

Sri Lanka Institute of Information Technology

**Seattle Airbnb Data Warehouse Solution**

**Assignment-1 Document**

IT3021 - Data Warehousing and Business Intelligence

Assignment 1

Submitted by: IT20273576 – Malhari H.H.T

Date of submission: 17/05/2022

Table of Contents

[1. Dataset Selection 3](#_Toc103708378)

[2. Preparation of data sources 5](#_Toc103708379)

[3. Solution architecture 7](#_Toc103708380)

[4. Data warehouse design & development 9](#_Toc103708381)

[4.1. Relational Model Schema 9](#_Toc103708382)

[4.2. Dimensional Model Schema 10](#_Toc103708383)

[4.1. Data Warehouse Implementation 12](#_Toc103708384)

[4.1.1. Staging Layer SQL Database Implementation 12](#_Toc103708385)

[4.1.2. Data Warehouse SQL Database Implementation 13](#_Toc103708386)

[5. ETL development 18](#_Toc103708387)

[5.1. SSIS Solution, Packages, and Project Connections 18](#_Toc103708388)

[5.2. Sources to Staging layer EtL 18](#_Toc103708389)

[5.2.1. Connections used 19](#_Toc103708390)

[5.2.2. Tasks of EtL from Sources to Staging Database Tables. 19](#_Toc103708391)

[5.3. Staging layer to Data Warehouse ETL 25](#_Toc103708392)

[5.3.1. Connections used 25](#_Toc103708393)

[5.3.2. Tasks of ETL from Staging to Data Warehouse in their Execution Order 26](#_Toc103708394)

[6. ETL development – Accumulating Fact Tables 37](#_Toc103708395)

[6.1. Loading updated Accumulated Fact Table FactBestListing 37](#_Toc103708396)

[6.2. Loading updated Accumulated Fact Table FactReview 42](#_Toc103708397)

# Dataset Selection

I selected Seattle Airbnb open data set for the assignment which includes full descriptions of **Listings** including all listing details of Seattle, **Reviews** including unique id for each reviewer and detailed comments and Calendar including listing id, price and availability for a particular day. Airbnb is basically an American company that operates an online marketplace for lodging, primarily homestays for vacation rentals and tourism activities. The original source files can be found using the link provided below.

Source link: <https://www.kaggle.com/datasets/airbnb/seattle>

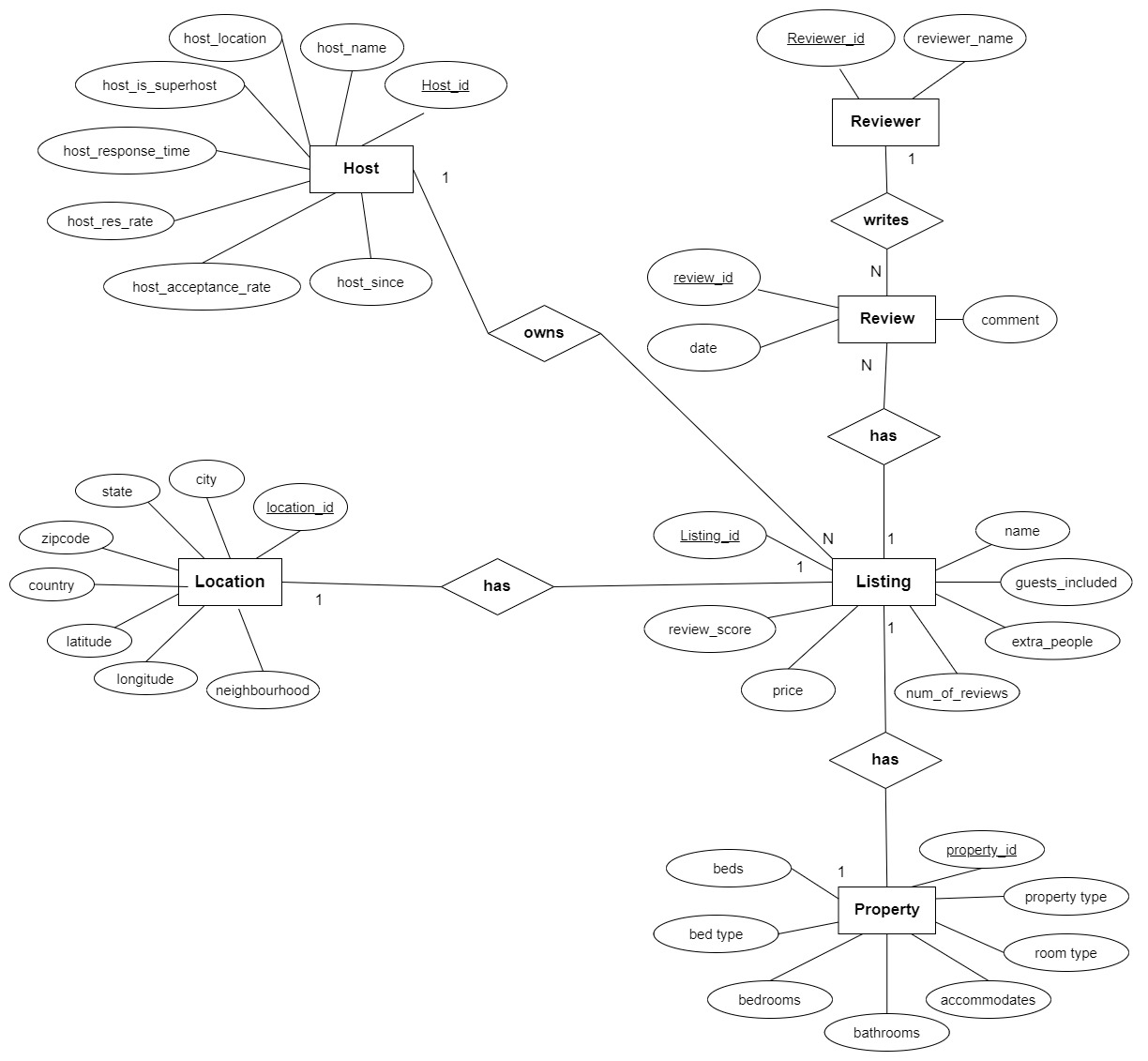
For the purpose of this assignment, I focused only about the Listings and Reviews. The Listings data file comprises of 3813 listings and the Reviews dataset comprises of 84849 review records.

While analyzing the original files it was observed that the dataset contained review records expanded across about six years. (From 2010 – 2015).

Data files had enough records and columns to be divided into several source files and also hierarchies such as Property Type -> Room Type -> Bed Type in listings, year ->month ->date in review records and country -> city -> state in listings were identified.

As shown in the below ER diagram, there are enough entities identified to create Dimension and Fact tables, Potential Lookups that can be used to figure out foreign key like scenarios when connecting dimension tables and fact tables. Thus, it is ensured beforehand that there will be enough data for later, when creating the SSAS cubes and generating reports, based on the above source file data.

Following is the ER Diagram for the chosen data set.



# Preparation of data sources

Among three CSV files I received as mentioned in the previous section I only used 2 CSV files namely Listings and Reviews.

In the initial Listings file, there were 92 columns and then I identified most important columns for the analysis purposes. As well as in the Listings.csv file there were information related to Listing’s basic information, host details, location details and property details of that Listings. So, I separated Listings.csv file into four separate source files.

1. Listing Information 🡪

* Separated Listings basic information into a **database table** **called AirBnbSeattle\_Listings**.
* This table with the help of necessary attributes focus on Listings’ basic information like listings\_id, name, price, guests\_included, extra\_people (price for extra people), number of reviews, review\_scores\_rating, last\_scraped\_date.
* Listings\_id taken as the primary key and host\_id, property\_id and location\_id took as foreign keys.

1. Host Information 🡪

* Airbnb Host related details were separated into a **database table called** **AirBnbSeattle\_Host.**
* This table contains information about host and includes following attributes ; host\_id, host\_name, host\_location, host\_is\_superhost, host\_response\_time, host\_response\_time, host\_response\_rate, host\_acceptance\_rate and host\_since.

1. Location Information 🡪

* Listings’ location related details were separated into a **text file called AirBnbSeattle\_Location.**
* This text file contains location\_id (pk), zip code, city, state, country, latitude, longitude and neighborhood.

1. Property Information 🡪

* Listings’ property related details were separated into a **CSV file called AirBnbSeattle\_Property.**
* This csv file contains following attributes. Property\_id, property\_type, room\_type, accommodates, bathrooms, bedrooms, beds, bed\_type

In the initial Reviews.csv file, there were information of reviews as well as reviewers. So, I separated Review details and Reviewer details into 2 different data sources.

1. Reviewer Information 🡪

* Reviewer id and reviewer name were separated into an **excel file named AirBnbSeattle\_Reviewers.**
* It contains only reviewer\_id and reviewer\_name.

1. Review Information 🡪

* Review details were separated into a **database table called reviews.**
* It contains review\_id, date, comments and it has listing\_id, reviewer\_id as foreign keys.

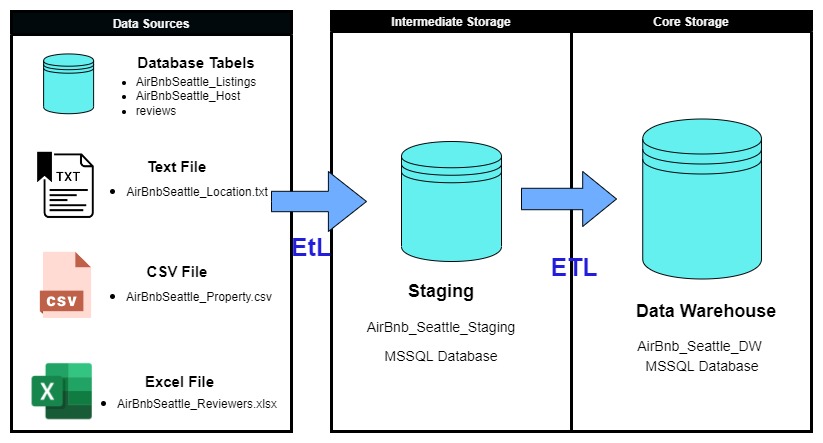
As mentioned above in each table I have included primary keys and relevant foreign keys were also included into the tables.

So, the types of different data sources available are: (6 Sources in Total)

* MS SQL Database with 3 tables.
* 1 Text Files
* 1 CSV File
* 1 Excel file

# Solution architecture

The Data Warehouse solution architecture can be shown as in the following diagram. Some components in the Core Storage Layer and the whole BI-Layer are not drawn since they have not been implemented in this solution.



**Data Sources**

Data Sources column contains all the source files prepared previously. From the source files, all the data are taken into the staging layer through the first **EtL** process which has a smaller number of transformations which mainly focuses on populating the staging layer tables in the staging database.

Various source data taken from the source files such as DB source, flat file sources and excel file sources will be taken into the staging layer (Intermediate storage) and saved in the staging database’s tables. However, other than that, no major transformations were done during the first EtL process while extracting and loading data from source files to the staging database.

**Intermediate Storage (Staging Database only) and EtL**

Staging database in the Intermediate storage layer, acts as an intermediate storage between the Data warehouse and the source files. The purpose of having a stage layer was to leave complex transformations during the first EtL process. The reason is that the EtL from source files/locations is preferred to be faster since there can be other delays such as network traffic, thus performing complex transformations during the extraction from the sources might slow down the whole process which will ultimately result in spending more time than required in that phase, which means the OLTP systems will be busy for more time than they really must. The solution for this was to divide the ETL process into two ETL processes with the staging database in the middle, which will speed up the first EtL process due to less transformations done during the process execution.

Note that the staging database tables are almost identical to the source tables which the data were originally taken from. The data types might have been slightly altered to get rid of data truncation while loading source data. These tables will get fully truncated (or deleted) which will flush out all existing data before the next EtL from sources to staging to only retain up-to-date data in the staging tables.

**Core Storage (Data Warehouse only)**

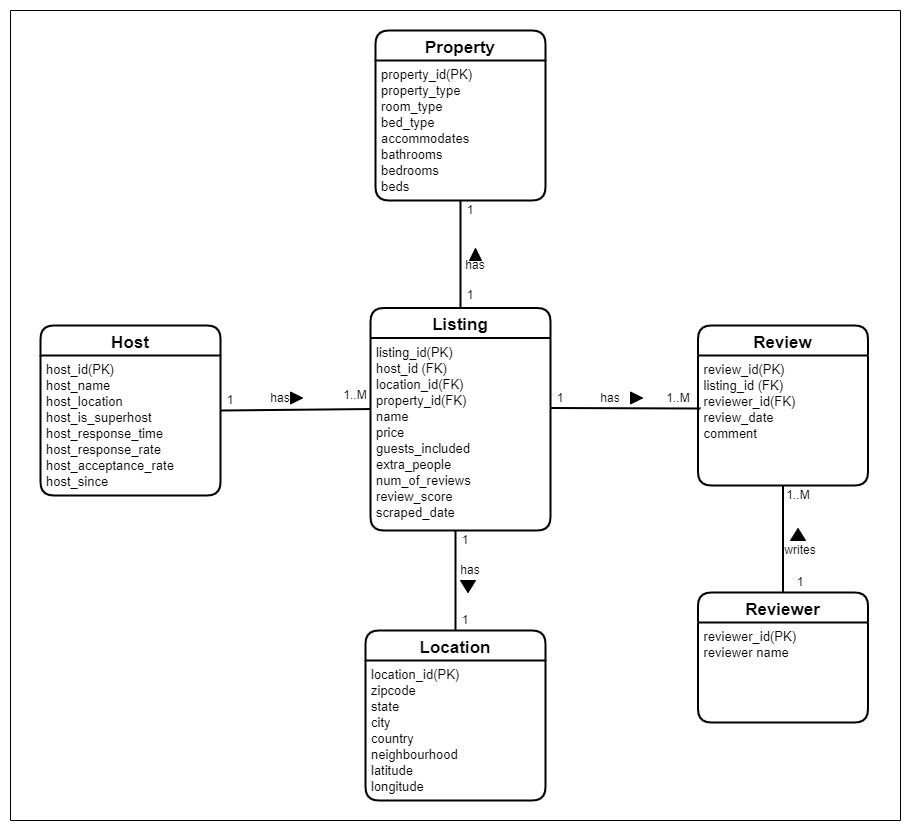
In the core storage layer, there is the Data Warehouse, and it contains the pre-created Dimension and Fact Tables. In this context, there is only one fact table present. The dimension tables have a special type of auto incrementing unique integer key which is known as the surrogate key to identify table records uniquely, while this key is also used to connect dimension tables and fact tables.

The data loaded into the staging database are taken into the Data Warehouse using the **ETL** process which is the second ETL process. The second ETL process has a set of complex Transformations and may take more time to finish the execution than the first EtL process. All the Derived column addition, lookups, table merging, sorting and many other transformations are done in the second ETL process.

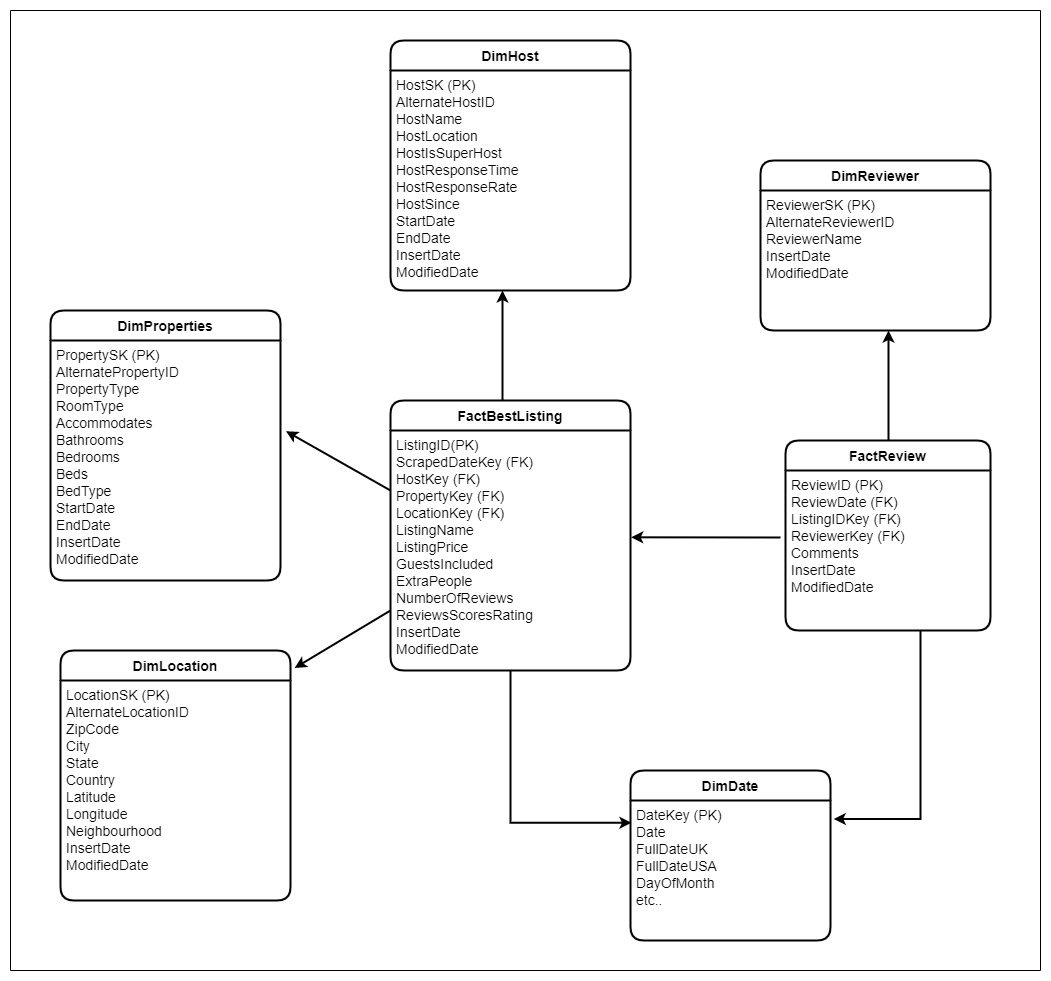
# Data warehouse design & development

In order to correctly identify fact tables, dimension tables and what is the most suitable schema for the data set I initially came up with the relational model and then converted the relational model into the dimensional model

## 4.1. Relational Model Schema

Below diagram is the relational model I identified for the Seattle Airbnb data set.

## 4.2. Dimensional Model Schema

After designing the relational model, it was identified that the most suitable schema for the dimensional modelling is **Star schema with two fact tables.** I identified that Listing and Review should be fact tables as the foreign keys are inside the Listing and Review tables and also most numerical values were inside those tables.

**Dimension Tables**

1. **DimHost –** Granularity by Host name

DimHost contains the essential information about the host such as HostName , HostLocation, HostResponseTime, ResponseRate, host is a superhost or not and since when the host is been asscociated with Airbnb and are represented by unique HostSK for each host. As this contains nouns and they are descriptive I took Host as a dimensional table. Furthermore I implemented DimHost as a slowly changing dimension. So the DimHost table has two derived fields to store StartDate and EndDate to maintain and indicate the historical records**.(Type 2)**

1. **DimProperties –** Granularity by RoomType, Bed Type, number of bed rooms

DimProperties contains thorough information of the facilities available in the Listing such as Property type, room type, bed type and other necessary information which is required to make correct decisions on the Listings. DimProperties table also considered as a slowly changing dimension as the certain attributes of property table also can be changed over the time. So it also contains two derived fields to store StartDate and EndDate to maintain the historical records .**(Type 2)**

1. **DimLocation –** Granularity by Neighborhood

DimLocation contains the location information of a particular Listing such as city, state, country, latitude, longitude and most importantly neighborhood.

1. **DimReviewer**

DimReviewer contains the reviewer id and the name.

1. **DimDate**

DimDate dimension contains date related entities and the values have been pre inserted. DimDate is a static dimension.

**Fact Tables**

For the Seattle Airbnb open data I have used two fact tables and both are coming under “Accumulated Fact Tables”.

1. **FactBestListing**

DimHost, DimProperty and DimLocation are joined using relevant foreign keys to this FactBestListing table. So, it will be able to analyze the Listings based on the Host, Properties of the Listing and Location details.

1. **FactReviews**

DimReviewer and FactBestListing tables have been joined together using relevant foreign keys. Other than that, each review has its own unique review id. Also, it has the review date and the comments also.

## Data Warehouse Implementation

### Staging Layer SQL Database Implementation

Graphical user interface, text

Description automatically generated with medium confidence

List of tables implemented in Staging database "AirBnb\_Seattle \_Staging"

### Graphical user interface, text Description automatically generated with medium confidenceData Warehouse SQL Database Implementation

AirBnb\_Seattle\_DW data warehouse Dimensional model Dim and Fact Tables

**Individual Dim and Fact Table Designs**

**Table

Description automatically generated**

Graphical user interface, table

Description automatically generated DimDate Static Table

DimHost SCD Table

Graphical user interface, table

Description automatically generated with medium confidence

DimLocation Table

Graphical user interface, table

Description automatically generated

DimProperties SCD Table

Graphical user interface, text, application, email

Description automatically generated

DimReviewer Table

Graphical user interface, table

Description automatically generated

FactBestListing Fact Table

Graphical user interface, table

Description automatically generated

FactReview Fact table

# ETL development

As shown in the high-level architectural design (figure 3.1) there are two distinct ETL processes. The first EtL is from Source files/locations to Staging EtL (less transformations) and the next ETL is from Staging to Data Warehouse that was already designed and implemented with empty set of Dimension and Fact tables. The second ETL process which Loads staging data to warehouse is configured to get executed right after the sources-to-staging EtL has finished execution. Within each independent ETL process, the control flow order has been created considering the order of execution as well.

First, a SSIS project has been created using Visual Studio 2017 and all the SQL Server databases related to the Staging and Data Warehouse has been created using SQL Server 2019 Developer edition.

## SSIS Solution, Packages, and Project Connections

The Project has been named as AirBnb\_Seattle\_DW and there are four main packages as follows.

**SSIS Packages**

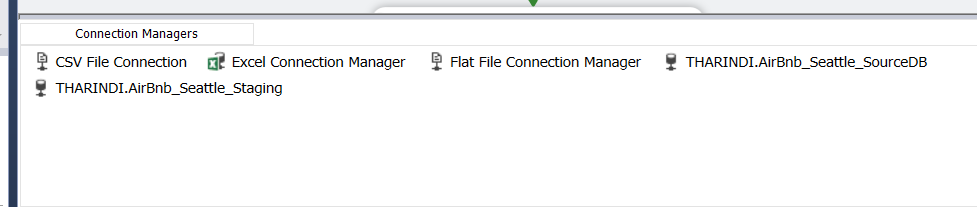
* AirBnb\_Seattle\_Load\_Staging.dtsx – Includes set of ordered Tasks which performs EtL from sources to the staging layer database tables.
* AirBnb\_Seattle\_Load\_DW.dtsx – contains all tasks which gets executed in a specific order which extract data from staging layer tables, do the necessary transformations, and load to the dimensions and fact tables in the data warehouse database.
* AirBnb\_Seattle\_AccumulateFactTables\_Load\_DW**.**dtsx -contains tasks mentioned in Step 6 of the assignment which is to convert fact tables into accumulating fact tables and update the DW fact tables again.
* Data\_Profiling.dtsx **–** contains data profiling tasks.

## Sources to Staging layer EtL

In the first EtL process, the number of transformations is quite close to zero. It is mainly focusing on loading the source data to the staging layer database tables for many valid reasons like reducing the impact on source systems.

In this EtL process the order does not that many matters because foreign key constraints are not validated during the data loading to staging tables.

### Connections used



List of connection Managers created to connect with the source file and the staging database in the AirBnb\_Seattle\_Load\_Staging.dtsx package. Different types of sources required different types of connection managers

|  |  |
| --- | --- |
| **Connection Manager Name** | **Target** |
| THARINDI.AirBnb\_Seattle\_SourceDB | MSSQL source database (AirBnbSeattle\_Host, AirBnbSeattle\_Listings and reviews tables) |
| CSV File Connection | Flat File connection used to load CSV Source  (AirBnbSeattle\_Property.csv) |
| Excel Connection Manager | Used to load AirBnbSeattle\_Reviewers.xlsx source file |
| Flat File Connection Manager | Flat File connection used to load AirBnbSeattle\_Location.txt source file. |
| THARINDI.AirBnb\_Seattle\_Staging | Native OLE DB connection to the Staging database. |

### Tasks of EtL from Sources to Staging Database Tables.

#### Extract Property Data from a CSV file to Staging

A screenshot of a computer

Description automatically generated with medium confidence

Flat File Source and OLE DB Destination components were used to extract Property data from CSV file to the StgProperty table of the MSSQL Staging database “AirBnb\_Seattle\_Staging”. No transformations were done during the process.

Graphical user interface, text, application, chat or text message

Description automatically generated

An “Execute SQL Task” was used inside the relevant event handler of the above-mentioned data flow task on Pre-execution to truncate the StgProperty table beforehand loading takes place to clear any previously staged data.

#### Extract Listings Data to Staging

A picture containing graphical user interface

Description automatically generated

OLE DB Source and OLE DB Destination components were used to extract Listings data from AirBnbSeattle\_Listings source database table to the StgListings table of the MSSQL Staging database “AirBnb\_Seattle\_Staging”. No transformations were done during the process.

Graphical user interface, text, application, chat or text message

Description automatically generated

Similar in the previous case, an OnPreExecute event handler was used to create a Execute SQL Task to truncate StgListings table before loading the new data.

#### Extract Location Data to Staging

A picture containing graphical user interface

Description automatically generated

Data Flow of the “Extract Location Data to Staging”. Data Flow Task uses a Flat file Source and then loads them to StgLocation table of the Staging database “AirBnb\_Seattle\_Staging” in MSSQL Server.

Graphical user interface, text, application, chat or text message

Description automatically generated

An event handler was used in the similar way shown in previous cases, to truncate the StgLocation table of the staging database prior to the EtL Task by specifying “OnPreExecute”.

#### Extract Reviewers Data to Staging

Graphical user interface, application

Description automatically generated

Reviewers’ data are getting extracted using an Excel source that connects to the excel file and using a OLE DB destination that is load into the staging database table, “StgReviewers”.

Graphical user interface, text, application, chat or text message

Description automatically generated

An event handler is in place and configured to run OnPreExecute to truncate the StgReviewers table before loading.

#### Extract Reviews Data to Staging

A picture containing graphical user interface

Description automatically generated

OLE DB Source and OLE DB Destination components were used to extract Reviews data from reviews source database table to the StgReviews table of the MSSQL Staging database “AirBnb\_Seattle\_Staging”. An event handler is in place to truncate the StgReviews table before loading data to it. No transformations are done during this data flow.

Graphical user interface, text, application, chat or text message

Description automatically generated

#### Extract Host Data to Staging

Diagram

Description automatically generated with medium confidence

OLE DB Source and OLE DB Destination components were used to extract Host data from AirBnbSeattle\_Host source database table to the StgHosts table of the MSSQL Staging database “AirBnb\_Seattle\_Staging”. An event handler is in place to truncate the StgHosts table before loading data to it. No transformations are done during this data flow.

**Graphical user interface, text, application

Description automatically generated**

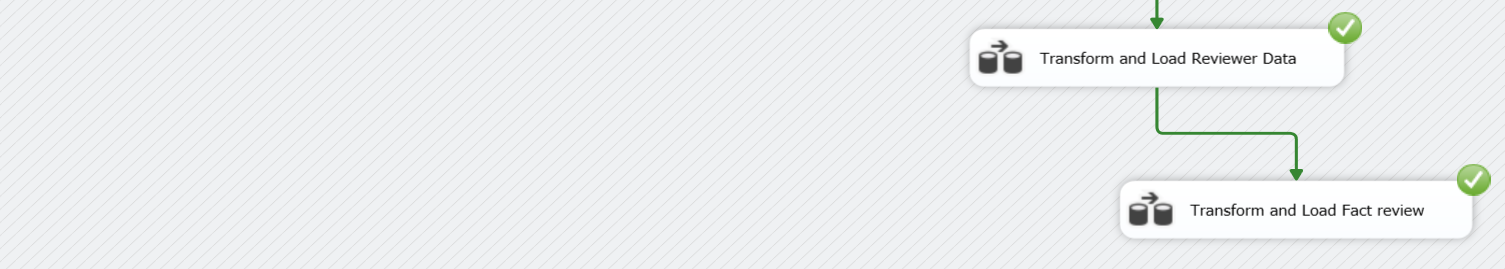
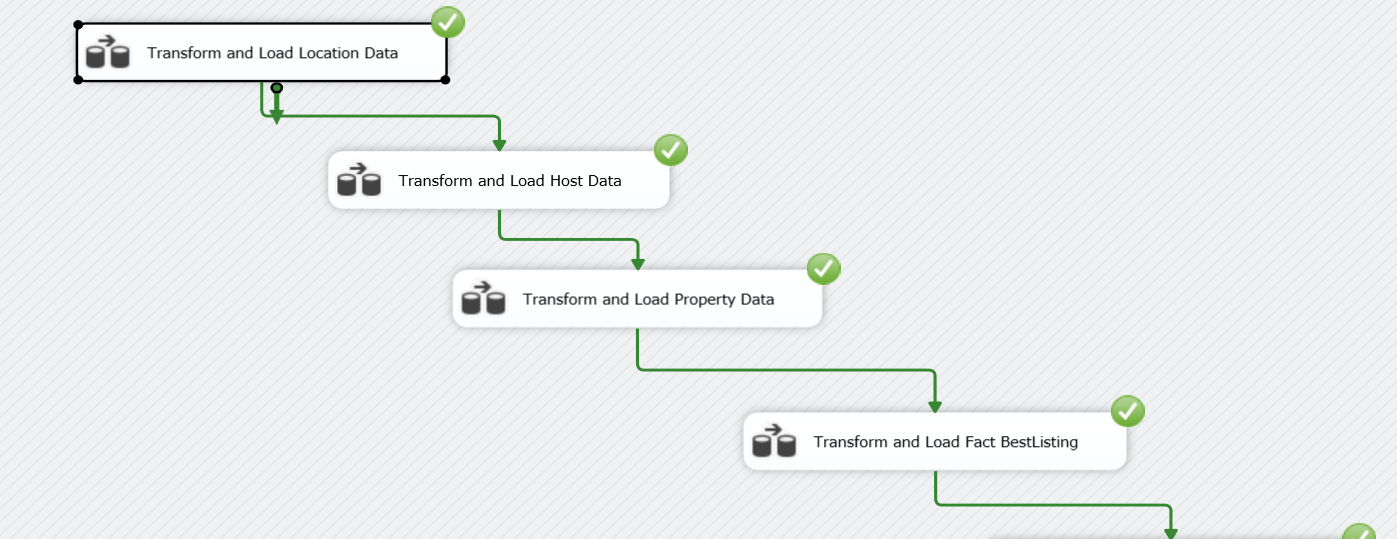
Similarly, an event handler is placed on pre-execution to clear the previously stored host data from the staging database table before the newly extracting data are stored

## Staging layer to Data Warehouse ETL

The second ETL process does more work than the first EtL explained earlier. As mentioned before, the earlier set of tasks had almost no transformations. However, this process has a lot of transformations to be done to the data before they can be loaded into the data warehouse dimension and fact tables.

Although in the first EtL Task order was not that much important, the second set of ETL task execution order is the most important part. So, an execution order was planned and the ETL tasks were added accordingly. Business keys and surrogate keys were used to preserve the links between dim and fact tables.

A separate package was created to implement these named, AirBnb\_Seattle\_Load\_DW.dtsx.



Data Flow tasks have not been assigned any event handlers to truncate any of the data warehouse tables because truncating data warehouse tables may cause issues of unexpected surrogate key changes. Data for all data warehouse tables are either updated or inserted only.

### Connections used

Graphical user interface, text, application

Description automatically generated

### Tasks of ETL from Staging to Data Warehouse in their Execution Order

#### Transform and Load Location Data to DimLocation

Diagram

Description automatically generated

As shown in the above figure an OLE DB source was added to extract the data from staging layer table “StgLocation” and since there are no transformations to be done, the extracted data is sent to an OLE DB command component to pass the data to a stored procedure in the data warehouse to do the update or insertion accordingly. The relevant stored procedure in this case is “UpdateDimLocation”. Stored procedure call: **exec dbo.UpdateDimLocation ?, ?, ?, ?, ?, ?, ?, ?**

CREATE PROCEDURE dbo.UpdateDimLocation

@location\_id int,

@city nvarchar(50),

@state nvarchar(50),

@zipcode nvarchar(15),

@country nvarchar(50),

@latitude nvarchar(50),

@longitude nvarchar(50),

@neighbourhood nvarchar(50)

AS

BEGIN

if not exists (select LocationSK

from dbo.DimLocation

where AlternateLocationID = @location\_id)

BEGIN

insert into dbo.DimLocation

(AlternateLocationID, City, State, ZipCode, Country, Latitude, Longitude, Neighbourhood, InsertDate, ModifiedDate)

values

(@location\_id, @city, @state, @zipcode, @country, @latitude, @longitude, @neighbourhood, GETDATE(), GETDATE())

END;

if exists (select LocationSK

from dbo.DimLocation

where AlternateLocationID = @location\_id)

BEGIN

update dbo.DimLocation

set City = @city,

State = @state,

ZipCode = @zipcode,

Country = @country,

Latitude = @latitude,

Longitude = @longitude,

Neighbourhood = @neighbourhood,

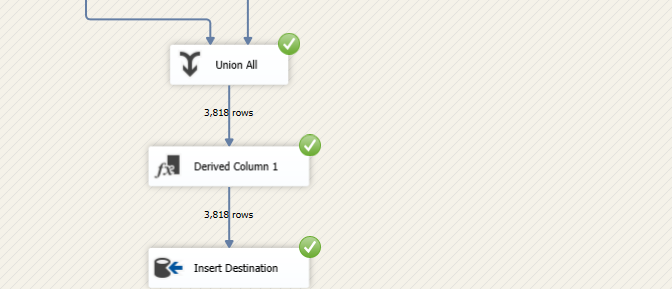
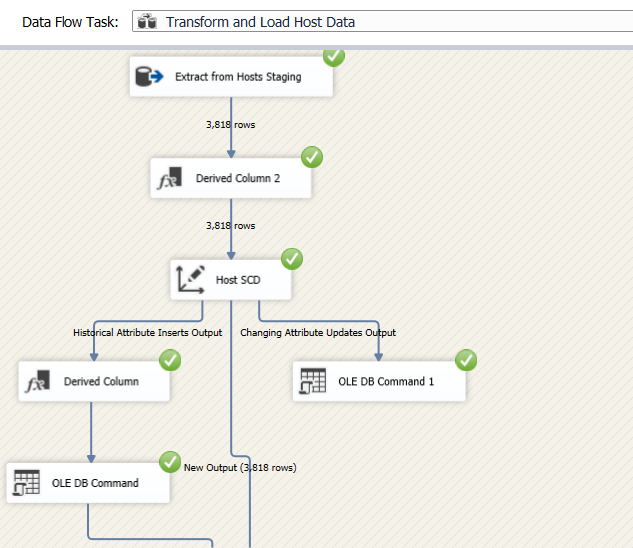
ModifiedDate = GETDATE()

where AlternateLocationID = @location\_id

END;

END;

#### Transform and Load Host Data to SCD DimHost



Like in the first data flow process, here also the Hosts data extracted from the staging table “StgHosts” using the OLE DB Source component. Then I added a derived column component to add InsertDate and ModifiedDate columns and the value is taken as the current timestamp from the GETDATE() function which has given by the Derived Column Component by default.Then I developed whole Host dimension as a Slowly Changing Dimension. I maintained slowly changing Host dimension attributes under following types.

Type 1(Changing) – HostAcceptanceRate, HostResponseRate, HostResponseTime, HostSince

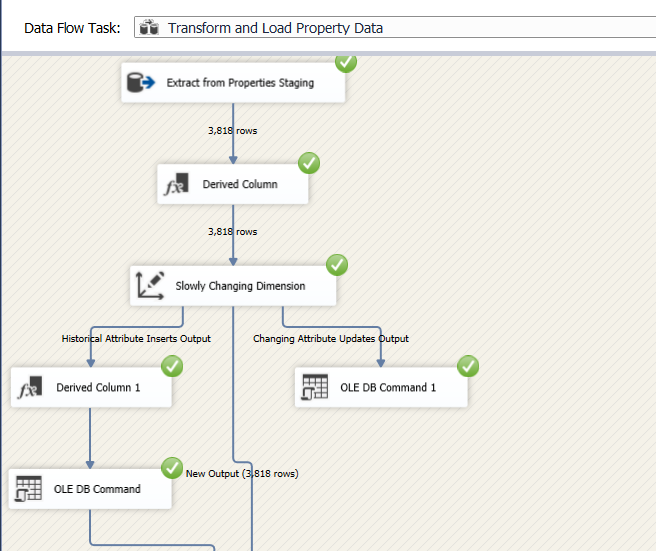
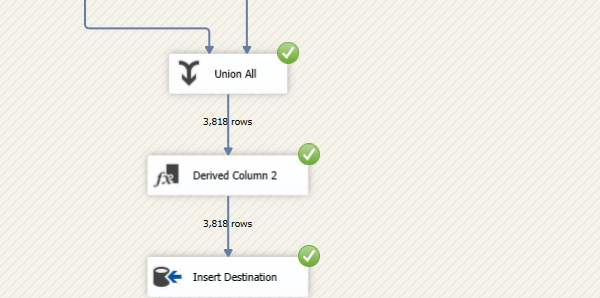
Type2(Historical) – HostName, HostISSuperhost

#### Transform and Load Property Data to DimProperties SCD

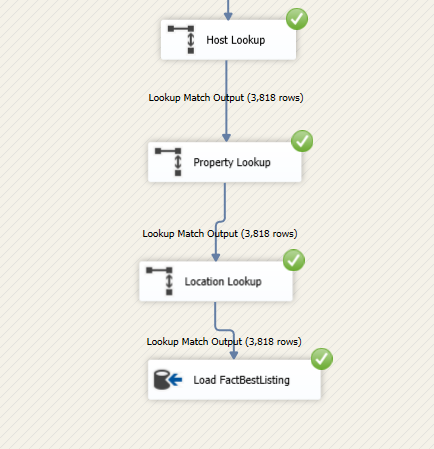
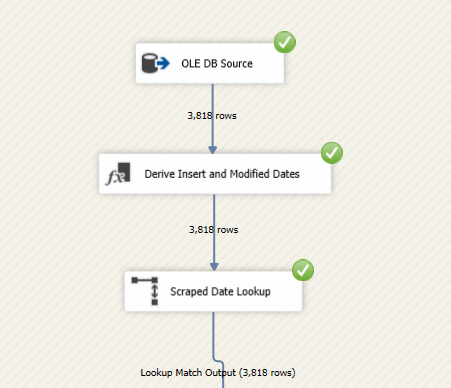
Here also “DimProperties” data warehouse table has considered as a Slowly Changing Dimension Table. First Property data is extracted from “StgProperty” table using OLE DB Source component. Then I added a Derived Column to add InsertDate and ModifiedDate columns.

After that as in the previous one I added Property dimension as a slowly changing dimension and finally load the data to the DimProperties dimension in the ‘AirBnb\_Seattle\_DW’.

In here Property Type column of the DimProperties table is maintained as a Historical attribute(Type 2). All the other attributes are maintained as Changing attributes(Type1).



#### Transform and Load Fact BestListing



Above dataflow belongs to the ETL task of loading the BestListing fact table. As this fact table refers to the Date, Host, Property and Location dimension tables lookups were used to get the relevant surrogate keys to establish the references between tables.

After extracting StgListings data from the staging database table, First a derived column added to add the InsertDate and ModifiedDate columns.

Then first lookup is done to retrieve the date key for the relevant scraped date.

Graphical user interface

Description automatically generated

2nd lookup is done to retrieve the HostSK from the DimHost.

Graphical user interface, application

Description automatically generated

3rd lookup is done to retrieve the PropertySK from the DimProperties.

Graphical user interface

Description automatically generated

4th lookup is done to retrieve the LocationSK from the DimLocation.

Graphical user interface, application

Description automatically generated

Finally output is sent to an OLE DB Destination component which refers to the FactBestListing table in the data warehouse.

#### Transform and Load DimReviewer

Diagram

Description automatically generated

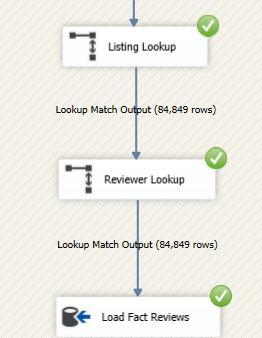
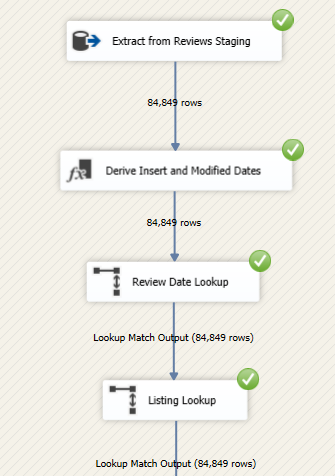
As DimReviewers table has no references to above loaded tables I considered to load DimReviewers table after loading FactBestListing table also. Then as in previous ones here also added a Derived Column to add InsertDate and ModifiedDate columns. Then finally DimReviewers table load to the data warehouse.

#### Transform and Load FactReview table

Like in the first data flow process, here also data extracted from the staging table “StgReviews” using the OLE DB Source component and then derived column component added to insert the current timestamp from the GETDATE() function to add InsertDate and ModifiedDate columns.

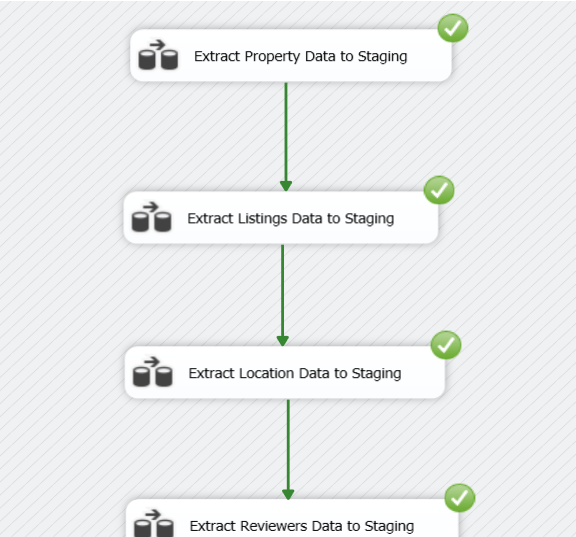
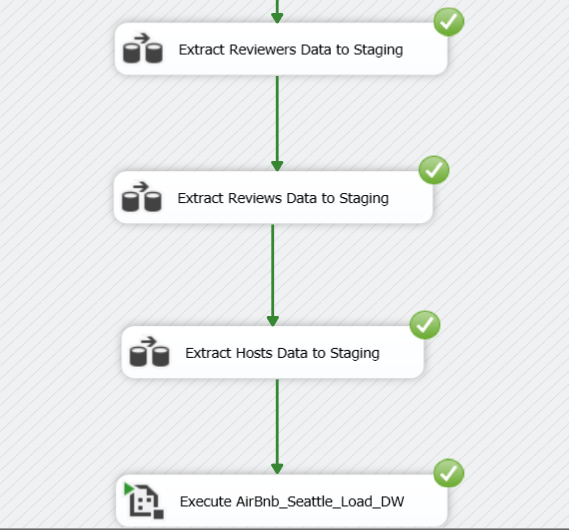
As in the previous fact table (FactBestListing), here also has few lookups to establish the references between tables. In here there are three lookups which refers to the DimDate, FactBestListing and DimReviewer tables.

At last output is sent to an OLE DB Destination component which refers to the FactReviews table in the data warehouse

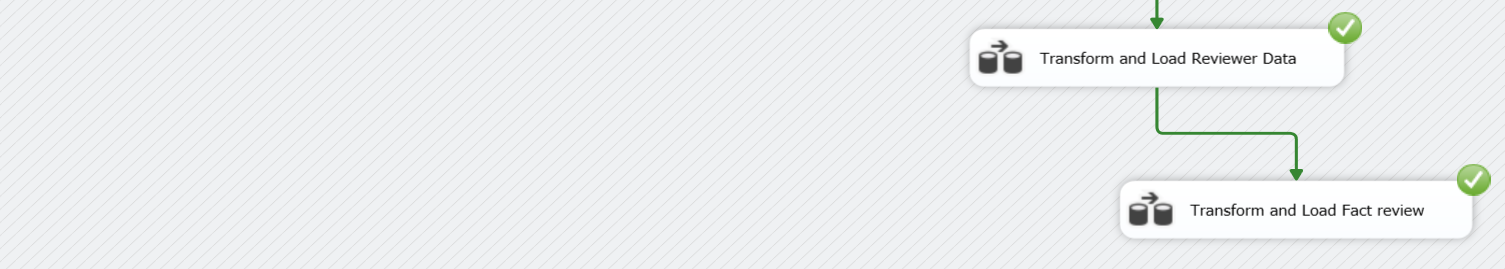
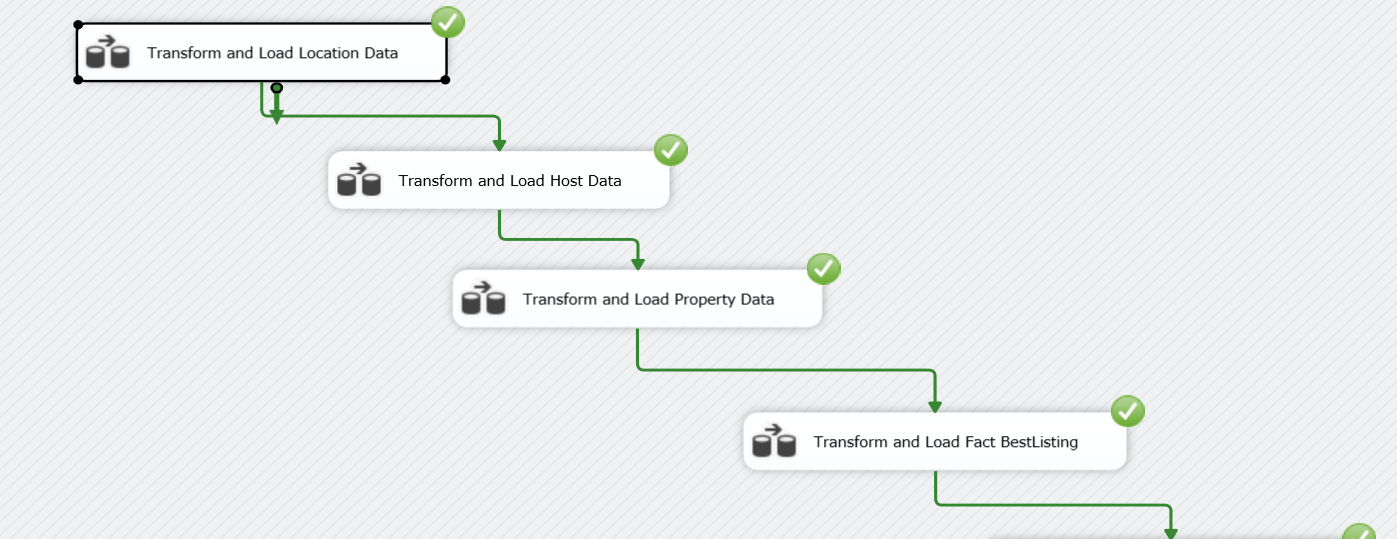


.

Then finally to connect AirBnb\_Seattle\_Load\_Staging.dtsx and AirBnb\_Seattle\_Load\_DW.dtsx, I used an Execute Package Task in a way that ‘Execute AirBnb\_Seattle\_Load\_DW’ component gets executed as the last step.



After successfully executing these tasks in the next step it executed AirBnb\_Seattle\_Load\_DW.dtsx package also as follows.



# ETL development – Accumulating Fact Tables

## Loading updated Accumulated Fact Table FactBestListing

In the task 6, I first extend the Fact BestListing table by adding 2 columns named as accm\_txn\_complete\_time and txn\_process\_time\_hours. Then factBestListing fact table’s InsertDate column renamed as the accm\_txn\_create\_time.

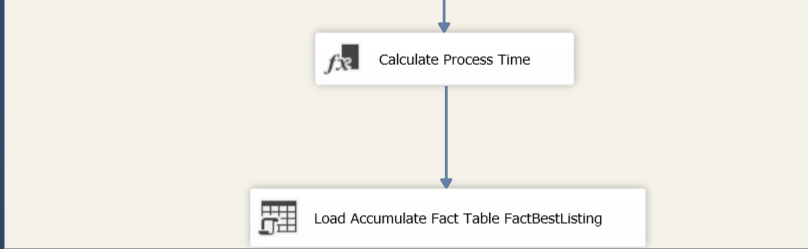
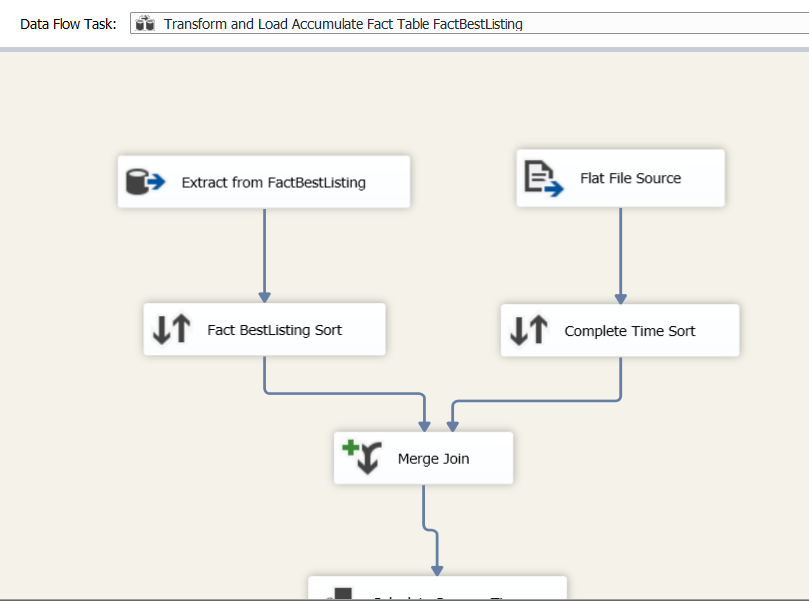
Then prepared a separate csv file named as complete\_time\_source by adding the 2 columns named as fact\_table\_natural\_key and accm\_txn\_complete\_time. FactBestListing table’s natural key which is the ListingID values are taken as the fact\_table\_natural\_key column values.

Then for the accm\_txn\_complete\_time column values, I generated random time for the date ahead of 2 days. Following is the function I used to create the random time.

=TEXT(RAND()\*("2022-5-12 12:00:00"-"2022-5-12 10:00")+"2022-5-12 10:00:00","YYYY-MM-DD HH:MM:SS")

Then created a package in SSDT named as “AirBnb\_Seattle\_AccumulateFactTables\_Load\_DW.dtsx” to update the FactBestListing table values.

In the control flow added a data flow task and renamed it as “Transform and Load Accumulate Fact Table FactBestListing”. Then implemented the relevant ETL task in the data flow sectionas follows.



Using OLE DB Source component and Flat file source component extracted data from FactBestListing and complete\_time\_process csv file respectively.

Then both are sorted using Sort components at the end. Sorting is done to aid faster joining of the two tables. Sorting is particularly done according to the fact table’s natural key which is the ListingID in both and the sorted rows of two tables are joined and configured using a merge join component. In the merge join, output columns was carefully ticked.

**Graphical user interface

Description automatically generated**

Next I added a derived column to calculate Process time.

Graphical user interface, text

Description automatically generated

To get the difference between the complete time and the create time I used DATEDIFF () function which has given by the Derived Column Component by default.

Graphical user interface, text, application, email

Description automatically generatedThen finally by using an OLE DB Command component, loaded the updated accumulated fact table to the Datawarehouse.

Graphical user interface

Description automatically generated with medium confidence

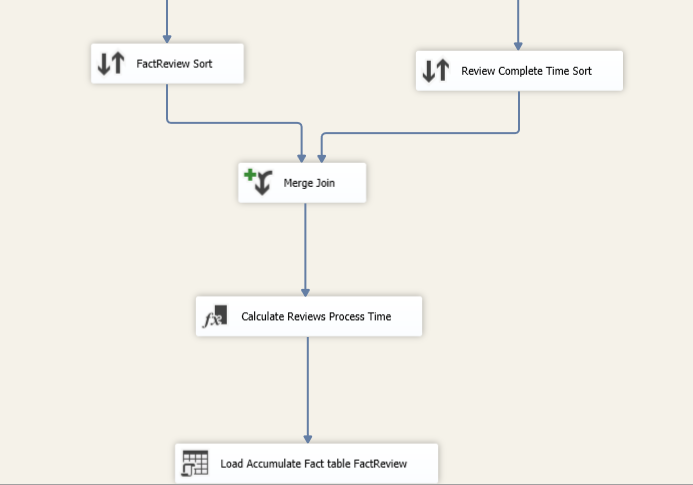
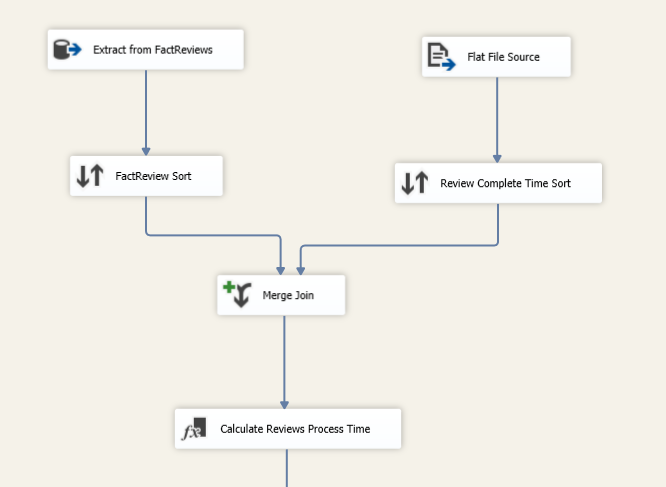
Graphical user interface, table, Excel

Description automatically generatedSo after loading the factBestListing table to data warehouse newly added columns values had been added in this way.

## Loading updated Accumulated Fact Table FactReview

As I have 2 fact tables I done the same procedure to the FactReview fact table also. In there also I created a separate csv file name and named it as complete\_time\_reviews.csv.

Following are the screenshots taken from the Visual Studio.



Graphical user interface

Description automatically generatedSort components

Graphical user interface, application

Description automatically generated

Table

Description automatically generatedMerge Join

Graphical user interface, text, application, email

Description automatically generatedLoading updated Accumulated Fact Table FactReview

Graphical user interface

Description automatically generated

(Note -> Both csv files which means complete\_time\_source.csv (Corresponds to FactBestListing) and complete\_time\_reviews.csv(Corresponds to FactReview) are inside the DataSources folder and I didn’t create separate staging tables Instead I loaded them directly to the data warehouse.)

So after loading the FactReview table to data warehouse ; DW table looked kied follows.

A picture containing table

Description automatically generated