FINALNY PROJEKT

1)Structure of financial database

a) What are the primary keys in the individual tables?

Card – card_id

Disp – disp_id

Loan – loan_id

Order – order_id

Trans – trans_id

Client – client_id

Account – account_id

District – district_id

b) What relationships do particular pairs of tables have?

1 to 1 -> logika je v tom že 1 záznam v tabuľke X súvisí len s 1 záznamom v tabuľke Y a opačne

v našom prípade to je napr tabuľka DISP a CLIENT

1 to N -> logika je v tom že 1 záznam v tabuľke X súvisí N záunammi v tabuľke Y - ale záznamy z tabuľky Y súvisia iba s 1 záznamom v tabuľke X

v našom prípade to je napr tabuľka Account a Order kde 1 account môže mať vicero záznamov v Order (viaccero objednávok) ale teto objednávky sa viažu iba na 1 account

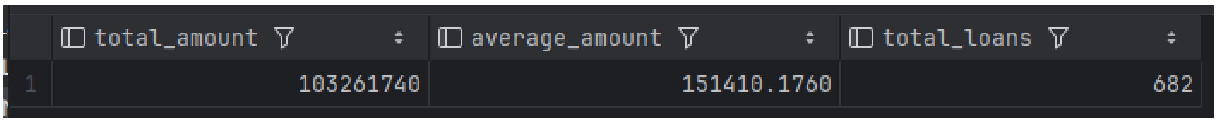
N to N -> logika je v tom že v tabuľke X môže vyť viacero záznamov, ktoré súvisia s N zaznammi v tabuľke Y a opačne - napr spojovacie tabuľky

ak sa nemýlim, v našom datasete takú tabuľku nemáme

2) History of granted loans

Display the following information as the result of the summary:

- total amount of loans,
- average loan amount,
- total number of given loans.



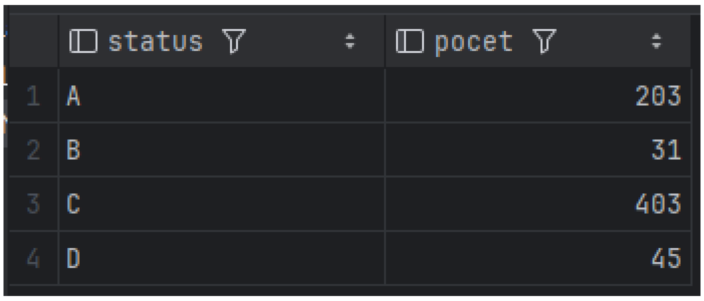
	total_amount	average_amount	total_loans
1	103261740	151410.1760	682

3) Loan status

Write a query to help you answer the question of which statuses represent repaid loans and which represent unpaid loans.

Unpaid loans – 76

Paid loans - 606



	status	pocet
1	A	203
2	B	31
3	C	403
4	D	45

4) Analysis of accounts

Write a query that ranks accounts according to the following criteria:

- number of given loans (decreasing),
- amount of given loans (decreasing),
- average loan amount,

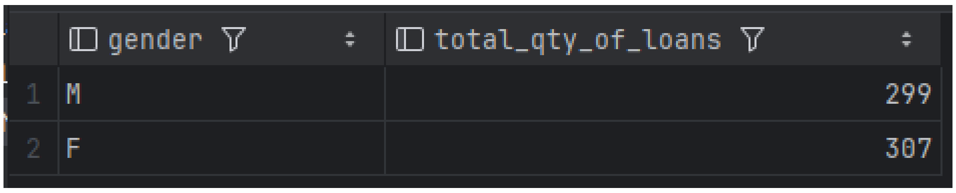
```
with cte_loans as (
    select
        account_id,
        count(loan_id) as nr_of_given_loans,
        sum(amount) as amount_of_given_loans,
        avg(amount) as avg_loan_amount
    from loan
    where status in ('A', 'C')
    group by account_id

)

select *,
    dense_rank() over (order by nr_of_given_loans desc) as ranking_sumy,
    dense_rank() over (order by amount_of_given_loans desc) as
ranking_sumy,
    dense_rank() over (order by avg_loan_amount desc) as ranking_sumy
from cte_loans;
```

5) Fully paid loans

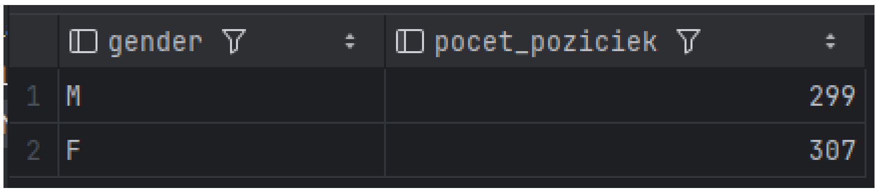
Find out the balance of repaid loans, divided by client gender.



	gender	total_qty_of_loans
1	M	299
2	F	307

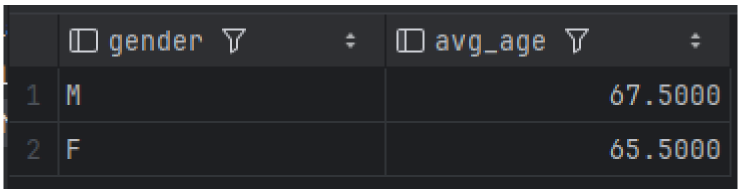
6) Client analysis - part 1

a) Who has more repaid loans - women or men?



	gender	pocet_poziciek
1	M	299
2	F	307

b) *What is the average age of the borrower divided by gender?*

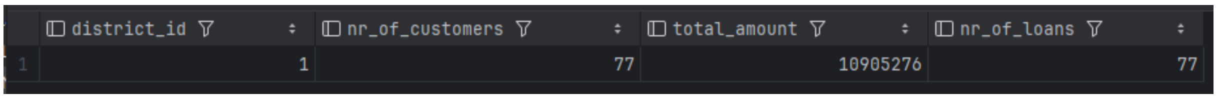


gender	avg_age
M	67.5000
F	65.5000

7) Client analysis - part 2

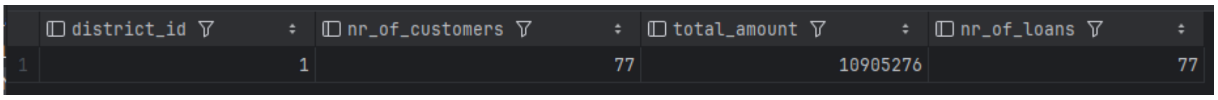
Make analyses that answer the questions:

a) *which area has the most clients,*



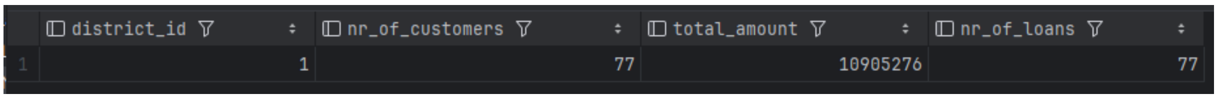
district_id	nr_of_customers	total_amount	nr_of_loans
1	77	10905276	77

b) *in which area the highest number of loans was paid,*



district_id	nr_of_customers	total_amount	nr_of_loans
1	77	10905276	77

c) *in which area the highest amount of loans was paid.*



district_id	nr_of_customers	total_amount	nr_of_loans
1	77	10905276	77

8) Client analysis - part 3

Use the query created in the previous task and modify it to determine the percentage of each district in the total amount of loans granted.

```
with cte as (
    select
        dt.district_id,
        count(c.client_id) as nr_of_customers,
        sum(amount) as total_amount,
        count(amount) as nr_of_loans
    from loan l
    join account a on l.account_id = a.account_id
    join disp d  on a.account_id = d.account_id
    join client c on d.client_id = c.client_id
    join district dt on a.district_id = dt.district_id
    where true
        and status in ('A', 'C')
        and type = 'OWNER'
    group by dt.district_id)

select *,
      total_amount / sum(total_amount) over() * 100 as share
from cte;
```

9)Selection - part 1

Check the database for the clients who meet the following results:

- their account balance is above 1000,
- they have more than 5 loans,
- they were born after 1990.

Tu nie je žiadny výsledok. Predchádzajúce analýzy ukázali, že v datasete je pre každého klienta iba 1 pôžička tj bod číslo 2 sa nedá splniť.

```
with cte as (
    select
        c.client_id ,
        c.birth_date,
        YEAR(c.birth_date) as year_of_birth,
        sum(amount - payments) as client_balance,
        count(loan_id) as loans_amount
    from loan l
    join account a on l.account_id = a.account_id
    join disp d  on a.account_id = d.account_id
    join client c on d.client_id = c.client_id
    join district dt on a.district_id = dt.district_id
    where true
        and status in ('A', 'C')
        and type = 'OWNER'
    group by c.client_id, c.birth_date)

select *
from cte
where year_of_birth > 1990
    and loans_amount >= 5
    and client_balance > 1000;
```

10) Selection part 2

From the previous exercise you probably already know that there are no customers who meet the requirements. Make an analysis to determine which condition caused the empty results.

```
with cte as (
    select
        c.client_id ,
        c.birth_date,
        YEAR(c.birth_date) as year_of_birth,
        sum(amount - payments) as client_balance,
        count(loan_id) as loans_amount
```

```sql
    from loan l
    join account a on l.account_id = a.account_id
    join disp d  on a.account_id = d.account_id
    join client c on d.client_id = c.client_id
    join district dt on a.district_id = dt.district_id
    where true
        and status in ('A', 'C')
        and type = 'OWNER'
    group by c.client_id, c.birth_date)

select *
from cte
where year_of_birth > 1990;
```

Z predchádajúcej analýzy sme zistili, že každý klient mal iba 1 pôžičku + pomocou querry vyššie vieme, že sa v datasete nenachádza klient, ktorý má date of bitrh po 1990.

11) *Expiring cards*

```sql
drop procedure tg_expirated_cards;
delimiter //
create procedure tg_expirated_cards(in in_date date)
begin
    with cte as (
        select
            c.client_id,
            card_id,
            cast(issued as date),
            A3,
            date_add(issued, interval 3 year) as expiration_date
        from card
        join disp d on card.disp_id = d.disp_id
        join client c on d.client_id = c.client_id
        join district as dt on c.district_id = dt.district_id),

    cte_2 as (
    select *,
           date_add(expiration_date, interval -7 day) as ready_to_contact
    from cte)

    select *
    from cte_2
    where ready_to_contact = in_date;
end; //

call tg_expirated_cards('2001-08-02');
```