Построение аналитики для корпоративных клиентов IT Resume

# Цель и метрики анализа

Целью анализа является выявление узких мест в программе обучения, вызывающих наибольшее затруднение у корпоративных студентов и определение точек роста.

В ходе анализа будет проверены гипотеза:

* самостоятельно обучающиеся студенты более мотивированы.

Для решение данной задачи были использованы следующие метрики:

* степень удовлетворенности обучением,
* rolling retention,
* длительность решения задач

Анализ проводился для компании N (company\_id = 1)

# Сравнительный анализ обучения корпоративных студентов компании N с обучением остальных студентов

## Сравнение метрик обучения студентов компании N с остальными студентами (удовлетворенность и rolling retention)

Прежде чем приступить к сравнению метрик подготовим дневник обучения, таблицу прогресса и представим основные модули задач компании N.

Код запроса SQL для дневника студента

**select**

**coalesce**(*u*.first\_name || ' ' || *u*.last\_name,

'Unknown') **as** *full\_name*,

*p*.name,

**max**(**case** **when** *c*.is\_false = 0 **then** 1 **when** *c*.is\_false = 1 **then** 0 **end**) *is\_done*

**from**

codesubmit *c*

**join** problem *p*

**on**

*c*.problem\_id = *p*.id

**join** users *u*

**on**

*c*.user\_id = *u*.id

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

**on**

*u*.company\_id = *ptc*.company\_id

**and** *p*.id = *ptc*.problem\_id

**where**

*u*.company\_id = 1

**group** **by**

*u*.id,

*p*.name

Код запроса SQL прогресса обучения студентов

**with**

*users\_company* **as** (

--сначала установим количество попыток решения задач

**select**

*u*.id **as** *user\_id*,

*u*.date\_joined::**date** **as** *dt*,

**coalesce**(*u*.first\_name || ' ' || *u*.last\_name,

'Unknown') **as** *full\_name*,

*u*.email,

*c*."name" **as** *company\_name*,

*p*.id **as** *problem\_id*

**from**

users *u*

**join** company *c*

**on**

*u*.company\_id = *c*.id

**join** codesubmit *c2*

**on**

*c2*.user\_id = *u*.id

**join** coderun *c3*

**on**

*c2*.user\_id = *c3*.user\_id

**and** *c2*.problem\_id = *c3*.problem\_id

**join** problem *p*

**on**

*c2*.problem\_id = *p*.id

**join** problem\_to\_company *ptc* **on**

*p*.id = *ptc*.problem\_id

**and** *u*.company\_id = *ptc*.company\_id

**where**

*c*.id = 1

),

*num\_attemps* **as**(

**select**

**distinct**

*user\_id*,

*full\_name*,

email,

*dt*,

**COUNT**(*problem\_id*) **over**(**partition** **by** *user\_id*,

*problem\_id*) **as** *cnt\_attemps*,

*company\_name*

**from**

*users\_company*

),

*all\_success* **as**(

**select**

**distinct**

*u*.id **as** *user\_id*,

*p*.id **as** *problem\_id*,

*u*.company\_id

**from**

users *u*

**join** codesubmit *c*

**on**

*c*.user\_id = *u*.id

**join** problem *p*

**on**

*c*.problem\_id = *p*.id

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

**on**

*p*.id = *ptc*.problem\_id

**and** *u*.company\_id = *ptc*.company\_id

**where**

*u*.company\_id = 1

**and** *c*.is\_false = 0

),

*all\_problems* **as**(

**select**

**distinct**

*ptc*.company\_id,

**count**(*ptc*.problem\_id) **over** (**partition** **by** ptc.company\_id) **as** total\_problems

**from**

problem\_to\_company *ptc*

**where**

*ptc*.company\_id = 1

**or** *ptc*.company\_id = 7

),

*num\_success* **as**(

**select**

**distinct**

*user\_id*,

*problem\_id*,

company\_id,

**COUNT**(*problem\_id*) **over**(**partition** **by** user\_id) **as** cnt\_success

**from**

*all\_success*

),

*total\_actions* **as**(

**select**

*na*.*user\_id*,

*full\_name*,

email,

*dt*,

cnt\_success **as** *num\_success*,

*company\_name*,

total\_problems,

**sum**(*cnt\_attemps*) **over**(**partition** **by** *na*.*user\_id*,

*problem\_id*) **as** *num\_attemps*

**from**

*num\_attemps* *na*

**join** *num\_success* *ns*

**on**

*na*.*user\_id* = *ns*.*user\_id*

**join** *all\_problems* *ap*

**on**

*ns*.company\_id = *ap*.company\_id

),

*grouped* **as**(

**select**

*full\_name*,

email,

*dt*,

*num\_attemps*,

*num\_success*,

total\_problems

**from**

*total\_actions*

**group** **by**

*full\_name*,

email,

*dt*,

*num\_attemps*,

*num\_success*,

total\_problems

)

**select**

\*,

**round**(*num\_success* / total\_problems::**numeric**,

2) **as** *progress*

**from**

*grouped*

**order** **by**

*dt*

По каким модулям подготовлены задачи для компании N

Код запроса SQL

**with** *full\_name\_problem* **as** (

**select**

**case**

**when** *p*.name **like** 'NEW! %' **then** **replace**(*p*.name,

'NEW! ',

'')

**else** *p*.name

**end** *problem*

**from**

problem\_to\_company *ptc*

**join** problem *p*

**on**

*ptc*.problem\_id = *p*.id

**where**

*ptc*.company\_id = 1

)

**select**

**split\_part**(*problem*,

'.',

1) **as** *modul*,

**count**(\*) **as** *cnt\_prblems*

**from**

*full\_name\_problem*

**group** **by**

**split\_part**(*problem*,

'.',

1)

Из круговой диаграммы видно, что большинство задач относится к модулю Python.

Далее посмотрим удовлетворенность студентов обучением (какой процент решенных задач) и сравним этот показатель с аналогичным показателем студентов, не относящихся к компании N. Сравнивать будем по одним и тем же модулям.

Запрос SQL

**select**

**cast**(**count**(**case** **when** *c*.is\_false = 0 **then** 1 **end**) **as** **numeric**)/ **count**(\*) **as** *share*

**from**

codesubmit *c*

**join** problem *p*

**on**

*c*.problem\_id = *p*.id

**join** users *u*

**on**

*c*.user\_id = *u*.id

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

**on**

*u*.company\_id = *ptc*.company\_id

**and** *p*.id = *ptc*.problem\_id

**where**

*u*.company\_id = 1

А теперь такой же показатель для остальных студентов, напомним, что показатель необходимо отразить по тем же модулям, по которым занимаются студенты компании N

Код запроса SQL

**with** *full\_name\_problem* **as** (

**select**

*p*.id **as** *problem\_id*,

**case**

**when** *p*.name **like** 'NEW! %' **then** **replace**(*p*.name,

'NEW! ',

'')

**else** *p*.name

**end** *problem*

**from**

problem *p*

),

*moduls* **as**(

**select**

*problem\_id*,

**split\_part**(*problem*,

'.',

1) **as** *modul*

**from**

*full\_name\_problem*

)

**select**

**cast**(**count**(**case** **when** *c*.is\_false = 0 **then** 1 **end**) **as** **numeric**)/ **count**(\*) **as** *share*

**from**

codesubmit *c*

**join** *moduls* *m*

**on**

*c*.problem\_id = *m*.*problem\_id*

**join** users *u*

**on**

*c*.user\_id = *u*.id

**where**

*u*.company\_id != 1

**and** (*modul* = 'SQL'

**or** *modul* = 'Pythom'

**or** *modul* = 'Numpy')

Сравнив два графика видим, что, хотя удовлетворенность обучением остальных студентов несколько выше, при отборе из данного сегмента студентов, обучающихся самостоятельно, уровень удовлетворенности еще больше увеличился, разрыв составил 0.06.

Прежде чем вывести заключительную метрику - rolling retention, - посмотрим динамику притока студентов компании N.

Код запроса SQL

**select**

**to\_char**(date\_joined,

'YYYY/mm'),

**count**(**case** **when** *u*.is\_active = 1 **then** 1 **end**) **as** *active\_students*

**from**

users *u*

**where**

*u*.company\_id = 1

**group** **by**

**to\_char**(date\_joined,

'YYYY/mm')

**order** **by**

**to\_char**(date\_joined,

'YYYY/mm')

Из графика видим, что основном приток приходится на ноябрь 2021г., дальше он постепенно уменьшался вплоть до периода апрель 2022г., общий период составил 5 месяцев.

Построим аналогичный график для остальных студентов.

Здесь код запроса не приводим, т.к. он такой же, только меняется условие на

**where**

**coalesce**(users.company\_id, 0) !=1

Из графика видно, что хотя и основной приток остальных студентов приходится на период февраль 2022г., в ноябре 2021г. количество студентов достаточное для сопоставления rolling retention.

Rolling retention будем смотреть за этот период с разбивкой на периоды:

* 1 день,
* 3 дня,
* 7 дней,
* 2 недели,
* месяц,
* 2 месяца,
* 3 месяца,
* 4 месяца,
* 5 месяцев

Код запроса SQL

**with** *rolling\_retentions* **as**(

**select**

u2.user\_id,

u.date\_joined **as** *date\_joined*,

u2.entry\_at **as** *entry\_at*,

**extract**(days

**from**

u2.entry\_at - u.date\_joined) **as** diff,

**to\_char**(u.date\_joined::**date**,

'YYYY-MM') **as** cohort

**from**

users u

**join** userentry u2

**on**

u.id = u2.user\_id

**where**

u.date\_joined::**date** > '2021-10-31'

**and** u.company\_id = 1

)

**select**

cohort,

**round**(**count**(**distinct** **case** **when** diff >= 0 **then** user\_id **end**) \* 100.0 / **count**(**distinct** **case** **when** diff >= 0 **then** user\_id **end**),

2) **as** **"0 (%)"**,

**round**(**count**(**distinct** **case** **when** diff >= 1 **then** user\_id **end**) \* 100.0 / **count**(**distinct** **case** **when** diff >= 0 **then** user\_id **end**),

2) **as** **"1 (%)"**,

**round**(**count**(**distinct** **case** **when** diff >= 3 **then** user\_id **end**) \* 100.0 / **count**(**distinct** **case** **when** diff >= 0 **then** user\_id **end**),

2) **as** **"3 (%)"**,

**round**(**count**(**distinct** **case** **when** diff >= 7 **then** user\_id **end**) \* 100.0 / **count**(**distinct** **case** **when** diff >= 0 **then** user\_id **end**),

2) **as** **"7 (%)"**,

**round**(**count**(**distinct** **case** **when** diff >= 14 **then** user\_id **end**) \* 100.0 / **count**(**distinct** **case** **when** diff >= 0 **then** user\_id **end**),

2) **as** **"14 (%)"**,

**round**(**count**(**distinct** **case** **when** diff >= 30 **then** user\_id **end**) \* 100.0 / **count**(**distinct** **case** **when** diff >= 0 **then** user\_id **end**),

2) **as** **"30 (%)"**,

**round**(**count**(**distinct** **case** **when** diff >= 60 **then** user\_id **end**) \* 100.0 / **count**(**distinct** **case** **when** diff >= 0 **then** user\_id **end**),

2) **as** **"60 (%)"**,

**round**(**count**(**distinct** **case** **when** diff >= 90 **then** user\_id **end**) \* 100.0 / **count**(**distinct** **case** **when** diff >= 0 **then** user\_id **end**),

2) **as** **"90 (%)"**,

**round**(**count**(**distinct** **case** **when** diff >= 120 **then** user\_id **end**) \* 100.0 / **count**(**distinct** **case** **when** diff >= 0 **then** user\_id **end**),

2) **as** **"120 (%)"**,

**round**(**count**(**distinct** **case** **when** diff >= 150 **then** user\_id **end**) \* 100.0 / **count**(**distinct** **case** **when** diff >= 0 **then** user\_id **end**),

2) **as** **"150 (%)"**

**from**

rolling\_retentions

**group** **by**

cohort

**order** **by**

cohort

Код запроса SQL для остальных студентов такой же, с поправкой на условие по id компании.

Видим, что студенты компании N активнее заходят на платформу, при этом удовлетворенность обучением у некорпоративных студентов выше.

Сравнительный анализ метрик верхнего уровня студентов компании N и остальных студентов показал, что, в целом, метрики схожи, однако, студенты компании N чуть больше вовлечены в процесс.

Далее посмотрим, как тщательно решали задачи студенты компании N по сравнению с остальными студентами.

## Сравнительный анализ попыток решения задач корпоративными студентами и студентами, не входящими в состав компании

Здесь сделаем оговорку, что при решении задачи сразу были созданы несколько таблиц для вывода результатов. Поскольку использовался СТЕ, а не подзапросы, ниже приведен полный код формирования таблиц для решения задач, далее по каждому промежуточному выводу будет приводится только результирующая таблица.

Код глобального запроса SQL

**with**

attempts\_company **as** (

--сначала установим количество попыток решения задач по модулям

**select**

u.date\_joined::**date**,

p.**name** **as** name\_problem,

c.user\_id **as** user\_id,

**coalesce**(u.first\_name || ' ' || u.last\_name,

'Unknown') **as** full\_name

**from**

codesubmit c

**join** users u **on**

c.user\_id = u.id

**join** problem p **on**

c.problem\_id = p.id

**join** problem\_to\_company ptc **on**

p.id = ptc.problem\_id

**and** u.company\_id = ptc.company\_id

**where**

u.company\_id = 1

),

attempts\_others **as** (

**select**

u.date\_joined::**date**,

p.**name** **as** name\_problem,

c.user\_id **as** user\_id,

**coalesce**(u.first\_name || ' ' || u.last\_name,

'Unknown') **as** full\_name

**from**

codesubmit c

**join** users u **on**

c.user\_id = u.id

**join** problem p **on**

c.problem\_id = p.id

**join** problem\_to\_company ptc **on**

p.id = ptc.problem\_id

**and** u.company\_id = ptc.company\_id

**where**

u.company\_id != 1

),

grouped\_by\_date **as** (

**select**

user\_id,

**case**

**when** name\_problem **like** 'NEW! %' **then** **replace**(name\_problem,

'NEW! ',

'')

**else** name\_problem

**end** name\_prb

**from**

attempts\_company

**group** **by**

user\_id,

name\_prb

),

--количество попыток решенных задач

moduls\_company **as** (

**select**

--date\_joined,

**split\_part**(name\_prb,

'.',

1) **as** modul,

**count**(\*) **as** cnt\_attempts\_company,

**count**(**distinct** user\_id) **as** cnt\_users\_company

**from**

grouped\_by\_date

**group** **by**

modul

),

--количество попыток решенных задач с детализацией по студентам

grouped\_by\_date\_and\_users **as** (

**select**

user\_id,

full\_name,

date\_joined,

**case**

**when** name\_problem **like** 'NEW! %' **then** **replace**(name\_problem,

'NEW! ',

'')

**else** name\_problem

**end** name\_prb

**from**

attempts\_company

**group** **by**

user\_id,

full\_name,

date\_joined,

name\_prb

),

moduls\_by\_student **as** (

**select**

user\_id,

full\_name,

date\_joined,

**split\_part**(name\_prb,

'.',

1) **as** modul,

**count**(\*)

**from**

grouped\_by\_date\_and\_users

**group** **by**

user\_id,

full\_name,

date\_joined,

modul

),

grouped\_by\_problems\_others **as** (

**select**

user\_id,

**case**

**when** name\_problem **like** 'NEW! %' **then** **replace**(name\_problem,

'NEW! ',

'')

**else** name\_problem

**end** name\_prb

**from**

attempts\_others

**group** **by**

user\_id,

name\_prb

),

--количество попыток решенных задач остальными студентами

moduls\_others **as** (

**select**

**split\_part**(name\_prb,

'.',

1) **as** modul,

**count**(name\_prb) **as** cnt\_attempts\_other,

**count**(**distinct** user\_id) **as** cnt\_users\_others

**from**

grouped\_by\_problems\_others

**group** **by**

modul

),

--количество попыток решенных задач студентами компании N и остальными студентами по модулям

moduls\_all **as** (

**select**

mc.modul,

mc.cnt\_attempts\_company,

mo.cnt\_attempts\_other,

mc.cnt\_users\_company,

mo.cnt\_users\_others,

**round**(

cnt\_attempts\_company \* 100.0 / (cnt\_attempts\_company + cnt\_attempts\_other) / 100,

2

) **as** share\_attempts\_company,

**round**(

cnt\_attempts\_other \* 100.0 / (cnt\_attempts\_company + cnt\_attempts\_other) / 100,

2

) **as** share\_attempts\_other,

**round**(

cnt\_users\_company \* 100.0 / (cnt\_users\_company + cnt\_users\_others) / 100,

2

) **as** share\_users\_company,

**round**(

cnt\_users\_others \* 100.0 / (cnt\_users\_company + cnt\_users\_others) / 100,

2

) **as** share\_users\_others

**from**

moduls\_company mc

**left** **join** moduls\_others mo **on**

mc.modul = mo.modul

)

**select**

\*

**from**

moduls\_all

Соотношение сегментов студентов (студенты компании N и остальные студенты). Заметим, что здесь приведено соотношение студентов, решавших одни и те же задачи

Код финального запроса SQL из глобального запроса

**select**

*modul*,

*cnt\_users\_company*,

*cnt\_users\_others*

**from**

*moduls\_all*

Определим долю попыток решения задач в разрезе модулей среди студентов компании N

Финальный запрос SQL для студентов компании N из глобального запроса

**select**

*modul*,

*cnt\_attempts\_company*

**from**

*moduls\_all*

и для остальных студентов

**select**

*modul*,

*cnt\_attempts\_other*

**from**

*moduls\_all*

Сравнив с круговой диаграммой количества задач, видим что, хотя задач по Python больше, чем задач по SQL, сложность у студентов вызвали именно задачи по SQL. Опять же у обоих сегментов студентов, но у сегмента студентов компании N задачи модуля SQL вызвали даже несколько больше сложностей, чем у студентов компании N.

## Сравнительный анализ выполнения задач

Далее нашей промежуточной целью была цель установить самые долго решаемые задачи.

При решении задачи в лоб выяснилось, что некоторые задачи решались более ста дней, что выглядит неправдоподобно, при выборочном анализе кода, отправленного на проверку, выяснилось, что код не менялся, просто студент почему-то решил снова нажать на кнопку Отправить через несколько месяцев.

К сожалению, объема памяти приложения DBeaver не хватило, чтобы программно исключить такие записи из глобального запроса SQL по количеству попыток решения задач, поэтому пришлось анализировать код по каждому такому студенту визуально и фильтр устанавливать непосредственно при формировании таблиц для вычисления duration.

Ниже представлена таблица таких пользователей и задач (длительность между первой и последней попытками не менее 45 дней), сразу же сделано сравнение текущего и предыдущего кода а также сравнение кода студента с эталонным кодом.

Код запроса SQL

**with** *problems* **as**(

**select**

*u*.id **as** *user\_id*,

*p*.id **as** *problem\_id*,

*p*.name **as** *name*,

*p*.company\_id,

*c*.created\_at,

**replace**(*p*.solution, '<inject-highlight>', '') **as** *solution*,

*c*.code,

*c*.is\_false

**from** codesubmit *c*

**join** users *u*

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

**join** problem *p*

**on** *c*.problem\_id = *p*.id

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

**on** *p*.id = *ptc*.problem\_id **and** *u*.company\_id = *ptc*.company\_id

**where** *u*.company\_id = 1 --and u.id = 63 --and (ptc.problem\_id = 117 or ptc.problem\_id = 123)

),

*min\_max\_duration* **as**(

**select**

*user\_id*,

created\_at,

*problem\_id*,

*name*,

code,

is\_false,

**replace**(*solution*, '</inject-highlight>', '') **as** *solution*,

**min**(created\_at) **over** (**partition** **by** *user\_id*, *problem\_id*) **as** *min\_duration*,

**max**(created\_at) **over** (**partition** **by** *user\_id*, *problem\_id*) **as** *max\_duration*

**from** *problems*

),

*most\_long\_problem* **as**(

--при определении самых долго решаемых задач в лоб выяснилось, что некоторые задачи решались более ста дней, что выглядит неправдоподобно,

--при выборочном анализе кода, отправленного на проверку, выяснилось, что код не менялся, просто студент почему-то решил снова нажать на кнопку Отправить через несколько месяцев.

--К сожалению, объема памяти приложения DBeaver не хватило, чтобы программно исключить такие записи, поэтому пришлось анализировать код по каждому такому студенту визуально

--и фильтр устанавливать непосредственно при формировании таблиц для вычисления duration

--ниже представлена таблица таких пользователей и задач,

--а визуальное сравнение кодов отправленных последними с предыдущим позволило установить записи, в которых код не менялся, с течением достаточно долгого времени:

--user\_id 50, problem\_id - 117,

--user\_id 263, problem\_id - 117,

--user\_id 861, problem\_id - 133,

--user\_id 1107, problem\_id - 134,

--user\_id 348, problem\_id - 48,

**select**

*user\_id*,

*problem\_id*,

**extract**(**day** **from** max\_duration - *min\_duration*) **as** *duration*

**from** *min\_max\_duration*

**where** **extract**(**day** **from** max\_duration - *min\_duration*) > 30

**group** **by** *user\_id*, *problem\_id*, *duration*

**order** **by** *duration* **desc**

),

*duration* **as**(

**select**

*user\_id*,

*problem\_id*,

created\_at,

*name*,

code,

is\_false,

*solution*,

**extract**(**day** **from** max\_duration - *min\_duration*) **as** *duration*

**from** *min\_max\_duration*

),

*change\_code* **as**(

**select** \*,

**lag**(code) **over** (**partition** **by** *user\_id*, *problem\_id* **order** **by** *user\_id*, *duration* **desc**, created\_at) **as** *code\_old*,

**row\_number**() **over**(**partition** **by** *user\_id*, *problem\_id* **order** **by** *user\_id*, created\_at) **as** *num\_row*

**from** *duration*

)

**select** \*,

**case**

**when** *code\_old* = code **then** 'no'

**else** 'yes'

**end** *code\_changed*,

**case**

**when** *solution* = code **then** 'yes'

**else** 'no'

**end** *as\_etalon*,

**max**(*num\_row*) **over** (**partition** **by** *user\_id*, *problem\_id*) **as** *num\_attempts*

**from** *change\_code*

**where** *duration* > 45

Собственно, детальное сравнение последующего и предыдущего кода, а также сравнение кода с эталонным и натолкнуло на гипотезу, насколько студенты компании N мотивированы получить именно знания, а не просто отчитаться перед работодателем о прохождении курса.

Итак, возьмем студента из самой первой строки (самое долгое решение задачи), заодно посмотрим и задачу с id 117, эту задачу студент решал долго и посмотрим, изменился ли код.

Удивительно, на первой, самой долго решаемой задаче было целых 6 попыток, но код ни разу не изменился по сравнению с изначальным, видимо, студент не видел прогресс, важный для него, и в отчаянии нажимал на кнопку. Со второй долго решаемой им задачей еще интереснее (id 123), код, действительно изменился, но он стал эталонным, т.е. студент в итоге не решил задачу, а просто вставил копию эталонной задачи.

Таким образом, визуальное сравнение предыдущего и последующего кода выявило студентов компании N, которые через несколько месяцев/недель, не меняли код, а просто снова нажимали на кнопку Отправить

| user\_id | problem\_id | duration\_total | duration\_lag |
| --- | --- | --- | --- |
| 50 | 117 | 126 | 125 |
| 263 | 117 | 124 | 124 |
| 861 | 133 | 79 | 63 |
| 1107 | 134 | 65 | 53 |
| 348 | 134 | 48 | 48 |

Теперь посмотрим, ведут ли себя схожим образом остальные студенты? В результирующей таблице только один студент (user\_id = 63).

Посмотрим, менял ли он свой код. Сравнив коды видим, только в одной задаче (problem\_id = 121) код не изменен по сравнению с предыдущей записью, в остальные случаях, а количество попыток иногда было больше 10, код менялся, причем, он становился лучше. Необходимо также отметить, что эту задачу студент смог решить только скопировав решение. В остальных случаях, если он и копировал решение из эталона, но, затем возвращался к задаче и старался решить её самостоятельно.

Выведем теперь пузырьковую диаграмму длительности решения задач. Запрос SQL использовался прежний, только убран фильтр по длительности попыток и добавлена сортировка по убыванию.

Пять самый долгих решений не учитываем, т.к. студенты не меняли код, только нажимали кнопку Отправить. Видно, что большинство задач решалось в течение 3 дней, среди задач, вызвавших сложности, нельзя выделить какую-то одну задачу, у разных студентов сложности вызывали разные задачи.

# Выводы

Как показал сравнительный анализ, в целом, поведение студентов компании N не отличается от поведения остальных студентов, студенты больше попыток совершают при решении задач модуля SQL, студенты компании N чуть чаще заходят на платформу.

Однако, более детальный анализ аномальных выбросов решения задач, которые некоторые студенты решали достаточно долго (несколько месяцев) выявил случаи, когда студенты не меняли код, а просто нажимали кнопку Отправить и даже отправляли на проверку эталонное решение, а не свой код.

Предложения для повышения мотивации студентов решать задачи собственными силами и стараться совершенствовать свой код.

1. Ввести рейтинговые очки, которые увеличиваются при отправке верного решения ДО просмотра эталонного решения и уменьшаются при просмотре эталонного решения до отправки первого своего решения. Причем, сам размер бонуса зависит от попыток решить задачу правильно самостоятельно. Эти рейтинговые очки отразить как kpi при отчете перед компанией
2. Настроить контроль решения студента, чтобы исключить копирование эталонного решения, ввести большой виртуальный штраф за подобные действия.

Также предложение по улучшению структуры базы - добавить данные по модулю в таблицы с задачами, что позволит сравнивать поведение корпоративных студентов и студентов, обучающихся самостоятельно, при решении задач одного и того же модуля, для понимания мотивации корпоративных студентов.

То, что мотивации могут различаться можно предположить хотя бы из-за того, что удовлетворенность обучением у самостоятельно обучающихся студентов выше, чем у студентов, проходящих обучение от компаний.

На далекую перспективу можно использовать ИИ для определения авторства кода, отправляемого на проверку, был ли код написан самостоятельно или просто переделан из эталонного решения.