ASSIGNMENT 1

Data Warehouse and Business Intelligence

SUBMITTED BY:
NUSHRATH AAISHA MAUROOF
IT20202354

Contents

INTRODUCTION

This is a project based on SQL Server Integration Services and has been completed using Visual Studio and Microsoft SQL Server Management Studio. The purpose of the project is to create an accurate and meaningful data warehouse using various data sets and by following appropriate ETL procedures.

DATA SELECTION

The basic data set that was selected for this project is derived from the OpenFlights database which contains information about the global aviation industry. It is a collection information about a large number airports, airlines and aircrafts. Additionally, details regarding the pilots associated with the flights and the fuel types that have been used for each flight are also included.

Details of a particular flight include the airline it is associated to, the source and destination airports, the type of aircraft, the pilots navigating it and the type of fuel that has been used for the journey. Additionally, it also includes other details such as the various operational expenses, total number of passengers and calculations based on the expenses of each flight.

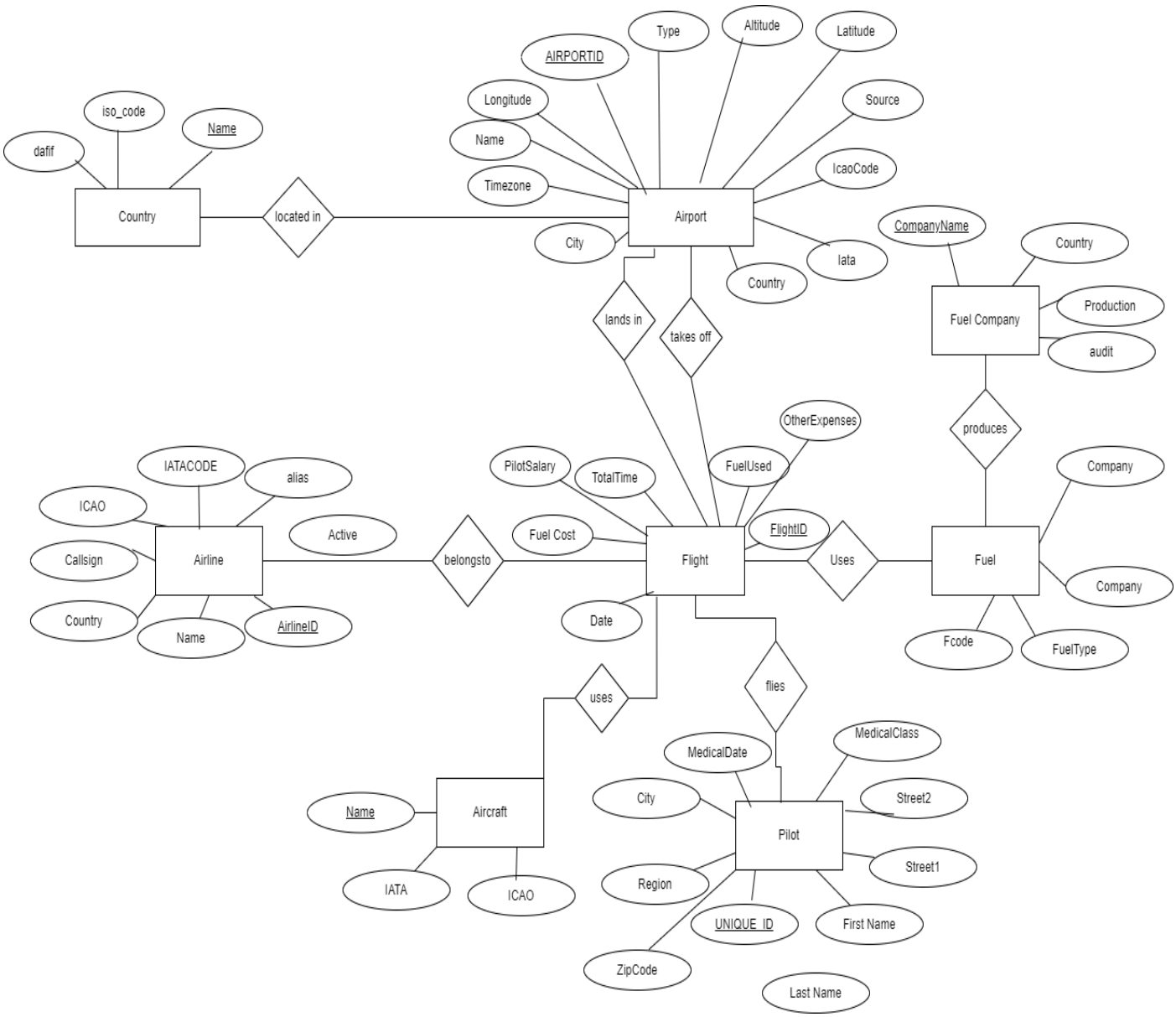
Detailed description about the attributes in all the tables that form the data set are given below.

Table Name	Table Description	Attribute and Attribute Description	
Airport	Incudes information of 7698 airports around the world	**Airport ID**	Unique OpenFlights identifier for this airport.
		Name	Name of airport.
		City	Main city served by airport.
		Country	Country or territory where airport is located.
		IATA	3-letter IATA code. Null if not assigned/unknown.
		ICAO	4-letter ICAO code. Null if not assigned.
		Latitude	Decimal degrees, usually to six significant digits. Negative is South, positive is North.
		Longitude	Decimal degrees, usually to six significant digits. Negative is West, positive is East.
		Altitude	In feet.
		Timezone	Hours offset from UTC. Fractional hours are expressed as decimals
		DST	Daylight savings time. One of E (Europe), A (US/Canada), S (South America), O (Australia), Z (New Zealand), N (None) or U (Unknown)
		Tz database time zone	Timezone in "tz" (Olson) format
		Type	Type of the airport. Value "airport" for air terminals.
		Source	Source of this data. "OurAirports" for data sourced from OurAirports, .
Airline	Includes Information of 6161 Airlines	**Airline ID**	Unique OpenFlights identifier for this airline.
		Name	Name of the airline.
		Alias	Alias of the airline. For example, All Nippon Airways is commonly known as "ANA".

		IATA 2-letter IATA code, if available. **ICAO** 3-letter ICAO code, if available. **Callsign** Airline callsign. **Country** Country or territory where airport is located **Active** "Y" if the airline is or has until recently been operational, "N" if it is defunct.
Aircraft	Includes details about 245 aircrafts that are operational currently.	**Name** Full name of the aircraft. **IATA code** Unique three-letter IATA identifier for the aircraft. **ICAO code** Unique four-letter ICAO identifier for the aircraft.
Pilot	Details of the over 15000 pilots who navigate the flights	**UNIQUE_ID** Unique Identifier for each pilot **First Name** First and Middle Name. **Last Name** Last Name of Pilot. **Street 1** Part of Address **Street 2** Part of Address **City** Part of Address **State** State Name. Blank if not from the US **ZipCode** **Country** **Region** **Medical Class** Takes values 1,2 or 3 for first second and third class **Medical Date** **Medical Expiry Dare** **MonthlyWorkHours**
Fuel	Details of Aviation fuels used for a particular flight	**FCode** Unique Identifier for each fuel type **Fuel_Type** One of the four types of aviation fuels used to fuel aircrafts. **Company** Production company of the fuel. This refers to fuel company table. **Octane** Octane level of the fuel.
Fuel Company	Details of the Company providing fuel for flights	
Country	This table contains details relevant to all countries.	**name** Full name of the country or territory. **iso_code** Unique two-letter ISO 3166-1 code for the country or territory. **dafif_code** FIPS country codes as used in DAFIF. Obsolete and primarily of historical interested.
Flight Details	Details of one flight. Will be considered the fact table.	**FlightID** Unique Identifier for each flight **Airline** Airline ID that flight belongs to. Refers to airline table. **DepartureAirport** Airport ID of source airport.Refers to airport table **Destination Airport** Airport ID of destination. Refers to airport table. **Aircraft** Plane used for the flight. Refers to aircraft table.

		FlightDate	Date of the flight.
		FuelType	Fuel used for the flight. Refers to Fuel table.
		FuelGallonsUsed	Nunber of fuel gallons used for the flight.
		GallonProce	Price of each Gallon
		Pilot1	UniqueID of pilot(captain). Refers to pilot table.
		Pilot1Salary	Salary per hour for pilot 1.
		Pilot2	UniqueID of pilot2. Refers to pilot table.
		Pilot1Salary	Salary per hour for pilot 2.
		NoOfHours	Total Time of journey
		OtherExpensesHour	Other Expenses per flightHour.
		NoOfPassenger	Number of passengers in the flight.

ER DIAGRAM

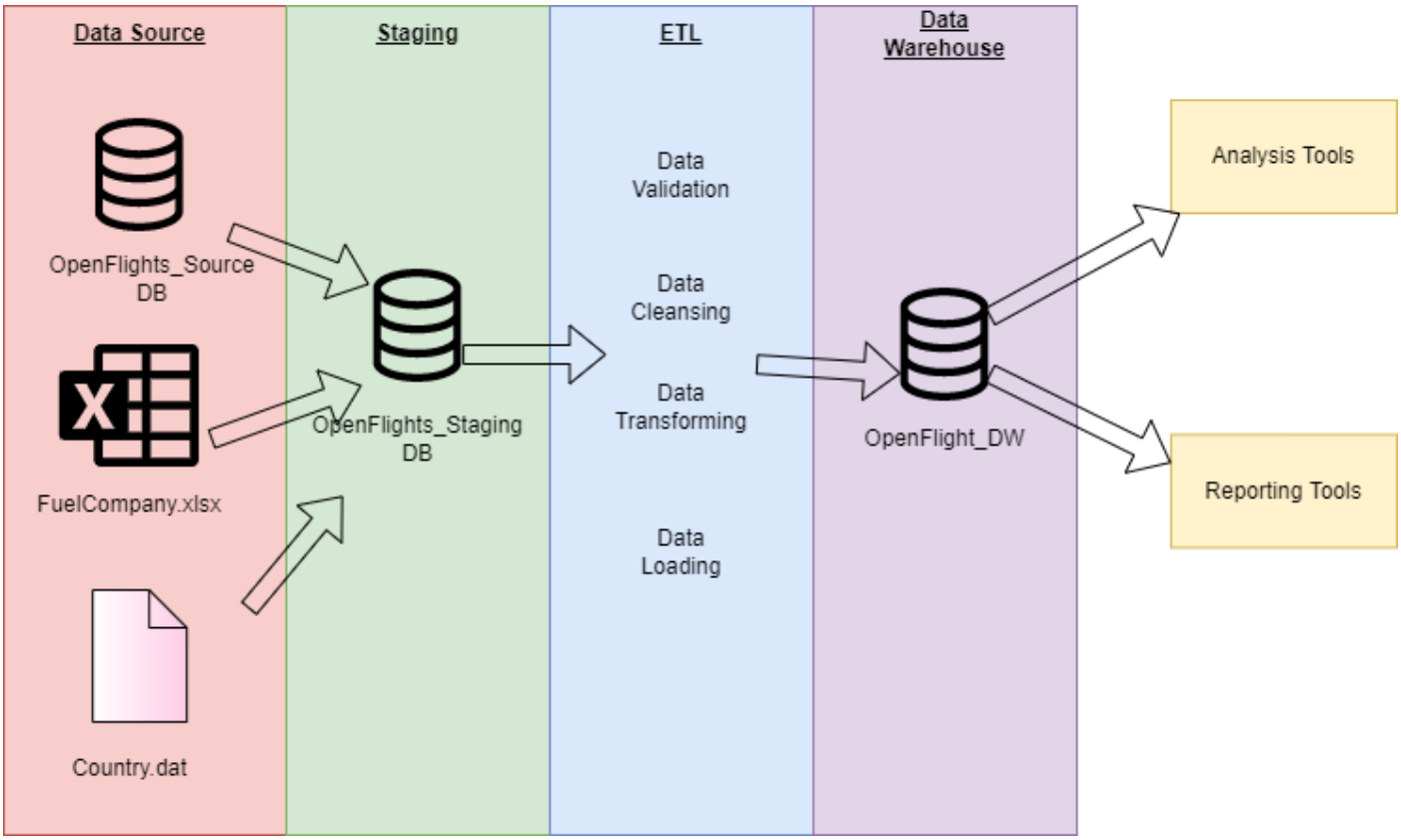


DATA SOURCES

The above data was derived in the form of Comma Separated Values(CSV) files and converted to flat files(.dat), Excel worksheet files(xlsx) and others were imported into the source database. Details of the data source of each table that was used for staging is given below.

- <u>From Source Database</u>
    1. Airport
    2. Airline
    3. Aircraft
    4. Fuel
    5. Pilot
    6. Flight Details
- <u>From Flat Files(.dat)</u>
    1. Countries
- <u>From Excel Worsheet(xlsx)</u>
    1. FuelCompany

SOLUTION ARCHITECTURE

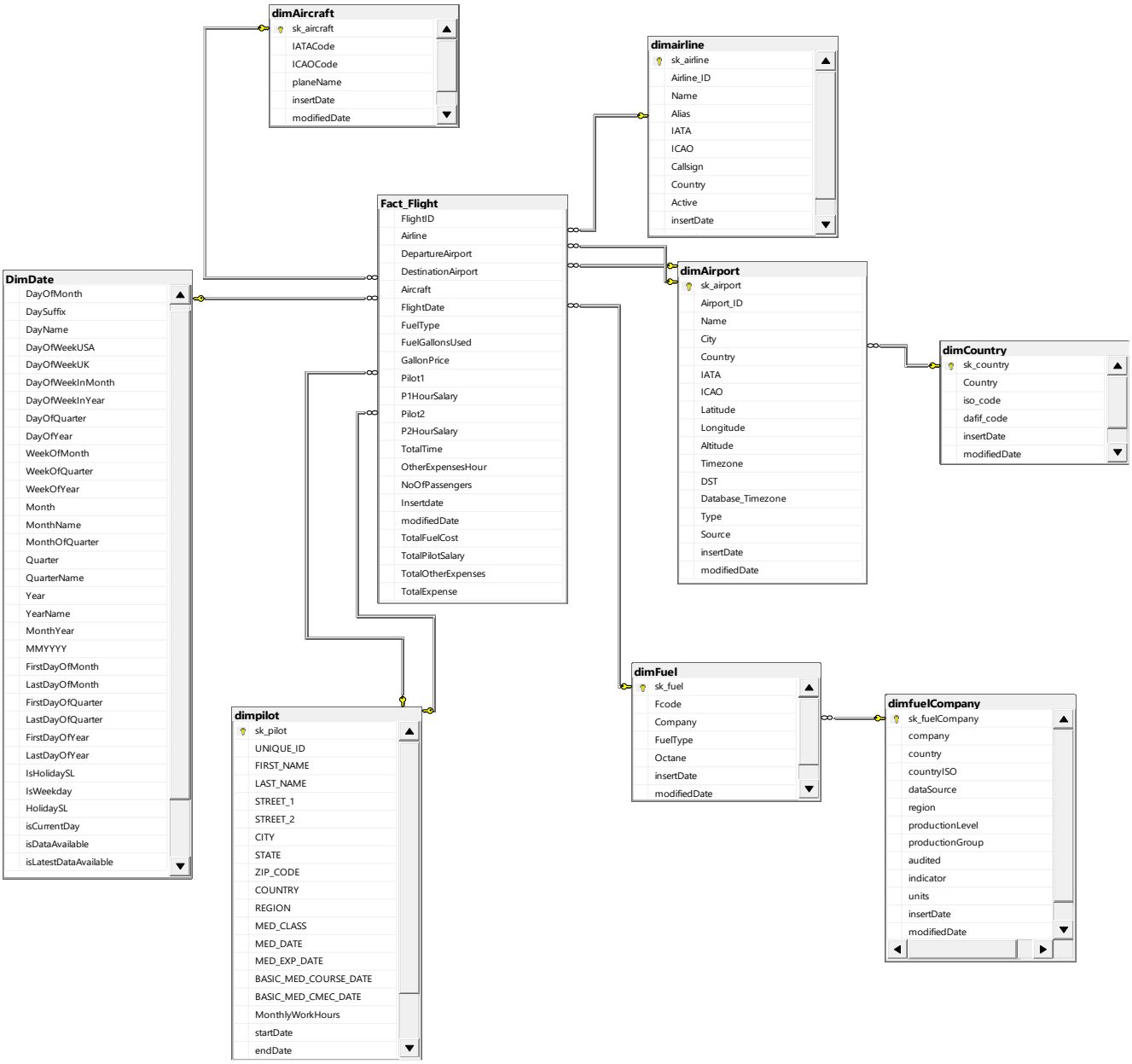


1. Data Source: Data is derived from various sources such as database, excel file and flat file.
2. Staging: Data is extracted from sources and loaded into the staging database.
3. ETL processes are implemented on the data extracted from the staging database.
4. Data is loaded into the Data warehouse.

5. Data from the warehouse can be used for analysis and create reports

DATA WAREHOUSE DESIGN AND DEVELOPMENT

RELATIONAL DIAGRAM



- The data warehouse is designed using the SNOWFLAKE SCHEMA.
- There is one fact table, six dimensions and two inherited dimensions.
- The inherited tables are as follow;
  1. DimAirport-> DimCountry
  2. DimFuel->DimFuelCompany
- The assumption that the pilot table is a slowly changing dimension is made.

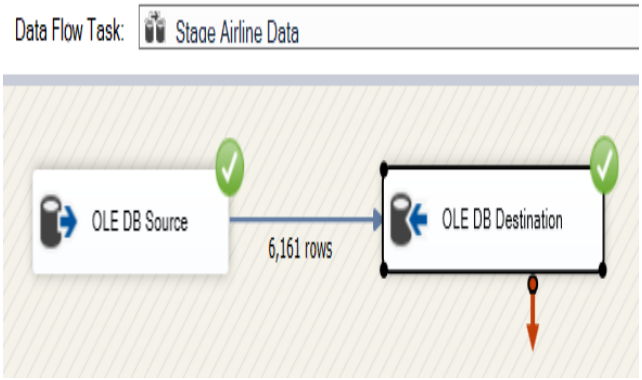
ETL DEVELOPMENT

1. Extract From Data Sources into Staging Tables

The first step in the ETL process is to extract data from the initial data sources to the staging tables in the staging database. For each extraction from the source to the staging table, data flow tasks were used. A truncate table operation is conducted before executing the extraction and loading into the staging table to ensure that records are not duplicated.

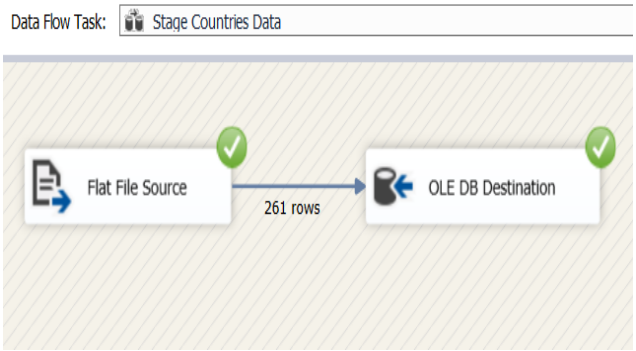
All data flow tasks used to load source data into the staging table are connected as shown in the following image.

- <u>Staging Airline Data</u>



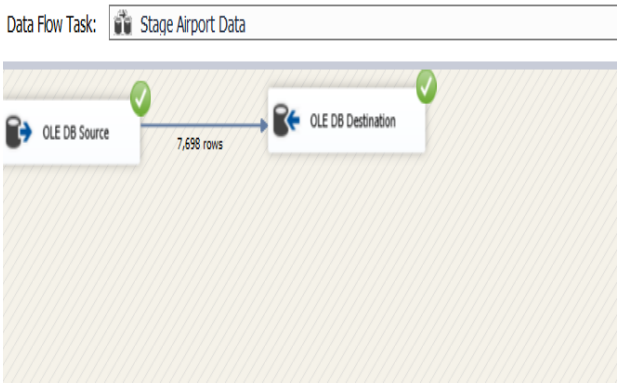
- Data is extracted from the Airline table in the OpenFlight_Source database.
- Data is loaded into the Airline table in the OpenFlights_Staging database.

- Staging Country Data



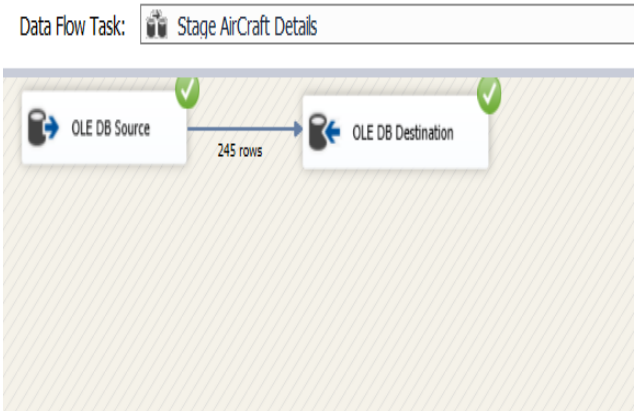
- Data is extracted from a flat file source.
- Data is loaded into the Country table in the OpenFlights_Staging database.

- Staging Airport Data



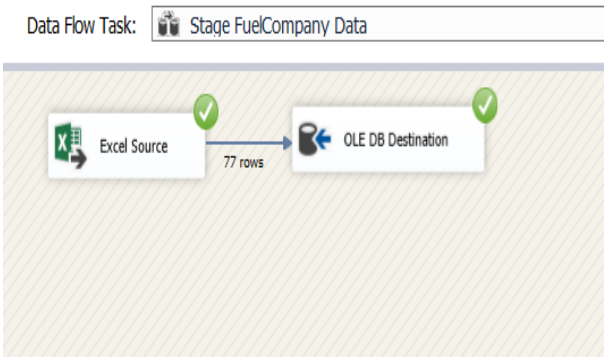
- Data is extracted from from the Airport table in the OpenFlight_Source database
- Data is loaded into the Airport table in the OpenFlights_Staging database.

- Staging Aircraft Data



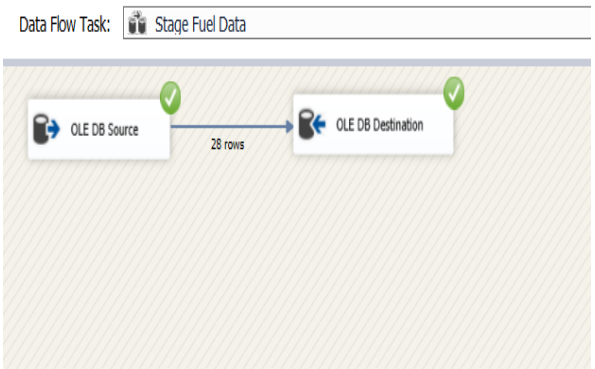
- Data is extracted from from the Aircraft table in the OpenFlight_Source database
- Data is loaded into the Aircraft table in the OpenFlights_Staging database.

- Staging Fuel Company Data



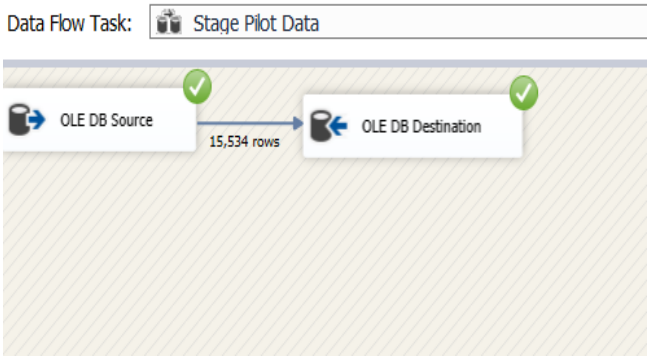
- Data is extracted from an Excel Worksheet file(.xlsx)
- Data is loaded into the Fuel Company table in the OpenFlights_Staging database.

- Staging Fuel Data



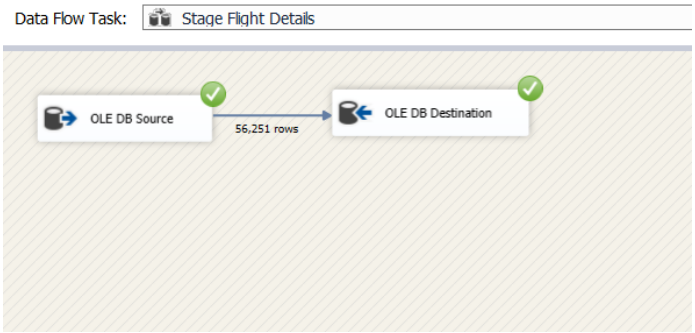
- Data is extracted from the Fuel table in the OpenFlight_Source database
- Data is loaded into the Fuel table in the OpenFlights_Staging database.

- Staging Pilot Data



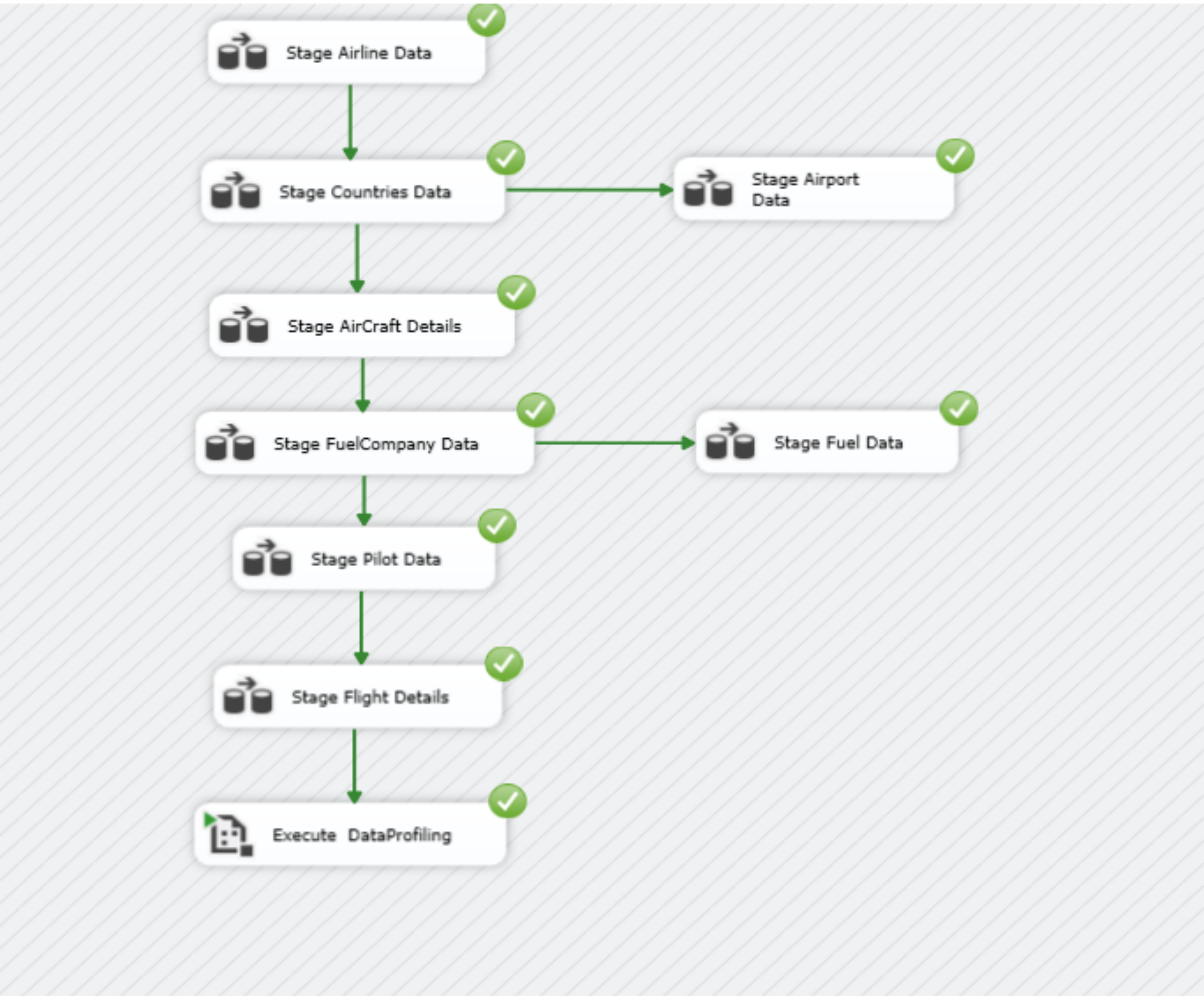
- Data is extracted from the Pilot table in the OpenFlight_Source database
- Data is loaded into the Pilot table in the OpenFlights_Staging database.

- Staging Flight Details



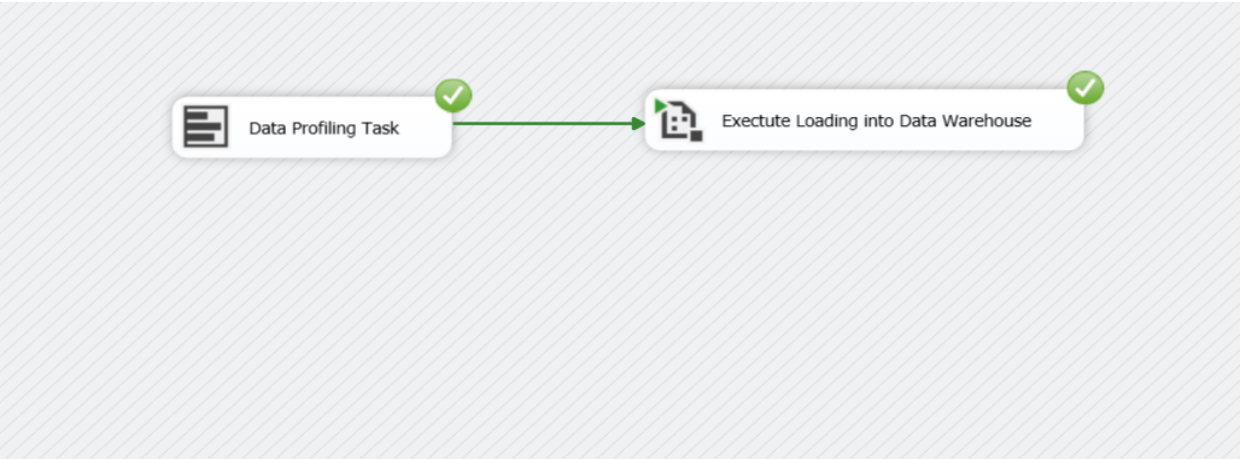
- Data is extracted from the Flight Details table in the OpenFlight_Source database
- Data is loaded into the Flight Details table in the OpenFlights_Staging database.

Given below is a snapshot to show the completion of Staging data.



Once, staging is completed, the data profiling package is implemented.

2. Data Profiling

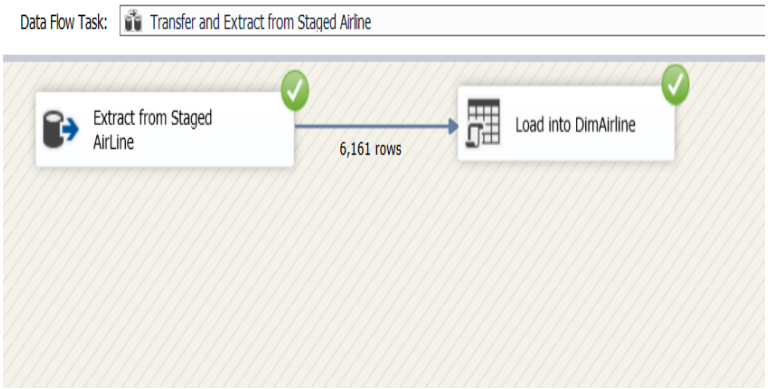


Data from the staging database is profiled and saved in an external location.

The process of loading data into the warehouse will be implemented next.

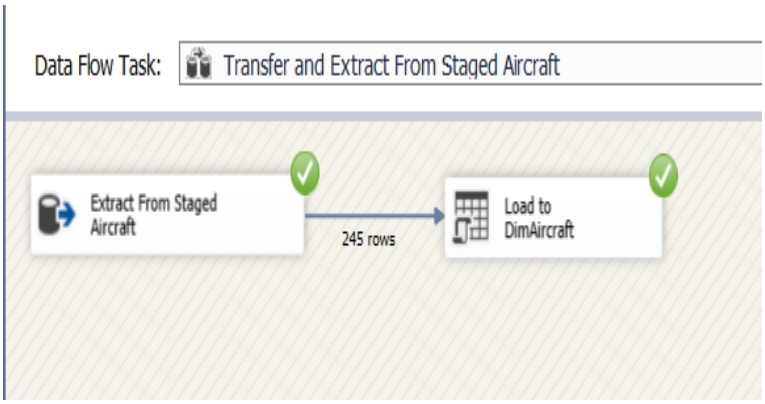
3. Transform and Load Into Data Warehouse

1. Load into Airline Dimension(Dimension Table)



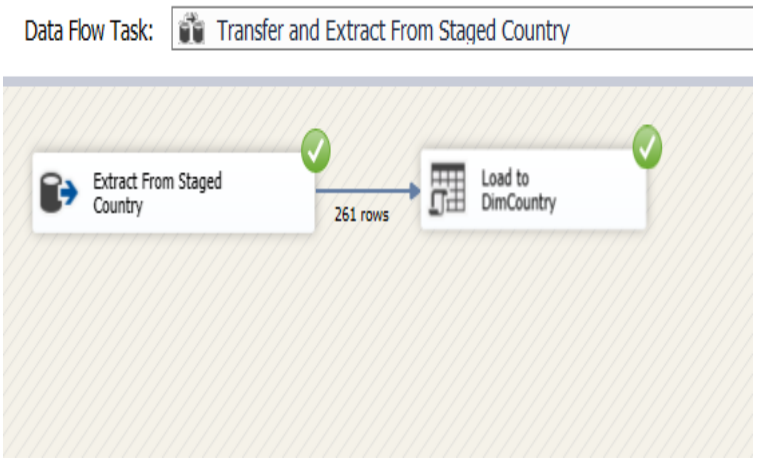
- Data is extracted from Airline tablein the OpenFlight_Staging database
- Data is loaded into DimAirline dimension OpenFlight_DW database.

2. Load into Aircraft Dimension(Dimension Table)



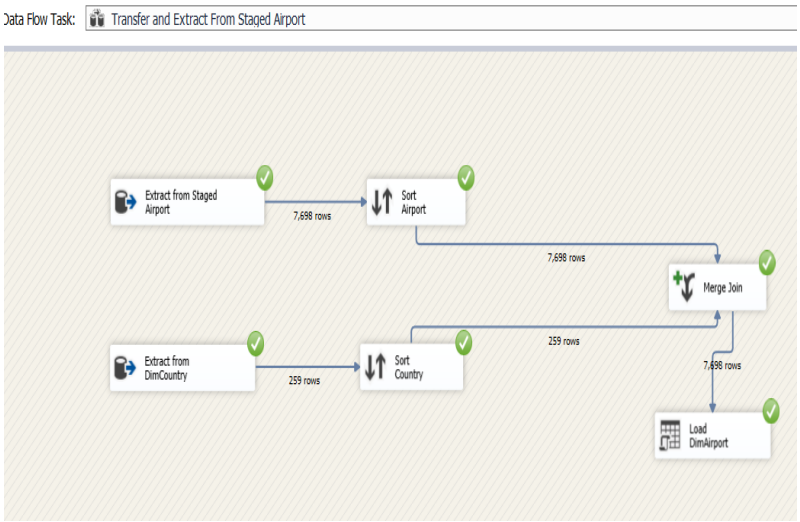
- Data is extracted from Aircraft table in the OpenFlight_Staging database
- Data is loaded into DimAircraft dimension OpenFlight_DW database.

3. Load into Country Dimension(Inherited Dimension Table)



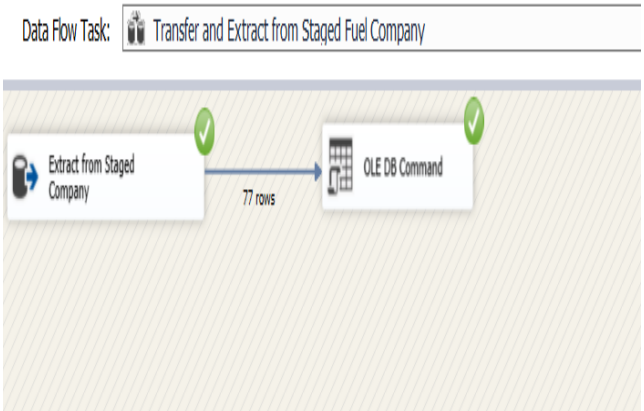
- Data is extracted from Country table in the OpenFlight_Staging database
- Data is loaded into DimCountry dimension OpenFlight_DW database.

4. Load into Airport Dimension (Dimension Table refer to Country Dimension)



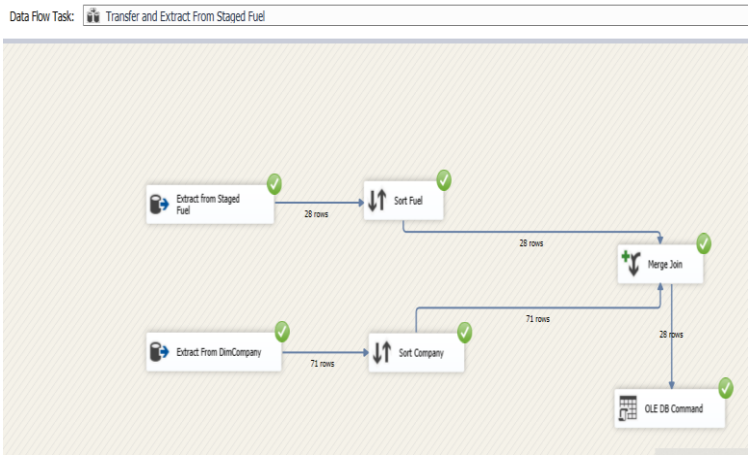
- Data is extracted from Airport table in the OpenFlight_Staging database and then sorted by country.
- Data is extracted from DimCountry and sorted by Country SK.
- Both tables are merged.
- Data is loaded into the DimAirport table in the OpenFlight_DW database.

5. Load into Company Dimension(Inherited Dimension Table)



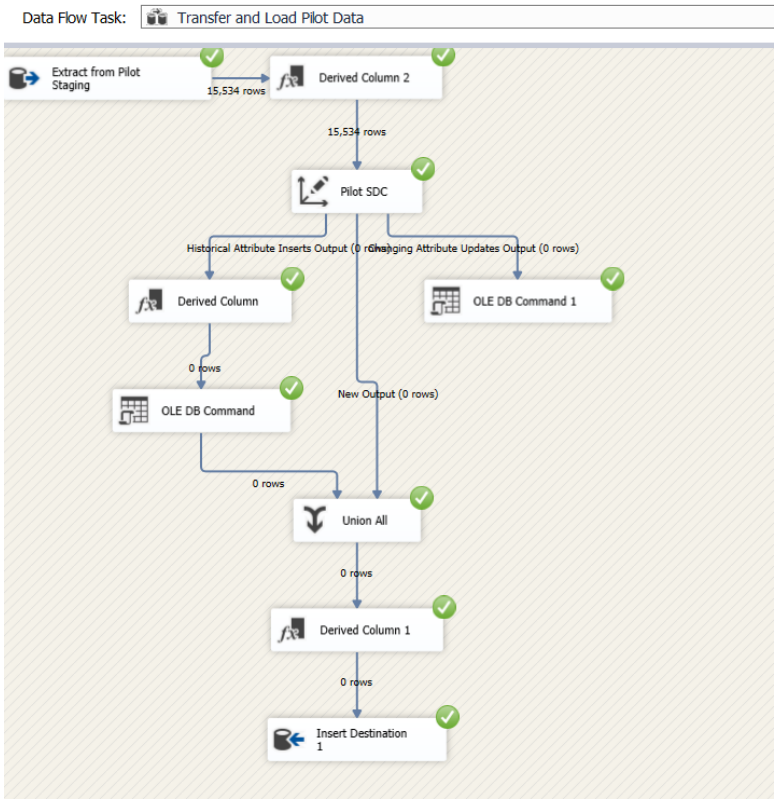
- Data is extracted from Fuel Company tablein the OpenFlight_Staging database
- Data is loaded into DimCompany dimension OpenFlight_DW database.

6. Load into Fuel Dimension(Dimension Table)



- Data is extracted from Fuel table in the OpenFlight_Staging database and then sorted by Company.
- Data is extracted from DimCompany and sorted by Company SK.
- Both tables are merged.
- Data is loaded into the DimCompany table in the OpenFlight_DW database.

7. Load into Pilot Dimension(Slowly Changing Dimension)



- Pilot table is considered a slowly changing dimension.
- Accordingly the following attributes are defined as historical attributes,
  - ✓ Street 1,Street 2,City,State,ZipCode,Country,Region,Medical Class ,Medical Date,Medical Expiry Date
- Monthly Work Hours is considered a changing attribute.
- Data is extracted from Pilot staging table in the Staging database.
- A derived column is used to assign values to insert date and modified date.
- The dimension is made into a slowly changing dimension and implemented accordingly.
- Data is loaded into the DimPilot dimension in the Data warehouse database.

<u>THE PROCEDURES GIVEN BELOW ARE USED FOR THE ACCURATE UPDATE OF ALL THE ABOVE DIMENSIONS</u>

```
--Procedure to update airline dw
table
CREATE PROCEDURE
dbo.UpdateDimAircraft
@iata nvarchar(50),
@icao nvarchar(50),
@pname nvarchar(50)
AS
BEGIN
if not exists (select sk_aircraft
from dbo.DimAircraft
where planeName = @pname)
BEGIN
insert into dbo.DimAircraft
(iatacode, icaocode, planeName,
InsertDate, ModifiedDate)
values
```

```
--Update Procedure for DimAirline
CREATE PROCEDURE dbo.UpdateDimAirline
@aid smallint,
@aName nvarchar(max),
@alias nvarchar(50),
@iata nvarchar(50),
@icao nvarchar(50),
@callsign nvarchar(50),
@country nvarchar(max),
@active bit
AS
BEGIN
if not exists (select sk_airline
from dbo.Dimairline
where airline_id = @aid)
BEGIN
insert into dbo.Dimairline
```

```
(@iata, @icao, @pname, GETDATE(),
GETDATE())
END;
if exists (select sk_aircraft
from dbo.DimAircraft
where planeName = @pname)
BEGIN
update dbo.DimAircraft
set iataCode = @iata,
icaoCode = @icao,
ModifiedDate = GETDATE()
where planeName = @pname
END;
END;
```

```
(airline_id, Name, Alias, IATA,
ICAO,Callsign,Country,Active,insertDate,modifie
dDate)
values
(@aid, @aName, @alias,
@iata,@icao,@callsign,@country,
@active ,GETDATE(), GETDATE())
END;
if exists (select sk_airline
from dbo.DimAirline
where Airline_ID = @aid)
BEGIN
update dbo.DimAirline
set Name = @aName,
alias = @alias,
IATA = @iata,
ICAO = @icao,
Callsign=@callsign,
Country=@country,
active=@active,
ModifiedDate = GETDATE()
where Airline_ID = @aid
END;
END;
```

```
--Update Procedure for DimCountry
CREATE PROCEDURE dbo.UpdateDimCountry
@cname varchar(50),
@iso varchar(50),
@dafif varchar(50)
AS
BEGIN
if not exists (select sk_country
from dbo.DimCountry
where country = @cname)
BEGIN
insert into dbo.DimCountry
(country, iso_code, dafif_code,
InsertDate, ModifiedDate)
values
(@cname, @iso, @dafif, GETDATE(),
GETDATE())
END;
if exists (select sk_country
from dbo.DimCountry
where country = @cname)
BEGIN
update dbo.DimCountry
set iso_code = @iso,
dafif_code = @dafif,
ModifiedDate = GETDATE()
where country = @cname
END;
END;
```

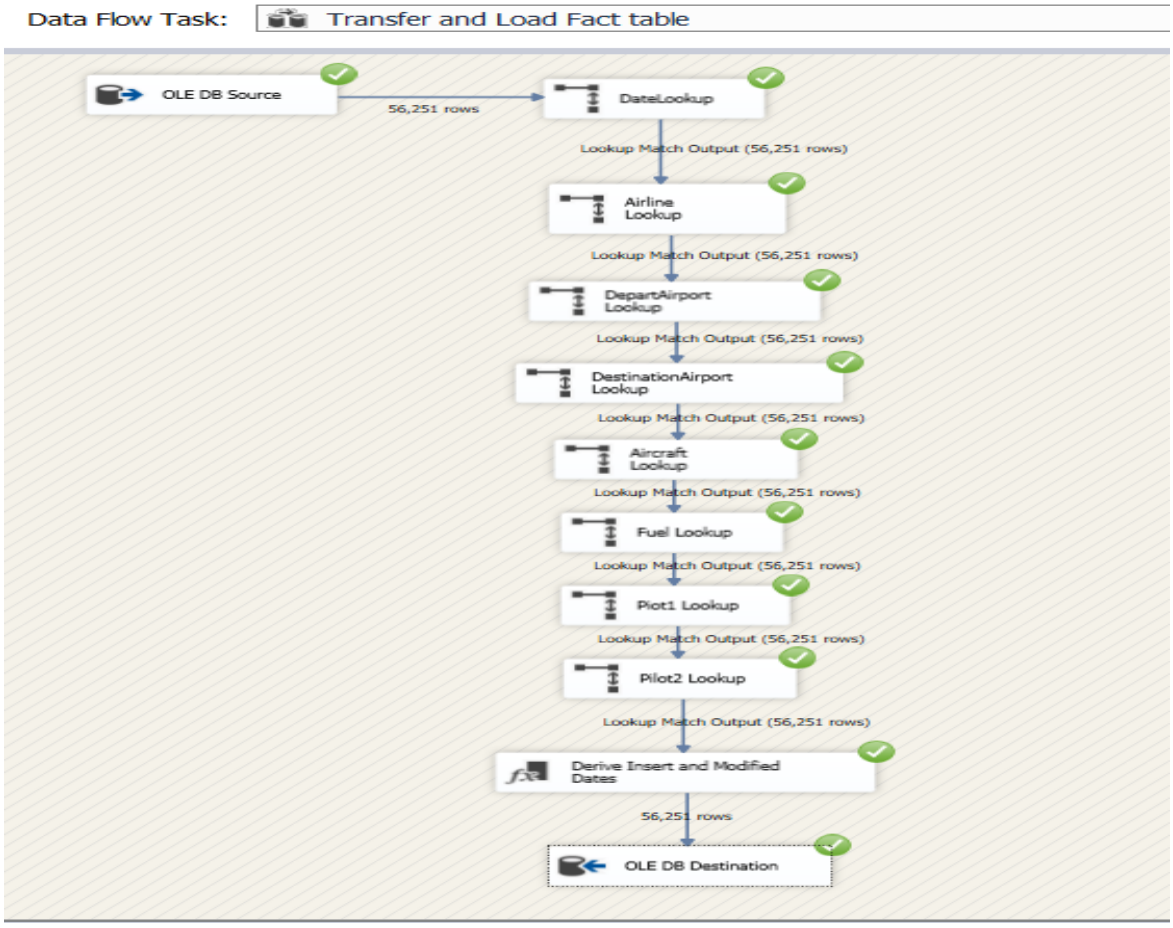
```
--Update Procedure for DimFuelCompany
CREATE PROCEDURE dbo.UpdateDimFC
@company nvarchar(255),
@country nvarchar(255),
@iso nvarchar(255),
@source nvarchar(255),
@region nvarchar(255),
@plevel nvarchar(255),
@pgroup nvarchar(255),
@audit nvarchar(255),
@indicator nvarchar(255),
@units nvarchar(255)

AS
BEGIN
if not exists (select sk_fuelCompany
from dbo.DimFuelCompany
where company = @company)
BEGIN
insert into dbo.DimFuelCompany
(company, country, countryISO,
dataSource,region,productionlevel,productiongro
up,
audited,indicator,units,InsertDate,
ModifiedDate)
values
(@company, @country, @iso,
@source,@region,@plevel,@pgroup,@audit,@indicat
or,
@units,GETDATE(), GETDATE())
END;
if exists (select sk_fuelCompany
from dbo.DimFuelCompany
where company = @company)
BEGIN
update dbo.DimFuelCompany
set country = @country,
countryiso = @iso,
dataSource=@source,
region=@region,
productionlevel=@plevel,
productiongroup=@pgroup,
audited=@audit,
indicator=@indicator,
```

```
units=@units,
ModifiedDate = GETDATE()
where company = @company
END;
END;
```

```
--Update Procedure for DimFuel
CREATE PROCEDURE dbo.UpdateDimFuel
@fcode int,
@company int,
@octane int,
@type nvarchar(max)
AS
BEGIN
if not exists (select sk_fuel
from dbo.DimFuel
where fcode = @fcode)
BEGIN
insert into dbo.Dimfuel
(fcode, company, fueltype,
octane,InsertDate, ModifiedDate)
values
(@fcode, @company, @type,@octane,
GETDATE(), GETDATE())
END;
if exists (select sk_fuel
from dbo.DimFuel
where fcode = @fcode)
BEGIN
update dbo.DimFuel
set company = @company,
FuelType = @type,
Octane = @octane,
ModifiedDate = GETDATE()
where Fcode = @fcode
END;
END;
```

```
--Update DimAirport Procedure
CREATE PROCEDURE dbo.UpdateDimAirPort
@apID smallint,
@apName nvarchar(100),
@city nvarchar(50),
@country int,
@iata nvarchar(50),
@icao nvarchar(50),
@latitude float,
@longitude float,
@altitude smallint,
@timezone float,
@dst nvarchar(50),
@dbtimezone nvarchar(50),
@type nvarchar(50),
@source nvarchar(50)
AS
BEGIN
if not exists (select sk_airport
from dbo.DimAirport
where airport_ID = @apID)
BEGIN
insert into dbo.DimAirport
(Airport_ID, Name, City,
Country,Iata,Icao,latitude,longitude,altitude,t
imezone,
dst,database_timezone,type,source,InsertDate,
ModifiedDate)
values
(@apID, @apName, @city,
@country,@iata,@icao,@latitude,@longitude,@alti
tude,
@timezone,@dst,@dbtimezone,@type,@source,GETDAT
E(), GETDATE())
END;
if exists (select sk_airport
from dbo.DimAirport
where Airport_ID = @apID)
BEGIN
update dbo.DimAirport
set Name = @apName,
City = @city,
Country = @country,
iata = @iata,
ICAO = @icao,
latitude = @latitude,
longitude = @longitude,
Altitude = @altitude,
Timezone = @timezone,
dst=@dst,
Database_Timezone=@dbtimezone,
Type=@type,
Source=@source,
ModifiedDate = GETDATE()
where Airport_ID = @apID
END;
END;
```

8. Load into FactDetails Fact Table



- Data is extracted from the Flight details transactional table in the staging database.
- Date is matched with the datekey attribute of DimDate to confirm the foreign key constraint and to derive the surrogate key using a lookup.
- Airline is matched with the airline ID attribute of DimAirline to confirm the foreign key constraint and to derive the surrogate key using a lookup.
- Departure and destination airports is matched with the airport ID attribute of DimAirport to confirm the foreign key constraint and to derive the surrogate key using a lookup.
- Aircraft is matched with the name attribute of DimAircraft to confirm the foreign key constraint and to derive the surrogate key using a lookup.
- Fuel type is matched with the Fcode attribute of DimFuel to confirm the foreign key constraint and to derive the surrogate key using a lookup.
- Pilot1 and Polit2 is matched with the Unique ID attribute of DimPilot to confirm the foreign key constraint and to derive the surrogate key using a lookup.
- Data is then laded into the fact table in the data warehouse

Given below is a snippet of the fact table in the data warehouse;

	FlightID	Airline	DepartureAirport	DestinationAirport	Aircraft	FlightDate	FuelType	FuelGallonsUsed	GallonPrice	Pilot1	P1HourSalary	Pilot2	P2HourSalary	TotalTime	OtherExpensesHour	NoOfPassengers	Insertdate
1	5984	24	6319	6485	85	20130506	23	19593	10	14071	70	3822	55	3	3819	575	2022-05-22 06:22:02.400
2	5985	24	6319	6650	86	20130506	24	17638	8	14072	71	3823	74	9	2346	471	2022-05-22 06:22:02.400
3	5986	24	6319	5764	87	20130506	25	20987	8	14073	82	3824	55	1	3578	315	2022-05-22 06:22:02.400
4	5987	24	6319	6668	88	20130506	27	5736	10	14074	71	3825	73	4	3544	350	2022-05-22 06:22:02.400
5	5988	24	6319	2082	89	20130506	28	26801	6	14075	89	3826	55	9	3953	337	2022-05-22 06:22:02.400
6	5989	24	6319	6628	90	20130506	26	26931	5	14076	77	3827	72	9	3780	309	2022-05-22 06:22:02.400
7	5990	24	6319	6256	91	20130506	19	21852	10	14077	86	3828	68	3	3213	436	2022-05-22 06:22:02.400
8	5991	24	6319	6683	92	20130506	1	35448	7	14078	87	3829	63	7	3013	439	2022-05-22 06:22:02.400
9	5992	24	6319	6722	93	20130506	3	13722	6	14079	71	3830	64	6	2733	321	2022-05-22 06:22:02.400
10	5993	24	6319	6163	94	20130507	6	15324	7	14080	70	3831	74	10	2670	266	2022-05-22 06:22:02.400
11	5994	24	6319	6679	95	20130507	8	27607	10	14081	75	3832	71	7	2490	173	2022-05-22 06:22:02.400
12	5995	24	6319	6020	96	20130507	9	35650	6	14082	79	3833	68	2	3236	592	2022-05-22 06:22:02.400
13	5996	24	6319	6083	97	20130507	11	21277	5	14083	70	3834	72	6	3525	350	2022-05-22 06:22:02.400
14	5997	24	6319	6678	98	20130507	13	11166	7	14084	72	3835	57	1	2225	359	2022-05-22 06:22:02.400
15	5998	24	6319	6717	99	20130507	14	31983	9	14085	86	3836	74	1	3105	514	2022-05-22 06:22:02.400
16	5999	24	6319	6627	100	20130507	15	20710	7	14086	76	3837	63	7	3305	570	2022-05-22 06:22:02.400

modifiedDate	TotalFuelCost	TotalPilotSalary	TotalOtherExpenses	TotalExpense
2022-05-22 06:22:02.400	195930	375	11457	207762
2022-05-22 06:22:02.400	141104	1305	21114	163523
2022-05-22 06:22:02.400	167896	137	3578	171611
2022-05-22 06:22:02.400	57360	576	14176	72112
2022-05-22 06:22:02.400	160806	1296	35577	197679
2022-05-22 06:22:02.400	134655	1341	34020	170016
2022-05-22 06:22:02.400	218520	462	9639	228621
2022-05-22 06:22:02.400	248136	1050	21091	270277
2022-05-22 06:22:02.400	82332	810	16398	99540
2022-05-22 06:22:02.400	107268	1440	26700	135408
2022-05-22 06:22:02.400	276070	1022	17430	294522
2022-05-22 06:22:02.400	213900	294	6472	220666
2022-05-22 06:22:02.400	106385	852	21150	128387
2022-05-22 06:22:02.400	78162	129	2225	80516
2022-05-22 06:22:02.400	287847	160	3105	291112
2022-05-22 06:22:02.400	144970	973	23135	169078

The computed columns were calculated as follows;

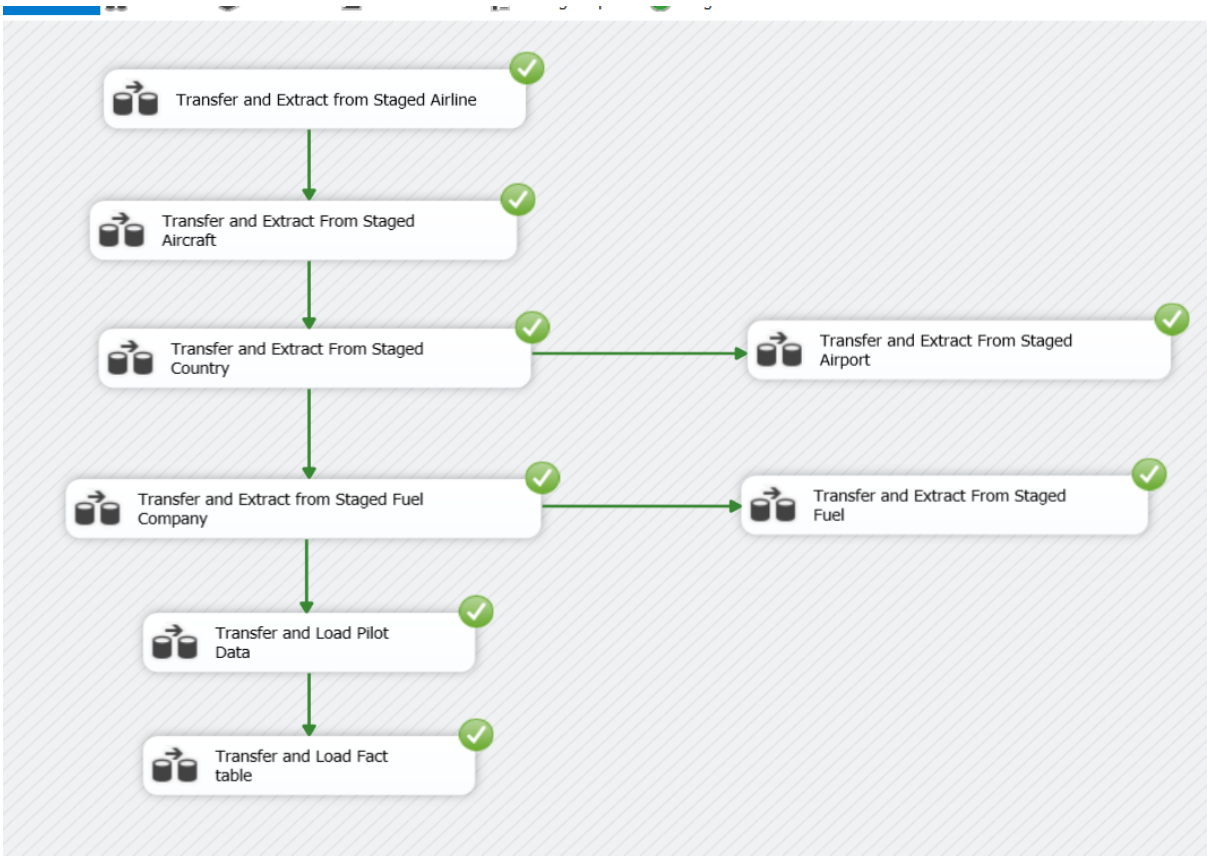
TotalFuelCost= ([FuelGallonsUsed]*[GallonPrice])

TotalPilotSalary= ([P1HourSalary]*[TotalTime]+[P2HourSalary]*[TotalTime])

TotalOtherExpenses= ([OtherExpensesHour]*[TotalTime])

TotalExpense=([FuelGallonsUsed]*[GallonPrice]) + ([P1HourSalary]*[TotalTime]+[P2HourSalary]*[TotalTime])+ ([OtherExpensesHour]*[TotalTime])

The following screen capture shows the completion of loading all data into the data ware house.



This end the process of extracting, transforming and loading data into the data warehouse.