SRILANKA INSTITUTE OF INFORMATION AND TECHNOLOGY

BSc (Hons) In Information Technology Specializing in Data Science

IT3021 - DATA WAREHOUSING AND BUSINESS INTELLIGENCE

Assignment 1

IT19175058
Mariyam M.S.S
Year 3 Semester 1
Y3S1.15(DS)

Date of Submission: 14/5/2021

Table of Contents

1. Data Set Selection and Introduction

1.1 Introduction to Data Set

The data set represents information about a collection of an anonymized financial information from a bank in Czech Republic to predict loan defaulters, starting from year 1993 to 1998. The data set has been modified to develop a scenario that meets the requirement of the assignment. The transactional details made on an account, the details of clients involved in an account, the account details, loan and standing order details are some of the information that are available through the data set.

1.2 Features of the data set

- Static characteristics like date of creation of account, and location of account can be observed in the 'account' relation. Dynamic characteristics like balances and payments can be observed in the 'transaction' relation.
- 'Client' relation contains information of the person who has access and manipulation to an account. A client can hold many accounts while an account can be manipulated by many clients. The client connects with the account via a relation named 'disposition.'
- 'Disposition' relation states the client, account and the type of disposition made by the client on the account.
- 'Loan' and 'Cards' are some services provided by the bank.
- The details of the account are represented in the 'account' relationship.
- The 'standing order' relation states the sequence of available transactions.
- 'District' relation provides demographic data of the client and account, which also help to deduce some client details.
- All transaction information is contained in the relation 'transactions.'

1.3 Link to access data set

https://www.kaggle.com/pranati25/predict-loan-defaulters

2. Preparations of Data Sources

The collected data set was analyzed and modified according to the requirement of the project. The primary link contains 8 text files, of which some were converted to csv files. Some additional information were added to few relations in order to achieve the missing requirements.

Ultimately, 2 main sources were created:

1. A database source: PredictLoanDefaultersSourceDB
2. A text file to maintain client address: clientAddress.txt

(Assumption: No client address details were provided in the data set. Since predicting how likely a client can default a loan is a challenging task most of the time the bank might require the address details of the clients for contact purposes. Therefore, a separate text source file was created to track these details. Further, it was required to meet the requirement of the assignment)

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

- account.csv
- AccountTransaction.csv
- client.csv
- card.txt
- disposition.csv
- district.txt
- loan.csv
- standingOrder.csv

Predict_loan_defaulters_staging database was created as a staging layer.

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

- DimAccount
- DimClient
- DimDistrict
- DimDisposition
- DimLoan
- DimStandingOrder
- DimDate
- DimCard

7.1 Dimension Creation Queries

3. ER Diagram Developed Using the Sources

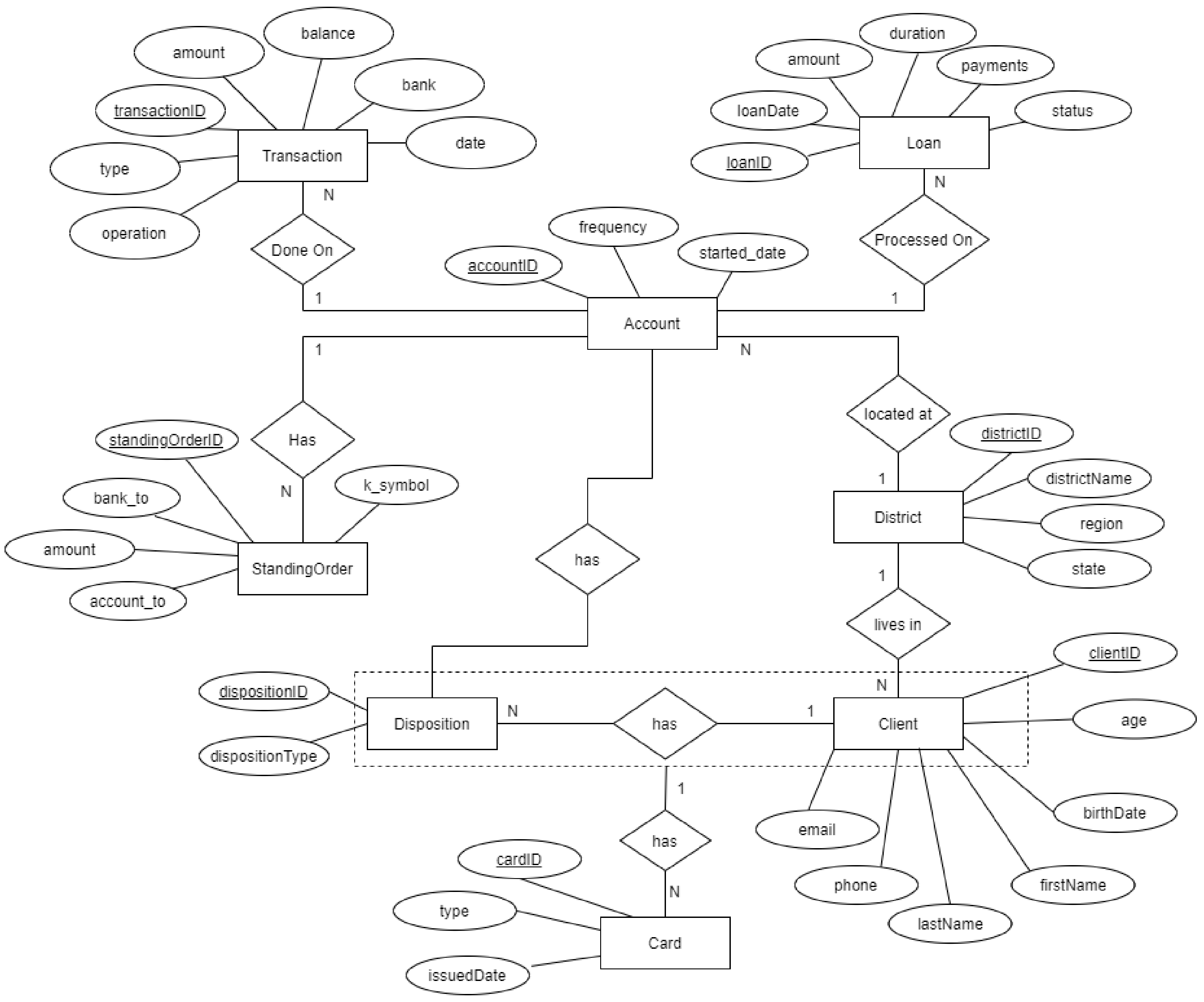


Figure 1: ER Diagram constructed using sources

4. Solution Architecture

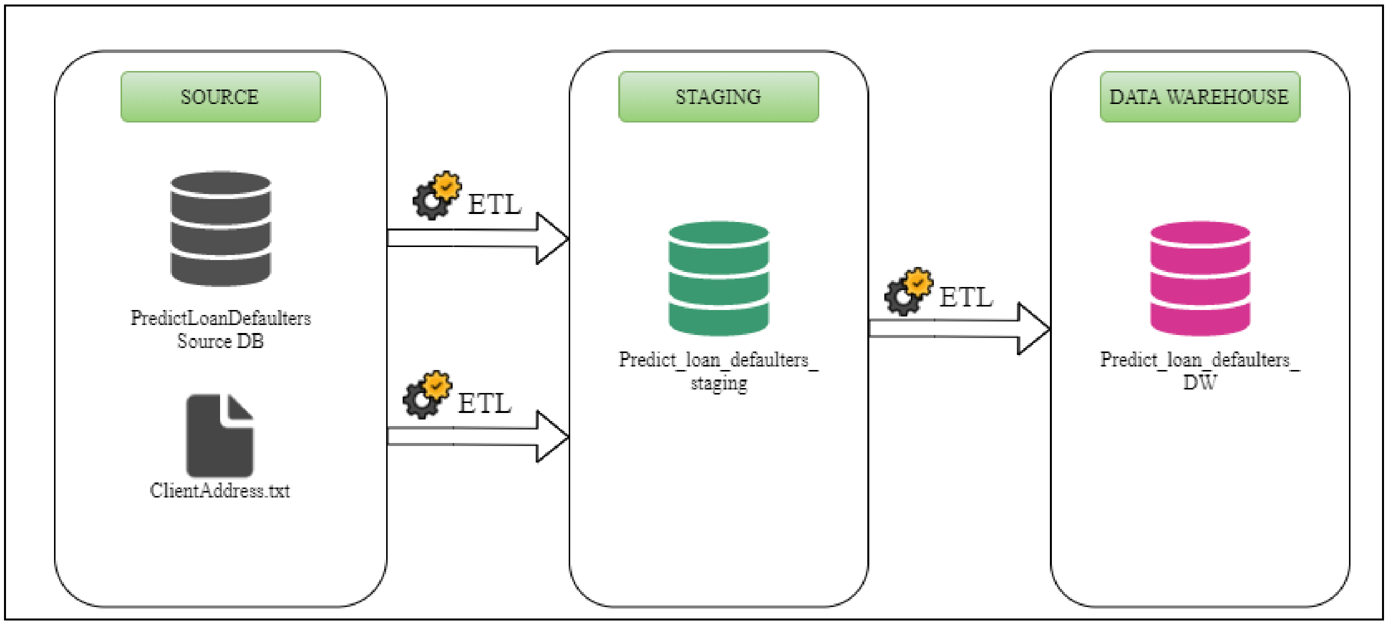

SOURCE
PredictLoanDefaulters Source DB
ClientAddress.txt
ETL
ETL
STAGING
Predict_loan_defaulters_ staging
ETL
DATA WAREHOUSE
Predict_loan_defaulters_ DW

Figure 2: Solution Architecture

Data 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 ‘PredictLoanDefaultersSourceDB’ serves as the primary data source and a flat file source namely ‘clientAddress.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. ‘Predict_loan_defaulters_staging’ 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 ‘Predict_loan_defaulters_DW’ is created in SQL as the data warehouse layer.

5. Data Warehouse Design and Implementation

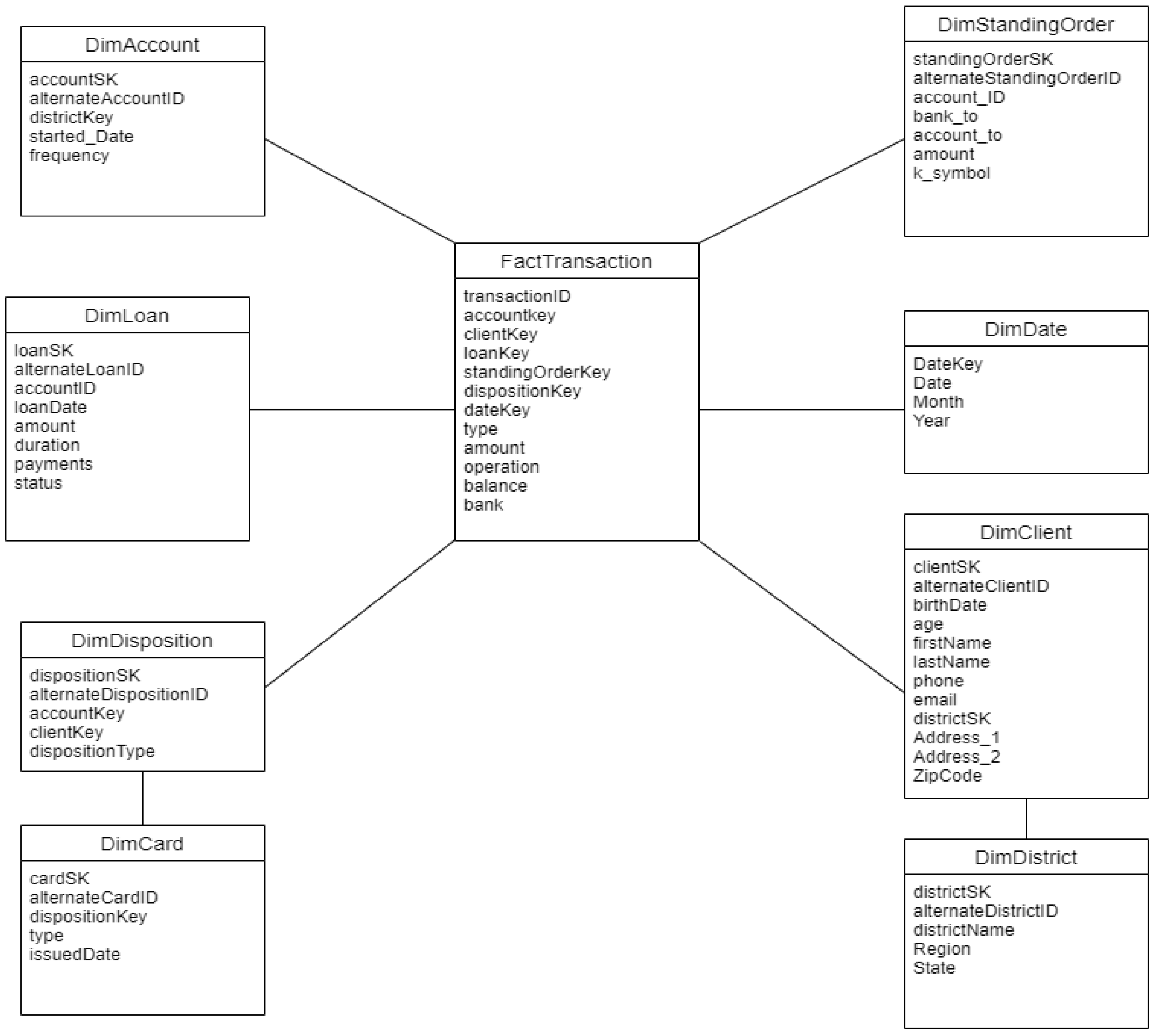


Figure 3:Snowflake schema

The data warehouse design was implemented using the snowflake schema. It is an extension of star schema and consists 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.

6. ETL Development

6.1 Data Extraction from Source to Staging Table

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	Staged Table
district	StgDistrict
account	StgAccount
client	StgClient
disposition	StgDisposition
card	StgCard
loan	StgLoan
standingOrder	StgStandingOrder
AccountTransaction	StgAccountTransaction
clientAddress.txt	StgClientAddress

clientAddress.txt	StgClientAddress

The control flow task:

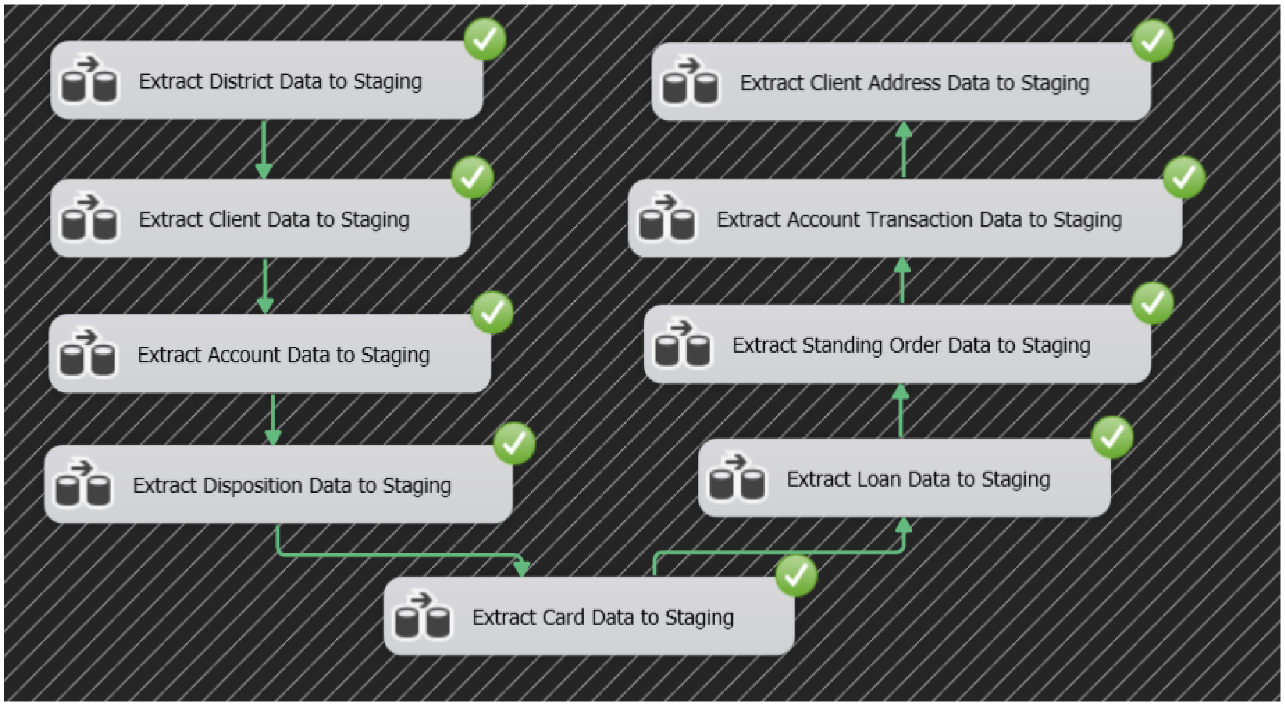

Extract District Data to Staging
Extract Client Data to Staging
Extract Account Data to Staging
Extract Disposition Data to Staging
Extract Card Data to Staging
Extract Loan Data to Staging
Extract Standing Order Data to Staging
Extract Account Transaction Data to Staging
Extract Client Address Data to Staging

Figure 4:Control flow task for staging

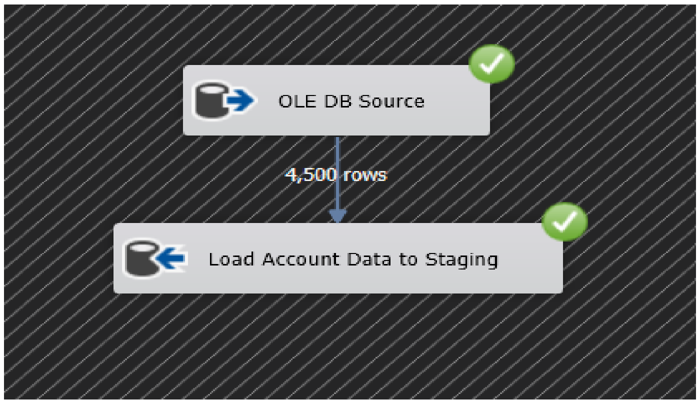


Figure 5: Staging from a database source to database destination

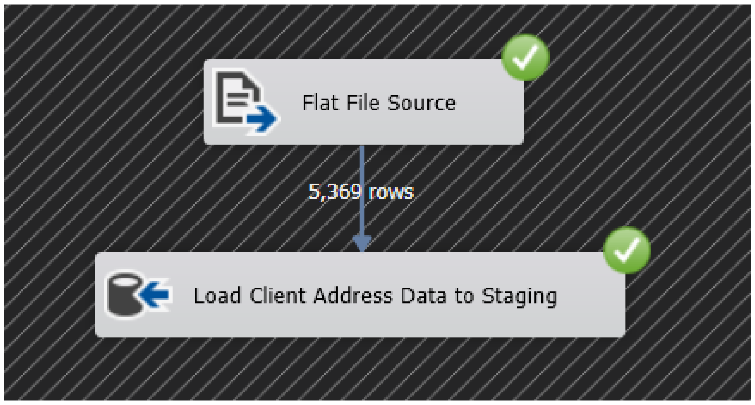


Figure 6: Staging from a flat file source to database destination

7.2 Staging Process Snapshots

6.2 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.

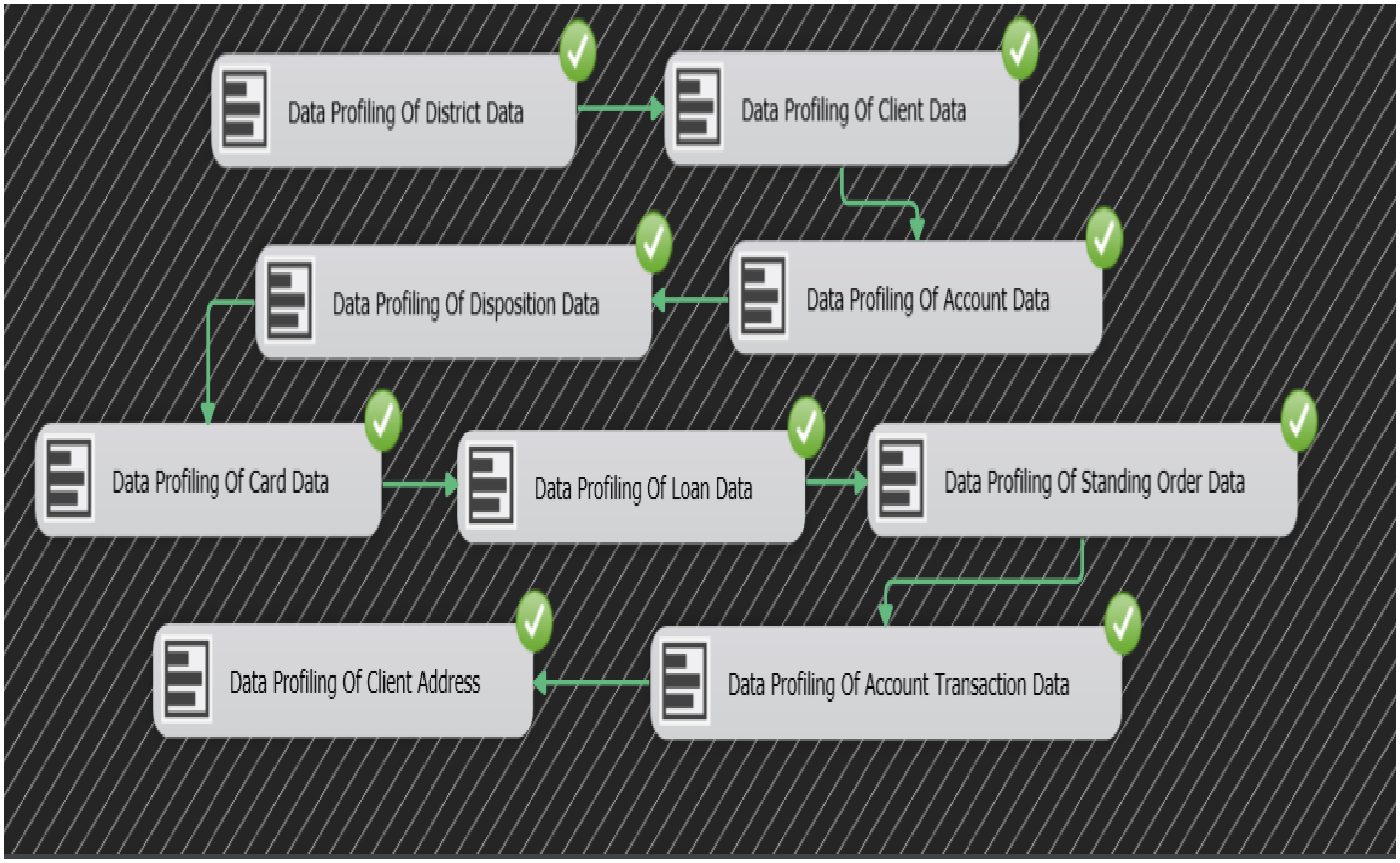

Data Profiling Of District Data
Data Profiling Of Client Data
Data Profiling Of Disposition Data
Data Profiling Of Account Data
Data Profiling Of Card Data
Data Profiling Of Loan Data
Data Profiling Of Standing Order Data
Data Profiling Of Client Address
Data Profiling Of Account Transaction Data

Figure 7:Data Profiling task flow

6.3 Transform and Load 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:

1. The district dimension has no dependencies with any other dimensions; therefore it is loaded first
2. Both account and client can be loaded next since they contain reference to the district. Here, the account dimension is loaded as the next dimension.
3. Loan and standing order contain reference to account; therefore loan dimension is loaded following account, followed by the standing order dimension.
4. Next, the client dimension is loaded.
5. Disposition dimension contains reference to both client and account dimensions. On that basis its loaded next after the loading of both client and account.
6. Card dimension contains reference to the disposition. Thus, its loaded after disposition.
7. Finally, the fact transaction is loaded as it contains references to many other dimensions.

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

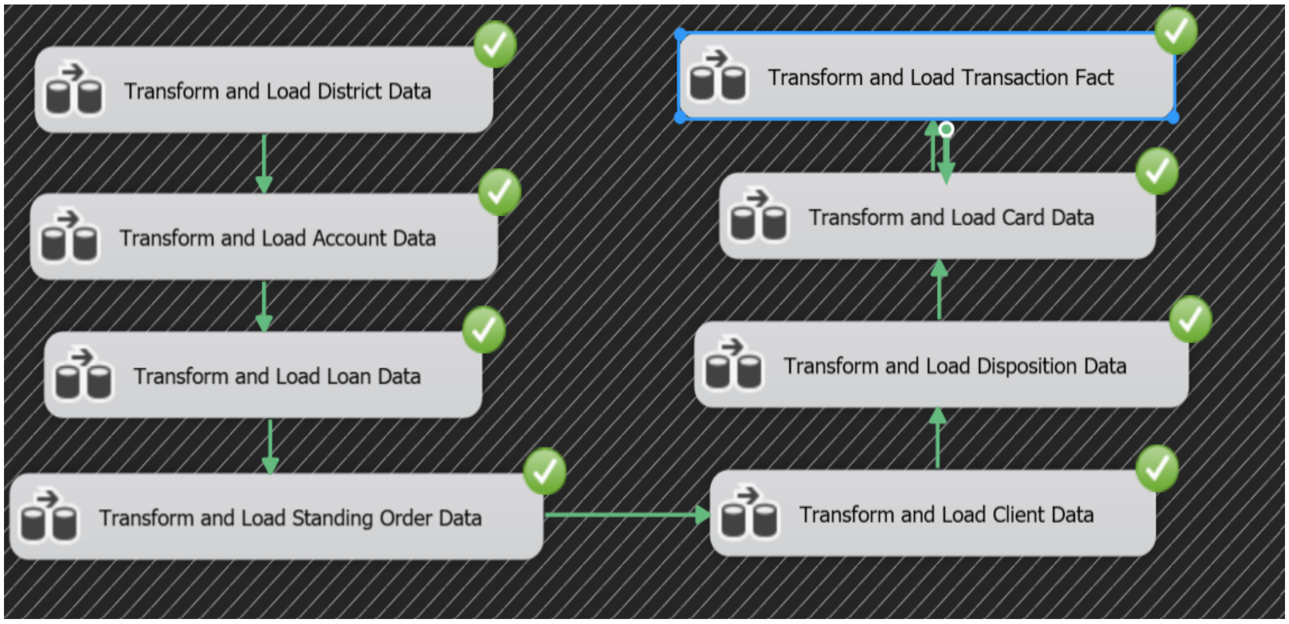

Transform and Load District Data
Transform and Load Account Data
Transform and Load Loan Data
Transform and Load Standing Order Data
Transform and Load Client Data
Transform and Load Disposition Data
Transform and Load Card Data
Transform and Load Transaction Fact

Figure 8: Control flow task of data warehouse transformation and loading

❖ Loading of Account data to DimAccount

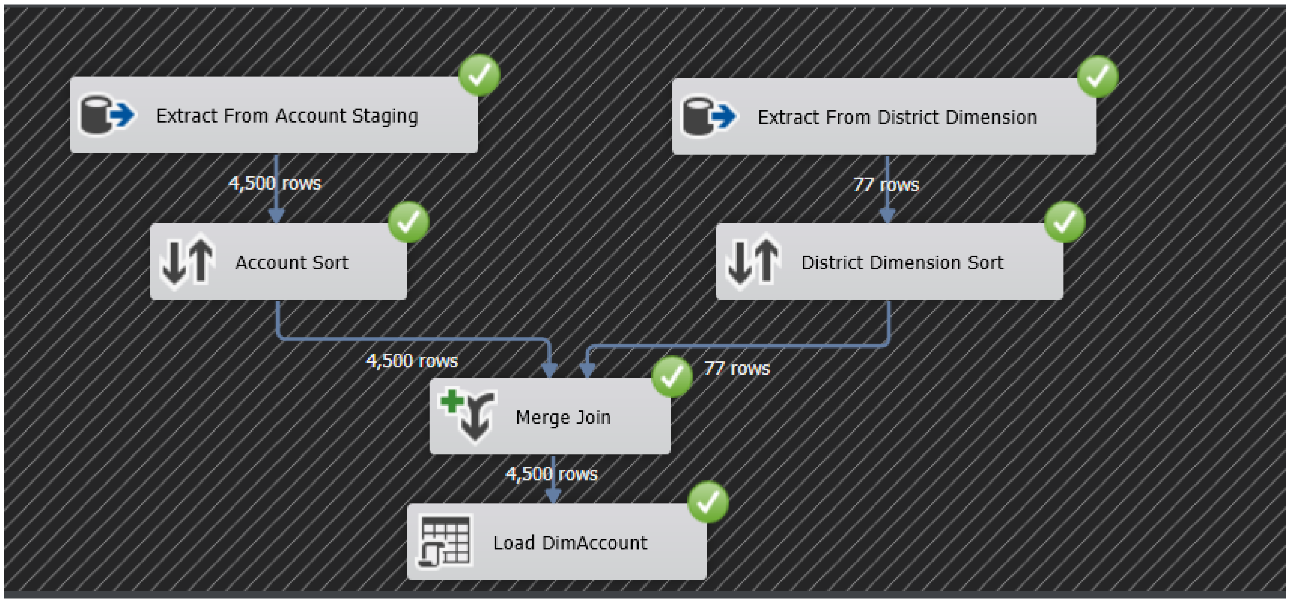


Figure 9: Data flow task of account dimension transformation and loading

DimAccount contains a reference to DimDistrict. In order to get the district surrogate key to account dimension, data was extracted from both dimensions and **sort**ed based on district id. Then they were **merge**d to load into DimAccount. Some accounts may not have a district id; thus the merge join was done using left outer join.

❖ Loading of standing order data to DimStandingOrder

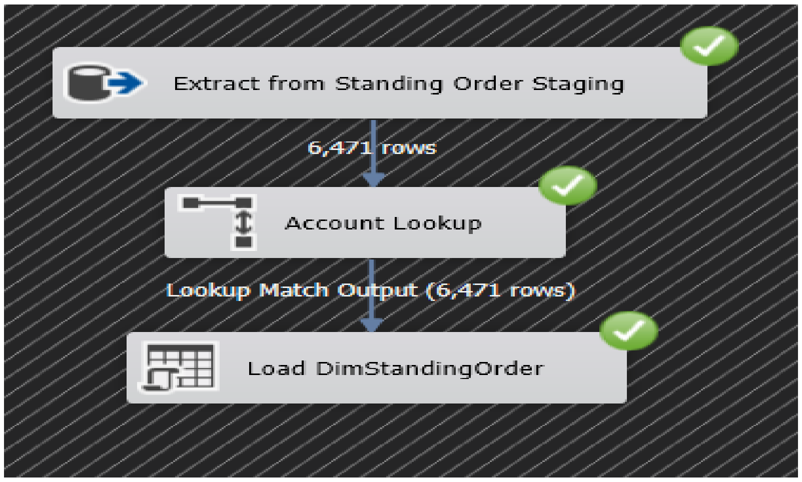

Extract from Standing Order Staging
6,471 rows
Account Lookup
Lookup Match Output (6,471 rows)
Load DimStandingOrder

Figure 10:Data flow task of standing order dimension transformation and loading

Standing order dimension contains a reference to account dimension. In order to get the surrogate key of account to standing order dimension a **lookup** process was performed.

- Loading Client data to DimClient

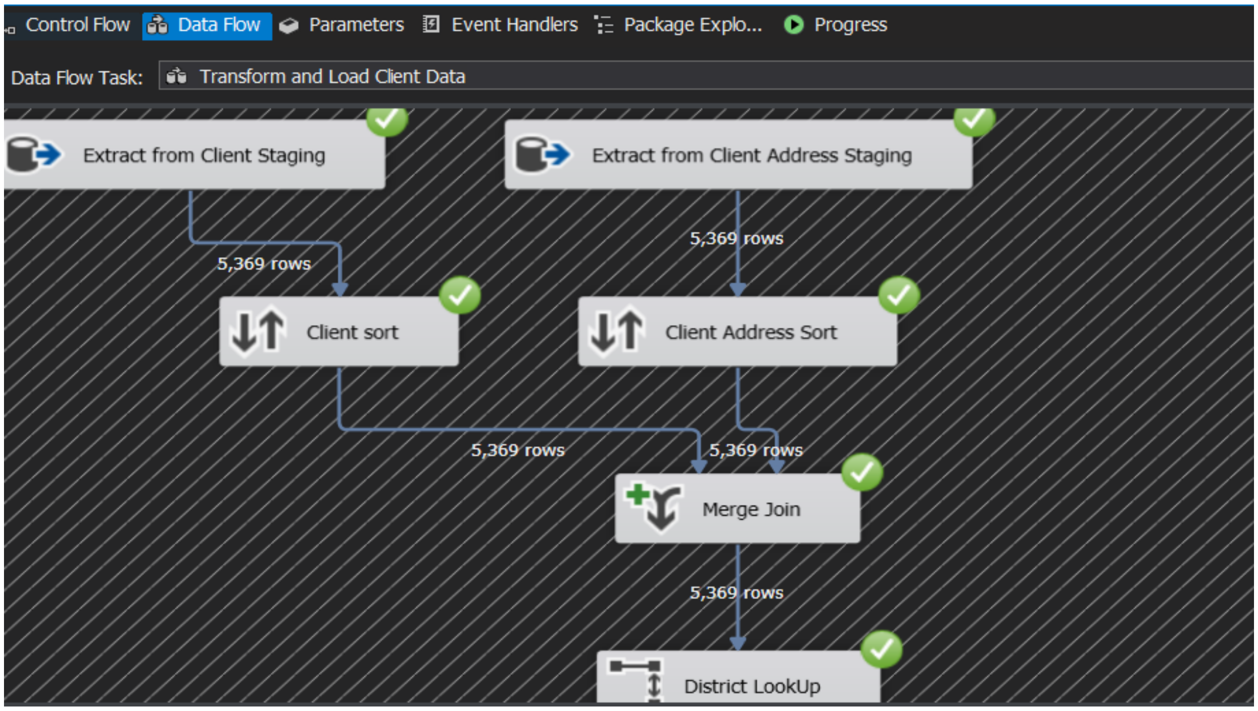
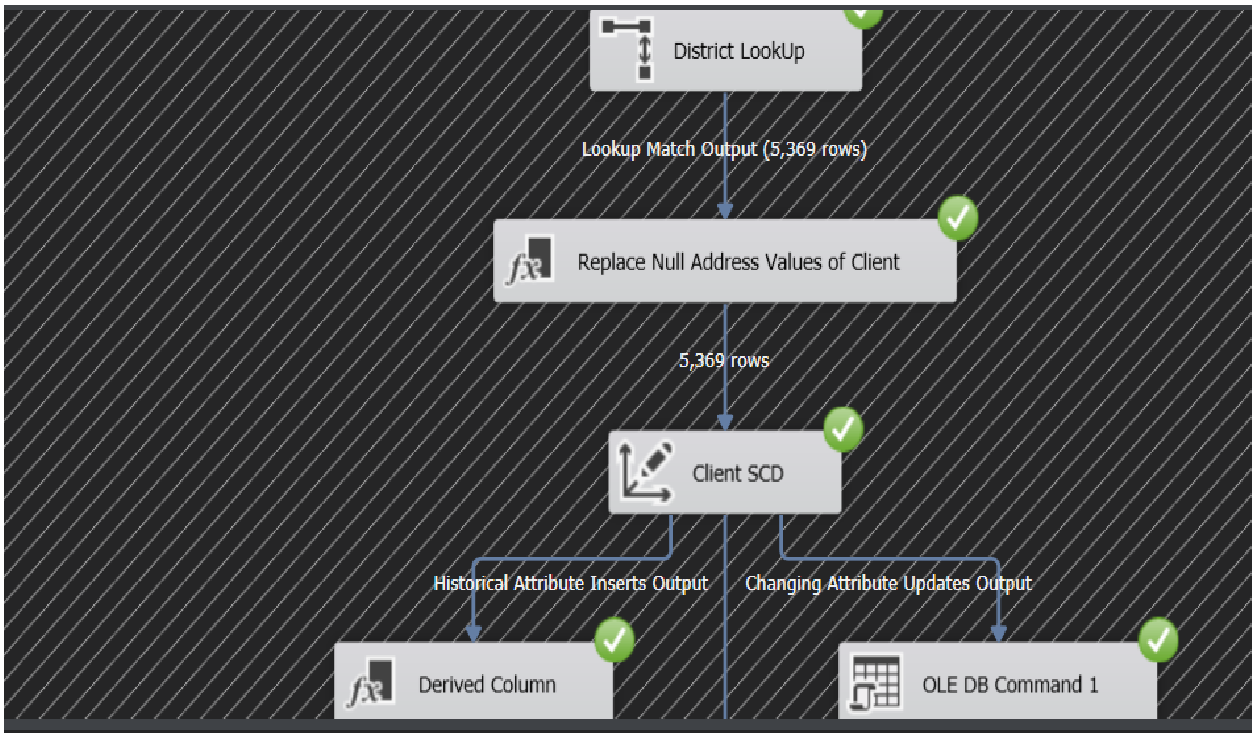


Figure 11



District LookUp
Lookup Match Output (5,369 rows)
Replace Null Address Values of Client
5,369 rows
Client SCD
Historical Attribute Inserts Output
Changing Attribute Updates Output
Derived Column
OLE DB Command 1

Figure 12

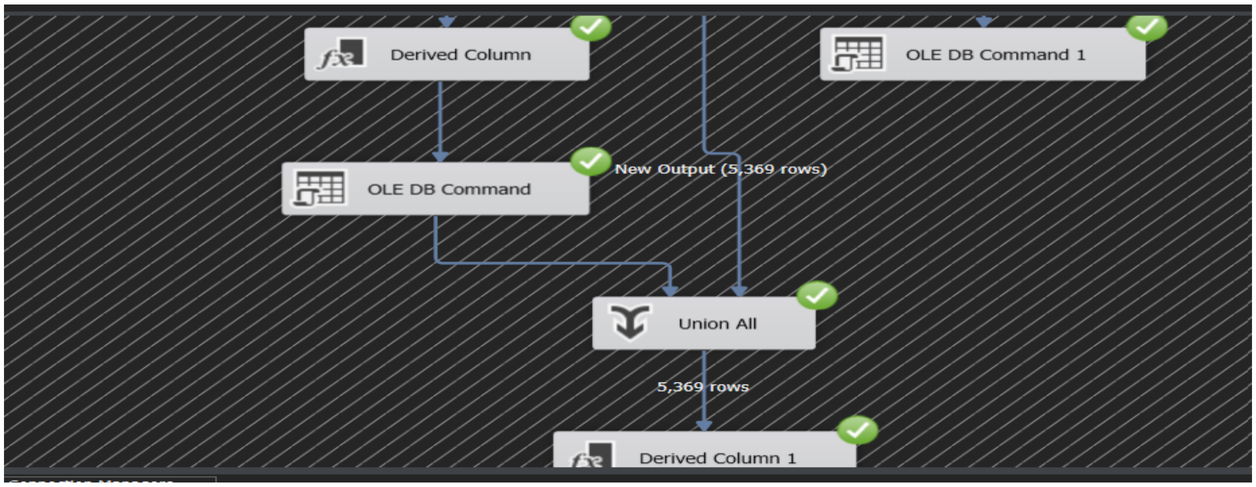


Figure 13

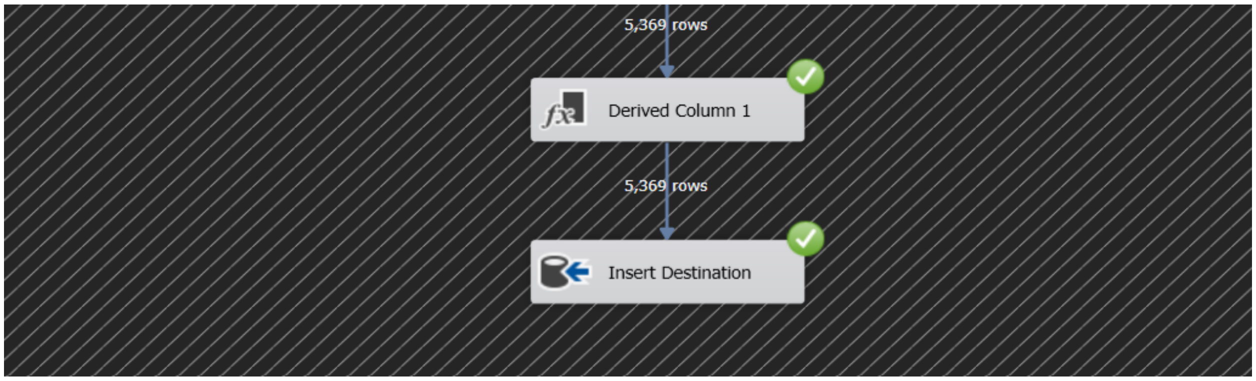


Figure 14: Data flow task of client dimension transformation and loading

DimClient contains the address details as well as other client details together. Therefore, the data of client was extracted from both client staging and client address staging, **sort**ed and **merge**d. 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.

❖ Loading Fact Transaction

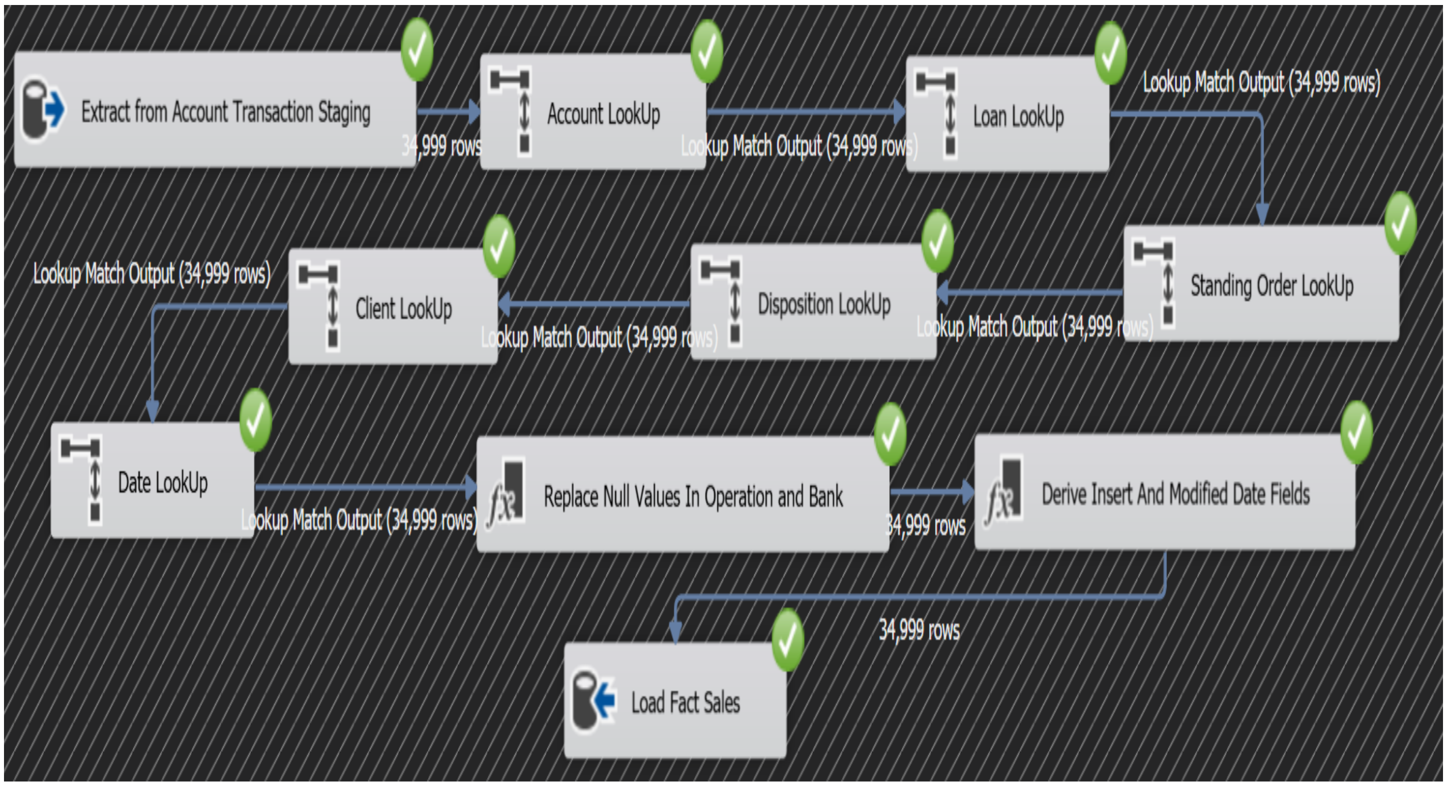

Extract from Account Transaction Staging
34,999 rows
Account LookUp
Lookup Match Output (34,999 rows)
Loan LookUp
Lookup Match Output (34,999 rows)
Standing Order LookUp
Lookup Match Output (34,999 rows)
Disposition LookUp
Lookup Match Output (34,999 rows)
Client LookUp
Lookup Match Output (34,999 rows)
Date LookUp
Lookup Match Output (34,999 rows)
Replace Null Values In Operation and Bank
34,999 rows
Derive Insert And Modified Date Fields
34,999 rows
Load Fact Sales

Figure 15:Transformation and loading of the fact table

The fact transaction contains references to account, loan, standing order, disposition, client and date. In order to get the surrogate keys as references, lookup processes were carried out for all references. The columns operation and bank contain **null values**, which was **replaced** as a data cleansing step. Insert date and modified date are **derived columns**. Finally, the fact table was loaded to its destination.

- Dimensions like account, loan, standing 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
- 7.4 Loading to Data Warehouse Snapshots

7. Appendix

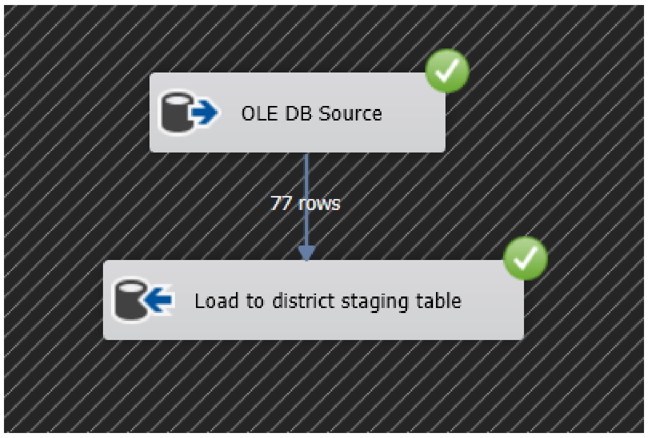
7.1 Dimension Creation Queries

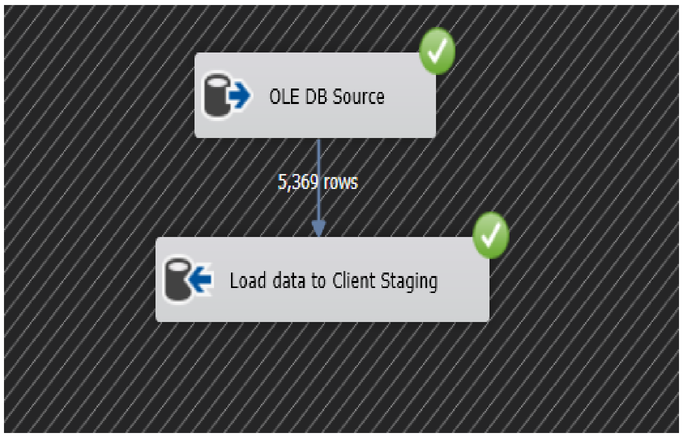
```
USE [Pedict_loan_defaulters_DW]
GO

/****** Object:  Table [dbo].[DimAccount]    Script Date: 5/10/2021 1:53:06 PM
******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[DimAccount](
 [accountSK] [int] IDENTITY(1,1) NOT NULL,
 [alternateAccountID] [varchar](50) NOT NULL,
 [districtKey] [int] NULL,
 [frequency] [varchar](50) NULL,
 [started_date] [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]

GO


/****** Object:  Table [dbo].[DimClient]    Script Date: 5/10/2021 1:53:46 PM
******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[DimClient](
 [clientSK] [int] IDENTITY(1,1) NOT NULL,
 [alternateClientID] [varchar](50) NOT NULL,
 [birthDate] [date] NULL,
 [age] [int] NULL,
 [firstName] [varchar](60) NULL,
 [lastName] [varchar](60) NULL,
 [phone] [varchar](50) NULL,
 [email] [varchar](50) NULL,
 [districtSK] [int] NULL,
 [Address_1] [nvarchar](100) NULL,
 [Address_2] [nvarchar](100) NULL,
 [ZipCode] [int] NULL,
 [StartDate] [datetime] NULL,
 [EndDate] [datetime] NULL,
 [InsertDate] [datetime] NULL,
 [ModifiedDate] [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]

GO


/****** Object:  Table [dbo].[DimDisposition]    Script Date: 5/10/2021 1:54:15 PM
******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[DimDisposition](
 [dispositionSK] [int] IDENTITY(1,1) NOT NULL,
 [alternateDispositionID] [int] NOT NULL,
 [clientKey] [int] NULL,
 [accountKey] [int] NULL,
 [dispositionType] [varchar](50) 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]

GO


/****** Object:  Table [dbo].[DimDistrict]    Script Date: 5/10/2021 1:54:50 PM
******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[DimDistrict](
 [districtSK] [int] IDENTITY(1,1) NOT NULL,
 [alternateDistrictID] [int] NOT NULL,
 [districtName] [nvarchar](50) NULL,
 [Region] [nvarchar](50) NULL,
 [State] [nvarchar](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]
```

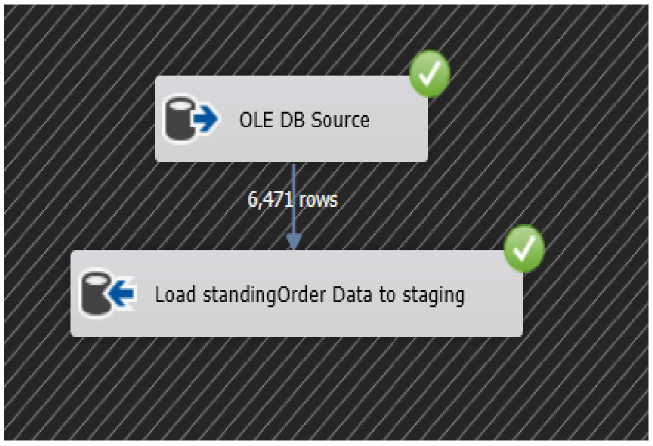
```
) ON [PRIMARY]

GO


/****** Object:  Table [dbo].[DimLoan]    Script Date: 5/10/2021 1:55:19 PM ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[DimLoan](
 [loanSK] [int] IDENTITY(1,1) NOT NULL,
 [alternateLoanID] [int] NOT NULL,
 [accountID] [int] NULL,
 [loanDate] [date] NULL,
 [amount] [numeric](38, 0) NULL,
 [duration] [int] NULL,
 [payments] [int] NULL,
 [status] [varchar](60) 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]

GO




/****** Object:  Table [dbo].[DimStandingOrder]    Script Date: 5/10/2021 1:55:37
PM ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[DimStandingOrder](
 [standingOrderSK] [int] IDENTITY(1,1) NOT NULL,
 [alternateStandingOrderID] [int] NOT NULL,
 [account_ID] [int] NULL,
 [bank_to] [varchar](50) NULL,
 [account_to] [int] NULL,
 [amount] [numeric](38, 0) NULL,
 [k_symbol] [varchar](100) NULL,
 [InsertDate] [datetime] NULL,
 [ModifiedDate] [datetime] NULL,
 CONSTRAINT [PK_DimStandingOrder] PRIMARY KEY CLUSTERED
(
 [standingOrderSK] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
```

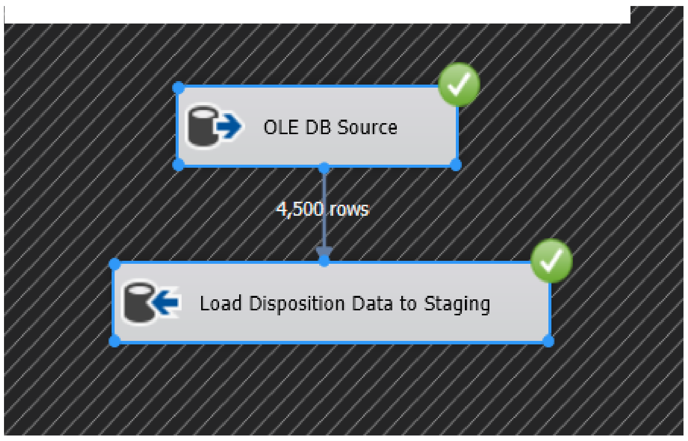
```
) ON [PRIMARY]

GO



/****** Object:  Table [dbo].[DimCard]    Script Date: 5/10/2021 1:56:08 PM ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[DimCard](
 [cardSK] [int] IDENTITY(1,1) NOT NULL,
 [alternateCardID] [int] NOT NULL,
 [dispositionKey] [int] NULL,
 [type] [nvarchar](60) 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]
GO


/****** Object:  Table [dbo].[FactTransaction]    Script Date: 5/10/2021 1:56:33 PM
******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[FactTransaction](
 [transactionID] [int] NOT NULL,
 [accountKey] [int] NULL,
 [clientKey] [int] NULL,
 [loanKey] [int] NULL,
 [standingOrderKey] [int] NULL,
 [dispositionKey] [int] NULL,
 [dateKey] [int] NULL,
 [type] [varchar](100) NULL,
 [amount] [numeric](38, 0) NULL,
 [balance] [numeric](38, 0) NULL,
 [operation] [varchar](100) NULL,
 [bank] [varchar](50) NULL,
 [InsertDate] [datetime] NULL,
 [ModifiedDate] [datetime] NULL
) ON [PRIMARY]
GO
```

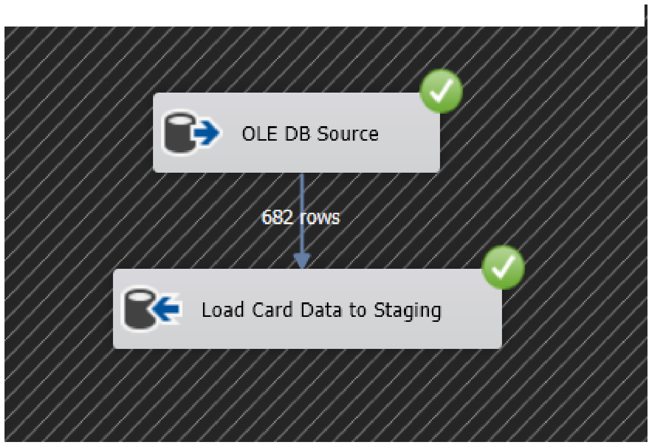
7.2 Staging Process Snapshot

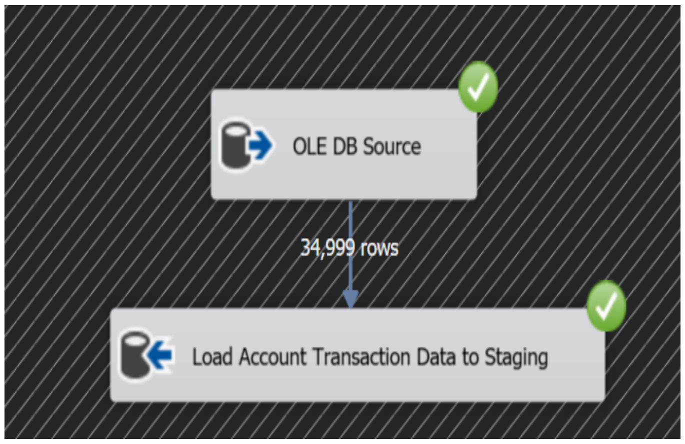












7.3 Stored Procedure Queries

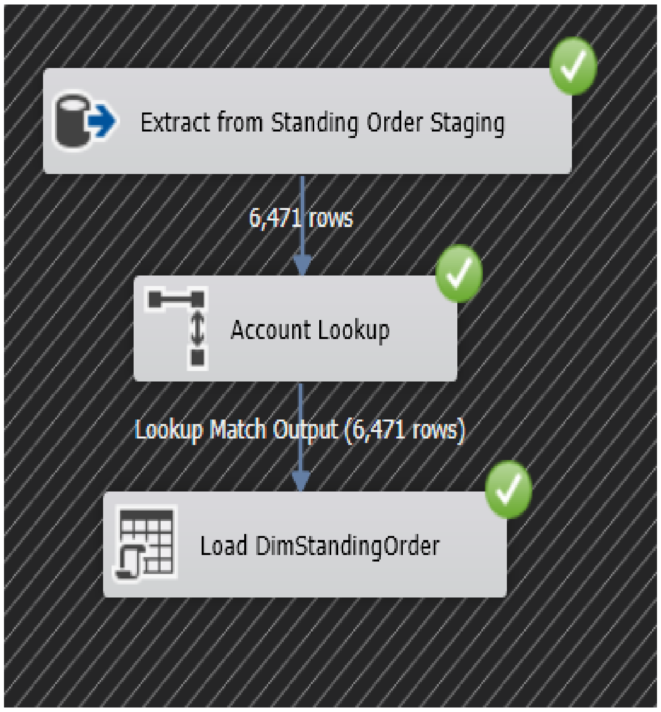
```
---------------------------------- Procedure Update DimDistrict --------------------------
-----------------------------
CREATE PROCEDURE dbo.UpdateDimDistrict
@districtID int,
@districtName nvarchar(50),
@Region nvarchar(50),
@State nvarchar(50)
AS
BEGIN
	if not exists (select districtSK
	from dbo.DimDistrict
	where alternateDistrictID = @districtID)
	BEGIN
		insert into dbo.DimDistrict (alternateDistrictID, districtName, Region,
State, InsertDate, ModifiedDate)
		values (@districtID, @districtName, @Region, @State, GETDATE(), GETDATE())
	END;
	if exists (select districtSK
	from dbo.DimDistrict
	where alternateDistrictID = @districtID)
	BEGIN
		update dbo.DimDistrict set
		districtName = @districtName,
		Region = @Region,
		State = @State,
		ModifiedDate = GETDATE()
		where alternateDistrictID = @districtID
	END;
END;



------------------------------------ Procedure Update Dim Account ------------------------
------------------------
CREATE PROCEDURE dbo.UpdateDimAccount
@accountID varchar(50),
@districtID int,
@frequency varchar(50),
@started_date date
AS
BEGIN
if not exists (select accountSK
from dbo.DimAccount
where alternateAccountID = @accountID)
BEGIN
insert into dbo.DimAccount
(alternateAccountID, districtKey, frequency, started_date, InsertDate, ModifiedDate)
values
(@accountID, @districtID, @frequency, @started_date, GETDATE(), GETDATE())
END;
if exists (select accountSK
from dbo.DimAccount
where alternateAccountID = @accountID)
BEGIN
```

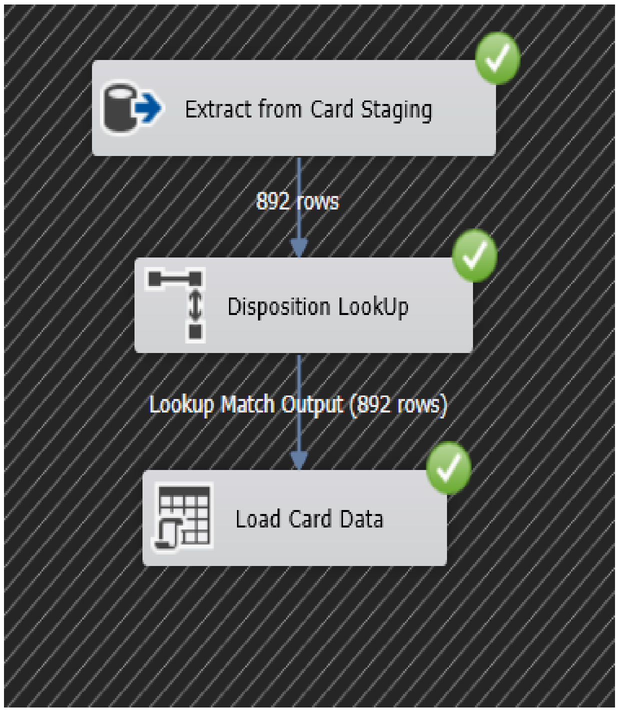
```
update dbo.DimAccount
set districtKey = @districtID,
frequency = @frequency,
started_date = @started_date,
ModifiedDate = GETDATE()
where alternateAccountID = @accountID
END;
END;

---------------------------- Procedure Update Loan --------------------------------------
----------
CREATE PROCEDURE dbo.UpdateDimLoan
@loanID int,
@accountID int,
@loanDate date,
@amount numeric(38,0),
@duration int,
@payments int,
@status varchar(60)
AS
BEGIN
if not exists (select loanSK from dbo.DimLoan where alternateLoanID = @loanID)
BEGIN
insert into dbo.DimLoan (alternateLoanID, accountID, loanDate, amount, duration,
payments, status,
InsertDate, ModifiedDate)
values(@loanID, @accountID, @loanDate, @amount, @duration, @payments, @status, GETDATE(),
GETDATE())
END;
if exists (select loanSK from dbo.DimLoan where alternateLoanID = @loanID)
BEGIN
update dbo.DimLoan
set accountID = @accountID,
loanDate = @loanDate,
amount = @amount,
duration = @duration,
payments = @payments,
status = @status,
ModifiedDate = GETDATE()
where alternateLoanID = @loanID
END;
END;

----------------------------- Procedure to update Standing Order ------------------------
--------------------------
CREATE PROCEDURE dbo.UpdateDimStandingOrder
@standingOrderID int,
@account_ID int,
@bank_to varchar(50),
@account_to int,
@amount numeric(38,0),
@k_symbol varchar(100)
AS
BEGIN
if not exists (select standingOrderSK from dbo.DimStandingOrder where
alternateStandingOrderID = @standingOrderID)
BEGIN
```

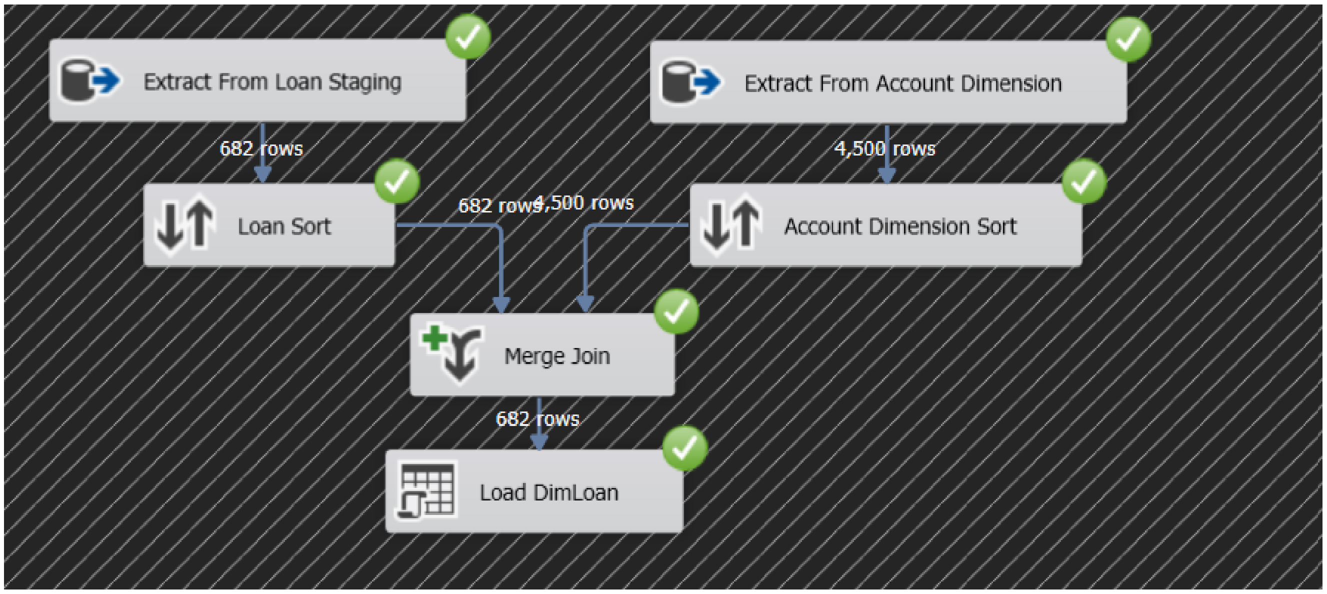
```sql
insert into DimStandingOrder (alternateStandingOrderID, account_ID, bank_to, account_to,
amount, k_symbol,
InsertDate, ModifiedDate)
values(@standingOrderID, @account_ID, @bank_to, @account_to, @amount, @k_symbol,
GETDATE(), GETDATE())
END;
if exists (select standingOrderSK from dbo.DimStandingOrder where
alternateStandingOrderID = @standingOrderID)
BEGIN
update dbo.DimStandingOrder
set account_ID = @account_ID,
bank_to = @bank_to,
account_to = @account_to,
amount = @amount,
k_symbol = @k_symbol,
ModifiedDate = GETDATE()
where alternateStandingOrderID = @standingOrderID
END;
END;

----------------------------- Procedure to update Disposition ---------------------------
-----------------------
CREATE PROCEDURE dbo.UpdateDimDisposition
@dispositionID int,
@clientKey int,
@accountKey int,
@type varchar(50)
AS
BEGIN
if not exists (select dispositionSK from dbo.DimDisposition where alternateDispositionID
= @dispositionID)
BEGIN
insert into DimDisposition (alternateDispositionID, clientKey, accountKey,
dispositionType,
InsertDate, ModifiedDate)
values(@dispositionID, @clientKey, @accountKey, @type, GETDATE(), GETDATE())
END;
if exists (select dispositionSK from dbo.DimDisposition where alternateDispositionID =
@dispositionID)
BEGIN
update dbo.DimDisposition
set clientKey = @clientKey,
accountKey = @accountKey,
dispositionType = @type,
ModifiedDate = GETDATE()
where alternateDispositionID = @dispositionID
END;
END;


----------------------------- Procedure to update Disposition ---------------------------
-----------------------
CREATE PROCEDURE dbo.UpdateDimCard
@cardID int,
@dispositionKey int,
@type nvarchar(60),
@issuedDate date
AS
```

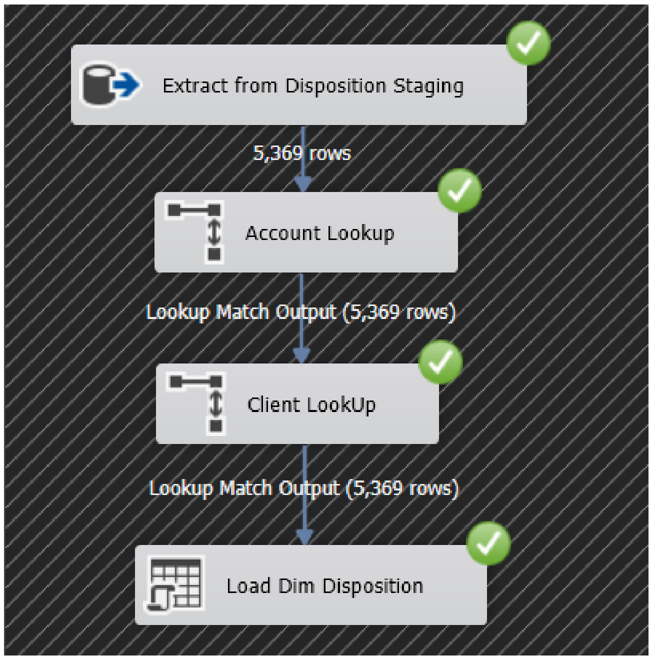
```sql
BEGIN
if not exists (select cardSK from dbo.DimCard where alternateCardID = @cardID)
BEGIN
insert into DimCard (alternateCardID, dispositionKey, type, issuedDate,
InsertDate, ModifiedDate)
values(@cardID, @dispositionKey, @type, @issuedDate, GETDATE(), GETDATE())
END;
if exists (select cardSK from dbo.DimCard where alternateCardID = @cardID)
BEGIN
update dbo.DimCard
set dispositionKey = @dispositionKey,
type = @type,
issuedDate = @issuedDate,
ModifiedDate = GETDATE()
where alternateCardID = @cardID
END;
             END;
```

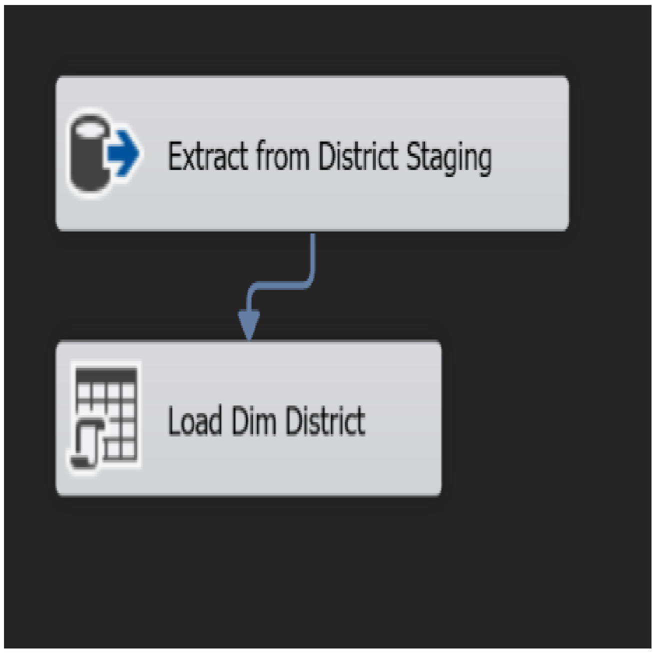
7.4 Loading to Data Warehouse Snapshots


Extract from Standing Order Staging
6,471 rows
Account Lookup
Lookup Match Output (6,471 rows)
Load DimStandingOrder






Extract from Disposition Staging
5,369 rows
Account Lookup
Lookup Match Output (5,369 rows)
Client LookUp
Lookup Match Output (5,369 rows)
Load Dim Disposition


Extract from District Staging
Load Dim District