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:

$$\frac{100000}{1000000} = \$0.1/1 \text{ грамм}$$

Тогда, в нашей монете никель стоит $0.1 * 1.25 = \$0.125$

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:
$\frac{X}{1000000} * 1.25$

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