Кейс Umbrella

Клонирование репозитория:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

mgpu@mgpu-VirtualBox:~/Downloads$ git clone https://github.com/BosenkoTM/DCCAS
Cloning into 'DCCAS'...
remote: Enumerating objects: 23, done.
remote: Counting objects: 100% (23/23), done.
remote: Compressing objects: 100% (21/21), done.
remote: Total 23 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (23/23), 8.18 KiB | 4.09 MiB/s, done.
```

Запуск контейнера с кейсом:

```
mgpu@mgpu-VirtualBox:~/Downloads$ cd DCCAS
mgpu@mgpu-VirtualBox:~/Downloads/DCCAS$ CD business_case_umbrella/
CD: command not found
mgpu@mgpu-VirtualBox:~/Downloads/DCCAS$ cd business_case_umbrella/
mgpu@mgpu-VirtualBox:~/Downloads/DCCAS/business_case_umbrella$ ls
dags  docker-compose.yml  README.md
mgpu@mgpu-VirtualBox:~/Downloads/DCCAS/business_case_umbrella$
```

Описание элементов

Конструкция 1:

```
dag = DAG(
    dag_id="01_umbrella",
    description="Umbrella example with DummyOperators.",
    start_date=airflow.utils.dates.days_ago(5),
    schedule_interval="@daily",
)
```

В конструкции задаются базовые параметры: ИД dag’а, его описание, дата начала работы и интервал выполнения (в данном случае – ежедневно).

Конструкция 2:

```
fetch_weather_forecast = DummyOperator(task_id="fetch_weather_forecast", dag=dag)
fetch_sales_data = DummyOperator(task_id="fetch_sales_data", dag=dag)
clean_forecast_data = DummyOperator(task_id="clean_forecast_data", dag=dag)
clean_sales_data = DummyOperator(task_id="clean_sales_data", dag=dag)
join_datasets = DummyOperator(task_id="join_datasets", dag=dag)
train_ml_model = DummyOperator(task_id="train_ml_model", dag=dag)
deploy_ml_model = DummyOperator(task_id="deploy_ml_model", dag=dag)
```

В конструкции задаются таски, которые необходимо выполнить (с помощью дамми операторов). Последовательность тасков:

1. Получение данных о прогнозах погоды;
2. Получение данных о продажах;
3. Очистка данных из шага 1;
4. Очистка данных из шага 2;
5. Объединение данных;
6. Обучение модели МЛ;
7. Применение модели из шага 6.

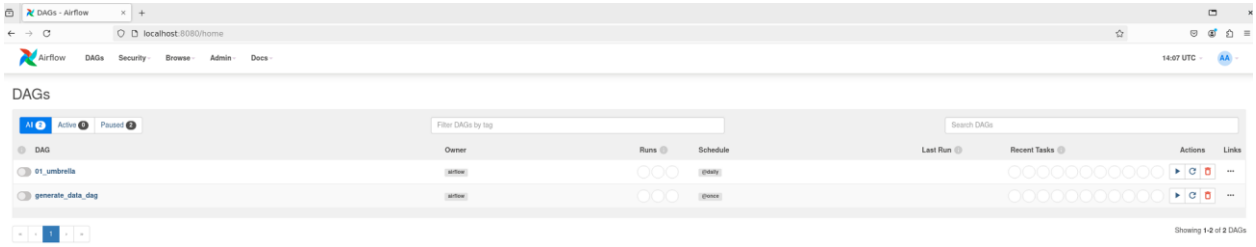
Конструкция 3:

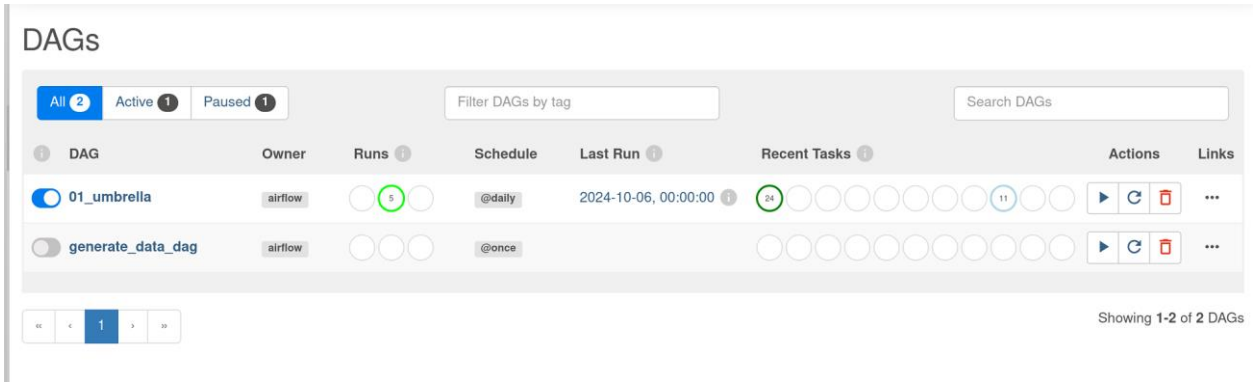
```
# Set dependencies between all tasks
fetch_weather_forecast >> clean_forecast_data
fetch_sales_data >> clean_sales_data
[clean_forecast_data, clean_sales_data] >> join_datasets
join_datasets >> train_ml_model >> deploy_ml_model
```

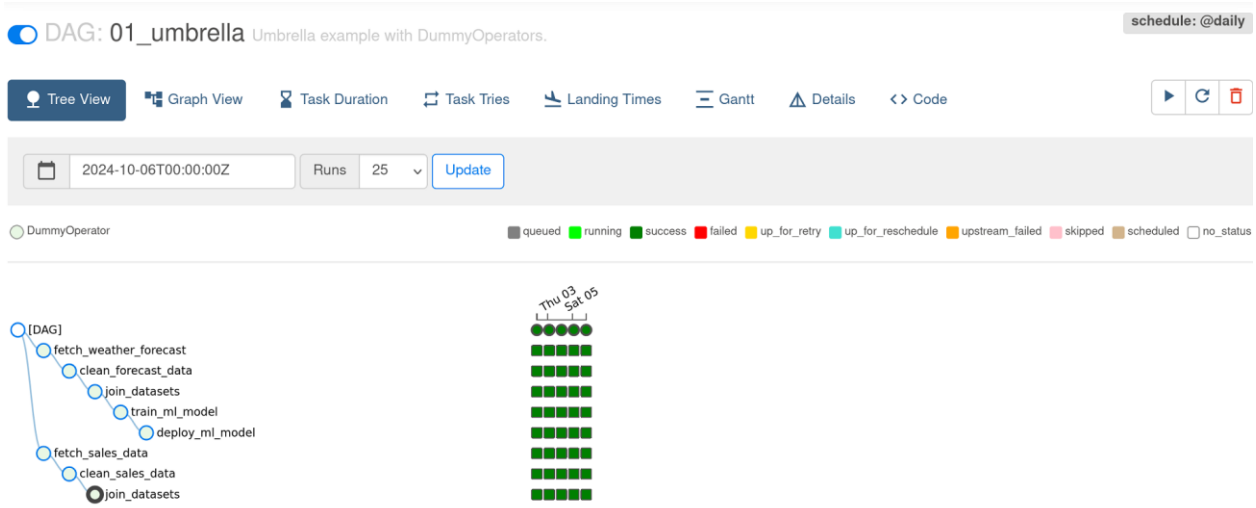
В конструкции задается зависимость между тасками, например, во второй строке устанавливается следующая зависимость: перед очисткой данных о прогнозе погоды их необходимо получить. В четвертой строке устанавливается следующая зависимость: перед объединением данных данные о прогнозе и данные о продаже необходимо очистить.

Запуск среды

```
mgpu@mgpu-VirtualBox:~/Downloads/DCCAS/business_case_umbrella$ sudo docker compose up -d
[sudo] password for mgpu:
[+] Running 4/5
 ⠿ Network business_case_umbrella_default        Created
 ✔ Container business_case_umbrella-postgres-1   Started
 ✔ Container business_case_umbrella-webserver-1  Started
 ✔ Container business_case_umbrella-scheduler-1  Started
 ✔ Container business_case_umbrella-init-1       Started
```

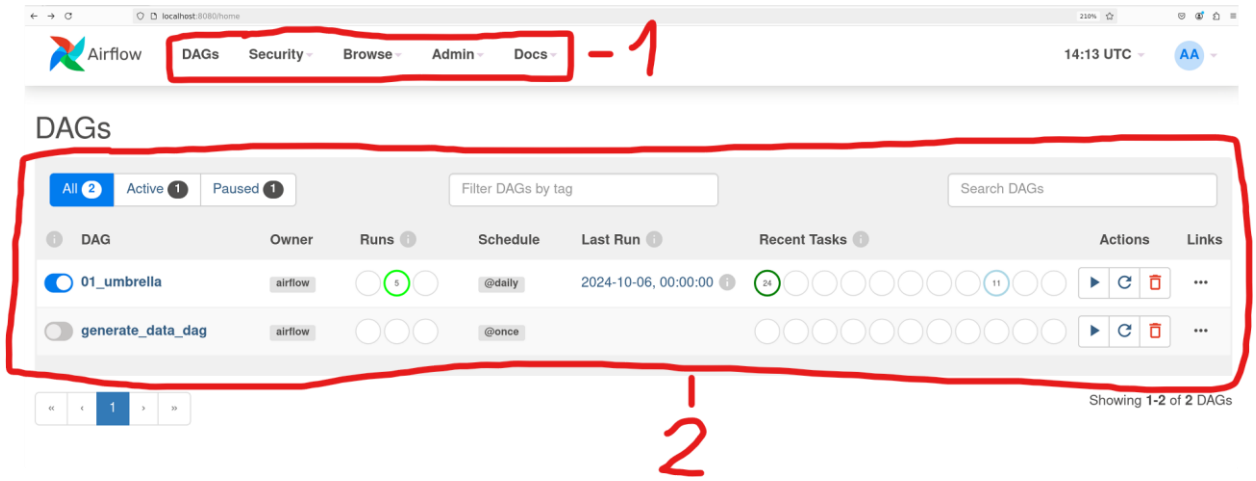






Описание интерфейса Airflow

При открытии страницы в браузере выводится следующий интерфейс:



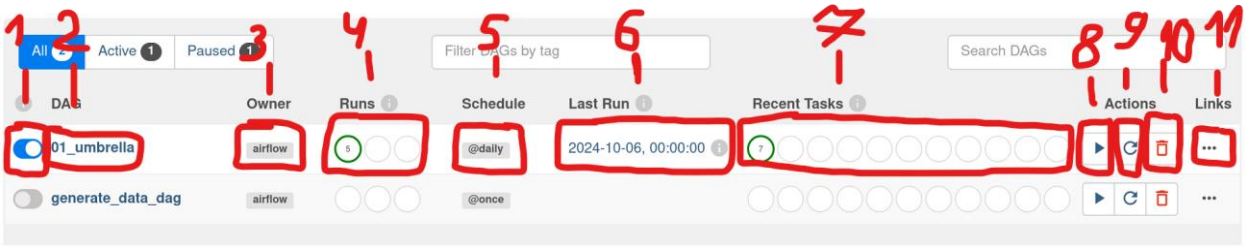
1 – Модули Airflow

2 – Блок с информацией о dag’ах

Доступны следующие модули:

1. DAGs – страница с информацией о dag’ах (открыта по умолчанию);
2. Security – управление безопасностью (пользователи – роли – разрешения);
3. Browse – просмотр информации о прогонах (runs) dag’ов, джобах, тасках и других элементах;
4. Admin – управление подключениями, плагинами и другими элементами;
5. Docs – документация.

Блок DAGs содержит следующие элементы:



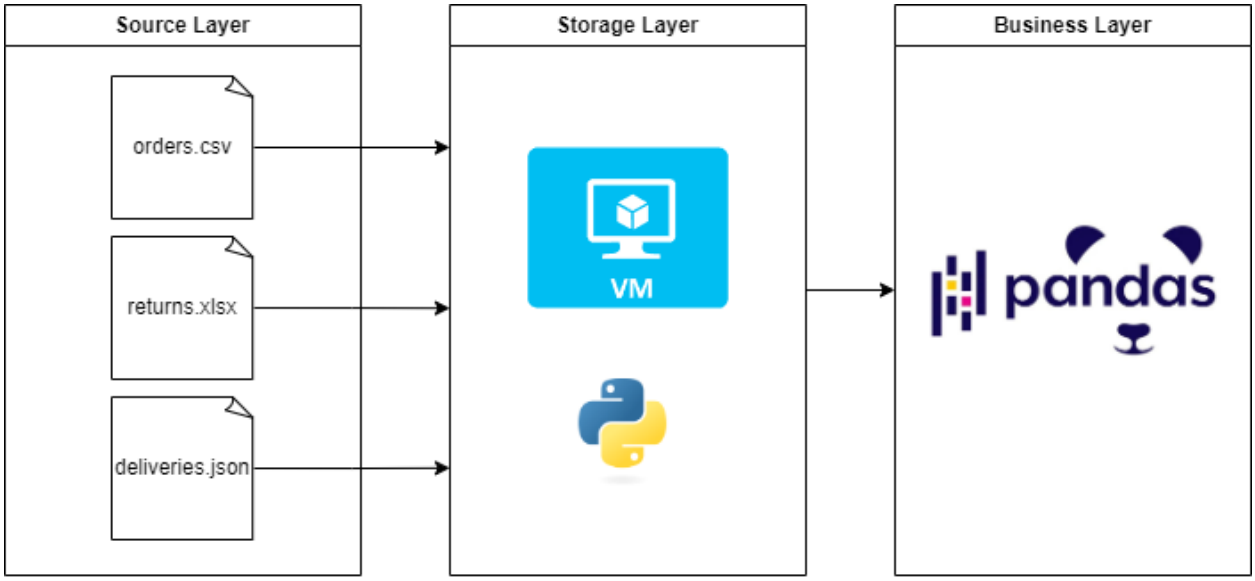
1. Кнопка включения/приостановки;
2. Название;
3. Владелец;
4. Прогоны (runs);
5. Частота выполнения (на скриншоте – ежедневно)
6. Время последнего выполнения;
7. Недавние таски с выделением статусов цветом;
8. Принудительное выполнение;
9. Обновление;
10. Удаление dag’а;
11. Быстрые ссылки на различные представления и информацию по dag’у.

Если нажать на название dag’а, можно изучить информацию о его работе в виде различных представлений:

1. Представления в виде дерева (отображает конкретные таски, их взаимосвязь и попытки выполнения);
2. Представление в виде графа;
3. Продолжительность выполнения тасков;
4. Попытки выполнения тасков;
5. Landing time – разница между временем завершения работы и временем запланированного запуска;
6. Диаграмма Ганта (для оценки протяженности выполнения тасков во времени);
7. Информация о dag’е;
8. Код dag’а;
9. Фильтры;
10. Статусы выполнения;
11. Выбранное представление.

Индивидуальное задание

Архитектура

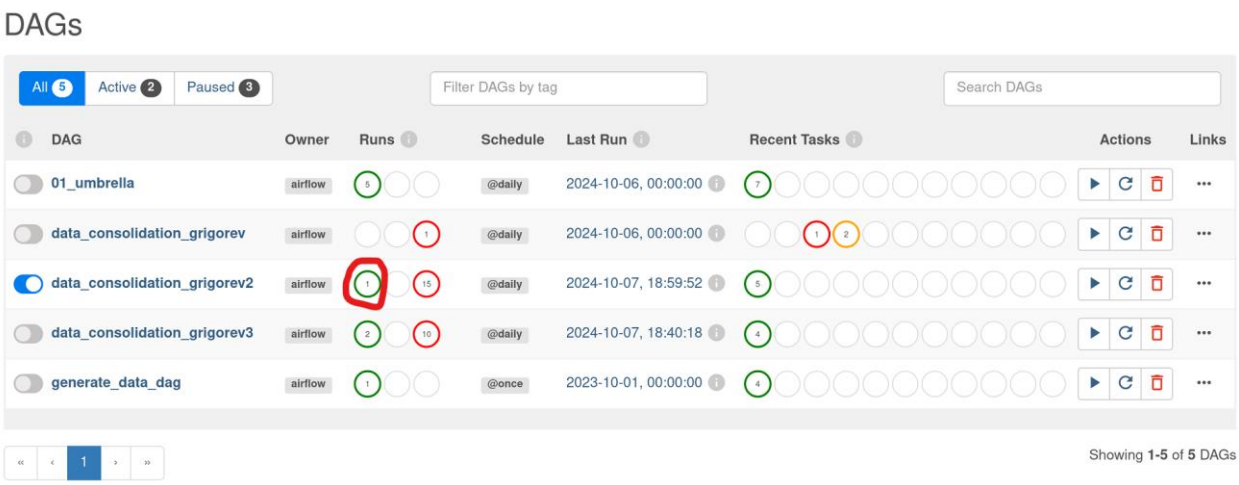


Создание DAG - код в приложении 1

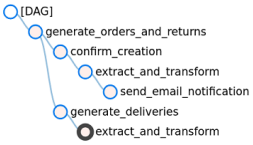
Результаты работы

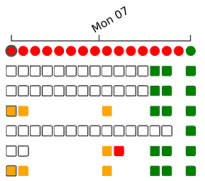
Была установлена проблема: наши даги не работали, так как в Python, который использует докер, отсутствует библиотека xlrd, следовательно, загрузить файл эксель невозможно и даг уходит на бесконечный retry. Было принято решение сгенерировать 2 .csv файла и 1 .json.

После множества итераций и тестов удалось устранить проблемы и выполнить даг:

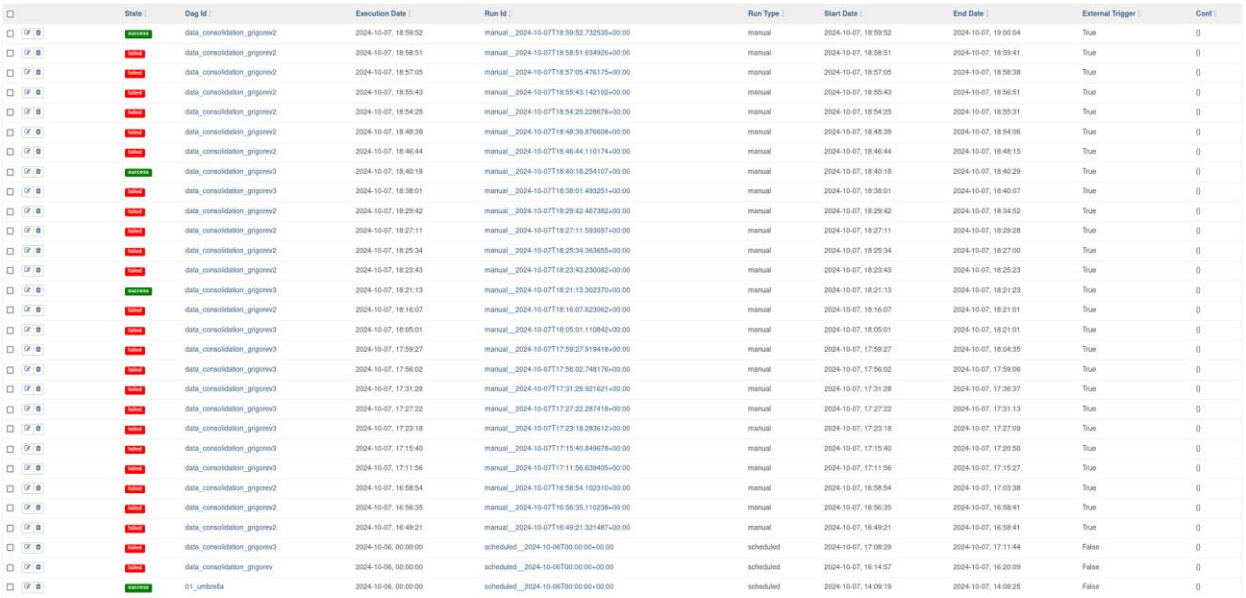


Как видно из диаграммы, во время последней попытки все таски были выполнены успешно:





Общая история запуска дагов:

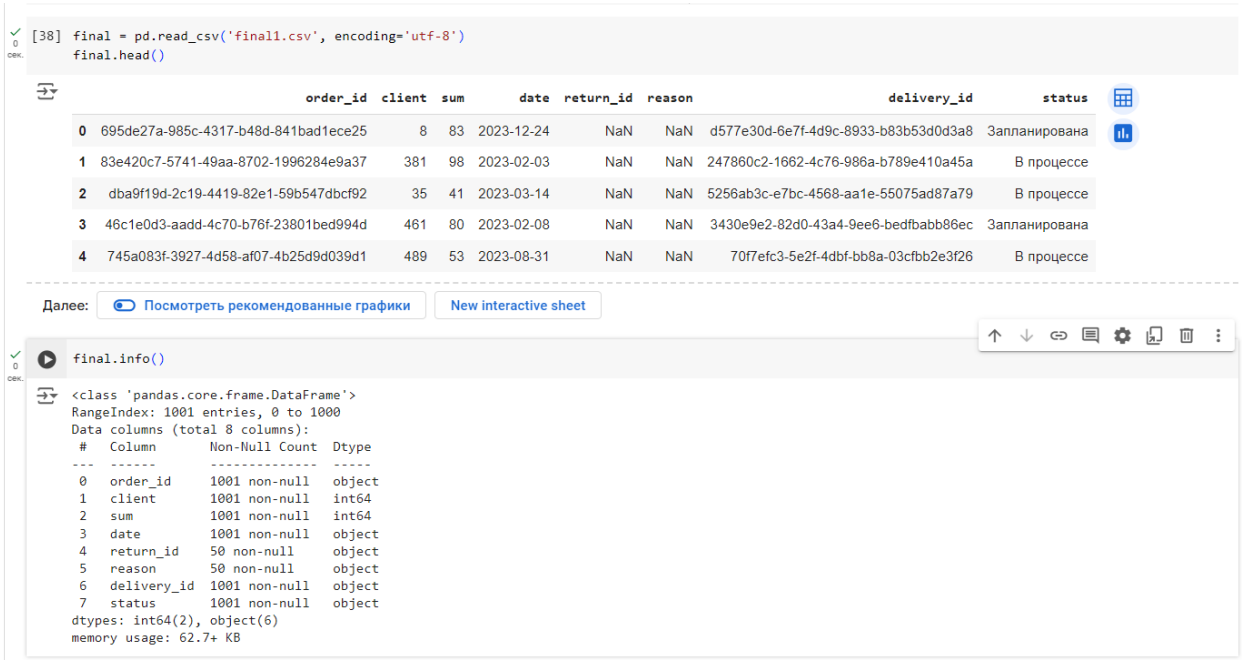


	State	Dag Id	Execution Date	Run Id	Run Type	Start Date	End Date	External Trigger	Conf
	success	data_consolidation_grigorev2	2024-10-07, 18:59:52	manual__2024-10-07T18:59:52.732535+00:00	manual	2024-10-07, 18:59:52	2024-10-07, 19:00:04	True	{}
	failed	data_consolidation_grigorev2	2024-10-07, 18:58:51	manual__2024-10-07T18:58:51.634926+00:00	manual	2024-10-07, 18:58:51	2024-10-07, 18:59:41	True	{}
	failed	data_consolidation_grigorev2	2024-10-07, 18:57:05	manual__2024-10-07T18:57:05.476175+00:00	manual	2024-10-07, 18:57:05	2024-10-07, 18:58:38	True	{}
	failed	data_consolidation_grigorev2	2024-10-07, 18:55:43	manual__2024-10-07T18:55:43.142102+00:00	manual	2024-10-07, 18:55:43	2024-10-07, 18:56:51	True	{}
	failed	data_consolidation_grigorev2	2024-10-07, 18:54:25	manual__2024-10-07T18:54:25.228676+00:00	manual	2024-10-07, 18:54:25	2024-10-07, 18:55:31	True	{}
	failed	data_consolidation_grigorev2	2024-10-07, 18:48:39	manual__2024-10-07T18:48:39.876608+00:00	manual	2024-10-07, 18:48:39	2024-10-07, 18:54:06	True	{}
	failed	data_consolidation_grigorev2	2024-10-07, 18:46:44	manual__2024-10-07T18:46:44.110174+00:00	manual	2024-10-07, 18:46:44	2024-10-07, 18:48:15	True	{}
	success	data_consolidation_grigorev3	2024-10-07, 18:40:18	manual__2024-10-07T18:40:18.254107+00:00	manual	2024-10-07, 18:40:18	2024-10-07, 18:40:29	True	{}
	failed	data_consolidation_grigorev3	2024-10-07, 18:38:01	manual__2024-10-07T18:38:01.493251+00:00	manual	2024-10-07, 18:38:01	2024-10-07, 18:40:07	True	{}
	failed	data_consolidation_grigorev2	2024-10-07, 18:29:42	manual__2024-10-07T18:29:42.467382+00:00	manual	2024-10-07, 18:29:42	2024-10-07, 18:34:52	True	{}
	failed	data_consolidation_grigorev2	2024-10-07, 18:27:11	manual__2024-10-07T18:27:11.593697+00:00	manual	2024-10-07, 18:27:11	2024-10-07, 18:29:28	True	{}
	failed	data_consolidation_grigorev2	2024-10-07, 18:25:34	manual__2024-10-07T18:25:34.363655+00:00	manual	2024-10-07, 18:25:34	2024-10-07, 18:27:00	True	{}
	failed	data_consolidation_grigorev2	2024-10-07, 18:23:43	manual__2024-10-07T18:23:43.230082+00:00	manual	2024-10-07, 18:23:43	2024-10-07, 18:25:23	True	{}
	success	data_consolidation_grigorev3	2024-10-07, 18:21:13	manual__2024-10-07T18:21:13.302370+00:00	manual	2024-10-07, 18:21:13	2024-10-07, 18:21:23	True	{}
	failed	data_consolidation_grigorev2	2024-10-07, 18:16:07	manual__2024-10-07T18:16:07.623062+00:00	manual	2024-10-07, 18:16:07	2024-10-07, 18:21:01	True	{}
	failed	data_consolidation_grigorev3	2024-10-07, 18:05:01	manual__2024-10-07T18:05:01.110842+00:00	manual	2024-10-07, 18:05:01	2024-10-07, 18:21:01	True	{}
	failed	data_consolidation_grigorev3	2024-10-07, 17:59:27	manual__2024-10-07T17:59:27.919418+00:00	manual	2024-10-07, 17:59:27	2024-10-07, 18:04:35	True	{}
	failed	data_consolidation_grigorev3	2024-10-07, 17:56:02	manual__2024-10-07T17:56:02.748176+00:00	manual	2024-10-07, 17:56:02	2024-10-07, 17:59:06	True	{}
	failed	data_consolidation_grigorev3	2024-10-07, 17:31:28	manual__2024-10-07T17:31:28.921621+00:00	manual	2024-10-07, 17:31:28	2024-10-07, 17:36:37	True	{}
	failed	data_consolidation_grigorev3	2024-10-07, 17:27:22	manual__2024-10-07T17:27:22.287418+00:00	manual	2024-10-07, 17:27:22	2024-10-07, 17:31:13	True	{}
	failed	data_consolidation_grigorev3	2024-10-07, 17:23:18	manual__2024-10-07T17:23:18.283612+00:00	manual	2024-10-07, 17:23:18	2024-10-07, 17:27:09	True	{}
	failed	data_consolidation_grigorev3	2024-10-07, 17:15:40	manual__2024-10-07T17:15:40.849678+00:00	manual	2024-10-07, 17:15:40	2024-10-07, 17:20:50	True	{}
	failed	data_consolidation_grigorev3	2024-10-07, 17:11:56	manual__2024-10-07T17:11:56.639405+00:00	manual	2024-10-07, 17:11:56	2024-10-07, 17:15:27	True	{}
	failed	data_consolidation_grigorev2	2024-10-07, 16:58:54	manual__2024-10-07T16:58:54.102310+00:00	manual	2024-10-07, 16:58:54	2024-10-07, 17:03:38	True	{}
	failed	data_consolidation_grigorev2	2024-10-07, 16:56:35	manual__2024-10-07T16:56:35.110238+00:00	manual	2024-10-07, 16:56:35	2024-10-07, 16:58:41	True	{}
	failed	data_consolidation_grigorev2	2024-10-07, 16:49:21	manual__2024-10-07T16:49:21.321487+00:00	manual	2024-10-07, 16:49:21	2024-10-07, 16:58:41	True	{}
	failed	data_consolidation_grigorev3	2024-10-06, 00:00:00	scheduled__2024-10-06T00:00:00+00:00	scheduled	2024-10-07, 17:08:29	2024-10-07, 17:11:44	False	{}
	failed	data_consolidation_grigorev	2024-10-06, 00:00:00	scheduled__2024-10-06T00:00:00+00:00	scheduled	2024-10-07, 16:14:57	2024-10-07, 16:20:09	False	{}
	success	01_umbrella	2024-10-06, 00:00:00	scheduled__2024-10-06T00:00:00+00:00	scheduled	2024-10-07, 14:09:19	2024-10-07, 14:09:25	False	{}

Скопируем файл final1.csv, полученный при выполнении дага, в файловую систему:

```
mgpu@mgpu-VirtualBox:~/Downloads$ sudo docker cp 691abd940c8b:/opt/airflow/final1.csv final1.csv
[sudo] password for mgpu:
Successfully copied 121kB to /home/mgpu/Downloads/final1.csv
```

Изучим файл с помощью pandas в соответствии с диаграммой:



```
[38] final = pd.read_csv('final1.csv', encoding='utf-8')
     final.head()
```

	order_id	client	sum	date	return_id	reason	delivery_id	status
0	695de27a-985c-4317-b48d-841bad1ece25	8	83	2023-12-24	NaN	NaN	d577e30d-6e7f-4d9c-8933-b83b53d0d3a8	Запланирована
1	83e420c7-5741-49aa-8702-1996284e9a37	381	98	2023-02-03	NaN	NaN	247860c2-1662-4c76-986a-b789e410a45a	В процессе
2	dba9f19d-2c19-4419-82e1-59b547dbcf92	35	41	2023-03-14	NaN	NaN	5256ab3c-e7bc-4568-aa1e-55075ad87a79	В процессе
3	46c1e0d3-aadd-4c70-b76f-23801bed994d	461	80	2023-02-08	NaN	NaN	3430e9e2-82d0-43a4-9ee6-bedfbabb86ec	Запланирована
4	745a083f-3927-4d58-af07-4b25d9d039d1	489	53	2023-08-31	NaN	NaN	70f7efc3-5e2f-4dbf-bb8a-03cfbb2e3f26	В процессе

Далее: Посмотреть рекомендованные графики | New interactive sheet

```
final.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1001 entries, 0 to 1000
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   order_id     1001 non-null   object
 1   client       1001 non-null   int64 
 2   sum          1001 non-null   int64 
 3   date         1001 non-null   object
 4   return_id    50 non-null     object
 5   reason       50 non-null     object
 6   delivery_id  1001 non-null   object
 7   status       1001 non-null   object
dtypes: int64(2), object(6)
memory usage: 62.7+ KB
```

Приложение 1 – код DAG

```
import airflow.utils.dates
from airflow import DAG
from airflow.operators.dummy import DummyOperator
from airflow.operators.python_operator import PythonOperator
from airflow.operators.email_operator import EmailOperator
from datetime import datetime, timedelta
import pandas as pd
import sqlite3
import os
import numpy as np
import json
import random
import uuid
```

```
default_args = {
    'owner': 'airflow',
    'depends_on_past': False,
    'start_date': datetime(2024, 10, 6),
    'email_on_failure': False,
    'email_on_retry': False,
    'retries': 1,
 }
dag = DAG(
    'data_consolidation_grigorev2',
    default_args=default_args,
    description='Consolidate data from CSV, Excel, JSON and save to SQLite',
    schedule_interval='@daily',
 )

def generate_orders_and_returns():
    def random_date(start, end):
        return start + timedelta(
         seconds=random.randint(0, int((end - start).total_seconds()))
        )
    start_order = datetime(2023, 1, 1)
    end_order = datetime(2023, 12, 31)
    orders = []

    for i in range(1000):
        order = {
            'order_id': str(uuid.uuid4()),
            'client': random.randint(1, 500),
            'sum': random.randint(5, 100),
            'date': random_date(start_order, end_order).strftime('%Y-%m-%d')
```

```
        }
        orders.append(order)
    orders_df = pd.DataFrame(orders)
    orders_df.to_csv(f'orders1.csv', index=False)

    returns = []
    reason= ['Брак', 'Долгая доставка', 'Нашли дешевле', 'Иная причина']
    for a in range(50):
        order = random.choice(orders)
        returno = {
            'return_id': str(uuid.uuid4()),
            'order_id': order['order_id'],
            'reason': random.choice(reason)
        }
        returns.append(returno)
    returns_df = pd.DataFrame(returns)
    returns_df.to_csv(f'returns1.csv', index=False)

def generate_deliveries():
    deliveries = []
    status = ['Запланирована', 'В процессе', 'Выполнена']
    orders_df = pd.read_csv(f'orders1.csv')
    orders_list = orders_df['order_id'].tolist()
    for b in orders_list:
        delivery = {
            'delivery_id': str(uuid.uuid4()),
            'order_id': b,
            'status': random.choice(status)
        }
        deliveries.append(delivery)
```

```
    with open(f'deliveries1.json', 'w', encoding='utf-8') as f:
        json.dump(deliveries, f, ensure_ascii=False, indent=4)

def extract_and_transform():
 # Пути к файлам данных
    csv_file = (f'orders1.csv')
    excel_file = (f'returns1.csv')
    json_file = (f'deliveries1.json')

 # Загрузка данных
    csv_data = pd.read_csv(csv_file)
    excel_data = pd.read_csv(excel_file)
    json_data = pd.read_json(json_file)
# Merging
    merged_data1 = pd.merge(csv_data, excel_data, on='order_id', how='left')
    merged_data2 = pd.merge(merged_data1, json_data, on='order_id', how='left')
    merged_data2.to_csv('final1.csv', index=False)

def send_email_notification():
 # Псевдокод для отправки email
    print("Отправить email уведомление")

def confirm_creation():
    print("Base created")



confirm_creation_task = PythonOperator(
    task_id='confirm_creation',
    python_callable=confirm_creation,
```

```
    dag=dag
)

generate_orders_and_returns_task = PythonOperator(
    task_id='generate_orders_and_returns',
    python_callable=generate_orders_and_returns,
    dag=dag,
 )


generate_deliveries_task = PythonOperator(
    task_id='generate_deliveries',
    python_callable=generate_deliveries,
    dag=dag,
 )

extract_transform_task = PythonOperator(
    task_id='extract_and_transform',
    python_callable=extract_and_transform,
    dag=dag,
 )

send_email_notification_task = PythonOperator(
    task_id='send_email_notification',
    python_callable=send_email_notification,
    dag=dag
)

generate_orders_and_returns_task >> generate_deliveries_task
generate_orders_and_returns_task >> confirm_creation_task
```

[confirm_creation_task, generate_deliveries_task] >> extract_transform_task >> send_email_notification_task