Sri Lanka Institute of Information Technology

Assignment I

Data Warehouse & Business Intelligence

2022

Pallawala P.K.B.D.S
IT20123772

Contents

1. Data set selection & Preparation

The selected data source is a collection of transactional data. The link to the source data set is mentioned below:

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

Modifications were done to the original data set derived from the source. This data set reflects combinations between retail banking of clients, related to credit card payments, loans and transactions. Client details, client transaction information, account details, loan details and credit card payment details are some of the key details included in the data set.

The three main sources are listed below:

- SQL Database
- One text file – District details
- One xml file – Extra loan details

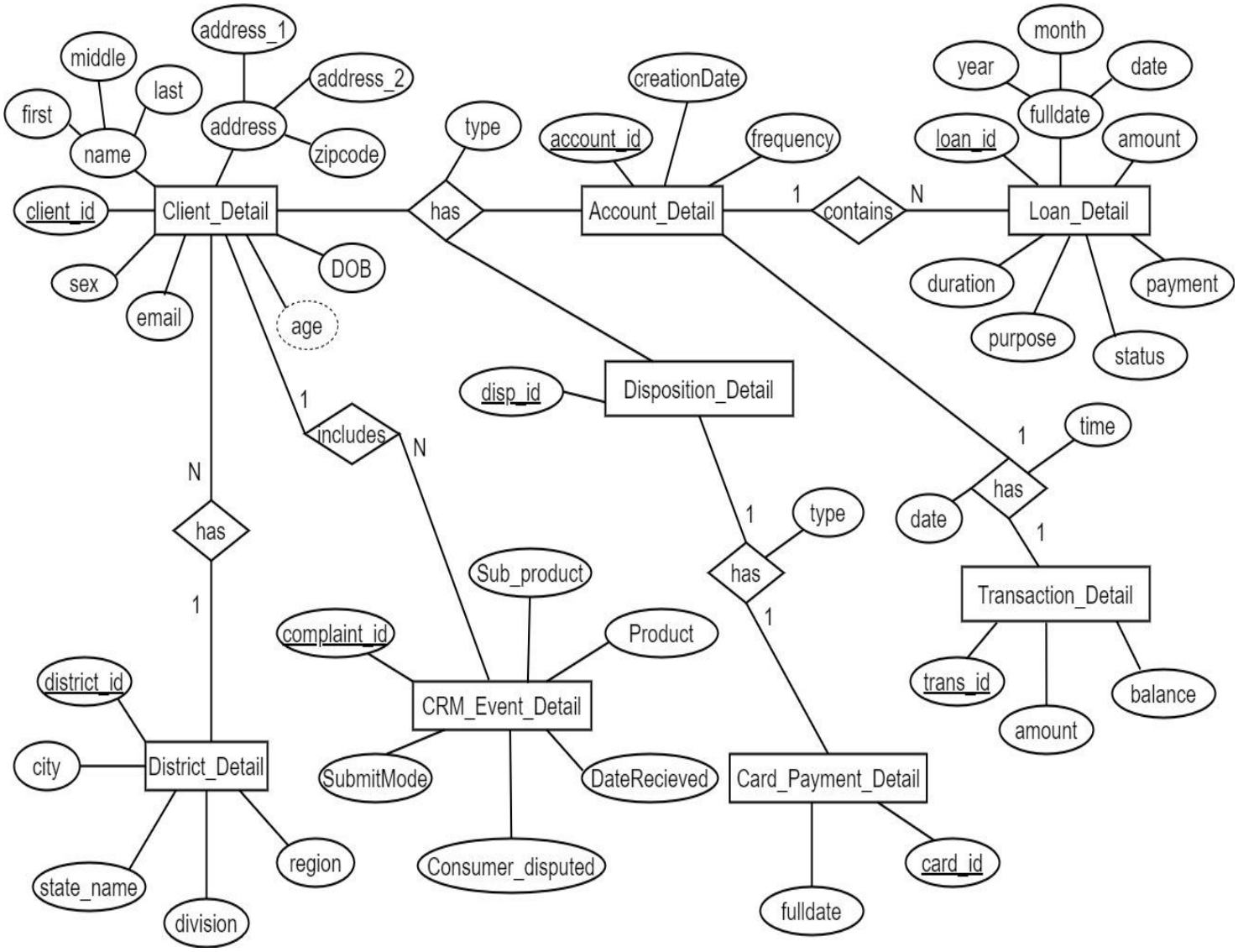
Also, the below mentioned CSV files were imported to the SQL source database.

- Account Details
- Client Details
- Card Payment Details
- Loan Details
- Transaction Details
- Disposition Details
- CRM Event Details

Description of the data set:

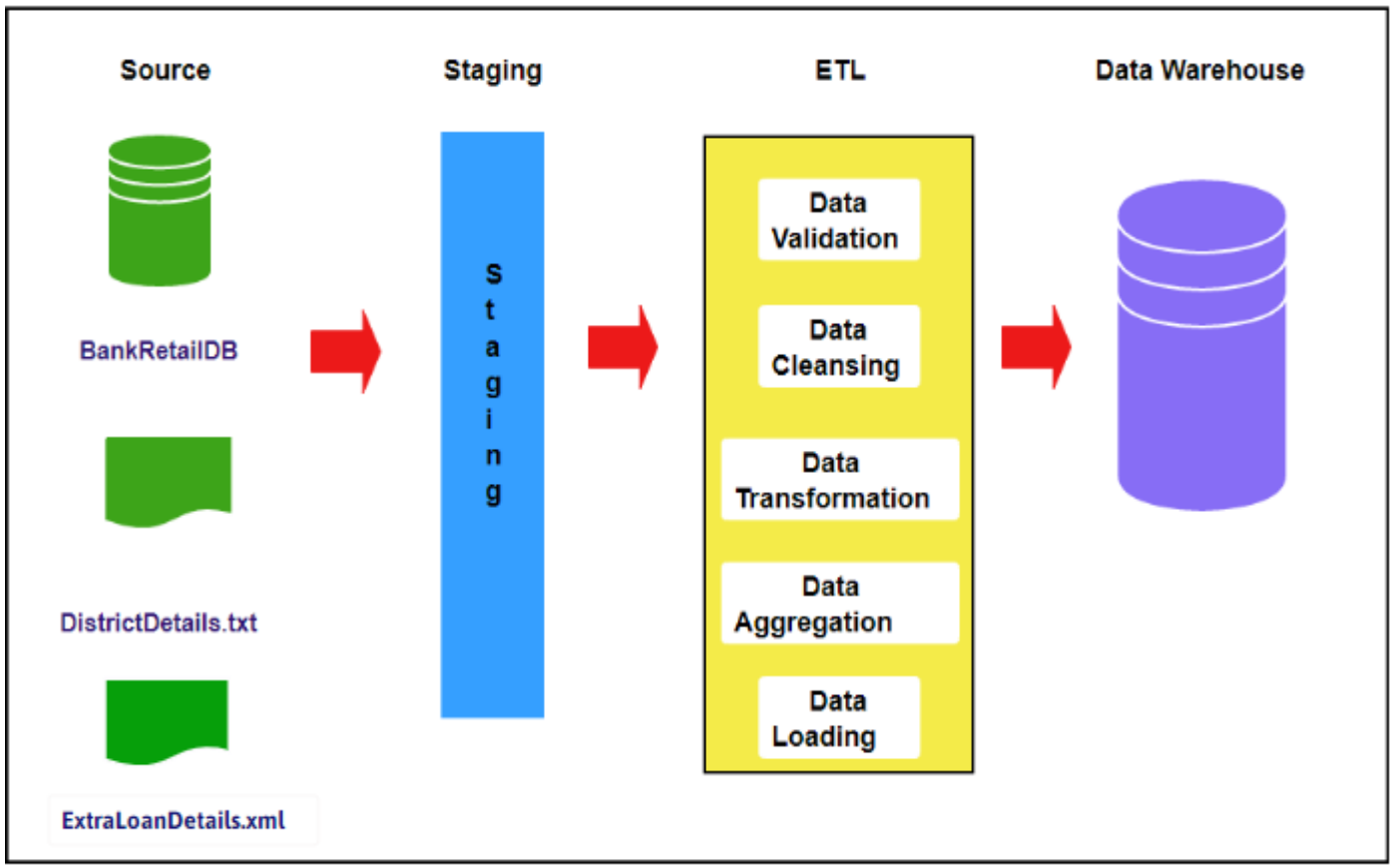
Source Table name	Column name	Data type	Target Table name	Description
Client Detail	client_id	nvarchar(50)	DimClient	Details of clients
	sex	nvarchar(20)		
	DOB	datetime		
	age	int		
	first	nvarchar(50)		
	middle	nvarchar(50)		
	last	nvarchar(50)		
	email	nvarchar(50)		
	address_1	nvarchar(50)		
	address_2	nvarchar(50)		
	zipcode	int		
	district_id	int		
Account Detail	account_id	nvarchar(50)	DimAccount	Client account details
	frequency	nvarchar(50)		
	creationDate	datetime		
Card Payment Detail	card_id	nvarchar(50)	DimCard	Credit card transaction Details
	type	nvarchar(50)		
	fulldate	datetime		
Loan Detail	loan_id	nvarchar(50)	DimLoan	Details of loans obtained by clients
	amount	money		
	payment	money		
	duration	int		
	fulldate	datetime		
	year	int		
	month	int		
	day	int		
Extra Loan Detail	status	nvarchar(50)		Extra Details of loans
	purpose	nvarchar(50)		
Transaction Detail	account_id	nvarchar(50)	Fact_Transactions	Details of the client transactions
	trans_id	nvarchar(50)		
	amount	money		
	balance	money		
	date	datetime		
	time	timestamp		
District Detail	district_id	int	DimDistrict	Details of districts which clients live in
	state	nvarchar(50)		
	city	nvarchar(50)		
	region	nvarchar(50)		
	division	nvarchar(50)		
Disposition Detail	disp_id	nvarchar(50)	DimDisposition	Details of dispositions done by clients using accounts
	account_id	nvarchar(50)		
	client_id	nvarchar(50)		
	type	nvarchar(30)		
CRM_Event_Detail	DateReceived	datetime	DimCRMEvent	Details of complaints done by clients regarding the services
	Product	nvarchar(50)		
	Sub_product	nvarchar(50)		
	Issue	nvarchar(50)		
	SubmitMode	nvarchar(50)		
	Consumer_disputed	nvarchar(50)		
	complaint_id	nvarchar(50)		
	client_id	nvarchar(50)		

ER Diagram



This diagram shows the connection between the entities in the data set

2. Solution Architecture



As explained, first step is staging the source data set. After the staging layer the belowmentioned staging tables are created:

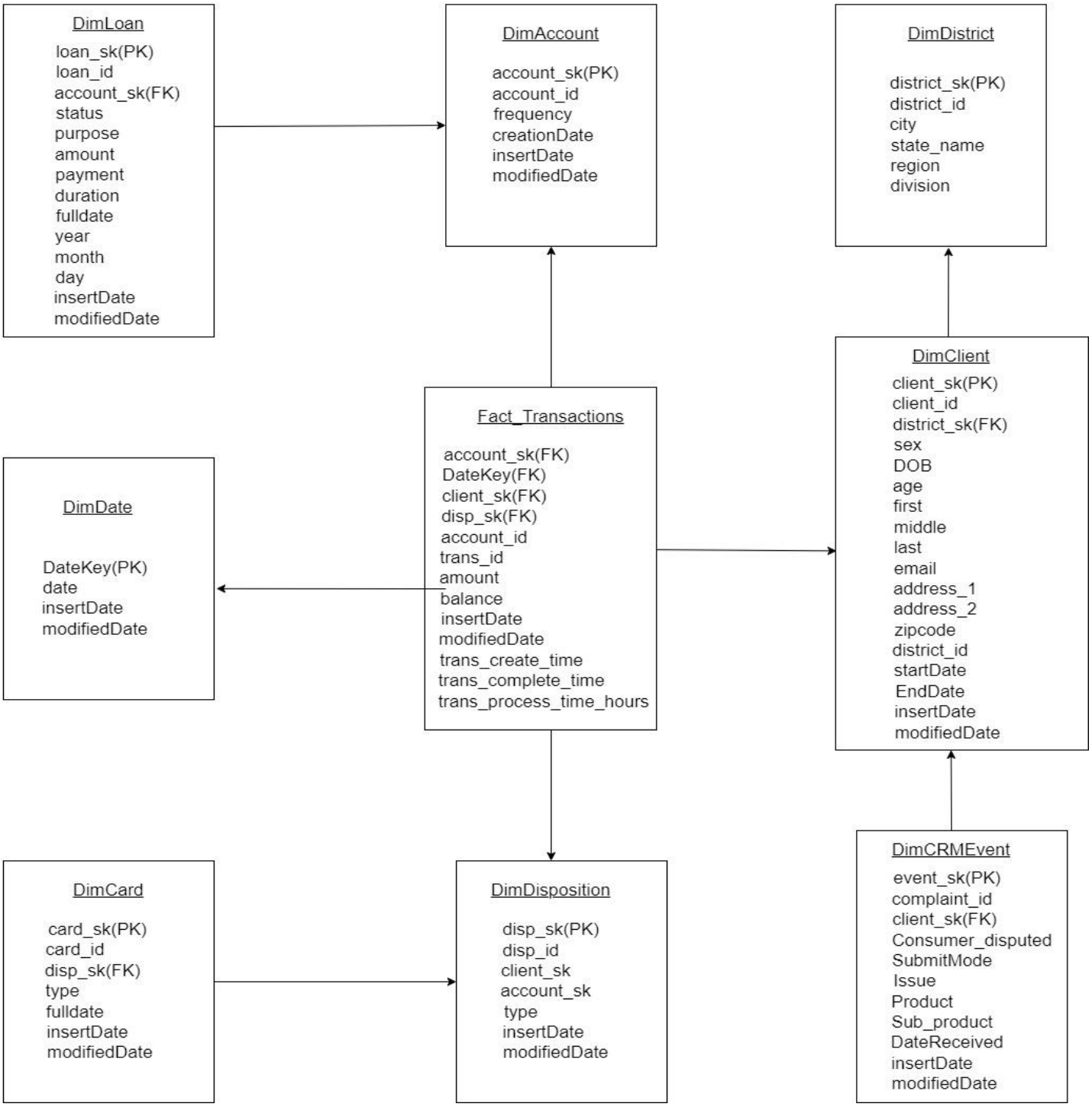
i. Account Details Staging - StgAccount
ii. Client Details Staging - StgClient
iii. Card Payment Details Staging - StgCard
iv. Loan Details Staging – StgLoan
v. Extra Loan Details Staging - StgLoanExtra
vi. Transaction Details Staging - StgTransaction
vii. Disposition Details Staging - StgDisposition
viii. District Details Staging – StgDistrict
ix. CRM Event Details Staging - StgCRMEvent

Next staged tables are profiled and aggregations are performed when necessary. As the next step data is transformed and loaded. After completing the described stages, the Data warehouse is created. (Dimension and Fact tables)

I. DimAccount
II. DimClient
III. DimCard
IV. DimLoan
V. DimDisposition
VI. DimDistrict
VII. DimDate
VIII. DimCRMEvent
IX. Fact_Transactions

BI results such as OLAP analysis, Reports, Data visualization,Data mining can be obtained as results by doing further modifications after the data warehouse is created.

3. Data warehouse design and development



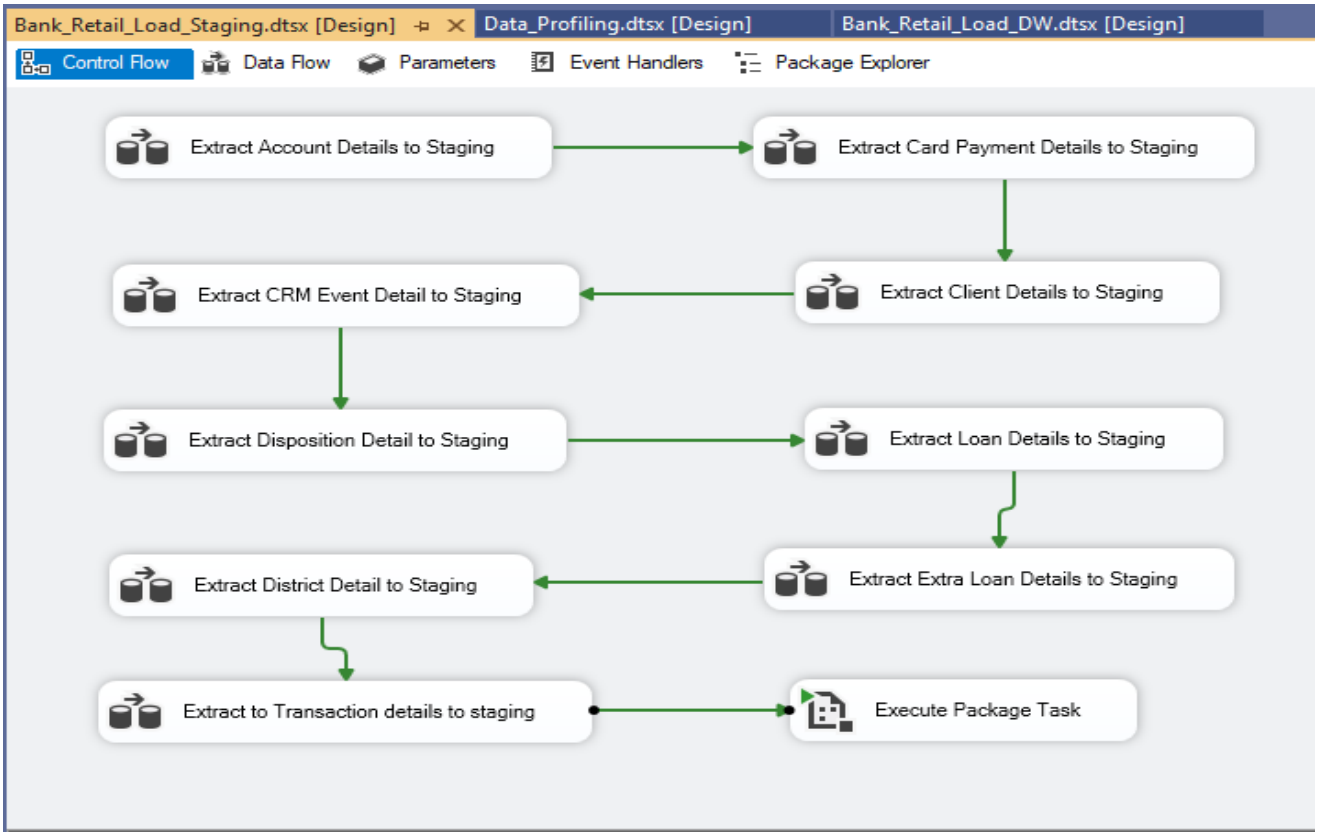
Snowflake schema is used to design the Data warehouse design. There is one fact table as transactions and 8 dimension tables. Also, the transactions per client was considered as the grain when designing.

Assumptions.

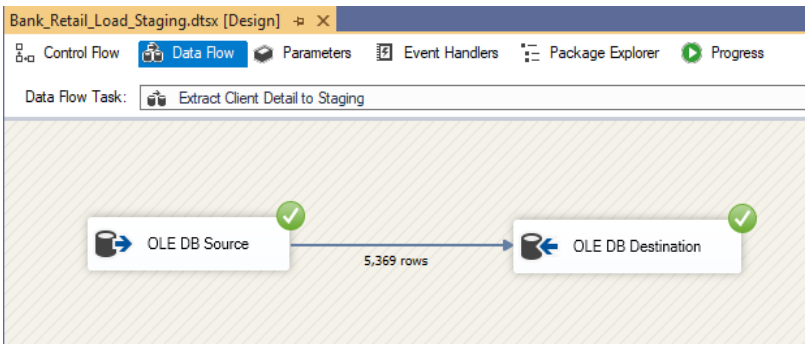
Client Details were considered as a slowly changing dimension.

4. ETL Development

The first step of ETL development process was to extract data from the sources (DB source & text file). For every extraction, data flow task was used and data was extracted from the source to the staging table. Then for every staging table a truncate table was created. All the data flowtasks were executed sequentially as shown below:



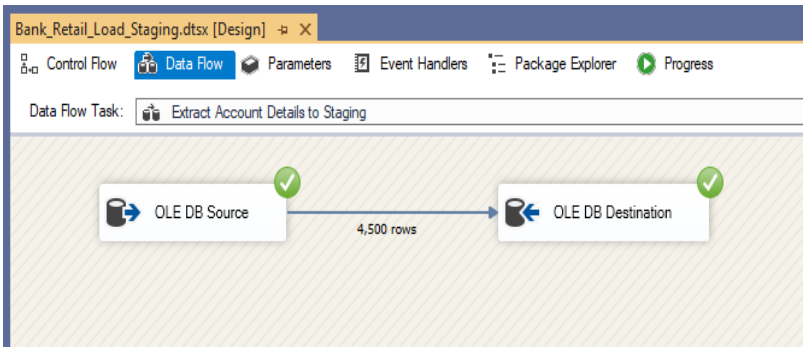
1.Staging client detail



Client Detail – Data is

5369 rows

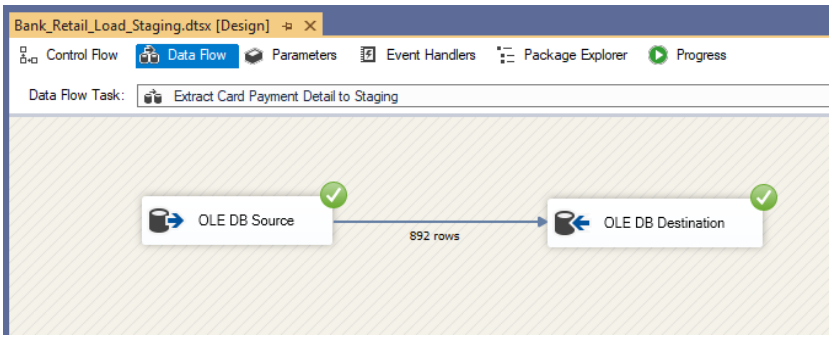
2. Staging Account details



Account Detail – Data is extracted from the account detail table in the source database and inserted to the account detail staging table (StgAccount).

4500 rows

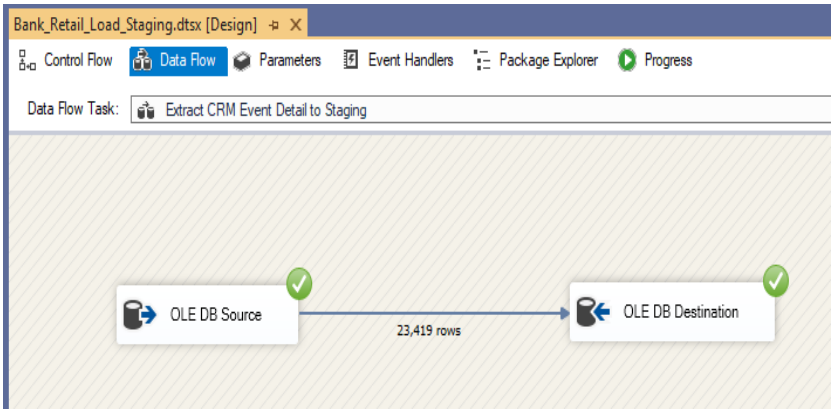
3. Staging Card Payment details



Card Payment Detail – Data is extracted from the card detail table in the source database and inserted to the card detail staging table (StgCard).

892 rows

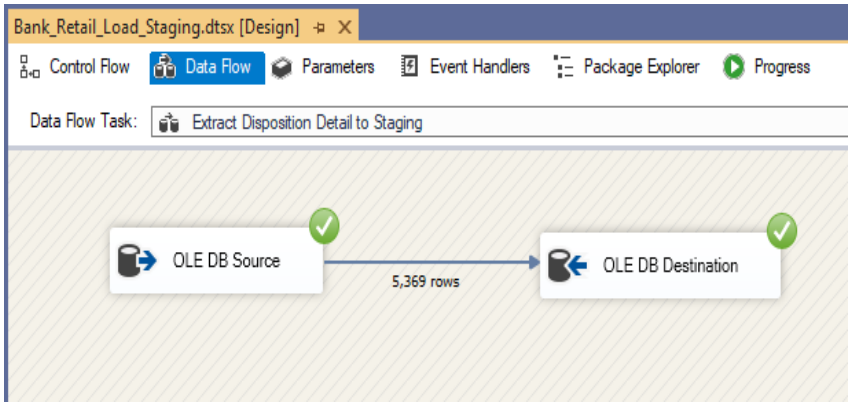
4. Staging CRM Event details



CRM Event Detail – Data is extracted from the event detail table in the source database and inserted to the event detail staging table (StgCRMEvent).

23 419 rows

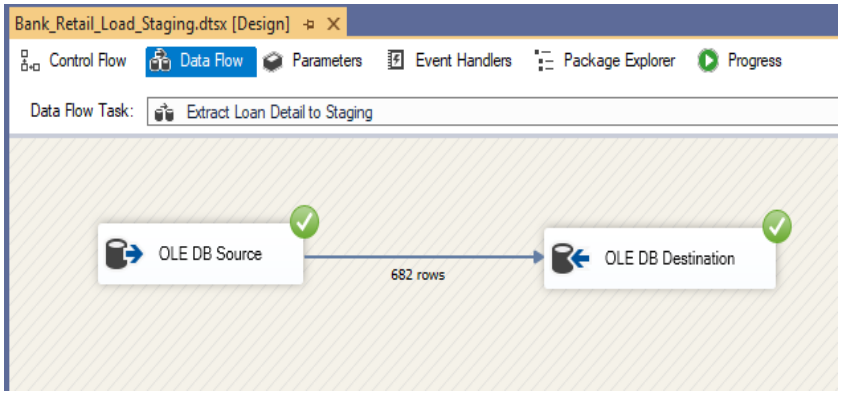
5. Staging disposition details



Disposition Detail – Data is extracted from the disposition detail table in the source database and inserted to the disposition detail staging table (StgDisposition).

5369 rows

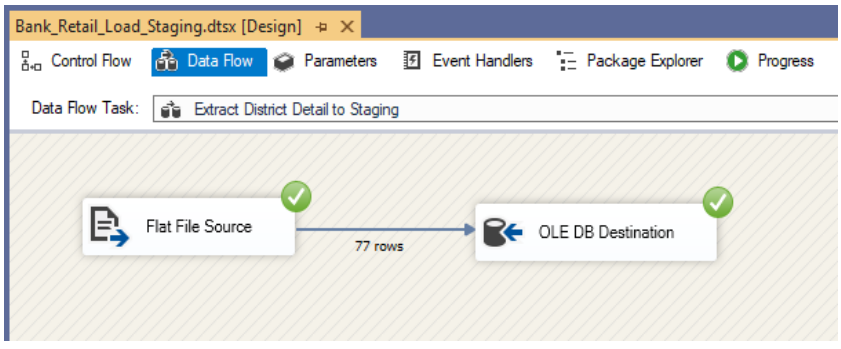
6. Staging Loan details



Loan Detail – Data is extracted from the loan detail table in the source database and inserted to the loan detail staging table (StgLoan).

682 rows

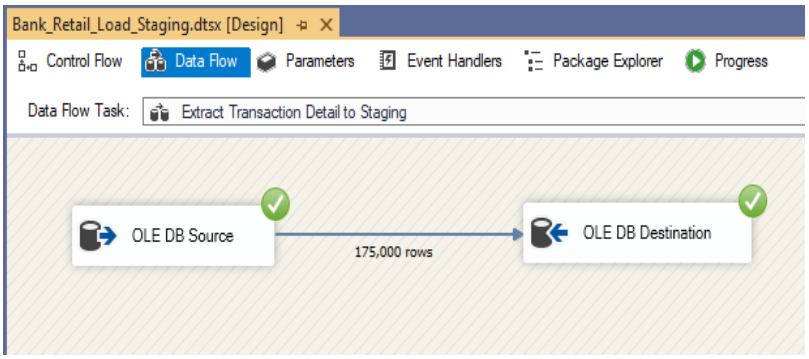
7.Staging District details



District Detail – Data is extracted from the district detail text file and inserted to the district detail staging table (StgDistrict).

77 rows

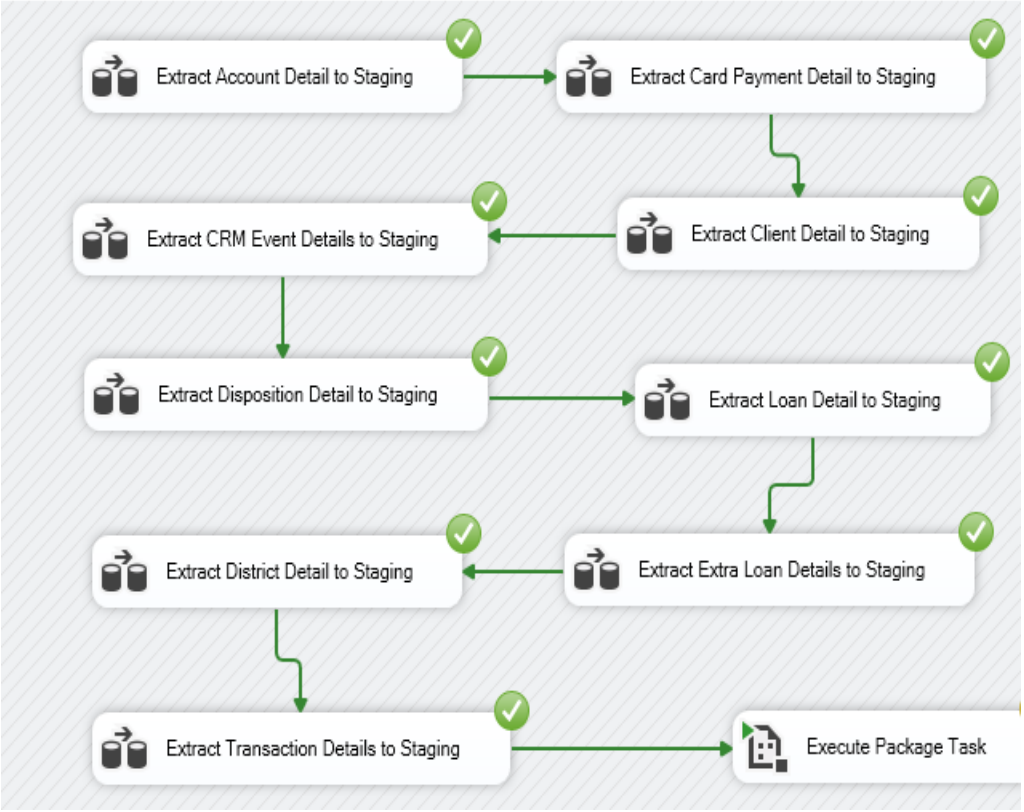
8. Staging Transaction details



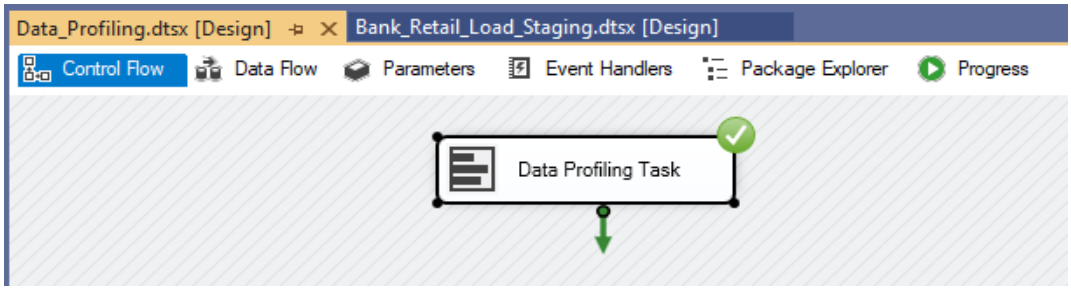
Transaction Detail – Data is extracted from the transaction detail table in the source database and inserted to the transaction detail staging table (StgTransaction).

175000 rows

Package executed successfully :

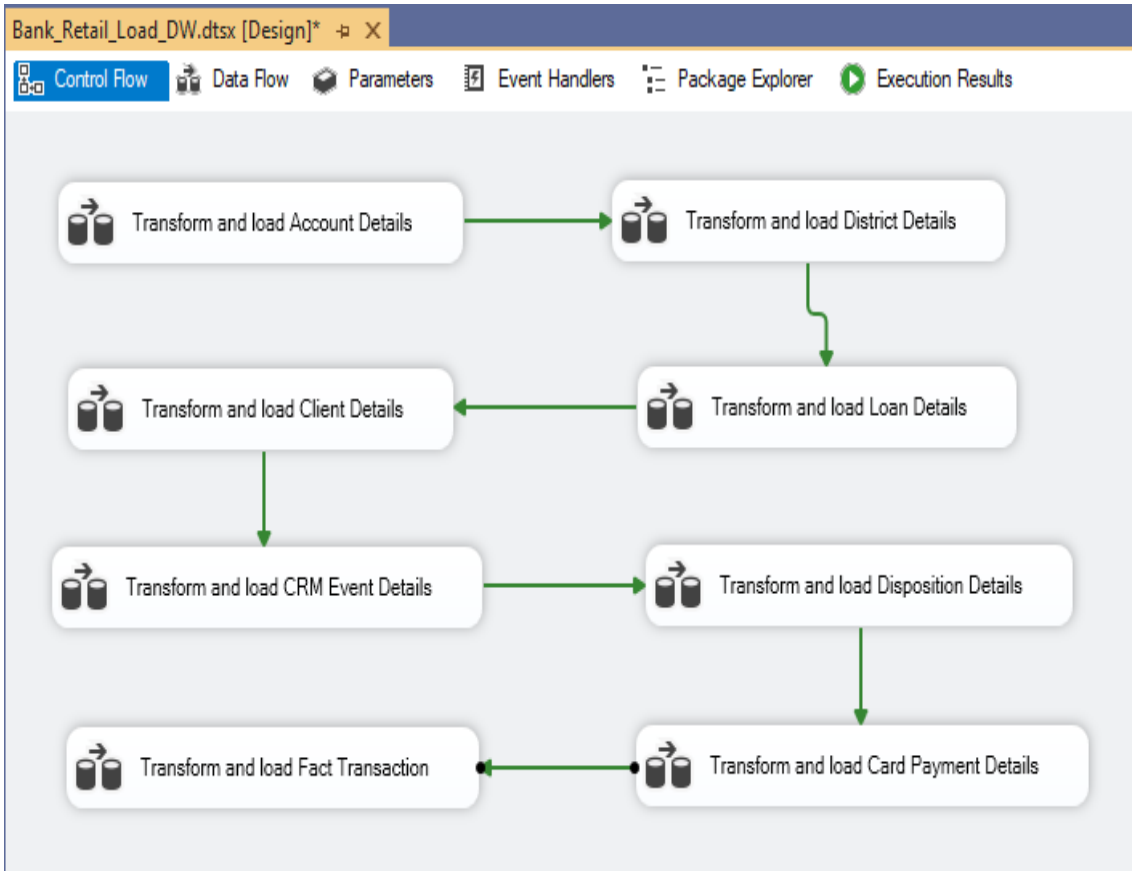


After that as the next step data profiling is done:

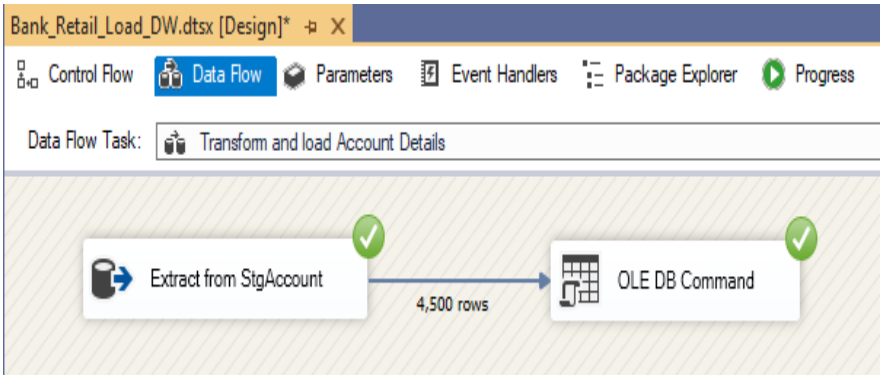


Data Transformation and loading

When data is transformed and loaded, the order of execution should be considered because of references related to the foreign keys. Therefore the order of execution is as below:



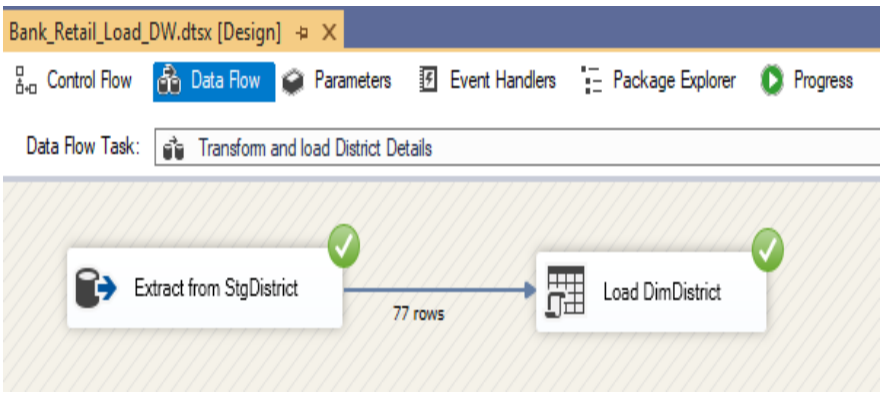
1. *Transform and load Account Details*



Account Detail – Data is extracted from the account staging table(StgAccount) and loaded to the account dimension table (DimAccount).

4500 rows

2. *Transform and load District Details*



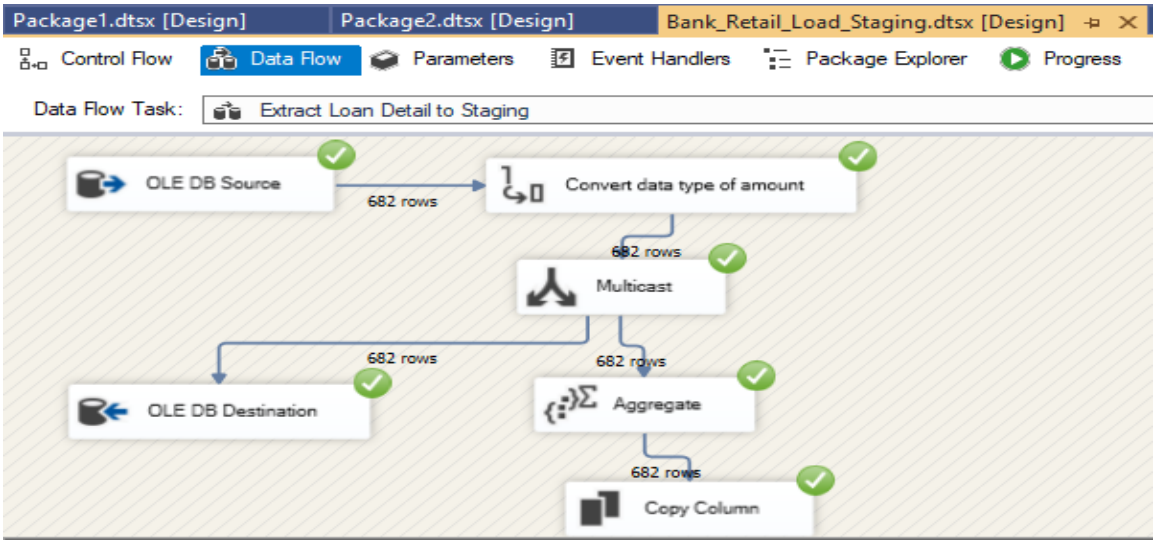
District Detail – Data is extracted from the district staging table(StgDistrict) and loaded to the district dimension table (DimDistrict).

77 rows

3. *Transform and load Loan Details*

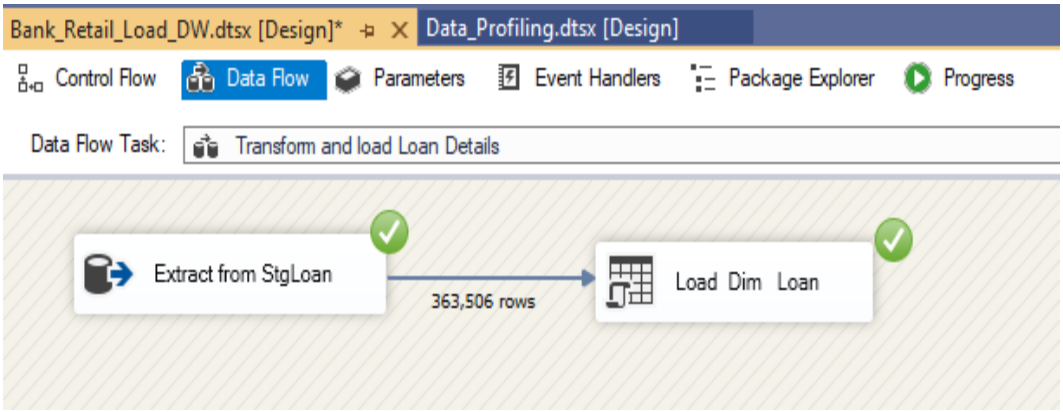
Loan details were loaded to DimLoan from 2 staging tables

- Load from loan staging



First of all data was extracted from loan staging and converted the data type of 'amount' column to money from nvarchar. Aggregation was done to check the number of loans obtained by each client by sorting according to account id and finally loaded to Loan dimension.

- Load from extra loan staging



4. *Transform and load Client Details*

Client details are considered as slowlychanging details.

Steps:-

1. Extracted from client staging table
2. Sorted by district id
3. Extracted from district dimension table
4. Sorted by AlternateDistrict id
5. Merge joined
6. Replaced null age column
7. Made into a slowly changing dimension
8. Loaded to client dimension

The below mentioned columns were set as changing attributes:

1. Age (client's age which is derived from birth date)

The below mentioned columns were set as historical attributes:

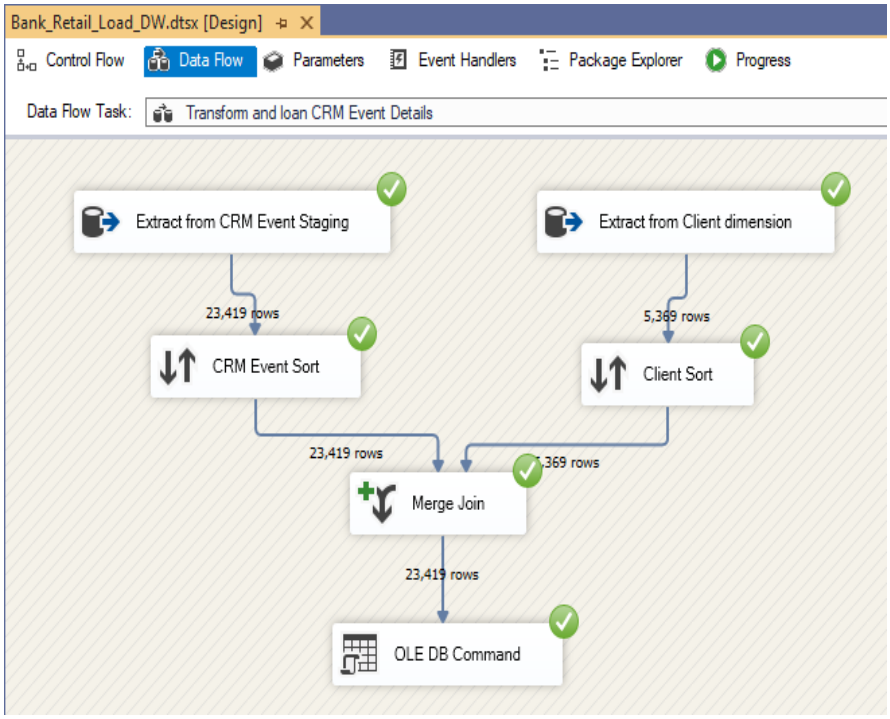
1. address_1
2. address_2
3. zipcode
4. district_id (district id corresponding to the relevant zipcode)

The remaining attributes were considered as fixed attributes:

1. client_id
2. DOB
3. sex
4. first
5. middle
6. last
7. email

After extracting data from the Client staging table, it was sorted according to the district id and also data was extracted from district dimension to get the district surrogate key as a foreign key and then it was identified as a slowly changing dimension. Finally the data was loaded to the Client dimension table.

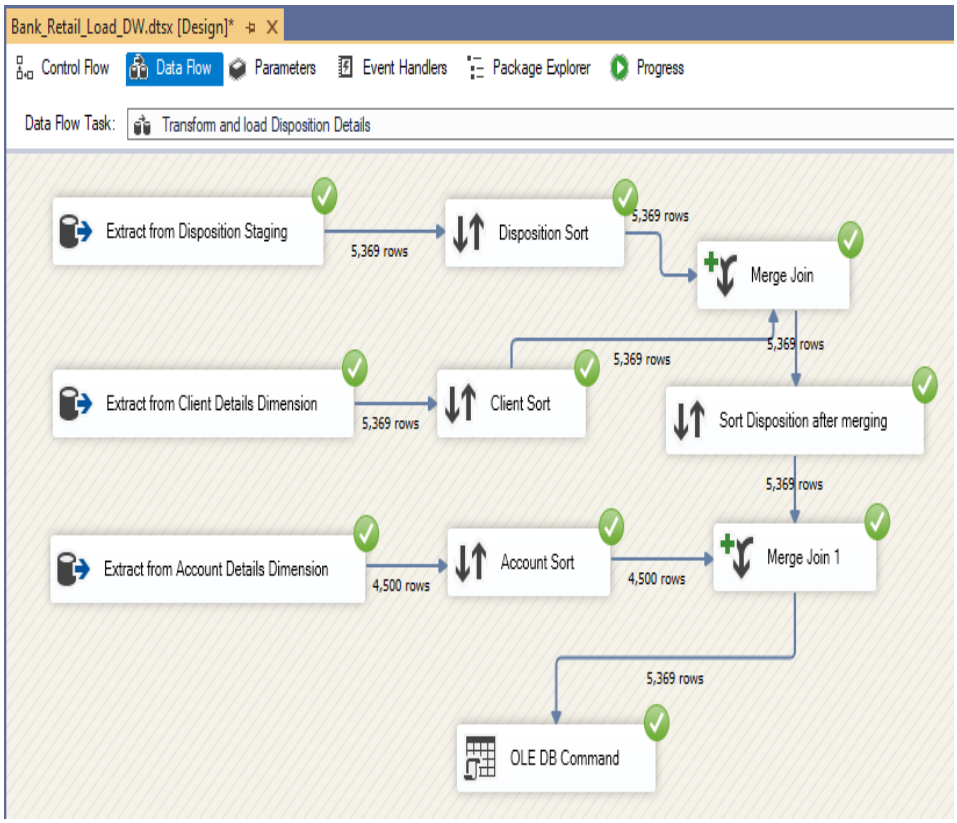
5. Transform and load CRM Event Details



1. Extracted from CRM Event Staging
2. Sorted according to client id
3. Extracted from Client dimension
4. Sorted according to AlternateClient id
5. Merged joined and loaded to DimCRMEvent

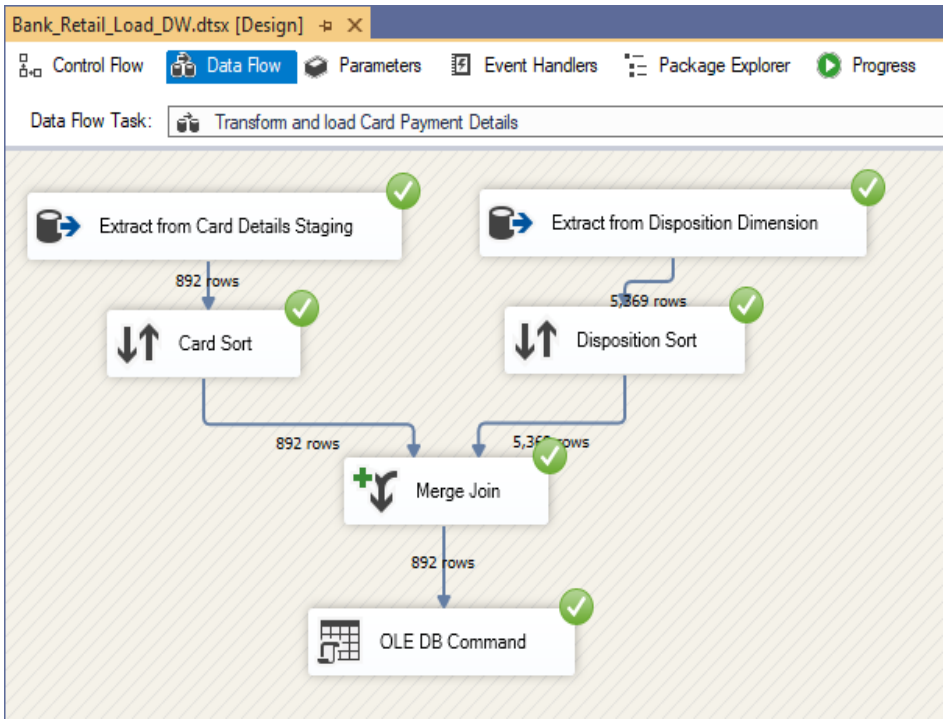
23419 rows

6. *Transform and load Disposition Details*



6. Extracted from Disposition Staging
7. Sorted according to client id and account id
8. Extracted from Client dimension
9. Sorted according to AlternateClient id
10. Extracted from Account dimension
11. Sorted according to AlternateAccount id
12. Merged joined and loaded to DimDisposition

7. *Transform and load Card Payment Details*



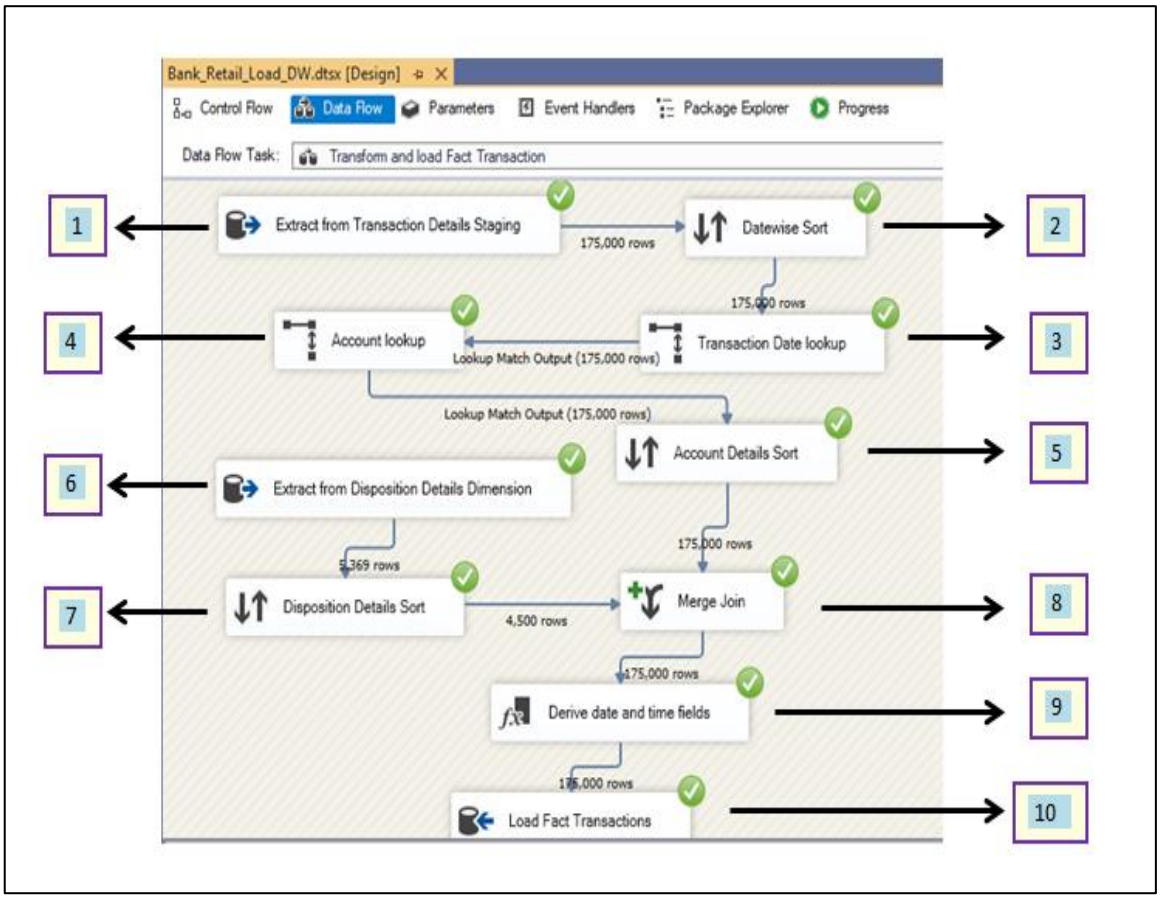
13. Extracted from Card Staging
14. Sorted according to disp id
15. Extracted from Disposition dimension
16. Sorted according to Alternatedisp id
17. Merged joined and loaded to DimCard

Procedures were linked to the OLE DB command and executed in order to load data. One of such procedure is attached below:

```
SQLQuery1.sql - not connected*

CREATE PROCEDURE dbo.updateDimCard
@card_id nvarchar(50),
@disp_id nvarchar(50),
@type nvarchar(50),
@fulldate datetime
AS BEGIN
if not exists (select card_sk from dbo.DimCard where AlternateCard_id = @card_id)
BEGIN
insert into dbo.DimCard
(AlternateCard_id,disp_sk,type,fulldate,insertDate,modifiedDate)
values
(@card_id,@disp_id,@type,@fulldate,GETDATE(),GETDATE()) END;
if exists (select card_sk from dbo.DimCard where AlternateCard_id = @card_id)
BEGIN
update dbo.DimCard
set disp_sk = @disp_id,type = @type,fulldate=@fulldate,modifiedDate = GETDATE()
where AlternateCard_id = @card_id END;
END;

100 %
Messages
Commands completed successfully.

Completion time: 2022-05-06T03:08:50.8687924-07:00
```

Procedure used to load data to DimCard

8. *Transform and load Fact Table*



After loading data to all dimensions, lastly data was loaded to the fact table. The below steps were followed:

1. Data extracted from the transaction details staging
2. Data sorted date wise
3. Performed lookup on date dimension
4. Performed lookup on account dimension
5. Sorted account details according to account surrogate key and transaction id
6. Data extracted from disposition details dimension
7. Sorted disposition details according to account surrogate key
8. Merge joined using account surrogate key
9. Derive insert date, modified date and transaction create time
10. Extracted and merged data are loaded to the Transaction Dimension

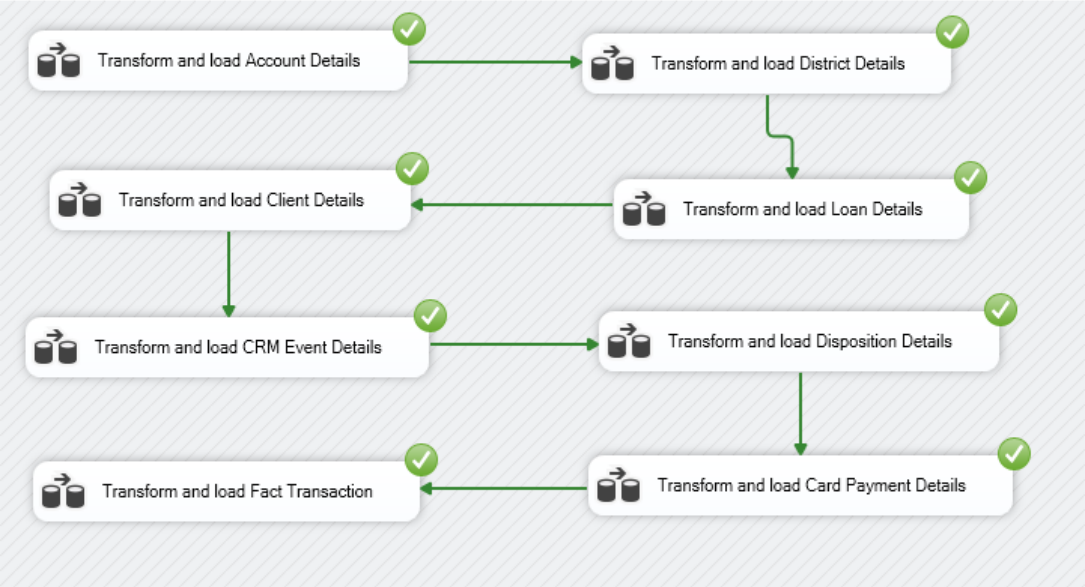
The query used to create the date dimension:

```
BEGIN TRY
        DROP TABLE [dbo].[DimDate]
END TRY
BEGIN CATCH
        /No Action/
END CATCH
/*****************************/
CREATE TABLE    [dbo].[DimDate]
        (       [DateKey] INT primary key,
                [Date] DATETIME,
                [FullDateUK] CHAR(10), -- Date in dd-MM-yyyy format
                [FullDateUSA] CHAR(10),-- Date in MM-dd-yyyy format
                [DayOfMonth] VARCHAR(2), -- Field will hold day number of Month
                [DaySuffix] VARCHAR(4), -- Apply suffix as 1st, 2nd ,3rd etc
                [DayName] VARCHAR(9), -- Contains name of the day, Sunday,
Monday
                [DayOfWeekUSA] CHAR(1),-- First Day Sunday=1 and Saturday=7
                [DayOfWeekUK] CHAR(1),-- First Day Monday=1 and Sunday=7
                [DayOfWeekInMonth] VARCHAR(2), --1st Monday or 2nd Monday in
Month
                [DayOfWeekInYear] VARCHAR(2),
                [DayOfQuarter] VARCHAR(3),
                [DayOfYear] VARCHAR(3),
                [WeekOfMonth] VARCHAR(1),-- Week Number of Month
                [WeekOfQuarter] VARCHAR(2), --Week Number of the Quarter
                [WeekOfYear] VARCHAR(2),--Week Number of the Year
                [Month] VARCHAR(2), --Number of the Month 1 to 12
                [MonthName] VARCHAR(9),--January, February etc
                [MonthOfQuarter] VARCHAR(2),-- Month Number belongs to Quarter
                [Quarter] CHAR(1),
                [QuarterName]  VARCHAR(9),--First,Second..
                [Year] CHAR(4),-- Year value of Date stored in Row
                [YearName] CHAR(7), --CY 2012,CY 2013
                [MonthYear] CHAR(10), --Jan-2013,Feb-2013
                [MMYYYY] CHAR(6),
                [FirstDayOfMonth] DATE,
                [LastDayOfMonth] DATE,
                [FirstDayOfQuarter] DATE,
                [LastDayOfQuarter] DATE,
                [FirstDayOfYear] DATE,
                [LastDayOfYear] DATE,
                [IsHolidaySL] BIT,-- Flag 1=National Holiday, 0-No National Holiday
                [IsWeekday] BIT,-- 0=Week End ,1=Week Day
                [HolidaySL] VARCHAR(50),--Name of Holiday in US
                [isCurrentDay] int, -- Current day=1 else = 0
                [isDataAvailable] int, -- data available for the day = 1, no data available
for the day = 0
                [isLatestDataAvailable] int
        )
GO
/*******************************/
--Specify Start Date and End date here
--Value of Start Date Must be Less than Your End Date
DECLARE @StartDate DATETIME = '01/01/1990' –Starting value of Date Range
DECLARE @EndDate DATETIME = '01/01/2099' –End Value of Date Range
--Temporary Variables To Hold the Values During Processing of Each Date of Year
DECLARE
        @DayOfWeekInMonth INT,
        @DayOfWeekInYear INT,
        @DayOfQuarter INT,
        @WeekOfMonth INT,
        @CurrentYear INT,
        @CurrentMonth INT,
        @CurrentQuarter INT
--Proceed only if Start Date(Current date ) is less than End date you specified above
WHILE @CurrentDate < @EndDate
BEGIN
/Begin day of week logic/
     /*Check for Change in Month of the Current date if Month changed then
      Change variable value*/
        IF @CurrentMonth != DATEPART(MM, @CurrentDate)
        BEGIN
                UPDATE @DayOfWeek
                SET MonthCount = 0
                SET @CurrentMonth = DATEPART(MM, @CurrentDate)
     -- Set values in table data type created above from variables

        UPDATE @DayOfWeek
        SET
                MonthCount = MonthCount + 1,
                QuarterCount = QuarterCount + 1,
                YearCount = YearCount + 1
        WHERE DOW = DATEPART(DW, @CurrentDate)

        SELECT
                @DayOfWeekInMonth = MonthCount,
                @DayOfQuarter = QuarterCount,

@DayOfWeekInYear = YearCount
        FROM @DayOfWeek
        WHERE DOW = DATEPART(DW, @CurrentDate)
/End day of week logic/
/* Populate Your Dimension Table with values*/
        INSERT INTO [dbo].[DimDate]
        SELECT
                CONVERT (char(8),@CurrentDate,112) as DateKey,
                @CurrentDate AS Date,
                CONVERT (char(10),@CurrentDate,103) as FullDateUK,
                CONVERT (char(10),@CurrentDate,101) as FullDateUSA,
                DATEPART(DD, @CurrentDate) AS DayOfMonth,
                --Apply Suffix values like 1st, 2nd 3rd etc..
                CASE
                        WHEN DATEPART(DD,@CurrentDate) IN (11,12,13)
                        THEN CAST(DATEPART(DD,@CurrentDate) AS VARCHAR)
+ 'th'
                        WHEN RIGHT(DATEPART(DD,@CurrentDate),1) = 1
                        THEN CAST(DATEPART(DD,@CurrentDate) AS VARCHAR)
+ 'st'
                        WHEN RIGHT(DATEPART(DD,@CurrentDate),1) = 2
                        THEN CAST(DATEPART(DD,@CurrentDate) AS VARCHAR)
+ 'nd'
                        WHEN RIGHT(DATEPART(DD,@CurrentDate),1) = 3
                        THEN CAST(DATEPART(DD,@CurrentDate) AS VARCHAR)
+ 'rd'
                        ELSE CAST(DATEPART(DD,@CurrentDate) AS VARCHAR) +
'th'
                        END AS DaySuffix,

                DATENAME(DW, @CurrentDate) AS DayName,
                DATEPART(DW, @CurrentDate) AS DayOfWeekUSA,

                -- check for day of week as Per US and change it as per UK format
                CASE DATEPART(DW, @CurrentDate)
                        WHEN 1 THEN 7
                        WHEN 2 THEN 1
                        WHEN 3 THEN 2
                        WHEN 4 THEN 3
                        WHEN 5 THEN 4
                        WHEN 6 THEN 5
                        WHEN 7 THEN 6
                        END
                        AS DayOfWeekUK,

                @DayOfWeekInMonth AS DayOfWeekInMonth,
                @DayOfWeekInYear AS DayOfWeekInYear,
                @DayOfQuarter AS DayOfQuarter,
                DATEPART(DY, @CurrentDate) AS DayOfYear,
                DATEPART(WW, @CurrentDate) + 1 – DATEPART(WW,
CONVERT(VARCHAR,
                DATEPART(MM, @CurrentDate)) + '/1/' + CONVERT(VARCHAR,
                DATEPART(YY, @CurrentDate))) AS WeekOfMonth,
                (DATEDIFF(DD, DATEADD(QQ, DATEDIFF(QQ, 0, @CurrentDate), 0),
                @CurrentDate) / 7) + 1 AS WeekOfQuarter,
                DATEPART(WW, @CurrentDate) AS WeekOfYear,
                DATEPART(MM, @CurrentDate) AS Month,
                DATENAME(MM, @CurrentDate) AS MonthName,
                CASE
                        WHEN DATEPART(MM, @CurrentDate) IN (1, 4, 7, 10)
THEN 1
```

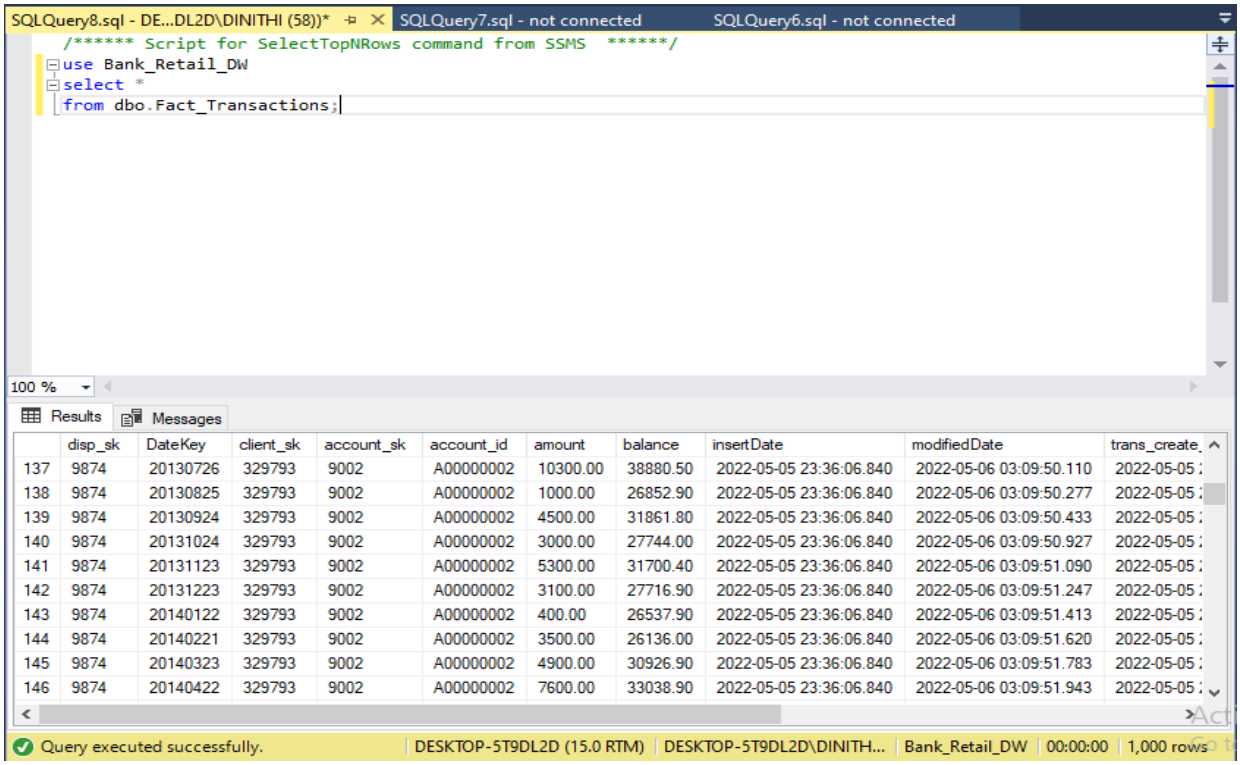
```
		END

	/* Check for Change in Quarter of the Current date if Quarter changed then change
	 Variable value*/

		IF @CurrentQuarter != DATEPART(QQ, @CurrentDate)
		BEGIN
			UPDATE @DayOfWeek
			SET QuarterCount = 0
			SET @CurrentQuarter = DATEPART(QQ, @CurrentDate)
		END

	/* Check for Change in Year of the Current date if Year changed then change
	 Variable value*/


		IF @CurrentYear != DATEPART(YY, @CurrentDate)
		BEGIN
			UPDATE @DayOfWeek
			SET YearCount = 0
			SET @CurrentYear = DATEPART(YY, @CurrentDate)
		END
/Table Data type to store the day of week count for the month and year/
DECLARE @DayOfWeek TABLE (DOW INT, MonthCount INT, QuarterCount INT, YearCount INT)
INSERT INTO @DayOfWeek VALUES (1, 0, 0, 0)
INSERT INTO @DayOfWeek VALUES (2, 0, 0, 0)
INSERT INTO @DayOfWeek VALUES (3, 0, 0, 0)
INSERT INTO @DayOfWeek VALUES (4, 0, 0, 0)
INSERT INTO @DayOfWeek VALUES (5, 0, 0, 0)
INSERT INTO @DayOfWeek VALUES (6, 0, 0, 0)
INSERT INTO @DayOfWeek VALUES (7, 0, 0, 0)
--Extract and assign various parts of Values from Current Date to Variable
DECLARE @CurrentDate AS DATETIME = @StartDate
SET @CurrentMonth = DATEPART(MM, @CurrentDate)
SET @CurrentYear = DATEPART(YY, @CurrentDate)
SET @CurrentQuarter = DATEPART(QQ, @CurrentDate)
/*********************************/
```

```
		WHEN DATEPART(MM, @CurrentDate) IN (2, 5, 8, 11)
THEN 2
		WHEN DATEPART(MM, @CurrentDate) IN (3, 6, 9, 12)
THEN 3
		END AS MonthOfQuarter,
	DATEPART(QQ, @CurrentDate) AS Quarter,
	CASE DATEPART(QQ, @CurrentDate)
		WHEN 1 THEN 'First'
		WHEN 2 THEN 'Second'
		WHEN 3 THEN 'Third'
		WHEN 4 THEN 'Fourth'
		END AS QuarterName,
	DATEPART(YEAR, @CurrentDate) AS Year,
	'CY ' + CONVERT(VARCHAR, DATEPART(YEAR, @CurrentDate)) AS
YearName,
	LEFT(DATENAME(MM, @CurrentDate), 3) + '-' + CONVERT(VARCHAR,
	DATEPART(YY, @CurrentDate)) AS MonthYear,
	RIGHT('0' + CONVERT(VARCHAR, DATEPART(MM, @CurrentDate)),2) +
	CONVERT(VARCHAR, DATEPART(YY, @CurrentDate)) AS MMYYYY,
	CONVERT(DATETIME, CONVERT(DATE, DATEADD(DD, -
(DATEPART(DD,
	@CurrentDate) – 1), @CurrentDate))) AS FirstDayOfMonth,
	CONVERT(DATETIME, CONVERT(DATE, DATEADD(DD, -
(DATEPART(DD,
	(DATEADD(MM, 1, @CurrentDate)))), DATEADD(MM, 1,
	@CurrentDate)))) AS LastDayOfMonth,
	DATEADD(QQ, DATEDIFF(QQ, 0, @CurrentDate), 0) AS
FirstDayOfQuarter,
	DATEADD(QQ, DATEDIFF(QQ, -1, @CurrentDate), -1) AS
LastDayOfQuarter,
	CONVERT(DATETIME, '01/01/' + CONVERT(VARCHAR, DATEPART(YY,
	@CurrentDate))) AS FirstDayOfYear,
	CONVERT(DATETIME, '12/31/' + CONVERT(VARCHAR, DATEPART(YY,
	@CurrentDate))) AS LastDayOfYear,
	NULL AS IsHolidaySL,
	CASE DATEPART(DW, @CurrentDate)
		WHEN 1 THEN 0
		WHEN 2 THEN 1
		WHEN 3 THEN 1
		WHEN 4 THEN 1
		WHEN 5 THEN 1
		WHEN 6 THEN 1
		WHEN 7 THEN 0
		END AS IsWeekday,
	NULL AS HolidaySL, (case when @CurrentDate = convert(date,
sysdatetime()) then 1 else 0 end), 0, 0

		SET @CurrentDate = DATEADD(DD, 1, @CurrentDate)
END
/*********************************/
/*********************************/
SELECT * FROM [dbo].[DimDate]
```

After loading data to all the dimensions and the fact table:



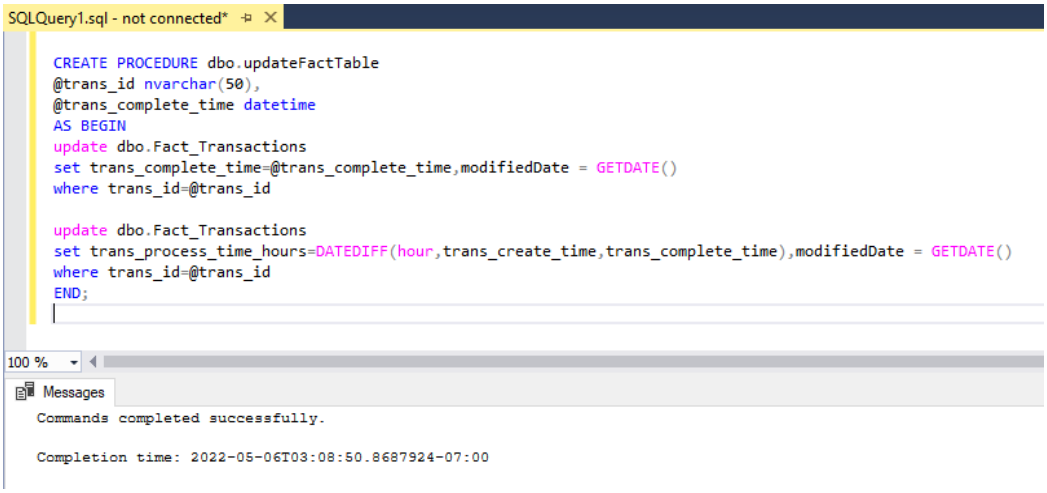
A print screen of the fact table:



	disp_sk	DateKey	client_sk	account_sk	account_id	amount	balance	insertDate	modifiedDate	trans_create_
137	9874	20130726	329793	9002	A00000002	10300.00	38880.50	2022-05-05 23:36:06.840	2022-05-06 03:09:50.110	2022-05-05 :
138	9874	20130825	329793	9002	A00000002	1000.00	26852.90	2022-05-05 23:36:06.840	2022-05-06 03:09:50.277	2022-05-05 :
139	9874	20130924	329793	9002	A00000002	4500.00	31861.80	2022-05-05 23:36:06.840	2022-05-06 03:09:50.433	2022-05-05 :
140	9874	20131024	329793	9002	A00000002	3000.00	27744.00	2022-05-05 23:36:06.840	2022-05-06 03:09:50.927	2022-05-05 :
141	9874	20131123	329793	9002	A00000002	5300.00	31700.40	2022-05-05 23:36:06.840	2022-05-06 03:09:51.090	2022-05-05 :
142	9874	20131223	329793	9002	A00000002	3100.00	27716.90	2022-05-05 23:36:06.840	2022-05-06 03:09:51.247	2022-05-05 :
143	9874	20140122	329793	9002	A00000002	400.00	26537.90	2022-05-05 23:36:06.840	2022-05-06 03:09:51.413	2022-05-05 :
144	9874	20140221	329793	9002	A00000002	3500.00	26136.00	2022-05-05 23:36:06.840	2022-05-06 03:09:51.620	2022-05-05 :
145	9874	20140323	329793	9002	A00000002	4900.00	30926.90	2022-05-05 23:36:06.840	2022-05-06 03:09:51.783	2022-05-05 :
146	9874	20140422	329793	9002	A00000002	7600.00	33038.90	2022-05-05 23:36:06.840	2022-05-06 03:09:51.943	2022-05-05 :

5. ETL Development – Accumulating Fact tables

- In order to update transaction complete time column in fact transactions table a separate CSV file was used as the source.
- A procedure was linked to the OLE DB command and executed in order to update 'trans_complete_time' column and 'trans_process_tome_hours' column.
- The procedure which was used is attached below:



Fact_Transactions table was successfully updated by following the below steps:

1. Data was extracted from a csv file containing transaction id and transaction complete time
2. A procedure was linked to the OLE DB command and executed in order to update Fact_Transactions table