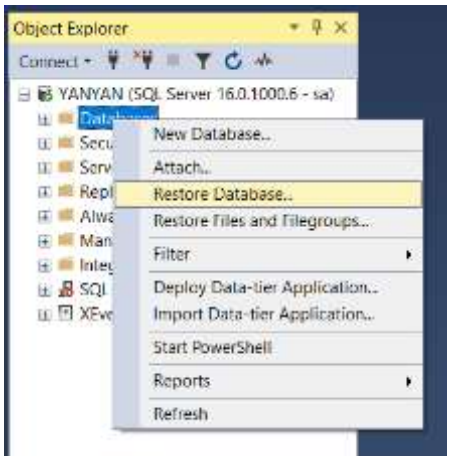
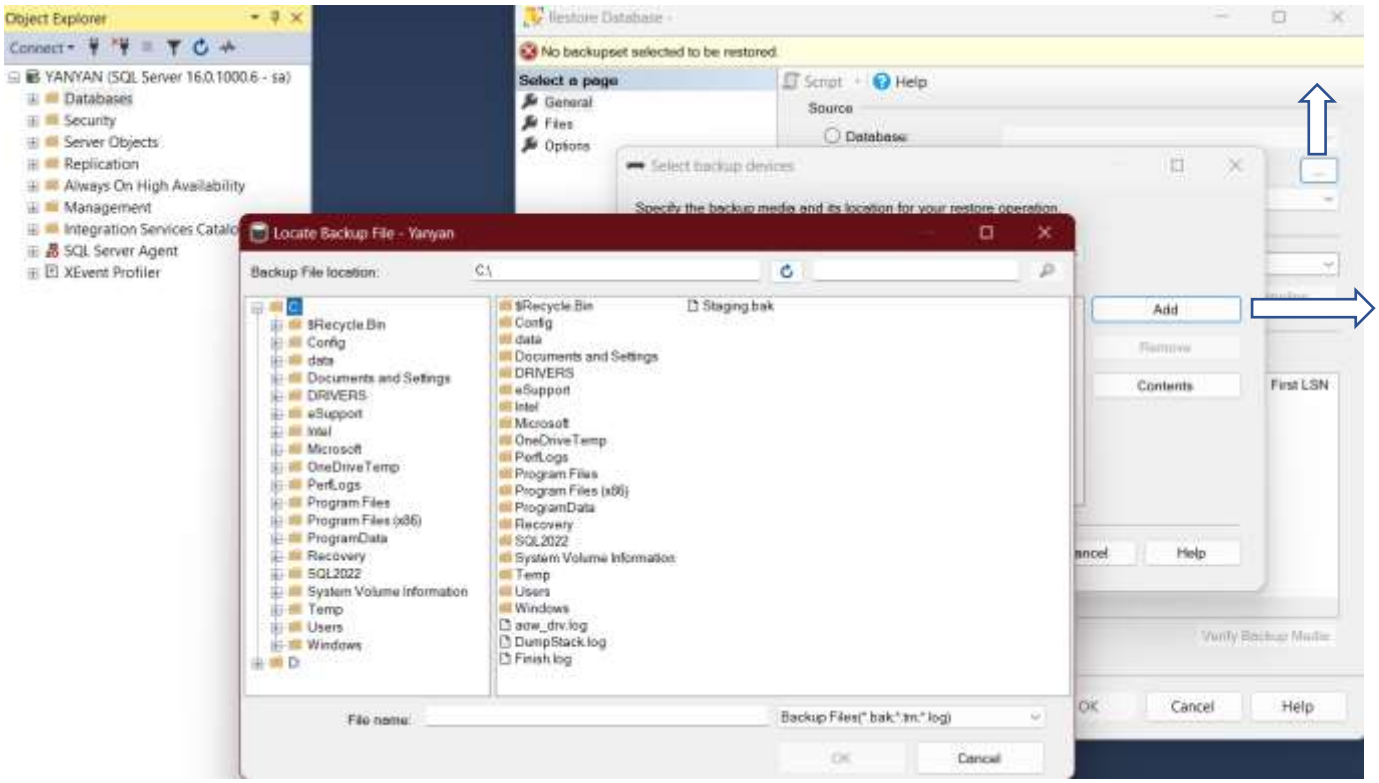
Step Final Project

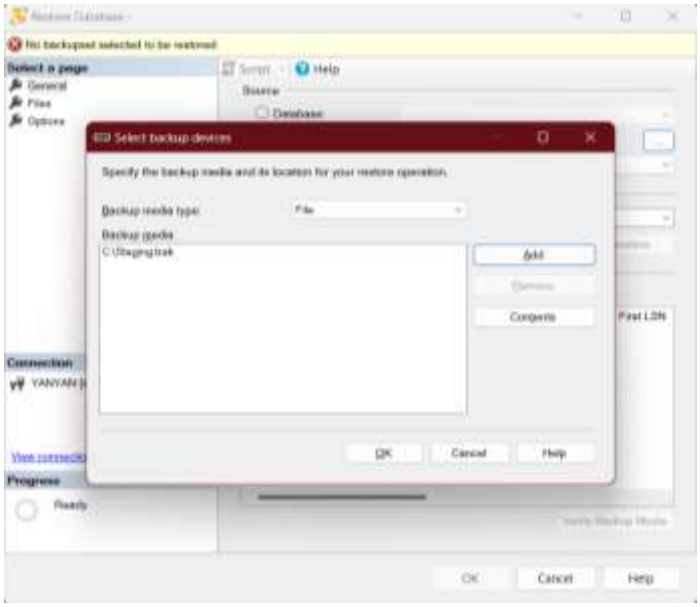
1. Perform Import/Restore of Staging Database.

When opening SSMS, the Staging database is not present in the databases list. To add the Staging database, you need to perform a database restore using the available database file provided in this challenge. To restore the database, I do it by right-clicking on the "Databases" folder and selecting "Restore Database."

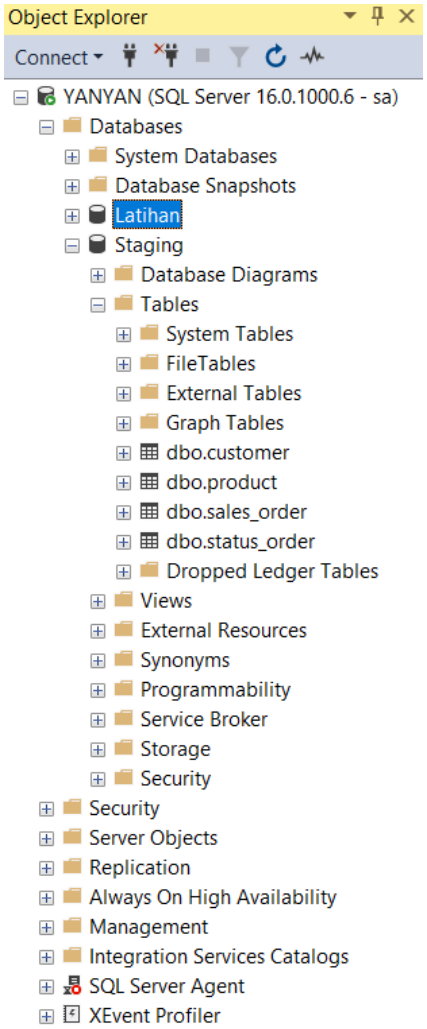


Next, a window will appear as shown below. The first step is to click on "Device" as source then click the "..." button as indicated by number 1 in the picture. This will bring up the "Select Backup Device" window. Click the "Add" button as shown in number 2 in the picture, then select "Staging.bak" and press OK.





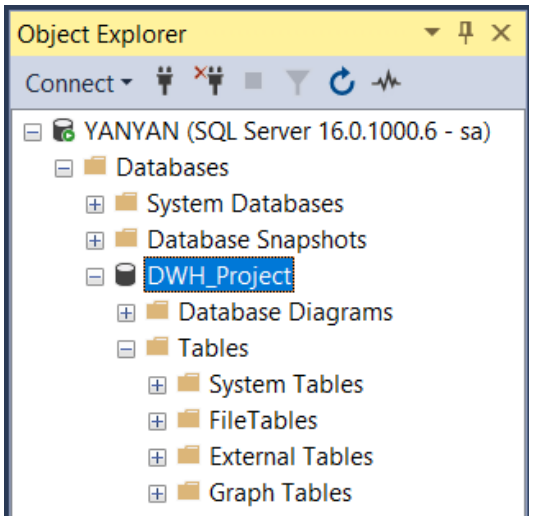
After clicking OK, the restored database with the name "Staging" will appear.



Here is the content of the Staging database. This database contains 4 tables: customer, product, sales_order, and status_order.

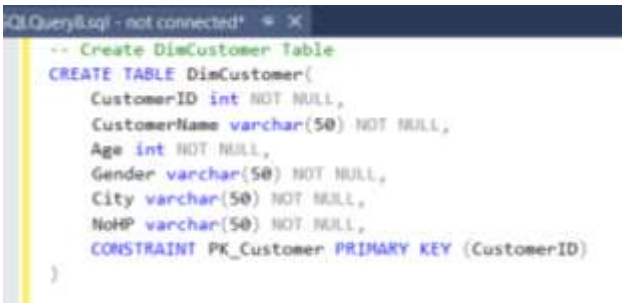
2. Create a Database named DWH_Project, and create Fact and Dimension Tables from the tables in the Staging Database.

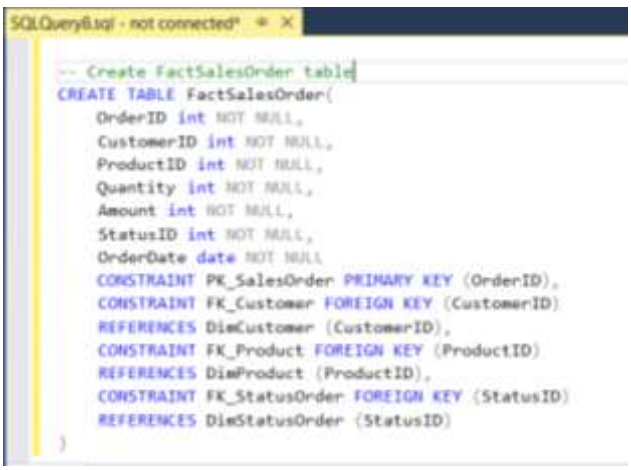
The first step is to create a query to create a database named "DWH_Project." The DWH_Project database has been created, as seen in the picture below. However, the database does not have any tables yet. The next step is to create tables for this DWH_Project database.

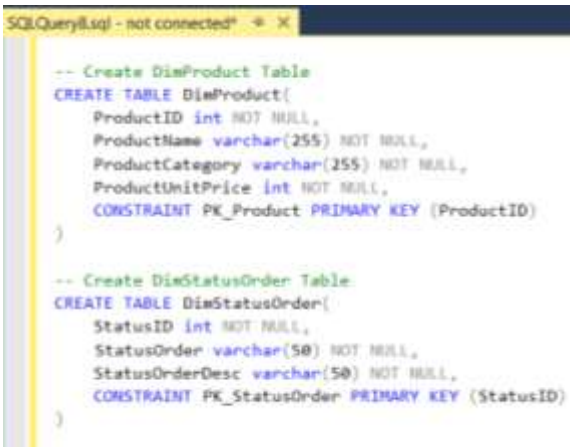


In this section, I created 4 tables: DimCustomer, DimProduct, DimStatusOrder, and FactSalesOrder.

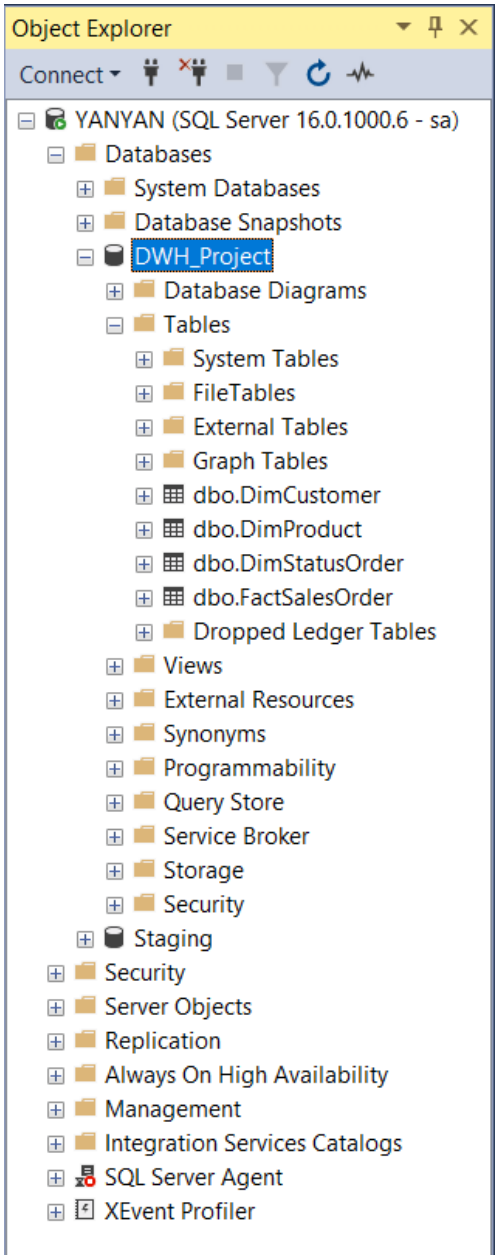
1. DimCustomer table has 6 attributes: 2 integer and 4 varchar with a max length of 50 characters. All attributes must have non-null values, and CustomerID is set as the primary key.
2. DimStatusOrder table has 4 attributes: 2 integers and 2 varchars with a max length of 50 characters. All attributes must have non-null values, and StatusID is set as the primary key.
3. DimProduct table has 4 attributes: 2 integers and 2 varchars with a max length of 255 characters. All attributes must have non-null values, and ProductID is set as the primary key.
4. FactSalesOrder table has 7 attributes: 6 integers and 1 date. All attributes must have non-null values, and OrderID is set as the primary key. CustomerID, ProductID, and StatusID are set as foreign keys.





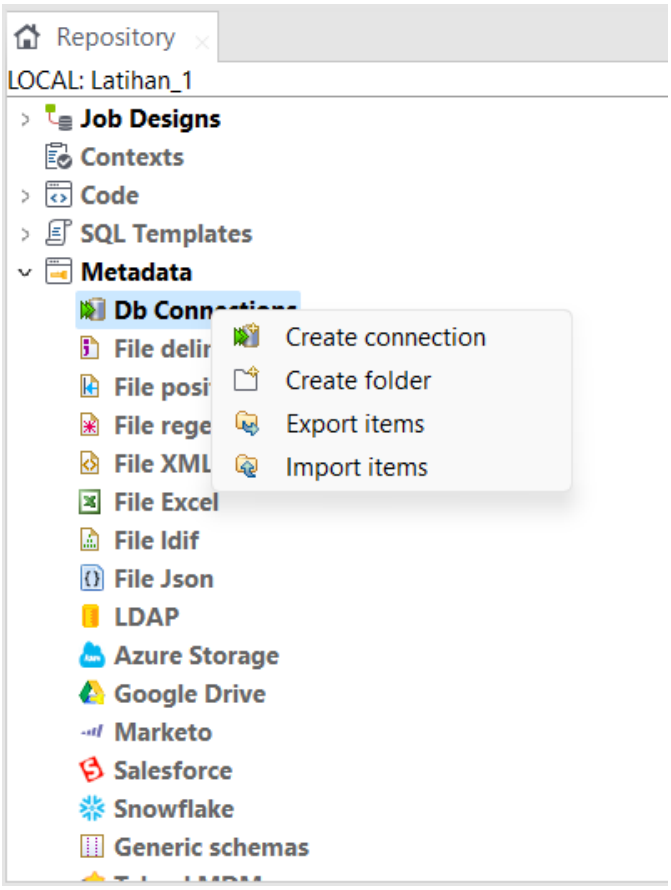


After done all this step we can see the image below, it is evident that the tables have been successfully created. The tables include DimCustomer, DimProduct, DimStatusOrder, and FactSalesOrder, with their respective attributes and keys properly defined. The database setup is now complete and ready for data insertion.

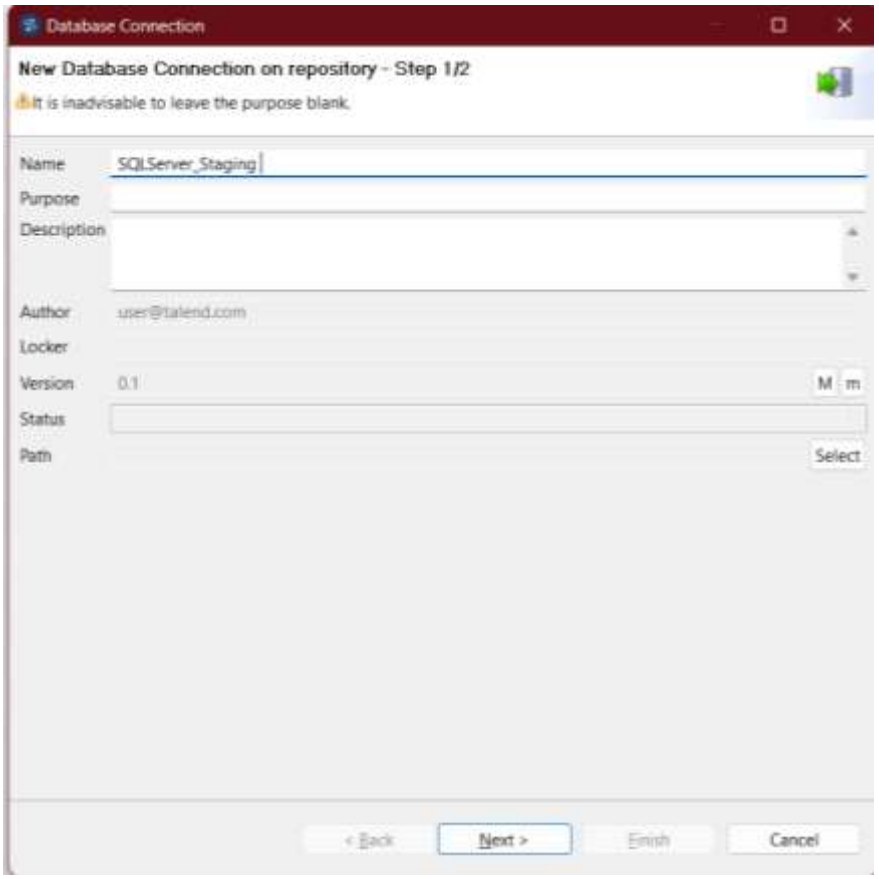


3. Build an ETL Job in Talend application to transfer data from Staging to Data Warehouse. Specifically, for the DimCustomer Table, perform data transformation by converting data in FirstName and LastName columns to all capital letters, then combine these two columns into a single column named CustomerName.

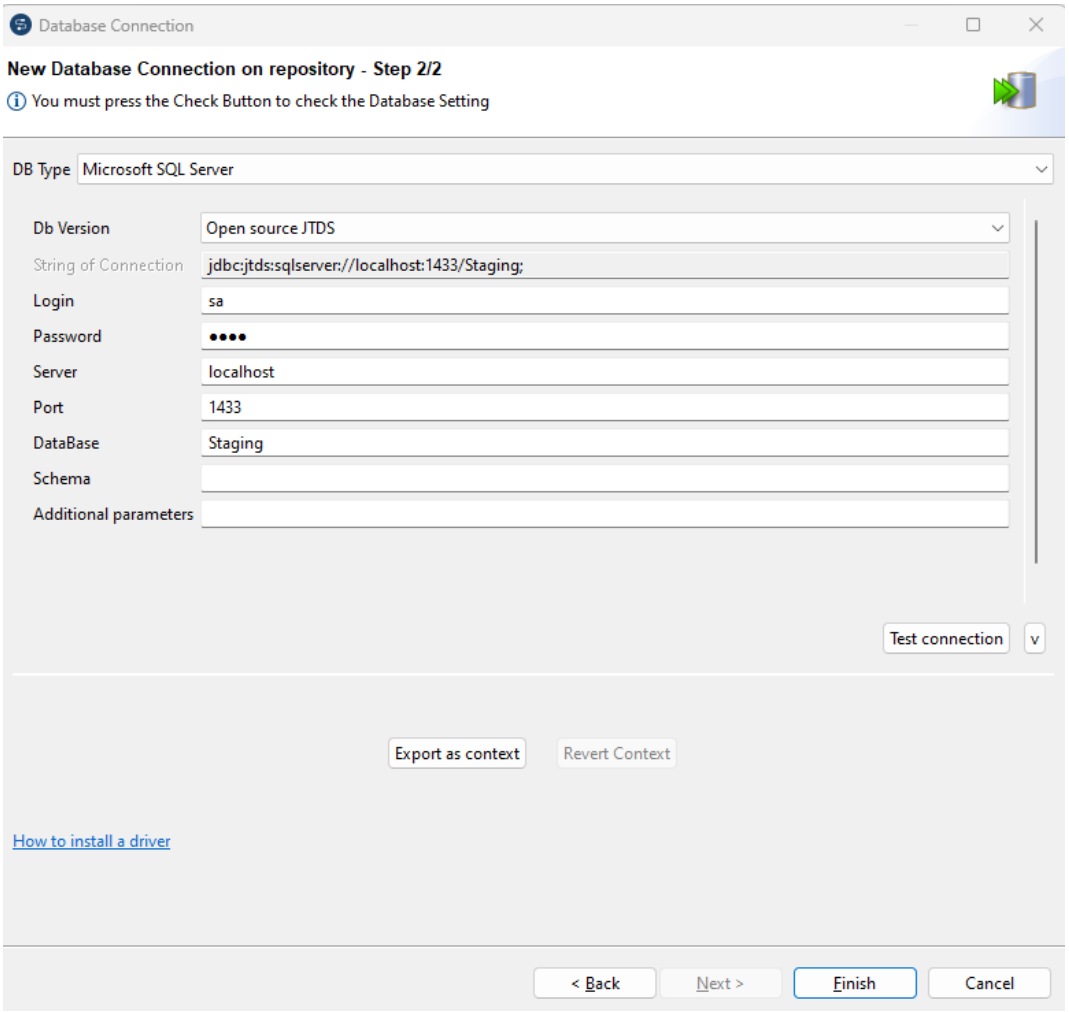
The first step is to create a database connection to connect Talend with SQL Server. To do this, click on "Metadata," then right-click on "DB Connections" and select "Create Connection."



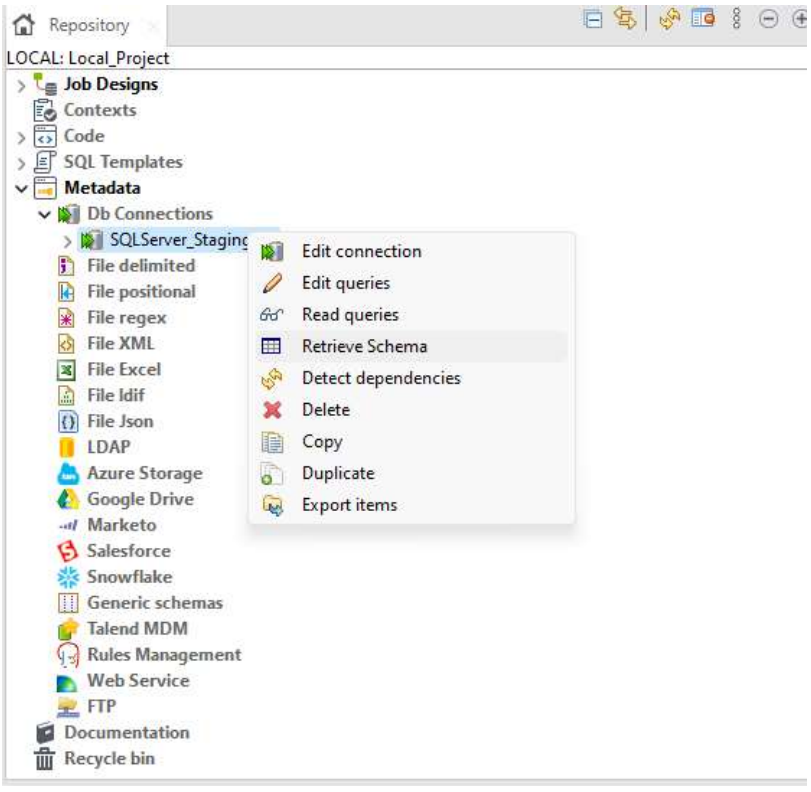
Then, a window will appear as shown below. Here, I give the connection a name, "SQLServer_Staging," and then click "Next."



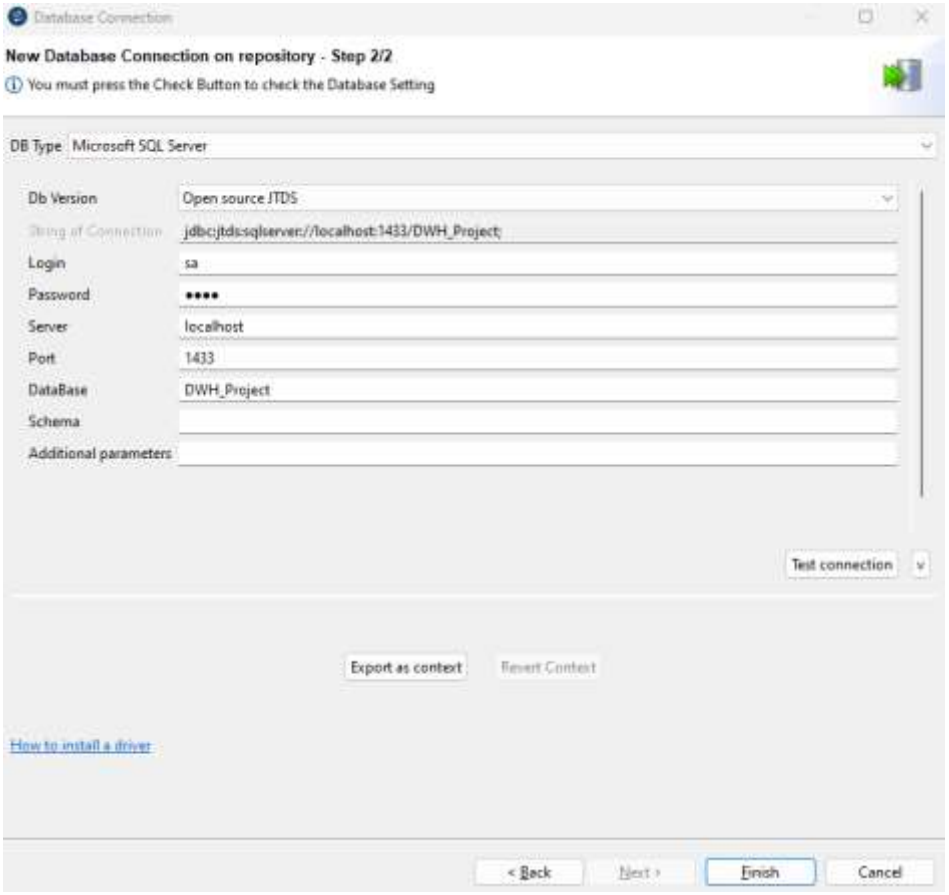
Then, fill in the details in the image below. I select DB type as Microsoft SQL Server, DB Version as Open source



JTDS, and choose the Staging database because it will be used as the source. Next, perform the "Retrieve Schema" action on the SQLServer_Staging database.

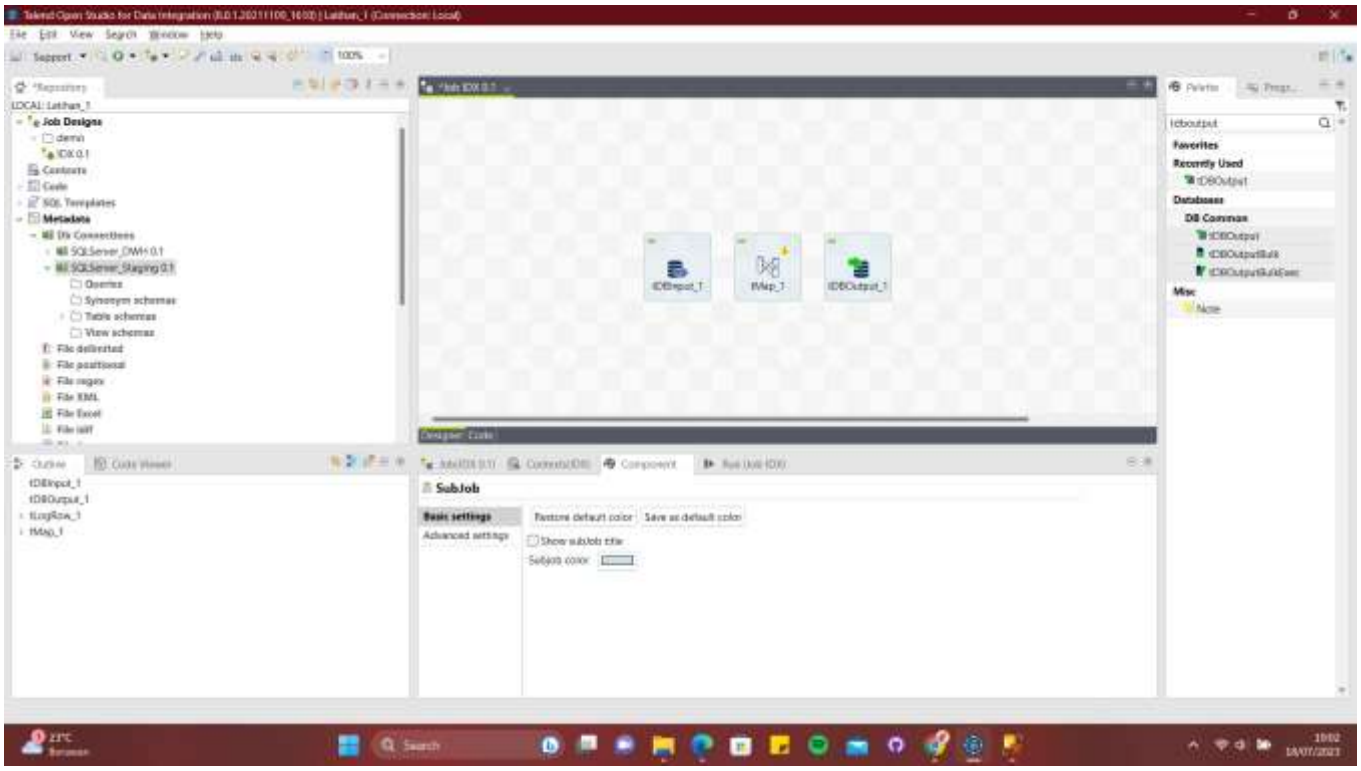


Then, I do the same process as creating a connection for SQLServer_Staging, but this time, I create a connection for the DWH_Project database. Here, I name the connection "SQLServer_DWH," and the database I input is "DWH_Project."

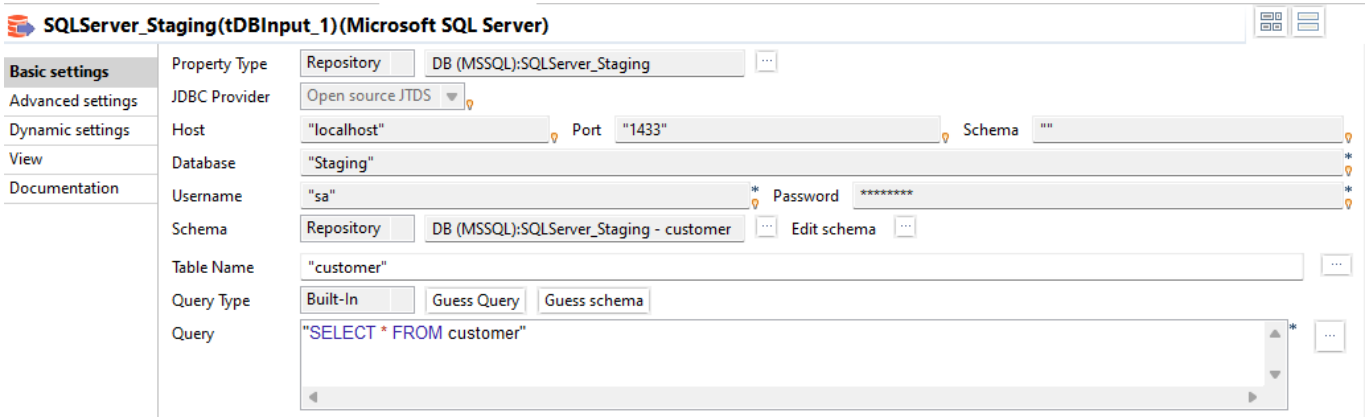


Afterwards, I add the tDBInput component for SQLServer_Staging as it serves as the source database. Then, I insert the tMap and tDBOutput components.

You can find the tDBInput, tMap, and tDBOutput components in the Palette section of the Talend Studio. To locate them, you can either use the search bar in the Palette and enter their names (tDBInput, tMap, tDBOutput), or you can directly drag and drop the data you want to use as tDBInput and the data you want to use as tDBOutput.

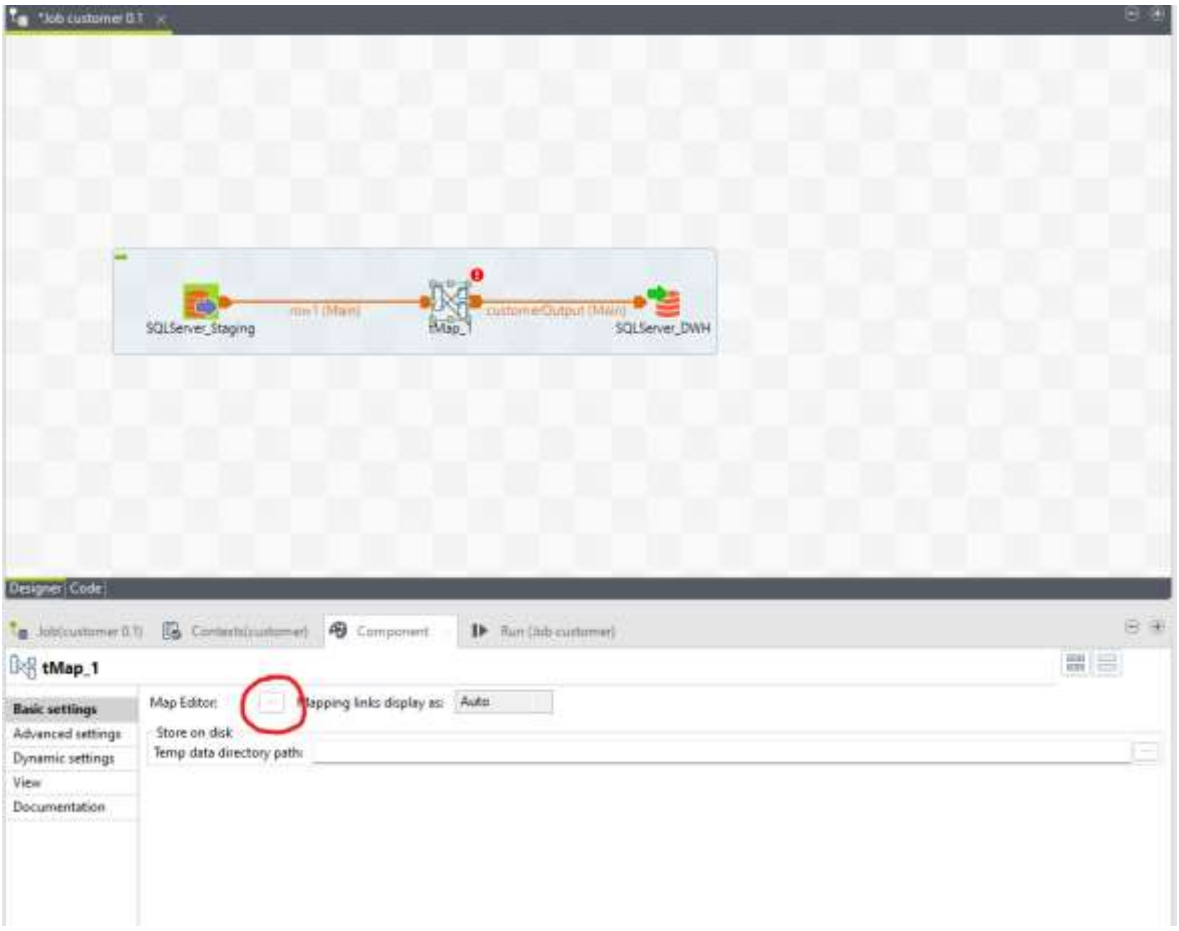


Next, on the tDBInput component, I select the schema repository and retrieve the "customer" table. Then, I input a query to fetch all attributes from the "customer" table.



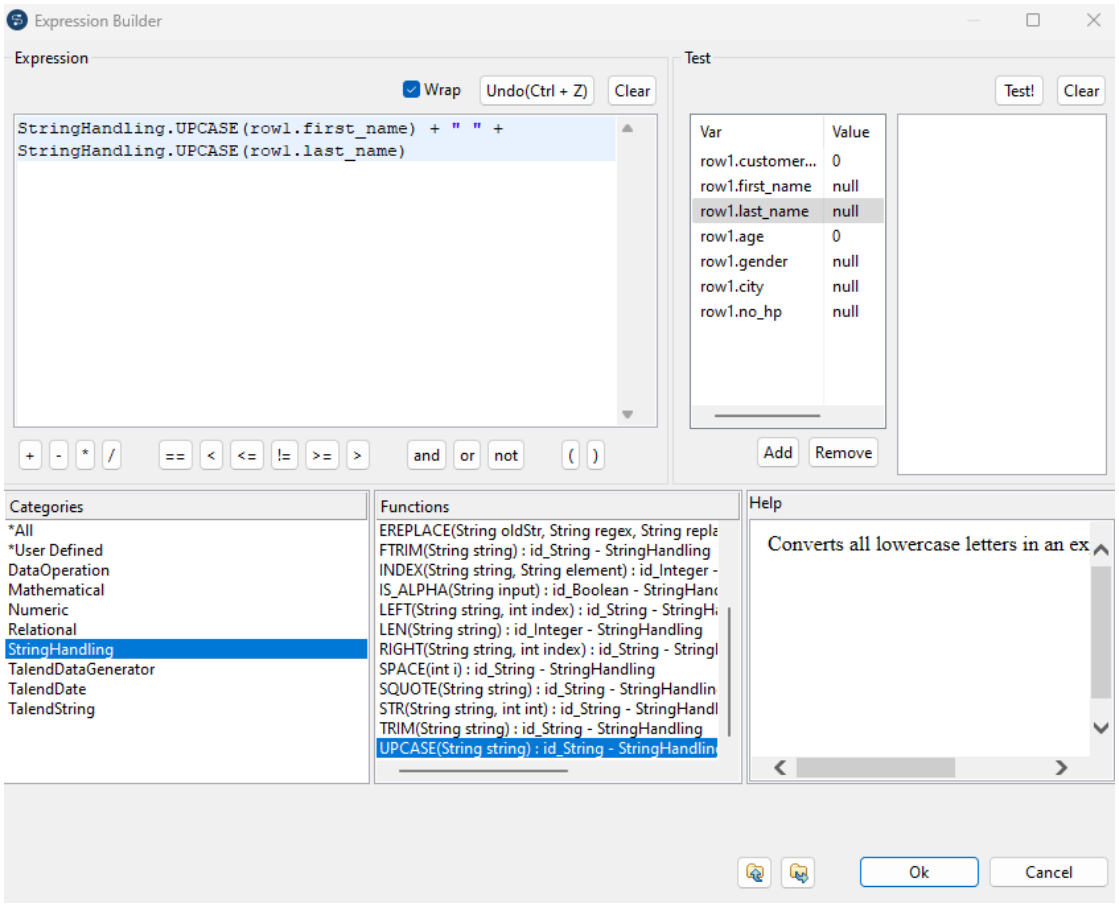
Next, on the tMap component, click the "..." button as shown in the picture below.

(This picture showing, I already input for the output so If I don't include the output, the line between tMap and SqlServer_DWH still need to make, the step just right clicks on tMap, raw and give name for newoutput)

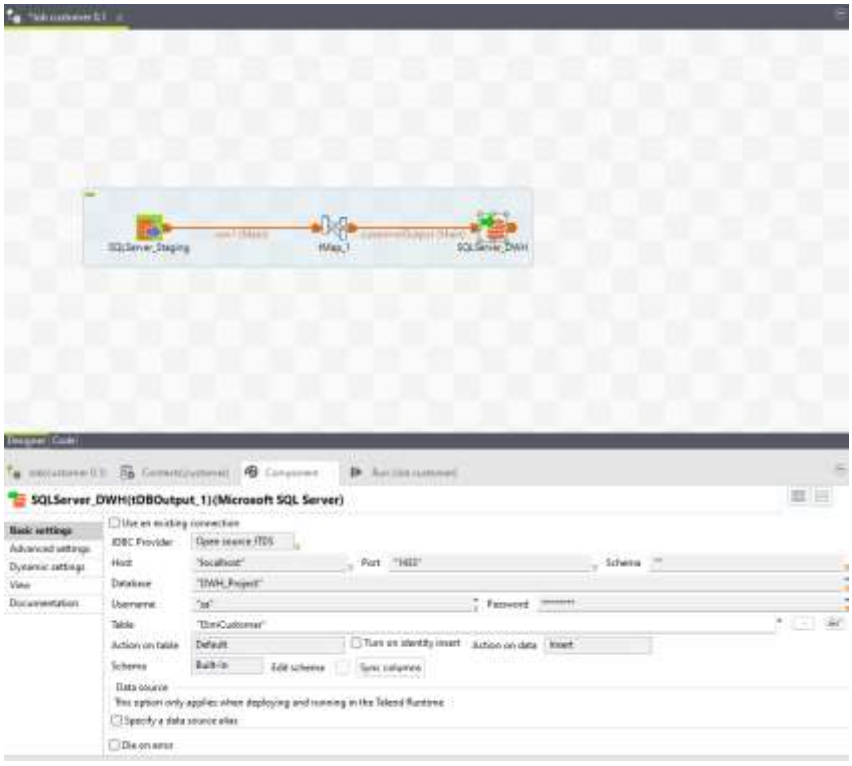


After that, connect the attributes from the "customer" table in the Staging database to the output for insertion into the "DimCustomer" table in the DWH_Project database. Then, perform editing on the Expression Builder as shown in the red-circled area in the picture below.

In the StringHandling section, I select UPCASE for the attributes "first_name" and "last_name" to convert the data to all uppercase letters and add a space. After that, click OK.



Then, on the output component, select the table "DimCustomer."



After that, I perform the same steps, but only with different tables, such as "product," "sales_order," and "order_status." Then, I run all the jobs that have been created.

NOTE: The changes in attributes are only applied to the "DimCustomer" column because we are combining the "first_name" and "last_name" columns from the Staging database into the "CustomerName" column in the DWH_Project database.

4. Develop a Stored Procedure (SP) to display a summary of sales orders based on the delivery status.

```
CREATE 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
```

In this stored procedure, we utilize several JOINs between the fact table and dimension tables (DimCustomer, DimProduct, DimStatusOrder) to retrieve the required columns (OrderID, CustomerName, ProductName, Quantity, StatusOrder). Additionally, there is a parameter @StatusID that acts as a filter for the query, where only data with a StatusID matching the parameter's value will be displayed.

Rseult :

```
EXEC summary_order_status @StatusID = 1;
```

Results | Messages

	OrderID	CustomerName	ProductName	Quantity	StatusOrder
1	1303	BUDI SANTOSO	Macbook Air 2020 13 inch	1	Awaiting Payment
2	1310	LIA RAHMAWATI	Kipas Angin Cosmos	2	Awaiting Payment

```
EXEC summary_order_status @StatusID = 2;
```

Results | Messages

	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

```
EXEC summary_order_status @StatusID = 3;
```

Results Messages

	OrderID	CustomerName	ProductName	Quantity	StatusOrder
1	1305	BAGUS PRAKOSO	Blender Philips 500 watt	3	Shipped
2	1308	BELA ADRILIA	Luciana Set Dress 2 in 1	2	Shipped

```
EXEC summary_order_status @StatusID = 4;
```

Results Messages

	OrderID	CustomerName	ProductName	Quantity	StatusOrder
1	1302	RIFKI MUHAMMAD	HP Elitebook 840 G4	1	Completed
2	1306	RIFKI MUHAMMAD	Asus Zenbook 800	1	Completed
3	1309	AJENG SRIASIH	Bagpack Navy Club	1	Completed

```
EXEC summary_order_status @StatusID = 5;
```

Results Messages

OrderID	CustomerName	ProductName	Quantity	StatusOrder

Note:

StatusID	Keterangan
1	Awaiting Payment
2	Awaiting Shipment
3	Shipped
4	Completed
5	Cancelled

In this database, there are no orders with a StatusID of 5, so no values will appear when StatusID 5 is called.