онлайн
университет

Итоговый проект по программе «Дата-инженер»

Разработал:

Олейников Михаил Николаевич

Цели проекта

онлайн университет

Разработка MVP системы, которая собирает, обрабатывает и анализирует данные по использованию услуги интерактивного телевидения, предоставляемой компанией Ростелеком. Система должна дать представление о поведении пользователей, популярности контента, частоте и длительности сессий просмотра.

Поставленные задачи

онлайн университет

Реализовать следующие пункты технического задания:

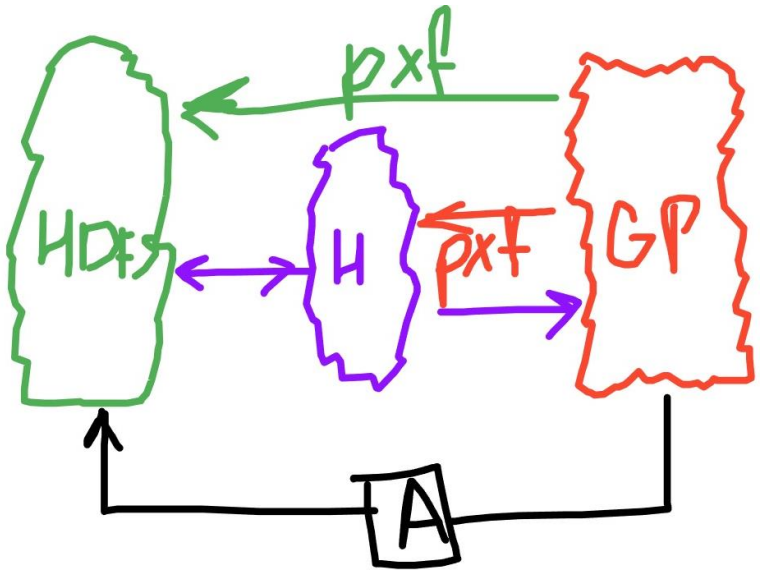
1. Сбор данных с использованием RT.Streaming:
   Создание продюсера на Python, который будет симулировать данные о поведении пользователей интерактивного телевидения и отправлять их в топик Kafka.
2. Хранение сырых данных в RT.DataLake:
   Создание потребителя на Python для RT.Streaming, который будет читать данные и сохранять их в HDFS в формате CSV.
3. Обработка и агрегация данных с использованием Apache Hive в RT.DataLake:
   Создание таблиц Hive для хранения данных из HDFS. Написание запросов для агрегации данных, таких как:
   - общее время просмотра по дням
   - популярность различного контента
   - активность пользователей по времени суток и т.д.

Поставленные задачи

онлайн университет

4. Перенос данных в GreenPlum (RT.Warehouse) и / или ClickHouse (RT.WideStore)
   Настройка процесса ETL на основании Apache Airflow / Nifi продукта RT.Streaming, чтобы перенести обработанные данные из HDFS/Hive в GreenPlum/ClickHouse для сложных аналитических запросов.
5. Аналитика с использованием Python (Apache Zeppelin + Apache Spark = RT.DataLake)
   Использование библиотек Python для глубокого анализа данных, выявления инсайтов по данным, прогнозирования поведения пользователей
6. Визуализация данных с использованием Apache Superset (продукт RT.DataVision)
   Создание интерактивных дашбордов на основе данных из GreenPlum и ClickHouse. Дашборды могут включать в себя графики такие как
   - активности пользователей
   - рейтинг просмотра каналов
   - гистограммы длительности просмотров

Поставленные задачи



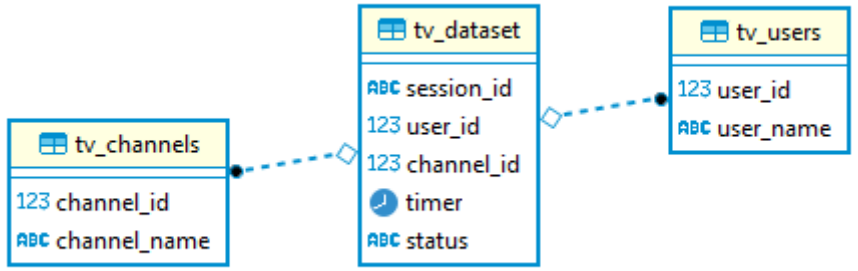
- сбор данных с использованием распределенного программного брокера сообщений Apache Kafka, являющегося компонентом продукта RT.Streaming;
- хранение собранных данных в распределенной файловой системе HDFS, являющейся элементом корпоративного хранилища данных RT.DataLake;
- обработка и агрегация данных с использованием СУБД Apache Hive, так же являющейся компонентом корпоративного хранилища данных RT.DataLake;
- перенос данных в массивно-параллельную СУБД GreenPlum (компонент RT.Warehouse);
- аналитика с помощью фреймворка Apache Spark, так же являющимся элементом корпоративного хранилища данных RT.DataLake;
- визуализация и построение дашборда при помощи платформы Apache Superset, являющейся компонентом продукта RT.Datavision.

Генерация датасета за предыдущую неделю

онлайн университет

Место	Телеканал	Рейтинг по времени просмотра	Рейтинг по количеству зрителей
1	Россия 1	7.98 %	4.04 %
2	Первый канал	6.79 %	4.29 %
3	РЕН ТВ	5.83 %	3.63 %
4	НТВ	5.7 %	3.23 %
5	ТНТ	4.84 %	2.85 %
6	СТС	4.79 %	3.65 %
7	Домашний	2.91 %	1.81 %
8	Россия 24	2.68 %	2.38 %
9	5 Канал	2.66 %	1.82 %
10	Пятница	2.46 %	1.82 %

Место	Телеканал	Рейтинг по времени пр...	Рейтинг по количеству зр...
1	Россия 1	7.98	4.04
2	Первый канал	6.79	4.29
3	РЕН ТВ	5.83	3.63
4	НТВ	5.7	3.23
5	ТНТ	4.84	2.85
6	СТС	4.79	3.65
7	Домашний	2.91	1.81
8	Россия 24	2.68	2.38
9	5 Канал	2.66	1.82
10	Пятница	2.46	1.82



session_id	user_id	channel_id	timer	status
c877f3d1fdfaceecabf95706b2e1ac82	1	1	2023-09-16 10:25:20	enabled
c877f3d1fdfaceecabf95706b2e1ac82	1	1	2023-09-16 17:25:45	disabled
8e489eb90bc7ca76c6b36858e3317a81	2	1	2023-09-16 16:56:03	enabled
8e489eb90bc7ca76c6b36858e3317a81	2	1	2023-09-16 23:57:01	disabled

channel_id	channel_name
1	Россия 1
2	Первый канал
3	РЕН ТВ
4	НТВ
5	ТНТ

user_id	user_name
1	Рыбаков Егор Иосипович
2	Денисов Эмиль Дмитриевич
3	Копылов Карп Трофимович
4	Никифор Харлампьевич Константинов
5	Большакова Наина Кузьминична

Генерация датасета - код

онлайн университет

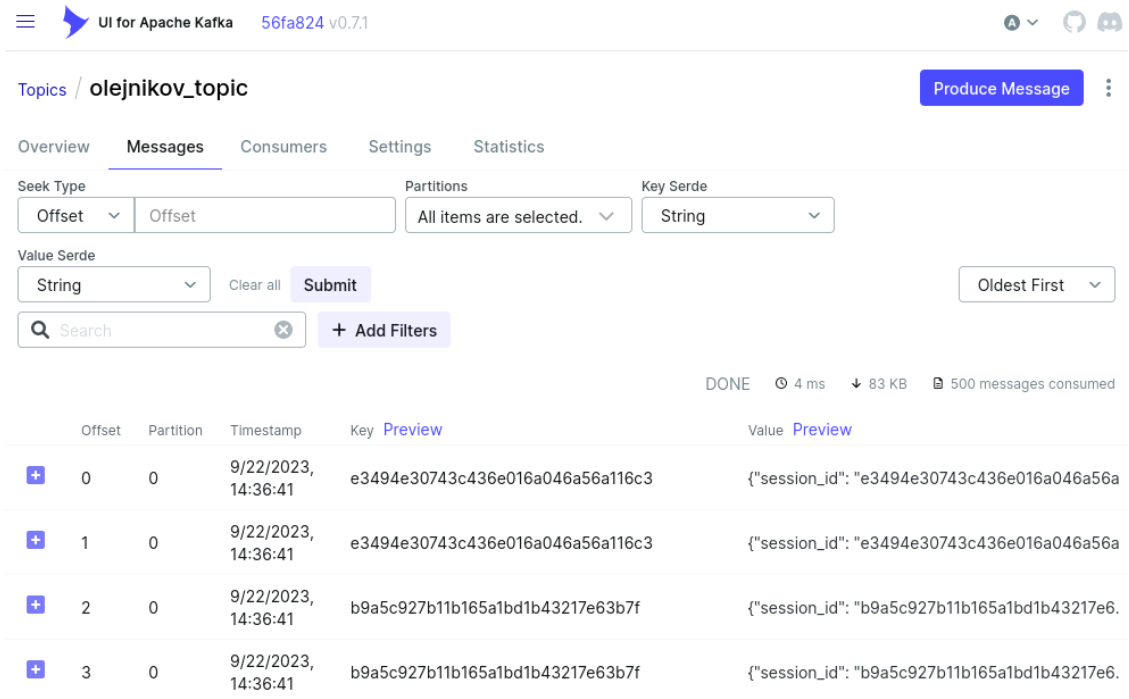
```
import csv
import random
import datetime
import math
from faker import Faker
import time


def make_record(session, user, channel, hour, minute):
    time_start = datetime.datetime.combine(day, datetime.time(hour, random.
    randint(0, 59), random.randint(0, 59)))
    roll = random.randint(1, 5)
    if time_start.weekday() < 5 and roll == 1:
        return
    time_end = time_start + datetime.timedelta(minutes=minute, seconds=random.
    randint(0, 59))
    writer_dataset.writerow([session, user, channel, time_start, 'enabled'])
    writer_dataset.writerow([session, user, channel, time_end, 'disabled'])


start = time.time()
header_dataset = ['session_id', 'user_id', 'channel_id', 'timer', 'status']
header_channels = ['channel_id', 'channel_name']
header_users = ['user_id', 'user_name']
days = 7
user_id = 0

with (open('../tv_rating.csv', 'r', encoding='utf-8') as file_in,
      open('../tv_dataset.csv', 'w', encoding='utf-8', newline='') as dataset,
      open('../tv_channels.csv', 'w', encoding='utf-8', newline='') as
      channels,
      open('../tv_users.csv', 'w', encoding='utf-8', newline='') as users):
    data = file_in.readlines()
    writer_dataset = csv.writer(dataset)
    writer_dataset.writerow(header_dataset)
    writer_channels = csv.writer(channels)
    writer_channels.writerow(header_channels)
    writer_users = csv.writer(users)
    writer_users.writerow(header_users)

    for d in range(days):
        day = datetime.date.today() - datetime.timedelta(days=days) + datetime
        .timedelta(days=d)
        for c in range(1, len(data)):
            print('day:', d + 1, 'channel:', c)
            row = data[c].split(',')
            users = int(float(row[3]) * 100)
            minutes = int(float(row[2])) * 60 + int((float(row[2]) - int(float
            (row[2]))) * 60 / 100)
            if d == 0:
                writer_channels.writerow([int(row[0]), row[1]])

            for i in range(user_id + 1, user_id + users + 1):
                sess = "%032x" % random.getrandbits(128)
                if d == 0:
                    writer_users.writerow([i, Faker('ru_RU').name()])
                per = random.randint(1, 5)
                if per == 1:
                    make_record(sess, i, int(row[0]), random.randint(0, 23 -
                    math.ceil(minutes / 60)), minutes)
                else:
                    make_record(sess, i, int(row[0]), 20 - round(minutes / 120
                    ), minutes)
            user_id += users
    user_id = 0

end = time.time() - start
print(end)
```

Сбор данных с использованием RT.Streaming

онлайн университет

Сбор данных реализован с использованием распределенного программного брокера сообщений Apache Kafka, являющегося компонентом продукта RT.Streaming. Сбор данных реализован на языке Python. Для этого был создан продюсер, который симулирует данные о поведении 10000 пользователей интерактивного телевидения и отправляет их в топик Kafka.

Генерируем активность 10000 пользователей за текущий день.



Сбор данных с использованием RT.Streaming - код

онлайн университет

```
from kafka import KafkaProducer
import json
import random
from datetime import datetime, date, time as dt, timedelta


producer = KafkaProducer(
    bootstrap_servers='vm-strmng-s-1.test.local:9092',
    value_serializer=lambda v: json.dumps(v).encode('utf-8'),
    key_serializer=lambda v: v.encode('utf-8')
)


def generate_record(session_id, status):
    if status == 'disabled':
        timer = queue[session_id][2]
        delta = timedelta(hours=random.randint(0, 3), minutes=random.
        randint(0, 59))
        rec = [session_id, queue[session_id][0], queue[session_id][1], str(
        timer + delta), status]
        queue.pop(session_id)
    else:
        user_id = random.randint(1, users)
        channel_id = random.randint(1, channels)
        per = random.randint(1, 3)
        if per == 1:
            timer = datetime.combine(date.today(), dt(random.randint(3, 20
            ), random.randint(0, 59), random.randint(0, 59)))
        else:
            timer = datetime.combine(date.today(), dt(19, random.randint(0,
             59), random.randint(0, 59)))
        rec = [session_id, user_id, channel_id, str(timer), status]
        queue[session_id] = [user_id, channel_id, timer]
    record = {
        "session_id": rec[0],
        "user_id": rec[1],
        "channel_id": rec[2],
        "timer": rec[3],
        "status": rec[4]
    }

    return record


queue = {}
users = 10000
channels = 148
# header_queue = ['session_id', 'user_id', 'channel_id', 'timer', 'status']

try:
    topic = 'olejnikov_topic'
    i = 0
    while i < users:  # True
        session = "%032x" % random.getrandbits(128)
        data = generate_record(session, 'enabled')
        producer.send(topic, key=data['session_id'], value=data)
        data = generate_record(session, 'disabled')
        producer.send(topic, key=data['session_id'], value=data)
        print(f'User {data["user_id"]} channel {data["channel_id"]} is
        generated')
        i += 1  # time.sleep(0.1)

finally:
    producer.close()
```

Хранение сырых данных в RT.DataLake

онлайн университет

Хранение собранных данных реализовано в распределенной файловой системе HDFS, являющейся элементом корпоративного хранилища данных RT.DataLake. Хранение осуществляется с помощью потребителя на Python, который читает данные с Kafka и сохраняет их в HDFS в формате CSV.

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxrw-rw-	olejnikov	hadoop	0 B	Aug 04 08:00	0	0 B	.Trash
drwxrwxrwx	olejnikov	hadoop	0 B	Aug 02 08:12	0	0 B	.sparkStaging
-rw-r--r--	dr.who	hadoop	3.14 KB	Sep 23 06:23	3	128 MB	tv_channels.csv
-rw-r--r--	dr.who	hadoop	8.13 MB	Sep 23 06:23	3	128 MB	tv_dataset.csv
-rw-r--r--	olejnikov	hadoop	1.33 MB	Sep 23 08:01	3	128 MB	tv_stream.csv
-rw-r--r--	dr.who	hadoop	551.02 KB	Sep 23 06:23	3	128 MB	tv_users.csv

Хранение сырых данных в RT.DataLake - код

онлайн университет

```
from kafka import KafkaConsumer
import json
import pandas as pd
from hdfs import InsecureClient

topic = 'olejnikov_topic'
hdfs_host = 'http://vm-dlake2-m-1:9870'
hdfs_path = '/user/olejnikov'
# header_dataset = ['session_id', 'user_id', 'channel_id', 'timer', 'status']
message_count = 10000  # users from tx
client = InsecureClient(hdfs_host, user='olejnikov')

consumer = KafkaConsumer(topic,
                         bootstrap_servers='vm-strmng-s-1.test.local:9092',
                         group_id='olejnikov_group'
                         )

rows = []  # [header_dataset]

try:
    i = 0
    for message in consumer:
        # print(f"New messgae: {message.value.decode('utf-8')}")
        data = json.loads(message.value.decode('utf-8'))
        rows.append(list(data.values()))
        if i == message_count * 2 - 1:
            df = pd.DataFrame(rows)
            with client.write(hdfs_path + '/tv_stream.csv', encoding='utf-8') as tv_stream:
                df.to_csv(tv_stream, header=None, index=False)
                print('Done')
        i += 1

finally:
    consumer.close()
```

Обработка и агрегация данных с использованием Apache Hive в RT.DataLake

онлайн университет

```
create external table olejnikov.tv_users (
    user_id int,
    user_name varchar(50)
)
row format delimited fields terminated by
','
lines terminated by '\n'
tblproperties("skip.header.line.count"="1");

load data inpath
'/user/olejnikov/tv_users.csv' overwrite
into table olejnikov.tv_users;

select * from olejnikov.tv_users limit 5
```

Обработка и агрегации данных реализована с использованием СУБД Apache Hive, являющейся компонентом корпоративного хранилища данных RT.DataLake. При это расположенные на HDFS сырые данные загружаются и преобразовываются в таблицы Hive.

session_id	user_id	channel_id	timer	status
c877f3d1fdfaceecabf95706b2e1ac82	1	1	2023-09-16 10:25:20.000	enabled
c877f3d1fdfaceecabf95706b2e1ac82	1	1	2023-09-16 17:25:45.000	disabled
8e489eb90bc7ca76c6b36858e3317a81	2	1	2023-09-16 16:56:03.000	enabled
8e489eb90bc7ca76c6b36858e3317a81	2	1	2023-09-16 23:57:01.000	disabled
42e3b49eabda9fd99e2cbff5b5719685	3	1	2023-09-16 16:02:53.000	enabled

Обработка и агрегация данных с использованием Apache Hive в RT.DataLake

онлайн университет

```
create view tv_dataset_v as
with ts as (
    select session_id, user_id, channel_id, timer as time_start
    from (select * from tv_dataset where status = 'enabled') t1
    union
    select session_id, user_id, channel_id, timer as time_start
    from (select * from tv_stream where status = 'enabled') t2
), te as (
    select session_id, timer as time_end
    from (select * from tv_dataset where status = 'disabled') t3
    union
    select session_id, timer as time_end
    from (select * from tv_stream where status = 'disabled') t4
)
select ts.session_id, user_id, channel_name, time_start, time_end
from ts
join te on ts.session_id = te.session_id
join tv_channels on ts.channel_id = tv_channels.channel_id

select * from tv_dataset_v limit 5
```

Для агрегации нескольких таблиц в одну используется виртуальная таблица. В ней уже нет колонки со статусом начала и конца сессии, а непосредственно указаны ее время начали и конца. Вместо id канала стоит его название.

session_id	user_id	channel_name	time_start	time_end
000718b18ee41c39a73c7aa40eb125cd	1,171	Русский бестселлер	2023-09-23 19:43:10.000	2023-09-23 21:42:10.000
00081155034cba88e8d8afb7aded6f81	66,006	Кинопоказ	2023-09-22 08:32:58.000	2023-09-22 08:33:10.000
000c0ff5e40fa1cc34b42624f67e6dbd	57,893	Иллюзион плюс	2023-09-21 20:01:07.000	2023-09-21 20:01:07.000
00113b378de9ca5d2ae0aa36ee9dd32e	9,375	Уникум	2023-09-16 20:38:52.000	2023-09-16 20:39:43.000
0016e6584372aaf8ae993c7cfb171034	32,208	5 Канал	2023-09-19 01:33:01.000	2023-09-19 03:33:55.000

Обработка и агрегация данных с использованием Apache Hive в RT.DataLake. Аналитические запросы.

онлайн университет

```
-- Рейтинг каналов
select channel_name, round((sum(UNIX_TIMESTAMP(time_end)
 - UNIX_TIMESTAMP(time_start))) / 3600) as sum_time,
count(channel_name) as count_views
from tv_dataset_v
group by channel_name
order by sum_time desc
```

channel_name	sum_time	count_views
Россия 1	15,839	2,313
Первый канал	14,515	2,460
РЕН ТВ	10,703	2,188
НТВ	9,587	1,948
ТНТ	8,526	1,752

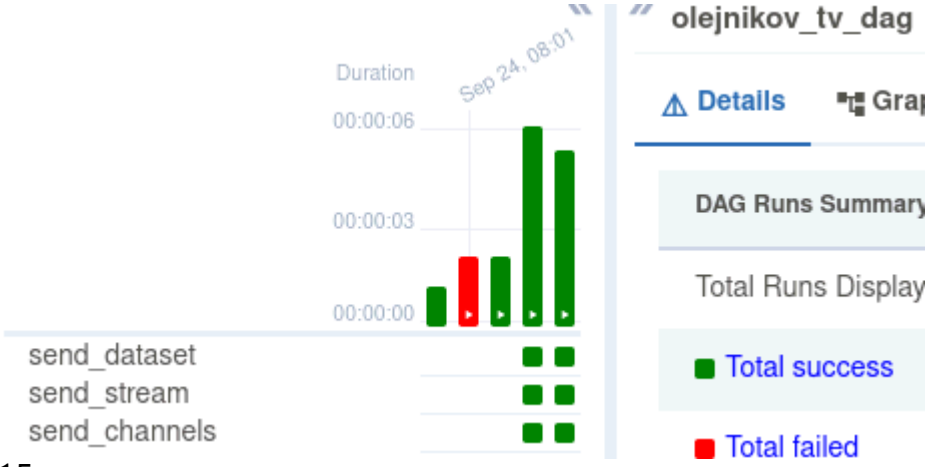
```
-- Количество просмотров по дням
select to_date(time_start) as time_dates, round((sum(
UNIX_TIMESTAMP(time_end) - UNIX_TIMESTAMP(time_start)))
/ 3600) as sum_time, count(*) as count_views
from tv_dataset_v
group by to_date(time_start)
order by to_date(time_start)
```

time_dates	sum_time	count_views
2023-09-16	16,169	9,990
2023-09-17	16,168	9,990
2023-09-18	12,935	8,002
2023-09-19	12,940	7,973
2023-09-20	12,925	8,017
2023-09-21	12,864	7,968
2023-09-22	12,866	7,964
2023-09-23	19,882	10,000

Перенос данных в GreenPlum (RT.Warehouse)

онлайн университет

Перенос обработанных данных из Hive в массивно-параллельную СУБД GreenPlum (компонент RT.Warehouse) реализован при помощи ETL процесса – ориентированного ациклического графа Airflow DAG. В Greenplum SQL-диалект свежее и можно построить более сложный запрос – например распределение количества просмотров в разрезе каждого часа.




```
select hours, count(hours) as
count_views
from (
    select generate_series(date_part(
    'hour', time_start::time)::int,
    date_part('hour', time_end::time
    )::int) as hours
    from olejnikov_tv_mv) h
group by hours
order by hours
```

123 hours	123 count_views
0	504
1	812
2	998
3	1,354
4	1,674
5	1,908
6	2,077
7	2,099
8	2,066
9	2,063
10	2,051
11	2,065
12	1,995
13	2,045
14	2,004
15	1,992
16	3,853
17	5,774
18	12,693
19	25,103
20	55,598
21	32,097
22	13,509
23	9,660

Перенос данных в GreenPlum (RT.Warehouse)

онлайн университет

```
from airflow import DAG
from airflow.utils.dates import days_ago
from airflow.operators.postgres_operator import PostgresOperator

DAG_NAME = 'olejnikov_tv_dag'
GP_CONN_ID = 'olejnikov_conn'

DATASET = '''
create external table olejnikov_tv_dataset (
    session_id varchar(32),
    user_id int,
    channel_id int,
    timer timestamp,
    status varchar(8)
)
location('pxf://olejnikov.tv_dataset?PROFILE=hive&server=hadoop')
format 'custom' (formatter='pxfwritable_import');
'''

STREAM = '''
create external table olejnikov_tv_stream (
    session_id varchar(32),
    user_id int,
    channel_id int,
    timer timestamp,
    status varchar(8)
)
location('pxf://olejnikov.tv_stream?PROFILE=hive&server=hadoop')
format 'custom' (formatter='pxfwritable_import');
'''

CHANNELS = '''
create external table olejnikov_tv_channels (
    channel_id int,
    channel_name varchar(50)
)
location('pxf://olejnikov.tv_channels?PROFILE=hive&server=hadoop')
format 'custom' (formatter='pxfwritable_import');
'''
```

```
args = {'owner': 'olejnikov',
        'start_date': days_ago(0),
        'depends_on_past': False}

with DAG(
    DAG_NAME, description='olejnikov_tv',
    schedule_interval='@once',
    catchup=False,
    max_active_runs=1,
    default_args=args,
    params={'labels': {'env': 'prod', 'priority': 'high'}}) as dag:

    send_dataset = PostgresOperator(
        task_id='send_dataset',
        sql=DATASET,
        postgres_conn_id=GP_CONN_ID,
        autocommit=True)

    send_stream = PostgresOperator(
        task_id='send_stream',
        sql=STREAM,
        postgres_conn_id=GP_CONN_ID,
        autocommit=True)

    send_channels = PostgresOperator(
        task_id='send_to_channels',
        sql=CHANNELS,
        postgres_conn_id=GP_CONN_ID,
        autocommit=True)

send_dataset >> send_stream >> send_channels
```

Аналитика (Apache Zeppelin + Apache Spark)

онлайн университет

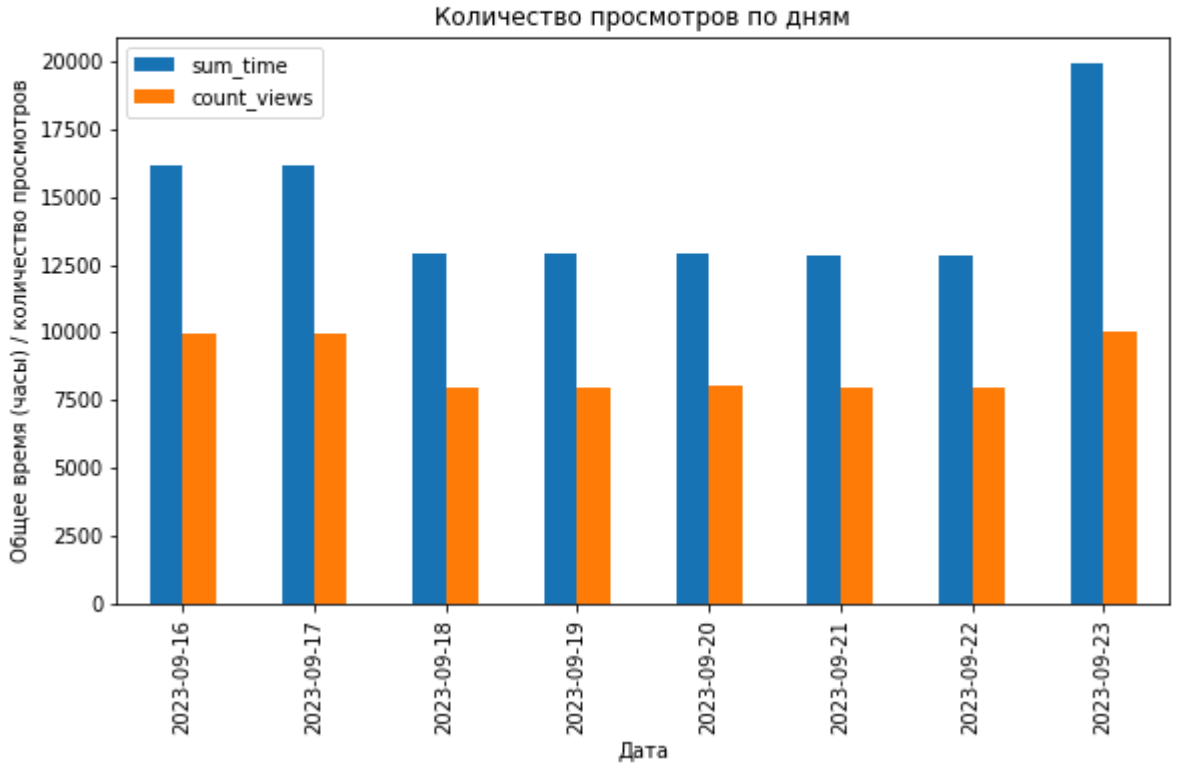
Загружаем виртуальную таблицу из Hive

olejnikov_tv

```
%spark.pyspark

from pyspark.sql import SparkSession

spark = SparkSession.builder\
        .master("local[*]")\
        .appName('olejnikov_tv')\
        .enableHiveSupport()\
        .getOrCreate()

df = spark.sql("select * from olejnikov.tv_dataset_v")

df.show()
```

```
+--------------------+-------+------------------+-------------------+-
|          session_id|user_id|      channel_name|         time_start|
+--------------------+-------+------------------+-------------------+-
|52ac12716e958a22c...|    503|      Первый канал|2023-09-16 17:30:45|2
|0312364efb5c75fc3...|    682|      Первый канал|2023-09-16 17:33:26|2
|8ee7986b4a5866c42...|    926|           РЕН ТВ|2023-09-16 16:03:54|2
|392d735affcf7a1fa...|    987|           РЕН ТВ|2023-09-16 18:54:29|2
|fd80418e594409f98...|   1037|           РЕН ТВ|2023-09-16 18:26:41|2
|67c66ea5ea6adc22c...|   1806|               СТС|2023-09-16 18:09:43|2
|f4c73750039200999...|   2251|         5 Канал|2023-09-16 19:17:58|2
```

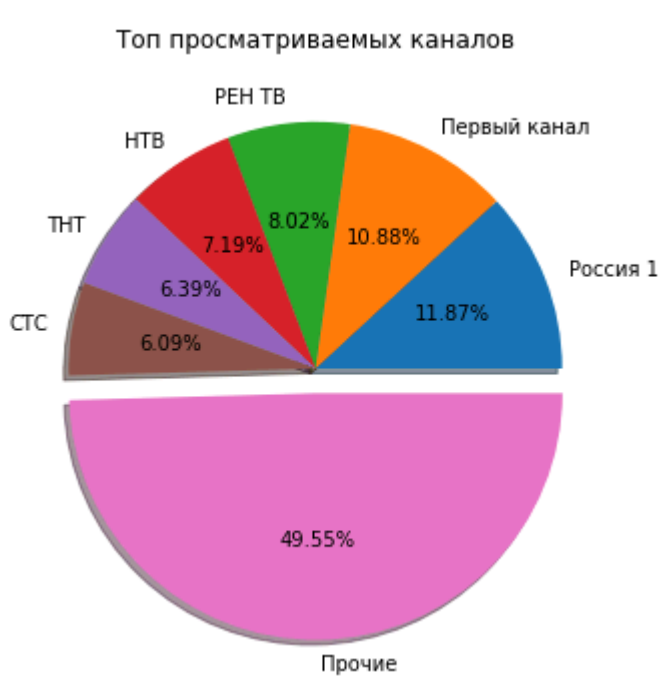
```
df2.plot(x='time_dates', y=['sum_time', 'count_views'], kind='bar')
plt.ylabel('Общее время (часы) / количество просмотров')
plt.xlabel("Дата")
plt.title("Количество просмотров по дням")
plt.tight_layout()
plt.show()
```



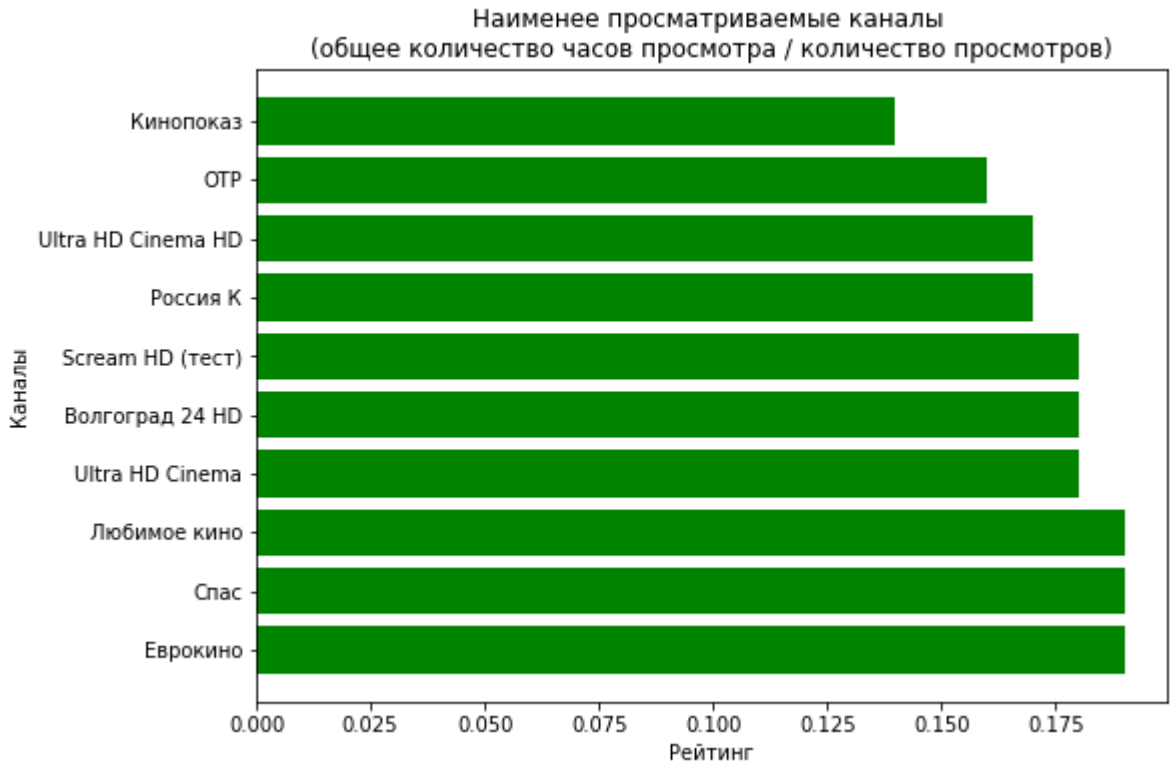
Аналитика (Apache Zeppelin + Apache Spark)

онлайн университет

```
hours = df1['sum_time'].head(6)
hours.loc[len(hours.index)] = df1['sum_time'].tail(144).sum()
channels = df1['channel_name'].head(6)
channels.loc[len(channels.index)] = 'Прочие'
myexplode = [0, 0, 0, 0, 0, 0, 0.1,]
fig, ax = plt.subplots()
ax.pie(hours, labels=channels, autopct='%1.2f%%', explode = \
myexplode, shadow = True)
plt.title("Топ просматриваемых каналов")
plt.show()
```




```
y = df4['channel_name'].tail(10)
x = df4['rating'].tail(10)
plt.barh(y, x, color='green')
plt.ylabel('Каналы')
plt.xlabel("Рейтинг")
plt.title("Наименее просматриваемые каналы \n(общее количество часов просмотра \
/ количество просмотров)")
plt.tight_layout()
plt.show()
```



Визуализация данных в Apache Superset

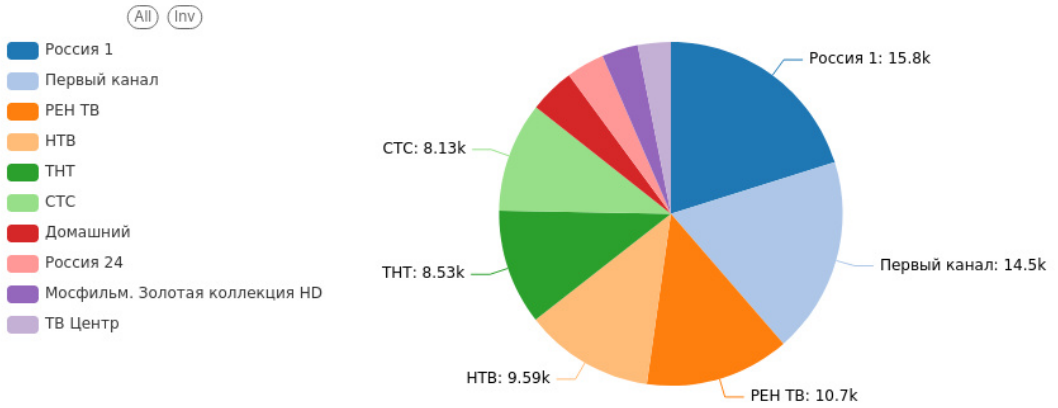
онлайн университет

Рейтинг каналов

Search 148 records...

Телеканал	Часы просмотра	Количество просмотров
Россия 1	15.8k	2.31k
Первый канал	14.5k	2.46k
РЕН ТВ	10.7k	2.19k
НТВ	9.59k	1.95k
ТНТ	8.53k	1.75k
СТС	8.13k	2.06k
Домашний	3.35k	1.13k
Россия 24	2.82k	1.41k
Мосфильм. Золотая коллекция HD	2.68k	1.34k
ТВ Центр	2.44k	1.22k

Топ 10 каналов по времени просмотра

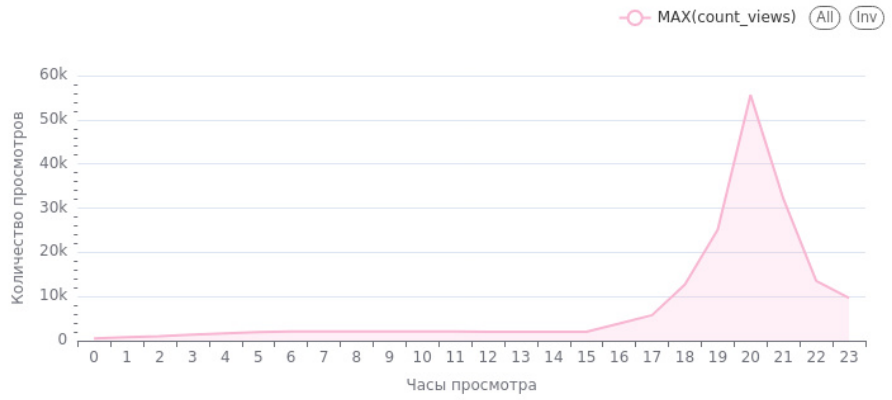


Количество просмотров по дням

Дата	Часы просмотра	Количество просмотров
2023-09-23	19.9k	10k
2023-09-16	16.2k	9.99k
2023-09-17	16.2k	9.99k
2023-09-19	12.9k	7.97k
2023-09-18	12.9k	8k
2023-09-20	12.9k	8.02k
2023-09-22	12.9k	7.96k
2023-09-21	12.9k	7.97k

Количество просмотров по дням

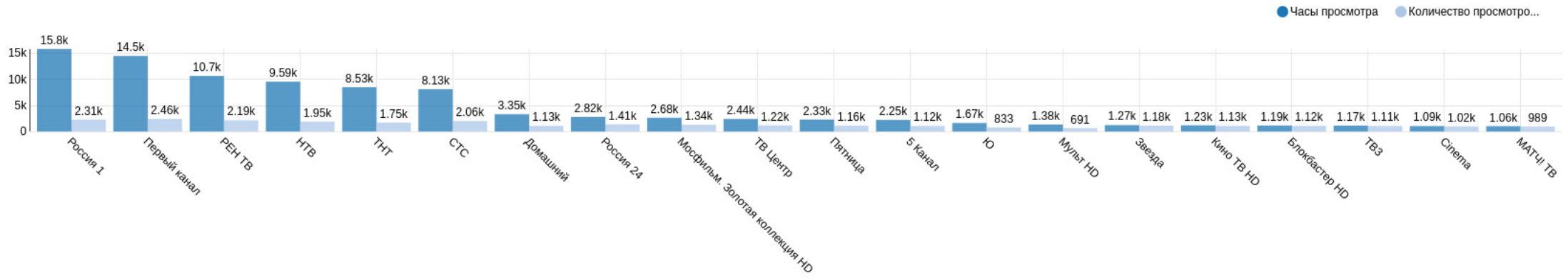
Количество просмотров в разрезе каждого часа



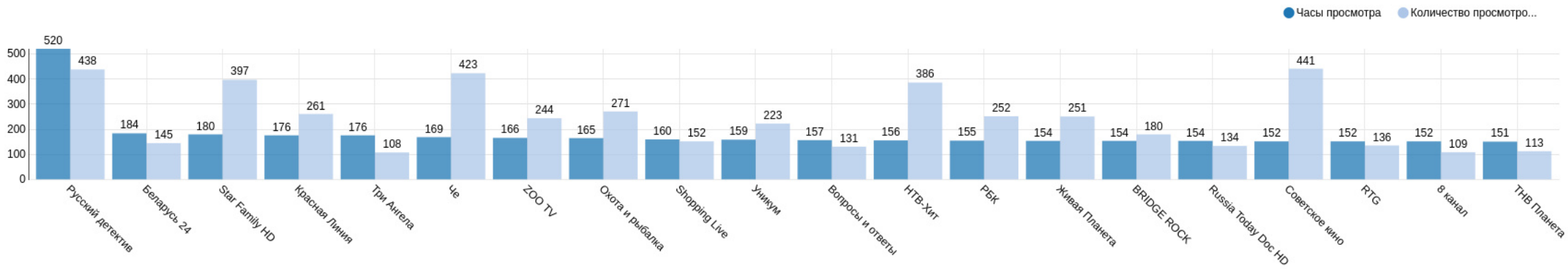
Визуализация данных в Apache Superset

онлайн университет

Топ 20 каналов



Топ 20 аутсайдеров



Итоги и инсайты

По итогам работы удалось реализовать систему анализа и сбора данных услуги интерактивного телевидения с использованием продуктов платформы управления данными ПАО «Ростелеком».

При помощи аналитических инструментов выявлены наиболее популярные каналы у пользователе как по количеству, так и по длительности просмотра. А так же каналы аутсайдеры. Выявлено использование услуги интерактивного телевидения в разрезе часов, дней недели.

На основании запросов были построены аналитические графики, витрина данных.

Список ссылок

онлайн университет

1. GitHub проекта: https://github.com/Olmeor/Data-engineer_Rostelecom_programming_school
2. Рейтинг популярности телеканалов: https://www.powernet.com.ru/channels-stat
3. Аналитика просмотров: https://journal.tinkoff.ru/television-stat/
4. Spark notebook - http://vm-dlake2-m-2:8180/#/notebook/2JBN33XNC (только внутри кластера)
5. Superset http://vm-datavision.test.local:8090/superset/dashboard/p/rYymq0Yxn3R/ (только внутри кластера)
6. Платформа управления данными ПАО «Ростелеком» https://data.rt.ru/products

онлайн
университет

Спасибо за внимание!

Готов ответить на ваши вопросы

Разработал:
Олейников Михаил Николаевич