WSL + VSCode with Ubuntu 22.04

Open Windows terminal with administrator

In Power Shell Type:

wsl --list –online (to see ubuntu versions)

wsl --install -d Ubuntu-22.04

For VSCode install normally at Windows

Open WSL in Project directory (create with mkdir accesswith cd out of /root/ use /tmp)

In VSCode go to extensions and download WSL

Close VSCode and reopen with code

Click left corner to initiate

Setting EC2 and Putty

Download Putty

Create Instance in EC2 with putty key

In Putty insert EC2 Id

Go to AUTH and set key

Click open and type Ubuntu

Installing Postgres

With Putty terminal open:

sudo apt-get update

sudo apt upgrade -y

sudo reboot (if necessary)

sudo apt install postgresql postgresql-contrib

chmod 755 /home/ubuntu

sudo chown ubuntu /home/ubuntu

sudo -u postgres psql

CREATE USER admin WITH PASSWORD 'admin';

ALTER USER admin WITH SUPERUSER;

Download airflow

Open VSCode in ubuntu terminal after accesing folder with cd

sudo apt update

sudo apt upgrade -y

sudo apt install python3-pip python3-venv -y

python3 -m venv NEW_VENV

source NEW_VENV/bin/activate

pip install apache-airflow

airflow standalone (Copy user and Password)

Weather API

```python
# Importing our packages
import pandas as pd
import requests
import json

# API Code
# Gathering the data

key = "f6f4e999c7394519b35142300232111"
city = "Belo Horizonte"
URL =
f"http://api.weatherapi.com/v1/forecast.json?key={key}&q={city}&days=7"

weather_data = requests.get(URL)
weather_json = weather_data.json()

# Create a json file to visualize, will remove later
with open('weather_data.json', 'w') as file:
    json.dump(weather_json, file, indent=4)

# Manage the data into a dictionary
# We need to make some intermidiate dictionarys

City = weather_json["location"]["name"]

day = []
max_temp =[]
min_temp = []
avg_temp =[]
rain_mm = []
rain_chance = []
```

```
for date in weather_json["forecast"]["forecastday"]:
    day.append(date["date"])

for i in weather_json["forecast"]["forecastday"]:
    max_temp.append(i["day"]["maxtemp_c"])
    min_temp.append(i["day"]["mintemp_c"])
    avg_temp.append(i["day"]["avgtemp_c"])
    rain_mm.append(i["day"]["totalprecip_mm"])
    rain_chance.append(i["day"]["daily_chance_of_rain"])


final_dict = {
    "City": City,
    "Max temperature [C]": max_temp,
    "Min Temperature [C]": min_temp,
    "Average Temperature [C]": avg_temp,
    "Rain [mm]": rain_mm,
    "Rain Chance": rain_chance
}
```

DAG

```
from datetime import timedelta
from airflow import DAG
from airflow.operators.python import PythonOperator
from airflow.utils.dates import days_ago
from datetime import datetime
from code import extract_weather_data
from code import connect_to_db
from code import load_data

default_args = {
    'owner': 'airflow',
    'depends_on_past': False,
    'retries': 1,
    'retry_delay': timedelta(minutes=1)
}

dag = DAG(
    'weather_api_data',
    default_args=default_args,
    description='Extract weather data from Weather API',
    schedule_interval=timedelta(days=1),
)
```

```
extract_task = PythonOperator(
    task_id='extract_weather_data',
    python_callable=extract_weather_data,
    dag=dag,
)

load_task = PythonOperator(
    task_id='load_data',
    python_callable=load_data,
    dag=dag,
)

# Set the task dependency
extract_task >> load_task
```

SQL TABLE

```
CREATE TABLE weather_BH(
    id serial PRIMARY KEY,
    city VARCHAR(100) NOT NULL,
    max_temp NUMERIC NOT NULL,
    min_temp NUMERIC NOT null,
    avg_temp NUMERIC NOT null,
    rain_mm NUMERIC NOT null,
    rain_chance NUMERIC NOT null
)
```

Final code

```
from datetime import timedelta
from airflow import DAG
from airflow.operators.python import PythonOperator
from airflow.utils.dates import days_ago
from datetime import datetime
import pandas as pd
import requests
import json
import io
import psycopg2
import pendulum

# Function

def extract_weather_data():
    key = "f6f4e999c7394519b35142300232111"
    city = "Belo Horizonte"
    URL =
f"http://api.weatherapi.com/v1/forecast.json?key={key}&q={city}&days=7"

    weather_data = requests.get(URL)
    weather_json = weather_data.json()

    # Create a json file to visualize, will remove later
    with open('weather_data.json', 'w') as file:
        json.dump(weather_json, file, indent=4)

    # Manage the data into a dictionary
    # We need to make some intermidiate dictionarys

    City = weather_json["location"]["name"]

    day = []
    max_temp =[]
    min_temp = []
    avg_temp =[]
    rain_mm = []
    rain_chance = []


    for date in weather_json["forecast"]["forecastday"]:
        day.append(date["date"])

    for i in weather_json["forecast"]["forecastday"]:
        max_temp.append(i["day"]["maxtemp_c"])
        min_temp.append(i["day"]["mintemp_c"])
        avg_temp.append(i["day"]["avgtemp_c"])
        rain_mm.append(i["day"]["totalprecip_mm"])
        rain_chance.append(i["day"]["daily_chance_of_rain"])


    final_dict = {
        "City": City,
        "Max temperature [C]": max_temp,
        "Min Temperature [C]": min_temp,
        "Average Temperature [C]": avg_temp,
        "Rain [mm]": rain_mm,
        "Rain Chance": rain_chance
    }

    df_weather = pd.DataFrame(final_dict)
    return df_weather

def connect_to_db(host, port, database, user, password):
    conncetion = psycopg2.connect(host=host, port=port,
database=database, user=user, password=password)
    return conncetion

def load_data(df, tabela, colunas):
    conn = psycopg2.connect(host='localhost', port='5432',
database='postgres', user='admin', password='admin')
    cur = conn.cursor()
    output = io.StringIO()
    df.to_csv(output, sep='\t', header = True, index = False)
    output.seek(0)
    try:
        cur.copy_from(output, tabela, null = "", columns = colunas)
        conn.commit()
    except Exception as e:
        print(e)
        conn.rollback()

# DAG

default_args = {
    'owner': 'airflow',
    'depends_on_past': False,
    'retries': 1,
    'retry_delay': timedelta(minutes=1),
    'start_date': pendulum.today('UTC').add(days=-1)
}

dag = DAG(
    'weather_api_data',
    default_args=default_args,
    description='Extract weather data from Weather API',
    schedule='0 0 * * 0',
)

def extract_and_return_data(**kwargs):
    extracted_data = extract_weather_data()
    kwargs['ti'].xcom_push(key='extracted_data', value=extracted_data)

extract_task = PythonOperator(
    task_id='extract_weather_data',
    python_callable=extract_and_return_data,
    dag=dag,
)

def load_data_from_xcom(**kwargs):
    extracted_data =
kwargs['ti'].xcom_pull(task_ids='extract_weather_data',
key='extracted_data')

    load_data(extracted_data, 'weather_bh', ['city', 'max_temp',
                                              'min_temp',
                                              'avg_temp',
                                              'rain_mm',
                                                            'rain_chance'])


load_task = PythonOperator(
    task_id='load_data',
    python_callable=load_data_from_xcom,
    dag=dag,
)

# Set the task dependency
extract_task >> load_task
```

/tmp/airflow_project

source venv_airflow/bin/activate

Final code

```
from datetime import timedelta
from airflow import DAG
from airflow.operators.python import PythonOperator
from airflow.utils.dates import days_ago
import pandas as pd
import requests
import io
import psycopg2
import pendulum

# Function

def extract_weather_data():
    key = "f6f4e999c7394519b35142300232111"
    city = "Belo Horizonte"
    URL =
f"http://api.weatherapi.com/v1/forecast.json?key={key}&q={city}&days=7
"

    weather_data = requests.get(URL)
    weather_json = weather_data.json()

    # Manage the data into a dictionary
    # We need to make some intermidiate dictionarys

    City = weather_json["location"]["name"]

    day = []
    max_temp =[]
    min_temp = []
    avg_temp =[]
    rain_mm = []
    rain_chance = []


    for date in weather_json["forecast"]["forecastday"]:
        day.append(date["date"])

    for i in weather_json["forecast"]["forecastday"]:
        max_temp.append(i["day"]["maxtemp_c"])
        min_temp.append(i["day"]["mintemp_c"])
        avg_temp.append(i["day"]["avgtemp_c"])
        rain_mm.append(i["day"]["totalprecip_mm"])
        rain_chance.append(i["day"]["daily_chance_of_rain"])


    final_dict = {
        "city": City,
        "max_temp": max_temp,
        "min_temp": min_temp,
        "avg_temp": avg_temp,
        "rain_mm": rain_mm,
        "rain_chance": rain_chance
    }

    df_weather = pd.DataFrame(final_dict)
    return df_weather

def load_data(df, tabela, colunas):
    conn = psycopg2.connect(host='localhost', port='5432', database='postgres', user='admin', password='admin')
    cur = conn.cursor()
    output = io.StringIO()
    df.to_csv(output, sep='\t', header = True, index = False)
    output.seek(0)
    try:
        cur.copy_from(output, tabela, null = "", columns = colunas)
        conn.commit()
    except Exception as e:
        print(e)
        conn.rollback()

# DAG

default_args = {
    'owner': 'airflow',
    'depends_on_past': False,
    'retries': 1,
    'retry_delay': timedelta(minutes=1),
    'start_date': pendulum.today('UTC').add(days=-1)
}
```

```python

dag = DAG(
    'weather_api_data',
    default_args=default_args,
    description='Extract weather data from Weather API',
    schedule='0 0 * * 0',
)

def extract_and_return_data(**kwargs):
    extracted_data = extract_weather_data()
    kwargs['ti'].xcom_push(key='extracted_data',
value=extracted_data)

extract_task = PythonOperator(
    task_id='extract_weather_data',
    python_callable=extract_and_return_data,
    dag=dag,
)

def load_data_from_xcom(**kwargs):
    extracted_data =
kwargs['ti'].xcom_pull(task_ids='extract_weather_data',
key='extracted_data')

    load_data(extracted_data, 'weather_bh', ['city', 'max_temp',
                                              'min_temp',
                                              'avg_temp',
                                              'rain_mm',
                                              'rain_chance'])


load_task = PythonOperator(
    task_id='load_data',
    python_callable=load_data_from_xcom,
    dag=dag,
)

# Set the task dependency
extract_task >> load_task
```