Comment from v_3 reviewer Ekaterina, hello! Great, all critical comments have been worked out! If you still have little experience with SQL, I recommend the following resources for further study: 1. https://datalearn.ru/kurs-po-sql An excellent course, with its community in slack. You can get answers to all incomprehensible points 2. https://stepik.org/course/3203/promo (if you want to study deeper) Good luck in further learning!

In this part of the project, you need to write some SQL queries in the Jupyter Notebook. These tasks will be checked manually, and you will receive comments on the compiled queries.

The required data is in the schema tables. Don't forget to connect to the database using SQLAlchemy. Remember the instruction from lesson 11 "Cohort analysis. Presentation of Results" in Topic 4 "Installing and Configuring the Database and the Database Client". An example of code for connecting to the database and uploading the results can be found in this notebook. `stackoverflow`

Some tasks include follow-up questions – don't miss them. Some of the questions can be answered with text, and some will require visualization. Remember that the query result can be uploaded to a dataframe.

To make the expected result easier to imagine, we added a small fragment of the final table to each task. In queries, you can use any suitable field names.

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
from sqlalchemy import create_engine
import seaborn as sns
```

Configuration for Database Connection `data-analyst-advanced-sql`

This database contains the schema that you will work with in the project `stackoverflow`

In [2]:
```
db_config = {
    'user': 'praktikum_student', # имя пользователя
    'pwd': 'Sdf4$2;d-d30pp', # пароль
    'host': 'rc1b-wcoijxj3yxfsf3fs.mdb.yandexcloud.net',
    'port': 6432, # порт подключения
    'db': 'data-analyst-advanced-sql' # название базы данных
}

connection_string = 'postgresql://{}:{}@{}:{}/{}'.format(
    db_config['user'],
    db_config['pwd'],
    db_config['host'],
    db_config['port'],
```

```
    db_config['db'],
)
```

To create a connection

In [3]:
```
engine = create_engine(connection_string)
```

Example of a database query

`sample_df` is a pandas dataframe.

In [4]:
```
query = '''
SELECT *
FROM stackoverflow.users
LIMIT 10;
'''

sample_df = pd.read_sql_query(query, con=engine)
```

In [5]:
```
sample_df
```

Out[5]:

	id	creation_date	display_name	last_access_date	location	reputation	views
0	1	2008-07-31 14:22:31	Jeff Atwood	2018-08-29 02:34:23	El Cerrito, CA	44300	408587
1	2	2008-07-31 14:22:31	Geoff Dalgas	2018-08-23 17:31:56	Corvallis, OR	3491	23966
2	3	2008-07-31 14:22:31	Jarrod Dixon	2018-08-30 20:56:24	Raleigh, NC, United States	13418	24396
3	4	2008-07-31 14:22:31	Joel Spolsky	2018-08-14 22:18:15	New York, NY	28768	73755
4	5	2008-07-31 14:22:31	Jon Galloway	2018-08-29 16:48:36	San Diego, CA	39172	11700
5	8	2008-07-31 21:33:24	Eggs McLaren	2018-04-09 02:04:56	None	942	6372
6	9	2008-07-31 21:35:27	Kevin Dente	2018-08-30 18:18:03	Oakland, CA	14337	4949
7	11	2008-08-01 00:59:11	Anonymous User	2008-08-01 00:59:11	None	1890	2123
8	13	2008-08-01 04:18:05	Chris Jester-Young	2018-08-30 02:47:23	Raleigh, NC, United States	177138	35414
9	17	2008-08-01 12:02:22	Nick Berardi	2018-01-22 01:35:38	Issaquah, WA	44443	4786

Task 1

Display the total number of views of posts for each month of 2008. If there is no data for any month in the database, you can skip such a month. Sort the result in descending order of the

total number of views.

month_date	total_views
2008-09-01	452928568
2008-10-01	365400138
...	...

In [6]:
```
query = '''
SELECT *
FROM stackoverflow.posts
LIMIT 5;
'''

sample_df = pd.read_sql_query(query, con=engine)
```

In [7]:
```
sample_df
```

Out[7]:

	id	title	creation_date	favorites_count	last_activity_date	last_edit_date	user_id	parent_id
0	4	Convert Decimal to Double?	2008-07-31 21:42:53	41	2018-07-02 17:55:27.247	2018-07-02 17:55:27	8	0
1	6	Percentage width child element in absolutely p...	2008-07-31 22:08:09	10	2016-03-19 06:10:52.170	2016-03-19 06:05:48	9	0
2	7	None	2008-07-31 22:17:58	0	2017-12-16 05:06:57.613	2017-12-16 05:06:58	9	4
3	9	How do I calculate someone's age in C#?	2008-07-31 23:41:00	399	2018-07-25 11:57:14.110	2018-04-21 17:48:14	1	0
4	11	Calculate relative time in C#	2008-07-31 23:55:38	529	2018-07-05 04:00:56.633	2017-06-04 15:51:20	1	0

In [8]:
```
# напишите запрос
query = '''
SELECT SUM(views_count) AS total_views ,
       DATE_TRUNC('month',creation_date)::date AS  month_date
FROM stackoverflow.posts
WHERE creation_date BETWEEN '01-01-2008' AND '31-12-2008'
GROUP BY DATE_TRUNC('month',creation_date)::date
ORDER BY 1 DESC;
'''

# выполните запрос
```

In [9]:
```
sample_1 = pd.read_sql_query(query, con=engine)
sample_1
```

Out[9]:

	total_views	month_date
0	452928568	2008-09-01
1	365400138	2008-10-01
2	221759651	2008-11-01
3	197792841	2008-12-01
4	131367083	2008-08-01
5	669895	2008-07-01

▶ Hint

Analyze the summary table. Is the data different for different months? What could be the reason for the differences?

Write your answer here Monthly data varies. The number of posts has been growing since the beginning of the year.

Task 2

Display the names of the most active users who gave more than 100 responses in the first month after registration (including the day of registration). Do not take into account the questions that users asked. For each user name, print the number of unique values. Sort the result by the name field in lexicographic order. `user_id`

display_name	count
1800 INFORMATION	1
Adam Bellaire	1
Adam Davis	1
...	...

In [10]:
```
# напишите запрос
query = '''
SELECT COUNT(DISTINCT p.user_id) AS count,
      display_name

FROM stackoverflow.users us
JOIN stackoverflow.posts p ON us.id=p.user_id
WHERE post_type_id = 2 AND DATE_TRUNC('day', p.creation_date)>= DATE_TRUNC('day', us.c
                    AND DATE_TRUNC('day', p.creation_date)<= DATE_TRUNC('day', us.crea

GROUP BY 2
HAVING COUNT(p.id)>100
ORDER BY 2;
'''
```

In [11]:
```
sample_2 = pd.read_sql_query(query, con=engine)
sample_2
```

Out[11]:

	count	display_name
0	1	1800 INFORMATION
1	1	Adam Bellaire
2	1	Adam Davis
3	1	Adam Liss
4	8	Alan
...	...	...
74	1	lomaxx
75	1	mattlant
76	1	paxdiablo
77	1	tvanfosson
78	1	tzot

79 rows × 2 columns

▶ Hint

What anomalies are observed in the data? What are they talking about?

Write your answer here

There is a strange name in the first line.

Task 3

Display the number of posts for 2008 by month. Select posts from users who registered in September 2008 and made at least one post in December of the same year. Sort the table by month value in descending order.

month	count
2008-12-01	17641
2008-11-01	18294
...	...

In [12]:
```
# напишите запрос
query = '''
```

```
WITH abc AS
(SELECT p.user_id AS users

FROM stackoverflow.users us
JOIN stackoverflow.posts p ON us.id=p.user_id

WHERE DATE_TRUNC('month', us.creation_date) BETWEEN '2008-09-01' AND '2008-09-30' AND
       (DATE_TRUNC('month', p.creation_date) BETWEEN '2008-12-01' AND '2008-12-31')

GROUP BY 1)

SELECT COUNT(id) AS count,
       DATE_TRUNC('month', p.creation_date)
FROM abc ab
JOIN stackoverflow.posts p ON ab.users=p.user_id
WHERE DATE_TRUNC('year', p.creation_date) BETWEEN '2008-01-01' AND '2008-12-31'
GROUP BY 2
ORDER BY 1;

'''
```

In [13]:
```
sample_3 = pd.read_sql_query(query, con=engine)
sample_3
```

Out[13]:

	count	date_trunc
0	32	2008-08-01
1	17641	2008-12-01
2	18294	2008-11-01
3	24870	2008-09-01
4	27171	2008-10-01

▶ Hint

Examine the data: Are there any anomalies in them? Guess why anomalous values might have appeared.

Write your answer here

Yes, there are anomalies - users registered in September - and the post date is in August. The assumption is that the person registered after a month of publishing the post, perhaps there is an anonymous function. And then a month later I registered. The number of posts is growing from month to month.

Task 4

Using the post data, print a few fields:

- ID of the user who wrote the post;
- the date the post was created;
- the number of views of the current post;
- the sum of views of the author's posts with accumulation.

The data in the table should be sorted in ascending order of user IDs, and data about the same user should be sorted in ascending order of the date the post was created.

user_id	creation_date	views_count	cumulative_count
1	2008-07-31 23:41:00	480476	480476
1	2008-07-31 23:55:38	136033	616509
1	2008-07-31 23:56:41	0	616509
...	...	...	...
2	2008-07-31 23:56:41	79087	79087
2	2008-08-01 05:09:56	65443	144530
...	...	...	...

In [16]:
```
# напишите запрос

query = '''

SELECT user_id,
       creation_date,
       views_count,
       SUM(views_count) OVER (ORDER BY user_id, creation_date) AS cumulative_count
FROM stackoverflow.posts;

'''
```

In [15]:
```
sample_4 = pd.read_sql_query(query, con=engine)
sample_4
```

Out[15]:

	user_id	creation_date	views_count	cumulative_count
0	1	2008-07-31 23:41:00	480476	480476
1	1	2008-07-31 23:55:38	136033	616509
2	1	2008-07-31 23:56:41	0	616509
3	1	2008-08-04 02:45:08	0	616509
4	1	2008-08-04 04:31:03	0	616509
...	...	...	...	...
243791	5696608	2008-12-23 16:00:37	0	1369918176
243792	5696608	2008-12-23 17:35:09	0	1369918176
243793	5696608	2008-12-24 01:02:48	0	1369918176
243794	5696608	2008-12-30 14:34:45	0	1369918176
243795	5696608	2008-12-30 16:32:12	0	1369918176

243796 rows × 4 columns

▶ Hint

Task 5

Find the average number of user posts per day for August 2008. Select data on users who published more than 120 posts in August. Do not take into account the days without publications.

Sort the result in ascending order of the average number of posts. Values can be left blank.

user_id	avg_daily
116	4.777778
234	5.208333
...	...

In [17]:

```
# напишите запрос

query = '''

WITH abc AS
(SELECT user_id
FROM   stackoverflow.posts
WHERE DATE_TRUNC('day', creation_date)::date BETWEEN '2008-08-01' AND '2008-08-31'
GROUP BY 1
HAVING COUNT(id)>120),

def AS
(SELECT po.user_id AS user_id,
```

```
       DATE_TRUNC('day', creation_date)::date AS days,
       COUNT(DATE_TRUNC('day', creation_date)) AS posts_counts

FROM abc ab
JOIN stackoverflow.posts po ON ab.user_id=po.user_id
WHERE DATE_TRUNC('day', creation_date)::date BETWEEN '2008-08-01' AND '2008-08-31'

GROUP BY 1,2)

SELECT user_id,
       AVG(posts_counts) AS jjj
FROM def
Group BY 1
ORDER BY 2;
'''
```

In [18]:
```
sample_5 = pd.read_sql_query(query, con=engine)
sample_5
```

Out[18]:

	user_id	jjj
0	116	4.777778
1	234	5.208333
2	91	5.681818
3	905	7.000000
4	383	7.277778

▶ Hint

Task 6

On average, how many days did users interact with the platform between December 1 and December 7, 2008? For each user, select the days on which he or she published at least one post. You need to get one integer - do not forget to round the result.

result
<integer>

In [19]:
```
query = '''

WITH abc AS
(SELECT user_id,
       COUNT(DISTINCT DATE_TRUNC('day', creation_date)) AS jjj
FROM stackoverflow.posts
WHERE DATE_TRUNC ('day', creation_date) BETWEEN '2008-12-01' AND '2008-12-07'
GROUP BY 1)


SELECT ROUND(AVG(jjj))::varchar
FROM abc;
```

```
'''
```

In [20]:
```
sample_6 = pd.read_sql_query(query, con=engine)
sample_6
```

Out[20]:

	round
0	2

▶ Hint

Analyze the final table - what conclusions can be drawn?

In [20]:
```
# напишите ваш ответ здесь
# В среднем активных дней у каждого пользователя было 2
```

Task 7

Display the history of each user's activity in the following form: user ID, date of publication of the post. Sort the output in ascending order of user IDs, and for each user, in ascending order of publication date.

Add a new field to the table: for each post, it will indicate the name of the month of the user's penultimate publication relative to the current one. If there is no such publication, specify . Python will automatically change to , but you don't need to convert the values additionally. `NULL NULL None None`

Look carefully at the sample table: there is no penultimate publication for the first two posts, but starting from the third post, the required month is included in the new field. For the next user, the first two records of the field will also include . `second_last_month NULL`

user_id	creation_date	second_last_month
1	2008-07-31 23:41:00	None
1	2008-07-31 23:55:38	None
1	2008-07-31 23:56:41	July
1	2008-08-04 02:45:08	July
1	2008-08-04 04:31:03	July
1	2008-08-04 08:04:42	August
...	...	...

In [21]:
```
query = '''

WITH abc AS
(SELECT user_id,
```

```
       creation_date
FROM   stackoverflow.posts
ORDER BY user_id,creation_date),


def AS
(SELECT user_id,
       creation_date,
       LAG(creation_date,2) OVER (PARTITION BY user_id ORDER BY creation_date) AS prec
FROM abc),



ghk AS
(SELECT user_id,
       creation_date,
       EXTRACT('month' FROM predposlednyi) AS second_last_month
FROM def)

SELECT user_id,
       creation_date,
       CASE second_last_month
       WHEN 1 THEN 'January'
       WHEN 2 THEN 'February'
       WHEN 3 THEN 'March'
       WHEN 4 THEN 'April'
       WHEN 5 THEN 'May'
       WHEN 6 THEN 'June'
       WHEN 7 THEN 'July'
       WHEN 8 THEN 'August'
       WHEN 9 THEN 'September'
       WHEN 10 THEN 'October'
       WHEN 11 THEN 'November'
       WHEN 12 THEN 'December'
       END AS second_last_month
FROM ghk;

'''
```

In [22]:

```
sample_7 = pd.read_sql_query(query, con=engine)
sample_7
```

Out[22]:

	user_id	creation_date	second_last_month
0	1	2008-07-31 23:41:00	None
1	1	2008-07-31 23:55:38	None
2	1	2008-07-31 23:56:41	July
3	1	2008-08-04 02:45:08	July
4	1	2008-08-04 04:31:03	July
...	...	...	...
243791	5696608	2008-12-23 16:00:37	December
243792	5696608	2008-12-23 17:35:09	December
243793	5696608	2008-12-24 01:02:48	December
243794	5696608	2008-12-30 14:34:45	December
243795	5696608	2008-12-30 16:32:12	December

243796 rows × 3 columns

▶ Hint

Task 8

Calculate the Retention Rate by month for StackOverflow users. Organize users into cohorts based on the month of their first post. Return is determined by the presence of a post in the current month.

cohort_dt	session_date	users_cnt	cohort_users_cnt	retention_rate
2008-07-01 00:00:00	2008-07-01 00:00:00	3	3	100
2008-07-01 00:00:00	2008-08-01 00:00:00	2	3	66,67
2008-07-01 00:00:00	2008-09-01 00:00:00	1	3	33,33
2008-07-01 00:00:00	2008-10-01 00:00:00	2	3	66,67
2008-07-01 00:00:00	2008-11-01 00:00:00	1	3	33,33
2008-07-01 00:00:00	2008-12-01 00:00:00	2	3	66,67
2008-08-01 00:00:00	2008-08-01 00:00:00	2151	2151	100
...	...	...	...	...

In [23]:

```
query = '''

WITH start_cohort AS

(SELECT  DISTINCT user_id,
FIRST_VALUE(DATE_TRUNC('month', creation_date))OVER(PARTITION BY user_id ORDER BY crea
FROM stackoverflow.posts),
```

```
profile AS
(SELECT   user_id,
          cohort_dt,
          COUNT(*) OVER (PARTITION BY cohort_dt) AS cohort_users_cnt
FROM start_cohort),

sessions AS
(SELECT  user_id,
        DATE_TRUNC('month', creation_date) ::date AS session_date
FROM stackoverflow.posts
GROUP BY 1,2)

SELECT cohort_dt,
       session_date,
       COUNT(p.user_id) AS users_cnt,
       cohort_users_cnt,
       ROUND(COUNT(p.user_id) * 100.0 / cohort_users_cnt, 2) AS retention_rate
FROM profile p
JOIN sessions s ON p.user_id = s.user_id
GROUP BY 1,2,4
ORDER BY 1,2;
'''
```

In [24]:
```
sample_8 = pd.read_sql_query(query, con=engine)
sample_8
```

Out[24]:

	cohort_dt	session_date	users_cnt	cohort_users_cnt	retention_rate
0	2008-07-01	2008-07-01	3	3	100.00
1	2008-07-01	2008-08-01	2	3	66.67
2	2008-07-01	2008-09-01	1	3	33.33
3	2008-07-01	2008-10-01	2	3	66.67
4	2008-07-01	2008-11-01	1	3	33.33
5	2008-07-01	2008-12-01	2	3	66.67
6	2008-08-01	2008-08-01	2151	2151	100.00
7	2008-08-01	2008-09-01	1571	2151	73.04
8	2008-08-01	2008-10-01	1275	2151	59.27
9	2008-08-01	2008-11-01	1050	2151	48.81
10	2008-08-01	2008-12-01	894	2151	41.56
11	2008-09-01	2008-09-01	7678	7678	100.00
12	2008-09-01	2008-10-01	4132	7678	53.82
13	2008-09-01	2008-11-01	2966	7678	38.63
14	2008-09-01	2008-12-01	2500	7678	32.56
15	2008-10-01	2008-10-01	3629	3629	100.00
16	2008-10-01	2008-11-01	1640	3629	45.19
17	2008-10-01	2008-12-01	1221	3629	33.65
18	2008-11-01	2008-11-01	2852	2852	100.00
19	2008-11-01	2008-12-01	1151	2852	40.36
20	2008-12-01	2008-12-01	2536	2536	100.00

▶ Hint

Build a heat map of the Retention Rate. What anomalies or other unusual phenomena have been identified? Formulate hypotheses about possible causes.

In [25]:
```
sample_8.pivot('cohort_dt', 'session_date', 'retention_rate')
```

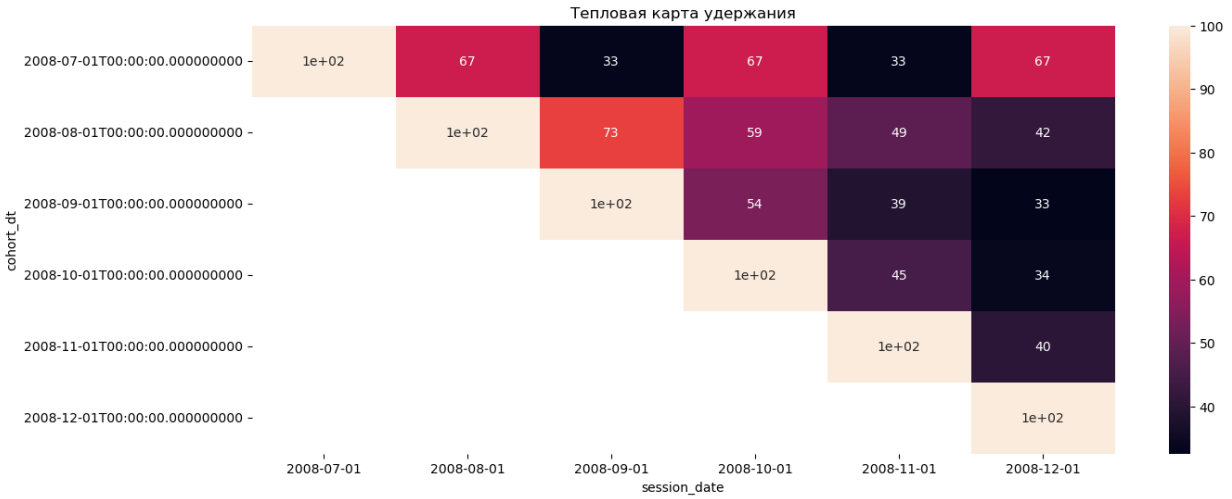
Out[25]:

session_date	2008-07-01	2008-08-01	2008-09-01	2008-10-01	2008-11-01	2008-12-01
cohort_dt						
2008-07-01	100.0	66.67	33.33	66.67	33.33	66.67
2008-08-01	NaN	100.00	73.04	59.27	48.81	41.56
2008-09-01	NaN	NaN	100.00	53.82	38.63	32.56
2008-10-01	NaN	NaN	NaN	100.00	45.19	33.65
2008-11-01	NaN	NaN	NaN	NaN	100.00	40.36
2008-12-01	NaN	NaN	NaN	NaN	NaN	100.00

In [26]:

```
glue = sample_8.pivot('cohort_dt', 'session_date', 'retention_rate')
plt.figure(figsize=(15, 6))  # задаём размер графика
sns.heatmap(
    glue,
    annot=True,
)
plt.title('Тепловая карта удержания')  # название графика

plt.show()
```



Describe anomalies or other unusual phenomena and formulate hypotheses

The coefficient should decrease over time, and here in the cohort on July 1, there is an alternation-more-less-more-less.

Task 9

By what percentage did the number of posts change monthly from September 1 to December 31, 2008? Display a table with the following fields:

- month number;
- the number of posts per month;
- percentage, which shows how much the number of posts has changed in the current month compared to the previous one.

If there are fewer posts, the percentage value should be negative, if more - positive. Round the percentage value to two decimal places.

Recall that if you divide one integer by another in PostgreSQL, the result is an integer rounded down to the nearest integer. To avoid this, convert the dividend to the type . `numeric`

creation_month	posts_count	percentage
9	70731	Nan
10	63102	-10.33
...	...	...

In [27]:
```
query = '''

WITH abc AS
(SELECT
      DISTINCT EXTRACT('month' FROM creation_date) AS creation_month,
      COUNT(*) OVER (PARTITION BY EXTRACT('month' FROM creation_date)) AS posts_count
FROM stackoverflow.posts
WHERE creation_date BETWEEN '2008-09-01' AND '2008-12-31')

SELECT *,
ROUND(((posts_count::numeric  - LAG(posts_count) OVER (ORDER BY creation_month))/LAG(p

FROM abc;

'''
```

In [29]:
```
sample_9 = pd.read_sql_query(query, con=engine)
sample_9.head(10)
```

Out[29]:

	creation_month	posts_count	percentage
0	9.0	70371	NaN
1	10.0	63102	-10.33
2	11.0	46975	-25.56
3	12.0	44592	-5.07

In [30]:
```
import plotly.express as px


data = {"Name": ["9", "10", "11", "12"], "Value": [70371, 63102, 46975, 44592]}
df = pd.DataFrame(data)


figure = px.pie(df, values='Value', names='Name',
```

```
                  title='КРУГОВАЯ ДИАГРАММА',)

figure.update_traces(textposition='inside',
                     text=df['Value'].map("{:,}".format),
                     textinfo='percent+label+text')

figure.show()
```

▶ Hint

Build a pie chart with the number of posts by month.

In [30]:
```
# постройте круговую диаграмму с количеством постов по месяцам
```

Task 10

Upload the activity data of the user who published the most posts of all time. Display the data for October 2008 in the following form:

- week number;
- The date and time of the last post published this week.

week_creation	creation_date
40	2008-10-05 09:00:58
41	2008-10-12 21:22:23
...	...

In [31]:
```
# напишите запрос
query = '''

WITH abc AS
(SELECT
       DISTINCT user_id AS pobeditel,
       COUNT(id) AS posts

FROM stackoverflow.posts
GROUP BY 1
ORDER BY 2 DESC
LIMIT 1),


def AS
(SELECT creation_date,
       EXTRACT('week' FROM creation_date) AS week_creation
FROM  abc ab
JOIN stackoverflow.posts po ON ab.pobeditel=po.user_id
WHERE DATE_TRUNC('day', creation_date )>= '2008-10-01' AND DATE_TRUNC('day', creation_
ORDER BY creation_date)

SELECT DISTINCT week_creation,
    LAST_VALUE(creation_date) OVER (PARTITION BY week_creation) AS creation_date
FROM def
ORDER BY 1;

'''
```

In [32]:
```
sample_10 = pd.read_sql_query(query, con=engine)
sample_10.head(10)
```

Out[32]:

	week_creation	creation_date
0	40.0	2008-10-05 09:00:58
1	41.0	2008-10-12 21:22:23
2	42.0	2008-10-19 06:49:30
3	43.0	2008-10-26 21:44:36
4	44.0	2008-10-31 22:16:01

▶ Hint