Tasks 3: Aggregate functions, Having, Order By, GroupBy and Joins:

1. Write a SQL query to Find the average account balance for all customers.
   select avg(balance) as average_balace from accounts;

```
+----------------+
| average_balace |
+----------------+
|    6570.190000 |
+----------------+
```

2. Write a SQL query to Retrieve the top 10 highest account balances.
   select balance
   -> from accounts
   -> order by balance desc
   -> limit 10;

```
+----------+
| balance  |
+----------+
| 18000.25 |
| 15000.00 |
| 12000.50 |
|  7200.40 |
|  6000.00 |
|  5000.00 |
|  2500.75 |
|     0.00 |
|     0.00 |
|     0.00 |
+----------+
```

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

select sum(amount) from transactions

-> where transaction_type='deposit'

-> and transaction_date=2025-06-05;

```
+-------------+
```

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

SELECT * FROM customers

-> ORDER BY DOB

-> LIMIT 1;

```
+-------------+------------+-----------+------------+----------------------+--------------+-------------+
| customer_id | first_name | last_name | DOB        | email                | phone_number | address     |
+-------------+------------+-----------+------------+----------------------+--------------+-------------+
```

| 9 | Mohan | Vel | 1994-12-23 | mohan.vel@example.com | 9876543218 | Thoothukudi |

+-------------+------------+-----------+------------+-----------------------+--------------+-------------+

mysql> SELECT * FROM customers

-> ORDER BY DOB DESC

-> LIMIT 1; -- Newest customer

+-------------+------------+-----------+------------+-----------------------+--------------+-------------+

| customer_id | first_name | last_name | DOB | email | phone_number | address |

+-------------+------------+-----------+------------+-----------------------+--------------+-------------+

| 10 | Divya | Shree | 2002-06-14 | divya.shree@example.com | 9876543219 | Kanyakumari |

+-------------+------------+-----------+------------+-----------------------+--------------+-------------+

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

SELECT t.*

-> FROM transactions t

-> JOIN accounts a ON a.account_id = t.account_id;

+----------------+------------+------------------+---------+------------------+

| transaction_id | account_id | transaction_type | amount | transaction_date |

+----------------+------------+------------------+---------+------------------+

| 1 | 2 | deposit | 2500.00 | 2025-03-01 |

| 2 | 5 | withdrawal | 700.00 | 2025-03-02 |

| 3 | 3 | deposit | 3200.75 | 2025-03-03 |

| 4 | 7 | transfer | 1800.50 | 2025-03-04 |

| 5 | 1 | withdrawal | 900.00 | 2025-03-05 |

| 6 | 4 | deposit | 5000.00 | 2025-03-06 |

| 7 | 6 | transfer | 1100.25 | 2025-03-07 |

| 8 | 9 | withdrawal | 400.00 | 2025-03-08 |

| 9 | 8 | deposit | 2000.40 | 2025-03-09 |

| 10 | 10 | transfer | 2300.00 | 2025-03-10 |

+----------------+------------+------------------+---------+------------------+

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

```
use hmbank;
select concat(c.first_name," ",c.last_name) as Name ,
a.account_id,a.account_type from accounts a
join customers c on c.customer_id=a.customer_id
```

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

```
select  t.*,c.* from transactions t join accounts a on t.account_id=a.account_id
join customers c on  c.customer_id=a.account_id
where a.account_id=4;
```

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

```
SELECT
   CONCAT(c.first_name, ' ', c.last_name) AS full_name,
   COUNT(a.account_id) AS totacc
FROM customers c
JOIN accounts a ON a.customer_id = c.customer_id
GROUP BY c.customer_id
HAVING COUNT(a.account_id) > 1
;
```

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

```
SELECT
   SUM(CASE WHEN transaction_type = 'Deposit' THEN amount ELSE 0 END) -
   SUM(CASE WHEN transaction_type = 'Withdrawal' THEN amount ELSE 0 END)
   AS transaction_difference
FROM transactions;
```

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

```
SELECT
   t.account_id,
   SUM(.amount) / COUNT(DISTINCT t.transaction_date) AS avg_daily_balance
FROM transactions t
WHERE t.transaction_date BETWEEN '2025-01-01' AND '2025-03-3'
GROUP BY t.account_id;
```

11. Calculate the total balance for each account type.

```
SELECT
   a.account_type,
   SUM(a.balance) AS total_balance
FROM accounts a
GROUP BY a.account_type;
```

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

```
SELECT
   t.account_id,
   COUNT(t.transaction_id) AS transaction_count
FROM transactions t
GROUP BY t.account_id
ORDER BY transaction_count DESC;
```

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

```
SELECT
   c.customer_id,
   CONCAT(c.first_name, ' ', c.last_name) AS customer_name,
   a.account_type,
   SUM(a.balance) AS total_balance
FROM customers c
JOIN accounts a ON c.customer_id = a.customer_id
GROUP BY c.customer_id, a.account_type
HAVING total_balance > 10000
ORDER BY total_balance DESC;
```

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

```
SELECT
   account_id,
```

```
    amount,
    transaction_date,
    COUNT(*) AS duplicate_count
FROM transactions
GROUP BY account_id, amount, transaction_date
HAVING COUNT(*) >1;
```