SQL SERVER CASE STUDY

CREATE TABLE regions (

region\_id INTEGER primary key,

region\_name VARCHAR(9)

);

INSERT INTO regions (region\_id, region\_name) VALUES

('1', 'Australia'),

('2', 'America'),

('3', 'Africa'),

('4', 'Asia'),

('5', 'Europe');

CREATE TABLE customer\_nodes (

customer\_id INTEGER,

CONSTRAINT "FK\_CUSTNODES\_custid" FOREIGN KEY ("customer\_id") REFERENCES "dbo"."customer\_transactions"("customer\_id"),

region\_id INTEGER,

CONSTRAINT "FK\_CUSTNODES\_regionid" FOREIGN KEY ("region\_id") REFERENCES "dbo"."regions"("region\_id"),

node\_id INTEGER,

start\_date DATE,

end\_date DATE

);

INSERT INTO customer\_nodes

(customer\_id, region\_id, node\_id, start\_date, end\_date) VALUES

('429', '3', '4', '2020-01-02', '2020-01-03'),

('155', '3', '5', '2020-01-03', '2020-01-17'),

('398', '5', '4', '2020-01-27', '2020-02-18'),

('255', '5', '4', '2020-01-07', '2020-01-19'),

('185', '3', '3', '2020-01-15', '2020-01-23'),

('309', '1', '1', '2020-01-11', '2020-02-06'),

('312', '2', '5', '2020-01-20', '2020-02-04'),

('376', '1', '2', '2020-01-15', '2020-01-28'),

('188', '4', '5', '2020-01-21', '2020-01-25'),

('138', '3', '4', '2020-01-13', '2020-01-14'),

('373', '2', '5', '2020-01-19', '2020-01-25'),

('361', '1', '2', '2020-01-13', '2020-01-14'),

('169', '2', '3', '2020-01-02', '2020-01-14'),

('402', '1', '2', '2020-01-25', '2020-01-25'),

('60', '1', '3', '2020-01-25', '2020-02-08'),

('378', '4', '4', '2020-01-13', '2020-01-18'),

('383', '2', '3', '2020-01-19', '2020-01-27'),

('292', '1', '3', '2020-01-17', '2020-02-15'),

('63', '2', '2', '2020-01-17', '2020-02-06'),

('499', '2', '4', '2020-01-18', '2020-02-09'),

('130', '3', '4', '2020-01-04', '2020-01-14'),

('130', '3', '3', '2020-01-18', '2020-02-09'),

('441', '5', '5', '2020-02-19', '2020-03-06'),

('53', '5', '4', '2020-01-20', '2020-02-13'),

('30', '3', '1', '2020-01-24', '2020-01-30'),

('429', '1', '1', '2020-02-07', '2020-02-29'),

('155', '2', '4', '2020-02-05', '2020-02-20'),

('398', '1', '1', '2020-01-29', '2020-02-12'),

('255', '4', '4', '2020-01-26', '2020-02-03'),

('185', '3', '1', '2020-01-15', '2020-01-30'),

('309', '2', '3', '2020-01-26', '2020-01-30'),

('312', '1', '2', '2020-01-15', '2020-01-17'),

('376', '2', '4', '2020-01-15', '2020-01-24'),

('188', '1', '1', '2020-01-26', '2020-02-04'),

('138', '1', '1', '2020-02-09', '2020-02-23'),

('373', '4', '2', '2020-01-19', '2020-02-16'),

('361', '2', '2', '2020-01-28', '2020-02-23');

CREATE TABLE customer\_transactions (

customer\_id INTEGER primary key,

txn\_date DATE,

txn\_type VARCHAR(10),

txn\_amount INTEGER

);

INSERT INTO customer\_transactions (customer\_id, txn\_date, txn\_type, txn\_amount) VALUES

('429', '2020-01-21', 'deposit', '82'),

('155', '2020-01-10', 'deposit', '712'),

('398', '2020-01-01', 'deposit', '196'),

('255', '2020-01-14', 'deposit', '563'),

('185', '2020-01-29', 'deposit', '626'),

('309', '2020-01-13', 'deposit', '995'),

('312', '2020-01-20', 'deposit', '485'),

('376', '2020-01-03', 'deposit', '706'),

('188', '2020-01-13', 'deposit', '601'),

('138', '2020-01-11', 'deposit', '520'),

('373', '2020-01-18', 'deposit', '596'),

('361', '2020-01-12', 'deposit', '797'),

('169', '2020-01-10', 'deposit', '628'),

('402', '2020-01-05', 'deposit', '435'),

('60', '2020-01-19', 'deposit', '495'),

('378', '2020-01-07', 'deposit', '193'),

('383', '2020-01-26', 'deposit', '889'),

('292', '2020-01-10', 'deposit', '136'),

('63', '2020-01-06', 'deposit', '234'),

('499', '2020-01-02', 'deposit', '147'),

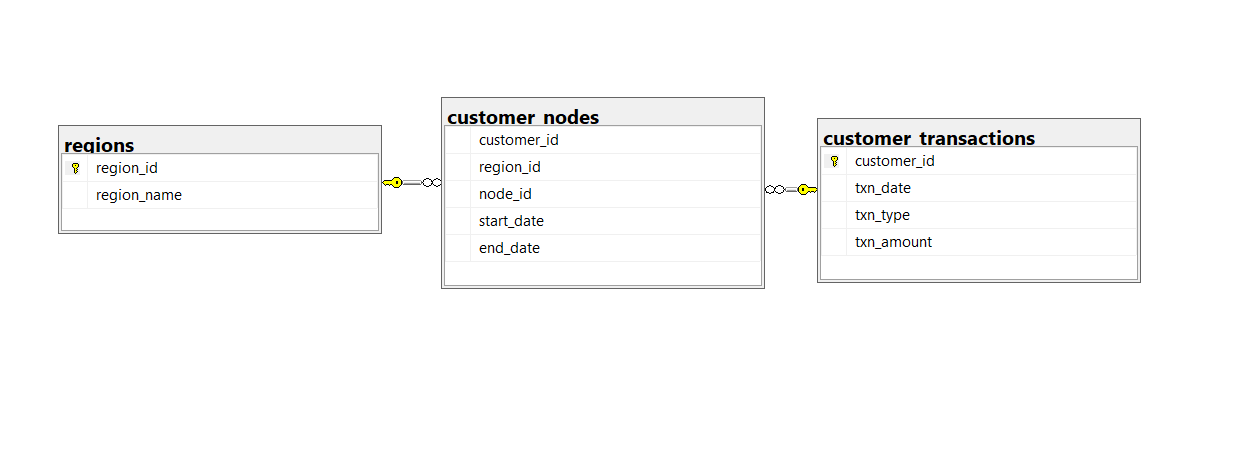
('130', '2020-01-02', 'deposit', '557'),

('441', '2020-01-12', 'deposit', '418'),

('53', '2020-01-24', 'deposit', '22'),

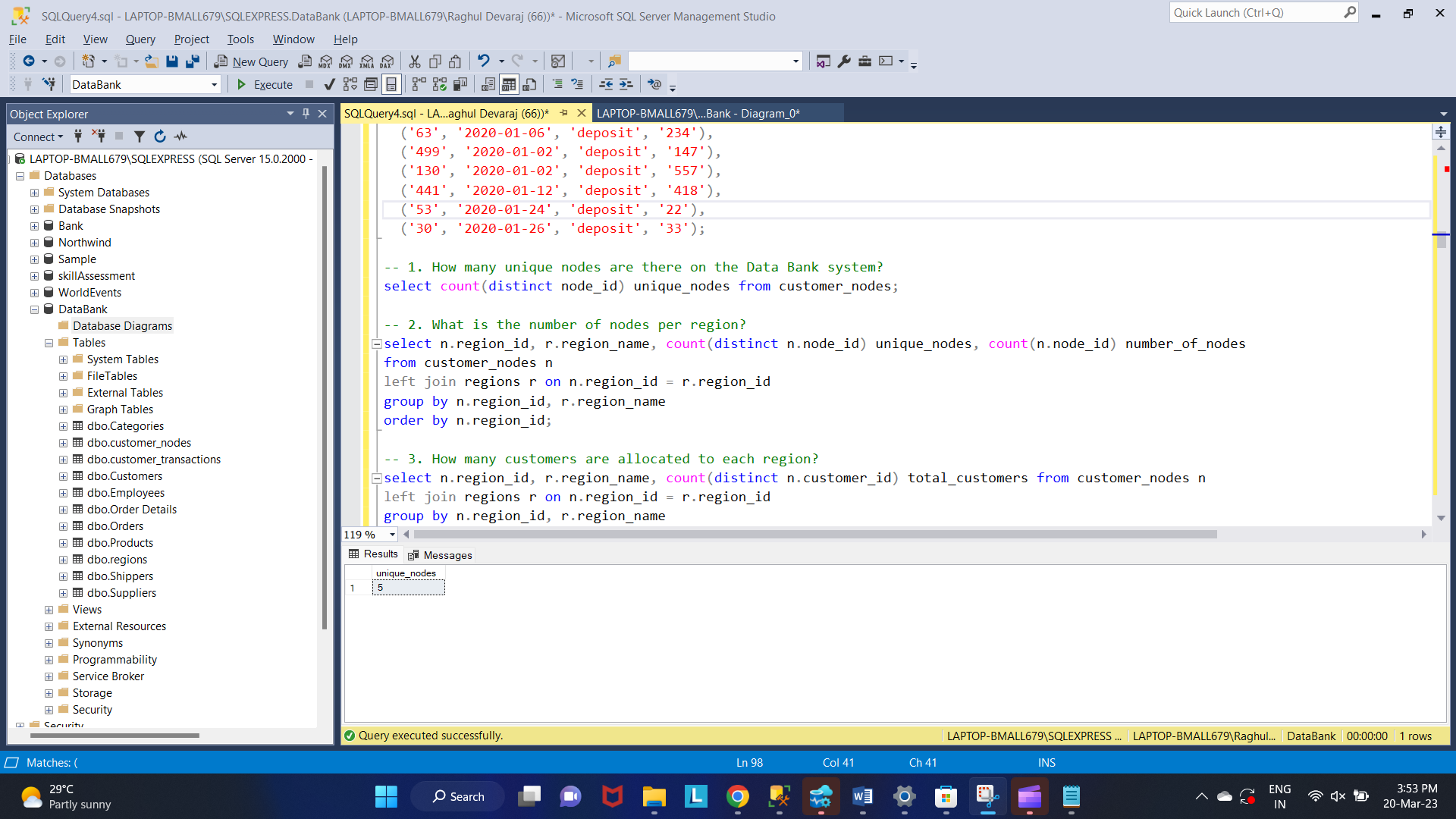
('30', '2020-01-26', 'deposit', '33');

ER DIAGRAM



Q1. How many unique nodes are there on the Data Bank system?

select count(distinct node\_id) unique\_nodes from customer\_nodes;



Q2. What is the number of nodes per region?

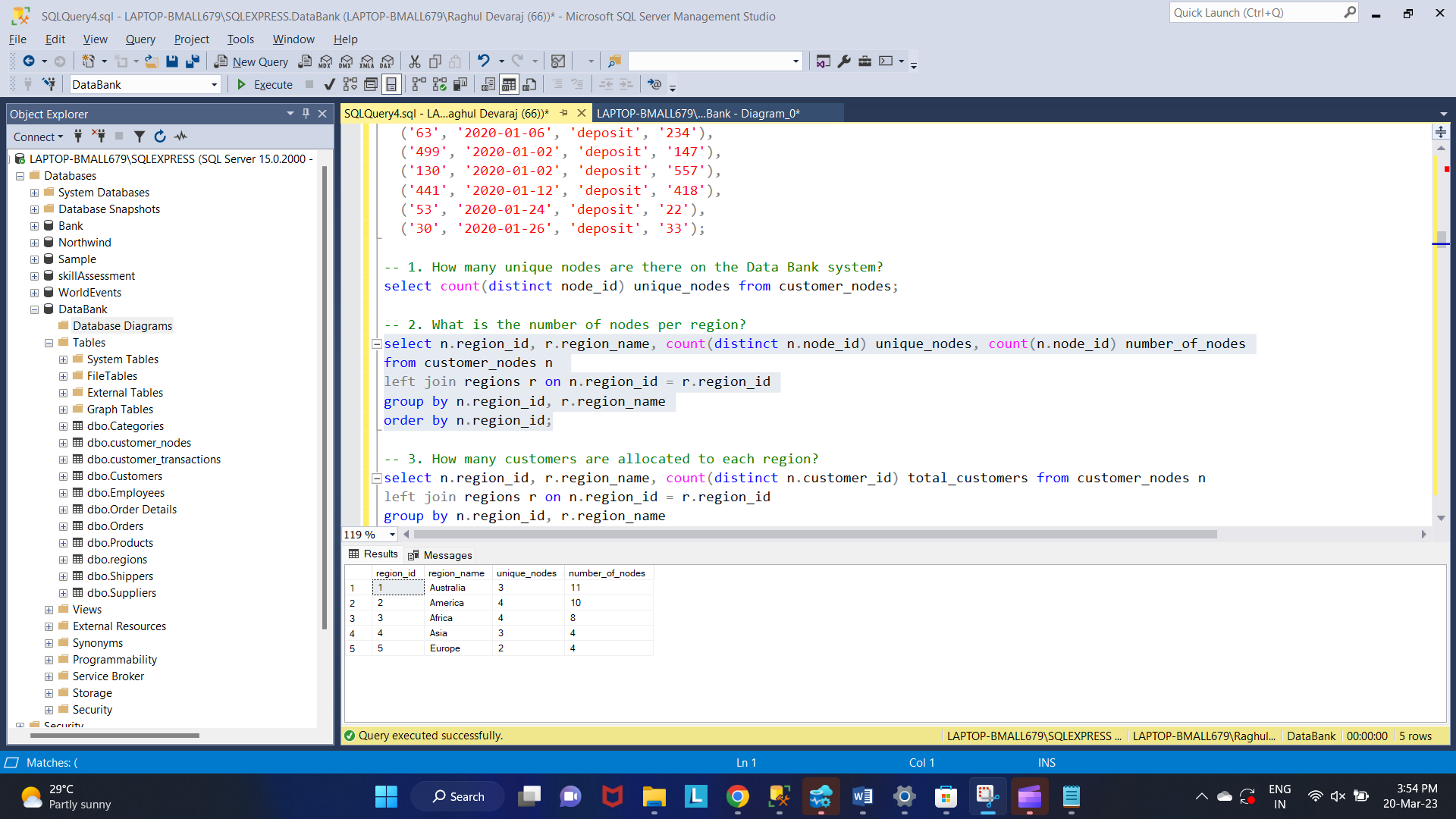
select n.region\_id, r.region\_name, count(distinct n.node\_id) unique\_nodes, count(n.node\_id) number\_of\_nodes

from customer\_nodes n

left join regions r on n.region\_id = r.region\_id

group by n.region\_id, r.region\_name

order by n.region\_id;



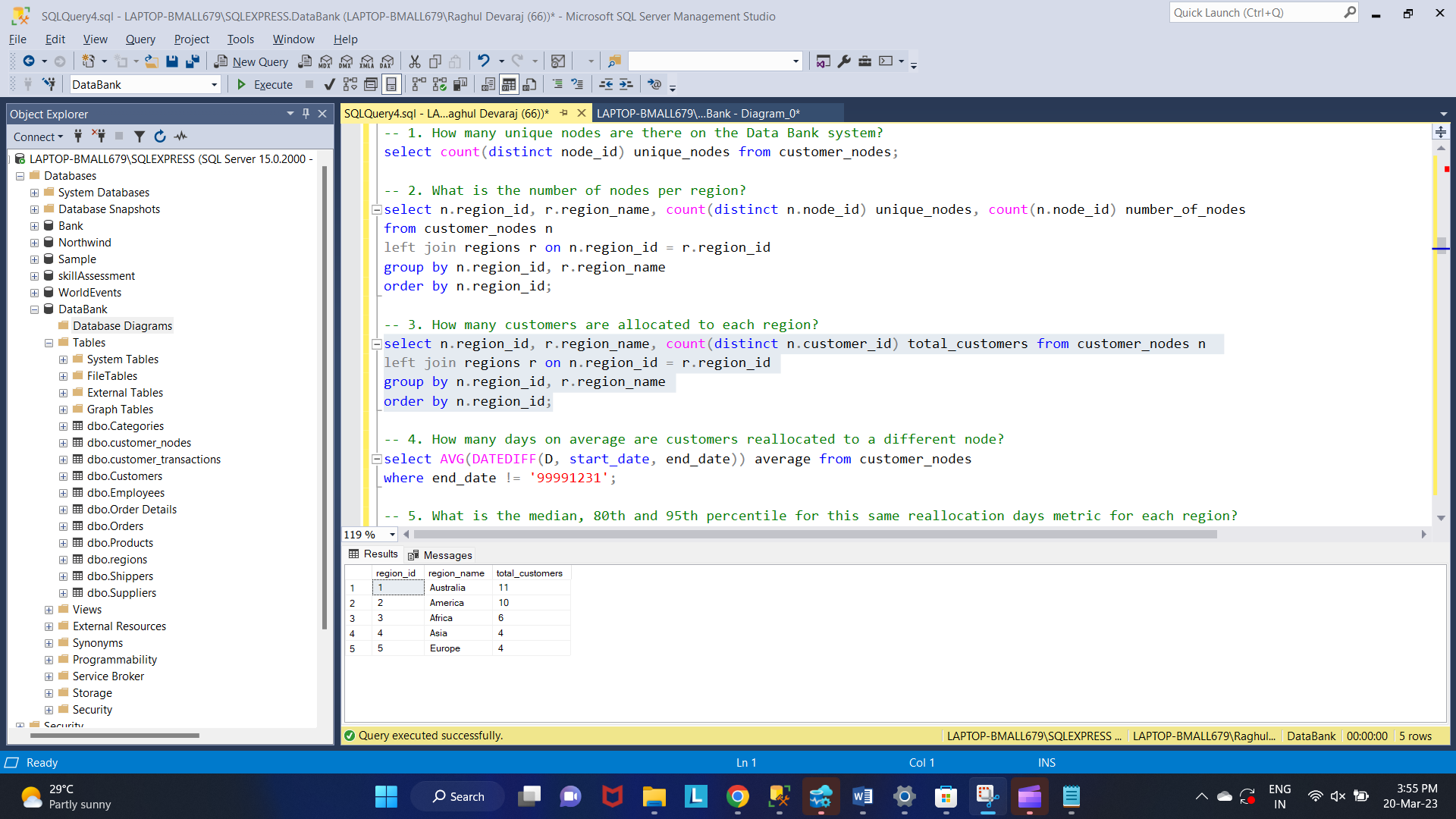
Q3. How many customers are allocated to each region?

select n.region\_id, r.region\_name, count(distinct n.customer\_id) total\_customers from customer\_nodes n

left join regions r on n.region\_id = r.region\_id

group by n.region\_id, r.region\_name

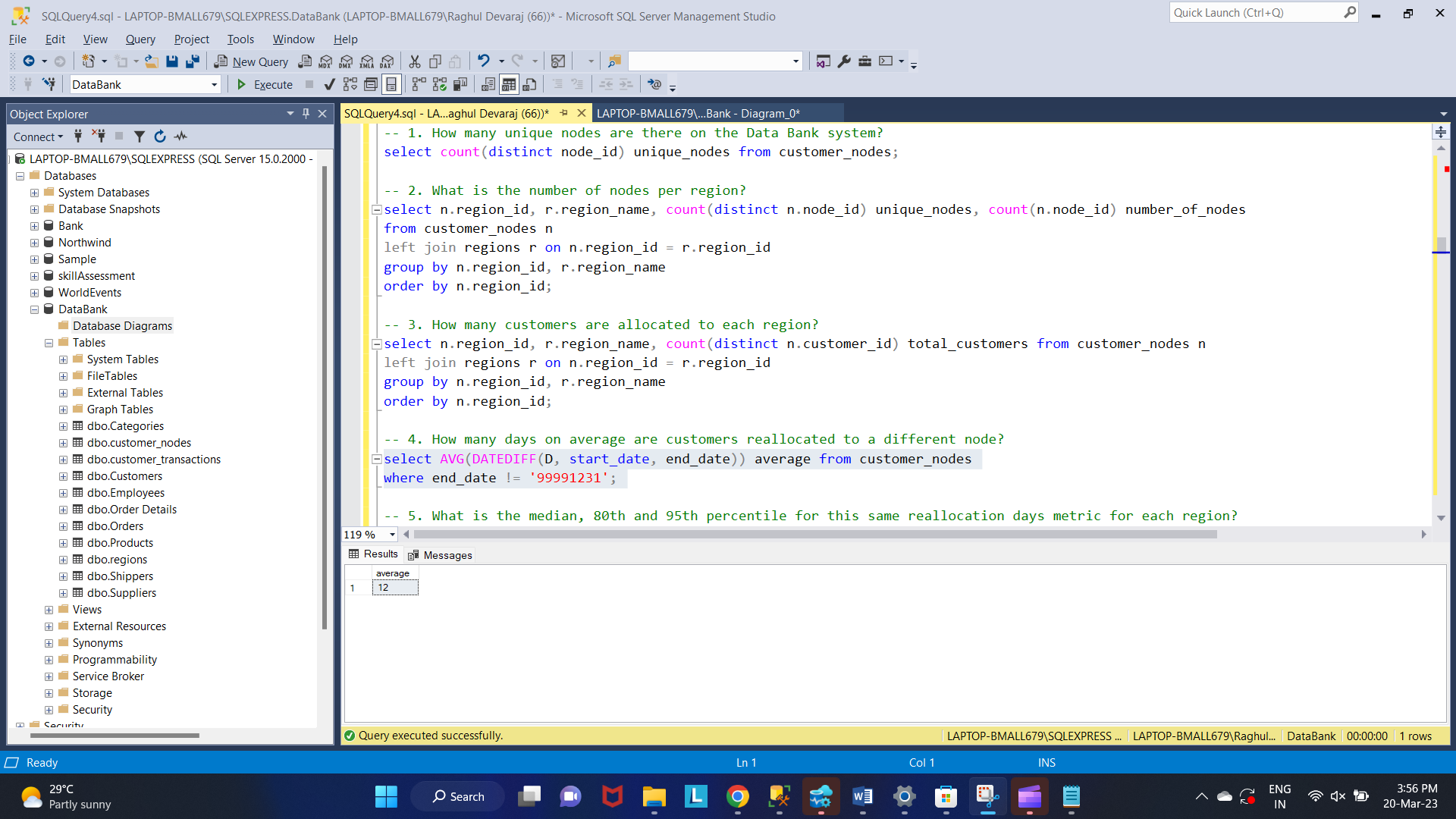
order by n.region\_id;



Q4. How many days on average are customers reallocated to a different node?

select AVG(DATEDIFF(D, start\_date, end\_date)) average from customer\_nodes

where end\_date != '99991231';



Q 5. What is the median, 80th and 95th percentile for this same reallocation days metric for each region?

WITH

diff\_data

AS

(

select

n.customer\_id,

n.region\_id,

r.region\_name,

DATEDIFF(D, n.start\_date, n.end\_date) diff

from customer\_nodes n

left join regions r on n.region\_id = r.region\_id

where end\_date != '99991231'

)

select distinct

region\_id,

region\_name,

PERCENTILE\_CONT(0.5) WITHIN GROUP (ORDER BY diff)

OVER (PARTITION BY region\_name) AS median,

PERCENTILE\_CONT(0.8) WITHIN GROUP (ORDER BY diff)

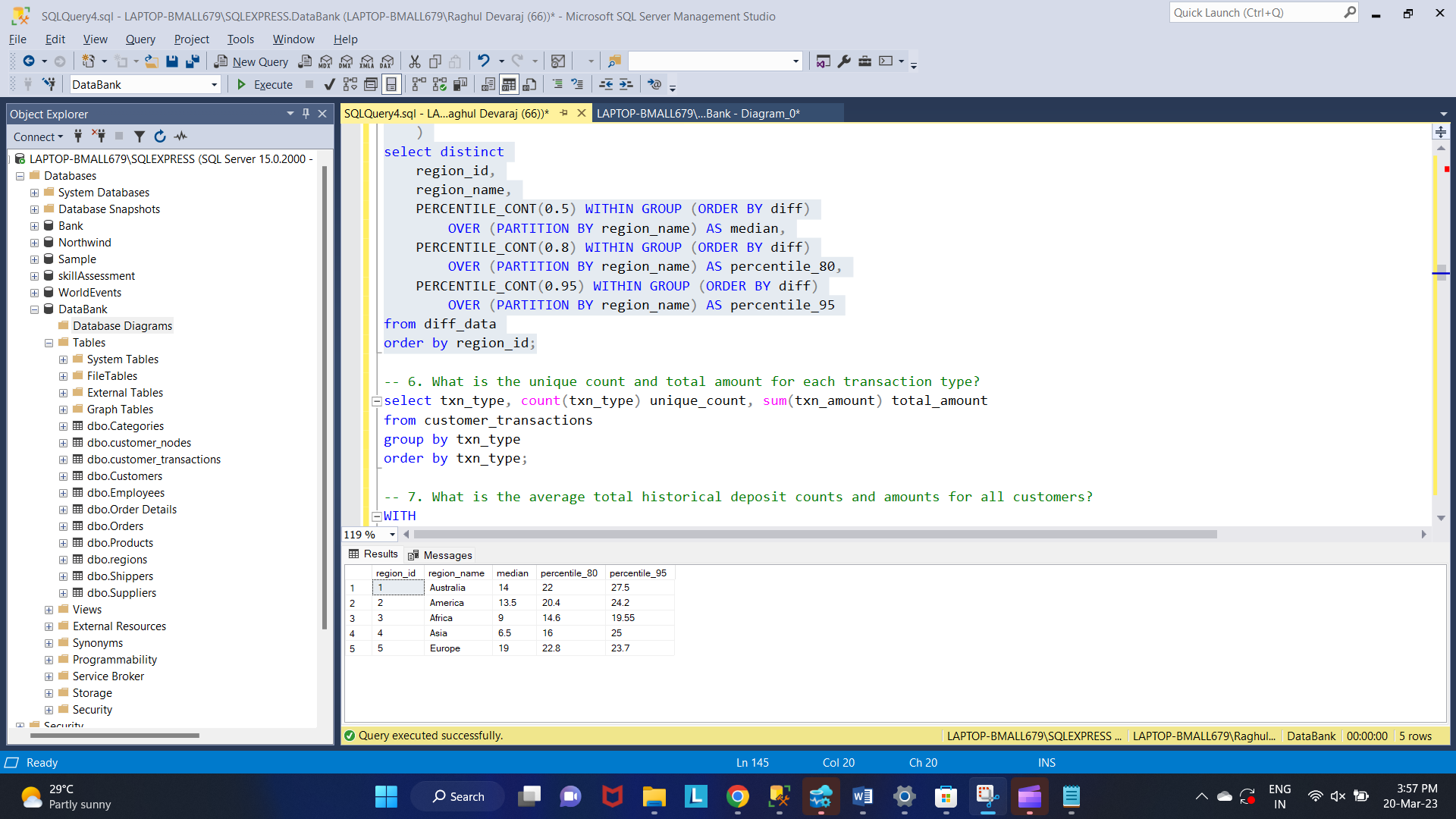
OVER (PARTITION BY region\_name) AS percentile\_80,

PERCENTILE\_CONT(0.95) WITHIN GROUP (ORDER BY diff)

OVER (PARTITION BY region\_name) AS percentile\_95

from diff\_data

order by region\_id;



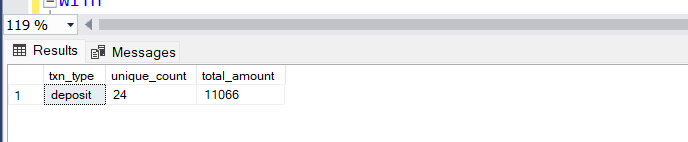
Q 6. What is the unique count and total amount for each transaction type?

select txn\_type, count(txn\_type) unique\_count, sum(txn\_amount) total\_amount

from customer\_transactions

group by txn\_type

order by txn\_type;



Q 7. What is the average total historical deposit counts and amounts for all customers?

WITH

historical

AS

(

select

n.customer\_id,

t.txn\_type,

count(t.txn\_type) count,

avg(t.txn\_amount) total\_amount

from customer\_transactions t

left join customer\_nodes n on t.customer\_id = n.customer\_id

left join regions r on n.region\_id = r.region\_id

group by n.customer\_id, t.txn\_type

)

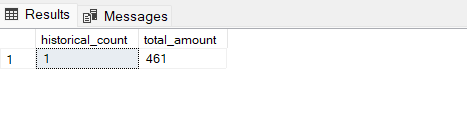
select

avg(count) historical\_count,

avg(total\_amount) total\_amount

from historical

where txn\_type = 'deposit';



Q 8. For each month - how many Data Bank customers make more than 1 deposit and either 1 purchase or 1 withdrawal in a single month?

WITH

historical --count data each type transactions

AS

(

select

n.customer\_id,

DATEPART(M, t.txn\_date) month\_id,

DATENAME(M, t.txn\_date) month\_name,

count(t.txn\_type) total

from customer\_transactions t

left join customer\_nodes n on t.customer\_id = n.customer\_id

left join regions r on n.region\_id = r.region\_id

group by n.customer\_id, DATEPART(M, t.txn\_date), DATENAME(M, t.txn\_date)

),

deposit

AS

(

select

n.customer\_id,

DATEPART(M, t.txn\_date) month\_id,

DATENAME(M, t.txn\_date) month\_name,

sum(case when t.txn\_type = 'deposit' then 1 else 0 end) deposit

from customer\_transactions t

left join customer\_nodes n on t.customer\_id = n.customer\_id

group by n.customer\_id, DATEPART(M, t.txn\_date), DATENAME(M, t.txn\_date)

),

purchase

AS

(

select

n.customer\_id,

DATEPART(M, t.txn\_date) month\_id,

sum(case when t.txn\_type = 'purchase' then 1 else 0 end) purchase

from customer\_transactions t

left join customer\_nodes n on t.customer\_id = n.customer\_id

group by n.customer\_id, DATEPART(M, t.txn\_date)

),

withdrawal -- type transactions = withdrawal

AS

(

select

n.customer\_id,

DATEPART(M, t.txn\_date) month\_id,

sum(case when t.txn\_type = 'withdrawal' then 1 else 0 end) withdrawal

from customer\_transactions t

left join customer\_nodes n on t.customer\_id = n.customer\_id

group by n.customer\_id, DATEPART(M, t.txn\_date)

),

data -- join all data

AS

(

select

h.customer\_id,

h.month\_id,

h.month\_name,

h.total,

d.deposit,

p.purchase,

w.withdrawal

from historical h

left join deposit d on h.customer\_id = d.customer\_id and h.month\_id = d.month\_id

left join purchase p on h.customer\_id = p.customer\_id and h.month\_id = p.month\_id

left join withdrawal w on h.customer\_id = w.customer\_id and h.month\_id = w.month\_id

)

select

month\_id,

month\_name,

COUNT(customer\_id) customer\_count

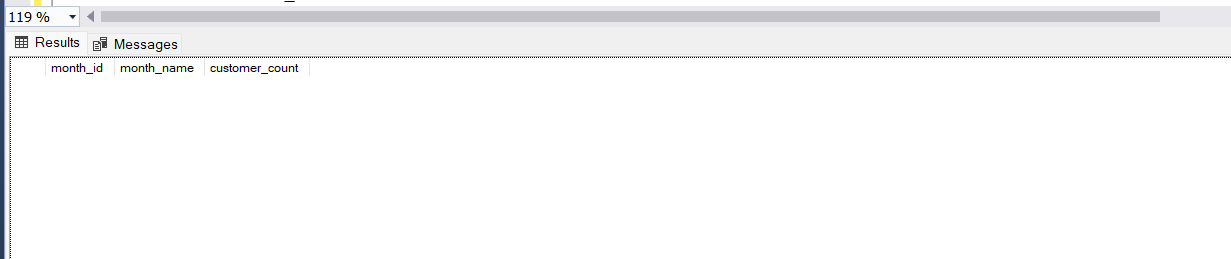
from data

where deposit > 1

and (purchase >= 1 or withdrawal >= 1)

group by month\_id, month\_name

order by month\_id;



Q 9. What is the closing balance for each customer at the end of the month?

WITH

first\_month

AS

(

SELECT

customer\_id,

CAST('20200131' as date) closing\_date,

MIN(DATEPART(M, txn\_date)) min\_month,

MAX(DATEPART(M, txn\_date)) max\_month

from customer\_transactions

group by customer\_id

),

months --recursive function (for closing\_date)

AS

(

SELECT

customer\_id,

closing\_date,

DATEPART(M, closing\_date) month\_id,

DATENAME(M, closing\_date) month\_name

, min\_month, max\_month

FROM first\_month

UNION ALL

SELECT

customer\_id,

DATEADD(M, 1, closing\_date) closing\_date,

DATEPART(M, DATEADD(M, 1, closing\_date)) closing\_id,

DATENAME(M, DATEADD(M, 1, closing\_date)) closing\_name

, min\_month, max\_month

FROM months b

WHERE closing\_date <= CAST('20200401' as date)

),

balance --count data each type transactions

AS

(

select

customer\_id,

DATEPART(M, txn\_date) month\_id,

DATENAME(M, txn\_date) month\_name,

sum(case when txn\_type in ('purchase','withdrawal') then -txn\_amount

else txn\_amount end) txn\_amount

from customer\_transactions

group by customer\_id, DATEPART(M, txn\_date), DATENAME(M, txn\_date)

)

select

m.customer\_id,

m.month\_id,

m.month\_name,

SUM(txn\_amount) OVER(PARTITION BY m.customer\_id ORDER BY m.month\_id

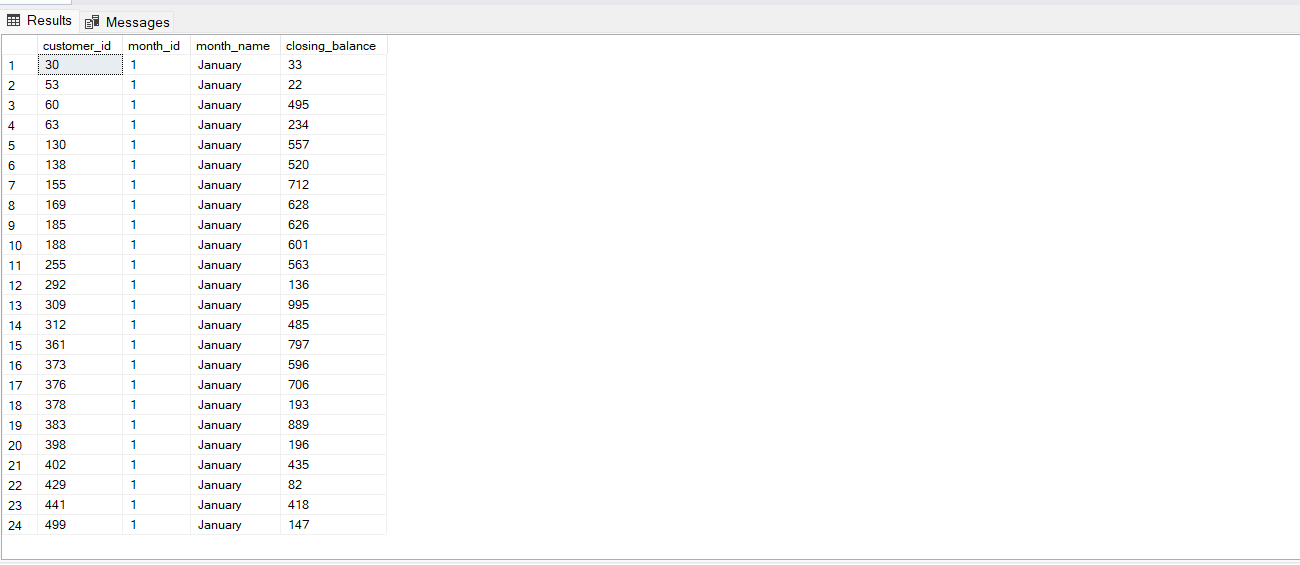
ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) closing\_balance

from months m

left join balance b on b.customer\_id = m.customer\_id and b.month\_id = m.month\_id

where m.month\_id between min\_month and max\_month

ORDER BY m.customer\_id, m.month\_id;



Qa10. What is the percentage of customers who increase their closing balance by more than 5%?

WITH

first\_month

AS

(

SELECT

customer\_id,

CAST('20200131' as date) closing\_date,

MIN(DATEPART(M, txn\_date)) min\_month,

MAX(DATEPART(M, txn\_date)) max\_month

from customer\_transactions

group by customer\_id

),

months --recursive function (for closing\_date)

AS

(

SELECT

customer\_id,

closing\_date,

DATEPART(M, closing\_date) month\_id,

DATENAME(M, closing\_date) month\_name

, min\_month, max\_month

FROM first\_month

UNION ALL

SELECT

customer\_id,

DATEADD(M, 1, closing\_date) closing\_date,

DATEPART(M, DATEADD(M, 1, closing\_date)) closing\_id,

DATENAME(M, DATEADD(M, 1, closing\_date)) closing\_name

, min\_month, max\_month

FROM months b

WHERE closing\_date <= CAST('20200401' as date)

),

balance --count data each type transactions

AS

(

select

customer\_id,

DATEPART(M, txn\_date) month\_id,

DATENAME(M, txn\_date) month\_name,

sum(case when txn\_type in ('purchase','withdrawal') then -txn\_amount

else txn\_amount end) txn\_amount

from customer\_transactions

group by customer\_id, DATEPART(M, txn\_date), DATENAME(M, txn\_date)

),

closing\_balances --first and closing balances

AS

(

select

m.customer\_id,

m.month\_id,

m.month\_name,

SUM(txn\_amount) OVER(PARTITION BY m.customer\_id ORDER BY m.month\_id

ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) closing\_balance

from months m

left join balance b on b.customer\_id = m.customer\_id and b.month\_id = m.month\_id

where m.month\_id between min\_month and max\_month

),

balances --first balances

AS

(

select

customer\_id,

month\_id,

month\_name,

coalesce(LAG(closing\_balance) OVER(PARTITION BY customer\_id ORDER BY month\_id),0) opening\_balance,

closing\_balance

from closing\_balances

),

cases --closing - opening balance

AS

(

select

customer\_id,

month\_id,

month\_name,

opening\_balance,

closing\_balance,

case when opening\_balance is null then cast((closing\_balance - 0) as float)

else cast((closing\_balance - opening\_balance) as float) end diff

from balances

),

percents --percentage increase

AS

(

select \*,

case when opening\_balance = 0 then round(cast(diff/1\*100 as float), 2)

else round(cast(diff/opening\_balance\*100 as float), 2) end percentage

from cases

),

minimum --when balance null then 0

AS

(

select \*,

MIN(percentage) OVER(PARTITION BY customer\_id) mins

from percents

)

select ROUND(100 \* CAST(COUNT(customer\_id) as float) /

(select count(\*) from customer\_transactions), 2) percentage\_of\_customers

from minimum

where mins > 5;

