Logo, company name

Description automatically generated

**Data Warehouse & Business Intelligence**

Assignment I

2021

Submitted By: Subasinghe .S.S

IT20273712

Contents

[Data set selection & Preparation 1](#_bookmark0)

[Solution Architecture 4](#_bookmark1)

[Data warehouse design and development 5](#_bookmark2)

[Test Planning and Test Data 6](#_bookmark3)

[ETL Development 16](#_bookmark4)

[Execution of Test Cases and TSR 27](#_bookmark5)

# Data set selection & Introduction

A collection of transactional data is used as the data source. The following is a link to the original data set:

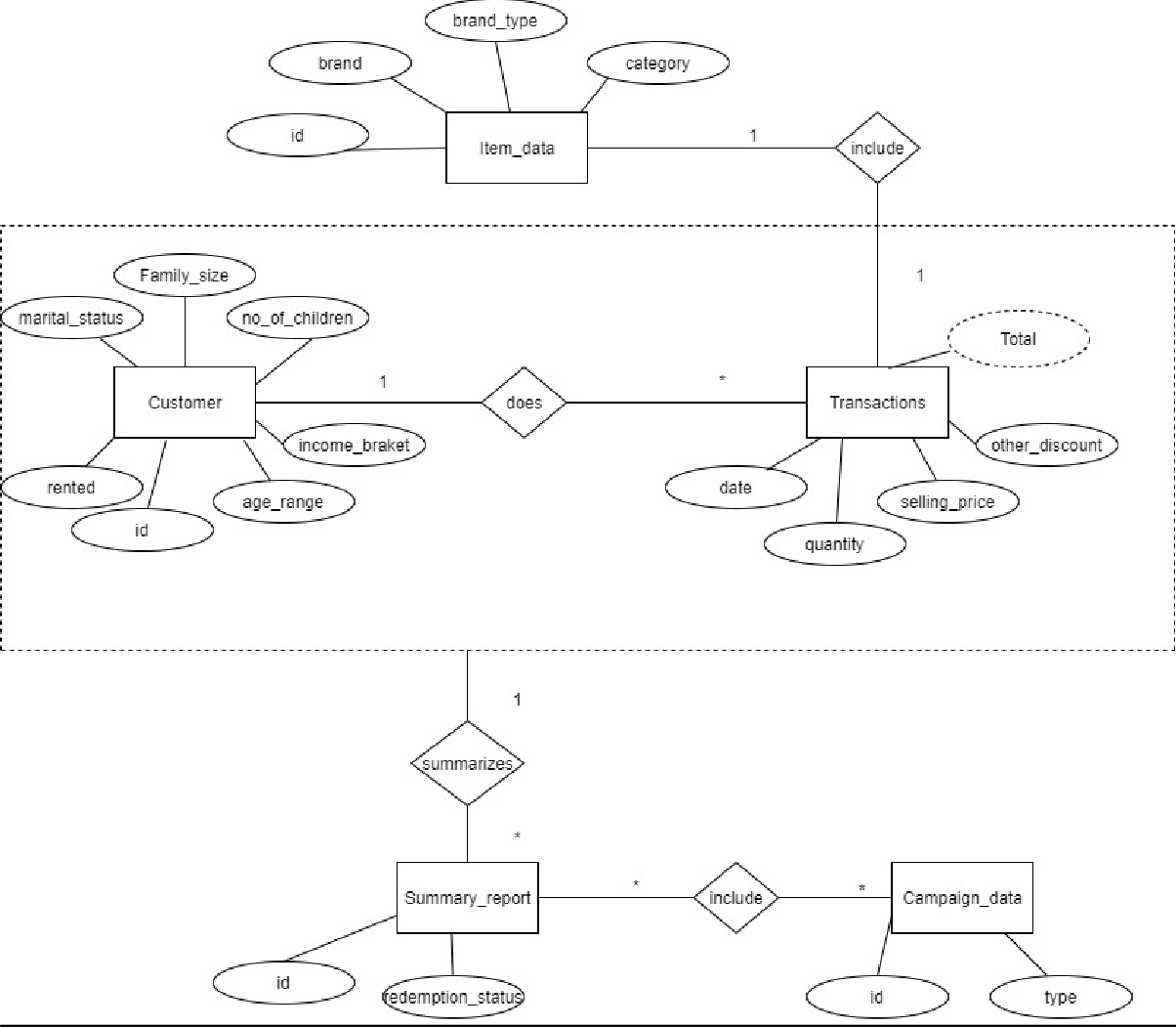
<https://www.kaggle.com/datasets/cityofLA/la-restaurant-market-health-data?select=restaurant-and-market-health-violations.csv>

Modifications were done accordingly to the data set derived from the source This data set reflects .

Description of the data set:

|  |  |  |  |
| --- | --- | --- | --- |
| **Table name** | **Column name** | **Data type** | **Description** |
| Train | id | int | Summary of the promotions held and customers participated |
| campaign\_id | numeric(18,0) |
| coupon\_id | int |
| customer\_id | int |
| redemption\_status | varchar(50) |
| CampaignData | campaign\_id | numeric(18,0) | Detals of promotion campaigns |
| campaign\_type | nvarchar(50) |
| CustomerDemographics | customer\_id | int | Customer Details |
| age\_range | varchar(50) |
| marital\_status | varchar(50) |
| rented | varchar(50) |
| family\_size | varchar(50) |
| no\_of\_children | varchar(50) |
| income\_bracket | int |
| ItemData | item\_id | int | Details of Items sold by the retailer |
| brand | varchar(50) |
| brand\_type | varchar(50) |
| category | varchar(50) |
| CustomerTransactionData | date | datetime | Details of the customer transactions |
| customer\_id | int |
| item\_id | int |
| quantity | int |
| selling\_price | float |
| other\_discount | float |
| coupon\_discount | float |

## ER Diagram



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

# Preparation of Data Sources

The two main sources are listed below:

SQL Database

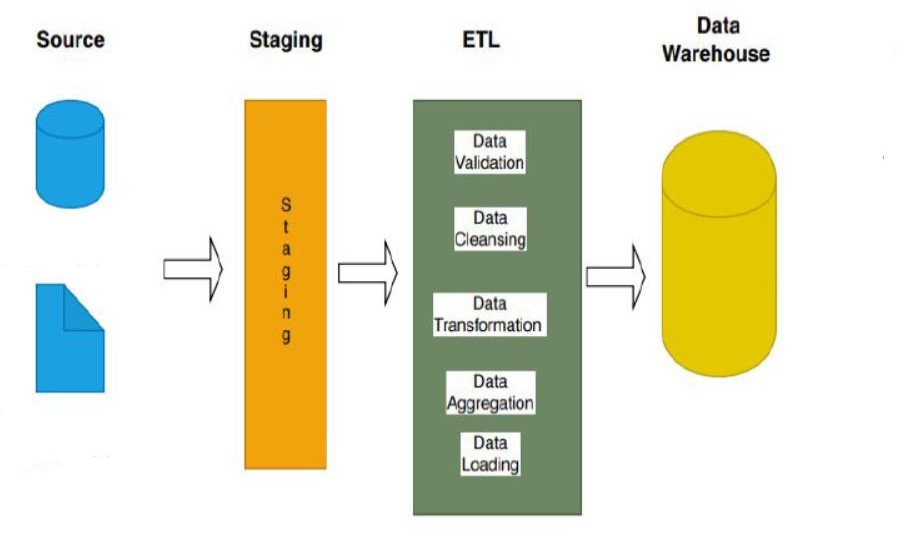
One text file – Campaign Data

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

Customer Details Item Data Summary Data

## Class Diagram using Data sources

# Solution Architecture



\_campaindata.txt

Creditcardsource DB

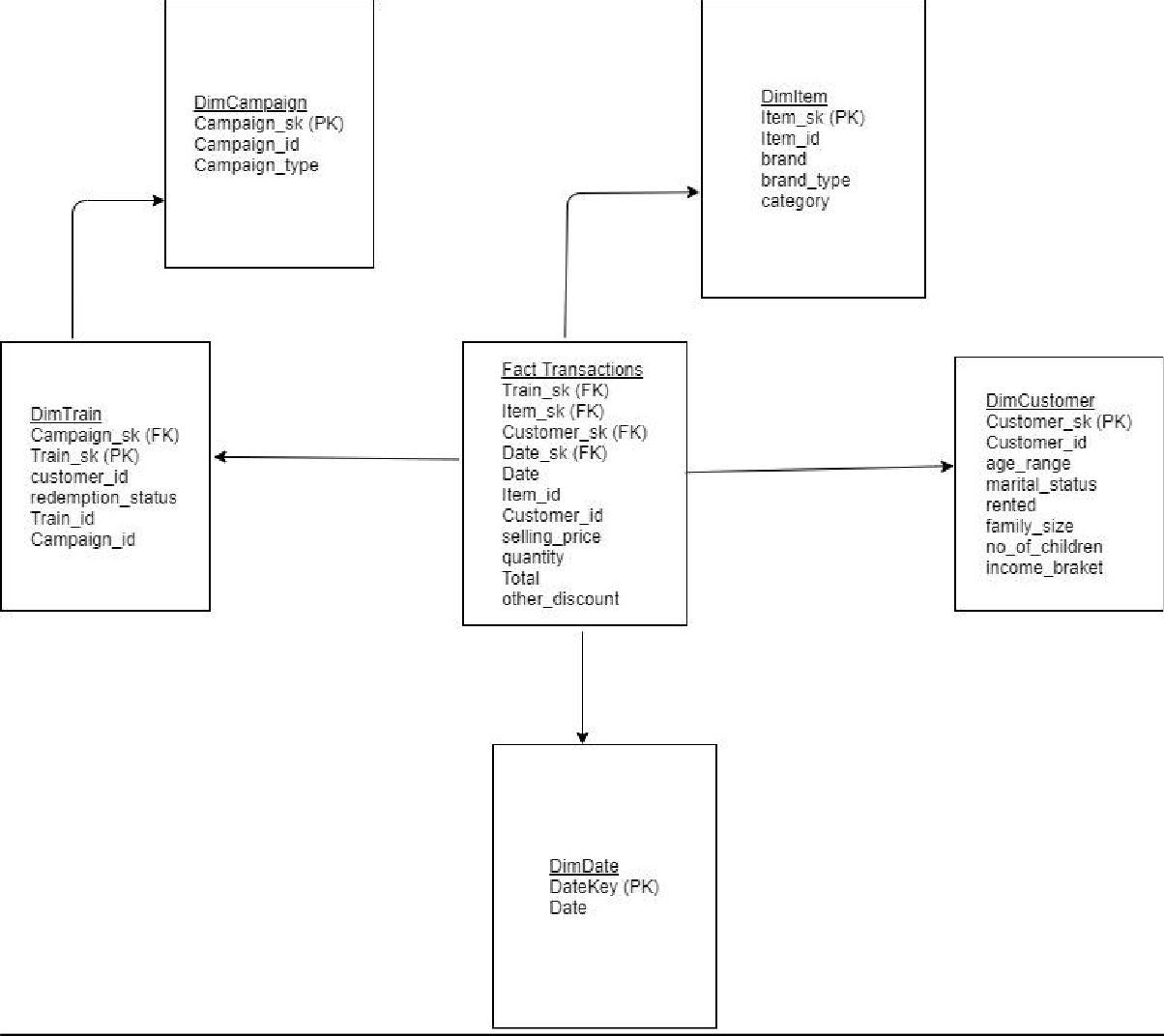
As explained First step is staging the source data set. After the staging layer the below mentioned staging tables are created:

1. Train Staging
2. Campaign Staging
3. Item Staging
4. Customer Demo Staging
5. Customer Transaction Staging

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, data is tested and validated and the Datawarehouse is created.

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

# Data warehouse design and development



Snowflake schema is used to design the Datawarehouse design. There is one fact table as transactions and 4 dimensions. Also, the transactions per customer was considered as the grain when designing.

Assumptions.

Customer Details were considered as a slowly changing dimension.

# Test Planning and Test Data

Testing is done to ensure that the data that has been loaded from source to the destination after the business transformation is accurate. It also involves verification of data at various middle stages that are being used between source and destination.

As this project contains two stages as mentioned below data was tested in both stages

1. Source to Staging (Middle stage)
2. Staging to Dimension tables (Final destination) Test Plan

|  |  |
| --- | --- |
| Scope | 1. Completeness of the data set testing   To conduct test cases to ensure that there are no data losses and that data is loaded completely   1. Data length testing   To make sure the data lengths tally when data is passed from source to middle stages as well as destination tables   1. Data type testing   Data types to be tested in order to refrain the process being interrupted due to data types as this is a common issue.   1. Data duplicity testing   To make sure quality of data is maintained and the data is not getting duplicated in the end to end process |
| Out of scope | Validity of data testing |
| Assumptions | There is no environment downtime during testing |
| Schedules | Start Date – 25/04/2020  End Date – 29/04/2020 |
| Roles and responsibilities | Natasha Amarasinghe   1. Create Test Plan 2. Create test cases 3. Execute test cases 4. Create Test Summary Report |
| Test Deliverables | 1. Test Plan 2. Test Cases and Test Results 3. Test Summary Report |
| Test Environment | Database Server: SQL Server Management Studio  Operating system: Windows 10 |
| Test tools | Microsoft SQL Server Data Tools for Visual Studio 2017 |

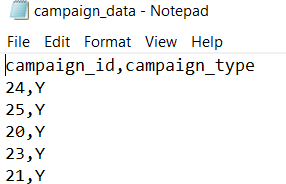
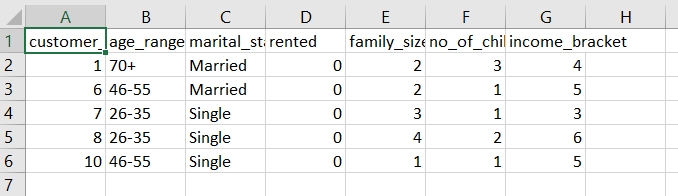
* All the execution of the test cases, snapshots and SQL queries are attached and described under **Execution of test Cases and TSR section** for the below listed test cases.

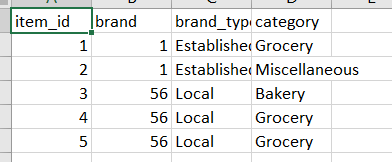
|  |  |
| --- | --- |
| 03 | Check for the count when transforming data from source to staging tables |
| 04 | Check for the count when transforming data from staging to dimension  tables |
| 05 | Check for duplicate values in the staging tables. |
| 06 | Check for duplicate values in the dimension tables. |
| 07 | Data length check for data in staging tables |
| 08 | Data length check for data in dimension tables |
| 09 | Data type check for data in staging tables |
| 10 | Data type check for data in dimension tables |

Test Data Set

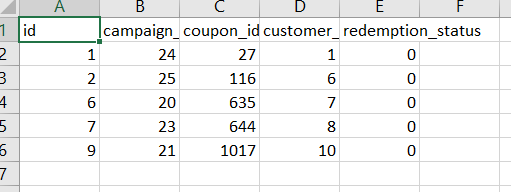
Before the development of the real data set and execution of the test cases, a small data set was derived from the Source and used for testing purposes to rectify issues in the process and to mitigate issues. The test data was loaded the same way as planned and tested in the below mentioned manner.

Please find below the mini data set used for testing purposes.

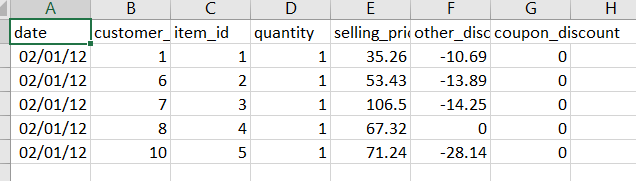
* Campaign Data Set
* Customer Data Set
* Item Data set



* Train Data set



* Transaction Data set

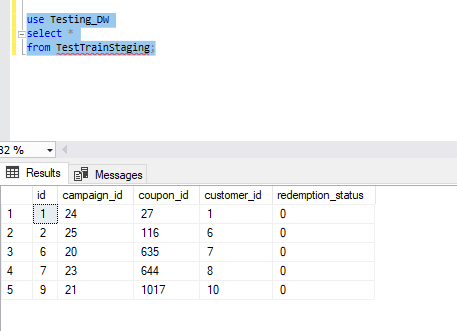


Test Data was imported to a separate test Database and was first staged, profiled and then transformed.

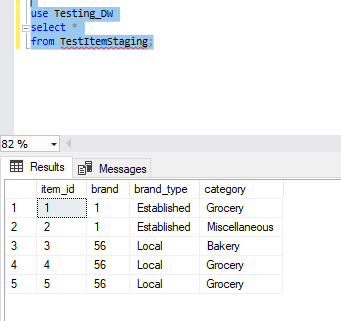
### Testing test data loaded from source to staging

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Scenario ID** | | 1 | | | | | |
| **Test Case**  **Description** | | Transform test data from source to staging tables | | | | | |
| **Pre-Requisite** | | Test Data loaded from source to staging tables in SQL tool | | | | | |
| **S.N**  **o** | **Action** | | **Sql Query** | **Expected Output** | **Actual Output** | **Test**  **Resul t** | **Test Comments** |
| 1 | Data passed from train source to Train  Staging | | use Test\_DW select \*  from TestTrainStaging; | All 5 rows displayed according  ly | All 5 rows displayed according  ly | Pass | Refer 1.1 attachment |
| 2 | Data passed from Item source to  Item Staging | | use Test\_DW select \*  from TestItemStaging; | All 5 rows displayed | All 5 rows displayed | Pass | Refer 1.2 attachment |

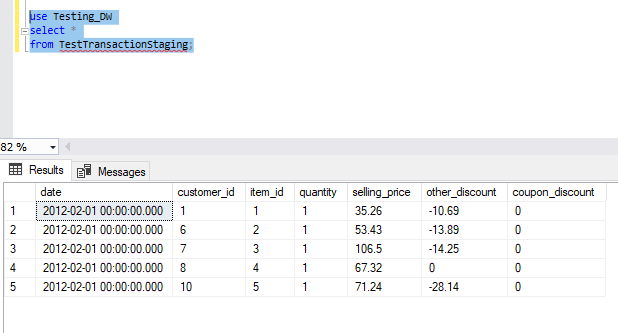
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  | according  ly | according  ly |  |  |
| 3 | Data passed from Transaction source to  Transaction Staging | use Test\_DW select \*  from TestTransactionS taging; | All 5 rows displayed according ly | All 5 rows displayed according ly | Pass | Refer 1.3 attachment |
| 4 | Data passed from Campaign source to Campaign  Staging | use Test\_DW select \*  from TestCampaignSta ging; | All 5 rows displayed according ly | All 5 rows displayed according ly | Pass | Refer 1.4 attachment |
| 5 | Data passed from Customer source to Customer Staging | use Test\_DW select \*  from TestCustomerSta ging; | All 5 rows displayed according ly | All 5 rows displayed according ly | Pass | Refer 1.5 attachment |



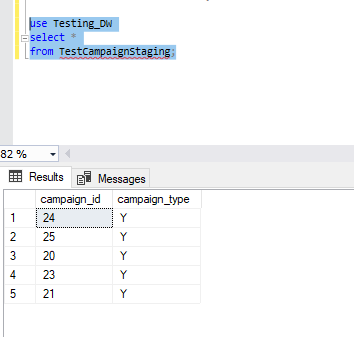
Attachment 1.1



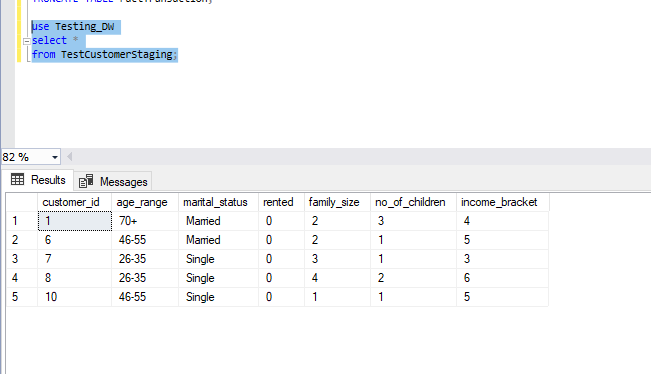
Attachment 1.2



Attachment 1.3



Attachment 1.4

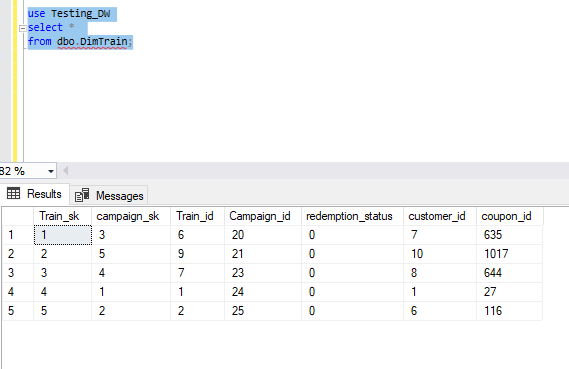


Attachment 1.5

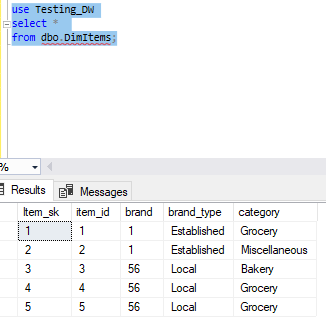
### Testing test data loaded from staging to dimension tables

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Scenario**  **ID** | | 2 | | | | | |
| **Test Case**  **Description** | | Transform test data from staging to dimension tables | | | | | |
| **Pre-Requisite** | | Test Data loaded from staging to dimension tables in SQL tool. | | | | | |
| **S.N**  **o** | **Action** | | **Sql Query** | **Expected Output** | **Actual Output** | **Test Resul**  **t** | **Test Comments** |
| 1 | Data passed from train  staging to | | use Test\_DW select \*  from DimTrain; | All 5 rows  displayed | All 5 rows  displayed | Pass | Refer 2.1 attachment |

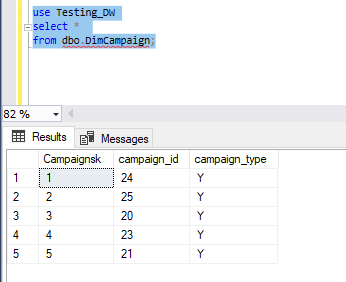
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Train  dimension |  | according  ly | according  ly |  |  |
| 2 | Data passed from Item staging to  Item dimension | use Test\_DW select \*  from DimItem; | All 5 rows displayed  according ly | All 5 rows displayed  according ly | Pass | Refer 2.2 attachment |
| 3 | Data passed from Campaign staging to Campaign  dimension | use Test\_DW select \*  from DimCampaign; | All 5 rows displayed according ly | All 5 rows displayed according ly | Pass | Refer 2.3 attachment |
| 4 | Data passed from Customer staging to Customer dimension | use Test\_DW select \*  from DimCustomer; | All 5 rows displayed according ly | All 5 rows displayed according ly | Pass | Refer 2.4 attachment |



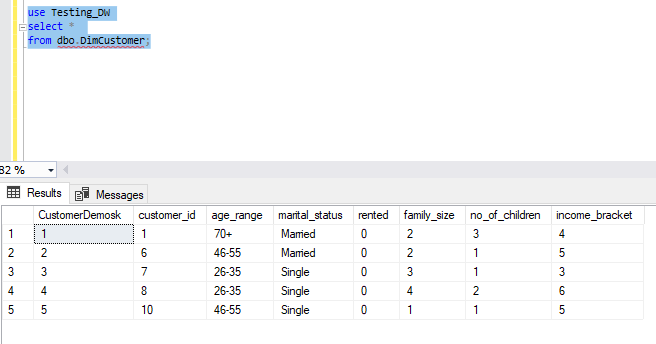
Attachment 2.1



Attachment 2.2

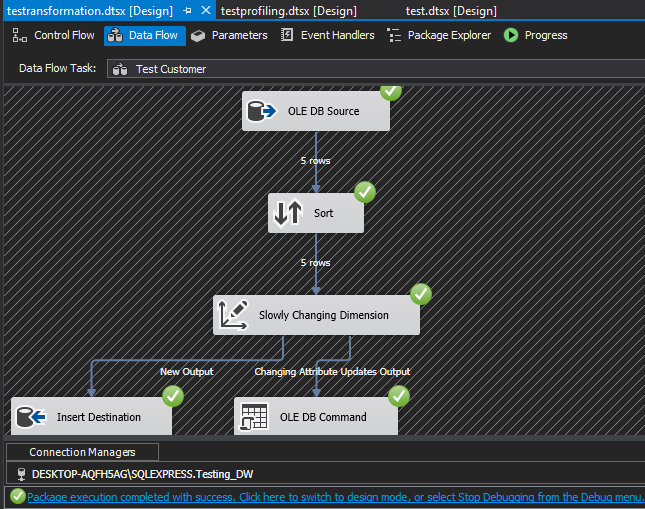
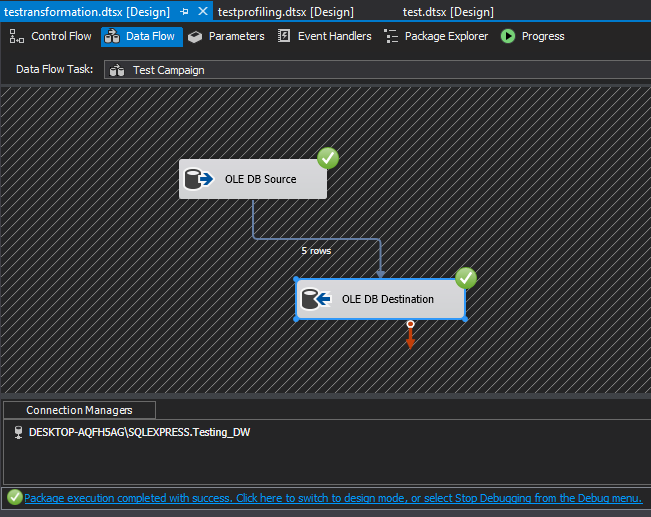


Attachment 2.3



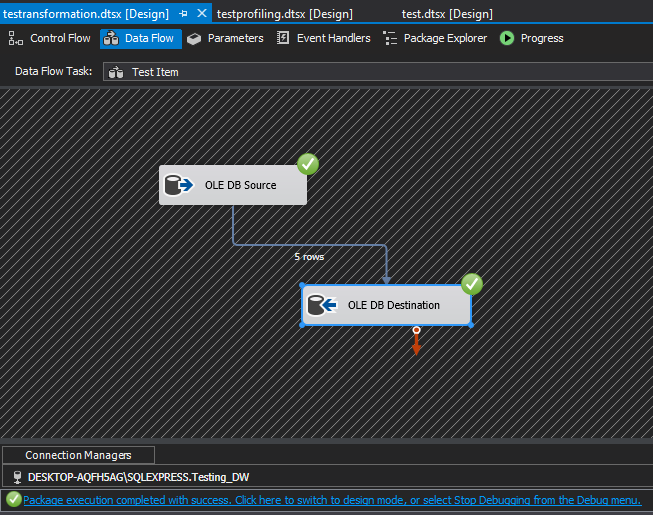
Attachment 2.4

Please find below screenshots of the successfully executed transformation process of the test data set.

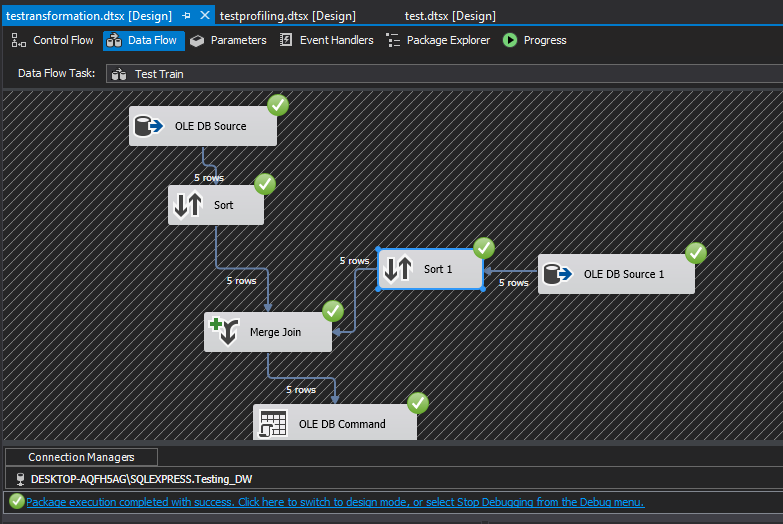


5 test data rows in campaign inserted successfully

5 test data rows in customer inserted successfully



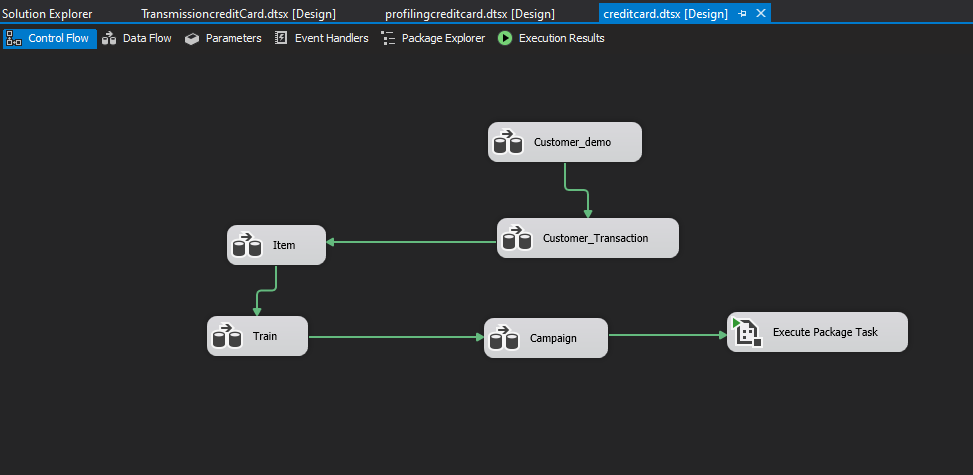
5 test data rows in Item inserted successfully



5 test data rows in Train inserted successfully

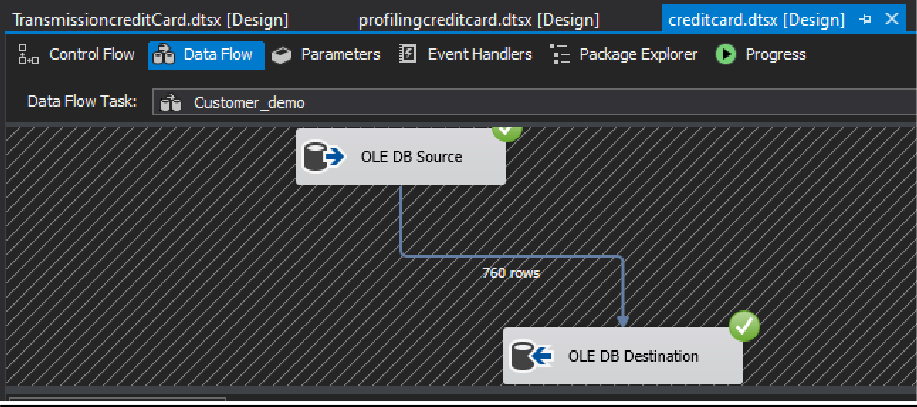
# ETL Development

As the first step data was extracted 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 flow tasks were joined as shown below at the end:



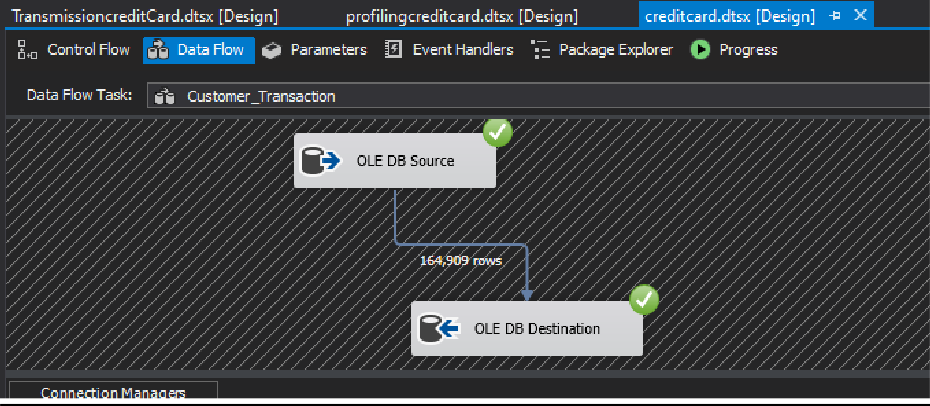
Screenshots of all the data sources that were staged and truncate tables created are attached below:

Staging customer details



Customer Demo – data is extracted from the Customer Demo table in the source database and inserted to the Customer Demo Staging Table

Staging Customer Transaction details

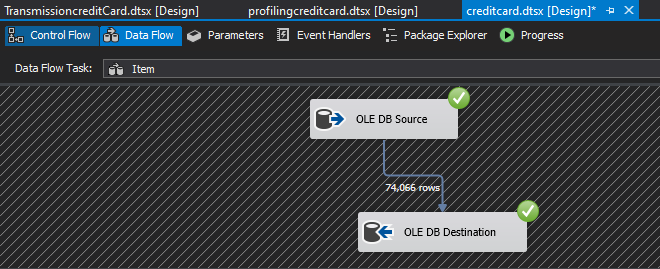


Customer transactions

– data is extracted from the Customer

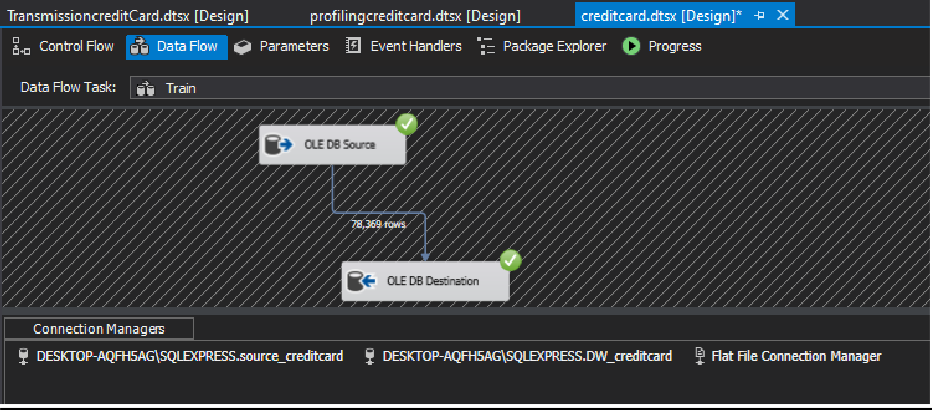
Transactions table in the source database and inserted to the Customer transactions Staging Table

Staging Item Details



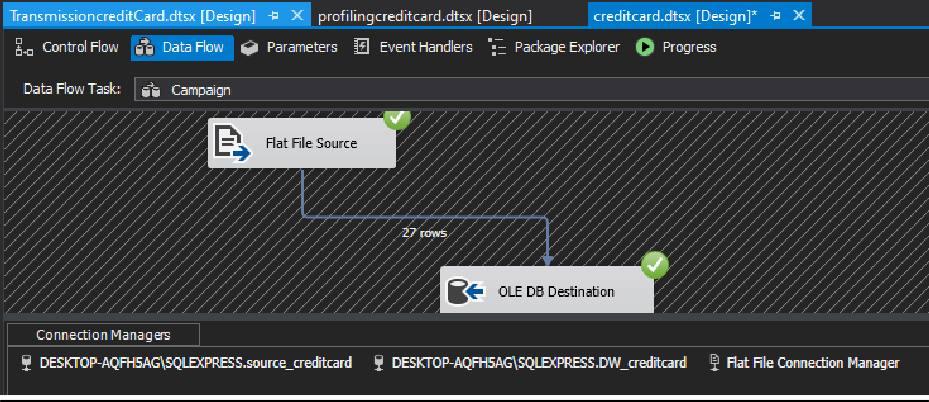
Item details – data is extracted from the items table in the source database and inserted to the item Staging Table

Staging train details (summary details)



summary details – data is extracted from the train table in the source database and inserted to the train staging Table

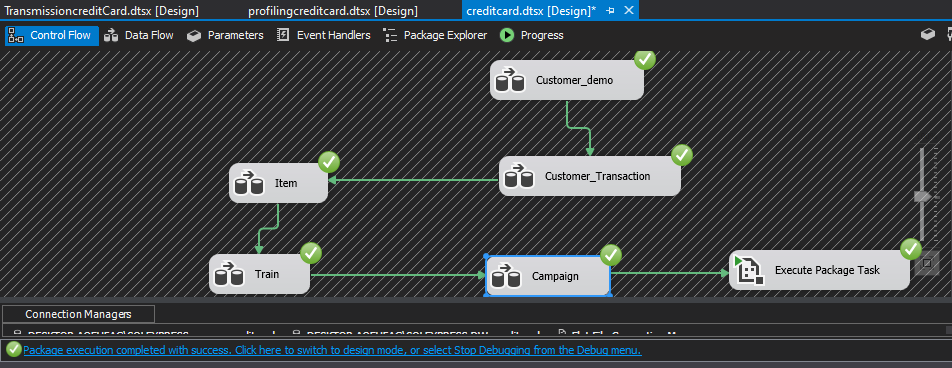
Staging campaign details



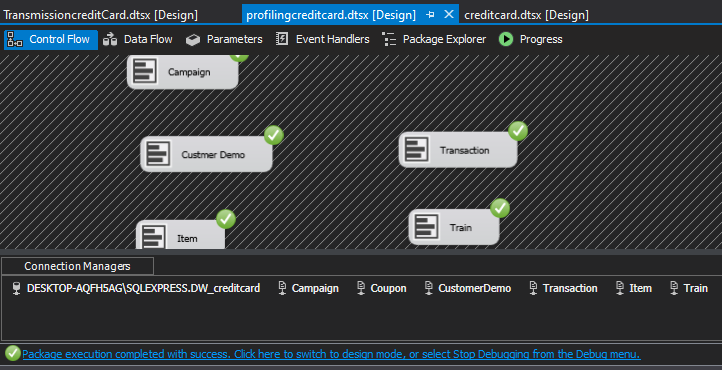
campaign details – data of the promotion campaigns held is extracted from the text file and inserted to the campaign staging table

The execution task connected to the last data flow task is linked to the transformations package.

After following the above steps and executing:

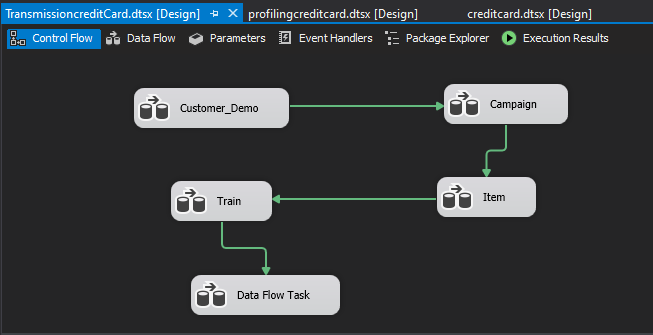


Next step is data profiling and it is done as shown below:



Every staging table is profiled and saved in a selected location

Next step is data transformation and as explained in a previous step, the execution task connected to the last data flow task of the first package is attached to the transformation package used for transformation.

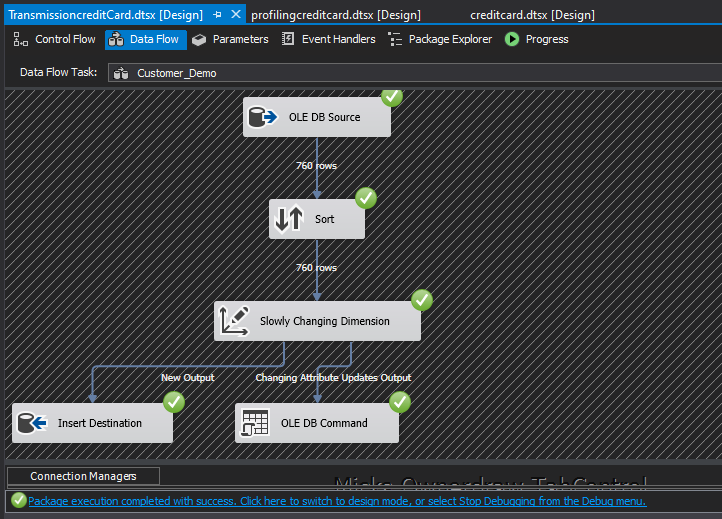


As mentioned earlier under assumptions, customer details were considered as slowly changing details.

The below mentioned columns were set as changing attributes:

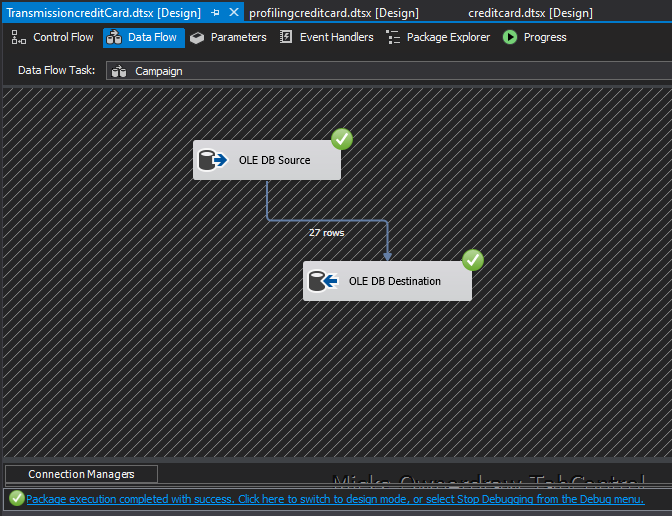
1. Age range (age range the customer belong to)
2. Income bracket (wallet size of the customer)
3. Marital status (Civil status)
4. Number of children
5. Family size (number of family members)
6. Rented

After extracting data from the Customer staging table, it was sorted according to the customer id and as it was identified as a slowly changing dimension, it was connected as shown below and loaded data to the Customer dimension table.



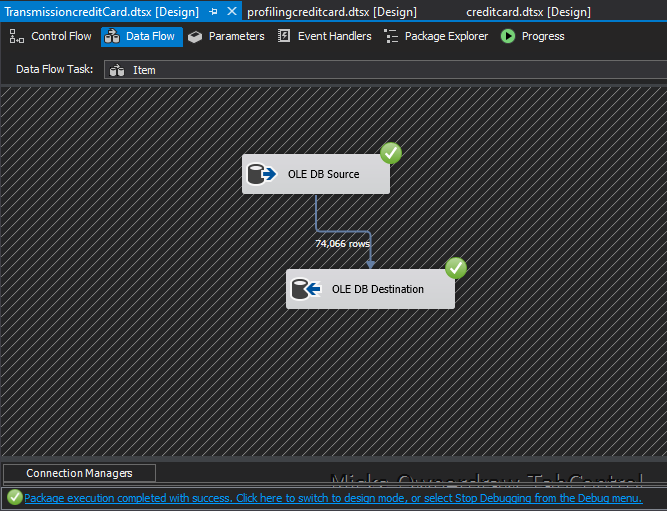
1. Extracted from Customer Staging table
2. Sorted by Customer ID
3. Making the dimension a Slowly Changing Dimension
4. Loaded to Customer Dimension

Next Data was loaded from the Campaign staging table to the Campaign Dimension.



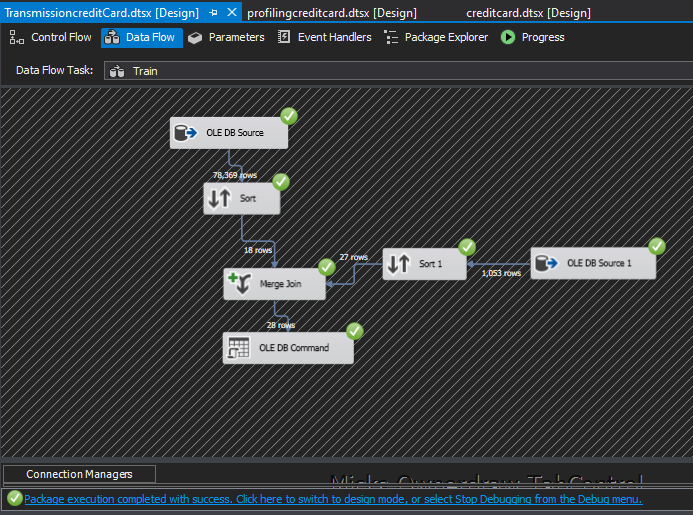
1. Extracted from Campaign Staging table
2. Loaded to Campaign Dimension

Same way Item Data was loaded from the Item staging table to item Dimension.



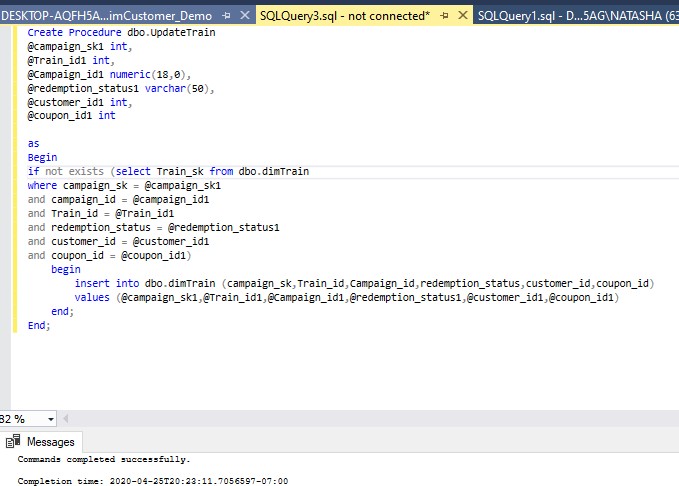
1. Extracted from Item Staging table
2. Loaded to Item Dimension

Next data was loaded from the train staging table to train dimension after merging the train staging table with the campaign dimension table. Both train staging table and campaign dimension was sorted according to the campaign id and then merge joined before loading in order to capture the train details from the train staging table and campaign surrogate key from the campaign dimension.



1. Extracted from Train Staging table
2. Train staging data sorted according to the campaign id
3. Campaign data extracted from the campaign dimension table and sorted based on the campaign id
4. Data loaded to the Train dimension (campaign SK extracted from campaign

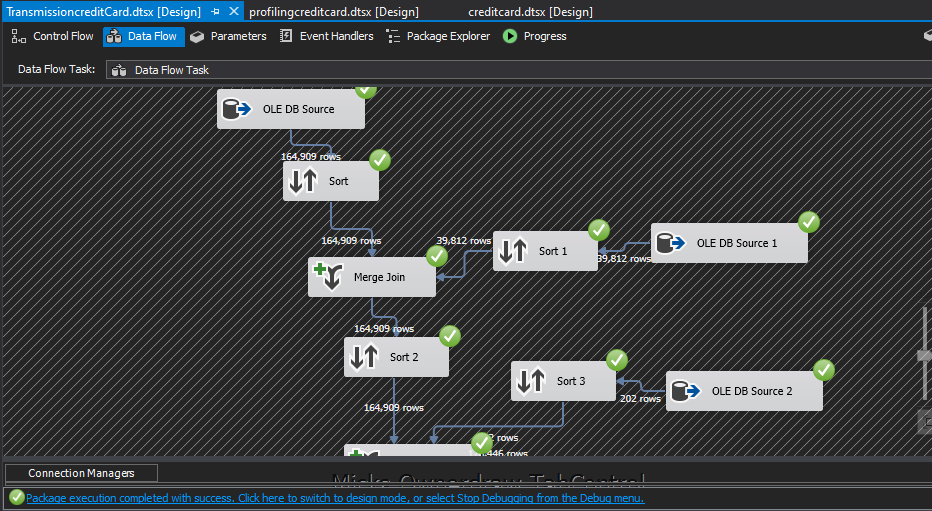
The update procedure used to update train details is attached below:



This procedure was linked to the DB command and executed in order to load data.

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

1. Data extracted from the customer transaction staging
2. Data sorted date wise
3. Date dimension loaded to another source database and sorted according to the date column
4. Merge join using the date and get the date surrogate key
5. Next sort the main data set based on the customer ID
6. Load the train dimension to another source database and sort customer ID wise
7. Merge join using the customer ID and get the Train surrogate key
8. Extracted data set on the left is next sorted by the item ID.
9. Load Item Dimension to a new source database and sort item ID wise
10. Merge join using the Item ID and get the Item surrogate key.
11. Finally, the data set is again sorted based on the customer ID.
12. Customer Demo dimension is loaded to a source database and sorted according to the customer ID.
13. Merge join using the sorted Customer ID and getthe Customer surrogate key.
14. Extracted and merged data are loaded to the Transaction Dimension.



1

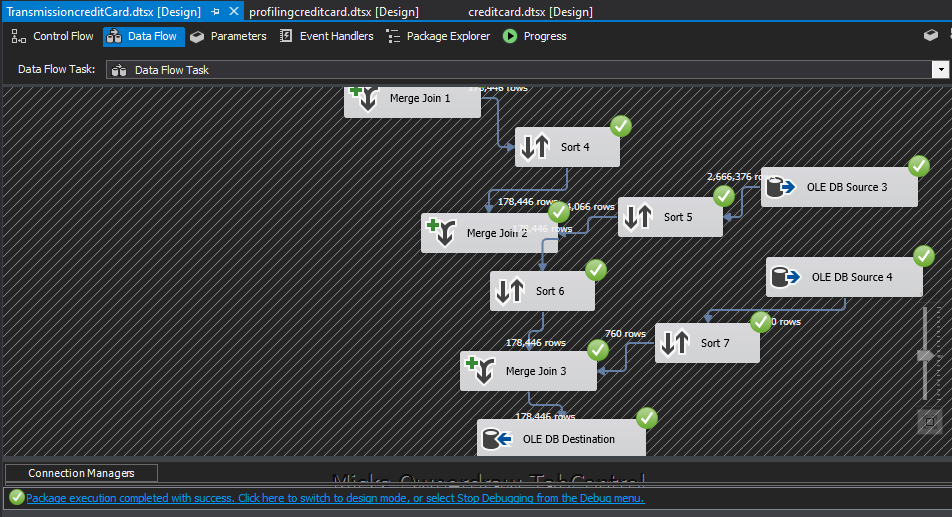
2

3

4

5

6



7

8

9

14

13

12

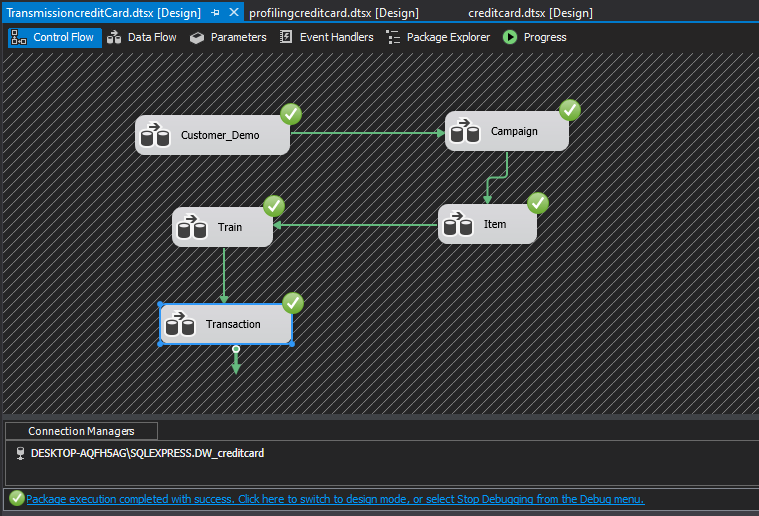
10 & 11

The query used to create the date dimension is mentioned below:

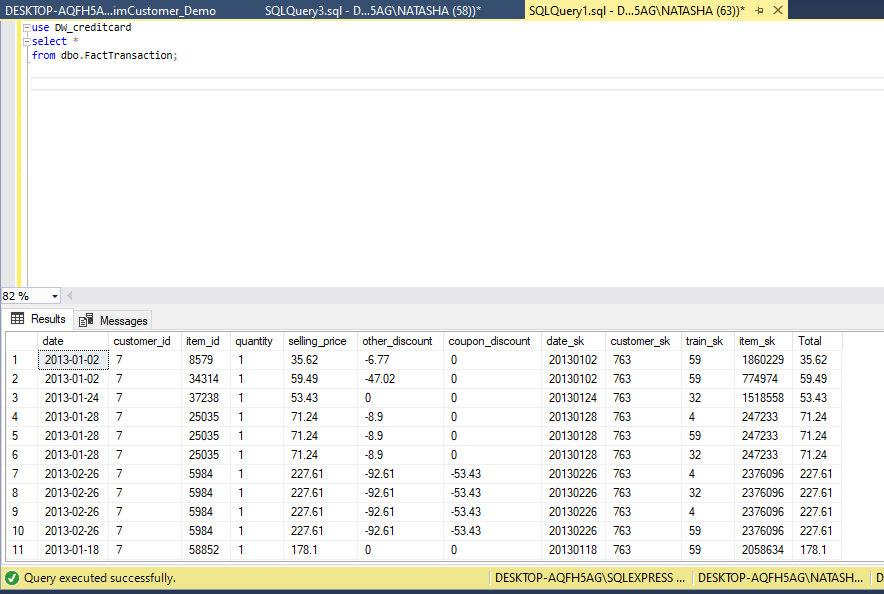
|  |  |
| --- | --- |
| BEGIN TRY  DROP TABLE [dbo].[DimDate]  END TRY BEGIN CATCH  /No Action/  END CATCH  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ CREATE TABLE [dbo].[DimDate]  ( [DateKey] INT primary key, [Date] DATETIME,  [FullDateUK] CHAR(10), -- Date in dd-MM-yyyy format [FullDateUSA] 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:



Please find below a printscreen of the fact table:



* The column Total is calculated the following way Total = quantity \* selling\_price

# Execution of Test Cases and TSR

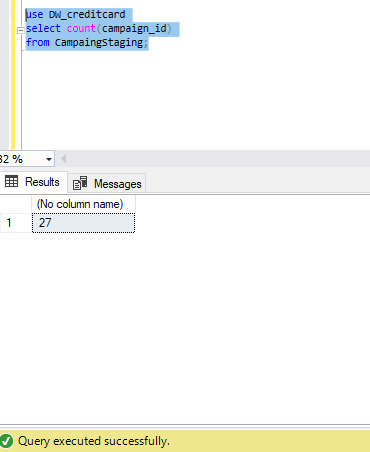
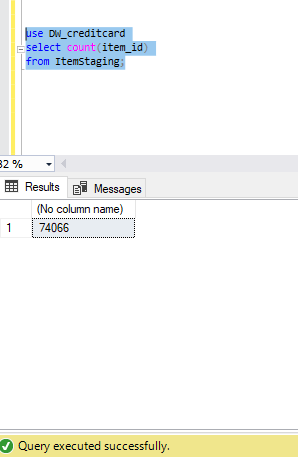
After testing using test data and passing all the test cases, data set was loaded (As explained earlier). The loaded data was tested using SQL queries as the data set is large and testing row by row is a tiresome compared to testing test data.

As mentioned earlier, ETL testing should be tested when data is being transformed from source to destinations not only at the two ends but also in the middle stages. In the test cases conducted it was tested that data was passed properly not only from source to staging but also from staging to the destination as expected.

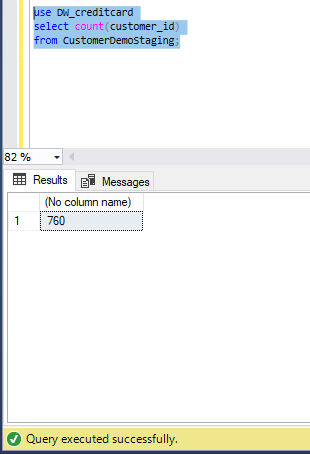
Execution of Test Cases

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Scenario**  **ID** | | 3 | | | | | |
| **Test Case**  **Description** | | Check for the count when transforming data from source to staging tables | | | | | |
| **Pre-Requisite** | | Data loaded from source to staging tables in SQL tool | | | | | |
| **S.N**  **o** | **Action** | | **Sql Query** | **Expecte d**  **Output** | **Actual Output** | **Test Resul**  **t** | **Test Comments** |
| 1 | Check for the count when transformin g data from Item source to Item  Staging | | use DW\_creditcard select count(item\_id) from ItemStaging; | 74,066 | 74,066 | Pass | Source count and the staging count of Item data tallies.  Refer 3.1 attachment |
| 2 | Check for the count when transformin g data from Campaign source to Campaign  Staging | | use DW\_creditcard select count(campaign\_i d)  from CampaignStaging; | 27 | 27 | Pass | Source count and the staging count of campaign data tallies. Refer 3.2 attachment |
| 3 | Check for the count when transformin  g data from | | use DW\_creditcard select count(customer\_i  d) | 760 | 760 | Pass | Source count and the staging count of customer data tallies.  Refer 3.3 attachment |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Customer source to  Customer Staging | from CustomerDemoSt aging; |  |  |  |  |



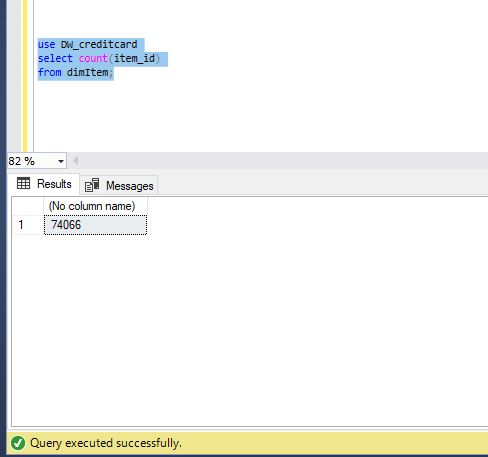
Attachment 3.1 Attachment 3.2



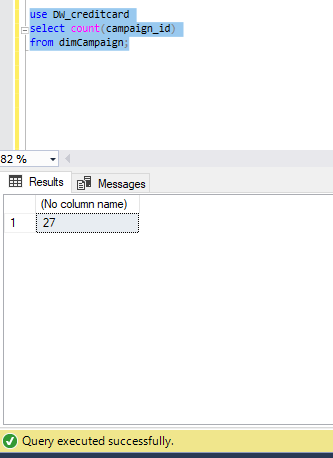
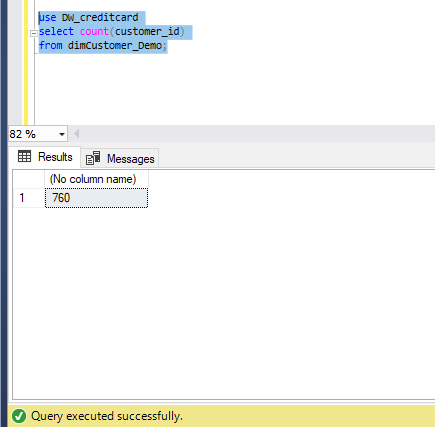
Attachment 3.3

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Scenario**  **ID** | | 4 | | | | | |
| **Test Case**  **Description** | | Check for the count when transforming data from staging to dimension tables | | | | | |
| **Pre-Requisite** | | Data loaded from staging to dimension tables in SQL tool | | | | | |
| **S.N**  **o** | **Action** | | **Sql Query** | **Expecte d**  **Output** | **Actual Output** | **Test Resul**  **t** | **Test Comments** |
| 1 | Check for the count when transformin g data from Item staging to Item  Dimension | | use DW\_creditcard select count(item\_id) from dimItem; | 74,066 | 74,066 | Pass | staging count and the dimension count of Item data tallies.  Refer 4.1 attachment |
| 2 | Check for the count when  transformin | | use DW\_creditcard | 27 | 27 | Pass | staging count and the dimension count of campaign data tallies.  Refer 4.2 attachment |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | g data from Campaign staging to  Campaign Dimension | select count(campaign\_i d)  from dimCampaign; |  |  |  |  |
| 3 | Check for | use | 760 | 760 | Pass | staging count and the |
|  | the count | DW\_creditcard |  |  |  | dimension count of |
|  | when | select |  |  |  | customer data tallies. |
|  | transformin | count(customer\_i |  |  |  | Refer 4.3 attachment |
|  | g data from | d) |  |  |  |  |
|  | Customer | from |  |  |  |  |
|  | staging to | CustomerDemoSt |  |  |  |  |
|  | Customer | aging; |  |  |  |  |
|  | Dimension |  |  |  |  |  |



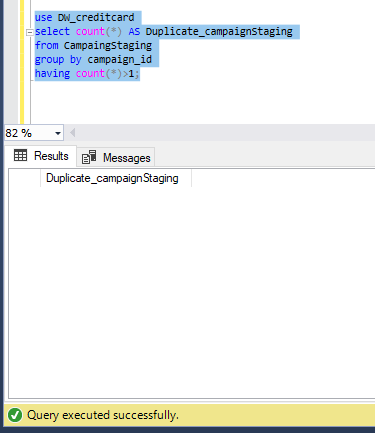
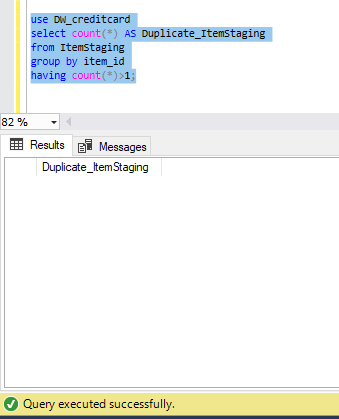
Attachment 4.1

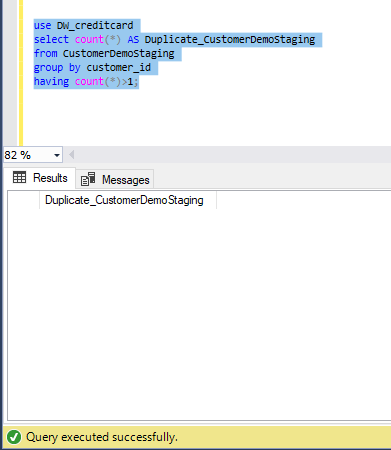
Attachment 3.2 Attachment 3.3

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Scenario**  **ID** | | 5 | | | | | |
| **Test Case**  **Description** | | Check for duplicate values in the staging tables. | | | | | |
| **Pre-Requisite** | | Data loaded from source to staging tables in SQL tool | | | | | |
| **S.N**  **o** | **Action** | | **Sql Query** | **Expecte d**  **Output** | **Actual Output** | **Test Resul**  **t** | **Test Comments** |
| 1 | Check whether data has got duplicated in Item staging | | use DW\_creditcard select count (\*) AS Duplicate\_ItemSta ging  from ItemStaging group by item\_id having count(\*)>1; | 0 | 0 | Pass | Data has not got duplicated in Item Staging Refer 5.1 attachment |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2 | Check whether data has got duplicated in Campaign staging | use DW\_creditcard select count (\*) AS Duplicate\_campai gnStaging  from CampaingStaging group by campaign\_id having count(\*)>1; | 0 | 0 | Pass | Data has not got duplicated in Campaign Staging  Refer 5.2 attachment |
| 3 | Check whether data has got duplicated in Customer staging | use DW\_creditcard select count (\*) AS Duplicate\_Custom erDemoStaging from CustomerDemoSt aging  group by customer\_id having count(\*)>1; | 0 | 0 | Pass | Data has not got duplicated in Customer Staging  Refer 5.3 attachment |



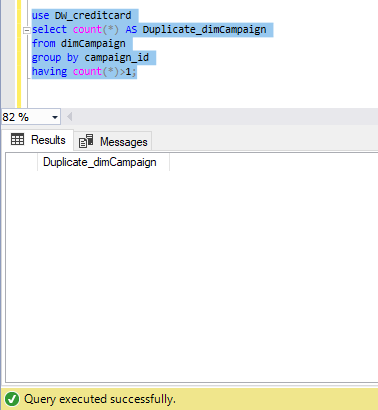
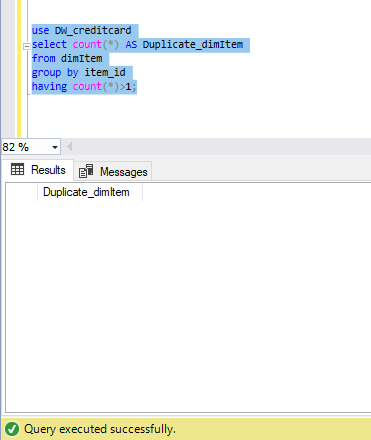
Attachment 5.1 Attachment 5.2



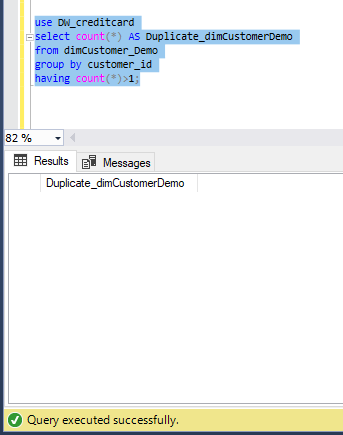
Attachment 5.3

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Scenario**  **ID** | | 6 | | | | | |
| **Test Case**  **Description** | | Check for duplicate values in the dimension tables. | | | | | |
| **Pre-Requisite** | | Data loaded from source to staging tables in SQL tool | | | | | |
| **S.N**  **o** | **Action** | | **Sql Query** | **Expecte d**  **Output** | **Actual Output** | **Test Resul**  **t** | **Test Comments** |
| 1 | Check whether data has got duplicated in Item Dimension | | use DW\_creditcard select count (\*) AS Duplicate\_dimIta m  from dimItem group by item\_id having count(\*)>1; | 0 | 0 | Pass | Data has not got duplicated in Item Dimension  Refer 6.1 attachment |
| 2 | Check whether  data has got | | use DW\_creditcard | 0 | 0 | Pass | Data has not got duplicated in Campaign  Dimension |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | duplicated in Campaign Dimension | select count (\*) AS Duplicate\_dimCa mpaign  from dimCampaign group by campaign\_id having count(\*)>1; |  |  |  | Refer 6.2 attachment |
| 3 | Check | use | 0 | 0 | Pass | Data has not got |
|  | whether | DW\_creditcard |  |  |  | duplicated in Customer |
|  | data has got | select count (\*) AS |  |  |  | Dimension |
|  | duplicated | Duplicate\_dimCus |  |  |  | Refer 6.3 attachment |
|  | in Customer | tomerDemo |  |  |  |  |
|  | Dimension | from |  |  |  |  |
|  |  | dimCustomer\_De |  |  |  |  |
|  |  | mo |  |  |  |  |
|  |  | group by |  |  |  |  |
|  |  | customer\_id |  |  |  |  |
|  |  | having count(\*)>1; |  |  |  |  |



Attachment 6.1 Attachment 6.2



Attachment 6.3

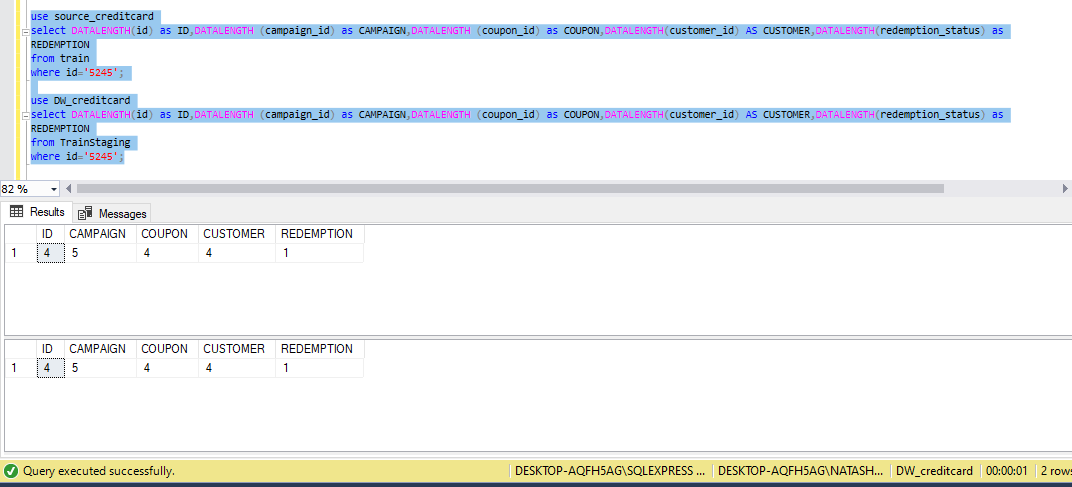
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Scenario**  **ID** | | 7 | | | | | |
| **Test Case**  **Description** | | Data length check for data in staging tables | | | | | |
| **Pre-Requisite** | | Data loaded from source to staging tables in SQL tool | | | | | |
| **S.N**  **o** | **Action** | | **Sql Query** | **Expecte d Output** | **Actual Output** | **Test Resul t** | **Test Comments** |
| 1 | Check | | use | ID - 4 | ID - 4 | Pass | Data lengths tally |
|  | whether the | | DW\_creditcard | BRAND - | BRAND - |  | Refer 7.1 attachment |
|  | data lengths | | select | 1 | 1 |  |  |
|  | in the Item | | DATALENGTH(ite | TYPE - | TYPE - 11 |  |  |
|  | source table | | m\_id) as | 11 | CATEGOR |  |  |
|  | and Item | | ID,DATALENGTH | CATEGO | Y - 7 |  |  |
|  | Staging | | (brand) as | RY - 7 |  |  |  |
|  | table are | | BRAND,DATALENG |  |  |  |  |
|  | the same | | TH (brand\_type) |  |  |  |  |
|  |  | | as |  |  |  |  |
|  |  | | TYPE,DATALENGT |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | H(category) AS CATEGORY  from ItemStaging where item\_id='1'; |  |  |  |  |
| 2 | Check | use | ID -4 | ID -4 | Pass | Data lengths tally |
|  | whether the | DW\_creditcard | CAMPAI | CAMPAIG |  | Refer 7.2 attachment |
|  | data lengths | select | GN - 5 | N - 5 |  |  |
|  | in the Train | DATALENGTH(id) | COUPO | COUPON |  |  |
|  | source table | as | N - 4 | - 4 |  |  |
|  | and Train | ID,DATALENGTH | CUSTO | CUSTOM |  |  |
|  | Staging | (campaign\_id) as | MER - 4 | ER - 4 |  |  |
|  | table are | CAMPAIGN,DATAL | REDEMP | REDEMP |  |  |
|  | the same | ENGTH | TION - 1 | TION - 1 |  |  |
|  |  | (coupon\_id) as |  |  |  |  |
|  |  | COUPON,DATALE |  |  |  |  |
|  |  | NGTH(customer\_i |  |  |  |  |
|  |  | d) AS |  |  |  |  |
|  |  | CUSTOMER,DATAL |  |  |  |  |
|  |  | ENGTH(redemptio |  |  |  |  |
|  |  | n\_status) as |  |  |  |  |
|  |  | REDEMPTION |  |  |  |  |
|  |  | from TrainStaging |  |  |  |  |
|  |  | where id='5245'; |  |  |  |  |
| 3 | Check | use | ID – 4 | ID – 4 | Pass | Data lengths tally |
|  | whether the | DW\_creditcard | AGE – 5 | AGE – 5 |  | Refer 7.3 attachment |
|  | data lengths | select | STATUS | STATUS – |  |  |
|  | in the | DATALENGTH(cust | – 6 | 6 |  |  |
|  | Customer | omer\_id) as | RENTED | RENTED |  |  |
|  | source table | ID,DATALENGTH | – 1 | – 1 |  |  |
|  | and | (age\_range) as | FAMILY | FAMILY – |  |  |
|  | Customer | AGE,DATALENGTH | – 2 | 2 |  |  |
|  | Staging | (marital\_status) as | CHILDRE | CHILDRE |  |  |
|  | table are | STATUS,DATALEN | N – 2 | N – 2 |  |  |
|  | the same | GTH(rented) AS | INCOME | INCOME - |  |  |
|  |  | RENTED,DATALEN | - 4 | 4 |  |  |
|  |  | GTH(family\_size) |  |  |  |  |
|  |  | as |  |  |  |  |
|  |  | FAMILY,DATALEN |  |  |  |  |
|  |  | GTH(no\_of\_childr |  |  |  |  |
|  |  | en) as CHILDREN, |  |  |  |  |
|  |  | DATALENGTH(inco |  |  |  |  |
|  |  | me\_bracket) as |  |  |  |  |
|  |  | INCOME |  |  |  |  |
|  |  | from |  |  |  |  |
|  |  | CustomerDemoSt |  |  |  |  |
|  |  | aging |  |  |  |  |

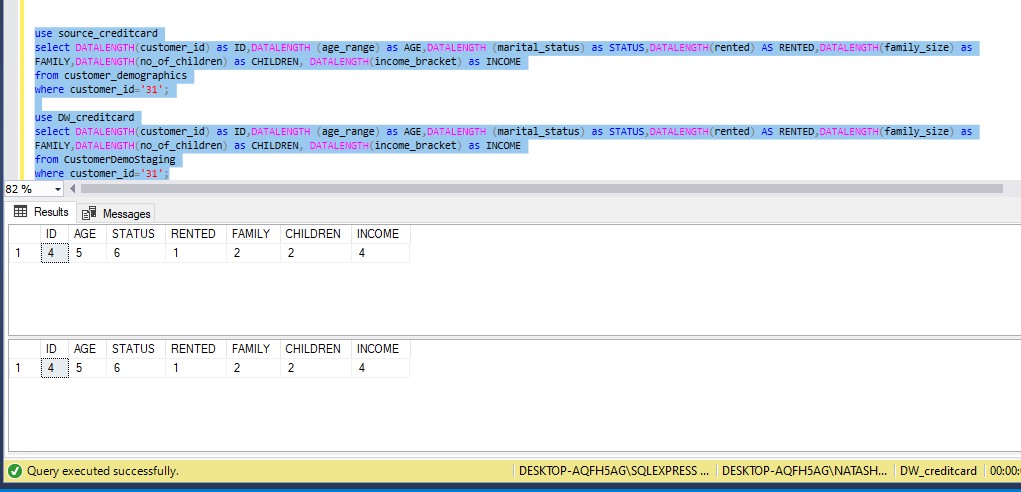
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | where  customer\_id='31'; |  |  |  |  |



Attachment 7.1



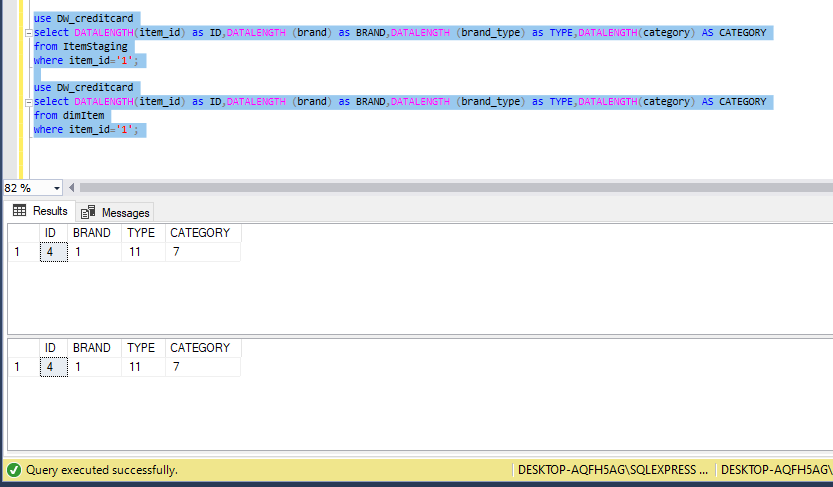
Attachment 7.2



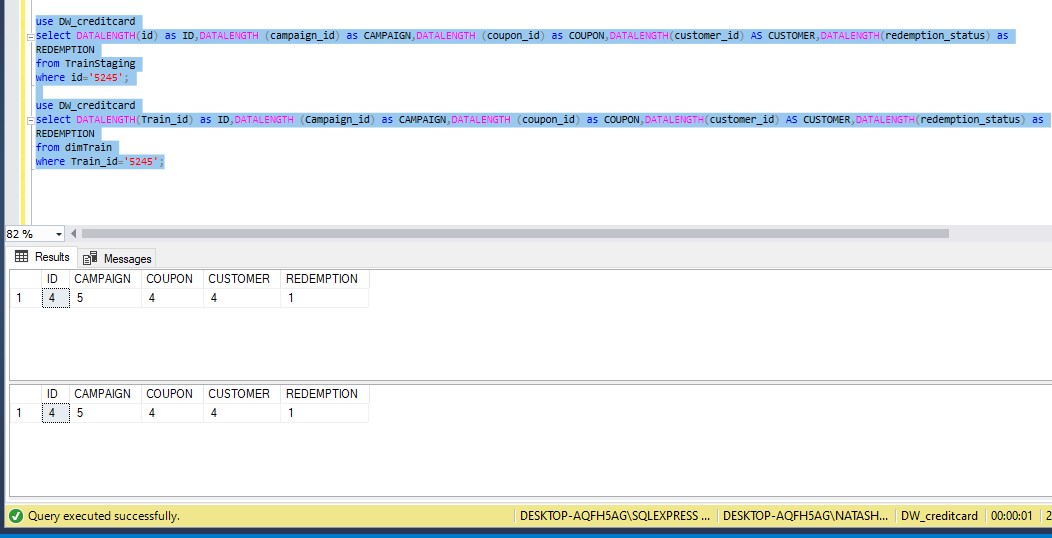
Attachment 7.3

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Scenario**  **ID** | | 8 | | | | | |
| **Test Case**  **Description** | | Data length check for data in dimension tables | | | | | |
| **Pre-Requisite** | | Data loaded from staging to dimension tables in SQL tool | | | | | |
| **S.N**  **o** | **Action** | | **Sql Query** | **Expecte**  **d Output** | **Actual Output** | **Test**  **Resul t** | **Test Comments** |
| 1 | Check | | use | ID - 4 | ID - 4 | Pass | Data lengths tally |
|  | whether the | | DW\_creditcard | BRAND - | BRAND - |  | Refer 8.1 attachment |
|  | data lengths | | select | 1 | 1 |  |  |
|  | in the Item | | DATALENGTH(ite | TYPE - | TYPE - 11 |  |  |
|  | staging | | m\_id) as | 11 | CATEGOR |  |  |
|  | table and | | ID,DATALENGTH | CATEGO | Y - 7 |  |  |
|  | Item | | (brand) as | RY - 7 |  |  |  |
|  | dimension | | BRAND,DATALENG |  |  |  |  |
|  | table are | | TH (brand\_type) |  |  |  |  |
|  | the same | | as |  |  |  |  |
|  |  | | TYPE,DATALENGT |  |  |  |  |
|  |  | | H(category) AS |  |  |  |  |
|  |  | | CATEGORY |  |  |  |  |
|  |  | | from dimItem |  |  |  |  |
|  |  | | where |  |  |  |  |
|  |  | | item\_id='1'; |  |  |  |  |
| 2 | Check | | use | ID -4 | ID -4 | Pass | Data lengths tally |
|  | whether the | | DW\_creditcard | CAMPAI | CAMPAIG |  | Refer 8.2 attachment |
|  | data lengths | | select | GN - 5 | N - 5 |  |  |
|  | in the Train | | DATALENGTH(Trai | COUPO | COUPON |  |  |
|  | staging | | n\_id) as | N - 4 | - 4 |  |  |
|  | table and | | ID,DATALENGTH | CUSTO | CUSTOM |  |  |
|  | Train | | (Campaign\_id) as | MER - 4 | ER - 4 |  |  |

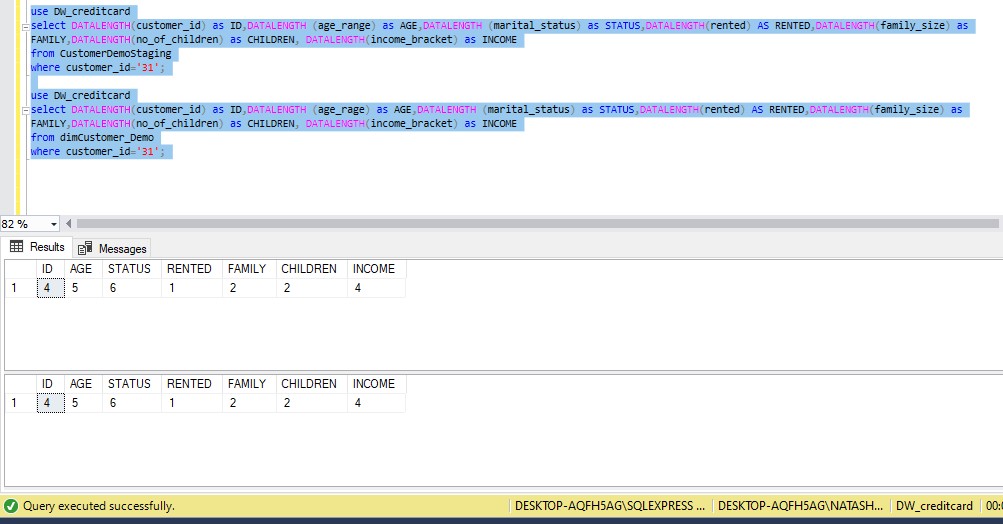
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | dimension table are the same | CAMPAIGN,DATAL ENGTH  (coupon\_id) as COUPON,DATALE  NGTH(customer\_i d) AS CUSTOMER,DATAL  ENGTH(redemptio n\_status) as REDEMPTION  from dimTrain where  Train\_id='5245'; | REDEMP TION - 1 | REDEMP TION - 1 |  |  |
| 3 | Check whether the data lengths in the Customer staging table and Customer dimension table are the same | use DW\_creditcard select DATALENGTH(cust  omer\_id) as ID,DATALENGTH  (age\_rage) as AGE,DATALENGTH  (marital\_status) as STATUS,DATALEN  GTH(rented) AS RENTED,DATALEN  GTH(family\_size) as FAMILY,DATALEN  GTH(no\_of\_childr en) as CHILDREN, DATALENGTH(inco  me\_bracket) as INCOME  from dimCustomer\_De mo  where  customer\_id='31'; | ID – 4  AGE – 5 STATUS  – 6  RENTED  – 1  FAMILY  – 2  CHILDRE N – 2 INCOME  - 4 | ID – 4  AGE – 5 STATUS – 6 RENTED  – 1  FAMILY – 2 CHILDRE N – 2 INCOME - 4 | Pass | Data lengths tally Refer 8.3 attachment |



Attachment 8.1



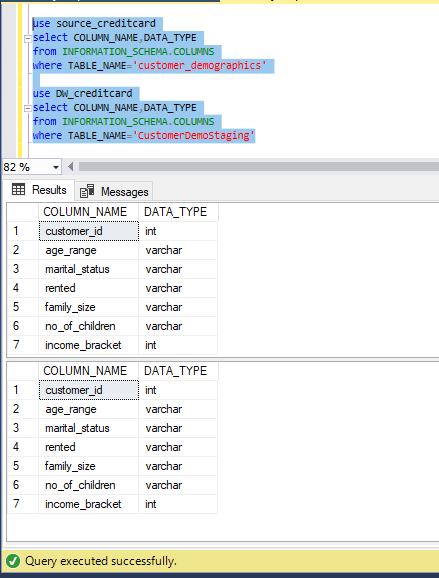
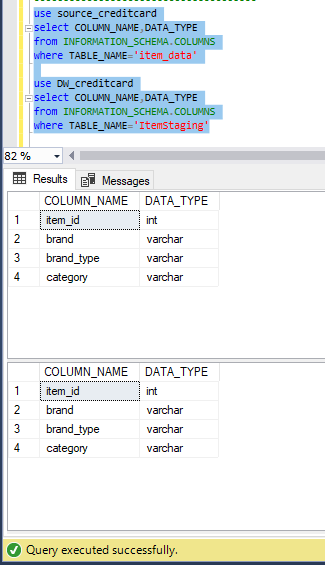
Attachment 8.2



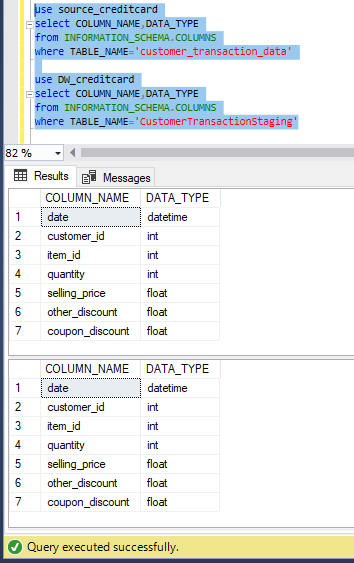
Attachment 8.3

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Scenario**  **ID** | | 9 | | | | | |
| **Test Case**  **Description** | | Data type check for data in staging tables | | | | | |
| **Pre-Requisite** | | Data loaded from source to staging tables in SQL tool | | | | | |
| **S.N**  **o** | **Action** | | **Sql Query** | **Expecte**  **d Output** | **Actual Output** | **Test**  **Resul t** | **Test Comments** |
| 1 | Check whether the data types in the customer source table and customer Staging table are the same | | use DW\_creditcard select DATA COLUMN, DATA TYPE  from INFORMATION\_SC HEME.COLUMNS  where  TABLE\_NAME=’Cu  stomerDemoStagi  ng’ | Custom er\_id - int age\_ran ge – varchar marital\_ status - varchar family\_s ize no\_of\_c hildren  –  varchar income\_ bracket - int | Customer  \_id -int age\_rang e – varchar marital\_s tatus - varchar family\_si ze no\_of\_ch ildren – varchar income\_ bracket - int | Pass | Data types tally Refer 9.1 attachment |
| 2 | Check whether the data types  in the item | | use DW\_creditcard | Item\_id  -int brand -  varchar | Item\_id - int brand -  varchar | Pass | Data types tally Refer 9.2 attachment |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | source table and item Staging table are the same | select DATA COLUMN, DATA TYPE  from INFORMATION\_SC HEME.COLUMNS  where  TABLE\_NAME=’Ite  mStaging’ | brand\_t ype – varchar category  -varchar | brand\_ty pe – varchar category  -varchar |  |  |
| 3 | Check | use | date – | date – | Pass | Data types tally |
|  | whether the | DW\_creditcard | datetim | datetime |  | Refer 9.3 attachment |
|  | data types | select DATA | e | customer |  |  |
|  | in the | COLUMN, DATA | custome | \_id -int |  |  |
|  | customer | TYPE | r\_id -int | item\_id – |  |  |
|  | source table | from | item\_id | int |  |  |
|  | and | INFORMATION\_SC | – int | quantity |  |  |
|  | customer | HEME.COLUMNS | quantity | – int |  |  |
|  | Staging | where | – int | selling\_pr |  |  |
|  | table are | TABLE\_NAME=’Cu | selling\_p | ice -flaot |  |  |
|  | the same | stomerTransactio | rice - | other\_dis |  |  |
|  |  | nStaging’ | flaot | count - |  |  |
|  |  |  | other\_di | float |  |  |
|  |  |  | scount - | coupon\_ |  |  |
|  |  |  | float | discount |  |  |
|  |  |  | coupon\_ | -float |  |  |
|  |  |  | discount |  |  |  |
|  |  |  | -float |  |  |  |

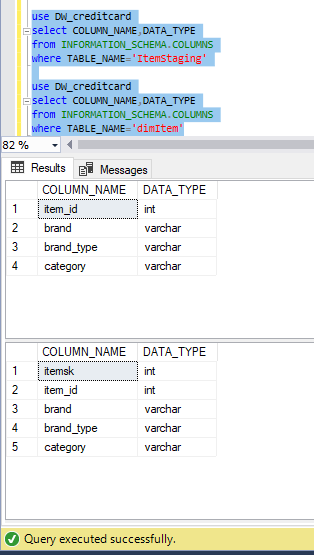
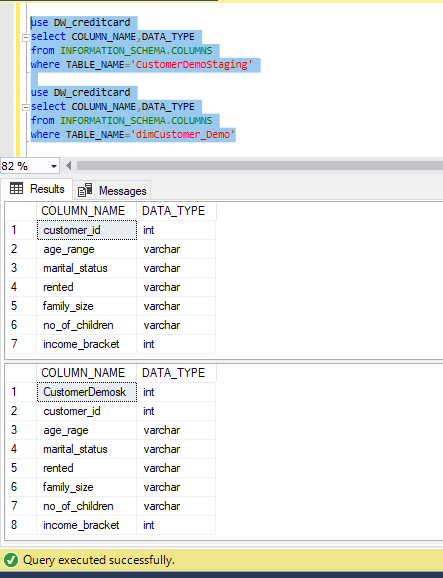
Attachment 9.1 Attachment 9.2



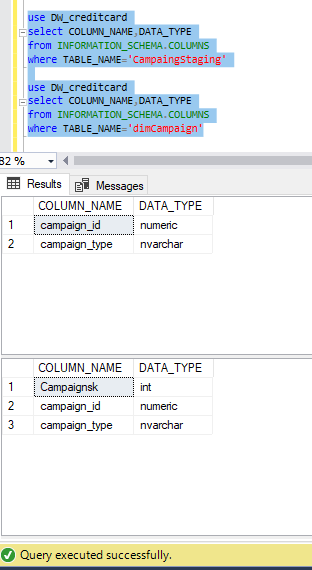
Attachment 9.3

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Scenario**  **ID** | | 10 | | | | | |
| **Test Case**  **Description** | | Data type check for data in dimension tables | | | | | |
| **Pre-Requisite** | | Data loaded from staging to dimension tables in SQL tool | | | | | |
| **S.N**  **o** | **Action** | | **Sql Query** | **Expecte d**  **Output** | **Actual Output** | **Test Resul**  **t** | **Test Comments** |
| 1 | Check | | use | Custom | Customer | Pass | Data types tally |
|  | whether the | | DW\_creditcard | er\_id - | \_id -int |  | Refer 10.1 attachment |
|  | data types | | select DATA | int | age\_rang |  |  |
|  | in the | | COLUMN, DATA | age\_ran | e – |  |  |
|  | customer | | TYPE | ge – | varchar |  |  |
|  | staging | | from | varchar | marital\_s |  |  |
|  | table and | | INFORMATION\_SC | marital\_ | tatus - |  |  |
|  | customer | | HEME.COLUMNS | status - | varchar |  |  |
|  | dimension | | where | varchar | family\_si |  |  |
|  | table are | | TABLE\_NAME=’di | family\_s | ze |  |  |
|  | the same | | mCustomerDemo’ | ize | no\_of\_ch |  |  |
|  |  | |  | no\_of\_c | ildren – |  |  |
|  |  | |  | hildren | varchar |  |  |
|  |  | |  | – | income\_ |  |  |
|  |  | |  | varchar | bracket - |  |  |
|  |  | |  | income\_ | int |  |  |
|  |  | |  | bracket - |  |  |  |
|  |  | |  | int |  |  |  |
| 2 | Check | | use | Item\_id | Item\_id - | Pass | Data types tally |
|  | whether the | | DW\_creditcard | -int | int |  | Refer 10.2 attachment |
|  | data types | | select DATA | brand - | brand - |  |  |
|  | in the item | | COLUMN, DATA | varchar | varchar |  |  |
|  | staging | | TYPE | brand\_t | brand\_ty |  |  |
|  | table and | | from | ype – | pe – |  |  |
|  | item | | INFORMATION\_SC | varchar | varchar |  |  |
|  | dimension | | HEME.COLUMNS | category | category |  |  |
|  | table are | | where | -varchar | -varchar |  |  |
|  | the same | | TABLE\_NAME=’di |  |  |  |  |
|  |  | | mItem’ |  |  |  |  |
| 3 | Check | | use | campaig | campaign | Pass | Data types tally |
|  | whether the | | DW\_creditcard | n\_id – | \_id – |  | Refer 10.3 attachment |
|  | data types | | select DATA | numeric | numeric |  |  |
|  | in the | | COLUMN, DATA | campaig | campaign |  |  |
|  | customer | | TYPE | n\_type - | \_type - |  |  |
|  | staging | | from | nvarchar | nvarchar |  |  |
|  | table and | | INFORMATION\_SC |  |  |  |  |
|  | campaign | | HEME.COLUMNS |  |  |  |  |
|  | dimension | |  |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | table are the same | where  TABLE\_NAME=’di  mCampaign’ |  |  |  |  |



Attachment 10.1 Attachment 10.2



Attachment 10.2

Test Summary Report

Please find below the summary of the test cases executed.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cycle number | Number of test cases | Passed | Failed | Comments |
| #1 | 8 | 6 | 2 | -Data types did not match in staging and in dimension tables in the transformation  process |
| #2 | 8 | 8 | 0 | All the test cases were passed |