Brittany Hancock
IT 420: Business Intelligence
Instructor: Lenore Montalbano
OLTP vs OLAP Client Consulting Report

Introduction: OLTP systems handle daily transactions and store detailed data that changes often. These databases capture raw information used later by OLAP systems. OLAP, on the other hand, gathers that transactional data in batches and organizes it for analysis. While OLTP focuses on accuracy and real-time updates, OLAP is designed for quick access when creating reports, dashboards, and charts.

Design differences: A normalized OLTP design keeps each piece of data in its own field, for instance, separating a customer's first and last name. OLAP systems use a denormalized approach, combining related details to make reports easier to build. Instead of calculating totals each time a query runs, OLAP stores results like total sales directly in the table, which saves time later. In contrast, OLTP must calculate those figures during each query, slowing performance as the database grows.

OLAP structures are built for large-scale analysis, while OLTP systems ensure reliable transactions. In a star schema, OLAP saves processed data for reuse, and each record has a surrogate key that allows quick, indexed joins between tables. For example, a CustomerKey in the dimension table connects directly to a matching CustomerKey in the fact table. A fully normalized OLTP system does not use this type of relationship.

Performance comparison: See Figures 1 through 14 in the Appendix for query results, ETL validation, and schema performance examples.

Business risks: Running analytical queries on an OLTP database can slow it down because it must repeatedly group and calculate results. These systems rely on multiple joins and aggregations, whereas OLAP stores summarized and indexed data ready for use. As the amount of information grows, OLTP performance drops and can disrupt regular operations.

Recommendation: OLAP offers a more efficient structure for analytics, reporting, and visualization. It supports high-volume summaries and scales smoothly as data expands. Surrogate keys make relationships between tables faster to query. Because OLAP organizes and pre-calculates data in advance, it provides accurate, dependable results for business planning and decision making.

Appendix

Figure 1. SSIS Control Flow Overview: displays the full ETL process from staging to warehouse, including data flow execution and truncation steps for both environments.

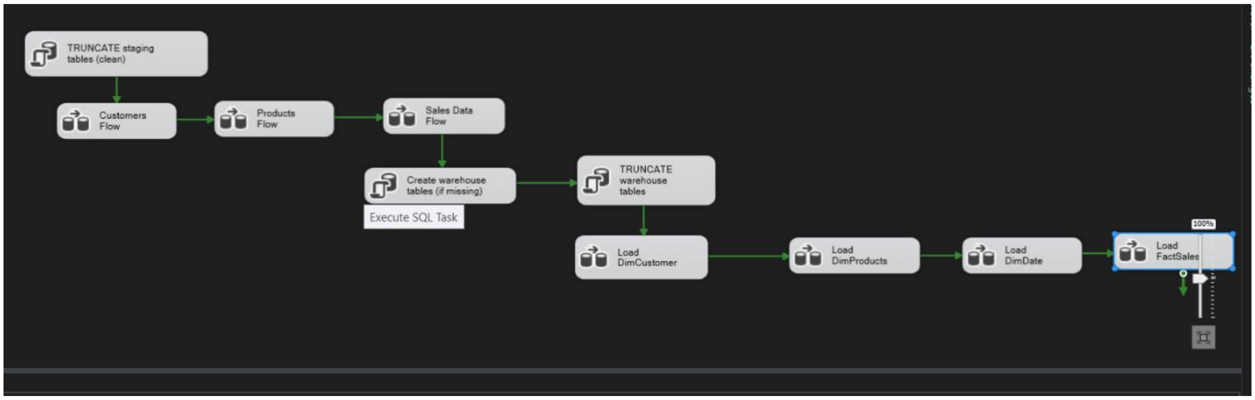
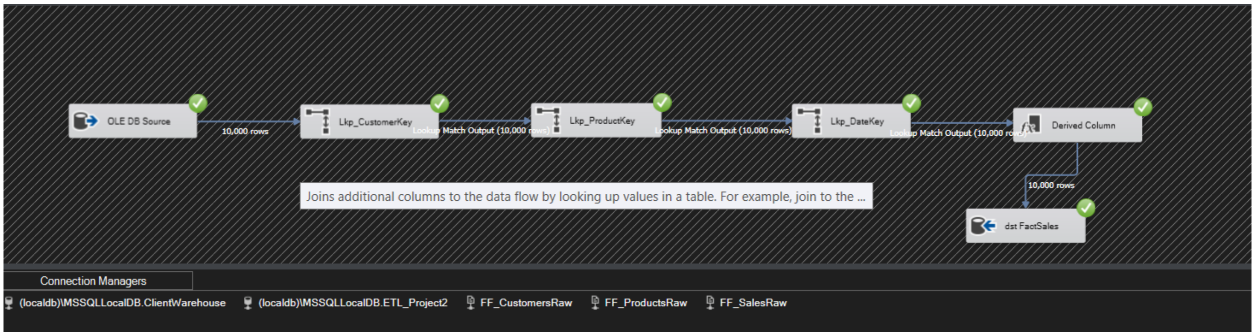


Figure 2. SSIS Data Flow For FactSales: shows the transformation pipeline where surrogate keys are looked up, derived columns are calculated, and clean data is loaded into the FactSales table.


OLE DB Source
10,000 rows
Lkp_CustomerKey
Lookup Match Output (10,000 rows)
Lkp_ProductKey
Lookup Match Output (10,000 rows)
Lkp_DateKey
Lookup Match Output (10,000 rows)
Derived Column
10,000 rows
dst FactSales
Connection Managers
(localdb)\MSSQLLocalDB.ClientWarehouse
(localdb)\MSSQLLocalDB.ETL_Project2
FF_CustomersRaw
FF_ProductsRaw
FF_SalesRaw

Joins additional columns to the data flow by looking up values in a table. For example, join to the ...

Figure 3. SSIS Package Successful Execution: indicates the ETL workflow ran successfully from start to finish with all data flow tasks completed.

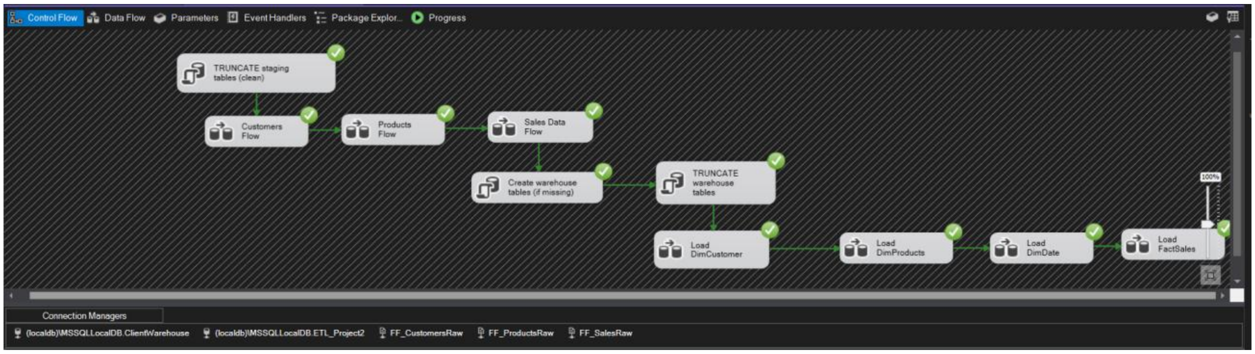
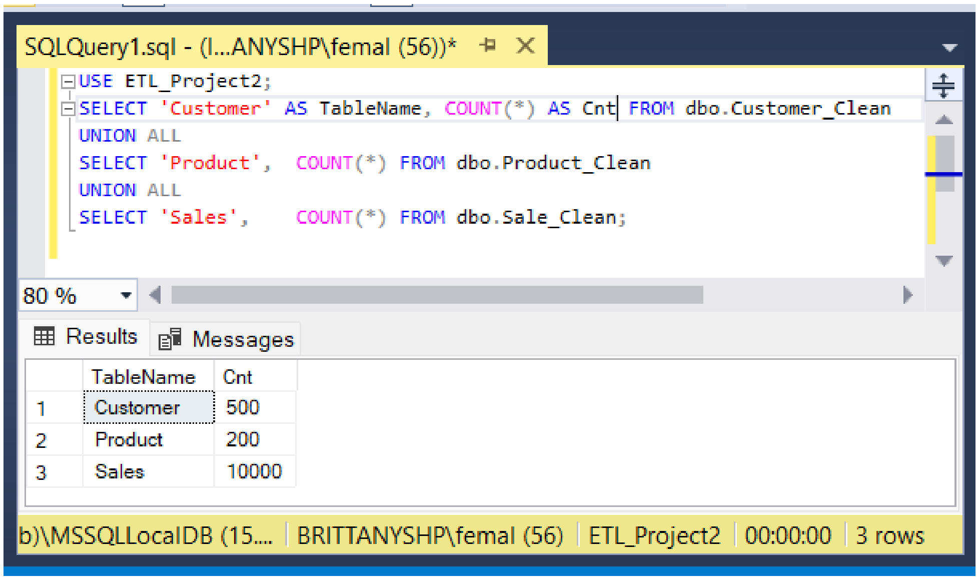


Figure 4. Staging Database Row Counts (ETL_Project2): row counts from the cleaned staging tables confirm successful data import: 500 customers, 200 products, and 10,000 sales records.

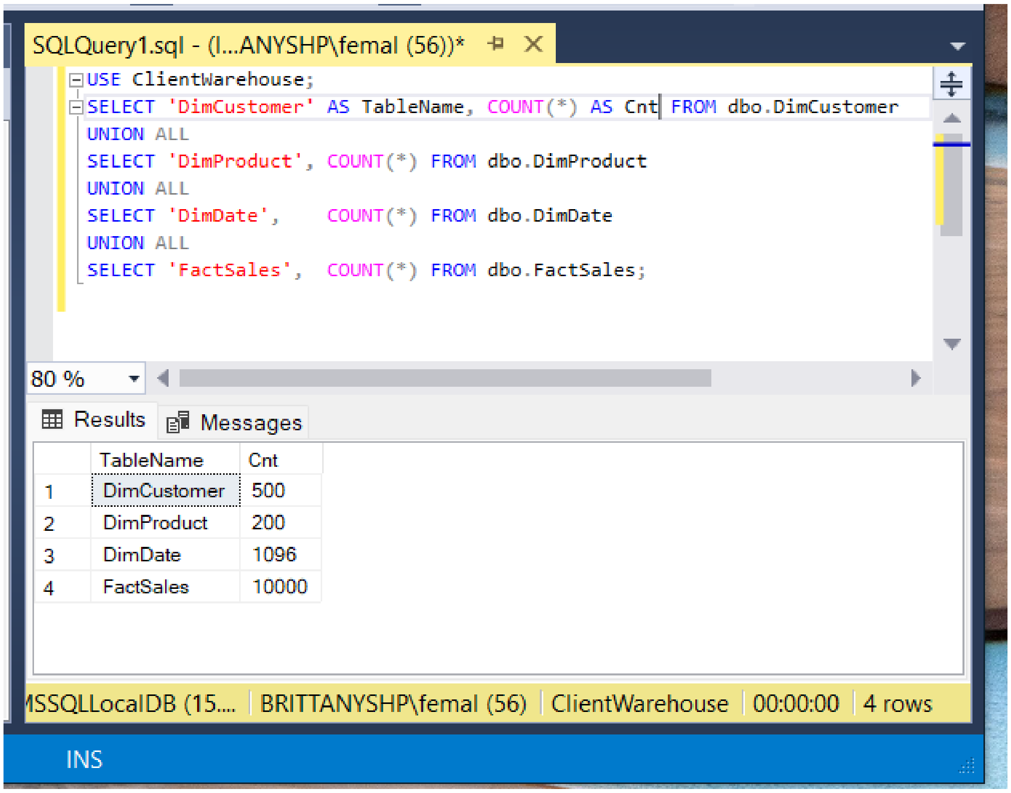


```sql
USE ETL_Project2;
SELECT 'Customer' AS TableName, COUNT(*) AS Cnt FROM dbo.Customer_Clean
UNION ALL
SELECT 'Product',  COUNT(*) FROM dbo.Product_Clean
UNION ALL
SELECT 'Sales',    COUNT(*) FROM dbo.Sale_Clean;
```

	TableName	Cnt
1	Customer	500
2	Product	200
3	Sales	10000

b)\MSSQLLocalDB (15.... | BRITTANYSHP\femal (56) | ETL_Project2 | 00:00:00 | 3 rows

Figure 5. Warehouse Database Row Counts (ClientWarehouse): row counts in dimension and fact tables confirm successful ETL transfer and matching totals across both environments.

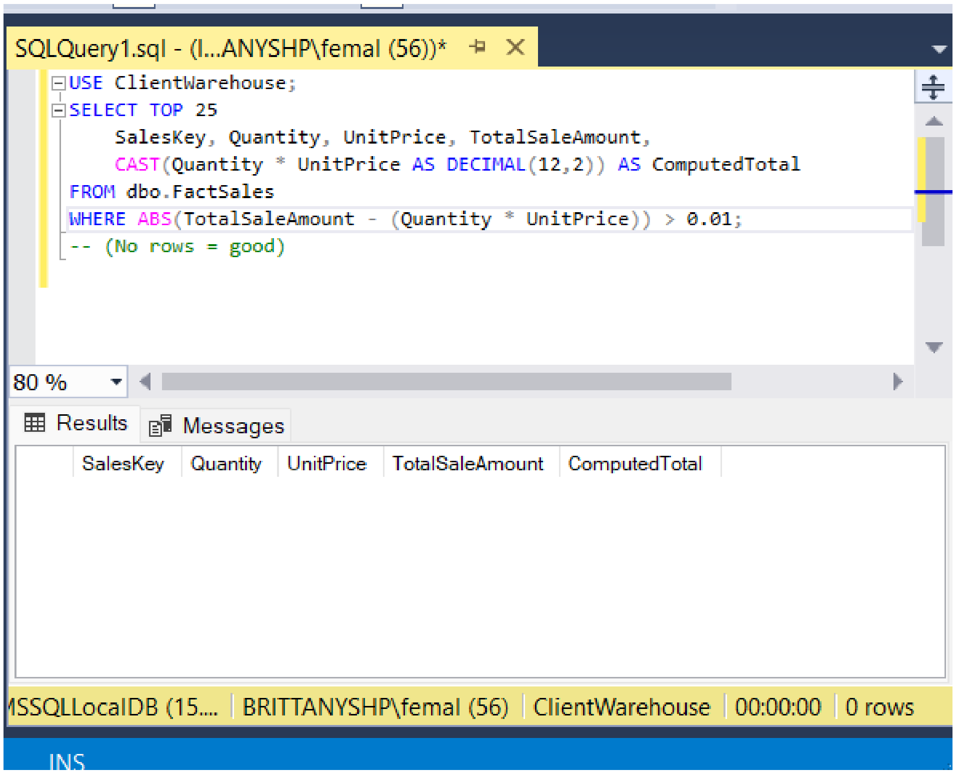


```sql
USE ClientWarehouse;
SELECT 'DimCustomer' AS TableName, COUNT(*) AS Cnt FROM dbo.DimCustomer
UNION ALL
SELECT 'DimProduct', COUNT(*) FROM dbo.DimProduct
UNION ALL
SELECT 'DimDate',    COUNT(*) FROM dbo.DimDate
UNION ALL
SELECT 'FactSales',  COUNT(*) FROM dbo.FactSales;
```

	TableName	Cnt
1	DimCustomer	500
2	DimProduct	200
3	DimDate	1096
4	FactSales	10000

MSSQLLocalDB (15.... | BRITTANYSHP\femal (56) | ClientWarehouse | 00:00:00 | 4 rows

Figure 6. TotalSaleAmount Validation Query: verification query checks consistency between computed totals and stored TotalSaleAmount values, confirming data accuracy (no mismatches found).

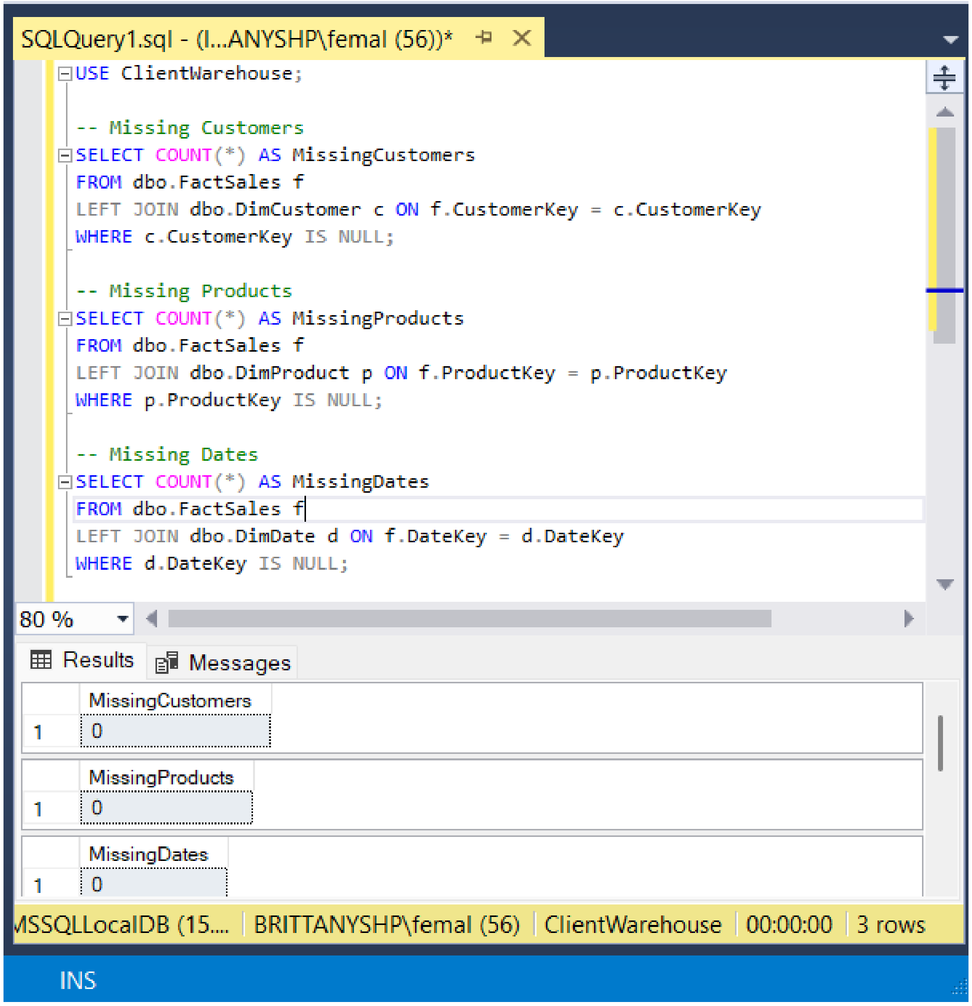


```sql
USE ClientWarehouse;
SELECT TOP 25
    SalesKey, Quantity, UnitPrice, TotalSaleAmount,
    CAST(Quantity * UnitPrice AS DECIMAL(12,2)) AS ComputedTotal
FROM dbo.FactSales
WHERE ABS(TotalSaleAmount - (Quantity * UnitPrice)) > 0.01;
-- (No rows = good)
```

	SalesKey	Quantity	UnitPrice	TotalSaleAmount	ComputedTotal

ISSQLLocalDB (15.... | BRITTANYSHP\femal (56) | ClientWarehouse | 00:00:00 | 0 rows

Figure 7. Refertial Integrity Validation Query: tests for missing Customer, Product, and Date keys in FactSales, confirming referential integrity with zero missing values.



```
USE ClientWarehouse;

-- Missing Customers
SELECT COUNT(*) AS MissingCustomers
FROM dbo.FactSales f
LEFT JOIN dbo.DimCustomer c ON f.CustomerKey = c.CustomerKey
WHERE c.CustomerKey IS NULL;

-- Missing Products
SELECT COUNT(*) AS MissingProducts
FROM dbo.FactSales f
LEFT JOIN dbo.DimProduct p ON f.ProductKey = p.ProductKey
WHERE p.ProductKey IS NULL;

-- Missing Dates
SELECT COUNT(*) AS MissingDates
FROM dbo.FactSales f
LEFT JOIN dbo.DimDate d ON f.DateKey = d.DateKey
WHERE d.DateKey IS NULL;
```

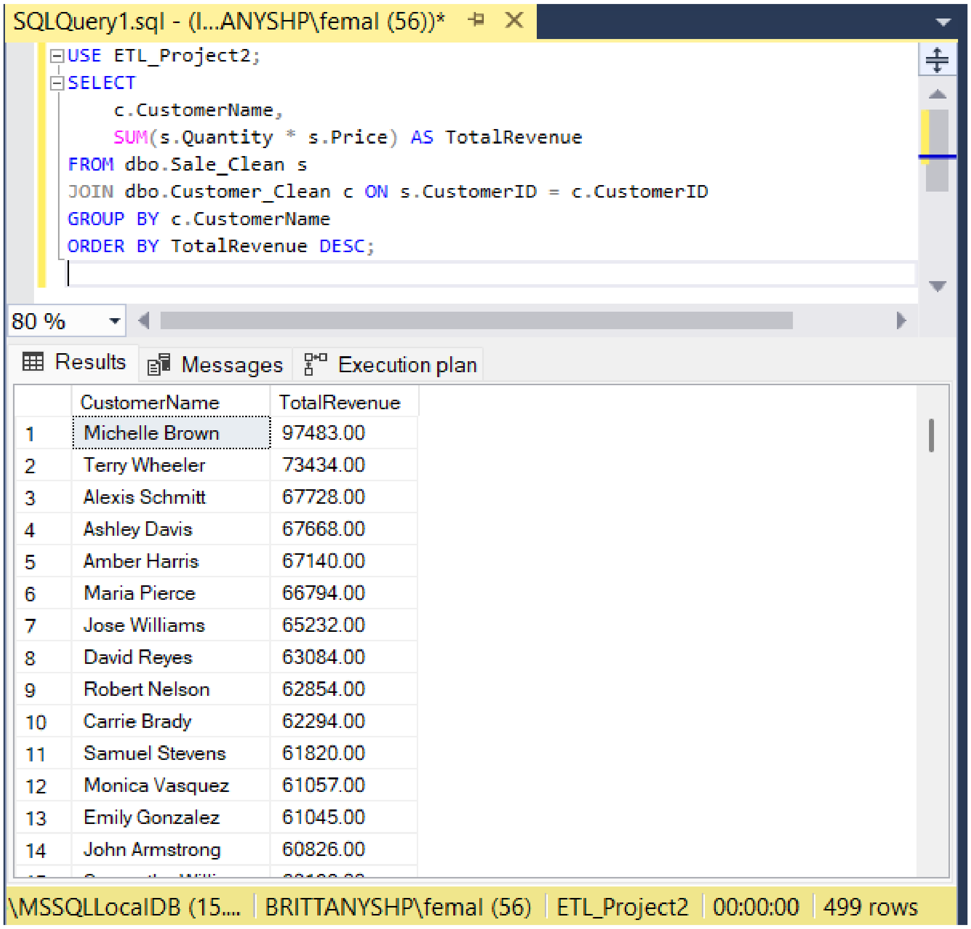
	MissingCustomers
1	0

	MissingProducts
1	0

	MissingDates
1	0

```
MSSQLLocalDB (15.... | BRITTANYSHP\femal (56) | ClientWarehouse | 00:00:00 | 3 rows
```

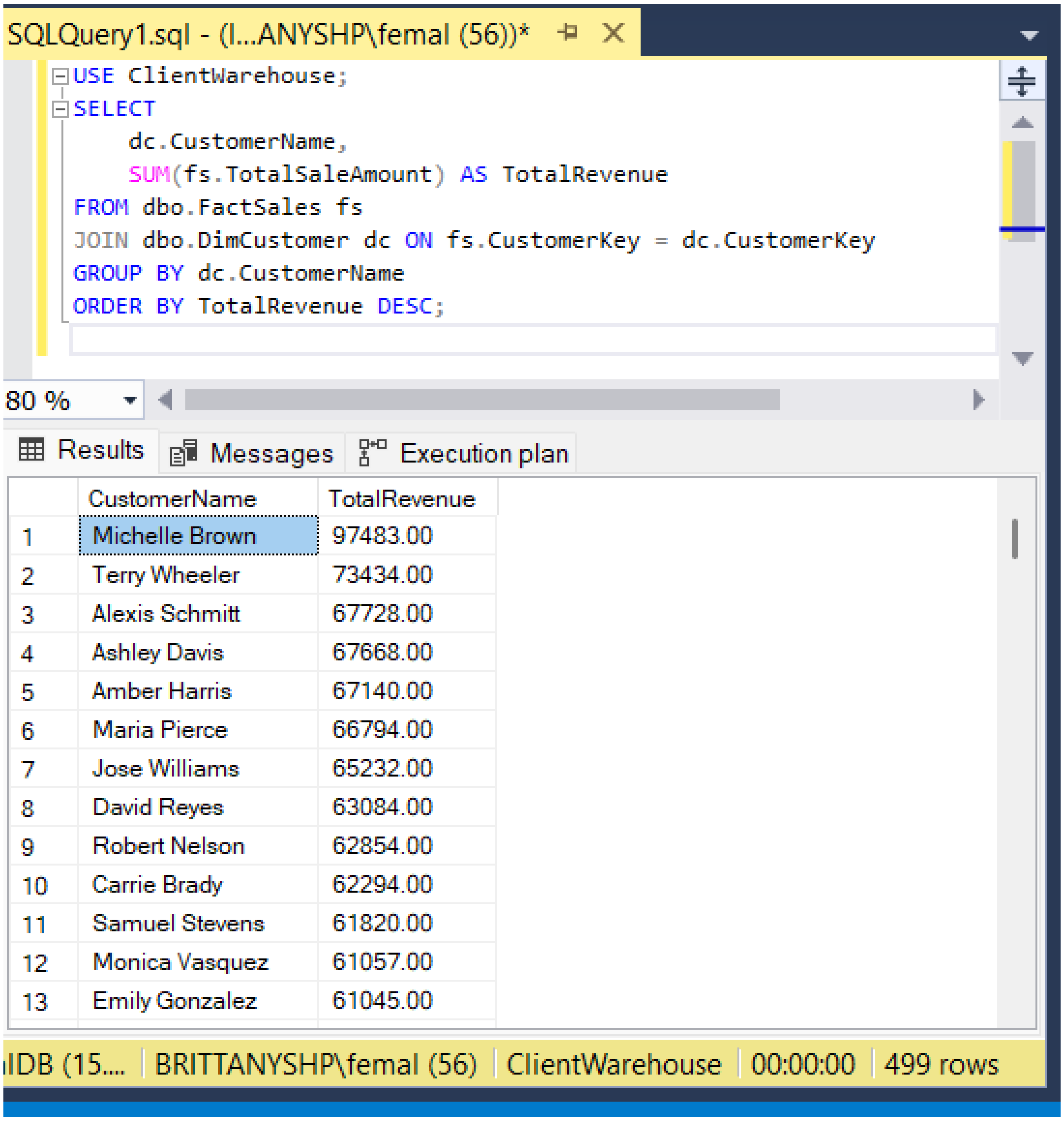
Figure 8. OLTP Query - Total Revenue by Customer: example of OLTP query aggregating total revenue by customer in the transactional database (ETL_Project2).



```sql
USE ETL_Project2;
SELECT
    c.CustomerName,
    SUM(s.Quantity * s.Price) AS TotalRevenue
FROM dbo.Sale_Clean s
JOIN dbo.Customer_Clean c ON s.CustomerID = c.CustomerID
GROUP BY c.CustomerName
ORDER BY TotalRevenue DESC;
```

	CustomerName	TotalRevenue
1	Michelle Brown	97483.00
2	Terry Wheeler	73434.00
3	Alexis Schmitt	67728.00
4	Ashley Davis	67668.00
5	Amber Harris	67140.00
6	Maria Pierce	66794.00
7	Jose Williams	65232.00
8	David Reyes	63084.00
9	Robert Nelson	62854.00
10	Carrie Brady	62294.00
11	Samuel Stevens	61820.00
12	Monica Vasquez	61057.00
13	Emily Gonzalez	61045.00
14	John Armstrong	60826.00

Figure 9. OLAP Query - Total Revenue by Customer: equivalent OLAP query aggregating pre-calculated TotalSaleAmount by customer in the warehouse (ClientWarehouse).

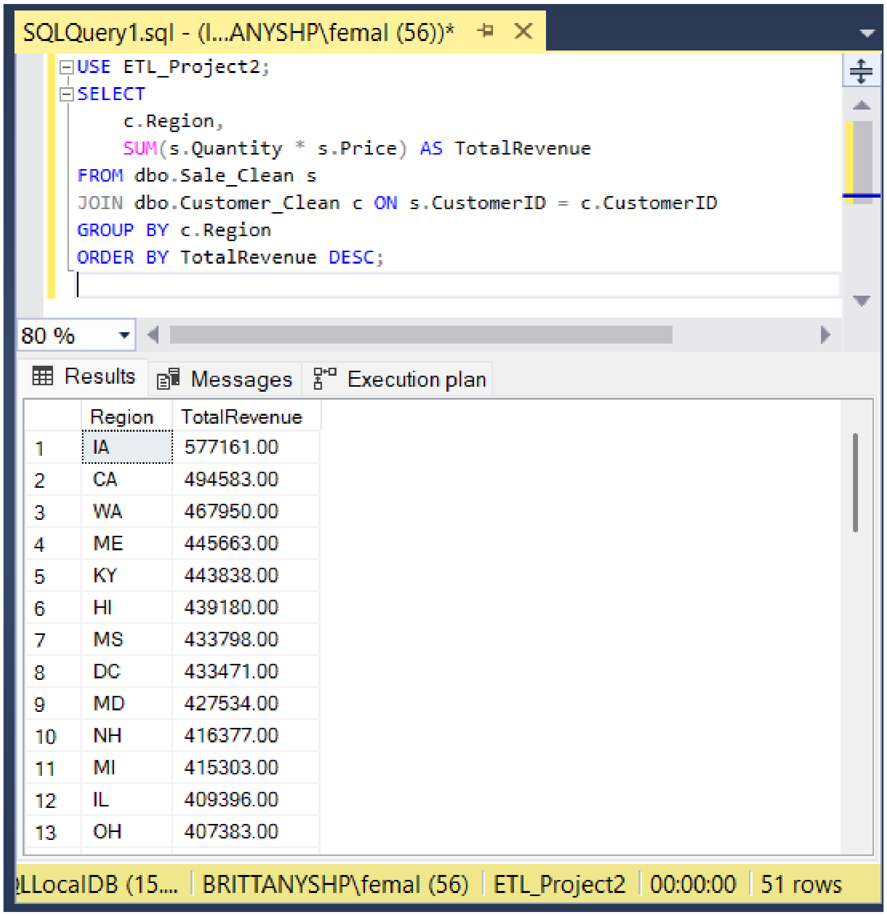


```sql
USE ClientWarehouse;
SELECT
    dc.CustomerName,
    SUM(fs.TotalSaleAmount) AS TotalRevenue
FROM dbo.FactSales fs
JOIN dbo.DimCustomer dc ON fs.CustomerKey = dc.CustomerKey
GROUP BY dc.CustomerName
ORDER BY TotalRevenue DESC;
```

	CustomerName	TotalRevenue
1	Michelle Brown	97483.00
2	Terry Wheeler	73434.00
3	Alexis Schmitt	67728.00
4	Ashley Davis	67668.00
5	Amber Harris	67140.00
6	Maria Pierce	66794.00
7	Jose Williams	65232.00
8	David Reyes	63084.00
9	Robert Nelson	62854.00
10	Carrie Brady	62294.00
11	Samuel Stevens	61820.00
12	Monica Vasquez	61057.00
13	Emily Gonzalez	61045.00

lDB (15.... | BRITTANYSHP\femal (56) | ClientWarehouse | 00:00:00 | 499 rows

Figure 10. OLTP Query - Total Revenue by Region: demonstrates regional revenue analysis on normalized OLTP data, requiring live aggregation and joins.

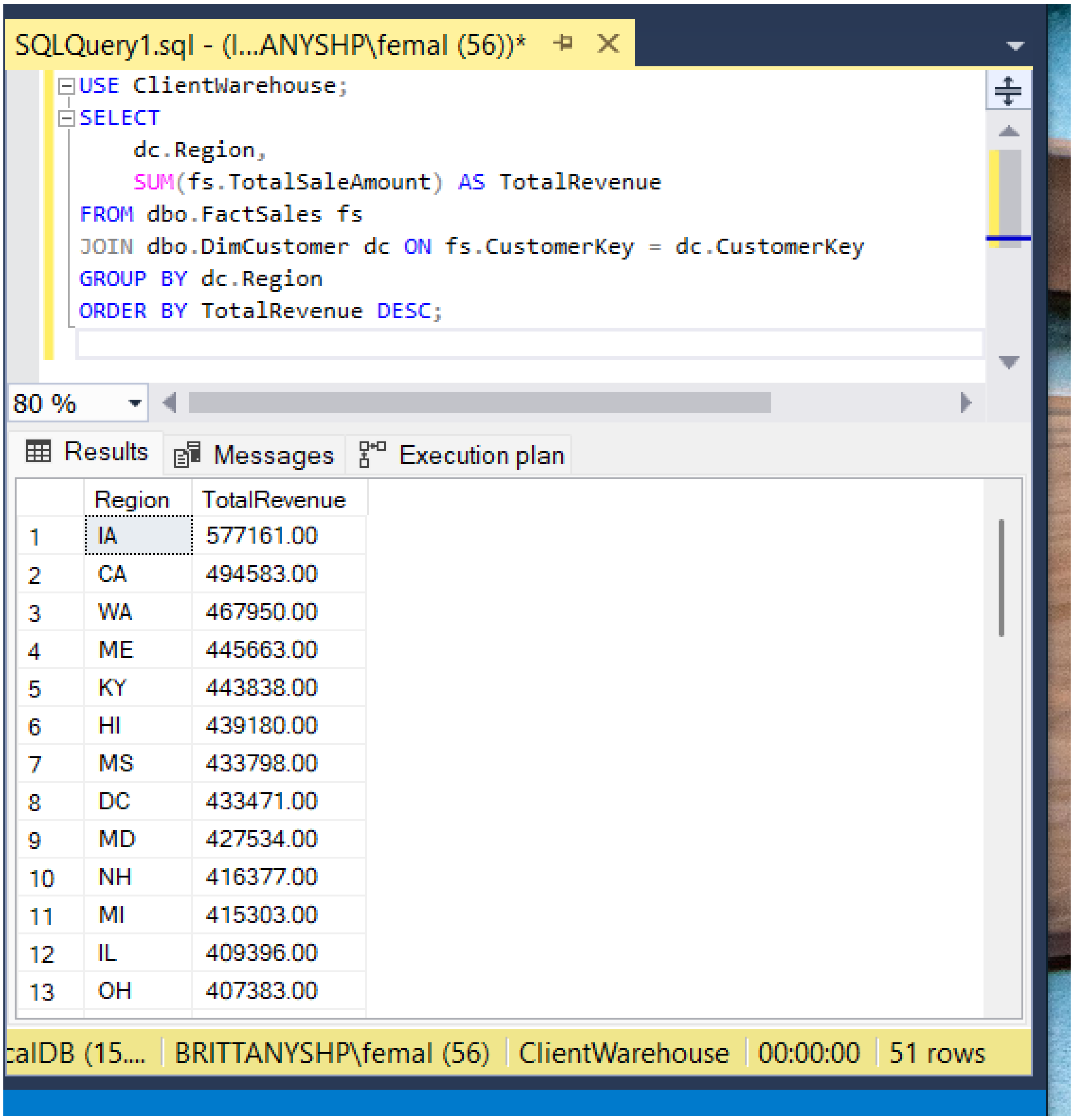


```sql
USE ETL_Project2;
SELECT
    c.Region,
    SUM(s.Quantity * s.Price) AS TotalRevenue
FROM dbo.Sale_Clean s
JOIN dbo.Customer_Clean c ON s.CustomerID = c.CustomerID
GROUP BY c.Region
ORDER BY TotalRevenue DESC;
```

	Region	TotalRevenue
1	IA	577161.00
2	CA	494583.00
3	WA	467950.00
4	ME	445663.00
5	KY	443838.00
6	HI	439180.00
7	MS	433798.00
8	DC	433471.00
9	MD	427534.00
10	NH	416377.00
11	MI	415303.00
12	IL	409396.00
13	OH	407383.00

LLocalDB (15.... | BRITTANYSHP\femal (56) | ETL_Project2 | 00:00:00 | 51 rows

Figure 11. OLAP Query - Total Revenue by Region: aggregrates pre-calculated TotalSaleAmount by region in the warehouse (ClientWarehouse), showing faster performance compated to OLTP query.

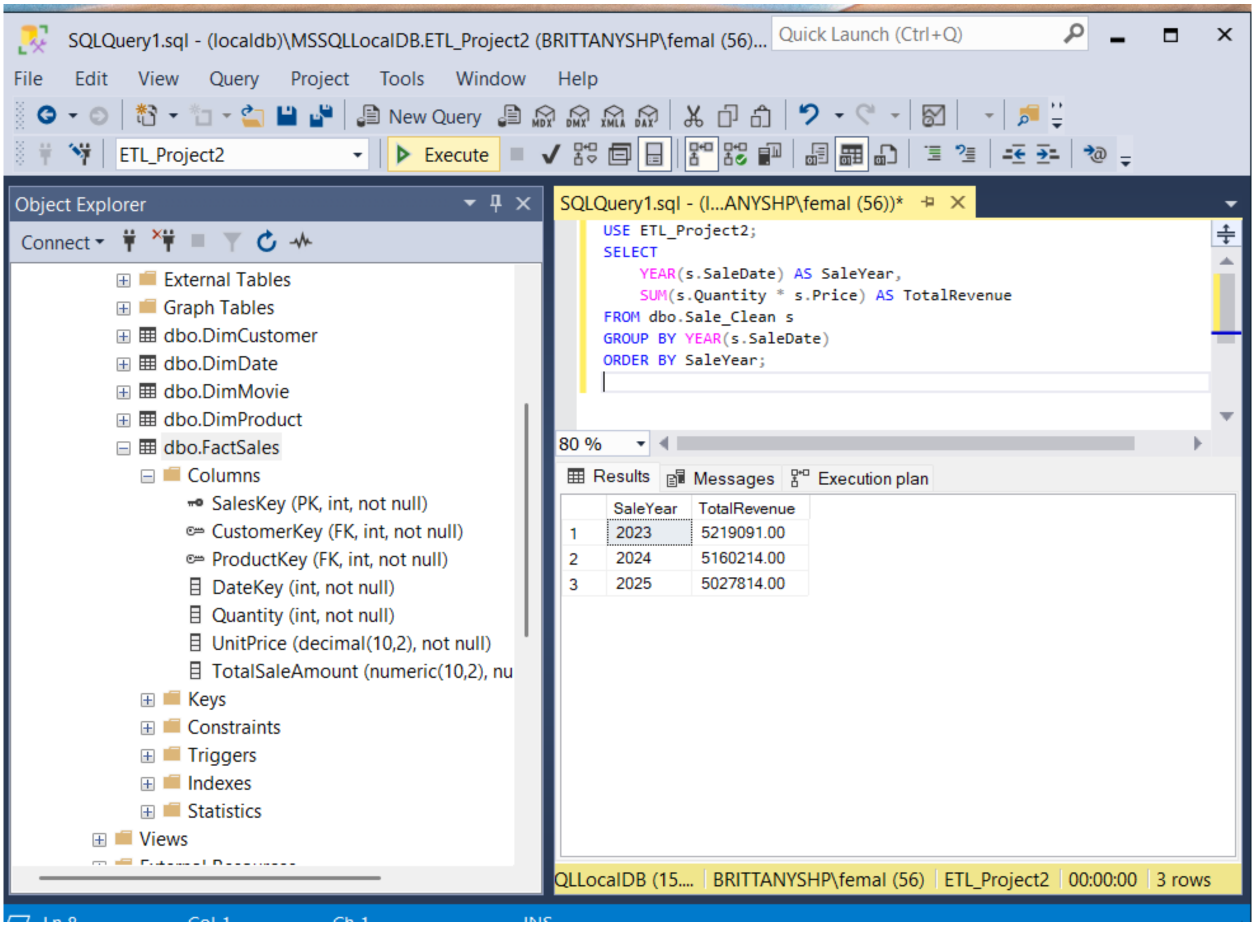


```
USE ClientWarehouse;
SELECT
    dc.Region,
    SUM(fs.TotalSaleAmount) AS TotalRevenue
FROM dbo.FactSales fs
JOIN dbo.DimCustomer dc ON fs.CustomerKey = dc.CustomerKey
GROUP BY dc.Region
ORDER BY TotalRevenue DESC;
```

	Region	TotalRevenue
1	IA	577161.00
2	CA	494583.00
3	WA	467950.00
4	ME	445663.00
5	KY	443838.00
6	HI	439180.00
7	MS	433798.00
8	DC	433471.00
9	MD	427534.00
10	NH	416377.00
11	MI	415303.00
12	IL	409396.00
13	OH	407383.00

calDB (15.... | BRITTANYSHP\femal (56) | ClientWarehouse | 00:00:00 | 51 rows

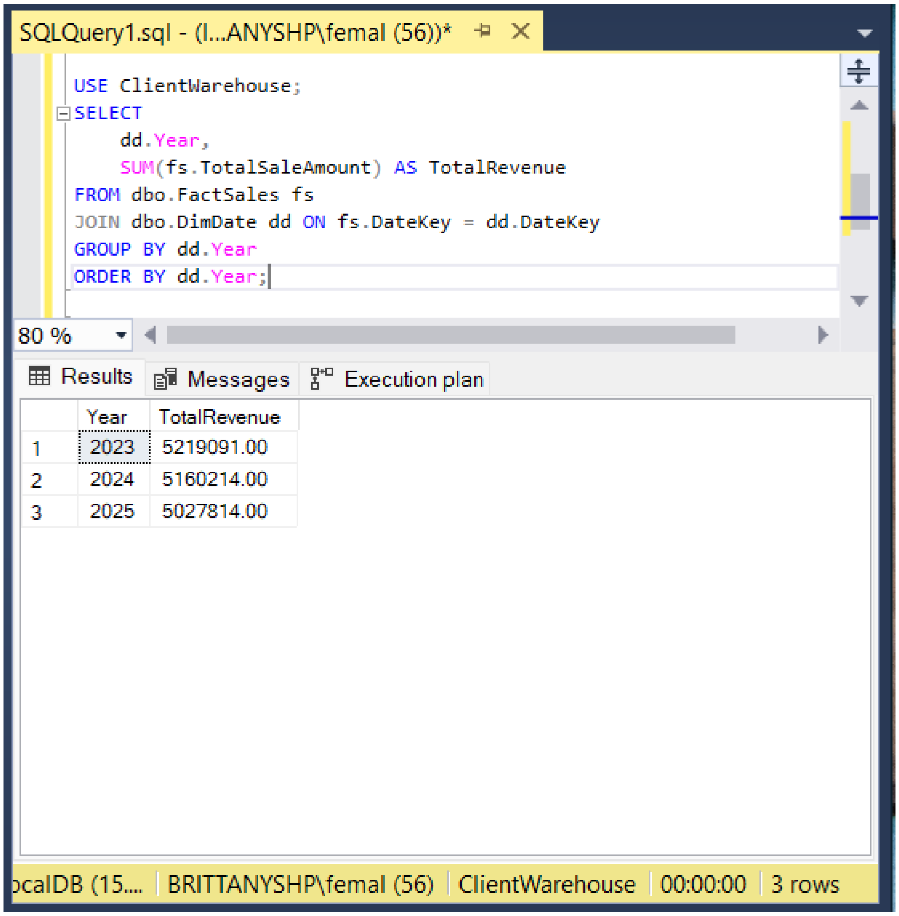
Figure 12. OLTP Query - Total Revenue by Year: displays yearly revenue totals computed in real time from the transactional database (ETL_Project2) using live aggregation.


SQLQuery1.sql - (localdb)\MSSQLLocalDB.ETL_Project2 (BRITTANYSHP\femal (56))...
Quick Launch (Ctrl+Q)
File
Edit
View
Query
Project
Tools
Window
Help
New Query
ETL_Project2
Execute
Object Explorer
Connect
External Tables
Graph Tables
dbo.DimCustomer
dbo.DimDate
dbo.DimMovie
dbo.DimProduct
dbo.FactSales
Columns
SalesKey (PK, int, not null)
CustomerKey (FK, int, not null)
ProductKey (FK, int, not null)
DateKey (int, not null)
Quantity (int, not null)
UnitPrice (decimal(10,2), not null)
TotalSaleAmount (numeric(10,2), nu
Keys
Constraints
Triggers
Indexes
Statistics
Views
SQLQuery1.sql - (l...ANYSHP\femal (56))*
80 %
Results
Messages
Execution plan
QLLocalDB (15.... | BRITTANYSHP\femal (56) | ETL_Project2 | 00:00:00 | 3 rows

```sql
USE ETL_Project2;
SELECT
    YEAR(s.SaleDate) AS SaleYear,
    SUM(s.Quantity * s.Price) AS TotalRevenue
FROM dbo.Sale_Clean s
GROUP BY YEAR(s.SaleDate)
ORDER BY SaleYear;
```

	SaleYear	TotalRevenue
1	2023	5219091.00
2	2024	5160214.00
3	2025	5027814.00

Figure 13. OLAP Query - Total Revenue by Year: shows the same yearly revenue analysis executed in the warehouse (ClientWarehouse), using pre-aggregated data for optimized query speed.

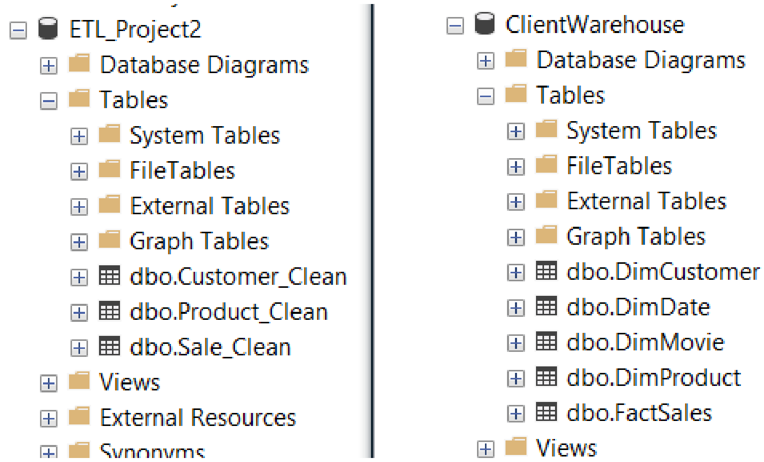


```sql
USE ClientWarehouse;
SELECT
    dd.Year,
    SUM(fs.TotalSaleAmount) AS TotalRevenue
FROM dbo.FactSales fs
JOIN dbo.DimDate dd ON fs.DateKey = dd.DateKey
GROUP BY dd.Year
ORDER BY dd.Year;
```

	Year	TotalRevenue
1	2023	5219091.00
2	2024	5160214.00
3	2025	5027814.00

ocalDB (15.... | BRITTANYSHP\femal (56) | ClientWarehouse | 00:00:00 | 3 rows

Figure 14. SSMS Schema Comparison: side-by-side view of he OLTP (ETL_Project2) and OLAP (ClientWarehouse) database structures, showing normalized staging tables versus denormalized warehouse dimensions and fact table.



```
⊟ ETL_Project2
    ⊞ Database Diagrams
    ⊟ Tables
        ⊞ System Tables
        ⊞ FileTables
        ⊞ External Tables
        ⊞ Graph Tables
        ⊞ dbo.Customer_Clean
        ⊞ dbo.Product_Clean
        ⊞ dbo.Sale_Clean
    ⊞ Views
    ⊞ External Resources
    ⊞ Synonyms
```

```
⊟ ClientWarehouse
    ⊞ Database Diagrams
    ⊟ Tables
        ⊞ System Tables
        ⊞ FileTables
        ⊞ External Tables
        ⊞ Graph Tables
        ⊞ dbo.DimCustomer
        ⊞ dbo.DimDate
        ⊞ dbo.DimMovie
        ⊞ dbo.DimProduct
        ⊞ dbo.FactSales
    ⊞ Views
```