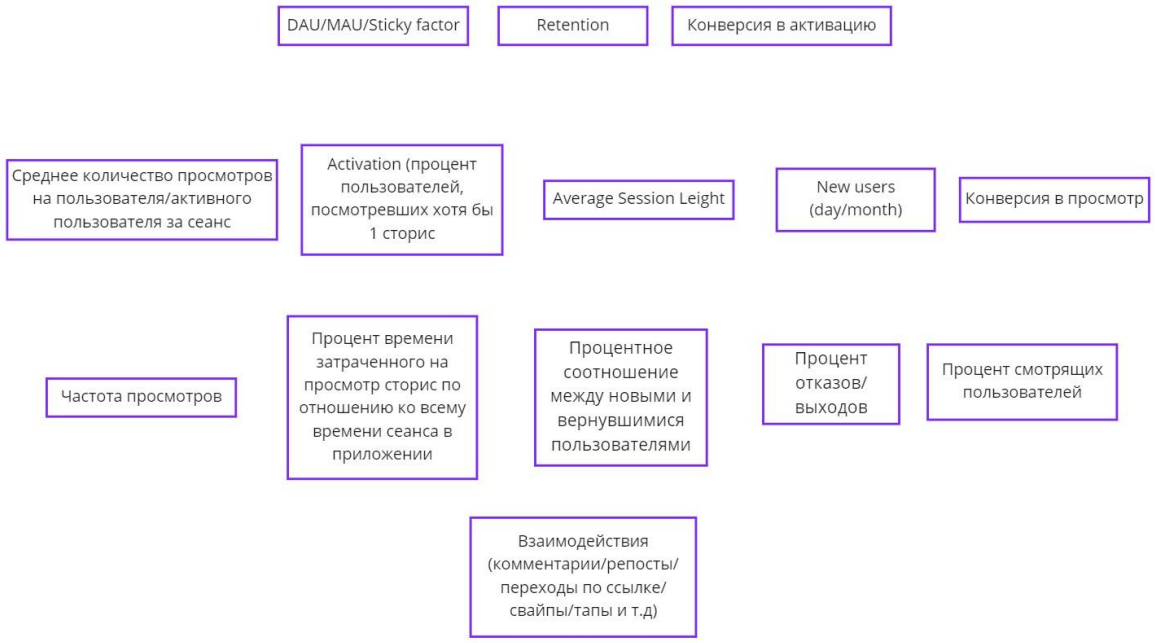
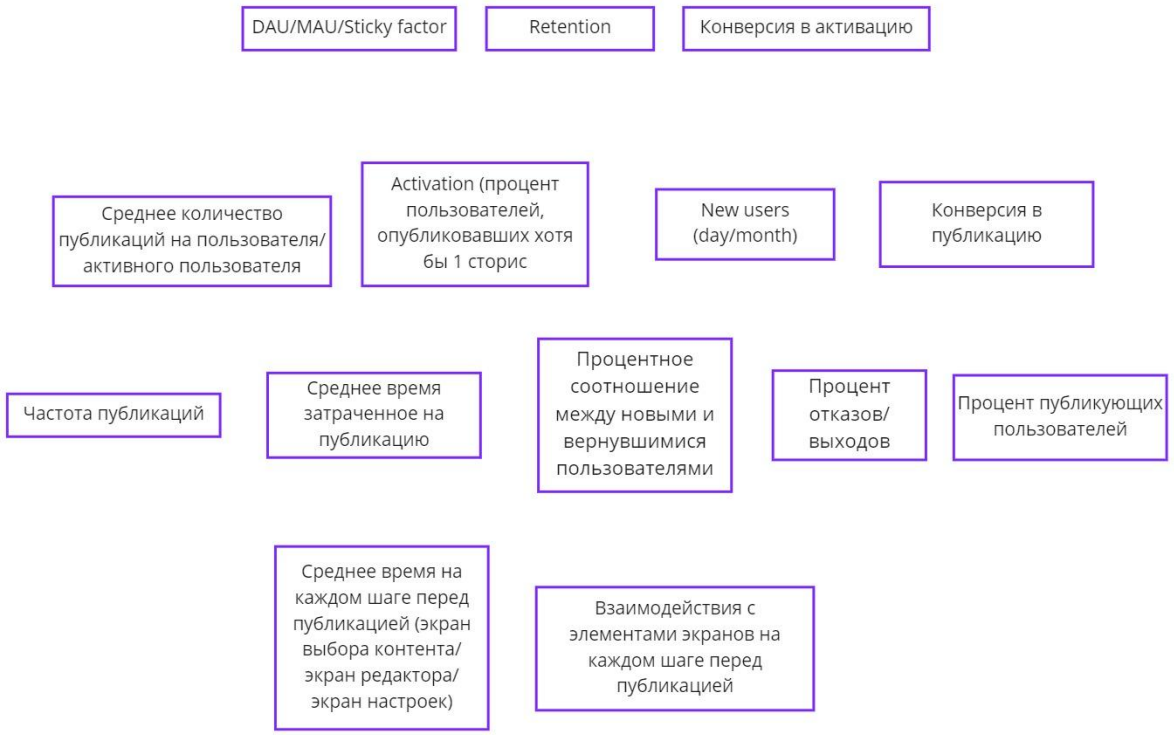
Задание 1.

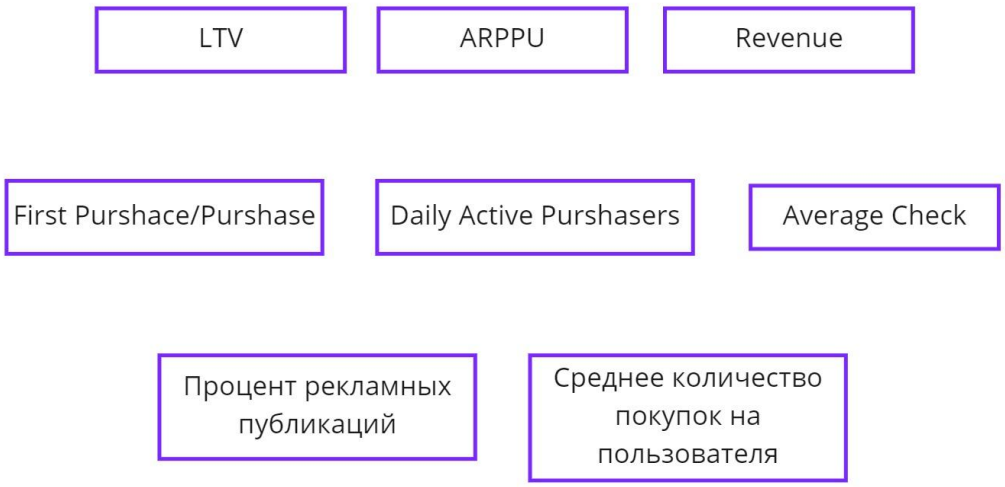
Метрики по просмотрам



Метрики по публикациям



Метрики по использованию продвижения в сторис



Также необходимо подобрать фильтры к каждой метрике:

- новые/старые пользователи
- активные/неактивные пользователи (по регулярности входов)
- платящие/неплатящие (по использованию продвижения)
- регулярность/количество платежей (для тех кто использует продвижение)
- платформа (ios, android, desktop)
- гео (регион, страна, язык и т.д)
- соц-дем (пол, возраст, соц. статус и т.д.)
- характеристика контента (какой контент публикуют, например - image/video)

Задание 2.

Ссылка на jupyter notebook

Задание 3.

Обычная SQL задача 1

1.

```
SELECT
	Employees.name,
	Departments.name
FROM Employees
	LEFT JOIN Departments ON Employees.dep_id = Departments.id
```

2.

```
SELECT
	Departments.name,
	MAX(Salary) AS max_salary
FROM Departments
	JOIN Employees ON Departments.id = Employees.dep_id
GROUP BY Departments.name
ORDER BY max_salary DESC
```

Обычная SQL задача 2

1.

```
SELECT SUM(price * items) AS revenue
FROM Purchases
WHERE user_gender LIKE 'f%'
```

2.

```
SELECT DISTINCT
	(
	SELECT SUM(price * items)
	FROM Purchases
	WHERE user_gender LIKE 'f%'
	) AS revenue_female,
	(
	SELECT SUM(price * items)
	FROM Purchases
	WHERE user_gender LIKE 'm%'
	) AS revenue_male
FROM Purchases
```

3.

```
SELECT COUNT(DISTINCT user_id)
FROM Purchases
WHERE
	user_gender LIKE 'm%'
	AND
	user_id IN (
		SELECT user_id
		FROM Purchases
		GROUP BY user_id
		HAVING SUM(items) > 3
		)
```

Не самая обычная SQL задача 3

1.

```
WITH transaction_numbers AS
	(
	SELECT
		user_id,
		item,
		ROW_NUMBER () OVER (PARTITION BY user_id ORDER BY
transaction_ts) AS transaction_number
	FROM transactions
	)
SELECT
	user_id,
	item
FROM transaction_numbers
WHERE transaction_number = 1
```

2.

```
WITH first_transactions AS
	(
	SELECT
		transaction_ts,
		user_id,
	MIN(transaction_ts) OVER (PARTITION BY user_id ORDER BY transaction_ts) AS
first_transaction
	FROM transactions
	)
SELECT ROUND(AVG(quantity), 1) AS avg_quantity
FROM (
```

```
SELECT
        user_id,
        COUNT(transaction_ts) AS quantity
FROM first_transactions
WHERE transaction_ts <= (first_transaction + INTERVAL '3 DAY')
GROUP BY user_id
) AS t_1
```