SELECT Queries

TOP 3 Queries based on Practical Utility

1. Find all customers who have a fixed deposit due for maturity within the next 30 days

```
SELECT FD_no, CONCAT(fname,' ',lname) AS name, maturity_date,
mobile_no

FROM Fixed_deposit

NATURAL JOIN Account

NATURAL JOIN Customer

WHERE DATE(maturity_date)>=DATE_TRUNC('month',DATE '2025-04-01')

AND DATE(maturity_date)<=DATE_TRUNC('month',DATE '2025-04-01') +
INTERVAL '1 month';
```

2. List all customers who have taken a loan but missed any installments before date ‘2024-06-15’

```
SELECT CONCAT(fname,' ',lname) AS name, loan_app_no,
loan_installment_no, due_date, due_amt

FROM Loan_repayment

NATURAL JOIN Loan_application

NATURAL JOIN Account

NATURAL JOIN Customer

WHERE DATE(due_date)<DATE '2024-06-15' AND settlement_date IS NULL;
```

3. List the transaction details of a particular account for 1 week

```
SELECT *

FROM Transaction

WHERE account_no = '1749938644158'

AND DATE(date) >= DATE_TRUNC('week', DATE '2024-04-05')

AND DATE(date) < DATE_TRUNC('week', DATE '2024-04-05') + INTERVAL '1
week';
```

<u>**All Other Queries**</u>

1. Check due dates and installment details for an account

```
SELECT ins_installment_no, due_date, due_amt

FROM Insurance_record

WHERE ins_app_no = 'INSAPP0009'

ORDER BY ins_installment_no;
```

2. Check if your service req is approved or not

```
SELECT status, additional_notes

FROM Service_Request

WHERE account_no = '9305612434851';
```

3. Transaction history for a particular account

```
SELECT * FROM Transaction

WHERE account_no = '4632857200008';
```

4. Transaction and all details of an account

```
SELECT T.transaction_ID, T.amount, T.transaction_type, T.date,
T.mode, T.account_no, T.receiver_acc_no, A.acc_Type,
A.available_balance, B.branch_name, B.location AS branch_location

FROM Transaction AS T

NATURAL JOIN Account AS A

NATURAL JOIN Branch AS B

NATURAL JOIN Customer AS C

WHERE T.account_no = '5790182026699'

ORDER BY T.date DESC;
```

5. Transaction between 2 dates

```
SELECT T.transaction_ID, T.amount, T.transaction_type, T.date,
T.mode, T.receiver_acc_no

FROM Transaction AS T

NATURAL JOIN Account AS A
```

```
WHERE A.account_no = '9305612434851' AND T.date BETWEEN '2024-04-10'
AND '2024-04-20';
```

6. Investment info for a customer

```
SELECT c.fname, c.lname, I.inv_app_no, I.annual_duration, I.inv_amt,
I.status, I.approved_date, I.profit

FROM Investment_application AS I

NATURAL JOIN Account AS A

NATURAL JOIN Customer AS C

WHERE A.account_no = '5790182026699';
```

7. FD info for a particular customer

```
SELECT c.fname, c.lname, FD.FD_no, FD.dep_amt, FD.maturity_date,
FD.maturity_amt, FD.opened_date

FROM Fixed_deposit AS FD

NATURAL JOIN Account AS A

NATURAL JOIN Customer AS C

WHERE A.account_no = '5212100743402';
```

8. Info for investment for an account number

```
SELECT * FROM Investment_application

WHERE account_no = '6518676373981';
```

9. Insurance info for a account number

```
SELECT * FROM Insurance_Application

WHERE account_no = '6518676373981';
```

10. Total transactions for a account

```
SELECT Transaction.account_no, COUNT(*) AS total_transactions

FROM Transaction

WHERE account_no = '6518676373981'

GROUP BY account_no;
```

11. Retrieve the total loan amount approved for each account holder

```
SELECT la.account_no, SUM(la.loan_amt) AS total_approved_loan_amount

FROM Loan_application as la

WHERE la.status = 'Approved'

GROUP BY la.account_no;
```

12. Total deposited money in each branch

```
SELECT a.Branch_code, SUM(a.Available_Balance) AS
total_deposit_amount

FROM Account a

GROUP BY a.Branch_code;
```

13. Customers with multiple accounts

```
SELECT a.UUID, COUNT(*) AS num_of_accounts

FROM Account a

GROUP BY a.UUID

HAVING COUNT(*) > 1;
```

14. Total loan amounts for approved accounts

```
SELECT c.fname, c.lname, SUM(r.loan_amt) AS total_loan_amount

FROM Customer AS c

NATURAL JOIN (Account AS a NATURAL JOIN Loan_application as l) as r

WHERE r.status = 'Approved'

GROUP BY c.fname, c.lname;
```

15. Service requests that are pending with info of a person

```
SELECT sr.req_ID, sr.req_date, sr.status, sr.additional_notes,
s.service_name, c.fname, c.lname

FROM Service_Request AS sr

NATURAL JOIN Service AS s

NATURAL JOIN Account AS a

NATURAL JOIN Customer AS c

WHERE sr.status = 'Pending';
```

16. The total balance for each customer who has approved loans

```
SELECT SUM(a.available_balance) AS total_amount_approved

FROM Customer c

NATURAL JOIN Account AS a

NATURAL JOIN Loan_application AS l

WHERE l.status = 'Approved';
```

17. The average transaction amount for each account type and month

```
SELECT a.acc_Type, EXTRACT(MONTH FROM t.date) AS transaction_month,
AVG(t.amount) AS avg_transaction_amount

FROM Account a

NATURAL JOIN Transaction AS t

GROUP BY a.acc_Type,EXTRACT(MONTH FROM t.date);
```

18. Update name of a customer:

```
UPDATE Customer

SET fname = 'Kathan', lname = 'Kadiya'

WHERE UUID = '469469101233';
```

19. Update email of a customer:

```
UPDATE Customer

SET email = 'newemail@example.com'

WHERE UUID = '729388133021';
```

20. Info for investment for a account number

```
SELECT *

FROM Investment_application

WHERE account_no = '6518676373981';
```

21. Insurance info for a account number

```
SELECT *

FROM Insurance_application

WHERE account_no = '6518676373981';
```

22. Total transactions for all accounts with atleast one transaction

```
SELECT account_no, COUNT(*) AS total_transactions

FROM Transaction

GROUP BY account_no;
```

23. Get all the information of a customer

```
SELECT CONCAT(fname,' ',lname) as full_name, account_no,
available_balance

FROM Customer NATURAL JOIN Account

WHERE UUID='769055090046';
```

24. Change the address of a particular customer

```
UPDATE Customer

SET location='DA-IICT', PIN='226005'

WHERE UUID='769055090046';
```

25. Retrieve and change the mobile_no of a customer

```
UPDATE Customer

SET mobile_no='9664758176'

WHERE UUID='769055090046';
```

26. Find all customers who have a pending loan application along with their account details

```
SELECT CONCAT(c.fname,' ', c.lname) AS full_name, r.account_no,
r.status

FROM Customer AS c

NATURAL JOIN (SELECT account_no, status FROM Account NATURAL JOIN
Loan_application

WHERE status = 'Pending') AS r;
```

27. Identify customers who have accounts with a balance higher than the average balance across all accounts in the bank

```
SELECT c.UUID, CONCAT(c.fname,' ',c.lname) AS full_name,
a.account_no, a.available_balance, r.avg_bal

FROM Customer AS c
```

```
NATURAL JOIN Account AS a

NATURAL JOIN (SELECT avg(available_balance) AS avg_bal FROM Account)
AS r

WHERE a.available_balance > avg_bal;
```

28. No of accounts of each Customer in descending order

```
SELECT c.UUID, CONCAT(c.fname,' ',c.lname) AS full_name,
COUNT(a.account_no) AS no_of_acc

FROM Customer AS c

NATURAL JOIN Account AS a

GROUP BY c.UUID

ORDER BY no_of_acc DESC;
```

29. Retrieve the top 5 branches with the highest total number of transactions overall

```
SELECT b.branch_code, b.branch_name, COUNT(t.transaction_ID) AS
no_of_transactions

FROM Branch AS b

NATURAL JOIN Account AS a

NATURAL JOIN Transaction AS t

GROUP BY b.branch_code

ORDER BY no_of_transactions DESC

LIMIT 5;
```

30. Identify the most common type of service requests made by customers

```
SELECT s.service_name, s.service_ID, COUNT(r.service_ID) AS
total_requests

FROM Service_request AS r

NATURAL JOIN Service AS s

GROUP BY s.service_ID

ORDER BY total_requests DESC

LIMIT 1;
```

31. Find the total amount of loans approved in the last year for each loan type

```
SELECT l.loan_type, SUM(la.loan_amt) AS total_loan_amt

FROM loan_application AS la

NATURAL JOIN loan_info as l

WHERE EXTRACT(YEAR FROM la.approved_date) = '2024'

GROUP BY l.loan_type;
```

32. Retrieve the employee(s) who have more than avg salary in each department.

```
SELECT r.dep_no, r.branch_code, e.emp_ID

FROM employee AS e

NATURAL JOIN (SELECT d.dep_no, b.branch_code, avg(e.salary) AS
avg_dep_salary

FROM employee AS e

NATURAL JOIN Branch AS b

NATURAL JOIN Department AS d

GROUP BY d.dep_no, b.branch_code) AS r

WHERE e.salary >= r.avg_dep_salary;
```

33. Find customers who have both a fixed deposit and an active loan with the bank.

```
SELECT DISTINCT r.account_no, CONCAT(c.fname,' ',c.lname) AS
full_name, fd.fd_id, l.loan_id

FROM account AS a

NATURAL JOIN Customer AS c

NATURAL JOIN

(SELECT account_no FROM account NATURAL JOIN fixed_deposit

INTERSECT

SELECT account_no FROM loan_application WHERE status = 'Approved')
AS r

NATURAL JOIN fixed_deposit AS fd

NATURAL JOIN loan_application AS la

NATURAL JOIN loan_info AS l;
```

34. Final all customers who applied for chequebook and passbook in current month

```
SELECT CONCAT(c.fname,' ',c.lname) AS full_name, r.req_id,
a.account_no, s.service_name, r.req_date

FROM customer AS c

NATURAL JOIN account AS a

NATURAL JOIN service_request AS r

NATURAL JOIN service AS s

WHERE (s.service_name = 'Chequebook' OR s.service_name = 'Passbook')
AND EXTRACT(MONTH FROM req_date) = '04';
```

35. Retrieve the total amount of transactions made from the account with account_no=5790182026699 of a current week

```
SELECT *

FROM Transaction

WHERE account_no = '1749938644158' AND DATE(date) >=
DATE_TRUNC('week', CURRENT_DATE)

AND DATE(date) < DATE_TRUNC('week', CURRENT_DATE) + INTERVAL '1
week';
```

36. List all employees along with their respective departments and branches they operate in

```
SELECT emp_ID, CONCAT(fname,' ',lname) as name, dep_name,
branch_name

FROM Employee

NATURAL JOIN Branch

NATURAL JOIN Department;
```

37. Retrieve the total number of insurance applications approved for each type of insurance

```
SELECT ins_type, COUNT(ins_app_no)

FROM Insurance_application

NATURAL JOIN Insurance_info

WHERE status='Approved'

GROUP BY ins_type;
```

38. Find the total amount of insurance premiums paid by customers in each insurance type for the current year

```
SELECT ins_type, SUM(due_amt)

FROM Insurance_record

NATURAL JOIN Insurance_application

NATURAL JOIN Insurance_Info

where EXTRACT(YEAR FROM settlement_date) = '2038'

GROUP BY ins_type;
```

39. Identify the delay penalty for each investment type

```
SELECT inv_type, delay_penalty

FROM Investment_info;
```

40. Find the total number of investment installments settled for each investment application

```
SELECT inv_app_no, count(inv_installment_no) AS
settled_installment_no

FROM Investment_payment

WHERE settlement_date is NOT NULL

GROUP BY inv_app_no;
```