Лабораторная работа 3. Программные средства консолидации данных из различных источников с использованием Python и Apache Airflow

Цель: научиться работать с Apache Airflow для автоматизации процессов ETL (Extract, Transform, Load). На практике освоить настройку и выполнение DAG в Airflow для извлечения данных из различных форматов (CSV, Excel, JSON), их обработки на Python, загрузки в базу данных SQLite и отправки уведомлений по электронной почте.

Оборудование и ПО:

- Ubuntu (с Docker)
- Apache Airflow
- SQLite
- Python
- Docker
- email сервис (например, Gmail или локальный SMTP-сервер)

Исходные данные:

- Набор файлов CSV, Excel и JSON, содержащих данные для обработки.
- Конфигурация email для отправки уведомлений.

Пример: создать DAG для извлечения данных из нескольких источников (CSV, Excel, JSON), их трансформации и сохранения в базу данных SQLite с отправкой уведомления через email.

Ход работы

Шаг 1: Установка Docker и настройка окружения.

1. Обновление пакетов.

```bash
sudo apt update && sudo apt upgrade -y
```

2. Установка Docker.

```bash
sudo apt install docker.io -y
sudo systemctl start docker
sudo systemctl enable docker
```

3. Установка Docker Compose.

```bash
sudo apt install docker-compose -y
```

4. Создание рабочей директории для проекта:

```bash
mkdir airflow_lab && cd airflow_lab
```

Шаг 2: Настройка Apache Airflow через Docker.

1. Создание файла **docker-compose.yaml** для Airflow.

Создайте файл с содержимым:

```yaml
version: '3'
services:
  postgres:
    image: postgres:13
    environment:
      POSTGRES_USER: airflow
      POSTGRES_PASSWORD: airflow
      POSTGRES_DB: airflow
    volumes:
      - ./pg/var/lib/postgresql/data

  webserver:
    image: apache/airflow:2.5.0
    restart: always
    environment:
      AIRFLOW__CORE__EXECUTOR: LocalExecutor
      AIRFLOW__CORE__SQL_ALCHEMY_CONN:
postgresql+psycopg2://airflow:airflow@postgres/airflow
      AIRFLOW__CORE__LOAD_EXAMPLES: 'false'
      AIRFLOW__WEBSERVER__SECRET_KEY: 'your_secret_key'
    volumes:
      - ./dags:/opt/airflow/dags
      - ./logs:/opt/airflow/logs
      - ./plugins:/opt/airflow/plugins
    ports:
      - "8080:8080"
    depends_on:
      - postgres
    command: webserver

  scheduler:
```

```
    image: apache/airflow:2.5.0
    restart: always
    volumes:
      - ./dags:/opt/airflow/dags
      - ./logs:/opt/airflow/logs
      - ./plugins:/opt/airflow/plugins
    environment:
      AIRFLOW__CORE__EXECUTOR: LocalExecutor
      AIRFLOW__CORE__SQL_ALCHEMY_CONN:
postgresql+psycopg2://airflow:airflow@postgres/airflow
    depends_on:
      - postgres
    command: scheduler
```

2. Запуск контейнеров.

```bash
docker-compose up -d
```

3. Инициализация базы данных Airflow.

```bash
docker-compose run webserver airflow db init
```

4. Создание пользователя для веб-интерфейса Airflow.

```bash
docker-compose run webserver airflow users create \
    --username admin \
    --password admin \
    --firstname Admin \
    --lastname Admin \
    --role Admin \
    --email admin@example.com
```

5. Проверка запуска.

Откройте браузер и перейдите по адресу `http://localhost:8080` для доступа к интерфейсу Airflow.

Шаг 3: Подготовка DAG для консолидации данных.

1. Создание папки для DAG.

```bash
mkdir -p dags
```

2. Создание файла DAG.

В папке **dags** создайте файл **data_consolidation_dag.py**.

Пример содержимого:

```python
from airflow import DAG
from airflow.operators.python_operator import PythonOperator
from airflow.operators.email_operator import EmailOperator
from datetime import datetime
import pandas as pd
import sqlite3
import os

default_args = {
    'owner': 'airflow',
    'depends_on_past': False,
    'start_date': datetime(2024, 9, 1),
    'email_on_failure': False,
    'email_on_retry': False,
    'retries': 1,
}

dag = DAG(
    'data_consolidation',
    default_args=default_args,
    description='Consolidate data from CSV, Excel, JSON and save to SQLite',
    schedule_interval='@daily',
)

def extract_and_transform():
    # Пути к файлам данных
    csv_file = '/opt/airflow/dags/data/sample_data.csv'
    excel_file = '/opt/airflow/dags/data/sample_data.xlsx'
    json_file = '/opt/airflow/dags/data/sample_data.json'
```

```
    # Загрузка данных
    csv_data = pd.read_csv(csv_file)
    excel_data = pd.read_excel(excel_file)
    json_data = pd.read_json(json_file)

    # Пример агрегирования данных
    merged_data = pd.concat([csv_data, excel_data, json_data],
ignore_index=True)

    # Преобразования (например, группировка по колонкам)
    aggregated_data = merged_data.groupby('category').sum()

    # Сохранение в SQLite
    conn = sqlite3.connect('/opt/airflow/dags/data/airflow_data.db')
    aggregated_data.to_sql('aggregated_data', conn, if_exists='replace',
index=False)
    conn.close()

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

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

email_task = EmailOperator(
    task_id='send_email',
    to='your_email@example.com',
    subject='Data Consolidation Completed',
    html_content='<p>The data consolidation task has been completed
successfully.</p>',
    dag=dag,
)

extract_transform_task >> email_task
```

Шаг 4: Подготовка данных и тестирование.

1. Создание папки для данных.

```bash
mkdir dags/data
```

2. Размещение файлов данных (CSV, Excel, JSON) в папке **dags/data/**.

- sample_data.csv
- sample_data.xlsx
- sample_data.json

3. Запуск DAG.

В веб-интерфейсе Airflow активируйте DAG **data_consolidation** и дождитесь его выполнения.

4. Проверка результатов.

- Проверьте базу данных SQLite, чтобы убедиться, что данные успешно загружены и агрегированы.
- Проверьте почту для получения уведомления об успешном завершении задачи.

Чтобы проверить, что данные успешно загружены и агрегированы в базу данных SQLite, воспользоваться утилитой командной строки **sqlite3**, которая уже установлена в большинстве систем Ubuntu.

Шаг 4.1: Установка sqlite3 (если не установлено).

Если sqlite3 не установлено, выполните следующую команду:

```bash
sudo apt install sqlite3
```

Шаг 4.2: Подключение к базе данных.

В командной строке перейдите в папку, где находится база данных (в нашем случае, это /opt/airflow/dags/data/) и подключитесь к базе данных **airflow_data.db**:

```bash
cd /opt/airflow/dags/data/
sqlite3 airflow_data.db
```

Шаг 4.3: Просмотр списка таблиц.

Чтобы убедиться, что таблица была создана, выполните команду:

```sql
.tables
```

Должны увидеть таблицу **aggregated_data**.

Шаг 4.4: Просмотр данных из таблицы

Чтобы просмотреть содержимое таблицы и проверить, что данные были загружены и агрегированы, выполните команду:

```sql
SELECT * FROM aggregated_data;
```

Эта команда покажет все данные, которые были сохранены в таблицу. Вы увидите строки, содержащие данные после объединения и агрегации из файлов CSV, Excel и JSON.

Шаг 4.5: Дополнительные проверки

- Просмотр нескольких строк:

```sql
SELECT * FROM aggregated_data LIMIT 10;
```

- Если нужно посмотреть структуру таблицы (какие столбцы существуют), выполните:

```sql
PRAGMA table_info(aggregated_data);
```

Шаг 4.6: Завершение работы

Чтобы выйти из утилиты `sqlite3`, введите:

```sql
.exit
```

Шаг 5: Завершение работы.

1. Остановить и удалить контейнеры.

```bash
docker-compose down
```

2. Очистить ресурсы.

```bash
sudo rm -rf dags/data
sudo rm -rf pgdata
```

Для всех вариантов студенты должны:

- Использовать Apache Airflow для автоматизации ETL процесса.
- Спланировать и создать DAG, который читает данные из файлов (CSV, Excel, JSON).
- Консолидировать данные с использованием Python и библиотеки pandas.
- Выполнить аналитические расчёты и представить результаты в виде отчёта или графиков.

Варианты заданий

Вариант 1.

1. Файл CSV: данные о продажах в розничных магазинах (магазин, товар, количество проданных единиц).
2. Файл Excel: данные о ценах на товары (товар, цена).
3. Файл JSON: данные о скидках на товары (товар, скидка).

Задача: настроить DAG для объединения данных и рассчитать итоговую выручку по каждому магазину с учётом скидок.

Вариант 2.

1. Файл CSV: список студентов (имя, факультет, курс).
2. Файл Excel: данные о посещаемости занятий (имя студента, дата, предмет).
3. Файл JSON: данные о рейтингах преподавателей по предметам.

Задача: с помощью Apache Airflow автоматизировать процесс объединения данных и рассчитать среднюю посещаемость по каждому предмету и её корреляцию с рейтингом преподавателя.

Вариант 3.

1. Файл CSV: данные о заказах в интернет-магазине (номер заказа, клиент, сумма).
2. Файл Excel: данные о возвратах товаров (номер заказа, причина возврата).
3. Файл JSON: данные о доставке (номер заказа, статус доставки).

Задача: настроить DAG в Airflow для обработки и объединения данных, после чего рассчитать процент возвратов и выявить причины возвратов по каждому статусу доставки.

Вариант 4.

1. Файл CSV: данные о клиентах (имя, возраст, доход).
2. Файл Excel: данные о покупках клиентов (имя, дата, товар).
3. Файл JSON: данные о промоакциях (товар, скидка, дата).

Задача: автоматизировать сбор данных через Apache Airflow и рассчитать, как часто клиенты с разным уровнем дохода участвуют в промоакциях.

Вариант 5.

1. Файл CSV: данные о сотрудниках компании (имя, отдел, зарплата).
2. Файл Excel: данные о проектах (проект, сотрудник, часы работы).

3. Файл JSON: данные о ставках оплаты труда в зависимости от должности.

Задача: настроить DAG для консолидации данных о часах работы сотрудников по проектам и расчёта итоговой зарплаты с учётом ставок.

Вариант 6.

1. Файл CSV: список продуктов (артикул, категория, количество на складе).
2. Файл Excel: данные о продажах (артикул, дата, количество проданных единиц).
3. Файл JSON: прогноз спроса на продукты по категориям.

Задача: автоматизировать объединение данных через Airflow и спрогнозировать остатки товаров на складе с учётом продаж и прогноза спроса.

Вариант 7.

1. Файл CSV: список клиентов (имя, город, сумма покупок).
2. Файл Excel: данные о программах лояльности (город, программа, скидка).
3. Файл JSON: данные о транзакциях клиентов по программам лояльности.

Задача: используя Airflow, объединить данные и рассчитать среднюю сумму покупок клиентов по каждому городу с учётом программ лояльности.

Вариант 8.

1. Файл CSV: данные о филиалах компании (город, количество сотрудников).
2. Файл Excel: данные о расходах филиалов (город, месяц, расходы).
3. Файл JSON: данные о планируемом бюджете на следующий год для каждого филиала.

Задача: настроить DAG в Airflow для автоматического сбора данных, сравнения планируемого и фактического бюджета по филиалам, и выявления филиалов с перерасходом средств.

Вариант 9.

1. Файл CSV: данные о товарах (артикул, категория, цена).
2. Файл Excel: данные о заказах (артикул товара, количество заказов, дата).
3. Файл JSON: данные о скидках на товары.

Задача: с помощью Airflow настроить конвейер для объединения данных, после чего рассчитать общую прибыль по каждому товару с учётом скидок.

Вариант 10.

1. Файл CSV: данные о спортсменах (имя, вид спорта, возраст).
2. Файл Excel: результаты соревнований (спортсмен, соревнование, результат).
3. Файл JSON: данные о тренерах и их спортсменах.

Задача: настроить DAG для объединения данных о результатах соревнований и спортсменах, после чего рассчитать средние результаты по каждому тренеру.

Задание 11.

Разработать DAG в Apache Airflow для извлечения данных из файлов форматов CSV, Excel и JSON, объединения данных в один Pandas DataFrame и сохранения результата в базу данных SQLite.

Задание 12.

Создать Airflow DAG, который выполняет ежедневную проверку обновлений данных в папке с файлами CSV, Excel и JSON. При изменении данных нужно автоматически консолидацию их в единый CSV файл и загружать в Google BigQuery.

Задание 13.

Реализовать DAG, который последовательно извлекает данные из Excel и JSON файлов, конвертирует их в CSV формат, а затем объединяет в один DataFrame с последующей выгрузкой в таблицу MySQL.

Задание 14.

Сконфигурировать Airflow для автоматической обработки файлов Excel, CSV и JSON по расписанию. Необходимо агрегировать данные в разрезе выбранных столбцов и формировать отчёт в формате Excel.

Задание 15

Разработать решение в Airflow для регулярного объединения файлов из различных источников (CSV, Excel, JSON), преобразования данных с помощью Python (очистка, фильтрация), и последующего сохранения в Parquet.

Задание 16

Создать DAG, который извлекает данные из CSV, Excel и JSON файлов, выполняет базовые операции по очистке (удаление дубликатов, обработка пропущенных значений) и сохраняет результат в Google Drive в формате Excel.

Задание 17.

Реализовать DAG для автоматизированной загрузки данных из файлов CSV, Excel и JSON в хранилище данных Amazon S3, с последующим объединением всех данных в единый JSON файл.

Задание 18.

Создать Python скрипт и DAG в Apache Airflow, который собирает данные из нескольких CSV, Excel и JSON файлов, анализирует данные и генерирует сводный отчёт (например, средние значения, медианы, и т.д.), сохраняемый в формате Excel.

Задание 19.

Реализовать DAG, который на регулярной основе объединяет данные из файлов CSV, Excel и JSON, проверяет консистентность данных (например, совпадение ключевых полей), и загружает очищенные данные в PostgreSQL базу данных.

Задание 20.

Настроить DAG в Airflow, который извлекает данные из нескольких источников (CSV, Excel и JSON), выполняет фильтрацию на основе определённых условий (например, даты или значения), и выгружает результат в MongoDB.

Задание 21.

Разработать DAG, который на регулярной основе обрабатывает данные из разных форматов (CSV, Excel, JSON), анализирует их и генерирует ежедневный отчёт в формате Excel с графиками и статистическими данными.

Задание 22.

Создать решение с использованием Apache Airflow, которое собирает данные из разных файловых форматов (CSV, Excel, JSON), применяет трансформации (например, объединение и нормализацию данных) и сохраняет результат в формате XML.

Задание 23.

Настроить DAG для периодической загрузки данных из CSV, Excel и JSON файлов в базу данных, объединения данных в единый DataFrame и последующей визуализации данных в виде диаграмм, которые сохраняются в виде изображений.

Задание 24.

Разработать DAG для автоматизированного создания сводных отчётов на основе данных из CSV, Excel и JSON файлов. Отчёт должен включать сводную таблицу и быть сохранён в Google Sheets.

Задание 25.

Создать DAG, который извлекает данные из нескольких CSV, Excel и JSON источников, использует Python для их трансформации (например, агрегирования данных по заданным полям), и сохраняет результат в базу данных SQLite с последующей отправкой уведомления через email.