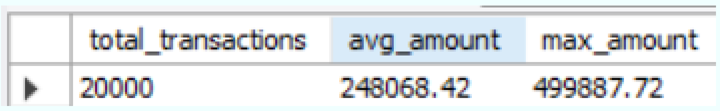
SQL project to detect fraud patterns, laundering detection and suspicious activity analysis in financial transactions

# 1. Transaction Overview
Total number of transactions, average transaction amount, and maximum amount.
Uses COUNT, AVG, and MAX on the amount column to summarize the dataset.

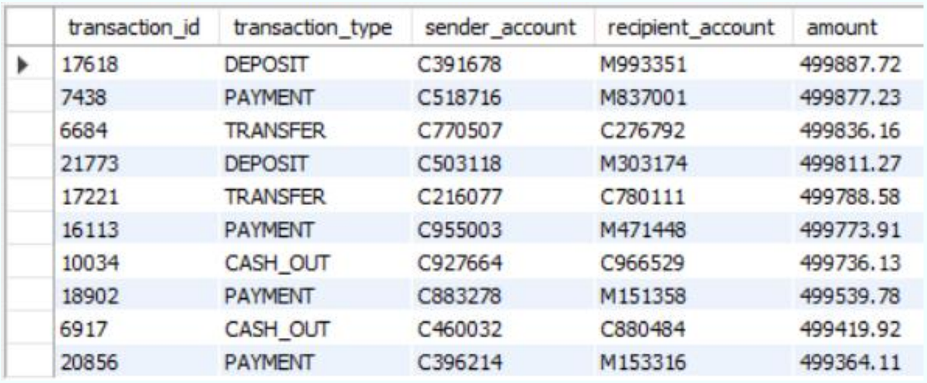
```sql
SELECT
    COUNT(*) AS total_transactions,
    ROUND(AVG(amount), 2) AS avg_amount,
    ROUND(MAX(amount), 2) AS max_amount
FROM transactions;
```



total_transactions	avg_amount	max_amount
20000	248068.42	499887.72

# 2. Largest Transactions (Top 10)
The top 10 largest single transactions.
Orders all transactions by amount DESC and takes the first 10.

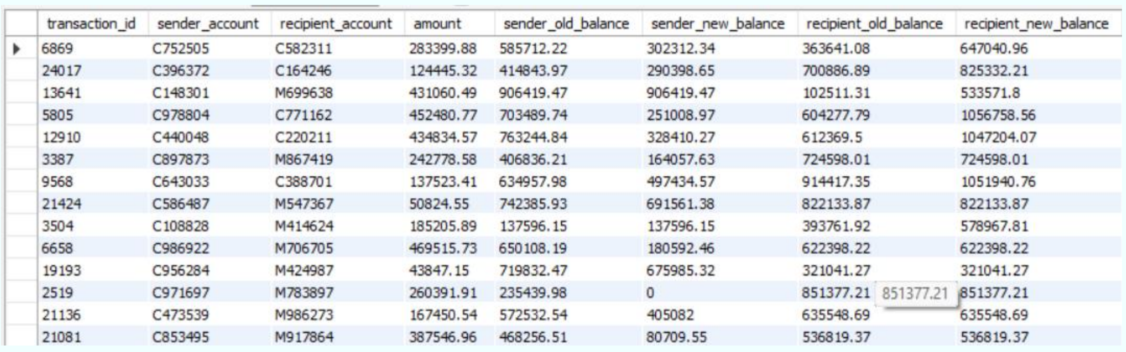
```sql
SELECT transaction_id, transaction_type, sender_account, recipient_account, amount
FROM transactions
ORDER BY amount DESC
LIMIT 10;
```



transaction_id	transaction_type	sender_account	recipient_account	amount
17618	DEPOSIT	C391678	M993351	499887.72
7438	PAYMENT	C518716	M837001	499877.23
6684	TRANSFER	C770507	C276792	499836.16
21773	DEPOSIT	C503118	M303174	499811.27
17221	TRANSFER	C216077	C780111	499788.58
16113	PAYMENT	C955003	M471448	499773.91
10034	CASH_OUT	C927664	C966529	499736.13
18902	PAYMENT	C883278	M151358	499539.78
6917	CASH_OUT	C460032	C880484	499419.92
20856	PAYMENT	C396214	M153316	499364.11

# 3. Transactions with Balance Mismatch
Checking if balances updating correctly after transactions?
Transactions where balances don’t match expected math
(old_balance - amount = new_balance).

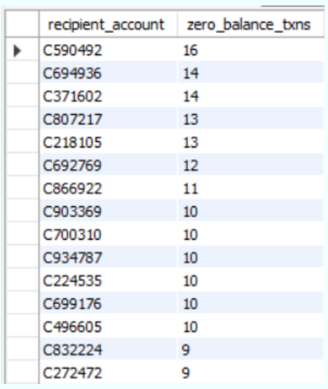
```
SELECT transaction_id, sender_account, recipient_account, amount,
       sender_old_balance, sender_new_balance,
       recipient_old_balance, recipient_new_balance
FROM transactions
WHERE (sender_old_balance - amount <> sender_new_balance)
   OR (recipient_old_balance + amount <> recipient_new_balance);
```



transaction_id	sender_account	recipient_account	amount	sender_old_balance	sender_new_balance	recipient_old_balance	recipient_new_balance
6869	C752505	C582311	283399.88	585712.22	302312.34	363641.08	647040.96
24017	C396372	C164246	124445.32	414843.97	290398.65	700886.89	825332.21
13641	C148301	M699638	431060.49	906419.47	906419.47	102511.31	533571.8
5805	C978804	C771162	452480.77	703489.74	251008.97	604277.79	1056758.56
12910	C440048	C220211	434834.57	763244.84	328410.27	612369.5	1047204.07
3387	C897873	M867419	242778.58	406836.21	164057.63	724598.01	724598.01
9568	C643033	C388701	137523.41	634957.98	497434.57	914417.35	1051940.76
21424	C586487	M547367	50824.55	742385.93	691561.38	822133.87	822133.87
3504	C108828	M414624	185205.89	137596.15	137596.15	393761.92	578967.81
6658	C986922	M706705	469515.73	650108.19	180592.46	622398.22	622398.22
19193	C956284	M424987	43847.15	719832.47	675985.32	321041.27	321041.27
2519	C971697	M783897	260391.91	235439.98	0	851377.21	851377.21
21136	C473539	M986273	167450.54	572532.54	405082	635548.69	635548.69
21081	C853495	M917864	387546.96	468256.51	80709.55	536819.37	536819.37

# 4. Zero-Balance Recipients (Mule Detection)
Recipient accounts that repeatedly show zero balances → suspicious “mule” accounts.
Filters for recipient_old_balance = 0 AND recipient_new_balance = 0
and groups by recipient.

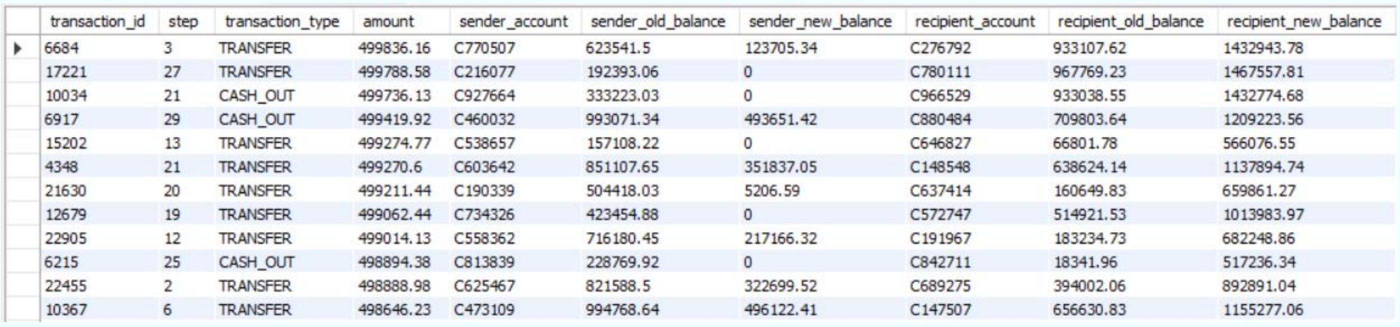
```
SELECT recipient_account, COUNT(*) AS zero_balance_txns
FROM transactions
WHERE recipient_old_balance = 0 AND recipient_new_balance = 0
GROUP BY recipient_account
HAVING COUNT(*) > 5
ORDER BY zero_balance_txns DESC;
```



recipient_account	zero_balance_txns
C590492	16
C694936	14
C371602	14
C807217	13
C218105	13
C692769	12
C866922	11
C903369	10
C700310	10
C934787	10
C224535	10
C699176	10
C496605	10
C832224	9
C272472	9

# 5. Suspiciously Large Transfers
Transfers or cash-outs greater than 200,000.
Use filter on amount > 200000 for TRANSFER and CASH_OUT.

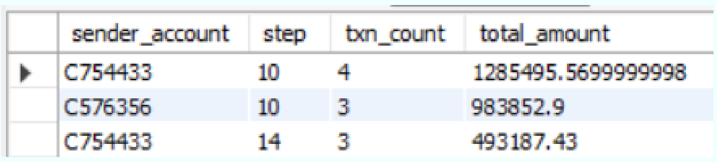
```sql
SELECT *
FROM transactions
WHERE transaction_type IN ('TRANSFER', 'CASH_OUT')
  AND amount > 200000
ORDER BY amount DESC;
```



transaction_id	step	transaction_type	amount	sender_account	sender_old_balance	sender_new_balance	recipient_account	recipient_old_balance	recipient_new_balance
6684	3	TRANSFER	499836.16	C770507	623541.5	123705.34	C276792	933107.62	1432943.78
17221	27	TRANSFER	499788.58	C216077	192393.06	0	C780111	967769.23	1467557.81
10034	21	CASH_OUT	499736.13	C927664	333223.03	0	C966529	933038.55	1432774.68
6917	29	CASH_OUT	499419.92	C460032	993071.34	493651.42	C880484	709803.64	1209223.56
15202	13	TRANSFER	499274.77	C538657	157108.22	0	C646827	66801.78	566076.55
4348	21	TRANSFER	499270.6	C603642	851107.65	351837.05	C148548	638624.14	1137894.74
21630	20	TRANSFER	499211.44	C190339	504418.03	5206.59	C637414	160649.83	659861.27
12679	19	TRANSFER	499062.44	C734326	423454.88	0	C572747	514921.53	1013983.97
22905	12	TRANSFER	499014.13	C558362	716180.45	217166.32	C191967	183234.73	682248.86
6215	25	CASH_OUT	498894.38	C813839	228769.92	0	C842711	18341.96	517236.34
22455	2	TRANSFER	498888.98	C625467	821588.5	322699.52	C689275	394002.06	892891.04
10367	6	TRANSFER	498646.23	C473109	994768.64	496122.41	C147507	656630.83	1155277.06

# 6. High-Frequency Transfers
Accounts that do multiple transfers in the same step (possible structuring).
Groups by sender_account and step, counts transfers, and flags those with high counts.

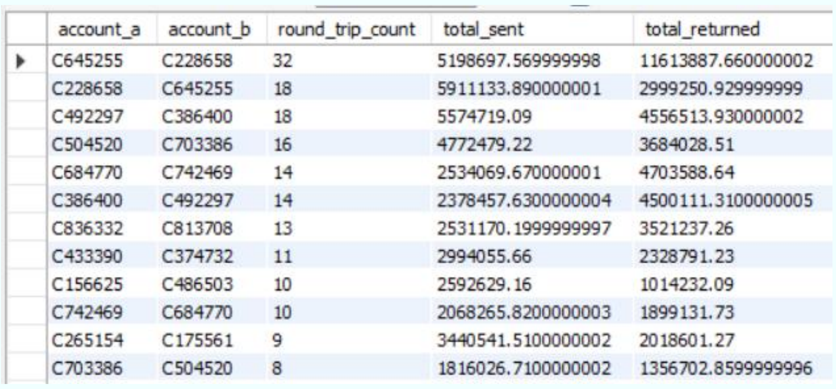
```sql
SELECT sender_account, step, COUNT(*) AS txn_count, SUM(amount) AS total_amount
FROM transactions
WHERE transaction_type = 'TRANSFER'
GROUP BY sender_account, step
HAVING COUNT(*) > 2
ORDER BY txn_count DESC;
```



sender_account	step	txn_count	total_amount
C754433	10	4	1285495.5699999998
C576356	10	3	983852.9
C754433	14	3	493187.43

# 7. Two-Step Laundering (Sender → Recipient → Sender)
Pairs of accounts that have “round-trip” transactions (classic laundering).
Joins the transactions table to itself (JOIN) where sender and recipient swap roles.

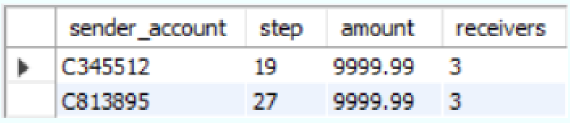
```
SELECT
    t1.sender_account AS account_a,
    t1.recipient_account AS account_b,
    COUNT(*) AS round_trip_count,
    SUM(t1.amount) AS total_sent,
    SUM(t2.amount) AS total_returned
FROM transactions t1
JOIN transactions t2
  ON t1.sender_account = t2.recipient_account
 AND t1.recipient_account = t2.sender_account
 AND t1.transaction_id < t2.transaction_id
GROUP BY t1.sender_account, t1.recipient_account
ORDER BY round_trip_count DESC, total_sent DESC;
```



	account_a	account_b	round_trip_count	total_sent	total_returned
▶	C645255	C228658	32	5198697.569999998	11613887.660000002
	C228658	C645255	18	5911133.890000001	2999250.929999999
	C492297	C386400	18	5574719.09	4556513.930000002
	C504520	C703386	16	4772479.22	3684028.51
	C684770	C742469	14	2534069.670000001	4703588.64
	C386400	C492297	14	2378457.6300000004	4500111.3100000005
	C836332	C813708	13	2531170.1999999997	3521237.26
	C433390	C374732	11	2994055.66	2328791.23
	C156625	C486503	10	2592629.16	1014232.09
	C742469	C684770	10	2068265.8200000003	1899131.73
	C265154	C175561	9	3440541.5100000002	2018601.27
	C703386	C504520	8	1816026.7100000002	1356702.8599999996

# 8. Structuring (Same Amount Sent to Many Accounts)
Accounts sending the same amount to multiple different recipients in the same step.
Groups by sender_account, step, and amount; flags when one amount goes to multiple recipients.

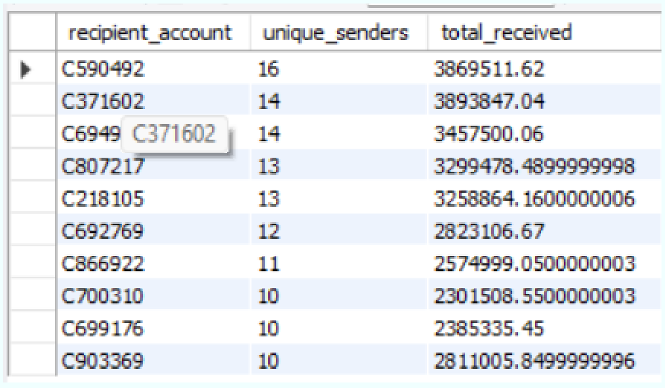
```
SELECT sender_account, step, amount, COUNT(DISTINCT recipient_account) AS receivers
FROM transactions
GROUP BY sender_account, step, amount
HAVING COUNT(DISTINCT recipient_account) > 2
ORDER BY receivers DESC;
```



	sender_account	step	amount	receivers
▶	C345512	19	9999.99	3
	C813895	27	9999.99	3

# 9. Top 10 Accounts Receiving from Many Different Senders
Recipients that receive from many unique senders (possible fraud hub).
Groups by recipient_account and counts unique senders.

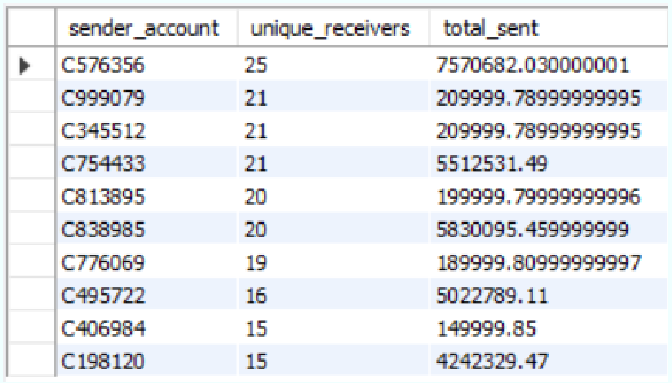
```sql
SELECT recipient_account, COUNT(DISTINCT sender_account) AS unique_senders, SUM(amount) AS
total_received
FROM transactions
GROUP BY recipient_account
ORDER BY unique_senders DESC
LIMIT 10;
```



recipient_account	unique_senders	total_received
C590492	16	3869511.62
C371602	14	3893847.04
C6949 C371602	14	3457500.06
C807217	13	3299478.4899999998
C218105	13	3258864.1600000006
C692769	12	2823106.67
C866922	11	2574999.0500000003
C700310	10	2301508.5500000003
C699176	10	2385335.45
C903369	10	2811005.8499999996

# 10. Top 10 Accounts Sending to Many Different Receivers.
Senders that transfer to many different recipients (possible layering).
Groups by sender_account and counts unique recipients.

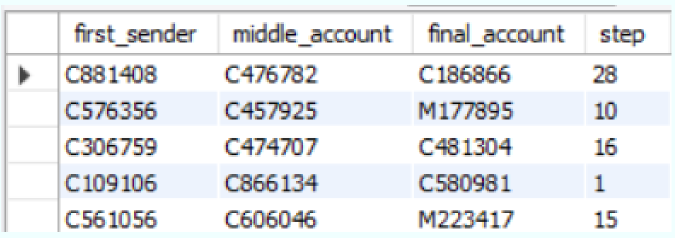
```sql
SELECT sender_account, COUNT(DISTINCT recipient_account) AS unique_receivers, SUM(amount)
AS total_sent
FROM transactions
GROUP BY sender_account
ORDER BY unique_receivers DESC
LIMIT 10;
```



sender_account	unique_receivers	total_sent
C576356	25	7570682.030000001
C999079	21	209999.78999999995
C345512	21	209999.78999999995
C754433	21	5512531.49
C813895	20	199999.79999999996
C838985	20	5830095.459999999
C776069	19	189999.80999999997
C495722	16	5022789.11
C406984	15	149999.85
C198120	15	4242329.47

# 11. Chain Laundering (A → B → C in Same Step).
Chains where money goes A → B → C in the same step.
Joins transactions table twice: one for A → B, another for B → C, with same step.

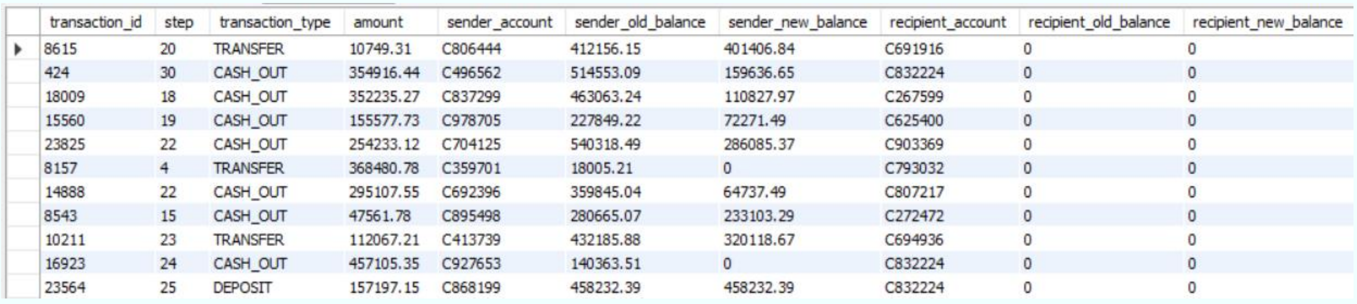
```sql
SELECT DISTINCT
    t1.sender_account AS first_sender,
    t1.recipient_account AS middle_account,
    t2.recipient_account AS final_account,
    t1.step
FROM transactions t1
JOIN transactions t2
  ON t1.recipient_account = t2.sender_account
 AND t1.step = t2.step
WHERE t1.sender_account <> t2.recipient_account;
```



first_sender	middle_account	final_account	step
C881408	C476782	C186866	28
C576356	C457925	M177895	10
C306759	C474707	C481304	16
C109106	C866134	C580981	1
C561056	C606046	M223417	15

# 12. Transactions Involving Mule Accounts.
All transactions that involve accounts already flagged as “mules” (from Query 4).
First identifies mule accounts with repeated zero balances, then joins back to find all their activity.

```sql
SELECT t.*
FROM transactions t
JOIN (
   SELECT recipient_account
   FROM transactions
   WHERE recipient_old_balance = 0 AND recipient_new_balance = 0
   GROUP BY recipient_account
   HAVING COUNT(*) > 5
) mules
ON t.sender_account = mules.recipient_account OR t.recipient_account =
mules.recipient_account;
```

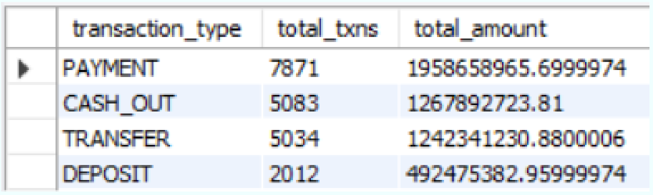


transaction_id	step	transaction_type	amount	sender_account	sender_old_balance	sender_new_balance	recipient_account	recipient_old_balance	recipient_new_balance
8615	20	TRANSFER	10749.31	C806444	412156.15	401406.84	C691916	0	0
424	30	CASH_OUT	354916.44	C496562	514553.09	159636.65	C832224	0	0
18009	18	CASH_OUT	352235.27	C837299	463063.24	110827.97	C267599	0	0
15560	19	CASH_OUT	155577.73	C978705	227849.22	72271.49	C625400	0	0
23825	22	CASH_OUT	254233.12	C704125	540318.49	286085.37	C903369	0	0
8157	4	TRANSFER	368480.78	C359701	18005.21	0	C793032	0	0
14888	22	CASH_OUT	295107.55	C692396	359845.04	64737.49	C807217	0	0
8543	15	CASH_OUT	47561.78	C895498	280665.07	233103.29	C272472	0	0
10211	23	TRANSFER	112067.21	C413739	432185.88	320118.67	C694936	0	0
16923	24	CASH_OUT	457105.35	C927653	140363.51	0	C832224	0	0
23564	25	DEPOSIT	157197.15	C868199	458232.39	458232.39	C832224	0	0

# 13. Transaction Volume by Type

Count and total amount for each type (PAYMENT, TRANSFER, CASH_OUT, DEPOSIT).
Groups by transaction_type and aggregates with COUNT and SUM.

```sql
SELECT transaction_type, COUNT(*) AS total_txns, SUM(amount) AS total_amount
FROM transactions
GROUP BY transaction_type
ORDER BY total_amount DESC;
```

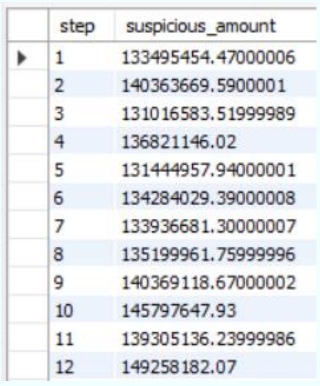


	transaction_type	total_txns	total_amount
▶	PAYMENT	7871	1958658965.6999974
	CASH_OUT	5083	1267892723.81
	TRANSFER	5034	1242341230.8800006
	DEPOSIT	2012	492475382.95999974

# 14. Daily Suspicious Exposure (By Step)

Total suspicious amounts per day, using rules: large transfers or mule accounts.
Groups by step, sums amounts with suspicious filters.

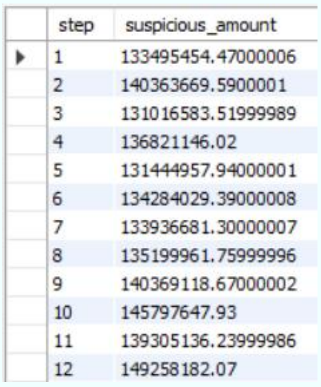
```sql
SELECT step, SUM(amount) AS suspicious_amount
FROM transactions
WHERE amount > 200000
   OR (recipient_old_balance = 0 AND recipient_new_balance = 0)
GROUP BY step
ORDER BY step;
```



	step	suspicious_amount
▶	1	133495454.47000006
	2	140363669.5900001
	3	131016583.51999989
	4	136821146.02
	5	131444957.94000001
	6	134284029.39000008
	7	133936681.30000007
	8	135199961.75999996
	9	140369118.67000002
	10	145797647.93
	11	139305136.23999986
	12	149258182.07

# 15. Accounts That Both Send and Receive.
Accounts playing dual roles (sender + recipient), typical of laundering intermediaries.
Combines sender and recipient roles into one table with UNION ALL, then aggregates by account.

```
SELECT account_id, SUM(sent_count) AS total_sent, SUM(received_count) AS total_received
FROM (
   SELECT sender_account AS account_id, COUNT(*) AS sent_count, 0 AS received_count
   FROM transactions
   GROUP BY sender_account
   UNION ALL
   SELECT recipient_account, 0, COUNT(*)
   FROM transactions
   GROUP BY recipient_account
) combined
GROUP BY account_id
ORDER BY total_sent DESC, total_received DESC
LIMIT 20;
```



step	suspicious_amount
1	133495454.47000006
2	140363669.5900001
3	131016583.51999989
4	136821146.02
5	131444957.94000001
6	134284029.39000008
7	133936681.30000007
8	135199961.75999996
9	140369118.67000002
10	145797647.93
11	139305136.23999986
12	149258182.07