E-commerceDB

Group-8

2024-02-27

Load necessary libraries

```
library(knitr)
library(kableExtra)
library(readr)
library(RSQLite)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following object is masked from 'package:kableExtra':
##
##      group_rows

## The following objects are masked from 'package:stats':
##
##      filter, lag

## The following objects are masked from 'package:base':
##
##      intersect, setdiff, setequal, union
```

```
library(stringr)
```

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

sql_file <- readLines("dbScript.sql")

for (sql_command in sql_file) {
  if (sql_command!=""){
   print(sql_command)
   dbExecute(con, sql_command)
   print("------------DONE---------")
  }
 }
```

```
## [1] "CREATE TABLE IF NOT EXISTS 'Category'( 'Category_ID' VARCHAR(250) PRIMARY KEY, 'Category_Name' '
## [1] "------------DONE---------"
## [1] "CREATE TABLE IF NOT EXISTS 'Suppliers'( 'Supplier_ID' VARCHAR(250) PRIMARY KEY, 'Supplier_Name'
## [1] "------------DONE---------"
## [1] "CREATE TABLE IF NOT EXISTS 'Discounts'( 'Discount_Code' VARCHAR(50) PRIMARY KEY, 'Discount_Amou
## [1] "------------DONE---------"
## [1] "CREATE TABLE IF NOT EXISTS 'Products' ( 'Product_ID' VARCHAR(250) PRIMARY KEY, 'Product_Name' V
```

```
## [1] "-------------DONE---------"
## [1] "CREATE TABLE IF NOT EXISTS 'Customers' ( 'Cust_ID' VARCHAR(250) PRIMARY KEY, 'Cust_Name' VARCHA
## [1] "-------------DONE---------"
## [1] "CREATE TABLE IF NOT EXISTS 'Reviews' ( 'Review_ID' VARCHAR(50) PRIMARY KEY,'Review_Timestamp' D
## [1] "-------------DONE---------"
## [1] "CREATE TABLE IF NOT EXISTS 'Order_Items'( 'Quantity' INT NOT NULL, 'Sum_Price' INT NOT NULL, 'O
## [1] "-------------DONE---------"
## [1] "CREATE TABLE IF NOT EXISTS 'Order_Details' ( 'Order_ID'  VARCHAR(250) PRIMARY KEY, 'Order_Date'
## [1] "-------------DONE---------"
```

```r
Customers <- read_csv("Files/Customers.csv")
```

```
## Rows: 150 Columns: 10
## -- Column specification ----------------------------------------------------
## Delimiter: ","
## chr (7): Cust_ID, Cust_First_Name, Cust_Last_Name, Cust_Building_Name, Cust_...
## dbl (3): Cust_Building_Number, Cust_Phone_Number, Phone_Country_Code
##
## 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.
```

```r
Products <- read_csv("Files/Products.csv")
```

```
## Rows: 150 Columns: 4
## -- Column specification ----------------------------------------------------
## Delimiter: ","
## chr (3): Product_ID, Product_Name, Product_Availability
## dbl (1): Product_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.
```

```r
Order_details <- read_csv("Files/Order_Details.csv",
    skip = 1)
```

```
## Rows: 150 Columns: 13
## -- Column specification ----------------------------------------------------
## Delimiter: ","
## chr  (10): Order_ID, Shipping_Building_Name, Shipping_Street_Name, Shipping_...
## dbl   (2): Shipping_Building_Number, Billing_Building_Number
## dttm  (1): Order_Date
##
## 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.
```

```r
Reviews <- read_csv("Files/Reviews.csv",
    skip = 1)
```

```
## Rows: 150 Columns: 5
## -- Column specification ----------------------------------------------------
## Delimiter: ","
## chr  (2): Review_ID, Review_Text
## dbl  (2): Product_Rating, Review_Likes
## dttm  (1): Review_Timestamp
##
## 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.
```

```r
Suppliers <- read_csv("Files/Suppliers.csv")
```

```
## Rows: 50 Columns: 8
## -- Column specification ---------------------------------------------------------
## Delimiter: ","
## chr (6): Supplier_ID, Supplier_Name, Supplier_Building_Name, Supplier_Street...
## dbl (2): Supplier_Building_Number, Supplier_Zip_Code
##
## 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.
```

```r
Product_Discounts <- read_csv("Files/Product_Discounts.csv")
```

```
## Rows: 50 Columns: 3
## -- Column specification ---------------------------------------------------------
## Delimiter: ","
## chr (1): Discount_Code
## dbl (1): Discount_Amount
## lgl (1): Discount_Status
##
## 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.
```

```r
Product_Category <- read_csv("Files/Product_Category.csv")
```

```
## Rows: 10 Columns: 2
## -- Column specification ---------------------------------------------------------
## Delimiter: ","
## chr (2): Category_ID, Category_Name
##
## 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.
```

```r
Order_Items <- read_csv("Files/Order_Items.csv")
```

```
## Rows: 56 Columns: 2
## -- Column specification ---------------------------------------------------------
## Delimiter: ","
## dbl (1): Quantity
## num (1): Sum_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.
```

```r
# To create empty column for the Products table
Products <- Products %>%
  mutate(Category_ID = NA)

# To apply the foreign key into the table
# Define a function to assign Category_ID based on keywords in Product_Name
assign_category_id <- function(Product_Name) {
  if (grepl("TV|Television", Product_Name, ignore.case = TRUE)) {
    return("CAT1")
  } else if (grepl("Laptop|Tablet|Computing|Book|Surface|Monitor", Product_Name, ignore.case = TRUE)) {
    return("CAT2")
  } else if (grepl("Phone|Galaxy|Mi|P Series|OnePlus", Product_Name, ignore.case = TRUE)) {
```

```r
    return("CAT3")
  } else if (grepl("Refrigerator|Washing Machine|Home Appliance|Microwave|Vacuum|Dishwasher", Product_Na
    return("CAT4")
  } else if (grepl("Headphones|Speakers|Sound System|Earbuds|Speaker|Technica|Soundbar", Product_Name, i
    return("CAT5")
  } else if (grepl("Camera|Photography|GoPro|Mirrorless|Nikon|Camcorder|Compact", Product_Name, ignore.c
    return("CAT6")
  } else if (grepl("Xbox|PS|Gaming|Switch", Product_Name, ignore.case = TRUE)) {
    return("CAT7")
  } else if (grepl("Smart Home|Echo|Smart Lock|Steam Deck|Hue Light", Product_Name, ignore.case = TRUE))
    return("CAT8")
  } else if (grepl("Watch|Wearable|Quest|Tracker|Gear|Band|Glasses", Product_Name, ignore.case = TRUE))
    return("CAT9")
  } else if (grepl("Keyboard|Mouse|Peripheral|Thermostat", Product_Name, ignore.case = TRUE)) {
    return("CAT10")
  } else {
    return(NA) # For products that do not match any category
  }
}

# Apply the function to assign Category_ID to each product
Products$Category_ID <- sapply(Products$Product_Name, assign_category_id)

# This is to add suppliers_id in Products
set.seed(123)

Products <- Products %>%
  mutate(Supplier_ID = NA)

# Create a function to find matching supplier ID or assign randomly if no match is found
assign_supplier_id <- function(Product_Name, Suppliers) {
  for (i in 1:nrow(Suppliers)) {
    if (str_detect(Product_Name, regex(Suppliers$Supplier_Name[i], ignore_case = TRUE))) {
      return(Suppliers$Supplier_ID[i])
    }
  }
  # If no match found, assign a random supplier ID
  random_supplier_id <- sample(Suppliers$Supplier_ID, 1)
  return(random_supplier_id)
}

Products$Supplier_ID <- sapply(Products$Product_Name, function(x) assign_supplier_id(x, Suppliers))

# Adding Discount_Code column into Products
set.seed(123) # This is to ensure reproducibility

Products <- Products %>%
  mutate(Discount_Code = NA)

codes_to_assign <- sample(1:nrow(Products), 50)

random_discounts <- sample(Product_Discounts$Discount_Code, 50)

Products$Discount_Code[codes_to_assign] <- random_discounts
```

```r
# Product_ID column for reviews table
set.seed(123)
Reviews <- Reviews %>%
  mutate(Product_ID = sample(Products$Product_ID, nrow(Reviews), replace = TRUE))

# Adding Cust_ID column for Order_details table.
set.seed(123)
Order_details <- Order_details %>%
  mutate(Cust_ID = sample(Customers$Cust_ID, nrow(Order_details), replace = TRUE))

# Filter out the rows from Products that have a Disocunt_Code assigned
discounted_products <- Products %>%
  filter(!is.na(Discount_Code)) %>%
  select(Product_ID, Discount_Code)

# Do a left join to join it together
Product_Discounts <- Product_Discounts %>%
  left_join(discounted_products, by = "Discount_Code")

# Same step for cat_id
# Filter out the rows from Products that have a discount code assigned on the cat ID
discounted_cat <- Products %>%
  filter(!is.na(Category_ID)) %>%
  select(Category_ID, Discount_Code)

# Do a left join to join it together, thus we get to see which discount code assign to which category o
Product_Discounts <- Product_Discounts %>%
  left_join(discounted_cat, by = "Discount_Code")
```

Part 1: Database Design and Implementation

1.1 Entity Relationship Diagram

- Here Insert ERD *

The E-R diagram above simulates a real-world e-commerce data ecosystem, capturing the detailed relationships between entities and attributes essential for facilitating online transactions. In addition, it provides a comprehensive view of the e-commerce system, which serves as a platform for users to browse products, make purchases, and securely complete their payments.

1.1.1.Assumptions

- The company only distributes products within the United Kingdom (UK).

- The Currency used is Pound Sterling (GBP).

- Attributes formats will be aligned with UK standard formats such as date , addresses , names . . . etc

1.1.2. Entities and Attributes

This section describes and illustrates the entities and their respective attributes of the ER diagram above.

- 1.1.2.1. Customer shows us the users who previously have at least once purchased products and placed an order.

- 1.1.2.2. Supplier Vendors who provide products. Represent the source of the product items.

- 1.1.2.3 Product Information about the model and price of the products as well as the availability of it.

- 1.1.2.4 Order_details Contain information of orders including billing and shipping address, order and payment status, as well as order date and payment type.

- 1.1.2.5 Product_Category Category of products offered by the e-commerce website.

- 1.1.2.6 Product_Discounts Discount code that could be applied and the amount of discount it offers as well as the status of the discount.

- 1.1.2.7 Reviews contain information of the reviews including text and rating on product and the likes of the top reviews as well as the time stamp of when the review was made.

Design Considerations

- Absence of an Order Entity:

- The model intentionally skips direct order management. Instead, it focuses on product management and customer interactions through reviews and payment methods.Additionally, This consideration will guarantee that products purchased by customers are not tracked or stored by the system to align with privacy policies.

- Order Entity not considered in this ER design in order to follow best practices by not having to include orderId as part of product table which might affect the overall performance of DB retrieval.

- Customer Engagement: By including Reviews, the model emphasizes customer engagement and feedback without directly managing transactions.

- Payment Information: Including Payment_Method without an Order entity suggests a pre-registration of payment preferences or a simplified wallet storage that could be expanded in the future.

Logical Schema

Customers ($\underline{Customer_id}$, Customer_Email, Cust_F_Name, Cust_L_Name, Phone_Country_Code, Phone_Num, Cust_Street_Name, Cust_Building_Name, Cust_Zip_Code)

Products ($\underline{Product_id}$, $Discount\_Code$, $Category\_id$, Product_Name, Product_Price, Product_Availability)

Suppliers ($\underline{Supplier_id}$, Supplier_Email, Supplier_Name, Supplier_Status, Sup_Building_Name, Sup_Street_Name, Sup_Zip_Code)