**Task 6. Snowflake**

*Для регистрации в Snowflake нужно использовать вайфай Варшавы или Вильнюса*

**Цель:** познакомиться ближе с таким популярным и мощным DWH как Snowflake.

**Задача:** построить ELT pipeline на базе Snowflake с использованием Airflow (local). Общий дизайн архитектуры должен выглядеть таким образом:

Diagram

Description automatically generated with medium confidence

Техническое описание:

1. Парсим [файлик](https://drive.google.com/file/d/1_p4WUcLGu2drvb3IJPbRcf3gkWlCnrkZ/view?usp=sharing), избавляясь от индексов (иначе Snowflake не прочитает его)
2. Создаем два [data streams](https://docs.snowflake.com/en/user-guide/streams-intro.html) в Snowflake и настраиваем их на две таблицы (RAW\_TABLE, STAGE\_TABLE).
3. Записываем данные из CSV в RAW\_TABLE
4. Записываем данные из RAW\_STREAM в STAGE\_TABLE
5. Записываем данные из STAGE\_STREAM в MASTER\_TABLE

#Install python virtual environment

pip3 install virtualenv

# Create virtual environment

virtualenv -p python3 airflow\_venv

# Activate virtual environment.

source airflow\_venv/bin/activate

# Install apache airflow

pip3 install apache-airflow

export AIRFLOW\_HOME=/Users/myadav/Downloads/airflow\_mohit

# Initialize the airflow db.

airflow db init

# Run airflow scheduler.

airflow scheduler

# Create a default user with admin,run airflow webserver.

airflow users  create --role Admin --username admin --email admin --firstname admin --lastname admin --password admin

airflow webserver

# open the webserver UI on localhost:8080

# install few python packages for snowflake integration with airflow.

pip3 install snowflake-connector-python

pip3 install snowflake-sqlalchemy

pip3 install apache-airflow-providers-snowflake

# create the DAG

import logging

from datetime import datetime, timedelta

import airflow

from airflow import DAG

from airflow.operators.python\_operator import PythonOperator

from airflow.contrib.hooks.snowflake\_hook import SnowflakeHook

from airflow.contrib.operators.snowflake\_operator import SnowflakeOperator

logging.basicConfig(level=[logging.INFO](http://logging.info/))

logger = logging.getLogger(\_\_name\_\_)

args = {"owner": "Airflow", "start\_date": datetime(2023,1,5,12,15)}

dag = DAG(

dag\_id="snowflake\_connector3", default\_args=args, schedule\_interval=None

)

query1 = [

"""select 1;""",

"""show tables in database abcd\_db;""",

]

def count1(\*\*context):

dwh\_hook = SnowflakeHook(snowflake\_conn\_id="snowflake\_conn")

result = dwh\_hook.get\_first("select count(\*) from abcd\_db.public.test3")

[logging.info](http://logging.info/)("Number of rows in `abcd\_db.public.test3` - %s", result[0])

with dag:

query1\_exec = SnowflakeOperator(

task\_id="snowfalke\_task1",

sql=query1,

snowflake\_conn\_id="snowflake\_conn",

)

count\_query = PythonOperator(task\_id="count\_query", python\_callable=count1)

query1\_exec >> count\_query

import pandas as pd

df = pd.read\_csv('763K\_plus\_IOS\_Apps\_Info.csv', index\_col= False )

df = df.reset\_index(drop=True)

df.to\_csv('763K\_plus\_IOS\_Apps.csv', index= False)

#Создаем два [data streams](https://docs.snowflake.com/en/user-guide/streams-intro.html) в Snowflake и настраиваем их на две таблицы (RAW\_TABLE, STAGE\_TABLE).

create stream RAW\_STREAM on table RAW\_TABLE

create stream STAGE\_STREAM on table STAGE\_TABLE

#Записываем данные из CSV в RAW\_TABLE

#Записываем данные из RAW\_STREAM в STAGE\_TABLE

#Записываем данные из STAGE\_STREAM в MASTER\_TABLE

create external table RAW\_TABLE (

date\_part date as to\_date(substr(metadata$filename, 1, 10), 'YYYY/MM/DD'),

ts timestamp as (value:time::timestamp),

user\_id varchar as (value:userId::varchar),

color varchar as (value:color::varchar)

) partition by (date\_part)

location=@my\_ext\_stage

auto\_refresh = false

file\_format=(type=csv);

create stream RAW\_STREAM on external table RAW\_TABLE insert\_only = true;

show streams;

create external table STAGE\_TABLE (

date\_part date as to\_date(substr(metadata$filename, 1, 10), 'YYYY/MM/DD'),

ts timestamp as (value:time::timestamp),

user\_id varchar as (value:userId::varchar),

color varchar as (value:color::varchar)

) partition by (date\_part)

location=@my\_ext\_stage

auto\_refresh = false

file\_format=(type=csv);

create stream STAGE\_STREAM on external table STAGE\_TABLE insert\_only = true;

show streams;

create table MASTER\_TABLE on external STAGE\_STREAM insert\_only = true;