Екатерина, привет!

Меня зовут Максим Попов, и я буду проверять твой проект. Давай будем общаться на «ты». Но если тебе комфортнее на «вы», обязательно сообщи мне об этом. По ходу тетрадки ты найдешь комментарии к проекту, они будут в отдельных ячейках Markdown с заголовком «Комментарий ревьюера». Постарайся учесть эти комментарии для дальнейших проектов.

Я буду отмечать комментарии цветом и примечанием. В конце заголовка комментария будет указан номер итерации проверки. Пожалуйста, не удаляй и не правь мои комментарии

Критично - требует исправления, влияет на удачное выполнение проекта

Рекомендация - комментарий является рекомендацией или советом. Можешь учесть их при выполнении будущих заданий. На твое усмотрение

Отлично - Так отмечены удачные решения и подходы, на которые стоит опираться в будущих проектах

Комментарий от ревьюера Критично Такой комментарий нужно исправить обязательно, он критически влияет на удачное выполнение проекта.

Комментарий от ревьюера Рекомендация Такой комментарий является рекомендацией или советом. Можешь учесть их при выполнении будущих заданий. На твое усмотрение.

Комментарий от ревьюера Отлично Так отмечены удачные решения и подходы, на которые стоит опираться в будущих проектах

Желательно реагировать на комментарии ('исправлено', 'не понятно как исправить ошибку' и т.п.) - так мы сможем более эффективно поработать над этим проектом.

Будет здорово, если ты будешь помечать свои действия следующим образом:

Комментарий студента:

Комментарий от ревьюера v_1 Екатерина, ты отлично разбираешься с запросами, ошибки, которые нашел, неочевидны Рекомендую всегда оставлять комментарии на подзапросы, такая привычка поможет тебе, когда придется возвращаться к старому коду Подробные комментарии по ходу работы Желтые комментарии-рекомендации на твое усмотрение. Если есть вопросы ко мне, можешь оставить их в отдельном комментарии

Комментарий от ревьюера v_2 Екатерина, привет! Почти все готово, осталось 6-й запрос допилить

Комментарий от ревьюера v_3 Екатерина, привет! Отлично, все критичные комментарии отработаны! Если у тебя пока небольшой опыт работы с SQL, для дальнейшего изучения рекомендую следующие ресурсы: 1. https://datalearn.ru/kurs-po-sql Превосходный курс, со своим сообществом в слаке. Можно получить ответы по всем непонятным моментам 2. https://stepik.org/course/3203/promo (если есть желание изучить поглубже) Успеха в дальнейшем обучении!

В этой части проекта вам нужно написать несколько SQL-запросов в Jupyter Notebook. Эти задания проверят вручную, и вы получите комментарии к составленным запросам.

Необходимые данные находятся в таблицах схемы `stackoverflow`. Не забудьте подключиться к базе с помощью SQLAlchemy. Вспомните инструкцию из урока 11 «Когортный анализ. Представление результатов» в теме 4 «Установка и настройка базы данных и клиента базы данных».. Пример кода для подключения к базе и выгрузки результатов вы найдёте и в этой тетрадке.

Некоторые задания включают дополнительные вопросы — не пропустите их. На часть вопросов можно ответить текстом, а для некоторых понадобится визуализация. Помните, что результат запроса можно выгрузить в датафрейм.

Чтобы ожидаемый результат было легче представить, мы добавили к каждому заданию небольшой фрагмент итоговой таблицы. В запросах вы можете использовать любые подходящие названия полей.

In [1]:
```
import pandas as pd
```

```
import matplotlib.pyplot as plt
from sqlalchemy import create_engine
import seaborn as sns
```

Конфигурация для подключения к базе данных `data-analyst-advanced-sql`

Эта база данных содержит схему `stackoverflow`, с которой вы будете работать в проекте

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'],
)
```

Создание подключения

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

Пример запроса к базе данных

`sample_df` является pandas-датафреймом.

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

Комментарий от ревьюера v_1 Отлично К базе подключились, приступаем к запросам

Задание 1

Выведите общую сумму просмотров постов за каждый месяц 2008 года. Если данных за какой-либо месяц в базе нет, такой месяц можно пропустить. Результат отсортируйте по убыванию общего количества просмотров.

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	post_type_id	score	views_count
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	573	37080
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	1	256	16306
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	2	401	0
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	1	1743	480476
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	1	1348	136033

In [7]:
```
# 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 [8]:
```
sample_1 = pd.read_sql_query(query, con=engine)
sample_1
```

Out[8]:

	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

▶ Подсказка

Проанализируйте итоговую таблицу. Отличаются ли данные за разные месяцы? С чем могут быть связаны отличия?

напишите ваш ответ здесь Данные за месяца различаются. Количество постов с начала года растёт.

Комментарий от ревьюера v_1 □□

Задание 2

Выведите имена самых активных пользователей, которые в первый месяц после регистрации (включая день регистрации) дали больше 100 ответов. Вопросы, которые задавали пользователи, не учитывайте. Для каждого имени пользователя выведите

количество уникальных значений `user_id`. Отсортируйте результат по полю с именами в лексикографическом порядке.

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

In [9]:
```
# 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.creation_date)
                    AND DATE_TRUNC('day', p.creation_date)<= DATE_TRUNC('day', us.creation_date) + INTERVAL '1 mo

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

In [10]:
```
sample_2 = pd.read_sql_query(query, con=engine)
sample_2
```

Out[10]:

	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

▶ Подсказка

Какие аномалии наблюдаются в данных? О чём они говорят?

напишите ваш ответ здесь

В первой строчке странное имя.

Задание 3

Выведите количество постов за 2008 год по месяцам. Отберите посты от пользователей, которые зарегистрировались в сентябре 2008 года и сделали хотя бы один пост в декабре того же года. Отсортируйте таблицу по значению месяца по убыванию.

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

In [11]:
```
# 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 [12]:

```
sample_3 = pd.read_sql_query(query, con=engine)
sample_3
```

Out[12]:

	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

▶ Подсказка

Изучите данные: есть ли в них аномалии? Предположите, почему могли появиться аномальные значения.

напишите ваш ответ здесь

Да есть аномалии-пользователи зарегестрировались в сентябре-а дата поста в августе. Предположение-человек зарегестрировался после месяца публикации поста-возможно есть функция-аноним. А потом через месяц зарегестрировался. Количество постов от месяца к месяцу-растет.

Задание 4

Используя данные о постах, выведите несколько полей:

- идентификатор пользователя, который написал пост;
- дата создания поста;
- количество просмотров у текущего поста;
- сумму просмотров постов автора с накоплением.

Данные в таблице должны быть отсортированы по возрастанию идентификаторов пользователей, а данные об одном и том же пользователе — по возрастанию даты создания поста.

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 [13]:

```
# 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 [14]:

```
sample_4 = pd.read_sql_query(query, con=engine)
sample_4
```

Out[14]:

	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

▶ Подсказка

Задание 5

Найдите среднее количество постов пользователей в день за август 2008 года. Отберите данные о пользователях, которые опубликовали больше 120 постов за август. Дни без публикаций не учитывайте.

Отсортируйте результат по возрастанию среднего количества постов. Значения можно не округлять.

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

In [15]:

```
# 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 [16]:
```
sample_5 = pd.read_sql_query(query, con=engine)
sample_5
```

Out[16]:

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

▶ Подсказка

Задание 6

Сколько в среднем дней в период с 1 по 7 декабря 2008 года пользователи взаимодействовали с платформой? Для каждого пользователя отберите дни, в которые он или она опубликовали хотя бы один пост. Нужно получить одно целое число — не забудьте округлить результат.

result
<целое число>

In [17]:
```
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 [18]:
```
sample_6 = pd.read_sql_query(query, con=engine)
sample_6
```

Out[18]:

	round
0	2

▶ Подсказка

Проанализируйте итоговую таблицу — какие выводы можно сделать?

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

Задание 7

Выведите историю активности каждого пользователя в таком виде: идентификатор пользователя, дата публикации поста. Отсортируйте вывод по возрастанию идентификаторов пользователей, а для каждого пользователя — по возрастанию даты публикации.

Добавьте в таблицу новое поле: для каждого поста в нём будет указано название месяца предпоследней публикации пользователя относительно текущей. Если такой публикации нет, укажите `NULL`. Python автоматически поменяет `NULL` на `None`, но дополнительно преобразовывать значения `None` вам не нужно.

Посмотрите внимательно на образец таблицы: для первых двух постов предпоследней публикации нет, но, начиная с третьего поста, в новое поле входит нужный месяц. Для следующего пользователя в первые две записи поля `second_last_month` тоже войдёт `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 [19]:
```
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 predposlednyi
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 [20]:
```
sample_7 = pd.read_sql_query(query, con=engine)
sample_7
```

Out[20]:

	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

▶ Подсказка

Задание 8

Рассчитайте аналог Retention Rate по месяцам для пользователей StackOverflow. Объедините пользователей в когорты по месяцу их первого поста. Возвращение определяйте по наличию поста в текущем месяце.

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

WITH start_cohort AS

(SELECT  DISTINCT user_id,
FIRST_VALUE(DATE_TRUNC('month', creation_date))OVER(PARTITION BY user_id ORDER BY creation_date) AS cohort_dt
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 [22]:
```
sample_8 = pd.read_sql_query(query, con=engine)
sample_8
```

Out[22]:

	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

▶ Подсказка

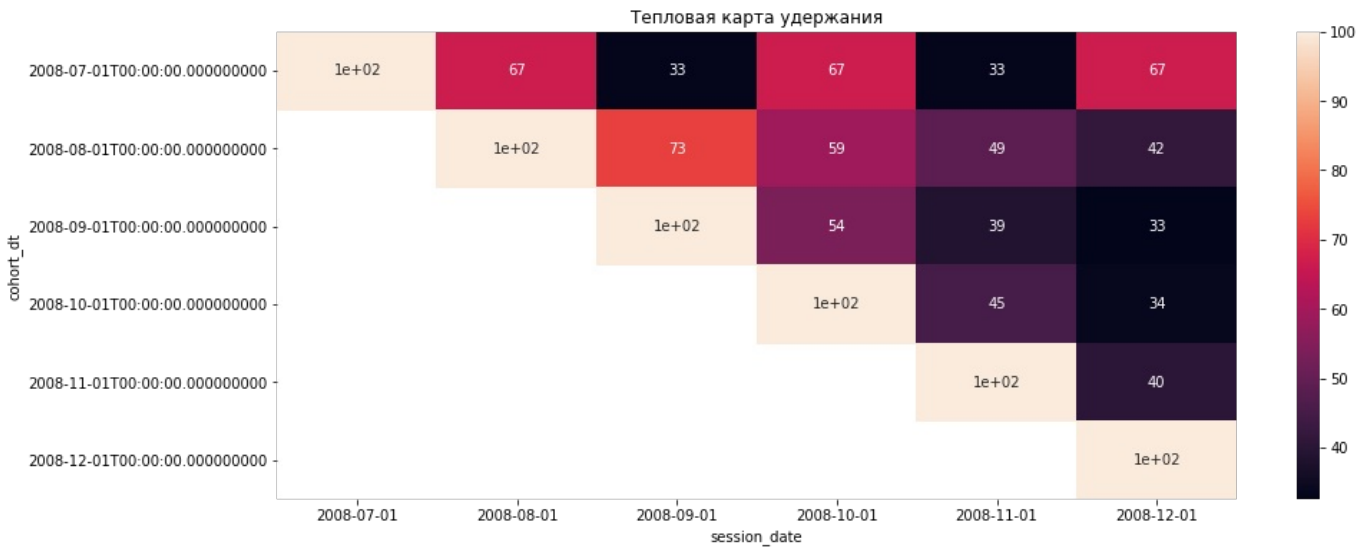
Постройте тепловую карту Retention Rate. Какие аномалии или другие необычные явления удалось выявить? Сформулируйте гипотезы о возможных причинах.

In [23]:
```
sample_8.pivot('cohort_dt', 'session_date', 'retention_rate')
```

Out[23]:

cohort_dt \ session_date	2008-07-01	2008-08-01	2008-09-01	2008-10-01	2008-11-01	2008-12-01
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 [24]:
```
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()
```



опишите аномалии или другие необычные явления и сформулируйте гипотезы

Коэффициент с течением времени должен уменьшаться-а тут в когорте на 1 июля-идет чередование-больше-меньше-больше-

меньше.

Задание 9

На сколько процентов менялось количество постов ежемесячно с 1 сентября по 31 декабря 2008 года? Отобразите таблицу со следующими полями:

- номер месяца;
- количество постов за месяц;
- процент, который показывает, насколько изменилось количество постов в текущем месяце по сравнению с предыдущим.

Если постов стало меньше, значение процента должно быть отрицательным, если больше — положительным. Округлите значение процента до двух знаков после запятой.

Напомним, что при делении одного целого числа на другое в PostgreSQL в результате получится целое число, округлённое до ближайшего целого вниз. Чтобы этого избежать, переведите делимое в тип `numeric`.

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

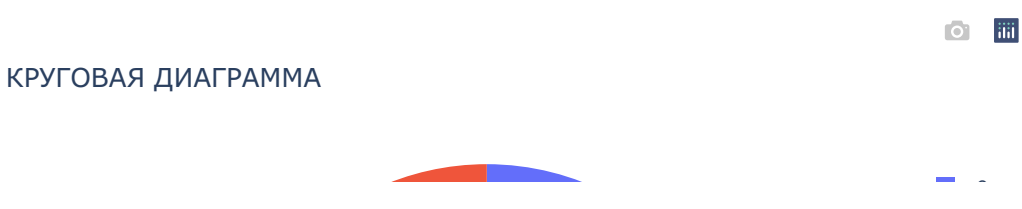
In [25]:
```
query = '''

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(posts_count) OVER (ORDER BY

FROM abc;

'''
```

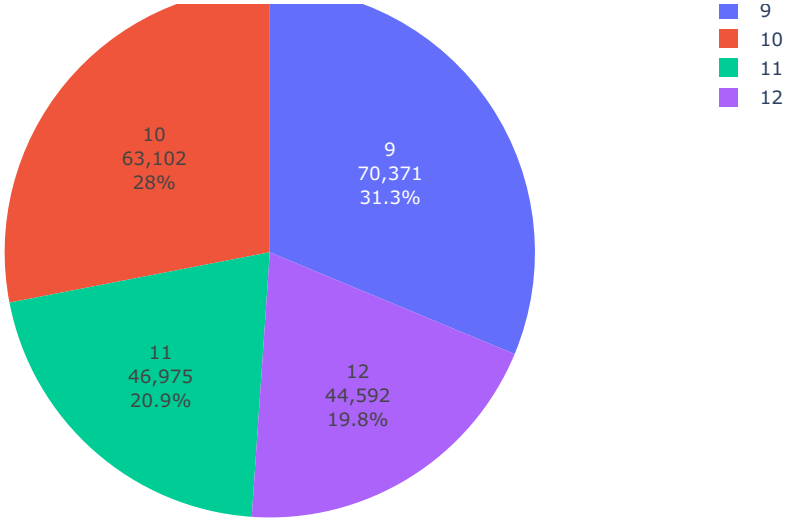
In [26]:
```
sample_9 = pd.read_sql_query(query, con=engine)
sample_9.head(10)
```

Out[26]:

	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 [27]:
```
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()
```





▶ Подсказка

Постройте круговую диаграмму с количеством постов по месяцам.

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

Задание 10

Выгрузите данные активности пользователя, который опубликовал больше всего постов за всё время. Выведите данные за октябрь 2008 года в таком виде:

- номер недели;
- дата и время последнего поста, опубликованного на этой неделе.

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

In [28]:
```
# 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_date ) <= '2008-10-31'
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 [29]:
```
sample_10 = pd.read_sql_query(query, con=engine)
```

```
sample_10 = pd.read_sql_query(query, con=engine)
sample_10.head(10)
```

Out[29]:

	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

▶ Подсказка