IT3021 – Data Warehousing and Business Intelligence

Assignment 1

Submitted by:

Ranasinghe R.A.B.N

IT20076016

Contents

Data set Selection

The data set selected is an OLTP dataset. The link to the dataset is given below:

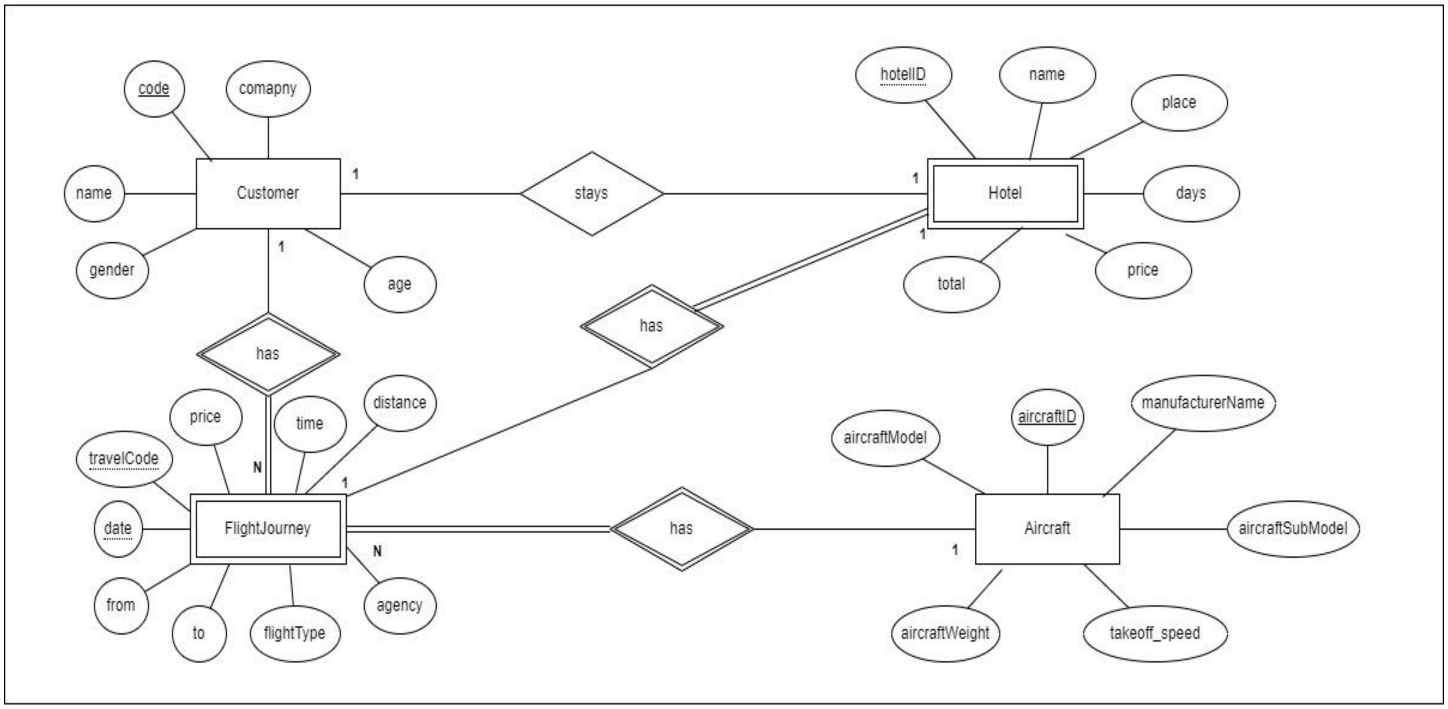
https://www.kaggle.com/datasets/leomauro/argodatathon2019

This data set shows customer details for three years who flew to different places using different travel agencies. The travel agency provides bookings of planes and hotels for customers in the journey. The data were modified according to the needs of this assignment like adding new columns and auto-generating files.

The table below provides a description of the data set :

Table Name	Column Name	Data type	Description
Customer	code	nvarchar(50)	Unique code for a customer
	name	nvarchar(50)	Name of the customer
	gender	nvarchar(10)	Gender of the customer
	company	nvarchar()	Travelling agency name
	age	int	Age of the customer
Aircraft	aircraftID	int	Unique id for an aircraft
	manufacturerName	nvarchar(50)	Name of the aircraft manufacturer
	aircraftModel	nvarchar(20)	Model of the aircraft
	aircraftSubModel	nvarchar(50)	Sub-model of the aircraft
	aircraftWeight	int	Weight of the aircraft
	takeoff_speed	int	Takeoff speed of the aircraft
FlightJourney	userCode	nvarchar(50)	Unique code for a customer
	travelCode	nvarchar(50)	Unique travel code for each customer
	date	date	Date when the journey started.
	from	nvarchar(50)	Where the customer came from
	to	nvarchar(50)	Where the customer flies to
	flightType	nvarchar(50)	Type of the flight
	agency	nvarchar(50)	Agency used by customers for the journey
	distance	float	Distance of the flight
	time	float	Time taken by the flight
	price	float	Cost of the flight
	aircraftID	int	Unique id for an aircraft
Hotel	hotelID	int	Unique id for a hotel
	travelCode	nvarchar(50)	Unique travel code for each customer
	date	date	Date
	userCode	nvarchar(50)	Unique code for a customer
	name	nvarchar(50)	Name of the hotel
	place	nvarchar(50)	Location of the hotel
	days	int	Number of days the customer is staying at the hotel.
	price	float	Cost of the hotel per day
	total	float	Total cost for all days (price * days)

ER Diagram



The ER diagram above represents the relationships between the customer, hotel, flightJourney and the aircraft entities.

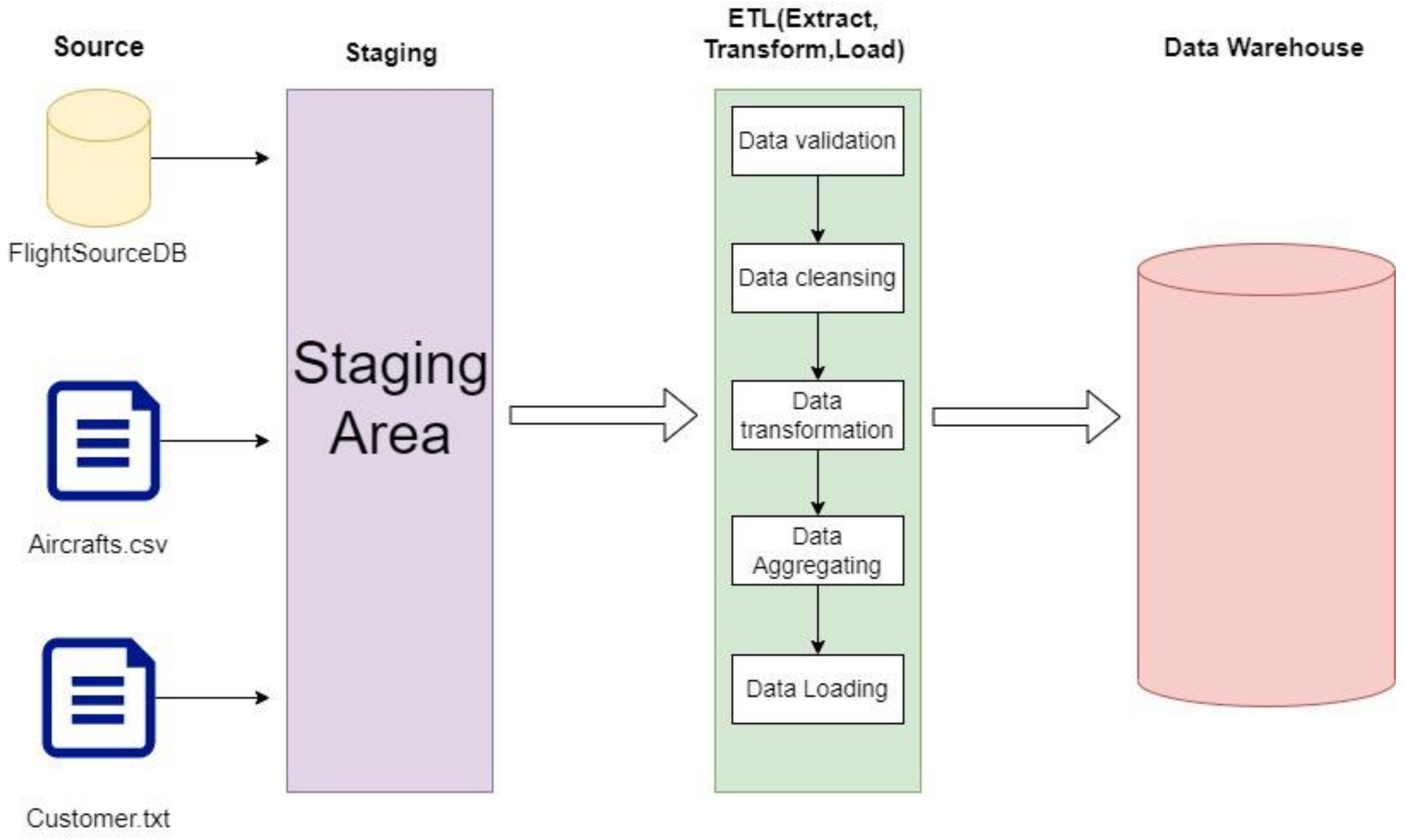
Prepration of data sources

Selection of files

The data set only had csv files so I divided them into two main types. There was a total of four csv files. The main file types are as follows :

- The csv files were imported into the Flights_SourceDB and they were converted into tables. The Flights_SourceDB has two tables; FlightJourney and Hotel. The Flights_SourceDB is exported into the DataSources folder with the name “Flight_SourceDB.bacpac”.
- From the remaining two csv files, one csv file is being modified into a text file. The text file contains customer details and it has the name “Customers.txt”.
- The remaining csv file was kept unchanged so the aircrafts details are stored in the csv file known as “Aircrafts.csv”.

Solution Architecture



The first step in creating the data warehouse is to load the data from the data sources (FlightSourceDB, Aircrafts.csv, and Customer.txt) to the staging area. To store the details in the staging area, tables should be created. Following are the names of the tables created:

Hotel Staging

Flight Journey Staging

Customers Staging

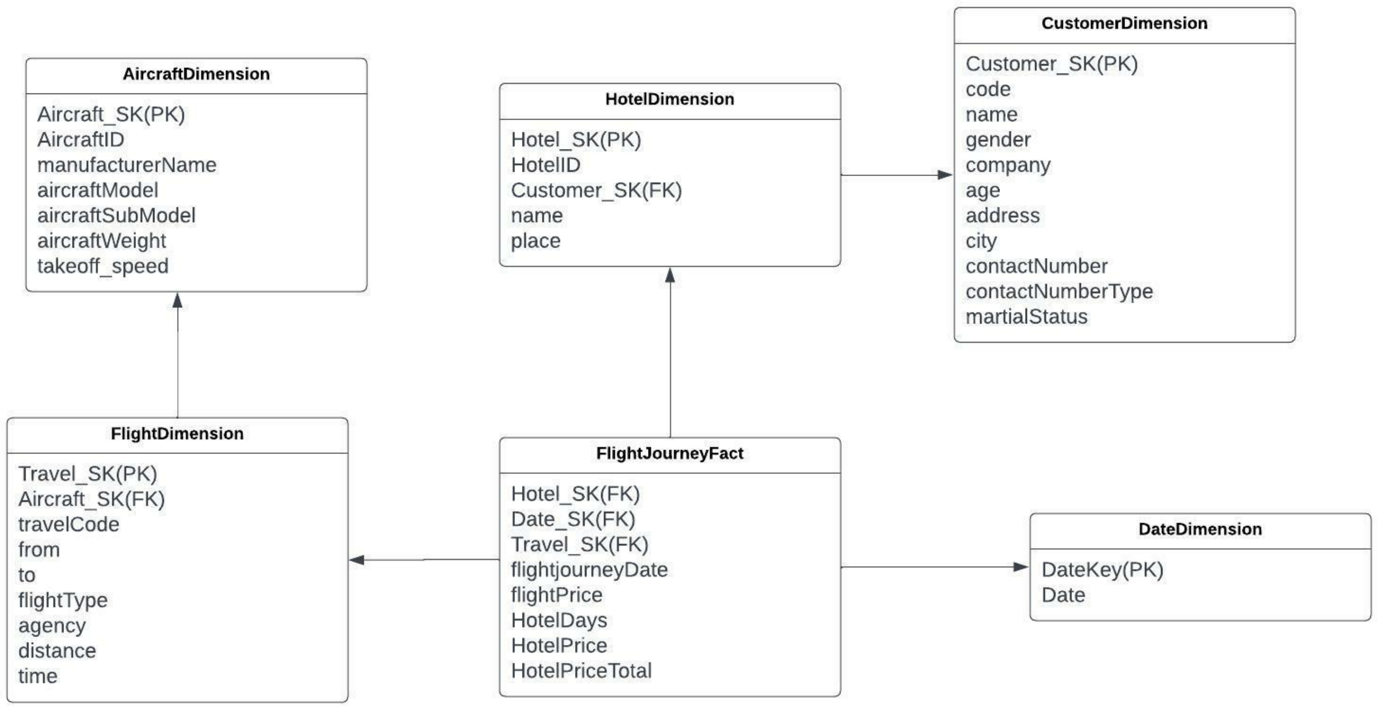
Aircrafts Staging

After the staging step the data tables are profiled and aggregations are performed. After the completion of these steps the data is validated using the ETL's and the data warehouse is created.

After the data warehouse is created the end-users (such as data engineers) can use this to generate Business Intelligence (BI) reports, data mining and for data visualization.

Data warehouse design and development

Snowflake schema



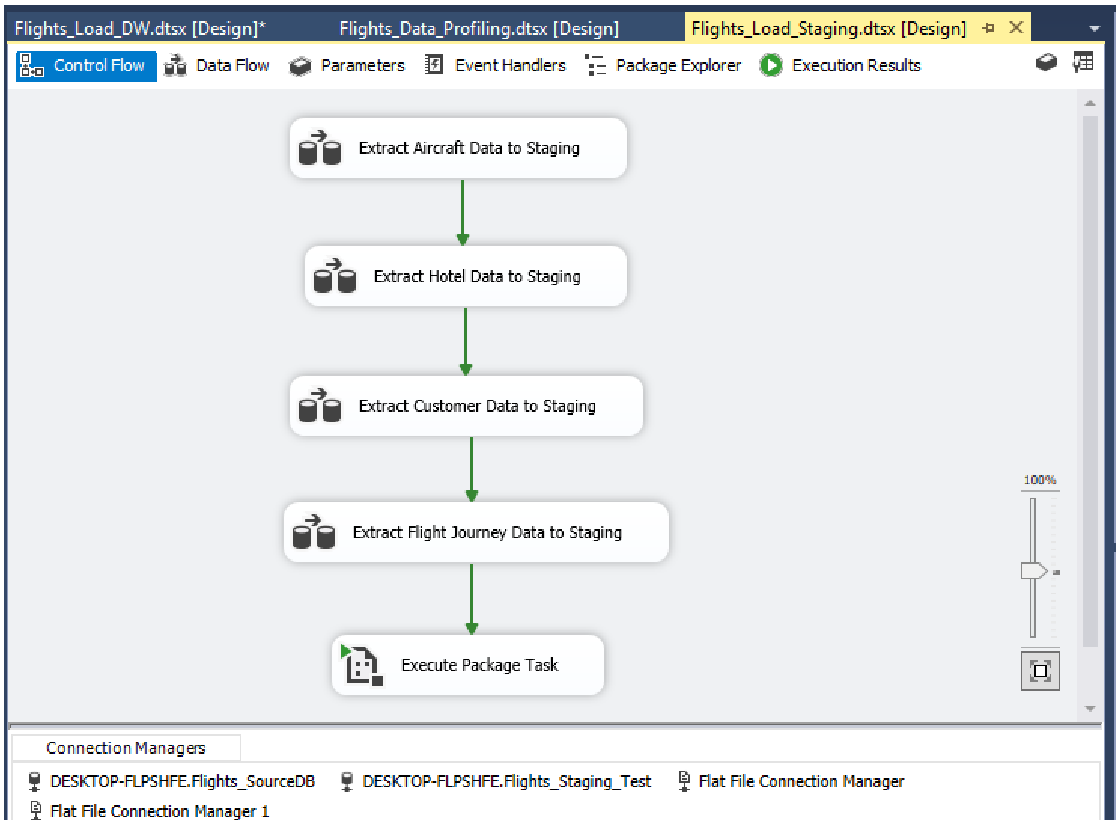
To design the data warehouse, the snowflake schema dimensional model is used. The snowflake schema consists of one fact table and five dimension tables. The flight journey for a single customer is identified as the lowest possible grain.

Assumptions : The customer dimension is considered as a slowly changing dimension since we need to have both the new address and the previous address.

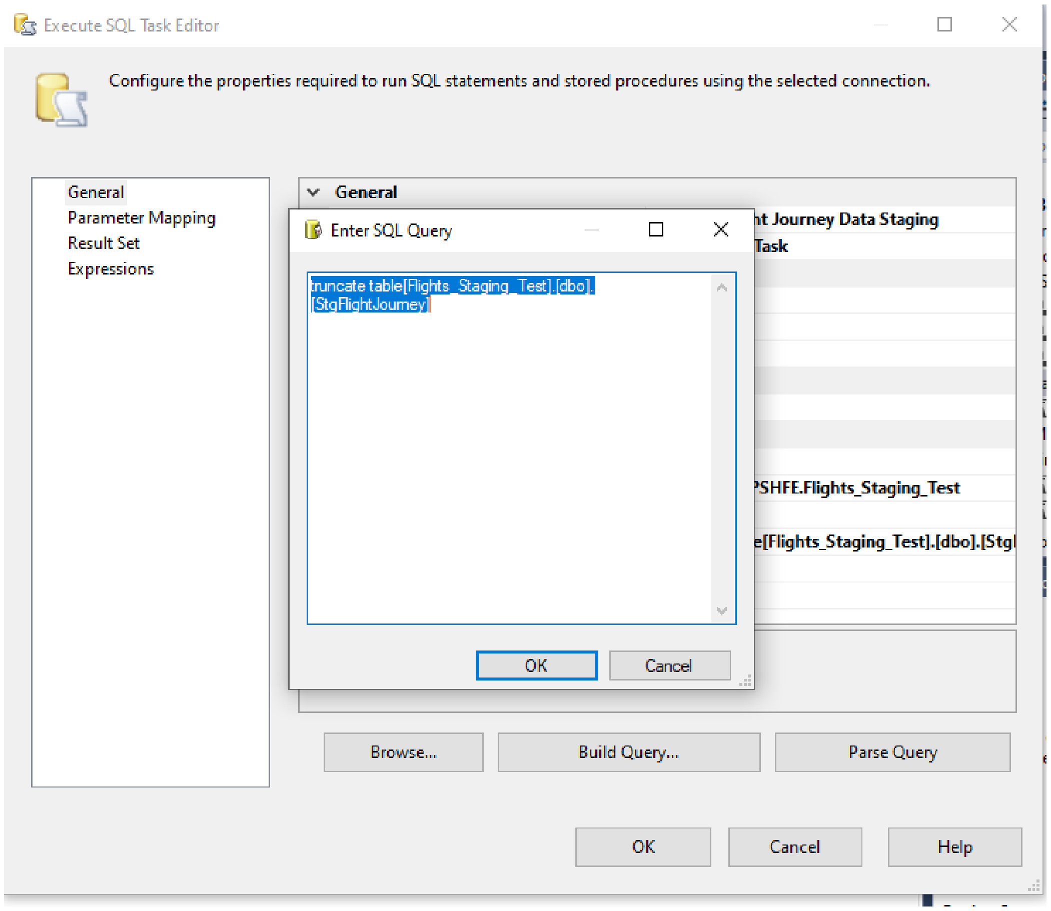
ETL Development

Extract

As the first step of the ETL development data is being extracted from the data sources (DB Source, text file, and csv file). A data flow task is being used to represent the data extracted and the data that was extracted is loaded from DB Source to the staging database (contains the staging tables). There are no staging tables in the staging database currently but the staging tables will be created through the SSIS so for each table in the source, a separate staging table is created in the staging database. The data flow tasks will be executed in the order as shown below.


Flights_Load_DW.dtsx [Design]*
Flights_Data_Profiling.dtsx [Design]
Flights_Load_Staging.dtsx [Design]
Control Flow
Data Flow
Parameters
Event Handlers
Package Explorer
Execution Results
Extract Aircraft Data to Staging
Extract Hotel Data to Staging
Extract Customer Data to Staging
Extract Flight Journey Data to Staging
Execute Package Task
100%
Connection Managers
DESKTOP-FLPSHFE.Flights_SourceDB
DESKTOP-FLPSHFE.Flights_Staging_Test
Flat File Connection Manager
Flat File Connection Manager 1

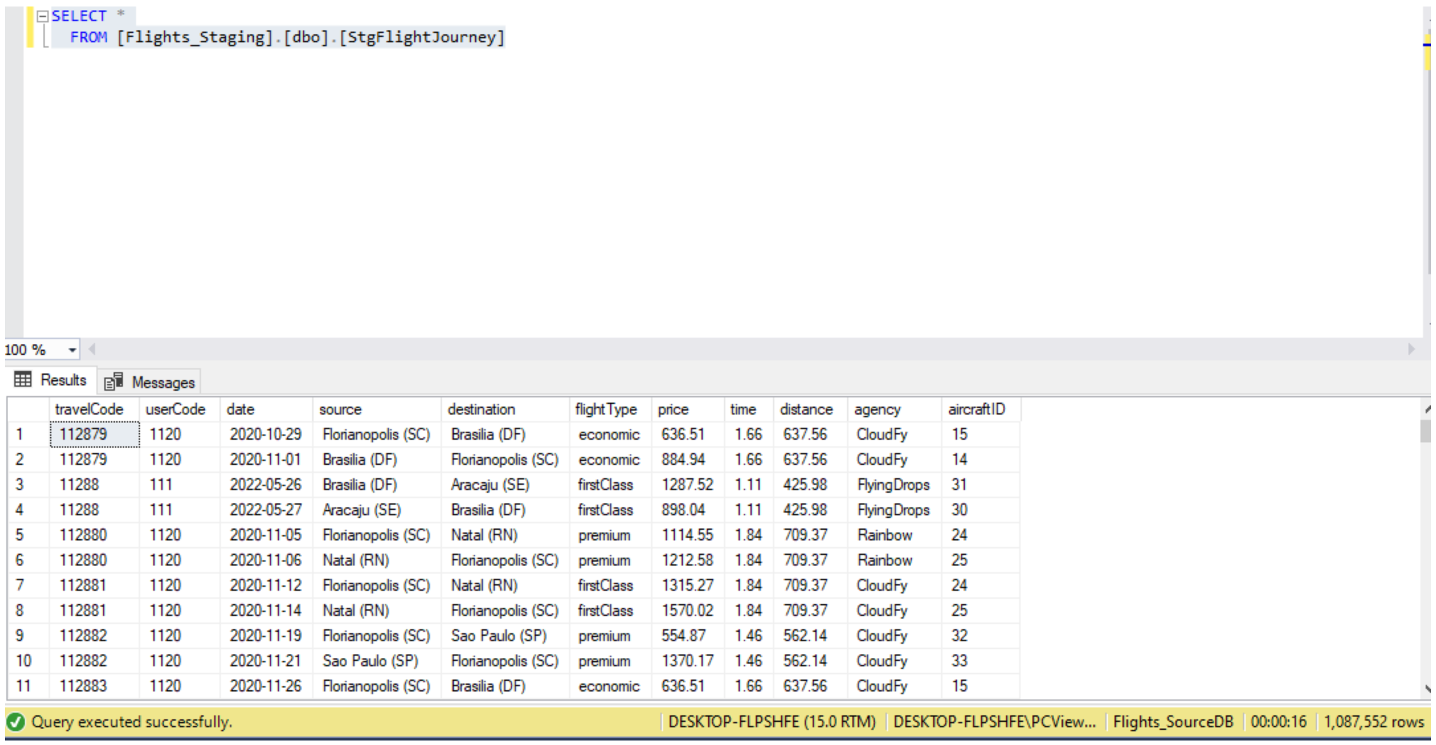
When we run the process for multiple times the staging tables will be repeatedly loaded with data so now the table has duplicate data as well so to prevent this, we will truncate the data available in the staging database tables. The screenshot below shows how we can achieve this.



```
truncate table[Flights_Staging_Test].[dbo].
[StgFlightJourney]
```

ht Journey Data Staging
Task
PSHFE.Flights_Staging_Test
e[Flights_Staging_Test].[dbo].[Stg

The screenshot below shows the StgFlightJourney table before truncating with lots of duplicate data.

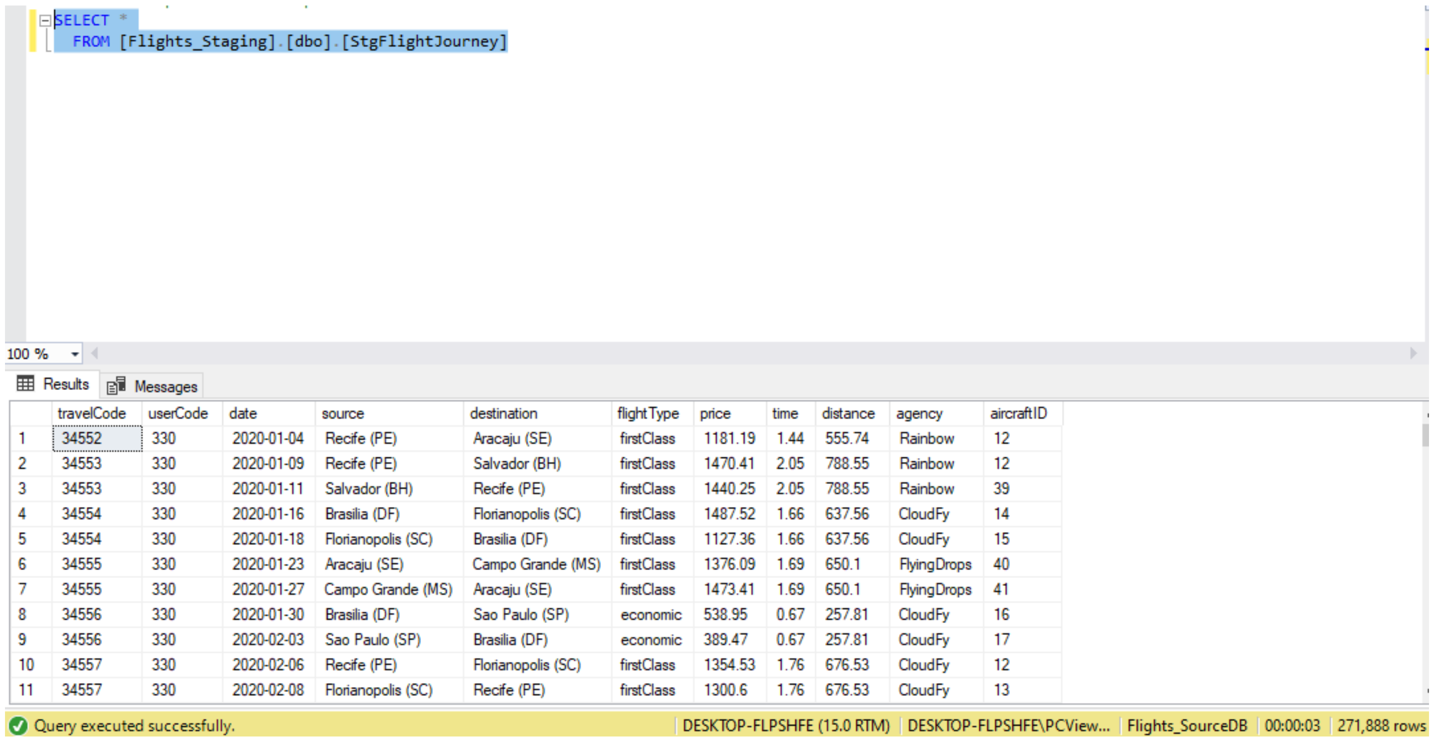


```
SELECT *
  FROM [Flights_Staging].[dbo].[StgFlightJourney]
```

	travelCode	userCode	date	source	destination	flightType	price	time	distance	agency	aircraftID
1	112879	1120	2020-10-29	Florianopolis (SC)	Brasilia (DF)	economic	636.51	1.66	637.56	CloudFy	15
2	112879	1120	2020-11-01	Brasilia (DF)	Florianopolis (SC)	economic	884.94	1.66	637.56	CloudFy	14
3	11288	111	2022-05-26	Brasilia (DF)	Aracaju (SE)	firstClass	1287.52	1.11	425.98	FlyingDrops	31
4	11288	111	2022-05-27	Aracaju (SE)	Brasilia (DF)	firstClass	898.04	1.11	425.98	FlyingDrops	30
5	112880	1120	2020-11-05	Florianopolis (SC)	Natal (RN)	premium	1114.55	1.84	709.37	Rainbow	24
6	112880	1120	2020-11-06	Natal (RN)	Florianopolis (SC)	premium	1212.58	1.84	709.37	Rainbow	25
7	112881	1120	2020-11-12	Florianopolis (SC)	Natal (RN)	firstClass	1315.27	1.84	709.37	CloudFy	24
8	112881	1120	2020-11-14	Natal (RN)	Florianopolis (SC)	firstClass	1570.02	1.84	709.37	CloudFy	25
9	112882	1120	2020-11-19	Florianopolis (SC)	Sao Paulo (SP)	premium	554.87	1.46	562.14	CloudFy	32
10	112882	1120	2020-11-21	Sao Paulo (SP)	Florianopolis (SC)	premium	1370.17	1.46	562.14	CloudFy	33
11	112883	1120	2020-11-26	Florianopolis (SC)	Brasilia (DF)	economic	636.51	1.66	637.56	CloudFy	15

Query executed successfully. | DESKTOP-FLPSHFE (15.0 RTM) | DESKTOP-FLPSHFE\PCView... | Flights_SourceDB | 00:00:16 | 1,087,552 rows

The screenshot below shows the StgFlightJourney after truncating.



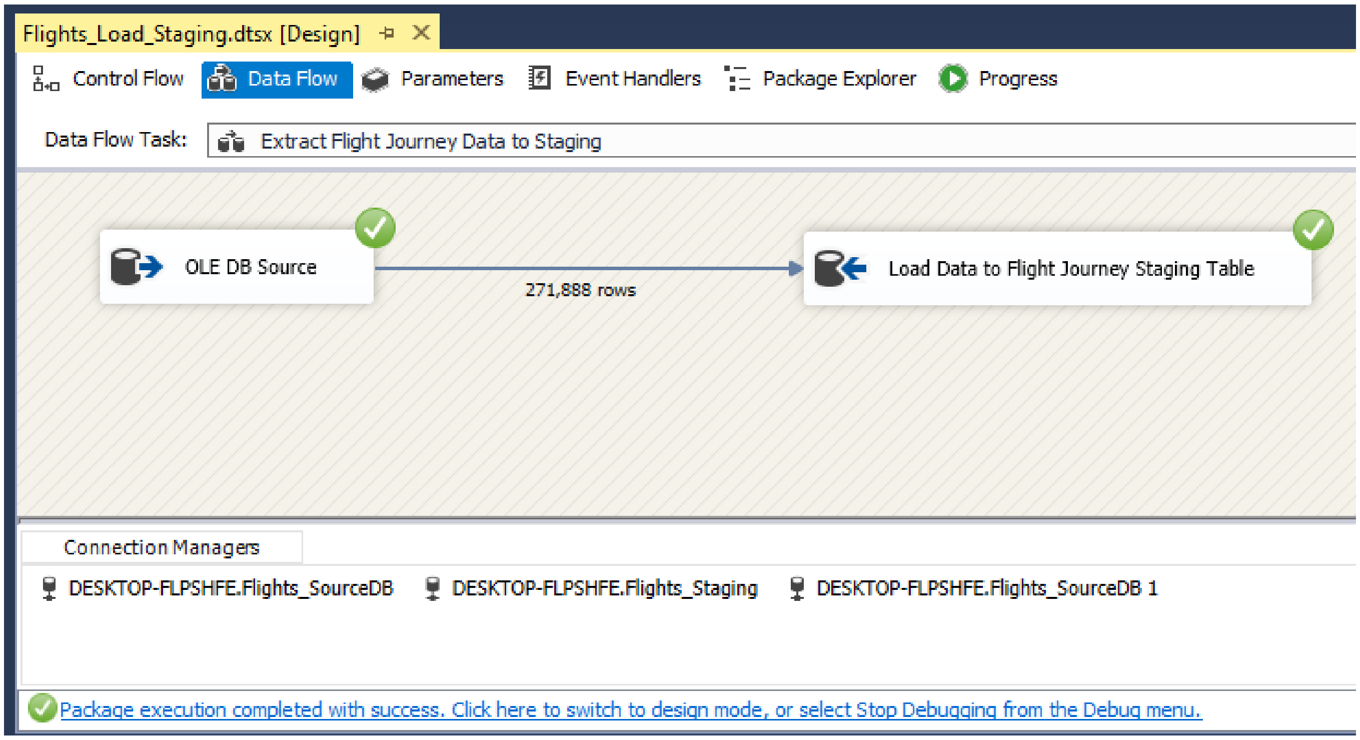
```
SELECT *
  FROM [Flights_Staging].[dbo].[StgFlightJourney]
```

	travelCode	userCode	date	source	destination	flightType	price	time	distance	agency	aircraftID
1	34552	330	2020-01-04	Recife (PE)	Aracaju (SE)	firstClass	1181.19	1.44	555.74	Rainbow	12
2	34553	330	2020-01-09	Recife (PE)	Salvador (BH)	firstClass	1470.41	2.05	788.55	Rainbow	12
3	34553	330	2020-01-11	Salvador (BH)	Recife (PE)	firstClass	1440.25	2.05	788.55	Rainbow	39
4	34554	330	2020-01-16	Brasilia (DF)	Florianopolis (SC)	firstClass	1487.52	1.66	637.56	CloudFy	14
5	34554	330	2020-01-18	Florianopolis (SC)	Brasilia (DF)	firstClass	1127.36	1.66	637.56	CloudFy	15
6	34555	330	2020-01-23	Aracaju (SE)	Campo Grande (MS)	firstClass	1376.09	1.69	650.1	FlyingDrops	40
7	34555	330	2020-01-27	Campo Grande (MS)	Aracaju (SE)	firstClass	1473.41	1.69	650.1	FlyingDrops	41
8	34556	330	2020-01-30	Brasilia (DF)	Sao Paulo (SP)	economic	538.95	0.67	257.81	CloudFy	16
9	34556	330	2020-02-03	Sao Paulo (SP)	Brasilia (DF)	economic	389.47	0.67	257.81	CloudFy	17
10	34557	330	2020-02-06	Recife (PE)	Florianopolis (SC)	firstClass	1354.53	1.76	676.53	CloudFy	12
11	34557	330	2020-02-08	Florianopolis (SC)	Recife (PE)	firstClass	1300.6	1.76	676.53	CloudFy	13

Query executed successfully. | DESKTOP-FLPSHFE (15.0 RTM) | DESKTOP-FLPSHFE\PCView... | Flights_SourceDB | 00:00:03 | 271,888 rows

The next following list of screenshots represents the staged and truncated tables.

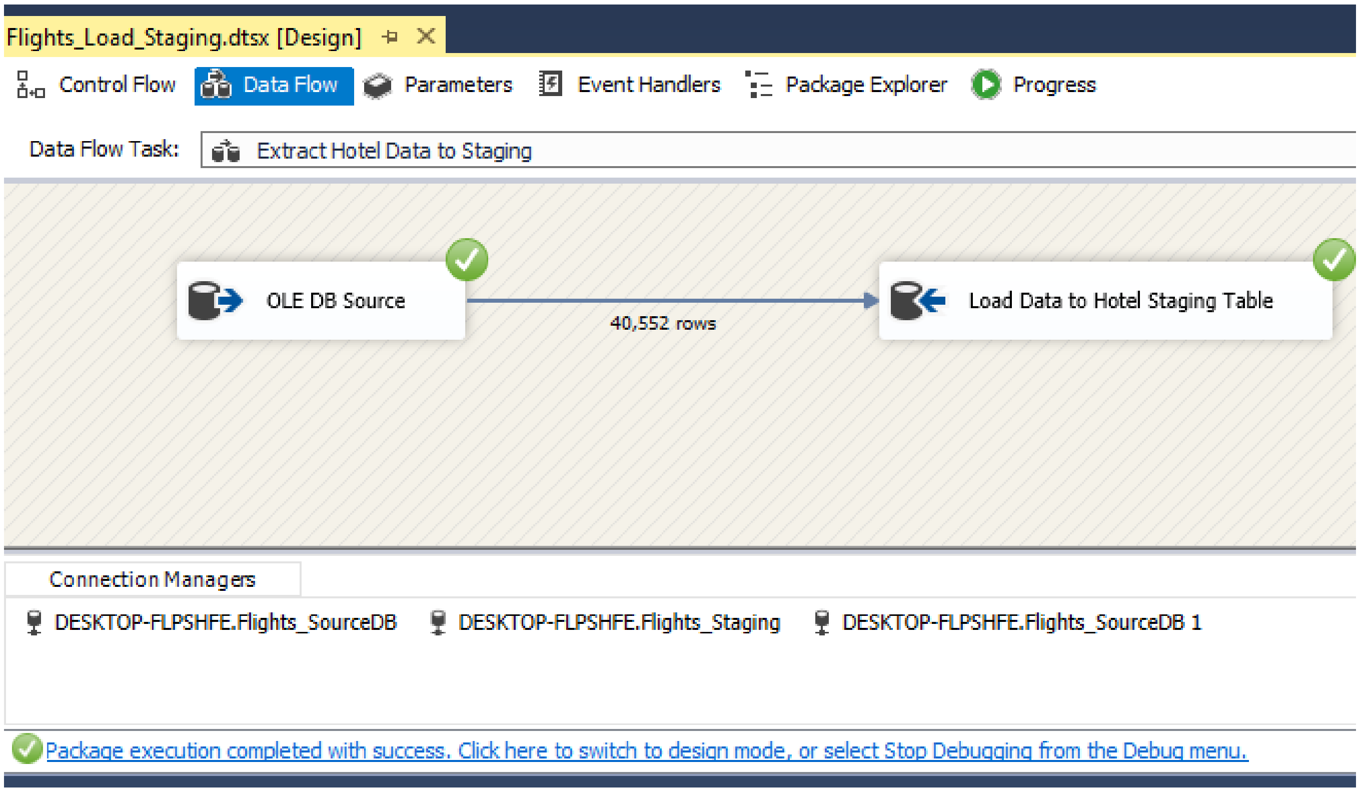
Staging Flight Journey Details


Flights_Load_Staging.dtsx [Design]
Control Flow
Data Flow
Parameters
Event Handlers
Package Explorer
Progress
Data Flow Task:
Extract Flight Journey Data to Staging
OLE DB Source
271,888 rows
Load Data to Flight Journey Staging Table
Connection Managers
DESKTOP-FLPSHFE.Flights_SourceDB
DESKTOP-FLPSHFE.Flights_Staging
DESKTOP-FLPSHFE.Flights_SourceDB 1

Package execution completed with success. Click here to switch to design mode, or select Stop Debugging from the Debug menu.

When staging the flight journey details data is extracted from the FlightJourney table in the source database and inserted into the FlightJourney staging table in the staging database.

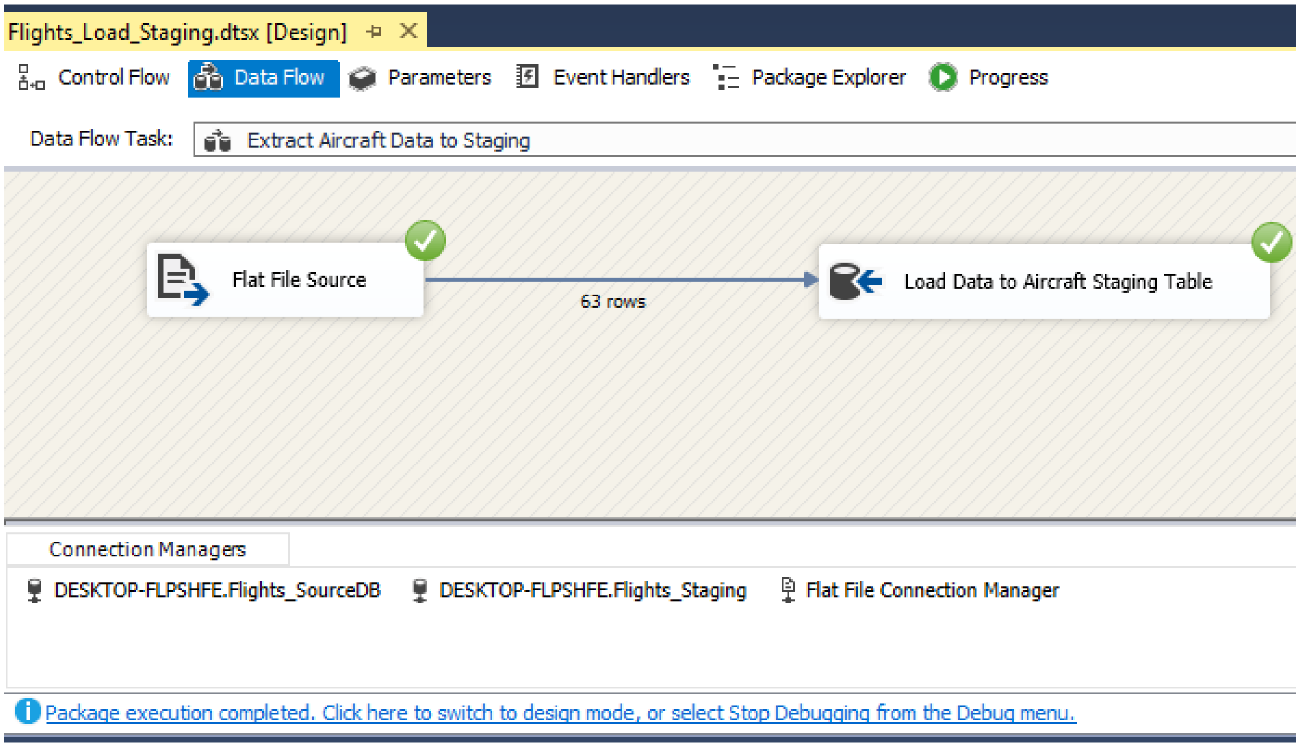
Staging Hotel Details


Flights_Load_Staging.dtsx [Design]
Control Flow
Data Flow
Parameters
Event Handlers
Package Explorer
Progress
Data Flow Task:
Extract Hotel Data to Staging
OLE DB Source
40,552 rows
Load Data to Hotel Staging Table
Connection Managers
DESKTOP-FLPSHFE.Flights_SourceDB
DESKTOP-FLPSHFE.Flights_Staging
DESKTOP-FLPSHFE.Flights_SourceDB 1

Package execution completed with success. Click here to switch to design mode, or select Stop Debugging from the Debug menu.

When staging the hotel details data is extracted from the Hotel table in the source database and inserted into the Hotel staging table in the staging database.

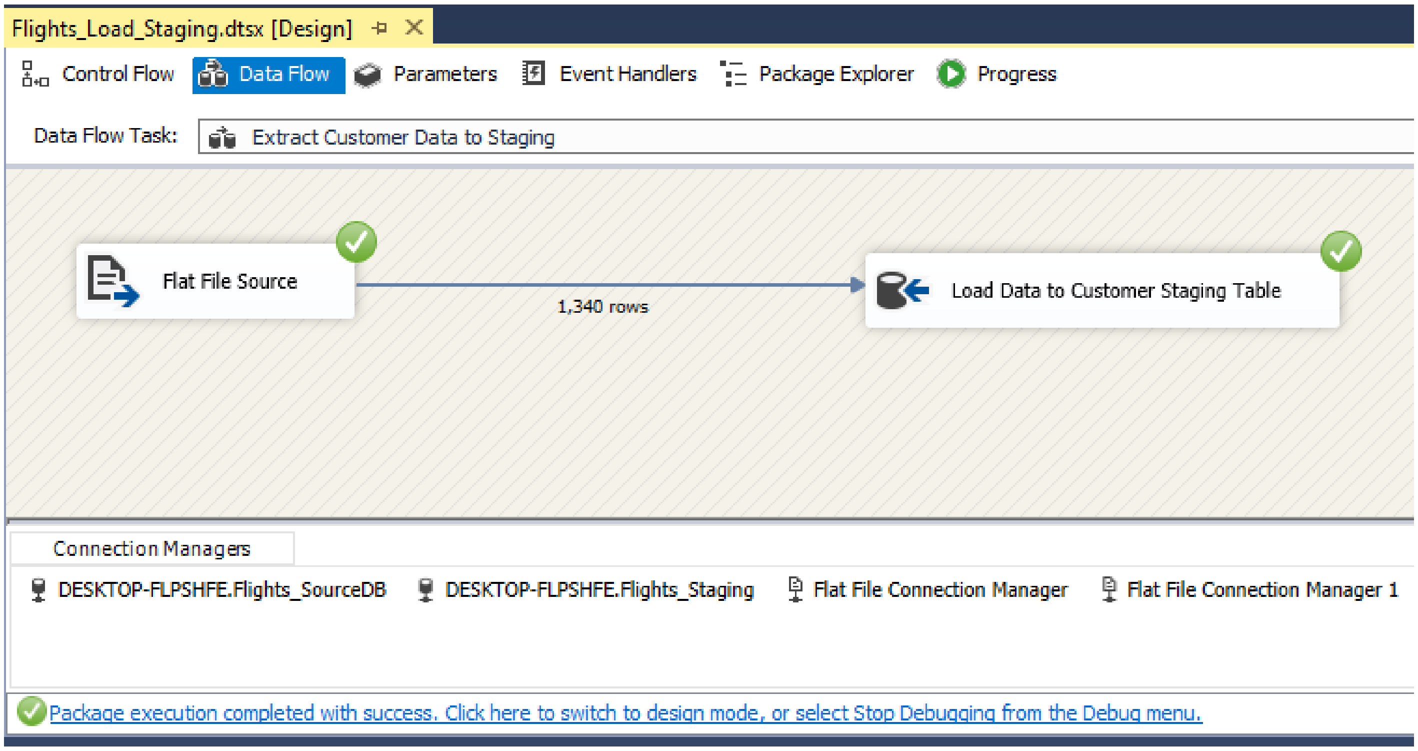
Aircraft details staging


Flights_Load_Staging.dtsx [Design]
Control Flow
Data Flow
Parameters
Event Handlers
Package Explorer
Progress
Data Flow Task:
Extract Aircraft Data to Staging
Flat File Source
63 rows
Load Data to Aircraft Staging Table
Connection Managers
DESKTOP-FLPSHFE.Flights_SourceDB
DESKTOP-FLPSHFE.Flights_Staging
Flat File Connection Manager

Package execution completed. Click here to switch to design mode, or select Stop Debugging from the Debug menu.

When staging the aircraft details data is extracted from the Aircraft.csv file and inserted into the Aircraft staging table in the staging database.

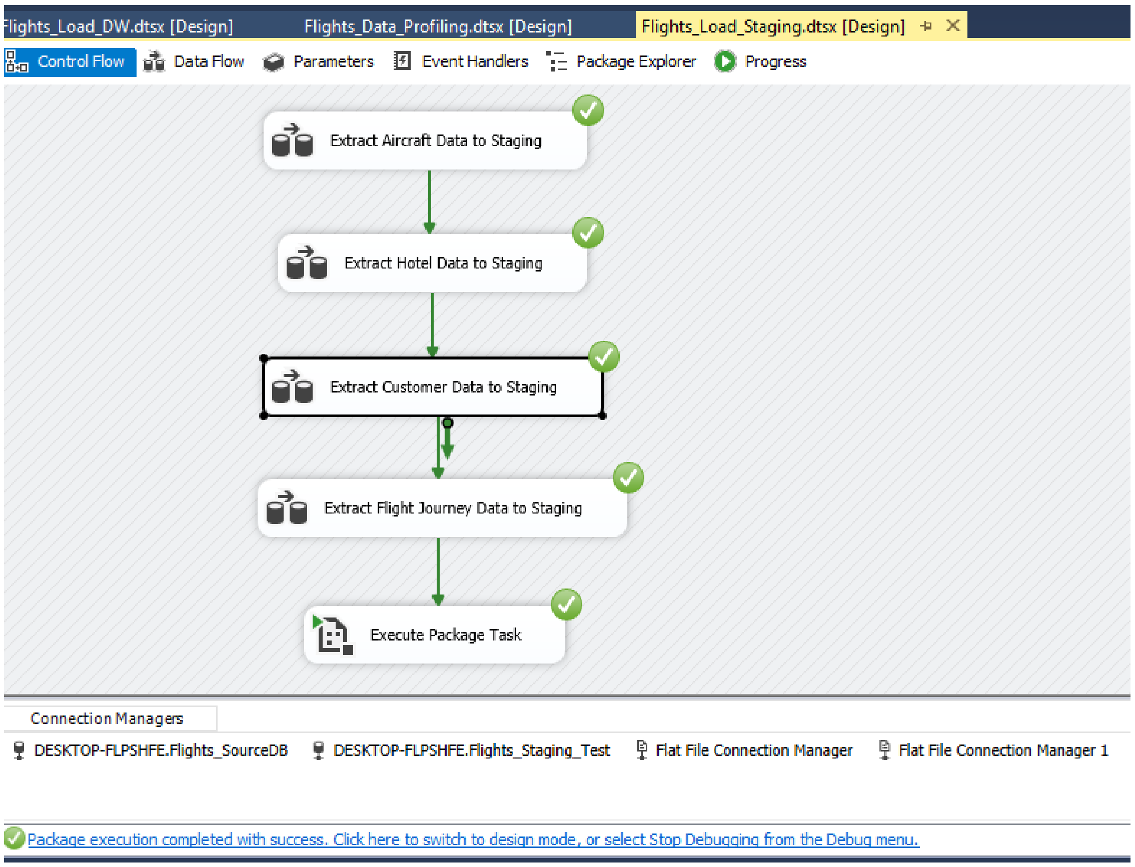
Staging customer details


Flights_Load_Staging.dtsx [Design]
Control Flow
Data Flow
Parameters
Event Handlers
Package Explorer
Progress
Data Flow Task:
Extract Customer Data to Staging
Flat File Source
1,340 rows
Load Data to Customer Staging Table
Connection Managers
DESKTOP-FLPSHFE.Flights_SourceDB
DESKTOP-FLPSHFE.Flights_Staging
Flat File Connection Manager
Flat File Connection Manager 1

Package execution completed with success. Click here to switch to design mode, or select Stop Debugging from the Debug menu.

When staging the customer details data is extracted from the Customers.txt file and inserted into the Customer staging table in the staging database.

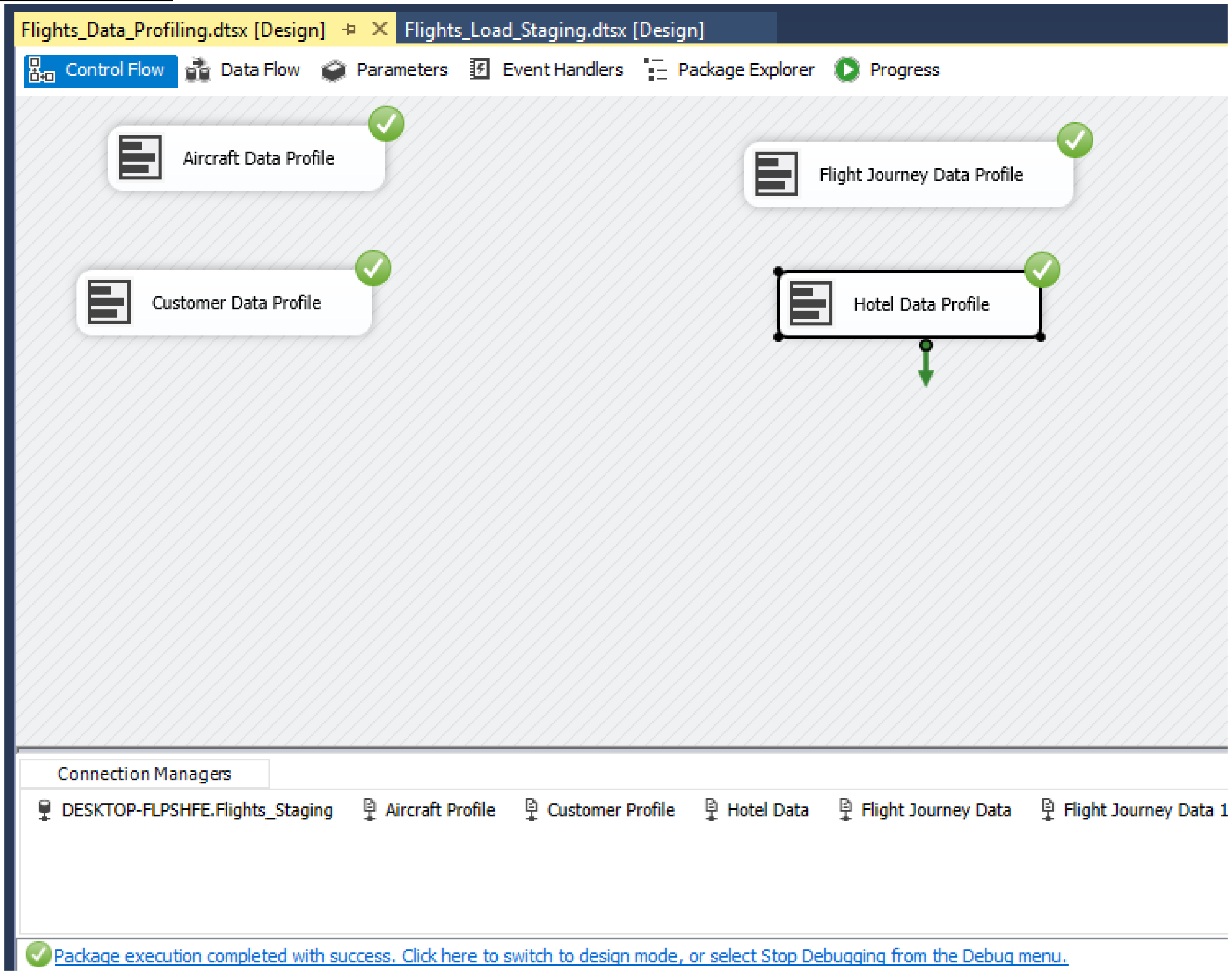
After connecting all the data flow tasks and executing them there were no errors shown as shown below.


Flights_Load_DW.dtsx [Design]
Flights_Data_Profiling.dtsx [Design]
Flights_Load_Staging.dtsx [Design]
Control Flow
Data Flow
Parameters
Event Handlers
Package Explorer
Progress
Extract Aircraft Data to Staging
Extract Hotel Data to Staging
Extract Customer Data to Staging
Extract Flight Journey Data to Staging
Execute Package Task
Connection Managers
DESKTOP-FLPSHFE.Flights_SourceDB
DESKTOP-FLPSHFE.Flights_Staging_Test
Flat File Connection Manager
Flat File Connection Manager 1

Package execution completed with success. Click here to switch to design mode, or select Stop Debugging from the Debug menu.

The next step in the ETL development is profiling the data and all the staged tables are being profiled. The data profiles are available in the "Data Profiles" folder. The data profiles were successfully executed as shown below.

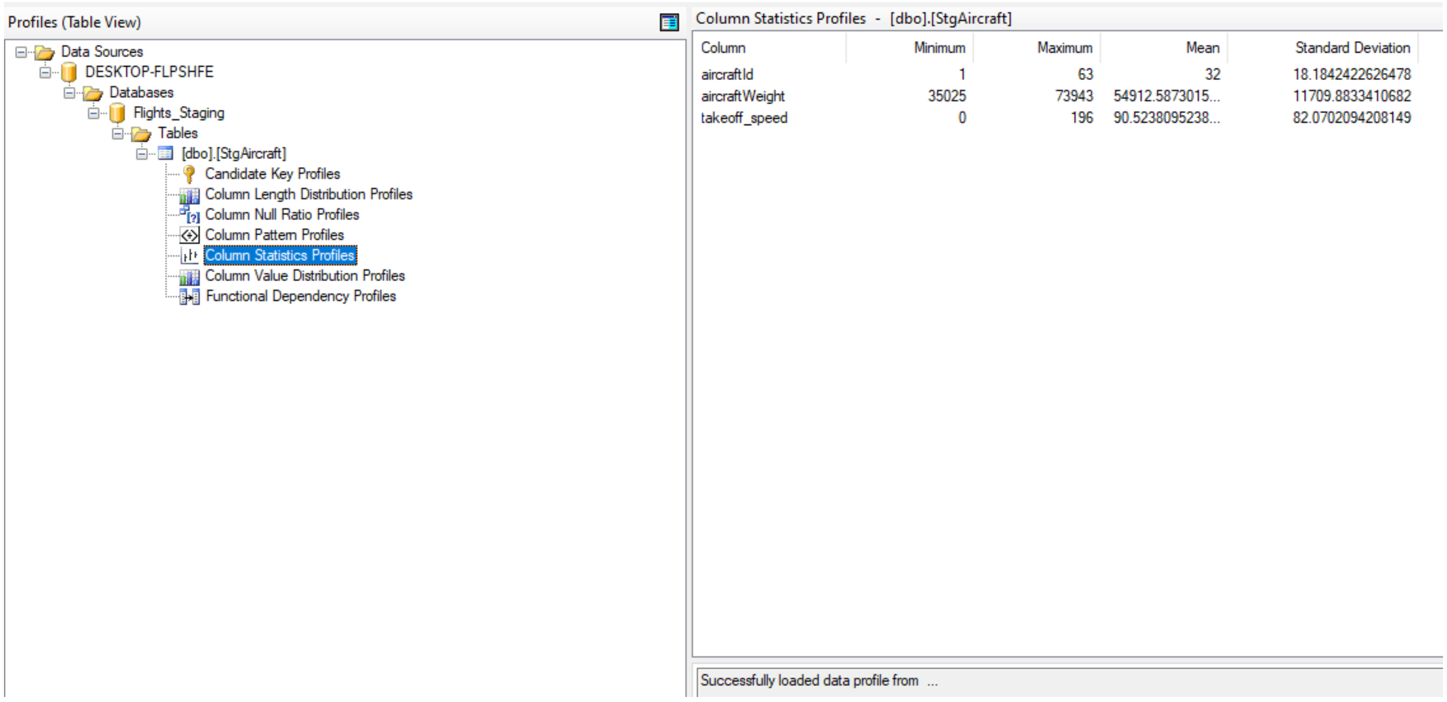
Data Profiling



Package execution completed with success. Click here to switch to design mode, or select Stop Debugging from the Debug menu.

The next series of screenshots shows the individual data profiles.

Aircrafts data profile



Profiles (Table View)

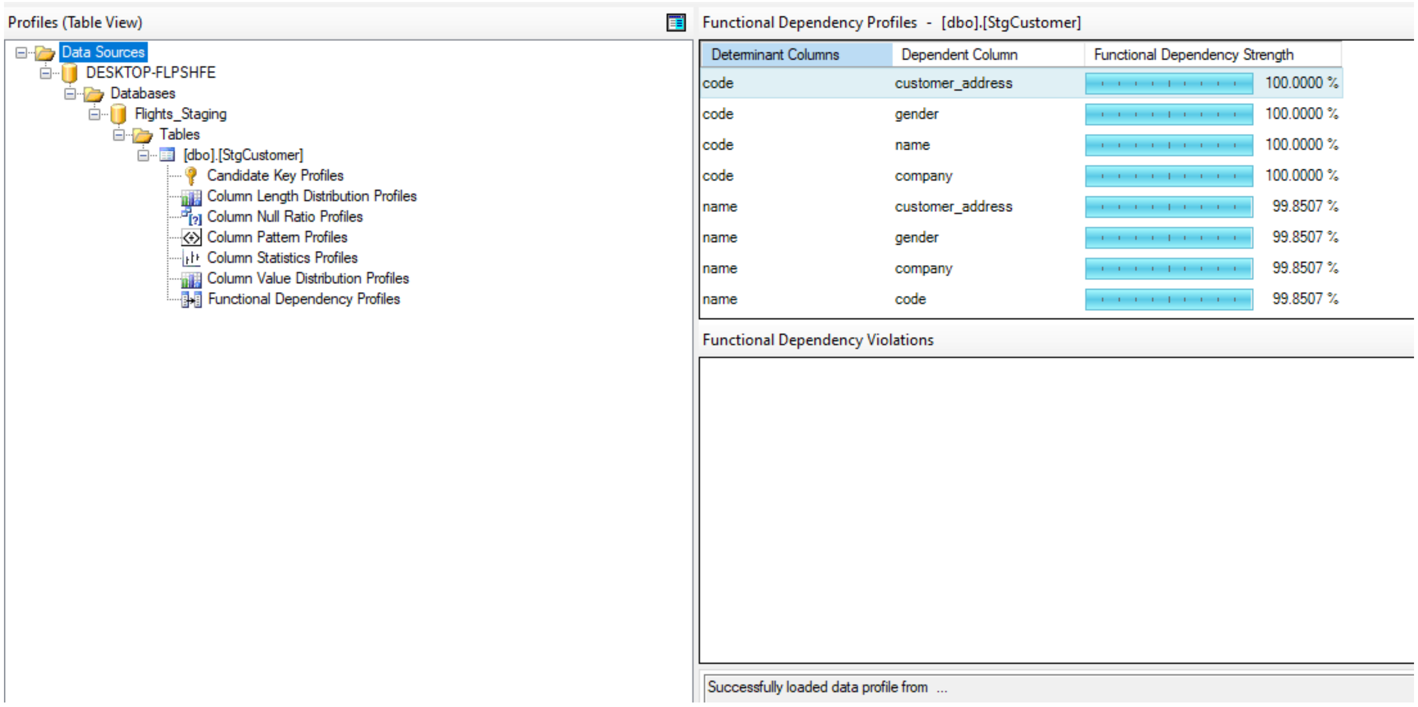
- Data Sources
  - DESKTOP-FLPSHFE
    - Databases
      - Flights_Staging
        - Tables
          - [dbo].[StgAircraft]
            - Candidate Key Profiles
            - Column Length Distribution Profiles
            - Column Null Ratio Profiles
            - Column Pattern Profiles
            - Column Statistics Profiles
            - Column Value Distribution Profiles
            - Functional Dependency Profiles

Column Statistics Profiles - [dbo].[StgAircraft]

Column	Minimum	Maximum	Mean	Standard Deviation
aircraftId	1	63	32	18.1842422626478
aircraftWeight	35025	73943	54912.5873015...	11709.8833410682
takeoff_speed	0	196	90.5238095238...	82.0702094208149

Successfully loaded data profile from ...

Customer Data Profile



Profiles (Table View)

- Data Sources
  - DESKTOP-FLPSHFE
    - Databases
      - Flights_Staging
        - Tables
          - [dbo].[StgCustomer]
            - Candidate Key Profiles
            - Column Length Distribution Profiles
            - Column Null Ratio Profiles
            - Column Pattern Profiles
            - Column Statistics Profiles
            - Column Value Distribution Profiles
            - Functional Dependency Profiles

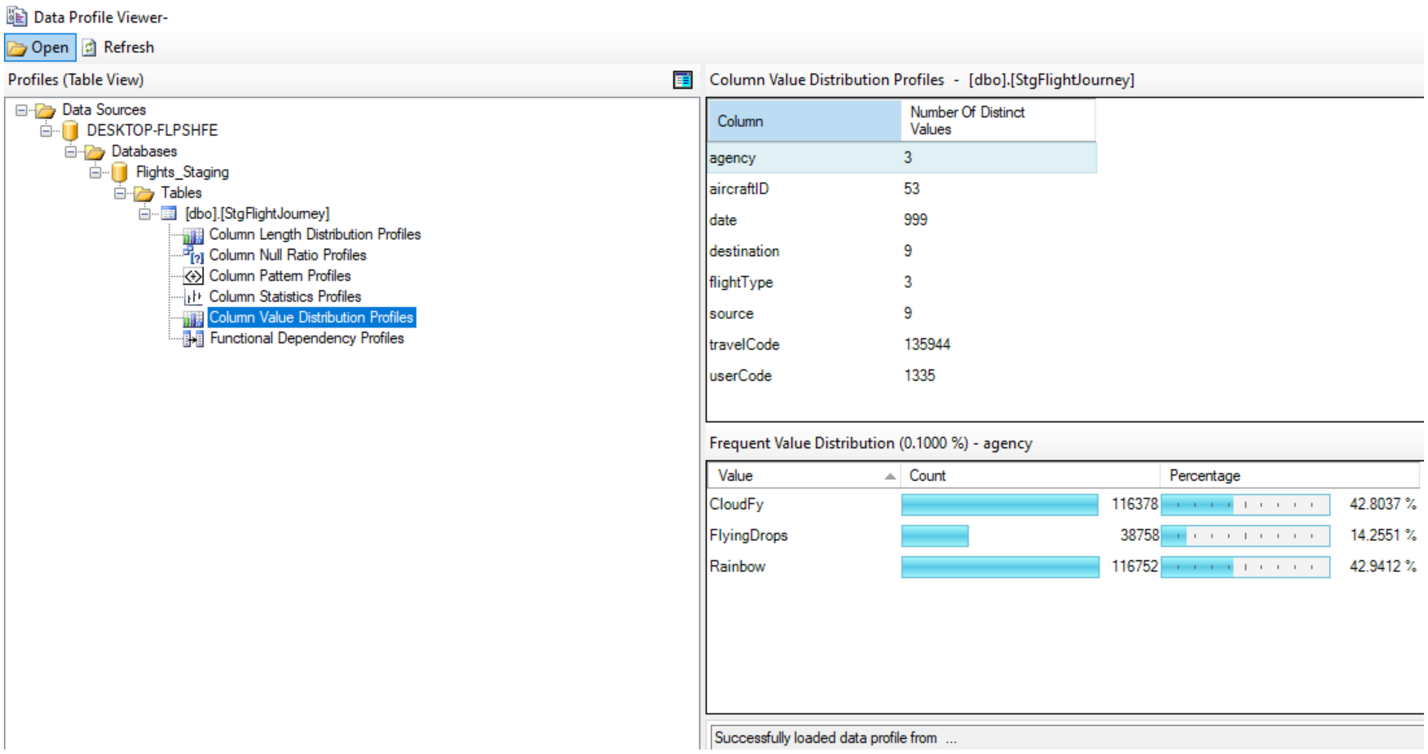
Functional Dependency Profiles - [dbo].[StgCustomer]

Determinant Columns	Dependent Column	Functional Dependency Strength
code	customer_address	100.0000 %
code	gender	100.0000 %
code	name	100.0000 %
code	company	100.0000 %
name	customer_address	99.8507 %
name	gender	99.8507 %
name	company	99.8507 %
name	code	99.8507 %

Functional Dependency Violations

Successfully loaded data profile from ...

Flight Journey Data Profile



Data Profile Viewer-

Open | Refresh

Profiles (Table View)

- Data Sources
  - DESKTOP-FLPSHFE
    - Databases
      - Flights_Staging
        - Tables
          - [dbo].[StgFlightJourney]
            - Column Length Distribution Profiles
            - Column Null Ratio Profiles
            - Column Pattern Profiles
            - Column Statistics Profiles
            - Column Value Distribution Profiles
            - Functional Dependency Profiles

Column Value Distribution Profiles - [dbo].[StgFlightJourney]

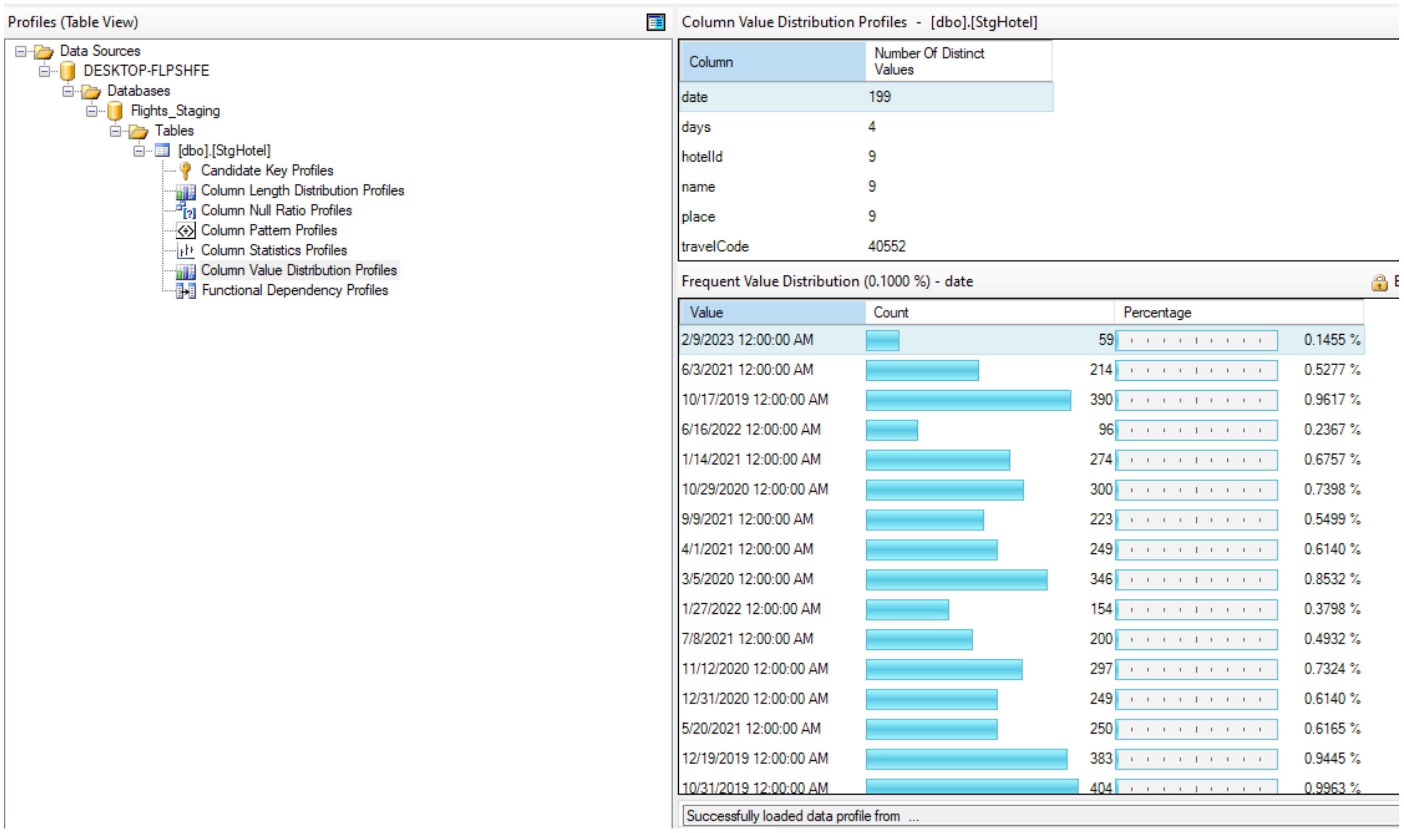
Column	Number Of Distinct Values
agency	3
aircraftID	53
date	999
destination	9
flightType	3
source	9
travelCode	135944
userCode	1335

Frequent Value Distribution (0.1000 %) - agency

Value	Count	Percentage
CloudFy	116378	42.8037 %
FlyingDrops	38758	14.2551 %
Rainbow	116752	42.9412 %

Successfully loaded data profile from ...

Hotel data profile



Profiles (Table View)

- Data Sources
  - DESKTOP-FLPSHFE
    - Databases
      - Flights_Staging
        - Tables
          - [dbo].[StgHotel]
            - Candidate Key Profiles
            - Column Length Distribution Profiles
            - Column Null Ratio Profiles
            - Column Pattern Profiles
            - Column Statistics Profiles
            - Column Value Distribution Profiles
            - Functional Dependency Profiles

Column Value Distribution Profiles - [dbo].[StgHotel]

Column	Number Of Distinct Values
date	199
days	4
hotelId	9
name	9
place	9
travelCode	40552

Frequent Value Distribution (0.1000 %) - date

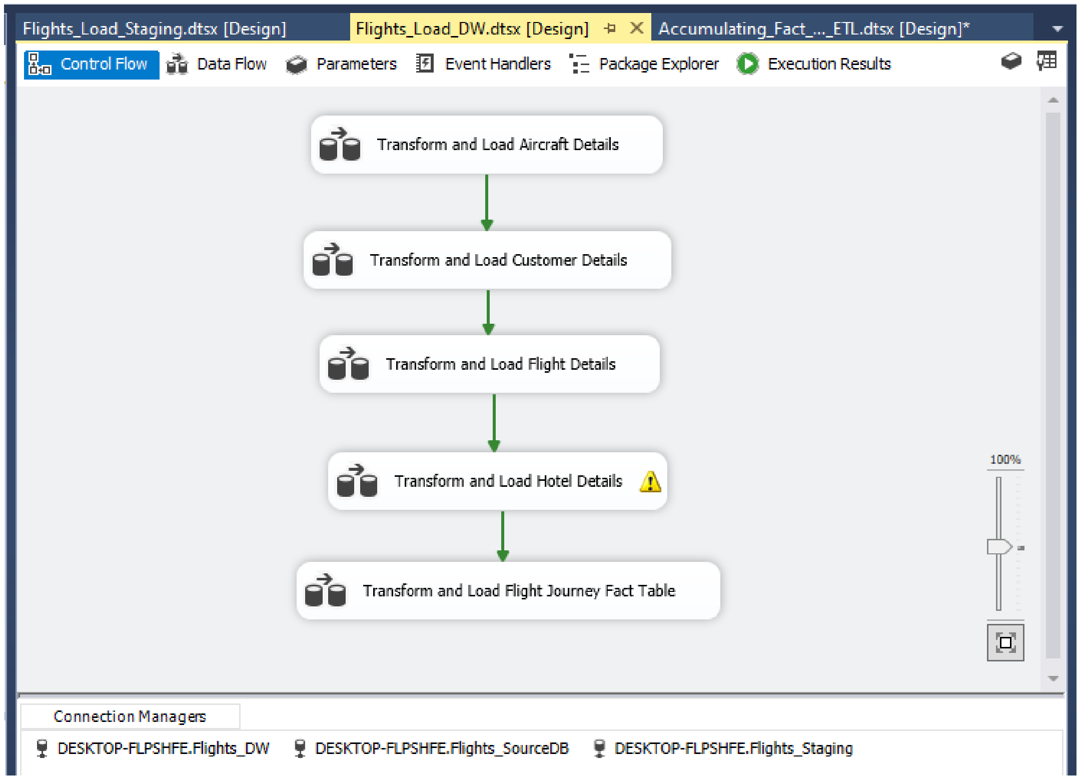
Value	Count	Percentage
2/9/2023 12:00:00 AM	59	0.1455 %
6/3/2021 12:00:00 AM	214	0.5277 %
10/17/2019 12:00:00 AM	390	0.9617 %
6/16/2022 12:00:00 AM	96	0.2367 %
1/14/2021 12:00:00 AM	274	0.6757 %
10/29/2020 12:00:00 AM	300	0.7398 %
9/9/2021 12:00:00 AM	223	0.5499 %
4/1/2021 12:00:00 AM	249	0.6140 %
3/5/2020 12:00:00 AM	346	0.8532 %
1/27/2022 12:00:00 AM	154	0.3798 %
7/8/2021 12:00:00 AM	200	0.4932 %
11/12/2020 12:00:00 AM	297	0.7324 %
12/31/2020 12:00:00 AM	249	0.6140 %
5/20/2021 12:00:00 AM	250	0.6165 %
12/19/2019 12:00:00 AM	383	0.9445 %
10/31/2019 12:00:00 AM	404	0.9963 %

Successfully loaded data profile from ...

Transform and Load

After data profiles were created the next step of ETL development is loading the data from the staging database tables to the data warehouse facts and dimension tables.

The screenshot below shows the order of execution of the data flow tasks.


Flights_Load_Staging.dtsx [Design]
Flights_Load_DW.dtsx [Design]
Accumulating_Fact_..._ETL.dtsx [Design]*
Control Flow
Data Flow
Parameters
Event Handlers
Package Explorer
Execution Results
Transform and Load Aircraft Details
Transform and Load Customer Details
Transform and Load Flight Details
Transform and Load Hotel Details
Transform and Load Flight Journey Fact Table
100%
Connection Managers
DESKTOP-FLPSHFE.Flights_DW
DESKTOP-FLPSHFE.Flights_SourceDB
DESKTOP-FLPSHFE.Flights_Staging

Since we are not maintaining the history of the product other than customer we can have the updated records in the data warehouse so to achieve this the stored procedure approach is used.

Date Dimension

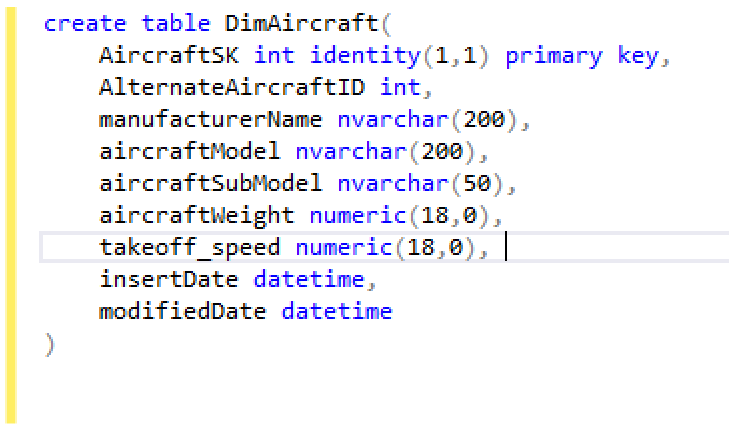
The query used to create the date dimension is shown below:

```
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

/*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)

/********************************************************************************************/
--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)
            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

        -- 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
        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
```

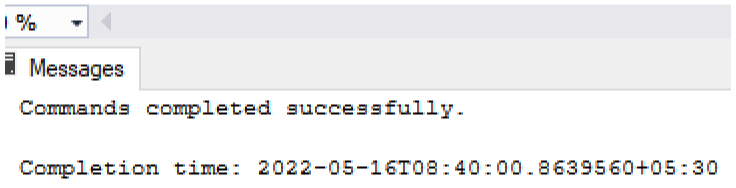
```
        SET @CurrentDate = DATEADD(DD, 1, @CurrentDate)
END

/*****************************************************************************************/

/**************************************************************************************/

SELECT * FROM [dbo].[DimDate]
```

Transforming and loading data to aircraft dimension

The query used to create the aircraft dimension is shown below:

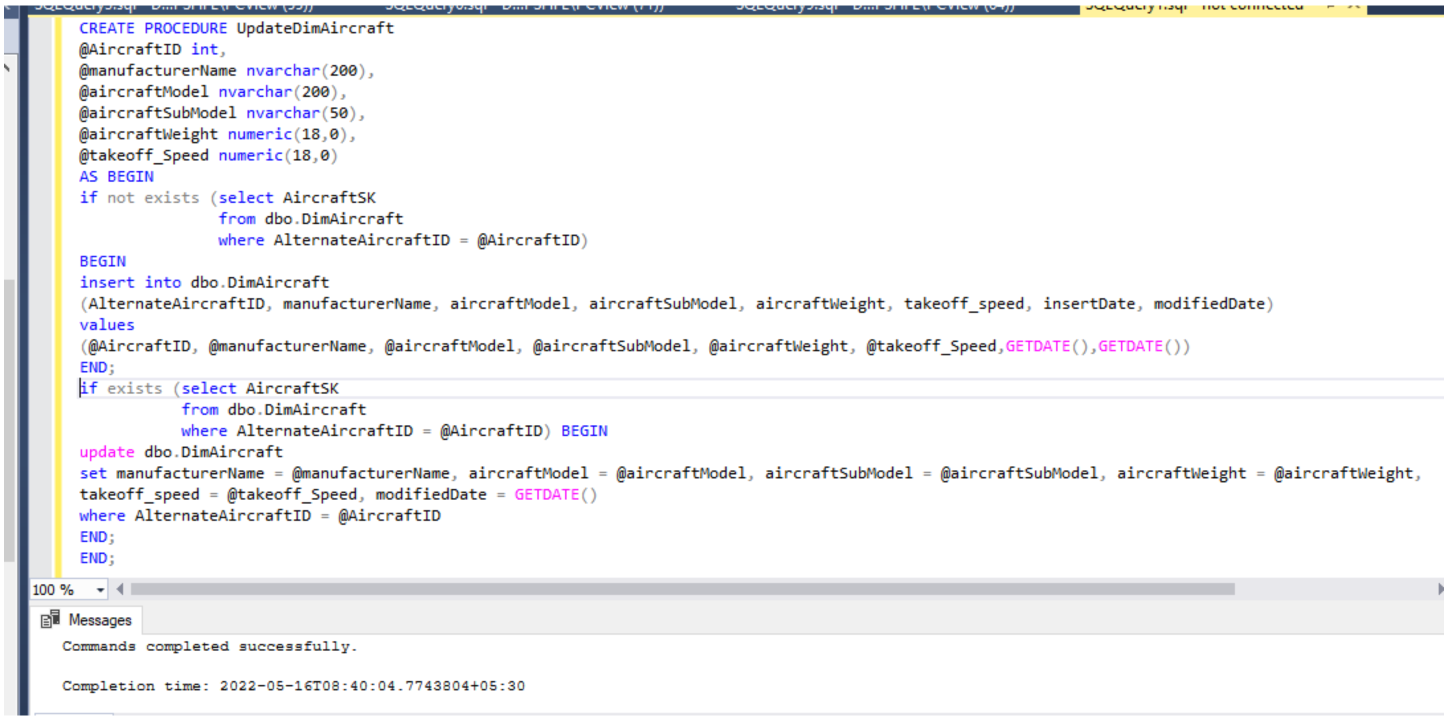


```
create table DimAircraft(
    AircraftSK int identity(1,1) primary key,
    AlternateAircraftID int,
    manufacturerName nvarchar(200),
    aircraftModel nvarchar(200),
    aircraftSubModel nvarchar(50),
    aircraftWeight numeric(18,0),
    takeoff_speed numeric(18,0),
    insertDate datetime,
    modifiedDate datetime
)
```



```
Commands completed successfully.

Completion time: 2022-05-16T08:40:00.8639560+05:30
```

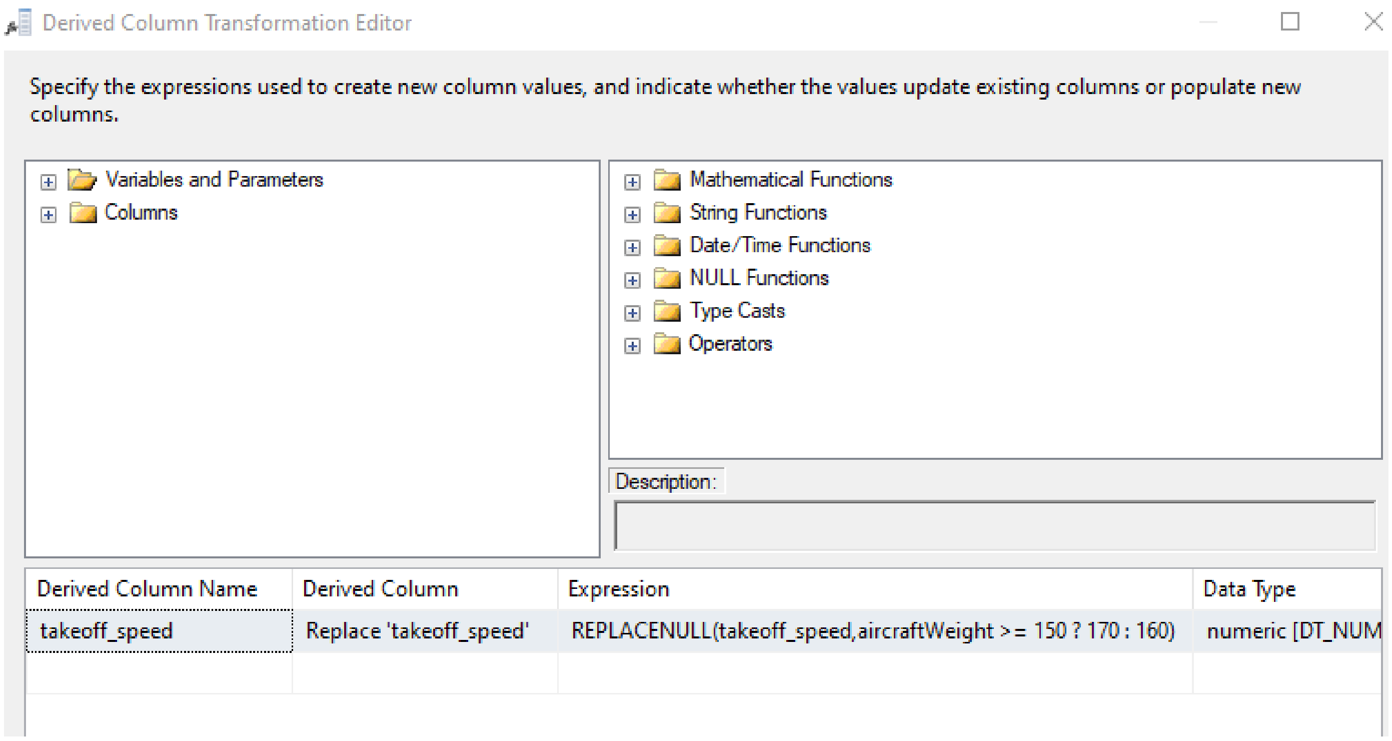
The query used to update the aircraft dimension is shown below:



```
CREATE PROCEDURE UpdateDimAircraft
@AircraftID int,
@manufacturerName nvarchar(200),
@aircraftModel nvarchar(200),
@aircraftSubModel nvarchar(50),
@aircraftWeight numeric(18,0),
@takeoff_Speed numeric(18,0)
AS BEGIN
if not exists (select AircraftSK
               from dbo.DimAircraft
               where AlternateAircraftID = @AircraftID)
BEGIN
insert into dbo.DimAircraft
(AlternateAircraftID, manufacturerName, aircraftModel, aircraftSubModel, aircraftWeight, takeoff_speed, insertDate, modifiedDate)
values
(@AircraftID, @manufacturerName, @aircraftModel, @aircraftSubModel, @aircraftWeight, @takeoff_Speed,GETDATE(),GETDATE())
END;
if exists (select AircraftSK
           from dbo.DimAircraft
           where AlternateAircraftID = @AircraftID) BEGIN
update dbo.DimAircraft
set manufacturerName = @manufacturerName, aircraftModel = @aircraftModel, aircraftSubModel = @aircraftSubModel, aircraftWeight = @aircraftWeight,
takeoff_speed = @takeoff_Speed, modifiedDate = GETDATE()
where AlternateAircraftID = @AircraftID
END;
END;
```

```
Commands completed successfully.

Completion time: 2022-05-16T08:40:04.7743804+05:30
```

To replace the NULL values in the takeoff_speed column with some predefined values the following screenshot shows how it is done.

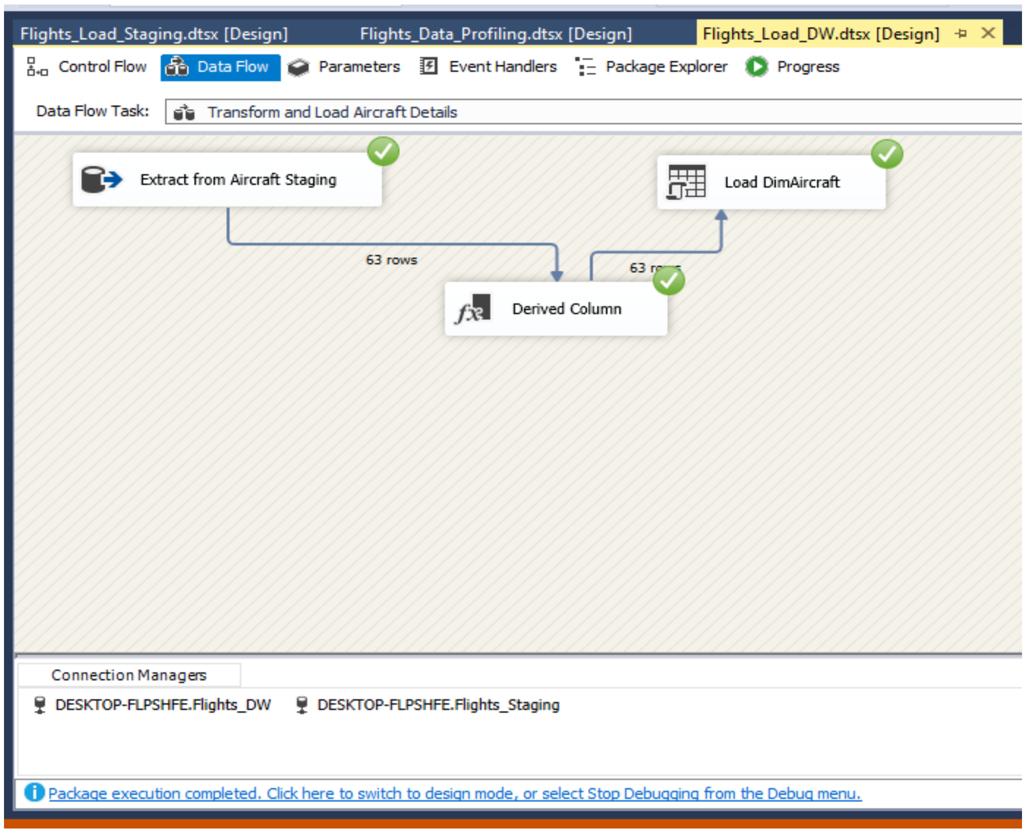


Derived Column Transformation Editor

Specify the expressions used to create new column values, and indicate whether the values update existing columns or populate new columns.

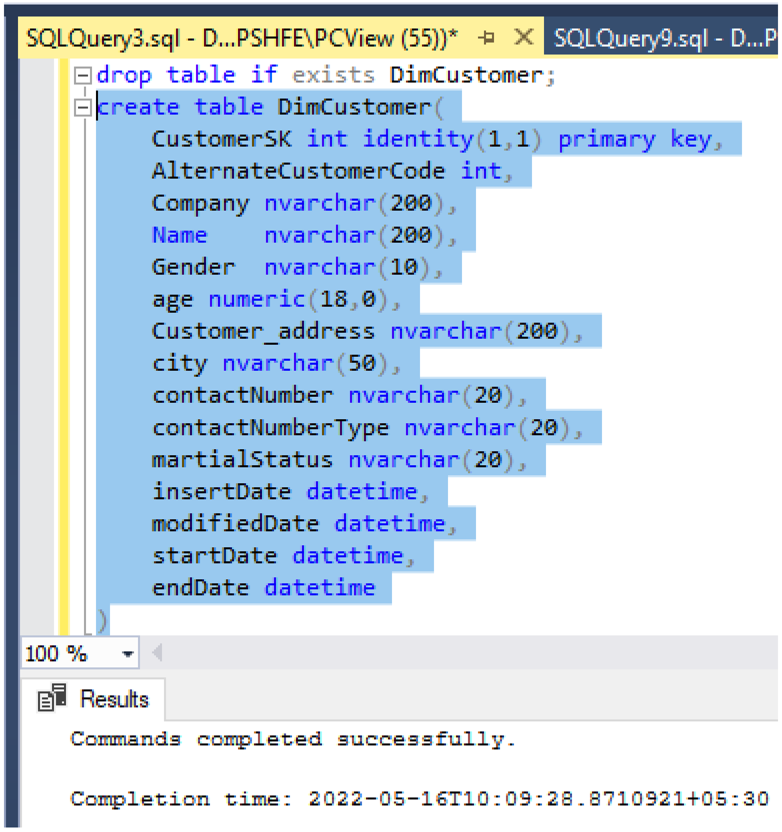
Derived Column Name	Derived Column	Expression	Data Type
takeoff_speed	Replace 'takeoff_speed'	REPLACENULL(takeoff_speed,aircraftWeight >= 150 ? 170 : 160)	numeric [DT_NUM

The data was being successfully extracted from the StgAircraft table in the Staging database and inserted into the DimAircraft table in the Data warehouse.


Flights_Load_Staging.dtsx [Design]
Flights_Data_Profiling.dtsx [Design]
Flights_Load_DW.dtsx [Design]
Control Flow
Data Flow
Parameters
Event Handlers
Package Explorer
Progress
Data Flow Task:
Transform and Load Aircraft Details
Extract from Aircraft Staging
Load DimAircraft
63 rows
63 rows
Derived Column
Connection Managers
DESKTOP-FLPSHFE.Flights_DW
DESKTOP-FLPSHFE.Flights_Staging
Package execution completed. Click here to switch to design mode, or select Stop Debugging from the Debug menu.

Transforming and loading data to customer dimension

The query used to create the table is shown below:



```
drop table if exists DimCustomer;
create table DimCustomer(
        CustomerSK int identity(1,1) primary key,
        AlternateCustomerCode int,
        Company nvarchar(200),
        Name    nvarchar(200),
        Gender  nvarchar(10),
        age numeric(18,0),
        Customer_address nvarchar(200),
        city nvarchar(50),
        contactNumber nvarchar(20),
        contactNumberType nvarchar(20),
        martialStatus nvarchar(20),
        insertDate datetime,
        modifiedDate datetime,
        startDate datetime,
        endDate datetime
)
```

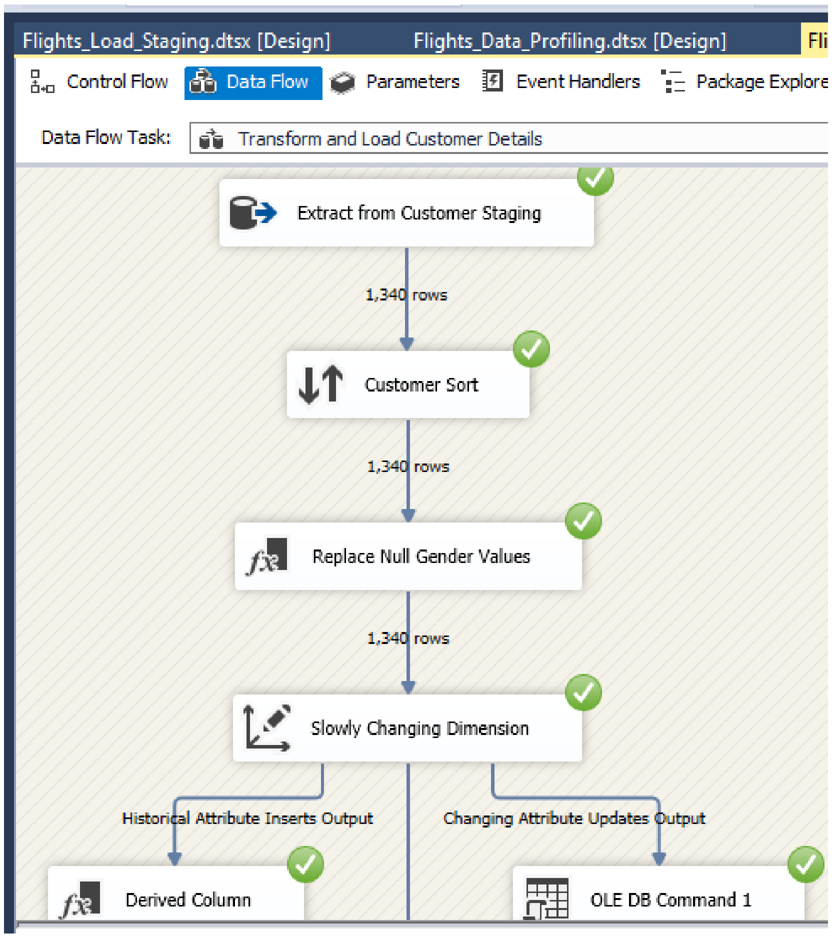
```
Commands completed successfully.

Completion time: 2022-05-16T10:09:28.8710921+05:30
```

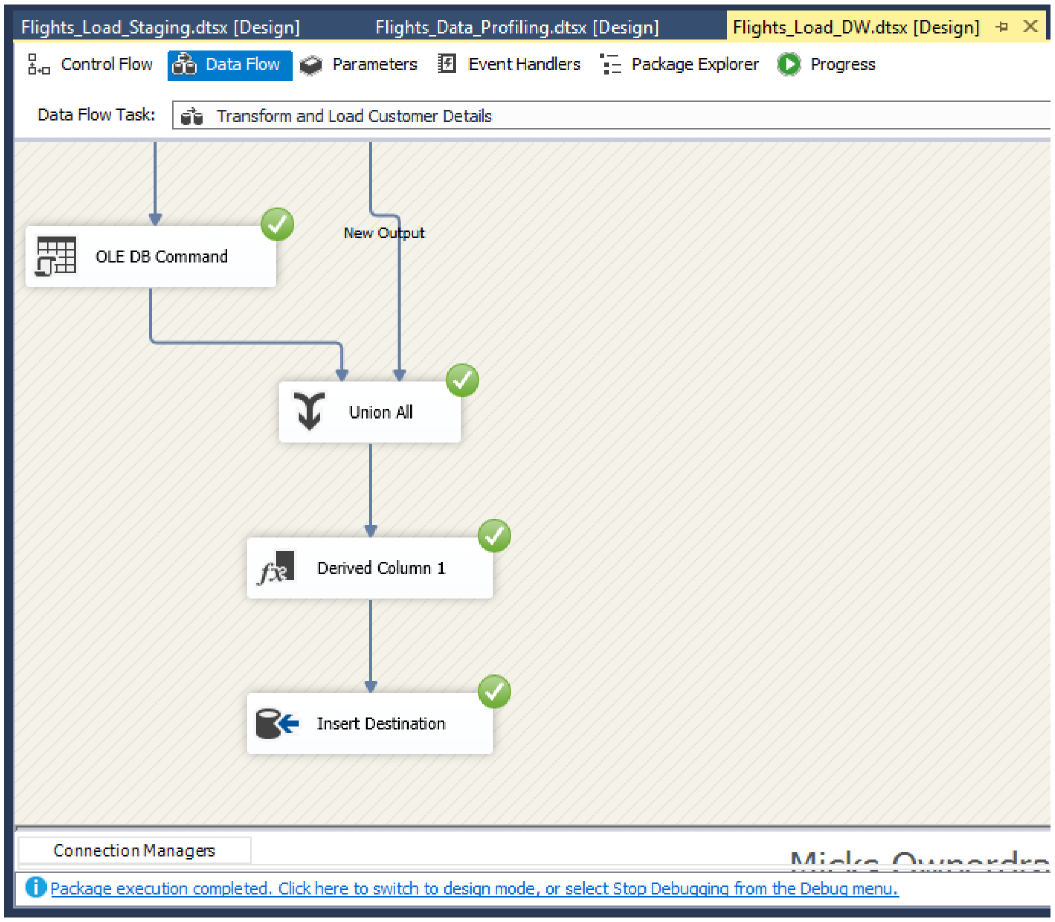
Customer dimension is considered as a slowly changing dimensions.

The historical attributes are company, customer_address, city.

The changing attributes are phoneno, phonenumbertype, martialstatus.

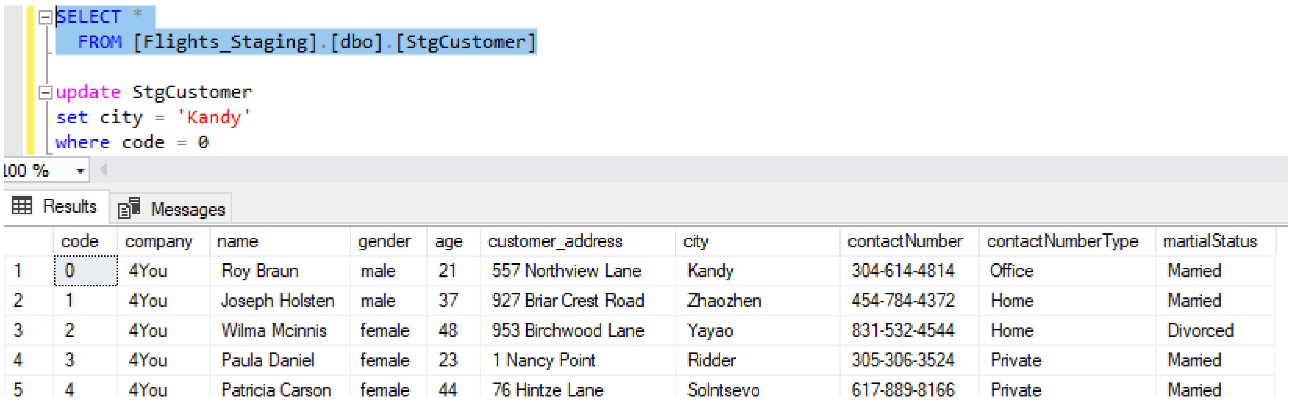
Data was extracted from the StgCustomer table in the Staging database and it was sorted according to the customer code. Null values were identified in the gender column so the data was cleansed using a derived column and then the data was passed into the slowly changing dimension. The data was successfully loaded as shown below from StgCustomer in Staging database to DimCustomer dimension in the data warehouse.


Flights_Load_Staging.dtsx [Design]
Flights_Data_Profiling.dtsx [Design]
Fli
Control Flow
Data Flow
Parameters
Event Handlers
Package Explore
Data Flow Task:
Transform and Load Customer Details
Extract from Customer Staging
1,340 rows
Customer Sort
1,340 rows
Replace Null Gender Values
1,340 rows
Slowly Changing Dimension
Historical Attribute Inserts Output
Changing Attribute Updates Output
Derived Column
OLE DB Command 1



Package execution completed. Click here to switch to design mode, or select Stop Debugging from the Debug menu.

The screenshot below shows when the StgCustomer table city column is updated from Insrom to Kandy.

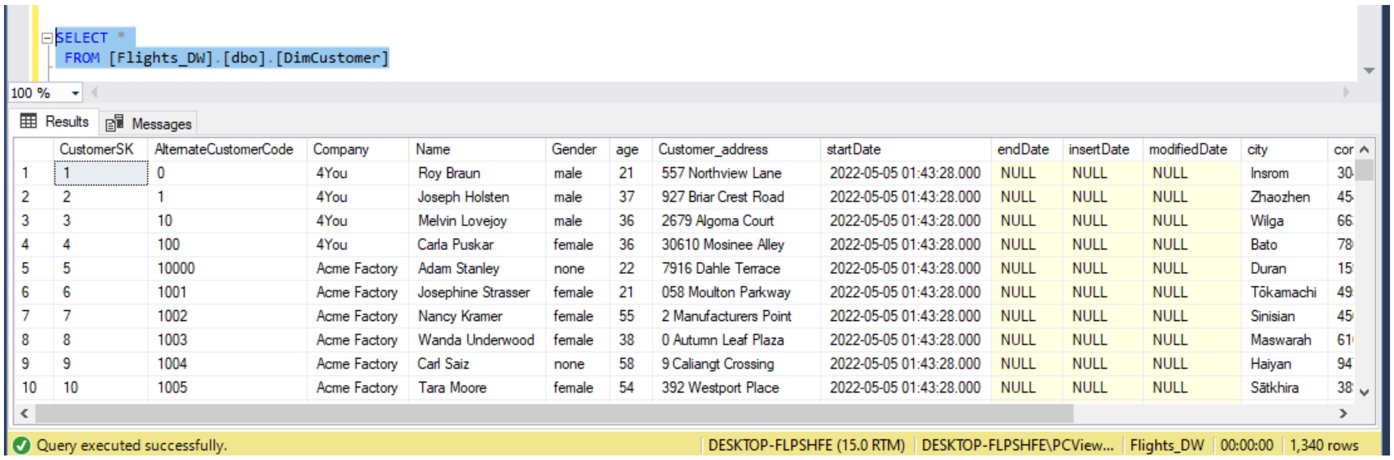


```
SELECT *
  FROM [Flights_Staging].[dbo].[StgCustomer]

update StgCustomer
set city = 'Kandy'
where code = 0
```

	code	company	name	gender	age	customer_address	city	contactNumber	contactNumberType	martialStatus
1	0	4You	Roy Braun	male	21	557 Northview Lane	Kandy	304-614-4814	Office	Married
2	1	4You	Joseph Holsten	male	37	927 Briar Crest Road	Zhaozhen	454-784-4372	Home	Married
3	2	4You	Wilma Mcinnis	female	48	953 Birchwood Lane	Yayao	831-532-4544	Home	Divorced
4	3	4You	Paula Daniel	female	23	1 Nancy Point	Ridder	305-306-3524	Private	Married
5	4	4You	Patricia Carson	female	44	76 Hintze Lane	Solntsevo	617-889-8166	Private	Married

After the customer staging table was updated the, process was executed again and the following changes were observed as follows:

Before updating the StgCustomer table (no rows were added or modified on DimCustomer table). In the original table there are only 1340 rows.

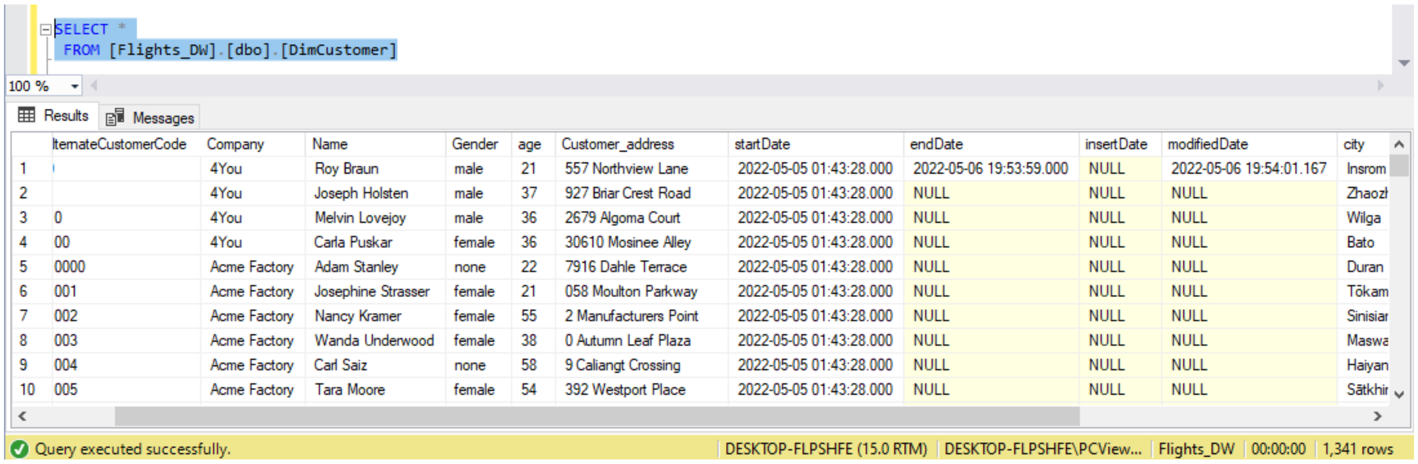


```
SELECT *
  FROM [Flights_DW].[dbo].[DimCustomer]
```

	CustomerSK	AlternateCustomerCode	Company	Name	Gender	age	Customer_address	startDate	endDate	insertDate	modifiedDate	city	cor
1	1	0	4You	Roy Braun	male	21	557 Northview Lane	2022-05-05 01:43:28.000	NULL	NULL	NULL	Insrom	30
2	2	1	4You	Joseph Holsten	male	37	927 Briar Crest Road	2022-05-05 01:43:28.000	NULL	NULL	NULL	Zhaozhen	45
3	3	10	4You	Melvin Lovejoy	male	36	2679 Algoma Court	2022-05-05 01:43:28.000	NULL	NULL	NULL	Wilga	66
4	4	100	4You	Carla Puskar	female	36	30610 Mosinee Alley	2022-05-05 01:43:28.000	NULL	NULL	NULL	Bato	78
5	5	10000	Acme Factory	Adam Stanley	none	22	7916 Dahle Terrace	2022-05-05 01:43:28.000	NULL	NULL	NULL	Duran	15
6	6	1001	Acme Factory	Josephine Strasser	female	21	058 Moulton Parkway	2022-05-05 01:43:28.000	NULL	NULL	NULL	Tōkamachi	49
7	7	1002	Acme Factory	Nancy Kramer	female	55	2 Manufacturers Point	2022-05-05 01:43:28.000	NULL	NULL	NULL	Sinisian	45
8	8	1003	Acme Factory	Wanda Underwood	female	38	0 Autumn Leaf Plaza	2022-05-05 01:43:28.000	NULL	NULL	NULL	Maswarah	61
9	9	1004	Acme Factory	Carl Saiz	none	58	9 Caliangt Crossing	2022-05-05 01:43:28.000	NULL	NULL	NULL	Haiyan	94
10	10	1005	Acme Factory	Tara Moore	female	54	392 Westport Place	2022-05-05 01:43:28.000	NULL	NULL	NULL	Sātkhira	38

Query executed successfully. | DESKTOP-FLPSHFE (15.0 RTM) | DESKTOP-FLPSHFE\PCView... | Flights_DW | 00:00:00 | 1,340 rows

After updating the StgCustomer table city column from Insrom to Kandy the endDate column and modifiedDate columns were updated accordingly and the updated row is added as a new row so the total row count has increased to 1341. The screenshot below shows the original record that was modified.

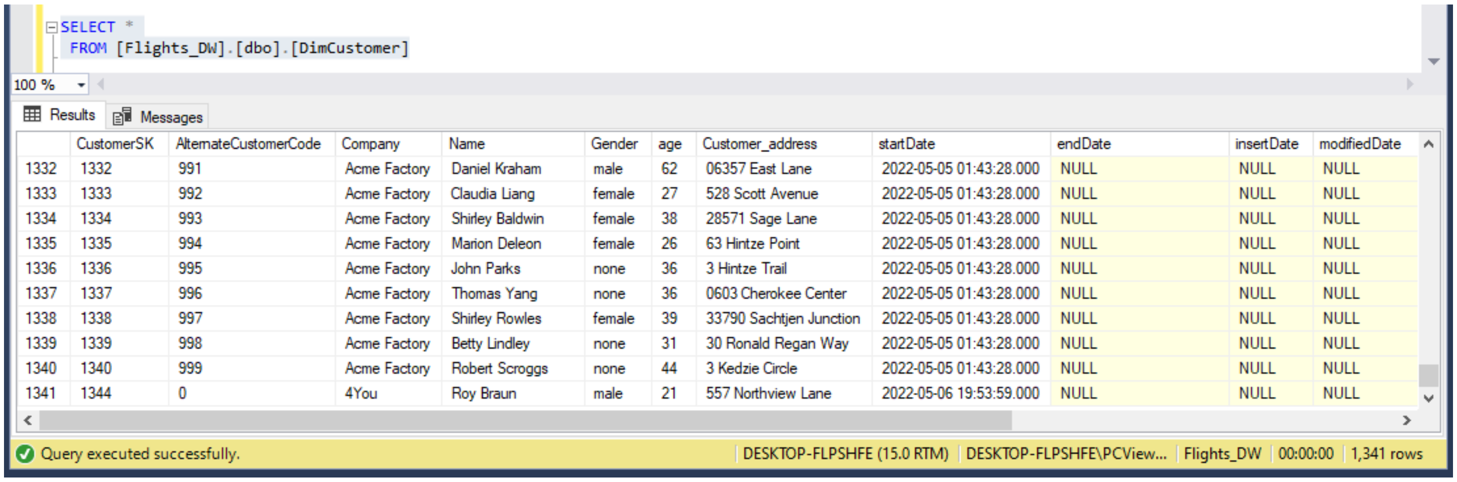


```
SELECT *
  FROM [Flights_DW].[dbo].[DimCustomer]
```

	lternateCustomerCode	Company	Name	Gender	age	Customer_address	startDate	endDate	insertDate	modifiedDate	city
1		4You	Roy Braun	male	21	557 Northview Lane	2022-05-05 01:43:28.000	2022-05-06 19:53:59.000	NULL	2022-05-06 19:54:01.167	Insrom
2		4You	Joseph Holsten	male	37	927 Briar Crest Road	2022-05-05 01:43:28.000	NULL	NULL	NULL	Zhaozh
3	0	4You	Melvin Lovejoy	male	36	2679 Algoma Court	2022-05-05 01:43:28.000	NULL	NULL	NULL	Wilga
4	00	4You	Carla Puskar	female	36	30610 Mosinee Alley	2022-05-05 01:43:28.000	NULL	NULL	NULL	Bato
5	0000	Acme Factory	Adam Stanley	none	22	7916 Dahle Terrace	2022-05-05 01:43:28.000	NULL	NULL	NULL	Duran
6	001	Acme Factory	Josephine Strasser	female	21	058 Moulton Parkway	2022-05-05 01:43:28.000	NULL	NULL	NULL	Tōkam
7	002	Acme Factory	Nancy Kramer	female	55	2 Manufacturers Point	2022-05-05 01:43:28.000	NULL	NULL	NULL	Sinisiar
8	003	Acme Factory	Wanda Underwood	female	38	0 Autumn Leaf Plaza	2022-05-05 01:43:28.000	NULL	NULL	NULL	Maswa
9	004	Acme Factory	Carl Saiz	none	58	9 Caliangt Crossing	2022-05-05 01:43:28.000	NULL	NULL	NULL	Haiyan
10	005	Acme Factory	Tara Moore	female	54	392 Westport Place	2022-05-05 01:43:28.000	NULL	NULL	NULL	Sātkhir

Query executed successfully. | DESKTOP-FLPSHFE (15.0 RTM) | DESKTOP-FLPSHFE\PCView... | Flights_DW | 00:00:00 | 1,341 rows

The screenshot below shows the newly added record at the bottom of the table.



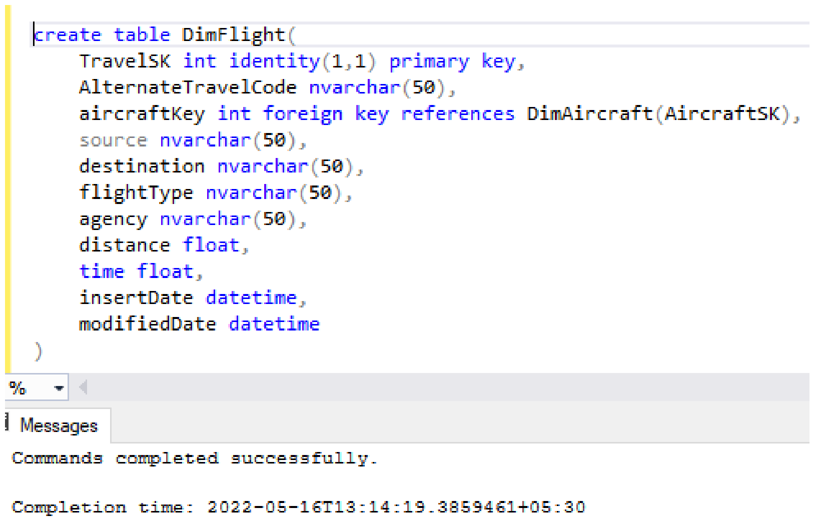
```
SELECT *
  FROM [Flights_DW].[dbo].[DimCustomer]
```

	CustomerSK	AlternateCustomerCode	Company	Name	Gender	age	Customer_address	startDate	endDate	insertDate	modifiedDate
1332	1332	991	Acme Factory	Daniel Kraham	male	62	06357 East Lane	2022-05-05 01:43:28.000	NULL	NULL	NULL
1333	1333	992	Acme Factory	Claudia Liang	female	27	528 Scott Avenue	2022-05-05 01:43:28.000	NULL	NULL	NULL
1334	1334	993	Acme Factory	Shirley Baldwin	female	38	28571 Sage Lane	2022-05-05 01:43:28.000	NULL	NULL	NULL
1335	1335	994	Acme Factory	Marion Deleon	female	26	63 Hintze Point	2022-05-05 01:43:28.000	NULL	NULL	NULL
1336	1336	995	Acme Factory	John Parks	none	36	3 Hintze Trail	2022-05-05 01:43:28.000	NULL	NULL	NULL
1337	1337	996	Acme Factory	Thomas Yang	none	36	0603 Cherokee Center	2022-05-05 01:43:28.000	NULL	NULL	NULL
1338	1338	997	Acme Factory	Shirley Rowles	female	39	33790 Sachtjen Junction	2022-05-05 01:43:28.000	NULL	NULL	NULL
1339	1339	998	Acme Factory	Betty Lindley	none	31	30 Ronald Regan Way	2022-05-05 01:43:28.000	NULL	NULL	NULL
1340	1340	999	Acme Factory	Robert Scroggs	none	44	3 Kedzie Circle	2022-05-05 01:43:28.000	NULL	NULL	NULL
1341	1344	0	4You	Roy Braun	male	21	557 Northview Lane	2022-05-06 19:53:59.000	NULL	NULL	NULL

Query executed successfully. | DESKTOP-FLPSHFE (15.0 RTM) | DESKTOP-FLPSHFE\PCView... | Flights_DW | 00:00:00 | 1,341 rows

Transforming and loading data to flight dimension

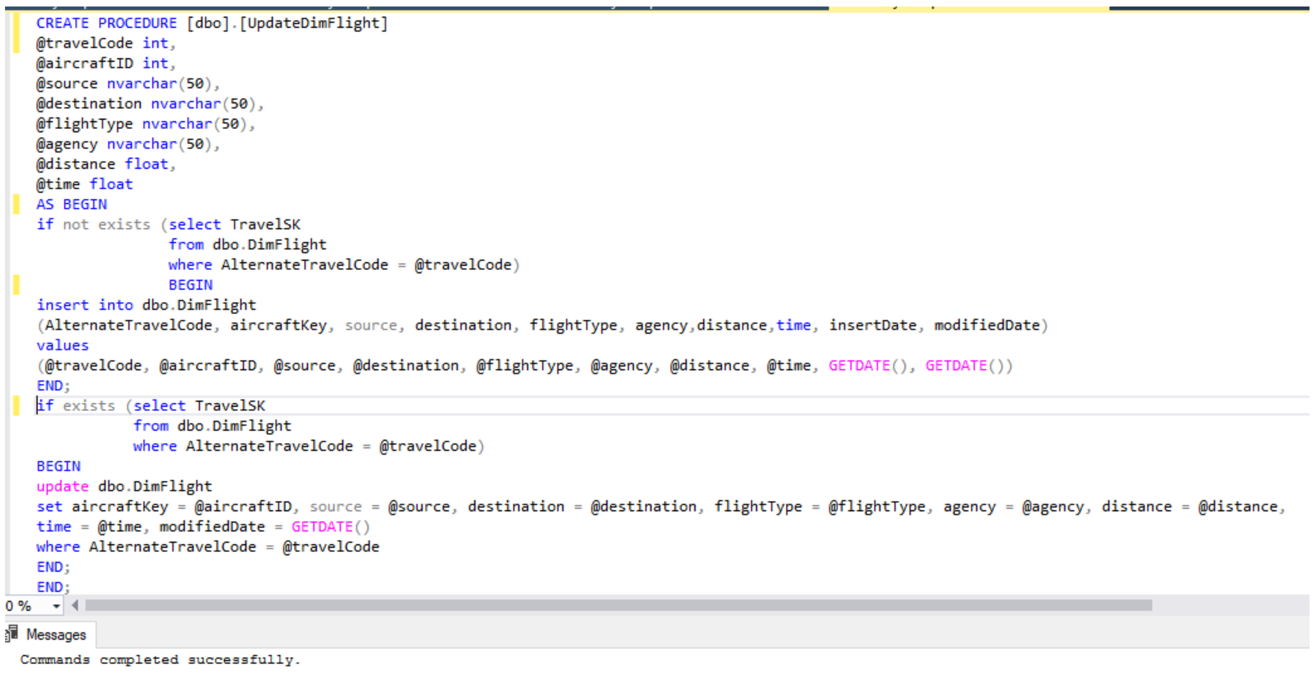
The query used to create the flight dimension is shown below.



```
create table DimFlight(
    TravelSK int identity(1,1) primary key,
    AlternateTravelCode nvarchar(50),
    aircraftKey int foreign key references DimAircraft(AircraftSK),
    source nvarchar(50),
    destination nvarchar(50),
    flightType nvarchar(50),
    agency nvarchar(50),
    distance float,
    time float,
    insertDate datetime,
    modifiedDate datetime
)
```

```
Commands completed successfully.

Completion time: 2022-05-16T13:14:19.3859461+05:30
```

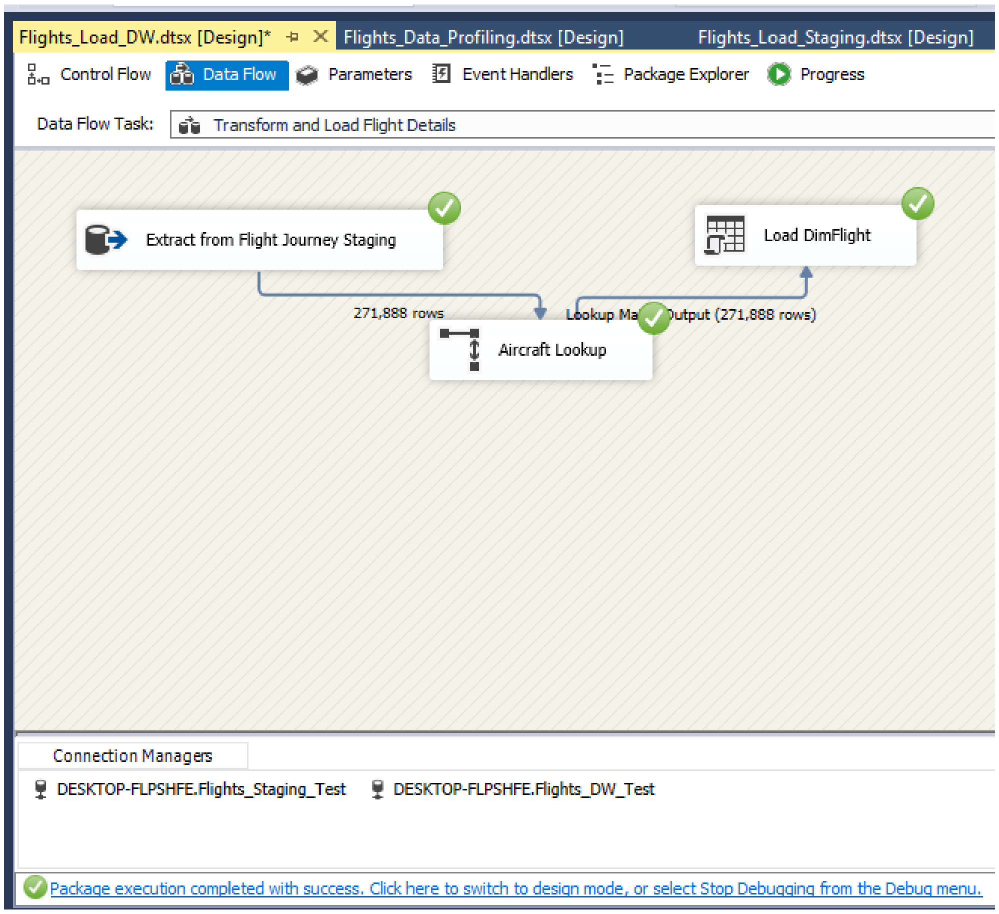
The query used to update the flight dimension is shown below.



```
CREATE PROCEDURE [dbo].[UpdateDimFlight]
@travelCode int,
@aircraftID int,
@source nvarchar(50),
@destination nvarchar(50),
@flightType nvarchar(50),
@agency nvarchar(50),
@distance float,
@time float
AS BEGIN
if not exists (select TravelSK
               from dbo.DimFlight
               where AlternateTravelCode = @travelCode)
               BEGIN
insert into dbo.DimFlight
(AlternateTravelCode, aircraftKey, source, destination, flightType, agency,distance,time, insertDate, modifiedDate)
values
(@travelCode, @aircraftID, @source, @destination, @flightType, @agency, @distance, @time, GETDATE(), GETDATE())
END;
if exists (select TravelSK
           from dbo.DimFlight
           where AlternateTravelCode = @travelCode)
BEGIN
update dbo.DimFlight
set aircraftKey = @aircraftID, source = @source, destination = @destination, flightType = @flightType, agency = @agency, distance = @distance,
time = @time, modifiedDate = GETDATE()
where AlternateTravelCode = @travelCode
END;
END;
```

```
Commands completed successfully.
```

Data was extracted from flight staging table and customer dimension. The flight staging data was sorted using the userCode while the customer dimension was sorted using the AlternateCustomerCode. After the data was sorted data from both tables were joined using a merge join and from the merge join the data is loaded to the flight dimension. The process was executed successfully as shown below.


Flights_Load_DW.dtsx [Design]*
Flights_Data_Profiling.dtsx [Design]
Flights_Load_Staging.dtsx [Design]
Control Flow
Data Flow
Parameters
Event Handlers
Package Explorer
Progress
Data Flow Task:
Transform and Load Flight Details
Extract from Flight Journey Staging
271,888 rows
Aircraft Lookup
Lookup Ma Output (271,888 rows)
Load DimFlight
Connection Managers
DESKTOP-FLPSHFE.Flights_Staging_Test
DESKTOP-FLPSHFE.Flights_DW_Test

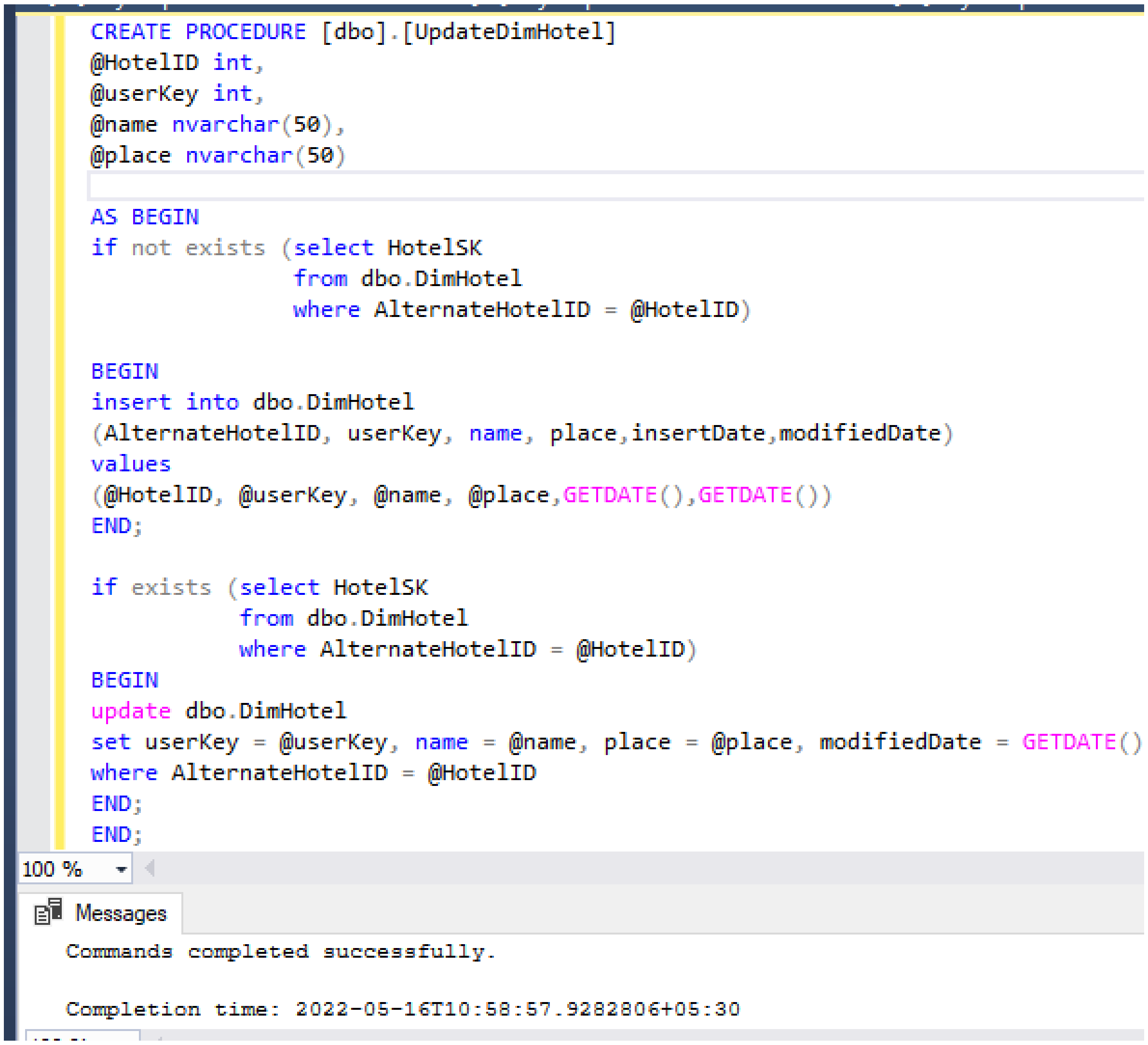
Package execution completed with success. Click here to switch to design mode, or select Stop Debugging from the Debug menu.

Transforming and loading data into Hotel dimension

The query used to create the hotel dimension is shown below

```
create table DimHotel(
    HotelSK int identity(1,1) primary key,
    AlternateHotelID int,
    userKey int foreign key references DimCustomer(CustomerSK),
    name nvarchar(50),
    place nvarchar(50),
    insertDate datetime,
    modifiedDate datetime
)
```

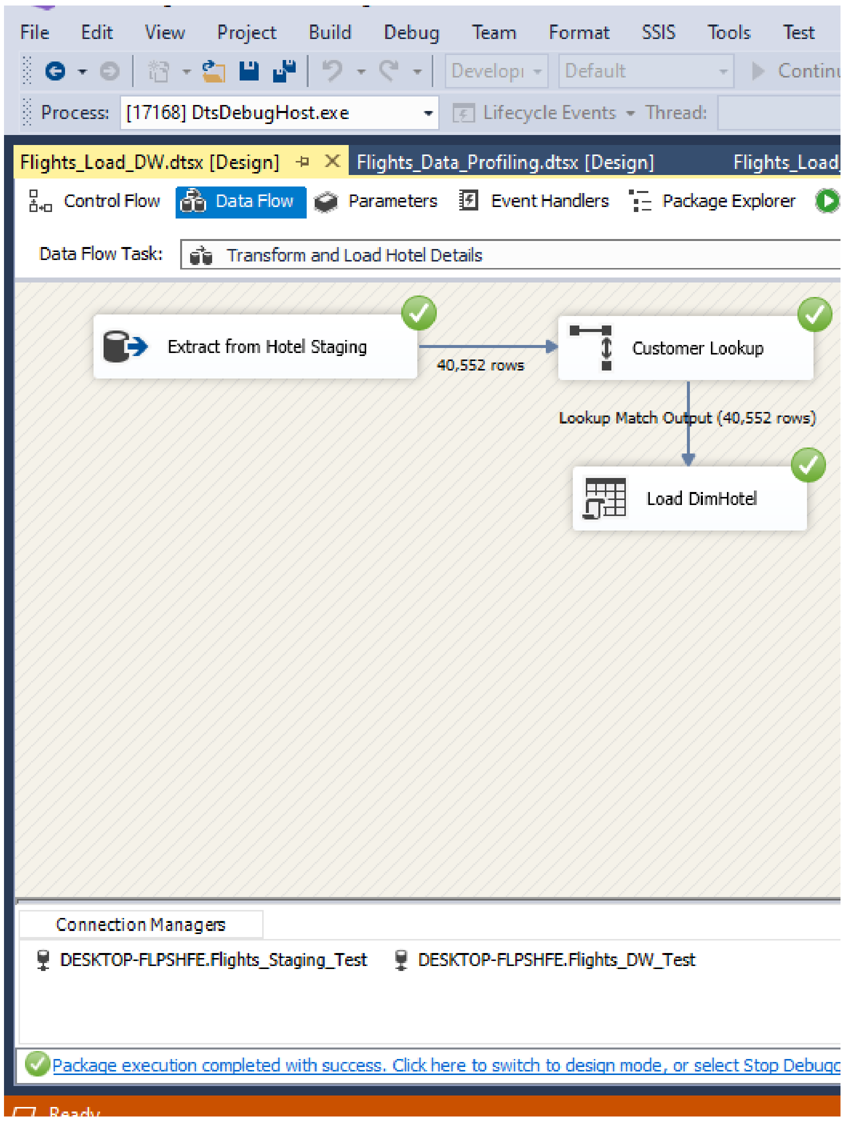
The query used to update the hotel dimension is shown below.



```
CREATE PROCEDURE [dbo].[UpdateDimHotel]
@HotelID int,
@userKey int,
@name nvarchar(50),
@place nvarchar(50)

AS BEGIN
if not exists (select HotelSK
               from dbo.DimHotel
               where AlternateHotelID = @HotelID)

BEGIN
insert into dbo.DimHotel
(AlternateHotelID, userKey, name, place,insertDate,modifiedDate)
values
(@HotelID, @userKey, @name, @place,GETDATE(),GETDATE())
END;

if exists (select HotelSK
           from dbo.DimHotel
           where AlternateHotelID = @HotelID)
BEGIN
update dbo.DimHotel
set userKey = @userKey, name = @name, place = @place, modifiedDate = GETDATE()
where AlternateHotelID = @HotelID
END;
END;
```

```
Commands completed successfully.

Completion time: 2022-05-16T10:58:57.9282806+05:30
```

The process was executed successfully as shown below. Data from hotel staging table was sorted according to userCode. Next data was extracted from customer dimension and sorted according to the AlternateCustomerCode. After that data was extracted from flight dimension and sorted according to the userKey. After sorting the details from the dimension table were joined using a merge join and after that again the details were sorted by the CustomerSK. Finally all the sorted data was merged using a merge join and the data was passed into the DimHotel dimension in the data warehouse.


File
Edit
View
Project
Build
Debug
Team
Format
SSIS
Tools
Test
Develop
Default
Continu
Process: [17168] DtsDebugHost.exe
Lifecycle Events
Thread:
Flights_Load_DW.dtsx [Design]
Flights_Data_Profiling.dtsx [Design]
Flights_Load
Control Flow
Data Flow
Parameters
Event Handlers
Package Explorer
Data Flow Task: Transform and Load Hotel Details
Extract from Hotel Staging
40,552 rows
Customer Lookup
Lookup Match Output (40,552 rows)
Load DimHotel
Connection Managers
DESKTOP-FLPSHFE.Flights_Staging_Test
DESKTOP-FLPSHFE.Flights_DW_Test
Ready

Package execution completed with success. Click here to switch to design mode, or select Stop Debugg

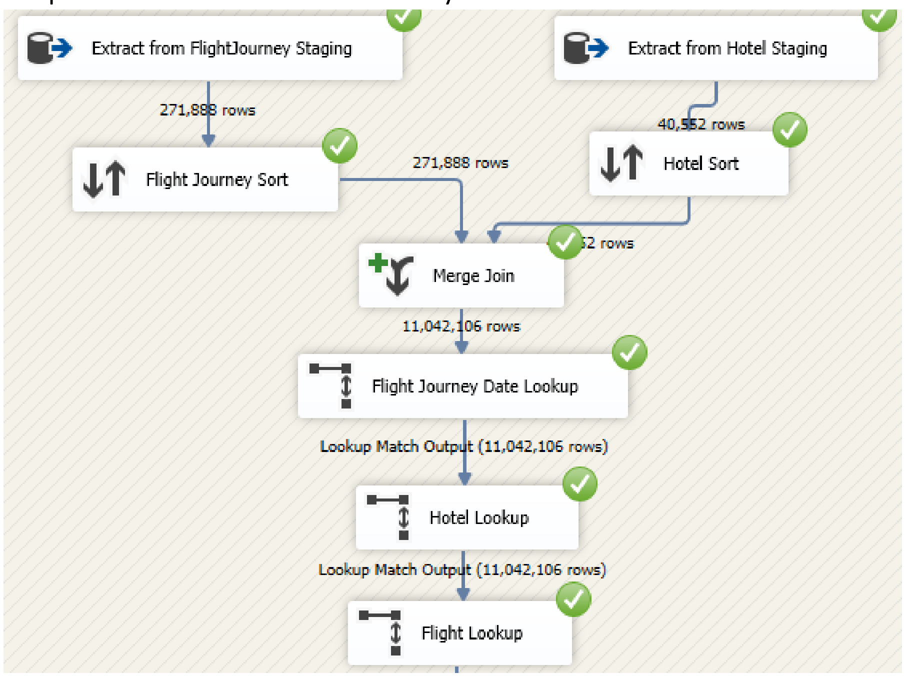
Now all the dimensions are loaded with the data and the next step will be loading the data into the fact table.

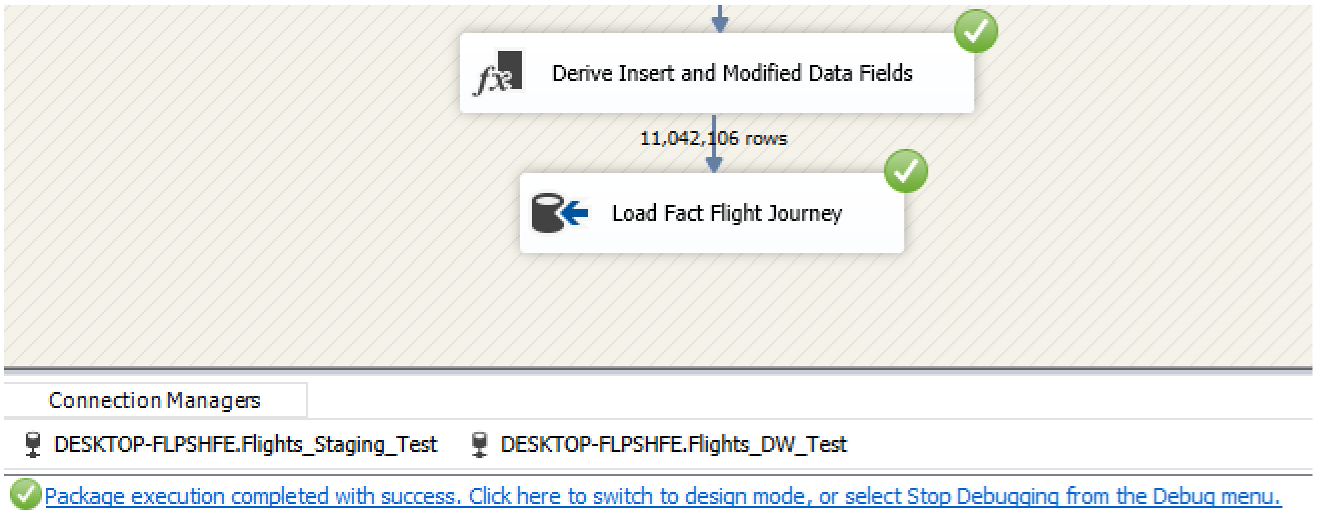
Transforming and loading data into FactFlightJourney

The following steps were followed while loading data into the fact flight journey fact table:

1. Data was extracted from the flight journey staging table.
2. Data was sorted according to the userCode.
3. Data was extracted from the hotel staging table.
4. The data was sorted according to the userCode.
5. After sorting data from flight journey staging table and hotel staging table; the tables were joined using a merge join. In the merge join inner join was used to join the tables. userCode from both the tables were used as the joining key.
6. Next the data was passed into a lookup where it extracts data from the date dimension and was used to find the relevant details.
7. Next the data was passed into another lookup where it extracts the data from the hotel dimension and this is used to obtain the hotelSK.
8. After that step the data was passed into another lookup where it extracts data from the flight dimension and this is used to obtain the travelSK.
9. Next the data was passed into a derived column function to obtain the insertDate and the modifiedDate.
10. After all these steps the data is finally loaded into the FactFlightJourney fact table in the data warehouse.

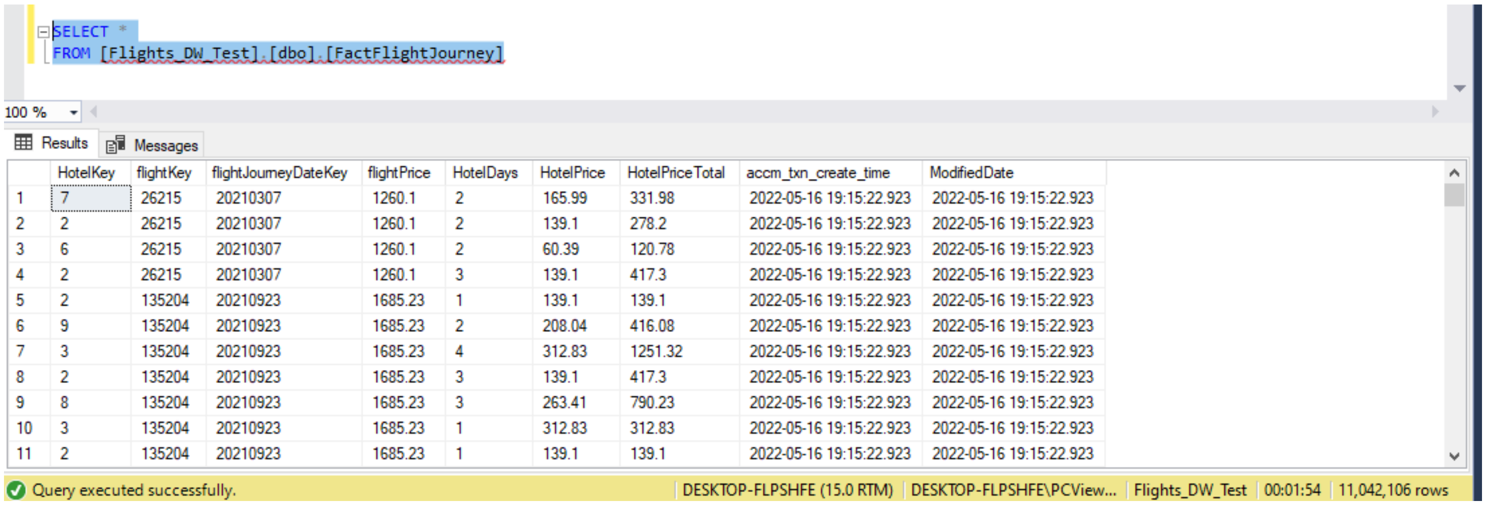
The process was executed successfully as shown in the screenshot below.


Extract from FlightJourney Staging
Extract from Hotel Staging
271,888 rows
40,552 rows
Flight Journey Sort
271,888 rows
Hotel Sort
2 rows
Merge Join
11,042,106 rows
Flight Journey Date Lookup
Lookup Match Output (11,042,106 rows)
Hotel Lookup
Lookup Match Output (11,042,106 rows)
Flight Lookup


Derive Insert and Modified Data Fields
11,042,106 rows
Load Fact Flight Journey
Connection Managers
DESKTOP-FLPSHFE.Flights_Staging_Test
DESKTOP-FLPSHFE.Flights_DW_Test

Package execution completed with success. Click here to switch to design mode, or select Stop Debugging from the Debug menu.

The screenshot of the fact table is shown below:



```
SELECT *
FROM [Flights_DW_Test].[dbo].[FactFlightJourney]
```

100 %

Results | Messages

	HotelKey	flightKey	flightJourneyDateKey	flightPrice	HotelDays	HotelPrice	HotelPriceTotal	accm_txn_create_time	ModifiedDate
1	7	26215	20210307	1260.1	2	165.99	331.98	2022-05-16 19:15:22.923	2022-05-16 19:15:22.923
2	2	26215	20210307	1260.1	2	139.1	278.2	2022-05-16 19:15:22.923	2022-05-16 19:15:22.923
3	6	26215	20210307	1260.1	2	60.39	120.78	2022-05-16 19:15:22.923	2022-05-16 19:15:22.923
4	2	26215	20210307	1260.1	3	139.1	417.3	2022-05-16 19:15:22.923	2022-05-16 19:15:22.923
5	2	135204	20210923	1685.23	1	139.1	139.1	2022-05-16 19:15:22.923	2022-05-16 19:15:22.923
6	9	135204	20210923	1685.23	2	208.04	416.08	2022-05-16 19:15:22.923	2022-05-16 19:15:22.923
7	3	135204	20210923	1685.23	4	312.83	1251.32	2022-05-16 19:15:22.923	2022-05-16 19:15:22.923
8	2	135204	20210923	1685.23	3	139.1	417.3	2022-05-16 19:15:22.923	2022-05-16 19:15:22.923
9	8	135204	20210923	1685.23	3	263.41	790.23	2022-05-16 19:15:22.923	2022-05-16 19:15:22.923
10	3	135204	20210923	1685.23	1	312.83	312.83	2022-05-16 19:15:22.923	2022-05-16 19:15:22.923
11	2	135204	20210923	1685.23	1	139.1	139.1	2022-05-16 19:15:22.923	2022-05-16 19:15:22.923

Query executed successfully. | DESKTOP-FLPSHFE (15.0 RTM) | DESKTOP-FLPSHFE\PCView... | Flights_DW_Test | 00:01:54 | 11,042,106 rows

The name of the source database is Flights_SourceDB. The name of the staging database is Flights_Staging_Test. The name of the data warehouse is Flights_DW_Test.

Accumulating fact table

The data set that was prepared for the accumulating fact table is in the FlightSourceDB AccumulatingFacts table.