### Урок 8. Специфика применения ETL в различных предметных сферах

1. Используя материалы семинара и s8dag.py нужно доработать задачу в части записи данных в mysq по погоде яндекса и open weather (поля - метка текущего времени и температура).

2. Создать еще одну задачу по отправке данных в телеграм. За основу взять данные таблиц платежей из 4-го семинара (все 360 периодов), конвертировать их в текстовый формат и отправить их в telegram.

3. Рассмотрите возможность применения разметки html либо markdown. Нужно выслать одну основную таблицу. Есть есть лимит по сообщениям, можно ограничить количество строк таблицы. Можете использовать функцию limit в sql запросе.

4. К ДЗ приложите код и скриншоты отрабоданных задач аирфлоу, а также отправленный слепкок из базы данных в вашем чаботе.

import datetime

import os

import requests

import pendulum

from airflow.decorators import dag, task

from airflow.providers.telegram.operators.telegram import TelegramOperator

from sqlalchemy import create\_engine

os.environ["no\_proxy"]="\*"

@dag(

    dag\_id="wether-tlegram-sql",

    schedule="@once",

    start\_date=pendulum.datetime(2024, 4, 30, tz="UTC"),

    catchup=False,

    dagrun\_timeout=datetime.timedelta(minutes=60),

)

def WetherETL():

    send\_message\_telegram\_task = TelegramOperator(

        task\_id='send\_message\_telegram',

        telegram\_conn\_id='telegram\_default',

        token='6875707033:AAG2TaDKrrLUwlUmcX9LIVA1uCm6S43pya0',

        chat\_id='547504860',

        text='Wether in Tyumen \nYandex: ' + "{{ ti.xcom\_pull(task\_ids=['yandex\_wether'],key='wether')[0]['temperature']}}" + " degrees at " + "{{ ti.xcom\_pull(task\_ids=['yandex\_wether'],key='wether')[0]['datetime']}}" +

    "\nOpen wether: " + "{{ ti.xcom\_pull(task\_ids=['open\_wether'],key='open\_wether')[0]['temperature']}}" + " degrees at " + "{{ ti.xcom\_pull(task\_ids=['open\_wether'],key='open\_wether')[0]['datetime']}}",

    )

    @task(task\_id='yandex\_wether')

    def get\_yandex\_wether(\*\*kwargs):

        ti = kwargs['ti']

        url = "https://api.weather.yandex.ru/v2/informers/?lat=57.152985&lon=65.527168"

        payload={}

        headers = {

        'X-Yandex-API-Key': '33f45b91-bcd4-46e4-adc2-33cfdbbdd88e'

        }

        response = requests.request("GET", url, headers=headers, data=payload)

        print("test")

        temperature = response.json()['fact']['temp']

        current\_datetime = datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')

        a = response.json()['fact']['temp']

        print(a)

        ti.xcom\_push(key='wether', value={'temperature': temperature, 'datetime': current\_datetime})

    @task(task\_id='open\_wether')

    def get\_open\_wether(\*\*kwargs):

        ti = kwargs['ti']

        url = "https://api.openweathermap.org/data/2.5/weather?lat=57.152985&lon=65.527168&appid=2cd78e55c423fc81cebc1487134a6300"

        payload={}

        headers = {}

        response = requests.request("GET", url, headers=headers, data=payload)

        print("test")

        temperature = round(float(response.json()['main']['temp']) - 273.15, 2)

        current\_datetime = datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')

        a = round(float(response.json()['main']['temp']) - 273.15, 2)

        print(a)

        ti.xcom\_push(key='open\_wether', value={'temperature': temperature, 'datetime': current\_datetime})

    @task(task\_id='save\_weather')

    def get\_save\_weather(\*\*kwargs):

        yandex\_data = kwargs['ti'].xcom\_pull(task\_ids='yandex\_wether', key='wether')

        open\_weather\_data = kwargs['ti'].xcom\_pull(task\_ids='open\_wether', key='open\_wether')

        temperature\_yandex = yandex\_data['temperature']

        datetime\_yandex = yandex\_data['datetime']

        service\_yandex = 'Yandex'

        temperature\_open\_weather = open\_weather\_data['temperature']

        datetime\_open\_weather = open\_weather\_data['datetime']

        service\_open\_weather = 'OpenWeather'

        engine = create\_engine("mysql://root:1@localhost:33061/spark")

        with engine.connect() as connection:

            connection.execute("""DROP TABLE IF EXISTS spark.`Temperature\_Weather`""")

            connection.execute("""CREATE TABLE IF NOT EXISTS spark.`Temperature\_Weather` (

                Service VARCHAR(255),

                Date\_time TIMESTAMP,

                City VARCHAR(255),

                Temperature FLOAT,

                PRIMARY KEY (Date\_time, Service)

            )COLLATE='utf8mb4\_general\_ci' ENGINE=InnoDB""")

            connection.execute(f"""INSERT INTO spark.`Temperature\_Weather` (Date\_time, City, Temperature, Service) VALUES ('{datetime\_yandex}', 'Tyumen', {temperature\_yandex}, '{service\_yandex}')""")

            connection.execute(f"""INSERT INTO spark.`Temperature\_Weather` (Date\_time, City, Temperature, Service) VALUES ('{datetime\_open\_weather}', 'Tyumen', {temperature\_open\_weather}, '{service\_open\_weather}')""")

    @task(task\_id='python\_wether')

    def get\_wether(\*\*kwargs):

        print("Yandex "+str(kwargs['ti'].xcom\_pull(task\_ids=['yandex\_wether'],key='wether')[0])+" Open "+str(kwargs['ti'].xcom\_pull(task\_ids=['open\_wether'],key='open\_wether')[0]))

    yandex\_wether = get\_yandex\_wether()

    open\_wether = get\_open\_wether()

    python\_wether = get\_wether()

    save\_weather = get\_save\_weather()

    yandex\_wether >> open\_wether >> python\_wether >> send\_message\_telegram\_task >> save\_weather

dag = WetherETL()





