**DOCKER**

**(Task Documentation 1)**

**What are containers?**

It is basically a way to package application with all the necessary dependencies (codes, libraries, runtime environment), which is portable and can move around developers in development to make deployment and development more efficient.

**What are images?**

Images those were created by the community or by the organisations, not directly by docker

**Building Container Images:**

Doker images are read only templates that contain instructions for creating a container.

Docker Engine is used to build docker container images

### Commands :

1. **docker --version**  
   Displays the installed Docker version.
2. **docker info**  
   Displays information about the Docker installation, including number of containers, images, storage driver, etc.
3. **docker run <image>**  
   Creates and starts a container from the specified image.
4. **docker run -d <image>**  
   Creates and starts a container in detached mode (in background).
5. **docker run -it <image>**  
   Creates and starts a container interacting with the terminal (interactive mode).
6. **docker ps**  
   Lists all running containers.
7. **docker ps -a**  
   Lists all containers (running and stopped).
8. **docker stop <container\_id>**  
   Stops a running container.
9. **docker start <container\_id>**  
   Starts a stopped container.
10. **docker restart <container\_id>**  
    Restarts a running or stopped container.
11. **docker rm <container\_id>**  
    Deletes a stopped container.
12. **docker exec -it <container\_id> <command>**  
    Executes a command inside a running container.
13. **docker logs <container\_id>**  
    Fetches logs from a specified container.

**Managing Docker Images**

1. **docker images**  
   Lists all Docker images available locally.
2. **docker rmi <image\_id>**  
   Removes a specified image.
3. **docker pull <image>**  
   Downloads an image from Docker Hub or another registry.
4. **docker push <image>**  
   Uploads an image to a Docker registry.
5. **docker build -t <image\_name>:<tag> <path>**  
   Builds an image from a Dockerfile located in the specified path.

**Tagging and Naming**

1. **docker tag <image\_id> <new\_image\_name>:<tag>**  
   Creates a new tag for an existing image.

**Managing Docker Networks**

1. **docker network ls**  
   Lists all Docker networks.
2. **docker network create <network\_name>**  
   Creates a new Docker network.
3. **docker network inspect <network\_name>**  
   Displays detailed information about a specified network.
4. **docker network rm <network\_name>**

**to name container by your side**

docker run --name <any\_name> -d -p <image\_name:version>

docker run --name <any\_name> -d -p 9000:80 <image\_name>

**Container Registries**:

A container registry is a repository—or collection of repositories—used to store and access container images.

**Running Containers:**

When you execute docker run, the container process that runs is isolated in that it has its own file system, its own networking

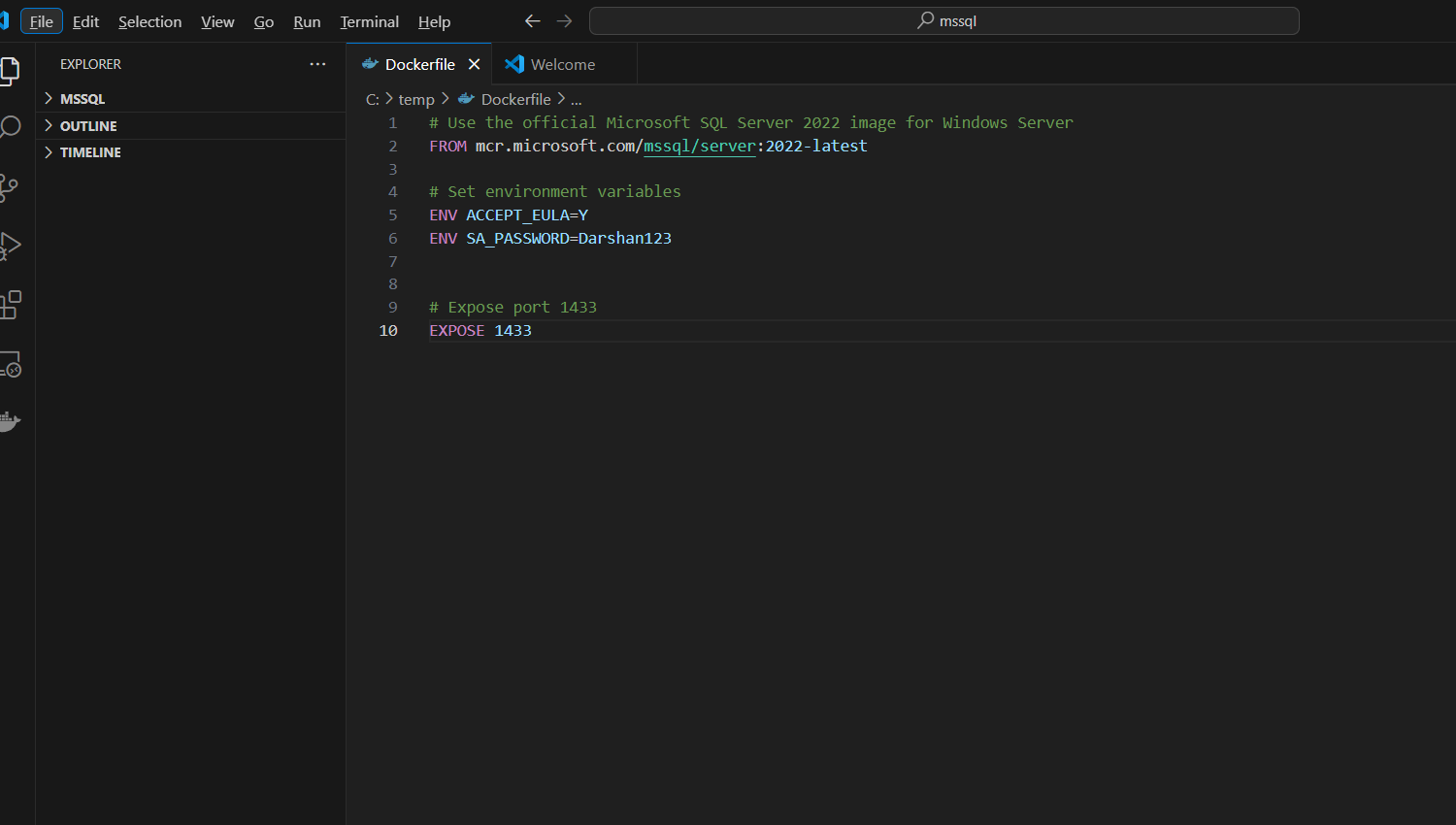
**Docker CLI**: Docker comes with command line interface to interact with docker server;

--------------------------------------------------------------------------------------------------------

**Task 1:**

* **Spin Off SQL Server Instance Using Docker**
* **Docker container should run SQL Server instance**
* **Connect to SQL Server instance from SSMS (SQL Server Management Studio)/ any other way.**
* **Manage the SQL Server Container**
* **create a Docker file and within that you can pull the database image then Build, Run/Deploy.**

Step 1: Create a file named Docker file without any extension



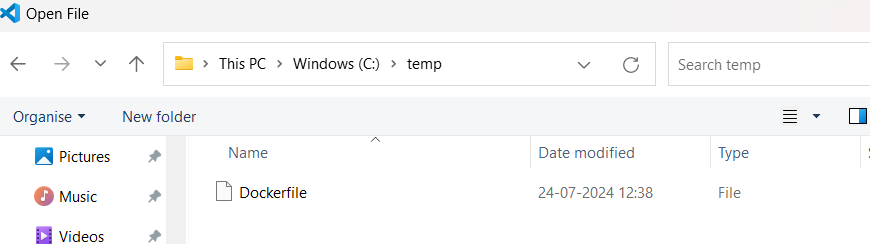
The above docker file performs above operations as follows:

1. **FROM mcr.microsoft.com/mssql/server:2022-latest**
   * This line specifies the base image for the Docker container. In this case, it uses the official Microsoft SQL Server 2022 image from the Microsoft Container Registry (mcr).
2. **ENV ACCEPT\_EULA=Y**
   * This line sets an environment variable called ACCEPT\_EULA with the value Y inside the container.
   * EULA stands for End User License Agreement. By setting this variable to 'Y'
3. **ENV SA\_PASSWORD=Darshan123**

* It actually sets the environment password

1. **EXPOSE 1433**
   * This line informs Docker that the container listens on port 1433 at runtime.
   * Port 1433 is the default port that SQL Server uses for its incoming connections. By exposing this port, you enable external systems to connect to the SQL Server instance running inside the container.

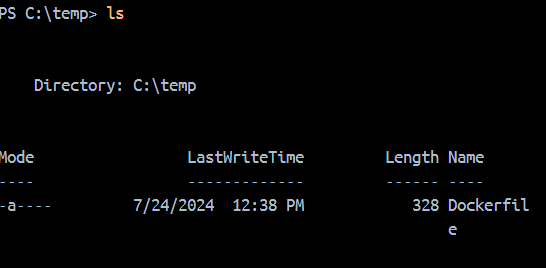
Step 2:



Created a temp folder where docker file is stored and took the path and pasted it on terminal.

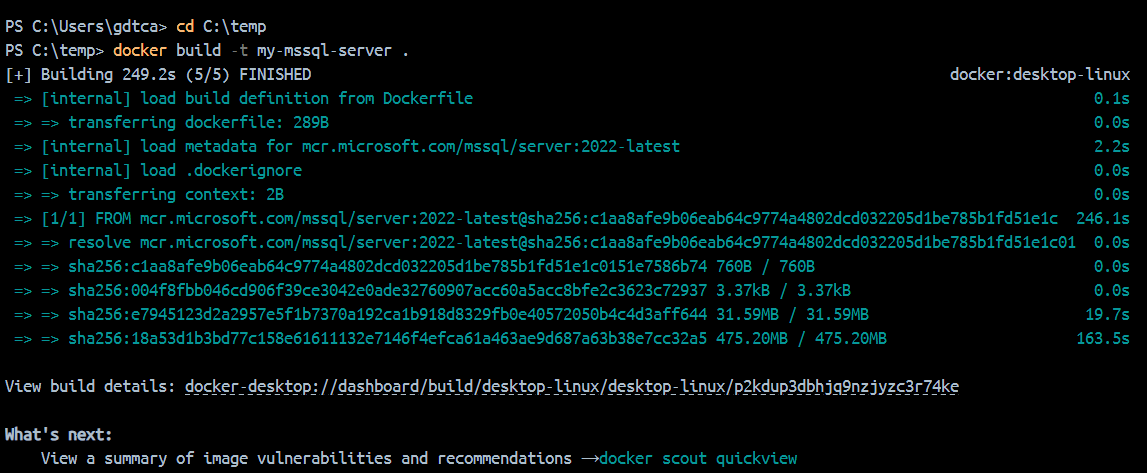
cd <file path>

**cd C:\temp**





Step 3:



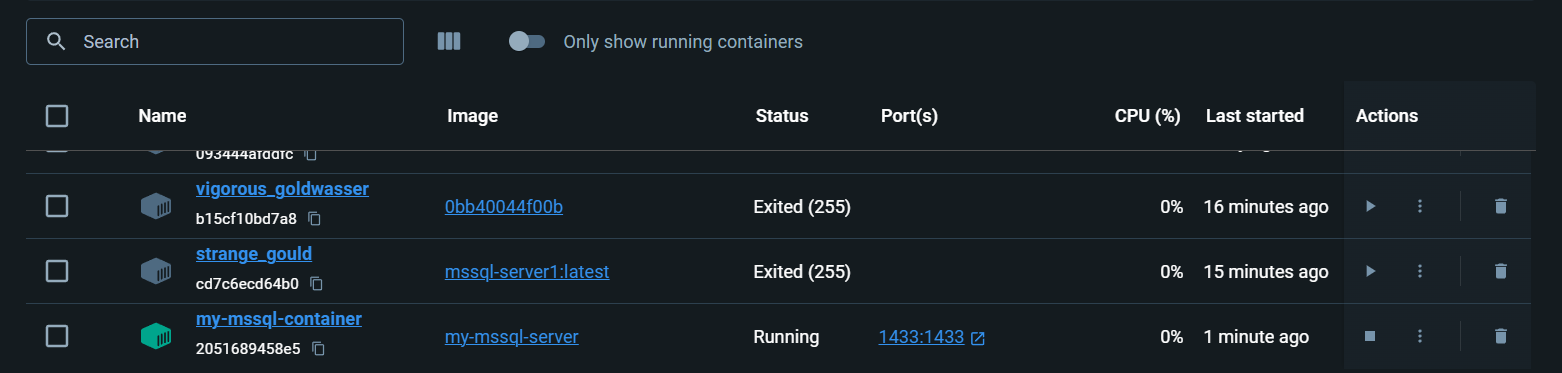
**-t it basically gives name to our image**

**Step 4:**



**-p option allows you to map a port on the host to a port on the container.**

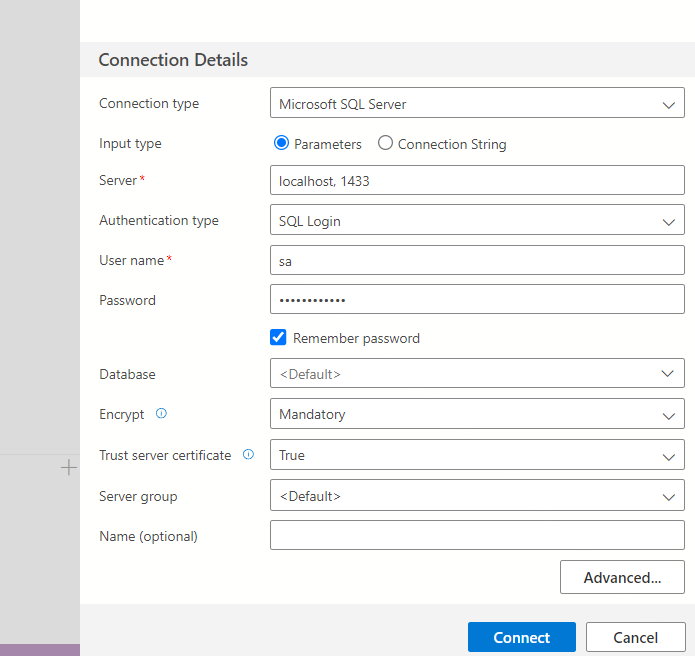
**-d is used to run the SQL Server container in detached mode.**



Here, we just created a container which is in running state

Step 5:

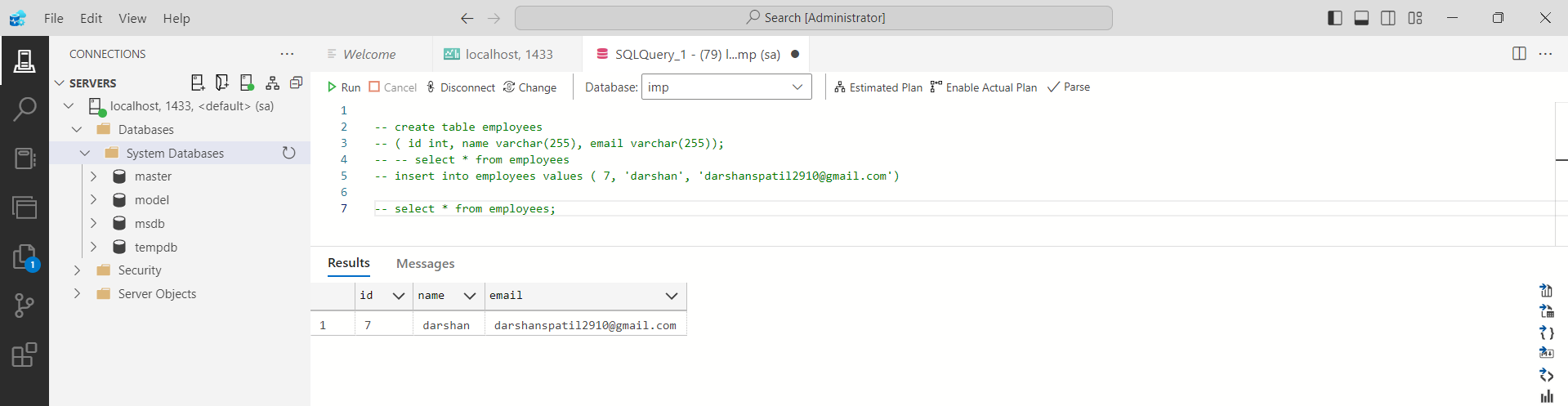
Now, we have made a connection of MS-SQL server **(run it as administrator**) and docker container which we created by following authentication by doing port binding at 1433.



Step 6:

Here, we created a database and use it for querying system

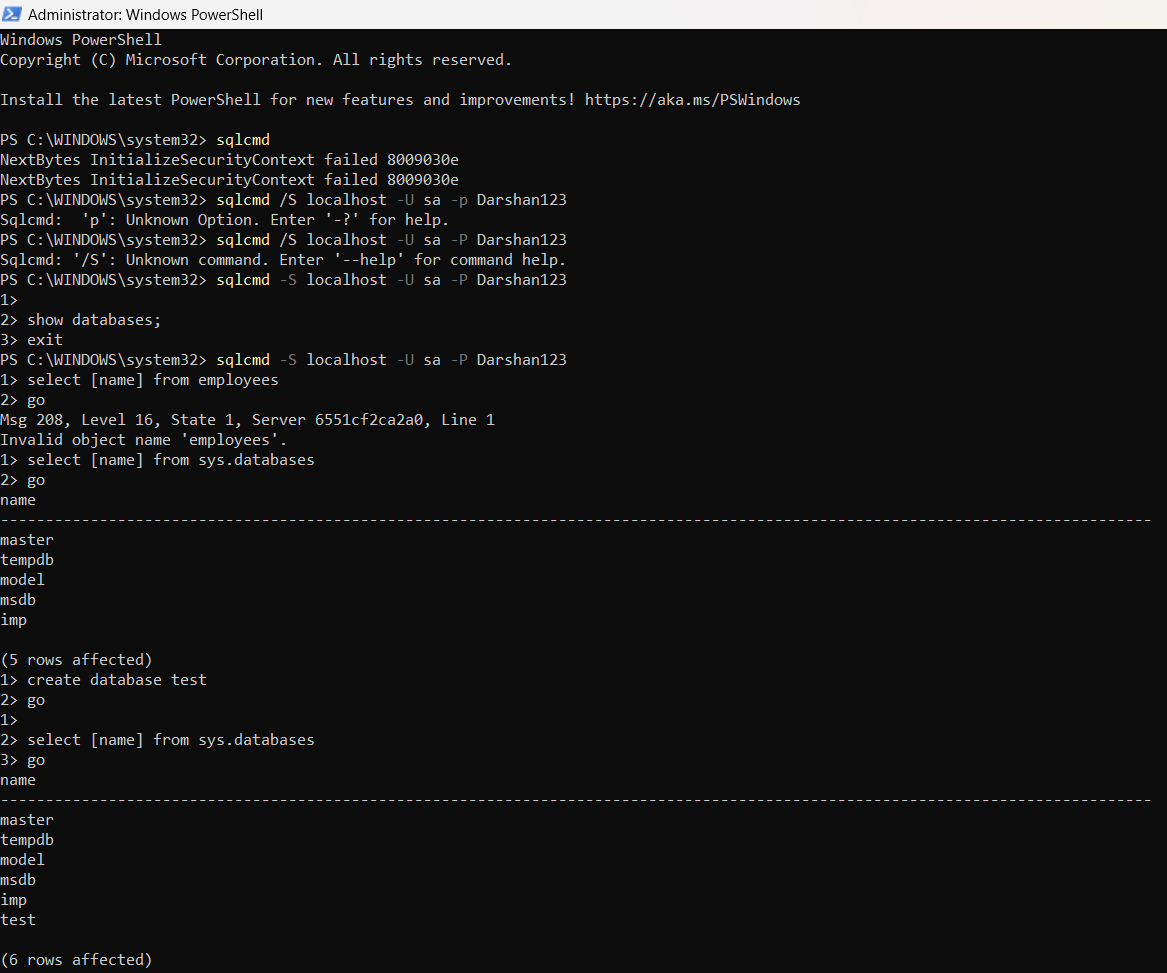




At last we created table and just randomly done some operations.

**Step 7:**

**Then open windows power shell (as administrator) sqlcmd**

****

**sqlcmd -S localhost -U sa -P Darshan123: This command actually connects sqlcmd to your sql server.**

**Where,**

**-S: specifies the server to connect**

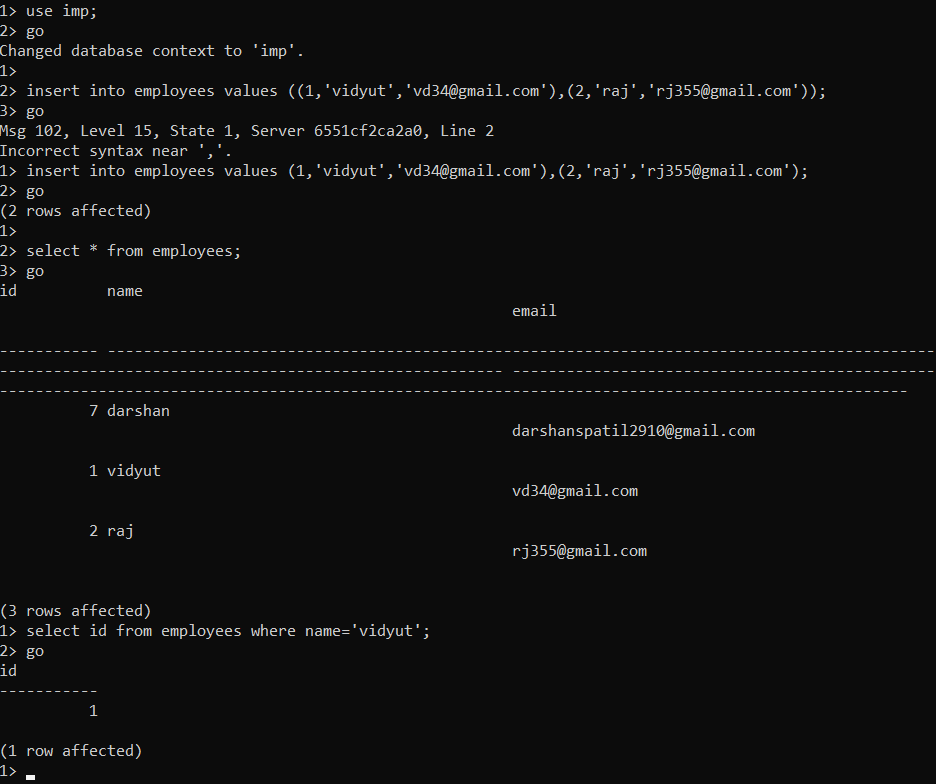
**-U: specifies the SQL Server login being used for authentication.**

**-P: This provides the password for the sa account. The password in this case is Darshan123**

**1> select [name] from sys.databases**

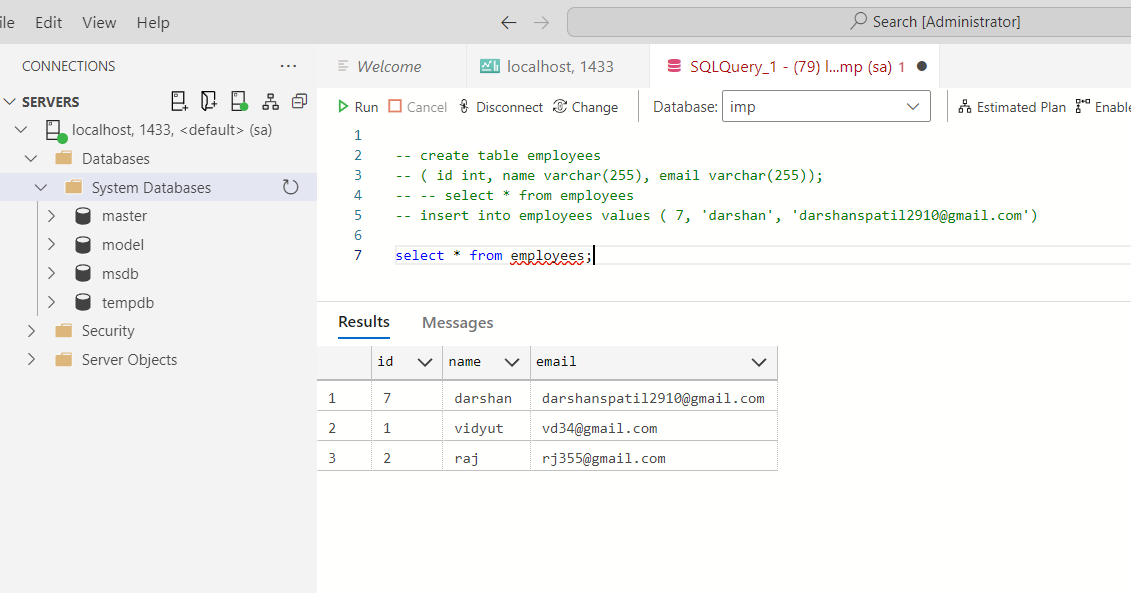
**2> go**

**This above command shows the databases available, mandatory to use ‘go’ on next line to execute the query.**

****

**Step 8:**

**Same data is reflecting in SQL server as well.**

****

**Hence, we are successfully connected the both system and software and can run at both the side.**