

Sri Lanka Institute of Information Technology



**Assignment 1 - Design, ETL Development, and Implementation
of a Data Warehouse**

Student Name: Dayana Priyadharshani Kumar

Student ID: IT22178640

Batch: Y3.S1.WE.DS.01

Module: Data Warehousing and Business Intelligence

Module Code: IT3021

Document Type: Assignment 1 – DWBI Solution Development

Date of Submission: 01/05/2025

Contents

STEP 01 - Data Set Selection	3
ER-Diagram	4
STEP 02 - Preparation of Data Sources	5
STEP 03 - Solution Architecture.....	7
STEP 04 - Data Warehouse Design & Development.....	9
STEP 05 - ETL Development	16
STEP 06 - ETL Development – Accumulating Fact Table	44
Summary	47

STEP 01 - Data Set Selection

I originally sourced this dataset from Kaggle (original file contains ~7,000 customer records with 21 unique columns, link is attached below) and then enriched and extended it to 10,000 unique customer snapshots spanning two calendar years (2021–2022). It represents an operational (OLTP) view of a real-world telcom business, capturing monthly billing snapshots for every subscriber on a daily cadence and reflecting live customer activities.

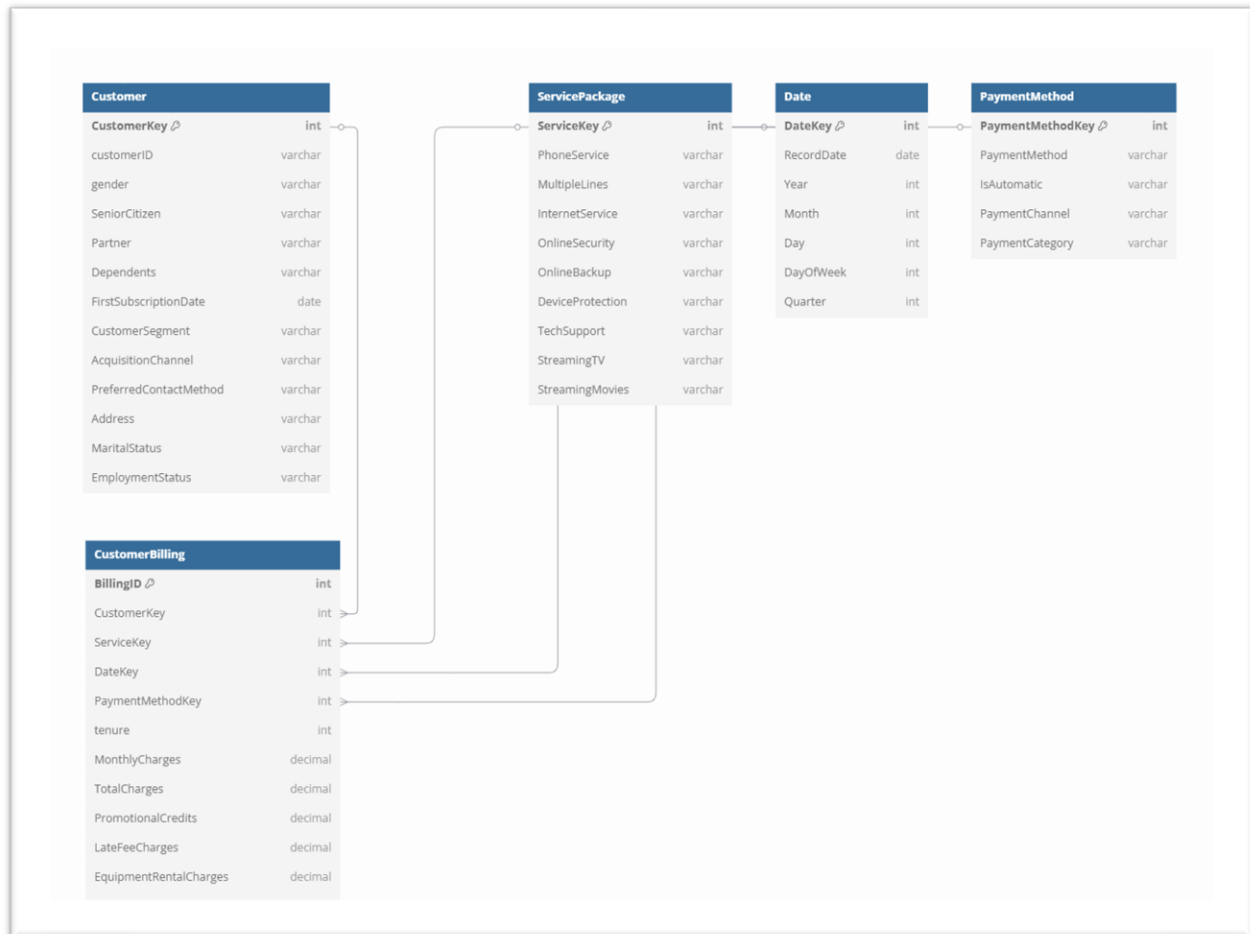
The dataset provides detailed information across several areas, including customer demographics (gender, senior citizen status, partner and dependent status, marital status, employment status), service subscriptions (voice, internet, online security, online backup, device protection, tech support, streaming TV, and streaming movies), and monthly account details (tenure, contract type, payment method, paperless billing status, monthly charges, total charges, and promotional credits). A churn flag is also included, indicating whether a customer left within the last month, serving as the primary target for retention analysis.

The source dataset is organised into multiple logical entities representing customer profiles, service packages, payment methods, billing activities, and time snapshots. These entities are interrelated to reflect the real-world business processes of customer management, service utilisation, and billing cycles.

Download link: <https://www.kaggle.com/datasets/blastchar/telco-customer-churn>

ER-Diagram

The ER diagram below represents the structure of the source system used for building the data warehouse. It shows how customer details, service subscriptions, payment methods, and daily snapshots are captured and linked through billing records, forming the foundation for further ETL processing and warehouse design.



STEP 02 - Preparation of Data Sources

The original dataset sourced from Kaggle contained both customer details and service subscription information combined within a single CSV file. To better align with data warehouse best practices and simulate a real-world ETL process, I divided the data into multiple logical entities based on their business meaning and purpose.

Specifically into 03 data sources:

- Customer demographic and account information was separated into a **Customer.csv** file.
- Service subscription details (phone, internet, streaming services) were extracted into a **ServicePackage.txt** file, formatted as a **tab-delimited text file** to introduce variation in file formats.
- An **Excel sheet (XLSX)**: customer_satisfaction_data.xlsx

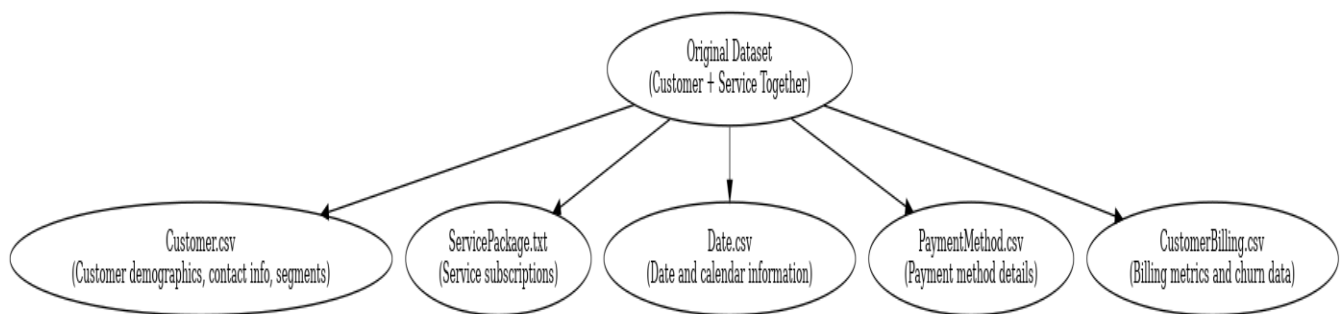
Using both **CSV**, **tab-delimited TXT**, **excel sheet (XLSX)** formats enables the ETL process to handle multiple file types, reflecting real-world operational data integration challenges where data often arrives from heterogeneous sources.

The following files, formats, and stored information are summarized below to provide a complete view of how the original dataset has been organized and structured for the ETL process:

- **Customer.csv** (CSV Format)
 - Contains demographic and personal information about customers, including gender, senior citizen status, partner and dependents status, customer segment, preferred contact method, marital status, employment status, address, and subscription start date.
- **Date.csv** (CSV Format)
 - Provides a full two-year daily calendar with attributes such as RecordDate, Year, Month, Day, DayOfWeek, and Quarter.
- **PaymentMethod.csv** (CSV Format)
 - Stores payment method details, indicating how customers prefer to pay (electronic check, mailed check, bank transfer, credit card) along with derived

fields like whether payments are automatic, the payment channel (online/offline), and payment category.

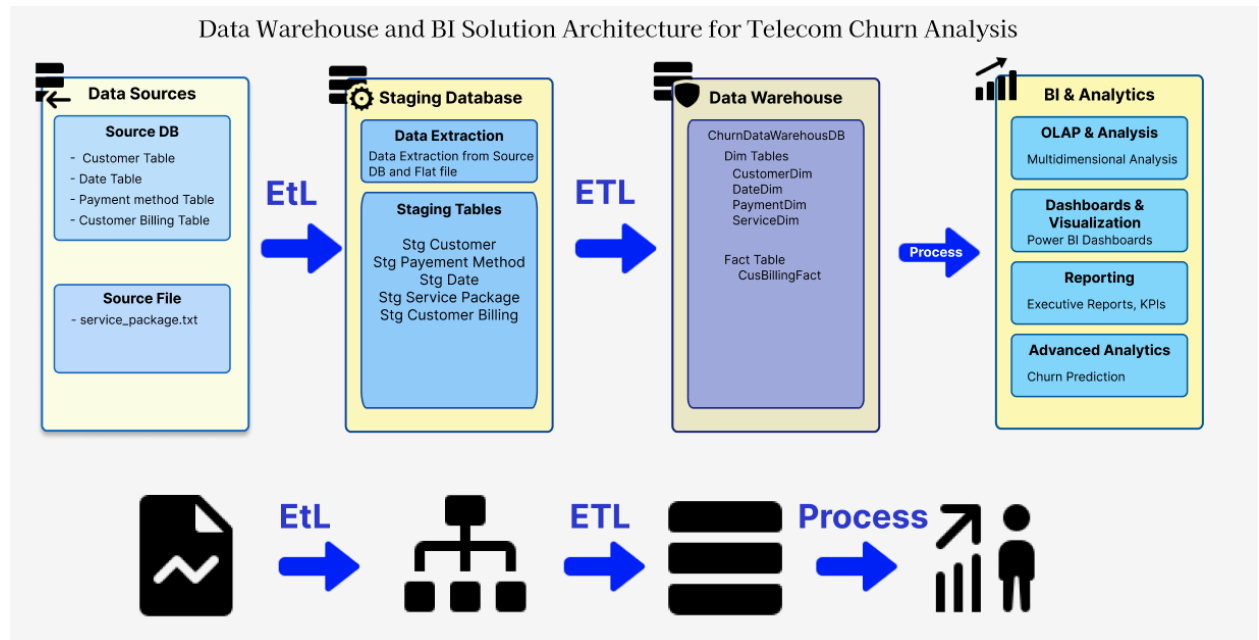
- **CustomerBilling.csv** (CSV Format)
 - Holds monthly billing and financial information for each customer, including tenure, monthly charges, total charges, promotional credits, late fees, equipment rental charges, and churn status.
- **ServicePackage.txt** (Tab-Delimited Text Format)
 - Describes service subscription details, including whether customers have phone services, multiple lines, internet services (DSL/Fiber/No service), and additional services like online security, backup, device protection, tech support, streaming TV, and movies.



Before the ETL process, file consistency checks were performed to ensure that delimiters were correctly applied, column headers matched the expected schema, and data types were compatible for loading into the staging environment. Minor cleaning steps, such as verifying date formats and handling any missing values, were also conducted to ensure smooth and accurate data ingestion.

This structured splitting of data not only supports smoother dimension and fact table creation during the Data Warehouse build but also satisfies the assignment's requirement to have sufficient and appropriately organized data to demonstrate key Data Warehouse (DW) concepts, such as: Dimensional modeling (fact and dimension tables), Slowly Changing Dimensions (SCD) handling and Accumulating fact updates.

STEP 03 - Solution Architecture



Data Sources

The data sources consist of structured CSV files (Customer, Date, PaymentMethod, and CustomerBilling) which were imported to a single source database called **ChurnAnalysisSourceDB** and a **tab-delimited text file** (ServicePackage) and **excel sheet** (customer_satisfaction_data.xlsx). These data sources represent the operational source systems capturing customer demographics, service subscriptions, billing transactions, and payment behaviors.

ETL Process

The ETL process, built using **SSIS packages**, handles four major steps:

- **Data Extraction:** Importing files from a Source database (**ChurnAnalysisSourceDB**) and Excel Sheet called customer_satisfaction_data.xlsx and a Text file for ServicePackage to staging database.
- **Data Transformation:** Cleansing and standardizing data (handling missing values, formatting dates, creating surrogate keys, creating derived columns, lookups).
- **Data Loading:** Performing incremental loads into staging and then into the final warehouse tables.

Data Warehouse

- Data Warehouse (ChurnDataWarehouseDB): Final schema consisting of one central fact table (CusBillingFact) and four supporting dimension tables (CustomerDim, DateDim, PaymentDim, ServiceDim) organized in a **star schema**. This design optimizes analytical querying, aggregation, and reporting

BI & Analytics Layer

The BI layer is intended to connect Power BI dashboards and reporting tools to the data warehouse. It will enable basic visualizations, churn trend monitoring, and executive reporting, supporting future expansion into more advanced analytics if needed.

STEP 04 - Data Warehouse Design & Development

1. Database Design Overview

To deploy the data warehouse solution for the telcom customer churn analysis, I created three separate databases in SQL Server:

- **ChurnAnalysisSourceDB:**

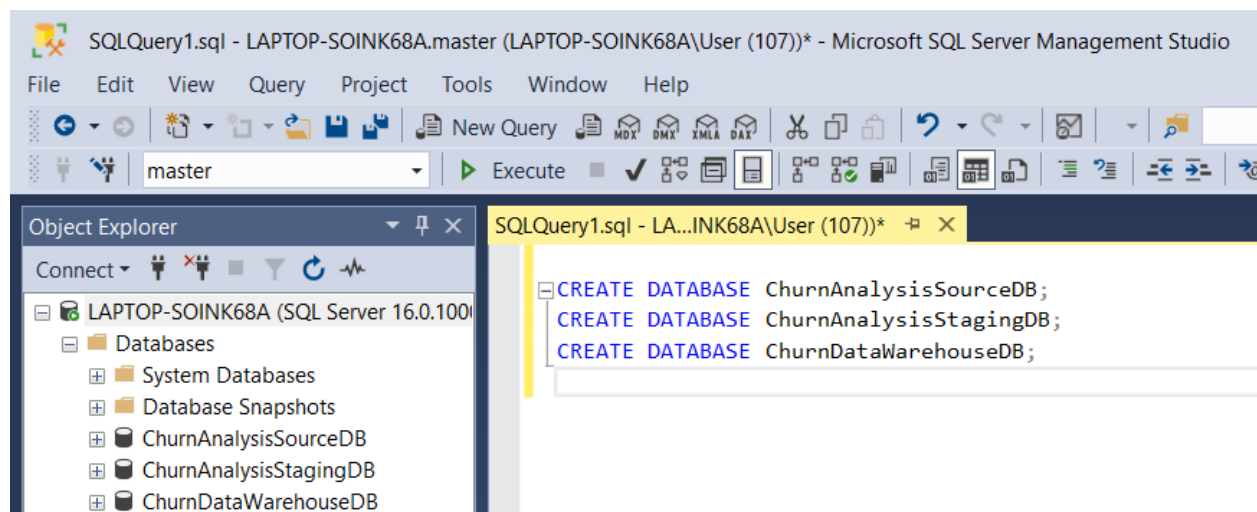
This database stores the initial raw data imported from CSV and tab-delimited TXT source files. It simulates the original operational data environment and holds the untransformed customer, service, billing, and payment records.

- **ChurnAnalysisStagingDB:**

This staging database serves as the intermediary layer where data cleansing, basic transformations, standardization, and validation are performed. It ensures data quality before final loading into the warehouse.

- **ChurnDataWarehouseDB:**

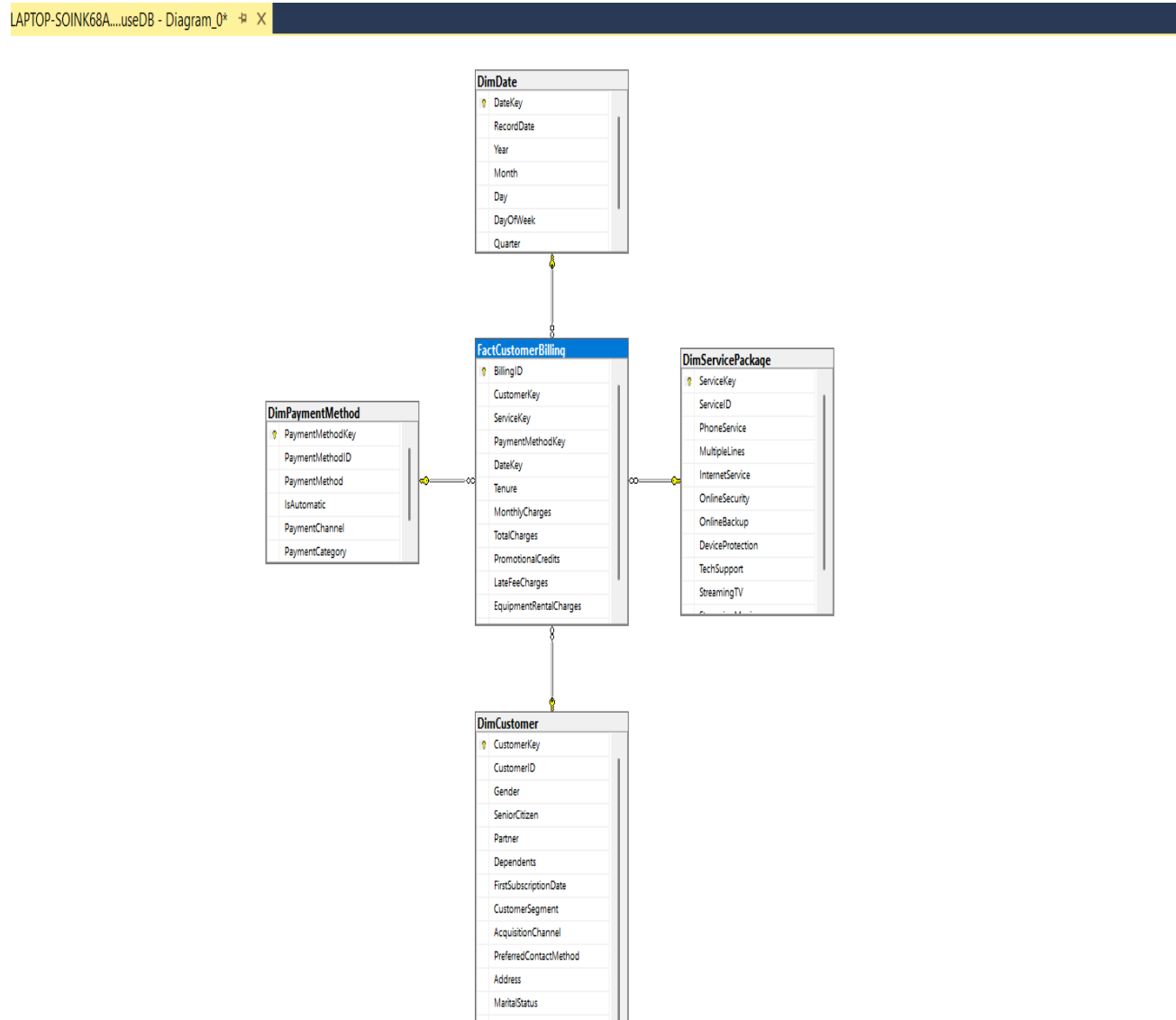
This is the final target data warehouse database designed following a star schema model. It contains cleaned, transformed, and fully relational fact and dimension tables. The fact table captures billing transactions, and the dimension tables represent customers, subscribed services, payment methods, and date hierarchies.



2. Data Warehouse Design Overview

The Data Warehouse schema was designed using a **Star Schema model**, which allows for optimized querying, simplified joins, and clear separation between facts and dimensions. The model centers around a single fact table (FactCustomerBilling) and is supported by four dimension tables: DimCustomer, DimServicePackage, DimPaymentMethod, and DimDate. This structure enables efficient analysis of telecom customer behavior, billing trends, and churn patterns over time.

Data Warehouse Database Diagram

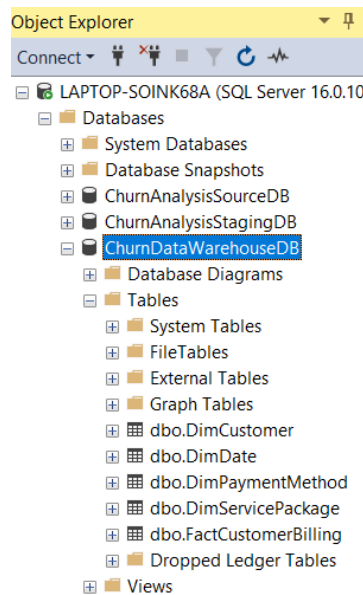


Dimensional Model- Star Schema Structure

At the core of the design is the FactCustomerBilling table, which stores quantitative measures (charges, tenure, churn, etc.) and references dimension tables using surrogate keys. Each dimension table provides descriptive context to enrich fact data and support flexible reporting.

The dimension tables include:

- DimCustomer – contains detailed customer information and acts as a **Slowly Changing Dimension (Type 2)**
- DimServicePackage – stores the service subscription details
- DimPaymentMethod – defines how the customer pays
- DimDate – provides full calendar context for 2 years



Slowly Changing Dimension: DimCustomer

The DimCustomer table has been modeled as a **Slowly Changing Dimension (SCD)** to track evolving customer attributes over time. This structure enables historical analysis and helps understand how customer characteristics impact behavior such as churn. Certain customer attributes change gradually over time, not on a daily or transactional basis. Fields such as:

- CustomerSegment
- PreferredContactMethod
- MaritalStatus
- EmploymentStatus are dynamic and may shift due to life events or changing preferences.

These changes do not overwrite past data in the source systems but are important for temporal analysis in the warehouse.

Table Descriptions & Keys

1. FactCustomerBilling

- **Primary Key:** BillingID (business key)
- **Foreign Keys:** CustomerKey, ServiceKey, PaymentMethodKey, DateKey
- Stores measures such as: MonthlyCharges, TotalCharges, PromotionalCredits, and Churn

```
SQLQuery1.sql - LA...INK68A\User (157))* -# X
CREATE TABLE FactCustomerBilling (
    BillingID VARCHAR(50) PRIMARY KEY, -- Business Key
    CustomerKey INT,
    ServiceKey INT,
    PaymentMethodKey INT,
    DateKey INT,
    Tenure INT,
    MonthlyCharges FLOAT,
    TotalCharges FLOAT,
    PromotionalCredits FLOAT,
    LateFeeCharges FLOAT,
    EquipmentRentalCharges FLOAT,
    Churn VARCHAR(10),

    FOREIGN KEY (CustomerKey) REFERENCES DimCustomer(CustomerKey),
    FOREIGN KEY (ServiceKey) REFERENCES DimServicePackage(ServiceKey),
    FOREIGN KEY (PaymentMethodKey) REFERENCES DimPaymentMethod(PaymentMethodKey),
    FOREIGN KEY (DateKey) REFERENCES DimDate(DateKey)
);
```

2. DimCustomer

- **Surrogate Key:** CustomerKey (INT IDENTITY)
- **Alternate Key:** CustomerID (business ID from source)
- Attributes: Gender, SeniorCitizen, Partner, CustomerSegment, etc.
- Includes Customer satisfaction table attributes which were merged during transformation in ETL.
- **Slowly Changing Dimension** – allows tracking changes in fields like CustomerSegment, PreferredContactMethod

```
SQLQuery1.sql - LA...INK68A\User (157))*  X
CREATE TABLE DimCustomer (
    CustomerKey INT IDENTITY(1,1) PRIMARY KEY,
    CustomerID VARCHAR(50), -- Business Key
    Gender VARCHAR(10),
    SeniorCitizen VARCHAR(10),
    Partner VARCHAR(10),
    Dependents VARCHAR(10),
    FirstSubscriptionDate DATE,
    CustomerSegment VARCHAR(50),
    AcquisitionChannel VARCHAR(50),
    PreferredContactMethod VARCHAR(50),
    Address VARCHAR(255),
    MaritalStatus VARCHAR(50),
    EmploymentStatus VARCHAR(50)
);
```

3. DimServicePackage

- **Surrogate Key:** ServiceKey
- **Alternate Key:** ServiceID
- Describes the services customers subscribe to (e.g., streaming, internet, security)

```
SQLQuery1.sql - LA...INK68A\User (157))* ✕  
  
CREATE TABLE DimServicePackage (  
    ServiceKey INT IDENTITY(1,1) PRIMARY KEY,  
    ServiceID VARCHAR(50), -- Business Key  
    PhoneService VARCHAR(20),  
    MultipleLines VARCHAR(20),  
    InternetService VARCHAR(20),  
    OnlineSecurity VARCHAR(20),  
    OnlineBackup VARCHAR(20),  
    DeviceProtection VARCHAR(20),  
    TechSupport VARCHAR(20),  
    StreamingTV VARCHAR(20),  
    StreamingMovies VARCHAR(20)  
);
```

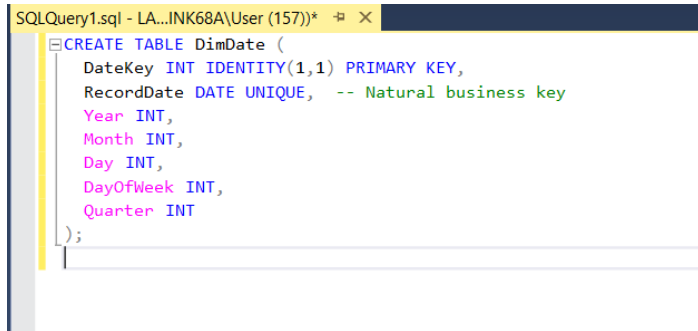
4. DimPaymentMethod

- **Surrogate Key:** PaymentMethodKey
- **Alternate Key:** PaymentMethodID
- Captures payment types and behavior (e.g., IsAutomatic, Channel)

```
SQLQuery1.sql - LA...INK68A\User (157))* ✕  
  
CREATE TABLE DimPaymentMethod (  
    PaymentMethodKey INT IDENTITY(1,1) PRIMARY KEY,  
    PaymentMethodID VARCHAR(50), -- Business Key  
    PaymentMethod VARCHAR(50),  
    IsAutomatic VARCHAR(10),  
    PaymentChannel VARCHAR(20),  
    PaymentCategory VARCHAR(20)  
);
```

5. DimDate

- **Surrogate Key:** DateKey
- **Alternate/Natural Key:** RecordDate
- Includes Year, Month, Quarter, DayOfWeek



```
SQLQuery1.sql - LA...INK68A\User (157)*  [  ] X
CREATE TABLE DimDate (
    DateKey INT IDENTITY(1,1) PRIMARY KEY,
    RecordDate DATE UNIQUE, -- Natural business key
    Year INT,
    Month INT,
    Day INT,
    DayOfWeek INT,
    Quarter INT
);
```

Assumptions Made

- Each table contains cleaned and transformed data from the staging layer
- Surrogate keys were added using INT IDENTITY to maintain uniqueness and optimize joins
- Alternate keys (original business IDs) are retained for traceability
- The DimCustomer table is considered a Slowly Changing Dimension (SCD), tracking changes in non-static fields
- The FactCustomerBilling uses existing business IDs (BillingID) while referencing surrogate keys from dimensions

Implementation

- All tables were created inside the ChurnDataWarehouseDB database using SQL Server Management Studio (SSMS)
- Dimensions were created first, followed by the fact table with appropriate foreign key constraints
- Surrogate keys were generated using INT IDENTITY(1,1) columns
- Business keys were stored as VARCHAR fields for external reference

STEP 05 - ETL Development

Data Extraction

The Data Extraction phase was implemented by designing a structured ETL pipeline using **Visual Studio (SSIS)**. The primary goal of this stage was to extract data from multiple types of source systems and safely load them into the Staging Database for further transformation.

I selected Three different types of data sources:

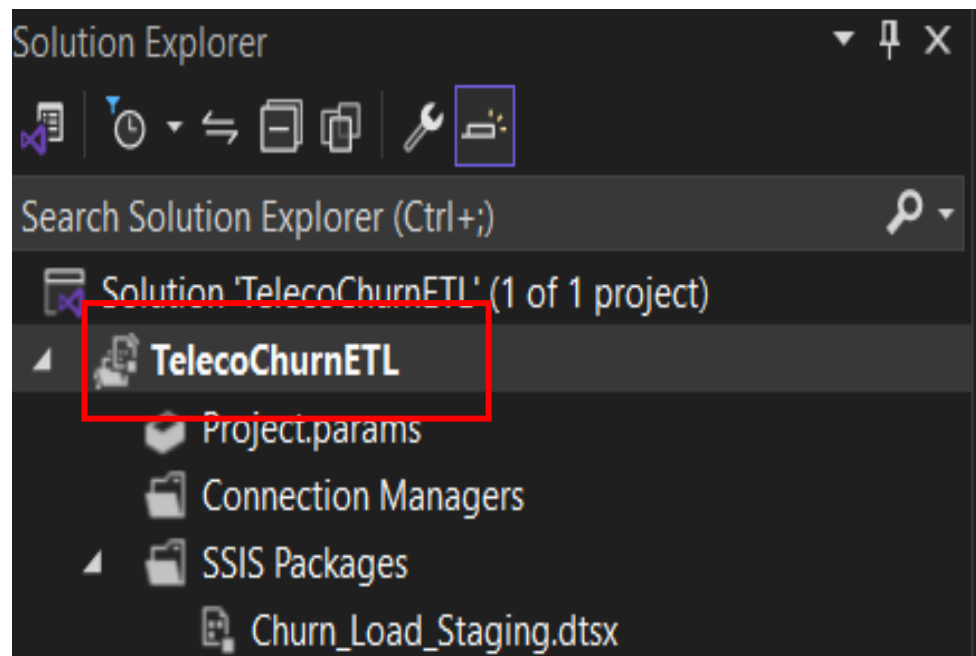
- A **SQL Server Database**: ChurnAnalysisSourceDB (Contains 4 tables with data)
- A **Flat File (TXT)**: service_package.txt (Tab-delimited text file)
- An Excel sheet (XLSX): customer_satisfaction_data.xlsx

This allowed the ETL process to demonstrate the capability of handling multiple data formats within the same extraction pipeline.

Steps Followed in Data Extraction Implementation:

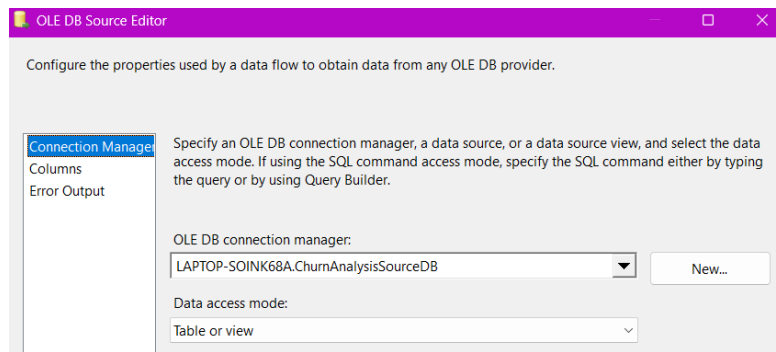
1. Project Setup:

Created a new **SSIS Project** in Visual Studio named TelecoChurnETL.

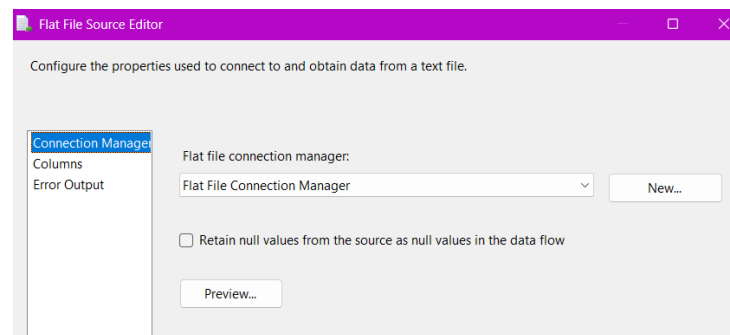


Added appropriate **Connection Managers**:

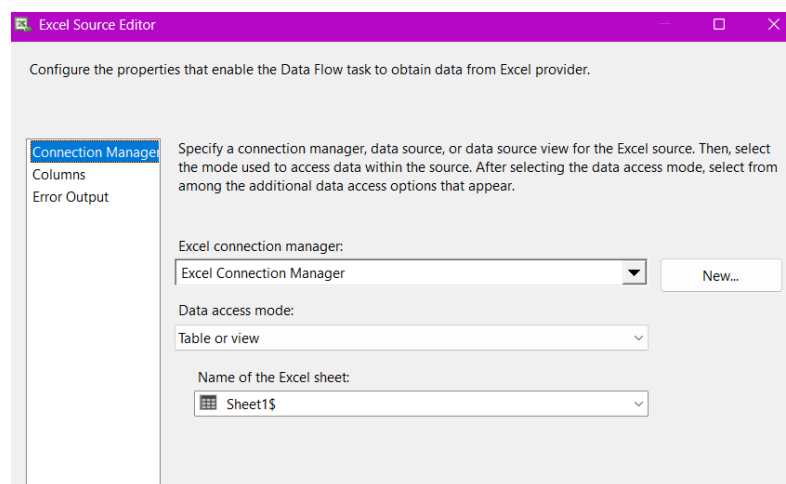
- OLE DB Connection for ChurnAnalysisSourceDB (SQL Database).



- Flat File Connection Manager for service_package.txt (Text file with Tab delimiter)



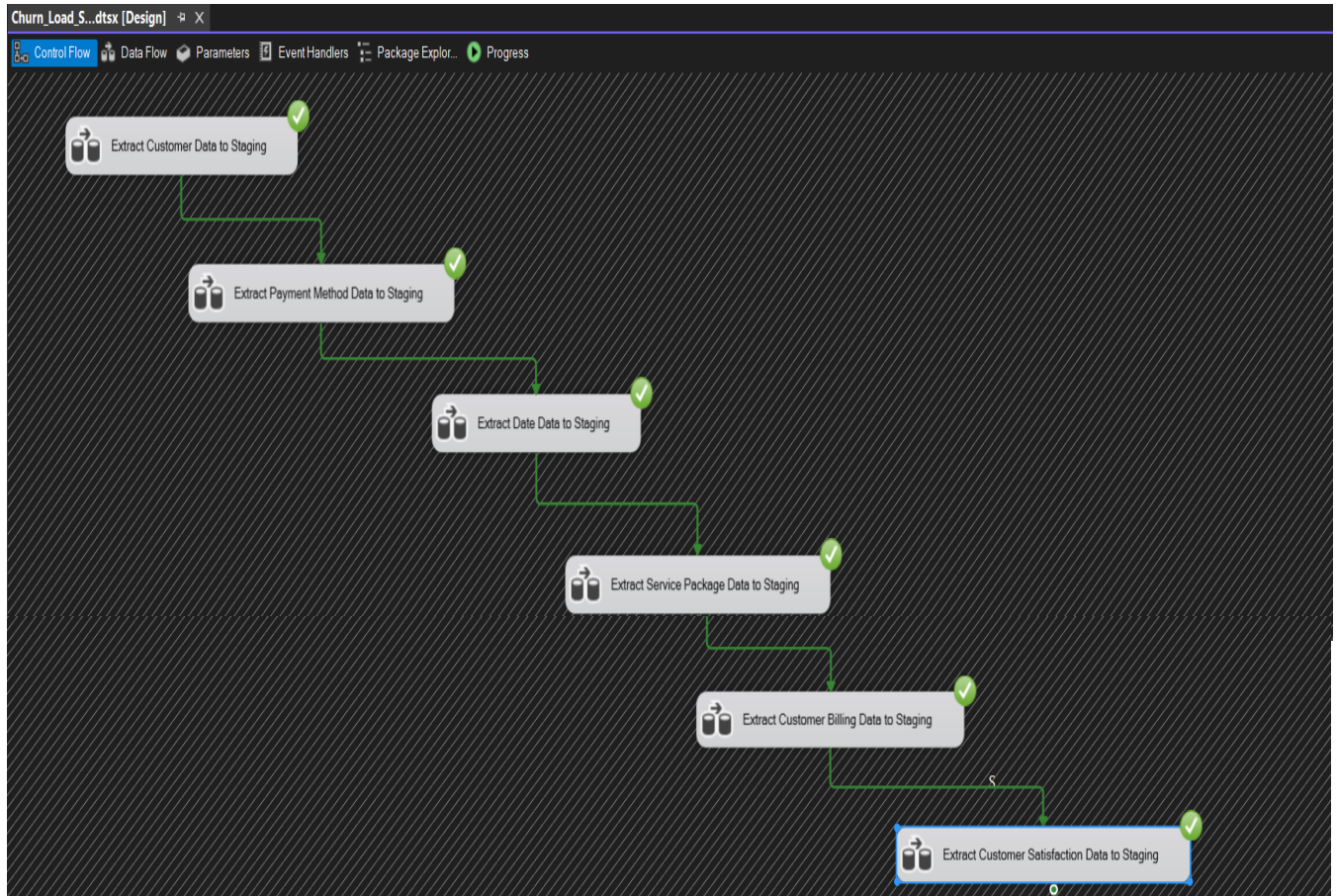
- Excel File Connection Manager for Customer Satisfaction.xlsx file.



2. Design of Control Flow:

Added separate Data Flow Tasks for each extraction process:

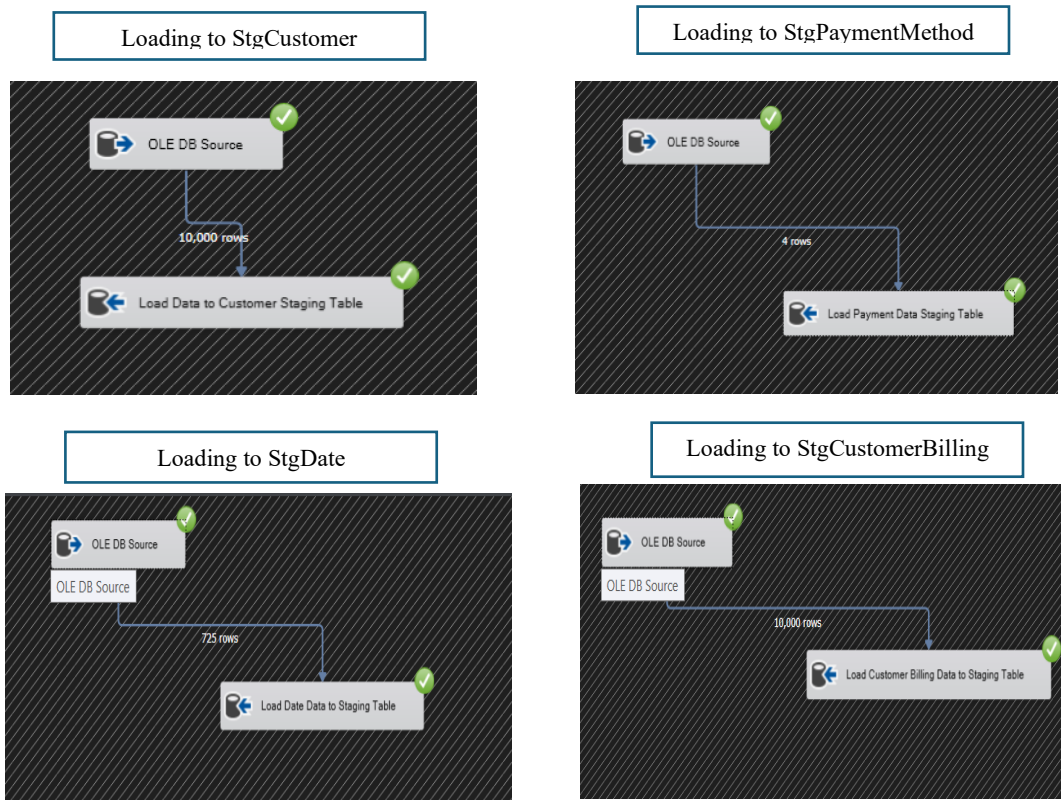
- ❖ Extracted Customer Data, Extract Payment Method, Extract Date, Extract Customer Billing data (from Source DB)
- ❖ Extracted Customer Satisfaction Data from Excel sheet.
- ❖ Extracted Service Package (from Text File).



Data Flow Configuration:

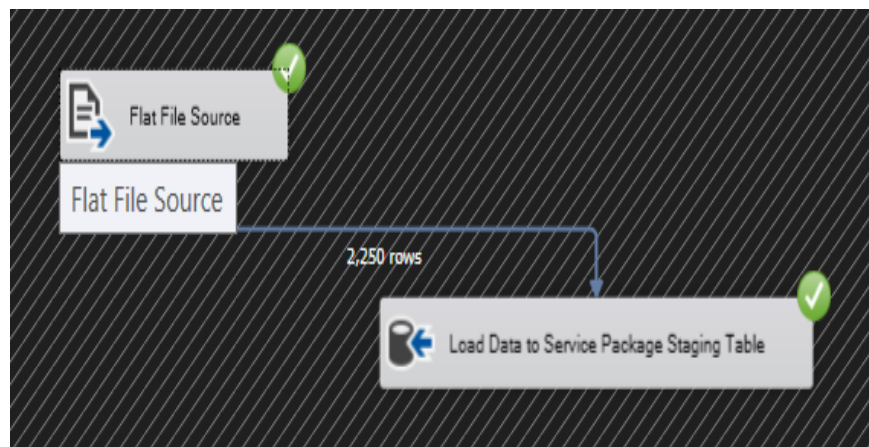
For database tables (Customer, PaymentMethod, Date, Customer Billing):

❖ Used **OLE DB Source** components to extract data from the SQL

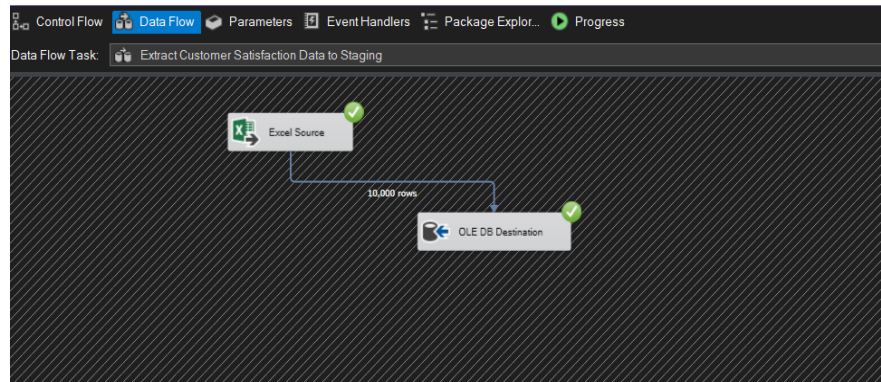


❖ For the flat file (ServicePackage):

- Used a **Flat File Source** component configured with **Tab delimiter** settings.

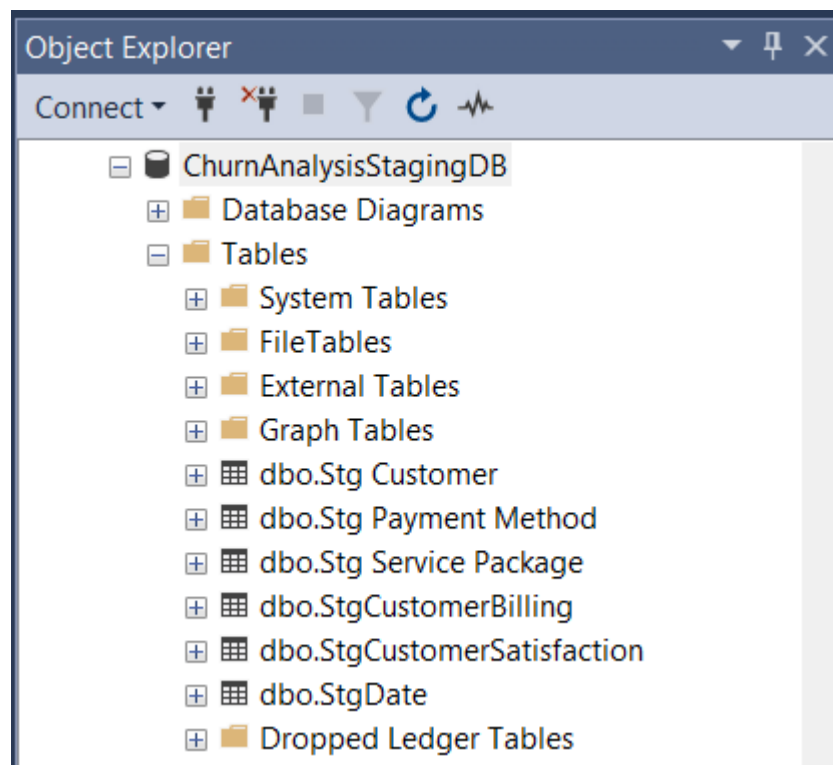


❖ For Excel Sheet



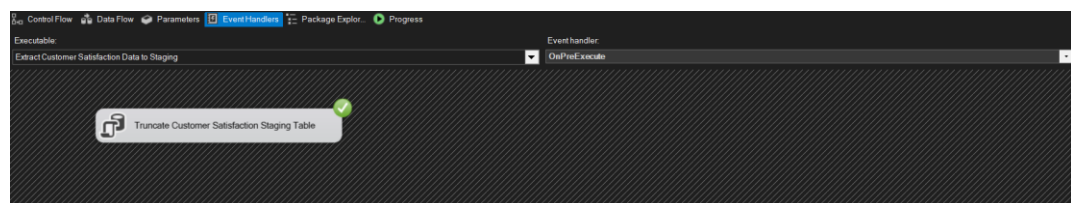
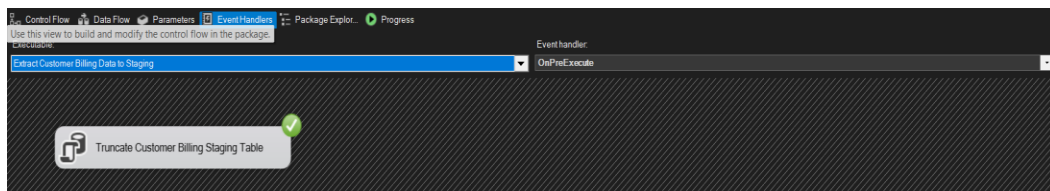
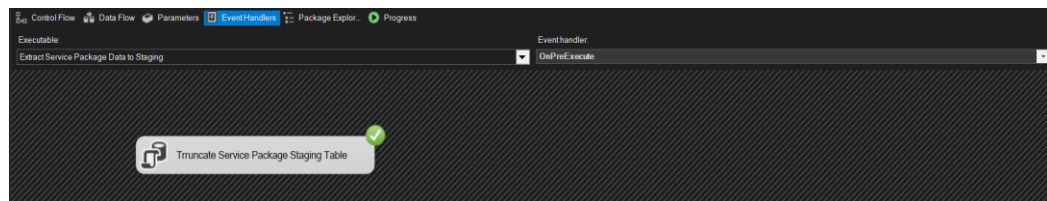
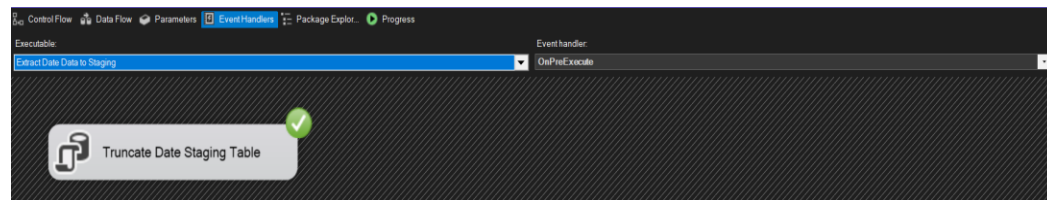
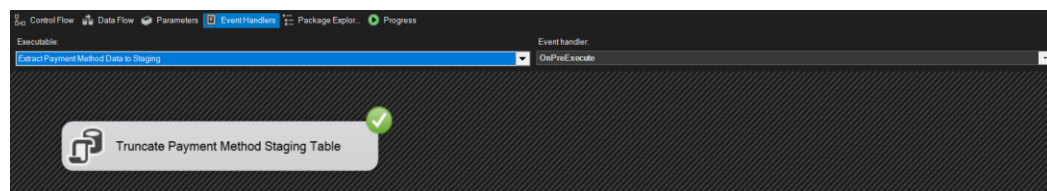
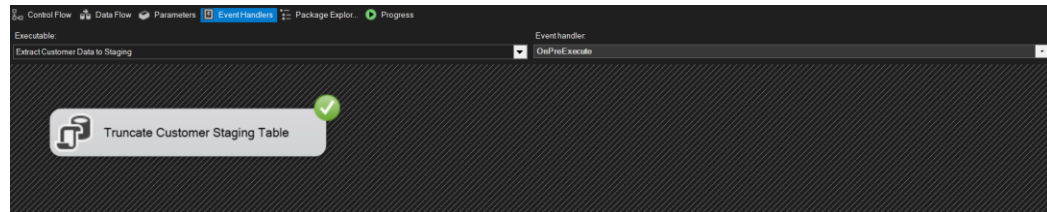
2. Loading into Staging:

- ❖ Each extracted dataset was loaded into corresponding tables in the Staging Database (ChurnAnalysisStagingDB) using OLE DB Destination components.
- ❖ Tables created in staging followed a clear naming convention (Stg Customer, Stg Payment Method, etc.)

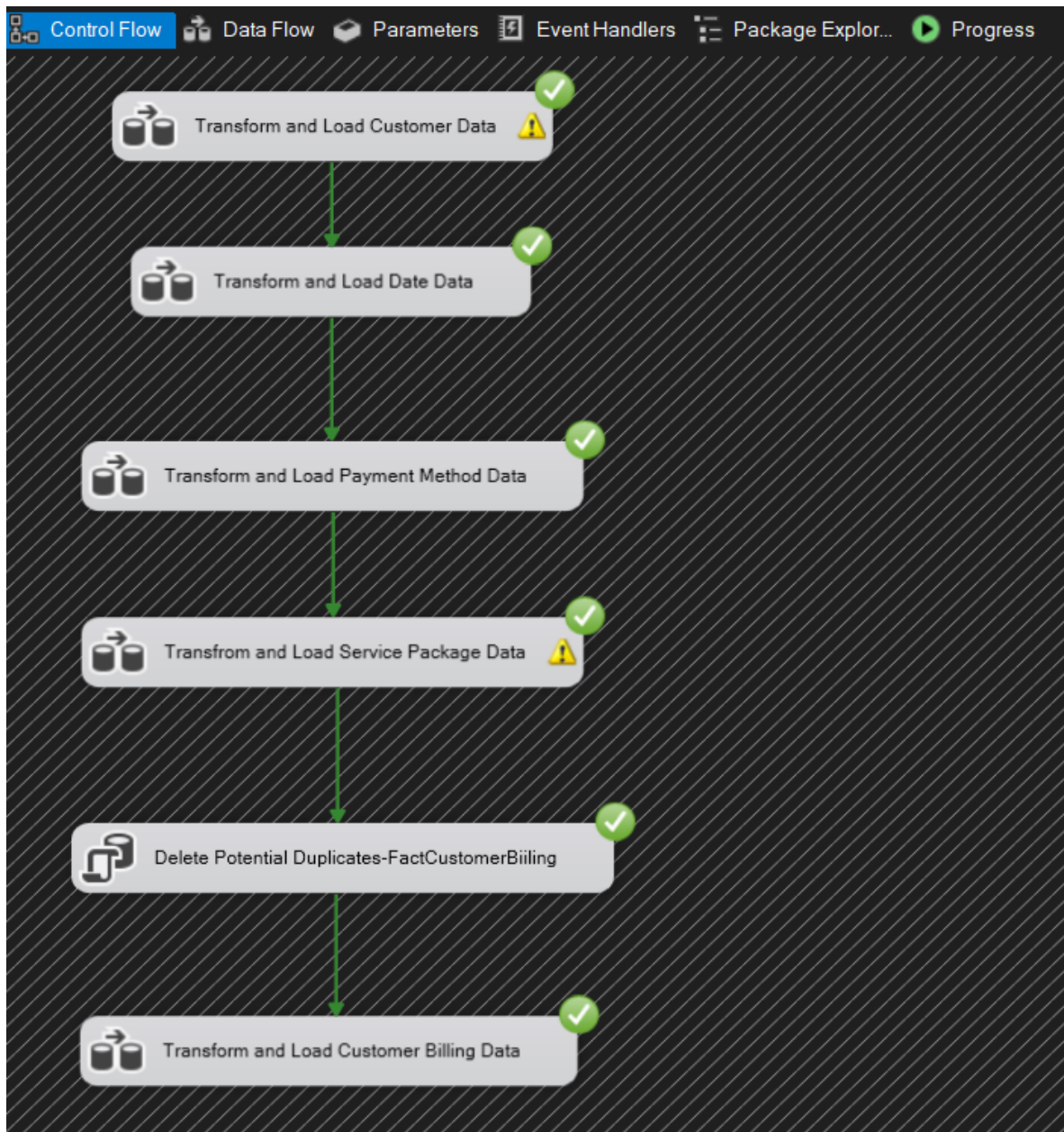


3. Event Handler Setup:

- Configured an **OnPreExecute** event handler for the Data Flow Tasks.
- Purpose: To ensure safe clearing (truncation) of staging tables before each load, avoiding duplication when running the ETL multiple times.
- Below images shows all the event handlers I have created for my project:

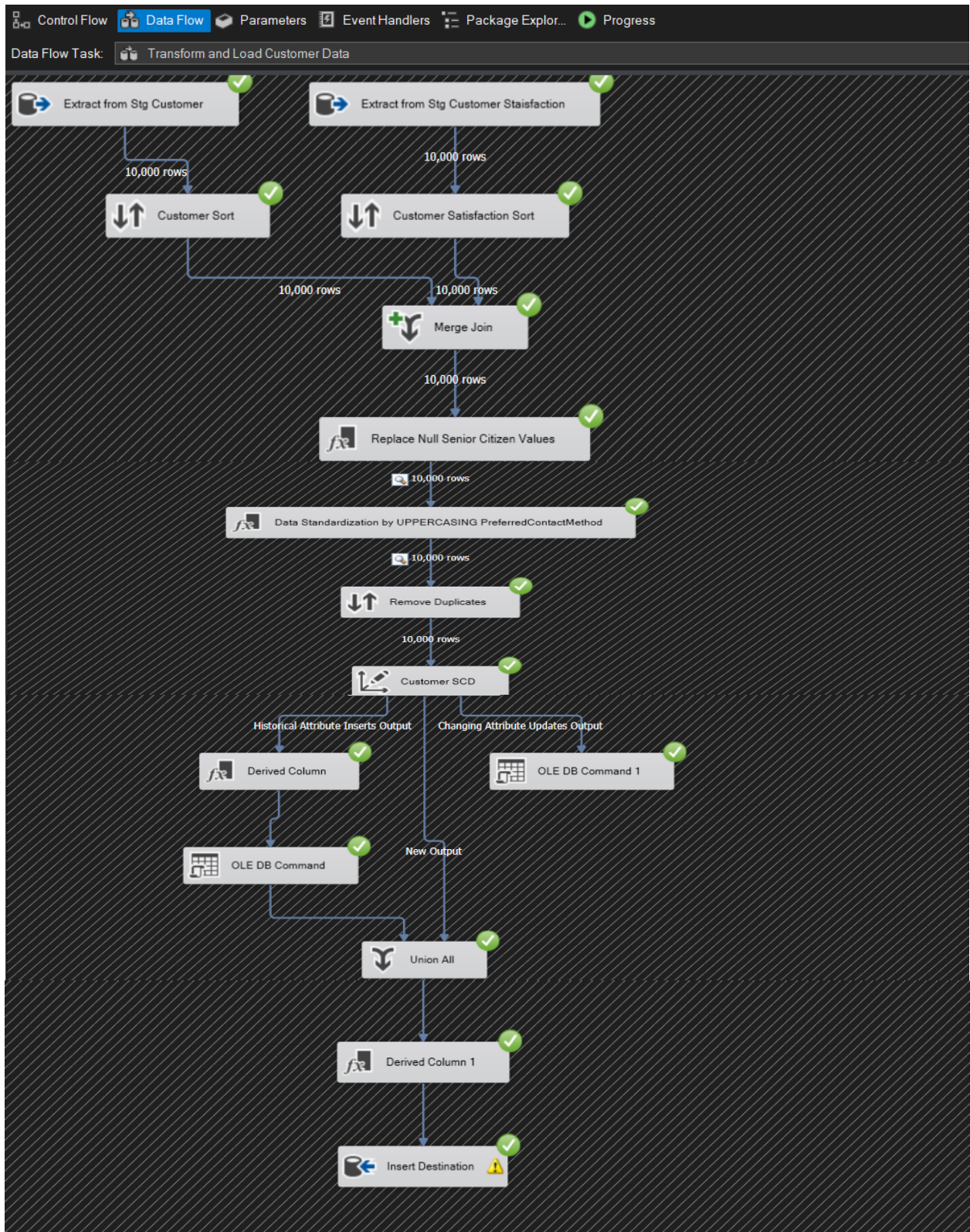


Control Flow Diagram with Data Flow Tasks for Transformation and Loading to Data Warehouse



Each section below describes the individual ETL tasks involved in the transformation and loading process. Each task has its own dedicated data flow configuration, with accompanying screenshots to illustrate the setup and execution.

ETL – Transformation and Load Process for DimCustomer Table



1. Source Configuration

- The process begins by extracting customer data and satisfaction data from two separate staging sources: StgCustomer and StgCustomerSatisfaction.
- An OLE DB Source was configured to connect to both StgCustomer and StgCustomerSatisfaction staging tables, pulling relevant columns such as CustomerID, CustomerName, SeniorCitizen, PreferredContactMethod, and SatisfactionScore.
- Each dataset was then sorted by CustomerID, ensuring that both datasets are aligned for merging.

2. Merge Join Transformation

- After sorting, a Merge Join Transformation was used to combine the two datasets (StgCustomer and StgCustomerSatisfaction) based on the CustomerID field.
- SSIS requires that both inputs be sorted for a Merge Join to work, which is why both staging tables went through the Sort Transformation prior to the Merge Join.
- The Join type used in the Merge Join was an Inner Join, combining only matching records from both datasets into a single stream of customer data.

3. Transformation Tasks After Merge

After merging the datasets, the following transformation tasks were performed to clean, standardize, and process the data before loading it into the **DimCustomer** table:

3.1 Data Cleaning – Null Handling

- **Task:** Derived Column transformation
- **Purpose:**
 - ✓ Replaced all NULL values in the SeniorCitizen column with 'N'.
 - ✓ Ensured consistent representation of categorical data across records (i.e., 'Y' or 'N' for SeniorCitizen).

- ✓ Prepared the data for downstream validation and analysis by ensuring no NULL values in critical columns.

3.2 Data Standardization

- **Task:** Derived Column transformation
- **Purpose:**
 - ✓ Standardized the PreferredContactMethod column by converting all values to UPPERCASE.
 - ✓ Ensured consistent formatting of communication channels (e.g., converting "email" to "EMAIL", "phone" to "PHONE").
 - ✓ Reduced data mismatches due to case sensitivity, particularly in queries and reporting.

3.3 Duplicate Removal

- **Task:** Sort transformation with Remove Duplicates enabled
- **Purpose:**
 - ✓ Identified and eliminated duplicate customer records based on CustomerID and other related fields.
 - ✓ Ensured that only unique customer entries were processed, preventing any redundancy in the DimCustomer table.
 - ✓ Maintained data integrity by ensuring no duplicate entries were loaded.

3.4 Slowly Changing Dimension (SCD) Management

- **Task:** Slowly Changing Dimension (SCD) transformation
- **Purpose:**
 - ✓ Managed changes in dimension attributes using Type 1 (overwrite) and Type 2 (history tracking) handling.
 - ✓ Historical Attribute Inserts Output: Inserted new versions of records to track historical changes (SCD Type 2).
 - ✓ Changing Attribute Updates Output: Applied in-place updates to records with non-historical changes (SCD Type 1).

- ✓ New Output: Captured and inserted brand-new customer records that didn't exist in the DimCustomer table.

3.5 Additional Data Transformation

- **Tasks:**
 - ✓ Derived Column transformation: Added fields such as InsertDate, StartDate, and ModifiedDate using expressions like GETDATE().
 - ✓ OLE DB Command: Executed an UPDATE SQL command to apply Type 1 updates to existing records in the DimCustomer table.
- **Purpose:**
 - ✓ Enhanced the completeness of customer records by adding important metadata fields like timestamps for insert and update actions.
 - ✓ Allowed for effective ETL-level metadata tracking, keeping track of when records were inserted or modified.

3.6 Data Flow Consolidation

- **Task:** Union All + Final Derived Column
- **Purpose:**
 - ✓ Merged the three SCD outputs (insert, update, new) into a single consolidated pipeline.
 - ✓ Applied final touch-ups via Derived Column transformations (e.g., setting system columns like InsertDate or ModifiedDate).
 - ✓ Streamlined the flow into a single unified structure before the data was loaded into the DimCustomer table.

3.7 Data Viewing and Monitoring

- **Task:** Data Viewer (SSIS built-in viewer)
- **Purpose:**
 - ✓ Enabled real-time monitoring of data flow between critical transformations.
 - ✓ Allowed inspection of the data for accuracy, null handling, and transformations during package development.

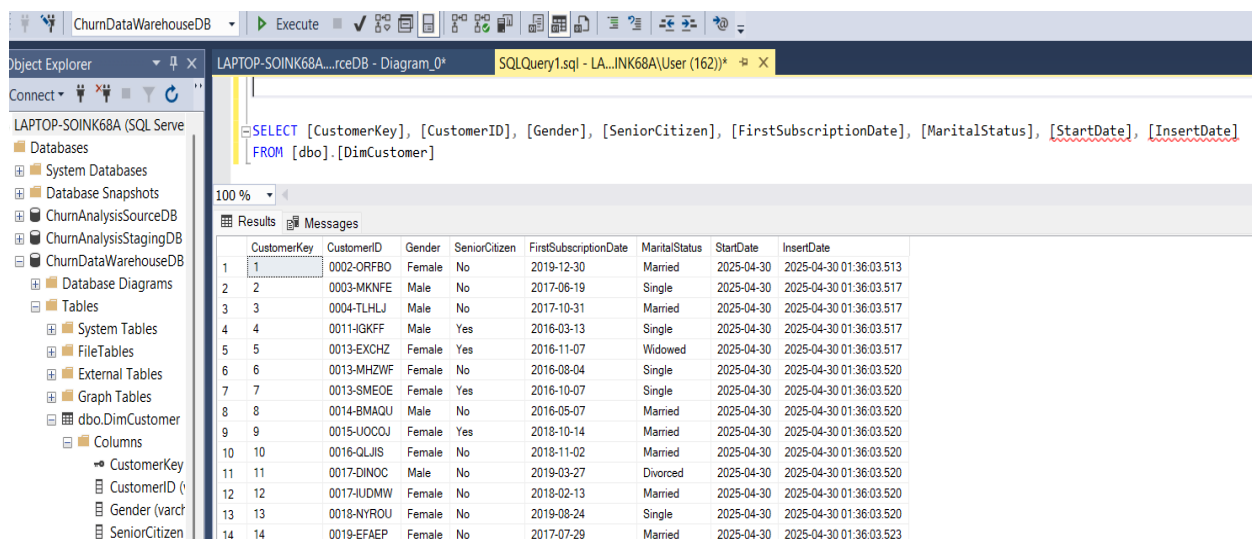
- ✓ Served as a debugging checkpoint to ensure that all data was correctly processed.

3.8 Data Loading to Data Warehouse

- **Task:** OLE DB Destination
- **Purpose:**
 - ✓ Loaded the final set of cleaned, enriched, and version-controlled customer records into the DimCustomer table.
 - ✓ Ensured that consistent customer dimension data was available for downstream analytics and reporting, adhering to SCD Type 1 and Type 2 rules.

4. Data Warehouse DimCustomer Table

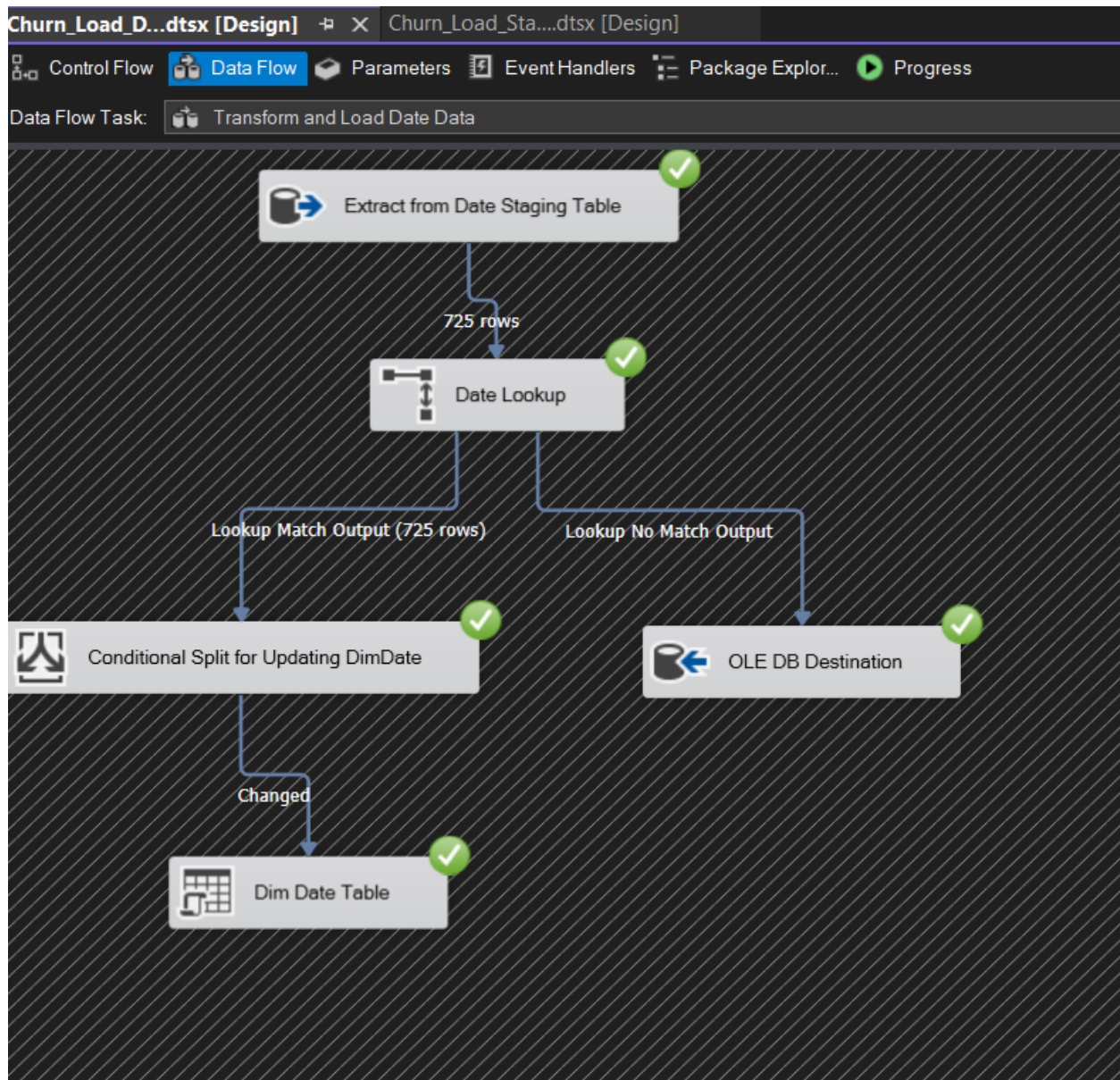
The final DimCustomer table in the Data Warehouse now contains data that was loaded from the Staging Tables, processed with change detection, historical tracking, and enrichment. The table serves as the primary source of truth for customer-related data and is used in various downstream analytical processes and reporting.



The screenshot displays the SQL Server Enterprise Manager interface. The left pane shows the 'Object Explorer' with the 'ChurnDataWarehouseDB' database selected. The right pane shows the 'Query Results' window displaying the data from the 'DimCustomer' table. The data is presented in a table with 14 rows and 8 columns. The columns are: CustomerKey, CustomerID, Gender, SeniorCitizen, FirstSubscriptionDate, MaritalStatus, StartDate, and InsertDate. The data includes various customer records with their respective attributes.

CustomerKey	CustomerID	Gender	SeniorCitizen	FirstSubscriptionDate	MaritalStatus	StartDate	InsertDate
1	0002-ORFBO	Female	No	2019-12-30	Married	2025-04-30	2025-04-30 01:36:03.513
2	0003-MKNFE	Male	No	2017-06-19	Single	2025-04-30	2025-04-30 01:36:03.517
3	0004-TLHLJ	Male	No	2017-10-31	Married	2025-04-30	2025-04-30 01:36:03.517
4	0011-IGKFF	Male	Yes	2016-03-13	Single	2025-04-30	2025-04-30 01:36:03.517
5	0013-EXCHZ	Female	Yes	2016-11-07	Widowed	2025-04-30	2025-04-30 01:36:03.517
6	0013-MHZWF	Female	No	2016-08-04	Single	2025-04-30	2025-04-30 01:36:03.520
7	0013-SMEOE	Female	Yes	2016-10-07	Single	2025-04-30	2025-04-30 01:36:03.520
8	0014-BMAQU	Male	No	2016-05-07	Married	2025-04-30	2025-04-30 01:36:03.520
9	0015-UOCOJ	Female	Yes	2018-10-14	Married	2025-04-30	2025-04-30 01:36:03.520
10	0016-QLJIS	Female	No	2018-11-02	Married	2025-04-30	2025-04-30 01:36:03.520
11	0017-DINOC	Male	No	2019-03-27	Divorced	2025-04-30	2025-04-30 01:36:03.520
12	0017-IUDMW	Female	No	2018-02-13	Married	2025-04-30	2025-04-30 01:36:03.520
13	0018-NYROU	Female	No	2019-08-24	Single	2025-04-30	2025-04-30 01:36:03.520
14	0019-EFAEP	Female	No	2017-07-29	Married	2025-04-30	2025-04-30 01:36:03.523

ETL – Transformation and Load Process for DimDate Table



1. Source Configuration

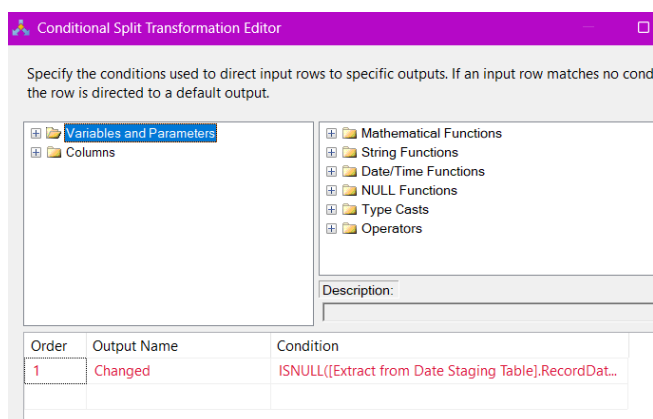
- ❖ The process starts by extracting data from the Date Staging Table (StgDate) using an OLE DB Source.
- ❖ In the OLE DB Source, I configured it to connect to the staging database and extract the following relevant columns: RecordDate, Year, Month, Day, DayOfWeek, Quarter, and DateKey.

2. Lookup Transformation (DimDate Lookup)

- ❖ To identify whether records already exist in the DimDate table, I added a Lookup Transformation.
- ❖ In the Lookup, I configured it to query the DimDate table by using RecordDate as the key to check if matching records exist.
- ❖ The Lookup transformation pulls back DateKey and RecordDate from DimDate, ensuring that the correct columns are available for comparison.
- ❖ I configured the Lookup to redirect non-matching rows (i.e., new records) to the "No Match Output" for further processing.

2. Change Detection

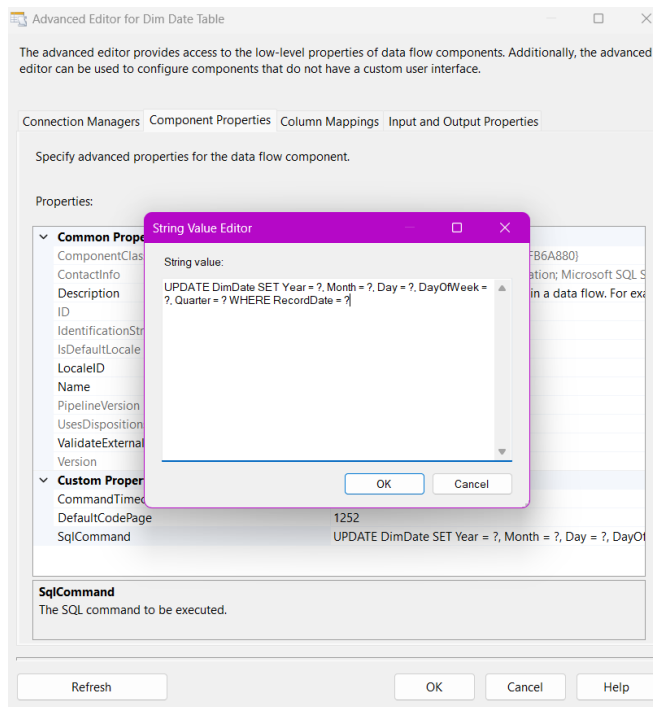
- ❖ In this step, I implemented change detection to determine whether to insert a new record or update an existing record in DimDate.
- ❖ For simplicity, I focused only on the RecordDate column as the primary key for change detection. In a production system, this would likely be extended to check for changes across all relevant columns.
- ❖ To check if the RecordDate has changed, I used a Conditional Split with the following expression:



- ❖ This expression compares the RecordDate from the staging table (StgDate) with the RecordDate from the DimDate table.

3. Destination Configuration

- The rows that do not have a match in the DimDate table (i.e., new records) are routed to the OLE DB Destination for insertion.
- The rows where RecordDate has changed (as detected by the Conditional Split) are routed to the OLE DB Command for updating existing records.
- In the OLE DB Command, I configured the SQL UPDATE statement to modify the appropriate columns (e.g., Year, Month, Day, etc.) for the existing records based on the RecordDate key.



4. Data Warehouse DimDate Table

The final **DimDate** table in the **Data Warehouse** now contains data that was loaded from the **Staging Table**.

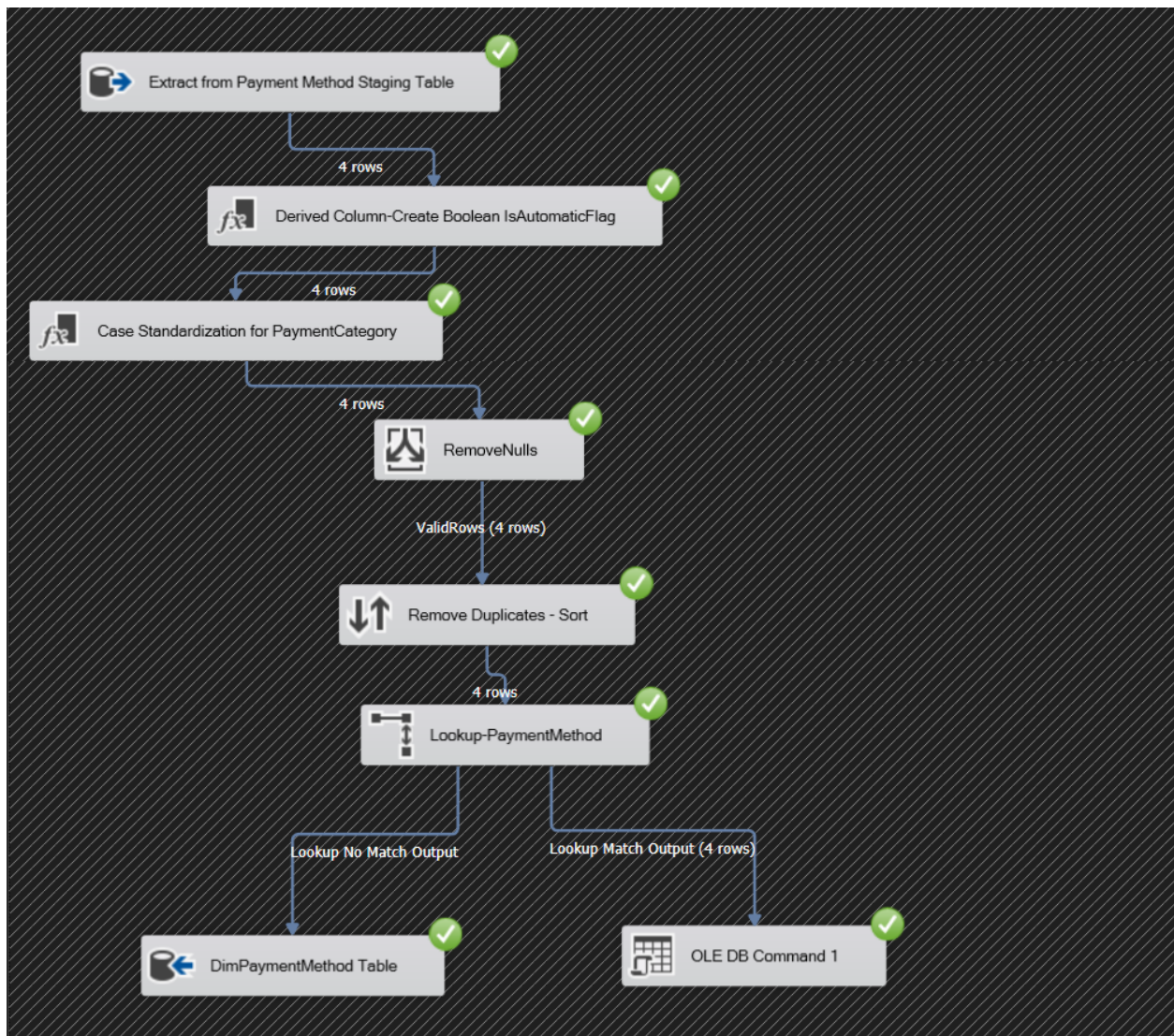
The screenshot shows the Microsoft SQL Server Management Studio interface. The 'Object Explorer' on the left shows the 'ChurnDataWarehouseDB' database. The 'Query Editor' window displays the following SQL query:

```
select count(*)
from [dbo].[DimDate]
select * from [dbo].[DimDate]
```

The 'Results' pane shows the output of the query, displaying a list of records from the DimDate table. The records are as follows:

DateKey	RecordDate	Year	Month	Day	DayOfWeek	Quarter
726	2021-01-01	2021	1	1	5	1
727	2021-01-02	2021	1	2	6	1
728	2021-01-03	2021	1	3	7	1
729	2021-01-04	2021	1	4	1	1
730	2021-01-05	2021	1	5	2	1
731	2021-01-06	2021	1	6	3	1
732	2021-01-07	2021	1	7	4	1
733	2021-01-08	2021	1	8	5	1
734	2021-01-09	2021	1	9	6	1
735	2021-01-10	2021	1	10	7	1
736	2021-01-11	2021	1	11	1	1
737	2021-01-12	2021	1	12	2	1
738	2021-01-13	2021	1	13	3	1
739	2021-01-14	2021	1	14	4	1
740	2021-01-15	2021	1	15	5	1
741	2021-01-16	2021	1	16	6	1
742	2021-01-17	2021	1	17	7	1

ETL – Transformation and Load Process for DimPaymentMethod Table



1. Source Configuration

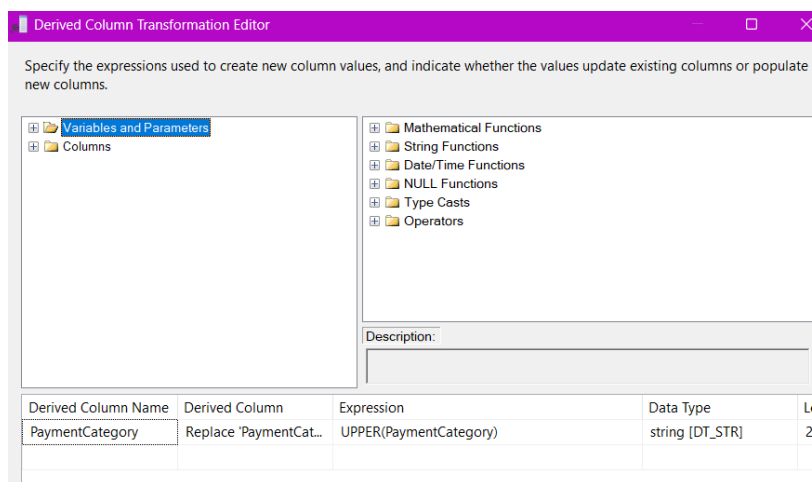
- ❖ The process starts by extracting data from the Payment Method Staging Table (StgPaymentMethod) using an OLE DB Source.
- ❖ In the OLE DB Source, I configured it to connect to the Staging Database (ChurnAnalysisStagingDB) and extract the following relevant columns:
 - ✓ PaymentMethodID, PaymentMethod, PaymentChannel, and PaymentCategory.

2. Derived Column

- ❖ In the ETL process, I added a new derived column called IsAutomatic. This column was not present in the original staging table (StgPaymentMethod) but was created during the transformation process to categorize the payment methods based on whether they are automatic or not.
- ❖ To create this Payment Method and Payment ID were utilized to derive the values as Yes and No.

3. Data Standardization

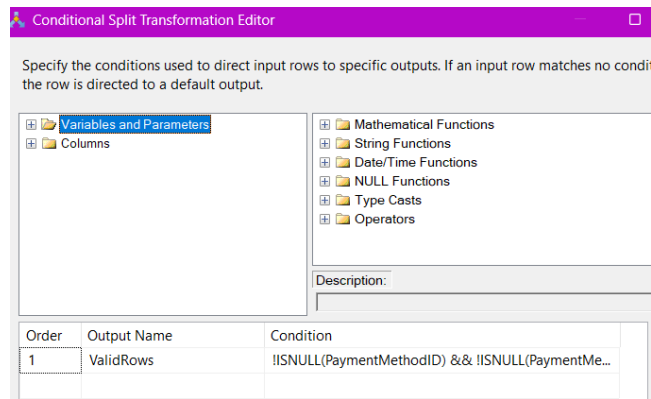
- ❖ To ensure consistency across all records, I added a Derived Column Transformation to standardize the PaymentCategory text.
- ❖ This transformation converts all PaymentCategory values to UPPERCASE, ensuring consistency in formatting and making the values suitable for reporting and comparison.



4. Data Cleansing

- To filter out rows with NULL values in essential columns (e.g., PaymentMethodID, PaymentMethod, PaymentCategory), I used a Conditional Split Transformation.
- Rows that contain NULL values in any critical columns are routed to an error output for further investigation or logging, ensuring that only complete records are processed.

✓ Condition for Filtering:



5. Duplicate Handling

- ❖ To eliminate duplicate rows based on PaymentMethodID, I used a Sort Transformation to ensure that records are sorted by PaymentMethodID.
- ❖ This makes it easy to remove any duplicate entries and maintain data integrity.
- ❖ The duplicate rows are removed by ensuring that each PaymentMethodID appears only once.

6. Lookup Transformation (DimPaymentMethod Lookup)

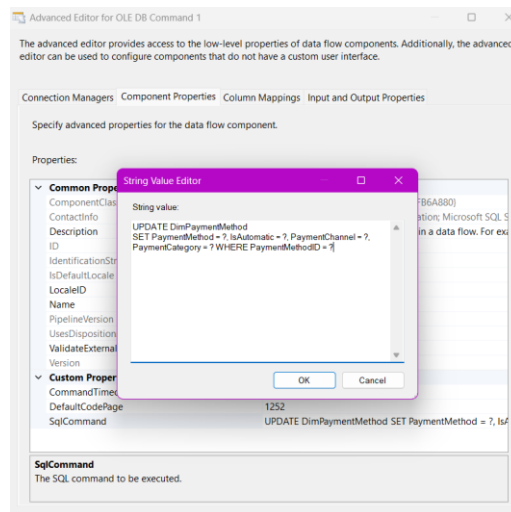
- ❖ To determine if a payment method already exists in the DimPaymentMethod table, I used a Lookup Transformation.
- ❖ The Lookup transformation queries the DimPaymentMethod table using the PaymentMethodID as the lookup key to check if a matching record exists.
- ❖ The Lookup pulls the PaymentMethodID from the DimPaymentMethod table to ensure that only new records or changed records will be processed.

✓ Routing:

- **No Match Output:** New records that don't exist in the dimension table are routed for insertion.
- **Match Output:** Existing records are sent for updates.

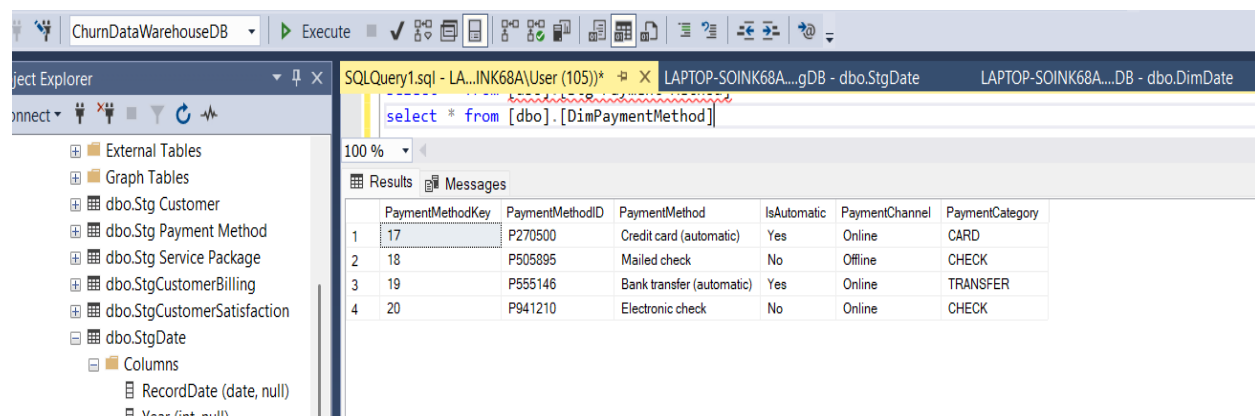
8. Update Existing Records

- ❖ For existing records in DimPaymentMethod, I used an OLE DB Command Transformation.
- ❖ This transformation executes an UPDATE SQL statement on the DimPaymentMethod table to modify attributes such as PaymentMethod, PaymentCategory, etc.
 - ✓ The SQL statement updates the existing records in the dimension table based on the PaymentMethodID.

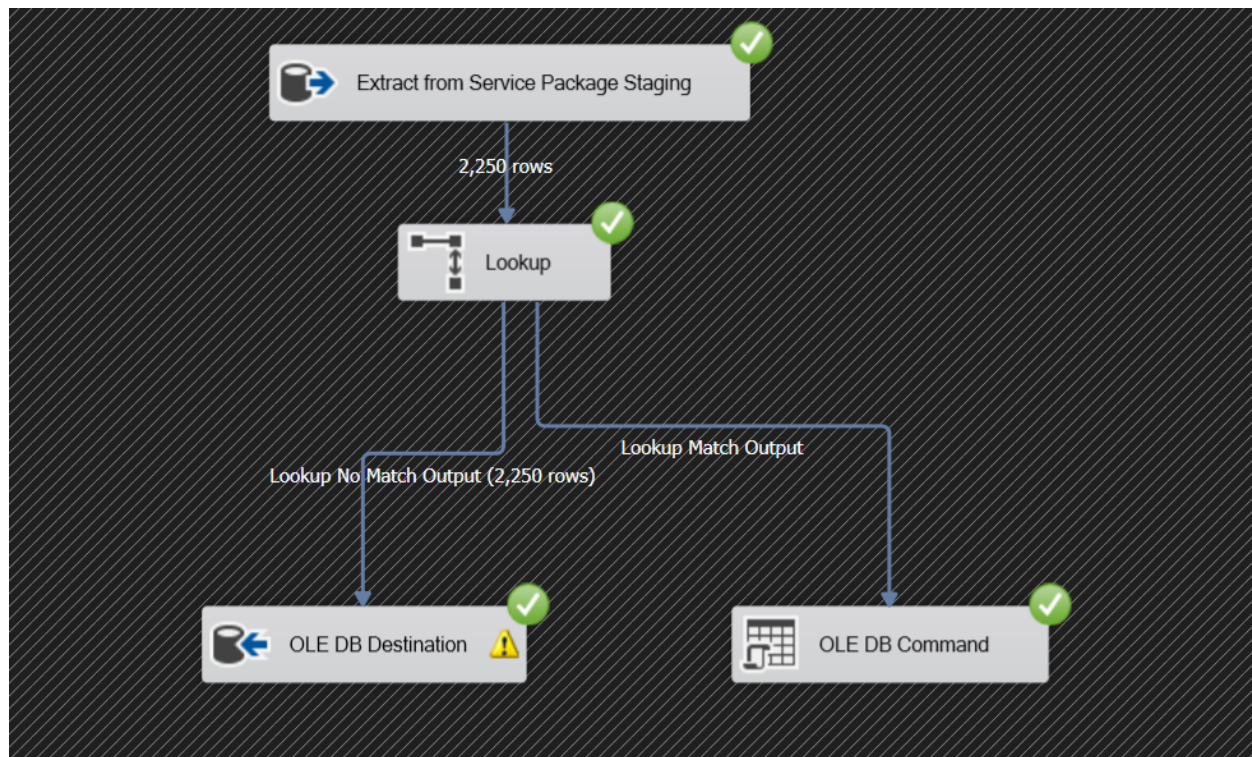


9. Data Warehouse DimPaymentMethod Table

The final DimPaymentMethod table in the Data Warehouse contains data loaded from the Staging Table after transformation and cleansing.



ETL – Transformation and Load Process for DimServicePackage Table



1. Source Configuration

- ❖ The process starts by extracting data from the Service Package Staging Table using an Extract from Service Package Staging component.
- ❖ In this extraction, all relevant data from the staging table is retrieved, with a total of 2,250 rows of data being extracted.
- ❖ This extraction serves as the initial step to gather all service package-related information for further processing.

2. Lookup Transformation

- ❖ After extracting the data, a Lookup Transformation is applied to determine if records already exist in the target dimension table (DimServicePackage).
- ❖ The Lookup transformation queries the DimServicePackage table using appropriate key fields to check if matching records exist.
- ❖ Based on the lookup results, the data flow is split into two paths:

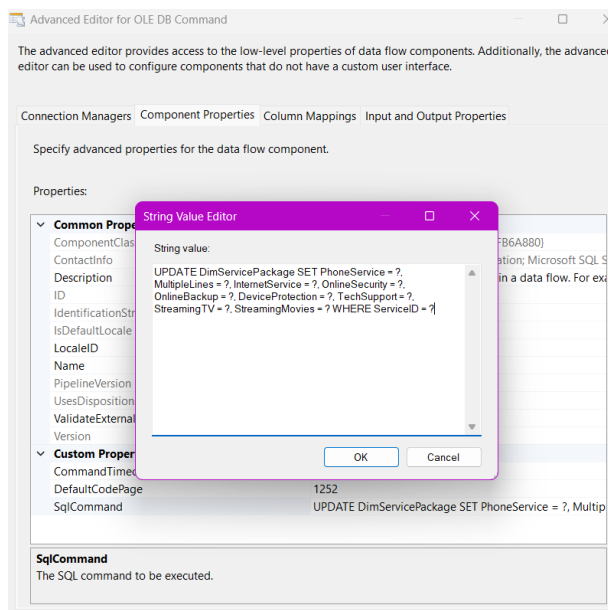
- ❖ Lookup No Match Output (2,250 rows): All 2,250 records were identified as new records that don't exist in the dimension table and are routed for insertion.
- ❖ Lookup Match Output: Existing records that match the lookup criteria are sent for updates, though in this case, there appear to be no matches.

3. Destination Management

- ❖ For new records (2,250 rows) that don't exist in DimServicePackage, an OLE DB Destination component is used.
- ❖ This component executes an INSERT operation to add these new service package records to the DimServicePackage table.
- ❖ The component shows a warning indicator (yellow triangle) which might indicate a potential configuration issue or a warning that should be addressed.

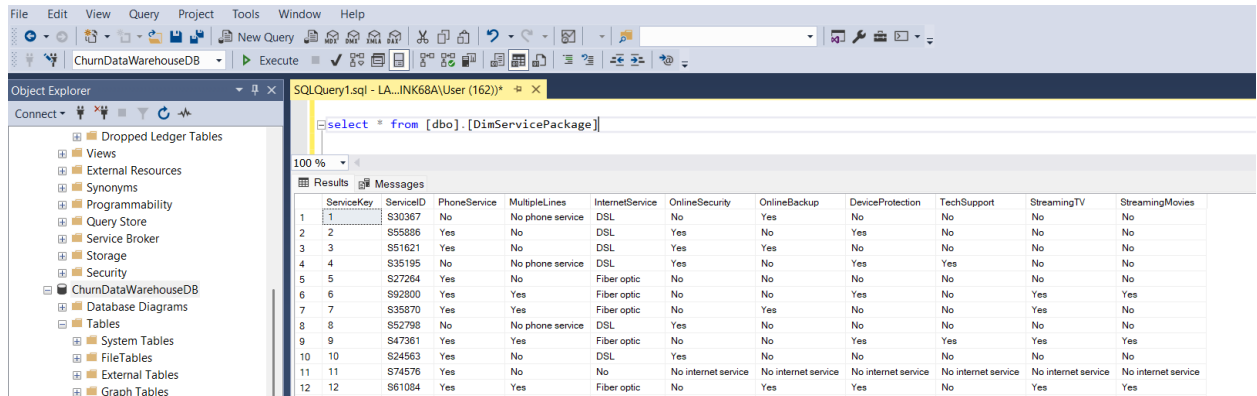
4. Update Existing Records

- ❖ For any existing records in DimServicePackage that need updating, an OLE DB Command Transformation is utilized.
- ❖ This transformation would execute an UPDATE SQL statement on the DimServicePackage table to modify attributes as needed.
- ❖ Since there are no records flowing through the Match Output path, no updates are being performed in this execution.



5. Data Flow Completion

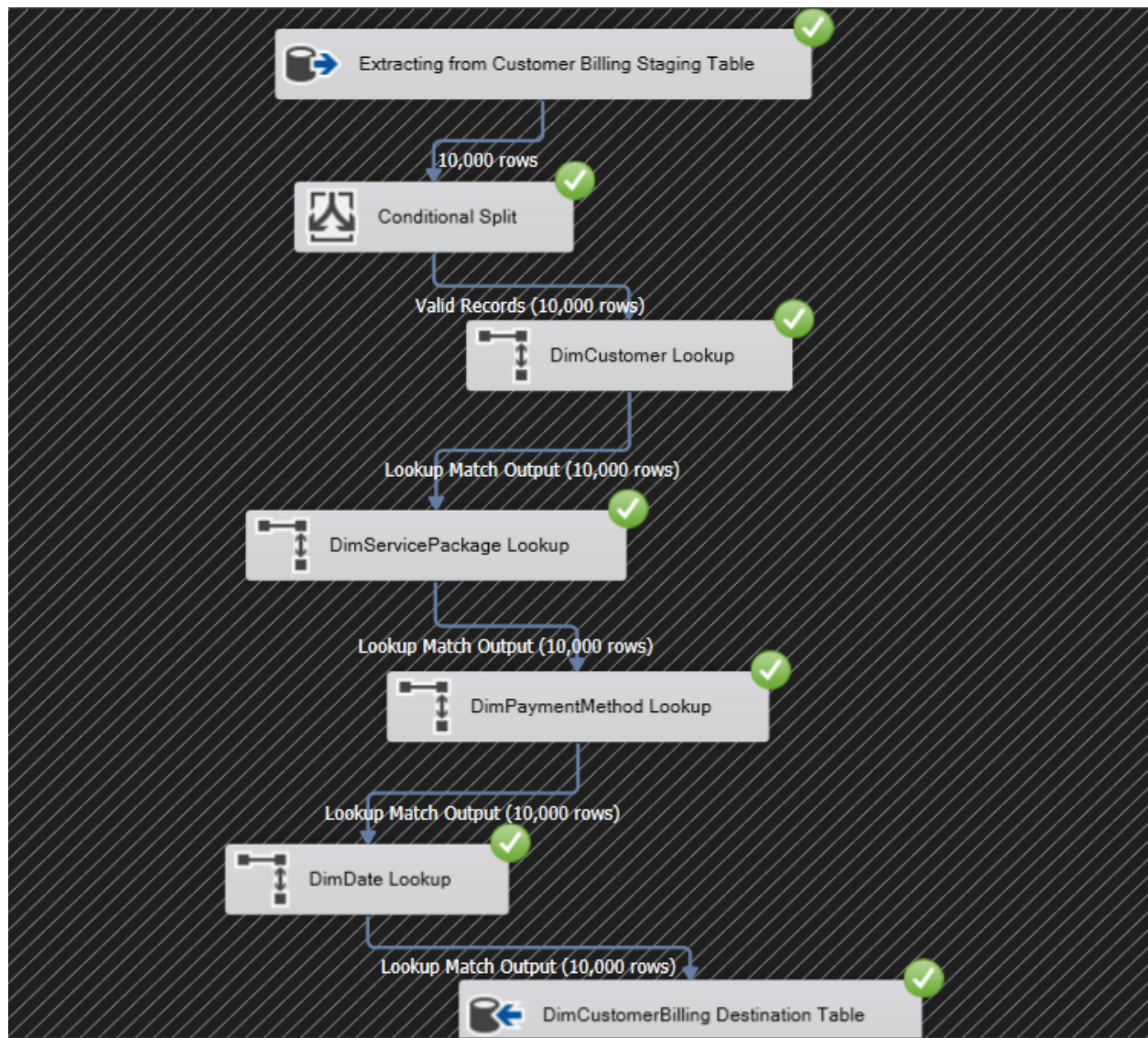
- Both the OLE DB Destination and OLE DB Command components show green checkmarks, indicating that they executed without errors.
- The final DimServicePackage table in the Data Warehouse contains data loaded from the Staging Table after transformation and cleansing.



The screenshot shows the SQL Server Enterprise Manager interface. The Object Explorer on the left displays the database structure, including the 'ChurnDataWarehouseDB' database. The central pane shows the execution of a query: `select * from [dbo].[DimServicePackage]`. The Results tab displays a table with 12 rows and 11 columns. The columns are: ServiceKey, ServiceID, PhoneService, MultipleLines, InternetService, OnlineSecurity, OnlineBackup, DeviceProtection, TechSupport, StreamingTV, and StreamingMovies. The data is as follows:

ServiceKey	ServiceID	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	StreamingTV	StreamingMovies
1	S30367	No	No phone service	DSL	No	Yes	No	No	No	No
2	S55886	Yes	No	DSL	Yes	No	Yes	No	No	No
3	S51621	Yes	No	DSL	Yes	Yes	No	No	No	No
4	S35195	No	No phone service	DSL	Yes	No	Yes	Yes	No	No
5	S27264	Yes	No	Fiber optic	No	No	No	No	No	No
6	S92800	Yes	Yes	Fiber optic	No	No	Yes	No	Yes	Yes
7	S35870	Yes	Yes	Fiber optic	No	Yes	No	No	Yes	No
8	S52798	No	No phone service	DSL	Yes	No	No	No	No	No
9	S47361	Yes	Yes	Fiber optic	No	No	Yes	Yes	Yes	Yes
10	S24563	Yes	No	DSL	Yes	No	No	No	No	No
11	S74576	Yes	No	No	No internet service	No internet service	No internet service	No internet service	No internet service	No internet service
12	S61084	Yes	Yes	Fiber optic	No	Yes	Yes	No	Yes	Yes

ETL – Transformation and Load Process for FactCustomerBilling Table

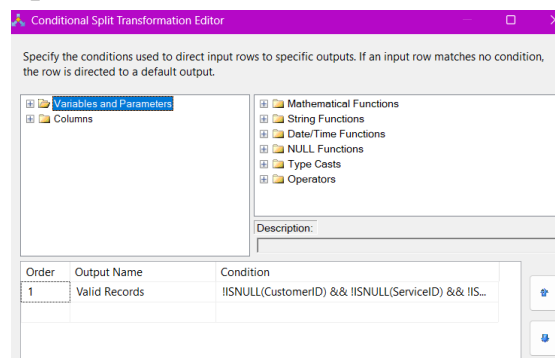


1. Source Configuration

- ❖ The process starts by extracting data from the Customer Billing Staging Table using an Extracting from Customer Billing Staging Table component.
- ❖ In this extraction, all relevant data from the staging table is retrieved to populate the fact table.
- ❖ This extraction serves as the initial step to gather all customer billing information for further processing and dimensional integration.

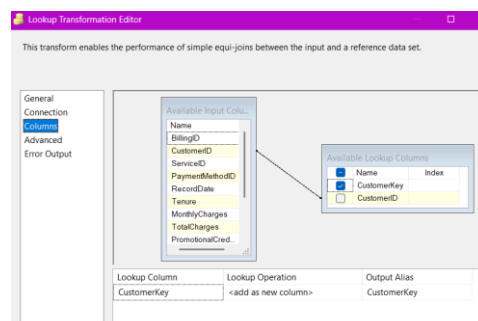
2. Data Validation with Conditional Split

- ❖ After extracting the data, a Conditional Split transformation is applied to filter out records with NULL key fields.
- ❖ The split uses the condition: `!ISNULL(CustomerID) && !ISNULL(ServiceID) && !ISNULL(PaymentMethodID) && !ISNULL(RecordDate)` to ensure data quality.
- ❖ Based on the split results, the data flow is divided into two outputs:
 - ✓ **Valid Records:** Records with all required foreign keys present are sent forward for dimension lookups.
 - ✓ **Records with NULL Keys:** Records with missing key fields are ignored in this implementation.



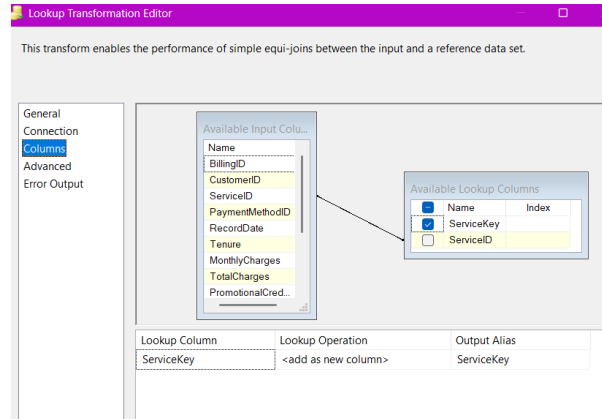
3. Dimension Key Lookups

- ❖ The Valid Records are processed through a series of four Lookup Transformations to retrieve surrogate keys:
 - ✓ **DimCustomer Lookup:** Maps CustomerID (varchar) to CustomerKey (int)
 - Join Condition: `CustomerID = CustomerID`
 - Output: CustomerKey



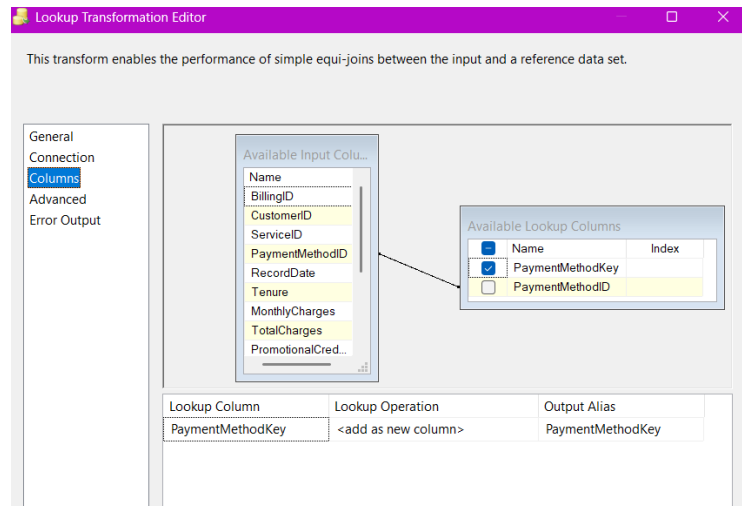
✓ **DimServicePackage Lookup:** Maps ServiceID (varchar) to ServiceKey (int)

- Join Condition: ServiceID = ServiceID
- Output: ServiceKey



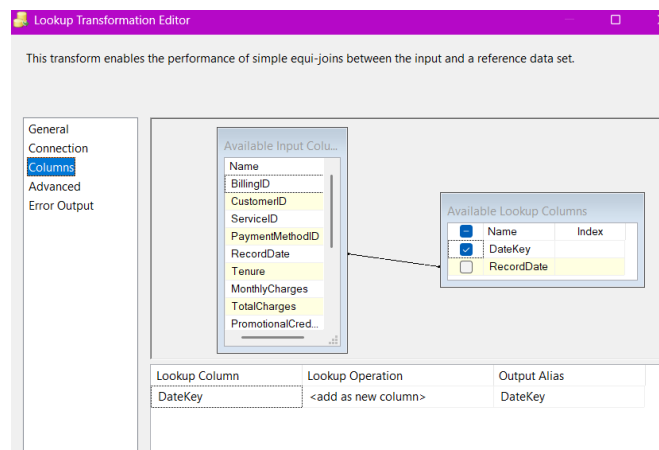
✓ **DimPaymentMethod Lookup:** Maps PaymentMethodID (varchar) to PaymentMethodKey (int)

- Join Condition: PaymentMethodID = PaymentMethodID
- Output: PaymentMethodKey



✓ **DimDate Lookup:** Maps RecordDate (date) to DateKey (int)

- Join Condition: RecordDate = Date
- Output: DateKey



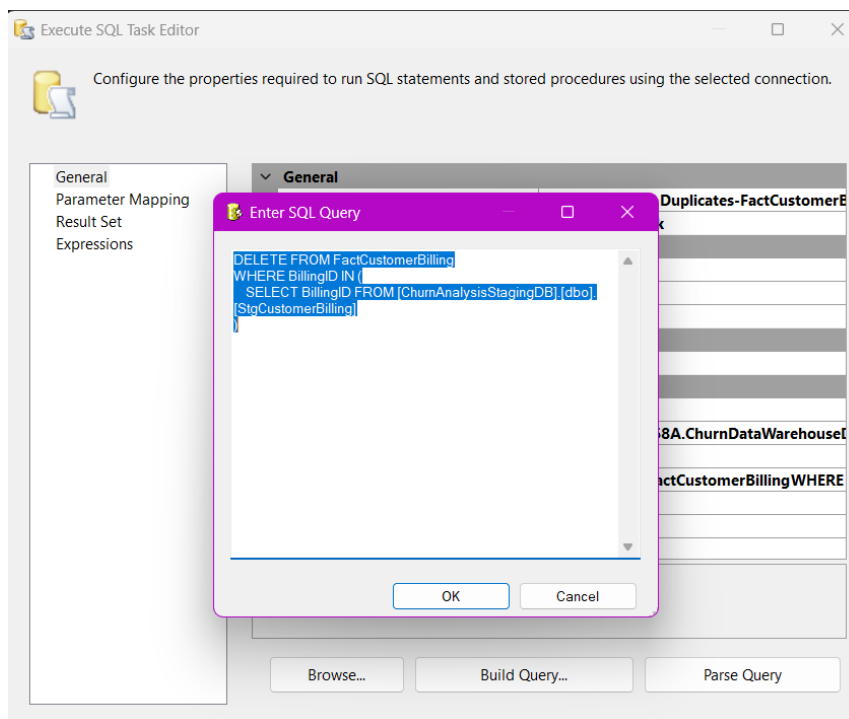
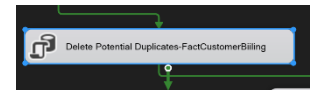
- Each lookup is configured to ignore failed lookups to ensure the process continues even if dimension references aren't found.

4. Destination Management

- ❖ After all dimension lookups are complete, an OLE DB Destination component loads data into the FactCustomerBilling table.
- ❖ The destination maps source columns and surrogate keys from lookups to the appropriate columns in the fact table.
- ❖ Key fields mapped include:
 - ✓ Primary fact table measures (MonthlyCharges, TotalCharges, etc.)
 - ✓ Surrogate keys from dimension lookups (CustomerKey, ServiceKey, PaymentMethodKey, DateKey)
 - ✓ Business keys and additional attributes

5. Duplicate Prevention

- ❖ Before the Data Flow Task executes, an Execute SQL Task runs to prevent duplicate records.
- ❖ This task executes a DELETE statement that removes any existing records in FactCustomerBilling that match incoming records based on BillingID:



- This approach ensures that when records are reprocessed, duplicates are not created in the fact table.

6. Control Flow Integration

- ❖ The overall package Control Flow consists of two main components:
 - **Execute SQL Task:** Handles duplicate prevention before loading
 - **Data Flow Task:** Performs the dimensional lookups and fact table loading
- ❖ The Data Flow Task is executed only after the SQL Task completes successfully, ensuring proper sequence of operations.

7. Data Flow Completion

- ❖ The completed process successfully loads customer billing data into the fact table with proper dimensional relationships.
- ❖ The resulting **FactCustomerBilling** table in the Data Warehouse contains transformed data with correct surrogate key references to all dimension tables, enabling efficient querying and reporting

```
select * from [dbo].[FactCustomerBilling]
```

100 %

Results Messages

	BillingID	CustomerKey	ServiceKey	PaymentMethodKey	DateKey	Tenure	MonthlyCharges	TotalCharges	PromotionalCredits	LateFeeCharges	EquipmentRentalCharges	Churn
215	B1163198	8735	2208	19	1054	26	63.75	226.199996948242	38.9500007629395	0	0	No
216	B1163646	1720	2240	17	860	49	69.4000015258789	69.4000015258789	19.9400005340576	0	0	Yes
217	B1163870	1938	1706	19	1049	7	74.3000030517578	1096.25	25.3700008392334	0	0	Yes
218	B1167207	8051	1179	18	1068	5	48.25	1293.80004882813	15.0900001525879	0	11.8400001525879	No
219	B1168077	8061	1320	19	786	24	69.8000030517578	69.8000030517578	39.2000007629395	12.840000152...	22.1599998474121	Yes
220	B1170940	2210	1483	19	901	11	24.7000007629395	1642.75	28.0699996948242	0	20.8700008392334	No
221	B1171080	2038	1830	17	869	4	27.1100006103516	670.919982910156	44.7900009155273	0	0	Yes
222	B1171364	9049	57	19	1372	39	23.7800006866455	812.390014648438	47.6699981689453	0	23.5200004577637	No
223	B1172259	1169	1359	20	1281	60	35.5099983215332	1785.4599609375	43.4099998474121	0	11.0600004196167	No
224	B1172748	9775	1901	19	788	31	56.9000015258789	2560.10009765625	27.9400005340576	0	13.6000003814697	No
225	B1174606	7458	1011	20	753	52	89.0500030517578	6254.4501953125	9.5	0	12.8900003433228	No
226	B1176409	924	1825	17	754	33	100.5	918.599975585938	34.9900016784668	0	0	Yes

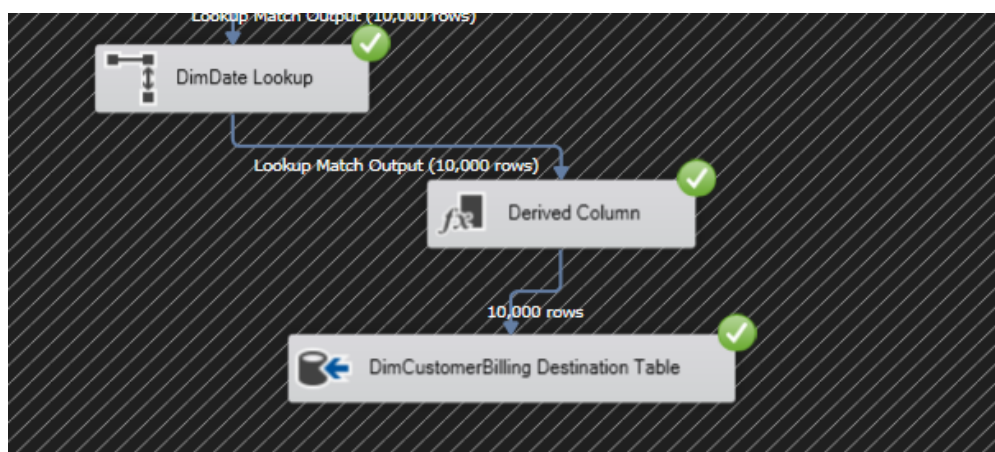
STEP 06 - ETL Development – Accumulating Fact Table

1. First I extended the required columns to FactCustomerBilling table in ChurnDataWarehouseDB:

```
ALTER TABLE FactCustomerBilling
ADD accm_txn_create_time DATETIME,
    accm_txn_complete_time DATETIME NULL,
    txn_process_time_hours FLOAT NULL;
```

Now I updated my existing ETL package (Churn_Load_DW.dtsx) to set the transaction creation time. So I added a **Derived Column Transformation** after last lookup (after DimDate lookup):

- Dragged a "Derived Column Transformation" from the SSIS Toolbox
- Connected the output from the DimDate lookup to this new component



2. The FactCustomerBilling table below shows a new column that has been created using the above component, which includes the creation time.

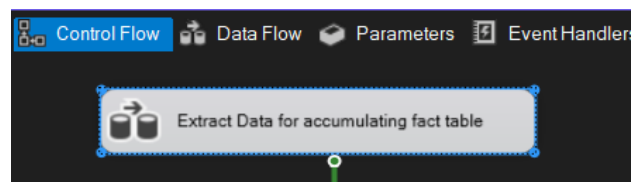
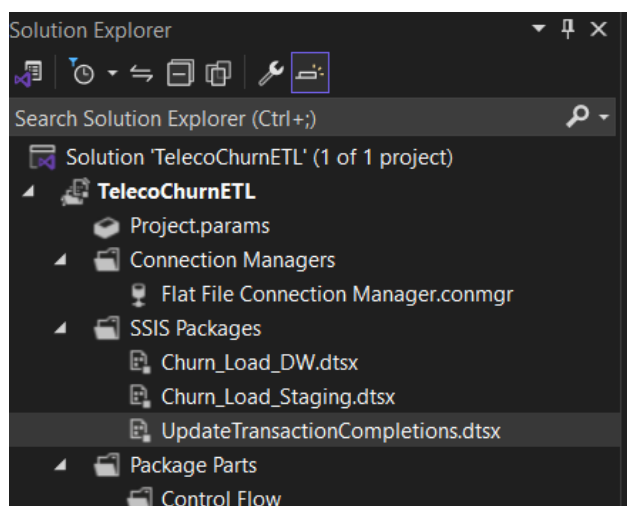
Results													
	BillingID	CustomerKey	ServiceKey	PaymentMethodKey	DateKey	Tenure	MonthlyCharges	TotalCharges	PromotionalCredits	LateFeeCharges	EquipmentRentalCharges	Churn	accm_txn_create_time
1	B1002182	5035	754	19	859	72	84.1999969482422	4299.75	40.4599990844727	0	21.1499996185303	No	2025-05-01 16:23:10.603
2	B1002545	8863	1989	20	835	26	85	2624.25	15.5200004577637	0	10.0699996948242	Yes	2025-05-01 16:23:10.603
3	B1003796	5689	111	17	991	30	20.6499996185303	1057	4.09000015258789	0	0	No	2025-05-01 16:23:10.603
4	B1005238	7248	651	17	995	12	89.1999969482422	990.299987792969	39.7799987792969	0	0	No	2025-05-01 16:23:10.603
5	B1006379	5833	291	17	747	50	71.5999984741211	3651.27001953125	5.46000003814697	0	16.69000005340576	No	2025-05-01 16:23:10.607
6	B1006408	2504	2217	18	731	64	45.7999992370605	45.7999992370605	41.439998626709	0	0	No	2025-05-01 16:23:10.607

3. I created a separate data set in a SQL table called Transactions, formatted exactly as requested in the assignment PDF, and inserted records for 10 billing IDs.

```
CREATE TABLE Transactions (  
    txn_id VARCHAR(20) PRIMARY KEY,  
    accm_txn_complete_time DATETIME  
);
```

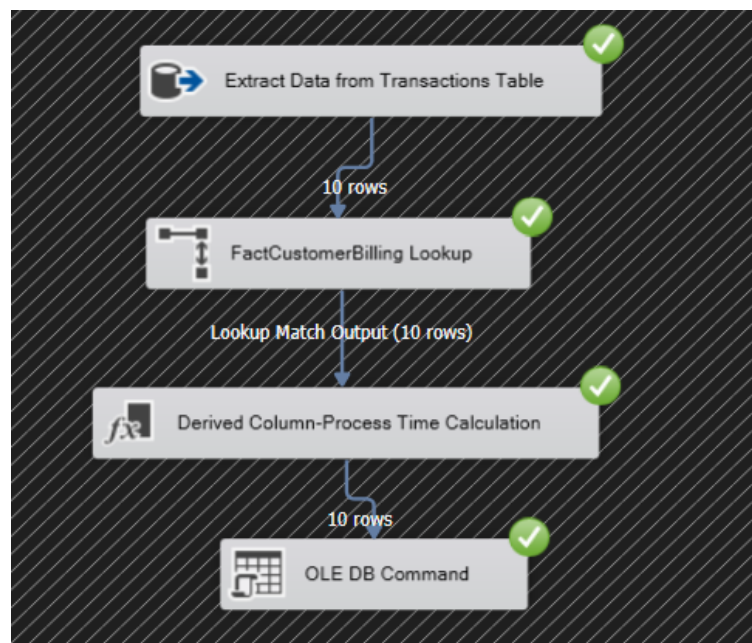
```
-- Insert records into the Transactions table  
INSERT INTO Transactions (txn_id, accm_txn_complete_time)  
VALUES  
('B1002182', '2025-05-01 16:23:10.603'),  
('B1002545', '2025-05-01 17:12:25.124'),  
('B1003796', '2025-05-02 08:30:45.763'),  
('B1005238', '2025-05-02 09:45:12.901'),  
('B1006379', '2025-05-03 10:23:59.232'),  
('B1006408', '2025-05-03 14:54:11.542'),  
('B1006843', '2025-05-04 11:10:35.890'),  
('B1007354', '2025-05-04 16:15:40.234'),  
('B1008170', '2025-05-05 13:20:50.456'),  
('B1008511', '2025-05-05 18:05:30.123');
```

4. I created a dedicated ETL SSIS package named "UpdateTransactionCompletions.dtsx." In this package, I added a Data Flow Task to the Control Flow. I then configured the Data Flow by including a Source Component, specifically by dragging an OLE DB Source to set it up to read from the Transactions sql table.



- ❖ I added a Lookup Transformation and connected the source to the Lookup component. After that, I configured the connection to the ChurnDataWarehouseDB.
- ❖ I set a lookup query:
 1. SELECT BillingID, accm_txn_create_time FROM FactCustomerBilling
- ❖ Then I mapped the 'txn_id' from the input to the 'BillingID' from the lookup as the joined columns and checked 'accm_txn_create_time' as a passing column.
- ❖ I connected the Lookup Match Output to a Derived Column component.
- ❖ I added a new column titled 'txn_process_time_hours'.
- ❖ The expression for calculating the process time in hours is:
 1. (DT_R8)DATEDIFF("hour", accm_txn_create_time, accm_txn_complete_time)

Finally, I added an OLE DB Command Transformation and connected the Derived Column to the OLE DB Command with the update SQL command..



5. Finally ETL pipeline has updated txn_process_time_hours in my Data Warehouse FactCustomerBilling Table as shown below:

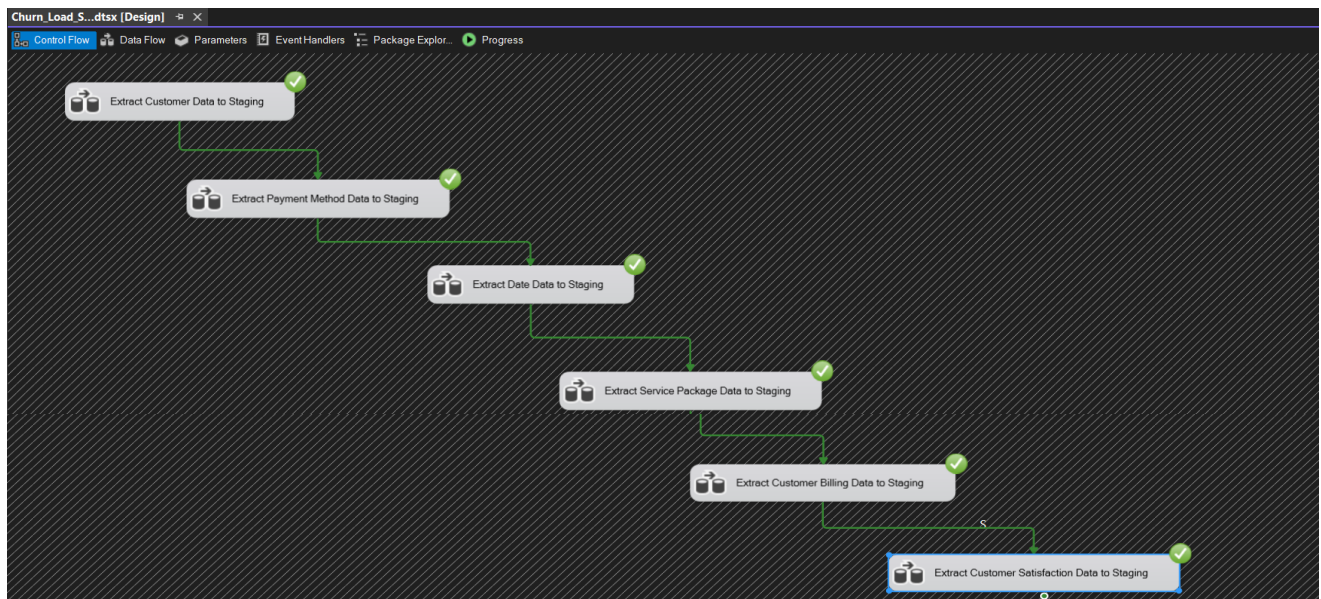
	BillingID	CustomerKey	ServiceKey	PaymentMethodKey	DateKey	Tenure	MonthlyCharges	TotalCharges	PromotionalCredits	LateFeeCharges	EquipmentRentalCharges	Churn	accm_txn_create_time	accm_txn_complete_time	txn_process_time_hours
1	B1002182	5035	754	19	859	72	84.1999969482422	4299.75	40.4599990844727	0	21.1499996185303	No	2025-05-01 16:23:10.603	2025-05-01 16:23:10.603	0
2	B1002545	8863	1989	20	835	26	85	2624.25	15.5200004577637	0	10.0699996948242	Yes	2025-05-01 16:23:10.603	2025-05-01 17:12:25.123	1
3	B1003796	5689	111	17	991	30	20.6499996185303	1057	4.09000015258789	0	0	No	2025-05-01 16:23:10.603	2025-05-02 08:30:45.763	16
4	B1005238	7248	651	17	995	12	89.1999969482422	990.299987792969	39.7799987792969	0	0	No	2025-05-01 16:23:10.603	2025-05-02 09:45:12.900	17
5	B1006379	5833	291	17	747	50	71.5999984741211	3651.27001953125	5.46000003814697	0	16.6900005340576	No	2025-05-01 16:23:10.607	2025-05-03 10:23:59.233	42
6	B1006408	2504	2217	18	731	64	45.7999982370605	45.7999982370605	41.439998626709	0	0	No	2025-05-01 16:23:10.607	2025-05-03 14:54:11.543	46
7	B1006843	7508	1535	18	864	16	20.0499982370605	470.200012207031	44.7400016784668	0	27.4799995422363	No	2025-05-01 16:23:10.607	2025-05-04 11:10:35.890	67
8	B1007354	5746	1136	18	982	41	99.6500015258789	2404.85009765625	38.430003051758	0	0	No	2025-05-01 16:23:10.607	2025-05-04 16:15:40.233	72
9	B1008170	2810	567	17	867	69	87.76000021362305	2886.6298828125	35.1500015258789	0	0	No	2025-05-01 16:23:10.607	2025-05-05 13:20:50.457	93
10	B1008511	157	182	18	942	54	19.8500003814697	19.8500003814697	35.430003051758	0	27.6100006103516	No	2025-05-01 16:23:10.607	2025-05-05 18:05:30.123	98

Summary

1. Explained my selected Kaggle data and its formats
2. Prepared data sources as SQL database, text file, and Excel (xlsx)
3. Proposed a solution architecture for ETL implementation and introduced my star schema and data warehouse design
4. Explained data extraction from sources (database, text file, and Excel) to staging database using SSIS
5. Detailed each Data Flow task created for transforming and loading data from staging database to dimension and fact tables
6. Implemented accumulating fact tables with requested modifications

Below I have attached control flow diagrams showing the complete ETL process implementation and execution flow for a quick glance:

Extraction Control Flow



Transformation and Loading to Data Warehouse

