Integrated ETL and Reporting System for Data-Driven Insights: A Personal Project Journey

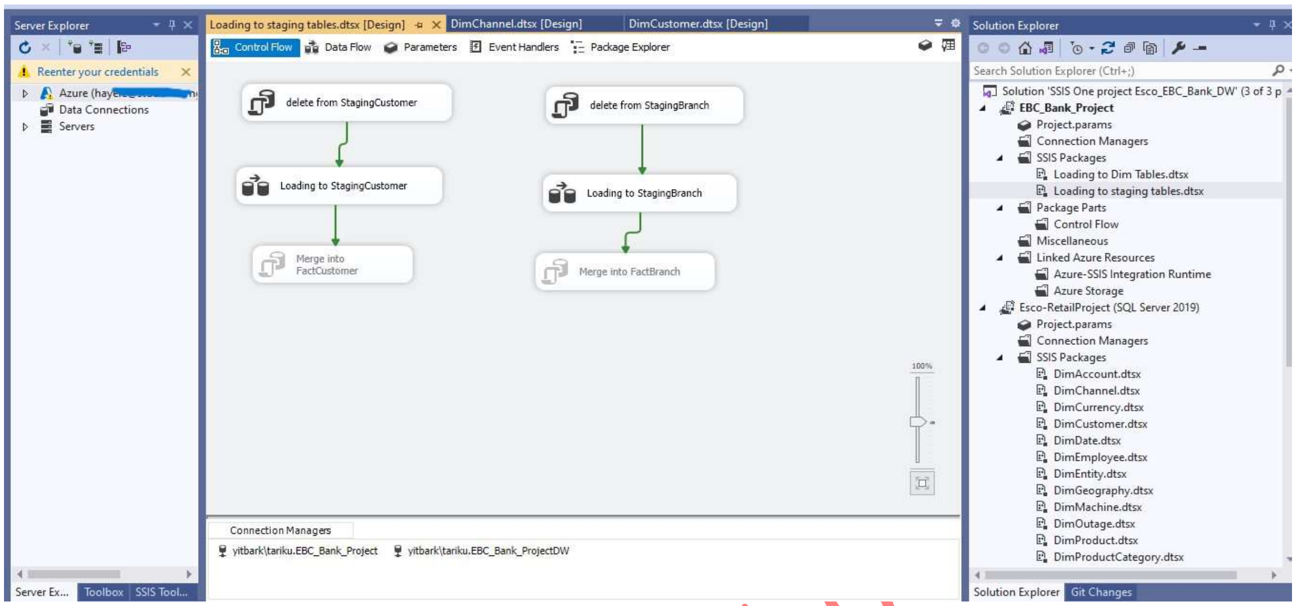
Introduction

My journey in data warehousing and ETL began with a personal project titled "Integrated ETL and Reporting System for Data-Driven Insights." This project was an exploration into the intricacies of data handling, transformation, and the creation of a robust data warehouse.

Project Genesis: The Why

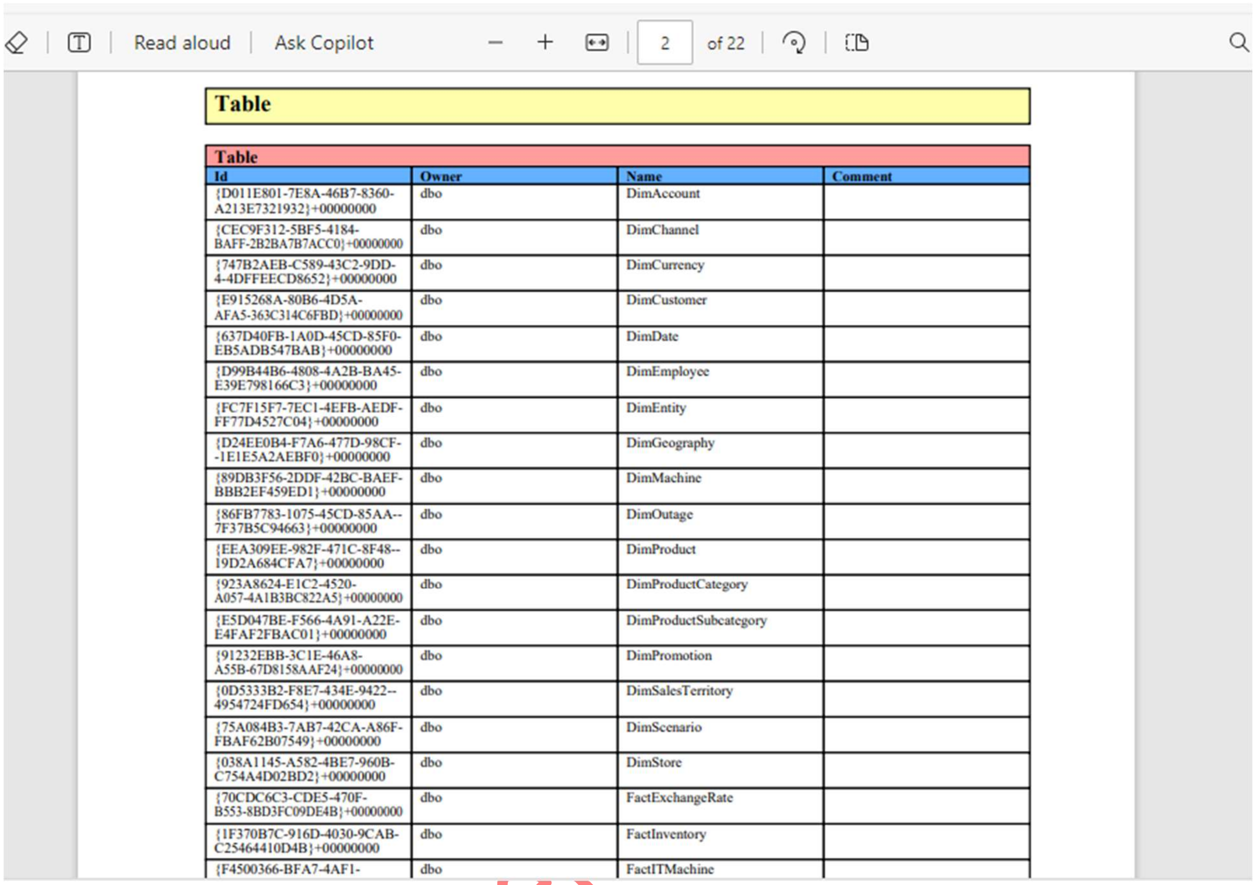
This project was born from a desire to:

1. **Explore Data Integration:** Understanding the nuances of consolidating varied data sources.
2. **Enhance Reporting Techniques:** Improving data reporting accuracy.
3. **Boost Performance:** Experimenting with efficient data processing methods.
4. **Scale for Growth:** Building a system capable of handling growing data volumes.



Data Mapping: Crafting the Blueprint

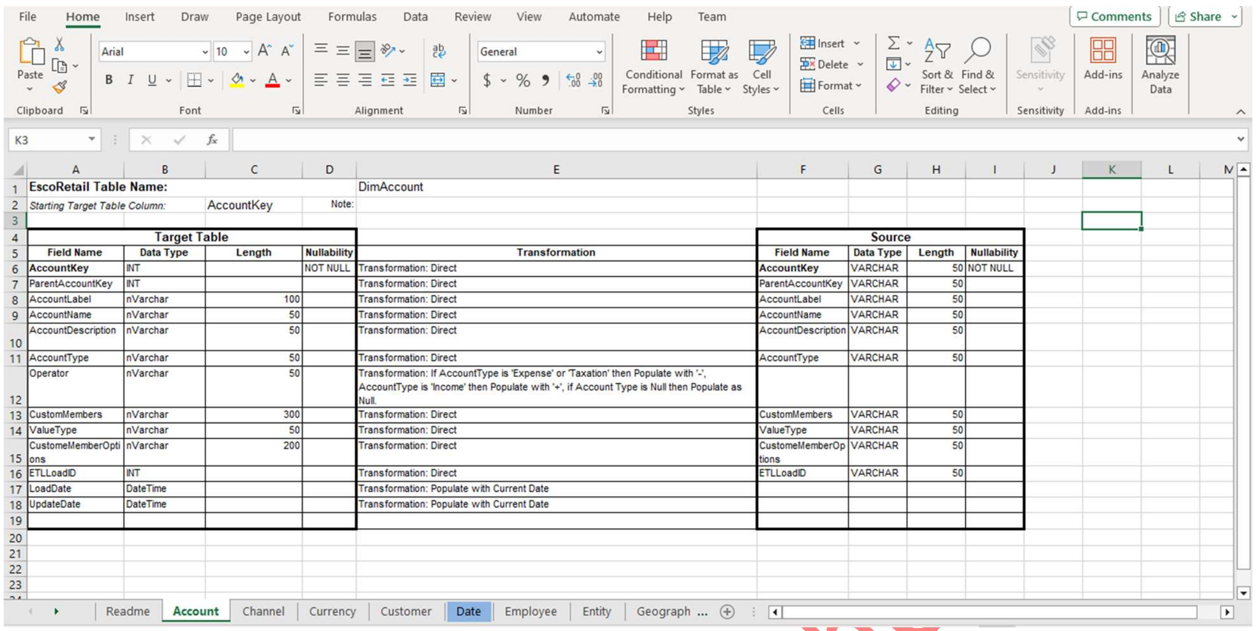
A key challenge was data mapping, where I aligned diverse data sources to staging tables. This process involved detailed planning and execution to ensure data was appropriately structured for ETL processing. Insights from the "DataModel_Document.pdf" were instrumental in guiding this process.



Table

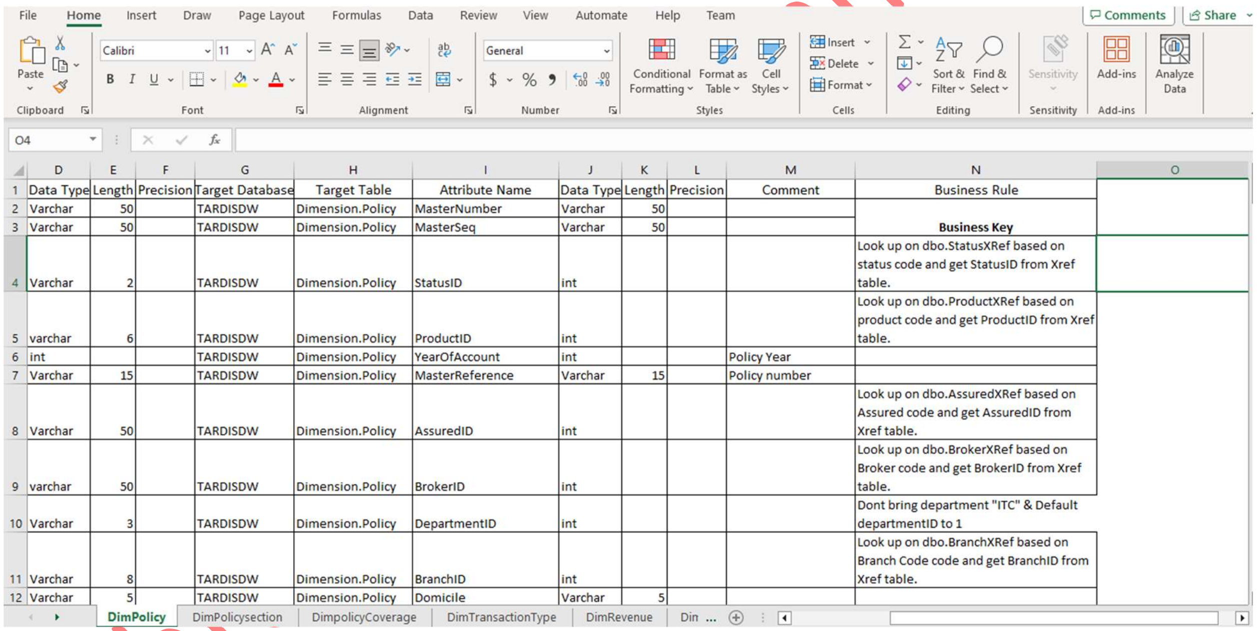
Table

Id	Owner	Name	Comment
{D011E801-7E8A-46B7-8360-A213E7321932}+00000000	dbo	DimAccount	
{CEC9F312-5BF5-4184-BAFF-2B2BA7B7ACC0}+00000000	dbo	DimChannel	
{747B2AEB-C589-43C2-9DD-4-4DFFEECD8652}+00000000	dbo	DimCurrency	
{E915268A-80B6-4D5A-AFA5-363C314C6FBD}+00000000	dbo	DimCustomer	
{637D40FB-1A0D-45CD-85F0-EB5ADB547BAB}+00000000	dbo	DimDate	
{D99B44B6-4808-4A2B-BA45-E39E798166C3}+00000000	dbo	DimEmployee	
{FC7F15F7-7EC1-4EFB-AEDF-FF77D4527C04}+00000000	dbo	DimEntity	
{D24EE0B4-F7A6-477D-98CF--1E1E5A2AEBF0}+00000000	dbo	DimGeography	
{89DB3F56-2DDF-42BC-BAEF-BBB2EF459ED1}+00000000	dbo	DimMachine	
{86FB7783-1075-45CD-85AA--7F37B5C94663}+00000000	dbo	DimOutage	
{EEA309EE-982F-471C-8F48--19D2A684CFA7}+00000000	dbo	DimProduct	
{923A8624-E1C2-4520-A057-4A1B3BC822A5}+00000000	dbo	DimProductCategory	
{E5D047BE-F566-4A91-A22E-E4FAF2FBAC01}+00000000	dbo	DimProductSubcategory	
{91232EBB-3C1E-46A8-A55B-67D8158AAF24}+00000000	dbo	DimPromotion	
{0D5333B2-F8E7-434E-9422--4954724FD654}+00000000	dbo	DimSalesTerritory	
{75A084B3-7AB7-42CA-A86F-FBAF62B07549}+00000000	dbo	DimScenario	
{038A1145-A582-4BE7-960B-C754A4D02BD2}+00000000	dbo	DimStore	
{70CDC6C3-CDE5-470F-B553-8BD3FC09DE4B}+00000000	dbo	FactExchangeRate	
{1F370B7C-916D-4030-9CAB-C25464410D4B}+00000000	dbo	FactInventory	
{F4500366-BFA7-4AF1-	dbo	FactITMachine	

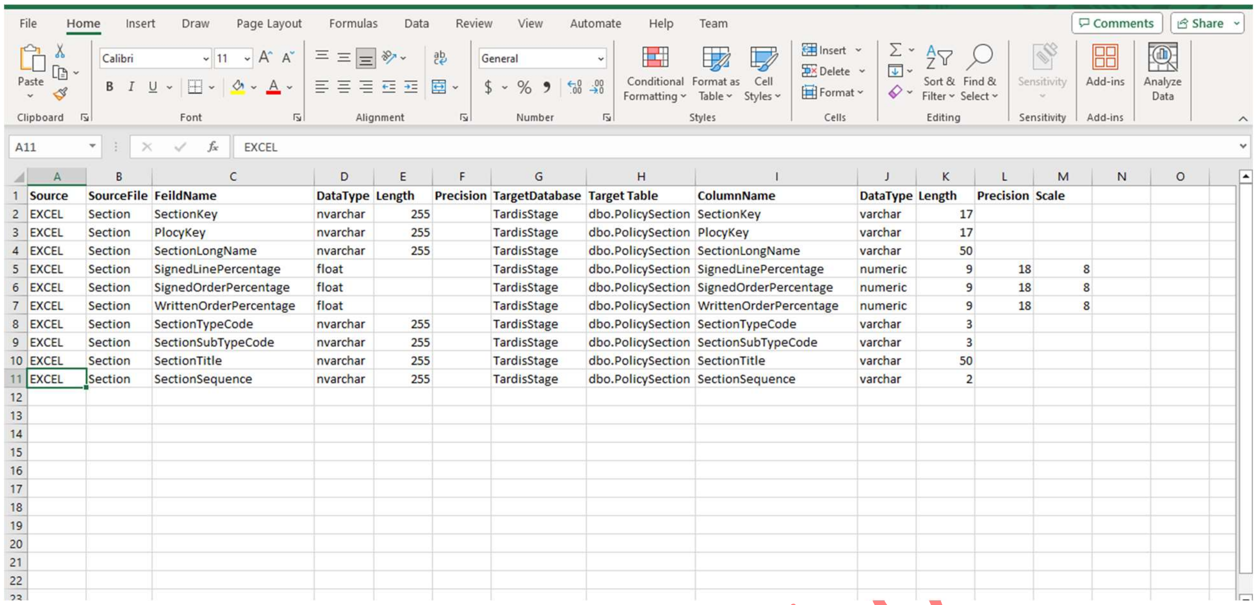


	A	B	C	D	E
1	**EscoRetail Table Name:**				DimAccount
2	*Starting Target Table Column:*		AccountKey	Note:	

Target Table					Source			
Field Name	**Data Type**	**Length**	**Nullability**	**Transformation**	**Field Name**	**Data Type**	**Length**	**Nullability**
AccountKey	INT		NOT NULL	Transformation: Direct	**AccountKey**	VARCHAR	50	NOT NULL
ParentAccountKey	INT			Transformation: Direct	ParentAccountKey	VARCHAR	50	
AccountLabel	nVarchar	100		Transformation: Direct	AccountLabel	VARCHAR	50	
AccountName	nVarchar	50		Transformation: Direct	AccountName	VARCHAR	50	
AccountDescription	nVarchar	50		Transformation: Direct	AccountDescription	VARCHAR	50	
AccountType	nVarchar	50		Transformation: Direct	AccountType	VARCHAR	50	
Operator	nVarchar	50		Transformation: If AccountType is 'Expense' or 'Taxation' then Populate with '-', AccountType is 'Income' then Populate with '+', if Account Type is Null then Populate as Null.				
CustomMembers	nVarchar	300		Transformation: Direct	CustomMembers	VARCHAR	50	
ValueType	nVarchar	50		Transformation: Direct	ValueType	VARCHAR	50	
CustomeMemberOptions	nVarchar	200		Transformation: Direct	CustomeMemberOptions	VARCHAR	50	
ETLLoadID	INT			Transformation: Direct	ETLLoadID	VARCHAR	50	
LoadDate	DateTime			Transformation: Populate with Current Date				
UpdateDate	DateTime			Transformation: Populate with Current Date				



	D	E	F	G	H	I	J	K	L	M	N
1	Data Type	Length	Precision	Target Database	Target Table	Attribute Name	Data Type	Length	Precision	Comment	Business Rule
2	Varchar	50		TARDISDW	Dimension.Policy	MasterNumber	Varchar	50			
3	Varchar	50		TARDISDW	Dimension.Policy	MasterSeq	Varchar	50			**Business Key**
4	Varchar	2		TARDISDW	Dimension.Policy	StatusID	int				Look up on dbo.StatusXRef based on status code and get StatusID from Xref table.
5	varchar	6		TARDISDW	Dimension.Policy	ProductID	int				Look up on dbo.ProductXRef based on product code and get ProductID from Xref table.
6	int			TARDISDW	Dimension.Policy	YearOfAccount	int			Policy Year	
7	Varchar	15		TARDISDW	Dimension.Policy	MasterReference	Varchar	15		Policy number	
8	Varchar	50		TARDISDW	Dimension.Policy	AssuredID	int				Look up on dbo.AssuredXRef based on Assured code and get AssuredID from Xref table.
9	varchar	50		TARDISDW	Dimension.Policy	BrokerID	int				Look up on dbo.BrokerXRef based on Broker code and get BrokerID from Xref table.
10	Varchar	3		TARDISDW	Dimension.Policy	DepartmentID	int				Dont bring department "ITC" & Default departmentID to 1
11	Varchar	8		TARDISDW	Dimension.Policy	BranchID	int				Look up on dbo.BranchXRef based on Branch Code code and get BranchID from Xref table.
12	Varchar	5		TARDISDW	Dimension.Policy	Domicile	Varchar	5			

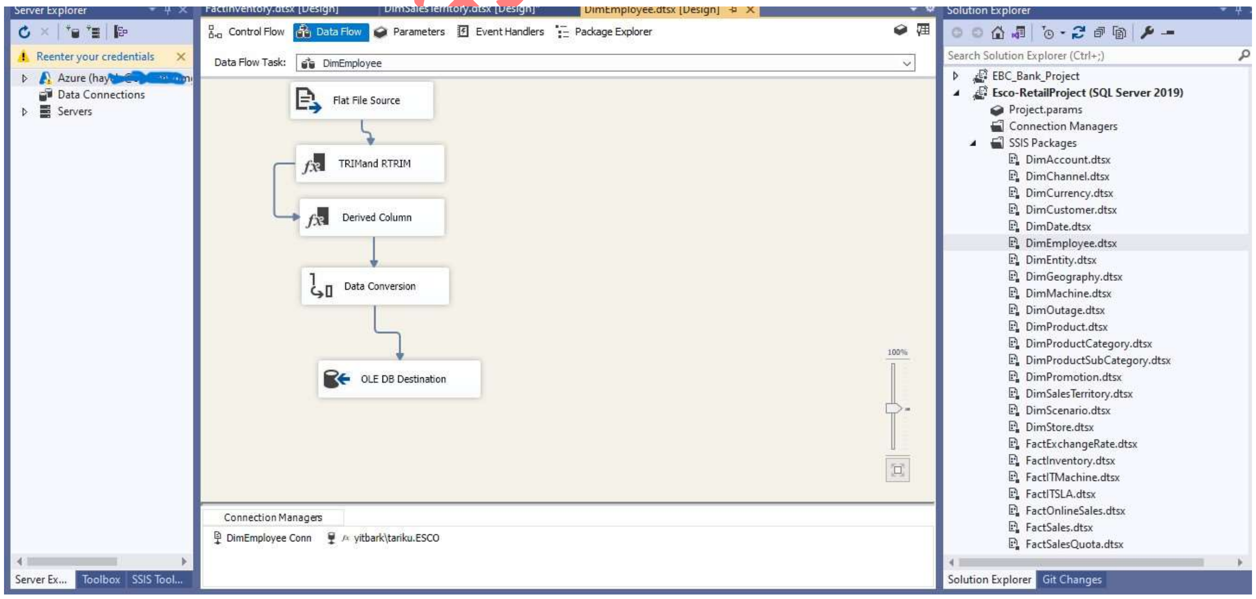


A11 — EXCEL

Source	SourceFile	FeildName	DataType	Length	Precision	TargetDatabase	Target Table	ColumnName	DataType	Length	Precision	Scale
EXCEL	Section	SectionKey	nvarchar	255		TardisStage	dbo.PolicySection	SectionKey	varchar	17		
EXCEL	Section	PlocyKey	nvarchar	255		TardisStage	dbo.PolicySection	PlocyKey	varchar	17		
EXCEL	Section	SectionLongName	nvarchar	255		TardisStage	dbo.PolicySection	SectionLongName	varchar	50		
EXCEL	Section	SignedLinePercentage	float			TardisStage	dbo.PolicySection	SignedLinePercentage	numeric	9	18	8
EXCEL	Section	SignedOrderPercentage	float			TardisStage	dbo.PolicySection	SignedOrderPercentage	numeric	9	18	8
EXCEL	Section	WrittenOrderPercentage	float			TardisStage	dbo.PolicySection	WrittenOrderPercentage	numeric	9	18	8
EXCEL	Section	SectionTypeCode	nvarchar	255		TardisStage	dbo.PolicySection	SectionTypeCode	varchar	3		
EXCEL	Section	SectionSubTypeCode	nvarchar	255		TardisStage	dbo.PolicySection	SectionSubTypeCode	varchar	3		
EXCEL	Section	SectionTitle	nvarchar	255		TardisStage	dbo.PolicySection	SectionTitle	varchar	50		
EXCEL	Section	SectionSequence	nvarchar	255		TardisStage	dbo.PolicySection	SectionSequence	varchar	2		

ETL Process: The Heart of Data Transformation

I utilized SQL Server Integration Services (SSIS) to develop complex ETL packages, each tailored to handle different data structures. This was pivotal in maintaining data integrity and uniformity.



Server Explorer: Reenter your credentials; Azure (hay[redacted]; Data Connections; Servers

Tabs: FactInventory.dtsx [Design] | DimSalesTerritory.dtsx [Design] | DimEmployee.dtsx [Design]

Control Flow | Data Flow | Parameters | Event Handlers | Package Explorer

Data Flow Task: DimEmployee

Flat File Source → TRIMand RTRIM → Derived Column → Data Conversion → OLE DB Destination

Connection Managers: DimEmployee Conn; yitbark\tariku.ESCO

Solution Explorer:
- EBC_Bank_Project
- Esco-RetailProject (SQL Server 2019)
  - Project.params
  - Connection Managers
  - SSIS Packages
    - DimAccount.dtsx
    - DimChannel.dtsx
    - DimCurrency.dtsx
    - DimCustomer.dtsx
    - DimDate.dtsx
    - DimEmployee.dtsx
    - DimEntity.dtsx
    - DimGeography.dtsx
    - DimMachine.dtsx
    - DimOutage.dtsx
    - DimProduct.dtsx
    - DimProductCategory.dtsx
    - DimProductSubCategory.dtsx
    - DimPromotion.dtsx
    - DimSalesTerritory.dtsx
    - DimScenario.dtsx
    - DimStore.dtsx
    - FactExchangeRate.dtsx
    - FactInventory.dtsx
    - FactITMachine.dtsx
    - FactITSLA.dtsx
    - FactOnlineSales.dtsx
    - FactSales.dtsx
    - FactSalesQuota.dtsx

Data Cleaning: Ensuring Quality and Precision

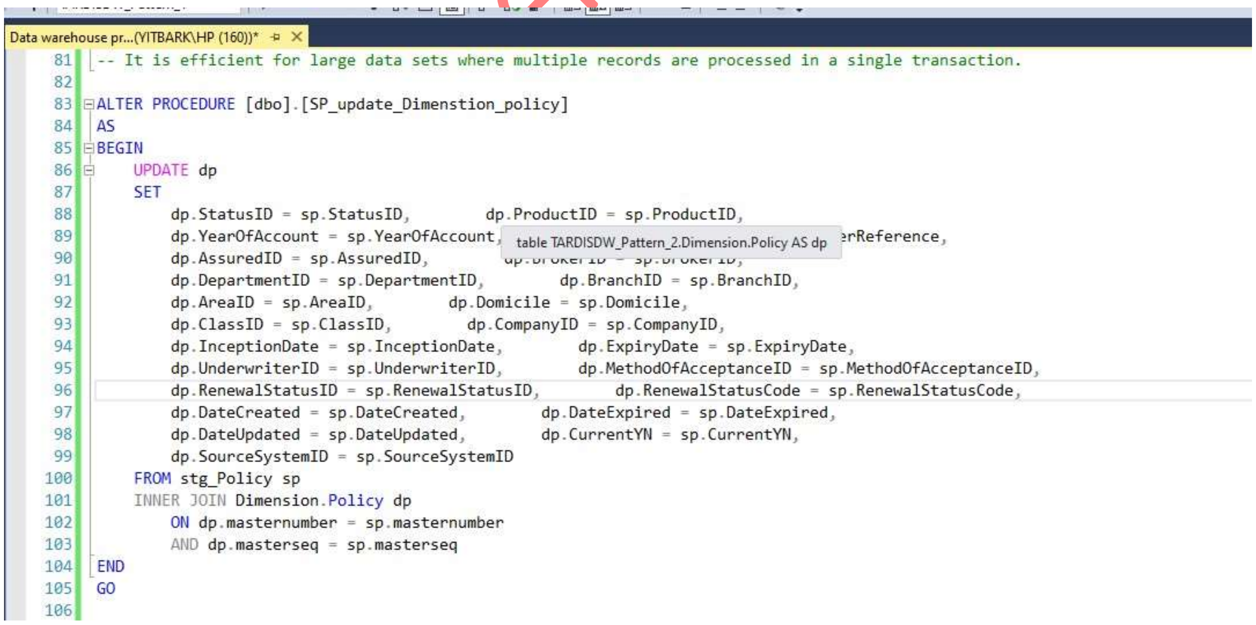
Data cleanliness was a major focus. I used data validation, deduplication, and error handling techniques to ensure high data quality.

Data Warehouse Population: A Multifaceted Approach

Various methods were employed to populate the data warehouse, including:

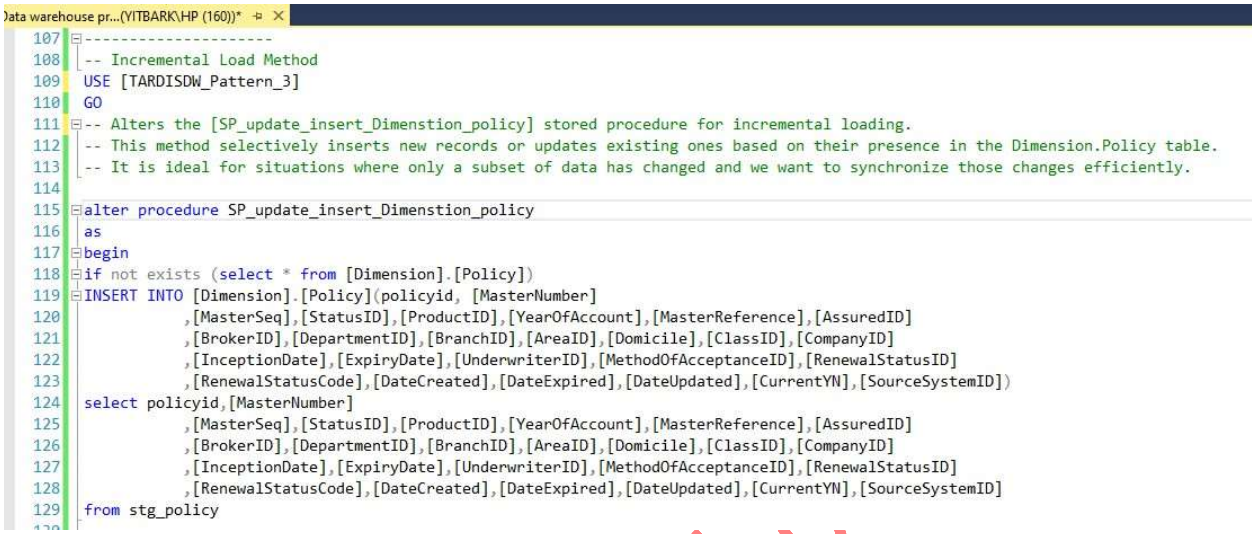
1. **Direct Insert:** For immediate data updates.
2. **Batch Insert/Update:** For large data sets.
3. **Incremental Load with SCD:** Managing historical data changes.
4. **CTE:** For complex query management.

1. **Direct Insert:** For immediate data updates.

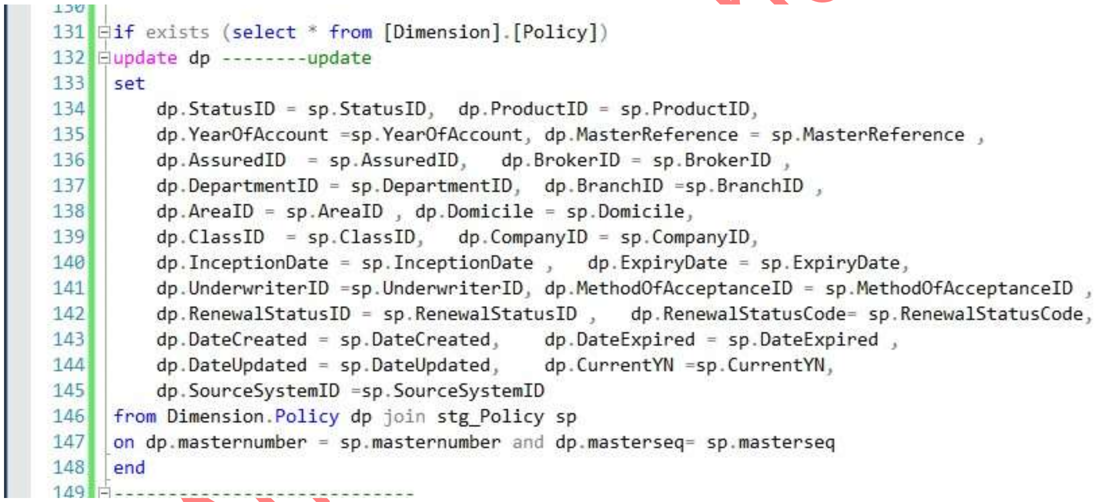


```
-- It is efficient for large data sets where multiple records are processed in a single transaction.

ALTER PROCEDURE [dbo].[SP_update_Dimenstion_policy]
AS
BEGIN
    UPDATE dp
    SET
        dp.StatusID = sp.StatusID,        dp.ProductID = sp.ProductID,
        dp.YearOfAccount = sp.YearOfAccount, table TARDISDW_Pattern_2.Dimension.Policy AS dp erReference,
        dp.AssuredID = sp.AssuredID,      dp.BrokerID = sp.BrokerID,
        dp.DepartmentID = sp.DepartmentID,        dp.BranchID = sp.BranchID,
        dp.AreaID = sp.AreaID,        dp.Domicile = sp.Domicile,
        dp.ClassID = sp.ClassID,        dp.CompanyID = sp.CompanyID,
        dp.InceptionDate = sp.InceptionDate,        dp.ExpiryDate = sp.ExpiryDate,
        dp.UnderwriterID = sp.UnderwriterID,        dp.MethodOfAcceptanceID = sp.MethodOfAcceptanceID,
        dp.RenewalStatusID = sp.RenewalStatusID,        dp.RenewalStatusCode = sp.RenewalStatusCode,
        dp.DateCreated = sp.DateCreated,        dp.DateExpired = sp.DateExpired,
        dp.DateUpdated = sp.DateUpdated,        dp.CurrentYN = sp.CurrentYN,
        dp.SourceSystemID = sp.SourceSystemID
    FROM stg_Policy sp
    INNER JOIN Dimension.Policy dp
        ON dp.masternumber = sp.masternumber
        AND dp.masterseq = sp.masterseq
END
GO
```

2. **Batch Insert/Update:** For large data sets.

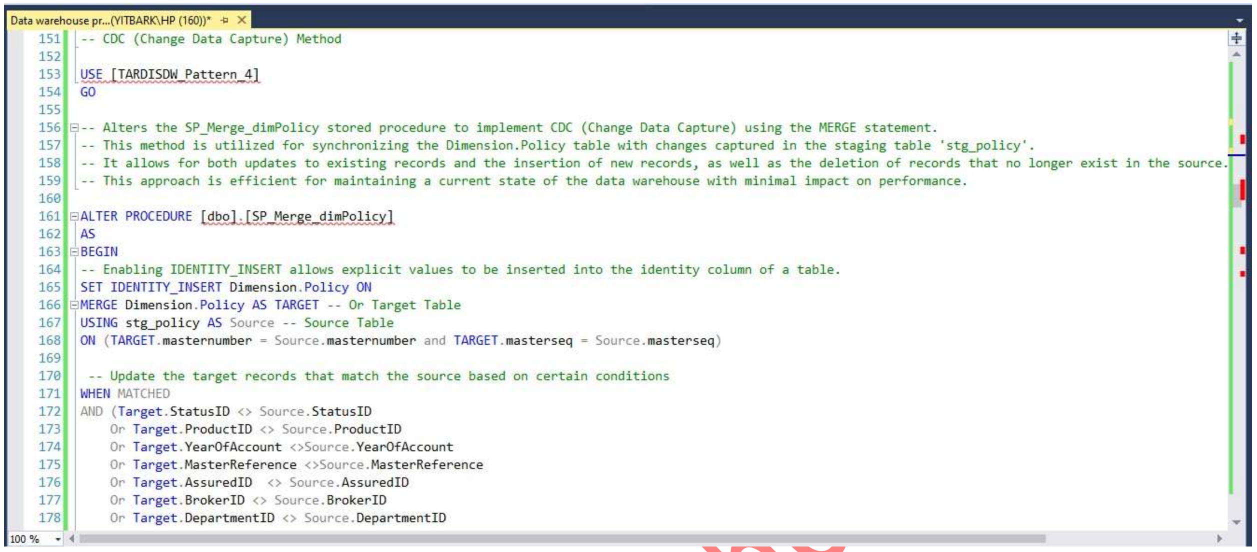


```
--------------------
-- Incremental Load Method
USE [TARDISDW_Pattern_3]
GO
-- Alters the [SP_update_insert_Dimenstion_policy] stored procedure for incremental loading.
-- This method selectively inserts new records or updates existing ones based on their presence in the Dimension.Policy table.
-- It is ideal for situations where only a subset of data has changed and we want to synchronize those changes efficiently.

alter procedure SP_update_insert_Dimenstion_policy
as
begin
if not exists (select * from [Dimension].[Policy])
INSERT INTO [Dimension].[Policy](policyid, [MasterNumber]
           ,[MasterSeq],[StatusID],[ProductID],[YearOfAccount],[MasterReference],[AssuredID]
           ,[BrokerID],[DepartmentID],[BranchID],[AreaID],[Domicile],[ClassID],[CompanyID]
           ,[InceptionDate],[ExpiryDate],[UnderwriterID],[MethodOfAcceptanceID],[RenewalStatusID]
           ,[RenewalStatusCode],[DateCreated],[DateExpired],[DateUpdated],[CurrentYN],[SourceSystemID])
select policyid,[MasterNumber]
           ,[MasterSeq],[StatusID],[ProductID],[YearOfAccount],[MasterReference],[AssuredID]
           ,[BrokerID],[DepartmentID],[BranchID],[AreaID],[Domicile],[ClassID],[CompanyID]
           ,[InceptionDate],[ExpiryDate],[UnderwriterID],[MethodOfAcceptanceID],[RenewalStatusID]
           ,[RenewalStatusCode],[DateCreated],[DateExpired],[DateUpdated],[CurrentYN],[SourceSystemID]
from stg_policy
```

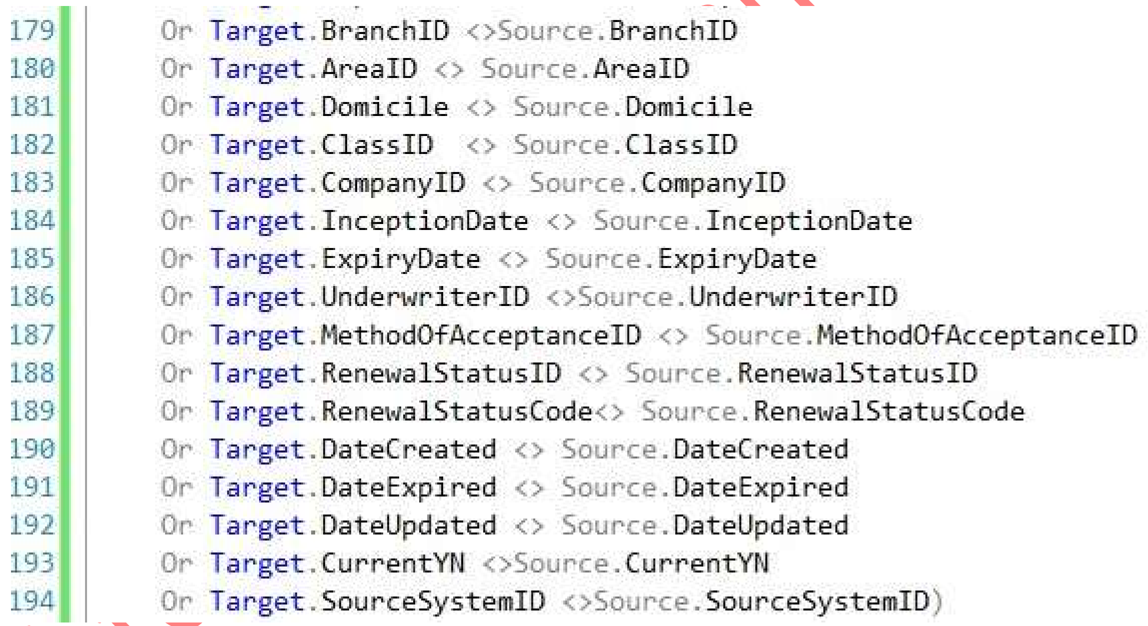


```
if exists (select * from [Dimension].[Policy])
update dp --------update
set
    dp.StatusID = sp.StatusID,  dp.ProductID = sp.ProductID,
    dp.YearOfAccount =sp.YearOfAccount, dp.MasterReference = sp.MasterReference ,
    dp.AssuredID  = sp.AssuredID,   dp.BrokerID = sp.BrokerID ,
    dp.DepartmentID = sp.DepartmentID,  dp.BranchID =sp.BranchID ,
    dp.AreaID = sp.AreaID , dp.Domicile = sp.Domicile,
    dp.ClassID  = sp.ClassID,   dp.CompanyID = sp.CompanyID,
    dp.InceptionDate = sp.InceptionDate ,  dp.ExpiryDate = sp.ExpiryDate,
    dp.UnderwriterID =sp.UnderwriterID, dp.MethodOfAcceptanceID = sp.MethodOfAcceptanceID ,
    dp.RenewalStatusID = sp.RenewalStatusID ,   dp.RenewalStatusCode= sp.RenewalStatusCode,
    dp.DateCreated = sp.DateCreated,    dp.DateExpired = sp.DateExpired ,
    dp.DateUpdated = sp.DateUpdated,    dp.CurrentYN =sp.CurrentYN,
    dp.SourceSystemID =sp.SourceSystemID
from Dimension.Policy dp join stg_Policy sp
on dp.masternumber = sp.masternumber and dp.masterseq= sp.masterseq
end
----------------------------
```

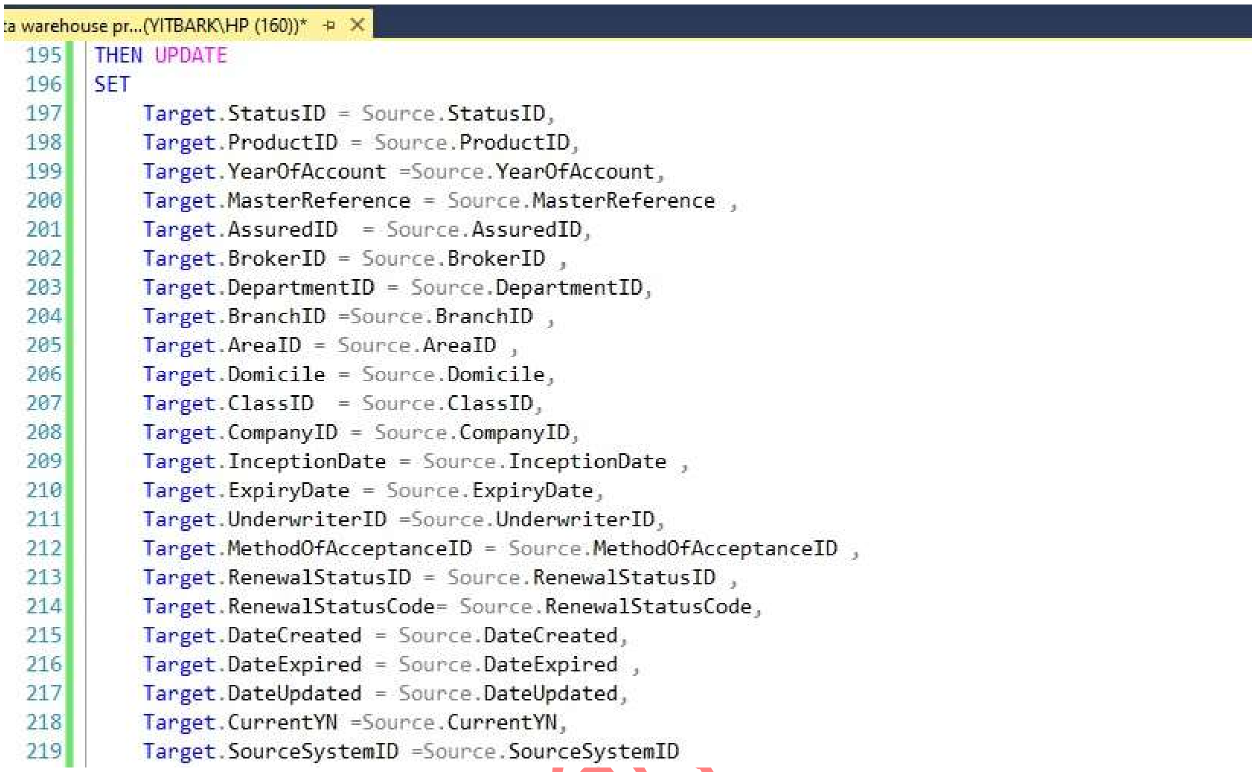
3. **Incremental Load with SCD:** Managing historical data changes.



```sql
-- CDC (Change Data Capture) Method

USE [TARDISDW_Pattern_4]
GO

-- Alters the SP_Merge_dimPolicy stored procedure to implement CDC (Change Data Capture) using the MERGE statement.
-- This method is utilized for synchronizing the Dimension.Policy table with changes captured in the staging table 'stg_policy'.
-- It allows for both updates to existing records and the insertion of new records, as well as the deletion of records that no longer exist in the source.
-- This approach is efficient for maintaining a current state of the data warehouse with minimal impact on performance.

ALTER PROCEDURE [dbo].[SP_Merge_dimPolicy]
AS
BEGIN
-- Enabling IDENTITY_INSERT allows explicit values to be inserted into the identity column of a table.
SET IDENTITY_INSERT Dimension.Policy ON
MERGE Dimension.Policy AS TARGET -- Or Target Table
USING stg_policy AS Source -- Source Table
ON (TARGET.masternumber = Source.masternumber and TARGET.masterseq = Source.masterseq)

 -- Update the target records that match the source based on certain conditions
WHEN MATCHED
AND (Target.StatusID <> Source.StatusID
    Or Target.ProductID <> Source.ProductID
    Or Target.YearOfAccount <>Source.YearOfAccount
    Or Target.MasterReference <>Source.MasterReference
    Or Target.AssuredID  <> Source.AssuredID
    Or Target.BrokerID <> Source.BrokerID
    Or Target.DepartmentID <> Source.DepartmentID
```

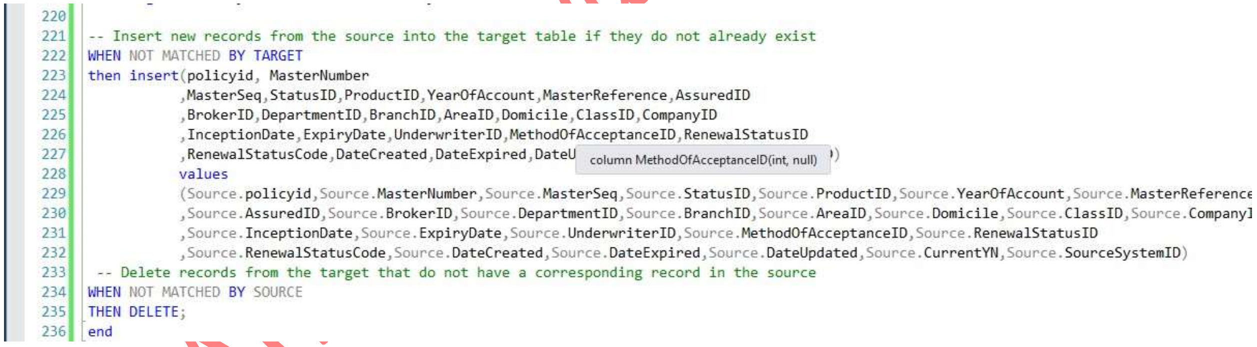


```sql
    Or Target.BranchID <>Source.BranchID
    Or Target.AreaID <> Source.AreaID
    Or Target.Domicile <> Source.Domicile
    Or Target.ClassID  <> Source.ClassID
    Or Target.CompanyID <> Source.CompanyID
    Or Target.InceptionDate <> Source.InceptionDate
    Or Target.ExpiryDate <> Source.ExpiryDate
    Or Target.UnderwriterID <>Source.UnderwriterID
    Or Target.MethodOfAcceptanceID <> Source.MethodOfAcceptanceID
    Or Target.RenewalStatusID <> Source.RenewalStatusID
    Or Target.RenewalStatusCode<> Source.RenewalStatusCode
    Or Target.DateCreated <> Source.DateCreated
    Or Target.DateExpired <> Source.DateExpired
    Or Target.DateUpdated <> Source.DateUpdated
    Or Target.CurrentYN <>Source.CurrentYN
    Or Target.SourceSystemID <>Source.SourceSystemID)
```



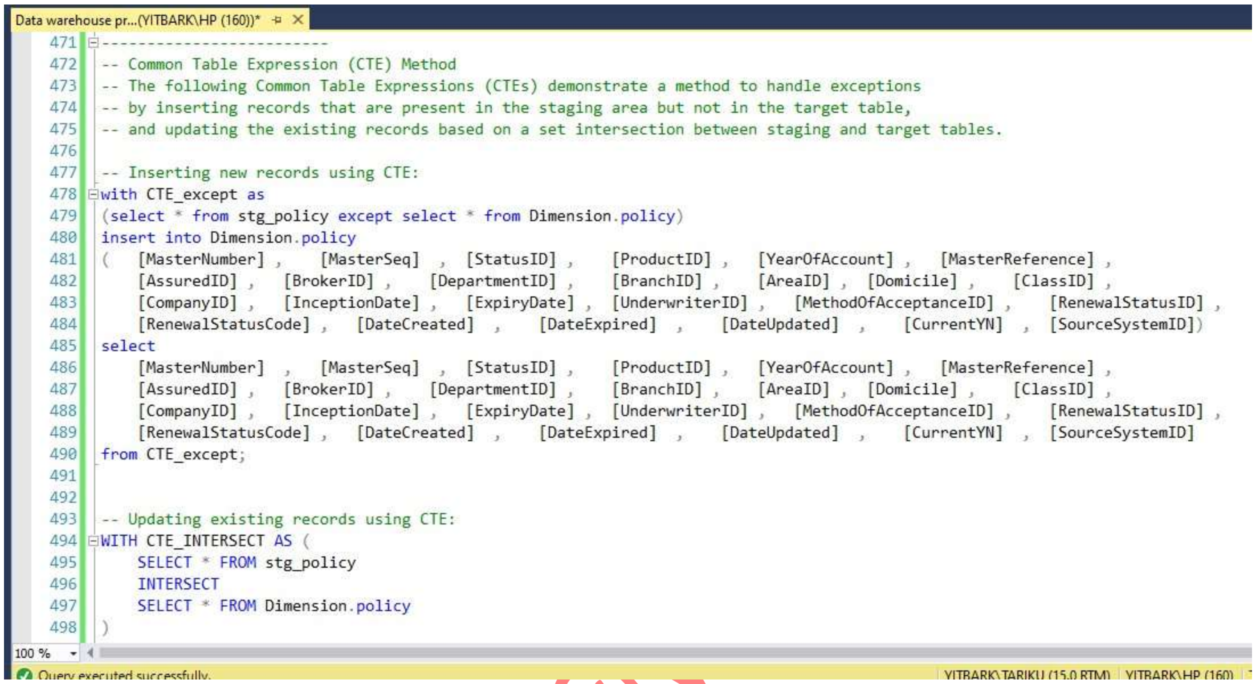
ta warehouse pr...(YITBARK\HP (160))*

```sql
THEN UPDATE
SET
    Target.StatusID = Source.StatusID,
    Target.ProductID = Source.ProductID,
    Target.YearOfAccount =Source.YearOfAccount,
    Target.MasterReference = Source.MasterReference ,
    Target.AssuredID  = Source.AssuredID,
    Target.BrokerID = Source.BrokerID ,
    Target.DepartmentID = Source.DepartmentID,
    Target.BranchID =Source.BranchID ,
    Target.AreaID = Source.AreaID ,
    Target.Domicile = Source.Domicile,
    Target.ClassID  = Source.ClassID,
    Target.CompanyID = Source.CompanyID,
    Target.InceptionDate = Source.InceptionDate ,
    Target.ExpiryDate = Source.ExpiryDate,
    Target.UnderwriterID =Source.UnderwriterID,
    Target.MethodOfAcceptanceID = Source.MethodOfAcceptanceID ,
    Target.RenewalStatusID = Source.RenewalStatusID ,
    Target.RenewalStatusCode= Source.RenewalStatusCode,
    Target.DateCreated = Source.DateCreated,
    Target.DateExpired = Source.DateExpired ,
    Target.DateUpdated = Source.DateUpdated,
    Target.CurrentYN =Source.CurrentYN,
    Target.SourceSystemID =Source.SourceSystemID
```



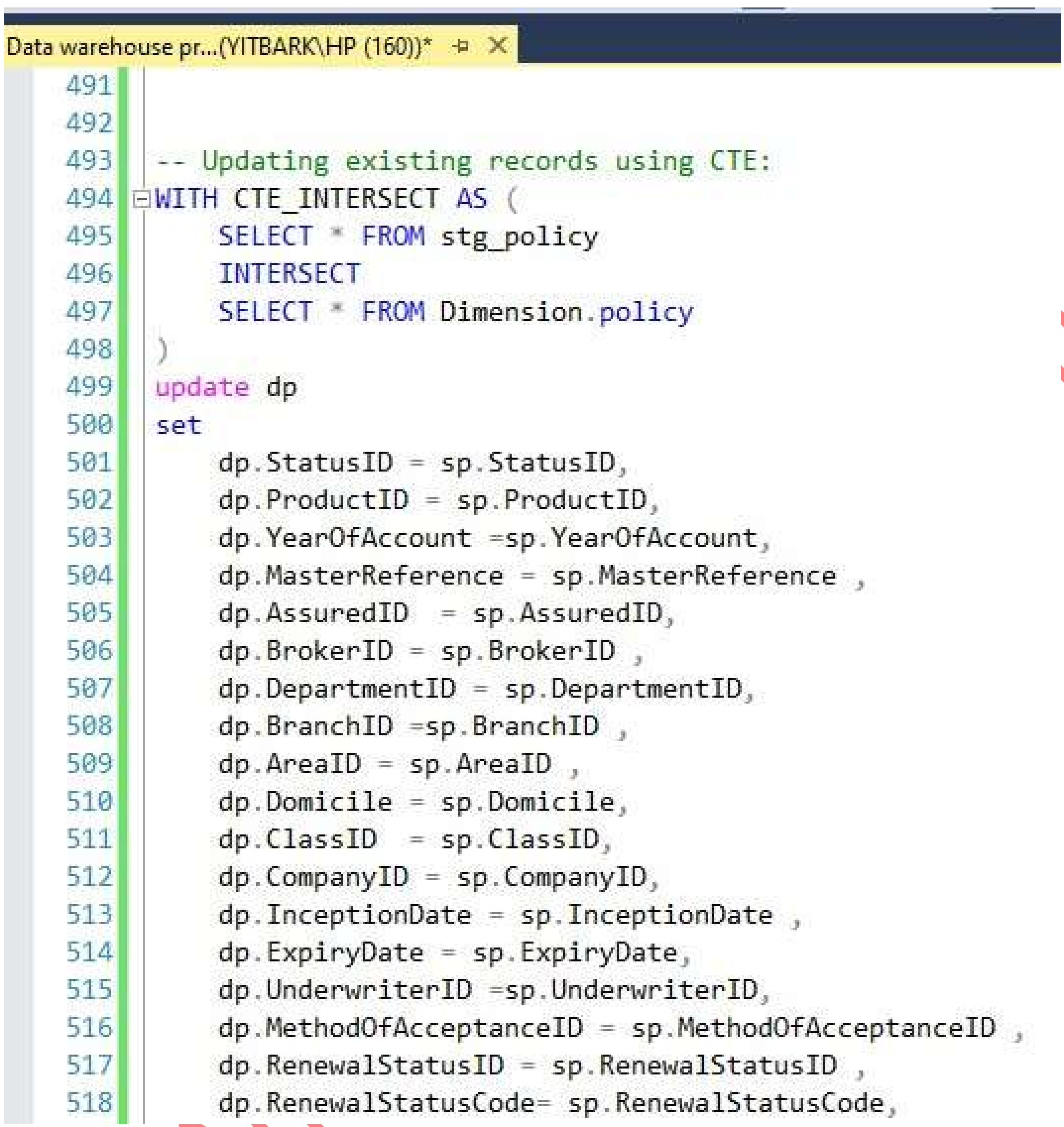
```sql

-- Insert new records from the source into the target table if they do not already exist
WHEN NOT MATCHED BY TARGET
then insert(policyid, MasterNumber
            ,MasterSeq,StatusID,ProductID,YearOfAccount,MasterReference,AssuredID
            ,BrokerID,DepartmentID,BranchID,AreaID,Domicile,ClassID,CompanyID
            ,InceptionDate,ExpiryDate,UnderwriterID,MethodOfAcceptanceID,RenewalStatusID
            ,RenewalStatusCode,DateCreated,DateExpired,DateU column MethodOfAcceptanceID(int, null) )
            values
            (Source.policyid,Source.MasterNumber,Source.MasterSeq,Source.StatusID,Source.ProductID,Source.YearOfAccount,Source.MasterReference
            ,Source.AssuredID,Source.BrokerID,Source.DepartmentID,Source.BranchID,Source.AreaID,Source.Domicile,Source.ClassID,Source.CompanyI
            ,Source.InceptionDate,Source.ExpiryDate,Source.UnderwriterID,Source.MethodOfAcceptanceID,Source.RenewalStatusID
            ,Source.RenewalStatusCode,Source.DateCreated,Source.DateExpired,Source.DateUpdated,Source.CurrentYN,Source.SourceSystemID)
 -- Delete records from the target that do not have a corresponding record in the source
WHEN NOT MATCHED BY SOURCE
THEN DELETE;
end
```

4. **CTE:** For complex query management.

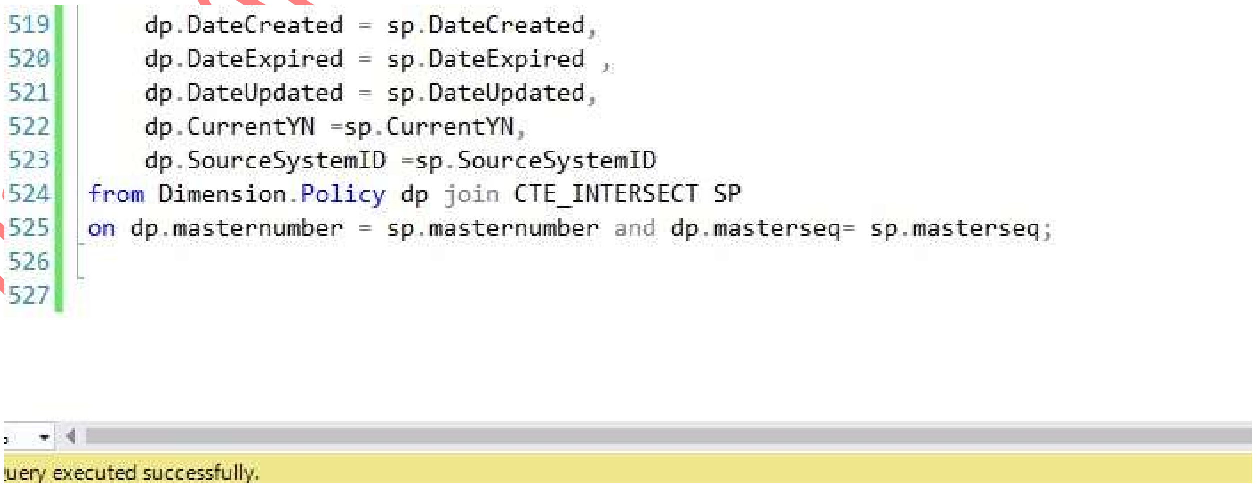


Data warehouse pr...(YITBARK\HP (160))*
100 %
Query executed successfully.
YITBARK\TARIKU (15.0 RTM)
YITBARK\HP (160)

```
-------------------------
-- Common Table Expression (CTE) Method
-- The following Common Table Expressions (CTEs) demonstrate a method to handle exceptions
-- by inserting records that are present in the staging area but not in the target table,
-- and updating the existing records based on a set intersection between staging and target tables.

-- Inserting new records using CTE:
with CTE_except as
(select * from stg_policy except select * from Dimension.policy)
insert into Dimension.policy
(    [MasterNumber] ,     [MasterSeq]  ,  [StatusID] ,    [ProductID] ,    [YearOfAccount] ,   [MasterReference] ,
     [AssuredID] ,    [BrokerID] ,    [DepartmentID] ,    [BranchID] ,    [AreaID] ,  [Domicile] ,    [ClassID] ,
     [CompanyID] ,    [InceptionDate] ,   [ExpiryDate] ,  [UnderwriterID] ,    [MethodOfAcceptanceID] ,    [RenewalStatusID] ,
     [RenewalStatusCode] ,   [DateCreated]  ,    [DateExpired]  ,    [DateUpdated]  ,    [CurrentYN]  ,  [SourceSystemID])
select
     [MasterNumber]  ,    [MasterSeq]  ,  [StatusID] ,    [ProductID] ,    [YearOfAccount] ,   [MasterReference] ,
     [AssuredID] ,    [BrokerID] ,    [DepartmentID] ,    [BranchID] ,    [AreaID] ,  [Domicile] ,    [ClassID] ,
     [CompanyID] ,    [InceptionDate] ,   [ExpiryDate] ,  [UnderwriterID] ,    [MethodOfAcceptanceID] ,    [RenewalStatusID] ,
     [RenewalStatusCode] ,   [DateCreated]  ,    [DateExpired]  ,    [DateUpdated]  ,    [CurrentYN]  ,  [SourceSystemID]
from CTE_except;


-- Updating existing records using CTE:
WITH CTE_INTERSECT AS (
     SELECT * FROM stg_policy
     INTERSECT
     SELECT * FROM Dimension.policy
)
```



```sql


-- Updating existing records using CTE:
WITH CTE_INTERSECT AS (
    SELECT * FROM stg_policy
    INTERSECT
    SELECT * FROM Dimension.policy
)
update dp
set
    dp.StatusID = sp.StatusID,
    dp.ProductID = sp.ProductID,
    dp.YearOfAccount =sp.YearOfAccount,
    dp.MasterReference = sp.MasterReference ,
    dp.AssuredID  = sp.AssuredID,
    dp.BrokerID = sp.BrokerID ,
    dp.DepartmentID = sp.DepartmentID,
    dp.BranchID =sp.BranchID ,
    dp.AreaID = sp.AreaID ,
    dp.Domicile = sp.Domicile,
    dp.ClassID  = sp.ClassID,
    dp.CompanyID = sp.CompanyID,
    dp.InceptionDate = sp.InceptionDate ,
    dp.ExpiryDate = sp.ExpiryDate,
    dp.UnderwriterID =sp.UnderwriterID,
    dp.MethodOfAcceptanceID = sp.MethodOfAcceptanceID ,
    dp.RenewalStatusID = sp.RenewalStatusID ,
    dp.RenewalStatusCode= sp.RenewalStatusCode,
```



```sql
    dp.DateCreated = sp.DateCreated,
    dp.DateExpired = sp.DateExpired ,
    dp.DateUpdated = sp.DateUpdated,
    dp.CurrentYN =sp.CurrentYN,
    dp.SourceSystemID =sp.SourceSystemID
from Dimension.Policy dp join CTE_INTERSECT SP
on dp.masternumber = sp.masternumber and dp.masterseq= sp.masterseq;


```

uery executed successfully.

Integrating Data Models

The data models, as detailed in the "DataModel_Document.pdf," formed the backbone of the reporting system. This document guided the structuring of data and relationships between entities, which was crucial for the integrity and efficiency of the data warehouse.

SSRS Reporting: Bringing Data to Life

The creation of SSRS reports was a key component of this project. These reports were designed to visualize the data processed and stored in the data warehouse, providing actionable insights.

SSRS Report Queries

Here are some examples of the queries used in the SSRS reports:

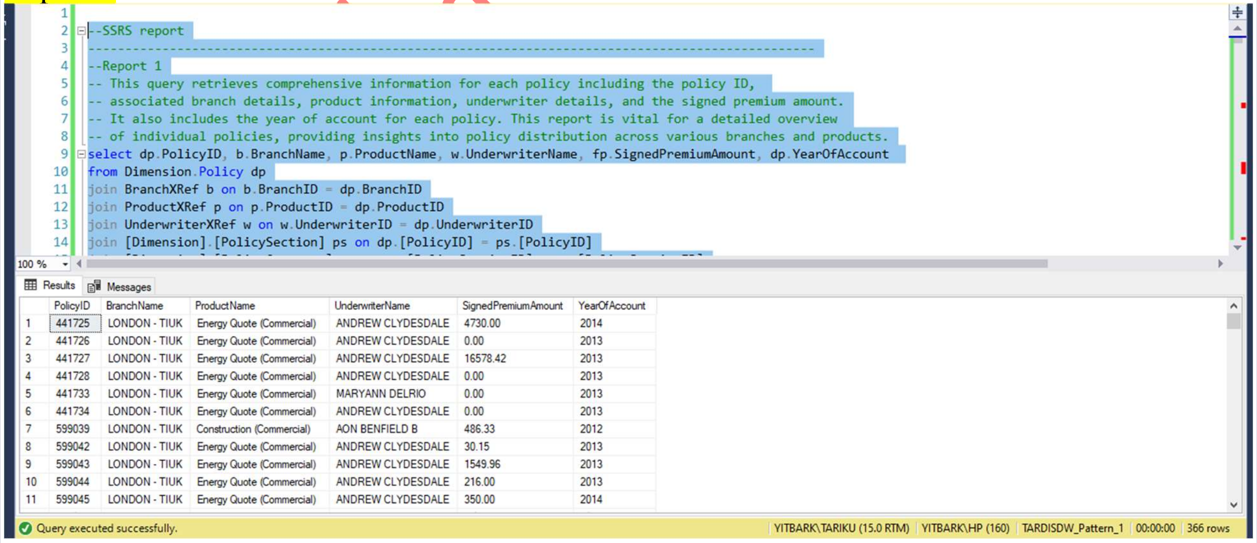
Report 1: Joins multiple tables to provide a complete overview of policy details and financials.

Report 2: Focuses on policy sections and coverage with performance metrics.

Report 3: Consolidates company and branch-wise policy and financial data.

Report 4: Calculates net amounts by considering policy premiums and deductions.

Report 1

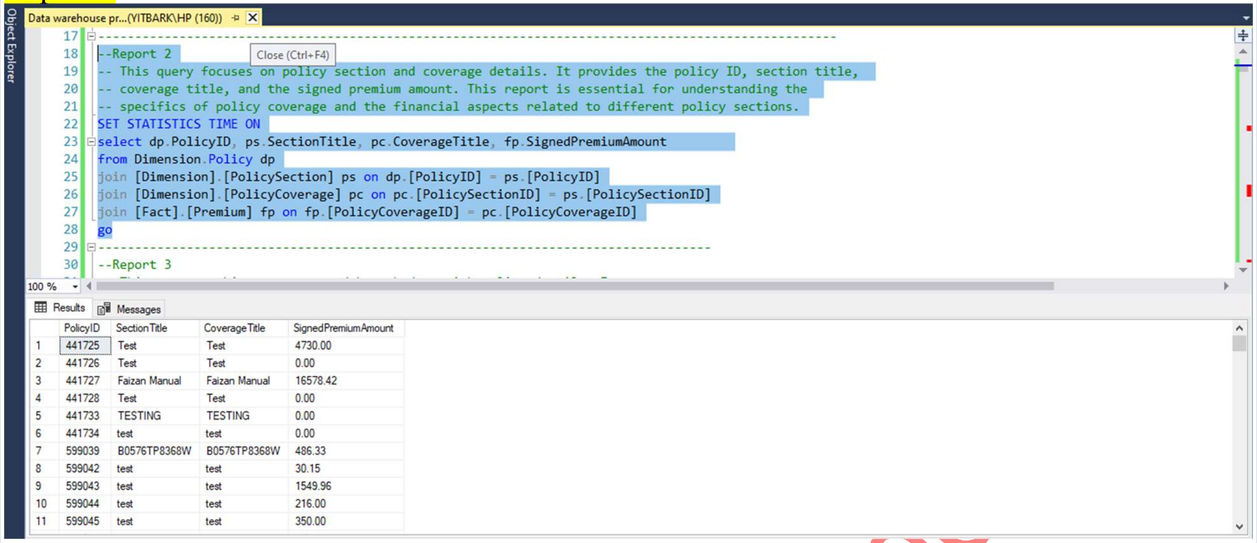


```
--SSRS report
---------------------------------------------------------------------------------------------------
--Report 1
-- This query retrieves comprehensive information for each policy including the policy ID,
-- associated branch details, product information, underwriter details, and the signed premium amount.
-- It also includes the year of account for each policy. This report is vital for a detailed overview
-- of individual policies, providing insights into policy distribution across various branches and products.
select dp.PolicyID, b.BranchName, p.ProductName, w.UnderwriterName, fp.SignedPremiumAmount, dp.YearOfAccount
from Dimension.Policy dp
join BranchXRef b on b.BranchID = dp.BranchID
join ProductXRef p on p.ProductID = dp.ProductID
join UnderwriterXRef w on w.UnderwriterID = dp.UnderwriterID
join [Dimension].[PolicySection] ps on dp.[PolicyID] = ps.[PolicyID]
```

	PolicyID	BranchName	ProductName	UnderwriterName	SignedPremiumAmount	YearOfAccount
1	441725	LONDON - TIUK	Energy Quote (Commercial)	ANDREW CLYDESDALE	4730.00	2014
2	441726	LONDON - TIUK	Energy Quote (Commercial)	ANDREW CLYDESDALE	0.00	2013
3	441727	LONDON - TIUK	Energy Quote (Commercial)	ANDREW CLYDESDALE	16578.42	2013
4	441728	LONDON - TIUK	Energy Quote (Commercial)	ANDREW CLYDESDALE	0.00	2013
5	441733	LONDON - TIUK	Energy Quote (Commercial)	MARYANN DELRIO	0.00	2013
6	441734	LONDON - TIUK	Energy Quote (Commercial)	ANDREW CLYDESDALE	0.00	2013
7	599039	LONDON - TIUK	Construction (Commercial)	AON BENFIELD B	486.33	2012
8	599042	LONDON - TIUK	Energy Quote (Commercial)	ANDREW CLYDESDALE	30.15	2013
9	599043	LONDON - TIUK	Energy Quote (Commercial)	ANDREW CLYDESDALE	1549.96	2013
10	599044	LONDON - TIUK	Energy Quote (Commercial)	ANDREW CLYDESDALE	216.00	2013
11	599045	LONDON - TIUK	Energy Quote (Commercial)	ANDREW CLYDESDALE	350.00	2014

Query executed successfully. | YITBARK\TARIKU (15.0 RTM) | YITBARK\HP (160) | TARDISDW_Pattern_1 | 00:00:00 | 366 rows

Report 2

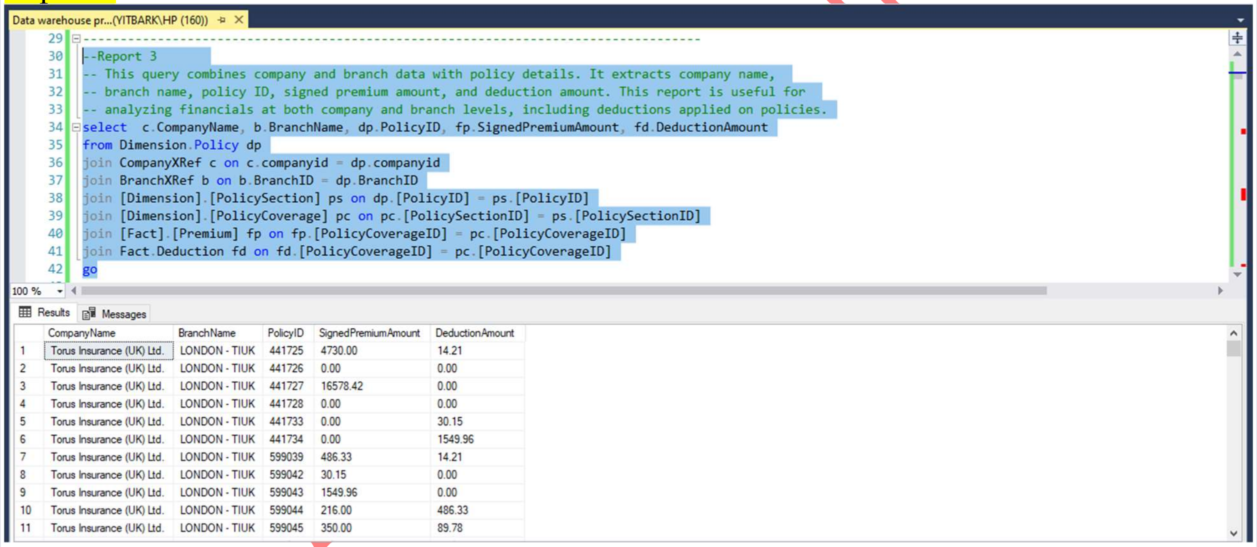


Data warehouse pr...(YITBARK\HP (160))

```sql
-------------------------------------------------------------------------------------------------
--Report 2
-- This query focuses on policy section and coverage details. It provides the policy ID, section title,
-- coverage title, and the signed premium amount. This report is essential for understanding the
-- specifics of policy coverage and the financial aspects related to different policy sections.
SET STATISTICS TIME ON
select dp.PolicyID, ps.SectionTitle, pc.CoverageTitle, fp.SignedPremiumAmount
from Dimension.Policy dp
join [Dimension].[PolicySection] ps on dp.[PolicyID] = ps.[PolicyID]
join [Dimension].[PolicyCoverage] pc on pc.[PolicySectionID] = ps.[PolicySectionID]
join [Fact].[Premium] fp on fp.[PolicyCoverageID] = pc.[PolicyCoverageID]
go
----------------------------------------------------------------------------------
--Report 3
```

	PolicyID	SectionTitle	CoverageTitle	SignedPremiumAmount
1	441725	Test	Test	4730.00
2	441726	Test	Test	0.00
3	441727	Faizan Manual	Faizan Manual	16578.42
4	441728	Test	Test	0.00
5	441733	TESTING	TESTING	0.00
6	441734	test	test	0.00
7	599039	B0576TP8368W	B0576TP8368W	486.33
8	599042	test	test	30.15
9	599043	test	test	1549.96
10	599044	test	test	216.00
11	599045	test	test	350.00

Report 3

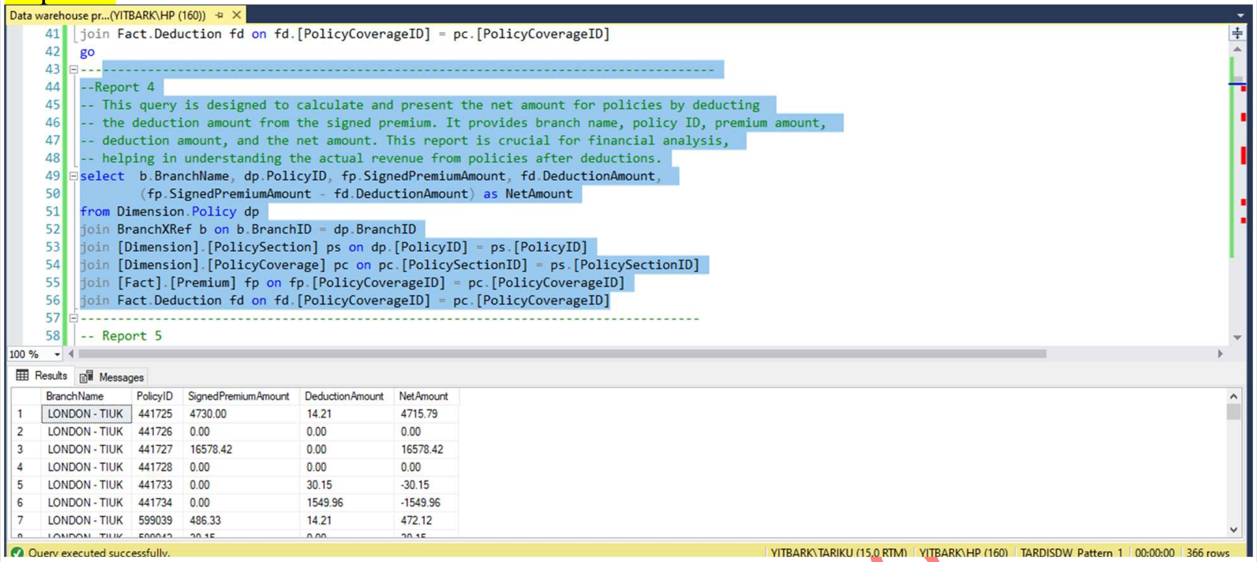


Data warehouse pr...(YITBARK\HP (160))

```sql
----------------------------------------------------------------------------------
--Report 3
-- This query combines company and branch data with policy details. It extracts company name,
-- branch name, policy ID, signed premium amount, and deduction amount. This report is useful for
-- analyzing financials at both company and branch levels, including deductions applied on policies.
select  c.CompanyName, b.BranchName, dp.PolicyID, fp.SignedPremiumAmount, fd.DeductionAmount
from Dimension.Policy dp
join CompanyXRef c on c.companyid = dp.companyid
join BranchXRef b on b.BranchID = dp.BranchID
join [Dimension].[PolicySection] ps on dp.[PolicyID] = ps.[PolicyID]
join [Dimension].[PolicyCoverage] pc on pc.[PolicySectionID] = ps.[PolicySectionID]
join [Fact].[Premium] fp on fp.[PolicyCoverageID] = pc.[PolicyCoverageID]
join Fact.Deduction fd on fd.[PolicyCoverageID] = pc.[PolicyCoverageID]
go
```

	CompanyName	BranchName	PolicyID	SignedPremiumAmount	DeductionAmount
1	Torus Insurance (UK) Ltd.	LONDON - TIUK	441725	4730.00	14.21
2	Torus Insurance (UK) Ltd.	LONDON - TIUK	441726	0.00	0.00
3	Torus Insurance (UK) Ltd.	LONDON - TIUK	441727	16578.42	0.00
4	Torus Insurance (UK) Ltd.	LONDON - TIUK	441728	0.00	0.00
5	Torus Insurance (UK) Ltd.	LONDON - TIUK	441733	0.00	30.15
6	Torus Insurance (UK) Ltd.	LONDON - TIUK	441734	0.00	1549.96
7	Torus Insurance (UK) Ltd.	LONDON - TIUK	599039	486.33	14.21
8	Torus Insurance (UK) Ltd.	LONDON - TIUK	599042	30.15	0.00
9	Torus Insurance (UK) Ltd.	LONDON - TIUK	599043	1549.96	0.00
10	Torus Insurance (UK) Ltd.	LONDON - TIUK	599044	216.00	486.33
11	Torus Insurance (UK) Ltd.	LONDON - TIUK	599045	350.00	89.78

Report 4



Data warehouse pr...(YITBARK\HP (160))

```sql
join Fact.Deduction fd on fd.[PolicyCoverageID] = pc.[PolicyCoverageID]
go
---------------------------------------------------------------------------------
--Report 4
-- This query is designed to calculate and present the net amount for policies by deducting
-- the deduction amount from the signed premium. It provides branch name, policy ID, premium amount,
-- deduction amount, and the net amount. This report is crucial for financial analysis,
-- helping in understanding the actual revenue from policies after deductions.
select  b.BranchName, dp.PolicyID, fp.SignedPremiumAmount, fd.DeductionAmount,
        (fp.SignedPremiumAmount - fd.DeductionAmount) as NetAmount
from Dimension.Policy dp
join BranchXRef b on b.BranchID = dp.BranchID
join [Dimension].[PolicySection] ps on dp.[PolicyID] = ps.[PolicyID]
join [Dimension].[PolicyCoverage] pc on pc.[PolicySectionID] = ps.[PolicySectionID]
join [Fact].[Premium] fp on fp.[PolicyCoverageID] = pc.[PolicyCoverageID]
join Fact.Deduction fd on fd.[PolicyCoverageID] = pc.[PolicyCoverageID]
---------------------------------------------------------------------------------
-- Report 5
```

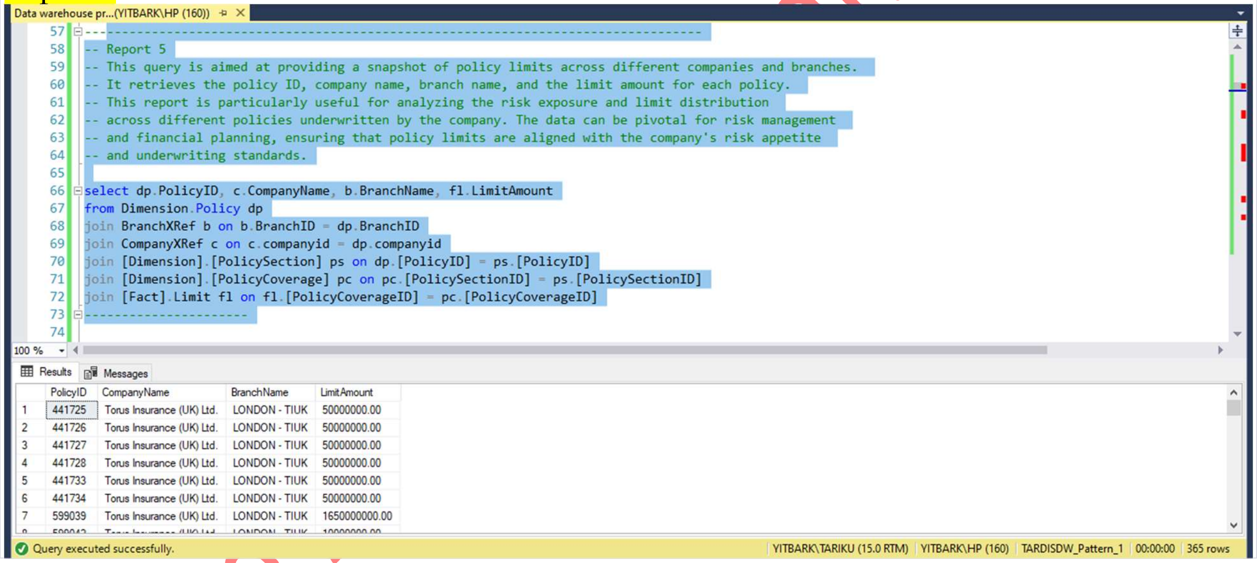
100 %

Results | Messages

	BranchName	PolicyID	SignedPremiumAmount	DeductionAmount	NetAmount
1	LONDON - TIUK	441725	4730.00	14.21	4715.79
2	LONDON - TIUK	441726	0.00	0.00	0.00
3	LONDON - TIUK	441727	16578.42	0.00	16578.42
4	LONDON - TIUK	441728	0.00	0.00	0.00
5	LONDON - TIUK	441733	0.00	30.15	-30.15
6	LONDON - TIUK	441734	0.00	1549.96	-1549.96
7	LONDON - TIUK	599039	486.33	14.21	472.12

Query executed successfully. | YITBARK\TARIKU (15.0 RTM) | YITBARK\HP (160) | TARDISDW_Pattern_1 | 00:00:00 | 366 rows

Report 5



Data warehouse pr...(YITBARK\HP (160))

```sql
---------------------------------------------------------------------------------
-- Report 5
-- This query is aimed at providing a snapshot of policy limits across different companies and branches.
-- It retrieves the policy ID, company name, branch name, and the limit amount for each policy.
-- This report is particularly useful for analyzing the risk exposure and limit distribution
-- across different policies underwritten by the company. The data can be pivotal for risk management
-- and financial planning, ensuring that policy limits are aligned with the company's risk appetite
-- and underwriting standards.

select dp.PolicyID, c.CompanyName, b.BranchName, fl.LimitAmount
from Dimension.Policy dp
join BranchXRef b on b.BranchID = dp.BranchID
join CompanyXRef c on c.companyid = dp.companyid
join [Dimension].[PolicySection] ps on dp.[PolicyID] = ps.[PolicyID]
join [Dimension].[PolicyCoverage] pc on pc.[PolicySectionID] = ps.[PolicySectionID]
join [Fact].Limit fl on fl.[PolicyCoverageID] = pc.[PolicyCoverageID]
---------------------

```

100 %

Results | Messages

	PolicyID	CompanyName	BranchName	LimitAmount
1	441725	Torus Insurance (UK) Ltd.	LONDON - TIUK	50000000.00
2	441726	Torus Insurance (UK) Ltd.	LONDON - TIUK	50000000.00
3	441727	Torus Insurance (UK) Ltd.	LONDON - TIUK	50000000.00
4	441728	Torus Insurance (UK) Ltd.	LONDON - TIUK	50000000.00
5	441733	Torus Insurance (UK) Ltd.	LONDON - TIUK	50000000.00
6	441734	Torus Insurance (UK) Ltd.	LONDON - TIUK	50000000.00
7	599039	Torus Insurance (UK) Ltd.	LONDON - TIUK	1650000000.00

Query executed successfully. | YITBARK\TARIKU (15.0 RTM) | YITBARK\HP (160) | TARDISDW_Pattern_1 | 00:00:00 | 365 rows

Results and Personal Growth

This project enhanced data processing speed and reporting accuracy and was a significant learning experience, reinforcing my understanding of data systems and their impact on decision-making.

Overcoming Challenges: A Learning Curve

Managing large volumes of diverse data was challenging. I tackled this by implementing scalable ETL packages and optimizing database queries, which was a testament to the project's learning curve.

Conclusion

This personal project in developing the "Integrated ETL and Reporting System" was an enriching journey, underscoring the transformative power of effectively managed data. It stands as a significant milestone in my exploration of data science and analytics.