**ПЕРВОЕ ЗАДАНИЕ**

Монета весит 5 грамм, в этих 5 граммах 25% никеля. Значит, в этой монете 1.25 грамма никеля.

По условию сказано, что стоимость никеля перевалила за $100к/тонну. Примем, что тонна никеля стоит $100к. Рассчитаем стоимость никеля за грамм:

Тогда, в нашей монете никель стоит

При том условии, что мы не знаем точную стоимость никеля, так как в условии просто сказано, что цена «перевалила», то будем считать, что стоимость никеля за тонну строго больше $100к.

Ответ: больше $0.125. Зная точную стоимость никеля за тонну (пусть будет X), можно рассчитать стоимость никеля в монете по этой формуле:

**ВТОРОЕ ЗАДАНИЕ**

Использую PostgreSQL.

Из-за того, что у нас нет общей записи номеров, то придётся использовать регулярные выражения. Я буду использовать следующий алгоритм «преобразования» номеров телефонов к одному виду:

1. В каждом номере телефона оставить только цифры, после этого, у нас может быть два варианта записи телефона: 7… и 8…
2. Для решения проблемы первой цифры, мы просто её уберём и оставим в номере телефона 10 цифр.

После этих манипуляций, можно будет выбирать пользователей, у которых регистрация была на один номер.

Кроме этого, я решил добавить дополнительный столбец, который будет содержать в себе id номера. То есть все аккаунты, заведённые с одного номера телефона, будут иметь одинаковый id.

Реализация без оконных функций:

SELECT  
 a.acc,  
 a.name,  
 a.email,  
 a.phone,  
 t.phone\_group\_id  
FROM  
 accounts a  
JOIN (  
 SELECT  
 normalized\_phone,  
 *MIN*(acc) AS representative\_acc,  
 *ROW\_NUMBER*() OVER (ORDER BY *MIN*(acc)) AS phone\_group\_id   
 FROM (  
 SELECT acc, substring(*regexp\_replace*(phone, '[^0-9]', '', 'g'), 2) AS normalized\_phone  
 FROM accounts  
 ) AS normalized\_accounts  
 GROUP BY normalized\_phone  
 HAVING COUNT(\*) > 1 *-- Оставляем только дублирующиеся телефоны*) t ON substring(*regexp\_replace*(a.phone, '[^0-9]', '', 'g'), 2) = t.normalized\_phone  
ORDER BY t.phone\_group\_id, a.acc;

Реализация с оконными функциями:

SELECT  
 acc,  
 name,  
 email,  
 phone,  
 phone\_id  
FROM (  
 SELECT  
 acc,  
 name,  
 email,  
 phone,  
 *DENSE\_RANK*() OVER (ORDER BY substring(*regexp\_replace*(phone, '[^0-9]', '', 'g'), 2)) AS phone\_id  
 FROM accounts  
) AS subquery  
WHERE phone\_id IN (SELECT phone\_id  
 FROM  
 (SELECT DENSE\_RANK() OVER  
 (ORDER BY substring(regexp\_replace(phone, '[^0-9]', '', 'g'), 2)) AS phone\_id,  
 substring(regexp\_replace(phone, '[^0-9]', '', 'g'), 2) AS normalized\_phone, COUNT(\*)  
 FROM accounts GROUP BY normalized\_phone HAVING COUNT(\*) > 1) AS dupes);

Вывод для таблицы из задания:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Acc | Name | Email | Phone | Phone\_id |
| 2 | Bob | bob@example.com | +79167654321 | 1 |
| 4 | Dylan | dylan@example.com | +79167654321 | 1 |
| 5 | Eve | eve@example.com | +79167654321 | 1 |
| 3 | Charlie | ch@example.com | 8(985)123-45-67 | 2 |
| 6 | Frank | frank@example.com | +79851234567 | 2 |