Contents

[Introduction 2](#_Toc72594286)

[Dataset 2](#_Toc72594287)

[Reasons for Selecting the Dataset 4](#_Toc72594288)

[Stakeholders 4](#_Toc72594289)

[Business Vision 4](#_Toc72594290)

[Insights and Reports 5](#_Toc72594291)

[Normalisation, ERD & EERD 6](#_Toc72594292)

[Schema 7](#_Toc72594293)

[Model Schema 7](#_Toc72594294)

[Application Schema 7](#_Toc72594295)

[ETL (Extract, Transform, Load) 8](#_Toc72594296)

[Extract 8](#_Toc72594297)

[Main Source 8](#_Toc72594298)

[External Source 11](#_Toc72594299)

[Transformation 13](#_Toc72594300)

[Source Layer 13](#_Toc72594301)

[Aggregation Layer 14](#_Toc72594302)

[Load 15](#_Toc72594303)

[Reporting Layer 19](#_Toc72594304)

[Reports 20](#_Toc72594305)

[SSRS 20](#_Toc72594306)

[Tableau 25](#_Toc72594307)

[Graph Database - Neo4j 31](#_Toc72594308)

[Node Creation Queries 31](#_Toc72594309)

[Creating indexes on all the nodes 32](#_Toc72594310)

[Creating relationship between routes and airlines 32](#_Toc72594311)

[SQL and CQL Comparison Queries 33](#_Toc72594312)

[Conclusion 40](#_Toc72594313)

[Appendix 41](#_Toc72594314)

[SSIS Project Screenshots 41](#_Toc72594315)

# Introduction

In the current day and age database systems have become a part and parcel of every single company, may it be IT, Technology, or otherwise. Data has become akin to fuel to these corporations and the database systems are like engines that utilise this fuel. We can use these systems to develop insights as well as gain business intelligence from our data, but this can only be done if the data is stored and maintained in an organised manner.

Many such methods exist but the ones that we will be focusing on as part of this assignment will be Relational and Graph databases. We will be exploring how to create, define, manipulate, query as well as generate insights and reports from our data.

# Dataset

Considering a flight and airport dataset of multiple airlines and countries from openflights.org around the world up to 2014. The data contains over 7000 airports, 6000 airlines, 66000 routes (approx.) and a country list of ISO 3166-1 country codes. An external dataset from a different source has been included to help with the analysis of data. The multiple tables and their attributes will be represented below,

**Source Table 1:** airlines.dat

|  |  |
| --- | --- |
| 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. See Countries to cross-reference to ISO 3166-1 codes.* |
| Active | *"Y" if the airline is or has until recently been operational, "N" if it is defunct. This field is not reliable: in particular, major airlines that stopped flying long ago, but have not had their IATA code reassigned (eg. Ansett/AN), will incorrectly show as "Y".* |

**Source Table 2:** countries.dat

|  |  |
| --- | --- |
| 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.* |

**Source Table 3:** airports.dat

|  |  |
| --- | --- |
| Airport ID | *Unique OpenFlights identifier for this airport.* |
| Name | *Name of airport. May or may not contain the City name.* |
| City | *Main city served by airport. May be spelled differently from Name.* |
| Country | *Country or territory where airport is located. See Countries to cross-reference to ISO 3166-1 codes.* |
| 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, eg. India is 5.5.* |
| DST | *Daylight savings time. One of E (Europe), A (US/Canada), S (South America), O (Australia), Z (New Zealand), N (None) or U (Unknown).* |
| Tzdatabasetimezone | *Timezone in "tz" (Olson) format, eg. "America/Los\_Angeles".* |
| Type | *Type of the airport. Value "airport" for air terminals, "station" for train stations, "port" for ferry terminals and "unknown" if not known. In airports.csv, only type=airport is included.* |
| Source | *Source of this data. "OurAirports" for data sourced from OurAirports, "Legacy" for old data not matched to OurAirports (mostly DAFIF), "User" for unverified user contributions*. |

**Source Table 4:** routes.dat

|  |  |
| --- | --- |
| Airline | *2-letter (IATA) or 3-letter (ICAO) code of the airline.* |
| Airline ID | *Unique OpenFlights identifier for airline (see Airline).* |
| Source airport | *3-letter (IATA) or 4-letter (ICAO) code of the source airport.* |
| Source airport ID | *Unique OpenFlights identifier for source airport (see Airport)* |
| Destination airport | *3-letter (IATA) or 4-letter (ICAO) code of the destination airport.* |
| Destination airport ID | *Unique OpenFlights identifier for destination airport (see Airport)* |
| Codeshare | *"Y" if this flight is a codeshare (that is, not operated by Airline, but another carrier), empty otherwise.* |
| Stops | *Number of stops on this flight ("0" for direct)* |
| Equipment | *3-letter codes for plane type(s) generally used on this flight, separated by spaces* |

**Source Table 5:** planes.dat

|  |  |
| --- | --- |
| Name | *Full name of the aircraft.* |
| IATA | *Unique three-letter IATA identifier for the aircraft.* |
| ICAO | *Unique four-letter ICAO identifier for the aircraft.* |

**External Data Source:**

**Source Table 1:** airline\_safety.csv

|  |  |
| --- | --- |
| Airline | *Airline (asterisk indicates that regional subsidiaries are included)* |
| avail\_seat\_km\_per\_week | *Available seat kilometers flown every week* |
| incidents\_85\_99 | *Total number of incidents, 1985–1999* |
| fatal\_accidents\_85\_99 | *Total number of fatal accidents, 1985–1999* |
| fatalities\_85\_99 | *Total number of fatalities, 1985–1999* |
| incidents\_00\_14 | *Total number of incidents, 2000–2014* |
| fatal\_accidents\_00\_14 | *Total number of fatal accidents, 2000–2014* |
| fatalities\_00\_14 | *Total number of fatalities, 2000–2014* |

# Reasons for Selecting the Dataset

OpenFlights dataset offers a large amount of readily available, appropriate and relevant data as well as scope for in-depth analysis on multiple attributes and available external sources that also contribute and provide various angles for analysis.

# Stakeholders

1. **Airline Companies**: To analyse routes and reach to countries and customers.
2. **Frequent Flyers**: To gain knowledge about airlines, routes, and safety standards.
3. **Plane Manufacturers**: To analyse past safety incidents and gain info on what types of planes are flying particular routes.
4. **Travel Companies**: To analyse routes and airlines as well as gain knowledge about countries and cities and the distance of routes.
5. **Air Travel Authorities**: To analyse the traffic between routes and airports to minimize air traffic delays.

# Business Vision

Provide in-depth and separate analysis from all angles of data and provide accurate and relevant insights into our data and domain for relevant stakeholders in a readable and presentable format.

# Insights and Reports

The insights will be divided into 2 parts with 4 points of analysis each. The first part will be included as SSRS reports with the latter as a Tableau Dashboard.

SSRS:

* Most active airline on basis of unique routes [Non-Parameterised]
* Most popular plane model used across all airlines [Non-Parameterised]
* Report of all the routes flown by a particular airline [Parameterised]
* Report of all the flights coming in and going out of a particular country [Parameterised]

Tableau:

* Analysing air traffic between routes
* Analysing traffic for airports
* Analysing safest and most dangerous airlines across year range
* Most distance covered by airlines

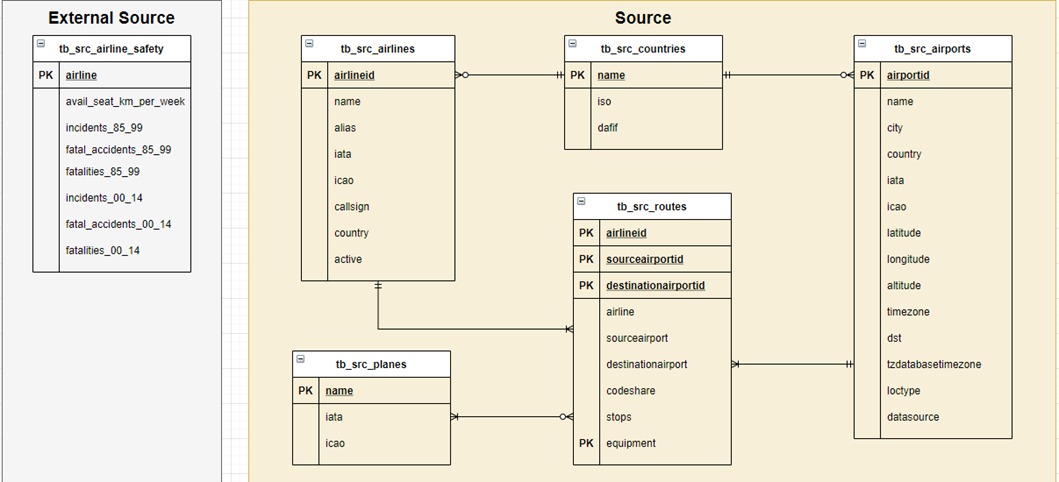
# Normalisation, ERD & EERD

**Normalisation**

As our tables are already divided into multiple files with only data relevant to the particular columns present in the table right from the data source, it was neither possible nor needed to normalise the data further as no transitive or partial dependencies were present and the data source itself was already in the 3-NF (3rd Normal Form)

**ERD**

Entity Relationship Diagram for the source tables and the external table are illustrated below,



Created on draw.io

In the source table the relations are,

* Airlines to Countries: An airline can only have one country while countries can have zero or more airlines that belong to them.
* Countries to Airports: An airport can only have one country while countries can have zero or more airports in them.
* Airports to Routes: An airport can have multiple routes while routes can one and only one airport.
* Routes to Airlines: A route can have only one airline while airlines can have multiple routes.
* Routes to Planes: A route can have multiple types of planes and planes can also belong to many routes. To remove this many-to-many relationship, an associative table has been added in the dimension layer.

The external table has no relations as it consists of only one table.

**EERD**

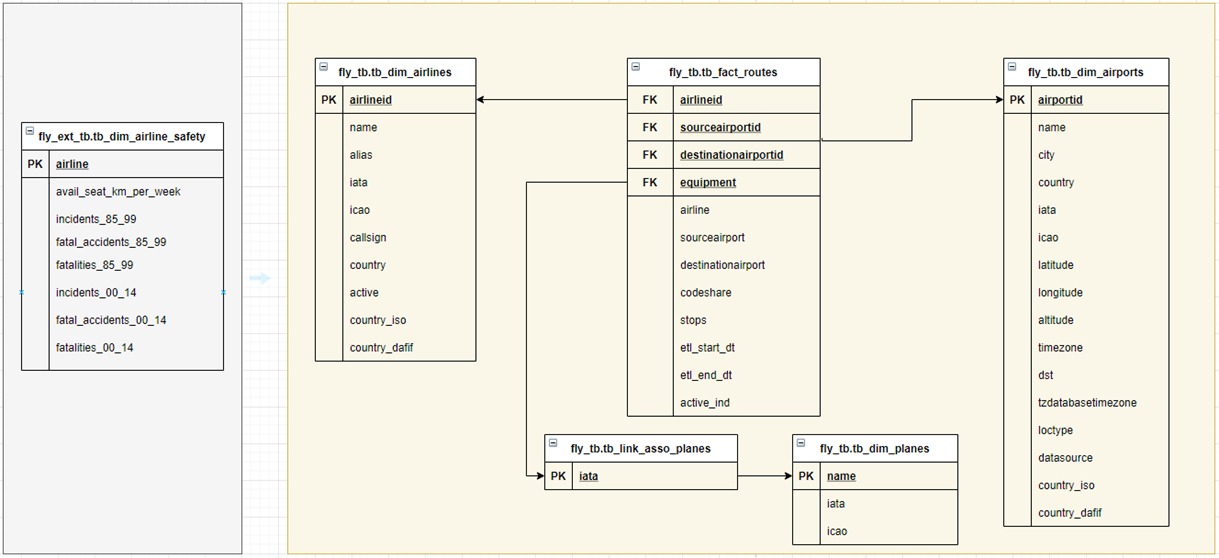
The tables cannot be further generalized or specialized.

# Schema

## Model Schema

The schema most suitable to our data is the snowflake schema as all the tables are normalised and because of the use of the associative table to remove the many-to-many relationship between the planes and routes table, the planes table is no longer able to directly connect to the routes fact table.

The external table has no schema as it is a singular table that cannot be further divided into dimensions or facts.



## Application Schema

We have created separate application schemas for segregating each of our areas of database objects like tables, views, and user-defined stored procedures. The purpose of creating an application schema is to restrict the access given to a particular role, and to implement the importance of DCL statements like GRANT and REVOKE. We have created [source: fly\_db, external source: fly\_ext\_db] for database [source: fly\_tb, external source: fly\_ext\_tb] for tables, [source: fly\_vw, external source: fly\_ext\_vw] for views and [source: fly\_sp] for the user-stored procedures.

# ETL (Extract, Transform, Load)

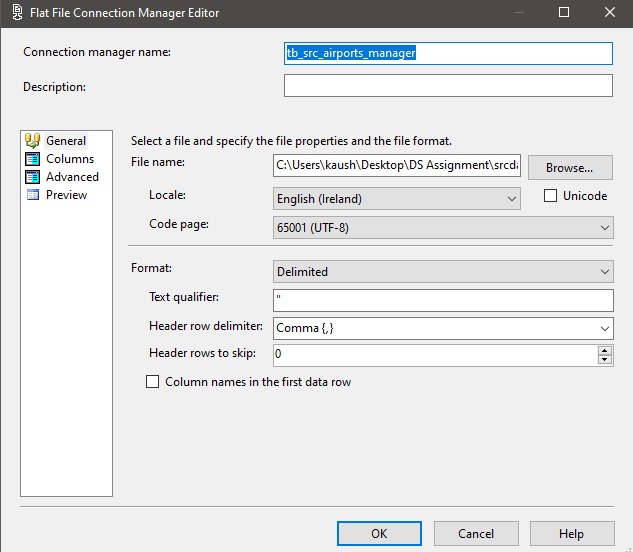
## Extract

### Main Source

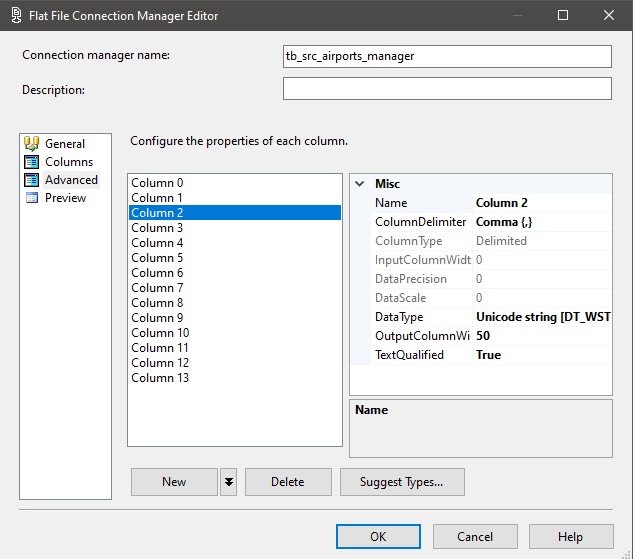
To extract our data from the [.dat] file sources to Microsoft SQL Server, we will be using SQL Server Integration Services (SSIS) in Microsoft Visual Studio for the Main Source Data and use the SSMS Import Wizard for our External Source Data.

The extraction process for airports.dat will be demonstrated in below steps, all other files were extracted in the same process and thus we are demonstrating only one table to avoid redundancy.

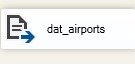
First, we selected the file we want to extract our data from in the flat file connection manager editor. This connection manager is later used in the flat file source editor.

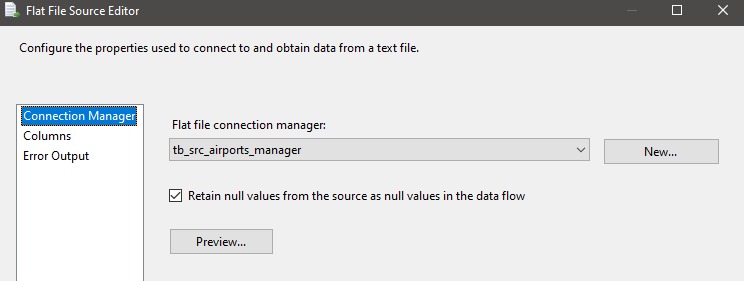


Further, we changed the data types of some of the columns in the flat file connection manager editor to appropriately match the data. For our data certain columns did not accept the default data type, string [DT\_STR], this is because of the presence of non-English characters in our data (like. ä, ö, ü) so we changed our column data type to Unicode string [DT\_WSTR] to accommodate the data and prevent data loss.



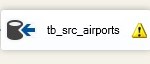
In the next step, we set up our Flat file source object in our Data Flow window and selected its connection manager as one we created earlier.



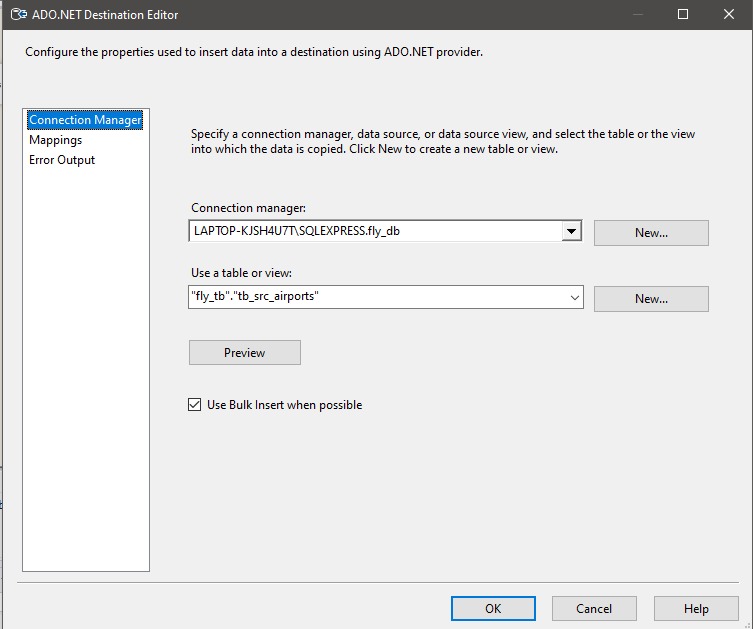


Since we need all the columns from our source file, we do not make any changes in the columns section of our Flat File Source Editor.

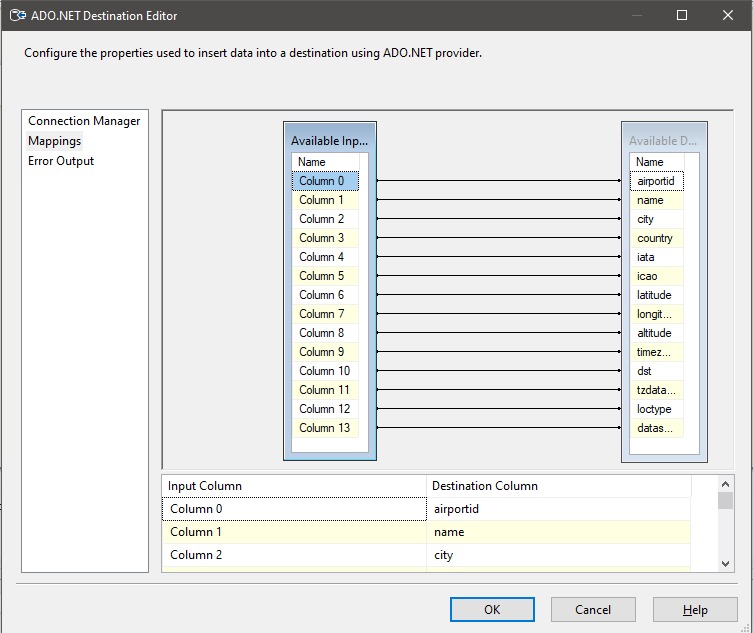
Now we set up our Destination Editor to export our data into the table we want to populate in SQL Server, and for this we use the ADO .NET Destination Editor Object.



We add the connection manager of the database we want to add the data to in the Connection Manager section, in this case the fly\_db database and add the table we want it loaded to, in this case [fly\_tb.tb\_src\_airports].



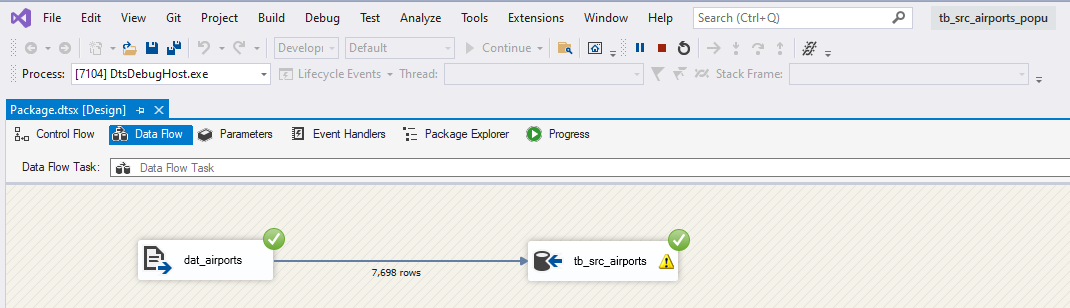
We also add the mappings for all the columns between the source and the target.



The final data flow looks like the image below,

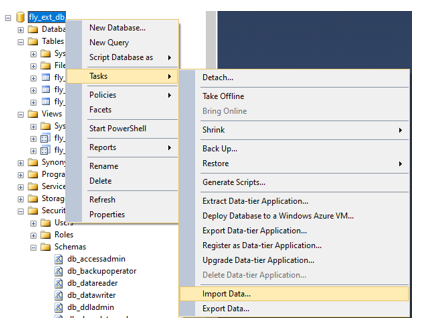


We start the workflow for SSIS to begin populating our table. We repeated these steps for all the other files to load all the tables from our main source.

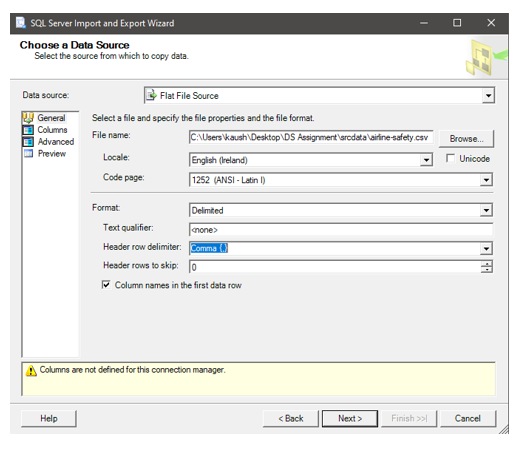


### External Source

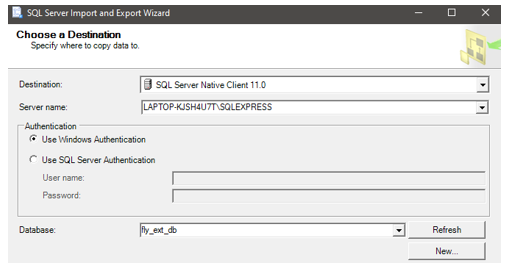
For loading the data from our external source to the [fly\_ext\_db] database, we used the SQL Server Import Data Wizard. The wizard is accessed by right clicking the database name and going into the ‘Tasks’ section.



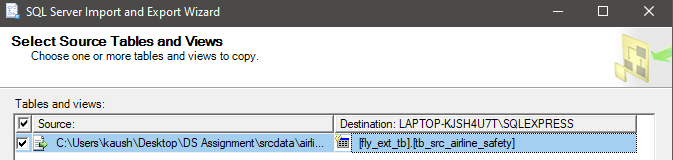
We select the source path, qualifiers and delimiters in the Data Source section.



Post that we select the destination server and Database, in this case our [fly\_ext\_db] database.



Confirm the source and destination tables, then click ‘Finish’ to begin importing the data. The tables should be populated shortly.



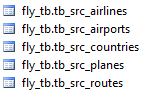
This concludes the extraction of our data from our Main Source and External Source.

## Transformation

### Source Layer

All the flat file source data is added to the [fly\_db] database and, [fly\_ext\_db] under the schema fly\_tb and fly\_ext\_tb.

fly\_db:

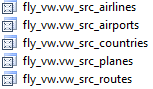


fly\_ext\_db:



For data security and integrity, we added non-updatable views on top of these tables, as the data was ingested in the VARCHAR datatype and later depending on the data format, we casted the datatypes accordingly in the views.

fly\_db:



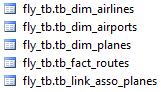
fly\_ext\_db:



This completes our list for all the objects in our source layer, and we move further to develop the aggregation layer. In moving the data from the source to the aggregation layer, some transformations were made to better suit our warehouse schema and as per our design of the dimensional model. These changes include ingesting our countries table into airlines and airports table and adding an associative table to link the planes and routes table and eliminate the many-to-many relationship between them.

### Aggregation Layer

fly\_db:

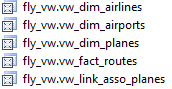


fly\_ext\_db:



Like the previous layer updatable views, instead of non-updatable views were created for this layer for data security and integrity reasons.

fly\_db:



fly\_ext\_db:



This concludes the transformation of our data.

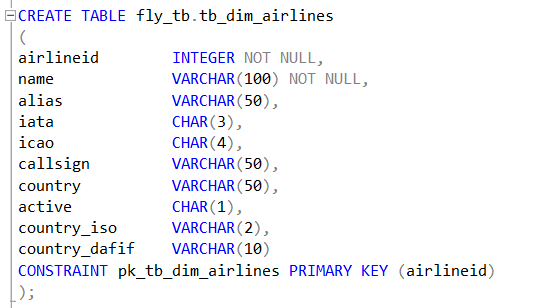
## Load

After the transformation queries were created, the data is inserted from the source views into the dimension tables developed in the transformation process. The loading is accomplished with the help of SQL queries in SQL Server.

The queries for the loading process are as follows,

Airlines Dimension:

Table Creation

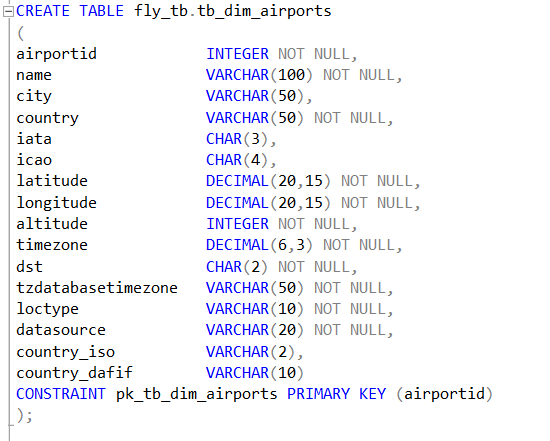


Data Insert

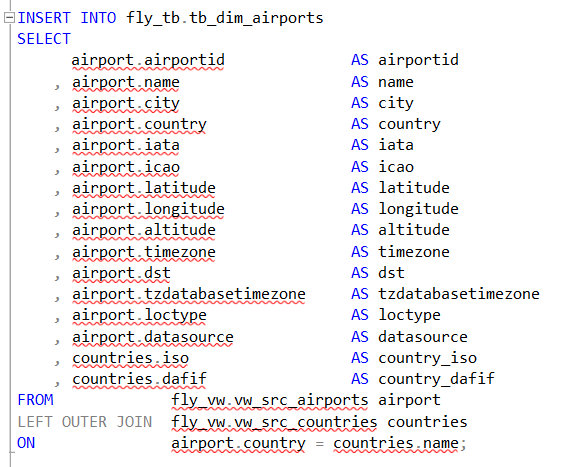


Airports Dimension:

Table Creation

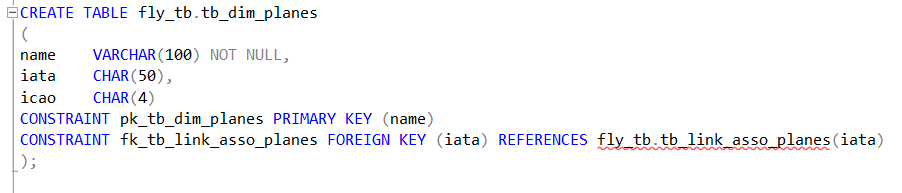


Data Insert

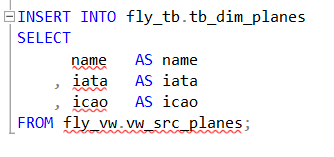


Planes Dimension

Table Creation

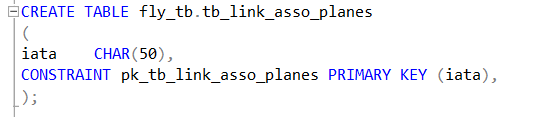


Data Insert

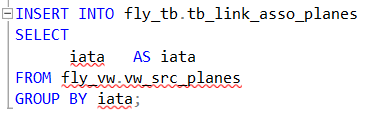


Planes Association Table:

Table Creation

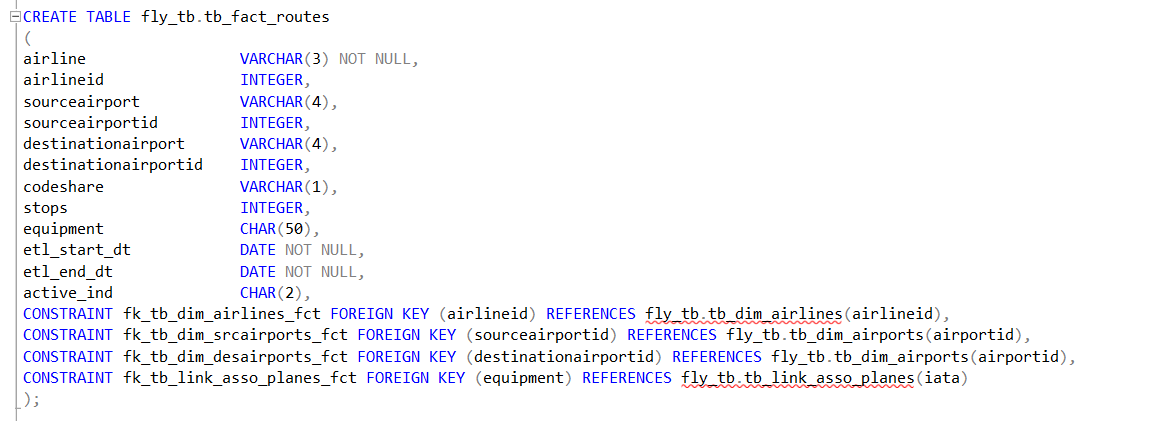


Data Insert

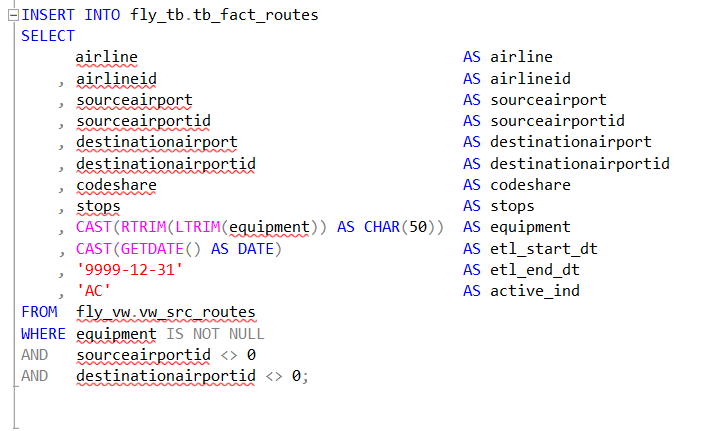


Routes Fact:

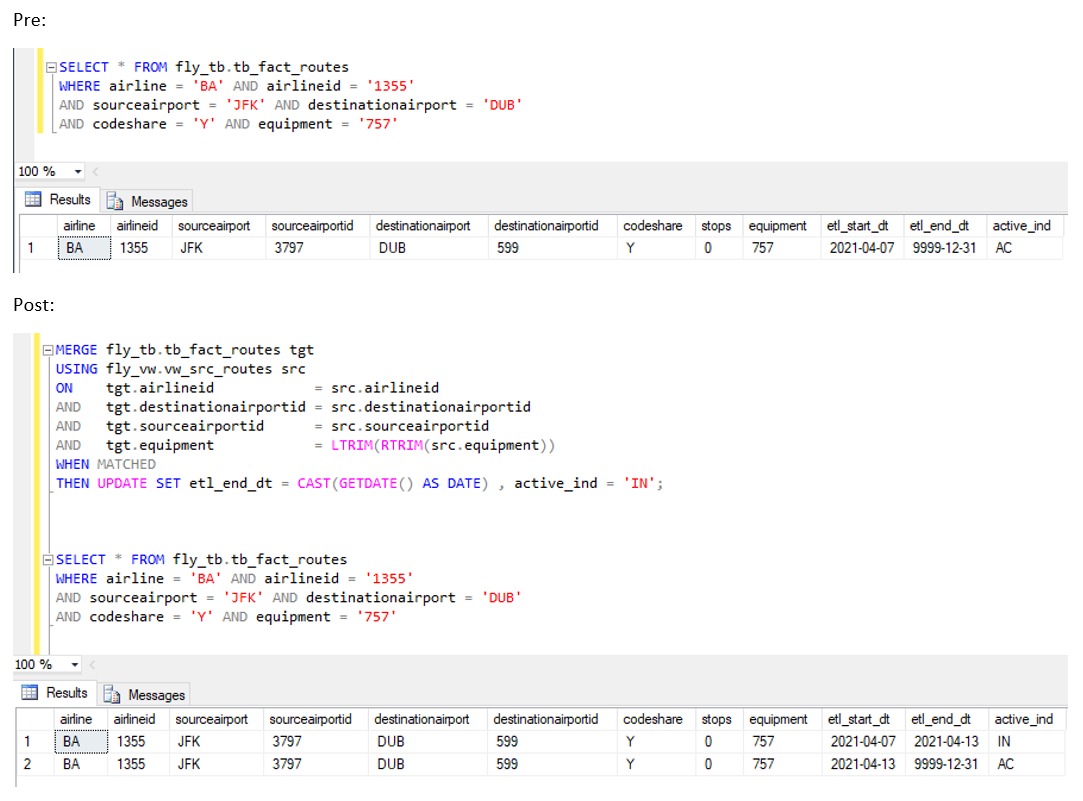
Table Creation



Data Insert

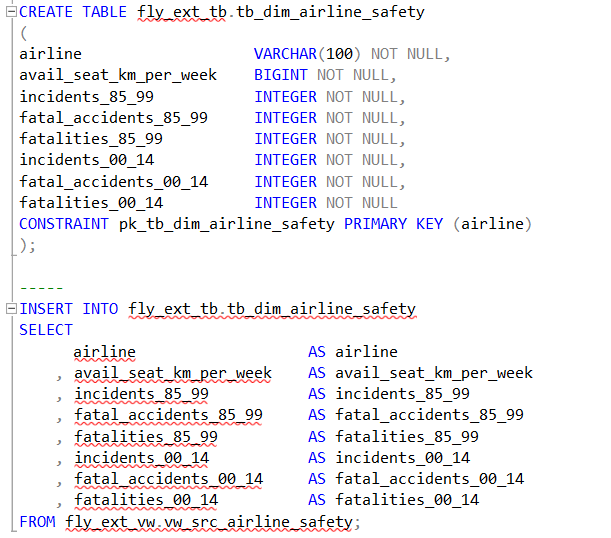


The above query was used to populate the routes fact table, and to implement SCD (Slowly Changing Dimensions) Type-2 into the table we used the below MERGE query, for the purpose of maintaining data integrity and mark older records based on the key columns.

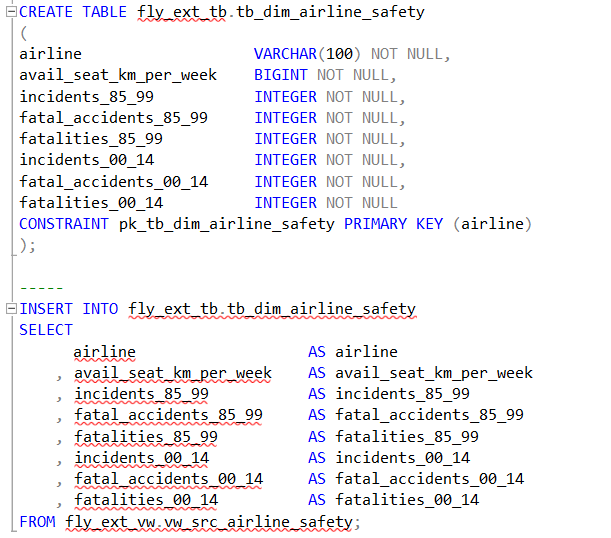


Externally Sourced Airline Safety Dimension:

Table Creation



Data Insert



### Reporting Layer

For the creation of reports another layer was added to the database - the reporting layer, where the data for the tableau reports as well as the stored procedures for the SSRS reports where stored.





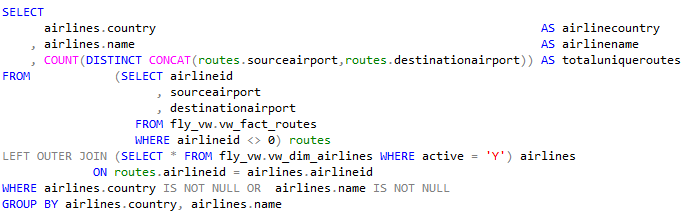


# Reports

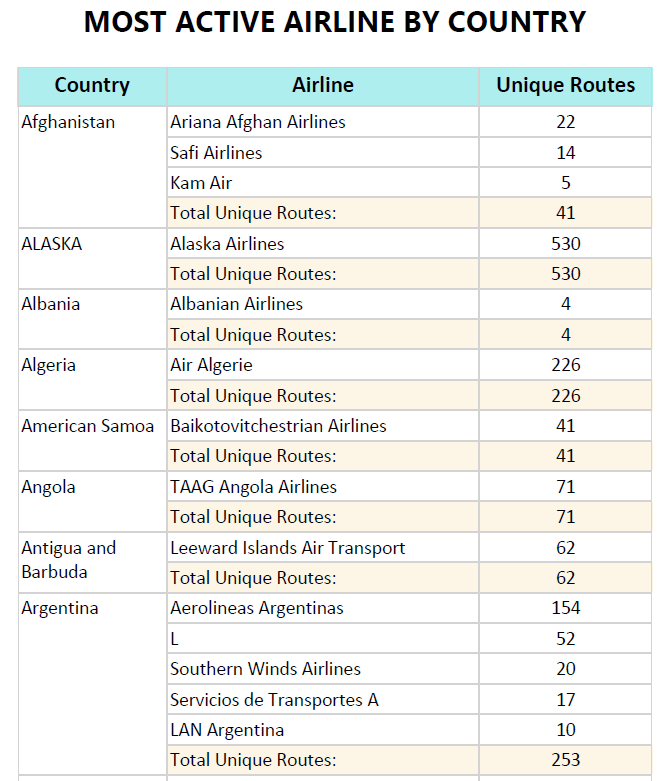
## SSRS

**Count the number of unique routes existing for each airline in each country**

Query:



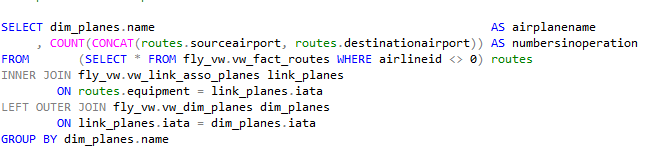
Report:



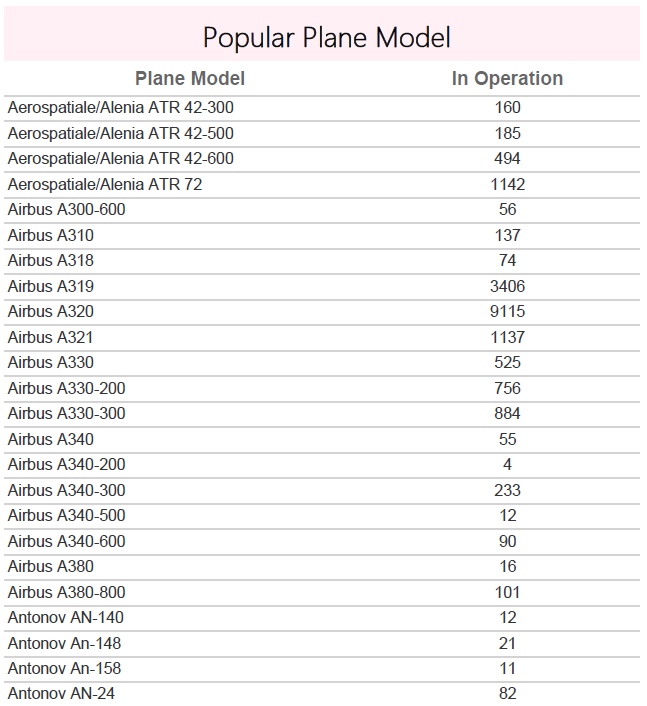
Notes: Analysing the total amount of unique routes in a country can help a business identify if enough of the country area is being covered or if there is a need or demand for more routes. The unique routes by airlines can give businesses a gauge of the competition in a country and if it can be worth it to expand their routes to a country with a low number of airlines that operate from the country.

**Popularity of plane models in operation**

Query:



Report:



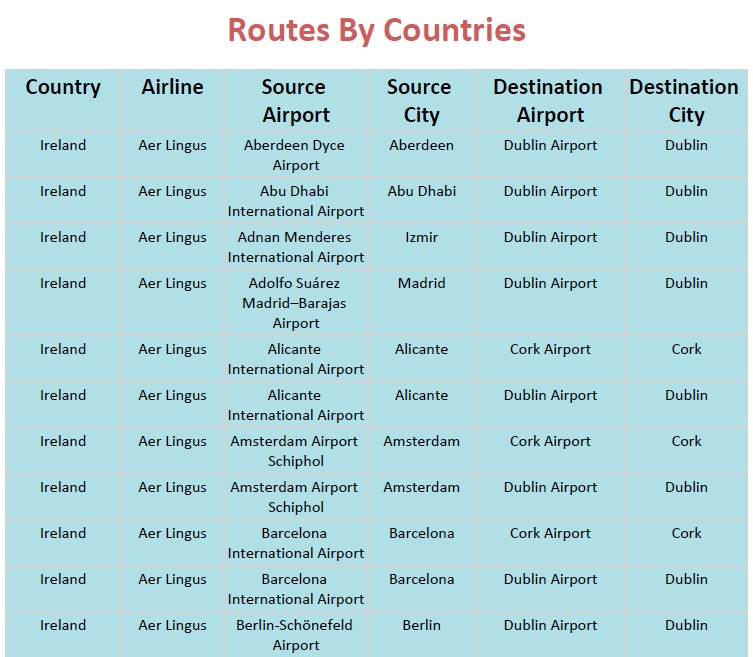
Notes: This report shows the popularity of plane models among airlines and routes. The popularity of a plane model can show a manufacturer or airline what are some desirable features in these models and analyse the reason for their popularity such as capacity, mileage etc. for existing and newer models to focus on, when manufacturing/acquiring.

**Parameterised report to get all flights in and out of a country**

Query:



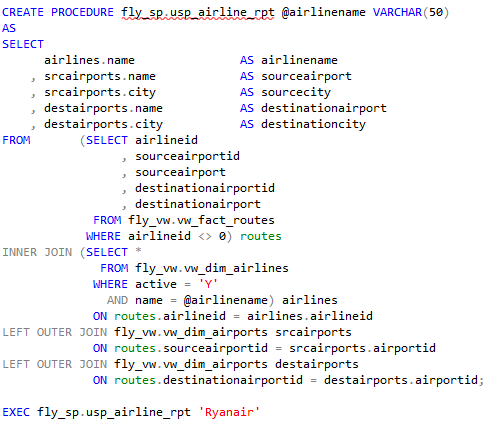
Report:



Notes: This report gives all flight details with the specified country parameter. The report includes information of flights going in/out from all airports located within the country. This analysis can help frequent flyers and travellers from a country to get information and conduct research on all incoming and outgoing flights to better plan their travels.

**Parameterised report to get flight details for an airline**

Query:



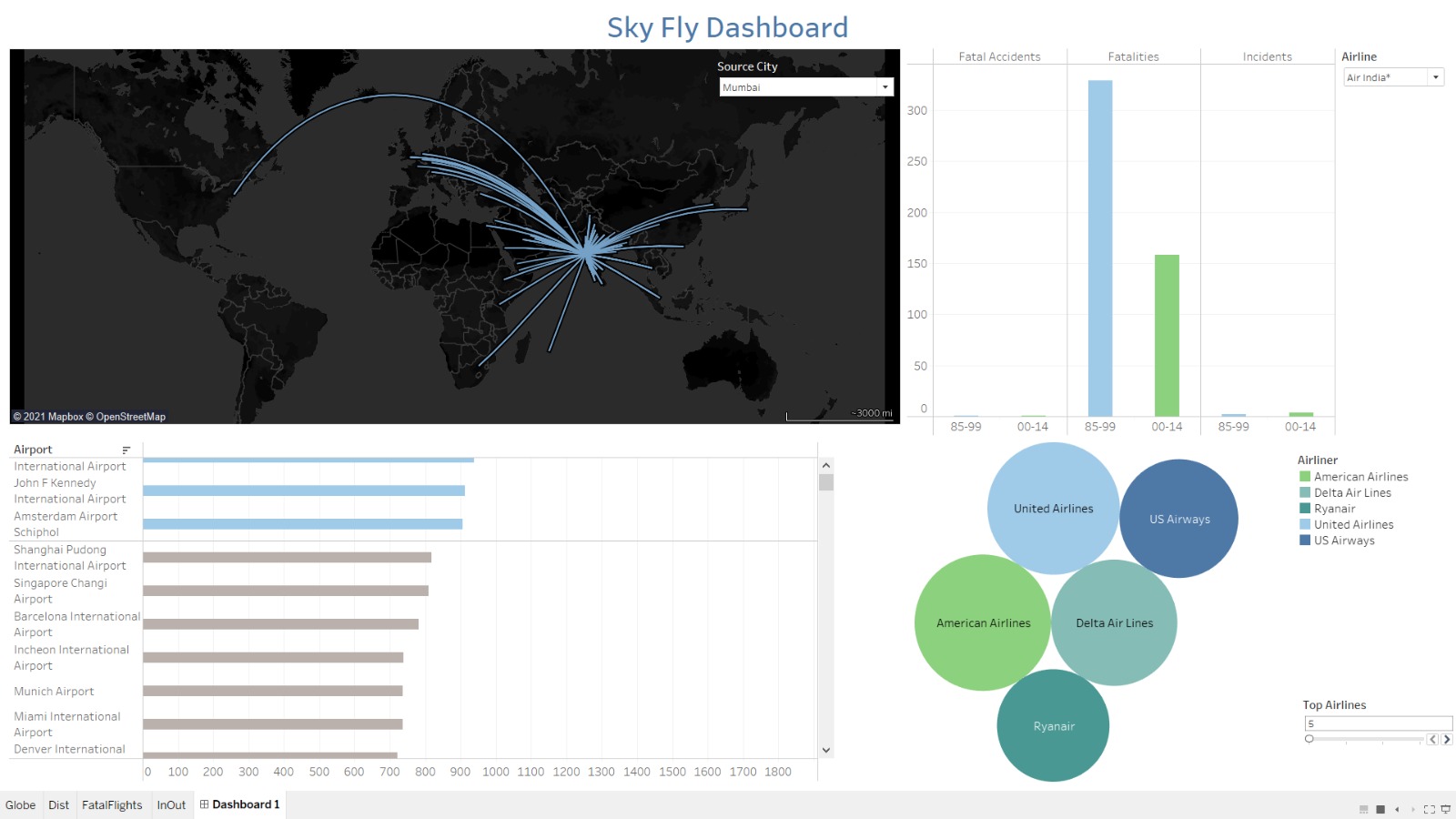
Report:



Notes: It is important for any airline or business to get a report, to analyse things such as coverage and number of flights as well as unique routes and airports and cities covered.

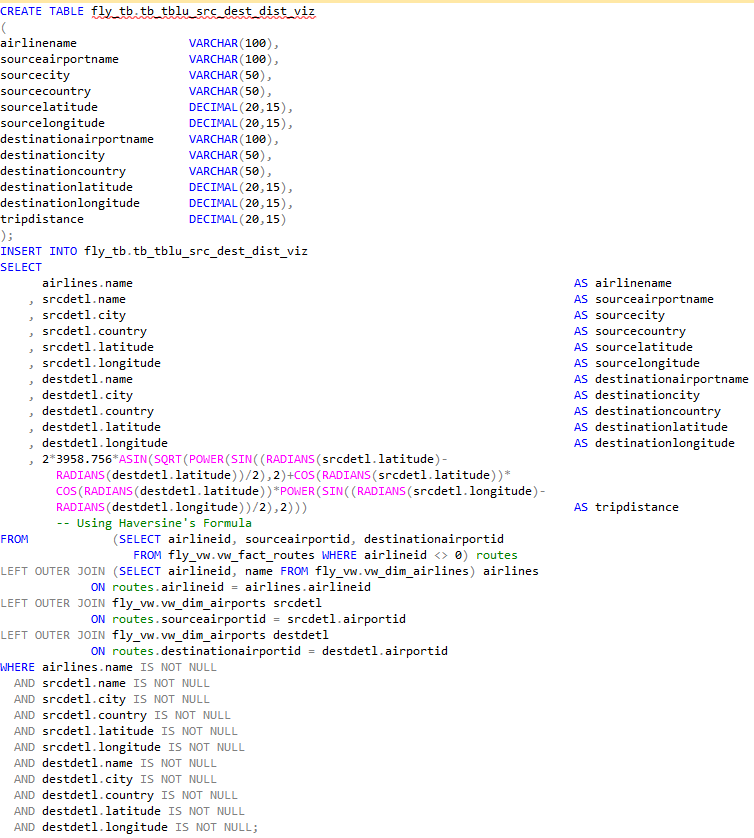
## Tableau

Complete Dashboard: [Sky Fly Dashboard]

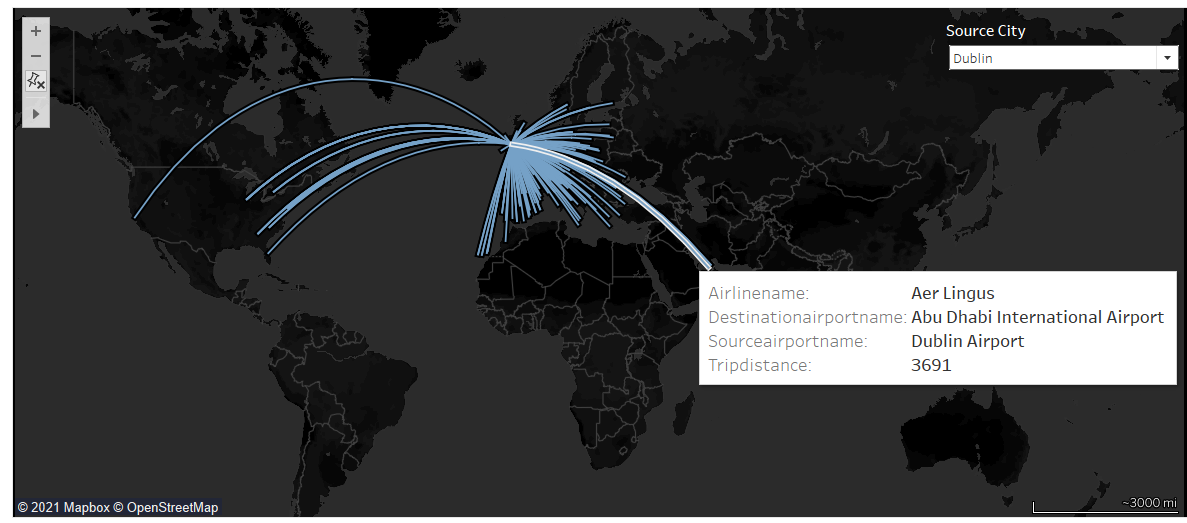


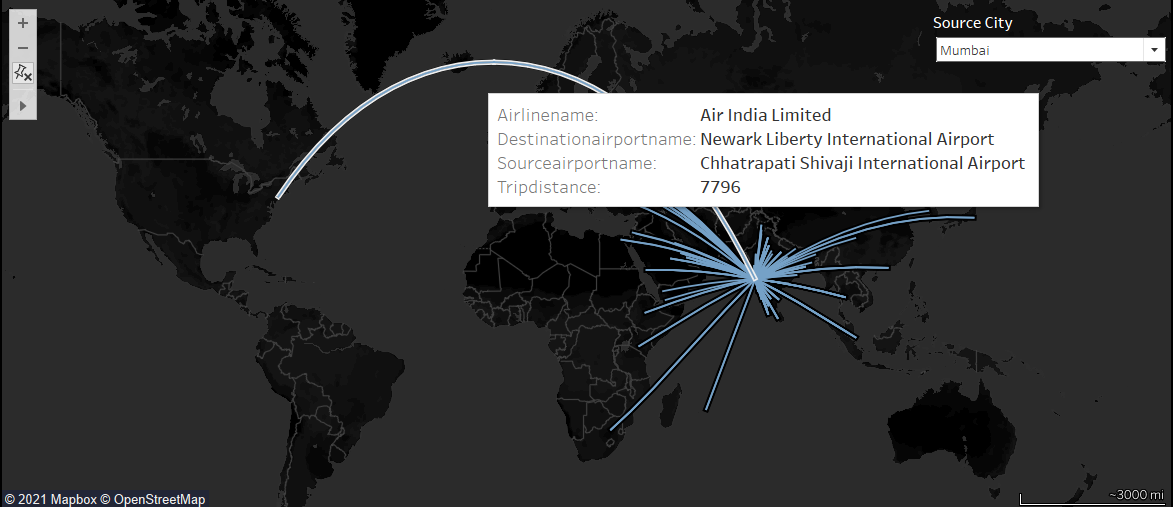
**Interactive report of outgoing routes from a city**

Query:



Report:

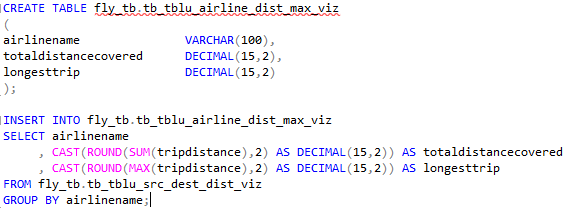




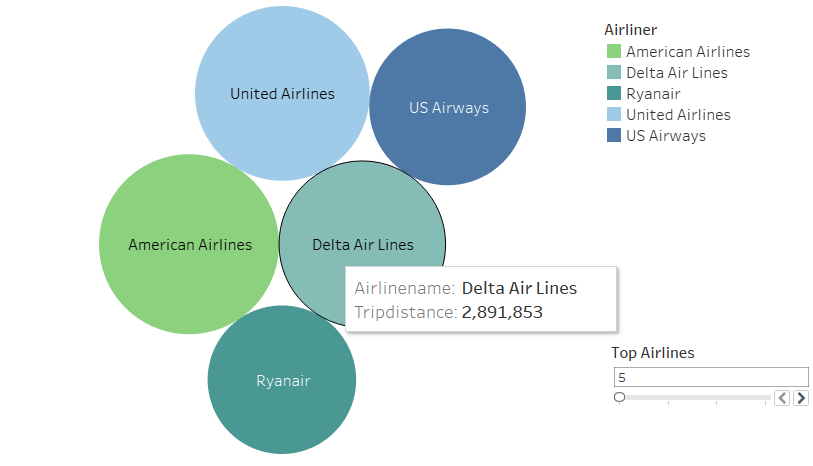
Interactive report with drop down menu that lets users select source city to see all outgoing flights from selected city, and the distance to the destination and their locations on the map. Also lets airlines check for coverage and reach of cities to know potential routes from the city if they do not yet exist. The distance between the cities has also been calculated [using Haversine’s Formula].

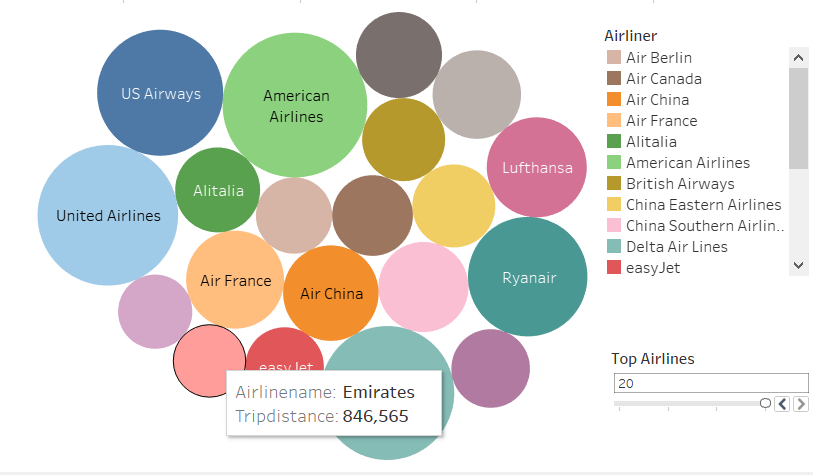
**Interactive Packed bubble chart of total distance covered by top airlines**

Query:



Report:

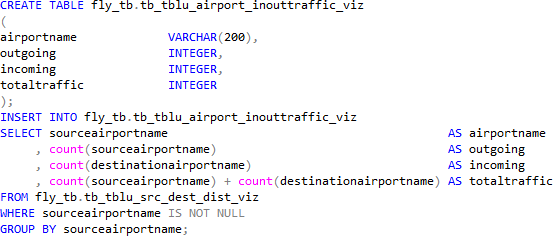




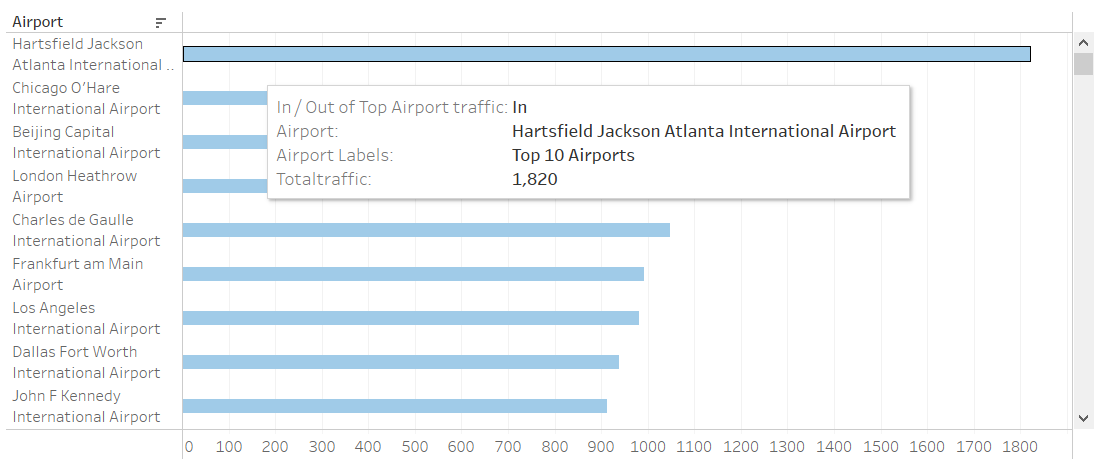
Packed bubble chart lets top airlines compare their standings on total distance covered by their airlines and how other top airlines (up to top 20) compared to them and analyse whether they want their distance covered to be increased or not and is also useful for comparing with other factors and analysis. The total distance covered is calculated as a SUM of trip distance calculated as part of the previous report.

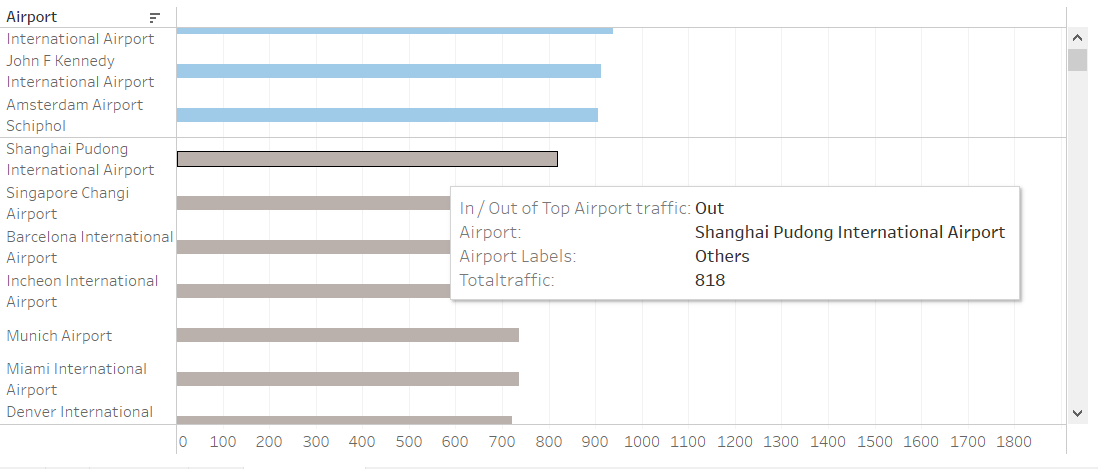
**Total air traffic of top 100 airports in the world**

Query:



Report:

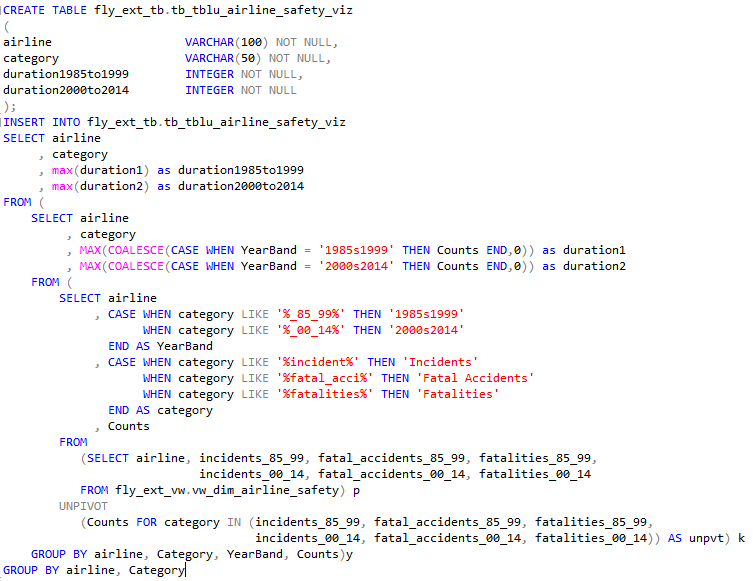




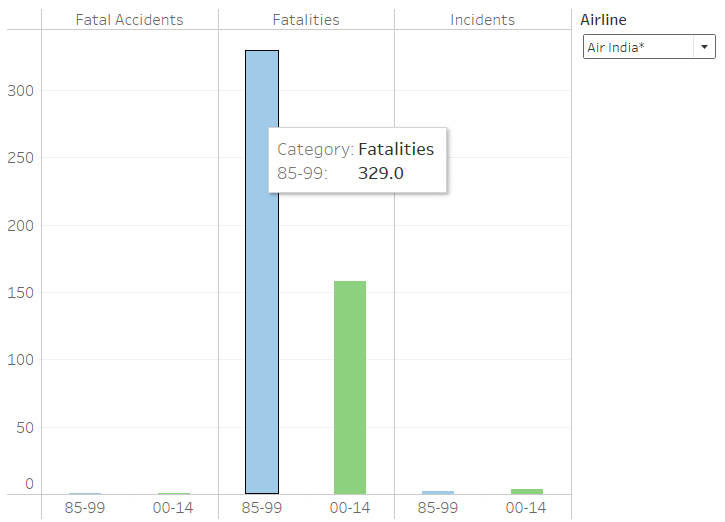
Bar chart comparing air traffic of top 100 airports in the world with top 10 airports highlighted in blue with rest being greyed out, also lets airlines and airport authorities know what to expect at airports and prepare accordingly in relation to how busy an airport might be with flight slots and passenger intake compared to other airports.

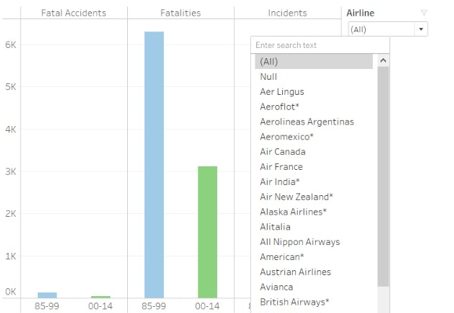
**Interactive airline safety summary report comparing accidents**

Query:



Report:







This report allows airlines and safety cautious flyers to check for statistics on past events that might affect their decision to travel such as fatal accidents, fatalities, and incidents with an interactive drop-down menu that lets users select any airline with any incident as well as a comparison timeframe for when these events occurred to provide additional context to the report. The menu also allows users to view and compare events across all airlines at once and to compare it to any individual airline’s safety history.

# Graph Database - Neo4j

## Node Creation Queries

Queries to create nodes from source files:

1. tb\_airlines node:

LOAD CSV WITH HEADERS FROM 'file:///airlines.csv' AS row

CREATE (n:tb\_airlines) SET n=row

1. tb\_airports node:

LOAD CSV WITH HEADERS FROM 'file:///airports.csv' AS row

CREATE (n:tb\_airports) SET n=row

1. tb\_countries node:

LOAD CSV WITH HEADERS FROM 'file:///countries.csv' AS row

CREATE (n:tb\_countries) SET n=row

1. tb\_planes node:

LOAD CSV WITH HEADERS FROM 'file:///planes.csv' AS row

CREATE (n:tb\_planes) SET n=row

1. tb\_routes node:

LOAD CSV WITH HEADERS FROM 'file:///routes.csv' AS row

CREATE (n:tb\_routes) SET n=row

1. tb\_airline\_safety node:

LOAD CSV WITH HEADERS FROM 'file:///airline-safety.csv' AS row

CREATE (n:tb\_airline\_safety) SET n=row

Creating indexes on all the nodes

CREATE INDEX ON :tb\_airline\_safety(airline)

CREATE INDEX ON :tb\_airlines(airlineid)

CREATE INDEX ON :tb\_airports(airportid)

CREATE INDEX ON :tb\_countries(dafif)

CREATE INDEX ON :tb\_planes(name)

CREATE INDEX ON :tb\_routes(airlineid,sourceairportid,destinationairportid,equipment)

## Creating relationship between routes and airlines

Query to create relationship between tb\_fact\_routes and tb\_dim\_airlines nodes:

MATCH(r:tb\_fact\_routes), (a:tb\_dim\_airlines) WHERE r.airlineid = a.airlineid

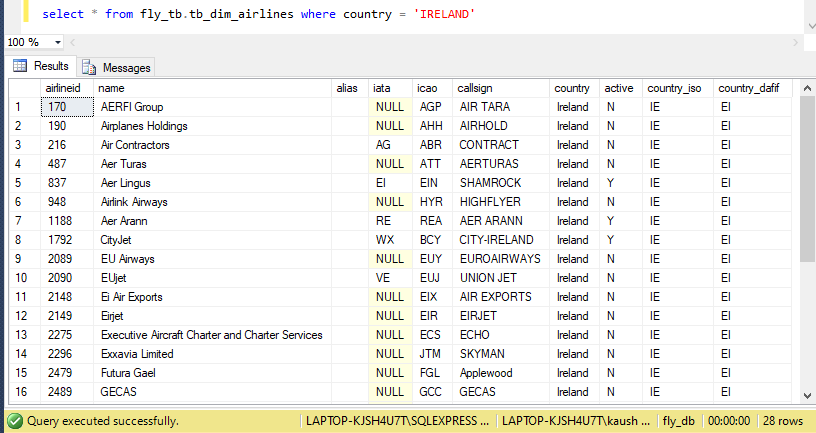
CREATE (a)-[:FLY\_ON]->(r)

## SQL and CQL Comparison Queries

**Query to fetch the airlines details from SQL Server and Neo4j for Ireland**

SQL: SELECT \* FROM fly\_tb.tb\_dim\_airlines WHERE country = 'IRELAND'

CQL: MATCH(n:tb\_airlines) WHERE n.country = 'Ireland' RETURN n.airlineid,n.name,n.icao,n.callsign,n.country,n.country\_iso

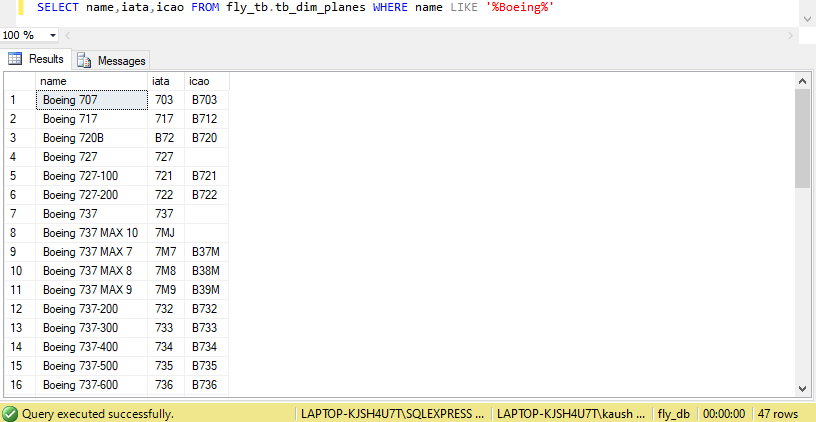


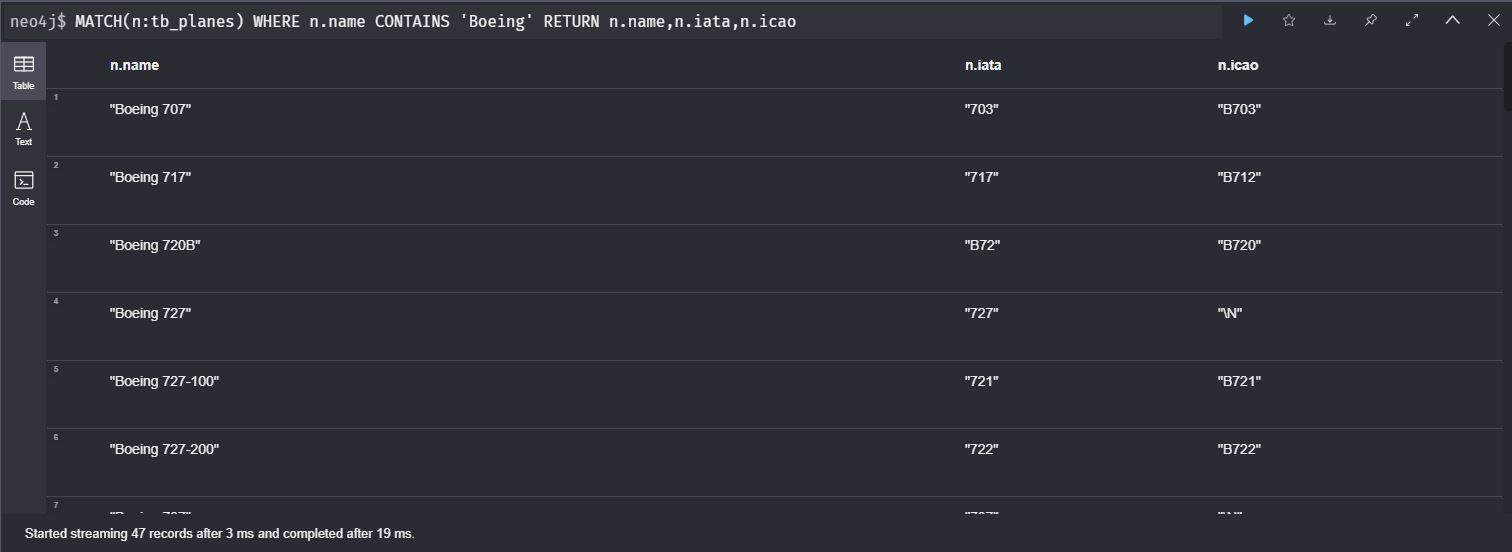


**Query to select plane models which are created by manufacturer Boeing from SQL Server and Neo4j**

SQL: SELECT \* FROM fly\_tb.tb\_dim\_planes WHERE name LIKE '%Boeing%'

CQL: MATCH(n:tb\_planes) WHERE n.name CONTAINS 'Boeing' RETURN n.name,n.iata,n.icao





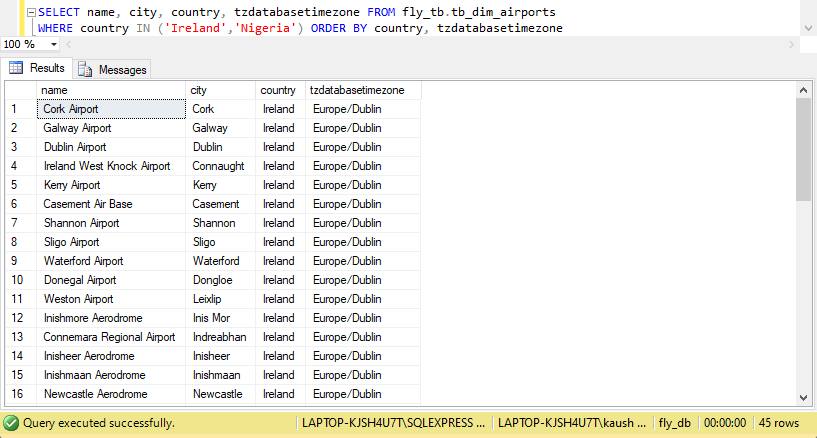
**Query to select name, city, country, and tzdatabasetimezone of all airports in Ireland and Nigeria from SQL Server and Neo4j**

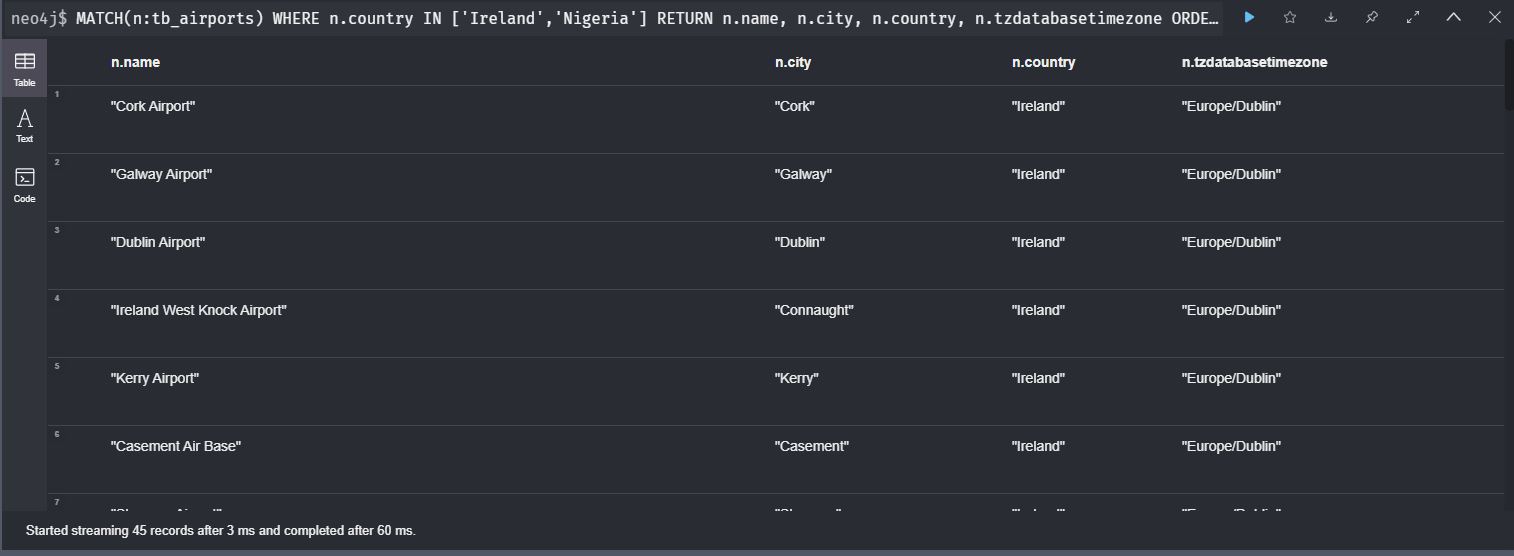
SQL: SELECT name, city, country, tzdatabasetimezone FROM fly\_tb.tb\_dim\_airports

WHERE country IN ('Ireland','Nigeria') ORDER BY country, tzdatabasetimezone

CQL: MATCH(n:tb\_airports) WHERE n.country IN ['Ireland','Nigeria']

RETURN n.name, n.city, n.country, n.tzdatabasetimezone ORDER BY n.country, n.tzdatabasetimezone

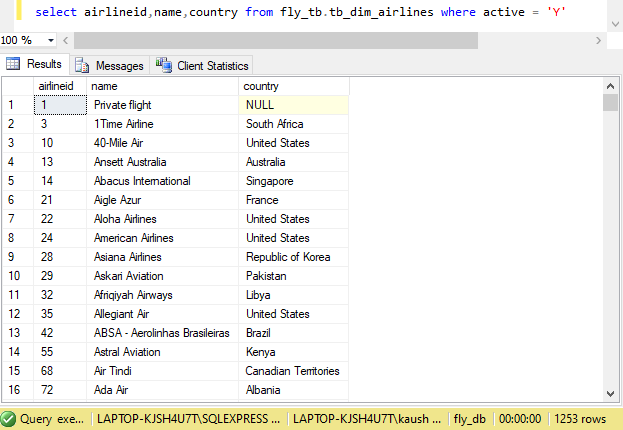




**Query to fetch the airline details from the airline table which are currently active from SQL Server and Neo4j**

SQL: SELECT airlineid, name, country FROM fly\_tb.tb\_dim\_airlines WHERE active = 'Y'

CQL: MATCH(n:tb\_airlines) WHERE n.active = 'Y' RETURN n.airlineid, n.name, n.country

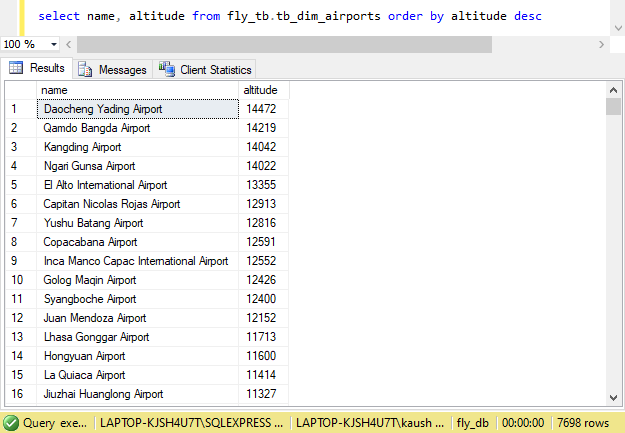




**Query to fetch the name and altitude of the airports ordered by the altitude of the airport in SQL Server and Neo 4j.**

SQL: SELECT name, altitude FROM fly\_tb.tb\_dim\_airports ORDER BY altitude DESC

CQL: MATCH(n:tb\_airports) RETURN n.name, n.altitude ORDER BY toInteger(n.altitude) DESC





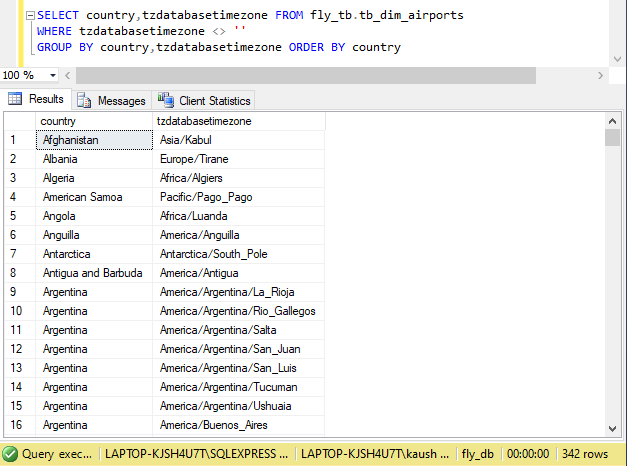
**Query to fetch the countries and their respective timezones where timezone is not an empty field in SQL Server and Neo4j.**

SQL: SELECT country,tzdatabasetimezone FROM fly\_tb.tb\_dim\_airports WHERE tzdatabasetimezone <> ''

GROUP BY country,tzdatabasetimezone ORDER BY country

CQL: MATCH(n:tb\_airports) WHERE n.tzdatabasetimezone <> '\N'

RETURN DISTINCT n.country, n.tzdatabasetimezone ORDER BY n.country

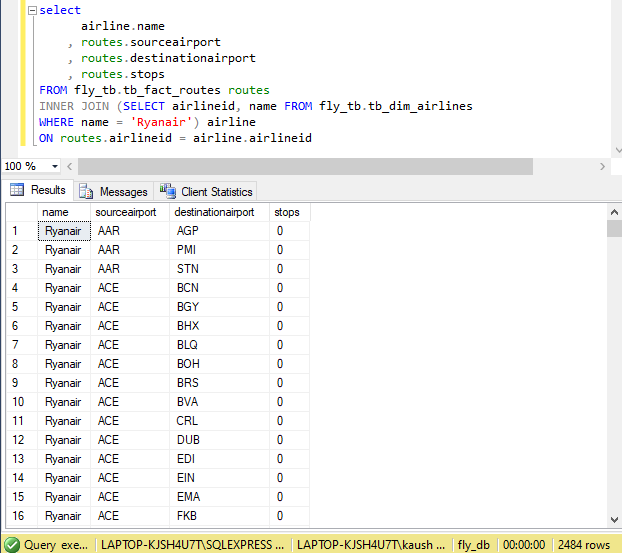


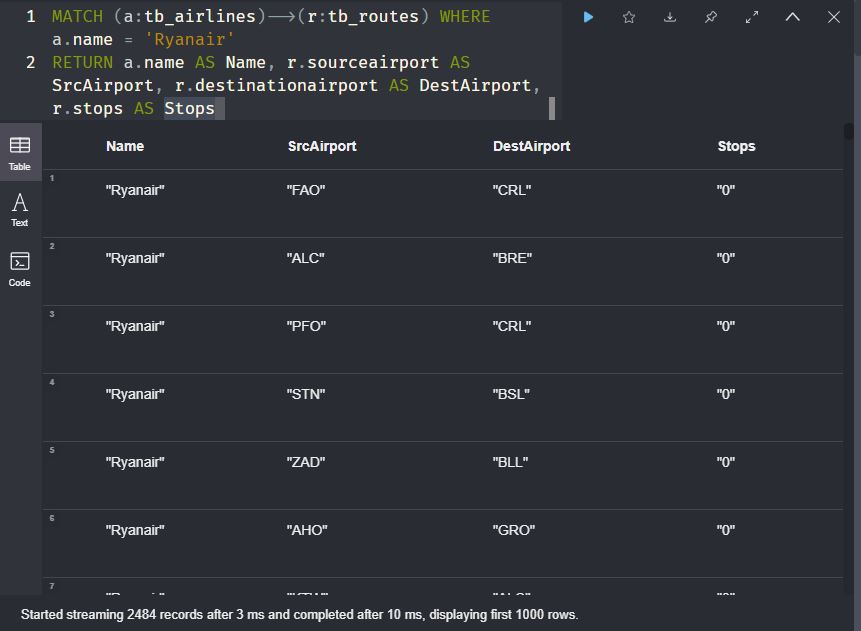


**Query for joining 2 tables (routes and airlines respectively) to fetch all the source and destination airports for an airline (Ryanair in this case) in SQL Server and Neo4j.**

SQL: select airline.name, routes.sourceairport, routes.destinationairport, routes.stops FROM fly\_tb.tb\_fact\_routes routes INNER JOIN (SELECT airlineid, name FROM fly\_tb.tb\_dim\_airlines WHERE name = 'Ryanair') airline ON routes.airlineid = airline.airlineid

CQL: MATCH (a:tb\_airlines)-->(r:tb\_routes) WHERE a.name = 'Ryanair' RETURN a.name AS Name, r.sourceairport AS SrcAirport, r.destinationairport AS DestAirport, r.stops AS Stops





# Conclusion

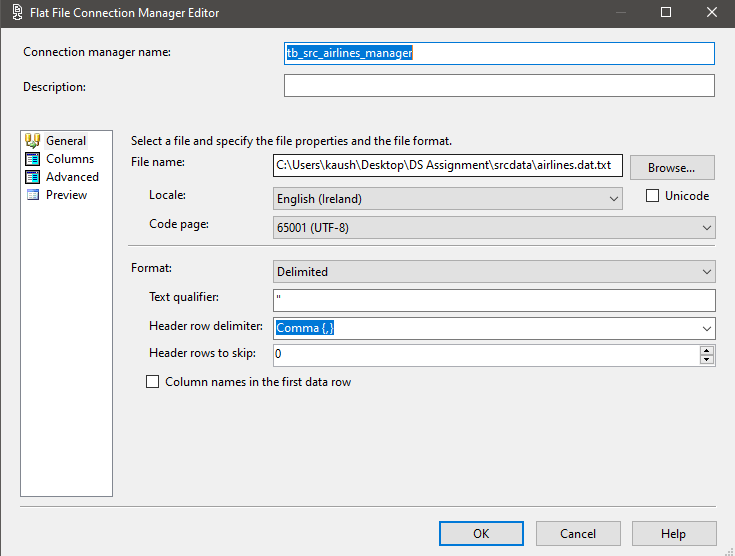
A data warehouse is an essential part of any modern business and its functionality to allow a business to perform analysis and create reports is integral if any growth or success is to be achieved by a business. Through this assignment we believe to have learned the entire process that goes into creating a useful and practical data warehouse that can not only help a business with analysis and reports but is also scalable, secure, and optimised. From inserting the data into the warehouse using multiple methods such as SSIS, SQL queries to using Import Wizard in SQL Server Management Studio. From maintaining history records using SCD methods to create and generating reports using multiple reporting tools such as SSRS and Tableau. We also gained understanding of the operating and functioning of a graph database and were able to query and demonstrate tabular reports of business significance across graph as well as relational databases.

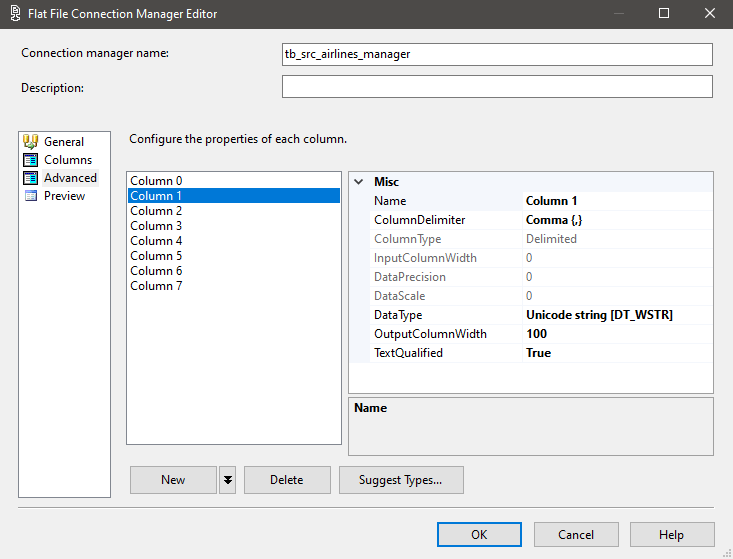
# Appendix

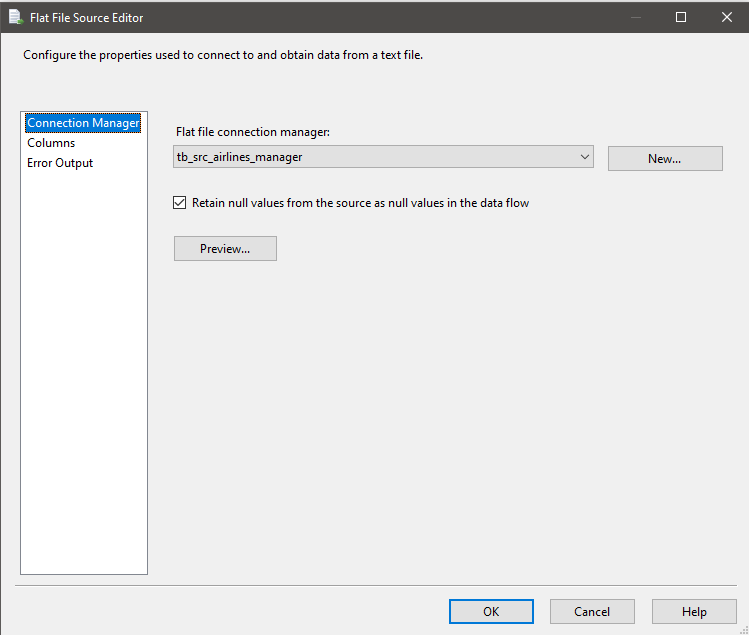
## SSIS Project Screenshots

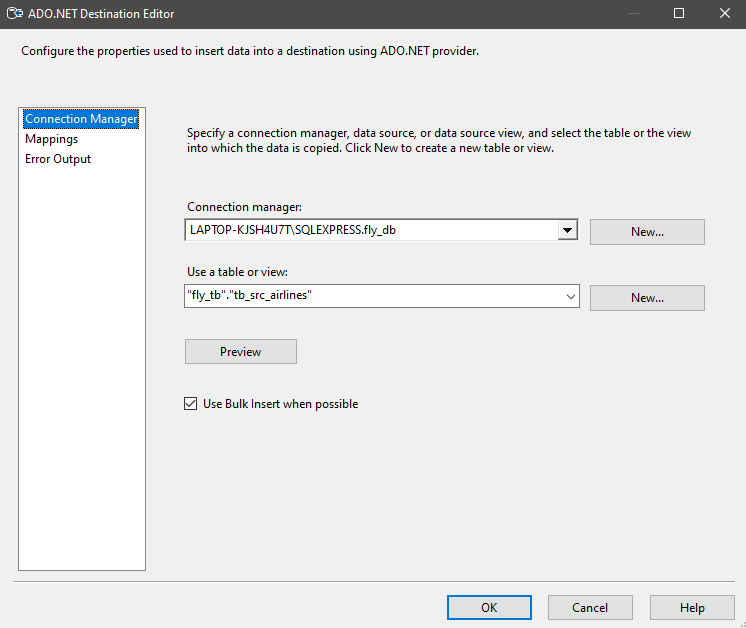
Loading airlines.dat to tb\_src\_airlines

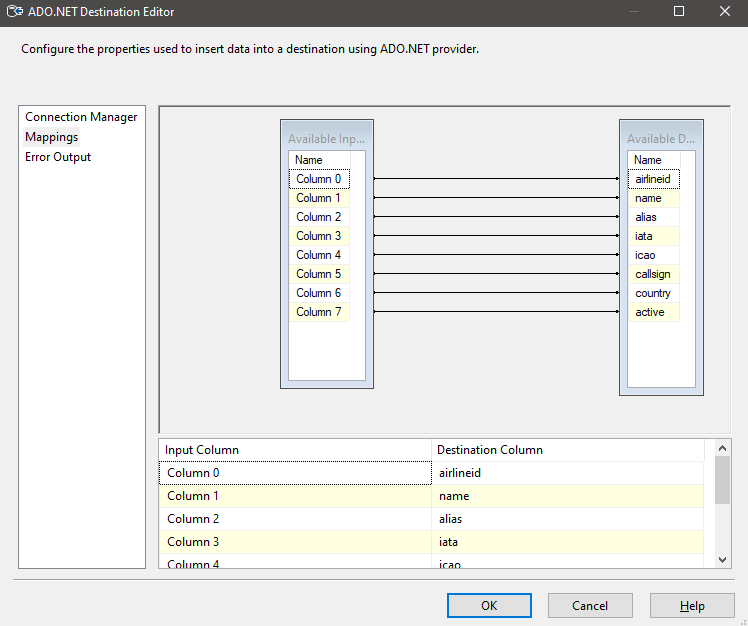


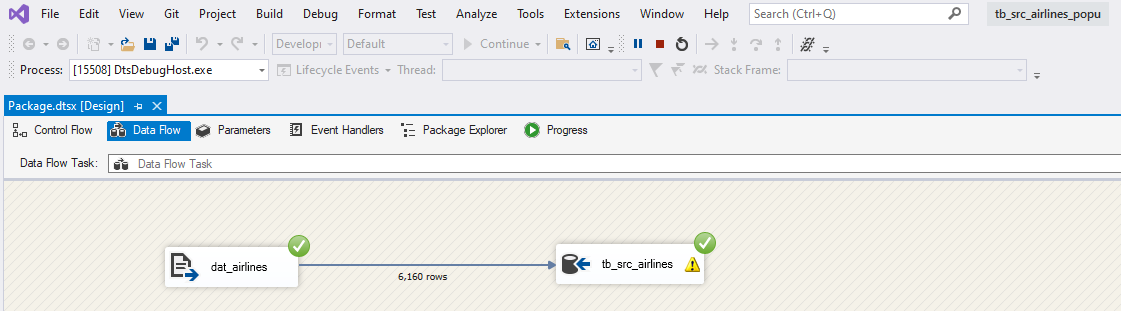






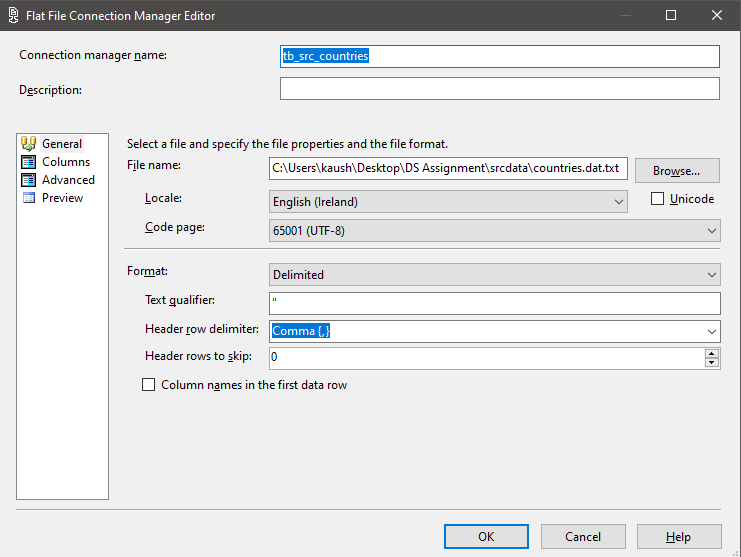


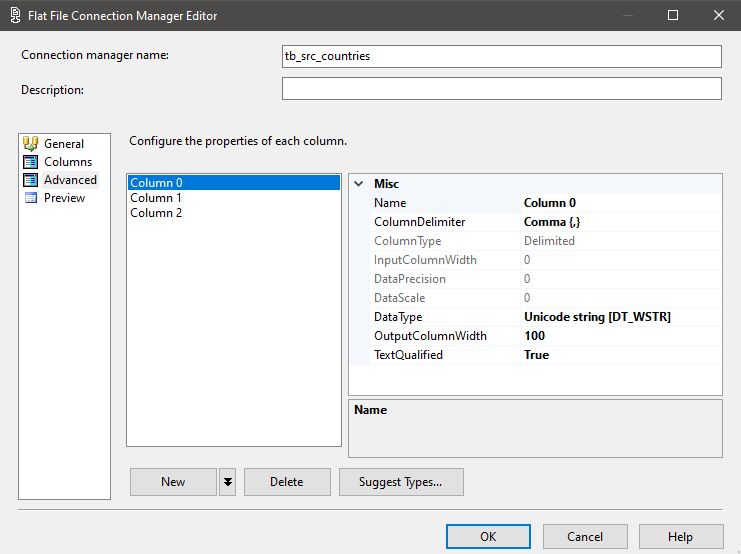


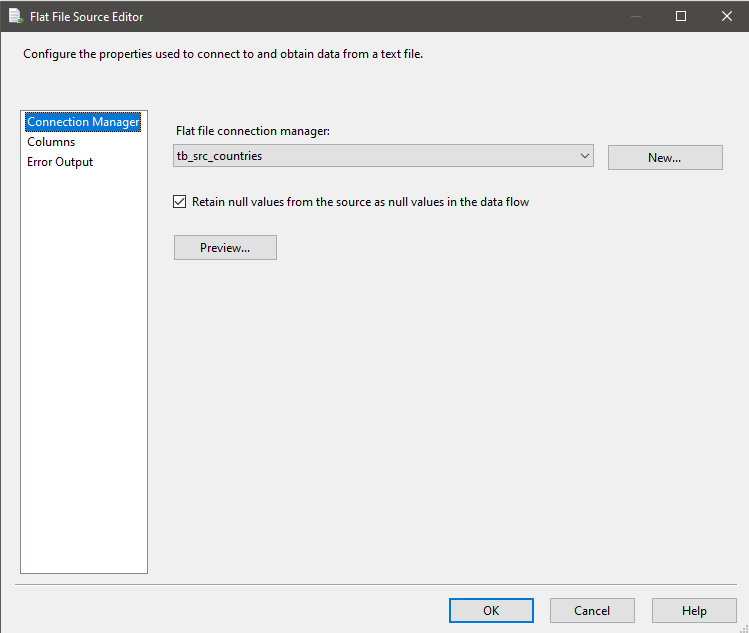


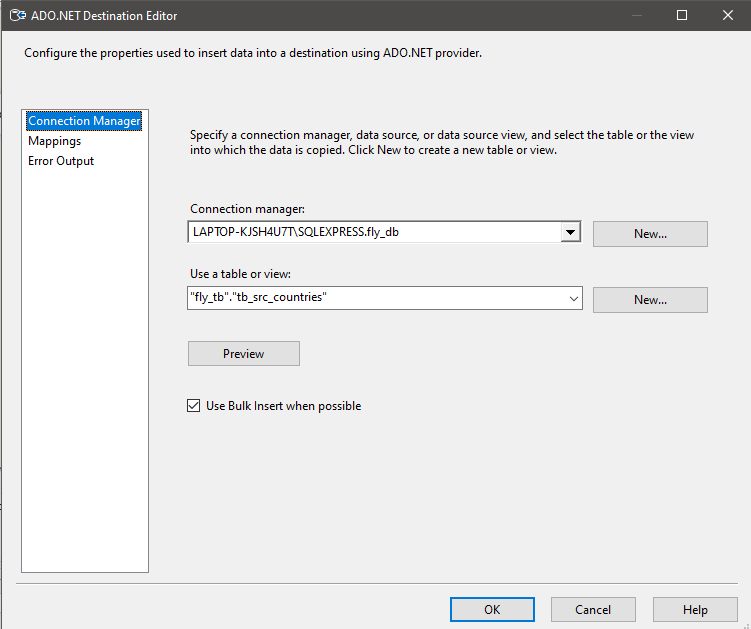
Loading countries.dat to tb\_src\_countries

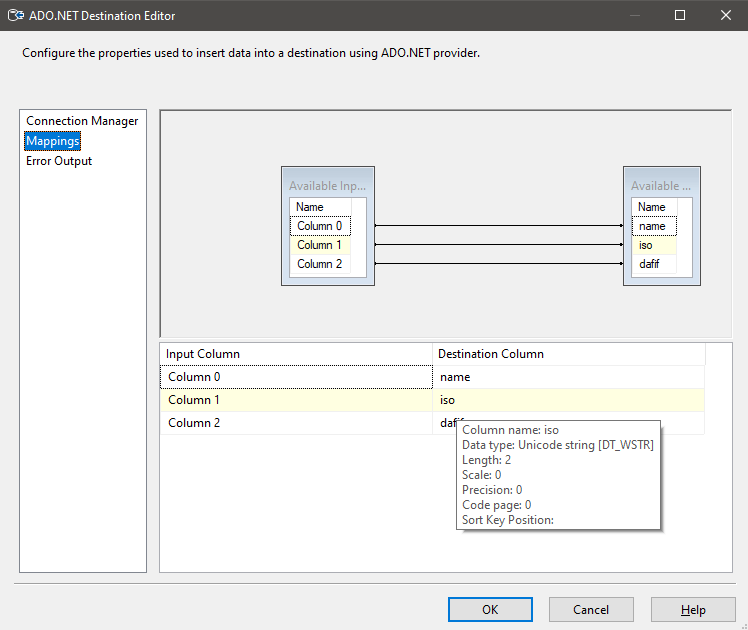


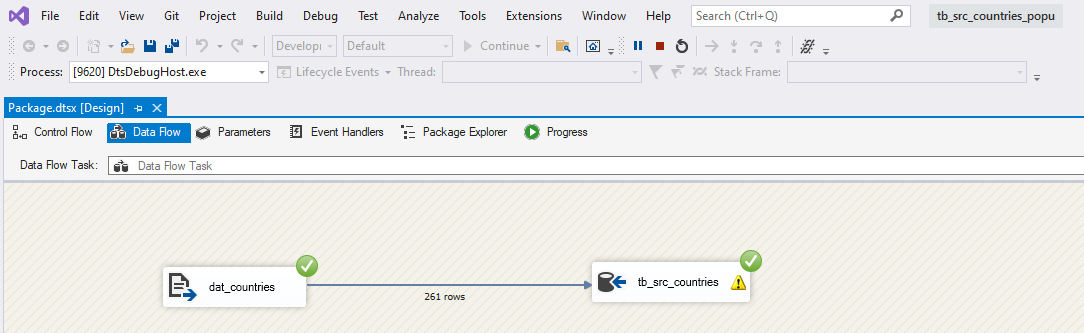






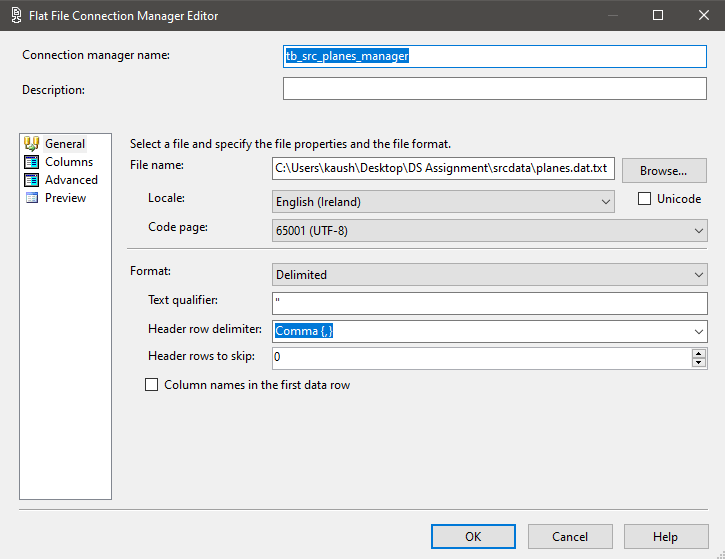


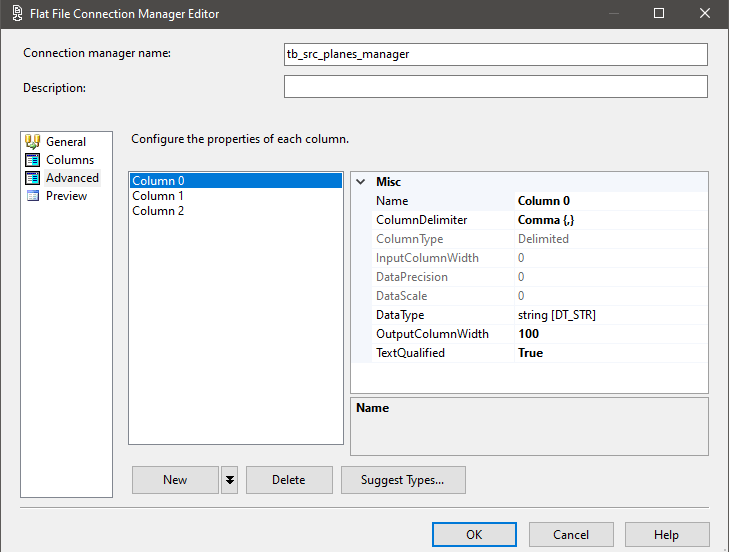


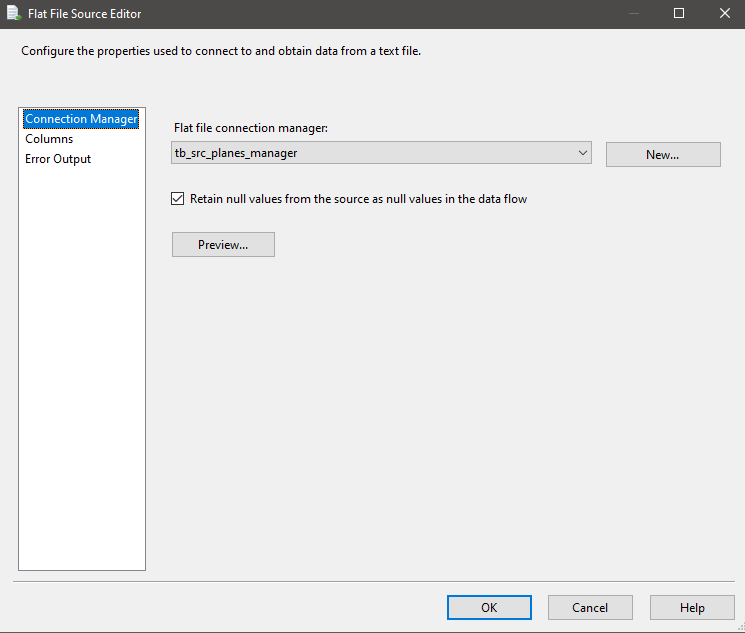


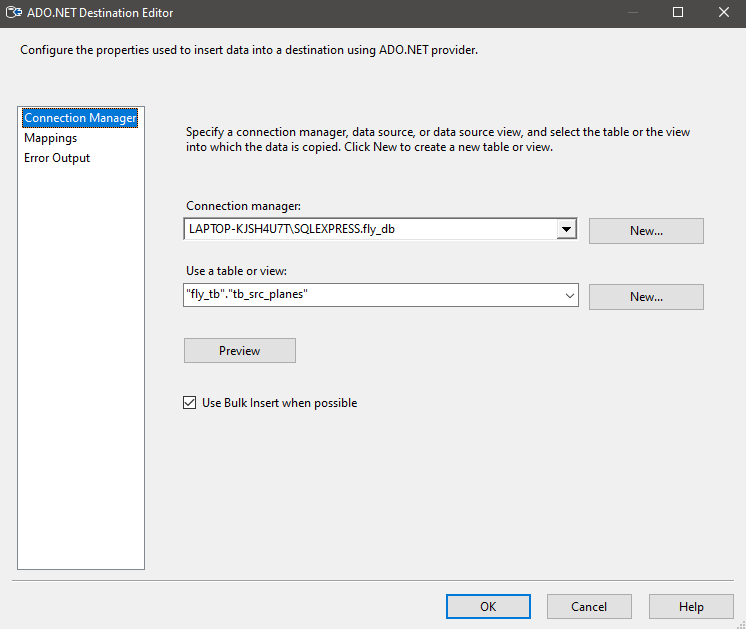
Loading planes.dat to tb\_src\_planes

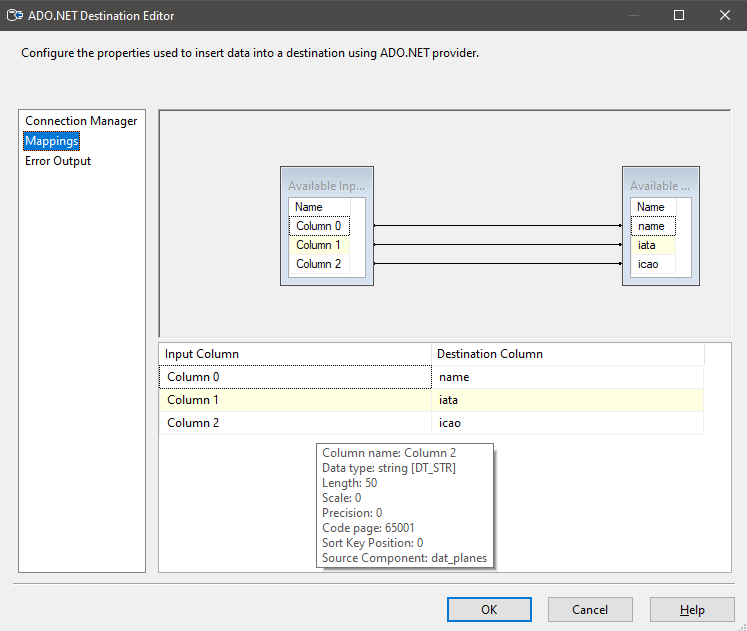


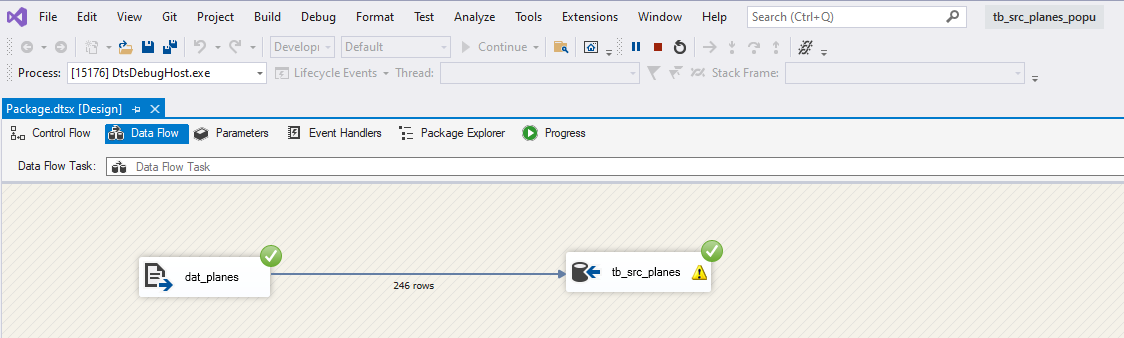






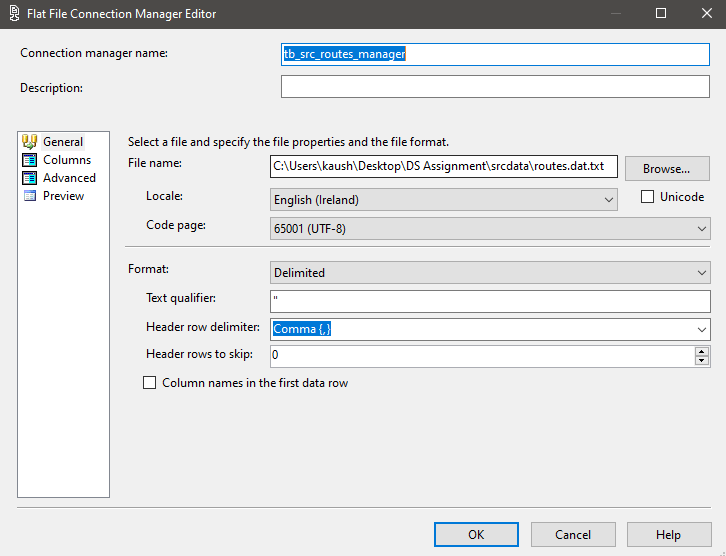


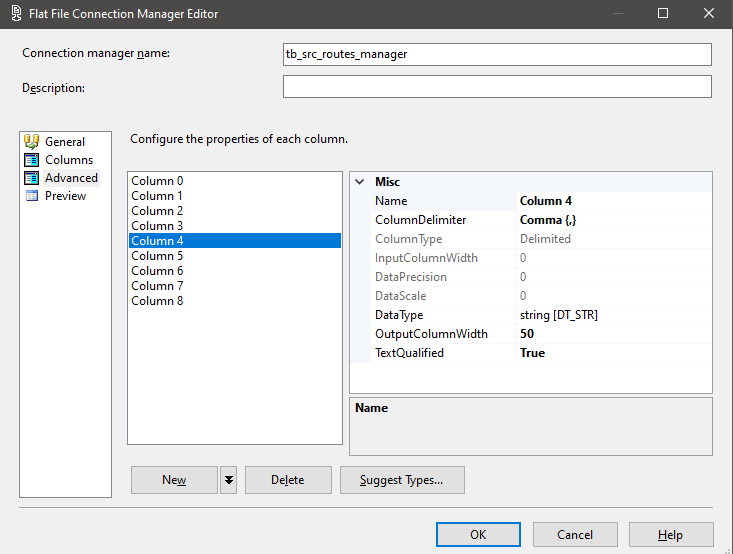


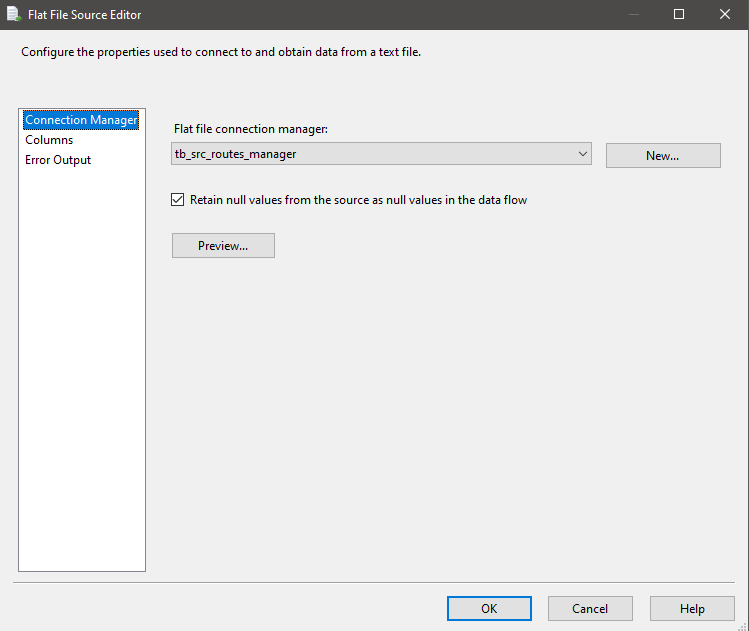


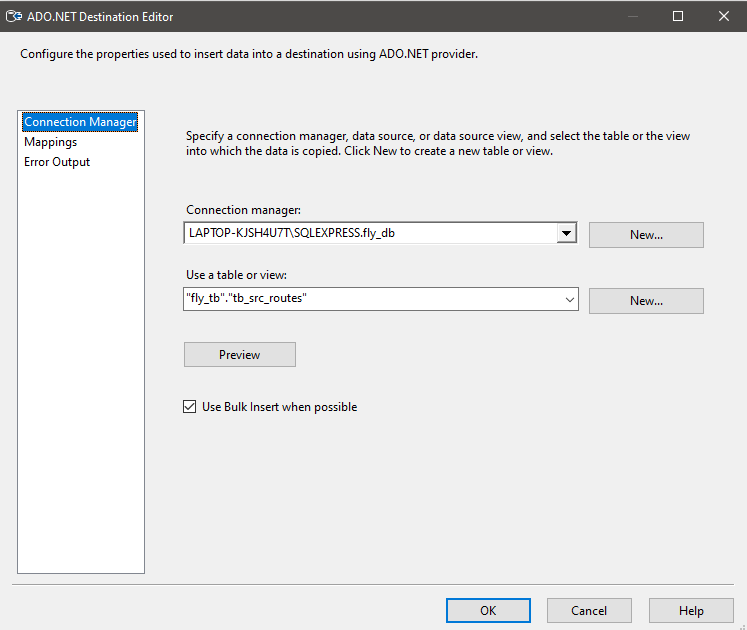
Loading routes.dat to tb\_src\_routes

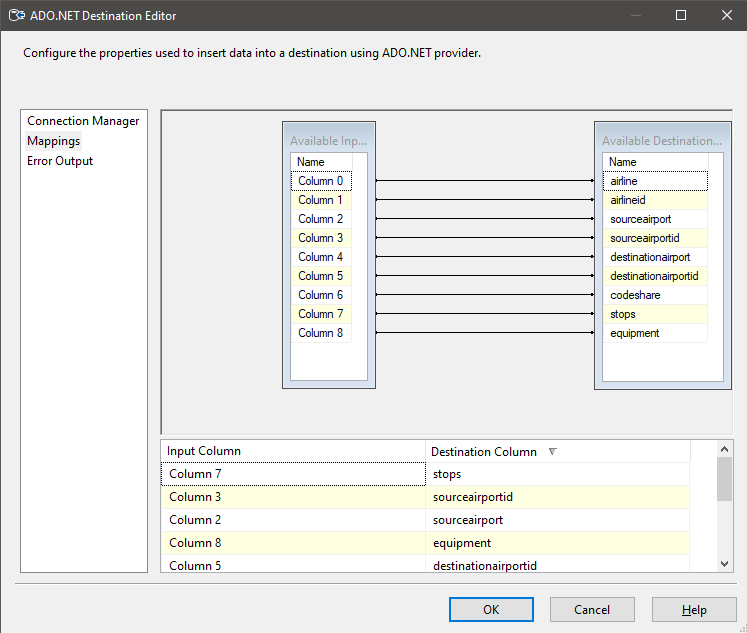


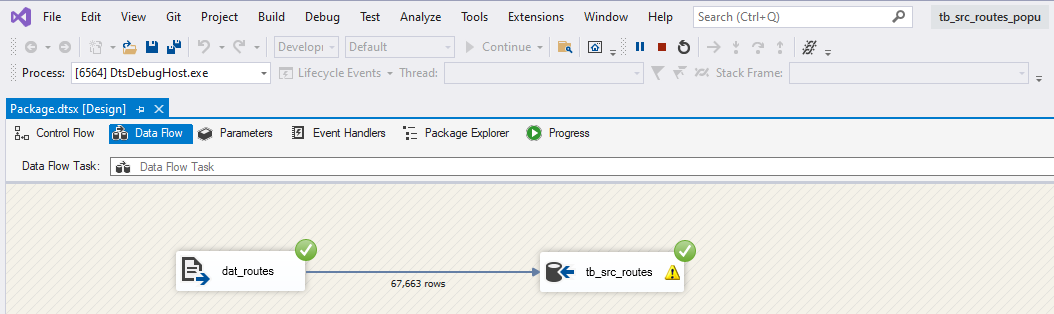












Loading airports.dat to tb\_src\_airports



