Ecomm Report

Group 15

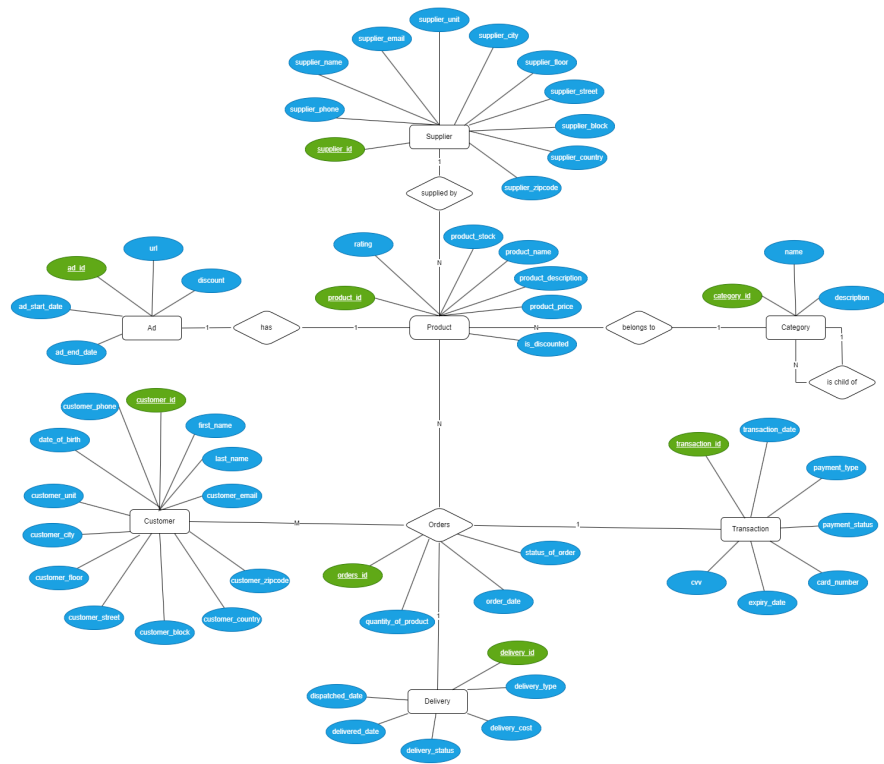## Part 1: Database Design and Implementation

Task 1.1: E-R Diagram Design

We are basing our database design and workflows on a real-world e-commerce data environment, inspired by Amazon. The e-commerce company only uses its own courier.
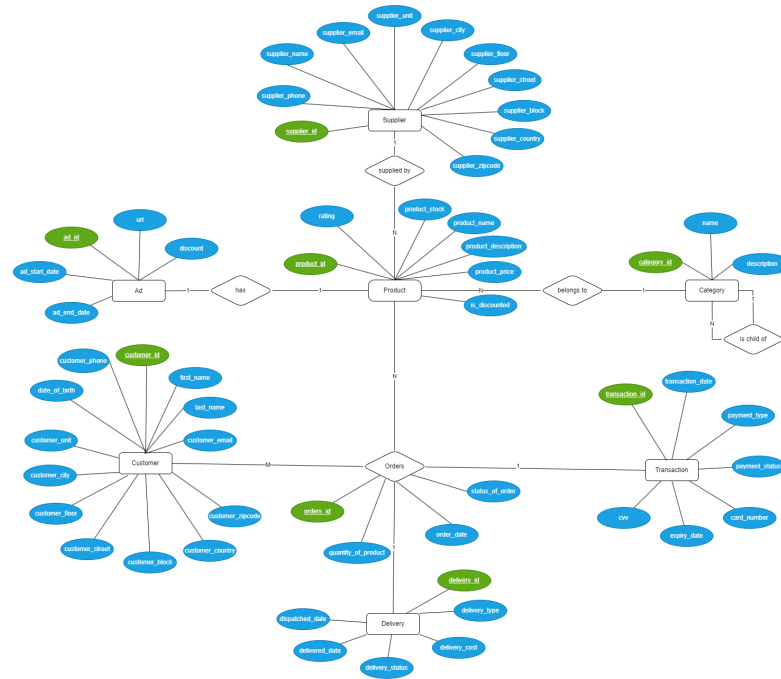
The ER diagram is in 1st normalised form.

1. We assume that 1 product must belong to 1 category, and a category must at least have 1 or more product(s).

2. Many categories can be the child of a parent category.

3. Each product can only have 1 advertisement, and each advertisement is always for 1 product.

4. Each product has a unique supplier, but a supplier can supply many types of products.

5. Each customer only has 1 address.

6. 'Order' is a relationship between 'Customer', 'Product' and 'Transaction' entities.

7. 1 order can have many products, and a type of product can be in multiple orders.

8. A customer can make multiple orders, but each order must belong to 1 customer.

9. Each order may or may not have a transaction. An order must only be paid in a single transaction, by credit or debit card.

10. The customer who makes the order is the one who pays as well.

11. Each order may or may not have a delivery. One delivery is made for only one order.

«««

Supplier

supplier_unit
supplier_email
supplier_city
supplier_name
supplier_floor
supplier_street
supplier_phone
supplier_block
supplier_country
supplier_id
supplier_zipcode

supplied by

N

rating
product_stock
product_name
name
product_description
ad_id
url
product_id
product_price
category_id
description
discount
ad_start_date
Ad
has
Product
belongs to
Category
ad_end_date
is_discounted

is child of

N

customer_phone
customer_id
first_name
date_of_birth
transaction_id
transaction_date
last_name
payment_type
customer_unit
customer_email
payment_status
customer_city
Customer
M
Orders
1
Transaction
customer_floor
status_of_order
cvv
card_number
customer_street
orders_id
order_date
expiry_date
customer_block
customer_country
customer_zipcode
quantity_of_product

delivery_id
dispatched_date
delivery_type
Delivery
delivered_date
delivery_cost
delivery_status

=======

»»»> 17eacce281bbfdbef578c3a1a37184852a8279bb

Relationships between key entities are shown in Figure 2. We can tell that:

1. One-to-Many relationship: CUSTOMER and TRANSACTION, SUPPLIER and PRODUCT, CATEGORY and PRODUCT.

2. Many-to-Many relationship: CUSTOMER and PRODUCT

3. One-to-One relationship: TRANSACTION and DELIVERY, ADVERTISEMENT and Product
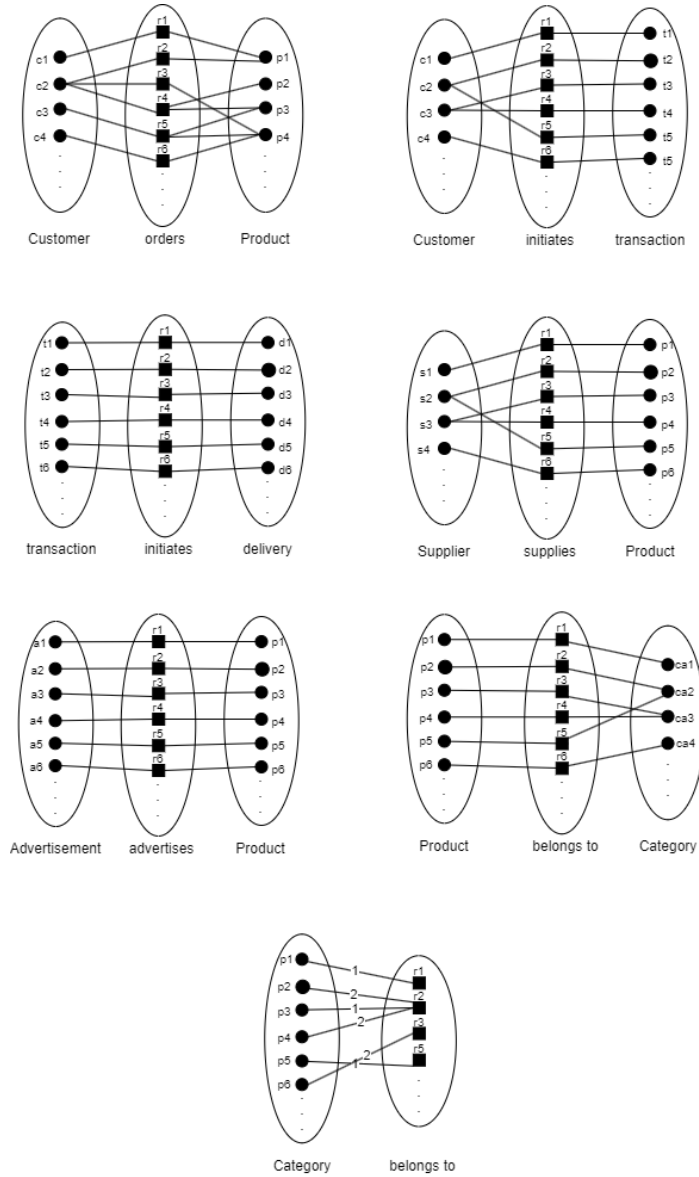
4. Recursive relationship: CATEGORY

Figure 1: Relationship Set

Task 1.2: SQL Database Schema Creation

To convert 1st normalised form to 3rd normalised form, we have removed all partial and transitive dependencies in the entity tables.

As a result, we have obtained these tables in the logical schema:

1. CUSTOMER (customer_id, first_name, last_name, date_of_birth, customer_phone, customer_email, customer_unit, customer_floor, customer_block, customer_street, customer_city, customer_country, customer_zipcode)

2. PRODUCT (product_id, supplier_id, category_id, product_name, product_description, product_price, product_stock, rating, is_discounted)

3. SUPPLIER (supplier_id, supplier_name, supplier_phone, supplier_email, supplier_unit, supplier_floor, supplier_block, supplier_street, supplier_city, supplier_country, supplier_zipcode)

4. ADVERTISEMENT (ad_id, product_id, url, ad_start_date, ad_end_date, discount)

5. TRANSACTIONS (transactions_id, order_id, transaction_date, payment_type, card_number, expiry_date, cvv, payment_status)

6. CATEGORY (category_id, name, description, parent_category_id)

7. ORDER (order_id, customer_id, order_date, status_of_order)

8. ORDER_DETAILS (order_id, product_id, quantity_of_product)

9. DELIVERY(delivery_id, order_id, delivery_type, dispatched_date, delivered_date, delivery_status, delivery_cost)

We have made these assumptions:

1. Customer address is the same as their billing and delivery address, so it is only stored once in the Customer table.

2. Delivery cost only depends on delivery type (e.g. standard, premium), not the delivery address. We assume that cost of delivery type is fixed for whole nation-wide delivery.

3. Delivered_date is an input of the actual timedate when the parcel reaches the customer. It is not a calculated field (ETA) based on delivery type and order date, because there may be delays for which the parcel reaches customer. Delivered_date must be after dispatched_date

4. Delivery status is dependent on dispatched date and delivered date: "Pending" (both dispatched_date and delivered_date is NULL), "shipped" means the parcel is on the way (delivered_date is NULL), "delivered" means the parcel has reached the customer. We want to capture multiple values of delivery status (hence, for each delivery, there are multiple rows showing different statuses. This helps us to find how long has a delivery stayed in a particular status)

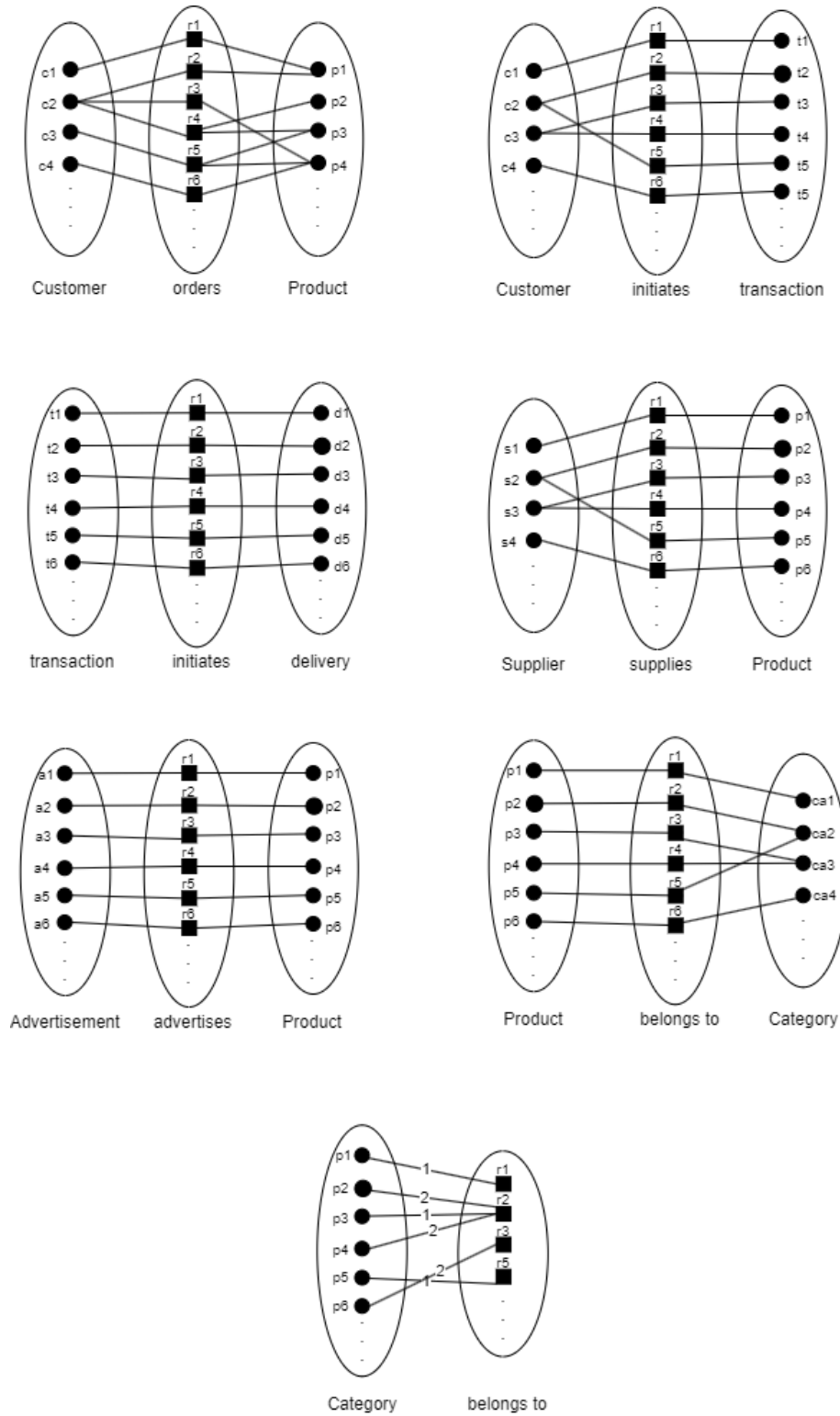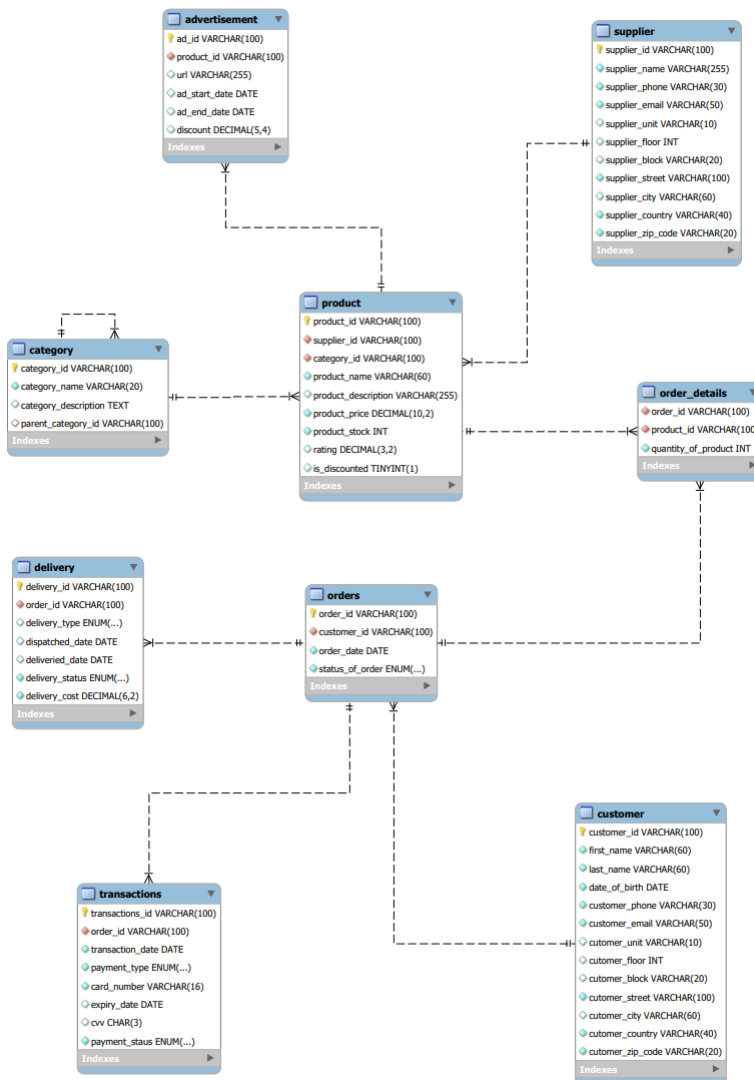Additionally, Figure 2 shows the relationship set of our database.
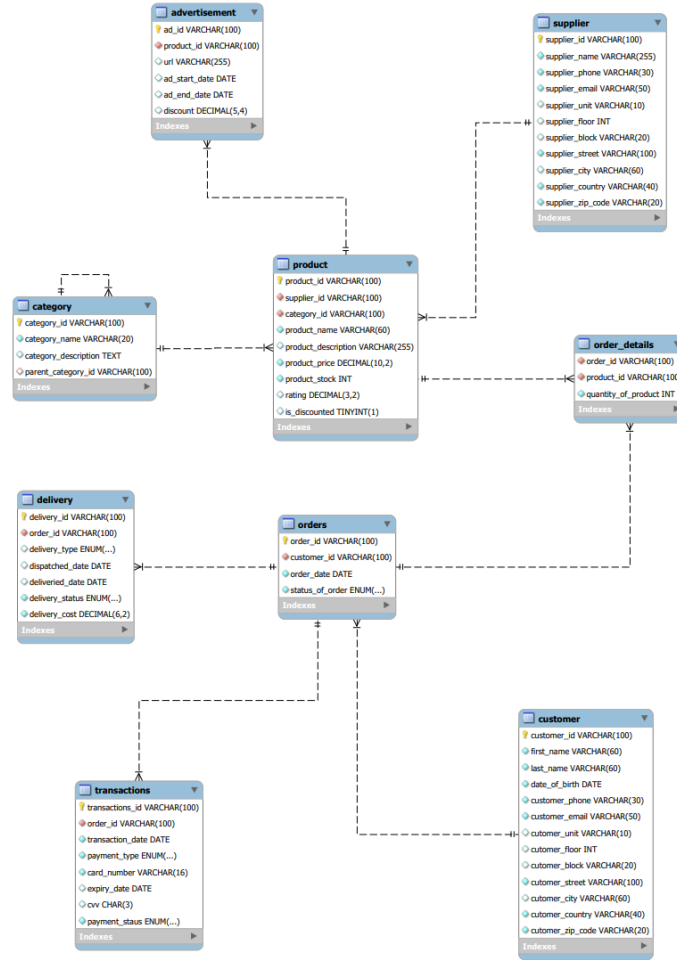
Figure 2: Relationship Set

In the database schema depicted in Figure 3, based on previous assumptions as well as the actual situation, the following data structuring conventions have been established:

1. Primary keys are designated in the VARCHAR format to accommodate future scalability and uniqueness, indicated by a yellow icon.

2. Date attributes are uniform of the DATE type, adhering to the yyyy-mm-dd format.

3. Customer and supplier contact information, including phone numbers and emails, are unique across all entries.

4. Type attributes are constrained by the ENUM type, restricting values to a predefined set of keywords.

5. Descriptive text fields utilize the TEXT data type to avoid truncation.

6. The advertisement table presumes a singular, active advertisement per product with defined start and end dates, and an associated discount percentage. This discount is recorded as DECIMAL(6, 4) to ensure precision up to four decimal places.

advertisement
- ad_id VARCHAR(100)
- product_id VARCHAR(100)
- url VARCHAR(255)
- ad_start_date DATE
- ad_end_date DATE
- discount DECIMAL(5,4)
- Indexes

supplier
- supplier_id VARCHAR(100)
- supplier_name VARCHAR(255)
- supplier_phone VARCHAR(30)
- supplier_email VARCHAR(50)
- supplier_unit VARCHAR(10)
- supplier_floor INT
- supplier_block VARCHAR(20)
- supplier_street VARCHAR(100)
- supplier_city VARCHAR(60)
- supplier_country VARCHAR(40)
- supplier_zip_code VARCHAR(20)
- Indexes

product
- product_id VARCHAR(100)
- supplier_id VARCHAR(100)
- category_id VARCHAR(100)
- product_name VARCHAR(60)
- product_description VARCHAR(255)
- product_price DECIMAL(10,2)
- product_stock INT
- rating DECIMAL(3,2)
- is_discounted TINYINT(1)
- Indexes

category
- category_id VARCHAR(100)
- category_name VARCHAR(20)
- category_description TEXT
- parent_category_id VARCHAR(100)
- Indexes

order_details
- order_id VARCHAR(100)
- product_id VARCHAR(100)
- quantity_of_product INT
- Indexes

delivery
- delivery_id VARCHAR(100)
- order_id VARCHAR(100)
- delivery_type ENUM(...)
- dispatched_date DATE
- deliveried_date DATE
- delivery_status ENUM(...)
- delivery_cost DECIMAL(6,2)
- Indexes

orders
- order_id VARCHAR(100)
- customer_id VARCHAR(100)
- order_date DATE
- status_of_order ENUM(...)
- Indexes

transactions
- transactions_id VARCHAR(100)
- order_id VARCHAR(100)
- transaction_date DATE
- payment_type ENUM(...)
- card_number VARCHAR(16)
- expiry_date DATE
- cvv CHAR(3)
- payment_staus ENUM(...)
- Indexes

customer
- customer_id VARCHAR(100)
- first_name VARCHAR(60)
- last_name VARCHAR(60)
- date_of_birth DATE
- customer_phone VARCHAR(30)
- customer_email VARCHAR(50)
- cutomer_unit VARCHAR(10)
- cutomer_floor INT
- cutomer_block VARCHAR(20)
- cutomer_street VARCHAR(100)
- cutomer_city VARCHAR(60)
- cutomer_country VARCHAR(40)
- cutomer_zip_code VARCHAR(20)
- Indexes

≪≪≪< HEAD   =======

»»»> 17eacce281bbfdbef578c3a1a37184852a8279bb

Part 2: Data Generation and Management

Task 2.1: Synthetic Data Generation

This section aims to generate synthetic data for an e-commerce business based on the structure of the database outlined in Part 1. To accomplish this task, it is crucial to thoroughly understand and identify the main components, including entities, attributes, primary keys, and foreign keys, within the data tables from the schema provided.

As per Part 1, there are nine tables for which data needs to be created in CSV format. These tables include advertisement, category, customer, delivery, order_details, orders, product, supplier, and transactions. AI and LLM tools such as ChatGPT and Mockaroo are utilized to

generate the synthetic database. This is achieved by providing commands to these tools to obtain data. Additionally, we have decided to generate 1,000 samples for each table to ensure an adequate amount of data is available for analyzing the e-commerce business thoroughly.

Task 2.2: Data Import and Quality Assurance

After creating the data, we loaded it into our database and performed various procedures to ensure the quality and integrity of the data. It was essential to establish a strong foundation for the analysis phase. Through the integrated use of R and SQL codes, we preserved the consistency of data types and connections, verified the uniqueness of attributes, and validated foreign keys. This careful approach guarantees that our synthetic data faithfully mirrors a real e-commerce database.

Data Import Workflow

We initiated a structured workflow within RStudio and Posit Cloud for an organized management of our GitHub repository. With a complete set of tables ready for import, we aimed to create a robust e-commerce environment.

Ensuring Data Quality and Integrity

Database and Table Creation: Using R scripts, we set up tables according to our schema, ensuring the structure adhered to our design principles.

1. Uniqueness Constraints: Uniqueness constraints were placed on critical attributes like customer emails and phone numbers to eliminate duplicates.

2. Referential Integrity Checks: We confirmed that all foreign keys corresponded to existing primary keys in their respective tables.

3. Data Import Execution: We connected to the database for importing data. Errors encountered were logged and resolved, ensuring accurate data import.

This method not only facilitated the efficient transfer of our synthetic data but also ensured its consistency and reliability, providing a firm foundation for subsequent analysis and reporting.

Part 3: Data Pipeline Generation

Task 3.1 GitHub Repository and Workflow Setup

Objective:

Utilize GitHub to streamline the database creation process and maintain up-to-date access for all team members.

GitHub Repository Structure:

We created ' github.com/diego-karsa/Data-Management-Group-15 ' repository for project management and version control. This repository contains all vital elements, including the SQL schema, setup scripts, query scripts, project documentation, and data analysis plots.
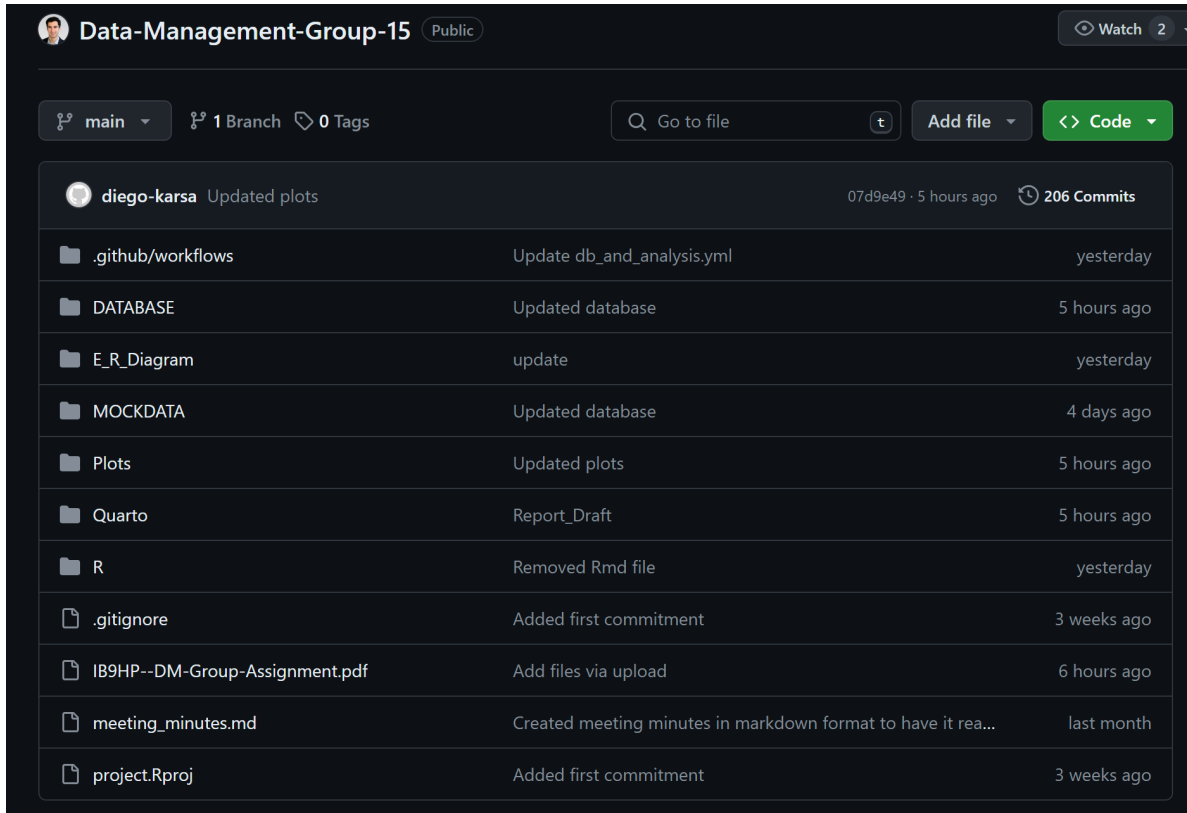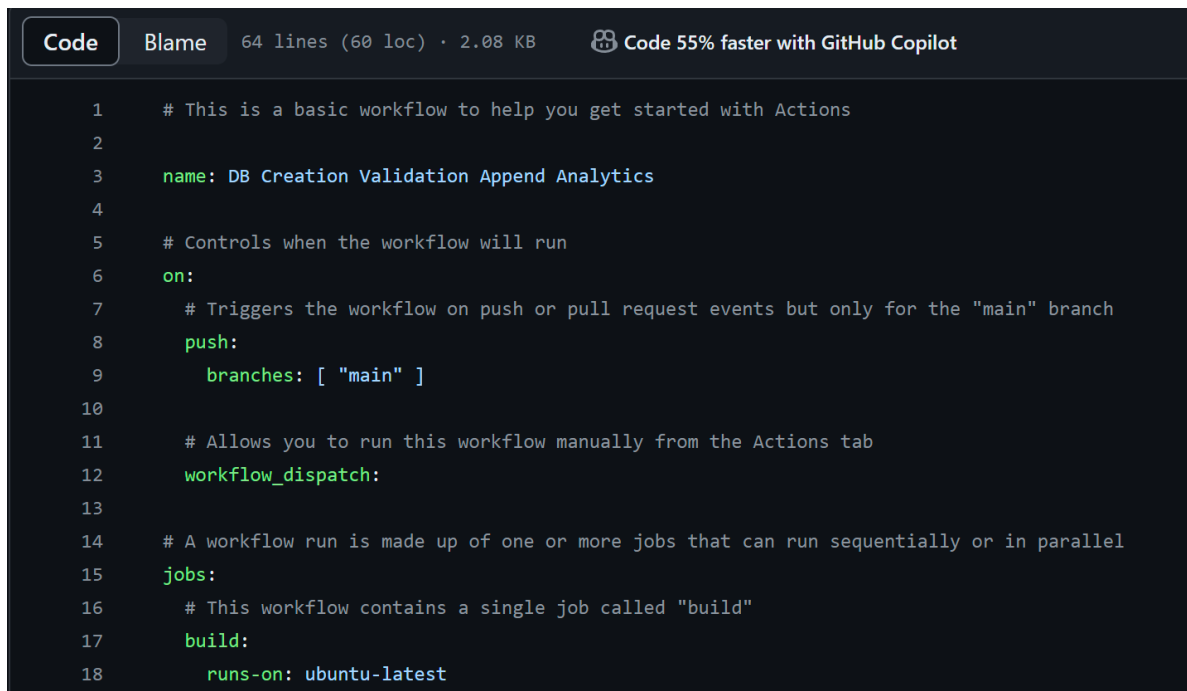

Figure 3: GitHub Repository

Task 3.2 GitHub Actions for Continuous Integration

Objective:

Implement GitHub Actions to automate crucial tasks like data validation, database updates, and initial data analysis.

Workflow Setup:

A Github action workflow named 'db_and_analysis.yml' was implemented to automate our data pipeline. Triggered by push events or manually, this workflow facilitates data validation, updates, and analysis on an Ubuntu server.

```
Code    Blame    64 lines (60 loc) · 2.08 KB        🦑 Code 55% faster with GitHub Copilot

  1       # This is a basic workflow to help you get started with Actions
  2
  3       name: DB Creation Validation Append Analytics
  4
  5       # Controls when the workflow will run
  6       on:
  7         # Triggers the workflow on push or pull request events but only for the "main" branch
  8         push:
  9           branches: [ "main" ]
 10
 11         # Allows you to run this workflow manually from the Actions tab
 12         workflow_dispatch:
 13
 14       # A workflow run is made up of one or more jobs that can run sequentially or in parallel
 15       jobs:
 16         # This workflow contains a single job called "build"
 17         build:
 18           runs-on: ubuntu-latest
```

Figure 4: Workflow Setup

Workflow Tasks:

1. Setting up R environment: The latest repository code is retrieved, and the R environment is set up with necessary package caching.

2. Package installations and database creation: Required R packages are installed, and the 'db_creation.R' script builds the database, followed by the addition of changes to the 'DATABASE' folder.

13

```yaml
16      # This workflow contains a single job called "build"
17      build:
18        runs-on: ubuntu-latest
19        steps:
20          - name: Checkout code
21            uses: actions/checkout@v2
22          - name: Setup R environment
23            uses: r-lib/actions/setup-r@v2
24            with:
25              r-version: '4.2.0'
26          - name: Cache R packages
27            uses: actions/cache@v2
28            with:
29              path: ${{ env.R_LIBS_USER }}
30              key: ${{ runner.os }}-r-${{ hashFiles('**/lockfile') }}
31              restore-keys: |
32                ${{ runner.os }}-r-
33          - name: Install packages
34            if: steps.cache.outputs.cache-hit != 'true'
35            run: |
36              Rscript -e 'install.packages(c("RSQLite", "readr", "purrr", "ggplot2",
37              "dplyr", "lubridate", "stringr", "forcats", "scales"))'
38          - name: Execute R script db creation update
39            run: |
40              Rscript R/db_creation.R
```

3. Data analysis and visualizations: The 'analysis.R' script runs for data analysis and visualization creation, with outputs added to the 'Plots' folder and then to the repository.

4. Committing changes: Changes are committed with a descriptive message and pushed to the main branch.

```
41          - name: Add files
42            run: |
43              git config --global user.email "diego.karsaclian@gmail.com"
44              git config --global user.name "diego-karsa"
45              git add --all DATABASE/
46          - name: Commit files
47            run: |
48              git commit -m "Updated database"
49          - name: Execute R script analysis
50            run: |
51              Rscript R/analysis.R
52          - name: Add files
53            run: |
54              git config --global user.email "diego.karsaclian@gmail.com"
55              git config --global user.name "diego-karsa"
56              git add --all Plots/
57          - name: Commit files
58            run: |
59              git commit -m "Updated plots"
60          - name: Push changes
61            uses: ad-m/github-push-action@v0.6.0
62            with:
63                github_token: ${{ secrets.GITHUB_TOKEN }}
64                branch: main
```

Figure 5: Workflow for Data Analysis using R and Pushing Changes

Through this workflow, we have automated the data management process, enhancing the efficiency and reliability of our system, showcasing GitHub Actions as a powerful tool for continuous integration in data-driven projects.

Part 4: Data Analysis and Reporting with Quarto in R

Part 4.1: Advanced Data Analysis in R

Objective:

Our objective was to conduct comprehensive data analysis on the e-commerce data, focusing on time-dependent quantitative insights. This involved using advanced R packages for data manipulation and visualization, such as dplyr, ggplot2, and lubridate.

Analysis Overview:

The analysis encompassed several key areas:

- Sales Insights: We analyzed sales by location and category in the latest year, identifying top-performing cities and categories.

- Order Fulfillment and Value: We evaluated the average time until order fulfillment by quarter and the average order value, providing insights into operational efficiency.

- Product Performance: Analysis of top-rated products by category offered a view into customer preferences.

Visualizations:

Using the ggplot2 library, we generated visualisations for each analysis domain, such as bar plots and line graphs. The plots illustrated periodic trends, including sales patterns and client expansion, and were specifically built to be effortlessly updated when fresh data is acquired through our data pipeline.

These insights provide vital information for making strategic decisions, aiding in the optimisation of sales strategies, product offerings, and customer engagement initiatives.

Part 4.2: Comprehensive Reporting with Quarto

Objective:

The intent of this section is to detail the insights derived from a sophisticated analysis of our e-commerce data over the years 2020-2023, employing data visualizations from part 4.1, to communicate the findings effectively to stakeholders.

Key Findings:

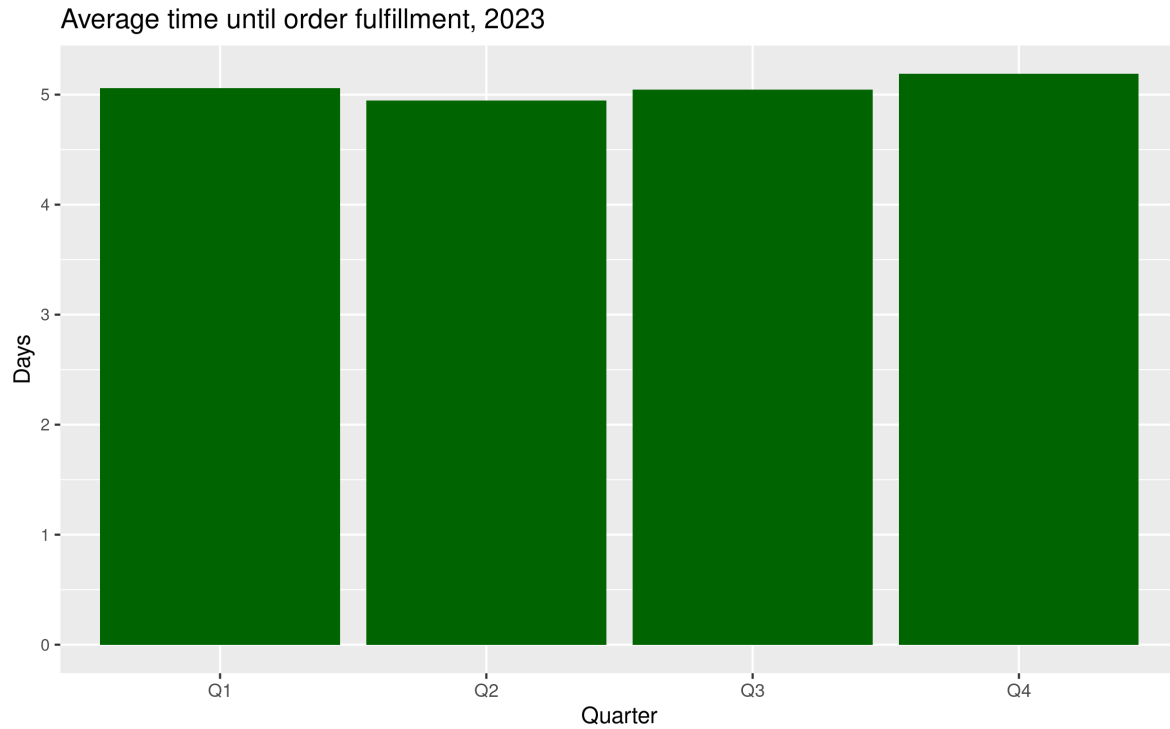Average Time Until Order Fulfillment:

Figure 6: Average Time Until Order Fulfillment

The 'Average time until order fulfillment, 2023' graph demonstrates a consistent delivery performance across all four quarters, indicating an effective and stable logistics system. This consistency is crucial for customer satisfaction and retention.
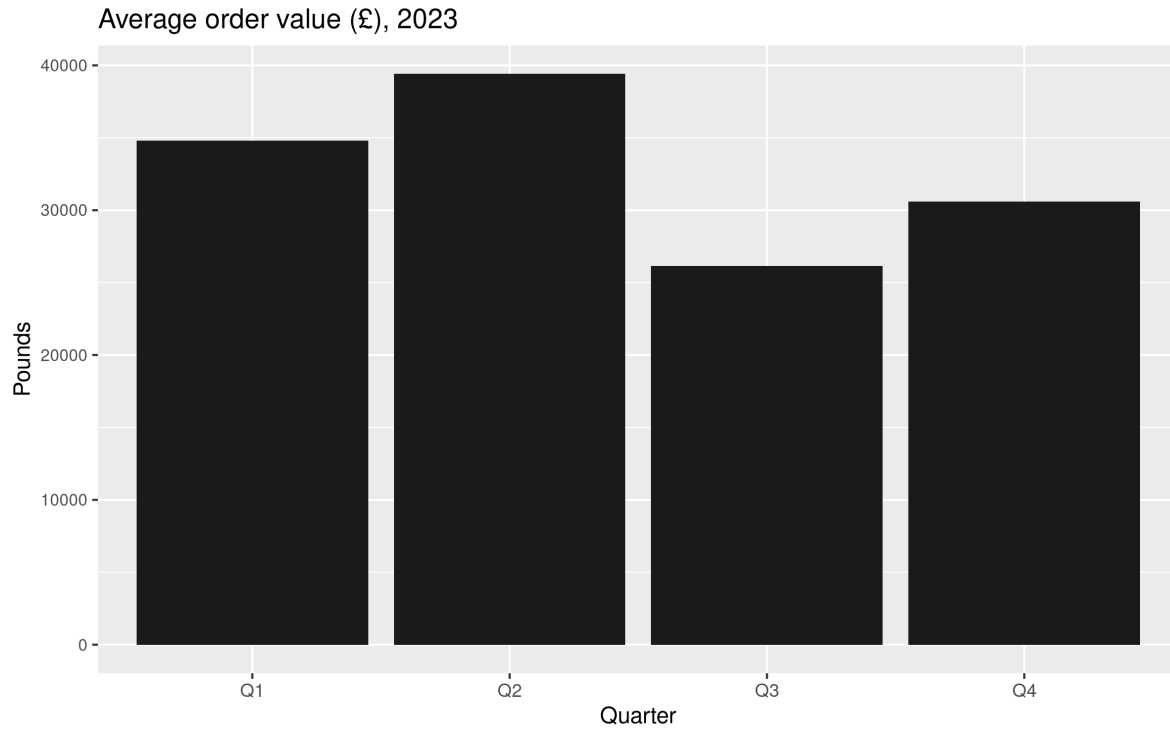
Average Order Value:

Figure 7: Average Order Value

Our analysis of the 'Average order value (£), 2023' highlighted seasonal fluctuations, with Q1 and Q4 experiencing higher values. This pattern suggests that promotions or seasonal demand are influencing shopping behavior, guiding marketing strategies for these periods.

Cumulative Customer Growth:
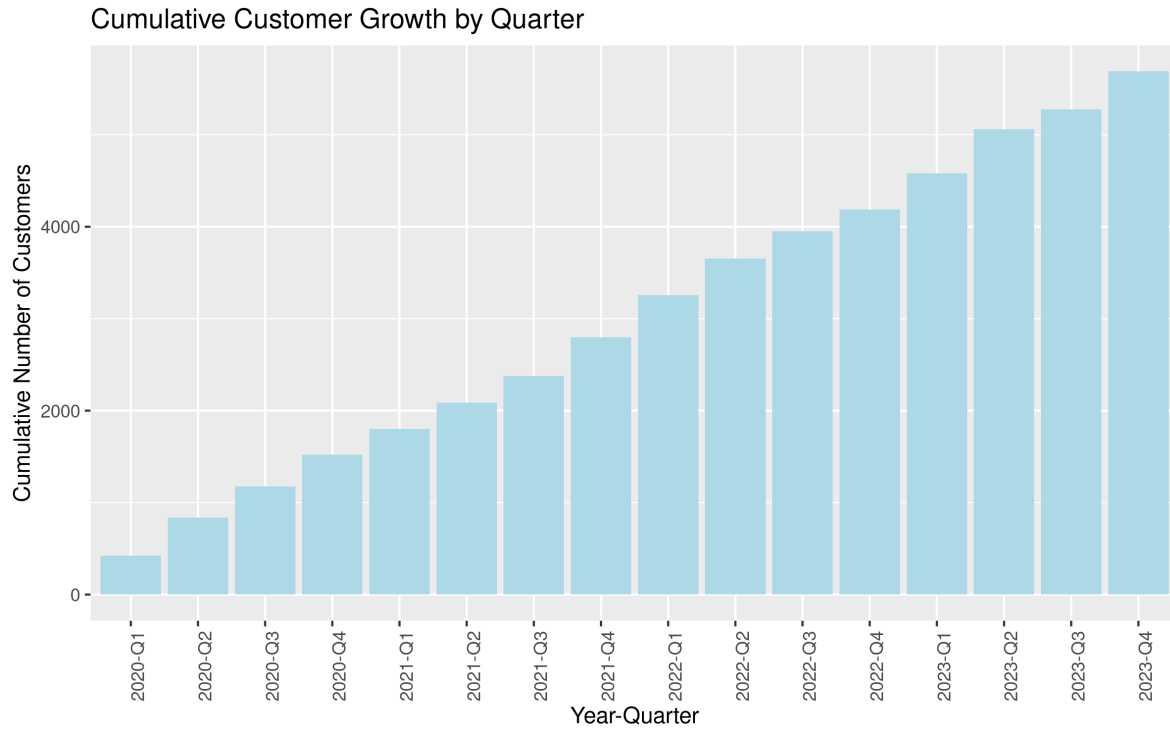
Figure 8: Cumulative Customer Growth

The 'Cumulative Customer Growth by Quarter' plot reveals a steady increase in our customer base. The consistent upward trajectory reflects successful marketing efforts and the growing appeal of our platform.

Age Distribution of Customers:
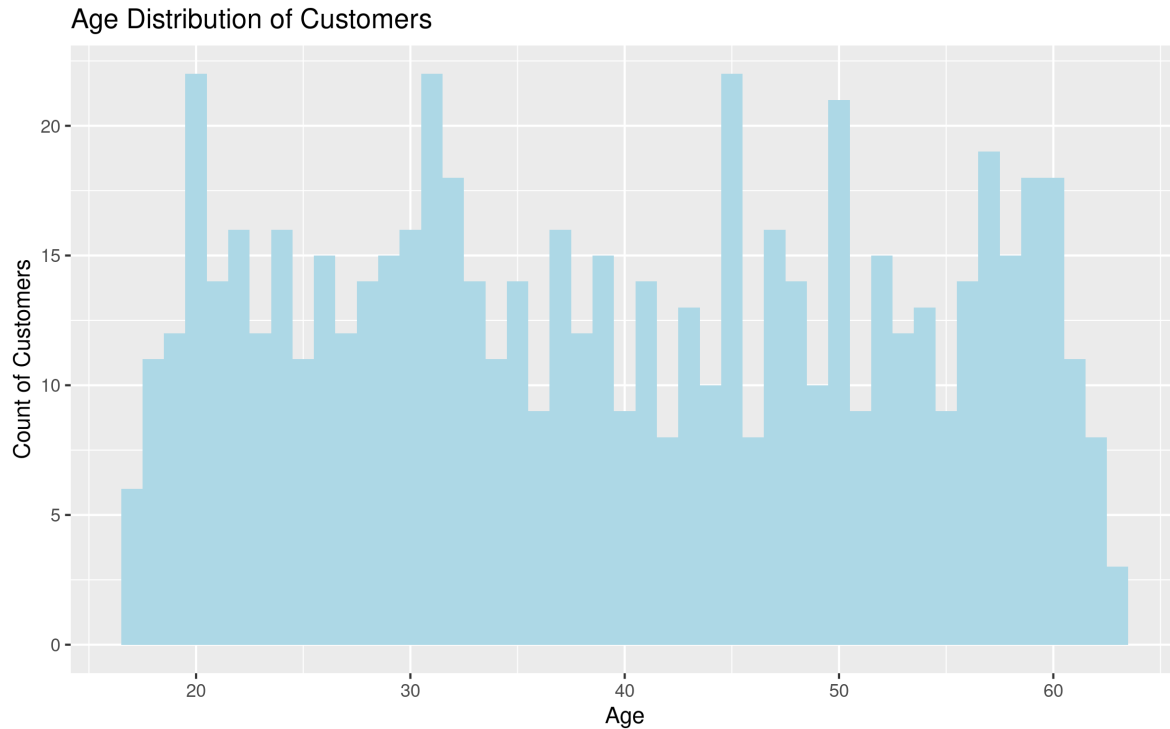
Figure 9: Age Distribution of Customers

The 'Age Distribution of Customers' histogram indicates a broad customer base, with noticeable concentrations around certain age groups. This diversity suggests that our products have a wide appeal, and also highlights key demographic segments for targeted marketing.

Monthly Sales Over Time:
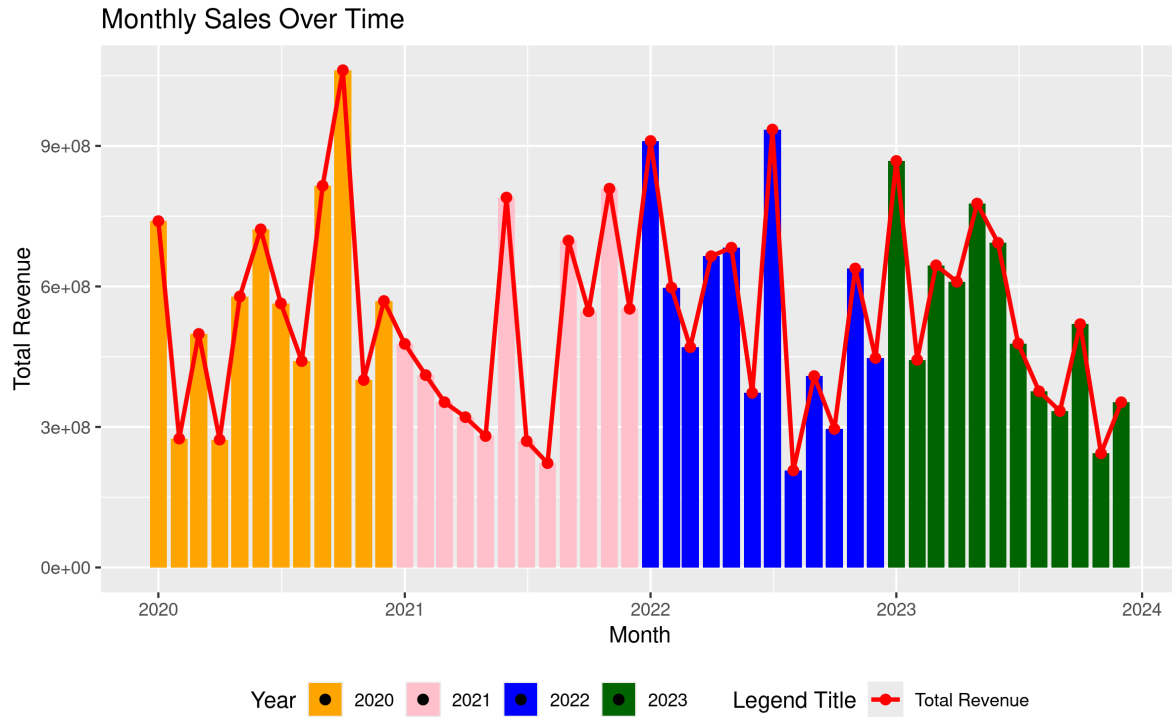
Figure 10: Monthly Sales Over Time

The 'Monthly Sales Over Time' line graph provides a dynamic view of sales performance, with peaks typically in Q4 each year, underscoring the impact of holiday seasons and possibly annual sales events.
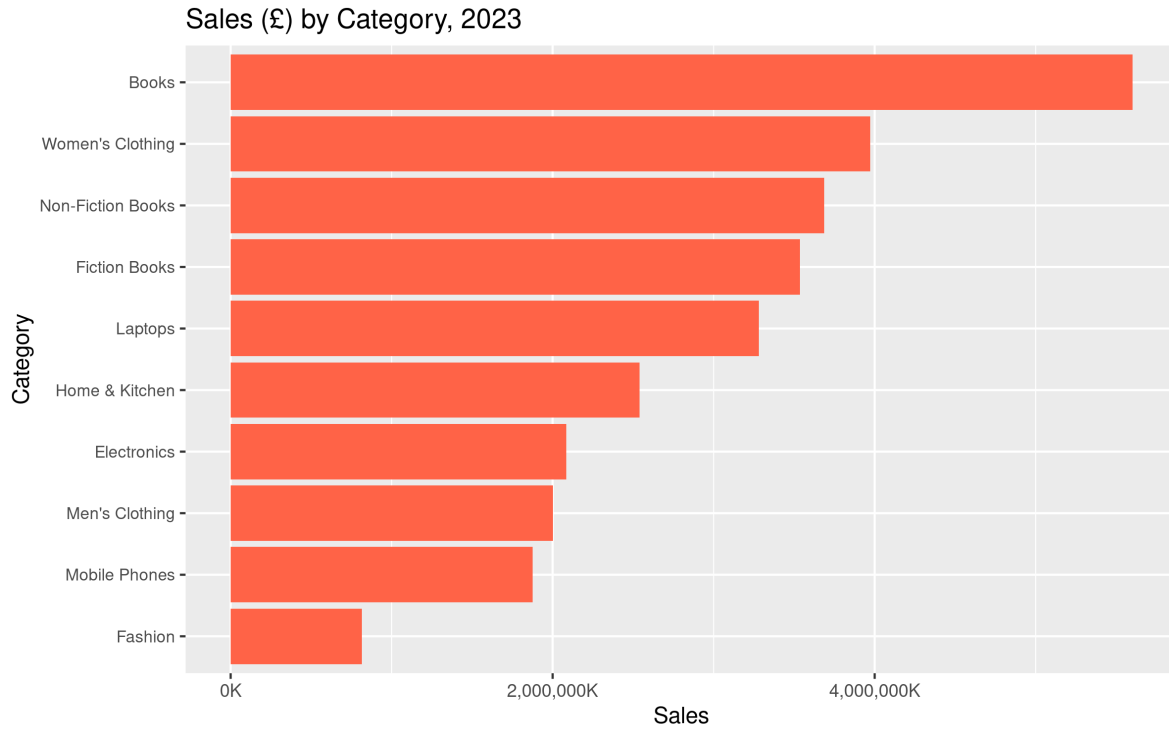
Sales by Category:

Figure 11: Sales by Category

Significant insights come from the 'Sales (£) by Category, 2023' graph, showing that certain categories, like Books and Women's Clothing, are leading sales. This indicates product categories that could be prioritized for stock and promotional activities.
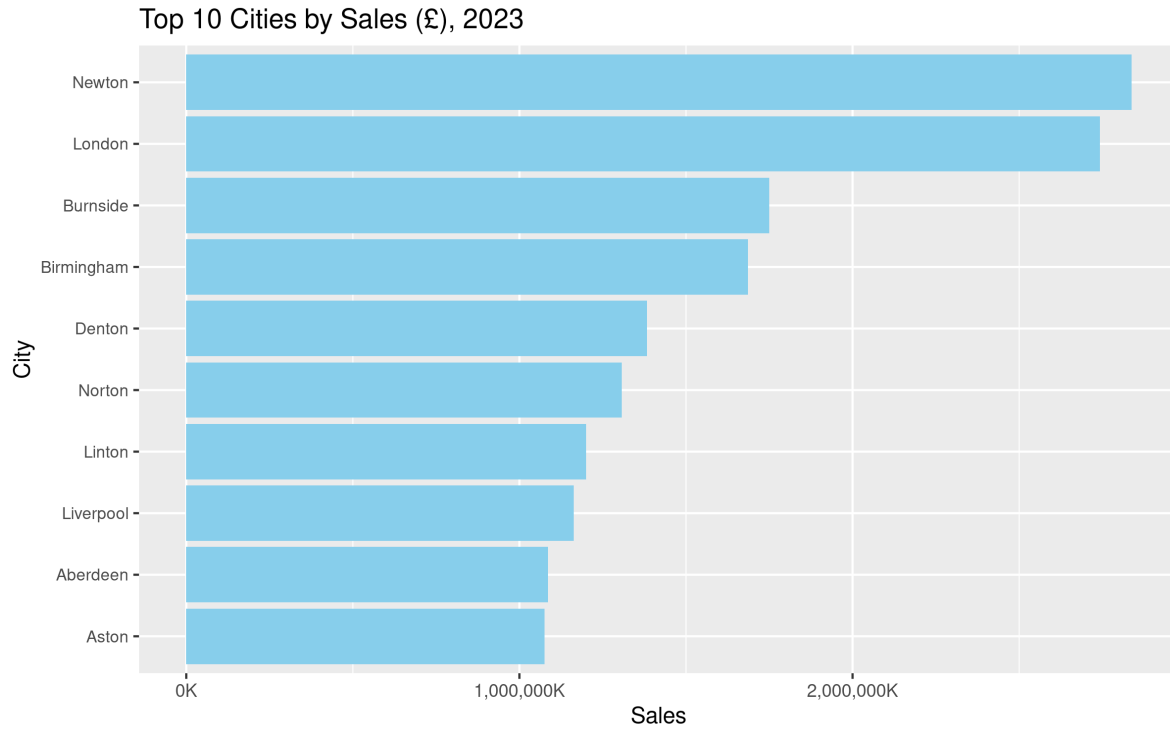
Top 10 Cities by Sales:

Figure 12: Top 10 Cities by Sales

In the 'Top 10 Cities by Sales (£), 2023', we observe that sales are concentrated in certain urban centers, suggesting areas where market penetration could be enhanced or where distribution networks might be optimized.
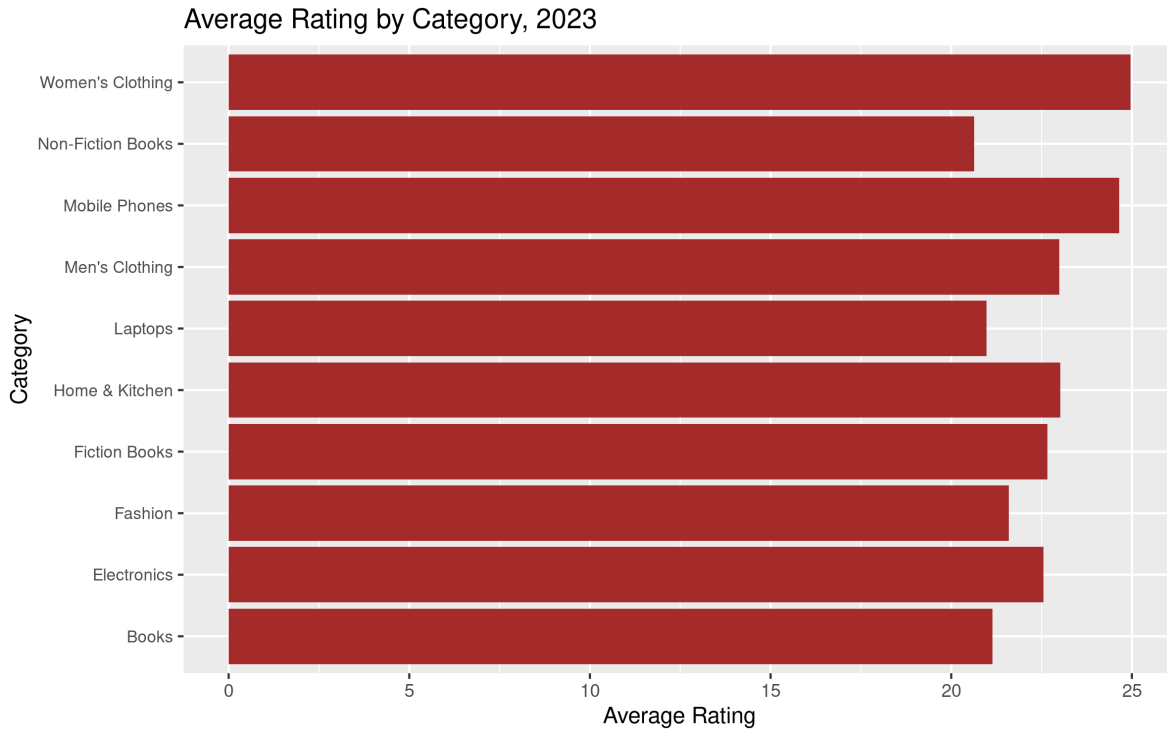
Average Rating by Category:

Figure 13: Average Rating by Category

Finally, the 'Average Rating by Category, 2023' plot illustrates customer satisfaction levels across different product categories, with some categories showing high ratings, which could inform inventory decisions and highlight areas for quality improvement.

Conclusion:

The analysed data presents a comprehensive view of the company's operations throughout the years and focuses on key figures in the year 2023. The insights provide a solid foundation for strategic planning, focusing on customer engagement, sales optimization, and product management for the upcoming year.