SRI LANKA INSTITUE OF INFORMATION TECHNOLOGY

ID No:	**IT20067342**
Name:	**Jayasuriya J.A.D.A.S**
Batch:	**DS weekday**
Assignment:	**01**

Table of Contents

01).Data set selection

Data set : Aviation Accident Database & Synopses

Source : Kaggle

Link to the source: https://www.kaggle.com/datasets/khsamaha/aviation-accident-database-synopses

The dataset contains the aviation accident information .The data set consists of five files: three csv file, one excel file and one text files (Necessary modifications has been done in order to meet the requirements).

ER Diagram

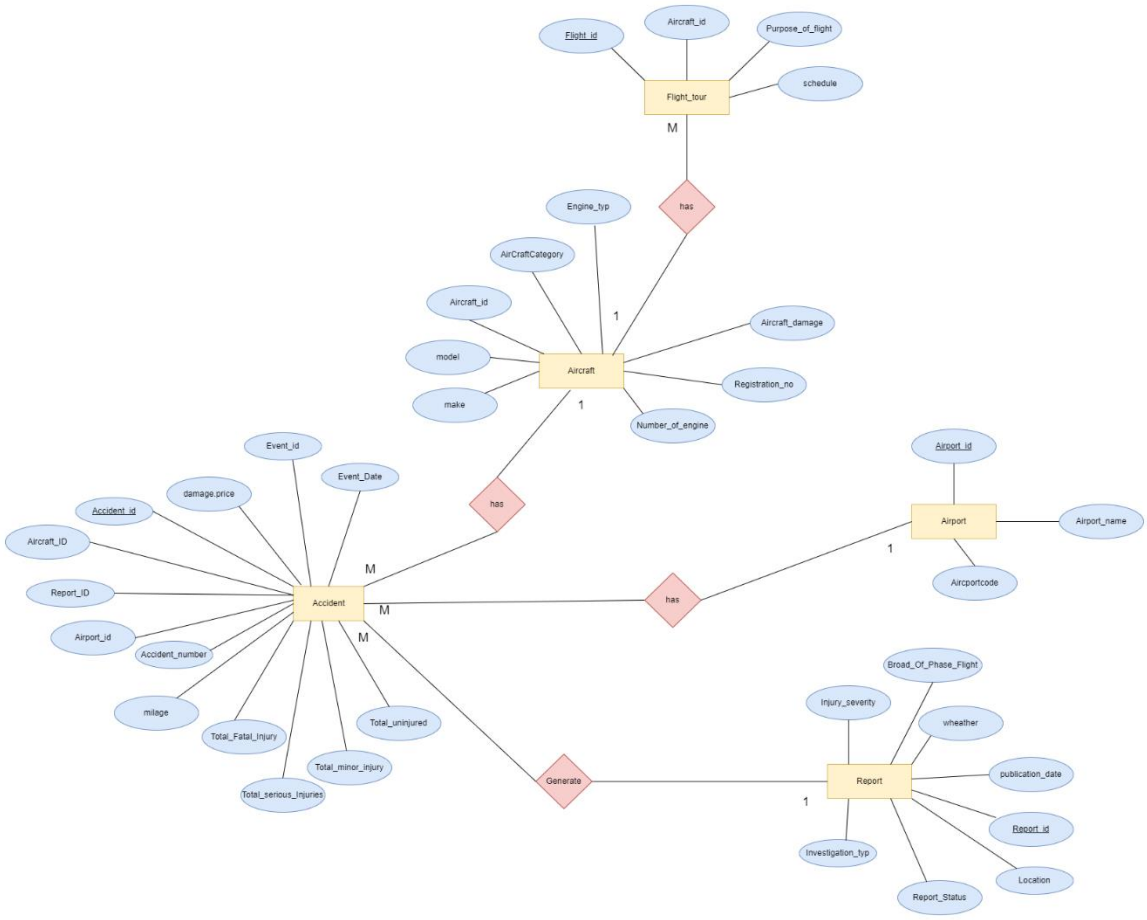


Figure 1.0-ER diagram

02).Preparation of Data Sources

The dataset was originally in the form of one excel file. Data in the file has been seperated into 5 different files of type Excel,CSV and text.

Table	File type
Accidents	Csv file(.csv)
Aircraft	Csv file(.csv)
Reports	Csv file(.csv)
Aircrafts	Excel file(.xlsx)
Flights	Text file(.txt)

Figure 1.1-sample source table creation

Similarly other tables has also been created and then the tables has been exported in relevant file types.

Final set of Sources:

Accident	5/10/2022 8:05 PM	Microsoft Excel Co...	777 KB
Aircraft	5/7/2022 1:19 PM	Microsoft Excel Co...	641 KB
Airport	5/6/2022 5:17 PM	Text Document	20 KB
Flight_Tours	5/9/2022 4:49 AM	Microsoft Excel W...	264 KB
Report	5/7/2022 2:06 PM	Microsoft Excel Co...	798 KB

03).Solution architecture

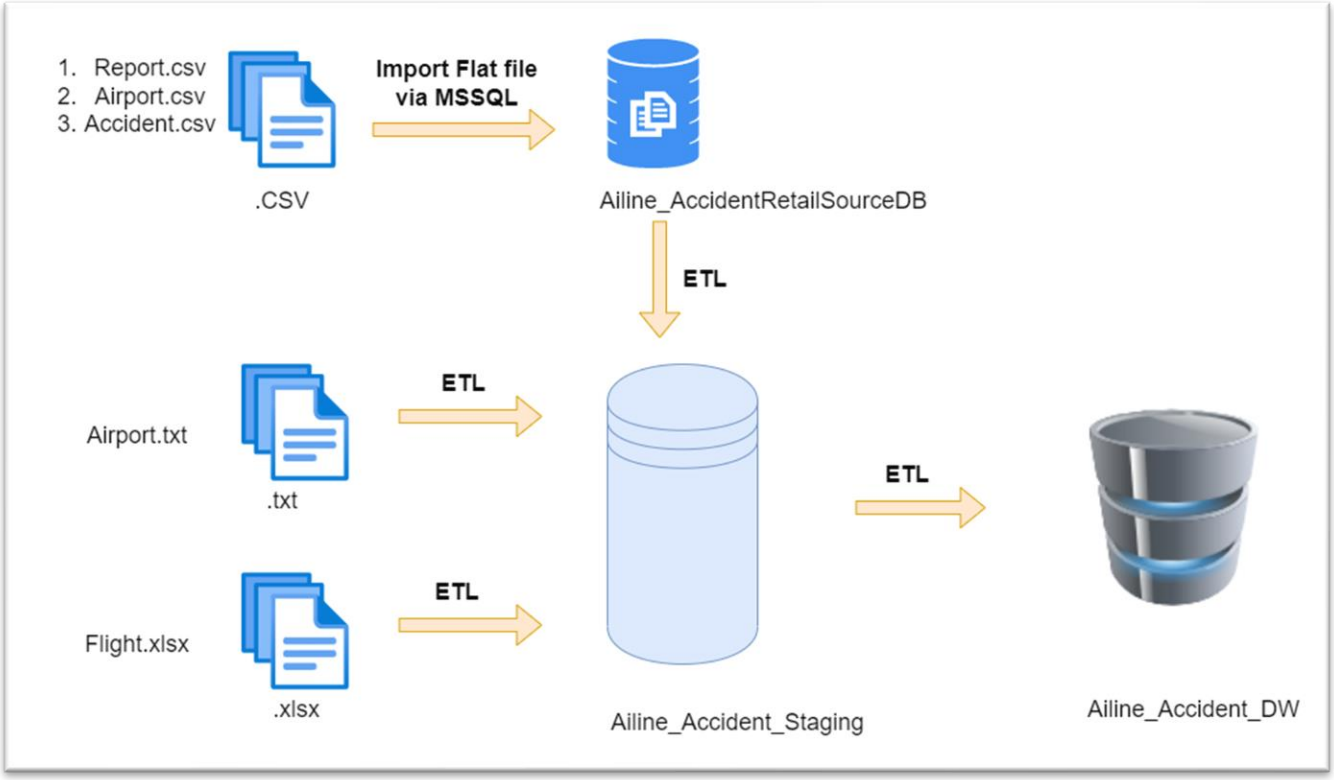


Figure 2.0-Solution architecture

As can be seen in the figure three different resource types has been used to extract data to staging. Staging layer has been used to have all the tables in a single location as in the below figure.

Figure 3.0-Staging

The tables at the staging are then profiled and after performing a rich set of ETL tasks, data is loaded to the data warehouse where from that several reporting tools and analysing tools can use data for reporting mining and analysing.

04). Data warehouse design & development

The datawarehouse is designed as a snow flake schema with one fact table and five dimension table.

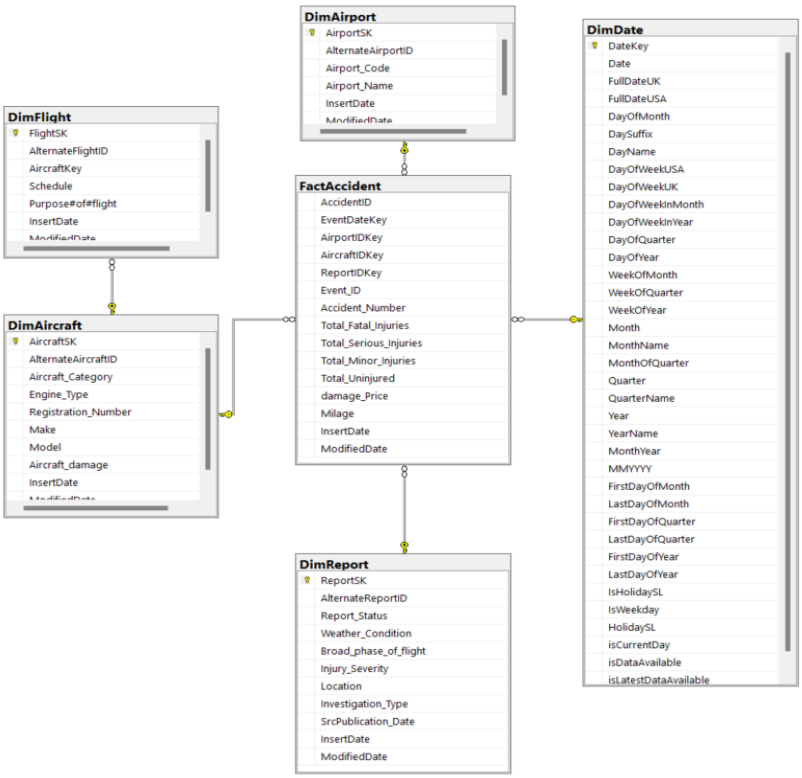


Figure 4.0-Datawarehouse design

05).ETL Development

As the first step data has been extracted from sources to staging area. Data flow task has been used for every extraction.

Data Extraction

5.1 Accident Data from Source to Staging

5.1.1 Data Flow

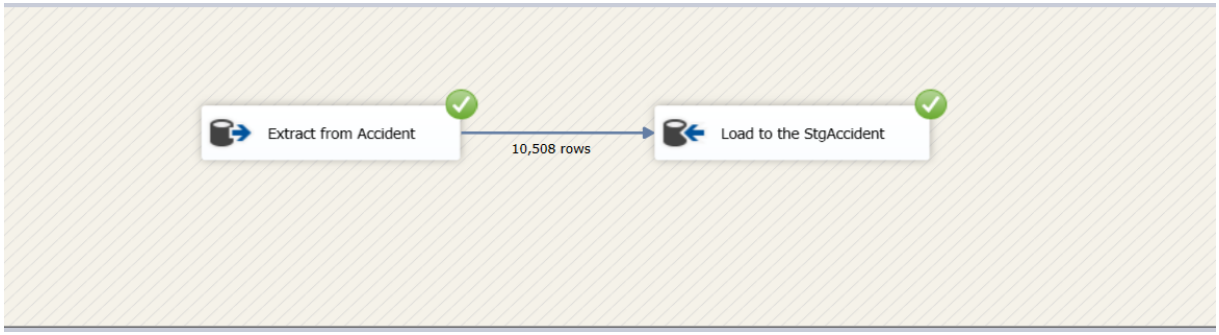


Figure 5.1.1-Accident data flow

5.1.2 Event handler

Before executing 'extract Accident to staging' existing data in the staging layer has been truncated.

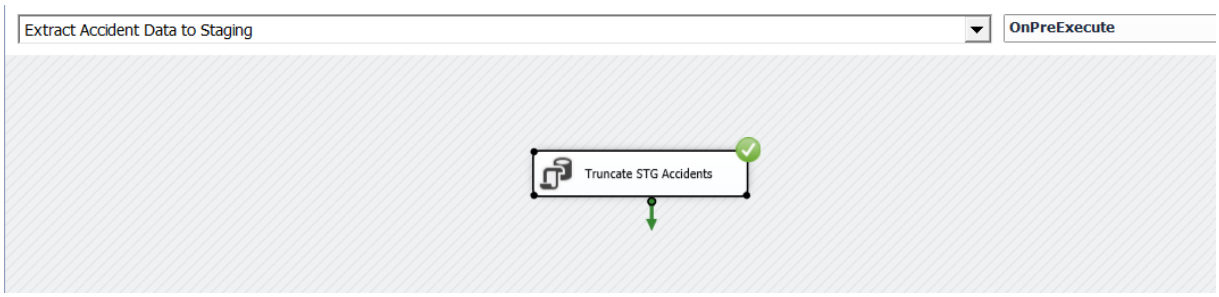


Figure 5.1.2.-Accident Event handler

5.2 Flight Data from Source to Staging

5.2.1 Data Flow

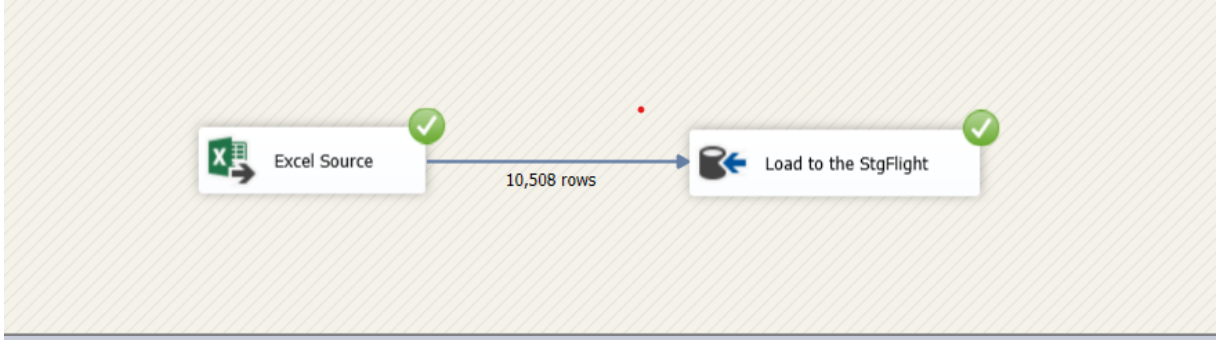


Figure 5.2.1-Flight data flow

5.2.2 Event handler

Before executing 'extract flight to staging' existing data in the staging layer has been truncated.

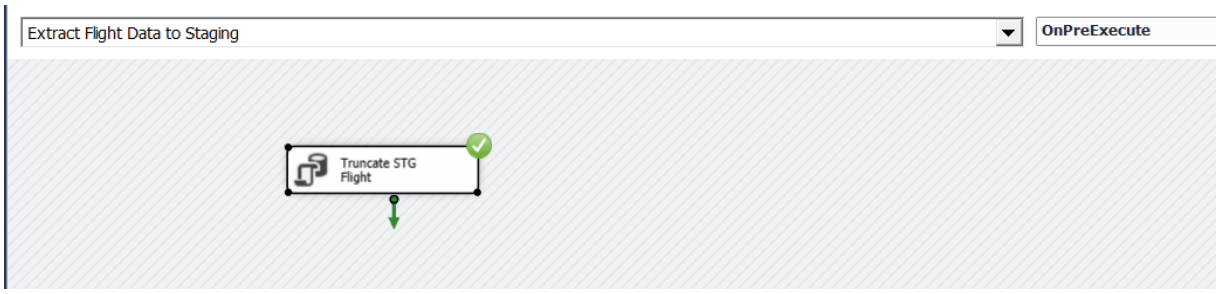


Figure 5.2.2.-Flight Event handler

5.3 Airport Data from Source to Staging

5.3.1 Data Flow

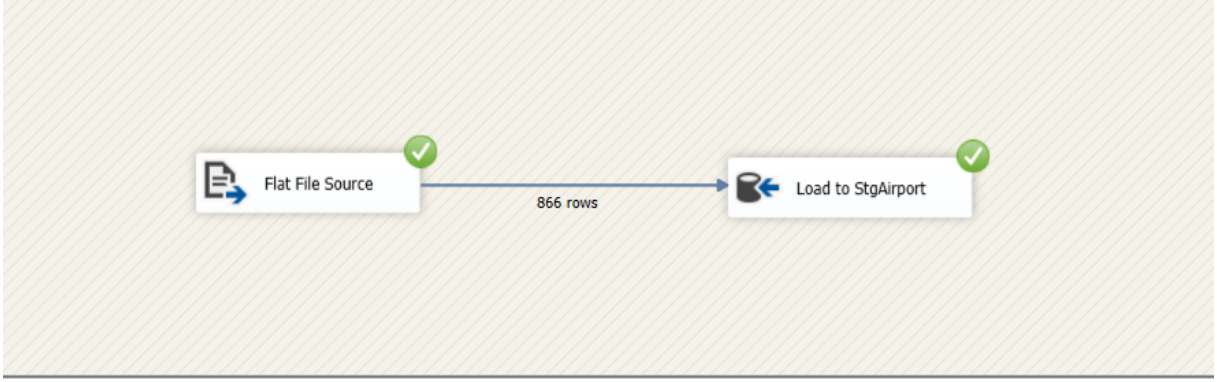


Figure 5.3.-Airport data flow

5.3.2 Event handler

Before executing 'extract Airport to staging' existing data in the staging layer has been truncated.

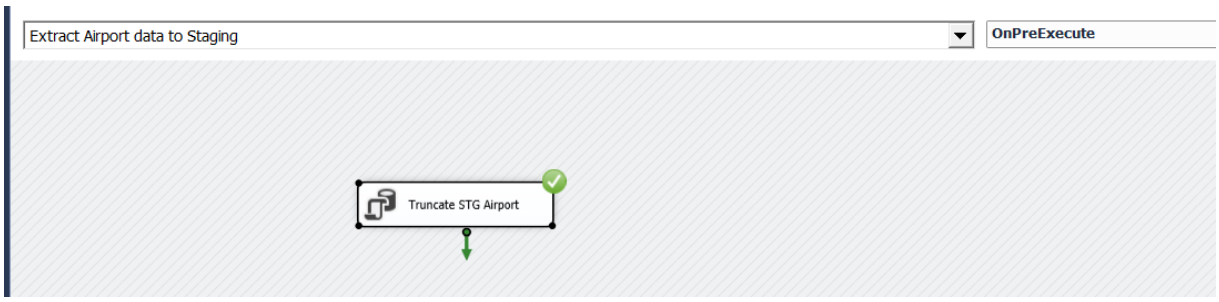


Figure 5.3.1.-Airport Event handler

Overall control flow

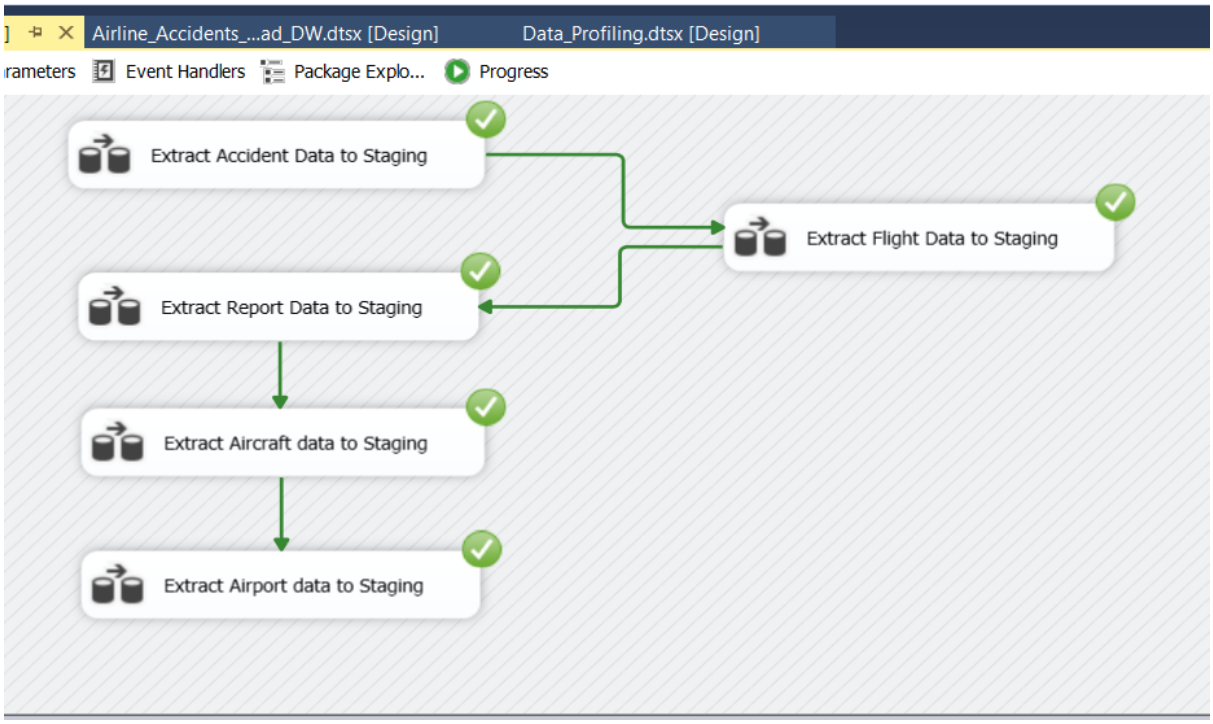


Figure 5.8.-source to staging control flow

Data Profiling

Before Loading staging tables to the data warehouse data has to be enriched to obtain the most suitable data for analysing. Data profiling has been done in order to identify what need to be corrected in ETL process in order to meet this requirement.

Each and every table at staging is profiled and stored in a specific file location.

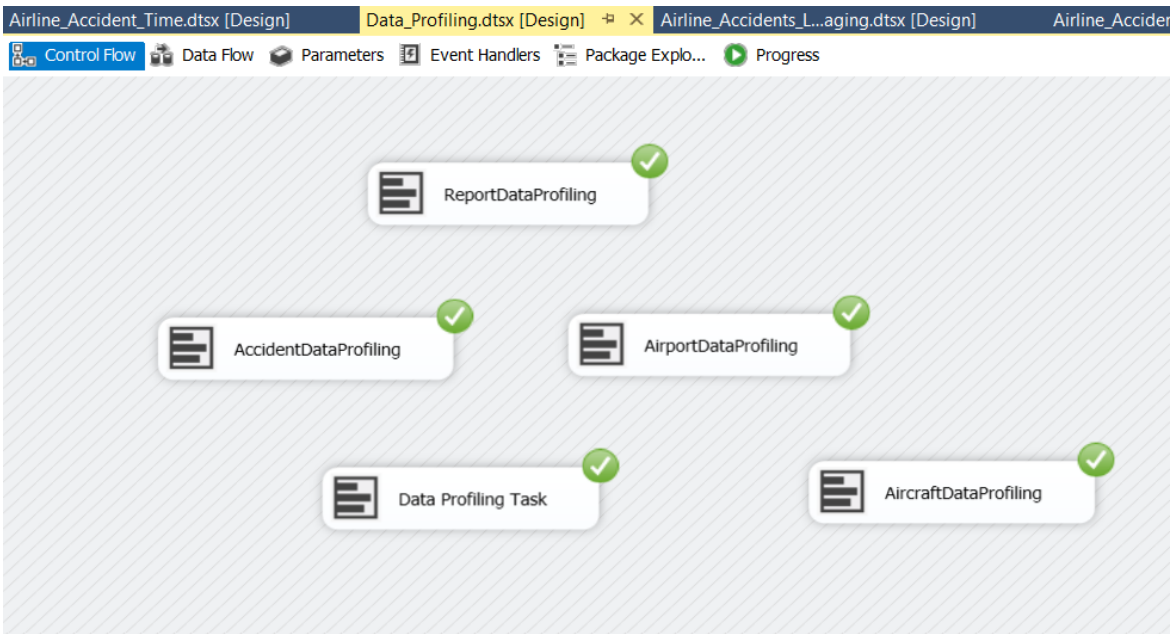
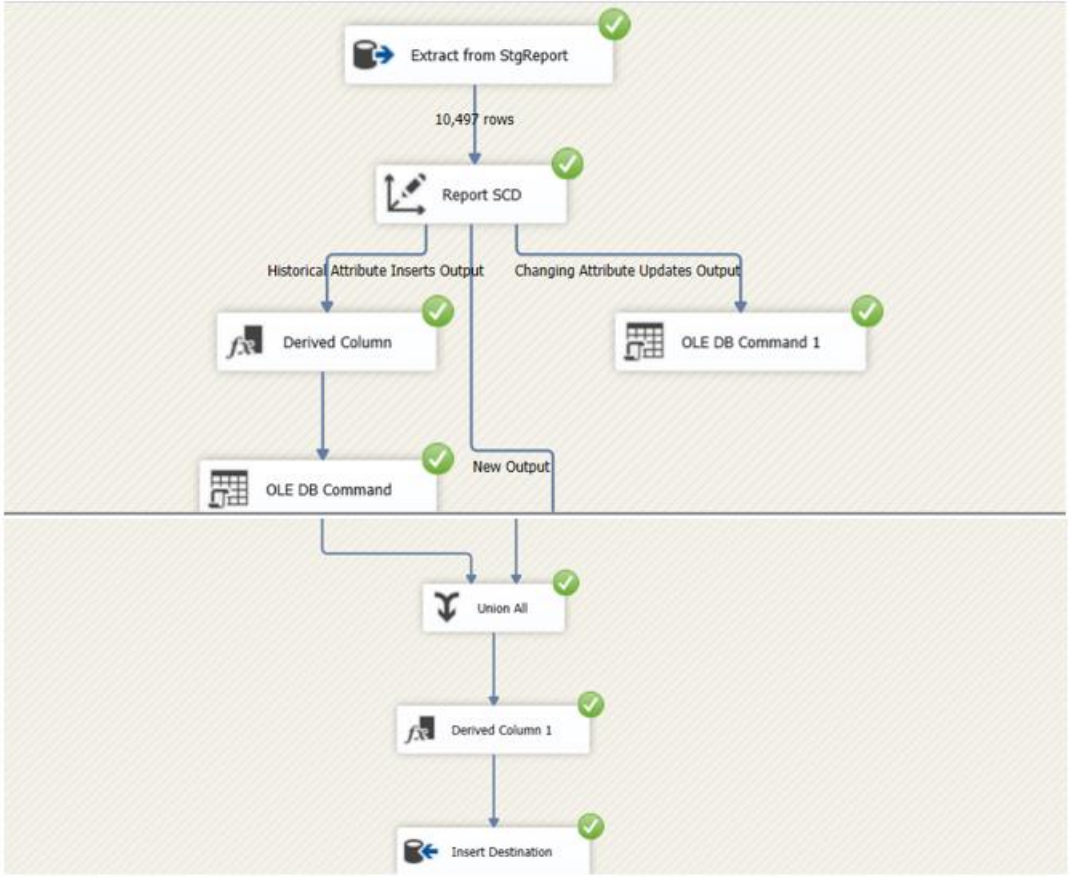


Figure 5.9.1-profiling diagram

Data Transforming and loading

b)Load Slowly changing Dimensions

5.11 Report Data from Staging to Data Warehouse



5.12 Airport Data from Staging to Datawarehouse

StgAirport data has been loaded to dimAirport

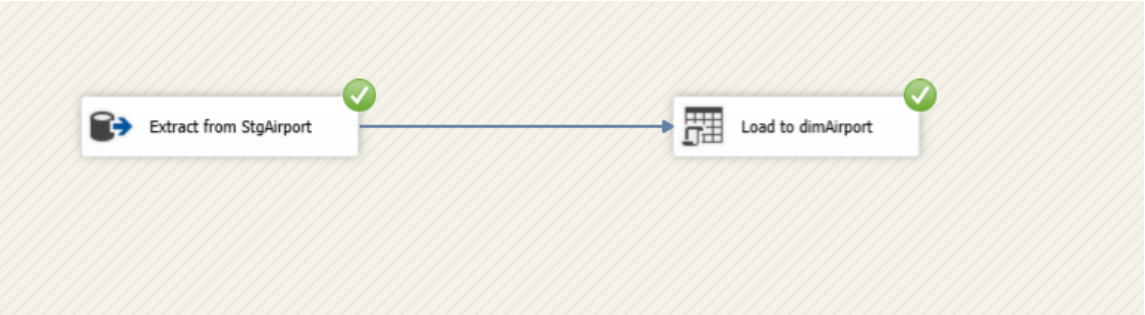


Figure 5.12.1-load to DimAirport

5.15 Creation of Date Dimension

```
CREATE TABLE [dbo].[DimDate](
[DateKey] [int] NOT NULL,
[Date] [datetime] NULL,
[FullDateUK] [char](10) NULL,
[FullDateUSA] [char](10) NULL,
[DayOfMonth] [varchar](2) NULL,
[DaySuffix] [varchar](4) NULL,
[DayName] [varchar](9) NULL,
[DayOfWeekUSA] [char](1) NULL,
[DayOfWeekUK] [char](1) NULL,
[DayOfWeekInMonth] [varchar](2) NULL,
[DayOfWeekInYear] [varchar](2) NULL,
[DayOfQuarter] [varchar](3) NULL,
[DayOfYear] [varchar](3) NULL,
[WeekOfMonth] [varchar](1) NULL,
[WeekOfQuarter] [varchar](2) NULL,
[WeekOfYear] [varchar](2) NULL,
[Month] [varchar](2) NULL,
[MonthName] [varchar](9) NULL,
[MonthOfQuarter] [varchar](2) NULL,
[Quarter] [char](1) NULL,
[QuarterName] [varchar](9) NULL,
[Year] [char](4) NULL,
[YearName] [char](7) NULL,
[MonthYear] [char](10) NULL,
[MMYYYY] [char](6) NULL,
[FirstDayOfMonth] [date] NULL,
[LastDayOfMonth] [date] NULL,
[FirstDayOfQuarter] [date] NULL,
[LastDayOfQuarter] [date] NULL,
[FirstDayOfYear] [date] NULL,
[LastDayOfYear] [date] NULL,
[IsHolidaySL] [bit] NULL,
[IsWeekday] [bit] NULL,
[HolidaySL] [varchar](50) NULL,
[IsCurrentDay] [int] NULL,
[isDataAvailable] [int] NULL,
[isLatestDataAvailable] [int] NULL,
PRIMARY KEY CLUSTERED
(
[DateKey] ASC
)WITH (PAD_INDEX = OFF,
STATISTICS_NORECOMPUTE = OFF,
 IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
```

Figure 5.14.1-Load to DimDate

```
INSERT INTO [dbo].[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])
```

```
VALUES
        (<DateKey, int,>
        ,<Date, datetime,>
        ,<FullDateUK, char(10),>
        ,<FullDateUSA, char(10),>
        ,<DayOfMonth, varchar(2),>
        ,<DaySuffix, varchar(4),>
        ,<DayName, varchar(9),>
        ,<DayOfWeekUSA, char(1),>
        ,<DayOfWeekUK, char(1),>
        ,<DayOfWeekInMonth, varchar(2),>
        ,<DayOfWeekInYear, varchar(2),>
        ,<DayOfQuarter, varchar(3),>
        ,<DayOfYear, varchar(3),>
        ,<WeekOfMonth, varchar(1),>
        ,<WeekOfQuarter, varchar(2),>
        ,<WeekOfYear, varchar(2),>
        ,<Month, varchar(2),>
        ,<MonthName, varchar(9),>
        ,<MonthOfQuarter, varchar(2),>
        ,<Quarter, char(1),>
        ,<QuarterName, varchar(9),>
        ,<Year, char(4),>
        ,<YearName, char(7),>
        ,<MonthYear, char(10),>
        ,<MMYYYY, char(6),>
        ,<FirstDayOfMonth, date,>
        ,<LastDayOfMonth, date,>
        ,<FirstDayOfQuarter, date,>
        ,<LastDayOfQuarter, date,>
        ,<FirstDayOfYear, date,>
        ,<LastDayOfYear, date,>
        ,<IsHolidaySL, bit,>
        ,<IsWeekday, bit,>
        ,<HolidaySL, varchar(50),>
        ,<isCurrentDay, int,>
        ,<isDataAvailable, int,>
        ,<isLatestDataAvailable, int,>)
GO
```

Query used to create and load data to date dimension is listed above. Date dimension is assumed to tally with publication_date in report table

c)Load Fact Table

StgAccident is loaded and merged to obtain Accident_ID. All required surrogate keys has been loaded to data warehouse after a lookup through alternate keys in dimension tables.

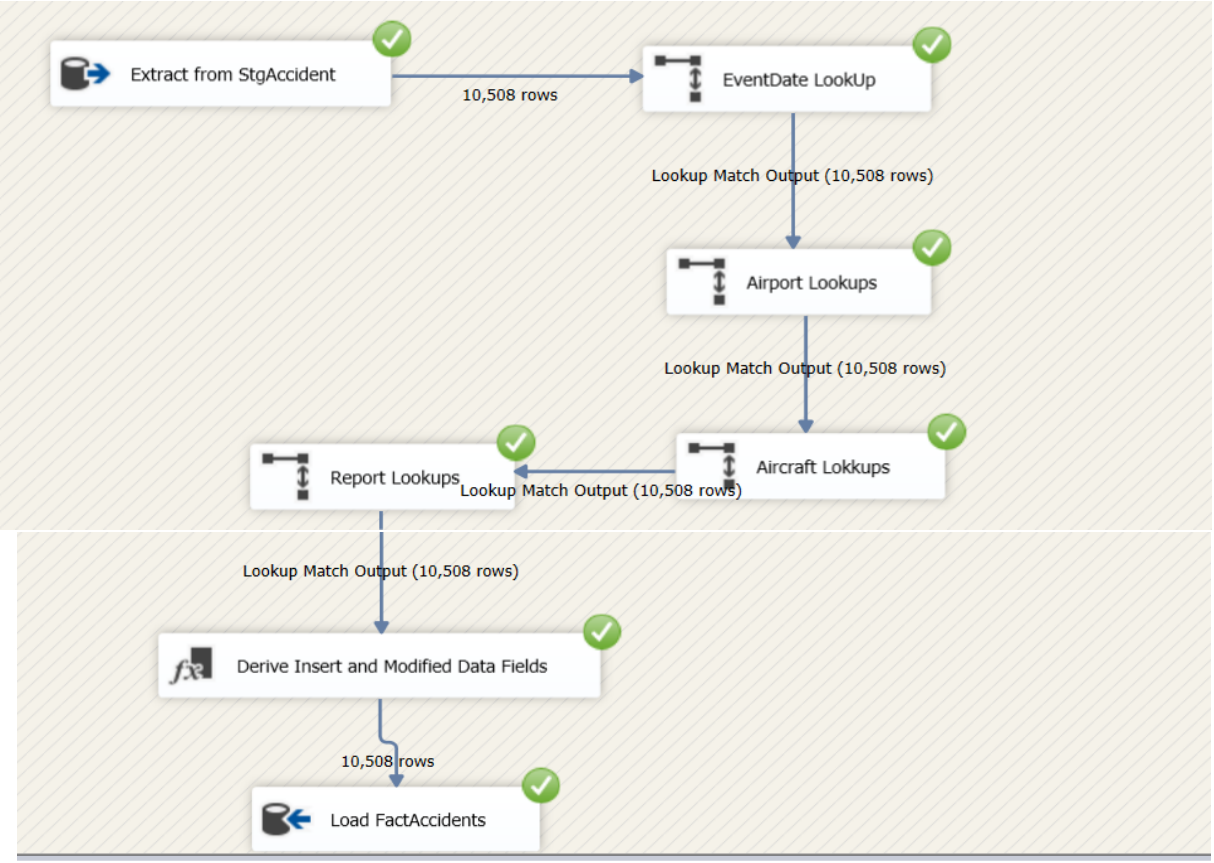


Figure 5.16.1-FactAccident ETL

Overall ETL Transformation

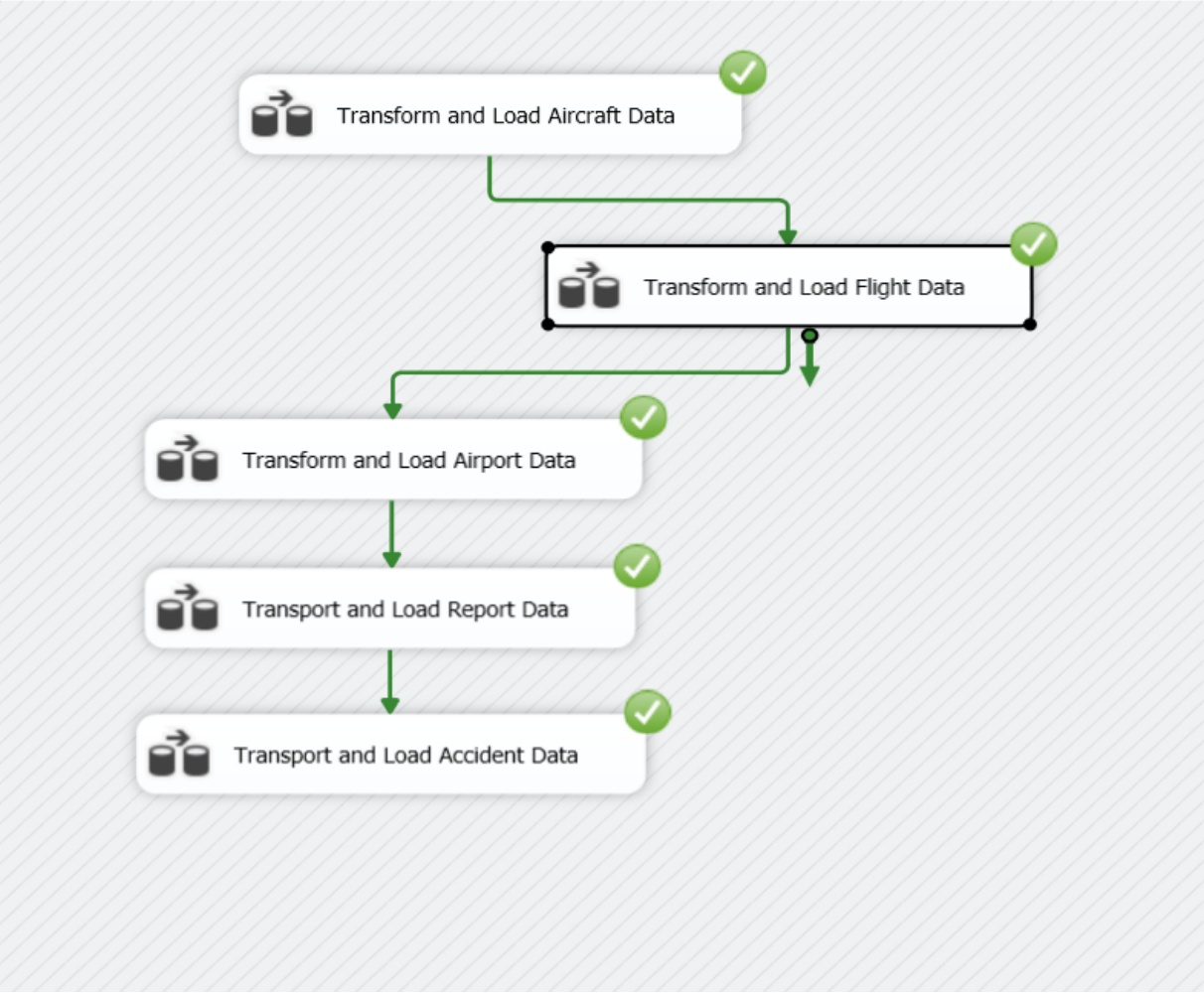


Figure 5.17.1-overall ETL to data warehouse

6).ETL development – Accumulating fact tables

Fact table was extended by adding last three attributes as shown below.

MSI\MSSQLSERVER_...dbo.FactAccident | SQLQuery4.sql - MSI...ing (MSI\user (71))

Column Name	Data Type	Allow Nulls
AccidentID	int	☑
EventDateKey	int	☑
AirportIDKey	int	☑
AircraftIDKey	int	☑
ReportIDKey	int	☑
Event_ID	nvarchar(50)	☑
Accident_Number	nvarchar(50)	☑
Total_Fatal_Injuries	int	☑
Total_Serious_Injuries	int	☑
Total_Minor_Injuries	int	☑
Total_Uninjured	int	☑
damage_Price	float	☑
Milage	float	☑
InsertDate	datetime	☑
ModifiedDate	datetime	☑
accm_txn_create_time	datetime	☑
accm_txn_complete_time	datetime	☑
txn_process_time_hours	int	☑
		☐

Then a separate data source (.txt) named Time.txt was created.

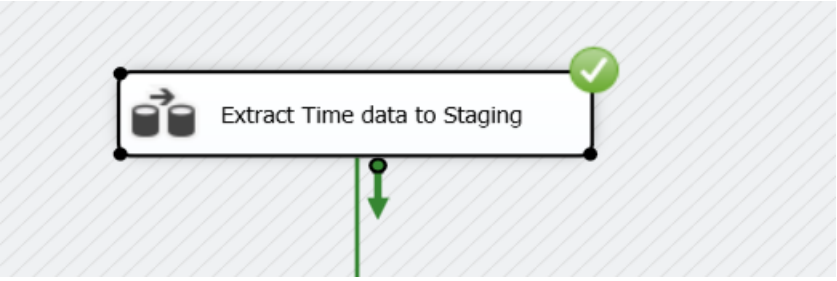
	A	B	C
1	fact_table_natural_key	accm_txn_complete_time	
2	1	5/21/2022 18:06	
3	2	5/25/2022 7:43	
4	3	5/21/2022 13:38	
5	4	5/23/2022 3:34	
6	5	5/25/2022 1:21	
7	6	5/19/2022 20:17	
8	7	5/23/2022 15:19	
9	8	5/16/2022 20:42	
10	9	5/19/2022 22:05	
11	10	5/25/2022 0:35	
12	11	5/21/2022 15:19	
13	12	5/21/2022 13:08	
14	13	5/16/2022 12:16	

Time - Notepad

File Edit Format View Help

```
fact_table_natural_key (txn_id)         accm_txn_complete_time
1       5/21/2022 18:06
2       5/25/2022 7:43
3       5/21/2022 13:38
4       5/23/2022 3:34
5       5/25/2022 1:21
6       5/19/2022 20:17
7       5/23/2022 15:19
8       5/16/2022 20:42
9       5/19/2022 22:05
10      5/25/2022 0:35
11      5/21/2022 15:19
12      5/21/2022 13:08
13      5/16/2022 12:16
```

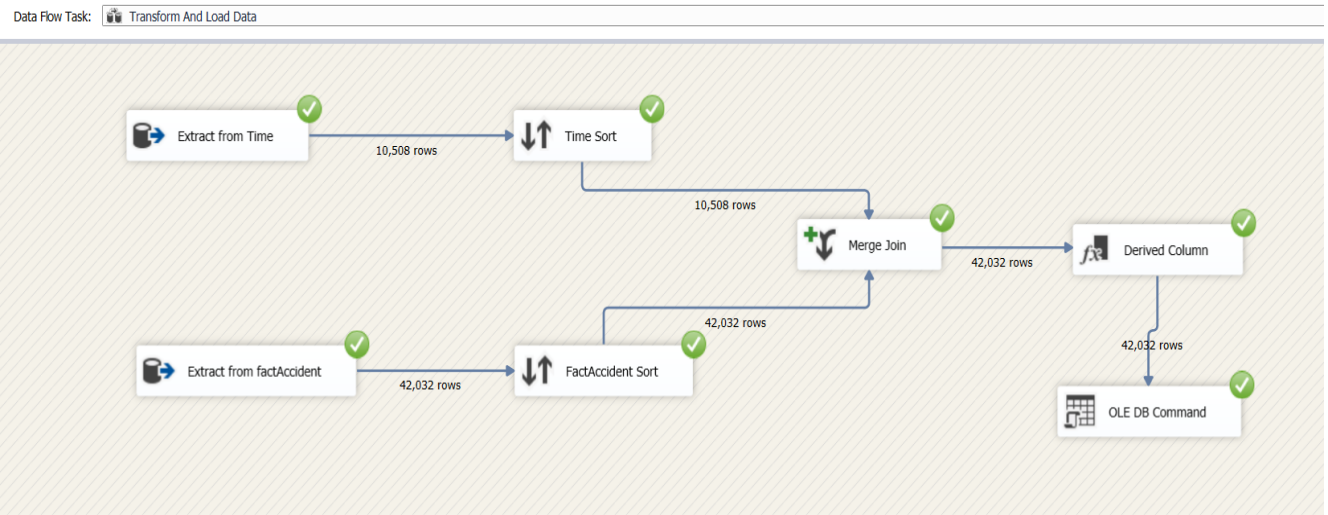
Extract Time data to Staging



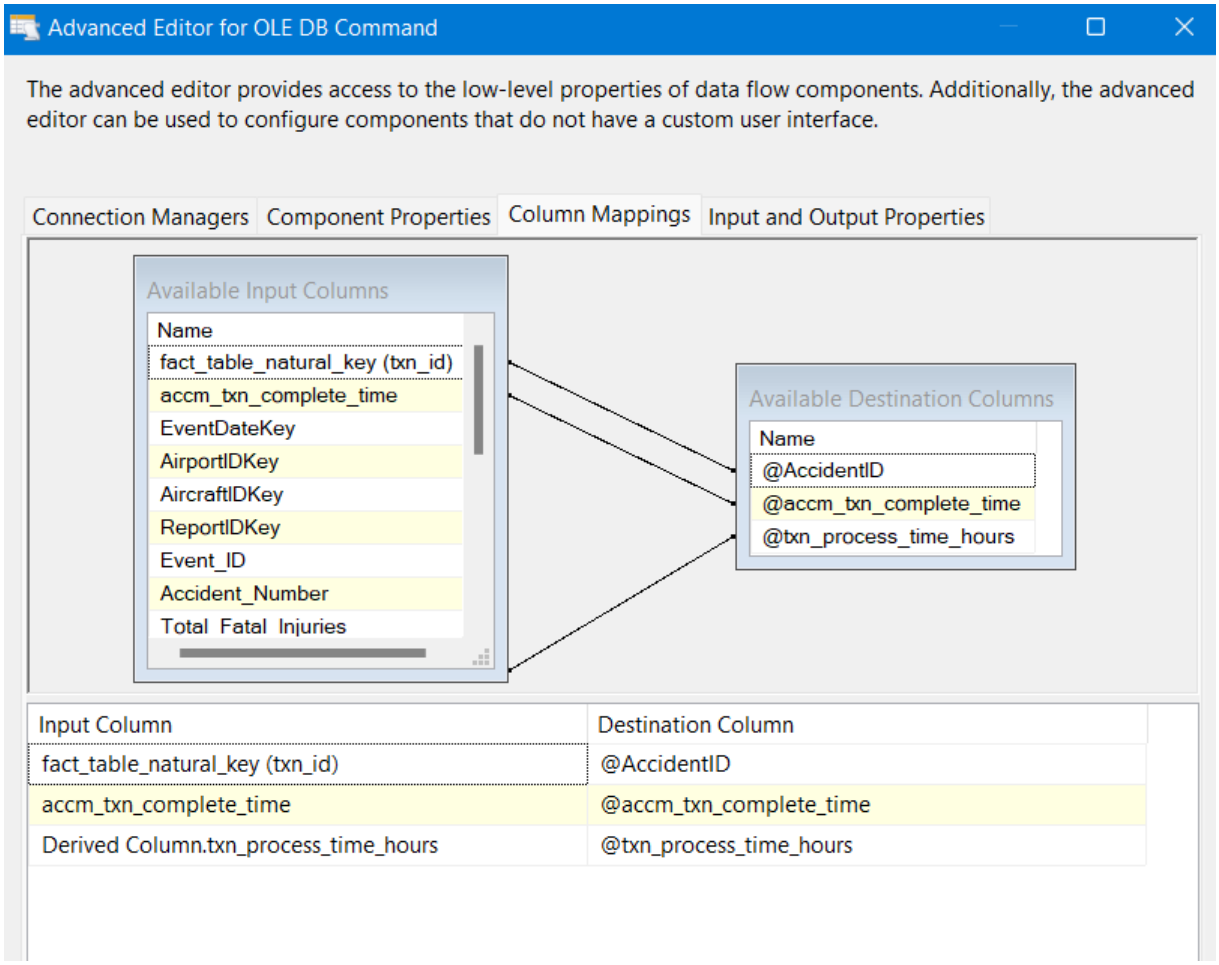
Time data from staging to datawarehouse

Time data source merge to the Fact table.

Data from Time Staging table and FactAccident fact table were extracted and merged. Merge has been performed by sorting both tables using the field fact_table_natural_key(txt_id)



Relevant column mapping is shown below.



A derived column task has been used to derive the values for txt_process_time_hours column by getting the date different of_acc_txt_complete_time and accn_txt_create_time

Derived Column Name	Derived Column	Expression	Data Type	L
txn_process_time_ho...	<add as new column>	DATEDIFF("hh",accm_txn_create_time,accm_tx...	four-byte signed inte...	

The following procedure is used to in order to load data.

A screenshot of FactAccident table after accumulating the (completing the step- 06) is given below

accm_txn_create_time	accm_txn_complete_time	txn_process_time_hours
2022-05-13 22:07:51.707	2022-05-23 03:34:00.000	294
2022-05-13 22:07:51.707	2022-05-25 01:21:00.000	267
2022-05-13 22:07:51.707	2022-05-19 20:17:00.000	142
2022-05-13 22:07:51.707	2022-05-23 15:19:00.000	233
2022-05-13 22:07:51.707	2022-05-16 20:42:00.000	70
2022-05-13 22:07:51.707	2022-05-19 22:05:00.000	144
2022-05-13 22:07:51.707	2022-05-25 00:35:00.000	339
2022-05-13 22:07:51.707	2022-05-21 15:19:00.000	258
2022-05-13 22:07:51.707	2022-05-21 13:08:00.000	183
2022-05-13 22:07:51.707	2022-05-16 12:16:00.000	135
2022-05-13 22:07:51.707	2022-05-22 03:22:00.000	270
2022-05-13 22:07:51.707	2022-05-23 04:28:00.000	222
2022-05-13 22:07:51.707	2022-05-17 05:01:00.000	79
2022-05-13 22:07:51.707	2022-05-17 01:08:00.000	75
2022-05-13 22:07:51.707	2022-05-17 10:19:00.000	157
2022-05-13 22:07:51.707	2022-05-24 07:40:00.000	222