**Задача 1: Написание SQL-запросов**

Дана база данных с тремя таблицами:

1. \*\*Customers\*\* (Клиенты):

   - `customer\_id` (уникальный идентификатор клиента)

   - `name` (имя клиента)

   - `email` (электронная почта клиента)

   - `registration\_date` (дата регистрации клиента)

2. \*\*Orders\*\* (Заказы):

   - `order\_id` (уникальный идентификатор заказа)

   - `customer\_id` (идентификатор клиента, который сделал заказ)

   - `order\_date` (дата заказа)

   - `total\_amount` (общая сумма заказа)

3. \*\*Products\*\* (Товары):

  -order\_id (уникальный идентификатор заказа)

- product\_id` (уникальный идентификатор товара)

-product\_name` (название товара)

- price` (цена товара)

**Запросы написаны на PostgreSQL**

**1. Напишите SQL-запрос, который выведет список клиентов, сделавших заказы на сумму более 1000 единиц за последний месяц.**

**SELECT** c.customer\_id

**FROM** Customers **AS** c

**JOIN** Orders **AS** o **ON** c.customer\_id = o.customer\_id

**JOIN** Products **AS** p **ON** o.order\_id = p.order\_id

**WHERE**

**DATE\_TRUNC**(**"month"**, o.order\_date)**::date** =

***--- “::date” чтобы точно распознало дату***

**DATE\_TRUNC**(**"month"**, CURRENT\_DATE)**::date**

***--- только текущий месяц из условия***

**GROUP BY** c.customer\_id, o.order\_id

**HAVING** **SUM**(p.price) **> 1000**

**LIMIT 1000**

***--- ограничение выдачи, чтобы избежать перегрузки сервера в случае огромного кол-ва строк [показал заботу, дальше LIMIT без потребности не указываю]***

**;**

**2. Напишите SQL-запрос, который покажет общую сумму заказов для каждого клиента за последний год.**

**SELECT** c.customer\_id

, **SUM**(p.price) **AS** total\_amount

**FROM** Customers **AS** c

**JOIN** Orders **AS** o **ON** c.customer\_id = o.customer\_id

**JOIN** Products **AS** p **ON** o.order\_id = p.order\_id

**WHERE**

**DATE\_TRUNC**(**"year"**, o.order\_date)**::date** =

**DATE\_TRUNC**(**"year"**, CURRENT\_DATE)**::date**

**GROUP BY** c.customer\_id

**;**

**3. Напишите SQL-запрос, который выведет топ-5 самых популярных товаров (по количеству заказов) за последний квартал.**

**SELECT** p.product\_name

, **COUNT**(p.product\_id) **AS** product\_cnt

**FROM** Orders **AS** o

**JOIN** Products **AS** p **ON** o.order\_id = p.order\_id

**WHERE**

**DATE\_TRUNC**(**"quarter"**,o.order\_date)**::date** =

**DATE\_TRUNC**(**"quarter"**,CURRENT\_DATE)**::date**

**GROUP BY** p.product\_id

**ORDER BY** product\_cnt **DESC**

**LIMIT 5**

**;**

**4. Напишите SQL-запрос, который выведет клиентов, которые зарегистрировались, но не сделали ни одного заказа.**

**SELECT** **DISTINCT** c.customer\_id

**FROM** Customers **AS** c

**LEFT** **JOIN** Orders **AS** o **ON** c.customer\_id = o.customer\_id

**WHERE** o.order\_id **IS NULL**

**;**

**Задание 2: Оптимизация запроса**

Дан запрос:

```sql

SELECT u.name, COUNT(o.order\_id)

FROM users u

LEFT JOIN orders o ON u.user\_id = o.user\_id WHERE u.registration\_date > '2020-01-01'

GROUP BY u.name

ORDER BY COUNT(o.order\_id) DESC;

**1. Почему запрос может работать медленно?**

Причина первая, вытекающая из второго вопроса – отсутствие индексов не добавляет скорости, замедление которой начинает ощущаться при больших объемах строк и кол-ва таблиц. В запросе присутствует фильтрация по дате, что требует сканирования всей таблицы пользователей.

Вторая: LEFT JOIN соединение таблиц и COUNT(o.order\_id).

Текущий вариант предполагает, что нужно сопоставить каждого пользователя с его заказами, а т.к. используется LEFT JOIN, база данных ещё вынуждена пересчитывать количество заказов для каждого пользователя, включая случаи, когда заказов нет.

Если в таблице orders много строк или в таблице users много пользователей, это может привести к медленной работе запроса.

Третья: GROUP BY. После LEFT JOIN запрос выполняет группировку по u.name. Если пользователей много, то операция группировки также может занять значительное время.

Четвертая: выражение в ORDER BY без алиаса (псевдонима), а значит COUNT(o.order\_id) пересчитывается заново.

**2. Какие индексы нужно добавить для ускорения?**

2.1 Индекс на поле u.user\_id – это ускорит соединение между таблицами users и orders, т.к. соединение происходит по полю user\_id:

**CREATE INDEX** idx\_user\_id **ON** users(user\_id)

**;**

**CREATE INDEX** idx\_order\_user\_id **ON** orders(user\_id)

**;**

2.2 Индекс на поле u.registration\_date – это ускорит фильтрацию пользователей:

**CREATE INDEX** idx\_registration\_date **ON** users(registration\_date)

;

**3. Предложите оптимизированную версию запроса.**

3.1. **Добавить псевдоним** **полю** **с количеством заказов**, чтобы при сортировке была сразу ссылка на рассчитанное ранее поле.

3.2. Заменить **LEFT JOIN** на **INNER JOIN** (если нам важно учитывать только пользователей   
с заказами).

**SELECT** u.name

, **COUNT**(o.order\_id) **AS** order\_cnt

**FROM** users u

**JOIN** orders o **ON** u.user\_id = o.user\_id

**WHERE** u.registration\_date **> '2020-01-01'**

**GROUP BY** u.name

**ORDER BY** order\_cnt **DESC**