Sri Lanka Institute of Information Technology

Assignment 01

Data Warehouse & Business Intelligence

IT22888716 – K.D.Y.Niwarthana

Y3.S1.WE.DS.02.02

Contents

1. Data Selection and Preparation

This dataset is about aviation accidents and incidents that were investigated between 2002 and 2007. According to international aviation rules (Annex 13 of the Convention on International Civil Aviation), an **aviation accident** happens during the operation of an aircraft — from the time someone boards with the intention to fly until everyone gets off — when one of the following occurs:

a) a person is seriously or fatally injured,
b) the aircraft suffers major damage or structural failure, or
c) the aircraft goes missing or becomes unreachable.

An aviation incident is different. It refers to an event that does not qualify as an accident but still affects or could affect the safety of flight operations. These accidents and incidents are investigated by government organizations such as the FAA (Federal Aviation Administration) and the NTSB (National Transportation Safety Board).

The FAA works to improve aviation safety by encouraging the sharing of safety information through systems like ASIAS (Aviation Safety Information Analysis and Sharing), which helps users search across multiple safety databases and get useful reports. The NTSB maintains records of aviation accidents and incidents dating back to 1962, while the World Aircraft Accident Summary (WAAS) gives global details about major aircraft accidents.

The original dataset, available here,

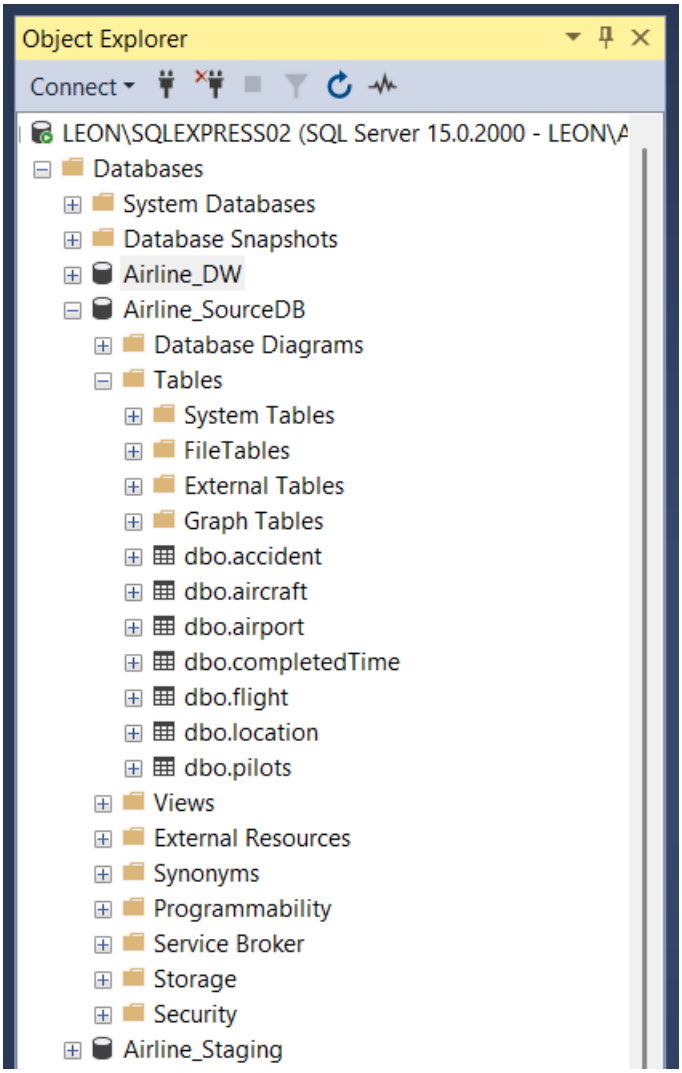
Aviation Accidents and Incidents (NTSB, FAA, WAAS)

combines data from multiple sources:

- Airline Accidents (NTSB investigation data from 1962 to 2007),
- FAA Accidents Data (covering incidents since 1978),
- NTSB Aviation Data (1982 to 2020), and
- World Aircraft Accident Summary (WAAS) (global accidents from 1990 to 2016).

For this assignment, I mainly focused on the Airline Accidents table because it already contains a rich set of data covering accident details, locations, aircraft involved, and more. Instead of using all the provided files, I used this one comprehensive table and created multiple data sources from it. To meet the assignment requirement of enriching the ETL process, I split the columns into separate logical tables, which allowed me to add more structure and create a clearer hierarchy.

In my customized version, I designed seven tables based on the different types of information found in the Airline Accidents data. These tables include accident details, location information, aircraft details, airport details, pilot details, and pilot addresses. This structured approach makes the data easier to manage and better suited for analysis, reporting, and dashboard creation.



Source	Source Type	Object Name	Description
Airline_SourceDB	CSV File	accident	Includes accident number, date, weather, injury severity, damage, and location.
	CSV File	aircraft	Contains detailed information about each aircraft including model and category.
	CSV File	airport	Lists airport names and countries for departure/landing locations.
	CSV File	flight	Main flight data including timings, purpose, aircraft ID, and pilot ID.
	CSV File	completedTime	Provides flight completion timestamps for accumulating fact processing.
	CSV File	location	Geographical location data linked to each accident.
	CSV File	pilots	Contains pilot personal details such as age, gender, and license.
Airline_SourceDB	TXT File	pilotaddress	Additional address details for pilots, joined on PilotID.

2. Description of the data set

Source	Source Type	Table Name	Column Name	Data Type
Airline_SourceDB	CSV File	accident	AccidentNumber	nvarchar(50)
		accident	Date	date
		accident	WeatherCondition	nvarchar(50)
		accident	InjurySeverity	nvarchar(50)
		accident	AircraftDamage	nvarchar(50)
		accident	LocationID	smallint

Source	Source Type	Table Name	Column Name	Data Type
Airline_SourceDB	CSV File	aircraft	RegistrationNumber	nvarchar(50)
		aircraft	AircraftCategory	nvarchar(50)
		aircraft	Make	nvarchar(50)
		aircraft	Model	nvarchar(50)
		aircraft	AmateurBuilt	bit
		aircraft	NumberofEngines	tinyint
		aircraft	EngineType	nvarchar(50)
		aircraft	passenger_seats	smallint
		aircraft	AirportCode	nvarchar(50)

Source	Source Type	Table Name	Column Name	Data Type
Airline_SourceDB	CSV File	completedTime	FlightNumber	nvarchar(50)
		completedTime	accm_txn_complete_time	datetime2

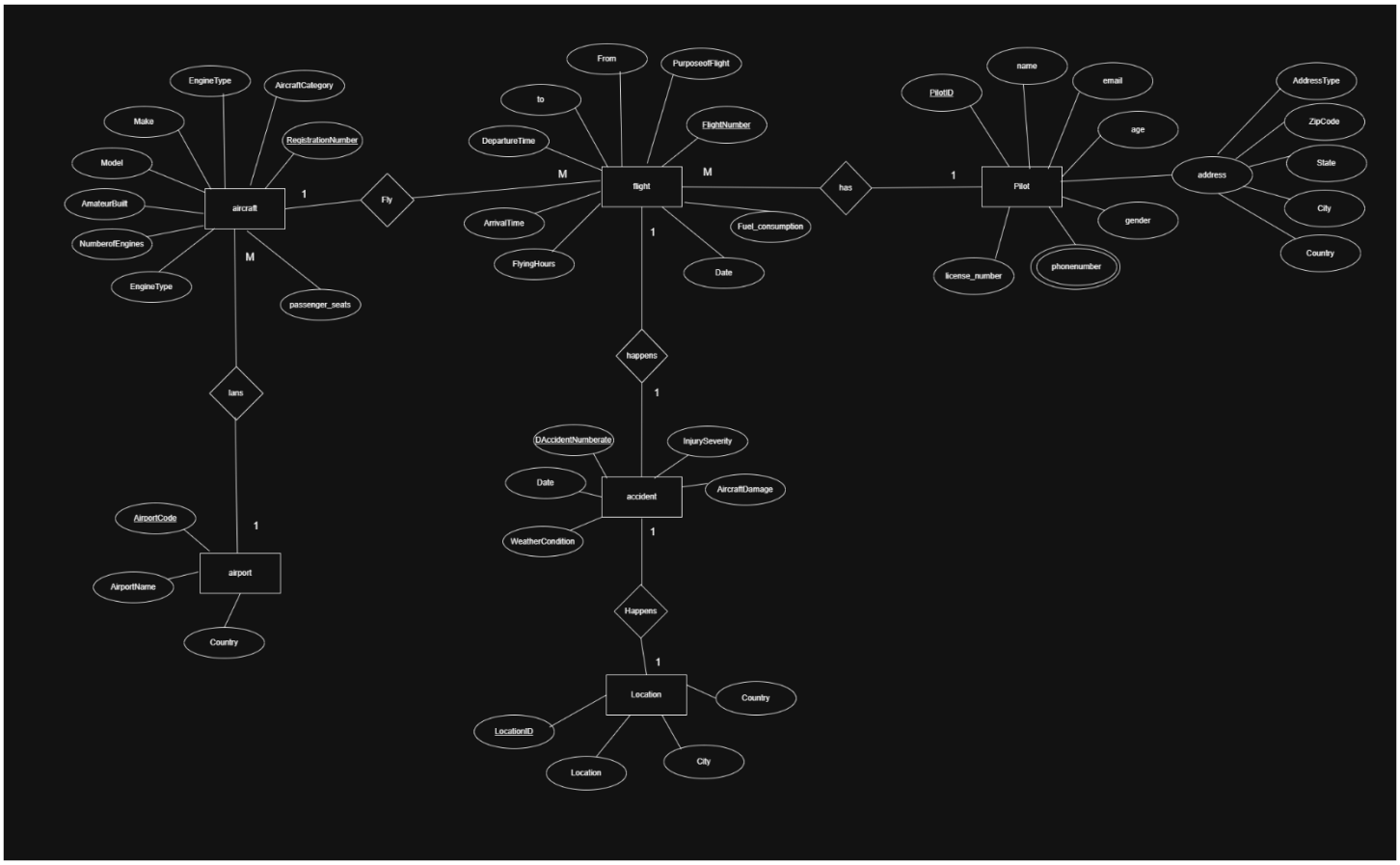
Source	Source Type	Table Name	Column Name	Data Type
Airline_SourceDB	CSV File	location	LocationID	smallint
		location	Location	nvarchar(200)
		location	City	nvarchar(100)
		location	Country	nvarchar(50)

Source	Source Type	Table Name	Column Name	Data Type
Airline_SourceDB	CSV File	pilot	PilotID	int
		pilot	Name	nvarchar(50)
		pilot	Email	nvarchar(50)
		pilot	PhoneNumber	nvarchar(50)
		pilot	Age	tinyint
		pilot	Gender	nvarchar(50)
		pilot	license_number	nvarchar(50)

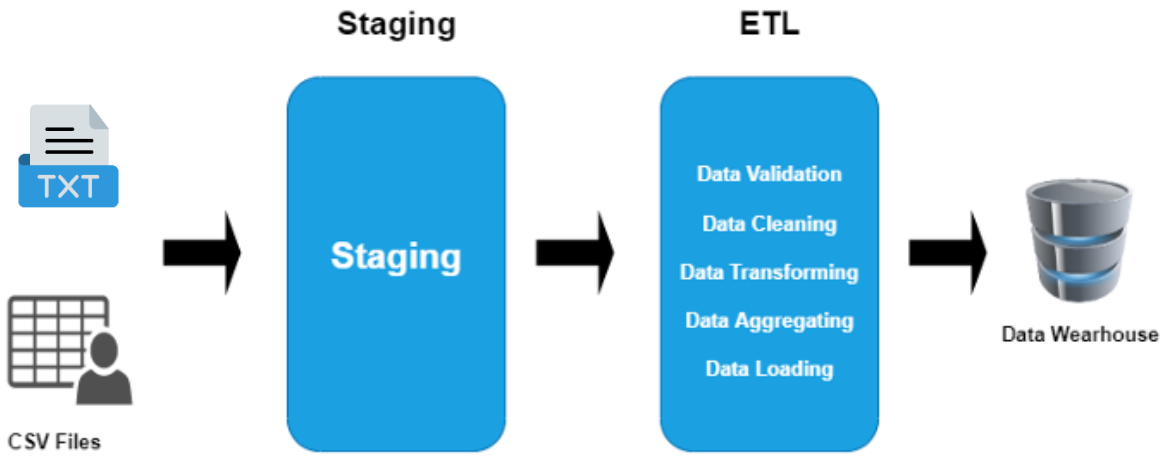
Source	Source Type	Table Name	Column Name	Data Type
Airline_SourceDB	TXT File	pilotAddress	PilotID	int
		pilotAddress	AddressType	nvarchar(50)
		pilotAddress	ZipCode	nvarchar(10)
		pilotAddress	State	nvarchar(100)
		pilotAddress	City	nvarchar(100)
		pilotAddress	Country	nvarchar(50)

Source	Source Type	Table Name	Column Name	Data Type
Airline_SourceDB	CSV File	airport	AirportCode	nvarchar(50)
		airport	AirportName	nvarchar(50)
		airport	Country	nvarchar(50)

3. ER Diagram


EngineType
AircraftCategory
Make
RegistrationNumber
Model
AmateurBuilt
aircraft
NumberofEngines
EngineType
passenger_seats
1
M
Fly
From
PurposeofFlight
to
FlightNumber
DepartureTime
M
flight
M
ArrivalTime
Fuel_consumption
FlyingHours
Date
1
has
1
PilotID
name
email
age
Pilot
gender
license_number
phonenumber
AddressType
ZipCode
address
State
City
Country
happens
1
lans
1
AirportCode
airport
AirportName
Country
DAccidentNumberate
InjurySeverity
Date
accident
AircraftDamage
WeatherCondition
1
Happens
1
Location
Country
LocationID
Location
City

4. Solution Architecture



Created Tables

- dbo.StgAccident
- dbo.StgAircraft
- dbo.StgAirport
- dbo.StgCompletedTime
- dbo.StgFlight
- dbo.StgLocation
- dbo.StgPilot
- dbo.StgPilotAddress

5. Data Warehouse Design and Development

Dimension Tables

- dbo.DimAccident
- dbo.DimAircraft
- dbo.DimAirport
- dbo.DimDate
- dbo.DimLocation
- dbo.DimPilot

Fact Table

- dbo.FactFlight

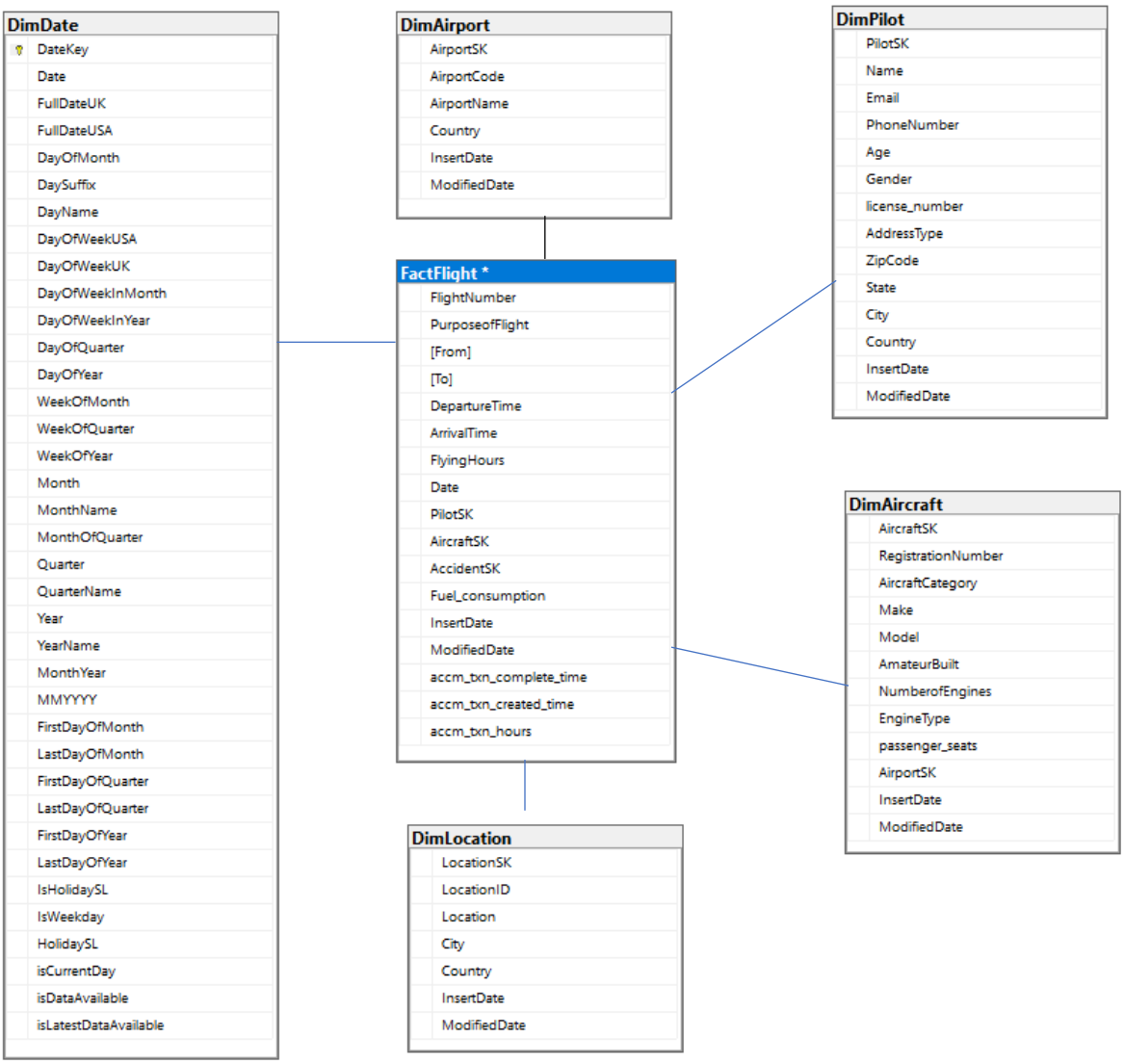
Dimension Types

1. Hierarchical Dimensions

- DimDate: Includes hierarchies like *Year → Quarter → Month → Day*.
- DimPilot (Addresses): Has a hierarchy of *Country → City → State → ZipCode*.
- DimLocation: Follows the hierarchy *Location → City → Country*.

2. Slowly Changing Dimensions

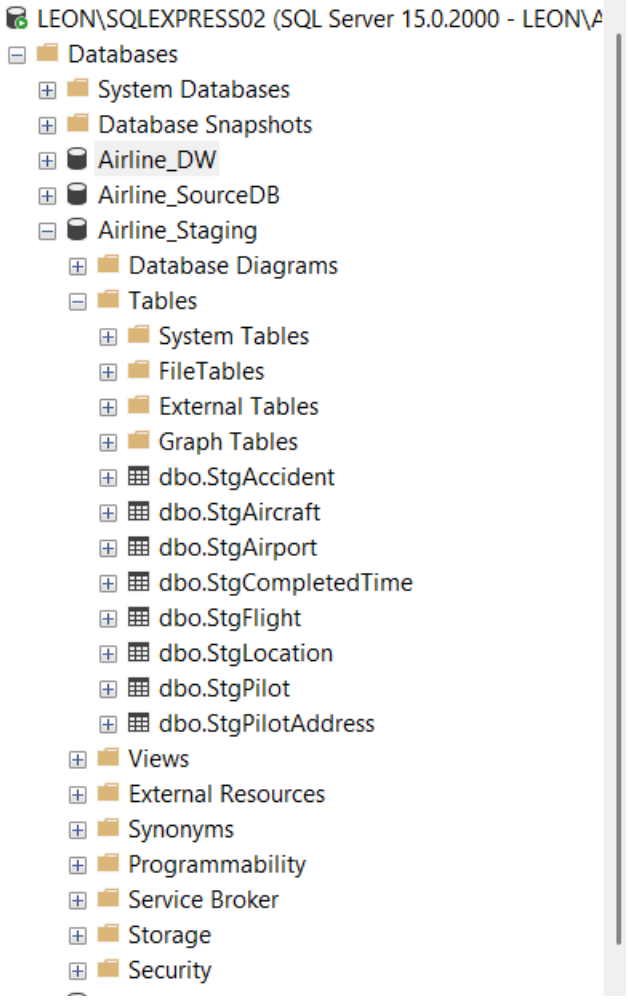
- DimPilot is considered a slowly changing dimension because attributes like PhoneNumber may change over time. Such changes are tracked to preserve historical accuracy.

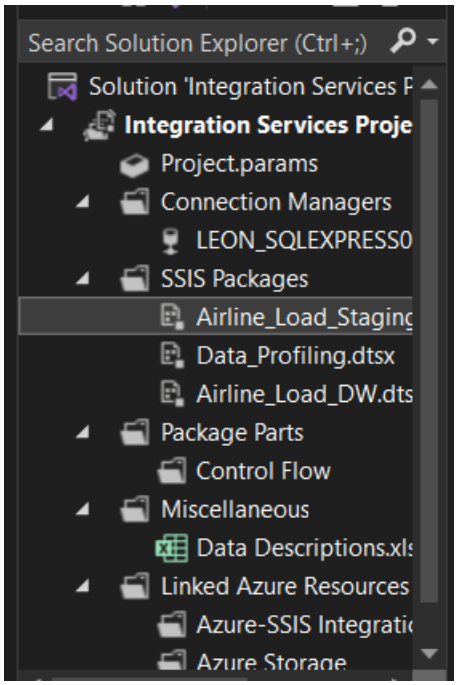

DimDate
DateKey
Date
FullDateUK
FullDateUSA
DayOfMonth
DaySuffix
DayName
DayOfWeekUSA
DayOfWeekUK
DayOfWeekInMonth
DayOfWeekInYear
DayOfQuarter
DayOfYear
WeekOfMonth
WeekOfQuarter
WeekOfYear
Month
MonthName
MonthOfQuarter
Quarter
QuarterName
Year
YearName
MonthYear
MMYYYY
FirstDayOfMonth
LastDayOfMonth
FirstDayOfQuarter
LastDayOfQuarter
FirstDayOfYear
LastDayOfYear
IsHolidaySL
IsWeekday
HolidaySL
isCurrentDay
isDataAvailable
isLatestDataAvailable
DimAirport
AirportSK
AirportCode
AirportName
Country
InsertDate
ModifiedDate
FactFlight *
FlightNumber
PurposeofFlight
[From]
[To]
DepartureTime
ArrivalTime
FlyingHours
Date
PilotSK
AircraftSK
AccidentSK
Fuel_consumption
InsertDate
ModifiedDate
accm_txn_complete_time
accm_txn_created_time
accm_txn_hours
DimLocation
LocationSK
LocationID
Location
City
Country
InsertDate
ModifiedDate
DimPilot
PilotSK
Name
Email
PhoneNumber
Age
Gender
license_number
AddressType
ZipCode
State
City
Country
InsertDate
ModifiedDate
DimAircraft
AircraftSK
RegistrationNumber
AircraftCategory
Make
Model
AmateurBuilt
NumberofEngines
EngineType
passenger_seats
AirportSK
InsertDate
ModifiedDate

6. ETL development

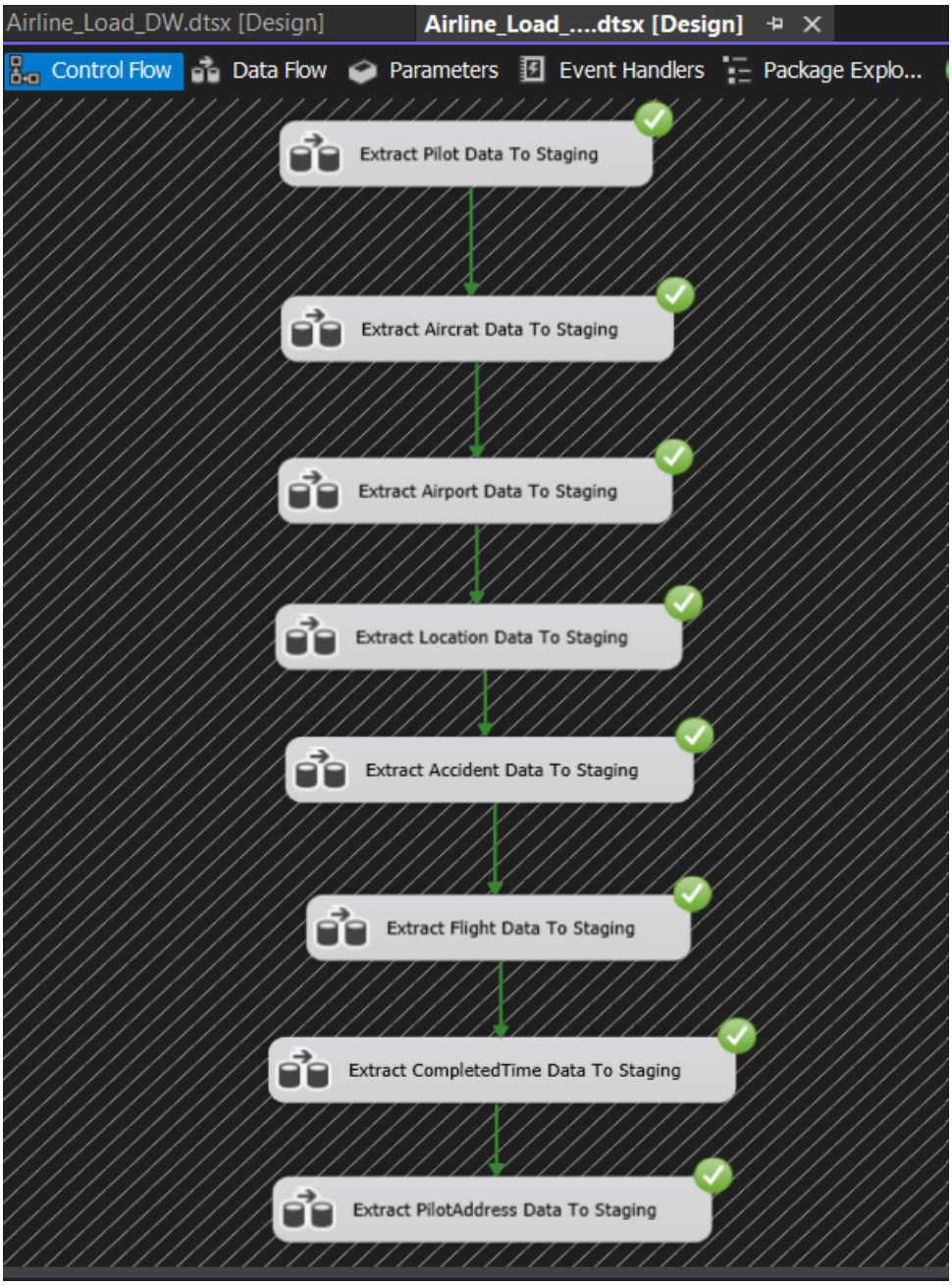
In this step, all data sources were imported into staging tables using the appropriate data connections.

- Flat File Connection was used to import data from .txt and .csv files.
- All the imported data was stored in the Airline_Staging database.



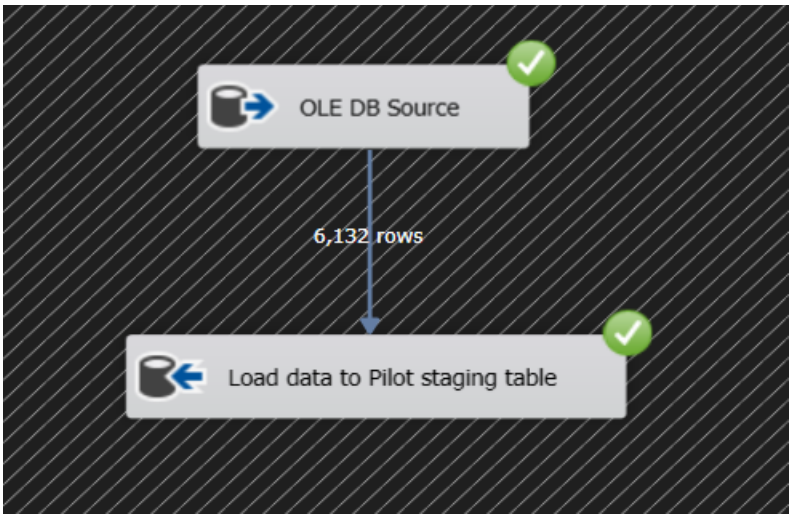


Visual Studio Control Flow of Extract

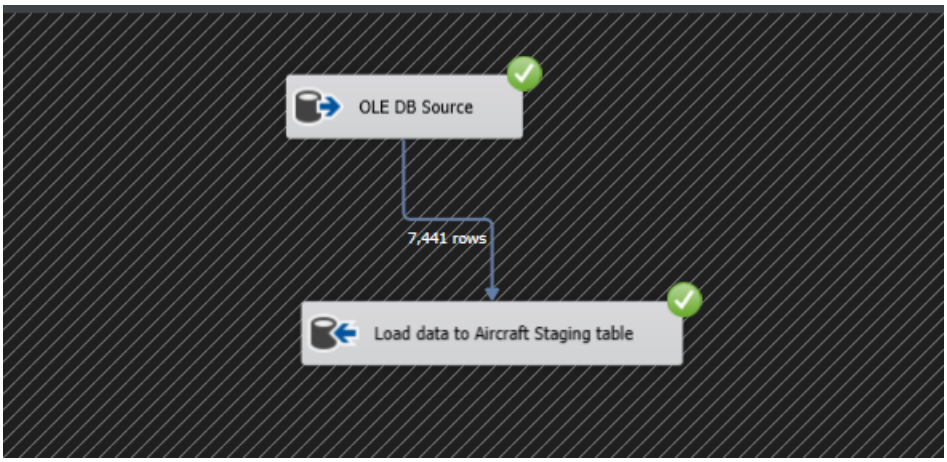


Data types of Data Flows

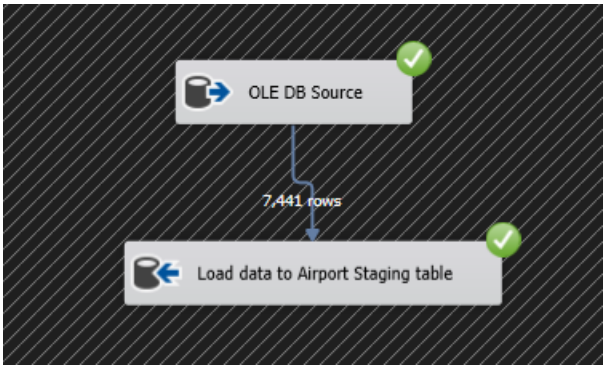
Extract Pilot Data to staging



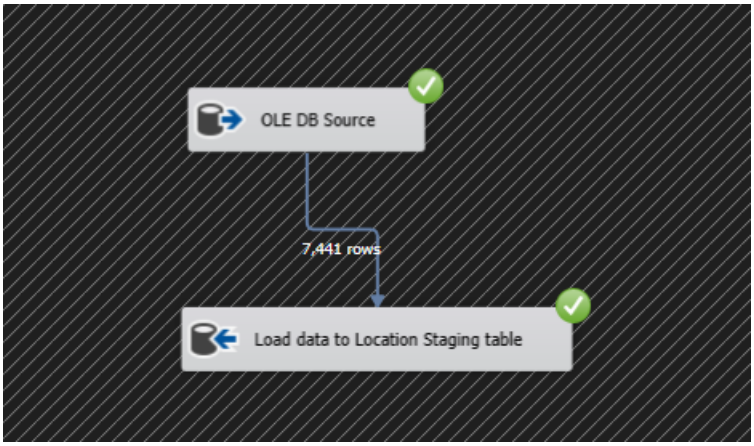
Extract Aircraft Data to staging



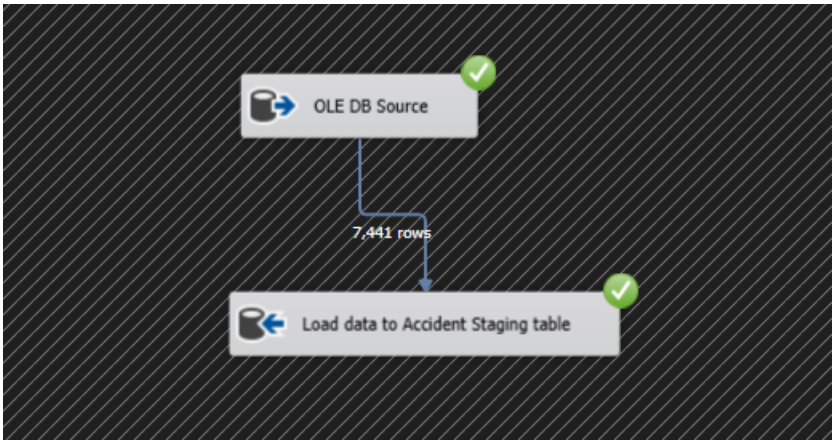
Extract Airport Data to staging



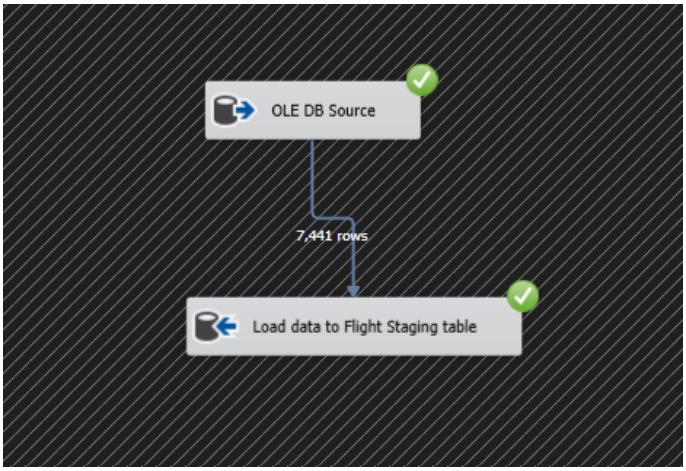
Extract Location Data to staging



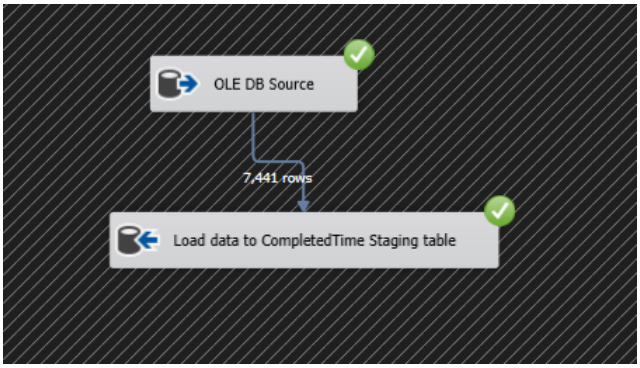
Extract Accident Data to staging



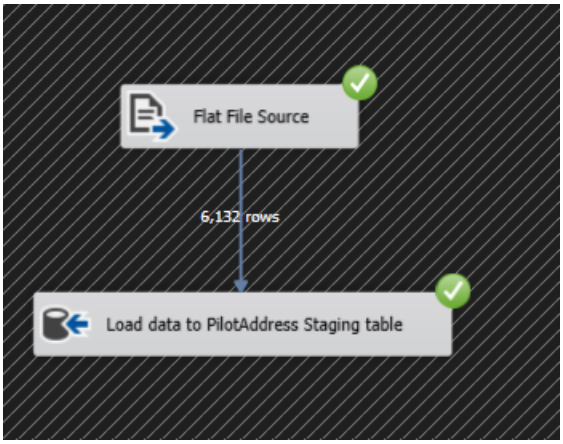
Extract Flight Data to staging



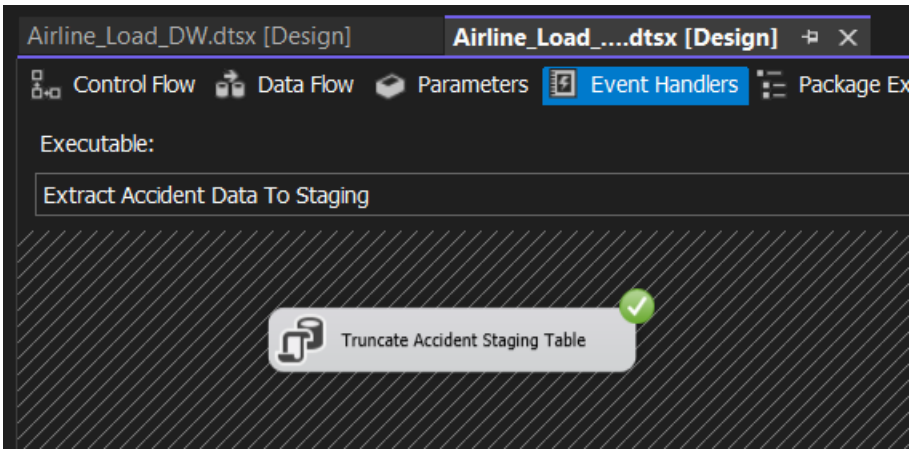
Extract CompletedTime Data to staging

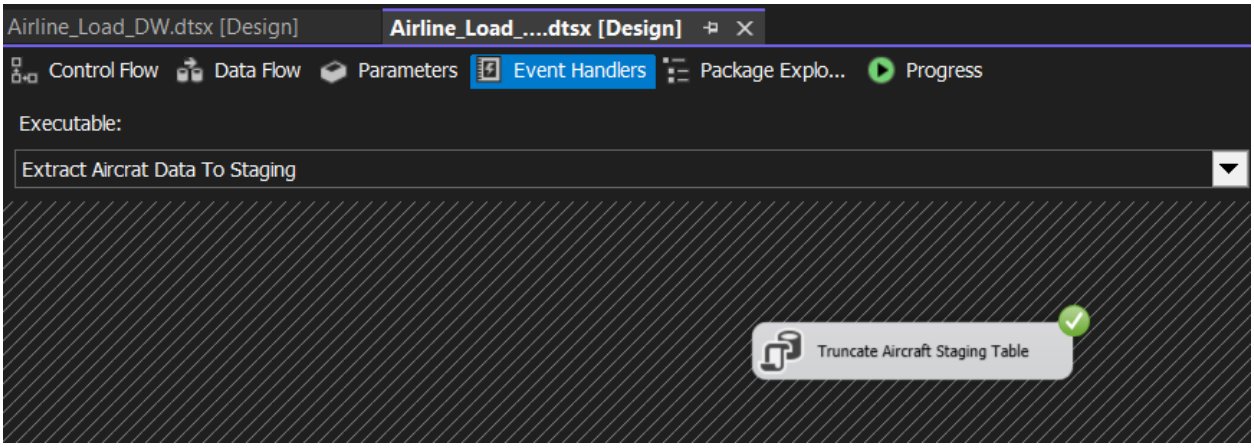


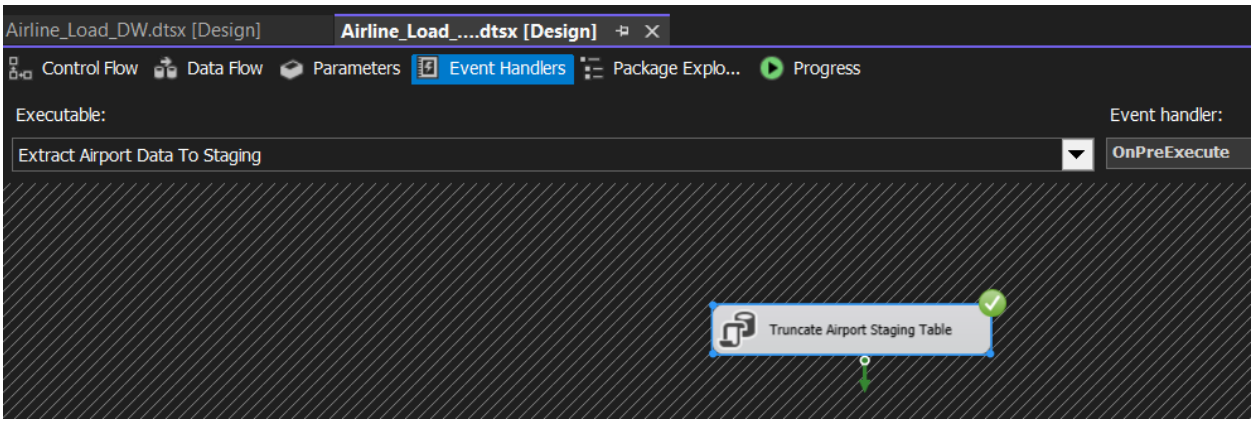
Extract PilotAddress Data to staging

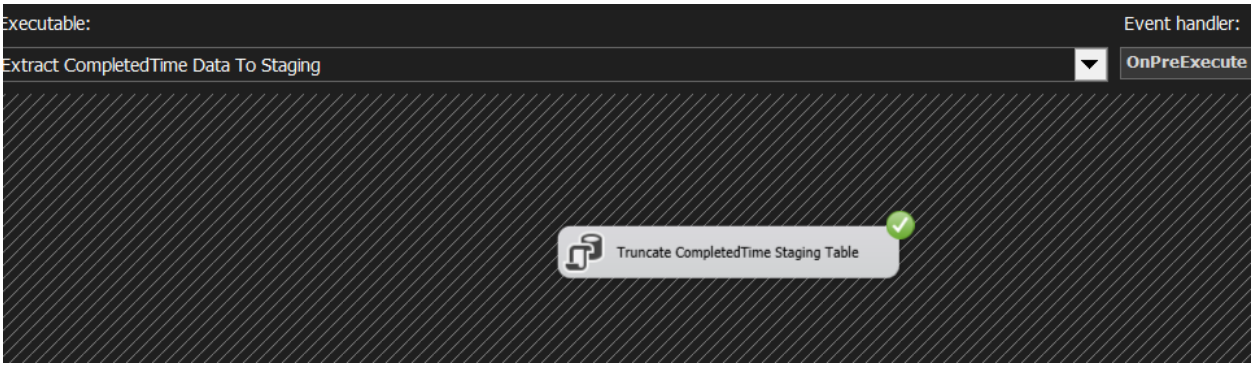


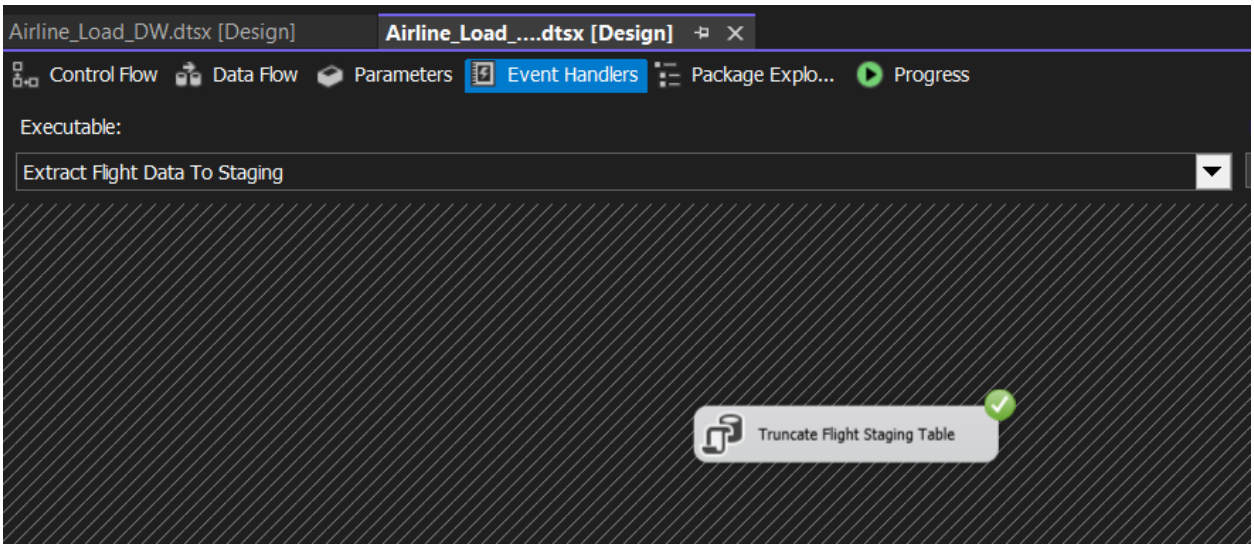
Event Handling (Truncate Staging Data)

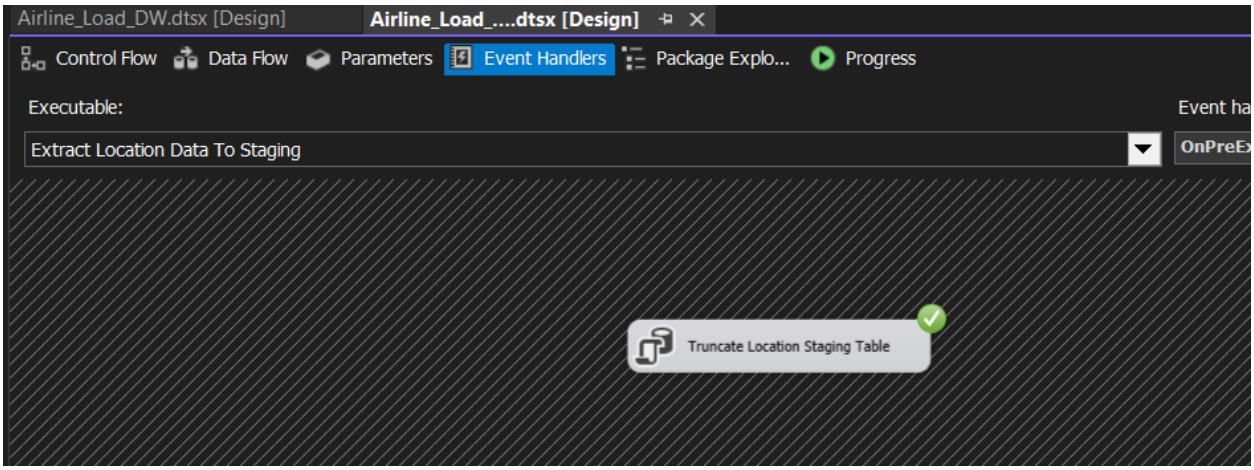


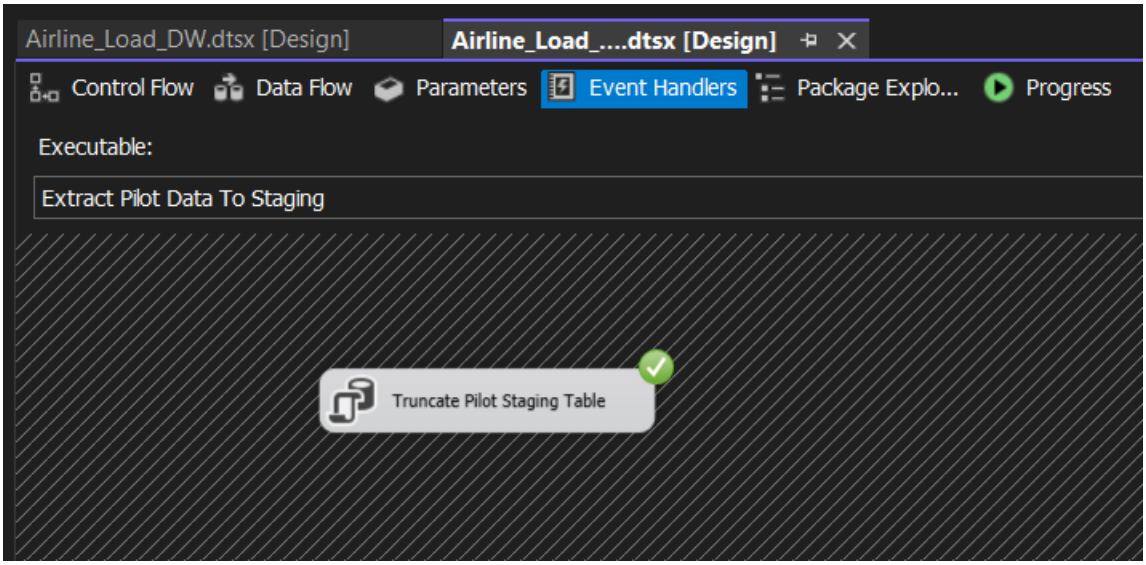


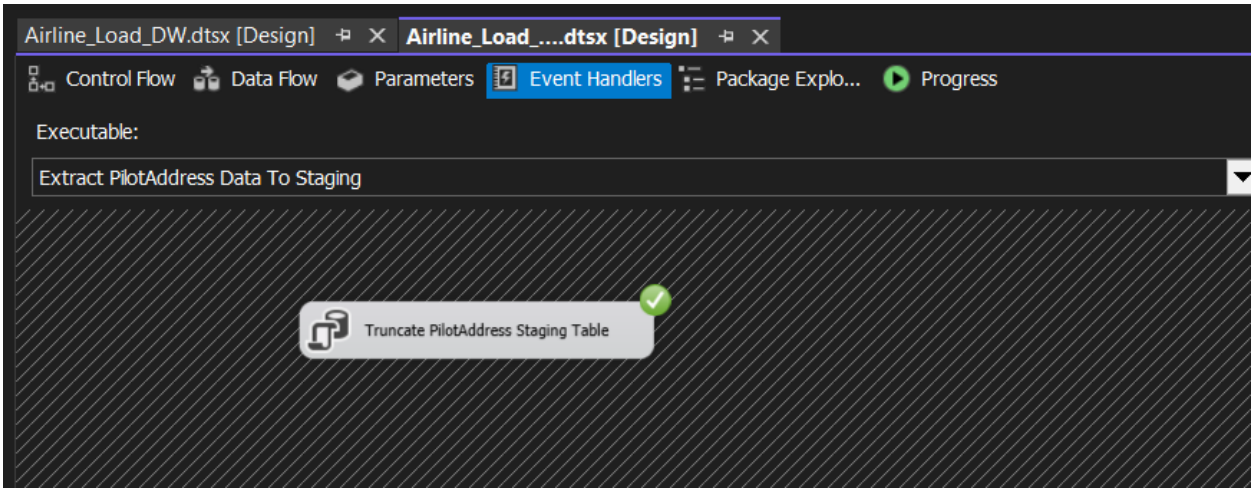



Executable:
Event handler:
Extract CompletedTime Data To Staging
OnPreExecute
Truncate CompletedTime Staging Table


Airline_Load_DW.dtsx [Design]
Airline_Load_....dtsx [Design]
Control Flow
Data Flow
Parameters
Event Handlers
Package Explo...
Progress
Executable:
Extract Flight Data To Staging
Truncate Flight Staging Table

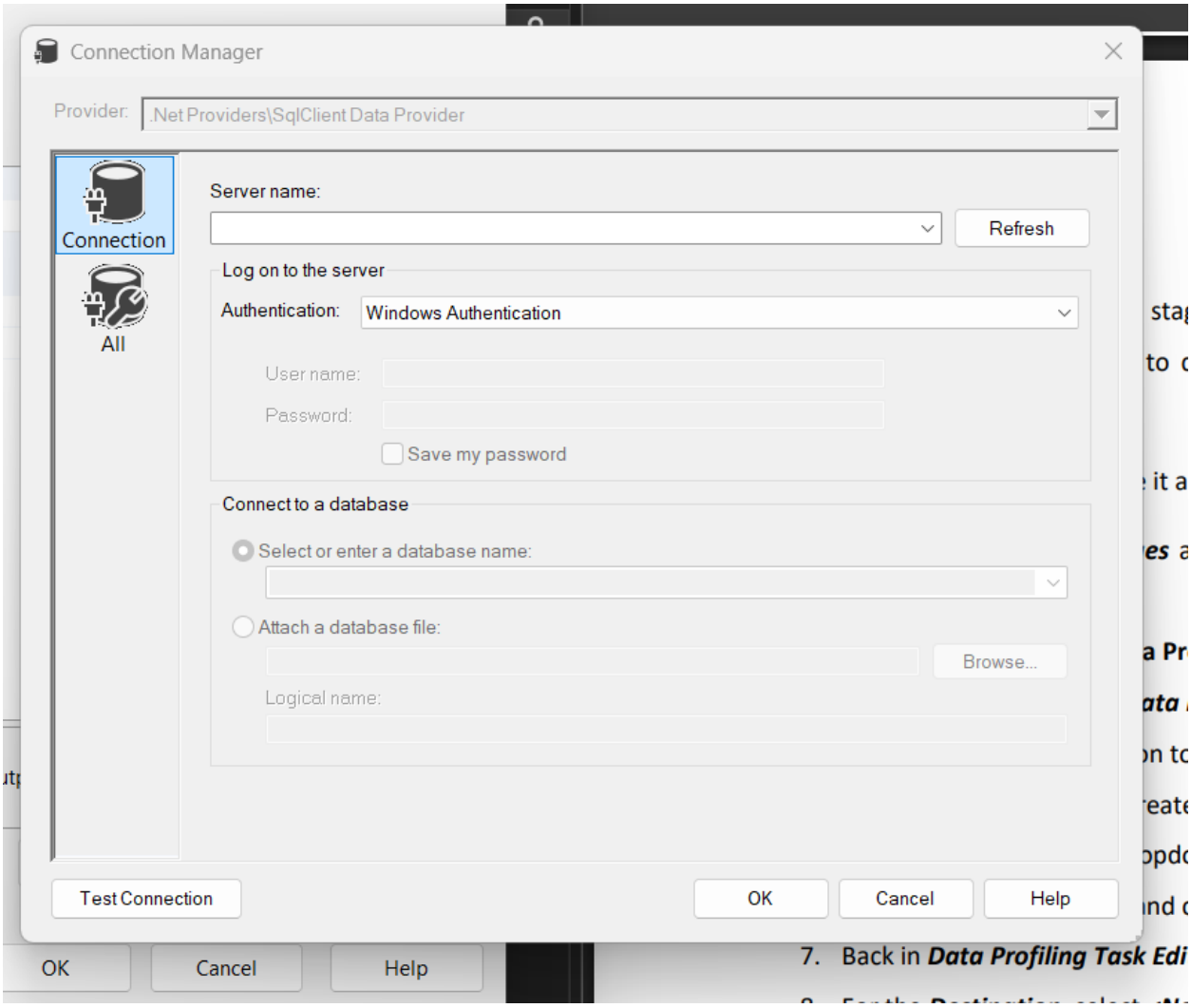

Airline_Load_DW.dtsx [Design]
Airline_Load_....dtsx [Design]
Control Flow
Data Flow
Parameters
Event Handlers
Package Explo...
Progress
Executable:
Extract Location Data To Staging
Truncate Location Staging Table





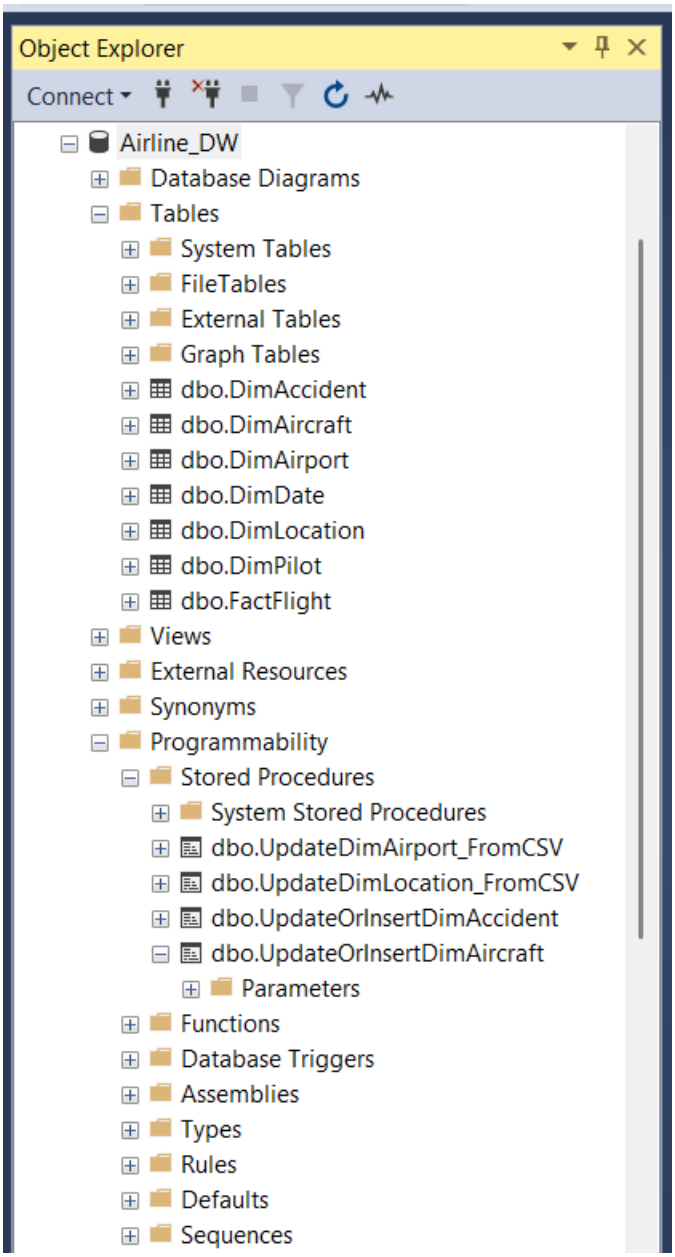
Data Profiling

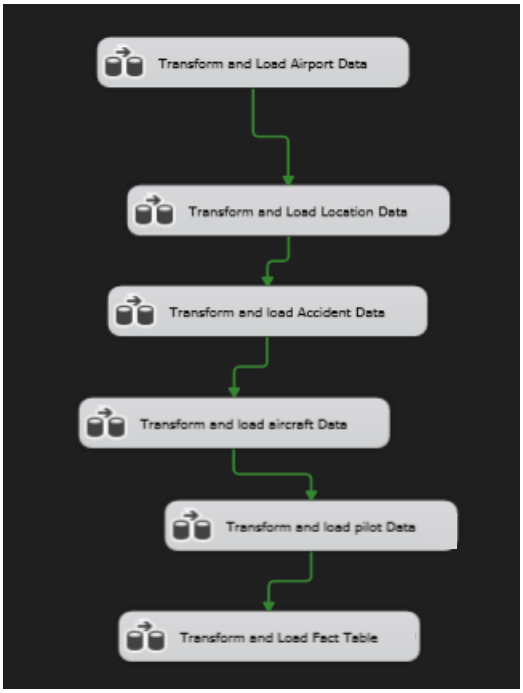
Unfortunately, I was unable to perform data profiling because the connection could not be established through the connection manager.



Transform & Load

In this step, both the Transform and Load processes were completed. First, the dimension tables were created in the Data Warehouse database. After that, using the appropriate components, data from the staging tables was loaded into the warehouse database, Airline_DW, which includes the following tables:


Object Explorer
Connect
Airline_DW
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.DimAccident
dbo.DimAircraft
dbo.DimAirport
dbo.DimDate
dbo.DimLocation
dbo.DimPilot
dbo.FactFlight
Views
External Resources
Synonyms
Programmability
Stored Procedures
System Stored Procedures
dbo.UpdateDimAirport_FromCSV
dbo.UpdateDimLocation_FromCSV
dbo.UpdateOrInsertDimAccident
dbo.UpdateOrInsertDimAircraft
Parameters
Functions
Database Triggers
Assemblies
Types
Rules
Defaults
Sequences


Transform and Load Airport Data
Transform and Load Location Data
Transform and load Accident Data
Transform and load aircraft Data
Transform and load pilot Data
Transform and Load Fact Table

Stored Procedures

UpdateDimAirport_FromCSV

SQLQuery60.sql -...W (LEON\ASUS (94)) | SQLQuery59.sql -...W (LEON\ASUS (90)) | LEON\SQLEXPRESS...e_DW

```sql
USE [Airline_DW]
GO
/****** Object:  StoredProcedure [dbo].[UpdateDimAirport_FromCSV]    Script Date: 5/1/2025 9:12:06 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[UpdateDimAirport_FromCSV]
    @AirportCode NVARCHAR(50),
    @AirportName NVARCHAR(50),
    @Country NVARCHAR(50)
AS
BEGIN
    -- Check if airport already exists
    IF NOT EXISTS (
        SELECT 1
        FROM dbo.DimAirport
        WHERE AirportCode = @AirportCode
    )
    BEGIN
        -- Insert new record
        INSERT INTO dbo.DimAirport (AirportCode, AirportName, Country, InsertDate, ModifiedDate)
        VALUES (@AirportCode, @AirportName, @Country, GETDATE(), GETDATE())
    END
    ELSE
    BEGIN
        -- Update existing record
        UPDATE dbo.DimAirport
        SET
            AirportName = @AirportName,
            Country = @Country,
            ModifiedDate = GETDATE()
        WHERE AirportCode = @AirportCode
    END
END
```

UpdateDimLocation_FromCSV

SQLQuery61.sql -...W (LEON\ASUS (97)) | SQLQuery60.sql -...W (LEON\ASUS (94)) | SQLQuery59.sql -...W (LEON\ASUS (90))

```sql
USE [Airline_DW]
GO
/****** Object:  StoredProcedure [dbo].[UpdateDimLocation_FromCSV]    Script Date: 5/1/2025 9:13:08 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

ALTER PROCEDURE [dbo].[UpdateDimLocation_FromCSV]
    @LocationID SMALLINT,
    @Location NVARCHAR(200),
    @City NVARCHAR(100),
    @Country NVARCHAR(50)
AS
BEGIN
    -- Check if LocationID exists
    IF NOT EXISTS (
        SELECT 1
        FROM dbo.DimLocation
        WHERE LocationID = @LocationID
    )
    BEGIN
        -- Insert new record
        INSERT INTO dbo.DimLocation (
            LocationID, Location, City, Country, InsertDate, ModifiedDate
        )
        VALUES (
            @LocationID, @Location, @City, @Country, GETDATE(), GETDATE()
        );
    END
    ELSE
    BEGIN
        -- Update existing record
        UPDATE dbo.DimLocation
        SET
            Location = @Location,
            City = @City,
            Country = @Country,
            ModifiedDate = GETDATE()
        WHERE LocationID = @LocationID;
    END
END
```

UpdateOrInsertDimAccident

SQLQuery62.sql - L...(LEON\ASUS (100)) | SQLQuery61.sql -...W (LEON\ASUS (97)) | SQLQuery60.sql -...W (LEON\ASUS (94)) | SQLQuery

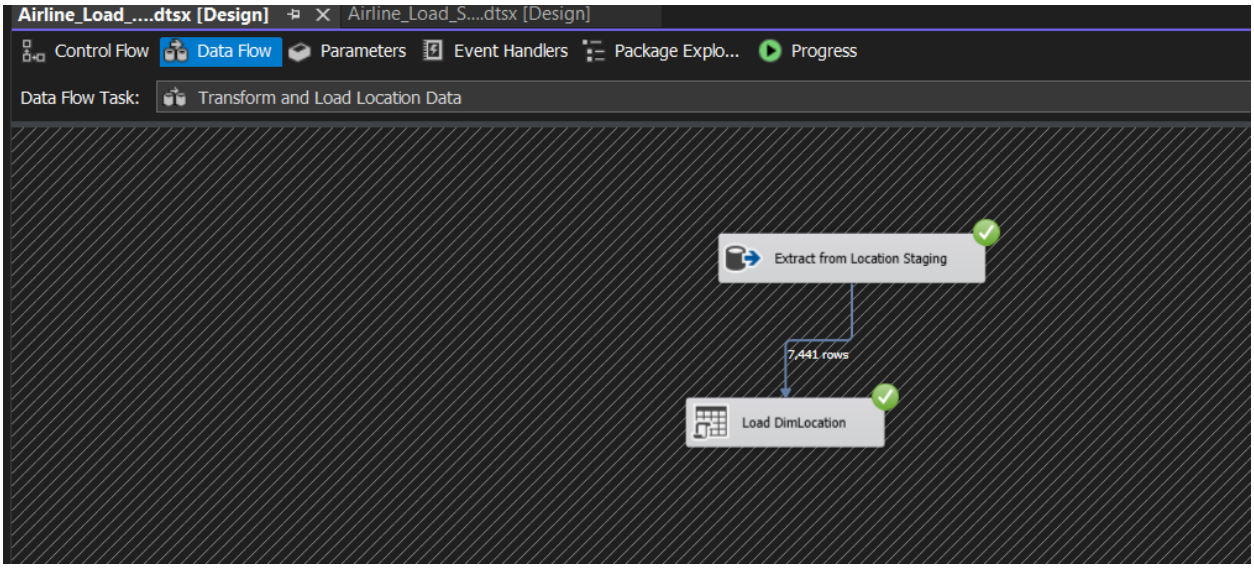
```sql
USE [Airline_DW]
GO
/****** Object:  StoredProcedure [dbo].[UpdateOrInsertDimAccident]    Script Date: 5/1/2025 9:13:49 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[UpdateOrInsertDimAccident]
    @AccidentNumber NVARCHAR(50),
    @Date DATETIME,
    @WeatherCondition NVARCHAR(50),
    @InjurySeverity NVARCHAR(50),
    @AircraftDamage NVARCHAR(50),
    @LocationSK INT      -- Use this if 'LocationSK' is correct
AS
BEGIN
    SET NOCOUNT ON;

    IF NOT EXISTS (
        SELECT 1
        FROM dbo.DimAccident
        WHERE AccidentNumber = @AccidentNumber
    )
    BEGIN
        INSERT INTO dbo.DimAccident
        (AccidentNumber, Date, WeatherCondition, InjurySeverity, AircraftDamage, LocationSK, InsertDate, ModifiedDate)
        VALUES
        (@AccidentNumber, @Date, @WeatherCondition, @InjurySeverity, @AircraftDamage, @LocationSK, GETDATE(), GETDATE());
    END
    ELSE
    BEGIN
        UPDATE dbo.DimAccident
        SET
            Date = @Date,
            WeatherCondition = @WeatherCondition,
            InjurySeverity = @InjurySeverity,
            AircraftDamage = @AircraftDamage,
            LocationSK = @LocationSK,
            ModifiedDate = GETDATE()
        WHERE AccidentNumber = @AccidentNumber;
    END
END
```

UpdateOrInsertDimAircraft

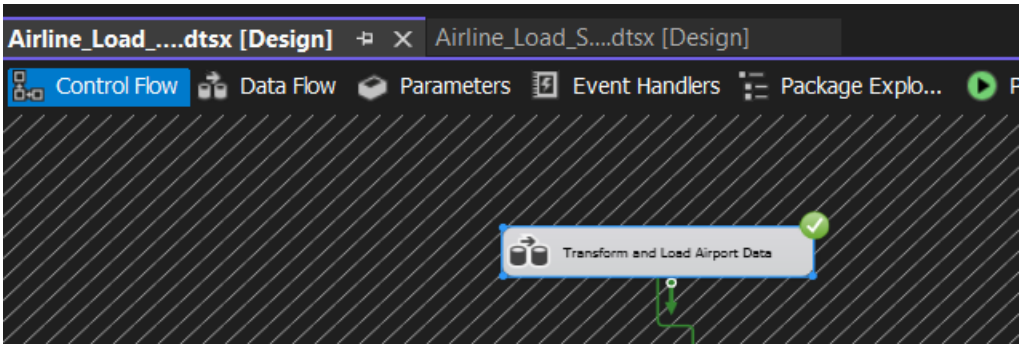
SQLQuery63.sql -...W (LEON\ASUS (88)) | SQLQuery62.sql - L...(LEON\ASUS (100))

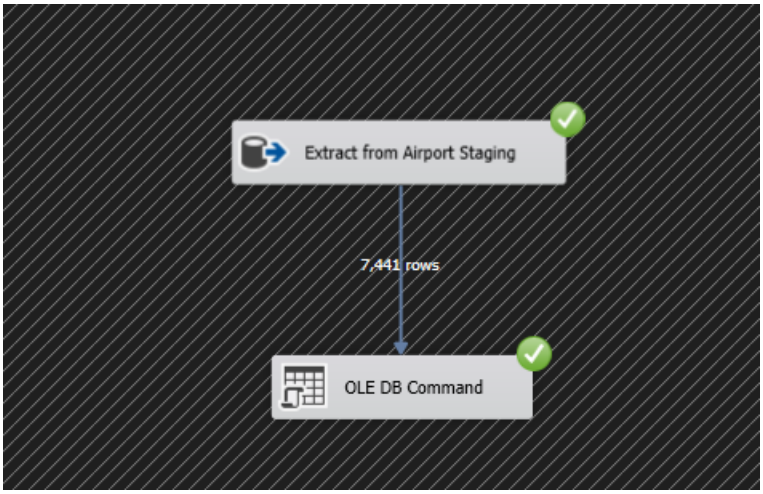
```sql
USE [Airline_DW]
GO
/****** Object:  StoredProcedure [dbo].[UpdateOrInsertDimAircraft]    Script Date: 5/1/2025 9:18:12 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

ALTER PROCEDURE [dbo].[UpdateOrInsertDimAircraft]
    @RegistrationNumber NVARCHAR(50),
    @AircraftCategory NVARCHAR(50),
    @Make NVARCHAR(50),
    @Model NVARCHAR(50),
    @AmateurBuilt BIT,
    @NumberofEngines TINYINT,
    @EngineType NVARCHAR(50),
    @PassengerSeats SMALLINT,
    @AirportSK INT
AS
BEGIN
    BEGIN TRY
        IF NOT EXISTS (
            SELECT 1
            FROM dbo.DimAircraft
            WHERE RegistrationNumber = @RegistrationNumber
        )
        BEGIN
            -- Insert new record
            INSERT INTO dbo.DimAircraft (
                RegistrationNumber, AircraftCategory, Make, Model, AmateurBuilt,
                NumberofEngines, EngineType, passenger_seats, AirportSK,
                InsertDate, ModifiedDate
            )
            VALUES (
                @RegistrationNumber, @AircraftCategory, @Make, @Model, @AmateurBuilt,
                @NumberofEngines, @EngineType, @PassengerSeats, @AirportSK,
                GETDATE(), GETDATE()
            );
        END
        ELSE
        BEGIN
            -- Update existing record
            UPDATE dbo.DimAircraft
            SET
                AircraftCategory = @AircraftCategory,
                Make = @Make,
                Model = @Model,
                AmateurBuilt = @AmateurBuilt,
                NumberofEngines = @NumberofEngines,
                EngineType = @EngineType,
                passenger_seats = @PassengerSeats,
                AirportSK = @AirportSK,
                ModifiedDate = GETDATE()
            WHERE RegistrationNumber = @RegistrationNumber;
        END
    END TRY
    BEGIN CATCH
        -- Optional: Log the error
        PRINT 'Error occurred in UpdateOrInsertDimAircraft: ' + ERROR_MESSAGE();
        -- Optionally: Rethrow if needed
        -- THROW;
    END CATCH
END
```

Transform and Load Location Data

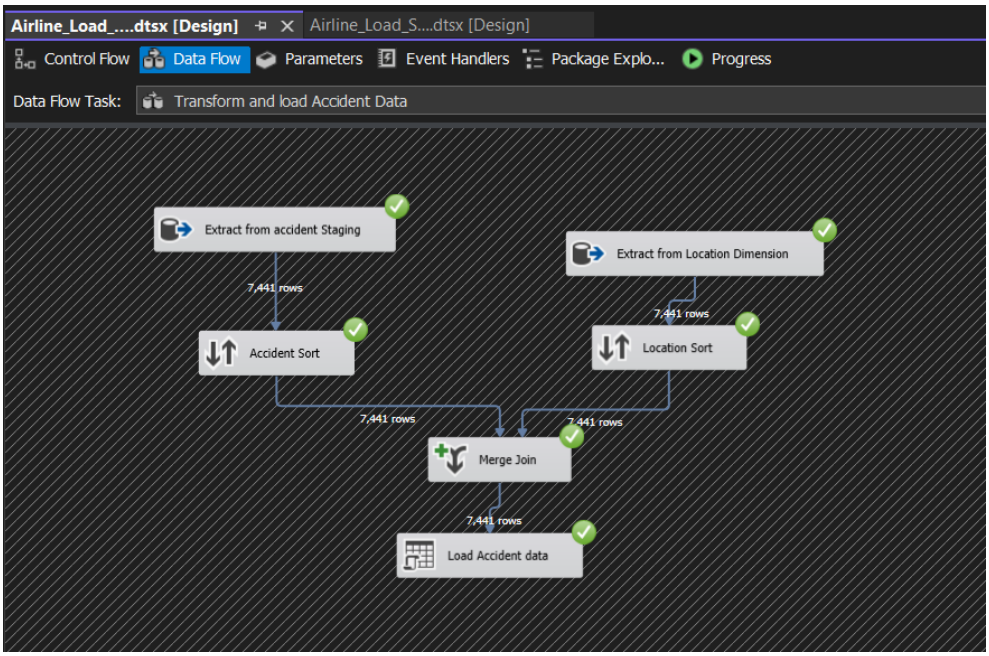


Transform and Load Airport Data

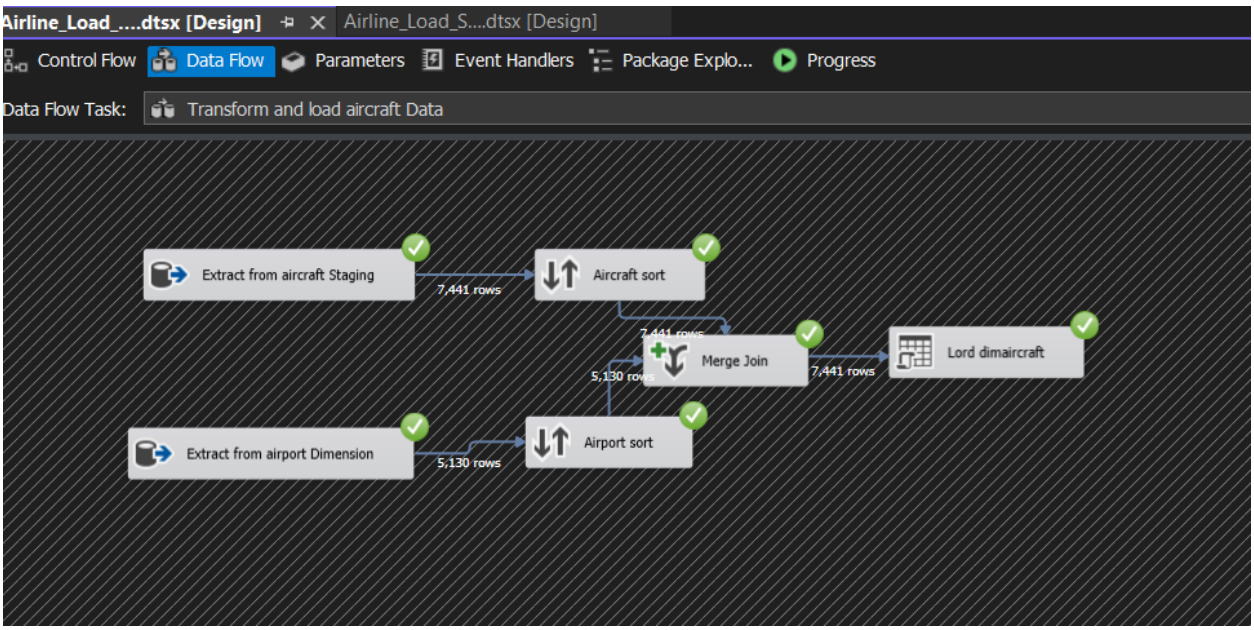




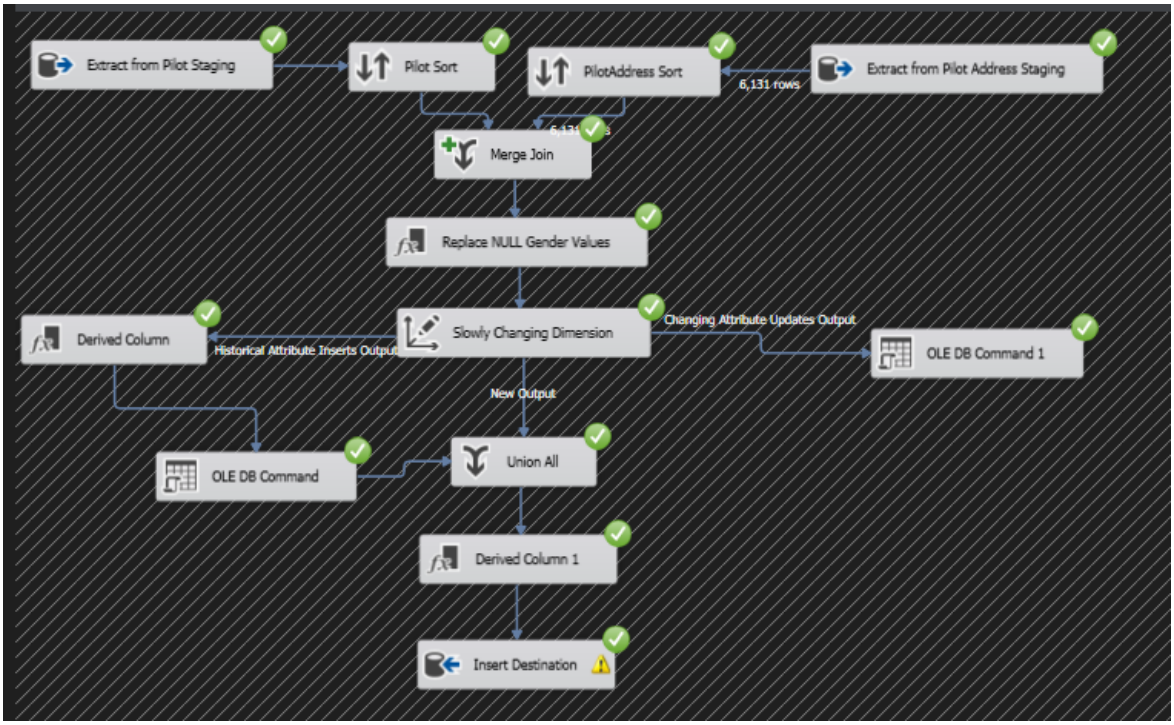
Transform and Load Accident Data



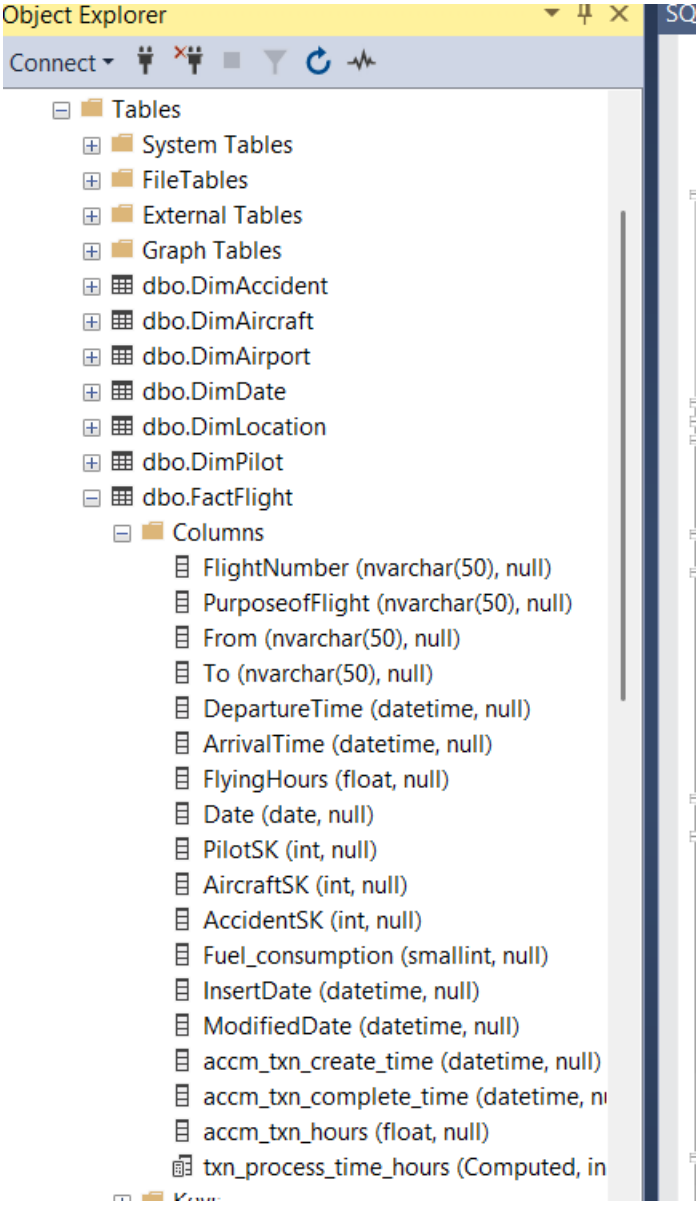
Transform and Load Aircraft Data



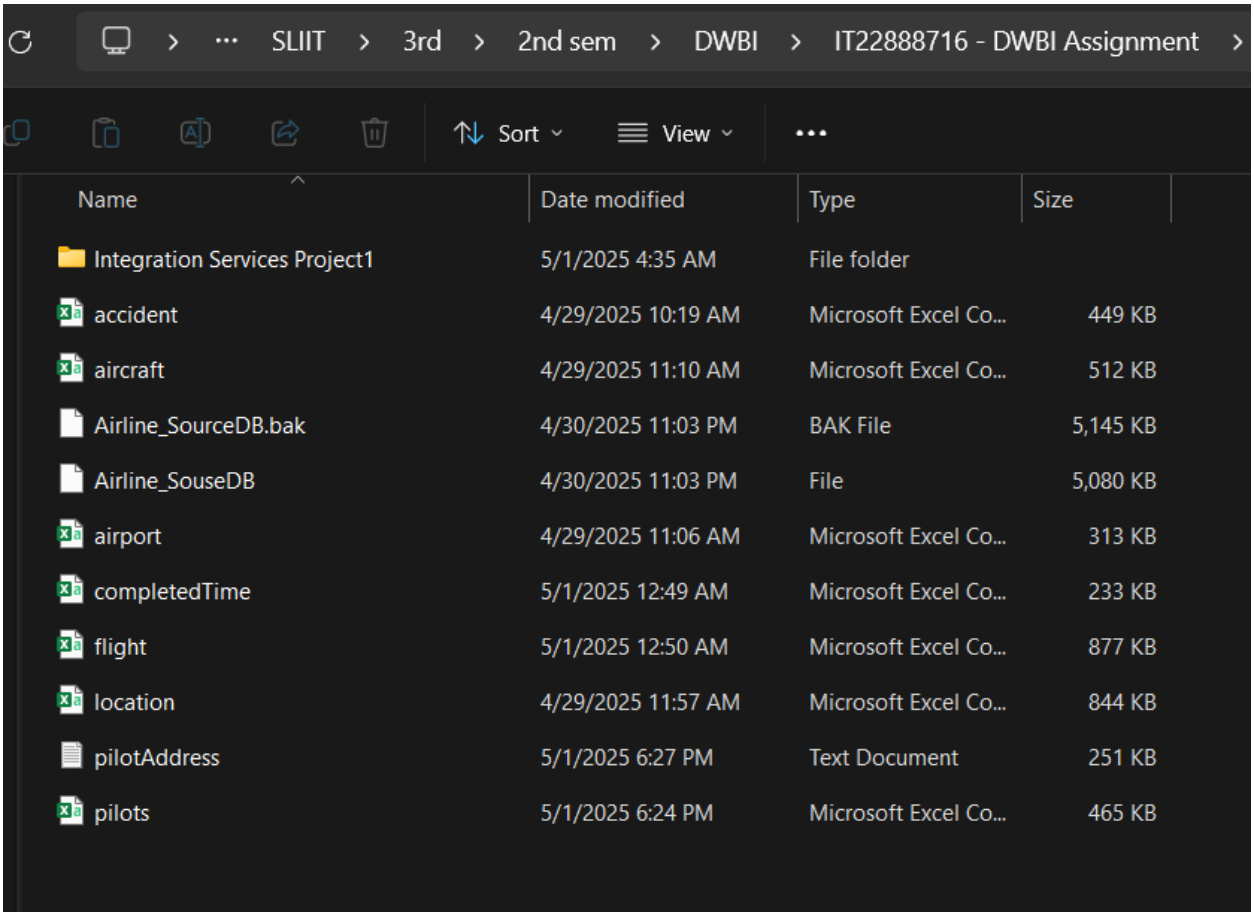
Transform and Load Pilot Data



Extending Fact Table with Additional Columns

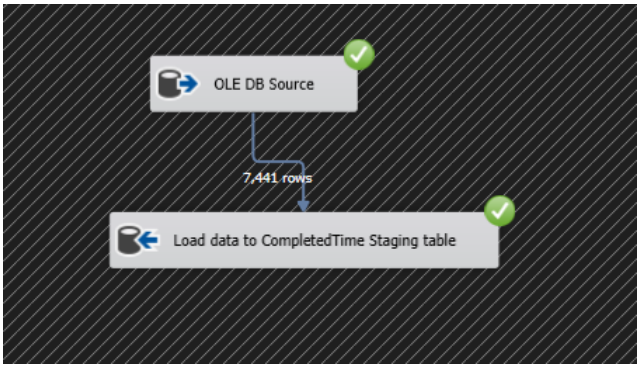


New Generated file



Sample Data

	A	B	C	D
1	FlightNumber	accm_txn_complete_time		
2	20080125X00106	12/31/2007 8:41		
3	20080206X00141	12/31/2007 9:41		
4	20080129X00122	12/30/2007 5:41		
5	20080114X00045	12/30/2007 5:41		
6	20080109X00032	12/30/2007 4:41		
7	20080129X00118	12/29/2007 7:41		
8	20080214X00193	12/29/2007 8:41		
9	20080215X00200	12/29/2007 8:41		
10	20071231X02014	12/29/2007 5:41		
11	20080103X00010	12/29/2007 6:41		
12	20080117X00071	12/29/2007 8:41		
13	20080111X00041	12/28/2007 9:41		


OLE DB Source
7,441 rows
Load data to CompletedTime Staging table