

Создание плана повышения retention платформы IT Resume

Последнее время мы работаем по подписочной модели - чтобы открыть полный доступ, люди покупают Месячную или Годовую подписку. И метрика удержания для нас - очень важна. Чем больше человек пользуется нашим сервисом, тем дольше его подписка будет активирована, а значит тем больше денег мы заработаем.

Вам нужно найти точки роста для метрики удержания и предложить продуктовые изменения, которые к этому могут привести.

Постарайтесь стать психологом - иногда людей стимулируют и мотивируют очень неочевидные вещи. Поставьте себя на место нашего клиента - что будет мотивировать вас постоянно заходить в сервис, решать задачи и не отписываться?

Краткое саммари

Цель: Увеличить метрику удержания пользователей сервиса, работающего по подписочной модели, путем выявления и внедрения продуктовых изменений, которые повысят вовлеченность и мотивацию пользователей. Это позволит продлить сроки активных подписок, увеличить доходность сервиса и укрепить его рыночные позиции.

Задачи проекта:

1. Анализ текущего поведения пользователей

- Изучить, на каких этапах пользователи чаще всего отписываются.
- Определить ключевые действия, которые совершают удержанные пользователи.

2. Выявление факторов, влияющих на retention

- Определить, какие функции/контент наиболее ценны для пользователей.
- Найти триггеры, которые мотивируют пользователей оставаться в сервисе.

Метрики:

1. Retention Rate (удержание)

- Rolling Retention (доля пользователей, оставшихся активными через N дней).

2. Поведенческие метрики

- Мы фокусируемся на метриках, которые напрямую влияют на Retention:
- Частота использования сервиса (DAU, MAU).
- Распределение активации аккаунтов (подтвердили/не подтвердили аккаунт).
- Воронка «Сделал попытку решить задачу – «Решил задачу успешно – Пополнил кошелек».
- Распределение, показывающее, на что пользователи тратят CodeCoins (внутренняя валюта): покупают задачи, подсказки, решения, тесты).
- Распределение первых и повторных покупок: сколько человек купило первый раз, а сколько совершают уже не первую покупку.

3. Финансовые метрики

- Churn Rate (отток) – процент отписавшихся за период.

Почему именно эти метрики?

- Они отражают реальную вовлеченность, а не просто регистрации.

- Позволяют выявить "точки оттока".
- Показывают, какие аспекты сервиса наиболее ценны для пользователей.

Гипотезы:

1. Недостаток мгновенной ценности

- Пользователи не видят быстрой отдачи от подписки → нужно усилить "wow-эффект" в первые дни.

2. Отсутствие привычки пользоваться сервисом

- Люди забывают заходить, потому что нет триггеров.

3. Отсутствие эмоциональной привязки

- Подписка воспринимается как транзакция, а не как часть карьеры.

4. Неочевидность долгосрочной выгоды

- Пользователи не понимают, как сервис поможет через полгода.

Итог:

Фокус на вовлечение, персонализацию и психологические триггеры (прогресс, социальное признание, страх потери) поможет увеличить retention. Проверка гипотез позволит найти самые эффективные решения.

Часть 1. Расчет Retention Rate (удержание)

- Rolling Retention – доля пользователей, оставшихся активными через N дней.

Метрика rolling retention будет более показательной для проводимого анализа, чем метрика n-day retention. Человек мог зайти на платформу спустя 20 дней после регистрации, но не прийти в 7, и тогда n-day retention 7 дня покажет, что мы потеряли человека, а rolling retention 7 дня покажет, что человек все еще с нами, потому что эта метрика учитывает текущий день и все последующие дни

Для анализа возьмем конкретные N-дни: 0, 1, 3, 7, 14, 30, 60 и 90 дня.

Пользователей разбиваем по когортам. В качестве признака когорты будем использовать месяц регистрации пользователя.

Основные моменты:

Заходы пользователя на платформу находятся в таблице UserEntry.

Когорты при расчете retention формируем по месяцам регистрации пользователей и только начиная с 2022 года, т.к. данных за 2021 год недостаточно для анализа.

Для некоторых пользователей в таблице UserEntry нет данных об их входе в самый первый день. Поэтому при расчете ретеншена 0 дня будем учитывать пользователей, которые заходили в 0 день.

При расчетах будем опираться на реальные дни - 24 часа. Если пользователь зарегистрировался сегодня в 23:59, то первый день начнется завтра в 23:59. Так корректнее.

Решение:

С помощью CTE **with users_filt** выберем все столбцы из таблицы users.

Далее с помощью CTE **a** выберем пользователей (**u.user_id**), разобьем их на когорты (**cohort**) с помощью функции **to_char(u2.date_joined, 'YYYY-MM')**, разницу (**diff**) между последней датой захода на платформу и датой регистрации в днях с помощью функции **extract(day from u.entry_at - u2.date_joined)** из таблицы **userentry u**, к которой прибавим с помощью **left join** первое CTE **users_filt u2**, а также зададим условие **where u.entry_at::date - u2.date_joined::date >= 0**, чтобы разница в днях между датой последнего захода на платформу и датой регистрации равнялась или была больше 0. Далее из CTE **a** выберем когорты, рассчитаем процент пользователей, проявляющих активность в 0, 1, 3, 7, 14, 30, 60 и 90 дни и последующие и сгруппируем получившиеся данные по когортам.

В результате SQL-запрос будет выглядеть следующим образом:

```
with users_filt as (
select
    *
from
    users
),
a as (
select
    u.user_id,
    to_char(u2.date_joined, 'YYYY-MM') as cohort,
    extract(day from u.entry_at - u2.date_joined) as diff
from
    userentry u
left join users_filt u2
    on
        u.user_id = u2.id
where
    u.entry_at::date - u2.date_joined::date >= 0
order by
```

```

        user_id,
        diff
    )
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), 1) as "Day 0",
    round(count(distinct case when diff >= 1 then user_id end) * 100.0 /
count(distinct case when diff >= 0 then user_id end), 1) as "Day 1",
    round(count(distinct case when diff >= 3 then user_id end) * 100.0 /
count(distinct case when diff >= 0 then user_id end), 1) as "Day 3",
    round(count(distinct case when diff >= 7 then user_id end) * 100.0 /
count(distinct case when diff >= 0 then user_id end), 1) as "Day 7",
    round(count(distinct case when diff >= 14 then user_id end) * 100.0 /
count(distinct case when diff >= 0 then user_id end), 1) as "Day 14",
    round(count(distinct case when diff >= 30 then user_id end) * 100.0 /
count(distinct case when diff >= 0 then user_id end), 1) as "Day 30",
    round(count(distinct case when diff >= 60 then user_id end) * 100.0 /
count(distinct case when diff >= 0 then user_id end), 1) as "Day 60",
    round(count(distinct case when diff >= 90 then user_id end) * 100.0 /
count(distinct case when diff >= 0 then user_id end), 1) as "Day 90"
from
    a
group by
    cohort
order by
    cohort

```

В результате обработки SQL-запроса получится следующая таблица, представленная на рисунке 1.

В ячейках таблицы здесь стоит отношение количества людей, которые пришли в день N (или любой последующий) и в день 0 (или любой последующий).

	A-Z cohort	123 Day 0	123 Day 1	123 Day 3	123 Day 7	123 Day 14	123 Day 30	123 Day 60	123 Day 90
1	2021-03	100	100	100	100	100	0	0	0
2	2021-04	100	100	66,7	55,6	44,4	0	0	0
3	2021-07	100	100	100	100	100	100	100	100
4	2021-08	100	100	100	100	100	100	100	100
5	2021-09	100	100	100	50	50	50	50	50
6	2021-11	100	71,4	65,6	63	60,8	52,4	40,7	29,6
7	2021-12	100	50,4	43,8	42,1	35,5	31,4	23,1	13,2
8	2022-01	100	41,8	35,5	29,7	25,3	15,3	8,8	4,6
9	2022-02	100	29,2	21	16,3	12,7	9,4	5,6	1,2
10	2022-03	100	30,6	24,8	18,3	14	8,2	1,2	0
11	2022-04	100	43,8	35,6	28,4	21,6	10,1	0	0

Рисунок 1 – Результат SQL-запроса для расчета rolling retention

Исходя из данных таблицы, можно сделать следующие выводы:

1. Общий тренд Rolling Retention:

- Для всех когорт наблюдается постепенное снижение удержания с течением времени, что ожидаемо. Исключением являются июльская (2021-07) и августовская (2021-08) когорты 2021 года – возможно, они являются аномальными значениями.
- Наибольший отток происходит в первые периоды (первые 30–60 дней) в мартовской когорте 2021 года (2021-03), апрельской когорте 2021 года (2021-04) и апрельской когорте 2022 года (2022-04).

2. Проблемные места:

- Низкое начальное удержание после 30 дня может указывать на:
 - Проблемы с onboarding (например, сложная регистрация).
 - Низкое качество трафика (рекламные каналы привлекли "холодных" пользователей).

3. Рекомендации:

- Улучшить первые 30 дней взаимодействия:
 - Оптимизировать onboarding (упростить регистрацию, добавить обучающие материалы).
 - Внедрить триггеры вовлечения (например, напоминания о недозаполненном профиле).
- Проанализировать причины долгосрочного оттока:
 - Провести опросы или A/B-тесты для выявления факторов, влияющих на удержание.

Заключение:

- Необходимо сфокусироваться на улучшении первых 30 дней пользовательского опыта.
- Можно запустить качественные исследования (опросы, интервью) для понимания причин оттока.

Часть 2. Расчет поведенческих метрик

- Частота использования сервиса (DAU, MAU).

Решение:

- Расчет DAU:

Напишем SQL-запрос для расчета DAU на основании заходов пользователей на платформу (таблица UserEntry).

Выберем дату захода на платформу - `entry_at::date` и посчитаем уникальное количество пользователей с помощью функции `count(distinct user_id)`. Также сгруппируем данные по дате.

В результате SQL-запрос будет выглядеть следующим образом:

```
select
    entry_at::date as "Дата",
    count(distinct user_id) as "Кол-во пользователей"
from
    userentry
group by
    entry_at::date
```

В результате обработки SQL-запроса будет получена таблица, содержащая 272 строки (часть получившейся таблицы представлена на рисунке 2).

	Дата	123 Кол-во пользователей
1	2021-03-29	2
2	2021-03-31	1
3	2021-04-01	2
4	2021-04-02	3
5	2021-04-03	1
6	2021-04-04	3
7	2021-04-05	2
8	2021-04-06	2
9	2021-04-09	3
10	2021-04-11	2
11	2021-04-12	1
12	2021-04-13	1
13	2021-04-14	3
14	2021-04-20	1
15	2021-04-21	2
16	2021-04-22	1
17	2021-04-23	1

Рисунок 2 – Результат SQL-запроса для расчета dau

— Расчет MAU:

Напишем SQL-запрос для расчета медианного и среднего MAU на основании заходов пользователей на платформу (таблица UserEntry), чтобы в результате получить два числа.

С помощью CTE **with for_mau** определим количество уникальных пользователей (*cnt*). Используем для это функцию **count(distinct user_id)**. Разобьем данные по месяцам (*ym*), для чего используем функцию **to_char(entry_at, 'YYYY-MM')**.

Используем таблицу userentry и. Сгруппируем данные по месяцам - **group by ym**.

Далее из CTE рассчитаем медианное MAU (*median_mau*) с помощью оконной функции **percentile_cont(0.5) WITHIN GROUP (ORDER BY cnt)** и среднее MAU (*avg_mau*) с помощью функции **avg(cnt)**.

В результате SQL-запрос будет выглядеть следующим образом:

```
with for_mau as (
    select to_char(entry_at, 'YYYY-MM') as ym, count(distinct user_id) as
cnt
    from userentry
    group by ym
)
select
    percentile_cont(0.5) WITHIN GROUP (ORDER BY cnt) as median_mau,
    avg(cnt) as avg_mau
from for_mau
```

В результате обработки SQL-запроса будут выведены два столбца – медианное и среднее MAU (рисунок 3).

	123 median_mau	123 avg_mau
1	195,5	300,8333333333

Рисунок 3 – Результат SQL-запроса для расчета median_mau и avg_mau

- Распределение активации аккаунтов (подтвердили/не подтвердили аккаунт).

Решение:

Напишем SQL-запрос для расчета распределения активации аккаунтов (таблица Users). Вычислим условные значения активированных и неактивированных аккаунтов с помощью выражения **case when is_active = 1 then 'Активированный аккаунт' else 'Неактивированный аккаунт' end** и посчитаем общее количество с помощью функции **count(*)**. Также сгруппируем данные по столбцу **is_active**. В результате SQL-запрос будет выглядеть следующим образом:

```
select
    case
        when is_active = 1 then 'Активированный аккаунт'
        else 'Неактивированный аккаунт'
    end,
    count (*)
from
    users
group by
    is_active
```

В результате обработки SQL-запроса будет получена следующая таблица с данными, представленная на рисунке 4.

	A-Z case	123 count
1	Неактивированный аккаунт	302
2	Активированный аккаунт	2 472

Рисунок 4 – Результат SQL-запроса для расчета распределения активации аккаунтов

- Воронка «Сделал попытку решить задачу – «Решил задачу успешно – Пополнил кошелек».

Решение:

Напишем SQL-запрос для расчета воронки «Сделал попытку решить задачу – «Решил задачу успешно – Пополнил кошелек»:

С помощью CTE **codesubmit_filt**, **coderun_filt** и **transaction_filt** выберем все столбцы в таблицах **codesubmit**, **coderun** и **transaction**.

Затем в CTE **attempts** объединим id пользователей (**user_id**) из предыдущих CTE **codesubmit_filt** и **coderun_filt** с помощью оператора **union all**.

Далее в CTE **results** посчитаем количество уникальных пользователей, сделавших попытку решить задачу, с помощью функции **select count(distinct user_id)** из CTE **attempts**. Повторим то же самое (рассчитаем количество уникальных пользователей, решивших задачу, и количество уникальных пользователей, пополнивших кошелек) для CTE **codesubmit_filt** и **transaction_filt**, указав с помощью условия **where** ограничения для выборки данных (**is_false = 0** для CTE **codesubmit_filt** и **type_id = 2** для CTE **transaction_filt**).

В последнем шаге выберем все столбцы из CTE **results**.

В результате SQL-запрос будет выглядеть следующим образом:

```
with codesubmit_filt as (
    select *
    from codesubmit
),
coderun_filt as (
    select *
    from coderun
),
transaction_filt as (
    select *
    from transaction
),
attempts as (
    select user_id
```

```

        from codesubmit_filt
        union all
        select user_id
        from coderun_filt
    ),
    results as (
        select count(distinct user_id) as "Количество", '1. Попытались решить задачу' as description
        from attempts
        union
        select count(distinct user_id) as "Количество", '2. Решили задачу' as description
        from codesubmit_filt
        where is_false = 0
        union
        select count(distinct user_id) as "Количество", '3. Пополнил кошелек' as description
        from transaction_filt
        where type_id = 2
    )
select *
from results
order by description

```

В результате обработки SQL-запроса будет выведена таблица, представленная на рисунке 5.

	123 Количество	A-Z description
1	926	1. Попытались решить задачу
2	714	2. Решили задачу
3	23	3. Пополнил кошелек

Рисунок 5 – Результат SQL-запроса для расчета воронки «Сделал попытку решить задачу – Решил задачу успешно – Пополнил кошелек»

- Распределение, показывающее, на что пользователи тратят CodeCoins: покупают задачи, подсказки, решения, тесты.

Решение:

Напишем SQL-запрос для расчета распределения, показывающее, на что пользователи тратят CodeCoins: покупают задачи, подсказки, решения, тесты:

С помощью CTE `transaction_filt` выберем все столбы в таблице `transaction`.

Далее из CTE выберем тип транзакции (`t2.description`) и посчитаем количество транзакций с помощью функции `count(t.id)`. К CTE `transaction_filt t` прибавим с помощью `left join` таблицу `transactiontype t2`, зададим условие `where t2."type" = 1 or t2."type" between 23 and 28` и сгруппируем данные по типу транзакции.

В результате SQL-запрос будет выглядеть следующим образом:

```

with transaction_filt as (
    select
        *
    from
        transaction
)
select
    t2.description as "Тип транзакции",
    count(t.id) as "Количество транзакций"
from
    transaction_filt t
left join transactiontype t2
on
    t.type_id = t2."type"

```



```

where
    t2."type" = 1
    or t2."type" between 23 and 28
group by
    t2.description
order by
    count(t.id) desc

```

В результате обработки SQL-запроса будет выведена таблица, представленная на рисунке 6.

	A-Z Тип транзакции	123 Количество транзакций
1	Купить задачу	1 675
2	Купить тест	989
3	Купить решение задачи	423
4	Купить подсказку к задаче	118
5	Списание	5

Рисунок 6 – Результат SQL-запроса для расчета распределения, показывающее, на что пользователи тратят CodeCoins

- Распределение первых и повторных покупок: сколько человек купило первый раз, а сколько совершают уже не первую покупку.

Решение:

Напишем SQL-запрос для расчета распределения первых и повторных покупок:

С помощью CTE **a** выберем из таблицы **"transaction"** id пользователей (**user_id**), посчитаем все строки в таблице с помощью функции **count(*)**, а также вычислим условные значения «купили один раз» и «купили больше одного раза» с помощью выражения **case**. К таблице **"transaction"** прибавим с помощью **left join** таблицу **transactiontype**, зададим условие **where transactiontype."type" = 1 or transactiontype."type" between 23 and 28** и сгруппируем данные по id пользователей.

Далее из CTE выберем **user_type** и посчитаем количество, а также сгруппируем данные по **user_type**.

В результате SQL-запрос будет выглядеть следующим образом:

```

with a as (
select
    user_id,
    count(*),
    case
        when count(*) = 1 then 'Купили один раз'
        else 'Купили больше одного раза'
    end user_type
from
    "transaction"
left join transactiontype
on
    "transaction".type_id = transactiontype."type"
where
    transactiontype."type" = 1
    or transactiontype."type" between 23 and 28
group by
    user_id
)
select
    user_type,
    count(*)
from
    a
group by
    user_type

```

В результате обработки SQL-запроса будет выведена таблица, представленная на рисунке 7.

	A-Z user_type	123 count
1	Купили один раз	734
2	Купили больше одного раза	406

Рисунок 7 – Результат SQL-запроса для расчета распределения первых и повторных покупок

Часть 3. Расчет финансовых метрик

- Churn Rate (отток) – процент отписавшихся за период.

Анализ показателей оттока (Churn Rate) позволяет выявить ключевые проблемы удержания пользователей и определить эффективность стратегий по их вовлечению.

Решение:

С помощью CTE **with users_filt** выберем все столбцы из таблицы users.

Далее с помощью CTE **user_activity** выберем пользователей (**u.user_id**), разобьем их на когорты (**cohort**) с помощью функции **to_char(u2.date_joined, 'YYYY-MM')**, разницу (**days_since_join**) между последней датой захода на платформу и датой регистрации в днях с помощью функции **extract(day from u.entry_at - u2.date_joined)**, максимальное количество дней (**max_active_day**) с помощью оконной функции **MAX(extract(day from u.entry_at - u2.date_joined)) over (partition by u.user_id)** из таблицы **userentry u**, к которой прибавим с помощью **left join** первое CTE **users_filt u2**, а также зададим условие **where u.entry_at::date - u2.date_joined::date >= 0**, чтобы разница в днях между датой последнего захода на платформу и датой регистрации равнялась или была больше 0. Далее в CTE **cohort_stats** выберем из CTE **user_activity** когорты, посчитаем количество уникальных пользователей (**cohort_size**) с помощью функции **COUNT(distinct user_id)** и рассчитаем процент пользователей, проявляющих активность в 0 и 1 дни и последующие, рассчитаем процент отписавшихся пользователей (Churn Rate) и сгруппируем получившиеся данные по когортам.

Последним шагом выберем из CTE **cohort_stats** интересующие нас данные.

В результате SQL-запрос будет выглядеть следующим образом:

```
with users_filt as (
select
    *
from
    users
),
user_activity as (
select
    u.user_id,
    to_char(u2.date_joined, 'YYYY-MM') as cohort,
    extract(day from u.entry_at - u2.date_joined) as days_since_join,
    MAX(extract(day from u.entry_at - u2.date_joined)) over (partition by
u.user_id) as max_active_day
from
    userentry u
left join users_filt u2
    on
        u.user_id = u2.id
where
    u.entry_at::date - u2.date_joined::date >= 0
),
cohort_stats as (
```

```

select
    cohort,
    COUNT(distinct user_id) as cohort_size,
    --Пакет Rolling Retention
    ROUND(COUNT(distinct case when days_since_join >= 0 then user_id end)
* 100.0 /
        COUNT(distinct case when days_since_join >= 0 then user_id
end), 1) as "Day 0",
    ROUND(COUNT(distinct case when days_since_join >= 1 then user_id end)
* 100.0 /
        COUNT(distinct case when days_since_join >= 0 then user_id
end), 1) as "Day 1",
    --Пакет Churn Rate (100% - Retention)
    100 - ROUND(COUNT(distinct case when days_since_join >= 1 then
user_id end) * 100.0 /
        COUNT(distinct case when days_since_join >= 0 then user_id
end), 1) as "Churn_Day1",
    100 - ROUND(COUNT(distinct case when days_since_join >= 7 then
user_id end) * 100.0 /
        COUNT(distinct case when days_since_join >= 0 then user_id
end), 1) as "Churn_Day7",
    100 - ROUND(COUNT(distinct case when days_since_join >= 30 then
user_id end) * 100.0 /
        COUNT(distinct case when days_since_join >= 0 then user_id
end), 1) as "Churn_Day30",
    100 - ROUND(COUNT(distinct case when days_since_join >= 90 then
user_id end) * 100.0 /
        COUNT(distinct case when days_since_join >= 0 then user_id
end), 1) as "Churn_Day90"
from
    user_activity
group by
    cohort
)
select
    cohort,
    cohort_size,
    "Day 0",
    "Day 1",
    "Churn_Day1",
    "Churn_Day7",
    "Churn_Day30",
    "Churn_Day90"
from
    cohort_stats
order by
    cohort

```

В результате обработки SQL-запроса будет выведена таблица, представленная на рисунке 8.

	A-Z cohort	123 cohort_size	123 Day 0	123 Day 1	123 Churn_Day1	123 Churn_Day7	123 Churn_Day30	123 Churn_Day90
1	2021-03	5	100	100	0	0	100	100
2	2021-04	9	100	100	0	44,4	100	100
3	2021-07	1	100	100	0	0	0	0
4	2021-08	1	100	100	0	0	0	0
5	2021-09	2	100	100	0	50	50	50
6	2021-11	189	100	71,4	28,6	37	47,6	70,4
7	2021-12	121	100	50,4	49,6	57,9	68,6	86,8
8	2022-01	498	100	41,8	58,2	70,3	84,7	95,4
9	2022-02	891	100	29,2	70,8	83,7	90,6	98,8
10	2022-03	415	100	30,6	69,4	81,7	91,8	100
11	2022-04	208	100	43,8	56,2	71,6	89,9	100

Рисунок 8 – Результат SQL-запроса для расчета Churn Rate

- Добавлен расчет Churn Rate как 100% - Retention Rate для ключевых временных точек (1, 7, 30, 90 дней).

- Сохранены оригинальные метрики Rolling Retention 0 и 1 дня для сравнения.
- Добавлен размер когорты (cohort_size) для контекста.

Интерпретация результатов:

- Churn_Day1 показывает процент пользователей, которые не вернулись на 1-й день.
- Churn_Day7 показывает процент пользователей, которые не вернулись через неделю.
- Churn_Day30 показывает месячный отток пользователей.
- Churn_Day90 показывает квартальный отток пользователей.

Исходя из данных таблицы, можно сделать следующие выводы:

1. Общий тренд Churn Rate:

- В первые месяцы (март-сентябрь 2021 года) Churn Rate был крайне низким (0% или близко к этому), что может быть связано с небольшим размером когорт (от 1 до 9 пользователей).
- Начиная с ноября 2021 года, Churn Rate значительно вырос, особенно в когортах с большим количеством пользователей.

2. Зависимость от размера когорты:

- Малые когорты (менее 10 пользователей) демонстрируют нестабильные показатели Churn Rate, вероятно, из-за недостатка данных.
- Крупные когорты (от 100 пользователей и более) показывают более устойчивые и высокие значения Churn Rate, что указывает на систематическую проблему с удержанием пользователей.

3. Динамика Churn Rate по месяцам:

- В марте и апреле 2022 года Churn Rate достиг 69,4% и 56,2% соответственно, что может свидетельствовать об ухудшении удержания пользователей или изменении стратегии.

4. Рекомендации:

- Необходимо проанализировать причины резкого роста Churn Rate, начиная с ноября 2021 года. Возможные факторы: изменения в продукте, маркетинговой стратегии или внешние условия.
- Улучшение раннего взаимодействия с пользователями (onboarding) может помочь снизить Churn Rate в первые месяцы.

Заключение:

- Данные показывают значительный рост Churn Rate в крупных когортах, что требует детального анализа и принятия мер по улучшению удержания пользователей. При этом retention на длительных интервалах также остается низким, что является отрицательным сигналом.

Общие выводы

- Подтверждение гипотез:

Гипотеза 1: Недостаток мгновенной ценности

- Пользователи не видят быстрой отдачи от подписки → нужно усилить "wow-эффект" в первые дни.

Решение:

- Персональные рекомендации сразу после оплаты.
- Быстрые победы (например, "Ты в топ-10% резюме по твоей специализации").

Гипотеза 2: Отсутствие привычки пользоваться сервисом

- Люди забывают заходить, потому что нет триггеров.

Решение:

- Еженедельные персонализированные дайджесты ("3 новых предложения для тебя").
- Push-уведомления с микрозадачами ("Обнови навыки — получи +20% к откликам").

Гипотеза 3: Отсутствие эмоциональной привязки

- Подписка воспринимается как транзакция, а не как часть карьеры.

Решение:

- Геймификация (уровни, бейджи, рейтинги).
- История прогресса ("За месяц ты улучшил навыки на 30%").

Гипотеза 4: Неочевидность долгосрочной выгоды

- Пользователи не понимают, как сервис поможет через полгода.

Решение:

- "Карьерная карта" с прогнозом роста.
- Кейсы успешных пользователей ("Как Иван получил оффер в FAANG за 4 месяца").

— Предлагаемые продуктовые изменения:

Быстрые победы (1-2 недели реализации)

- "Первые 3 шага" после оплаты (например, "Заполни профиль на 100% → получи доступ к закрытым заданиям").
- Еженедельные отчеты.

Среднесрочные (1-3 месяца)

- Геймификация:
 - Бейджи за активность.
 - Прогресс-бар заполнения профиля.
- Персонализация:
 - "Карьерный GPS" (какие навыки прокачать для целевой зарплаты).
 - Автоматические советы на основе A/B-тестов резюме.

Долгосрочные (3-6 месяцев)

- Комьюнити:
 - Вебинары с HR.
 - Возможность нетворкинга внутри платформы.

Ожидаемый результат

- +15-30% Retention за счет привычки и мгновенной ценности.
- Снижение Churn Rate благодаря FOMO (Fear of missing out – Боязнь пропустить интересное) и прогрессу.
- Рост LTV (пожизненной ценности пользователя).