Part 1: Design Document

A. Business Problem & Proposed Solution

1. Business Problem

EcoMart currently relies on flat CSV files to manage and analyze sales across multiple countries and regions. This setup lacks data relationships, leading to difficulties in efficiently tracking sales performance, comparing results across regions, and producing accurate reports. The growing volume of transactions further hinders performance, as the system cannot scale or support advanced analytics.

2. Proposed Data Structure

A relational star schema will be implemented, with the following structure:

- A central **Orders fact table** to track sales transactions.
- Supporting **dimension tables** for Regions, Countries, Item_Types, Sales_Channels, and Order_Priorities.
- This structure promotes data normalization, scalability, and clarity.

3. Justification for Database Solution

- **Data Relationships**: Fact and dimension tables enable linking and aggregating data (e.g., by country, region, or item type).
- **Performance**: Queries can be run efficiently to extract insights like revenue trends and best-selling products.

- **Scalability**: The schema supports partitioning and indexing, ensuring performance with larger datasets.

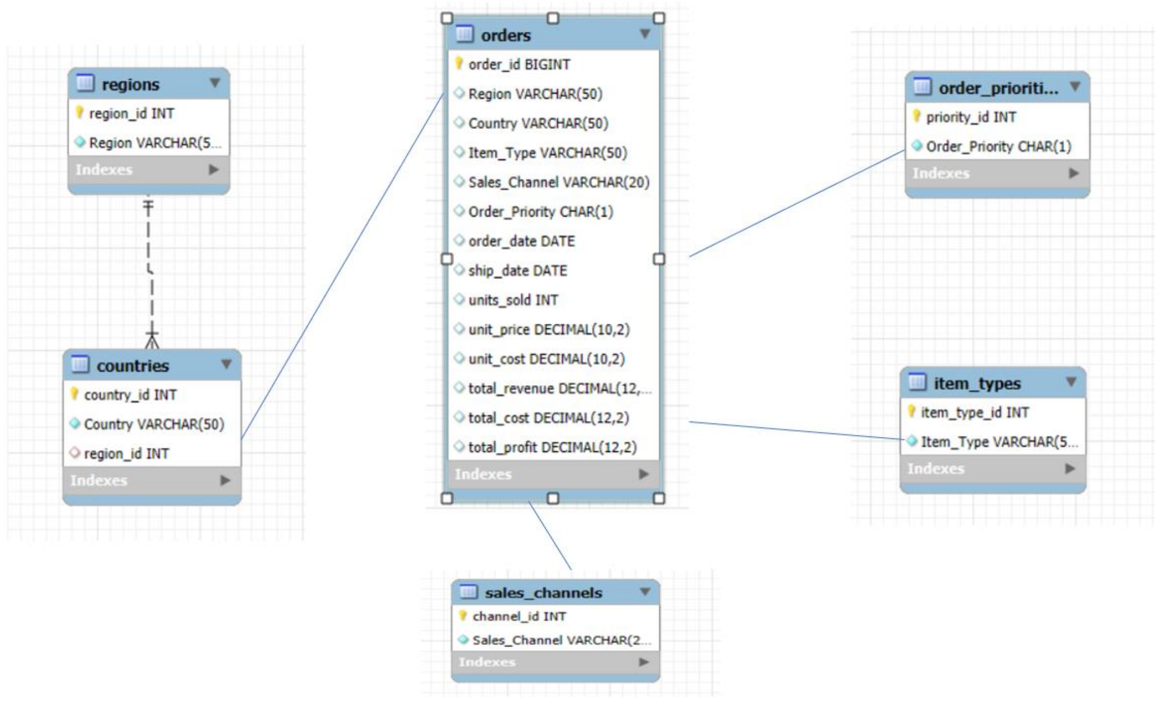
4. Business Data Usage

- **Sales Reports**: Identify top-performing products and regions.
- **Inventory Planning**: Analyze product demand across different channels.
- **Strategic Planning**: Use trends in order priorities and channels to optimize operations.

B. Logical Data Model

A star schema includes:

- **Fact Table: Orders**
  - order_id, Region, Country, Item_Type, Sales_Channel, Order_Priority, order_date, ship_date, units_sold, unit_price, unit_cost, total_revenue, total_cost, total_profit
- **Dimension Tables:**
  - Regions: region_id, Region
  - Countries: country_id, Country, region_id
  - Item_Types: item_type_id, Item_Type
  - Sales_Channels: channel_id, Sales_Channel
  - Order_Priorities: priority_id, Order_Priority



C. Database Objects and Storage

- **Tables**: Created using SQL Server Management Studio (SSMS) with appropriate data types.
- **Storage Considerations**:
  - Use BIGINT for order_id to handle high transaction volumes.
  - Use DECIMAL for financial columns to maintain precision.
  - Normalize country and region data to reduce redundancy.

D. Scalability Considerations

- **Partitioning**: The Orders table will be partitioned by order_date to improve performance and manage large datasets.
- **Indexes**: Create indexes on Item_Type, Region, and order_date to speed up query performance.
- **Star Schema**: Separating dimension data reduces data duplication and allows fast joins and aggregations.

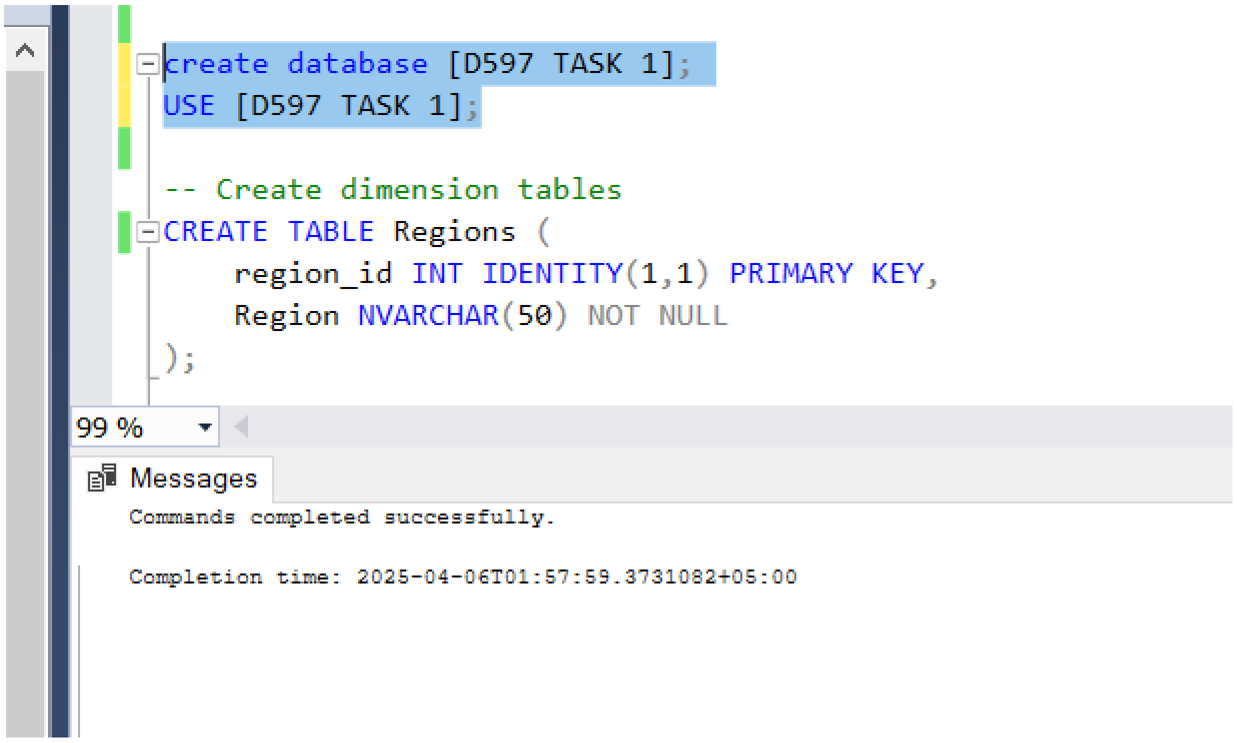
E. Privacy and Security Measures

- **Access Control**: Implement role-based access to restrict sensitive operations like updates or deletes.
- **Data Integrity**: Use foreign key constraints between dimension and fact tables.
- **Backup Strategy**: Set up regular backups and transaction logging to prevent data loss.
- **Encryption**: Ensure data-at-rest encryption for any sensitive business data fields.

Part 2: Implementation

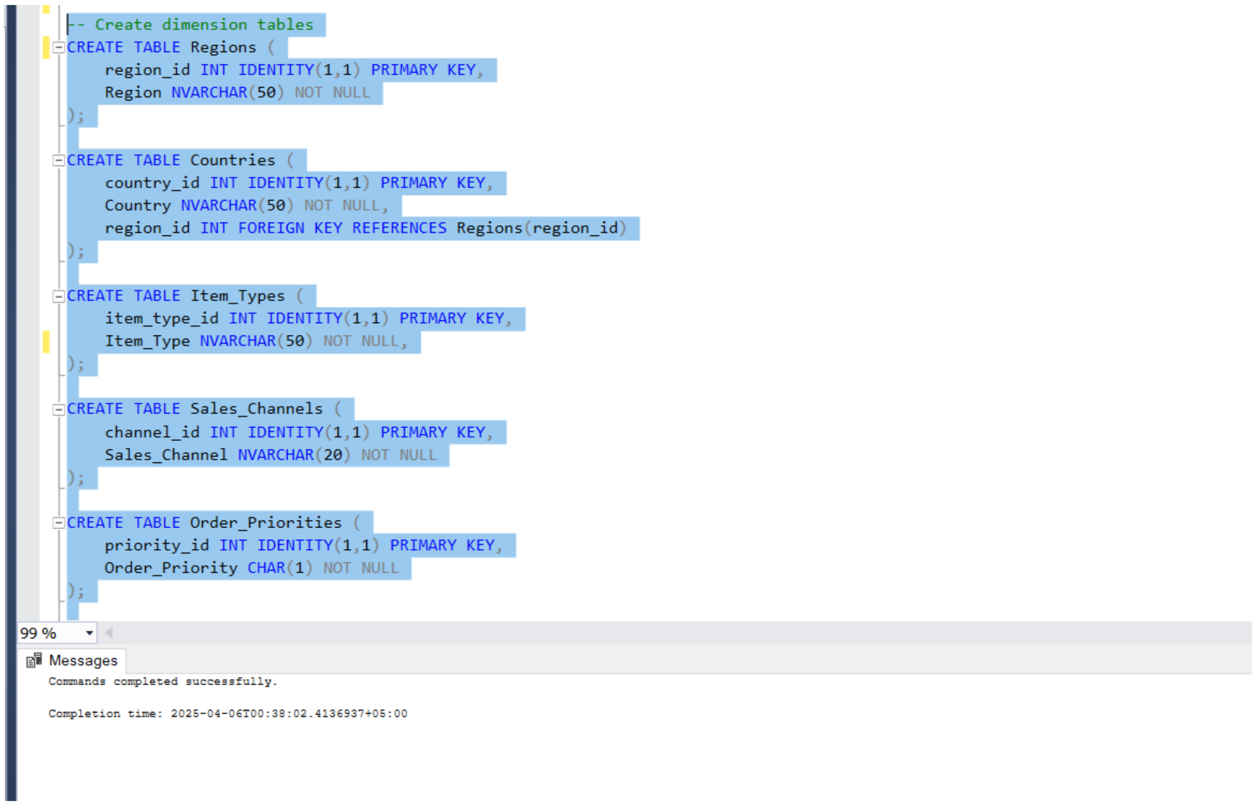
F1. Write script to create a database instance named “D597 Task 1” using the appropriate query language, based on the logical data model in part B. Provide a screenshot showing the script and the database instance in the platform.

Create SQL scripts



```
create database [D597 TASK 1];
USE [D597 TASK 1];

-- Create dimension tables
CREATE TABLE Regions (
    region_id INT IDENTITY(1,1) PRIMARY KEY,
    Region NVARCHAR(50) NOT NULL
);
```

```
Commands completed successfully.

Completion time: 2025-04-06T01:57:59.3731082+05:00
```

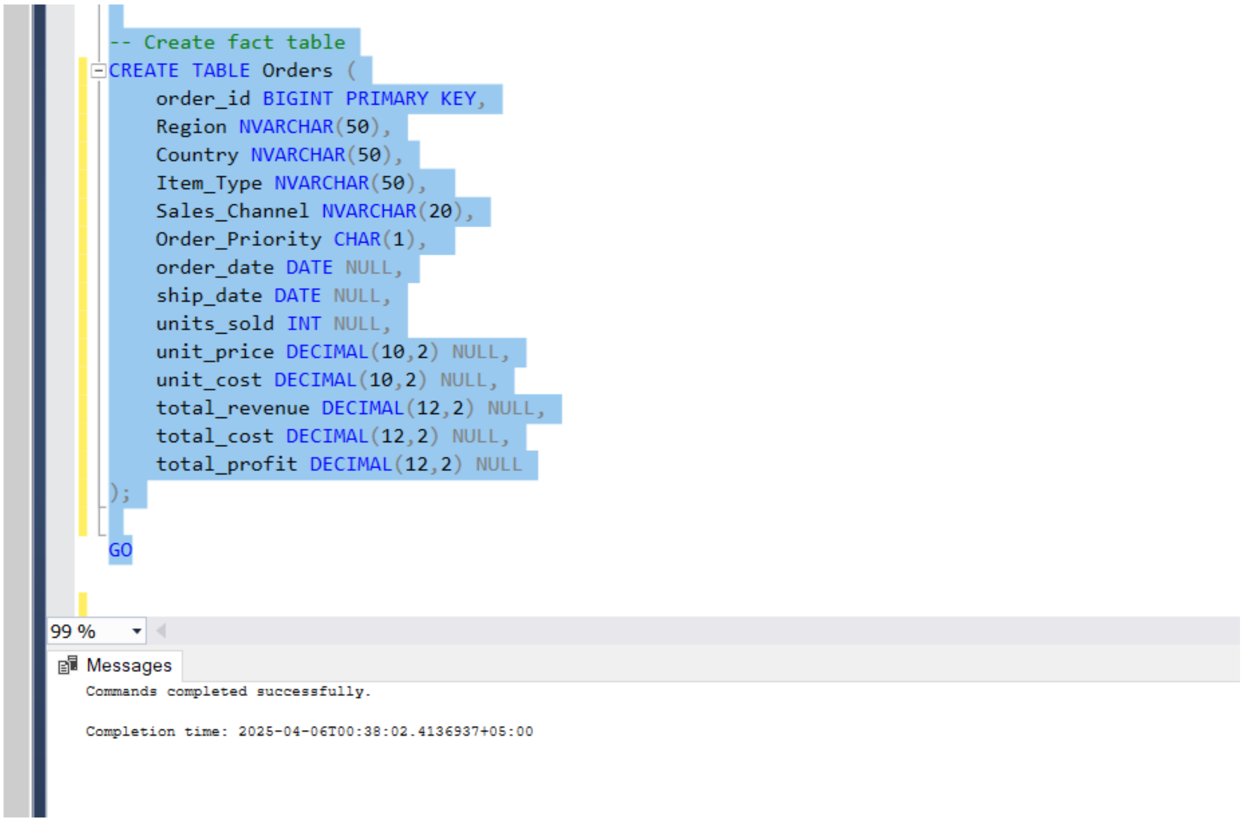


```
-- Create dimension tables
CREATE TABLE Regions (
    region_id INT IDENTITY(1,1) PRIMARY KEY,
    Region NVARCHAR(50) NOT NULL
);

CREATE TABLE Countries (
    country_id INT IDENTITY(1,1) PRIMARY KEY,
    Country NVARCHAR(50) NOT NULL,
    region_id INT FOREIGN KEY REFERENCES Regions(region_id)
);

CREATE TABLE Item_Types (
    item_type_id INT IDENTITY(1,1) PRIMARY KEY,
    Item_Type NVARCHAR(50) NOT NULL,
);

CREATE TABLE Sales_Channels (
    channel_id INT IDENTITY(1,1) PRIMARY KEY,
    Sales_Channel NVARCHAR(20) NOT NULL
);

CREATE TABLE Order_Priorities (
    priority_id INT IDENTITY(1,1) PRIMARY KEY,
    Order_Priority CHAR(1) NOT NULL
);
```

99 %

Messages

```
Commands completed successfully.

Completion time: 2025-04-06T00:38:02.4136937+05:00
```



```
-- Create fact table
CREATE TABLE Orders (
    order_id BIGINT PRIMARY KEY,
    Region NVARCHAR(50),
    Country NVARCHAR(50),
    Item_Type NVARCHAR(50),
    Sales_Channel NVARCHAR(20),
    Order_Priority CHAR(1),
    order_date DATE NULL,
    ship_date DATE NULL,
    units_sold INT NULL,
    unit_price DECIMAL(10,2) NULL,
    unit_cost DECIMAL(10,2) NULL,
    total_revenue DECIMAL(12,2) NULL,
    total_cost DECIMAL(12,2) NULL,
    total_profit DECIMAL(12,2) NULL
);

GO
```

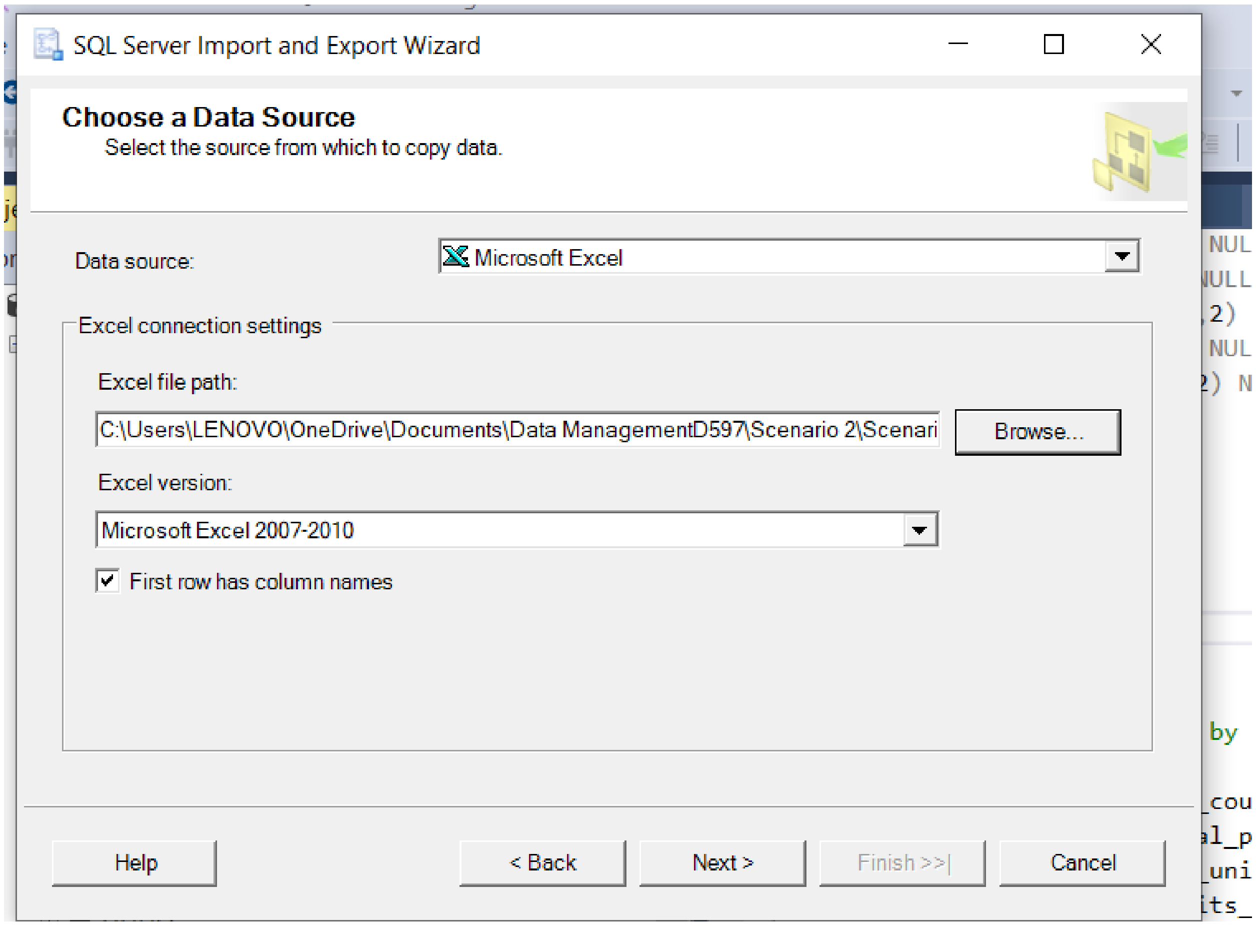
99 %

Messages

```
Commands completed successfully.

Completion time: 2025-04-06T00:38:02.4136937+05:00
```

Data Insertion

F2. Write script to import the data records from the chosen scenario CSV files into the database instance. Provide a screenshot showing the script and the data correctly inserted or mapped into the database.



Choose a Data Source

Select the source from which to copy data.

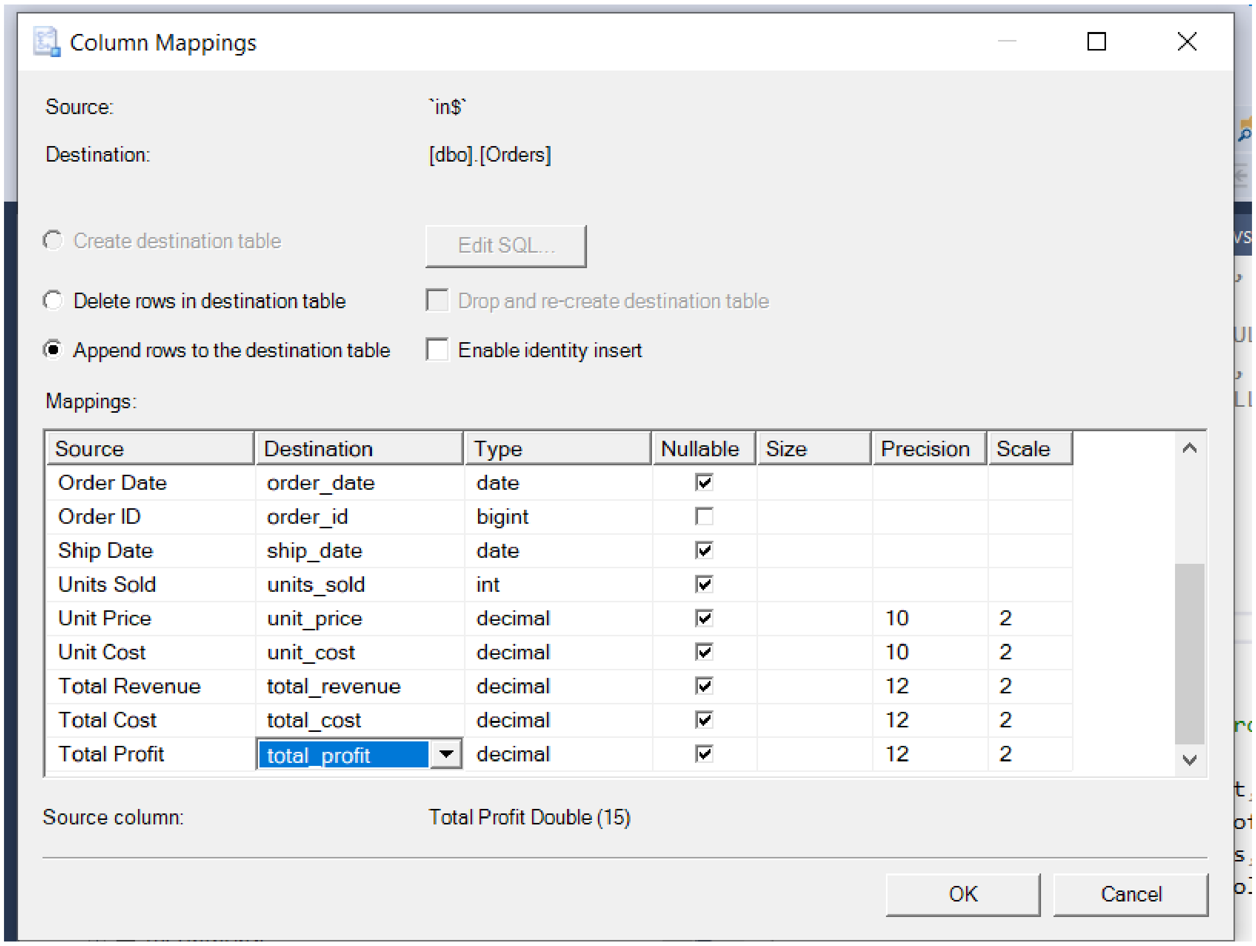
Data source: Microsoft Excel

Excel connection settings

Excel file path: C:\Users\LENOVO\OneDrive\Documents\Data ManagementD597\Scenario 2\Scenari

Excel version: Microsoft Excel 2007-2010

[✓] First row has column names



Column Mappings

Source: `in$`

Destination: [dbo].[Orders]

Create destination table — Edit SQL...

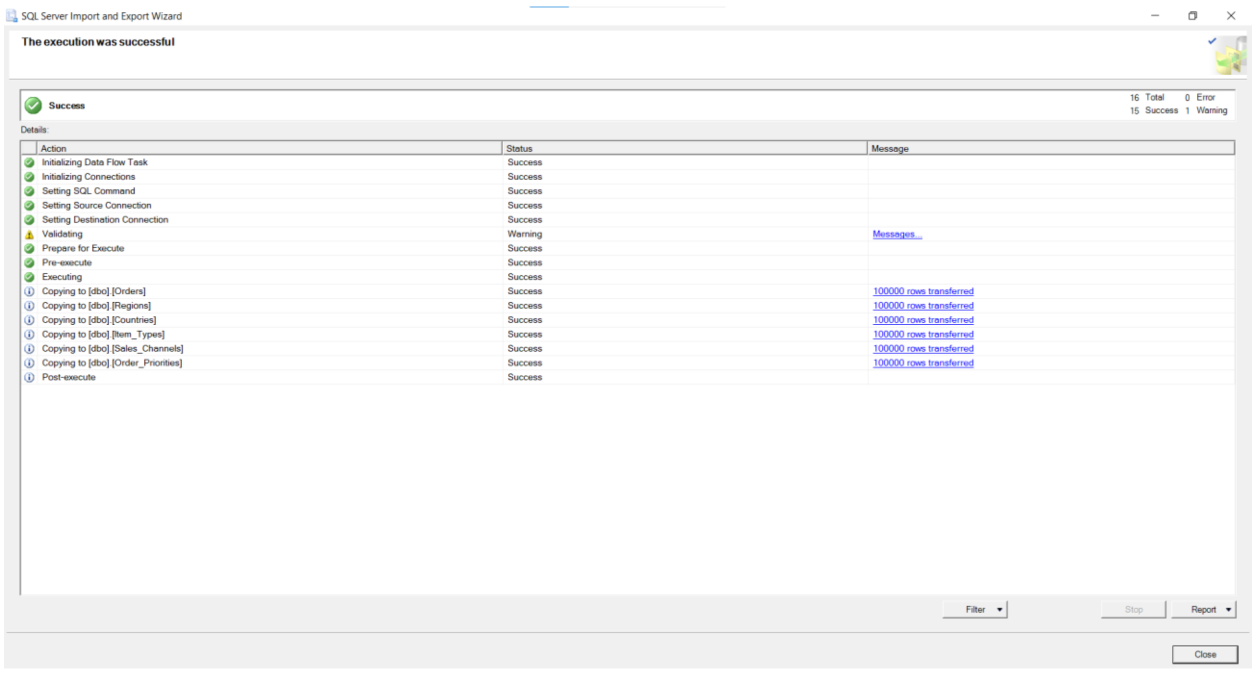
Delete rows in destination table — Drop and re-create destination table

[✓ Append rows to the destination table] — Enable identity insert

Mappings:

Source	Destination	Type	Nullable	Size	Precision	Scale
Order Date	order_date	date	✓			
Order ID	order_id	bigint				
Ship Date	ship_date	date	✓			
Units Sold	units_sold	int	✓			
Unit Price	unit_price	decimal	✓		10	2
Unit Cost	unit_cost	decimal	✓		10	2
Total Revenue	total_revenue	decimal	✓		12	2
Total Cost	total_cost	decimal	✓		12	2
Total Profit	total_profit	decimal	✓		12	2

Source column: Total Profit Double (15)



SQL Server Import and Export Wizard

The execution was successful

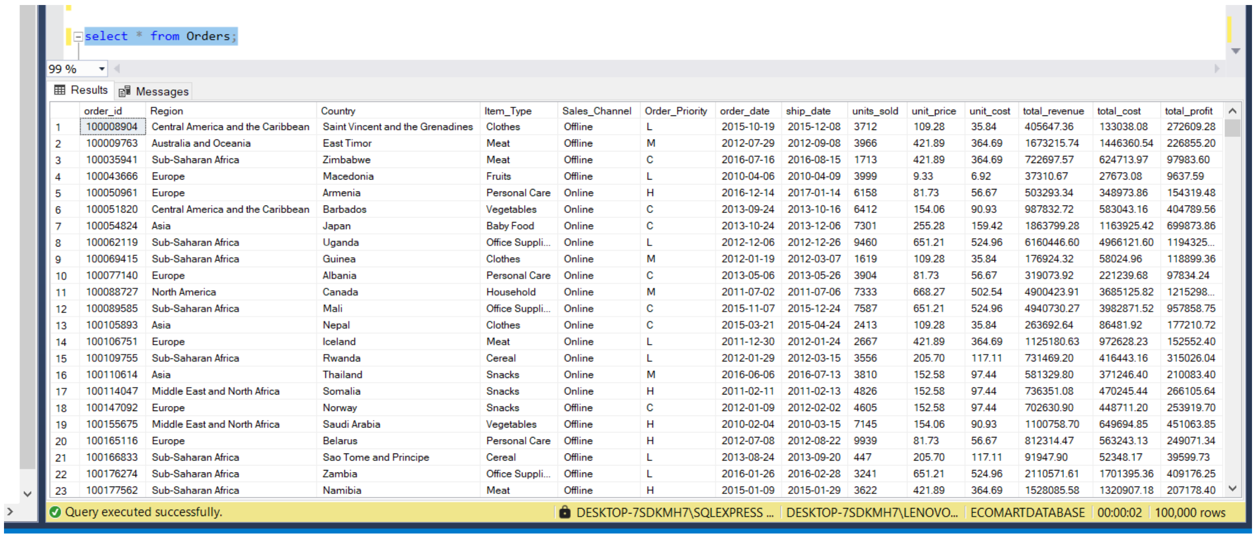
Success

16 Total 0 Error
15 Success 1 Warning

Details:

Action	Status	Message
Initializing Data Flow Task	Success	
Initializing Connections	Success	
Setting SQL Command	Success	
Setting Source Connection	Success	
Setting Destination Connection	Success	
Validating	Warning	Messages...
Prepare for Execute	Success	
Pre-execute	Success	
Executing	Success	
Copying to [dbo].[Orders]	Success	100000 rows transferred
Copying to [dbo].[Regions]	Success	100000 rows transferred
Copying to [dbo].[Countries]	Success	100000 rows transferred
Copying to [dbo].[Item_Types]	Success	100000 rows transferred
Copying to [dbo].[Sales_Channels]	Success	100000 rows transferred
Copying to [dbo].[Order_Priorities]	Success	100000 rows transferred
Post-execute	Success	

To check Data insert correctly:



```
select * from Orders;
```

99 %

Results | Messages

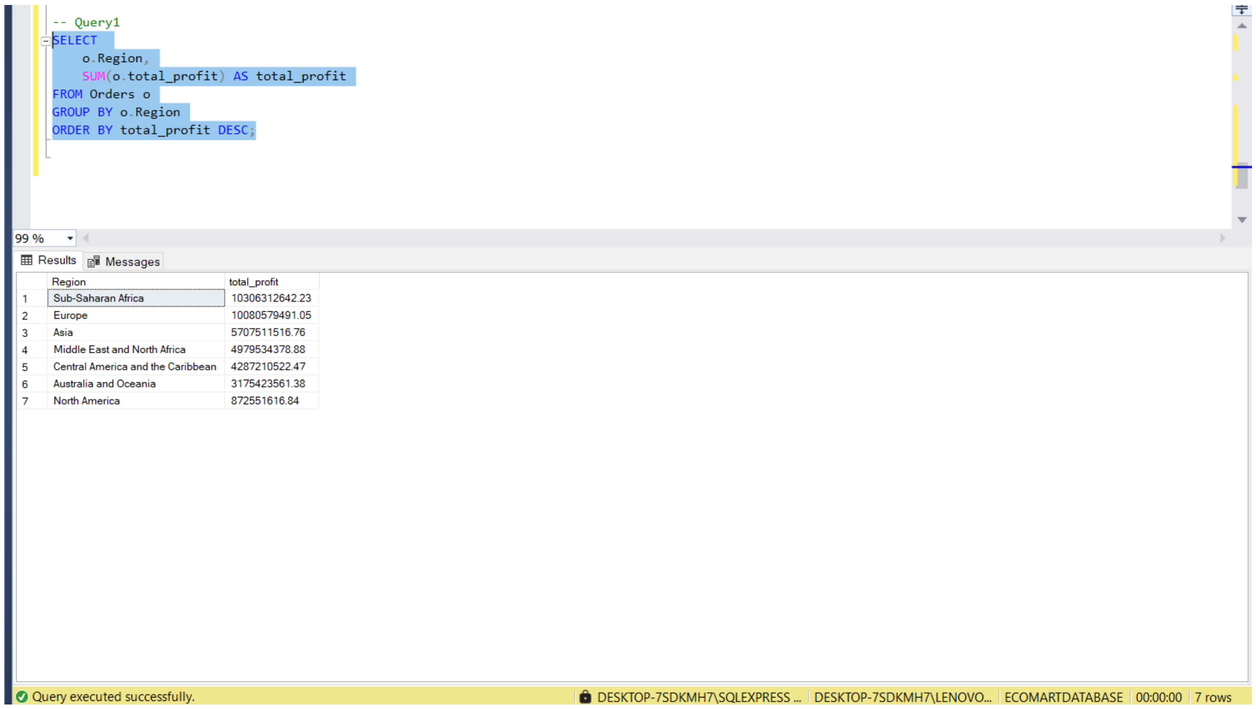
	order_id	Region	Country	Item_Type	Sales_Channel	Order_Priority	order_date	ship_date	units_sold	unit_price	unit_cost	total_revenue	total_cost	total_profit
1	100008904	Central America and the Caribbean	Saint Vincent and the Grenadines	Clothes	Offline	L	2015-10-19	2015-12-08	3712	109.28	35.84	405647.36	133038.08	272609.28
2	100009763	Australia and Oceania	East Timor	Meat	Offline	M	2012-07-29	2012-09-08	3966	421.89	364.69	1673215.74	1446360.54	226855.20
3	100035941	Sub-Saharan Africa	Zimbabwe	Meat	Offline	C	2016-07-16	2016-08-15	1713	421.89	364.69	722697.57	624713.97	97983.60
4	100043666	Europe	Macedonia	Fruits	Offline	L	2010-04-06	2010-04-09	3999	9.33	6.92	37310.67	27673.08	9637.59
5	100050961	Europe	Armenia	Personal Care	Online	H	2016-12-14	2017-01-14	6158	81.73	56.67	503293.34	348973.86	154319.48
6	100051820	Central America and the Caribbean	Barbados	Vegetables	Online	C	2013-09-24	2013-10-16	6412	154.06	90.93	987832.72	583043.16	404789.56
7	100054824	Asia	Japan	Baby Food	Online	C	2013-10-24	2013-12-06	7301	255.28	159.42	1863799.28	1163925.42	699873.86
8	100062119	Sub-Saharan Africa	Uganda	Office Suppli...	Online	L	2012-12-06	2012-12-26	9460	651.21	524.96	6160446.60	4966121.60	1194325...
9	100069415	Sub-Saharan Africa	Guinea	Clothes	Online	M	2012-01-19	2012-03-07	1619	109.28	35.84	176924.32	58024.96	118899.36
10	100077140	Europe	Albania	Personal Care	Online	C	2013-05-06	2013-05-26	3904	81.73	56.67	319073.92	221239.68	97834.24
11	100088727	North America	Canada	Household	Online	M	2011-07-02	2011-07-06	7333	668.27	502.54	4900423.91	3685125.82	1215298...
12	100089585	Sub-Saharan Africa	Mali	Office Suppli...	Online	C	2015-11-07	2015-12-24	7587	651.21	524.96	4940730.27	3982871.52	957858.75
13	100105893	Asia	Nepal	Clothes	Online	C	2015-03-21	2015-04-24	2413	109.28	35.84	263692.64	86481.92	177210.72
14	100106751	Europe	Iceland	Meat	Online	L	2011-12-30	2012-01-24	2667	421.89	364.69	1125180.63	972628.23	152552.40
15	100109755	Sub-Saharan Africa	Rwanda	Cereal	Online	L	2012-01-29	2012-03-15	3556	205.70	117.11	731469.20	416443.16	315026.04
16	100110614	Asia	Thailand	Snacks	Online	M	2016-06-06	2016-07-13	3810	152.58	97.44	581329.80	371246.40	210083.40
17	100114047	Middle East and North Africa	Somalia	Snacks	Online	H	2011-02-11	2011-02-13	4826	152.58	97.44	736351.08	470245.44	266105.64
18	100147092	Europe	Norway	Snacks	Offline	C	2012-01-09	2012-02-02	4605	152.58	97.44	702630.90	448711.20	253919.70
19	100155675	Middle East and North Africa	Saudi Arabia	Vegetables	Offline	H	2010-02-04	2010-03-15	7145	154.06	90.93	1100758.70	649694.85	451063.85
20	100165116	Europe	Belarus	Personal Care	Offline	H	2012-07-08	2012-08-22	9939	81.73	56.67	812314.47	563243.13	249071.34
21	100166833	Sub-Saharan Africa	Sao Tome and Principe	Cereal	Offline	L	2013-08-24	2013-09-20	447	205.70	117.11	91947.90	52348.17	39599.73
22	100176274	Sub-Saharan Africa	Zambia	Office Suppli...	Offline	L	2016-01-26	2016-02-28	3241	651.21	524.96	2110571.61	1701395.36	409176.25
23	100177562	Sub-Saharan Africa	Namibia	Meat	Offline	H	2015-01-09	2015-01-29	3622	421.89	364.69	1528085.58	1320907.18	207178.40

Query executed successfully. | DESKTOP-7SDKMH7\SQLEXPRESS ... | DESKTOP-7SDKMH7\LENOVO... | ECOMARTDATABASE | 00:00:02 | 100,000 rows

F3. Write script for three queries **to retrieve specific information from the database that will help to solve the identified business problem. Provide a screenshot showing the script for** each **query and** each **query successfully executed.**

Query 1: Total Profit by Region

Purpose: Identify which regions are generating the most profit. Helps in regional performance evaluation and strategic planning.



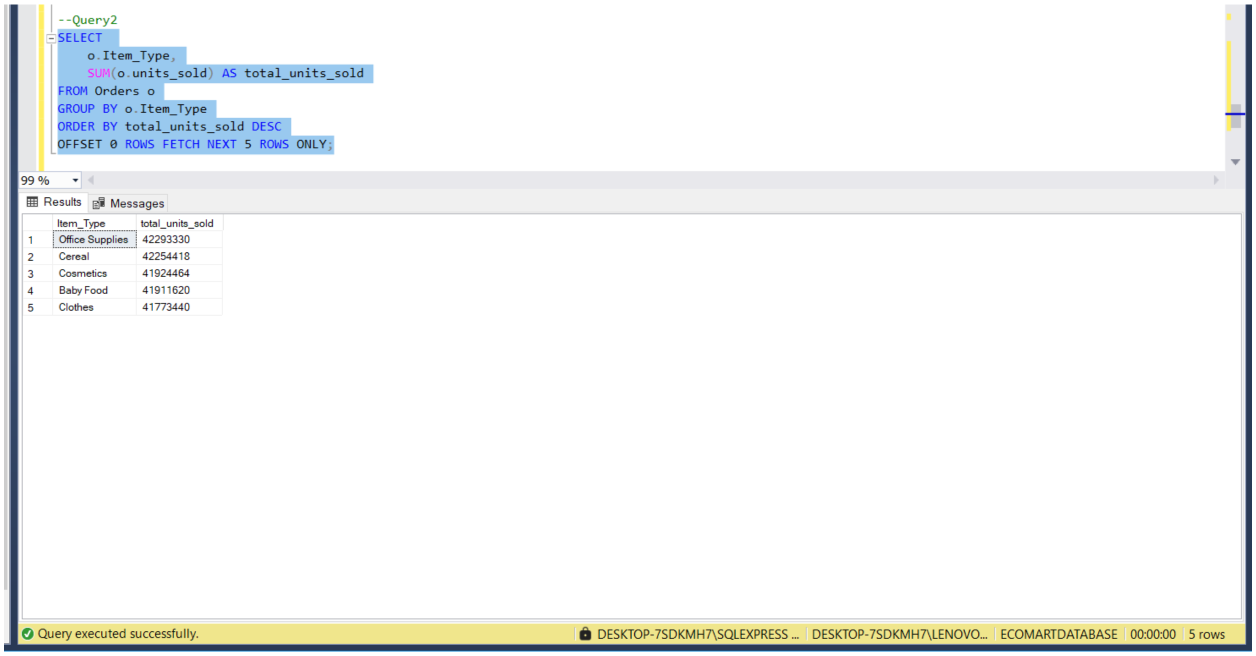
```sql
-- Query1
SELECT
    o.Region,
    SUM(o.total_profit) AS total_profit
FROM Orders o
GROUP BY o.Region
ORDER BY total_profit DESC;
```

	Region	total_profit
1	Sub-Saharan Africa	10306312642.23
2	Europe	10080579491.05
3	Asia	5707511516.76
4	Middle East and North Africa	4979534378.88
5	Central America and the Caribbean	4287210522.47
6	Australia and Oceania	3175423561.38
7	North America	872551616.84

```
Query executed successfully.    DESKTOP-7SDKMH7\SQLEXPRESS ...  DESKTOP-7SDKMH7\LENOVO...  ECOMARTDATABASE  00:00:00  7 rows
```

Query 2: Top 5 Selling Item Types by Units Sold

Purpose: Determine the best-selling products. Useful for inventory management and marketing focus.



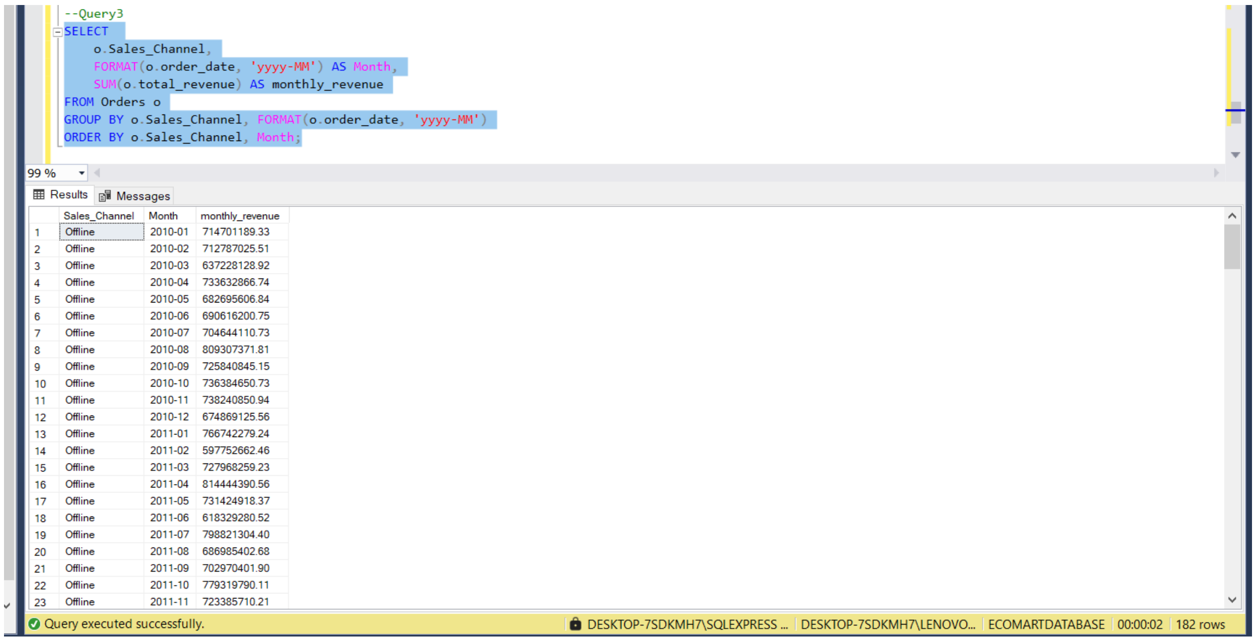
```sql
--Query2
SELECT
    o.Item_Type,
    SUM(o.units_sold) AS total_units_sold
FROM Orders o
GROUP BY o.Item_Type
ORDER BY total_units_sold DESC
OFFSET 0 ROWS FETCH NEXT 5 ROWS ONLY;
```

	Item_Type	total_units_sold
1	Office Supplies	42293330
2	Cereal	42254418
3	Cosmetics	41924464
4	Baby Food	41911620
5	Clothes	41773440

```
Query executed successfully.    DESKTOP-7SDKMH7\SQLEXPRESS ...  DESKTOP-7SDKMH7\LENOVO...  ECOMARTDATABASE  00:00:00  5 rows
```

Query 3: Monthly Revenue by Sales Channel

Purpose: Understand sales trends over time across different sales channels (Online vs. Offline)



```
--Query3
SELECT
    o.Sales_Channel,
    FORMAT(o.order_date, 'yyyy-MM') AS Month,
    SUM(o.total_revenue) AS monthly_revenue
FROM Orders o
GROUP BY o.Sales_Channel, FORMAT(o.order_date, 'yyyy-MM')
ORDER BY o.Sales_Channel, Month;
```

	Sales_Channel	Month	monthly_revenue
1	Offline	2010-01	714701189.33
2	Offline	2010-02	712787025.51
3	Offline	2010-03	637228128.92
4	Offline	2010-04	733632866.74
5	Offline	2010-05	682695606.84
6	Offline	2010-06	690616200.75
7	Offline	2010-07	704644110.73
8	Offline	2010-08	809307371.81
9	Offline	2010-09	725840845.15
10	Offline	2010-10	736384650.73
11	Offline	2010-11	738240850.94
12	Offline	2010-12	674869125.56
13	Offline	2011-01	766742279.24
14	Offline	2011-02	597752662.46
15	Offline	2011-03	727968259.23
16	Offline	2011-04	814444390.56
17	Offline	2011-05	731424918.37
18	Offline	2011-06	618329280.52
19	Offline	2011-07	798821304.40
20	Offline	2011-08	686985402.68
21	Offline	2011-09	702970401.90
22	Offline	2011-10	779319790.11
23	Offline	2011-11	723385710.21

```
Query executed successfully.    DESKTOP-7SDKMH7\SQLEXPRESS ... | DESKTOP-7SDKMH7\LENOVO... | ECOMARTDATABASE | 00:00:02 | 182 rows
```

F4. Apply optimization techniques to improve the run time of your queries from part F3, providing output results via a screenshot.

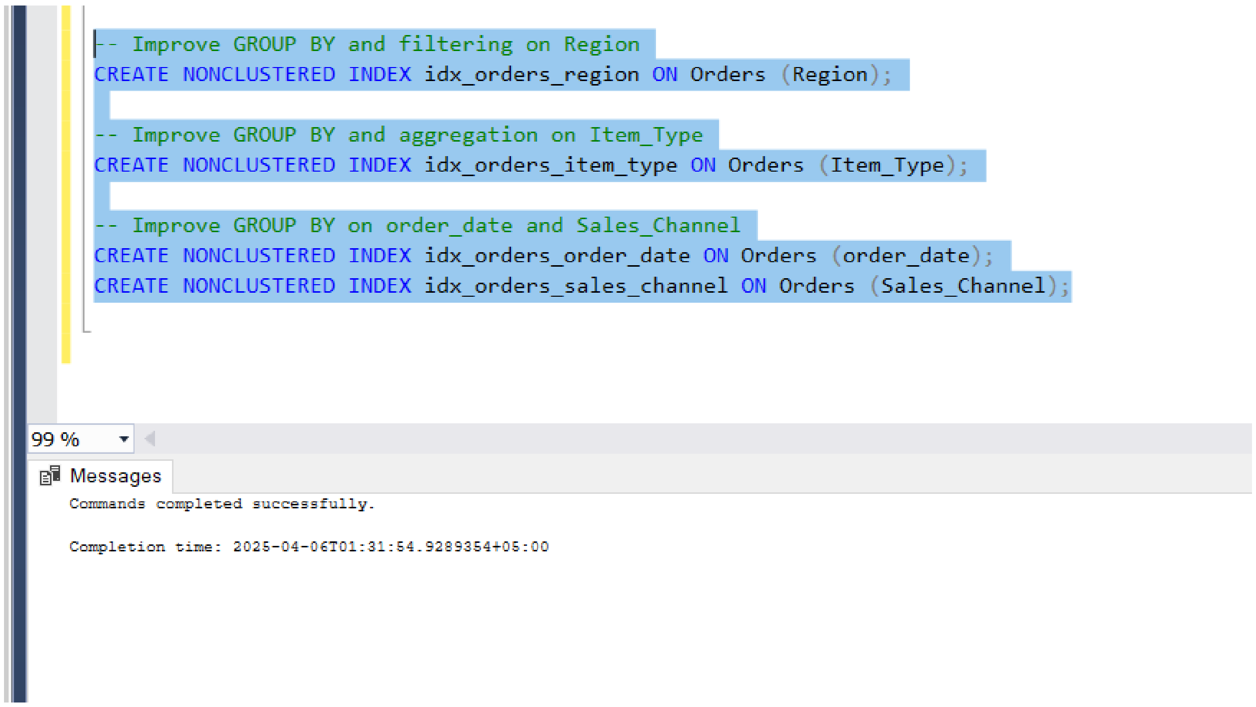
These use indexing recommendations and CTEs where helpful to improve performance in larger datasets.

Step 1: Add Indexes to Improve Query Performance

Based on your table structure, we'll optimize for the following:

- **Grouping** and **filtering** by fields like Region, Item_Type, order_date, and Sales_Channel
- **Aggregations** like SUM(units_sold) and SUM(total_profit)

Index Scripts

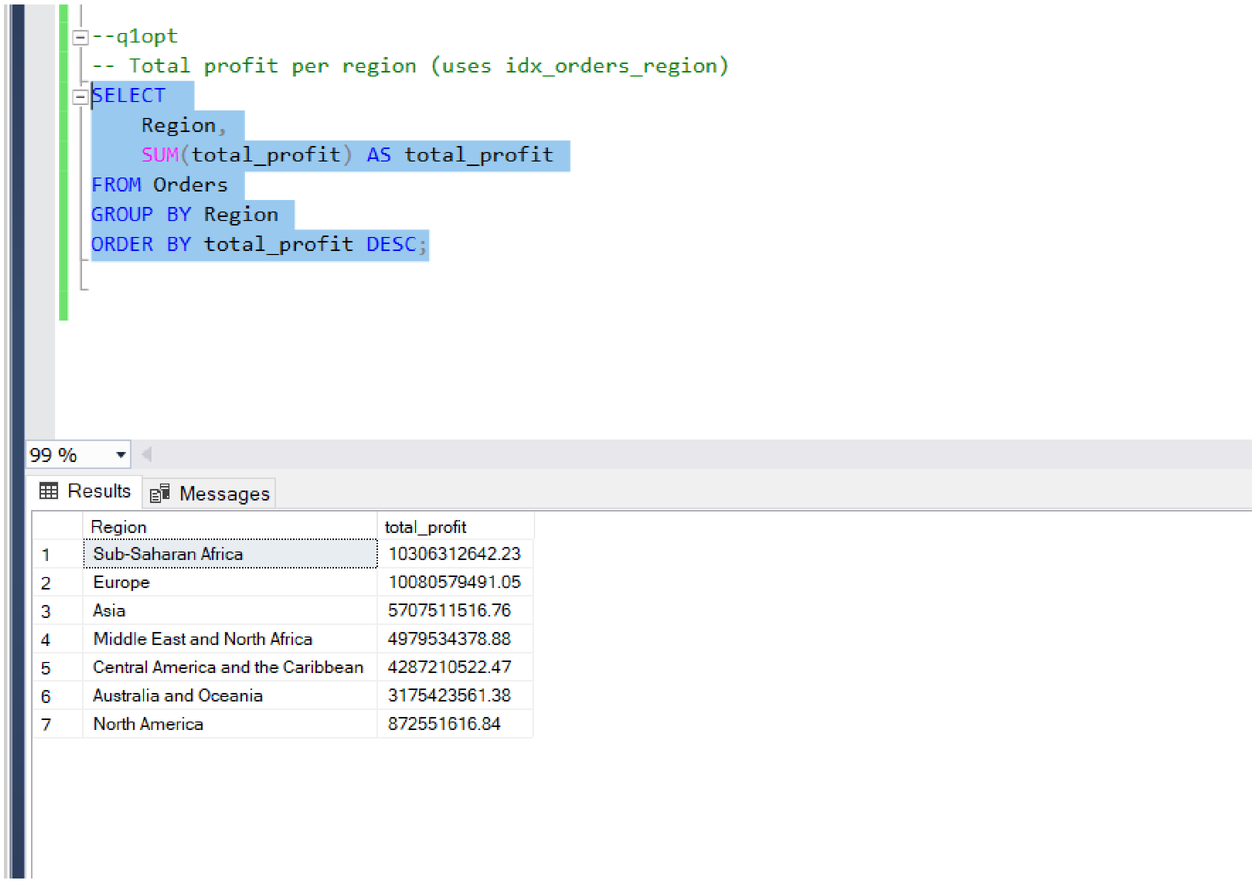


```
-- Improve GROUP BY and filtering on Region
CREATE NONCLUSTERED INDEX idx_orders_region ON Orders (Region);

-- Improve GROUP BY and aggregation on Item_Type
CREATE NONCLUSTERED INDEX idx_orders_item_type ON Orders (Item_Type);

-- Improve GROUP BY on order_date and Sales_Channel
CREATE NONCLUSTERED INDEX idx_orders_order_date ON Orders (order_date);
CREATE NONCLUSTERED INDEX idx_orders_sales_channel ON Orders (Sales_Channel);
```

```
Commands completed successfully.

Completion time: 2025-04-06T01:31:54.9289354+05:00
```

Step 2: Optimized Queries for F4

Optimized Query 1: Total Profit by Region

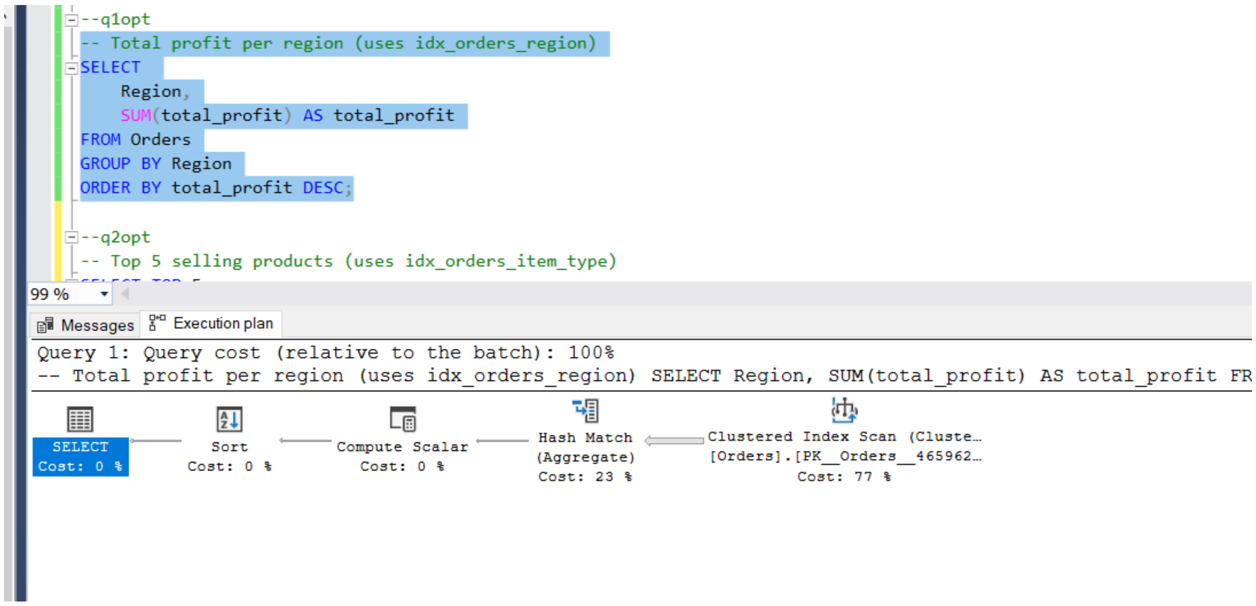


```sql
--q1opt
-- Total profit per region (uses idx_orders_region)
SELECT
    Region,
    SUM(total_profit) AS total_profit
FROM Orders
GROUP BY Region
ORDER BY total_profit DESC;
```

99 %

Results | Messages

	Region	total_profit
1	Sub-Saharan Africa	10306312642.23
2	Europe	10080579491.05
3	Asia	5707511516.76
4	Middle East and North Africa	4979534378.88
5	Central America and the Caribbean	4287210522.47
6	Australia and Oceania	3175423561.38
7	North America	872551616.84



```sql
--q1opt
-- Total profit per region (uses idx_orders_region)
SELECT
    Region,
    SUM(total_profit) AS total_profit
FROM Orders
GROUP BY Region
ORDER BY total_profit DESC;

--q2opt
-- Top 5 selling products (uses idx_orders_item_type)
```

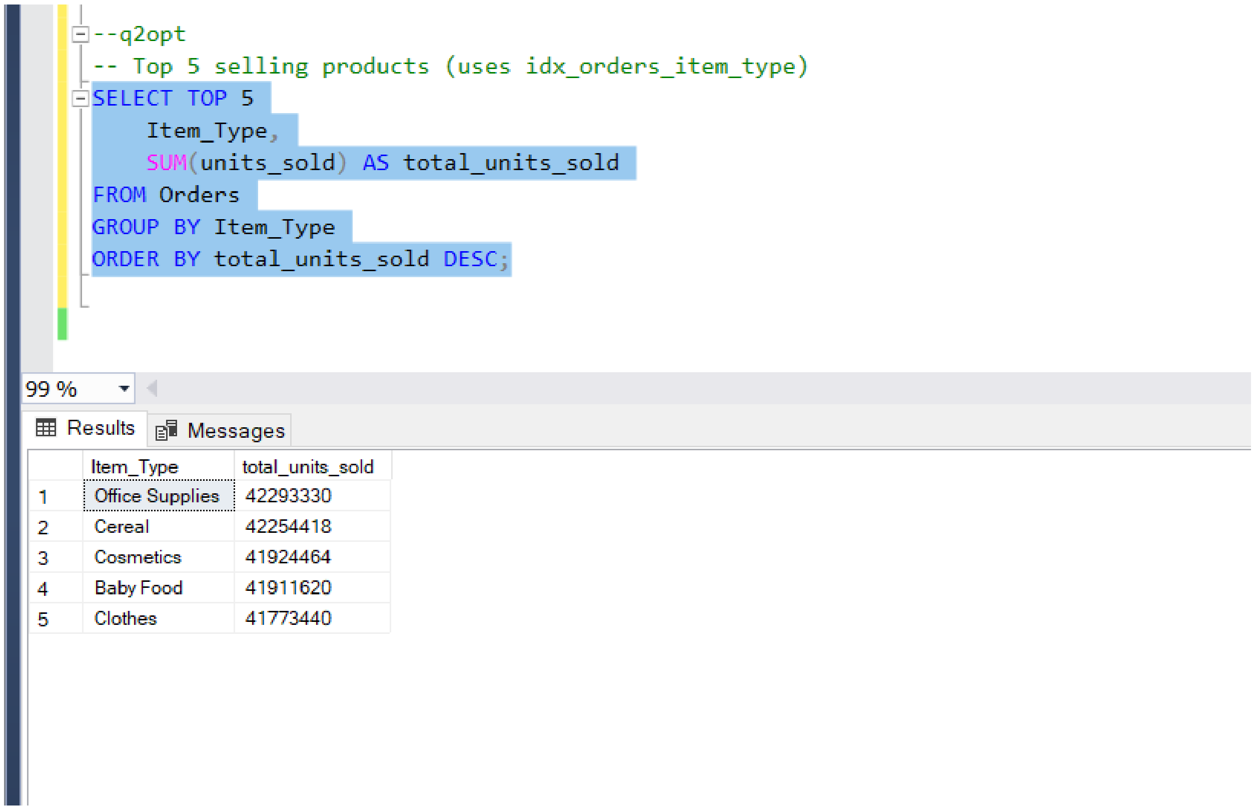
99 %

Messages | Execution plan

```
Query 1: Query cost (relative to the batch): 100%
-- Total profit per region (uses idx_orders_region) SELECT Region, SUM(total_profit) AS total_profit FR
```

Improvement: The index on Region makes the GROUP BY operation faster, especially with large datasets.

Optimized Query 2: Top 5 Selling Item Types by Units Sold

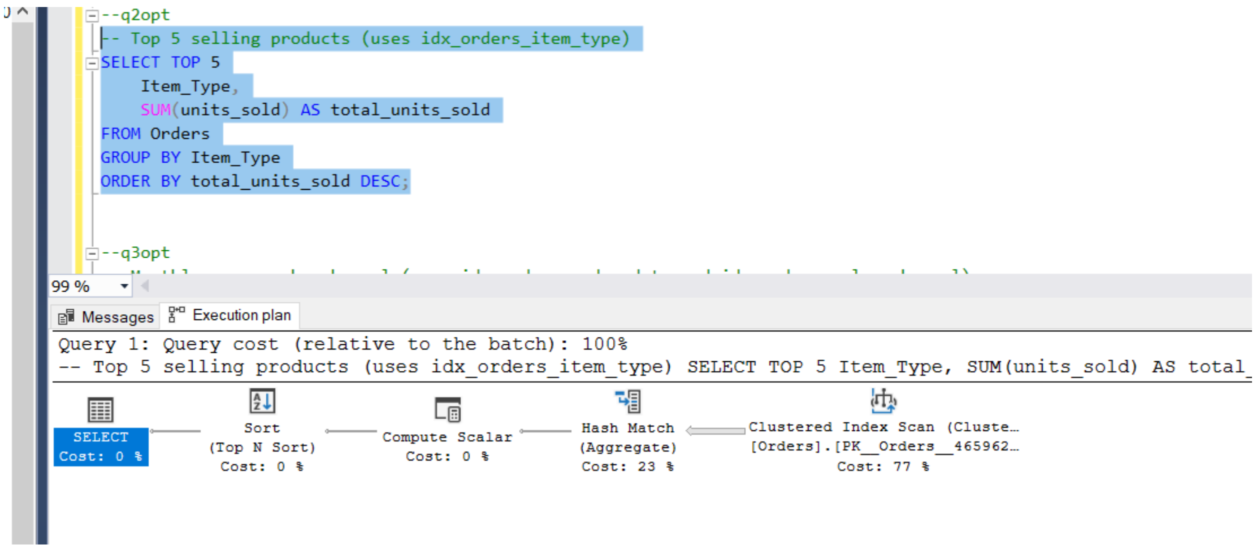


```sql
--q2opt
-- Top 5 selling products (uses idx_orders_item_type)
SELECT TOP 5
    Item_Type,
    SUM(units_sold) AS total_units_sold
FROM Orders
GROUP BY Item_Type
ORDER BY total_units_sold DESC;
```

99 %

Results | Messages

	Item_Type	total_units_sold
1	Office Supplies	42293330
2	Cereal	42254418
3	Cosmetics	41924464
4	Baby Food	41911620
5	Clothes	41773440



```sql
--q2opt
-- Top 5 selling products (uses idx_orders_item_type)
SELECT TOP 5
    Item_Type,
    SUM(units_sold) AS total_units_sold
FROM Orders
GROUP BY Item_Type
ORDER BY total_units_sold DESC;

--q3opt
```

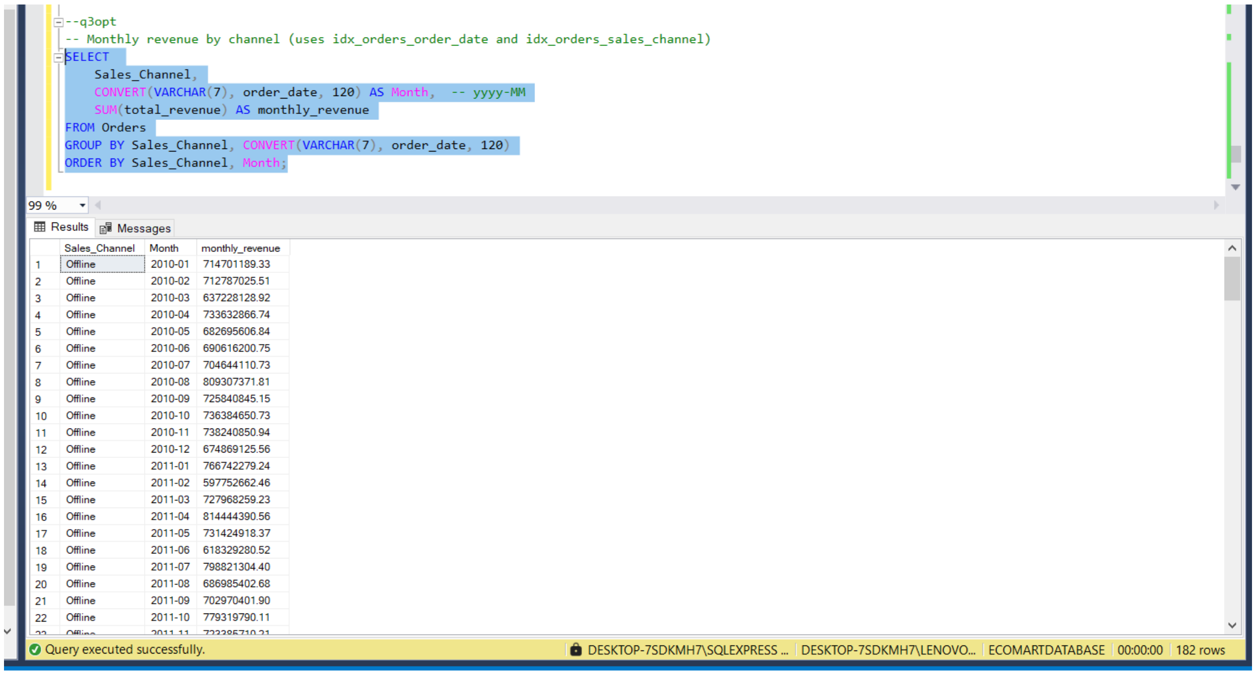
99 %

Messages | Execution plan

```
Query 1: Query cost (relative to the batch): 100%
-- Top 5 selling products (uses idx_orders_item_type) SELECT TOP 5 Item_Type, SUM(units_sold) AS total_

SELECT        Sort             Compute Scalar   Hash Match     Clustered Index Scan (Cluste…
Cost: 0 %     (Top N Sort)     Cost: 0 %        (Aggregate)    [Orders].[PK__Orders__465962…
              Cost: 0 %                         Cost: 23 %     Cost: 77 %
```

Improvement: Index on Item_Type allows faster access to relevant rows for grouping and sorting.

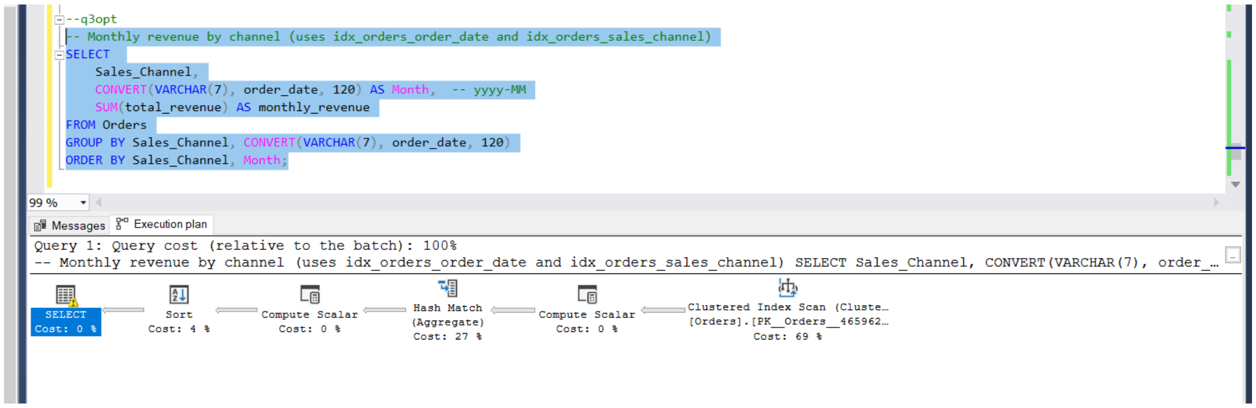
Optimized Query 3: Monthly Revenue by Sales Channel



```
--q3opt
-- Monthly revenue by channel (uses idx_orders_order_date and idx_orders_sales_channel)
SELECT
    Sales_Channel,
    CONVERT(VARCHAR(7), order_date, 120) AS Month,  -- yyyy-MM
    SUM(total_revenue) AS monthly_revenue
FROM Orders
GROUP BY Sales_Channel, CONVERT(VARCHAR(7), order_date, 120)
ORDER BY Sales_Channel, Month;
```

	Sales_Channel	Month	monthly_revenue
1	Offline	2010-01	714701189.33
2	Offline	2010-02	712787025.51
3	Offline	2010-03	637228128.92
4	Offline	2010-04	733632866.74
5	Offline	2010-05	682695606.84
6	Offline	2010-06	690616200.75
7	Offline	2010-07	704644110.73
8	Offline	2010-08	809307371.81
9	Offline	2010-09	725840845.15
10	Offline	2010-10	736384650.73
11	Offline	2010-11	738240850.94
12	Offline	2010-12	674869125.56
13	Offline	2011-01	766742279.24
14	Offline	2011-02	597752662.46
15	Offline	2011-03	727968259.23
16	Offline	2011-04	814444390.56
17	Offline	2011-05	731424918.37
18	Offline	2011-06	618329280.52
19	Offline	2011-07	798821304.40
20	Offline	2011-08	686985402.68
21	Offline	2011-09	702970401.90
22	Offline	2011-10	779319790.11
23	Offline	2011-11	723385710.21

Query executed successfully. DESKTOP-7SDKMH7\SQLEXPRESS ... DESKTOP-7SDKMH7\LENOVO... ECOMARTDATABASE 00:00:00 182 rows



```
--q3opt
-- Monthly revenue by channel (uses idx_orders_order_date and idx_orders_sales_channel)
SELECT
    Sales_Channel,
    CONVERT(VARCHAR(7), order_date, 120) AS Month,  -- yyyy-MM
    SUM(total_revenue) AS monthly_revenue
FROM Orders
GROUP BY Sales_Channel, CONVERT(VARCHAR(7), order_date, 120)
ORDER BY Sales_Channel, Month;
```

```
Query 1: Query cost (relative to the batch): 100%
-- Monthly revenue by channel (uses idx_orders_order_date and idx_orders_sales_channel) SELECT Sales_Channel, CONVERT(VARCHAR(7), order_…
```

Improvement: Indexes on order_date and Sales_Channel boost performance of GROUP BY and ORDER BY.

References

Coronel, C., & Morris, S. (2019). *Database systems: Design, implementation, & management* (13th ed.). Cengage Learning.

Kimball, R., & Ross, M. (2013). *The data warehouse toolkit: The definitive guide to dimensional modeling* (3rd ed.). Wiley.

Microsoft. (n.d.). *CREATE INDEX (Transact-SQL)*. Microsoft Learn. https://learn.microsoft.com/en-us/sql/t-sql/statements/create-index-transact-sql

Western Governors University. (2024). *D597 data management – applications course material*. Western Governors University.