CREATE DATABASE data\_bank;

REGIONS TABLE

CREATE TABLE data\_bank.dbo.regions (

region\_id INTEGER,

region\_name VARCHAR(9)

);

CUSTOMER NODES TABLE

CREATE TABLE data\_bank.dbo.customer\_nodes (

customer\_id INTEGER,

region\_id INTEGER,

node\_id INTEGER,

start\_date DATE,

end\_date DATE

);

CUSTOMER TRANSACTIONS TABLE

CREATE TABLE data\_bank.dbo.customer\_transactions (

customer\_id INTEGER,

txn\_date DATE,

txn\_type VARCHAR(10),

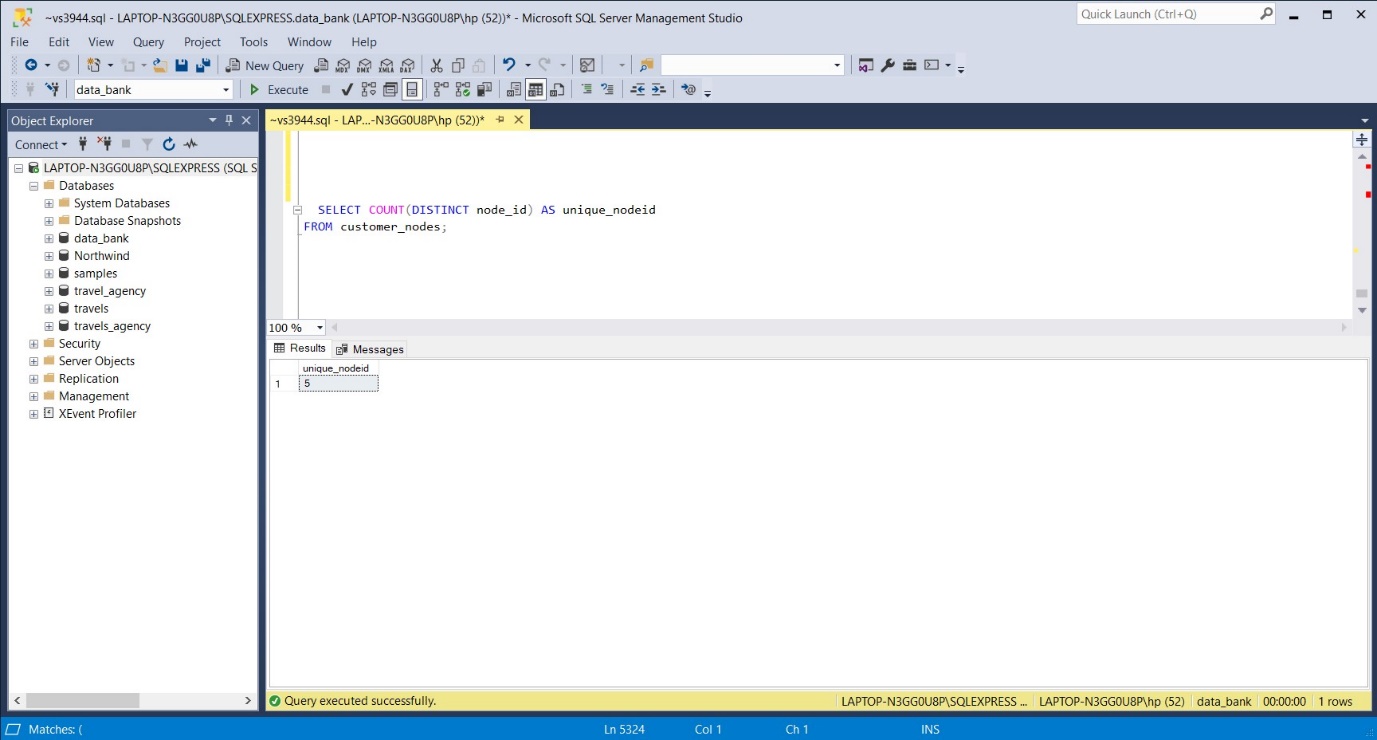
txn\_amount INTEGER

);

1)

SELECT COUNT(DISTINCT node\_id) AS unique\_nodes

FROM customer\_nodes;



2)

SELECT

r.region\_id,

r.region\_name,

COUNT(n.node\_id) AS nodes

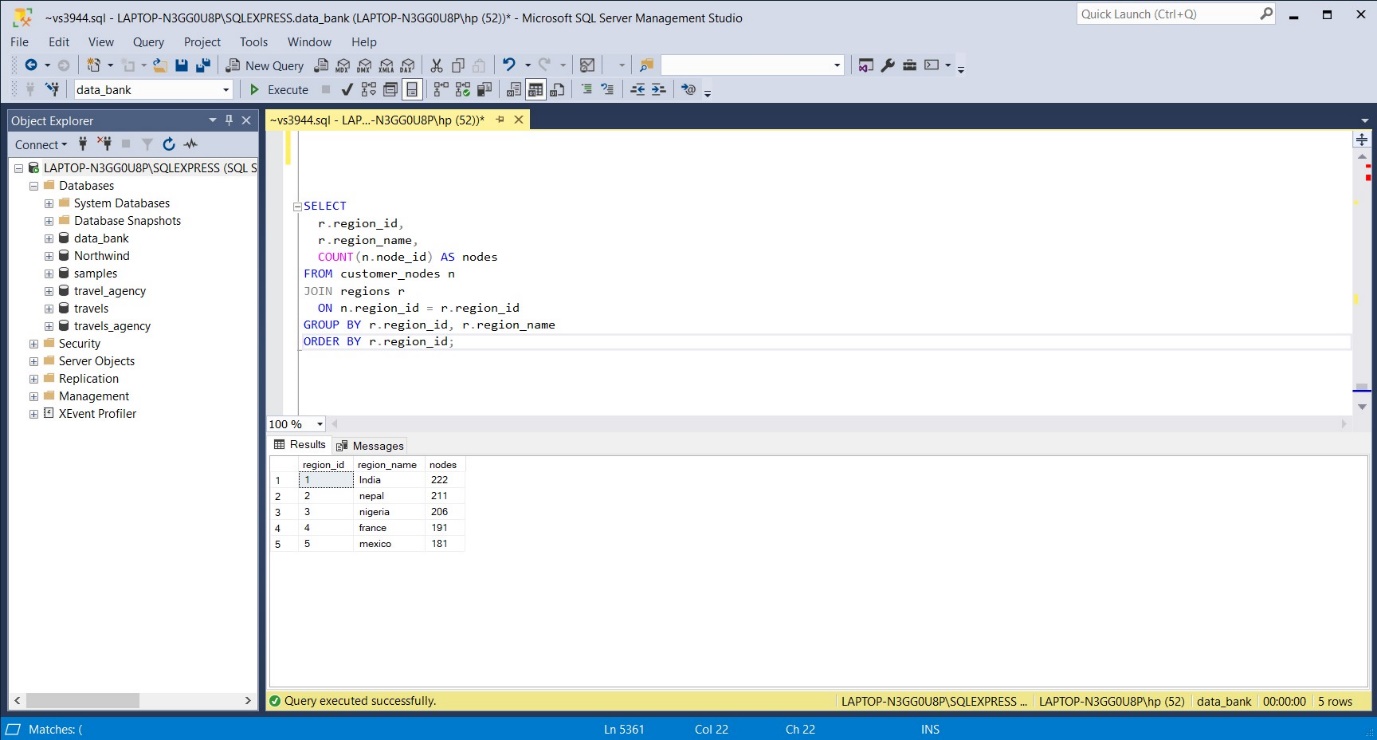
FROM customer\_nodes n

JOIN regions r

ON n.region\_id = r.region\_id

GROUP BY r.region\_id, r.region\_name

ORDER BY r.region\_id;



3)

SELECT

r.region\_id,

r.region\_name,

COUNT(DISTINCT n.customer\_id) AS customers

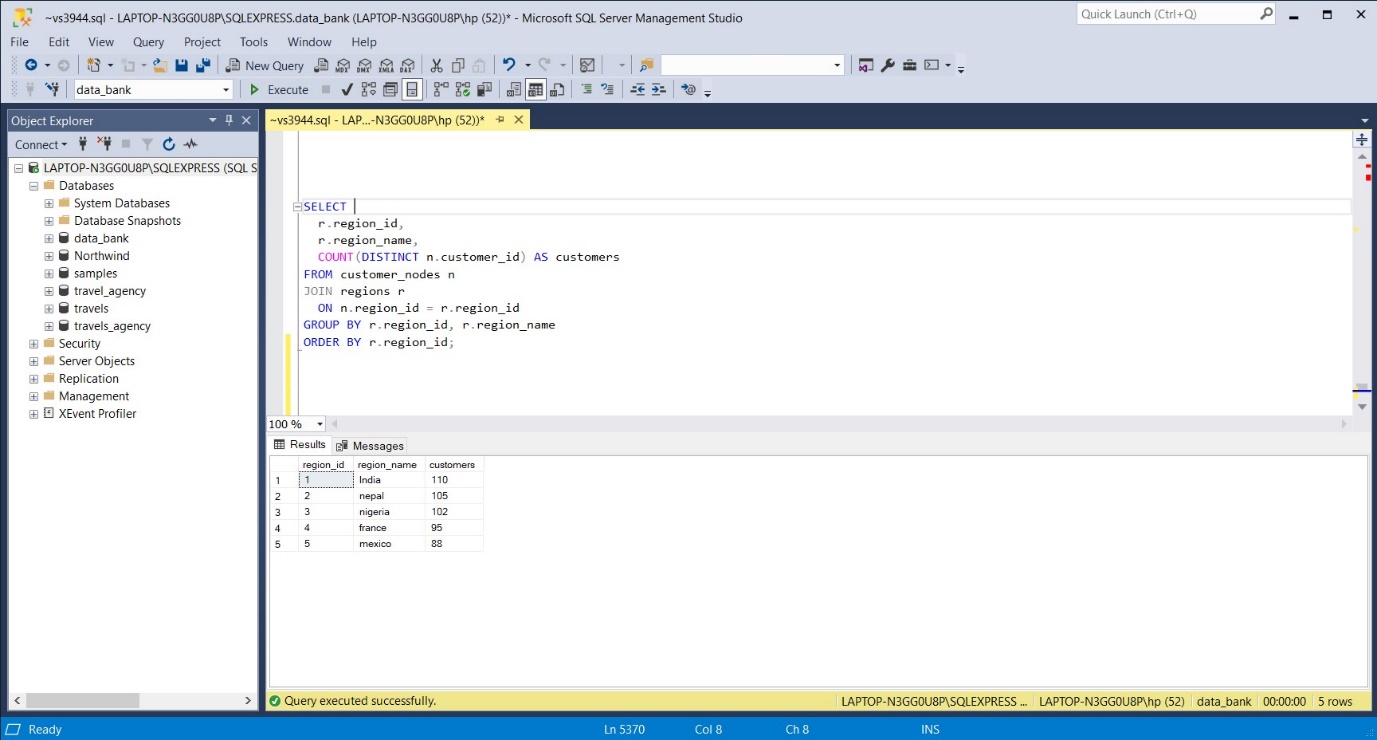
FROM customer\_nodes n

JOIN regions r

ON n.region\_id = r.region\_id

GROUP BY r.region\_id, r.region\_name

ORDER BY r.region\_id;



4)

WITH customerDates AS (

SELECT

customer\_id,

region\_id,

node\_id,

MIN(start\_date) AS first\_date

FROM customer\_nodes

GROUP BY customer\_id, region\_id, node\_id

),

reallocation AS (

SELECT

customer\_id,

node\_id,

region\_id,

first\_date,

DATEDIFF(DAY, first\_date,

LEAD(first\_date) OVER(PARTITION BY customer\_id

ORDER BY first\_date)) AS moving\_days

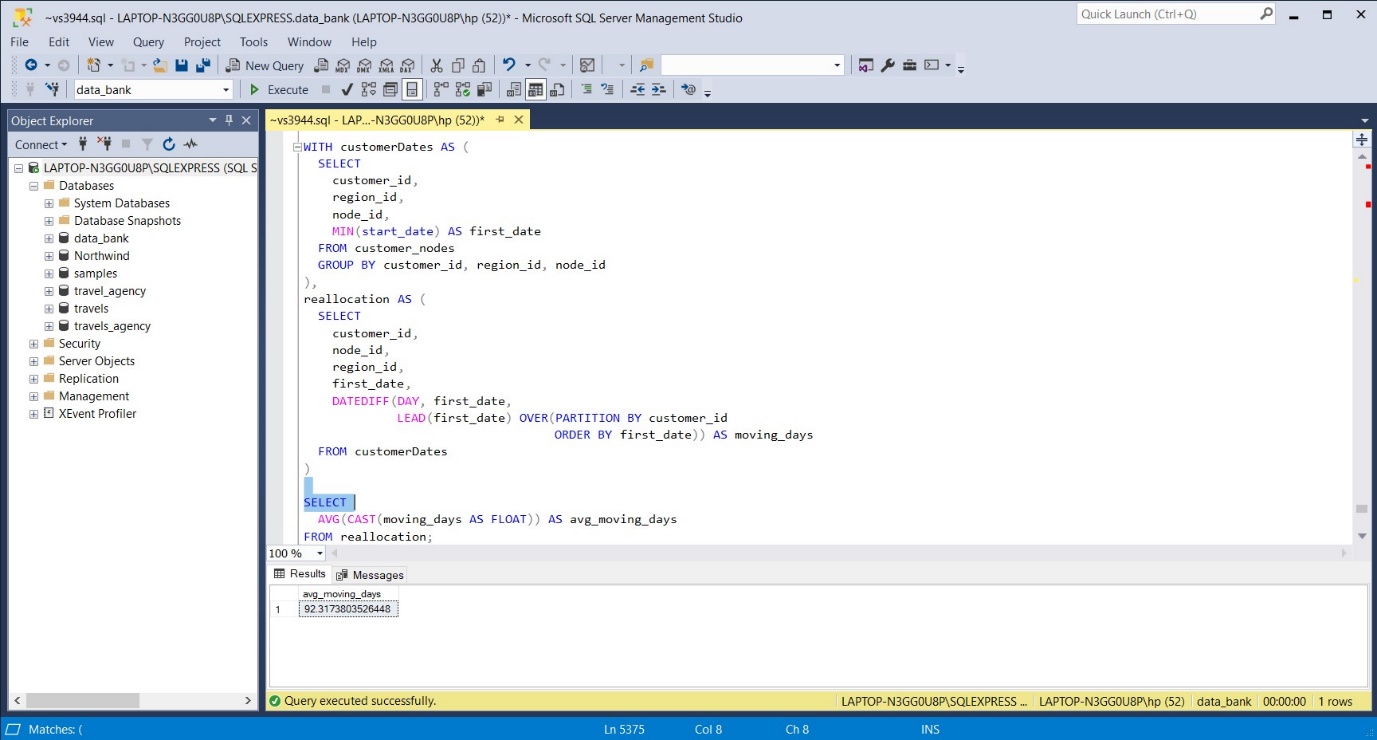
FROM customerDates

)

SELECT

AVG(CAST(moving\_days AS FLOAT)) AS avg\_moving\_days

FROM reallocation;



5)

WITH customerDates AS (

SELECT

customer\_id,

region\_id,

node\_id,

MIN(start\_date) AS first\_date

FROM customer\_nodes

GROUP BY customer\_id, region\_id, node\_id

),

reallocation AS (

SELECT

customer\_id,

region\_id,

node\_id,

first\_date,

DATEDIFF(DAY, first\_date,

LEAD(first\_date) OVER(PARTITION BY customer\_id

ORDER BY first\_date)) AS moving\_days

FROM customerDates

)

SELECT

DISTINCT r.region\_id,

rg.region\_name,

PERCENTILE\_CONT(0.5) WITHIN GROUP (ORDER BY r.moving\_days) OVER(PARTITION BY r.region\_id) AS median,

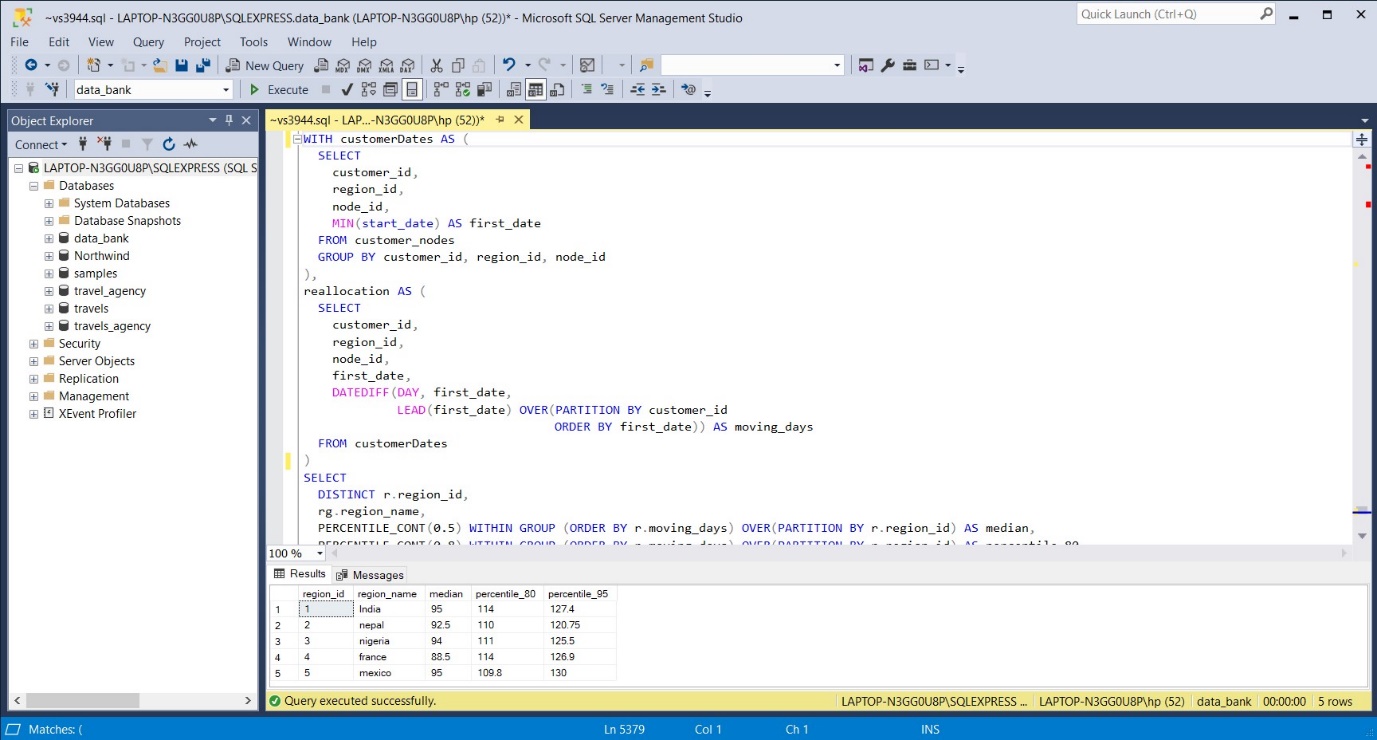
PERCENTILE\_CONT(0.8) WITHIN GROUP (ORDER BY r.moving\_days) OVER(PARTITION BY r.region\_id) AS percentile\_80,

PERCENTILE\_CONT(0.95) WITHIN GROUP (ORDER BY r.moving\_days) OVER(PARTITION BY r.region\_id) AS percentile\_95

FROM reallocation r

JOIN regions rg ON r.region\_id = rg.region\_id

WHERE moving\_days IS NOT NULL;



6)

SELECT

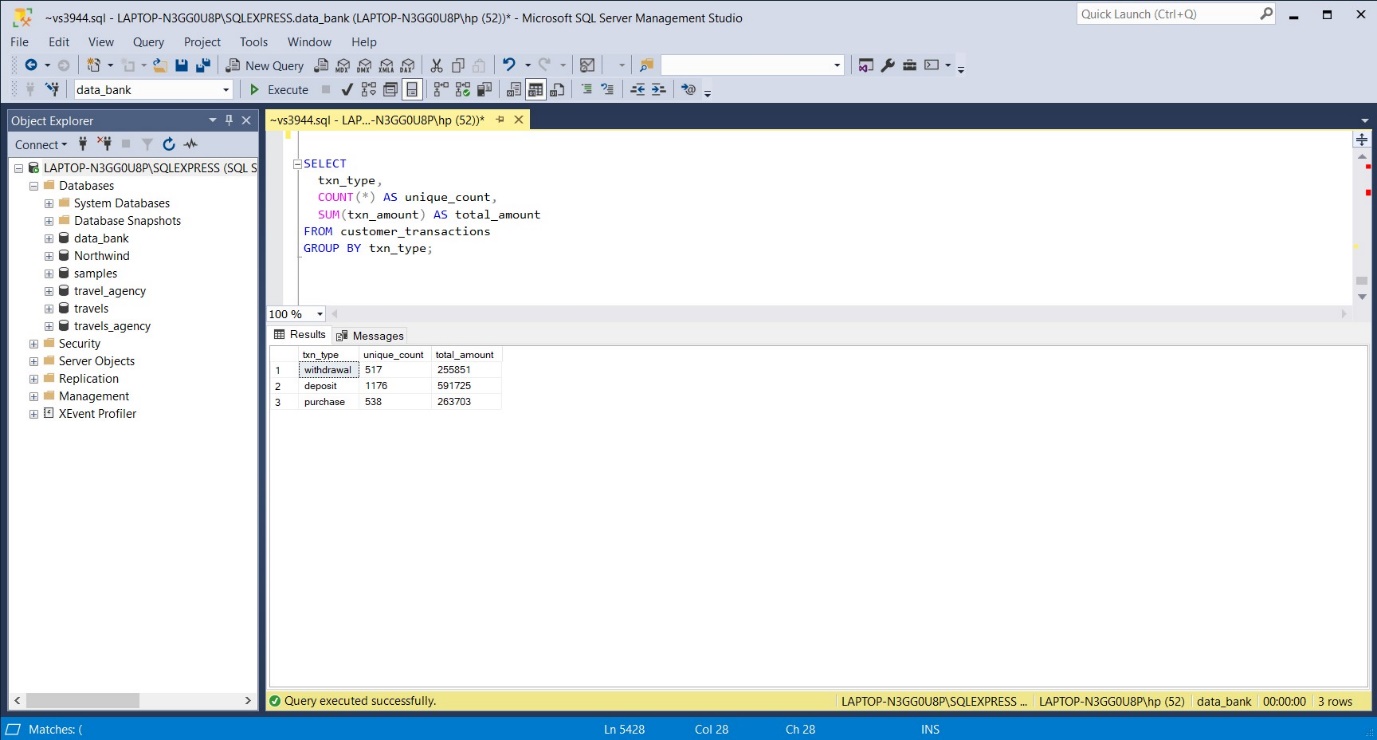
txn\_type,

COUNT(\*) AS unique\_count,

SUM(txn\_amount) AS total\_amount

FROM customer\_transactions

GROUP BY txn\_type;



7)

WITH customerDeposit AS (

SELECT

customer\_id,

txn\_type,

COUNT(\*) AS dep\_count,

SUM(txn\_amount) AS dep\_amount

FROM customer\_transactions

WHERE txn\_type = 'deposit'

GROUP BY customer\_id, txn\_type

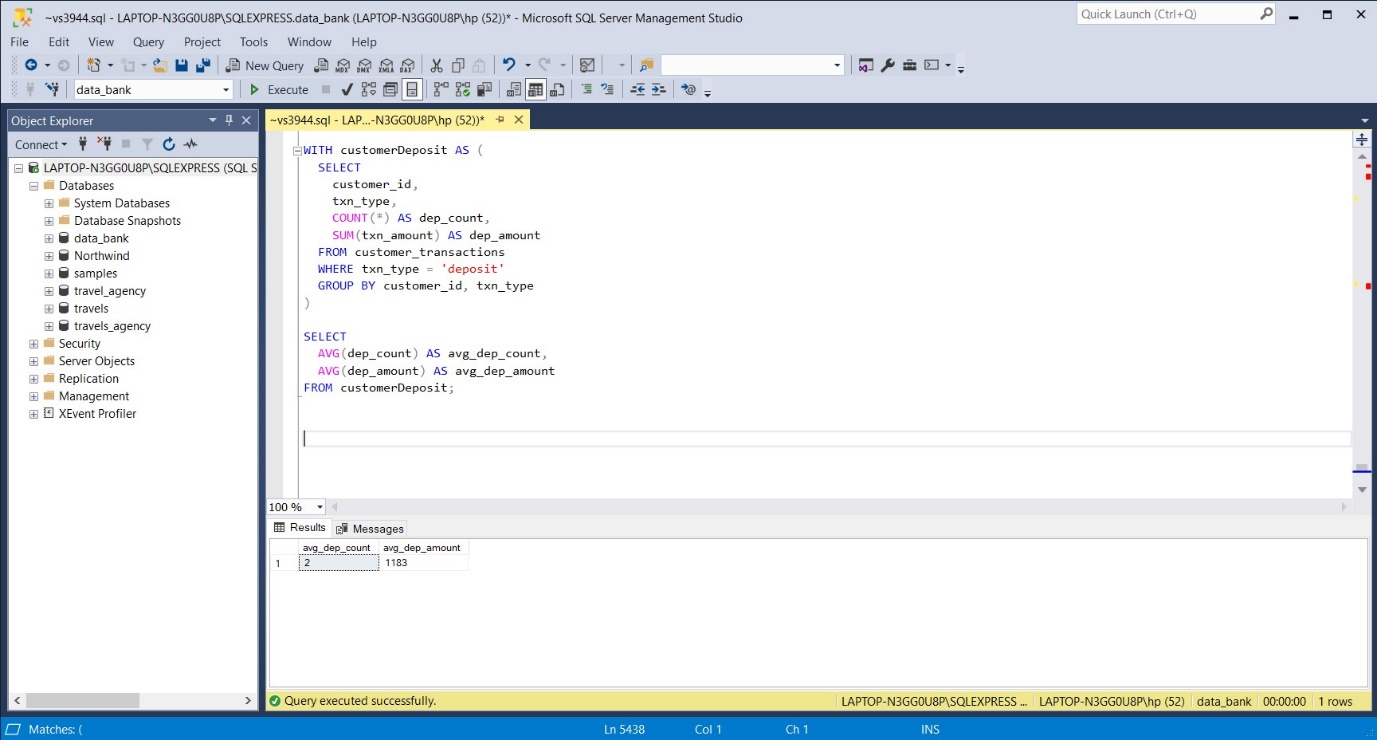
)

SELECT

AVG(dep\_count) AS avg\_dep\_count,

AVG(dep\_amount) AS avg\_dep\_amount

FROM customerDeposit;



8)

WITH cte\_transaction AS (

SELECT

customer\_id,

MONTH(txn\_date) AS months,

SUM(CASE WHEN txn\_type = 'deposit' THEN 1 ELSE 0 END) AS deposit\_count,

SUM(CASE WHEN txn\_type = 'purchase' THEN 1 ELSE 0 END) AS purchase\_count,

SUM(CASE WHEN txn\_type = 'withdrawal' THEN 1 ELSE 0 END) AS withdrawal\_count

FROM customer\_transactions

GROUP BY customer\_id, MONTH(txn\_date)

)

SELECT

months,

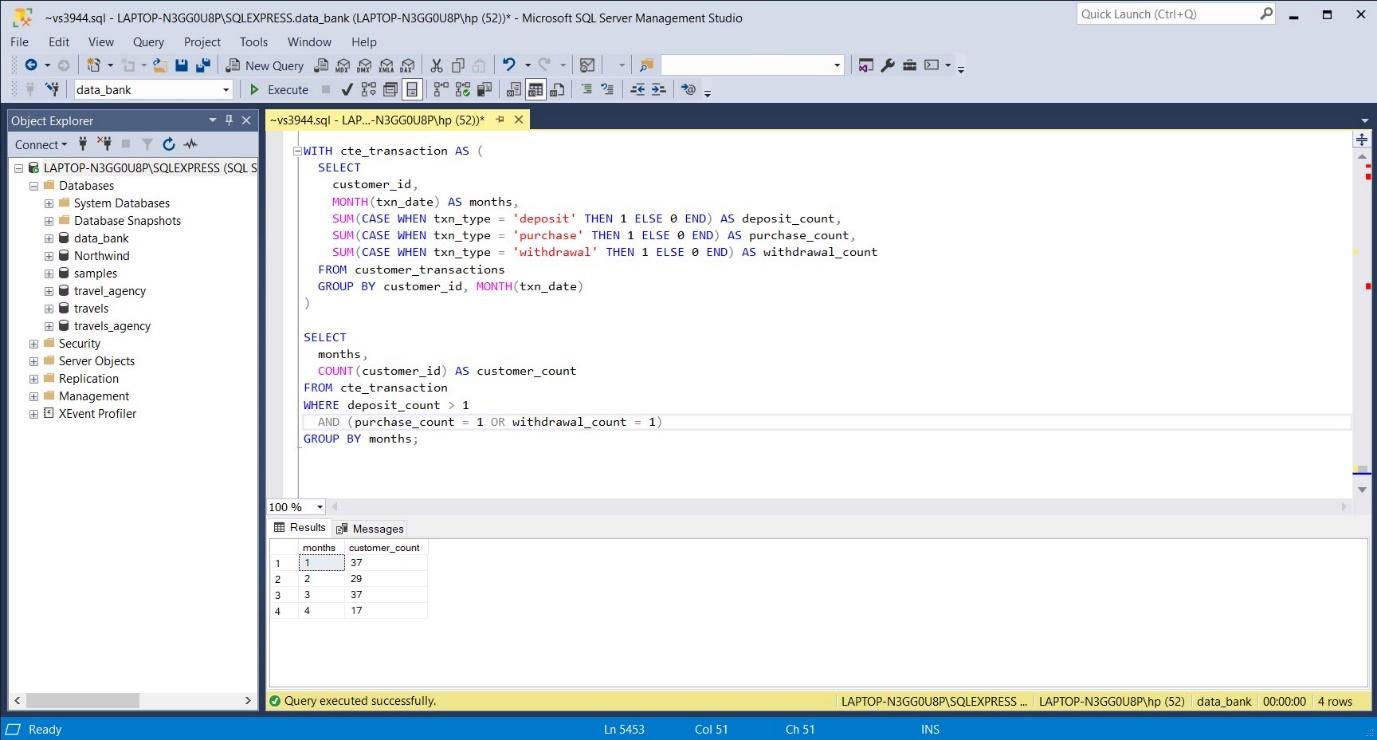
COUNT(customer\_id) AS customer\_count

FROM cte\_transaction

WHERE deposit\_count > 1

AND (purchase\_count = 1 OR withdrawal\_count = 1)

GROUP BY months;



9)

--End date in the month of the max date of our dataset

DECLARE @maxDate DATE;

SET @maxDate = (SELECT EOMONTH(MAX(txn\_date)) FROM customer\_transactions);

--CTE 1: Monthly transactions of each customer

WITH monthly\_transactions AS (

SELECT

customer\_id,

EOMONTH(txn\_date) AS end\_date,

SUM(CASE WHEN txn\_type IN ('withdrawal', 'purchase') THEN -txn\_amount

ELSE txn\_amount END) AS transactions

FROM customer\_transactions

GROUP BY customer\_id, EOMONTH(txn\_date)

),

--CTE 2: Increment last days of each month till they are equal to @maxDate

recursive\_dates AS (

SELECT

DISTINCT customer\_id,

CAST('2020-01-31' AS DATE) AS end\_date

FROM customer\_transactions

UNION ALL

SELECT

customer\_id,

EOMONTH(DATEADD(MONTH, 1, end\_date)) AS end\_date

FROM recursive\_dates

WHERE EOMONTH(DATEADD(MONTH, 1, end\_date)) <= @maxDate

)

SELECT

r.customer\_id,

r.end\_date,

COALESCE(m.transactions, 0) AS transactions,

SUM(m.transactions) OVER (PARTITION BY r.customer\_id ORDER BY r.end\_date

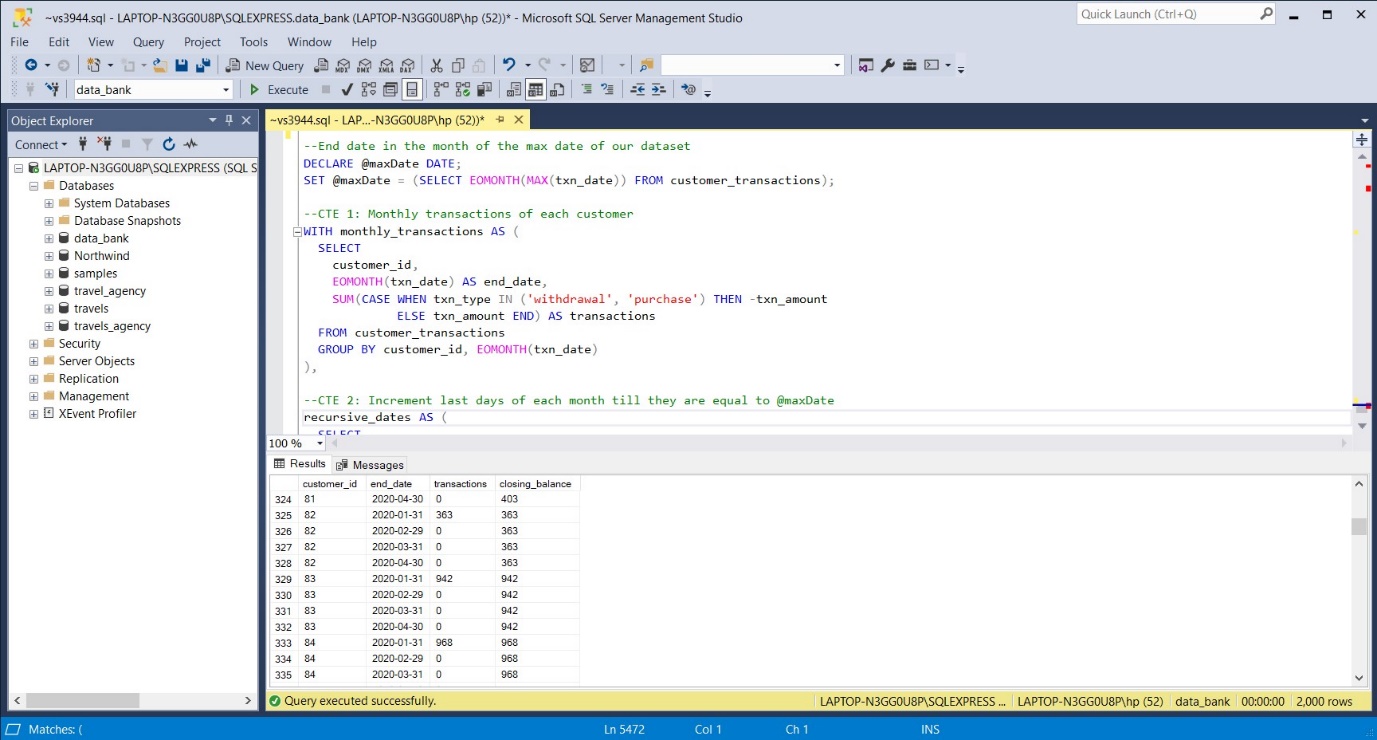
ROWS UNBOUNDED PRECEDING) AS closing\_balance

FROM recursive\_dates r

LEFT JOIN monthly\_transactions m

ON r.customer\_id = m.customer\_id

AND r.end\_date = m.end\_date;



10)

--End date in the month of the max date of our dataset (Q4)

DECLARE @maxDate DATE;

SET @maxDate = (SELECT EOMONTH(MAX(txn\_date)) FROM customer\_transactions);

--CTE 1: Monthly transactions of each customer (Q4)

WITH monthly\_transactions AS (

SELECT

customer\_id,

EOMONTH(txn\_date) AS end\_date,

SUM(CASE WHEN txn\_type IN ('withdrawal', 'purchase') THEN -txn\_amount

ELSE txn\_amount END) AS transactions

FROM customer\_transactions

GROUP BY customer\_id, EOMONTH(txn\_date)

),

--CTE 2: Increment last days of each month till they are equal to @maxDate (Q4)

recursive\_dates AS (

SELECT

DISTINCT customer\_id,

CAST('2020-01-31' AS DATE) AS end\_date

FROM customer\_transactions

UNION ALL

SELECT

customer\_id,

EOMONTH(DATEADD(MONTH, 1, end\_date)) AS end\_date

FROM recursive\_dates

WHERE EOMONTH(DATEADD(MONTH, 1, end\_date)) <= @maxDate

),

-- CTE 3: Closing balance of each customer by monthly (Q4)

customers\_balance AS (

SELECT

r.customer\_id,

r.end\_date,

COALESCE(m.transactions, 0) AS transactions,

SUM(m.transactions) OVER (PARTITION BY r.customer\_id ORDER BY r.end\_date

ROWS UNBOUNDED PRECEDING) AS closing\_balance

FROM recursive\_dates r

LEFT JOIN monthly\_transactions m

ON r.customer\_id = m.customer\_id

AND r.end\_date = m.end\_date

),

--CTE 4: CTE 3 & next\_balance

customers\_next\_balance AS (

SELECT \*,

LEAD(closing\_balance) OVER(PARTITION BY customer\_id ORDER BY end\_date) AS next\_balance

FROM customers\_balance

),

--CTE 5: Calculate the increase percentage of closing balance for each customer

pct\_increase AS (

SELECT \*,

100.0\*(next\_balance-closing\_balance)/closing\_balance AS pct

FROM customers\_next\_balance

WHERE closing\_balance ! = 0 AND next\_balance IS NOT NULL

)

--Create a temporary table because of the error: Null value is eliminated by an aggregate or other SET operation

SELECT \*

INTO #temp

FROM pct\_increase;

--Calculate the percentage of customers whose closing balance increasing 5% compared to the previous month

SELECT CAST(100.0\*COUNT(DISTINCT customer\_id) AS FLOAT)

/ (SELECT COUNT(DISTINCT customer\_id) FROM customer\_transactions) AS pct\_customers

FROM #temp

WHERE pct > 5;

