

# Data Managment E-commerce Data Base Project

Group 8

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Database Design and Implementation</b>	<b>2</b>
2.1	Entity Relationship Diagram . . . . .	2
2.2	Design Considerations . . . . .	4
2.3	Relationships and Cardinalities . . . . .	4
2.4	Logical Schema . . . . .	7
2.5	Physical Schema Creation . . . . .	9
2.6	Normalisation . . . . .	9
<b>3</b>	<b>Part 2: Data Generation and Management</b>	<b>11</b>
3.1	Synthetic Data Generation . . . . .	11
3.2	Data Import and Quality Assurance . . . . .	12
<b>4</b>	<b>Part 3: Data Pipeline Generation</b>	<b>12</b>
4.1	Github Repository Creation . . . . .	12
4.2	Automated Workflow . . . . .	12
<b>5</b>	<b>Part 4: Data Analysis and Reporting with Quarto in R</b>	<b>12</b>
<b>6</b>	<b>Conclusion</b>	<b>17</b>
<b>7</b>	<b>References</b>	<b>17</b>
<b>8</b>	<b>Appendices</b>	<b>17</b>
8.1	Appendix 1 . . . . .	17
8.2	Appendix 2 . . . . .	17

## Load Necessary Libraries

```
library(DBI)
library(readr)
library(RSQLite)
library(dplyr)
library(stringr)
```

# 1 Introduction

This project simulates a real-world e-commerce data environment, including all stages of data management, from database configuration to data analysis and reporting. As the database is created and managed with SQLite, GitHub Actions are used to automate processes enhancing workflow efficiency. In addition, Quarto is used in conjunction with R to conduct comprehensive data analysis and generate reports. Through this approach, technical skills and competencies necessary for effective data management are examined in-depth, mirroring the complexities and dynamics encountered in e-commerce.

## 2 Database Design and Implementation

### 2.1 Entity Relationship Diagram

The E-R diagram above simulates a real-world e-commerce data ecosystem, capturing the detailed relationships between entities and attributes essential for facilitating online transactions. In addition, it provides a comprehensive view of the e-commerce system, which serves as a platform for users to browse products, make purchases, and securely complete their payments.

#### 2.1.1 Assumptions

- The company only distributes products within the United Kingdom (UK).
- The Currency used is Pound Sterling (GBP).
- Attributes formats will be aligned with UK standard formats such as date , addresses , names ...etc

#### 2.1.2 Entities and Attributes

This section describes and illustrates the entities in the above ERD and their respective attributes.

- Customer

Shows us the users who previously have at least once purchased products and placed an order. Its attributes including names , emails, phone numbers , and addresses.

- Supplier

Vendors who provide products. Represent the source of the product items. The entity store contains attributes of names , addresses, emails, and status that indicates Whether the supplier is currently supplying items or not (Active/Inactive).

- Product

Describes all products in the stock and available for sale. Its attributes are name, price, and availability of the products.

- Order Details

Emphasises all details related to placed orders. Some of its attributes including order total , billing and shipping addresses, type and status of payment , date and status of the order.

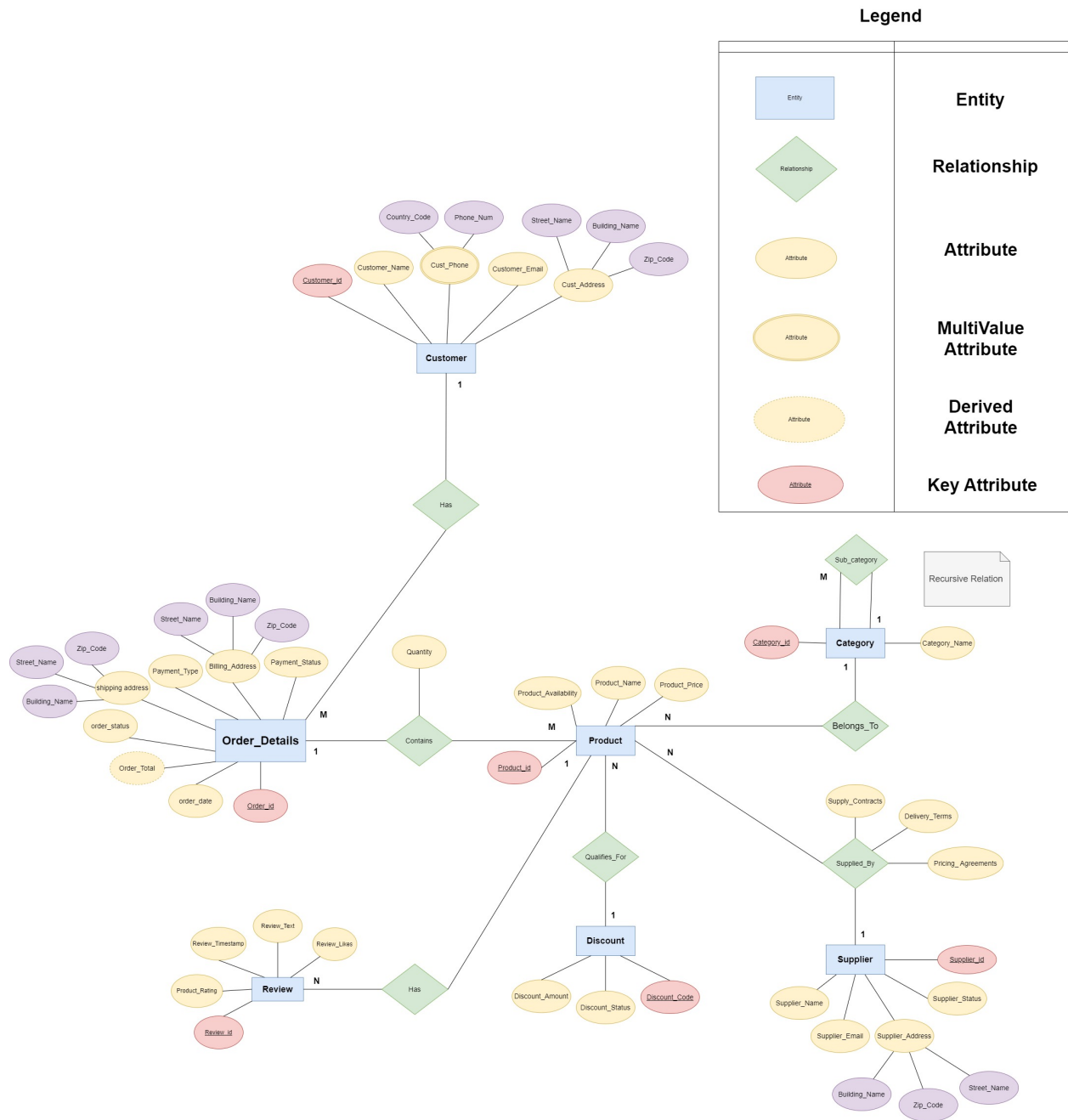


Figure 1: E-Commerce Entity Relationship Diagram

- Category and Sub-Category

Category is the broad classification of products that share common features or are intended for a similar purpose. A sub-category is a more specific grouping of products within a category based on finer distinctions or attributes.

Sub-categories fall under a primary category and help to further organise products into narrower groups, making the product search process even more straightforward for customers.

- Product Discounts

The voucher number or offer code to be applied to eligible products. The amount of discount it offers as well as the status of the discount are the main attributes.

- Reviews

Contains Written comments and rating of product sold by verified buyers, the likes of the top reviews as well as the time stamp of when the review was made.

## 2.2 Design Considerations

### 2.2.1 Absence of an Order Entity

The model intentionally skips direct order management. Instead, it focuses on product management and customer interactions through reviews and payment methods. Additionally, This consideration will guarantee that products purchased by customers are not tracked or stored by the system to align with privacy policies.

Order Entity not considered in this ER design in order to follow best practices by not having to include orderId as part of product table which might affect the overall performance of DB retrieval.

Including Payment methods without an Order entity suggests a pre-registration of payment preferences or a simplified wallet storage that could be expanded in the future.

## 2.3 Relationships and Cardinalities

### 2.3.1 Order Detail Contains Products

This relation indicates that each order detail contains multiple products that have been placed by the customer. It aid to track the date , status , addresses, and payment method used in the transaction.

*Associative Attribute* (Quantity: The number of units of the product ordered in this line item.)

### 2.3.2 Customer Has Order Details

This relation will be created when customers order their first product or service. They will be linked with a particular order detail indicating what they ordered, reflecting the current state or progression throughout the process. One customer can be associated with multiple order statuses at any given time. Moreover, it is good for tracking an order's life cycle, allowing for updates, customer notifications, and management of the order fulfillment process.

### 2.3.3 Product Belongs to Category

Each Product is classified under a specific Category where products can belong to only one category. This enables customers to browse products by category and helps retailers manage product listings more efficiently.

### 2.3.4 Category Self-Reference Relation

A category can have multiple subcategories, creating a hierarchically nested structure and making it easier for users to navigate the product catalogue. For example, "Home Appliances" is the parent category contains subcategories like Washing Machines , Food Processors, Dishwashers etc.

1:N

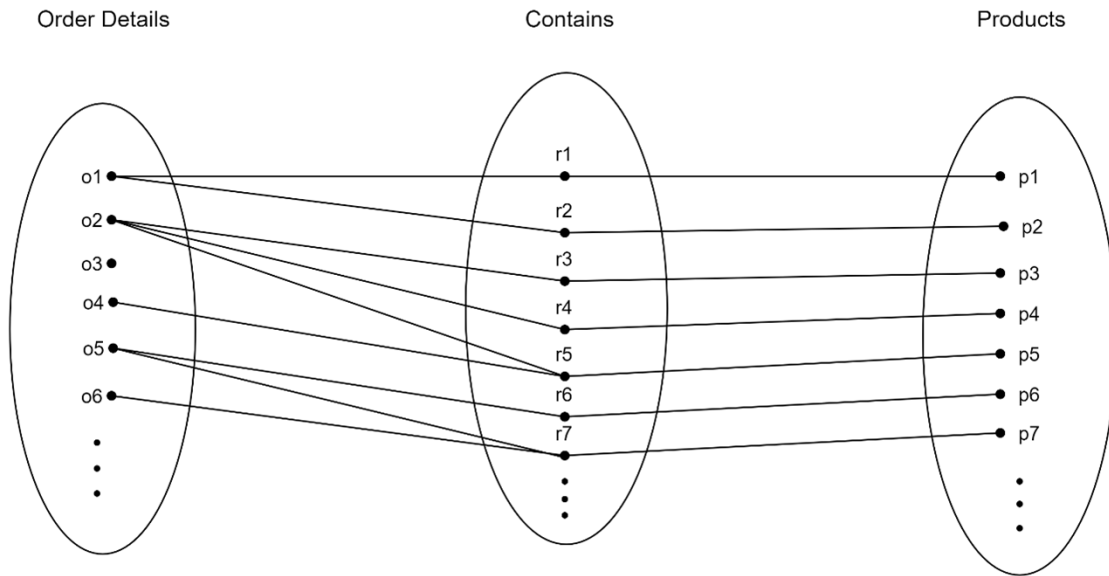


Figure 2: Order Details Contains Products

1:N

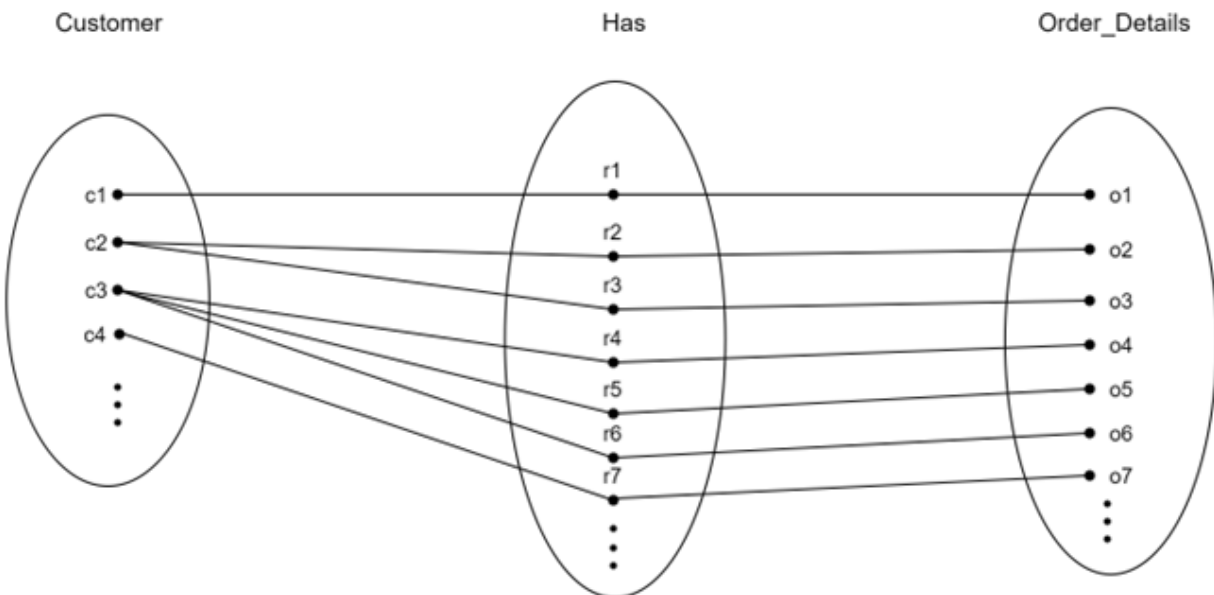


Figure 3: Customer Has Order Details

M:1

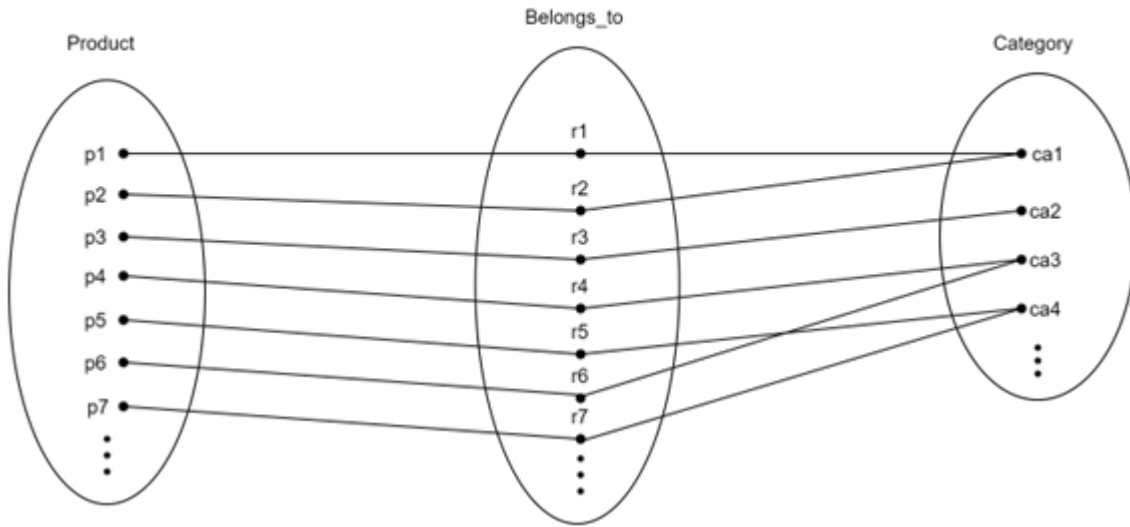


Figure 4: Product Belong to Category

1:N

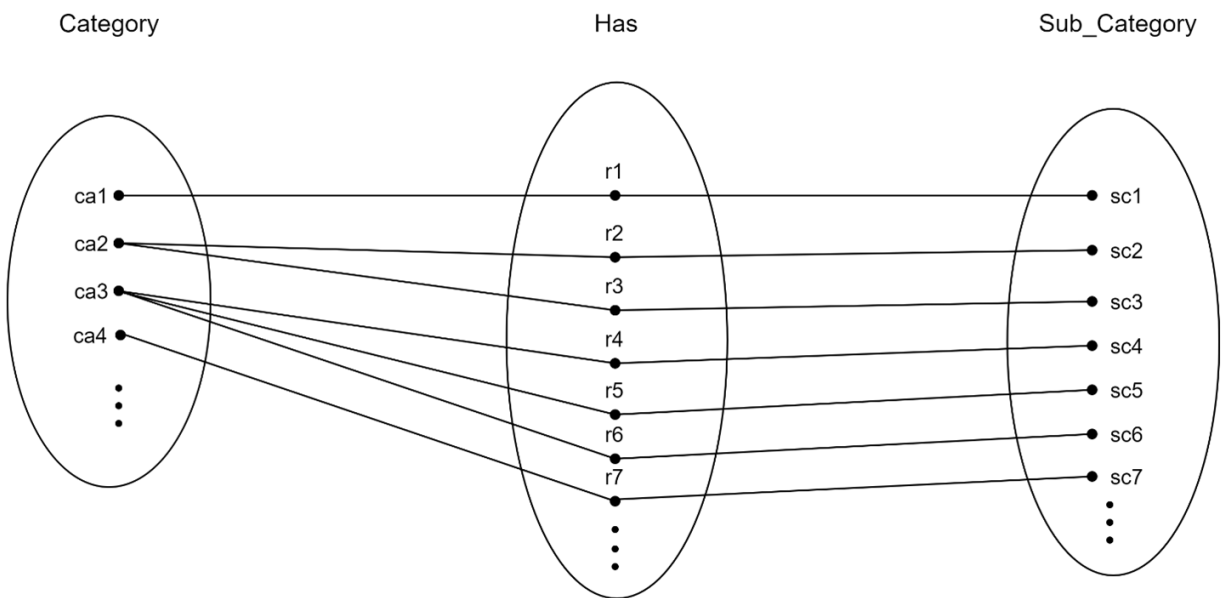


Figure 5: Category Has Sub-Categories

### 2.3.5 Product Supplied By Supplier

This relation is important for inventory management, reflecting the real-world scenario where a single supplier might provide various products, but each product is supplied by only one specific supplier. It is essential to understand how products are sourced in a database model, where tracking the source of each product is crucial. The relation helps track inventory sources, manage supplier relationships, and ensure product availability.

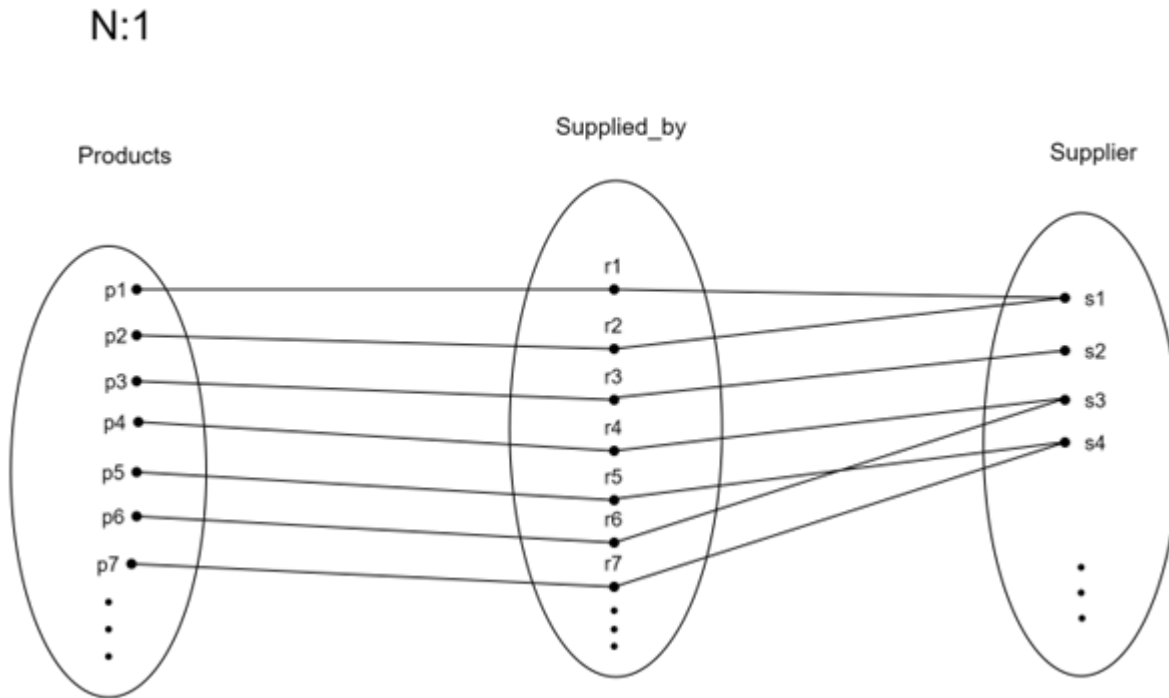


Figure 6: Products Supplied By Supplier

### 2.3.6 Product Qualifies For Discount

The relation signifying that the product is eligible for certain promotional discount enabling dynamic pricing strategies, encouraging sales, and providing customers with various savings opportunities on different products. In this context and for simplicity the relation representing one discount code or voucher that is valid to apply on multiple eligible products.

### 2.3.7 Product Has Reviews

A product's reviews are generated or provided by Customers reflecting the action of providing feedback or evaluation for a specific product or service to improve product offerings and customer service. However, one product can have multiple reviews over time.

## 2.4 Logical Schema

### 2.4.1 Customers

Customers (Cust\_ID, Cust\_Email, Cust\_First\_Name, Cust\_Last\_Name, Cust\_Country\_Code, Cust\_Phone\_Number, Cust\_Street\_Name, Cust\_Building\_Name, Cust\_Zip\_Code)

M:1

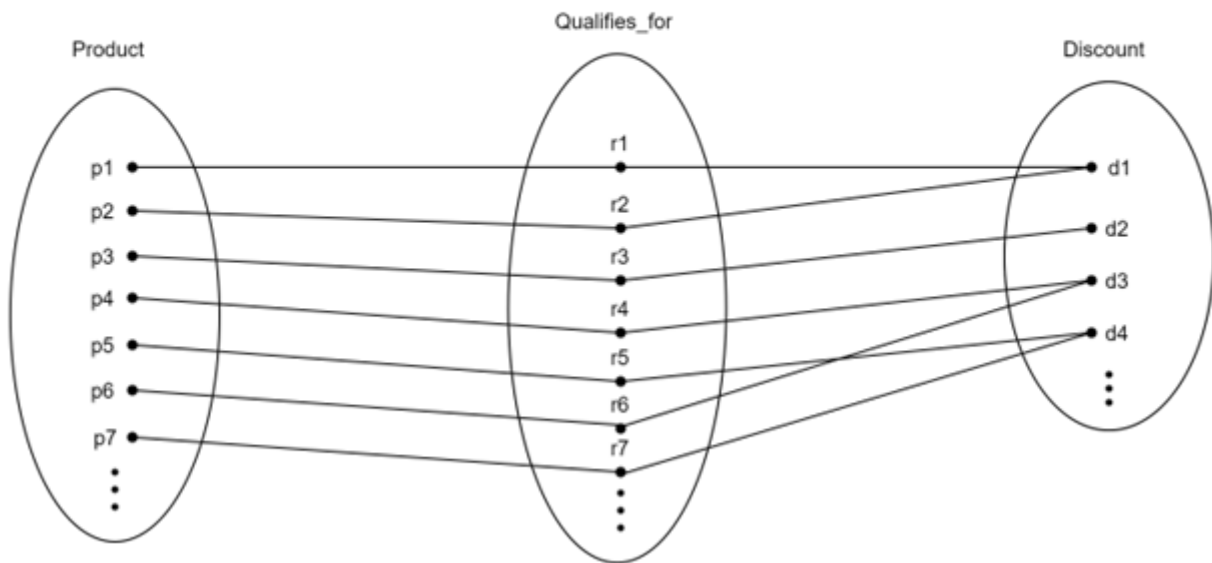


Figure 7: Product Qualifies For Discount

1:N

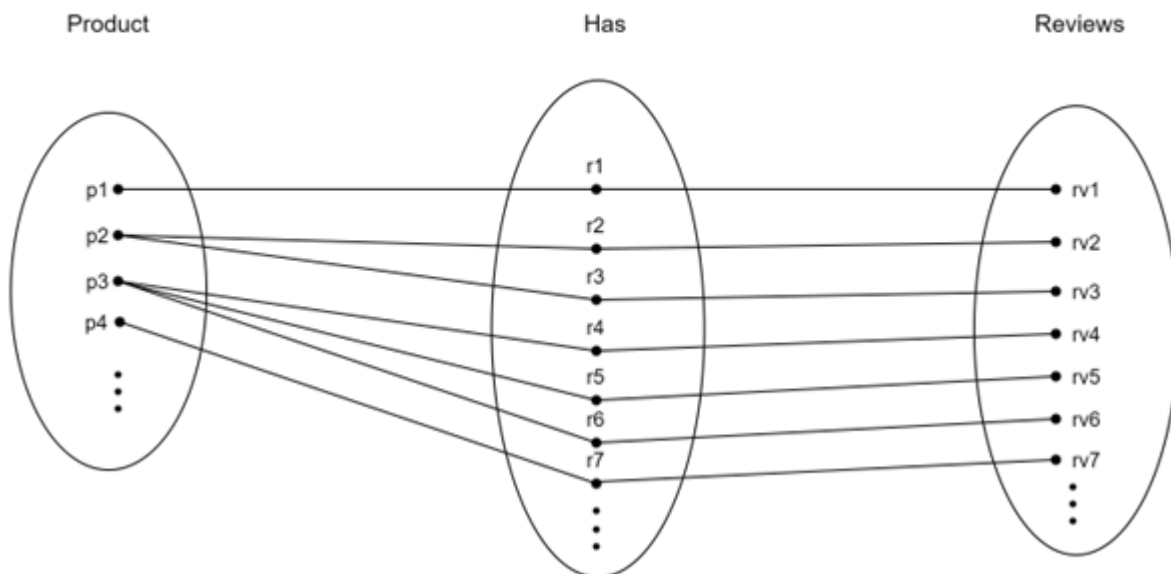


Figure 8: Product Has Reviews



### 2.4.2 Products

Products (Product\_ID, *Discount\_Code*, *Category\_ID*, *Supplier\_ID*, Product\_Name, Product\_Price, Product\_Availability)

### 2.4.3 Suppliers

Suppliers (Supplier\_ID, Supplier\_Name, Supplier\_Email, Supplier\_Status, Supplier\_Building\_Name, Supplier\_Street\_Name, Supplier\_Zip\_Code)

### 2.4.4 Order\_Details

Order\_Details (Order\_ID, *Customer\_ID*, Order\_Date, Order\_Total, Order\_Status, Shipping\_Building\_Name, Shipping\_Street\_Name, Shipping\_Zip\_Code, Street\_Name, Billing\_Building\_Name, Billing\_Street\_Name, Billing\_Zip\_Code, Payment\_Type, Payment\_Status)

### 2.4.5 Discounts

Discounts (Discount\_Code, Discount\_Status, Discount\_Amount)

### 2.4.6 Reviews

Reviews (Review\_ID, *Product\_ID*, Product\_Rating, Review\_Timestamp, Review\_Text, Review\_Likes)

### 2.4.7 Categories

Categories (Category\_ID, Category\_Name)

### 2.4.8 Many to Many : Order\_details - Product

Order\_Items (Order\_ID, Product\_ID, Quantity)

## 2.5 Physical Schema Creation

```
#Creating the db schema in SQL
con <- dbConnect(RSQLite::SQLite(), "ecommerce.db")

sql_file <- readLines("dbScript.sql")

for (sql_command in sql_file) {
  if (sql_command!=""){
    print(sql_command)
    dbExecute(con,sql_command)
    print("-----DONE-----")
  }
}
```

## 2.6 Normalisation

- First Normal Form (1NF) requires a table to have atomic values, unique rows through a primary key, and no duplicate columns.
- Second Normal Form (2NF) is achieved when a table is in 1NF and all non-key attributes are fully functional and dependent on the primary key, eliminating partial dependency on any part of a composite key.

- For a table to be in third normal form (3NF), it should not have any transitive dependencies, and all non-prime attributes are fully functionally dependent on the primary key.

### 2.6.1 Customers:

Looking at the table, we can identify the following dependencies:

Cust\_ID  $\rightarrow$  Cust\_First\_Name  
 Cust\_ID  $\rightarrow$  Cust\_Last\_Name  
 Cust\_ID  $\rightarrow$  Cust\_Building\_Number  
 Cust\_ID  $\rightarrow$  Cust\_Building\_Name  
 Cust\_ID  $\rightarrow$  Cust\_Street\_Name  
 Cust\_ID  $\rightarrow$  Cust\_Country\_Code  
 Cust\_ID  $\rightarrow$  Cust\_Email  
 Cust\_ID  $\rightarrow$  Cust\_Phone\_Number

- Based on this analysis, the table appears to be in 3NF. Each non-key attribute depends only on the primary key (Cust\_ID), and there are no transitive dependencies.

### 2.6.2 Order\_items:

Product\_ID , Order\_ID  $\rightarrow$  Quantity

Product\_ID and Order\_ID combined are representing composite key, and Quantity is functionally dependent on Product\_ID and Order\_ID. There are no transitive dependencies here.

All non-prime attributes are fully functionally dependent on the primary key.

### 2.6.3 Order\_details:

Order\_ID  $\rightarrow$  Order\_Date  
 Order\_ID  $\rightarrow$  Order\_Status  
 Order\_ID  $\rightarrow$  Payment\_Type  
 Order\_ID  $\rightarrow$  Payment\_Status  
 Order\_ID  $\rightarrow$  Cust\_ID  
 Order\_ID  $\rightarrow$  Shipping\_Building\_Name  
 Order\_ID  $\rightarrow$  Shipping\_Street\_Name  
 Order\_ID  $\rightarrow$  Shipping\_Zip\_Code  
 Order\_ID  $\rightarrow$  Billing\_Building\_Name  
 Order\_ID  $\rightarrow$  Billing\_Street\_Name  
 Order\_ID  $\rightarrow$  Billing\_Zip\_Code

Given that Cust\_ID is a foreign key in this table and that there is a separate Customers table where Cust\_ID serves as the primary key, it suggests that Cust\_ID is not functionally dependent on any other attribute within this table. It is merely referencing the primary key of another table.

Therefore, the dependency involving Cust\_ID does not violate 3rd Normal Form (3NF). The table appears to be in 3NF as long as all other attributes are functionally dependent on the primary key (Order\_ID) and not on any non-key attributes.

### 2.6.4 Product Category

Category\_ID  $\rightarrow$  Category\_Name

Category\_ID is the primary key and it uniquely identifies Category\_Name.

### 2.6.5 Product Discount

Discount\_Code  $\rightarrow$  Discount\_Amount  
Discount\_Code  $\rightarrow$  Discount\_Status

Analyzing the functional dependencies, it is found that Discount\_Code is the primary key and it uniquely identifies Discount\_Amount and Discount\_Status.

Each Discount\_Code corresponds to a specific Discount\_Amount and Discount\_Status, ensuring that the table adheres to the principles of 3rd Normal Form (3NF). There are also no transitive dependencies or non-key attributes determining other attributes within the table.

### 2.6.6 Products

Product\_ID  $\rightarrow$  Product\_Name  
Product\_ID  $\rightarrow$  Product\_Price  
Product\_ID  $\rightarrow$  Product\_Availability

*Missed FKs*

Examining the attributes' functional dependencies, Product\_ID emerges as the primary key, in which Product\_Name, Product\_Price and Product\_Availability appears to be functionally dependent on it.

### 2.6.7 Reviews

Review\_ID  $\rightarrow$  Review\_Timestamp  
Review\_ID  $\rightarrow$  Product\_Rating  
Review\_ID  $\rightarrow$  Review\_Text  
Review\_ID  $\rightarrow$  Review\_Likes

In the Reviews table, Review\_ID acts as the primary key, ensuring uniqueness. Attributes like Review\_Timestamp, Product\_Rating, Review\_Text and Review\_Likes are dependent on Review\_ID.

### 2.6.8 Suppliers

Supplier\_ID  $\rightarrow$  Supplier\_Name  
Supplier\_ID  $\rightarrow$  Supplier\_Building\_Name  
Supplier\_ID  $\rightarrow$  Supplier\_Building\_Number  
Supplier\_ID  $\rightarrow$  Supplier\_Street\_Name  
Supplier\_ID  $\rightarrow$  Supplier\_Zip\_Code  
Supplier\_ID  $\rightarrow$  Supplier\_Email  
Supplier\_ID  $\rightarrow$  Supplier\_Status

In the Suppliers table, Supplier\_ID acts as the primary key. Attributes such as Supplier\_Name, Supplier\_Building\_Name, Supplier\_Building\_Number, Supplier\_Street\_Name, Supplier\_Zip\_Code, Supplier\_Email and Supplier\_Status are dependent on Supplier\_ID.

Given these functional dependencies where each attribute seems to be functionally dependent on the primary key, with no non-key attributes determining other, all tables appears to adhere to the principles of 3rd Normal Form (3NF).

## 3 Part 2: Data Generation and Management

### 3.1 Synthetic Data Generation

After the agreement on the schema mentioned in the previous section, the team started to generate synthetic data that to some extent, imitated realistic e-commerce as much as possible.

ChatGPT has been used as the main tool for this step as an alternative to Mockaroo, as the former produces more structural and logical data than the latter. [Appendix 1,2]. For tables consisting of foreign keys, R code has been used for assignment based of logic set by the team.

*Appendix — Take Snippt from data\_update file*

### 3.2 Data Import and Quality Assurance

To enhance data quality for e-commerce analysis and validate the csv data generated, we use R to perform 4 main checks (Refer data\_validate.R): • Primary key integrity check • Duplicate entry check • Email id pattern check • Phone number pattern check

Once the data has gone through all the checks, the updated csv files will be used to import into SQL. Importing the data into SQL is also done in R (Refer data\_update.R)

This method focused on refining datasets for accurate analysis by identifying and eliminating invalid entries and duplicates, thereby maintaining the dataset's uniqueness and reliability before inserting into Data Base.

## 4 Part 3: Data Pipeline Generation

To efficiently collaborate on this assignment, we have uploaded the project to Github. This was done so that the team could track changes, follow progress, collaborate on bug fixes etc. Moving the project to Github also allowed for implementation of version control and continuous integration.

Using Github Workflows, we were able to automate the process of validating data, updating data and data analysis. The workflow is triggered on pushing new data to the project. Once all 3 steps are completed successfully, the new files are committed and pushed to the project.

Figure below shows the new data reflecting in our analyses and sql db after being picked up by the workflow.

### 4.1 Github Repository Creation

The pipeline generation process consists of two phases. In the first phase, github repository has been created<sup>[1]</sup> followed by integrating the posit cloud project. Additionally, all teams members have been added as collaborators.

### 4.2 Automated Workflow

Second phase was the automation process utilising workflow that trigger on specific events like push request to perform data updates, validation, and analysis. This will pick up any new data added to the data base and subsequently perform the required validation and analysis as illustrated in *Figures* .

The below figure shows the record count after 1st run of workflow (49) and then after second run of workflow (150).

Following snippet is Timestamps that showed as an evidence.

## 5 Part 4: Data Analysis and Reporting with Quarto in R

Information regarding product ratings, sales revenue, customer distribution, and order status was obtained through the establishment of a connection to an SQLite database and the execution of SQL queries. Subsequently, data visualisation techniques were used to identify the distribution and patterns within the data, offering insights into e-commerce operational dimensions.

The graph shows the distribution of product ratings from 0 to 5 coloured by category name.

The graph above shows the top 5 products in terms of revenue. [How Revenue is calculated by using join statement]

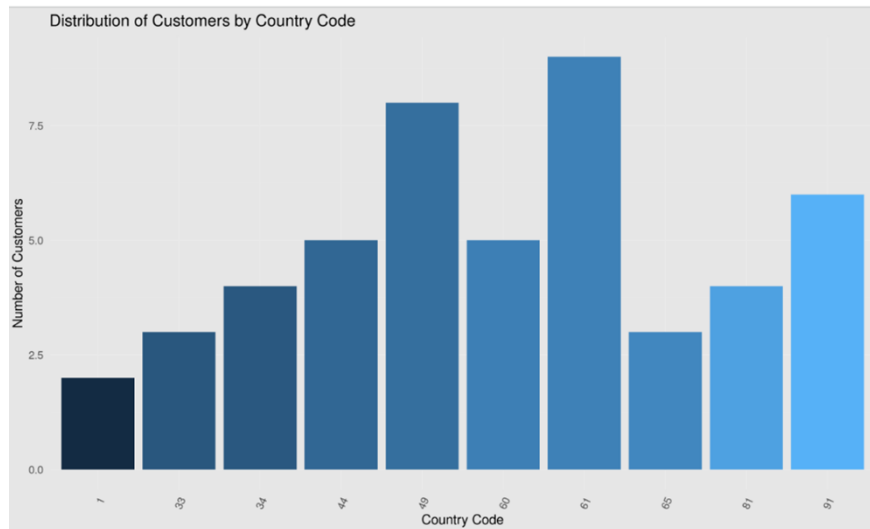


Figure 9: The histogram of customers data for 50 records

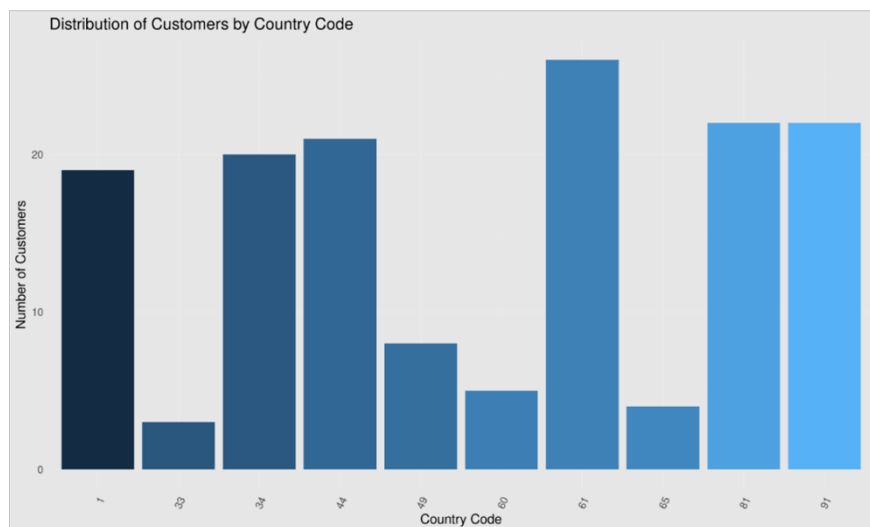


Figure 10: The histogram after workflow run and inserted additional 100 records

```

sqlite> .quit
/cloud/project$ sqlite3 ecommerce.db
SQLite version 3.31.1 2020-01-27 19:55:54
Enter ".help" for usage hints.
sqlite> select count(*) from Customers
...> ;
49
sqlite> .quit
/cloud/project$ sqlite3 ecommerce.db
SQLite version 3.31.1 2020-01-27 19:55:54
Enter ".help" for usage hints.
sqlite> select count(*) Customers;
1
sqlite> select count(*) from Customers;
150
sqlite> █

```

Figure 11: Record count after data update using workflow

All workflows
Filter workflow runs

Showing runs from all workflows

**Help us improve GitHub Actions**  
Tell us how to make GitHub Actions work better for you with three quick questions.
Give feedback

61 workflow runs	Event ▾	Status ▾	Branch ▾	Actor ▾
<b>added new records in Customers</b> workflow for Task3.2 #61: Commit <a href="#">5429e4c</a> pushed by abigail-g	main		19 minutes ago 3m 13s	...
<b>changed column name in customers.csv</b> workflow for Task3.2 #60: Commit <a href="#">392bd3a</a> pushed by abigail-g	main		25 minutes ago 3m 21s	...

Figure 12: Workflow Timestamps

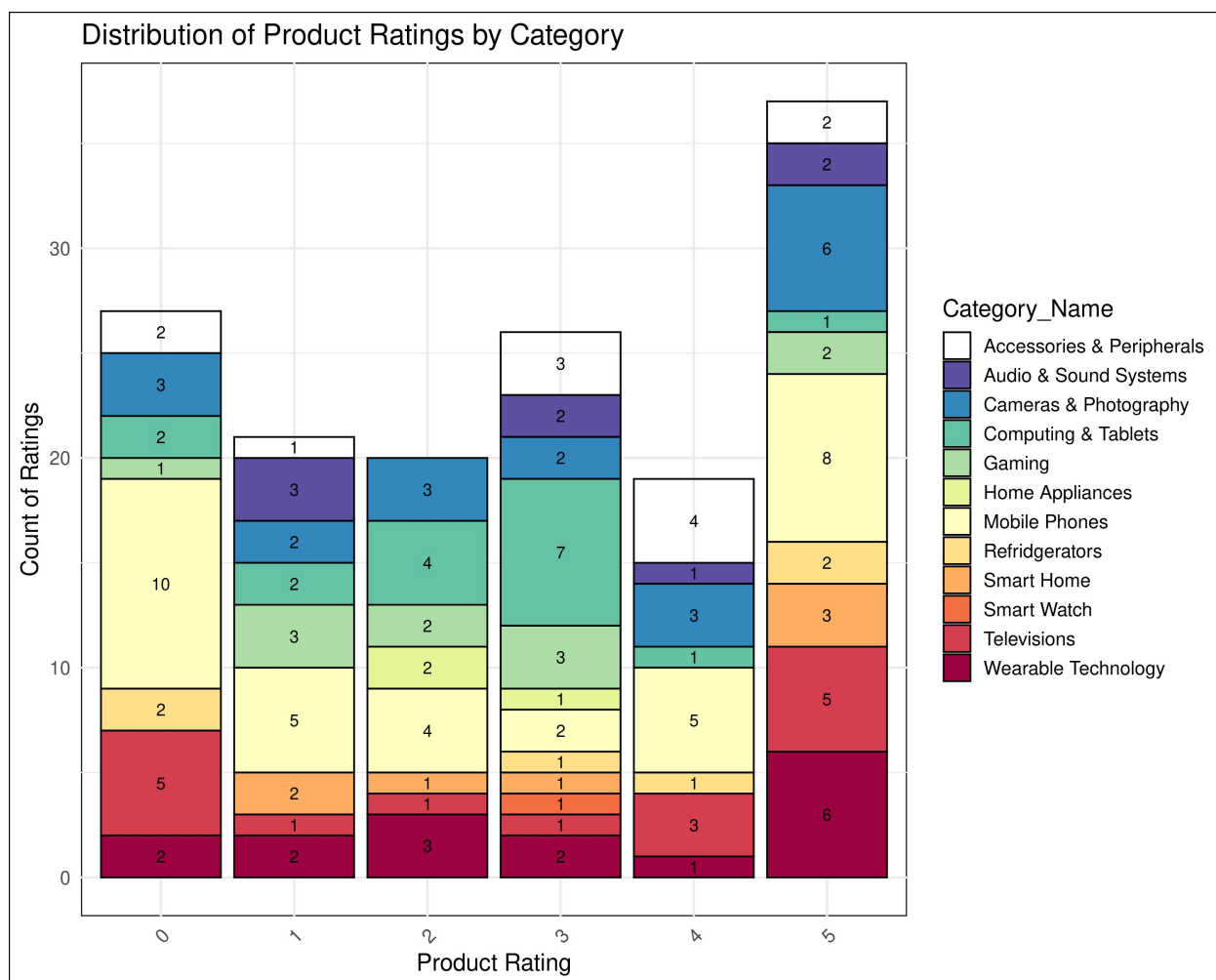


Figure 13: Distribution of Product Ratings by Category

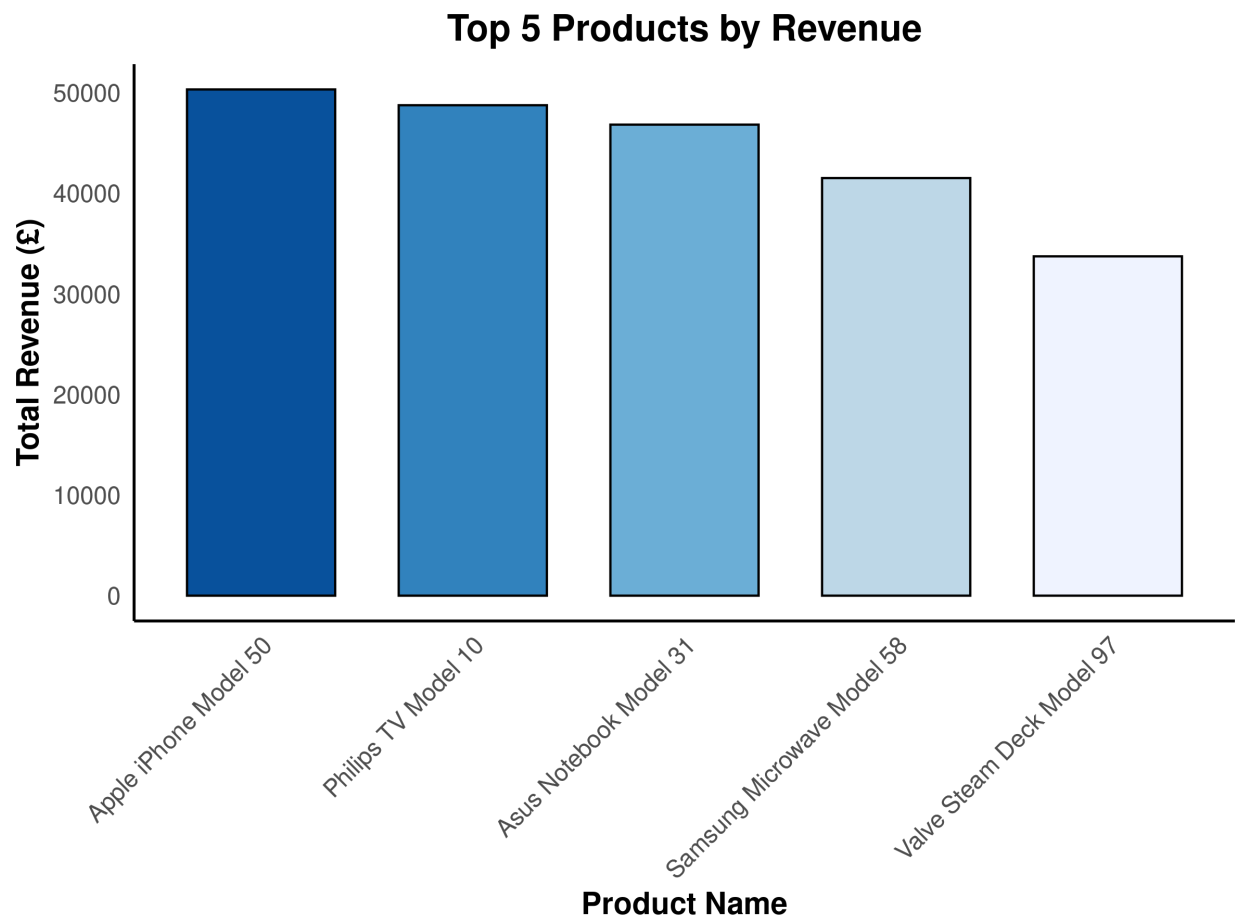


Figure 14: Top 5 Products by Revenue



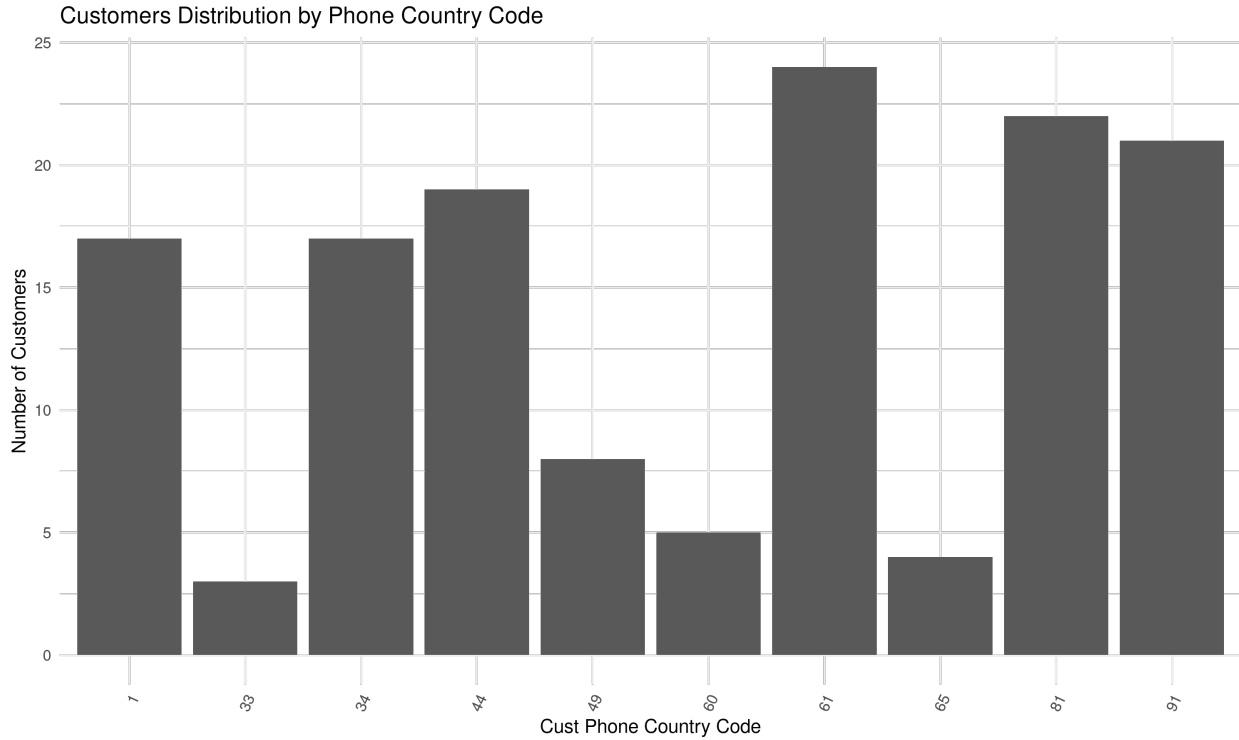


Figure 15: Distribution of Customers by Phone Country Code

The graph above shows the distribution of customers by phone country code.

The graph above shows the order status count of shipped, processing, delivered and cancelled orders.

## 6 Conclusion

Upon completion of this project, valuable insights are gained regarding the challenges and methodologies associated with an e-commerce context, including the design of databases, analysis of data, and presentation of findings in a clear and impactful manner. Ultimately, this project serves as a reference for future e-commerce projects in particular and data-driven projects in general, providing valuable insight into the modern data management landscape.

## 7 References

[1][Github Repo](<https://github.com/abigail-g/E-commerceDB>)

## 8 Appendices

### 8.1 Appendix 1

### 8.2 Appendix 2

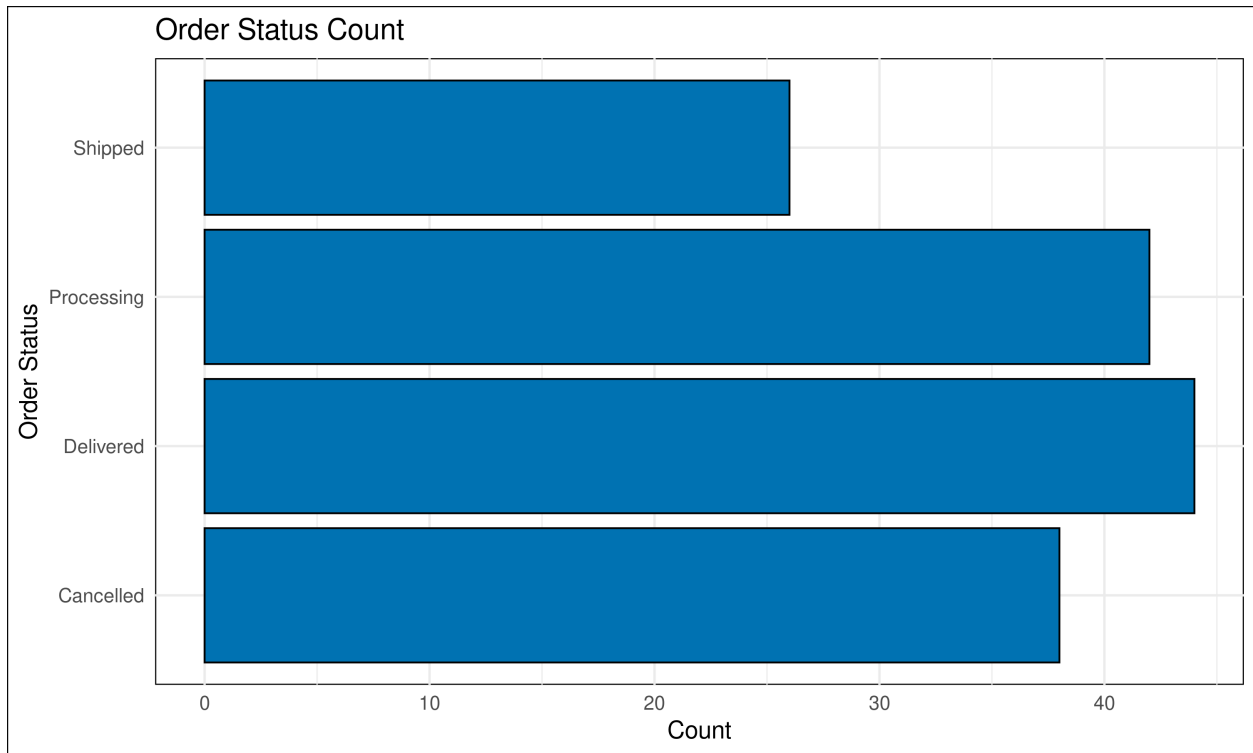


Figure 16: Order Status Count

#### ChatGPT 4 ▾

**You**

```
CREATE TABLE 'Products' (
  'Product_ID' VARCHAR(250) PRIMARY KEY,
  'Product_Name' VARCHAR(250) NOT NULL,
  'Product_Price' FLOAT(10,2) NOT NULL,
  'Product_Availability' VARCHAR(25) );
```

generate a csv file with 150 observations, each product ID needs to be P with random 7 digits, product price must range between 50 to 2000. The product name must be from these category including televisions, computing & tablets, mobile phones, home appliances, audio&sound systems, cameras and photography, gaming (PS5, Xbox etc), Smart home, Wearable Technology, Accessories & Peripherals , every column must be unique, the product name must be unique

**ChatGPT**


The CSV file with 150 observations has been generated successfully. Each product ID consists of a "P" followed by 7 random digits, with product prices ranging between \$50 and \$2000. The product names are unique and categorized into televisions, computing & tablets, mobile phones, home appliances, audio & sound systems, cameras and photography, gaming, smart home, wearable technology, and accessories & peripherals. The availability of each product is also noted.

You can download the CSV file from the following link: [products.csv](#). [\[-\]](#)

Figure 17: ChatGPT\_1

ChatGPT 4 ▾

 You

 Customers.csv  
Spreadsheet

can you generate more data, from 50 to 150, do not duplicate, make sure that each columns are distinct

 ChatGPT

The uploaded CSV file, "Customers.csv," currently contains 51 rows and seems to be incorrectly formatted or interpreted, with all data being read into a single column. This might be due to the delimiters or the structure of the file itself.

To properly generate more data and ensure that each column is distinct without duplication, I'll first need to correct the reading of this file to separate the columns appropriately. Let's try to read the file again with a more suitable delimiter. [\[-\]](#)

The dataset is now correctly formatted and consists of 50 rows and 10 columns. Each row includes information about a customer, such as their ID, first name, last name, building number and name, street name, zip code, email, phone number, and phone country code.

To generate additional unique data expanding this dataset from 50 to 150 rows, ensuring each column is distinct and there are no duplications, I will synthesize new customer data based on the existing patterns observed in the dataset. Let's proceed with generating this additional data. [\[-\]](#)

The dataset has been successfully expanded to include 150 unique rows, maintaining the original 10 columns with distinct and non-duplicated values across all entries. This enriched dataset now comprises a wider variety of customer information, ready for any intended analysis or processing. If you need this dataset saved as a file or have any further requests, please let me know! [\[-\]](#)

Figure 18: ChatGPT\_\_2