**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

**Рекомендации:**

1 - Лучше использовать английские названия для таблиц/столбцов и CTE — так код будет понятнее и более универсален.  
2 - Имена CTE стоит делать осмысленными.  
3 - Там, где идут длинные условия с перечислениями, можно заменить на BETWEEN — получится короче и читаемее.  
4 - CTE A1 и B можно объединить, чтобы убрать дублирование.  
5 - Для читаемости стоит добавлять пробелы вокруг знаков = и >=.  
6 - При расчётах пользователей нужно использовать COUNT(DISTINCT user\_id), чтобы не учитывать одного человека несколько раз.

**Измененный код:**

**with** *user\_transactions* **as** (

**select**

user\_id,

**sum**(**case** **when** type\_id **between** 2 **and** 22 **or** type\_id = 29 **then** value **end**) **as** *total\_income*,

**sum**(**case** **when** type\_id **between** 23 **and** 28 **or** type\_id=1 **then** value **end**) **as** *total\_outcome*

**from** transaction

**where** value <= 500

**and** user\_id >= 94

**and** type\_id **between** 1 **and** 29

**group** **by** user\_id

),

*average\_calculations* **as** (

**select**

**round**(**sum**(*total\_income*) \* 1.0 / **count**(**distinct** user\_id), 2) **as** *avg\_income*,

**round**(**sum**(*total\_outcome*) \* 1.0 / **count**(**distinct** user\_id), 2) **as** *avg\_outcome*

**from** *user\_transactions*

)

**select**

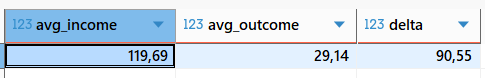
*avg\_income*,

*avg\_outcome*,

*avg\_income* - *avg\_outcome* **as** *delta*

**from** *average\_calculations*;

**Результат:**



**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.   
  
Этой задачи уже нет в симуляторе - она была в старой версии, но мы ее убрали. Поэтому вот небольшие пояснения перед задачей:

За покупку задач в таблице транзакций отвечает type\_id 23, за покупку тестов - 27. Такие задачи считаются закрытыми, и не доступны до покупки. Все остальные задачи (хранящиеся в codesubmit, coderun и problem\_to\_company) - считаем открытыми.

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

**Рекомендации:**

1 - Лучше использовать английские названия для CTE и столбцов.

2 - Подзапросы в SELECT стоит выносить в отдельные CTE.

3 - При подсчёте открытых задач важно учитывать обе таблицы (codesubmit и coderun).

4 - Базовый CTE с пользователями помогает правильно считать средние значения по всей выборке, а не только по платящим.

5 - Названия агрегатов (open\_users, avg\_per\_paying\_user, avg\_per\_all\_users) лучше делать осмысленными.

6 - Для объединения выборок стоит использовать UNION ALL.

7 - Для читаемости кода ставить пробелы вокруг операторов (=, >=, /).

**Изменённый код:**

**with** *paid\_activity* **as** (

**select**

user\_id,

**count**(**case** **when** type\_id = 23 **then** 1 **end**) **as** *bought\_problems*,

**count**(**case** **when** type\_id = 27 **then** 1 **end**) **as** *bought\_tests*

**from** "transaction" *t*

**where** user\_id >= 94

**group** **by** user\_id

),

*free\_problem\_attempts* **as** (

**select** *c*.user\_id, *c*.problem\_id

**from** coderun *c*

**join** users *u* **on** *c*.user\_id = *u*.id

**left** **join** problem\_to\_company *ptc*

**on** *c*.problem\_id = *ptc*.problem\_id **and** *u*.company\_id = *ptc*.company\_id

**where** *ptc*.company\_id **is** **null**

**union**

**select** *c*.user\_id, *c*.problem\_id

**from** codesubmit *c*

**join** users *u* **on** *c*.user\_id = *u*.id

**left** **join** problem\_to\_company *ptc*

**on** *c*.problem\_id = *ptc*.problem\_id **and** *u*.company\_id = *ptc*.company\_id

**where** *ptc*.company\_id **is** **null**

),

*problem\_summary* **as** (

**select** user\_id, **count**(problem\_id) **as** *total\_problems*

**from** *free\_problem\_attempts*

**group** **by** user\_id

),

*test\_summary* **as** (

**select** user\_id, **count**(**distinct** test\_id) **as** *total\_tests*

**from** teststart *t*

**group** **by** user\_id

)

**select**

**sum**(**coalesce**(*pa*.*bought\_problems*,0)) **as** *total\_paid\_problems*,

**sum**(**coalesce**(*pa*.*bought\_tests*,0)) **as** *total\_paid\_tests*,

**sum**(**coalesce**(pa.bought\_problems,0))::**numeric** / **count**(*u*.id) **as** *avg\_paid\_problems*,

**sum**(**coalesce**(pa.bought\_tests,0))::**numeric** / **count**(*u*.id) **as** *avg\_paid\_tests*,

**sum**(**case** **when** **coalesce**(*pa*.*bought\_problems*,0) >= 1 **then** 1 **else** 0 **end**) **as** *users\_with\_paid\_problems*,

**sum**(**case** **when** **coalesce**(*pa*.*bought\_tests*,0) >= 1 **then** 1 **else** 0 **end**) **as** *users\_with\_paid\_tests*,

**sum**(**case** **when** **coalesce**(*ps*.*total\_problems*,0) >= 1 **then** 1 **else** 0 **end**)

- **sum**(**case** **when** **coalesce**(*pa*.*bought\_problems*,0) >= 1 **then** 1 **else** 0 **end**) **as** *users\_only\_free\_problems*,

**sum**(**case** **when** **coalesce**(*ts*.*total\_tests*,0) >= 1 **then** 1 **else** 0 **end**)

- **sum**(**case** **when** **coalesce**(*pa*.*bought\_tests*,0) >= 1 **then** 1 **else** 0 **end**) **as** *users\_only\_free\_tests*,

**count**(**case** **when** *ps*.*total\_problems* **is** **null** **then** 1 **end**) **as** *users\_with\_no\_problems*,

**count**(**case** **when** *ts*.*total\_tests* **is** **null** **then** 1 **end**) **as** *users\_with\_no\_tests*

**from** users *u*

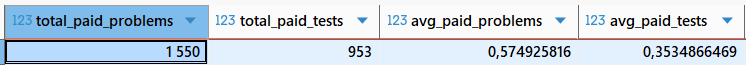
**left** **join** *problem\_summary* *ps* **on** *ps*.user\_id = *u*.id

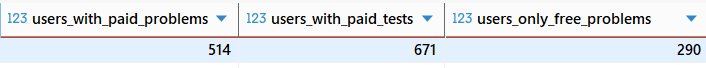
**left** **join** *test\_summary* *ts* **on** *ts*.user\_id = *u*.id

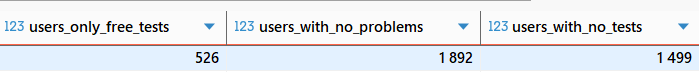
**left** **join** *paid\_activity* *pa* **on** *u*.id = *pa*.user\_id

**where** *u*.id >= 94;

**Результат:**







**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

**Рекомендации:**

1 - Нужно учитывать все источники активности (coderun, codesubmit, teststart), иначе данные будут неполные.

2 - Нет смысла строить «сессии» через интервалы между заходами. Достаточно сравнивать даты захода и активности.

3 - Окна (LEAD/LAG) и self-join здесь избыточны.

4 - Последние два CTE (D1, D2) можно объединить в один.

5 - Стоит использовать понятные имена CTE и столбцов.

6 - Здесь не нужно сортировать CTE с помощью ORDER BY.

7 - Процент неактивных заходов лучше вычислять через LEFT JOIN и COUNT CASE, а не через деление двух CTE.

**Изменённый код:**

**WITH** *user\_activity\_days* **AS** (

**SELECT** user\_id, **date**(created\_at) **AS** *activity\_date*

**FROM** coderun

**WHERE** user\_id >= 94

**UNION**

**SELECT** user\_id, **date**(created\_at)

**FROM** codesubmit

**WHERE** user\_id >= 94

**UNION**

**SELECT** user\_id, **date**(created\_at)

**FROM** teststart

**WHERE** user\_id >= 94

),

*user\_entry\_days* **AS** (

**SELECT** user\_id, **date**(entry\_at) **AS** *entry\_date*

**FROM** userentry

**WHERE** user\_id >= 94

**GROUP** **BY** user\_id, *entry\_date*

)

**SELECT**

**ROUND**(

100.0 \* **COUNT**(**CASE** **WHEN** *uad*.*activity\_date* **IS** **NULL** **THEN** 1 **END**) / **COUNT**(\*),

2

) **AS** *percent\_entries\_without\_activity*

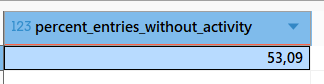
**FROM** *user\_entry\_days* *ued*

**LEFT** **JOIN** *user\_activity\_days* *uad*

**ON** *ued*.user\_id = *uad*.user\_id

**AND** *ued*.*entry\_date* = *uad*.*activity\_date*;

**Результат:**

****