NextGenLead_6

PROJECT OVERVIEW

- **Goal**: Build a system where dealers manage opportunities (customers) and keep track of information like opportunity details, account names, close dates, amounts, etc.
- **Requirements**:
  - Create the **Account**, **Dealer**, and **Opportunity** tables manually in PostgreSQL using SQL scripts.
  - Handle all payloads in the **POST** method in the Flask API.
  - Validate data when interacting with the database (e.g., check if account and dealer exist before creating an opportunity).

STEP 1: POSTGRESQL TABLE CREATION SCRIPTS

YOU WILL CREATE THREE TABLES: `ACCOUNT`, `DEALER`, AND `OPPORTUNITY`. THESE TABLES WILL BE CREATED MANUALLY USING THE FOLLOWING SQL SCRIPTS.

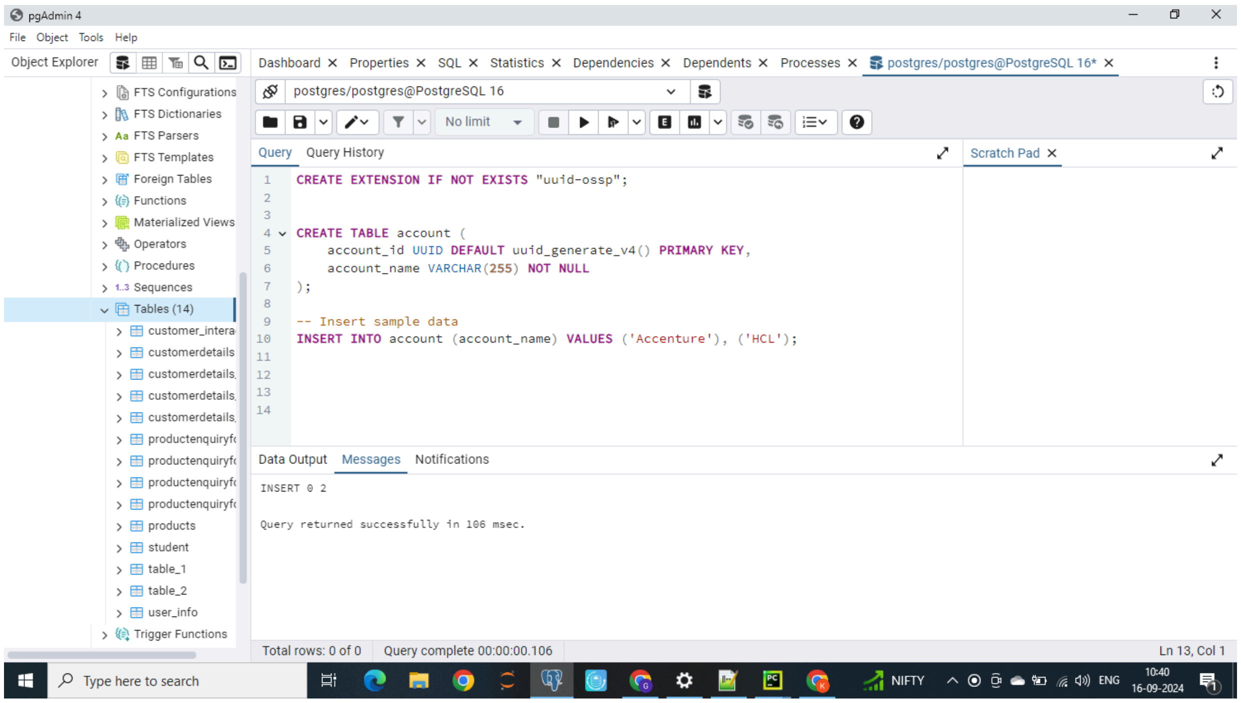
Create `account` Table

```
CREATE EXTENSION IF NOT EXISTS "uuid-ossp";

CREATE TABLE account (
                        account_id UUID DEFAULT uuid_generate_v4() PRIMARY KEY,
                        account_name VARCHAR(255) NOT NULL
                       );

-- Insert sample data
INSERT INTO account (account_name) VALUES ('Accenture'), ('HCL');
```

`uuid_generate_v4()` is a function provided by the `uuid-ossp` extension, but the extension is not installed or enabled by default on every PostgreSQL installation.



```
CREATE EXTENSION IF NOT EXISTS "uuid-ossp";


CREATE TABLE account (
    account_id UUID DEFAULT uuid_generate_v4() PRIMARY KEY,
    account_name VARCHAR(255) NOT NULL
);

-- Insert sample data
INSERT INTO account (account_name) VALUES ('Accenture'), ('HCL');
```

```
INSERT 0 2

Query returned successfully in 106 msec.
```

Create `dealer` Table

```
CREATE TABLE dealer (
    dealer_id UUID DEFAULT uuid_generate_v4() PRIMARY KEY,
    dealer_code VARCHAR(50) NOT NULL,
    opportunity_owner VARCHAR(255) NOT NULL
);

-- Insert sample data
INSERT INTO dealer (dealer_code, opportunity_owner)
VALUES ('CH12', 'Komal Sai'),
       ('CH12', 'Dinesh'),
       ('BL04', 'Mahesh'),
       ('HY01', 'Sainath');
```

Create `opportunity` Table

```
CREATE TABLE opportunity (
    opportunity_id UUID DEFAULT uuid_generate_v4() PRIMARY KEY,
    opportunity_name VARCHAR(255),
    account_id UUID REFERENCES account(account_id),
    close_date DATE,
    amount DECIMAL(10, 2),
    description TEXT,
    dealer_id UUID REFERENCES dealer(dealer_id),
    dealer_code VARCHAR(50),
    dealer_name_or_opportunity_owner VARCHAR(255),
    stage VARCHAR(50),
    probability INTEGER,
    next_step VARCHAR(255),
    created_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

STEP 2: PYTHON (FLASK) PROJECT SETUP

Folder Structure:

```
project/
│
├── app.py                # Main entry point for the Flask application
├── models.py             # SQLAlchemy models for the tables
├── database.py           # Database connection setup
└── requirements.txt      # Python dependencies
```

Step 3: Setup Virtual Environment & Install Dependencies

Create a `requirements.txt` file:

```
Flask==2.3.2
SQLAlchemy==2.0.0
psycopg2-binary==2.9.3
```

To install the dependencies later:

```
pip install -r requirements.txt
```

Step 4: Database Connection (`database.py`)

```
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker

# PostgreSQL connection URL
DATABASE_URL = "postgresql://postgres:1234@localhost:5432/postgres"

# Create engine and session
engine = create_engine(DATABASE_URL, echo=True)
Session = sessionmaker(bind=engine)
session = Session()
```

Step 5: SQLAlchemy Models (`models.py`)

```
from sqlalchemy import Column, String, Date, Text, DECIMAL, Integer, TIMESTAMP, Foreig
from sqlalchemy.ext.declarative import declarative_base
import uuid
from datetime import datetime

Base = declarative_base()

# Define the Account model
class Account(Base):
    __tablename__ = 'account'
    account_id = Column(String, primary_key=True, default=lambda: str(uuid.uuid4()))
    account_name = Column(String(255), nullable=False)

# Define the Dealer model
class Dealer(Base):
    __tablename__ = 'dealer'
    dealer_id = Column(String, primary_key=True, default=lambda: str(uuid.uuid4()))
    dealer_code = Column(String(50), nullable=False)
    opportunity_owner = Column(String(255), nullable=False)

# Define the Opportunity model
class Opportunity(Base):
    __tablename__ = 'opportunity'
    opportunity_id = Column(String, primary_key=True, default=lambda: str(uuid.uuid4()
    opportunity_name = Column(String(255))
    account_id = Column(String, ForeignKey('account.account_id'))
    close_date = Column(Date)
    amount = Column(DECIMAL(10, 2))
    description = Column(Text)
    dealer_id = Column(String, ForeignKey('dealer.dealer_id'))
    dealer_code = Column(String(50))
    dealer_name_or_opportunity_owner = Column(String(255))
    stage = Column(String(50))
    probability = Column(Integer)
    next_step = Column(String(255))
    created_date = Column(TIMESTAMP, default=datetime.utcnow)
```

STEP 6: FLASK APPLICATION (`APP.PY`)

THIS IS THE MAIN FILE FOR YOUR FLASK APPLICATION, WHICH WILL HANDLE API REQUESTS.

```
from flask import Flask, request, jsonify
from sqlalchemy.orm import sessionmaker
from database import engine, session
from models import Account, Dealer, Opportunity
import uuid
from datetime import datetime

app = Flask(__name__)

# POST: Create a new customer (opportunity)
@app.route('/new_customer', methods=['POST'])
def create_new_customer():
    payload = request.get_json()
    # Validate account_name in the account table
    account = session.query(Account).filter_by(account_name=payload.get('account_name'
    if not account:
        return jsonify({"error": "Account does not exist"}), 400

    # Validate dealer information in the dealer table
    dealer = session.query(Dealer).filter_by(dealer_id=payload.get('dealer_id'),
                                             dealer_code=payload.get('dealer_code'),
                                             opportunity_owner=payload.get('dealer_nam
    if not dealer:
        return jsonify({"error": "Dealer does not exist"}), 400

    # Insert new opportunity record
    new_opportunity = Opportunity(
        opportunity_name=payload.get('opportunity_name'),
        account_id=account.account_id,
        close_date=payload.get('close_date'),
        amount=payload.get('amount'),
        description=payload.get('description'),
        dealer_id=dealer.dealer_id,
        dealer_code=payload.get('dealer_code'),
        dealer_name_or_opportunity_owner=payload.get('dealer_name_or_opportunity_owner
        stage=payload.get('stage'),
        probability=payload.get('probability'),
        next_step=payload.get('next_step'),
        created_date=datetime.now()
    )

    session.add(new_opportunity)
    session.commit()

    return jsonify({"message": "Customer (opportunity) created successfully", "opportu

# GET: Retrieve all customers for a dealer
@app.route('/get_customers', methods=['GET'])
def get_customers():
    dealer_id = request.args.get('dealer_id')
    dealer_code = request.args.get('dealer_code')
    opportunity_owner = request.args.get('opportunity_owner')

    # Validate dealer information
    dealer = session.query(Dealer).filter_by(dealer_id=dealer_id, dealer_code=dealer_c
    if not dealer:
        return jsonify({"error": "Dealer does not exist"}), 401

    # Fetch opportunities for the given dealer
    opportunities = session.query(Opportunity).filter_by(dealer_code=dealer_code).all(

    return jsonify([{
        "opportunity_name": opp.opportunity_name,
        "account_id": opp.account_id,
        "close_date": opp.close_date,
        "amount": opp.amount,
        "description": opp.description,
        "stage": opp.stage
    } for opp in opportunities])

if __name__ == '__main__':
    app.run(debug=True)
```

STEP 7: TESTING THE APPLICATION

1. **Run the Flask app**:

Test the POST method to create a new customer:
Endpoint: `http://localhost:5000/new_customer`

Payload:Post method

```
{
  "opportunity_name": "Dinesh Sai",
  "account_name": "Accenture",
  "close_date": "2024-09-20",
  "amount": 95000.00,
  "description": "Dinesh interested in RX100 bike",
  "dealer_id": "4354",
  "dealer_code": "CH12",
  "dealer_name_or_opportunity_owner": "Komal Sai",
  "stage": "Propose",
  "probability": 50,
  "next_step": "Follow Up"
}
```

```
http://127.0.0.1:5000/get_customers?dealer_id=07573256-4786-4542-a4f0-f4c9834082be&dea
```

Get Method end points