# 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.

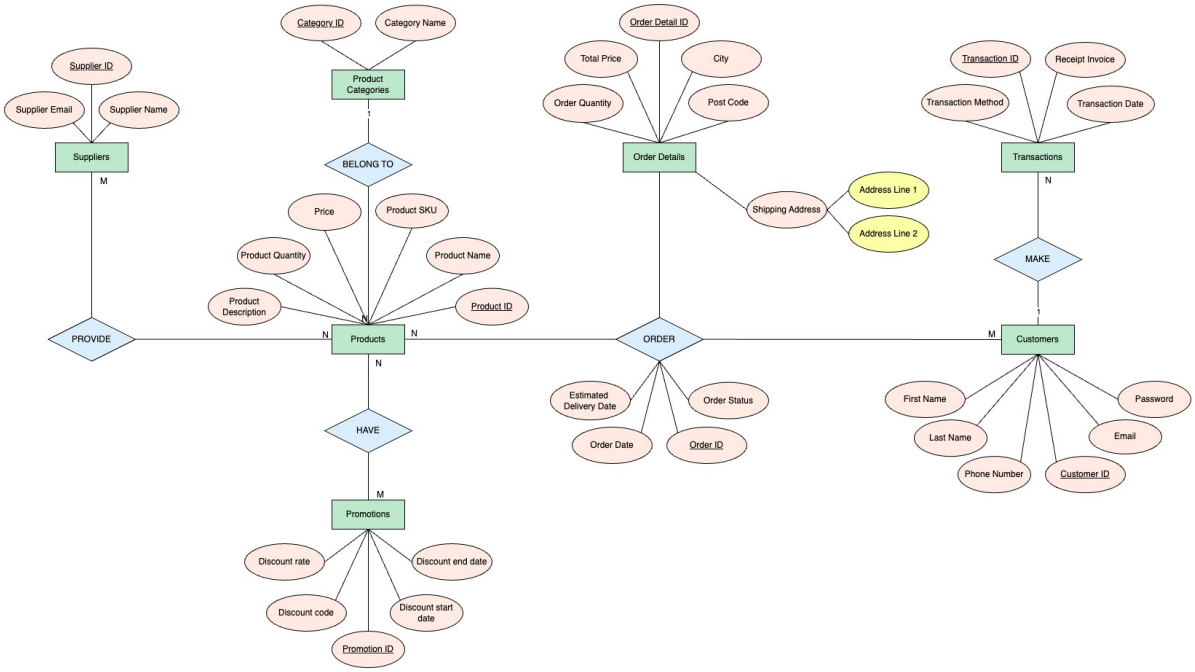


Figure 1: Entity-Relationship Diagram

2. Relationship Sets:

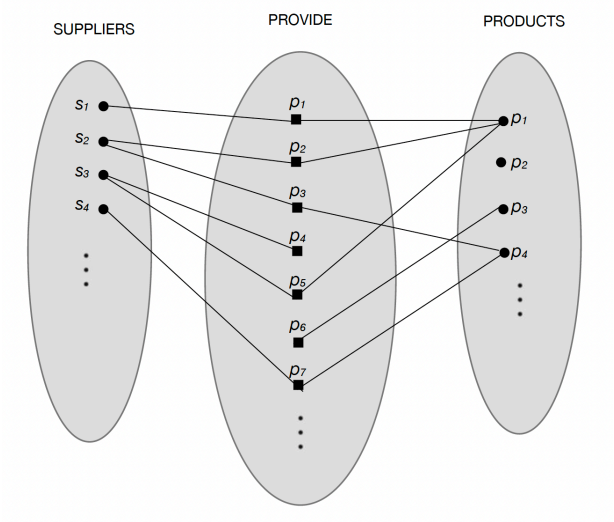


Figure 2: Suppliers PROVIDE products. Many-to-many relationship (M:N).

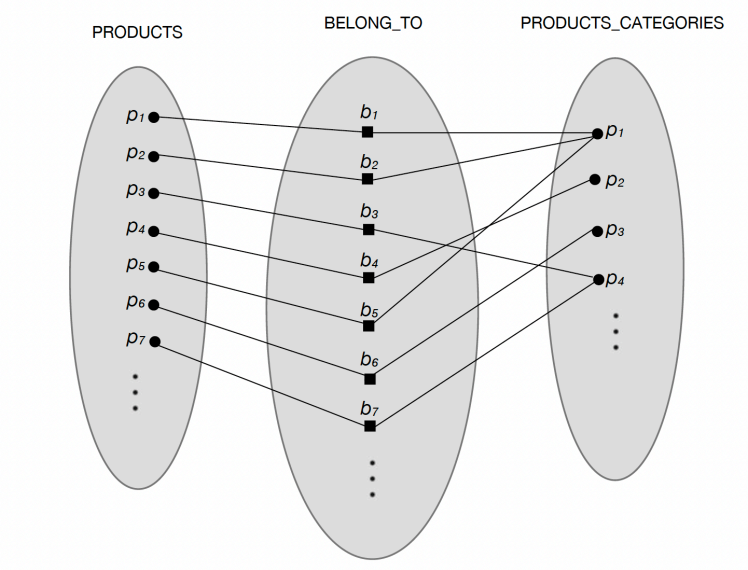


Figure 3: Products BELONG TO Product Categories. Many-to-one relationship (N:1).

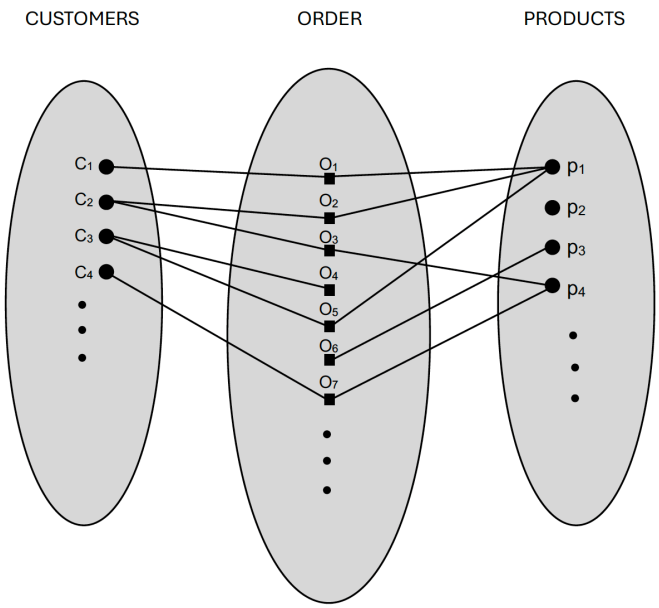


Figure 4: Customers ORDER products. Many-to-many relationship (M:N).

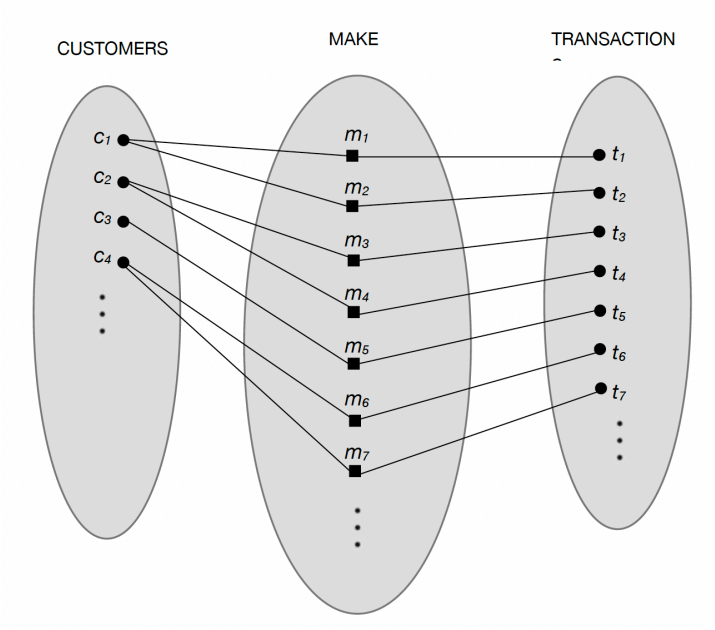


Figure 5: Customers MAKE Transaction. Many-to-many relationship (M:N).

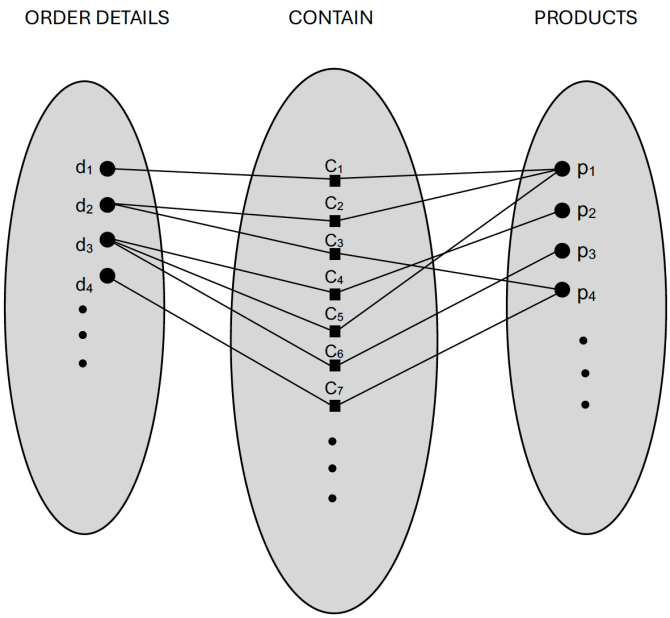


Figure 6: Order Details CONTAIN Products. One-to-many relationship (1:N).

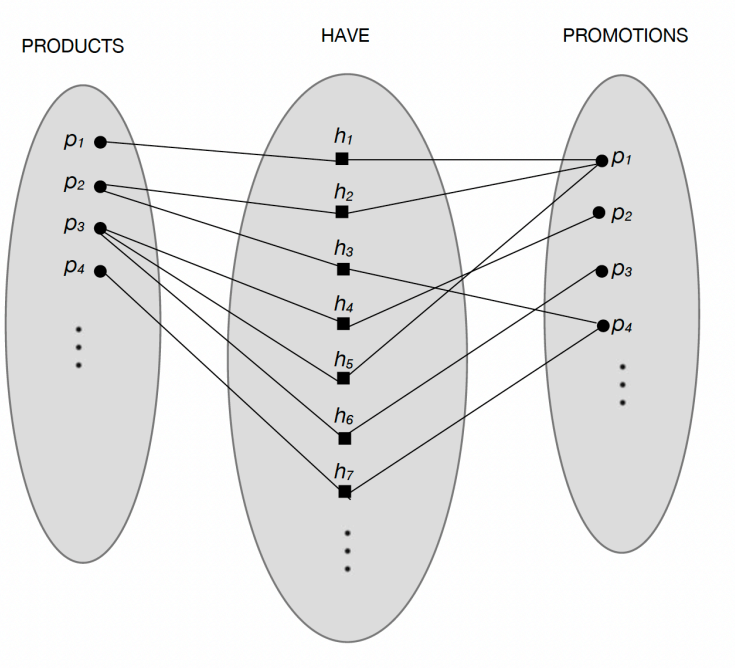


Figure 7: Products HAVE Promotions. Many-to-many relationship (N:M).

## 1.2 SQL Database Schema Creation:

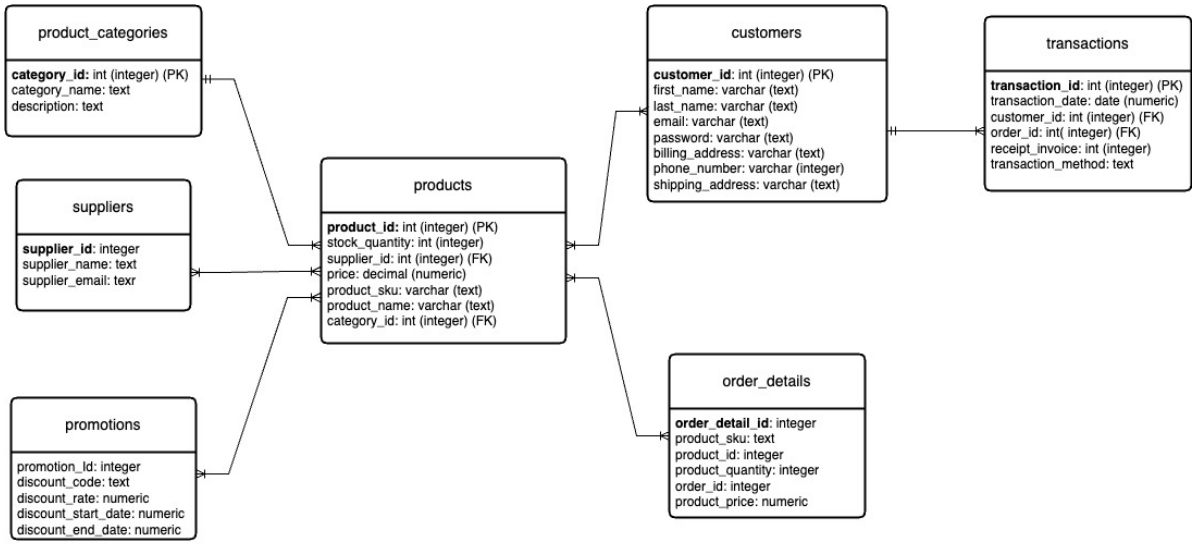


Figure 8: Logical Database Schema

“‘{.1 .cell-code}

:::

## Data Generation and Management

### 2.1 Synthetic Data Generation

### 2.2 Data Import and Quality Assurance

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


::: {.cell}

```{.r .cell-code}
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:

```
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