1. Tenemos creada la siguiente tabla:

CREATE TABLE EMPLEADOS (

Cod\_Cliente VARCHAR(5) PRIMARY KEY,

Nombre VARCHAR(10),

Apellidos VARCHAR(25),

Sueldo FLOAT);

1. Ahora queremos poner una restricción a sueldo para que tome valores entre 1000 y 1200, ¿cómo lo harías?

* constraint ch\_salario check (salario between >= 1000 and <=1200)

1. **¿Cuál de las siguientes afirmaciones no es cierta en una relación(=tabla)?**

* Todos los atributos deben estar en el mismo dominio.
* No puede haber dos tuplas completamente iguales.
* Cada atributo de la tabla toma un único valor en cada tupla.
* Podemos tener tablas con el mismo nombre en la misma base de datos.

**4.Relaciona estos términos utilizados para las restricciones en la creación de tablas con su significado o función:**

Principio del formulario

| **Términos.** | **Relación.** | **Función.** |
| --- | --- | --- |
| CHECK | check | 1. Comprueba que los valores que se introducen son los adecuados para un campo. |
| DEFAULT | fk | 2. Designa a un campo como clave ajena. |
| PRIMARY KEY | Not null | 3. Impide que un campo pueda contener valores nulos |
| FOREIGN KEY | unique | 4. Impide que se repitan valores para un campo. |
| NOT NULL | pk | 5. Designa a un campo como clave principal. |
| UNIQUE | default | 6. Incluye un valor en un campo de forma predeterminada. |

Final del formulario

5.Tenemos creada la siguiente tabla:

CREATE TABLE EMPLEADOS (

Cod\_Cliente VARCHAR(5) PRIMARY KEY,

Nombre VARCHAR(10),

Apellidos VARCHAR(25),

Sueldo FLOAT);

Crea un índice con el campo Apellidos, luego elimínalo.

* create index idx\_apellidos on empleados(apellidos);
* drop index idx\_apellidos on empleados;

**¿Qué es un índice?//** **¿Para qué sirve un índice?**

1. **Mejora la velocidad de búsqueda: Un índice permite realizar búsquedas más rápidas al reducir el número de registros que deben ser examinados.**
2. **Facilita la ordenación de datos: Los índices proporcionan un orden lógico de los datos, lo que puede ser beneficioso para consultas que requieren resultados ordenados.**
3. **Optimiza la eficiencia de las consultas: Al utilizar índices, las consultas suelen ser más eficientes y requieren menos recursos del sistema.**
4. **Acelera las operaciones de JOIN: En operaciones de JOIN, donde se combinan datos de varias tablas, los índices pueden mejorar significativamente el rendimiento.**
5. **Ayuda en la unicidad y la integridad: Los índices pueden utilizarse para garantizar la unicidad de los valores en una columna (clave única) y mantener la integridad referencial entre tablas.**
6. **Es importante tener en cuenta que, si bien los índices mejoran la velocidad de recuperación, también pueden afectar el rendimiento durante las operaciones de escritura (inserciones, actualizaciones y eliminaciones). Por lo tanto, la decisión de crear un índice debe basarse en la naturaleza de las consultas que se realizarán con frecuencia en la base de datos.**
7. Suponemos que tenemos la siguiente tabla creada

**CREATE** **TABLE** usuario (

**id** INT UNSIGNED AUTO\_INCREMENT **PRIMARY** **KEY**,

nombre VARCHAR(25)

);

Y queremos modificar la columna nombre para que pueda almacenar 50 caracteres y además que sea NOT NULL. En este caso usaríamos la sentencia…

**ALTER TABLE usuario**

**MODIFY COLUMN nombre VARCHAR(50) NOT NULL;**

1. Define brevemente el significado de las siglas **SQL**

* **Lenguaje de consulta estructurado.**

1. Define brevemente qué es **MySQL WorkBench**.

* MySQL Workbench es una herramienta visual de diseño, desarrollo y administración de bases de datos para MySQL. Proporciona un entorno gráfico que facilita la creación, modelado, mantenimiento y documentación de bases de datos MySQL

1. ¿Qué diferencia hay entre los tipos de datos DATETIME y TIMESTAMP?

* Ambos tipos de datos, DATETIME y TIMESTAMP, se utilizan para almacenar información de fecha y hora en MySQL, pero hay algunas diferencias clave entre ellos:

1. ***Rango de Almacenamiento:***

* DATETIME: Puede almacenar fechas y horas en el rango '1000-01-01 00:00:00' hasta '9999-12-31 23:59:59'.
* TIMESTAMP: Puede almacenar fechas y horas en el rango '1970-01-01 00:00:01' UTC hasta '2038-01-19 03:14:07' UTC. El límite está relacionado con el problema conocido como el "efecto del año 2038".

1. ***Precision:***

* DATETIME: Almacena fechas y horas con una precisión hasta segundos.
* TIMESTAMP: Almacena fechas y horas con una precisión hasta segundos, pero también tiene una fracción de segundos (microsegundos) para mayor precisión.

1. ***Zonas Horarias:***

* DATETIME: No almacena información de la zona horaria. La fecha y hora se interpretan según la configuración del servidor o la conexión del cliente.
* TIMESTAMP: Almacena la fecha y hora en formato UTC (tiempo coordinado universal) y se convierte al huso horario del servidor al recuperar los datos. Esto permite trabajar con zonas horarias de manera más eficiente.

1. ***Almacenamiento:***

* DATETIME: Utiliza 8 bytes de almacenamiento.
* TIMESTAMP: Utiliza 4 bytes de almacenamiento.

1. ***Valores Predeterminados Automáticos:***

* DATETIME: No puede tener un valor predeterminado automático que utilice la función CURRENT\_TIMESTAMP hasta MySQL 8.0.13.
* TIMESTAMP: Puede tener un valor predeterminado automático utilizando la función CURRENT\_TIMESTAMP, y esto se actualiza automáticamente al realizar cambios en el registro.

En resumen, la elección entre DATETIME y TIMESTAMP dependerá de los requisitos específicos de la aplicación. TIMESTAMP es a menudo preferido para registros de cambios o eventos que requieren seguimiento temporal preciso, mientras que DATETIME es más versátil y puede ser utilizado cuando la información de la zona horaria no es crítica o cuando el rango de almacenamiento más amplio es necesario.

1. ¿Qué puerto es el que utiliza por defecto el servicio de **MySQL Server**?

* **3306**

1. Ejecute el siguiente script para crear una base de datos llamada ventas.

**DROP** **DATABASE** **IF** **EXISTS** ventas;

**CREATE** **DATABASE** ventas CHARACTER **SET** latin1;

**USE** ventas;

**CREATE** **TABLE** cliente (

**id** INT UNSIGNED AUTO\_INCREMENT **PRIMARY** **KEY**,

nombre VARCHAR(25),

primer\_apellido VARCHAR(15) **NOT** **NULL**,

ciudad VARCHAR(100),

categoria INT UNSIGNED

);

**CREATE** **TABLE** comercial (

**id** INT UNSIGNED AUTO\_INCREMENT **PRIMARY** **KEY**,

nombre VARCHAR(100) **NOT** **NULL**,

apellido1 VARCHAR(100) **NOT** **NULL**,

apellido2 VARCHAR(100),

ciudad VARCHAR(100),

comision FLOAT

);

1. Una vez que ha ejecutado el script y ha creado la base de datos en MySQL Server, ¿qué sentencia SQL debería ejecutar para modificar el encoding de la base de datos y utilizar utf8mb4 en lugar de latin1?

**alter database ventas character set utf8mb4 collate utf8mb4\_unicode\_ci;**

1. Modifique la columna nombre de la tabla cliente para que pueda almacenar cadenas de hasta 100 caracteres y para que no pueda ser NULL.

alter table cliente

**modify column nombre varchar(100) not null;**

1. ¿Qué comando puede ejecutar para comprobar que el cambio que se ha realizado en el paso anterior se ha ejecutado correctamente?

**show columns from cliente like 'nombre'; ---esto muestra solo la columna que especificas**

**DESCRIBE cliente; ---esto muestra todas las columnas de la tabla**

1. Modifique el nombre de la columna primer\_apellido y asígnele apellido1. También tendrá que permitir que pueda almacenar hasta 100 caracteres y que no pueda ser un valor NULL.

**alter table cliente**

**change column primer\_apellido apellido1 varchar(100) not null;**

1. Añada una nueva columna a la tabla cliente para poder almacenar el segundo apellido. La columna se debe llamar apellido2 debe estar entre la columna apellido1 y ciudad, puede almacenar hasta 100 caracteres y puede ser NULL.

**alter table cliente**

**add column apellido2 varchar(100) after apellido1 ;**

1. Elimine la columna categoria de la tabla cliente.

**alter table cliente**

**drop column categoria;**

1. Modifique la columna comision de la tabla comercial para que almacene por defecto el valor 10.

**alter table comercial**

**modify column comision float default 10;**

1. Insertar dos clientes nuevos en la tabla clientes.

**insert into cliente (nombre,apellido1, apellido2, ciudad)**

**values ('jonatan','tajada', 'rico', 'barakaldo'),**

**('maria','sanchez', 'perez', 'bilbao');**