Update the blog website to use a MySQL database instead of the default SQLite.

To update your Django blog website to use a MySQL database instead of the default SQLite, follow these steps:

Step 1: Install MySQL and MySQL Client for Python

1. **Install MySQL Server:**
   - If you don't have MySQL installed, download and install it from the MySQL website.
   - During installation, note the username (usually `root`) and password you set up for the MySQL server.
2. **Install MySQL Client:**
   - Install the MySQL client for Python using pip:

```
pip install mysqlclient
```

Step 2: Create a MySQL Database for Your Project

1. **Access the MySQL Command Line:**
   - Open your terminal/command prompt and log in to the MySQL server:

```
mysql -u root -p
```

   - Enter your MySQL root password when prompted.
2. **Create a New Database:**
   - Run the following command to create a new database for your Django project (replace `blog_db` with your desired database name):

sql
```
CREATE DATABASE blog_db;
```

   - You can also create a specific user for the database and grant it privileges:

sql

```
CREATE USER 'blog_user'@'localhost' IDENTIFIED BY
'your_password';
GRANT ALL PRIVILEGES ON blog_db.* TO 'blog_user'@'localhost';
FLUSH PRIVILEGES;
```

Step 3: Update Django Settings

1. **Update `settings.py` to Use MySQL:**
   - Open your Django project's `settings.py` file and update the `DATABASES` configuration to use MySQL:

```
python
```

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'blog_db',  # your database name
        'USER': 'blog_user',  # your database user
        'PASSWORD': 'your_password',  # your database password
        'HOST': 'localhost',
        'PORT': '3306',
    }
}
```

Step 4: Apply Migrations

1. **Create Initial Migrations for Your App:**
   - If you haven't created migrations yet, run the following command:

```
python manage.py makemigrations
```

2. **Apply the Migrations to Your MySQL Database:**
   - Run the following command to apply the migrations:

```
python manage.py migrate
```

Step 5: Create a Superuser (If Not Already Done)

1. **Create a Superuser Account:**
   - If you haven't created a superuser yet, run the following command:

```
python manage.py createsuperuser
```

Step 6: Test the Application

1. **Run the Development Server:**
   - Start the development server:

```
python manage.py runserver
```

2. **Access Your Application:**
   - Navigate to `http://127.0.0.1:8000/` to see your blog website.
   - Access the admin interface at `http://127.0.0.1:8000/admin/` and log in with the superuser account.

Step 7: Verify the Database Change

1. **Verify Data Storage:**
   - Check your MySQL database to verify that the Django models and data are stored in the MySQL database.
   - You can use MySQL Workbench or any other MySQL client tool to connect to your MySQL server and inspect the `blog_db` database.

By following these steps, you will have updated your Django blog website to use a MySQL database instead of the default SQLite. This setup provides a more robust and scalable database solution for your application.