Data Warehousing & Business Intelligence (IT)

3rd Year, 1st Semester

Assignment 1

Submitted to

Sri Lanka Institute of Information Technology

IT18125726

Kumarage L.V.A

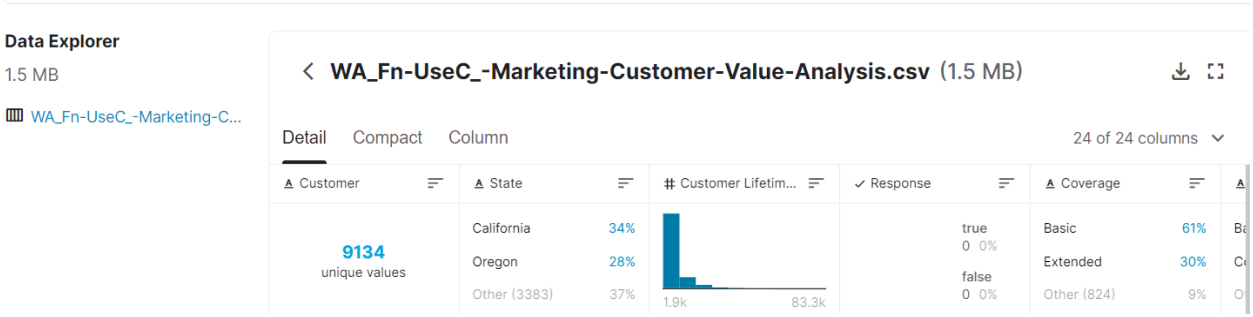
Weekday Batch

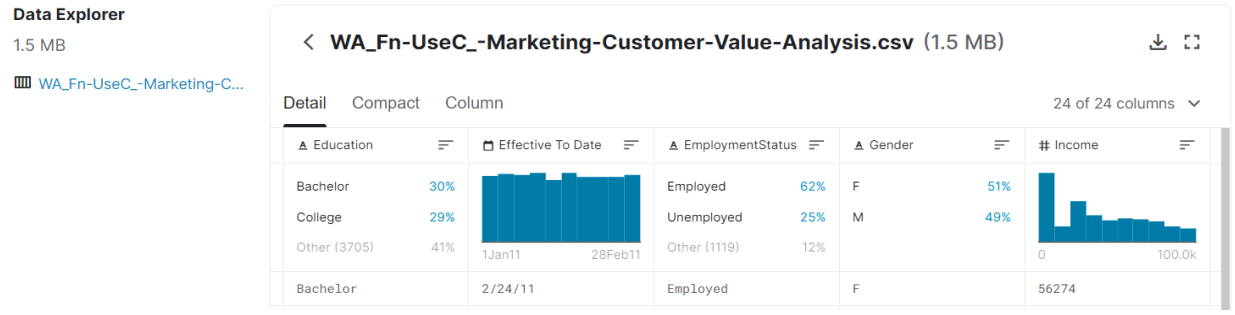
Step 1: Data set selection

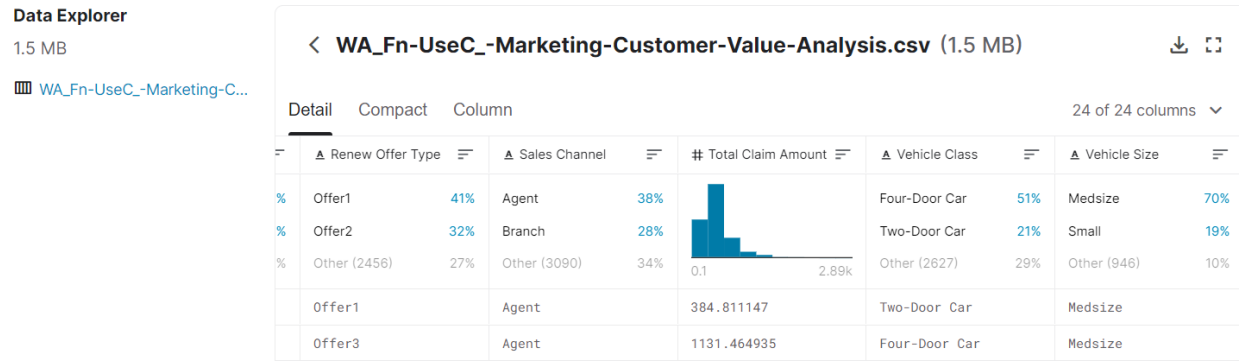
I have selected IBM Watson Marketing Customer Value Data as data set. It consists of one CSV fie with sufficient data with 24 columns. Furthermore, I have partitioned the main large CSV file into small sub-CSV files. The sub-CSV files consist of new IDs. And, I have manually modified some data records according to the requirements.

The data set was initiated with sufficient data, according to the assignment criteria. It has more the 9000 unique values and it is enriched with transactional data and data hierarchies.

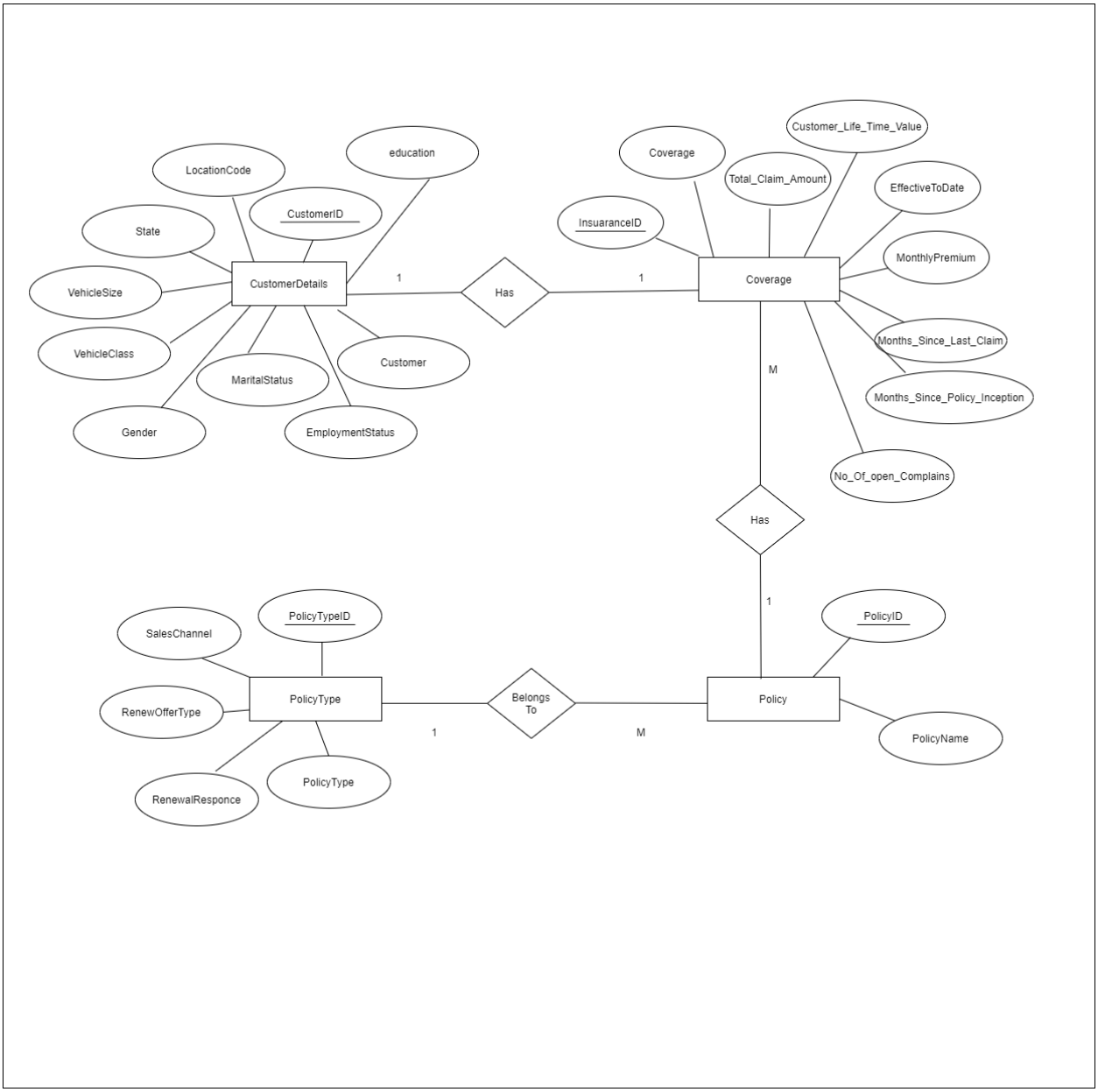
Data Set - https://www.kaggle.com/pankajjsh06/ibm-watson-marketing-customer-value-data?select=WA_Fn-UseC_-Marketing-Customer-Value-Analysis.csv







ER Diagram -


education
LocationCode
CustomerID
State
VehicleSize
VehicleClass
MaritalStatus
Gender
EmploymentStatus
Customer
CustomerDetails
1
Has
1
InsuaranceID
Coverage
Total_Claim_Amount
Customer_Life_Time_Value
EffectiveToDate
MonthlyPremium
Months_Since_Last_Claim
Months_Since_Policy_Inception
No_Of_open_Complains
Coverage
M
Has
1
PolicyID
Policy
PolicyName
Belongs To
M
1
PolicyTypeID
SalesChannel
RenewOfferType
RenewalResponce
PolicyType
PolicyType

Step 2: Preparation of Data Sources

First, main data set was separated into sub-CSV files (CustomerDetails, InsuaranceDetails, CustomerLocations, PolicyType and Policy) and categorized related data into same csv files. Then csv files were imported into the tables which were in the newly created database called IT18125726_SourceDB (Except CustomerLocation)

And Customer Location details saved into text file format. This file contains all the customer Location Information.

	A	B	C	D	E	F	G	H
1	CustomerID	Customer	Education	EmploymentStatus	Gender	Marital Status	Vehicle Class	Vehicle Size
2	1	BU79786	Bachelor	Employed	F	Married	Two-Door Car	Medsize
3	2	QZ44356	Bachelor	Unemployed	F	Single	Four-Door Car	Medsize
4	3	AI49188	Bachelor	Employed	F	Married	Two-Door Car	Medsize
5	4	WW63253	Bachelor	Unemployed	M	Married	SUV	Medsize
6	5	HB64268	Bachelor	Employed	M	Single	Four-Door Car	Medsize
7	6	OC83172	Bachelor	Employed	F	Married	Two-Door Car	Medsize
8	7	XZ87318	College	Employed	F	Married	Four-Door Car	Medsize
9	8	CF85061	Master	Unemployed	M	Single	Four-Door Car	Medsize
10	9	DY87989	Bachelor	Medical Leave	M	Divorced	Four-Door Car	Medsize
11	10	BQ94931	College	Employed	F	Married	Four-Door Car	Medsize
12	11	SX51350	College	Unemployed	M	Single	Four-Door Car	Small
13	12	VQ65197	College	Unemployed	F	Married	SUV	Medsize
14	13	DP39365	Master	Employed	M	Married	Four-Door Car	Medsize
15	14	SJ95423	High School or Below	Employed	M	Married	SUV	Medsize
16	15	IL66569	College	Employed	M	Single	Four-Door Car	Medsize
17	16	BW63560	Bachelor	Employed	F	Married	Four-Door Car	Medsize
18	17	FV94802	High School or Below	Medical Leave	M	Married	Two-Door Car	Medsize
19	18	OE15005	College	Medical Leave	M	Married	SUV	Medsize
20	19	WC83389	College	Employed	M	Married	Four-Door Car	Medsize
21	20	FL50705	High School or Below	Employed	F	Married	Four-Door Car	Small
22	21	ZK25313	High School or Below	Employed	M	Single	Two-Door Car	Medsize
23	22	SV62436	Bachelor	Disabled	F	Divorced	Four-Door Car	Medsize
24	23	YH23384	Bachelor	Medical Leave	M	Divorced	Four-Door Car	Medsize
25	24	TZ98966	Bachelor	Unemployed	F	Single	Four-Door Car	Medsize
26	25	HM55802	Bachelor	Disabled	F	Married	Four-Door Car	Medsize
27	26	FS42516	College	Employed	M	Married	Four-Door Car	Large
28	27	US89481	Bachelor	Unemployed	F	Single	Four-Door Car	Small
29	28	HO30839	Master	Disabled	F	Married	SUV	Medsize

CustomerDetails.CSV

	A	B	C	D	E	F	G	H	I	J	K
1	InsuranceID	Coverage	CustomerId	PolicyID	Customer Lifetime Value	Monthly rate [Premium Auto]	Months Since Last Claim	Months Since Policy Inception	Number of Open Complaints	Effective To Date	Total Claim Amount
2	1	Basic	1	1	2763.519279	69	32	5	0	1/1/2011	384.811147
3	2	Extended	2	2	6979.535903	94	13	42	0	1/2/2011	1131.464935
4	3	Premium	3	2	12887.43165	108	18	38	0	1/3/2011	566.472247
5	4	Basic	4	3	7645.861827	106	18	65	0	1/5/2011	529.881344
6	5	Basic	5	4	2813.692575	73	12	44	0	1/6/2011	138.130879
7	6	Basic	6	2	8256.2978	69	14	94	0	1/7/2011	159.383042
8	7	Basic	7	1	5380.898636	67	0	13	0	1/8/2011	321.6
9	8	Premium	8	1	7216.100311	101	0	68	0	1/9/2011	363.02968
10	9	Basic	9	1	24127.50402	71	13	3	0	1/10/2011	511.2
11	10	Extended	10	5	7388.178085	93	17	7	0	1/11/2011	425.527834
12	11	Basic	11	2	4738.992022	67	23	5	0	1/12/2011	482.4
13	12	Basic	12	2	8197.197078	110	27	87	0	1/13/2011	528
14	13	Premium	13	6	8798.797003	110	9	82	2	1/14/2011	472.029737
15	14	Basic	14	1	8819.018934	110	23	25	1	1/15/2011	528
16	15	Basic	15	1	5384.431665	70	21	10	2	1/16/2011	307.139132
17	16	Basic	16	3	7463.139377	64	12	50	1	1/17/2011	42.920271
18	17	Basic	17	2	2566.867823	67	14	7	0	1/18/2011	454.245098
19	18	Basic	18	7	3945.241604	101	12	59	0	1/19/2011	647.442031
20	19	Basic	19	7	5710.333115	72	9	1	0	1/20/2011	308.981664
21	20	Premium	20	3	8162.617053	101	11	21	0	1/21/2011	484.8
22	21	Basic	21	7	2872.051273	74	31	21	0	1/22/2011	355.2
23	22	Extended	22	2	3041.791561	79	8	49	0	1/23/2011	379.2
24	23	Basic	23	2	24127.50402	71	13	3	0	1/24/2011	511.2
25	24	Basic	24	1	2450.190996	73	4	44	3	1/25/2011	554.376763
26	25	Basic	25	3	2392.10789	61	30	91	0	1/26/2011	439.2
27	26	Basic	26	2	5802.065978	72	22	1	0	1/28/2011	389.185006
28	27	Premium	27	2	3946.372085	111	15	47	0	1/29/2011	799.2

Insurance.CSV

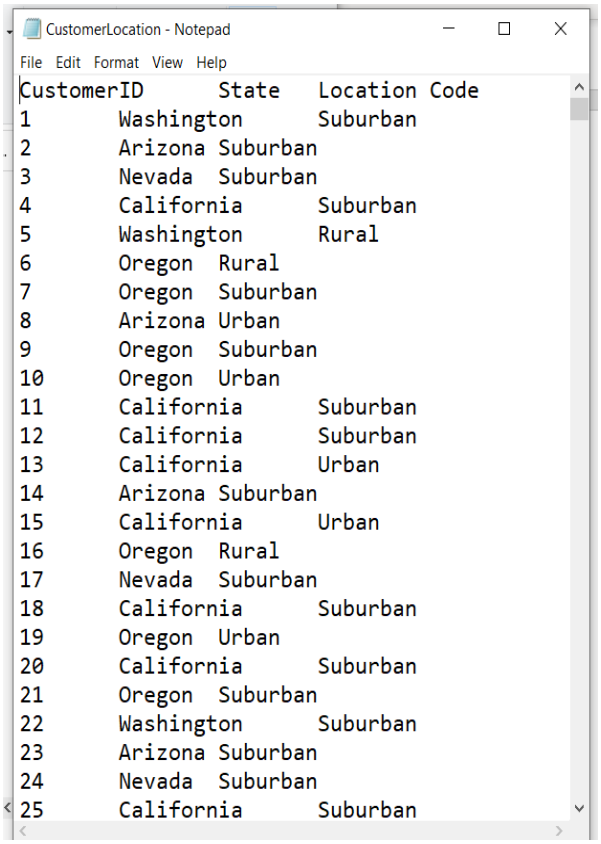
	A	B	C
1	PolicyID	PolicyTypeID	PolicyName
2	1	1	Corporate L3
3	2	2	Personal L3
4	3	1	Corporate L2
5	4	2	Personal L1
6	5	3	Special L2
7	6	1	Corporate L1
8	7	2	Personal L2
9	8	3	Special L1
10	9	3	Special L3
11			

Policy.CSV

	A	B	C	D	E
1	PolicyID	Policy Typ	Renew Off	Response	Sales Channel
2	1	Corporate	Offer1	No	Agent
3	2	Personal A	Offer3	No	Agent
4	3	Special Au	Offer2	No	Branch
5					

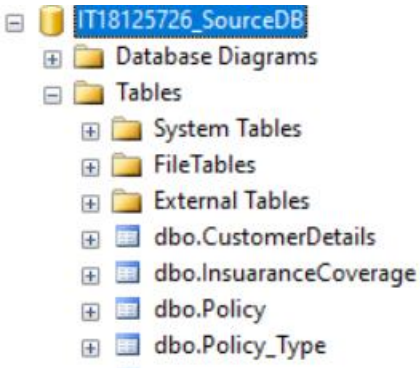
PolicyType.CSV

A	B	C
CustomerI	State	Location Code
1	Washingto	Suburban
2	Arizona	Suburban
3	Nevada	Suburban
4	California	Suburban
5	Washingto	Rural
6	Oregon	Rural
7	Oregon	Suburban
8	Arizona	Urban
9	Oregon	Suburban
10	Oregon	Urban
11	California	Suburban
12	California	Suburban
13	California	Urban
14	Arizona	Suburban
15	California	Urban
16	Oregon	Rural
17	Nevada	Suburban
18	California	Suburban
19	Oregon	Urban
20	California	Suburban
21	Oregon	Suburban
22	Washingto	Suburban
23	Arizona	Suburban
24	Nevada	Suburban
25	California	Suburban
26	Oregon	Suburban



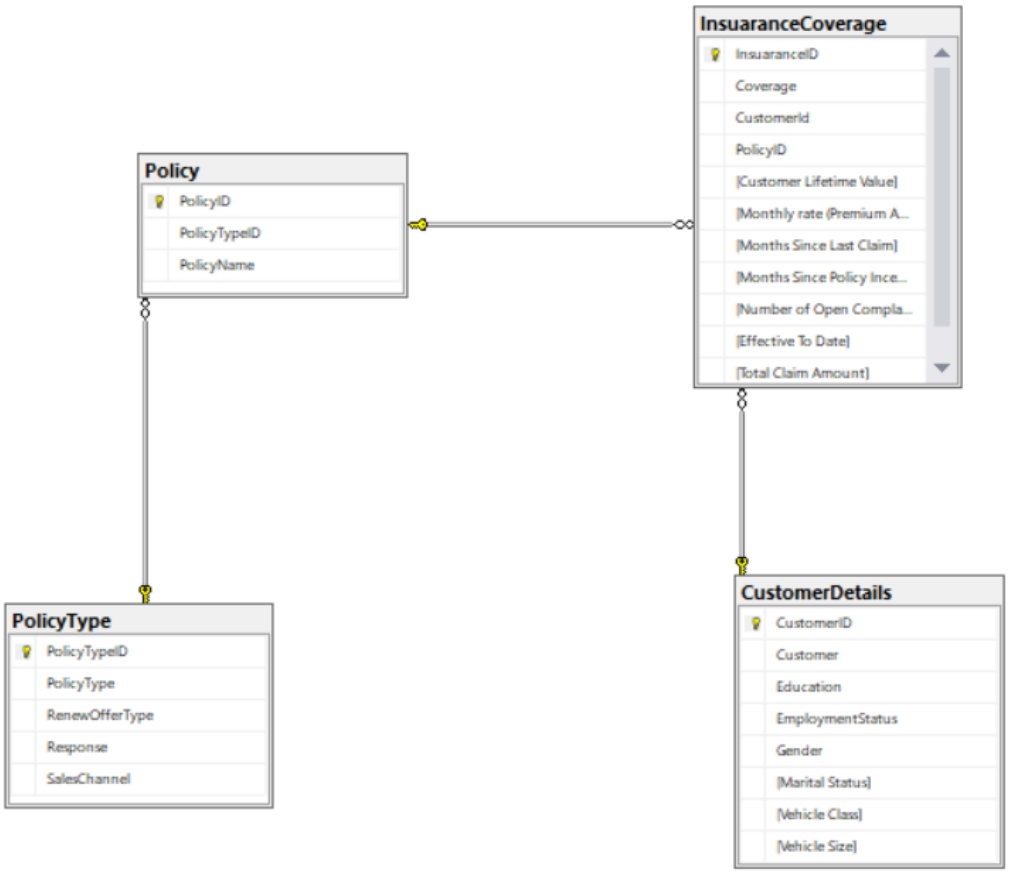
CustomerLocation.CSV

CustomerLocation.txt



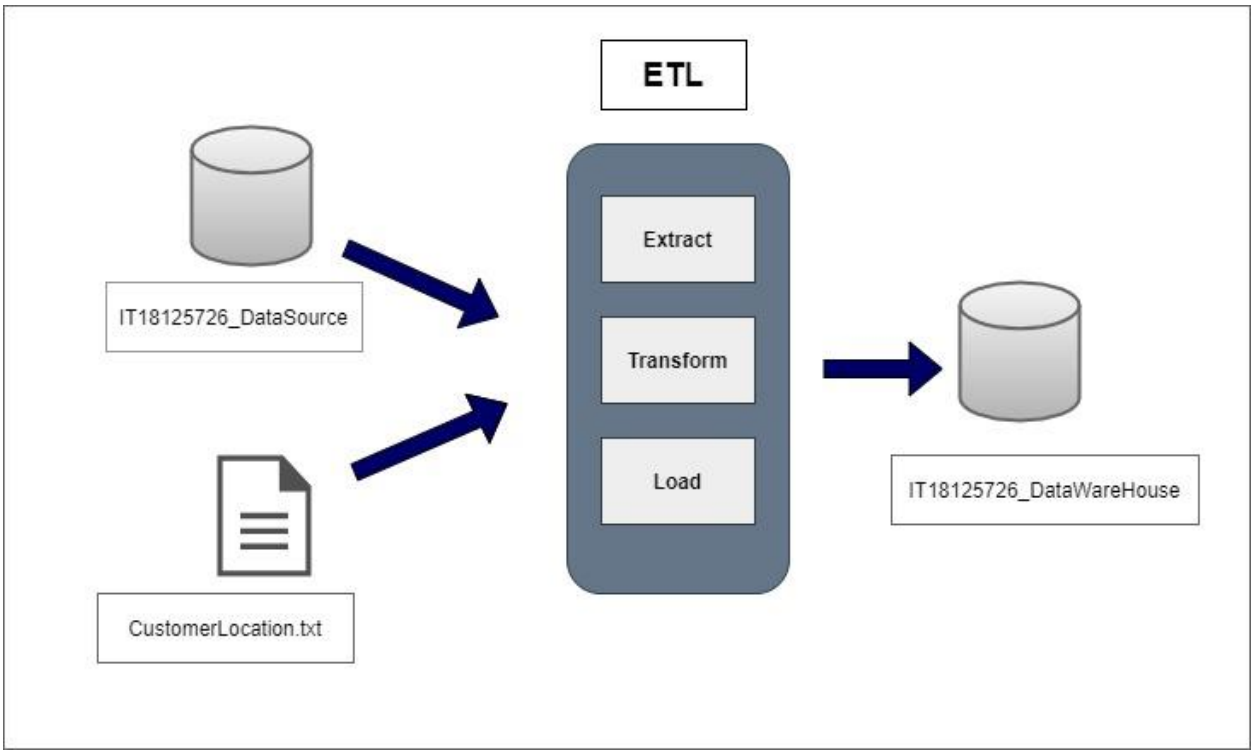
IT18125726_SourceDB

Database Diagram for IT18125726_SourceDB:



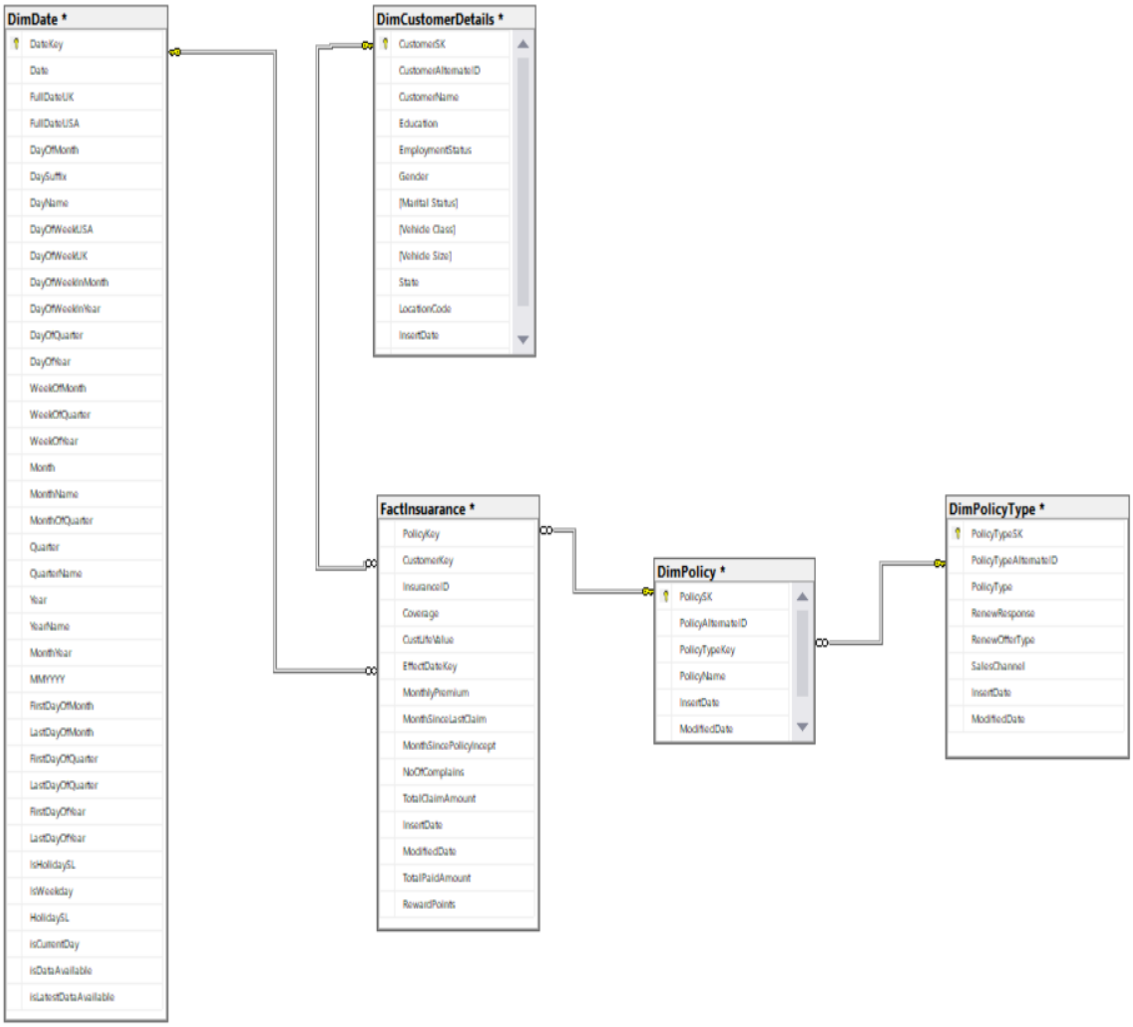
IT18125726_SourceDB

Step 03 – Solution Architecture



Using different processes, architectures, and technologies we can manage data from various sources and convert them to business insights to make decisions, analysis data and report building. This will bring new dimension to the data as well.

Step 04: Data Warehouse design & development



Assumptions

- Here I have used snowflake schema for data warehouse design and add 3 dimensions apart from the date dimension.
- I have taken Dim CustomerDetails as slowly changing dimension, customer location code and state can change time to time, and we need to keep track of their historical location details.

Before creating the Insurance fact table & other dimensions, started by creating the Date dimension using ‘DateMaster.sql’ file code dimension using ‘DateMaster.sql’ file code

MSI.IT18125726_DW...imCustomerDetails

Column Name	Data Type	Allow Nulls
CustomerSK	int	☐
CustomerAlternateID	int	☐
CustomerName	varchar(50)	☑
Education	varchar(50)	☑
EmploymentStatus	varchar(50)	☑
Gender	varchar(50)	☑
[Marital Status]	varchar(50)	☑
[Vehicle Class]	varchar(50)	☑
[Vehicle Size]	varchar(50)	☑
State	varchar(50)	☑
LocationCode	varchar(50)	☑
InsertDate	datetime	☑
ModifiedDate	datetime	☑
		☐

MSI.IT18125726_DW - dbo.DimPolicy

Column Name	Data Type	Allow Nulls
PolicySK	int	☐
PolicyAlternateID	int	☐
PolicyTypeKey	int	☑
PolicyName	nvarchar(255)	☑
InsertDate	datetime	☑
ModifiedDate	datetime	☑
		☐

MSI.IT18125726_DW...dbo.DimPolicyType

Column Name	Data Type	Allow Nulls
PolicyTypeSK	int	☐
PolicyTypeAlternateID	int	☐
PolicyType	nvarchar(255)	☑
RenewResponse	nvarchar(255)	☑
RenewOfferType	nvarchar(255)	☑
SalesChannel	nvarchar(255)	☑
InsertDate	datetime	☑
ModifiedDate	datetime	☑
		☐

MSI.IT18125726_DW...bo.FactInsuarance

Column Name	Data Type	Allow Nulls
PolicyKey	int	☐
CustomerKey	int	☐
InsuranceID	int	☐
Coverage	nvarchar(255)	☑
CustLifeValue	float	☑
EffectDateKey	int	☐
MonthlyPremium	float	☑
MonthSinceLastClaim	float	☑
MonthSincePolicyIncept	float	☑
NoOfComplains	float	☑
TotalClaimAmount	float	☑
InsertDate	datetime	☑
ModifiedDate	datetime	☑
TotalPaidAmount		☑
RewardPoints		☑

MSI.IT18125726_DW - dbo.DimDate

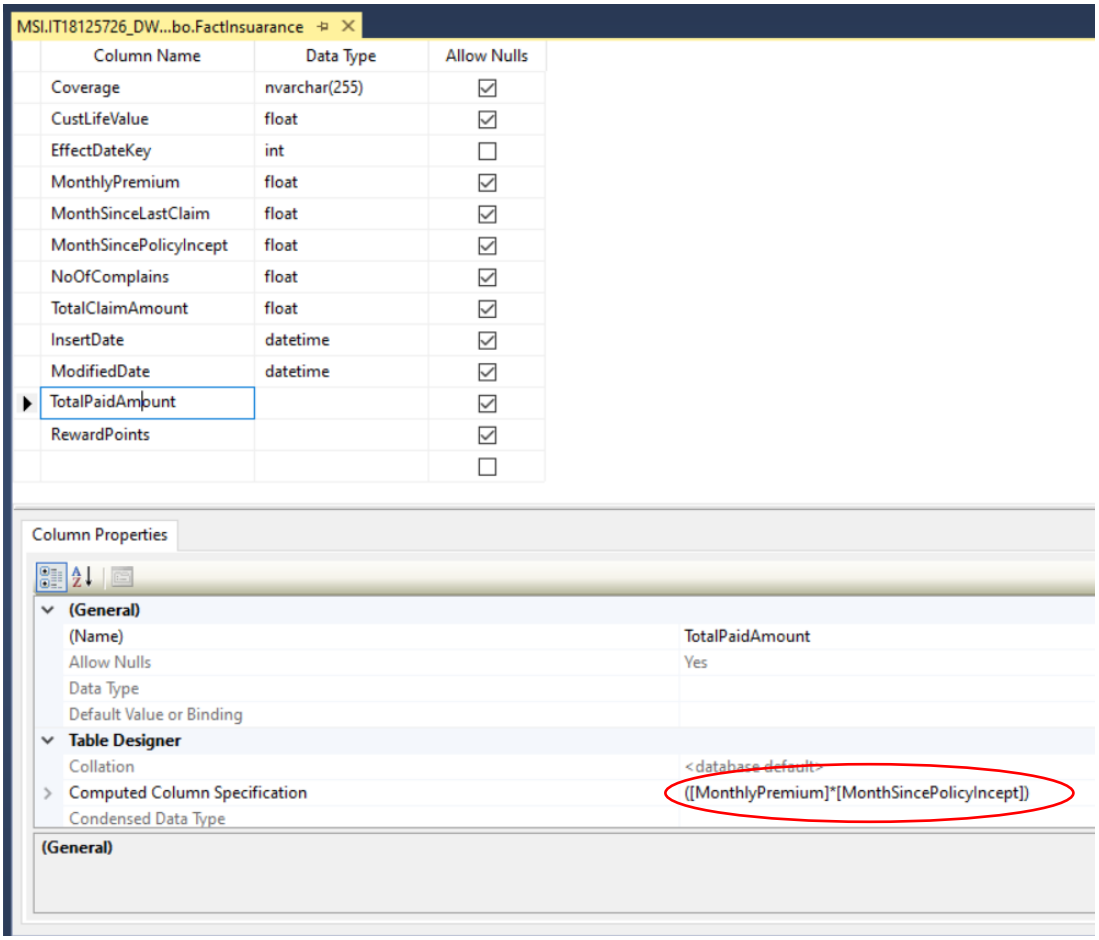
Column Name	Data Type	Allow Nulls
DateKey	int	☐
Date	datetime	☑
FullDateUK	char(10)	☑
FullDateUSA	char(10)	☑
DayOfMonth	varchar(2)	☑
DaySuffix	varchar(4)	☑
DayName	varchar(9)	☑
DayOfWeekUSA	char(1)	☑
DayOfWeekUK	char(1)	☑
DayOfWeekInMonth	varchar(2)	☑
DayOfWeekInYear	varchar(2)	☑
DayOfQuarter	varchar(3)	☑
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	☑

When crating fact tables I have added two extra attributed which are not include in my data set. I have derived those attributes using existing attributes in data set.
These derived attributes are transactional attributes which calculate essential data in fact table.

Those attributes and their functionality shown below:

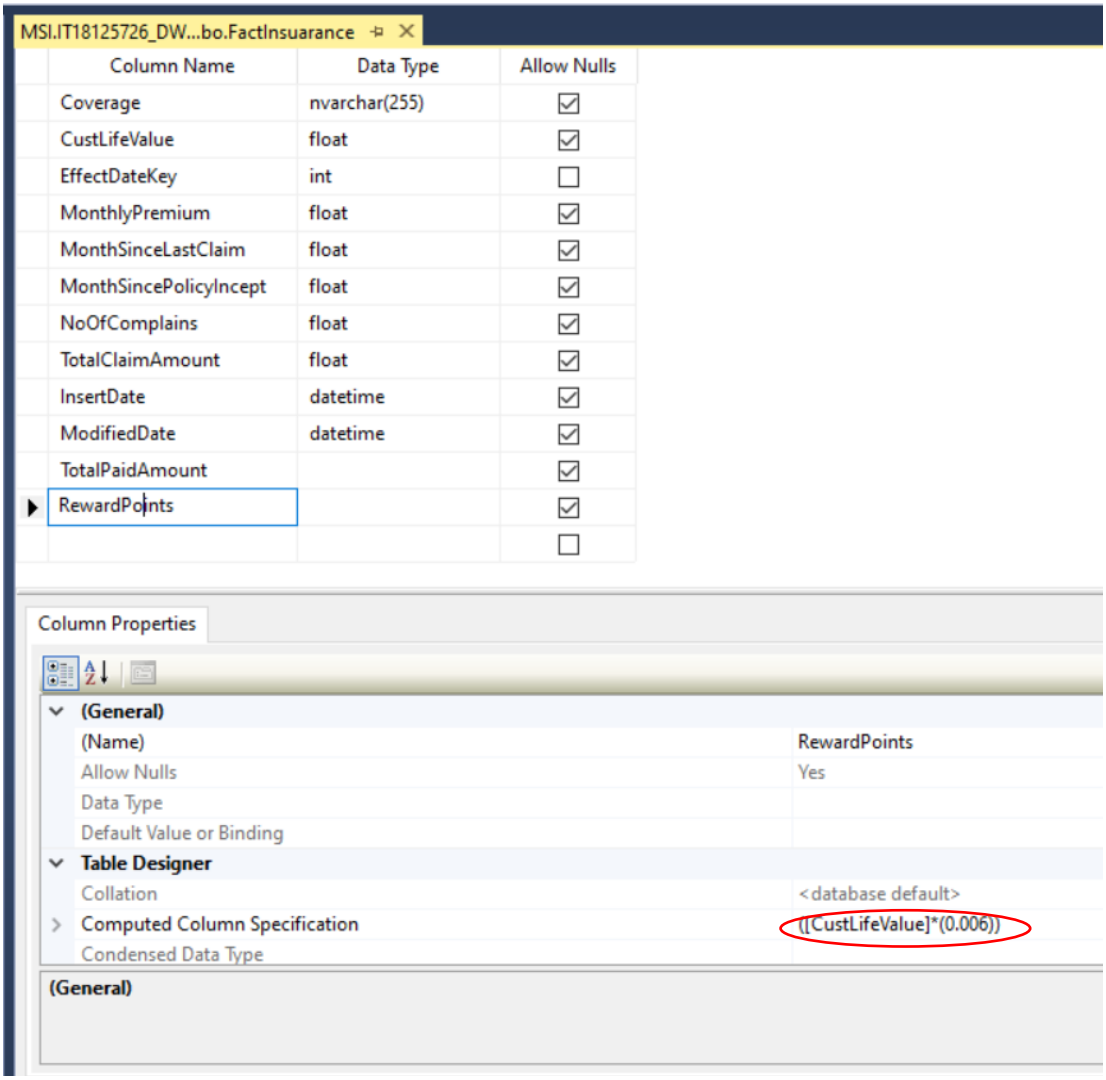
- TotalPaidAmount = (MonthlyPremium * MonthSincePolicyInception)

This will calculate the total paid amount done by customer, from the day which he or she opened their new coverage.

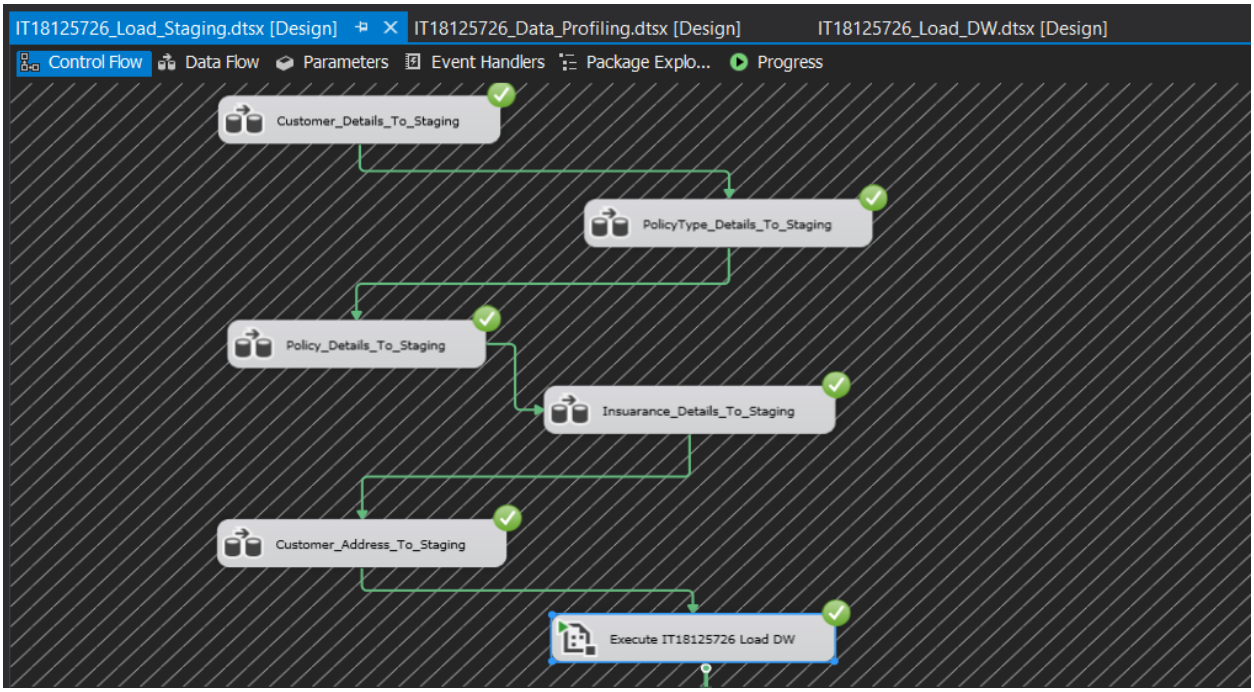


- RewardPoints = ([CustomerLifeTimeValue] * [0.006])

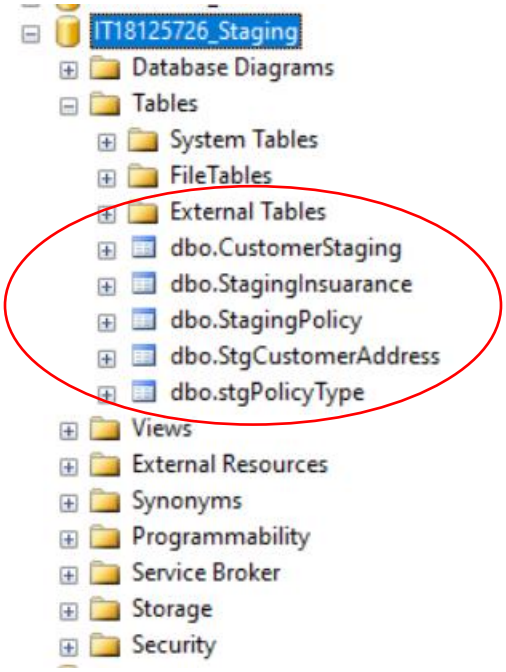
This will calculate Reward Points which will be offer to the customer based on their loyalty. I have assumed 0.006 as a constant value which will multiplied by the customer lifetime value.



Step 06: ETL Development

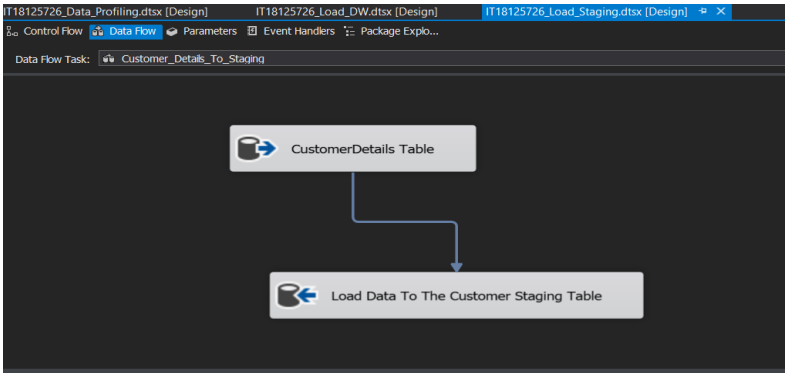


First extracted all the data from the tables which are in the IT18125726_SourceDB and CustomerLocation.txt to the separate staging DB called IT18125726_Load_Staging as shown in the below using SQL Server Integration Service Software.

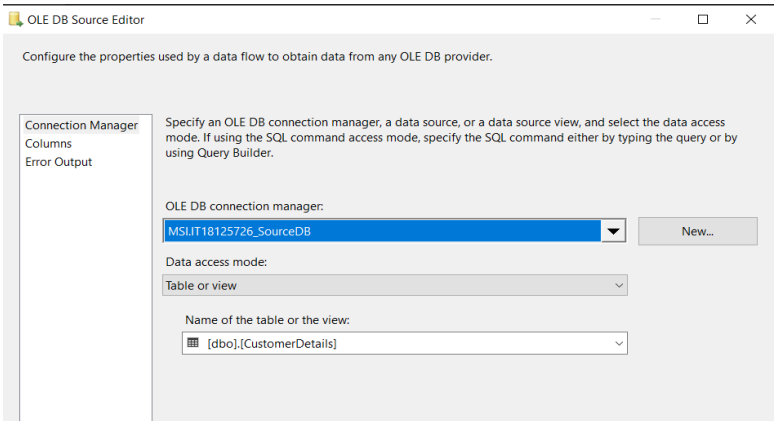


IT18125726_Staging

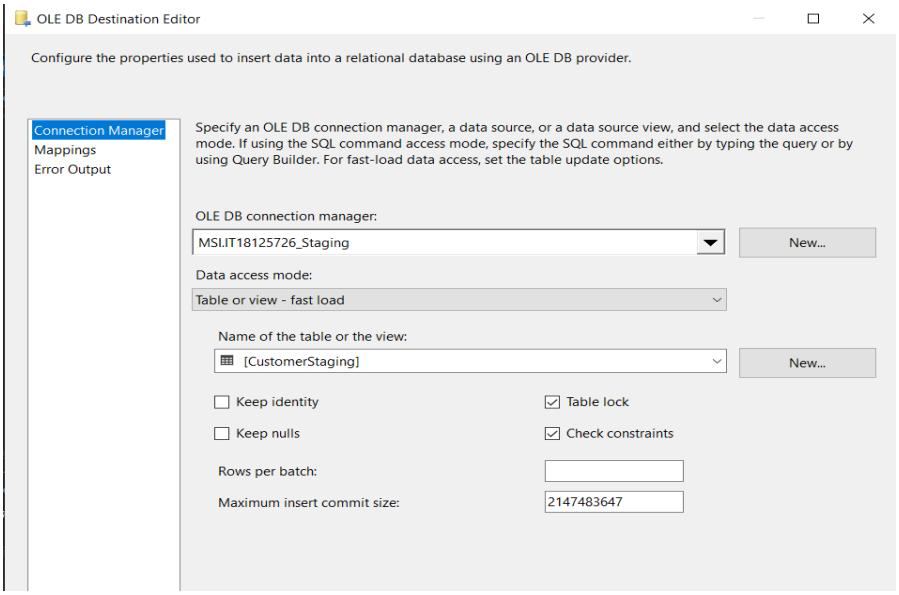
Extract Customer Details



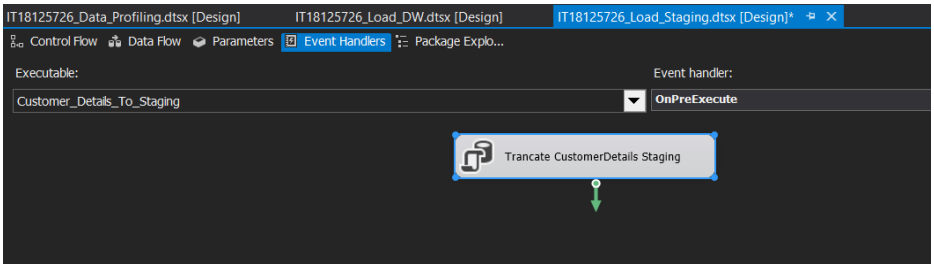
Used OLEDB data source as dbo.CustomerDetails table in IT18125726_SourceDB



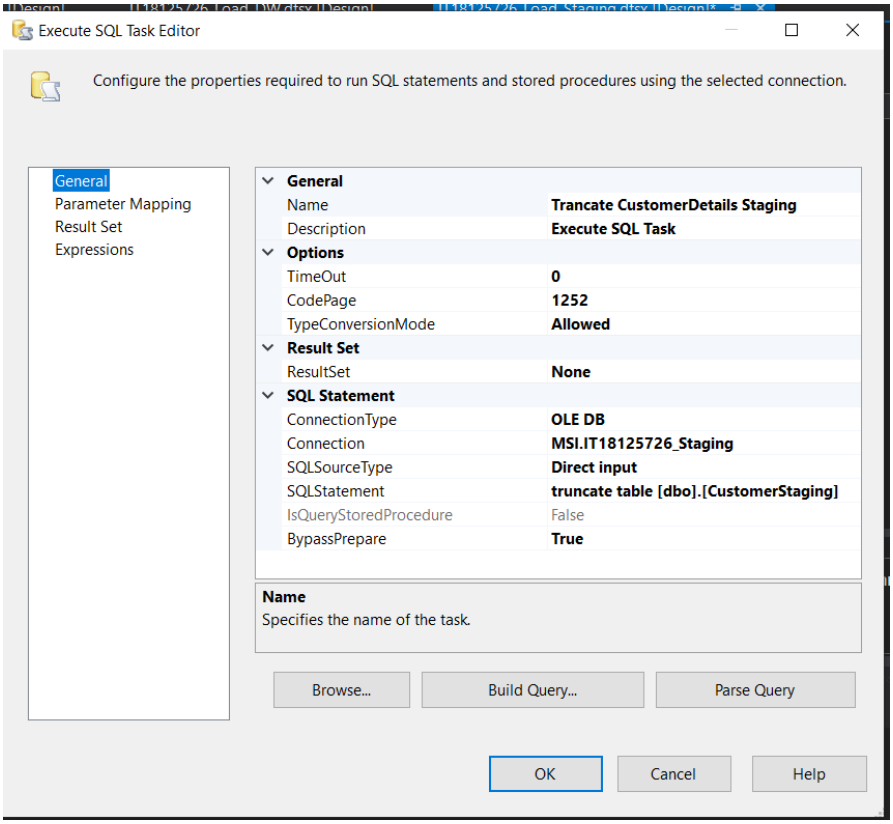
OLE DB destination for create new table as dbo.CustomerStaging in the IT18125726_Staging database



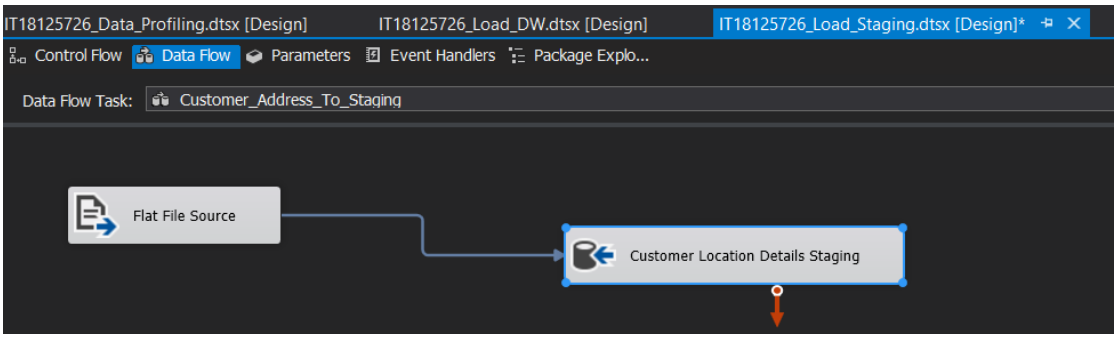
Event Handlers



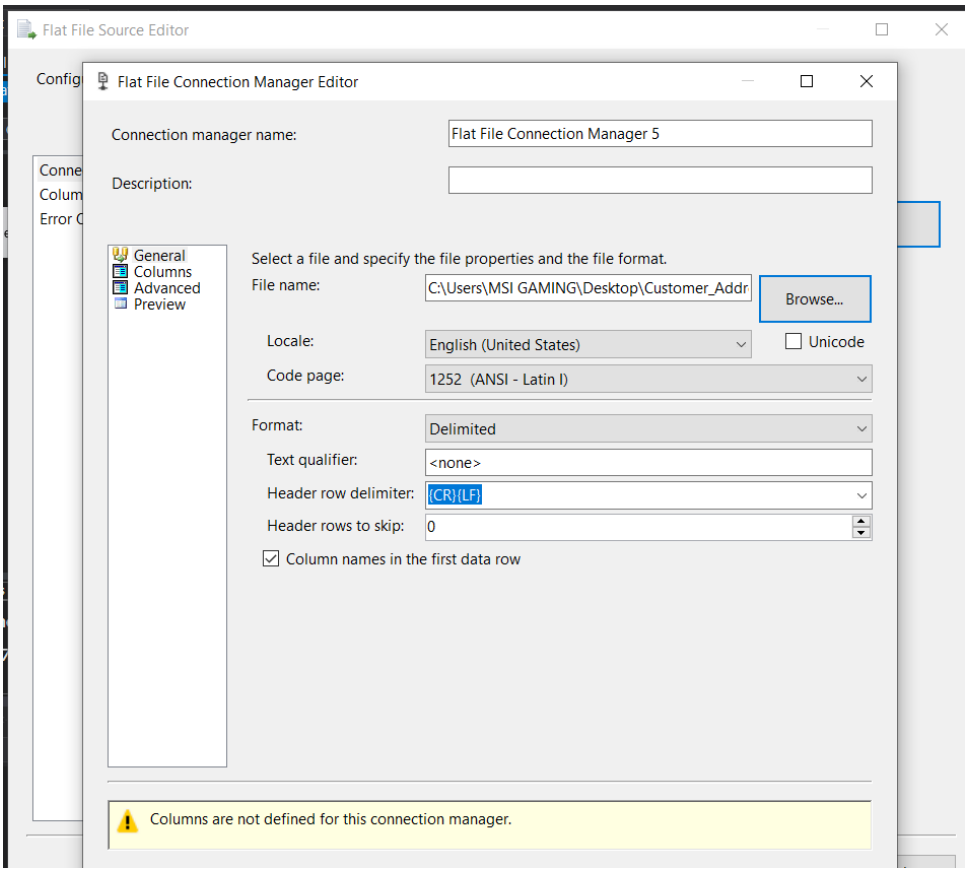
Used Execute SQL Task SSIS tool Truncate table for SQL command as truncate table dbo.CustomerStaging in IT18125726_Staging Database



Customer Address Details Extraction (Data Flow)

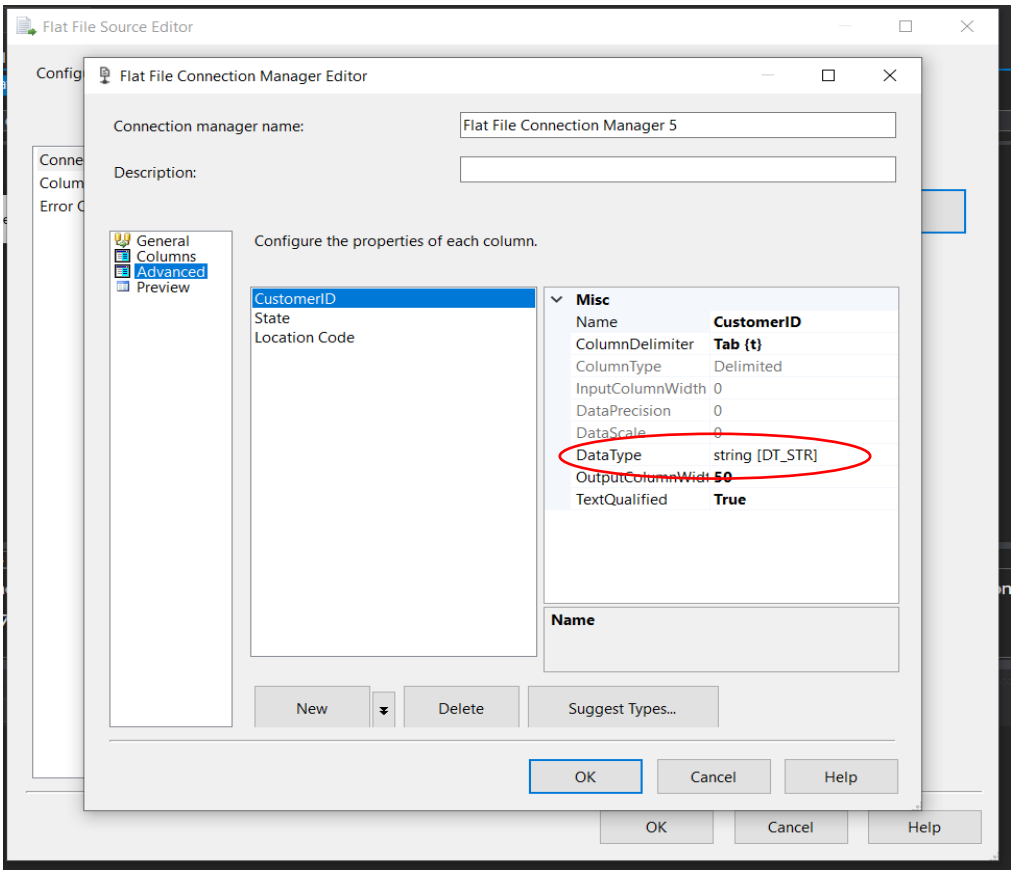


Used Flat file Source SSIS tool, to extract Customer_addresses.txt data

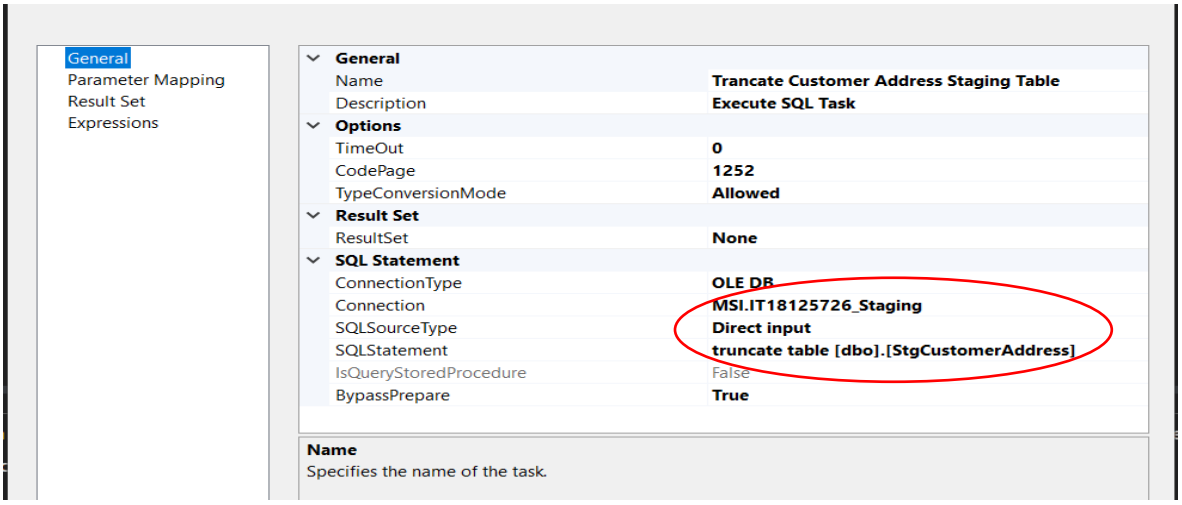


Used OLE DB Destination SSIS tool to create new table as StgCustomerAddress to load text file's data into IT18125726_Staging Database

In Advanced tab, text file data has columns then every column data type was selected as a String [DT_STR]



Used Execute SQL Task SSIS tools Truncate table for SQL command as truncate table

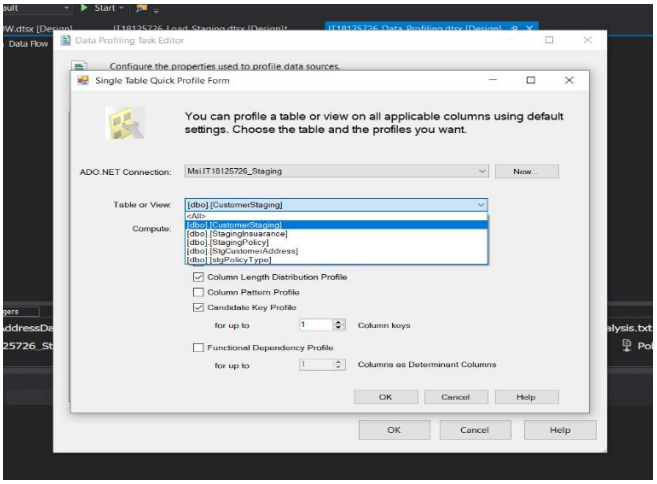


Note: Followed exact process to extract other source tables data in to staging

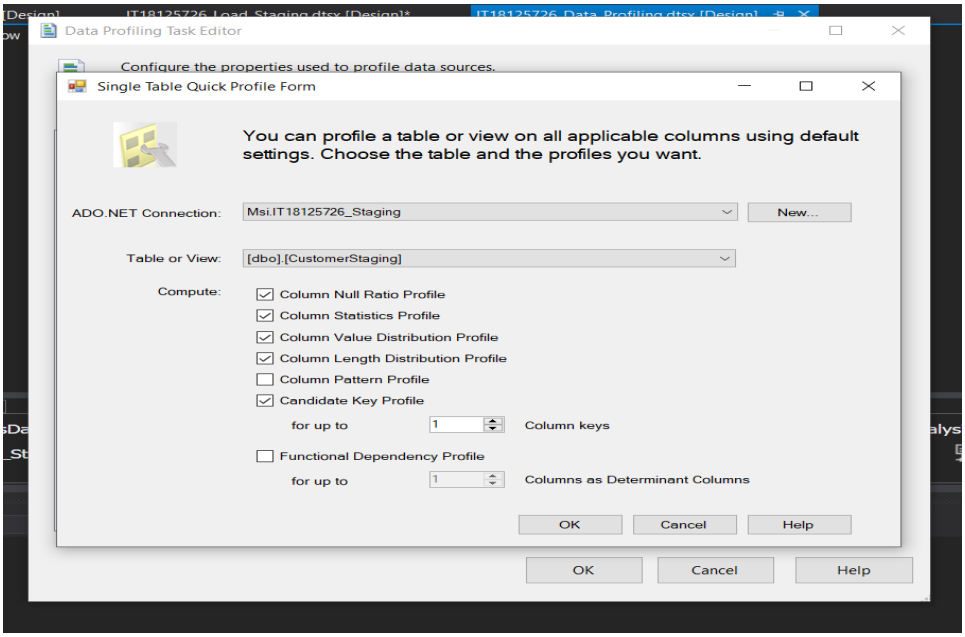
Data Profiling

Staging table data was used to analyzed and determined what types of transformation was needed to perform on the data.

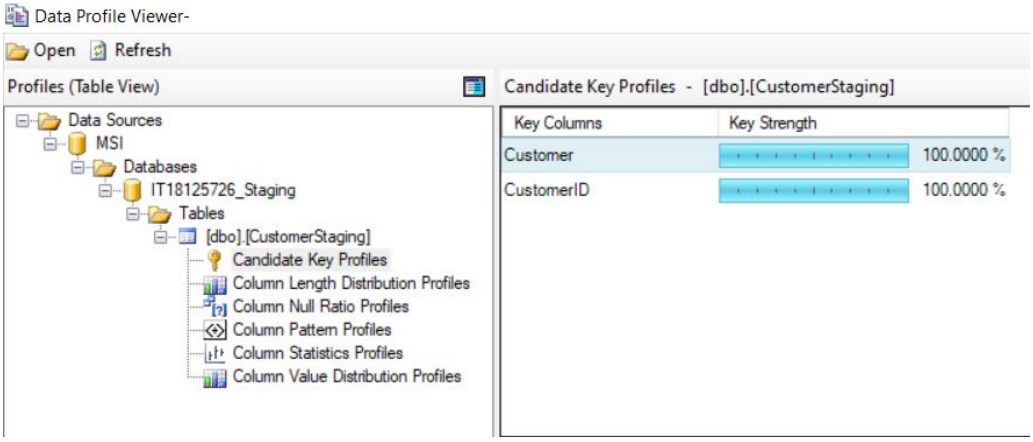
- Right click on SSIS Packages and New SSIS Package selected.
- In the Control Flow of IT18125726_Data_Profiling.dtsx, drag and drop the Data Profiling Task and double click to open the configuration.
- Clicked on Quick Profile button to open Single Table Quick Profile Form.
- Clicked on New button and create the connection to IT18125726_Staging
- From the Table or View dropdown, ConsumerDetails _staging table selected.

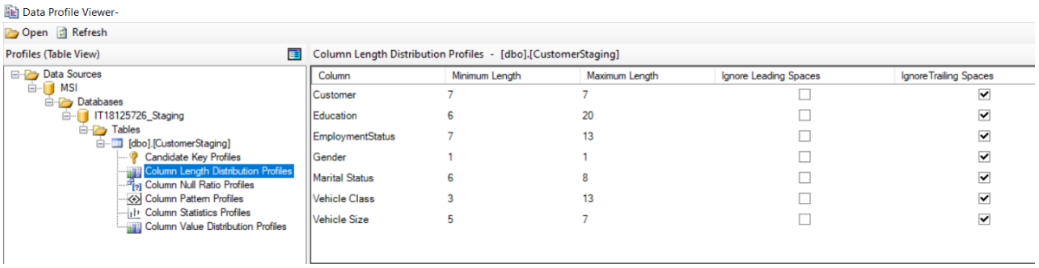


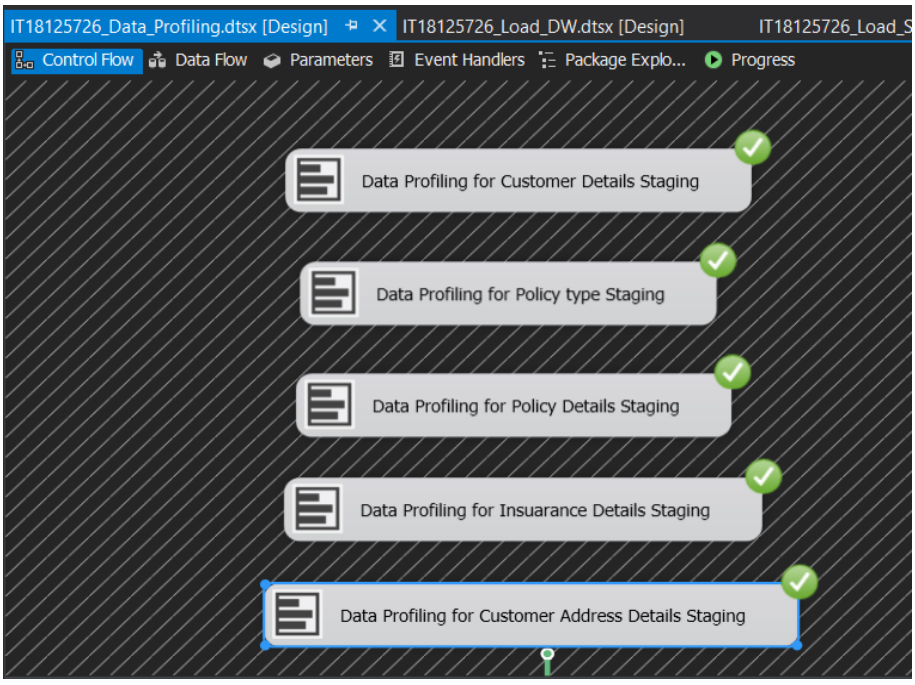
Selected all check boxes and click on OK button to complete the configuration



- Save the package and Run the Data profiling Pack to profile the ConsumerDetails
- Once the green tick appeared, double click the Data Profiling Task and Click on the Open Profile Viewer to view the analyzed data







Data Transformation

Customer Data Transformation was created by using below mentioned steps

• Created new package called IT18125726_Load_DW.dtsx.

• Then Dragged and dropped a Data Flow Task, renamed it as Transform and Load Customer Data and go the Data Flow tab.

• Dragged and dropped OLE DB Source, renamed as Extract from Customer Staging and configure it to access the CustomerStaging table.

• And I used another OLE DB Source, renamed as Extract from Customer Address Staging and configure it to access the StgCustomerAddress table. And selected all the columns.

• Then I Dragged and dropped two Sort items and connect each OLE DB Source to them.

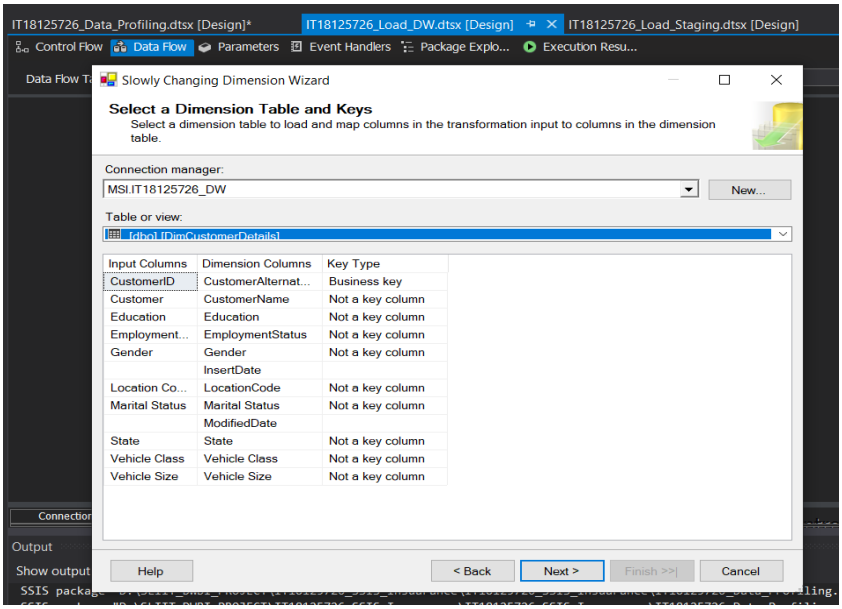
• After that I Double click Sort that is connected to Extract from Customer Staging and select CustomerID as the Sort option by ticking on the checkbox in from of Customer_ID Then I did the same for the other Sort item connected to Customer Address Staging.

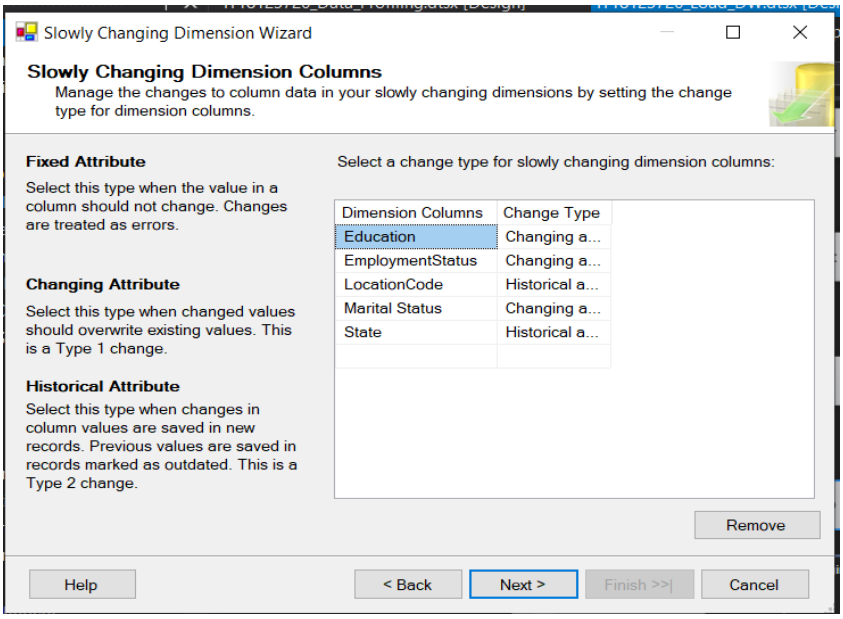
• Dragged and dropped Merge Join and link above two sort items to the Merge join.

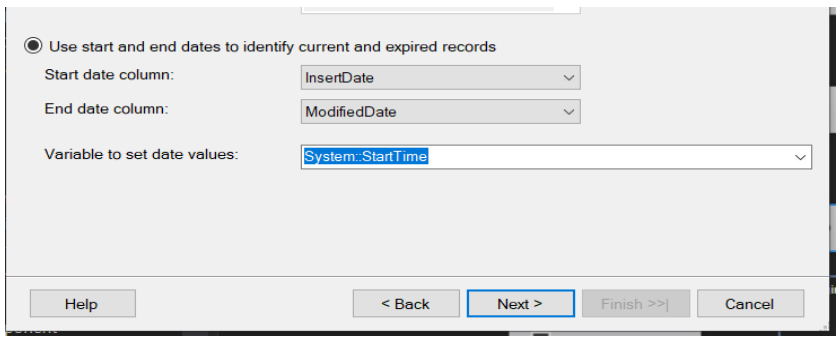
• In the Input Output Selection popup, I have selected Merge Join Left Input.

• After that I dragged and dropped Slowly Changing Dimension item and connect the last Merge Join to that.

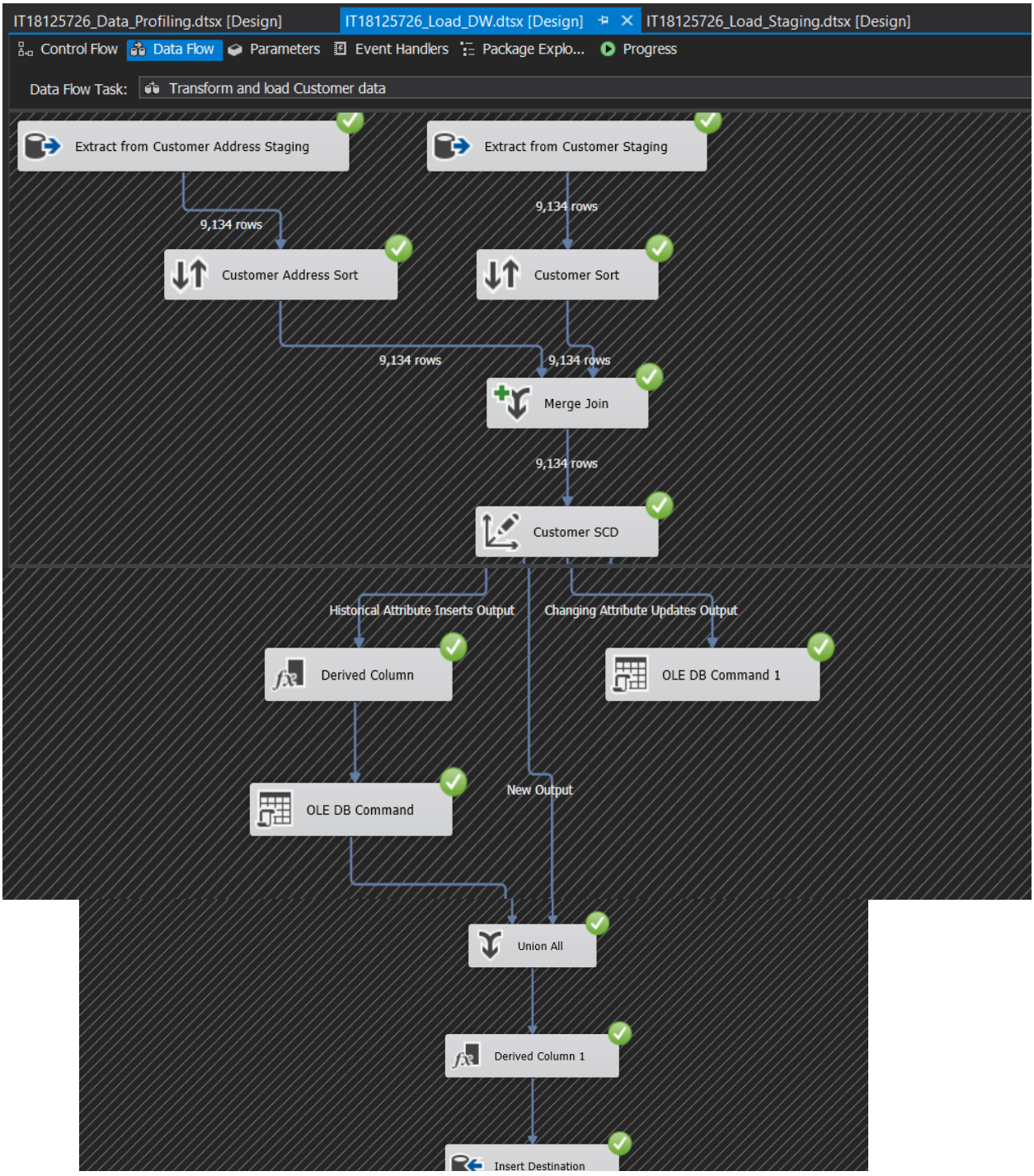
• In the SCD Configuration Wizard I set the configurations as below


IT18125726_Data_Profiling.dtsx [Design]*
IT18125726_Load_DW.dtsx [Design]
IT18125726_Load_Staging.dtsx [Design]
Control Flow
Data Flow
Parameters
Event Handlers
Package Explo...
Execution Resu...
Slowly Changing Dimension Wizard
Select a Dimension Table and Keys
Select a dimension table to load and map columns in the transformation input to columns in the dimension table.
Connection manager:
MSI.IT18125726_DW
New...
Table or view:
[dbo].[DimCustomerDetails]
Input Columns
Dimension Columns
Key Type
CustomerID
CustomerAlternat...
Business key
Customer
CustomerName
Not a key column
Education
Education
Not a key column
Employment...
EmploymentStatus
Not a key column
Gender
Gender
Not a key column
InsertDate
Location Co...
LocationCode
Not a key column
Marital Status
Marital Status
Not a key column
ModifiedDate
State
State
Not a key column
Vehicle Class
Vehicle Class
Not a key column
Vehicle Size
Vehicle Size
Not a key column
Output
Show output
Help
< Back
Next >
Finish >>|
Cancel


Slowly Changing Dimension Wizard
Slowly Changing Dimension Columns
Manage the changes to column data in your slowly changing dimensions by setting the change type for dimension columns.
Fixed Attribute
Select this type when the value in a column should not change. Changes are treated as errors.
Changing Attribute
Select this type when changed values should overwrite existing values. This is a Type 1 change.
Historical Attribute
Select this type when changes in column values are saved in new records. Previous values are saved in records marked as outdated. This is a Type 2 change.
Select a change type for slowly changing dimension columns:
Dimension Columns
Change Type
Education
Changing a...
EmploymentStatus
Changing a...
LocationCode
Historical a...
Marital Status
Changing a...
State
Historical a...
Remove
Help
< Back
Next >
Finish >>|
Cancel


Use start and end dates to identify current and expired records
Start date column:
InsertDate
End date column:
ModifiedDate
Variable to set date values:
System::StartTime
Help
< Back
Next >
Finish >>|
Cancel

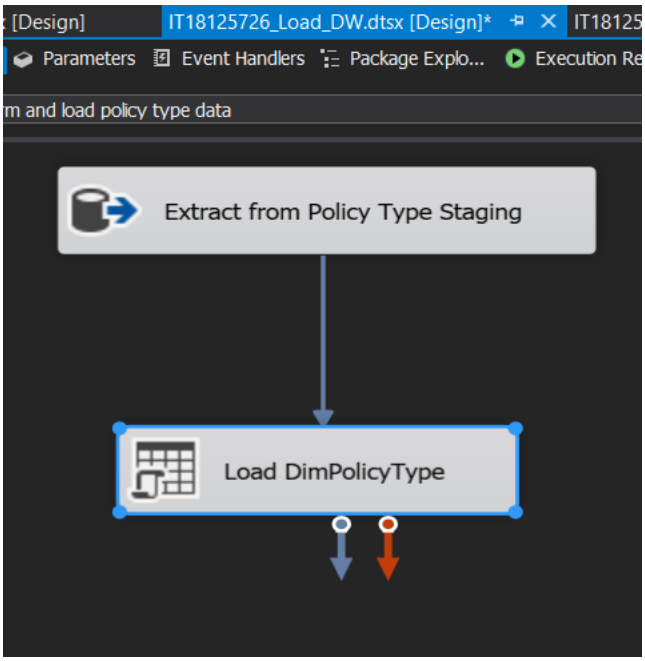
Once All Configurations done properly, it will automatically create the Slowly Changing Dimension as shown below.



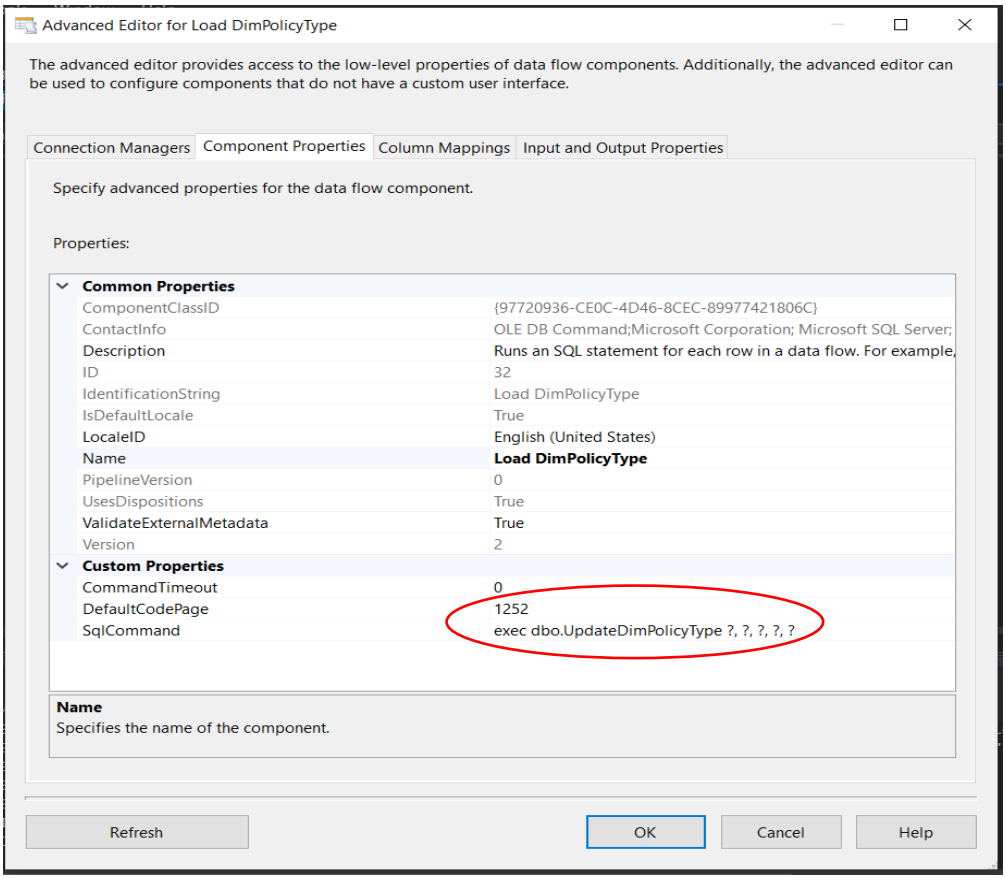
Transform & Load Policy Type

First created a Procedure called UpdateDimPolicyType and executed in the IT18125726_DW database

```
CREATE PROCEDURE [dbo]. [UpdateDimPolicyType]
@PolicyTypeID int,
@PolicyType nvarchar (255),
@Response nvarchar (255),
@RenewOfferType nvarchar(255),
@SalesChannel nvarchar(255)

AS
BEGIN
if not exists (select PolicyTypeSK
from dbo.DimPolicyType
where PolicyTypeAlternateID = @PolicyTypeID)

BEGIN
insert into dbo.DimPolicyType
(PolicyTypeAlternateID,
PolicyType,RenewResponse,RenewOfferType,SalesChannel,InsertDate,ModifiedDate)
values
(@PolicyTypeID, @Response, @PolicyType, @RenewOfferType, @SalesChannel, GETDATE
(), GETDATE())
END;
if exists (select PolicyTypeSK
from dbo.DimPolicyType
where PolicyTypeAlternateID = @PolicyTypeID)
BEGIN
update dbo.DimPolicyType
set PolicyType = @PolicyType,
RenewResponse = @Response,
RenewOfferType= @RenewOfferType,
SalesChannel = @SalesChannel,
ModifiedDate = GETDATE()

where PolicyTypeAlternateID = @PolicyTypeID
END;
END;
```



OLE DB Command SSIS tool used to execute, UpdateDimPolicyType procedure, it used to insert data into StgPolicyType to DimPolicyType without data duplication.



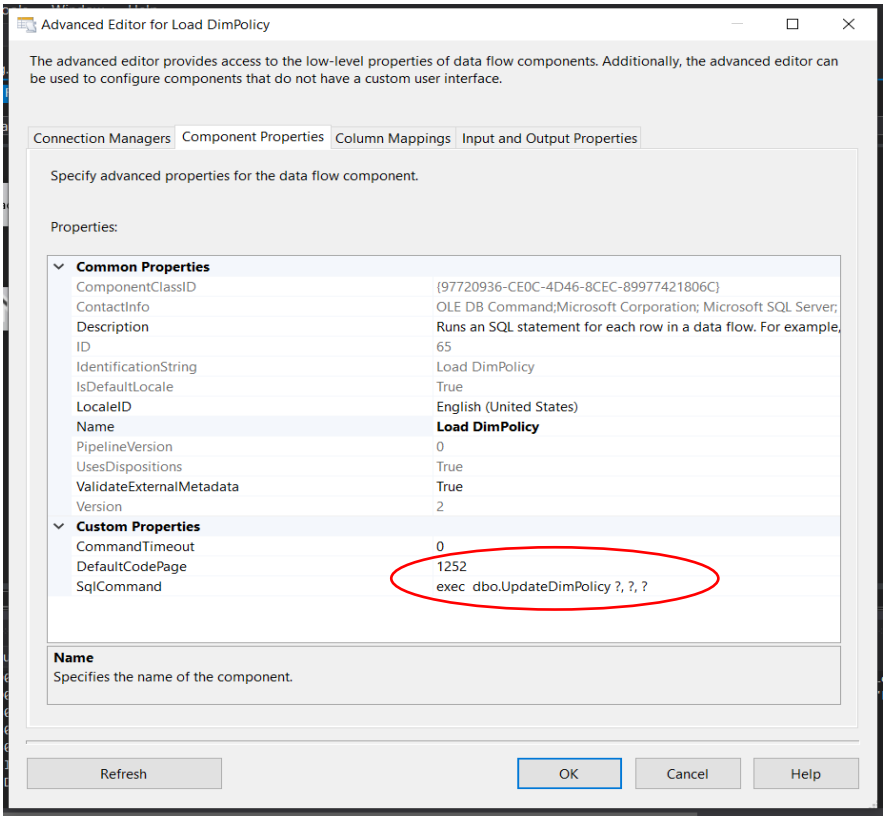
Transform and Load Policy

Used same approached which is followed in loading Policy type data into the data warehouse (using a stored procedure to insert new records and update existing records) , as policy data also does not maintain history

- Using SQL Server Management Studio, I have created the stored Procedure given below, in the IT18125726_DW database

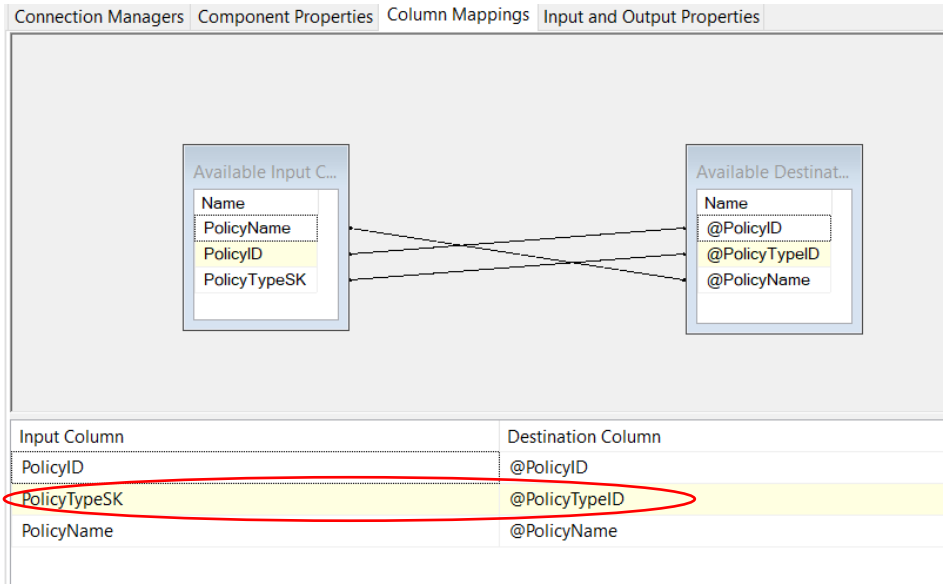
```
CREATE PROCEDURE dbo.UpdateDimPolicy
@PolicyID int,
@PolicyTypeID int,
@PolicyName nvarchar(255)

AS
BEGIN
if not exists (select PolicySK
from dbo.DimPolicy
where PolicyAlternateID = @PolicyID)

BEGIN
insert into dbo.DimPolicy
(PolicyAlternateID, PolicyTypeKey, PolicyName,InsertDate, ModifiedDate)
values
(@PolicyID,@PolicyTypeID,@PolicyName, GETDATE(), GETDATE())
END;
if exists (select PolicySK
from dbo.DimPolicy
where PolicyAlternateID = @PolicyID)
BEGIN
update dbo.DimPolicy
set PolicyTypeKey = @PolicyTypeID,
PolicyName = @PolicyName,
ModifiedDate = GETDATE()
where PolicyAlternateID = @PolicyID
END;
END;
```

Now Drag and drop OLE DB Command task, rename It as 'Load DimPolicy' link it with merge join. After double clicking on Load DimPolicy we can view Advanced Editor for Load DimPolicy window. Then used bellow code to execute above stored Procedures.

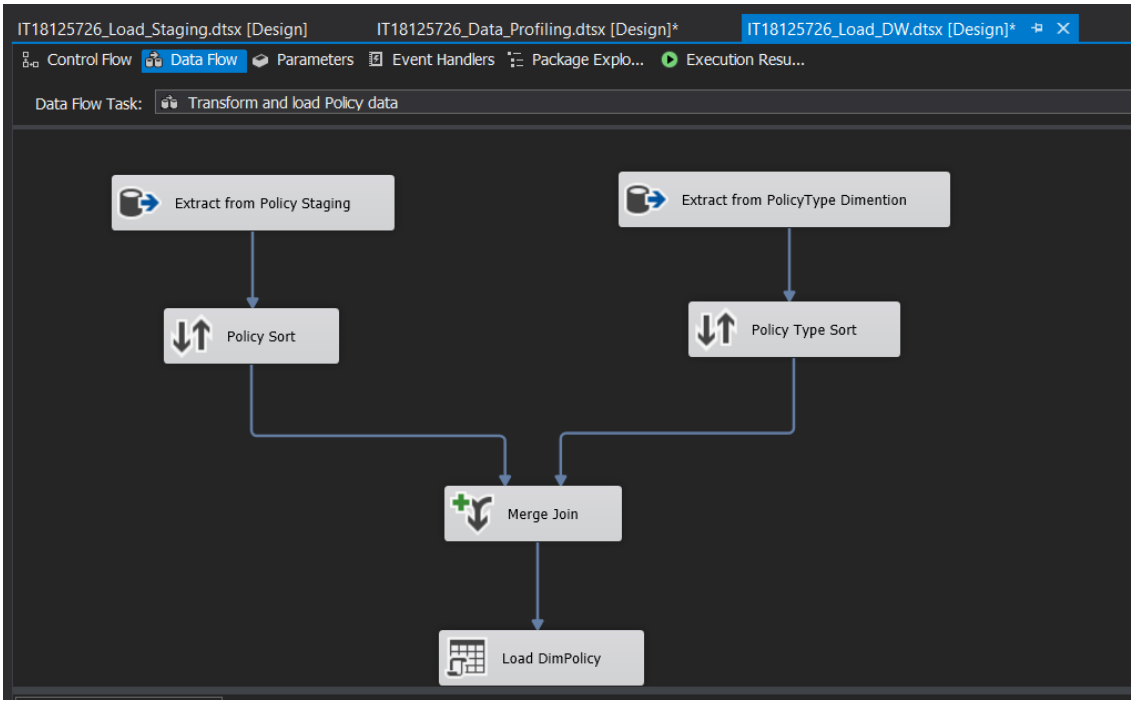


Above Stored Procedure ensure no duplicates are entered into the data warehouse table 'DimPolicy'. If there is an existing policy record, it will be updated with the latest record coming from staging table 'StgPolicy' else, if it is a new record, just insert it.

In Column Mappings tab map, the columns to the variables accordingly. here I had map PolicyTypeSK as the input for @PolicyTypeID.



Finally, 'Transform and Load Policy Data' data flow design should look like below:



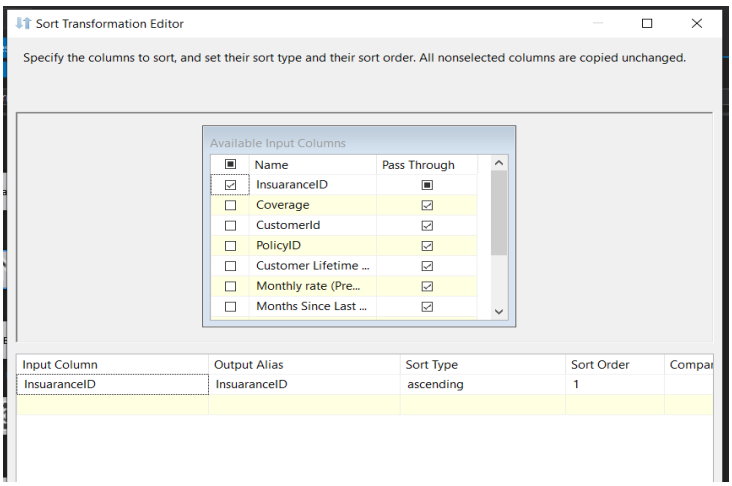
Load to Insurance Fact

In the IT18125726_DW.dtsx, add another Data Flow Task and join the previous data flow task with the new data flow task.

Renamed the new Data Flow Task as Transform and Load Insurance data

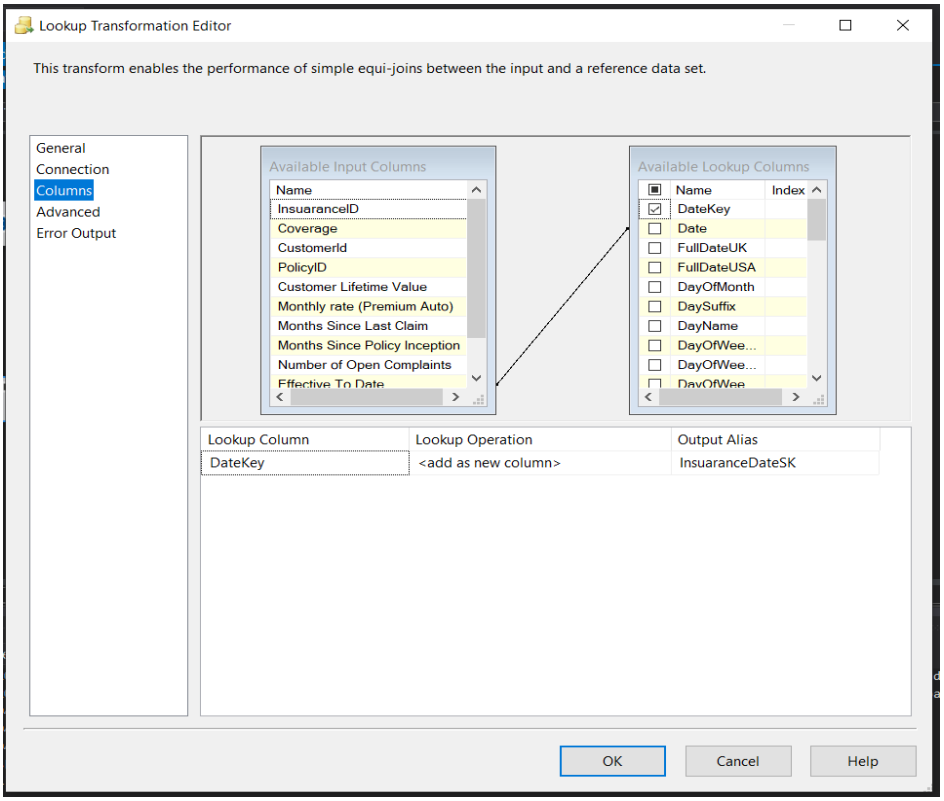
Then I dragged and dropped the OLE DB source and configure it to fetch data from dbo.StagingInsuarance table

Then I used Sort component to sort data in between sources.



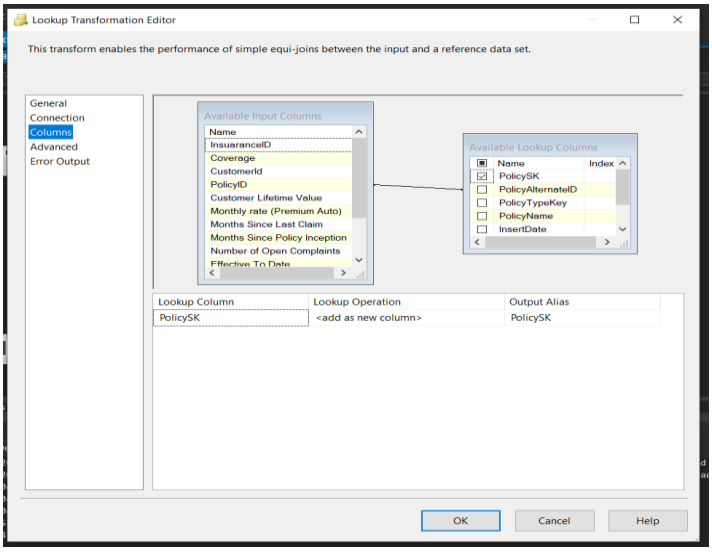
Effective to Date Lookup –

Here I map Effective to Date Input with Date lookup to obtain date key.



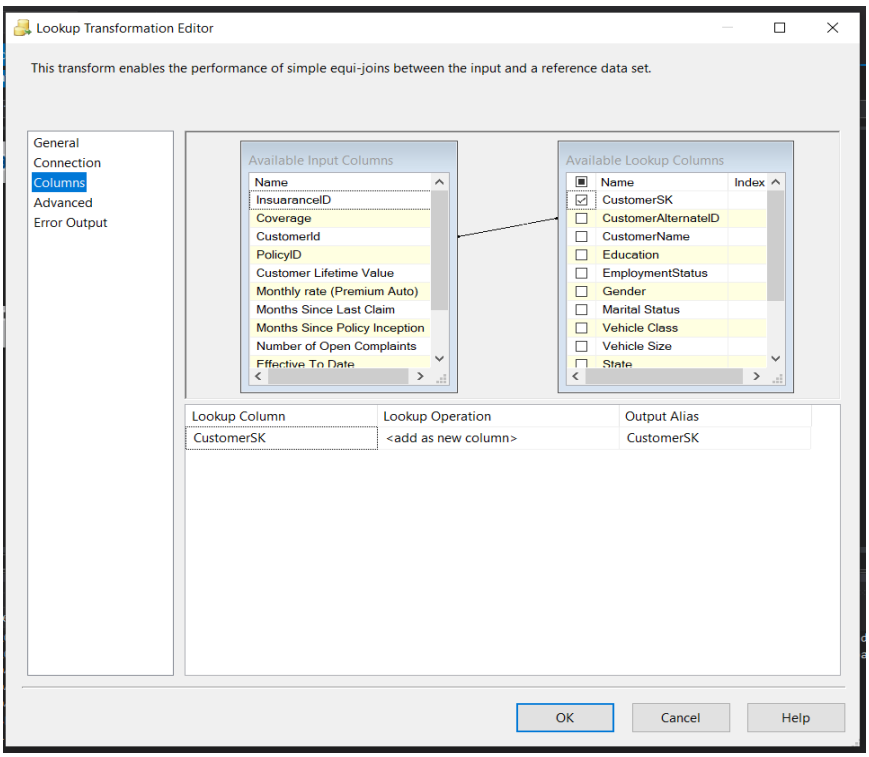
Policy Lookup –

Here I map PolicyID with PolicyAlternateID to obtain PolicySK

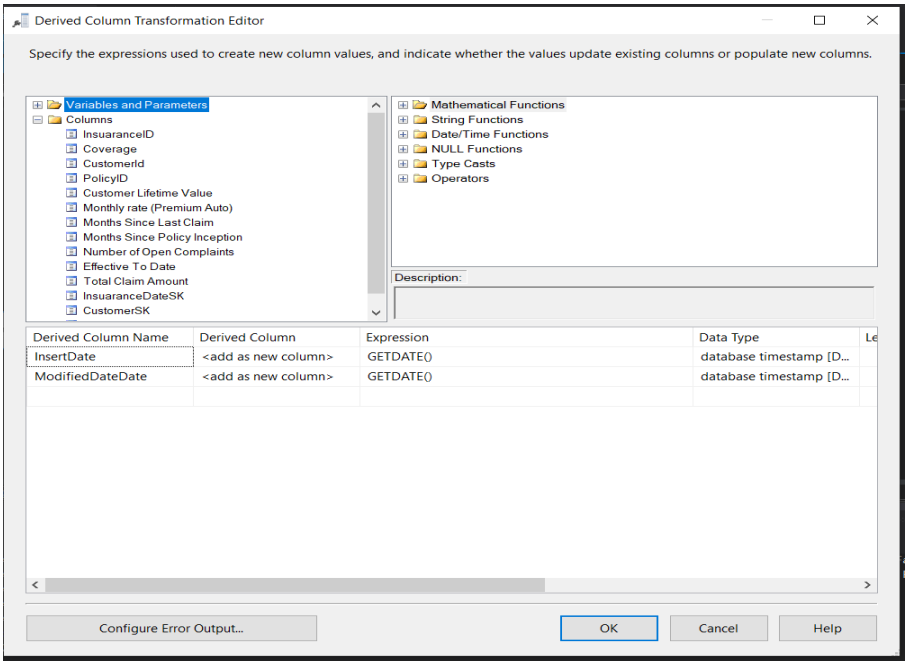


Customer Lookup

Here I map Customerid in input columns with CustomerAlternateID in Lookup column to obtain CustomerSK

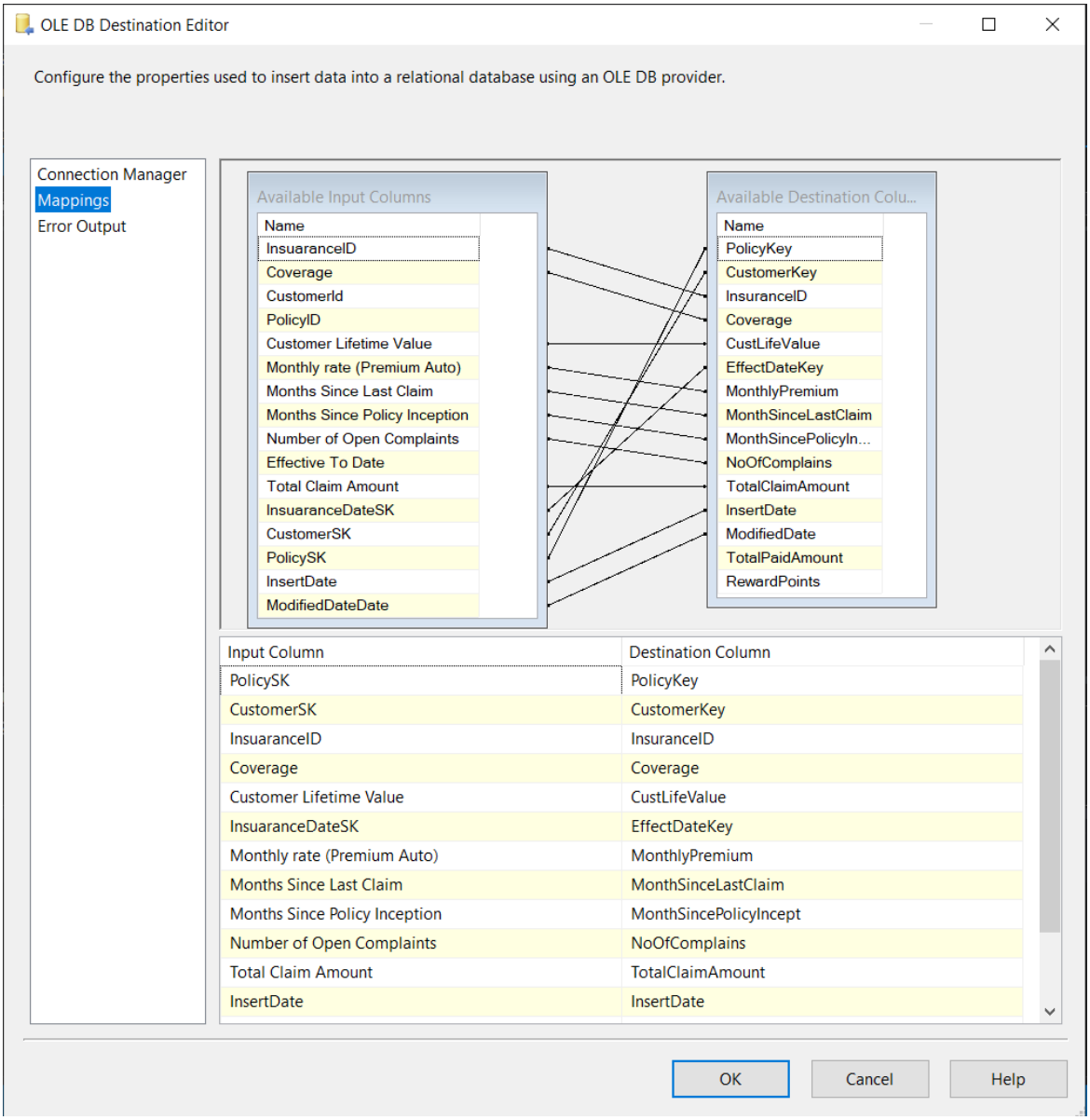


Derived Insert and Modified Date Fields –

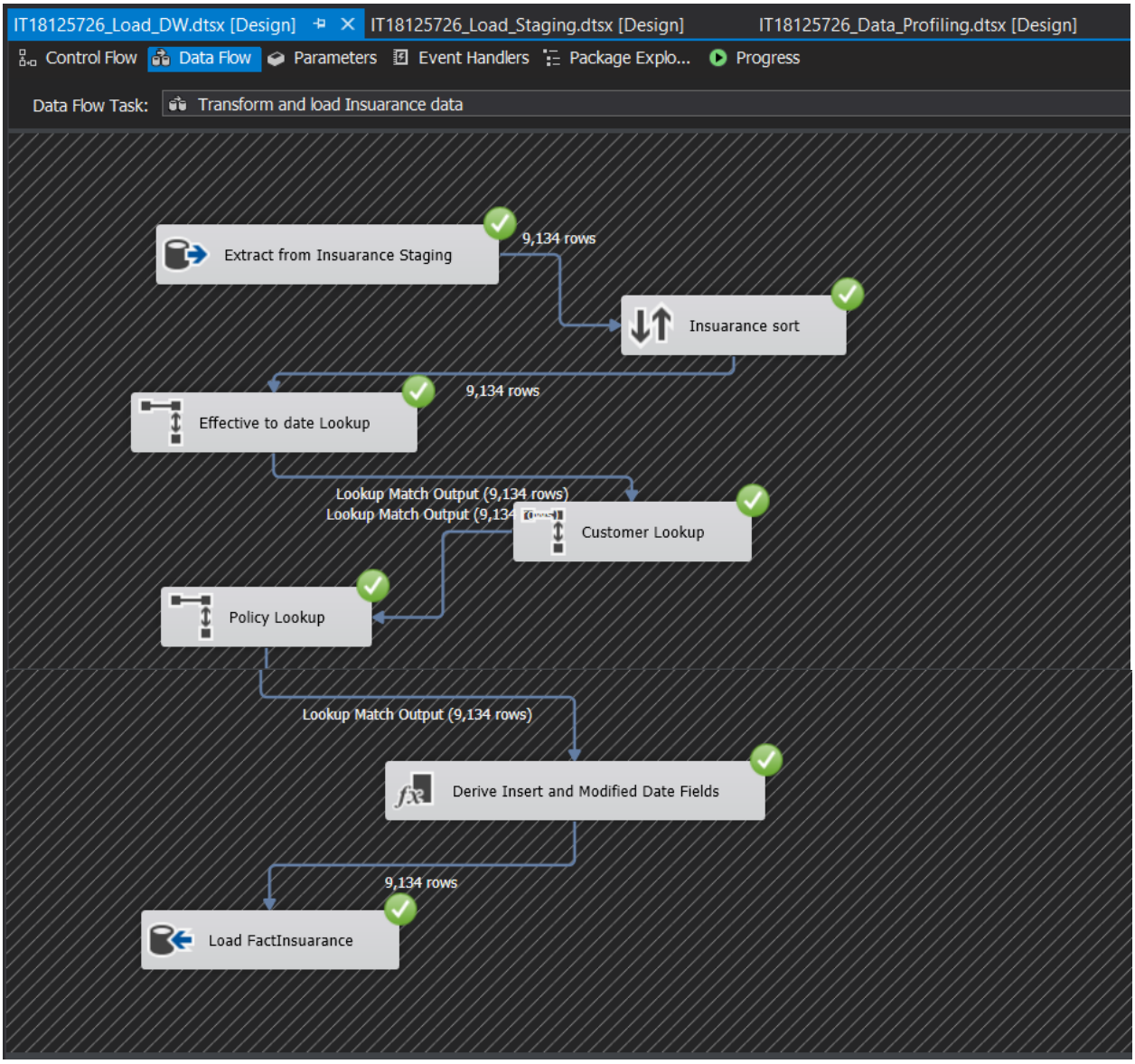


Take a another OLE DB Destination component and rename it as LooadFactInsuarance
And connected it with above ‘Derive Insert and modified Date Fields’

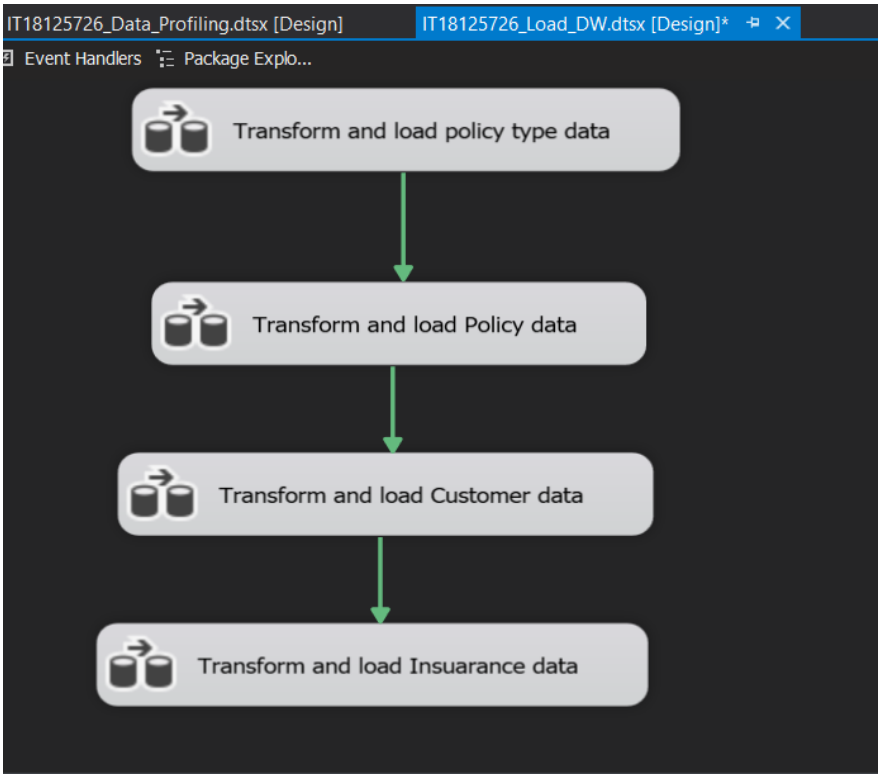
I mapped the input columns with Destination Columns as below.



Outcome of the Transform and Load FactInsuarance Table:



Final IT18125726_DW.dtsx package Control Flow:



At the end of the staging I have connected the data warehousing package to the end of the data staging package using an execute package task editor.
So, when executing the staging it will execute the data warehousing package as well

References:

https://courseweb.sliit.lk/course/view.php?id=4696

https://www.geeksforgeeks.org/etl-process-in-data-warehouse/

https://www.javatpoint.com/etl-process-in-data-warehouse

https://www.youtube.com/watch?v=oF_2uDb7DvQ

END