Data Warehouse Project

Adventurework Company

By: Choun Sambat

I. Problem Overview

Adventure Works Cycles is a global manufacturing company that sells its products through a network of resellers and an online web portal.

Currently, the company stores all operational data in a transactional database, AdventureWorks2019, which captures detailed information about products, customers, orders, and sales transactions. While this system effectively supports daily operations, it was not designed for analytical or strategic purposes. As a result, the management team faces significant challenges when trying to answer critical business questions such as:

- Which products are the most profitable over time?
- How are sales trends performing across different regions and channels?
- Which customers contribute most to revenue growth?

Therefore, there is a critical need to develop a departmental data warehouse solution that provides reliable, integrated, and historical data to support business intelligence and strategic analysis.

II. Project Objectives

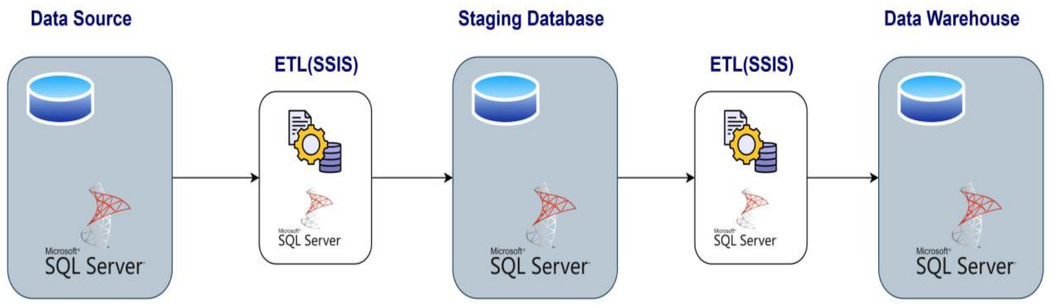
The main objective of this project is to design and implement a Sales Data Warehouse (AdventureWorks_DWH) that enables fast, accurate, and insightful data analysis. The focus is to transform the company's existing transactional data into a structured analytical environment that supports strategic reporting and business intelligence.

III. Project Architecture Design and Workflow

Warehouse Architecture Workflow design

The architecture of the AdventureWorks Data Warehouse is designed to ensure efficient data extraction, transformation, loading, and reporting. It provides a structured flow of data from the transactional database (AdventureWorks2019) to the analytical layer (AdventureWorks_DWH) and finally to the Business Intelligence tools for reporting and analysis.

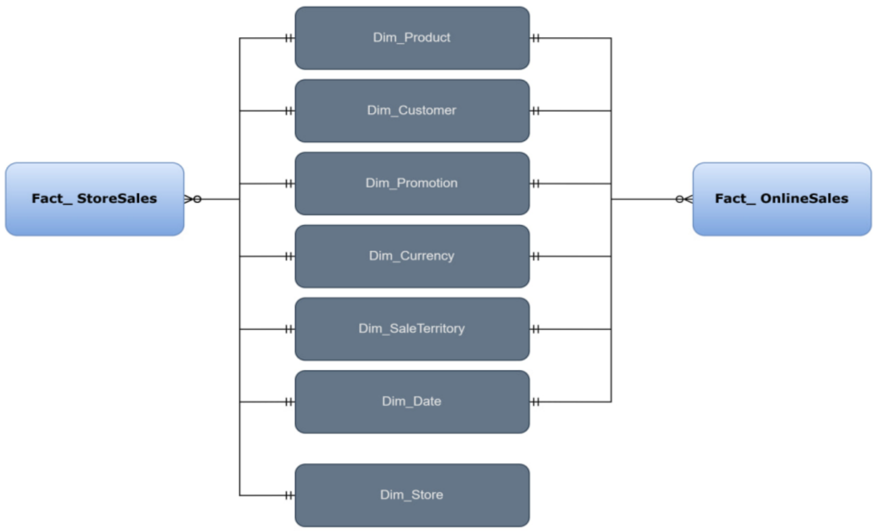
The solution follows a typical three-layer data warehouse architecture which includes Source Contains the AdventureWorks2019 transactional database, stores day-to-day operational data including customers, products, orders, and sales transactions. Data in this layer is optimized for transactional efficiency (OLTP system). Next, Staging Database is a temporary storage area used to extract data from the source system. It Performs data cleansing, transformation, and validation and manage incremental load logic and identifies orphan data. Finally, Data Warehouse which is a centralized analytical database structured in **Galaxy Schema** contains dimension tables (e.g., Dim_Customer, Dim_Product, DimDate, DimSalesTerritory, Dim_Store, Dim_Promotion and Dim_Currency) and fact tables (e.g., FactStoreSales and FactOnlineSales). It stores historical and aggregated data optimized for analytical queries (OLAP system).



DWH_Architecture Workflow

Data Modeling Design Structure

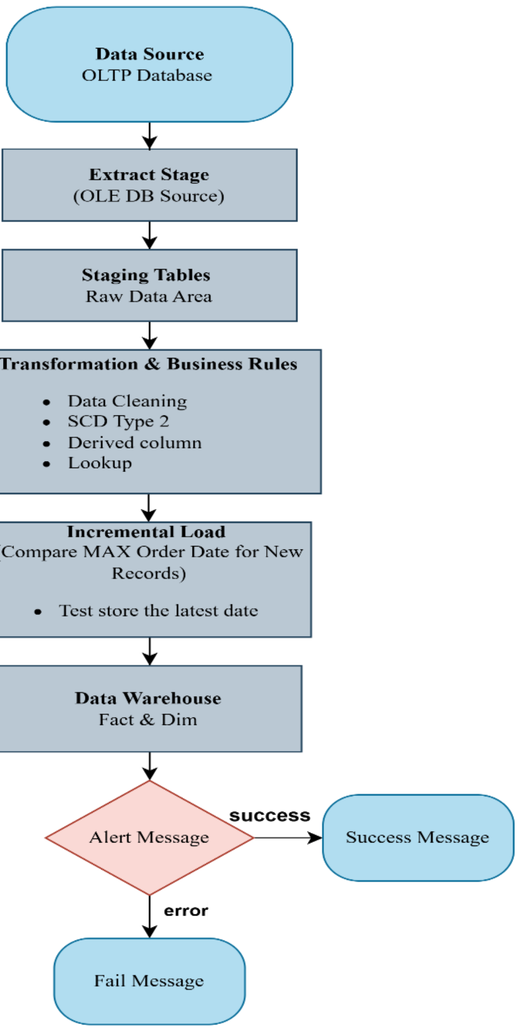
For the AdventureWorks_DWH, the data model is designed using a Galaxy Schema approach, which simplifies reporting and improves performance for BI tools.



Data Modeling Design Structure

ETL Workflow Diagram Structure

ETL Workflow Diagram is talking about how data flows through the ETL pipeline — from extraction to staging, transformation, and final loading into the data warehouse.



ETL Workflow Diagram Structure

a) Extract Stage

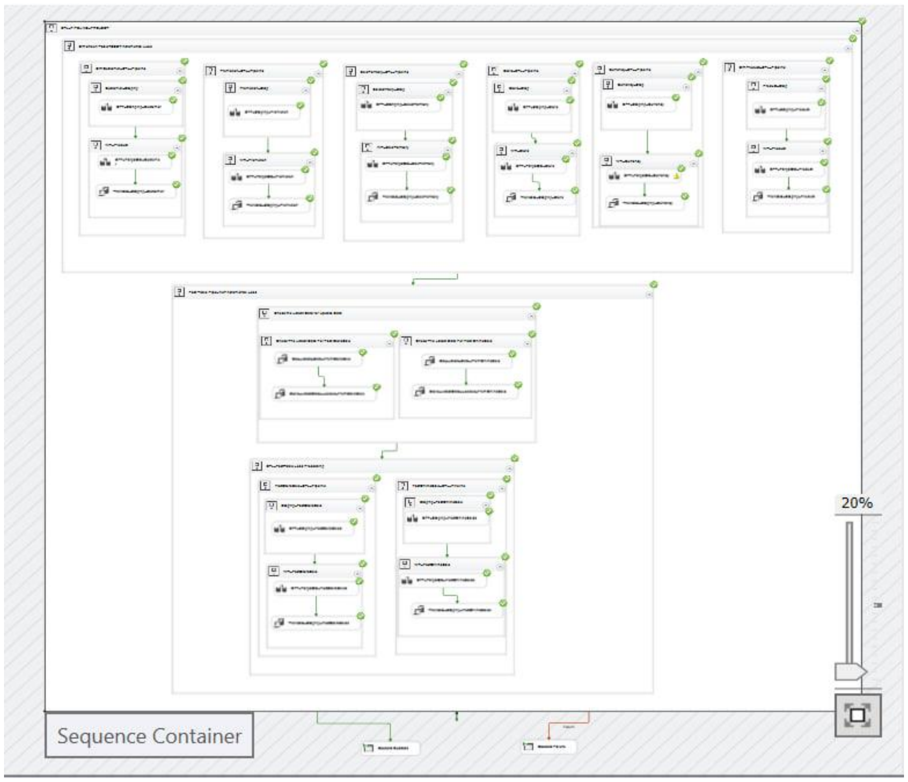
It is the process to collect raw data from various source system OLTP database into a staging area for processing. Data is extract from Adventurework2019 (e.g., tables SalesOrderHeader, SalesOrderDetail, Customer, Product, Currency, ProductCategory, ProductModel, ProductSubcategory, SalesTerritory…) and views like vStoreWithDemographics, vPersonDemographics by use **OLE DB Source** components in SSIS. Moreover, to extract some data that stored in multiple tables I created view in SSMS to join relevant tables.

b) *Transform Stage*

To clean, validate, and apply business rules to the data before loading it into the data warehouse. Transformations performed include data cleaning, Slowly Changing Dimension (SCD Type 2) to track historical changes in dimension tables (insert new version when attribute changes), Lookup Transformation in fact tables to match foreign keys (e.g., CustomerID → CustomerKey from DimCustomer), Derived Column to add new column (Load_date, OrderDateKey, ShipDateKey, DueDateKey) and replace Null values to USD.

c) *Load Stage*

Load the transformed data into the target Data Warehouse tables — dimension tables (DimCustomer, DimProduct, DimDate, DimStore, DimCurrency, DimTerritory, DimPromotion) first, then fact tables (FactStoreSales, FactOnlineSale) using the foreign keys from dimensions. Use **Incremental Load Strategy** — only load new or changed records since the last ETL run, after successful load update TEST_STORE_LAST_UPDATE with the latest order date loaded. Finally, log success or failure send a notification to telegram alert.

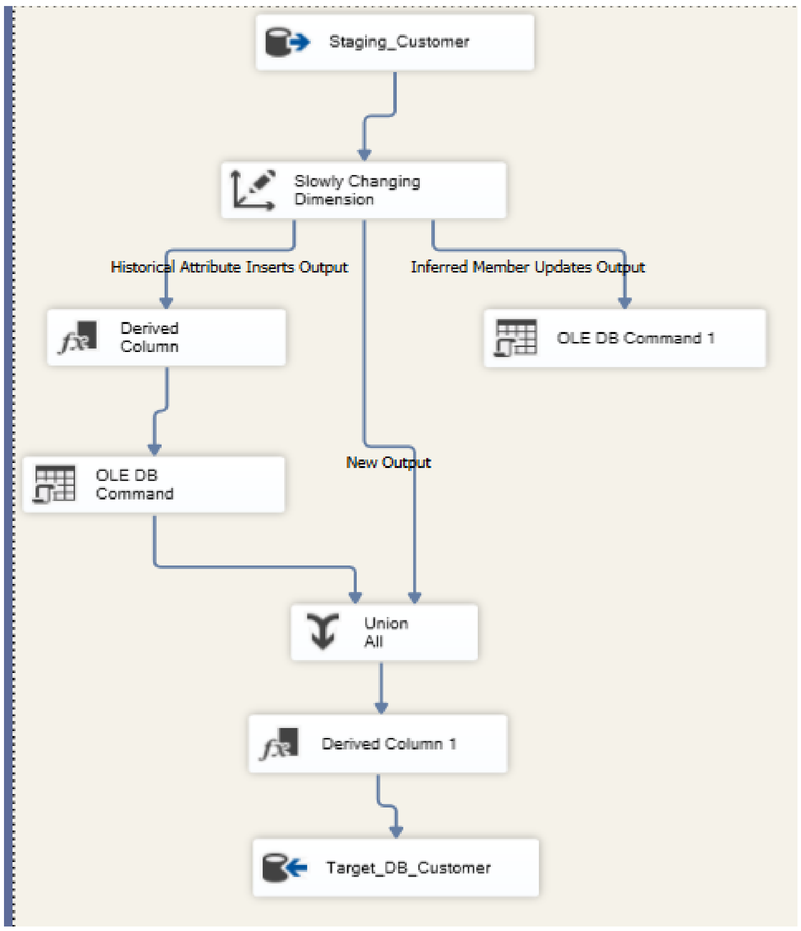


Screenshots of successful package execution

IV. Incremental Load

a. Incremental Load for Dimension Tables

In a Data Warehouse environment, Dimension Tables store descriptive information about business entities such as DimCustomer, DimProduct, DimCurrency, DimStore, Dimdate, DimTerritory and DimPromotion. Over time, the attributes of these entities can change — for example, a customer changes their address or a product's category changes. To maintain historical accuracy and allow time-based analysis, **Slowly Changing Dimensions (SCD Type2)** are used. When an attribute of a record changes, instead of overwriting the old data, it kept the historical data then insert a **new version** and update EndDate of old one. Otherwise, for the new record that not found in Dimension table insert as a new row with StartDate.



Incremental Load for Dimension Table

b. Incremental Load for Fact Table

In Data Warehouse, incremental Load is used to load only new or updated records from the source system into the Fact Tables instead of reloading all data every time. The Incremental Load for Fact Tables in SSIS (or SQL-based ETL) is typically designed in three key steps:

1) **Step 1: Create a Table to Store the Latest Date**

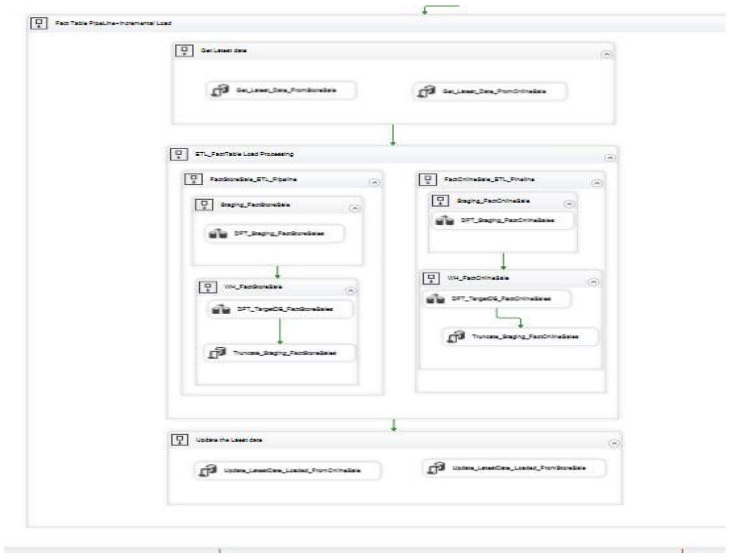
Create a control table in OLTP Source that stores the last date when data was successfully loaded into the Fact Table. In this case we need to create two fact tables for Fact_StoreSales and Fact_OnlineSales.

2) **Step 2: Using Execute SQL Task to Retrieve the Latest Date**

In SSIS, use an **Execute SQL Task** to query the latest date from the control table. For the first run the control table don't have value of latest date yet, so set the logic to add it by default when no value. Then, when extract Data from source use SQL code to compare the date from source and control table by filter only new data (Where OrderDate > Latest_Date.

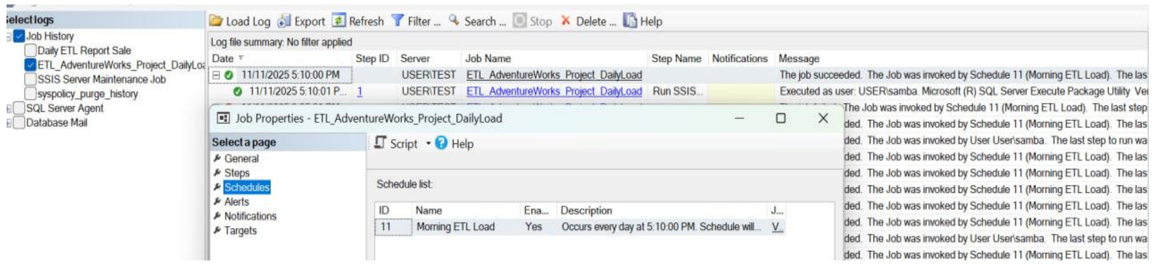
3) **Step 3: Using Execute SQL Task Again to Update the Latest Date**

After data successfully loaded into the Fact Table, use another Execute SQL Task to update the control table with the new latest date from the source data by set the column Latest_Date = (SELECT MAX(OrderDate) from source.


Fact Table PipeLine-Incremental Load
Get Latest date
Get_Latest_Date_FromStoreSale
Get_Latest_Date_FromOnlineSale
ETL_FactTable Load Processing
FactStoreSale_ETL_Pipeline
Staging_FactStoreSale
DFT_Staging_FactStoreSales
WH_FactStoreSale
DFT_TargetDB_FactStoreSales
Truncate_Staging_FactStoreSales
FactOnlineSale_ETL_Pipeline
Staging_FactOnlineSale
DFT_Staging_FactOnlineSales
WH_FactOnlineSale
DFT_TargetDB_FactOnlineSales
Truncate_Staging_FactOnlineSales
Update the Latest date
Update_LatestDate_Loaded_FromOnlineSale
Update_LatestDate_Loaded_FromStoreSale

Incremental Load for Fact Table

V. The Result of deploy and Scheduling



Log file summary: No filter applied

Date	Step ID	Server	Job Name	Step Name	Notifications	Message
11/11/2025 5:10:00 PM		USER\TEST	ETL_AdventureWorks_Project_DailyLoad			The job succeeded. The Job was invoked by Schedule 11 (Morning ETL Load). The las
11/11/2025 5:10:01 P...	1	USER\TEST	ETL_AdventureWorks_Project_DailyLoad	Run SSIS...		Executed as user: USER\samba. Microsoft (R) SQL Server Execute Package Utility Ve

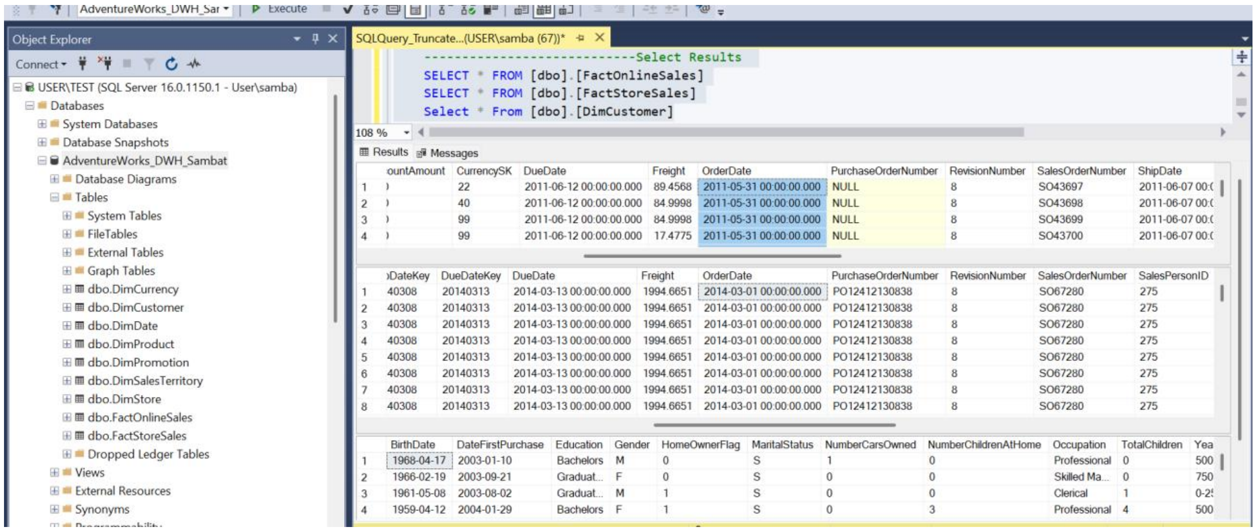
```
The Job was invoked by Schedule 11 (Morning ETL Load). The last step
ded. The Job was invoked by Schedule 11 (Morning ETL Load). The las
ded. The Job was invoked by User User\samba. The last step to run wa
ded. The Job was invoked by Schedule 11 (Morning ETL Load). The las
ded. The Job was invoked by Schedule 11 (Morning ETL Load). The las
ded. The Job was invoked by Schedule 11 (Morning ETL Load). The las
ded. The Job was invoked by Schedule 11 (Morning ETL Load). The las
ded. The Job was invoked by Schedule 11 (Morning ETL Load). The las
ded. The Job was invoked by User User\samba. The last step to run wa
ded. The Job was invoked by Schedule 11 (Morning ETL Load). The las
```

Job Properties - ETL_AdventureWorks_Project_DailyLoad

Schedule list:

ID	Name	Ena...	Description	J...
11	Morning ETL Load	Yes	Occurs every day at 5:10:00 PM. Schedule will...	V...

The job succeeded history



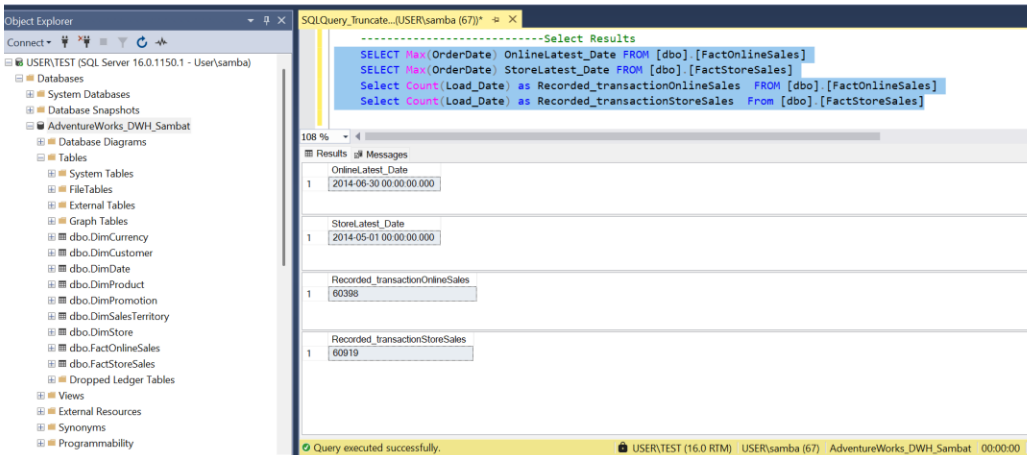
```
---------------------------Select Results
SELECT * FROM [dbo].[FactOnlineSales]
SELECT * FROM [dbo].[FactStoreSales]
Select * From [dbo].[DimCustomer]
```

	ountAmount	CurrencySK	DueDate	Freight	OrderDate	PurchaseOrderNumber	RevisionNumber	SalesOrderNumber	ShipDate
1	)	22	2011-06-12 00:00:00.000	89.4568	2011-05-31 00:00:00.000	NULL	8	SO43697	2011-06-07 00:(
2	)	40	2011-06-12 00:00:00.000	84.9998	2011-05-31 00:00:00.000	NULL	8	SO43698	2011-06-07 00:(
3	)	99	2011-06-12 00:00:00.000	84.9998	2011-05-31 00:00:00.000	NULL	8	SO43699	2011-06-07 00:(
4	)	99	2011-06-12 00:00:00.000	17.4775	2011-05-31 00:00:00.000	NULL	8	SO43700	2011-06-07 00:(

	)DateKey	DueDateKey	DueDate	Freight	OrderDate	PurchaseOrderNumber	RevisionNumber	SalesOrderNumber	SalesPersonID
1	40308	20140313	2014-03-13 00:00:00.000	1994.6651	2014-03-01 00:00:00.000	PO12412130838	8	SO67280	275
2	40308	20140313	2014-03-13 00:00:00.000	1994.6651	2014-03-01 00:00:00.000	PO12412130838	8	SO67280	275
3	40308	20140313	2014-03-13 00:00:00.000	1994.6651	2014-03-01 00:00:00.000	PO12412130838	8	SO67280	275
4	40308	20140313	2014-03-13 00:00:00.000	1994.6651	2014-03-01 00:00:00.000	PO12412130838	8	SO67280	275
5	40308	20140313	2014-03-13 00:00:00.000	1994.6651	2014-03-01 00:00:00.000	PO12412130838	8	SO67280	275
6	40308	20140313	2014-03-13 00:00:00.000	1994.6651	2014-03-01 00:00:00.000	PO12412130838	8	SO67280	275
7	40308	20140313	2014-03-13 00:00:00.000	1994.6651	2014-03-01 00:00:00.000	PO12412130838	8	SO67280	275
8	40308	20140313	2014-03-13 00:00:00.000	1994.6651	2014-03-01 00:00:00.000	PO12412130838	8	SO67280	275

	BirthDate	DateFirstPurchase	Education	Gender	HomeOwnerFlag	MaritalStatus	NumberCarsOwned	NumberChildrenAtHome	Occupation	TotalChildren	Yea
1	1968-04-17	2003-01-10	Bachelors	M	0	S	1	0	Professional	0	500
2	1966-02-19	2003-09-21	Graduat...	F	0	S	0	0	Skilled Ma...	0	750
3	1961-05-08	2003-08-02	Graduat...	M	1	S	0	0	Clerical	1	0-2!
4	1959-04-12	2004-01-29	Bachelors	F	1	S	0	3	Professional	4	500

The Result query Select Tables from WH



```
---------------------------Select Results
SELECT Max(OrderDate) OnlineLatest_Date FROM [dbo].[FactOnlineSales]
SELECT Max(OrderDate) StoreLatest_Date FROM [dbo].[FactStoreSales]
Select Count(Load_Date) as Recorded_transactionOnlineSales  FROM [dbo].[FactOnlineSales]
Select Count(Load_Date) as Recorded_transactionStoreSales  From [dbo].[FactStoreSales]
```

	OnlineLatest_Date
1	2014-06-30 00:00:00.000

	StoreLatest_Date
1	2014-05-01 00:00:00.000

	Recorded_transactionOnlineSales
1	60398

	Recorded_transactionStoreSales
1	60919

Query executed successfully. USER\TEST (16.0 RTM) USER\samba (67) AdventureWorks_DWH_Sambat 00:00:00

The Result query Max (OderDate) And Count Rows from WH