There are several **cloud database services** that offer **free tiers** that are suitable for small-scale or development applications. These free tiers are typically limited in terms of storage, number of connections, or performance but are perfectly fine for small projects or prototypes. Below, I’ll list some of the most popular free cloud database providers and guide you through how to set them up and deploy them with your Flask app.

**1. Supabase (PostgreSQL)**

**Supabase** is a free, open-source alternative to Firebase that provides a managed PostgreSQL database with RESTful APIs, real-time functionality, and authentication. It offers a **generous free tier** that is ideal for small apps.

**Features of Supabase Free Tier:**

* **1 GB storage** for the database.
* **500 MB storage** for file storage.
* **1,000 monthly active users**.
* **Unlimited API requests**.
* **Real-time capabilities**.

**How to Set Up Supabase:**

1. **Sign Up for Supabase**:
   * Go to [Supabase](https://supabase.io/) and sign up.
   * After logging in, click **"New Project"**.
   * Enter a project name, choose a password for your database, and select a region.
2. **Get the Database Connection URL**:
   * Once the project is created, go to the **Database** tab.
   * Click on **Settings > Database** to find the connection string (DATABASE\_URL).
   * This connection string will look something like this:
   * postgres://username:password@host:5432/database\_name
3. **Set Environment Variables**:
   * In the Supabase dashboard, navigate to **Settings > API** to get the credentials for connecting to the database.
   * You'll need to add the DATABASE\_URL environment variable to your Flask app.
4. **Install Dependencies**: In your Flask project, install the necessary dependencies:
5. pip install Flask psycopg2-binary
6. **Connect to Supabase Database**: Use the connection string to set up a connection in your Flask app (as shown in previous examples). Store the DATABASE\_URL as an environment variable.
7. **Deploy to Vercel**:
   * Push your code to GitHub and link it to Vercel.
   * In **Vercel Settings > Environment Variables**, add your DATABASE\_URL as an environment variable.
   * Deploy your app and start using the database!

**2. ElephantSQL (PostgreSQL)**

**ElephantSQL** offers a **free plan** with a small PostgreSQL database, which is suitable for development and small-scale applications.

**Features of ElephantSQL Free Tier:**

* **20 MB of storage**.
* **Up to 5,000 rows**.
* **1 connection** allowed at a time.

**How to Set Up ElephantSQL:**

1. **Sign Up for ElephantSQL**:
   * Go to [ElephantSQL](https://www.elephantsql.com/) and sign up for a free account.
   * After logging in, click on **Create New Instance**.
   * Select the **Little Elephant** plan (which is the free tier).
   * Give your instance a name, and select a region (usually defaults to US or EU).
2. **Get the Database Connection URL**:
   * After creating the instance, you'll be taken to a dashboard with your instance details.
   * Look for the **URL** under **Connection Info**. This will look like:
   * postgres://username:password@host:5432/database\_name
3. **Install Dependencies**: In your Flask project, install the required libraries:
4. pip install Flask psycopg2-binary
5. **Connect to ElephantSQL Database**: Use the connection string to configure your Flask app as shown in the previous examples.
6. **Deploy to Vercel**:
   * Link your project to GitHub.
   * Add the DATABASE\_URL environment variable in **Vercel > Settings > Environment Variables**.
   * Deploy your app, and it will connect to the ElephantSQL database!

**3. PlanetScale (MySQL)**

**PlanetScale** offers a **free MySQL database** in the cloud, which is a great option for applications that require MySQL.

**Features of PlanetScale Free Tier:**

* **5 databases**.
* **1 database** with **up to 1 GB storage**.
* **Unlimited connections** (no connection limits).
* **No downtime** or disruptions.

**How to Set Up PlanetScale:**

1. **Sign Up for PlanetScale**:
   * Go to [PlanetScale](https://planetscale.com/) and create a free account.
   * Once logged in, create a new **database** and choose a region.
2. **Create a Database and Get Connection Information**:
   * After creating the database, go to the **Credentials** tab and copy the **connection string** (e.g., mysql://username:password@host:3306/database\_name).
3. **Install Dependencies**: In your Flask project, install the **MySQL** client:
4. pip install Flask mysql-connector-python
5. **Connect to PlanetScale Database**: Use the connection string to set up your Flask app:
6. import mysql.connector
7. import os
8. from flask import Flask
9. app = Flask(\_\_name\_\_)
10. db\_url = os.getenv('DATABASE\_URL') # e.g., mysql://username:password@host:3306/database\_name
11. conn = mysql.connector.connect(
12. host=db\_url,
13. user='your-username',
14. password='your-password',
15. database='your-database-name'
16. )
17. @app.route('/')
18. def home():
19. return "Connected to PlanetScale MySQL Database"
20. if \_\_name\_\_ == '\_\_main\_\_':
21. app.run(debug=True)
22. **Deploy to Vercel**:
    * Push your code to GitHub.
    * Add the DATABASE\_URL environment variable to **Vercel** settings.
    * Deploy your app.

**4. MongoDB Atlas (MongoDB)**

**MongoDB Atlas** is a cloud service for **MongoDB**, providing a free tier for small projects. This is ideal if you're looking for a **NoSQL** database.

**Features of MongoDB Atlas Free Tier:**

* **512 MB** storage.
* **Shared clusters** with **NoSQL** flexibility.
* **1 GB** of data transfer per month.
* Suitable for small apps or prototypes.

**How to Set Up MongoDB Atlas:**

1. **Sign Up for MongoDB Atlas**:
   * Go to [MongoDB Atlas](https://www.mongodb.com/cloud/atlas) and sign up for a free account.
   * Create a **new cluster** (select the **M0 Free Tier** plan).
   * Choose a cloud provider (e.g., AWS, GCP) and a region.
2. **Create a Database and Get Connection Info**:
   * Once the cluster is created, go to **Database Access > Create New Database User** and create a user with read and write permissions.
   * Go to **Clusters > Connect > Connect Your Application** and copy the **connection string**. It will look like this:
   * mongodb+srv://<username>:<password>@cluster0.mongodb.net/test?retryWrites=true&w=majority
3. **Install Dependencies**: Install the necessary libraries for Flask to connect to MongoDB:
4. pip install Flask pymongo
5. **Connect to MongoDB Atlas**: Here’s how to connect to your MongoDB Atlas database in your Flask app:
6. import os
7. from flask import Flask
8. from pymongo import MongoClient
9. app = Flask(\_\_name\_\_)
10. db\_url = os.getenv('DATABASE\_URL') # MongoDB Atlas connection string
11. client = MongoClient(db\_url)
12. db = client.get\_database()
13. @app.route('/')
14. def home():
15. collection = db.my\_collection # Reference to a collection in MongoDB
16. data = collection.find\_one() # Query the collection
17. return f"Hello, MongoDB! Data: {data}"
18. if \_\_name\_\_ == '\_\_main\_\_':
19. app.run(debug=True)
20. **Deploy to Vercel**:
    * Push your code to GitHub.
    * Add the DATABASE\_URL environment variable in **Vercel > Settings > Environment Variables**.
    * Deploy your app.

**5. Heroku Postgres (PostgreSQL)**

**Heroku** offers a free tier for PostgreSQL through their **Heroku Postgres** service. The **Hobby Dev** plan is free and provides a limited number of rows and storage.

**Features of Heroku Free Tier:**

* **Up to 1,000 rows**.
* **Free dyno (app server)** with **low monthly hours**.
* **Free PostgreSQL database**.

**How to Set Up Heroku Postgres:**

1. **Sign Up for Heroku**:
   * Go to [Heroku](https://www.heroku.com/) and create a free account.
   * Create a new **Heroku app**.
2. **Add Heroku Postgres Add-on**:
   * In the Heroku dashboard, go to your app’s **Resources** tab.
   * Search for **Heroku Postgres** and add the **Hobby Dev** plan.
3. **Get Database Connection Information**:
   * After