A picture containing icon

Description automatically generated

**SRI LANKA INSTITUTE OF INFORMATION TECHNOLOGY**

**IT3021 – DATA WAREHOUSE AND BUSINESS INTELLIGENCE**

**ASSIGNMENT – 01**

**2022**

**SUBMITTED BY:**

**IT19991986 – RAEESUL ISLAM S.Z.**

**Y3S2.WE.DS.03**

Table of Contents

[**1.** **Data Selection and Introduction** 3](#_Toc103707292)

[**1.1.** **Introduction to Data set** 3](#_Toc103707293)

[**1.2.** **Features of Dataset** 3](#_Toc103707294)

[**1.3.** **Link to Access Dataset** 3](#_Toc103707295)

[**2.** **Preparations of Data Sources** 4](#_Toc103707296)

[**3.** **ER Diagram Developed Using the Sources** 7](#_Toc103707297)

[**4.** **Solution Architecture** 8](#_Toc103707298)

[**5.** **Data Warehouse Design and Implementation** 10](#_Toc103707299)

[**6.** **ETL Development** 12](#_Toc103707300)

[**6.1.** **Data Extraction from Source to Staging** 12](#_Toc103707301)

[**6.2.** **Data Profiling** 18](#_Toc103707302)

[**6.3.** **Transformation and Loading to Data Warehouse** 19](#_Toc103707303)

[**7.** **Appendix** 32](#_Toc103707304)

[**7.1.** **Staging Tables Creation** 32](#_Toc103707305)

[**7.2.** **Dimension Tables and Fact Table Creation** 34](#_Toc103707306)

[**7.3.** **Stored Procedures SQL Queries** 37](#_Toc103707307)

# **Data Selection and Introduction**

## **Introduction to Data set**

* Data Set Name : **Predicting Coupon Redemption**

The data set represents information about a collection of an anonymized Sales Transaction information from an Export Company to Predict Coupon Redemption, in 2012. The data set has been modified to develop a scenario that meets the requirement of the assignment.

Dataset contains 5 csv files with information about Customers, Train, Items, Campaign and Transaction. Modifications were done accordingly to the data set derived from the source This data set reflects combinations between customer transactions and promotion campaigns.

## **Features of Dataset**

* **Train Data**: containing the coupons offered to the given customers under the 18 campaigns.
* **Customer Data**: Customer Data containing  information for some customers.
* **Campaign Data**: Campaign information for each of the campaign.
* **Transaction Data**: Transaction data for all customers for duration of campaigns in the train data
* **Item Data**: Item information for each item sold by the retailer

## **Link to Access Dataset**

<https://www.kaggle.com/datasets/meghakanojia/predicting-coupon-redemption>

# **Preparations of Data Sources**

All the data sources are provided in csv format by the web site. In preparation of data sources, some changes have done for the source format (some columns were added, separated into another table) of the given files as converting into text files and importing csv files into a source database.

Ultimately, 2 main sources were created:

1. A database source : **PredictingCouponRedemptionSourceDB**
2. A text file to maintain customer details : **customer\_data.txt**

A database named **PredictingCouponRedemptionSourceDB** was created in SQL and the below mentioned files were imported:

* campaign\_data.csv
* train\_data.csv
* transaction\_data.csv
* category\_data.csv
* item\_data.csv
* district\_data.txt

A picture containing table

Description automatically generated

Figure 1. PredictingCouponRedemptionSourceDB

**PredictingCouponRedemptionStaging** database was created as a staging layer.

[**8.1. Staging Tables Creations**](#_Staging_Tables_Creation)

Graphical user interface, application

Description automatically generated with medium confidence

Figure 2. PredictingCouponRedemptionStaging

For data warehousing purposes a database named **PredictingCouponRedemptionDW** was created in SQL, including the dimensions and fact tables mentioned below.

* **DimCampaign**
* **DimTrain**
* **DimCategory**
* **DimItem**
* **DimCustomer**
* **DimDistrict**
* **DimDate**
* **FactTransaction**

[**8.2. Dimension Creations**](#_Dimension_Tables_and)

**Graphical user interface

Description automatically generated with medium confidence**

Figure 3. PredictingCouponRedemptionDW

# **ER Diagram Developed Using the Sources**

Diagram

Description automatically generated

Figure 4. ER Diagram constructed using sources

* The above diagram shows the connection between the entities in the data set.
* **Assumptions:**
* The particular transaction includes only a single item .
* One summary report(train) summarizes many customer transactions.
* There can be many campaign data sets in a single summary report.
* One customer can have many transactions.

# **Solution Architecture**

Diagram

Description automatically generated

Figure 5. Solution Architecture

**Data source:**

* Several data sources can be available when implementing a data warehouse solution. Sources are simply the origin of location of the used data. A data source may be a database, flat file, live measurements from physical device, scraped web data, etc. Here, a database source namely ‘**PredictingCouponRedemptionSourceDB**’ serves as the primary data source and a flat file source namely ‘**customer\_data.txt**’ serves as a secondary data source.

**ETL:**

* ETL is the abbreviation for the standard ‘Extraction-Transformation-Loading.’ It is the process of extracting data from one source, transform those data and finally load them to a destination. The extraction process followed here is a full extraction (Load all data in the source without filtering conditions). While performing the ETL process to load data to data warehouse, necessary steps like cleaning and aggregation were performed.

**Staging Layer:**

* This is an intermediate storage layer. This layer is added to prevent practical problems that could arise while transforming data to data warehouse. It is similar to the data source but contains all the data required for warehousing in a centralized location. A less amount of transformation is performed during the ETL process from source to staging.
* ‘**PredictingCouponRedemptionStaging**’ is the database created as a staging layer in the scenario.

**Data Warehouse:**

* Data warehouse is a large collection of business data. Aggregated and transactional data are stored here for analytical purposes. It is a core component of business intelligence. A database named ‘**PredictingCouponRedemptionDW**’ is created in SQL as the data warehouse layer.

# **Data Warehouse Design and Implementation**

Diagram

Description automatically generated

Figure 6. Snowflake Schema

The data warehouse design was implemented using the snowflake schema. It is an extension of star schema and consists of some dimensions that are normalized. According to the schema above, there are 7 dimensions and 1 fact table.

**Hierarchies:**

* **DimCategory** is applied as a hierarchical dimension of **DimItem** table.
* **DimCampaign** is applied as a hierarchical dimension of **DimTrain** table.
* **DimDistrict** is applied as a hierarchical dimension of **DimCustomer** table.

**Calculation :**

* Line Total is calculated in dbo.FactTransaction.LineTotal
* ([SellingPrice] \* [Quantity])
* Total Discount is calculated in dbo.FactTransaction.TotalDiscount
* (-([OtherDiscount] + [CouponDiscount]))
* Total Amount is calculated in dbo.FactTransaction.TotalAmount
* ([SellingPrice]\*[Quantity]+([CouponDiscount]+[OtherDiscount]))

**Assumption:**

* **Customer** dimension is considered as a **Slowly Changing Dimension(SCD).**

# **ETL Development**

## **Data Extraction from Source to Staging**

As the initial step, the data from sources were extracted to a staging layer. These data were then transformed and loaded to the staging tables. The data flow task was used to perform this process.

Source table and staging tables are as below:

|  |  |
| --- | --- |
| Source Table | Staging Table |
| Category | StgCategory |
| Item | StgItem |
| Campaign | StgCampaign |
| Train | StgTrain |
| District | StgDistrict |
| Transaction | StgTransaction |
|  |  |
| Customer\_data.csv | StgCustomer |

Control Flow:

Text

Description automatically generated with medium confidence

Figure 7. Control Flow Task for Staging

A screenshot of a computer

Description automatically generated with medium confidence A screenshot of a computer

Description automatically generated with low confidence

A screenshot of a computer

Description automatically generated with medium confidence A screenshot of a computer

Description automatically generated with low confidence

A screenshot of a computer

Description automatically generated with low confidence A screenshot of a computer

Description automatically generated with medium confidence

Figure 8. Staging from a Database Source to Database Destination

A screenshot of a computer

Description automatically generated with medium confidence

Figure 9. Staging from a Flat File Source to Database Destination

**Test Data Loaded from Source to Staging**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Scenario ID** | | | 1 | | | |
| **Test Case Description** | | | Transform test data from source to staging | | | |
| **Pre-Requisite** | | | Test loaded from source to staging table | | | |
| **ID** | **Action** | **SQL Queries** | **Expected Output** | **Actual Output** | **Result** | **Refer** |
| 1 | Data passed from Category Source to StgCategory Staging | Select \*  From StgCategory | Display selected rows | Displayed selected rows | PASS | Figure 1.1 |
| 2 | Data passed from Item Source to StgItem Staging | Select \*  From StgItem | Display selected rows | Displayed selected rows | PASS | Figure 1.2 |
| 3 | Data passed from Campaign Source to StgCampaign Staging | Select \*  From StgCampaign | Display selected rows | Displayed selected rows | PASS | Figure 1.3 |
| 4 | Data passed from Train Source to StgTrain Staging | Select \*  From StgTrain | Display selected rows | Displayed selected rows | PASS | Figure 1.4 |
| 5 | Data passed from District Source to StgDistrict Staging | Select \*  From StgDistrict | Display selected rows | Displayed selected rows | PASS | Figure 1.5 |
| 6 | Data passed from Customer Source to StgCustomer Staging | Select \*  From StgCustomer | Display selected rows | Displayed selected rows | PASS | Figure 1.6 |
| 7 | Data passed from Transaction Source to StgTransaction Staging | Select \*  From StgTransaction | Display selected rows | Displayed selected rows | PASS | Figure 1.7 |

Graphical user interface, text, application

Description automatically generated

Figure 1. 1. StgCategory

Graphical user interface, text, application, email

Description automatically generated

Figure 1. 2. StgItem

Graphical user interface, text, application

Description automatically generated

Figure 1. 3. StgCampaign

Graphical user interface, application

Description automatically generated

Figure 1. 4. StgTrain

Graphical user interface, application, Word

Description automatically generated

Figure 1. 5. StgDistrict

Graphical user interface, application, table

Description automatically generated

Figure 1. 6. StgCustomer

Graphical user interface, table

Description automatically generated

Figure 1. 7. StgTransaction

## **Data Profiling**

Data profiling is the process of reviewing data to understand the structure, content and inter relationships. It uncovers the issues related to data quality that can be corrected in ETL process.

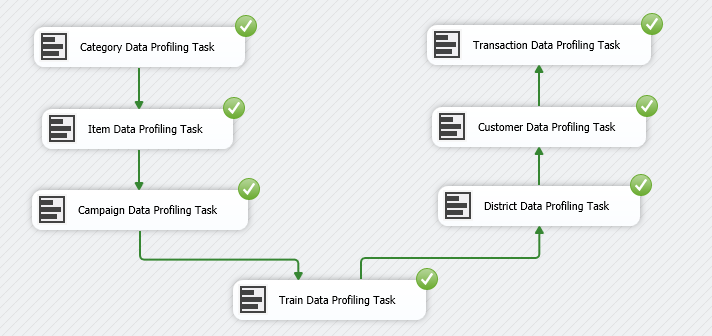


Figure 10. Data Profiling Task Flow

## **Transformation and Loading to Data Warehouse**

When loading data from staged layer to Data Warehouse, the order of execution is very important. The reason for this is that the dimensions and facts contain dependencies with each other:

The order of execution is shown below in the control flow task of ETL:

Graphical user interface, text, chat or text message

Description automatically generated

Figure 11. Control Flow Task of Data Warehouse Transformation and Loading

1. **Loading Category Data to DimCategory**

The **Category** dimension has no dependencies with any other dimensions; therefore, it is loaded first.

**A screenshot of a computer

Description automatically generated with medium confidence**

Figure 12. Data Flow Task of Category Dimension Transformation and Loading

1. **Loading Item Data to DimItem**

**Item** Data can be loaded next since they contain reference to the **Category**.

**Diagram

Description automatically generated**

Figure 13. Data Flow Task of Item Dimension Transformation and Loading

**DimItem** contains a reference to **DimCategory**. In order to get the **Category** surrogate key to **Item** dimension, data was extracted from both dimensions and sorted based on **Category ID**. Then they were merged to load into **DimItem**.

Some **Items** may not have a **Category** **ID**; thus, the Merge join was done using left outer join.

1. **Loading Campaign Data to DimCampaign**

The **Campaign** dimension also has no dependencies with any other dimensions; therefore, it is loaded next.

**A picture containing graphical user interface

Description automatically generated**

Figure 14. Data Flow Task of Campaign Dimension Transformation and Loading

1. **Loading Train Data to DimTrain**

**Train** Data can be loaded next since they contain reference to the **Campaign**.

**Diagram

Description automatically generated**

Figure 15. Data Flow Task of Train Dimension Transformation and Loading

**Train** dimension contains a reference to **Campaign** dimension. In order to get the surrogate keys of **Campaign** to **Train** dimension a lookup process was performed.

1. **Loading District Data to DimDistrict**

The **District** dimension also has no dependencies with any other dimensions; therefore, it is loaded next.

A picture containing diagram

Description automatically generated

Figure 16. Data Flow Task of District Dimension Transformation and Loading

1. **Load Customer Data to DimCustomer**

Next, the Customer Dimension loaded

Diagram

Description automatically generated

Diagram

Description automatically generated

Diagram

Description automatically generated

Figure 17. Data Flow Task of Customer Dimension Transformation and Loading

**Customer** dimension contains a reference to **District** dimension. In order to get the surrogate keys of **District** to **Customer** dimension a lookup process was performed.

**DimCustomer** is the **Slowly Changing Dimension** in this dimensional modeling. In order to load data to Dimension table, the Slowly Changing Dimensions (historical) have two specific columns as StartDate & EndDate to ensure that the data is valid at the moment.

Slowly Changing Dimension wizard let the developer to select the Dimension table, Business keys of the dimension and what would be the slowly changing attributes.

Initially data cleansing is done in order to remove null values from the data source table. Based on data profiling result, null values from Gender column, and Marital Status column were removed.

Therefore, following attributes were set as **changing attributes** and **historical attributes**.

* Marital Status – Changing Attribute
* Phone No – Changing Attribute
* Age Range – Historical Attribute
* DistrictKey – Historical Attribute

After performing these tasks, the **Customer** dimension was loaded.

1. **Load Transaction Data to FactTransaction**

Finally, the **Fact Transaction** is loaded as it contains references to many other dimensions.

Diagram

Description automatically generated

Diagram

Description automatically generated

A picture containing text

Description automatically generated

Figure 18. Data Flow Task of Transaction Fact Table Transformation and Loading

The **Fact Transaction** contains references to **Customer**, **Train, Item**, and **Date**. In order to get the surrogate keys as references, lookup processes were carried out for all references. Insert date and modified date are derived columns. Finally, the fact table was loaded to its destination.

For the Accumulating Fact Table need to update AccmTxnCreateTime, AccmTxnCompleteTime, and TxnProcessTimeHour columns.

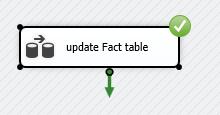
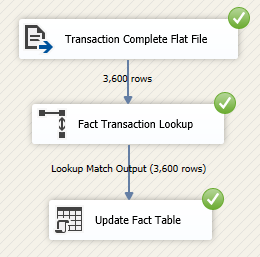
 

Figure 19. Update Fact Table Control Flow Task

* Dimensions like Category, Item, Campaign, and Train does not maintain history. Therefore, in order to maintain the latest record, stored procedures were created.

[**8.3. Stored Procedure Queries**](#_Stored_Procedures_SQL)

**Test Data Loaded from Staging to Datawarehouse**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Scenario ID** | | | 2 | | | |
| **Test Case Description** | | | Transform staging data to dimension tables | | | |
| **Pre-Requisite** | | | Test loaded from staging to dimension table | | | |
| **ID** | **Action** | **SQL Queries** | **Expected Output** | **Actual Output** | **Result** | **Refer** |
| 1 | Data passed from StgCategory to DimCategory | Select \*  From DimCategory | Display selected rows | Displayed selected rows | PASS | Figure 2.1. |
| 2 | Data passed from StgItem to DimItem | Select \*  From DimItem | Display selected rows | Displayed selected rows | PASS | Figure 2.2. |
| 3 | Data passed from StgCampaign to DimCampaign | Select \*  From DimCampaign | Display selected rows | Displayed selected rows | PASS | Figure 2.3. |
| 4 | Data passed from StgTrain to DimTrain | Select \*  From DimTrain | Display selected rows | Displayed selected rows | PASS | Figure 2.4. |
| 5 | Data passed from StgDistrict to DimDistrict | Select \*  From DimDistrict | Display selected rows | Displayed selected rows | PASS | Figure 2.5. |
| 6 | Data passed from StgCustomer to DimCustomer | Select \*  From DimCustomer | Display selected rows | Displayed selected rows | PASS | Figure 2.6. |
| 7 | Data passed from StgTransaction to FactTransaction | Select \*  From FactTransaction | Display selected rows | Displayed selected rows | PASS | Figure 2.6. |

Graphical user interface, table

Description automatically generated with medium confidence

Figure 2. 1. DimCategory

Table

Description automatically generated

Figure 2. 2. DimItem

Graphical user interface, application, table

Description automatically generated

Figure 2. 3. DimCampaign

Table

Description automatically generated

Figure 2. 4. DimTrain

Graphical user interface, application, table

Description automatically generated

Figure 2. 5. DimDistrict

Graphical user interface, application, table

Description automatically generated

Figure 2. 6. DimCustomer

Table

Description automatically generated

Figure 2. 7. FactTransaction

# **Appendix**

## **Staging Tables Creation**

**--- Create StgCategory Table ---**

CREATE TABLE [dbo].[StgCategory](

[CategoryID] [int] NULL,

[CategoryName] [varchar](50) NULL,

[SrcModifiedDate] [datetime] NULL

) ON [PRIMARY]

**--- Create StgItem Table ---**

CREATE TABLE [dbo].[StgItem](

[ItemID] [int] NULL,

[Brand] [int] NULL,

[BrandType] [varchar](50) NULL,

[CategoryID] [int] NULL

) ON [PRIMARY]

**--- Create StgCampaign Table ---**

CREATE TABLE [dbo].[StgCampaign](

[CampaignID] [int] NULL,

[CampaignType] [varchar](1) NULL,

[StartDate] [datetime] NULL,

[EndDate] [datetime] NULL

) ON [PRIMARY]

**--- Create StgTrain Table ---**

CREATE TABLE [dbo].[StgTrain](

[TrainID] [int] NULL,

[CampaignID] [int] NULL,

[CouponID] [int] NULL,

[RedemptionStatus] [int] NULL

) ON [PRIMARY]

**--- Create StgDistrict Table ---**

CREATE TABLE [dbo].[StgDistrict](

[DistrictID] [int] NULL,

[DistrictName] [varchar](50) NULL,

[Region] [varchar](50) NULL,

[State] [varchar](50) NULL

) ON [PRIMARY]

**--- Create StgCustomer Table ---**

CREATE TABLE [dbo].[StgCustomer](

[CustomerID] [int] NULL,

[Name] [varchar](50) NULL,

[Gender] [varchar](1) NULL,

[AgeRange] [varchar](50) NULL,

[MaritalStatus] [varchar](50) NULL,

[PhoneNo] [varchar](25) NULL,

[Rented] [int] NULL,

[FamilySize] [int] NULL,

[NoOfChildren] [int] NULL,

[IncomeBracket] [int] NULL,

[DistrictID] [int] NULL

) ON [PRIMARY]

**--- Create StgTransaction Table ---**

CREATE TABLE [dbo].[StgTransaction](

[TransactionID] [int] NULL,

[CustomerID] [int] NULL,

[ItemID] [int] NULL,

[TrainID] [int] NULL,

[Date] [datetime] NULL,

[Quantity] [int] NULL,

[SellingPrice] [money] NULL,

[OtherDiscount] [money] NULL,

[CouponDiscount] [money] NULL

) ON [PRIMARY]

## **Dimension Tables and Fact Table Creation**

**--- Create DimCategory Table ---**

CREATE TABLE [dbo].[DimCategory](

[CategorySK] [int] IDENTITY(1,1) NOT NULL,

[CategoryAlternateID] [int] NULL,

[CategoryName] [varchar](50) NULL,

[SrcModifiedDate] [datetime] NULL,

[InsertDate] [datetime] NULL,

[ModifiedDate] [datetime] NULL,

CONSTRAINT [PK\_DimCategory] PRIMARY KEY CLUSTERED

(

[CategorySK] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

**--- Create DimItem Table ---**

CREATE TABLE [dbo].[DimItem](

[ItemSK] [int] IDENTITY(1,1) NOT NULL,

[ItemAlternateID] [int] NULL,

[Brand] [int] NULL,

[BrandType] [varchar](50) NULL,

[CategoryKey] [int] NULL,

[InsertDate] [datetime] NULL,

[ModifiedDate] [datetime] NULL,

CONSTRAINT [PK\_DimItem] PRIMARY KEY CLUSTERED

(

[ItemSK] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

**--- Create DimCampaign Table ---**

CREATE TABLE [dbo].[DimCampaign](

[CampaignSK] [int] IDENTITY(1,1) NOT NULL,

[CampaignAlternateID] [int] NULL,

[CampaignType] [varchar](1) NULL,

[StartDate] [datetime] NULL,

[EndDate] [datetime] NULL,

[InsertDate] [datetime] NULL,

[ModifiedDate] [datetime] NULL,

CONSTRAINT [PK\_DimCampaign] PRIMARY KEY CLUSTERED

(

[CampaignSK] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

**--- Create DimTrain Table ---**

CREATE TABLE [dbo].[DimTrain](

[TrainSK] [int] IDENTITY(1,1) NOT NULL,

[TrainAlternateID] [int] NULL,

[CouponID] [int] NULL,

[CampaignKey] [int] NULL,

[RedemptionStatus] [int] NULL,

[InsertDate] [datetime] NULL,

[ModifiedDate] [datetime] NULL,

CONSTRAINT [PK\_DimTrain] PRIMARY KEY CLUSTERED

(

[TrainSK] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

**--- Create DimDistrict Table ---**

CREATE TABLE [dbo].[DimDistrict](

[DistrictSK] [int] IDENTITY(1,1) NOT NULL,

[DistrictAlternateID] [int] NULL,

[City] [varchar](50) NULL,

[Region] [varchar](50) NULL,

[State] [varchar](50) NULL,

[InsertDate] [datetime] NULL,

[ModifiedDate] [datetime] NULL,

CONSTRAINT [PK\_DimDistrict] PRIMARY KEY CLUSTERED

(

[DistrictSK] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

**--- Create DimCustomer Table ---**

CREATE TABLE [dbo].[DimCustomer](

[CustomerSK] [int] IDENTITY(1,1) NOT NULL,

[CustomerAlternateID] [int] NULL,

[Name] [varchar](50) NULL,

[Gender] [varchar](1) NULL,

[AgeRange] [varchar](50) NULL,

[MaritalStatus] [varchar](50) NULL,

[PhoneNo] [varchar](30) NULL,

[Rented] [int] NULL,

[FamilySize] [int] NULL,

[NoOfChildren] [int] NULL,

[IncomeBracket] [int] NULL,

[DistrictKey] [int] NULL,

[StartDate] [datetime] NULL,

[EndDate] [datetime] NULL,

[InsertDate] [datetime] NULL,

[ModifiedDate] [datetime] NULL,

CONSTRAINT [PK\_DimCustomer] PRIMARY KEY CLUSTERED

(

[CustomerSK] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

**--- Create FactTransaction Table ---**

CREATE TABLE [dbo].[FactTransaction](

[TransactionID] [int] NULL,

[CustomerKey] [int] NULL,

[ItemKey] [int] NULL,

[TrainKey] [int] NULL,

[DateKey] [int] NULL,

[Quantity] [int] NULL,

[SellingPrice] [money] NULL,

[OtherDiscount] [money] NULL,

[CouponDiscount] [money] NULL,

[LineTotal] AS ([SellingPrice]\*[Quantity]),

[TotalDiscount] AS ( -([OtherDiscount]+[CouponDiscount])),

[TotalAmount] AS ([SellingPrice]\*[Quantity]+([CouponDiscount]+[OtherDiscount])),

[InsertDate] [datetime] NULL,

[ModifiedDate] [datetime] NULL,

[AccmTxnCompleteTime] [datetime] NULL,

[AccmTxnCreateTime] [datetime] NULL,

[TxnProcessTimeHours] AS (datediff(hour,[AccmTxnCompleteTime],[AccmTxnCreateTime]))

) ON [PRIMARY]

## **Stored Procedures SQL Queries**

--- Stored Procedure for DimCategory ---

CREATE PROCEDURE dbo.UpdateDimCategory

@CategoryID int,

@CategoryName varchar(50),

@ModifiedDate datetime

AS

BEGIN

if not exists (select CategorySK

from dbo.DimCategory

where CategoryAlternateID= @CategoryID)

BEGIN

insert into dbo.DimCategory

(CategoryAlternateID, CategoryName, SrcModifiedDate, InsertDate, ModifiedDate)

values

(@CategoryID, @CategoryName, @ModifiedDate, GETDATE(), GETDATE())

END;

if exists (select CategorySK

from dbo.DimCategory

where CategoryAlternateID = @CategoryID)

BEGIN

update dbo.DimCategory

set CategoryName = @CategoryName,

SrcModifiedDate = @ModifiedDate,

ModifiedDate = GETDATE()

where CategoryAlternateID = @CategoryID

END;

END;

--- Stored Procedure for DimItem ---

CREATE PROCEDURE dbo.UpdateDimItem

@ItemID int,

@brand int,

@brandtype varchar(50),

@categoryKey int

AS

BEGIN

if not exists (select ItemSK

from dbo.DimItem

where ItemAlternateID = @ItemID)

BEGIN

insert into dbo.DimItem

(ItemAlternateID, Brand, BrandType, CategoryKey, InsertDate, ModifiedDate)

values

(@ItemID, @brand, @brandtype, @categoryKey, GETDATE(), GETDATE())

END;

if exists (select ItemSK

from dbo.DimItem

where ItemAlternateID = @ItemID)

BEGIN

update dbo.DimItem

set Brand = @brand,

BrandType = @brandtype,

CategoryKey = @categoryKey,

ModifiedDate = GETDATE()

where ItemAlternateID = @ItemID

END;

END;

--- Stored Procedure for DimCampaign ---

CREATE PROCEDURE dbo.UpdateDimCampaign

@CampaignID int,

@type varchar(1),

@startdate datetime,

@enddate datetime

AS

BEGIN

if not exists (select CampaignSK

from dbo.DimCampaign

where CampaignAlternateID = @CampaignID)

BEGIN

insert into dbo.DimCampaign

(CampaignAlternateID , CampaignType, StartDate, EndDate, InsertDate, ModifiedDate)

values

(@CampaignID, @type, @startdate, @enddate, GETDATE(), GETDATE())

END;

if exists (select CampaignSK

from dbo.DimCampaign

where CampaignAlternateID = @CampaignID)

BEGIN

update dbo.DimCampaign

set CampaignType = @type,

StartDate = @startdate,

EndDate = @enddate,

ModifiedDate = GETDATE()

where CampaignAlternateID = @CampaignID

END;

END;

--- Stored Procedure for DimTrain ---

CREATE PROCEDURE dbo.UpdateDimTrain

@trainID int,

@couponID int,

@campaignKey int,

@redemption int

AS

BEGIN

if not exists (select TrainSK

from dbo.DimTrain

where TrainAlternateID = @trainID)

BEGIN

insert into dbo.DimTrain

(TrainAlternateID, CouponID, CampaignKey, RedemptionStatus, InsertDate, ModifiedDate)

values

(@trainID, @couponID, @campaignKey, @redemption, GETDATE(), GETDATE())

END;

if exists (select TrainSK

from dbo.DimTrain

where TrainAlternateID = @trainID)

BEGIN

update dbo.DimTrain

set CouponID = @couponID,

CampaignKey = @campaignKey,

RedemptionStatus = @redemption,

ModifiedDate = GETDATE()

where TrainAlternateID = @trainID

END;

END;

--- Stored Procedure for DimDistrict ---

CREATE PROCEDURE dbo.UpdateDimDistrict

@districtID int,

@DistrictName varchar(50),

@region varchar(50),

@state varchar(50)

AS BEGIN

if not exists (

select DistrictSK

from dbo.DimDistrict

where DistrictAlternateID = @districtID)

BEGIN

insert into dbo.DimDistrict

(DistrictAlternateID, DistrictName, Region, State, InsertDate, ModifiedDate)

values

(@districtID, @DistrictName, @region, @state, GETDATE(), GETDATE())

END;

if exists (

select DistrictSK

from dbo.DimDistrict

where DistrictAlternateID = @districtID)

BEGIN

update dbo.DimDistrict

set DistrictName = @DistrictName,

Region = @region,

State = @state,

ModifiedDate = GETDATE()

where DistrictAlternateID = @districtID

END;

END;

--- Stored Procedure for FactTransaction ---

CREATE PROCEDURE dbo.UpdateFactTransaction

@tranID int,

@complete datetime,

@create datetime

AS

BEGIN

if exists (select TransactionID

from dbo.FactTransaction

where TransactionID = @tranID)

BEGIN

update dbo.FactTransaction

set AccmTxnCompleteTime = @complete,

AccmTxnCreateTime = @create

where TransactionID = @tranID

END;

END;