BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE,
PILANI DATA WAREHOUSING SS-G515
LAB-2[Multi-dimensional Databases]

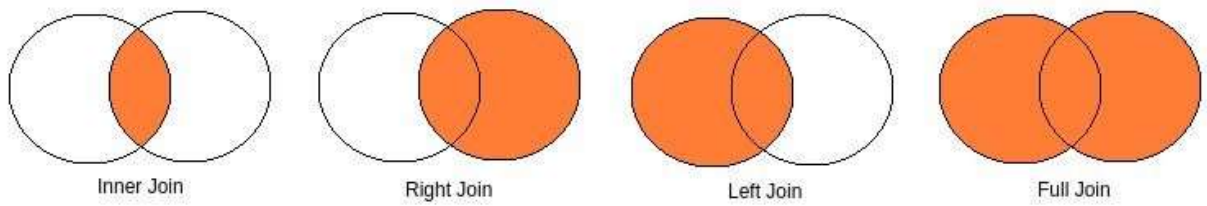
DATE: 05/04/2021 **TIME: 02 Hours**

1. **JOINS**: The join clause in SQL is used to combine records from two or more tables based on a related column between them. It is generally used when we want to retrieve data from tables that have some kind of relationship (one-to-many, many-to-many etc.) between them.

   Syntax:

   SELECT table_name1.column, table_name2.column
   FROM table_name1 JOIN table_name2
   ON (table_name1.column = table_name2.name)
   WHERE .....

   There are four types of JOINS namely, inner, full, left and right join. These can be represented as shown in the figure given below.



a. **Inner Join:** The inner join returns only those records that have matching values in both the tables.

   Example 1. Retrieve the Order ID, first and last name of the associated employee and the name of the company that placed the order after the year 1998.

```
SELECT o.OrderID, c.CompanyName, e.FirstName, e.LastName
FROM Orders o
JOIN Employees e ON (e.EmployeeID = o.EmployeeID)
JOIN Customers c ON (c.CustomerID = o.CustomerID)
WHERE o.OrderDate > '1-Jan-1998'
ORDER BY c.CompanyName;
```

Example 2. Retrieve the total quantity of products (from the Order_Details table) ordered. Only show records for products for which the quantity ordered is fewer than 200.

```
SELECT p.ProductName, SUM(od.Quantity) AS TotalUnits
FROM [Order Details] od
JOIN Products p ON (p.ProductID = od.ProductID)
GROUP BY p.ProductName
HAVING SUM(od.Quantity) < 200;
```

Example 3. Retrieve company name, order id, and total price of all products of which Northwind has sold more than $10,000 worth.

```
SELECT c.CompanyName, o.OrderID, od.UnitPrice * od.Quantity * (1-od.Discount)
AS TotalPrice
FROM Order_Details od
JOIN Orders o ON (o.OrderID = od.OrderID)
JOIN Customers c ON (c.CustomerID = o.CustomerID)
WHERE od.UnitPrice * od.Quantity * (1-od.Discount) > 10000 ORDER
BY TotalPrice DESC;
```

Example 4. Retrieve the total number of orders by Customer since December 31, 1996. The report should only return rows for which the number of orders is greater than 15.

```
SELECT c.CompanyName, COUNT(o.OrderID) AS NumOrders
FROM Customers c
JOIN Orders o ON (c.CustomerID = o.CustomerID)
WHERE OrderDate > '1996-12-31'
GROUP BY c.CompanyName
HAVING COUNT(o.OrderID) > 15
ORDER BY NumOrders DESC;
```

Example 5. Retrieve Product sales for the year 1997.

```
SELECT     dbo.Categories.CategoryName,dbo.Products.ProductName,
SUM(CONVERT(money, (dbo.[Order Details].UnitPrice * dbo.[Order
Details].Quantity) * (1 - dbo.[Order Details].Discount) / 100) * 100) AS
ProductSales
FROM     dbo.Categories
INNER JOIN    dbo.Products
```

```
ON dbo.Categories.CategoryID = dbo.Products.CategoryID
INNER JOIN dbo.Orders
INNER JOIN dbo.[Order Details]
ON dbo.Orders.OrderID = dbo.[Order Details].OrderID
ON dbo.Products.ProductID = dbo.[Order Details].ProductID
WHERE dbo.Orders.ShippedDate BETWEEN '19970101' AND '19971231'
GROUP BY dbo.Categories.CategoryName,dbo.Products.ProductName
```

b. **Outer Joins:**

i. **Left Join**: A LEFT JOIN (also called a LEFT OUTER JOIN) returns all the records from the first table even if there are no matches in the second table. Syntax:

SELECT table1.column, table2.column
FROM table1
LEFT [OUTER] JOIN table2 ON (table1.column=table2.column) WHERE conditions

Example: Retrieve the number of employees and customers from each city that has employees in it.

```
SELECT COUNT(DISTINCT e.EmployeeID) AS numEmployees,
COUNT(DISTINCT c.CustomerID) AS numCompanies,e.City, c.City
FROM Employees e
LEFT JOIN Customers c ON (e.City = c.City)
GROUP BY e.City, c.City
ORDER BY numEmployees DESC;
```

ii. **Right Join**: A RIGHT JOIN (also called a RIGHT OUTER JOIN) returns all the records from the second table even if there are no matches in the first table. Syntax:

SELECT table1.column, table2.column
FROM table1
RIGHT [OUTER] JOIN table2 ON (table1.column=table2.column) WHERE conditions

Example: Retrieve the number of employees and customers from each city that has customers in it.

```
SELECT COUNT(DISTINCT e.EmployeeID) AS numEmployees,
COUNT(DISTINCT c.CustomerID) AS numCompanies, e.City, c.City
FROM Employees e
RIGHT JOIN Customers c ON (e.City = c.City)
GROUP BY e.City, c.City
ORDER BY numEmployees DESC;
```

iii. **Full Outer Join**: A FULL JOIN (also called a FULL OUTER JOIN) returns all the records from each table even if there are no matches in the joined table. Syntax:

SELECT table1.column, table2.column
FROM table1
FULL [OUTER] JOIN table2 ON (table1.column=table2.column)
WHERE conditions

Example: Retrieve the number of employees and customers from each city.

```
SELECT COUNT(DISTINCT e.EmployeeID) AS numEmployees,
COUNT(DISTINCT c.CustomerID) AS numCompanies, e.City, c.City
FROM Employees e
FULL JOIN Customers c ON (e.City = c.City)
GROUP BY e.City, c.City
ORDER BY numEmployees DESC;
```

2. **Unions**: Unions are used to retrieve records from multiple tables or to get multiple record sets from a single table.

Example 1: Retrieve the phone numbers of all shippers, customers, and suppliers

```
SELECT CompanyName, Phone
FROM Shippers
        UNION
SELECT CompanyName, Phone
FROM Customers
        UNION
SELECT CompanyName, Phone
FROM Suppliers
ORDER BY CompanyName;
```

Union All: By default, all duplicates are removed in UNIONs. To include duplicates, use UNION ALL instead of UNION.

Example 2. Retrieve contact name and phone numbers for all employees, customers, and suppliers.

```
SELECT FirstName + ' ' + LastName AS Contact, HomePhone As Phone
FROM Employees
        UNION
SELECT ContactName, Phone
FROM Customers
        UNION
SELECT ContactName, Phone
FROM Suppliers
ORDER BY Contact;
```

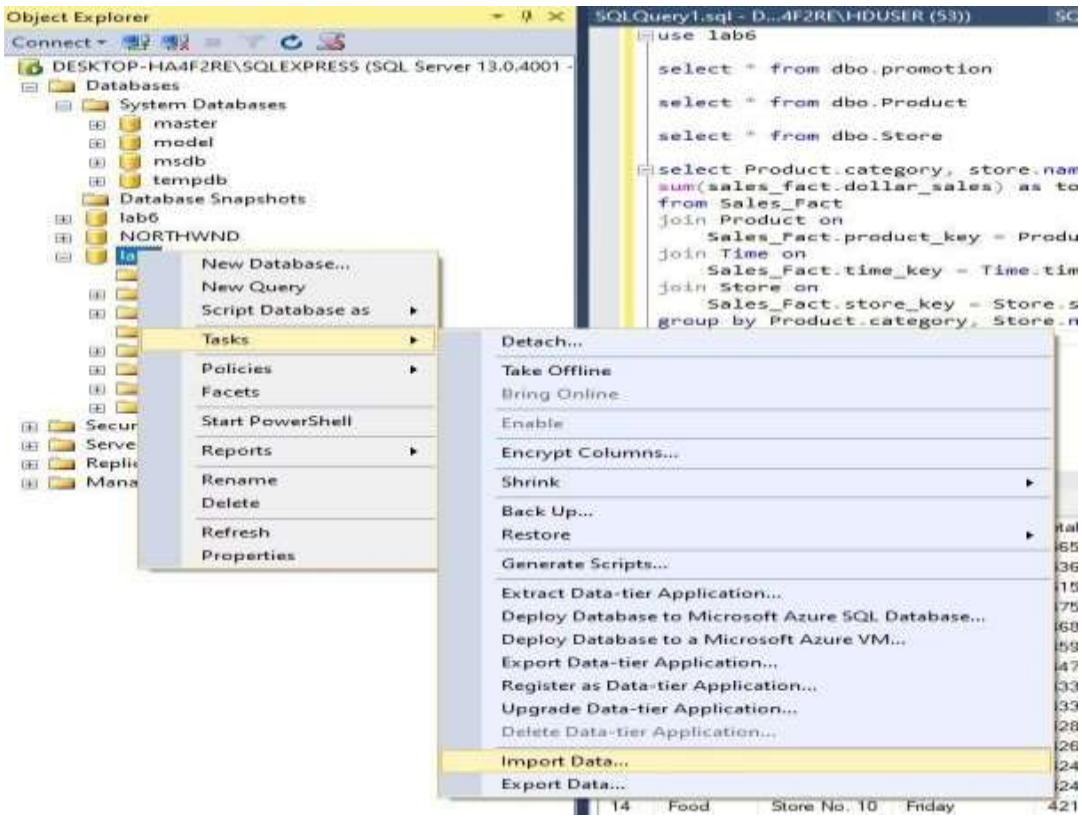
****

Exercise: Consider the schema of a grocery store which has following tables:-

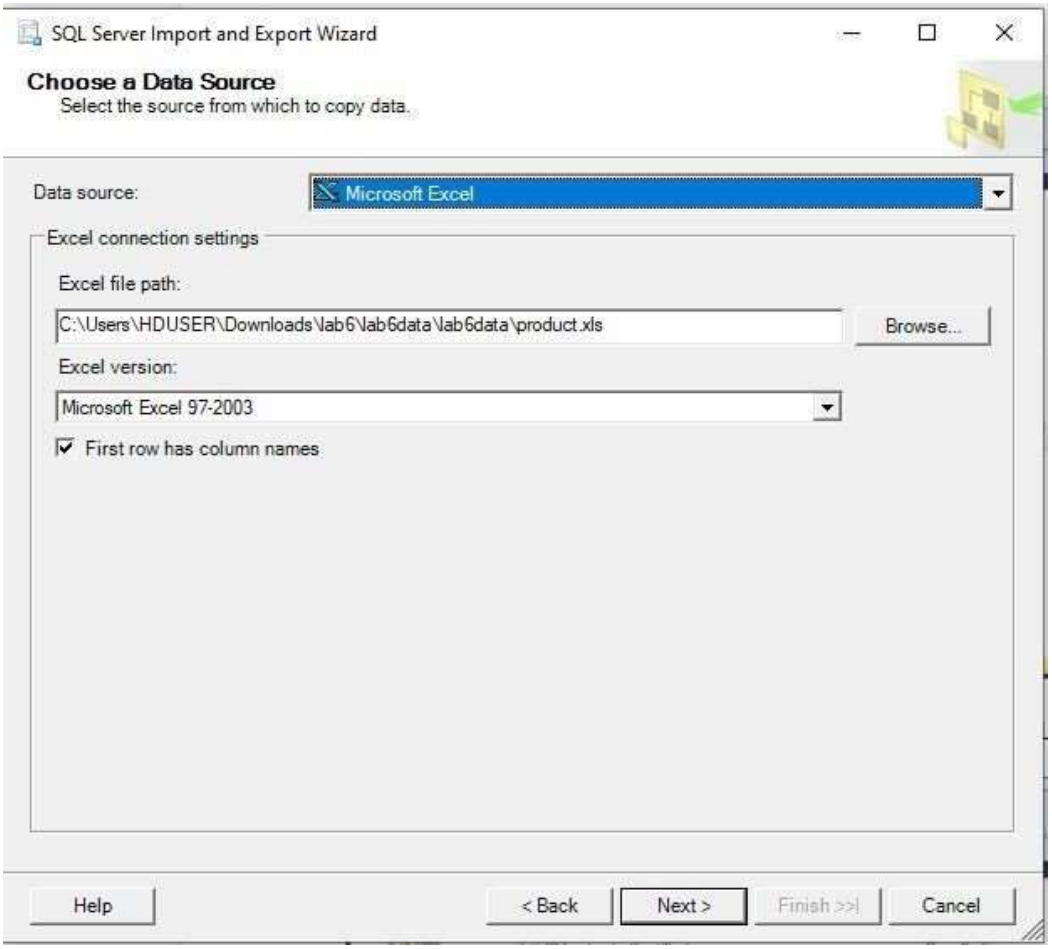
- Product
- Promotion coverage
- Promotion
- Sales
- Store
- Time

Import the tables given in excel sheet to Microsoft SQL Server Management Studio. To do so, follow the following steps:

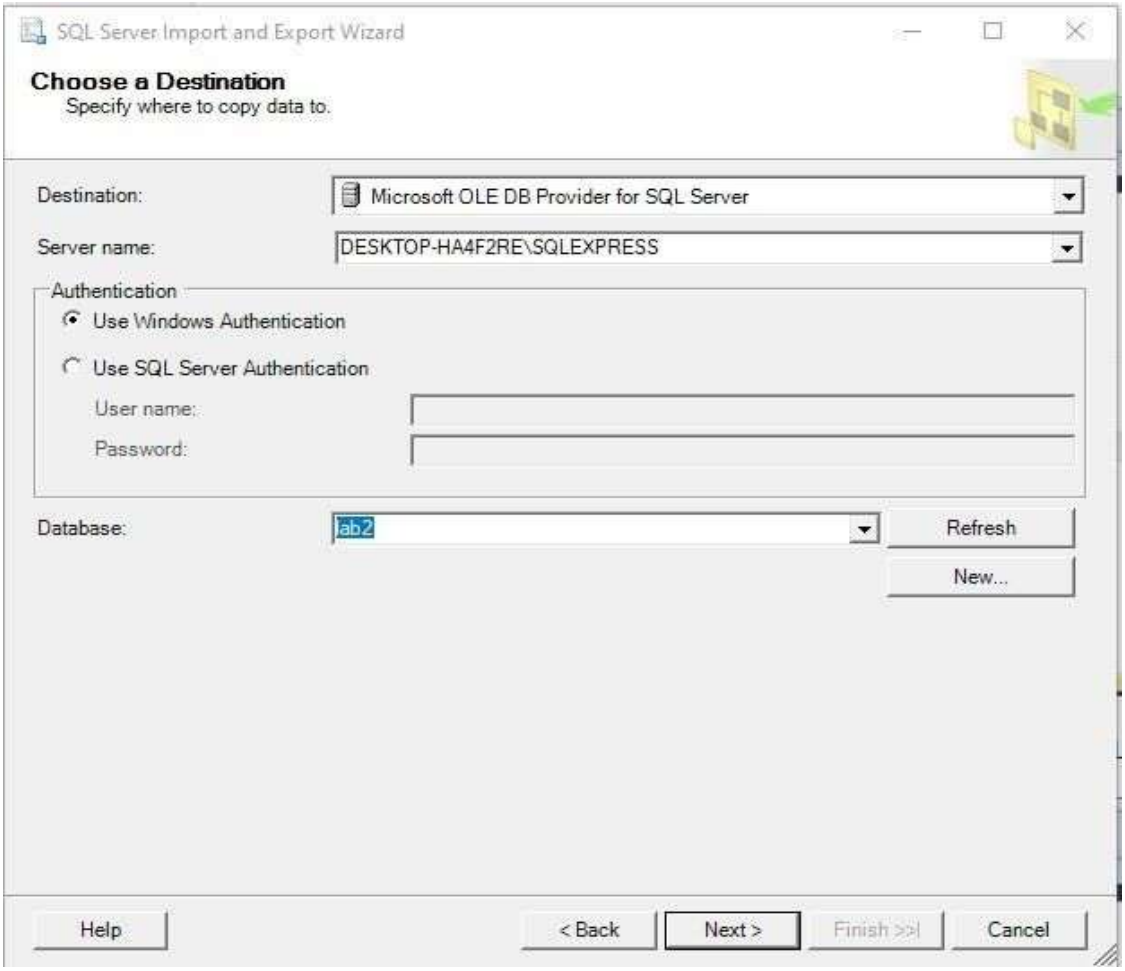
Step 1. Create a new database ‘Lab2’. Right click on the database created and select ‘Tasks’ and then ’Import Data’ as shown in figure given below. A wizard will open.



Step 2. Select Microsoft Excel as the data source since all the files are in .xls format. Give the path to the source folder containing the files and click on Next.

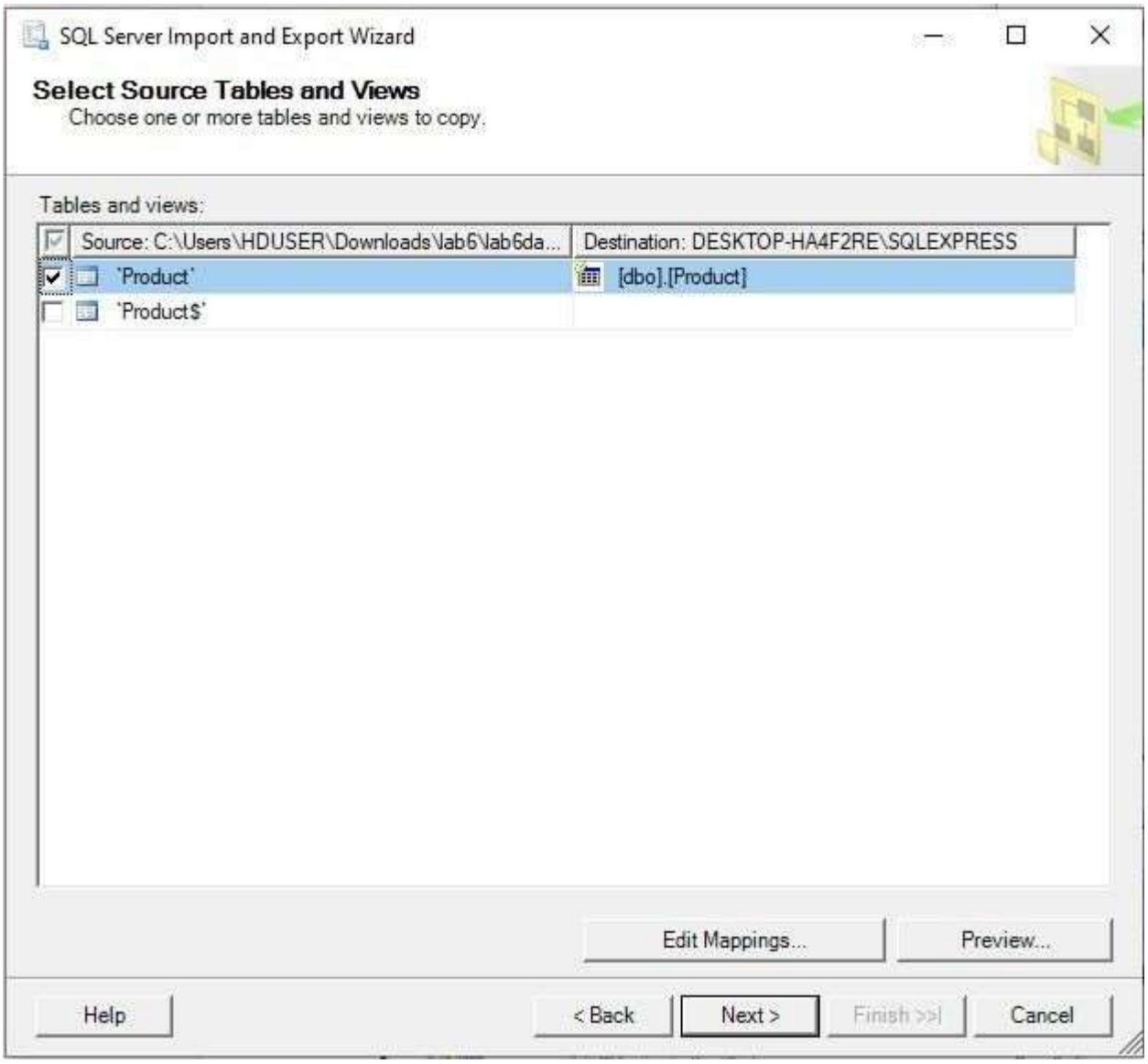


Step 3. Select destination as ‘Microsoft OLE DB Provider for SQL Server, and click on Next. Click on ‘Copy data from one or more tables or views’.

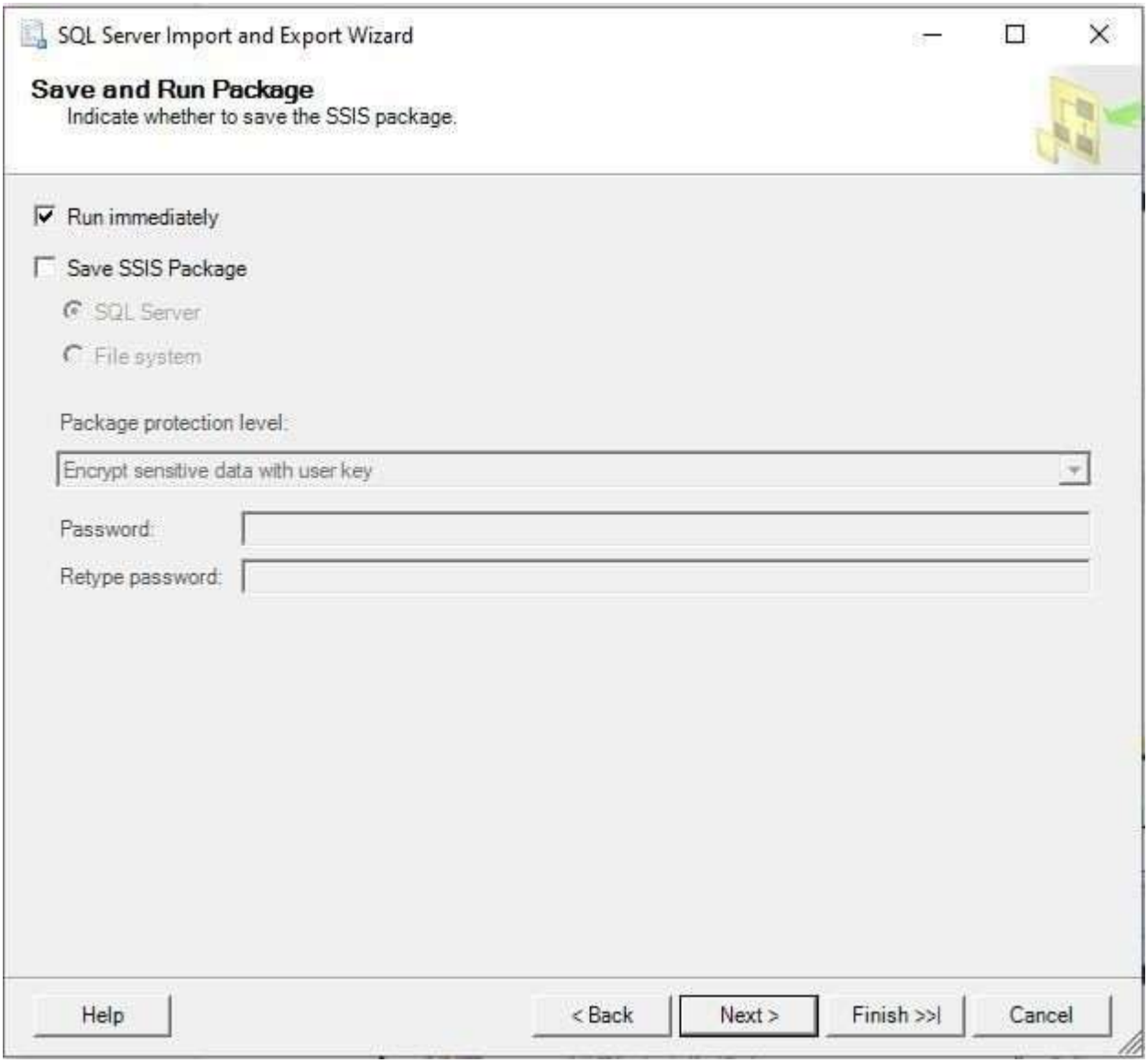


Step 4. Here, you can see the default names for source and destination. If you want to create a new table with default name then no need to change anything and click next.

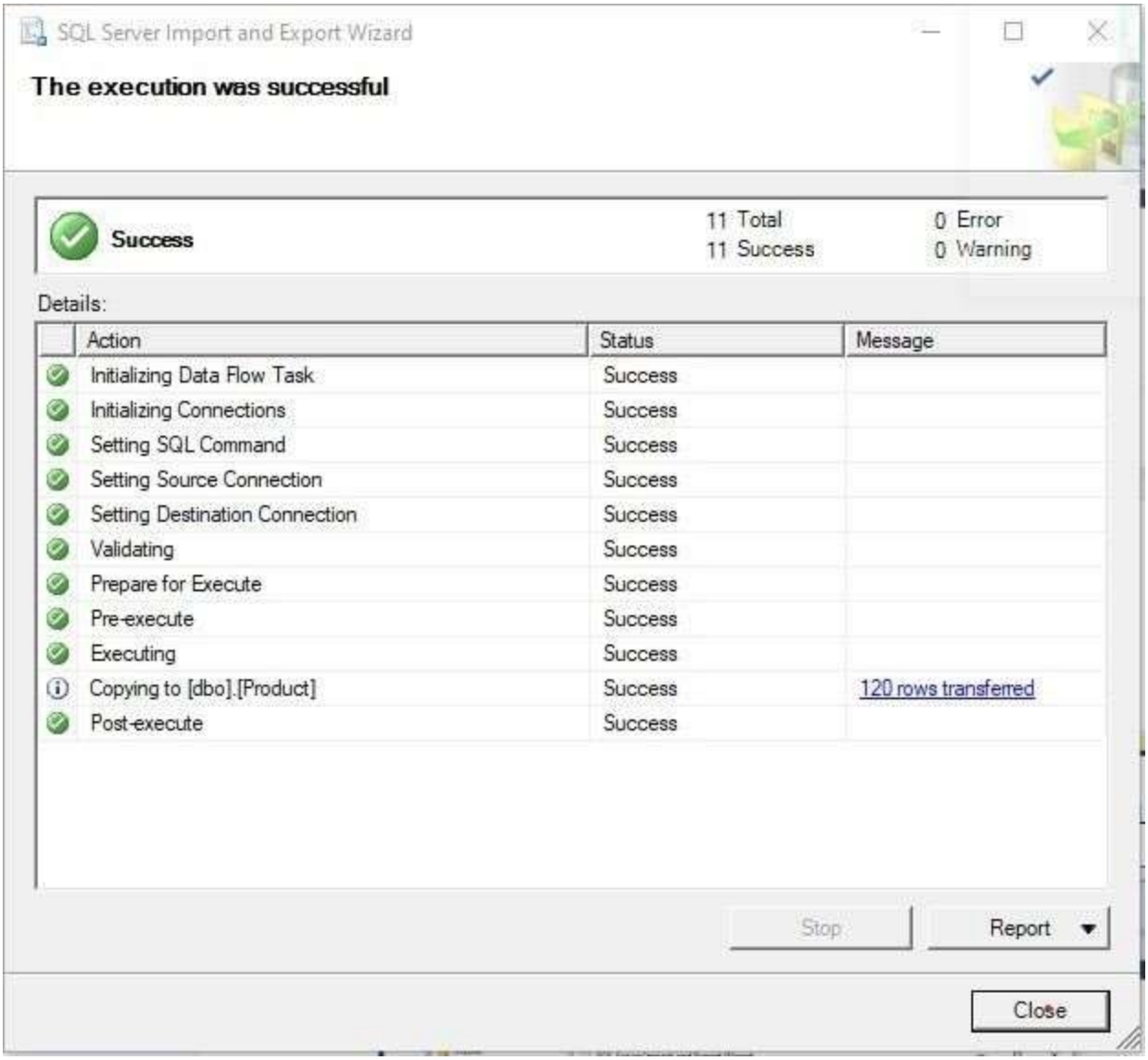
Note: If you want to change mapping, then click on Edit mappings like drop and recreate destination table or enable identity insert etc.



Step 5. Click on Run Immediately and Finish.



Step 6. Final wizard will show the information regarding the number of rows transferred and the success status.



Now that you have imported the data, you can check the same by writing a select query on the same table. Repeat the same steps for the remaining tables as well. Write queries on this database to retrieve the following:

1. Sales total w.r.t. Categories by store by day
2. Sales Total by store by day
3. Sales Total of district by product by day
4. Sales total for a month by product by store
5. Sales total for a year by product by store
6. Sales Total by year by All stores by product
7. Sales Total of category by month by district
8. Sales Total of category by store by year
9. Sales Total of category by store by month
10. Calculate average selling price for a given period of time.