Data Management Group Assignment

DM Group 5

Contents

Database Design and Implementation

1. E-R Diagram Design:

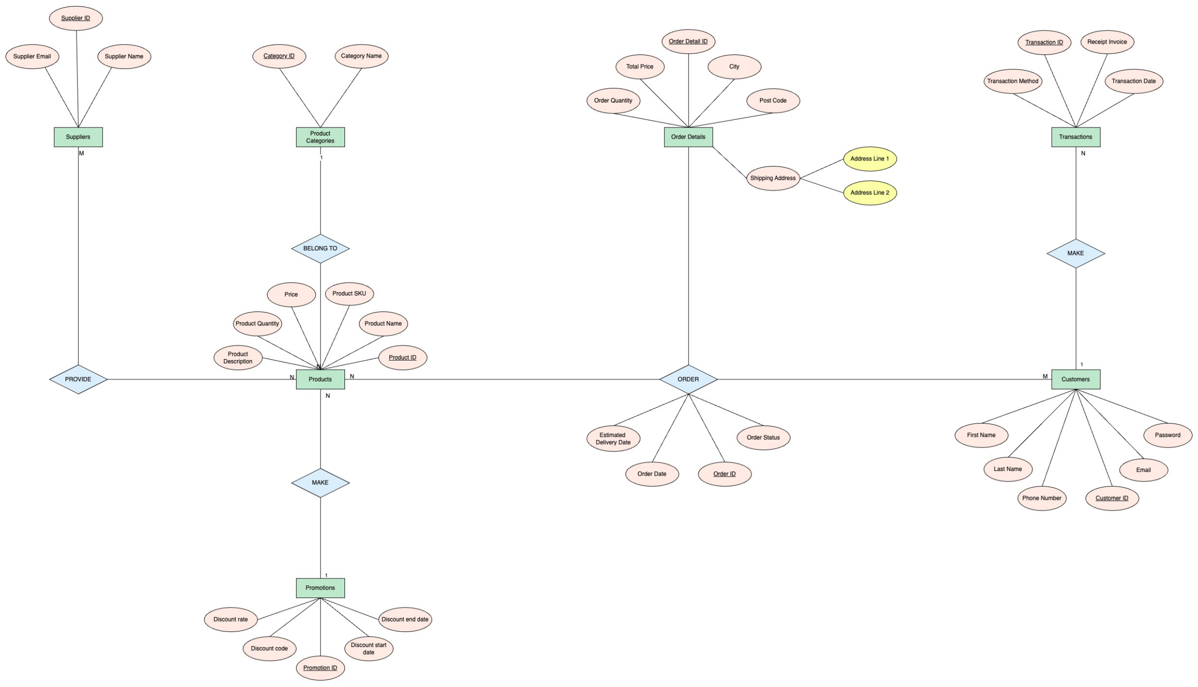

Figure 1: E-R Diagram

1.2 SQL Database Schema Creation

“‘{.1 .cell-code}

~~~
:::


::: {.cell}

:::


## 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 datasets 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")
```

```r
# 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)
}
```

```
Rows: 1000 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (6): customer_id, first_name, last_name, email, password, phone

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.

[1] "The file: customers.csv has: 1,000 rows and 6 columns"

Rows: 1000 Columns: 8
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (5): order_id, product_id, shipping_Address, city, postcode
dbl (3): order_detail_id, product_price, order_quantity
```

```
i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.

[1] "The file: order_details.csv has: 1,000 rows and 8 columns"

Rows: 1000 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (5): order_id, customer_id, estimated_delivery_date, order_date, order_s...
dbl (1): total_price

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.

[1] "The file: orders.csv has: 1,000 rows and 6 columns"

New names:
Rows: 1000 Columns: 5
-- Column specification
------------------------------------------------------- Delimiter: "," chr
(2): category_id, category_name lgl (3): ...3, ...4, ...5
i Use `spec()` to retrieve the full column specification for this data. i
Specify the column types or set `show_col_types = FALSE` to quiet this message.
* `` -> `...3`
* `` -> `...4`
* `` -> `...5`

[1] "The file: product_categories.csv has: 1,000 rows and 5 columns"

Rows: 1000 Columns: 8
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (6): product_id, category_id, supplier_id, product_name, product_sku, pr...
dbl (2): price, product_quantity

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.

[1] "The file: products.csv has: 1,000 rows and 8 columns"
```

```
Rows: 1000 Columns: 5
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (5): promotion_id, discount _start_date, discount_end_date, discount_cod...

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
[1] "The file: promotion.csv has: 1,000 rows and 5 columns"
```

```
New names:
Rows: 1000 Columns: 5
-- Column specification
-------------------------------------------------------- Delimiter: "," chr
(3): supplier_id, supplier_email, supplier_name lgl (2): ...4, ...5
i Use `spec()` to retrieve the full column specification for this data. i
Specify the column types or set `show_col_types = FALSE` to quiet this message.
* `` -> `...4`
* `` -> `...5`
```

```
[1] "The file: supplier.csv has: 1,000 rows and 5 columns"
```

```
Rows: 1000 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (5): customer_id, order_id, receipt_invoice, transaction_method, transac...
dbl (1): transaction_id

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
[1] "The file: transactions.csv has: 1,000 rows and 6 columns"
```

```r
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