RFM-анализ клиентской базы аптечной сети

Проверим количество строк:

```
select
	count(*)
from
	bonuscheques
```

- 38486 строк

Проверка дубликатов:

```
select
	count(*)
from
	(
	select
		datetime,
		shop,
		card,
		bonus_earned,
		bonus_spent,
		summ,
		summ_with_disc,
		doc_id,
		count(*) as records
	from
		bonuscheques
	group by 1, 2, 3, 4, 5, 6, 7, 8
) a
where
	records > 1
```

Дубликатов нет.

Количество уникальных клиентов:

```
select
	count(distinct card)
from
	bonuscheques
```

9394 уникальных клиента.

Для RFM-анализа сразу определим количество дней с последнего заказа, количество заказов и суммы, принесенных денег по каждому клиенту:

```
with const as (select max(datetime) as max_date from bonuscheques)
```

```
select
        card,
        extract(day from const.max_date::timestamp - max(bonuscheques.datetime)::timestamp) as recency_score,
          count(distinct doc_id) as frequency_score,
          sum(summ_with_disc) as monetery_score
from
        const, bonuscheques
group by
        card, max_date
```

Получилось три группы характеристик по количеству дней, проведенных с аптекой, количеству заказов и деньгам:

- **recency_score** - количество дней, прошедших с последнего заказа
- **frequency_score** - количество заказов
- **monetery_score** - количество денег, принесенных клиентами.

Запрос общей суммы прибыли:

```
with const as (select max(datetime) as max_date from bonuscheques),
sum_monetery as (select
        card,
        extract(day  from const.max_date::timestamp - max(bonuscheques.datetime)::timestamp) as recency_score,
          count(distinct doc_id) as frequency_score,
          sum(summ_with_disc) as monetery_score
from
        const, bonuscheques
group by
        card, max_date)
select sum(monetery_score)
from sum_monetery
```

- общая сумма равна 32 097 608.

Метрики

Посмотрим на медиану, моду, среднее, максимальное и минимальное значения по данным категориям:

```
with const as (select max(datetime) as max_date from bonuscheques),
rfm as (select
        card,
        extract(day from const.max_date::timestamp - max(bonuscheques.datetime)::timestamp) as recency_score,
          count(distinct doc_id) as frequency_score,
          sum(summ_with_disc) as monetery_score
from
        const, bonuscheques
group by
        card, max_date)
```

```
select
	round(avg(recency_score::numeric),
	2) as recency_mean,
	PERCENTILE_CONT(0.5) within group(
order by
	recency_score) as recency_median,
		min(recency_score) as recency_min,
		max(recency_score) as recency_max,
		mode() within group (
order by
	recency_score) as recency_mode,
	round(avg(frequency_score::numeric),
	2) as frequency_mean,
	PERCENTILE_CONT(0.5) within group(
order by
	frequency_score) as frequency_median,
		min(frequency_score) as frequency_min,
		max(frequency_score) as frequency_max,
		mode() within group (
order by
	frequency_score) as frequency_mode,
	round(avg(monetery_score::numeric),
	2) as monetery_mean,
	PERCENTILE_CONT(0.5) within group(
order by
	monetery_score) as monetery_median,
		min(monetery_score) as monetery_min,
		max(monetery_score) as monetery_max,
		mode() within group (
order by
	monetery_score) as monetery_mode
from
	rfm
```

По метрикам есть разброс между медианами и средними значениями в категории recency_score (среднее 112,92 и медиана 87) и monetery_score (среднее 3416.82 и 1586 медиана). Небольшое среднее (4,1) и медиану (2) в frequency_score и их малый разброс можно объяснить тем, что люди не часто посещают аптеки, в отличии, например, от продуктовых магазинов. Мода “1” по frequency_score тоже говорит о нечастых заказах. Посмотрим на количество мод. Запрос:

```
with const as (select max(datetime) as max_date from bonuscheques),
rfm as (select
	card,
	extract(day from const.max_date::timestamp -
max(bonuscheques.datetime)::timestamp) as recency_score,
	  count(distinct doc_id) as frequency_score,
	  sum(summ_with_disc) as monetery_score
from
	const, bonuscheques
group by
	card, max_date)
```

```
select
        count(recency_score) filter (
where
        recency_score = 2) as count_recency_mode,
                count(frequency_score) filter (
where
        frequency_score = 1) as count_frequency_mode,
                count(monetery_score) filter (
where
        monetery_score = 619) as count_monetery_mode
from
        rfm
```

- count_monetery_mode в количестве 11 особо ни о чем не говорит, а count_frequency_mode в количестве 3784 говорит, что более трети клиентов делают лишь один заказ и никогда не возвращаются. Мода 133 по count_recency_mode говорит, что небольшая часть клиентов была два дня назад. Это не особо информативно. Посмотрим, сколько клиентов было 0 и 1 день назад:

```
with const as (select max(datetime) as max_date from bonuscheques),
rfm as (select
        card,
        extract(day  from const.max_date::timestamp - max(bonuscheques.datetime)::timestamp)
as recency_score,
          count(distinct doc_id) as frequency_score,
          sum(summ_with_disc) as monetery_score
from
        const, bonuscheques
group by
        card, max_date)
select
        count(recency_score) filter (
where
        recency_score in (0, 1)) as count_recency_mode
from
        rfm
```

- 248 были сегодня (0) и день (1) назад. Половина этого числа приближена к 133. Из этого можно сделать вывод, что ежедневно аптеку посещают 120-130 человек

Однако, необходимо посмотреть, сколько денег принесли те, кто был лишь раз. Запрос:

```
with const as (select max(datetime) as max_date from bonuscheques),
rfm as (select
        card,
        extract(day  from  const.max_date::timestamp  -
max(bonuscheques.datetime)::timestamp) as recency_score,
          count(distinct doc_id) as frequency_score,
          sum(summ_with_disc) as monetery_score
from
        const, bonuscheques
group by
```

```
        card, max_date)
select
        monetery_score
from
        rfm
where frequency_score = 1
order by
        monetery_score desc
```

- лишь три клиента сделали заказ чуть более чем на 10000, остальные на меньшую сумму.

Посмотрим, сколько всего денег принесли данные клиенты всего и относительно всей суммы по monetery_score. Запрос:

```
with const as (select max(datetime) as max_date from bonuscheques),
rfm as (select
        card,
        extract(day  from const.max_date::timestamp - max(bonuscheques.datetime)::timestamp)
as recency_score,
          count(distinct doc_id) as frequency_score,
          sum(summ_with_disc) as monetery_score
from
        const, bonuscheques
group by
        card, max_date)
select
        sum(monetery_score) as sum_monetary,
        sum(monetery_score) filter (
where frequency_score = 1) as sum_filter,
        round((sum(monetery_score) filter (
where frequency_score = 1) / sum(monetery_score)) * 100,
        2) as "% monetery"
from
        rfm
```

- около 12 процентов составляет сумма, принесенная клиентами, которые сделали лишь один заказ. 12 процентов относительно всей суммы цифра немалая, но учитывая, что ее принесли более одной трети клиентов, а значит остальные деньги принесли другие две-трети клиентов, тех, кто был лишь один раз и сделал лишь один заказ при построении итогового RFM-запроса мы отсеивать не будем - любой клиент, даже сделавший один заказ, в будущем это потенциальный постоянный клиент. Так как есть разбросы в медианах и средних значениях, посмотрим и сравним данные значения с усеченными данными, уберем по 5% от минимума и максимума. Запрос:

```
with const as (select max(datetime) as max_date from bonuscheques),
rfm as (select
        card,
        extract(day  from const.max_date::timestamp - max(bonuscheques.datetime)::timestamp)
as recency_score,
          count(distinct doc_id) as frequency_score,
          sum(summ_with_disc) as monetery_score
from
```

```
        const, bonuscheques
group by
        card, max_date),
ntiled_metrics AS (
select recency_score, frequency_score, monetery_score,
                NTILE(20) OVER (ORDER BY recency_score) AS ntiled_recency,
                NTILE(20) OVER (ORDER BY frequency_score) AS ntiled_frequency,
                NTILE(20) OVER (ORDER BY monetery_score) AS ntiled_monetery,
  COUNT(*) OVER () AS counted
 FROM
  rfm)
select
        round(avg(recency_score::numeric),
        2) as recency_mean,
        PERCENTILE_CONT(0.5) within group(
order by
        recency_score) as recency_median,
                min(recency_score) as recency_min,
                max(recency_score) as recency_max,
                mode() within group (
order by
        recency_score) as recency_mode,
        round(avg(frequency_score::numeric),
        2) as frequency_mean,
        PERCENTILE_CONT(0.5) within group(
order by
        frequency_score) as frequency_median,
                min(frequency_score) as frequency_min,
                max(frequency_score) as frequency_max,
                mode() within group (
order by
        frequency_score) as frequency_mode,
        round(avg(monetery_score::numeric),
        2) as monetery_mean,
        PERCENTILE_CONT(0.5) within group(
order by
        monetery_score) as monetery_median,
                min(monetery_score) as monetery_min,
                max(monetery_score) as monetery_max,
                mode() within group (
order by
        monetery_score) as monetery_mode
FROM
 ntiled_metrics
where (ntiled_recency BETWEEN 2 AND 19) and (ntiled_frequency BETWEEN 2 AND 19) and
(ntiled_monetery BETWEEN 2 AND 19)
union
select
        round(avg(recency_score::numeric),
        2) as recency_mean,
        PERCENTILE_CONT(0.5) within group(
order by
```

```
        recency_score) as recency_median,
                min(recency_score) as recency_min,
                max(recency_score) as recency_max,
                mode() within group (
order by
        recency_score) as recency_mode,
        round(avg(frequency_score::numeric),
        2) as frequency_mean,
        PERCENTILE_CONT(0.5) within group(
order by
        frequency_score) as frequency_median,
                min(frequency_score) as frequency_min,
                max(frequency_score) as frequency_max,
                mode() within group (
order by
        frequency_score) as frequency_mode,
        round(avg(monetery_score::numeric),
        2) as monetery_mean,
        PERCENTILE_CONT(0.5) within group(
order by
        monetery_score) as monetery_median,
                min(monetery_score) as monetery_min,
                max(monetery_score) as monetery_max,
                mode() within group (
order by
        monetery_score) as monetery_mode
from
        rfm
```

- очень приближенные совпадения по средним и медианам по recency и frequency, моды по frequency совпали, небольшое расхождение мод по monetery. Разные моды по recency говорят, что люди ходят в аптеки по разному, по необходимости. Среднее усеченное значение по monetery приближено к обычному, но смысла его как-то выделять нет - товаров с разными ценами много, заказов много, потому и расхождения. Из-за небольших расхождений можно сделать вывод, что метрики по обычному не усеченному вполне подходящие.

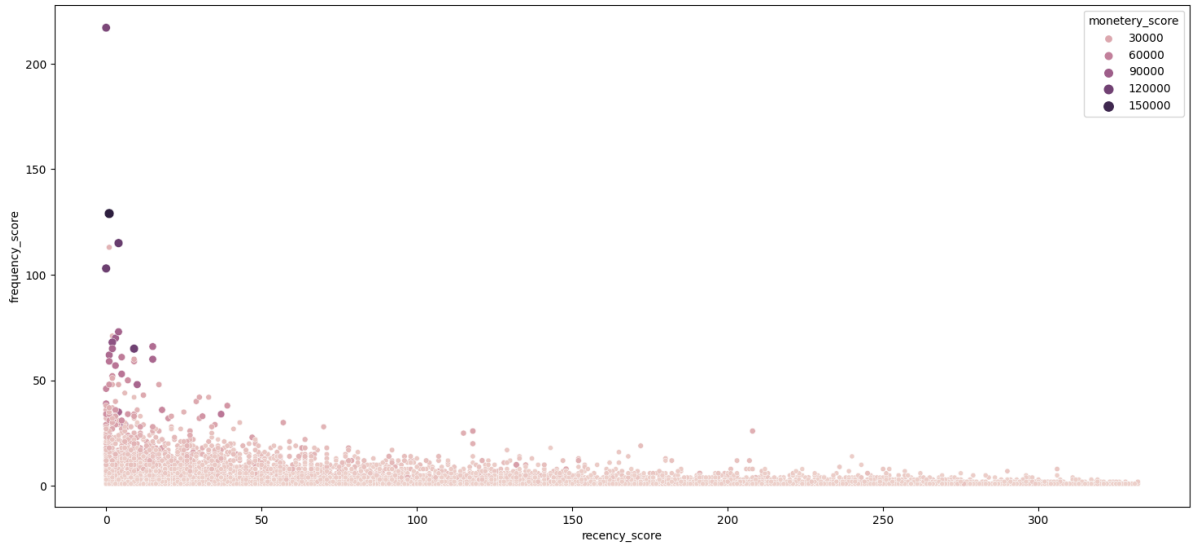
Отдельным запросом посмотрим на дисперсии и стандартное отклонеие групп:

```
with const as (
select
        max(datetime) as max_date
from
        bonuscheques),
rfm as (
select
        card,
        extract(day
from
        const.max_date::timestamp - max(bonuscheques.datetime)::timestamp) as
recency_score,
          count(distinct doc_id) as frequency_score,
```

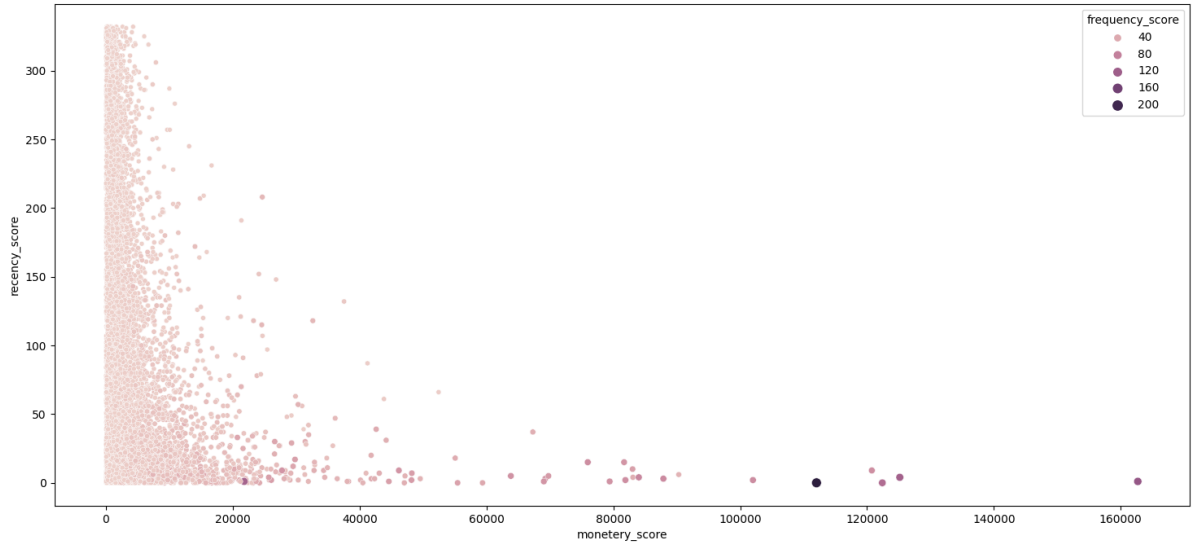
```
        sum(summ_with_disc) as monetery_score
from
        const,
        bonuscheques
group by
        card,
        max_date
having
        extract(day
from
        max(datetime)::timestamp - min(datetime)::timestamp) <> 0)
select
        'recency_score' as groups_rfm,
        variance(recency_score) as var,
        stddev(recency_score) as std
from
        rfm
union
select
        'frequency_score' as groups_rfm,
        variance(frequency_score) as var,
        stddev(frequency_score) as std
from
        rfm
union
select
        'monetery_score' as groups_rfm,
        variance(monetery_score) as var,
        stddev(monetery_score) as std
from
        rfm
```

- Стандартные отклонения подтверждают предыдущие выводы: 1) recency - кто-то приходил сегодня или вчера, а кто-то два-три месяца назад; 2) frequency - кто-то делал лишь один заказ, а кто-то является постоянным клиентом; 3) monetery - цены на товары и объемы заказов разнятся.

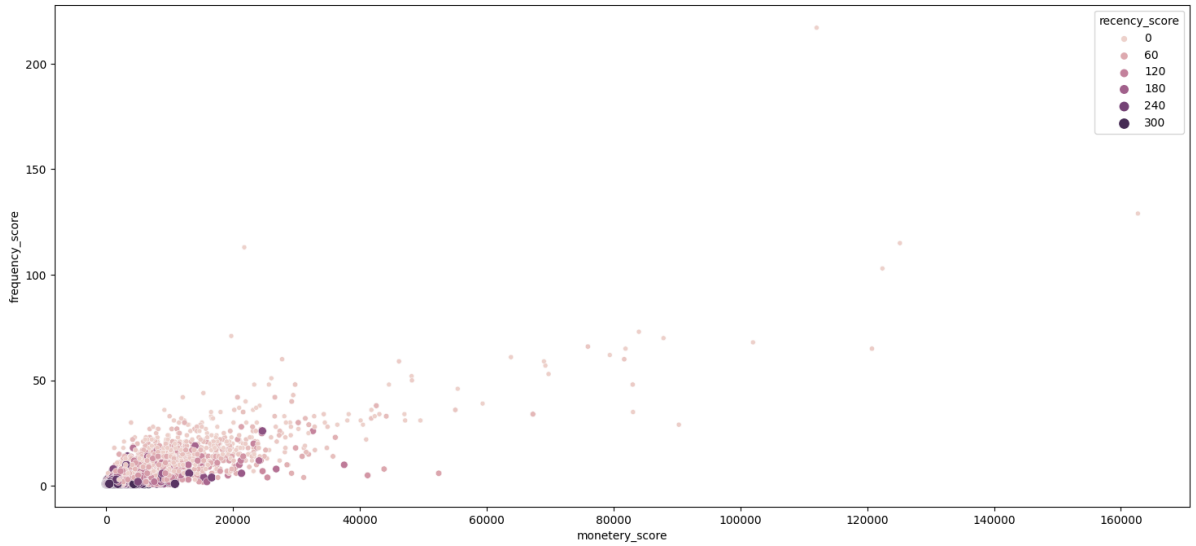
Посмотрим на график по всем трем группам. Первый по распределению recency и frequency относительно денег:



Второй распределение recency и monetery относительно frequency:



И третий распределение frequency и monetery относительно recency:



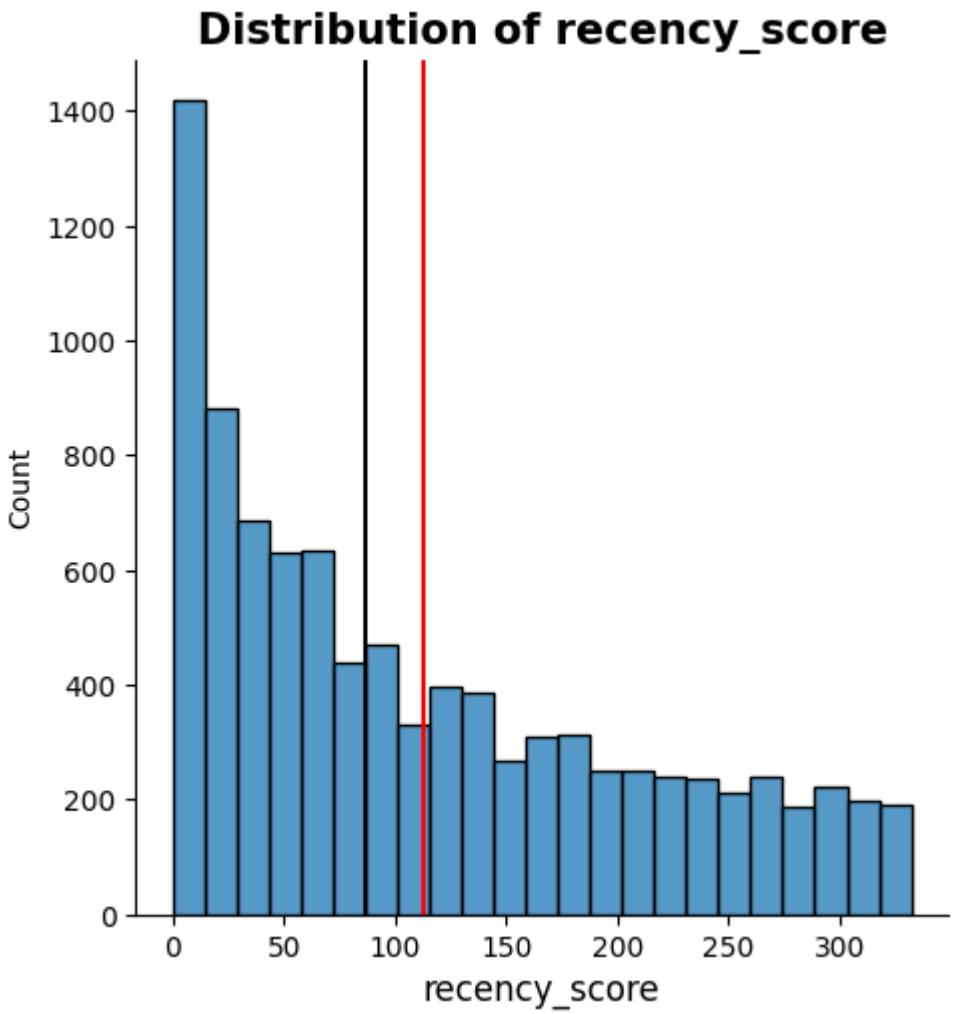
По всем трем можно сделать вывод - чем чаще клиент посещает аптеку, тем больше делает заказов и приносит денег.

Группа recency_score

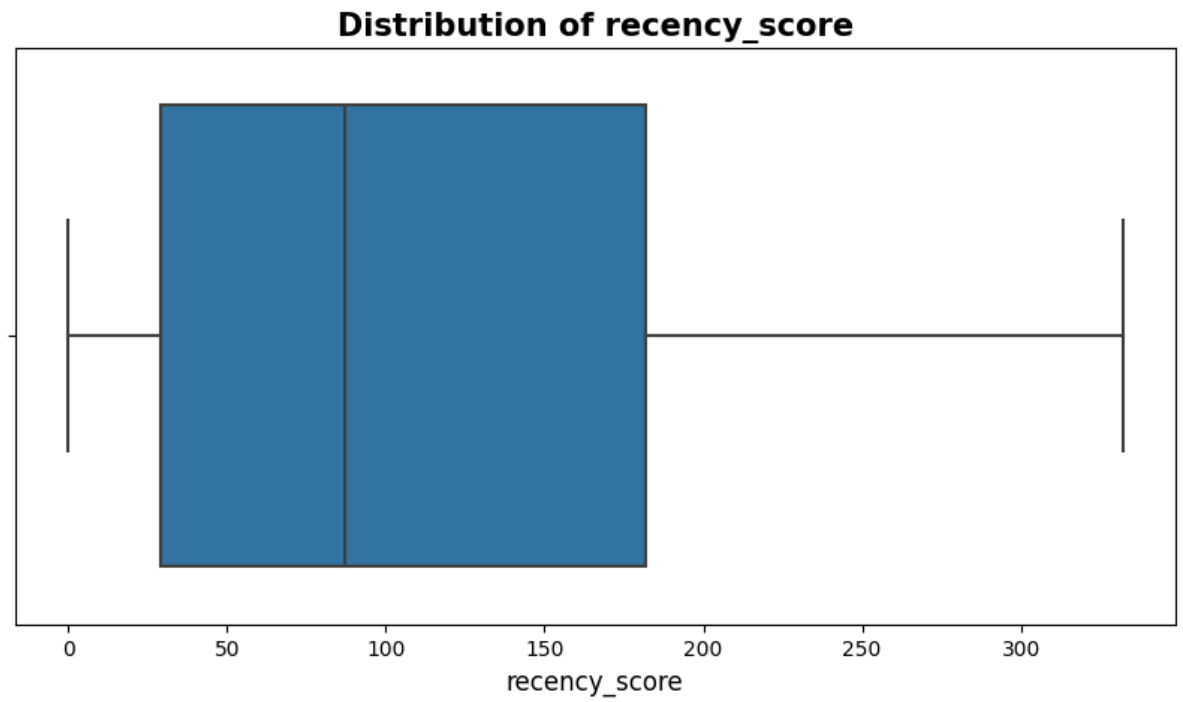
Отдельным запросом рассмотрим количество клиентов, условно разделив на группы по дням, которые скажут нам, как давно клиент сделал свой последний заказ. Так как максимальное количество дней 331 (recency_max), пусть будет четыре группы: менее 100 (ближе к среднему 112,92 по recency) дней назад, от 100 до 200, от 200 до 300, и от 300 дней. Запрос:

```
with const as (
select
	max(datetime) as max_date
from
	bonuscheques),
rfm as (
select
	card,
	extract(day
from
	const.max_date::timestamp - max(bonuscheques.datetime)::timestamp) as recency_score,
	  count(distinct doc_id) as frequency_score,
	  sum(summ_with_disc) as monetery_score
from
	const,
	bonuscheques
group by
	card,
	max_date)
select
	count(case when recency_score < 100 then recency_score end) as "< 100 days",
		count(case when recency_score >= 100 and recency_score < 200 then
recency_score end) as ">= 100 and < 200 days",
		count(case when recency_score >= 200 and recency_score < 300 then
recency_score end) as ">= 200 and < 300 days",
		count(case when recency_score >= 300 then recency_score end) as ">= 300 days"
from
	rfm
```

Результат - 5100, 2274, 1588 и 432 (если сложить все четыре итога, то получится 9394, количество уникальных клиентов, значит, потерь нет). 5100 клиентов посетили аптеку за последние сто дней, это более половины из всех. Посмотрим на график с медианой (черная линия) средним (красная линия) по recency_score:



- те же самые среднее и медиана. Еще один график:



- 75 перцентиль график показывает в районе 180 дней. Посчитаем заработанные на клиентах деньги, но учтем сначала медиану, округлив (запрос с метриками) до 90. Запрос:

```
with const as (
select
	max(datetime) as max_date
from
	bonuscheques),
rfm as (
select
	card,
	extract(day
from
	const.max_date::timestamp - max(bonuscheques.datetime)::timestamp) as
recency_score,
	  count(distinct doc_id) as frequency_score,
	  sum(summ_with_disc) as monetery_score
from
	const,
	bonuscheques
group by
	card,
	max_date)
select
	sum(monetery_score) as sum_monetary,
	sum(monetery_score) filter (
where
	recency_score < 90) as min_sum_filter,
	sum(monetery_score) filter (
where
```

```
        recency_score >= 90
        and recency_score < 180) as per_sum_filter,
        sum(monetery_score) filter (
where
        recency_score >= 180) as max_sum_filter,
        round((sum(monetery_score) filter (
where
        recency_score < 90) / sum(monetery_score)) * 100,
        2) as "%"
from
        rfm
```

- 23515509 приносят клиенты, которые сделали заказ за последние 90 дней, это 73% от всей суммы. Нужно посчитать количество этих клиентов. Запрос:

```
with const as (
select
        max(datetime) as max_date
from
        bonuscheques),
rfm as (
select
        card,
        extract(day
from
        const.max_date::timestamp - max(bonuscheques.datetime)::timestamp) as recency_score,
          count(distinct doc_id) as frequency_score,
          sum(summ_with_disc) as monetery_score
from
        const,
        bonuscheques
group by
        card,
        max_date)
select
        count(monetery_score) as count_monetary,
        count(monetery_score) filter (
where
        recency_score < 90) as min_count_filter,
        count(monetery_score) filter (
where
        recency_score >= 90
        and recency_score < 180) as per_count_folter,
        count(monetery_score) filter (
where
        recency_score >= 180) as max_count_filter
from
        rfm
```

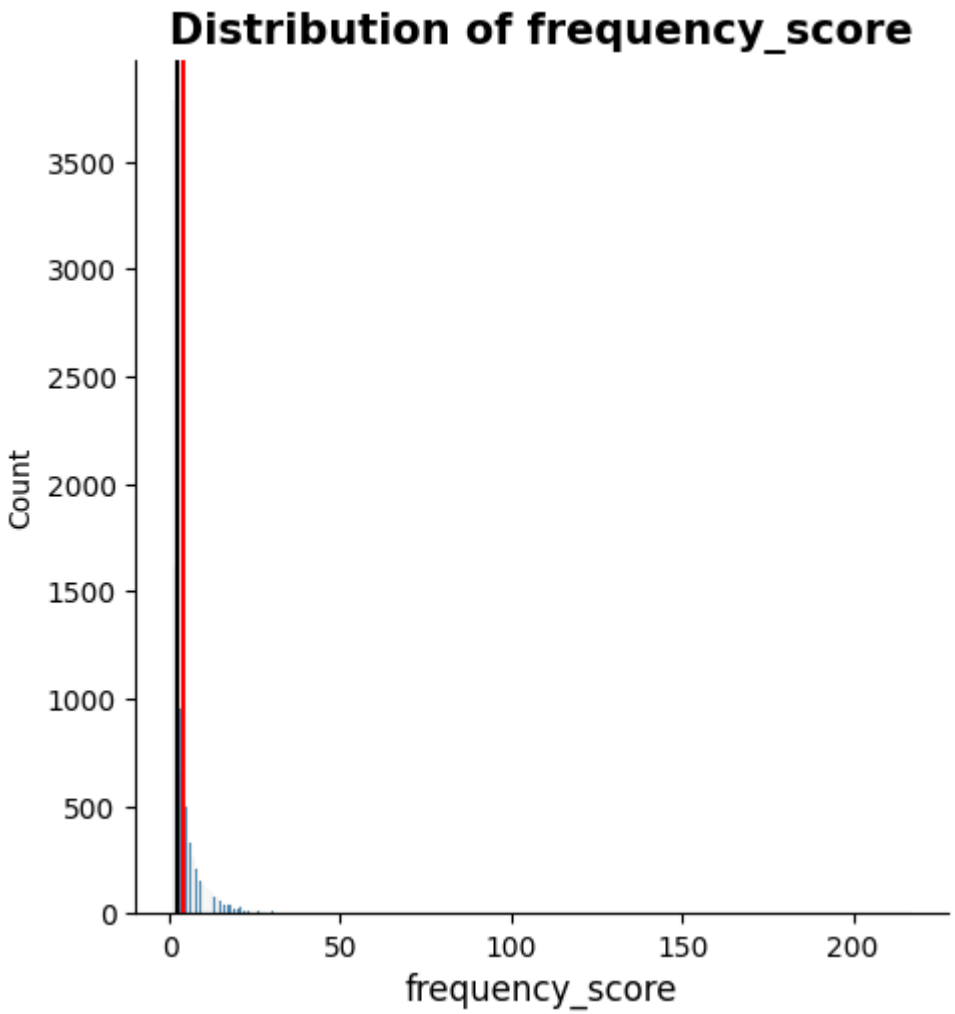
- 4787 клиентов, это приблизительно половина клиентов. 75-й перцентиль (recency_score >= 180) также станет ориентиром для финального расчета RFM. Среднее в таком случае рассматривать не будем.

Группа frequency_score

По метрикам данная группа хотя и показала вполне логичные значения по среднему и медиане (люди не часто ходят в аптеки), все же разброс довольно велик - 217 заказов как максимальное. Поэтому будем исходить из среднего:

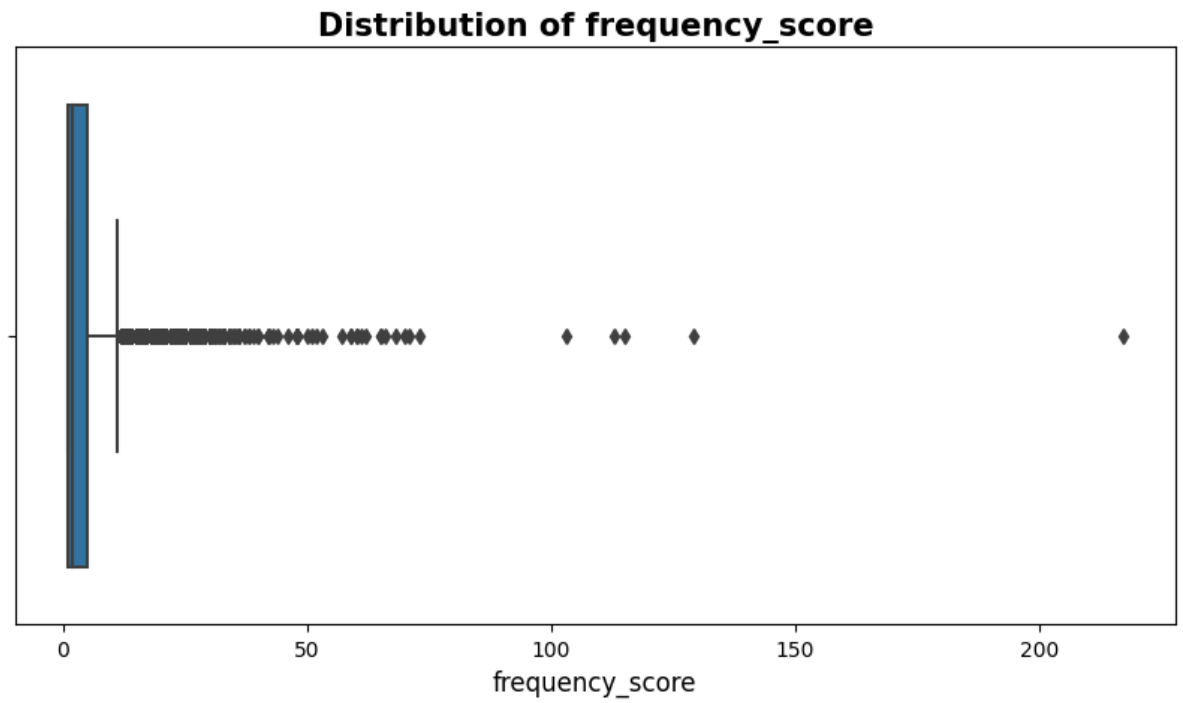
```
with const as (
select
	max(datetime) as max_date
from
	bonuscheques),
rfm as (
select
	card,
	extract(day
from
	const.max_date::timestamp - max(bonuscheques.datetime)::timestamp) as
recency_score,
	  count(distinct doc_id) as frequency_score,
	  sum(summ_with_disc) as monetery_score
from
	const,
	bonuscheques
group by
	card,
	max_date)
select
	count(case when frequency_score < 5 then frequency_score end) as "< 5",
		count(case when frequency_score >= 5 and frequency_score < 10 then
frequency_score end) as ">= 5 and < 10",
		count(case when frequency_score >= 10 and frequency_score < 150 then
frequency_score end) as ">= 10 and < 150",
		count(case when frequency_score >= 150 and frequency_score < 200 then
frequency_score end) as ">= 150 and < 200",
		count(case when frequency_score >= 200 then frequency_score end) as ">=
200"
from
	rfm
```

- 7000 клиентов сделали менее пяти заказов. График с медианой (черная линия) и средним (красная линия):



- те же среднее 4 и медиана 2, что в запросе выше по метрикам.

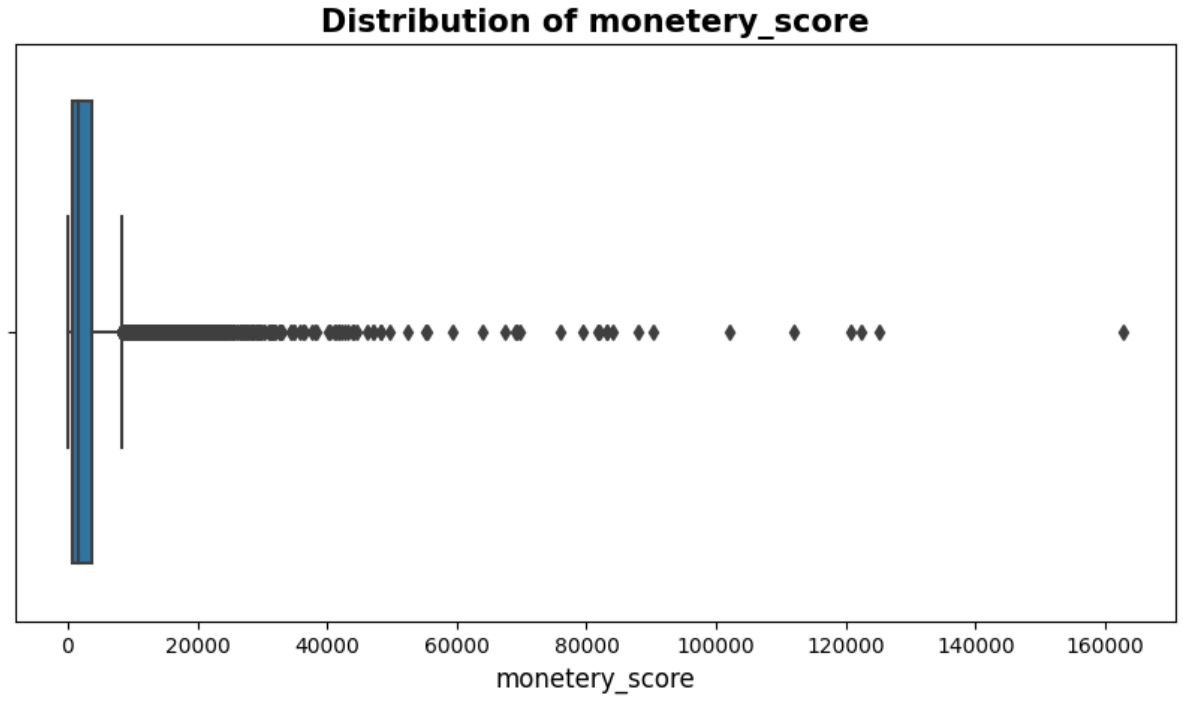
Снова применим график boxplot, теперь уже по группе frequency_score:

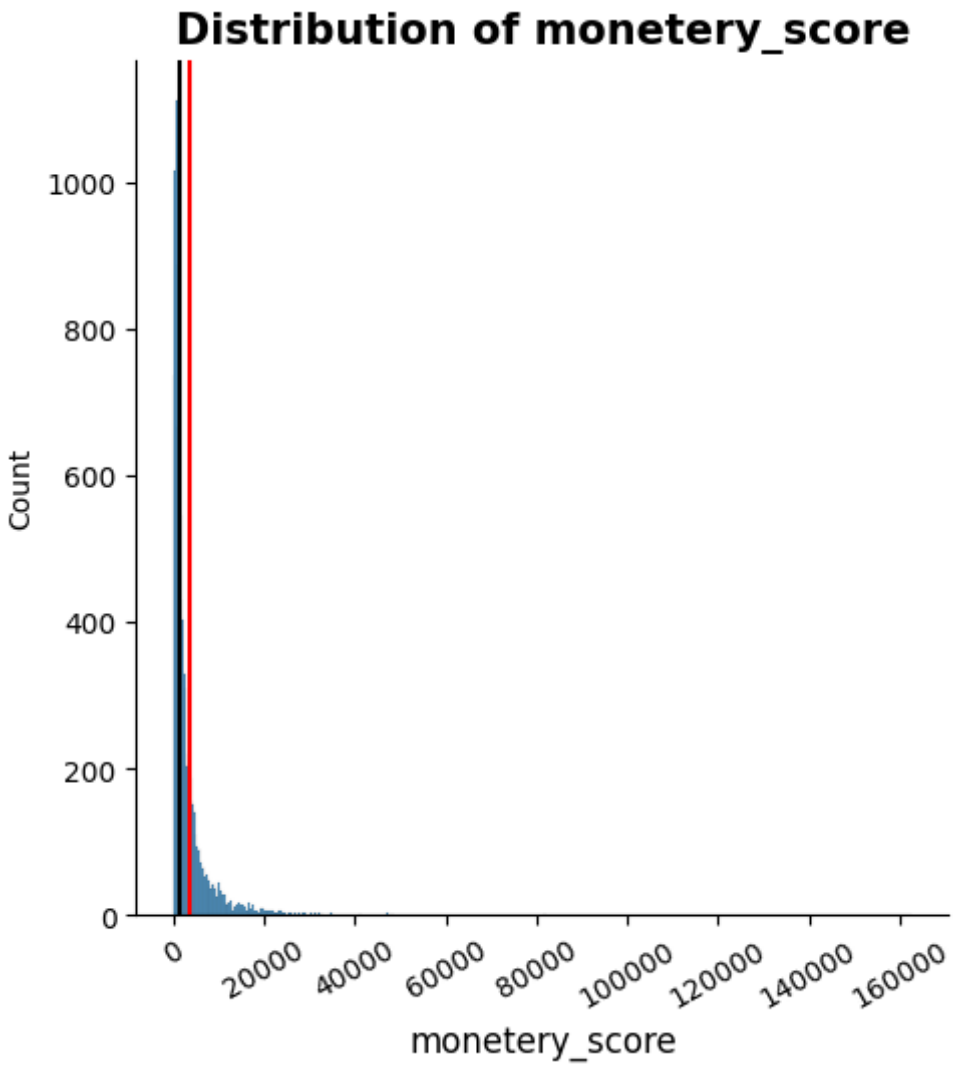


И по нему можно подтвердить выводы о малом количестве заказов у большинства клиентов. Однако есть и две другие группы (1446 и 945 клиентов), сделавшие больше 5 заказов.

Группа monetery_score

Теперь обратим внимание на группу monetery_score и повторим тот же запрос и рассмотрим такие же графики. Но начнем с графиков:





По обоим графикам видно, что большинство клиентов тратят менее 20000 в аптеке, хотя и есть выброс в более 160000. Продолжим условно, в тоже время исходя из графиков, деление клиентов на группы и посмотрим на их количество. Запрос:

```
with count_scores as (
select
	card,
	sum(summ_with_disc) as monetery_score
from
	bonuscheques
group by
	card)
select
	count(case when monetery_score < 20000 then monetery_score end) as "< 20000",
		count(case when monetery_score >= 20000 and monetery_score < 40000 then
frequency_score end) as ">= 20000 and < 40000",
		count(case when monetery_score >= 40000 and monetery_score < 100000 then
frequency_score end) as ">= 40000 and < 100000",
		count(case when monetery_score >= 100000 then monetery_score end) as ">=
100000"
from
```

count_scores

- из 9394 клиентов (9218 + 134 + 36 + 6 = 9394 - потерь по клиентам нет) 9218 приносят менее 20000 аптеке.

RFM- запрос

Исходя из проделанного анализа составим RFM-запрос. Цифрой “1” обозначим тех клиентов, кто был в аптеке не позднее 90 дней (медианное значение по recency) относительно максимальной даты по всей базе данных, сделал не менее 50 заказов, а также принесшие аптеке не менее 100000. Цифрой “3” обозначим тех клиентов, которые были в аптеке более 180 дней (75-й перцентиль по боксплоту) назад, кто сделал менее 5 заказов (среднее по frequency_score). Минимальное качественное значение по monetery_score, исходя из анализа преобразованных данных, возьмем медиану, округлив до 1500. Цифрой “2” обозначим всех остальных. Запрос:

```
with const as (
select
	max(datetime) as max_date
from
	bonuscheques),
count_scores as (
select
	card,
	extract(day
from
	const.max_date::timestamp - max(bonuscheques.datetime)::timestamp) as recency_score,
	  count(distinct doc_id) as frequency_score,
	  sum(summ_with_disc) as monetery_score
from
	const,
	bonuscheques
group by
	card,
	max_date),
rfm as (
select
	card,
	recency_score,
	frequency_score,
	monetery_score,
		case
		when recency_score < 90 then '1'
		when recency_score >= 90
		and recency_score < 180 then '2'
		when recency_score >= 180 then '3'
	end as R,
		case
		when frequency_score < 5 then '3'
		when frequency_score >= 5
```

```
            and frequency_score < 50 then '2'
            when frequency_score >= 50 then '1'
        end as F,
            case
            when monetery_score < 1500 then '3'
            when monetery_score >= 1500
            and monetery_score < 100000 then '2'
            when monetery_score >= 100000 then '1'
        end as M
from
        count_scores
group by
        card,
        recency_score,
        frequency_score,
        monetery_score)
select
        card,
        recency_score,
        frequency_score,
        monetery_score,
        R,
        F,
        M,
        concat(R,
        F,
        M) as RFM
from rfm
order by rfm
```

В этом же запросе мы “складываем” эти цифры. В результате получились вариации качества клиентов:

111 недавние, частые, высокий чек	**121** недавние, редкие, высокий чек	**131** Недавние, разовые, высокий чек
112 недавние, частые, средний чек	**122** недавние, редкие, средний чек	**132** недавние, разовые, средний чек
113 недавние, частые, низкий чек	**123** недавние, редкие, низкий чек	**133** недавние, разовые, низкий чек
211 спящие, частые, высокий чек	**221** спящие, редкие, средний чек	**231** спящие, разовые, высокий чек
212 спящие, частые, средний чек	**222** спящие, редкие, средний чек	**232** спящие, разовые, средний чек
213 спящие, частые, низкий чек	**223** спящие, редкие, низкий чек	**233** спящие, разовые, низкий чек
311 уходящие, частые, высокий чек	**321** уходящие, редкие, высокий чек	**331** уходящие, разовые, высокий чек
312 уходящие, частые, средний чек	**322** уходящие, редкие, средний чек	**332** уходящие, разовые, средний чек
313 уходящие, частые, низкий чек	**323** уходящие, редкие, низкий чек	**333** уходящие, разовые, низкий чек

Клиенты “262be3a0-1838-4e35-804c-ad5bf22c4fe7”, “2000200196600”, “2000200189985”, “2000200204541”, “2000200170860”, “2000200196556” с качеством “111” являются самыми важными и в большей степени показательными или идеальными: лишь один был 9 дней назад, лишь у двух заказов меньше 70, но все шесть принесли более 100000 каждый. Такими же показательными являются клиенты с качеством “333”: были очень давно, заказов мало, денег приносят мало. Отдельным запросом можно достать количество клиентов и средние значения во всех качественных группах:

```
with const as (
select
	max(datetime) as max_date
from
	bonuscheques),
count_scores as (
select
	card,
	extract(day
from
	const.max_date::timestamp - max(bonuscheques.datetime)::timestamp) as recency_score,
	  count(distinct doc_id) as frequency_score,
	  sum(summ_with_disc) as monetery_score
from
	const,
	bonuscheques
group by
	card,
	max_date),
rfm as (
select
	card,
	recency_score,
	frequency_score,
	monetery_score,
		case
		when recency_score < 90 then '1'
		when recency_score >= 90
		and recency_score < 180 then '2'
		when recency_score >= 180 then '3'
	end as R,
		case
		when frequency_score < 5 then '3'
		when frequency_score >= 5
		and frequency_score < 50 then '2'
		when frequency_score >= 50 then '1'
	end as F,
		case
		when monetery_score < 1500 then '3'
		when monetery_score >= 1500
		and monetery_score < 100000 then '2'
		when monetery_score >= 100000 then '1'
	end as M
```

```
from
	count_scores
group by
	card,
	recency_score,
	frequency_score,
	monetery_score),
count_rfm as (
select
	card,
	recency_score,
	frequency_score,
	monetery_score,
	R,
	F,
	M,
	concat(R,
	F,
	M) as RFM
from
	rfm)
select
	'count' as groups_rfm,
	count(case when RFM = '111' then '111' end) as "111",
	count(case when RFM = '112' then '112' end) as "112",
	count(case when RFM = '122' then '112' end) as "122",
	count(case when RFM = '123' then '123' end) as "123",
	count(case when RFM = '132' then '132' end) as "132",
	count(case when RFM = '133' then '133' end) as "133",
	count(case when RFM = '222' then '222' end) as "222",
	count(case when RFM = '223' then '223' end) as "223",
	count(case when RFM = '232' then '232' end) as "232",
	count(case when RFM = '233' then '233' end) as "233",
	count(case when RFM = '322' then '322' end) as "322",
	count(case when RFM = '323' then '323' end) as "323",
	count(case when RFM = '332' then '332' end) as "332",
	count(case when RFM = '333' then '333' end) as "333"
from
	count_rfm
union
select
	'avg_recency' as groups_rfm,
	round(avg(case when RFM = '111' then recency_score end)) as "111",
	round(avg(case when RFM = '112' then recency_score end)) as "112",
	round(avg(case when RFM = '122' then recency_score end)) as "122",
	round(avg(case when RFM = '123' then recency_score end)) as "123",
	round(avg(case when RFM = '132' then recency_score end)) as "132",
	round(avg(case when RFM = '133' then recency_score end)) as "133",
	round(avg(case when RFM = '222' then recency_score end)) as "222",
	round(avg(case when RFM = '223' then recency_score end)) as "223",
	round(avg(case when RFM = '232' then recency_score end)) as "232",
	round(avg(case when RFM = '233' then recency_score end)) as "233",
```

```
        round(avg(case when RFM = '322' then recency_score end)) as "322",
        round(avg(case when RFM = '323' then recency_score end)) as "323",
        round(avg(case when RFM = '332' then recency_score end)) as "332",
        round(avg(case when RFM = '333' then recency_score end)) as "333"
from
        count_rfm
union
select
        'avg_frequency' as groups_rfm,
        round(avg(case when RFM = '111' then frequency_score end)) as "111",
        round(avg(case when RFM = '112' then frequency_score end)) as "112",
        round(avg(case when RFM = '122' then frequency_score end)) as "122",
        round(avg(case when RFM = '123' then frequency_score end)) as "123",
        round(avg(case when RFM = '132' then frequency_score end)) as "132",
        round(avg(case when RFM = '133' then frequency_score end)) as "133",
        round(avg(case when RFM = '222' then frequency_score end)) as "222",
        round(avg(case when RFM = '223' then frequency_score end)) as "223",
        round(avg(case when RFM = '232' then frequency_score end)) as "232",
        round(avg(case when RFM = '233' then frequency_score end)) as "233",
        round(avg(case when RFM = '322' then frequency_score end)) as "322",
        round(avg(case when RFM = '323' then frequency_score end)) as "323",
        round(avg(case when RFM = '332' then frequency_score end)) as "332",
        round(avg(case when RFM = '333' then frequency_score end)) as "333"
from
        count_rfm
union
select
        'avg_monetary' as groups_rfm,
        round(avg(case when RFM = '111' then monetery_score end)) as "111",
        round(avg(case when RFM = '112' then monetery_score end)) as "112",
        round(avg(case when RFM = '122' then monetery_score end)) as "122",
        round(avg(case when RFM = '123' then monetery_score end)) as "123",
        round(avg(case when RFM = '132' then monetery_score end)) as "132",
        round(avg(case when RFM = '133' then monetery_score end)) as "133",
        round(avg(case when RFM = '222' then monetery_score end)) as "222",
        round(avg(case when RFM = '223' then monetery_score end)) as "223",
        round(avg(case when RFM = '232' then monetery_score end)) as "232",
        round(avg(case when RFM = '233' then monetery_score end)) as "233",
        round(avg(case when RFM = '322' then monetery_score end)) as "322",
        round(avg(case when RFM = '323' then monetery_score end)) as "323",
        round(avg(case when RFM = '332' then monetery_score end)) as "332",
        round(avg(case when RFM = '333' then monetery_score end)) as "333"
from
        count_rfm
union
select
        'sum_monetary' as groups_rfm,
        sum(case when RFM = '111' then monetery_score end) as "111",
        sum(case when RFM = '112' then monetery_score end) as "112",
        sum(case when RFM = '122' then monetery_score end) as "122",
        sum(case when RFM = '123' then monetery_score end) as "123",
        sum(case when RFM = '132' then monetery_score end) as "132",
```

```
        sum(case when RFM = '133' then monetery_score end) as "133",
        sum(case when RFM = '222' then monetery_score end) as "222",
        sum(case when RFM = '223' then monetery_score end) as "223",
        sum(case when RFM = '232' then monetery_score end) as "232",
        sum(case when RFM = '233' then monetery_score end) as "233",
        sum(case when RFM = '322' then monetery_score end) as "322",
        sum(case when RFM = '323' then monetery_score end) as "323",
        sum(case when RFM = '332' then monetery_score end) as "332",
        sum(case when RFM = '333' then monetery_score end) as "333"
from
        count_rfm
```

groups_rfm	111	112	122	123	132	133	222	223	232	233	322	323	332	333
avg_frequency	116	64	11	6	3	2	7	5	2	1	7	6	2	1
avg_monetary	124 177	58 873	8 937	1 095	3 098	763	6 021	1 152	3 203	737	5 214	1 069	2 879	683
avg_recency	3	5	28	32	39	41	124	134	129	135	221	231	244	255
count	6	17	1 873	46	1 203	1 642	344	24	715	1 115	75	7	630	1 697
sum_monetary	745 063	1 000 843	16 738 818	50 365	3 727 084	1 253 336	2 071 372	27 639	2 290 284	821 815	391 040	7 480	1 813 729	1 158 740

Как и по количеству клиентов, по сумме денег потерь нет.

“**111**” - были в аптеке недавно, очень часто совершают покупки и потому приносят много денег. К ним можно отнести и клиентов с качеством “**112**”: те же частые посещения и заказы несмотря на меньший, но все равно внушительный средний чек. Для удержания подобных клиентов нужны персональные именные карты с большими скидками. Возможны даже какие либо консультации или личные с ними встречи, чтобы выяснить в чем конкретно они постоянно нуждаются и в каких объемах. Если много заказывают и часто, есть вероятность оптовых покупок на взаимных с аптекой условиях. И их небольшое количество (6 и 17), а значит при правильных расчетах на привлечение их внимания будет нужно немного рекламно-маркетинговых средств.

“**333**” - имеют низкий средний чек (как раз немногим выше моды по метрикам), были очень давно и имеют лишь одну покупку по среднему. Не смотря на то, что чуть более полутора тысяч клиентов принесли в сумме немногим более миллиона, каждый клиент важен и может стать постоянным, или хотя бы зайти в аптеку еще раз. Возможно для них подойдут стандартные массовые рассылки, которые могут охватить практически всех, несмотря на качество. Например, сезонные скидки на препараты от простуды к началу зимы. К ним можно отнести и клиентов с качеством “**323**” и “**332**”, которые делают чуть больше покупок (от 2 до 6), а значит могут вполне стать постоянными. Клиенты с качеством “**233**”, “**232**” и “**223**” также были давно, в районе трех-четырех месяцев по среднему, и какое нибудь смс-напоминание может их привлечь.

Клиенты с качеством “**322**” имеют в два-три раза больший средний чек, чем предыдущие, и неплохое количество покупок (7 по среднему), хотя также были очень давно. Ориентиром для их возвращения может стать анализ товаров, которые они приобретали в своих заказах. И если в аптеке планируются какие либо скидки и акции на конкретные позиции или группы товаров, стоит вспомнить об этих, пусть и давних, но вполне выгодных клиентах

Клиенты с качествами “**133**”, “**132**” и “**123**” похожи на предыдущих, но с отличием - они были в аптеке недавно, в районе месяца по среднему. В отличие от давних, в их памяти еще могло сохранится посещение аптеки и их заказ. Поэтому им тоже подойдут массовые рассылки со скидками и выгодными предложениями. Если будут какие либо ограничения рекламных средств на привлечение покупателей, именно те, кто был недавно, должны быть в приоритете.

Клиенты “**222**” имеют очень хороший средний чек и количество заказов, при этом находятся в зоне их потери, так как были несколько месяцев назад. С ними схожи клиенты “**122**”, которые имеют чуть больший средний чек и принесли больше всего денег аптеке и при этом были сравнительно недавно, в районе месяца по среднему. Именно количество денег, принесенных их заказами, должны стать ориентиром для их привлечения. А для этого необходимо проанализировать корзины продуктов в их покупках: какие, количество, качество и даже срок годности. Из этого анализа покупок и их периодичности и стоит выстроить стратегию привлечения и удержания данной группы клиентов.