SQL для анализа данных с Глебом Михайловым

Мой курс на Юдеми https://glebmikhaylov.com/sql (https://glebmikhaylov.com/sql)
Мой канал в Телеграм: https://t.me/mikhaylovgleb (https://t.me/mikhaylovgleb)
Мой канал на Ютюб: https://www.youtube.com/c/GlebMikhaylov (https://www.youtube.com/c/GlebMikhaylov)
Мой сайт: https://glebmikhaylov.com/ (https://glebmikhaylov.com/)

Все файлы и данные можно найти в репозитории на GitHub (https://github.com/glebmikha/sql-course). Основной ноутбук курса лучше открывать сразу на Colab. Вот ссылка (https://colab.research.google.com/drive/1Og4wDz-BELxR6izJyWFX-Wn3HVFPHE3W?usp=sharing) на основной ноутбук со всеми примерами.

In [1]:

```
import pandas as pd
import numpy as np
```

01-connect-create-table

Подключение к бд и заливка данных

Установка и подключение SQLAlchemy к базе данных: mysql, postgresql, sqlite3 и oracle (https://pythonru.com/biblioteki/ustanovka-i-podklyuchenie-sqlalchemy-k-baze-dannyh)

```
pip install sqlalchemy
```

In [2]:

```
import sqlalchemy
```

In [3]:

```
sqlalchemy.__version__
```

Out[3]:

```
'1.4.39'
```

```
!pip install pyodbc
```

In [4]:

```
import pyodbc
```

In [5]:

```
import warnings
warnings.filterwarnings('ignore')
```

In [6]:

```
conn = pyodbc.connect('DSN=TestDB;Trusted_Connection=yes;')
```

In [7]:

```
def select(sql):
  return pd.read_sql(sql,conn)
```

In [8]:

```
cur = conn.cursor()
sql = '''
drop table if exists Employee
create table Employee(Id int, Salary int)
insert into Employee(Id, Salary) values (1, 100)
insert into Employee(Id, Salary) values (2, 200)
insert into Employee(Id, Salary) values (3, 300)
'''
cur.execute(sql)
conn.commit()
cur.close()
sql = '''select * from Employee t'''
select(sql)
```

Out[8]:

	Id	Salary
0	1	100
1	2	200
2	3	300

Создание, подключение и заливка данных

In [9]:

```
df = pd.read_csv('../data/german_credit_augmented.csv')
# df
```

In [10]:

```
df['contract_dt'] = pd.to_datetime(df['contract_dt'],format='%Y-%m-%d %H:%M:%S')
```

In [11]:

```
df.dtypes
```

Out[11]:

```
age                            int64
sex                           object
job                            int64
housing                       object
saving_accounts               object
checking_account              object
credit_amount                  int64
duration                       int64
purpose                       object
default                        int64
contract_dt           datetime64[ns]
client_id                      int64
dtype: object
```

```
# не работает
df.fillna(sqlalchemy.sql.null(), inplace=True)
```

In [12]:

```
df = df.replace({np.nan:None})
# df
```

[Что выбрать, text или varchar (MAX)? (https://vc.ru/dev/245799-chto-vybrat-text-ili-varchar-max)](https://vc.ru/dev/245799-chto-vybrat-text-ili-varchar-max)

[Вставка кадра данных Python в таблицу SQL (https://learn.microsoft.com/ru-ru/sql/machine-learning/data-exploration/python-dataframe-sql-server?view=azuresqldb-current)](https://learn.microsoft.com/ru-ru/sql/machine-learning/data-exploration/python-dataframe-sql-server?view=azuresqldb-current)
[Сопоставления типов данных между Python и SQL Server (https://learn.microsoft.com/ru-ru/sql/machine-learning/python/python-libraries-and-data-types?source=recommendations&view=sql-server-ver16)](https://learn.microsoft.com/ru-ru/sql/machine-learning/python/python-libraries-and-data-types?source=recommendations&view=sql-server-ver16)

In [13]:

```
cur = conn.cursor()
sql = '''
drop table if exists german_credit;
CREATE TABLE german_credit (
    age              INTEGER,
    sex              VARCHAR(max),
    job              INTEGER,
    housing          VARCHAR(max),
    saving_accounts  VARCHAR(max),
    checking_account VARCHAR(max),
    credit_amount    INTEGER,
    duration         INTEGER,
    purpose          VARCHAR(max),
    [default]        INTEGER,
    contract_dt      DATETIME,
    client_id        INTEGER
);
'''
cur.execute(sql)
conn.commit()

for index,row in df.head(1000).iterrows():
    cur.execute('''INSERT INTO german_credit(
                    [age],[sex],[job],[housing],[saving_accounts],
                    [checking_account],[credit_amount],[duration],[purpose],[default],
                    [contract_dt],[client_id])
                    values (?,?,?,?,?,?,?,?,?,?,?,?)
    ''',
                    row['age'],
                    row['sex'],
                    row['job'],
                    row['housing'],
                    row['saving_accounts'],
                    row['checking_account'],
                    row['credit_amount'],
                    row['duration'],
                    row['purpose'],
                    row['default'],
                    row['contract_dt'],
                    row['client_id'])

conn.commit()
cur.close()
sql = '''select * from german_credit t'''
select(sql)
```

Out[13]:

	age	sex	job	housing	saving_accounts	checking_account	credit_amount	duration
0	33	male	2	own	None	None	3074	9
1	43	male	1	own	little	little	1344	12
2	52	male	2	own	quite rich	None	936	9
3	35	female	3	own	little	None	1393	11

	age	sex	job	housing	saving_accounts	checking_account	credit_amount	duration
4	28	male	2	own	little	None	776	12
...	...	...	...	...	...	...	...	...
995	65	male	2	free	little	little	2600	18
996	30	male	3	own	little	moderate	4455	36
997	33	male	2	own	little	moderate	6403	24
998	29	female	2	own	None	None	5003	21
999	44	male	2	own	moderate	moderate	1804	12

1000 rows × 12 columns

In [14]:

```
transactions = pd.read_csv('../data/german_credit_augmented_transactions.csv')
transactions['dt'] = pd.to_datetime(transactions['dt'],format='%Y-%m-%d %H:%M:%S')
transactions = transactions.replace({np.nan:None})

cur = conn.cursor()
sql = '''
drop table if exists client_transactions;
CREATE TABLE client_transactions (
    dt                  datetime,
    client_id           int,
    amount              decimal(19,4)
);
'''
cur.execute(sql)
conn.commit()

for index,row in transactions.iterrows():
    cur.execute('''INSERT INTO client_transactions(
                    [dt],[client_id],[amount]
                    )
                    values (?,?,?)
    ''',
                    row['dt'],
                    row['client_id'],
                    row['amount']
               )

conn.commit()
cur.close()
sql = '''select * from client_transactions t'''
select(sql)
```

Out[14]:

	dt	client_id	amount
0	2008-04-06 11:54:47	950	161.38
1	2007-07-28 00:00:19	418	35.34
2	2008-03-14 20:43:54	131	146.50
3	2007-12-18 13:03:24	353	119.21
4	2007-11-09 05:18:30	849	105.24
...	...	...	...
4270	2007-08-18 04:05:05	185	10063.07
4271	2007-06-04 15:23:32	375	156.91
4272	2007-12-06 21:34:06	418	10053.82
4273	2008-04-19 17:30:07	409	10050.35
4274	2007-11-07 19:44:50	674	165.60

4275 rows × 3 columns

02-select

In [15]:

```
sql = '''select * from german_credit t'''
select(sql)
```

Out[15]:

	age	sex	job	housing	saving_accounts	checking_account	credit_amount	duration
0	33	male	2	own	None	None	3074	9
1	43	male	1	own	little	little	1344	12
2	52	male	2	own	quite rich	None	936	9
3	35	female	3	own	little	None	1393	11
4	28	male	2	own	little	None	776	12
...	...	...	...	...	...	...	...	...
995	65	male	2	free	little	little	2600	18
996	30	male	3	own	little	moderate	4455	36
997	33	male	2	own	little	moderate	6403	24
998	29	female	2	own	None	None	5003	21
999	44	male	2	own	moderate	moderate	1804	12

1000 rows × 12 columns

3. Select

1. Псевдонимы

```
In [16]:
```

```
sql = '''
SELECT t.age * 3 AS age_mult3,
   t.housing
FROM german_credit AS t
'''
select(sql)
```

```
Out[16]:
```

	age_mult3	housing
0	99	own
1	129	own
2	156	own
3	105	own
4	84	own
...	...	...
995	195	free
996	90	own
997	99	own
998	87	own
999	132	own

1000 rows × 2 columns

2. Базовые операции со столбцами

In [17]:

```
sql = '''
select t.*,
    t.age * 3 as age_mult3,
    t.age + t.credit_amount as age_plus_amount,
    t.age * 1.0 / t.credit_amount as age_div_amount,
    t.age as age_2
from german_credit t
'''
select(sql)
```

Out[17]:

	age	sex	job	housing	saving_accounts	checking_account	credit_amount	duration
0	33	male	2	own	None	None	3074	9
1	43	male	1	own	little	little	1344	12
2	52	male	2	own	quite rich	None	936	9
3	35	female	3	own	little	None	1393	11
4	28	male	2	own	little	None	776	12
...	...	...	...	...	...	...	...	...
995	65	male	2	free	little	little	2600	18
996	30	male	3	own	little	moderate	4455	36
997	33	male	2	own	little	moderate	6403	24
998	29	female	2	own	None	None	5003	21
999	44	male	2	own	moderate	moderate	1804	12

1000 rows × 16 columns

3. Where

In [18]:

```
sql = '''
select count(1)
from german_credit t
where t.contract_dt between
    Convert(Date, '01.01.2007', 104) and Convert(Date, '31.12.2007', 104)
'''
select(sql)
```

Out[18]:

0	573

In [19]:

```
sql = '''
select * from german_credit t
where t.contract_dt between
        Convert(Date, '01.01.2007', 104) and Convert(Date, '31.12.2007', 104)
    and t.purpose in ('car' ,'repairs')
order by t.contract_dt desc, credit_amount
'''
select(sql)
```

Out[19]:

	age	sex	job	housing	saving_accounts	checking_account	credit_amount	duration	p
0	36	male	3	rent	None	moderate	7057	20	
1	30	male	2	own	little	moderate	639	12	
2	25	male	2	rent	moderate	moderate	1264	15	
3	48	male	2	own	little	None	2134	9	
4	67	female	2	own	little	moderate	3872	18	
...	...	...	...	...	...	...	...	...	
192	55	male	2	own	rich	None	1413	12	
193	55	female	0	free	little	little	1190	18	
194	47	male	3	own	little	moderate	1209	6	
195	36	male	2	own	little	moderate	884	18	
196	31	male	2	own	little	None	2775	18	

197 rows × 12 columns

5. Case when

Доля клиентов с размером кредита > 1000:

In [20]:

```
sql = '''
select count(*) from german_credit t
'''
select(sql)
```

Out[20]:

```
0   1000
```

In [21]:

```
sql = '''
select count(*) from german_credit t
where t.credit_amount > 1000
'''
select(sql)
```

Out[21]:

```
0   884
```

In [22]:

```
884/1000
```

Out[22]:

```
0.884
```

In [23]:

```
sql = '''
select t.credit_amount,
    case when
            t.credit_amount > 1000 then 1
        else 0
    end as greater_1000_flag,
    iif(t.credit_amount > 1000,1,0) as greater_1000_flag2
from german_credit t
'''
select(sql)
```

Out[23]:

	credit_amount	greater_1000_flag	greater_1000_flag2
0	3074	1	1
1	1344	1	1
2	936	0	0
3	1393	1	1
4	776	0	0
...	...	...	...
995	2600	1	1
996	4455	1	1
997	6403	1	1
998	5003	1	1
999	1804	1	1

1000 rows × 3 columns

In [24]:

```
sql = '''
select
    avg(
        case when t.credit_amount > 1000 then 1.0 else 0 end
        ) as greater_1000_frac
from german_credit t
'''
select(sql)
```

Out[24]:

	greater_1000_frac
0	0.884

7. Создание таблицы

In [25]:

```
cur = conn.cursor()
sql = '''
drop table if exists greater_1000_credit;

select *
into greater_1000_credit
from german_credit t
where t.credit_amount > 1000
'''
cur.execute(sql)
conn.commit()
cur.close()
```

In [26]:

```
sql = '''select * from greater_1000_credit t'''
select(sql)
```

Out[26]:

	age	sex	job	housing	saving_accounts	checking_account	credit_amount	duration	
0	40	male	1	own	little	little	3939	11	
1	58	female	1	free	little	little	6143	48	
2	52	male	3	own	None	moderate	6468	12	
3	32	male	2	own	little	moderate	6078	12	
4	61	male	1	own	little	None	1255	12	
...	...	...	...	...	...	...	...	...	
879	40	male	3	own	None	little	1977	36	
880	23	male	1	own	little	moderate	1048	10	
881	25	male	2	rent	moderate	moderate	1264	15	
882	55	female	3	free	little	little	2578	12	fı
883	40	male	3	own	moderate	None	4623	15	

884 rows × 12 columns

03-group-by

In [27]:

```
sql = '''select * from german_credit t'''
select(sql)
```

Out[27]:

	age	sex	job	housing	saving_accounts	checking_account	credit_amount	duration
0	33	male	2	own	None	None	3074	9
1	43	male	1	own	little	little	1344	12
2	52	male	2	own	quite rich	None	936	9
3	35	female	3	own	little	None	1393	11
4	28	male	2	own	little	None	776	12
...	...	...	...	...	...	...	...	...
995	65	male	2	free	little	little	2600	18
996	30	male	3	own	little	moderate	4455	36
997	33	male	2	own	little	moderate	6403	24
998	29	female	2	own	None	None	5003	21
999	44	male	2	own	moderate	moderate	1804	12

1000 rows × 12 columns

4. Group By

1. Сводная таблица

В сводных таблицах всегда дожен быть *count*

In [28]:

```
sql = '''
select
    t.sex,
    count(*) as cnt,
    -- поля FLOAT должны, поэтому и не точность
    avg(t.credit_amount * 1.0) as credit_amount_avg
from german_credit t
group by t.sex
'''
select(sql)
```

Out[28]:

	sex	cnt	credit_amount_avg
0	female	310	2877.774193
1	male	690	3448.040579

In [29]:

```
df.groupby('sex')['credit_amount'].agg(['count','mean'])
```

Out[29]:

	count	mean
sex		
female	310	2877.774194
male	690	3448.040580

Уникальные значения:

In [30]:

```
sql = '''
select
    count(distinct t.housing),
    count(t.housing)
from german_credit t
'''
select(sql)
```

Out[30]:

0	3	1000

In [31]:

```
sql = '''
select t.housing,
    count(*) as cnt,
    avg(t.credit_amount * 1.0) as credit_amount_avg
from german_credit t
group by t.housing
'''
select(sql)
```

Out[31]:

	housing	cnt	credit_amount_avg
0	free	108	4906.212962
1	own	713	3060.939691
2	rent	179	3122.553072

2. Пропущенные значения (null)

In [32]:

```
sql = '''
select
    count(t.checking_account),
    count(0)
from german_credit t
'''
select(sql)
```

Out[32]:

0	606	1000

In [33]:

```
sql = '''
select t.checking_account,
    count(*) as cnt,
    avg(t.credit_amount) as credit_amount_avg
from german_credit t
group by t.checking_account
'''
select(sql)
```

Out[33]:

	checking_account	cnt	credit_amount_avg
0	None	394	3133
1	little	274	3175
2	moderate	269	3827
3	rich	63	2177

In [34]:

```
df.groupby('checking_account',dropna=False)['credit_amount'].count()
```

Out[34]:

```
checking_account
little      274
moderate    269
rich         63
NaN         394
Name: credit_amount, dtype: int64
```

In [35]:

```
sql = '''
select
    sum(
        case when t.checking_account is null then 1 else 0 end
        ) as is_null,
    count(
        case when t.checking_account is null then 1 else null end
        ) as is_null2
from german_credit t
'''
select(sql)
```

Out[35]:

	is_null	is_null2
0	394	394

потренируемся:

In [36]:

```
t = pd.DataFrame({'col1':[1,np.nan,2]})
t = t.replace({np.nan:None})
# t
```

In [37]:

```
cur = conn.cursor()
sql = '''
drop table if exists null_test;
CREATE TABLE null_test (
    col1   money
);
'''
cur.execute(sql)
conn.commit()

for index,row in t.iterrows():
    cur.execute('''INSERT INTO null_test(
                    [col1]
                    )
                    values (?)
    ''',

                row['col1']
               )

conn.commit()
cur.close()

sql = '''select * from null_test t'''
select(sql)
```

Out[37]:

	col1
0	1.0
1	NaN
2	2.0

In [38]:

```
(1 + 2) / 2
```

Out[38]:

```
1.5
```

In [39]:

```
(1 + 0 + 2) / 3
```

Out[39]:

```
1.0
```

In [40]:

```
sql = '''
select avg(t.col1) from null_test t
'''
select(sql)
```

Out[40]:

0	1.5

заменим пропуски:

In [41]:

```
sql = '''
select
    t.checking_account,
    coalesce(t.checking_account,'no_info')
from german_credit t
'''
select(sql)
```

Out[41]:

	checking_account	
0	None	no_info
1	little	little
2	None	no_info
3	None	no_info
4	None	no_info
...	...	...
995	little	little
996	moderate	moderate
997	moderate	moderate
998	None	no_info
999	moderate	moderate

1000 rows × 2 columns

coalesce:

In [42]:

```
t = pd.DataFrame({'col1':[1,np.nan,2],
                  'col2':[np.nan,np.nan,1],
                  'col3':[1,2,3]})
t = t.replace({np.nan:None})
# t
```

In [43]:

```
cur = conn.cursor()
sql = '''
drop table if exists null_test;
CREATE TABLE null_test (
    col1        INTEGER,
    col2        INTEGER,
    col3        INTEGER
);
'''
cur.execute(sql)
conn.commit()

for index,row in t.iterrows():
    cur.execute('''INSERT INTO null_test(
                    [col1],[col2],[col3]
                    )
                    values (?,?,?)
    ''',
                row['col1'],
                row['col2'],
                row['col3'],
               )

conn.commit()
cur.close()

sql = '''select * from null_test t'''
select(sql)
```

Out[43]:

	col1	col2	col3
0	1.0	NaN	1
1	NaN	NaN	2
2	2.0	1.0	3

In [44]:

```
sql = '''
select t.*,
coalesce(t.col1, t.col2, t.col3) as res
from null_test t
'''
select(sql)
```

Out[44]:

	col1	col2	col3	res
0	1.0	NaN	1	1
1	NaN	NaN	2	2
2	2.0	1.0	3	2

3. Дубликаты

3. Дубликаты

In [45]:

```
t = pd.DataFrame({'id':[1,1,2],'name':['a','a','b']})
# t
```

In [46]:

```
cur = conn.cursor()
sql = '''
drop table if exists dupl_test;
CREATE TABLE dupl_test (
    id          INTEGER,
    name        VARCHAR(max)
);
'''
cur.execute(sql)
conn.commit()

for index,row in t.iterrows():
    cur.execute('''INSERT INTO dupl_test(
                    [id],[name]
                    )
                    values (?,?)
    ''',
                row['id'],
                row['name']
               )

conn.commit()
cur.close()

sql = '''select * from dupl_test t'''
select(sql)
```

Out[46]:

	id	name
0	1	a
1	1	a
2	2	b

группируем на все поля и посчитаем строки:

In [47]:

```
sql = '''
select t.id, t.name,
    count(1) as cnt
from dupl_test t
group by t.id, t.name
'''
select(sql)
```

Out[47]:

	id	name	cnt
0	1	a	2
1	2	b	1

In [48]:

```
sql = '''
select t.id, t.name,
    count(1) as cnt
from dupl_test t
group by t.id, t.name
having count(1) > 1
'''
select(sql)
```

Out[48]:

	id	name	cnt
0	1	a	2

дубликат Id:

In [49]:

```
t = pd.DataFrame({'id':[1,1,2,2,3],
                  'name':['a','b','c','d','e']})
# t
```

In [50]:

```
cur = conn.cursor()
sql = '''
drop table if exists dupl_test;
CREATE TABLE dupl_test (
    id         INTEGER,
    name       VARCHAR(max)
);
'''
cur.execute(sql)
conn.commit()

for index,row in t.iterrows():
    cur.execute('''INSERT INTO dupl_test(
                    [id],[name]
                    )
                    values (?,?)
    ''',
                row['id'],
                row['name']
               )

conn.commit()
cur.close()

sql = '''select * from dupl_test t'''
select(sql)
```

Out[50]:

	id	name
0	1	a
1	1	b
2	2	c
3	2	d
4	3	e

In [51]:

```
sql = '''
select t.id,
    count(1) as cnt from dupl_test t
group by t.id
having count(1) > 1
'''
select(sql)
```

Out[51]:

	id	cnt
0	1	2
1	2	2

In [52]:

```
sql = '''
select * from dupl_test t
where t.id in (1,2)
'''
select(sql)
```

Out[52]:

	id	name
0	1	a
1	1	b
2	2	c
3	2	d

Используя подзапросы:

In [53]:

```
sql = '''
select t.id as cnt
from dupl_test t
group by t.id
having count(1) > 1
'''
select(sql)
```

Out[53]:

	cnt
0	1
1	2

In [54]:

```
sql = '''
select *
from dupl_test t
where t.id in (
        select t.id as cnt
    from dupl_test t
    group by t.id
    having count(1) > 1
    )
'''
select(sql)
```

Out[54]:

	id	name
0	1	a
1	1	b
2	2	c
3	2	d

4. Агрегация

In [55]:

```
sql = '''
select year(t.contract_dt) as year,  month(t.contract_dt) as month,
    count(1) as credit_cnt,
    count(distinct t.client_id) as client_id_unique,
    sum(t.credit_amount) as credit_amount_sum,
    avg(t.credit_amount * 1.0) as credit_amount_avg
from german_credit t
group by year(t.contract_dt),  month(t.contract_dt)
order by year(t.contract_dt),  month(t.contract_dt)
'''
select(sql)
```

Out[55]:

	year	month	credit_cnt	client_id_unique	credit_amount_sum	credit_amount_avg
0	2007	5	81	81	207663	2563.740740
1	2007	6	74	74	239594	3237.756756
2	2007	7	71	71	224333	3159.619718
3	2007	8	57	57	178569	3132.789473
4	2007	9	58	58	186909	3222.568965
5	2007	10	70	70	188534	2693.342857
6	2007	11	87	87	300504	3454.068965
7	2007	12	77	77	273973	3558.090909
8	2008	1	93	93	288080	3097.634408
9	2008	2	55	55	211128	3838.690909
10	2008	3	63	63	204944	3253.079365
11	2008	4	85	85	305409	3593.047058
12	2008	5	67	67	263043	3926.014925
13	2008	6	62	62	198575	3202.822580

5. Создание интервалов (или бинов или бакетов)

Уникальные значения:

In [56]:

```
sql = '''
select
    count(distinct t.credit_amount)
from german_credit t
'''
select(sql)
```

Out[56]:

0	921

Введём диапозоны:

In [57]:

```
sql = '''
select t.credit_amount,
    case
        when t.credit_amount < 1000 then '1. <1000'
        when t.credit_amount < 2000 then '2. 1000-2000'
        when t.credit_amount < 3000 then '3. 2000-3000'
        when t.credit_amount >= 3000 then '4. >= 3000'
        else 'other'
    end as credit_amount_bin
from german_credit t
'''
select(sql)
```

Out[57]:

	credit_amount	credit_amount_bin
0	3074	4. >= 3000
1	1344	2. 1000-2000
2	936	1. <1000
3	1393	2. 1000-2000
4	776	1. <1000
...	...	...
995	2600	3. 2000-3000
996	4455	4. >= 3000
997	6403	4. >= 3000
998	5003	4. >= 3000
999	1804	2. 1000-2000

1000 rows × 2 columns

In [58]:

```
sql = '''
select
    case
        when t.credit_amount < 1000 then '1. <1000'
        when t.credit_amount < 2000 then '2. 1000-2000'
        when t.credit_amount < 3000 then '3. 2000-3000'
        when t.credit_amount >= 3000 then '4. >= 3000'
        else 'other'
    end as credit_amount_bin,
    count(1) as credit_cnt
from german_credit t
group by
    case
        when t.credit_amount < 1000 then '1. <1000'
        when t.credit_amount < 2000 then '2. 1000-2000'
        when t.credit_amount < 3000 then '3. 2000-3000'
        when t.credit_amount >= 3000 then '4. >= 3000'
        else 'other'
    end
order by
    case
        when t.credit_amount < 1000 then '1. <1000'
        when t.credit_amount < 2000 then '2. 1000-2000'
        when t.credit_amount < 3000 then '3. 2000-3000'
        when t.credit_amount >= 3000 then '4. >= 3000'
        else 'other'
    end
'''
select(sql)
```

Out[58]:

	credit_amount_bin	credit_cnt
0	1. <1000	116
1	2. 1000-2000	316
2	3. 2000-3000	188
3	4. >= 3000	380

6. Переменные в столбцах сводной таблицы

Pivot таблицы:

In [59]:

```
sql = '''
select t.housing,
    count(
        case when t.sex = 'female' then 1 else null end
        ) as female,
    count(
        case when t.sex = 'male' then 1 else null end
        ) as male,
    count(1) as cnt
from german_credit t
group  by t.housing
'''
select(sql)
```

Out[59]:

	housing	female	male	cnt
0	free	19	89	108
1	own	196	517	713
2	rent	95	84	179

автоматизируем в Python:

In [60]:

```
sql = '''
select distinct t.purpose
from german_credit t
'''
select(sql)
```

Out[60]:

	purpose
0	business
1	car
2	domestic appliances
3	education
4	furniture/equipment
5	radio/TV
6	repairs
7	vacation/others

In [61]:

```
purpose = list(select(sql)['purpose'].values)
purpose
```

Out[61]:

```
['business',
 'car',
 'domestic appliances',
 'education',
 'furniture/equipment',
 'radio/TV',
 'repairs',
 'vacation/others']
```

In [62]:

```
for p in purpose:
  print(f"count(case when t.purpose = '{p}' then 1 else null end) as {p.lower().replace(' '
```

```
count(case when t.purpose = 'business' then 1 else null end) as business,
count(case when t.purpose = 'car' then 1 else null end) as car,
count(case when t.purpose = 'domestic appliances' then 1 else null end) as d
omesticappliances,
count(case when t.purpose = 'education' then 1 else null end) as education,
count(case when t.purpose = 'furniture/equipment' then 1 else null end) as f
urnitureequipment,
count(case when t.purpose = 'radio/TV' then 1 else null end) as radiotv,
count(case when t.purpose = 'repairs' then 1 else null end) as repairs,
count(case when t.purpose = 'vacation/others' then 1 else null end) as vacat
ionothers,
```

In [63]:

```
sql = '''
select t.housing,
    count(case when t.purpose = 'radio/TV' then 1 else null end) as radiotv,
    count(case when t.purpose = 'car' then 1 else null end) as car,
    count(case when t.purpose = 'education' then 1 else null end) as education,
    count(case when t.purpose = 'furniture/equipment' then 1 else null end) as furnitureequ
    count(case when t.purpose = 'repairs' then 1 else null end) as repairs,
    count(case when t.purpose = 'business' then 1 else null end) as business,
    count(case when t.purpose = 'domestic appliances' then 1 else null end) as domesticappl
    count(case when t.purpose = 'vacation/others' then 1 else null end) as vacationothers,
    count(1) as cnt
from german_credit t
group  by t.housing
'''
select(sql)
```

Out[63]:

	housing	radiotv	car	education	furnitureequipment	repairs	business	domesticappliances
0	free	15	55	15	11	3	5	0
1	own	227	219	34	122	17	76	10
2	rent	38	63	10	48	2	16	2

7. Создание категорий из текстовых данных (like)

пример разрозненных данных:

In [64]:

```
t = pd.DataFrame({'purpose':['машина','машина','машина','на машину','на покупку машины',
                             'автомобиль','на возвращение 2007',
                             'на свадьбу','свадьба','свадьба','свадьба','для свадьбы',
                             'недвижимость','на покупку недвижимости']})
# t
```

In [65]:

```
cur = conn.cursor()
sql = '''
drop table if exists purpose;
CREATE TABLE purpose (
    purpose      VARCHAR(max)
);
'''
cur.execute(sql)
conn.commit()

for index,row in t.iterrows():
    cur.execute('''INSERT INTO purpose(
                    [purpose]
                    )
                    values (?)
    ''',
                    row['purpose']
               )

conn.commit()
cur.close()

sql = '''select * from purpose t'''
select(sql)
```

Out[65]:

	purpose
0	машина
1	машина
2	машина
3	на машину
4	на покупку машины
5	автомобиль
6	на возвращение 2007
7	на свадьбу
8	свадьба
9	свадьба
10	свадьба
11	для свадьбы
12	недвижимость
13	на покупку недвижимости

проверим на уникальные значения:

In [66]:

```
sql = '''
select t.purpose,
    count(1) from purpose t
group by t.purpose
order by count(1) desc
'''
select(sql)
```

Out[66]:

	purpose	
0	машина	3
1	свадьба	3
2	автомобиль	1
3	для свадьбы	1
4	на возвращение 2007	1
5	на машину	1
6	на покупку машины	1
7	на покупку недвижимости	1
8	на свадьбу	1
9	недвижимость	1

выберем общее:

In [67]:

```
cat = '''
select t.purpose,
    case
        when t.purpose like '%свадьб%' then 'свадьба'
        when t.purpose like '%машин%' or t.purpose like '%авто%' then 'машина'
        when t.purpose like '%недвиж%' then 'недвижимость'
        else 'другое'
    end as purpose_cat
from purpose t
'''
select(cat)
```

Out[67]:

	purpose	purpose_cat
0	машина	машина
1	машина	машина
2	машина	машина
3	на машину	машина
4	на покупку машины	машина
5	автомобиль	машина
6	на возвращение 2007	другое
7	на свадьбу	свадьба
8	свадьба	свадьба
9	свадьба	свадьба
10	свадьба	свадьба
11	для свадьбы	свадьба
12	недвижимость	недвижимость
13	на покупку недвижимости	недвижимость

In [68]:

```
sql = '''
select t.purpose_cat,
    count(1)
from (
    select t.purpose,
    case
        when t.purpose like '%свадьб%' then 'свадьба'
        when t.purpose like '%машин%' or t.purpose like '%авто%' then 'машина'
        when t.purpose like '%недвиж%' then 'недвижимость'
        else 'другое'
    end as purpose_cat
    from purpose t
    ) t
group by t.purpose_cat
'''
select(sql)
```

Out[68]:

	purpose_cat	
0	другое	1
1	машина	6
2	недвижимость	2
3	свадьба	5

In [69]:

```
sql = f'''
select t.purpose_cat,
    count(1)
from ({cat}) t
group by t.purpose_cat
'''
select(sql)
```

Out[69]:

	purpose_cat	
0	другое	1
1	машина	6
2	недвижимость	2
3	свадьба	5

In [70]:

```
sql = f'''
select t.purpose,
    count(1)
from ({cat}) t
where t.purpose_cat = 'другое'
group by t.purpose
order by count(1) desc'''
select(sql)
```

Out[70]:

	purpose	
0	на возвращение 2007	1

04- subqueries

5. Подзапросы

1. Простой подзапрос

In [71]:

```
t = pd.DataFrame({'id':[1,1,2,2,3],
                  'name':['a','b','c','d','e']})
# t
```

In [72]:

```
cur = conn.cursor()
sql = '''
drop table if exists dupl_test;
CREATE TABLE dupl_test (
    id          INTEGER,
    name        VARCHAR(max)
);
'''
cur.execute(sql)
conn.commit()

for index,row in t.iterrows():
    cur.execute('''INSERT INTO dupl_test(
                    [id],[name]
                    )
                    values (?,?)
    ''',
                    row['id'],
                    row['name']
               )

conn.commit()
cur.close()
sql = '''select * from dupl_test t'''
select(sql)
```

Out[72]:

	id	name
0	1	a
1	1	b
2	2	c
3	2	d
4	3	e

Дубликаты Id:

In [73]:

```
sql = '''
select t.id
from dupl_test t
group by t.id
having count(1) > 1
'''
select(sql)
```

Out[73]:

	id
0	1
1	2

In [74]:

```
sql = '''
select *
from dupl_test t
where t.id in (
    select t.id
    from dupl_test t
    group by t.id
    having count(1) > 1
)
'''
select(sql)
```

Out[74]:

	id	name
0	1	a
1	1	b
2	2	c
3	2	d

с созданием промежуточной таблицы:

In [75]:

```
cur = conn.cursor()
sql = '''
drop table if exists dupls;

select t.id
into dupls
from dupl_test t
group by t.id
having count(1) > 1
'''
cur.execute(sql)
conn.commit()
cur.close()
sql = '''select * from dupls t'''
select(sql)
```

Out[75]:

	id
0	1
1	2

In [76]:

```
sql = '''
select *
from dupl_test t
where t.id in (
    select id from dupls
)
'''
select(sql)
```

Out[76]:

	id	name
0	1	a
1	1	b
2	2	c
3	2	d

having в подзапросах:

In [77]:

```
sql = '''
select t.id,
    count(1) as cnt
from dupl_test t
group by t.id
having count(1) > 1'''
select(sql)
```

Out[77]:

	id	cnt
0	1	2
1	2	2

In [78]:

```
sql = '''
select * from (
    select t.id,
        count(1) as cnt
    from dupl_test t
    group by t.id
) t
where t.cnt > 1
'''
select(sql)
```

Out[78]:

	id	cnt
0	1	2
1	2	2

2. CTE (with)

In [79]:

```
sql = '''
select * from (
    select * from (
        select t.id,
            count(1) as cnt
        from dupl_test t
        group by t.id
    ) t
    where t.cnt > 1
) t
where t.id = 1
'''
select(sql)
```

Out[79]:

	id	cnt
0	1	2

In [80]:

```
sql = '''
with
id_cnt as (
    select t.id,
        count(1) as cnt
    from dupl_test t
    group by t.id
),
id_cnt_2 as (
    select *
    from id_cnt t
    where t.cnt > 1
)
select * from id_cnt_2 t
where t.id = 1
'''
select(sql)
```

Out[80]:

	id	cnt
0	1	2

закрепим понимание:

In [81]:

```
cat = '''
select t.purpose,
case
    when t.purpose like '%свадьб%' then 'свадьба'
    when t.purpose like '%машин%' or t.purpose like '%авто%' then 'машина'
    when t.purpose like '%недвиж%' then 'недвижимость'
    else 'другое'
end as purpose_cat
from purpose t
'''
print(cat)
```

```
select t.purpose,
case
    when t.purpose like '%свадьб%' then 'свадьба'
    when t.purpose like '%машин%' or t.purpose like '%авто%' then 'машина'
    when t.purpose like '%недвиж%' then 'недвижимость'
    else 'другое'
end as purpose_cat
from purpose t
```

In [82]:

```
sql = f'''
select t.purpose_cat,
    count(1)
from ({cat}) t
group by t.purpose_cat
'''
```

In [83]:

```
print(sql)
```

```

select t.purpose_cat,
    count(1)
from (
select t.purpose,
case
    when t.purpose like '%свадьб%' then 'свадьба'
    when t.purpose like '%машин%' or t.purpose like '%авто%' then 'машина'
    when t.purpose like '%недвиж%' then 'недвижимость'
    else 'другое'
end as purpose_cat
from purpose t
) t
group by t.purpose_cat
```

In [84]:

```
select(sql)
```

Out[84]:

	purpose_cat	
0	другое	1
1	машина	6
2	недвижимость	2
3	свадьба	5

```
In [85]:
```

```
sql = '''
with
categories as (
    select t.purpose,
    case
        when t.purpose like '%свадьб%' then 'свадьба'
        when t.purpose like '%машин%' or t.purpose like '%авто%' then 'машина'
        when t.purpose like '%недвиж%' then 'недвижимость'
        else 'другое'
    end as purpose_cat
    from purpose t
)
select t.purpose_cat,
count(1)
from categories t
group by t.purpose_cat
'''
select(sql)
```

```
Out[85]:
```

	purpose_cat	
0	другое	1
1	машина	6
2	недвижимость	2
3	свадьба	5

3. Когда лучше создать таблицу, а не использовать подзапрос

In [86]:

```
cur = conn.cursor()
sql = '''
drop table if exists categories;

select t.purpose,

case when t.purpose like '%свадьб%' then 'свадьба'
when t.purpose like '%машин%' or t.purpose like '%авто%' then 'машина'
when t.purpose like '%недвиж%' then 'недвижимость'

else 'другое' end as purpose_cat

into categories

from purpose t
'''
cur.execute(sql)
conn.commit()
cur.close()
sql = '''select * from categories t'''
select(sql)
```

Out[86]:

	purpose	purpose_cat
0	машина	машина
1	машина	машина
2	машина	машина
3	на машину	машина
4	на покупку машины	машина
5	автомобиль	машина
6	на возвращение 2007	другое
7	на свадьбу	свадьба
8	свадьба	свадьба
9	свадьба	свадьба
10	свадьба	свадьба
11	для свадьбы	свадьба
12	недвижимость	недвижимость
13	на покупку недвижимости	недвижимость

In [87]:

```
sql = '''
select t.purpose_cat,
    count(1)
from categories t
group by t.purpose_cat
'''
select(sql)
```

Out[87]:

	purpose_cat	
0	другое	1
1	машина	6
2	недвижимость	2
3	свадьба	5

In [88]:

```
sql = '''
select t.purpose,
    count(1)
from categories t
where t.purpose_cat = 'другое'
group by t.purpose
order by count(1) desc
'''
select(sql)
```

Out[88]:

	purpose	
0	на возвращение 2007	1

берёт временную (with categories) а не categories в БД:

In [89]:

```
sql = '''
with
categories as (
    select 1 as p
    from purpose t
)
select * from categories t
'''
select(sql)
```

Out[89]:

```
    p
 0  1
 1  1
 2  1
 3  1
 4  1
 5  1
 6  1
 7  1
 8  1
 9  1
10  1
11  1
12  1
13  1
```

05-join

6. Джойны

In [90]:

```
users = pd.DataFrame({'id':[1,2,3],'name':['gleb','jon snow','tyrion']})
```

In [91]:

```
items = pd.DataFrame({'user_id':[1,3,3],'item_name':['hleb','gold','wine'],'value':[5,100,2
```

In [92]:

```
cur = conn.cursor()
sql = '''
drop table if exists users;
CREATE TABLE users (
    id        INTEGER,
    name      VARCHAR(max)
);
'''
cur.execute(sql)
conn.commit()
for index,row in users.iterrows():
    cur.execute('''INSERT INTO users(
                    [id],[name]
                    )
                    values (?,?)
    ''',
                    row['id'],
                    row['name']
               )
conn.commit()
cur.close()
sql = '''select t.* from users t'''
select(sql)
```

Out[92]:

	id	name
0	1	gleb
1	2	jon snow
2	3	tyrion

In [93]:

```
cur = conn.cursor()
sql = '''
drop table if exists items;
CREATE TABLE items (
    user_id         INTEGER,
    item_name       VARCHAR(max),
    value           MONEY
);
'''
cur.execute(sql)
conn.commit()
for index,row in items.iterrows():
    cur.execute('''INSERT INTO items(
                    [user_id],[item_name],[value]
                    )
                    values (?,?,?)
    ''',
                    row['user_id'],
                    row['item_name'],
                    row['value']
               )
conn.commit()
cur.close()
sql = '''select t.* from items t'''
select(sql)
```

Out[93]:

	user_id	item_name	value
0	1	hleb	5.0
1	3	gold	100.0
2	3	wine	20.0

2. Лефт и иннер джойн

In [94]:

```
sql = '''
select
    t.*, i.item_name, i.value, i.user_id
from users t
left join items i on t.id = i.user_id
'''
select(sql)
```

Out[94]:

	id	name	item_name	value	user_id
0	1	gleb	hleb	5.0	1.0
1	2	jon snow	None	NaN	NaN
2	3	tyrion	gold	100.0	3.0
3	3	tyrion	wine	20.0	3.0

In [95]:

```
sql = '''
select
    t.*, i.item_name, i.value, i.user_id
from users t
left join items i on t.id = i.user_id
where i.item_name is not null
'''
select(sql)
```

Out[95]:

	id	name	item_name	value	user_id
0	1	gleb	hleb	5.0	1
1	3	tyrion	gold	100.0	3
2	3	tyrion	wine	20.0	3

In [96]:

```
sql = '''
select
    t.*, i.item_name
from users t
join items i on t.id = i.user_id
'''
select(sql)
```

Out[96]:

	id	name	item_name
0	1	gleb	hleb
1	3	tyrion	gold
2	3	tyrion	wine

3. Агрегируй перед джойном!

In [97]:

```
users = pd.DataFrame({'id':[1,2,3],'name':['gleb','jon snow','tyrion'],
                      'victory':[2,10,1]})
```

In [98]:

```
cur = conn.cursor()
sql = '''
drop table if exists users;
CREATE TABLE users (
    id         INTEGER,
    name       VARCHAR(max),
    victory    INTEGER
);
'''
cur.execute(sql)
conn.commit()

for index,row in users.iterrows():
    cur.execute('''INSERT INTO users(
                    [id],[name],[victory]
                    )
                    values (?,?,?)
    ''',
                    row['id'],
                    row['name'],
                    row['victory']
               )

conn.commit()
cur.close()
sql = '''select t.* from users t'''
select(sql)
```

Out[98]:

	id	name	victory
0	1	gleb	2
1	2	jon snow	10
2	3	tyrion	1

In [99]:

```
sql = '''
select t.*,
    i.item_name, i.value, i.user_id
from users t
left join items i on t.id = i.user_id
'''
```

In [100]:

```
t = select(sql)
t
```

Out[100]:

	id	name	victory	item_name	value	user_id
0	1	gleb	2	hleb	5.0	1.0
1	2	jon snow	10	None	NaN	NaN
2	3	tyrion	1	gold	100.0	3.0
3	3	tyrion	1	wine	20.0	3.0

In [101]:

```
t['victory'].sum()
```

Out[101]:

```
14
```

In [102]:

```
sql = '''
select sum(t.victory)
from users t
'''
select(sql)
```

Out[102]:

0	13

После джойнов:

1. Проверяй контрольную сумму
2. Проверяй дубликаты

In [103]:

```
sql = '''
select
    t.*, i.item_name, i.value, i.user_id
from users t
join items i on t.id = i.user_id
'''
select(sql)
```

Out[103]:

	id	name	victory	item_name	value	user_id
0	1	gleb	2	hleb	5.0	1
1	3	tyrion	1	gold	100.0	3
2	3	tyrion	1	wine	20.0	3

Как правильно:

In [104]:

```
sql = '''
select t.id, t.name, t.victory,
    count(i.item_name) as item_cnt,
    coalesce(sum(i.value),0) as value_sum
from users t
left join items i on t.id = i.user_id
group by t.id, t.name, t.victory
'''
select(sql)
```

Out[104]:

	id	name	victory	item_cnt	value_sum
0	1	gleb	2	1	5.0
1	2	jon snow	10	0	0.0
2	3	tyrion	1	2	120.0

Надо перед джойном сгруппировать items:

In [105]:

```
sql = '''
select t.user_id,
    count(t.item_name) as item_cnt,
    sum(value) as value_sum from items t
group by t.user_id
'''
select(sql)
```

Out[105]:

	user_id	item_cnt	value_sum
0	1	1	5.0
1	3	2	120.0

In [106]:

```
sql = '''
with
items_agg as (
    select t.user_id,
        count(t.item_name) as item_cnt,
        sum(value) as value_sum
    from items t
    group by t.user_id
)
select t.id, t.name, t.victory,
    coalesce(i.item_cnt,0) as item_cnt,
    coalesce(i.value_sum,0) as value_sum
from users t
left join items_agg i on t.id = i.user_id
'''
select(sql)
```

Out[106]:

	id	name	victory	item_cnt	value_sum
0	1	gleb	2	1	5.0
1	2	jon snow	10	0	0.0
2	3	tyrion	1	2	120.0

4. Как не надо делать джойны

всегда надо писать псевдонимы:

In [107]:

```
sql = '''
with
items_agg as (
    select t.user_id,
        count(t.item_name) as item_cnt,
        sum(value) as value_sum
    from items t
    group by t.user_id
)
select t.id, t.name, t.victory,
    coalesce(item_cnt,0) as item_cnt,
    coalesce(value_sum,0) as value_sum
from users t
left join items_agg i on t.id = i.user_id
'''
select(sql)
```

Out[107]:

	id	name	victory	item_cnt	value_sum
0	1	gleb	2	1	5.0
1	2	jon snow	10	0	0.0
2	3	tyrion	1	2	120.0

5. Никогда не используй right join!

In [108]:

```
users = pd.DataFrame({'id':[1,2,3],'name':['gleb','jon snow','tyrion']})
```

In [109]:

```
items = pd.DataFrame({'user_id':[1,3,3,4],'item_name':['hleb','gold','wine','sword'],'value
```

In [110]:

```
cur = conn.cursor()
sql = '''
drop table if exists users;
CREATE TABLE users (
    id         INTEGER,
    name       VARCHAR(max)
);
'''
cur.execute(sql)
conn.commit()
for index,row in users.iterrows():
    cur.execute('''INSERT INTO users(
                    [id],[name]
                    )
                    values (?,?)
    ''',
                    row['id'],
                    row['name']
               )
conn.commit()
cur.close()
sql = '''select t.* from users t'''
select(sql)
```

Out[110]:

	id	name
0	1	gleb
1	2	jon snow
2	3	tyrion

In [111]:

```
cur = conn.cursor()
sql = '''
drop table if exists items;
CREATE TABLE items (
    user_id         INTEGER,
    item_name       VARCHAR(max),
    value           MONEY
);
'''
cur.execute(sql)
conn.commit()
for index,row in items.iterrows():
    cur.execute('''INSERT INTO items(
                    [user_id],[item_name],[value]
                    )
                    values (?,?,?)
    ''',
                    row['user_id'],
                    row['item_name'],
                    row['value']
               )
conn.commit()
cur.close()
sql = '''select t.* from items t'''
select(sql)
```

Out[111]:

	user_id	item_name	value
0	1	hleb	5.0
1	3	gold	100.0
2	3	wine	20.0
3	4	sword	50.0

In [112]:

```
sql = '''
select t.*, i.*
from users t
left join items i on t.id = i.user_id
'''
select(sql)
```

Out[112]:

	id	name	user_id	item_name	value
0	1	gleb	1.0	hleb	5.0
1	2	jon snow	NaN	None	NaN
2	3	tyrion	3.0	gold	100.0
3	3	tyrion	3.0	wine	20.0

In [113]:

```
sql = '''
select t.*, u.*
from items t
left join users u on t.user_id = u.id
'''
select(sql)
```

Out[113]:

	user_id	item_name	value	id	name
0	1	hleb	5.0	1.0	gleb
1	3	gold	100.0	3.0	tyrion
2	3	wine	20.0	3.0	tyrion
3	4	sword	50.0	NaN	None

In [114]:

```
sql = '''
select t.*, i.*
from users t
right join items i on t.id = i.user_id
'''
select(sql)
```

Out[114]:

	id	name	user_id	item_name	value
0	1.0	gleb	1	hleb	5.0
1	3.0	tyrion	3	gold	100.0
2	3.0	tyrion	3	wine	20.0
3	NaN	None	4	sword	50.0

6. Full join

In [115]:

```
sql = '''
select t.*, i.*
from users t
full join items i on t.id = i.user_id
'''
select(sql)
```

Out[115]:

	id	name	user_id	item_name	value
0	1.0	gleb	1.0	hleb	5.0
1	2.0	jon snow	NaN	None	NaN
2	3.0	tyrion	3.0	gold	100.0
3	3.0	tyrion	3.0	wine	20.0
4	NaN	None	4.0	sword	50.0

Если вдруг не можешь вспомнить как делать full join (да и вообще что либо) -- всегда гугли.
[sql - FULL OUTER JOIN with SQLite - Stack Overflow (https://stackoverflow.com/questions/1923259/full-outer-join-with-sqlite)](https://stackoverflow.com/questions/1923259/full-outer-join-with-sqlite)

имитация full join:

In [116]:

```
sql = '''
select t.*, i.*
from users t
left join items i on t.id = i.user_id
union
select u.*, t.*
from items t
left join users u on t.user_id = u.id
'''
select(sql)
```

Out[116]:

	id	name	user_id	item_name	value
0	NaN	None	4.0	sword	50.0
1	1.0	gleb	1.0	hleb	5.0
2	2.0	jon snow	NaN	None	NaN
3	3.0	tyrion	3.0	gold	100.0
4	3.0	tyrion	3.0	wine	20.0

7. Фишки с inner join

сопоставление с "присланным" файлом:

In [117]:

```
sql = '''
select top(5) *
from german_credit t
'''
select(sql)
```

Out[117]:

	age	sex	job	housing	saving_accounts	checking_account	credit_amount	duration	pu
0	33	male	2	own	None	None	3074	9	ra
1	43	male	1	own	little	little	1344	12	
2	52	male	2	own	quite rich	None	936	9	edu
3	35	female	3	own	little	None	1393	11	
4	28	male	2	own	little	None	776	12	ra

In [118]:

```
clients = pd.DataFrame({'client_id':[200,45],'data':[1.0, 2.0]})
```

In [119]:

```
cur = conn.cursor()
sql = '''
drop table if exists clients_task_name;
CREATE TABLE clients_task_name (
    client_id        int,
    data             int
);
'''
cur.execute(sql)
conn.commit()
for index,row in clients.iterrows():
    cur.execute('''INSERT INTO clients_task_name(
                    [client_id],[data]
                    )
                    values (?,?)
    ''',
                    row['client_id'],
                    row['data']
               )
conn.commit()
cur.close()
sql = '''select t.* from clients_task_name t'''
select(sql)
```

Out[119]:

	client_id	data
0	200	1
1	45	2

In [120]:

```
sql = '''
select t.*, ctn.data
from german_credit t
join clients_task_name ctn on t.client_id = ctn.client_id
'''
select(sql)
```

Out[120]:

	age	sex	job	housing	saving_accounts	checking_account	credit_amount	duration	pu
0	52	male	2	own	quite rich	None	936	9	edu
1	35	female	3	own	little	None	1393	11	

генерация заготовок под отчёт:

In [121]:

```
sql = '''
select 1 as user_id
union all
select 2 as user_id
union all
select 3 as user_id
'''
select(sql)
```

Out[121]:

	user_id
0	1
1	2
2	3

In [122]:

```
sql = '''
select convert(date, '01.03.2021', 104) as month
union all
select convert(date, '01.04.2021', 104) as month
'''
select(sql)
```

Out[122]:

	month
0	2021-03-01
1	2021-04-01

In [123]:

```
sql = '''
with
users as (
    select 1 as user_id
    union all
    select 2 as user_id
    union all
    select 3 as user_id
),
month as (
    select convert(date, '01.03.2021', 104) as month
    union all
    select convert(date, '01.04.2021', 104) as month
)
select * from users t
join month m on 1=1
'''
select(sql)
```

Out[123]:

	user_id	month
0	1	2021-03-01
1	2	2021-03-01
2	3	2021-03-01
3	1	2021-04-01
4	2	2021-04-01
5	3	2021-04-01

06-join-practical-examples

In [124]:

```
sql = '''select * from client_transactions t'''
select(sql)
```

Out[124]:

	dt	client_id	amount
0	2008-04-06 11:54:47	950	161.38
1	2007-07-28 00:00:19	418	35.34
2	2008-03-14 20:43:54	131	146.50
3	2007-12-18 13:03:24	353	119.21
4	2007-11-09 05:18:30	849	105.24
...	...	...	...
4270	2007-08-18 04:05:05	185	10063.07
4271	2007-06-04 15:23:32	375	156.91
4272	2007-12-06 21:34:06	418	10053.82
4273	2008-04-19 17:30:07	409	10050.35
4274	2007-11-07 19:44:50	674	165.60

4275 rows × 3 columns

6. Джойны (Продолжение)

8. Ежемесячный отчет (практический пример)

прислали транзакции по клиентам:

```
transactions = pd.read_csv('../data/german_credit_augmented_transactions.csv')
```

In [125]:

```
sql = '''select top(5) * from client_transactions t'''
select(sql)
```

Out[125]:

	dt	client_id	amount
0	2008-04-06 11:54:47	950	161.38
1	2007-07-28 00:00:19	418	35.34
2	2008-03-14 20:43:54	131	146.50
3	2007-12-18 13:03:24	353	119.21
4	2007-11-09 05:18:30	849	105.24

In [126]:

```
sql = '''select count(*) from client_transactions t'''
select(sql)
```

Out[126]:

0	4275

сгруппируем:

In [127]:

```
sql = '''
select year(t.dt) as year,  month(t.dt) as month,
    count(1) as transaction_cnt,
    sum(t.amount) as amount_sum
from client_transactions t
group by year(t.dt), month(t.dt)
order by year(t.dt), month(t.dt)
'''
select(sql)
```

Out[127]:

	year	month	transaction_cnt	amount_sum
0	2007	5	338	450912.77
1	2007	6	379	551664.83
2	2007	7	304	494134.50
3	2007	8	255	426903.23
4	2007	10	332	634846.49
5	2007	11	389	500420.98
6	2007	12	364	561449.89
7	2008	1	413	630137.22
8	2008	2	228	337043.47
9	2008	3	309	425599.09
10	2008	4	383	677194.97
11	2008	5	310	474962.34
12	2008	6	271	383710.84

нет сентября...

надо сгенерить заготовку, чтобы были все месяцы:

[Как создать диапазон дат в SQL Server (https://stackovergo.com/ru/q/3063246/how-to-generate-a-range-of-dates-in-sql-server)](https://stackovergo.com/ru/q/3063246/how-to-generate-a-range-of-dates-in-sql-server)

In [128]:

```
sql = '''
Declare   @FromDate   Date,
          @ToDate     Date
select @FromDate = min(t.dt) from client_transactions t
select @ToDate = max(t.dt) from client_transactions t;

WITH n AS
(
  SELECT TOP (DATEDIFF(DAY, @FromDate, @ToDate) + 1)
    n = ROW_NUMBER() OVER (ORDER BY [object_id])
  FROM sys.all_objects
),
p as
(
SELECT DATEADD(DAY, n-1, @FromDate) as dt
FROM n
)
select year(dt) year, month(dt) month from p
'''
select(sql)
```

Out[128]:

	year	month
0	2007	5
1	2007	5
2	2007	5
3	2007	5
4	2007	5
...	...	...
421	2008	6
422	2008	6
423	2008	6
424	2008	6
425	2008	6

426 rows × 2 columns

In [129]:

```
sql = '''
Declare   @FromDate   Date,
          @ToDate     Date
select @FromDate = min(t.dt) from client_transactions t
select @ToDate = max(t.dt) from client_transactions t;

WITH n AS
(
  SELECT TOP (DATEDIFF(DAY, @FromDate, @ToDate) + 1)
    n = ROW_NUMBER() OVER (ORDER BY [object_id])
  FROM sys.all_objects
),
p as
(
SELECT DATEADD(DAY, n-1, @FromDate) as dt
FROM n
),
ym as(
select year(dt) year, month(dt) month from p
group by year(dt), month(dt)
),
tr as(
select
year(t.dt) as year,  month(t.dt) as month,
count(1) as transaction_cnt,
sum(t.amount) as amount_sum
from client_transactions t
group by year(t.dt), month(t.dt)
--order by year(t.dt), month(t.dt)
)

--select * from ym

--/*
select ym.year, ym.month,
coalesce(tr.transaction_cnt,0) as transaction_cnt,
coalesce(tr.amount_sum,0) as amount_sum
from ym
left join tr on tr.year = ym.year and tr.month = ym.month
--*/
'''
select(sql)
```

Out[129]:

	year	month	transaction_cnt	amount_sum
0	2007	5	338	450912.77
1	2007	6	379	551664.83
2	2007	7	304	494134.50
3	2007	8	255	426903.23
4	2007	9	0	0.00
5	2007	10	332	634846.49
6	2007	11	389	500420.98
7	2007	12	364	561449.89

	year	month	transaction_cnt	amount_sum
8	2008	1	413	630137.22
9	2008	2	228	337043.47
10	2008	3	309	425599.09
11	2008	4	383	677194.97
12	2008	5	310	474962.34
13	2008	6	271	383710.84

9. Ежемесячный отчет на пользователя (практический пример)

In [130]:

```
sql = '''select * from german_credit t'''
select(sql)
```

Out[130]:

	age	sex	job	housing	saving_accounts	checking_account	credit_amount	duration
0	33	male	2	own	None	None	3074	9
1	43	male	1	own	little	little	1344	12
2	52	male	2	own	quite rich	None	936	9
3	35	female	3	own	little	None	1393	11
4	28	male	2	own	little	None	776	12
...	...	...	...	...	...	...	...	...
995	65	male	2	free	little	little	2600	18
996	30	male	3	own	little	moderate	4455	36
997	33	male	2	own	little	moderate	6403	24
998	29	female	2	own	None	None	5003	21
999	44	male	2	own	moderate	moderate	1804	12

1000 rows × 12 columns

In [131]:

```
sql = '''
select distinct t.client_id
from german_credit t
'''
select(sql)
```

Out[131]:

	client_id
0	0
1	1
2	2
3	3
4	4
...	...
995	995
996	996
997	997
998	998
999	999

1000 rows × 1 columns

In [132]:

```
sql = '''
Declare   @FromDate   Date,
          @ToDate     Date
select @FromDate = min(t.dt) from client_transactions t
select @ToDate = max(t.dt) from client_transactions t;

WITH n AS
(
  SELECT TOP (DATEDIFF(DAY, @FromDate, @ToDate) + 1)
    n = ROW_NUMBER() OVER (ORDER BY [object_id])
  FROM sys.all_objects
),
p as
(
SELECT DATEADD(DAY, n-1, @FromDate) as dt
FROM n
),

--список дат
dates as(
select year(dt) year, month(dt) month from p
group by year(dt), month(dt)
),

--клиенты
clients as (
select distinct t.client_id from german_credit t
),

--привязка каждого клиента к дате
clients_month as
(SELECT t.year, t.month, c.client_id FROM dates t
join clients c on 1=1),

--реестр транзакций (из файла)
trans_month as(
select
year(t.dt) as year,  month(t.dt) as month,
t.client_id,
count(1) as transaction_cnt,
sum(t.amount) as amount_sum
from client_transactions t
group by year(t.dt), month(t.dt), t.client_id
)

--/*
,client_trans_month as (

select t.client_id, t.year, t.month,
tm.transaction_cnt,
tm.amount_sum,
1 as [user],
case when tm.transaction_cnt > 0 then 1 else 0 end as active
from clients_month t
left join trans_month tm on t.client_id = tm.client_id
    and t.year = tm.year and t.month = tm.month
)
--*/
```

```
/*
select * from client_trans_month
where client_id=900
order by client_id, year, month
--*/
--/*
select t.year, t.month, sum(t.[user]) as user_cnt, sum(t.amount_sum) as amount_sum ,
sum(t.active) as active_cnt from client_trans_month t
group by t.year, t.month
order by t.year, t.month
--*/
'''
select(sql)
```

Out[132]:

	year	month	user_cnt	amount_sum	active_cnt
0	2007	5	1000	450912.77	288
1	2007	6	1000	551664.83	297
2	2007	7	1000	494134.50	259
3	2007	8	1000	426903.23	222
4	2007	9	1000	NaN	0
5	2007	10	1000	634846.49	283
6	2007	11	1000	500420.98	323
7	2007	12	1000	561449.89	287
8	2008	1	1000	630137.22	325
9	2008	2	1000	337043.47	204
10	2008	3	1000	425599.09	267
11	2008	4	1000	677194.97	301
12	2008	5	1000	474962.34	263
13	2008	6	1000	383710.84	237

проверим:

In [133]:

```
t = select(sql)
```

In [134]:

```
t['amount_sum'].sum()
```

Out[134]:

```
6548980.619999999
```

In [135]:

```
sql = '''
select sum(t.amount)
from client_transactions t
'''
select(sql)
```

Out[135]:

```
0   6548980.62
```

11. Джойн таблицы самой на себя (нарастающий итог)

In [136]:

```
t = pd.DataFrame({'dt':pd.to_datetime(['2021-04-01','2021-04-02','2021-04-03'],format='%Y-%
                  'revenue':[1,2,3]})
```

In [137]:

```
cur = conn.cursor()
sql = '''
drop table if exists revenue;
CREATE TABLE revenue (
    dt         datetime,
    revenue    int
);
'''
cur.execute(sql)
conn.commit()
for index,row in t.iterrows():
    cur.execute('''INSERT INTO revenue(
                    [dt],[revenue]
                    )
                    values (?,?)
    ''',
                    row['dt'],
                    row['revenue']
               )
conn.commit()
cur.close()
sql = '''select * from revenue t'''
select(sql)
```

Out[137]:

```
           dt  revenue
0  2021-04-01        1
1  2021-04-02        2
2  2021-04-03        3
```

In [138]:

```
sql = '''
select t.dt,t.revenue,
    sum(r.revenue) as cumsum
from revenue t
join revenue r on r.dt <= t.dt
group by t.dt, t.revenue
'''
select(sql)
```

Out[138]:

	dt	revenue	cumsum
0	2021-04-01	1	1
1	2021-04-02	2	3
2	2021-04-03	3	6

07-over

7. Оконные функции

1. Что такое оконная функция

Нарастающий итог:

In [139]:

```
sql = '''
select t.*,
    sum(t.revenue) over (order by t.dt) as cum_sum
from revenue t
'''
select(sql)
```

Out[139]:

	dt	revenue	cum_sum
0	2021-04-01	1	1
1	2021-04-02	2	3
2	2021-04-03	3	6

In [140]:

```
t = pd.DataFrame({'user_id':[1,1,1,2,2,2],'dt':pd.to_datetime(['2021-04-01','2021-04-02','2
                                                                 '2021-04-01','2021-04-02','2
                  'revenue':[1,2,3,2,3,4]})
```

In [141]:

```
cur = conn.cursor()
sql = '''
drop table if exists revenue;
CREATE TABLE revenue (
    user_id   int,
    dt        datetime,
    revenue   int
);
'''
cur.execute(sql)
conn.commit()
for index,row in t.iterrows():
    cur.execute('''INSERT INTO revenue(
                    [user_id],[dt],[revenue]
                    )
                    values (?,?,?)
    ''',
                    row['user_id'],
                    row['dt'],
                    row['revenue']
               )
conn.commit()
cur.close()
sql = '''select * from revenue t'''
select(sql)
```

Out[141]:

	user_id	dt	revenue
0	1	2021-04-01	1
1	1	2021-04-02	2
2	1	2021-04-03	3
3	2	2021-04-01	2
4	2	2021-04-02	3
5	2	2021-04-03	4

In [142]:

```
sql = '''
select t.*,
    sum(t.revenue) over (partition by t.user_id order by t.dt) as cum_sum
from revenue t
'''
select(sql)
```

Out[142]:

	user_id	dt	revenue	cum_sum
0	1	2021-04-01	1	1
1	1	2021-04-02	2	3
2	1	2021-04-03	3	6
3	2	2021-04-01	2	2
4	2	2021-04-02	3	5
5	2	2021-04-03	4	9

2. rank и row_number

In [143]:

```
t = pd.DataFrame({'user_id':[1,1,1,1,2,2,2],'dt':pd.to_datetime(['2021-04-01','2021-04-02',
                                                               '2021-04-03','2021-04-04','2
                  'revenue':[1,2,3,1,2,3,4]})
```

In [144]:

```
cur = conn.cursor()
sql = '''
drop table if exists revenue;
CREATE TABLE revenue (
    user_id   int,
    dt        datetime,
    revenue   int
);
'''
cur.execute(sql)
conn.commit()
for index,row in t.iterrows():
    cur.execute('''INSERT INTO revenue(
                    [user_id],[dt],[revenue]
                    )
                    values (?,?,?)
    ''',
                    row['user_id'],
                    row['dt'],
                    row['revenue']
               )
conn.commit()
cur.close()
sql = '''select * from revenue t'''
select(sql)
```

Out[144]:

	user_id	dt	revenue
0	1	2021-04-01	1
1	1	2021-04-02	2
2	1	2021-04-03	3
3	1	2021-04-03	1
4	2	2021-04-03	2
5	2	2021-04-04	3
6	2	2021-04-05	4

последняя дата активности каждого пользователя:

rank():

In [145]:

```
sql = '''
select t.*,
    rank() over (partition by t.user_id order by t.dt desc) as rnk
from revenue t
'''
select(sql)
```

Out[145]:

	user_id	dt	revenue	rnk
0	1	2021-04-03	3	1
1	1	2021-04-03	1	1
2	1	2021-04-02	2	3
3	1	2021-04-01	1	4
4	2	2021-04-05	4	1
5	2	2021-04-04	3	2
6	2	2021-04-03	2	3

In [146]:

```
sql = '''
with
dt_rank as (
    select t.*,
        rank() over (partition by t.user_id order by t.dt desc) as rnk
    from revenue t
)
select * from dt_rank t
where t.rnk = 1
'''
select(sql)
```

Out[146]:

	user_id	dt	revenue	rnk
0	1	2021-04-03	3	1
1	1	2021-04-03	1	1
2	2	2021-04-05	4	1

row_number():

In [147]:

```
sql = '''
select t.*,
    row_number() over (partition by t.user_id order by t.dt desc) as rnk
from revenue t
'''
select(sql)
```

Out[147]:

	user_id	dt	revenue	rnk
0	1	2021-04-03	3	1
1	1	2021-04-03	1	2
2	1	2021-04-02	2	3
3	1	2021-04-01	1	4
4	2	2021-04-05	4	1
5	2	2021-04-04	3	2
6	2	2021-04-03	2	3

In [148]:

```
sql = '''
with
dt_rank as (
    select t.*,
        row_number() over (partition by t.user_id order by t.dt desc) as rnk
    from revenue t
)
select * from dt_rank t
where t.rnk = 1
'''
select(sql)
```

Out[148]:

	user_id	dt	revenue	rnk
0	1	2021-04-03	3	1
1	2	2021-04-05	4	1

стандартным способом:

In [149]:

```
t = pd.DataFrame({'user_id':[1,1,1,2,2,2],'dt':pd.to_datetime(['2021-04-01','2021-04-02','2
                                                                 '2021-04-03','2021-04-04','2
                  'revenue':[1,2,3,2,3,4]})
```

In [150]:

```
cur = conn.cursor()
sql = '''
drop table if exists revenue;
CREATE TABLE revenue (
    user_id   int,
    dt        datetime,
    revenue   int
);
'''
cur.execute(sql)
conn.commit()
for index,row in t.iterrows():
    cur.execute('''INSERT INTO revenue(
                    [user_id],[dt],[revenue]
                    )
                    values (?,?,?)
    ''',
                    row['user_id'],
                    row['dt'],
                    row['revenue']
                )
conn.commit()
cur.close()
sql = '''select * from revenue t'''
select(sql)
```

Out[150]:

	user_id	dt	revenue
0	1	2021-04-01	1
1	1	2021-04-02	2
2	1	2021-04-03	3
3	2	2021-04-03	2
4	2	2021-04-04	3
5	2	2021-04-05	4

In [151]:

```
sql = '''
select t.user_id,
    max(t.dt) as max_dt
from revenue t
group by t.user_id
'''
select(sql)
```

Out[151]:

	user_id	max_dt
0	1	2021-04-03
1	2	2021-04-05

In [152]:

```
sql = '''
with
last_dt as (
    select t.user_id,
        max(t.dt) as max_dt
    from revenue t
    group by t.user_id
)
select t.* from revenue t
join last_dt ld on t.user_id = ld.user_id and t.dt = ld.max_dt
order by t.user_id
'''
select(sql)
```

Out[152]:

	user_id	dt	revenue
0	1	2021-04-03	3
1	2	2021-04-05	4

3. Топ 3 зарплаты в отделе (задача на интервью)

In [153]:

```
t = pd.DataFrame({'dep':['a','a','a','a','a',
                         'b','b','b','b','b'],
                  'emp':['aa','bb','cc','dd','ee',
                         'aa','bb','cc','dd','ee'],
                  'sal':[5,5,3,2,1,
                         5,4,3,2,1]})
```

In [154]:

```
cur = conn.cursor()
sql = '''
drop table if exists salary;
CREATE TABLE salary (
    dep       varchar(max),
    emp       varchar(max),
    sal       int
);
'''
cur.execute(sql)
conn.commit()
for index,row in t.iterrows():
    cur.execute('''INSERT INTO salary(
                    [dep],[emp],[sal]
                    )
                    values (?,?,?)
    ''',
                    row['dep'],
                    row['emp'],
                    row['sal']
               )
conn.commit()
cur.close()
sql = '''select * from salary t'''
select(sql)
```

Out[154]:

	dep	emp	sal
0	a	aa	5
1	a	bb	5
2	a	cc	3
3	a	dd	2
4	a	ee	1
5	b	aa	5
6	b	bb	4
7	b	cc	3
8	b	dd	2
9	b	ee	1

In [155]:

```
sql = '''
select t.*,
    rank() over (partition by t.dep order by t.sal desc) as rnk_rank,
    dense_rank() over (partition by t.dep order by t.sal desc) as rnk
from salary t
'''
select(sql)
```

Out[155]:

	dep	emp	sal	rnk_rank	rnk
0	a	aa	5	1	1
1	a	bb	5	1	1
2	a	cc	3	3	2
3	a	dd	2	4	3
4	a	ee	1	5	4
5	b	aa	5	1	1
6	b	bb	4	2	2
7	b	cc	3	3	3
8	b	dd	2	4	4
9	b	ee	1	5	5

In [156]:

```
sql = '''
with
salary_rnk as (
    select t.*,
        dense_rank() over (partition by t.dep order by t.sal desc) as rnk
    from salary t
)
select * from salary_rnk t
where t.rnk <= 3
'''
select(sql)
```

Out[156]:

	dep	emp	sal	rnk
0	a	aa	5	1
1	a	bb	5	1
2	a	cc	3	2
3	a	dd	2	3
4	b	aa	5	1
5	b	bb	4	2
6	b	cc	3	3

4. Расчет сессий клиентов (задача из тестового)

действия клиентов по времени:

In [157]:

```
user1 = pd.DataFrame({'user_id':[1,1,1,1,1],
                  'dt':pd.to_datetime(['2021-04-01 07:31','2021-04-01 07:35',
                                       '2021-04-01 08:20','2021-04-01 12:31',
                                       '2021-04-03 07:31'],format='%Y-%m-%d %H:%M')})
```

In [158]:

```
user2 = pd.DataFrame({'user_id':[2,2,2,2],
                  'dt':pd.to_datetime(['2021-04-01 07:31','2021-04-01 07:35',
                                       '2021-04-01 08:20','2021-04-01 9:10',
                                      ],format='%Y-%m-%d %H:%M')})
```

In [159]:

```
user3 = pd.DataFrame({'user_id':[3,3,3],
                  'dt':pd.to_datetime(['2021-04-01 07:31','2021-04-02 07:35',
                                       '2021-04-03 08:20'
                                      ],format='%Y-%m-%d %H:%M')})
```

In [160]:

```
t = pd.concat([user1,user2,user3])
# t
```

In [161]:

```
cur = conn.cursor()
sql = '''
drop table if exists client_log;
CREATE TABLE client_log (
    user_id   int,
    dt        datetime
);
'''
cur.execute(sql)
conn.commit()
for index,row in t.iterrows():
    cur.execute('''INSERT INTO client_log(
                    [user_id],[dt]
                    )
                    values (?,?)
    ''',
                    row['user_id'],
                    row['dt']
               )
conn.commit()
cur.close()
sql = '''select * from client_log t'''
select(sql)
```

Out[161]:

	user_id	dt
0	1	2021-04-01 07:31:00
1	1	2021-04-01 07:35:00
2	1	2021-04-01 08:20:00
3	1	2021-04-01 12:31:00
4	1	2021-04-03 07:31:00
5	2	2021-04-01 07:31:00
6	2	2021-04-01 07:35:00
7	2	2021-04-01 08:20:00
8	2	2021-04-01 09:10:00
9	3	2021-04-01 07:31:00
10	3	2021-04-02 07:35:00
11	3	2021-04-03 08:20:00

Надо посчитать количество сессий клиентов:

Одна сессия, если между действиями проходит меньше часа. Надо посчитать количество сессий клиетов.
(для 1 клиента 2-я сессия начинается в 12:31... = 3 сессии
2: 1 сессия, 3: 2 сессии)

На каждое действие показать предыдущее действие:

lag():

In [162]:

```
sql = '''
select *,
    lag(t.dt) over (partition by t.user_id order by t.dt) as prev_dt
from client_log t
'''
select(sql)
```

Out[162]:

	user_id	dt	prev_dt
0	1	2021-04-01 07:31:00	NaT
1	1	2021-04-01 07:35:00	2021-04-01 07:31:00
2	1	2021-04-01 08:20:00	2021-04-01 07:35:00
3	1	2021-04-01 12:31:00	2021-04-01 08:20:00
4	1	2021-04-03 07:31:00	2021-04-01 12:31:00
5	2	2021-04-01 07:31:00	NaT
6	2	2021-04-01 07:35:00	2021-04-01 07:31:00
7	2	2021-04-01 08:20:00	2021-04-01 07:35:00
8	2	2021-04-01 09:10:00	2021-04-01 08:20:00
9	3	2021-04-01 07:31:00	NaT
10	3	2021-04-02 07:35:00	2021-04-01 07:31:00
11	3	2021-04-03 08:20:00	2021-04-02 07:35:00

Сколько времени прошло между текущей активностью и предыдущей:

```
# Поскольку мы там видели, что операция расчета разноси двух дат в секундах может давать
странные результаты
# с числами после запятой, то чтобы себя обезопасить и успокоить, можно округлить эту
разницу до целых. Вот так:
#  case when round((julianday(t.dt) - julianday(lag(t.dt) over (partition by t.user_id
order by t.dt))) * 24 * 60 * 60) >= 3600
# then 1 else 0 end as new_session

sql = '''
select *,
lag(t.dt) over (partition by t.user_id order by t.dt) as prev_dt,
round((julianday(t.dt) - julianday(lag(t.dt) over (partition by t.user_id order by t.dt)))
* 24 * 60 * 60) as dt_diff
from client_log t
'''
```

[DATEDIFF (Transact-SQL) - SQL Server | Microsoft Learn (https://learn.microsoft.com/ru-RU/sql/t-sql/functions/datediff-transact-sql?view=sql-server-ver15&viewFallbackFrom=sqlallproducts-allversions)](https://learn.microsoft.com/ru-RU/sql/t-sql/functions/datediff-transact-sql?view=sql-server-ver15&viewFallbackFrom=sqlallproducts-allversions)

```
In [163]:
```

```
sql = '''
SELECT DATEDIFF(second, '2021-04-01 07:31:00.0000000', '2021-04-01 07:35:00.0000000');
'''
select(sql)
```

```
Out[163]:
```

0	240

```
In [164]:
```

```
sql = '''
select *,
    lag(t.dt) over (partition by t.user_id order by t.dt) as prev_dt,
    DATEDIFF(second, lag(t.dt) over (partition by t.user_id order by t.dt), t.dt) as dt_dif
from client_log t
'''
select(sql)
```

```
Out[164]:
```

	user_id	dt	prev_dt	dt_diff
0	1	2021-04-01 07:31:00	NaT	NaN
1	1	2021-04-01 07:35:00	2021-04-01 07:31:00	240.0
2	1	2021-04-01 08:20:00	2021-04-01 07:35:00	2700.0
3	1	2021-04-01 12:31:00	2021-04-01 08:20:00	15060.0
4	1	2021-04-03 07:31:00	2021-04-01 12:31:00	154800.0
5	2	2021-04-01 07:31:00	NaT	NaN
6	2	2021-04-01 07:35:00	2021-04-01 07:31:00	240.0
7	2	2021-04-01 08:20:00	2021-04-01 07:35:00	2700.0
8	2	2021-04-01 09:10:00	2021-04-01 08:20:00	3000.0
9	3	2021-04-01 07:31:00	NaT	NaN
10	3	2021-04-02 07:35:00	2021-04-01 07:31:00	86640.0
11	3	2021-04-03 08:20:00	2021-04-02 07:35:00	89100.0

Работаем с сессиями (номер сессии, начиная с 0):

In [165]:

```
sql = '''
with
new_session as (
    select *,
        --lag(t.dt) over (partition by t.user_id order by t.dt) as prev_dt,
        --DATEDIFF(second, lag(t.dt) over (partition by t.user_id order by t.dt), t.dt) as
        --условия сессий:
        case
            when DATEDIFF(second, lag(t.dt) over (partition by t.user_id order by t.dt), t.
            else 0
        end as new_session
    from client_log t
)
--select * from new_session t
--/*
select t.*,
--нарастающий итог (номер сессии, начиная с 0):
sum(t.new_session) over (partition by t.user_id order by t.dt) as session_id
from new_session t
--*/
'''
select(sql)
```

Out[165]:

	user_id	dt	new_session	session_id
0	1	2021-04-01 07:31:00	0	0
1	1	2021-04-01 07:35:00	0	0
2	1	2021-04-01 08:20:00	0	0
3	1	2021-04-01 12:31:00	1	1
4	1	2021-04-03 07:31:00	1	2
5	2	2021-04-01 07:31:00	0	0
6	2	2021-04-01 07:35:00	0	0
7	2	2021-04-01 08:20:00	0	0
8	2	2021-04-01 09:10:00	0	0
9	3	2021-04-01 07:31:00	0	0
10	3	2021-04-02 07:35:00	1	1
11	3	2021-04-03 08:20:00	1	2

кол-во активностей в каждой сессии:

In [166]:

```
sql = '''
with
new_session as (
    select *,
        case
            when DATEDIFF(second, lag(t.dt) over (partition by t.user_id order by t.dt), t.
            else 0
        end as new_session
    from client_log t
),
client_sessions as (
    select t.*,
    sum(t.new_session) over (partition by t.user_id order by t.dt) as session_id
    from new_session t
)
--select * from client_sessions t
--/*
select t.user_id, t.session_id, count(1) as action_cnt from client_sessions t
group by t.user_id, t.session_id
order by t.user_id, t.session_id
--*/
'''
select(sql)
```

Out[166]:

	user_id	session_id	action_cnt
0	1	0	3
1	1	1	1
2	1	2	1
3	2	0	4
4	3	0	1
5	3	1	1
6	3	2	1

всего количество сессий:

In [167]:

```
sql = '''
with
new_session as (
    select *,
        case
            when DATEDIFF(second, lag(t.dt) over (partition by t.user_id order by t.dt), t.
            else 0
        end as new_session
    from client_log t
),
client_sessions as (
    select t.*,
    sum(t.new_session) over (partition by t.user_id order by t.dt) as session_id
    from new_session t
),
client_sessions_agg as (
    select t.user_id, t.session_id,
    count(1) as action_cnt
    from client_sessions t
    group by t.user_id, t.session_id
)
--select * from client_sessions_agg t order by t.user_id, t.session_id
--/*
select count(*) from client_sessions_agg t
--*/
'''
select(sql)
```

Out[167]:

```
0  7
```

6. Скользящее среднее

In [168]:

```
t = pd.DataFrame({'user_id':[1,1,1,1,1,1,
                             2,2,2,2,2],
                  'dt':[1,2,3,4,5,6,
                        1,2,3,4,5],
                  'revenue':[1.0,2,3,4,5,6,
                             3,4,5,6,7]})
```

In [169]:

```
cur = conn.cursor()
sql = '''
drop table if exists revenue;
CREATE TABLE revenue (
    user_id   int,
    dt        int,
    revenue   int
);
'''
cur.execute(sql)
conn.commit()
for index,row in t.iterrows():
    cur.execute('''INSERT INTO revenue(
                    [user_id],[dt],[revenue]
                    )
                    values (?,?,?)
    ''',
                row['user_id'],
                row['dt'],
                row['revenue']
               )
conn.commit()
cur.close()
sql = '''select * from revenue t'''
select(sql)
```

Out[169]:

	user_id	dt	revenue
0	1	1	1
1	1	2	2
2	1	3	3
3	1	4	4
4	1	5	5
5	1	6	6
6	2	1	3
7	2	2	4
8	2	3	5
9	2	4	6
10	2	5	7

Среднее для каждой строчки, включая саму строчку и две предыдущие:

In [170]:

```
sql = '''
select t.*,
avg(t.revenue * 1.0) over (
        partition by t.user_id order by t.dt rows between 2 preceding and current row
    ) as moving_avg
from revenue t
'''
select(sql)
```

Out[170]:

	user_id	dt	revenue	moving_avg
0	1	1	1	1.0
1	1	2	2	1.5
2	1	3	3	2.0
3	1	4	4	3.0
4	1	5	5	4.0
5	1	6	6	5.0
6	2	1	3	3.0
7	2	2	4	3.5
8	2	3	5	4.0
9	2	4	6	5.0
10	2	5	7	6.0

08-conclusion

In [171]:

```
sql = '''drop table if exists Employee
create table Employee(Id int, Salary int)
insert into Employee(Id, Salary) values (1, 100)
insert into Employee(Id, Salary) values (2, 200)
insert into Employee(Id, Salary) values (3, 300)
'''
```

поставим ; после каждой строки:

In [172]:

```
sql = ';\n'.join(sql.split('\n'))
print(sql)
```

```
drop table if exists Employee;
create table Employee(Id int, Salary int);
insert into Employee(Id, Salary) values (1, 100);
insert into Employee(Id, Salary) values (2, 200);
insert into Employee(Id, Salary) values (3, 300);
```

In [173]:

```
conn.close()
```

8. Заключение

2. Где тренироваться

https://sql-ex.ru/ (https://sql-ex.ru/)

https://leetcode.com/ (https://leetcode.com/) - платный (дорогой)

Интерактивный тренажер по SQL · Stepik (https://stepik.org/course/63054/promo)

In [ ]:

```

```