# Task 3

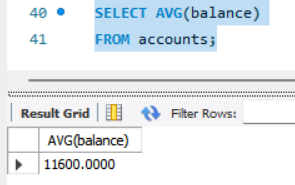
1. Write a SQL query to Find the average account balance for all customers.

Query:

SELECT AVG(balance)

FROM accounts;

Output:



1. Write a SQL query to Retrieve the top 10 highest account balances.

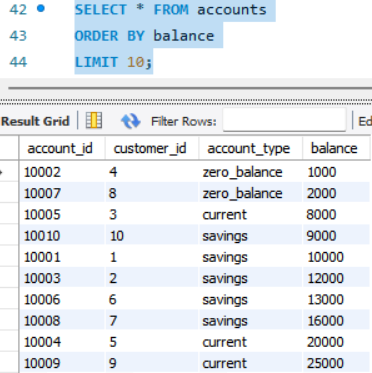
Query:

SELECT \* FROM accounts

ORDER BY balance

LIMIT 10;

Output:



1. Write a SQL query to Calculate Total Deposits for All Customers in specific date.

Query:

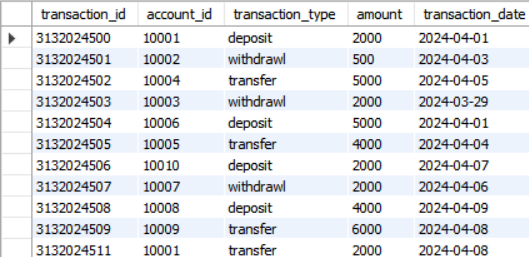
SELECT SUM(amount) AS Total Deposits

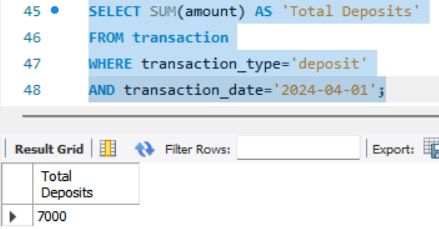
FROM transaction

WHERE transaction\_type=’deposit’

AND transaction\_date=’2024-04-08’;

Output:





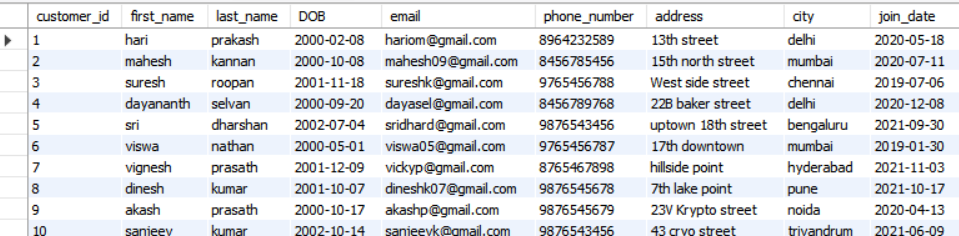
1. Write a SQL query to Find the Oldest and Newest Customers.

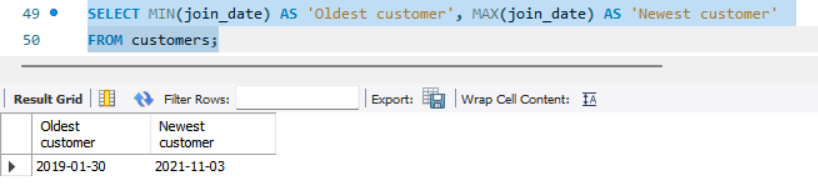
Query:

SELECT MIN(join\_date) AS Oldest customer, MAX(join\_date) AS Newest customer

FROM customers;

Output:





1. Write a SQL query to Retrieve transaction details along with the account type.

Query:

SELECT T.transaction\_id, T.transaction\_type,T.amount,T.transaction\_date,

A.account\_type

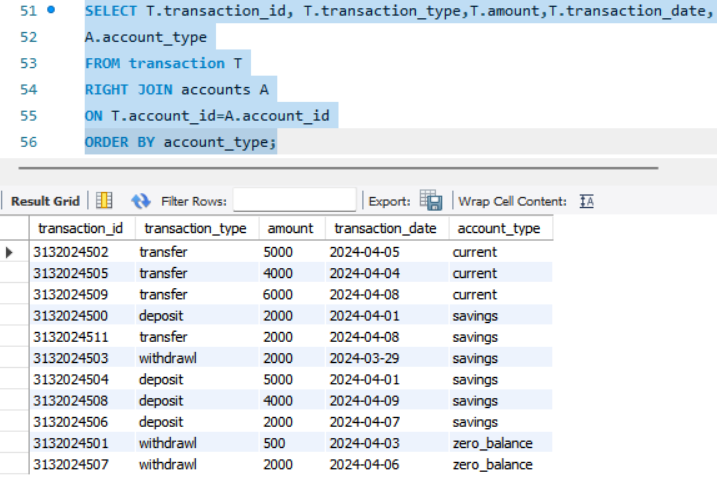
FROM transaction T

RIGHT JOIN accounts A

ON T.account\_id=A.account\_id

ORDER BY account\_type;

Output:



1. Write a SQL query to Get a list of customers along with their account details.

Query:

SELECT C.customer\_id, CONCAT(first\_name,last\_name) AS name, A.account\_id,A.account\_type,A.balance

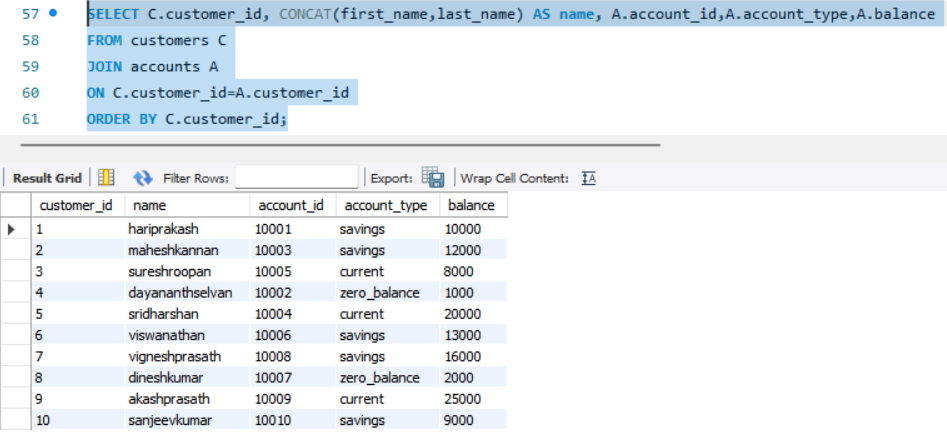
FROM customers C

JOIN accounts A

ON C.customer\_id=A.customer\_id

ORDER BY C.customer\_id;

Output:



1. Write a SQL query to Retrieve transaction details along with customer information for a specific account.

Query:

SELECT C.customer\_id,CONCAT(C.first\_name,C.last\_name)

AS name,T.transaction\_id,T.account\_id,T.transaction\_type,

T.amount,T.transaction\_date

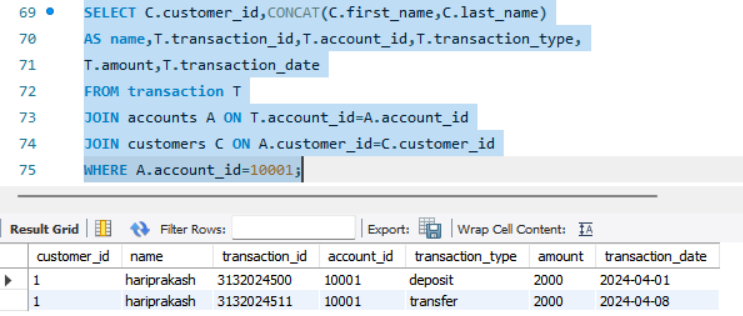
FROM transaction T

JOIN accounts A ON T.account\_id=A.account\_id

JOIN customers C ON A.customer\_id=C.customer\_id

WHERE A.account\_id=10001;

Output:



1. Write a SQL query to Identify customers who have more than one account.

Query:

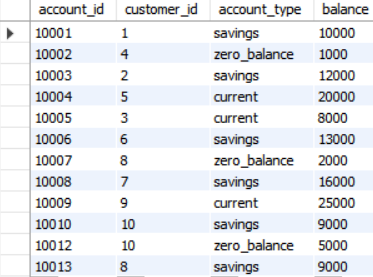
SELECT customer\_id,COUNT(\*) AS 'Count'

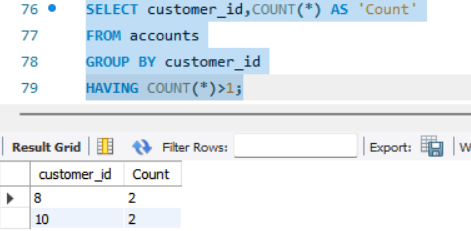
FROM accounts

GROUP BY customer\_id

HAVING COUNT(\*)>1;

Output:





1. Write a SQL query to Calculate the difference in transaction amounts between deposits and withdrawals.

Query:

SELECT

SUM(CASE

WHEN transaction\_type='deposit' THEN amount ELSE 0 END) -

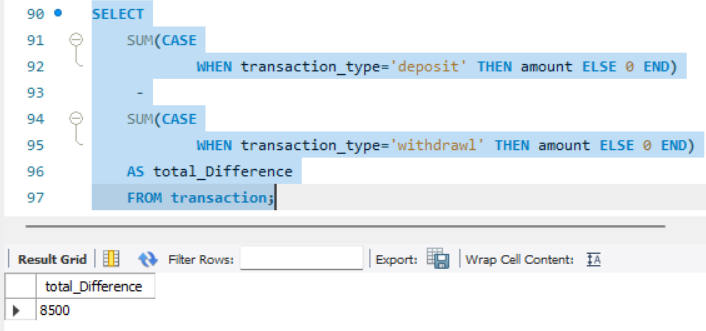
SUM(CASE

WHEN transaction\_type='withdrawl' THEN amount ELSE 0 END)

AS total\_Difference

FROM transaction;

Output:



1. Write a SQL query to Calculate the average daily balance for each account over a specified period.

Query:

SELECT account\_id,

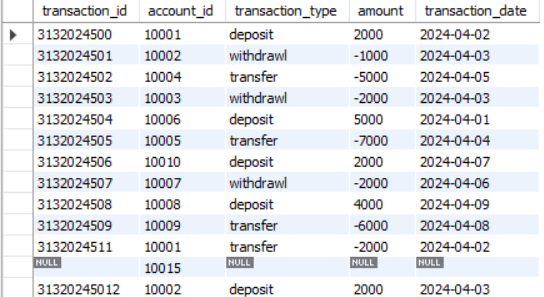
AVG(amount) AS avg\_daily\_balance

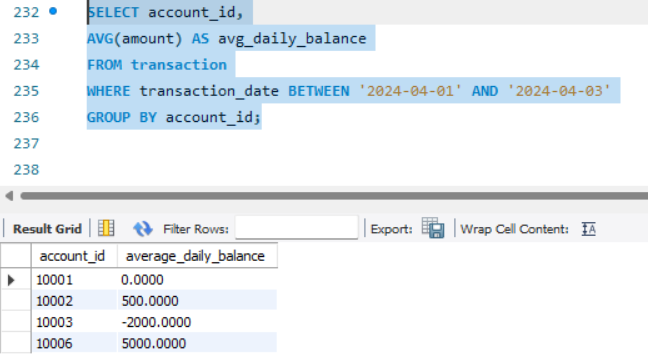
FROM transaction

WHERE transaction\_date BETWEEN '2024-04-01' AND '2024-04-03'

GROUP BY account\_id;

Output:





1. Calculate the total balance for each account type.

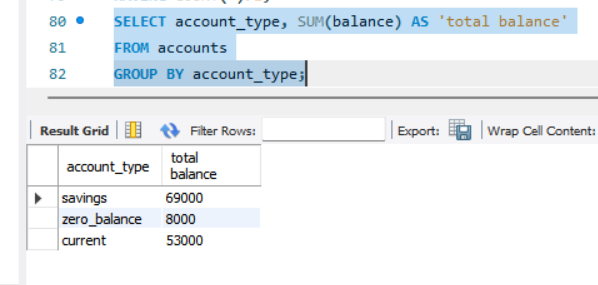
Query:

SELECT account\_type, SUM(balance) AS ‘total balance’

FROM accounts

GROUP BY account\_type;

Output:



1. Identify accounts with the highest number of transactions order by descending order.

Query:

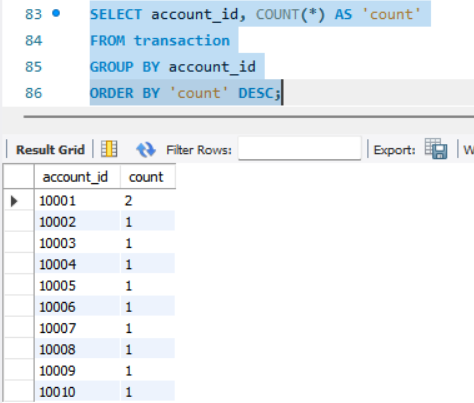
SELECT account\_id, COUNT(\*) AS ‘count’

FROM transaction

GROUP BY account\_id

ORDER BY ‘count’ DESC;

Output:



1. List customers with high aggregate account balances, along with their account types.

Query:

With TotalBalance AS(

SELECT customer\_id, SUM(balance) AS tot\_bal

FROM accounts

GROUP BY customer\_id)

SELECT C.customer\_id, CONCAT(C.first\_name,C.last\_name)AS name,

A.account\_type,T.tot\_bal

FROM customers C

JOIN TotalBalance T

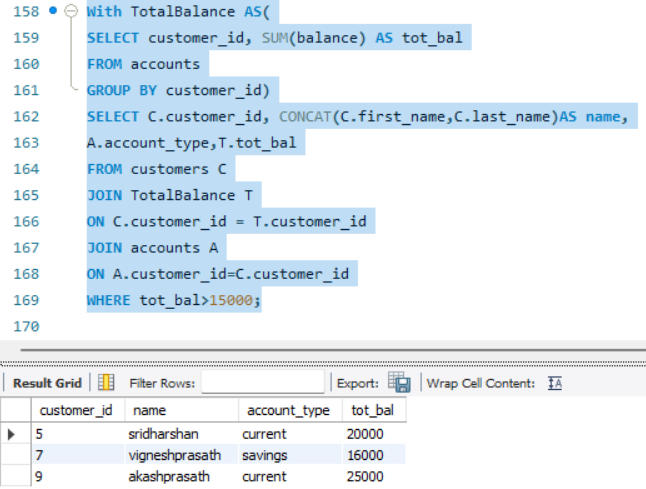
ON C.customer\_id = T.customer\_id

JOIN accounts A

ON A.customer\_id=C.customer\_id

WHERE tot\_bal>15000;

Output:



1. Identify and list duplicate transactions based on transaction amount, date, and account.

Query:

SELECT account\_id, amount, transaction\_date ,COUNT(\*) AS dupcount

FROM transaction

GROUP BY amount, transaction\_date, account\_id HAVING COUNT(\*) > 1;

Output:

