Weather Data Streaming with Apache Kafka

Erickson Figueroa

Executive Summary:
The Weather Data Processing System collects, processes, and analyzes weather data from multiple cities and stores it in a PostgreSQL database. The system consists of four main components: producer, consumer, weather data module, and weather summary module.

Producer Component:

- The producer component fetches weather data from the OpenWeatherMap API for specified cities.
- It uses a Kafka producer to send the weather data to a Kafka topic named "weather-data."
- City names and weather data are retrieved using the get_weather_data function from the weather_data module.
- If the weather data fetches successfully, it sends it to the Kafka topic, and logging generates records for success, warning, and error events.

Consumer Component:

- The consumer component listens to the "weather-data" Kafka topic and retrieves weather data messages.
- Retrieved weather data is inserted into a PostgreSQL database table named "weather_detail."
- Generate logging records for successful data insertion and any encountered errors.

Weather Data Module:

- The weather_data module provides functionality to retrieve weather data for a given city from the OpenWeatherMap API.
- The get_weather_data function constructs the API URL, makes a GET request to the API, and parses the JSON response to extract relevant weather information.
- Negative temperature values are handled by converting them to strings and adding a "-" sign if they are negative.

Weather Summary Module:

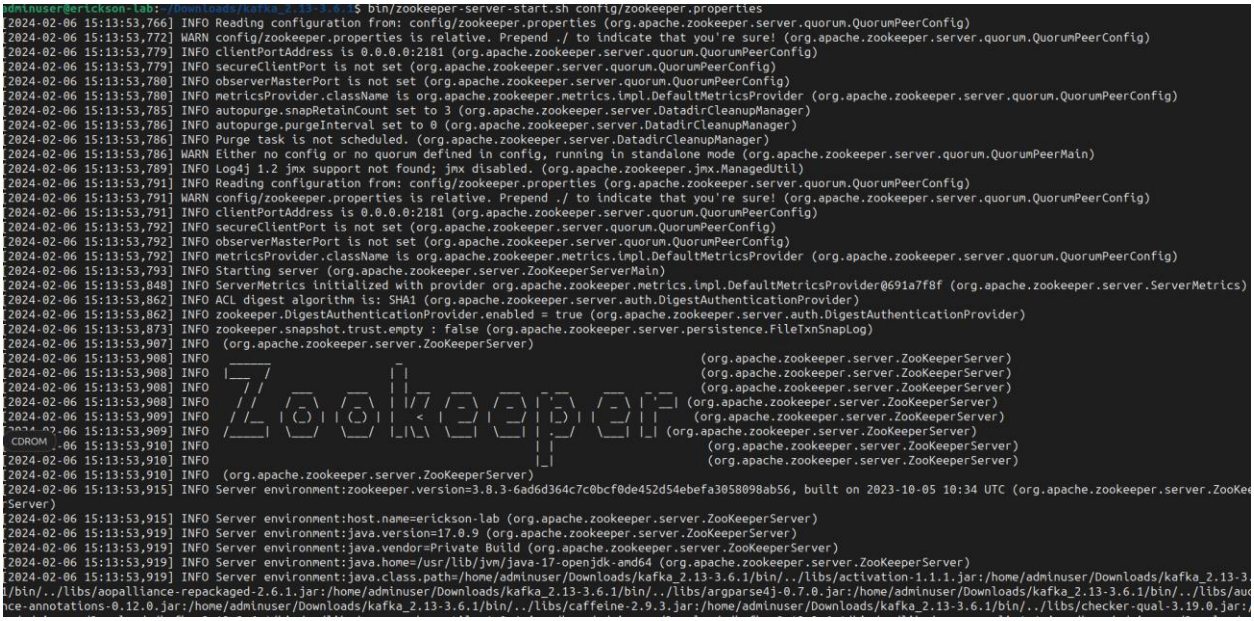
- The weather_summary module summarizes the weather data stored in the "weather_detail" table.
- It calculates the minimum and maximum values for each city's temperature, wind speed, and humidity.

- The summarized data is formatted and inserted into a PostgreSQL table named "weather_summary."
- Implemented error handling to catch and log exceptions during the summary process.

Closing: The Weather Data Processing System provides a solution for collecting, storing, and summarizing weather data. It leverages Kafka for real-time data streaming and PostgreSQL for data persistence and integrates error logging to ensure system reliability. With its modular architecture and comprehensive functionality, the system effectively meets weather data management and analysis requirements.

Zookeeper service

```
adminuser@erickson-lab:~/Downloads/kafka_2.13-3.6.1$ bin/zookeeper-server-start.sh  zookeeper.properties
```



Kafka service

```
adminuser@erickson-lab:~/Downloads/kafka_2.13-3.6.1$ bin/kafka-server-start.sh config/server.properties
```

Creating a Kafka topic

```
(kafkaenv) adminuser@erickson-lab:~/Downloads/kafka_2.13-3.6.1$ ./bin/kafka-topics.sh --create --topic weather-data --bootstrap-server localhost:9092 --replication-factor 1 --partitions 2
```

```
(kafkaenv) adminuser@erickson-lab:~/Downloads/kafka_2.13-3.6.1$ ./bin/kafka-topics.sh --list --bootstrap-server localhost:9092
__consumer_offsets
quickstart-events
weather-data
(kafkaenv) adminuser@erickson-lab:~/Downloads/kafka_2.13-3.6.1$
```

The producer's performance (obtaining serialized data from OpenWeatherMap API and send to the consumer)

```
(kafkaenv) adminuser@erickson-lab:~/kafka_app_venv$ /home/adminuser/kafka_app_venv/kafkaenv/bin/python /home/adminuser/kafka_app_venv/kafkaenv/kafka_weather_app/producer.py
{'city_name': 'Winnipeg', 'current_temperature': '1.42', 'current_wind_speed': 4.63, 'current_humidity': 98}
{'city_name': 'Vancouver', 'current_temperature': '4.69', 'current_wind_speed': 7.72, 'current_humidity': 87}
{'city_name': 'Winnipeg', 'current_temperature': '1.78', 'current_wind_speed': 5.14, 'current_humidity': 98}
{'city_name': 'Vancouver', 'current_temperature': '4.69', 'current_wind_speed': 7.72, 'current_humidity': 87}
{'city_name': 'Winnipeg', 'current_temperature': '1.42', 'current_wind_speed': 4.63, 'current_humidity': 98}
{'city_name': 'Vancouver', 'current_temperature': '4.69', 'current_wind_speed': 7.72, 'current_humidity': 87}
{'city_name': 'Winnipeg', 'current_temperature': '1.55', 'current_wind_speed': 5.14, 'current_humidity': 98}
{'city_name': 'Vancouver', 'current_temperature': '4.69', 'current_wind_speed': 7.72, 'current_humidity': 87}
{'city_name': 'Winnipeg', 'current_temperature': '1.42', 'current_wind_speed': 4.63, 'current_humidity': 98}
'city_name': 'Vancouver', 'current_temperature': '4.69', 'current_wind_speed': 7.72, 'current_humidity': 87}
city_name': 'Winnipeg', 'current_temperature': '1.78', 'current_wind_speed': 5.14, 'current_humidity': 98}
{'city_name': 'Vancouver', 'current_temperature': '4.69', 'current_wind_speed': 7.72, 'current_humidity': 87}
{'city_name': 'Winnipeg', 'current_temperature': '1.42', 'current_wind_speed': 4.63, 'current_humidity': 98}
{'city_name': 'Vancouver', 'current_temperature': '4.69', 'current_wind_speed': 7.72, 'current_humidity': 87}
{'city_name': 'Winnipeg', 'current_temperature': '1.55', 'current_wind_speed': 5.14, 'current_humidity': 98}
{'city_name': 'Vancouver', 'current_temperature': '4.69', 'current_wind_speed': 7.72, 'current_humidity': 87}
{'city_name': 'Winnipeg', 'current_temperature': '1.42', 'current_wind_speed': 4.63, 'current_humidity': 98}
{'city_name': 'Vancouver', 'current_temperature': '4.69', 'current_wind_speed': 7.72, 'current_humidity': 87}
{'city_name': 'Winnipeg', 'current_temperature': '1.78', 'current_wind_speed': 5.14, 'current_humidity': 98}
{'city_name': 'Vancouver', 'current_temperature': '4.69', 'current_wind_speed': 7.72, 'current_humidity': 87}
{'city_name': 'Winnipeg', 'current_temperature': '1.42', 'current_wind_speed': 4.63, 'current_humidity': 98}
{'city_name': 'Vancouver', 'current_temperature': '4.69', 'current_wind_speed': 7.72, 'current_humidity': 87}
{'city_name': 'Winnipeg', 'current_temperature': '1.55', 'current_wind_speed': 5.14, 'current_humidity': 98}
{'city_name': 'Vancouver', 'current_temperature': '4.69', 'current_wind_speed': 7.72, 'current_humidity': 87}
{'city_name': 'Winnipeg', 'current_temperature': '1.42', 'current_wind_speed': 4.63, 'current_humidity': 98}
{'city_name': 'Vancouver', 'current_temperature': '4.69', 'current_wind_speed': 7.72, 'current_humidity': 87}
{'city_name': 'Winnipeg', 'current_temperature': '1.78', 'current_wind_speed': 5.14, 'current_humidity': 98}
{'city_name': 'Vancouver', 'current_temperature': '4.69', 'current_wind_speed': 7.72, 'current_humidity': 87}
{'city_name': 'Winnipeg', 'current_temperature': '1.42', 'current_wind_speed': 4.63, 'current_humidity': 98}
```

The consumer's performance (obtain the deserialized data from the producer and insert it into the database)

```
(kafkaenv) adminuser@erickson-lab:~/kafka_app_venv$ /home/adminuser/kafka_app_venv/kafkaenv/bin/python /home/adminuser/kafka_app_venv/kafkaenv/kafka_weather_app/consumer.py
/home/adminuser/kafka_app_venv/kafkaenv/kafka_weather_app/weather_summary.py:16: UserWarning: pandas only supports SQLAlchemy connectable (engine/connection) or database string URI or sqli
te3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please consider using SQLAlchemy.
  weather_df = pd.read_sql_query(query, conn)
Received weather data: {'city_name': 'Winnipeg', 'current_temperature': '1.55', 'current_wind_speed': 3.6, 'current_humidity': 98}
The weather data were inserted into the database
Received weather data: {'city_name': 'Vancouver', 'current_temperature': '4.85', 'current_wind_speed': 6.17, 'current_humidity': 87}
The weather data were inserted into the database
Received weather data: {'city_name': 'Vancouver', 'current_temperature': '4.85', 'current_wind_speed': 6.17, 'current_humidity': 87}
The weather data were inserted into the database
Received weather data: {'city_name': 'Winnipeg', 'current_temperature': '1.55', 'current_wind_speed': 3.6, 'current_humidity': 98}
The weather data were inserted into the database
Received weather data: {'city_name': 'Vancouver', 'current_temperature': '4.69', 'current_wind_speed': 7.72, 'current_humidity': 87}
The weather data were inserted into the database
Received weather data: {'city_name': 'Winnipeg', 'current_temperature': '1.42', 'current_wind_speed': 4.63, 'current_humidity': 98}
The weather data were inserted into the database
Received weather data: {'city_name': 'Vancouver', 'current_temperature': '4.69', 'current_wind_speed': 7.72, 'current_humidity': 87}
The weather data were inserted into the database
Received weather data: {'city_name': 'Winnipeg', 'current_temperature': '1.55', 'current_wind_speed': 5.14, 'current_humidity': 98}
The weather data were inserted into the database
Received weather data: {'city_name': 'Vancouver', 'current_temperature': '4.69', 'current_wind_speed': 7.72, 'current_humidity': 87}
The weather data were inserted into the database
Received weather data: {'city_name': 'Winnipeg', 'current_temperature': '1.42', 'current_wind_speed': 4.63, 'current_humidity': 98}
The weather data were inserted into the database
Received weather data: {'city_name': 'Vancouver', 'current_temperature': '4.69', 'current_wind_speed': 7.72, 'current_humidity': 87}
The weather data were inserted into the database
Received weather data: {'city_name': 'Vancouver', 'current_temperature': '4.69', 'current_wind_speed': 7.72, 'current_humidity': 87}
The weather data were inserted into the database
Received weather data: {'city_name': 'Vancouver', 'current_temperature': '4.69', 'current_wind_speed': 7.72, 'current_humidity': 87}
The weather data were inserted into the database
Received weather data: {'city_name': 'Winnipeg', 'current_temperature': '1.42', 'current_wind_speed': 4.63, 'current_humidity': 98}
The weather data were inserted into the database
Received weather data: {'city_name': 'Winnipeg', 'current_temperature': '1.78', 'current_wind_speed': 5.14, 'current_humidity': 98}
The weather data were inserted into the database
Received weather data: {'city_name': 'Vancouver', 'current_temperature': '4.69', 'current_wind_speed': 7.72, 'current_humidity': 87}
The weather data were inserted into the database
Received weather data: {'city_name': 'Vancouver', 'current_temperature': '4.69', 'current_wind_speed': 7.72, 'current_humidity': 87}
```

Table to capture summary data (weather_summary)

```
22  select * from weather_summary order by 1 desc
```

Data Output | Messages | Notifications

	id [PK] integer	city character varying (100)	temperature_deg_c character varying (100)	wind_speed_km_h character varying (100)	humidity_percentage character varying (100)	received_at timestamp without time zone
1	12	Winnipeg	-1.11 to 1.92	3.6 to 4.58	70 to 98	2024-02-08 06:31:24.233372
2	11	Vancouver	4.85 to 5.01	6.17 to 6.71	86 to 87	2024-02-08 06:31:24.233372

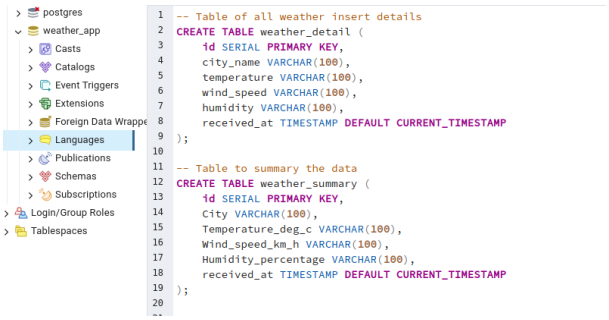
Table to capture the complete data without summarizing (weather_detail)

```
25  select * from weather_detail
26
```

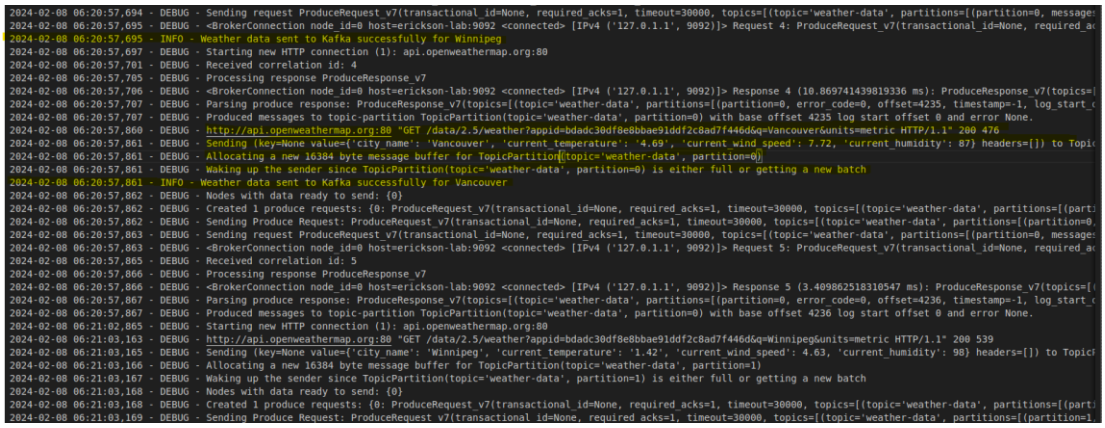
Data Output | Messages | Notifications

	id [PK] integer	city_name character varying (100)	temperature character varying (100)	wind_speed character varying (100)	humidity character varying (100)	received_at timestamp without time zone
1	1	Winnipeg	1.92	3.6	98	2024-02-08 04:29:52.326443
2	2	Vancouver	5.01	6.71	86	2024-02-08 04:29:52.335367
3	3	Winnipeg	1.92	3.6	98	2024-02-08 04:29:52.343267
4	4	Vancouver	5	6.71	87	2024-02-08 04:29:52.34836
5	5	Winnipeg	1.92	3.6	98	2024-02-08 04:29:52.352756
6	6	Vancouver	5	6.71	87	2024-02-08 04:29:52.355707
7	7	Winnipeg	1.92	3.6	98	2024-02-08 04:29:52.358019
8	8	Vancouver	5	6.71	86	2024-02-08 04:29:52.360764
9	9	Winnipeg	1.92	3.6	98	2024-02-08 04:29:52.362917
10	10	Winnipeg	1.92	3.6	98	2024-02-08 04:29:52.366994
11	11	Winnipeg	1.92	3.6	98	2024-02-08 04:29:52.369207
12	12	Vancouver	5.01	6.71	86	2024-02-08 04:29:52.371673
13	13	Vancouver	5	6.71	87	2024-02-08 04:29:52.373255
14	14	Winnipeg	1.92	3.6	98	2024-02-08 04:29:52.375045

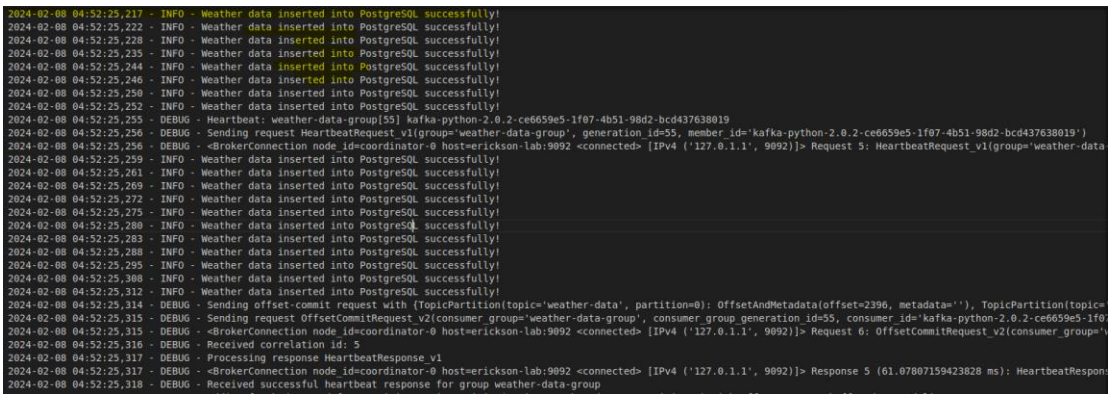
Table structures



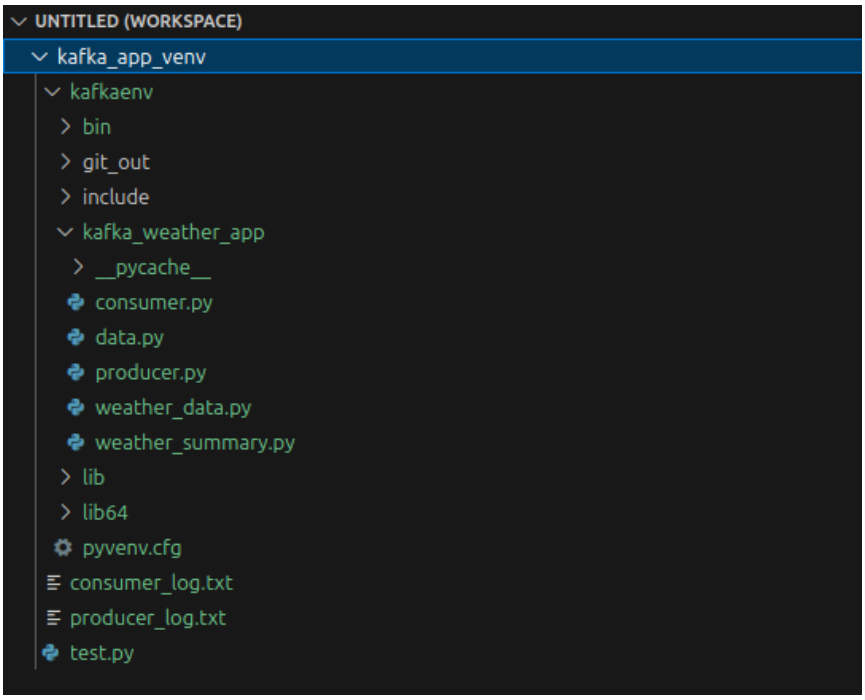
Producer's log file:



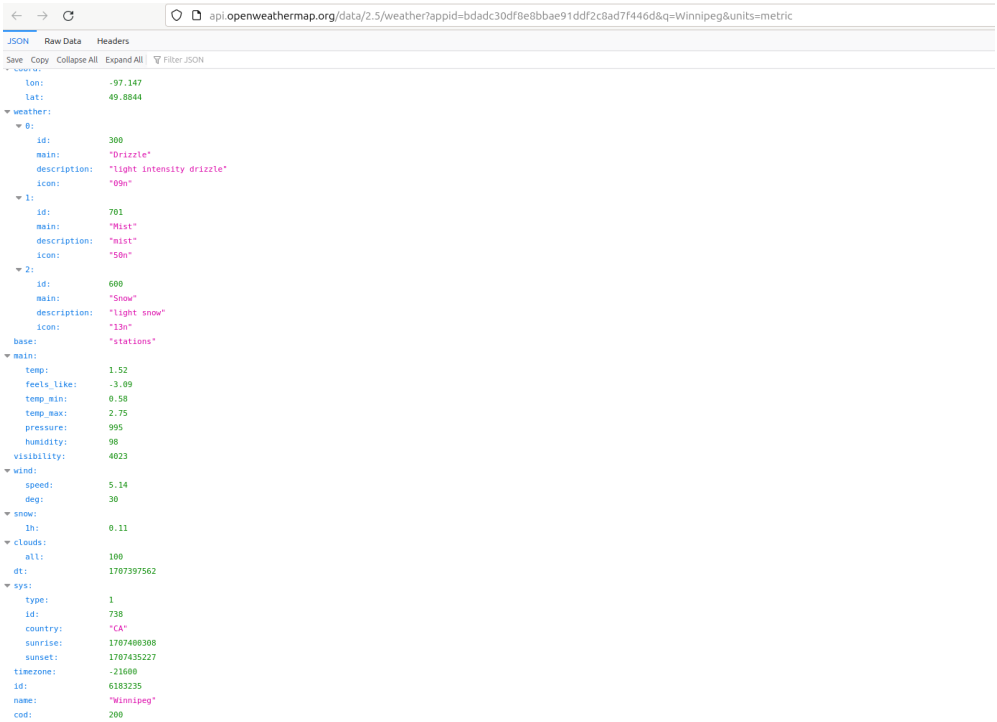
Consumer's log file:



Project structure



Sample API response result from OpenWeatherMap:



Source Code

Weather_data.py (to get the data from the OpenWeatherMap API):

```
import requests

# API key from openweathermap.org
API_KEY = "bdadc30df8e8bbae91ddf2c8ad7f446d"

# Base URL for the OpenWeatherMap API
BASE_URL = "http://api.openweathermap.org/data/2.5/weather?appid="

# Constant to define the parameter units as metrics (Celsius)
CELSIUS = "metric"

def convert_temperature(temperature):
"""
Convert temperature to a string and add "-" if it's negative.

Args:
temperature (float): The temperature value.

Returns:
str: The temperature as a string.
"""
return str(temperature) if temperature >= 0 else "-" + str(abs(temperature))


def get_weather_data(city_name):
"""
Retrieve weather data for a given city from the OpenWeatherMap API.

Args:
city_name (str): The name of the city.

Returns:
dict: A dictionary containing the weather data, or None if the city is not found.
"""

# Construct the complete URL
url = f"{BASE_URL}{API_KEY}&q={city_name}&units={CELSIUS}"

# Make a GET request to the OpenWeatherMap API
response = requests.get(url)

# Check if the city is found
if response.status_code == 200:
```

```
# Parse the JSON response
data = response.json()

# Extract relevant weather information
weather_data = {
"city_name": city_name,
"current_temperature": convert_temperature(data["main"]["temp"]),
"current_wind_speed": data["wind"]["speed"],
"current_humidity": data["main"]["humidity"]
}
return weather_data
else:
print(f"Error retrieving weather data: {response.status_code}")
return None
```

producer.py (to get the data from the OpenWeatherMap API and send to Kafka topic):

```
import logging
from kafka import KafkaProducer
import json
import time
from weather_data import get_weather_data

# Configure logging
logging.basicConfig(filename='producer_log.txt', level=logging.DEBUG,
format='%(asctime)s - %(levelname)s - %(message)s')

# Kafka producer settings
producer = KafkaProducer(bootstrap_servers=['localhost:9092'],
value_serializer=lambda x: json.dumps(x).encode('utf-8'))

# List of city names to get the weather data
city_names = ["Winnipeg", "Vancouver"]

# Process to send the weather data to kafka consumer
if __name__ == "__main__":
while True:
for city_name in city_names:
try:
# Get weather data
weather_data = get_weather_data(city_name)
print(weather_data)
if weather_data:
# Send weather data to Kafka
producer.send("weather-data", weather_data)
logging.info(f"Weather data sent to Kafka successfully for {city_name}")
else:
```

```
logging.warning(f"City {city_name} not found.")
except Exception as e:
logging.error(f"An error occurred sending the data for {city_name}: {str(e)}")

# Wait for 5 seconds before sending the next request
time.sleep(5)
```

consumer.py (to get the data from the producer and insert to the database):

```
import logging
from kafka import KafkaConsumer
import json
import psycopg2
from weather_summary import weather_summary

# Configure logging
logging.basicConfig(filename='consumer_log.txt', level=logging.DEBUG,
format='%(asctime)s - %(levelname)s - %(message)s')

# Kafka consumer settings
consumer = KafkaConsumer('weather-data',
group_id='weather-data-group',
bootstrap_servers=['localhost:9092'],
value_deserializer=lambda x: json.loads(x.decode('utf-8')))

# PostgreSQL connection settings
conn = psycopg2.connect(dbname='weather_app', user='postgres', password='admin@01',
host='localhost')
cur = conn.cursor()

if __name__ == "__main__":
for message in consumer:
weather_data = message.value
print("Received weather data:", weather_data)

# Insert weather data into PostgreSQL database
try:
cur.execute("""
INSERT INTO weather_detail (city_name, temperature, wind_speed, humidity)
VALUES (%s, %s, %s, %s)
""", (weather_data['city_name'], weather_data['current_temperature'],
weather_data['current_wind_speed'], weather_data['current_humidity']))
conn.commit()
print("The weather data were inserted into the database")
logging.info("Weather data inserted into PostgreSQL successfully!")
except Exception as e:
logging.error("Error inserting weather data into PostgreSQL: %s", str(e))
```

```
cur.close()
conn.close()

# After processing the data, call the weather_summary function to sumarize the final data
weather_summary()
```

weather_summary.py (to summarize the final data into the weather_summary table):

```
import pandas as pd
import psycopg2

def weather_summary():
try:
# PostgreSQL connection settings
conn = psycopg2.connect(dbname='weather_app', user='postgres',
password='admin@01', host='localhost')

# Query to fetch weather details
query = """
SELECT city_name, temperature, wind_speed, humidity
FROM weather_detail
"""

# Fetch data from PostgreSQL into a DataFrame
weather_df = pd.read_sql_query(query, conn)

# Process the data to calculate min and max values for each city
summary_df = weather_df.groupby('city_name').agg({
'temperature': ['min', 'max'],
'wind_speed': ['min', 'max'],
'humidity': ['min', 'max']
}).reset_index()

# Format the data as required
summary_df.columns = ['City', 'Min Temperature, deg C', 'Max Temperature,
deg C', 'Min Wind speed, km/h', 'Max Wind speed, km/h', 'Min Humidity, %',
'Max Humidity, %']
summary_df['Min Temperature, deg C'] = summary_df['Min Temperature, deg
C'].astype(str) + " to " + summary_df['Max Temperature, deg
C'].astype(str)
summary_df['Min Wind speed, km/h'] = summary_df['Min Wind speed,
km/h'].astype(str) + " to " + summary_df['Max Wind speed,
km/h'].astype(str)
```

```
summary_df['Min Humidity, %'] = summary_df['Min Humidity, %'].astype(str)
+ " to " + summary_df['Max Humidity, %'].astype(str)
summary_df.drop(columns=['Max Temperature, deg C', 'Max Wind speed, km/h',
'Max Humidity, %'], inplace=True)

# Insert the summarized data into the weather_summary table
with conn.cursor() as cur:
for _, row in summary_df.iterrows():
cur.execute("""
INSERT INTO weather_summary (City, Temperature_deg_c, Wind_speed_km_h,
Humidity_percentage)
VALUES (%s, %s, %s, %s)
""", (row['City'], row['Min Temperature, deg C'], row['Min Wind speed,
km/h'], row['Min Humidity, %']))
conn.commit()

# Close the database connection
conn.close()
except Exception as e:
print(f"An error occurred: {e}")

# Calling the funcion weather_summary to summarize the final data
weather_summary()
```