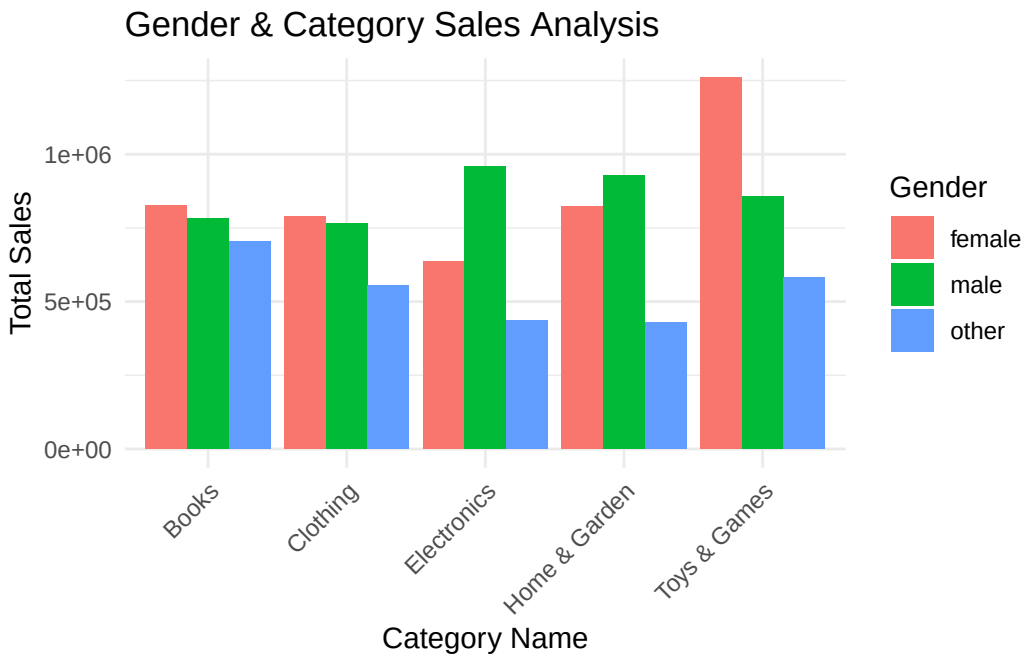
The above graph depicts in terms of popularity of categories and total sales.It can be seen that Home & Garden and Toys & Games categories have the highest amount of sales more than 200,000 and electronics have the lowest amount which is again understandable as Electronics generally do not have a lot of sales

4 Gender & Category Visualization

```
# Execute SQL query in R
category_gender_sales <- dbGetQuery(my_connection, "
SELECT SUBSTRING(product.category_id, 1, 6) AS category,
       customer.gender,
       SUM(CASE
               WHEN customer.membership = 'yes' THEN orders_details.sub_quantity * product.pr
               ELSE orders_details.sub_quantity * product.product_price
           END) AS total_sales,
       category.category_name
FROM product
JOIN orders_details ON product.product_id = orders_details.product_id
JOIN orders ON orders_details.order_id = orders.order_id
JOIN customer ON orders.customer_id = customer.customer_id
JOIN category ON SUBSTRING(product.category_id, 1, 6) = category.category_id
GROUP BY SUBSTRING(product.category_id, 1, 6), category.category_name, customer.gender
```

```
")

# Plot gender-category sales relationship
ggplot(category_gender_sales, aes(x = category_name, y = total_sales, fill = gender)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Gender & Category Sales Analysis",
       x = "Category Name",
       y = "Total Sales",
       fill = "Gender") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

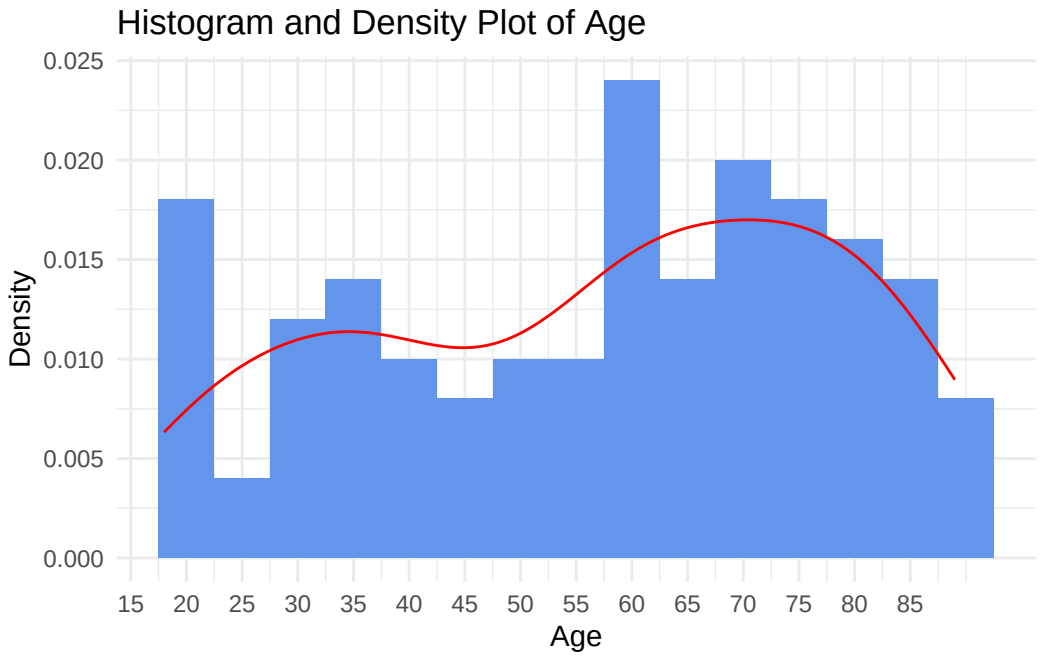


The above graph depicts that in general all genders equally buy Home & Garden products but there is a clear difference in gender differences for other categories. For clothing there are more sales to females and Electronics, Games & Toys are more sold to boys. For Books there is somewhat an equal amount of sales to both boys and girls. Along with this, Figure13 represents the density of users are more after 55 years of age suggesting that most of the users of the website are senior citizens and not much sales are done for users who are young and middle aged. This suggests that more marketing and advertisements have to be run for these age groups.

5 Age Visualization

```
customer1 <- dbGetQuery(my_connection, "
  SELECT customer.*,
  FLOOR((julianday('now') - julianday(customer.date_of_birth)) / 365.25) AS age1
  FROM customer")

# Plot the distribution of Age
ggplot(customer1, aes(x = age1)) +
  geom_histogram(aes(y = after_stat(density)), fill = "#6495ED", binwidth = 5) +
  geom_density(color = "red") +
  labs(title = "Histogram and Density Plot of Age", x = "Age", y = "Density") +
  scale_x_continuous(breaks = seq(0, max(customer1$age1), by = 5)) +
  theme_minimal()
```



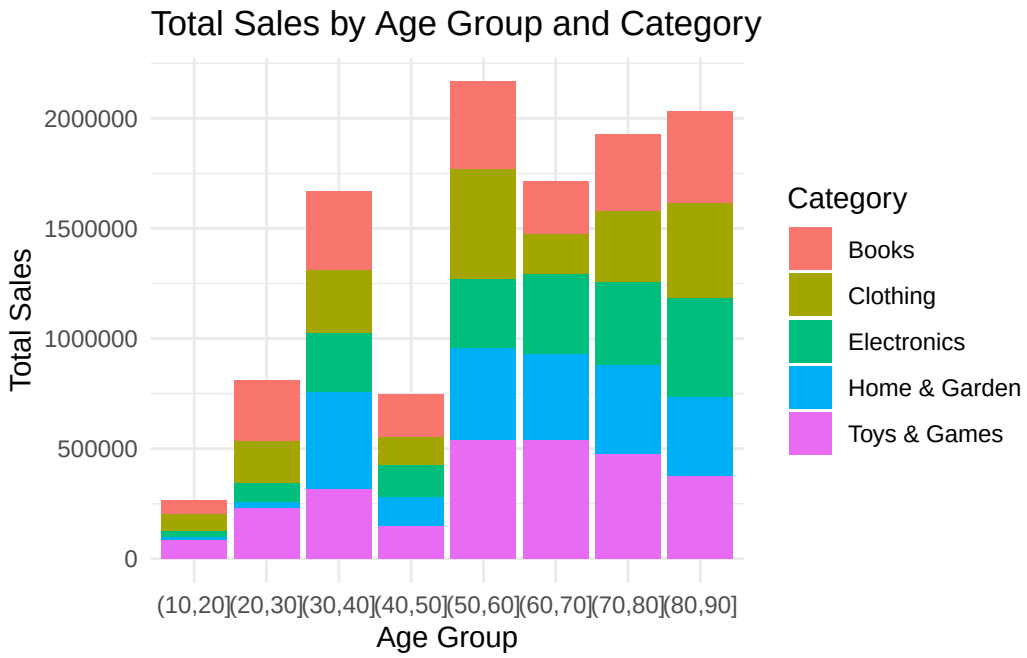
The density of users are more after 55 years of age suggesting that most of the users of the website are senior citizens and not much sales are done for users who are young and middle aged. This suggests that more marketing and advertisements have to be run for these age groups.

6 Age & Category Visualization (pass)

```
# Execute SQL query to get relevant data
customer_data <- dbGetQuery(my_connection, "
  SELECT FLOOR((julianday('now') - julianday(customer.date_of_birth)) / 365.25) AS age,
         SUM(CASE
                 WHEN customer.membership = 'yes' THEN orders_details.sub_quantity * product
                 ELSE orders_details.sub_quantity * product.product_price
             END) AS total_sales,
         SUBSTRING(product.category_id, 1, 6) AS category,
         category.category_name
  FROM customer
  JOIN orders ON customer.customer_id = orders.customer_id
  JOIN orders_details ON orders.order_id = orders_details.order_id
  JOIN product ON orders_details.product_id = product.product_id
  JOIN category ON SUBSTRING(product.category_id, 1, 6) = category.category_id
  GROUP BY age, customer.gender, category, category.category_name
")

# Define age groups with intervals of 10 years
customer_data$age_group <- cut(customer_data$age, breaks = seq(0, max(customer_data$age) + 1

# Plot Age & Category Visualization
ggplot(customer_data, aes(x = age_group, y = total_sales, fill = category_name)) +
  geom_bar(position = "stack", stat = "identity") +
  labs(title = "Total Sales by Age Group and Category",
       x = "Age Group",
       y = "Total Sales",
       fill = "Category") +
  theme_minimal()
```



The above graph depicts that although members are mostly senior citizens most 20-30 and 30-40 age group members have made the most amount of sales suggesting that although the amount of users are low they make high valued purchases. 10-20 and 40-50 aged customers have purchased products at the lowest value. Therefore we can do more marketing to them to increase sales

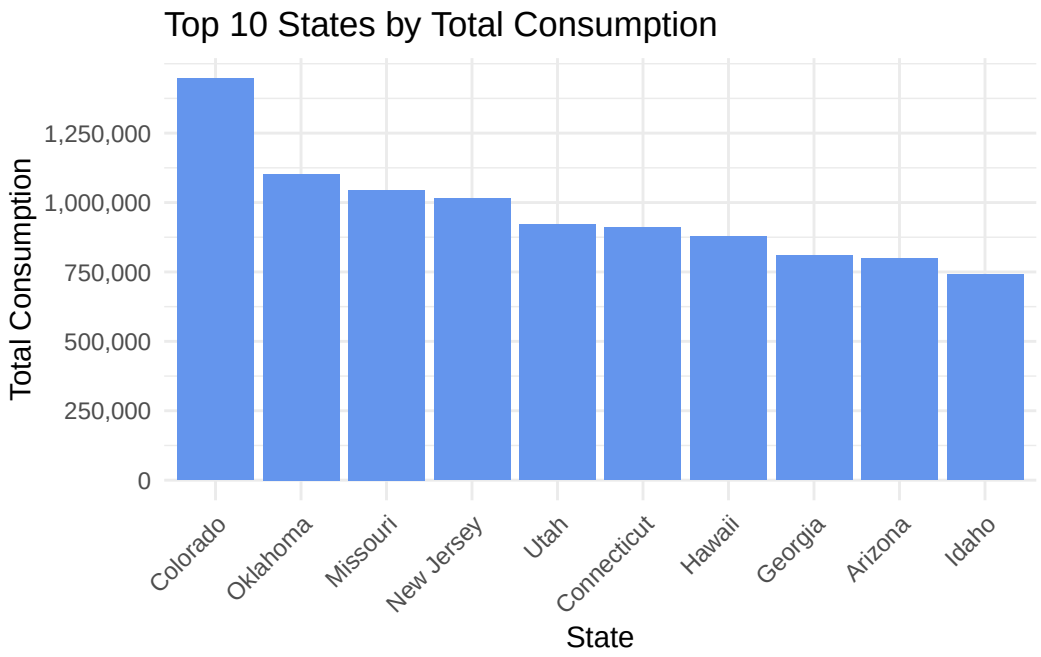
7 Region Revenue Analysis

```
# Execute SQL query to calculate total revenue for each state
state_total_revenue <- dbGetQuery(my_connection, "
  SELECT customer_address.state,
         SUM(
           CASE
             WHEN customer.membership = 'yes' THEN orders_details.sub_quantity * product.proc
             ELSE orders_details.sub_quantity * product.product_price
           END
         ) AS total_revenue
  FROM orders
  JOIN orders_details ON orders.order_id = orders_details.order_id
  JOIN product ON orders_details.product_id = product.product_id
  JOIN customer ON orders.customer_id = customer.customer_id
  JOIN customer_address ON customer.customer_id = customer_address.customer_id
```

```
  GROUP BY customer_address.state
")

# Select the top 10 states by total revenue
top_10_state_total_revenue <- state_total_revenue %>%
  arrange(desc(total_revenue)) %>%
  head(10)

# Plot the total revenue for the top 10 states
ggplot(top_10_state_total_revenue, aes(x = reorder(state, -total_revenue), y = total_revenue
  geom_bar(stat = "identity", fill = "#6495ED") +
  labs(title = "Top 10 States by Total Consumption",
       x = "State",
       y = "Total Consumption") +
  theme_minimal() +
  scale_y_continuous(labels = scales::comma, breaks = seq(0, max(top_10_state_total_revenue$t
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



The above graph shows that Colarado has the most amount of sales and Idaho has the lowest amount of sales however it is still at a considerable level as all sales are above 750,000 How-

ever we can try to increase more sales in Idaho by doing more promotion for users in that particular region

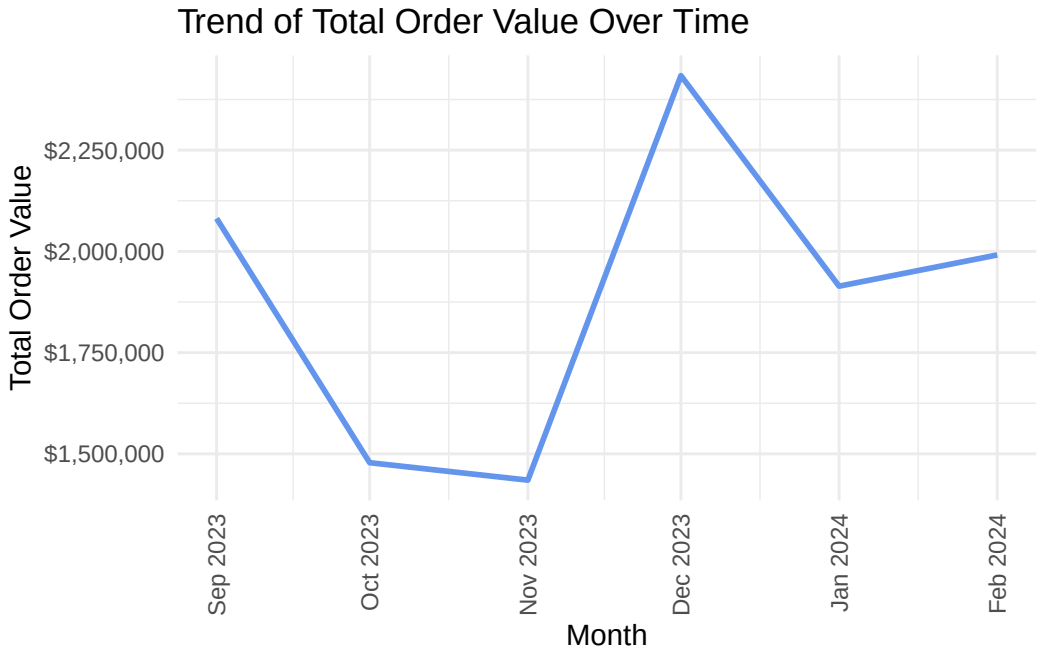
Time Analysis

8 Total Order Value Trend Analysis

```
# Convert Order_Date to month-year format directly in SQL and calculate total order value per
total_order_value_by_month <- dbGetQuery(my_connection, "
  SELECT strftime('%Y-%m', orders.order_date) AS month_year,
         SUM(
           CASE
             WHEN customer.membership = 'yes' THEN orders_details.sub_quantity * product.prod
             ELSE orders_details.sub_quantity * product.product_price
           END
         ) AS total_order_value
  FROM orders
  JOIN orders_details ON orders.order_id = orders_details.order_id
  JOIN product ON orders_details.product_id = product.product_id
  JOIN customer ON orders.customer_id = customer.customer_id
  GROUP BY month_year
")

# Plot the trend of total order value over time
ggplot(total_order_value_by_month, aes(x = as.Date(paste0(month_year, "-01"), "%Y-%m-%d"), y
  geom_line(color = "#6495ED", size = 1, aes(group = 1)) +
  labs(title = "Trend of Total Order Value Over Time",
       x = "Month",
       y = "Total Order Value") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1)) +
  scale_y_continuous(labels = scales::dollar) +  # Adjust y-axis labels to use dollar format
  scale_x_date(labels = scales::date_format("%b %Y")) +  # Format x-axis labels as Sep 2023
  theme(legend.position = "none")  # Remove legend to avoid redundancy
```

```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
i Please use `linewidth` instead.
```

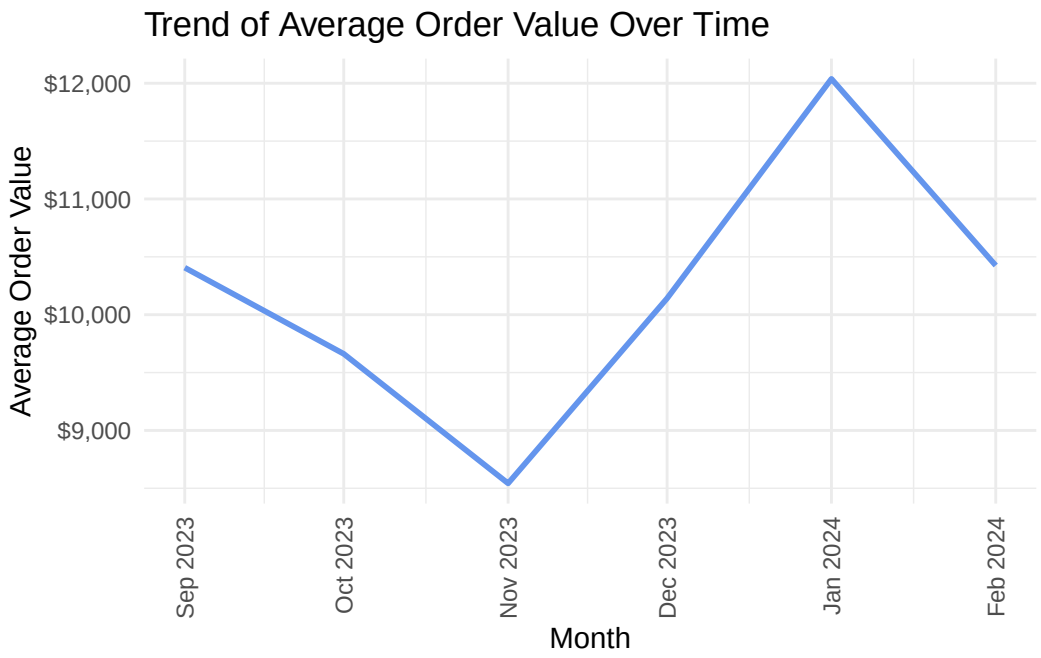


the graph shows the total sales from all orders for each month. The figure increases considerably between October and December 2023, reaching a peak in December, followed by a rapid decrease in the next month.

9 Average Order Value Trend Analysis

```
# Convert Order_Date to month-year format directly in SQL and calculate average order value p
average_order_value_by_month <- dbGetQuery(my_connection, "
  SELECT strftime('%Y-%m', orders.order_date) AS month_year,
         AVG(
           CASE
             WHEN customer.membership = 'yes' THEN orders_details.sub_quantity * product.proc
             ELSE orders_details.sub_quantity * product.product_price
           END
         ) AS average_order_value
  FROM orders
  JOIN orders_details ON orders.order_id = orders_details.order_id
  JOIN product ON orders_details.product_id = product.product_id
  JOIN customer ON orders.customer_id = customer.customer_id
  GROUP BY month_year
")
```

```r
# Plot the trend of average order value over time
ggplot(average_order_value_by_month, aes(x = as.Date(paste0(month_year, "-01"), "%Y-%m-%d"),
  geom_line(color = "#6495ED", size = 1, aes(group = 1)) +
  labs(title = "Trend of Average Order Value Over Time",
       x = "Month",
       y = "Average Order Value") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1)) +
  scale_y_continuous(labels = scales::dollar) +  # Adjust y-axis labels to use dollar format
  scale_x_date(labels = scales::date_format("%b %Y")) +  # Format x-axis labels as Sep 2023
  theme(legend.position = "none")  # Remove legend to avoid redundancy
```



This graph illustrates the average sales value of all orders for each month. There is a significant increase between October and November 2023, while the average sales drop dramatically from January to February 2024

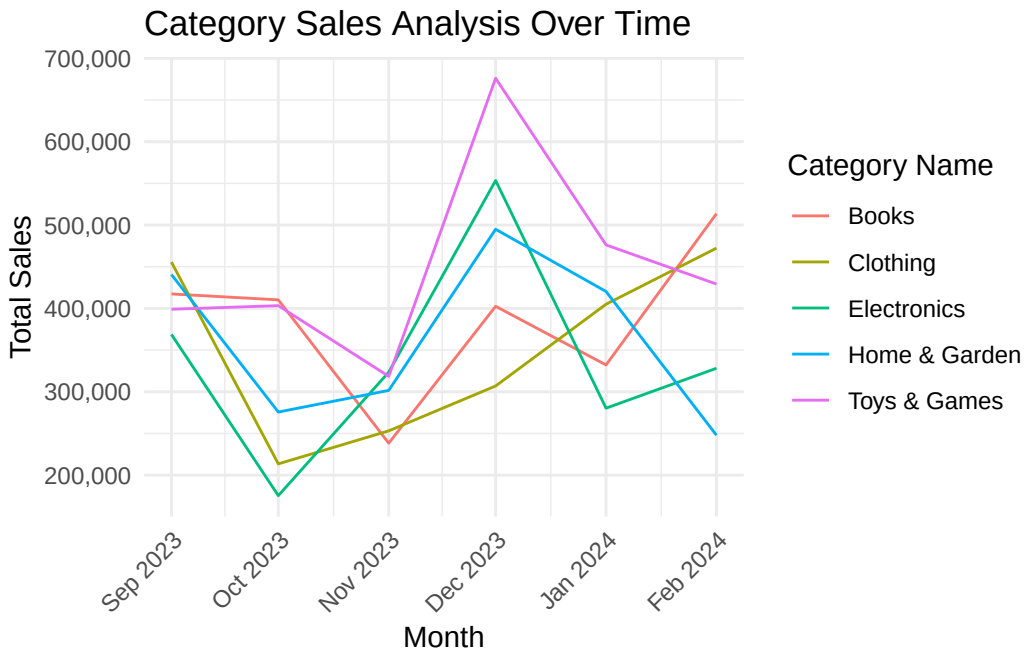
10 Category Sales Trend Analysis

```r
# Execute SQL query in R
category_sales_time <- dbGetQuery(my_connection, "
SELECT strftime('%Y-%m', orders.order_date) AS category_month_year,
       SUBSTRING(product.category_id, 1, 6) AS category,
```

```
       SUM(CASE
               WHEN customer.membership = 'yes' THEN orders_details.sub_quantity * product.pr
               ELSE orders_details.sub_quantity * product.product_price
           END) AS total_sales,
       category.category_name
FROM product
JOIN orders_details ON product.product_id = orders_details.product_id
JOIN orders ON orders_details.order_id = orders.order_id
JOIN customer ON orders.customer_id = customer.customer_id
JOIN category ON SUBSTRING(product.category_id, 1, 6) = category.category_id
GROUP BY strftime('%Y-%m', orders.order_date), SUBSTRING(product.category_id, 1, 6), category
")

# Convert category_month_year to Date format
category_sales_time$category_month_year <- as.Date(paste0(category_sales_time$category_month_

# Plot the line chart of category sales over time for the five main categories
ggplot(category_sales_time, aes(x = category_month_year, y = total_sales, color = category_na
  labs(title = "Category Sales Analysis Over Time",
       x = "Month",
       y = "Total Sales",
       color = "Category Name") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  scale_x_date(date_labels = "%b %Y") +
  scale_y_continuous(labels = scales::comma)
```



The above graph depicts the trend in total consumption with and without membership for each month.

It is apparent that the total consumption for non-members surpasses that of members throughout the entire period, which is counterintuitive. While the consumption trend closely resembles that depicted in Figure 16, the figures for members reached a peak in November 2023.

According to the sales analysis, over time Home & Garden and Electronics sales have met with a sharp decline from January to February. But all categories of books, clothing, electronics, Home & Garden and Toys & Games have had a steady increase from September 2023 to December 2023 suggesting that the products sold through the website are in demand and preferred by customers. The decline from December to January could be because of the year end and most sales happening during the Christmas and new year time for seasonal changes. However, books, clothing and Toys & Games have started to increase again, which suggests that sales will increase, and the company has nothing to worry about

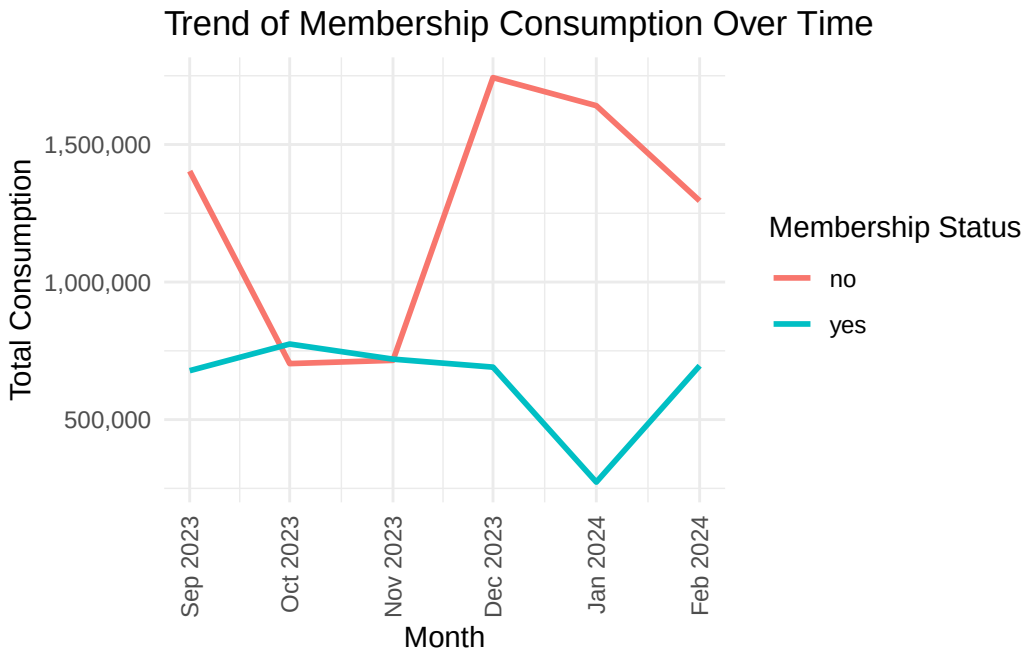
11 Membership Revenue Trend Analysis

```
# Execute SQL query in the database and fetch the results
membership_revenue_by_month <- dbGetQuery(my_connection, "
  SELECT strftime('%Y-%m', orders.order_date) AS membership_month_year,
         customer.membership,
         SUM(CASE
```

```
              WHEN customer.membership = 'yes' THEN orders_details.sub_quantity * product.pr
              ELSE orders_details.sub_quantity * product.product_price
            END) AS total_revenue
  FROM orders
  JOIN orders_details ON orders.order_id = orders_details.order_id
  JOIN product ON orders_details.product_id = product.product_id
  JOIN customer ON orders.customer_id = customer.customer_id
  GROUP BY membership_month_year, customer.membership
")

# Convert the date column to a date-time format
membership_revenue_by_month$membership_month_year <- ym(membership_revenue_by_month$membershi

# Plot the trend of membership revenue over time
ggplot(membership_revenue_by_month, aes(x = membership_month_year, y = total_revenue, group =
  geom_line(size = 1) +
  labs(title = "Trend of Membership Consumption Over Time",
       x = "Month",
       y = "Total Consumption",
       color = "Membership Status") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1)) +
  scale_x_date(date_labels = "%b %Y") +
  scale_y_continuous(labels = scales::comma)
```



The above graph shows the trend in total consumption with and without membership for each month. It is apparent that the total consumption for non-members surpasses that of members throughout the entire period, which is counterintuitive. The figures for members reached a peak in November 2023.

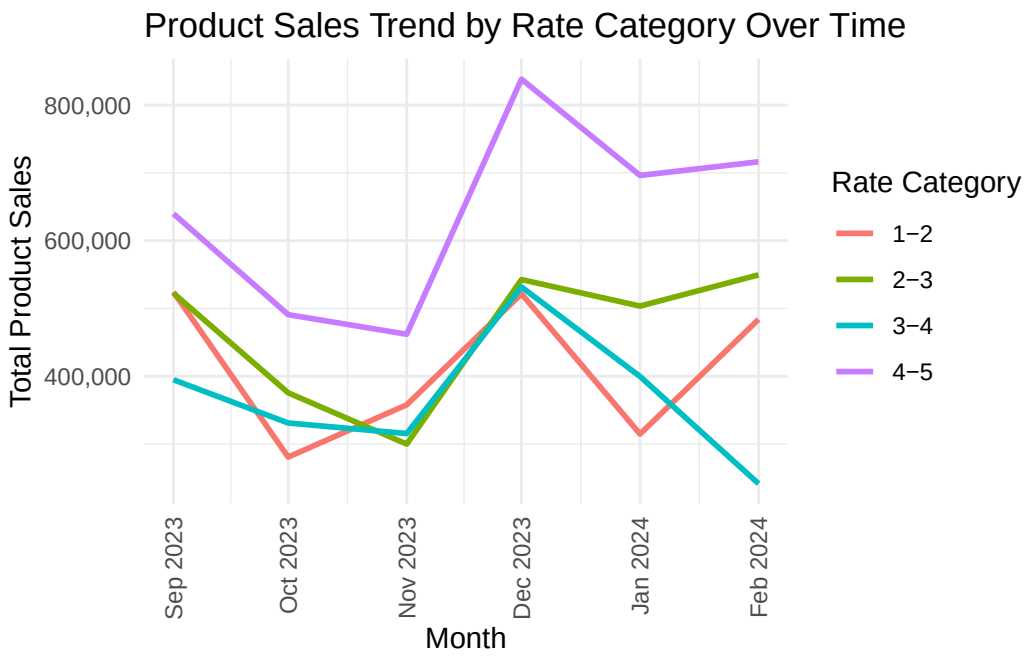
12 Total Product Revenue Analysis by Rate

```
# Query to retrieve product revenue by rate category
product_revenue_by_rate <- dbGetQuery(my_connection, "
  SELECT strftime('%Y-%m', orders.order_date) AS product_month_year,
         SUM(
           CASE
             WHEN customer.membership = 'yes' THEN orders_details.sub_quantity * product.proc
             ELSE orders_details.sub_quantity * product.product_price
           END
         ) AS total_revenue,
         CASE
           WHEN product.rate_value BETWEEN 1 AND 2 THEN '1-2'
           WHEN product.rate_value BETWEEN 2 AND 3 THEN '2-3'
           WHEN product.rate_value BETWEEN 3 AND 4 THEN '3-4'
           WHEN product.rate_value BETWEEN 4 AND 5 THEN '4-5'
           ELSE 'Unknown'
```

```
        END AS rate_category
  FROM orders_details
  JOIN orders ON orders_details.order_id = orders.order_id
  JOIN product ON orders_details.product_id = product.product_id
  JOIN customer ON orders.customer_id = customer.customer_id
  GROUP BY product_month_year, rate_category
")

# Convert the date column to a date-time format
product_revenue_by_rate$product_month_year <- ym(product_revenue_by_rate$product_month_year)

# Plot the trend of product revenue by rate category over time
ggplot(product_revenue_by_rate, aes(x = product_month_year, y = total_revenue, group = rate_
  geom_line(size = 1) +
  labs(title = "Product Sales Trend by Rate Category Over Time",
       x = "Month",
       y = "Total Product Sales",
       color = "Rate Category") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1)) +
  scale_x_date(date_labels = "%b %Y") +
  scale_y_continuous(labels = scales::comma)
```



The graph demonstrates the trend of total product sales categorized into four rate groups.

Product sales in all rate categories exhibit a similar trend to that of Figure 16. However, it appears that the total product sales for both the highest and lowest rate categories maintain similar figures throughout the entire period, except for last month.

It is seen that most of the rates have been given in the month of December where the greatest number of sales also happened. It could have been because of Christmas. It is questionable as the range of rates is quite mixed at 4-5 and 1-2, meaning we have made quality sales but also sales which are unsatisfactory. A further analysis has to be done to gauge for what products by which sellers have got the unsatisfactory rates and steps must be taken to rectify the quality to increase the rates.

Disconnect from the database

```
dbDisconnect(my_connection)
```