

**SRI LANKA INSTITUE OF INFORMATION TECHNOLOGY (SLIIT)**

**IT3021 – DATA WAREHOUSE AND BUSINESS INTELLIGENCE**

ASSIGNMENT – 01

**S.Z. RAEESUL ISLAM**

**IT19991986**

Table Contents

[**Figures Contents** 3](#_Toc94028539)

[**1.** **Dataset Selection and Introduction** 4](#_Toc94028540)

[**1.1.** **Introduction to Dataset** 4](#_Toc94028541)

[**1.2.** **Features of the dataset** 4](#_Toc94028542)

[**1.3.** **Link to Access Dataset** 4](#_Toc94028543)

[**2.** **Preparation of Data Source** 5](#_Toc94028544)

[**3.** **ER Diagram** 6](#_Toc94028545)

[**4.** **Solution Architecture** 7](#_Toc94028546)

[**5.** **Data Warehouse Design and Implementation.** 8](#_Toc94028547)

[**6.** **ETL Development** 9](#_Toc94028548)

[**6.1.** **Data Extraction from Source to Staging** 9](#_Toc94028549)

[**6.2.** **Data Profiling** 11](#_Toc94028550)

[**6.3.** **Transform and Load Data to Datawarehouse** 12](#_Toc94028551)

[**7.** **Appendix** 19](#_Toc94028552)

[**7.1.** **Dimension and Fact Tables Creation** 19](#_Toc94028553)

[**7.2.** **Stored Procedure SQL Queries** 23](#_Toc94028554)

# **Figures Contents**

[Figure 1. ER Diagram 6](#_Toc94028556)

[Figure 2. Solution Architecture 7](#_Toc94028557)

[Figure 3. SnowFlake Schema 8](#_Toc94028558)

[Figure 4. Control Flow Task for Staging 9](#_Toc94028559)

[Figure 5. Staging from a database source to database destination 10](#_Toc94028560)

[Figure 6. Staging from a Flat File source to database destination 10](#_Toc94028561)

[Figure 7. Data Profiling Task Flow 11](#_Toc94028562)

[Figure 8. Control flow task of Data warehouse Transformation and Loading 12](#_Toc94028563)

[Figure 9. Data flow task of District Dimension Transformation and Loading 12](#_Toc94028564)

[Figure 10. Data flow task of Account Dimension transformation and loading 13](#_Toc94028565)

[Figure 11. Data flow task of Loan Dimension transformation and loading 14](#_Toc94028566)

[Figure 12. Data flow task of Permanent Order Dimension Transformation and Loading 14](#_Toc94028567)

[Figure 13. Data Flow task of Client Dimension Transformation and Loading 15](#_Toc94028568)

[Figure 14. Data flow task of Disposition Dimension Transformation and Loading 16](#_Toc94028569)

[Figure 15. Data flow task of Card Dimension Transformation and Loading 17](#_Toc94028570)

[Figure 16. Data Flow task of Transaction Fact Transformation and Loading 18](#_Toc94028571)

# **Dataset Selection and Introduction**

## **Introduction to Dataset**

The dataset represents information about a collection of an anonymized financial information from a bank in **Czech Republic** to mine and analyze this bank data in order to extrapolate from it the type of customer who makes a good candidate for a credit card. The dataset has been modified to develop a scenario that meets the requirements of the assignment. The transactional details made on an Account, the details of Clients involved in an Account, the Account details, Loan, and Permanent Order details are some of the information that are available through the dataset.

## **Features of the dataset**

* **Account:** each record describes static characteristics of an account. (4500 records in the file)
* **Client:** each record describes characteristics of a client. (5369 records in the file)
* **Disposition:** each record relates together a **Client** with an **Account**. This relation describes the rights of clients to operate accounts. (5369 records in the file)
* **Permanent Order:** each record describes characteristics of a payment orders. (6471 records in the file)
* **Credit Card:** each record describes a credit card issued to an account. (892 records in the file)
* **Loan:** each record describes a loan granted for a given account. (682 records in the file)
* **Transaction:** each record describes one transaction on an account. (1056320 records in the file, but get only 50000)
* **District:** each record describes demographic characteristics of a district. (77 records in the file)
* Each account has both static characteristics (date of creation, address of the branch) given in relation "**Account**" and dynamic characteristics (payments debited or credited, balances) given in relations "**Permanent** **Order**" and "**Transaction**".
* Relation "**Client**" describes characteristics of persons who can manipulate with the accounts. One client can have more accounts, more clients can manipulate with single account; clients and accounts are related together in relation "**Disposition**".
* Relations "**Loan**" and "**Credit** **Card**" describe some services which the bank offers to its clients; more credit cards can be issued to an account, at most one loan can be granted for an account.
* Relation "**District**" gives some publicly available information about the districts. Additional information about the clients can be deduced from this.

## **Link to Access Dataset**

<https://data.world/lpetrocelli/retail-banking-demo-data>

# **Preparation of Data Source**

The collected dataset was analyzed and modified according to the requirements of the project. The primary link contains 8 csv files. Some additional information was added to few relations in order to achieve the missing requirements.

Ultimately, 2 main sources were created:

1. **Database source**: CreditCardAnalysis\_SourceDB
2. **Text file to maintain district data**: Client.csv, ClientAddress.csv

**Assumption**:

No client address details were provided in the data set. The bank might require the address details of the clients for contact purposes. Therefore, another CSV file was created to keep track of these details. Further, it was required to meet the requirement of the assignment

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

* Account.csv
* Transaction.csv
* District.csv
* CreditCard.csv
* Disposition.csv
* Loan.csv
* PermenantOrder.csv

**CreditCardAnalysis\_StagingDB** database was created as a Staging layer.

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

* DimClient
* DimAccount
* DimDisposition
* DimDistrict
* DimLoan
* DimCreditCard
* DimPermenantOrder
* DimDate
* FactTransaction

[7.2. Dimension Creation Queries](#_Dimension_and_Fact)

# **ER Diagram**

Diagram, schematic

Description automatically generated

Figure 1. ER Diagram

# **Solution Architecture**

Diagram

Description automatically generated

Figure 2. Solution Architecture

**Date Sources**:

* 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 ‘**CreditCardAnalysis\_SourceDB**’ serves as the primary data source and a flat file source namely ‘**District.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.
* ‘**CreditCardAnalysis\_StagingDB**’ 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 ‘**CreditCardAnalysis\_DW**’ is created in SQL as the data warehouse layer.

# **Data Warehouse Design and Implementation.**

Diagram

Description automatically generated

Figure 3. SnowFlake Schema

* The data warehouse design was implemented using the **Snowflake** **schema**. Snowflake schema is an extension of Star schema and consists of some dimensions that are normalized. According to the schema above, there are **8** **Dimensions** and **1 Fact table**. The **District** dimension contains hierarchical data. The schema was designed considering the transactions per date as the level of grain.
* **Assumption**:

**Client** 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:

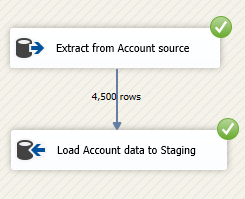
|  |  |
| --- | --- |
| Source Tables | Staging Tables |
| Account | StgAccount |
| District | StgDistrict |
| Disposition | StgDisposition |
| Card | StgCard |
| PermanentOrder | StgPermanentOrder |
| Loan | StgLoan |
| AccountTransaction | StgAccountTransaction |
|  |  |
| Client.csv | StgClient |
| ClientAddress.csv | StgClientAddress |

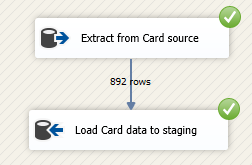
Control Flow Task:

Diagram

Description automatically generated

Figure 4. Control Flow Task for Staging

 A screenshot of a computer

Description automatically generated with medium confidence 

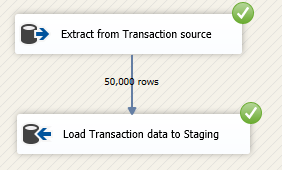
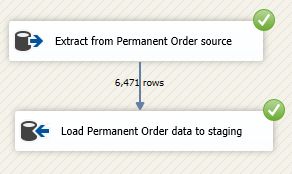
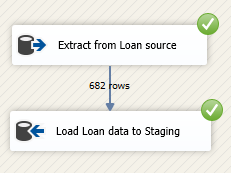
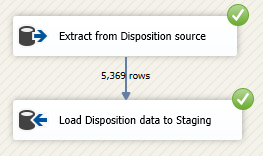


Figure 5. Staging from a database source to database destination

A screenshot of a computer

Description automatically generated with medium confidence A screenshot of a computer

Description automatically generated with medium confidence

Figure 6. Staging from a Flat File source to database destination

## **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.

Diagram

Description automatically generated

Figure 7. Data Profiling Task Flow

## **Transform and Load Data to Datawarehouse**

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:

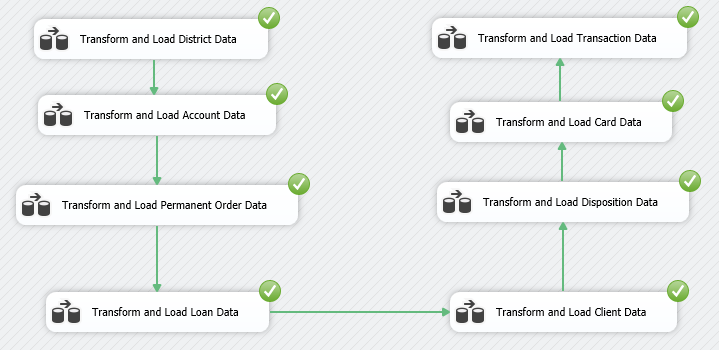


Figure 8. Control flow task of Data warehouse Transformation and Loading

1. **Loading District Data to DimDistrict**

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

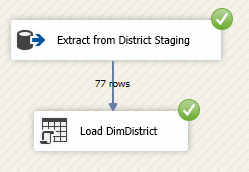


Figure 9. Data flow task of District Dimension Transformation and Loading

1. **Loading Account Data to DimAccount**

Both **Account** and **Client** Data can be loaded next since they contain reference to the **District**.

Here, the **Account** dimension is loaded as the next dimension.

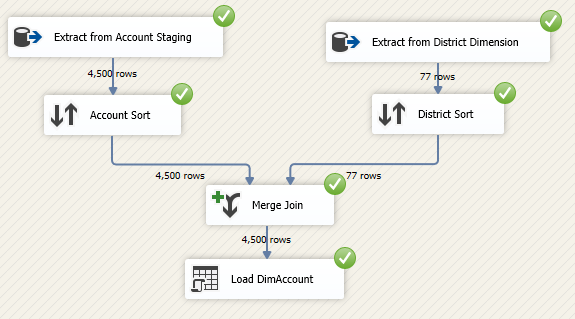


Figure 10. Data flow task of Account Dimension transformation and loading

**DimAccount** contains a reference to **DimDistrict**. In order to get the **District** surrogate key to account dimension, data was extracted from both dimensions and sorted based on **District ID**. Then they were merged to load into **DimAccount**.

Some Accounts may not have a District ID; thus, the Merge join was done using left outer join.

1. **Loading Loan Data to DimLoan**

**Loan** and **Permanent Order** contain reference to **Account**;

Therefore, **Loan** dimension is loaded following **Account**, followed by the **Permanent Order** dimension.

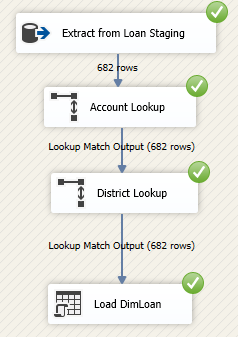


Figure 11. Data flow task of Loan Dimension transformation and loading

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

1. **Loading Permanent Data to DimPermanentOrder**

Permanent Order Dimension loaded following Loan

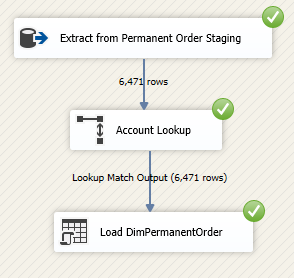


Figure 12. Data flow task of Permanent Order Dimension Transformation and Loading

**Permanent** **Order** dimension contains a reference to **Account** dimension. In order to get the surrogate key of **Account** to **Permanent** **Order** dimension a lookup process was performed.

1. **Loading Client Data to DimClient**

Next, the **Client** dimension is loaded.

Diagram

Description automatically generated

Diagram

Description automatically generated

Diagram

Description automatically generated

Figure 13. Data Flow task of Client Dimension Transformation and Loading

**DimClient** contains the **Client Address** details as well as other **Client** details together. Therefore, the Client Data was extracted from both **Client staging** and **Client Address Staging**, sorted, and merged. **Client** dimension contains a reference to the **District**. In order to get the **District** surrogate key a lookup process was created. Next, the null values of the address\_2 fields were replaced.

**Client** is a **Slowly Changing Dimension**.

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

* Phone – changing attribute
* Address\_1 – historical attribute
* Address\_2 – historical attribute
* Zipcode – historical attribute
* DistrickSK – historical attribute

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

1. **Loading Disposition Data to DimDisposition**

**Disposition** dimension contains reference to both **Client** and **Account** dimensions. On that basis its loaded next after the loading of both **Client** and **Account**.

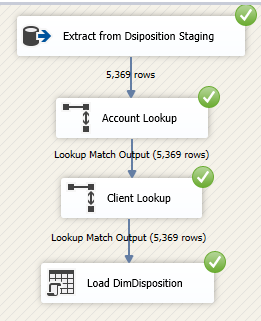


Figure 14. Data flow task of Disposition Dimension Transformation and Loading

**Disposition** dimension contains a reference to **Account** and **Client** dimension. In order to get the surrogate keys of **Account** and **Client** to **Disposition** dimension a lookup process was performed.

1. **Loading Card Data to DimCard**

**Card** dimension contains reference to the **Disposition**. Thus, its loaded after **Disposition**.

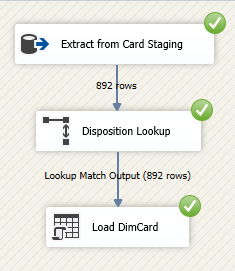


Figure 15. Data flow task of Card Dimension Transformation and Loading

**Card** dimension contains a reference to **Disposition** dimension. In order to get the surrogate key of **Disposition** to **Card** dimension a lookup process was performed.

1. **Loading Transaction Data to FactTransaction**

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

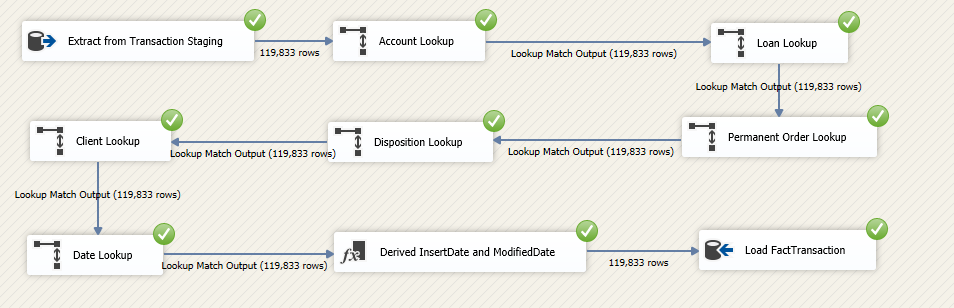


Figure 16. Data Flow task of Transaction Fact Transformation and Loading

The **Fact** **Transaction** contains references to **Account**, **Loan**, **Permanent** **Order**, **Disposition**, **Client**, 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.

* Dimensions like Account, Loan, Permanent Order, District, Disposition and Card does not maintain history. Therefore, in order to maintain the latest record, stored procedures were created. [7.3 Stored Procedure Queries](#_Stored_Procedure_SQL)

# **Appendix**

## **Dimension and Fact Tables Creation**

-- Create DimAccount Table --

CREATE TABLE [dbo].[DimAccount](

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

[AlternativeAccountID] [varchar](10) NOT NULL,

[DistrictSK] [int] NULL,

[Frequency] [varchar](50) NULL,

[StartedDate] [date] NULL,

[InsertDate] [datetime] NULL,

[ModifiedDate] [datetime] NULL,

CONSTRAINT [PK\_DimAccount] PRIMARY KEY CLUSTERED

(

[AccountSK] ASC

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

) ON [PRIMARY]

-- Create DimCard Table --

CREATE TABLE [dbo].[DimCard](

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

[AlternativeCardID] [varchar](10) NOT NULL,

[DispositionSK] [int] NULL,

[CardType] [varchar](25) NULL,

[IssuedDate] [date] NULL,

[InsertDate] [datetime] NULL,

[ModifiedDate] [datetime] NULL,

CONSTRAINT [PK\_DimCard] PRIMARY KEY CLUSTERED

(

[CardSK] ASC

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

) ON [PRIMARY]

-- Create DimClient Table --

CREATE TABLE [dbo].[DimClient](

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

[AlternativeClientID] [varchar](10) NOT NULL,

[FirstName] [varchar](30) NULL,

[LastName] [varchar](30) NULL,

[DistrictSK] [int] NULL,

[Gender] [varchar](10) NULL,

[DateOfBirth] [date] NULL,

[Age] [int] NULL,

[Phone] [varchar](20) NULL,

[Email] [varchar](60) NULL,

[Address\_1] [varchar](60) NULL,

[Address\_2] [varchar](60) NULL,

[Zipcode] [int] NULL,

[InsertDate] [datetime] NULL,

[ModifiedDate] [datetime] NULL,

[StartDate] [datetime] NULL,

[EndDate] [datetime] NULL,

CONSTRAINT [PK\_DimClient] PRIMARY KEY CLUSTERED

(

[ClientSK] ASC

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

) ON [PRIMARY]

-- Create DimDistrict Table --

CREATE TABLE [dbo].[DimDistrict](

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

[AlternativeDistrictID] [int] NOT NULL,

[City] [varchar](50) NULL,

[State] [varchar](50) NULL,

[Region] [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) ON [PRIMARY]

) ON [PRIMARY]

-- Create DimDisposition Table --

CREATE TABLE [dbo].[DimDisposition](

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

[AlternativeDispositionID] [varchar](10) NOT NULL,

[AccountSK] [int] NULL,

[ClientSK] [int] NULL,

[DispositionType] [varchar](25) NULL,

[InsertDate] [datetime] NULL,

[ModifiedDate] [datetime] NULL,

CONSTRAINT [PK\_DimDisposition] PRIMARY KEY CLUSTERED

(

[DispositionSK] ASC

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

) ON [PRIMARY]

-- Create DimLoan Table --

CREATE TABLE [dbo].[DimLoan](

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

[AlternativeLoanID] [varchar](10) NOT NULL,

[AccountSK] [int] NULL,

[DistrictSK] [int] NULL,

[Amount] [numeric](18, 2) NULL,

[Duration] [int] NULL,

[Payments] [numeric](18, 2) NULL,

[Status] [varchar](5) NULL,

[Loan\_date] [date] NULL,

[Purpose] [varchar](50) NULL,

[InsertDate] [datetime] NULL,

[ModifiedDate] [datetime] NULL,

CONSTRAINT [PK\_DimLoan] PRIMARY KEY CLUSTERED

(

[LoanSK] ASC

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

) ON [PRIMARY]

-- Create DimPermanentOrder Table --

CREATE TABLE [dbo].[DimPermanentOrder](

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

[AlternativeOrderID] [int] NOT NULL,

[AccountSK] [int] NULL,

[Bank\_to] [varchar](10) NULL,

[Account\_to] [int] NULL,

[Amount] [numeric](18, 2) NULL,

[OrderType] [varchar](100) NULL,

[InsertDate] [datetime] NULL,

[ModifiedDate] [datetime] NULL,

CONSTRAINT [PK\_DimPermanentOrder] PRIMARY KEY CLUSTERED

(

[OrderSK] ASC

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

) ON [PRIMARY]

-- Create FactTransaction Table --

CREATE TABLE [dbo].[FactTransaction](

[TransactionID] [varchar](10) NOT NULL,

[AccounSK] [int] NULL,

[ClientSK] [int] NULL,

[LoanSK] [int] NULL,

[OrderSK] [int] NULL,

[DispositionSK] [int] NULL,

[DateSK] [int] NULL,

[Type] [varchar](25) NULL,

[Operation] [varchar](50) NULL,

[Amount] [decimal](18, 2) NULL,

[Balance] [decimal](18, 2) NULL,

[InsertDate] [datetime] NULL,

[ModifiedDate] [datetime] NULL

) ON [PRIMARY]

## **Stored Procedure SQL Queries**

-- Stored Procedure for DimAccount --

CREATE PROCEDURE [dbo].[UpdateDimAccount]

@AccountID varchar(10),

@DistrictSK int,

@Freq varchar(50),

@startDate date

AS

BEGIN

if not exists (select AccountSK

from dbo.DimAccount

where AlternativeAccountID = @AccountID)

BEGIN

insert into dbo.DimAccount

(AlternativeAccountID, DistrictSK, Frequency, StartedDate, InsertDate, ModifiedDate)

values

(@AccountID, @DistrictSK, @Freq, @startDate, GETDATE(), GETDATE())

END;

if exists (select AccountSK

from dbo.DimAccount

where AlternativeAccountID = @AccountID)

BEGIN

update dbo.DimAccount

set DistrictSK = @DistrictSK,

Frequency = @Freq,

StartedDate = @startDate,

ModifiedDate = GETDATE()

where AlternativeAccountID = @AccountID

END;

END;

-- Stored Procedure for DimCard --

CREATE PROCEDURE [dbo].[UpdateDimCard]

@CardID varchar(10),

@DispositionKey int,

@Type varchar(25),

@IssuedDate date

AS

BEGIN

if not exists (select CardSK

from dbo.DimCard

where AlternativeCardID = @CardID)

BEGIN

insert into dbo.DimCard

(AlternativeCardID, DispositionSK, CardType, IssuedDate, InsertDate, ModifiedDate)

values

(@CardID, @DispositionKey, @Type, @IssuedDate, GETDATE(), GETDATE())

END;

if exists (select CardSK

from dbo.DimCard

where AlternativeCardID = @CardID)

BEGIN

update dbo.DimCard

set DispositionSK = @DispositionKey,

CardType = @Type,

IssuedDate = @IssuedDate,

ModifiedDate = GETDATE()

where AlternativeCardID = @CardID

END;

END;

-- Stored Procedure for DimDisposition --

CREATE PROCEDURE [dbo].[UpdateDimDisposition]

@DispositionID varchar(10),

@AccountKey int,

@ClientKey int,

@Type varchar(25)

AS

BEGIN

if not exists (select DispositionSK

from dbo.DimDisposition

where AlternativeDispositionID = @DispositionID)

BEGIN

insert into dbo.DimDisposition

(AlternativeDispositionID, AccountSK, ClientSK, DispositionType, InsertDate, ModifiedDate)

values

(@DispositionID, @AccountKey, @ClientKey, @Type, GETDATE(), GETDATE())

END;

if exists (select DispositionSK

from dbo.DimDisposition

where AlternativeDispositionID = @DispositionID)

BEGIN

update dbo.DimDisposition

set AccountSK = @AccountKey,

ClientSK = @ClientKey,

DispositionType = @Type,

ModifiedDate = GETDATE()

where AlternativeDispositionID = @DispositionID

END;

END;

-- Stored Procedure for DimDistrict --

CREATE PROCEDURE [dbo].[UpdateDimDistrict]

@DistrictID int,

@City varchar(50),

@State varchar(50),

@Region varchar(50)

AS

BEGIN

if not exists (select DistrictSK

from dbo.DimDistrict

where AlternativeDistrictID = @DistrictID)

BEGIN

insert into dbo.DimDistrict

(AlternativeDistrictID, City, State, Region, InsertDate, ModifiedDate)

values

(@DistrictID, @City, @State, @Region, GETDATE(), GETDATE())

END;

if exists (select DistrictSK

from dbo.DimDistrict

where AlternativeDistrictID = @DistrictID)

BEGIN

update dbo.DimDistrict

set City = @City,

State = @State,

Region = @Region,

ModifiedDate = GETDATE()

where AlternativeDistrictID = @DistrictID

END;

END;

-- Stored Procedure for DimLoan --

Create PROCEDURE [dbo].[UpdateDimLoan]

@LoanID varchar(10),

@AccountSK int,

@DistrictSk int,

@Amount numeric(18, 2),

@Duration int,

@Payments numeric(18, 2),

@Status varchar(5),

@LoanDate date,

@Purpose varchar(50)

AS

BEGIN

if not exists (select LoanSK

from dbo.DimLoan

where AlternativeLoanID = @LoanID)

BEGIN

insert into dbo.DimLoan

(AlternativeLoanID, AccountSK, DistrictSK, Amount, Duration, Payments, Status, Loan\_date,

Purpose, InsertDate, ModifiedDate)

values

(@LoanID, @AccountSK, @DistrictSk, @Amount, @Duration, @Payments, @Status, @LoanDate,

@Purpose, GETDATE(), GETDATE())

END;

if exists (select LoanSK

from dbo.DimLoan

where AlternativeLoanID = @LoanID)

BEGIN

update dbo.DimLoan

set AccountSK = @AccountSK,

DistrictSK = @DistrictSk,

Amount = @Amount,

Duration = @Duration,

Payments = @Payments,

Status = @Status,

Loan\_date = @LoanDate,

Purpose = @Purpose,

ModifiedDate = GETDATE()

where AlternativeLoanID = @LoanID

END;

END;

-- Stored Procedure for DimPermanentOrder --

CREATE PROCEDURE [dbo].[UpdateDimPermanentOrder]

@OrderID int,

@AccountSK int,

@Bankto varchar(10),

@AccountTo int,

@Amount numeric(18, 0),

@ordertype varchar(100)

AS

BEGIN

if not exists (select OrderSK

from dbo.DimPermanentOrder

where AlternativeOrderID = @OrderID)

BEGIN

insert into dbo.DimPermanentOrder

(AlternativeOrderID, AccountSK, Bank\_to, Account\_to, Amount, OrderType, InsertDate, ModifiedDate)

values

(@OrderID, @AccountSK, @Bankto, @AccountTo, @Amount, @ordertype, GETDATE(), GETDATE())

END;

if exists (select OrderSK

from dbo.DimPermanentOrder

where AlternativeOrderID = @OrderID)

BEGIN

update dbo.DimPermanentOrder

set AccountSK = @AccountSK,

Bank\_to = @Bankto,

Account\_to = @AccountTo,

Amount = @Amount,

OrderType = @ordertype,

ModifiedDate = GETDATE()

where AlternativeOrderID = @OrderID

END;

END;