5. QUESTION

[AMBITIOUS]

Multi-tenant architecture with single instance of application layer which will have separate database for each client. Expected the architecture and approach document as output.

Source Code GIT Repository:

Web Browser Link:

https://github.com/Ajay-Kumar-Aspiring-Minds-Round-2/MultiTenantApp

GIT Clone Link:

https://github.com/Ajay-Kumar-Aspiring-Minds-Round-2/MultiTenantApp.git

Technology Stack:

What is Multi-tenancy?

- Multi-tenancy is an architecture in which a single instance of a software application serves multiple customers. Each customer is called a tenant.
- We can implement multi-tenancy using any of the following approaches:
  1. **Database per Tenant:** Each Tenant has its own database and is isolated from other tenants.
  2. **Shared Database, Shared Schema:** All Tenants share a database and tables. Every table has a Column with the Tenant Identifier that shows the owner of the row.
  3. **Shared Database, Separate Schema:** All Tenants share a database, but have their own database schemas and tables.

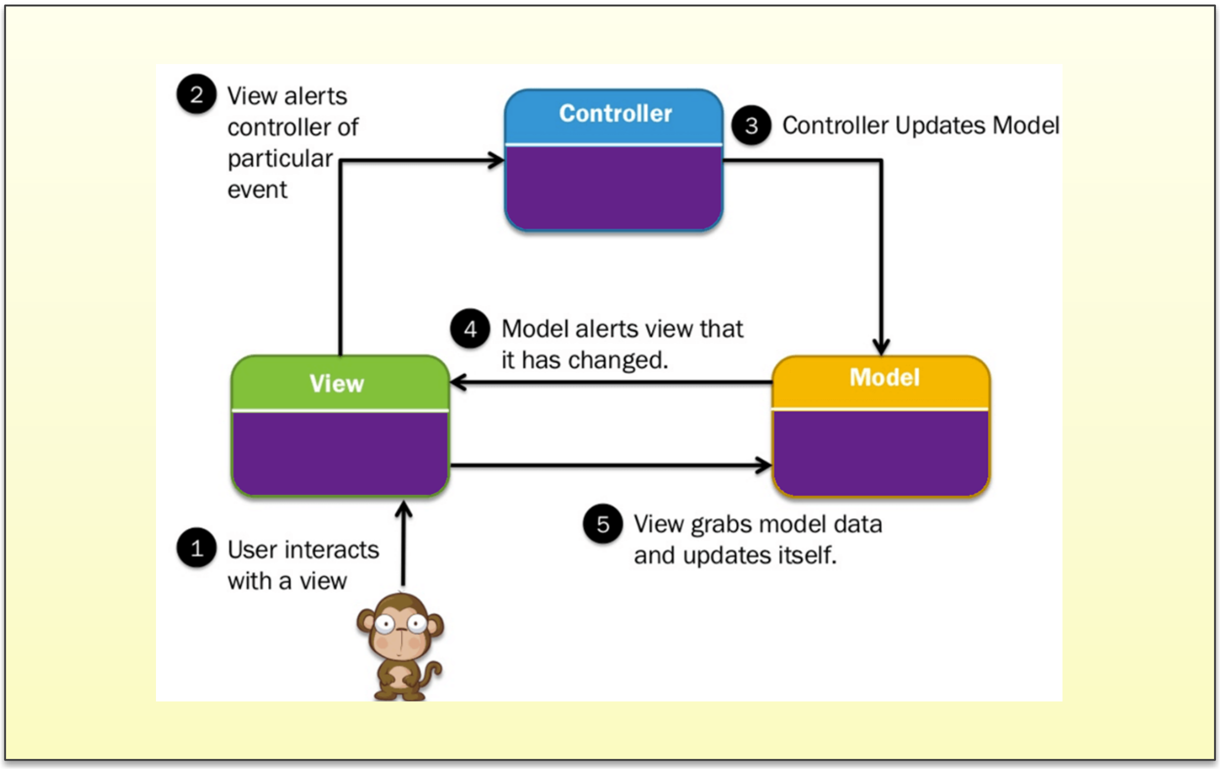
Key Application Features & Configurations:

- **Add Employee UI** for the Tenants (tenant_1, tenant_2 & tenant_3)
- **Employee List UI** for Tenants (tenant_1, tenant_2 & tenant_3)
- Tenant ID found from URL and operations are performed based on current tenant id. (Ex: http://app.com/api/tenant_id/, http://localhost:8080/tenant_id)
- **Database per Tenant** approach is followed to implement the sample Multi-tenant App.
- We are going to use the **separate database approach**, with this approach Hibernate requires you to specify a **MultiTenantConnectionProvider**. This is an interface you need to implement and will allow Hibernate to obtain connections in a tenant specific manner.
- Hibernate also requires you to implement the interface **CurrentTenantIdentifierResolver**, as this is the contract for Hibernate to able to resolve what the application considers the current tenant identifier.
- In order to identify the current tenant id, we will intercept the incoming request by using a handler interceptor. Spring provides an abstract class (**HandlerInterceptorAdapter**) which contains a simplified implementation of the HandlerInterceptor interface for pre-only/post-only interceptors.
- **MultiTenancyInterceptor** class which extends **HandlerInterceptorAdapter** class is used to extract current tenant id from URL or incoming request which is then used by **CurrentTenantIdentifierResolverImpl**.
- **MultiTenancyJpaConfiguration** switches database connection, defined in **DataSourceConfig**. In **LocalContainerEntityManagerFactoryBean** we set the **MultiTenancyStrategy** as **DATABASE** and override the connection provider and Tenant Identifier Resolver.
- **MultitenancyProperties** class is used to map the datasource properties defined in the application.properties file. We have defined 3 datasources in our properties file.
- For providing the implementation of **MultiTenantConnectionProvider,** we extend the **AbstractDataSourceBasedMultiTenantConnectionProviderImpl** provided by Hibernate, we let Spring inject our datasources, load them in a map and resolve the datasource based on the tenantIdentifier.

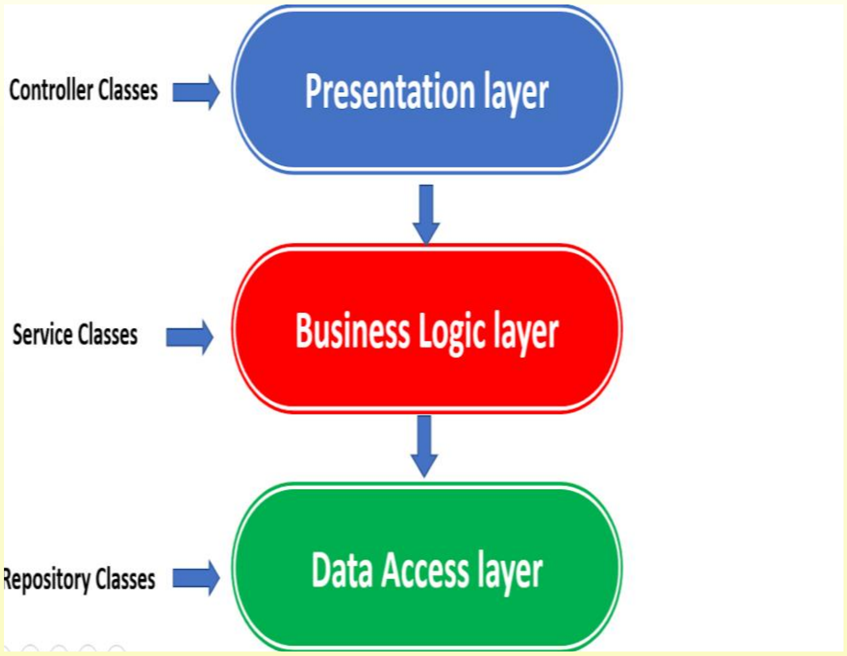
MVC Framework:

- Application is built using the **MVC** web application framework.
- The **Model-View-Controller** (**MVC**) is an architectural pattern that separates an application into three main logical components: **the model, the view, and the controller**.
  - The **Model** is responsible for managing the data of the application. It responds to the request from the view and it also responds to instructions from the controller to update itself.
  - The **View** is responsible for rendering the model data and in general it generates HTML output that the client's browser can interpret.
  - The **Controller** is responsible for processing user requests and building an appropriate model and passes it to the view for rendering.
- **Employee** is our Domain/POJO class.
- Creating a **Repository** for our **Employee** class is pretty simple using Spring. For our basic CRUD operations we only need to create an interface that extends Spring's own CrudRepository. Again nothing specific needs to be done here to support multi-tenancy.
- JPA with spring data repository model.

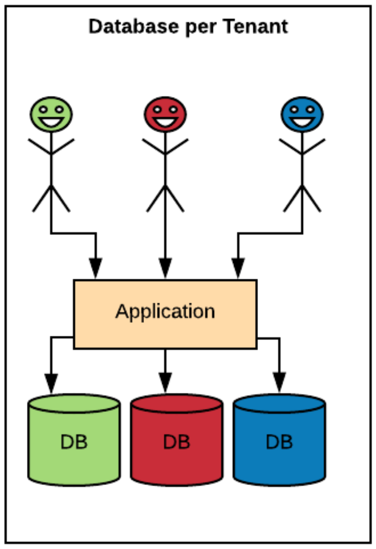
MVC Architecture Diagram


2 View alerts controller of particular event
Controller
3 Controller Updates Model
4 Model alerts view that it has changed.
View
Model
5 View grabs model data and updates itself.
1 User interacts with a view

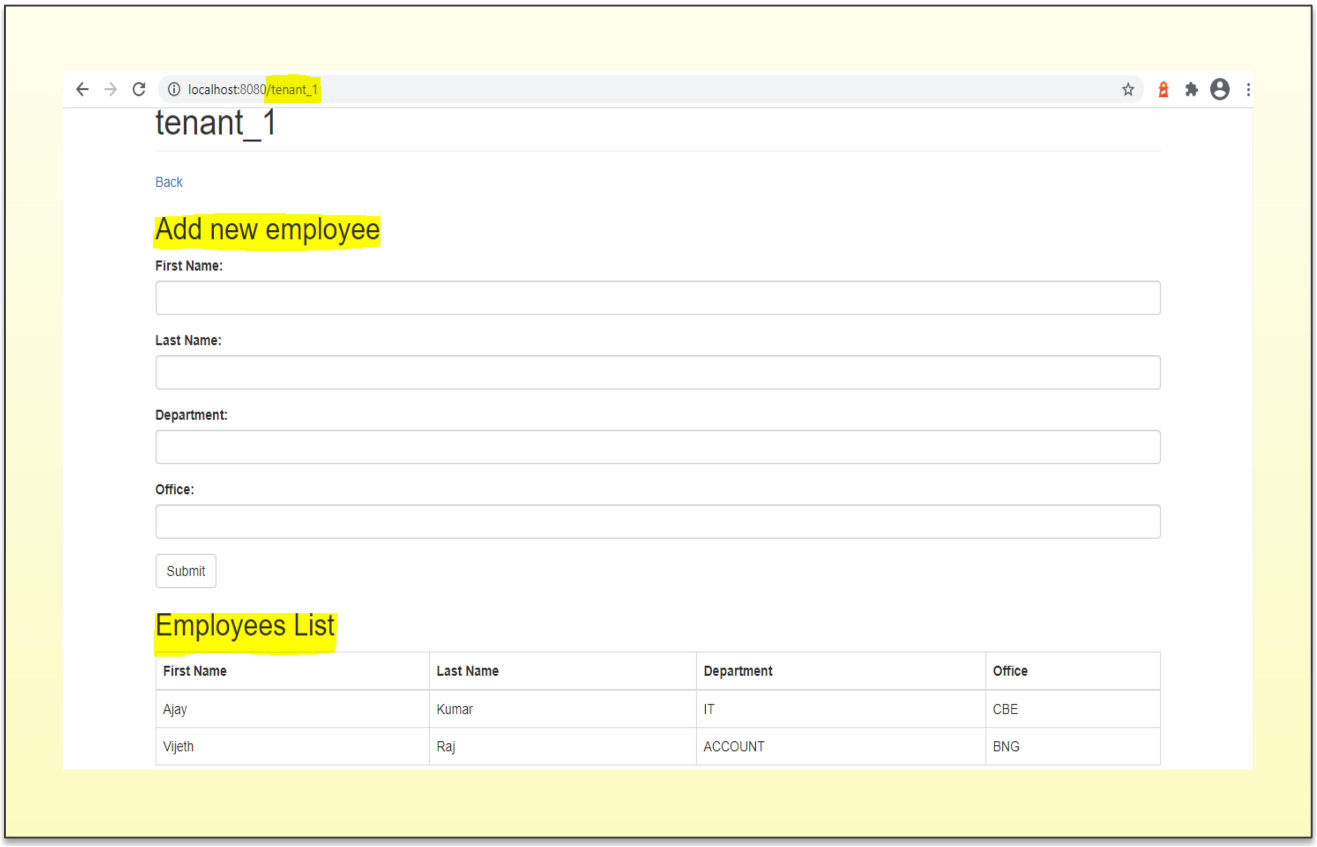
Application Architecture Diagram



Database per Tenant Architecture View



tenant_1 – Add New Employee & Employees List View



localhost:8080/tenant_1

tenant_1

Back

Add new employee

First Name:

Last Name:

Department:

Office:

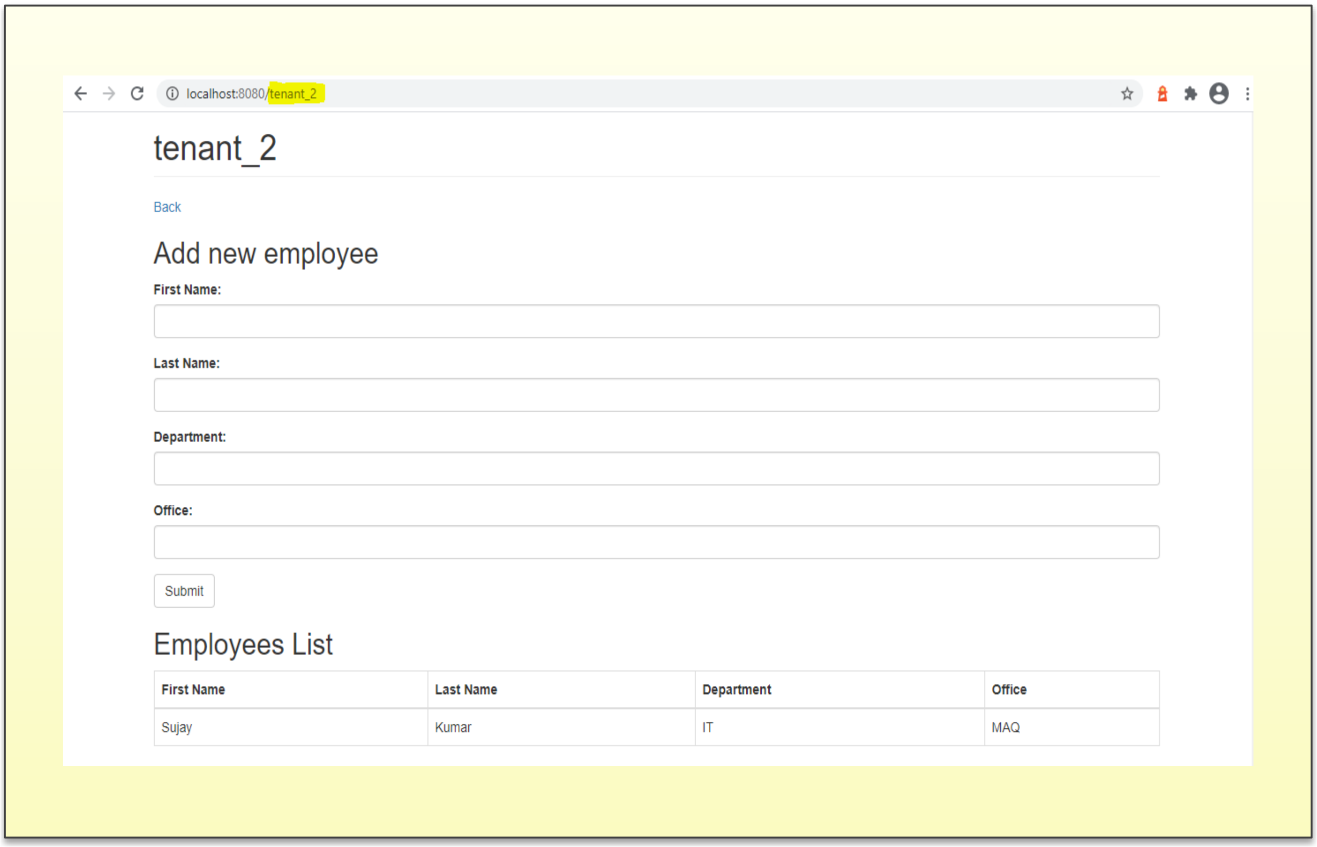
Submit

Employees List

First Name	Last Name	Department	Office
Ajay	Kumar	IT	CBE
Vijeth	Raj	ACCOUNT	BNG

KGISL-GSS (C) Copyright (2021) All Rights Reserved

tenant_2 – Add New Employee & Employees List View



localhost:8080/tenant_2

tenant_2

Back

Add new employee

First Name:

Last Name:

Department:

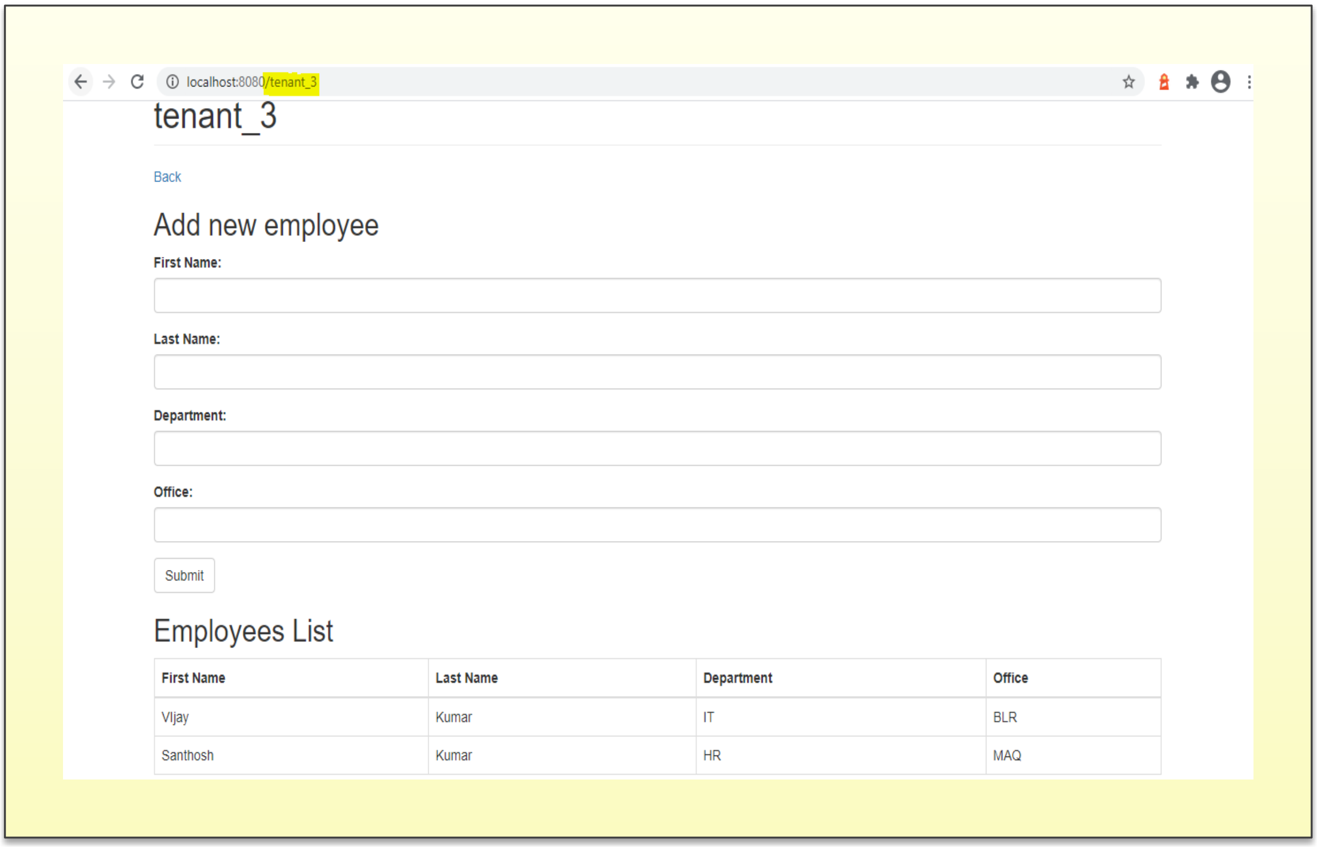
Office:

Submit

Employees List

First Name	Last Name	Department	Office
Sujay	Kumar	IT	MAQ

tenant_3 – Add New Employee & Employees List View



localhost:8080/tenant_3

tenant_3

Back

Add new employee

First Name:

Last Name:

Department:

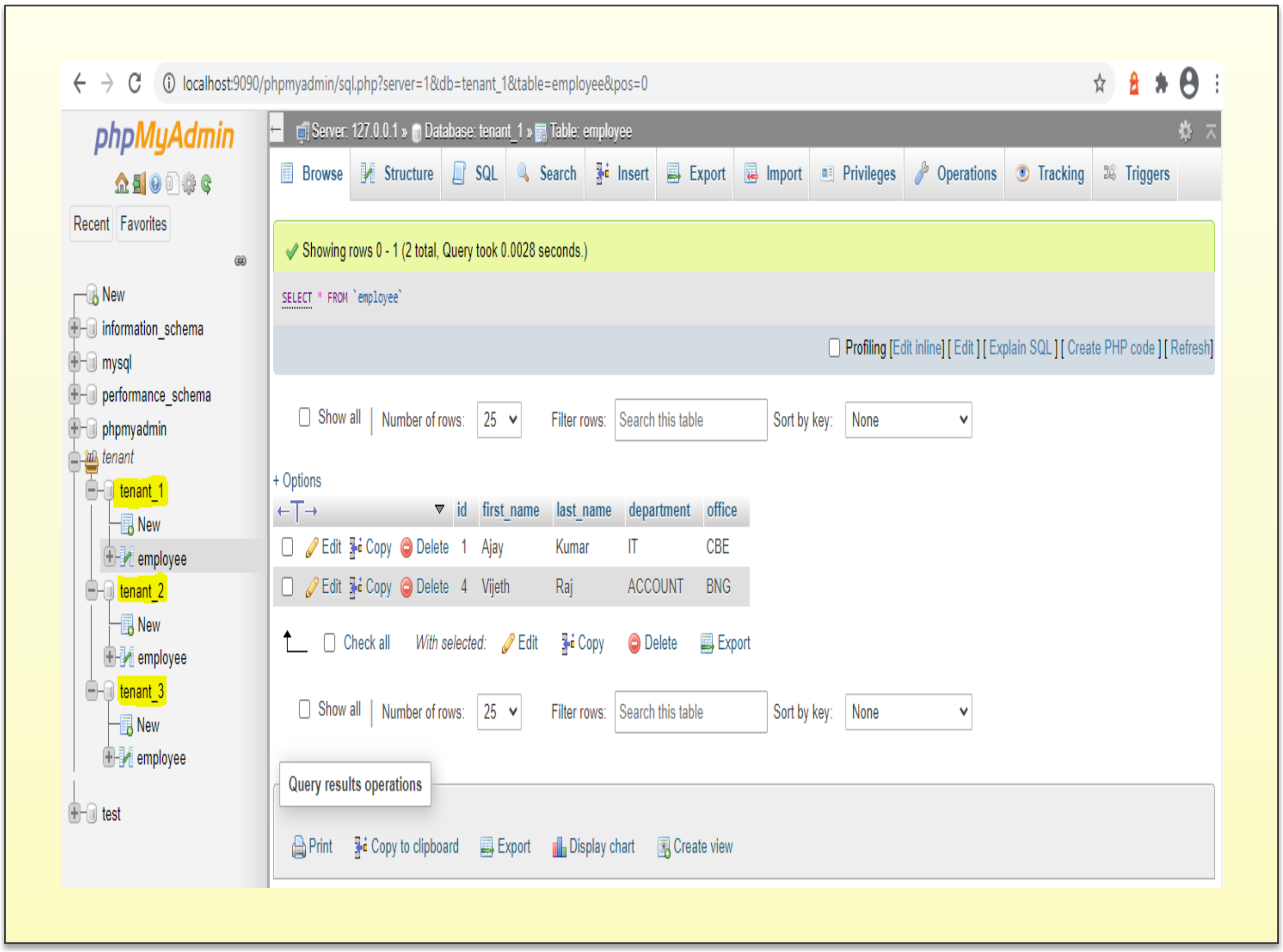
Office:

Submit

Employees List

First Name	Last Name	Department	Office
VIjay	Kumar	IT	BLR
Santhosh	Kumar	HR	MAQ

Multi-tenant Database View:


localhost:9090/phpmyadmin/sql.php?server=1&db=tenant_1&table=employee&pos=0
phpMyAdmin
Recent Favorites
New
information_schema
mysql
performance_schema
phpmyadmin
tenant
tenant_1
New
employee
tenant_2
New
employee
tenant_3
New
employee
test
Server: 127.0.0.1 » Database: tenant_1 » Table: employee
Browse Structure SQL Search Insert Export Import Privileges Operations Tracking Triggers

Showing rows 0 - 1 (2 total, Query took 0.0028 seconds.)

```
SELECT * FROM `employee`
```

id	first_name	last_name	department	office
1	Ajay	Kumar	IT	CBE
4	Vijeth	Raj	ACCOUNT	BNG