ETL. Семинар №4

1. Викторина
2. Блок 1

Задание 1

Создайте новый пайплайн. Скачайте файл ССЫЛКА. Сделайте пайплайн который будет читать файл, выполнять удаление дубликатов и загружать очищенные данные в базу данных postgress после этого отправлять письмо с количеством записей добавленных в базу данных,

40 минут

Решение:

```python
import datetime
import pendulum
import os

import requests
from airflow.decorators import dag, task
from airflow.providers.postgres.hooks.postgres import PostgresHook
from airflow.providers.postgres.operators.postgres import PostgresOperator


@dag(
   dag_id="process-employees",
   schedule_interval="0 0 * * *",
   start_date=pendulum.datetime(2021, 1, 1, tz="UTC"),
   catchup=False,
   dagrun_timeout=datetime.timedelta(minutes=60),
)
def ProcessEmployees():
   create_employees_table = PostgresOperator(
      task_id="create_employees_table",
      postgres_conn_id="pg_conn",
      sql="""
         CREATE TABLE IF NOT EXISTS employees (
            "Serial Number" NUMERIC PRIMARY KEY,
            "Company Name" TEXT,
            "Employee Markme" TEXT,
            "Description" TEXT,
            "Leave" INTEGER
         );""",
   )
```

```
create_employees_temp_table = PostgresOperator(
    task_id="create_employees_temp_table",
    postgres_conn_id="pg_conn",
    sql="""
        DROP TABLE IF EXISTS employees_temp;
        CREATE TABLE employees_temp (
            "Serial Number" NUMERIC PRIMARY KEY,
            "Company Name" TEXT,
            "Employee Markme" TEXT,
            "Description" TEXT,
            "Leave" INTEGER
        );""",
)

@task
def get_data():
    # NOTE: configure this as appropriate for your airflow environment
    data_path = PATH_TO_FOLDER"
    os.makedirs(os.path.dirname(data_path), exist_ok=True)

    url = "https://raw.githubusercontent.com/apache/airflow/main/docs/apache-airflow/tutorial/pipeline_example.csv"

    response = requests.request("GET", url)

    with open(data_path, "w") as file:
        file.write(response.text)

    postgres_hook = PostgresHook(postgres_conn_id="pg_conn")
    conn = postgres_hook.get_conn()
    cur = conn.cursor()
    with open(data_path, "r") as file:
        cur.copy_expert(
            "COPY employees_temp FROM STDIN WITH CSV HEADER DELIMITER AS ',' QUOTE '\"'",
            file,
        )
    conn.commit()

@task
def merge_data():
    query = """
        INSERT INTO employees
        SELECT *
        FROM (
            SELECT DISTINCT *
            FROM employees_temp
        )
        ON CONFLICT ("Serial Number") DO UPDATE
        SET "Serial Number" = excluded."Serial Number";
    """
    try:
        postgres_hook = PostgresHook(postgres_conn_id="pg_conn")
        conn = postgres_hook.get_conn()
        cur = conn.cursor()
        cur.execute(query)
        conn.commit()
        return 0
    except Exception as e:
        return 1

[create_employees_table, create_employees_temp_table] >> get_data() >> merge_data()
```

```
dag = ProcessEmployees()
```

URL – ссылка на файл

Задание 2

Скачайте файл ССЫЛКА. Измените пайплайн так чтобы он читал файл, выполнять удаление дубликатов и загружать недостающие данные в базу данных postgress после этого отправлять письмо с количеством записей добавленных в базу данных, но только в случае если данные были добавлены.

30 минут

Решение:

Инструкция по предварительной настройке:
https://www.projectpro.io/recipes/use-emailoperator-airflow-dag

```
import datetime
import pendulum
import os

import requests
from airflow.decorators import dag, task, XComArg
from airflow.providers.postgres.hooks.postgres import PostgresHook
from airflow.providers.postgres.operators.postgres import PostgresOperator
from airflow.operators.email import EmailOperator


@dag(
    dag_id="process-employees",
    schedule_interval="0 0 * * *",
    start_date=pendulum.datetime(2021, 1, 1, tz="UTC"),
    catchup=False,
    dagrun_timeout=datetime.timedelta(minutes=60),
)
def ProcessEmployees():
    create_employees_table = PostgresOperator(
        task_id="create_employees_table",
        postgres_conn_id="pg_conn",
        sql="""
        CREATE TABLE IF NOT EXISTS employees (
            "Serial Number" NUMERIC PRIMARY KEY,
            "Company Name" TEXT,
            "Employee Markme" TEXT,
            "Description" TEXT,
            "Leave" INTEGER
        );""",
)

create_employees_temp_table = PostgresOperator(
    task_id="create_employees_temp_table",
    postgres_conn_id="pg_conn",
    sql="""
        DROP TABLE IF EXISTS employees_temp;
        CREATE TABLE employees_temp (
            "Serial Number" NUMERIC PRIMARY KEY,
            "Company Name" TEXT,
            "Employee Markme" TEXT,
            "Description" TEXT,
            "Leave" INTEGER
        );""",
)

@task
def get_data():
    # NOTE: configure this as appropriate for your airflow environment
    data_path = "/Users/i344537/airflow/dags/files/employees.csv"
    os.makedirs(os.path.dirname(data_path), exist_ok=True)

    url = "https://raw.githubusercontent.com/apache/airflow/main/docs/apache-airflow/tutorial/pipeline_example.csv"

    response = requests.request("GET", url)

    with open(data_path, "w") as file:
        file.write(response.text)

    postgres_hook = PostgresHook(postgres_conn_id="pg_conn")
    conn = postgres_hook.get_conn()
    cur = conn.cursor()
    with open(data_path, "r") as file:
        cur.copy_expert(
            "COPY employees_temp FROM STDIN WITH CSV HEADER DELIMITER AS ',' QUOTE '\"'",
            file,
        )
    conn.commit()

@task
def merge_data(ti=None):
    count_q = """
        SELECT COUNT(*)
        FROM employees
    """
    query = """
        INSERT INTO employees
        SELECT *
        FROM (
            SELECT DISTINCT *
            FROM employees_temp
        )
        ON CONFLICT ("Serial Number") DO UPDATE
```

```
            SET "Serial Number" = excluded."Serial Number";
        """
        try:
            postgres_hook = PostgresHook(postgres_conn_id="pg_conn")
            conn = postgres_hook.get_conn()
            cur = conn.cursor()
            res_before = int(cur.execute(count_q))
            cur.execute(query)
            conn.commit()
            res_after = int(cur.execute(count_q))
            if res_after - res_before > 0:
                ti.xcom_push(key="Number of added strings", value=res_after - res_before)
            else: ti.xcom_push(key="Number of added strings", value=0)
            return 0
        except Exception as e:
            return 1

    def send_email(ti=None):
        number_of_added_rows = ti.xcom_pull(task_ids='merge_data')
        if number_of_added_rows > 0:
            email = 'example@email.com'
            msg = f"Added {number_of_added_rows} rows"
            subject = "Added rows"
            EmailOperator(to=email, subject=subject, html_content=msg)

    [create_employees_table, create_employees_temp_table] >> get_data() >> merge_data() >> send_email()


dag = ProcessEmployees()
```