Data Management Group Assignment

DM Group 5

Contents

Database Design and Implementation

1.1 E-R Diagram Design:

1. E-R Diagram:
   We created an Entity-Relationship diagram for an e-commerce store by first identifying a total of 7 key entities namely suppliers, customers, products, product categories, order details, transactions and promotions. Relationships were created to better define the connection between two entities which included provide, belong to, order and make. Attributes were created to detail the information held within every entity. Order relationship was given multiple attributes to better explain the nature of the relationship. After analysis of the ER diagram, to efficiently define the data held within shipping address attribute, it was given further attributes and used as a composite attribute for order details.

   Assumptions: Multiple Suppliers provide Multiple Products, Multiple products belong to a single product category, Multiple customers can order multiple products, multiple customers can make multiple transactions, multiple products can be contained in a single order detail and multiple promotions can be applied to multiple products.

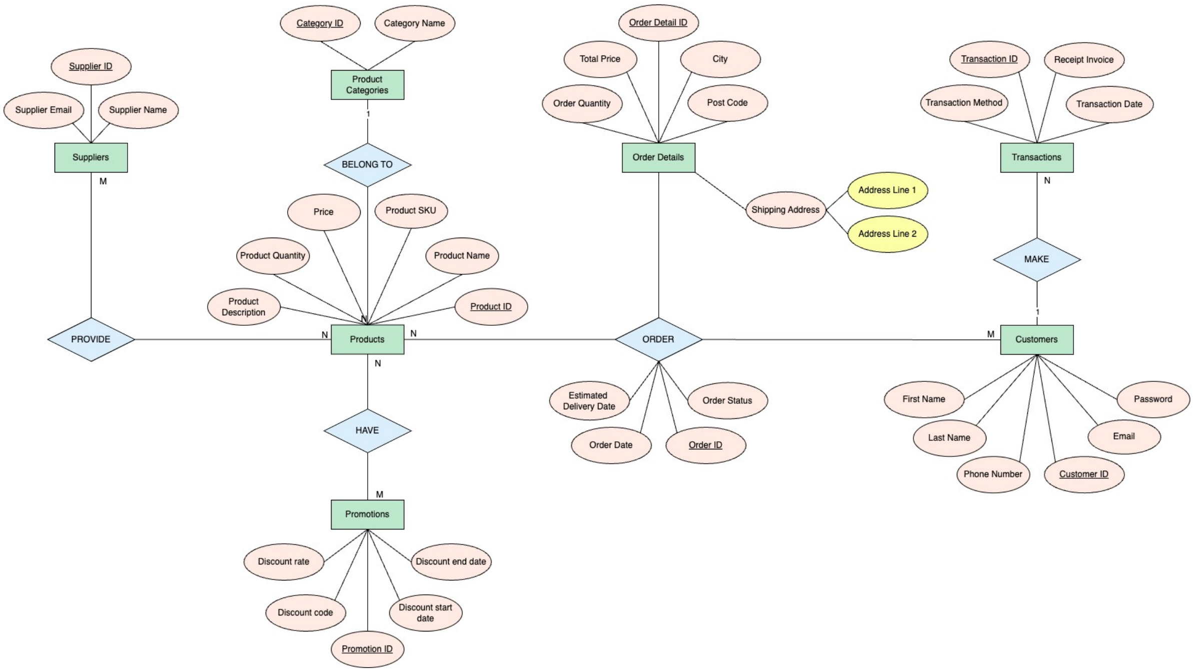


Figure 1: Entity-Relationship Diagram

2. Relationship Sets:

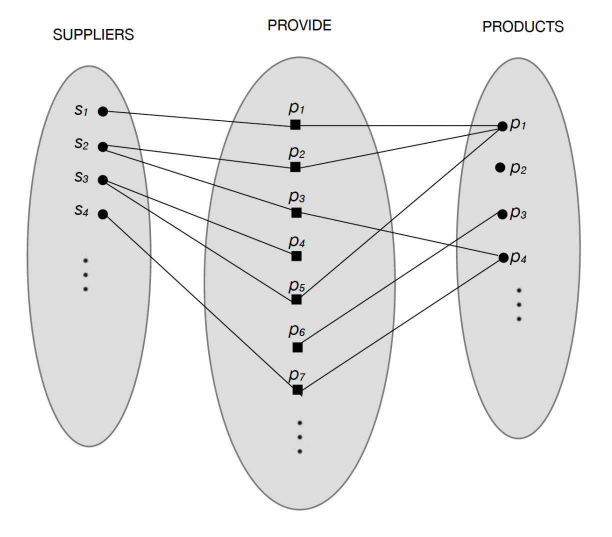


Figure 2: Suppliers PROVIDE products. Many-to-many relationship (M:N).

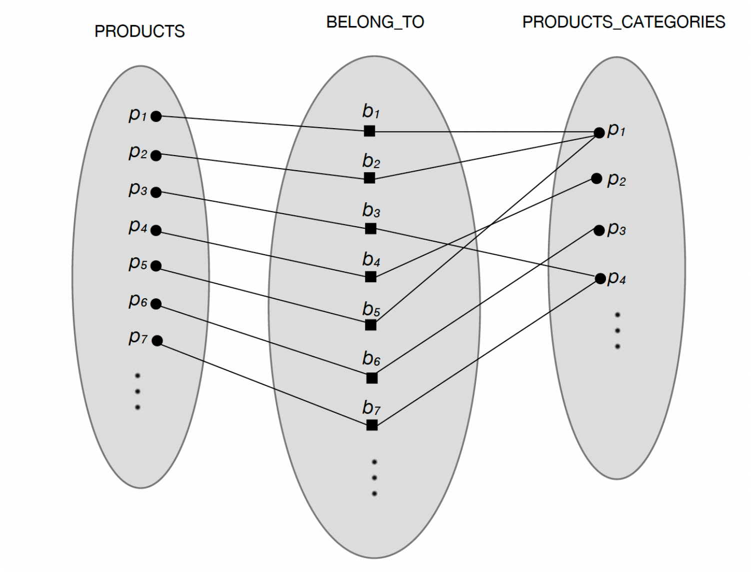


Figure 3: Products BELONG TO Product Categories. Many-to-one relationship (N:1).

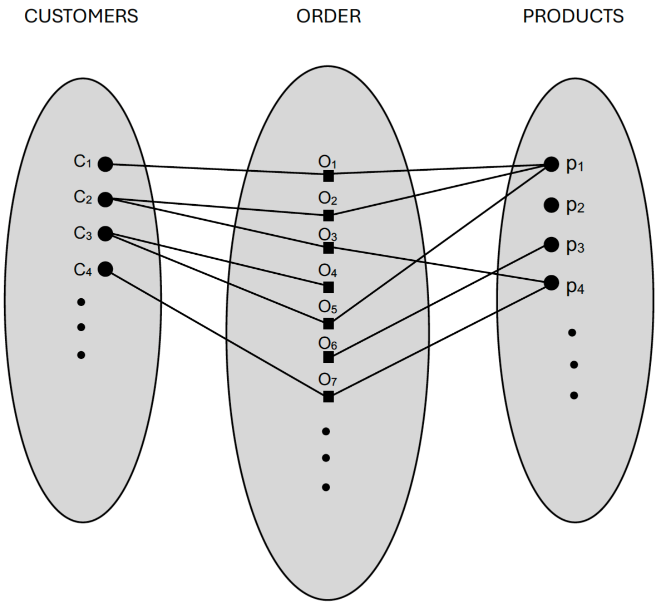


Figure 4: Customers ORDER products. Many-to-many relationship (M:N).

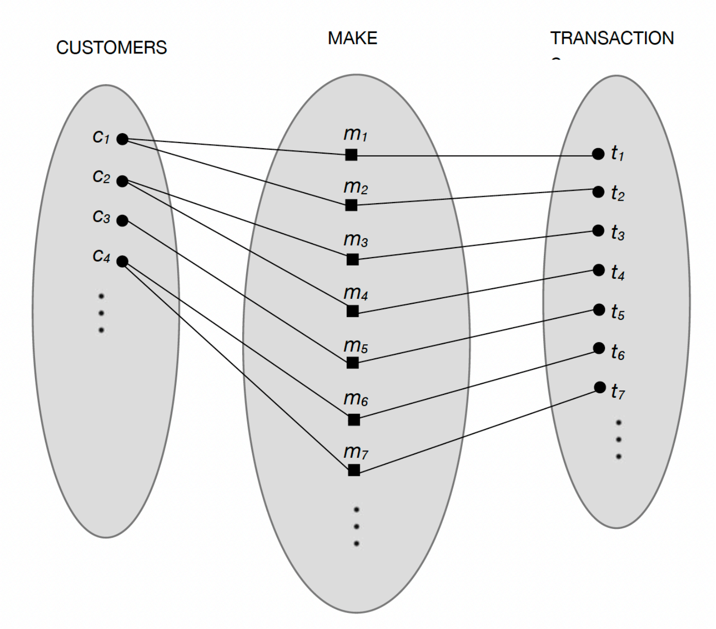


Figure 5: Customers MAKE Transaction. Many-to-many relationship (M:N).

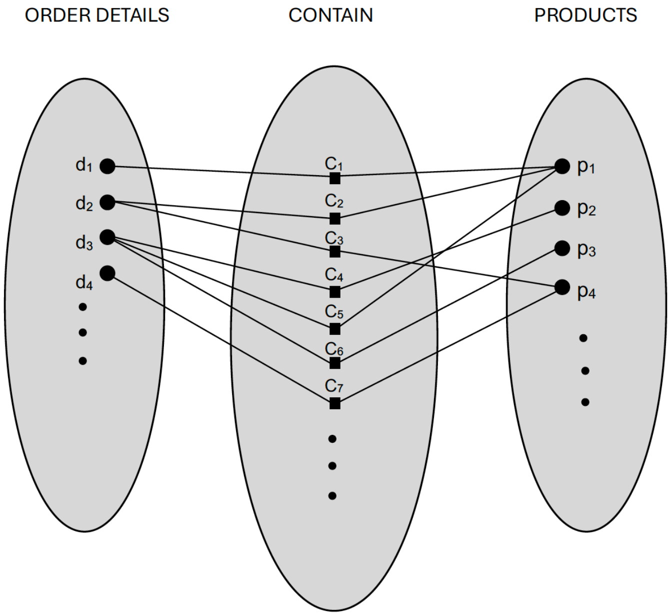


Figure 6: Order Details CONTAIN Products. One-to-many relationship (1:N).

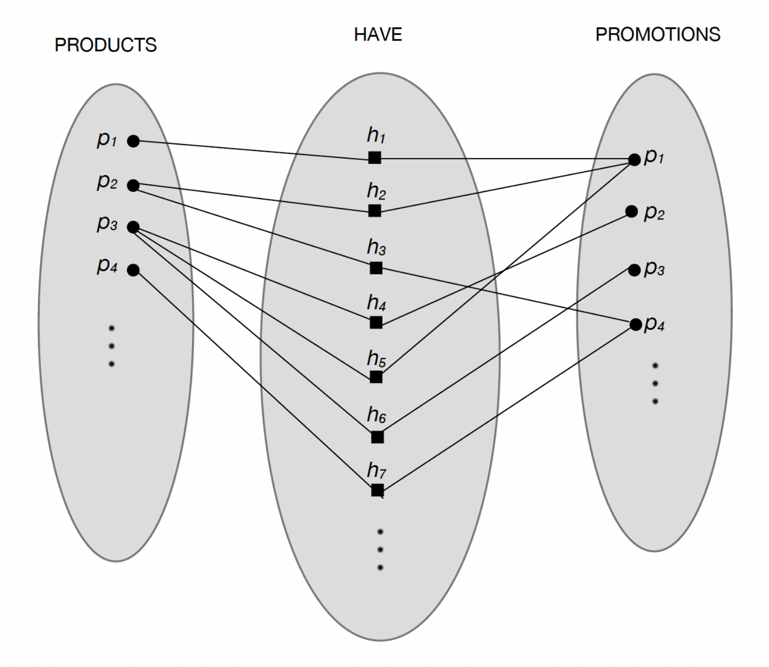


Figure 7: Products HAVE Promotions. Many-to-many relationship (N:M).

1.2 SQL Database Schema Creation:

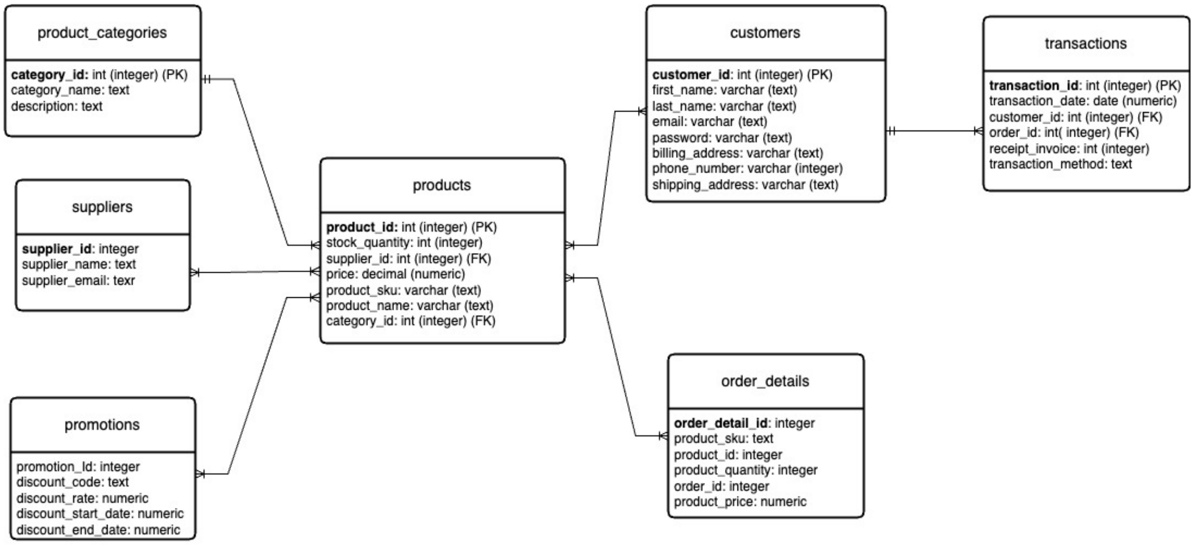

product_categories
category_id: int (integer) (PK)
category_name: text
description: text
suppliers
supplier_id: integer
supplier_name: text
supplier_email: texr
promotions
promotion_Id: integer
discount_code: text
discount_rate: numeric
discount_start_date: numeric
discount_end_date: numeric
products
product_id: int (integer) (PK)
stock_quantity: int (integer)
supplier_id: int (integer) (FK)
price: decimal (numeric)
product_sku: varchar (text)
product_name: varchar (text)
category_id: int (integer) (FK)
customers
customer_id: int (integer) (PK)
first_name: varchar (text)
last_name: varchar (text)
email: varchar (text)
password: varchar (text)
billing_address: varchar (text)
phone_number: varchar (integer)
shipping_address: varchar (text)
transactions
transaction_id: int (integer) (PK)
transaction_date: date (numeric)
customer_id: int (integer) (FK)
order_id: int(integer) (FK)
receipt_invoice: int (integer)
transaction_method: text
order_details
order_detail_id: integer
product_sku: text
product_id: integer
product_quantity: integer
order_id: integer
product_price: numeric

Figure 8: Logical Database Schema

Table 1: Entity Attribute Data Dictionary

Entity	Attribute	Description
1. Suppliers	supplier_ID	Primary Key Identifier for each unqiue supplier
	supplier_name	Name of the supplier
	supplier_email	Email Address of the supplier
2. Products	product_id	Primary Key Identifier for each unqiue Product
	category_id	Foreign Key Identifier for product categories since product has a many-to-one cardinality (N:1) with product categories
	supplier_id	Foreign Key identifier for suppliers since product has a many-to-many cardinality (N:M) with suppliers
	product_sku	Unique code assigned to each product for inventory tracking and management
	product_name	Name of the product
	price	Price of the product
	product_quantity	Number of products
	product_description	Short Description of the product
3. Product Categories	category_id	Primary Key Identifier for each unqiue product category
	category_name	Name of the product category
4. Customers	customer_id	Primary Key Identifier for each unqiue customer
	first_name	First Name of the customer
	last_name	Last Name of the customer
	email	Email Address of the customer
	password	Account Password of the customer
	phone_number	Phone Number of the customer
5. Transactions	transaction_id	Primary Key Identifier for each unqiue transaction
	customer_id	Foreign Key Identifier for customers since transactions has a many-to-many cardinality (N:M) with product categories
	order_id	Foreign Key Identifier for Orders since multiple transactions can have multiple orders.
	receipt_invoice	Invoice of the transaction

Entity	Attribute	Description
	transaction_method	Chosen method of transaction
	transaction_date	Date of transaction
6. Promotions	promotion_id	Primary Key Identifier for each unqiue promotion
	discount_code	Unique code for the discount
	discount_rate	Percentage of discount
	discount_start_date	Start date of the discount
	discount_end_date	End Date of the discount
7. Order details	order_detail_id	Primary Key Identifier for each unqiue Order detail
	order_id	Foreign Key Identifier for Orders since multiple order details can have multiple orders.
	product_id	Foreign Key Identifier for Products since multiple order details can have multiple order details.
	total_price	Total price of all the orders
	order_quantity	Number of orders
	shipping_address	Delivery address of the order
	city	City location of the order
	postcode	Postal code of the location

Table 2: Relationship Attribute Data Dictionary

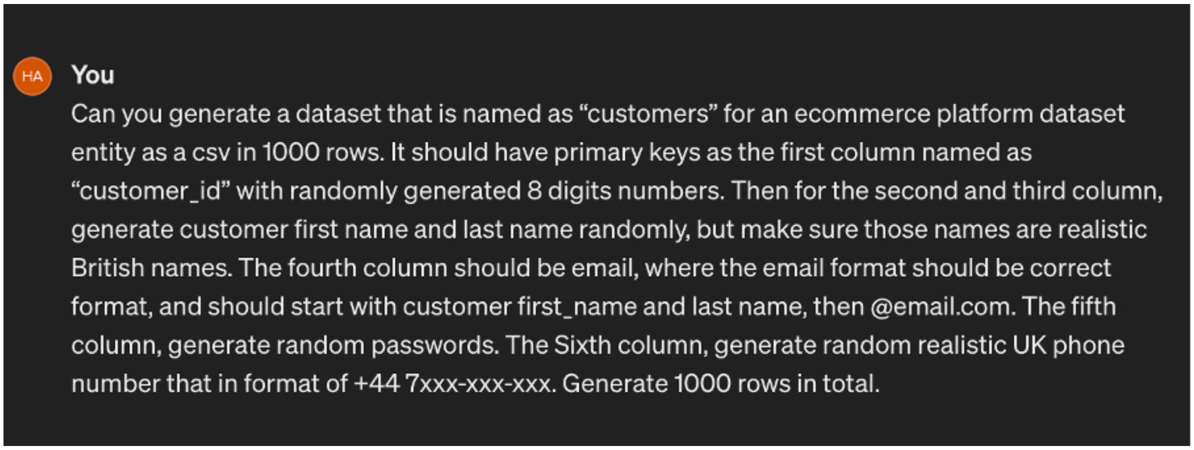
Relationship	Attribute	Description
Order	order_id	Primary Key Identifier for each unqiue Order
	customer_id	Foreign Key Identifier for customers since multiple orders can come from multiple orders.
	order_date	Date of order placement
	estimate_delivery_date	Delivery date of the order
	order_status	Classification of the order depending on its current status i.e., shipped, cancelled or pending

Data Generation and Management

2.1 Synthetic Data Generation:

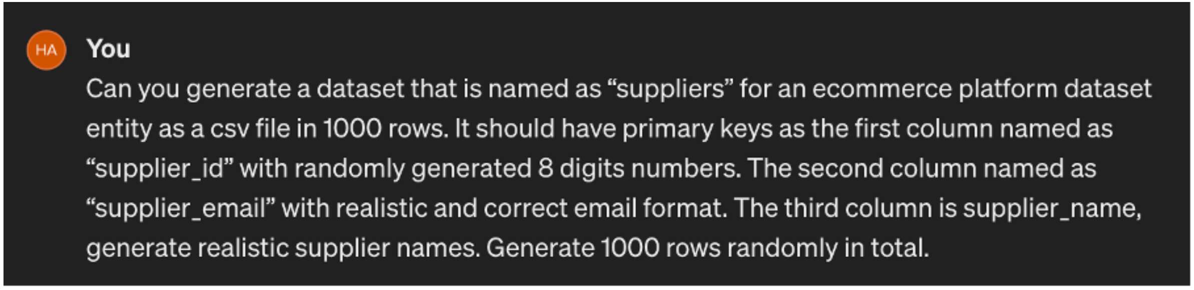
To generate synthetic data for our e-commerce database, we utilised ChatGPT. Our approach involved providing ChatGPT with detailed prompts specifying the attributes, relationships, and constraints relevant to each database entity. These prompts ensured that the data generated would reflect realistic e-commerce scenarios while maintaining consistency across entities.

Customers:
We constrained the “Customers” dataset to consist of UK phone numbers, as well as names with matching emails, ensuring a dataset that would closely resemble an actual customer database in the UK. See the following prompt:



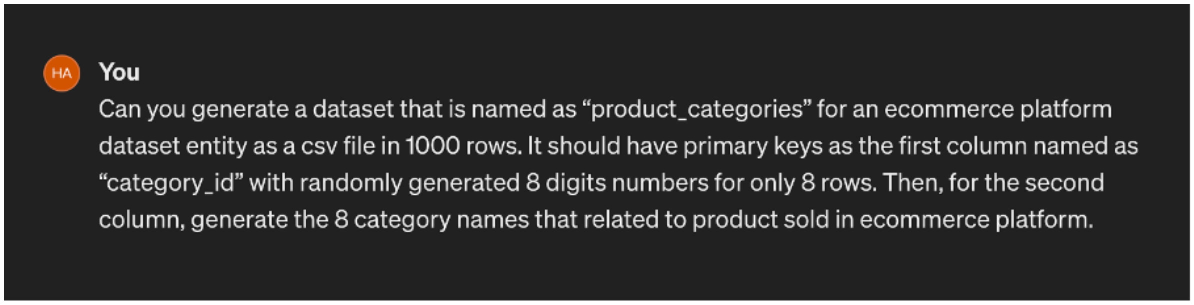
Can you generate a dataset that is named as “customers” for an ecommerce platform dataset entity as a csv in 1000 rows. It should have primary keys as the first column named as “customer_id” with randomly generated 8 digits numbers. Then for the second and third column, generate customer first name and last name randomly, but make sure those names are realistic British names. The fourth column should be email, where the email format should be correct format, and should start with customer first_name and last name, then @email.com. The fifth column, generate random passwords. The Sixth column, generate random realistic UK phone number that in format of +44 7xxx-xxx-xxx. Generate 1000 rows in total.

Suppliers:
In the “Suppliers” data set, we ensured that supplier names matched email formats. See the following prompt:



Can you generate a dataset that is named as “suppliers” for an ecommerce platform dataset entity as a csv file in 1000 rows. It should have primary keys as the first column named as “supplier_id” with randomly generated 8 digits numbers. The second column named as “supplier_email” with realistic and correct email format. The third column is supplier_name, generate realistic supplier names. Generate 1000 rows randomly in total.

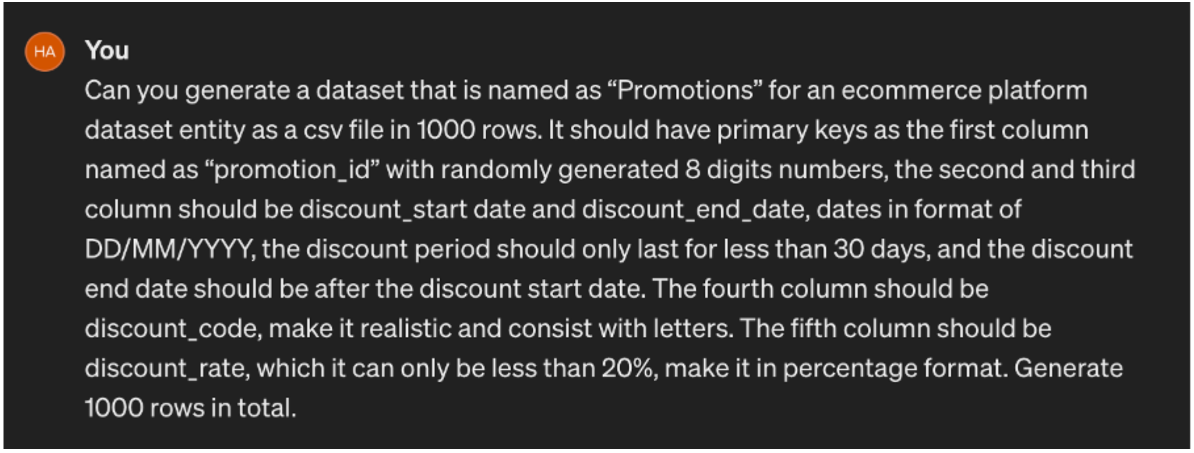
Product Categories:
The prompt is for the generation of “Product Categories”, with 8 unique category identifiers and corresponding names across 8 entries. See the following prompt:



You

Can you generate a dataset that is named as “product_categories” for an ecommerce platform dataset entity as a csv file in 1000 rows. It should have primary keys as the first column named as “category_id” with randomly generated 8 digits numbers for only 8 rows. Then, for the second column, generate the 8 category names that related to product sold in ecommerce platform.

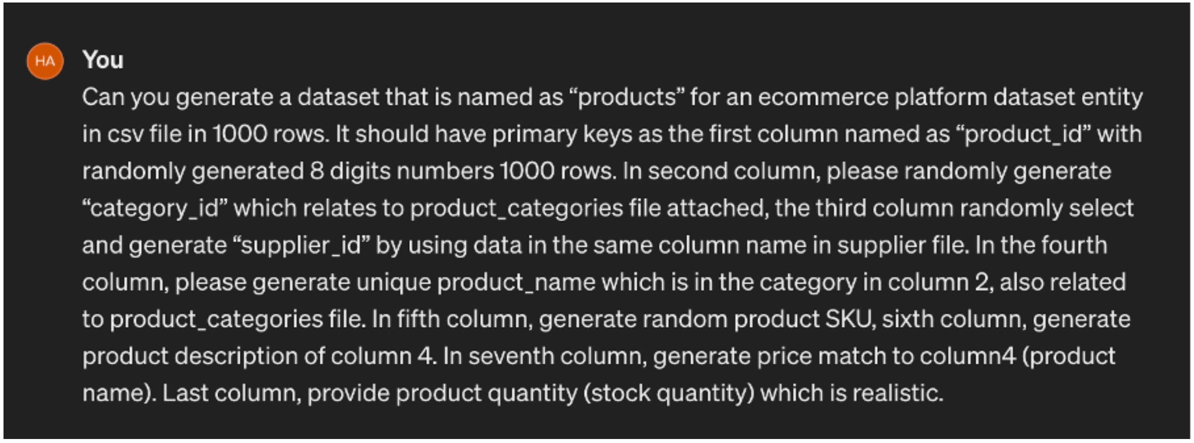
Promotions:
For the “Promotions” entity, we focused on generating realistic discount codes and rates, alongside logically constrained start, and end dates for the discounts. See the following prompt:



You

Can you generate a dataset that is named as “Promotions” for an ecommerce platform dataset entity as a csv file in 1000 rows. It should have primary keys as the first column named as “promotion_id” with randomly generated 8 digits numbers, the second and third column should be discount_start date and discount_end_date, dates in format of DD/MM/YYYY, the discount period should only last for less than 30 days, and the discount end date should be after the discount start date. The fourth column should be discount_code, make it realistic and consist with letters. The fifth column should be discount_rate, which it can only be less than 20%, make it in percentage format. Generate 1000 rows in total.

Products:
The prompt describes generating a “Products” dataset for an e-commerce platform, featuring unique product IDs, associated category and supplier IDs, product names, SKUs, descriptions, prices, and stock quantities, across 1000 rows. See the following prompt:

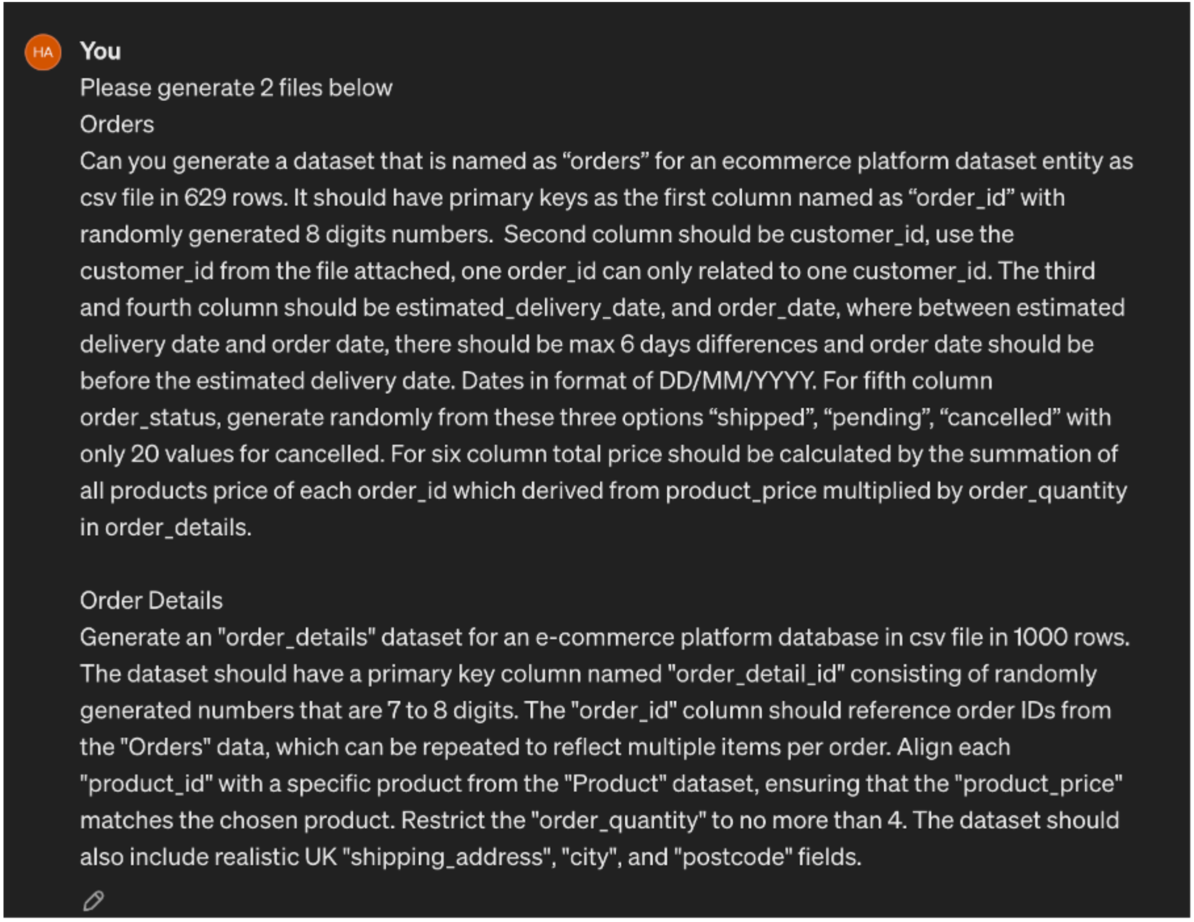


You

Can you generate a dataset that is named as “products” for an ecommerce platform dataset entity in csv file in 1000 rows. It should have primary keys as the first column named as “product_id” with randomly generated 8 digits numbers 1000 rows. In second column, please randomly generate “category_id” which relates to product_categories file attached, the third column randomly select and generate “supplier_id” by using data in the same column name in supplier file. In the fourth column, please generate unique product_name which is in the category in column 2, also related to product_categories file. In fifth column, generate random product SKU, sixth column, generate product description of column 4. In seventh column, generate price match to column4 (product name). Last column, provide product quantity (stock quantity) which is realistic.

Orders and Order Details:

This prompt outlines the creation of two interconnected datasets, “Orders” and “Order Details,” for an e-commerce platform, necessitating their simultaneous generation to maintain consistency. Within the datasets, we focused on imitating a realistic relationship between orders, customers, and products while incorporating accurate order statuses and delivery timelines. See the following prompt:



You

Please generate 2 files below

Orders

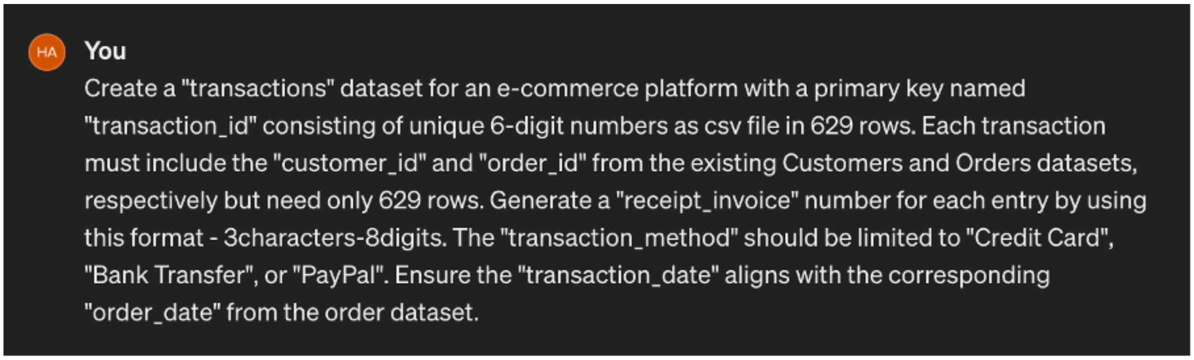
Can you generate a dataset that is named as “orders” for an ecommerce platform dataset entity as csv file in 629 rows. It should have primary keys as the first column named as “order_id” with randomly generated 8 digits numbers. Second column should be customer_id, use the customer_id from the file attached, one order_id can only related to one customer_id. The third and fourth column should be estimated_delivery_date, and order_date, where between estimated delivery date and order date, there should be max 6 days differences and order date should be before the estimated delivery date. Dates in format of DD/MM/YYYY. For fifth column order_status, generate randomly from these three options “shipped”, “pending”, “cancelled” with only 20 values for cancelled. For six column total price should be calculated by the summation of all products price of each order_id which derived from product_price multiplied by order_quantity in order_details.

Order Details

Generate an "order_details" dataset for an e-commerce platform database in csv file in 1000 rows. The dataset should have a primary key column named "order_detail_id" consisting of randomly generated numbers that are 7 to 8 digits. The "order_id" column should reference order IDs from the "Orders" data, which can be repeated to reflect multiple items per order. Align each "product_id" with a specific product from the "Product" dataset, ensuring that the "product_price" matches the chosen product. Restrict the "order_quantity" to no more than 4. The dataset should also include realistic UK "shipping_address", "city", and "postcode" fields.

Transactions:

In this prompt, we linked each transaction to customer and order details, including distinct invoice numbers and specific payment methods, while aligning transaction dates with their respective order dates. See the following prompt:



You

Create a "transactions" dataset for an e-commerce platform with a primary key named "transaction_id" consisting of unique 6-digit numbers as csv file in 629 rows. Each transaction must include the "customer_id" and "order_id" from the existing Customers and Orders datasets, respectively but need only 629 rows. Generate a "receipt_invoice" number for each entry by using this format - 3characters-8digits. The "transaction_method" should be limited to "Credit Card", "Bank Transfer", or "PayPal". Ensure the "transaction_date" aligns with the corresponding "order_date" from the order dataset.

The following prompts enabled us to generate 8 datasets for each entity, setting a strong foundation for robust analysis, given the realistic constraints and patterns put in place.

2.2 Data Import and Quality Assurance

After uploading the data files, the rows and columns of each dataset was displayed using the below code:

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

suffix <- "-Table 1"

# Rename files
for (file in data_files) {
  # Create a new filename
  new_filename <- paste0("data_uploads/Dataset/", gsub(suffix, "", file))
  file <- paste0("data_uploads/Dataset/", file)
  # Rename the file
  file.rename(from = file, to = new_filename)
}
```

To import the data sets into the SQLite database, we used the below code:

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

db_connection <- RSQLite::dbConnect(RSQLite::SQLite(),"ecommerce.db")

# To display the rows and columns of each dataset and
# To import each csv file into the database table
for (file in data_files) {
  this_filepath <- paste0("data_uploads/Dataset/", file)
  this_file_contents <- readr::read_csv(this_filepath)
  number_of_rows <- nrow(this_file_contents)
  number_of_columns <- ncol(this_file_contents)

  #To print the number of columns and rows of each dataset
  print(paste0("The file: ",file,
               " has: ",
               format(number_of_rows,big.mark = ","),
               " rows and ",
               number_of_columns," columns"))

  table_name <- gsub(".csv","",file)

  #Writing the csv file contents to the database and
  #creating the table with the table_name
  RSQLite::dbWriteTable(db_connection,table_name,this_file_contents,overwrite=TRUE)

  #To list the database tables
  RSQLite::dbListTables(db_connection)
}

RSQLite::dbDisconnect(db_connection)
```

3 Data Pipeline Generation

3.1 Github Repository and Workflow Setup

ETL (Extract, transform, load)

3.2 Github Actions for Continuous Integration

4. Data Analysis