E-commerceDB

Group-8

2024-02-27

Load necessary libraries

```
library(DBI)
library(readr)
library(RSQLite)
library(dplyr)
```

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

```
## 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' V
## [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 'Products' ( 'Product_ID' VARCHAR(250) PRIMARY KEY, 'Product_Name' VA
## [1] "-------------DONE---------"
## [1] "CREATE TABLE IF NOT EXISTS 'Customers' ( 'Cust_ID' VARCHAR(250) PRIMARY KEY, 'Cust_Name' VARCHAR
## [1] "-------------DONE---------"
## [1] "CREATE TABLE IF NOT EXISTS 'Reviews' ( 'Review_ID' VARCHAR(50) PRIMARY KEY,'Review_Timestamp' DA
## [1] "-------------DONE---------"
## [1] "CREATE TABLE IF NOT EXISTS 'Discounts'( 'Discount_Code' VARCHAR(50) PRIMARY KEY, 'Discount_Amoun
## [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---------"
```

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

```
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.
```

```
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.
```

```
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.
```

```
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_Item <- read_csv("Files/Order_Item.csv",
    skip = 1)
```

```
## Rows: 200 Columns: 1
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (1): Order_Item
##
## 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)) {
```

```
    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, 
    return("CAT5")
  } else if (grepl("Camera|Photography|GoPro|Mirrorless|Nikon|Camcorder|Compact", Product_Name, ignore.
    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))
```

```r
# 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))
```

```r
# 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")
```

```r
# Order_Items, product_ID
# Randomly assign a product ID, and order id (Order ID can repeat for at least 30 rows)
set.seed(123)
Order_Item <- Order_Item %>%
  mutate(Product_ID = NA)

# Assign first 150 unique Product_IDs to the first 150 rows
Order_Item$Product_ID[1:150] <- sample(Products$Product_ID, size = 150, replace = FALSE)

# For the remaining 50 rows, randomly assign Product_IDs (allowing repeats)
Order_Item$Product_ID[151:200] <- sample(Products$Product_ID, size = 50, replace = TRUE)

# Joining Order_Item with Products to get the Price for each Product_ID
Order_Item <- merge(Order_Item, Products[, c("Product_ID", "Product_Price")], by = "Product_ID", all.x =

Order_Item <- Order_Item %>% rename(Quantity = Order_Item)
# Calculating Sum_Price as Price * Quantity
Order_Item <- Order_Item %>%
  mutate(Sum_Price = Product_Price * Quantity)

# Remove the Product_Price column
Order_Item$Product_Price <- NULL
```

```r
set.seed(123)
Order_Item <- Order_Item %>%
```

```r
  mutate(Order_ID = NA)

# Assign first 150 unique Order_IDs to the first 150 rows
Order_Item$Order_ID[1:150] <- sample(Order_details$Order_ID, size = 150, replace = FALSE)

# Function to assign unique Order_IDs avoiding duplicates for each Product_ID, it intended to assign un
assign_unique_order_ids_full_range <- function(order_item, order_details) {
  # Get all unique Order_IDs from Order_Details
  all_order_ids <- unique(Order_details$Order_ID)

  # Iterate over each row in order_item
  for (i in 151:nrow(Order_Item)) { # The first 150 are pre-assigned
    product_id <- Order_Item$Product_ID[i]

    # Find Order_IDs used by the same Product_ID
    used_order_ids <- Order_Item$Order_ID[Order_Item$Product_ID == product_id]

    # Available Order_IDs are those not yet used by this Product_ID
    available_order_ids <- setdiff(all_order_ids, used_order_ids)

    if (length(available_order_ids) == 0) {
      stop("Ran out of unique Order_IDs to assign for Product_ID: ", product_id)
    }

    # Randomly select an available Order_ID for the Product_ID
    Order_Item$Order_ID[i] <- sample(available_order_ids, 1)
  }

  return(Order_Item)
}

# To apply this function:
Order_Item <- assign_unique_order_ids_full_range(Order_Item, Order_details)
```

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 (*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 (*Product_id*, *Discount_Code*, *Category_id*, Product_Name, Product_Price, Product_Availability)

Suppliers (*Supplier_id*, Supplier_Email, Supplier_Name, Supplier_Status, Sup_Building_Name, Sup_Street_Name, Sup_Zip_Code)

```
# We need to rename and delete columns like building number, as it does not match or does not exist in
# Suppliers amendment
Suppliers$Supplier_Building_Number <- NULL
Suppliers <- Suppliers %>% rename(Supplier_Zip = Supplier_Zip_Code)

# Customers amendment
Customers <- Customers %>% rename(Cust_Zip = Cust_Zip_Code)
Customers <- Customers %>% rename(Cust_Country_Code = Phone_Country_Code)


# Order_details
```

```r
# Don't have these in the database
Order_details$Billing_Building_Number <- NULL
Order_details$Shipping_Building_Number <- NULL

#This is one of the ways of doing it , havent do order_details pending for changes from abigail.
RSQLite::dbWriteTable(con, "Category", Product_Category, overwrite=TRUE)
RSQLite::dbWriteTable(con, "Suppliers", Suppliers, overwrite=TRUE)
RSQLite::dbWriteTable(con, "Discounts", Product_Discounts, overwrite=TRUE)
RSQLite::dbWriteTable(con, "Products", Products, overwrite=TRUE)
RSQLite::dbWriteTable(con, "Customers", Customers, overwrite=TRUE)
RSQLite::dbWriteTable(con, "Reviews", Reviews, overwrite=TRUE)
RSQLite::dbWriteTable(con, "Order_Items", Order_Item, overwrite=TRUE)
RSQLite::dbWriteTable(con, "Order_Details", Order_details, overwrite=TRUE)
```

```r
SELECT *
FROM Reviews
```

Table 1: Displaying records 1 - 10

Review_ID	Review_Timestamp	Product_Rating	Review_Text	Review_Likes	Product_ID
R1775367	1680742068	0	Poor quality, broke after a few uses.	101	P5362182
R6792460	1691648504	5	Absolutely love it! Exceeded my expectations.	750	P8478278
R0255778	1701195297	3	It's okay, does the job but nothing special.	150	P7038427
R9658932	1680766145	4	Fantastic product, highly recommend.	824	P7596348
R0871453	1692394329	5	Works perfectly, couldn't be happier.	474	P5362182
R0110957	1705246817	4	Works perfectly, would buy again.	558	P7038427
R2016495	1690974238	0	Terrible customer service, not worth the hassle.	935	P8699814
R4565193	1706209154	2	Regret this purchase, expected much more.	577	P8864249
R5969076	1704683697	3	Satisfactory, but I had higher expectations.	842	P8864249
R2500257	1704252515	4	Great value for the price, very satisfied.	972	P6026468

```r
SELECT *
FROM Products
WHERE Supplier_ID = "S001"
```

Table 2: 9 records

Product_ID	Product_Name	Product_Price	Product_Availability	Category_ID	Supplier_ID	Discount_Code
P4637031	Apple iPhone Model 49	1421.52	Out of Stock	CAT3	S001	NA
P1961681	Apple Watch Model 135	1184.14	Limited Availability	CAT9	S001	NA
P4081539	Apple MacBook Model 24	1239.98	In Stock	CAT2	S001	NA
P6530444	Apple iPhone Model 51	63.14	Limited Availability	CAT3	S001	NA

Product_ID	Product_Name	Product_Price	Product_Availability	Category_ID	Supplier_ID	Discount_Code
P1857653	Apple MacBook Model 22	1885.11	Limited Availability	CAT2	S001	NA
P0038292	Apple MacBook Model 23	1464.99	In Stock	CAT2	S001	NA
P8699814	Apple iPhone Model 50	1679.01	In Stock	CAT3	S001	DISCOUNT16
P8038804	Apple Watch Model 134	619.29	Limited Availability	CAT9	S001	NA
P5084276	Apple Watch Model 133	1813.57	Limited Availability	CAT9	S001	DISCOUNT11

```
# I want to create a for loop so we can do it one shot. Still trying to think through it.
```