

Final Task

Data Engineer – ID/X Partners

Presented by
Farradita Imanda



Farradita Imanda

About You

A Software Enthusiast who wants to learn many things. Majoring in Computer Engineering and graduate in 2020 at Telkom University.

Keywords : Curious, Detailed, and Determined

Work Experience

- National Institute of Aeronautics and Space
 - Designed website's mockup using Figma
 - Developed atmosphere website using HTML, CSS, Javascript, and Django framework
- PT POS Indonesia
 - Organized staff data
 - Input stationery office data into excel
 - Distributed pension fund in August 2020

Case Study

Salah satu client dari ID/X Partners yang bergerak di bidang e-commerce memiliki kebutuhan untuk membuat sebuah Data Warehouse yang berasal dari beberapa tabel dari database sumber. Data Warehouse ini nantinya terdiri dari satu tabel Fact dan beberapa tabel Dimension.

Case Study

Sebagai Data Engineer, ada beberapa task yang perlu anda lakukan yaitu :

1. Melakukan Import/Restore Database Staging.
2. Membuat sebuah Database bernama DWH_Project, serta membuat Tabel Fact dan Dimension dari tabel yang ada di database Staging.
3. Membuat Job ETL di aplikasi talend untuk memindahkan data dari Staging ke Data Warehouse. Khusus untuk Tabel DimCustomer, lakukan transformasi data dengan merubah data dari kolom FirstName dan LastName menjadi huruf kapital semua, lalu gabungkan kedua kolom tersebut menjadi satu kolom yang bernama CustomerName.
4. Membuat Store Procedure (SP) untuk menampilkan summary sales order berdasarkan status pengiriman

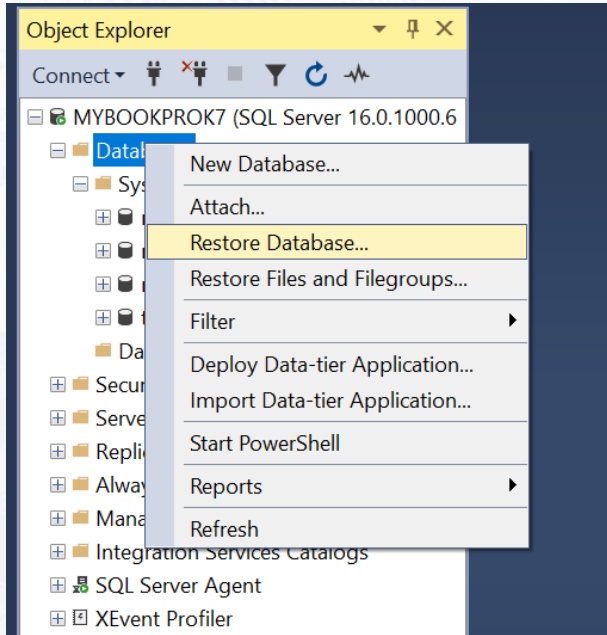
Result

1. Restore Database Staging

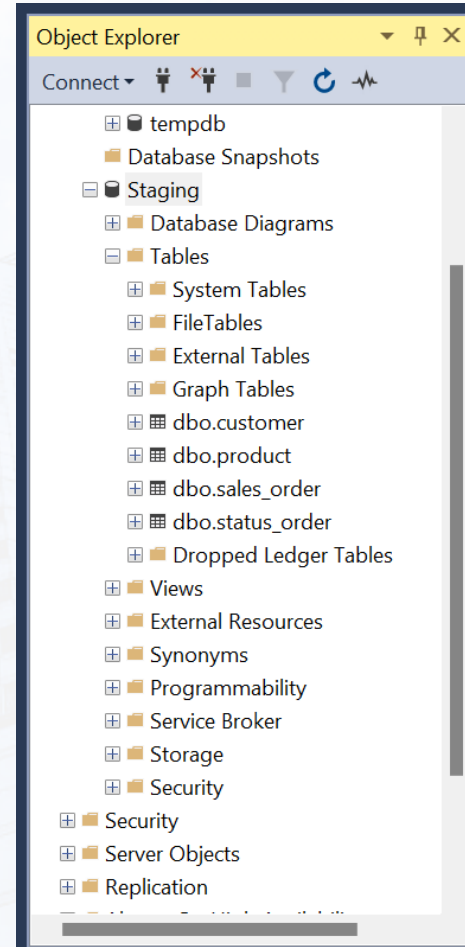
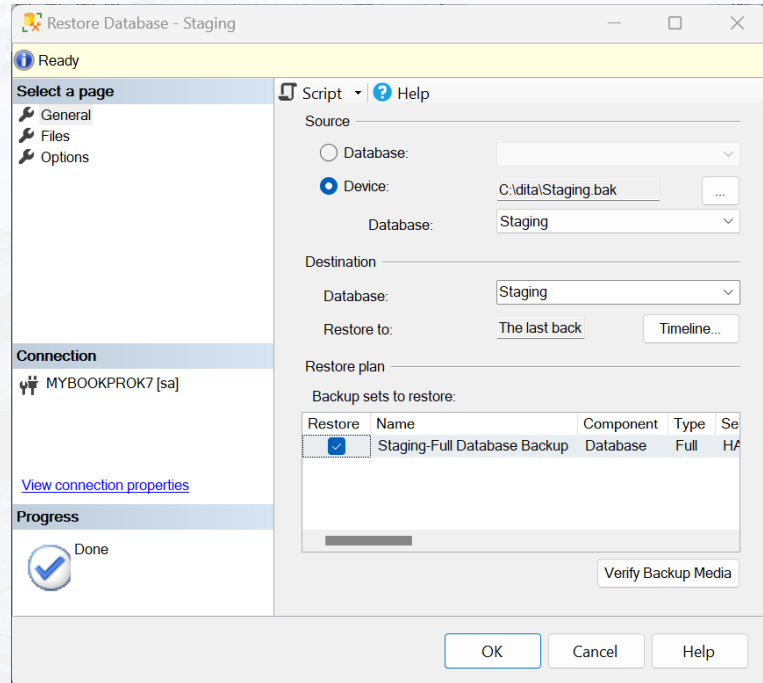
Untuk me-restore database, pertama kita buka dahulu SSMS, kemudian pada folder Database klik kanan pilih Restore Database

Setelah itu pilih device -> klik titik tiga di sampingnya -> klik add -> pilih lokasi dan file database yang ingin di restore -> setelah itu klik OK -> OK , tunggu hingga muncul pop up "Database restored suscessfully"

Proses dapat dilihat pula pada slide setelah ini



Restore Database Staging



Restore Database
Staging success

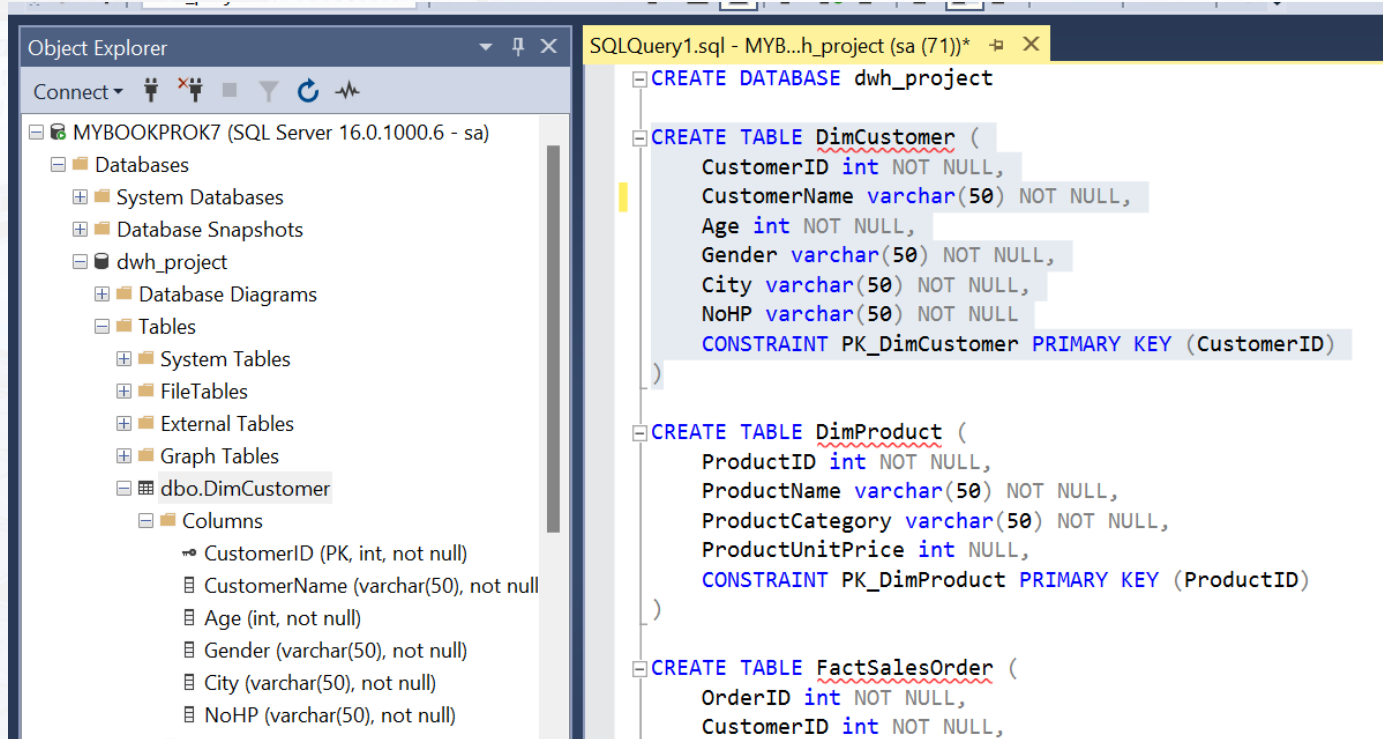
2. Membuat Database dwh_project serta Tabel Fact dan Dimesion

Setelah merestore database Staging, tugas selanjutnya yaitu membuat database bernama dwh_project dan membuat tabel fact dan dimension. Database dwh_project ini nanti akan berisi data dari database Staging

Untuk membuat database, tuliskan query sebagai berikut :

CREATE DATABASE dwh_project

Tabel Fact dan Dimension yang akan dibuat berjumlah 4 tabel, terdiri dari 1 Fact Table dan 3 Dimension Table. Isi kolom dari tabel-tabel ini disesuaikan dengan kolom pada database Staging. Berikut adalah query dari masing-masing tabel



The screenshot shows the SQL Server Enterprise Manager interface. On the left, the Object Explorer displays the hierarchy: MYBOOKPROK7 (SQL Server 16.0.1000.6 - sa) > Databases > dwh_project > Tables > dbo.DimCustomer. The table's columns are listed: CustomerID (PK, int, not null), CustomerName (varchar(50), not null), Age (int, not null), Gender (varchar(50), not null), City (varchar(50), not null), and NoHP (varchar(50), not null). On the right, the SQL Query window shows the following T-SQL script:

```
CREATE DATABASE dwh_project

CREATE TABLE DimCustomer (
    CustomerID int NOT NULL,
    CustomerName varchar(50) NOT NULL,
    Age int NOT NULL,
    Gender varchar(50) NOT NULL,
    City varchar(50) NOT NULL,
    NoHP varchar(50) NOT NULL
    CONSTRAINT PK_DimCustomer PRIMARY KEY (CustomerID)
)

CREATE TABLE DimProduct (
    ProductID int NOT NULL,
    ProductName varchar(50) NOT NULL,
    ProductCategory varchar(50) NOT NULL,
    ProductUnitPrice int NULL,
    CONSTRAINT PK_DimProduct PRIMARY KEY (ProductID)
)

CREATE TABLE FactSalesOrder (
    OrderID int NOT NULL,
    CustomerID int NOT NULL,
```

Tabel
DimCustomer

Tabel
DimProduct



```
SQLQuery1.sql - MYB...h_project (sa (56))*  
NoHP varchar(50) NOT NULL  
CONSTRAINT PK_DimCustomer PRIMARY KEY (CustomerID)  
  
CREATE TABLE DimProduct (  
    ProductID int NOT NULL,  
    ProductName varchar(50) NOT NULL,  
    ProductCategory varchar(50) NOT NULL,  
    ProductUnitPrice int NULL,  
    CONSTRAINT PK_DimProduct PRIMARY KEY (ProductID)  
)  
  
CREATE TABLE FactSalesOrder (  
    OrderID int NOT NULL,
```

100 %

Messages

Commands completed successfully.

Completion time: 2023-09-03T15:11:01.5336427+07:00

```
MYB...h_project (sa (56))*  
erDate date NOT NULL,  
CONSTRAINT PK_FactSalesOrder PRIMARY KEY (OrderID),  
CONSTRAINT FK_SalesCustomer FOREIGN KEY (CustomerID) REFEREN  
CONSTRAINT FK_SalesProduct FOREIGN KEY (ProductID) REFERENCE  
CONSTRAINT FK_SalesStatus FOREIGN KEY (StatusID) REFERENCES
```

Tabel
DimStatusOrder



Tables

- System Tables
- FileTables
- External Tables
- Graph Tables
- dbo DimCustomer
- dbo DimProduct
- dbo DimStatusOrder

Columns

- StatusID (PK, int, not null)
- StatusOrder (varchar(50), not null)
- StatusOrderDesc (varchar(50), not null)

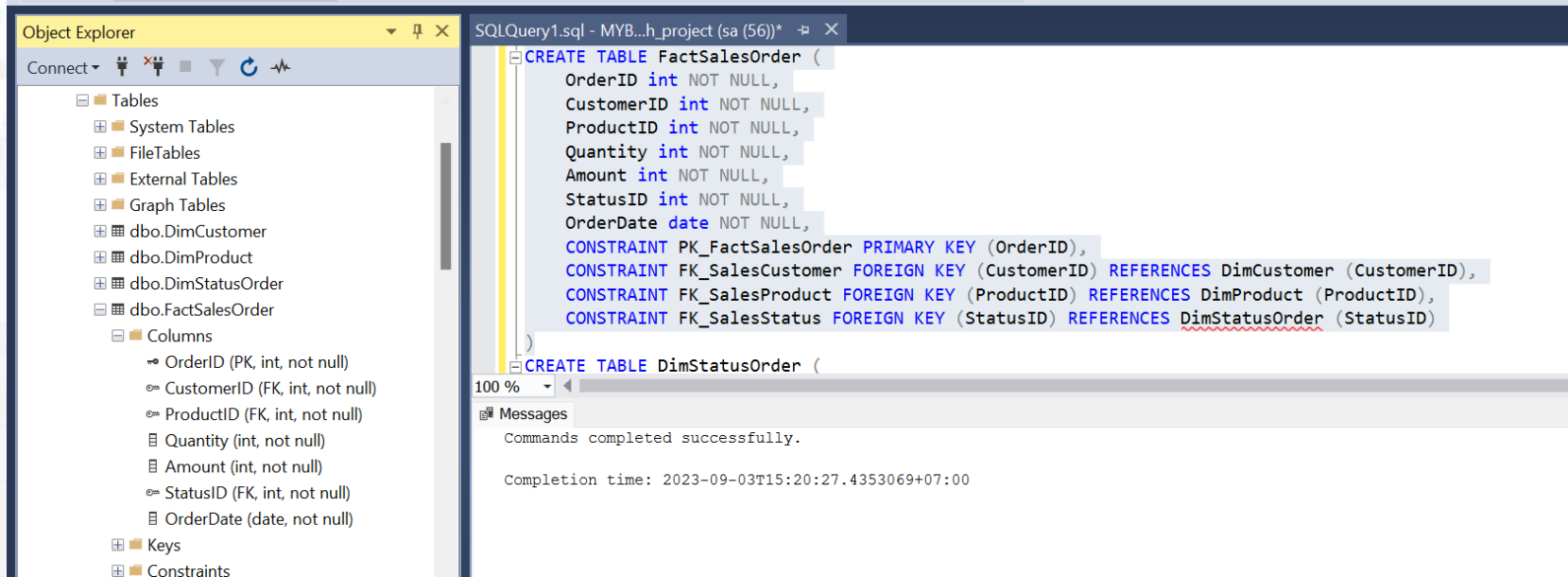
Keys

```
)  
CREATE TABLE DimStatusOrder (  
    StatusID int NOT NULL,  
    StatusOrder varchar(50) NOT NULL,  
    StatusOrderDesc varchar(50) NOT NULL,  
    CONSTRAINT PK_DimStatusOrder PRIMARY KEY (StatusID)  
)  
  
100 %
```

Messages

Commands completed successfully.

Completion time: 2023-09-03T15:19:34.3096103+07:00



The screenshot displays the SQL Server Enterprise Manager interface. On the left, the Object Explorer shows the database structure, including tables and columns. The main pane shows the SQL query editor with the following code:

```
CREATE TABLE FactSalesOrder (  
    OrderID int NOT NULL,  
    CustomerID int NOT NULL,  
    ProductID int NOT NULL,  
    Quantity int NOT NULL,  
    Amount int NOT NULL,  
    StatusID int NOT NULL,  
    OrderDate date NOT NULL,  
    CONSTRAINT PK_FactSalesOrder PRIMARY KEY (OrderID),  
    CONSTRAINT FK_SalesCustomer FOREIGN KEY (CustomerID) REFERENCES DimCustomer (CustomerID),  
    CONSTRAINT FK_SalesProduct FOREIGN KEY (ProductID) REFERENCES DimProduct (ProductID),  
    CONSTRAINT FK_SalesStatus FOREIGN KEY (StatusID) REFERENCES DimStatusOrder (StatusID)  
)  
  
CREATE TABLE DimStatusOrder (  
    StatusID int NOT NULL,  
    StatusName varchar(50) NOT NULL,  
    OrderDate date NOT NULL,  
    CONSTRAINT PK_DimStatusOrder PRIMARY KEY (StatusID)  
)
```

The Messages pane at the bottom indicates that the commands were completed successfully.

Completion time: 2023-09-03T15:20:27.4353069+07:00

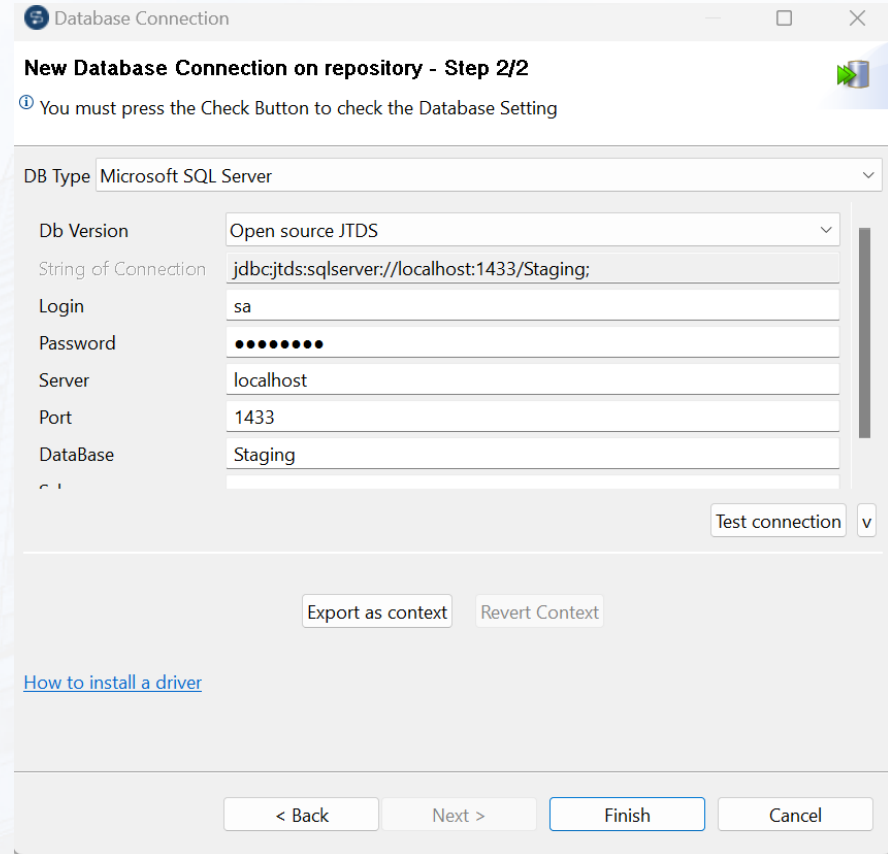
Tabel FactSalesOrder

3. Membuat Job ETL di Talend

Job ETL dapat dibuat pada aplikasi Talend.
Job yang akan dibuat berfungsi untuk
menyalin data yang ada pada tabel di
database Staging ke tabel pada database
dwh_project

Sebelum mulai membuat job, kita harus
membuat koneksi pada db connection.
Caranya yaitu klik kanan dan pilih create
connection. Isi nama db connection,
kemudian isi kolom yang diperlukan.

Isian kolom dapat dilihat pada gambar di samping ini



3. Membuat Job ETL di Talend

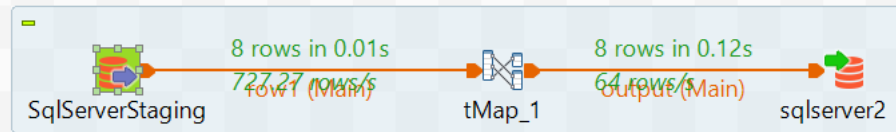
Setelah membuat db connection untuk database Staging dan dwh_project, selanjutnya yaitu membuat job. Mula-mula kita akan membuat folder baru. Setelah folder dibuat, klik kanan pada folder dan pilih create job lalu isi nama job lalu klik finish.

Pada task ini, kita akan membuat 4 job diantaranya CustomerJob, ProductJob, StatusJob, dan SalesOrderJob.

Setelah itu, kita drag and drop database untuk input dan output.

Sebagai input, kita drag database Staging dan dwh_project sebagai output. Disini kita menggunakan komponen **tMSSqlInput** dan **tMSSqlOutput** untuk memindahkan data dari source (staging) ke target (dwh_project)

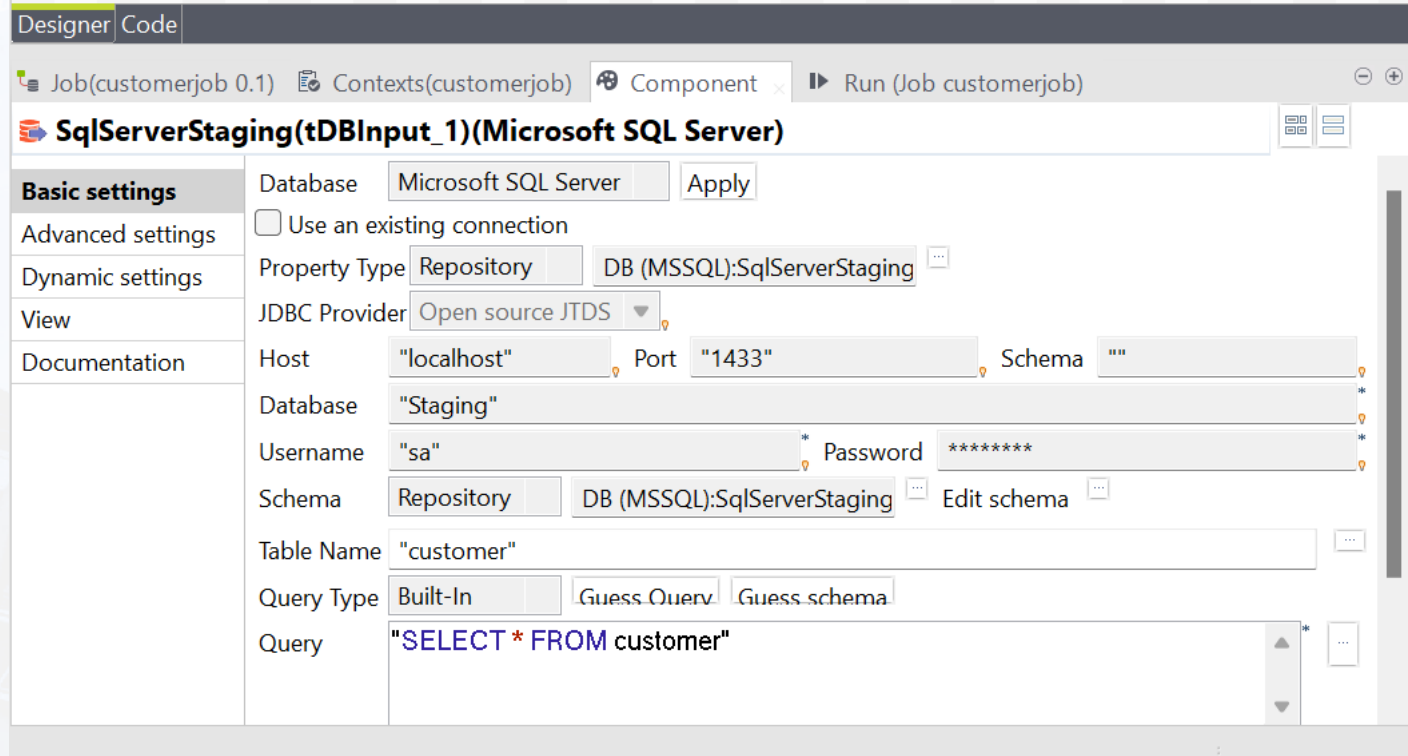
Maka tampilannya akan seperti ini :



*Note: tulisan row dan waktu tersebut didapat setelah running

Pada database source (Staging), klik 2x untuk mengatur komponennya, maka akan muncul seperti tulisan di bawah ini

Isi Database, table name, dan query sesuai dengan data source yang ingin kita copy.
Pada schema pilih Repository dan table schemas yang ingin di-copy. Berikut ini adalah untuk CustomerJob



The screenshot shows a configuration window titled "SqlServerStaging(tDBInput_1)(Microsoft SQL Server)". The window has a "Designer" tab and a "Code" tab. The "Basic settings" tab is selected, showing the following configuration:

- Database: Microsoft SQL Server
- Property Type: Repository
- JDBC Provider: Open source JTDS
- Host: localhost
- Port: 1433
- Schema: ""
- Database: "Staging"
- Username: "sa"
- Password: "*****"
- Schema: Repository
- Table Name: "customer"
- Query Type: Built-In
- Query: "SELECT * FROM customer"

Khusus untuk CustomerJob, kita menggunakan komponen tMap yang ditujukan untuk mentransformasi data.

Berikut adalah hasil dari tMap yang sudah disesuaikan dengan kebutuhan kita.

Talend Open Studio for Data Integration - tMap - tMap_1

Find :

Var

Auto map!

Expression	Column
row1.customer_id	CustomerID
StringHandling.UPCASE(row1.first_name)	CustomerName
row1.age	Age
row1.gender	Gender
row1.city	City
row1.no_hp	NoHP

Schema editor

Expression editor

row1

Column	K...	Type	<input checked="" type="checkbox"/> N	Date Patt...	Leng...	Preci...	Def...	Com...
customer_id	<input checked="" type="checkbox"/>	int	<input type="checkbox"/>		10	0		
first_name	<input type="checkbox"/>	String	<input type="checkbox"/>		50	0		
last_name	<input type="checkbox"/>	String	<input type="checkbox"/>		50	0		
age	<input type="checkbox"/>	int	<input type="checkbox"/>		10	0		
gender	<input type="checkbox"/>	String	<input type="checkbox"/>		50	0		
city	<input type="checkbox"/>	String	<input type="checkbox"/>		50	0		
no_hp	<input type="checkbox"/>	String	<input type="checkbox"/>		50	0		

output

Column	K...	Type	<input checked="" type="checkbox"/> N	Date Patt...	Leng...	Preci...	Def...	Com...
CustomerID	<input checked="" type="checkbox"/>	int	<input type="checkbox"/>		10	0		
CustomerName	<input type="checkbox"/>	String	<input type="checkbox"/>		50	0		
Age	<input type="checkbox"/>	int	<input type="checkbox"/>		10	0		
Gender	<input type="checkbox"/>	String	<input type="checkbox"/>		50	0		
City	<input type="checkbox"/>	String	<input type="checkbox"/>		50	0		
NoHP	<input type="checkbox"/>	String	<input type="checkbox"/>		50	0		

Expression Builder

Expression

☒ Wrap

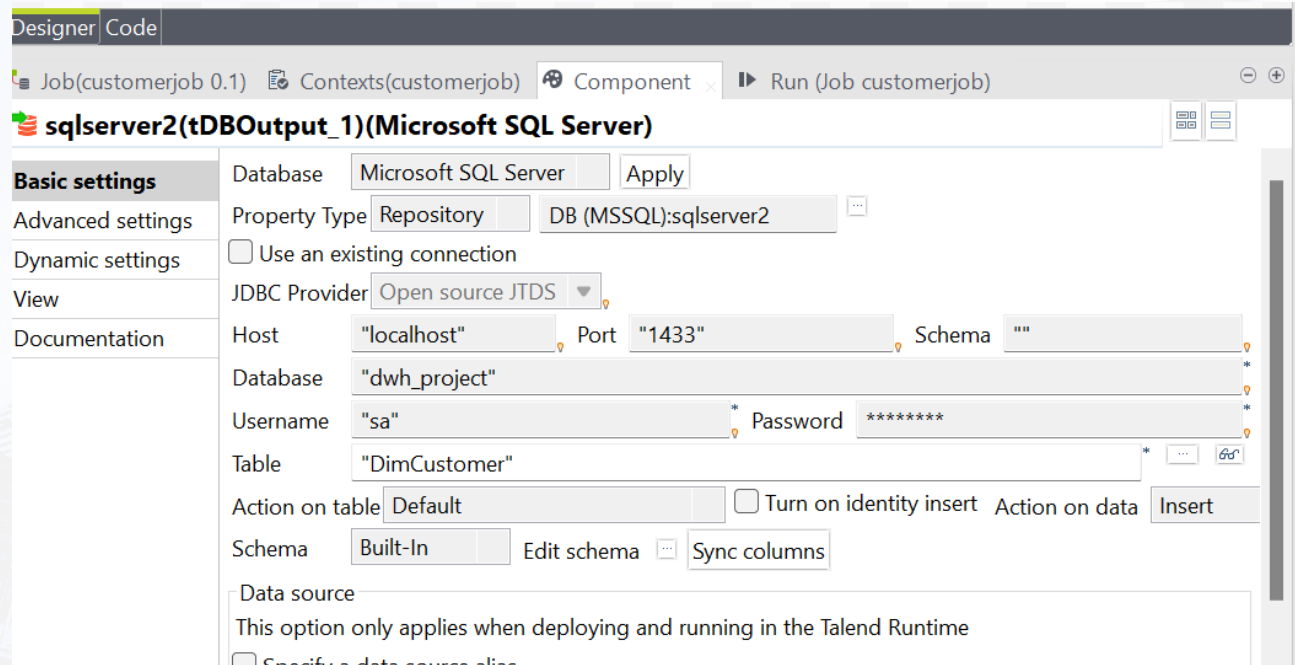
```
StringHandling.UPCASE
(row1.first_name) + " " +
StringHandling.UPCASE
(row1.last_name)
```

Categories

Pada database target (dwh_project), klik 2x untuk mengatur komponennya, maka akan muncul seperti tulisan di bawah ini

Isi Database, table name dengan database dan tabel target yang ingin diisi datanya.

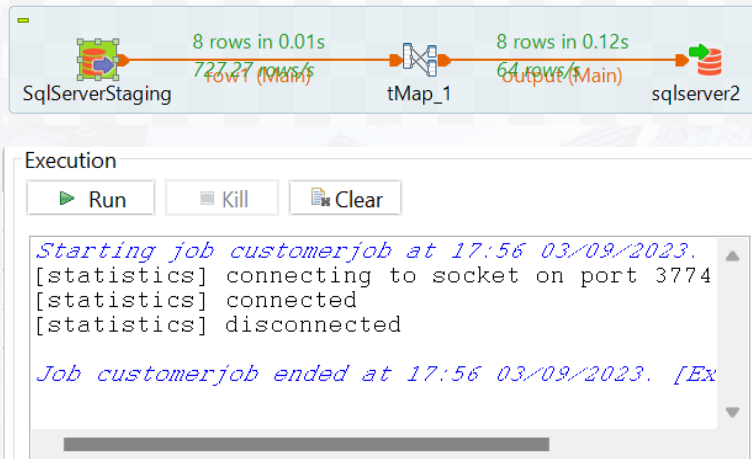
Di samping ini adalah komponen dari database output



The screenshot shows the configuration window for the 'sqlserver2(tDBOutput_1)(Microsoft SQL Server)' component in Talend Designer. The window has a 'Designer' tab and a 'Code' tab. The 'Basic settings' tab is selected, showing the following configuration:

- Database: Microsoft SQL Server (with an 'Apply' button)
- Property Type: Repository (with a dropdown arrow)
- DB (MSSQL): sqlserver2 (with a dropdown arrow)
- ☐ Use an existing connection
- JDBC Provider: Open source JTDS (with a dropdown arrow)
- Host: "localhost" (with a dropdown arrow)
- Port: "1433" (with a dropdown arrow)
- Schema: "" (with a dropdown arrow)
- Database: "dwh_project" (with a dropdown arrow)
- Username: "sa" (with a dropdown arrow)
- Password: "*****" (with a dropdown arrow)
- Table: "DimCustomer" (with a dropdown arrow and a '...' button)
- Action on table: Default (with a dropdown arrow)
- ☐ Turn on identity insert
- Action on data: Insert (with a dropdown arrow)
- Schema: Built-In (with a dropdown arrow)
- Edit schema: ... (with a dropdown arrow)
- Sync columns: (with a dropdown arrow)
- Data source: This option only applies when deploying and running in the Talend Runtime
- ☐ Specify a data source alias

Selanjutnya lakukan run job dengan mengklik Run, maka akan muncul gambar seperti di kiri bawah ini. Untuk mengecek apakah data sudah terisi, kita bisa pergi ke SSMS kemudian ketik query `SELECT * FROM DimCustomer` lalu execute query, jika sukses maka akan muncul tabel seperti di kanan bawah ini.



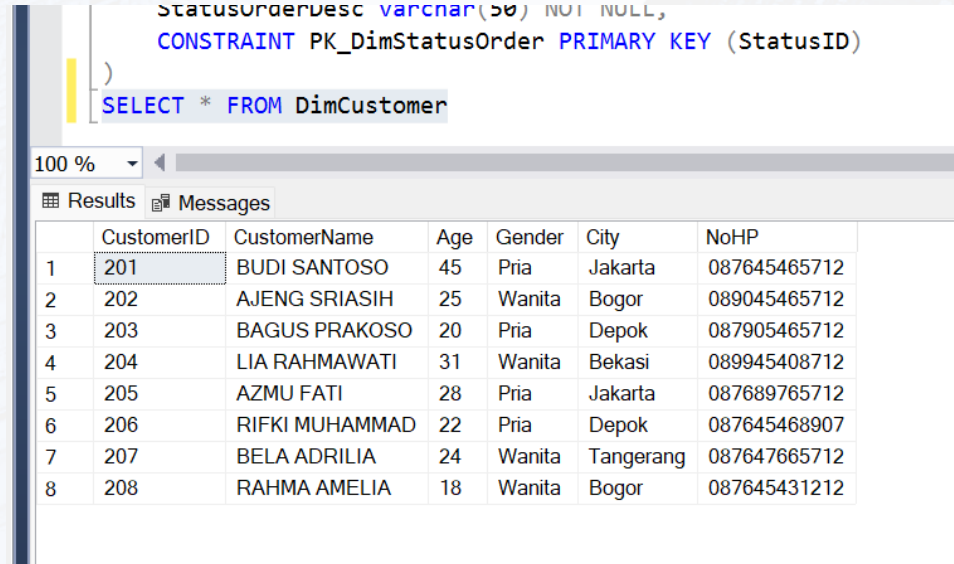
The screenshot shows an SSIS job execution progress bar with three tasks: 'SqlServerStaging' (8 rows in 0.01s), 'tMap_1' (8 rows in 0.12s), and 'sqlserver2'. Below the progress bar is the 'Execution' window showing the job status and logs.

Execution

Run Kill Clear

Starting job customerjob at 17:56 03/09/2023.
 [statistics] connecting to socket on port 3774
 [statistics] connected
 [statistics] disconnected
 Job customerjob ended at 17:56 03/09/2023. [Ex

Sukses running job



The screenshot shows a SQL Server query window with the query `SELECT * FROM DimCustomer` and its results. The results are displayed in a table with 8 rows and 7 columns.

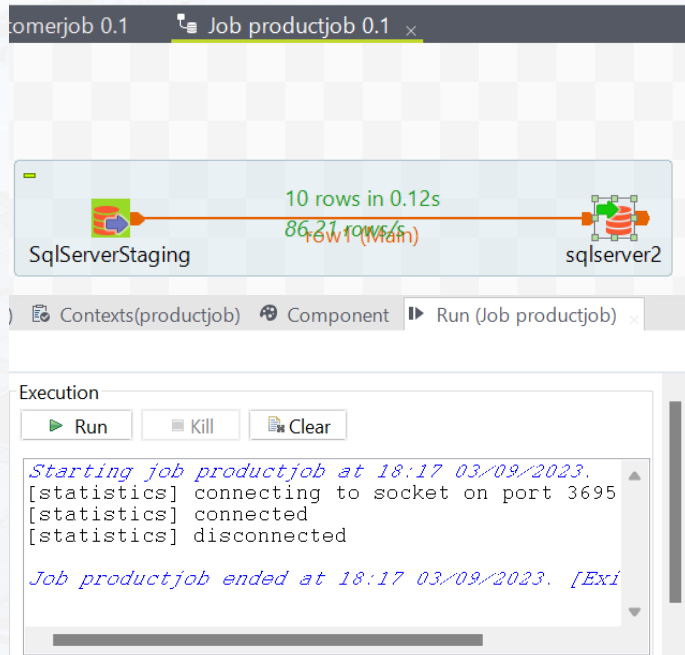
100 %

Results Messages

	CustomerID	CustomerName	Age	Gender	City	NoHP
1	201	BUDI SANTOSO	45	Pria	Jakarta	087645465712
2	202	AJENG SRIASIH	25	Wanita	Bogor	089045465712
3	203	BAGUS PRAKOSO	20	Pria	Depok	087905465712
4	204	LIA RAHMAWATI	31	Wanita	Bekasi	089945408712
5	205	AZMU FATI	28	Pria	Jakarta	087689765712
6	206	RIFKI MUHAMMAD	22	Pria	Depok	087645468907
7	207	BELA ADRILIA	24	Wanita	Tangerang	087647665712
8	208	RAHMA AMELIA	18	Wanita	Bogor	087645431212

DimCustomer Table output

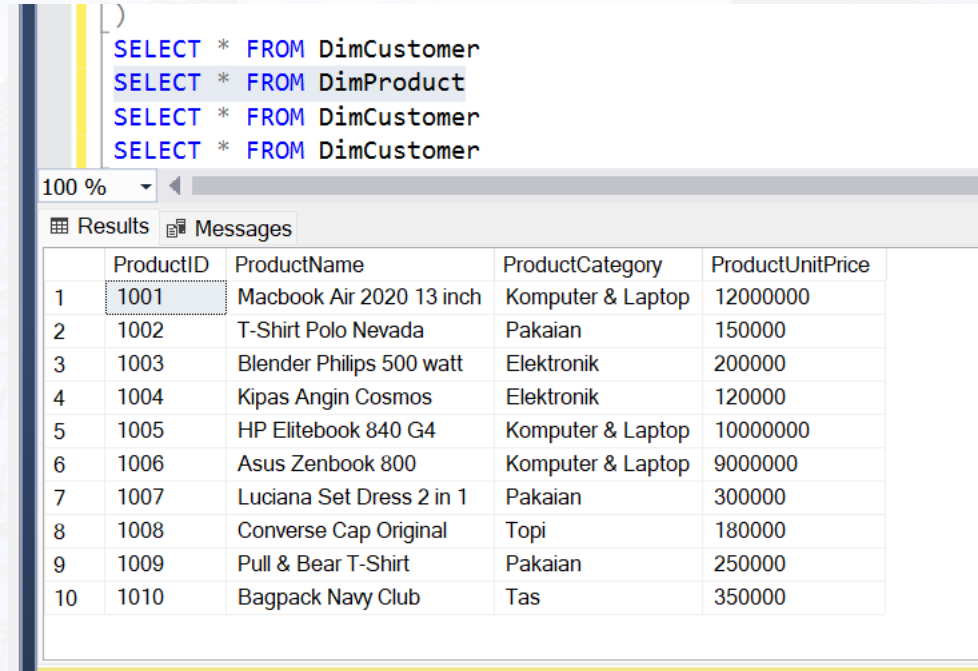
Pembuatan Job untuk Product, SalesOrder, dan StatusOrder sama seperti Job Customer, tetapi tidak memakai tMap. Berikut adalah ProductJob dan outputnya



The screenshot shows the 'Job productjob 0.1' window. It displays a flow from 'SqlServerStaging' to 'sqlserver2'. The execution progress bar indicates '10 rows in 0.12s' and '86.21 rows/s'. Below the flow, the 'Execution' panel shows the following log:

```
Starting job productjob at 18:17 03/09/2023.
[statistics] connecting to socket on port 3695
[statistics] connected
[statistics] disconnected
Job productjob ended at 18:17 03/09/2023. [Exi
```

Sukses running job



The screenshot shows the SQL query results for the 'DimProduct' table. The query is:

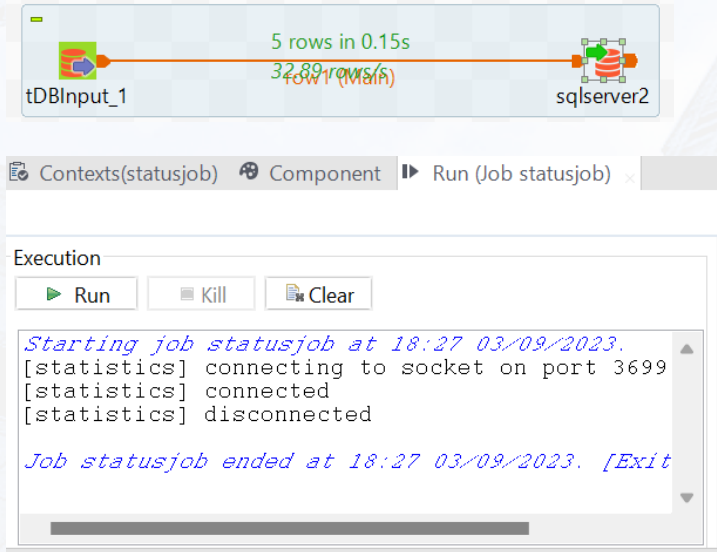
```
SELECT * FROM DimCustomer
SELECT * FROM DimProduct
SELECT * FROM DimCustomer
SELECT * FROM DimCustomer
```

The results are displayed in a table with the following columns: ProductID, ProductName, ProductCategory, and ProductUnitPrice. The table contains 10 rows of data.

	ProductID	ProductName	ProductCategory	ProductUnitPrice
1	1001	Macbook Air 2020 13 inch	Komputer & Laptop	12000000
2	1002	T-Shirt Polo Nevada	Pakaian	150000
3	1003	Blender Philips 500 watt	Elektronik	200000
4	1004	Kipas Angin Cosmos	Elektronik	120000
5	1005	HP Elitebook 840 G4	Komputer & Laptop	10000000
6	1006	Asus Zenbook 800	Komputer & Laptop	9000000
7	1007	Luciana Set Dress 2 in 1	Pakaian	300000
8	1008	Converse Cap Original	Topi	180000
9	1009	Pull & Bear T-Shirt	Pakaian	250000
10	1010	Bagpack Navy Club	Tas	350000

DimProduct Table output

Gambar untuk StatusOrder Job dan output DimStatusOrder Table



5 rows in 0.15s
32.89 rows/s

tDBInput_1 sqlserver2

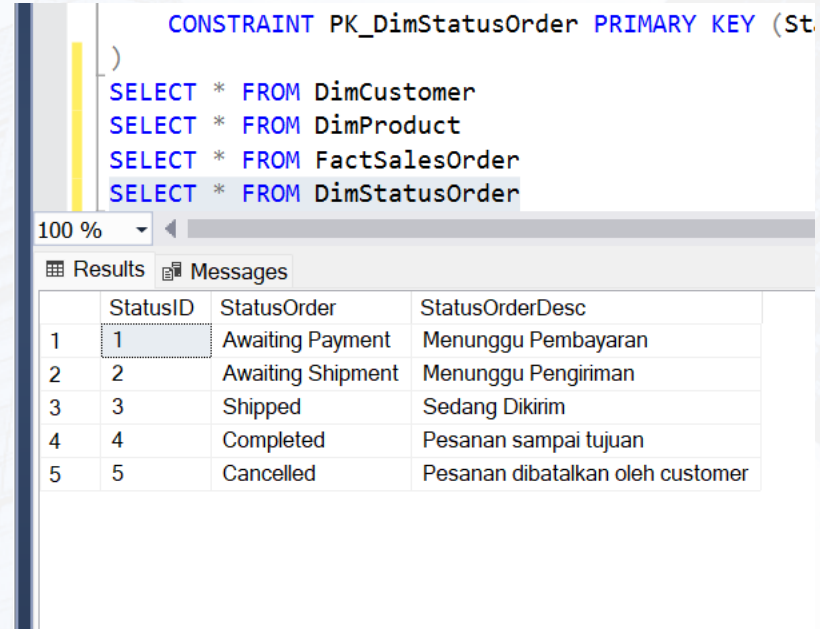
Contexts(statusjob) Component ▶ Run (Job statusjob)

Execution

Run Kill Clear

Starting job statusjob at 18:27 03/09/2023.
[statistics] connecting to socket on port 3699
[statistics] connected
[statistics] disconnected
Job statusjob ended at 18:27 03/09/2023. [Exit]

Sukses running job



```
CONSTRAINT PK_DimStatusOrder PRIMARY KEY (StatusID)
SELECT * FROM DimCustomer
SELECT * FROM DimProduct
SELECT * FROM FactSalesOrder
SELECT * FROM DimStatusOrder
```

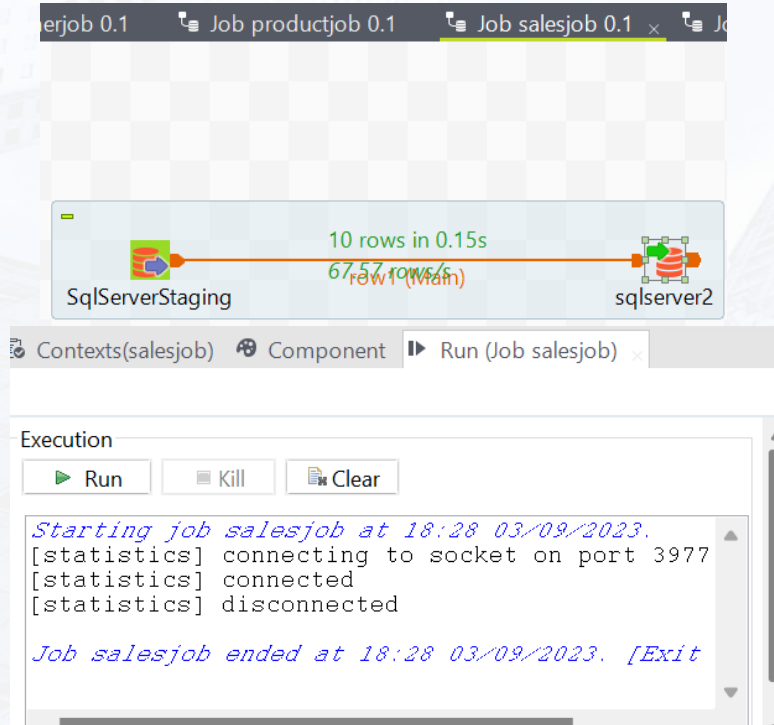
100 %

Results Messages

	StatusID	StatusOrder	StatusOrderDesc
1	1	Awaiting Payment	Menunggu Pembayaran
2	2	Awaiting Shipment	Menunggu Pengiriman
3	3	Shipped	Sedang Dikirim
4	4	Completed	Pesanan sampai tujuan
5	5	Cancelled	Pesanan dibatalkan oleh customer

DimStatusOrder Table output

Gambar SalesOrderJob dan output dari FactSalesOrder Table



Job salesjob 0.1 x

10 rows in 0.15s
67.57 rows/s
row1 (Main)

SqlServerStaging sqlserver2

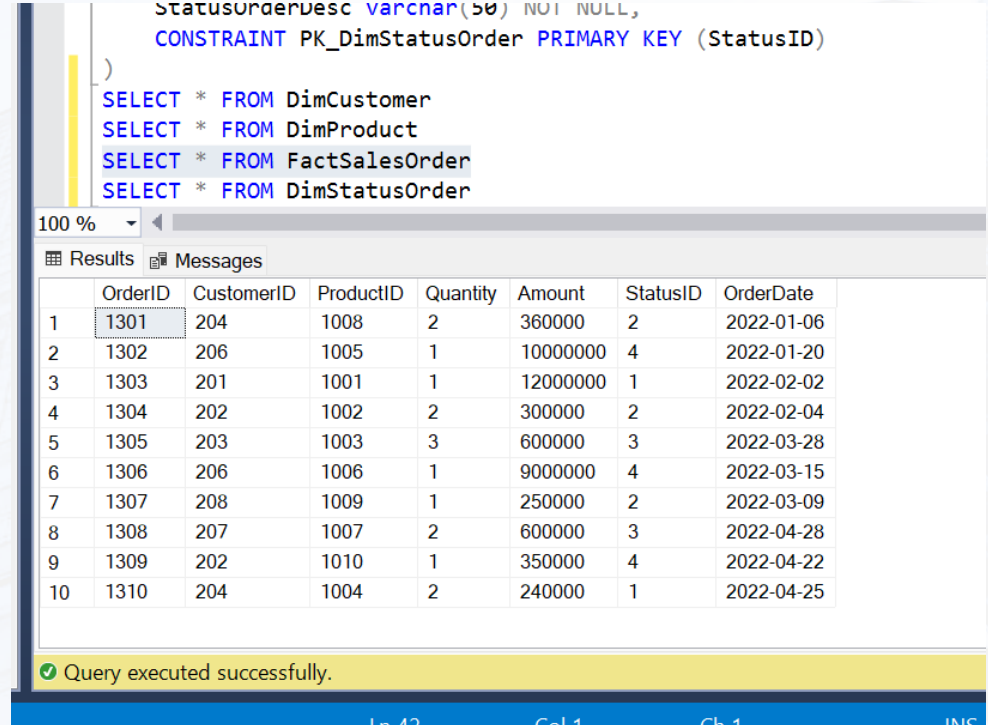
Contexts(salesjob) Component Run (Job salesjob)

Execution

Run Kill Clear

Starting job salesjob at 18:28 03/09/2023.
[statistics] connecting to socket on port 3977
[statistics] connected
[statistics] disconnected
Job salesjob ended at 18:28 03/09/2023. [Exit]

Sukses running job



```

StatusOrderDesc varchar(50) NOT NULL,
CONSTRAINT PK_DimStatusOrder PRIMARY KEY (StatusID)
)
SELECT * FROM DimCustomer
SELECT * FROM DimProduct
SELECT * FROM FactSalesOrder
SELECT * FROM DimStatusOrder
    
```

100 %

Results Messages

	OrderID	CustomerID	ProductID	Quantity	Amount	StatusID	OrderDate
1	1301	204	1008	2	360000	2	2022-01-06
2	1302	206	1005	1	10000000	4	2022-01-20
3	1303	201	1001	1	12000000	1	2022-02-02
4	1304	202	1002	2	300000	2	2022-02-04
5	1305	203	1003	3	600000	3	2022-03-28
6	1306	206	1006	1	9000000	4	2022-03-15
7	1307	208	1009	1	250000	2	2022-03-09
8	1308	207	1007	2	600000	3	2022-04-28
9	1309	202	1010	1	350000	4	2022-04-22
10	1310	204	1004	2	240000	1	2022-04-25

Query executed successfully.

FactSalesOrder Table output

4. Membuat Store Procedure Untuk Menampilkan Summary Sales Order Berdasarkan Status Pengiriman

Dengan tabel-tabel pada database dwh_project telah terisi, kita bisa membuat suatu perintah berdasarkan kondisi tertentu. Kita dapat mencapainya dengan menggunakan Store Procedure.

Untuk tugas kali ini, kita akan membuat Store Procedure untuk menampilkan Summary Sales Order berdasarkan status pengiriman. Query dari Store Procedure tersebut dapat dilihat pada slide berikut ini

```
SELECT FROM DimStatusOrder  
  
CREATE OR ALTER PROCEDURE summary_order_status  
    (@StatusID int) AS  
BEGIN  
    SELECT  
        a.OrderID,  
        b.CustomerName,  
        c.ProductName,  
        a.Quantity,  
        d.StatusOrder  
    FROM FactSalesOrder a  
    INNER JOIN DimCustomer b ON a.CustomerID = b.CustomerID  
    INNER JOIN DimProduct c ON a.ProductID = c.ProductID  
    INNER JOIN DimStatusOrder d ON a.StatusID = d.StatusID  
    WHERE d.StatusID = @StatusID  
END  
  
EXEC summary_order_status @StatusID = 2
```

	OrderID	CustomerName	ProductName	Quantity	StatusOrder
1	1301	LIA RAHMAWATI	Converse Cap Original	2	Awaiting Shipment
2	1304	AJENG SRIASIH	T-Shirt Polo Nevada	2	Awaiting Shipment
3	1307	RAHMA AMELIA	Pull & Bear T-Shirt	1	Awaiting Shipment

Kita menggunakan perintah SELECT dan INNER JOIN untuk menampilkan kolom OrderID, CustomerName, ProductName, Quantity, dan StatusOrder.

Untuk menjalankan Store Procedure dimana StatusID = 2, maka query dapat dituliskan seperti ini : EXEC summary_order_status @statusID = 2

Link Project

Github : <https://github.com/farraditai/Final-Task-VIX-Data-Engineer/tree/main>

Video Presentation

<https://drive.google.com/drive/folders/1go1lkBygEjvCKNK5MbzNiv9V2KvAof4Z?usp=sharing>

Thank You

