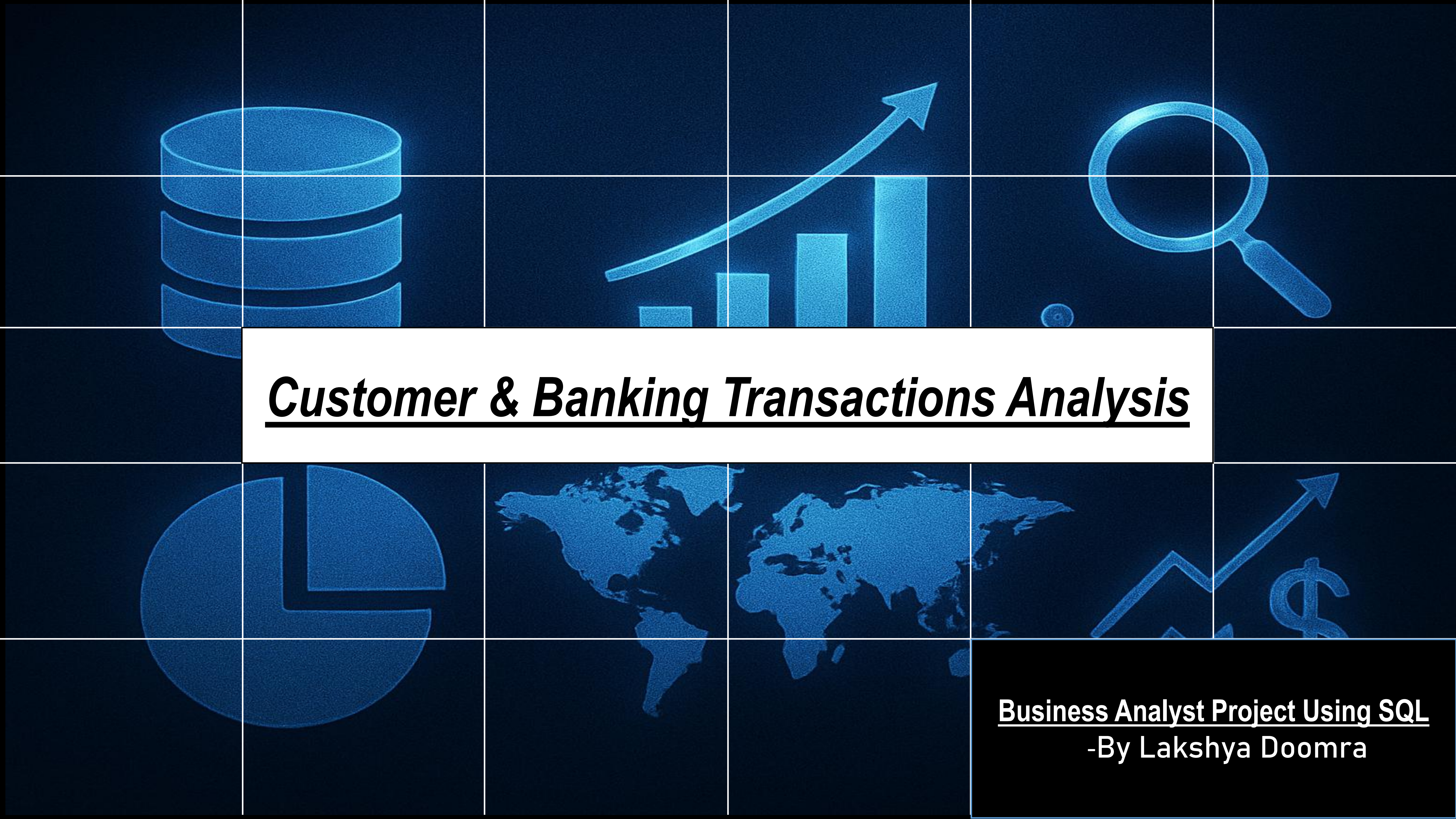
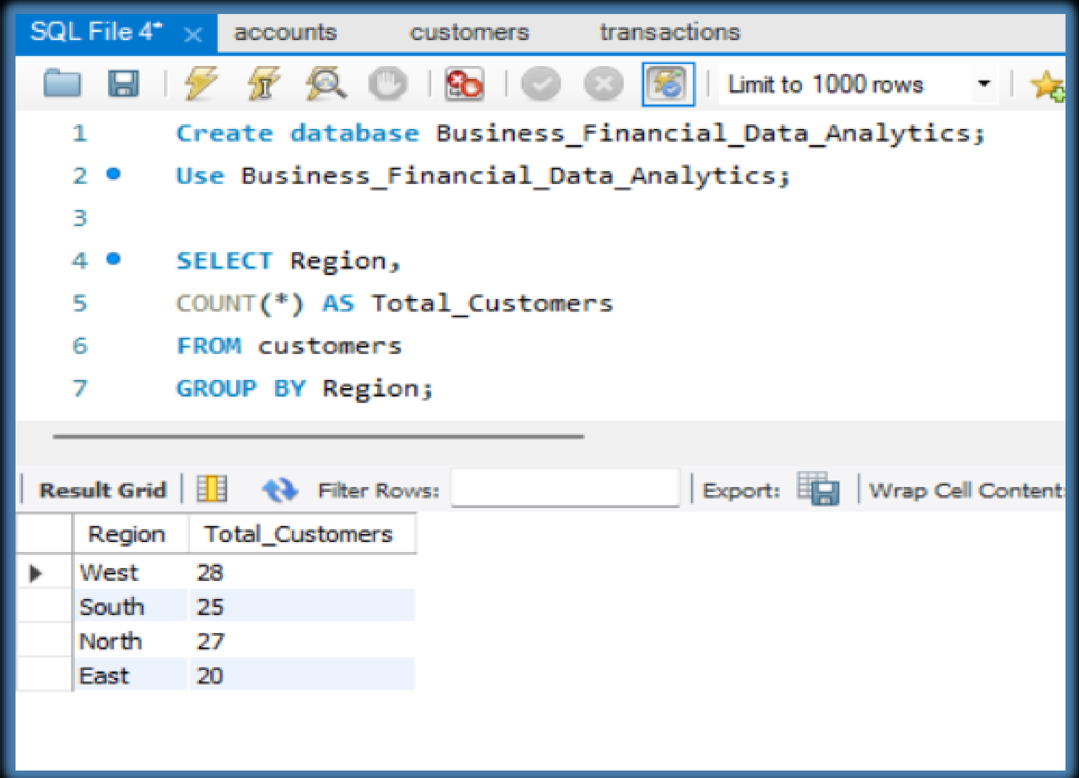


Customer & Banking Transactions Analysis

Business Analyst Project Using SQL

-By Lakshya Doomra

1. Customers per region



```
Create database Business_Financial_Data_Analytics;
Use Business_Financial_Data_Analytics;

SELECT Region,
COUNT(*) AS Total_Customers
FROM customers
GROUP BY Region;
```

Region	Total_Customers
West	28
South	25
North	27
East	20

SQL QUERY (Input)

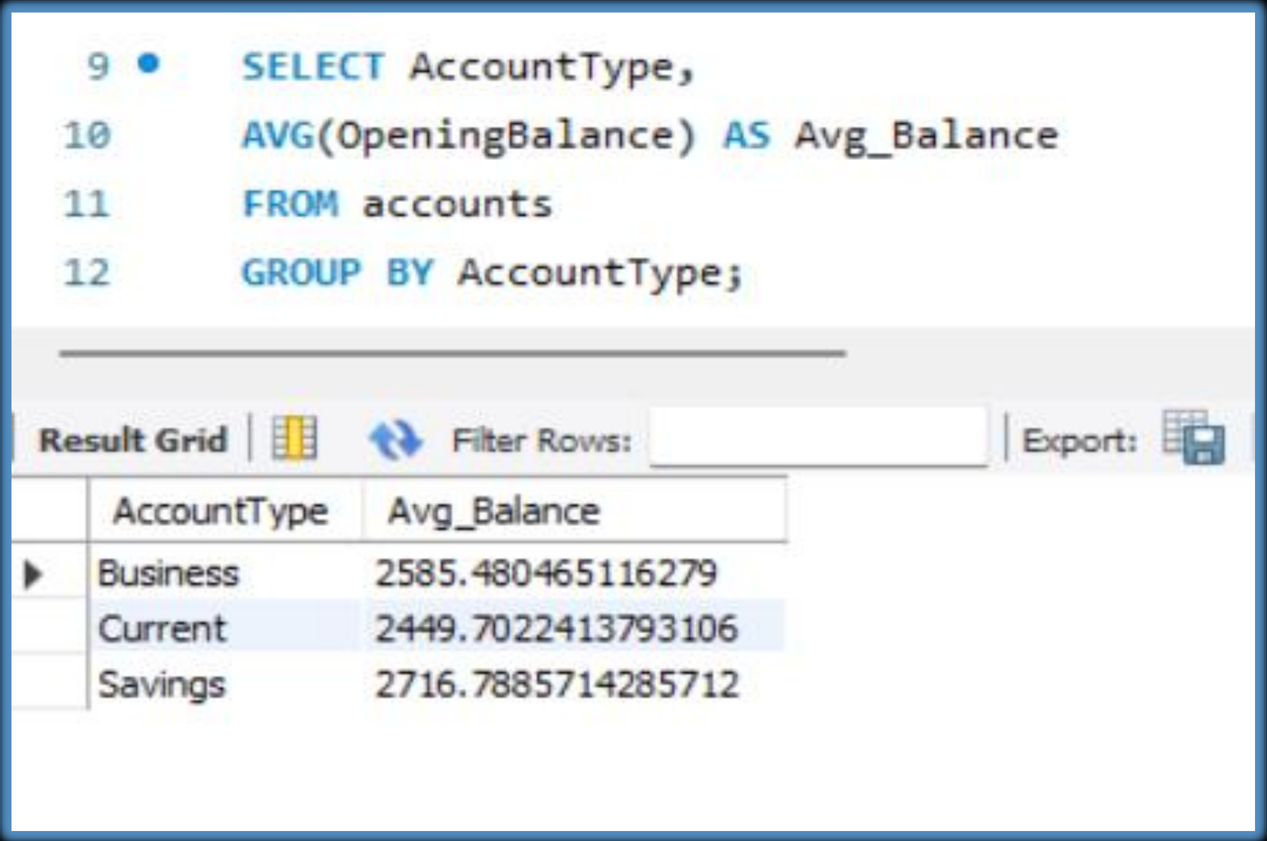
OUTPUT

Insights:

1. The **East region** should be targeted with **marketing campaigns, local offers, or new branch/digital service rollouts** to close the gap.

2. Since customer spread is fairly balanced, the bank is **not overly dependent on one region** — reducing regional risk.

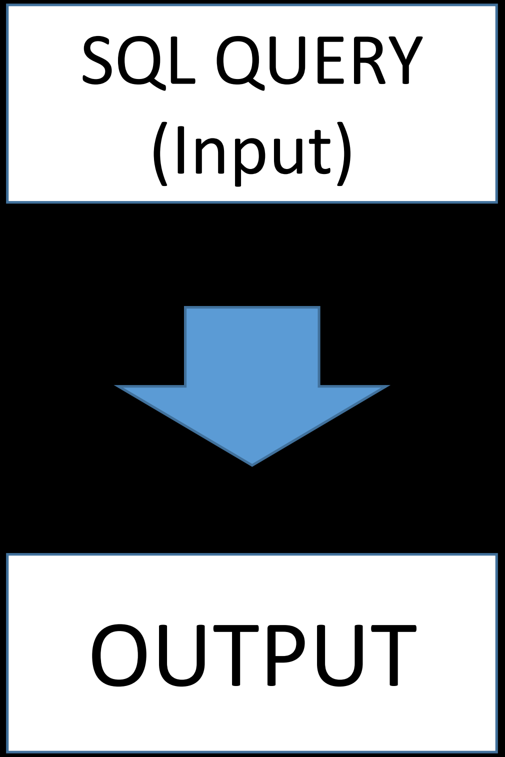
2. Average account balance by type



```
SELECT AccountType,
AVG(OpeningBalance) AS Avg_Balance
FROM accounts
GROUP BY AccountType;
```

Result Grid | Filter Rows: | Export:

AccountType	Avg_Balance
Business	2585.480465116279
Current	2449.7022413793106
Savings	2716.7885714285712

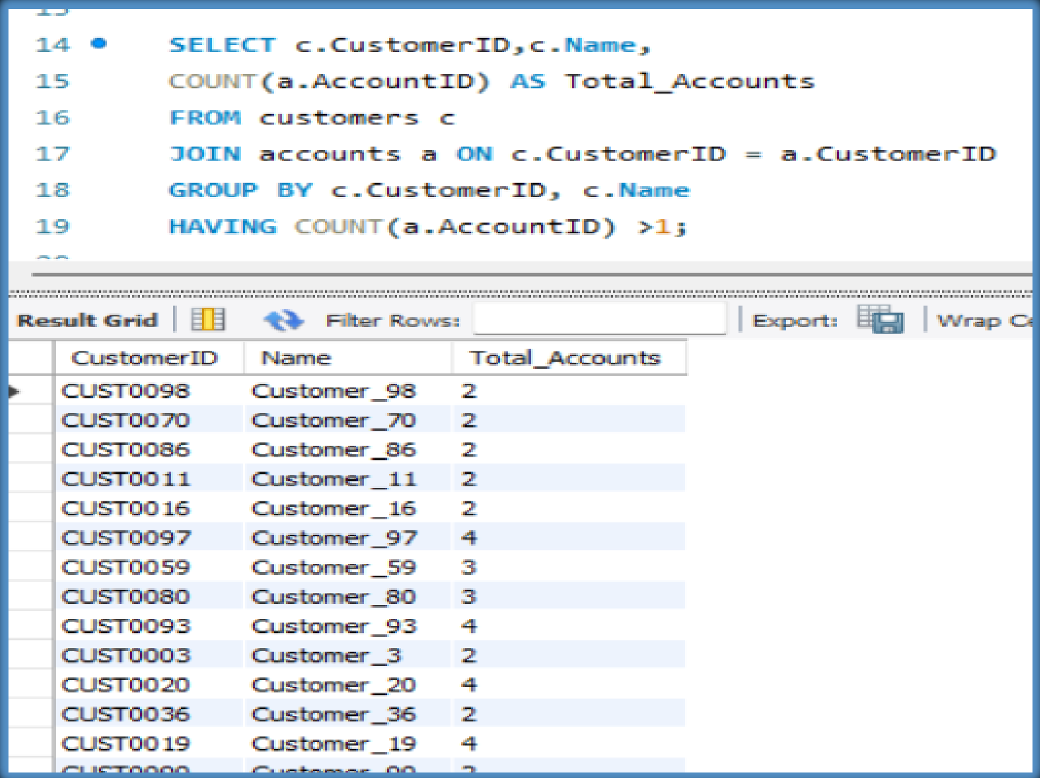


Insights:

-> Here, Savings holders maintain slightly higher balances than business/current accounts.

-> Again, not much difference, Current account customers may need **working capital or overdraft products** since balances are little lower.

3. Multi-account customers



```sql
SELECT c.CustomerID,c.Name,
COUNT(a.AccountID) AS Total_Accounts
FROM customers c
JOIN accounts a ON c.CustomerID = a.CustomerID
GROUP BY c.CustomerID, c.Name
HAVING COUNT(a.AccountID) >1;
```

CustomerID	Name	Total_Accounts
CUST0098	Customer_98	2
CUST0070	Customer_70	2
CUST0086	Customer_86	2
CUST0011	Customer_11	2
CUST0016	Customer_16	2
CUST0097	Customer_97	4
CUST0059	Customer_59	3
CUST0080	Customer_80	3
CUST0093	Customer_93	4
CUST0003	Customer_3	2
CUST0020	Customer_20	4
CUST0036	Customer_36	2
CUST0019	Customer_19	4

SQL QUERY (Input)

OUTPUT

📈 Insights:

Here. 40+ customers have multiple accounts (up to **6 accounts each**).

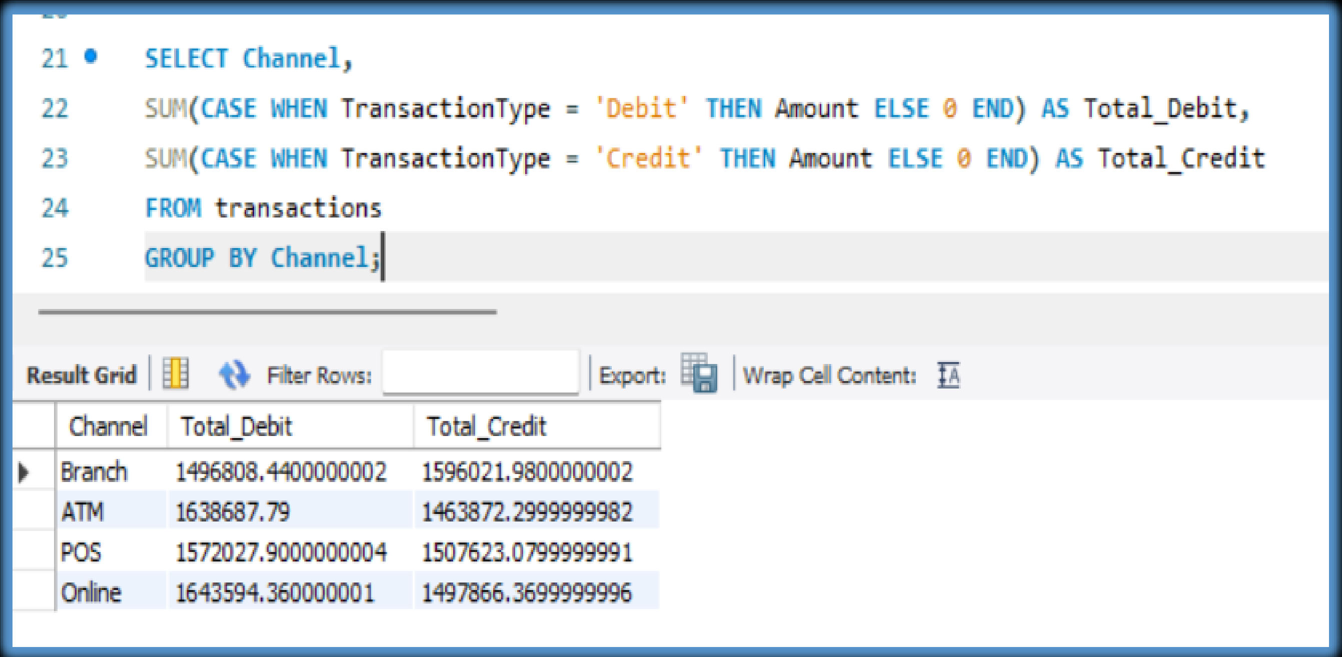
Many customers trust the bank enough to open multiple accounts.

Customers with 4–6 accounts **(e.g., CUST0033, CUST0048)** are strong candidates for priority banking.

Offer **loyalty programs** for customers with ≥3 accounts.

Monitor **high-activity customers** for cross-selling loans or investment products.

4. Debit vs. Credit volume by channel



```
SELECT Channel,
SUM(CASE WHEN TransactionType = 'Debit' THEN Amount ELSE 0 END) AS Total_Debit,
SUM(CASE WHEN TransactionType = 'Credit' THEN Amount ELSE 0 END) AS Total_Credit
FROM transactions
GROUP BY Channel;
```

```
Channel  Total_Debit          Total_Credit
Branch   1496808.4400000002   1596021.9800000002
ATM      1638687.79           1463872.2999999982
POS      1572027.9000000004   1507623.0799999991
Online   1643594.360000001    1497866.3699999996
```

SQL QUERY (Input)

OUTPUT

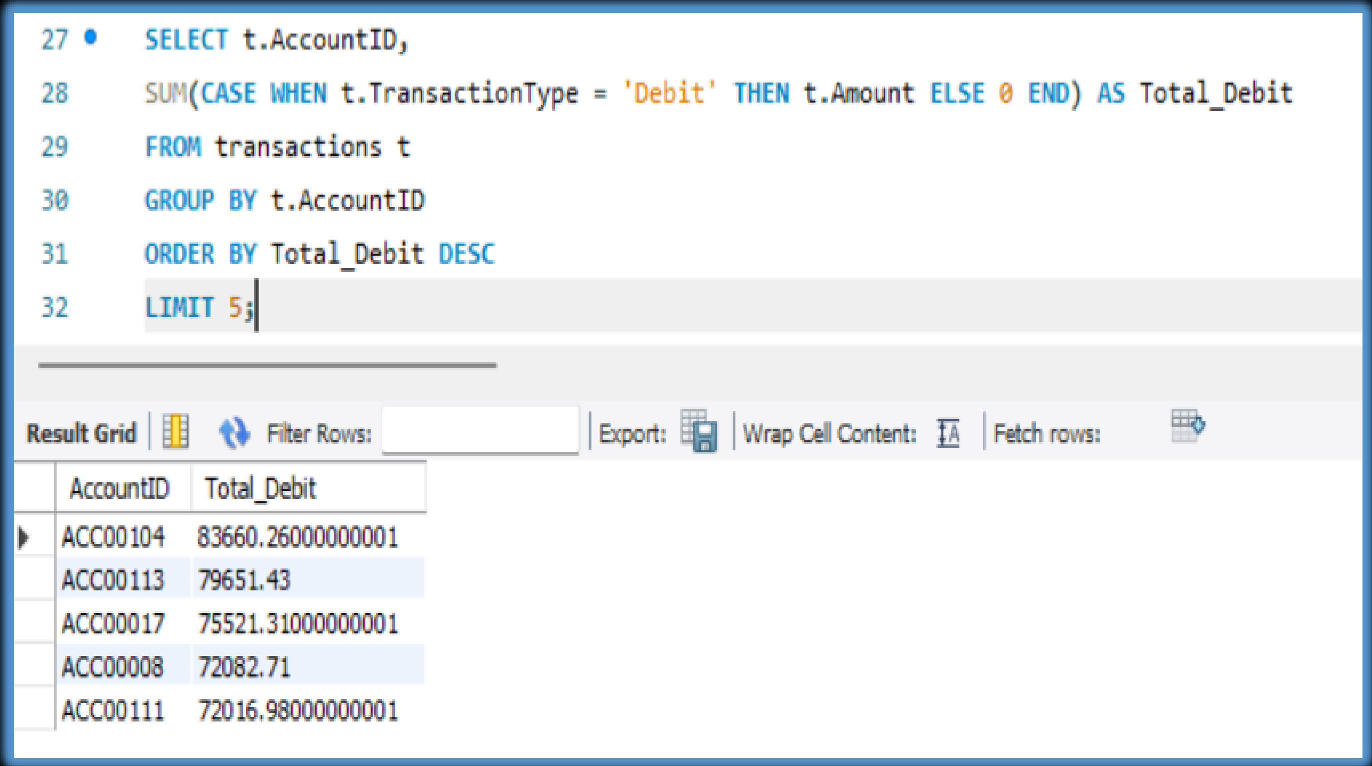
Insights:

Online & ATM Dominate (1.6M each) → Digital transactions are **more active than branch/POS**.

Traditional banking remains relevant and usage is still strong (1.5–1.6M)

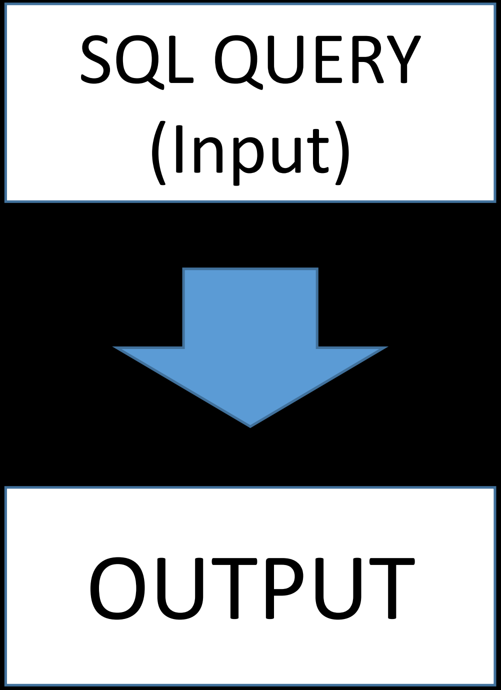
We can Push **digital banking campaigns** to reduce branch load.

5. Top debit accounts



```
SELECT t.AccountID,
SUM(CASE WHEN t.TransactionType = 'Debit' THEN t.Amount ELSE 0 END) AS Total_Debit
FROM transactions t
GROUP BY t.AccountID
ORDER BY Total_Debit DESC
LIMIT 5;
```

AccountID	Total_Debit
ACC00104	83660.26000000001
ACC00113	79651.43
ACC00017	75521.31000000001
ACC00008	72082.71
ACC00111	72016.98000000001



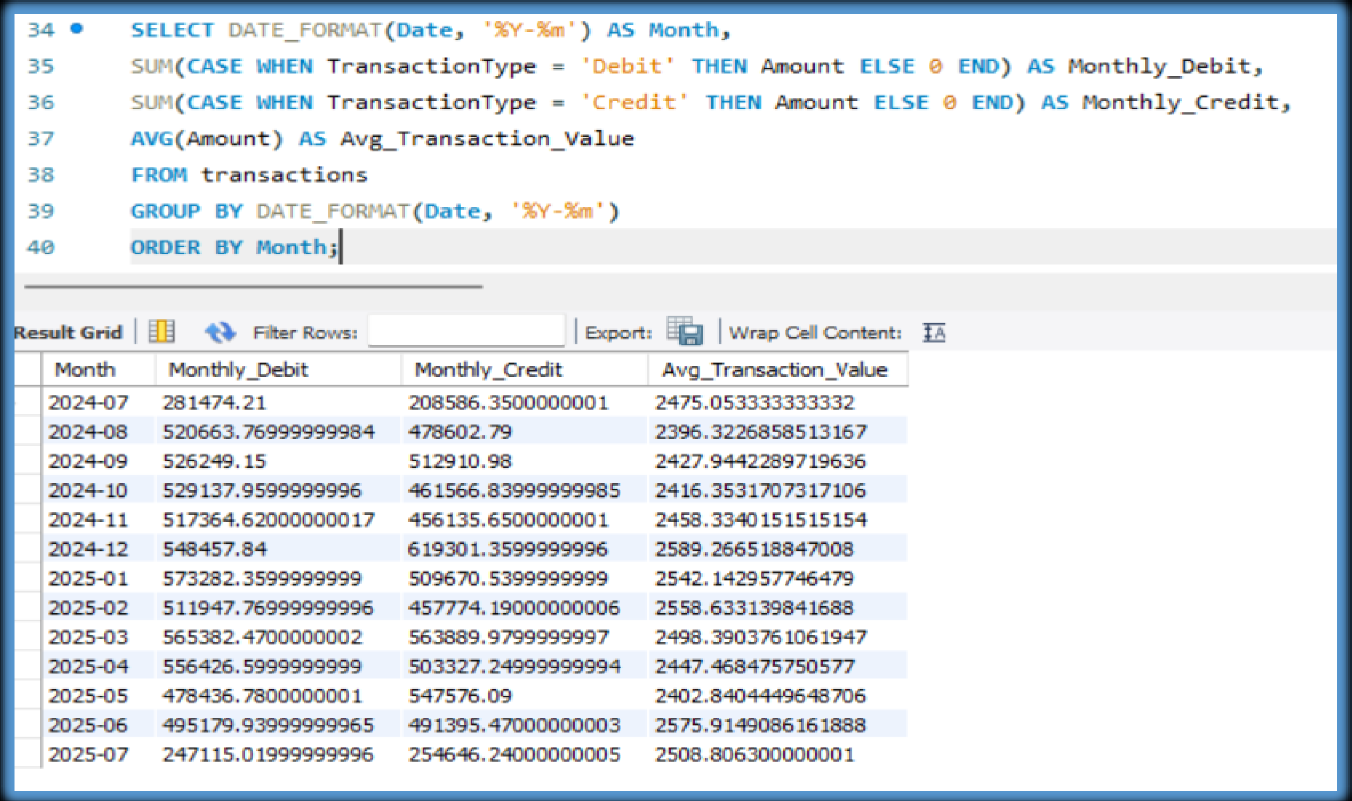
Insights:

Top 5 accounts contribute very high debit volume (₹380K total).

These accounts are likely business or high-income clients.

We can Assign **relationship managers** to these accounts

6. Monthly transaction trends



```sql
SELECT DATE_FORMAT(Date, '%Y-%m') AS Month,
SUM(CASE WHEN TransactionType = 'Debit' THEN Amount ELSE 0 END) AS Monthly_Debit,
SUM(CASE WHEN TransactionType = 'Credit' THEN Amount ELSE 0 END) AS Monthly_Credit,
AVG(Amount) AS Avg_Transaction_Value
FROM transactions
GROUP BY DATE_FORMAT(Date, '%Y-%m')
ORDER BY Month;
```

Month	Monthly_Debit	Monthly_Credit	Avg_Transaction_Value
2024-07	281474.21	208586.3500000001	2475.053333333332
2024-08	520663.76999999984	478602.79	2396.3226858513167
2024-09	526249.15	512910.98	2427.9442289719636
2024-10	529137.9599999996	461566.83999999985	2416.3531707317106
2024-11	517364.62000000017	456135.6500000001	2458.3340151515154
2024-12	548457.84	619301.3599999996	2589.266518847008
2025-01	573282.3599999999	509670.5399999999	2542.142957746479
2025-02	511947.76999999996	457774.19000000006	2558.633139841688
2025-03	565382.4700000002	563889.9799999997	2498.3903761061947
2025-04	556426.5999999999	503327.24999999994	2447.468475750577
2025-05	478436.7800000001	547576.09	2402.8404449648706
2025-06	495179.93999999965	491395.47000000003	2575.9149086161888
2025-07	247115.01999999996	254646.24000000005	2508.8063000000001

SQL QUERY
(Input)

OUTPUT

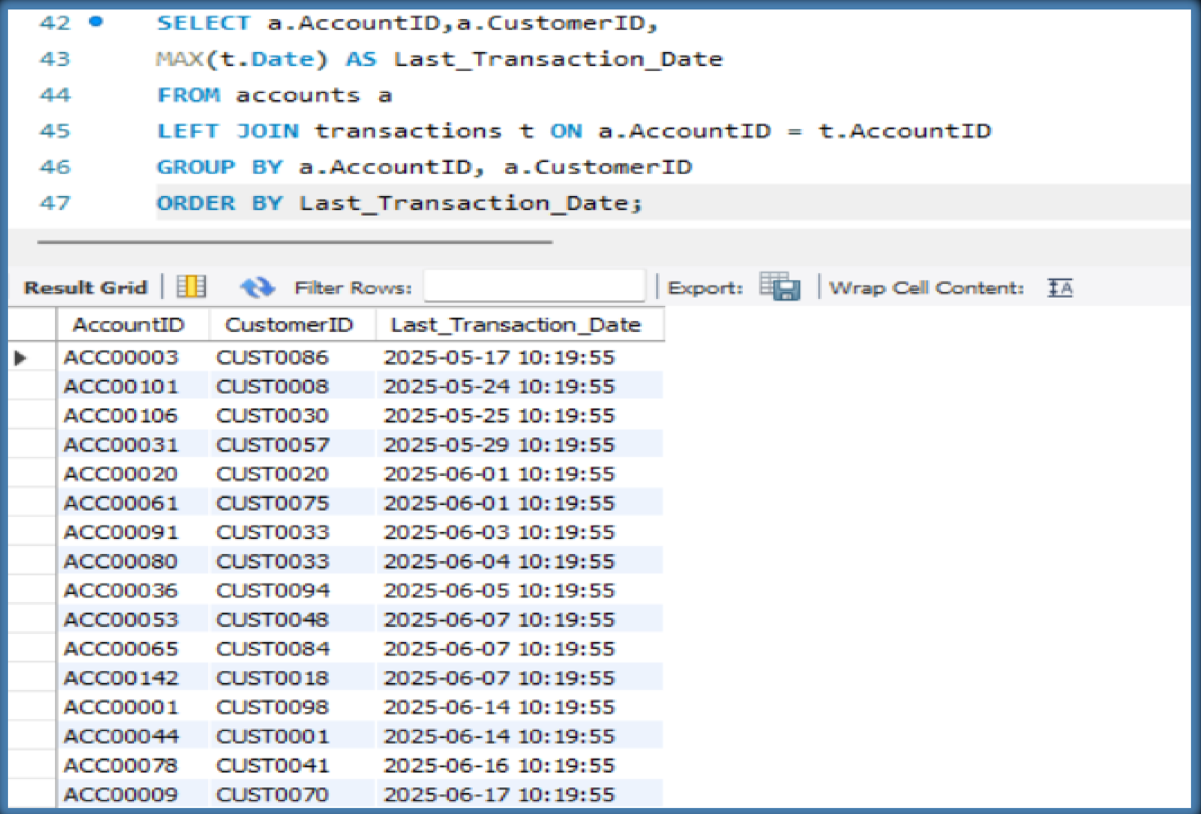
Insights:

Seasonal Peaks (Dec–Jan) → Likely due to festive season spending & year-end bonuses.

Plan **campaigns during festive months (Nov–Jan)** → maximize revenue

Investigate **July 2025 slowdown** → push targeted offers to boost activity.

7. Last Transaction Date per Account



```sql
SELECT a.AccountID,a.CustomerID,
MAX(t.Date) AS Last_Transaction_Date
FROM accounts a
LEFT JOIN transactions t ON a.AccountID = t.AccountID
GROUP BY a.AccountID, a.CustomerID
ORDER BY Last_Transaction_Date;
```

AccountID	CustomerID	Last_Transaction_Date
ACC00003	CUST0086	2025-05-17 10:19:55
ACC00101	CUST0008	2025-05-24 10:19:55
ACC00106	CUST0030	2025-05-25 10:19:55
ACC00031	CUST0057	2025-05-29 10:19:55
ACC00020	CUST0020	2025-06-01 10:19:55
ACC00061	CUST0075	2025-06-01 10:19:55
ACC00091	CUST0033	2025-06-03 10:19:55
ACC00080	CUST0033	2025-06-04 10:19:55
ACC00036	CUST0094	2025-06-05 10:19:55
ACC00053	CUST0048	2025-06-07 10:19:55
ACC00065	CUST0084	2025-06-07 10:19:55
ACC00142	CUST0018	2025-06-07 10:19:55
ACC00001	CUST0098	2025-06-14 10:19:55
ACC00044	CUST0001	2025-06-14 10:19:55
ACC00078	CUST0041	2025-06-16 10:19:55
ACC00009	CUST0070	2025-06-17 10:19:55

SQL QUERY (Input)

OUTPUT

Insights:

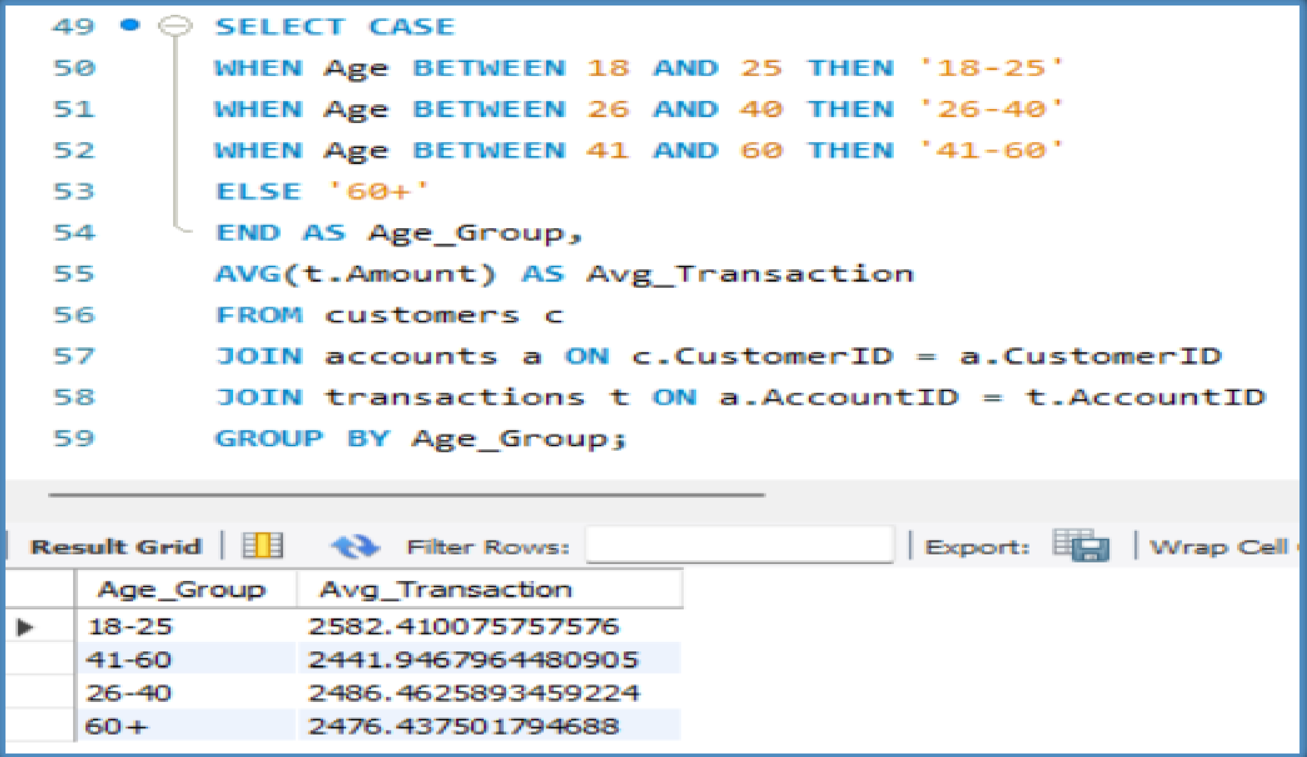
High Activity in June–July 2025 : Customers are active; recent transactions confirm strong engagement.

Multi-account customers (CUST0099, CUST0048) show **regular activity across accounts**, suggesting high-value profiles.

Flag accounts with **>60 days inactivity** for **re-engagement campaigns**

Prioritize **multi-account active users** for loyalty offers.

8. Average transaction amount by age group



```
SELECT CASE
WHEN Age BETWEEN 18 AND 25 THEN '18-25'
WHEN Age BETWEEN 26 AND 40 THEN '26-40'
WHEN Age BETWEEN 41 AND 60 THEN '41-60'
ELSE '60+'
END AS Age_Group,
AVG(t.Amount) AS Avg_Transaction
FROM customers c
JOIN accounts a ON c.CustomerID = a.CustomerID
JOIN transactions t ON a.AccountID = t.AccountID
GROUP BY Age_Group;
```

```
Age_Group   Avg_Transaction
18-25       2582.410075757576
41-60       2441.9467964480905
26-40       2486.4625893459224
60+         2476.437501794688
```

SQL QUERY (Input)

OUTPUT

Insights:

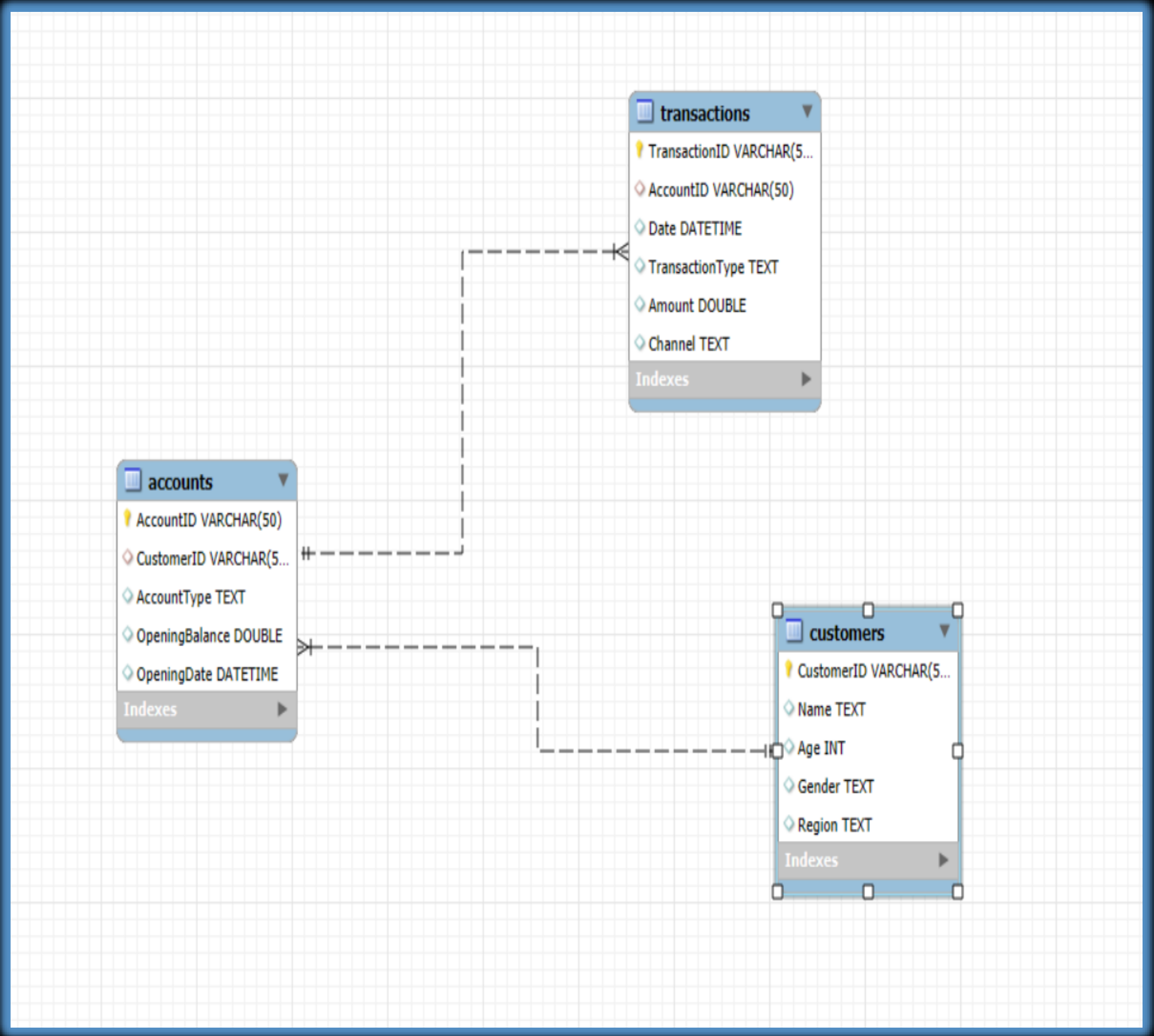
Youngest group (18–25) spends the most (₹2582): Indicates **higher adoption of digital/payment channels**.

Target **18–25 group** with **youth-oriented financial products (UPI, credit cards, micro-investments)**.

41–60 group spends least (~₹2442): Possibly more cautious or diversified spending habits.

Retain **41–60 group** with **wealth products (mutual funds, insurance, retirement)**.

ER DIAGRAM:



This ER diagram represents the relational structure of the database, highlighting three tables – customers, accounts and transactions. It shows how customers can have multiple accounts, and each account can record multiple transactions. This design ensures data integrity and allows efficient analysis of customer behavior, account balances, and transaction patterns