Overview

The most crucial aspect of any thriving e-commerce business is the skilled management of data. Recognizing this, our project simulates a real-world e-commerce data environment, where our primary goal is to construct a robust database that can not only handle the high volume of transactions characteristic of busy online platforms but also organize data effectively to streamline operations. By recreating a realistic e-commerce scenario, we aim to ensure that businesses remain adaptable, perceptive, and ahead in the fast-paced world of e-commerce.

Part 1: Database Design and Implementation

1.1 E-R Diagram Design

```
knitr::include_graphics("ER.jpeg")
```

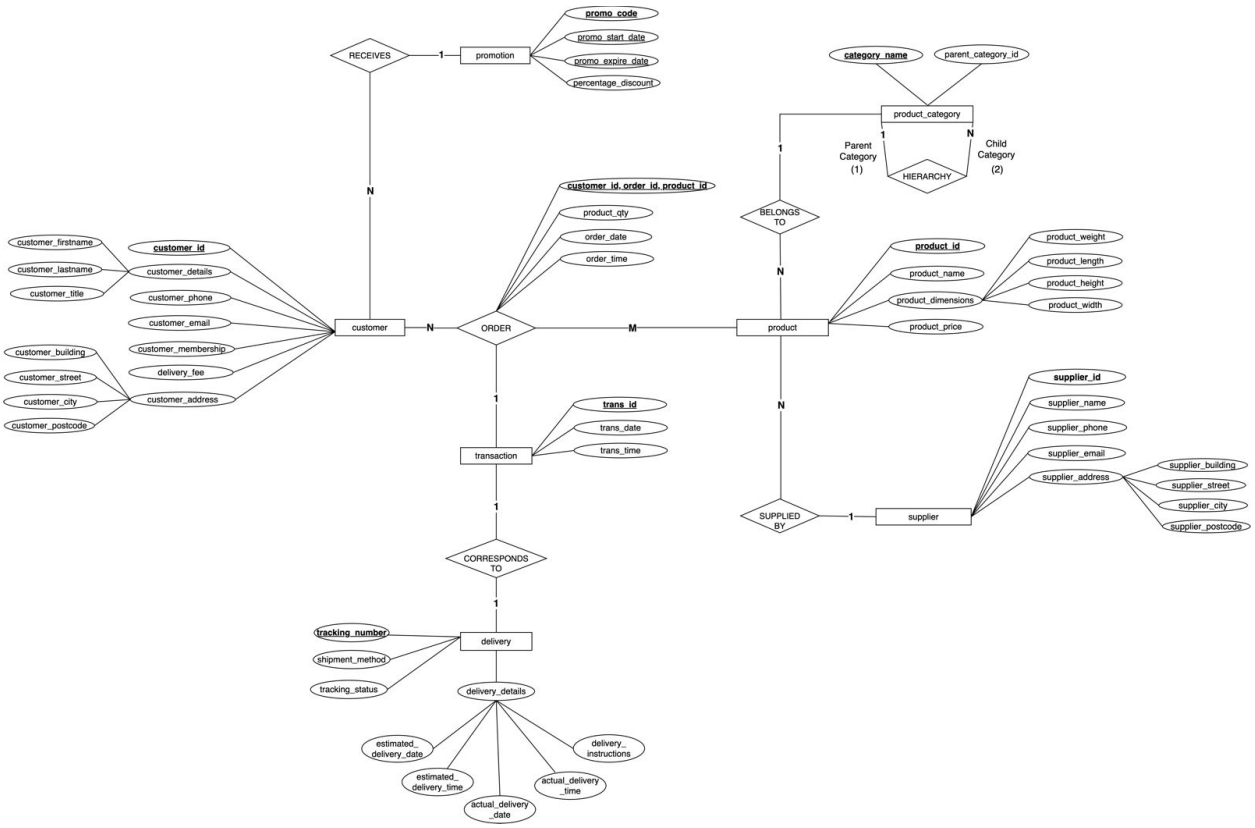

Figure 5: ER Diagram

The architecture of our e-commerce database is structured around seven key entities: customer, product, product category, supplier, promotion, transaction and delivery. These entities are intricately interconnected through a spectrum of relationships, including one-to-one, one-to-many, many-to-many, and self-referencing, in addition to a central ternary relationship.

The "customer" entity, uniquely identified by "customer_id", holds detailed attributes and forms a many-to-many relationship with "product" entity, signifying that customers can purchase multiple products, and products can be purchased by various customers.

The "product" entity has "product_id" as the primary key and is linked to "product category". It has a one-to-many relationship where one category can encompass numerous products. Furthermore, "product" is

similarly linked with “supplier” entity that is uniquely identified by “supplier_id”. It is connected through a one-to-many relationship, under the assumption that one supplier supplies numerous products.

The “product category” entity has a self-referencing relationship with “category_name” as the primary key. It is a hierarchical category structure with one-to-many relationship, as a single parent category can have multiple child categories, but each child category has only one parent category. For instance, “Beauty” is the parent category, and “Body Wash”, “Perfume” and “Hair Styling Product” are the child categories. Each of these child categories would refer to “Beauty” as their parent, creating a self-referencing relationship.

```
knitr::include_graphics("Self-referencing.png")
```

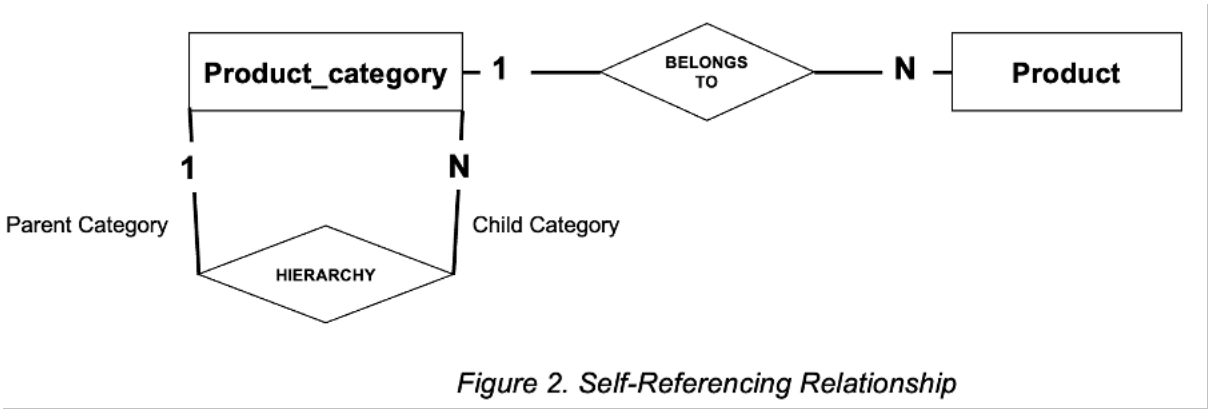

Figure 2. Self-Referencing Relationship

The “promotion” entity with “promo_code” as the primary key engages in a one-to-many relationship with “Customer” entity, under the assumption that one customer can be associated with one promo code, and many promo code can be associated with multiple customers.

Central to the database, “transaction” captures the financial exchanges and is part of a ternary relationship with “customer” and “product” entities. The three are connected with “order” relationship as is shown below:
• A “customer” can have multiple “transaction” (N:1), and within each transaction, multiple “products” can be involved (M:1).

• A “product” can be part of multiple “transaction” through different “customers” (M:N), but within a specific transaction, it is uniquely identified.

```
knitr::include_graphics("MN1 Ternary Relationship.png")
```

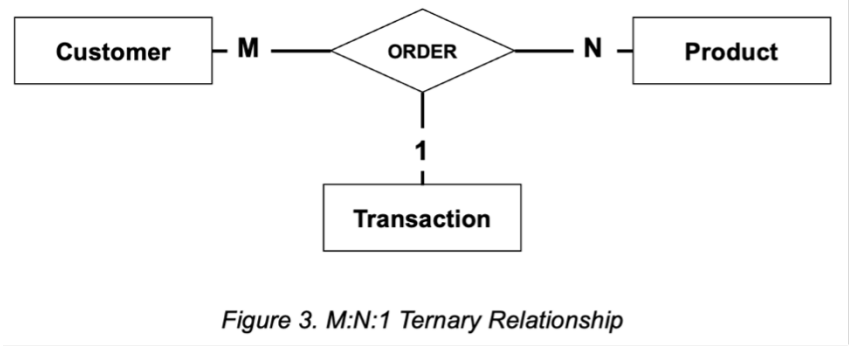

Figure 3. M:N:1 Ternary Relationship

Lastly, the “delivery” entity, with “delivery_id” as the primary key, correlates with “transaction” table in a one-to-one relationship, where each transaction results in a single delivery instance, ensuring that every purchase is accurately fulfilled.

Each relationship sets are illustrated below based on each relationship between two entities:

```
knitr::include_graphics("Relationship Sets.jpeg")
```

Assumptions

To shape our database's logical structure, we have established several assumptions. First, we assume that our E-commerce business commenced in June 2023. We have a singular supplier relationship where a product is supplied by only one supplier. Each customer and supplier have only one address within the United Kingdom, with every customer and supplier allowed to add only one email address and contact number. Furthermore, our operating procedure presumed that the recipient name for each order corresponds to the name provided by the customer during the ordering process. All product prices are listed in GBP, and a free delivery fee is applied to membership holders. On the contrary, there's a delivery fee for customers without membership. All deliveries are handled by our own company, and not by an external company. The delivery time is between 7am – 5pm, and each parcel is delivered to each customer's unique address. All transactions are considered as completed, with no pending or failed statuses, whereas orders placed prior to March 2024 are presumed to be delivered and completed.

Logical Schema

Upon completion of the ER diagram, we moved to the logical schema phase where each attribute corresponds to a column within that table. Consequently, seven tables are established to represent the seven entities: product_category, promotion, supplier, customer, delivery, product and transaction.

When designing the logical schema, a careful review of entity relationships is crucial to determine any additional tables are required. Relationships with cardinalities of 1:1 or N:1 typically do not require extra tables or changes. However, in N:1 relationships, where multiple instances of one entity are associated with a single instance of another, the primary key from the entity on the "1"- weak side transforms into a foreign key in the table on the "N" – strong side. In our ER diagram, we observe that multiple products belong to one product_category (N:1), and multiple products are supplied by one supplier (N:1). Thus, the primary keys "category_name" and "supplier_id" of the weak side are both transferred to the product table as a foreign key. This is the same case for "promo_code" from the promotion table transferred to the customer table as a foreign key. Below is a list of logical schemas for all entities:

```
knitr::include_graphics("Logical Schema.png")
```

1. Customer

customer (customer_id, promo_code, customer_firstname, customer_lastname, customer_title, customer_phone, customer_email, customer_membership, delivery_fee, customer_building, customer_street, customer_city, customer_postcode)

2. Delivery

delivery (tracking_number, trans_id, shipment_method, tracking_status, estimated_delivery_date, estimated_delivery_time, actual_delivery_date, actual_delivery_time, delivery_instructions)

3. Supplier

supplier (supplier_id, supplier_name, supplier_phone, supplier_email, supplier_building, supplier_street, supplier_city, supplier_postcode)

4. Transaction

transaction (trans_id, order_id, trans_date, trans_time)

5. Product category

product_category (category_name, parent_category_id)

6. Product

product (product_id, supplier_id, category_name, product_name, product_weight, product_length, product_height, product_width, product_price)

7. Promotion

promotion (promo_code, promo_start_date, promo_expire_date, percentage_discount)

Finally, due to the many-to-many (M:N) relationship between the customer and product tables, it is necessary to create an additional, separate table named 'order', which will contain additional attributes. The three attributes “order_id”, “customer_id” and “product_id” form a composite key that serves as a primary key for the “order” table, while “customer_id” and “product_id” also serve as a foreign key for the customer table and product table respectively.

8. Order

order (order_id , customer_id, product_id, product_qty, order_date, order_time)

Part 1.2: Database Schema Creation

The database schema creation process starts with establishing a connection and creating SQL tables for each entity, proactively dropping any pre-existing tables to avoid potential issues.

Load Files in an sqlite database

```
#setup the connection
connection <- RSQLite::dbConnect(RSQLite::SQLite(),"hi_import.db")
```

Drop tables

```
DROP TABLE IF EXISTS product
```

```
DROP TABLE IF EXISTS product_category
```

```
DROP TABLE IF EXISTS promotion
```

```
DROP TABLE IF EXISTS supplier
```

```
DROP TABLE IF EXISTS "transaction"
```

```
DROP TABLE IF EXISTS order_datetime
```

```
DROP TABLE IF EXISTS order_products_info
```

```
DROP TABLE IF EXISTS actual_delivery_date
```

```
DROP TABLE IF EXISTS delivery_tracking
```

```
DROP TABLE IF EXISTS estimated_delivery_date
```

```
DROP TABLE IF EXISTS customer_membership
```

```
DROP TABLE IF EXISTS customer_basic_info
```

```
DROP TABLE IF EXISTS customer
```

```
DROP TABLE IF EXISTS delivery
```

```
DROP TABLE IF EXISTS "order"
```

Create SQL tables

product_category

```
-- product_category
CREATE TABLE "product_category" (
  category_name VARCHAR(50) PRIMARY KEY,
  parent_category_id CHAR NULL
);
```

```
SELECT * FROM "product_category";
```

Table 1: 0 records

category_name	parent_category_id

promotion

```
-- promotion
CREATE TABLE "promotion" (
  promo_code INT PRIMARY KEY,
  promo_start_date DATE NULL,
  promo_expire_date DATE NULL,
  percentage_discount NUMERIC NOT NULL
  );
```

```
SELECT * FROM "promotion";
```

Table 2: 0 records

promo_code	promo_start_date	promo_expire_date	percentage_discount

supplier

```
-- supplier
CREATE TABLE supplier (
  supplier_id INT PRIMARY KEY,
  supplier_name CHAR NOT NULL,
  supplier_phone INT NOT NULL,
  supplier_email VARCHAR(50) NOT NULL,
  supplier_building INT NOT NULL,
  supplier_street VARCHAR(50) NOT NULL,
  supplier_city VARCHAR(50) NOT NULL,
  supplier_postcode VARCHAR(50) NOT NULL
) ;
```

```
SELECT * FROM "supplier";
```

Table 3: 0 records

supplier_id	supplier_name	supplier_phone	supplier_email	supplier_building	supplier_street	supplier_city	supplier_postcode

customer

```
-- customer
CREATE TABLE "customer" (
  customer_id INT PRIMARY KEY,
  promo_code INT,
  customer_firstname VARCHAR(50) NOT NULL,
  customer_lastname VARCHAR(50) NOT NULL,
  customer_title VARCHAR(25) NOT NULL,
  customer_phone VARCHAR(50) NOT NULL,
```

```
  customer_email VARCHAR(50) NOT NULL,
  customer_membership TEXT NOT NULL,
  delivery_fee NUMERIC NOT NULL,
  customer_building INT NOT NULL,
  customer_street VARCHAR(50) NOT NULL,
  customer_city VARCHAR(50) NOT NULL,
  customer_postcode VARCHAR(50) NOT NULL,
  FOREIGN KEY (promo_code) REFERENCES "promotion"(promo_code)
) ;
```

```
SELECT * FROM "customer";
```

Table 4: 0 records

customer_id	promo_code	customer_title	customer_first_name	customer_last_name	customer_phone	customer_email	customer_membership	delivery_fee	customer_building	customer_street	customer_city	customer_postcode

delivery

```
-- delivery
CREATE TABLE "delivery" (
  tracking_number INT PRIMARY KEY,
  trans_id INT,
  shipment_method VARCHAR(50) NOT NULL,
  tracking_status VARCHAR(50) NOT NULL,
  estimated_delivery_date DATE NOT NULL,
  estimated_delivery_time TIME NOT NULL,
  actual_delivery_date DATE NULL,
  actual_delivery_time TIME NULL,
  delivery_instructions VARCHAR(125) NOT NULL,
  FOREIGN KEY (trans_id) REFERENCES "transaction"(trans_id)
);
```

```
SELECT * FROM "delivery";
```

Table 5: 0 records

tracking_number	trans_id	shipment_method	tracking_status	estimated_delivery_date	estimated_delivery_time	actual_delivery_date	actual_delivery_time	delivery_instructions

product

```
-- product
CREATE TABLE "product" (
  product_id INT PRIMARY KEY,
  supplier_id INT,
  category_name VARCHAR(50),
  product_name VARCHAR(25) NOT NULL,
  product_weight NUMERIC NOT NULL,
  product_length NUMERIC NOT NULL,
  product_height NUMERIC NOT NULL,
  product_width NUMERIC NOT NULL,
  product_price NUMERIC NOT NULL,
```

```
  FOREIGN KEY (supplier_id) REFERENCES "supplier"(supplier_id),
  FOREIGN KEY (category_name) REFERENCES "product_category"(category_name)
) ;
```

```
SELECT * FROM "product";
```

Table 6: 0 records

product_id	supplier_id	category_name	product_name	product_weight	product_length	product_height	product_width	product_price

order

```
-- order
CREATE TABLE "order" (
  order_id INT,
  customer_id INT,
  product_id INT,
  product_qty INT NOT NULL,
  order_date DATE NOT NULL,
  order_time TIME NOT NULL,
  PRIMARY KEY (order_id, customer_id, product_id),
  FOREIGN KEY (customer_id) REFERENCES "customer"(customer_id),
  FOREIGN KEY (product_id) REFERENCES "product"(product_id)
) ;
```

```
SELECT * FROM "order";
```

Table 7: 0 records

order_id	customer_id	product_id	product_qty	order_date	order_time

transaction

```
-- transaction
CREATE TABLE "transaction" (
  trans_id INT PRIMARY KEY,
  order_id INT,
  trans_date DATE NOT NULL,
  trans_time TIME NOT NULL,
  FOREIGN KEY (order_id) REFERENCES "order"(order_id)
);
```

```
SELECT * FROM "transaction";
```

Table 8: 0 records

trans_id	order_id	trans_date	trans_time

Normalization to 3NF

For customer

1. customer_basic_info

```
-- customer_basic_info
CREATE TABLE "customer_basic_info" (
  customer_id INT PRIMARY KEY,
  promo_code INT,
  customer_firstname VARCHAR(50) NOT NULL,
  customer_lastname VARCHAR(50) NOT NULL,
  customer_title VARCHAR(25) NOT NULL,
  customer_phone VARCHAR(50) NOT NULL,
  customer_email VARCHAR(50) NOT NULL,
  customer_building INT NOT NULL,
  customer_street VARCHAR(50) NOT NULL,
  customer_city VARCHAR(50) NOT NULL,
  customer_postcode VARCHAR(50) NOT NULL,
  FOREIGN KEY (promo_code) REFERENCES "promotion"(promo_code)
) ;
```

```
SELECT * FROM customer_basic_info
```

Table 9: 0 records

customer_id	promo_code	customer_firstname	customer_lastname	customer_title	customer_phone	customer_email	customer_building	customer_street	customer_city	customer_postcode

2. customer_membership

```
-- customer_membership
CREATE TABLE "customer_membership" (
  customer_id INT,
  customer_membership TEXT,
  delivery_fee NUMERIC NOT NULL,
  PRIMARY KEY (customer_id, customer_membership),
  FOREIGN KEY (customer_id) REFERENCES "customer_basic_info"(customer_id)
);
```

```
SELECT * FROM customer_membership
```

Table 10: 0 records

customer_id	customer_membership	delivery_fee

For order

1. order_products_info

```
-- order_products_info
CREATE TABLE "order_products_info" (
  order_id INT,
  customer_id INT,
  product_id INT,
  product_qty INT NOT NULL,
```

```
  PRIMARY KEY (order_id, customer_id, product_id),
  FOREIGN KEY (customer_id) REFERENCES "customer_basic_info"(customer_id),
  FOREIGN KEY (product_id) REFERENCES "product"(product_id)
) ;
```

```
SELECT * FROM order_products_info
```

Table 11: 0 records

order_id	customer_id	product_id	product_qty

2. order_datetime

```
-- order_datetime
CREATE TABLE "order_datetime" (
  order_id INT,
  customer_id INT,
  order_date DATE NOT NULL,
  order_time TIME NOT NULL,
  PRIMARY KEY (order_id, customer_id),
  FOREIGN KEY (customer_id) REFERENCES "customer_basic_info"(customer_id)
) ;
```

```
SELECT * FROM order_datetime
```

Table 12: 0 records

order_id	customer_id	order_date	order_time

For Delivery

1. delivery_tracking

```
-- delivery_tracking
CREATE TABLE "delivery_tracking" (
  tracking_number INT PRIMARY KEY,
  trans_id INT,
  delivery_instructions VARCHAR(125) NOT NULL,
  FOREIGN KEY (trans_id) REFERENCES "transaction"(trans_id)
);
```

```
SELECT * FROM delivery_tracking
```

Table 13: 0 records

tracking_number	trans_id	delivery_instructions

2. estimated_delivery_date

```
-- estimated_delivery_date
CREATE TABLE "estimated_delivery_date" (
  tracking_number INT,
  shipment_method VARCHAR(50),
  estimated_delivery_date DATE NOT NULL,
```

```
  estimated_delivery_time TIME NOT NULL,
  PRIMARY KEY (tracking_number, shipment_method),
  FOREIGN KEY (tracking_number) REFERENCES "delivery_tracking"(tracking_number)
);
```

```
SELECT * FROM estimated_delivery_date
```

Table 14: 0 records

tracking_number	shipment_method	estimated_delivery_date	estimated_delivery_time

3. actual_delivery_date

```
-- actual_delivery_date
CREATE TABLE "actual_delivery_date" (
  tracking_number INT,
  tracking_status VARCHAR(50),
  actual_delivery_date DATE NULL,
  actual_delivery_time TIME NULL,
  PRIMARY KEY (tracking_number, tracking_status),
  FOREIGN KEY (tracking_number) REFERENCES "delivery_tracking"(tracking_number)
);
```

```
SELECT * FROM actual_delivery_date
```

Table 15: 0 records

tracking_number	tracking_status	actual_delivery_date	actual_delivery_time

```
knitr::include_graphics("ER after Normalisation.jpeg")
```

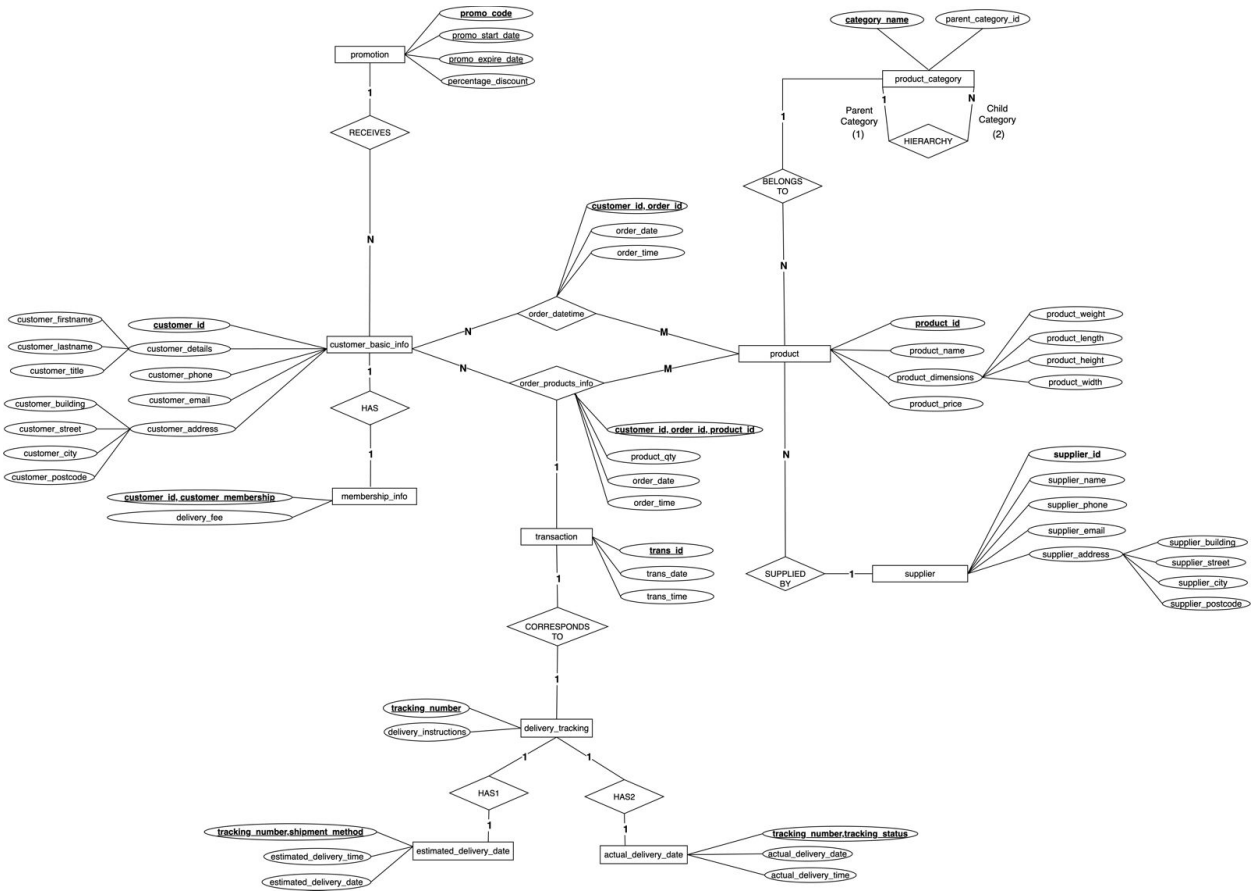

Figure 5: ER Diagram - After Normalisation

Part 2: Data Generation and Management

Part 2.1: Synthetic Data Generation

We have initially generated synthetic data using Mockaroo, a random test data generator, and ChatGPT, a generative AI, which we downloaded the data into an excel sheet by updating the field names, data types, and options. However, there are some fields related to date and time which need to be generated with conditions. We use R code to generate these data to ensure the chronological order of date and time for promotion table, order table, transaction table and delivery table.

Part 2.2: Data Import and Quality Assurance

Import csv files into SQL table

For loop to read csv and then import into sql

```
# Read datasets
#order
order_datetime_dataset <- read.csv("Dataset/hi_order_datetime_dataset.csv")
order_products_info_dataset <- read.csv("Dataset/hi_order_products_info_dataset.csv")

#delivery
```

```r
actual_delivery_date_dataset <- read.csv("Dataset/hi_actual_delivery_date_dataset.csv")
delivery_tracking_dataset <- read.csv("Dataset/hi_delivery_tracking_dataset.csv")
estimated_delivery_date_dataset <- read.csv("Dataset/hi_estimated_delivery_date_dataset.csv")

#customer
customer_basic_info_dataset <- read.csv("Dataset/hi_customer_basic_info_dataset.csv")
customer_membership_dataset <- read.csv("Dataset/hi_customer_membership_dataset.csv")
product_dataset <- read.csv("Dataset/hi_product_dataset.csv")
product_category_dataset <- read.csv("Dataset/hi_product_category_dataset.csv")
promotion_dataset <- read.csv("Dataset/hi_promotion_dataset.csv")
supplier_dataset <- read.csv("Dataset/hi_supplier_dataset.csv")
transaction_dataset <- read.csv("Dataset/hi_transaction_dataset.csv")
```

```r
dbWriteTable(connection, "product", product_dataset, append = TRUE, row.names = FALSE)
dbWriteTable(connection, "product_category", product_category_dataset, append = TRUE, row.names = FALSE)
dbWriteTable(connection, "promotion", promotion_dataset, append = TRUE, row.names = FALSE)
dbWriteTable(connection, "supplier", supplier_dataset, append = TRUE, row.names = FALSE)
dbWriteTable(connection, "transaction", transaction_dataset, append = TRUE, row.names = FALSE)

#order
dbWriteTable(connection, "order_datetime", order_datetime_dataset, append = TRUE, row.names = FALSE)
dbWriteTable(connection, "order_products_info", order_products_info_dataset, append = TRUE, row.names =

#delivery
dbWriteTable(connection, "actual_delivery_date", actual_delivery_date_dataset, append = TRUE, row.names
dbWriteTable(connection, "delivery_tracking", delivery_tracking_dataset, append = TRUE, row.names = FALS
dbWriteTable(connection, "estimated_delivery_date", estimated_delivery_date_dataset, append = TRUE, row

#customer
dbWriteTable(connection, "customer_membership", customer_membership_dataset, append = TRUE, row.names =
dbWriteTable(connection, "customer_basic_info", customer_basic_info_dataset, append = TRUE, row.names =
```

Check the tables using select

```sql
SELECT * FROM "product" LIMIT 5
```

Table 16: 5 records

product_id	supplier_id	category_name	product_name	product_weight	product_length	product_height	product_width	product_price
34-100-2931	RSH-48812	Biography	Unveiled: The Life of a Visionary	4322	67	35	30	11.62
60-215-8627	HNW-87364	Educational Toys	Learn & Play Alphabet Blocks	614	24	22	98	30.83
10-395-8862	QIS-31117	Fresh Produce	Organic Harvest Bundle	1942	65	10	30	26.30
84-465-9981	TJZ-16253	History	Epochs in Time: A Historical Analysis	3825	74	73	47	43.25

product_id	supplier_id	category_name	product_name	product_weight	product_length	product_height	product_width	product_price
51-355-5771	OTM-80847	Mystery	Whispers in the Shadows: Mystery Novel	471	67	31	87	14.78

```
SELECT * FROM "product_category" LIMIT 5
```

Table 17: 5 records

category_name	parent_category_id
Beauty	
Books	
Clothing	
Electronics	
Grocery	

```
SELECT * FROM "promotion" LIMIT 5
```

Table 18: 5 records

promo_code	promo_start_date	promo_expire_date	percentage_discount
BU86M505PYD	07/07/2023	27/12/2023	50
BG70Z584RFB	09/11/2023	03/03/2024	25
JI24S173EUJ	20/03/2024	07/07/2024	45
FU87P552XKO	24/11/2023	06/04/2024	45
SD70E981QNT	16/02/2024	04/08/2024	40

```
SELECT * FROM "supplier" LIMIT 5
```

Table 19: 5 records

supplier_id	supplier_name	supplier_phone	supplier_email	supplier_building	supplier_street	supplier_city	supplier_postcode
XTE-60952	Hyatt and Sons	+44 336 825 7695	mkilpatrick0@nyu.edu	796	Garden Road	London	KY2Y 6JZ
WLS-09227	Huels-Krajcik	+44 515 420 8651	cbritt1@unesco.org	881	Meadow Road	Birmingham	B12 7TB
RCO-72629	Morissette LLC	+44 213 964 1394	ctrussler2@hao123.com	66	Garden Road	Birmingham	DE9W 6WF
KCV-52154	Koepp, Bechtelar and Weimann	+44 404 383 6574	egoddman3@mtv.com	921	River Road	London	W1 9SG
UTZ-90791	Stamm-Schmidt	+44 312 442 5804	lharome4@oaic.gov.au	254	Lake Road	Bristol	L43 3FX

```
SELECT * FROM "transaction" LIMIT 5
```

Table 20: 5 records

trans_id	order_id	trans_date	trans_time
EHD-784624	AAD-4091	10/11/2023	07:56:40
SIZ-926836	AAK-0526	09/02/2024	15:23:52
GEP-276863	ADJ-6838	02/08/2023	12:30:16
YRS-371629	ADV-4775	26/10/2023	14:57:31
VMS-478374	ADX-1928	28/02/2024	00:15:39

Order:

```
SELECT * FROM "order_datetime" LIMIT 5
```

Table 21: 5 records

order_id	customer_id	order_date	order_time
AAD-4091	YTS-92438	10/11/2023	05:43:29
AAK-0526	BWU-36083	09/02/2024	14:20:43
AAK-6361	CLZ-73501	28/03/2024	20:29:45
ADJ-5614	KCQ-71974	19/11/2023	23:34:47
ADJ-6838	GSB-19226	02/08/2023	11:28:04

```
SELECT * FROM "order_products_info" LIMIT 5
```

Table 22: 5 records

order_id	customer_id	product_id	product_qty
AAD-4091	YTS-92438	42-811-3974	9
AAD-4091	YTS-92438	72-217-8555	10
AAK-0526	BWU-36083	85-279-1314	5
AAK-0526	BWU-36083	43-612-9451	19
AAK-0526	BWU-36083	42-665-9904	16

Delivery

```
SELECT * FROM "actual_delivery_date" LIMIT 5
```

Table 23: 5 records

tracking_number	tracking_status	actual_delivery_date	actual_delivery_time
581-6200	Delivered	24/06/2023	12:45:44
004-1482	Delivered	22/11/2023	08:18:48
064-9023	In process	NA	NA
657-4120	Delivered	01/02/2024	07:43:05
983-4613	Delivered	12/08/2023	11:01:41

```
SELECT * FROM "delivery_tracking" LIMIT 5
```

Table 24: 5 records

tracking_number	trans_id	delivery_instructions
581-6200	AAA-067232	ring bell
004-1482	AAC-152328	delivery box
064-9023	AAD-850184	ring bell
657-4120	AAR-680860	leave infront of door
983-4613	AAT-296188	ring bell

```
SELECT * FROM "estimated_delivery_date" LIMIT 5
```

Table 25: 5 records

tracking_number	shipment_method	estimated_delivery_date	estimated_delivery_time
581-6200	express	23/06/2023	11:20:33
004-1482	next day	21/11/2023	14:41:18
064-9023	express	06/03/2024	07:27:46
657-4120	standard	01/02/2024	15:10:18
983-4613	next day	11/08/2023	16:02:04

Customer:

```
SELECT * FROM "customer_membership" LIMIT 5
```

Table 26: 5 records

customer_id	customer_membership	delivery_fee
NAT-21446	membership	0.00
MQV-12400	not membership	4.99
MLY-44705	not membership	5.99
RFE-59474	membership	0.00
WBO-40739	membership	0.00

```
SELECT * FROM "customer_basic_info" LIMIT 5
```

Table 27: 5 records

customer_id	password_hash	customer_title	customer_first_name	customer_last_name	customer_phone	customer_email	customer_building_number	customer_street	customer_city	customer_postcode
NAT-21446	VS04A9350NO	Dr	Berriball	Dr	+44 482 422 6609	mberriball0@abc.net.au	103	Willow Street	Birmingham	B1D 6RT
MQV-12400	OQ50R1A0HWT	Mr	Lujan	Mr	+44 941 356 9889	alujan1@qq.com	436	Spruce Street	Birmingham	G1A 8DD
MLY-44705	DU63L7Z7XKV	Honorable	Llewellen	Honorable	+44 398 412 8484	mllewellen2@hud.gov	861	Willow Street	Bristol	G4H 0NH
RFE-59474	IW96D8I2NGT	Honorable	Philson	Honorable	+44 114 708 3717	iphilson3@github.com	271	Maple Street	Bristol	M04 5UF

customer_id	prod_id	customer_first_name	customer_last_name	customer_title	customer_phone	customer_email	building_no	street	city	postcode
WBO-40739	IX31K07ZeHZ	Strangeway	Mrs	+44 782 865 8414	lstrangeway4@utexas.edu		107	Ash Street	Birmingham	M8 3LL

Calculated Field

Transaction amount

```
SELECT o.order_id, prm.percentage_discount, m.delivery_fee, SUM(p.product_price*o.product_qty) AS order_
FROM "order_products_info" o, "product" p, "promotion" prm, "customer_basic_info" c, "customer_membershi
WHERE o.customer_id = m.customer_id AND o.product_id = p.product_id AND c.promo_code = prm.promo_code AN
GROUP BY
    o.order_id;
```

Table 28: Displaying records 1 - 10

order_id	percentage_discount	delivery_fee	order_price	trans_amount
AAD-4091	25	0.00	822.15	616.61
AAK-0526	30	0.00	767.93	537.55
AAK-6361	25	6.99	1058.78	801.08
ADJ-5614	45	3.99	193.16	110.23
ADJ-6838	35	2.99	681.37	445.88
ADV-4775	45	5.99	1113.84	618.60
ADV-7947	20	5.99	116.87	99.49
ADX-1928	45	0.00	892.56	490.91
ADX-2356	40	0.00	350.07	210.04
AFD-0715	15	0.00	827.32	703.22

Store natively it in R

```
# Customer
customer_membership <- dbReadTable(connection, "customer_membership")
customer_basic_info <- dbReadTable(connection, "customer_basic_info")

#Delivery
actual_delivery_date <- dbReadTable(connection, "actual_delivery_date")
delivery_tracking <- dbReadTable(connection, "delivery_tracking")
estimated_delivery_date <- dbReadTable(connection, "estimated_delivery_date")

#Order
order_datetime <- dbReadTable(connection, "order_datetime")
order_products_info <- dbReadTable(connection, "order_products_info")

product <- dbReadTable(connection, "product")
product_category <- dbReadTable(connection, "product_category")
promotion <- dbReadTable(connection, "promotion")
supplier <- dbReadTable(connection, "supplier")
transaction <- dbReadTable(connection, "transaction")
```

List tables

```
RSQLite::dbListTables(connection)
```

```
##  [1] "actual_delivery_date"    "customer"
##  [3] "customer_basic_info"     "customer_membership"
##  [5] "customer_memebership"    "delivery"
##  [7] "delivery_tracking"       "estimated_delivery_date"
##  [9] "order"                   "order_datetime"
## [11] "order_products_info"     "product"
## [13] "product_category"        "promotion"
## [15] "supplier"                "transaction"
```

Disconnect SQL

```
RSQLite::dbDisconnect(connection)
```

Data Validation

List all files

```
all_files <- list.files("Dataset/")
all_files
```

```
##  [1] "hi_actual_delivery_date_dataset.csv"
##  [2] "hi_customer_basic_info_dataset.csv"
##  [3] "hi_customer_membership_dataset.csv"
##  [4] "hi_delivery_tracking_dataset.csv"
##  [5] "hi_estimated_delivery_date_dataset.csv"
##  [6] "hi_order_datetime_dataset.csv"
##  [7] "hi_order_products_info_dataset.csv"
##  [8] "hi_product_category_dataset.csv"
##  [9] "hi_product_dataset.csv"
## [10] "hi_promotion_dataset.csv"
## [11] "hi_supplier_dataset.csv"
## [12] "hi_transaction_dataset.csv"
```

```
prefix <- "hi_"
suffix <- "_dataset.csv"
all_files <- gsub("hi_","",all_files)
all_files <- gsub("_dataset.csv","",all_files)
all_files
```

```
##  [1] "actual_delivery_date"    "customer_basic_info"
##  [3] "customer_membership"     "delivery_tracking"
##  [5] "estimated_delivery_date" "order_datetime"
##  [7] "order_products_info"     "product_category"
##  [9] "product"                 "promotion"
## [11] "supplier"                "transaction"
```

In our pursuit of ensuring data integrity, we implemented a comprehensive quality assurance process which included:

Check number of rows and columns

```
all_files <- list.files("Dataset/")

for (variable in all_files) {
  this_filepath <- paste0("Dataset/",variable)
  this_file_contents <- readr::read_csv(this_filepath)

  number_of_rows <- nrow(this_file_contents)
  number_of_columns <- ncol(this_file_contents)

  print(paste0("The file: ",variable,
               " has: ",
               format(number_of_rows,big.mark = ","),
               " rows and ",
               number_of_columns," columns"))
}
```

```
## [1] "The file: hi_actual_delivery_date_dataset.csv has: 1,000 rows and 4 columns"
## [1] "The file: hi_customer_basic_info_dataset.csv has: 1,000 rows and 11 columns"
## [1] "The file: hi_customer_membership_dataset.csv has: 1,000 rows and 3 columns"
## [1] "The file: hi_delivery_tracking_dataset.csv has: 1,000 rows and 3 columns"
## [1] "The file: hi_estimated_delivery_date_dataset.csv has: 1,000 rows and 4 columns"
## [1] "The file: hi_order_datetime_dataset.csv has: 1,681 rows and 4 columns"
## [1] "The file: hi_order_products_info_dataset.csv has: 3,401 rows and 4 columns"
## [1] "The file: hi_product_category_dataset.csv has: 88 rows and 2 columns"
## [1] "The file: hi_product_dataset.csv has: 1,000 rows and 9 columns"
## [1] "The file: hi_promotion_dataset.csv has: 1,000 rows and 4 columns"
## [1] "The file: hi_supplier_dataset.csv has: 1,000 rows and 8 columns"
## [1] "The file: hi_transaction_dataset.csv has: 1,000 rows and 4 columns"
```

Check the data structure

```
all_files <- list.files("Dataset/")

for (variable in all_files) {
  this_filepath <- paste0("Dataset/",variable)
  this_file_contents <- readr::read_csv(this_filepath)
  data_structure<-str(this_file_contents)

  print(paste0(data_structure,
               "The file: ",variable,
               " has above data structure"))
}
```

```
## spc_tbl_ [1,000 x 4] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ tracking_number     : chr [1:1000] "581-6200" "004-1482" "064-9023" "657-4120" ...
##  $ tracking_status     : chr [1:1000] "Delivered" "Delivered" "In process" "Delivered" ...
##  $ actual_delivery_date: chr [1:1000] "24/06/2023" "22/11/2023" NA "01/02/2024" ...
##  $ actual_delivery_time: 'hms' num [1:1000] 12:45:44 08:18:48 NA 07:43:05 ...
##   ..- attr(*, "units")= chr "secs"
##  - attr(*, "spec")=
##   .. cols(
##   ..   tracking_number = col_character(),
```

```
##   ..   tracking_status = col_character(),
##   ..   actual_delivery_date = col_character(),
##   ..   actual_delivery_time = col_time(format = "")
##   .. )
##  - attr(*, "problems")=<externalptr>
## [1] "The file: hi_actual_delivery_date_dataset.csv has above data structure"
## spc_tbl_ [1,000 x 11] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ customer_id       : chr [1:1000] "NAT-21446" "MQV-12400" "MLY-44705" "RFE-59474" ...
##  $ customer_firstname: chr [1:1000] "Mike" "Antone" "Moria" "Ichabod" ...
##  $ customer_lastname : chr [1:1000] "Berriball" "Lujan" "Llewellen" "Philson" ...
##  $ customer_title    : chr [1:1000] "Dr" "Mr" "Honorable" "Honorable" ...
##  $ customer_phone    : chr [1:1000] "+44 482 422 6609" "+44 941 356 9889" "+44 398 412 8484" "+44 114
##  $ customer_email    : chr [1:1000] "mberriball0@abc.net.au" "alujan1@qq.com" "mllewellen2@hud.gov" "
##  $ customer_building : num [1:1000] 103 436 861 271 107 72 701 21 615 122 ...
##  $ customer_street   : chr [1:1000] "Willow Street" "Spruce Street" "Willow Street" "Maple Street" .
##  $ customer_city     : chr [1:1000] "Birmingham" "Birmingham" "Bristol" "Bristol" ...
##  $ customer_postcode : chr [1:1000] "B1D 6RT" "G1A 8DD" "G4H 0NH" "M04 5UF" ...
##  $ promo_code        : chr [1:1000] "VS04A935ONO" "OQ50R17OHWT" "DU63L727XKV" "IW96D852NGT" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   customer_id = col_character(),
##   ..   customer_firstname = col_character(),
##   ..   customer_lastname = col_character(),
##   ..   customer_title = col_character(),
##   ..   customer_phone = col_character(),
##   ..   customer_email = col_character(),
##   ..   customer_building = col_double(),
##   ..   customer_street = col_character(),
##   ..   customer_city = col_character(),
##   ..   customer_postcode = col_character(),
##   ..   promo_code = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
## [1] "The file: hi_customer_basic_info_dataset.csv has above data structure"
## spc_tbl_ [1,000 x 3] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ customer_id        : chr [1:1000] "NAT-21446" "MQV-12400" "MLY-44705" "RFE-59474" ...
##  $ customer_membership: chr [1:1000] "membership" "not membership" "not membership" "membership" ...
##  $ delivery_fee       : num [1:1000] 0 4.99 5.99 0 0 2.99 5.99 0 6.99 5.99 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   customer_id = col_character(),
##   ..   customer_membership = col_character(),
##   ..   delivery_fee = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
## [1] "The file: hi_customer_membership_dataset.csv has above data structure"
## spc_tbl_ [1,000 x 3] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ tracking_number      : chr [1:1000] "581-6200" "004-1482" "064-9023" "657-4120" ...
##  $ delivery_instructions: chr [1:1000] "ring bell" "delivery box" "ring bell" "leave infront of door"
##  $ trans_id             : chr [1:1000] "AAA-067232" "AAC-152328" "AAD-850184" "AAR-680860" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   tracking_number = col_character(),
##   ..   delivery_instructions = col_character(),
```

```
##   ..   trans_id = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
## [1] "The file: hi_delivery_tracking_dataset.csv has above data structure"
## spc_tbl_ [1,000 x 4] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ tracking_number       : chr [1:1000] "581-6200" "004-1482" "064-9023" "657-4120" ...
##  $ shipment_method       : chr [1:1000] "express" "next day" "express" "standard" ...
##  $ estimated_delivery_date: chr [1:1000] "23/06/2023" "21/11/2023" "06/03/2024" "01/02/2024" ...
##  $ estimated_delivery_time: 'hms' num [1:1000] 11:20:33 14:41:18 07:27:46 15:10:18 ...
##   ..- attr(*, "units")= chr "secs"
##  - attr(*, "spec")=
##   .. cols(
##   ..   tracking_number = col_character(),
##   ..   shipment_method = col_character(),
##   ..   estimated_delivery_date = col_character(),
##   ..   estimated_delivery_time = col_time(format = "")
##   .. )
##  - attr(*, "problems")=<externalptr>
## [1] "The file: hi_estimated_delivery_date_dataset.csv has above data structure"
## spc_tbl_ [1,681 x 4] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ customer_id: chr [1:1681] "YTS-92438" "BWU-36083" "CLZ-73501" "KCQ-71974" ...
##  $ order_id   : chr [1:1681] "AAD-4091" "AAK-0526" "AAK-6361" "ADJ-5614" ...
##  $ order_date : chr [1:1681] "10/11/2023" "09/02/2024" "28/03/2024" "19/11/2023" ...
##  $ order_time : 'hms' num [1:1681] 05:43:29 14:20:43 20:29:45 23:34:47 ...
##   ..- attr(*, "units")= chr "secs"
##  - attr(*, "spec")=
##   .. cols(
##   ..   customer_id = col_character(),
##   ..   order_id = col_character(),
##   ..   order_date = col_character(),
##   ..   order_time = col_time(format = "")
##   .. )
##  - attr(*, "problems")=<externalptr>
## [1] "The file: hi_order_datetime_dataset.csv has above data structure"
## spc_tbl_ [3,401 x 4] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ customer_id: chr [1:3401] "YTS-92438" "YTS-92438" "BWU-36083" "BWU-36083" ...
##  $ order_id   : chr [1:3401] "AAD-4091" "AAD-4091" "AAK-0526" "AAK-0526" ...
##  $ product_id : chr [1:3401] "42-811-3974" "72-217-8555" "85-279-1314" "43-612-9451" ...
##  $ product_qty: num [1:3401] 9 10 5 19 16 20 14 4 13 10 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   customer_id = col_character(),
##   ..   order_id = col_character(),
##   ..   product_id = col_character(),
##   ..   product_qty = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
## [1] "The file: hi_order_products_info_dataset.csv has above data structure"
## spc_tbl_ [88 x 2] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ category_name     : chr [1:88] "Beauty" "Books" "Clothing" "Electronics" ...
##  $ parent_category_id: chr [1:88] NA NA NA NA ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   category_name = col_character(),
```

```
##   ..   parent_category_id = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
## [1] "The file: hi_product_category_dataset.csv has above data structure"
## spc_tbl_ [1,000 x 9] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ product_id    : chr [1:1000] "34-100-2931" "60-215-8627" "10-395-8862" "84-465-9981" ...
##  $ product_name  : chr [1:1000] "Unveiled: The Life of a Visionary" "Learn & Play Alphabet Blocks" "
##  $ product_weight: num [1:1000] 4322 614 1942 3825 471 ...
##  $ product_length: num [1:1000] 67 24 65 74 67 83 75 63 68 67 ...
##  $ product_height: num [1:1000] 35 22 10 73 31 87 100 47 33 92 ...
##  $ product_width : num [1:1000] 30 98 30 47 87 94 100 56 61 58 ...
##  $ product_price : num [1:1000] 11.6 30.8 26.3 43.2 14.8 ...
##  $ supplier_id   : chr [1:1000] "RSH-48812" "HNW-87364" "QIS-31117" "TJZ-16253" ...
##  $ category_name : chr [1:1000] "Biography" "Educational Toys" "Fresh Produce" "History" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   product_id = col_character(),
##   ..   product_name = col_character(),
##   ..   product_weight = col_double(),
##   ..   product_length = col_double(),
##   ..   product_height = col_double(),
##   ..   product_width = col_double(),
##   ..   product_price = col_double(),
##   ..   supplier_id = col_character(),
##   ..   category_name = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
## [1] "The file: hi_product_dataset.csv has above data structure"
## spc_tbl_ [1,000 x 4] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ promo_code         : chr [1:1000] "BU86M505PYD" "BG70Z584RFB" "JI24S173EUJ" "FU87P552XKO" ...
##  $ promo_start_date   : chr [1:1000] "07/07/2023" "09/11/2023" "20/03/2024" "24/11/2023" ...
##  $ promo_expire_date  : chr [1:1000] "27/12/2023" "03/03/2024" "07/07/2024" "06/04/2024" ...
##  $ percentage_discount: num [1:1000] 50 25 45 45 40 30 35 10 40 45 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   promo_code = col_character(),
##   ..   promo_start_date = col_character(),
##   ..   promo_expire_date = col_character(),
##   ..   percentage_discount = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
## [1] "The file: hi_promotion_dataset.csv has above data structure"
## spc_tbl_ [1,000 x 8] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ supplier_id      : chr [1:1000] "XTE-60952" "WLS-09227" "RCO-72629" "KCV-52154" ...
##  $ supplier_name    : chr [1:1000] "Hyatt and Sons" "Huels-Krajcik" "Morissette LLC" "Koepp, Bechtela
##  $ supplier_phone   : chr [1:1000] "+44 336 825 7695" "+44 515 420 8651" "+44 213 964 1394" "+44 404
##  $ supplier_email   : chr [1:1000] "mkilpatrick0@nyu.edu" "cbritt1@unesco.org" "ctrussler2@hao123.com
##  $ supplier_building: num [1:1000] 796 881 66 921 254 968 44 554 774 679 ...
##  $ supplier_street  : chr [1:1000] "Garden Road" "Meadow Road" "Garden Road" "River Road" ...
##  $ supplier_city    : chr [1:1000] "London" "Birmingham" "Birmingham" "London" ...
##  $ supplier_postcode: chr [1:1000] "KY2Y 6JZ" "B12 7TB" "DE9W 6WF" "W1 9SG" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   supplier_id = col_character(),
```

```
##   ..   supplier_name = col_character(),
##   ..   supplier_phone = col_character(),
##   ..   supplier_email = col_character(),
##   ..   supplier_building = col_double(),
##   ..   supplier_street = col_character(),
##   ..   supplier_city = col_character(),
##   ..   supplier_postcode = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
## [1] "The file: hi_supplier_dataset.csv has above data structure"
## spc_tbl_ [1,000 x 4] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ trans_id  : chr [1:1000] "EHD-784624" "SIZ-926836" "GEP-276863" "YRS-371629" ...
##  $ order_id  : chr [1:1000] "AAD-4091" "AAK-0526" "ADJ-6838" "ADV-4775" ...
##  $ trans_date: chr [1:1000] "10/11/2023" "09/02/2024" "02/08/2023" "26/10/2023" ...
##  $ trans_time: 'hms' num [1:1000] 07:56:40 15:23:52 12:30:16 14:57:31 ...
##   ..- attr(*, "units")= chr "secs"
##  - attr(*, "spec")=
##   .. cols(
##   ..   trans_id = col_character(),
##   ..   order_id = col_character(),
##   ..   trans_date = col_character(),
##   ..   trans_time = col_time(format = "")
##   .. )
##  - attr(*, "problems")=<externalptr>
## [1] "The file: hi_transaction_dataset.csv has above data structure"
```

Check for NULL values

```r
all_files <- list.files("Dataset/")

for (variable in all_files) {
  this_filepath <- paste0("Dataset/",variable)
  this_file_contents <- readr::read_csv(this_filepath)
  null<-sum(is.na(this_file_contents))

  print(paste0("The file: ",variable,
               " has a total of ", null,
               " NULL values"))
}
```

```
## [1] "The file: hi_actual_delivery_date_dataset.csv has a total of 172 NULL values"
## [1] "The file: hi_customer_basic_info_dataset.csv has a total of 0 NULL values"
## [1] "The file: hi_customer_membership_dataset.csv has a total of 0 NULL values"
## [1] "The file: hi_delivery_tracking_dataset.csv has a total of 0 NULL values"
## [1] "The file: hi_estimated_delivery_date_dataset.csv has a total of 0 NULL values"
## [1] "The file: hi_order_datetime_dataset.csv has a total of 0 NULL values"
## [1] "The file: hi_order_products_info_dataset.csv has a total of 0 NULL values"
## [1] "The file: hi_product_category_dataset.csv has a total of 8 NULL values"
## [1] "The file: hi_product_dataset.csv has a total of 0 NULL values"
## [1] "The file: hi_promotion_dataset.csv has a total of 0 NULL values"
## [1] "The file: hi_supplier_dataset.csv has a total of 0 NULL values"
## [1] "The file: hi_transaction_dataset.csv has a total of 0 NULL values"
```

Check that each primary key is unique in each table except for order

```
all_files <- list.files("Dataset/")

for (variable in all_files) {
  this_filepath <- paste0("Dataset/",variable)
  this_file_contents <- readr::read_csv(this_filepath)
  hi <- nrow(unique(this_file_contents[,1]))== nrow(this_file_contents)

  print(paste0("The file: ",variable,
               " has unique primary key ",
               hi," columns"))
}
```

```
## [1] "The file: hi_actual_delivery_date_dataset.csv has unique primary key TRUE columns"
## [1] "The file: hi_customer_basic_info_dataset.csv has unique primary key TRUE columns"
## [1] "The file: hi_customer_membership_dataset.csv has unique primary key TRUE columns"
## [1] "The file: hi_delivery_tracking_dataset.csv has unique primary key TRUE columns"
## [1] "The file: hi_estimated_delivery_date_dataset.csv has unique primary key TRUE columns"
## [1] "The file: hi_order_datetime_dataset.csv has unique primary key FALSE columns"
## [1] "The file: hi_order_products_info_dataset.csv has unique primary key FALSE columns"
## [1] "The file: hi_product_category_dataset.csv has unique primary key TRUE columns"
## [1] "The file: hi_product_dataset.csv has unique primary key TRUE columns"
## [1] "The file: hi_promotion_dataset.csv has unique primary key TRUE columns"
## [1] "The file: hi_supplier_dataset.csv has unique primary key TRUE columns"
## [1] "The file: hi_transaction_dataset.csv has unique primary key TRUE columns"
```

For order dataset

We have a composite primary key orderdate_dataset composed of 2 attribute and a composite primary key orderproductsinfo_dataset composed of 3 attribute, we'll check these one separately.

```
orderdate_dataset <- read.csv("Dataset/hi_order_datetime_dataset.csv")
orderproductsinfo_dataset <- read.csv("Dataset/hi_order_products_info_dataset.csv")
nrow(unique(orderdate_dataset[,1:2])) == nrow(orderdate_dataset)
```

```
## [1] TRUE
```

```
nrow(unique(orderproductsinfo_dataset[,1:3])) == nrow(orderproductsinfo_dataset)
```

```
## [1] TRUE
```

```
#sum(nrow(unique(orders[,1:2])))
#length((unique(orders$cutomer_id)))
#length((unique(orders$order_id)))
#The file: hi_order_dataset.csv has unique primary composite key TRUE columns
```

In Customer dataset, there are e-mail and phone number columns, which required specific format checks. For e-mail, special characters "@" and ending domain must be validated. For phone number, IT must be ensured to have 10 numeric characters.

Validate phone number

```
# Supplier
length(grepl("\\+44\\s\\d{3}\\s\\d{3}\\s\\d{4}", supplier$supplier_phone)) == nrow(supplier)
```

```
## [1] TRUE
```

```r
# Consumer
length(grepl("\\+44\\s\\d{3}\\s\\d{3}\\s\\d{4}", customer_basic_info$customer_phone)) == nrow(customer_
```

```
## [1] TRUE
```

Validate emails

```r
# Supplier
length(grepl("@", supplier$supplier_email)) == nrow(supplier)
```

```
## [1] TRUE
```

```r
# Consumer
length(grepl("@", customer_basic_info$customer_email)) == nrow(customer_basic_info)
```

```
## [1] TRUE
```

Part 3: Data Pipeline Generation

Part 3.1: GitHub Repository and Workflow Setup

Our primary objective was to utilize GitHub repositories for efficient version control of our project. Initially, we transformed our R Markdown file into an R script, encompassing all the steps outlined in section 2.2. We proceeded to load the requisite files into a SQLite database, creating tables and importing datasets from the "Dataset" file. Despite solely utilizing the R script, we ensured all pertinent files were maintained within the repository. This was achieved by leveraging the repository's URL and executing commits and pushes whenever modifications were made. Subsequently, we established a new repository on GitHub and extended invitations to collaborators, granting them appropriate access levels.

Part 3.2: GitHub Actions for Continuous Integration

For the implementation of GitHub Actions for Continuous Integration, we aimed to streamline our project's development workflow. This involved adopting continuous integration practices to automate tasks such as data validation, database updates, and data analysis. For instance, whenever changes occurred in the repository, such as importing new data, the updates were seamlessly integrated into the database, and relevant outputs were generated only if significant changes were detected. Scheduled workflows were set to run at regular intervals, such as every three hours, ensuring consistent and efficient execution of these tasks.

Within workflow, we configured the R environment and cached any necessary R packages that were not already installed. Essential packages such as "ggplot" and "dplyr" were installed for data visualization purposes. The workflow then executed the R script, incorporating any changes made during the process. Finally, all modifications were committed to the local repository and subsequently pushed to the main branch using a unique access token for authentication. This comprehensive approach to CI enabled seamless integration of updates and ensured the project's integrity and efficiency.