**#Extract, Transform, Load in R studio**

#installing the required packages

install.packages("dplyr")

library("dplyr")

install.packages("lubridate")

library(lubridate)

#country dataset

country <- read.table("C:\\Users\\LabStudent-55-706949\\Downloads\\Orion's Sword Data-20231018T204118Z-001\\Orion\_s Sword Data\\country.txt", header = TRUE, sep = ",")

View(country)

#customer dataset

customer <- read.table("C:\\Users\\LabStudent-55-706949\\Downloads\\Orion's Sword Data-20231018T204118Z-001\\Orion\_s Sword Data\\customer.txt", header = TRUE, sep = ",")

View(customer)

#customer type dataset

customer\_type <- read.table("C:\\Users\\LabStudent-55-706949\\Downloads\\Orion's Sword Data-20231018T204118Z-001\\Orion\_s Sword Data\\customer\_type.txt", header = TRUE, sep = ",")

View(customer\_type)

#discount dataset

discount <- read.table("C:\\Users\\LabStudent-55-706949\\Downloads\\Orion's Sword Data-20231018T204118Z-001\\Orion\_s Sword Data\\discount.txt", header = TRUE, sep = ",")

View(discount)

#order dataset

order <- read.table("C:\\Users\\LabStudent-55-706949\\Downloads\\Orion's Sword Data-20231018T204118Z-001\\Orion\_s Sword Data\\order.txt", header = TRUE, sep = ",")

View(order)

#order\_item dataset

order\_item <- read.table("C:\\Users\\LabStudent-55-706949\\Downloads\\Orion's Sword Data-20231018T204118Z-001\\Orion\_s Sword Data\\order\_item.txt", header = TRUE, sep = ",")

View(order\_item)

#price\_list dataset

price\_list <- read.table("C:\\Users\\LabStudent-55-706949\\Downloads\\Orion's Sword Data-20231018T204118Z-001\\Orion\_s Sword Data\\price\_list.txt", header = TRUE, sep = ",")

View(price\_list)

#product dataset

product <- read.table("C:\\Users\\LabStudent-55-706949\\Downloads\\Orion's Sword Data-20231018T204118Z-001\\Orion\_s Sword Data\\product.txt", header = TRUE, sep = ",")

View(product)

#product\_level dataset

product\_level <- read.table("C:\\Users\\LabStudent-55-706949\\Downloads\\Orion's Sword Data-20231018T204118Z-001\\Orion\_s Sword Data\\product\_level.txt", header = TRUE, sep = ",")

View(product\_level)

#feature selection:

country <- country %>% select(-Population,-Country\_ID, -Continent\_ID , -Country\_Former\_Name)

View(country)

customer <- customer %>% select(-Gender, -Personal\_ID, -Birth\_Date, -Customer\_Address, -Street\_ID, -Street\_Number)

View(customer)

order <- order %>% select(-Employee\_ID)

View(order)

product <- product %>% select(-Supplier\_ID, -Product\_Ref\_ID)

View(product)

#Transforming the data

#country data

#record count before

row <- nrow(country)

column <- ncol(country)

# Check for missing values

missing\_values <- colSums(is.na(country))

cat("Missing values:\n")

print(missing\_values)

# Remove rows with missing values

country <- na.omit(country)

# Check for missing values after transformation

missing\_values <- colSums(is.na(country))

cat("Missing values:\n")

print(missing\_values)

# Check data types

data\_types <- sapply(country, class)

cat("Data Types:\n")

print(data\_types)

# Check for duplicates

duplicates <- anyDuplicated(country)

cat("Duplicates:\n")

print(duplicates)

#record count after

rowafter <- nrow(country)

columnafter <- ncol(country)

cat("Number of rows before:\n")

print(row)

cat("Number of columns before:\n")

print(column)

cat("Number of rows after:\n")

print(rowafter)

cat("Number of columns after:\n")

print(columnafter)

A screenshot of a computer program

Description automatically generated

#customer data

#record count before

row <- nrow(customer)

column <- ncol(customer)

# Check for missing values

missing\_values <- colSums(is.na(customer))

cat("Missing values:\n")

print(missing\_values)

# Check data types

data\_types <- sapply(customer, class)

cat("Data Types:\n")

print(data\_types)

# Check for duplicates

duplicates <- anyDuplicated(customer)

cat("Duplicates:\n")

print(duplicates)

#record count after

rowafter <- nrow(customer)

columnafter <- ncol(customer)

cat("Number of rows before:\n")

print(row)

cat("Number of columns before:\n")

print(column)

cat("Number of rows after:\n")

print(rowafter)

cat("Number of columns after:\n")

print(columnafter)

A screenshot of a computer code

Description automatically generated

#customer\_type data:

#record count before

row <- nrow(customer\_type)

column <- ncol(customer\_type)

# Check for missing values

missing\_values <- colSums(is.na(customer\_type))

cat("Missing values:\n")

print(missing\_values)

# Check data types

data\_types <- sapply(customer\_type, class)

cat("Data Types:\n")

print(data\_types)

# Check for duplicates

duplicates <- anyDuplicated(customer\_type)

cat("Duplicates:\n")

print(duplicates)

#record count after

rowafter <- nrow(customer\_type)

columnafter <- ncol(customer\_type)

cat("Number of rows before:\n")

print(row)

cat("Number of columns before:\n")

print(column)

cat("Number of rows after:\n")

print(rowafter)

cat("Number of columns after:\n")

print(columnafter)

A screenshot of a computer code

Description automatically generated

#order data

#record count before

row <- nrow(order)

column <- ncol(order)

# Check for missing values

missing\_values <- colSums(is.na(order))

cat("Missing values:\n")

print(missing\_values)

# Check data types

data\_types <- sapply(order, class)

cat("Data Types:\n")

print(data\_types)

# Convert the 'date\_column' from character to date YYYY-MM-DD

order$Order\_Date <- as.Date(order$Order\_Date, format = "%d%b%Y")

order$Delivery\_Date <- as.Date(order$Delivery\_Date, format = "%d%b%Y")

# Check data types data quality check

data\_types <- sapply(order, class)

cat("Data Types:\n")

print(data\_types)

# Check for duplicates

duplicates <- anyDuplicated(order)

cat("Duplicates:\n")

print(duplicates)

#record count after

rowafter <- nrow(order)

columnafter <- ncol(order)

cat("Number of rows before:\n")

print(row)

cat("Number of columns before:\n")

print(column)

cat("Number of rows after:\n")

print(rowafter)

cat("Number of columns after:\n")

print(columnafter)

A screenshot of a computer code

Description automatically generated

#order\_item data

#record count before

row <- nrow(order\_item)

column <- ncol(order\_item)

# Check for missing values

missing\_values <- colSums(is.na(order\_item))

cat("Missing values:\n")

print(missing\_values)

# Check data types

data\_types <- sapply(order\_item, class)

cat("Data Types:\n")

print(data\_types)

# Remove dollar sign and commas, then convert to numeric

order\_item$Total\_Retail\_Price <- as.numeric(gsub("[^0-9.]", "", order\_item$Total\_Retail\_Price))

#convert discount from character to integer

order\_item$Discount <- gsub("\\.", "0%", order\_item$Discount)

order\_item$Discount <- gsub("\\%", "", order\_item$Discount)

order\_item$Discount <- as.numeric(order\_item$Discount)

# Check data types after transformation

data\_types <- sapply(order\_item, class)

cat("Data Types:\n")

print(data\_types)

# Check for duplicates

duplicates <- anyDuplicated(order\_item)

cat("Duplicates:\n")

print(duplicates)

#record count after

rowafter <- nrow(order\_item)

columnafter <- ncol(order\_item)

cat("Number of rows before:\n")

print(row)

cat("Number of columns before:\n")

print(column)

cat("Number of rows after:\n")

print(rowafter)

cat("Number of columns after:\n")

print(columnafter)

A screenshot of a computer code

Description automatically generated

#product data

#record count before

row <- nrow(product)

column <- ncol(product)

# Check for missing values

missing\_values <- colSums(is.na(product))

cat("Missing values:\n")

print(missing\_values)

# Check data types

data\_types <- sapply(product, class)

cat("Data Types:\n")

print(data\_types)

# Check for duplicates

duplicates <- anyDuplicated(product)

cat("Duplicates:\n")

print(duplicates)

#record count after

rowafter <- nrow(product)

columnafter <- ncol(product)

cat("Number of rows before:\n")

print(row)

cat("Number of columns before:\n")

print(column)

cat("Number of rows after:\n")

print(rowafter)

cat("Number of columns after:\n")

print(columnafter)

A screenshot of a computer code

Description automatically generated

#product\_level data

#record count before

row <- nrow(product\_level)

column <- ncol(product\_level)

# Check for missing values

missing\_values <- colSums(is.na(product\_level))

cat("Missing values:\n")

print(missing\_values)

# Check data types

data\_types <- sapply(product\_level, class)

cat("Data Types:\n")

print(data\_types)

# Check for duplicates

duplicates <- anyDuplicated(product\_level)

cat("Duplicates:\n")

print(duplicates)

#record count after

rowafter <- nrow(product\_level)

columnafter <- ncol(product\_level)

cat("Number of rows before:\n")

print(row)

cat("Number of columns before:\n")

print(column)

cat("Number of rows after:\n")

print(rowafter)

cat("Number of columns after:\n")

print(columnafter)

A screenshot of a computer code

Description automatically generated

#price\_list data

#record count before

row <- nrow(price\_list)

column <- ncol(price\_list)

# Check for missing values

missing\_values <- colSums(is.na(price\_list))

cat("Missing values:\n")

print(missing\_values)

# Check data types

data\_types <- sapply(price\_list, class)

cat("Data Types:\n")

print(data\_types)

# Convert the 'date\_column' from character to date YYYY-MM-DD

price\_list$Start\_Date <- as.Date(price\_list$Start\_Date, format = "%d%b%Y")

price\_list$End\_Date <- as.Date(price\_list$End\_Date, format = "%d%b%Y")

#convert price from character to integer

# Remove dollar sign and commas, then convert to numeric

price\_list$Unit\_Cost\_Price <- as.numeric(gsub("[^0-9.]", "", price\_list$Unit\_Cost\_Price))

price\_list$Unit\_Sales\_Price <- as.numeric(gsub("[^0-9.]", "", price\_list$Unit\_Sales\_Price))

# Check data types after transformation

data\_types <- sapply(price\_list, class)

cat("Data Types:\n")

print(data\_types)

# Check for duplicates

duplicates <- anyDuplicated(price\_list)

cat("Duplicates:\n")

print(duplicates)

#record count after

rowafter <- nrow(price\_list)

columnafter <- ncol(price\_list)

cat("Number of rows before:\n")

print(row)

cat("Number of columns before:\n")

print(column)

cat("Number of rows after:\n")

print(rowafter)

cat("Number of columns after:\n")

print(columnafter)

A screenshot of a computer code

Description automatically generated

#discount data

#record count before

row <- nrow(discount)

column <- ncol(discount)

# Check for missing values

missing\_values <- colSums(is.na(discount))

cat("Missing values:\n")

print(missing\_values)

# Check data types

data\_types <- sapply(discount, class)

cat("Data Types:\n")

print(data\_types)

# Convert the 'date\_column' from character to date YYYY-MM-DD

discount$Start\_Date <- as.Date(discount$Start\_Date, format = "%d%b%Y")

discount$End\_Date <- as.Date(discount$End\_Date, format = "%d%b%Y")

#convert price from character to integer

discount$Discount <- as.numeric(gsub("[^0-9.]", "", discount$Discount))

discount$Unit\_Sales\_Price <- as.numeric(gsub("[^0-9.]", "", discount$Unit\_Sales\_Price))

# Check data types after transformation

data\_types <- sapply(discount, class)

cat("Data Types:\n")

print(data\_types)

# Check for duplicates

duplicates <- anyDuplicated(discount)

cat("Duplicates:\n")

print(duplicates)

A screenshot of a computer code

Description automatically generated

#overview of the data using

install.packages("ggplot2")

library(ggplot2)

#dataset overview

ggplot(customer, aes(x = Country)) + geom\_bar(fill = "lightgreen", color = "black") + labs(title = "Distribution of Customers by Country", x = "Country", y = "Count")

ggplot(product, aes(x = Product\_Level)) + geom\_bar(fill = "coral", color = "black") + labs(title = "Distribution of Product Levels", x = "Product Level", y = "Count")

#merging the data

sales <- order %>% inner\_join(order\_item, by = "Order\_ID") %>% inner\_join(customer, by = "Customer\_ID") %>% inner\_join(product, by = "Product\_ID") %>% inner\_join(customer\_type, by = "Customer\_Type\_ID") %>% inner\_join(product, by = "Product\_ID")%>% select(Order\_ID, Customer\_ID, Order\_Date, Order\_Type, Product\_ID, Quantity, Total\_Retail\_Price, Customer\_Type, Country)

View(sales)

print(sales)

customers <- customer %>% inner\_join (country, by="Country") %>% select( Customer\_ID, Customer\_Name, Customer\_Type\_ID, Country, Country\_Name) %>% distinct()

View(customers)

print(customers)

products <- product %>% inner\_join (product\_level, by="Product\_Level") %>% select( Product\_ID, Product\_Name, Product\_Level, Product\_Level\_Name) %>% distinct()

View(products)

print(products)

customer\_type <- customer\_type %>% select( Customer\_Type\_ID, Customer\_Type, Customer\_Group\_ID, Customer\_Group) %>% distinct()

View(customer\_type)

print(customer\_type)

date <- order %>% select(Order\_Date) %>% distinct() %>% mutate( Year = year(Order\_Date), Month = month(Order\_Date),Day = day(Order\_Date))

View(date)

print(date)

# Filter rows with the latest date for each Product\_ID

price\_list <- price\_list %>% group\_by(Product\_ID) %>% filter(End\_Date == max(End\_Date))

View(price\_list)

product\_price <- price\_list %>% select( Product\_ID, Start\_Date, End\_Date, Unit\_Cost\_Price, Unit\_Sales\_Price, Factor) %>% distinct()

View(product\_price)

print(product\_price)

# Filter rows with the latest date for each Product\_ID

discount <- discount %>% group\_by(Product\_ID) %>% filter(End\_Date == max(End\_Date))

View(discount)

product\_discount <- discount %>% select( Product\_ID, Start\_Date, End\_Date, Unit\_Sales\_Price, Discount) %>% distinct()

View(product\_discount)

print(product\_discount)

write.csv(customers, "C:/Users/LabStudent-55-706949/admp\_project/customers.csv", row.names = FALSE)

write.csv(products, "C:/Users/LabStudent-55-706949/admp\_project/products.csv", row.names = FALSE)

write.csv(customer\_type, "C:/Users/LabStudent-55-706949/admp\_project/customer\_type.csv", row.names = FALSE)

write.csv(date, "C:/Users/LabStudent-55-706949/admp\_project/date.csv", row.names = FALSE)

write.csv(product\_price, "C:/Users/LabStudent-55-706949/admp\_project/product\_price.csv", row.names = FALSE)

write.csv(product\_discount, "C:/Users/LabStudent-55-706949/admp\_project/product\_discount.csv", row.names = FALSE)

write.csv(sales, "C:/Users/LabStudent-55-706949/admp\_project/sales.csv", row.names = FALSE)

install.packages("odbc")

install.packages("DBI")

install.packages("data.table")

# Load necessary data manipulation library

library(data.table)

library(DBI)

library(odbc)

**#Connecting R studio with Hive**

# Establish a connection to Hive

con <- DBI::dbConnect(odbc::odbc(), .connection\_string = "Driver={Cloudera ODBC Driver for Apache Hive};", Host = "sandbox-hdp.hortonworks.com", UID = "hive", PWD = "", Port = 10000)

database\_name <- "admp\_project"

# Create the database if it doesn't exist

dbExecute(con, paste("CREATE DATABASE IF NOT EXISTS", database\_name))

dbExecute(con, paste("USE", database\_name))

# Create the Hive table dim\_customer

dbExecute(con, paste("CREATE TABLE IF NOT EXISTS dim\_customer (Customer\_ID INT, Customer\_Name STRING, Customer\_Type\_ID INT, Country STRING, Country\_Name STRING)"))

# Define the path to the customer CSV file

csv\_path <- "C:/Users/LabStudent-55-706949/admp\_project/customers.csv"

# Load data from CSV to the Hive table dim\_customer

dbWriteTable(con, "dim\_customer", value = data.frame(read.csv(csv\_path)), overwrite = TRUE, append = FALSE)

# Query the first 10 rows from the customer table

query <- "SELECT \* FROM dim\_customer LIMIT 10"

result <- dbGetQuery(con, query)

print(result)

# Create the Hive table dim\_customer\_type

dbExecute(con, paste("CREATE TABLE IF NOT EXISTS dim\_customer\_type (Customer\_Type\_ID INT, Customer\_Type STRING, Customer\_Group\_ID INT, Customer\_Group STRING)"))

# Define the path to the customer\_type CSV file

csv\_path <- "C:/Users/LabStudent-55-706949/admp\_project/customer\_type.csv"

# Load data from CSV to the Hive table dim\_customer\_type

dbWriteTable(con, "dim\_customer\_type", value = data.frame(read.csv(csv\_path)), overwrite = TRUE, append = FALSE)

# Query the first 10 rows from the customer\_type table

query <- "SELECT \* FROM dim\_customer\_type LIMIT 10"

result <- dbGetQuery(con, query)

print(result)

# Create the Hive table dim\_discount

dbExecute(con, paste("CREATE TABLE IF NOT EXISTS dim\_discount (Product\_ID INT, Start\_Date DATE, End\_Date DATE, Unit\_Sales\_Price DECIMAL(10, 2), Discount DECIMAL(5, 2))"))

# Define the path to the discount CSV file

csv\_path <- "C:/Users/LabStudent-55-706949/admp\_project/product\_discount.csv"

# Load data from CSV to the Hive table dim\_discount

dbWriteTable(con, "dim\_discount", value = data.frame(read.csv(csv\_path)), overwrite = TRUE, append = FALSE)

# Query the first 10 rows from the discount table

query <- "SELECT \* FROM dim\_discount LIMIT 10"

result <- dbGetQuery(con, query)

print(result)

# Create the Hive table dim\_price

dbExecute(con, paste("CREATE TABLE IF NOT EXISTS dim\_price (Product\_ID BIGINT, Start\_Date DATE, End\_Date DATE, Unit\_Cost\_Price DECIMAL(10, 2), Unit\_Sales\_Price DECIMAL(10, 2), Factor INT)"))

# Define the path to the price CSV file

csv\_path <- "C:/Users/LabStudent-55-706949/admp\_project/product\_price.csv"

# Load data from CSV to the Hive table dim\_discount

dbWriteTable(con, "dim\_price", value = data.frame(read.csv(csv\_path)), overwrite = TRUE, append = FALSE)

# Query the first 10 rows from the price table

query <- "SELECT \* FROM dim\_price LIMIT 10"

result <- dbGetQuery(con, query)

print(result)

# Create the Hive table dim\_product

dbExecute(con, paste("CREATE TABLE IF NOT EXISTS dim\_product (Product\_ID BIGINT, Product\_Name STRING, Product\_Level STRING, Product\_Level\_Name STRING)"))

# Define the path to the product CSV file

csv\_path <- "C:/Users/LabStudent-55-706949/admp\_project/products.csv"

# Load data from CSV to the Hive table dim\_product

dbWriteTable(con, "dim\_product", value = data.frame(read.csv(csv\_path)), overwrite = TRUE, append = FALSE)

# Query the first 10 rows from the price table

query <- "SELECT \* FROM dim\_poduct LIMIT 10"

result <- dbGetQuery(con, query)

print(result)

# Create the Hive table dim\_date

dbExecute(con, paste("CREATE TABLE IF NOT EXISTS dim\_time\_date (Order\_Date DATE, Year INT, Month INT, Day INT)"))

# Define the path to the date CSV file

csv\_path <- "C:/Users/LabStudent-55-706949/admp\_project/date.csv"

# Load data from CSV to the Hive table dim\_date

dbWriteTable(con, "dim\_date", value = data.frame(read.csv(csv\_path)), overwrite = TRUE, append = FALSE)

# Query the first 10 rows from the date table

query <- "SELECT \* FROM dim\_date LIMIT 10"

result <- dbGetQuery(con, query)

print(result)

# Create the Hive table fact\_sales

dbExecute(con, paste("CREATE TABLE IF NOT EXISTS fact\_sales (Order\_ID INT, Customer\_ID INT, Order\_Date DATE, Order\_Type STRING, Product\_ID BIGINT, Quantity INT, Total\_Retail\_Price DECIMAL(15, 2), Customer\_Type STRING,Country STRING)"))

# Define the path to the sales CSV file

csv\_path <- "C:/Users/LabStudent-55-706949/admp\_project/sales.csv"

# Load data from CSV to the Hive table fact\_sales

dbWriteTable(con, "fact\_sales", value = data.frame(read.csv(csv\_path)), overwrite = TRUE, append = FALSE)

# Query the first 10 rows from the fact table

query <- "SELECT \* FROM fact\_sales LIMIT 10"

result <- dbGetQuery(con, query)

print(result)

A computer screen with text

Description automatically generated