

Задание 1. А/В–тестирование

1.1 Условие

Одной из основных задач аналитика в нашей команде является корректное проведение экспериментов. Для этого мы применяем метод А/В–тестирования. В ходе тестирования одной гипотезы целевой группе была предложена новая механика оплаты услуг на сайте, у контрольной группы оставалась базовая механика. В качестве задания вам необходимо проанализировать итоги эксперимента и сделать вывод, стоит ли запускать новую механику оплаты на всех пользователей.

1.2 Входные данные

В качестве входных данных вы имеете 4 csv-файла:

- groups.csv – файл с информацией о принадлежности пользователя к контрольной или экспериментальной группе (**A – контроль, B – целевая группа**)
- groups_add.csv – дополнительный файл с пользователями, который вам прислали спустя 2 дня после передачи данных
- active_studs.csv – файл с информацией о пользователях, которые зашли на платформу в дни проведения эксперимента.
- checks.csv – файл с информацией об оплатах пользователей в дни проведения эксперимента.

1.3 Вопросы

Предлагаем вам ответить на следующие вопросы:

- На какие метрики вы смотрите в ходе анализа и почему?
- Имеются ли различия в показателях и с чем они могут быть связаны?
- Являются ли эти различия статистически значимыми?
- Стоит ли запускать новую механику на всех пользователей?
- Данный список вопросов не является обязательным, и вы можете при своём ответе опираться на собственный план.

1.4 Требования к ответу

При выполнении тестового задания необходимо использовать язык программирования Python. В файле обязательно должны быть комментарии к тем действиям, которые вы выполняете с данными. Файл должен представлять собой законченный отчёт с выводами, сделанными в ходе исследования.

In [2]: 1 !pip install requests

```
Requirement already satisfied: requests in c:\users\ainzha\anaconda3\lib\site-packages (2.31.0)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\ainzha\anaconda3\lib\site-packages (from requests) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\ainzha\anaconda3\lib\site-packages (from requests) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\ainzha\anaconda3\lib\site-packages (from requests) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\ainzha\anaconda3\lib\site-packages (from requests) (2024.2.2)
```

In [3]: 1 import numpy as np
2 import pandas as pd

In [8]: 1 import requests # для запросов
2 from io import StringIO # для чтения текста как файла
3
4 #скачиваем csv по публичной ссылке яндекс диска и возвращает pd.DataFrame
5 def download_csv_from_disk(public_url):
6 api_url = 'https://cloud-api.yandex.net/v1/disk/public/resources/download
7 params = {'public_key': public_url} # параметры запроса
8
9 resp = requests.get(api_url, params=params) # запрос к апи
10 resp.raise_for_status() # проверка ответа
11 download_url = resp.json()['href'] # прямая ссылка на файл
12
13 file_resp = requests.get(download_url) # скачиваем файл
14 file_resp.raise_for_status() # проверка скачивания
15
16 text = file_resp.text # текст файла
17 sep = ',' if text.count(',') >= text.count(';') else ';' # определяем разделитель
18
19 return pd.read_csv(StringIO(text), sep=sep) # в dataframe
20
21 #загрузим все датасеты
22 groups = download_csv_from_disk('https://disk.yandex.ru/d/UhyYx41rTt3c1Q')
23 groups_add = download_csv_from_disk('https://disk.yandex.ru/d/5Kxrz02m3IBUwQ')
24 active_studs = download_csv_from_disk('https://disk.yandex.ru/d/Tbs44Bm6H_Fw')
25 checks = download_csv_from_disk('https://disk.yandex.ru/d/pH1q-VqcxXjsVA')

In [38]: 1 checks.head()

Out[38]: student_id rev

	student_id	rev
0	1627	990.0
1	3185	690.0
2	25973	690.0
3	26280	690.0
4	100300	990.0

In [39]: 1 groups_add.head()

Out[39]:

	id	grp
0	5694584	B
1	5694830	B
2	5695057	B
3	5698872	B
4	5699067	B

In [40]: 1 groups.head()

Out[40]:

	id	grp
0	1489	B
1	1627	A
2	1768	B
3	1783	B
4	1794	A

In [41]: 1 active_studs.head()

Out[41]:

	student_id
0	581585
1	5723133
2	3276743
3	4238589
4	4475369

Тк датасет group_add у нас добавочный, нужно проверить не продублируются ли данные два раза, если мы соединим group b group_add. Проверим, учтены ли данные в датасетах

In [45]: 1 groups_add['id'].isin(groups['id']).any()

Out[45]: False

In [46]: 1 groups_add['id'].isin(chcks['student_id']).any()

Out[46]: True

In [47]: 1 groups_add['id'].isin(active_studs['student_id']).any()

Out[47]: True

Видим, что из добавочное таблицы (groups_add) не учтены в groups, то соединим их.

In [48]:

```
1 group_full = pd.concat([groups_add, groups], ignore_index=True)
2 group_full.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 74576 entries, 0 to 74575
Data columns (total 2 columns):
 #   Column Non-Null Count Dtype  
--- 
 0   id      74576 non-null  int64  
 1   grp     74576 non-null  object 
dtypes: int64(1), object(1)
memory usage: 1.1+ MB
```

Проверим все ли данные из таблицы с оплатами учтены в активных пользователях

In [49]:

```
1 # создаем пустой список для неактивных пользователей
2 not_active_users = []
3
4 # проходим по всем student_id из таблицы с оплатами
5 for student_id in checks['student_id']:
6     # если пользователя нет среди активных
7     if student_id not in list(active_studs['student_id']):
8         not_active_users.append(student_id)
9
10 active_checks = len(not_active_users)
11 print(active_checks, "пользователей совершили оплату, но их нет в таблице ак")
12
```

149 пользователей совершили оплату, но не считаются активными

Исключим пользователей, которые есть в таблице checks, но нет в таблице active_studs. Они не нужны для нашего анализа. Поэтому удалим из таблицы checks неактивных пользователей , тк нам нужны только пользователи, которые зашли на платформу в дни проведения эксперимента.

In [53]:

```
1 checks = checks[checks['student_id'].isin(active_studs['student_id'])] #убираем неактивных
2 checks.count()
```

Out[53]:

student_id	392
rev	392
	dtype: int64

Метрики!!

Изначально проведение A/B тестов планируется из-за сравнения новой механики и базовой(старой) оплаты услуг на сайте . Это означает, что пользователи не доходили до конечной цели (оплаты-приобретения продукта), возможно из-за сложной схемы оплаты. Значит отсюда можно понять, что основными метриками будут конверсии в платящего пользователя. Основная метрика - CR.

- CR - конверсия активного пользователя в платящеого.
- ARPU - средний чек по пользователю.
- ARPPU - средний чек по платящему пользователю.

In [68]:

```

1 #переименуем столбцы, чтобы смерджить таблицы
2 active_studs = active_studs.rename(columns={'student_id':'id'})
3 checks = checks.rename(columns={'student_id':'id'})
4 #создадим один общий датафрейм, соединив все нужные таблицы в одну
5 full_df = active_studs.merge(checks, how='left', on='id').merge(group_full,

```

In [69]:

```
1 full_df.head()
```

Out[69]:

	id	rev	grp
0	581585	NaN	A
1	5723133	NaN	A
2	3276743	NaN	B
3	4238589	NaN	A
4	4475369	NaN	B

In [72]:

```
1 full_df.shape
```

Out[72]:

(8341, 3)

In [71]:

```
1 full_df.id.nunique()
```

Out[71]:

8341

Число уникальных id совпадает с общим числом id. Значит, выручка от всех покупок каждого конкретного пользователя уже сведена в единое значение, и записи не нужно группировать по id студентов.

Теперь нужно сгруппировать данные по тестовой и контрольной группам.

In [95]:

```

1 # группируем по A и B и считаем количество записей по столбцам id и rev
2 grouped = full_df.groupby('grp', as_index=False)[['id', 'rev']].count()
3 # добавляем колонку с конверсией: платящие / активные
4 grouped['active_cr'] = grouped['rev'] / grouped['id']
5 grouped = grouped.rename(columns={'id': 'n_active_students', 'rev': 'n_paying_'
6 grouped

```

Out[95]:

	grp	n_active_students	n_paying_students	active_cr
0	A	1538	78	0.050715
1	B	6803	314	0.046156

Отсюда мы видим, что конверсии не сильно различимы. Хотя конверсия в контрольной группе даже чуть больше чем в тестовой. Нужно удостовериться, что новая механика не

Гипотезы:

- H0 - нет никакой взаимосвязи между сг и группами(различия статистически незначимы)
 - H1 - различия статистически значимы(сг зависит от групп)
- Здесь применим критерий ХИ-квадрат, тк у нас две категориальные переменные, либо *купили - 1*, либо *не купили - 0*.

```
In [96]: 1 full_df['status'] = np.where(full_df['rev'].fillna(0) > 0, '1', '0')
```

```
In [97]: 1 full_df.head()
```

Out[97]:

	id	rev	grp	status
0	581585	NaN	A	0
1	5723133	NaN	A	0
2	3276743	NaN	B	0
3	4238589	NaN	A	0
4	4475369	NaN	B	0

```
In [103]: 1 #таблица сопряженности
2 crosstab = pd.crosstab(full_df.status, full_df.grp)
3 crosstab
```

Out[103]:

grp	A	B
status		
0	1460	6489
1	78	314

```
In [104]: 1 from scipy import stats
```

```
In [106]: 1 chi2_stat, p_value, dof, expected = stats.chi2_contingency(crosstab)
2 print(f"критерий хи-квадрат: {chi2_stat}, p-value: {p_value}")
```

критерий хи-квадрат: 0.48480826522557896, p-value: 0.48625240457231045

```
In [112]: 1 if p_value>0.05:
2     print(f" Принимаем H0. p-value - {p_value}.\n Различия статистически нез
3 else:
4     print(f"Отклоняем H0. p-value - {p_value} ")
```

Принимаем H0. p-value - 0.48625240457231045.

Различия статистически незначимы!

Критерий Хи-квадрат показывает, что различия между значениями конверсии из активного пользователя в платящего в двух группах не являются стат значимыми. Это означает, что новая механика проведения оплаты не имеет как таковой пользы. Она не повышает нашу главную метрику.

In [116]: 1 full_df

Out[116]:

	id	rev	grp	status
0	581585	NaN	A	0
1	5723133	NaN	A	0
2	3276743	NaN	B	0
3	4238589	NaN	A	0
4	4475369	NaN	B	0
...
8336	1794344	NaN	B	0
8337	296883	NaN	B	0
8338	3598448	NaN	B	0
8339	3610547	290.0	B	1
8340	205705	NaN	B	0

8341 rows × 4 columns

In [119]:

```

1 #считаем ARPU
2 arpu = full_df.groupby('grp', as_index=False).agg({'rev': 'sum', 'id': 'count'}
3 arpu['ARPU'] = (arpu['rev'] / arpu['id']).round(2)
4 arpu = arpu.rename(columns={'id': 'n_active_users', 'rev': 'total_revenue'})
5
6 # считаем ARPPU
7 arppu = full_df[full_df['rev'] > 0].groupby('grp', as_index=False).agg({'rev':
8 arppu['ARPPU'] = (arppu['rev'] / arppu['id']).round(2)
9 arppu = arppu.rename(columns={'id': 'n_paying_users', 'rev': 'total_revenue'})
10
11 print("ARPU:")
12 print(arpu[['grp', 'ARPU']])
13
14 print("ARPPU:")
15 print(arppu[['grp', 'ARPPU']])
16

```

ARPU:

grp	ARPU
0 A	47.35
1 B	58.06

ARPPU:

grp	ARPPU
0 A	933.59
1 B	1257.88

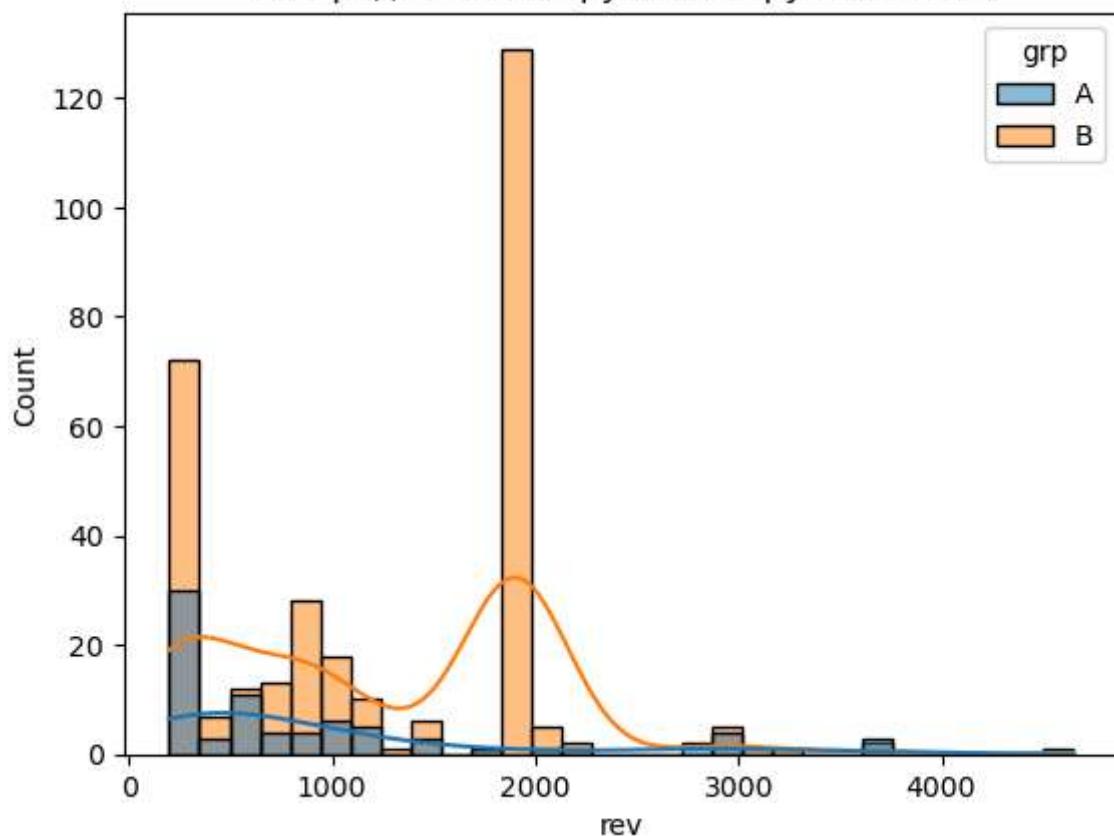
И ARPU и ARPPU с одного покупателя в тестовой группе по сравнению с контрольной выросли довольно значительно. Проверим равенство дисперсий для выбора стат теста.

In [130]:

```
1 import seaborn as sns
2 import matplotlib.pyplot as plt
3
4 sns.histplot(data=full_df, x="rev", hue="grp", kde=True, bins=30)
5 plt.title('Распределение выручки по группам А и В')
6 plt.show()
7
```

```
C:\Users\ainzha\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
    with pd.option_context('mode.use_inf_as_na', True):
C:\Users\ainzha\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1075: FutureWarning: When grouping with a length-1 list-like, you will need to pass a length-1 tuple to get_group in a future version of pandas. Pass `'(name,)` instead of `name` to silence this warning.
    data_subset = grouped_data.get_group(pd_key)
C:\Users\ainzha\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1075: FutureWarning: When grouping with a length-1 list-like, you will need to pass a length-1 tuple to get_group in a future version of pandas. Pass `'(name,)` instead of `name` to silence this warning.
    data_subset = grouped_data.get_group(pd_key)
C:\Users\ainzha\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1075: FutureWarning: When grouping with a length-1 list-like, you will need to pass a length-1 tuple to get_group in a future version of pandas. Pass `'(name,)` instead of `name` to silence this warning.
    data_subset = grouped_data.get_group(pd_key)
C:\Users\ainzha\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1075: FutureWarning: When grouping with a length-1 list-like, you will need to pass a length-1 tuple to get_group in a future version of pandas. Pass `'(name,)` instead of `name` to silence this warning.
    data_subset = grouped_data.get_group(pd_key)
```

Распределение выручки по группам А и В



Из графика видно, что распределение выручки не является нормальным, поэтому для проверки равенства дисперсий нужно воспользоваться тестом Левена.

Гипотезы:

- H0- Дисперсии выручки в двух группах равны (различия стат незначимы)
- H1 - Дисперсии выручки в двух группах различаются (различия стат значимы)

```
In [133]: 1 levene_stat, p_value = stats.levene(full_df.query('grp=="A"')['rev'].dropna()
2
3 print(f"levene's test: stat={levene_stat}, p-value={p_value}")
4
```

levene's test: stat=3.0756816999816055, p-value=0.08025675761347022

```
In [135]: 1 if p_value>0.05:
2     print(f" Принимаем H0. p-value - {p_value}.\n Дисперсии равны!")
3 else:
4     print(f"Отклоняем H0. p-value - {p_value} ")
```

Принимаем H0. p-value - 0.08025675761347022.
Дисперсии равны!

Видим, что разброс значений(дисперсии) выручки в обеих группах статистически не отличается. Следовательно, мы можем использовать t-тест с предположением о равенстве дисперсий для сравнения средних значений между группами

Гипотезы:

- H0 - средние выручки равны в A и B(стат незначимы)
- H1 - средние выручки неравны A и B (стат значимы)

```
In [140]: 1 t_stat, p_value = stats.ttest_ind(full_df.query('grp == "A"')['rev'], full_d
2 print(f"t-test: stat={t_stat}, p-value={p_value}")
```

t-test: stat=-3.1337795050930985, p-value=0.0018562025265765867

```
In [141]: 1 if p_value>0.05:
2     print(f" Принимаем H0. p-value - {p_value}.\n Средние равны!")
3 else:
4     print(f"Отклоняем H0. p-value - {p_value} ")
```

Отклоняем H0. p-value - 0.0018562025265765867

Вывод:

По результатам теста видим, что различия статистически значимы. Несмотря на то, что конверсия в оплату не изменилась, средняя выручка на платящего пользователя выросла значительно. Внесённые изменения, а именно новая механика оплаты, положительно повлияла на поведение пользователей, тк они приносят больше выручки. Значит, нужно выкатывать в прод изменения!

Задание 2. SQL

2.1 Очень усердные ученики.

2.1.1 Условие

Образовательные курсы состоят из различных уроков, каждый из которых состоит из нескольких маленьких заданий. Каждое такое маленькое задание называется "горошиной".

Назовём очень усердным учеником того пользователя, который хотя бы раз за текущий месяц правильно решил 20 горошин.

2.1.2 Задача

Дана таблица peas:

Название атрибута	Тип атрибута	Смыслоное значение
st_id	int	ID ученика
timestamp	timestamp	Время решения карточки
correct	bool	Правильно ли решена горошина?
subject	text	Дисциплина, в которой находится горошина

Необходимо написать оптимальный запрос, который даст информацию о количестве очень усердных студентов.

Сначала смотрим на месяцы в нашей таблице. С помощью запроса, видим, что у нас только один единственный месяц в таблице(текущий месяц - 10).

```
select count(DISTINCT EXTRACT(MONTH from timest))
from peas
```

Напишем другой запрос, чтобы увидеть, какое количество дней в текущем месяце попали в нашу базу данных.

```
select count(distinct DATE(timest))
from peas
```

Он нам показал, что у нас всего 2 дня с данными. Следовательно, данных за текущий месяц у нас нет.

С помощью CTE получим список студентов за текущий месяц с правильно решенными заданиями, сгруппировав их по id. Затем посчитаем количество уникальных id (студентов), у которых количество правильно решенных задач равно или больше 20.

```
WITH table1 as(
SELECT count(correct) as count_true, st_id
FROM peas
where correct is true
group by st_id
)

SELECT count(distinct st_id) as n_successful_Students
From table1
where count_true >= 20
```

В итоге получаем 136 очень усердных студентов

n_successful_students
136

2.2 Оптимизация воронки

2.2.1 Условие

Образовательная платформа предлагает пройти студентам курсы по модели trial: студент может решить бесплатно лишь 30 горошин в день. Для неограниченного количества заданий в определенной дисциплине студенту необходимо приобрести полный доступ. Команда провела эксперимент, где был протестирован новый экран оплаты.

2.2.2 Задача

Дана таблицы: **peas** (см. выше), **studs**:

Название атрибута	Тип атрибута	Смыслоное значение
st_id	int	ID ученика
test_grp	text	Метка ученика в данном эксперименте

и **final_project_check**:

Название атрибута	Тип атрибута	Смыслоное значение
st_id	int	ID ученика
sale_time	timestamp	Время покупки
money	int	Цена, по которой приобрели данный курс
subject	text	Дисциплина

Необходимо в одном запросе выгрузить следующую информацию о группах пользователей:

- ARPU
- ARPAU
- CR в покупку
- CR активного пользователя в покупку
- CR пользователя из активности по математике (`subject = 'math'`) в покупку курса по математике

ARPU считается относительно всех пользователей, попавших в группы.

Активным считается пользователь, за все время решивший больше 10 задач правильно в любых дисциплинах. Активным по математике считается пользователь, за все время решивший 2 или больше задач правильно по математике. Все данные хранятся в PostgreSQL

1. Для работы с таблицами перейдите в Redash <https://redash.lab.karpov.courses> (<https://redash.lab.karpov.courses>)
2. Выберите схему StartDA
3. Работайте с таблицами `peas`, `studs` и `final_project_check`, другие таблицы вам не нужны.

```
WITH
--все пользователи и группы
users AS (
    SELECT st_id, test_grp
    FROM studs
),
-- покупки
purchases AS (
    SELECT st_id, sum(money) AS revenue
    FROM final_project_check
    GROUP BY st_id
),
-- пользователи купившие курс по математике
math_buyers AS (
    SELECT st_id
    FROM final_project_check
    WHERE subject = 'Math'
),
-- пользователи, правильно решившие задачи
correct_peas AS (
    SELECT st_id, subject
    FROM peas
    WHERE correct = TRUE
),
--активные пользователи(более 10 задач)
active_users AS (
    SELECT st_id
    FROM correct_peas
    GROUP BY st_id
    HAVING COUNT(*) > 10
),
--активные по математике(2 и более задач по математике)
active_math_users AS (
    SELECT DISTINCT st_id
    FROM correct_peas
    WHERE subject = 'Math'
    GROUP BY st_id
    HAVING COUNT(*) >= 2
),
final_table AS (
    SELECT
        u.test_grp,
        u.st_id,
        COALESCE(p.revenue, 0) AS revenue,
        CASE WHEN p.st_id IS NOT NULL THEN 1 ELSE 0 END AS is_buyer,
```

```

CASE WHEN a.st_id IS NOT NULL THEN 1 ELSE 0 END AS is_active,
CASE WHEN am.st_id IS NOT NULL THEN 1 ELSE 0 END AS is_active_ma
th,
CASE WHEN mb.st_id IS NOT NULL THEN 1 ELSE 0 END AS bought_math_
course
FROM users u
LEFT JOIN purchases p ON p.st_id = u.st_id
LEFT JOIN active_users a ON a.st_id = u.st_id
LEFT JOIN active_math_users am ON am.st_id = u.st_id
LEFT JOIN math_buyers mb ON mb.st_id = u.st_id
)

--итоговый расчет
SELECT
    test_grp,
    sum(revenue) * 1.0 / count(st_id) as arpu,
    sum(revenue) * 1.0 / sum(is_active) as arpau,
    sum(is_buyer) * 1.0 / count(st_id) as cr,
    sum(is_buyer)* 1.0 / sum(is_active) as cr_active,
    sum(bought_math_course) * 1.0 / sum(is_active_math) AS cr_active_mat
h
FROM final_table
GROUP BY test_grp

```

Вывод

test_grp	arpu	arpau	cr	cr_active	cr_active_math
control	4,540.98	10,905.51	0.05	0.12	0.16
pilot	11,508.47	35,364.58	0.11	0.33	0.38

Задание 3. Python

3.1 Задача

Реализуйте функцию, которая будет автоматически подгружать информацию из дополнительного файла groups_add.csv (заголовки могут отличаться) и на основании дополнительных параметров пересчитывать метрики. Реализуйте функцию, которая будет строить графики по получаемым метрикам.

Разобьем задание на подзадачи:

1. Первая функция:

- Принимет файл groups_add.csv(заголовки могут быть разные!!).
- Подгружает и объединяет с основными данными(groups).
- Пересчитывает метрики(CR,ARPU,ARPPU)

2. Вторая функция:

In [44]:

```

1 import requests
2 import pandas as pd
3 from io import StringIO
4 import seaborn as sns
5 import matplotlib.pyplot as plt
6
7 def auto_analysis(groups_url, groups_add_url, active_studs_url, checks_url):
8     # функция для скачивания CSV
9     def download_csv_from_disk(public_url):
10         api_url = 'https://cloud-api.yandex.net/v1/disk/public/resources/dow
11         params = {'public_key': public_url}
12         resp = requests.get(api_url, params=params)
13         resp.raise_for_status()
14         download_url = resp.json()['href']
15         file_resp = requests.get(download_url)
16         file_resp.raise_for_status()
17         text = file_resp.text
18         sep = ',' if text.count(',') >= text.count(';') else ';'
19         return pd.read_csv(StringIO(text), sep=sep)
20
21     try:
22         # загружаем данные
23         groups = download_csv_from_disk(groups_url)
24         groups_add = download_csv_from_disk(groups_add_url)
25         active_studs = download_csv_from_disk(active_studs_url)
26         checks = download_csv_from_disk(checks_url)
27
28         # объединяем группы
29         groups_full = pd.concat([groups, groups_add], ignore_index=True)
30
31         # переименовываем столбцы у groups_add
32         groups_full = groups_full.rename(columns={groups_add.columns[0]: 'st
33
34         # оставляем только активных пользователей
35         active_groups = groups_full.query('student_id in @active_studs.stude
36
37         # объединяем с оплатами
38         active_ab = active_groups.merge(checks, how='left', on='student_id')
39
40         # создаём колонку, кто платил
41         active_ab['paying'] = active_ab['rev'].fillna(0).apply(lambda x: 1 if
42
43         # добавляем проверку на пустые данные
44         print("Проверка на количество строк после объединения: ", active_ab.
45         print(active_ab.head()) # для отладки
46
47         # считаем метрики
48         metrics = active_ab.groupby('grp').agg(
49             n_active_users=('student_id', 'nunique'),
50             n_paying_users=('paying', 'sum'),
51             total_revenue=('rev', 'sum')
52         ).reset_index()
53
54         # конверсия и метрики
55         metrics['CR'] = metrics['n_paying_users'] / metrics['n_active_users']
56         metrics['ARPU'] = metrics['total_revenue'] / metrics['n_active_users']
57         metrics['ARPPU'] = metrics['total_revenue'] / metrics['n_paying_user

```

```
58     print("\n Данные успешно загружены и метрики пересчитаны!")
59
60
61     return metrics
62
63 except Exception as e:
64     print(f"Ошибка: {e}")
65     return None
66
67
```

In [45]:

```
1 # Функция для построения графиков
2 def plot_metrics(metrics_df):
3     fig, axes = plt.subplots(1, 3, figsize=(18, 5))
4
5     sns.barplot(data=metrics_df, x='grp', y='CR', palette='Blues', ax=axes[0])
6     axes[0].set_title('CR')
7     axes[0].grid(axis='y')
8
9     sns.barplot(data=metrics_df, x='grp', y='ARPU', palette='Greens', ax=axes[1])
10    axes[1].set_title('ARPU')
11    axes[1].grid(axis='y')
12
13    sns.barplot(data=metrics_df, x='grp', y='ARPPU', palette='Oranges', ax=axes[2])
14    axes[2].set_title('ARPPU')
15    axes[2].grid(axis='y')
16
17    plt.tight_layout()
18    plt.show()
19
```

In [46]:

```

1 # загружаем и считаем метрики
2 metrics = auto_analysis(
3     'https://disk.yandex.ru/d/UhyYx41rTt3c1Q', # groups
4     'https://disk.yandex.ru/d/5Kxrz02m3IBUwQ', # groups_add
5     'https://disk.yandex.ru/d/Tbs44Bm6H_FwFQ', # active_studs
6     'https://disk.yandex.ru/d/pH1q-VqcxXjsVA' # checks
7 )
8
9 metrics

```

Проверка на количество строк после объединения: (8341, 4)

	student_id	grp	rev	paying
0	1627	A	990.0	1
1	2085	B	NaN	0
2	2215	B	NaN	0
3	3391	A	NaN	0
4	3401	B	NaN	0

Данные успешно загружены и метрики пересчитаны!

Out[46]:

	grp	n_active_users	n_paying_users	total_revenue	CR	ARPU	ARPPU
0	A	1538		72820.0000	0.050715	47.347204	933.589744
1	B	6803		394974.0035	0.046156	58.058798	1257.878992

In [47]:

```
1 plot_metrics(metrics)
```

