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

Solution Architecture ……6

Data Warehouse Design & Development……..7

Test Planning & Test Data……8

ETL Development……..18

Execution of Test Cases and TSR……..31

Data Selection & Preparation

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

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.

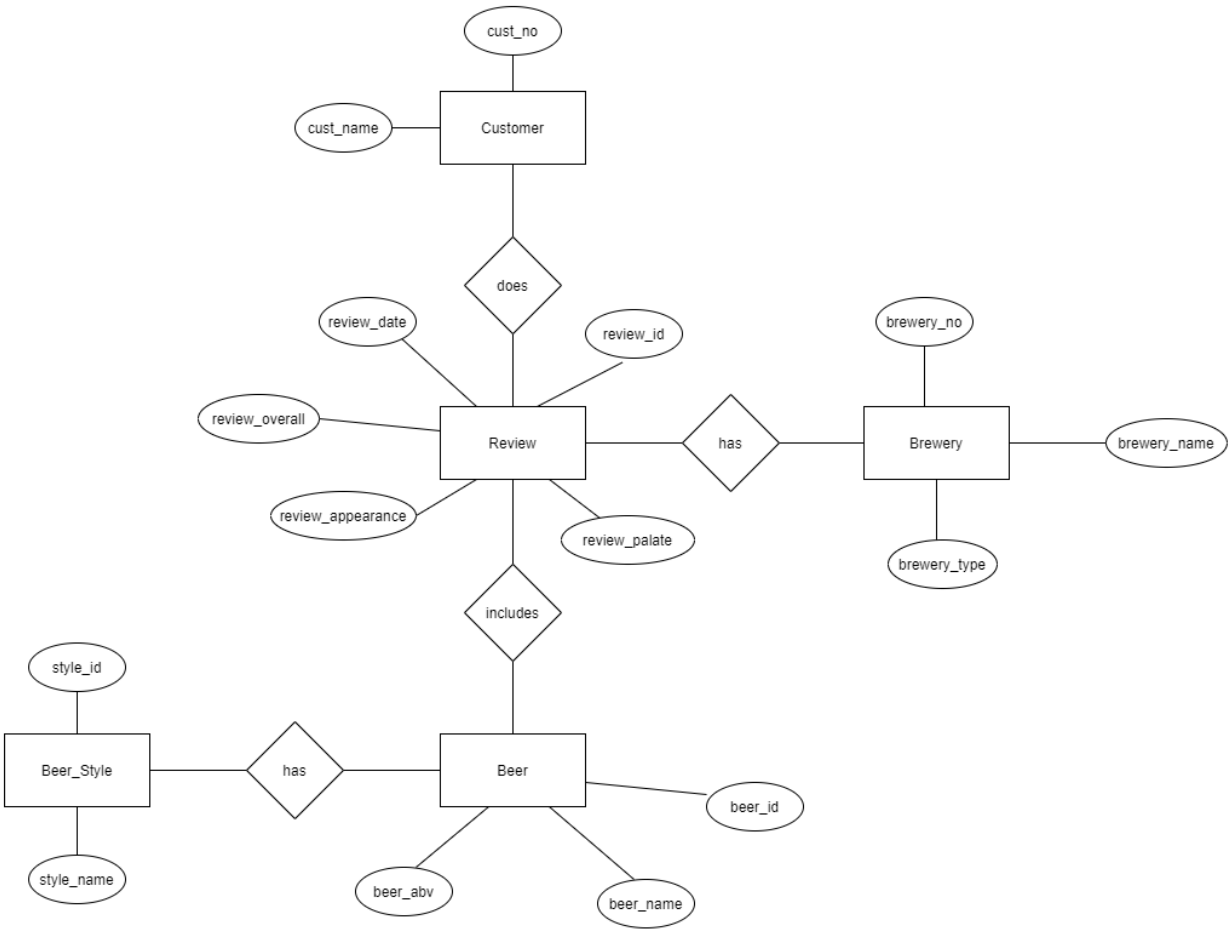
Also, the below mentioned CSV files were imported to the SQL source database.

- Beer Details.
- Brewery Details.
- Beer Style Details.
- Review Details.

Description of The Dataset

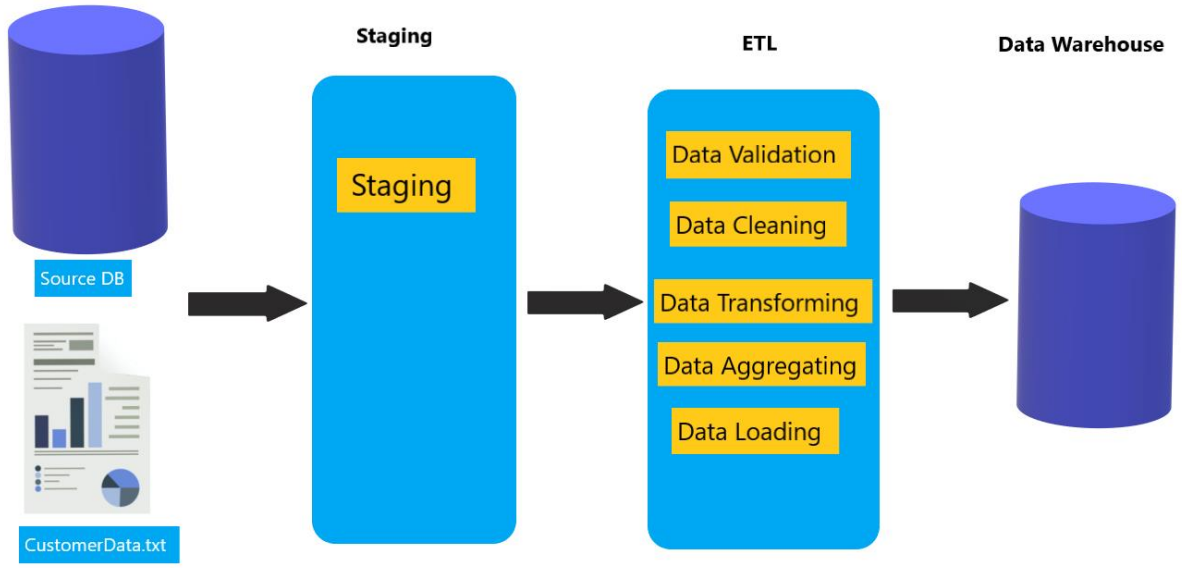
Table Name	**Column Name**	**Data type**	**Description**
Beer	Beer_id	Varchar(16)	Details of Beers
	Beer_name	Nvarchar(255)	
	Beer_abv	Float	
	Style_id	Varchar(16)	
Beer Style	Style_id	varchar()16	Details of Beer Styles(Categories)
	Beer_style	Varchar(255)	
Brewery	Brewery_no	int	Brewery Details
	Brewery_name	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



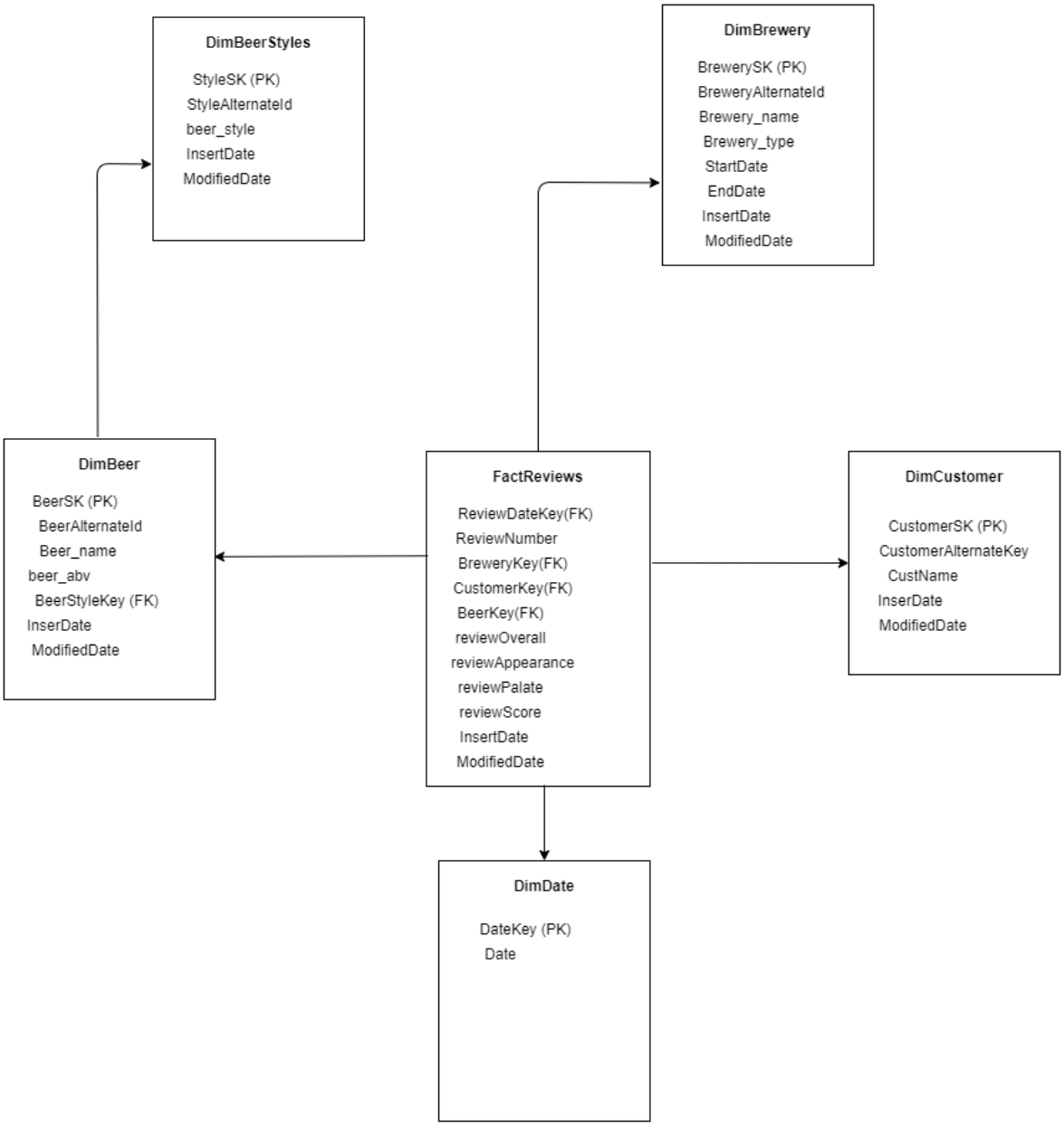
In the Staging layer Below Tables are created

- 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.

Data Warehouse Design & Development



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

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.

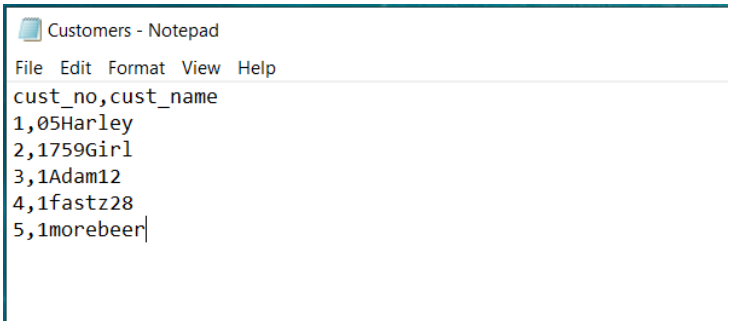
03	Check for the count when transforming data from source to staging tables
04	Check for the count when transforming data from staging to dimension tables
05	Check for duplicate values in the staging tables.
06	Check for duplicate values in the dimension tables.
07	Data length check for data in staging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

Test Data Set

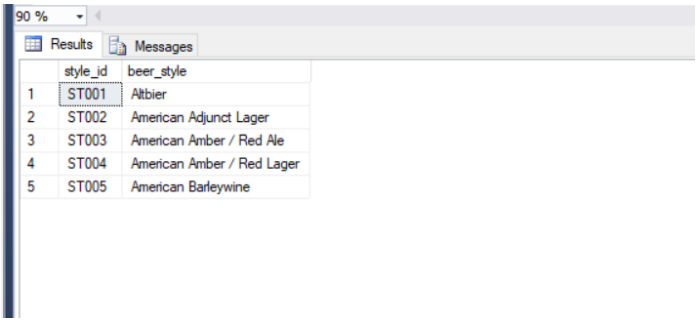
Before the development of the real data set and execution of the test cases, a small data set was derived from the Source and used for testing purposes to rectify issues in the process and to mitigate issues. The test data was loaded the same way as planned and tested in the below mentioned manner.

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

- Customer Dataset



- BeerStyles Dataset



	style_id	beer_style
1	ST001	Altbier
2	ST002	American Adjunct Lager
3	ST003	American Amber / Red Ale
4	ST004	American Amber / Red Lager
5	ST005	American Barleywine

- Brewery Dataset

90 %

Results | Messages

	brewery_no	brewery_name	brewery_type
1	1	Cambridge Brewing Company	Microbrewery
2	2	Cooperstown Brewing Company	Taproom Brewery
3	3	Amstel Brouwerij B. V.	Microbrewery
4	4	Pike Pub & Brewery	Taproom Brewery
5	5	Bluegrass Brewing Co. - East St. Matthew's	Taproom Brewery

- Review Dataset

90 %

Results | Messages

	reviewId	review_date	brewery_no	review_overall	review_appearance	review_palate	cust_no	beer_id
1	1	2012-09-29 00:00:00.000	105	4	4	5	2177	BEER0047695
2	2	2012-08-21 00:00:00.000	105	4.5	4	5	1533	BEER0047695
3	3	2012-01-08 00:00:00.000	105	5	4	5	1214	BEER0047695
4	4	2012-03-25 00:00:00.000	105	4	4.5	4	3473	BEER0047695
5	5	2012-05-19 00:00:00.000	105	3.5	4.5	3.5	50	BEER0033644

- Beer Dataset

90 %

Results | Messages

	beer_id	beer_name	beer_abv	style_id
1	BEER00175	Nine Man Ale Golden Ale	4.3	ST007
2	BEER00176	Benchwarmer Porter	6.3	ST0049
3	BEER00178	Strike Out Stout	4.6	ST0050
4	BEER00436	Amstel Light	3.5	ST0071
5	BEER00454	Bannatyne's Scotch Ale	9.2	ST0088

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

90 %

Results | Messages

	cust_no	cust_name
1	1	05Harley
2	2	1759Girl
3	3	1Adam12
4	4	1fastz28
5	5	1morebeer

Attachment 1.1

90 %

Results | Messages

	style_id	beer_style
1	ST001	Altbier
2	ST002	American Adjunct Lager
3	ST003	American Amber / Red Ale
4	ST004	American Amber / Red Lager
5	ST005	American Barleywine

Attachment 1.2

90 %

Results | Messages

	beer_id	beer_name	beer_abv	style_id
1	BEER00175	Nine Man Ale Golden Ale	4.3	ST007
2	BEER00176	Benchwarmer Porter	6.3	ST0049
3	BEER00178	Strike Out Stout	4.6	ST0050
4	BEER00436	Amstel Light	3.5	ST0071
5	BEER00454	Bannatyne's Scotch Ale	9.2	ST0088

Attachment 1.3

90 %

Results | Messages

	brewery_no	brewery_name	brewery_type
1	1	Cambridge Brewing Company	Microbrewery
2	2	Cooperstown Brewing Company	Taproom Brewery
3	3	Amstel Brouwerij B. V.	Microbrewery
4	4	Pike Pub & Brewery	Taproom Brewery
5	5	Bluegrass Brewing Co. - East St. Matthew's	Taproom Brewery

Attachment 1.4

90 %

Results | Messages

	reviewId	review_date	brewery_no	review_overall	review_appearance	review_palate	cust_no	beer_id
1	1	2012-09-29 00:00:00.000	105	4	4	5	2177	BEER0047695
2	2	2012-08-21 00:00:00.000	105	4.5	4	5	1533	BEER0047695
3	3	2012-01-08 00:00:00.000	105	5	4	5	1214	BEER0047695
4	4	2012-03-25 00:00:00.000	105	4	4.5	4	3473	BEER0047695
5	5	2012-05-19 00:00:00.000	105	3.5	4.5	3.5	50	BEER0033644

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

90 %

Results | Messages

	CustomerSK	CustomerAlternativeID	CustName	InsertDate	ModifiedDate
1	1	1	05Harley	2021-05-03 23:32:51.133	2021-05-13 01:05:59.097
2	2	2	1759Girl	2021-05-03 23:32:51.197	2021-05-13 01:05:59.100
3	3	3	1Adam12	2021-05-03 23:32:51.197	2021-05-13 01:05:59.100
4	4	4	1fastz28	2021-05-03 23:32:51.197	2021-05-13 01:05:59.100
5	5	5	1morebeer	2021-05-03 23:32:51.197	2021-05-13 01:05:59.103

Attachment 2.1

90 %

Results | Messages

	StyleSK	StyleAlternativeID	beer_style	InsertDate	ModifiedDate
1	1	ST001	Altbier	2021-05-02 00:19:27.857	2021-05-13 01:06:08.620
2	2	ST002	American Adjunct Lager	2021-05-02 00:19:27.873	2021-05-13 01:06:08.630
3	3	ST003	American Amber / Red Ale	2021-05-02 00:19:27.873	2021-05-13 01:06:08.630
4	4	ST004	American Amber / Red Lager	2021-05-02 00:19:27.873	2021-05-13 01:06:08.630
5	5	ST005	American Barleywine	2021-05-02 00:19:27.873	2021-05-13 01:06:08.630

Attachment 2.2

90 %

Results | Messages

	beerSK	beerAlternativeId	beer_name	beer_abv	beerStyleKey	InsertDate	ModifiedDate
1	1	BEER00175	Nine Man Ale Golden Ale	4.3	7	2021-05-02 01:21:29.887	2021-05-13 01:06:09.077
2	2	BEER00176	Benchwarmer Porter	6.3	49	2021-05-02 01:21:29.900	2021-05-13 01:06:09.080
3	3	BEER00178	Strike Out Stout	4.6	50	2021-05-02 01:21:29.900	2021-05-13 01:06:09.083
4	4	BEER00436	Amstel Light	3.5	71	2021-05-02 01:21:29.900	2021-05-13 01:06:09.083
5	5	BEER00454	Bannatyne's Scotch Ale	9.2	88	2021-05-02 01:21:29.917	2021-05-13 01:06:09.083

Attachment 2.3

90 %

Results | Messages

	brewerySK	breweryAlternativeId	brewery_name	brewery_type	StartDate	EndDate	InsertDate	ModifiedDate
1	1	1	Cambridge Brewing Company	Microbrewery	2021-05-03 19:05:47.000	NULL	2021-05-03 19:05:48.447	2021-05-03 19
2	2	2	Cooperstown Brewing Company	Taproom Brewery	2021-05-03 19:05:47.000	NULL	2021-05-03 19:05:48.447	2021-05-03 19
3	3	3	Amstel Brouwerij B. V.	Microbrewery	2021-05-03 19:05:47.000	NULL	2021-05-03 19:05:48.447	2021-05-03 19
4	4	4	Pike Pub & Brewery	Taproom Brewery	2021-05-03 19:05:47.000	NULL	2021-05-03 19:05:48.447	2021-05-03 19
5	5	5	Bluegrass Brewing Co. - East St. Matthew's	Taproom Brewery	2021-05-03 19:05:47.000	NULL	2021-05-03 19:05:48.447	2021-05-03 19

Attachment 2.4

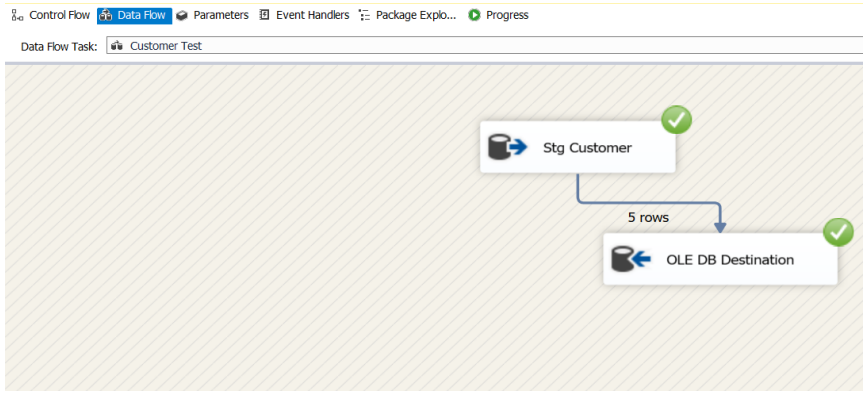
90 %

Results | Messages

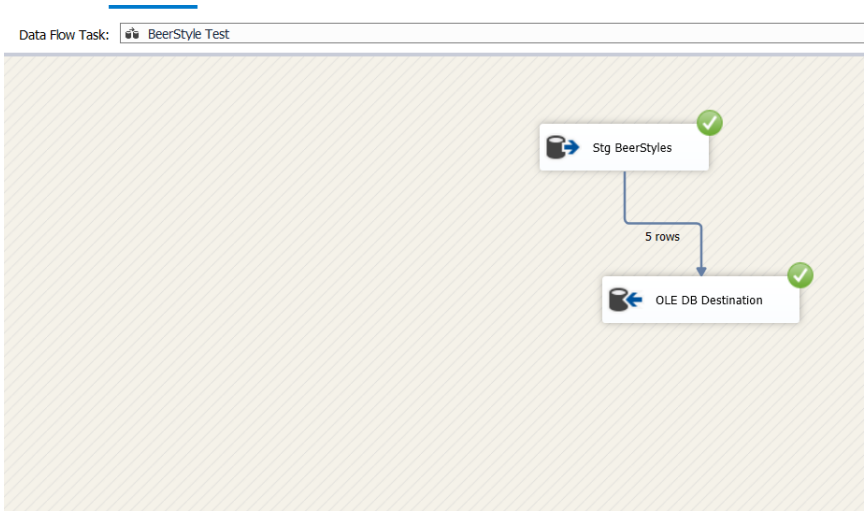
	ReviewDateKey	ReviewNumber	BreweryKey	CustomerKey	BeerKey	review_overall	review_appearance	review_palate	review_score	InsertDate
1	20120929	1	105	2177	977	4	4	5	4.33333333333333	2021-05-11 23:55:37.877
2	20120821	2	105	1533	977	4.5	4	5	4.5	2021-05-11 23:55:37.877
3	20120108	3	105	1214	977	5	4	5	4.66666666666667	2021-05-11 23:55:37.877
4	20120325	4	105	3473	977	4	4.5	4	4.16666666666667	2021-05-11 23:55:37.877
5	20120519	5	105	50	670	3.5	4.5	3.5	3.83333333333333	2021-05-11 23:55:37.877

Attachment 2.5

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



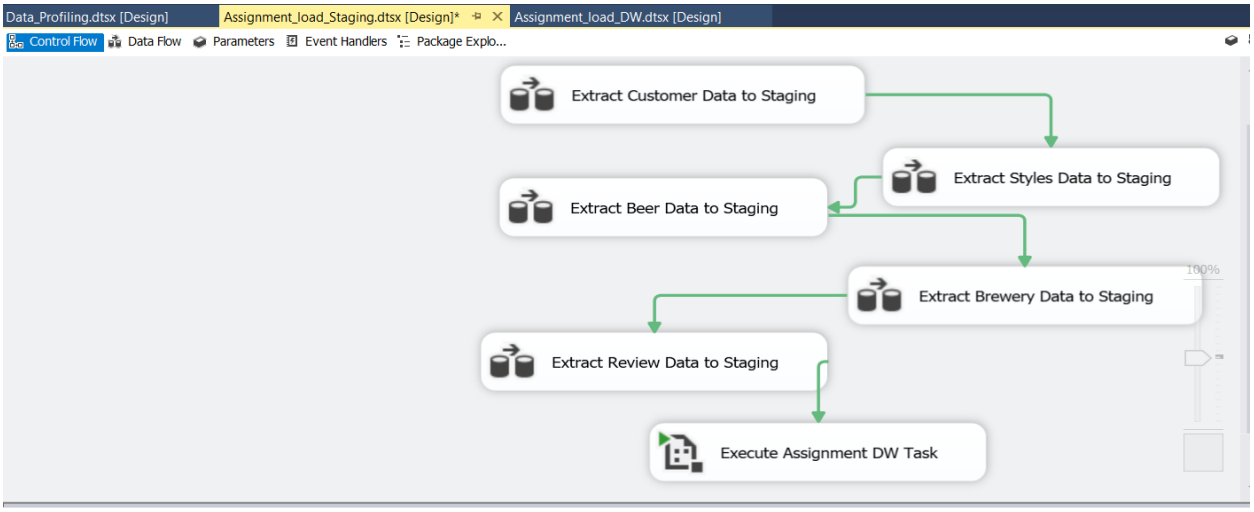
5 test data rows in Customer inserted Successfully



5 test data rows in Beer styles inserted Successfully

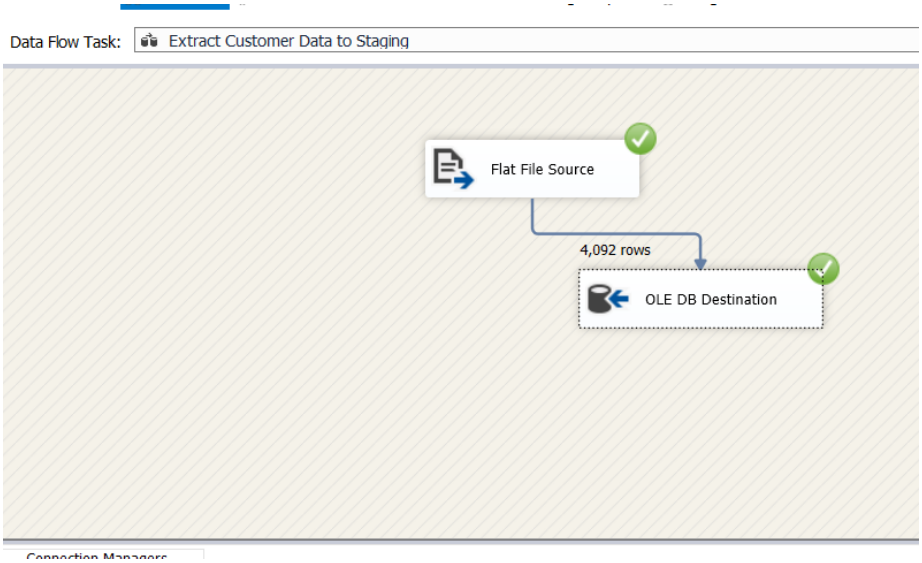
ETL Development

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:



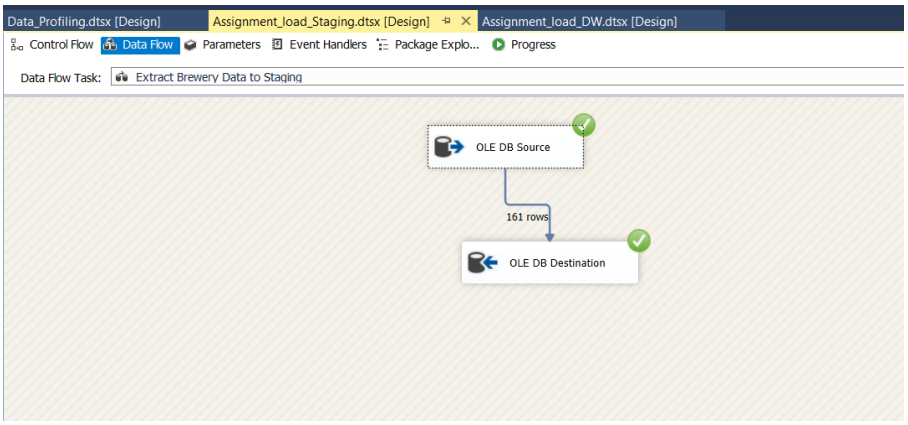
Screenshots of all the data sources that were staged and truncate tables created are attached below:

Staging customer details



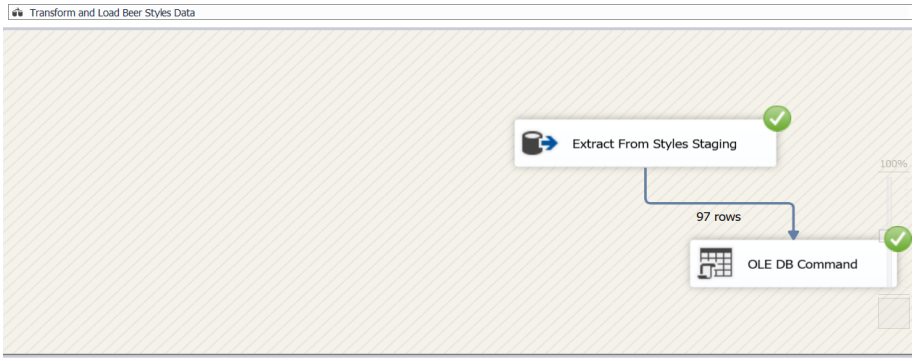
Customer Details - Data is extracted from Customer text file and insterted into customer staging Table

Staging Brewery Data



Brewery Details - Data is extracted from Brewery table in source DB and insterted into Brewery staging Table

Staging Beer Styles Data

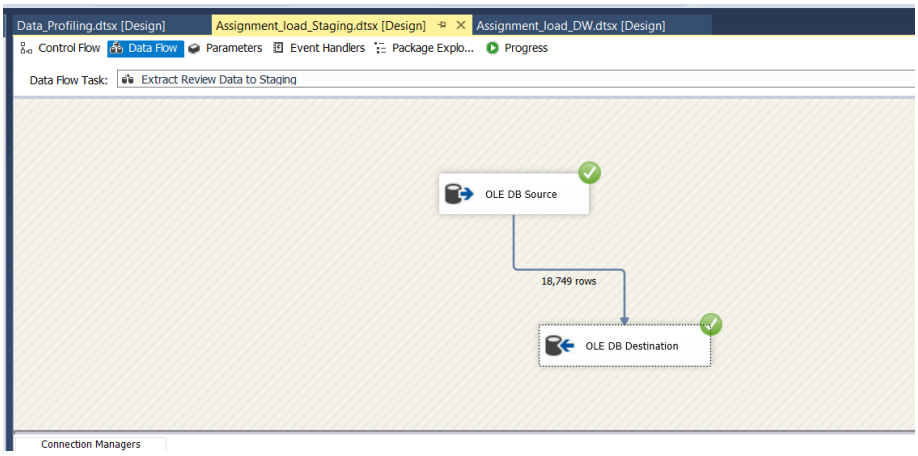


Beer Styles Details - Data is extracted from Beer Styles table in source DB and insterted into Beer Styles staging Table

Staging Beer Data

Beer Details - Data is extracted from Beer table in source DB and insterted into Beer staging Table

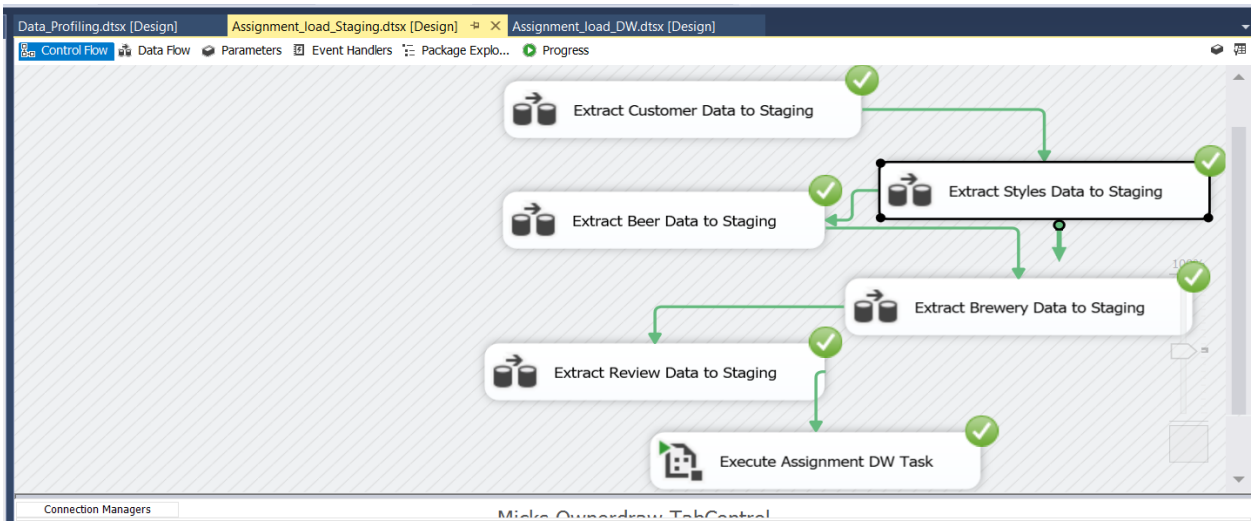
Staging Review Data



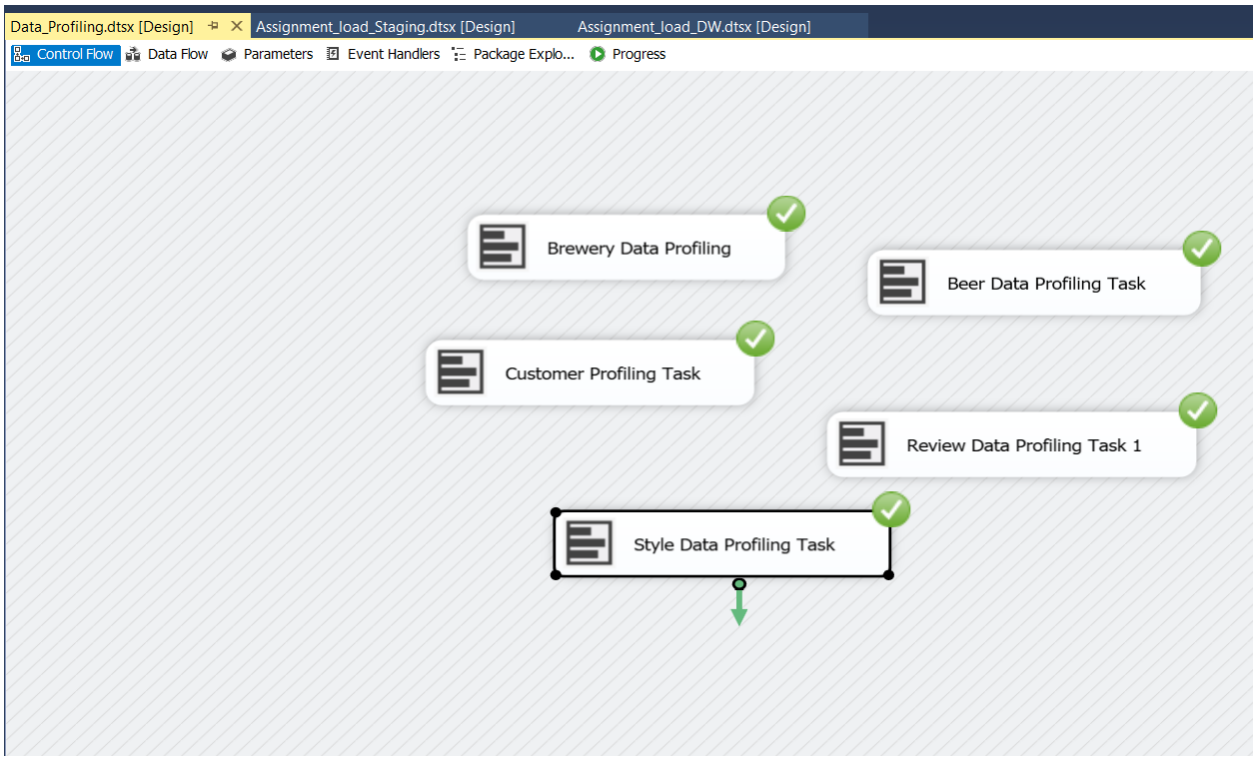
Review Details - Data is extracted from Review table in source DB and insterted into Review staging Table

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

- After following the above steps and executing:

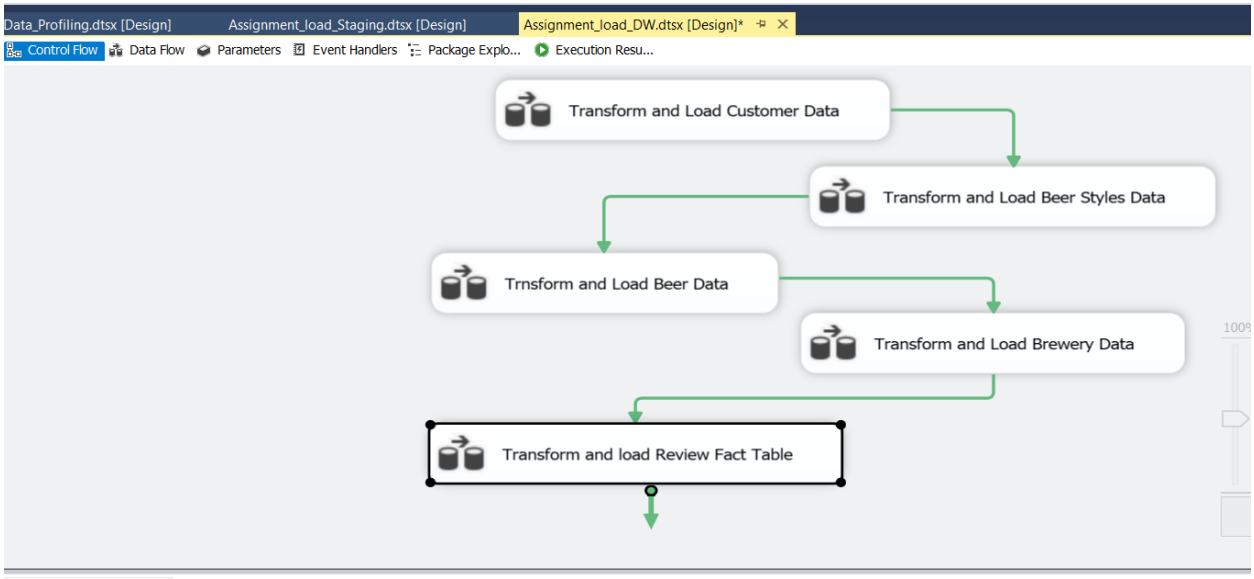


Next step is data profiling, and it is done as shown below



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, Brewery details were considered as slowly changing details.

Brewery type column was set as changing attribute.

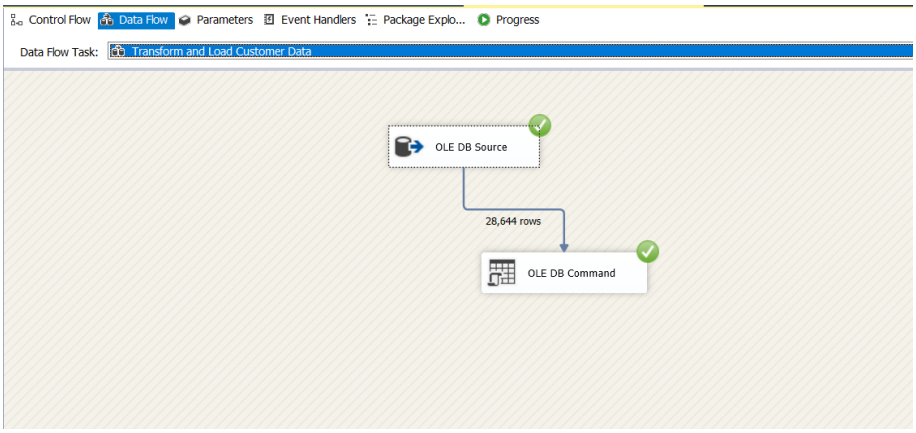
After extracting data from the Brewery staging table, it was identified as a slowly changing dimension, it was connected as shown below and loaded data to the Brewery dimension table.

1)Extracted from StgBrewery Table

2)Make the dimension a slowly changing dimension

3)Loaded to Brewery Dimension

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



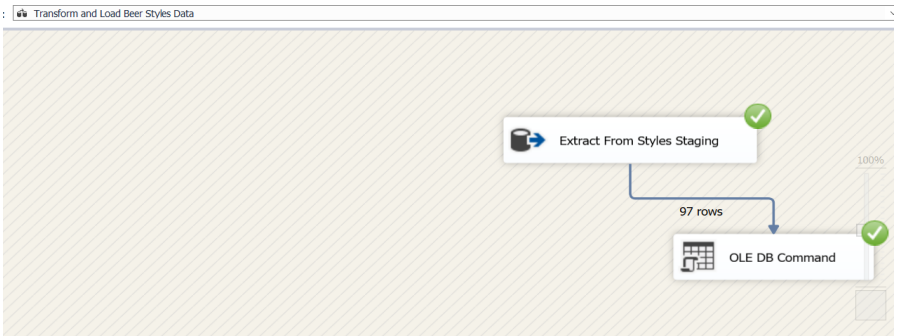
1)Extracted from Customer Table

2)Load into 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



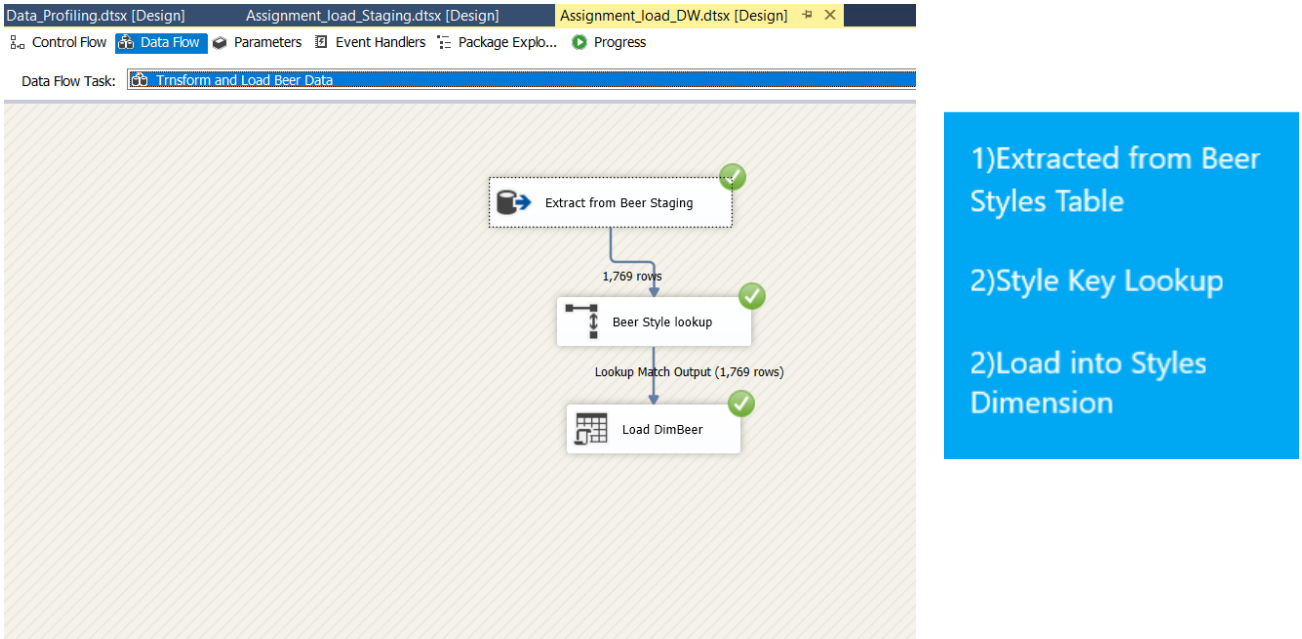
1)Extracted from Beer Styles Table

2)Load into Styles 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



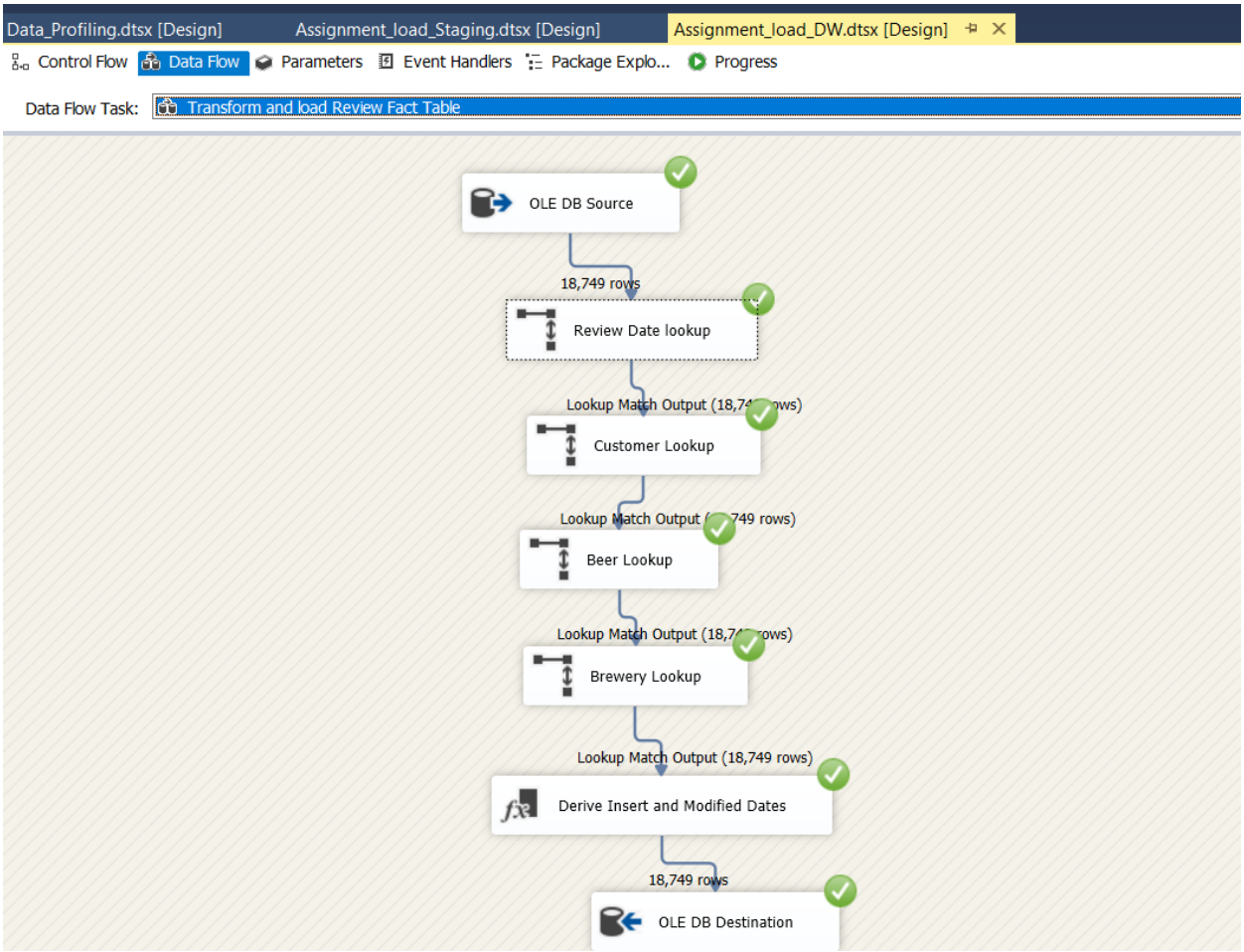
The update procedure used to update Beer details:

```
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.



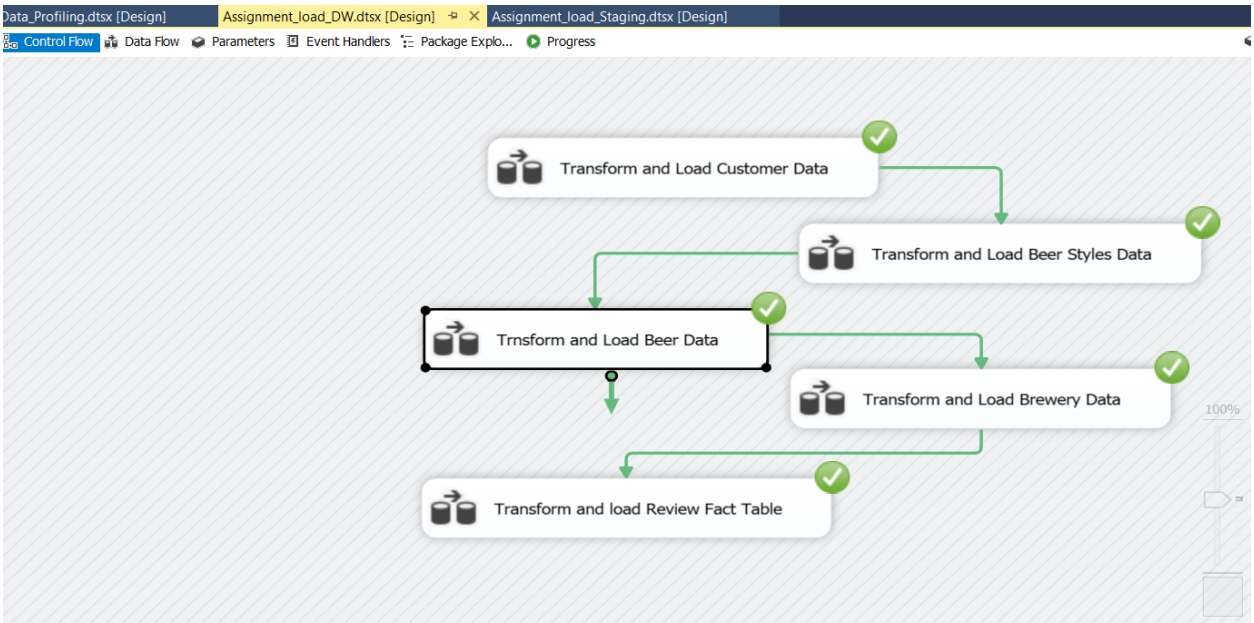
The query used to create the date dimension is mentioned below:

```
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
```

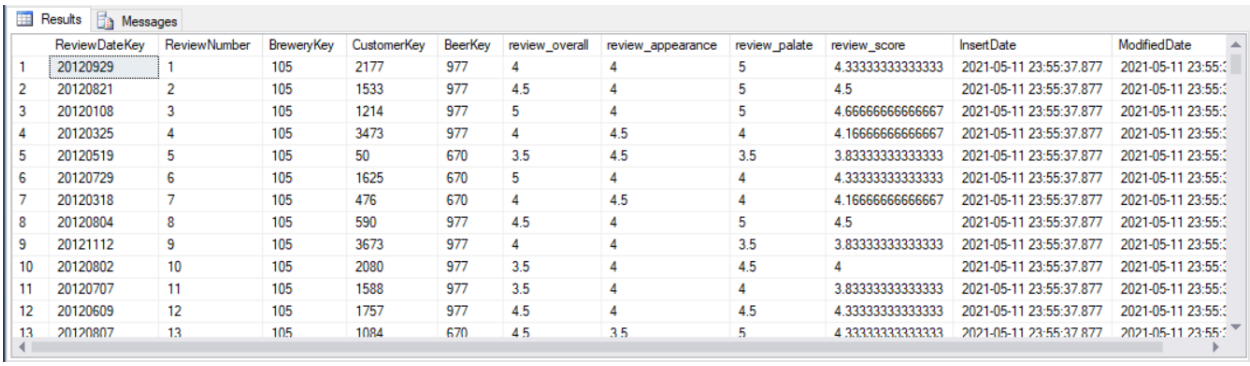
```
        END

    /* 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:



Print screen of the fact table:



Results | Messages

	ReviewDateKey	ReviewNumber	BreweryKey	CustomerKey	BeerKey	review_overall	review_appearance	review_palate	review_score	InsertDate	ModifiedDate
1	20120929	1	105	2177	977	4	4	5	4.33333333333333	2021-05-11 23:55:37.877	2021-05-11 23:55:
2	20120821	2	105	1533	977	4.5	4	5	4.5	2021-05-11 23:55:37.877	2021-05-11 23:55:
3	20120108	3	105	1214	977	5	4	5	4.66666666666667	2021-05-11 23:55:37.877	2021-05-11 23:55:
4	20120325	4	105	3473	977	4	4.5	4	4.16666666666667	2021-05-11 23:55:37.877	2021-05-11 23:55:
5	20120519	5	105	50	670	3.5	4.5	3.5	3.83333333333333	2021-05-11 23:55:37.877	2021-05-11 23:55:
6	20120729	6	105	1625	670	5	4	4	4.33333333333333	2021-05-11 23:55:37.877	2021-05-11 23:55:
7	20120318	7	105	476	670	4	4.5	4	4.16666666666667	2021-05-11 23:55:37.877	2021-05-11 23:55:
8	20120804	8	105	590	977	4.5	4	5	4.5	2021-05-11 23:55:37.877	2021-05-11 23:55:
9	20121112	9	105	3673	977	4	4	3.5	3.83333333333333	2021-05-11 23:55:37.877	2021-05-11 23:55:
10	20120802	10	105	2080	977	3.5	4	4.5	4	2021-05-11 23:55:37.877	2021-05-11 23:55:
11	20120707	11	105	1588	977	3.5	4	4	3.83333333333333	2021-05-11 23:55:37.877	2021-05-11 23:55:
12	20120609	12	105	1757	977	4.5	4	4.5	4.33333333333333	2021-05-11 23:55:37.877	2021-05-11 23:55:
13	20120807	13	105	1084	670	4.5	3.5	5	4.33333333333333	2021-05-11 23:55:37.877	2021-05-11 23:55:

• The column review Score is calculated the following way

Review_Score = (review_overall+review_appearance+review_palate)/3

Execution of Test Cases and TSR

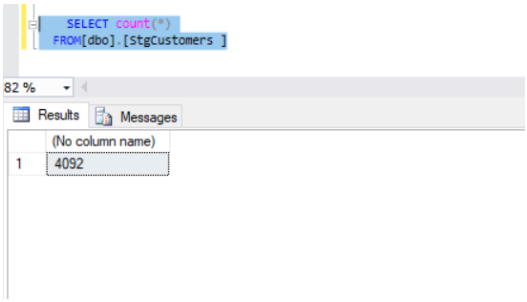
After testing using test data and passing all the test cases, data set was loaded (As explained earlier). The loaded data was tested using SQL queries as the data set is large and testing row by row is a tiresome compared to testing test data.

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.

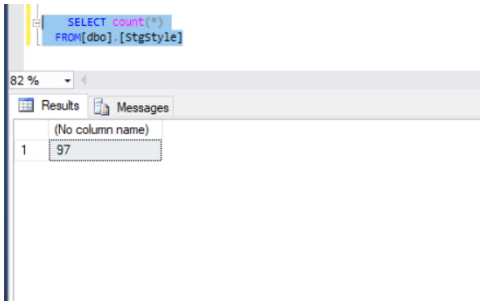
Execution of test cases

Test Scenario ID	3					
Test Case Description	Check for the count when transforming data from source to staging tables					
Pre-Requisite	Data loaded from 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

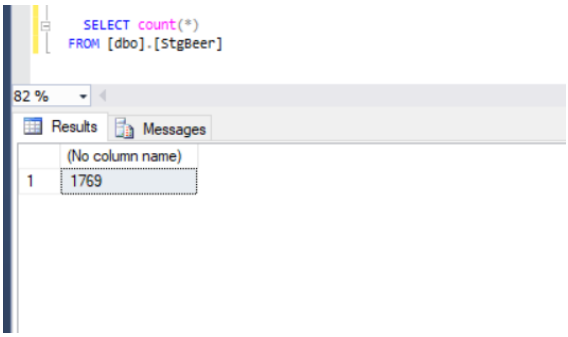
5	Check for the count when transforming data from Review source to Review Staging	select count(*) from StgReview;	18749	18749	Pass	Refer 3.5 attachment



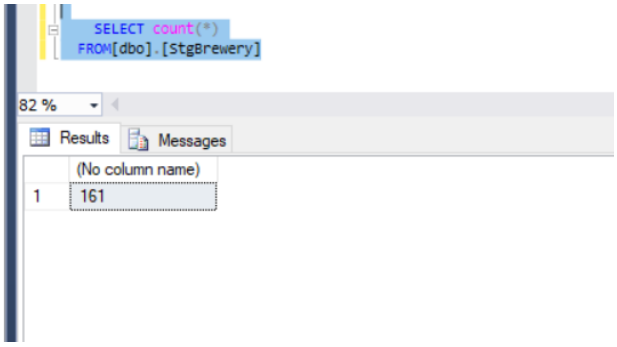
Attachment 3.1



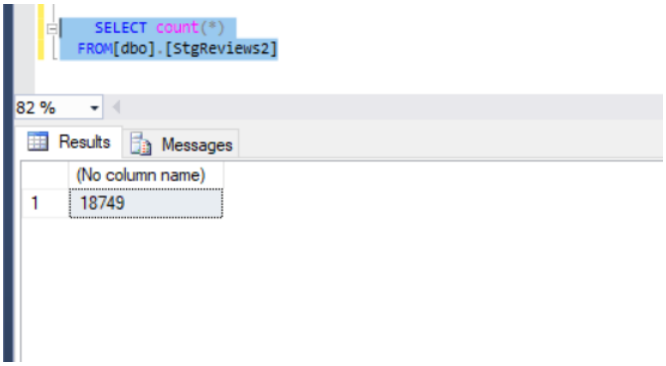
Attachment 3.2



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

```
SELECT count(*)
FROM[dbo].[DimCustomer]
```

82 %

Results | Messages

	(No column name)
1	4092

Attachment 4.1

```
SELECT count(*)
FROM[dbo].[DimBeerStyle]
```

82 %

Results | Messages

	(No column name)
1	97

Attachment 4.2

```
SELECT count(*)
FROM[dbo].[DimBeer]
```

82 %

Results | Messages

	(No column name)
1	1769

Attachment 4.3

```
SELECT count(*)
FROM[dbo].[DimBrewery]
```

82 %

Results | Messages

	(No column name)
1	161

Attachment 4.4

```
SELECT count(*)
FROM[dbo].[FactReviews]
```

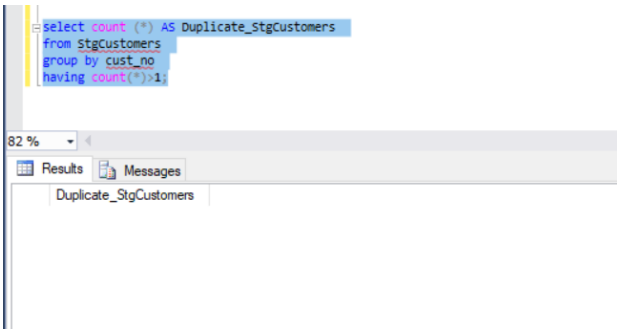
2 %

Results | Messages

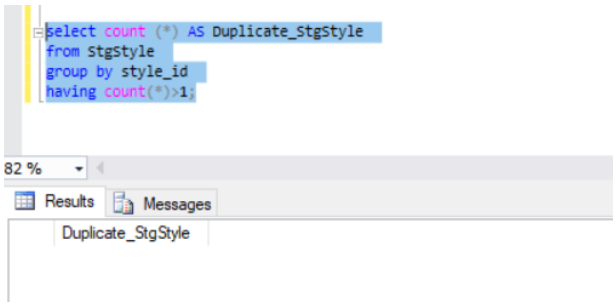
	(No column name)
1	18749

Attachment 4.5

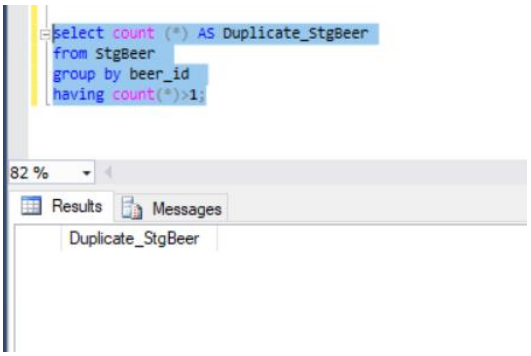
Test Scenario ID	5					
Test Case Description	Check for duplicate values in the staging tables.					
Pre-Requisite	Data loaded from source to staging tables in SQL tool					
SNO	**Action**	**Sql Query**	**Expected Output**	**Actual Output**	**Test Result**	**Test Comments**
1	Check whether data has got 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



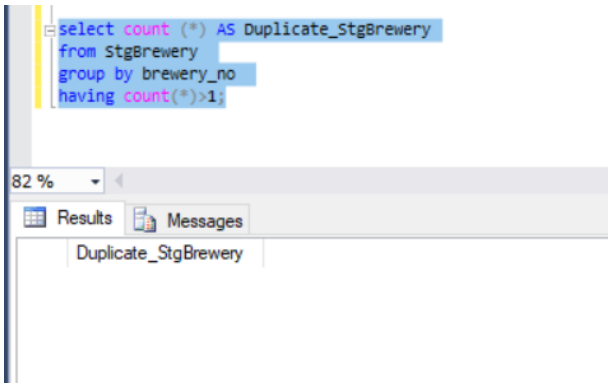
Attachment 5.1



Attachment 5.2

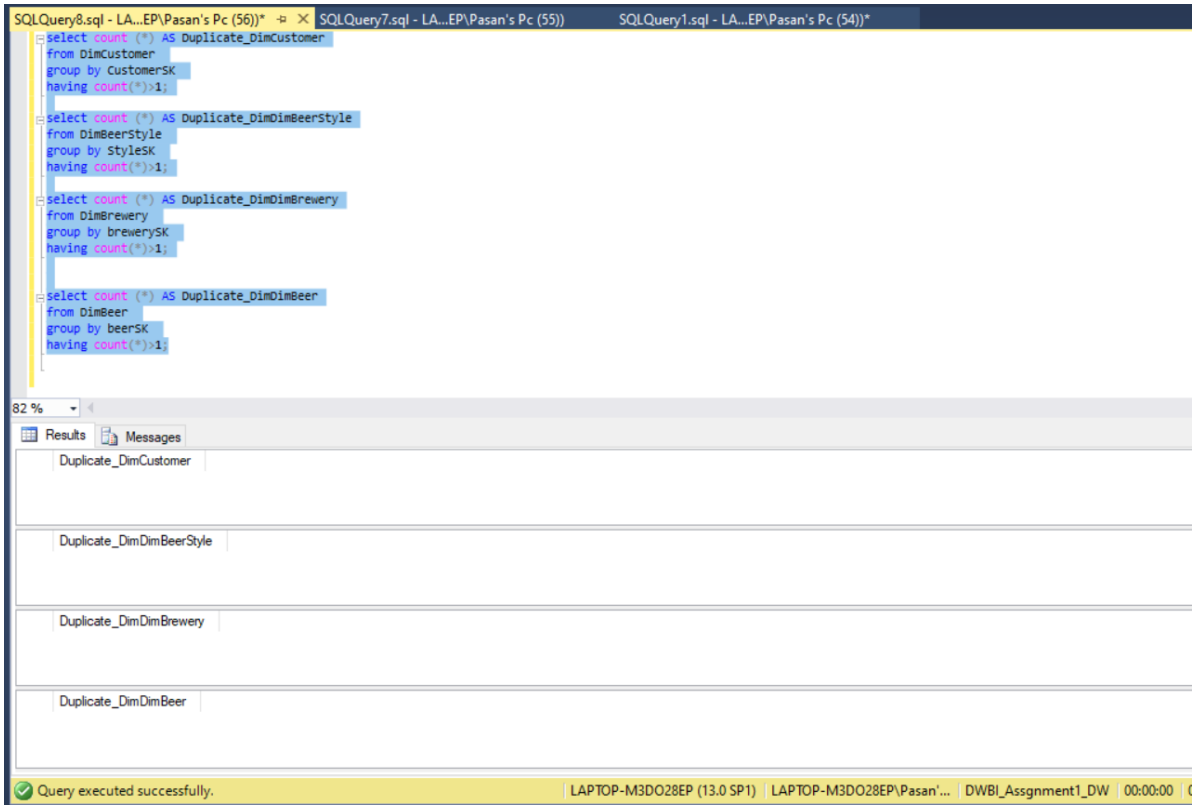


Attachment 5.3



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

```
select DATALENGTH(style_id)as style_id ,DATALENGTH(beer_style) as beer_style
from [dbo].[StgStyle]
where style_id='ST0020';
```

82 %

Results | Messages

	style_id	beer_style
1	6	19

```
select DATALENGTH(style_id)as style_id ,
DATALENGTH(beer_style) as beer_style
from [dbo].[StylesTable] where style_id='ST0020';
```

81 %

Results | Messages

	style_id	beer_style
1	6	19

Attachment 7.1

```
select DATALENGTH(beer_id)as cust_no ,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';
```

82 %

Results | Messages

	cust_no	beer_name	beer_abv	style_id
1	9	46	8	5

```
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].BeerTable where beer_id='BEER00175';
```

81 %

Results | Messages

	beer_id	beer_name	beer_abv	style_id
1	9	46	8	5

Attachment 7.2

```
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;
```

81 %

Results | Messages

	brewery_no	brewery_name	brewery_type
1	4	24	32

```
select DATALENGTH(brewery_no)as brewery_no ,DATALENGTH(brewery_name) as brewery_name,
DATALENGTH(brewery_type) as brewery_type
from [dbo].BreweryTable where brewery_no=10;
```

%

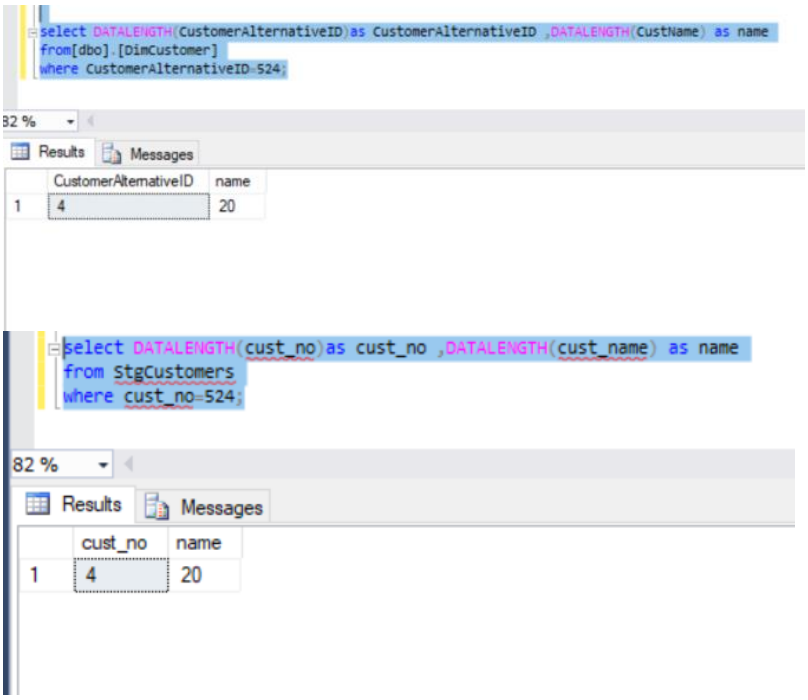
Results | Messages

	brewery_no	brewery_name	brewery_type
1	4	24	32

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	select DATALENGTH(breweryAlternativeId)as breweryAlternativeId ,DATALENGTH(brewery_type) as brewery_type from	Brewery_no=4 Brewery_type=32	Brewery_no=4 Brewery_type=32	Pass	Refer 8.4 attachment

	Dimension table are the same	[dbo].[DimBrewery] where breweryAlternativeId=10) as beer_abv,DATALENGTH(style_id) as style_id from [dbo].[StgBeer] where beer_id='BEER00175';				



Attachment 8.1

```
select DATALENGTH(style_id)as style_id ,DATALENGTH(beer_style) as beer_style
from [dbo].[StgStyle]
where style_id='ST0020';
```

82 %

Results | Messages

	style_id	beer_style
1	6	19

```
select DATALENGTH(StyleAlternativeID)as StyleAlternativeID ,DATALENGTH(beer_style) as beer_style
from [dbo].[DimBeerStyle]
where StyleAlternativeID='ST0020';
```

2 %

Results | Messages

	StyleAlternativeID	beer_style
1	6	19

Attachment 8.2

```
select DATALENGTH(beerAlternativeId)as beerAlternativeId ,DATALENGTH(beer_name) as beer_name,
    DATALENGTH(beer_abv) as beer_abv
from [dbo].[DimBeer]
where beerAlternativeId='BEER00175';
```

82 %

Results | Messages

	beerAlternativeId	beer_name	beer_abv
1	9	46	8

```
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';
```

%

Results | Messages

beer_id	beer_name	beer_abv	style_id
9	46	8	5

Attachment 8.3

```
select DATALENGTH(brewery_no)as brewery_no ,DATALENGTH(brewery_type) as brewery_type
from [dbo].[stgBrewery]
where brewery_no=10;
```

82 %

Results | Messages

	brewery_no	brewery_type
1	4	32

```
select DATALENGTH(breweryAlternativeId)as breweryAlternativeId ,DATALENGTH(brewery_type) as brewery_type
from [dbo].[DimBrewery]
where breweryAlternativeId=10;
```

82 %

Results | Messages

	breweryAlternativeId	brewery_type
1	4	32

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

```
use DWBI_Assignment1_Staging
SELECT COLUMN_NAME,DATA_TYPE
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'StgCustomers'

use DWBI_Assgnment1_DW
SELECT COLUMN_NAME,DATA_TYPE
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'DimCustomer'
```

81 %

Results | Messages

	COLUMN_NAME	DATA_TYPE
1	cust_no	int
2	cust_name	nvarchar

	COLUMN_NAME	DATA_TYPE
1	CustomerSK	int
2	CustomerAlternativeID	int
3	CustName	nvarchar
4	InsertDate	datetime
5	ModifiedDate	datetime

Attachment 9.1

```
use DWBI_Assignment1_Staging
SELECT COLUMN_NAME,DATA_TYPE
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'StgStyle'

use DWBI_Assgnment1_DW
SELECT COLUMN_NAME,DATA_TYPE
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'DimBeerStyle'
```

81 %

Results | Messages

	COLUMN_NAME	DATA_TYPE
1	style_id	varchar
2	beer_style	varchar

	COLUMN_NAME	DATA_TYPE
1	StyleSK	int
2	StyleAlternativeID	varchar
3	beer_style	varchar
4	InsertDate	datetime
5	ModifiedDate	datetime

Attachment 9.2

```
use DWBI_Assignment1_Staging
SELECT COLUMN_NAME,DATA_TYPE
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'StgBeer'

use DWBI_Assgnment1_DW
SELECT COLUMN_NAME,DATA_TYPE
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'DimBeer'
```

Results | Messages

	COLUMN_NAME	DATA_TYPE
1	beer_id	varchar
2	beer_name	nvarchar
3	beer_abv	float
4	style_id	varchar

	COLUMN_NAME	DATA_TYPE
1	beerSK	int
2	beerAlternativeId	varchar
3	beer_name	nvarchar
4	beer_abv	float
5	beerStyleKey	int
6	InsertDate	datetime
7	ModifiedDate	datetime

Attachment 9.3

```
use DWBI_Assignment1_Staging
SELECT COLUMN_NAME,DATA_TYPE
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'StgBrewery'

use DWBI_Assgnment1_DW
SELECT COLUMN_NAME,DATA_TYPE
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'DimBrewery'
```

Results | Messages

	COLUMN_NAME	DATA_TYPE
1	brewery_no	int
2	brewery_name	nvarchar
3	brewery_type	nvarchar

	COLUMN_NAME	DATA_TYPE
1	brewerySK	int
2	breweryAlternativeId	int
3	brewery_name	nvarchar
4	brewery_type	nvarchar
5	StartDate	datetime
6	EndDate	datetime
7	InsertDate	datetime
8	ModifiedDate	datetime

Attachment 9.4