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

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

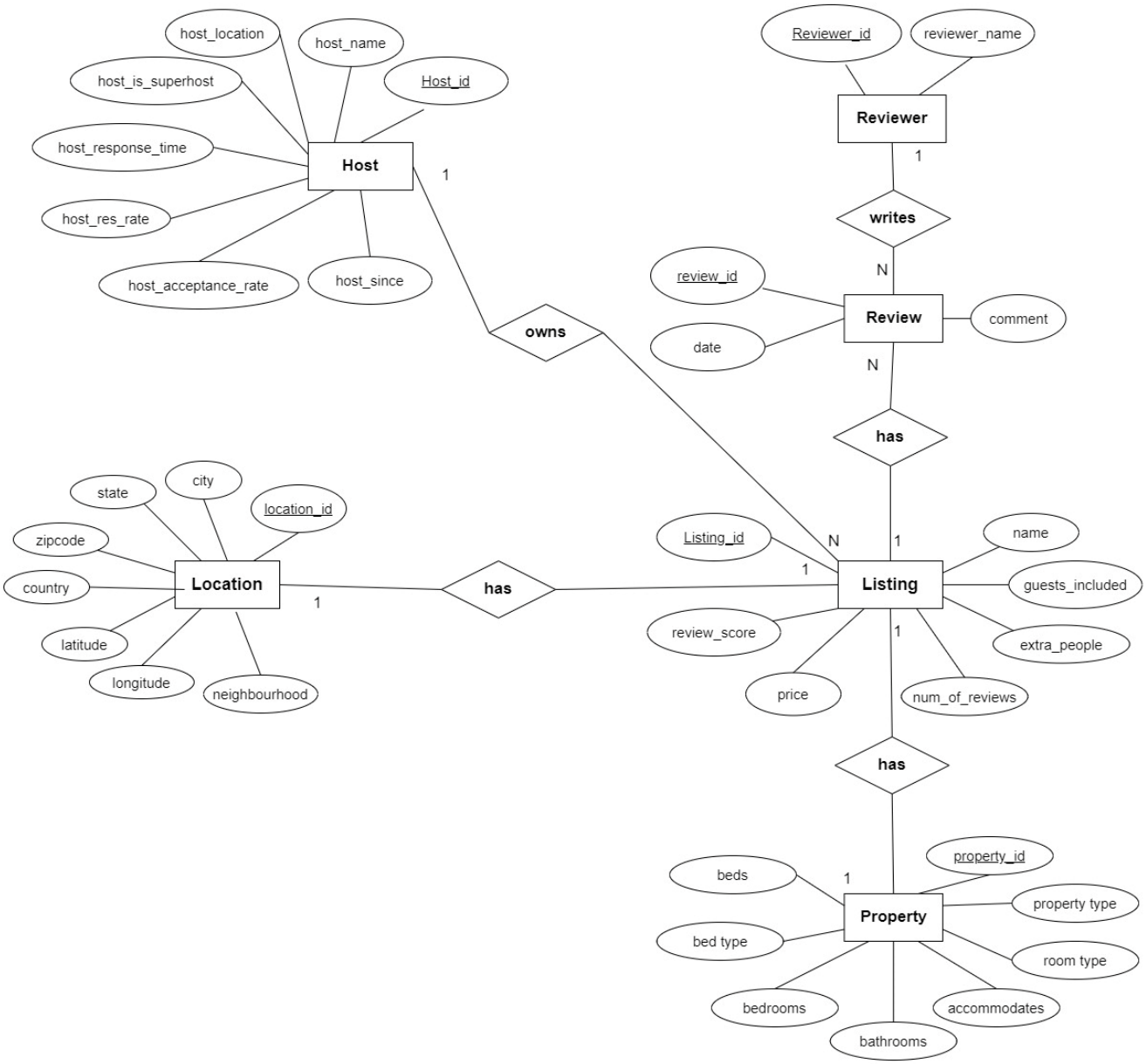
For the purpose of this assignment, I focused only about the Listings and Reviews. The Listings data file comprises of 3813 listings and the Reviews dataset comprises of 84849 review records.

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.



2. 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.

2. 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.

3. 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.

4. 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.

5. Reviewer Information →

- Reviewer id and reviewer name were separated into an **excel file named AirBnbSeattle_Reviewers.**
- It contains only reviewer_id and reviewer_name.

6. 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.

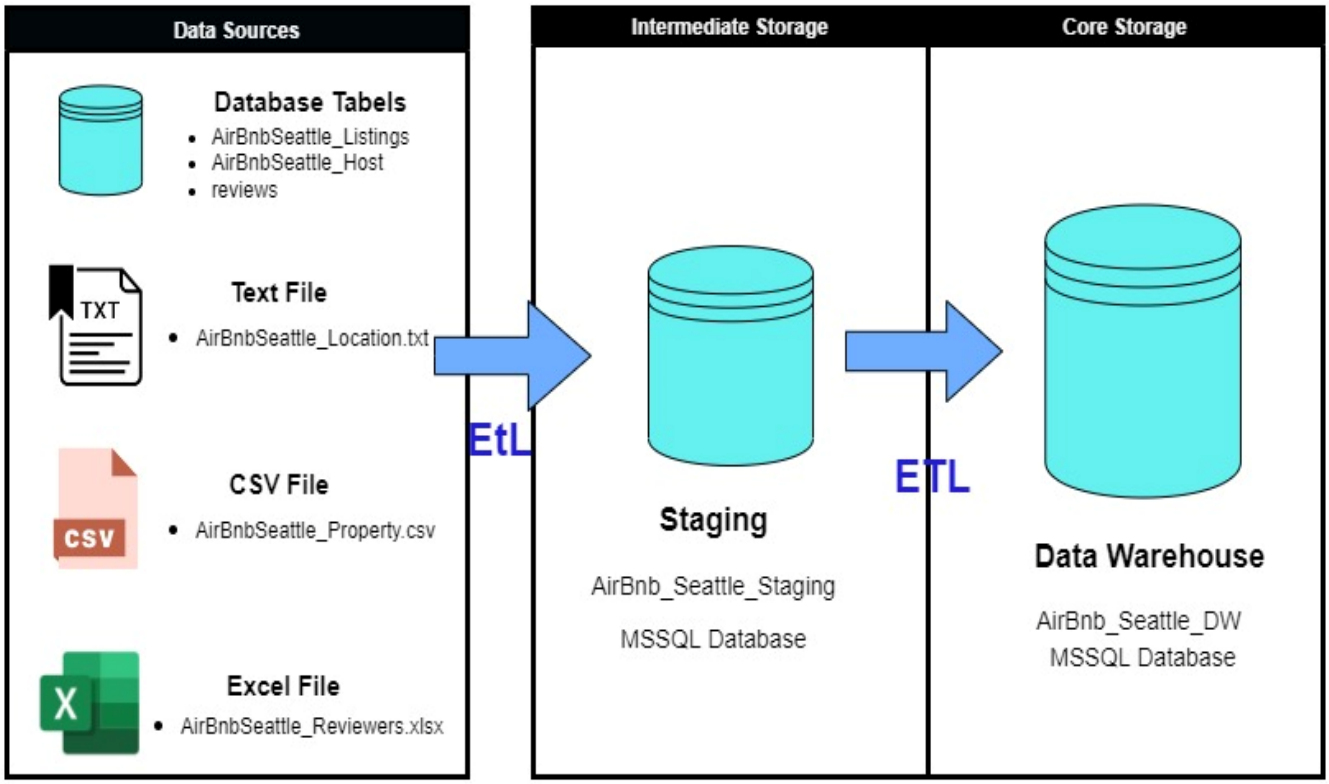
As mentioned above in each table I have included primary keys and relevant foreign keys were also included into the tables.

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

3. 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.

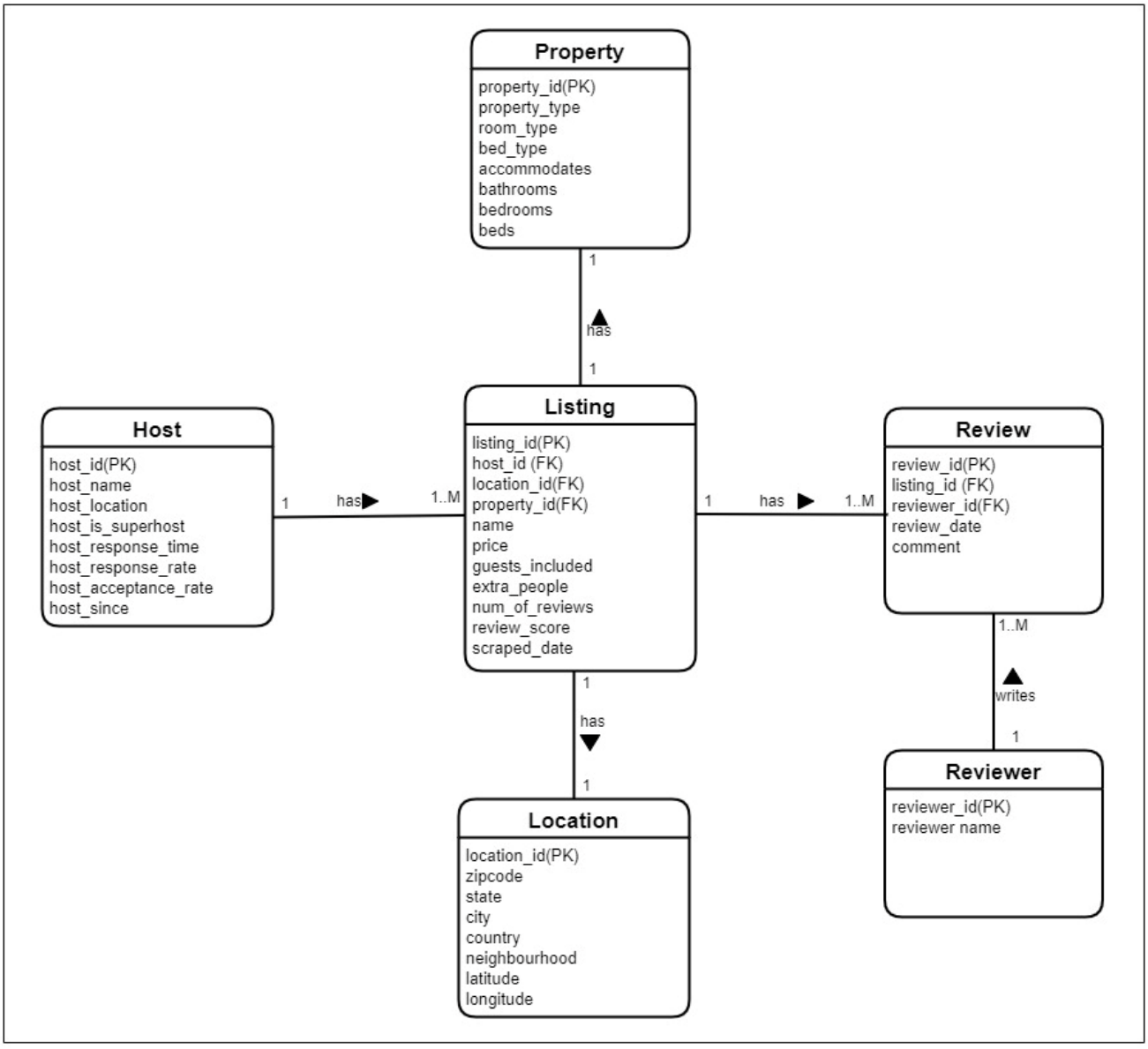
The data loaded into the staging database are taken into the Data Warehouse using the **ETL** process which is the second ETL process. The second ETL process has a set of complex Transformations and may take more time to finish the execution than the first EtL process. All the Derived column addition,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

4. 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

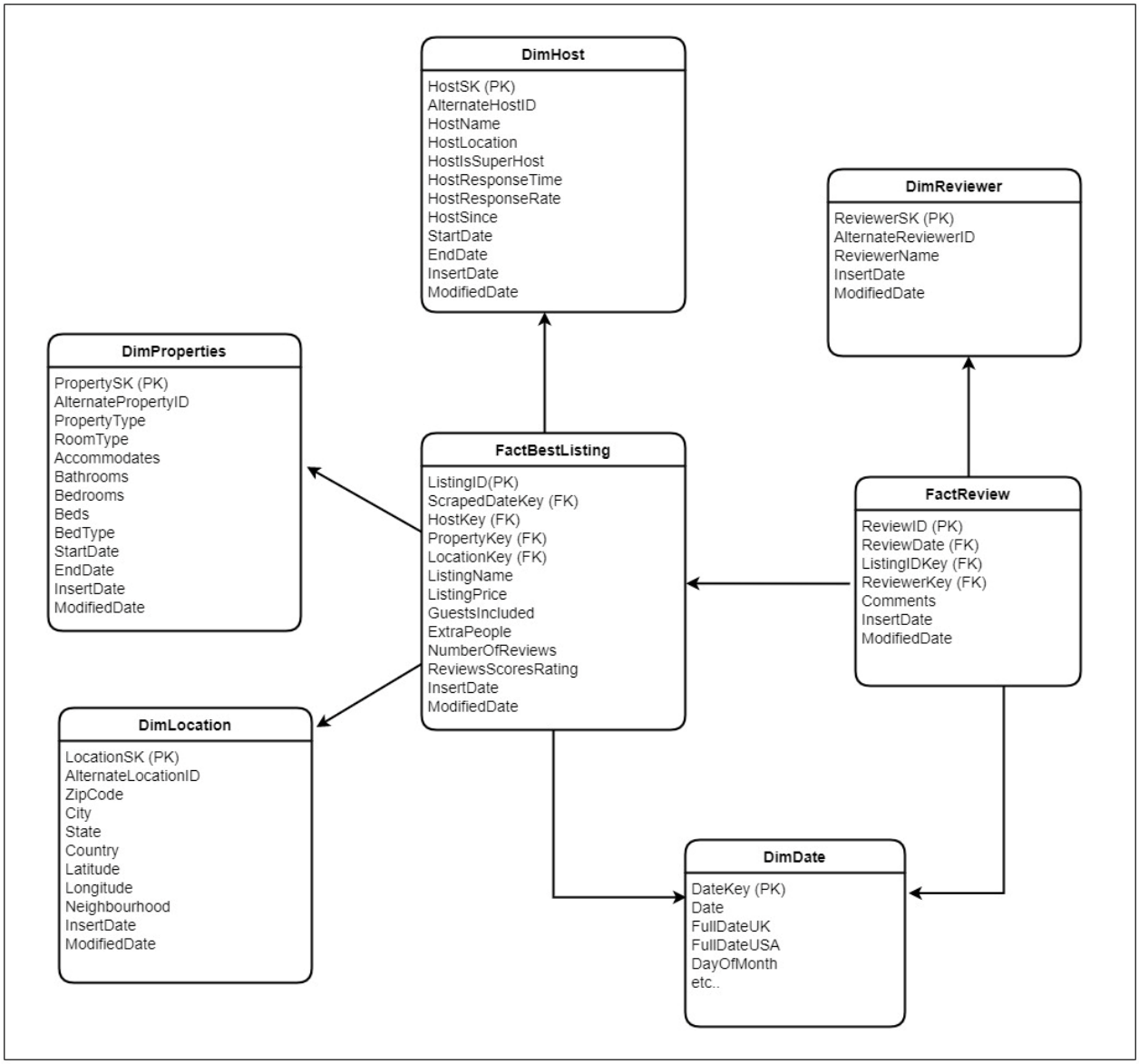
4.1. Relational Model Schema

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)**

2. **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)**

3. DimLocation – Granularity by Neighborhood

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

4. DimReviewer

DimReviewer contains the reviewer id and the name.

5. 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

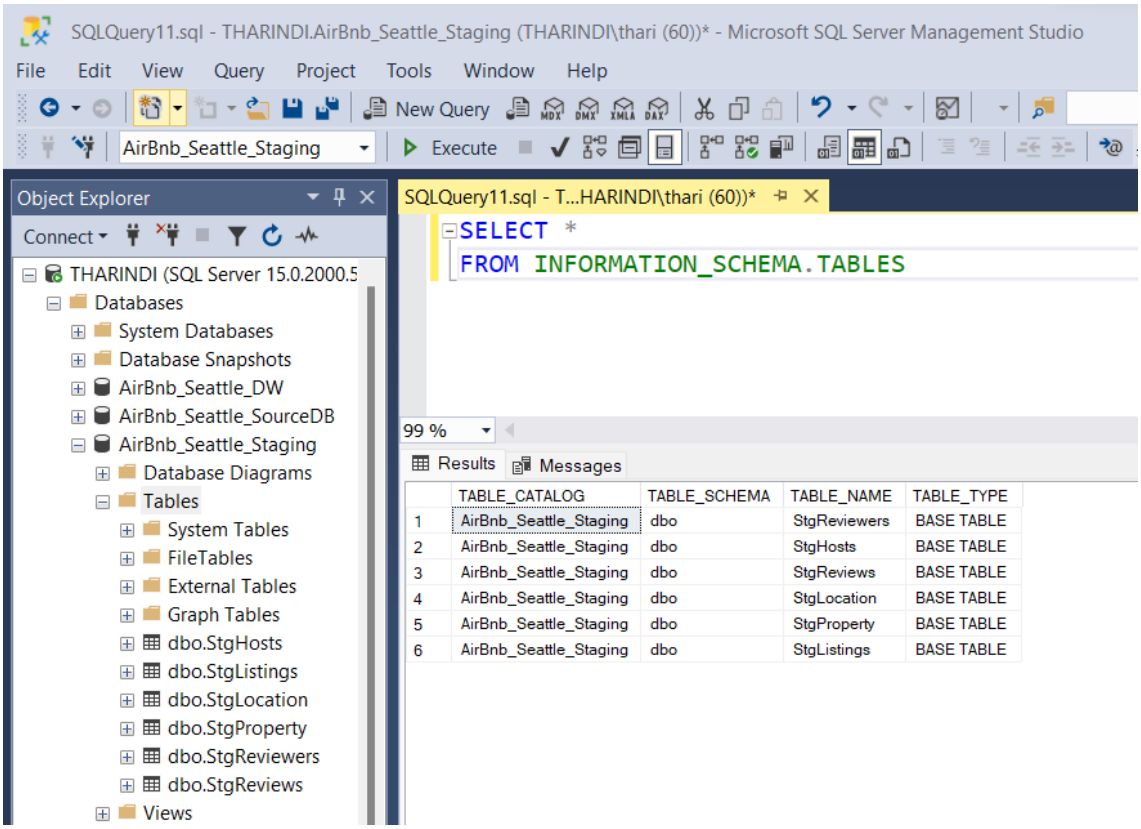
DimHost, DimProperty and DimLocation are joined using relevant foreign keys to this FactBestListing table. So, it will be able to analyze the Listings based on the Host, Properties of the Listing and Location details.

2. 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.

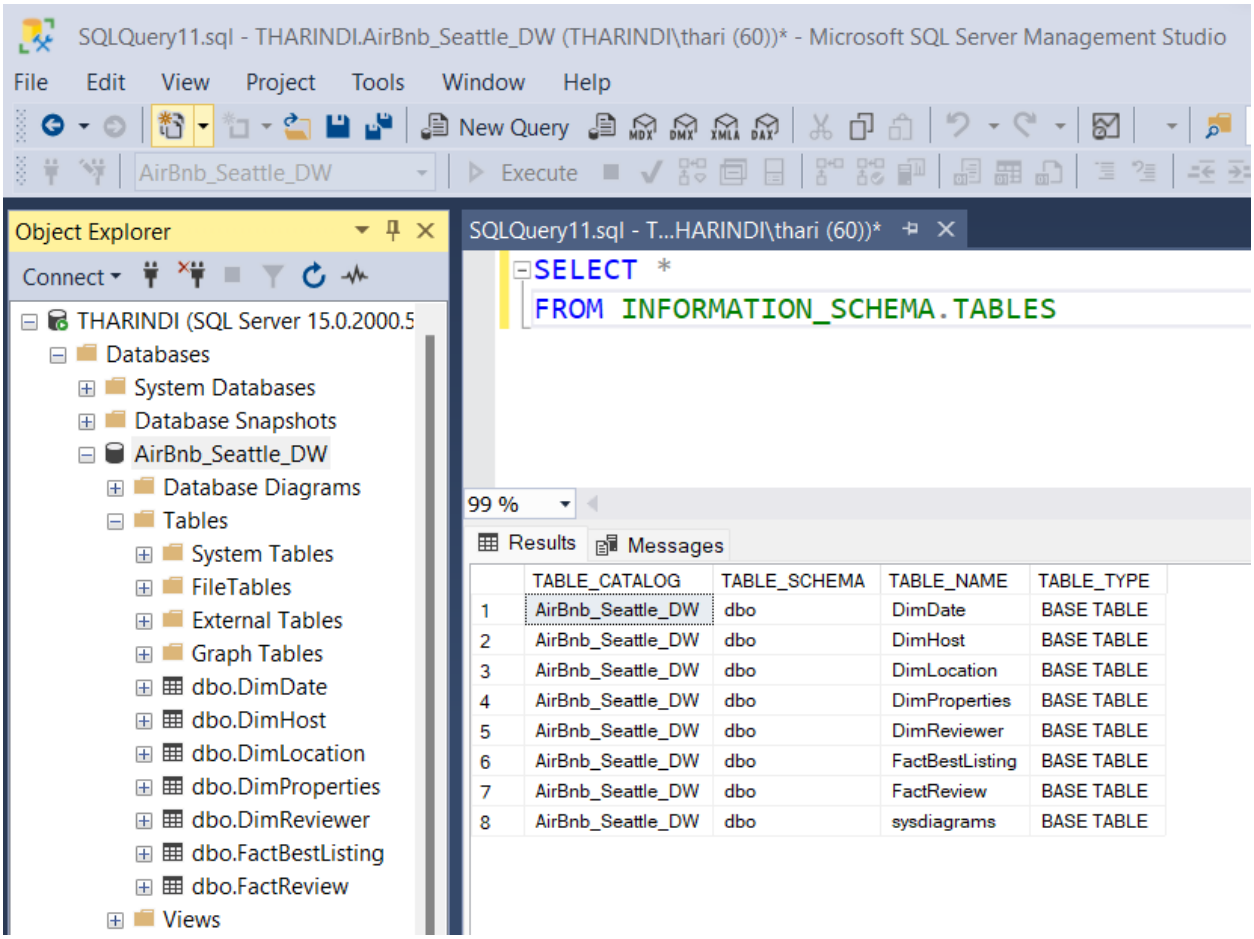
4.1. Data Warehouse Implementation

4.1.1. Staging Layer SQL Database Implementation



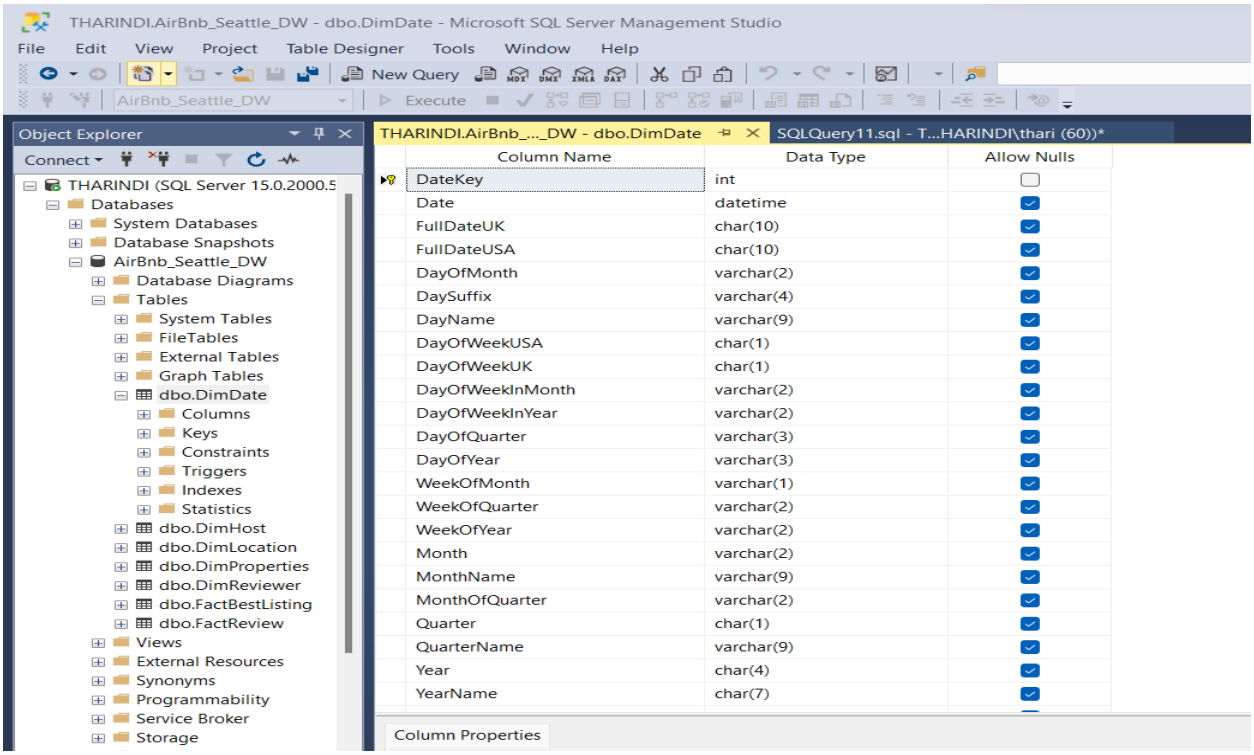
List of tables implemented in Staging database "AirBnb_Seattle _Staging"

4.1.2. Data Warehouse SQL Database Implementation

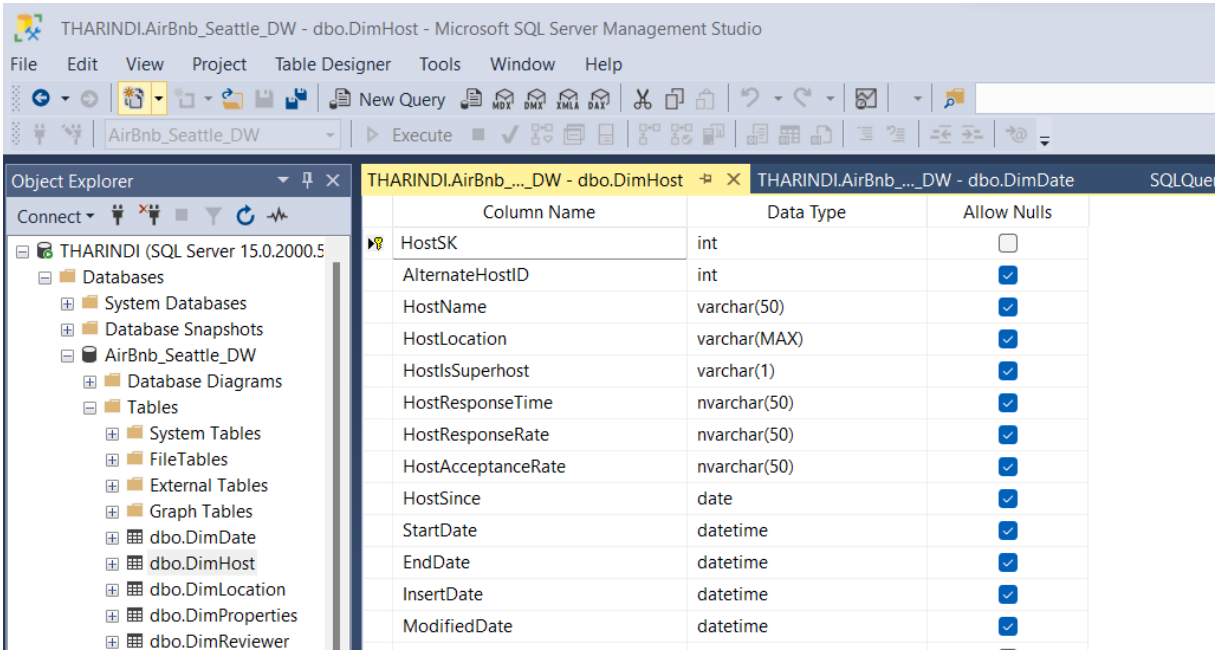


AirBnb_Seattle_DW data warehouse Dimensional model Dim and Fact Tables

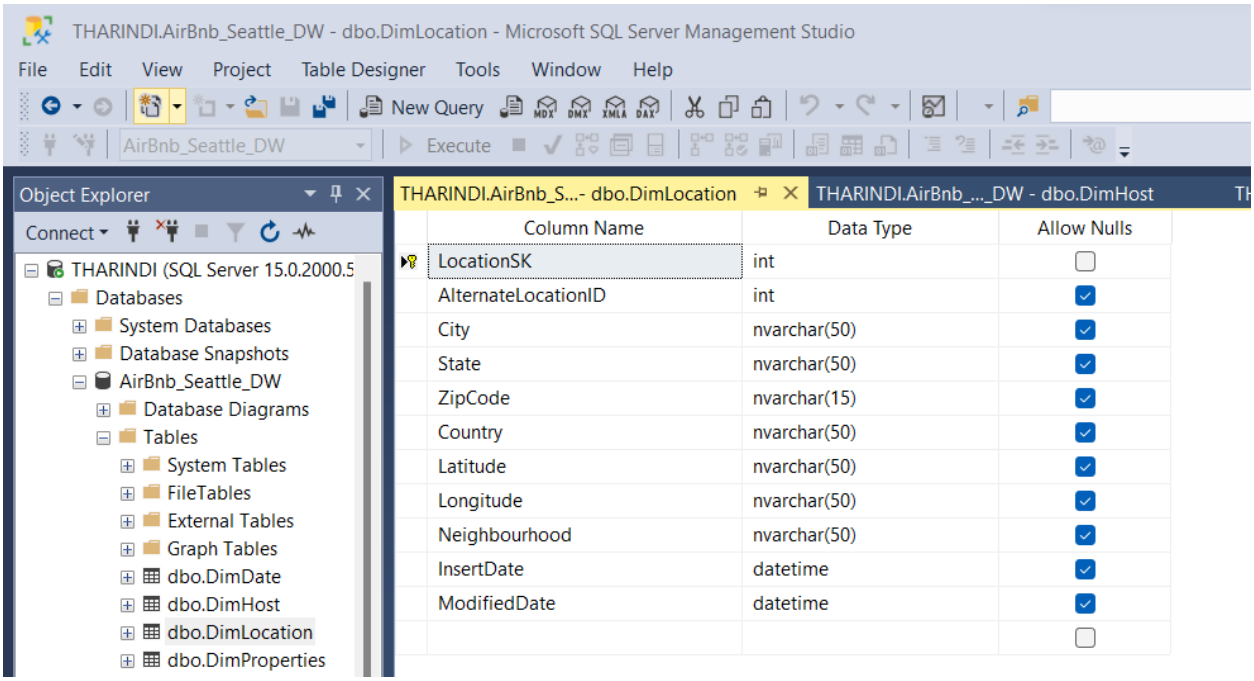
Individual Dim and Fact Table Designs



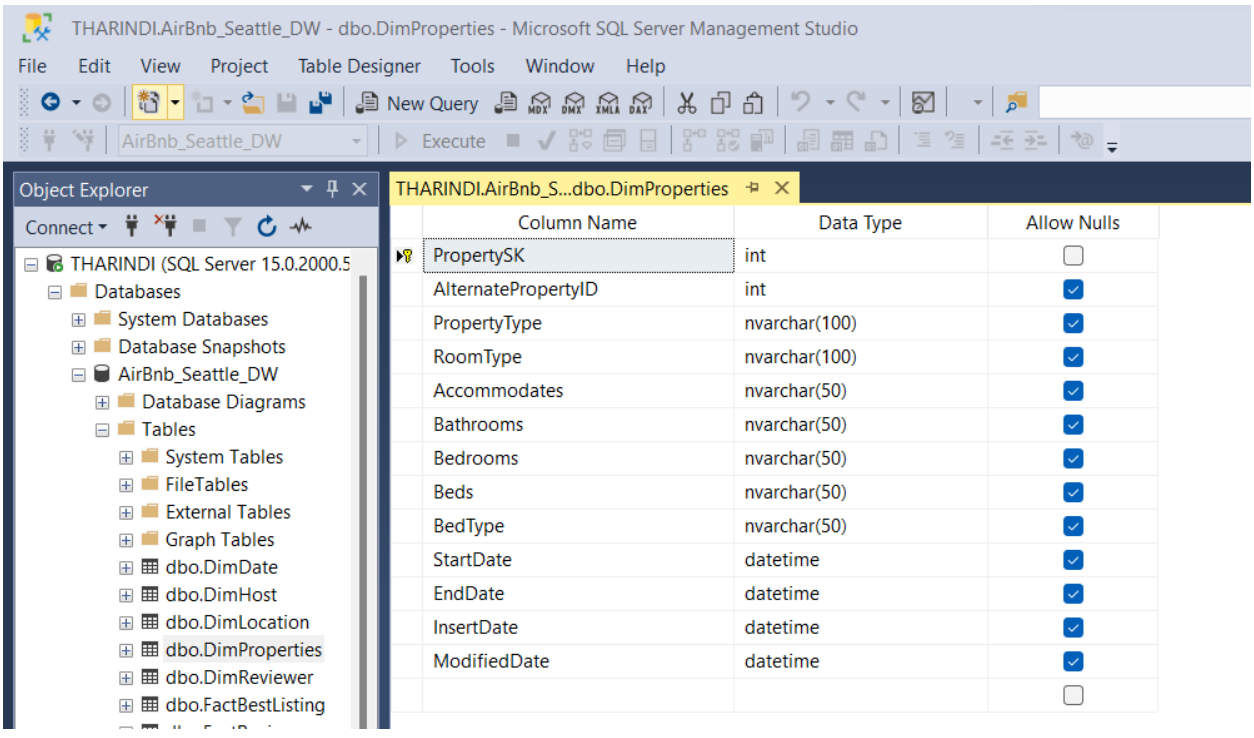
DimDate Static Table



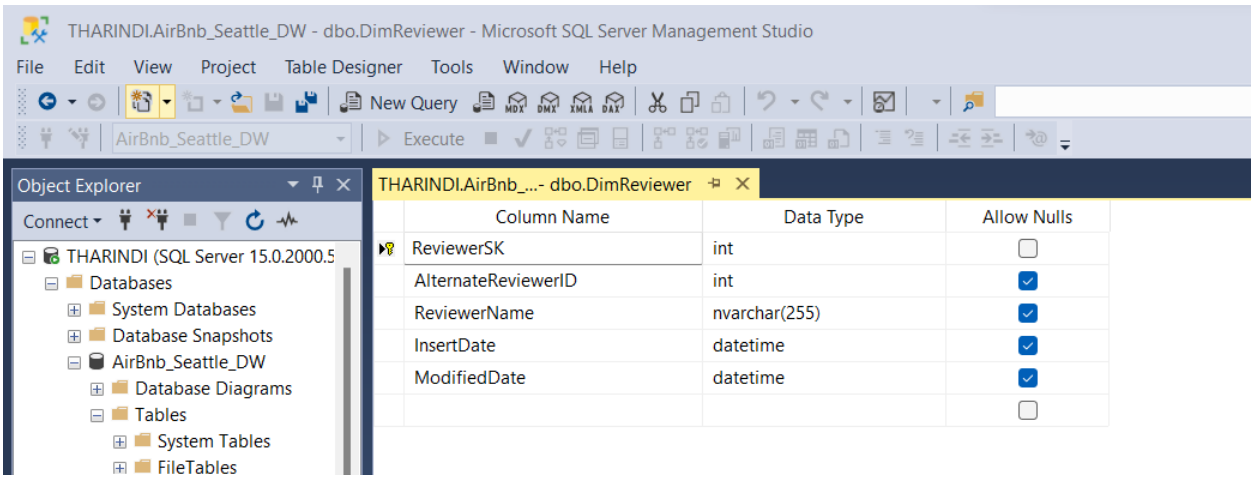
DimHost SCD Table



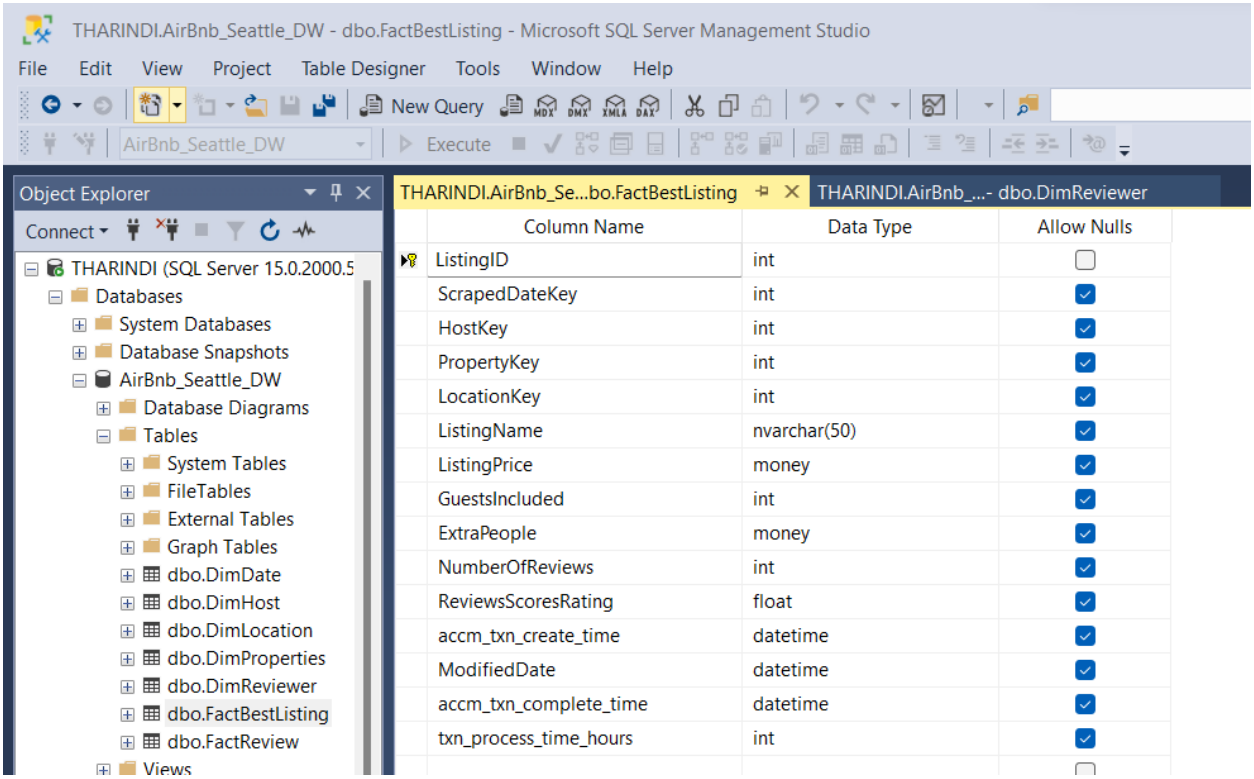
DimLocation Table



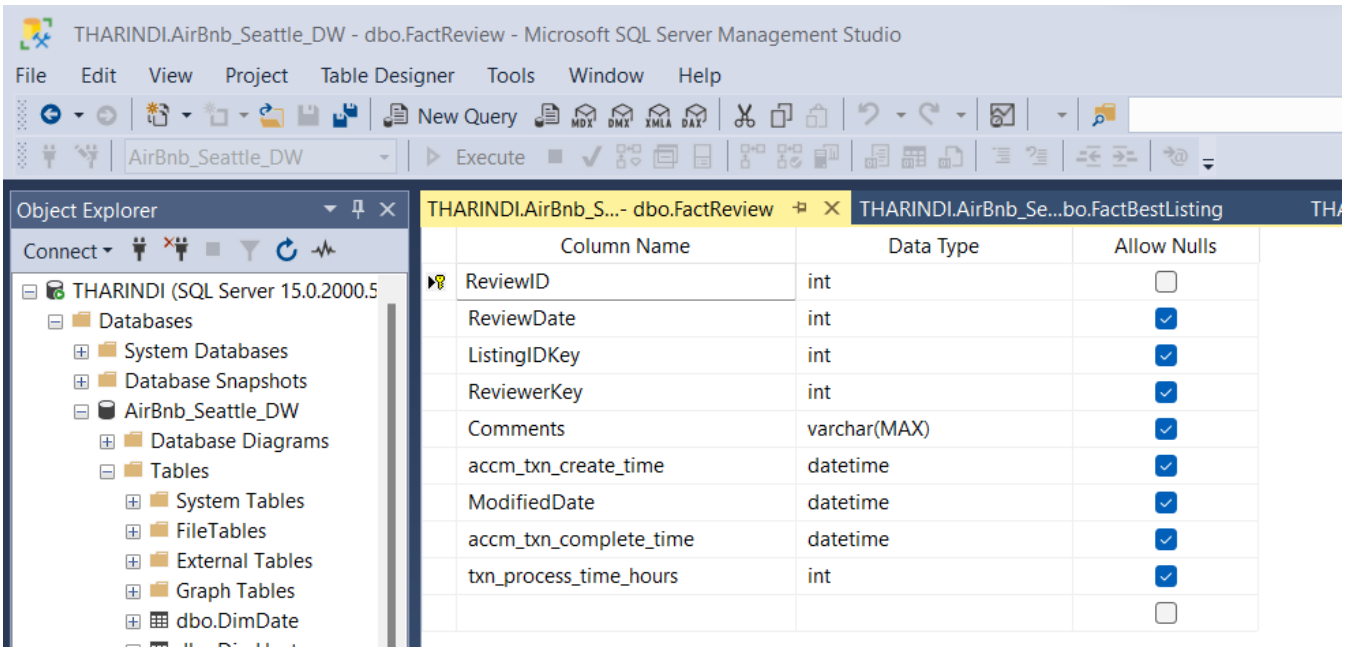
DimProperties SCD Table



DimReviewer Table



FactBestListing Fact Table



FactReview Fact table

5. 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.

5.1. 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

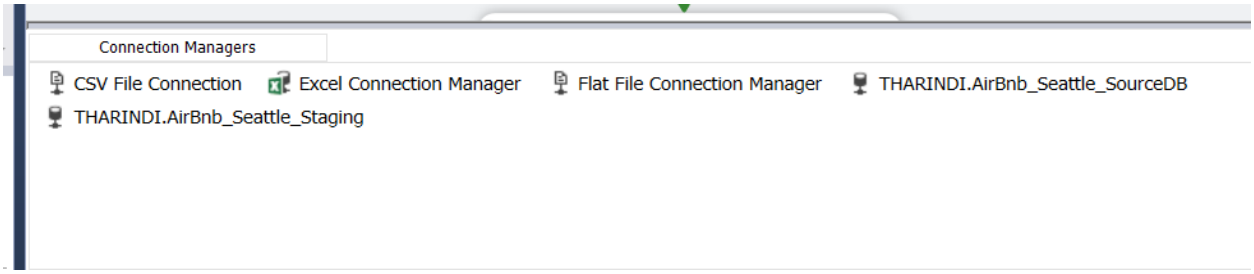
- AirBnb_Seattle_Load_Staging.dtsx – Includes set of ordered Tasks which performs EtL from sources to the staging layer database tables.
- AirBnb_Seattle_Load_DW.dtsx – contains all tasks which gets executed in a specific order which extract data from staging layer tables, do the necessary transformations, 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.

5.2.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.

5.2.1. Connections used

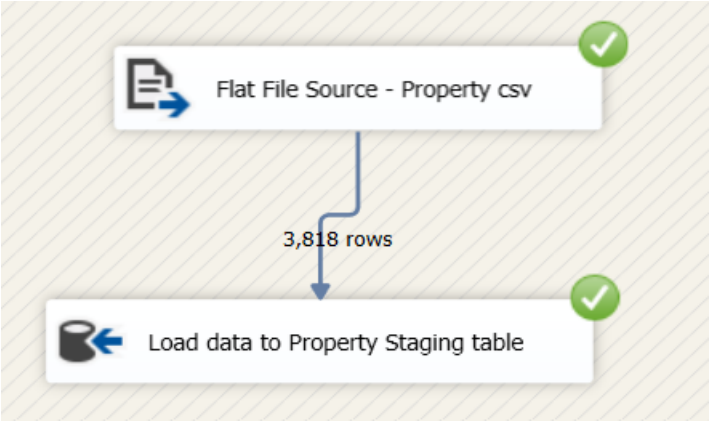


List of connection Managers created to connect with the source file and the staging database in the AirBnb_Seattle_Load_Staging.dtsx package. Different types of sources required different types of connection managers

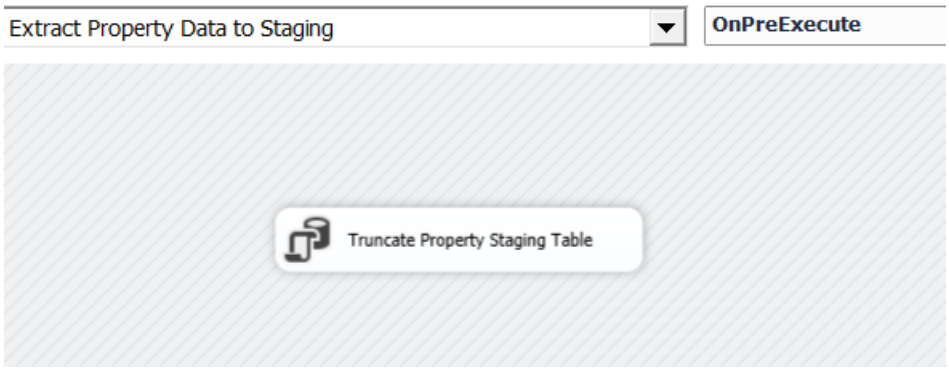
Connection Manager Name	Target
THARINDI.AirBnb_Seattle_SourceDB	MSSQL source database (AirBnbSeattle_Host, AirBnbSeattle_Listings and reviews tables)
CSV File Connection	Flat File connection used to load CSV Source (AirBnbSeattle_Property.csv)
Excel Connection Manager	Used to load AirBnbSeattle_Reviewers.xlsx 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.

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

5.2.2.1. Extract Property Data from a CSV file to Staging

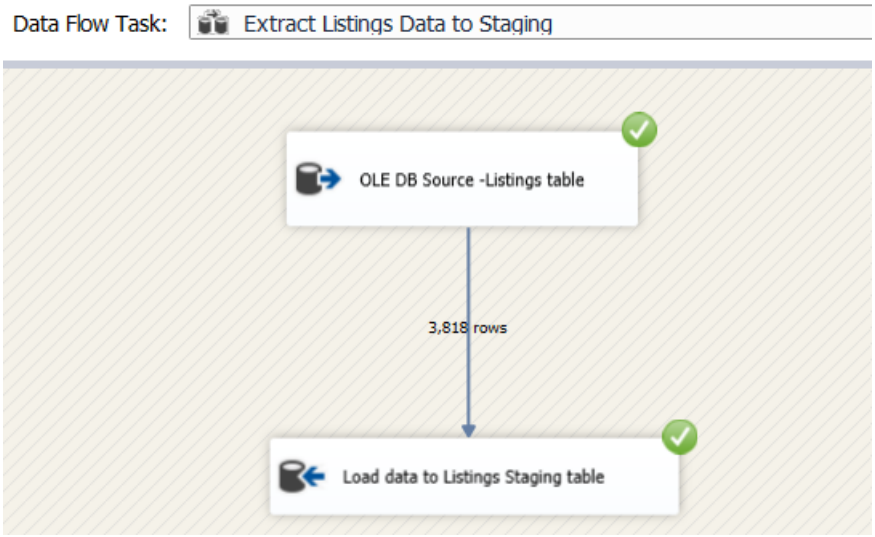


Flat File Source and OLE DB Destination components were used to extract Property data from CSV file to the StgProperty table of the MSSQL Staging database "AirBnb_Seattle_Staging". No transformations were done during the process.

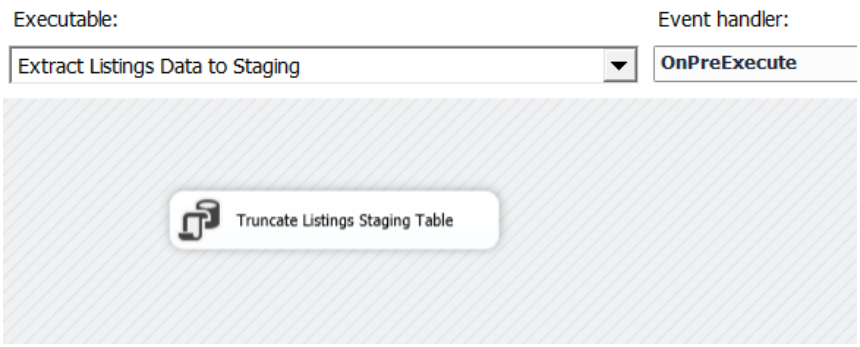


An "Execute SQL Task" was used inside the relevant event handler of the above-mentioned data flow task on Pre-execution to truncate the StgProperty table beforehand loading takes place to clear any previously staged data.

5.2.2.2. Extract Listings Data to Staging

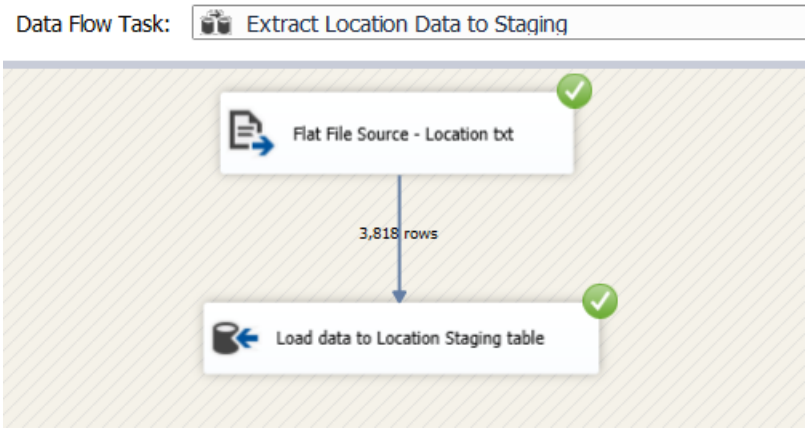


OLE DB Source and OLE DB Destination components were used to extract Listings data from AirBnbSeattle_Listings source database table to the StgListings table of the MSSQL Staging database "AirBnb_Seattle_Staging". No transformations were done during the process.

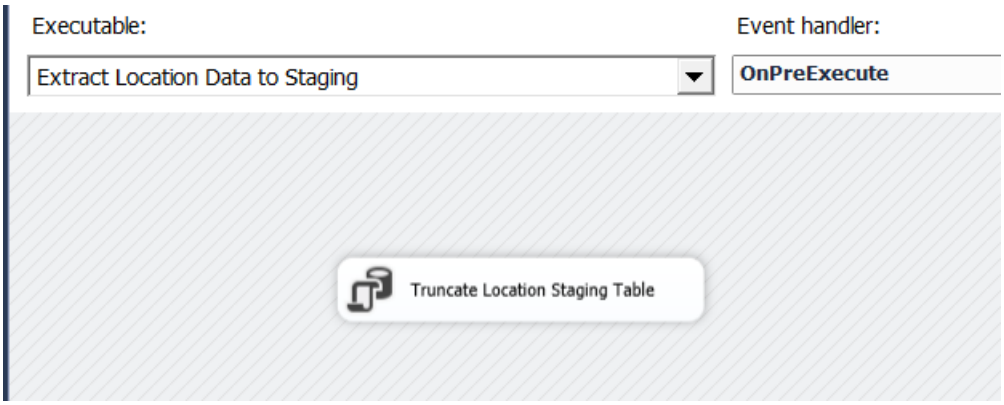


Similar in the previous case, an OnPreExecute event handler was used to create a Execute SQL Task to truncate StgListings table before loading the new data.

5.2.2.3. Extract Location Data to Staging

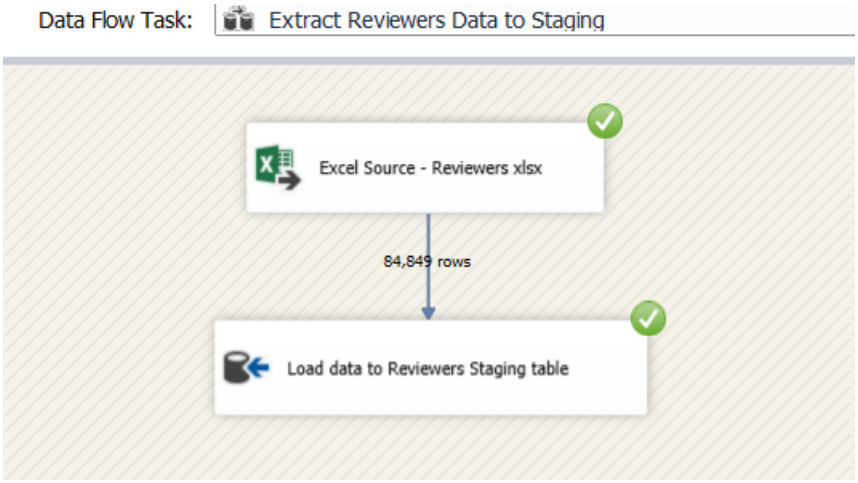


Data Flow of the "Extract Location Data to Staging". Data Flow Task uses a Flat file Source and then loads them to StgLocation table of the Staging database "AirBnb_Seattle_Staging" in MSSQL Server.

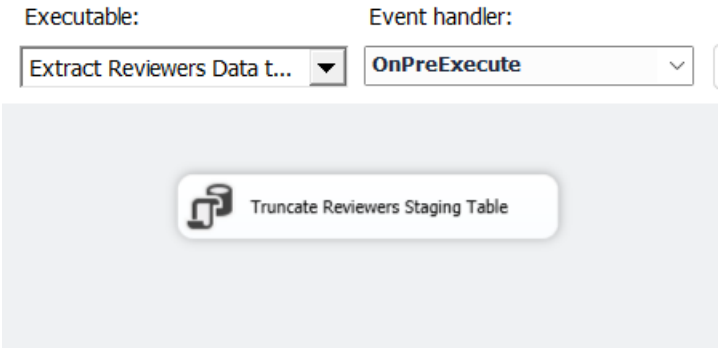


An event handler was used in the similar way shown in previous cases, to truncate the StgLocation table of the staging database prior to the EtL Task by specifying “OnPreExecute”.

5.2.2.4. Extract Reviewers Data to Staging

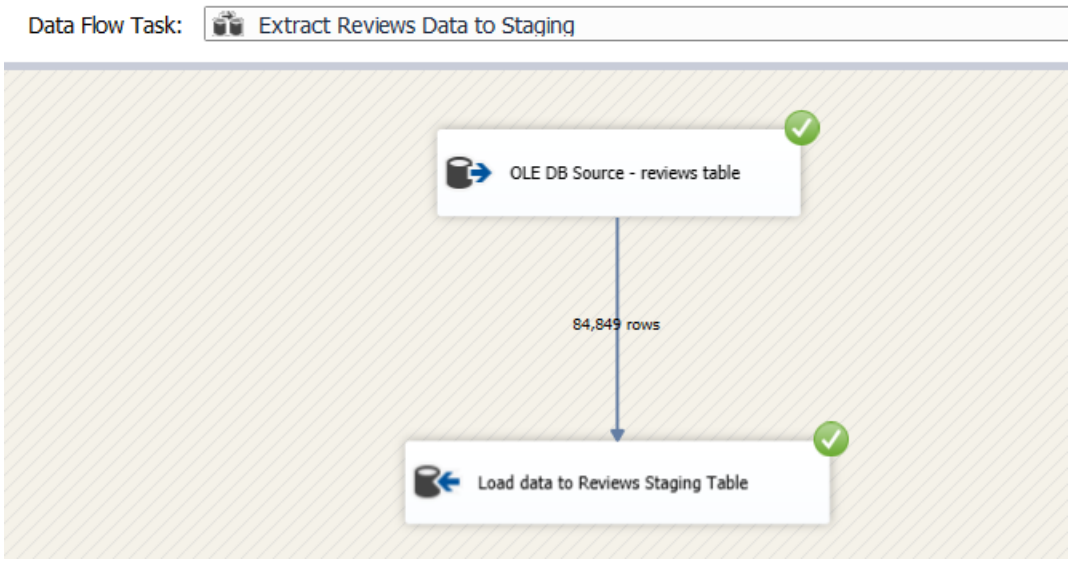


Reviewers’ data are getting extracted using an Excel source that connects to the excel file and using a OLE DB destination that is load into the staging database table, “StgReviewers”.

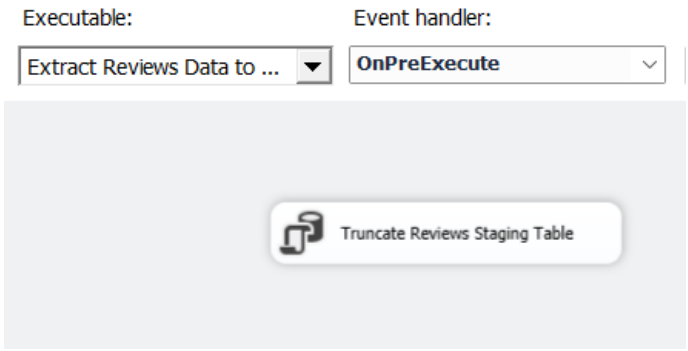


An event handler is in place and configured to run OnPreExecute to truncate the StgReviewers table before loading.

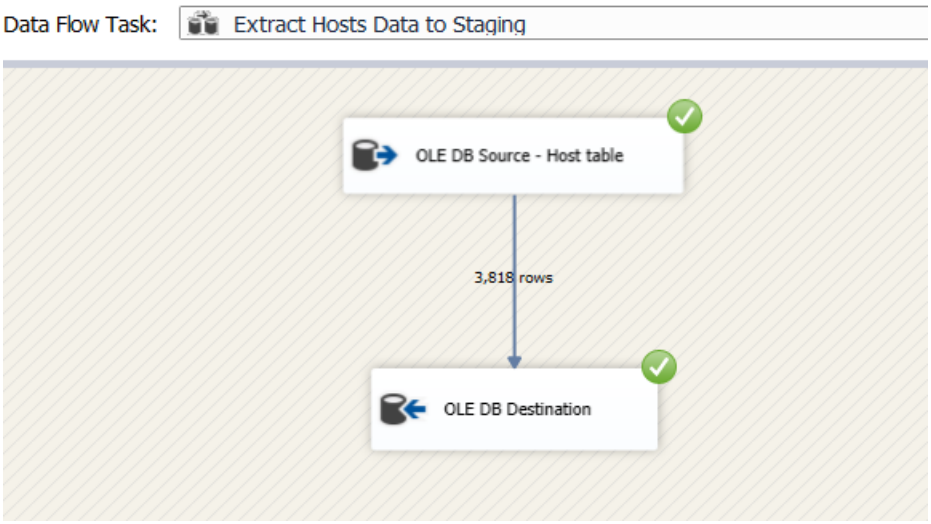
5.2.2.5. Extract Reviews Data to Staging



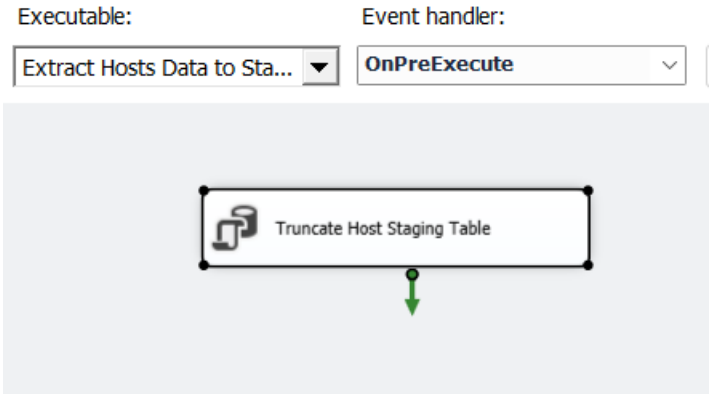
OLE DB Source and OLE DB Destination components were used to extract Reviews data from reviews source database table to the StgReviews table of the MSSQL Staging database "AirBnb_Seattle_Staging". An event handler is in place to truncate the StgReviews table before loading data to it. No transformations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



5.2.2.6. Extract Host Data to Staging



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.



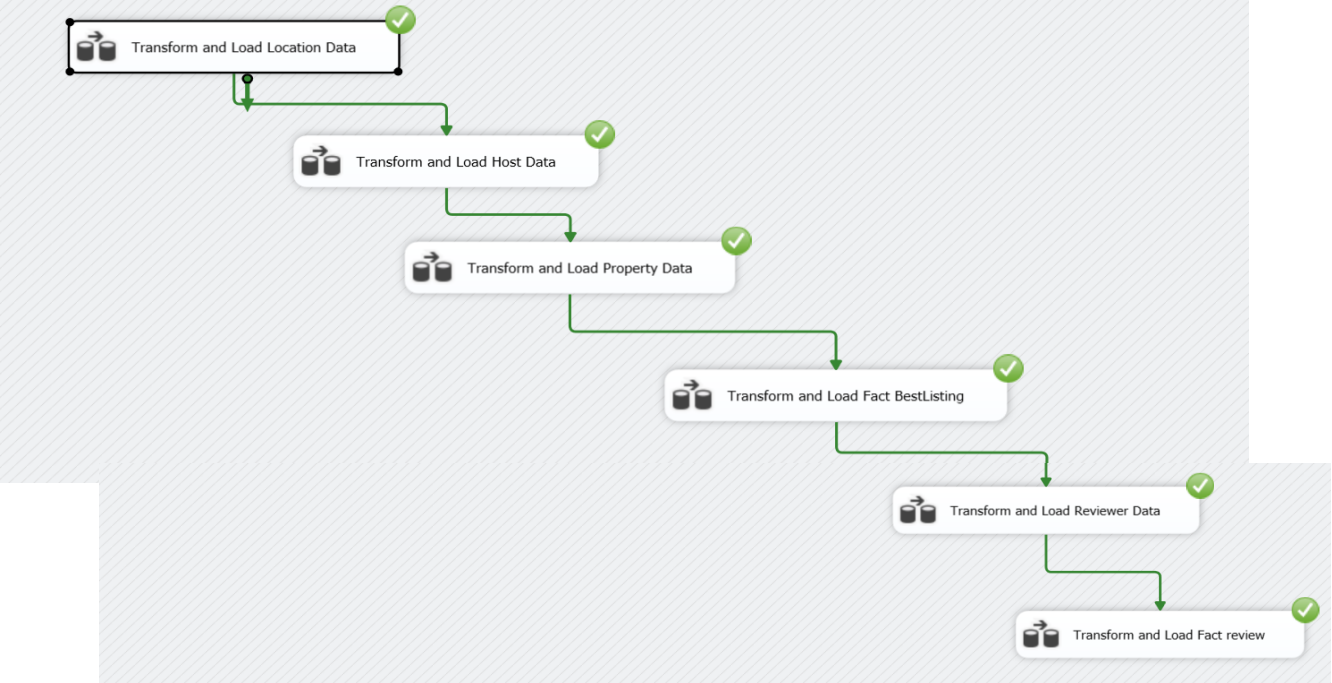
Similarly, an event handler is placed on pre-execution to clear the previously stored host data from the staging database table before the newly extracting data are stored

5.3. 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.

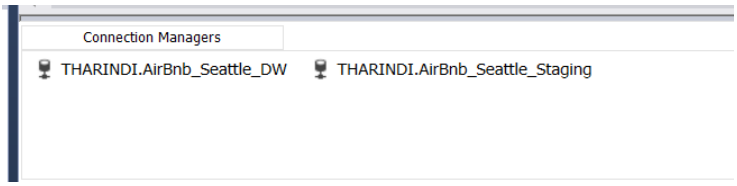
Although in the first EtL Task order was not that much important, the second set of ETL task execution order is the most important part. So, an execution order was planned and the ETL tasks were added accordingly. Business keys and surrogate keys were used to preserve the links between dim and fact tables.

A separate package was created to implement these named, AirBnb_Seattle_Load_DW.dtsx.



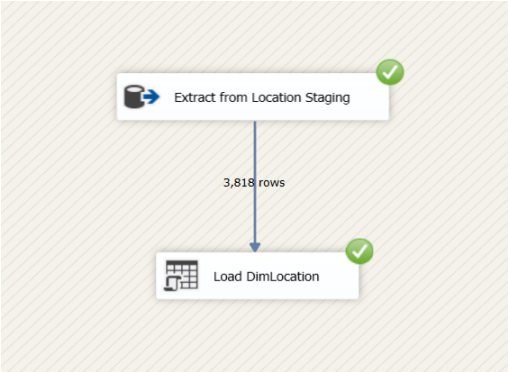
Data Flow tasks have not been assigned any event handlers to truncate any of the data warehouse tables because truncating data warehouse tables may cause issues of unexpected surrogate key changes. Data for all data warehouse tables are either updated or inserted only.

5.3.1. Connections used



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

5.3.2.1. Transform and Load Location Data to DimLocation



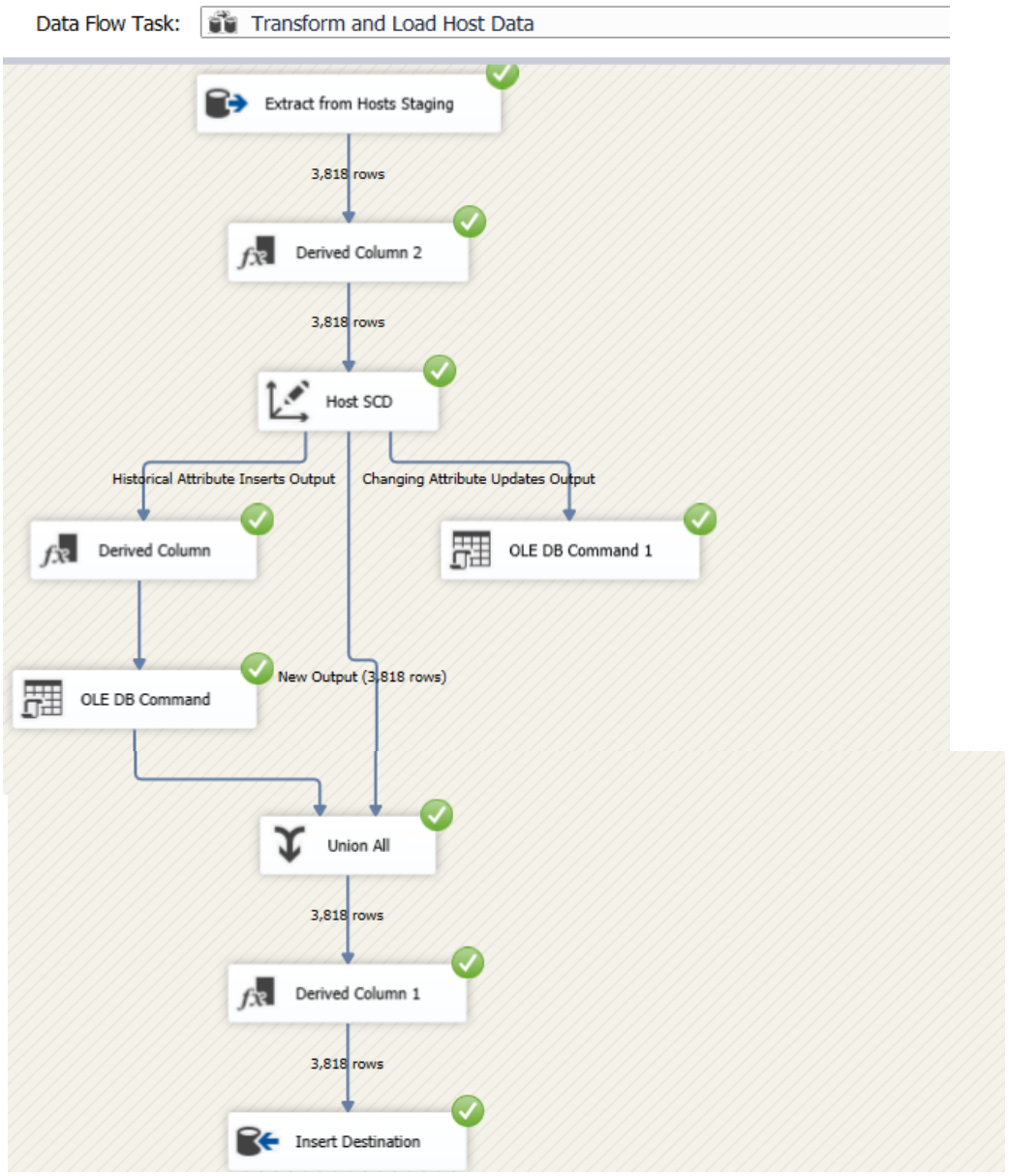
As shown in the above figure an OLE DB source was added to extract the data from staging layer table "StgLocation" and since there are no transformations to be done, the extracted data is sent to an OLE DB command component to pass the data to a stored procedure in the data warehouse to do the update 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;
```

5.3.2.2. 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

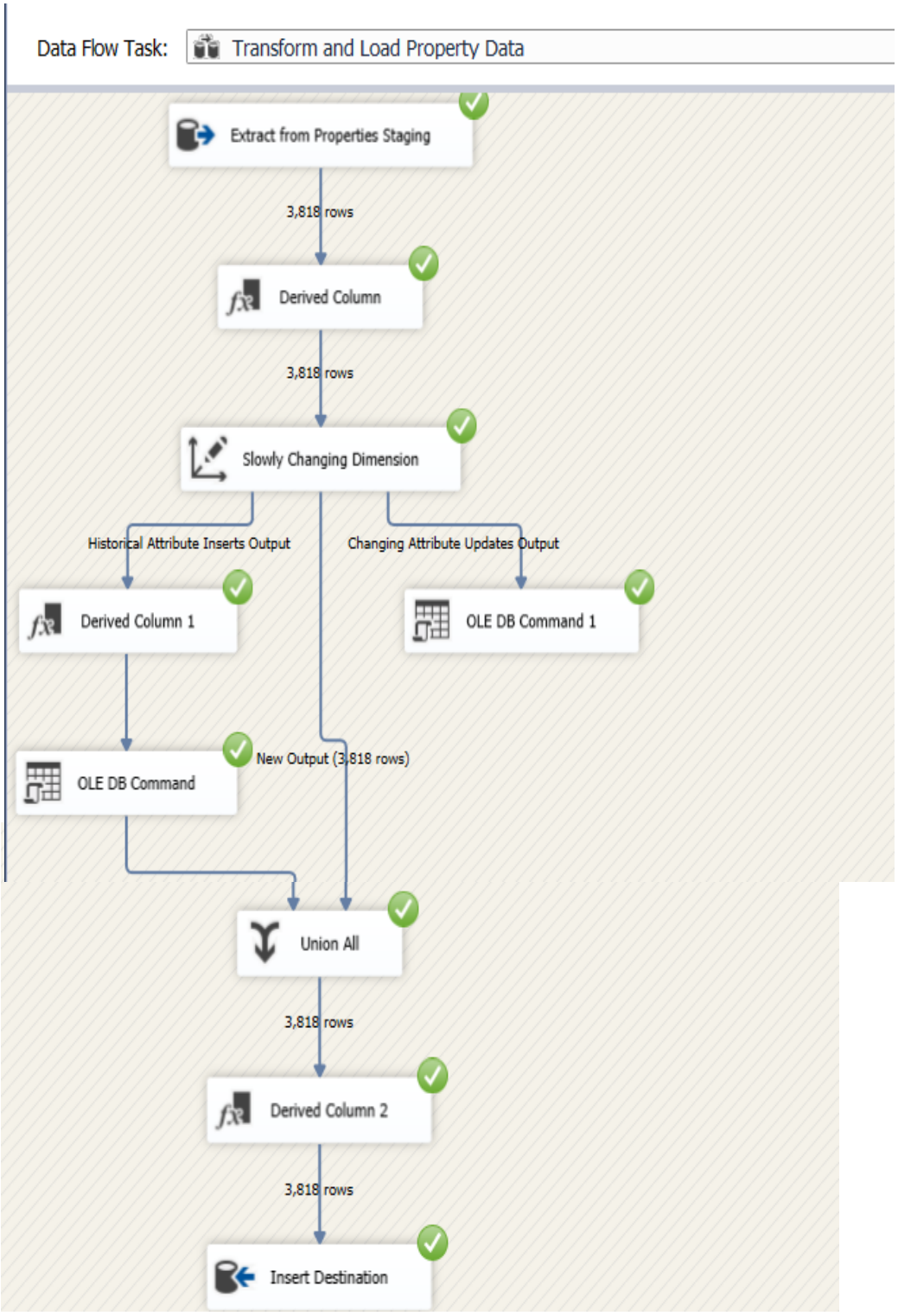
Type2(Historical) – HostName, HostISSuperhost

5.3.2.3.Transform and Load Property Data to DimProperties SCD

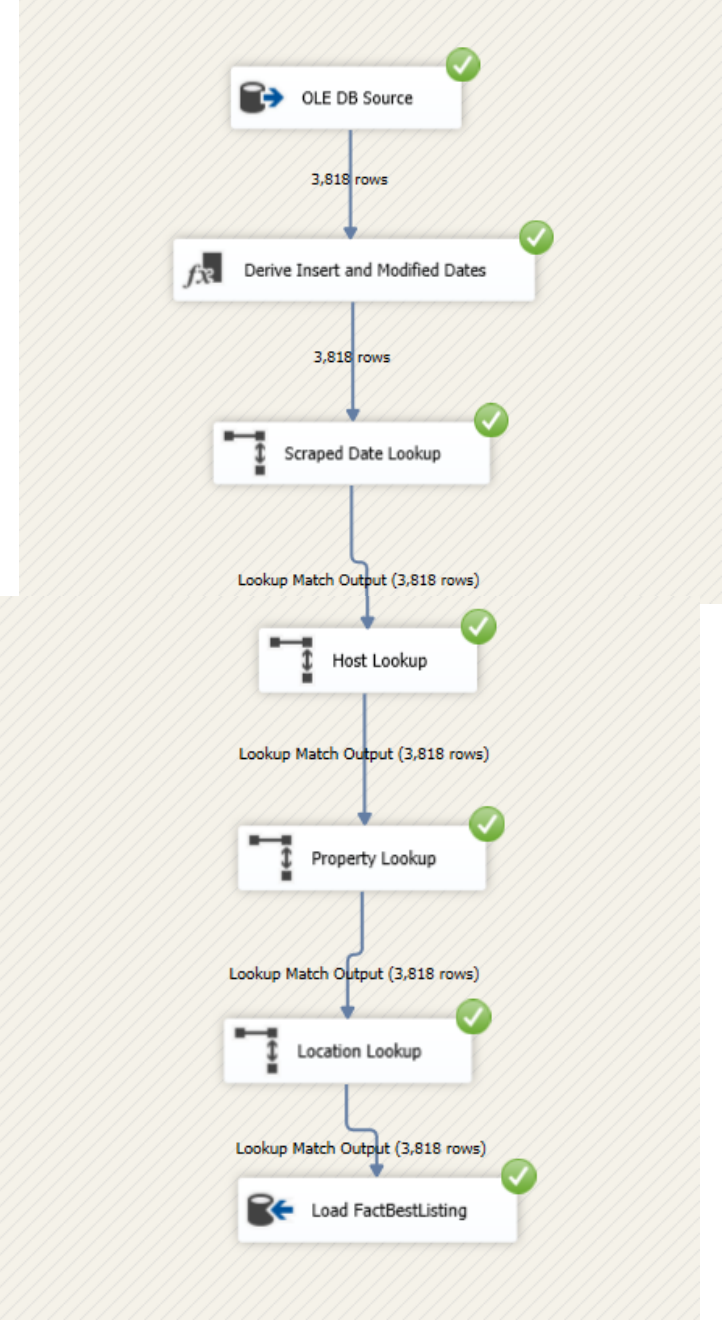
Here also "DimProperties" data warehouse table has considered as a Slowly Changing Dimension Table. First Property data is extracted from "StgProperty" table using OLE DB Source component. Then I added a Derived Column to add InsertDate and ModifiedDate columns.

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).


Data Flow Task:
Transform and Load Property Data
Extract from Properties Staging
3,818 rows
Derived Column
3,818 rows
Slowly Changing Dimension
Historical Attribute Inserts Output
Changing Attribute Updates Output
Derived Column 1
OLE DB Command 1
OLE DB Command
New Output (3,818 rows)
Union All
3,818 rows
Derived Column 2
3,818 rows
Insert Destination

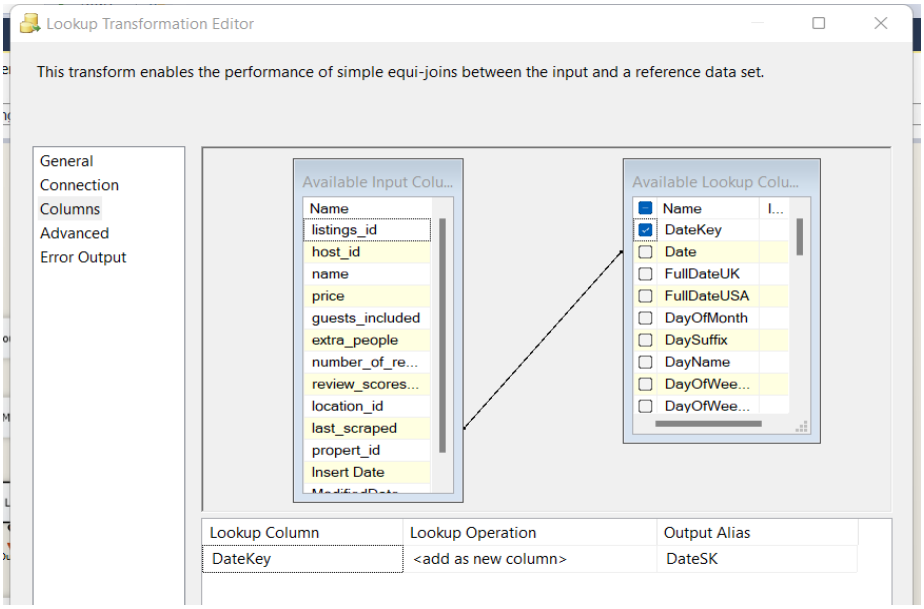
5.3.2.4.Transform and Load Fact BestListing



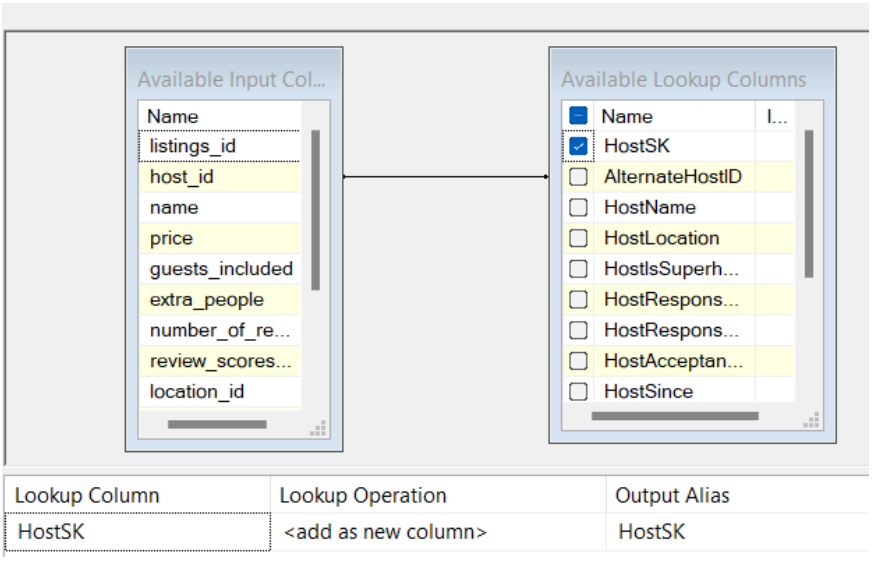
Above dataflow belongs to the ETL task of loading the BestListing fact table. As this fact table refers to the Date, Host, Property and Location dimension tables lookups were used to get the relevant surrogate keys to establish the references between tables.

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

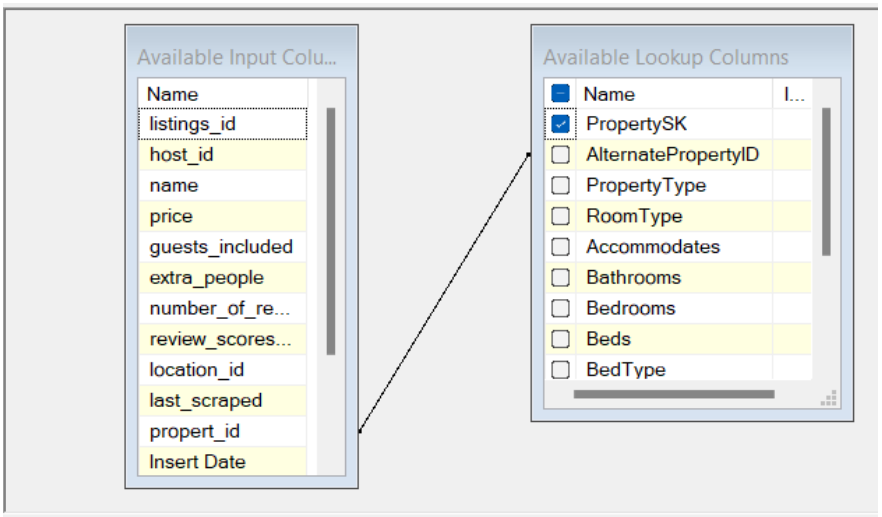
Then first lookup is done to retrieve the date key for the relevant scraped date.



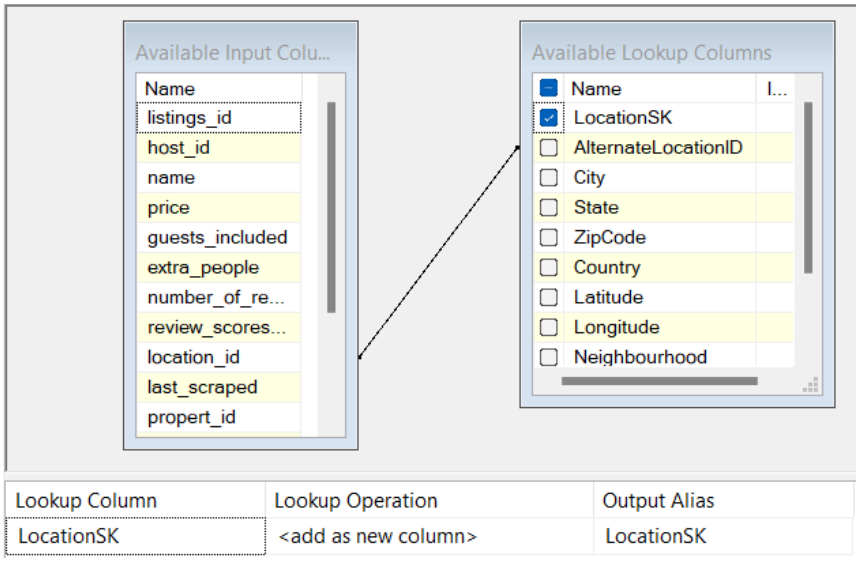
2nd lookup is done to retrieve the HostSK from the DimHost.



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

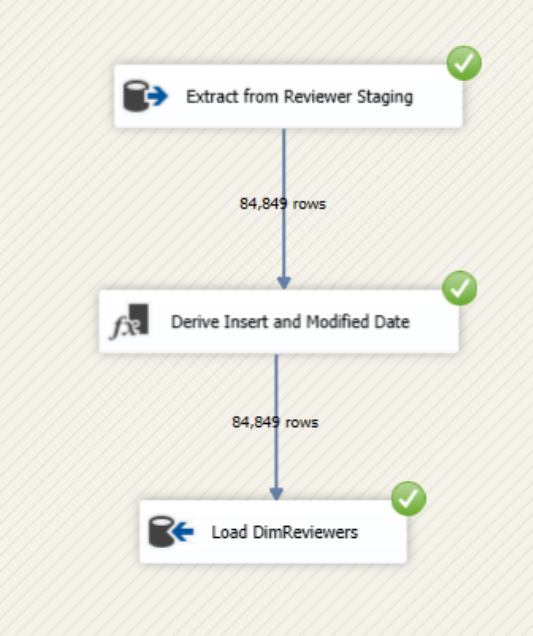


4th lookup is done to retrieve the LocationSK from the DimLocation.



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

5.3.2.5. Transform and Load DimReviewer



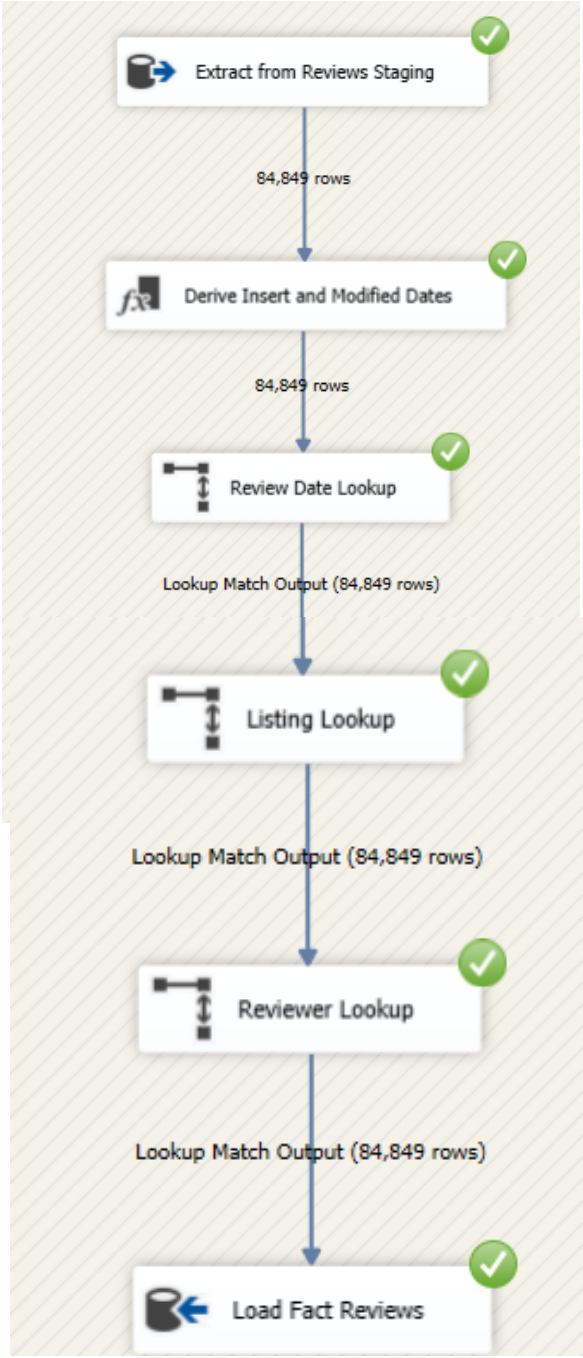
As DimReviewers table has no references to above loaded tables I considered to load DimReviewers table after loading FactBestListing table also. Then as in previous ones here also added a Derived Column to add InsertDate and ModifiedDate columns. Then finally DimReviewers table load to the data warehouse.

5.3.2.6. Transform and Load FactReview table

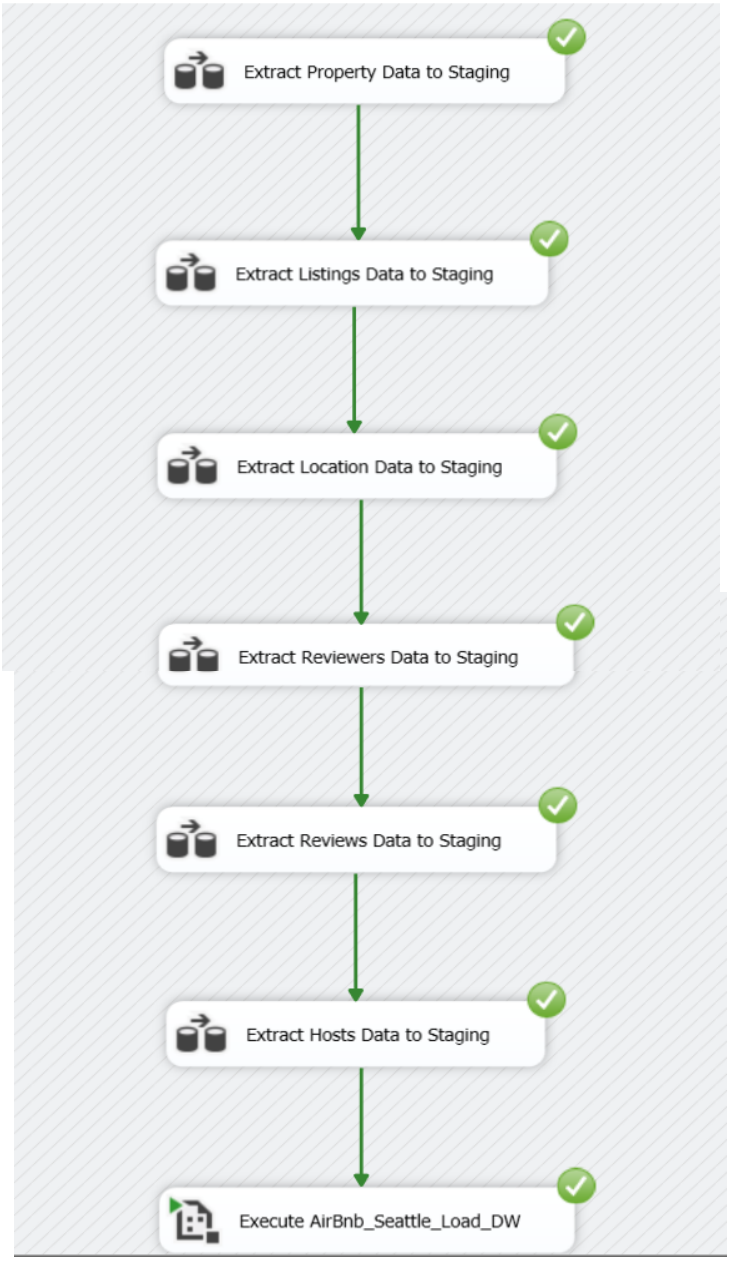
Like in the first data flow process, here also data extracted from the staging table "StgReviews" using the OLE DB Source component and then derived column component added to insert the current timestamp from the GETDATE() function to add InsertDate and ModifiedDate columns.

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.

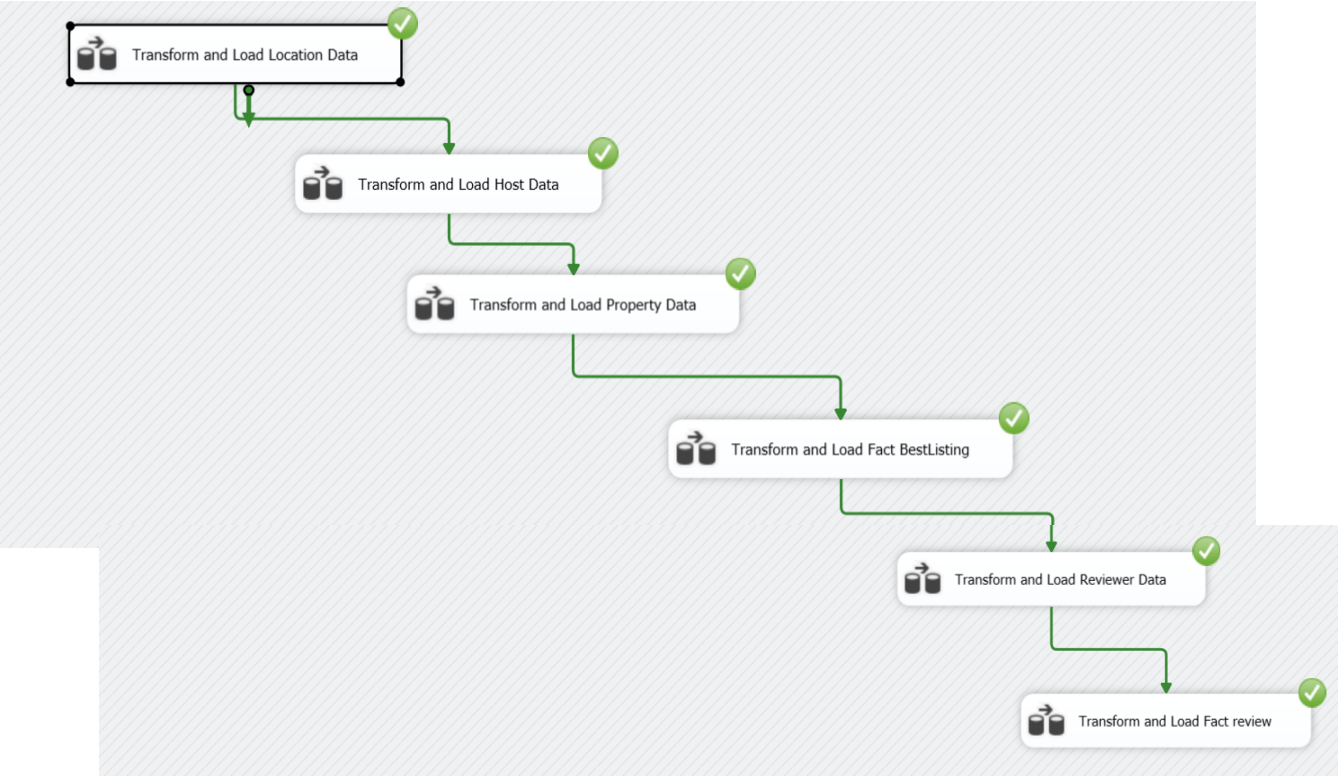
At last output is sent to an OLE DB Destination component which refers to the FactReviews table in the data warehouse


Extract from Reviews Staging
84,849 rows
Derive Insert and Modified Dates
84,849 rows
Review Date Lookup
Lookup Match Output (84,849 rows)
Listing Lookup
Lookup Match Output (84,849 rows)
Reviewer Lookup
Lookup Match Output (84,849 rows)
Load Fact Reviews

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.



6. ETL development – Accumulating Fact Tables

6.1.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.

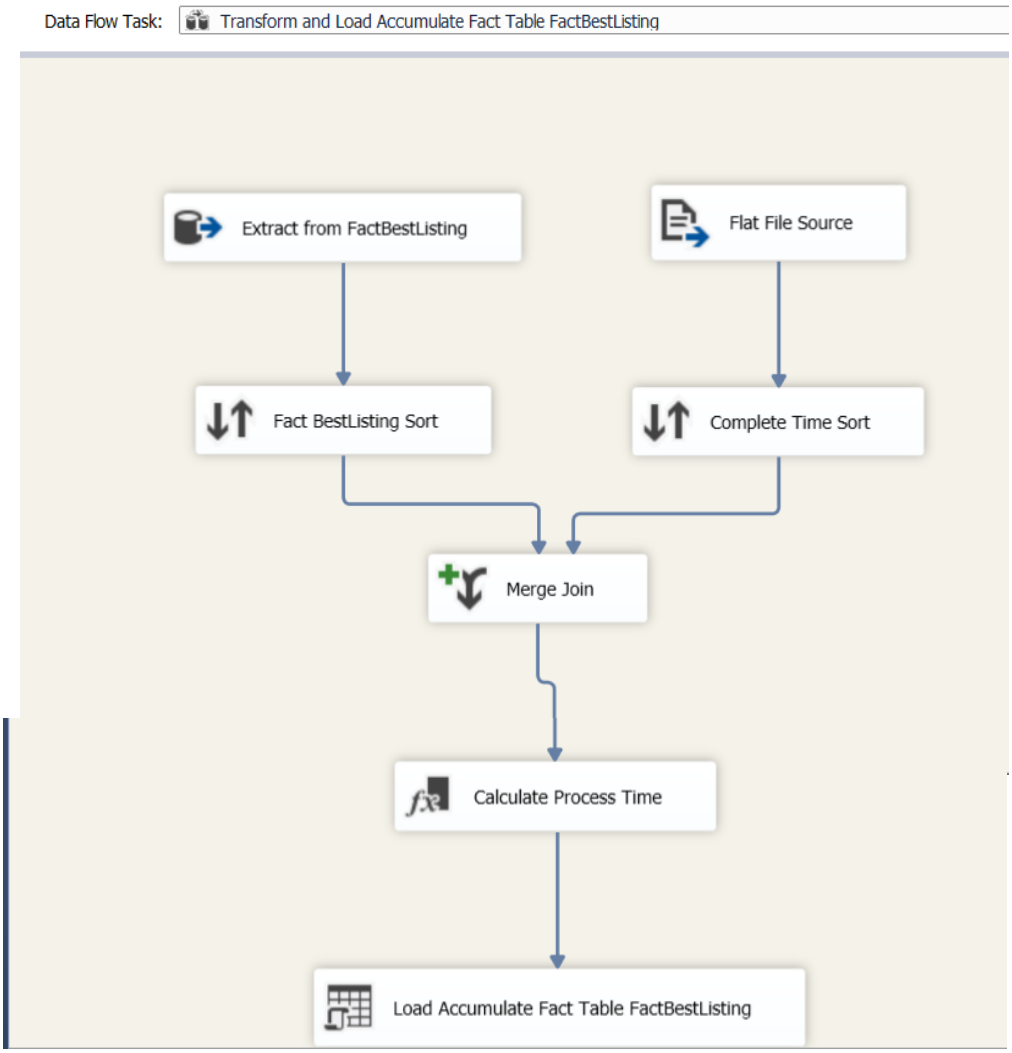
Then prepared a separate csv file named as complete_time_source by adding the 2 columns named as fact_table_natural_key and accm_txn_complete_time. FactBestListing table's natural key which is the ListingID values are taken as the fact_table_natural_key column values.

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")

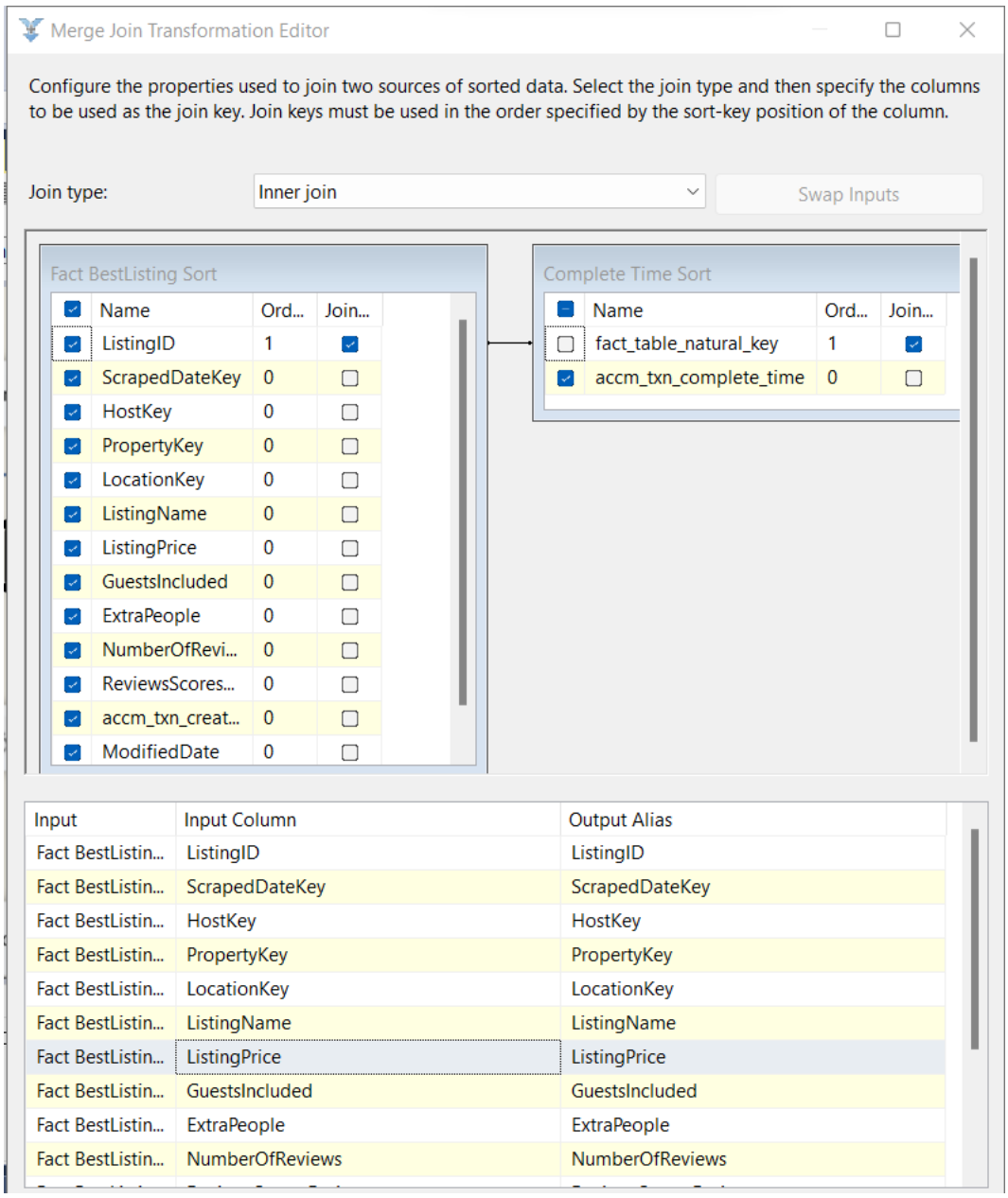
Then created a package in SSDT named as "AirBnb_Seattle_AccumulateFactTables_Load_DW.dtsx" to update the FactBestListing table values.

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.

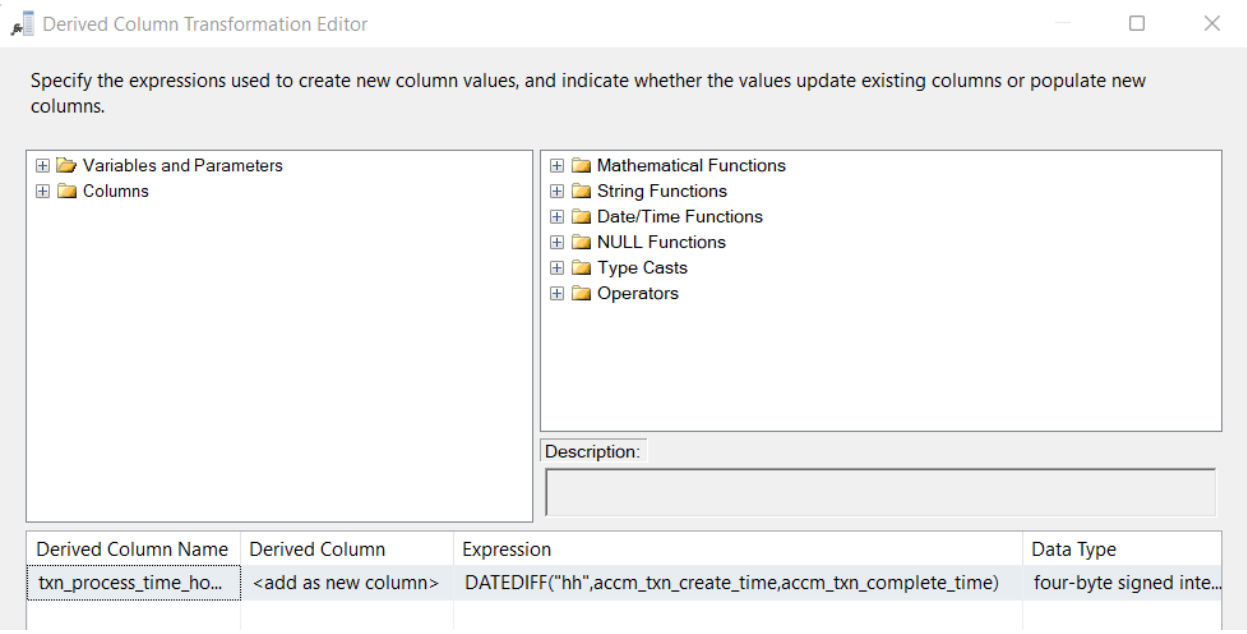


Using OLE DB Source component and Flat file source component extracted data from FactBestListing and complete_time_process csv file respectively.

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.

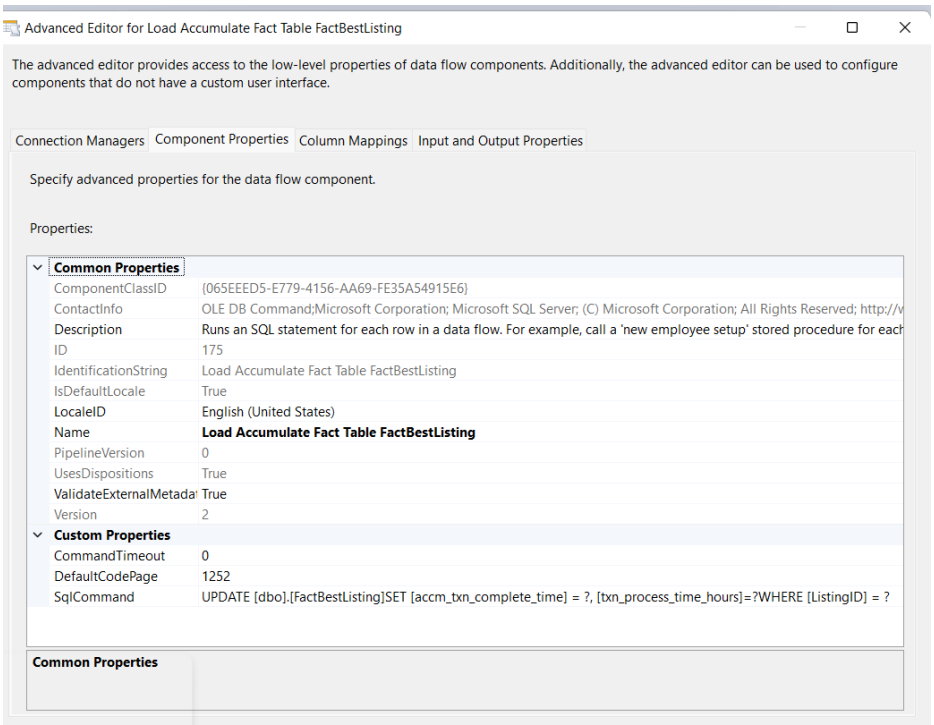

Merge Join Transformation Editor
Configure the properties used to join two sources of sorted data. Select the join type and then specify the columns to be used as the join key. Join keys must be used in the order specified by the sort-key position of the column.
Join type:
Inner join
Swap Inputs
Fact BestListing Sort
Name
Ord...
Join...
ListingID 1
ScrapedDateKey 0
HostKey 0
PropertyKey 0
LocationKey 0
ListingName 0
ListingPrice 0
GuestsIncluded 0
ExtraPeople 0
NumberOfRevi... 0
ReviewsScores... 0
accm_txn_creat... 0
ModifiedDate 0
Complete Time Sort
Name
Ord...
Join...
fact_table_natural_key 1
accm_txn_complete_time 0
Input
Input Column
Output Alias
Fact BestListin... ListingID ListingID
Fact BestListin... ScrapedDateKey ScrapedDateKey
Fact BestListin... HostKey HostKey
Fact BestListin... PropertyKey PropertyKey
Fact BestListin... LocationKey LocationKey
Fact BestListin... ListingName ListingName
Fact BestListin... ListingPrice ListingPrice
Fact BestListin... GuestsIncluded GuestsIncluded
Fact BestListin... ExtraPeople ExtraPeople
Fact BestListin... NumberOfReviews NumberOfReviews

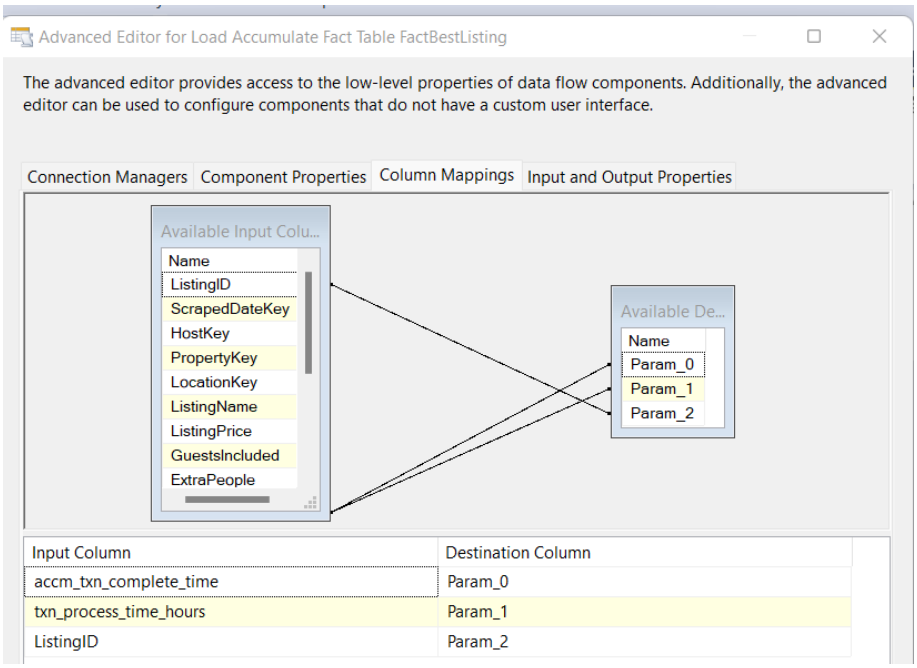
Next I added a derived column to calculate Process time.



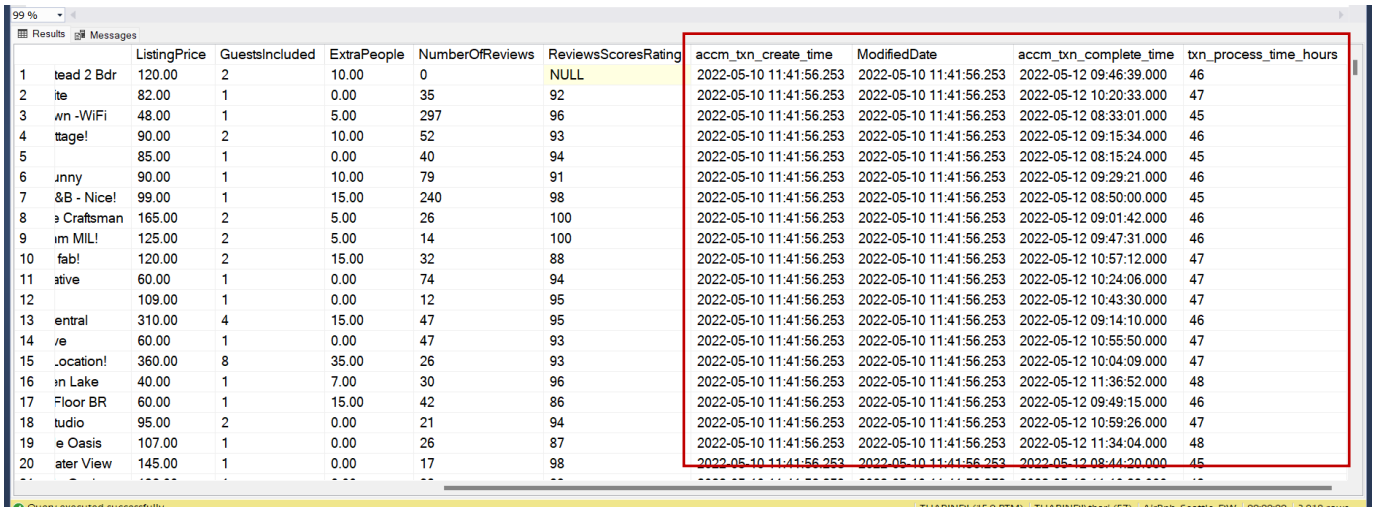
To get the difference between the complete time and the create time I used DATEDIFF () function which has given by the Derived Column Component by default.

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





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

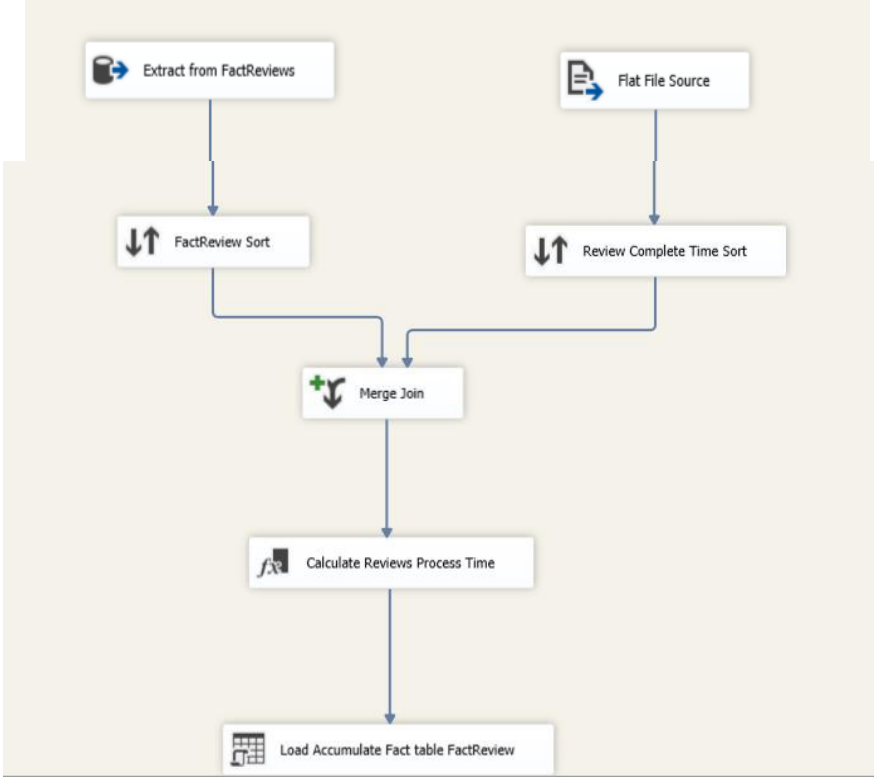


		ListingPrice	GuestsIncluded	ExtraPeople	NumberOfReviews	ReviewsScoresRating	accm_txn_create_time	ModifiedDate	accm_txn_complete_time	txn_process_time_hours
1	tead 2 Bdr	120.00	2	10.00	0	NULL	2022-05-10 11:41:56.253	2022-05-10 11:41:56.253	2022-05-12 09:46:39.000	46
2	ite	82.00	1	0.00	35	92	2022-05-10 11:41:56.253	2022-05-10 11:41:56.253	2022-05-12 10:20:33.000	47
3	wn -WiFi	48.00	1	5.00	297	96	2022-05-10 11:41:56.253	2022-05-10 11:41:56.253	2022-05-12 08:33:01.000	45
4	ttage!	90.00	2	10.00	52	93	2022-05-10 11:41:56.253	2022-05-10 11:41:56.253	2022-05-12 09:15:34.000	46
5		85.00	1	0.00	40	94	2022-05-10 11:41:56.253	2022-05-10 11:41:56.253	2022-05-12 08:15:24.000	45
6	unny	90.00	1	10.00	79	91	2022-05-10 11:41:56.253	2022-05-10 11:41:56.253	2022-05-12 09:29:21.000	46
7	&B - Nice!	99.00	1	15.00	240	98	2022-05-10 11:41:56.253	2022-05-10 11:41:56.253	2022-05-12 08:50:00.000	45
8	Craftsman	165.00	2	5.00	26	100	2022-05-10 11:41:56.253	2022-05-10 11:41:56.253	2022-05-12 09:01:42.000	46
9	m MIL!	125.00	2	5.00	14	100	2022-05-10 11:41:56.253	2022-05-10 11:41:56.253	2022-05-12 09:47:31.000	46
10	fab!	120.00	2	15.00	32	88	2022-05-10 11:41:56.253	2022-05-10 11:41:56.253	2022-05-12 10:57:12.000	47
11	ative	60.00	1	0.00	74	94	2022-05-10 11:41:56.253	2022-05-10 11:41:56.253	2022-05-12 10:24:06.000	47
12		109.00	1	0.00	12	95	2022-05-10 11:41:56.253	2022-05-10 11:41:56.253	2022-05-12 10:43:30.000	47
13	entral	310.00	4	15.00	47	95	2022-05-10 11:41:56.253	2022-05-10 11:41:56.253	2022-05-12 09:14:10.000	46
14	/e	60.00	1	0.00	47	93	2022-05-10 11:41:56.253	2022-05-10 11:41:56.253	2022-05-12 10:55:50.000	47
15	.ocation!	360.00	8	35.00	26	93	2022-05-10 11:41:56.253	2022-05-10 11:41:56.253	2022-05-12 10:04:09.000	47
16	n Lake	40.00	1	7.00	30	96	2022-05-10 11:41:56.253	2022-05-10 11:41:56.253	2022-05-12 11:36:52.000	48
17	Floor BR	60.00	1	15.00	42	86	2022-05-10 11:41:56.253	2022-05-10 11:41:56.253	2022-05-12 09:49:15.000	46
18	tudio	95.00	2	0.00	21	94	2022-05-10 11:41:56.253	2022-05-10 11:41:56.253	2022-05-12 10:59:26.000	47
19	e Oasis	107.00	1	0.00	26	87	2022-05-10 11:41:56.253	2022-05-10 11:41:56.253	2022-05-12 11:34:04.000	48
20	ater View	145.00	1	0.00	17	98	2022-05-10 11:41:56.253	2022-05-10 11:41:56.253	2022-05-12 08:44:20.000	45

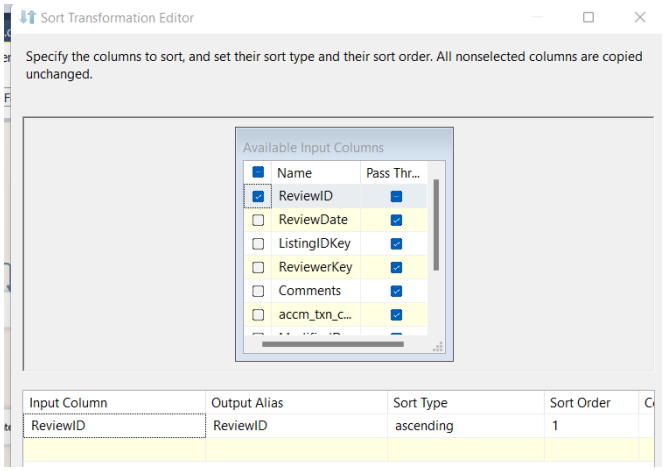
6.2.Loading updated Accumulated Fact Table FactReview

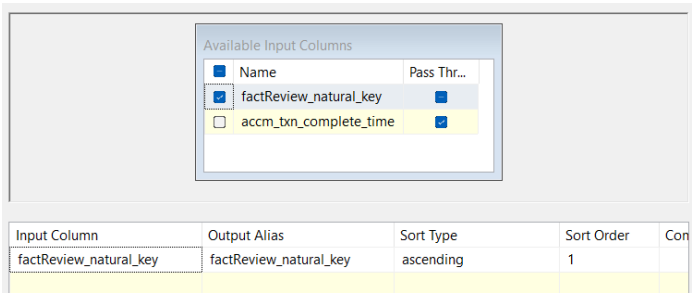
As I have 2 fact tables I done the same procedure to the FactReview fact table also. In there also I created a separate csv file name and named it as complete_time_reviews.csv.

Following are the screenshots taken from the Visual Studio.

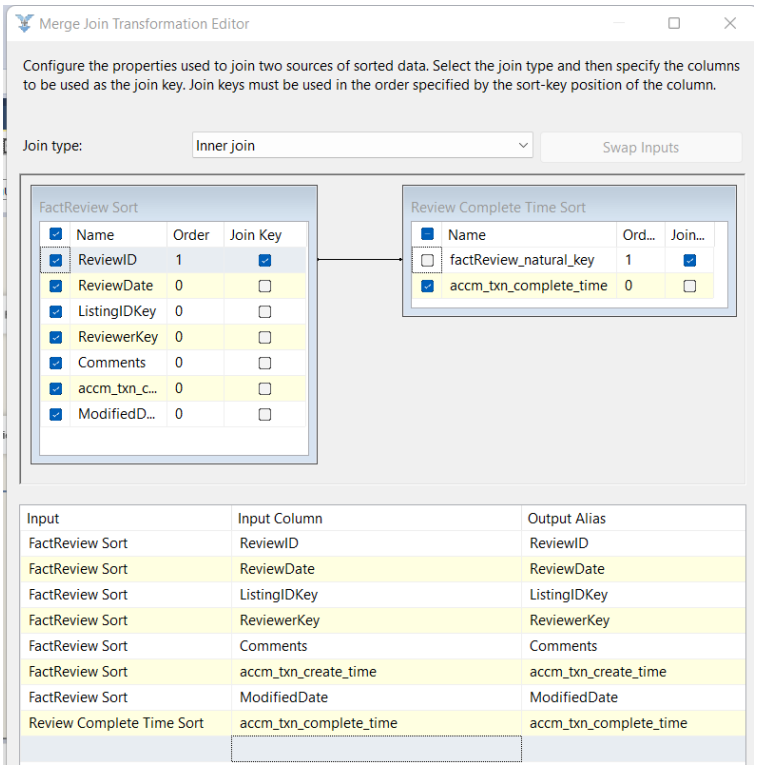


Sort components

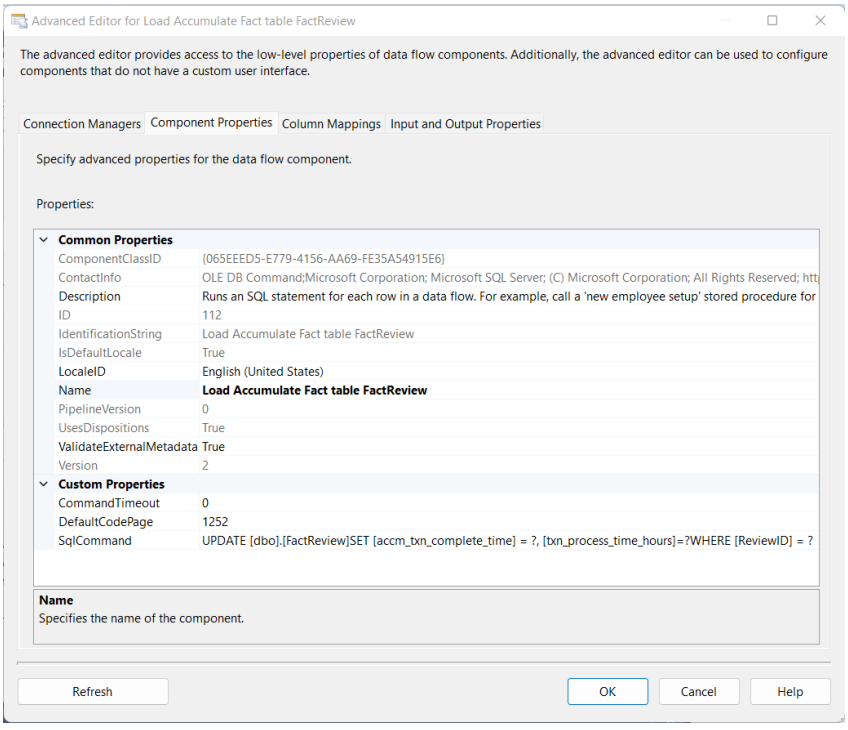


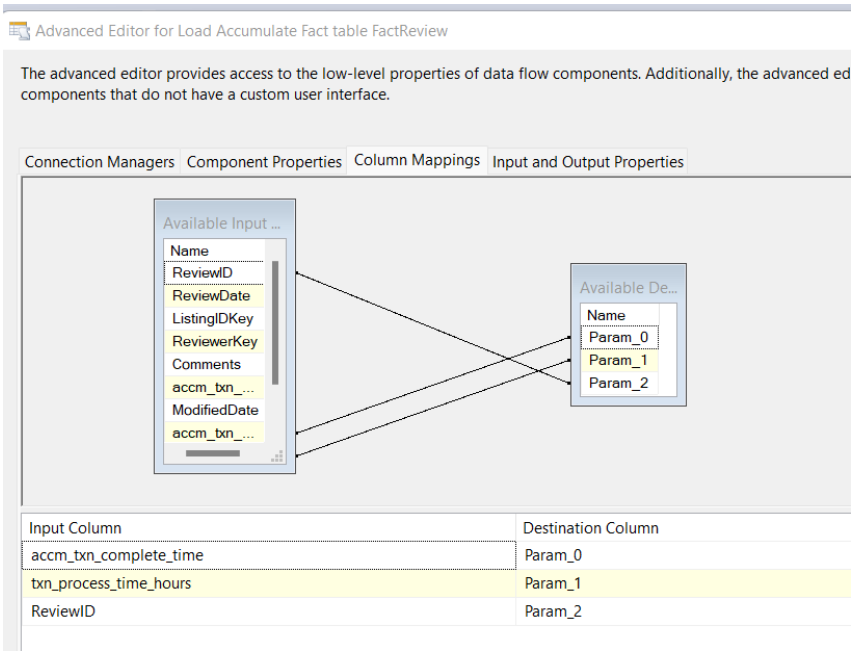


Merge Join



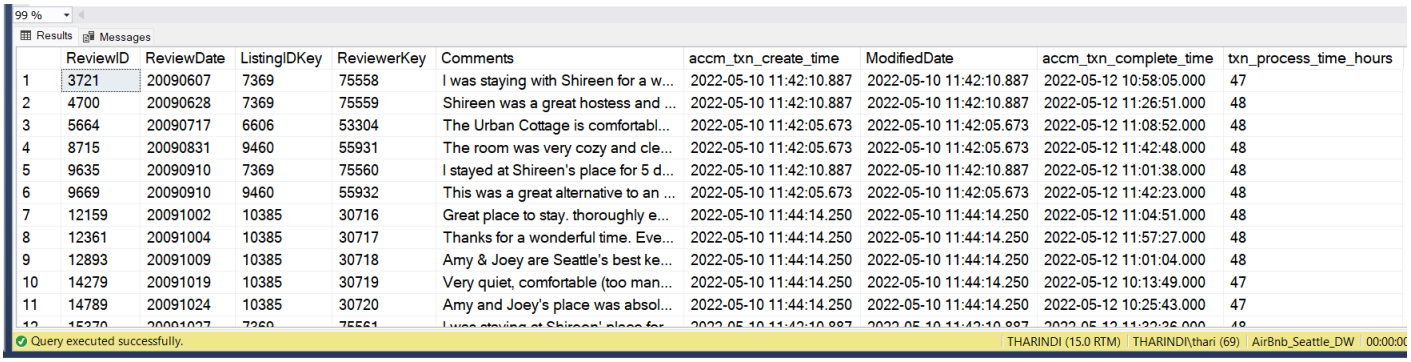
Loading updated Accumulated Fact Table FactReview





(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.



99 %

Results | Messages

	ReviewID	ReviewDate	ListingIDKey	ReviewerKey	Comments	accm_txn_create_time	ModifiedDate	accm_txn_complete_time	txn_process_time_hours
1	3721	20090607	7369	75558	I was staying with Shireen for a w...	2022-05-10 11:42:10.887	2022-05-10 11:42:10.887	2022-05-12 10:58:05.000	47
2	4700	20090628	7369	75559	Shireen was a great hostess and ...	2022-05-10 11:42:10.887	2022-05-10 11:42:10.887	2022-05-12 11:26:51.000	48
3	5664	20090717	6606	53304	The Urban Cottage is comfortabl...	2022-05-10 11:42:05.673	2022-05-10 11:42:05.673	2022-05-12 11:08:52.000	48
4	8715	20090831	9460	55931	The room was very cozy and cle...	2022-05-10 11:42:05.673	2022-05-10 11:42:05.673	2022-05-12 11:42:48.000	48
5	9635	20090910	7369	75560	I stayed at Shireen's place for 5 d...	2022-05-10 11:42:10.887	2022-05-10 11:42:10.887	2022-05-12 11:01:38.000	48
6	9669	20090910	9460	55932	This was a great alternative to an ...	2022-05-10 11:42:05.673	2022-05-10 11:42:05.673	2022-05-12 11:42:23.000	48
7	12159	20091002	10385	30716	Great place to stay. thoroughly e...	2022-05-10 11:44:14.250	2022-05-10 11:44:14.250	2022-05-12 11:04:51.000	48
8	12361	20091004	10385	30717	Thanks for a wonderful time. Eve...	2022-05-10 11:44:14.250	2022-05-10 11:44:14.250	2022-05-12 11:57:27.000	48
9	12893	20091009	10385	30718	Amy & Joey are Seattle's best ke...	2022-05-10 11:44:14.250	2022-05-10 11:44:14.250	2022-05-12 11:01:04.000	48
10	14279	20091019	10385	30719	Very quiet, comfortable (too man...	2022-05-10 11:44:14.250	2022-05-10 11:44:14.250	2022-05-12 10:13:49.000	47
11	14789	20091024	10385	30720	Amy and Joey's place was absol...	2022-05-10 11:44:14.250	2022-05-10 11:44:14.250	2022-05-12 10:25:43.000	47

Query executed successfully. | THARINDI (15.0 RTM) | THARINDI\thari (69) | AirBnb_Seattle_DW | 00:00:0