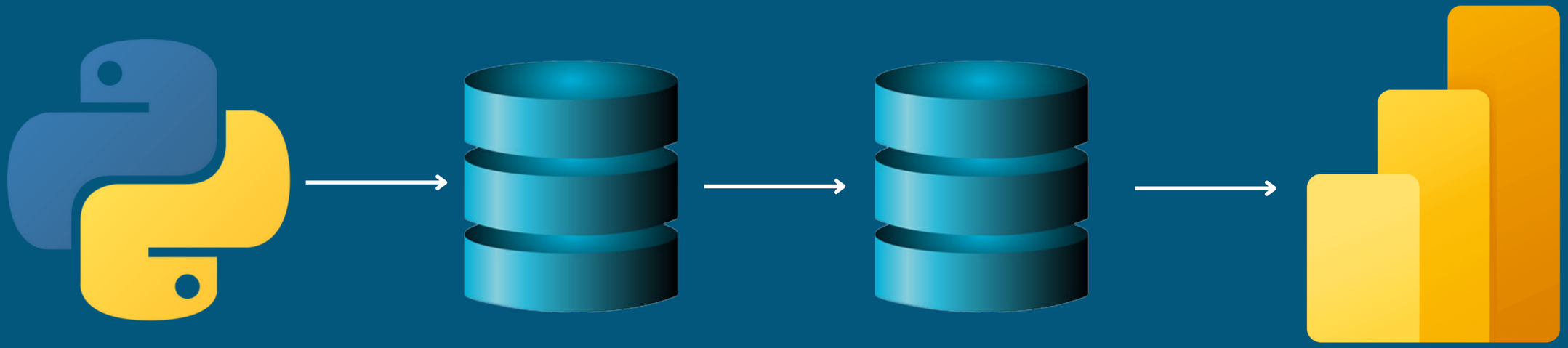
DATA ANALYSIS

PROJECT: SENSOR DATA ANALYSIS

MY TASK?

TASK STRUCTURE



Python

We generate sensor data for process.

Database A

We scrap sensor data in normal form from Python to Database A.

Database B

In Database B, we get data in denormalize form from Database A.

Power BI

Finally, we visualize sensor data with Power BI.

CONTENT

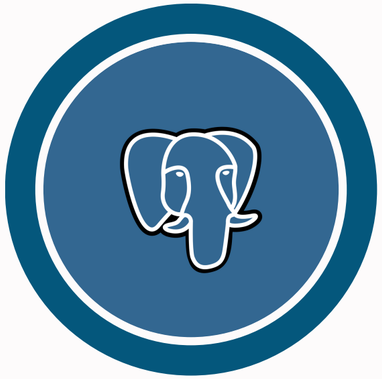
01 TECHNICAL SIDE

02 BUSINESS SIDE

TOOLS:

PYTHON

Generate data /
Send to database



PostgreSQL

Create database /
Using pgagent jobs

Power BI

Data visualization

TECHNICAL SIDE

PROCESS

01 GENERATION OF SENSOR DATA

02 CONNECT TO DATABASE A

03 CREATE DATABASE B

04 INSTALL PG AGENT

05 CONNECT WITH POWER BI

06 RESULT

COLUMNS:

- TEMPERATURE
- WIND SPEED
- HUMIDITY
- WIND_DIRECTION
- SOLAR RADIATION
- SOIL MOISTURE
- TEMPERATURE
- SOIL PH LEVELS
- NUTRIENT LEVELS

PYTHON

01

```python
import psycopg2
import random

while True:
    # Insert data into soil_sensor
    insert_sscript = 'INSERT INTO soil_sensor (id, soil_moisture_content, soil_
    soil_moisture_content = random.uniform(10, 50)
    soil_pH_levels = random.uniform(5.5, 7.0)
    soil_temperature = random.uniform(5, 40)
    nutrient_levels = ['N', 'P', 'K']
    soil_nutrient_levels = random.choice(nutrient_levels)
    insert_svalue = (id, soil_moisture_content,soil_pH_levels,soil_temperature,
    cur.execute(insert_sscript,insert_svalue)

    conn.commit()
    id += 1
    time.sleep(10)
```

We generate data in python using 'while loop'. When we create this data, we set the limits for the features accordingly. Example: (temp Min: 0 , Max: 40) , (wind speed Min: 0 , Max: 30) etc.

COLUMNS AND VALUE RANGE:

TEMPERATURE	-	(0,40)
WIND SPEED	-	(0,30)
HUMIDITY	-	(30,80)
WIND_DIRECTION	-	(0,360)
SOLAR RADIATION	-	(0,2000)
SOIL MOISTURE	-	(10,50)
TEMPERATURE	-	(5,40)
SOIL PH LEVELS	-	(5.5,7.0)
NUTRIENT LEVELS	-	['N','P','K']

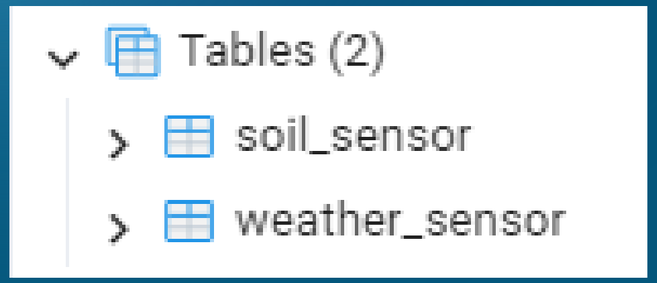
02

We are contacting database A. Then we send the data generated by python to the database. For this we use the "psycopg2" library.

```
import psycopg2
import random
import time
# Let's define your PostgreSQL database connection parameters
host = 'localhost'
database = 'databaseA'
username = 'postgres'
pwd = 'eriya494949'
port = 5432
conn = None
cur = None

conn = psycopg2.connect(
        host=host,
        dbname=database,
        user=username,
        password=pwd,
        port=port
    )
cur = conn.cursor()
```

Check-up...



	id [PK] integer	soil_moisture_content double precision	soil_ph_levels double precision	soil_temperature double precision	soil_nutrient_levels character varying (10)
1	1	34.976367805933165	6.026315932107942	9.785870049288185	N
2	2	20.8926517072365	6.951530980908673	16.554476617225312	K
3	3	48.75662559204609	5.811128472497669	17.545042149707164	K

	id [PK] integer	temperature double precision	wind_speed double precision	humidity double precision	wind_direction double precision	solar_radiation double precision
1	1	37.50818659144495	19.870317969893357	60	299.5789644899922	325.0176853954021
2	2	17.125175657123727	1.495384839464664	54	269.16792491312094	275.74438432502825
3	3	29.222790670553273	14.508518157274528	51	125.02904192657596	1706.5261572720283

METHOD

1

03

OPERATION:

- We write all operation in Python which :
- 1) Create Table in DatabaseB
- 2) Create Link Between DatabaseB and DatabaseA
- 3) Adding Data To DatabaseB


```
#Next, establish a connection to databaseB
connect = """SELECT dblink_connect('myconn1',
'dbname=databaseA port=5432 host=localhost user=postgres password=eriya494949');
"""
```

```
#Now, you can execute queries on databaseB from databaseA
add_to_b = f"""insert into raw_data(w_temperature, wind_speed, w_humidity, wind_
SELECT *
FROM dblink('myconn1',
'SELECT w.temperature, w.wind_speed, w.humidity, w.wind_direction, w.solar_radia
AS p(temperature FLOAT, wind_speed FLOAT, humidity FLOAT, wind_direction FLOAT,
"""
```

BUT HOW WE DETERMINE SCHEDULES?

```
import datetime

current_date = datetime.date.today()
day_of_week = current_date.weekday()+1
if day_of_week in [6,7]:
    print(f"Day of the week (number): {day_of_week}.
          No reporting today.")
continue
```

Week day (Exception)

METHOD

2

POSTGRESQL

03

```
-- First, let's create the dblink extension if not already created
CREATE EXTENSION IF NOT EXISTS dblink;

SELECT dblink_connect('myconn',
   'dbname=databaseA port=5432 host=localhost
                    user=postgres password=eriya494949');

insert into raw_data(w_temperature, wind_speed, w_humidity,
                     wind_direction, solar_radiation, sensor_id,
                     soil_moisture_content, soil_pH_levels, soil_temperature,
                     soil_nutrient_levels)
SELECT *
FROM dblink('myconn',
   'SELECT w.temperature, w.wind_speed, w.humidity, w.wind_direction,
            w.solar_radiation, s.id, s.soil_moisture_content, s.soil_pH_levels,
            s.soil_temperature, s.soil_nutrient_levels FROM soil_sensor s
            INNER JOIN weather_sensor w ON s.id = w.id')
AS p(temperature FLOAT, wind_speed FLOAT, humidity FLOAT, wind_direction FLOAT,
     solar_radiation FLOAT, id INT, soil_moisture_content FLOAT,
     soil_pH_levels FLOAT, soil_temperature FLOAT, soil_nutrient_levels VARCHAR(10));
```

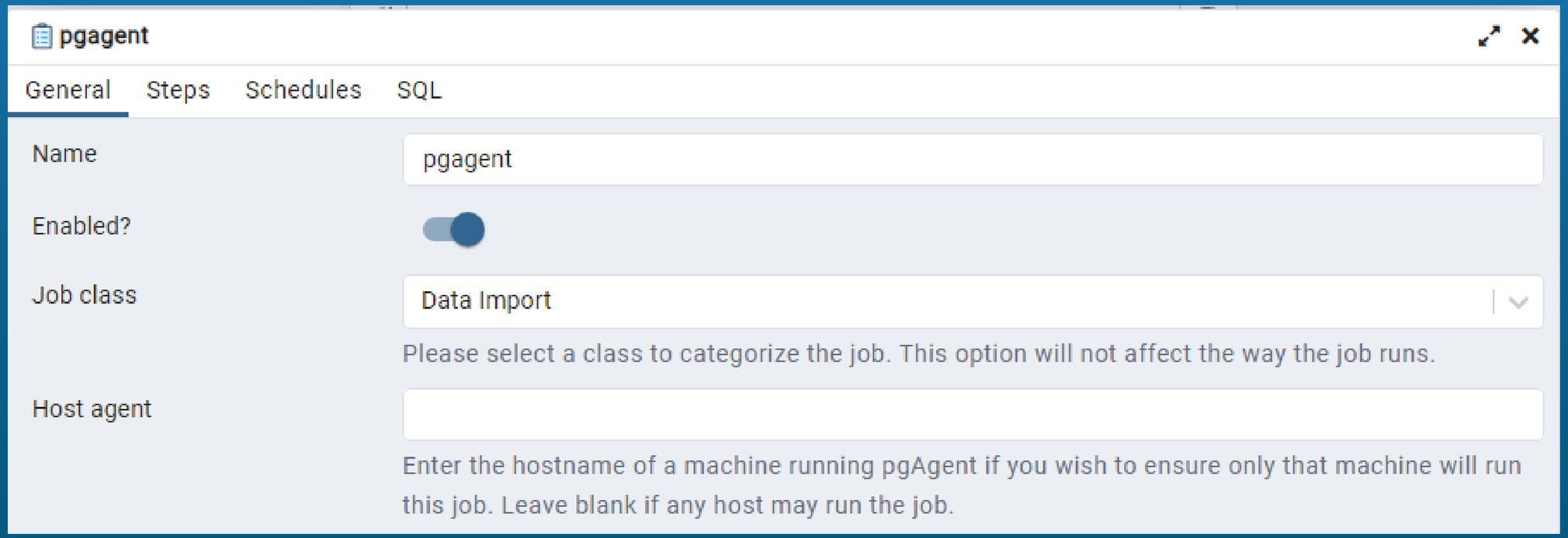
In the 3rd stage, we denormalize and move the data from database A to database B. We will use DatabaseB for visualization.

Check-up...

	w_temperature double precision	wind_speed double precision	w_humidity double precision	wind_direction double precision	solar_radiation double precision
1	37.50818659144495	19.870317969893357	60	299.5789644899922	325.0176853954021
2	17.125175657123727	1.495384839464664	54	269.16792491312094	275.74438432502825
3	29.222790670553273	14.508518157274528	51	125.02904192657596	1706.5261572720283

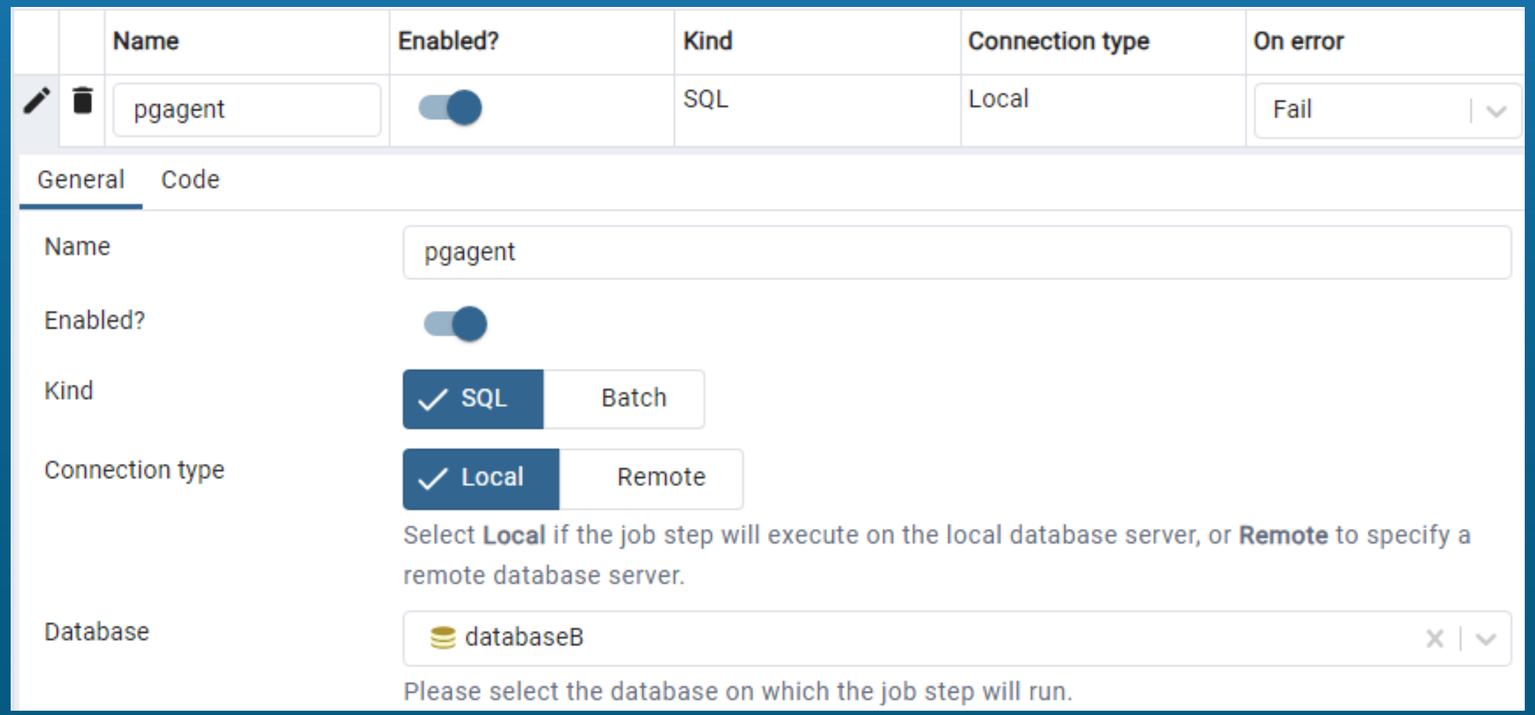
sensor_id [PK] integer	soil_moisture_content double precision	soil_ph_levels double precision	soil_temperature double precision	soil_nutrient_levels character varying (10)
1	34.976367805933165	6.026315932107942	9.785870049288185	N
2	20.8926517072365	6.951530980908673	16.554476617225312	K
3	48.75662559204609	5.811128472497669	17.545042149707164	K

04



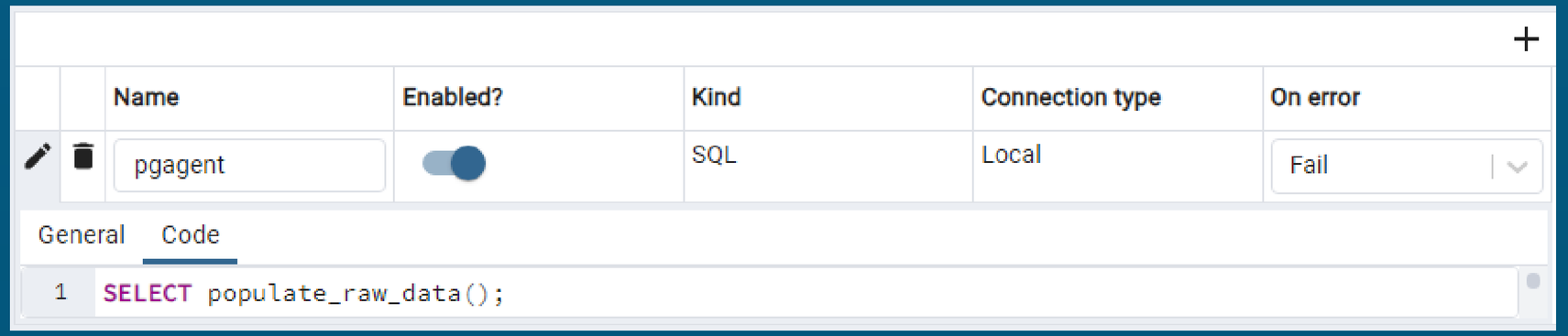
We are building a pagent for databaseB to automatically download information. First, we fill in the "general" part.

04...



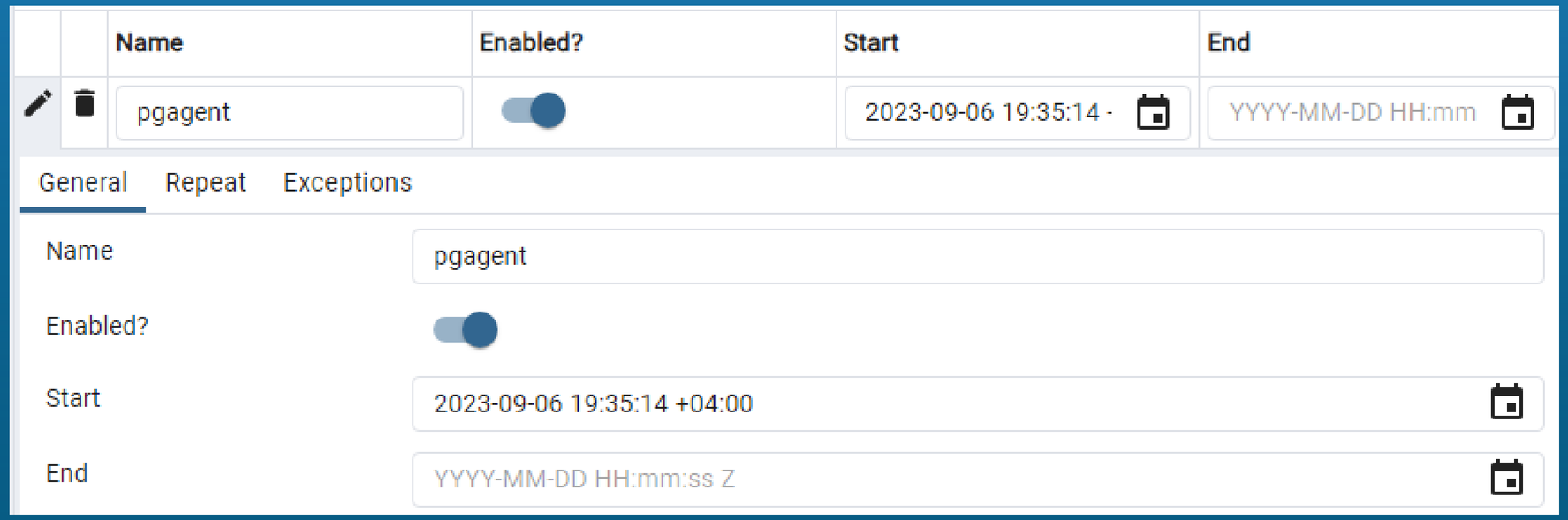
Then we fill in the "steps" part. In this section, we specify which database we will use pgagent.

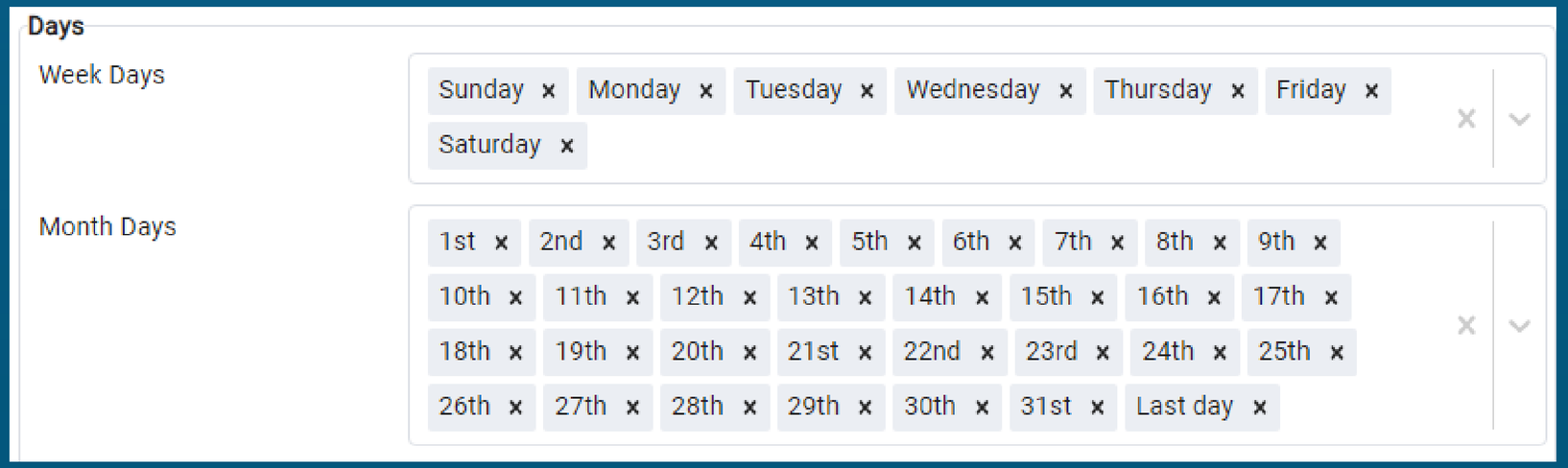
In the code section, we write the code we want to repeat automatically.



04...

Then we fill in the "schedules" field. In this section, I record the start and end time of pgagent.

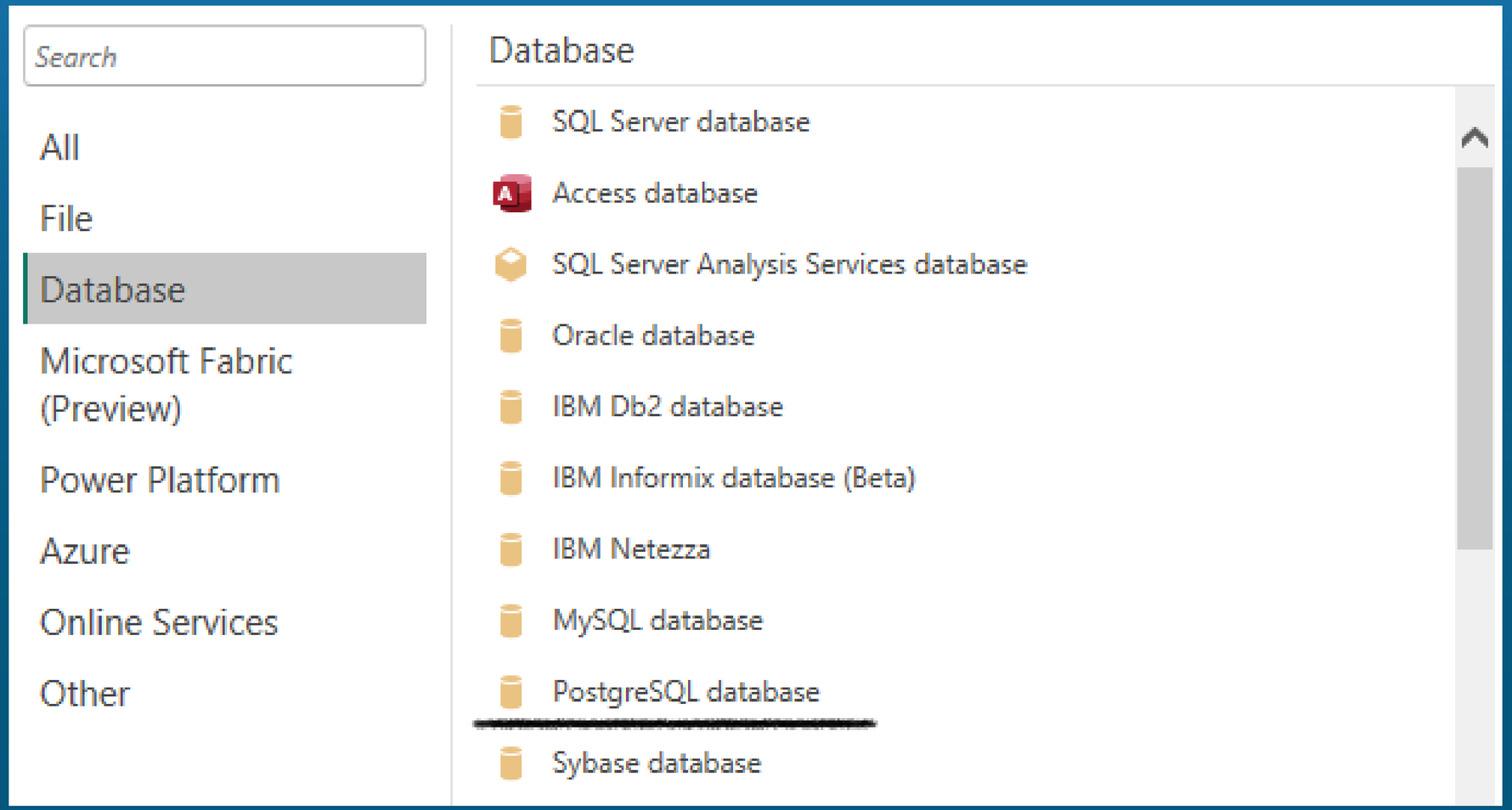




In the next section, we specify the dates we want the pgagent to be active. (Months, Years, Days, Hours, Minutes)

POWER BI

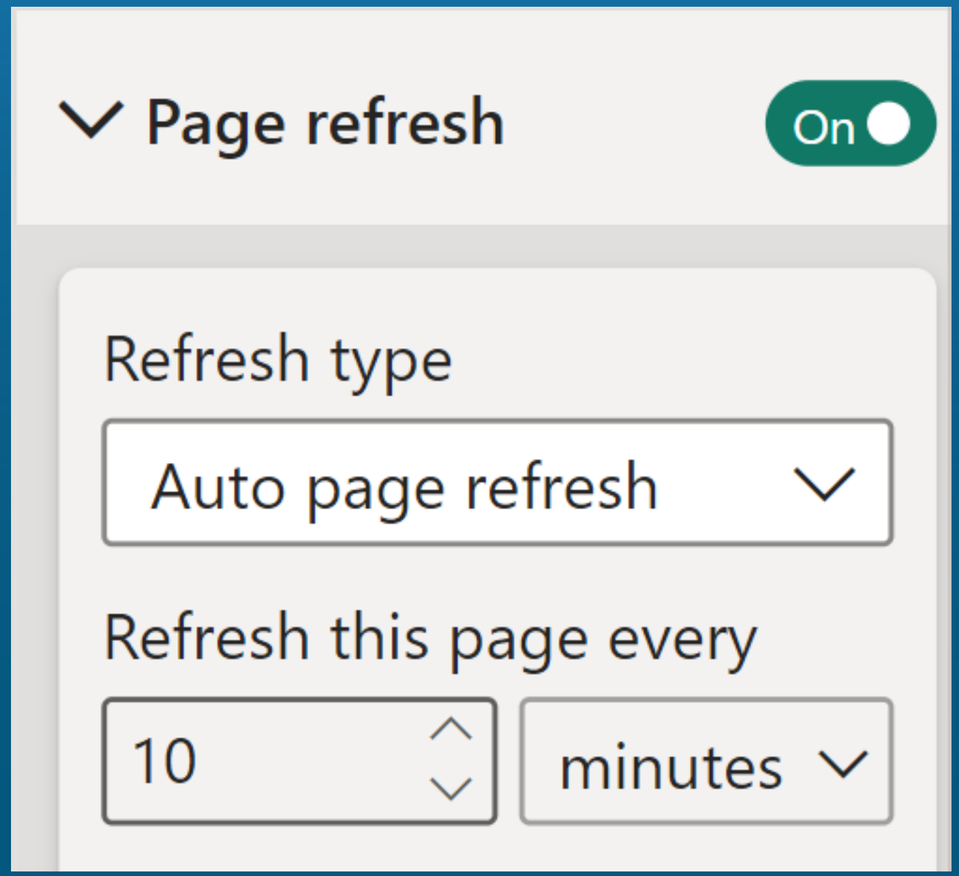
05



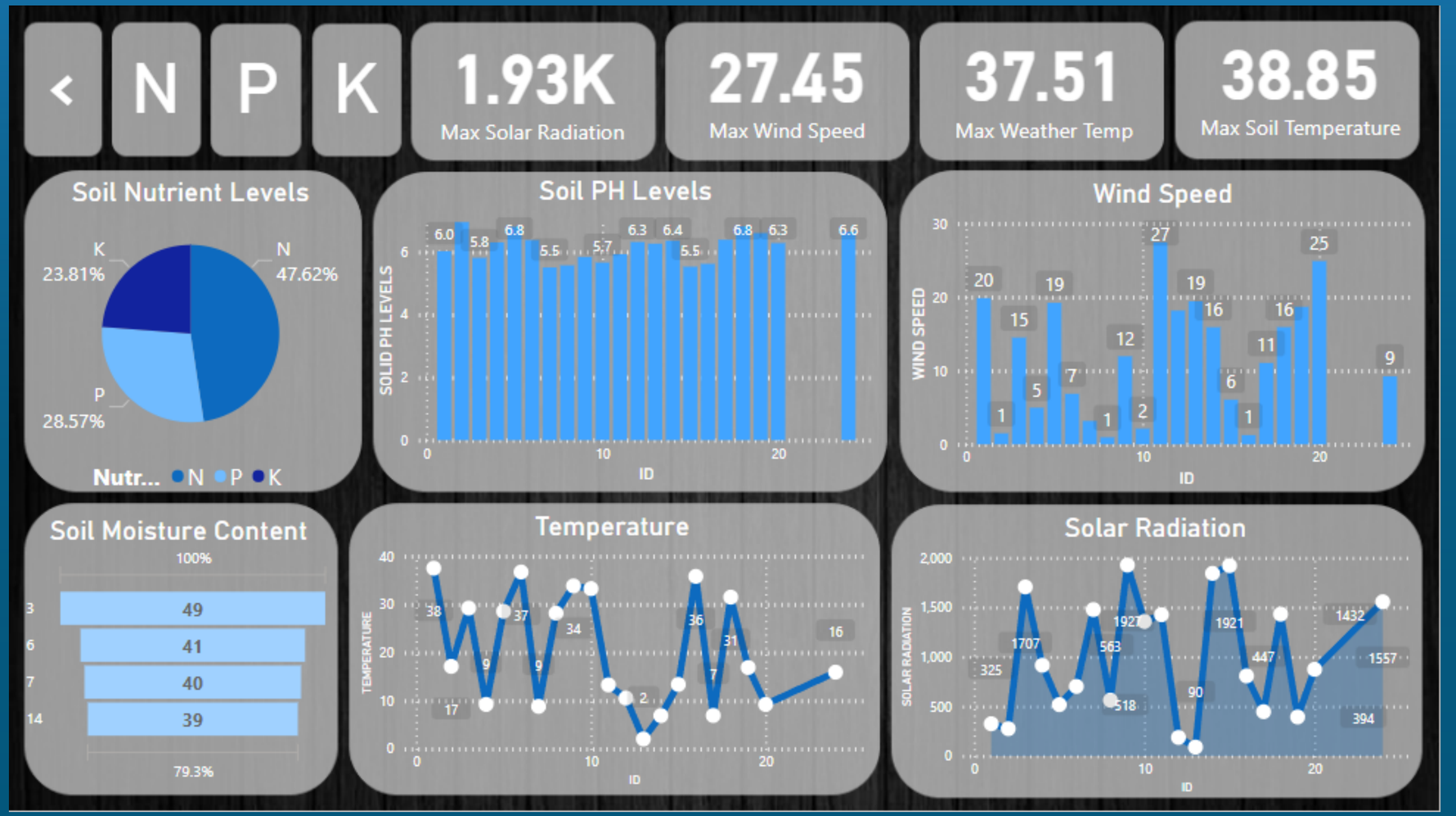
We enter the data in DatabaseB into Power Bi. Accordingly, we select the PostgreSQL share.

05...

Finally, I fix the "page refresh" section. In this section I set the time for automatic update.



06



Finally, we prepare a dashboard using PowerBI to visualize the data.

BUSINESS SIDE

BUSINESSES

02

WEATHER FORECASTING

04

SMART AGRICULTURE SOLUTIONS

AGRICULTURE AND FARMING

01

ENVIRONMENTAL MONITORING

03

01

AGRICULTURE AND FARMING

- Optimizing irrigation based on soil moisture levels.
- Managing soil conditions (pH, temperature, nutrient levels) to improve crop yields.
- Reducing resource usage (water, fertilizers) through data-driven decisions.

02

- Parameters such as temperature, wind speed, humidity, wind direction, and solar radiation.
- Industries : 1. meteorological agencies, 2. research institutions, and 3. businesses.

WEATHER FORECASTING

03

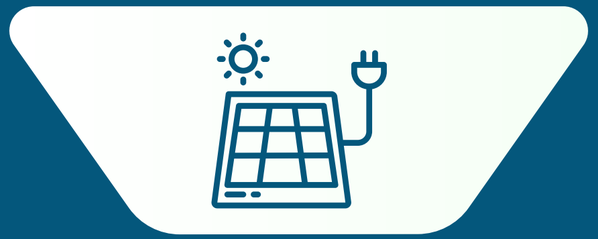
ENVIRONMENTAL MONITORING

- Sensor data is pivotal for understanding environmental dynamics. It helps us monitor weather impacts on soil health and climate change effects.
- Long-term data collection supports informed decisions and environmental protection.

04

- Smart agriculture solutions provide real-time data to farmers and offer data-driven recommendations for crop management, pest control, and irrigation.
- These innovations enhance efficiency and profitability in agriculture, empowering farmers with actionable insights for optimal decision-making.

SMART AGRICULTURE SOLUTIONS



CONCLUSION

- In summary, our sensor data project opens doors to smarter, more sustainable practices in agriculture and environmental monitoring. With real-time insights and data-driven solutions, we can enhance efficiency and make informed decisions, ultimately contributing to a greener and more productive world.

THANK'S FOR WATCHING