HURTOWNIE DANYCH

Lista 5 – Tworzenie wymiaru czasowego, przygotowanie procesu ETL

ALEKSANDER STEPANIUK
NR. INDEKSU: 272644
Politechnika Wrocławska, Informatyka Stosowana

Rozwiązania:

Zadanie 1.

```
IF EXISTS(
    SELECT *
    FROM INFORMATION_SCHEMA.TABLES
    WHERE TABLE_SCHEMA = 'Stepaniuk'
    AND TABLE_NAME = 'FACT_SALES'
)
BEGIN
     DROP TABLE Stepaniuk.FACT_SALES;
END

IF EXISTS(
    SELECT *
    FROM INFORMATION_SCHEMA.TABLES
    WHERE TABLE_SCHEMA = 'Stepaniuk'
    AND TABLE_NAME = 'DIM_CUSTOMER'
)
BEGIN
    DROP TABLE Stepaniuk.DIM_CUSTOMER;
END

IF EXISTS(
    SELECT *
    FROM INFORMATION_SCHEMA.TABLES
    WHERE TABLE_SCHEMA = 'Stepaniuk'
    AND TABLE_NAME = 'DIM_PRODUCT'
)
BEGIN
    DROP TABLE Stepaniuk.DIM_PRODUCT;
END

IF EXISTS(
    SELECT *
    FROM INFORMATION_SCHEMA.TABLES
    WHERE TABLE_SCHEMA = 'Stepaniuk'
    AND TABLE_NAME = 'DIM_SALESPERSON'
)
BEGIN
    DROP TABLE Stepaniuk.DIM_SALESPERSON;
END
```

Zadanie 2.
Tworzenie DIM_TIME

```sql
CREATE TABLE Stepaniuk.DIM_TIME (
    PK_TIME INT PRIMARY KEY,
    Year INT NOT NULL,
    Quarter INT NOT NULL,
    Month INT NOT NULL,
    MonthName NVARCHAR(50) NOT NULL,
    WeekdayName NVARCHAR(50) NOT NULL,
    DayOfMonth INT NOT NULL
);
```

Tabele pomocnicze:

```sql
CREATE TABLE Stepaniuk.Months (
    MonthID INT IDENTITY(1,1) PRIMARY KEY,
    MonthName NVARCHAR(50) NOT NULL
);
INSERT INTO Stepaniuk.Months (MonthName) VALUES
('Styczeń'),
('Luty'),
('Marzec'),
('Kwiecień'),
('Maj'),
('Czerwiec'),
('Lipiec'),
('Sierpień'),
('Wrzesień'),
('Październik'),
('Listopad'),
('Grudzień');

CREATE TABLE Stepaniuk.Weekdays (
    WeekdayID INT IDENTITY(1,1) PRIMARY KEY,
    WeekdayName NVARCHAR(50) NOT NULL
);
INSERT INTO Stepaniuk.Weekdays (WeekdayName) VALUES
('Poniedziałek'),
('Wtorek'),
('Środa'),
('Czwartek'),
('Piątek'),
('Sobota'),
('Niedziela');
```

Wypełniamy danymi:

QLQuery1.sql - ZA...-LAPTOP\aliks (75))*

```sql
WITH data AS (
    SELECT DISTINCT
        DATEPART(YYYY, soh.OrderDate) * 10000 + DATEPART(MM, soh.OrderDate) * 100 + DATEPART(DD, soh.OrderDate) AS PK_TIME,
        DATEPART(YYYY, soh.OrderDate) AS Year,
        DATEPART(QQ, soh.OrderDate) AS Quarter,
        DATEPART(MM, soh.OrderDate) AS Month,
        m.MonthName AS MonthName,
        d.WeekdayName AS WeekdayName,
        DATEPART(DD, soh.OrderDate) AS DayOfMonth
    FROM
        Sales.SalesOrderHeader AS soh
        JOIN Stepaniuk.Months m ON DATEPART(MM, soh.OrderDate) = m.MonthID
        JOIN Stepaniuk.Weekdays d ON DATEPART(WEEKDAY, DATEADD(DAY, -1, soh.OrderDate)) = d.WeekdayID

    UNION

    SELECT DISTINCT
        DATEPART(YYYY, soh.ShipDate) * 10000 + DATEPART(MM, soh.ShipDate) * 100 + DATEPART(DD, soh.ShipDate) AS PK_TIME,
        DATEPART(YYYY, soh.ShipDate) AS Year,
        DATEPART(QQ, soh.ShipDate) AS Quarter,
        DATEPART(MM, soh.ShipDate) AS Month,
        m.MonthName AS MonthName,
        d.WeekdayName AS WeekdayName,
        DATEPART(DD, soh.ShipDate) AS DayOfMonth

    FROM
        Sales.SalesOrderHeader AS soh
        JOIN Stepaniuk.Months m ON DATEPART(MM, soh.ShipDate) = m.MonthID
        JOIN Stepaniuk.Weekdays d ON DATEPART(WEEKDAY, DATEADD(DAY, -1, soh.ShipDate)) = d.WeekdayID
)
INSERT INTO Stepaniuk.DIM_TIME (PK_TIME, Year, Quarter, Month, MonthName, WeekdayName, DayOfMonth)
SELECT PK_TIME, Year, Quarter, Month, MonthName, WeekdayName, DayOfMonth
FROM data;
```

Zadanie 3.

SQLQuery1.sql - ZA...-LAPTOP\aliks (75))*

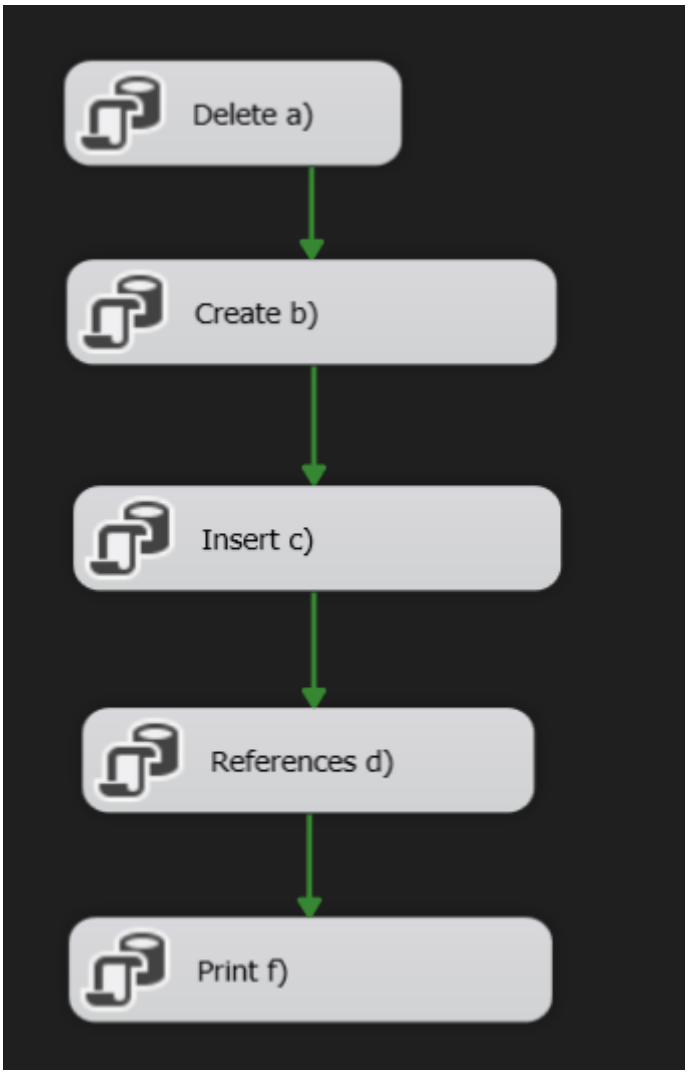
```sql
UPDATE Stepaniuk.DIM_PRODUCT
SET Color = COALESCE(Color, 'Unknown'),
WHERE Color IS NULL;

UPDATE Stepaniuk.DIM_PRODUCT
SET SubCategoryName = COALESCE(SubCategoryName, 'Unknown'),
WHERE SubCategoryName IS NULL;

UPDATE Stepaniuk.DIM_CUSTOMER
SET CountryRegionCode = COALESCE(CountryRegionCode, '000'),
WHERE CountryRegionCode IS NULL;

UPDATE Stepaniuk.DIM_CUSTOMER
SET [Group] = COALESCE([Group], 'Unknown'),
WHERE [Group] IS NULL;
```

91 %

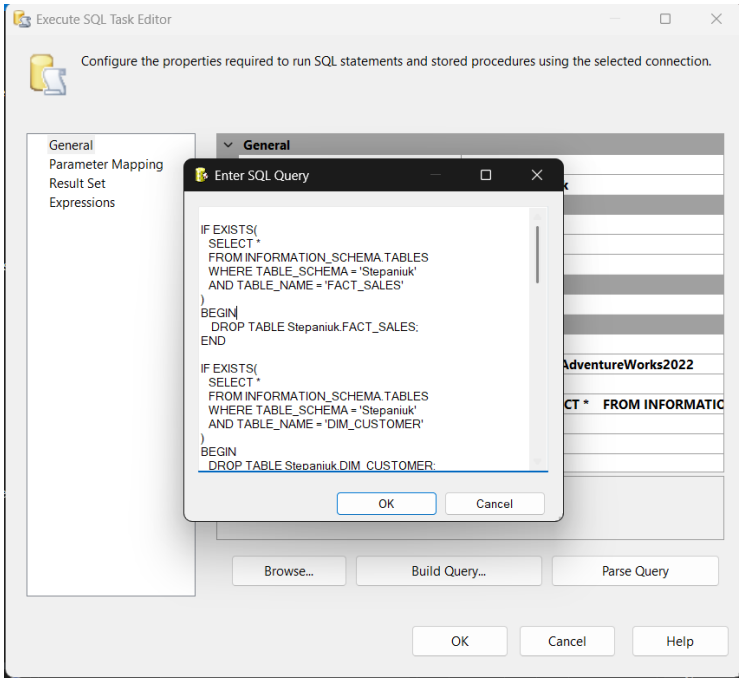
Results | Messages

	ProductID	Name	ListPrice	Color	SubCategoryName	CategoryName	Weight	Size	IsPurchased
1	1	Adjustable Race	0,00	Unknown	Unknown	NULL	NULL	NULL	0
2	2	Bearing Ball	0,00	Unknown	Unknown	NULL	NULL	NULL	0
3	3	BB Ball Bearing	0,00	Unknown	Unknown	NULL	NULL	NULL	0
4	4	Headset Ball Bearings	0,00	Unknown	Unknown	NULL	NULL	NULL	0
5	316	Blade	0,00	Unknown	Unknown	NULL	NULL	NULL	0
6	317	LL Crankarm	0,00	Black	Unknown	NULL	NULL	NULL	0

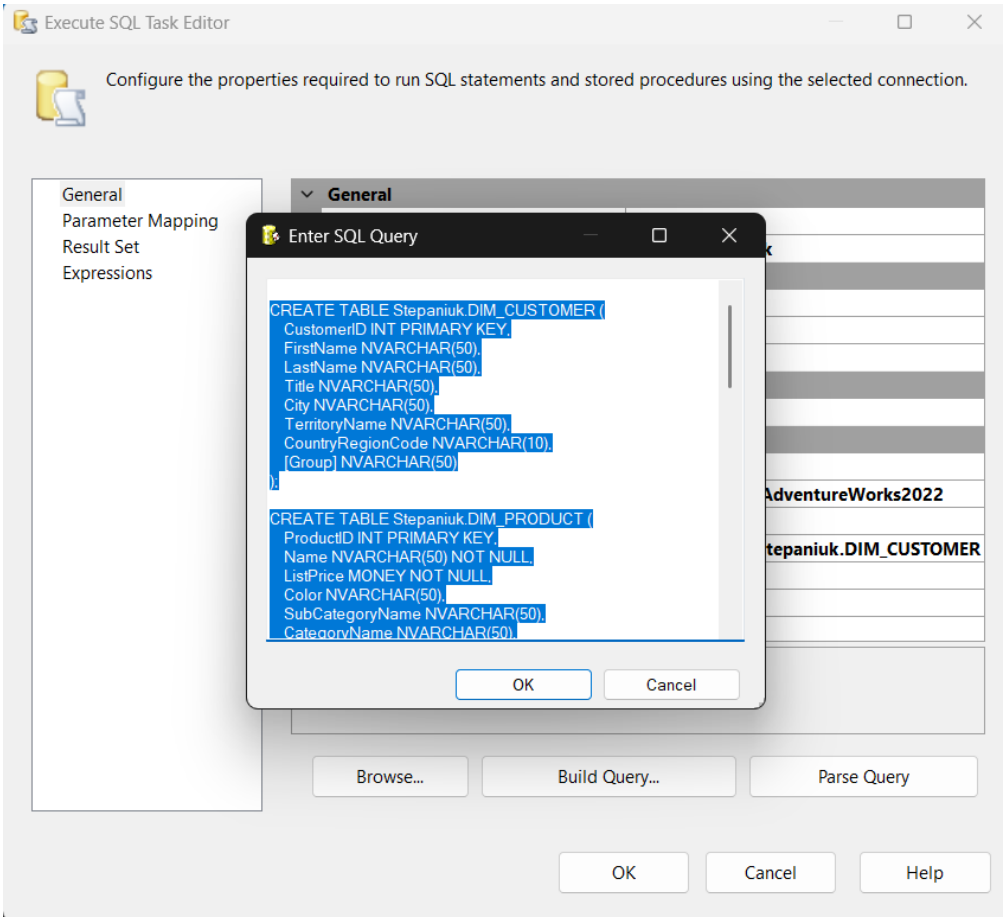
Zadanie 4.


Delete a)
Create b)
Insert c)
References d)
Print f)

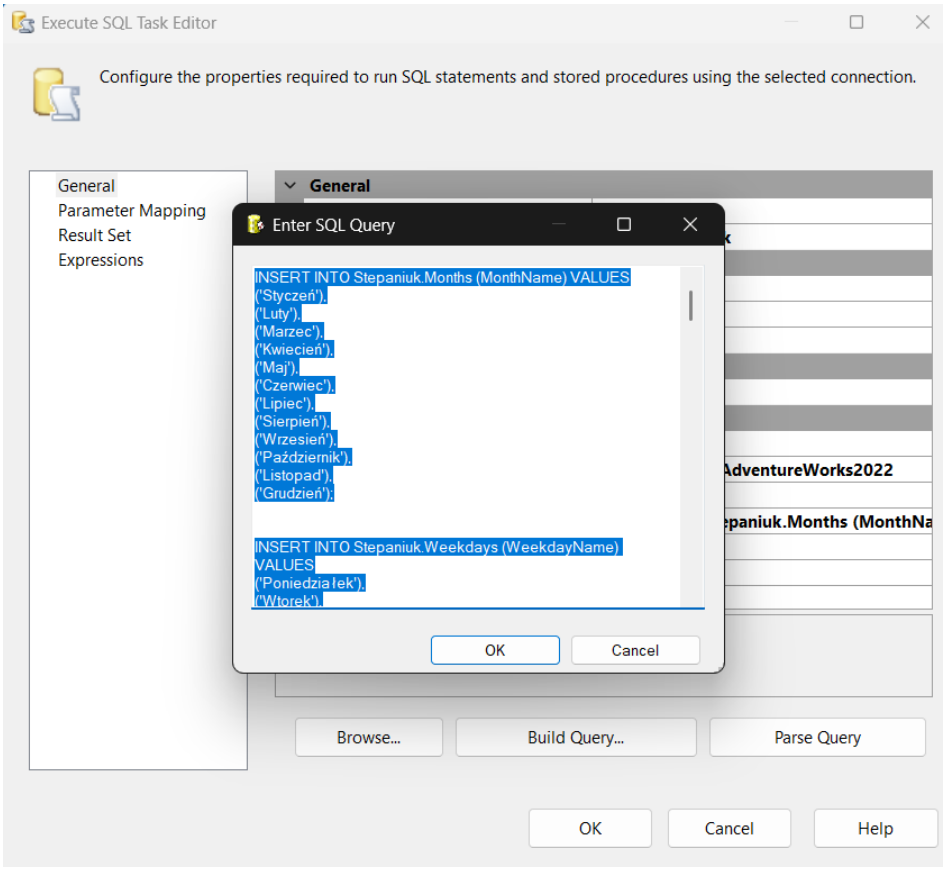
a)


Execute SQL Task Editor
Configure the properties required to run SQL statements and stored procedures using the selected connection.
General
Parameter Mapping
Result Set
Expressions
General
Enter SQL Query
IF EXISTS(
SELECT *
FROM INFORMATION_SCHEMA.TABLES
WHERE TABLE_SCHEMA = 'Stepaniuk'
AND TABLE_NAME = 'FACT_SALES'
)
BEGIN
DROP TABLE Stepaniuk.FACT_SALES;
END
IF EXISTS(
SELECT *
FROM INFORMATION_SCHEMA.TABLES
WHERE TABLE_SCHEMA = 'Stepaniuk'
AND TABLE_NAME = 'DIM_CUSTOMER'
)
BEGIN
DROP TABLE Stepaniuk.DIM_CUSTOMER;
OK
Cancel
AdventureWorks2022
CT * FROM INFORMATIC
Browse...
Build Query...
Parse Query
OK
Cancel
Help

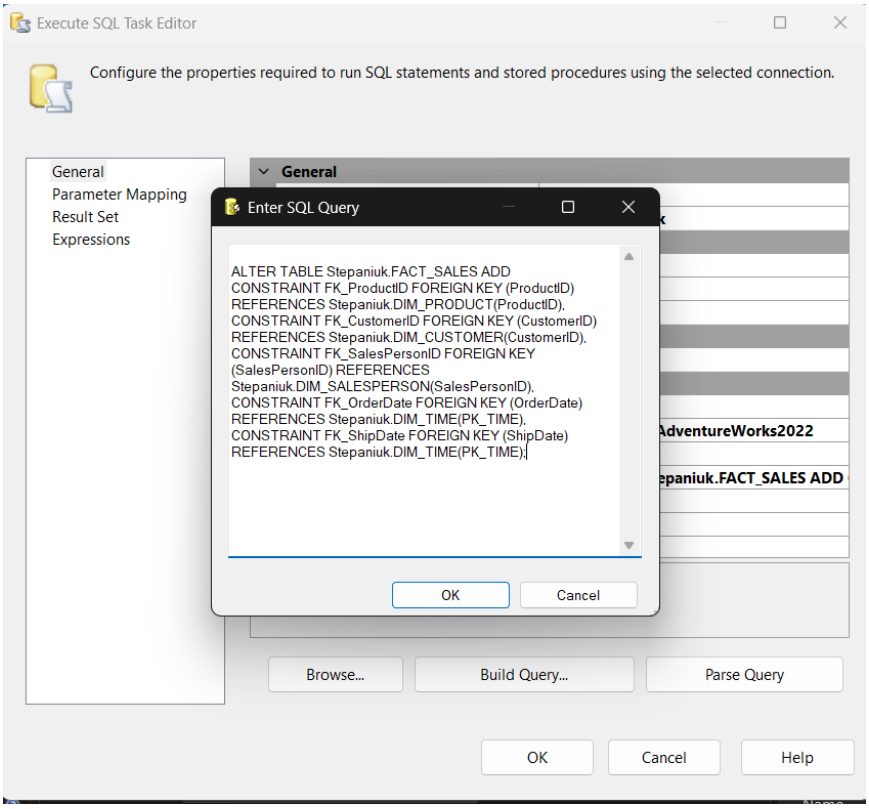
b)


Execute SQL Task Editor
Configure the properties required to run SQL statements and stored procedures using the selected connection.
General
Parameter Mapping
Result Set
Expressions
General
Enter SQL Query
CREATE TABLE Stepaniuk.DIM_CUSTOMER (
CustomerID INT PRIMARY KEY,
FirstName NVARCHAR(50),
LastName NVARCHAR(50),
Title NVARCHAR(50),
City NVARCHAR(50),
TerritoryName NVARCHAR(50),
CountryRegionCode NVARCHAR(10),
[Group] NVARCHAR(50)
);
CREATE TABLE Stepaniuk.DIM_PRODUCT (
ProductID INT PRIMARY KEY,
Name NVARCHAR(50) NOT NULL,
ListPrice MONEY NOT NULL,
Color NVARCHAR(50),
SubCategoryName NVARCHAR(50),
CategoryName NVARCHAR(50)
OK
Cancel
AdventureWorks2022
tepaniuk.DIM_CUSTOMER
Browse...
Build Query...
Parse Query
OK
Cancel
Help

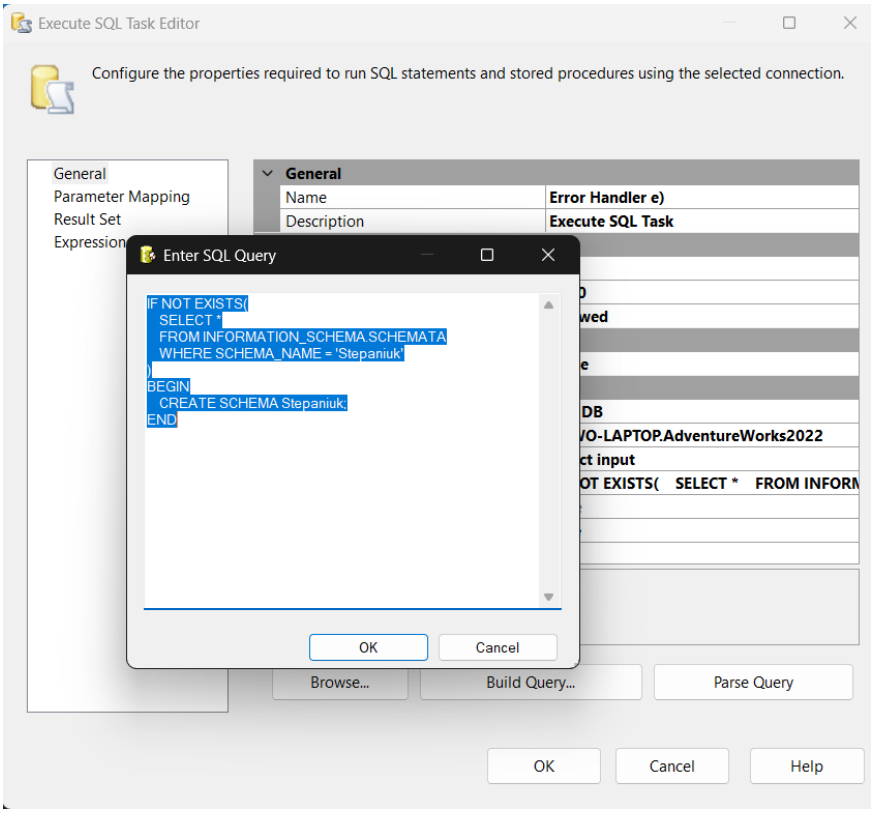
c)


Execute SQL Task Editor
Configure the properties required to run SQL statements and stored procedures using the selected connection.
General
Parameter Mapping
Result Set
Expressions
General
Enter SQL Query
INSERT INTO Stepaniuk.Months (MonthName) VALUES
('Styczeń'),
('Luty'),
('Marzec'),
('Kwiecień'),
('Maj'),
('Czerwiec'),
('Lipiec'),
('Sierpień'),
('Wrzesień'),
('Październik'),
('Listopad'),
('Grudzień');
INSERT INTO Stepaniuk.Weekdays (WeekdayName)
VALUES
('Poniedziałek'),
('Wtorek')
OK
Cancel
AdventureWorks2022
paniuk.Months (MonthNa
Browse...
Build Query...
Parse Query
OK
Cancel
Help

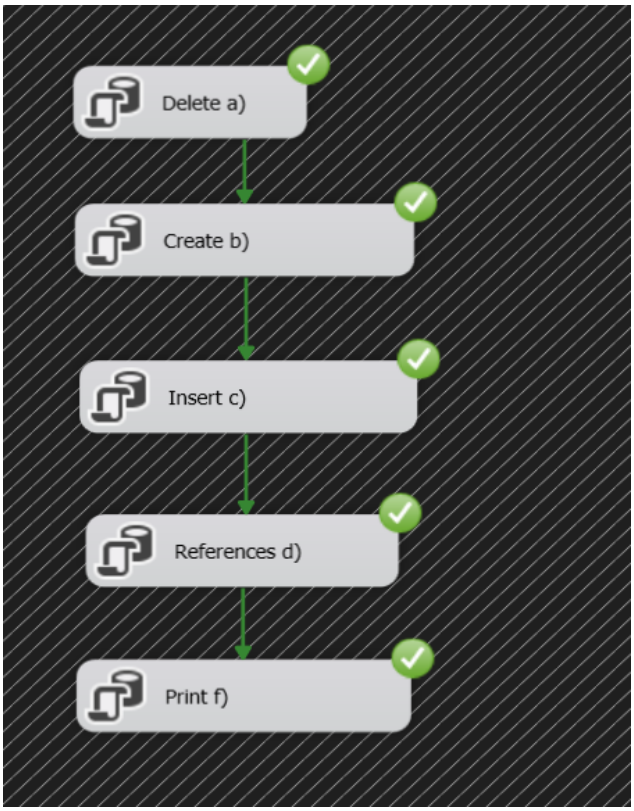
d)



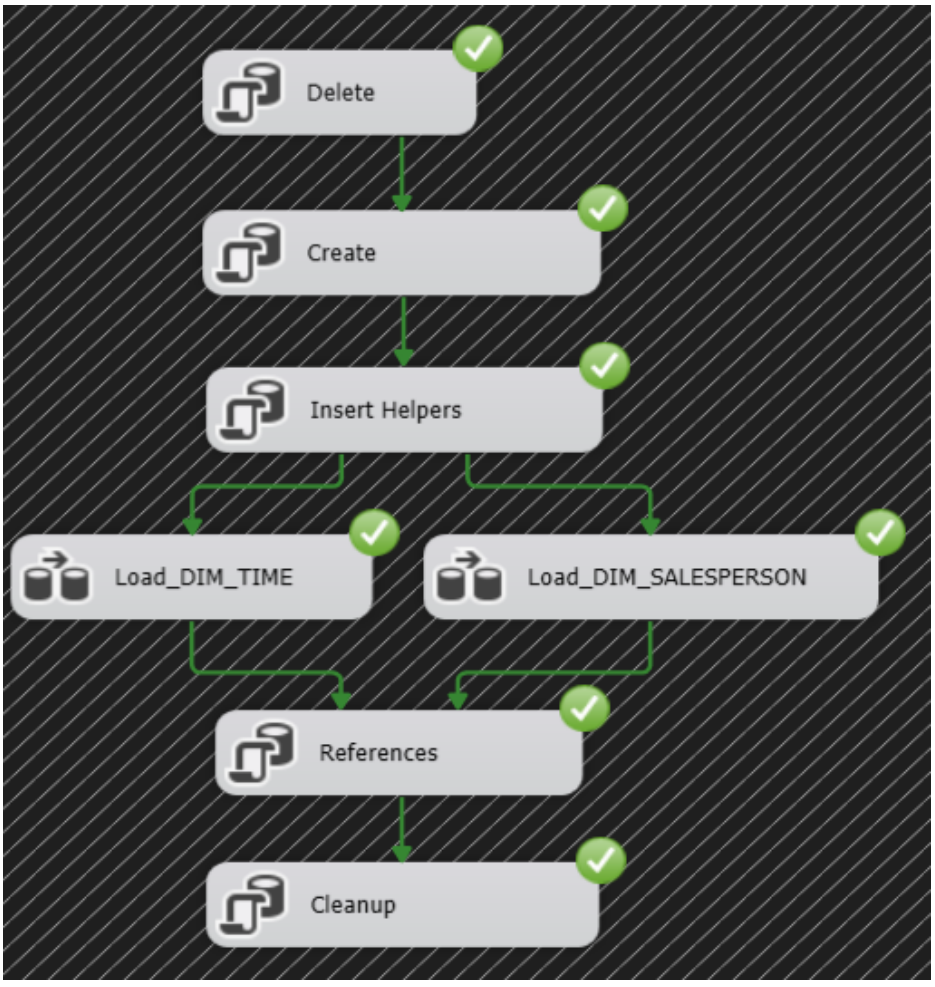
e)

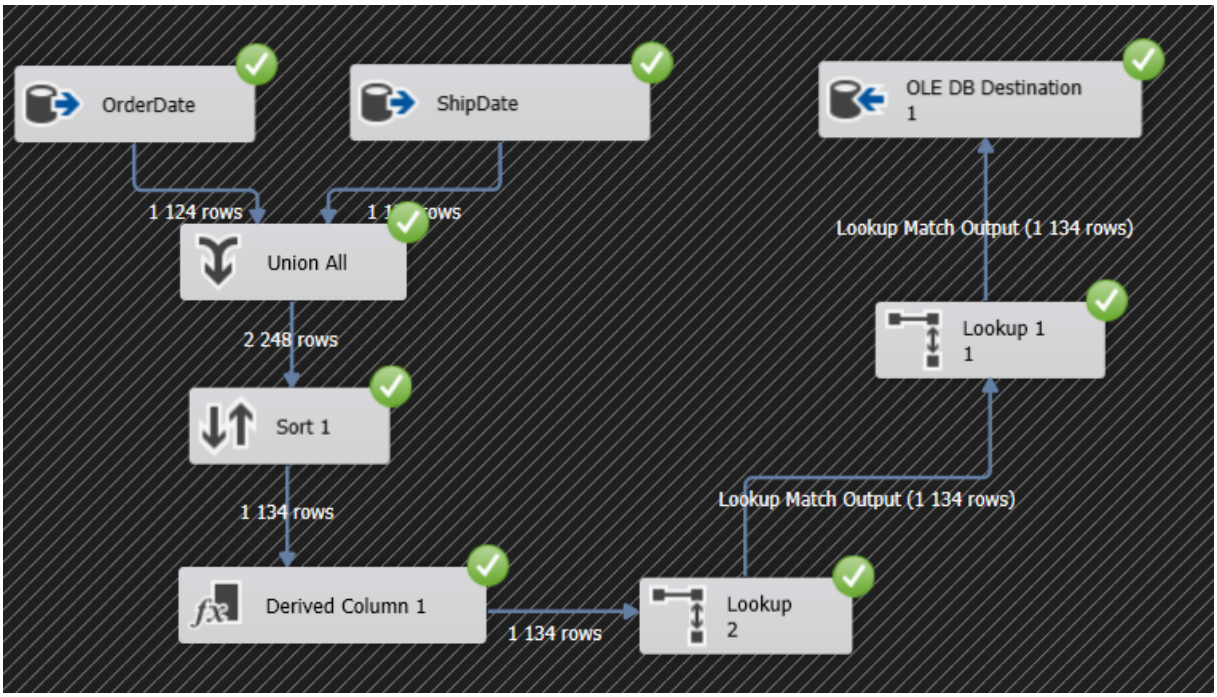


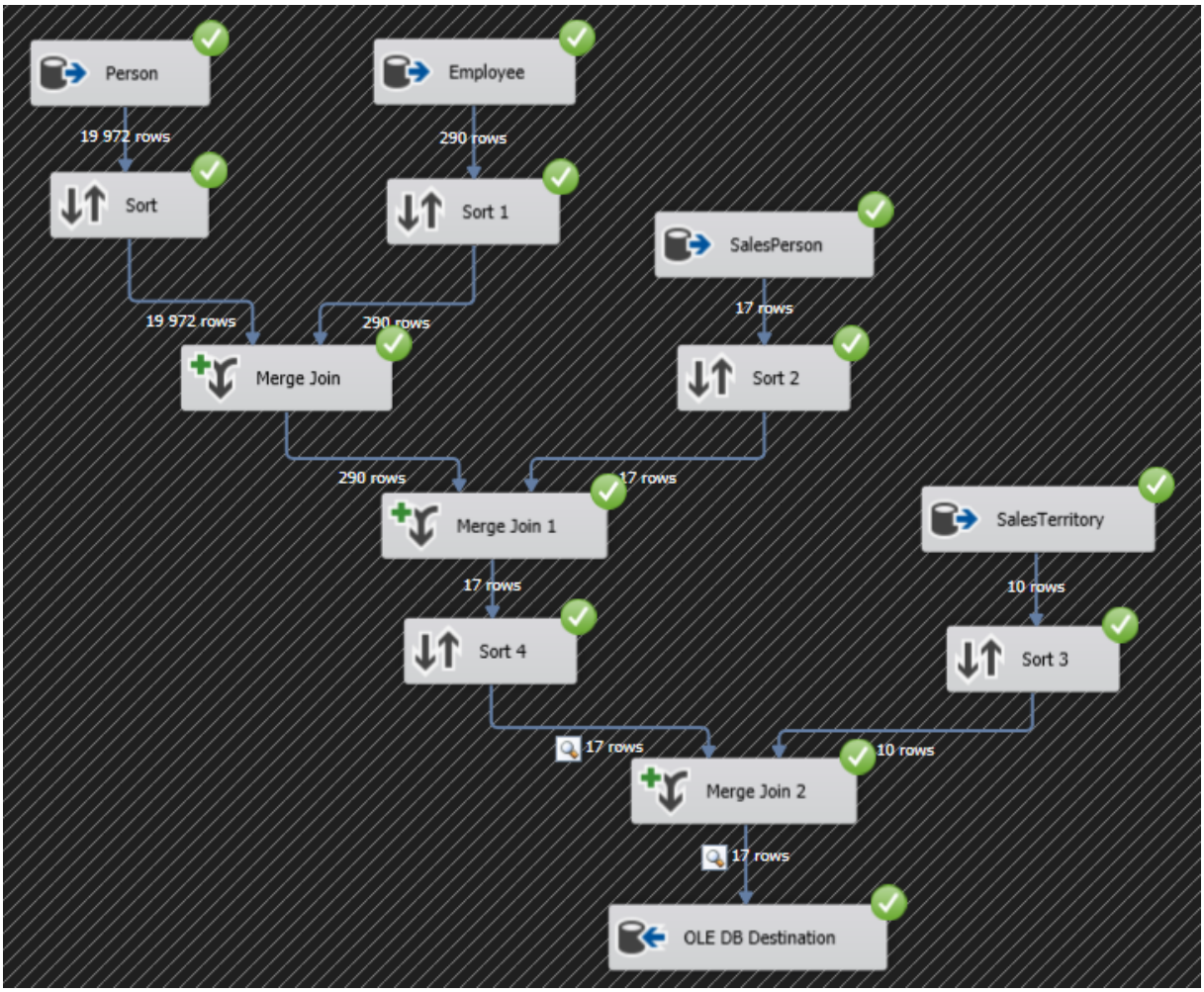
f) PRINT ‘Proces zakończony pomyślnie’;


Delete a)
Create b)
Insert c)
References d)
Print f)

Zadanie 5.


Delete
Create
Insert Helpers
Load_DIM_TIME
Load_DIM_SALESPERSON
References
Cleanup


OrderDate
ShipDate
OLE DB Destination 1
1 124 rows
Union All
Lookup Match Output (1 134 rows)
2 248 rows
Lookup 1 1
Sort 1
1 134 rows
Lookup Match Output (1 134 rows)
Derived Column 1
1 134 rows
Lookup 2


Person
Employee
19 972 rows
290 rows
Sort
Sort 1
SalesPerson
19 972 rows
290 rows
17 rows
Merge Join
Sort 2
290 rows
17 rows
Merge Join 1
SalesTerritory
17 rows
10 rows
Sort 4
Sort 3
17 rows
10 rows
Merge Join 2
17 rows
OLE DB Destination

Wnioski:

Napisanie skryptu, który automatycznie usuwa wszystkie tabele DIM i FACT, pozwoliło na to, że za każdym razem mamy czyste środowisko do pracy.

Przy próbie implementacji importowania danych do DIM_TIME w SSIS w ETL w formie noSQL moje pierwsze podejście obejmowało próbę obróbki dat ShipDate oraz OrderDate bez wielokrotnego pobierania źródła danych (np. za pomocą Unpivot), ale końcowo uznałem za prostsze (i działające) użycie Union All oraz dwóch źródeł danych.

W Derived Column nauczyłem się uważać na wartości NULL i najpierw je odfiltrować, żeby nie napotkać błędu przy rzutowaniu typów.

Korzystanie z Lookup, gdzie można było dołączhć nazwy miesięcy i dni pokazało, jak łatwo uzupełnić atrybuty bez pisania skomplikowanych zapytań.

Czyste zapytania SQL w Execute SQL Task dały nam pewność, że każda kolejna instrukcja wykona się w ustalonej kolejności, a ewentualny błąd zatrzyma proces dokładnie tam, gdzie trzeba – można też było łatwiej obserwować przebieg takiego procesu i jego ewentualna naprawa w przypadku błędu była o wiele prostsza niż dla klasycznego kodu w SQL.

Event Handlers służą do globalnego łapania wyjątków i logowania ich do plików, jest to rozwiązanie, które w niektórych sytuacjach może być wygodne i praktyczne, jednak brak domyślnego error outputu w konsoli mnie osobiście irytował i nie był zbyt intuicyjny przy próbie naprawy kodu.

Przy pracy nad tymi zadaniami przekonać się można, że komponenty SSIS potrafią naprawdę dużo, zwłaszcza gdy chcemy zbudować wygodny do wykorzystania prototyp ETL. Takie rozwiązanie pozwala łączyć szerokie możliwości języka SQL z wizualną wygodą graficznej reprezentacji w SSIS, co sprawia że końcowy projekt staje się bardziej solidny, elastyczny i odporny na typowe błędy danych. Jednak warto wspomnieć że robienie tego typu zapytań graficznie jest o wiele bardziej czasochłonne niż napisanie krótkiego kodu w czystym SQL.