

Banking Data Analysis

With  SQL

By Piyush Kadam

Table Of Content :



Introduction



**Drive Through Each
Objective**



Overview



Conclusion



1.Introduction :

Welcome to my presentation on bank analysis using SQL. This project aims to provide comprehensive insights into banking data through the effective use of SQL for data extraction, transformation, and analysis. By leveraging SQL's robust querying capabilities, we delve into various aspects of banking operations, from transaction patterns and customer behaviors to financial performance metrics.

2. Overview-

01

Accounts Table :-

- Table Consisting Information related to bank accounts of customers with the primary key as customer_id.
- This table consist the remaining balance of each customer.

02

Branches :-

- Table consist the each branch information here branch_id is the primary key.

03

Customers :-

- Gathering all the useful data about customers with customer_id as a unique id.
- Key highlight of this table is it contain the date of account created

04

Transactions :-

- This is nothing but the giant of our project giving us valuable information related customers transaction.
- The primary key of this table is transaction_id and account_number.

3. Drive Through Each Query



Q.1

Write a query to list all customers who haven't made any transactions in the last year. How can we make them active again?

```
SELECT C.customer_id,c.first_name,c.last_name
from customers c join accounts a on
c.customer_id=a.customer_id left join transactions t on
a.account_number=t.account_number
where t.transaction_id is null
or
t.transaction_date<date_sub(curdate(),interval 1 year);
```

984 Customers havent made transaction in last year from now.

	customer_id	first_name	last_name
▶	93	Jayant	Joshi
	93	Jayant	Joshi
	93	Jayant	Joshi
	93	Jayant	Joshi
	79	Mannat	Maharaj
	79	Mannat	Maharaj

Q.2

Summarize the total transaction amount per account per month.

```
SELECT account_number,year(transaction_date) as year
,month(transaction_date) as month,sum(amount) as
total_amount
from transactions t
group by
account_number,year(transaction_date),month(transaction
_date)
order by account_number,year,month;
```

	account_number	year	month	total_amount
	1013036421	2024	4	451.97
	1013036421	2024	5	3774.17
	1014303562	2022	8	916.69
	1014303562	2022	11	937.95
	1014303562	2023	3	3067.54
	1014303562	2023	4	796.23
	1014303562	2023	5	4106.88
	1014303562	2023	6	4078.78
	1014303562	2023	11	4901.29

Q.3

Rank branches based on the total amount of deposits made in the last quarter.

```
SELECT a.branch_id,sum(t.amount) , dense_rank()  
over(order by sum(t.amount) desc) as branch_rank  
from accounts a inner join transactions t  
using(account_number)  
where t.transaction_type="Deposit" and  
t.transaction_date>=date_sub(current_date(),interval 3  
month) group by a.branch_id  
order by branch_rank;
```

	branch_id	sum(t.amount)	branch_rank
▶	28	95916.67	1
	15	87606.380000000002	2
	27	85783.27	3
	20	81546.319999999998	4
	14	80660.49	5
	32	79532.869999999998	6
	18	76752.95	7
	2	76213.989999999999	8
	25	65362.260000000001	9
	17	57318.23	10
	29	56953.880000000005	11

Q.4

Find the name of the customer who has deposited the highest amount.

```
SELECT concat(c.first_name," ",c.last_name) as  
full_name, t.amount  
from customers c inner join accounts a on  
c.customer_id=a.customer_id inner join transactions t  
on t.account_number=a.account_number  
where t.transaction_type="Deposit"  
order by t.amount desc  
limit 1;
```

	full_name	amount
▶	Gokul Sengupta	4990.4

Q.5

Identify any accounts that have made more than two transactions in a single day, which could indicate fraudulent activity.

```
SELECT a.account_number as
fraud_accounts,count(t.transaction_id) as
no_of_trans,day(t.transaction_date) as single_day
from accounts a inner join transactions t
using(account_number)
group by fraud_accounts,single_day
having no_of_trans>2;
```

	fraud_accounts	no_of_trans	single_day
	1192971011	3	29
▶	1032168449	3	3
	1079185403	3	20
	1097521618	4	22
	1122354785	3	7
	1079081946	3	25
	1198065485	3	18

Q.6

Calculate the average number of transactions per customer per account per month over the last year.

	customer_id	account_number	avg_month_trans
▶	8	1100254561	3.00
	67	1035570643	2.33
	15	157415716	2.00
	31	1035499319	2.00
	63	1075015813	2.00
	28	1012982863	2.00
	5	1129054601	2.00
	34	1023672566	2.00
	65	1181559102	2.00
	25	1036932307	1.80
	22	1040006067	1.00

```
WITH MonthlyTransactions AS (  
  SELECT  
    a.customer_id,  
    a.account_number,  
    YEAR(t.transaction_date) AS transaction_year,  
    MONTH(t.transaction_date) AS transaction_month,  
    COUNT(t.transaction_id) AS num_transactions  
  FROM  
    accounts a  
  INNER JOIN  
    transactions t ON a.account_number = t.account_number  
  WHERE  
    t.transaction_date >= DATE_SUB(CURRENT_DATE(), INTERVAL 1 YEAR)  
  GROUP BY  
    a.customer_id,  
    a.account_number,  
    YEAR(t.transaction_date),  
    MONTH(t.transaction_date)  
)  
SELECT  
  customer_id,  
  account_number,  
  ROUND(AVG(num_transactions), 2) AS avg_month_trans  
FROM  
  MonthlyTransactions  
GROUP BY  
  customer_id,  
  account_number  
ORDER BY  
  avg_month_trans DESC;
```

Q.7

Write a query to find the daily transaction volume (total amount of all transactions) for the past month.

```
SELECT date(transaction_date) as  
transaction_day,round(sum(amount),3) as  
Transactions_volume  
from transactions  
where  
transaction_date>=date_sub(current_date(),interval 1  
month)  
group by transaction_day  
order by transaction_day;
```

	transaction_day	Transactions_volume
►	2024-06-11	7513.08
	2024-06-12	19277.54
	2024-06-14	14557.73
	2024-06-15	8140.17
	2024-06-16	4627.48
	2024-06-17	7027.12
	2024-06-18	9822.37
	2024-06-19	1134.2
	2024-06-20	14478.25

Q.8

Calculate the total transaction amount performed by each age group in the past year. (Age groups: 0-17, 18-30, 31-60, 60+)

SELECT case

When floor((datediff(current_date(),c.date_of_birth)/365)) between 0 and 17 then "0-17"

When floor((datediff(current_date(),c.date_of_birth)/365)) between 18 and 30 then "18-30"

When floor((datediff(current_date(),c.date_of_birth)/365)) between 31 and 60 then "31-60"

else "60+"

end as age_group,

sum(t.amount) as total_trans_amt

from customers c inner join accounts a on c.customer_id=a.customer_id

inner join transactions t on t.account_number=a.account_number

where t.transaction_date>=date_sub(current_date(),interval 1 year)

group by age_group;

	age_group	total_trans_amt
▶	60+	1031761.9599999989
	18-30	520538.68
	31-60	1166461.7999999998

Q.9

Find the branch with the highest average account balance.

```
SELECT branch_id, avg(balance) as avg_balance  
from accounts  
group by branch_id  
order by avg_balance desc  
limit 1;
```

	branch_id	avg_balance
▶	2	6997.228

Q.10

Calculate the average balance per customer at the end of each month in last year.

```
SELECT a.customer_id,year(t.transaction_date) as
year,month(t.transaction_date) as month
,round(avg(a.balance),2)as balance
from accounts a join transactions t
using(account_number)
where
t.transaction_date>=date_sub(current_date,interval 1
year)
group by customer_id,year,month
order by year,month;
```

	customer_id	year	month	balance
	9	2023	7	4905.88
	6	2023	7	2754.92
	79	2023	7	7213.44
	72	2023	7	3298.51
	59	2023	7	2260.43
	95	2023	7	7556.41
	14	2023	7	4471.8
	12	2023	7	5536.71
	53	2023	7	3899.15

4. Conclusion :

This project effectively leveraged SQL to extract, analyze, and interpret banking data, yielding actionable insights that can drive strategic decisions. The detailed analysis of customer behavior, transaction patterns, and branch performance equips the bank with the knowledge to enhance customer engagement, improve financial performance, and detect potential fraud. By continuing to utilize data analytics, the bank can foster a data-driven culture that supports growth and innovation. A one-line description of it



Thanks!

Regards-Piyush Kadam

