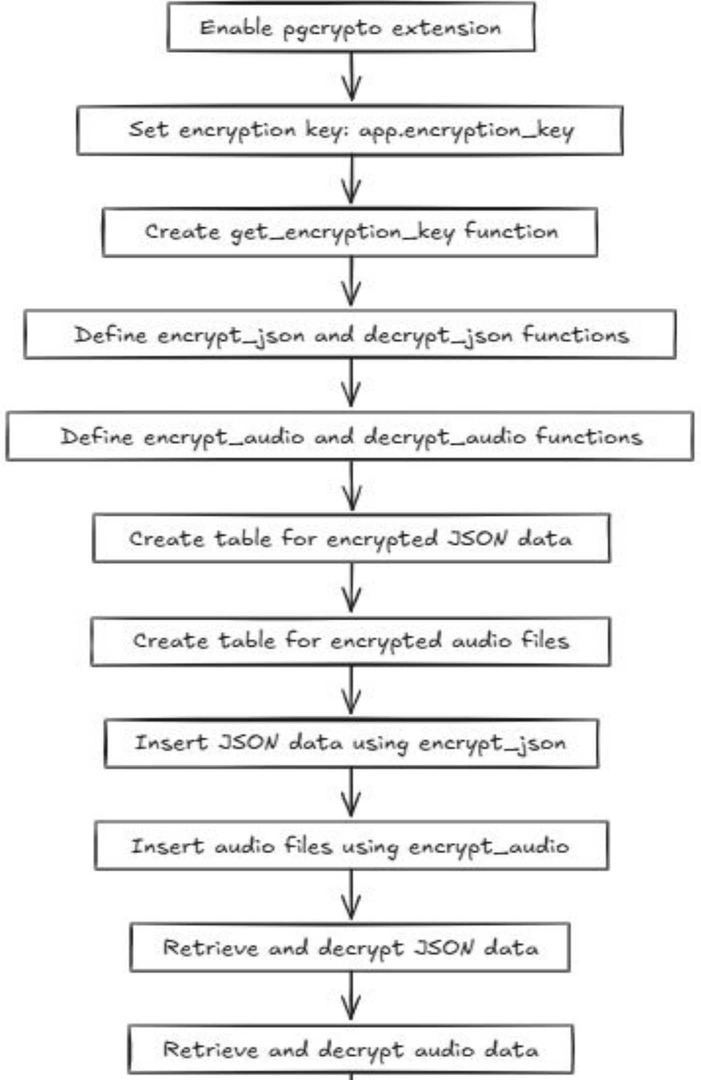
Importing JSON/Audio Data & Encryption - Decryption

Musab Ahmed Khan Umair

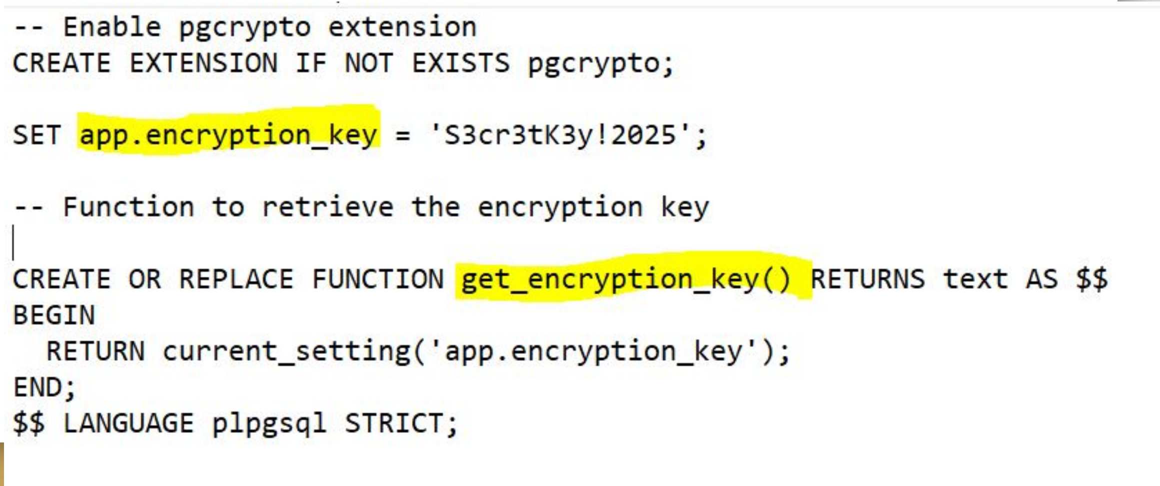
GithubLink For code - https://github.com/musabaku/pgCryptoPractice/blob/main/Sql%20code.txt

FlowChart


Enable pgcrypto extension
Set encryption key: app.encryption_key
Create get_encryption_key function
Define encrypt_json and decrypt_json functions
Define encrypt_audio and decrypt_audio functions
Create table for encrypted JSON data
Create table for encrypted audio files
Insert JSON data using encrypt_json
Insert audio files using encrypt_audio
Retrieve and decrypt JSON data
Retrieve and decrypt audio data

Configuring pgcrypto extension & Key

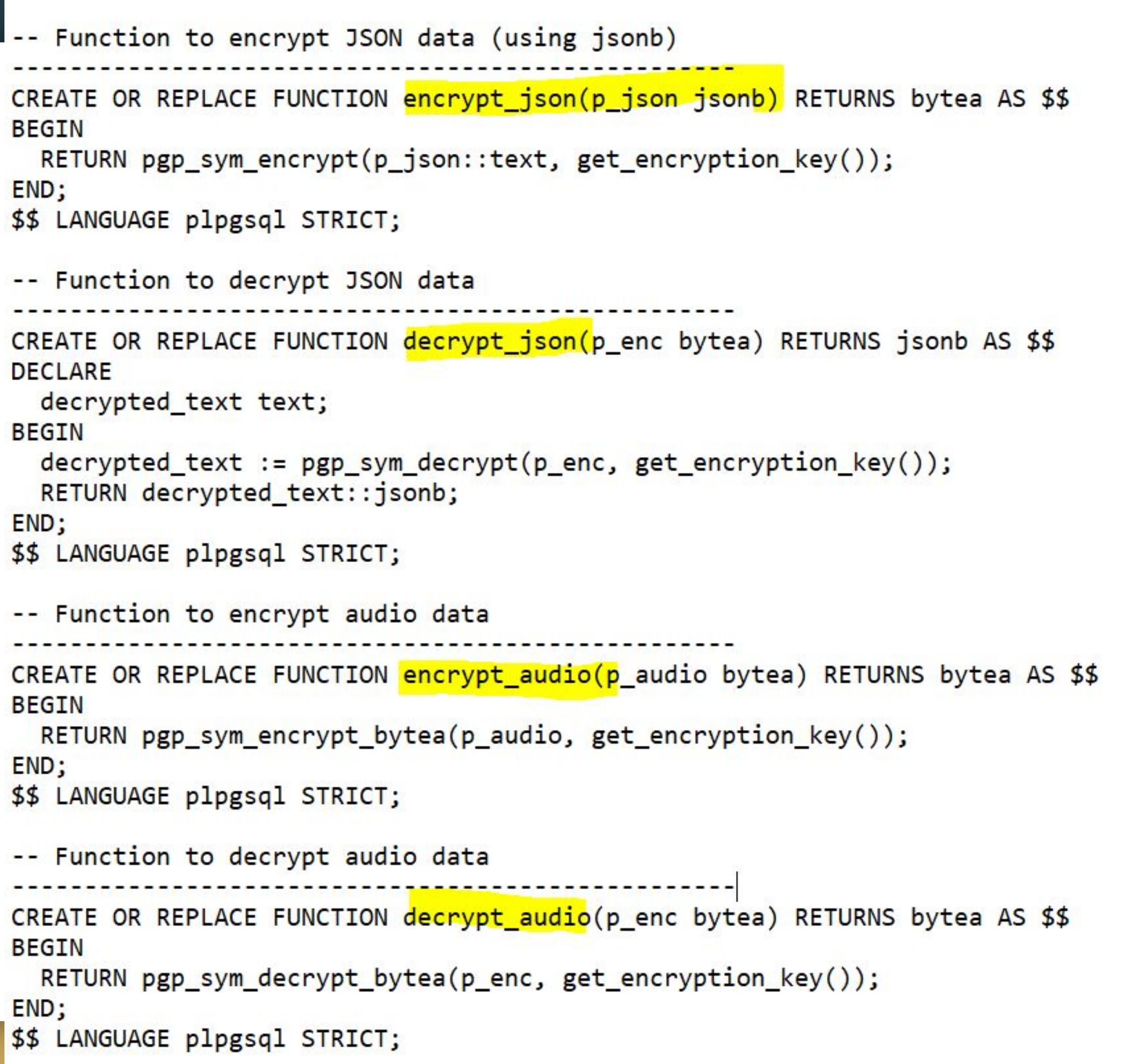
- I enabled the pgcrypto extension
- Next, I stored encryption key in (app.encryption_key)
- I created function get_encryption_key to retrieve the key



```
-- Enable pgcrypto extension
CREATE EXTENSION IF NOT EXISTS pgcrypto;

SET app.encryption_key = 'S3cr3tK3y!2025';

-- Function to retrieve the encryption key

CREATE OR REPLACE FUNCTION get_encryption_key() RETURNS text AS $$
BEGIN
  RETURN current_setting('app.encryption_key');
END;
$$ LANGUAGE plpgsql STRICT;
```

Encryption & Decryption Functions

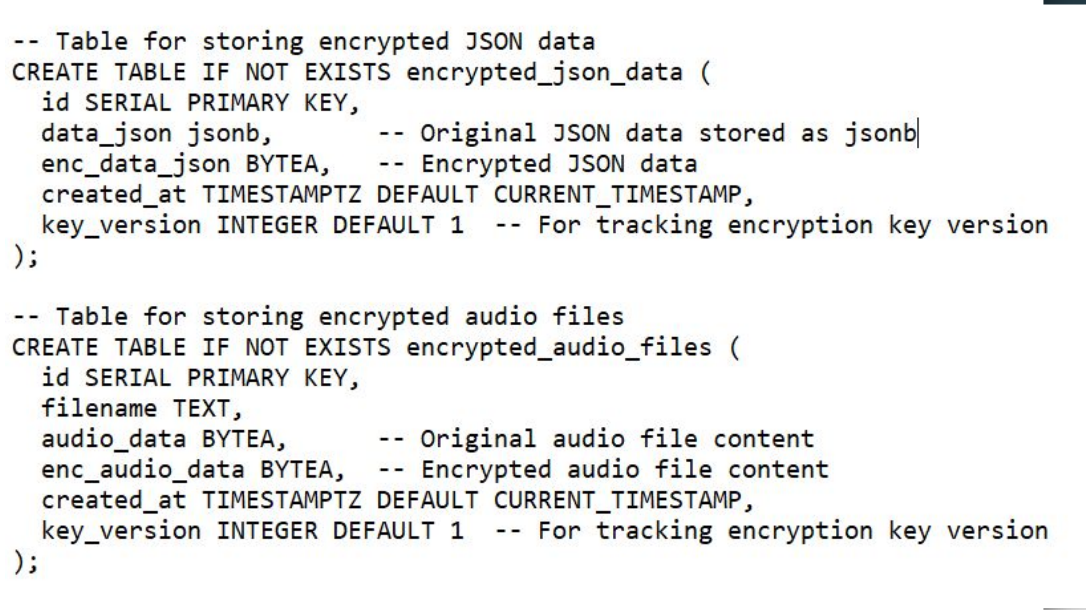
- JSON Functions:
  - encrypt_json: Converts JSONB to text and encrypts it
  - decrypt_json: Decrypts the bytea back to JSONB
- Audio Functions:
  - encrypt_audio: Encrypts binary audio data
  - decrypt_audio: Decrypts the binary data back



```
-- Function to encrypt JSON data (using jsonb)
-------------------------------------------------
CREATE OR REPLACE FUNCTION encrypt_json(p_json jsonb) RETURNS bytea AS $$
BEGIN
  RETURN pgp_sym_encrypt(p_json::text, get_encryption_key());
END;
$$ LANGUAGE plpgsql STRICT;

-- Function to decrypt JSON data
-------------------------------------------------
CREATE OR REPLACE FUNCTION decrypt_json(p_enc bytea) RETURNS jsonb AS $$
DECLARE
  decrypted_text text;
BEGIN
  decrypted_text := pgp_sym_decrypt(p_enc, get_encryption_key());
  RETURN decrypted_text::jsonb;
END;
$$ LANGUAGE plpgsql STRICT;

-- Function to encrypt audio data
-------------------------------------------------
CREATE OR REPLACE FUNCTION encrypt_audio(p_audio bytea) RETURNS bytea AS $$
BEGIN
  RETURN pgp_sym_encrypt_bytea(p_audio, get_encryption_key());
END;
$$ LANGUAGE plpgsql STRICT;

-- Function to decrypt audio data
-------------------------------------------------
CREATE OR REPLACE FUNCTION decrypt_audio(p_enc bytea) RETURNS bytea AS $$
BEGIN
  RETURN pgp_sym_decrypt_bytea(p_enc, get_encryption_key());
END;
$$ LANGUAGE plpgsql STRICT;
```

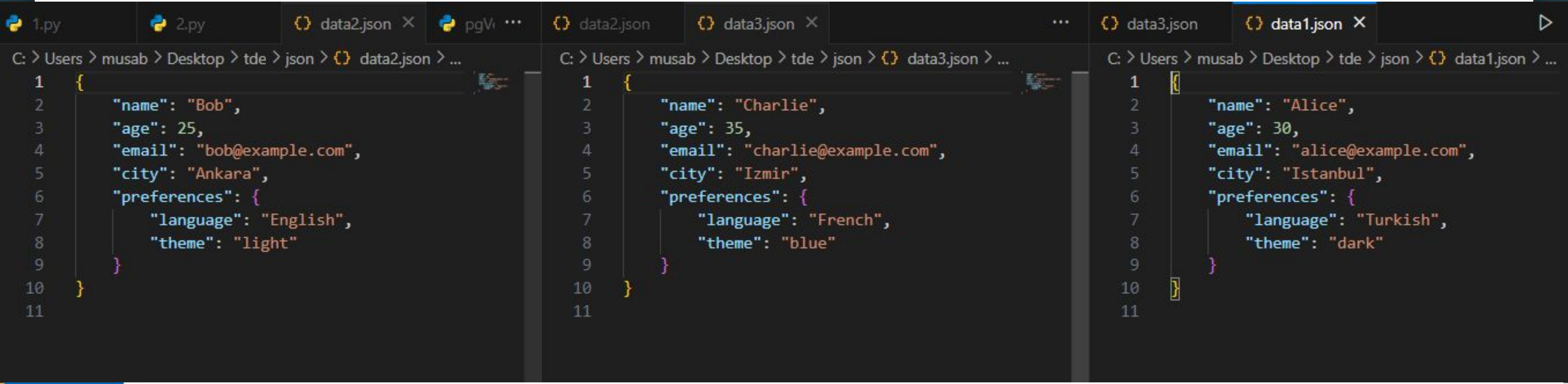
Creating Tables to store data

- JSON Table:
  - It stores, json data
- Audio Table:
  - It stores, audio data



```
-- Table for storing encrypted JSON data
CREATE TABLE IF NOT EXISTS encrypted_json_data (
  id SERIAL PRIMARY KEY,
  data_json jsonb,       -- Original JSON data stored as jsonb
  enc_data_json BYTEA,   -- Encrypted JSON data
  created_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP,
  key_version INTEGER DEFAULT 1  -- For tracking encryption key version
);

-- Table for storing encrypted audio files
CREATE TABLE IF NOT EXISTS encrypted_audio_files (
  id SERIAL PRIMARY KEY,
  filename TEXT,
  audio_data BYTEA,      -- Original audio file content
  enc_audio_data BYTEA,  -- Encrypted audio file content
  created_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP,
  key_version INTEGER DEFAULT 1  -- For tracking encryption key version
);
```

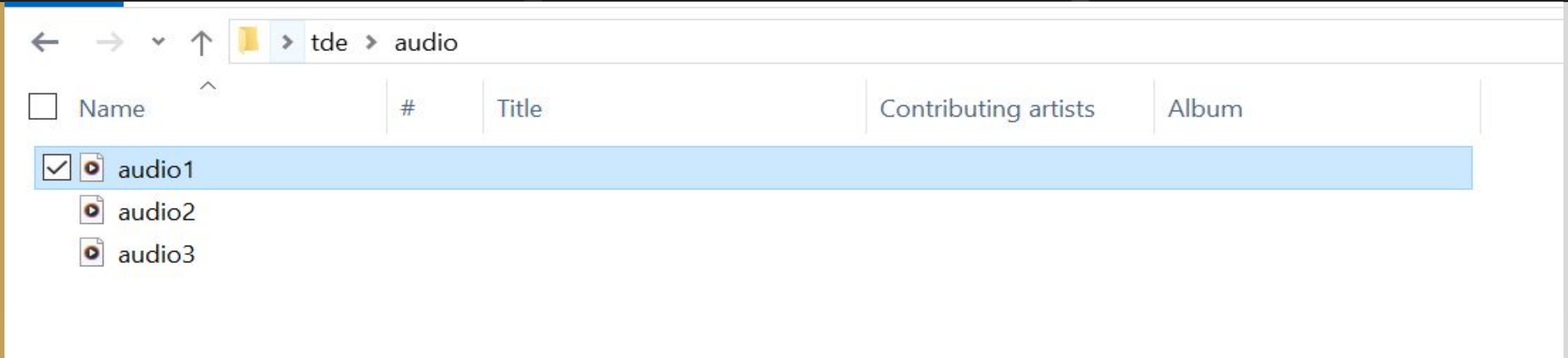
Json & Audio Data


1.py
2.py
data2.json
pgV
data2.json
data3.json
data3.json
data1.json
C: > Users > musab > Desktop > tde > json > data2.json > ...
C: > Users > musab > Desktop > tde > json > data3.json > ...
C: > Users > musab > Desktop > tde > json > data1.json > ...

```json
{
    "name": "Bob",
    "age": 25,
    "email": "bob@example.com",
    "city": "Ankara",
    "preferences": {
        "language": "English",
        "theme": "light"
    }
}
```

```json
{
    "name": "Charlie",
    "age": 35,
    "email": "charlie@example.com",
    "city": "Izmir",
    "preferences": {
        "language": "French",
        "theme": "blue"
    }
}
```

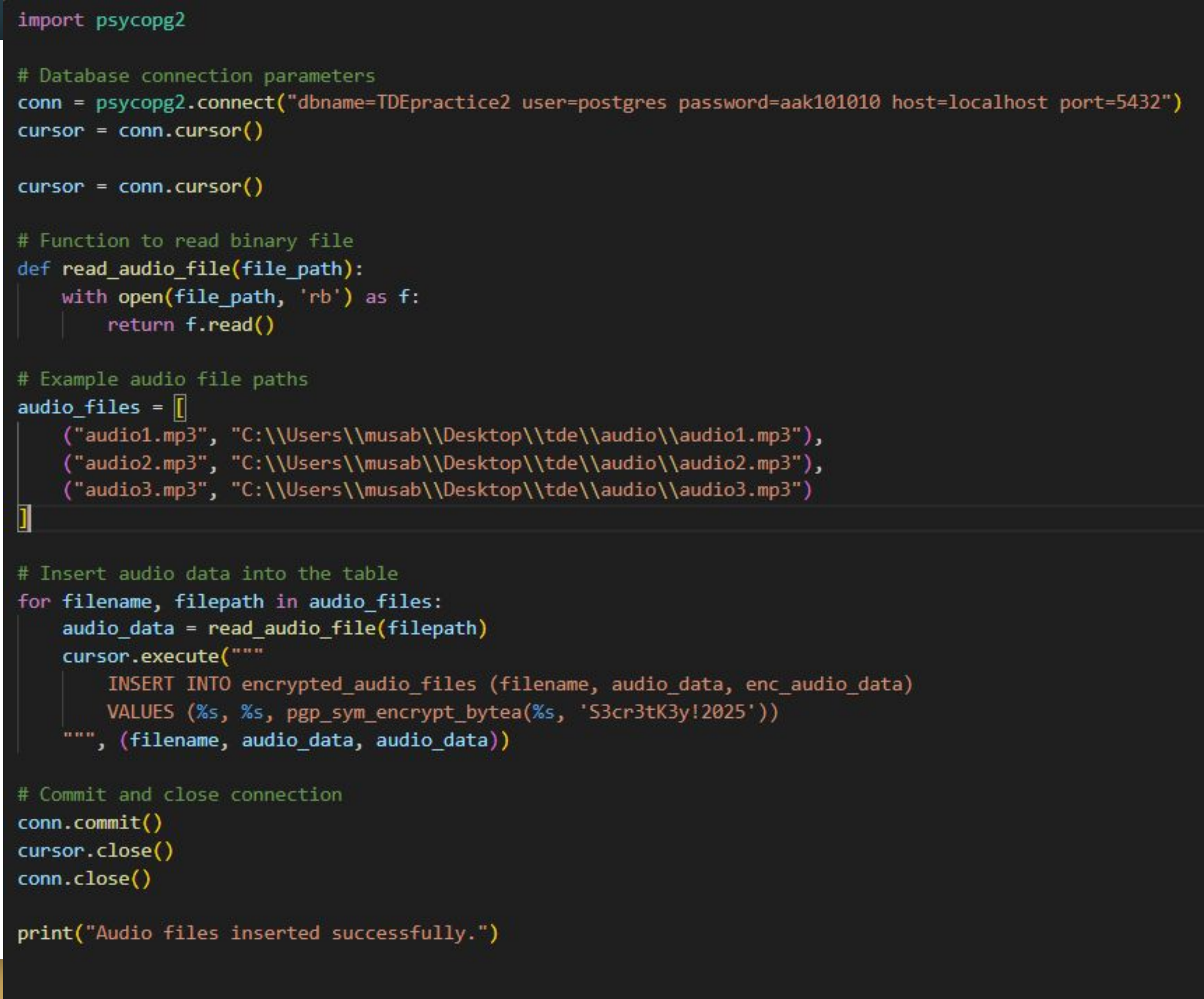
```json
{
    "name": "Alice",
    "age": 30,
    "email": "alice@example.com",
    "city": "Istanbul",
    "preferences": {
        "language": "Turkish",
        "theme": "dark"
    }
}
```



Name	#	Title	Contributing artists	Album
audio1				
audio2				
audio3				

Audio Data Insertion

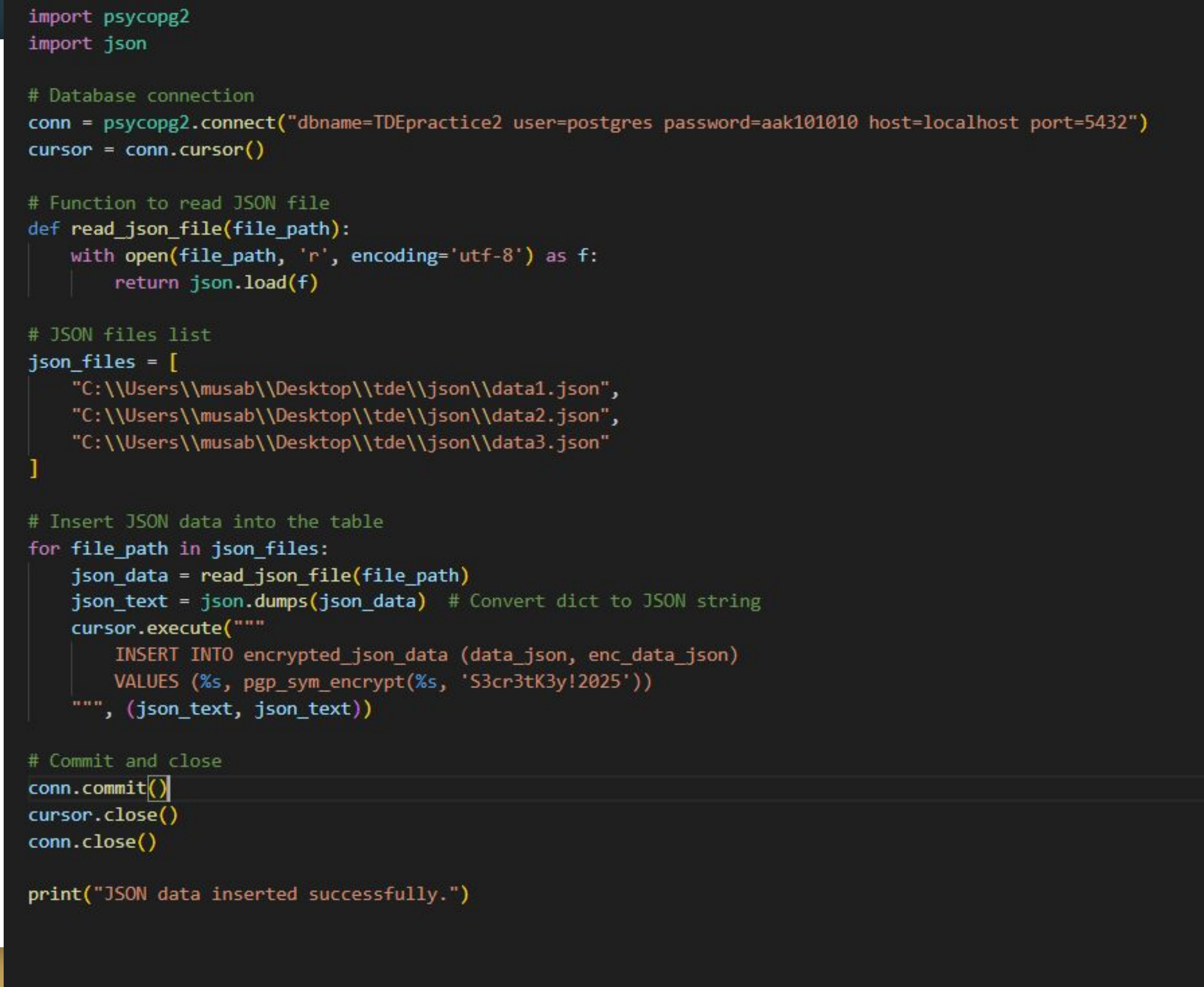
Insert audio data using encryption functions



```python
import psycopg2

# Database connection parameters
conn = psycopg2.connect("dbname=TDEpractice2 user=postgres password=aak101010 host=localhost port=5432")
cursor = conn.cursor()

cursor = conn.cursor()

# Function to read binary file
def read_audio_file(file_path):
    with open(file_path, 'rb') as f:
        return f.read()

# Example audio file paths
audio_files = [
    ("audio1.mp3", "C:\\Users\\musab\\Desktop\\tde\\audio\\audio1.mp3"),
    ("audio2.mp3", "C:\\Users\\musab\\Desktop\\tde\\audio\\audio2.mp3"),
    ("audio3.mp3", "C:\\Users\\musab\\Desktop\\tde\\audio\\audio3.mp3")
]

# Insert audio data into the table
for filename, filepath in audio_files:
    audio_data = read_audio_file(filepath)
    cursor.execute("""
        INSERT INTO encrypted_audio_files (filename, audio_data, enc_audio_data)
        VALUES (%s, %s, pgp_sym_encrypt_bytea(%s, 'S3cr3tK3y!2025'))
    """, (filename, audio_data, audio_data))

# Commit and close connection
conn.commit()
cursor.close()
conn.close()

print("Audio files inserted successfully.")
```

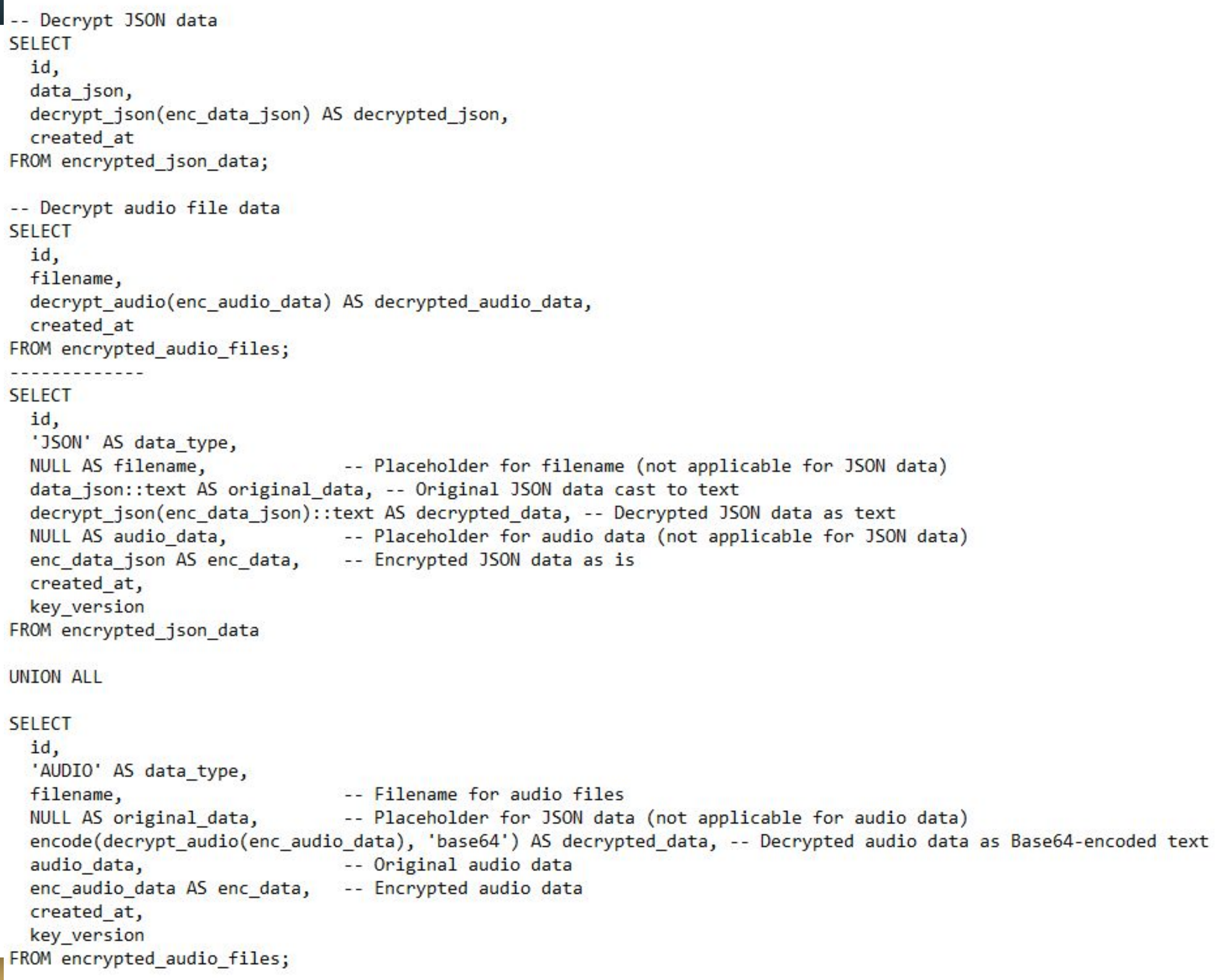
Json Data Insertion

Insert json data using encryption functions



```python
import psycopg2
import json

# Database connection
conn = psycopg2.connect("dbname=TDEpractice2 user=postgres password=aak101010 host=localhost port=5432")
cursor = conn.cursor()

# Function to read JSON file
def read_json_file(file_path):
    with open(file_path, 'r', encoding='utf-8') as f:
        return json.load(f)

# JSON files list
json_files = [
    "C:\\Users\\musab\\Desktop\\tde\\json\\data1.json",
    "C:\\Users\\musab\\Desktop\\tde\\json\\data2.json",
    "C:\\Users\\musab\\Desktop\\tde\\json\\data3.json"
]

# Insert JSON data into the table
for file_path in json_files:
    json_data = read_json_file(file_path)
    json_text = json.dumps(json_data)  # Convert dict to JSON string
    cursor.execute("""
        INSERT INTO encrypted_json_data (data_json, enc_data_json)
        VALUES (%s, pgp_sym_encrypt(%s, 'S3cr3tK3y!2025'))
    """, (json_text, json_text))

# Commit and close
conn.commit()
cursor.close()
conn.close()

print("JSON data inserted successfully.")
```

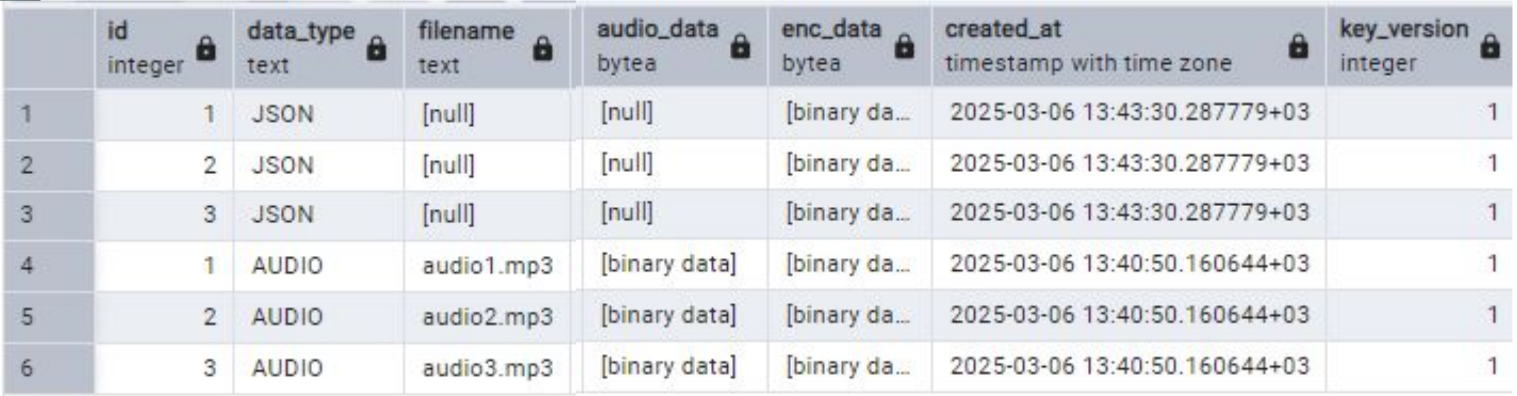
Queries For Retrieving data



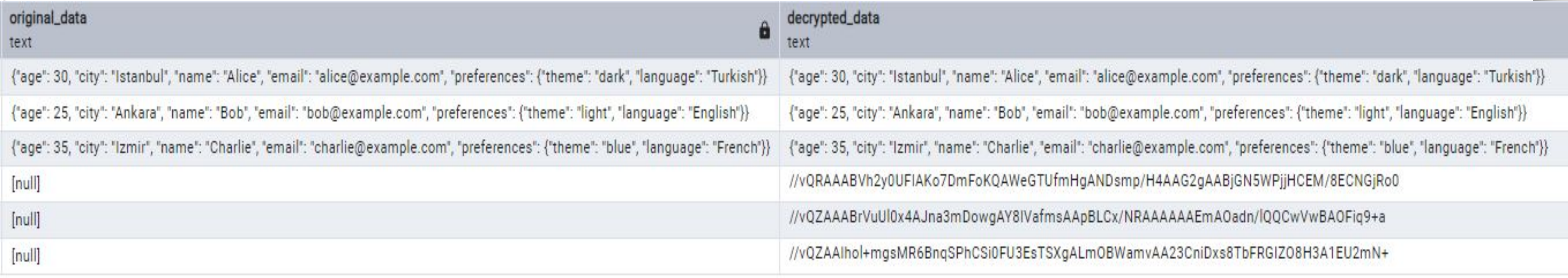
```
-- Decrypt JSON data
SELECT
  id,
  data_json,
  decrypt_json(enc_data_json) AS decrypted_json,
  created_at
FROM encrypted_json_data;

-- Decrypt audio file data
SELECT
  id,
  filename,
  decrypt_audio(enc_audio_data) AS decrypted_audio_data,
  created_at
FROM encrypted_audio_files;
-------------
SELECT
  id,
  'JSON' AS data_type,
  NULL AS filename,             -- Placeholder for filename (not applicable for JSON data)
  data_json::text AS original_data, -- Original JSON data cast to text
  decrypt_json(enc_data_json)::text AS decrypted_data, -- Decrypted JSON data as text
  NULL AS audio_data,           -- Placeholder for audio data (not applicable for JSON data)
  enc_data_json AS enc_data,    -- Encrypted JSON data as is
  created_at,
  key_version
FROM encrypted_json_data

UNION ALL

SELECT
  id,
  'AUDIO' AS data_type,
  filename,                     -- Filename for audio files
  NULL AS original_data,        -- Placeholder for JSON data (not applicable for audio data)
  encode(decrypt_audio(enc_audio_data), 'base64') AS decrypted_data, -- Decrypted audio data as Base64-encoded text
  audio_data,                   -- Original audio data
  enc_audio_data AS enc_data,   -- Encrypted audio data
  created_at,
  key_version
FROM encrypted_audio_files;
```

Results After Retrieving data

- For json table it shows null in audio data column.
- For audio table it shows null for json column



	id integer	data_type text	filename text	audio_data bytea	enc_data bytea	created_at timestamp with time zone	key_version integer
1	1	JSON	[null]	[null]	[binary da...	2025-03-06 13:43:30.287779+03	1
2	2	JSON	[null]	[null]	[binary da...	2025-03-06 13:43:30.287779+03	1
3	3	JSON	[null]	[null]	[binary da...	2025-03-06 13:43:30.287779+03	1
4	1	AUDIO	audio1.mp3	[binary data]	[binary da...	2025-03-06 13:40:50.160644+03	1
5	2	AUDIO	audio2.mp3	[binary data]	[binary da...	2025-03-06 13:40:50.160644+03	1
6	3	AUDIO	audio3.mp3	[binary data]	[binary da...	2025-03-06 13:40:50.160644+03	1



original_data text	decrypted_data text
{"age": 30, "city": "Istanbul", "name": "Alice", "email": "alice@example.com", "preferences": {"theme": "dark", "language": "Turkish"}}	{"age": 30, "city": "Istanbul", "name": "Alice", "email": "alice@example.com", "preferences": {"theme": "dark", "language": "Turkish"}}
{"age": 25, "city": "Ankara", "name": "Bob", "email": "bob@example.com", "preferences": {"theme": "light", "language": "English"}}	{"age": 25, "city": "Ankara", "name": "Bob", "email": "bob@example.com", "preferences": {"theme": "light", "language": "English"}}
{"age": 35, "city": "Izmir", "name": "Charlie", "email": "charlie@example.com", "preferences": {"theme": "blue", "language": "French"}}	{"age": 35, "city": "Izmir", "name": "Charlie", "email": "charlie@example.com", "preferences": {"theme": "blue", "language": "French"}}
[null]	//vQRAAABVh2y0UFIAKo7DmFoKQAWeGTUfmHgANDsmp/H4AAG2gAABjGN5WPjjHCEM/8ECNGjRo0
[null]	//vQZAAABrVuUl0x4AJna3mDowgAY8IVafmsAApBLCx/NRAAAAAAEmAOadn/lQQCwVwBAOFiq9+a
[null]	//vQZAAIhol+mgsMR6BnqSPhCSi0FU3EsTSXgALmOBWamvAA23CniDxs8TbFRGIZO8H3A1EU2mN+