Sri Lanka Institute of Information Technology

IT3021- Data Warehousing and Business Intelligence

Assignment 1

2022

Submitted By:
Nirmal M.D.S
IT20074340
Batch :- Y3.S1.WD.DS.05.01

Contents

Step 01:- Data Set Selection

Data Set Name : Brazilian E-Commerce Public Dataset by Olist

Source URL :- https://www.kaggle.com/datasets/olistbr/brazilian-ecommerce

About Dataset:

This Public Dataset is obtained by a Brazilian ecommerce company. This includes order information from 2016 – 2018. There are nearly 100,000 records about orders that happened at multiple marketplaces at Brazil. This dataset gives data about the customers, sellers, their geolocations, products, product category names, orders ,order items, payments, and order reviews. Original dataset only includes .csv files

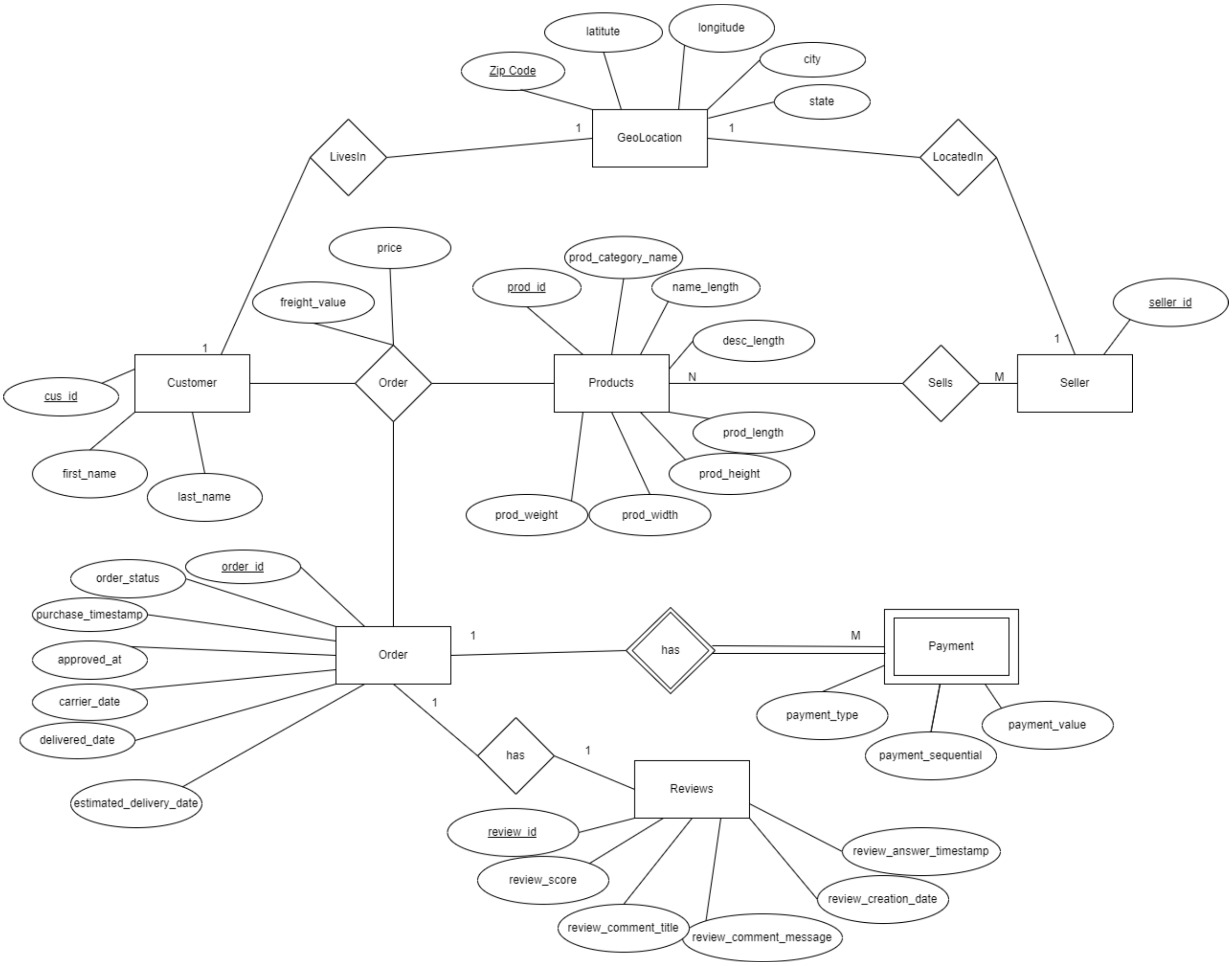
Altogether it had 9 csv files as follows.

- **Customers.csv -** This dataset has information about the customer and its location. Used to identify unique customers.
- **Sellers.csv -** This dataset includes data about the sellers.
- **Geolocation.csv -** This dataset has information Brazilian zip codes and its latitude and longitude coordinates
- **Products.csv -** This dataset includes data about the products sold by sellers
- **Product_category_name_translation.csv -** Translates the product category name to english.
- **Orders.csv** – This dataset includes data about orders.
- **Order_items.csv -** This dataset includes data about the items purchased within each order.
- **Order_payments.csv -** This dataset includes data about the orders payment options.
- **Order_reviews.csv -** This dataset includes data about the reviews made by the customers.

Here we have converted some of the csv files data into a Database source.

The dataset obtained was customized. It includes 5 tables in **Database** format within the '**E_CommerceSourceDB**' database. And the remaining 4 files in '**.csv**' format.

ER Diagram



The above diagram shows the connection between the entities in the dataset.

Assumptions:

1. An order might have multiple items.
2. Each item might be fulfilled by a distinct seller.
3. One Seller Lives in only one location
4. One Customer lives in one location.
5. One order has one review
6. One customer can have many products in many orders

Step 02:- Preparation of Data Sources

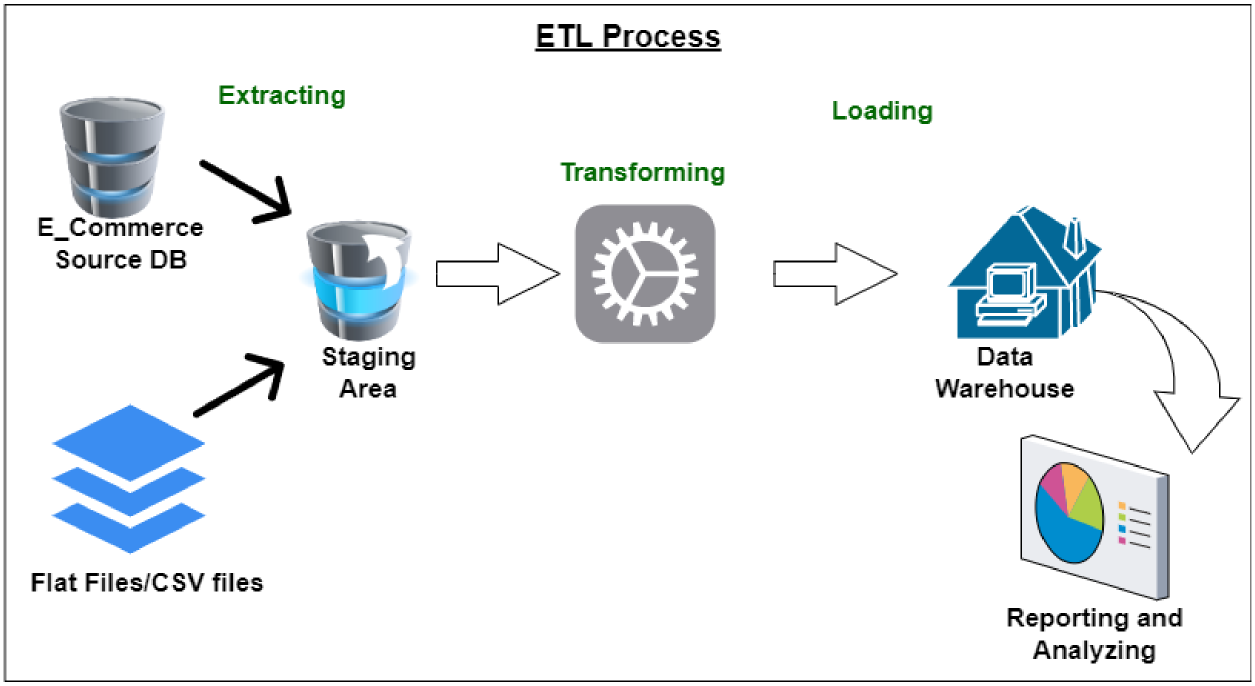
All the data sources provided in the web in .csv format. Since we need at least 2 types of data sources, here some of the .csv file data was imported to a database and some minor changes were done(some columns were removed/added in the process) to the original dataset.

Final Source Dataset was in the below formats before transforming into the Data Warehouse.

Two main sources formats listed below:

- **SQL DataBase → E_commerceSourceDB**
    - **Customer Table**
    - **Geolocation Table**
    - **Product Table**
    - **ProductCategoryNameTranslate Table**
    - **Seller Table**
- **CSV files →**
    - **orders.csv**
    - **order_items.csv**
    - **order_payments.csv**
    - **order_reviews.csv**

Step 03:-Solution Architecture



Source Database:-E_CommerceSourceDB

Flat Files/CSV files :- orders.csv, order_items.csv, order_payments.csv, order_reviews.csv

Step 04:-Data Warehouse Design & Development

Design

Dimensional Tables

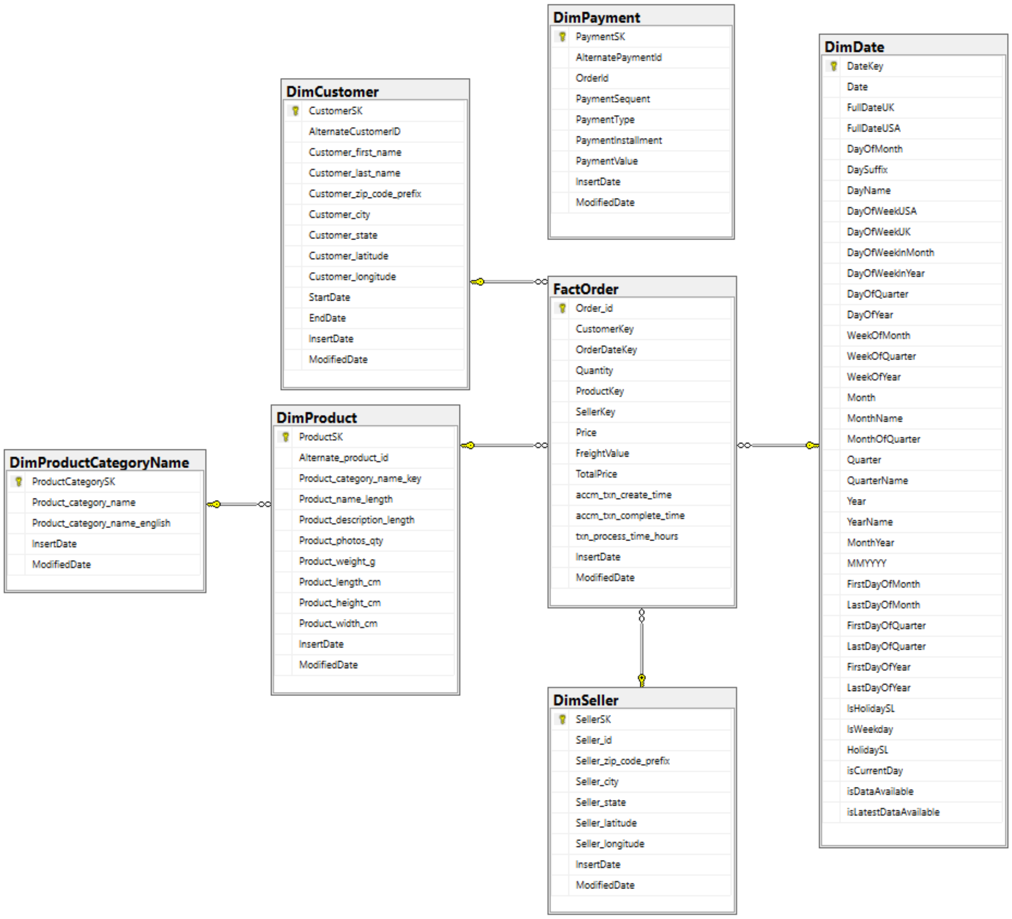
- **DimProductCategoryName -** Includes Product SubCategory names and their translation data.
- **DimProduct** – Includes Product data.
- **DimSeller** – Includes Sellers data.
- **DimCustomer-** Includes customer data.
- **DimDate** – Includes Dates and Date key.
- **DimPayment** – Includes Payment information.

Fact Table

- **FactOrder -** Is a combination of factual data such as Surrogate keys of other dimensions for reference and the measurable data about the orders that happened in this Ecommerce company.

As mentioned above ; there are 6 Dimensional tables including the Date Dimension and the Slowly Changing Dimension which is the **DimCustomer** table. And one fact table called **FactOrder** along with them.

The Data Warehouse model is Snowflake.



- ➢ Hierarchies
  - DimCustomer has hierarchical attributes about Customer GeoLocation.

- ➢ Calculations
  - Total Price is calculated in the FactOrder table.
    TotalPrice = quantity * price

- Assumptions
  - Transaction table used for creating Fact Table
  - Transaction per Customer considered as the grain.

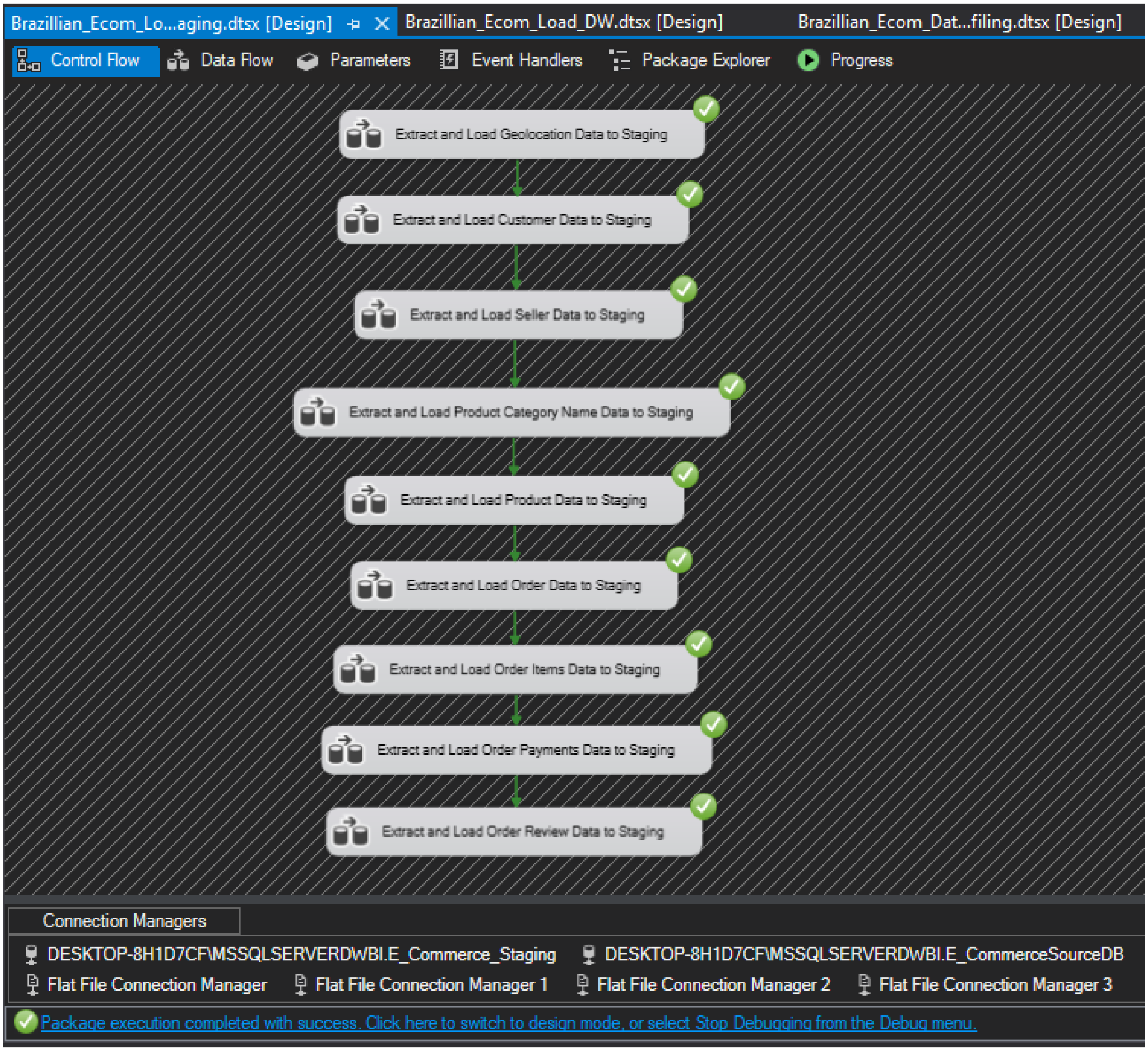
- Slowly Changing Dimensions
  - DimCustomer was considered as the Slowly changing Dimension

Fixed Attributes	**Historical Attributes**
Customer_first_name	Customer_latitude
Customer_last_name	Customer_longitude
	Customer_State
	Customer_city
	Customer_zip_code

Step 05:-ETL Development

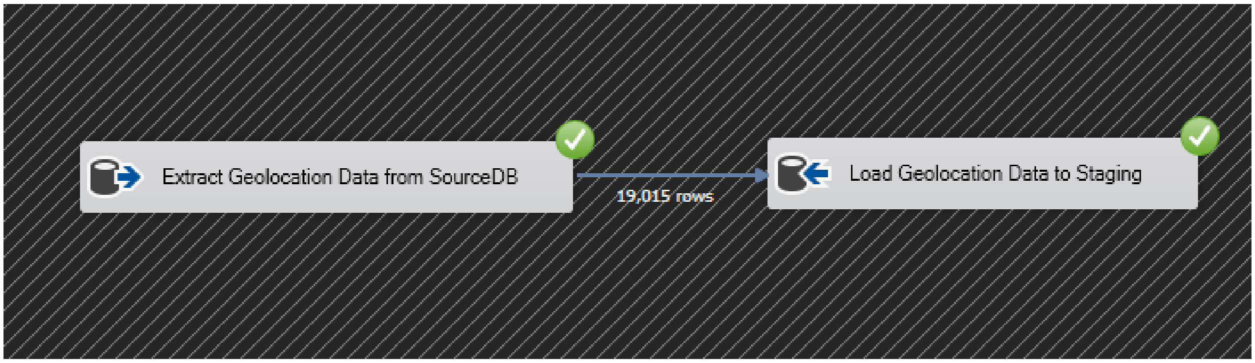
I) Data Extraction and Staging : Initially in the SSIS project , I have created the Staging Database connecting to Flat File sources and the OLE DB Sources.

Control Flow

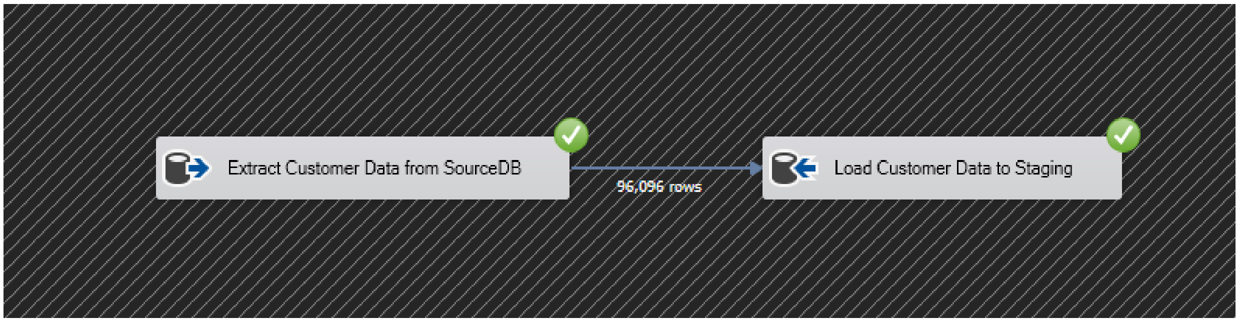

Brazillian_Ecom_Lo...aging.dtsx [Design]
Brazillian_Ecom_Load_DW.dtsx [Design]
Brazillian_Ecom_Dat...filing.dtsx [Design]
Control Flow
Data Flow
Parameters
Event Handlers
Package Explorer
Progress
Extract and Load Geolocation Data to Staging
Extract and Load Customer Data to Staging
Extract and Load Seller Data to Staging
Extract and Load Product Category Name Data to Staging
Extract and Load Product Data to Staging
Extract and Load Order Data to Staging
Extract and Load Order Items Data to Staging
Extract and Load Order Payments Data to Staging
Extract and Load Order Review Data to Staging
Connection Managers
DESKTOP-8H1D7CF\MSSQLSERVERDWBI.E_Commerce_Staging
DESKTOP-8H1D7CF\MSSQLSERVERDWBI.E_CommerceSourceDB
Flat File Connection Manager
Flat File Connection Manager 1
Flat File Connection Manager 2
Flat File Connection Manager 3

✓ Package execution completed with success. Click here to switch to design mode, or select Stop Debugging from the Debug menu.

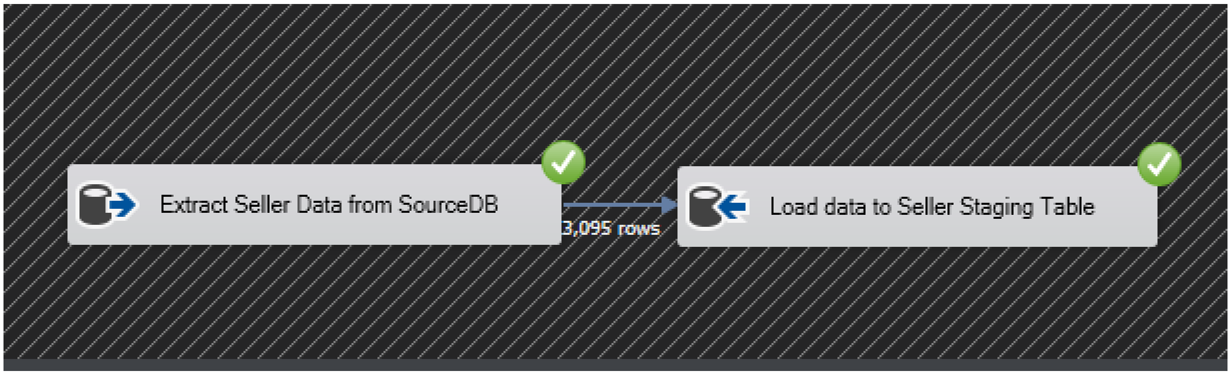
Staging Geolocation Data


Extract Geolocation Data from SourceDB
19,015 rows
Load Geolocation Data to Staging

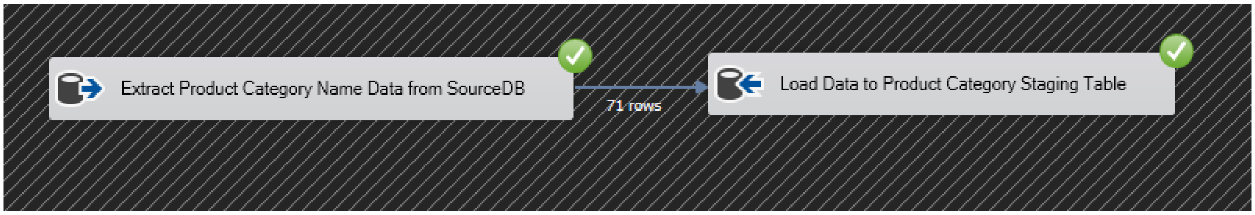
Staging Customer Data


Extract Customer Data from SourceDB
96,096 rows
Load Customer Data to Staging

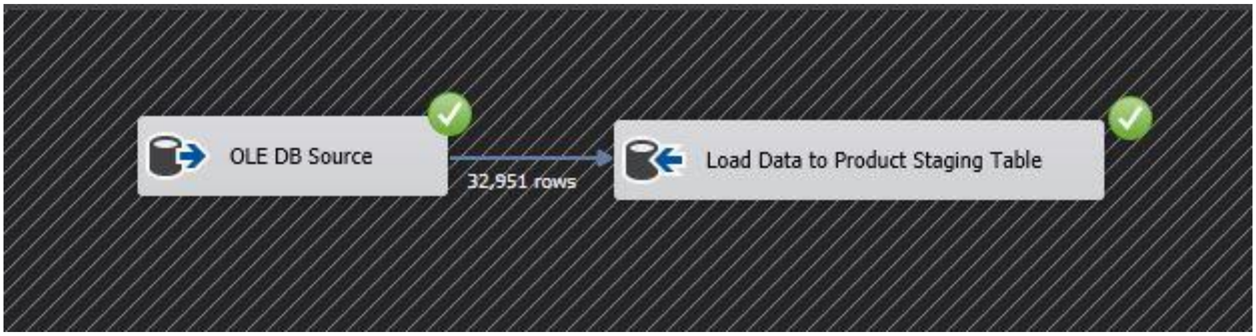
Staging Seller Data


Extract Seller Data from SourceDB
3,095 rows
Load data to Seller Staging Table

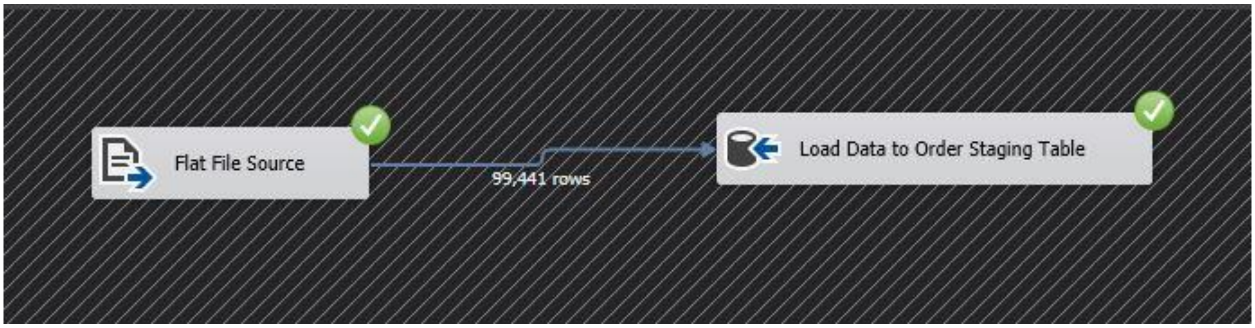
Staging Product Category Name Data


Extract Product Category Name Data from SourceDB
71 rows
Load Data to Product Category Staging Table

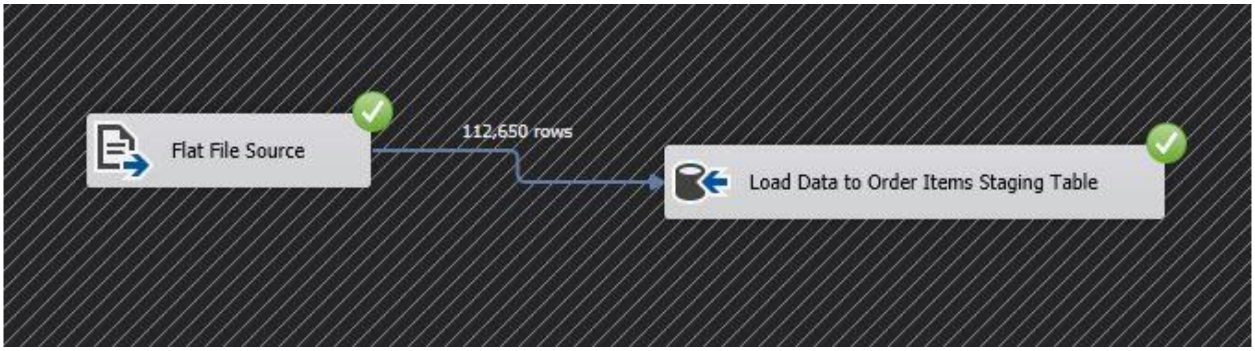
Staging Product Data



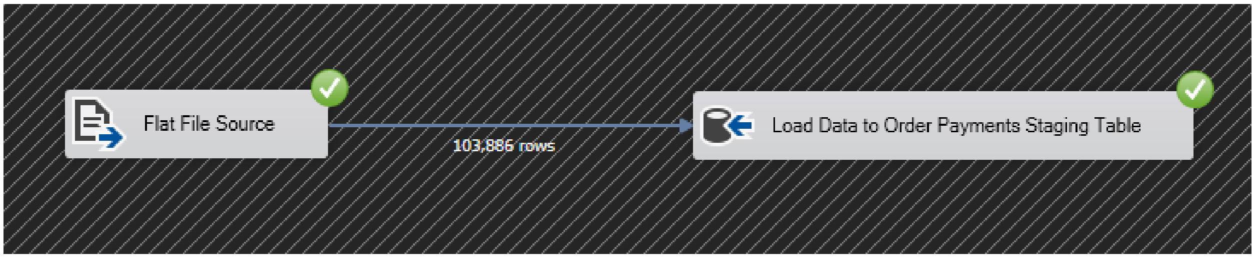
Staging Order Data



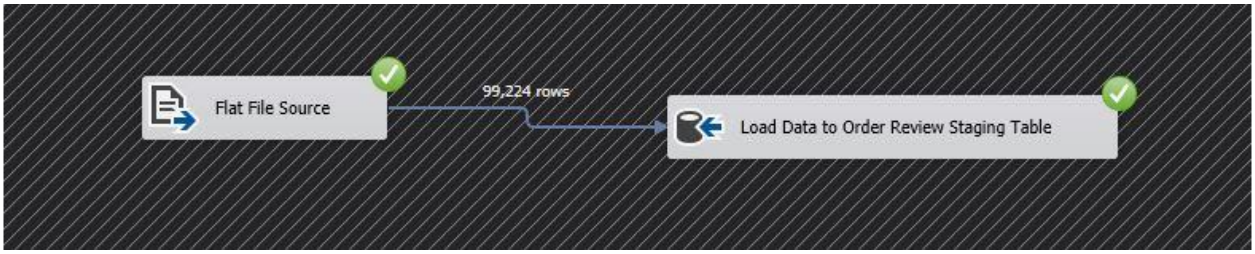
Staging Order Items Data



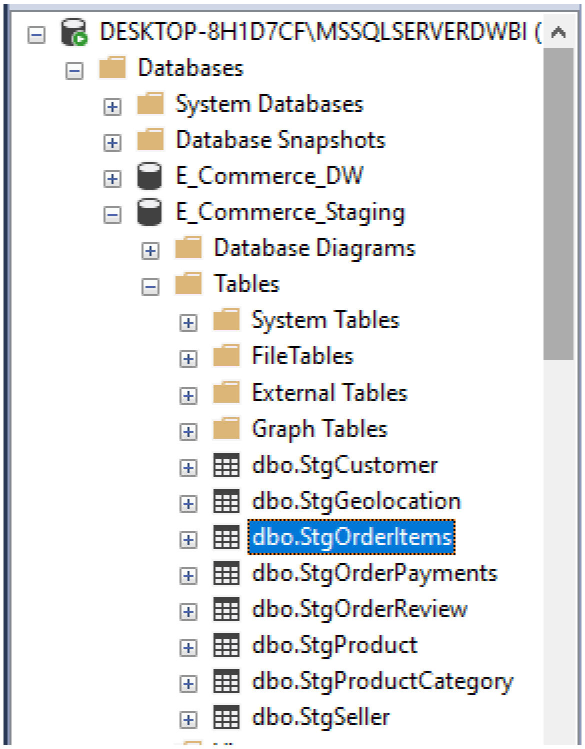
Staging Payments Data



Staging Order Review Data

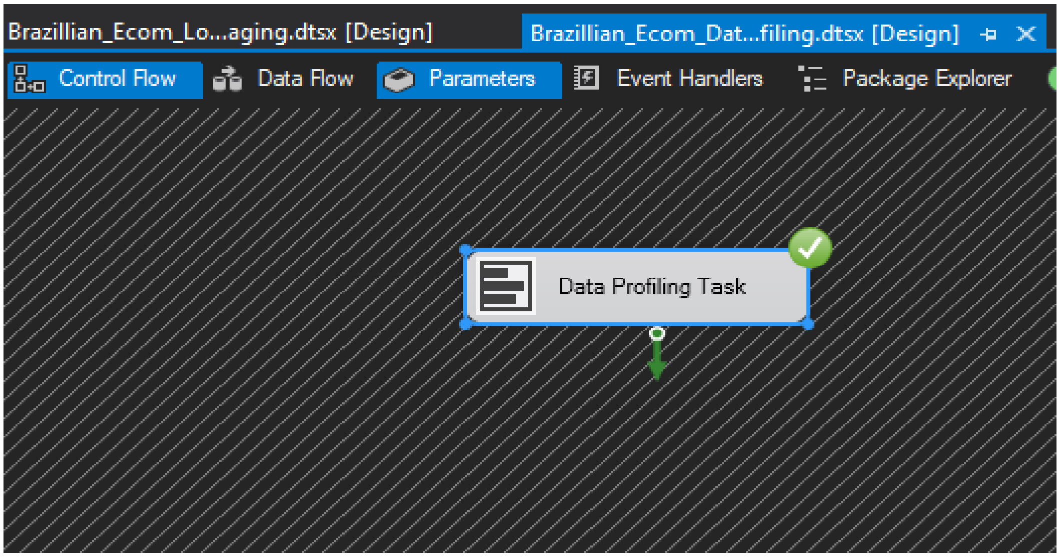


Staging Tables created and values inserted



II) Data Profiling

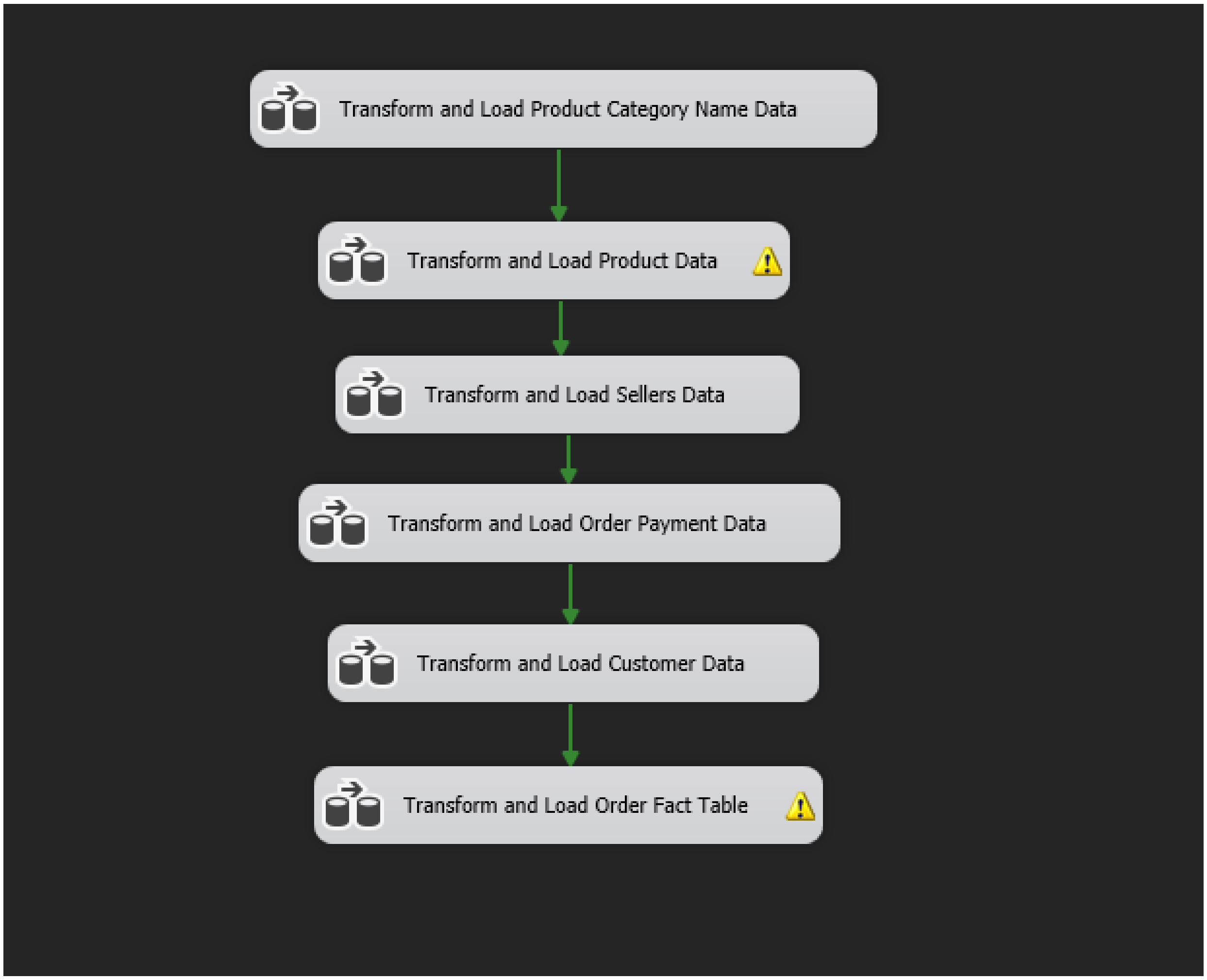
Data Profiling gives some statistics about data in our tables. Now since we have data in our Staging tables, we can run a Data Profiling task to get a better sense of data that are available in source systems. And determine what type of transformations need to be performed on the data.


Brazillian_Ecom_Lo...aging.dtsx [Design]
Brazillian_Ecom_Dat...filing.dtsx [Design]
Control Flow
Data Flow
Parameters
Event Handlers
Package Explorer
Data Profiling Task

III) Data Transformation and Loading :

Data Transformation is developed according to the dimensional modelling.

Dimension tables are loaded with data from relevant staging tables in this step.


Transform and Load Product Category Name Data
Transform and Load Product Data
Transform and Load Sellers Data
Transform and Load Order Payment Data
Transform and Load Customer Data
Transform and Load Order Fact Table

(*The Control flow was executed successfully.I couldn’t take the screenshot while the process was running.I had to take it afterwards.)

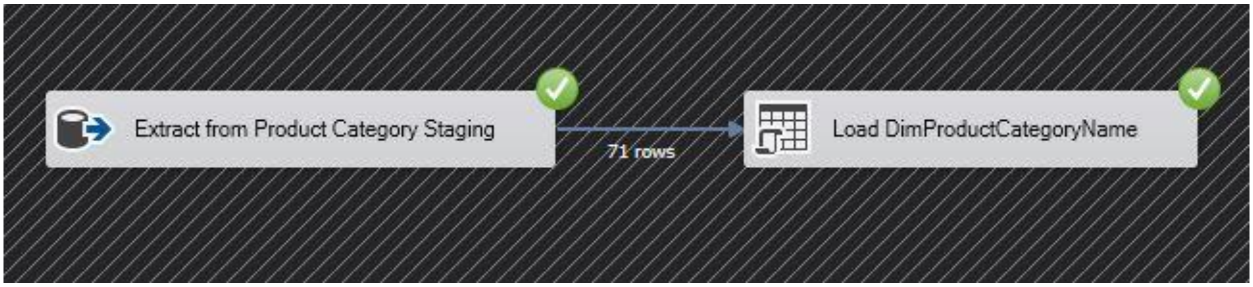
In this step, the data in the Staging area(E_Commerce_Staging Database) is loaded to the dimension and Fact Tables in the E_Commerce_DW.

a)Transform and Load Product Category Name Data

Procedure used to identify whether to insert or update the data in DimProductCategory Table

```
CREATE PROCEDURE dbo.UpdateDimProductCategoryName
@ProductCategoryName nvarchar(50),
@ProductCategoryEnglish nvarchar(50)
AS BEGIN
if not exists (select ProductCategorySK
from dbo.DimProductCategoryName
where Product_category_name = @ProductCategoryName) BEGIN
insert into dbo.DimProductCategoryName
(Product_category_name, Product_category_name_english, InsertDate, ModifiedDate)
values
(@ProductCategoryName, @ProductCategoryEnglish, GETDATE(), GETDATE()) END;
if exists (select ProductCategorySK
from dbo.DimProductCategoryName
where Product_category_name = @ProductCategoryName) BEGIN
update dbo.DimProductCategoryName
set Product_category_name_english = @ProductCategoryEnglish,
ModifiedDate = GETDATE()
where Product_category_name = @ProductCategoryName END;

END;
```

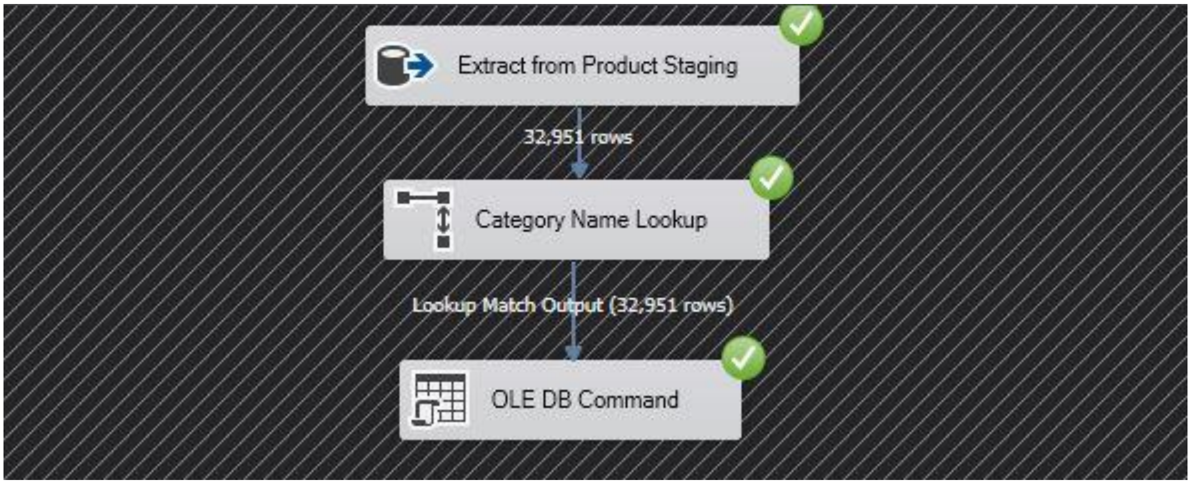

Extract from Product Category Staging
71 rows
Load DimProductCategoryName

b) Transform and Load Product Data

Procedure used to identify whether to insert or update the data in DimProduct Table

```
CREATE PROCEDURE dbo.UpdateDimProduct
@AlternateId nvarchar(50),
@ProductCategoryNameKey int,
@ProductNameLength int,
@ProductDescriptionLength int,
@ProductPhotosQty int,
@ProductWeightG int,
@ProductLengthCm int,
@ProductHeightCm int,
@ProductWidthCm int
AS BEGIN
if not exists (select ProductSK
from dbo.DimProduct
where Alternate_product_id = @AlternateId) BEGIN
insert into dbo.DimProduct
(Alternate_product_id, Product_category_name_key, Product_name_length,
Product_description_length,
Product_photos_qty, Product_weight_g, Product_length_cm, Product_height_cm,
Product_width_cm, InsertDate, ModifiedDate)
values
(@AlternateId, @ProductCategoryNameKey, @ProductNameLength , @ProductDescriptionLength,
@ProductPhotosQty,
@ProductWeightG, @ProductLengthCm, @ProductHeightCm, @ProductWidthCm,  GETDATE(),
GETDATE()) END;
if exists (select ProductSK
from dbo.DimProduct
where Alternate_product_id = @AlternateId) BEGIN
update dbo.DimProduct
set ModifiedDate = GETDATE()
where Alternate_product_id = @AlternateId END;

END;
```


Extract from Product Staging
32,951 rows
Category Name Lookup
Lookup Match Output (32,951 rows)
OLE DB Command

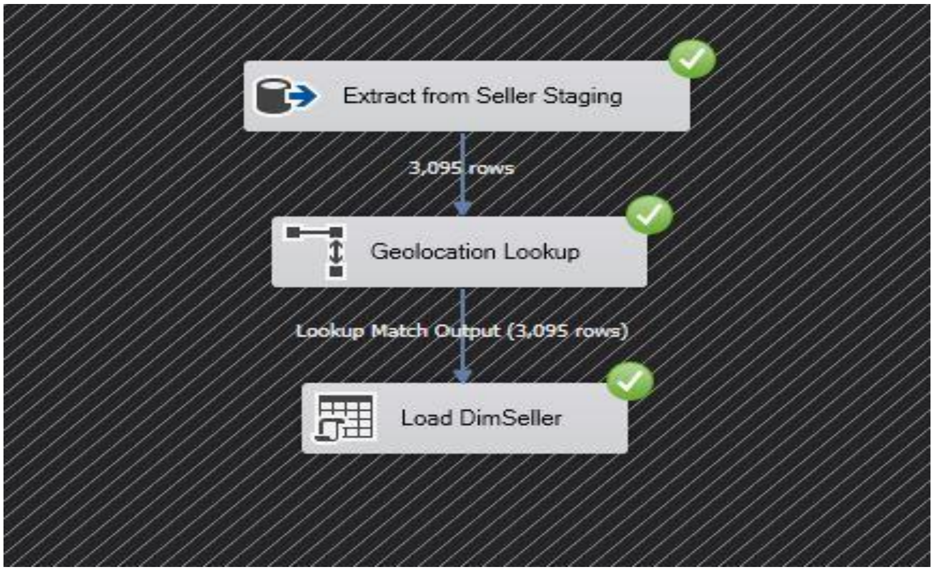
c) Transform and Load Sellers Data

Procedure used to identify whether to insert or update the data in DimSeller Table

```sql
CREATE PROCEDURE dbo.UpdateDimSeller
@SellerID nvarchar(50),
@SellerZip nvarchar(50),
@SellerCity nvarchar(50),
@SellerState nvarchar(10),
@Latitude float,
@Longitude float

AS BEGIN
if not exists (select SellerSK
from dbo.DimSeller
where Seller_id = @SellerID) BEGIN
insert into dbo.DimSeller
(Seller_id, Seller_zip_code_prefix, Seller_city, Seller_state, Seller_latitude,
Seller_longitude, InsertDate, ModifiedDate)
values
(@SellerID, @SellerZip , @SellerCity , @SellerState, @Latitude, @Longitude, GETDATE(),
GETDATE()) END;
if exists (select SellerSK
from dbo.DimSeller
where Seller_id = @SellerID) BEGIN
update dbo.DimSeller
set Seller_zip_code_prefix = @SellerZip, Seller_city = @SellerCity, Seller_state =
@SellerState, ModifiedDate = GETDATE()
where Seller_id = @SellerID END;

END;
```

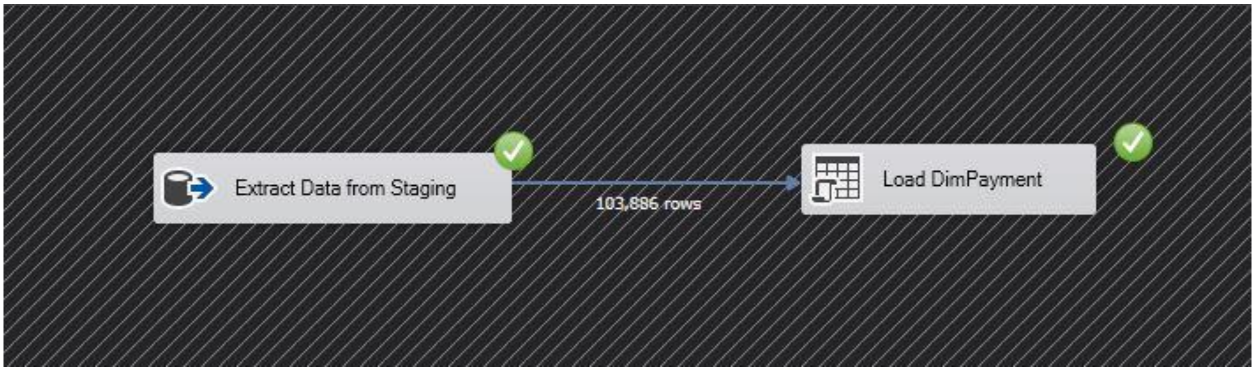


d)Transform and Load Payment Data

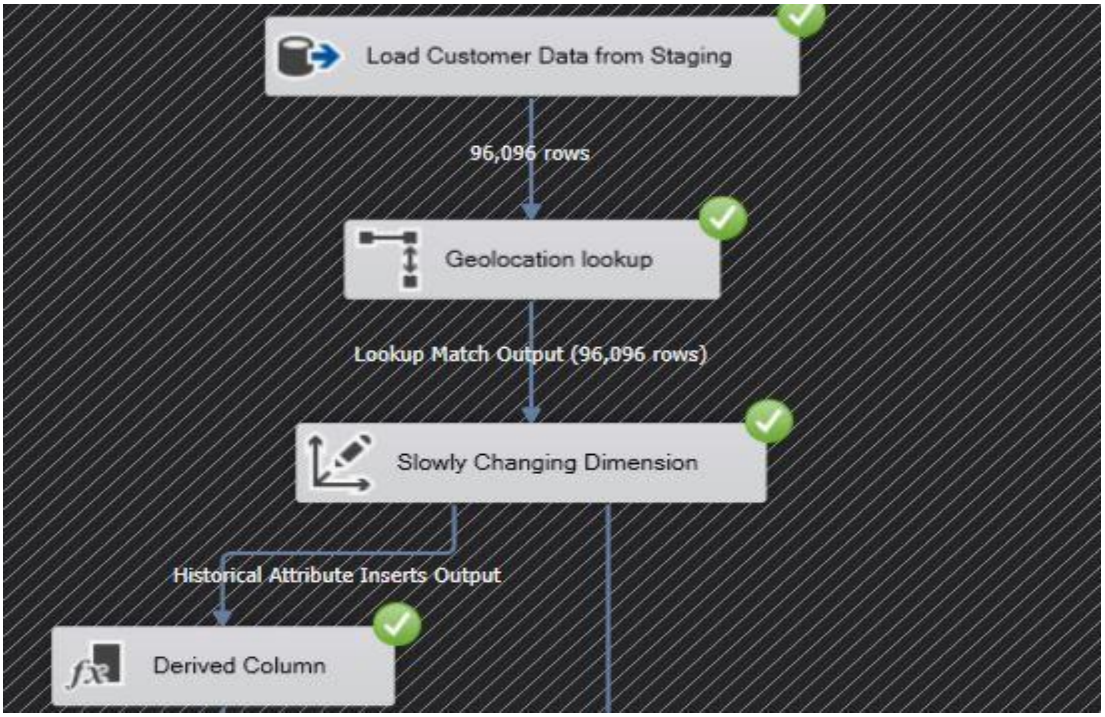
Procedure used to identify whether to insert or update the data in DimPayment Table

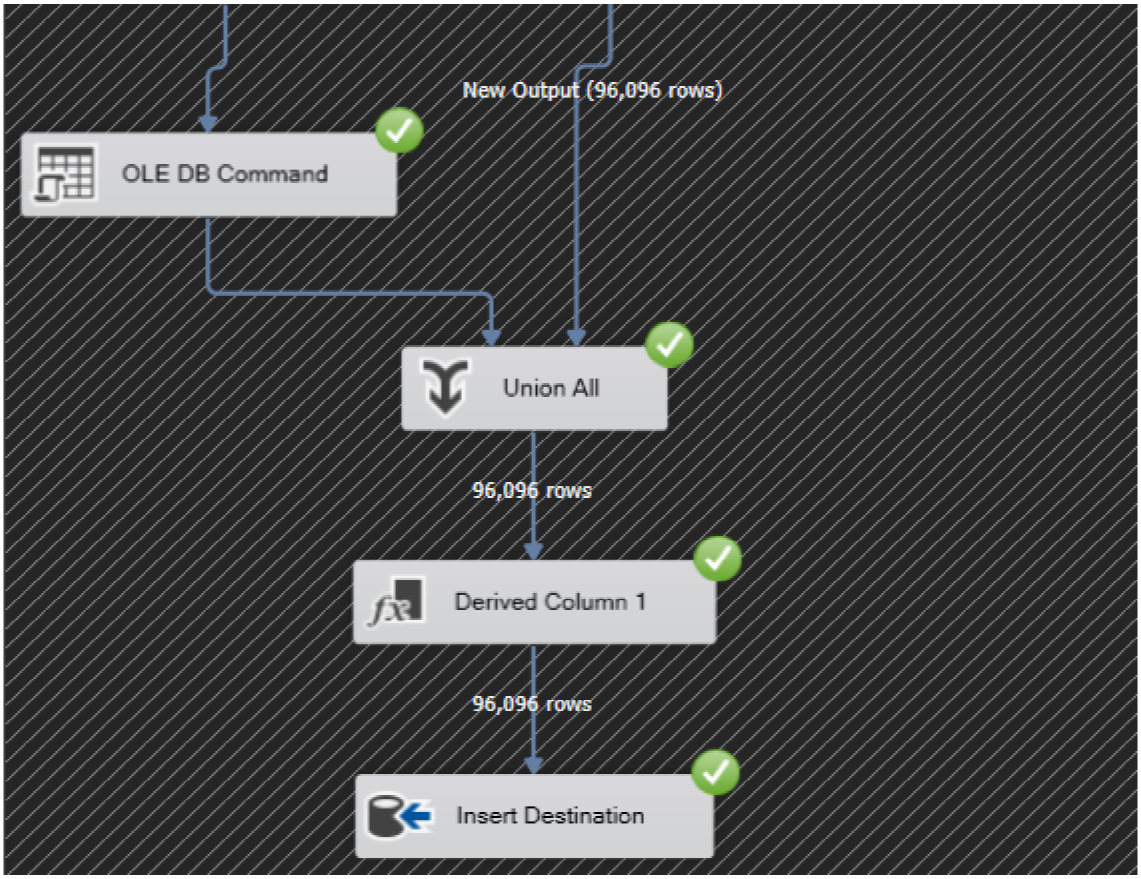
```
CREATE PROCEDURE dbo.UpdateDimOrderPayment
@AlternateId int,
@OrderID nvarchar(50),
@PaymentSeq int,
@PaymentType nvarchar(50),
@PaymentIns int,
@PaymentVal float
AS BEGIN
if not exists (select PaymentSK
from dbo.DimPayment
where AlternatePaymentId = @AlternateId) BEGIN
insert into dbo.DimPayment
(AlternatePaymentId, OrderId, PaymentSequent, PaymentType, PaymentInstallment,
PaymentValue, InsertDate, ModifiedDate)
values
(@AlternateId, @OrderID, @PaymentSeq , @PaymentType , @PaymentIns, @PaymentVal,
GETDATE(), GETDATE()) END;
if exists (select PaymentSK
from dbo.DimPayment
where AlternatePaymentId = @AlternateId) BEGIN
update dbo.DimPayment
set PaymentSequent = @PaymentSeq, PaymentType = @PaymentType, PaymentInstallment =
@PaymentIns, PaymentValue = @PaymentVal, ModifiedDate = GETDATE()
where AlternatePaymentId = @AlternateId END;

END;
```

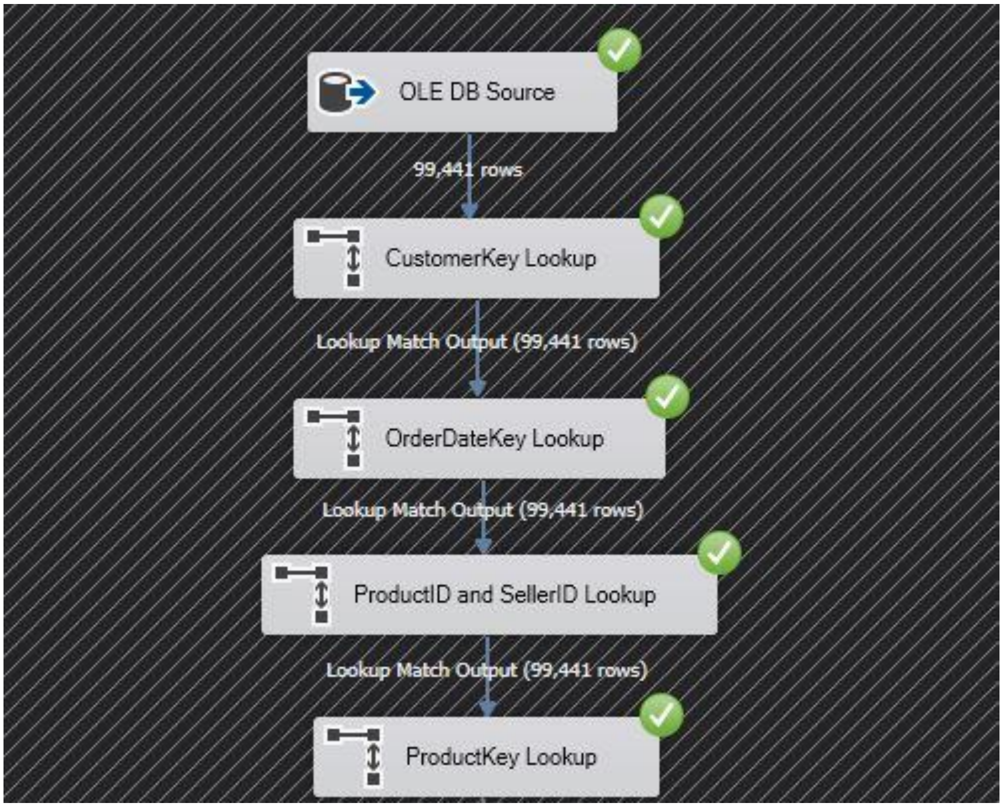


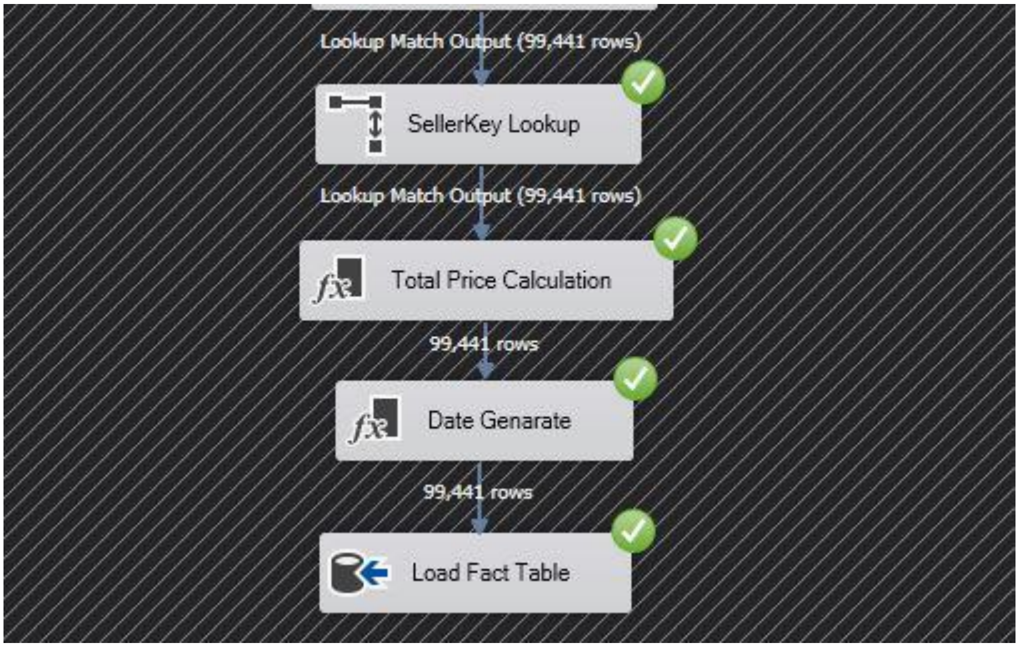
e) Transform and Load Customer Data(Slowly Changing Dimension)


Load Customer Data from Staging
96,096 rows
Geolocation lookup
Lookup Match Output (96,096 rows)
Slowly Changing Dimension
Historical Attribute Inserts Output
Derived Column



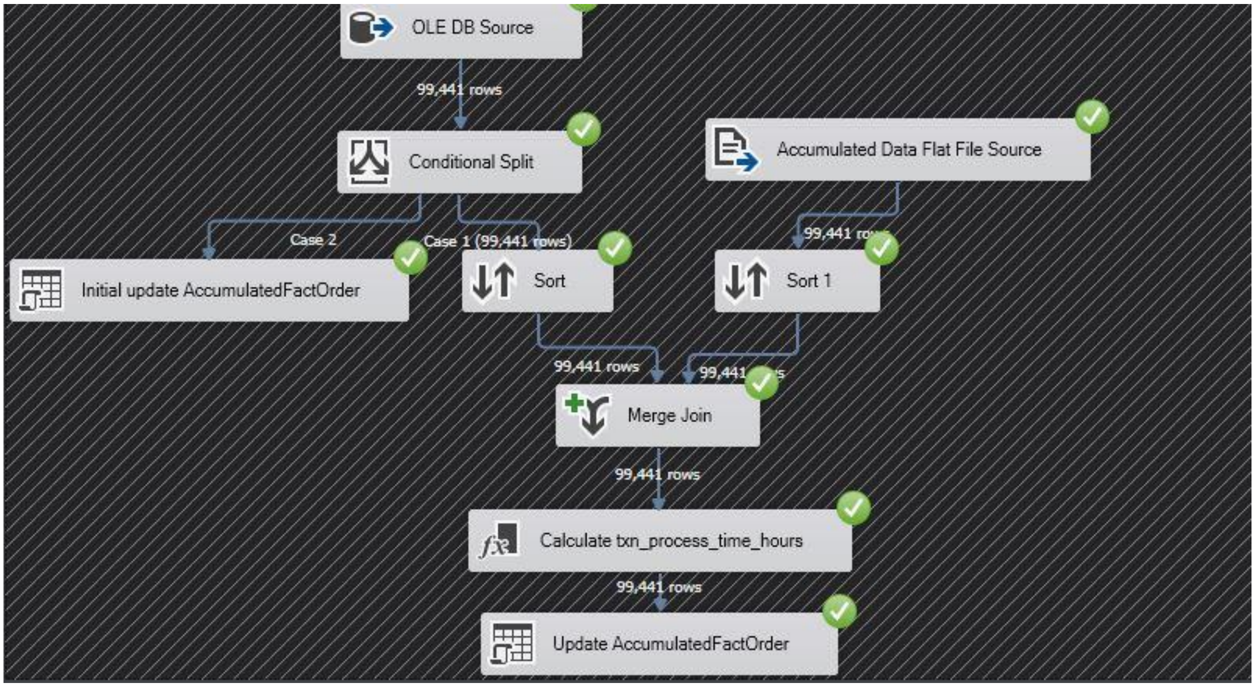
f) Load Order Fact Table

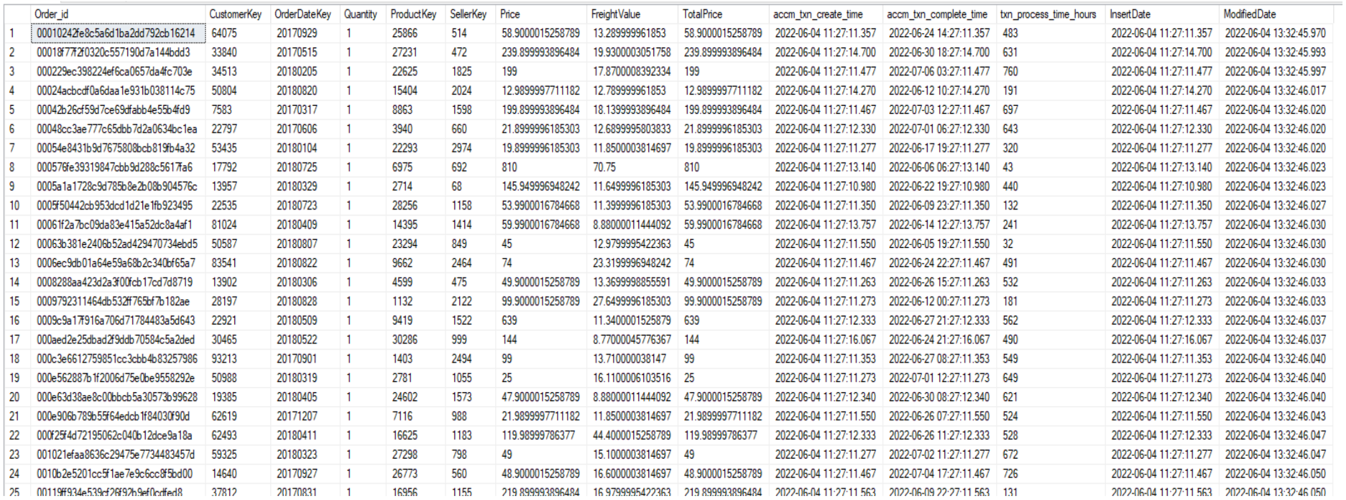

OLE DB Source
99,441 rows
CustomerKey Lookup
Lookup Match Output (99,441 rows)
OrderDateKey Lookup
Lookup Match Output (99,441 rows)
ProductID and SellerID Lookup
Lookup Match Output (99,441 rows)
ProductKey Lookup



Step 06:-ETL Development - Accumulating Fact Tables

In order to create the Accumulated Fact table I created a new SSIS Package called ‘Brazillian_Ecom_AccumulatedFact.dtsx’ and updated the accm_txn_complete_time and txn_process_time_hours as follows.





	Order_id	CustomerKey	OrderDateKey	Quantity	ProductKey	SellerKey	Price	FreightValue	TotalPrice	accm_txn_create_time	accm_txn_complete_time	txn_process_time_hours	InsertDate	ModifiedDate
1	00010242fe8c5a6d1ba2dd792cb16214	64075	20170929	1	25866	514	58.9000015258789	13.289999961853	58.9000015258789	2022-06-04 11:27:11.357	2022-06-24 14:27:11.357	483	2022-06-04 11:27:11.357	2022-06-04 13:32:45.970
2	00018f77f2f0320c557190d7a144bdd3	33840	20170515	1	27231	472	239.899993896484	19.9300003051758	239.899993896484	2022-06-04 11:27:14.700	2022-06-30 18:27:14.700	631	2022-06-04 11:27:14.700	2022-06-04 13:32:45.993
3	000229ec398224ef6ca0657da4fc703e	34513	20180205	1	22625	1825	199	17.8700008392334	199	2022-06-04 11:27:11.477	2022-07-06 03:27:11.477	760	2022-06-04 11:27:11.477	2022-06-04 13:32:45.997
4	00024acbcdf0a6daa1e931b038114c75	50804	20180820	1	15404	2024	12.9899997711182	12.789999961853	12.9899997711182	2022-06-04 11:27:14.270	2022-06-12 10:27:14.270	191	2022-06-04 11:27:14.270	2022-06-04 13:32:46.017
5	00042b26cf59d7ce69dfabb4e55b4fd9	7583	20170317	1	8863	1598	199.899993896484	18.1399993896484	199.899993896484	2022-06-04 11:27:11.467	2022-07-03 12:27:11.467	697	2022-06-04 11:27:11.467	2022-06-04 13:32:46.020
6	00048cc3ae777c65dbb7d2a0634bc1ea	22797	20170606	1	3940	660	21.8999996185303	12.6899995803833	21.8999996185303	2022-06-04 11:27:12.330	2022-07-01 06:27:12.330	643	2022-06-04 11:27:12.330	2022-06-04 13:32:46.020
7	00054e8431b9d7675808bcb819fb4a32	53435	20180104	1	22293	2974	19.8999996185303	11.8500003814697	19.8999996185303	2022-06-04 11:27:11.277	2022-06-17 19:27:11.277	320	2022-06-04 11:27:11.277	2022-06-04 13:32:46.020
8	000576fe39319847cbb9d288c5617fa6	17792	20180725	1	6975	692	810	70.75	810	2022-06-04 11:27:13.140	2022-06-06 06:27:13.140	43	2022-06-04 11:27:13.140	2022-06-04 13:32:46.023
9	0005a1a1728c9d785b8e2b08b904576c	13957	20180329	1	2714	68	145.949996948242	11.6499996185303	145.949996948242	2022-06-04 11:27:10.980	2022-06-22 19:27:10.980	440	2022-06-04 11:27:10.980	2022-06-04 13:32:46.023
10	0005f50442cb953dcd1d21e1fb923495	22535	20180723	1	28256	1158	53.9900016784668	11.3999996185303	53.9900016784668	2022-06-04 11:27:11.350	2022-06-09 23:27:11.350	132	2022-06-04 11:27:11.350	2022-06-04 13:32:46.027
11	00061f2a7bc09da83e415a52dc8a4af1	81024	20180409	1	14395	1414	59.9900016784668	8.88000011444092	59.9900016784668	2022-06-04 11:27:13.757	2022-06-14 12:27:13.757	241	2022-06-04 11:27:13.757	2022-06-04 13:32:46.030
12	00063b381e2406b52ad429470734ebd5	50587	20180807	1	23294	849	45	12.9799995422363	45	2022-06-04 11:27:11.550	2022-06-05 19:27:11.550	32	2022-06-04 11:27:11.550	2022-06-04 13:32:46.030
13	0006ec9db01a64e59a68b2c340bf65a7	83541	20180822	1	9662	2464	74	23.3199996948242	74	2022-06-04 11:27:11.467	2022-06-24 22:27:11.467	491	2022-06-04 11:27:11.467	2022-06-04 13:32:46.030
14	0008288aa423d2a3f00fcb17cd7d8719	13902	20180306	1	4599	475	49.9000015258789	13.3699998855591	49.9000015258789	2022-06-04 11:27:11.263	2022-06-26 15:27:11.263	532	2022-06-04 11:27:11.263	2022-06-04 13:32:46.033
15	0009792311464db532ff765bf7b182ae	28197	20180828	1	1132	2122	99.9000015258789	27.6499996185303	99.9000015258789	2022-06-04 11:27:11.273	2022-06-12 00:27:11.273	181	2022-06-04 11:27:11.273	2022-06-04 13:32:46.033
16	0009c9a17f916a706d71784483a5d643	22921	20180509	1	9419	1522	639	11.3400001525879	639	2022-06-04 11:27:12.333	2022-06-27 21:27:12.333	562	2022-06-04 11:27:12.333	2022-06-04 13:32:46.037
17	000aed2e25dbad2f9ddb70584c5a2ded	30465	20180522	1	30286	999	144	8.77000045776367	144	2022-06-04 11:27:16.067	2022-06-24 21:27:16.067	490	2022-06-04 11:27:16.067	2022-06-04 13:32:46.037
18	000c3e6612759851cc3cbb4b83257986	93213	20170901	1	1403	2494	99	13.71000038147	99	2022-06-04 11:27:11.353	2022-06-27 08:27:11.353	549	2022-06-04 11:27:11.353	2022-06-04 13:32:46.040
19	000e562887b1f2006d75e0be9558292e	50988	20180319	1	2781	1055	25	16.1100006103516	25	2022-06-04 11:27:11.273	2022-07-01 12:27:11.273	649	2022-06-04 11:27:11.273	2022-06-04 13:32:46.040
20	000e63d38ae8c00bbcb5a30573b99628	19385	20180405	1	24602	1573	47.9000015258789	8.88000011444092	47.9000015258789	2022-06-04 11:27:12.340	2022-06-30 08:27:12.340	621	2022-06-04 11:27:12.340	2022-06-04 13:32:46.040
21	000e906b789b55f64edcb1f84030f90d	62619	20171207	1	7116	988	21.9899997711182	11.8500003814697	21.9899997711182	2022-06-04 11:27:11.550	2022-06-26 07:27:11.550	524	2022-06-04 11:27:11.550	2022-06-04 13:32:46.043
22	000f25f4d72195062c040b12dce9a18a	62493	20180411	1	16625	1183	119.98999786377	44.4000015258789	119.98999786377	2022-06-04 11:27:12.333	2022-06-26 11:27:12.333	528	2022-06-04 11:27:12.333	2022-06-04 13:32:46.047
23	001021efaa8636c29475e7734483457d	59325	20180323	1	27298	798	49	15.1000003814697	49	2022-06-04 11:27:11.277	2022-07-02 11:27:11.277	672	2022-06-04 11:27:11.277	2022-06-04 13:32:46.047
24	0010b2e5201cc5f1ae7e9c6cc8f5bd00	14640	20170927	1	26773	560	48.9000015258789	16.6000003814697	48.9000015258789	2022-06-04 11:27:11.467	2022-07-04 17:27:11.467	726	2022-06-04 11:27:11.467	2022-06-04 13:32:46.050
25	00119ff934e539cf26f92b9ef0cdfed8	37812	20170831	1	16956	1155	219.899993896484	16.9799995422363	219.899993896484	2022-06-04 11:27:11.563	2022-06-09 22:27:11.563	131	2022-06-04 11:27:11.563	2022-06-04 13:32:46.050

**** END ****