**ОРГАНИЗАЦИОННАЯ ИНФОРМАЦИЯ**

**Чек-лист проверки:**

* ***Проверьте, правильно ли вообще решена задача***
* ***Посмотрите — возможно можно решить задачу как-то поизящней?***

Распишите, что человек сделал не так, где можно было оптимальней, как его решение можно улучшить.

* ***Финально проверьте - хорошо ли отформатирован код?***

Отступы, пробелы, расстановка запятых и так далее.

**ВАЖНО:** *Нужно делать ревью так, чтобы студент, который потом будет его читать, понял ваши замечания и принял их. Это важно - иначе код-ревью превращается просто в негатив и отрицание (так часто, к сожалению, встречается в больших компаниях).*

**Пример проверки:**

select email,

score,

case when score >= 500 then 'A'

when score >= 100 then 'B'

when score >= 20 then 'C'

when score >= 0 then 'D'

END as class

from users

Комментарий ревьюера:

Задача решена верно. Однако, код можно улучшить.

Во-первых, лучше сначала расположить более частые группы. Это снизит количество проверок, т.к. оператор CASE не будет проверять условия дальше, если нашел совпадение. Очевидно, что группа D чаще группы C, группа С - группы В и так далее.

Во-вторых, нет смысла делать последнюю проверку - это только увеличивает количество проверок. Если первые три не прошли, то это точно группа D (в исходном варианте).

Пример измененного кода:

select

email,

score,

case

when score < 20 then 'D'

when score < 100 then 'C'

when score < 500 then 'B'

else 'A'

end as “class”

from users

Что касается форматирования:

* Добавил отступы для разных уровней, перенес на разные строки в зависимости от оператора
* Убрал лишние пробелы
* Зарезервированное слово “class” взял в кавычки
* Привел все в один регистр

***Переходим к задачам ===>***

*Скопируйте себе этот файл и пишите свои комментарии и правки прям под каждой задачей. Не удаляйте исходные данные!*

**ЗАДАЧА 1**

***Формулировка***

Сколько монет в среднем списывает пользователь за весь срок жизни? Сколько монет ему начисляется? Какая в среднем по пользователям разница между начислениями и списаниями?

Примечание: В расчетах “среднего” учитывайте только тех пользователей, у которых в целом были какие-то транзакции.

В расчетах исключайте всегда пользователей с id < 94 - это наши внутренние аккаунты. Когда будете работать с таблицей Транзакций, не берите в расчет транзакции больше или равные 500 монетам - это начисления бета-тестерам,

они будут сильно мешать.

***Решение студента:***

with A as (select sum(t.value) as sum\_value,

t.user\_id

from transaction t

where t.type\_id in(2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,29) --id начислений

and t.value<=500

and t.user\_id>=94

group by t.user\_id

), --начисление по каждому юзеру

A1 as (select round (sum(A.sum\_value)/count(A.user\_id), 2) as среднее\_начисление

from A), /\*вывод среднего начисления \*/

B as (select round(sum(t.value)\*1.0/count(A.user\_id ), 2) as среднее\_списание

from A

left join transaction t on t.user\_id =A.user\_id

where t.type\_id in(1,23,24,25,26,27,28) --id списаний

and t.value<=500

and t.user\_id>=94

) /\*вывод среднего списания \*/

select A1.среднее\_начисление,

B.среднее\_списание,

A1.среднее\_начисление-B.среднее\_списание as дельта

from A1, B

Задача решена верно в части начислений и списаний, а вот по разнице ошибка, т.к. требуется вычислить среднее значение разницы, а не разницу между средним значением начислений и списаний.

Кроме того, сам код можно улучшить.

1. Вычисление начислений, списаний и разницы между ними можно сделать в одной таблице, а затем только вывести среднее значение начислений, списаний и разниц между начислением и списанием.
2. Интервал лучше указать не в виде перечисления, а с помощью конструкции between, что сделает код более читаемым.
3. Название таблиц и столбцов в них можно сделать интуитивно понятными, что поможет нам понять код по истечении времени.

Я бы задачу решила следующим образом:

**with *balance* as(**

**select**

***t*.user\_id,**

**sum(case**

**when *t*.type\_id = 1 or *t*.type\_id between 23 and 29 then *t*.value**

**end) *write\_off*,**

**sum(case**

**when *t*.type\_id = 29 or *t*.type\_id between 2 and 22 then *t*.value**

**end) *accruls*,**

**sum(case**

**when *t*.type\_id = 1 or *t*.type\_id between 23 and 29 then *t*.value \* (-1) else *t*.value**

**end) *balance***

**from**

**"transaction" *t***

**where**

***t.user\_id>=94***

**and *t*.value < 500**

**group by**

***t*.user\_id)**

**select**

**round(avg(*write\_off*),**

**2) as *write\_off*,**

**round(avg(*accruls*),**

**2) as *accruls*,**

**round(avg(*balance*),**

**2) as *balance***

**from**

***balance***

**ЗАДАЧА 2**

***Формулировка***

Нужно посчитать количество купленных задач и тестов (опираемся на таблицу Транзакций). Интересно, сколько в сумме купили закрытые задачи и тесты (отдельно - задачи, отдельно - тесты), а также в среднем на 1 пользователя платформы (даже если он вообще не проявлял никакой активности) - также отдельно по задачам и тестам. Также стоит посмотреть, сколько людей купили хотя бы 1 задачу/тест, а сколько решали только бесплатные. Дополнительно выведите, сколько людей вообще не решали ни одной задачи, а также не делали ни одной попытки пройти тест.

В расчетах исключайте всегда пользователей с id < 94 - это наши внутренние аккаунты. При анализе задач также исключайте из расчета те задачи, которые присутствуют в problem\_to\_company и имеют отношение к компании пользователя, решавшего задачу - это домашние задания для студентов, они тоже искажают статистику.

Например, если у пользователя 10 компания и он решает задачу с номером 300, а в таблице problem\_to\_company эта задача фигурирует как домашняя для компании 10 - ее нужно исключить.

Ответ можете сформировать в виде таблицы из 1 строки и 10 столбцов, а можете организовать 2 строки (одна для тестов, другая - для задач) и сократить число столбцов до 5.

***Решение студента***

with A as (

--выводим расчеты по задачам

select (select count(\*)

from codesubmit c

where c.user\_id>=94

) - count( t.id) as open , --кол-во открытых задач

count(distinct t.user\_id) as count\_user, --кол-во людей купивших хотя бы 1 задачу

count( t.id) as count\_closed, --сумма купленных закрытых задач

round(count(t.id)\*1.0/count(distinct t.user\_id),2) as avg\_count\_closed, --среднее кол-во задач на пользователя

(select count(distinct c.user\_id)

from codesubmit c

where c.user\_id>=94

) - count(distinct t.user\_id) as count\_user\_open --количество пользователей, решавших только бесплатные задачи

from transaction t

where t.type\_id =23 --покупка задач

and t.user\_id>=94

),

B as (

--выводим расчеты по тестам

select (select count(\*)

from teststart t2

where t2.user\_id>=94

) - count( t.id) as open, --кол-во открытых тестов

count(distinct t.user\_id) as count\_user, --кол-во людей купивших хотя бы 1 тест

count( t.id) as count\_closed, --сумма купленных закрытых тестов

round(count(t.id)\*1.0/count(distinct t.user\_id),2) as avg\_count\_closed, --среднее кол-во тестов на пользователя

(select count(distinct t2.user\_id)

from teststart t2

where t2.user\_id>=94

) - count(distinct t.user\_id) as count\_user\_open --количество пользователей, решавших только бесплатные тесты

from transaction t

where t.type\_id =27 --покупка теста

and t.user\_id>=94)

--сводим результаты

select 'problem',\*

from A

union

select 'test',\*

from B

**ЗАДАЧА 3**

***Формулировка***

Как связана дата захода на платформу и активность пользователя. Под активностью имеется в виду попытка решить задачу/тест. Надо посмотреть - какой % заходов не сопровождается активностью (считаем заходы в разрезе дня: например, если в один день заходил и решал задачи, считаем, что заход сопровождался активностью).

В расчетах исключайте всегда пользователей с id < 94 - это наши внутренние аккаунты.

***Решение студента***

with A as (select u.user\_id ,

u.entry\_at,

ROW\_NUMBER() OVER(PARTITION BY u.user\_id ORDER BY u.entry\_at) as num /\*нумерация в пределах группы (u.user\_id) для таблицы В\*/

from userentry u

where u.user\_id >=94

order by 1

),

B as (select A.user\_id,

A.entry\_at as entry\_start ,

case when A1.entry\_at is not null then A1.entry\_at

else CURRENT\_TIMESTAMP end as entry\_end

from A

left join A as A1 on A1.user\_id=A.user\_id

and A1.num=A.num+1

), /\*вывод даты захода на платформу - entry\_start и даты следующего захода для каждого юзера - (период, в пределах которого будет рассчитана активность),если следующего захода нет-выводим текущую дату\*/

C as (select t.user\_id ,

t.created\_at

from teststart t

where t.user\_id >=94 /\* попытки решить тесты\*/

union

select c.user\_id ,

c.created\_at

from codesubmit c

where c.user\_id >=94

order by 1,2 /\* попытки решить задачи\*/

), /\* попытки решить тесты и задачи\*/

D as (select B.user\_id,

B.entry\_start,

B.entry\_end,

count(C1.created\_at) /\*кол -во попыток решить задачи или тесты в период entry\_start-entry\_end\*/

from B

left join C as C1 on C1.user\_id=B.user\_id

and C1.created\_at>B.entry\_start

and C1.created\_at<B.entry\_end

group by B.user\_id,

B.entry\_start,

B.entry\_end

having count(C1.created\_at)=0 /\*вывод заходов на платформу не сопровождающихся активностью\*/

),

D1 as (select count (\*)

from D

), /\*количество заходов на платформу не сопровождающихся активностью\*/

D2 as (select count(A.entry\_at)

from A

) /\*общее количество заходов на платформу\*/

select round (D1.count\*1.0/D2.count\*100, 0)

from D1, D2

Ответ в задаче неверный, потому что мы не учли решения задач в таблице coderun, ведь, пользователь мог только запустить решение, чтобы посмотреть таблицу, но не отправить решение на проверку, а это уже активность.

Кроме того, представляется излишним вычисление разницы между первым заходом на платформу пользователя и его последующим заходом, чтобы определить день его захода. Для определения дня захода пользователя достаточно изменить формат временных ячеек до дня, например, с помощью такой конструкции **userentry.entry\_at::date**.

Могу предложить реализацию решения задачи в несколько иной логике – сначала найти всех активных пользователей по дням, затем найти всех пользователей, зашедших на платформу также по дням, объединить обе таблицы, причем, по таблице зашедших пользователей, - логично, что этот список более полный и для решения тестов и задач, и для простого просмотра страниц на сайте, пользователь должен зайти на платформу. В новой таблице у нас будут столбцы по активным пользователям и по всем пользователям, зашедшим на платформу, дальше уже дело техники – рассчитать долю неактивных пользователей.

Кроме того, таблицы и столбцы лучше называть понятными именами, чтобы самим не запутаться, прежде всего.

Могу предложить такое решение:

**with** *users\_created* **as**(

**select**

created\_at::date **as** *date\_created*,

user\_id **as** *user\_created*

**from**

codesubmit

**where** u.user\_id >=94

**union**

**select**

created\_at::date,

user\_id

**from**

coderun

**where** u.user\_id >=94

**union**

**select**

created\_at::date,

user\_id

**from**

teststart

**where** u.user\_id >=94

),

*user\_entry* **as**(

**select**

entry\_at::date **as** *date\_entry*,

user\_id **as** *user\_entry*

**from**

userentry

**where** u.user\_id >=94

),

*total\_users* **as**(

**select**

\*

**from**

*user\_entry*

**left** **join** *users\_created*

**on**

*user\_entry* = *user\_created*

**and** *date\_entry* = *date\_created*

),

*all\_users* **as**(

**select**

*date\_entry*,

**count**(**distinct** *user\_created*) **as** *cnt\_created*,

**count**(**distinct** *user\_entry*) **as** *cnt\_entry*

**from**

*total\_users*

**group** **by**

*date\_entry*

)

**select**

-(**round**(**sum**(*cnt\_created*) \* 100.0 / **sum**(*cnt\_entry*),

2) - 100) **as** *entries\_without\_activities*

**from**

*all\_users*