 Sri Lanka Institute of Information Technology

**Assignment 1**

Dara Warehouse & Business Intelligence

2021

Submitted By:

Bandara P.M.P.C

IT19243818

**Contents**

[Data Selection & Preparation………………………………………………………3](#Dataselection2)

[Solution Architecture ………………………………………………………………6](#sol2)

[Data Warehouse Design & Development…………………………………………..7](#DW2)

[Test Planning & Test Data………………………………………………………….8](#TestPlanning2)

[ETL Development…………………………………………………………………..18](#ETL2)

[Execution of Test Cases and TSR…………………………………………………..31](#ExecuteTest2)

**[Data Selection & Preparation](#DataSelection)**

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

[**https://www.kaggle.com/rdoume/beerreviews**](https://www.kaggle.com/rdoume/beerreviews)

Modifications were done accordingly to the data set derived from the source . This Dataset reflects Customer reviews on beers in different breweries.

The two main sources are listed below:

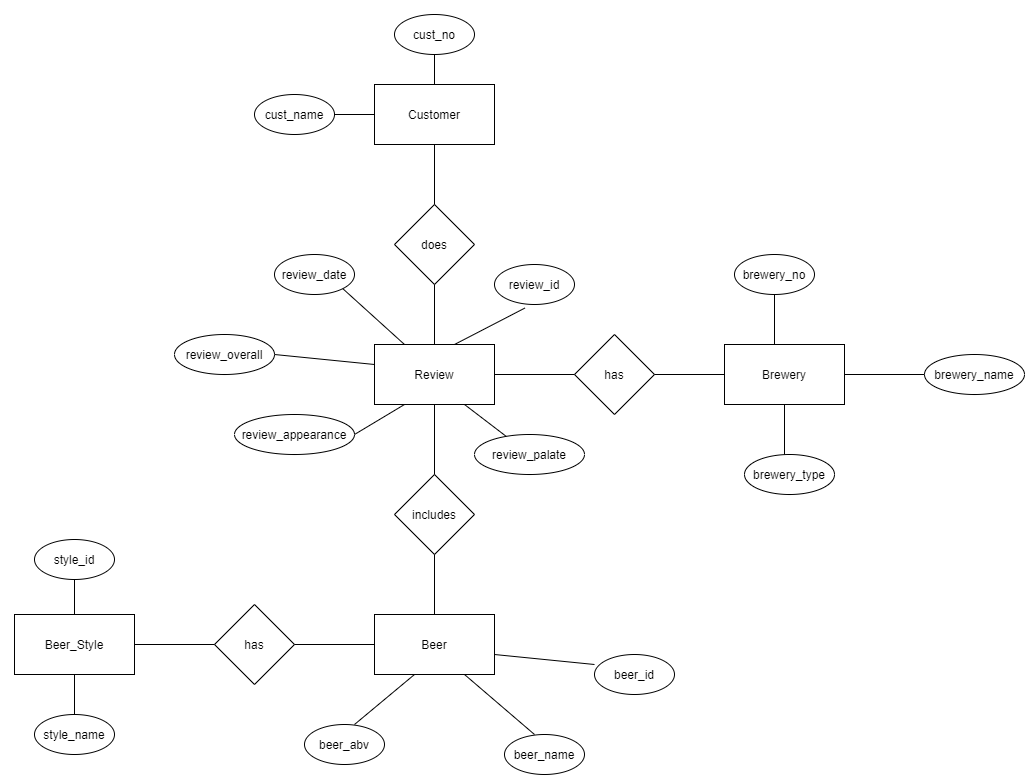
* SQL Database .
* One text file – Customer Data.

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

* Beer Details.
* Brewery Details.
* Beer Style Details.
* Review Details.

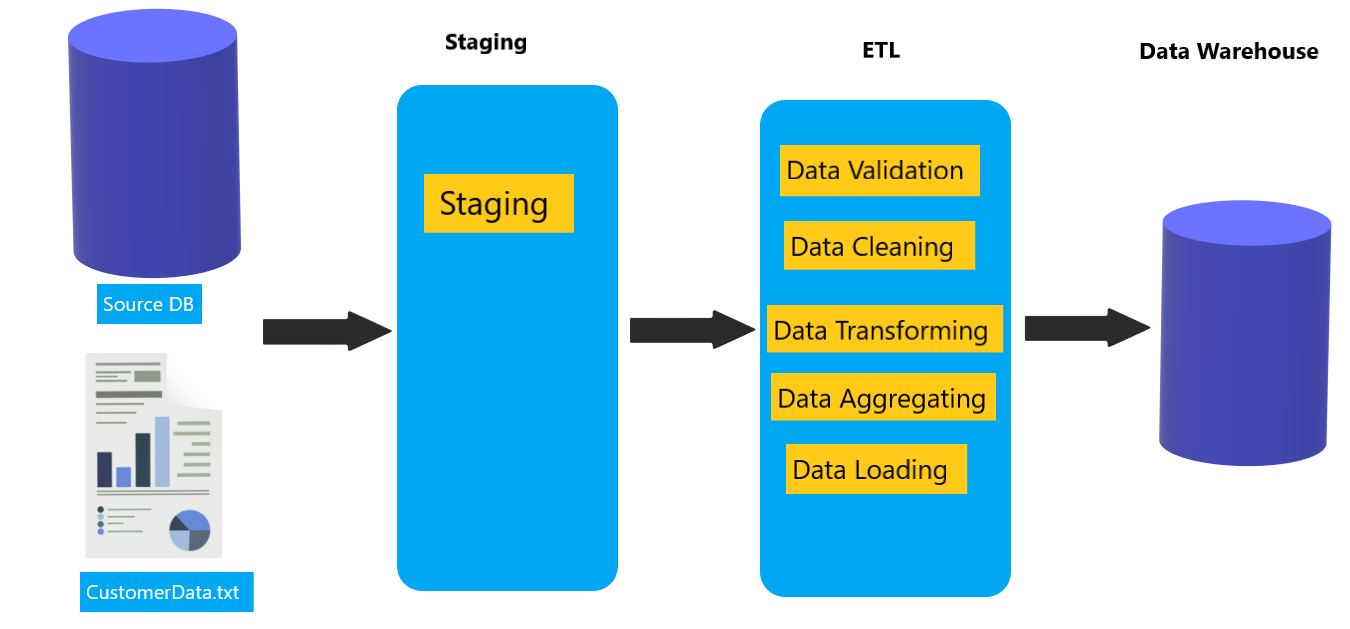
Description of The Dataset

|  |  |  |  |
| --- | --- | --- | --- |
| **Table Name** | **Column Name** | **Data type** | **Description** |
| Beer | Beer\_id | Varchar(16) | Details of Beers |
| Beer\_name | Nvarchar(255) |
| Beer\_abv | Float |
| Style\_id | Varchar(16) |
| Beer Style | Style\_id | varchar()16 | Details of Beer Styles(Categories) |
| Beer\_style | Varchar(255) |
| Brewery | Brewery\_no | int | Brewery Details |
| Brewery\_name | Nvarchar(255) |
| Brewery\_type | Nvarchar(255) |
| Customer | Customer\_no | int | Details of reviewed Customers |
| Customer\_name | Nvarchar(50) |
| Review | Review\_id | Int | Details of Reviews |
| Review\_date | Datetime |
| Brewey\_no | Int |
| Review\_overall | Float |
| Review\_appearance | Float |
| Review\_palate | Float |
| Cust\_no | Int |
| Beer\_id | Varchar(29) |

ER Diagram

Above diagram shows the connection between entities

**[Solution Architecture](#Solution)**

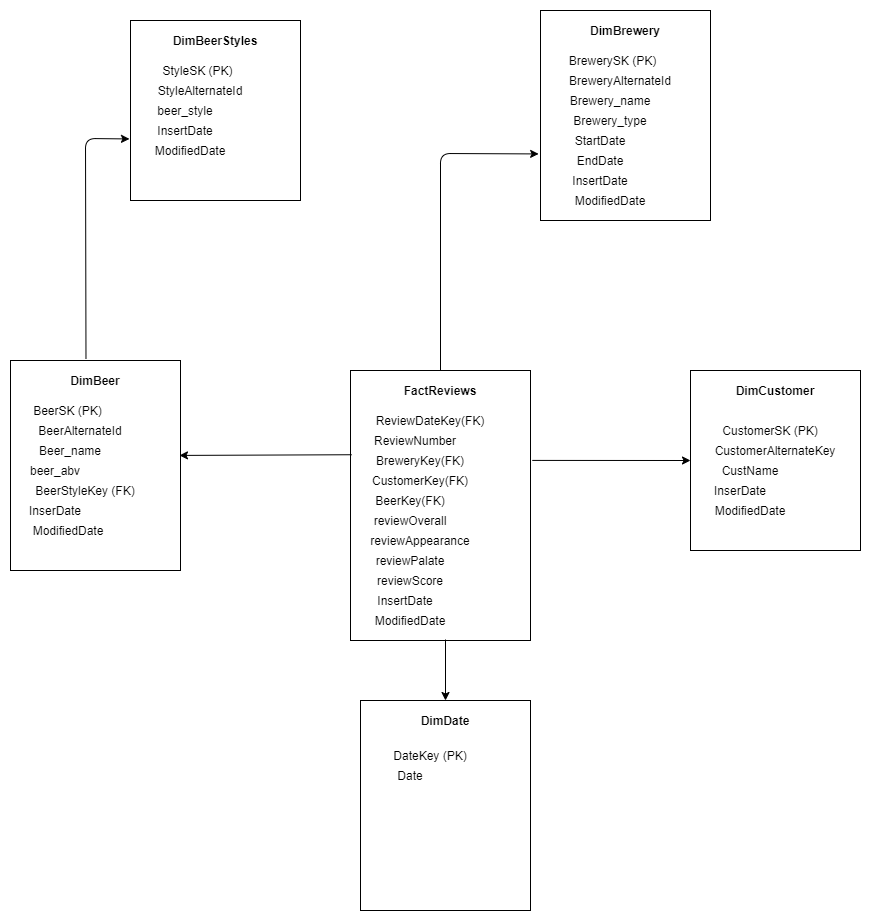
****

In the Staging layer Below Tables are created

* Stg Beer
* Stg Styles
* Stg Brewery
* Stg Customer
* Stg Reviews

Then 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.

****[**D**](#DW)**[ata Warehouse Design & Development](#DW)**

Snowflake schema is used to design the Datawarehouse design. There is one fact table as transactions and 5 dimensions. Review per Customer was considered as the grain.

Assumptions

* Brewery Dimension is considered as a Slowly changing dimension

**[Test Planning and Test Data](#TestPlanning)**

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
2. Staging to DW

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  2. Data length testing To make sure the data lengths tally when data is passed from source to middle stages as well as destination tables  3. Data type testing Data types to be tested to refrain the process being interrupted due to data types as this is a common issue.  4. 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 – 30/04/2021  End Date – 10/05/2021 |
| 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 2015 |

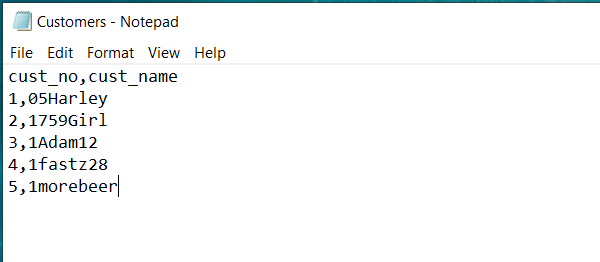
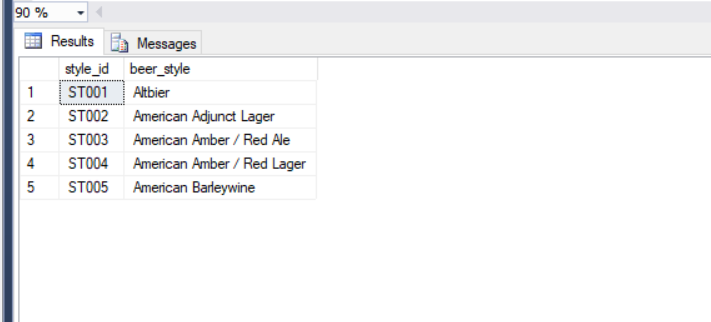
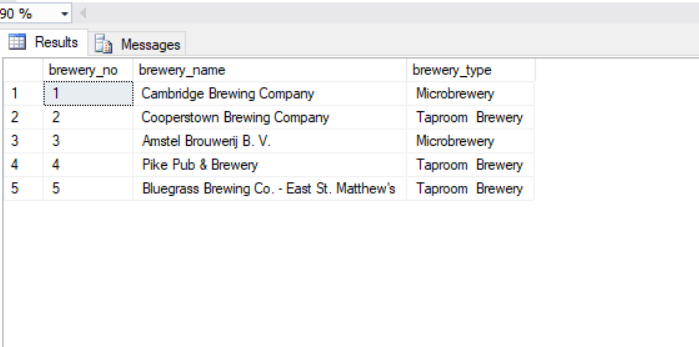
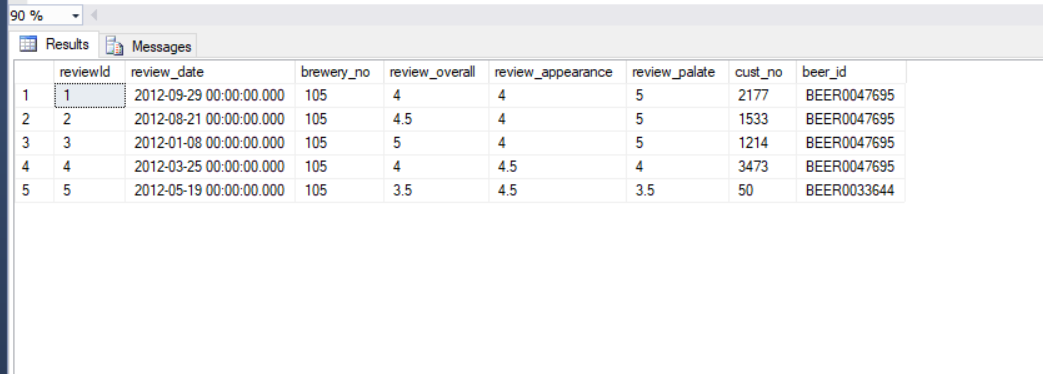
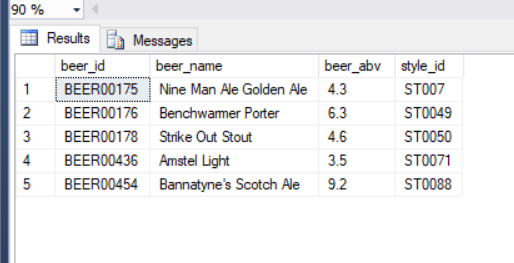
* All the execution of the test cases, snapshots and SQL queries are attached and described under Execution of test Cases and TSR section for the below listed test cases.

|  |  |
| --- | --- |
| 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 dimension tables |

Test Data Set

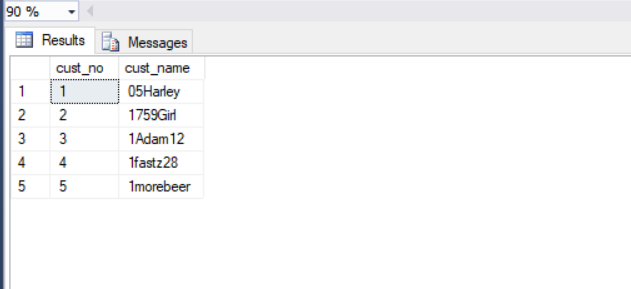
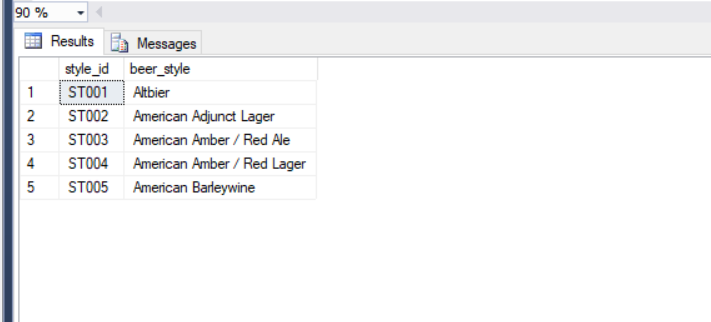
Before the development of the real data set and execution of the test cases, a small data set was derived from the Source and used for testing purposes to rectify issues in the process and to mitigate issues. The test data was loaded the same way as planned and tested in the below mentioned manner.

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

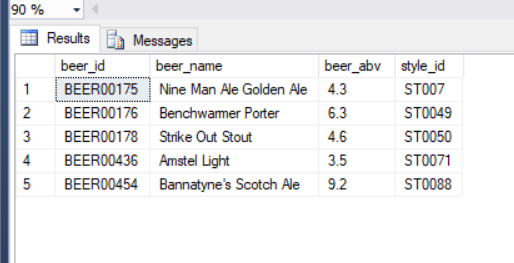
* Customer Dataset
* BeerStyles Dataset
* Brewery Dataset
* Review Dataset
* Beer Dataset

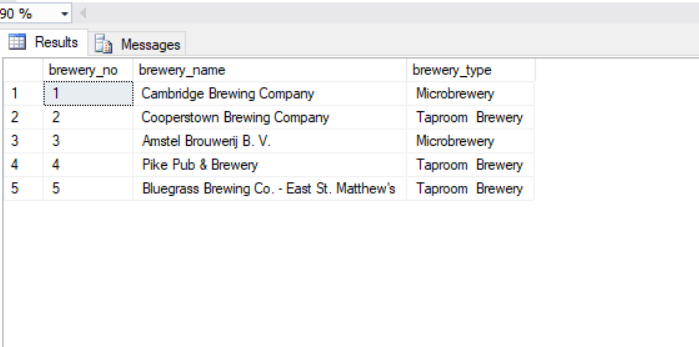
**Testing test data loaded from source to staging**

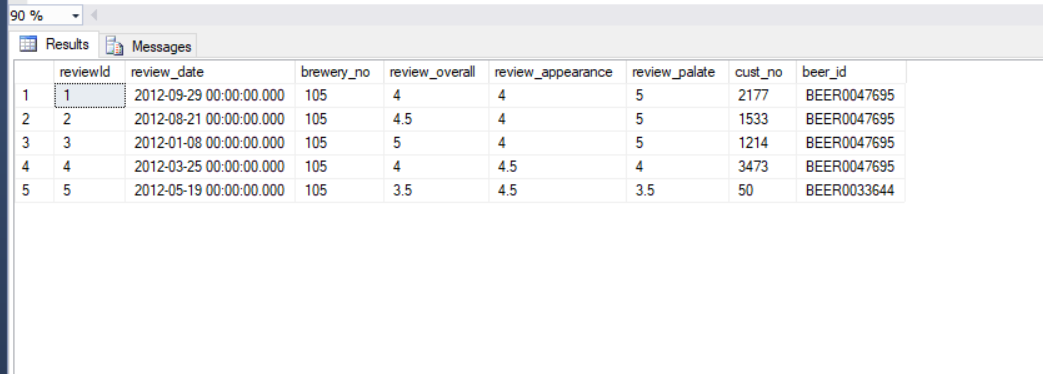
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Scenario ID** | | 1 | | | | | |
| **Test Case Description** | | Transform test data from source to staging tables | | | | | |
| **Pre-Requisite** | | Test Data loaded from source to staging tables in SQL tool | | | | | |
| **SNO** | **Action** | | **Sql Query** | **Expected Output** | **Actual Output** | **Test Result** | **Test Comments** |
| 1 | Data passed from customer source to Customer Staging | | select \* from Customer\_test; | All 5 rows displayed accordingly | All 5 rows displayed accordingly | Pass | Refer 1.1 attachment |
| 2 | Data passed from Styles source to Styles Staging | | select \* from Styles\_test; | All 5 rows displayed accordingly | All 5 rows displayed accordingly | Pass | Refer 1.2  attachment |
| 3 | Data passed from Beer source to Beer Staging | | select \* from Beer\_test; | All 5 rows displayed accordingly | All 5 rows displayed accordingly | Pass | Refer 1.3 attachment |
| 4 | Data passed from Brewery source to Brewery Staging | | select \* from Brewery\_test; | All 5 rows displayed accordingly | All 5 rows displayed accordingly | Pass | Refer 1.4 attachment |
| 5 | Data passed from Review source to Review Staging | | select \* from Review \_test; | All 5 rows displayed accordingly | All 5 rows displayed accordingly | Pass | Refer 1.5 attachment |

Attachment 1.1

Attachment 1.2

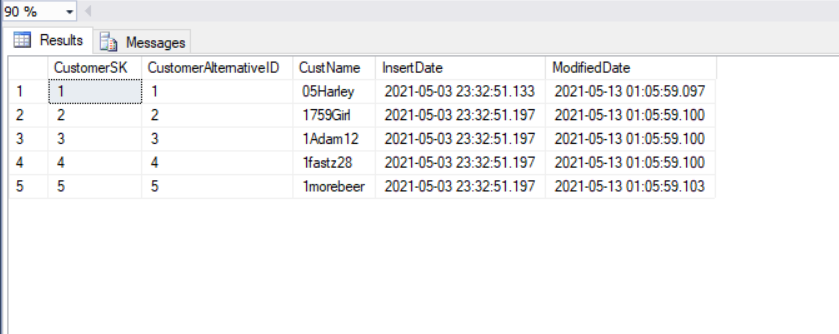
 Attachment 1.3

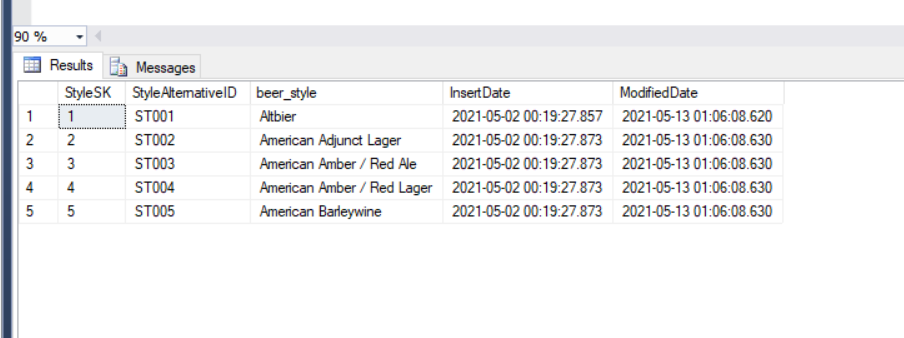
 Attachment 1.4



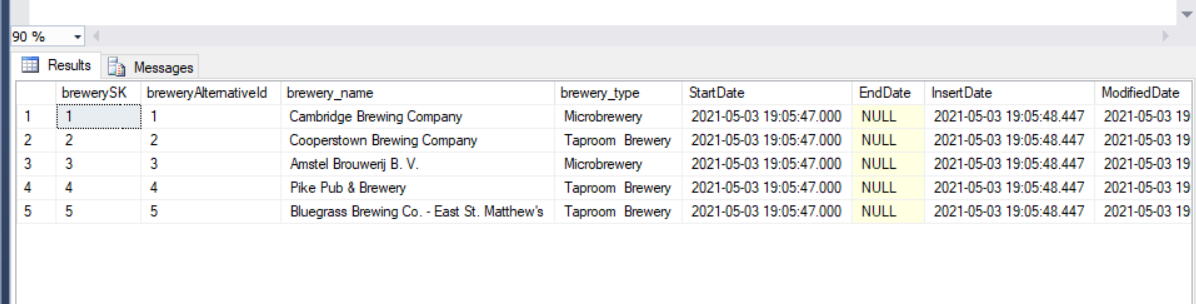
Attachment 1.5

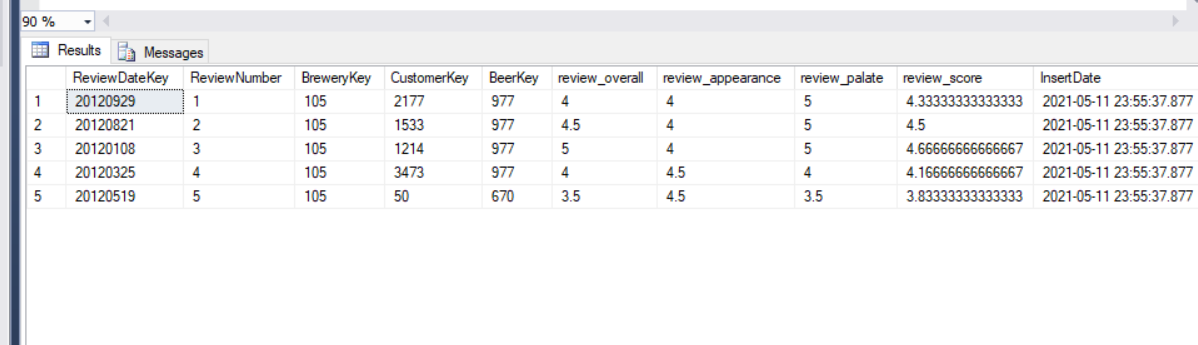
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Scenario ID** | | 2 | | | | | |
| **Test Case Description** | | Transform test data from staging to Dimension tables | | | | | |
| **Pre-Requisite** | | Test Data loaded from staging to dimension tables in SQL tool | | | | | |
| **SNO** | **Action** | | **Sql Query** | **Expected Output** | **Actual Output** | **Test Result** | **Test Comments** |
| 1 | Data passed from Stg customer to Customer Dimension | | select \* from DimCustomer; | All 5 rows displayed accordingly | All 5 rows displayed accordingly | Pass | Refer 2.1 attachment |
| 2 | Data passed from Stg Styles to Styles Dimension | | select \* from DimStyles; | All 5 rows displayed accordingly | All 5 rows displayed accordingly | Pass | Refer 2.2  attachment |
| 3 | Data passed from Stg Beer to Beer Dimension | | select \* from DimBeer; | All 5 rows displayed accordingly | All 5 rows displayed accordingly | Pass | Refer 2.3 attachment |
| 4 | Data passed from Stg Brewery to Brewery Dimension | | select \* from DimBrewery; | All 5 rows displayed accordingly | All 5 rows displayed accordingly | Pass | Refer 2.4 attachment |
| 5 | Data passed from Stg Review to Review Dimension | | select \* from DimReview; | All 5 rows displayed accordingly | All 5 rows displayed accordingly | Pass | Refer 2.5 attachment |

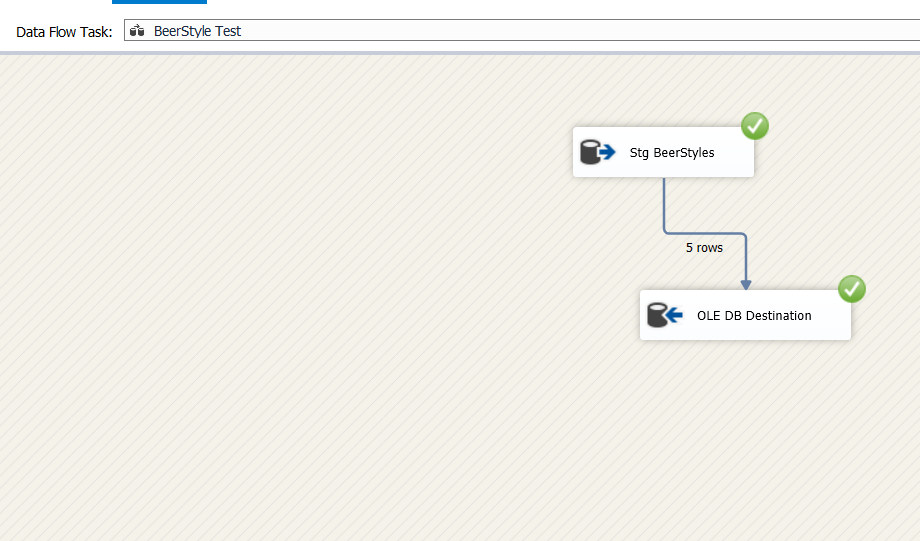
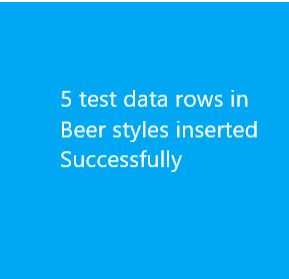
 Attachment 2.1

 Attachment 2.2

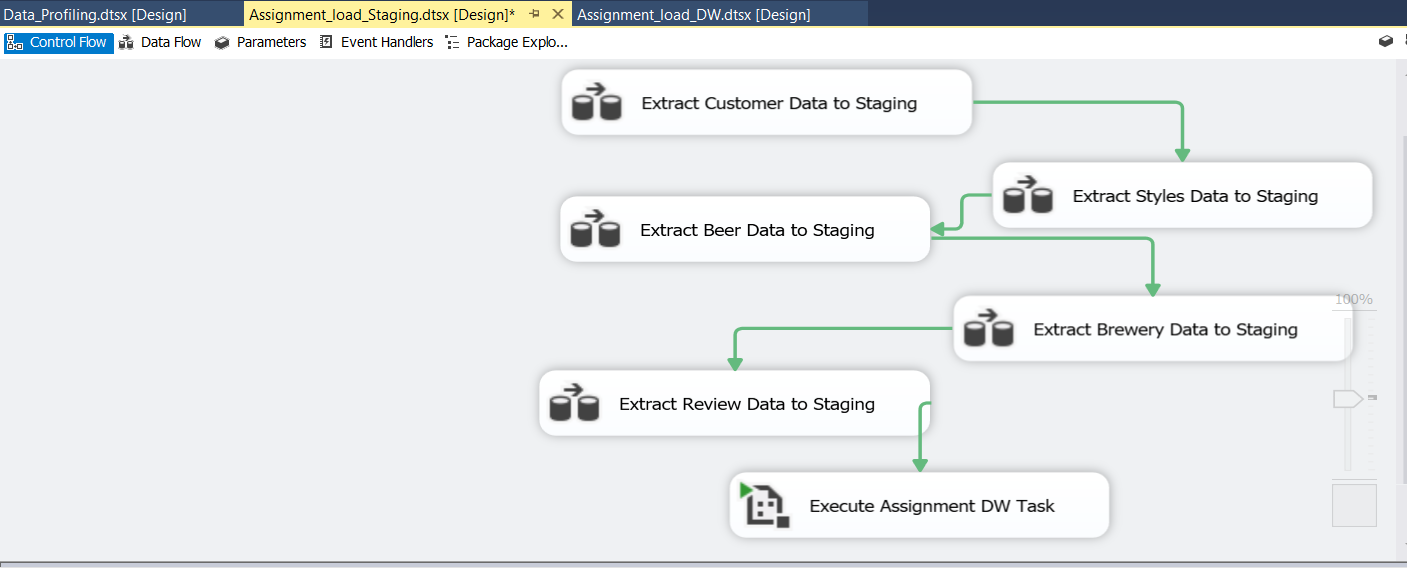
 Attachment 2.3

 Attachment 2.4

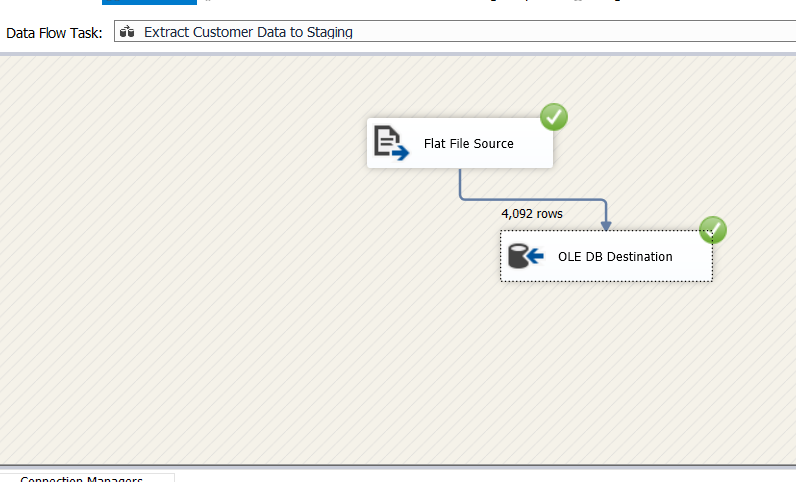
 Attachment 2.5

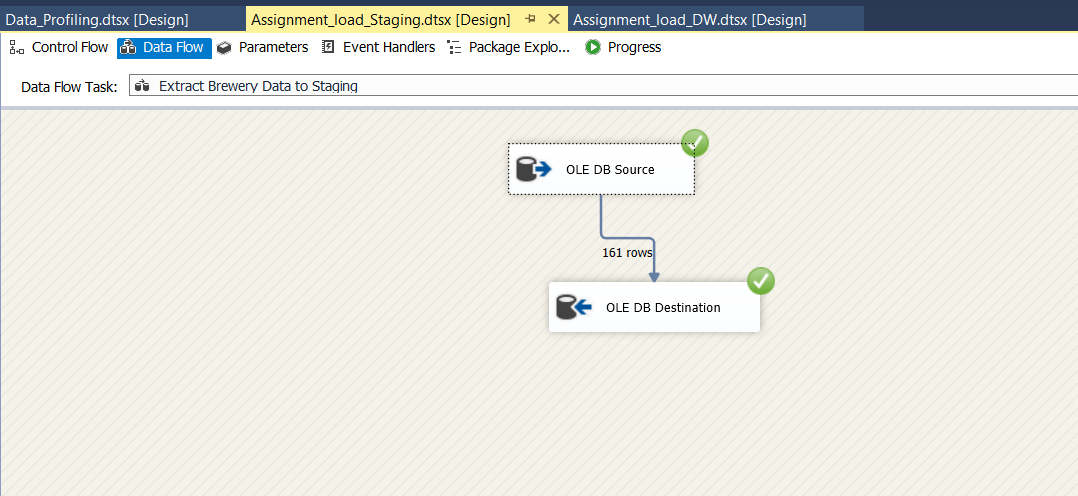
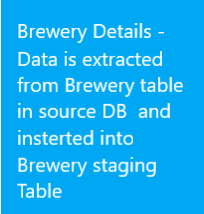
Please find below screenshots of the Some successfully executed transformation process of the test data set.

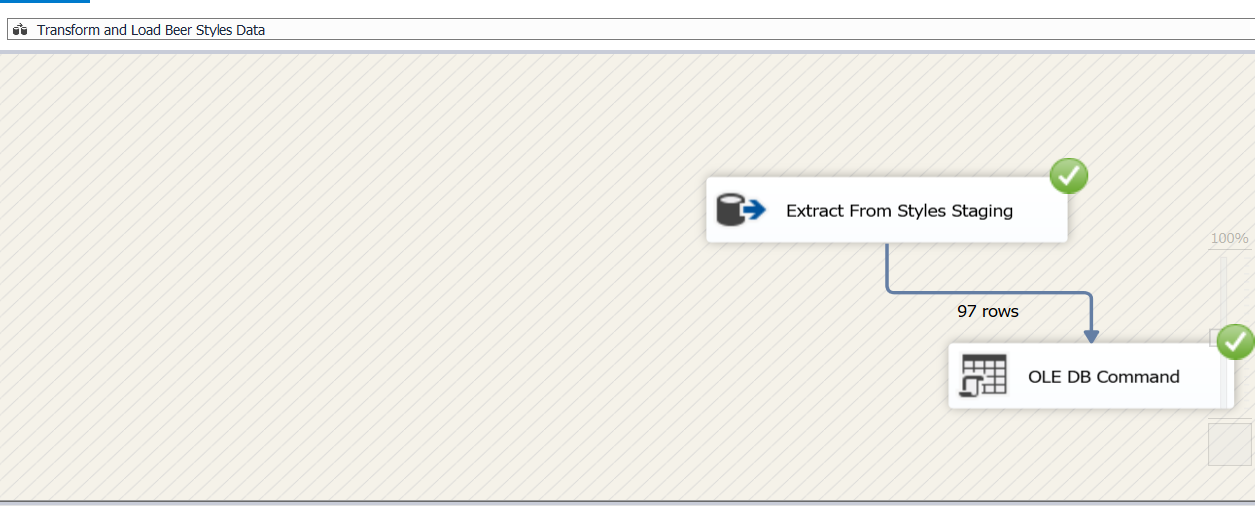
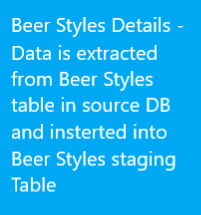
**[ETL Development](#ETL)**

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:

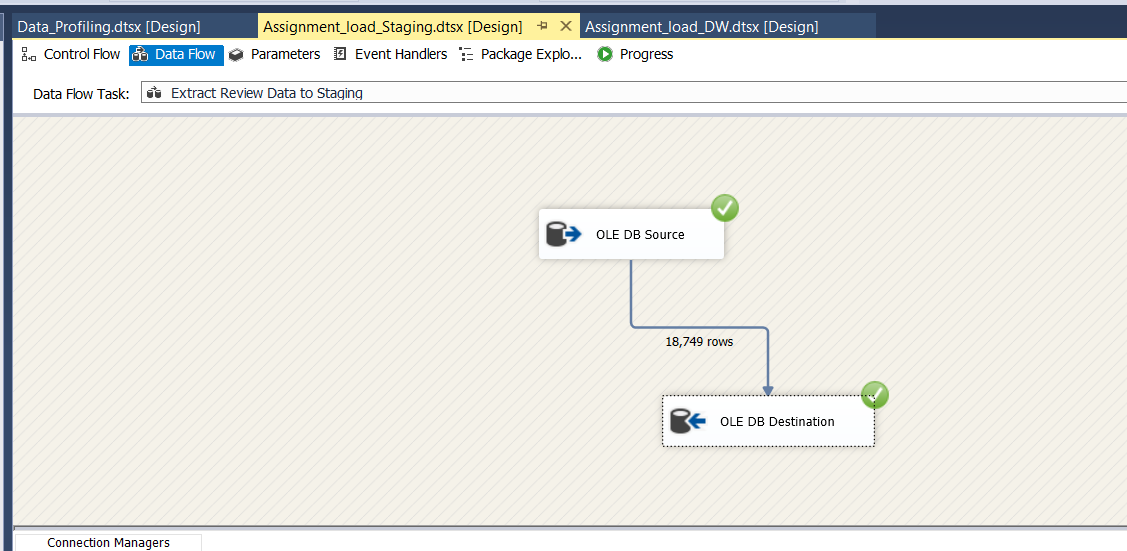
Screenshots of all the data sources that were staged and truncate tables created are attached below:

Staging customer details

Staging Brewery Data

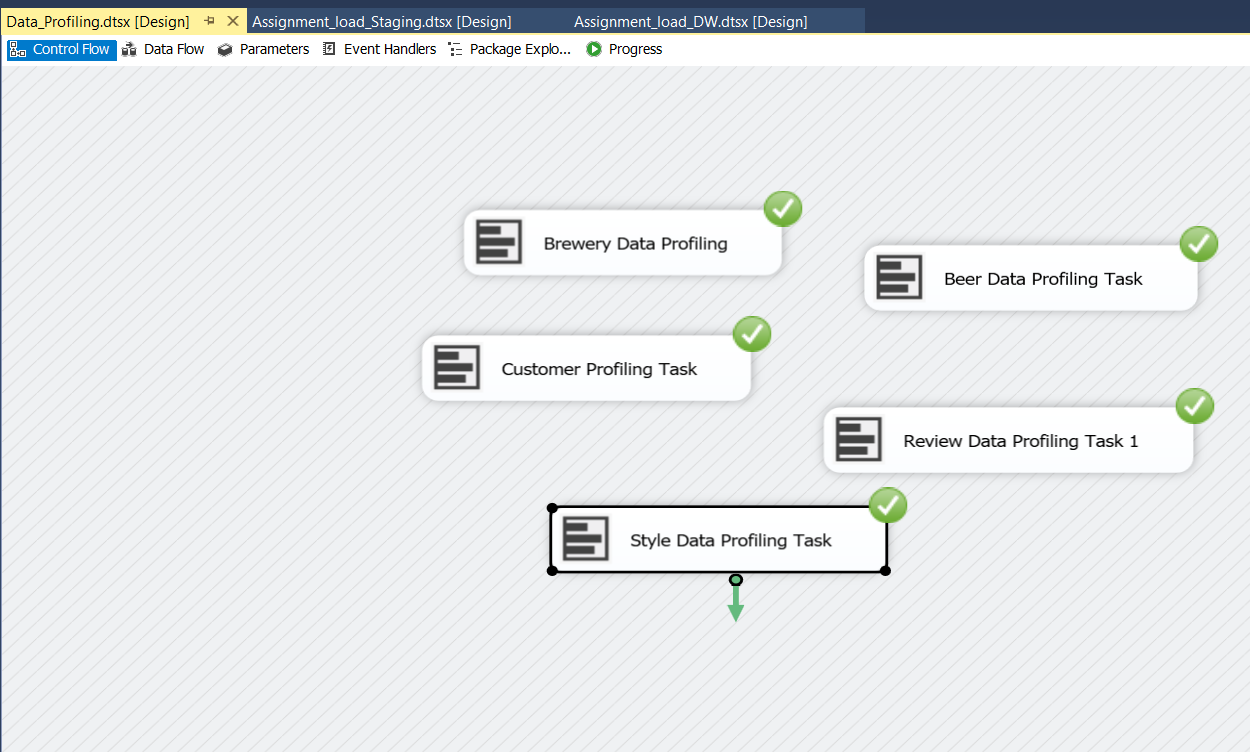
Staging Beer Styles Data

Staging Beer Data

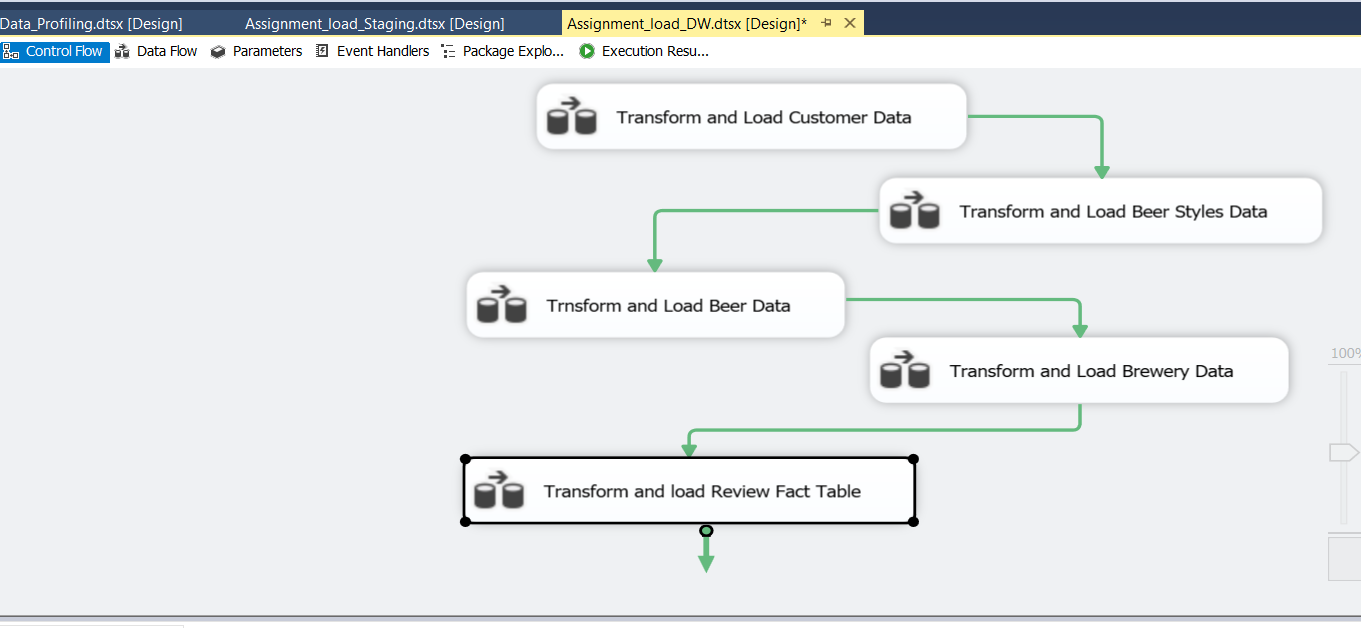
Staging Review Data

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

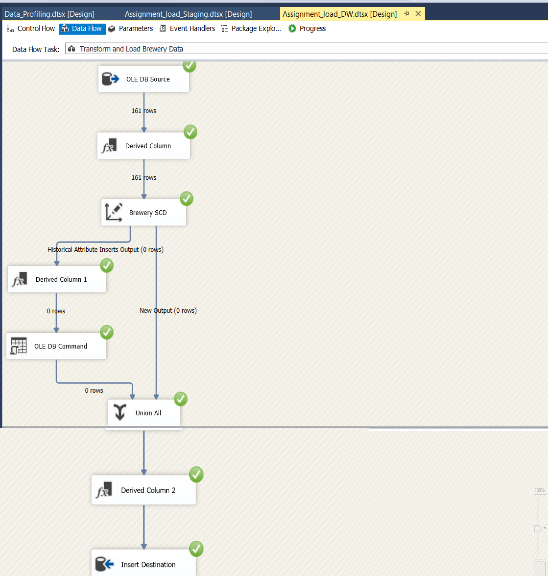
Every staging table is profiled and saved in a selected location.

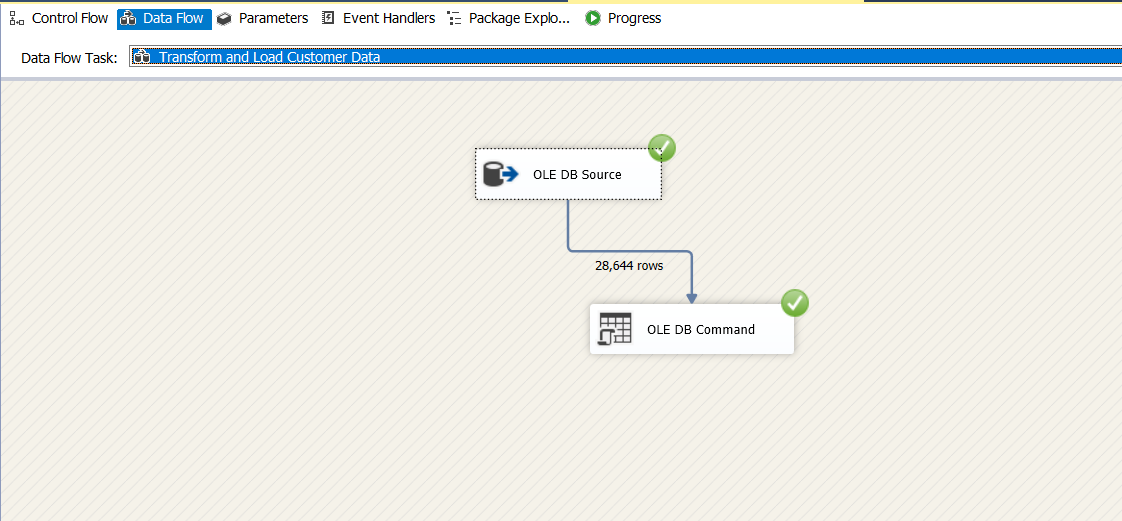
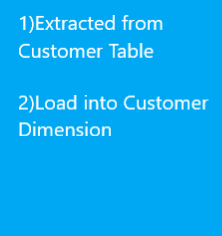
Next step is data transformation and as explained in a previous step, the execution task connected to the last data flow task of the first package is attached to the transformation package used for transformation.

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

Brewery type column was set as changing attribute.

After extracting data from the Brewery staging table, it was identified as a slowly changing dimension, it was connected as shown below and loaded data to the Brewery dimension table.



Next step was loaded from the Customer staging table to the Customer Dimension

The update procedure used to update Customer details:

USE [DWBI\_Assgnment1\_DW]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[UpdateDimCustomer] Script Date: 5/13/2021 5:04:58 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER PROCEDURE [dbo].[UpdateDimCustomer]

@CustomerID int,

@CustomerName nvarchar(50)

AS

BEGIN

if not exists (select CustomerSK

from dbo.DimCustomer

where CustomerAlternativeID = @CustomerID)

BEGIN

insert into dbo.DimCustomer

(CustomerAlternativeID, CustName, InsertDate, ModifiedDate)

values

(@CustomerID, @CustomerName, GETDATE(), GETDATE())

END;

if exists (select CustomerSK

from dbo.DimCustomer

where CustomerAlternativeID = @CustomerID)

BEGIN

update dbo.DimCustomer

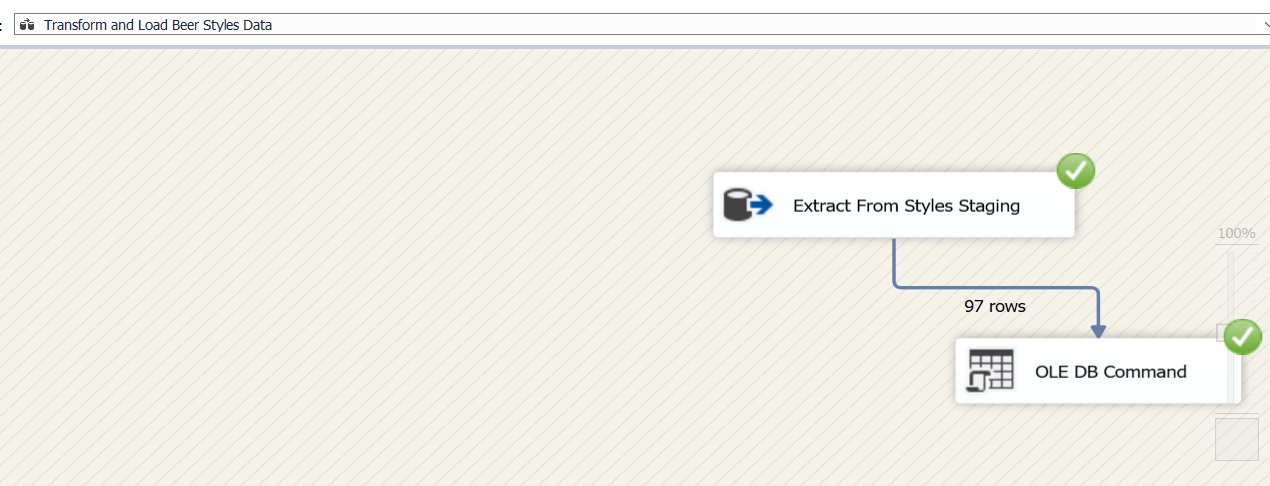
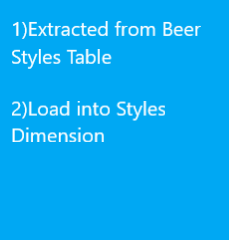
set CustName = @CustomerName,

ModifiedDate = GETDATE()

where CustomerAlternativeID = @CustomerID

END;

END;

Next step was loaded from the Beer Style staging table to the Style Dimension

The update procedure used to update Beer Style details:

USE [DWBI\_Assgnment1\_DW]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[UpdateDimBeerStyle] Script Date: 5/13/2021 5:07:14 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER PROCEDURE [dbo].[UpdateDimBeerStyle]

@StyleID varchar(16),

@BeerStyle varchar(255)

AS

BEGIN

if not exists (select StyleSK

from dbo.DimBeerStyle

where StyleAlternativeID = @StyleID)

BEGIN

insert into dbo.DimBeerStyle

(StyleAlternativeID, beer\_style, InsertDate, ModifiedDate)

values

(@StyleID, @BeerStyle, GETDATE(), GETDATE())

END;

if exists (select StyleSK

from dbo.DimBeerStyle

where StyleAlternativeID = @StyleID)

BEGIN

update dbo.DimBeerStyle

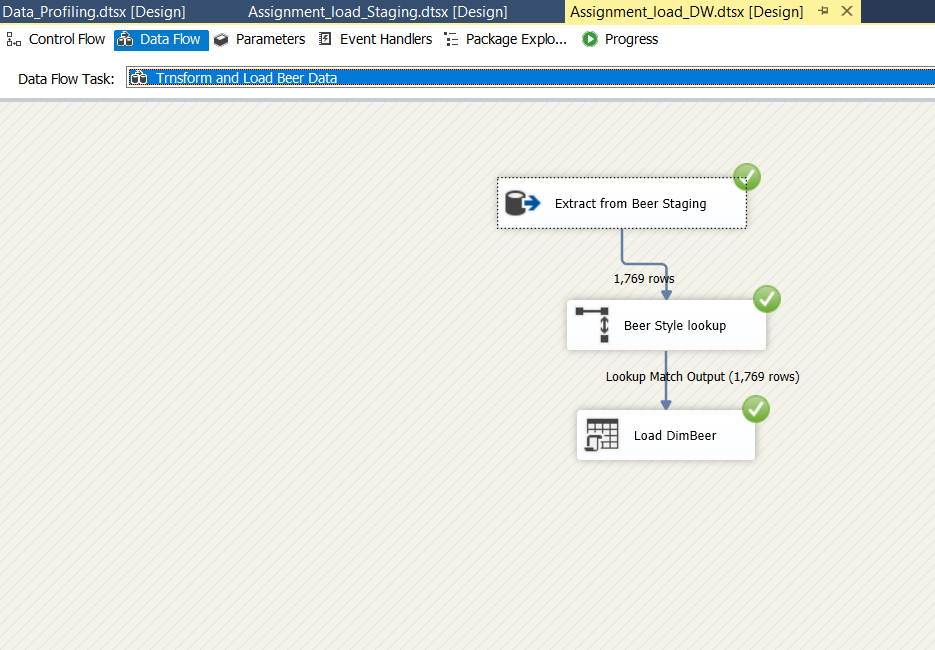
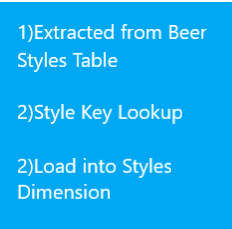
set beer\_style = @BeerStyle,

ModifiedDate = GETDATE()

where StyleAlternativeID = @StyleID

END;

END;

Then Beer staging table loaded to the Beer Dimension

The update procedure used to update Beer details:

USE [DWBI\_Assgnment1\_DW]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[UpdateDimBeer] Script Date: 5/13/2021 5:04:06 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER PROCEDURE [dbo].[UpdateDimBeer]

@BeerID varchar(29),

@BeerName nvarchar(255),

@BeerAbv float,

@beerStyleKey int

AS

BEGIN

if not exists (select beerSK

from dbo.DimBeer

where beerAlternativeId = @BeerID)

BEGIN

insert into dbo.DimBeer

(beerAlternativeId, beer\_name,beer\_abv,beerStyleKey, InsertDate, ModifiedDate)

values

(@BeerID, @BeerName,@BeerAbv,@beerStyleKey, GETDATE(), GETDATE())

END;

if exists (select beerSK

from dbo.DimBeer

where beerAlternativeId = @BeerID)

BEGIN

update dbo.DimBeer

set beer\_name = @BeerName,

beer\_abv = @BeerAbv,

@beerStyleKey = @beerStyleKey,

ModifiedDate = GETDATE()

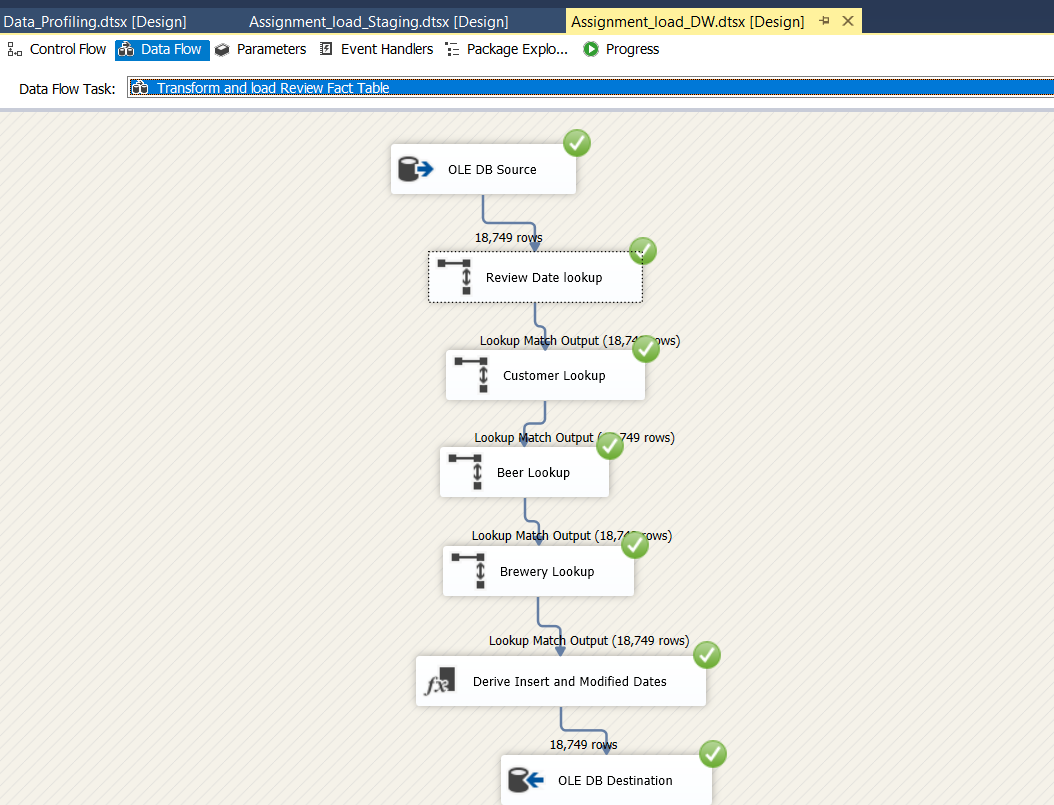
where beerAlternativeId = @BeerID

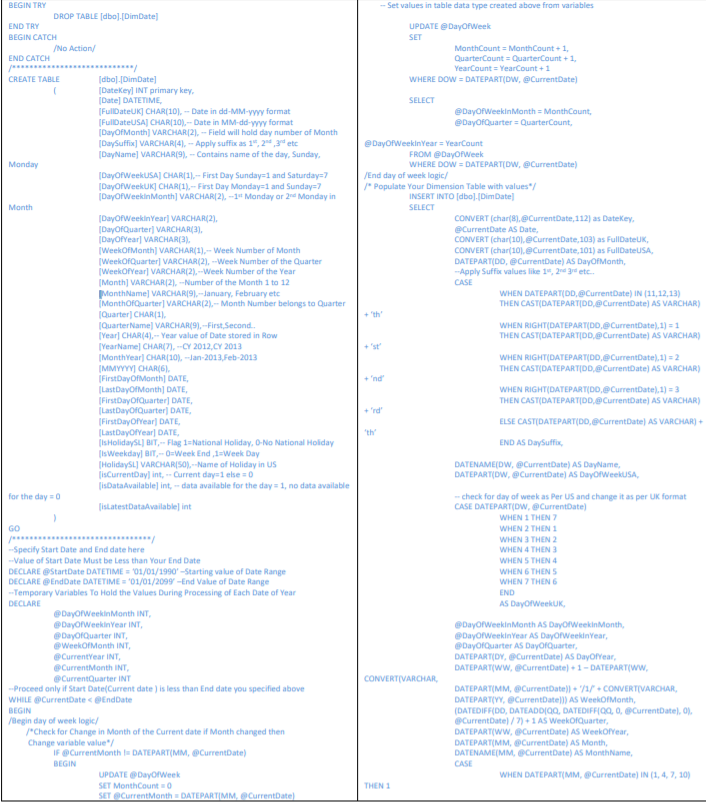
END;

END;

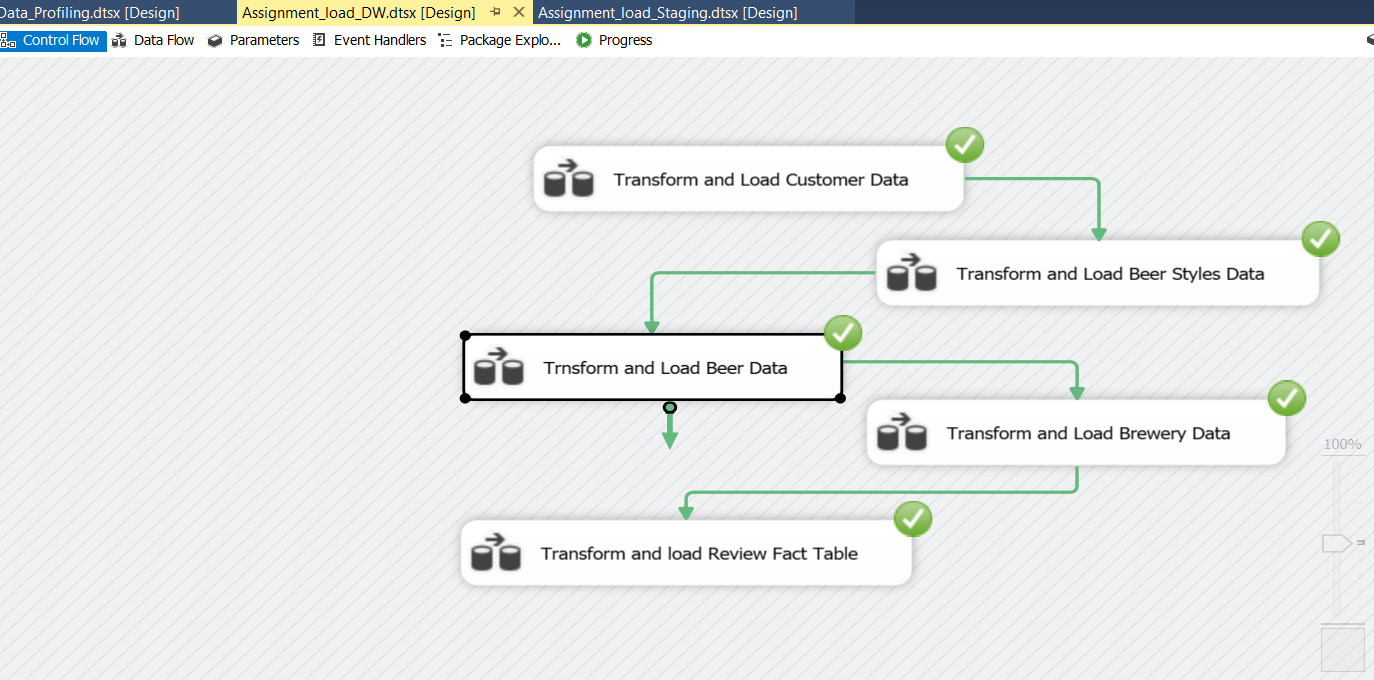
After loading to all the dimensions, lastly data was loaded to the fact table. The below steps were followed:

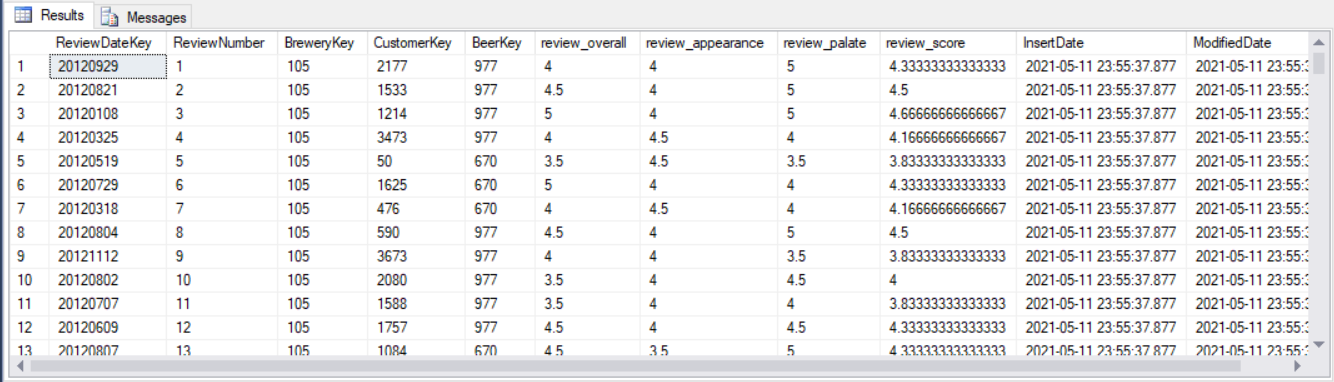
1. Data extracted from the customer transaction staging .
2. Review Day lookup from Date Dimension.
3. Customer lookup from Customer Dimension.
4. Beer lookup from Beer Dimension.
5. Brewery lookup from brewery Dimension.
6. Derive Inserted and modified date.
7. Finally insert Data to Review Fact table.



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



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

Print screen of the fact table:

• The column review Score is calculated the following way

Review\_Score = (review\_overall+review\_appearance+review\_palate)/3

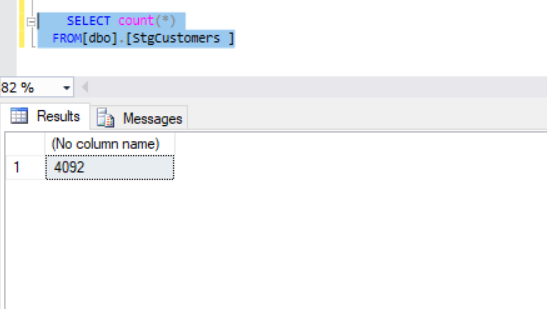
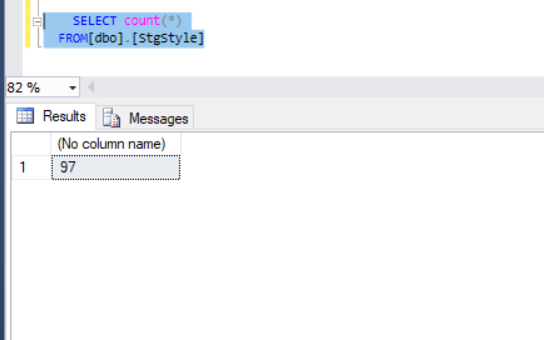
**[Execution of Test Cases and TSR](#ExecuteTest)**

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.

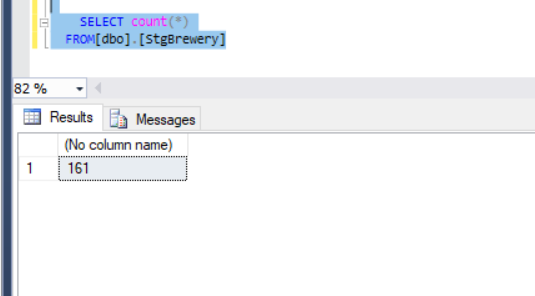
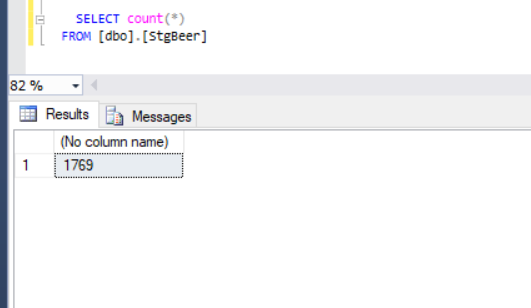
As mentioned earlier, ETL testing should be tested when data is being transformed from source to destinations not only at the two ends but also in the middle stages. In the test cases conducted it was tested that data was passed properly not only from source to staging but also from staging to the destination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

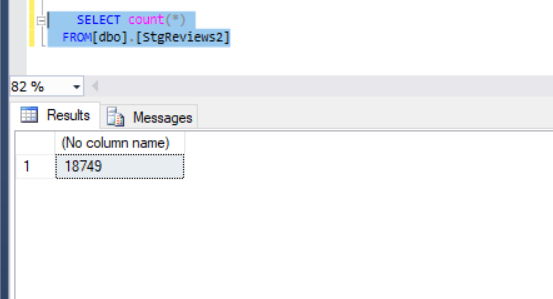
Execution of test cases

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Scenario ID** | | 3 | | | | | |
| **Test Case Description** | | Check for the count when transforming data from source to staging tables | | | | | |
| **Pre-Requisite** | | Data loaded from staging to dimension tables in SQL tool | | | | | |
| **SNO** | **Action** | | **Sql Query** | **Expected Output** | **Actual Output** | **Test Result** | **Test Comments** |
| 1 | Check for the count when transforming Customer source to Customer Staging | | select count(\*) from StgCustomer; | 4092 | 4092 | Pass | Refer 3.1 attachment |
| 2 | Check for the count when transforming data from Beer Style source to Style Staging | | select count(\*) from StgStyle; | 97 | 97 | Pass | Refer 3.2  attachment |
| 3 | Check for the count when transforming data from Beer source to Beer Staging | | select count(\*) from StgBeer; | 1769 | 1769 | Pass | Refer 3.3 attachment |
| 4 | Check for the count when transforming data from Brewery source to Brewery Staging | | select count(\*) from StgBrewery; | 161 | 161 | Pass | Refer 3.4 attachment |
| 5 | Check for the count when transforming data from Review source to Review Staging | | select count(\*) from StgReview; | 18749 | 18749 | Pass | Refer 3.5 attachment |



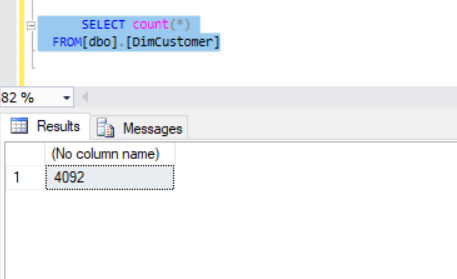
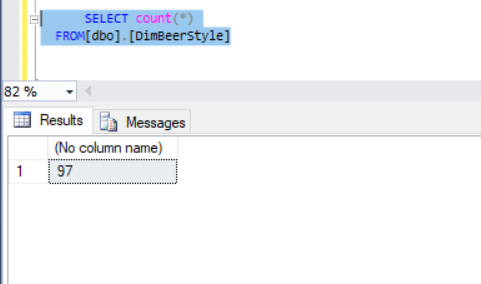
Attachment 3.1 Attachment 3.2

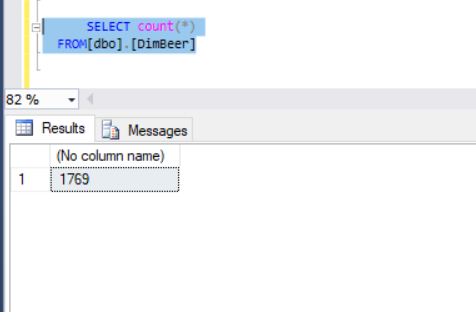
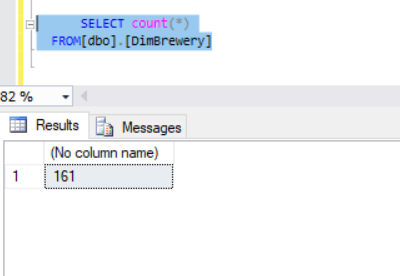


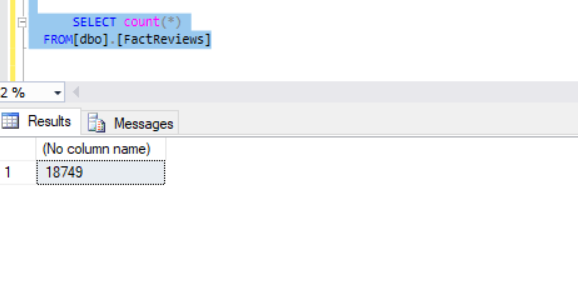
 Attachment 3.3 Attachment 3.4

Attachment 3.5

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **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 | | | | | |
| **SNO** | **Action** | | **Sql Query** | **Expected Output** | **Actual Output** | **Test Result** | **Test Comments** |
| 1 | Check for the count when transforming Customer staging to Customer Dimension | | select count(\*) from DimCustomer; | 4092 | 4092 | Pass | Refer 4.1 attachment |
| 2 | Check for the count when transforming data from Beer Style staging to Style Dimension | | select count(\*) from DimStyle; | 97 | 97 | Pass | Refer 4.2  attachment |
| 3 | Check for the count when transforming data from Beer staging to Beer Dimension | | select count(\*) from DimBeer; | 1769 | 1769 | Pass | Refer 4.3 attachment |
| 4 | Check for the count when transforming data from Brewery staging to Brewery Dimension | | select count(\*) from DimBrewery; | 161 | 161 | Pass | Refer 4.4 attachment |
| 5 | Check for the count when transforming data from Review staging to Review Dimension | | select count(\*) from FactReview; | 18749 | 18749 | Pass | Refer 4.5 attachment |

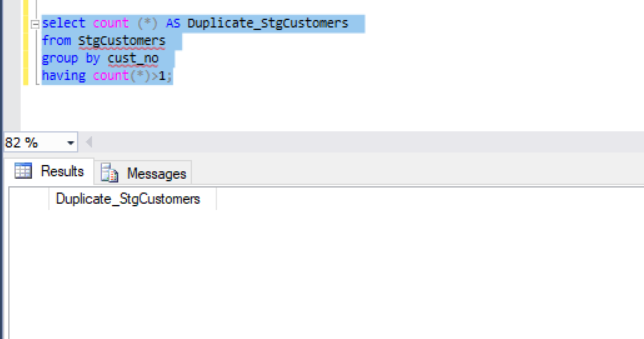
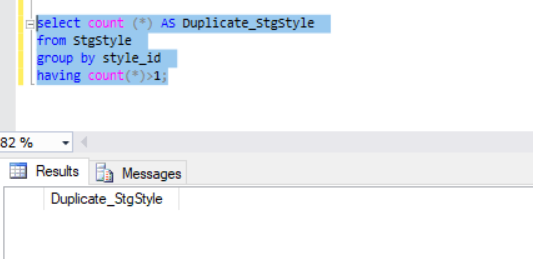
 Attachment 4.1 Attachment 4.2



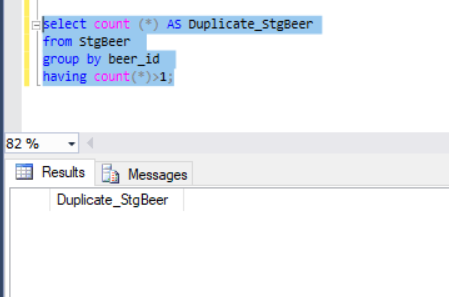
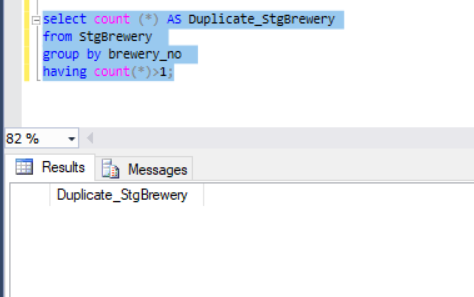
Attachment 4.3 Attachment 4.4

Attachment 4.5

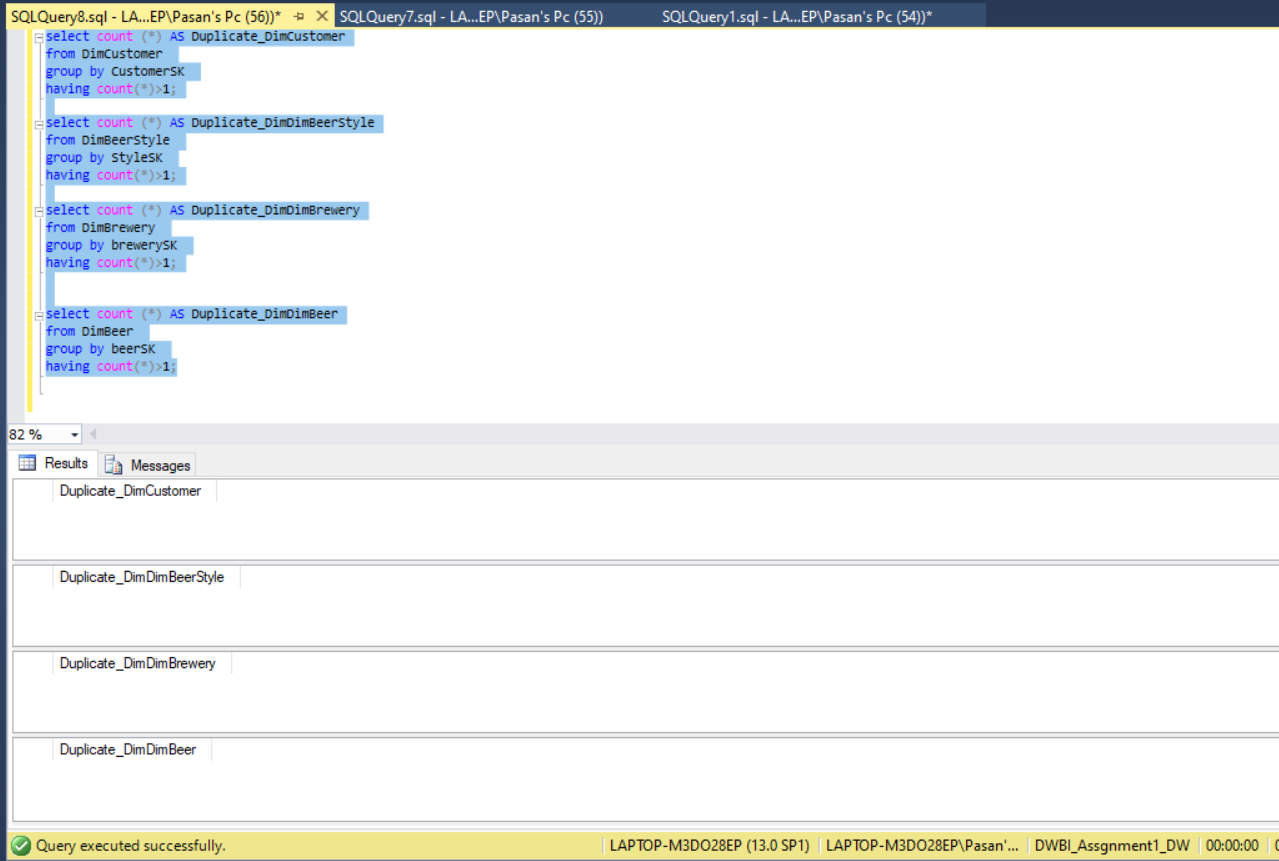
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Scenario ID** | | 5 | | | | |
| **Test Case Description** | | Check for duplicate values in the staging tables. | | | | |
| **Pre-Requisite** | | Data loaded from source to staging tables in SQL tool | | | | |
| **SNO** | **Action** | **Sql Query** | **Expected Output** | **Actual Output** | **Test Result** | **Test Comments** |
| 1 | Check whether data has got duplicated in Customer staging | select count (\*) AS Duplicate\_StgCustomers from StgCustomers group by cust\_no having count(\*)>1; | 0 | 0 | Pass | Refer 5.1 attachment |
| 2 | Check whether data has got duplicated in BeerStyle staging | select count (\*) AS Duplicate\_StgStyle from StgStyle group by style\_id having count(\*)>1; | 0 | 0 | Pass | Refer 5.2  attachment |
| 3 | Check whether data has got duplicated in Beer staging | select count (\*) AS Duplicate\_StgBeer from StgBeer group by beer\_id having count(\*)>1; | 0 | 0 | Pass | Refer 5.3 attachment |
| 4 | Check whether data has got duplicated in Brewery staging | select count (\*) AS Duplicate\_StgBrewery from StgBrewery group by brewery\_no having count(\*)>1; | 0 | 0 | Pass | Refer 5.4 attachment |



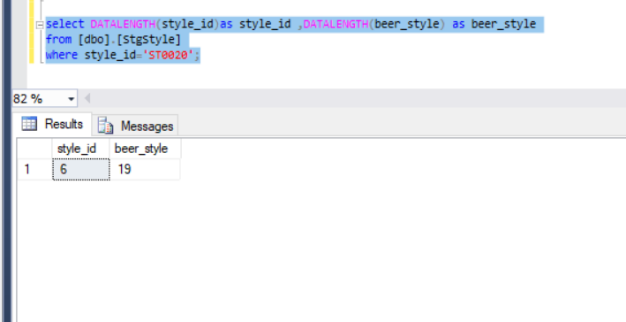
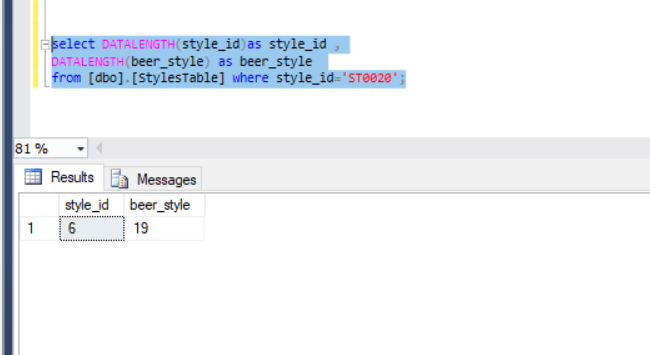
Attachment 5.1 Attachment 5.2

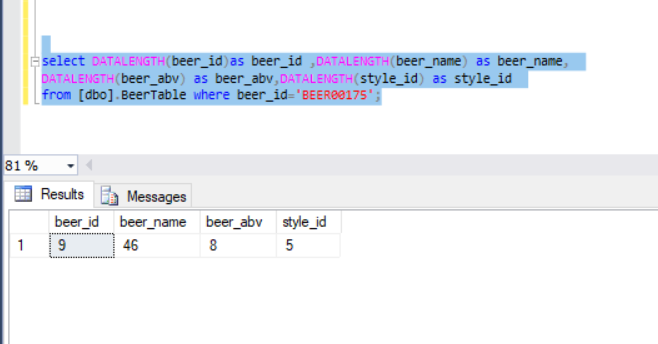
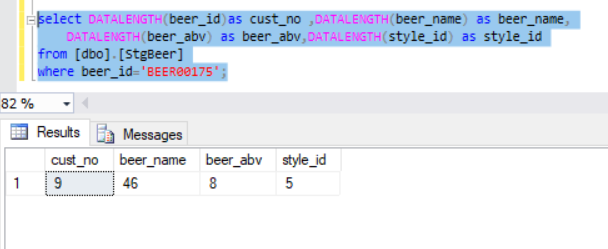
Attachment 5.3 Attachment 5.4

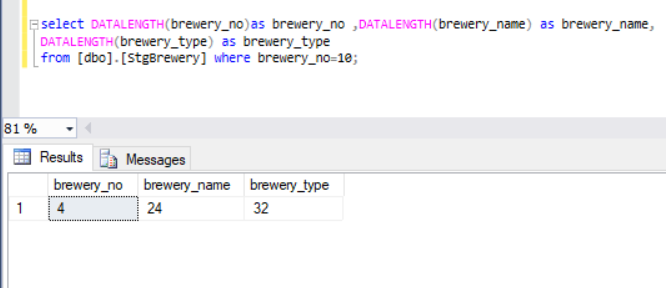
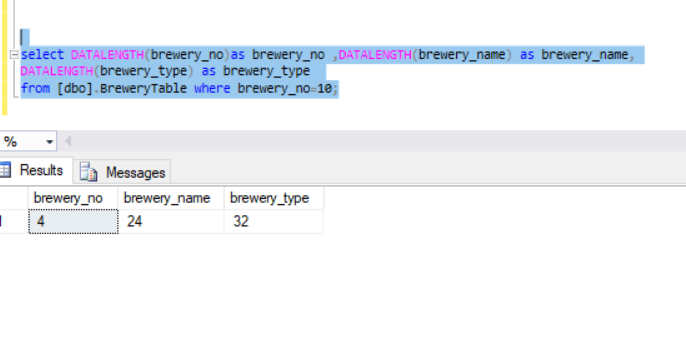
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Scenario ID** | | 6 | | | | |
| **Test Case Description** | | Check for duplicate values in the Dimension tables. | | | | |
| **Pre-Requisite** | | Data loaded from source to staging tables in SQL tool | | | | |
| **SNO** | **Action** | **Sql Query** | **Expected Output** | **Actual Output** | **Test Result** | **Test Comments** |
| 1 | Check whether data has got duplicated in Customer Dimension | select count (\*) AS Duplicate\_DimCustomers from DimCustomers group by customerSK having count(\*)>1; | 0 | 0 | Pass | Refer 6  attachment |
| 2 | Check whether data has got duplicated in BeerStyle Dimension | select count (\*) AS Duplicate\_DimStyle from DimStyle group by styleSK having count(\*)>1; | 0 | 0 | Pass | Refer 6  attachment |
| 3 | Check whether data has got duplicated in Beer Dimension | select count (\*) AS Duplicate\_DimBeer from DimBeer group by beerSK having count(\*)>1; | 0 | 0 | Pass | Refer 6 attachment |
| 4 | Check whether data has got duplicated in Brewery Dimension | select count (\*) AS Duplicate\_DimBrewery from DimBrewery group by brewerySK having count(\*)>1; | 0 | 0 | Pass | Refer 6 attachment |

 Attachment 6

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **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 | | | | |
| **SNO** | **Action** | **Sql Query** | **Expected Output** | **Actual Output** | **Test Result** | **Test Comments** |
| 1 | Check whether the data lengths in the BeerStyles source table and BeerStyles Staging table are the same | select DATALENGTH(style\_id)as style\_id ,DATALENGTH(beer\_style) as beer\_style from [dbo].[StgStyle] where style\_id='ST0020'; | Style\_id=6  Beer\_style=19 | Style\_id=6  Beer\_style=19 | Pass | Refer 7.1  attachment |
| 2 | Check whether the data lengths in the Beer source table and Beer Staging table are the same | select DATALENGTH(beer\_id)as beer\_id ,DATALENGTH(beer\_name) as beer\_name, DATALENGTH(beer\_abv) as beer\_abv,DATALENGTH(style\_id) as style\_id from [dbo].[StgBeer] where beer\_id='BEER00175'; | Beer\_id=9  Beer\_name=46  Beer\_abv=8  Beer\_style=5 | Beer\_id=9  Beer\_name=46  Beer\_abv=8  Beer\_style=5 | Pass | Refer 7.2 attachment |
| 3 | Check whether the data lengths in the Brewery source table and Brewery Staging table are the same | select DATALENGTH(brewery\_no)as brewery\_no ,DATALENGTH(brewery\_name) as brewery\_name,DATALENGTH(brewery\_type) as brewery\_type from [dbo].[StgBrewery] where brewery\_no=10; | Brewery\_no=4  Breery\_name=24  Brewery\_type=32 | Brewery\_no=4  Breery\_name=24  Brewery\_type=32 | Pass | Refer 7.3 attachment |

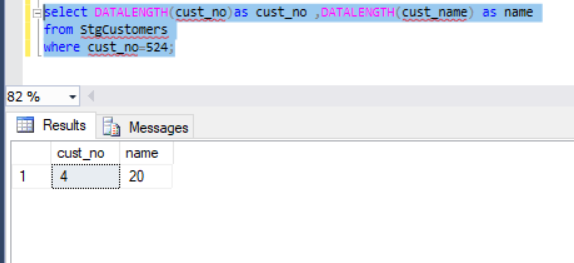


Attachment 7.1

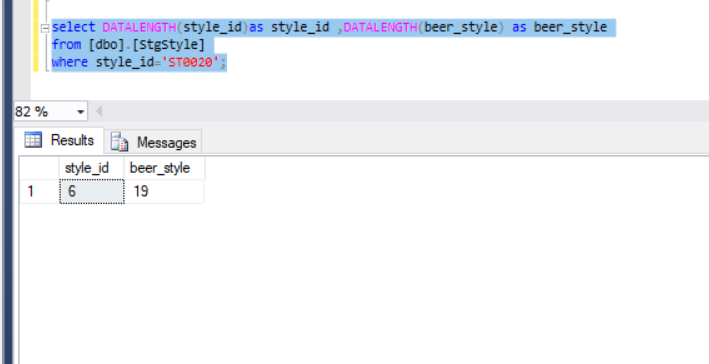
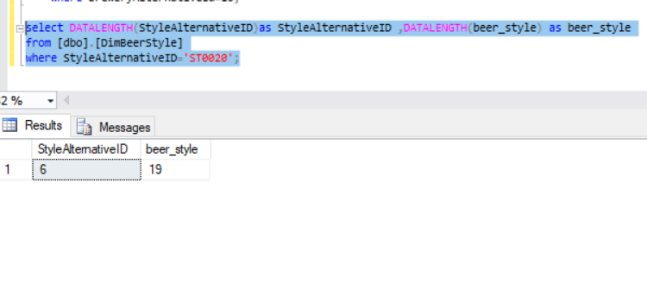
Attachment 7.2

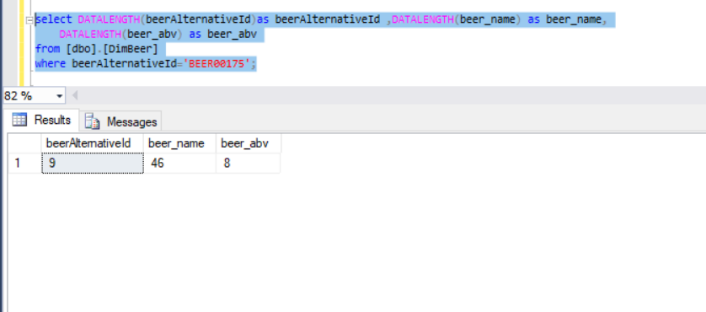
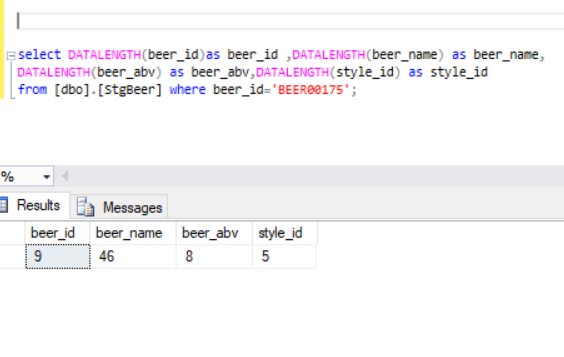
Attachment 7.3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Scenario ID** | | 8 | | | | |
| **Test Case Description** | | Data length check for data in dimension tables | | | | |
| **Pre-Requisite** | | Data loaded from source to staging tables in SQL tool | | | | |
| **SNO** | **Action** | **Sql Query** | **Expected Output** | **Actual Output** | **Test Result** | **Test Comments** |
| 1 | Check whether the data lengths in the Customer Staging table and Customer Dimension are the same | select DATALENGTH(CustomerAlternativeID)as CustomerAlternativeID ,DATALENGTH(CustName) as name from[dbo].[DimCustomer] where CustomerAlternativeID=524; | Cust\_no=4  Cust\_name=20 | Cust\_no=4  Cust\_name=20 | Pass | Refer 8.1 attachment |
| 2 | Check whether the data lengths in the BeerStyles Staging table and BeerStyles Dimension table are the same | select DATALENGTH(StyleAlternativeID)as StyleAlternativeID ,DATALENGTH(beer\_style) as beer\_style from [dbo].[DimBeerStyle] where StyleAlternativeID='ST0020'; | Style\_id=6  Beer\_style=19 | Style\_id=6  Beer\_style=19 | Pass | Refer 8.2  attachment |
| 3 | Check whether the data lengths in the Beer Staging table and Beer Dimension table are the same | select DATALENGTH(beerAlternativeId)as beerAlternativeId ,DATALENGTH(beer\_name) as beer\_name, DATALENGTH(beer\_abv) as beer\_abv from [dbo].[DimBeer] where beerAlternativeId='BEER00175'; | Beer\_id=9  Beer\_name=46  Beer\_abv=8  Beer\_style=5 | Beer\_id=9  Beer\_name=46  Beer\_abv=8  Beer\_style=5 | Pass | Refer 8.3 attachment |
| 4 | Check whether the data lengths in the Brewery Staging table and Brewery Dimension table are the same | select DATALENGTH(breweryAlternativeId)as breweryAlternativeId ,DATALENGTH(brewery\_type) as brewery\_type from [dbo].[DimBrewery] where breweryAlternativeId=10 ) as beer\_abv,DATALENGTH(style\_id) as style\_id from [dbo].[StgBeer] where beer\_id='BEER00175'; | Brewery\_no=4  Brewery\_type=32 | Brewery\_no=4  Brewery\_type=32 | Pass | Refer 8.4 attachment |

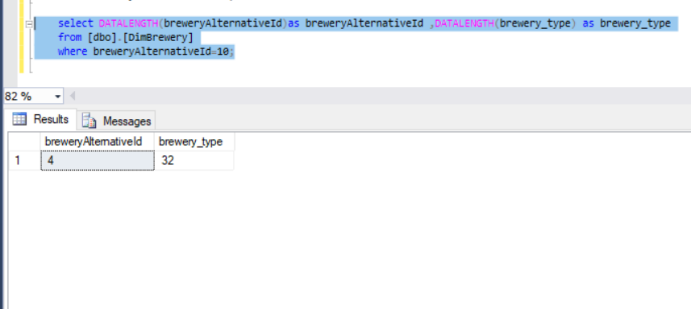
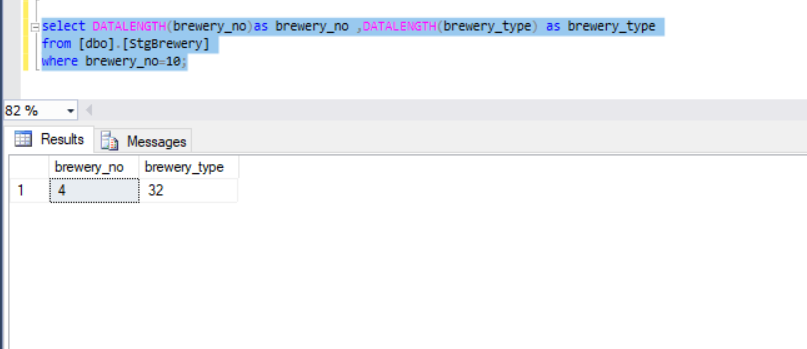


Attachment 8.1



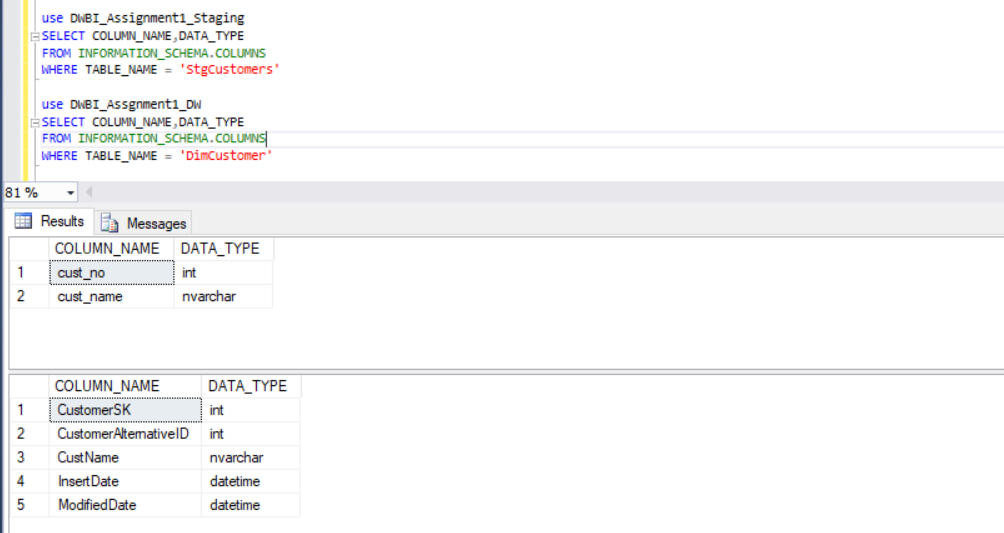
Attachment 8.2

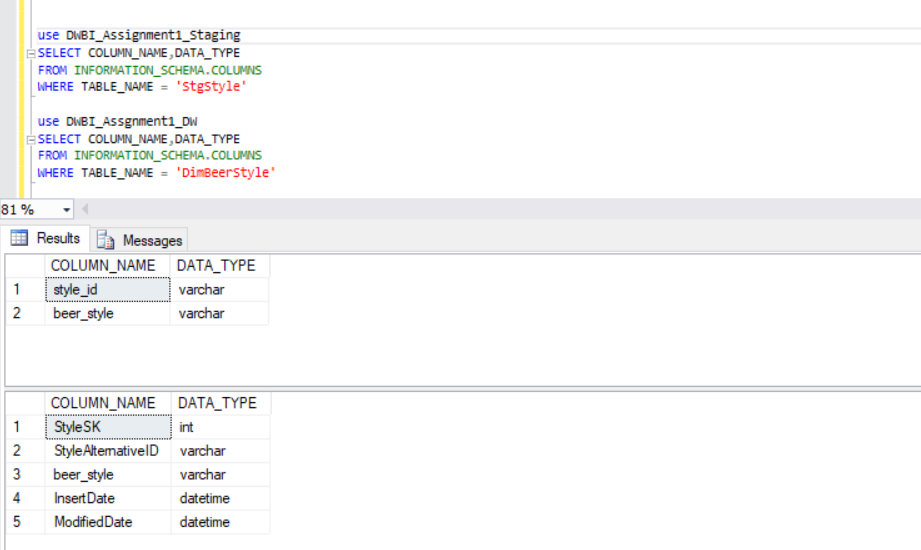
Attachment 8.3



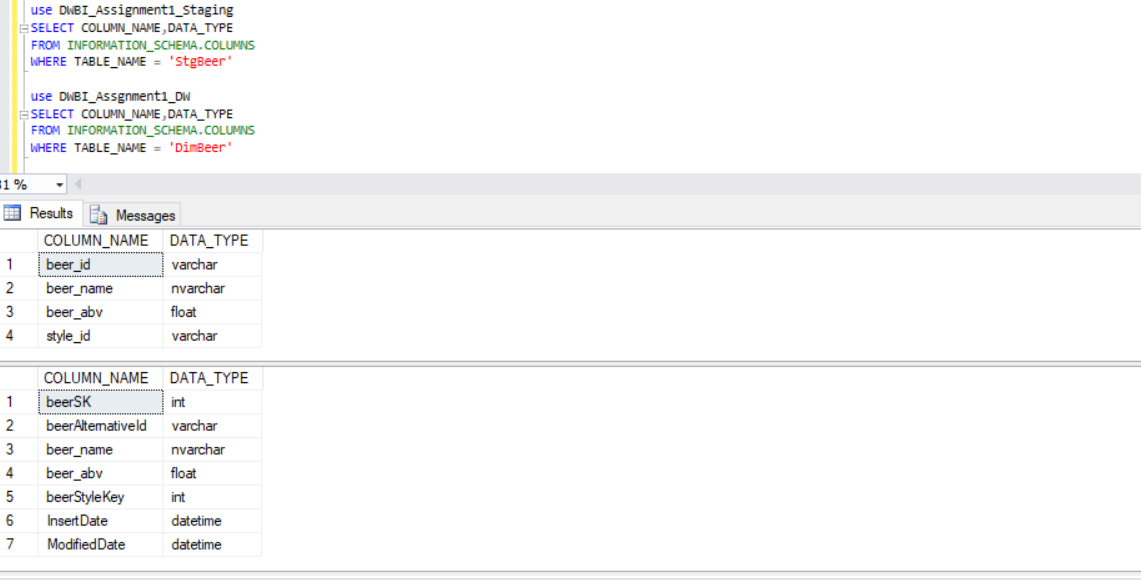
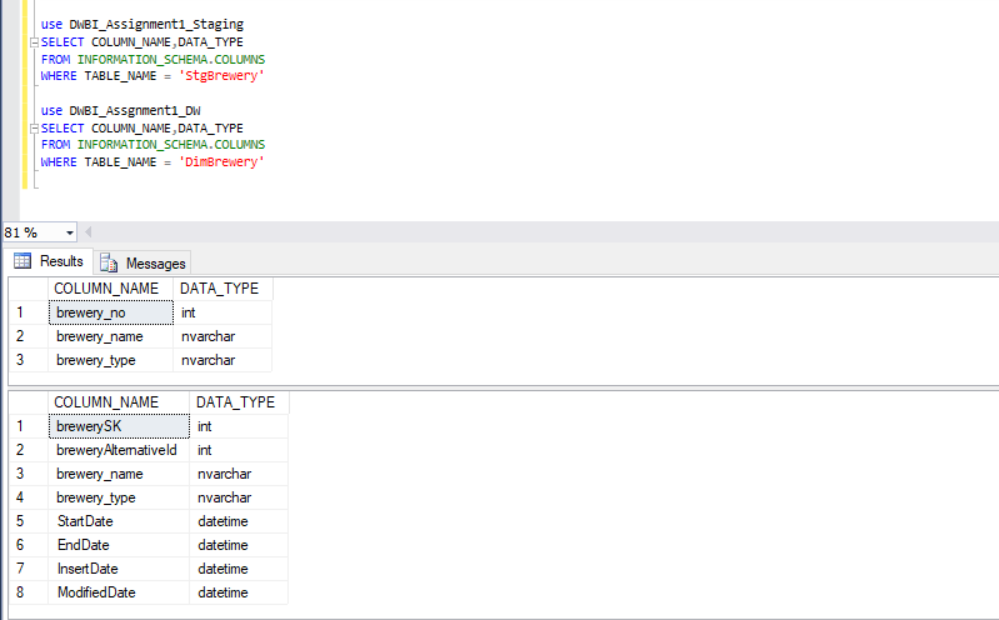
Attachment 8.4

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Scenario ID** | | 9 | | | | |
| **Test Case Description** | | Data type check for data in dimension tables | | | | |
| **Pre-Requisite** | | Data loaded from source to staging tables in SQL tool | | | | |
| **SNO** | **Action** | **Sql Query** | **Expected Output** | **Actual Output** | **Test Result** | **Test Comments** |
| 1 | Check whether the data types in the customer staging table and customer dimension table are the same | use DWBI\_Assgnment1\_DW SELECT COLUMN\_NAME,DATA\_TYPE FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = ‘DimCustomer’; | Cust\_no=int  Cust\_name=nvarchar | Cust\_no=4  Cust\_name=20 | Pass | Data types tally  Refer 9.1 attachment |
| 2 | Check whether the data types in the BeerStyles Staging table and BeerStyles Dimension table are the same | use DWBI\_Assgnment1\_DW SELECT COLUMN\_NAME,DATA\_TYPE FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = ‘DimStyles’ | Style\_id=varchar  Beer\_style=varchar | Style\_id=6  Beer\_style=19 | Pass | Data types tally  Refer 9.2  attachment |
| 3 | Check whether the data types in the Beer Staging table and Beer Dimension table are the same | use DWBI\_Assgnment1\_DW SELECT COLUMN\_NAME,DATA\_TYPE FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = ‘DimBeer’ | Beer\_id=varchar  Beer\_name=nvarchar  Beer\_abv=float  Beer\_style=varchar | Beer\_id=9  Beer\_name=46  Beer\_abv=8  Beer\_style=5 | Pass | Data types tally  Refer 9.3 attachment |
| 4 | Check whether the data types in the Brewery Staging table and Brewery Dimension table are the same | use DWBI\_Assgnment1\_DW SELECT COLUMN\_NAME,DATA\_TYPE FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'DimBrewery' | Brewery\_no=int  Brewery\_name=nvarchar  Brewery\_type=nvarchar | Brewery\_no=4  Brewery\_type=32 | Pass | Data types tally  Refer 9.4 attachment |

 Attachment 9.1



Attachment 9.2

 Attachment 9.3

Attachment 9.4