Продвинутый SQL

Описание данных

Проект состоит из двух частей:

. В первой части вы решите несколько задач в SQL-тренажёре, чтобы закрепить пройденный материал.

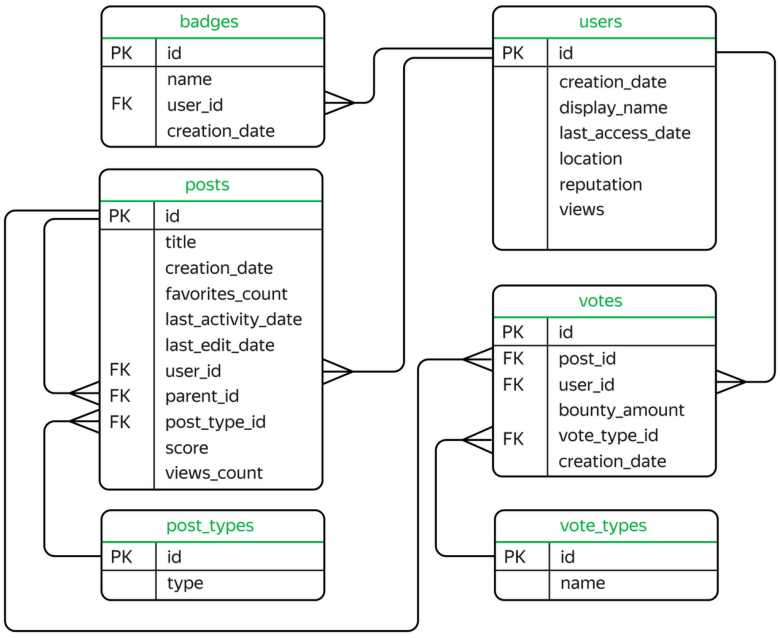
. Вторая часть проекта – аналитическая. Проверять задачи по-прежнему будет тренажёр. Однако мы рекомендуем самостоятельно анализировать полученные результаты и формулировать выводы.

Внимательно читайте условия. Формулировки задач в проекте приближены к реальной жизни. В работе часто встретятся такие формулировки: «рассчитать LTV за последние 30 дней», «отобразить самых активных покупателей» или даже «определить, почему в компании упали продажи».

В самостоятельном проекте вы будете работать с базой данных StackOverflow – сервиса вопросов и ответов о программировании. StackOverflow похож на социальную сеть – пользователи сервиса задают вопросы, отвечают на посты, оставляют комментарии и ставят оценки другим ответам.

Вы будете работать с версией базы, где хранятся данные о постах за 2008 год, но в таблицах вы найдёте информацию и о более поздних оценках, которые эти посты получили.

Изучите ER-диаграмму базы:



Теперь познакомьтесь с данными таблиц.

Таблица `stackoverflow.badges`

Хранит информацию о значках, которые присуждаются за разные достижения. Например, пользователь, правильно ответивший на большое количество вопросов про PostgreSQL, может получить значок postgresql.

Поле	Описание
id	Идентификатор значка, первичный ключ таблицы
name	Название значка
user_id	Идентификатор пользователя, которому присвоили значок, внешний ключ, отсылающий к таблице `users`
creation_date	Дата присвоения значка

Таблица stackoverflow.post_types

Содержит информацию о типе постов. Их может быть два:

. Question — пост с вопросом;

. Answer — пост с ответом.

Поле	Описание
id	Идентификатор типа поста, первичный ключ таблицы
type	Тип поста

Таблица stackoverflow.posts

Содержит информацию о постах.

Поле	Описание
id	Идентификатор поста, первичный ключ таблицы
title	Заголовок поста
creation_date	Дата создания поста
favorites_count	Число, которое показывает, сколько раз пост добавили в «Закладки»
last_activity_date	Дата последнего действия в посте, например комментария
last_edit_date	Дата последнего изменения поста
user_id	Идентификатор пользователя, который создал пост, внешний ключ к таблице users
parent_id	Если пост написали в ответ на другую публикацию, в это поле попадёт идентификатор поста с вопросом
post_type_id	Идентификатор типа поста, внешний ключ к таблице post_types

score	Количество очков, которое набрал пост
views_count	Количество просмотров

Таблица `stackoverflow.users`

Содержит информацию о пользователях.

Поле	Описание
id	Идентификатор пользователя, первичный ключ таблицы
creation_date	Дата регистрации пользователя
display_name	Имя пользователя
last_access_date	Дата последнего входа
location	Местоположение
reputation	Очки репутации, которые получают за хорошие вопросы и полезные ответы
views	Число просмотров профиля пользователя

Таблица `stackoverflow.vote_types`

Содержит информацию о типах голосов. Голос – это метка, которую пользователи ставят посту. Типов бывает несколько:

- `UpMod` – такую отметку получают посты с вопросами или ответами, которые пользователи посчитали уместными и полезными.
- `DownMod` – такую отметку получают посты, которые показались пользователям наименее полезными.
- `Close` – такую метку ставят опытные пользователи сервиса, если заданный вопрос нужно доработать или он вообще не подходит для платформы.
- `Offensive` – такую метку могут поставить, если пользователь ответил на вопрос в грубой и оскорбительной манере, например, указав на неопытность автора поста.

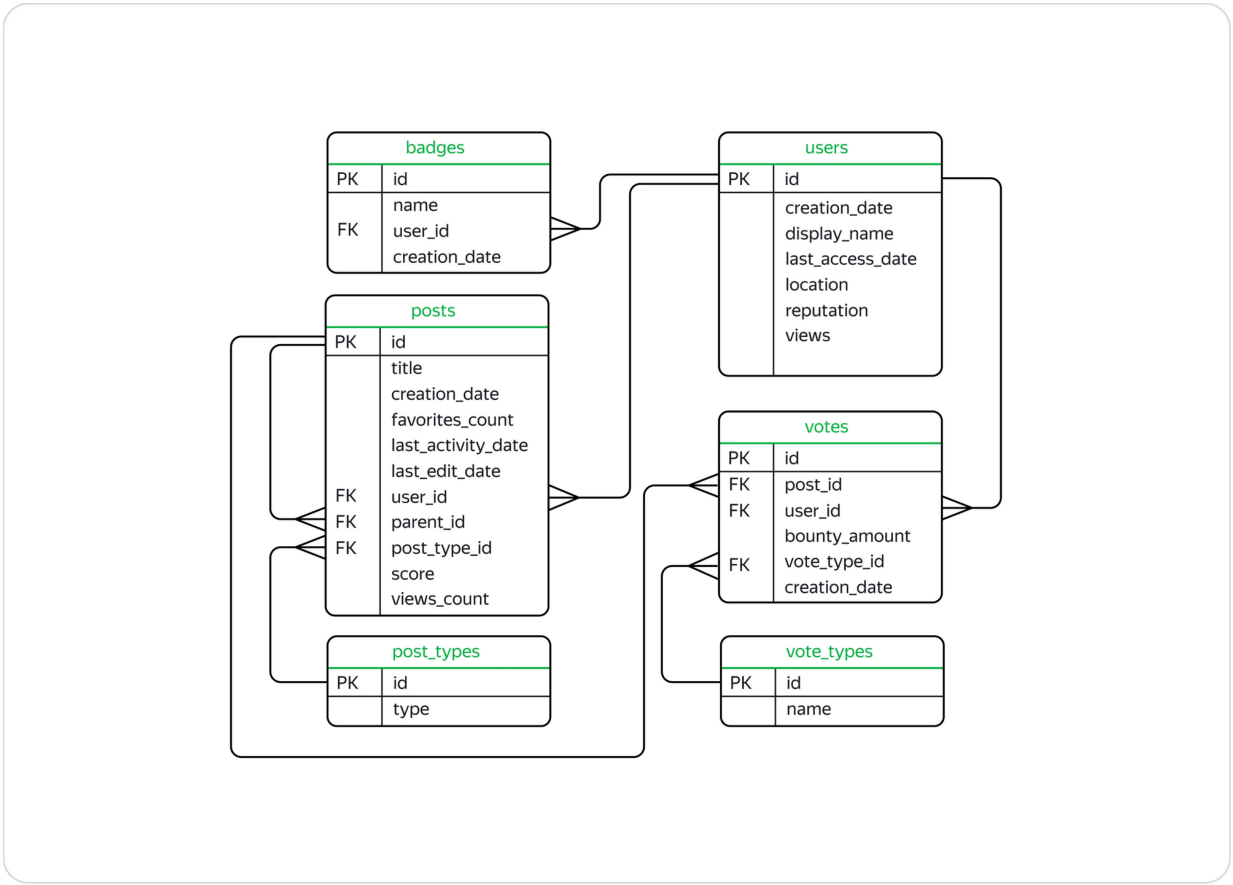
. `Spam` – такую метку ставят в случае, если пост пользователя выглядит откровенной рекламой.

Поле	Описание
id	Идентификатор типа голоса, первичный ключ
name	Название метки

Таблица `stackoverflow.votes`

Содержит информацию о голосах за посты.

Поле	Описание
id	Идентификатор голоса, первичный ключ
post_id	Идентификатор поста, внешний ключ к таблице `posts`
user_id	Идентификатор пользователя, который поставил посту голос, внешний ключ к таблице `users`
bounty_amount	Сумма вознаграждения, которое назначают, чтобы привлечь внимание к посту
vote_type_id	Идентификатор типа голоса, внешний ключ к таблице `vote_types`
creation_date	Дата назначения голоса



Задания (Первая часть)

1.Найдите количество вопросов, которые набрали больше 300 очков или как минимум 100 раз были добавлены в «Закладки».

```
SELECT COUNT(*)
FROM stackoverflow.posts --добавьте название таблицы и условие
WHERE post_type_id = 1 AND (score > 300 OR favorites_count >= 100)
```

2.Сколько в среднем в день задавали вопросов с 1 по 18 ноября 2008 включительно? Результат округлите до целого числа.

```
SELECT ROUND(AVG(cnt)) as avg
FROM
(SELECT DATE_TRUNC('day', creation_date)::date as dt,
        COUNT(*) as cnt
```

```
FROM
    stackoverflow.posts
WHERE
    post_type_id = 1
        AND CAST(creation_date as date) BETWEEN '2008-11-01' AND '2008-11-18'
GROUP BY 1) as t1;
```

3.Сколько пользователей получили значки сразу в день регистрации? Выведите количество уникальных пользователей.

```
SELECT COUNT(DISTINCT t1.id)
FROM stackoverflow.users as t1
INNER JOIN stackoverflow.badges as t2 ON t1.id = t2.user_id
WHERE t1.creation_date::date = t2.creation_date::date
```

4.Сколько уникальных постов пользователя с именем Joel Coehoorn получили хотя бы один голос?

```
SELECT COUNT(DISTINCT t1.id)
FROM stackoverflow.posts t1
JOIN stackoverflow.votes t2 ON t1.id = t2.post_id
WHERE t1.user_id IN
            (SELECT id
            FROM stackoverflow.users
            WHERE display_name LIKE '%Joel Coehoorn%')
```

5.Выгрузите все поля таблицы `vote_types`. Добавьте к таблице поле `rank`, в которое войдут номера записей в обратном порядке. Таблица должна быть отсортирована по полю `id`.

```
SELECT *,
        ROW_NUMBER() OVER (ORDER BY id DESC) as rank
FROM stackoverflow.vote_types
ORDER BY id
```

6.Отберите 10 пользователей, которые поставили больше всего голосов типа `Close`. Отобразите таблицу из двух полей: идентификатором пользователя и количеством голосов. Отсортируйте данные сначала по убыванию количества голосов, потом по убыванию значения идентификатора пользователя.

```
SELECT user_id,
       COUNT(vote_type_id)
FROM stackoverflow.votes
WHERE vote_type_id IN (SELECT id
                       FROM stackoverflow.vote_types
                       WHERE name = 'Close')
GROUP BY 1
ORDER BY 2 DESC,
         1 DESC
LIMIT 10;
```

7.Отберите 10 пользователей по количеству значков, полученных в период с 15 ноября по 15 декабря 2008 года включительно. Отобразите несколько полей:

. идентификатор пользователя;

. число значков;

. место в рейтинге – чем больше значков, тем выше рейтинг.

Пользователям, которые набрали одинаковое количество значков, присвойте одно и то же место в рейтинге.

Отсортируйте записи по количеству значков по убыванию, а затем по возрастанию значения идентификатора пользователя.

```
SELECT *,
       DENSE_RANK() OVER(ORDER BY cnt DESC)
FROM
(SELECT user_id,
       COUNT(id) as cnt
FROM stackoverflow.badges
WHERE DATE_TRUNC('day', creation_date) BETWEEN '2008-11-15' AND '2008-12-15'
GROUP BY 1
```

```
ORDER BY 2 DESC, user_id) as t1
LIMIT 10;
```

8.Сколько в среднем очков получает пост каждого пользователя?

Сформируйте таблицу из следующих полей:

. заголовок поста;

. идентификатор пользователя;

. число очков поста;

. среднее число очков пользователя за пост, округлённое до целого числа.

Не учитывайте посты без заголовка, а также те, что набрали ноль очков.

```
SELECT title,
        user_id,
        score,
        ROUND(AVG(score) OVER(PARTITION BY user_id)) as avg_score
FROM stackoverflow.posts
WHERE score <> 0 AND title <> '';
```

9.Отобразите заголовки постов, которые были написаны пользователями, получившими более 1000 значков. Посты без заголовков не должны попасть в список.

```
SELECT title
FROM stackoverflow.posts
WHERE title <> '' AND user_id IN (SELECT user_id FROM stackoverflow.badges GROUP BY 1
HAVING COUNT(id) > 1000)
```

10.Напишите запрос, который выгрузит данные о пользователях из Канады (англ. Canada). Разделите пользователей на три группы в зависимости от количества просмотров их профилей:

. пользователям с числом просмотров больше либо равным 350 присвойте группу `1`;

. пользователям с числом просмотров меньше 350, но больше либо равно 100 – группу `2`;

. пользователям с числом просмотров меньше 100 – группу `3`.

Отобразите в итоговой таблице идентификатор пользователя, количество просмотров профиля и группу. Пользователи с количеством просмотров меньше либо равным нулю не должны войти в итоговую таблицу.

```
SELECT id,
       views,
       CASE
           WHEN views >= 350 THEN 1
           WHEN views >= 100 AND views < 350 THEN 2
           ELSE 3
       END
FROM stackoverflow.users
WHERE location LIKE '%Canada%' AND views <> 0
```

11.Дополните предыдущий запрос. Отобразите лидеров каждой группы – пользователей, которые набрали максимальное число просмотров в своей группе. Выведите поля с идентификатором пользователя, группой и количеством просмотров. Отсортируйте таблицу по убыванию просмотров, а затем по возрастанию значения идентификатора.

```
SELECT t2.id,
        t2.gr,
        t2.views
FROM
(SELECT *,
        MAX(views) OVER(PARTITION BY gr) as max_score
FROM
(SELECT id,
       views,
       CASE
           WHEN views >= 350 THEN 1
           WHEN views >= 100 AND views < 350 THEN 2
           ELSE 3
       END as gr
```

```
FROM stackoverflow.users
WHERE location LIKE '%Canada%' AND views <> 0) as  t1
) as t2
WHERE max_score = views
ORDER BY 3 DESC, id;
```

12.Посчитайте ежедневный прирост новых пользователей в ноябре 2008 года. Сформируйте таблицу с полями:

. номер дня;

. число пользователей, зарегистрированных в этот день;

. сумму пользователей с накоплением.

```
SELECT DISTINCT EXTRACT(DAY FROM creation_date) as dt,
       COUNT(id) OVER (PARTITION BY creation_date::date) as cnt,
       COUNT(id) OVER (ORDER BY creation_date::date) as sum_user
FROM stackoverflow.users
WHERE DATE_TRUNC('month', creation_date) = '2008-11-01'
ORDER BY 1;
```

13.Для каждого пользователя, который написал хотя бы один пост, найдите интервал между регистрацией и временем создания первого поста. Отобразите:

. идентификатор пользователя;

. разницу во времени между регистрацией и первым постом.

```
SELECT id,
        t2.min_dt - creation_date
FROM stackoverflow.users
INNER JOIN
(SELECT DISTINCT user_id,
        MIN(creation_date) OVER(PARTITION BY user_id ORDER BY creation_date) as
min_dt
FROM stackoverflow.posts) as t2
ON stackoverflow.users.id = t2.user_id
```

Задания (Вторая часть)

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

```
SELECT DATE_TRUNC('month', creation_date)::date,
       SUM(views_count)
FROM stackoverflow.posts
WHERE DATE_TRUNC('year', creation_date)::date = '2008-01-01'
GROUP BY 1
ORDER BY 2 DESC;
```

2.Выведите имена самых активных пользователей, которые в первый месяц после регистрации (включая день регистрации) в сумме по имени дали более 100 ответов. Вопросы, которые задавали пользователи, не учитывайте. Для каждого имени пользователя выведите количество уникальных значений `user_id`, соответствующих этому имени. Отсортируйте результат по полю с именами в лексикографическом порядке.

```
SELECT u.display_name,
       COUNT(DISTINCT user_id)
FROM stackoverflow.posts p
JOIN stackoverflow.post_types pt ON p.post_type_id = pt.id
JOIN stackoverflow.users u ON p.user_id = u.id
WHERE pt.type = 'Answer'
AND DATE_TRUNC('day', p.creation_date) >= DATE_TRUNC('day', u.creation_date)
AND DATE_TRUNC('day', p.creation_date) <= DATE_TRUNC('day', u.creation_date) +
INTERVAL '1 month'
GROUP BY u.display_name
HAVING COUNT(*) > 100
ORDER BY display_name;
```

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.

```
SELECT DATE_TRUNC('month', creation_date)::date as dt,
        COUNT(id) as cnt_posts
FROM stackoverflow.posts
WHERE user_id IN (
SELECT u.id
FROM stackoverflow.users u
JOIN stackoverflow.posts p ON p.user_id = u.id
WHERE DATE_TRUNC('month', u.creation_date) = '2008-09-01' AND
DATE_TRUNC('month', p.creation_date) = '2008-12-01')
GROUP BY 1
ORDER BY 1 DESC;
```

4.Используя данные о постах, выведите несколько полей:

- идентификатор пользователя, который написал пост;
- дата создания поста;
- количество просмотров у текущего поста;
- сумма 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.

```
SELECT user_id,
        creation_date,
        views_count,
        SUM(views_count) OVER(PARTITION BY user_id ORDER BY creation_date)
FROM stackoverflow.posts
GROUP BY 1,2,3
ORDER BY 1
```

5.Сколько в среднем дней в период с 1 по 7 декабря 2008 года включительно 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.

```sql
WITH t1 as (
SELECT DISTINCT user_id,
    COUNT(dt) OVER(PARTITION BY user_id) cnt_day
FROM
    (SELECT DISTINCT user_id,
         DATE_TRUNC('day', creation_date::date) as dt
FROM stackoverflow.posts
WHERE DATE_TRUNC('day', creation_date) BETWEEN '2008-12-01' AND '2008-12-07') as t1
)

SELECT ROUND(AVG(cnt_day)) as avg_dt
FROM t1
```

6.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`.

```sql
SELECT *,
       ROUND(((m_cnt::numeric / (LAG(m_cnt) OVER (ORDER BY dt))-1) * 100.0), 2) as
proc
FROM (SELECT DISTINCT EXTRACT(MONTH FROM creation_date::date) as dt,
             COUNT(id) OVER (PARTITION BY DATE_TRUNC('month', creation_date)::date)
as m_cnt
      FROM stackoverflow.posts
      WHERE creation_date::date BETWEEN '2008-09-01' AND '2008-12-31') as t1
```

7.Найдите пользователя, который опубликовал больше всего постов за всё время с момента регистрации. Выведите данные его активности за октябрь 2008 года в таком виде:

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

```
WITH t1 as (
SELECT user_id,
        COUNT(id) as cnt_post
FROM stackoverflow.posts
GROUP BY 1
ORDER BY 2 DESC
LIMIT 1),

t2 as (
    SELECT creation_date::timestamp as dt,
    EXTRACT(WEEK FROM creation_date) as dt_week
FROM stackoverflow.posts
WHERE user_id = (SELECT user_id FROM t1)
)

SELECT DISTINCT dt_week,
        MAX(dt) OVER(PARTITION BY dt_week) as max_post
FROM t2
WHERE dt BETWEEN '2008-10-01 00:00:00' AND '2008-10-31 23:59:59'
```