Customer Segmentation using SQL

Project Overview Designed and implemented a SQL-based data cleaning and transformation pipeline on customer orders dataset to enable accurate customer segmentation and insights.

Steps followed: 1. Created database, staging and final tables. 2. Loaded raw data into the staging table using bulk insert. 3. Performed data cleaning (null handling, duplicates removal, special character removal, type conversions). 4. Transferred clean data into the final orders table. 5. Showcased SQL queries for solving real-world business problems and insights.

Create Database

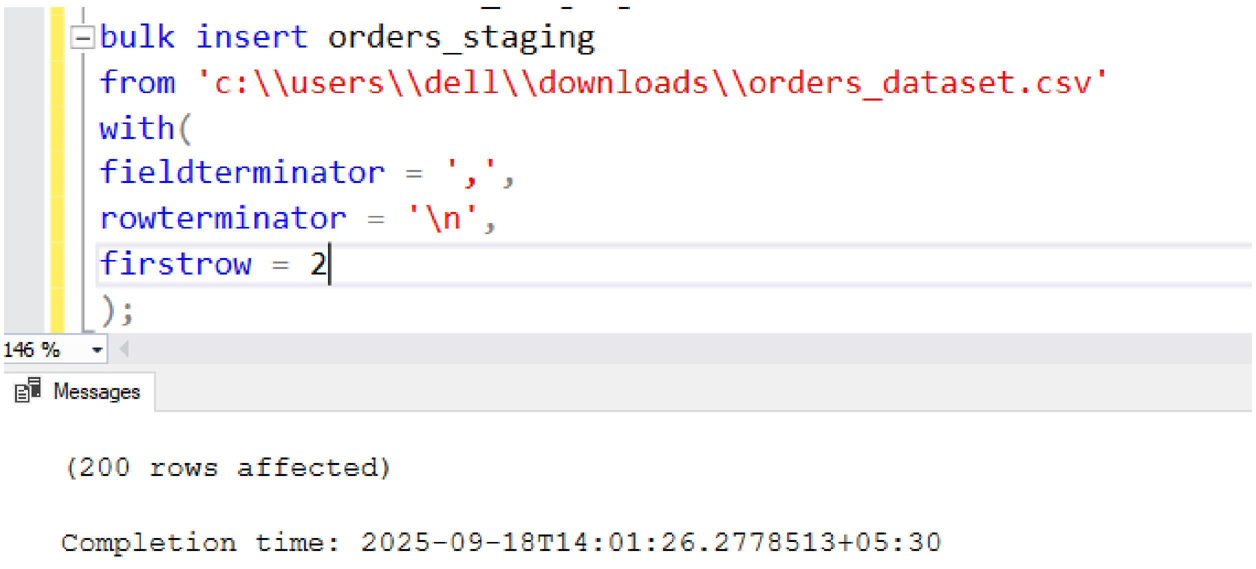
```
create database CustomerSegmentation;

use CustomerSegmentation;
```

Create Staging Table

Staging table stores raw data from csv for cleaning. Using nvarchar to handle any special characters.

```
create table orders_staging(

orderid nvarchar(50),

customerid nvarchar(50),

orderdate nvarchar(50),

amount nvarchar(50)

);
```

Load Data into Staging



```
bulk insert orders_staging
from 'c:\\users\\dell\\downloads\\orders_dataset.csv'
with(
fieldterminator = ',',
rowterminator = '\n',
firstrow = 2
);
```

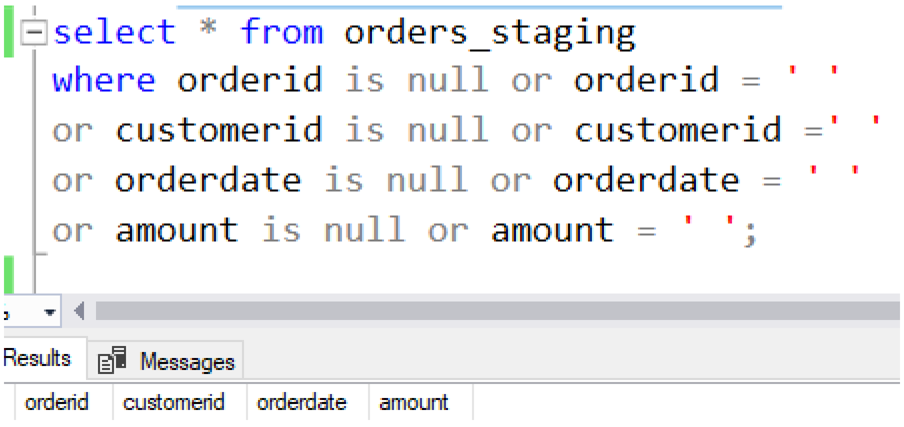
146 %

Messages

```
(200 rows affected)

Completion time: 2025-09-18T14:01:26.2778513+05:30
```

Data Cleaning in Staging

*** check for null / empty values**

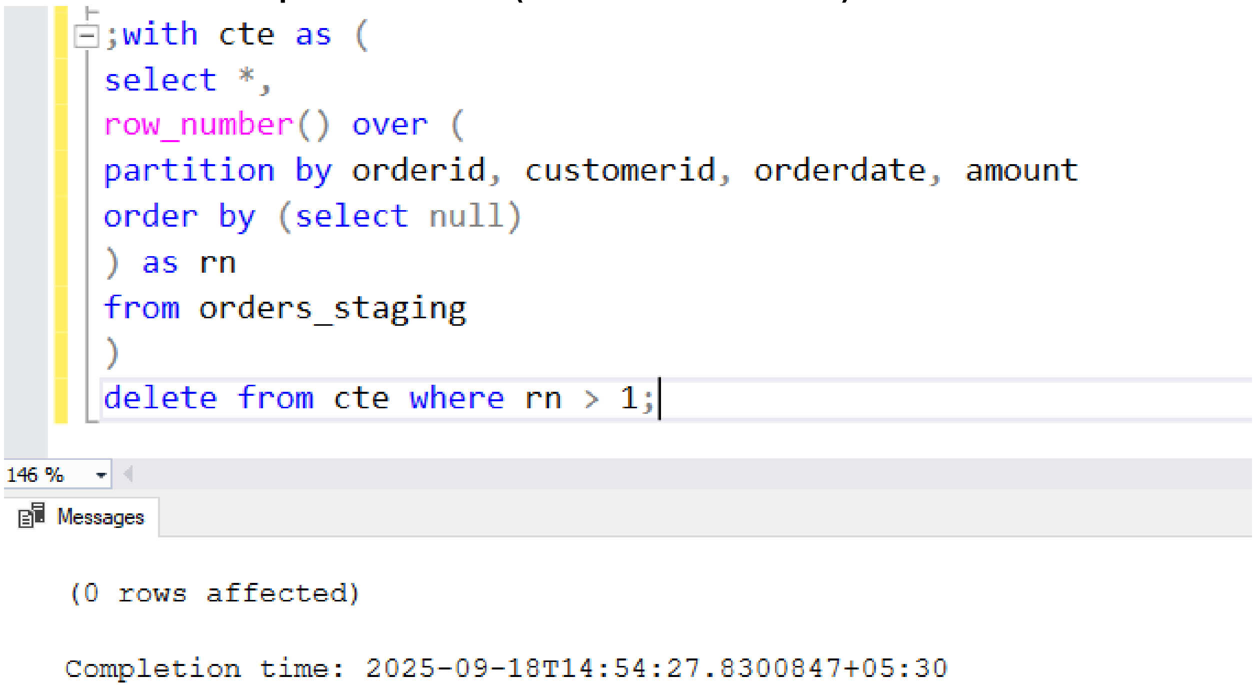


```
select * from orders_staging
where orderid is null or orderid = ' '
or customerid is null or customerid =' '
or orderdate is null or orderdate = ' '
or amount is null or amount = ' ';
```

Results | Messages

orderid	customerid	orderdate	amount

*** Remove Exact Duplicate Records (Based on All Columns)**



```
;with cte as (
select *,
row_number() over (
partition by orderid, customerid, orderdate, amount
order by (select null)
) as rn
from orders_staging
)
delete from cte where rn > 1;
```

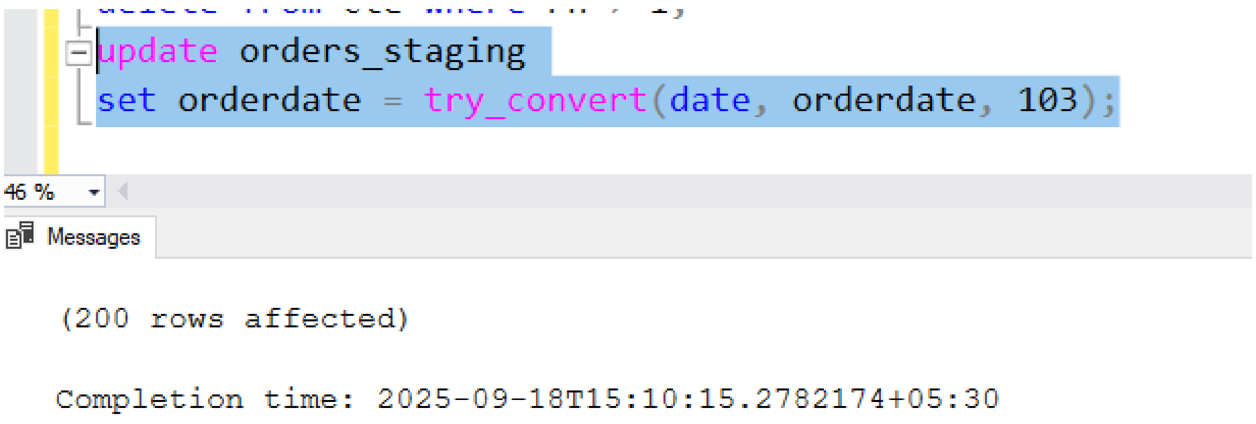
146 %

Messages

```
(0 rows affected)

Completion time: 2025-09-18T14:54:27.8300847+05:30
```

*** Fix data types**

-- convert orderdate to date



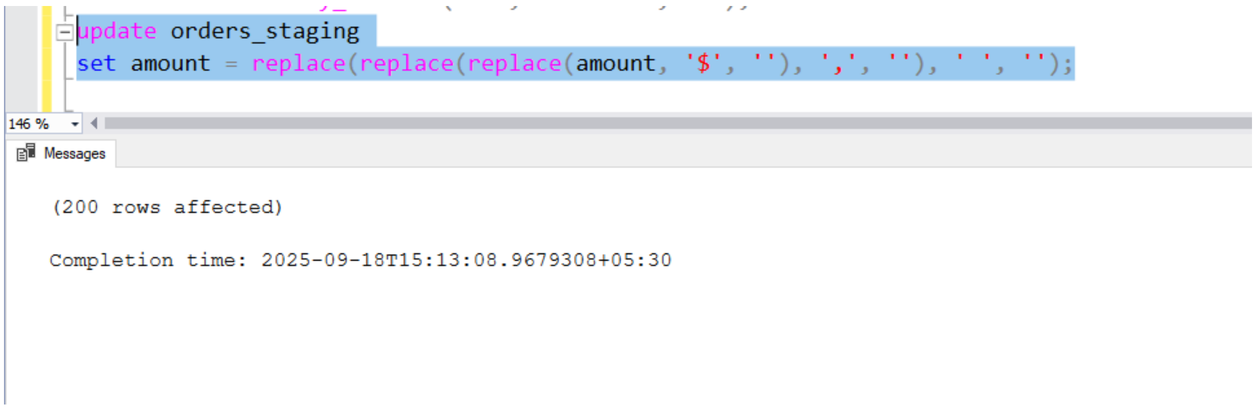
```
update orders_staging
set orderdate = try_convert(date, orderdate, 103);
```

46 %

Messages

```
(200 rows affected)

Completion time: 2025-09-18T15:10:15.2782174+05:30
```

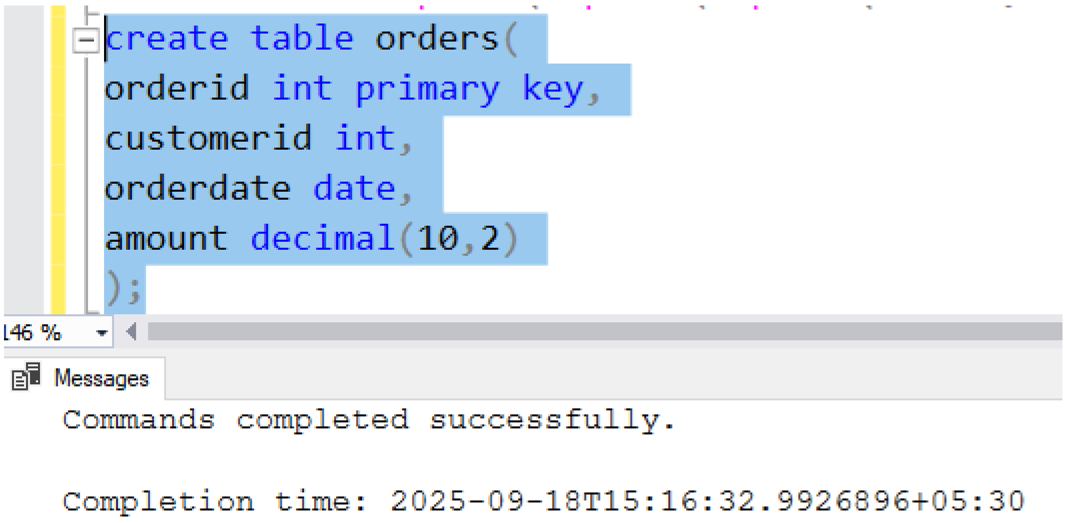
```
-- remove special characters from amount
```



```sql
update orders_staging
set amount = replace(replace(replace(amount, '$', ''), ',', ''), ' ', '');
```

```
(200 rows affected)

Completion time: 2025-09-18T15:13:08.9679308+05:30
```

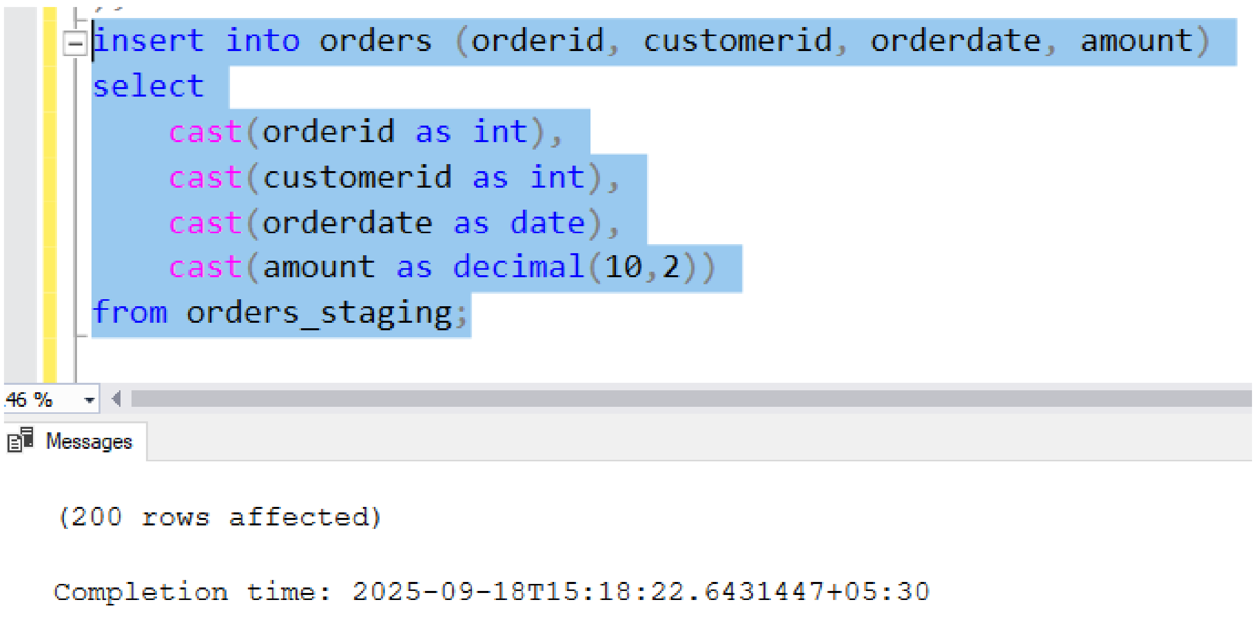
Create Final Orders Table



```sql
create table orders(
orderid int primary key,
customerid int,
orderdate date,
amount decimal(10,2)
);
```

```
Commands completed successfully.

Completion time: 2025-09-18T15:16:32.9926896+05:30
```

Insert Clean Data into Final Table



```
insert into orders (orderid, customerid, orderdate, amount)
select
    cast(orderid as int),
    cast(customerid as int),
    cast(orderdate as date),
    cast(amount as decimal(10,2))
from orders_staging;
```

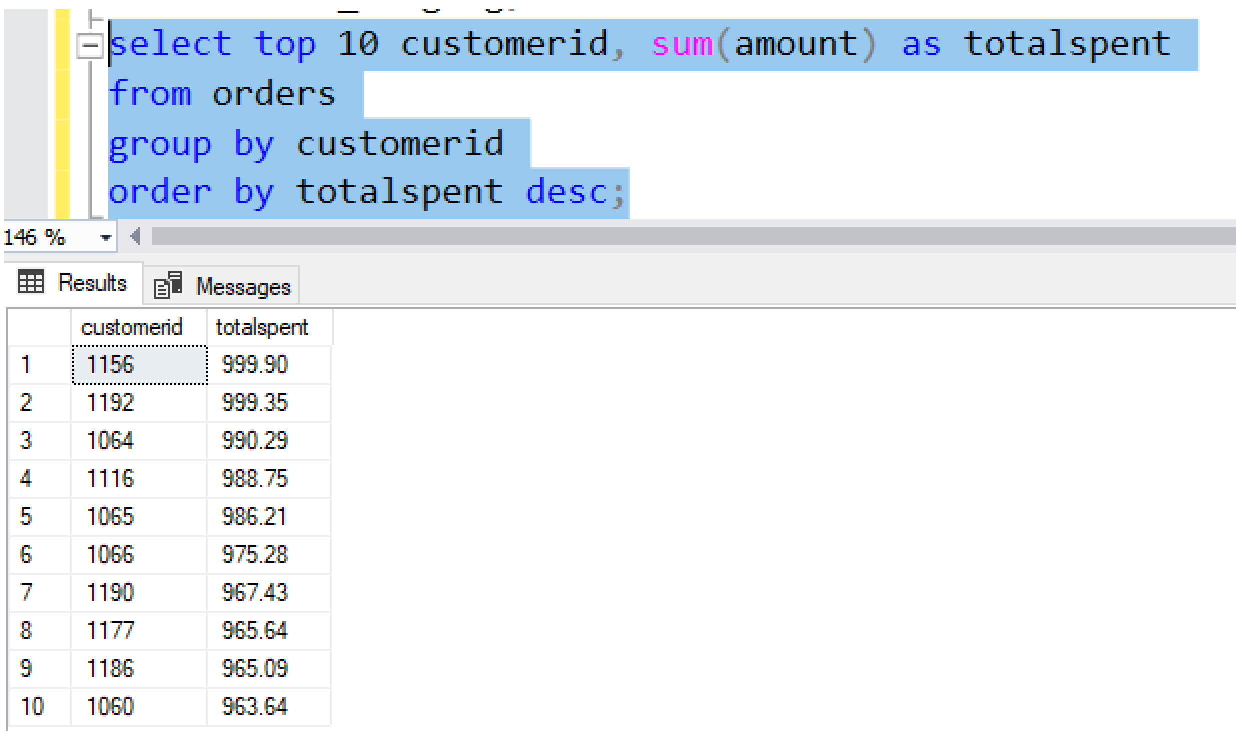
46 %

Messages

```
(200 rows affected)

Completion time: 2025-09-18T15:18:22.6431447+05:30
```

Sample Business Queries / Customer Insights

Top Customers by Spending



```
select top 10 customerid, sum(amount) as totalspent
from orders
group by customerid
order by totalspent desc;
```

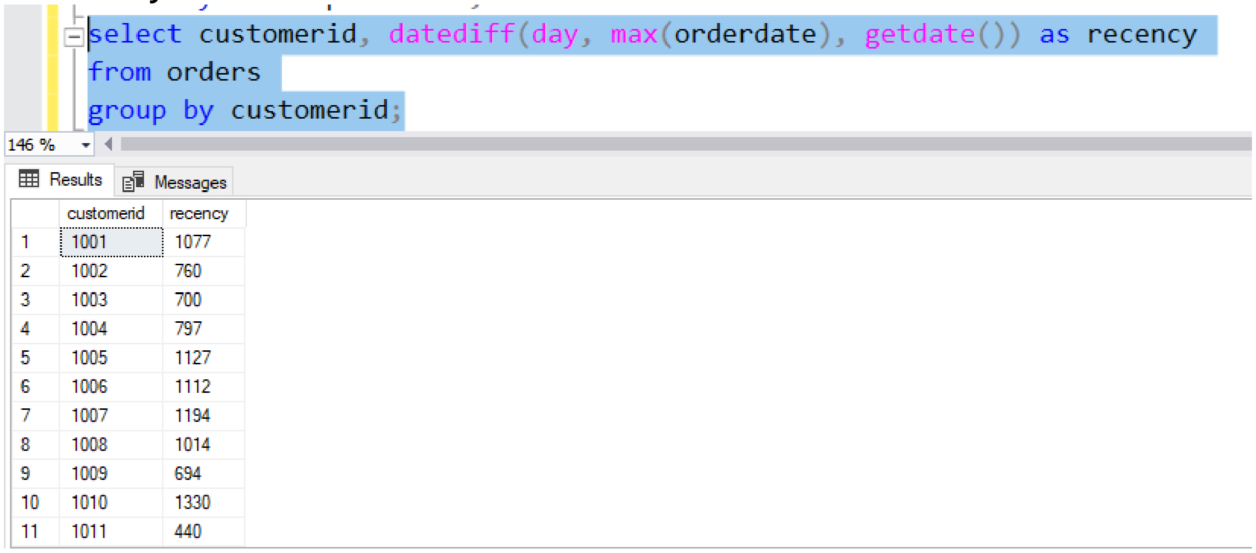
146 %

Results | Messages

	customerid	totalspent
1	1156	999.90
2	1192	999.35
3	1064	990.29
4	1116	988.75
5	1065	986.21
6	1066	975.28
7	1190	967.43
8	1177	965.64
9	1186	965.09
10	1060	963.64

Recency, Frequency, Monetary (RFM) Analysis

-- recency



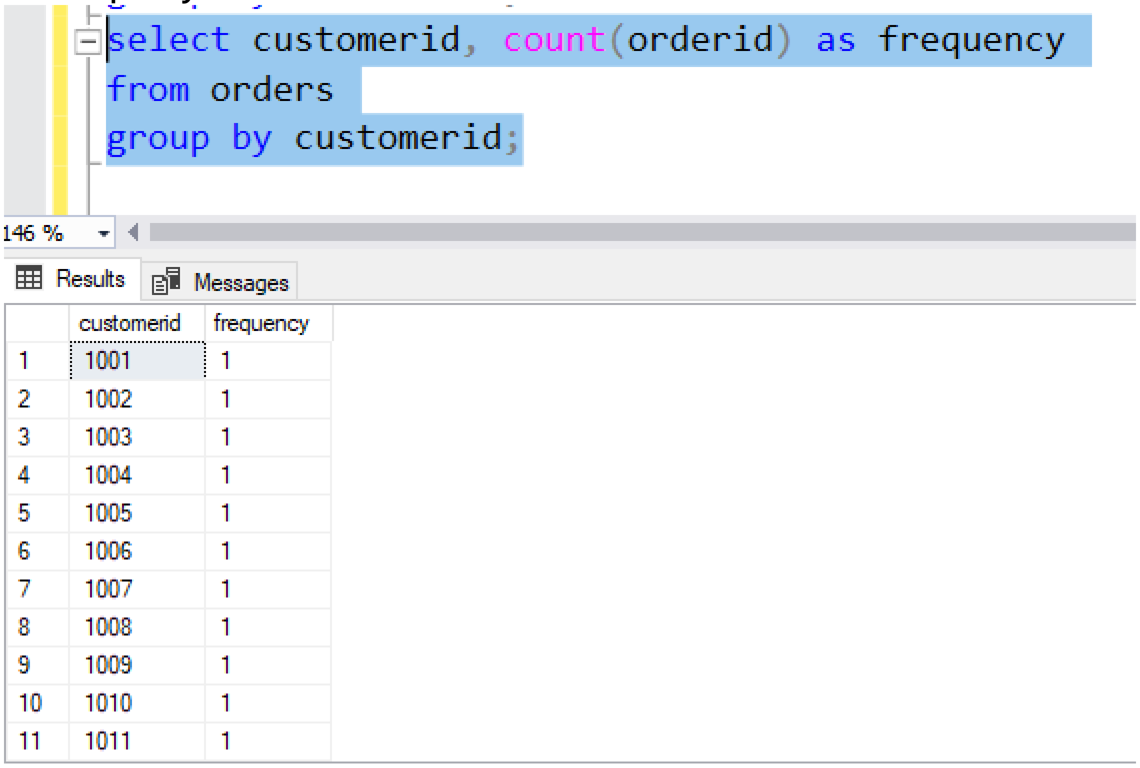
```
select customerid, datediff(day, max(orderdate), getdate()) as recency
from orders
group by customerid;
```

146 %

Results | Messages

	customerid	recency
1	1001	1077
2	1002	760
3	1003	700
4	1004	797
5	1005	1127
6	1006	1112
7	1007	1194
8	1008	1014
9	1009	694
10	1010	1330
11	1011	440

-- frequency



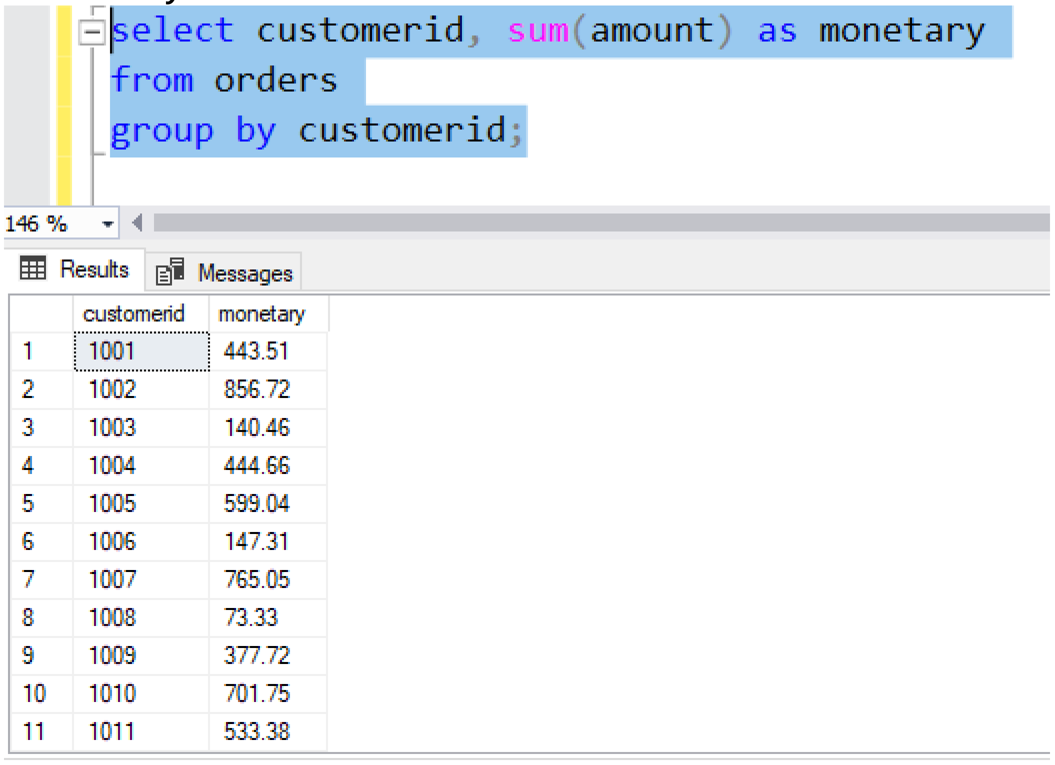
```
select customerid, count(orderid) as frequency
from orders
group by customerid;
```

146 %

Results | Messages

	customerid	frequency
1	1001	1
2	1002	1
3	1003	1
4	1004	1
5	1005	1
6	1006	1
7	1007	1
8	1008	1
9	1009	1
10	1010	1
11	1011	1

```
-- monetary
```



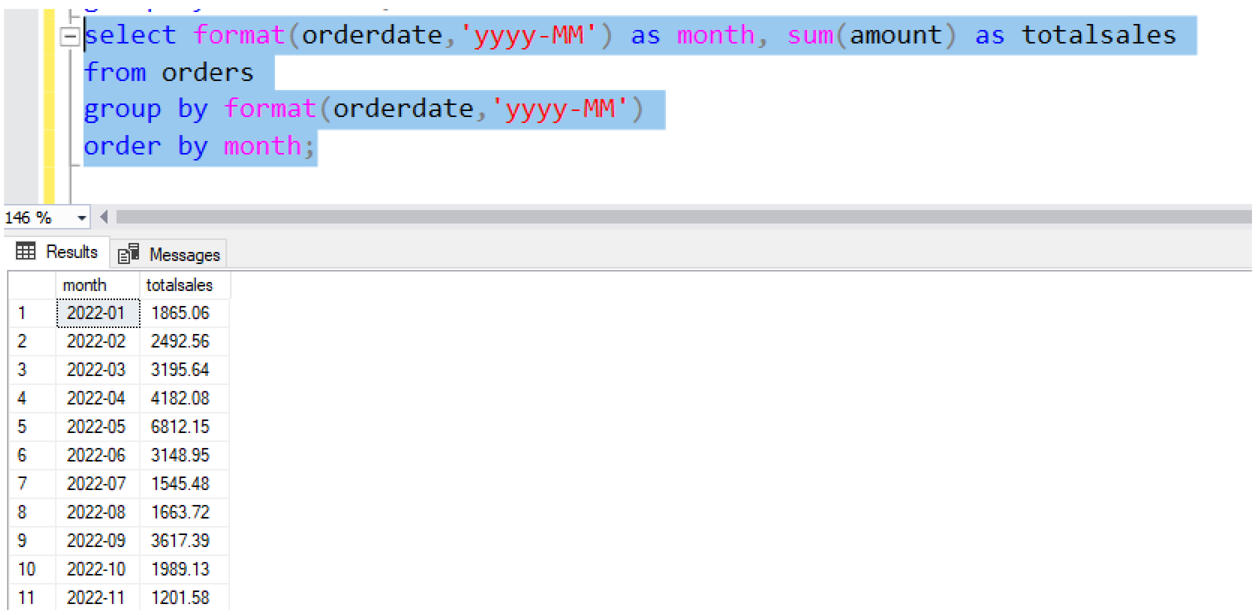
```
select customerid, sum(amount) as monetary
from orders
group by customerid;
```

146 %

Results | Messages

	customerid	monetary
1	1001	443.51
2	1002	856.72
3	1003	140.46
4	1004	444.66
5	1005	599.04
6	1006	147.31
7	1007	765.05
8	1008	73.33
9	1009	377.72
10	1010	701.75
11	1011	533.38

Monthly Sales Trend



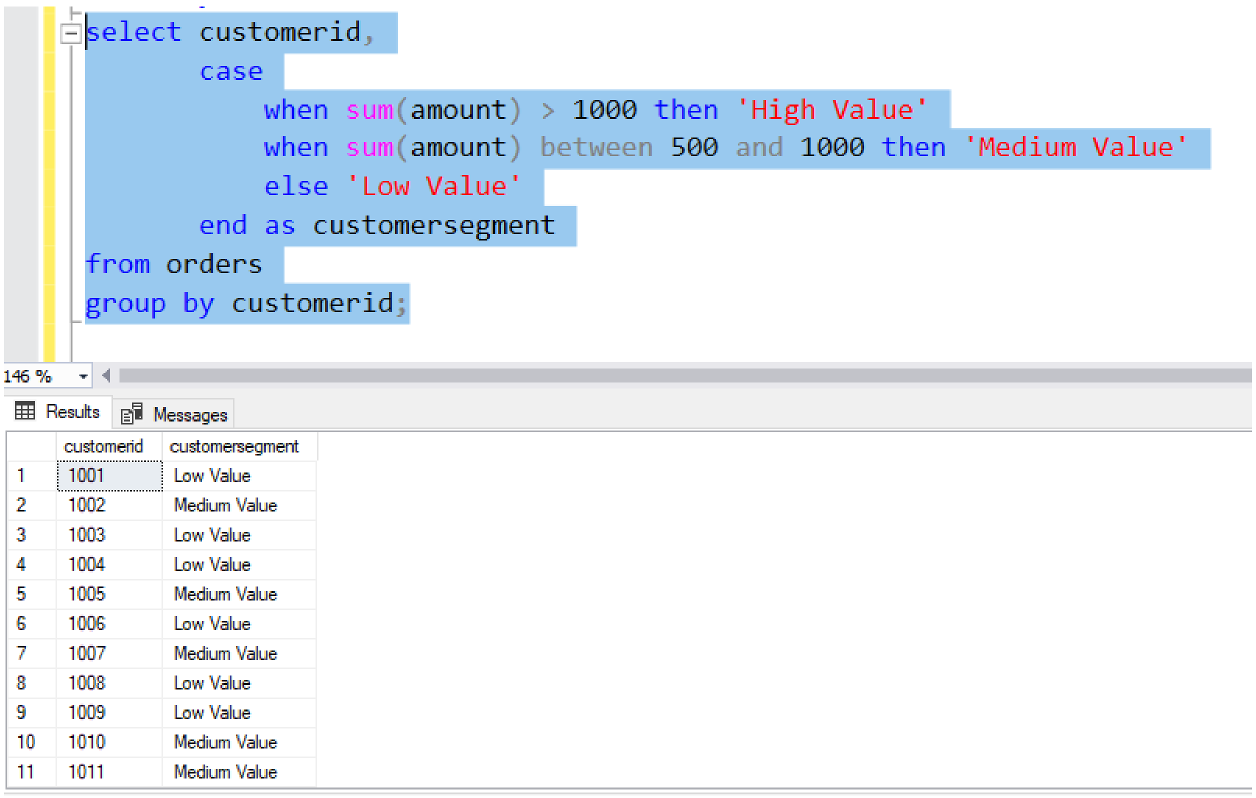
```
select format(orderdate,'yyyy-MM') as month, sum(amount) as totalsales
from orders
group by format(orderdate,'yyyy-MM')
order by month;
```

146 %

Results | Messages

	month	totalsales
1	2022-01	1865.06
2	2022-02	2492.56
3	2022-03	3195.64
4	2022-04	4182.08
5	2022-05	6812.15
6	2022-06	3148.95
7	2022-07	1545.48
8	2022-08	1663.72
9	2022-09	3617.39
10	2022-10	1989.13
11	2022-11	1201.58

Customer Segmentation Example



```sql
select customerid,
        case
            when sum(amount) > 1000 then 'High Value'
            when sum(amount) between 500 and 1000 then 'Medium Value'
            else 'Low Value'
        end as customersegment
from orders
group by customerid;
```

	customerid	customersegment
1	1001	Low Value
2	1002	Medium Value
3	1003	Low Value
4	1004	Low Value
5	1005	Medium Value
6	1006	Low Value
7	1007	Medium Value
8	1008	Low Value
9	1009	Low Value
10	1010	Medium Value
11	1011	Medium Value

Conclusion & Insights

This project demonstrates end-to-end SQL skills, including database creation, data cleaning, transformation, and business insights. The RFM analysis and segmentation reveal valuable patterns in customer behavior, helping businesses target high-value customers and improve decision-making.
