LONDON METROPOLITAN UNIVERSITY

Islington college
(इस्लिङ्टन कलेज)

CS7079NI Data Warehousing and Big Data

60% Individual Coursework

2021 Autumn

Student Name: Pravash Karki

London Met ID: 11071480

College ID: NP01MS7S210070

Assignment Due Date: 21st Jan 2022

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a marks of zero will be awarded.

Table of Contents

LIST OF FIGURES ..... **III**

LIST OF TABLES ..... **VI**

LIST OF ABBREVIATION ..... **VII**

INTRODUCTION **1**

1. DATA WAREHOUSE & WAREHOUSING **2**

1.1 ETL PROCESS 2

1.1.1 Creating Data Warehouse. *3*

1.1.2 Data Flow Task: Using ETL Process to load Source Data *4*

1.1.3 Configure OLE DB Source and OLE DB Destination. *13*

1.1.4 Execution of the ETL Process *33*

1.2 DIMENSIONAL DATA MODELLING (DM) 35

1.2.1 Fact Table *35*

1.2.2 Dimension Table *36*

2. ANALYSIS & DESIGN OF DIMENSIONAL DATA MODEL **38**

2.1 GRAINS 38

2.1.1 Grain of fact_sales Table Detail *38*

2.2 DIMENSIONS OF THE CENTRAL FACT TABLE 39

2.2.1 Attribute List, Query and Value Results of Fact Table *39*

2.2.2 Attribute List, Query and Value Results of Dimensions of Fact Table *43*

2.2.2.1 dimension_categories Table 43

2.2.2.2 dimension_date Table 44

2.2.2.3 dimension_departments Table 48

2.2.2.4 dimension_order_items Table 49

2.2.2.5 dimension_orders Table 51

2.2.2.6 dimension_products Table 52

2.3 STRUCTURE OF MODEL TABLE AS FACT TABLES IN DIMENSION 53

2.3.1 Star Schema *53*

2.3.2 Business Reports *54*

2.3.2.1 Business Reports for XYZ Retail Store 54

3. APACHE (HADOOP, SQOOP AND HIVE) **63**

3.1 EXPORT WAREHOUSE DATABASE TO HIVE STAGING 63

3.2 DATA TRANSFORMATION FOR ANALYSIS & REPORTING IN HIVE 68

3.2.1 Creating xyz_dmart Database to Store Dimension and Fact Table *68*

3.3 LOADING DATA INTO XYZ_DMART AND DATA MANIPULATION DEMONSTRATION 71

3.3.1 Verifying the created table inside Hive and HDFS *72*

3.3.2 Demonstration of Data Manipulation on Loaded Data *73*

3.4 FILE FORMATS DEMONSTRATION AND OTHER OPTIMISATION TECHNIQUES IN SQOOP AND HIVE. 75

3.4.1 Use of ORC File Format *75*

3.4.2 Use of Single Mapper in Sqoop *75*

3.4.3 Use of Hive Partitioning *75*

3.4.4 Using native tool to extract data from database in Sqoop *75*

3.4.5 Importing tables directly in hive with the use of Sqoop *75*

4. PERSONAL REFLECTION 76

BIBLIOGRAPHY..... 77

List of Figures

Figure 1: A contemporary ETL process using a Data Warehouse 2
Figure 2: Creating xyz_dwh Data Warehouse in SQL Server..... 3
Figure 3: Creating a Data Flow Task. 4
Figure 4: Testing the DB Connection between Source and Data Warehouse Database 5
Figure 5: Creating schema of the categories table in Data Warehouse [xyz_dwh] 6
Figure 6: Creating schema of the “customers” table in Data Warehouse [xyz_dwh] 7
Figure 7: Creating schema of the “departments” table in Data Warehouse [xyz_dwh] 8
Figure 8: Creating schema of the orders_items table in Data Warehouse [xyz_dwh]..... 9
Figure 9: Creating schema of the orders table in Data Warehouse [xyz_dwh] 10
Figure 10: Creating schema of the products table in Data Warehouse [xyz_dwh]..... 11
Figure 11: Successful Creation of all tables in Data Warehouse [xyz_dwh] 12
Figure 12: Configuring Source categories table. 13
Figure 13: Configuring Destination categories table. 14
Figure 14: Configuring Destination and Source categories table column. 15
Figure 15: Configuring Source customers table. 16
Figure 16: Configuring Destination customers table. 17
Figure 17: Configuring Destination and Source customers table column. 18
Figure 18: Configuring Source departments table. 19
Figure 19: Configuring Destination departments table. 20
Figure 20: Configuring Destination and Source departments table column. 21
Figure 21: Configuring Source order_items table. 22
Figure 22: Configuring Destination order_items table. 23
Figure 23: Configuring Destination and Source order_items table column. 24
Figure 24: Configuring Source orders table. 25
Figure 25: Configuring Destination orders table. 26
Figure 26: Configuring Destination and Source orders table column. 27
Figure 27: Configuring Source products table..... 28
Figure 28: Configuring Destination products table. 29
Figure 29: Configuring Destination and Source products table column..... 30
Figure 30: Final Data Flow Diagram..... 31
Figure 31: Setting up the Truncate SQL query in Execute SQL Task..... 32
Figure 32: Renaming the Data Flow and Execute SQL query..... 32
Figure 33: Execution of Control Flow 33
Figure 34: Successfully transferred the data of all the rows and columns..... 34
Figure 35: SQL query to create a fact table. 40
Figure 36: SQL query to insert values in the fact table..... 41
Figure 37: Top 15 Values of the fact table 42
Figure 38: SQL Query to Create Table and Insert data on dimension_categories Table..... 43
Figure 39: Top 17 Values of dimension_categories Table 43
Figure 40: SQL Query to Create Table and Insert data on dimension_date Table..... 46
Figure 41: Top 10 Values of dimension_date Table 47

Figure 42: SQL Query to Create Table and Insert Data on dimesion_departments Table 48
Figure 43: All Values of dimension_ departments Table 48
Figure 44: SQL Query to Create Table and Insert Data on dimension_order_items Table 49
Figure 45: Top 20 Value of the dimension_order_items Table 50
Figure 46: SQL Query to Create Table and Insert Data on dimension_orders table 51
Figure 47: Top 15 Values of dimension_orders Table 51
Figure 48: SQL Query to Create Table and Insert Data on dimension_products Table 52
Figure 49: Top 15 Values of dimension_products Table 52
Figure 50: Star Schema modelling to define fact and dimension tables 53
Figure 51: Monthly Query 54
Figure 52: Monthly Result 55
Figure 53: Weekly Query 56
Figure 54: Weekly Query Result 56
Figure 55: Profit Analysis Query based on Departments 57
Figure 56: Profit Result based on Departments 57
Figure 57: Query to analyse the top revenue-generating product. 58
Figure 58: List of top five revenue-generating items. 58
Figure 59: Query to find out the Canceled Orders 59
Figure 60: List of Canceled Orders 59
Figure 61: Query to find out the top-selling item. 60
Figure 62: List of top-selling items in the order. 60
Figure 63: Query to find the popular month of the top sold item. 61
Figure 64: Month Result 61
Figure 65: Query Daily Analysis 62
Figure 66: Daily Analysis Result 62
Figure 67: Using “staging_database.hql” script to create Hive Staging Area. 63
Figure 68: Commands in Script. 64
Figure 69: retail_db Database inside MySQL 64
Figure 70: Importing all tables into Hive Database 65
Figure 71: Executing import_all_tables.sh script 66
Figure 72: Staging database and tables imported 67
Figure 73: staging_retail database tables stored in HDFS 67
Figure 74: Creating xyz_dmart Database along with Dimension and Fact Tables – Part 1 68
Figure 75: Creating xyz_dmart Database along with Dimension and Fact Tables – Part 2 69
Figure 76: Creating xyz_dmart Database along with Dimension and Fact Tables – Part 1 70
Figure 77: Inserting Data into External Table 71
Figure 78: Inserting Dato into Fact Table 71
Figure 79: All table list stored in xyz_dmart 72
Figure 80: Table Stored in HDFS file system 72
Figure 81: Query of Total Profit of each Department. 73
Figure 82: Result on Profit of each Department 73
Figure 83: Top Revenue Generating Product Query 73
Figure 84: List of Top Revenue Generating Product 74
Figure 85: Top Selling/Popular Product Query 74

Figure 86: Top Selling/Popular Product List..... 74

List of Tables

Table 1: Fact Table fact_sales Table Structure 35
Table 2: Dimension Table dimension_categories Table Structure 36
Table 3: Dimension Table dimension_date Table Structure 36
Table 4: Dimension Table dimension_departments Table Structure 37
Table 5: Dimension Table dimension_order_items Table Structure 37
Table 6: Dimension Table dimension_orders Table Structure 37
Table 7: Dimension Table dimension_products Table Structure 37
Table 8: Attributes and Description in Fact Table 38
Table 9: Detail Attributes List of fact_sales table 39
Table 10: dimension_categories Table details with Attributes 43
Table 11: dimension_date Table details with Attributes 44
Table 12: dimension_departments Table Details with Attributes 48
Table 13: dimension_order_items Table details with Attributes 49
Table 14: dimension_orders Table details with Attributes 51
Table 15: dimension_products Table details with Attributes 52

List of Abbreviation

BI	Business intelligence
DW	Data Warehouse
HQL	Hive-QL
PK	Primary Key
FK	Foreign Key
ORC	Optimized Row Columnar
ETL	Extract, Transform and Load
HDFS	Hadoop Distributed File System
SQL	Structured Query Language

Introduction

XYZ retail store, founded in 2012, is a retail service provider for a better shopping experience in the industry. XYZ runs on data, unlike any business, which offers little value without analytics. This report provides the solutions for the company on how to use analytical tools to optimise better, ease, and predict customer behaviour for shopping. It also generates reports which help stakeholders in the decision-making process for the inflow and outflow of the products. The analytical solution presented in the report will help XYZ retail store increase their profit and optimise their resources. This report demonstrates the stores' better predicting model for forecasting its sales, flow of the products, customer segmentation and the ability to process data to make behaviour predictions of individual customers and the markets, diagnose systems or situations, or prescribe actions for customers. XYZ retail also needed a file-storage solution with high scalability to hold data and faster processing of data using Apache Hadoop platform, Apache Sqoop to import and export data from the database in Hive for further analysis of Big Data.

The provided retail_db is used to process all the requirements of sales analysis which is an essential business process of the store. The report's goal is to design and develop an excellent analytical solution to efficiently manage the inventory for-casting the items that are more likely to be in high demand based on the seasons and location of the company outlets.

This document mainly emphasises the following five reporting and analysis requirements:

1. Monthly/Weekly report of all products as per departments.
2. Profit analysis for different departments.
3. Top revenue-generating items.
4. Find the reasons for rejecting/cancelled orders.
5. Daily and monthly analysis of top-selling products.

1. Data Warehouse & Warehousing

Data Warehouse is a decision support database maintained separately from the operational database. Data Warehousing is an essential data collection and management process where the data collection include different sources stored in a Data Warehouse, an environment to store all the business data. It is a core BI system built for data analysis and reporting. (Taylor, 2021)

1.1 ETL Process

ETL process encompasses data extraction, transformation, and load. The abbreviation implies a neat, three-step process – extract, transform, load. (Talend SA, 2021)

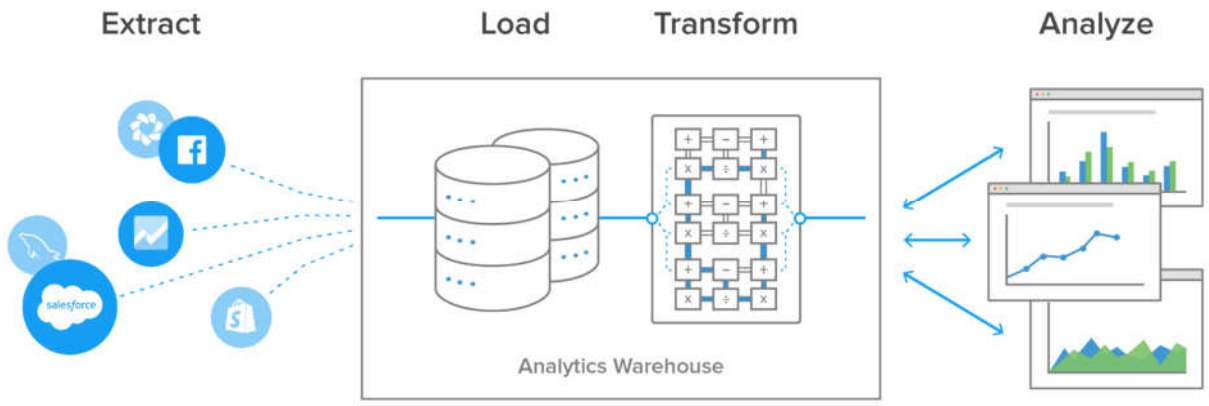


Figure 1: A contemporary ETL process using a Data Warehouse

1.1.1 Creating Data Warehouse

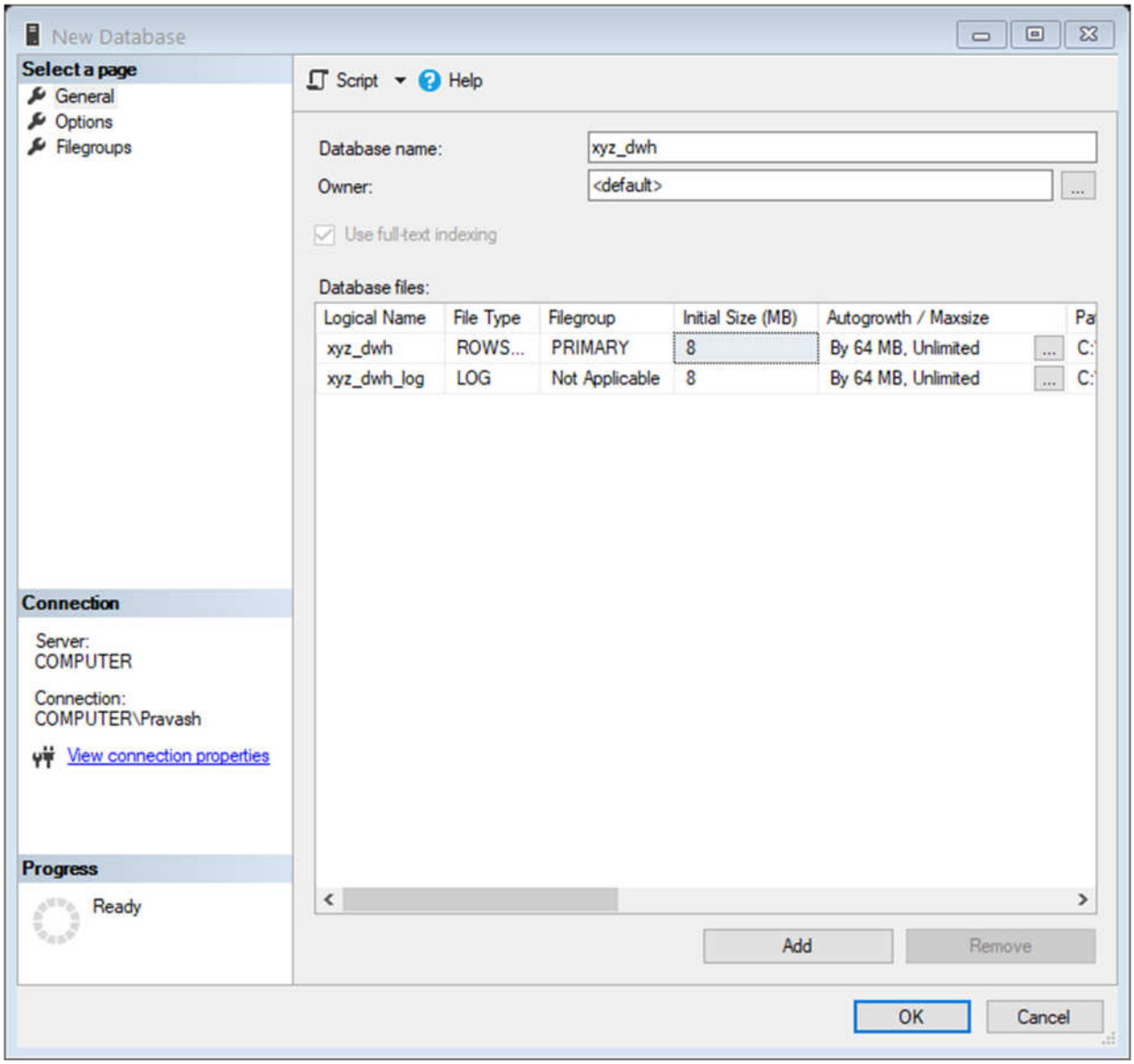


Figure 2: Creating xyz_dwh Data Warehouse in SQL Server.

1.1.2 Data Flow Task: Using ETL Process to load Source Data

Step 1: Using Microsoft Visual Studio (SSDT) software, create a data flow task to extract data from the source using the ETL process.

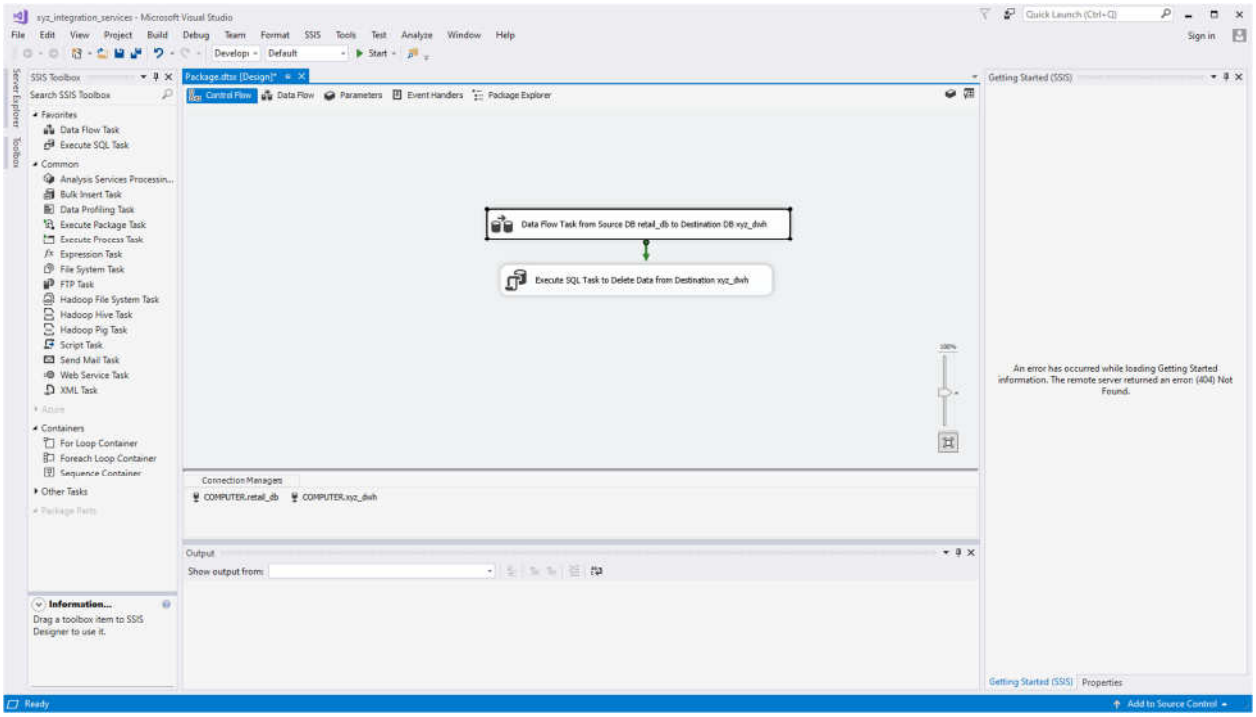


Figure 3: Creating a Data Flow Task.

Step 2: The tables from the source data is analysed to use the same database scheme to create a Data Warehouse database.

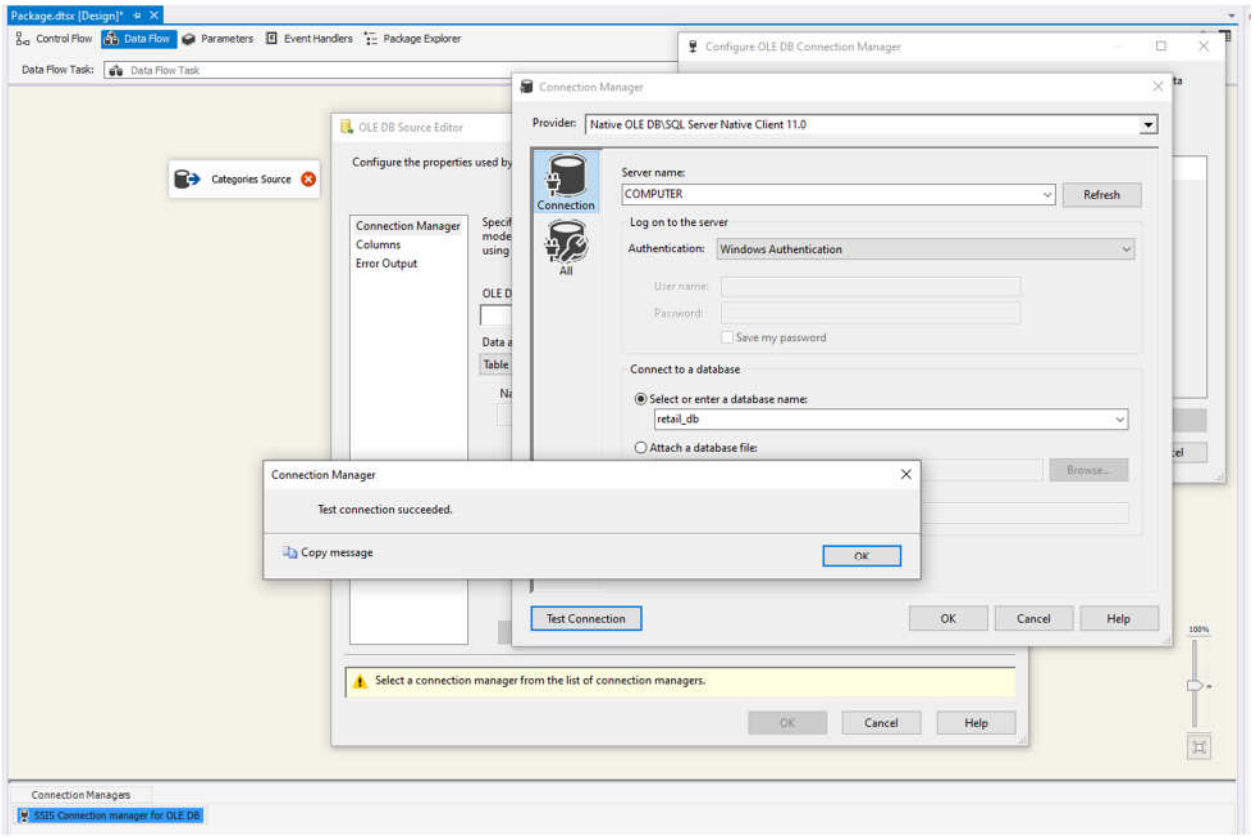


Figure 4: Testing the DB Connection between Source and Data Warehouse Database

Step 3: Creating all tables using the same source tables' schema into the destination Data Warehouse database.

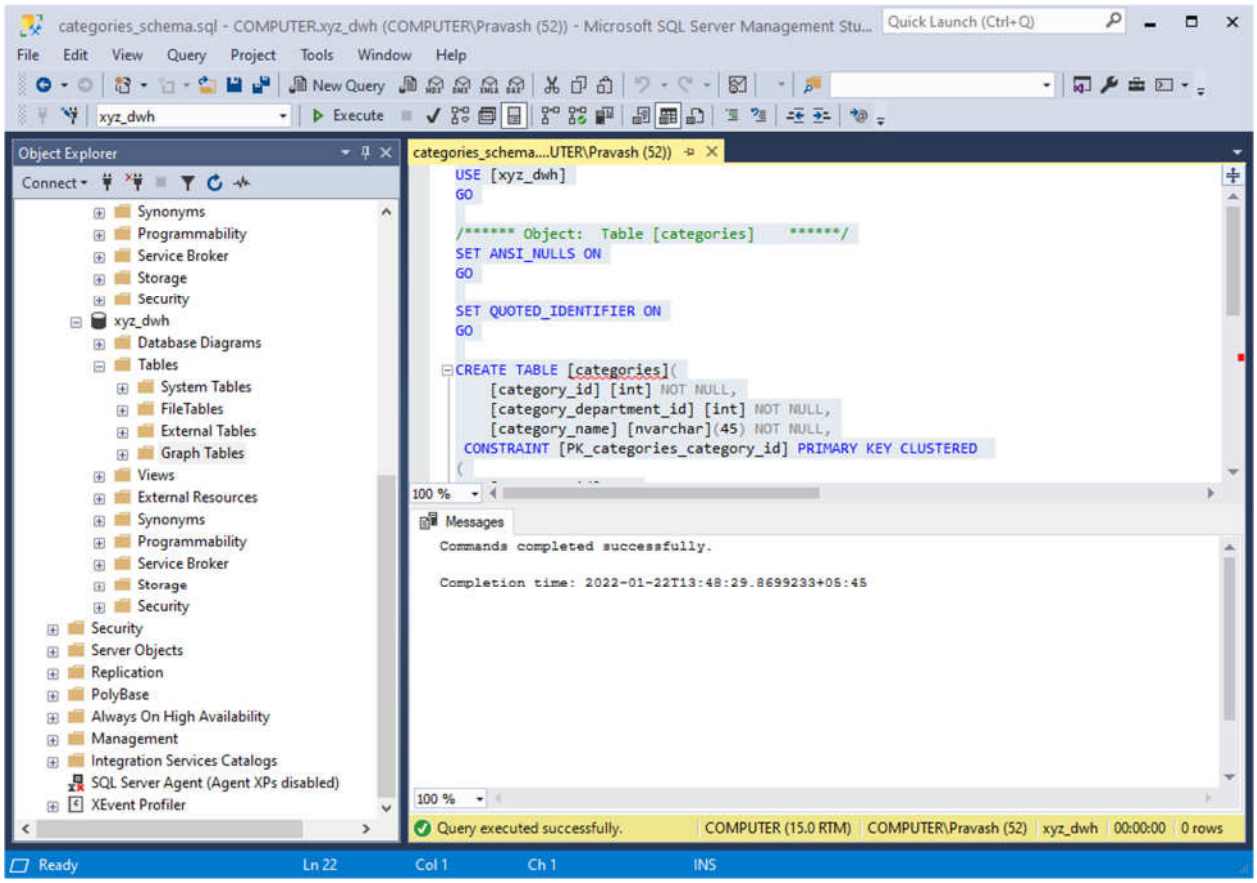


Figure 5: Creating schema of the categories table in Data Warehouse [xyz_dwh]

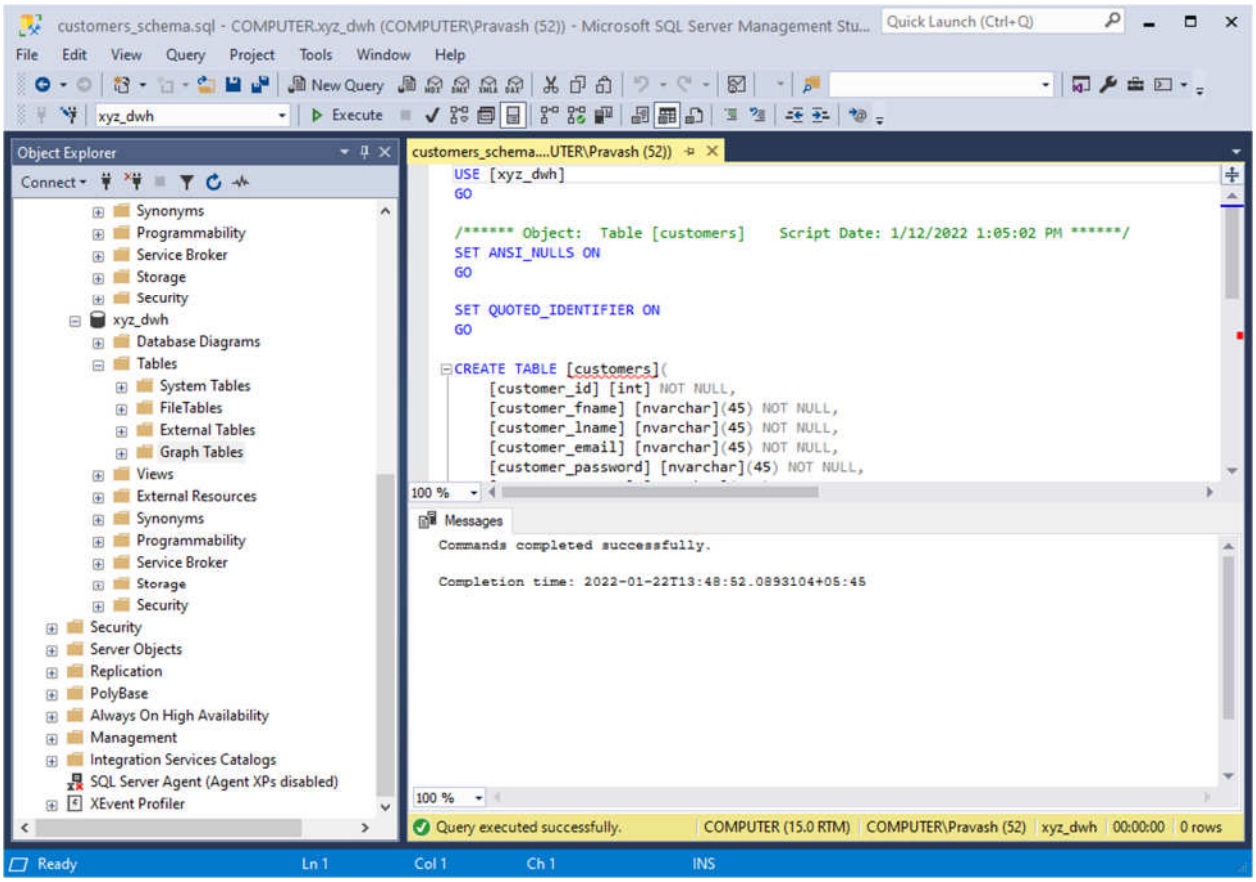


Figure 6: Creating schema of the "customers" table in Data Warehouse [xyz_dwh]

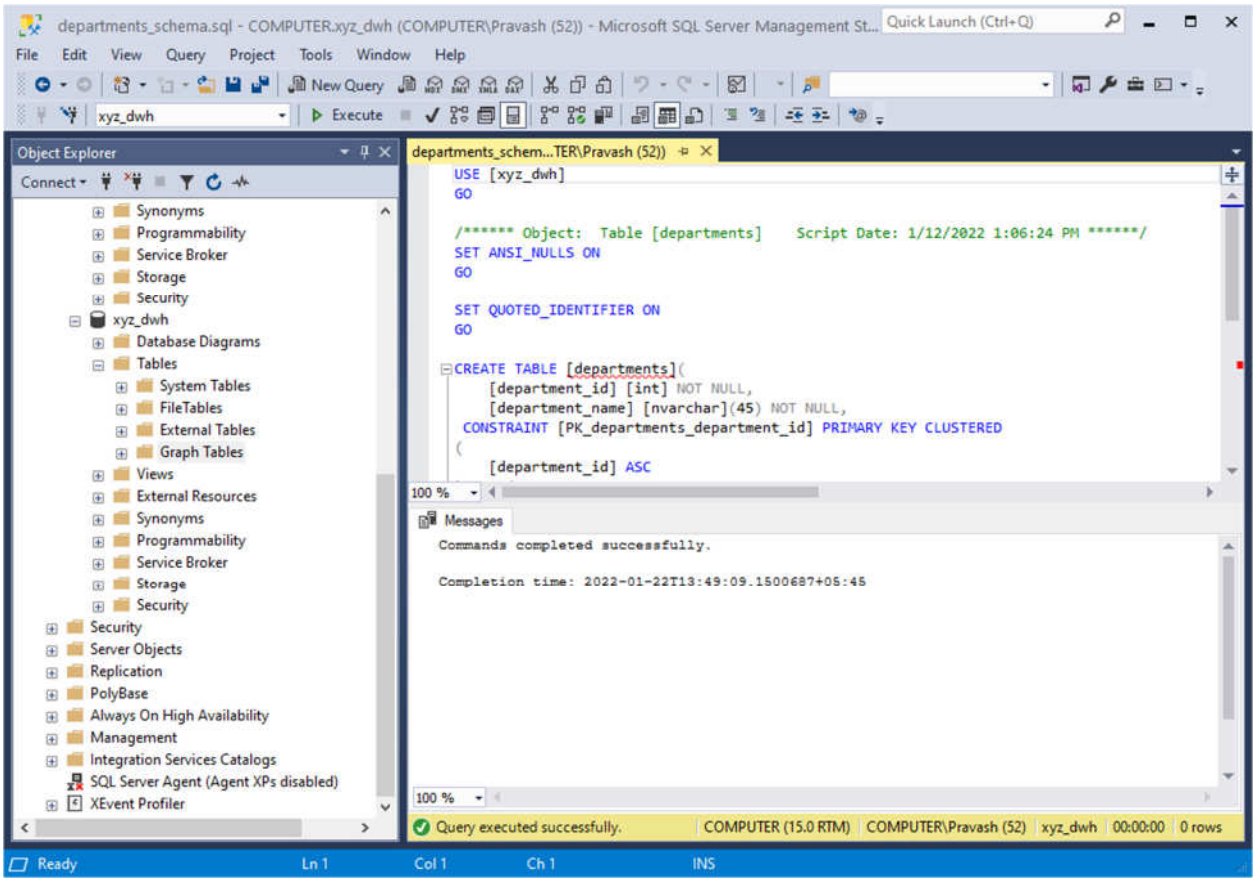


Figure 7: Creating schema of the "departments" table in Data Warehouse [xyz_dwh]

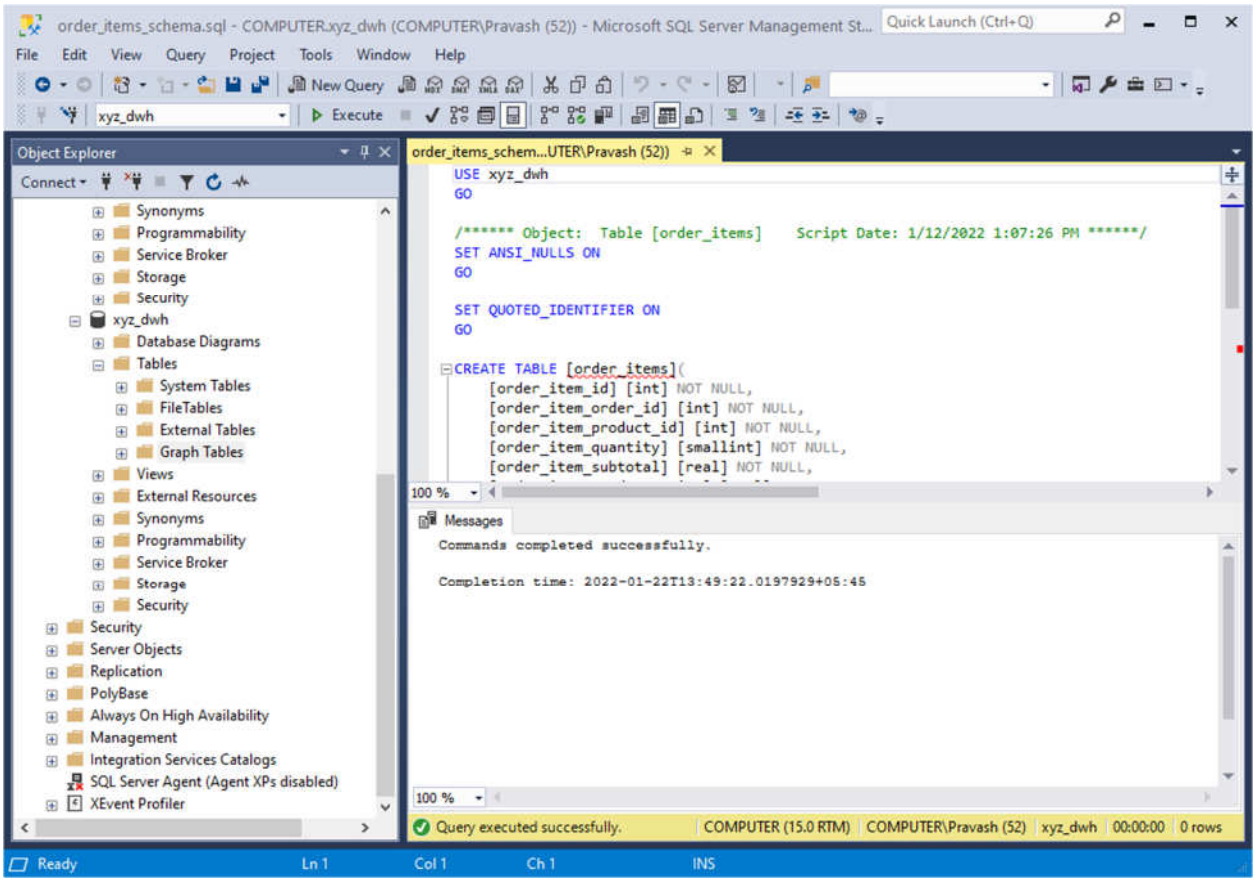


Figure 8: Creating schema of the orders_items table in Data Warehouse [xyz_dwh]

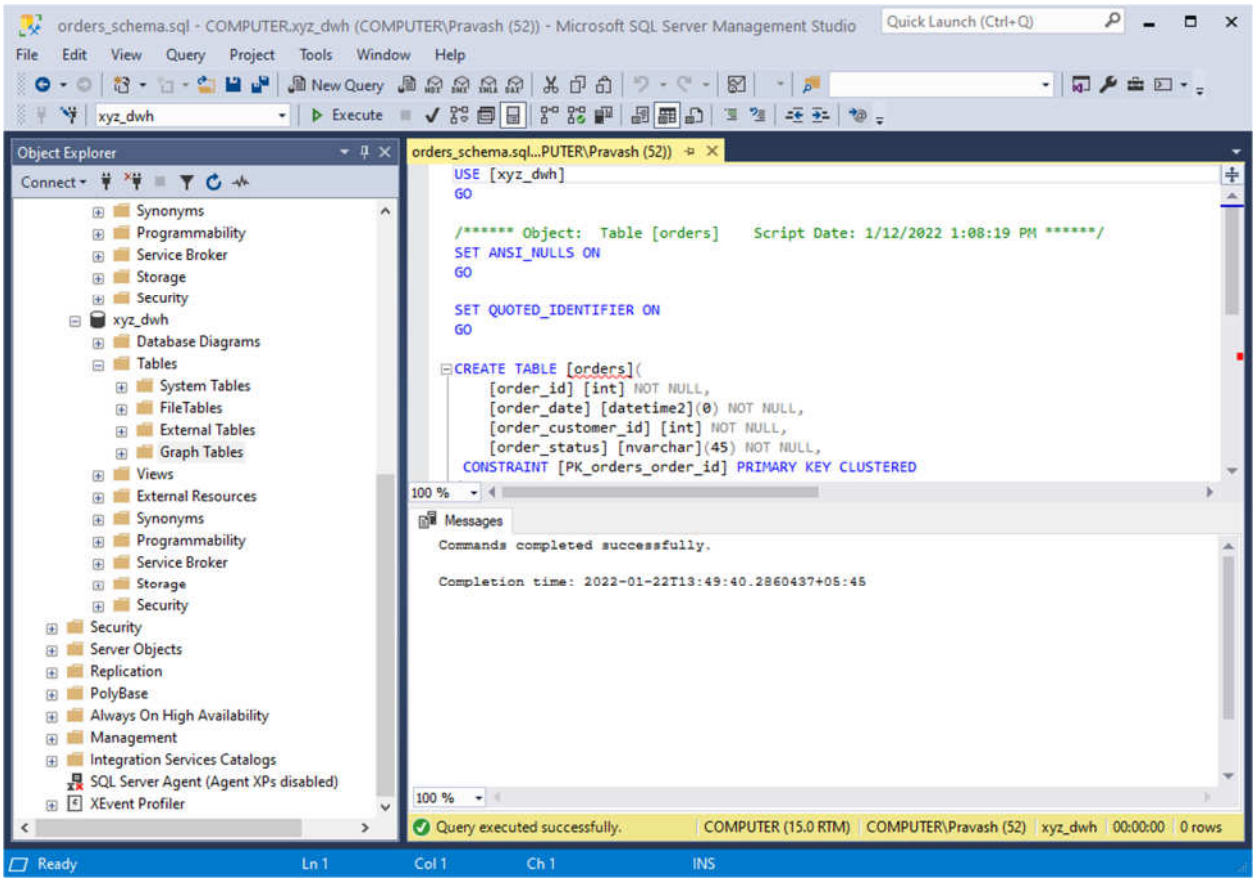


Figure 9: Creating schema of the orders table in Data Warehouse [xyz_dwh]

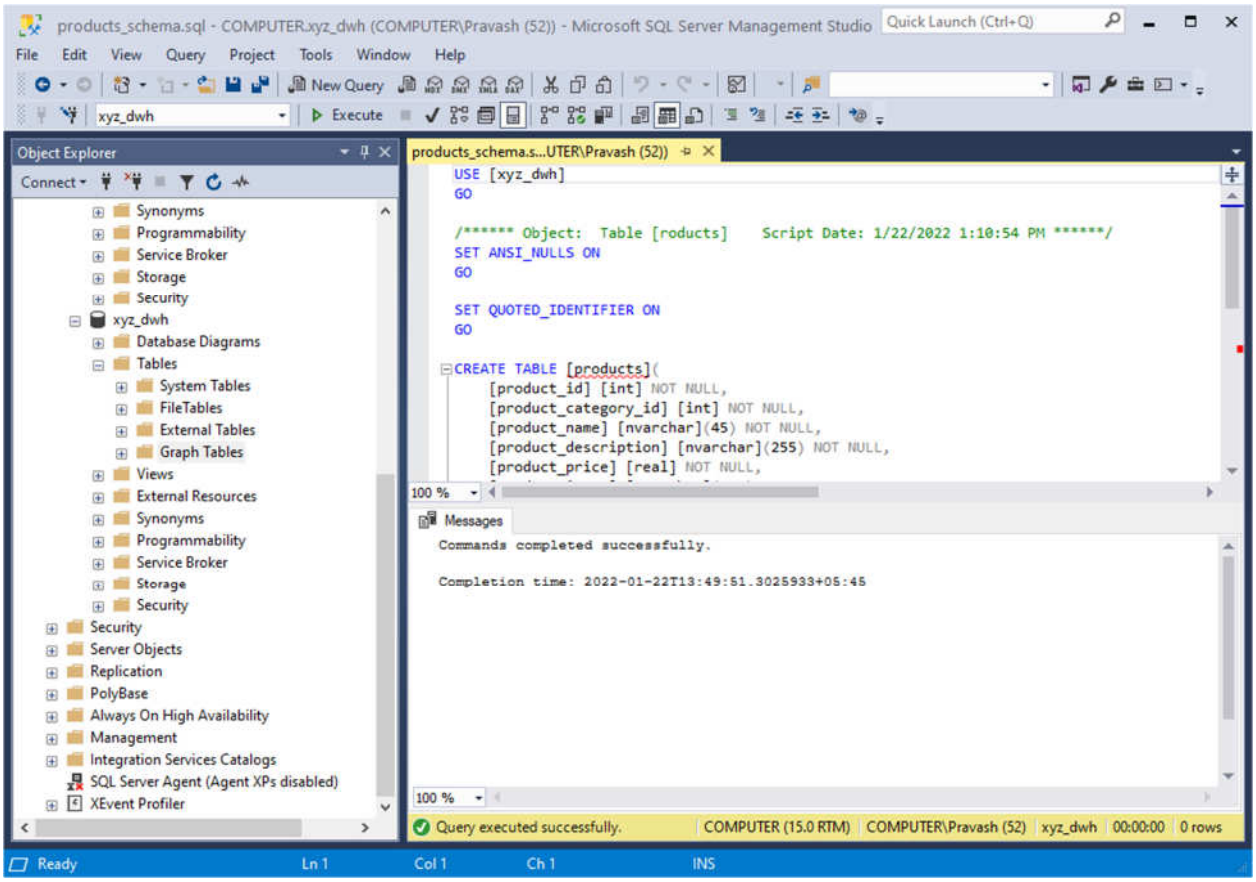


Figure 10: Creating schema of the products table in Data Warehouse [xyz_dwh]

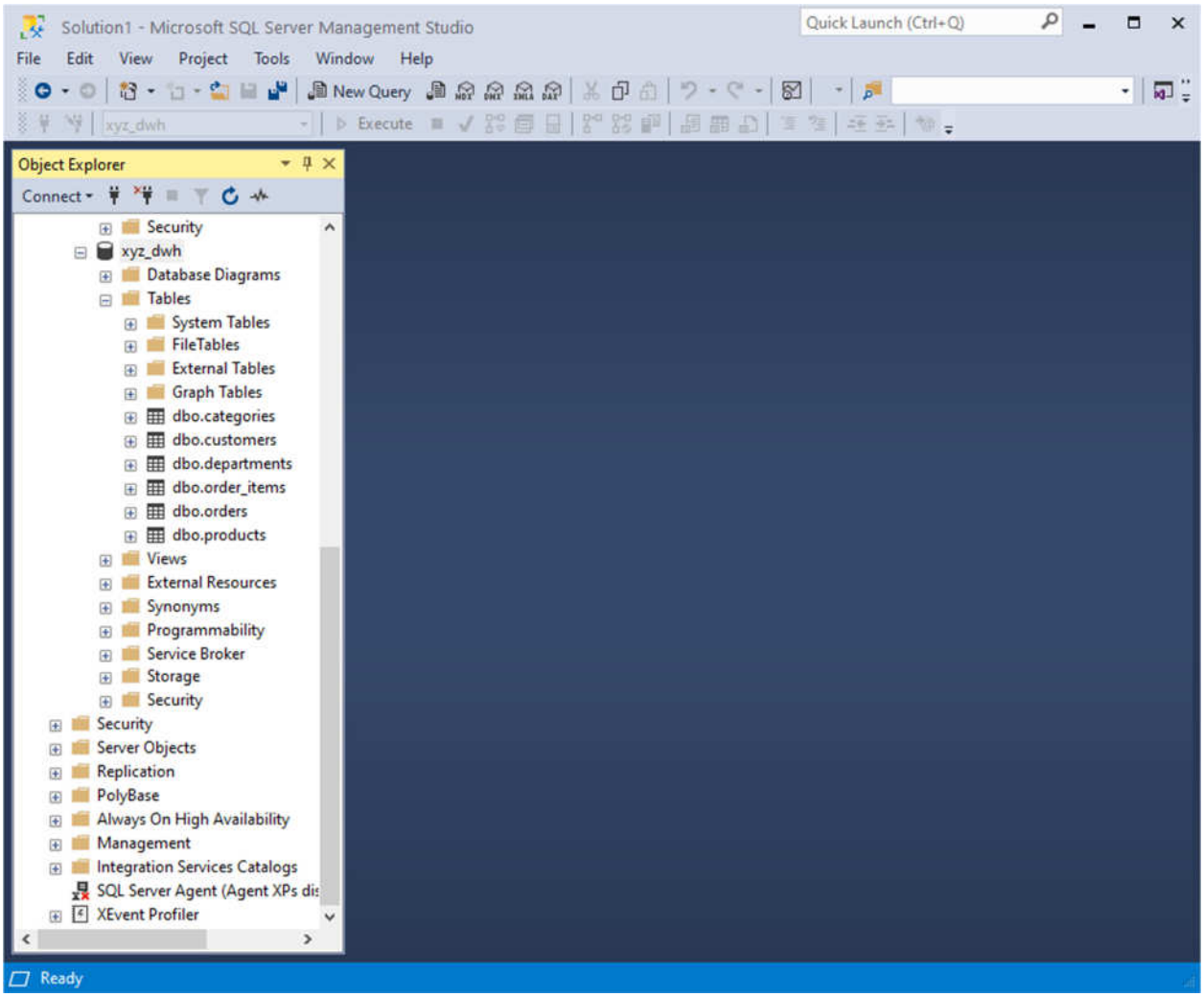


Figure 11: Successful Creation of all tables in Data Warehouse [xyz_dwh]

1.1.3 Configure OLE DB Source and OLE DB Destination.

Configure the properties used by a data flow to obtain data from OLE DB Source and OLE DB Destination. For example, along with mapping the column names of both the OLE DB Source and OLEDB Destination table, check whether the number of columns is equal, and configure the properties used to insert data into a relational database using an OlE DB provider.

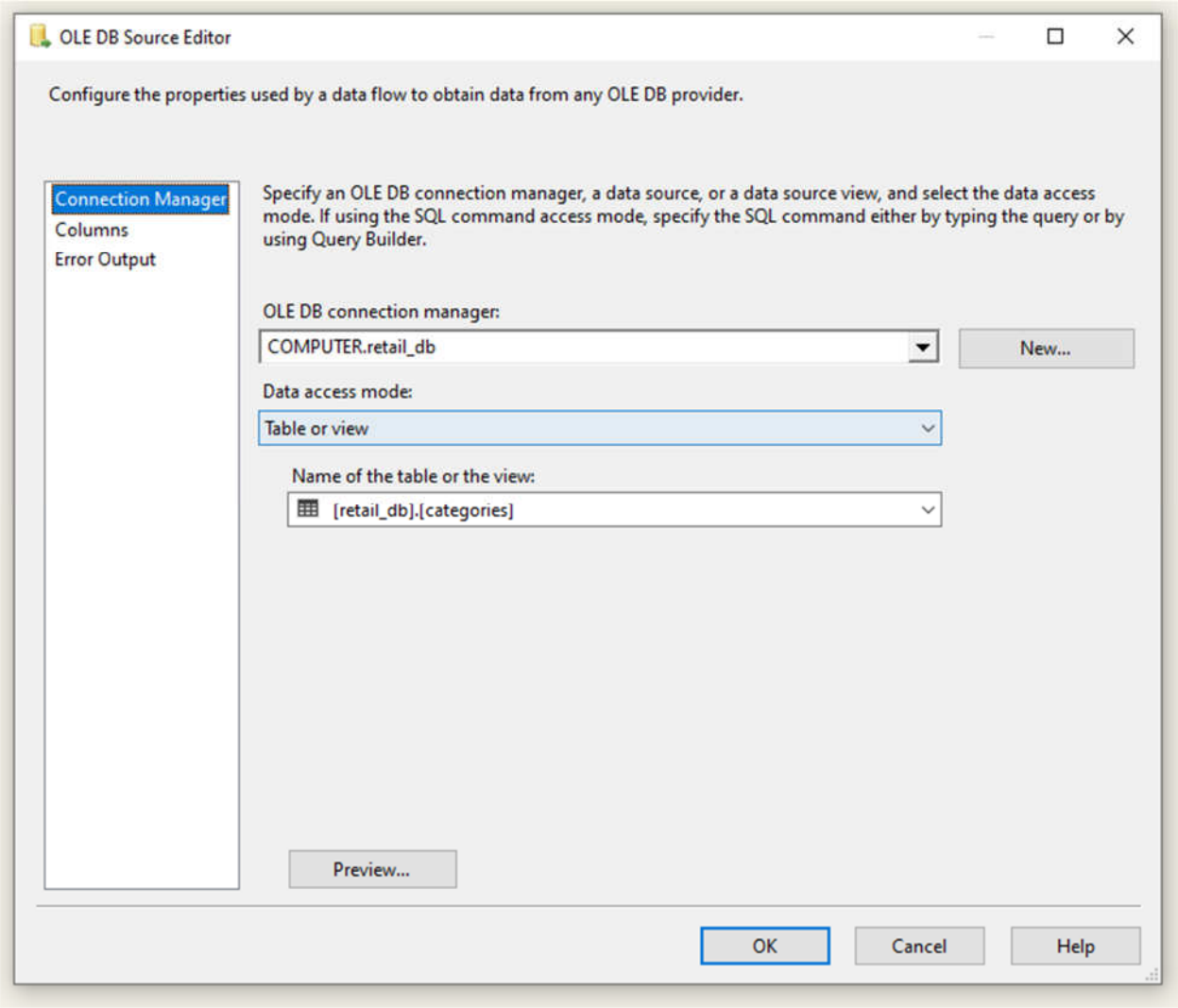


Figure 12: Configuring Source categories table.

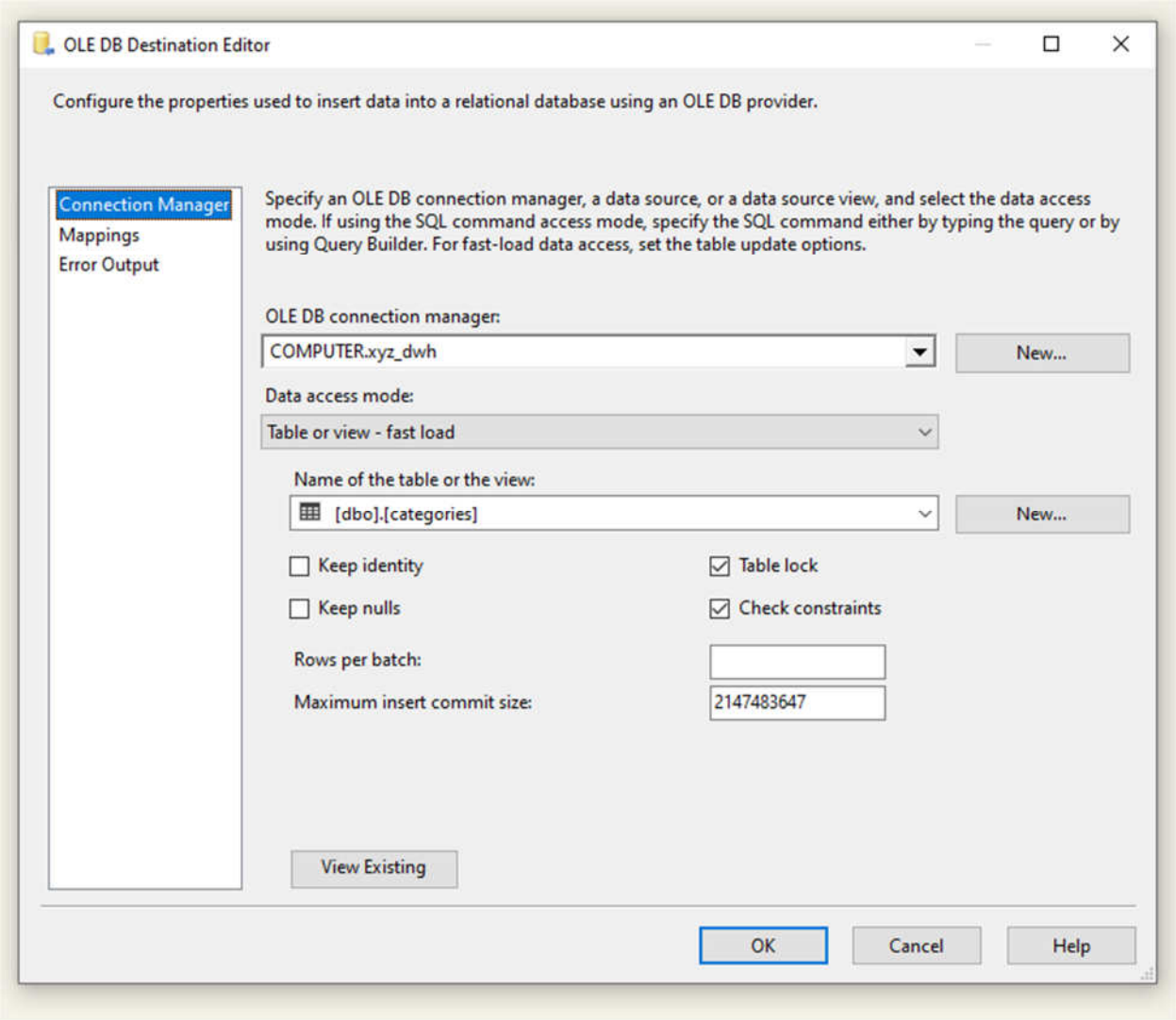


Figure 13: Configuring Destination categories table.

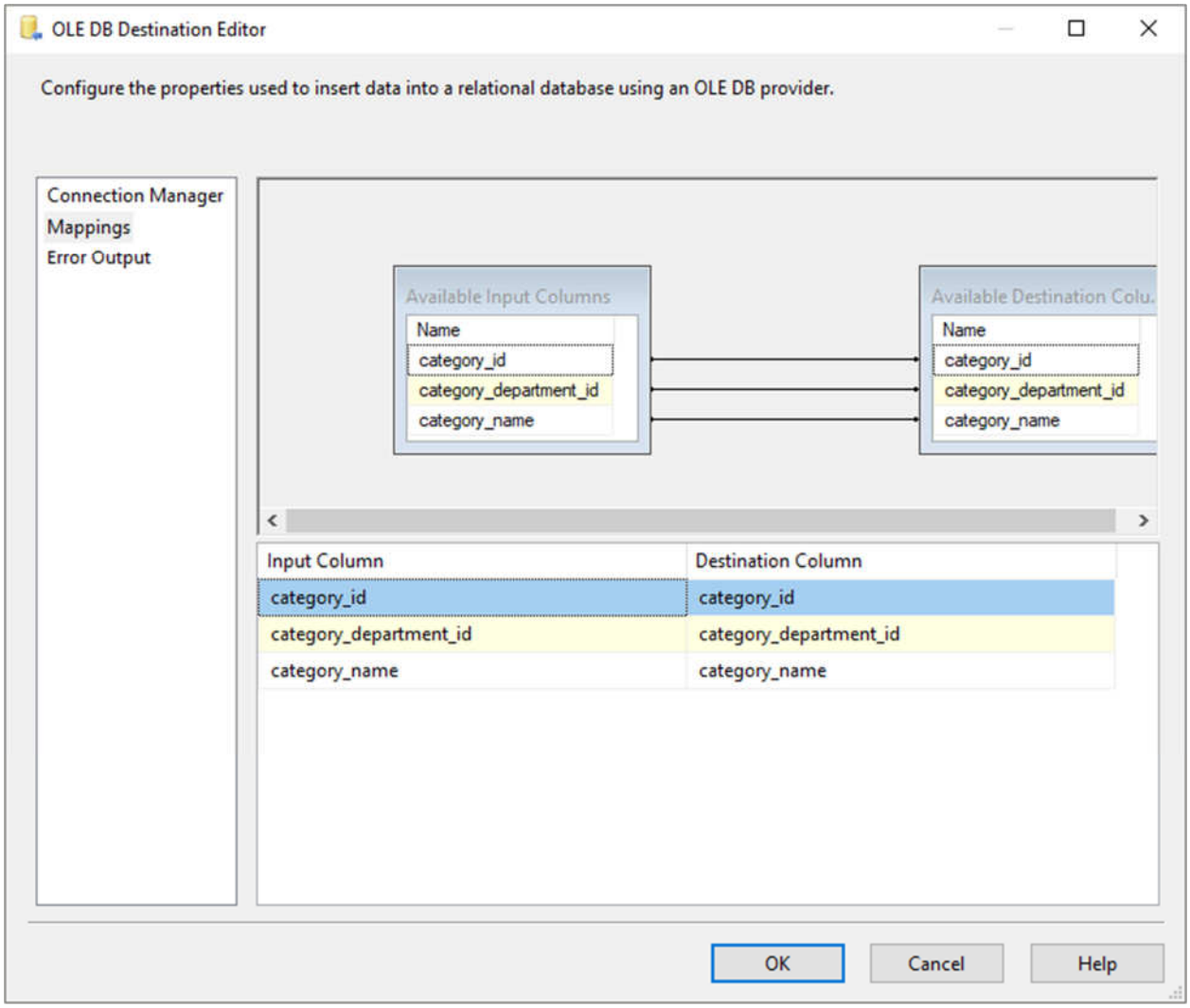


Figure 14: Configuring Destination and Source categories table column.

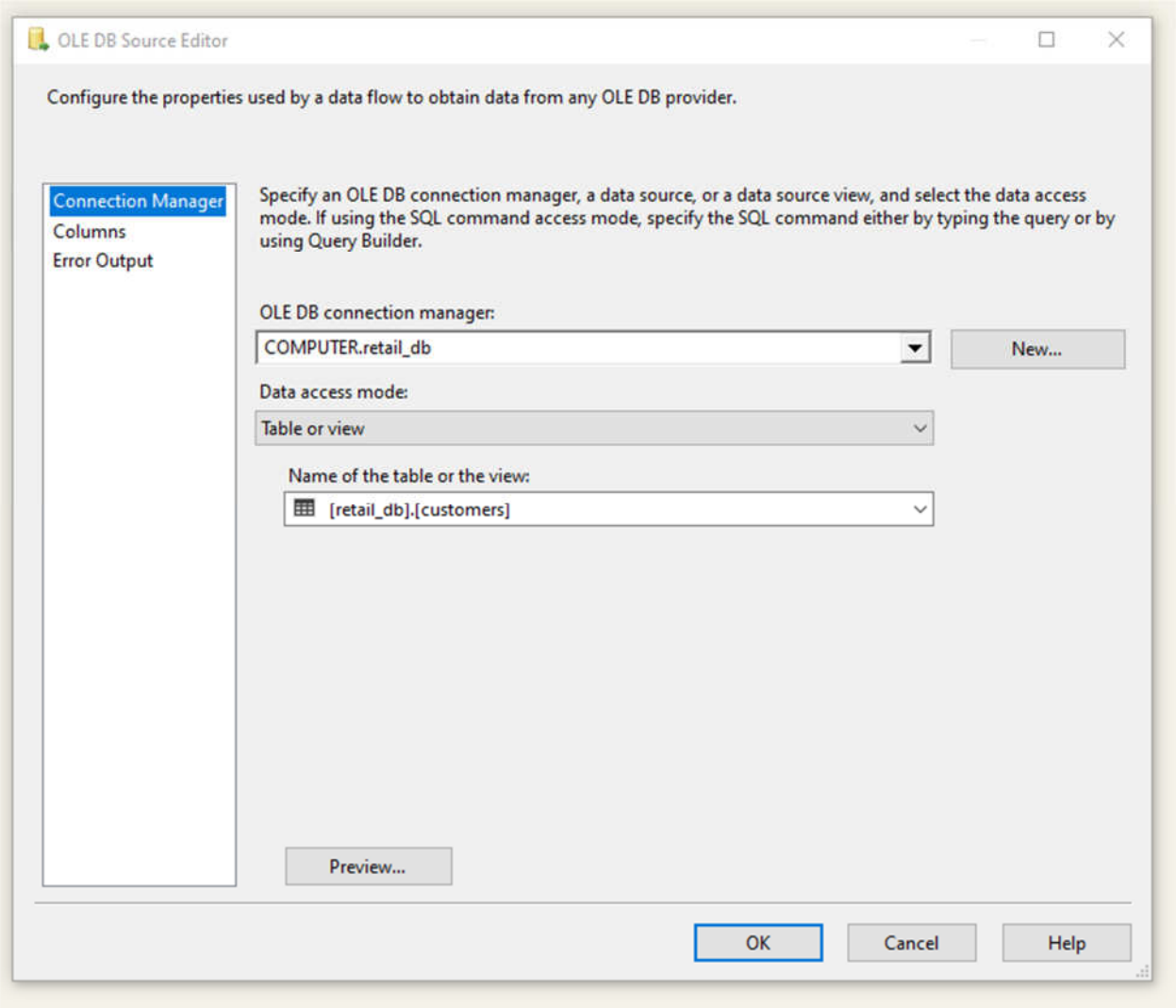


Figure 15: Configuring Source customers table.

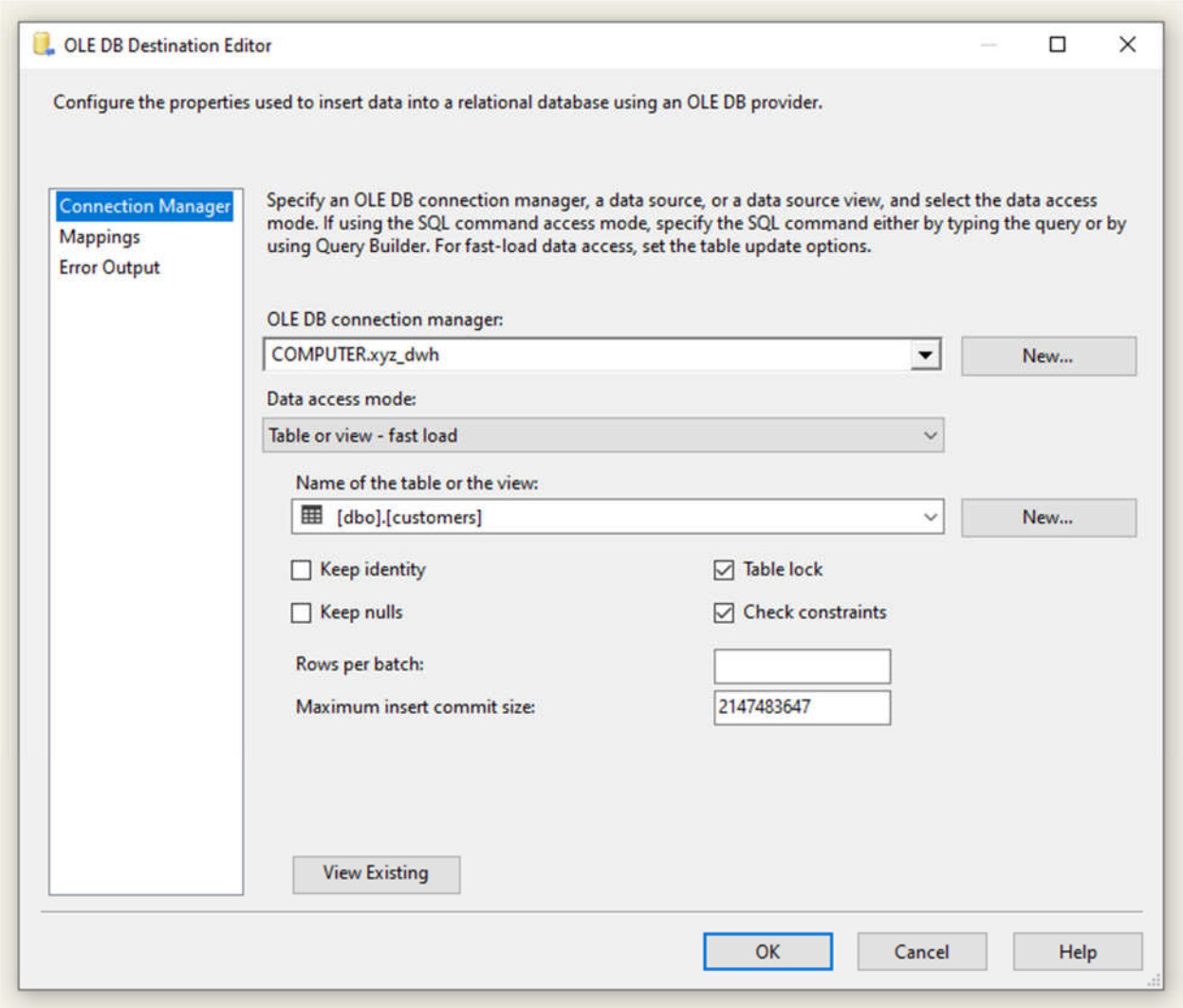


Figure 16: Configuring Destination customers table.

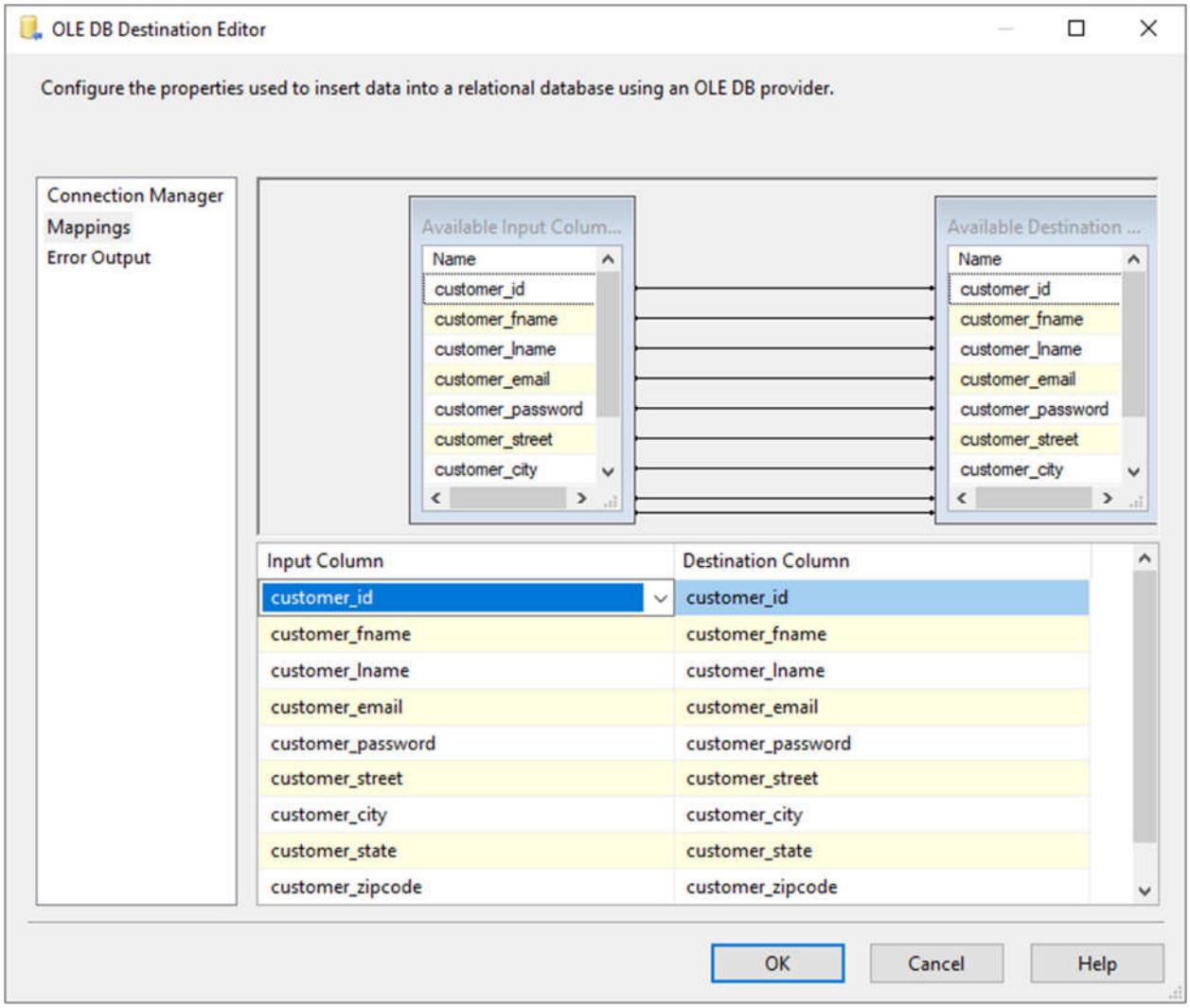


Figure 17: Configuring Destination and Source customers table column.

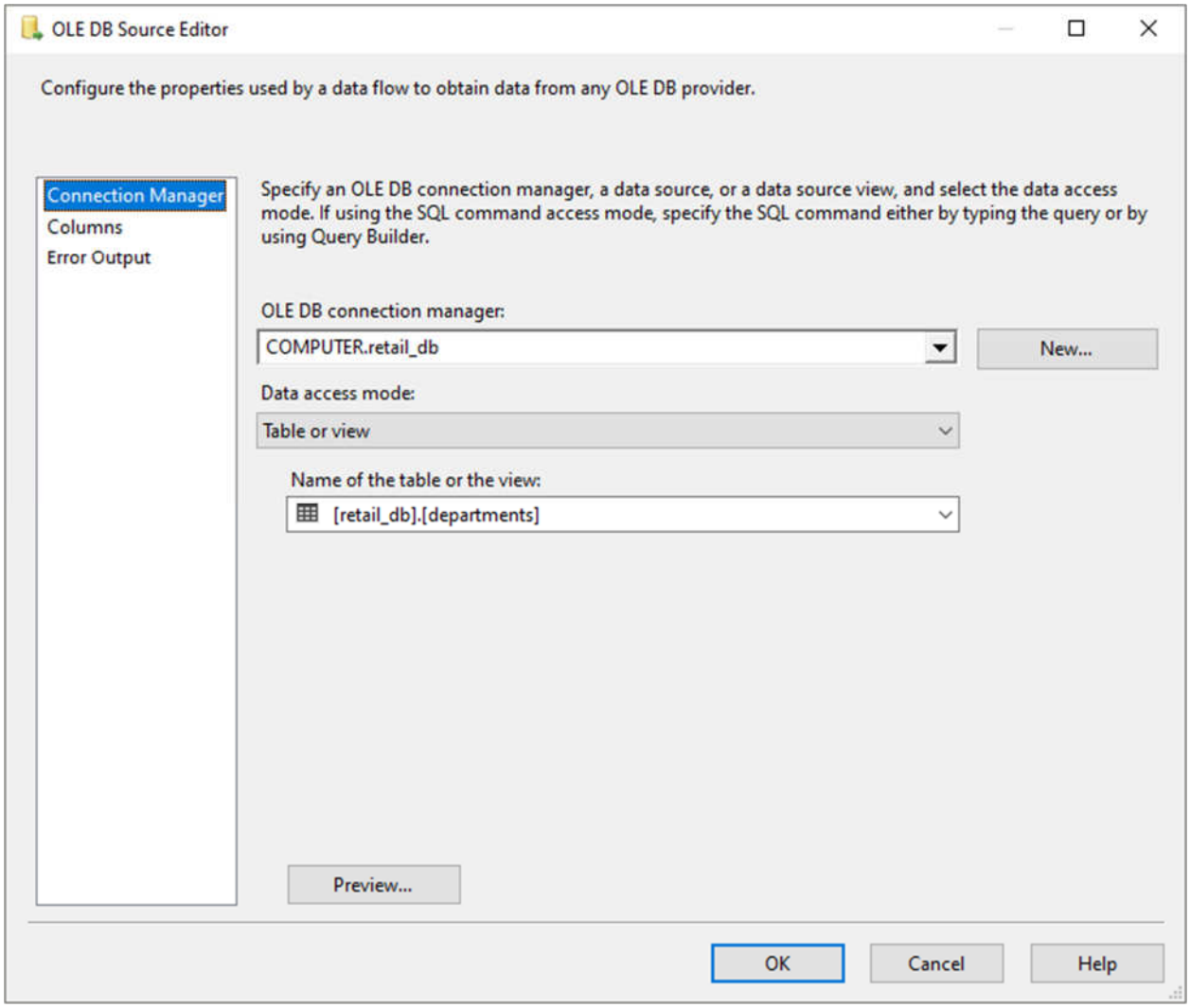


Figure 18: Configuring Source departments table.

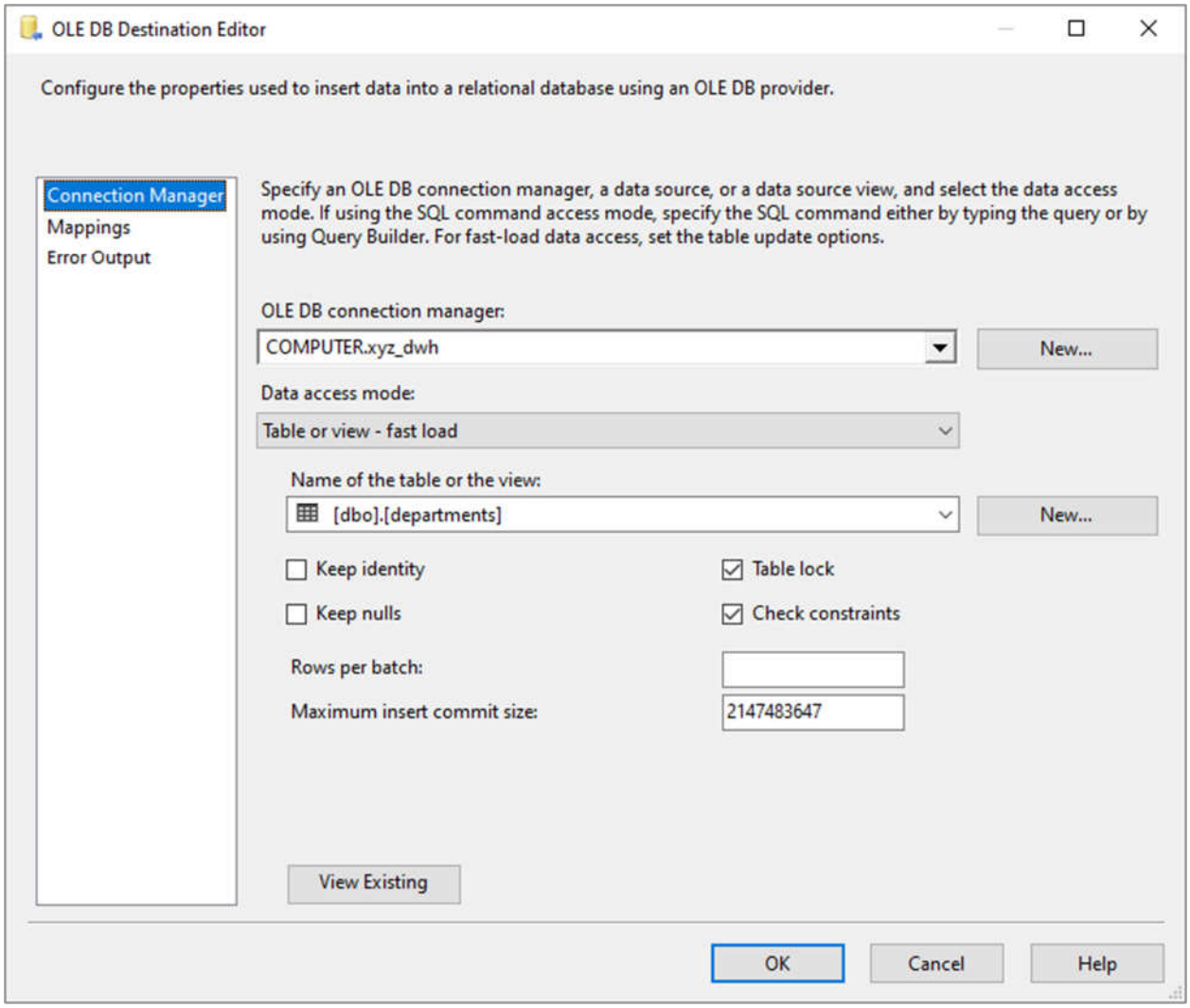


Figure 19: Configuring Destination departments table.

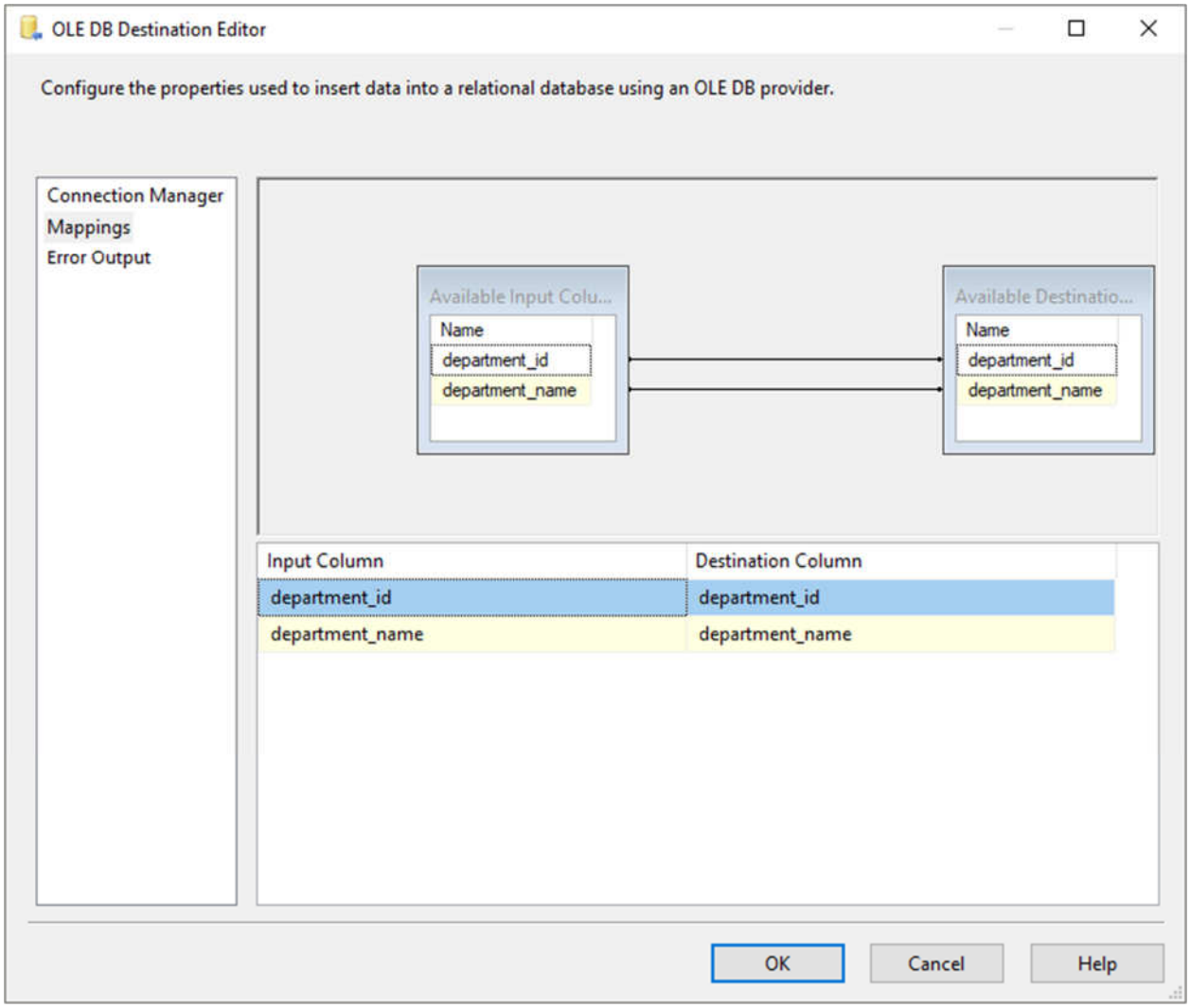


Figure 20: Configuring Destination and Source departments table column.

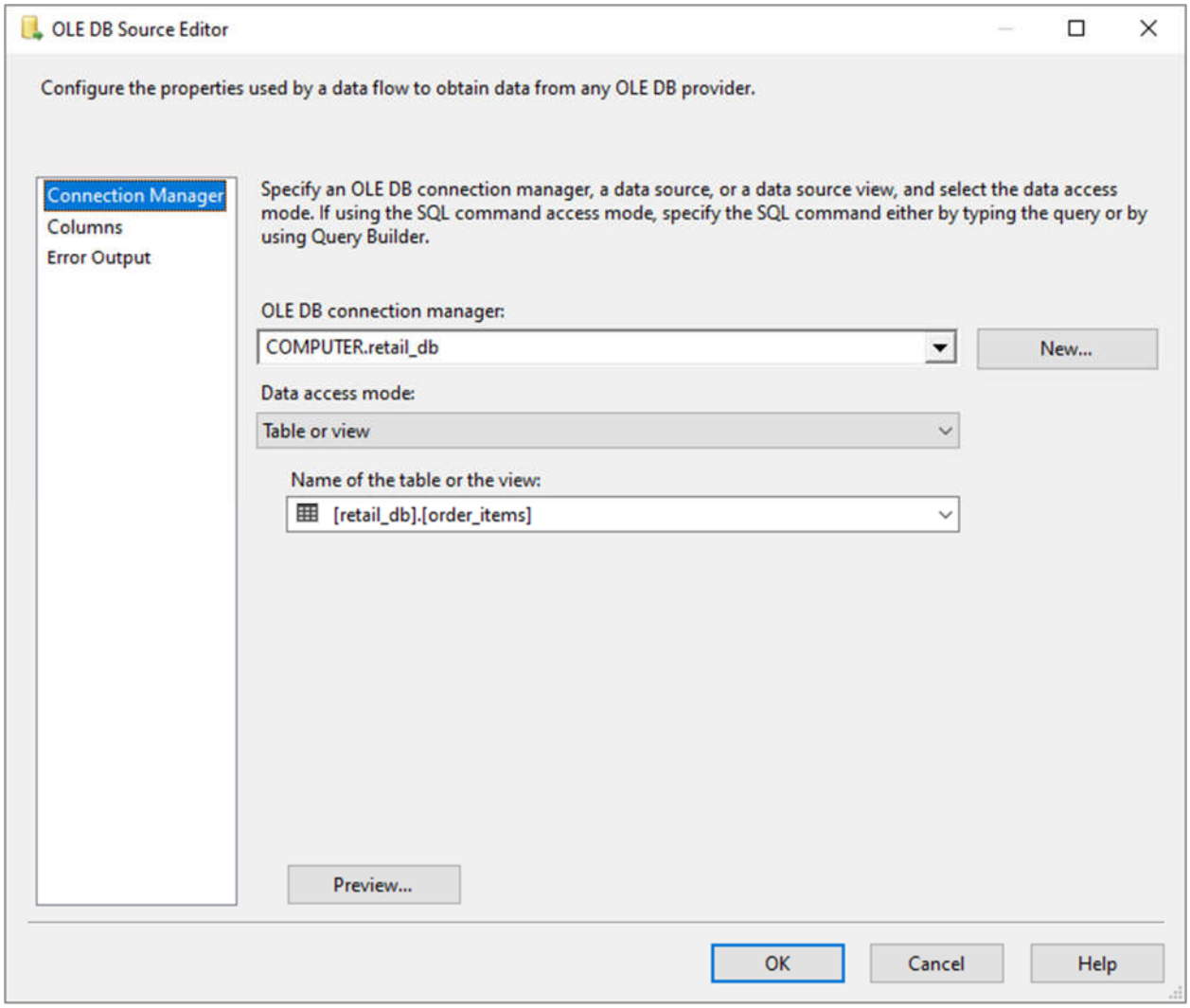


Figure 21: Configuring Source order_items table.

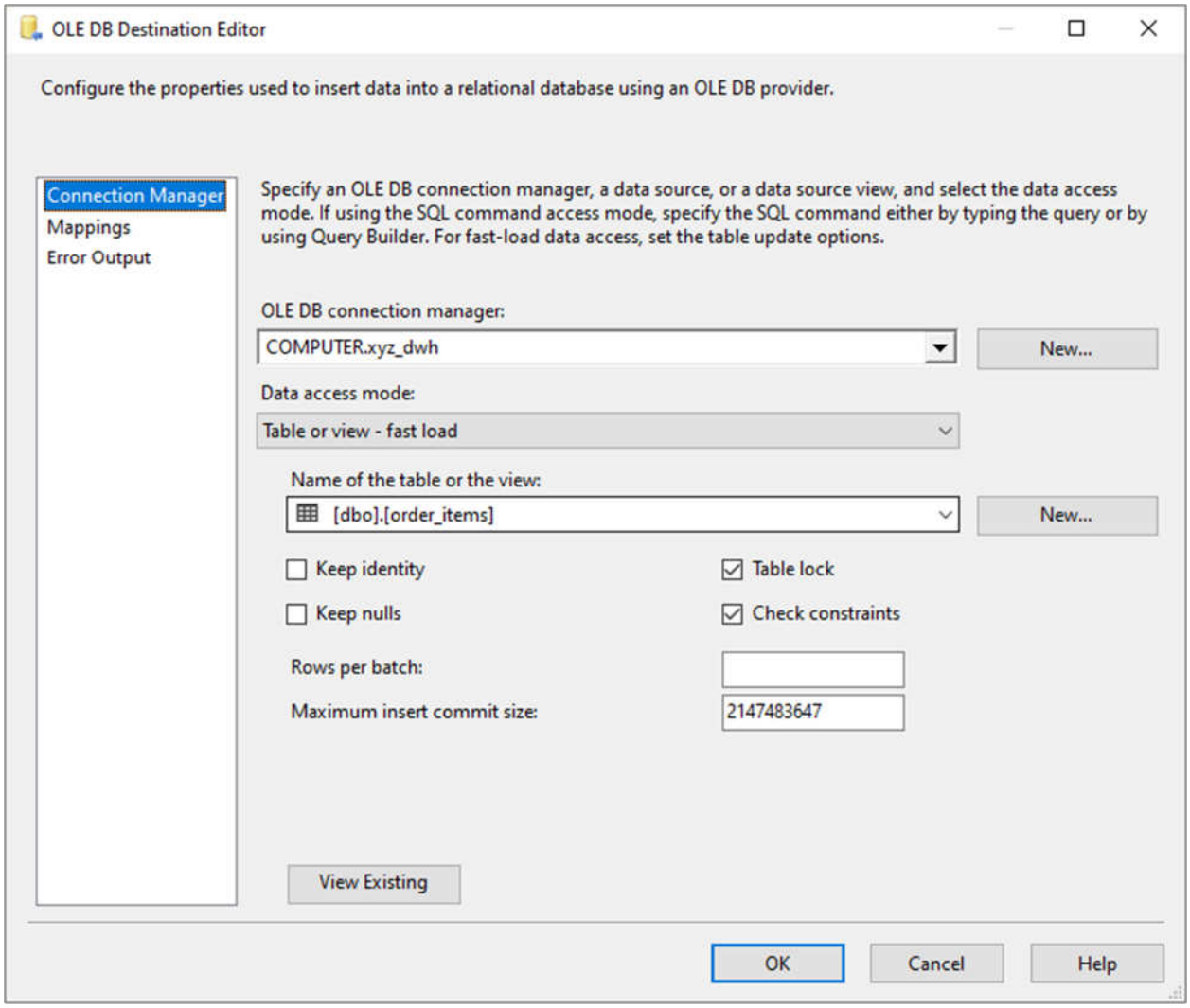


Figure 22: Configuring Destination order_items table.

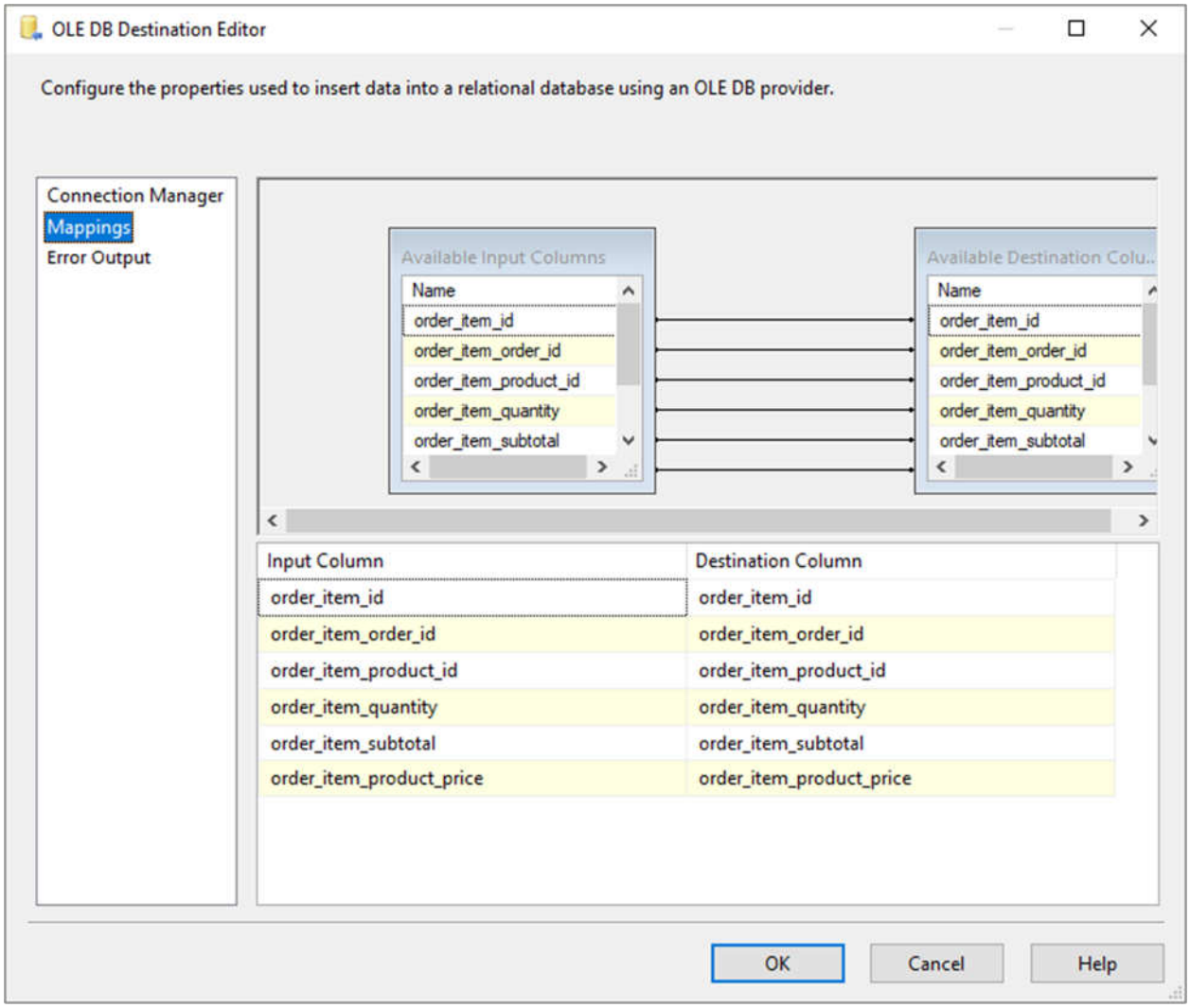


Figure 23: Configuring Destination and Source order_items table column.

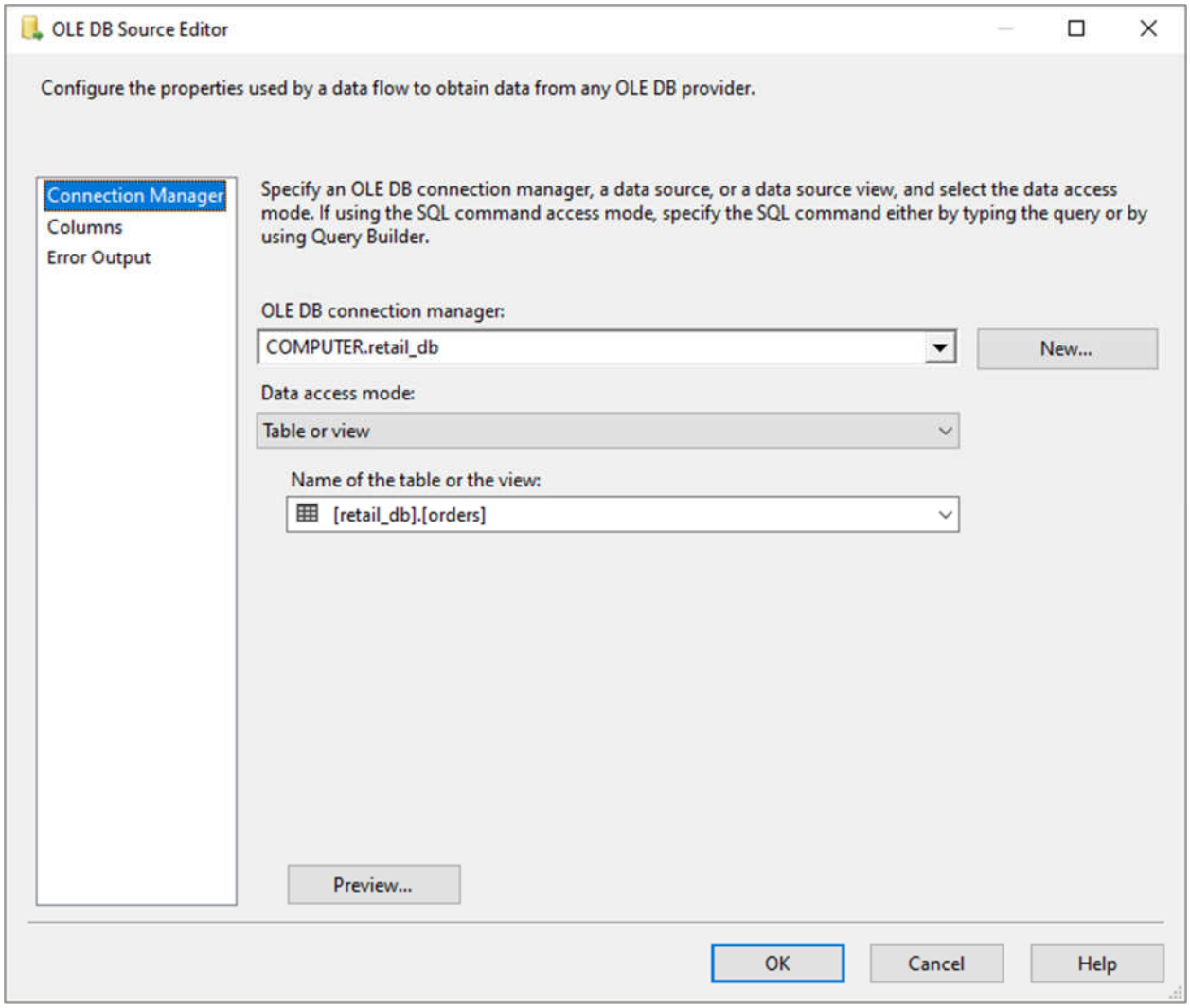


Figure 24: Configuring Source orders table.

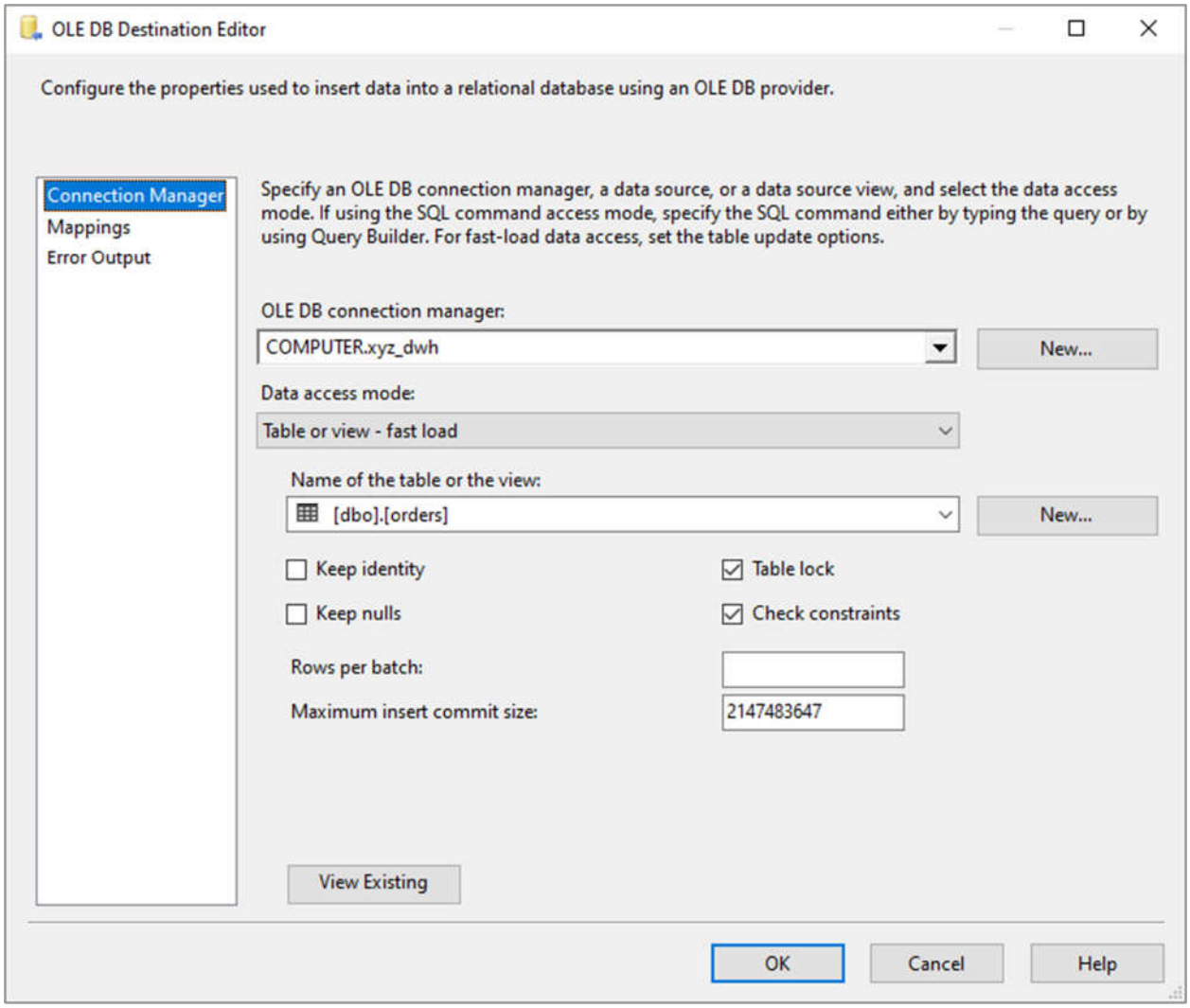


Figure 25: Configuring Destination orders table.

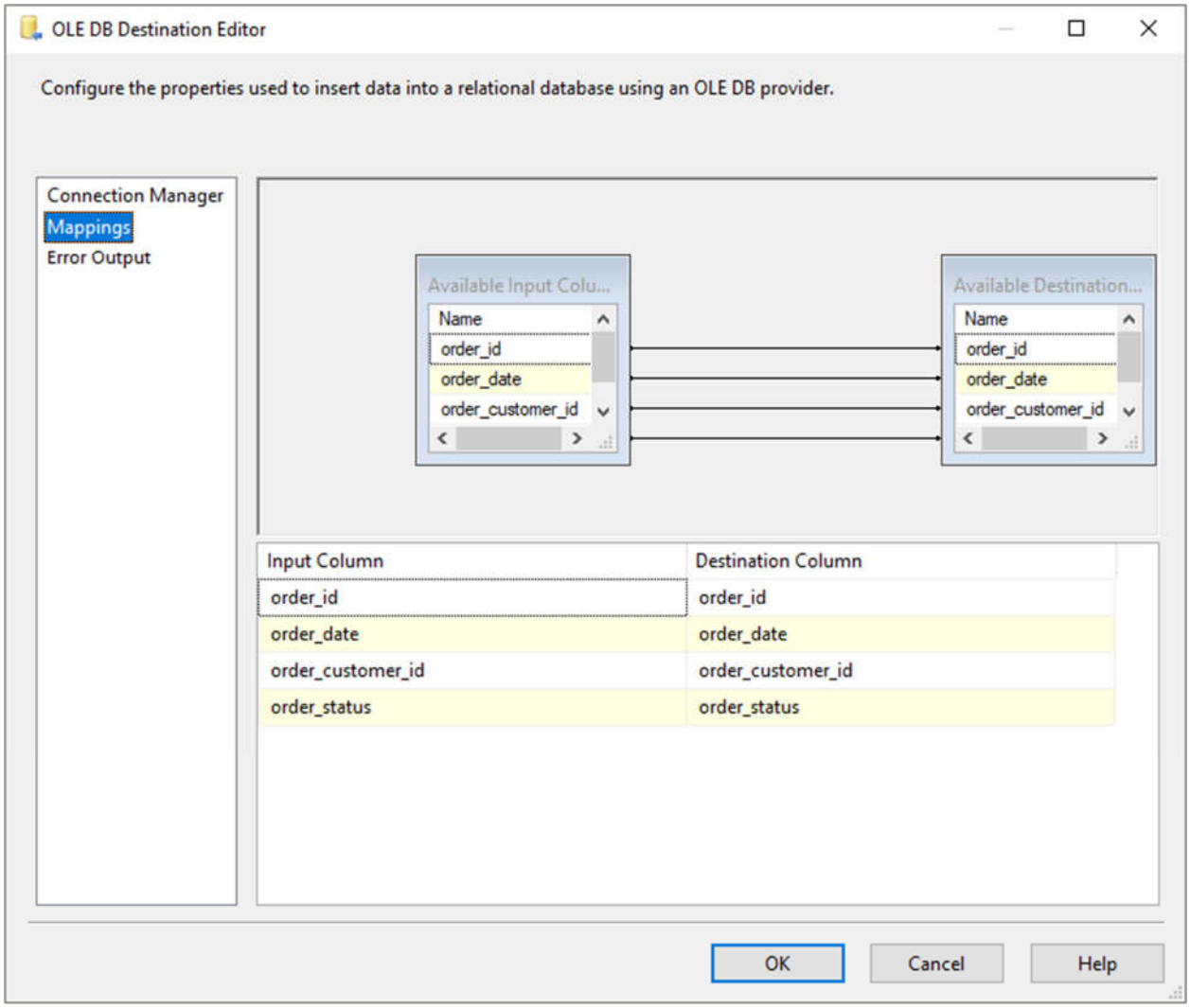


Figure 26: Configuring Destination and Source orders table column.

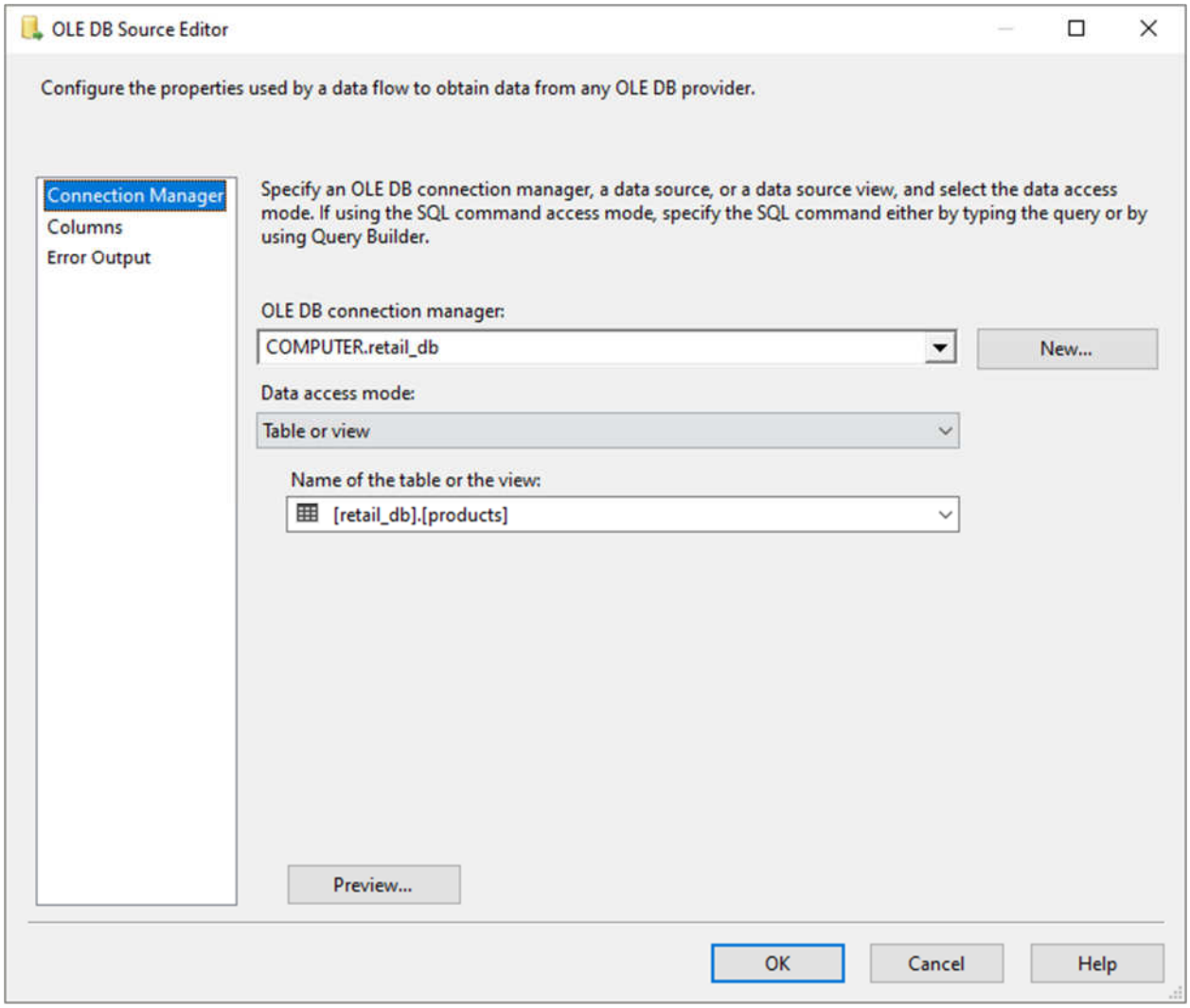


Figure 27: Configuring Source products table.

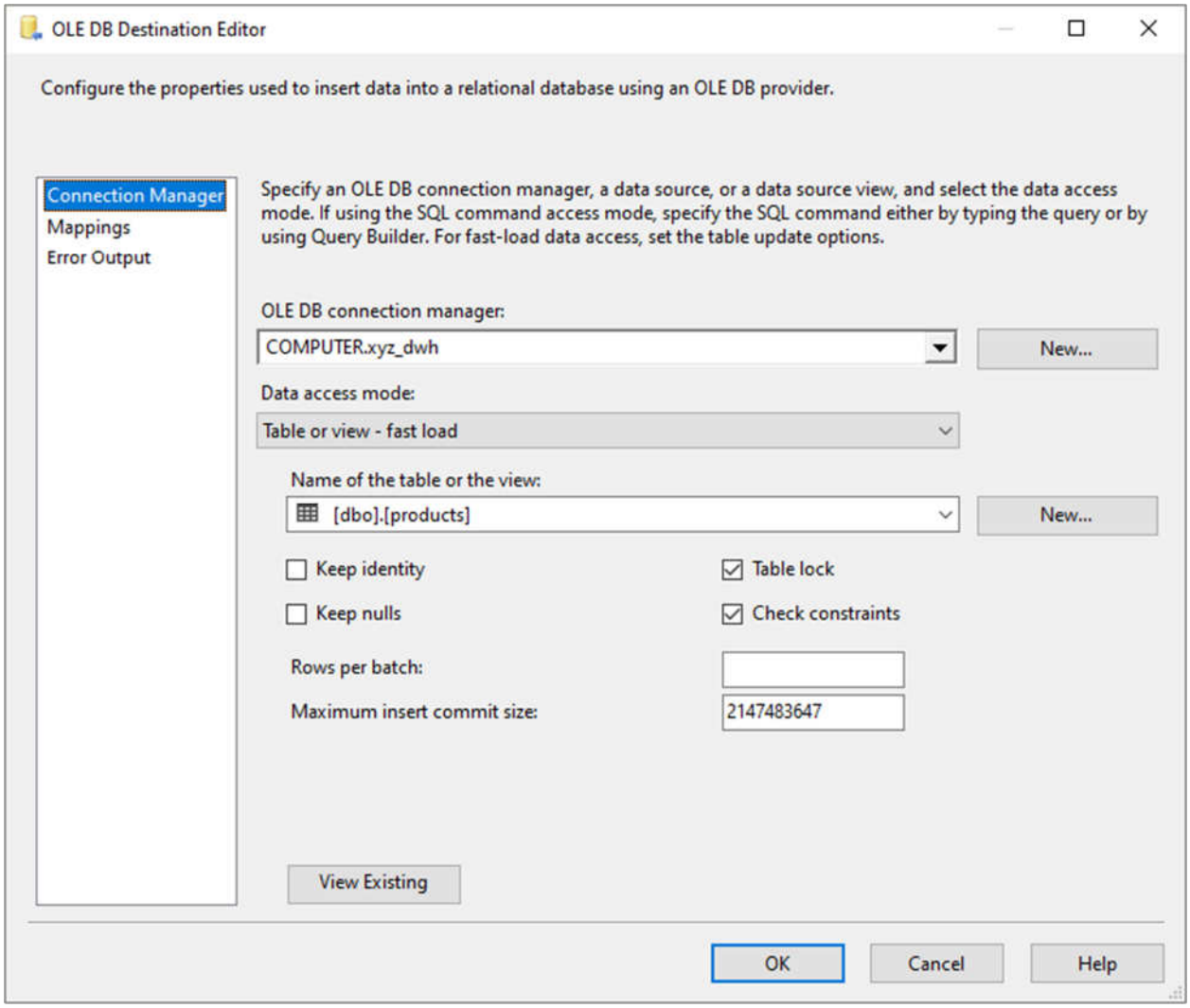


Figure 28: Configuring Destination products table.

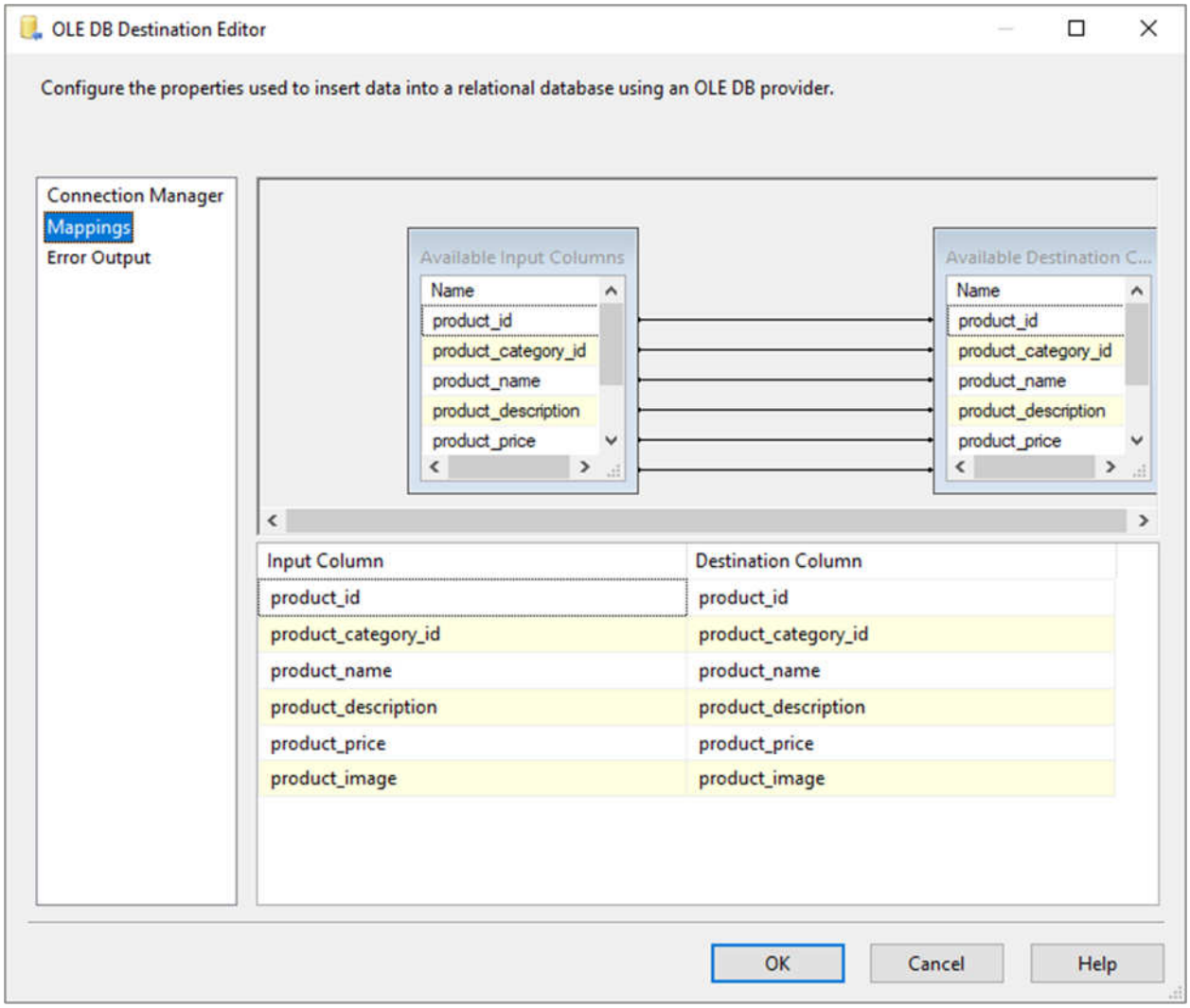


Figure 29: Configuring Destination and Source products table column.

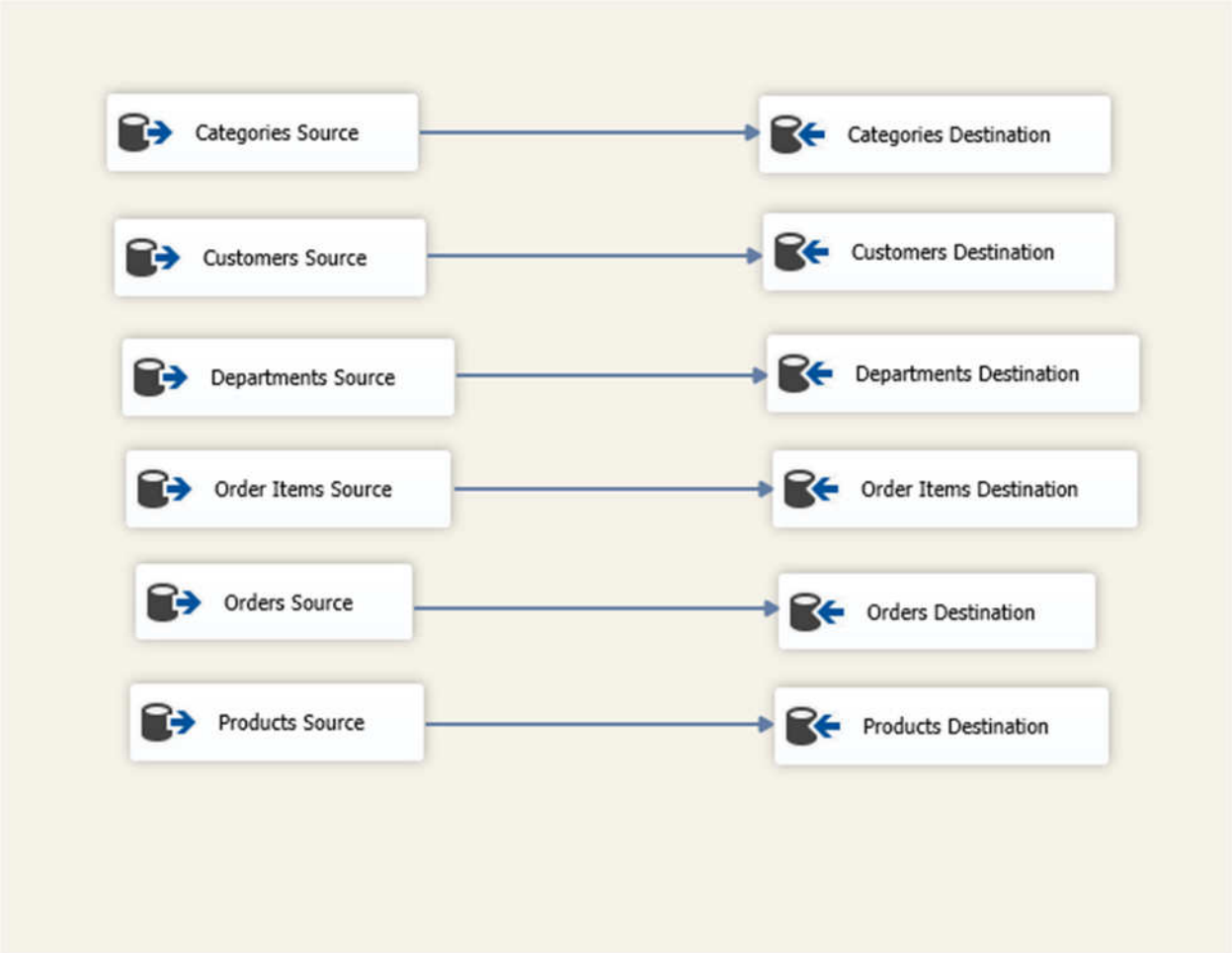


Figure 30: Final Data Flow Diagram

Develop truncate query to avoid duplicate key errors while executing the extraction package every Time.

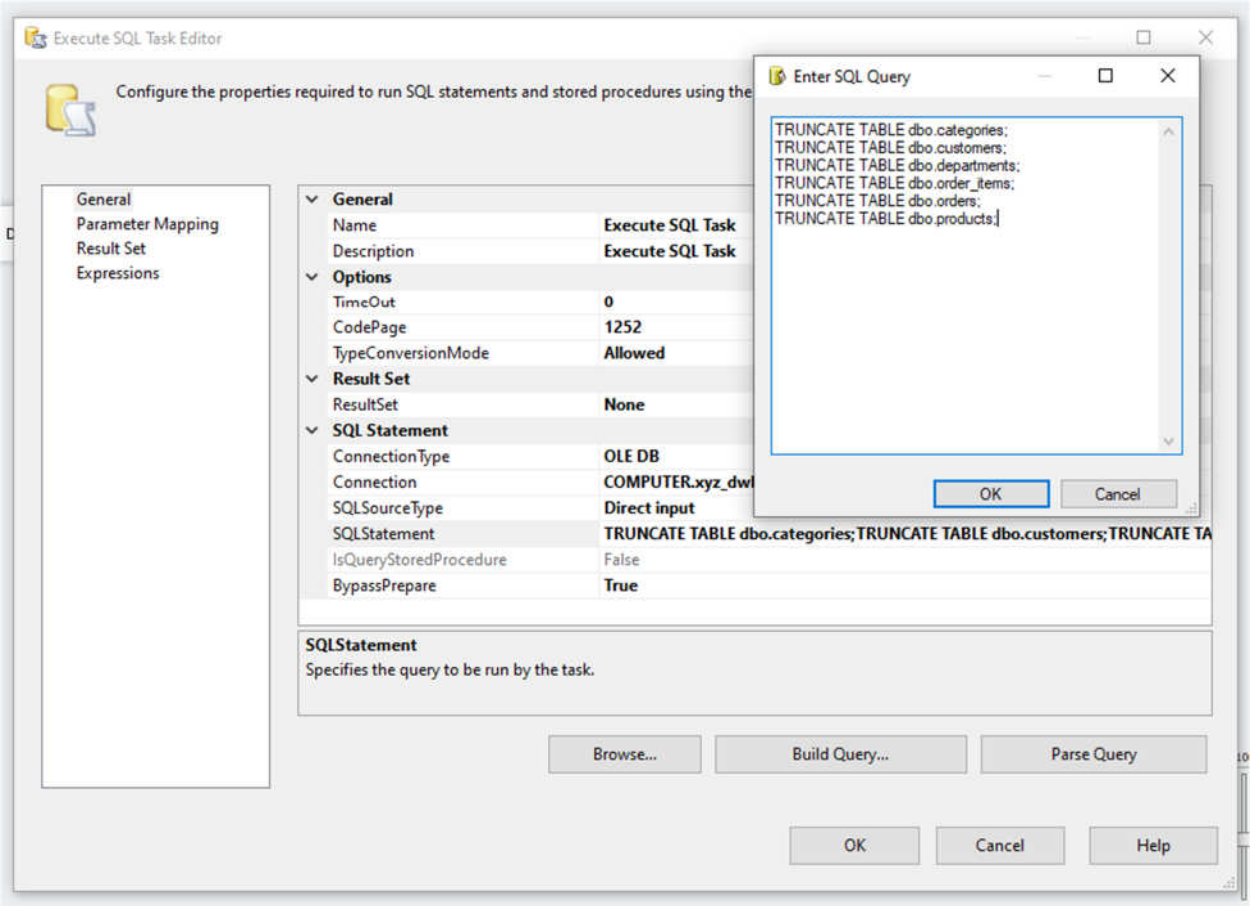


Figure 31: Setting up the Truncate SQL query in Execute SQL Task

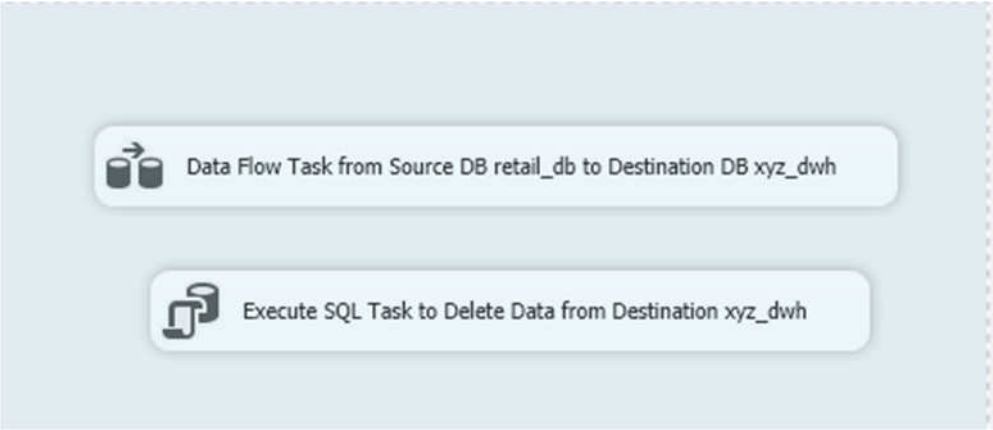


Figure 32: Renaming the Data Flow and Execute SQL query

1.1.4 Execution of the ETL Process

Execution of the ETL process is successful, and there are no errors on the source, destination, mapping, and truncate SQL query, and the data flow execution is successful from retail_db to xyz_dwh.

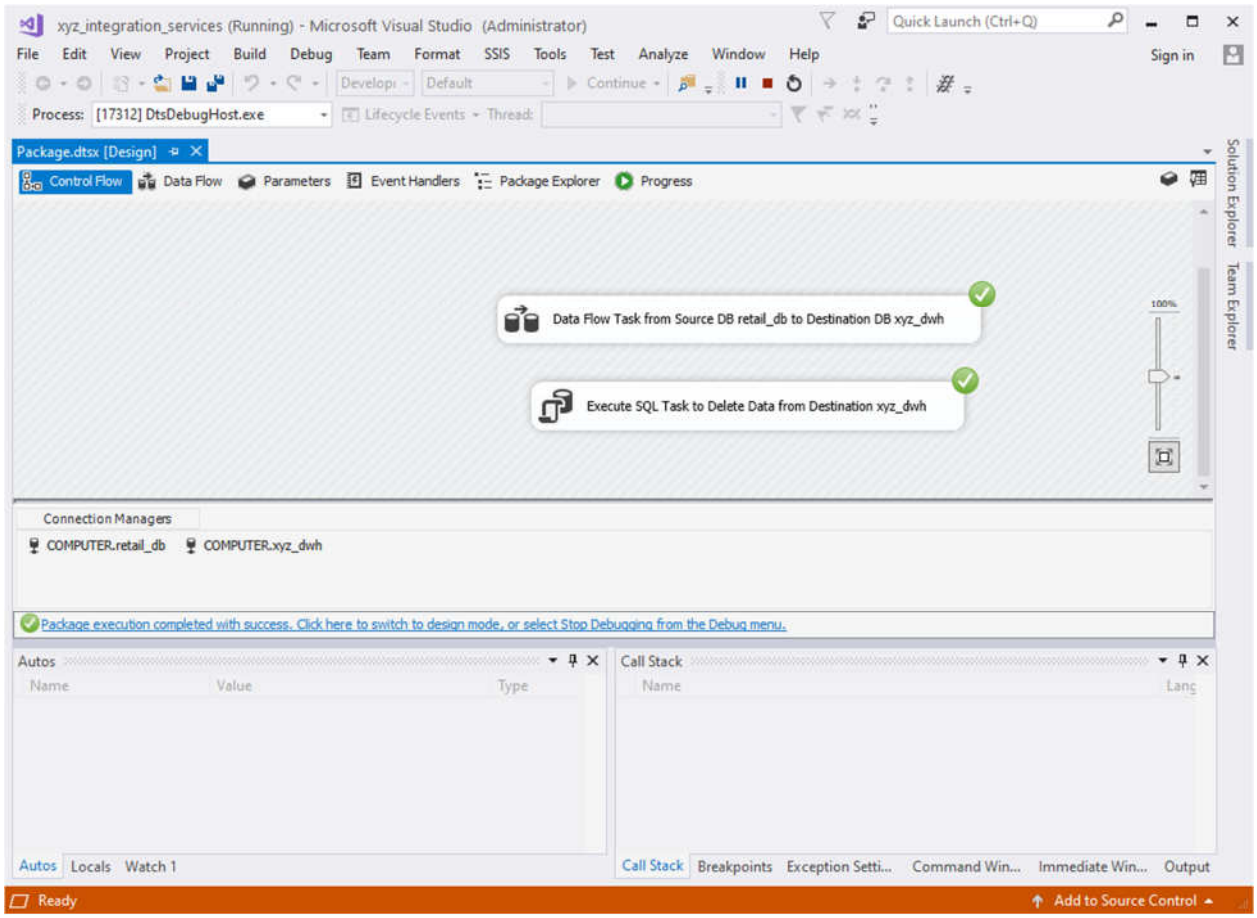


Figure 33: Execution of Control Flow

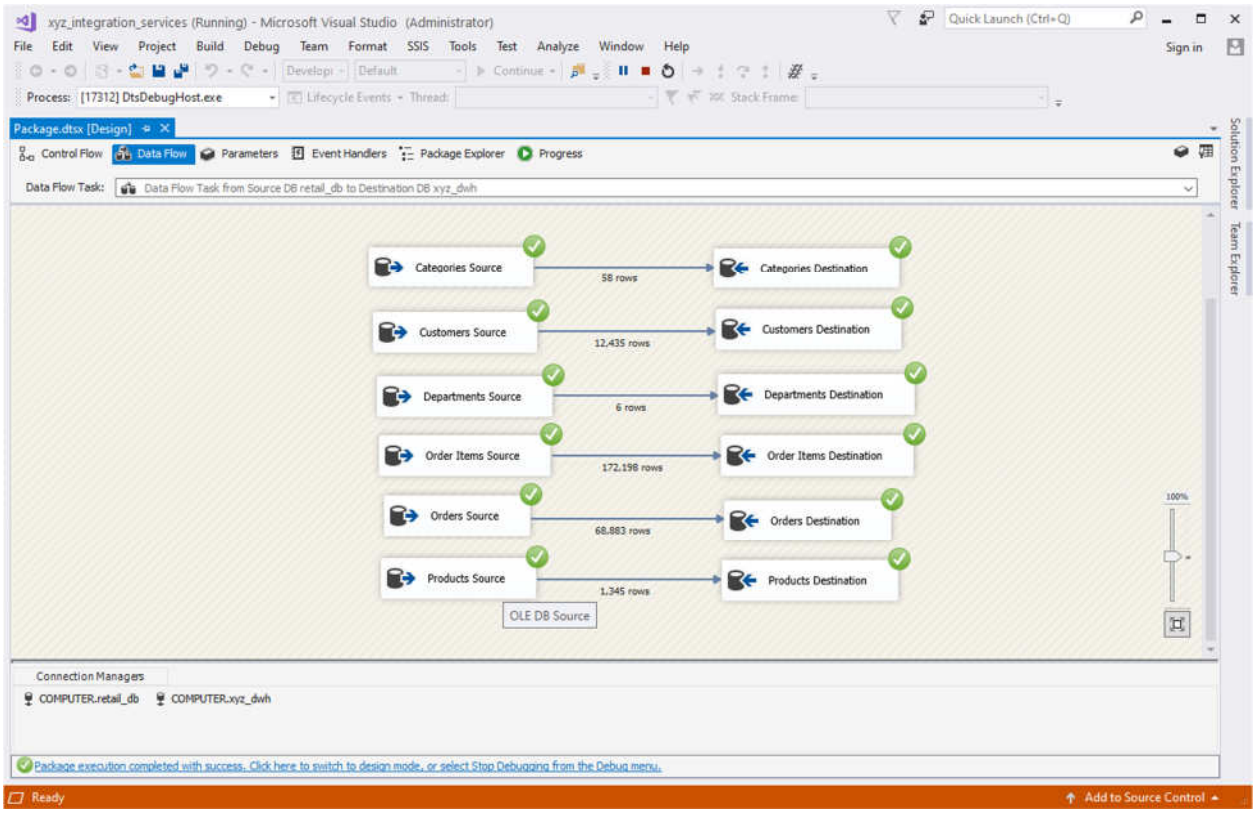


Figure 34: Successfully transferred the data of all the rows and columns

1.2 Dimensional Data Modelling (DM)

Dimensional Modeling is a data structure technique used to optimise data storage for faster retrieval. In this model of Data Warehouse, the design focuses on reading, summarising and analysing numeric information like counts, values, weights. This model consists of “Fact” and “Dimension” tables. (David, 2021)

1.2.1 Fact Table

The fact table is a primary table in dimension modelling, consisting of a foreign key (FK) to the dimension table. Facts are the measurements and metrics from the business process that consists of numbers. (David, 2021)

Fact tables in the given scenario:

KEY TYPE	ATTRIBUTE NAME
Primary Key (PK)	fact_sales_id
Foreign Key (FK)	product_key
	product_cost_price
	subtotal_product_cost_price
	subtotal_sold_price
	profit
	order_key
	order_items_key
	category_key
	department_key
	sales_date_key
	quantity
	the_month
	the_week
	the_day_of_week

Table 1: Fact Table fact_sales Table Structure

1.2.2 Dimension Table

Dimension table joins the fact table via a foreign key. Dimension provides the context surrounding a business process event. In simple terms, dimension gives a fact table who, where, and what. For example, in the Sales business process, for the fact quarterly sales number, dimensions would be customer names (who), location (where), product name (what). Dimension is a way to view the information in the facts. (David, 2021)

Dimension tables in the given scenario:

KEY TYPE	ATTRIBUTE NAME
Primary Key (PK)	category_key
Foreign Key (FK)	department_key
	category_name

Table 2: Dimension Table dimension_categories Table Structure

KEY TYPE	ATTRIBUTE NAME
Primary Key (PK)	DateKey
	TheDate
	TheDayOfWeek
	TheDayName
	WeekDayType
	DayNumberOfMonth
	DayNumberOfYear
	WeekNumberOfYear
	TheMonthName
	TheMonth
	QuarterNumberCalendar
	QuarterNameCalendar
	SemesterNumberCalendar
	SemesterNameCalendar
	YearCalendar
	MonthNumberFiscal
	QuarterNumberFiscal
	QuarterNameFiscal
	SemesterNumberFiscal
	SemesterNameFiscal
	YearFiscal

Table 3: Dimension Table dimension_date Table Structure

KEY TYPE	ATTRIBUTE NAME
Primary Key (PK)	depart_key
Foreign Key (FK)	depart_name

Table 4: Dimension Table dimension_departments Table Structure

KEY TYPE	ATTRIBUTE NAME
Primary Key (PK)	order_items_key
Foreign Key (FK)	order_key
Foreign Key (FK)	product_key
	subtotal_cost_price
	order_quantity

Table 5: Dimension Table dimension_order_items Table Structure

KEY TYPE	ATTRIBUTE NAME
Primary Key (PK)	order_key
	order_status
	order_date

Table 6: Dimension Table dimension_orders Table Structure

KEY TYPE	ATTRIBUTE NAME
Primary Key (PK)	product_key
Foreign Key (FK)	category_key
	product_name
	product_price

Table 7: Dimension Table dimension_products Table Structure

2. Analysis & Design of Dimensional Data Model

2.1 Grains

The grain communicates the level of detail associated with the fact table measurements. When identifying the grain, it also decide**s** the level of detail necessary to make available in the dimensional model. The more detail is included, the lower the level of granularity, and the less the detail is included, the higher the level of granularity. The qualitative data**,** in fact**,** and dimension table are bonded to the grain.

2.1.1 Grain of fact_sales Table Detail

The Fact Table (fact_sales) grain provides the quality of the sold products, the Time and date of the product order list, and the detailed product information like the department associated with it. The table also consists of cost price, sold price of each product, and the quantity sold, which will help determine the profit and the most profitable products list.

List of the attributes in fact table:

ATTRIBUTE NAME	DESCRIPTION
product_cost_price	The actual cost of each product
subtotal_product_cost_price	The total price of the products with the number of quantities sold
subtotal_sold_price	Details of the total price of the product sold
quantity	It helps to find the total order quantity of each product

Table 8: Attributes and Description in Fact Table

2.2 Dimensions of the Central Fact Table

Attributes List: including key type, the data type of dimension table and fact table.

2.2.1 Attribute List, Query and Value Results of Fact Table

Detail Attributes List of fact_sales table.

KEY TYPE	ATTRIBUTE NAME	DATA TYPE	DESCRIPTION
PK	fact_sales_id	INT	Unique identifier to distinguish each facts
FK	product_key	INT	Distinguish products sold
	product_cost_price	DEC	The actual cost price of the product
	subtotal_product_cost_price	DEC	Sold quantity multiplied by cost price
	subtotal_sold_price	DEC	20% increment on sold price for profit calculation
	profit	DEC	Profit from the actual sale after deducting the cost price of the product
FK	order_key	INT	Unique identifier to distinguish each order placed
FK	order_items_key	INT	The final status of the order, including payment details
FK	category_key	INT	Associate product with its category
FK	department_key	INT	Department info which handles the product category
	sales_date_key	DATETIME	Date and Time of each order placed
	quantity	INT	Total number of products sold
	the_month	INT	Month details of sales
	the_week	INT	Week details of sales
	the_day_of_week	INT	Day of the sales details in the week

Table 9: Detail Attributes List of fact_sales table

SQL query to create a fact table.

```
USE [xyz_dmart]

CREATE TABLE fact_sales (
       fact_sales_id int identity (1,1),
       product_key int NULL,
       product_cost_price decimal (25,2) NULL,
       subtotal_product_cost_price decimal (25,2) NULL,
    subtotal_sold_price decimal (25,2) NULL,
       profit decimal (25,2) NULL,
       order_key int NULL,
       order_items_key int NULL,
       category_key int NULL,
       department_key int NULL,
       sales_date_key int NULL,
       quantity int NULL,
       the_month int NULL,
       the_week int NULL,
       the_day_of_week int NULL
)
```

Figure 35: SQL query to create a fact table.

SQL query to populate values in the fact table.

```
INSERT INTO fact_sales(product_key, product_cost_price, subtotal_product_cost_price,
subtotal_sold_price, profit, order_key,
order_items_key, category_key, department_key, sales_date_key, quantity, the_month,
the_week, the_day_of_week)
SELECT
pro.product_key product_key,
sum(pro.product_price) product_cost_price,
sum(doi.subtotal_cost_price) subtotal_product_cost_price,
(sum(doi.subtotal_cost_price) + sum(doi.subtotal_cost_price)* 0.2) subtotal_sold_price,
((sum(doi.subtotal_cost_price) + sum(doi.subtotal_cost_price)*0.2) -
(sum(doi.subtotal_cost_price))) Profit,
doi.order_key order_key,
doi.order_items_key order_items_key,
pro.category_key category_key,
depart.depart_key department_key,
salesdate.DateKey sales_date_key,
doi.order_quantity quantity,
salesdate.TheMonth the_month,
salesdate.WeekNumberOfYear the_week,
salesdate.TheDayOfWeek the_day_of_week

FROM dbo.dimension_order_items doi
LEFT JOIN dbo.dimension_products pro ON pro.product_key = doi.product_key
LEFT JOIN dbo.dimension_categories cat ON pro.category_key = cat.category_key
LEFT JOIN dbo.dimension_departments depart ON cat.department_key = depart.depart_key
LEFT JOIN dbo.dimension_orders ord ON ord.order_key = doi.order_key
LEFT JOIN dbo.dimension_date salesdate ON ord.order_date = salesdate.TheDate

GROUP BY pro.product_key,
doi.order_key,
doi.order_items_key,
pro.category_key,
depart.depart_key,
salesdate.DateKey,
doi.order_quantity,
salesdate.TheMonth,
salesdate.WeekNumberOfYear,
salesdate.TheDayOfWeek
```

Figure 36: SQL query to insert values in the fact table.

	fact_sales_id	product_key	product_cost_price	subtotal_product_cost_price	subtotal_sold_price	profit	order_key	order_items_key	category_key	department_key	sales_date_key	quantity	the_month	the_week	the_day_of_week
1	1	1014	49.98	199.92	239.90	39.98	50066	125169	46	7	20140603	4	6	23	3
2	2	885	24.99	74.97	89.96	14.99	50116	125274	40	6	20140603	3	6	23	3
3	3	1073	199.99	199.99	239.99	40.00	17243	43129	48	7	20131109	1	11	45	7
4	4	403	129.99	129.99	155.99	26.00	19233	48056	18	4	20131122	1	11	47	6
5	5	502	50.00	50.00	60.00	10.00	19715	49294	24	5	20131124	1	11	48	1
6	6	1014	49.98	149.94	179.93	29.99	19804	49499	46	7	20131124	3	11	48	1
7	7	825	31.99	159.95	191.94	31.99	20716	51761	37	6	20131130	5	11	48	7
8	8	1014	49.98	49.98	59.98	10.00	21838	54607	46	7	20131206	1	12	49	6
9	9	792	14.99	59.96	71.95	11.99	22311	55858	36	6	20131208	4	12	50	1
10	10	1073	199.99	199.99	239.99	40.00	48198	120531	48	7	20140520	1	5	21	3
11	11	1004	399.98	399.98	479.98	80.00	48461	121195	45	7	20140521	1	5	21	4
12	12	957	299.98	299.98	359.98	60.00	63264	158167	43	7	20140219	1	2	8	4
13	13	365	59.99	239.96	287.95	47.99	65711	164216	17	4	20140523	4	5	21	6
14	14	502	50.00	100.00	120.00	20.00	44744	111757	24	5	20140427	2	4	18	1
15	15	365	59.99	59.99	71.99	12.00	11808	29527	17	4	20131005	1	10	40	7

Figure 37: Top 15 Values of the fact table

2.2.2 Attribute List, Query and Value Results of Dimensions of Fact Table

2.2.2.1 dimension_categories Table

KEY TYPE	ATTRIBUTE NAME	DATA TYPE	DESCRIPTION
PK	category_key	INT	Unique identifier for each category
FK	department_key	INT	Department based product categorisation
	category_name	VARCHAR	Know the name of a category

Table 10: dimension_categories Table details with Attributes

```
CREATE TABLE dbo.dimension_categories (
       [category_key] int NOT NULL,
       [department_key] int NOT NULL,
       [category_name] VARCHAR (100) NOT NULL
       );

INSERT INTO dimension_categories(category_key, department_key, category_name)
SELECT category_id, category_department_id as departmentKey, category_name  FROM
xyz_dwh.dbo.categories;
```

Figure 38: SQL Query to Create Table and Insert data on dimension_categories Table

	category_key	department_key	category_name
1	1	2	Football
2	2	2	Soccer
3	3	2	Baseball & Softball
4	4	2	Basketball
5	5	2	Lacrosse
6	6	2	Tennis & Racquet
7	7	2	Hockey
8	8	2	More Sports
9	9	3	Cardio Equipment
10	10	3	Strength Training
11	11	3	Fitness Accessories
12	12	3	Boxing & MMA
13	13	3	Electronics
14	14	3	Yoga & Pilates
15	15	3	Training by Sport
16	16	3	As Seen on TV!
17	17	4	Cleats

Figure 39: Top 17 Values of dimension_categories Table

2.2.2.2 dimension_date Table

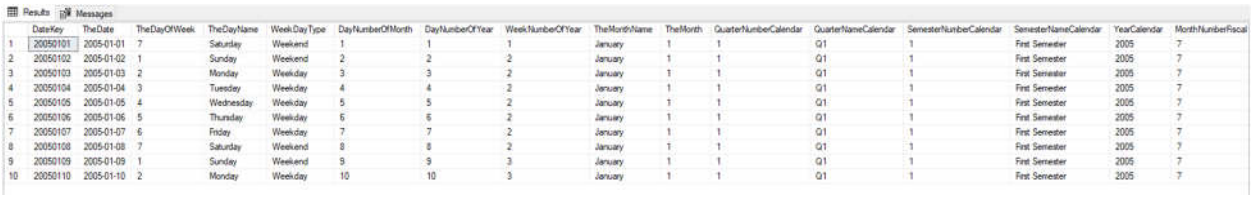
KEY TYPE	ATTRIBUTE NAME	DATA TYPE	DESCRIPTION
PK	DateKey	INT	The unique key associated with each date to reference the fact table
	TheDate	DATE	The date from 2010-01-01 to 2039-12-31
	TheDayOfWeek	TINYINT	Days number associated with the date
	TheDayName	NVARCHAR	Name of the day
	TheMonthName	NVARCHAR	Name of the month
	TheMonth	TINYINT	Month number associated with the date
	QuarterNumberCalendar	TINYINT	Quarter relation with the date

Table 11: dimension_date Table details with Attributes

```
CREATE TABLE dbo.dimension_date
(
    DateKey                 int not null,
    TheDate                 date not null,
    TheDayOfWeek            tinyint not null,
    TheDayName              nvarchar(10) not null,
    WeekDayType             nvarchar(7) not null,
    DayNumberOfMonth        tinyint not null,
    DayNumberOfYear         smallint not null,
    WeekNumberOfYear        tinyint not null,
    TheMonthName            nvarchar(10) not null,
    TheMonth                tinyint not null,
    QuarterNumberCalendar   tinyint not null,
    QuarterNameCalendar     nchar(2) not null,
    SemesterNumberCalendar  tinyint not null,
    SemesterNameCalendar    nvarchar(15) not null,
    YearCalendar            smallint not null,
    MonthNumberFiscal       tinyint not null,
    QuarterNumberFiscal     tinyint not null,
    QuarterNameFiscal       nchar(2) not null,
    SemesterNumberFiscal    tinyint not null,
    SemesterNameFiscal      nvarchar(15) not null,
    YearFiscal              smallint not null

    constraint PK_dimension_date primary key clustered
    (
        DateKey asc
    )
)
go

declare @DateCalendarStart  datetime,
        @DateCalendarEnd    datetime,
        @FiscalCounter      datetime,
        @FiscalMonthOffset  int;

set @DateCalendarStart = '2005-01-01';
set @DateCalendarEnd = '2030-12-31';

-- Set this to the number of months to add or extract to the current date to get the
beginning
-- of the Fiscal Year. Example: If the Fiscal Year begins July 1, assign the value of 6
-- to the @FiscalMonthOffset variable. Negative values are also allowed thus if your
-- 2012 Fiscal Year begins in July of 2011, assign a value of -6.
set @FiscalMonthOffset = 6;

with DateDimension
as
(
    select  @DateCalendarStart as DateCalendarValue,
            dateadd(m, @FiscalMonthOffset, @DateCalendarStart) as FiscalCounter

    union all

    select  DateCalendarValue + 1,
            dateadd(m, @FiscalMonthOffset, (DateCalendarValue + 1)) as FiscalCounter
    from    DateDimension
    where   DateCalendarValue + 1 < = @DateCalendarEnd )
```

```
insert into dbo.dimension_date (DateKey, TheDate, TheDayOfWeek, TheDayName, WeekDayType,
DayNumberOfMonth, DayNumberOfYear, WeekNumberOfYear, TheMonthName, TheMonth,
QuarterNumberCalendar, QuarterNameCalendar, SemesterNumberCalendar, SemesterNameCalendar,
YearCalendar, MonthNumberFiscal, QuarterNumberFiscal, QuarterNameFiscal,
SemesterNumberFiscal, SemesterNameFiscal, YearFiscal)

select  cast(convert(varchar(25), DateCalendarValue, 112) as int) as 'DateKey',
        cast(DateCalendarValue as date) as 'TheDate',
        datepart(weekday, DateCalendarValue) as 'TheDayOfWeek',
        datename(weekday, DateCalendarValue) as 'TheDayName',
        case datename(dw, DateCalendarValue)
            when 'Saturday' then 'Weekend'
            when 'Sunday' then 'Weekend'
        else 'Weekday'
        end as 'WeekDayType',
        datepart(day, DateCalendarValue) as'DayNumberOfMonth',
        datepart(dayofyear, DateCalendarValue) as 'DayNumberOfYear',
        datepart(week, DateCalendarValue) as 'WeekNumberOfYear',
        datename(month, DateCalendarValue) as 'TheMonthName',
        datepart(month, DateCalendarValue) as 'TheMonth',
        datepart(quarter, DateCalendarValue) as 'QuarterNumberCalendar',
        'Q' + cast(datepart(quarter, DateCalendarValue) as nvarchar) as
'QuarterNameCalendar',
        case
            when datepart(month, DateCalendarValue) <= 6 then 1
            when datepart(month, DateCalendarValue) > 6 then 2
        end as 'SemesterNumberCalendar',
        case
            when datepart(month, DateCalendarValue) < = 6 then 'First Semester'
            when datepart(month, DateCalendarValue) > 6 then 'Second Semester'
        end as 'SemesterNameCalendar',
        datepart(year, DateCalendarValue) as 'YearCalendar',
        datepart(month, FiscalCounter) as 'MonthNumberFiscal',
        datepart(quarter, FiscalCounter) as 'QuarterNumberFiscal',
        'Q' + cast(datepart(quarter, FiscalCounter) as nvarchar) as 'QuarterNameFiscal',
        case
            when datepart(month, FiscalCounter) < = 6 then 1
            when datepart(month, FiscalCounter) > 6 then 2
        end as 'SemesterNumberFiscal',
        case
            when datepart(month, FiscalCounter) < = 6 then 'First Semester'
            when  datepart(month, FiscalCounter) > 6 then 'Second Semester'
        end as 'SemesterNameFiscal',
        datepart(year, FiscalCounter) as 'YearFiscal'
from    DateDimension
order by
        DateCalendarValue
option (maxrecursion 0);
```

Figure 40: SQL Query to Create Table and Insert data on dimension_date Table



Results Messages

	DateKey	TheDate	TheDayOfWeek	TheDayName	WeekDayType	DayNumberOfMonth	DayNumberOfYear	WeekNumberOfYear	TheMonthName	TheMonth	QuarterNumberCalendar	QuarterNameCalendar	SemesterNumberCalendar	SemesterNameCalendar	YearCalendar	MonthNumberFiscal
1	20050101	2005-01-01	7	Saturday	Weekend	1	1	1	January	1	1	Q1	1	First Semester	2005	7
2	20050102	2005-01-02	1	Sunday	Weekend	2	2	2	January	1	1	Q1	1	First Semester	2005	7
3	20050103	2005-01-03	2	Monday	Weekday	3	3	2	January	1	1	Q1	1	First Semester	2005	7
4	20050104	2005-01-04	3	Tuesday	Weekday	4	4	2	January	1	1	Q1	1	First Semester	2005	7
5	20050105	2005-01-05	4	Wednesday	Weekday	5	5	2	January	1	1	Q1	1	First Semester	2005	7
6	20050106	2005-01-06	5	Thursday	Weekday	6	6	2	January	1	1	Q1	1	First Semester	2005	7
7	20050107	2005-01-07	6	Friday	Weekday	7	7	2	January	1	1	Q1	1	First Semester	2005	7
8	20050108	2005-01-08	7	Saturday	Weekend	8	8	2	January	1	1	Q1	1	First Semester	2005	7
9	20050109	2005-01-09	1	Sunday	Weekend	9	9	3	January	1	1	Q1	1	First Semester	2005	7
10	20050110	2005-01-10	2	Monday	Weekday	10	10	3	January	1	1	Q1	1	First Semester	2005	7

Figure 41: Top 10 Values of dimension_date Table

2.2.2.3 dimension_departments Table

KEY TYPE	ATTRIBUTE NAME	DATA TYPE	DESCRIPTION
PK	depart_key	INT	Unique identifier for each department
FK	depart_name	NVARCHAR	Know the department name

Table 12: dimension_departments Table Details with Attributes

```
CREATE TABLE dimension_departments (
       [depart_key] int NOT NULL,
       [depart_name] nvarchar (45) NOT NULL
       );

INSERT INTO dimension_departments(depart_key, depart_name)
SELECT department_id, department_name FROM xyz_dwh.dbo.departments;
```

Figure 42: SQL Query to Create Table and Insert Data on dimesion_departments Table

Results Messages

	depart_key	depart_name
1	2	Fitness
2	3	Footwear
3	4	Apparel
4	5	Golf
5	6	Outdoors
6	7	Fan Shop

Figure 43: All Values of dimension_ departments Table

2.2.2.4 dimension_order_items Table

KEY TYPE	ATTRIBUTE NAME	DATA TYPE	DESCRIPTION
PK	order_items_key	INT	Unique key to differentiate each order items with quantity
FK	order_key	INT	Differentiate order as per the order made
FK	product_key	INT	Differentiate product as per the sold products
	subtotal_cost_price	DECIMAL	The actual price of a product with the total quantity sold
	order_quantity	SMALLINT	The total quantity of the sold products

Table 13: dimension_order_items Table details with Attributes

```
CREATE TABLE dimension_order_items (
       [order_items_key] int NOT NULL,
       [order_key] int NOT NULL,
       [product_key] int NOT NULL,
       [subtotal_cost_price] decimal (25,2)  NOT NULL,
       [order_quantity] smallint NOT NULL
       );

INSERT INTO dimension_order_items(order_items_key, order_key, product_key,
subtotal_cost_price, order_quantity)
SELECT order_item_id, order_item_order_id, order_item_product_id, order_item_subtotal,
order_item_quantity FROM xyz_dwh.dbo.order_items;
```

Figure 44: SQL Query to Create Table and Insert Data on dimension_order_items Table

Results | Messages

	order_items_key	order_key	product_key	subtotal_cost_price	order_quantity
1	1	1	957	299.98	1
2	2	2	1073	199.99	1
3	3	2	502	250.00	5
4	4	2	403	129.99	1
5	5	4	897	49.98	2
6	6	4	365	299.95	5
7	7	4	502	150.00	3
8	8	4	1014	199.92	4
9	9	5	957	299.98	1
10	10	5	365	299.95	5
11	11	5	1014	99.96	2
12	12	5	957	299.98	1
13	13	5	403	129.99	1
14	14	7	1073	199.99	1
15	15	7	957	299.98	1
16	16	7	926	79.95	5
17	17	8	365	179.97	3
18	18	8	365	299.95	5
19	19	8	1014	199.92	4
20	20	8	502	50.00	1

Figure 45: Top 20 Value of the dimension_order_items Table

2.2.2.5 dimension_orders Table

KEY TYPE	ATTRIBUTE NAME	DATA TYPE	DESCRIPTION
PK	order_key	INT	Unique key to differentiate each customer order
	order_status	NVARCHAR	Order status of each customer order
	order_date	DATETIME	Time and Date info of each customer order

Table 14: dimension_orders Table details with Attributes

```
CREATE TABLE dimension_orders (
       [order_key] int NOT NULL,
       [order_status] nvarchar (45) NOT NULL,
       [order_date] DATETIME2
       );

INSERT INTO dimension_orders(order_key, order_status, order_date)
SELECT order_id, order_status, order_date FROM xyz_dwh.dbo.orders;
```

Figure 46: SQL Query to Create Table and Insert Data on dimension_orders table

Results | Messages

	order_key	order_status	order_date
1	1	CLOSED	2013-07-25 00:00:00.0000000
2	2	PENDING_PAYMENT	2013-07-25 00:00:00.0000000
3	3	COMPLETE	2013-07-25 00:00:00.0000000
4	4	CLOSED	2013-07-25 00:00:00.0000000
5	5	COMPLETE	2013-07-25 00:00:00.0000000
6	6	COMPLETE	2013-07-25 00:00:00.0000000
7	7	COMPLETE	2013-07-25 00:00:00.0000000
8	8	PROCESSING	2013-07-25 00:00:00.0000000
9	9	PENDING_PAYMENT	2013-07-25 00:00:00.0000000
10	10	PENDING_PAYMENT	2013-07-25 00:00:00.0000000
11	11	PAYMENT_REVIEW	2013-07-25 00:00:00.0000000
12	12	CLOSED	2013-07-25 00:00:00.0000000
13	13	PENDING_PAYMENT	2013-07-25 00:00:00.0000000
14	14	PROCESSING	2013-07-25 00:00:00.0000000
15	15	COMPLETE	2013-07-25 00:00:00.0000000

Figure 47: Top 15 Values of dimension_orders Table

2.2.2.6 dimension_products Table

KEY TYPE	ATTRIBUTE NAME	DATA TYPE	DESCRIPTION
PK	product_key	INT	
FK	category_key	INT	
	product_name	VARCHAR	
	product_price	DECIMAL	

Table 15: dimension_products Table details with Attributes

```
CREATE TABLE dimension_products (
       [product_key] int NOT NULL,
       [category_key] int NOT NULL,
       [product_name] VARCHAR(200)  NOT NULL,
       [product_price] decimal(25,2) NOT NULL,
       );

INSERT INTO dimension_products(product_key, category_key, product_name, product_price)
SELECT product_id, product_category_id, product_name, product_price FROM
xyz_dwh.dbo.products;
```

Figure 48: SQL Query to Create Table and Insert Data on dimension_products Table

Results | Messages

	product_key	category_key	product_name	product_price
1	1	2	Quest Q64 10 FT. x 10 FT. Slant Leg Instant U	59.98
2	2	2	Under Armour Men's Highlight MC Football Clea	129.99
3	3	2	Under Armour Men's Renegade D Mid Football Cl	89.99
4	4	2	Under Armour Men's Renegade D Mid Football Cl	89.99
5	5	2	Riddell Youth Revolution Speed Custom Footbal	199.99
6	6	2	Jordan Men's VI Retro TD Football Cleat	134.99
7	7	2	Schutt Youth Recruit Hybrid Custom Football H	99.99
8	8	2	Nike Men's Vapor Carbon Elite TD Football Cle	129.99
9	9	2	Nike Adult Vapor Jet 3.0 Receiver Gloves	50.00
10	10	2	Under Armour Men's Highlight MC Football Clea	129.99
11	11	2	Fitness Gear 300 lb Olympic Weight Set	209.99
12	12	2	Under Armour Men's Highlight MC Alter Ego Fla	139.99
13	13	2	Under Armour Men's Renegade D Mid Football Cl	89.99
14	14	2	Quik Shade Summit SX170 10 FT. x 10 FT. Canop	199.99
15	15	2	Under Armour Kids' Highlight RM Alter Ego Sup	59.99

Figure 49: Top 15 Values of dimension_products Table

2.3 Structure of Model Table as Fact Tables in Dimension

2.3.1 Star Schema

Star Schema is a mature data modelling approach in relational data warehouses. It requires modellers to classify the model tables either as *dimension* or *fact.* All the facts tables are included by indexing the number of dimensional tables in the star schema. (Peter, 2021)

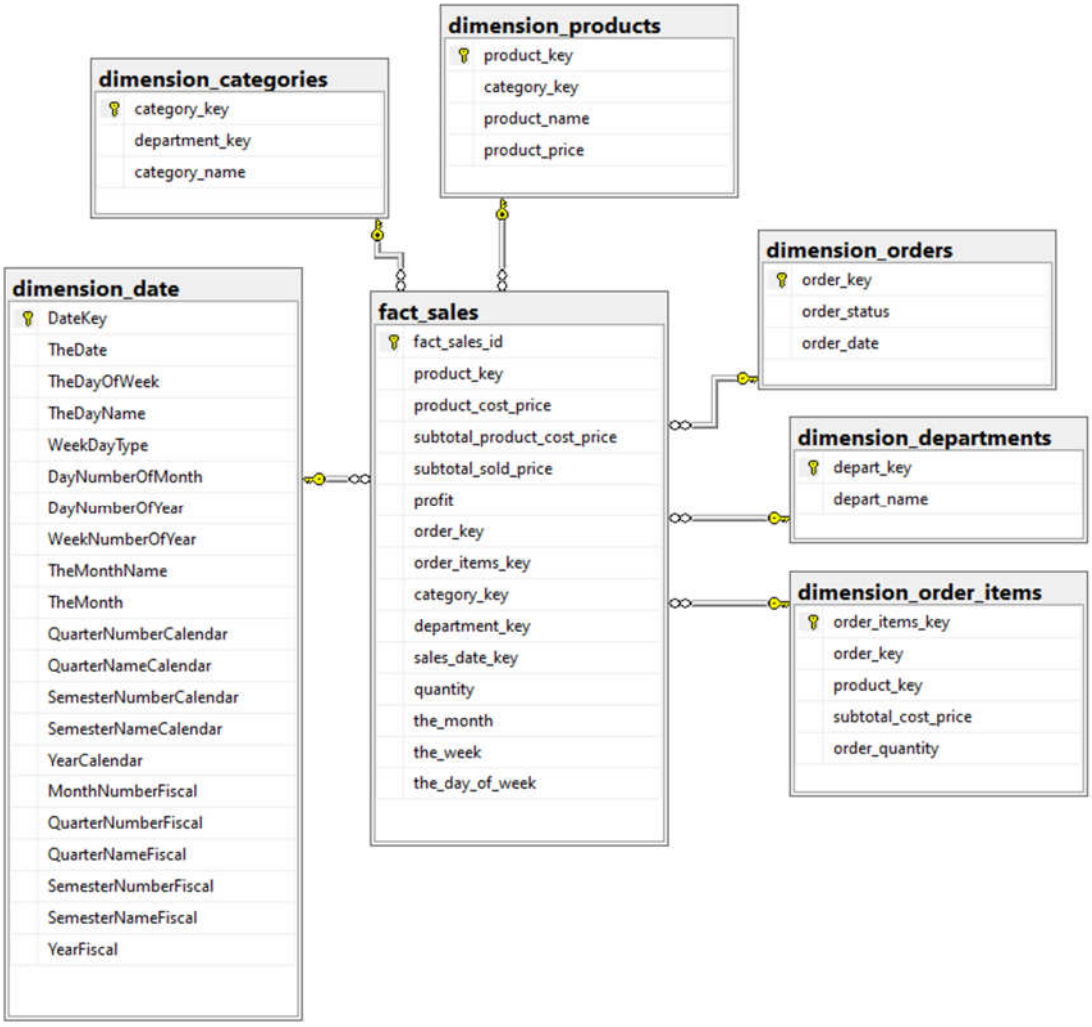


Figure 50: Star Schema modelling to define fact and dimension tables

2.3.2 Business Reports

Business reports are meaningful information that helps to drive profitable business actions. Business reports directly impact the organisation's strategic, tactical, and operational decisions based on facts instead of assumptions and gut feelings. (Taylor, 2021)

2.3.2.1 Business Reports for XYZ Retail Store

2.3.2.2 Monthly/Weekly Report of all Products as per Departments

Monthly Report

The below query results show the Monthly total sales along with the product name and the department in November month and department seven is the most selling month and most selling department.

```
USE [xyz_dmart]

SELECT
       MONTH(dat.TheDate) TheMonth,
       SUM(fas.subtotal_sold_price) total_sales_per_month,
       pro.product_name productName,
       fas.department_key departmentKey
FROM dbo.dimension_date dat
INNER JOIN dbo.fact_sales fas
ON fas.sales_date_key = dat.DateKey
INNER JOIN dbo.dimension_products pro
ON pro.product_key = fas.product_key
GROUP BY MONTH(dat.TheDate), fas.department_key, pro.product_name
ORDER BY total_sales_per_month DESC;
```

Figure 51: Monthly Query

Results | Messages

	TheMonth	total_sales_per_month	productName	departmentKey
1	11	757888.42	Field & Stream Sportsman 16 Gun Fire Safe	7
2	12	742529.06	Field & Stream Sportsman 16 Gun Fire Safe	7
3	1	739169.20	Field & Stream Sportsman 16 Gun Fire Safe	7
4	7	709890.42	Field & Stream Sportsman 16 Gun Fire Safe	7
5	9	699810.84	Field & Stream Sportsman 16 Gun Fire Safe	7
6	3	696930.96	Field & Stream Sportsman 16 Gun Fire Safe	7
7	4	685411.44	Field & Stream Sportsman 16 Gun Fire Safe	7
8	8	674851.88	Field & Stream Sportsman 16 Gun Fire Safe	7
9	5	666212.24	Field & Stream Sportsman 16 Gun Fire Safe	7
10	2	664292.32	Field & Stream Sportsman 16 Gun Fire Safe	7
11	6	648932.96	Field & Stream Sportsman 16 Gun Fire Safe	7
12	10	629733.76	Field & Stream Sportsman 16 Gun Fire Safe	7
13	11	485558.96	Perfect Fitness Perfect Rip Deck	4

Figure 52: Monthly Result

Weekly Report

The below query results show the total weekly sales, the product name and the department. The 45th month is generating maximum revenue compared to other months.

```
SELECT
       fas.the_week WeekD,
       SUM(fas.subtotal_sold_price) total_sales_per_month,
       pro.product_name product_name,
       department_key department_key
FROM dbo.dimension_date dat
INNER JOIN dbo.fact_sales fas
ON fas.sales_date_key = dat.DateKey
INNER JOIN dbo.dimension_products pro
ON pro.product_key = fas.product_key
GROUP BY fas.the_week, department_key, pro.product_name
ORDER BY total_sales_per_month DESC;
```

Figure 53: Weekly Query

Results | Messages

	WeekD	total_sales_per_month	product_name	department_key
1	45	233750.26	Field & Stream Sportsman 16 Gun Fire Safe	7
2	14	201111.62	Field & Stream Sportsman 16 Gun Fire Safe	7
3	31	200151.66	Field & Stream Sportsman 16 Gun Fire Safe	7
4	30	196311.82	Field & Stream Sportsman 16 Gun Fire Safe	7
5	39	195351.86	Field & Stream Sportsman 16 Gun Fire Safe	7
6	49	191992.00	Field & Stream Sportsman 16 Gun Fire Safe	7
7	2	183832.34	Field & Stream Sportsman 16 Gun Fire Safe	7
8	20	183832.34	Field & Stream Sportsman 16 Gun Fire Safe	7
9	10	178552.56	Field & Stream Sportsman 16 Gun Fire Safe	7
10	48	178072.58	Field & Stream Sportsman 16 Gun Fire Safe	7
11	5	176152.66	Field & Stream Sportsman 16 Gun Fire Safe	7
12	24	175672.68	Field & Stream Sportsman 16 Gun Fire Safe	7
13	52	175192.70	Field & Stream Sportsman 16 Gun Fire Safe	7

Figure 54: Weekly Query Result

2.3.2.3 Profit Analysis for Different Departments

The “Fan Shop” department has been generating more profit than other departments based on the analysis.

```
USE [xyz_dmart]

SELECT
       SUM(fas.Profit) total_profit,
       depart.depart_name department_name,
       fas.department_key
FROM dbo.fact_sales fas
INNER JOIN dbo.dimension_departments depart
ON depart.depart_key = fas.department_key
GROUP BY depart.depart_name, fas.department_key
ORDER BY Total_profit DESC;
```

Figure 55: Profit Analysis Query based on Departments

Results | Messages

	total_profit	department_name	department_key
1	3421708.79	Fan Shop	7
2	1464784.27	Apparel	4
3	921806.16	Golf	5
4	801299.47	Footwear	3
5	199118.15	Outdoors	6
6	56009.26	Fitness	2

Figure 56: Profit Result based on Departments

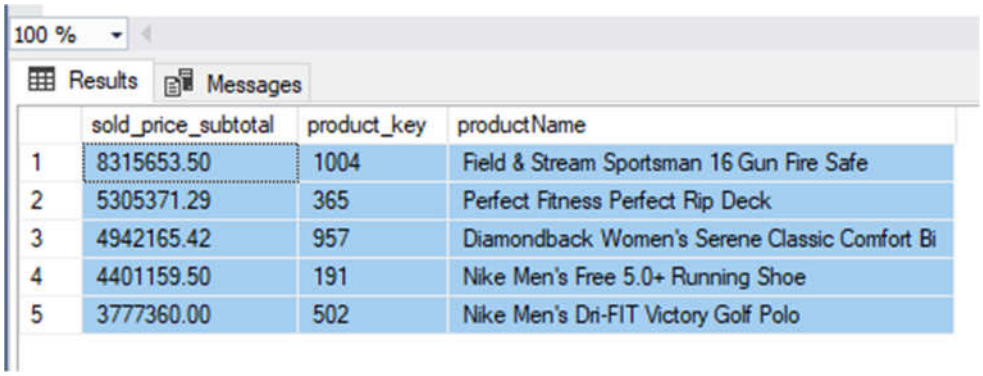
2.3.2.4 Top Revenue Generating Items

Top Five Revenue Generating Product

Based on the revenue analysis, the top five revenue-generating items are listed below, and the most selling item is “Field & Stream Sportsman 16 Gun Fire Safe”.

```
USE [xyz_dmart]

SELECT TOP 5
       SUM(fas.subtotal_sold_price) as sold_price_subtotal,
       fas.product_key,
       pro.product_name productName
FROM dbo.fact_sales fas
INNER JOIN dbo.dimension_products pro ON pro.product_key = fas.product_key
GROUP BY fas.product_key, pro.product_name
ORDER BY sold_price_subtotal DESC;
```

Figure 57: Query to analyse the top revenue-generating product.



	sold_price_subtotal	product_key	productName
1	8315653.50	1004	Field & Stream Sportsman 16 Gun Fire Safe
2	5305371.29	365	Perfect Fitness Perfect Rip Deck
3	4942165.42	957	Diamondback Women's Serene Classic Comfort Bi
4	4401159.50	191	Nike Men's Free 5.0+ Running Shoe
5	3777360.00	502	Nike Men's Dri-FIT Victory Golf Polo

Figure 58: List of top five revenue-generating items.

2.3.2.5 Reasons for Rejecting/Cancelled Orders

```
USE [xyz_dmart]

SELECT
	MONTH(dat.TheDate) TheMonth,
	SUM(subtotal_sold_price) subtotal_sold_price,
	fas.order_key, fas.category_key, pro.product_name AS product_name,
ord.order_status
FROM dbo.fact_sales fas
INNER JOIN dbo.dimension_orders ord ON ord.order_key = fas.order_key
INNER JOIN dbo.dimension_products pro ON pro.product_key = fas.product_key
INNER JOIN dbo.dimension_date dat ON dat.DateKey = fas.sales_date_key
GROUP BY MONTH(dat.TheDate), fas.order_key, fas.category_key, ord.order_status,
pro.product_name
HAVING order_status = 'CANCELED'
ORDER BY fas.category_key desc;
```

Figure 59: Query to find out the Canceled Orders

Results | Messages

	TheMonth	subtotal_sold_price	order_key	category_key	product_name	order_status
1	1	239.99	26105	48	Pelican Sunstream 100 Kayak	CANCELED
2	1	239.99	28091	48	Pelican Sunstream 100 Kayak	CANCELED
3	1	239.99	28182	48	Pelican Sunstream 100 Kayak	CANCELED
4	1	239.99	28596	48	Pelican Sunstream 100 Kayak	CANCELED
5	1	239.99	28925	48	Pelican Sunstream 100 Kayak	CANCELED
6	1	239.99	29112	48	Pelican Sunstream 100 Kayak	CANCELED
7	1	479.98	29452	48	Pelican Sunstream 100 Kayak	CANCELED
8	1	239.99	29497	48	Pelican Sunstream 100 Kayak	CANCELED
9	1	239.99	29774	48	Pelican Sunstream 100 Kayak	CANCELED
10	1	239.99	29817	48	Pelican Sunstream 100 Kayak	CANCELED
11	1	239.99	30110	48	Pelican Sunstream 100 Kayak	CANCELED
12	1	239.99	30182	48	Pelican Sunstream 100 Kayak	CANCELED
13	1	239.99	30715	48	Pelican Sunstream 100 Kayak	CANCELED
14	1	239.99	62047	48	Pelican Sunstream 100 Kayak	CANCELED
15	1	239.99	62509	48	Pelican Sunstream 100 Kayak	CANCELED
16	1	479.98	62546	48	Pelican Sunstream 100 Kayak	CANCELED
17	1	239.99	62610	48	Pelican Sunstream 100 Kayak	CANCELED
18	1	239.99	62680	48	Pelican Sunstream 100 Kayak	CANCELED
19	2	239.99	31207	48	Pelican Sunstream 100 Kayak	CANCELED

Figure 60: List of Canceled Orders

2.3.2.6 Daily and Monthly Analysis of Top-selling Products

Analysis of Top Selling Item based on Quantity

```
USE [xyz_dmart]

SELECT
	SUM(quantity) total_quanity,
	pro.product_name,
	fas.product_key
FROM dbo.fact_sales fas
INNER JOIN dbo.dimension_products pro
ON fas.product_key = pro.product_key
GROUP BY pro.product_name, fas.product_key
ORDER BY total_quanity desc;
```

Figure 61: Query to find out the top-selling item.

Results | Messages

	total_quanity	product_name	product_key
1	73698	Perfect Fitness Perfect Rip Deck	365
2	62956	Nike Men's Dri-FIT Victory Golf Polo	502
3	57803	O'Brien Men's Neoprene Life Vest	1014
4	36680	Nike Men's Free 5.0+ Running Shoe	191
5	31735	Under Armour Girls' Toddler Spine Surge Runni	627
6	22246	Nike Men's CJ Elite 2 TD Football Cleat	403
7	17325	Field & Stream Sportsman 16 Gun Fire Safe	1004
8	15500	Pelican Sunstream 100 Kayak	1073
9	13729	Diamondback Women's Serene Classic Comfort Bi	957
10	998	ENO Atlas Hammock Straps	977
11	991	Nike Men's Comfort 2 Slide	116
12	969	adidas Youth Germany Black/Red Away Match Soc	565
13	958	Team Golf St. Louis Cardinals Putter Grip	885
14	946	LIJA Women's Eyelet Sleeveless Golf Polo	728
15	941	Glove It Women's Imperial Golf Glove	804
16	939	adidas Men's F10 Messi TRX FG Soccer Cleat	44
17	930	Glove It Imperial Golf Towel	926
18	930	Glove It Women's Mod Oval 3-Zip Carry All Gol	917
19	928	Bridgestone e6 Straight Distance NFL Carolina	835

Figure 62: List of top-selling items in the order.

The top-selling product based on quantities is “Perfect Fitness Perfect Rip Deck”

Month analysis of popular selling product "Perfect Fitness Perfect Rip Deck."

```
SELECT
	MONTH(dat.TheDate) TheMonth,
	SUM(fas.subtotal_sold_price) total_sales_per_month,
	SUM(fas.profit) total_profit_generated,
	pro.product_name product_name,
	fas.department_key department_key
FROM dbo.dimension_date dat
INNER JOIN dbo.fact_sales fas
ON fas.sales_date_key = dat.DateKey
INNER JOIN dbo.dimension_products pro
ON pro.product_key = fas.product_key
WHERE pro.product_name = 'Perfect Fitness Perfect Rip Deck'
GROUP BY MONTH(dat.TheDate), department_key, pro.product_name
ORDER BY total_sales_per_month DESC;
```

Figure 63: Query to find the popular month of the top sold item.

Results | Messages

	TheMonth	total_sales_per_month	total_profit_generated	product_name	department_key
1	11	485558.96	80926.41	Perfect Fitness Perfect Rip Deck	4
2	7	461659.15	76943.28	Perfect Fitness Perfect Rip Deck	4
3	1	456404.11	76067.51	Perfect Fitness Perfect Rip Deck	4
4	2	454604.03	75767.18	Perfect Fitness Perfect Rip Deck	4
5	9	442942.00	73823.53	Perfect Fitness Perfect Rip Deck	4
6	3	440206.59	73367.74	Perfect Fitness Perfect Rip Deck	4
7	8	438334.94	73055.83	Perfect Fitness Perfect Rip Deck	4
8	5	432935.70	72155.84	Perfect Fitness Perfect Rip Deck	4
9	12	430704.26	71784.09	Perfect Fitness Perfect Rip Deck	4
10	4	428976.69	71496.28	Perfect Fitness Perfect Rip Deck	4
11	6	422785.21	70463.94	Perfect Fitness Perfect Rip Deck	4
12	10	410259.65	68376.64	Perfect Fitness Perfect Rip Deck	4

Figure 64: Month Result

It is November Month when the total sales are 485558.96

Daily analysis of popular selling product "Perfect Fitness Perfect Rip Deck."

```
SELECT
	DAY(dat.TheDate) TheDay,
	SUM(fas.subtotal_sold_price) total_sales_per_month,
	SUM(fas.profit) total_profit_per_month,
	pro.product_name product_name,
	fas.department_key department_key
FROM dbo.dimension_date dat
INNER JOIN dbo.fact_sales fas
ON fas.sales_date_key = dat.DateKey
INNER JOIN dbo.dimension_products pro
ON pro.product_key = fas.product_key
WHERE pro.product_name = 'Perfect Fitness Perfect Rip Deck'
GROUP BY DAY(dat.TheDate), fas.department_key, pro.product_name
ORDER BY total_sales_per_month DESC;
```

Figure 65: Query Daily Analysis

Results | Messages

	TheDay	total_sales_per_month	total_profit_per_month	product_name	department_key
1	3	204589.86	34098.28	Perfect Fitness Perfect Rip Deck	4
2	10	200990.54	33498.46	Perfect Fitness Perfect Rip Deck	4
3	5	197319.07	32886.48	Perfect Fitness Perfect Rip Deck	4
4	1	193503.59	32250.47	Perfect Fitness Perfect Rip Deck	4
5	6	191632.09	31938.71	Perfect Fitness Perfect Rip Deck	4
6	18	191416.19	31902.78	Perfect Fitness Perfect Rip Deck	4
7	11	186808.80	31134.75	Perfect Fitness Perfect Rip Deck	4
8	15	185081.18	30846.89	Perfect Fitness Perfect Rip Deck	4
9	20	184001.30	30666.86	Perfect Fitness Perfect Rip Deck	4
10	26	179250.17	29875.07	Perfect Fitness Perfect Rip Deck	4
11	28	179178.07	29862.96	Perfect Fitness Perfect Rip Deck	4
12	25	177954.41	29659.13	Perfect Fitness Perfect Rip Deck	4
13	2	177162.34	29526.95	Perfect Fitness Perfect Rip Deck	4
14	23	176010.59	29335.04	Perfect Fitness Perfect Rip Deck	4
15	27	172843.22	28807.23	Perfect Fitness Perfect Rip Deck	4
16	24	172699.21	28783.20	Perfect Fitness Perfect Rip Deck	4
17	7	171763.42	28627.28	Perfect Fitness Perfect Rip Deck	4
18	14	171043.43	28507.19	Perfect Fitness Perfect Rip Deck	4
19	9	170323.77	28387.43	Perfect Fitness Perfect Rip Deck	4

Figure 66: Daily Analysis Result

On every 3rd day of the month product 'Perfect Fitness Perfect Rip Deck' is maximum sold.

3. Apache (Hadoop, Sqoop and Hive)

Apache Hadoop is a software library framework that allows distributed process of extensive data set across a cluster of computer systems using programming models. It is scalable from a single machine to thousands of machines, offering local computation and storage. Hadoop can process both structured and unstructured data types. (The Apache Software Foundation, 2022)

Apache Sqoop ™ is a tool that efficiently transfers bulk data between Apache Hadoop and structured data stores. The latest version of Apache Sqoop is Sqoop2 (1.99.7). (The Apache Software Foundation, 2022)

The Apache Hive ™ data warehouse software facilitates reading, writing, and managing large datasets residing in distributed storage using SQL. A structure can be projected onto data already in storage. A command-line tool and JDBC driver are provided to connect users to Hive. (The Apache Software Foundation, 2022)

3.1 Export Warehouse Database to Hive Staging

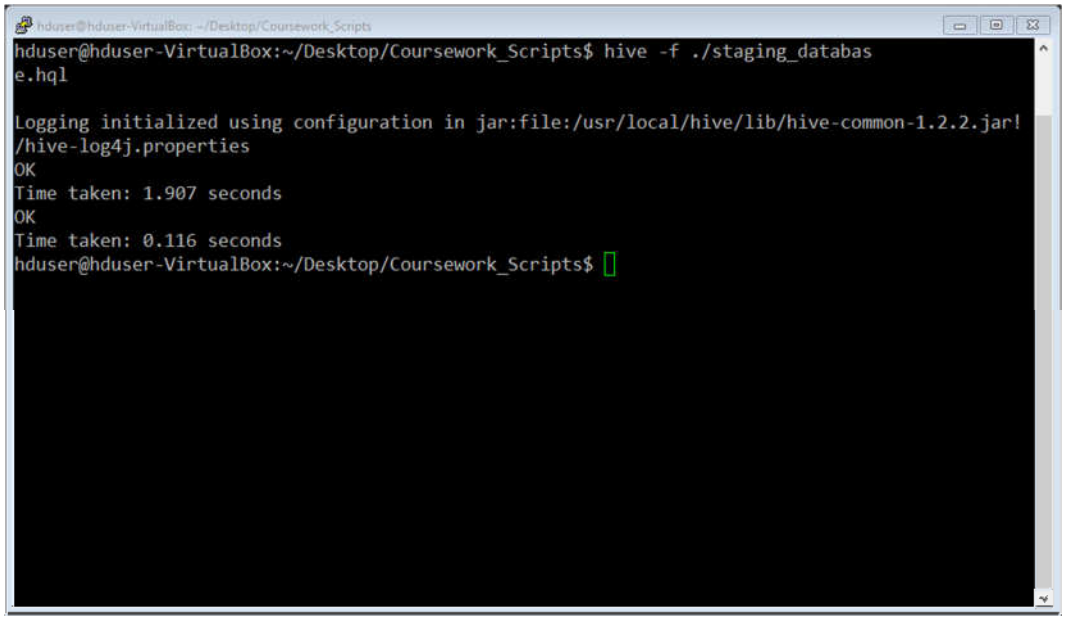


Figure 67: Using "staging_database.hql" script to create Hive Staging Area.

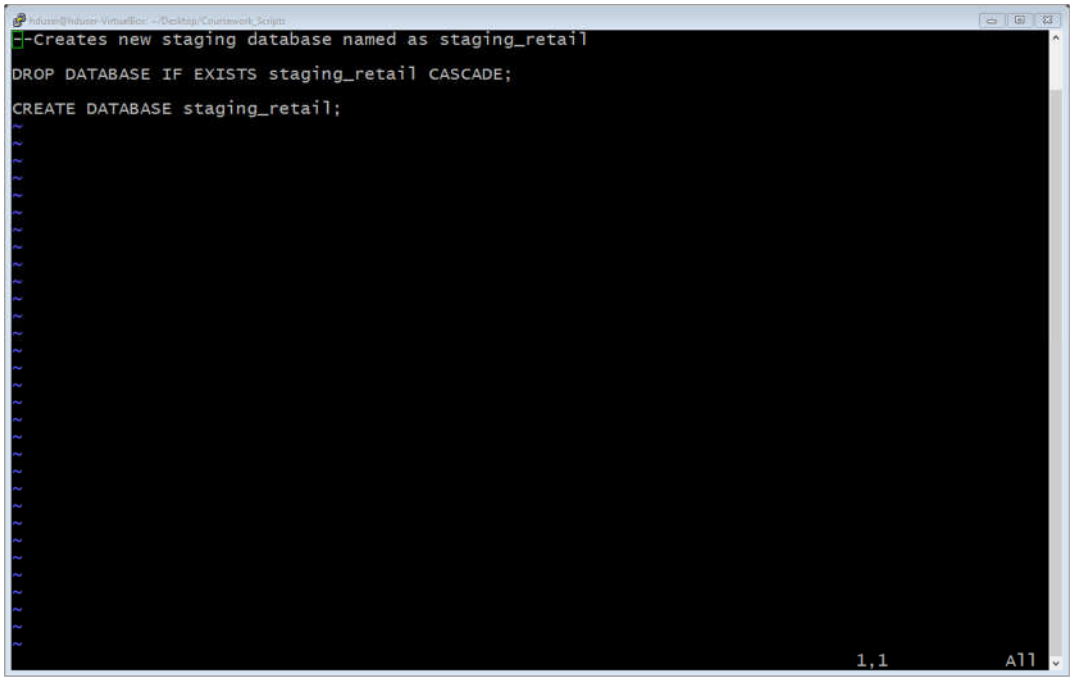


Figure 68: Commands in Script.

Importing all the retail_db database tables from MySQL into hive staging database using sqoop command.

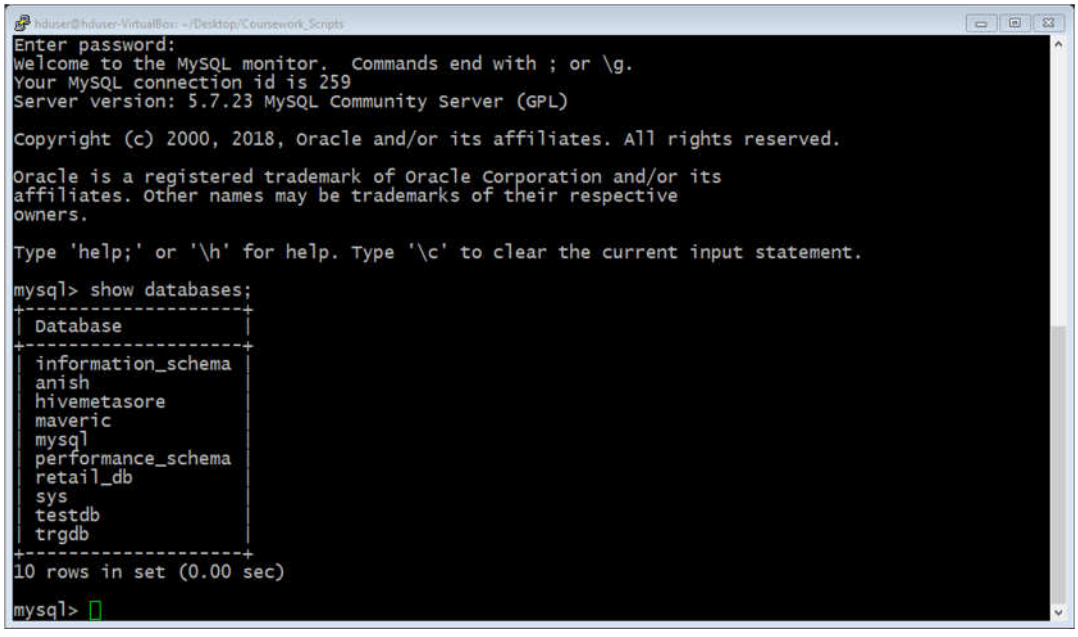


Figure 69: retail_db Database inside MySQL

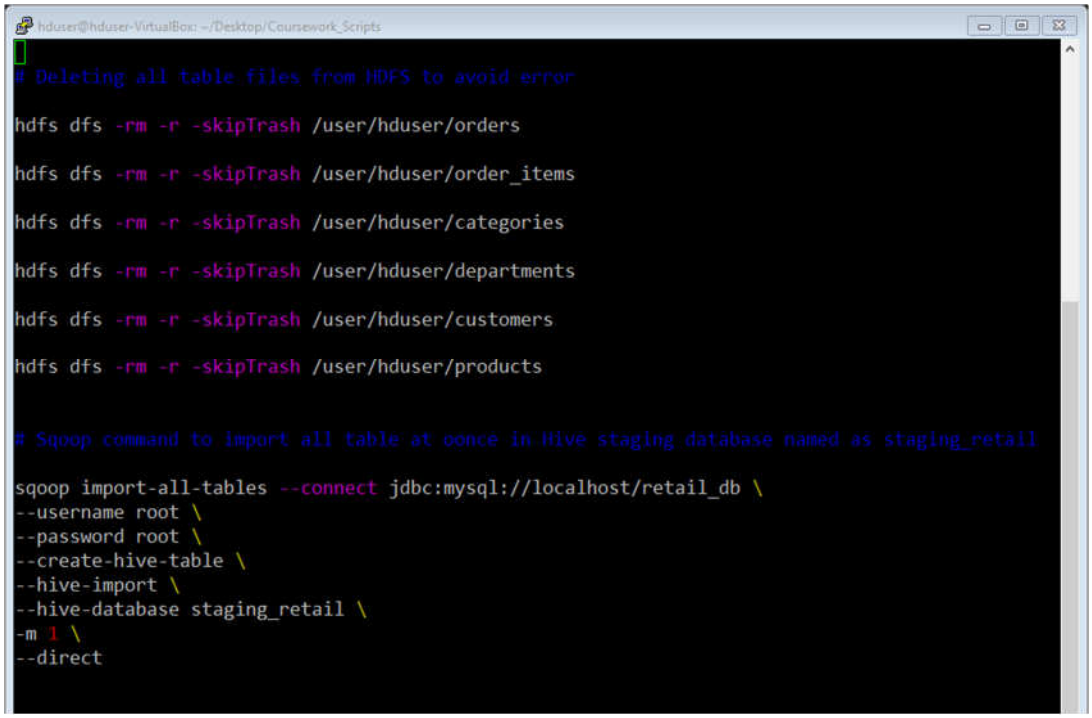


Figure 70: Importing all tables into Hive Database

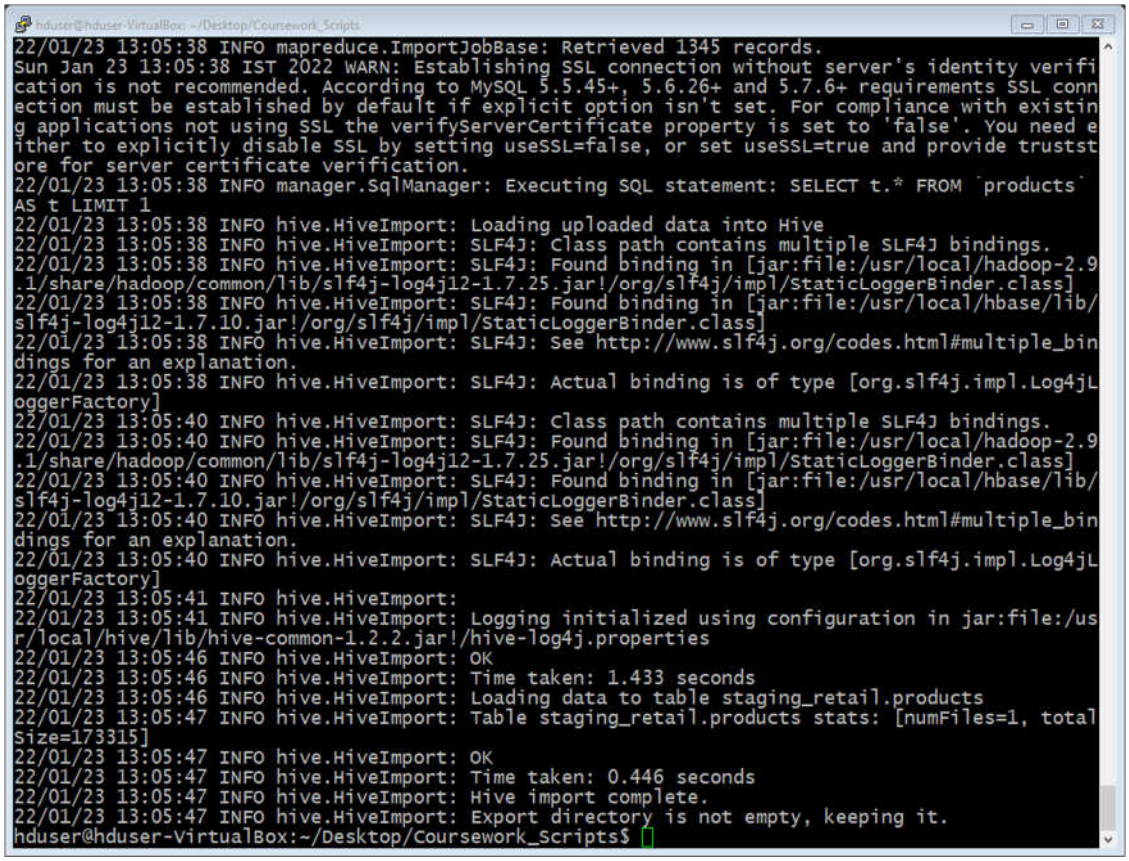


Figure 71: Executing import_all_tables.sh script

Verify all the tables and data import from retail_db (MySQL) to staging_retail (Hive) and HDFS.

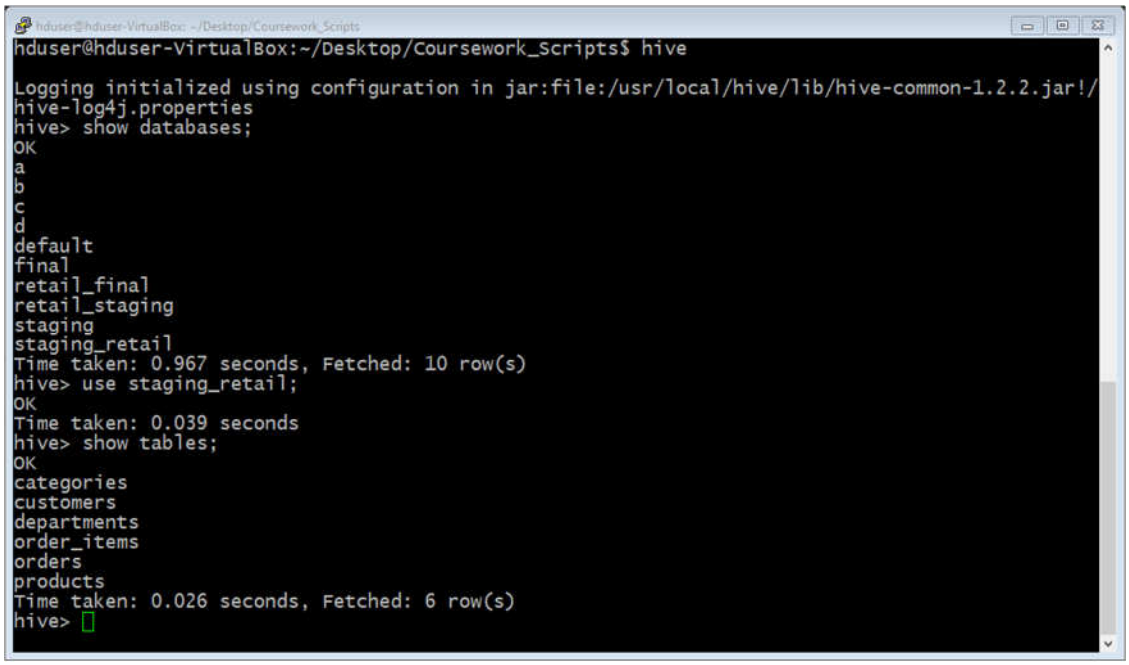


Figure 72: Staging database and tables imported

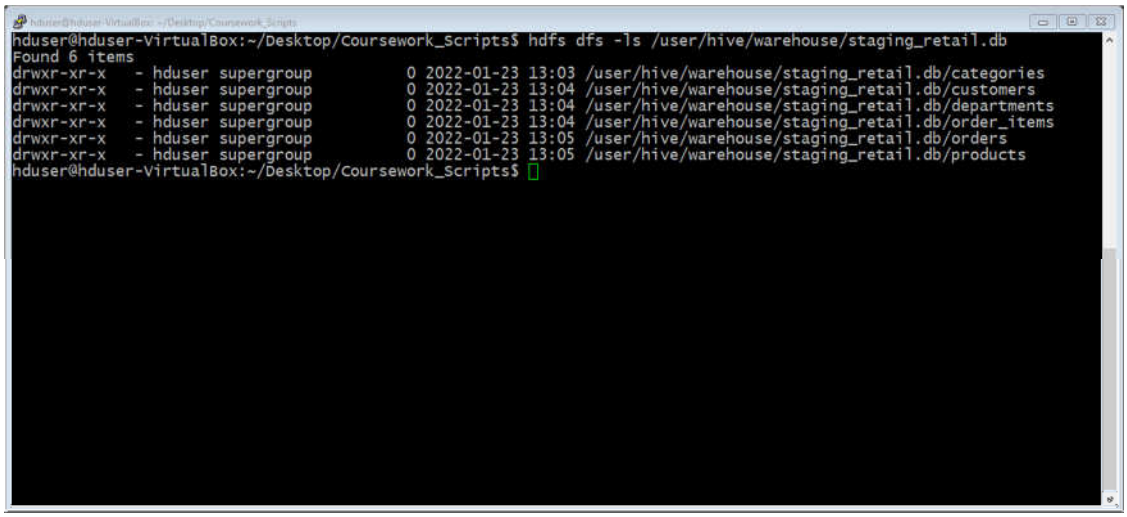


Figure 73: staging_retail database tables stored in HDFS

3.2 Data Transformation for Analysis & Reporting in Hive

3.2.1 Creating xyz_dmart Database to Store Dimension and Fact Table

hduser@hduser-VirtualBox: ~/Desktop/Coursework_Scripts

```
--Creating data mart database to store dimension and fact tables

DROP DATABASE IF EXISTS xyz_dmart CASCADE;

CREATE DATABASE xyz_dmart;

SHOW DATABASES;

USE xyz_dmart;

--Creating dimension and fact tables

DROP TABLE IF EXISTS dimension_departments;

CREATE EXTERNAL TABLE dimension_departments (
depart_key int
)
partitioned by (depart_name string)
row format delimited
fields terminated by ','
stored as ORC;

DROP TABLE IF EXISTS dimension_orders;

CREATE EXTERNAL TABLE dimension_orders (
order_key int,
order_status string,
order_date timestamp
)
row format delimited
fields terminated by ','
stored as ORC;
```

Figure 74: Creating xyz_dmart Database along with Dimension and Fact Tables – Part 1

```
DROP TABLE IF EXISTS dimension_products;

CREATE EXTERNAL TABLE dimension_products (
product_key int,
category_key int,
product_name string,
product_price DECIMAL (15,3)
)
row format delimited
fields terminated by ','
stored as ORC;


DROP TABLE IF EXISTS dimension_order_items;

CREATE EXTERNAL TABLE dimension_order_items (
order_items_key int,
order_key int,
product_key int,
subtotal_cost_price DECIMAL (15, 3)
)
partitioned by (order_quantity int)
row format delimited
fields terminated by ','
stored as ORC;
```

Figure 75: Creating xyz_dmart Database along with Dimension and Fact Tables – Part 2

```
CREATE EXTERNAL TABLE dimension_products (
product_key int,
category_key int,
product_name string,
product_price DECIMAL (15,3)
)
row format delimited
fields terminated by ','
stored as ORC;


DROP TABLE IF EXISTS dimension_order_items;

CREATE EXTERNAL TABLE dimension_order_items (
order_items_key int,
order_key int,
product_key int,
subtotal_cost_price DECIMAL (15, 3)
)
partitioned by (order_quantity int)
row format delimited
fields terminated by ','
stored as ORC;

DROP TABLE IF EXISTS dimension_categories;

CREATE EXTERNAL TABLE dimension_categories (
category_key int,
department_key int,
category_name string
)
row format delimited
fields terminated by ','
stored as ORC;

DROP TABLE IF EXISTS FactRetail_sales;

CREATE EXTERNAL TABLE fact_sales (
product_key int,
product_cost_price decimal (25,2),
subtotal_product_cost_price decimal (25,2),
subtotal_sold_price decimal (25,2),
profit decimal (25,2),
order_key int,
order_items_key int,
category_key int,
department_key int,
salesdate timestamp,
Quantity int
)
row format delimited
fields terminated by ','
stored as ORC;

set hive.exec.dynamic.partition.mode=nonstrict;
```

Figure 76: Creating xyz_dmart Database along with Dimension and Fact Tables – Part 1

3.3 Loading Data into xyz_dmart and Data Manipulation Demonstration

```
set hive.exec.dynamic.partition.mode=nonstrict;


--Loading required data in tables

INSERT INTO dimension_orders
SELECT order_id, order_status, order_date FROM staging_retail.orders;


INSERT INTO dimension_departments partition (depart_name)
SELECT department_id, department_name FROM staging_retail.departments;


INSERT INTO dimension_products
SELECT product_id, product_category_id, product_name, product_price FROM staging_retail.products;

INSERT INTO dimension_order_items partition (order_quantity)
SELECT order_item_id, order_item_order_id, order_item_product_id, order_item_subtotal, order_item_quantity FROM staging_retail.order_items;


INSERT INTO dimension_categories
SELECT category_id, category_department_id, category_name FROM staging_retail.categories;
```

Figure 77: Inserting Data into External Table

```
INSERT INTO fact_sales
SELECT
pro.product_key product_key,
sum(pro.product_price) product_cost_price,
sum(doi.subtotal_cost_price) subtotal_product_cost_price,
(sum(doi.subtotal_cost_price) + sum(doi.subtotal_cost_price)* 0.2) subtotal_sold_price,
((sum(doi.subtotal_cost_price) + sum(doi.subtotal_cost_price)*0.2) - (sum(doi.subtotal_cost_price))) Profit,
doi.order_key order_key,
doi.order_items_key order_items_key,
pro.category_key category_key,
depart.depart_key department_key,
ord.order_date sales_date,
doi.order_quantity quantity
FROM dimension_order_items doi
LEFT JOIN dimension_products pro ON pro.product_key = doi.product_key
LEFT JOIN dimension_categories cat ON pro.category_key = cat.category_key
LEFT JOIN dimension_departments depart ON cat.department_key = depart.depart_key
LEFT JOIN dimension_orders ord ON ord.order_key = doi.order_key
GROUP BY pro.product_key,
doi.order_key,
doi.order_items_key,
pro.category_key,
depart.depart_key,
ord.order_date,
doi.order_quantity
```

Figure 78: Inserting Dato into Fact Table

3.3.1 Verifying the created table inside Hive and HDFS

```
Logging initialized using configuration in jar:file:/usr/local/hive/lib/hive-common-1.2.2.jar!/hive-log4j.properties
hive> show databases;
OK
default
staging_retail
xyz_dmart
Time taken: 0.936 seconds, Fetched: 3 row(s)
hive> use xyz_dmart;
OK
Time taken: 0.028 seconds
hive> show tables;
OK
dimension_categories
dimension_departments
dimension_order_items
dimension_orders
dimension_products
fact_sales
Time taken: 0.024 seconds, Fetched: 6 row(s)
hive>
```

Figure 79: All table list stored in xyz_dmart

```
hduser@hduser-VirtualBox: ~/Desktop/Coursework_Scripts
hduser@hduser-VirtualBox:~/Desktop/Coursework_Scripts$ hdfs dfs -ls /user/hive/warehouse/xyz_dmart.db
Found 6 items
drwxr-xr-x   - hduser supergroup          0 2022-01-23 16:37 /user/hive/warehouse/xyz_dmart.db/dimension_categories
drwxr-xr-x   - hduser supergroup          0 2022-01-23 16:36 /user/hive/warehouse/xyz_dmart.db/dimension_departments
drwxr-xr-x   - hduser supergroup          0 2022-01-23 16:37 /user/hive/warehouse/xyz_dmart.db/dimension_order_items
drwxr-xr-x   - hduser supergroup          0 2022-01-23 16:36 /user/hive/warehouse/xyz_dmart.db/dimension_orders
drwxr-xr-x   - hduser supergroup          0 2022-01-23 16:37 /user/hive/warehouse/xyz_dmart.db/dimension_products
drwxr-xr-x   - hduser supergroup          0 2022-01-23 16:38 /user/hive/warehouse/xyz_dmart.db/fact_sales
hduser@hduser-VirtualBox:~/Desktop/Coursework_Scripts$
```

Figure 80: Table Stored in HDFS file system

3.3.2 Demonstration of Data Manipulation on Loaded Data

```
Logging initialized using configuration in jar:file:/usr/local/hive/lib/hive-common-1.2.2.jar!/hive-log4j.properties
hive> use xyz_dmart;
OK
Time taken: 0.764 seconds
hive> SELECT
    > SUM(fas.Profit) total_profit,
    > depart.depart_name department_name,
    > fas.department_key
    > FROM fact_sales fas
    > INNER JOIN dimension_departments depart
    > ON depart.depart_key = fas.department_key
    > GROUP BY depart.depart_name, fas.department_key
    > ORDER BY total_profit desc;
```

Figure 81: Query of Total Profit of each Department.

```
sec
MapReduce Total cumulative CPU time: 2 seconds 920 msec
Ended Job = job_1643009993128_0006
MapReduce Jobs Launched:
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 6.29 sec   HDFS Read: 105474
HDFS Write: 287 SUCCESS
Stage-Stage-3: Map: 1  Reduce: 1   Cumulative CPU: 2.92 sec   HDFS Read: 5202 HD
FS Write: 121 SUCCESS
Total MapReduce CPU Time Spent: 9 seconds 210 msec
OK
3421708.79      Fan Shop        7
1464784.27      Apparel 4
921806.16       Golf    5
801299.47       Footwear        3
199118.15       Outdoors        6
56009.26        Fitness 2
Time taken: 47.749 seconds, Fetched: 6 row(s)
hive>
```

Figure 82: Result on Profit of each Department

```
hive> SELECT
    > SUM(fas.subtotal_sold_price) as sold_price_subtotal,
    > fas.product_key,
    > pro.product_name product_name
    > FROM fact_sales fas
    > INNER JOIN dimension_products pro ON pro.product_key = fas.product_key
    > GROUP BY fas.product_key, pro.product_name
    > ORDER BY sold_price_subtotal desc
    > LIMIT 5;
```

Figure 83: Top Revenue Generating Product Query

```
MapReduce Jobs Launched:
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 6.24 sec   HDFS Read: 102018 HDFS Write: 6305 SUCCESS
Stage-Stage-3: Map: 1  Reduce: 1   Cumulative CPU: 2.99 sec   HDFS Read: 11344 HDFS Write: 263 SUCCESS
Total MapReduce CPU Time Spent: 9 seconds 230 msec
OK
8315653.5       1004    Field & Stream Sportsman 16 Gun Fire Safe
5305371.29      365     Perfect Fitness Perfect Rip Deck
4942165.42      957     Diamondback Women's Serene Classic Comfort Bi
4401159.5       191     Nike Men's Free 5.0+ Running Shoe
3777360 502     Nike Men's Dri-FIT Victory Golf Polo
Time taken: 46.074 seconds, Fetched: 5 row(s)
```

Figure 84: List of Top Revenue Generating Product

```
Time taken: 46.074 seconds, Fetched: 5 row(s)
hive> SELECT
    > SUM(quantity) total_quantity,
    > pro.product_name,
    > fas.product_key
    > FROM fact_sales fas
    > INNER JOIN dimension_products pro
    > ON fas.product_key = pro.product_key
    > GROUP BY pro.product_name, fas.product_key
    > ORDER BY total_quantity desc;
```

Figure 85: Top Selling/Popular Product Query

```
Starting Job = job_1643009993128_0010, Tracking URL = http://hduser-VirtualBox:8088/proxy/application_1643009993128_0010/
Kill Command = /usr/local/hadoop-2.9.1/bin/hadoop job  -kill job_1643009993128_0010
Hadoop job information for Stage-3: number of mappers: 1; number of reducers: 1
2022-01-24 14:08:27,119 Stage-3 map = 0%,  reduce = 0%
2022-01-24 14:08:32,260 Stage-3 map = 100%,  reduce = 0%, Cumulative CPU 1.21 sec
2022-01-24 14:08:37,423 Stage-3 map = 100%,  reduce = 100%, Cumulative CPU 3.0 sec
MapReduce Total cumulative CPU time: 3 seconds 0 msec
Ended Job = job_1643009993128_0010
MapReduce Jobs Launched:
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 5.65 sec   HDFS Read: 74491 HDFS Write: 6007 SUCCESS
Stage-Stage-3: Map: 1  Reduce: 1   Cumulative CPU: 3.0 sec   HDFS Read: 10700 HDFS Write: 4463 SUCCESS
Total MapReduce CPU Time Spent: 8 seconds 650 msec
OK
73698   Perfect Fitness Perfect Rip Deck        365
62956   Nike Men's Dri-FIT Victory Golf Polo    502
57803   O'Brien Men's Neoprene Life Vest        1014
36680   Nike Men's Free 5.0+ Running Shoe       191
31735   Under Armour Girls' Toddler Spine Surge Runni   627
22246   Nike Men's CJ Elite 2 TD Football Cleat 403
17325   Field & Stream Sportsman 16 Gun Fire Safe       1004
15500   Pelican Sunstream 100 Kayak     1073
13729   Diamondback Women's Serene Classic Comfort Bi   957
998     ENO Atlas Hammock Straps        977
991     Nike Men's Comfort 2 Slide      116
```

Figure 86: Top Selling/Popular Product List

3.4 File formats Demonstration and other optimisation Techniques in Sqoop and Hive.

3.4.1 Use of ORC File Format

ORC is a self-describing type-aware columnar file format designed for Hadoop workloads optimized for large streaming reads with integrated support for finding required rows quickly. Storing data in a columnar format lets the reader read, decompress, and process only the values that are required for the current query. (Apache Software Foundation, 2022)

3.4.2 Use of Single Mapper in Sqoop

Mappers form a connection to source database to extract the data. This project is done using single clustere mode of Hadoop the time to extract data will be high by using more than one mapper in single cluster mode. Using a single cluster there is no use of parallelism.

3.4.3 Use of Hive Partitioning

Data analysis require data filters on quality, department name. Use of Hive portioning technique all the data from column name quantity and department name are segregated and stored in their respective partition. Query execution time to generate the result is reduced while using partitioning.

3.4.4 Using native tool to extract data from database in Sqoop

Native tool helps us to import data faster compared to jdbc in sqoop command. The tool varies based on the data ware house. The MySQL database use '—direct' is the native tool for this database. So, it is used in sqoop for importing data.

3.4.5 Importing tables directly in hive with the use of Sqoop

Importing tables in HDFS and reimporting it on Hive from HDFS the sqoop command is developed to import tables from source dataset into Hive database directly. The sqoop command will create table based on source database tables into hive database. By simply specifying the database of Hive the tables are imported..

4. Personal Reflection

The coursework provide me an insight on Big Data World why data is being so valuable the amount of data in the world is increasing day by day. Data is being collected, processed, and stored at unprecedented rates. I learned the challenge of the data store, analysis, and to use the analysis in a meaningful way for business intelligence and growth of the Business.

The type of the forms of the data is changing and updating the unstructured data. Now the collected data need a more comprehensive data warehouse. Unstructured data is the fasted growing type of data that includes images, video, documents, logs, XML files and data files. There are several ways to handle, store and process these data's in the modern world. Most of these tools and techniques share scaling high availability and elasticity. In conjunction with the HDFS and HBase database as part of the Apache Hadoop project, MapReduce is a modern approach to analysing unstructured data.

This coursework help me to demonstrates the predicting model for forecasting sales data, their distribution of products, customer participation and the company value in the market, and the mechanisms used to provide the file-storage solution with high scalability availability for faster data processing.

The coursework has helped on understande the scenario of how traditional and contemporary data use and storage has changed.

Bibliography

David, T., 2021. *What is Dimensional Modeling in Data Warehouse? Learn Types.* [Online]
Available at: https://www.guru99.com/dimensional-model-data-warehouse.html
[Accessed 14 January 2022].

Peter, M. K. D. K. M. A., 2021. *Understand star schema and the importance for Power BI.* [Online]
Available at: https://docs.microsoft.com/en-us/power-bi/guidance/star-schema
[Accessed 15 Jan 2022].

Talend SA, 2021. *ETL Database.* [Online]
Available at: stitchdata.com
[Accessed 12 January 2022].

Taylor, D., 2021. *What is Business Intelligence? Definition & Example.* [Online]
Available at: https://www.guru99.com/business-intelligence-definition-example.html
[Accessed 16 Jan 2022].

Taylor, D., 2021. *What is Data Warehouse? Types, Definition & Example.* [Online]
Available at: https://www.guru99.com/data-warehousing.html
[Accessed 11 January 2022].

The Apache Software Foundation, 2022. *Apache Hadoop.* [Online]
Available at: https://hadoop.apache.org
[Accessed 17 January 2022].