

Департамент образования и науки города Москвы
Государственное автономное образовательное учреждение
высшего образования города Москвы
«Московский городской педагогический университет»
Институт цифрового образования
Департамент информатики, управления и технологий

ОТЧЕТ

по дисциплине «Проектный практикум по разработке ETL-решений»
Лабораторная работа 6.1: «Разработка полного ETL-процесса. Оркестровка
конвейера данных»

Направление подготовки – 38.03.05 «Бизнес-информатика».
профиль подготовки – «Аналитика данных и эффективное управление»

Выполнила:

St92

Руководитель:

Москва
2025 год

Цель работы: научиться извлекать данные из архива для последующей загрузки в БД и решения бизнес-задачи

Задачи

1. Скачать архив Wikimedia по интересующим датам.
2. Извлечь данные из архива Wikimedia.
3. Загрузить извлеченные данные в таблицу postgres.
4. С помощью SQL-запроса вывести необходимые для решения бизнес-задачи показатели.
5. Визуализировать показатели для решения бизнес-задачи.

Ход работы:

- 6.1.1. Развернем VM ubuntu_mgpu.ova в VirtualBox.

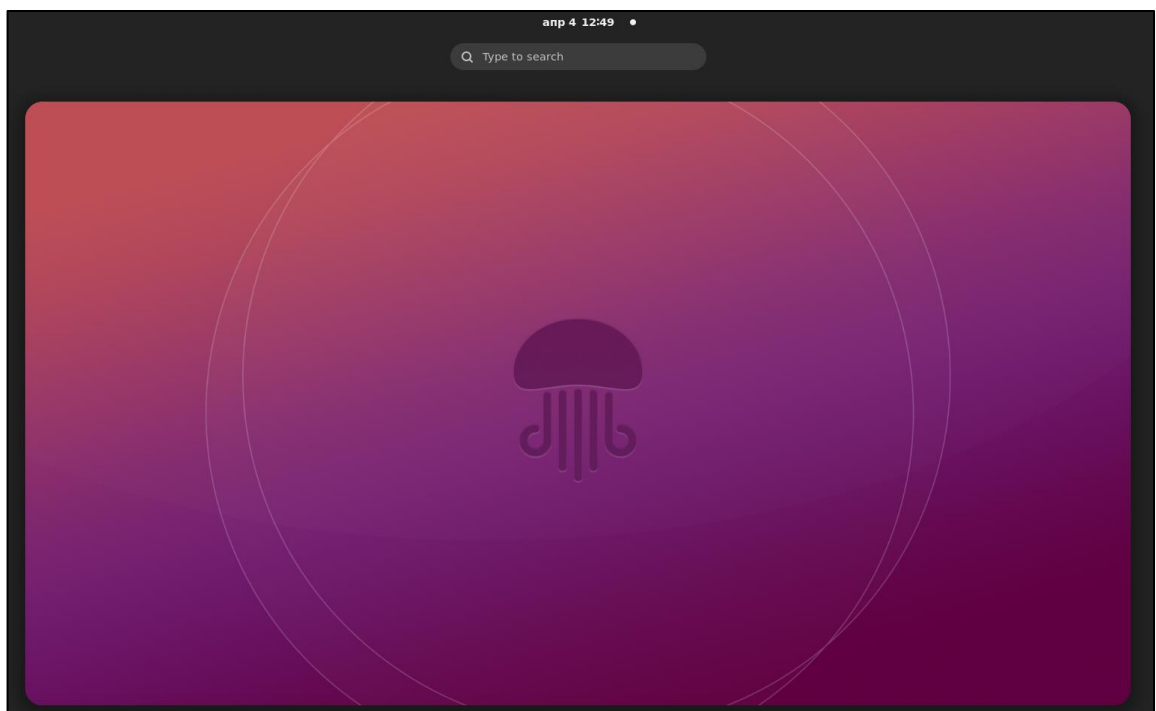


Рисунок 1 – Развернутая ВМ в VirtualBox

6.1.2. Клонировем на ПК задание Бизнес-кейс «StockSense» в домашний каталог ВМ.

Клонировем репозиторий с проектом:

```
dev@dev-vm:~$ ls
Desktop  google-chrome-stable_current_amd64.deb  Public  thinclient_drives
Documents Music                                snap    Videos
Downloads Pictures                               Templates
dev@dev-vm:~$ cd Downloads
dev@dev-vm:~/Downloads$ git clone https://github.com/BosenkoTM/workshop-on-ETL.git
Cloning into 'workshop-on-ETL'...
remote: Enumerating objects: 637, done.
remote: Counting objects: 100% (30/30), done.
remote: Compressing objects: 100% (29/29), done.
remote: Total 637 (delta 18), reused 1 (delta 1), pack-reused 607 (from 1)
Receiving objects: 100% (637/637), 5.83 MiB | 8.05 MiB/s, done.
Resolving deltas: 100% (316/316), done.
```

Рисунок 2 – Выполнение команды для клонирования репозитория

```
dev@dev-vm:~/Downloads$ ls
dba  de  lab_etl  progs  workshop-on-ETL
dev@dev-vm:~/Downloads$
```

Рисунок 3 – Просмотр содержимого каталога

```
dev@dev-vm:~/Downloads$ wget https://dumps.wikimedia.org/other/pageviews/2025/20
25-04/pageviews-20250404-060000.gz
--2025-04-04 12:59:02-- https://dumps.wikimedia.org/other/pageviews/2025/2025-0
4/pageviews-20250404-060000.gz
Resolving dumps.wikimedia.org (dumps.wikimedia.org)... 208.80.154.71, 2620:0:861
:3:208:80:154:71
Connecting to dumps.wikimedia.org (dumps.wikimedia.org)|208.80.154.71|:443... co
nnected.
HTTP request sent, awaiting response... 200 OK
Length: 52035377 (50M) [application/octet-stream]
Saving to: 'pageviews-20250404-060000.gz'
pageviews-20250404-060 23%[====>] 1 11 63M 4 14MB/s
```

Рисунок 4 – Выполнение команды для скачивания архива с wikimedia

```
dev@dev-vm:~/Downloads$ gunzip pageviews-20250404-060000.gz
dev@dev-vm:~/Downloads$ ls
dba  de  lab_etl  pageviews-20250404-060000  progs  workshop-on-ETL
dev@dev-vm:~/Downloads$
```

Рисунок 5 – Выполнение команды для распаковки архива

```
dev@dev-vm:~/Downloads$ awk -F ' ' '{print $1}' pageviews-20250404-060000 | sort | uniq -c |
sort -nr | head
1179498 en.m
939848 en
230264 ru.m
222239 ja.m
220389 ja
164292 de.m
164249 es.m
142722 zh.m
138193 ru
132422 zh
```

Рисунок 6

6.1.3. Запустим контейнер с Бизнес-кейсом «StockSense», изучим основные элементы DAG в Apache Airflow.

Перейдем в директорию с бизнес-кейсом " StockSense ", остановим контейнеры и удалим их:

```
dev@dev-vm:~/Downloads/workshop-on-ETL/business_case_stocksense_25$ sudo docker stop $(sudo docker ps -a -q)
[sudo] password for dev:
676b1a644fd9
dfd339d83cd2
a5bb7206afbd
ec4919b750da
dev@dev-vm:~/Downloads/workshop-on-ETL/business_case_stocksense_25$ sudo docker rm -f $(sudo docker ps -a -q)
[sudo] password for dev:
676b1a644fd9
dfd339d83cd2
a5bb7206afbd
ec4919b750da
```

Рисунок 7 – Выполнение команд для остановки и удаления контейнеров

Удалим образы

```
dev@dev-vm:~/Downloads/workshop-on-ETL/business_case_stocksense_25$ sudo docker rmi -f $(sudo docker images -q)
Untagged: postgres:16
Untagged: postgres@sha256:361c84f2bbe0f91135990841eb4eb4d2fe491f6e950686d9c746415dd1ffc65e
Deleted: sha256:a299406cefae24c297ae4a13d269da1cf35eda3c70ceed45634212f54cf2edc9
Deleted: sha256:ab262fea3120a4ba9fce71f18846198f0e0efcd69119071240ec67c71346775
Deleted: sha256:3af22cf14e62e3588be1fab998d651e87165a007b180abb6a586a2d93e255575
Deleted: sha256:44af15f5e9ff82558ca8e8afadeb6eb66862f826b47a57dd88764c414d6ac3ea
Deleted: sha256:77e2647bca93a2f78f6af57bb860c09ceacc2d5a32facbe835fe16e234498191
Deleted: sha256:f2751330f09031fd00a22e40a4fbbfad9ec62e944d0ed89b49b278aa0a1aeec
Deleted: sha256:6e313cc19a0fa3af6311a35d658e6056ad9f6f355a23628da491d68a48ee546a
Deleted: sha256:f6a062c257a1b7bc3ca7e9edd329006f52a39ab7c56985f4c38faa06a82b8c83
Deleted: sha256:15f9836b82fde94b66a82911236452edf9e9d782a6d1911b44990069e5e7c0b4
Deleted: sha256:8e288c329c3cd5ae9f57b9d5ce2d93490841308d052d64c1f2eb6cea3648fc01
Deleted: sha256:b3ea0fc720554f01d6e3b65c8f2f6f8cd6dd489c4e8366e9cbff2f860bc93079
Deleted: sha256:b63c158606d2234fe44c9c3cf56892ea01bd6757d897aa1bed83e71b1a62663b
Deleted: sha256:008ff5a1f2dff5c94b3baa8c1563e94d4f7e0b471e9434838f94cc49cde410e6
Deleted: sha256:1cefcdaf62a5a0db3d87628e95266c5618cf7d24f1d6235cf758693ec24141fa
Deleted: sha256:7914c8f600f532b7adbd0b003888e3aa921687d62dbe2f1f829d0ab6234a158a
Untagged: mongo:7.0.17-rc1-jammy
```

Рисунок 8 – Выполнение команды для удаления образов

Билдим новый Docker образ:

```
dev@dev-vm:~/Downloads/workshop-on-ETL/business_case_stocksense_25$ sudo docker build -t custom-airflow:slim-2.8.1-python3.11 .
2025/04/04 13:23:39 in: [string{}]
2025/04/04 13:23:39 Parsed entitlements: []
[+] Building 1.5s (1/2)
=> [internal] load build definition from Dockerfile
=> transferring dockerfile: 615B
=> [internal] load metadata for docker.io/apache/airflow:slim-2.8.1-python3.11
```

Рисунок 9 – Выполнение команды для сборки Docker образа

Запустим среду

```
dev@dev-vm:~/Downloads/workshop-on-ETL/business_case_stocksense_25$ sudo docker compose up --build
[+] Running 10/13
: postgres Pulling
: wiki results [=====] 97.23MB / 106.1MB Pulling
  ✓ 1f3e46996e29 Pull complete
  ✓ 47e20ba03731 Pull complete
  ✓ 101b82465a4f Pull complete
  ✓ 319529a7ccb0 Pull complete
  ✓ c2f9392cfd4c Pull complete
  : 4e04446ce95d Downloading [=====] 92.45MB/101.3MB
  ✓ 47bfe778b869 Download complete
  ✓ b1d66b287aa8 Download complete
  ✓ 7865e52a4759 Download complete
  ✓ 7d75f14147c2 Download complete
  ✓ 11052a5424e7 Download complete
```

Рисунок 10 – Запуск контейнера

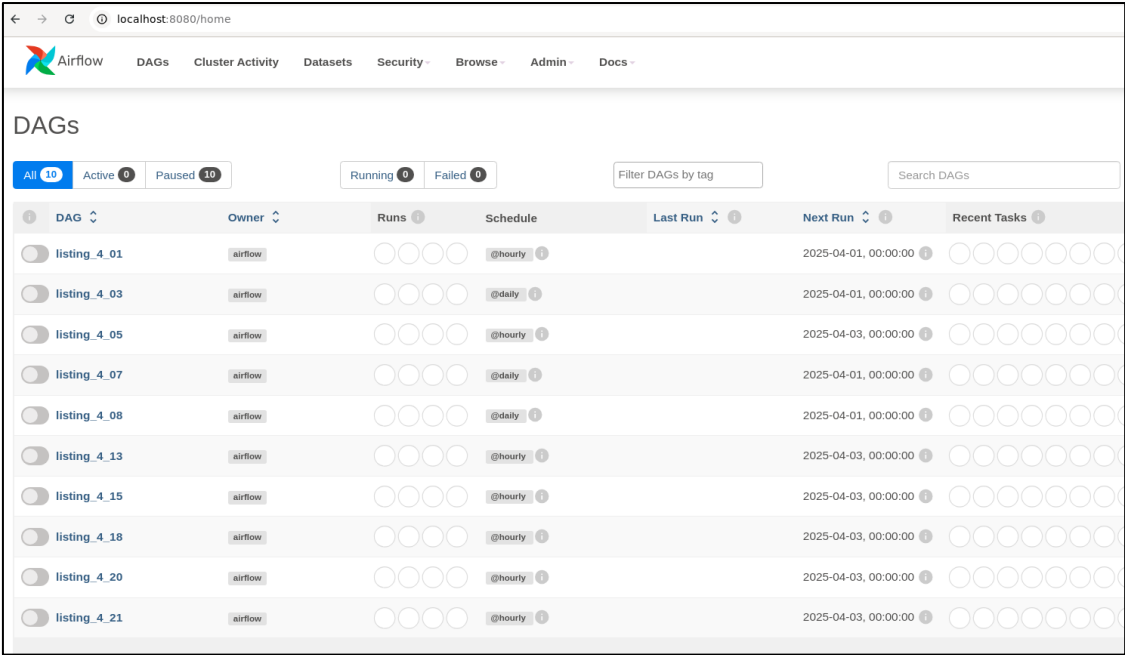


Рисунок 11 – Успешное соединение

Создадим DAG согласно алгоритму, который предоставил преподаватель.

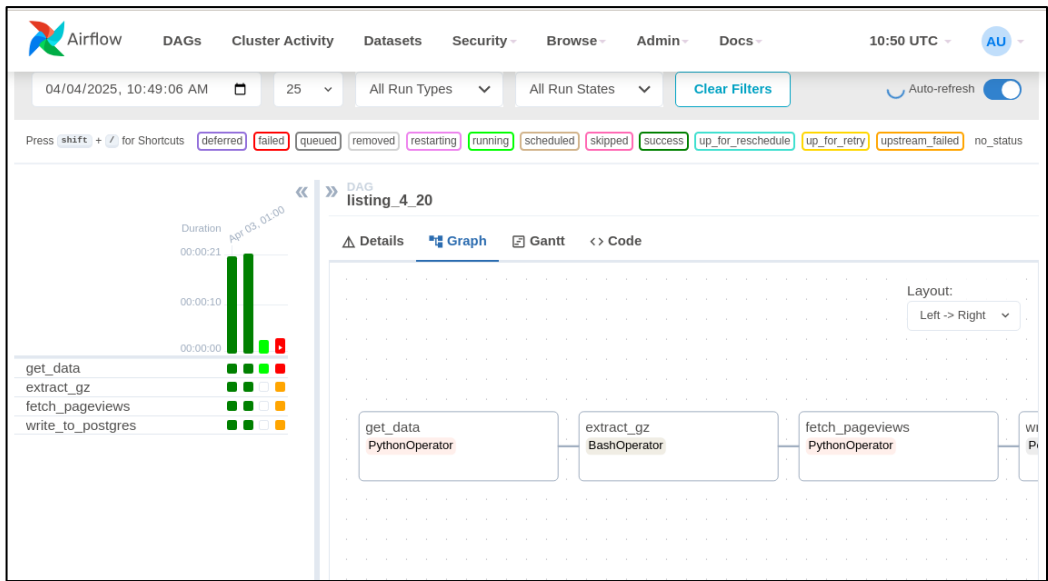


Рисунок 12 – Граф DAG

Изучим логи, выполненного DAG.

Dag Audit Log					
This view displays selected events and operations that have been taken on this dag. The included and excluded events are set in the Airflow configuration, which by default remove view only actions. For a full list of events regarding this DAG, click here .					
Time	Task ID	Event	Logical Date	Owner	Details
2025-04-04, 18:22:04	get_data	confirm	None	admin	[{"dag_id": "listing_4_20", "dag_run_id": "scheduled__2025-04-03T06:00:00+00:00", "past": "false", "future": "false", "upstream": "false", "downstream": "false", "state": "success", "task_id": "get_data"}]
2025-04-04, 18:22:04	get_data	clear	2025-04-03 06:00:00+00:00	admin	[{"dag_id": "listing_4_20", "dag_run_id": "scheduled__2025-04-03T06:00:00+00:00", "confirmed": "false", "execution_date": "2025-04-03T06:00:00+00:00", "past": "false", "future": "false", "upstream": "false", "downstream": "true", "recursive": "true", "only_failed": "false", "task_id": "get_data"}]
2025-04-04, 18:21:55	get_data	cli_task_run	None	airflow	{"host_name": "53be4afa4ebb", "full_command": "[\"/home/airflow/.local/bin/airflow\", \"scheduler\"]"}
2025-04-04, 18:21:55	get_data	running	2025-04-03 07:00:00+00:00	airflow	None
2025-04-04, 18:21:55	get_data	cli_task_run	None	airflow	{"host_name": "53be4afa4ebb", "full_command": "[\"/home/airflow/.local/bin/airflow\", \"scheduler\"]"}
2025-04-04, 18:21:55	get_data	cli_task_run	None	airflow	{"host_name": "53be4afa4ebb", "full_command": "[\"/home/airflow/.local/bin/airflow\", \"scheduler\"]"}

Рисунок 13 – Логи DAG

Убедимся, что контейнер PostgreSQL работает
Создадим новое подключение в DBeaver

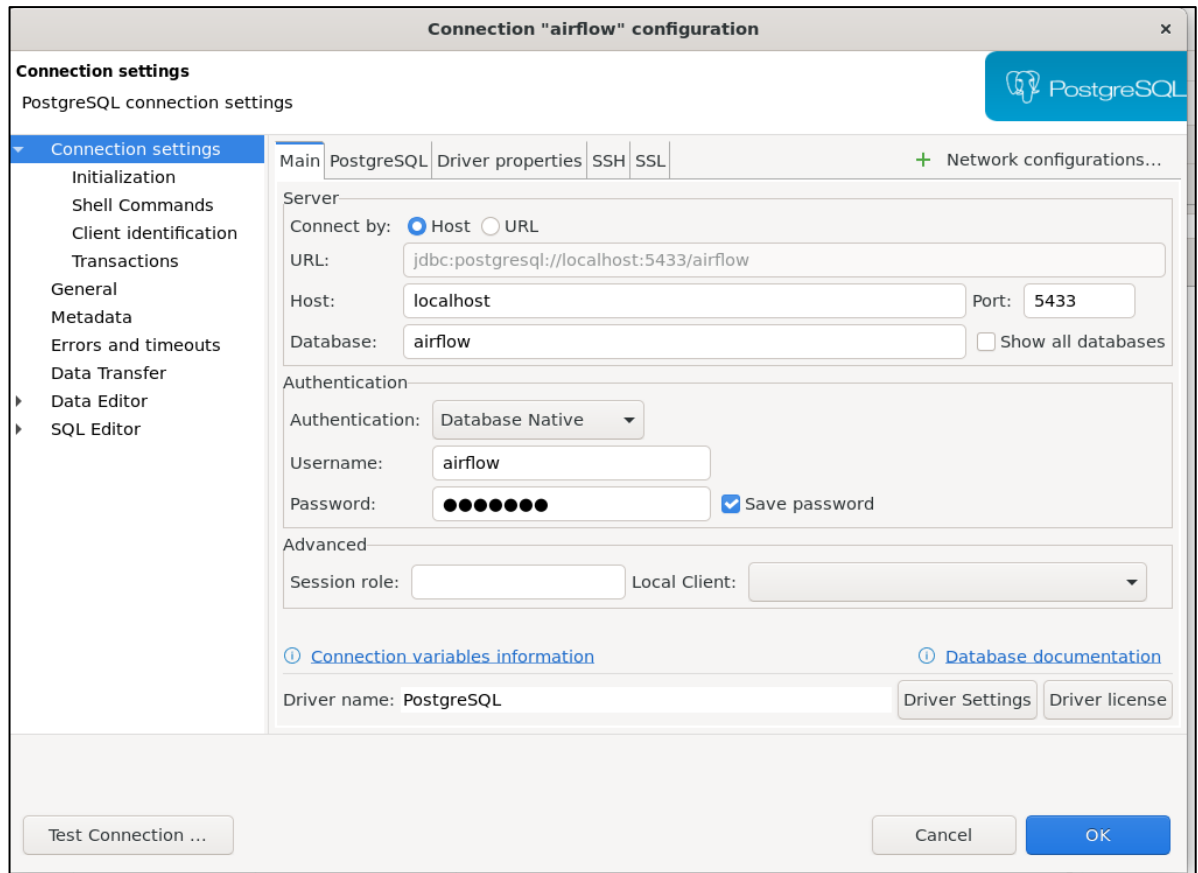


Рисунок 14 – Параметры подключения

Просмотрим данные в таблице:

Database Navigator x Projects

Enter a part of object name here

airflow localhost:5433

Databases

airflow

Schemas

public

Tables

pageview_counts

Columns

Constraints

Foreign Keys

Indexes

Dependencies

References

Partitions

Triggers

Rules

Policies

pageview_counts x

Properties Data Diagram

pageview_counts Enter a SQL expression to filter results (use Ctrl+Space)

	A-Z pagename	123 pageviewcount	datetime
1	Facebook	215	2025-04-03 00:00:00.000
2	Amazon	6	2025-04-03 00:00:00.000
3	Apple	38	2025-04-03 00:00:00.000
4	Microsoft	137	2025-04-03 00:00:00.000
5	Google	330	2025-04-03 00:00:00.000
6	Facebook	206	2025-04-03 01:00:00.000
7	Amazon	6	2025-04-03 01:00:00.000
8	Apple	48	2025-04-03 01:00:00.000
9	Microsoft	131	2025-04-03 01:00:00.000
10	Google	348	2025-04-03 01:00:00.000
11	Facebook	230	2025-04-03 02:00:00.000
12	Amazon	7	2025-04-03 02:00:00.000
13	Apple	42	2025-04-03 02:00:00.000
14	Microsoft	127	2025-04-03 02:00:00.000
15	Google	351	2025-04-03 02:00:00.000

Рисунок 15 – Заполненная таблица

6.1.5. Спроектировать верхнеуровневую архитектуру аналитического решения Бизнес-кейса «StockSense» в draw.io. Необходимо использовать:

- Source Layer - слой источников данных.
- Storage Layer - слой хранения данных.
- Business Layer - слой для доступа к данным пользователей.

Спроектируем верхнеуровневую архитектуру аналитического решения задания Бизнес-кейса «Rocket»:

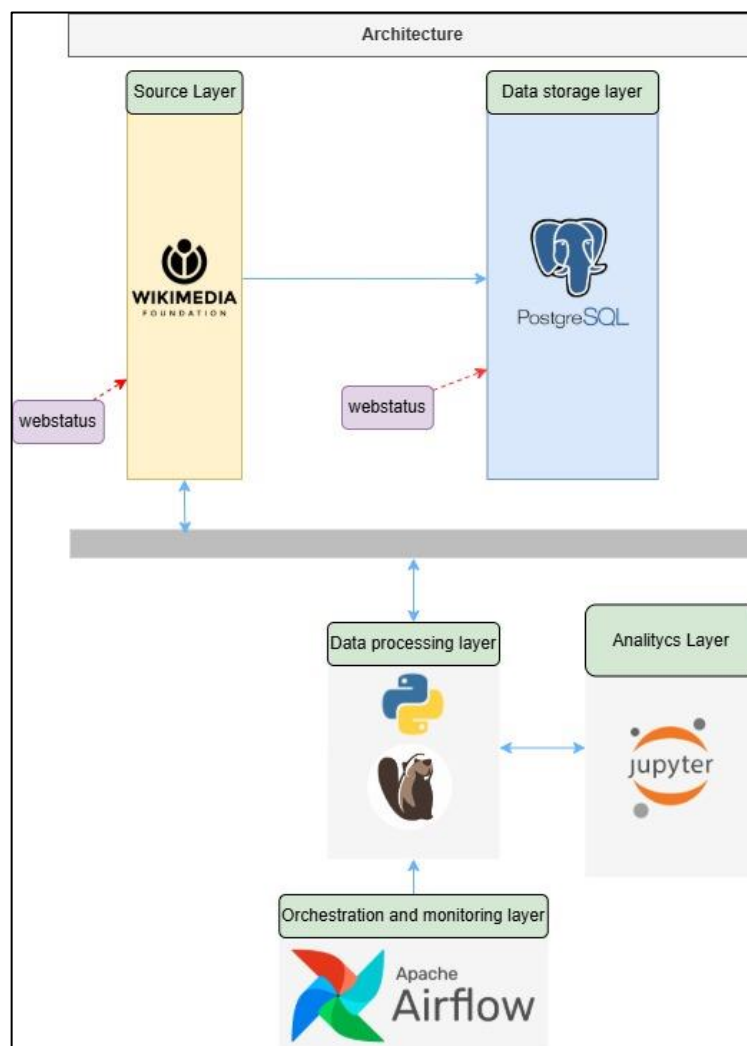


Рисунок 16 – Верхнеуровневая архитектура бизнес-кейса

6.1.6. Спроектировать архитектуру DAG Бизнес-кейса «StockSense» в draw.io. Необходимо использовать:

- Source Layer - слой источников данных.
- Storage Layer - слой хранения данных.
- Business Layer - слой для доступа к данным пользователей.

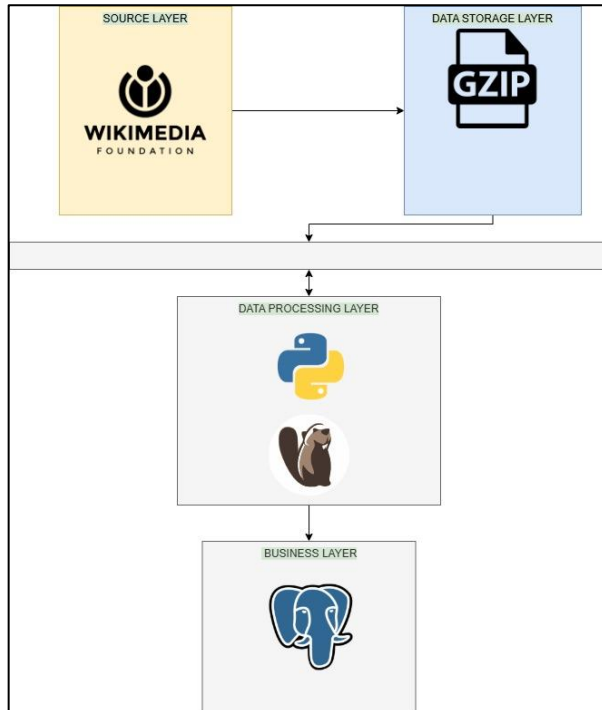


Рисунок 17 – Верхнеуровневая архитектура DAG

6.1.7. Построить диаграмму Ганта работы DAG в Apache Airflow.

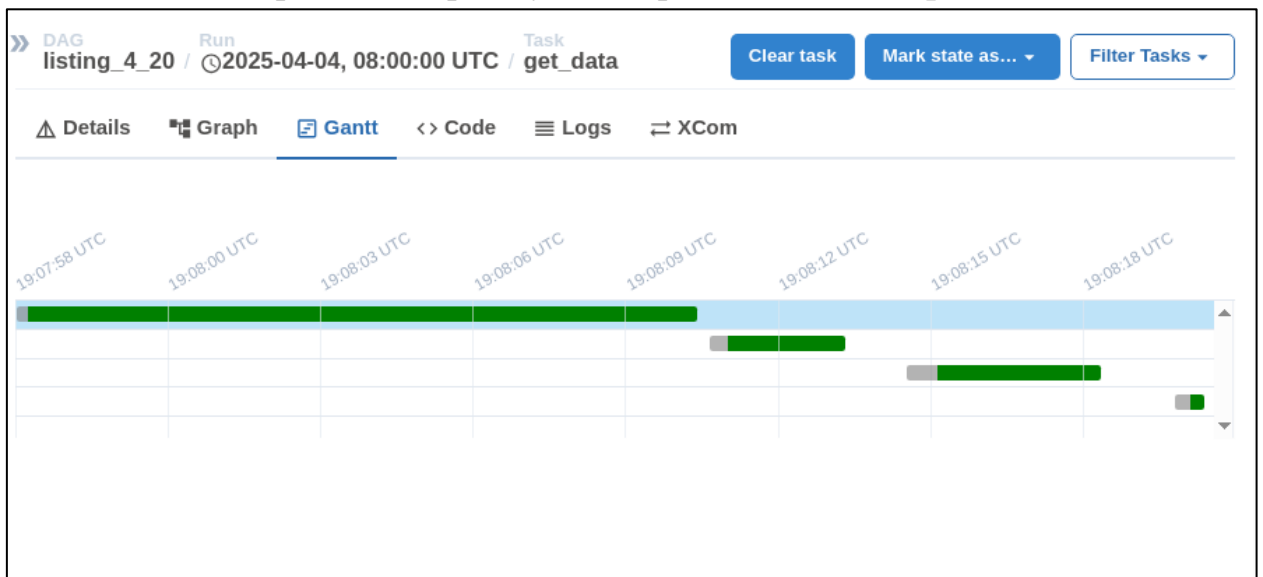


Рисунок 18 – Диаграмма ганта в Apache Airflow

ERD-схема базы данных Postgre SQL;

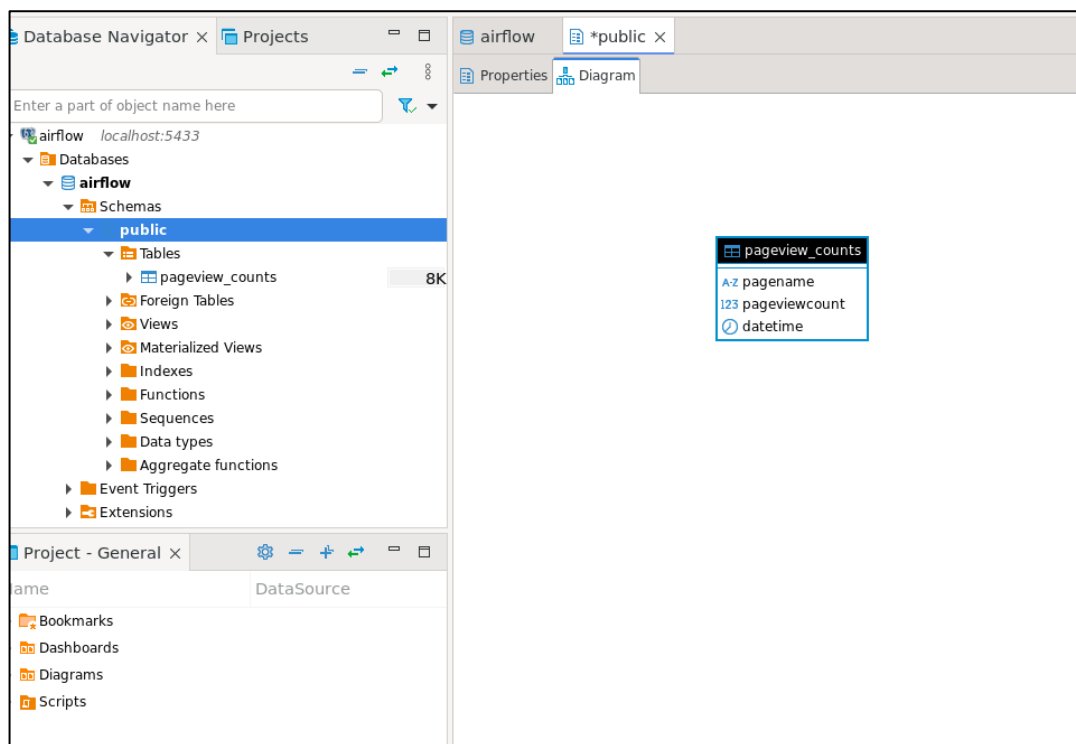


Рисунок 19 – ERD-схема schema

SQL-запросы, позволяющие проверить наличие выгруженных агрегированных данных бизнес-задачи.

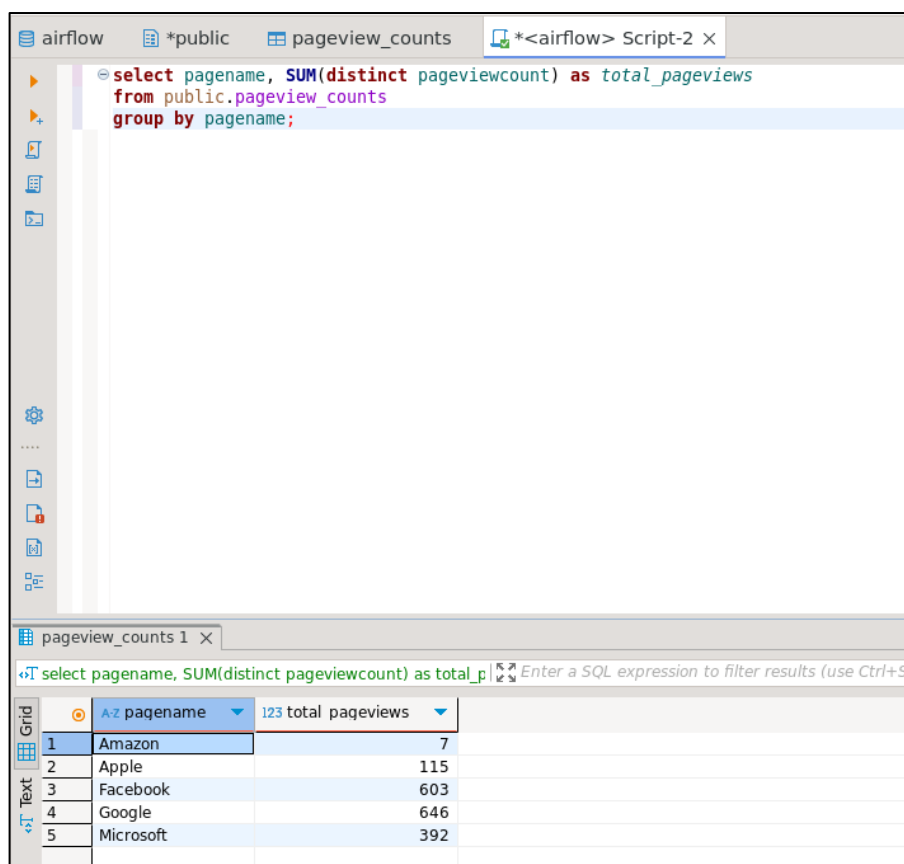
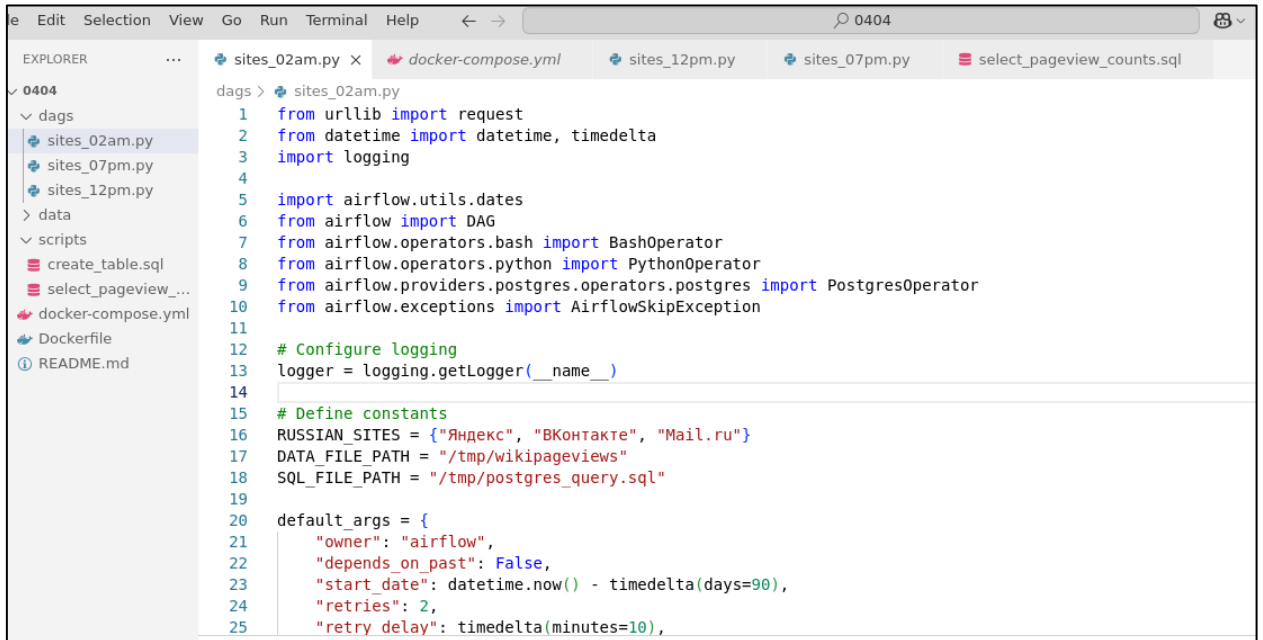


Рисунок 20 – Вывод SQL-запроса

Индивидуальное задание

1. Получить данные за 3 месяца для сайта Yandex, VK, Mail.ru

Создадим sites_12pm.py, sites_02am.py, sites_07pm.py:



```
e Edit Selection View Go Run Terminal Help 0404
EXPLORER  ...  sites_02am.py x  docker-compose.yml  sites_12pm.py  sites_07pm.py  select_pageview_counts.sql
0404
  dags
    sites_02am.py
    sites_07pm.py
    sites_12pm.py
  data
  scripts
    create_table.sql
    select_pageview_...
    docker-compose.yml
    Dockerfile
    README.md

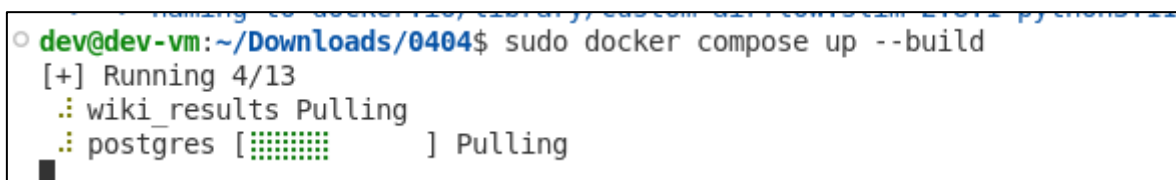
dags > sites_02am.py
1  from urllib import request
2  from datetime import datetime, timedelta
3  import logging
4
5  import airflow.utils.dates
6  from airflow import DAG
7  from airflow.operators.bash import BashOperator
8  from airflow.operators.python import PythonOperator
9  from airflow.providers.postgres.operators.postgres import PostgresOperator
10 from airflow.exceptions import AirflowSkipException
11
12 # Configure logging
13 logger = logging.getLogger(__name__)
14
15 # Define constants
16 RUSSIAN_SITES = {"Яндекс", "ВКонтакте", "Mail.ru"}
17 DATA_FILE_PATH = "/tmp/wikipageviews"
18 SQL_FILE_PATH = "/tmp/postgres_query.sql"
19
20 default_args = {
21     "owner": "airflow",
22     "depends_on_past": False,
23     "start_date": datetime.now() - timedelta(days=90),
24     "retries": 2,
25     "retry_delay": timedelta(minutes=10),
```

Рисунок 21 – Код для DAG

Изменим select_pageview_counts.sql для просмотра данных о средних просмотрах и сумме :

```
SELECT x.pagename, x.hr AS "hour", x.average AS "average pageviews"
FROM (
SELECT
pagename,
date_part('hour', datetime) AS hr,
AVG(pageviewcount) AS average,
ROW_NUMBER() OVER (PARTITION BY pagename ORDER BY AVG(pageviewcount) DESC) AS row_number
FROM pageview_counts
WHERE pagename IN ("Яндекс", "ВКонтакте", "Mail.ru")
GROUP BY pagename, hr
) AS x
WHERE x.row_number = 1;
```

Забилдим образ



```
dev@dev-vm:~/Downloads/0404$ sudo docker compose up --build
[+] Running 4/13
  :: wiki_results Pulling
  :: postgres [.....] Pulling
```

Рисунок 22

Посмотрим, что происходит в таблице, убедимся, что данные записываются:

airflow

public

pageview_counts x

<airflow> Script-2

Properties

Data

Diagram

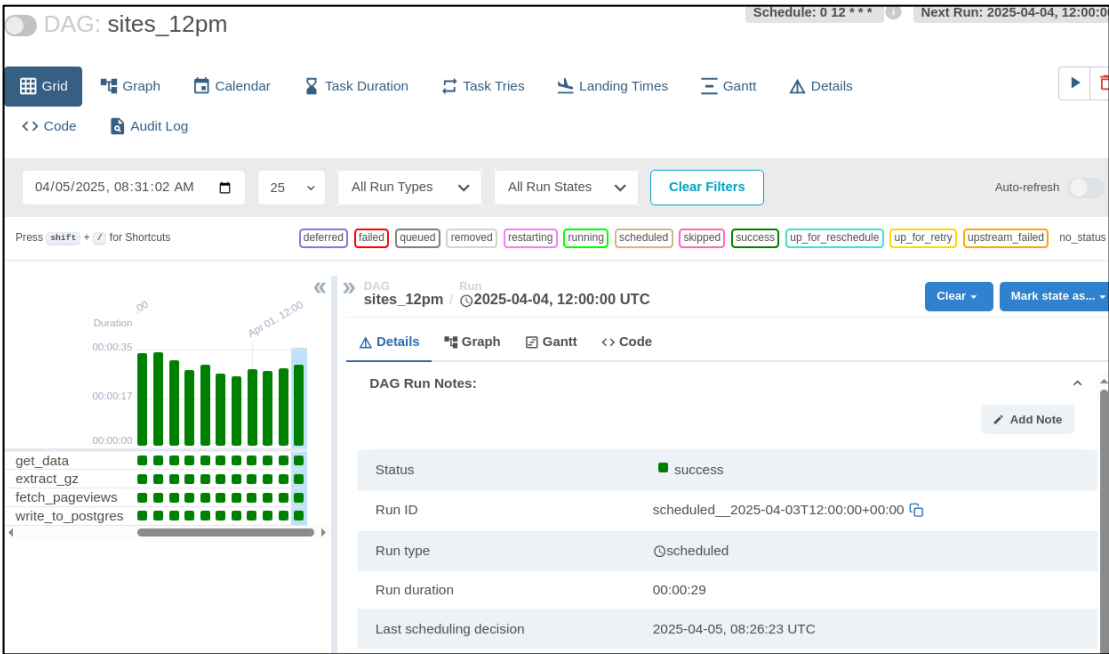
pageview_counts

Enter a SQL expression to filter results (use Ctrl+Space)

	A-Z pagename	123 pageviewcount	datetime
1	Яндекс	930	2025-01-05 12:00:00.000
2	Mail.ru	93	2025-01-05 12:00:00.000
3	ВКонтакте	481	2025-01-05 12:00:00.000
4	Яндекс	991	2025-01-06 12:00:00.000
5	Mail.ru	117	2025-01-06 12:00:00.000
6	ВКонтакте	542	2025-01-06 12:00:00.000
7	Яндекс	1,153	2025-01-07 12:00:00.000
8	Mail.ru	85	2025-01-07 12:00:00.000
9	ВКонтакте	506	2025-01-07 12:00:00.000
10	Яндекс	1,172	2025-01-08 12:00:00.000
11	Mail.ru	126	2025-01-08 12:00:00.000
12	ВКонтакте	523	2025-01-08 12:00:00.000
13	Яндекс	1,247	2025-01-09 12:00:00.000
14	Mail.ru	169	2025-01-09 12:00:00.000
15	ВКонтакте	626	2025-01-09 12:00:00.000
16	Яндекс	1,138	2025-01-10 12:00:00.000
17	Mail.ru	180	2025-01-10 12:00:00.000
18	ВКонтакте	589	2025-01-10 12:00:00.000

Рисунок 23

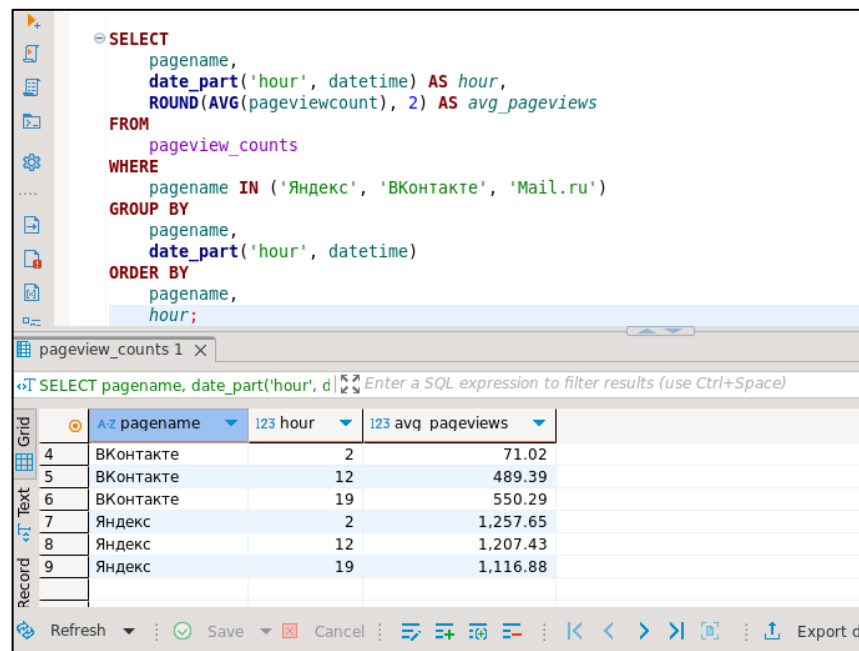
Каждый день обрабатывается по 30 секунд:



2. Постройте график для отображения средних значений просмотров по часам для данных сайтов, используя информацию, сохраненную в базе данных.

Выведем среднее кол-во просмотров по часам

```
SELECT
    pagename,
    date_part('hour', datetime) AS hour,
    ROUND(AVG(pageviewcount), 2) AS avg_pageviews
FROM
    pageview_counts
WHERE
    pagename IN ('Яндекс', 'ВКонтакте', 'Mail.ru')
GROUP BY
    pagename,
    date_part('hour', datetime)
ORDER BY
    pagename,
    hour;
```



The screenshot shows a SQL IDE interface. The top pane displays a SQL query that selects page names, the hour of the day, and the average page view count, grouped by page name and hour. The bottom pane shows the results of the query in a grid view. The grid has three columns: 'pagename', 'hour', and 'avg pageviews'. The data is sorted by page name and then by hour. The results show that 'Яндекс' has significantly higher average page views than 'ВКонтакте' and 'Mail.ru'.

	A-Z pagename	hour	avg pageviews
4	ВКонтакте	2	71.02
5	ВКонтакте	12	489.39
6	ВКонтакте	19	550.29
7	Яндекс	2	1,257.65
8	Яндекс	12	1,207.43
9	Яндекс	19	1,116.88

Рисунок 25 – Вывод SQL-запроса

Реализуем выполнение кода для создания графика по выгруженному csv:

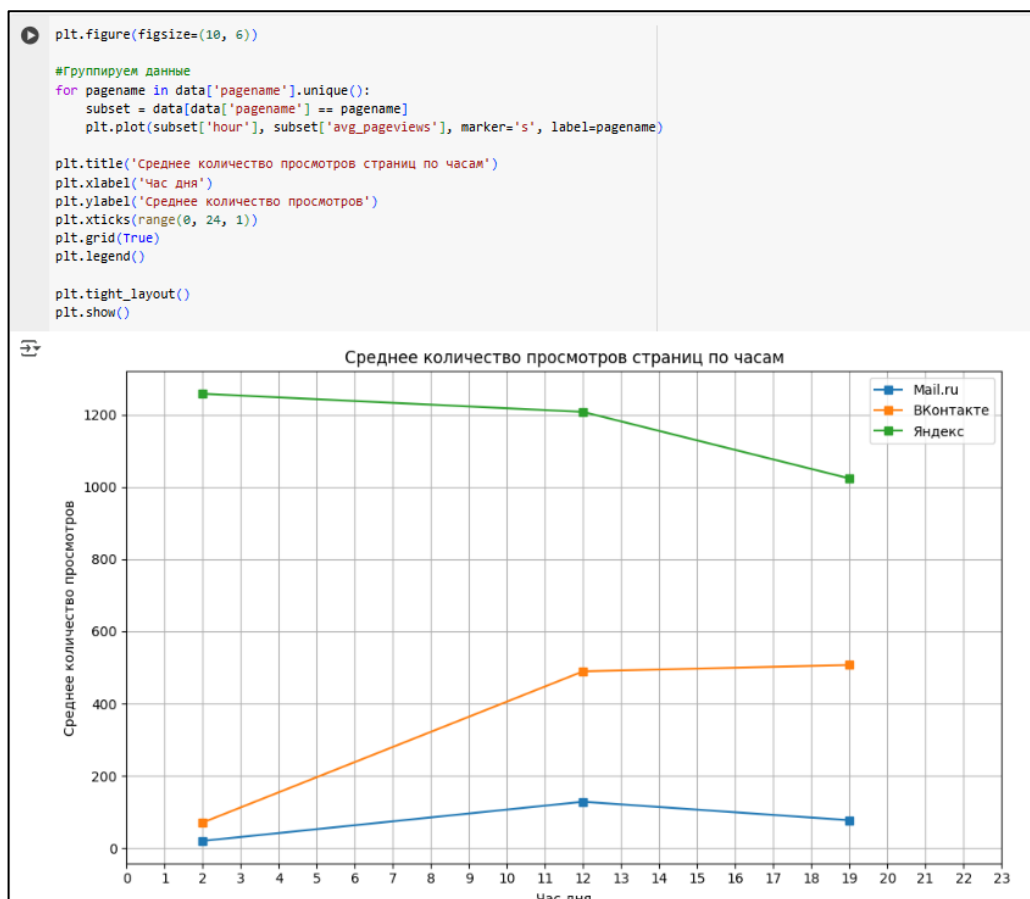


Рисунок 26 – График, отражающий среднее кол-во просмотров страниц по часам

По графику видно, что:

- Высокая активность в Яндекс приходится на 02:00 UTC;
- Высокая активность в ВКонтакте приходится на 19:00 UTC;
- Высокая активность Mail.ru приходится на 12:00 UTC.

По этим данным можно ориентироваться при выборе времени для публикации рекламы на том или ином сайте.

Выводы:

В ходе работы нам удалось выполнить запрос с сайта Wikimedia для анализа числа просмотров сайтов. Мы выгрузили информацию в postgres и с помощью SQL запросов выполнили бизнес-задачу для решения на каком сайте закупать рекламу.