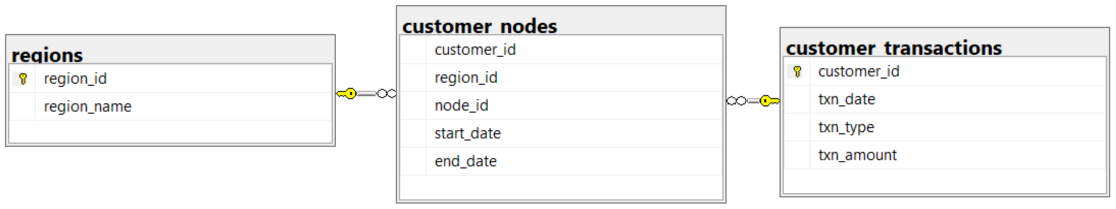
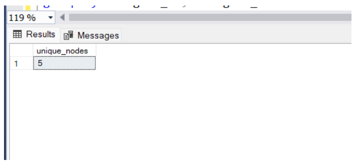
SQL SERVER CASE STUDY

```
use 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

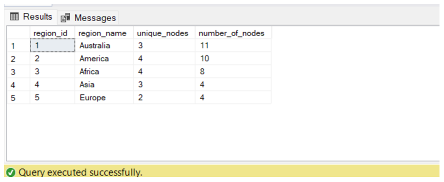


```
-- 1. How many unique nodes are there on the Data Bank system?

select count(distinct node_id) unique_nodes from customer_nodes;
```



```
   unique_nodes
1  5
```

```sql
-- 2. 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;
```

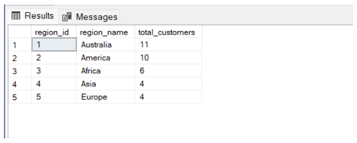


Results | Messages

	region_id	region_name	unique_nodes	number_of_nodes
1	1	Australia	3	11
2	2	America	4	10
3	3	Africa	4	8
4	4	Asia	3	4
5	5	Europe	2	4

Query executed successfully.

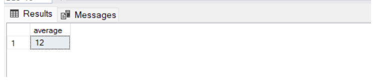
```sql
-- 3. How many customers are allocated to each region?
select n.region_id, r.region_name, count(distinct n.customer_id) total_customers from
customer_nodes n
left join regions r on n.region_id = r.region_id
group by n.region_id, r.region_name
order by n.region_id;
```



Results | Messages

	region_id	region_name	total_customers
1	1	Australia	11
2	2	America	10
3	3	Africa	6
4	4	Asia	4
5	5	Europe	4

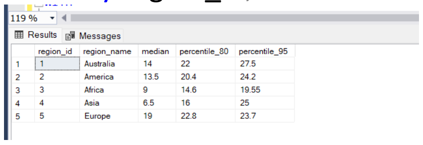
```sql
-- 4. How many days on average are customers reallocated to a different node?
select AVG(DATEDIFF(D, start_date, end_date)) average from customer_nodes
where end_date != '99991231';
```



Results | Messages

	average
1	12

```sql
-- 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;
```

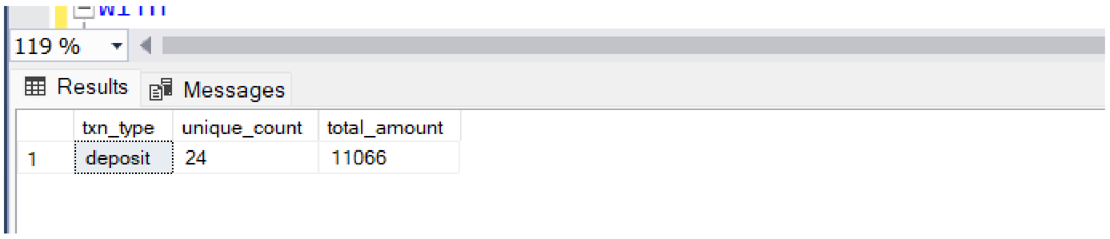


119 %

Results | Messages

	region_id	region_name	median	percentile_80	percentile_95
1	1	Australia	14	22	27.5
2	2	America	13.5	20.4	24.2
3	3	Africa	9	14.6	19.55
4	4	Asia	6.5	16	25
5	5	Europe	19	22.8	23.7

```
-- 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;
```

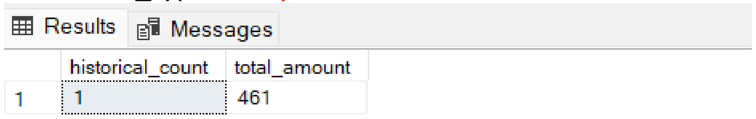


119 %

Results | Messages

	txn_type	unique_count	total_amount
1	deposit	24	11066

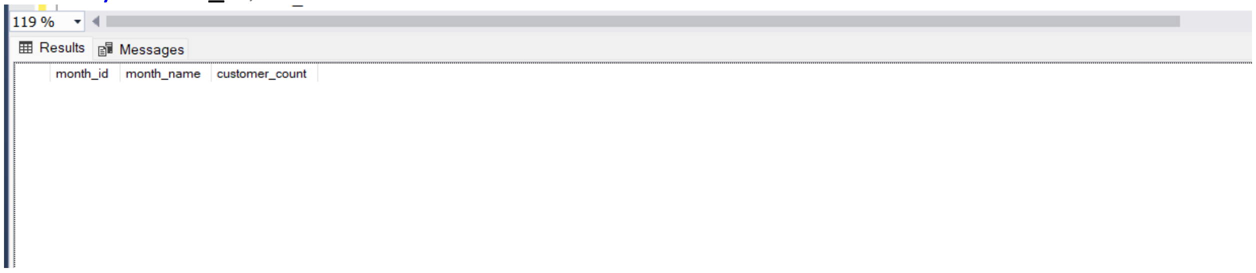
```
-- 7. What is the average total historical deposit counts and amounts for all customers?
WITH
		historical
AS
		(
				select
					n.customer_id,
					t.txn_type,
					count(t.txn_type) count,
					avg(t.txn_amount) total_amount
				from customer_transactions t
				left join customer_nodes n on t.customer_id = 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';
```



```
   historical_count  total_amount
1  1                 461
```

```
-- 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 -- type transactions = 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 -- type transactions = 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
```

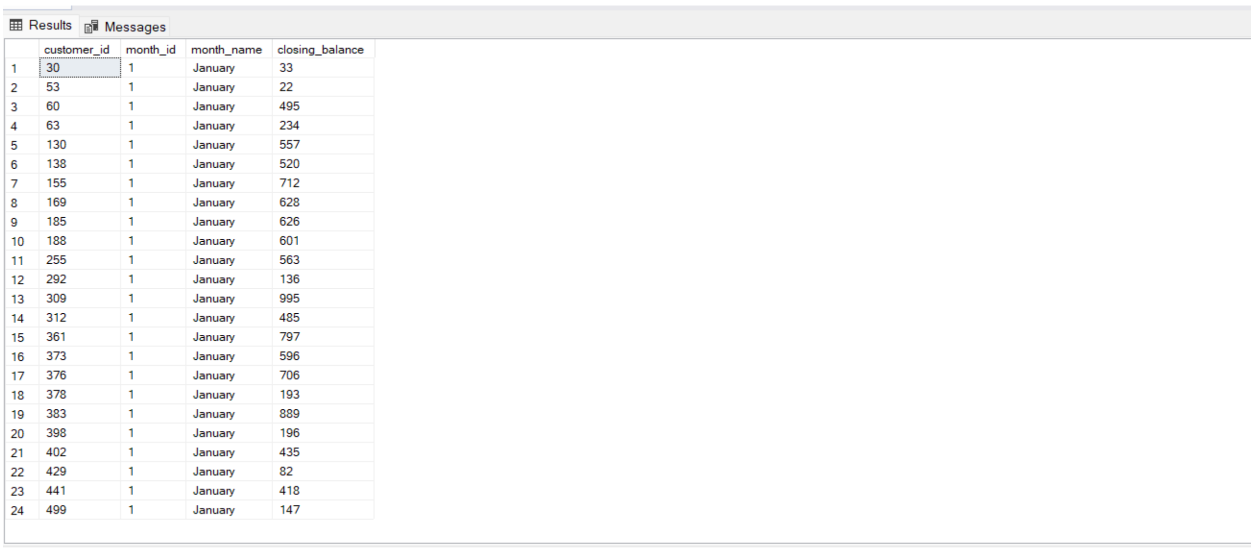
```
		left join withdrawal w on h.customer_id = w.customer_id and h.month_id =
w.month_id
	)
select
	month_id,
	month_name,
	COUNT(customer_id) customer_count
from data
where deposit > 1
	and (purchase >= 1 or withdrawal >= 1)
group by month_id, month_name
order by month_id;
```



```
month_id  month_name  customer_count
```

```
-- 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
```

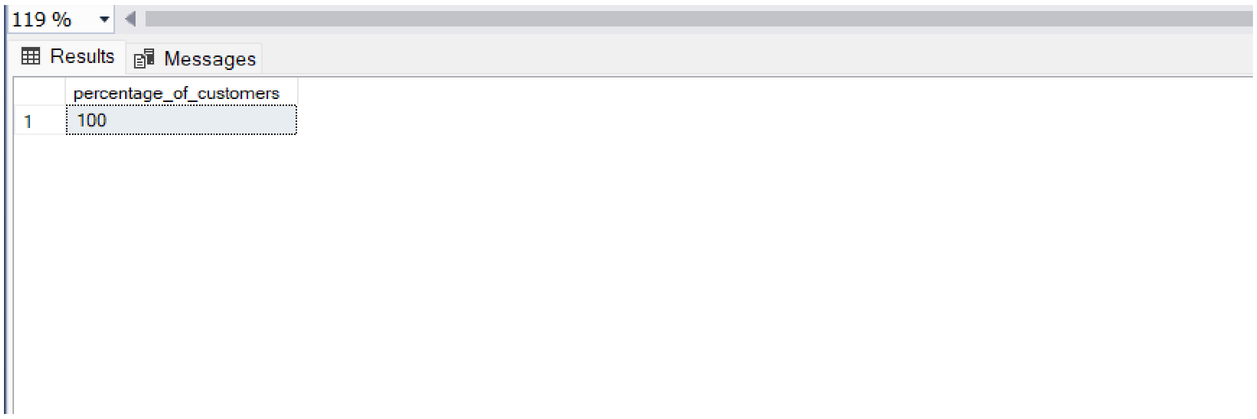
```
                FROM first_month

                        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;
```


Results Messages

	customer_id	month_id	month_name	closing_balance
1	30	1	January	33
2	53	1	January	22
3	60	1	January	495
4	63	1	January	234
5	130	1	January	557
6	138	1	January	520
7	155	1	January	712
8	169	1	January	628
9	185	1	January	626
10	188	1	January	601
11	255	1	January	563
12	292	1	January	136
13	309	1	January	995
14	312	1	January	485
15	361	1	January	797
16	373	1	January	596
17	376	1	January	706
18	378	1	January	193
19	383	1	January	889
20	398	1	January	196
21	402	1	January	435
22	429	1	January	82
23	441	1	January	418
24	499	1	January	147

```
-- 10. 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;
```



```
   percentage_of_customers
1  100
```