**Step 1: Install MySQL Connector for Python**

Use this in your terminal or CMD:

bash

pip install mysql-connector-python

This lets your Python app talk to MySQL databases.

**🧠 Step 2: Write a Connection Module**

Create a new file in your Brokerage/utils/ folder called db.py and add:

python

import mysql.connector

def get\_connection():

return mysql.connector.connect(

host="localhost", # or your cloud DB host

user="your\_username",

password="your\_password",

database="brokerage\_db"

)

This function returns a live connection to your database.

**🧱 Step 3: Use SQL in Your App**

Anywhere in your app (e.g. client.py, main.py), you can now do:

python

from Brokerage.utils.db import get\_connection

conn = get\_connection()

cursor = conn.cursor()

# Example: Insert a new customer

cursor.execute("""

INSERT INTO customers (name, contact, flat\_type, amount, brokerage)

VALUES (%s, %s, %s, %s, %s)

""", (custName, contactNo, flatType, amount, brokerageAmount))

conn.commit()

cursor.close()

conn.close()

You can also write queries to retrieve, update, or delete records.

**☁️ Step 4: Prepare for Cloud Deployment**

When you're ready to deploy online (e.g. on a cloud server or web host):

* Replace localhost with your cloud database host (e.g. AWS RDS, Azure MySQL, etc.)
* Make sure your cloud DB allows remote connections (port 3306 open)
* Use environment variables to store credentials securely:

python

import os

from dotenv import load\_dotenv

load\_dotenv()

conn = mysql.connector.connect(

host=os.getenv("DB\_HOST"),

user=os.getenv("DB\_USER"),

password=os.getenv("DB\_PASS"),

database=os.getenv("DB\_NAME")

)

**🧪 Step 5: Test Locally Before Deployment**

Run your app locally and confirm:

* Customers are saved to the database
* Brokerage amounts are retrieved correctly
* Transactions and portfolios are stored and updated

Once that’s solid, you’re ready to deploy with confidence.

If you want help writing the SQL schema for your customers table, or setting up .env files for secure credentials, I can walk you through it. You’re building a real-world system

### ✅ Local Database Flow

If your MySQL server is installed on your machine (localhost):

1. You run your app with:

bash

python Brokerage/main.py

1. You input customer details.
2. Your app connects to MySQL using:

python

host="localhost"

1. It executes SQL queries to insert or retrieve data.
2. The data is stored instantly in your local MySQL database.

### ☁️ Cloud Database Flow

If your MySQL database is hosted in the cloud (e.g. AWS RDS, Azure, or another provider):

1. You update your connection settings:

python

host="your-cloud-db-endpoint"

1. You run your app locally.
2. Your app connects to the cloud database over the internet.
3. Data is stored remotely — just like it would be locally.

### 🔐 Important Notes for Cloud Access

* Make sure your cloud database allows **remote connections** (port 3306 open).
* Use **environment variables** or a .env file to store credentials securely.
* Test the connection with a simple query before integrating it fully.

### 🧪 Example Integration

In your client.py or wherever you handle customer creation:

python

from Brokerage.utils.db import get\_connection

def save\_customer\_to\_db(customer):

conn = get\_connection()

cursor = conn.cursor()

cursor.execute("""

INSERT INTO customers (name, contact, flat\_type, amount, brokerage)

VALUES (%s, %s, %s, %s, %s)

""", (customer.custName, customer.contactNo, customer.flatType, customer.amount, customer.brokerageAmount))

conn.commit()

cursor.close()

conn.close()

Call this function right after adding a customer to your list.

So yes — once the connection is set, your app becomes a live data pipeline. Whether it’s local or cloud, Python handles it seamlessly. Let me know if you want help writing the SQL schema or testing the connection. You’re ready to go full-stack with this.