

**TECNOLOGICO NACIONAL DE MEXICO CAMPUS CULIACAN**

**INGENIERIA EN SISTEMAS COMPUTACIONALES**



**MATERIA**

**ADMINISTRACION DE BASE DE DATOS**

**INTEGRANTES**

**LOZANO CORVERA DANIEL ANTONIO**

**GARCIA AGUILAR JOSE ALFREDO**

**MAESTRO**

**DANIEL ESPARZA SOTO**

**FECHA**

**30-AGOSTO-2022**

**PROYECTO 1 MYSQL**

## **1- HISTORIA**

MySQL es un sistema gestor de base de datos que sigue el modelo relacional y utiliza SQL (Structured Query Language). En el año de 1994 Michael Widenius y David Axmark crearon una empresa y desarrollaron la primera versión de MySQL AB, en el año 2008 Sun Microsystems compro dicha empresa, pero al poco tiempo tan solo en 2010 Oracle compro la compañía Sun Microsystems. Actualmente existen varias ediciones como Free Community Edition que es libre totalmente, y ediciones comerciales como la Standard Edition, Enterprise Edition, y Cluster Carrier Grade Edition, estas ya son de pago y ofrecen mejores prestaciones. Lo que quería MySQL era conseguir un sistema veloz y capaz de manejar grandes cantidades de datos y usuarios.

## **2- VERSIONES**

MySQL está disponible en dos versiones:

- Una versión libre: acceso libre a los fuentes del programa (open source) y gratuito (free). Se distribuye bajo licencia GPL desde la versión 3.23.19 (junio de 2000).
- Una versión comercial de pago, con el nombre de MySQL Enterprise.

La última versión disponible (finales 2012) es la 5.5. MySQL es portable a un gran número

## **3- CARACTERISTICAS**

Una de las principales características de MySQL es que puede utilizarse en diferentes sistemas tales como Linux, Windows, AIX, Solaris... También, tiene múltiples motores de almacenamiento para adaptarse a las necesidades concretas de cada entorno.

La rapidez es otro de los puntos fuertes a la hora de realizar operaciones en MySQL. Además de ser capaz de soportar una gran cantidad de diversos tipos de datos.

En lo relativo a la seguridad, MySQL tiene un sistema de contraseñas que permite verificación basada en el host lo que hace que sea de confianza.

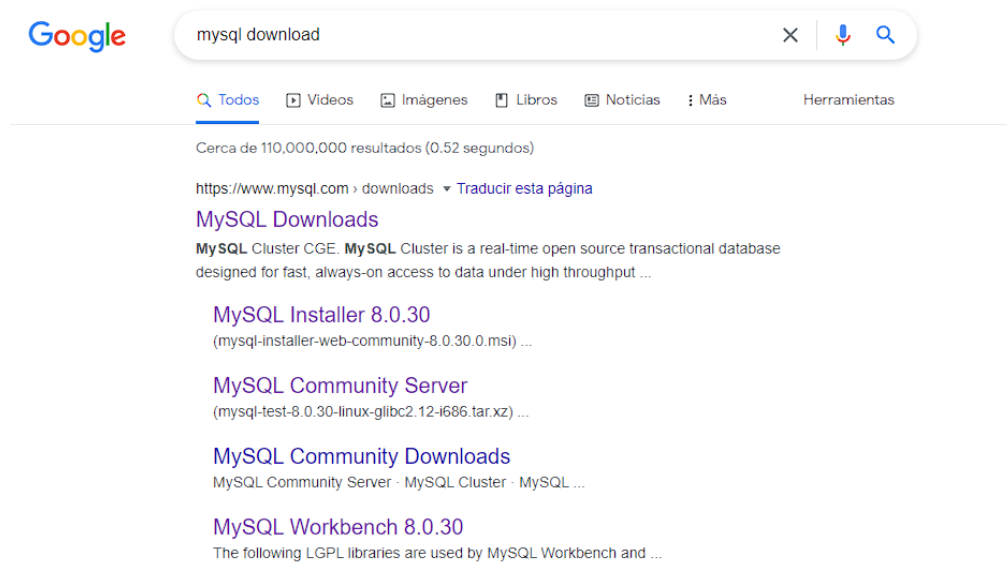
Y sin duda una de las ventajas de MySQL es que cuenta con una gran comunidad con la que intercambiar dudas y conocimientos. Es escalable y fácil de aprender lo que la convierte en una de las bases de datos más utilizada.

Los programadores web utilizan con gran frecuencia MySQL para poder realizar cambios en los sitios web de manera simple sin tener que modificar el código web. Combinado con PHP se convierte en una poderosa herramienta para realizar aplicaciones que requieran el uso de una base de datos rápida, segura y potente. Velocidad y Robustez.

- Soporte de gran cantidad de tipos de datos para las columnas.
- Gran portabilidad entre sistemas operativos y distintas plataformas.
- Base Datos cuenta con 3 archivos: Archivo de estructura, Archivo de datos y Archivo de índice.
- Potencia de sistemas multiproceso, gracias a su implementación multihilo.
- Flexibilidad de contraseñas, Gestión de usuarios, Alto nivel de seguridad en datos
- Soporte de mensajes de error en distintos lenguajes

#### 4- PROCESO INSTALACION (WINDOWS)

1. Nos dirigimos al navegador de nuestra preferencia y buscamos MySQL download



2. Seleccionamos Descargas y presionamos la versión de MySQL Community Server.



3. Luego vamos por el instalador y seleccionamos el archivo que este mas completo que en este caso es el de abajo.



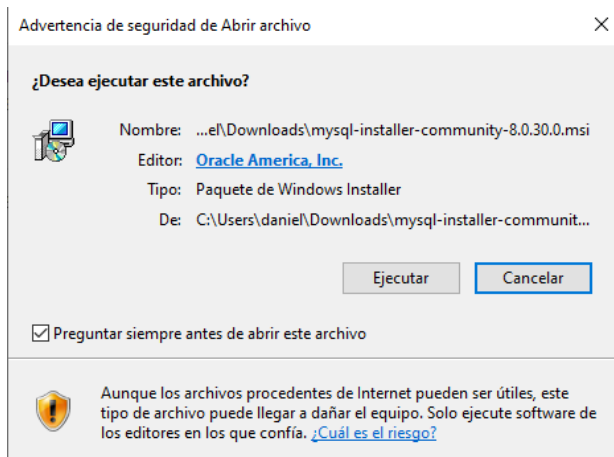
<b>Windows (x86, 32-bit), MSI Installer</b> (mysql-installer-web-community-8.0.30.0.msi)	8.0.30	5.5M	<a href="#">Download</a>
MD5: c095cf221e8023fd8391f81eadce65fb   <a href="#">Signature</a>			
<b>Windows (x86, 32-bit), MSI Installer</b> (mysql-installer-community-8.0.30.0.msi)	8.0.30	448.3M	<a href="#">Download</a>
MD5: c9cbd5d788f45605dae914392a1dfeea   <a href="#">Signature</a>			

4. Una vez descargado el instalador, damos clic derecho y seleccionamos instalar.

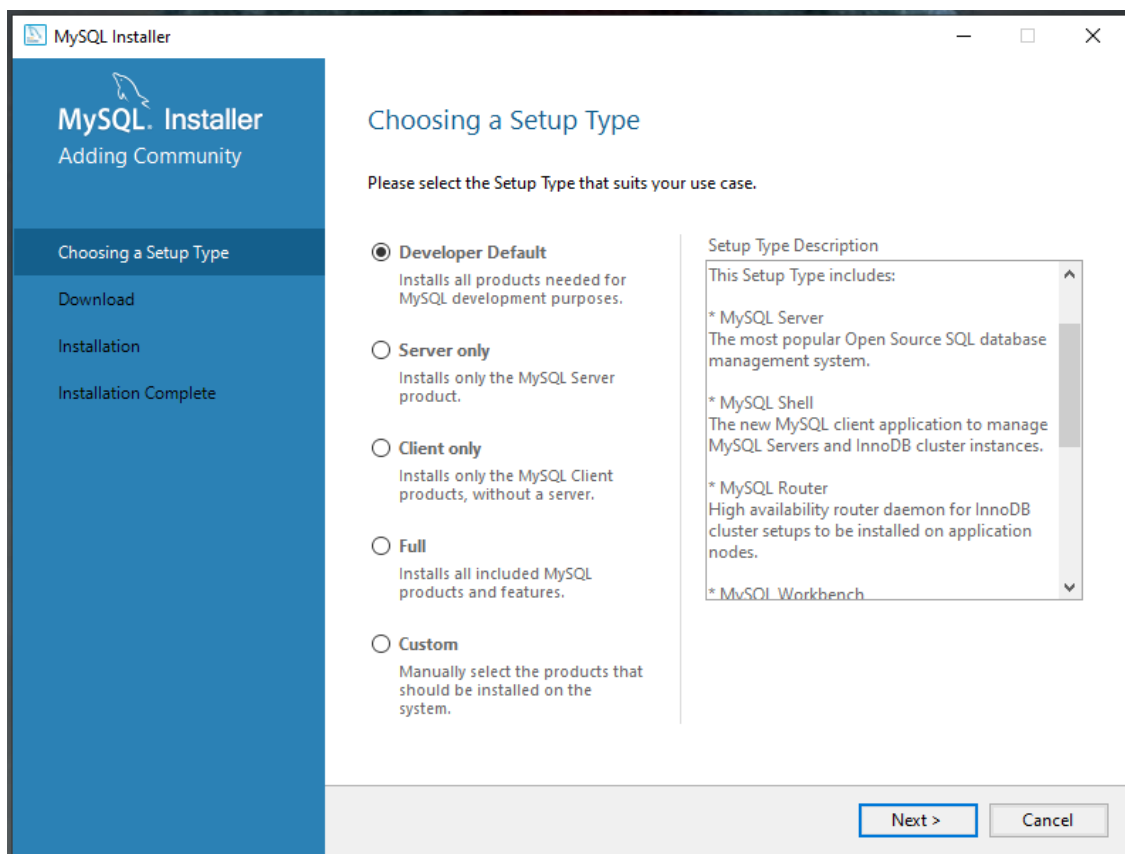


**mysql-installer-community-8.0.30.0**

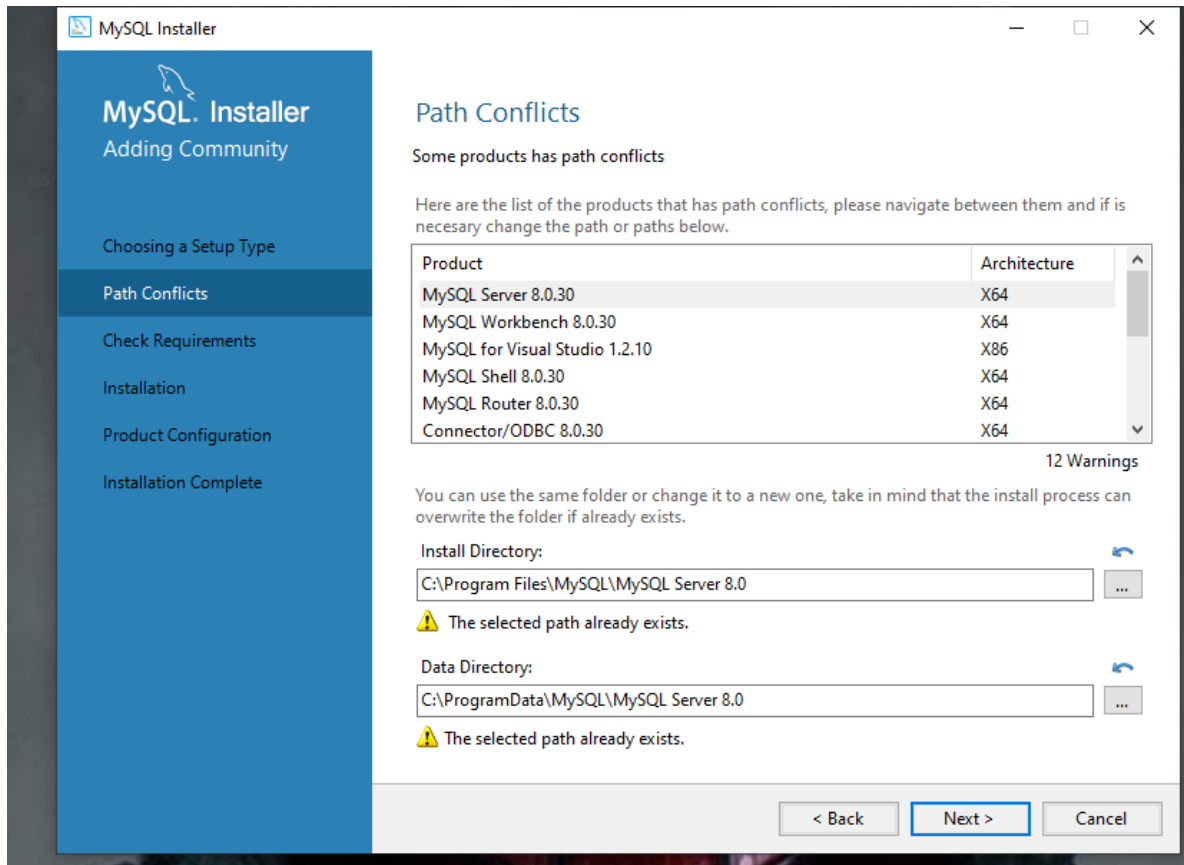
5. Nos aparecerá una ventana para dar acceso a la aplicación y seleccionamos que sí y se ejecuta.



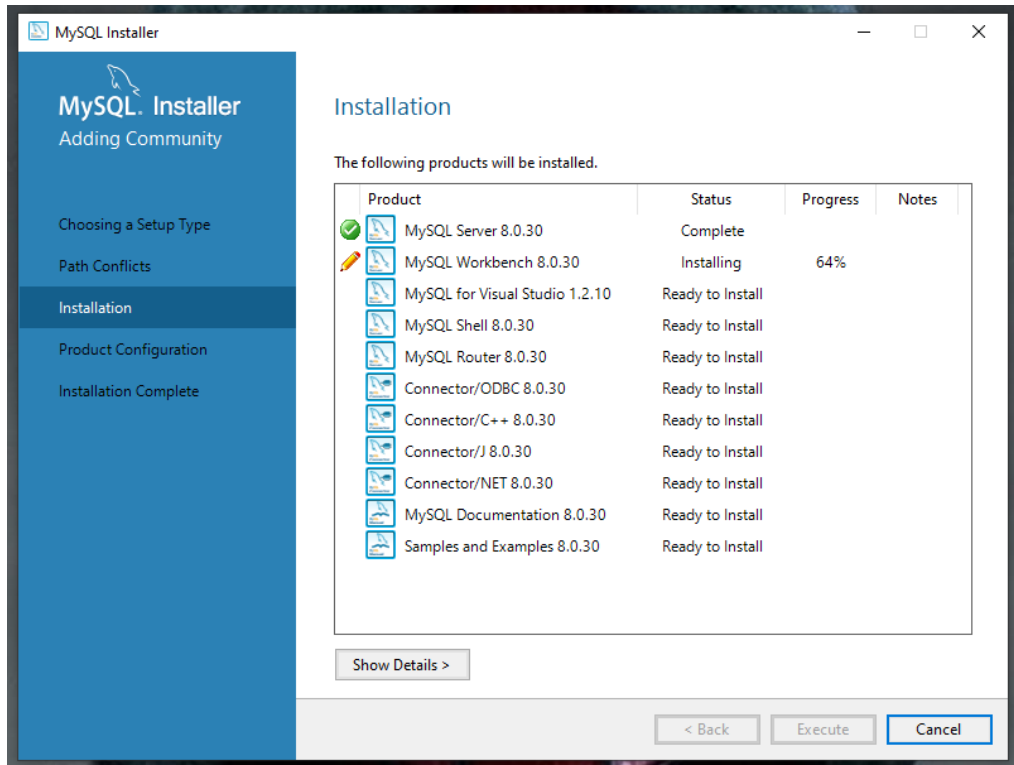
6. Se nos mostrara una ventana donde nos mostrara el tipo de instalación en este caso seleccionamos el “Desarrollador”.



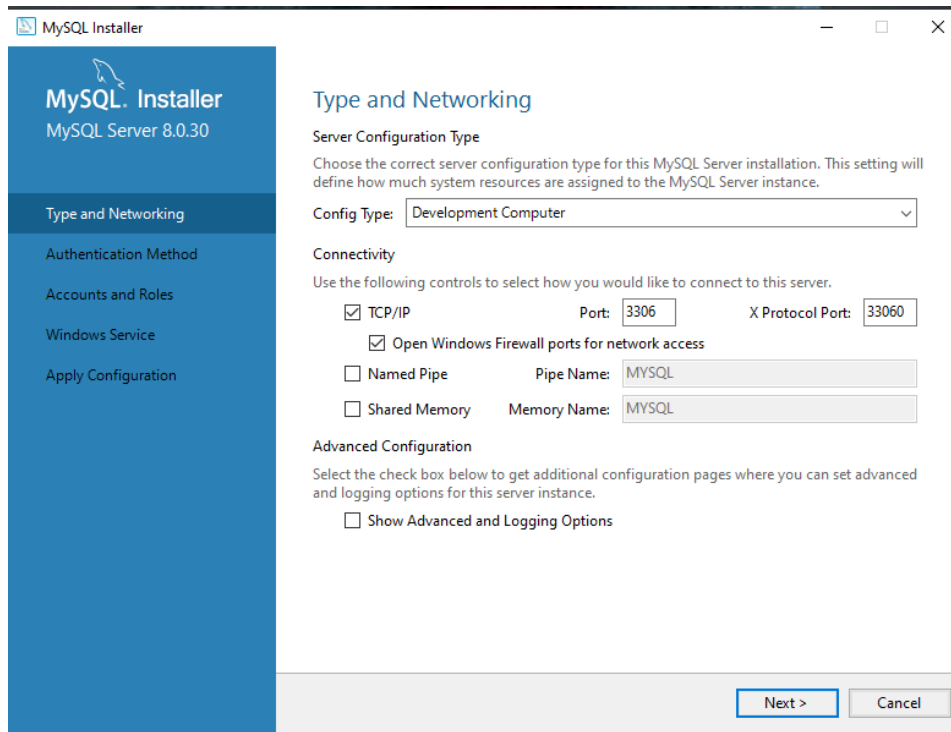
7. Otra ventana muestra los requerimientos para conectar con MySQL, le damos a next.



8. Muestra otra ventana donde se instalarán las aplicaciones de MySQL y seleccionamos ejecutar, al finalizar le damos siguiente.

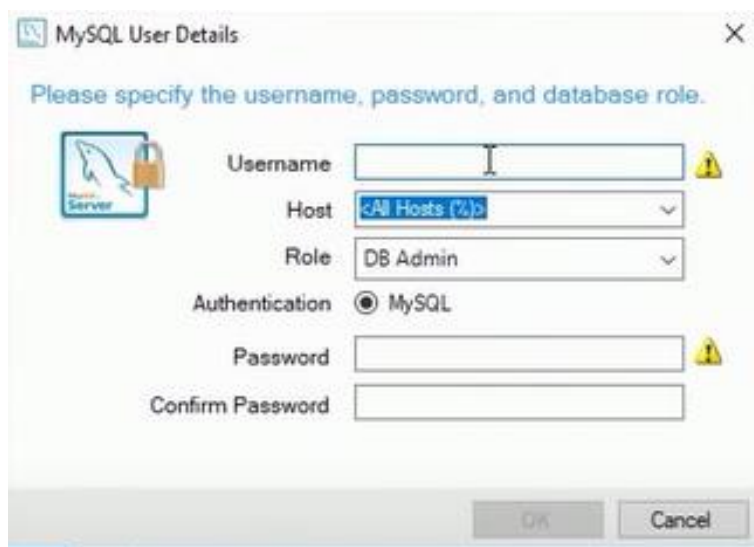
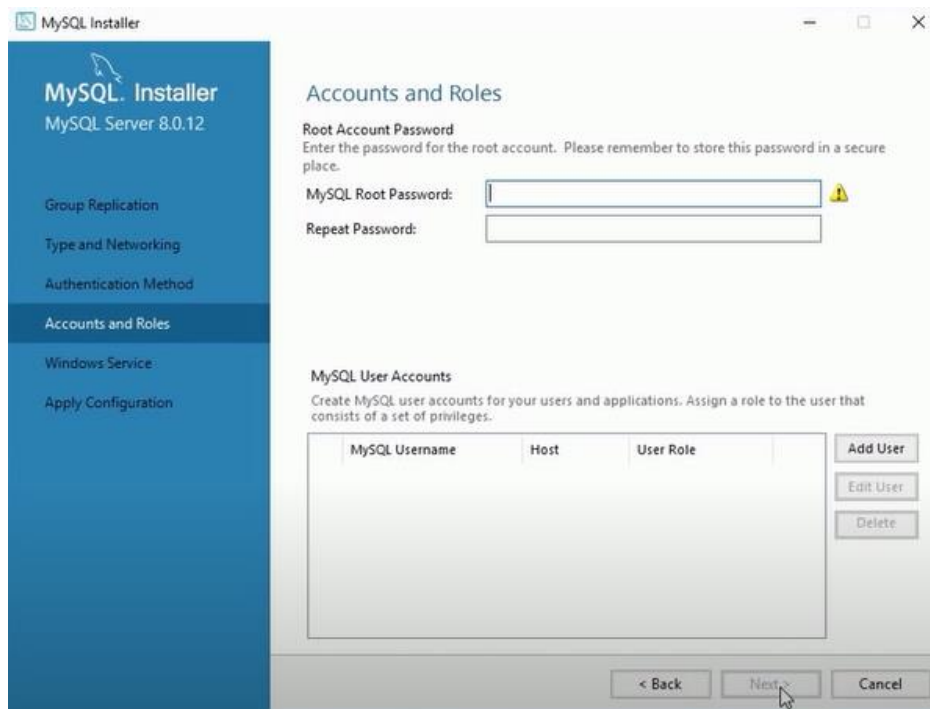


9. Muestra una ventana con el tipo y conexión de red, le damos siguiente.

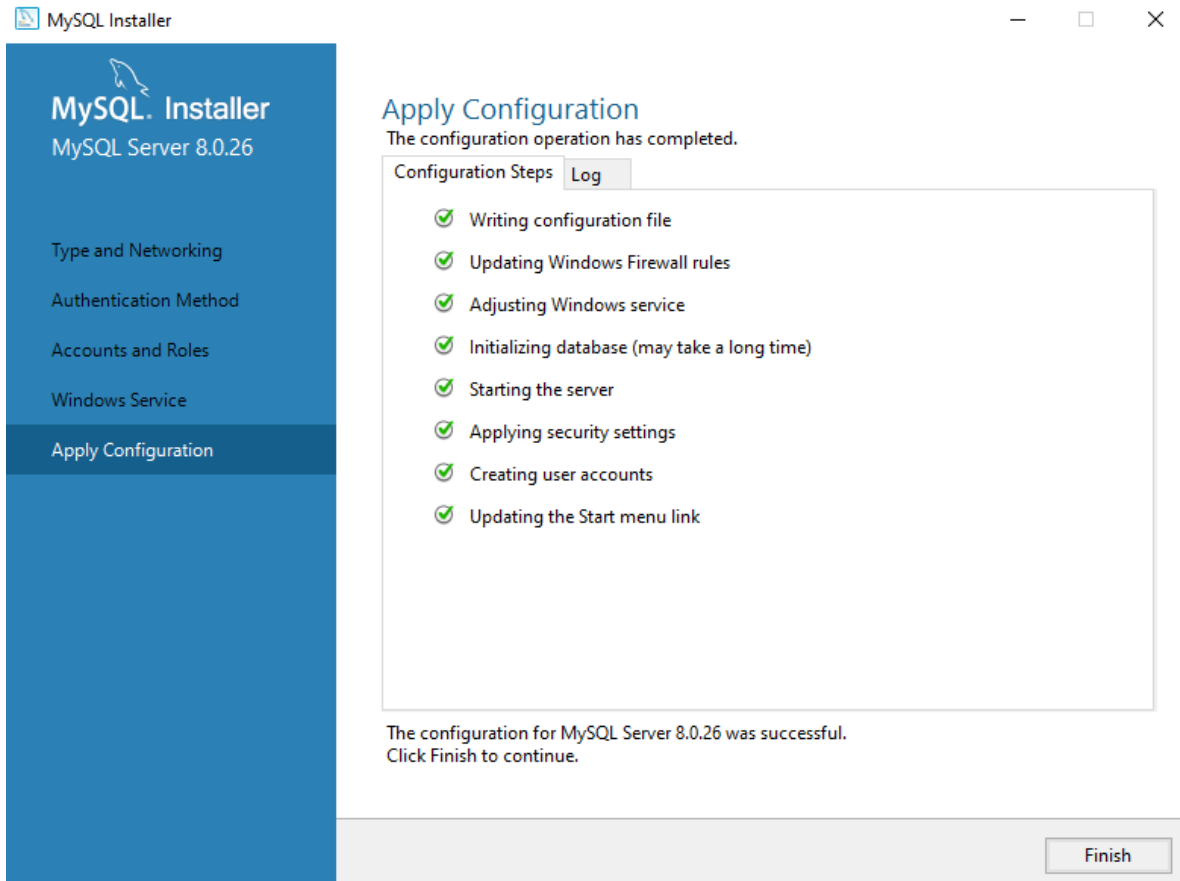




10. Otra ventana para crear lo que es el usuario y contraseña para dar acceso a MySQL.

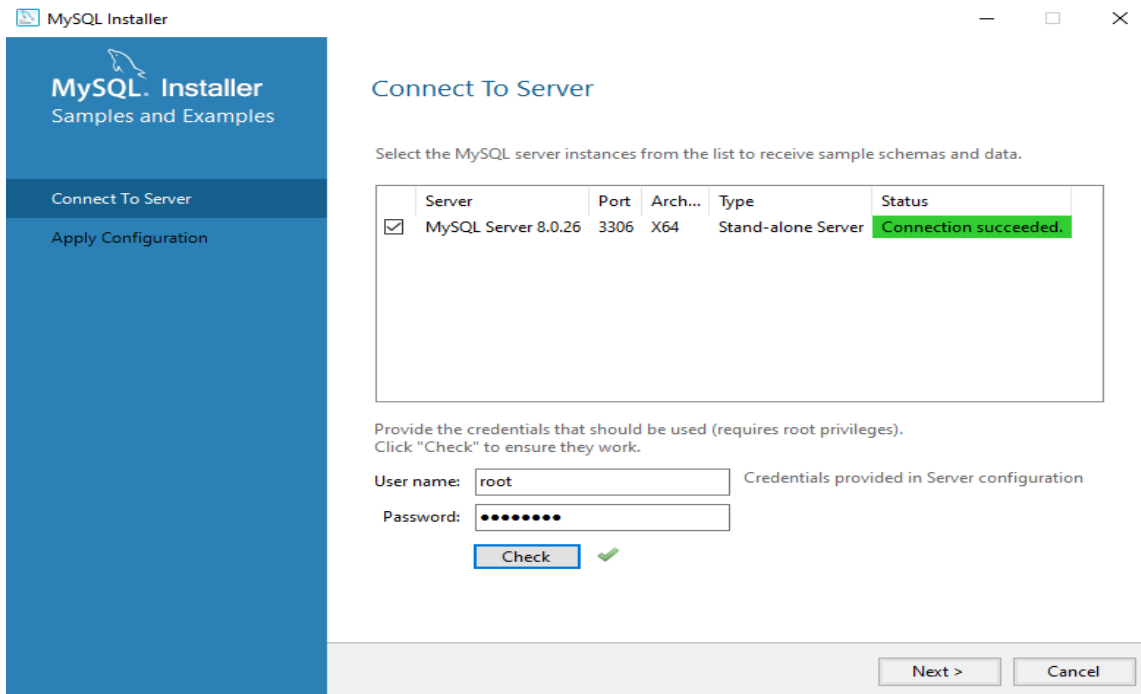


## 11. Finalizamos la instalación.

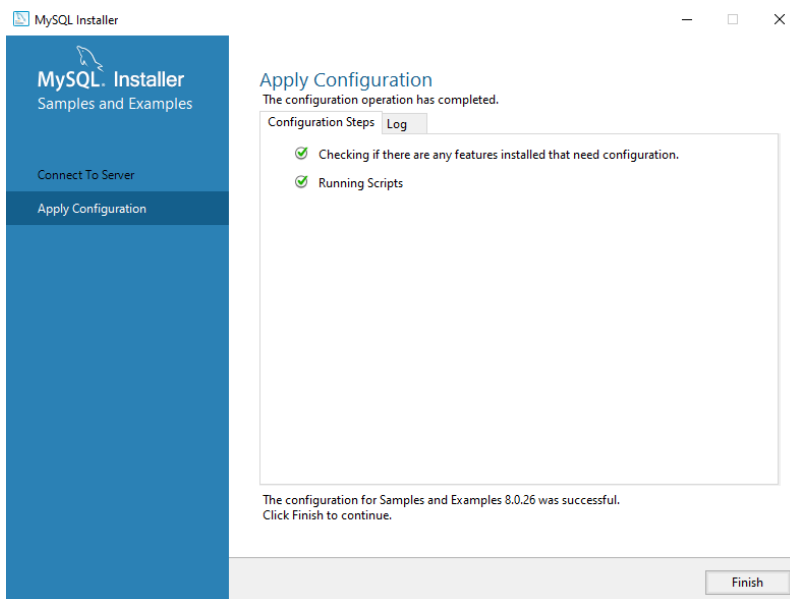


## 5- CONEXIÓN AL SERVIDOR

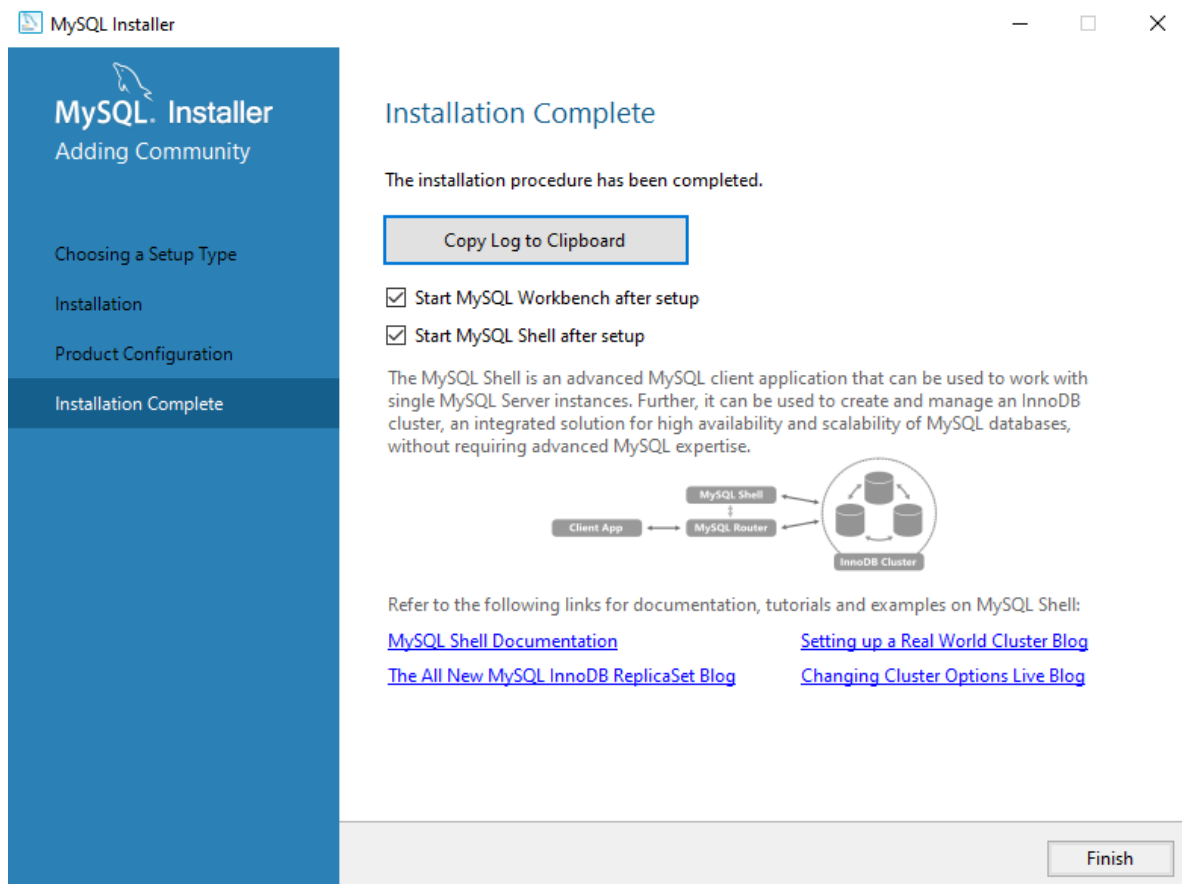
Aquí ingresamos la contraseña para poder Conectar al Servidor y comprobación de credenciales



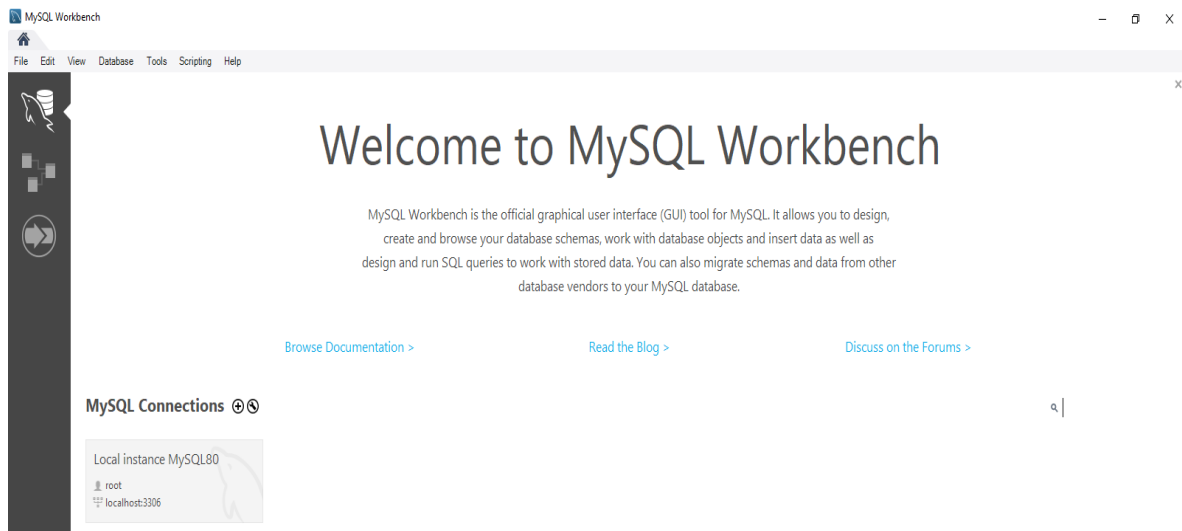
Se aplican las configuraciones y finalizamos



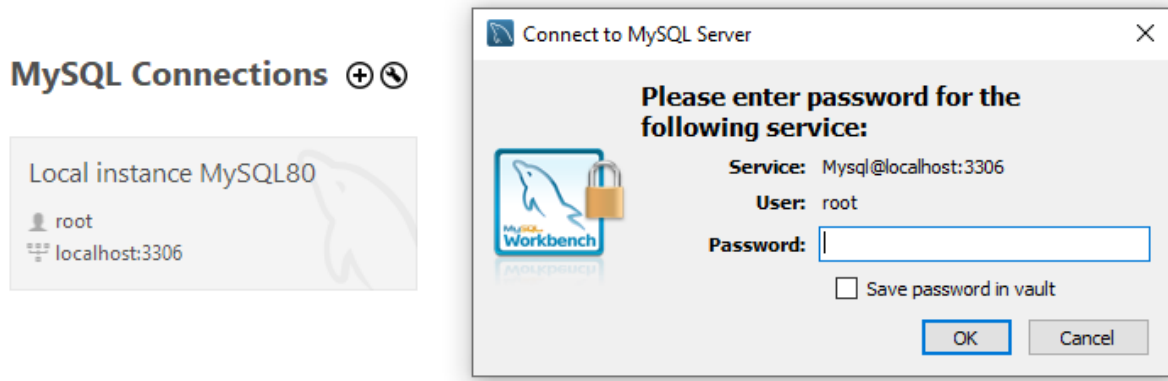
Y de nuevo finalizamos todo.



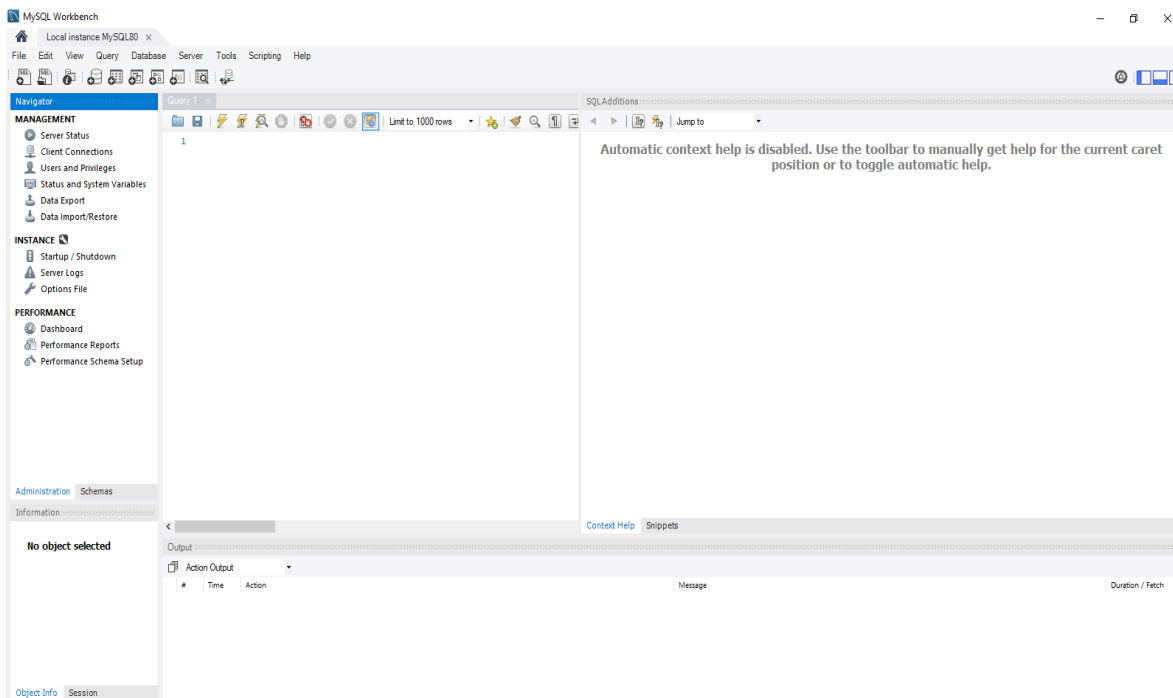
Y ya tendremos instalado MySQL y listo para usar. Para acceder solo necesitamos dar clic en la instancia local.



Una vez seleccionada la instancia local nos pedirá ingresar nuestra contraseña como se muestra.

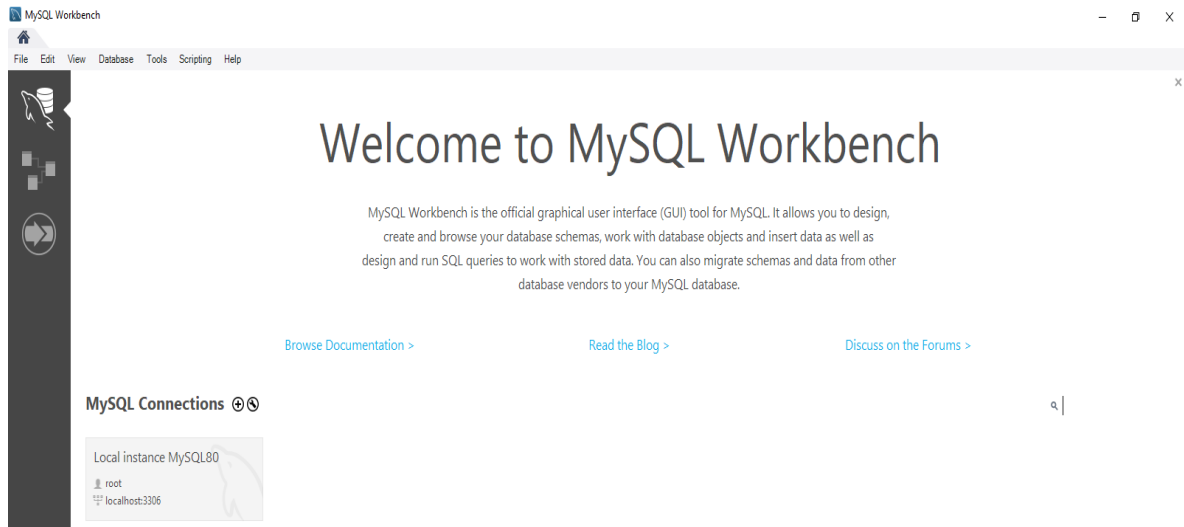


Y así se mira el entorno de trabajo de MySQL una vez que da acceso a nuestra cuenta

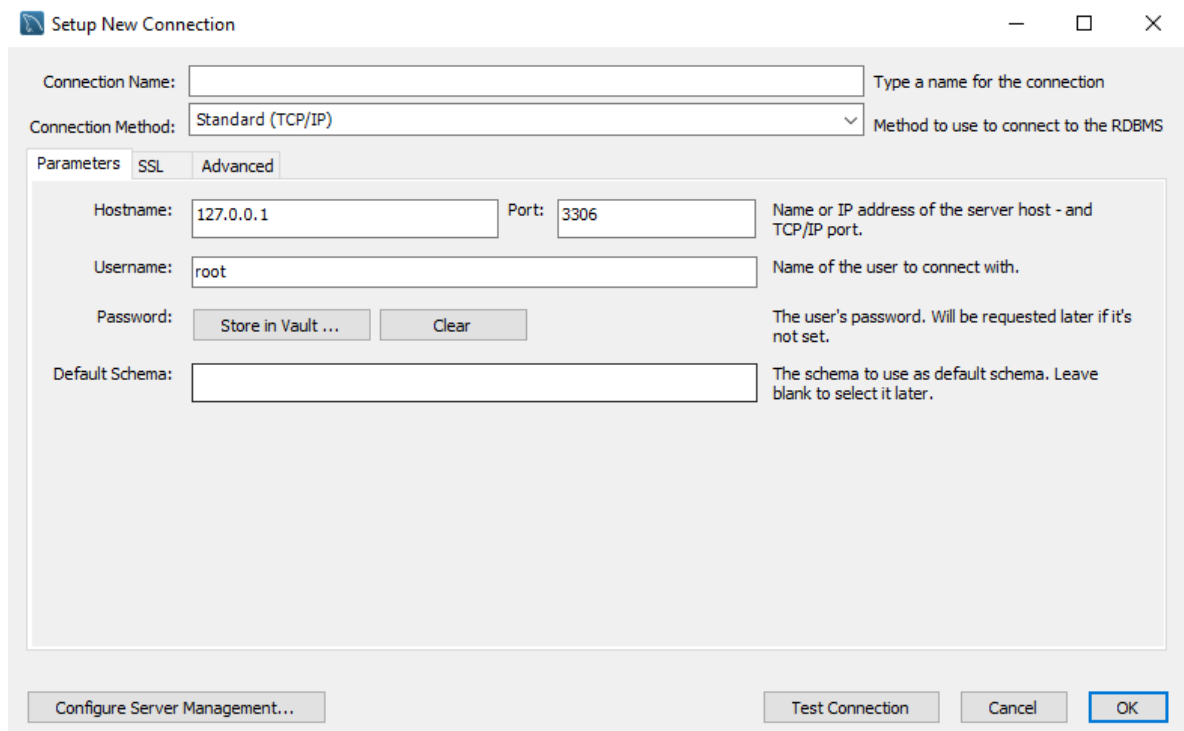


## CREAR NUEVA CONEXIÓN

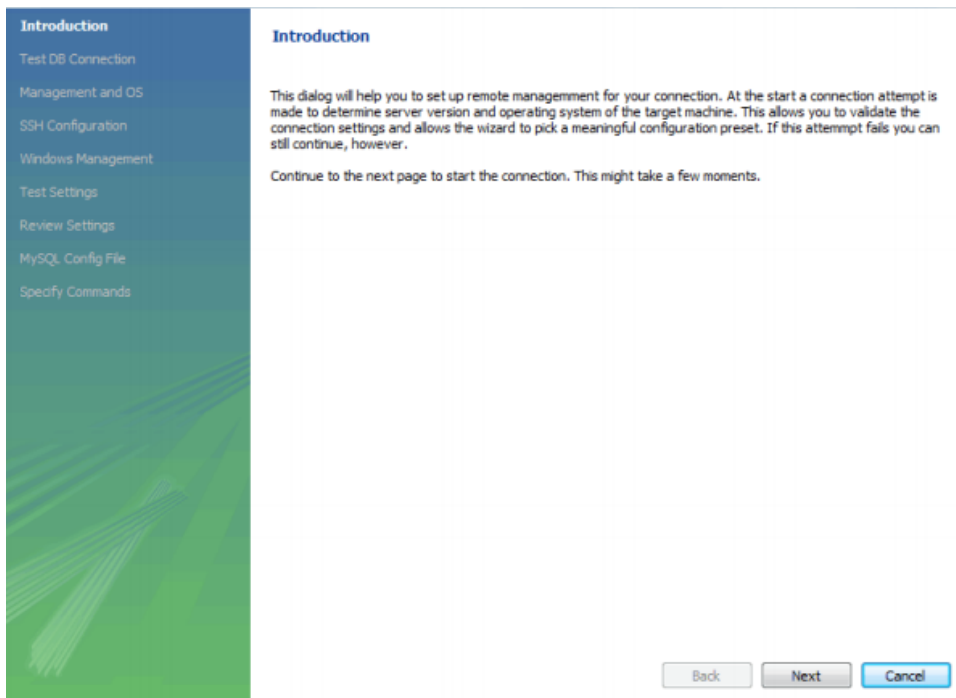
Si se desea crear una conexión nueva vas al inicio de MySQL y seleccionas donde esta el símbolo (+)



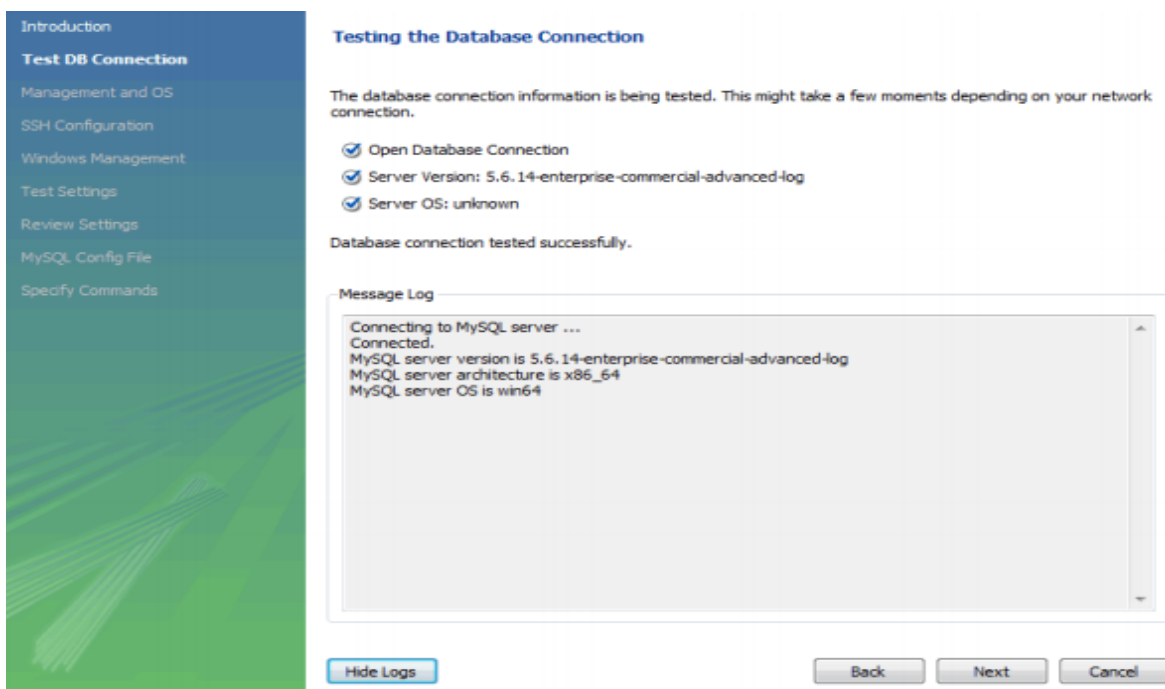
Te aparecerá esta venta en donde



Le agregaras nombre y editaras los datos necesarios y seleccionas Configure Server Management, te mostrara la venta siguiente en donde seleccionaras next.



Después se realizará una prueba de la conexión con la base de datos



Se van a realizar las configuraciones correspondientes con la maquina

**Set Windows configuration parameters for this machine**

Windows management requires a user account on this machine which has the required privileges to query system status and to control services. For configuration file manipulation read/write access to the file is needed.

Select the service to manage from the list below. It will also help to find the configuration file.

MySQL56\_1 (Running, Start mode: Auto)

Path to Configuration File: C:\ProgramData\MySQL\MySQL Server 5.6\my.ini

Back Next Cancel

Una vez realizada la prueba y las revisiones se muestran las configuraciones de MySQL y donde se guardará, luego seleccionamos Check Path.

**Information about MySQL configuration**

In order to manage the settings of the MySQL Server it is necessary to know where its configuration file resides. The configuration file may consist of several sections, each of them belonging to a different tool or server instance. Hence it is also necessary to know which section belongs to the server we are managing.

Please specify this information below.

MySQL Server Version: il-advanced-log

Path to Configuration File: C:\ProgramData\MySQL\MySQL Server 5.6\my.ini

Check Path The config file path is valid.

Section of the Server Instance: mysqld

Check Name Click to test if your section is correct.

Back Next Cancel



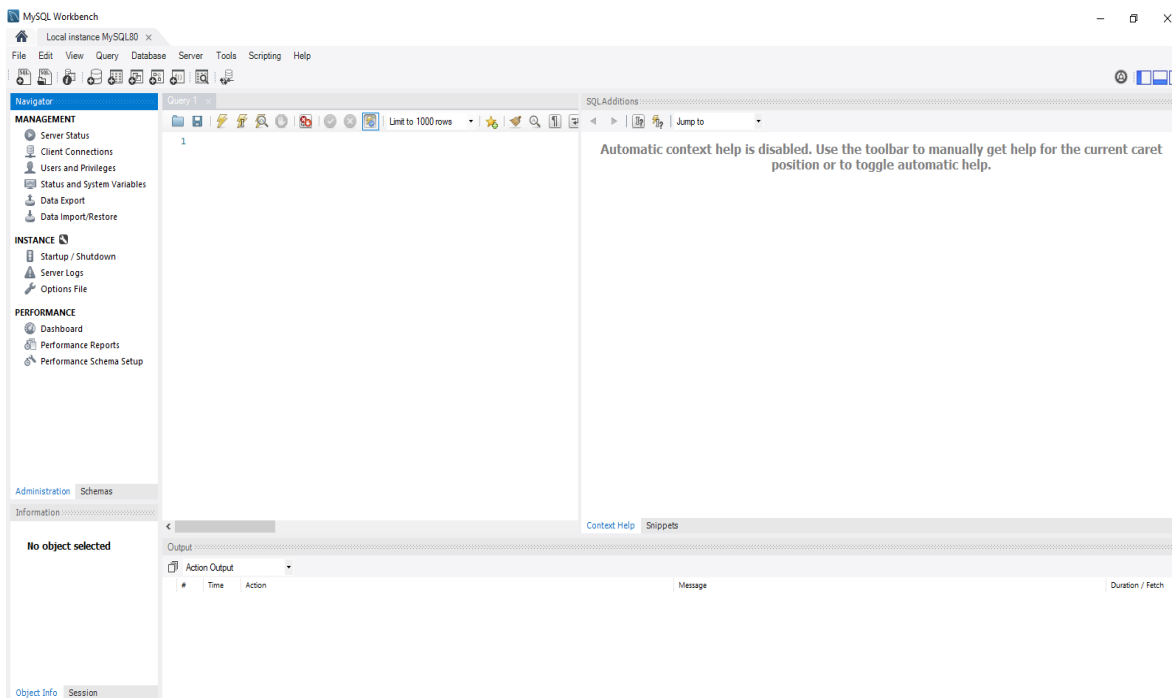
Y se nos muestra la siguiente pantalla donde se agregarán los comandos específicos para administrar MySQL y luego finalizaremos.

The screenshot shows a web-based configuration interface. On the left is a vertical sidebar with a blue-to-green gradient, containing a list of menu items: 'Introduction', 'Test DB Connection', 'Management and OS', 'SSH Configuration', 'Windows Management', 'Test Settings', 'Review Settings', 'MySQL Config File', and 'Specify Commands' (which is highlighted). The main content area has a white background. At the top, it says 'Specify commands to be used to manage the MySQL server.' followed by explanatory text: 'The values on this page comprise rather low level commands, which are used to control the MySQL server instance, check its status and so on.' and 'If you are unsure what these values mean leave them untouched. The defaults are usually a good choice already (for single server machines).' Below this, there are three input fields: 'Command to start the MySQL server:', 'Command to stop the MySQL server:', and 'Status check command:'. A checkbox is checked, with the label 'Check this box if you want or need the above commands to be executed with elevated Operating System Privileges.' At the bottom right, there are three buttons: 'Back', 'Finish', and 'Cancel'.

Se muestran todas las configuraciones hechas, seleccionamos test de conexión para asegurarnos de que este no falle y seleccionamos OK

The screenshot shows a 'Test Connection' dialog box. At the top, 'Connection Name:' is 'MyFirstConnection' and 'Connection Method:' is 'Standard (TCP/IP)'. Below are tabs for 'Parameters', 'SSL', and 'Advanced'. The 'Parameters' tab is active, showing fields for 'Hostname:' (127.0.0.1), 'Port:' (3306), 'Username:' (root), 'Password:' (with 'Store in Vault ...' and 'Clear' buttons), and 'Default Schema:'. To the right of these fields are descriptive labels. At the bottom, there are four buttons: 'Configure Server Management...', 'Test Connection' (highlighted with a red box), 'Cancel', and 'OK' (highlighted with a blue box).

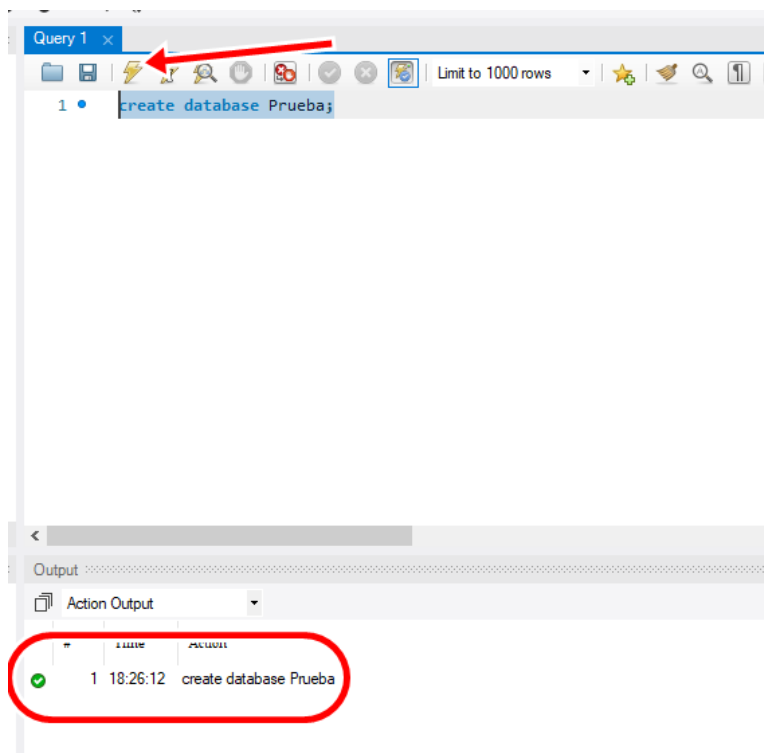
Después de esto se nos mostrara el ambiente de trabajo como el siguiente



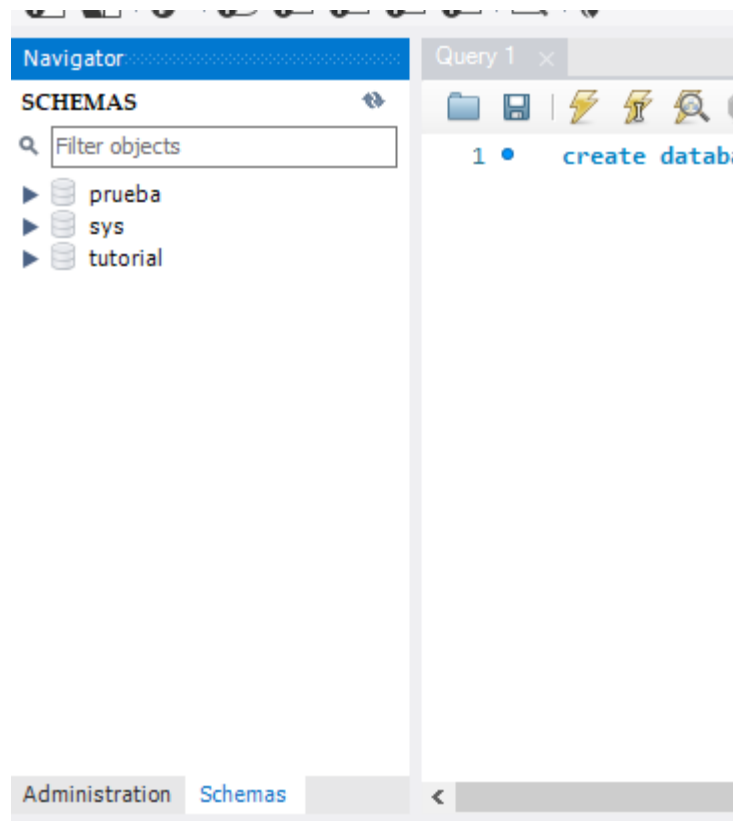
## Cómo ejecutar instrucciones en MySQL

En caso de querer ejecutar todo el Query, simplemente hacemos click en el rayito amarillo, pero en caso de querer ejecutar solamente alguna o algunas instrucciones en específico las elegimos o marcamos con el mouse e igual click en el rayito.

Abajo nos aparecerá un mensaje que indica que fue hecho correctamente.

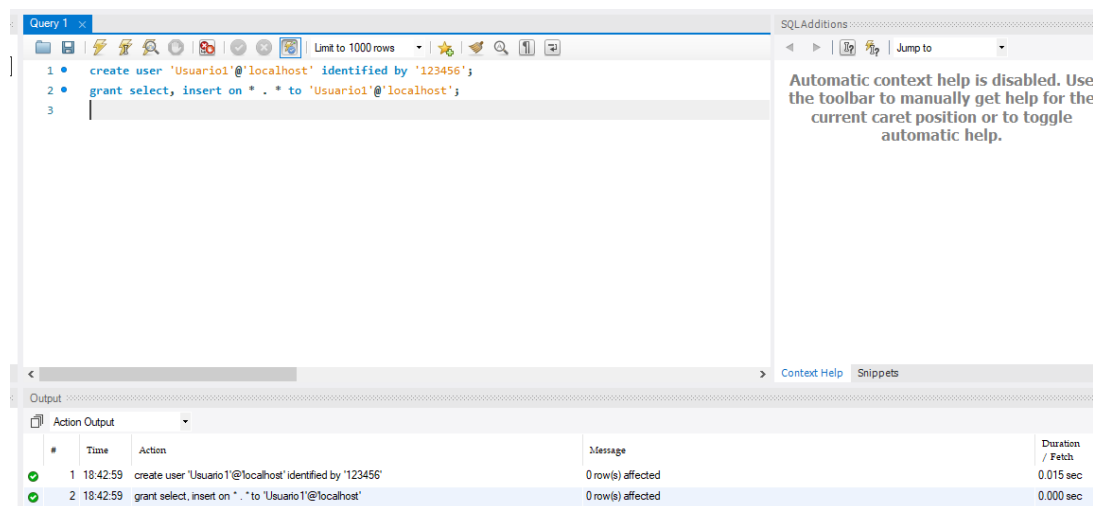


Podemos comprobar que la base de datos se creó, mirando el Navegador



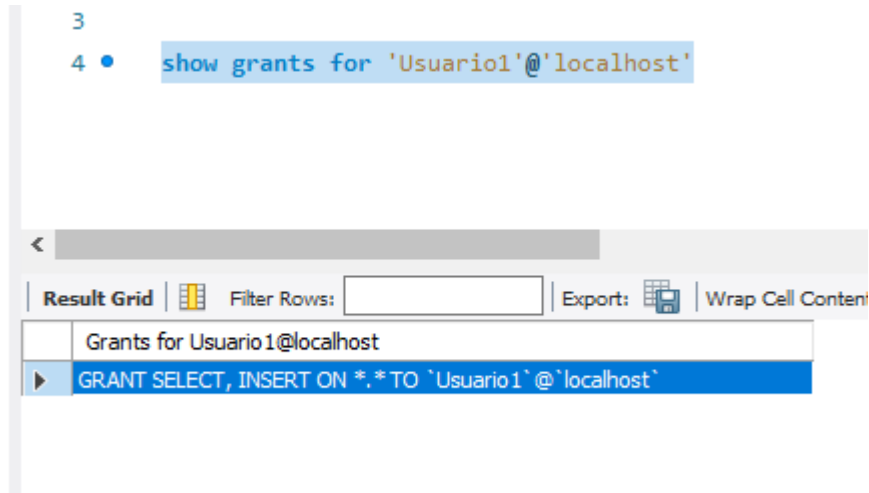
## 6- CREACIÓN DE USUARIOS Y OTORGARLE PERMISO DE EJECUTAR SOLO SELECT/INSERT.

Primeramente creamos el usuario y le damos contraseña, abajo en una sola línea le damos ambos permisos que son para Seleccionar e Insertar. Podemos comprobar que se ejecutaron ambas líneas correctamente viendo el Output.



```
create user 'Usuario1'@'localhost' identified by '123456';  
grant select, insert on * . * to 'Usuario1'@'localhost';
```

Comprobamos que tiene los permisos el usuario:



## 7- SCRIPT PARA CARGAR LA BASE DE DATOS NORTHWIND (RESUMEN CON LOS 20 TIPOS DE DATOS QUE CAMBIARON)

Investigamos y para poder ejecutar la Northwind en MySQL tuvimos que hacer varios cambios, entre ellos:

- Las comillas dobles las cambiamos por comillas simples de las invertidas, por ejemplo de create table “employees” pasamos a create table `employees`
- Ya no usamos el go para separar lotes, ahora usamos ; punto y coma al terminar cada línea
- La palabra identity la sustituimos por auto\_increment
- La declaración de llaves primarias y foráneas se hace de otra manera. En SQL server pondríamos CONSTRAINT "PK\_Employees" PRIMARY KEY CLUSTERED ("EmployeeID"), en cambio en MySQL ponemos CONSTRAINT `PK\_Employees` PRIMARY KEY (`EmployeeID`). Es decir, nos quitamos la palabra Clustered y la sintaxis queda un poco más corta.

Son diferencias de sintaxis más que nada pero es muy similar a la de SQL Server solo que de manera más corta y simple.

Script de la creación de las tablas (no incluye todo el llenado de datos):

```
drop database if exists northwind;
```

```
create database if not exists northwind;
```

```
use northwind;
```

```
CREATE TABLE `Categories` (  
    `CategoryID` INTEGER NOT NULL AUTO_INCREMENT,  
    `CategoryName` VARCHAR(15) NOT NULL,  
    `Description` MEDIUMTEXT,  
    `Picture` LONGBLOB,  
    CONSTRAINT `PK_Categories` PRIMARY KEY (`CategoryID`)  
);
```

```
CREATE INDEX `CategoryName` ON `Categories` (`CategoryName`);
```

```
CREATE TABLE `CustomerCustomerDemo` (  
    `CustomerID` VARCHAR(5) NOT NULL,  
    `CustomerTypeID` VARCHAR(10) NOT NULL,  
    CONSTRAINT `PK_CustomerCustomerDemo` PRIMARY KEY (`CustomerID`,  
    `CustomerTypeID`)  
);
```

```
CREATE TABLE `CustomerDemographics` (  
    `CustomerTypeID` VARCHAR(10) NOT NULL,  
    `CustomerDesc` MEDIUMTEXT,  
    CONSTRAINT `PK_CustomerDemographics` PRIMARY KEY (`CustomerTypeID`)  
);
```

```
CREATE TABLE `Customers` (  
    `CustomerID` VARCHAR(5) NOT NULL,  
    `CompanyName` VARCHAR(40) NOT NULL,  
    `ContactName` VARCHAR(30),  
    `ContactTitle` VARCHAR(30),  
    `Address` VARCHAR(60),  
    `City` VARCHAR(15),  
    `Region` VARCHAR(15),  
    `PostalCode` VARCHAR(10),  
    `Country` VARCHAR(15),  
    `Phone` VARCHAR(24),  
    `Fax` VARCHAR(24),  
    CONSTRAINT `PK_Customers` PRIMARY KEY (`CustomerID`)  
);
```

```
CREATE INDEX `City` ON `Customers` (`City`);
```

```
CREATE INDEX `CompanyName` ON `Customers` (`CompanyName`);
```

```
CREATE INDEX `PostalCode` ON `Customers` (`PostalCode`);
```

```
CREATE INDEX `Region` ON `Customers` (`Region`);
```

```
CREATE TABLE `Employees` (  
    `EmployeeID` INTEGER NOT NULL AUTO_INCREMENT,  
    `LastName` VARCHAR(20) NOT NULL,  
    `FirstName` VARCHAR(10) NOT NULL,  
    `Title` VARCHAR(30),
```

```

`TitleOfCourtesy` VARCHAR(25),
`BirthDate` DATETIME,
`HireDate` DATETIME,
`Address` VARCHAR(60),
`City` VARCHAR(15),
`Region` VARCHAR(15),
`PostalCode` VARCHAR(10),
`Country` VARCHAR(15),
`HomePhone` VARCHAR(24),
`Extension` VARCHAR(4),
`Photo` LONGBLOB,
`Notes` MEDIUMTEXT NOT NULL,
`ReportsTo` INTEGER,
`PhotoPath` VARCHAR(255),
`Salary` FLOAT,
CONSTRAINT `PK_Employees` PRIMARY KEY (`EmployeeID`)
);

```

```

CREATE INDEX `LastName` ON `Employees` (`LastName`);

```

```

CREATE INDEX `PostalCode` ON `Employees` (`PostalCode`);

```

```

CREATE TABLE `EmployeeTerritories` (
  `EmployeeID` INTEGER NOT NULL,
  `TerritoryID` VARCHAR(20) NOT NULL,
  CONSTRAINT `PK_EmployeeTerritories` PRIMARY KEY (`EmployeeID`,
  `TerritoryID`)
);

```

```
CREATE TABLE `Order Details` (  
    `OrderID` INTEGER NOT NULL,  
    `ProductID` INTEGER NOT NULL,  
    `UnitPrice` DECIMAL(10,4) NOT NULL DEFAULT 0,  
    `Quantity` SMALLINT(2) NOT NULL DEFAULT 1,  
    `Discount` REAL(8,0) NOT NULL DEFAULT 0,  
    CONSTRAINT `PK_Order Details` PRIMARY KEY (`OrderID`, `ProductID`)  
);
```

```
CREATE TABLE `Orders` (  
    `OrderID` INTEGER NOT NULL AUTO_INCREMENT,  
    `CustomerID` VARCHAR(5),  
    `EmployeeID` INTEGER,  
    `OrderDate` DATETIME,  
    `RequiredDate` DATETIME,  
    `ShippedDate` DATETIME,  
    `ShipVia` INTEGER,  
    `Freight` DECIMAL(10,4) DEFAULT 0,  
    `ShipName` VARCHAR(40),  
    `ShipAddress` VARCHAR(60),  
    `ShipCity` VARCHAR(15),  
    `ShipRegion` VARCHAR(15),  
    `ShipPostalCode` VARCHAR(10),  
    `ShipCountry` VARCHAR(15),  
    CONSTRAINT `PK_Orders` PRIMARY KEY (`OrderID`)  
);
```

```
CREATE INDEX `OrderDate` ON `Orders` (`OrderDate`);
```



```
CREATE INDEX `ShippedDate` ON `Orders` (`ShippedDate`);
```

```
CREATE INDEX `ShipPostalCode` ON `Orders` (`ShipPostalCode`);
```

```
CREATE TABLE `Products` (  
  `ProductID` INTEGER NOT NULL AUTO_INCREMENT,  
  `ProductName` VARCHAR(40) NOT NULL,  
  `SupplierID` INTEGER,  
  `CategoryID` INTEGER,  
  `QuantityPerUnit` VARCHAR(20),  
  `UnitPrice` DECIMAL(10,4) DEFAULT 0,  
  `UnitsInStock` SMALLINT(2) DEFAULT 0,  
  `UnitsOnOrder` SMALLINT(2) DEFAULT 0,  
  `ReorderLevel` SMALLINT(2) DEFAULT 0,  
  `Discontinued` BIT NOT NULL DEFAULT 0,  
  CONSTRAINT `PK_Products` PRIMARY KEY (`ProductID`)  
);
```

```
CREATE INDEX `ProductName` ON `Products` (`ProductName`);
```

```
CREATE TABLE `Region` (  
  `RegionID` INTEGER NOT NULL,  
  `RegionDescription` VARCHAR(50) NOT NULL,  
  CONSTRAINT `PK_Region` PRIMARY KEY (`RegionID`)  
);
```

```
CREATE TABLE `Shippers` (  
  `ShipperID` INTEGER NOT NULL,  
  `ShipperName` VARCHAR(40) NOT NULL,  
  `Address` VARCHAR(100) NOT NULL,  
  `City` VARCHAR(40) NOT NULL,  
  `Region` VARCHAR(40) NOT NULL,  
  `PostalCode` VARCHAR(10) NOT NULL,  
  `Phone` VARCHAR(20) NOT NULL,  
  CONSTRAINT `PK_Shippers` PRIMARY KEY (`ShipperID`)  
);
```

```
`ShipperID` INTEGER NOT NULL AUTO_INCREMENT,  
`CompanyName` VARCHAR(40) NOT NULL,  
`Phone` VARCHAR(24),  
CONSTRAINT `PK_Shippers` PRIMARY KEY (`ShipperID`)  
);
```

```
CREATE TABLE `Suppliers` (  
  `SupplierID` INTEGER NOT NULL AUTO_INCREMENT,  
  `CompanyName` VARCHAR(40) NOT NULL,  
  `ContactName` VARCHAR(30),  
  `ContactTitle` VARCHAR(30),  
  `Address` VARCHAR(60),  
  `City` VARCHAR(15),  
  `Region` VARCHAR(15),  
  `PostalCode` VARCHAR(10),  
  `Country` VARCHAR(15),  
  `Phone` VARCHAR(24),  
  `Fax` VARCHAR(24),  
  `HomePage` MEDIUMTEXT,  
  CONSTRAINT `PK_Suppliers` PRIMARY KEY (`SupplierID`)  
);
```

```
CREATE INDEX `CompanyName` ON `Suppliers` (`CompanyName`);
```

```
CREATE INDEX `PostalCode` ON `Suppliers` (`PostalCode`);
```

```
CREATE TABLE `Territories` (  
  `TerritoryID` VARCHAR(20) NOT NULL,
```

```

`TerritoryDescription` VARCHAR(50) NOT NULL,
`RegionID` INTEGER NOT NULL,
CONSTRAINT `PK_Territories` PRIMARY KEY (`TerritoryID`)
);

```

## 8- SCRIPT PARA CREAR LA FAMILIA DE VISTAS DE LA BASE DE DATOS NORTHWIND

```

create view vw_products as #Vista Products

select p.productid, p.productname, p.quantityperunit, p.unitprice as produnitprice,
p.unitsinstock, p.unitsonorder, p.reorderlevel, p.discontinued,
s.supplierid, s.companyname, s.contactname, s.contacttitle, s.address, s.city, s.region,
s.postalcode, s.country, s.phone, s.fax, s.homepage, c.categoryid, c.categoryname,
c.description #Ya no existe el campo c.picture por ser de tipo image por lo tanto no lo
ponemos

from products p

inner join suppliers s on p.supplierid = s.supplierid

inner join categories c on p.categoryid = c.categoryid

;#Sustituimos go por punto y coma

```

```

create view vw_orders as #Vista Orders

select o.orderid, o.orderdate, o.requireddate, o.shippeddate, o.freight, o.shipname,
o.shipaddress, o.shipcity,
o.shipregion, s.shipperid, s.companyname as nomcomenvio, s.phone as envphone,
c.customerid, c.companyname as nomcliente,
c.contactname as ctecontactname, c.contacttitle as ctecontacttitle,
c.address as cteaddress, c.city as ctecity, c.region as cteregion, c.postalcode as ctepostalcode,
c.country as ctecountry, c.phone as ctephone, c.fax as ctefax, e.employeeid, e.lastname,
e.firstname,
e.title, e.titleofcourtesy, e.birthdate, e.hiredate,
e.address as empaddress, e.city as empcity, e.region as empregion, e.postalcode as
emppostalcode,

```

e.country as empcountry, e.homephone, e.extension, e.reportsto #Ya no existen los campos e.photo, e.notes, e.photopath por ser de tipo image por lo tanto no los ponemos

from orders o

inner join shippers s on o.shipvia = s.shipperid

inner join customers c on o.customerid = c.customerid

inner join employees e on o.employeeid = e.employeeid

;

create view vw\_order\_details as #Vista Order\_Details

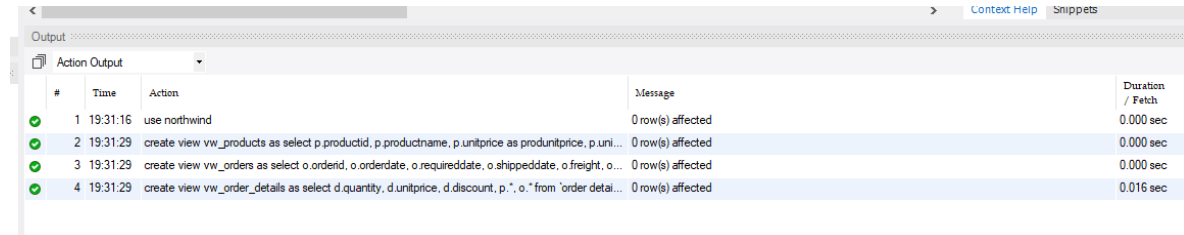
select d.quantity, d.unitprice, d.discount, p.\*, o.\* #Todos los campos de las vistas vw\_products y vw\_orders

from `order details` d #Ya no se pone entre [] como en sql server, ahora es con ``

inner join vw\_products p on d.productid = p.productid

inner join vw\_orders o on d.orderid = o.orderid

;



#	Time	Action	Message	Duration / Fetch
1	19:31:16	use northwind	0 row(s) affected	0.000 sec
2	19:31:29	create view vw_products as select p.productid, p.productname, p.unitprice as produnitprice, p.uni...	0 row(s) affected	0.000 sec
3	19:31:29	create view vw_orders as select o.orderid, o.orderdate, o.requireddate, o.shippeddate, o.freight, o...	0 row(s) affected	0.000 sec
4	19:31:29	create view vw_order_details as select d.quantity, d.unitprice, d.discount, p.*, o.* from 'order detail...	0 row(s) affected	0.016 sec

## 9- SCRIPT CON EL PROCEDIMIENTO ALMACENADO QUE INSERTE Y MODIFIQUE LA TABLA TERRITORIES

delimiter \$\$

create procedure sp\_mttoterritories (out territoryid nvarchar (40), territorydescription nchar (100),regionid int)

begin

if exists (select \* from territories where territoryid = territory)

then

update territories set territoryid = territory, territorydescription = territorydes, regionid = region

where territoryid = territory;

```

elseif error<>0
then
    signal sqlstate '02000' set message_text = 'error al actualizar en la tabla
territories';

else
    select @territoryid = coalesce(max(@territoryid),0) + 1 from territories;
    insert territories(territoryid, territorydescription,regionid)
    values (@territoryid,@territorydescription,@regionid);

if error<>0
then
    signal sqlstate '02000' set message_text = 'error al actualizar en la tabla
territories';

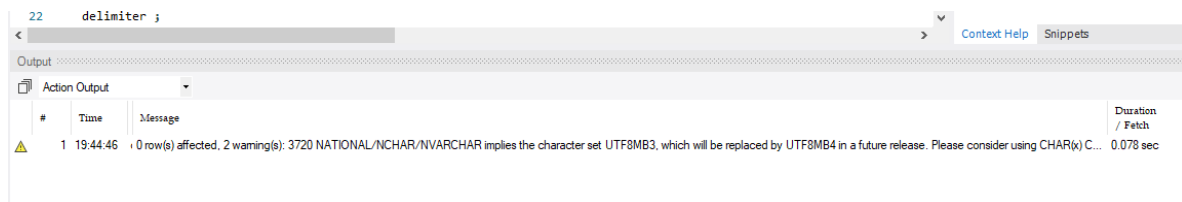
end if;

end if;

end $$

```

delimiter ;#ahora sí cerramos el procedimiento almacenado con la palabra delimiter y su punto y coma



## 10- SCRIPT CON UN TRIGGER QUE NO PERMITA ELIMINAR TERRITORIES

```

delimiter $$

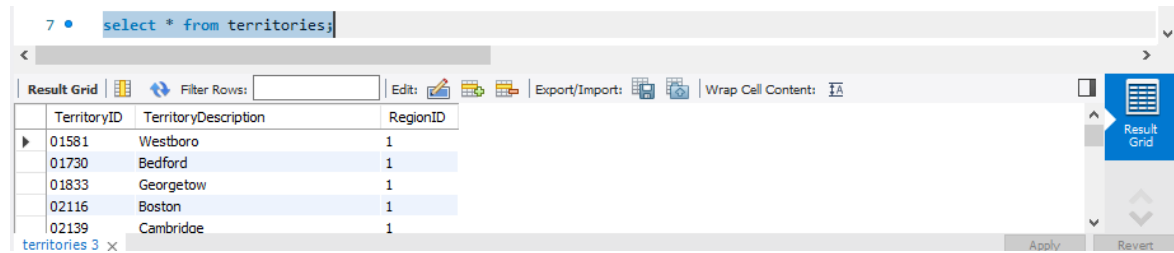
create trigger tr_eliminar_territories before delete on territories for each row
begin

    signal sqlstate '02000' set message_text = 'por el momento no se puede
eliminar registros de territories';

```

```
end $$;
```

```
delimiter ;
```



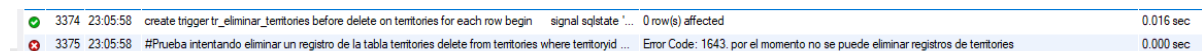
The screenshot shows a SQL query window with the text `select * from territories;`. Below the query, a 'Result Grid' displays the data from the 'territories' table. The grid has three columns: TerritoryID, TerritoryDescription, and RegionID. The data is as follows:

TerritoryID	TerritoryDescription	RegionID
01581	Westboro	1
01730	Bedford	1
01833	Georgetow	1
02116	Boston	1
02139	Cambridge	1

Consulta en la tabla Territories, intentaremos eliminar el registro con TerritoryId = 01581:

```
delete from territories
```

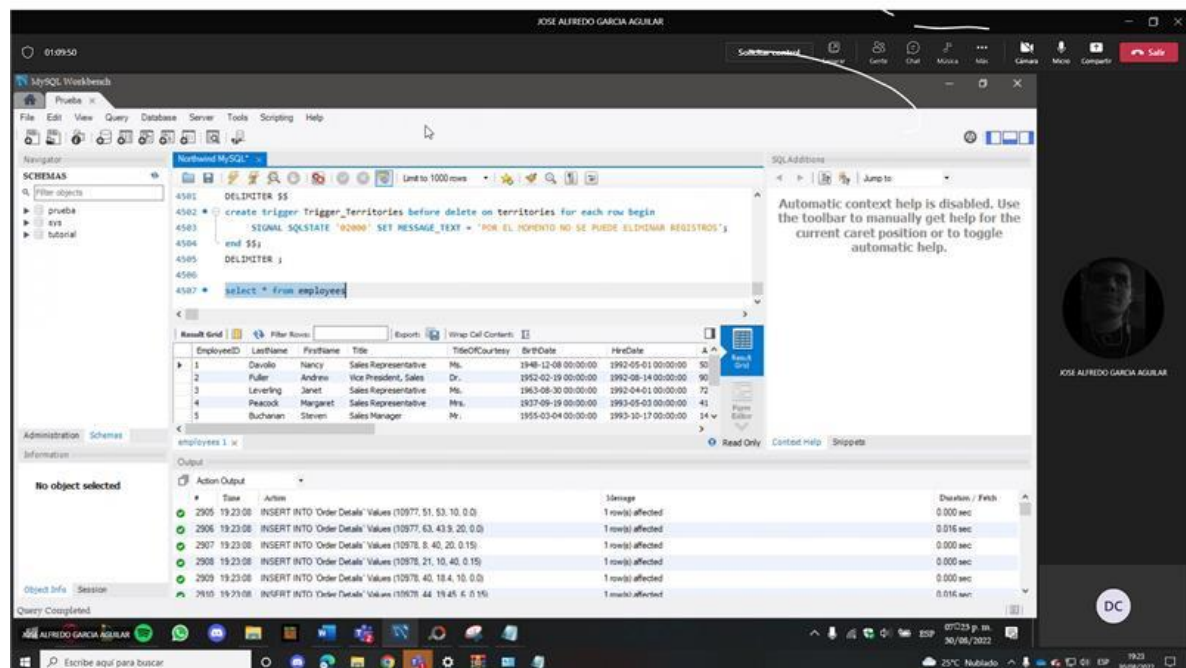
```
where territoryid = 01581;
```



The screenshot shows the execution log of a SQL query. The log contains two entries:

Line	Time	Text	Signal	State	Message	Duration
3374	23:05:58	create trigger tr_eliminar_territories before delete on territories for each row begin	signal	sqlstate '...	0 row(s) affected	0.016 sec
3375	23:05:58	#Prueba intentando eliminar un registro de la tabla territories delete from territories where territoryid = 01581;	Error Code: 1643		por el momento no se puede eliminar registros de territories	0.000 sec

Evidentemente no nos deja eliminar información de la tabla Territories y a la vez nos pone el aviso que creamos en el trigger de arriba.



The screenshot shows the MySQL Workbench interface. The main window displays a SQL query window with the following text:

```
DELIMITER $$
create trigger Trigger_Territories before delete on territories for each row begin
    SIGNAL SQLSTATE '00000' SET MESSAGE_TEXT = 'POR EL MOMENTO NO SE PUEDE ELIMINAR REGISTROS';
end $$
DELIMITER ;

select * from employees
```

Below the query, a 'Result Grid' displays the data from the 'employees' table. The grid has five columns: EmployeeID, LastName, FirstName, Title, and HireDate. The data is as follows:

EmployeeID	LastName	FirstName	Title	HireDate
1	Davolio	Nancy	Sales Representative	1992-12-08 00:00:00
2	Fuller	Andrew	Vice President, Sales	1992-08-14 00:00:00
3	Levinson	Janet	Sales Representative	1992-08-30 00:00:00
4	Peacock	Margaret	Sales Representative	1992-09-09 00:00:00
5	Buchanan	Steven	Sales Manager	1995-03-04 00:00:00