**Часть 1. SQL**

ALTER TABLE test\_1 ADD result INT;

DECLARE @i INT, @res INT, @tmp INT

SET @res = 0

SELECT @i = MIN(years) FROM test\_1

WHILE (@i IS NOT NULL)

BEGIN

    SELECT @tmp = accr FROM test\_1 WHERE years = @i

    IF @tmp >= @res

        SET @res = @res + @tmp

    ELSE

        SET @res = @res

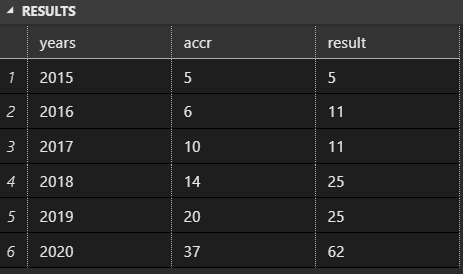
    UPDATE test\_1 SET result = @res WHERE years = @i

    SELECT @i = MIN(years) FROM dbo.test\_1 WHERE years > @i

END

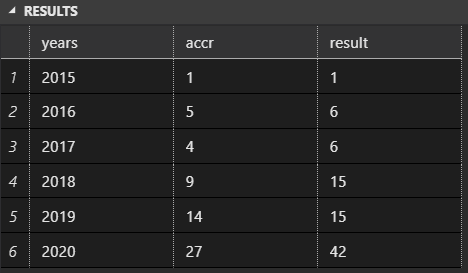
SELECT \* FROM test\_1;

1. Имеется таблица с начислениями за каждый год:



|  |  |  |
| --- | --- | --- |
| year | accrual | result |
| 2015 | 5 | 5 |
| 2016 | 6 | 11 |
| 2017 | 10 | 11 |
| 2018 | 14 | 25 |
| 2019 | 20 | 25 |
| 2020 | 37 | 62 |

Необходимо посчитать величину начисления нарастающим итогом в порядке увеличения года. При этом если начисление текущего года меньше нарастающего итога предыдущих лет, то этот год не вносит свой вклад в нарастающий итог. Т.е. результат будет:



|  |  |  |
| --- | --- | --- |
| year | accrual | result |
| 2015 | 1 | 1 |
| 2016 | 5 | 6 |
| 2017 | 4 | 6 |
| 2018 | 9 | 15 |
| 2019 | 14 | 15 |
| 2020 | 27 | 42 |

2. Необходимо написать запрос, который по таблице с интервалами объединит эти интервалы в случае если они пересекаются или граничат.

DECLARE @i DATE, @counter INT, @change\_grp INT,

@counter\_lim INT, @group\_num INT, @cur\_grp INT;

DECLARE @tmp\_tb\_2 TABLE

(

   dates DATE,

   group\_num INT

);

-- Make group labels

WITH tmp\_tb\_1 AS

    (SELECT start\_dates, end\_dates,

            ROW\_NUMBER() OVER(ORDER BY start\_dates) AS group\_num

        FROM test\_2)

-- To timeline

    INSERT INTO @tmp\_tb\_2

    SELECT start\_dates as dates, group\_num FROM tmp\_tb\_1

    UNION

    SELECT end\_dates as dates, group\_num FROM tmp\_tb\_1

-- Loop

SET @counter = 0

SET @counter\_lim = 2

SELECT @i = MIN(dates) FROM @tmp\_tb\_2

SELECT @change\_grp = group\_num FROM @tmp\_tb\_2 WHERE dates = @i

SET @cur\_grp = @change\_grp

WHILE (@i IS NOT NULL)

BEGIN

    SELECT @group\_num = group\_num FROM @tmp\_tb\_2 WHERE dates = @i

    IF @change\_grp != @group\_num

    BEGIN

        UPDATE @tmp\_tb\_2 SET group\_num = @change\_grp WHERE group\_num = @group\_num

        SET @counter = @counter + 1

        SET @counter\_lim = @counter\_lim + 2

        SET @cur\_grp = @cur\_grp + 1

    END

    ELSE

    BEGIN

        SET @counter = @counter + 1

    END

    IF @counter\_lim = @counter

    BEGIN

        SET @cur\_grp = @cur\_grp + 1

        SET @change\_grp = @cur\_grp

        SET @counter = 0

        SET @counter\_lim = 2

        SELECT @i = MIN(dates) FROM @tmp\_tb\_2 WHERE group\_num = @change\_grp

    END

    ELSE

    BEGIN

        SELECT @i = MIN(dates) FROM @tmp\_tb\_2 WHERE dates > @i

    END

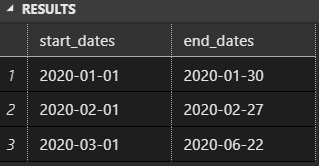
END

SELECT MIN(dates) AS start\_dates, MAX(dates) AS end\_dates

FROM @tmp\_tb\_2 GROUP BY group\_num;

Исходные данные:

|  |  |
| --- | --- |
| start\_date | end\_date |
| 2020-01-01 | 2020-01-15 |
| 2020-01-13 | 2020-01-30 |
| 2020-02-01 | 2020-02-05 |
| 2020-02-15 | 2020-02-27 |
| 2020-02-03 | 2020-02-16 |
| 2020-03-01 | 2020-05-04 |
| 2020-06-12 | 2020-06-22 |
| 2020-04-17 | 2020-06-19 |

Результат: (Видимо ошибка в примере результата, т.к. присутствуют две идентичных строки).

|  |  |
| --- | --- |
| start\_date | end\_date |
| 2020-01-01 | 2020-01-30 |
| 2020-02-01 | 2020-02-27 |
| 2020-02-01 | 2020-02-27 |
| 2020-03-01 | 2020-06-22 |

|  |  |
| --- | --- |
| start\_date | end\_date |
| 2020-01-01 | 2020-01-30 |
| 2020-02-01 | 2020-02-27 |
| 2020-03-01 | 2020-06-22 |

3. Имеется таблица с атрибутами пользователей:

USE test\_db\_2019;

DECLARE @i INT, @j INT, @k INT, @grp\_id INT, @grp\_def VARCHAR(MAX);

DECLARE @result\_tb TABLE

(

   grp INT,

   user\_id INT,

   attributes VARCHAR(250)

);

DECLARE @tmp\_tb\_1 TABLE

(

   id INT NOT NULL IDENTITY(1, 1),

   user\_id INT

);

DECLARE @tmp\_tb\_2 TABLE

(

   id INT NOT NULL IDENTITY(1, 1),

   attribute VARCHAR(50)

);

SET @grp\_id = 1;

SET @k = 1;

SELECT @i = MIN(user\_id) FROM test\_4;

WHILE (@i IS NOT NULL)

BEGIN

    INSERT INTO @tmp\_tb\_1 (user\_id) VALUES (@i);

    SELECT @j = MIN(id) FROM @tmp\_tb\_1;

    WHILE (@j IS NOT NULL)

    BEGIN

        INSERT INTO @tmp\_tb\_2 (attribute)

            SELECT attribute FROM test\_4

                WHERE user\_id = (SELECT user\_id FROM @tmp\_tb\_1 WHERE id = @j)

                EXCEPT SELECT attribute FROM @tmp\_tb\_2;

        WHILE ((SELECT attribute FROM @tmp\_tb\_2

                WHERE id = @k) IS NOT NULL)

        BEGIN

            INSERT INTO @tmp\_tb\_1 (user\_id)

                SELECT DISTINCT user\_id FROM test\_4

                WHERE attribute

                IN (SELECT attribute FROM @tmp\_tb\_2

                WHERE id = @k)

                EXCEPT (SELECT user\_id FROM @tmp\_tb\_1)

            SET @k = @k + 1

        END

        SELECT @j = MIN(id) FROM @tmp\_tb\_1 WHERE id > @j;

    END

    SELECT @grp\_def = STRING\_AGG(attribute, ', ') FROM @tmp\_tb\_2;

    INSERT INTO @result\_tb (user\_id) SELECT user\_id FROM @tmp\_tb\_1;

    UPDATE @result\_tb SET grp = @grp\_id,

        attributes = @grp\_def

        WHERE user\_id IN (SELECT user\_id FROM @tmp\_tb\_1);

    SET @grp\_id = @grp\_id + 1;

    SELECT @i = MIN(user\_id) FROM test\_4

        WHERE (user\_id > @i) AND user\_id NOT IN

        (SELECT user\_id FROM @result\_tb);

    DELETE FROM @tmp\_tb\_1;

    DELETE FROM @tmp\_tb\_2;

END

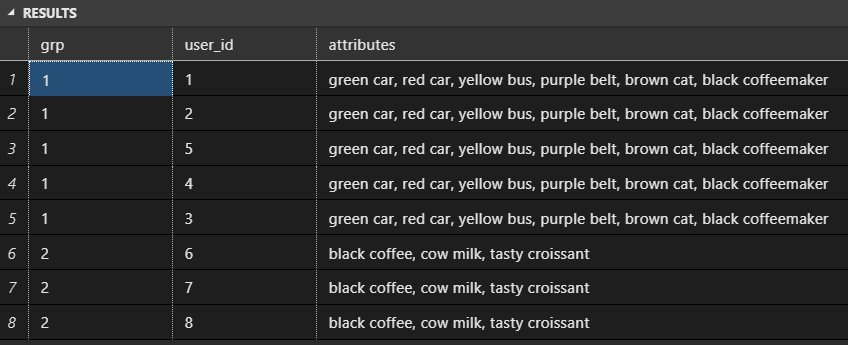
SELECT \* FROM @result\_tb;

|  |  |
| --- | --- |
| user\_id | attribute |
| 1 | "red car" |
| 1 | "green car" |
| 2 | "green car" |
| 2 | "yellow bus" |
| 3 | "brown cat" |
| 3 | "black coffeemaker" |
| 4 | "brown cat" |
| 4 | "purple belt" |
| 5 | "purple belt" |
| 5 | "red car" |
| 6 | "black coffee" |
| 7 | "black coffee" |
| 7 | "cow milk" |
| 8 | "cow milk" |
| 8 | "tasty croissant" |

Необходимо написать запрос который объединит пользователей в группы по признаку обладания общими атрибутами с выводом списка общих атрибутов для группы (порядок не важен). Если два пользователя не имеют общего атрибута, но попадают в одну группу с одним и тем же третьим пользователем, всех троих нужно объединить в одну группу.

Результат:

|  |  |  |
| --- | --- | --- |
| group\_id | user\_id | attribute |
| 1 | 1 | "red car, green car, yellow bus, brown cat, purple belt, black coffeemaker" |
| 1 | 2 | "red car, green car, yellow bus, brown cat, purple belt, black coffeemaker" |
| 1 | 3 | "red car, green car, yellow bus, brown cat, purple belt, black coffeemaker" |
| 1 | 4 | "red car, green car, yellow bus, brown cat, purple belt, black coffeemaker" |
| 1 | 5 | "red car, green car, yellow bus, brown cat, purple belt, black coffeemaker" |
| 2 | 6 | " black coffee, cow milk, tasty croissant" |
| 2 | 7 | " black coffee, cow milk, tasty croissant" |
| 2 | 8 | " black coffee, cow milk, tasty croissant" |



**Часть 2. a/b- тестирование.**

1. Компании поставлена цель – сильно вырастить MAU. При всей клевости менеджеров по закупке трафика, при текущем Retention бюджета для достижения цели недостаточно.

Для помощи своей команде, ты решил(-а) провести эксперимент - запустить look-alike кампанию на Facebook, чтобы привлечь более качественную аудиторию. Под качеством аудитории мы тут понимаем пользователей, которые продолжают пользоваться приложением спустя месяц (1 Month Retention = 1). Самый большой Churn происходит у пользователей в 0 месяц, поэтому можно грубо ориентироваться только на 1 Month Retention.

Эксперимент ты решил(-а) провести A/B-тестом – запустить один и тот же креатив на кампанию с использованием look-alike алгоритмов Facebook и тестовую группу без использования дополнительных инструментов, только обычная закупка.

Ты сделал(-а) выборку пользователей с высоким Retention, выгрузил(-а) и передал(-а) менеджерам по закупке трафика. Теперь надо определиться с размером аудитории, которую им нужно привлечь, чтобы ты смог(-ла) оценить результаты эксперимента.

Вот тут входящие данные

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Платформа | Количество показов | Конверсия из показа в установку | CPI | 1 Month Retention | Cost Per Active User |
| Android | 1 000 000 | 0,45% | 34р | 44% | 77,3р |
| iOS | 1 000 000 | 0,31% | 38р | 55% | 69,1р |

1. **Достаточно ли данных для проведения теста? Задай максимум 3 уточняющих вопроса, если они появились.**
   1. Какого уровня 1 Month Retention хотим достичь?
   2. Или какого % уменьшения Cost Per Active User хотим достичь?
2. **Посчитай стоимость активного пользователя**

1 000 000 \* 0,0045 \* 34 / (1 000 000 \* 0,0045 \* 0,44) =153 000 / 1 980 = 77,3 р

1 000 000 \* 0,0031 \* 38 / (1 000 000 \* 0,0031 \* 0,55) =117 800 / 1 705 = 69,1 р

1. **Реши, конверсию из чего во что ты возьмешь для измерения результатов теста. Обоснуй свой выбор. Считай, что экспериментальная группа сделает его на 5% больше.**

Возьму конверсию из людей, установивших приложение в людей, которые пользуется приложением хотя бы раз в месяц (в нашем случае 1 Moth Retention). Так как метрика, на которую мы ориентируемся MAU.

1. **Посчитай размер аудитории на каждую кампанию, которую менеджерам по трафику необходимо привлечь.**

Достоверность 95%, мощность 80%:

* + - Android
      * 1% - По 38 680 человек
      * 2% - По 9 670 человек
      * 5% - По 1 547 человек
    - iOS
      * 1% - По 38 853 человек
      * 2% - По 9 713 человек
      * 5% - По 1 554 человек

1. **Посчитай бюджет на эксперимент и размер экономии, если перевести весь Facebook на look-alike (10 000 установок в месяц)**

Бюджет на эксперимент при предполагаемом увеличении конверсии из установок в 1 Month Retention на 5%:

* + - Android: 1 547 \* 2 \* 34 = 105 196 р
    - iOS: 1 554 \* 2 \* 38 = 118 104 р
    - Итого: 223 300 р

При переводе всего Facebook на look-alike экономия (в Cost Per Active User) составит:

* + - Android:
      * Cost Per Active User: 34 / 0,49 = 69,4 р
      * Абсолютная экономия: 77,3 -69,4 = 7,9 р
      * Относительная экономия: 7,9 / 77,3 \* 100% = 10,2 %
    - iOS: 38 / 0,6= 63,3 р
      * Cost Per Active User: 38 / 0,6 = 63,3 р
      * Абсолютная экономия: 69,1 -63,3 = 5,8 р
      * Относительная экономия: 5,8 / 69,1 \* 100% = 8,4 %

Нам важно услышать твои рассуждения!

2. Ниже 2 задания. Оба задания – продолжение эксперимента из первой части. Данные для них лежат в архиве (см. аттач)

Часть 1.

Для эксперимента, цель которого описана в предыдущем задании, тебе надо подготовить базу новых пользователей, у которых по прогнозу будет высокая активность. Эту базу ты передашь менеджерам по закупке трафика, чтобы они использовали ее в look-alike кампании. Ты решаешь взять именно новых пользователей, потому что они наилучшим образом отражают и целевую аудиторию для закупки здесь и сейчас, и поведение аудитории на последних версиях приложения обеих платформ (ОС).

У тебя есть 2 датасэта:

1. Данные по активности в приложении рандомной выборки пользователей за период с 2018-01-01 по 2020-03-26 для обучения модели
2. Данные по активности в приложении новых пользователей за период с 2020-03-01 по 2020-03-25 для прогнозирования

С помощью Python спрогнозируй для второй выборки активность минимум на 15 дней (без value, только life time). При обучении модели смотри на достоверность и % ошибки. Выбери топ20% пользователей по прогнозной активности. Для них ты планируешь выгрузить идентификаторы для look-alike кампании.

Результаты предоставь в таком виде:

* График, на котором будет видно фактическую активность пользователей и ту, которую предсказывает модель. Он покажет визуально, насколько хорошо модель обучилась
* На какой период прогнозирует модель
* Ошибка прогноза в количестве событий
* Достоверность прогноза в количестве активных пользователей
* Notebook или в любом другом виде код

Часть 2.

Пришло время подвести итоги эксперимента. У тебя есть данные для обеих групп (тестовой и контрольной) по:

* факту возвращения пользователя в приложение через месяц после его первой сессии (1 Month Retention = 1/0)
* суммарному Revenue пользователя

Для упрощения считаем, что все пользователи пришли примерно в одно время.

Посчитай, какая из двух групп победила по каждому показателю раздельно. С какой вероятностью. Предоставь код в любом удобном для тебя виде.