**UNIDAD 2: MySQL**

1. [UNIDAD 2: Introducción](https://learn.nextu.com/mod/page/view.php?id=10253&pid=P_WEB_DATABASE)

**MySQL**

Con un enfoque que busca que puedas aplicar los conceptos de bases de datos aprendidos hasta el momento, en esta unidad usaremos MySQL, con la ayuda de la línea de comando y la herramienta gráfica MySQL Workbench, para que te vayas familiarizando con la utilización de los conceptos de bases de datos relacionales y su representación en MySQL. Continuando con nuestro enfoque teórico-práctico, cada lección proporciona un número de talleres, donde tendrás el tiempo y espacio de poder explorar y practicar los comandos y sentencias explicadas en video y lecturas.

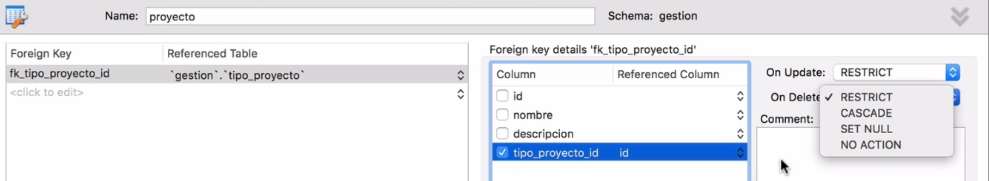
**Objetivos de aprendizaje**

1. Crear **Triggers** y **Procedures** en MySQL para operar nuestra base de datos de forma automática.
2. Crear claves primarias y foráneas y sus restricciones a una solución de base de datos en MySQL.
3. Usar las sentencias SQL **GROUP BY**, **HAVING** y **JOIN** en MySQL para mezclar tablas de diversas maneras.
4. Diferenciar los distintos tipos de funciones predefinidas en MySQL con la finalidad de poder realizar operaciones más avanzadas sobre nuestra base de datos.
5. Crear queries en MySQL con la finalidad de realizar consultas a bases de datos de mediana complejidad.
6. Lección 1: Merge
   1. [Creando Tablas con Índices y Claves Foráneas](https://learn.nextu.com/mod/lesson/view.php?id=10254&pid=P_WEB_DATABASE)

Recomendado utf8 – Ya que permite diversidad de caracteres. (default collation)

Create Scheme o create database, en MySQL funciona

Restricciones de llaves foráneas en actualización o en eliminación:

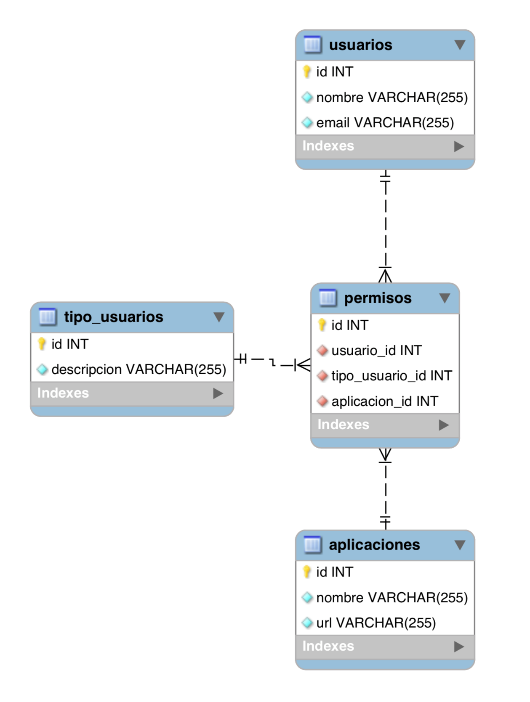


* 1. Taller 1

**Aprende haciendo**

**Taller de Creación de Índices: Usuarios de la Organización**

¡Bienvenido! La idea de este taller es que puedas practicar con índices y claves foráneas.  
  
Dadas las siguientes tablas:



Donde:

* **usuarios:** Representa a los usuarios de una empresa.
* **aplicaciones:** Representa las aplicaciones que manejan internamente.
* **tipo\_usuarios:** Es una tabla con los posibles tipos de usuarios, por ejemplo, Administrador, Analista, etc.
* **permisos:** Asociación que indica el tipo de usuario en una aplicación particular.

Se desea que ejecutes los siguientes pasos usando la herramienta MySQL Workbench:

1. Crear las tablas según el diagrama.
2. Crear un índice de unicidad para el campo email en la tabla usuarios.
3. Crear las claves foráneas de la tabla permisos definiendo un constrain CASCADE para ON UPDATE y ON DELETE
   * fk\_usuario\_id
   * fk\_tipo\_usuario\_id
   * fk\_aplicacion\_id
4. Crear los índices para cada una de las claves foráneas.
   * ix\_usuario\_id
   * ix\_tipo\_usuario\_id
   * ix\_aplicacion\_id

**Puntos de verificación**

Para confirmar que tu experiencia ha sido llevada a cabo con éxito, revisa los siguientes puntos:

1. Verifica las tablas: puedes usar la consulta SHOW TABLES para ver las tablas creadas usando el terminal
2. Verifica el índice de unicidad: Intenta crear dos registros con el mismo email y MySQL no debe dejar guardar el segundo registro.
3. Verifica clave foránea: Cambia los ids de la tabla tipo\_usuarios y verifica que cambiaron en la tabla usuario
4. Índices creados: Verifica que tengas índices, puedes hacerlo con la consulta DESCRIBE permisos

Es muy importante definir las claves foráneas y los índices para mantener la consistencia de la base de datos y tener buen rendimiento del sistema.

* 1. JOINS en MySQL

Tienes 2 tipos adicionales de JOIN:

LEFT JOIN – Ejemplo:

SELECT \* FROM lista\_a LEFT JOIN lista\_b ON lista\_a.email = lista\_b.email

En este ejemplo deja a lista\_a como una columna fija

RIGHT JOIN – deja fija la lista de lista\_b

* 1. ¿Sabías qué? -1

**Cápsula de conocimiento**

**Tipos de Datos en MySQL**

¡Bienvenido! En esta lectura te proporcionamos un resumen de los tipos de datos más utilizados en MySQL, categorizados en tres grupos:



**Strings**

| **Tipo** | **Descripción** |
| --- | --- |
| **CHAR(largo)** | Secuencia de caracteres (string) de tamaño fijo hasta 255 |
| **VARCHAR(largo)** | Secuencia de caracteres (string) de tamaño variable hasta 255 |
| **TEXT** | String hasta 65535 caracteres |
| **MEDIUMTEXT** | String hasta 16777215 caracteres |
| **LONGTEXT** | String hasta 4294967295 caracteres |



**Numéricos**

| **Tipo** | **Descripción** |
| --- | --- |
| **TINYINT** | Entero desde -128 a 127 (usado sin signo para booleanos) |
| **INT** | Entero desde -2147483648 a 2147483647 |
| **BIGINT** | Entero desde -9223372036854775808 a 9223372036854775807 |
| **FLOAT** | Decimal Pequeño (4 bytes) |
| **DOUBLE** | Decimal Grande (8 bytes) |



**Otros**

| **Tipo** | **Descripción** |
| --- | --- |
| **DATE** | Fecha en formato YYYY-MM-DD |
| **DATETIME** | Fecha y hora en formato YYYY-MM-DD HH:MM:SS |
| **TIME** | Hora en formato HH:MM:SS |
| **ENUM** | Enumerado |

Esperamos que esta guía te sea útil a la hora de escoger tus datos en MySQL.

* 1. ¿Sabías qué? – 2

**Cápsula de conocimiento**

**Resumen de Comandos**

En esta lectura encontrarás un resumen de los comandos de MySQL para crear tablas con sus claves e índices, así como también de la sentencia Join para unir y comparar tablas basados en el uso de ciertos campos de nuestras tablas.

* **Índices:**

CREATE TABLE [tabla] ([col 1] [tipo 1],....[col 2] [tipo 2],PRIMARY KEY ([campo\_pk])KEY [key\_name1] ([campos2]),...KEY [key\_name2] ([campos2]),)

* **Claves Foráneas:**

CREATE TABLE [tabla] ([col 1] [tipo 1],....[col 2] [tipo 2],PRIMARY KEY ([campo\_pk])KEY [key\_name1] ([campos2]),...CONSTRAIN [nombre\_fk1] FOREIGN KEY ([campo\_origen1]) REFERENCES [tabla destino1]([campo\_destino1]),CONSTRAIN [nombre\_fk2] FOREIGN KEY ([campo\_origen2]) REFERENCES [tabla destino2]([campo\_destino2]))

* **Left Join:**

SELECT \* FROM [tabla\_izquierda] LEFT JOIN [tabla\_derecha] ON [condición que une las tablas]

* **Right Join:**

SELECT \* FROM [tabla\_izquierda] RIGHT JOIN [tabla\_derecha] ON [condición que une las tablas]

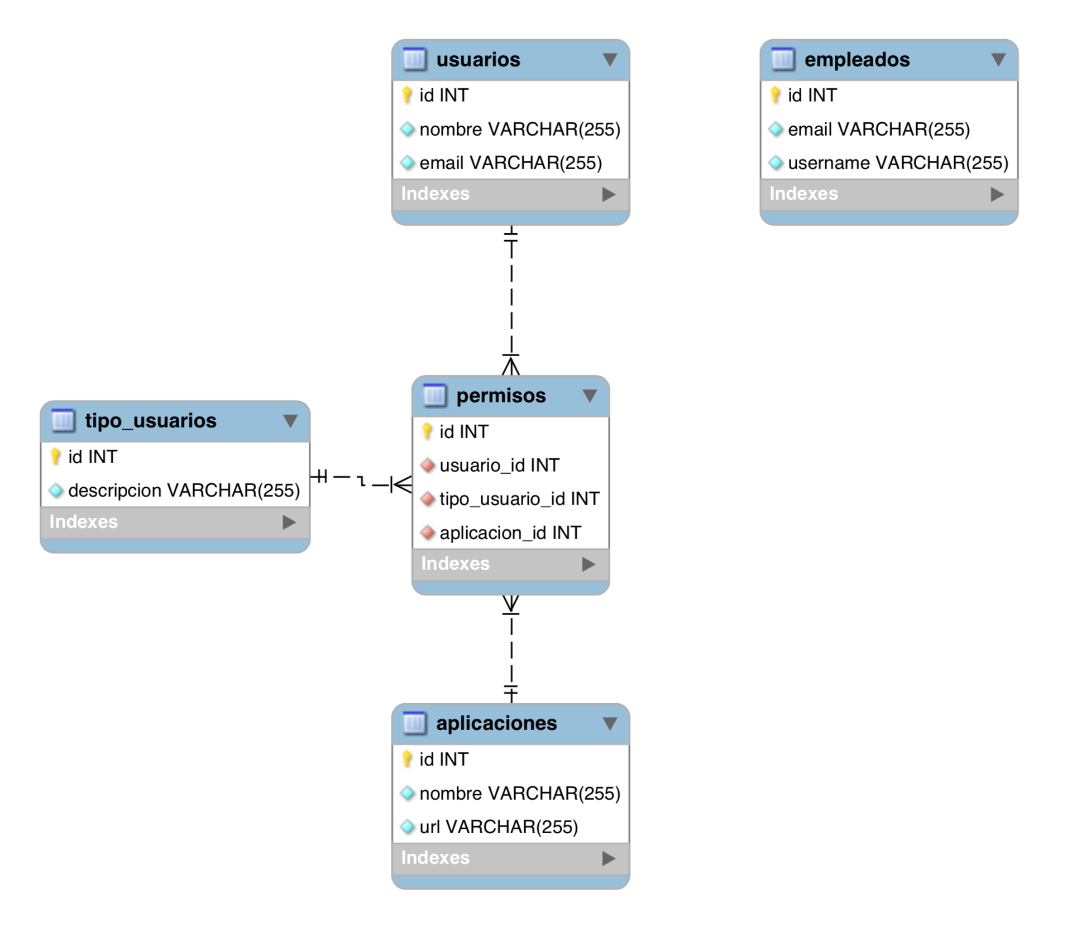
Recuerda que hay variantes de estas instrucciones, que hemos omitido por simplicidad pero que puedes encontrar en el sitio oficial de MySQL.   
Esperamos que esta guía te sea útil en la construcción y manejo de tus tablas.

* 1. Taller 2

**Aprende haciendo**

**Taller de LEFT JOIN y RIGHT JOIN: Usuarios de la Organización**

¡Bienvenido! Hemos diseñado este taller para que puedas practicar con la sentencia Join, usando la siguiente base de datos:



Donde:

* **usuarios:** Representa a los usuarios de una empresa.
* **aplicaciones:** Representa las aplicaciones que manejan internamente.
* **tipo\_usuarios:** Es una tabla con los posibles tipos de usuarios, por ejemplo, Administrador, Analista, etc.
* **permisos:** Asociación que indica el tipo de usuario en una aplicación particular.
* **empleados:** representa una tabla del sistema central antiguo.

Se desea que ejecutes los siguientes pasos usando la herramienta MySQL Workbench:

1. Crear las tablas según el diagrama.
2. Llenar las tablas con valor de ejemplo.
   * Crear 3 usuarios en la tabla usuarios (luis, maría, pedro)
   * Crear 2 aplicaciones (app1, app2)
   * Crear 2 tipo\_usuarios (admin, analista)
   * Crear permisos
   * Crear 4 empleados (luis, maría, ana, camila)
3. Hacer una consulta con LEFT JOIN para ver cuáles usuarios están en la tabla empleados y no están en la tabla usuarios a través del correo electrónico.
4. Hacer una consulta con RIGHT JOIN para ver cuáles usuarios están en la tabla empleados y no están en la tabla usuarios a través del correo electrónico.
5. Hacer una consulta con JOIN para ver cuáles aplicaciones están permitidas para los usuarios.

**Puntos de verificación**

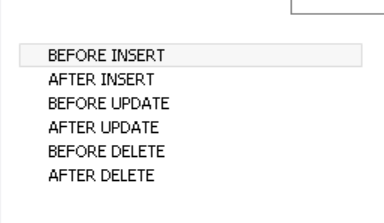
Para confirmar que tu experiencia ha sido llevada a cabo con éxito, revisa los siguientes puntos:

1. Verifica las tablas: puedes usar la consulta SHOW TABLES para ver las tablas creadas usando el terminal
2. Verificar datos: Puedes hacer un SELECT de cada tabla para verificar el contenido
3. LEFT JOIN: El resultado debe mostrar: ana y camila
4. RIGHT JOIN: El resultado debe mostrar: pedro
5. JOIN: Verifica la información de tu tabla permisos con tu consulta JOIN

La sentencia **JOIN** es comúnmente utilizada en las operaciones realizadas sobre una base de datos, de allí la importancia de que te familiarices con ella.

1. Lección 2: Triggers y Procedures
   1. [Triggers](https://learn.nextu.com/mod/lesson/view.php?id=10255&pid=P_WEB_DATABASE)

Triggers



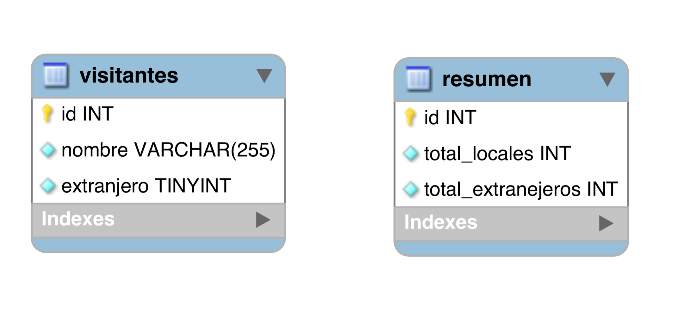
NEW o OLD, para insertar el nuevo o el registro anterior. Esto podría generar un historial

* 1. [Taller 1](https://learn.nextu.com/mod/lesson/view.php?id=10255&pageid=46621&pid=P_WEB_DATABASE)

**Aprende haciendo**

**Taller de Triggers: Visitantes de una atracción turística**

El sistema ABC mantiene un registro de las personas que visitan una atracción turística muy conocida en la ciudad de un país maravilloso. Las tablas que usa el sistema son:



Donde:

* **visitantes:** guarda el nombre de cada persona y si es extranjero o no. Se guarda como TINYINT donde 0 indica no es extranjero y 1 indica que sí lo es.
* **resumen:** tiene un solo registro con los valores de la cantidad de visitas totales para cada caso. Esta tabla fue inicializada en 0. Representa las aplicaciones que manejan internamente.

**Meta**

Crea un trigger para actualizar la tabla resumen cada vez que se ingrese un registro en la tabla visitantes. Para esto debes crear las tablas e ir llenándolas con valores.

**Tips**

Puedes usar un

**SQL UPDATE**

de la tabla resumen haciendo cada total = total + 1 pero debes colocar un IF para saber cuál total debes afectar. Dentro del cuerpo del trigger puedes poner condicionales, tales como IF, THEN, ELSE, END, IF.

**Puntos de verificación**

Para confirmar que tu experiencia ha sido llevada a cabo con éxito, revisa los siguientes puntos:

1. Verifica las tablas: puedes usar la consulta SHOW TABLES para ver las tablas creadas usando el terminal
2. Crear registros: Agrega varios visitantes
3. Verifica el resumen: Cuenta los visitantes y compara con la tabla resumen

En este taller has podido comprobar la fortaleza de los triggers en la definición de ciertas acciones que se disparan cuando ocurre una condición determinada en el sistema.

Solución:

CREATE DEFINER=`root`@`localhost` TRIGGER `abc`.`visitantes\_AFTER\_INSERT` AFTER INSERT ON `visitantes` FOR EACH ROW

BEGIN

DECLARE locales,extranjeros,totales\_locales,totales\_extranjeros INT;

IF NEW.extranjero = 0 THEN

SET locales = (SELECT total\_locales FROM abc.resumen LIMIT 1);

SET totales\_locales = locales + 1;

UPDATE resumen SET total\_locales = totales\_locales WHERE id=1;

END IF;

IF NEW.extranjero = 1 THEN

SET extranjeros = (SELECT total\_extranjeros FROM abc.resumen LIMIT 1);

SET totales\_extranjeros = extranjeros + 1;

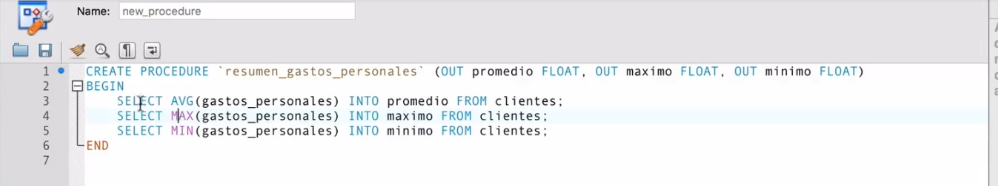
UPDATE resumen SET total\_extranjeros = totales\_extranjeros WHERE id=1;

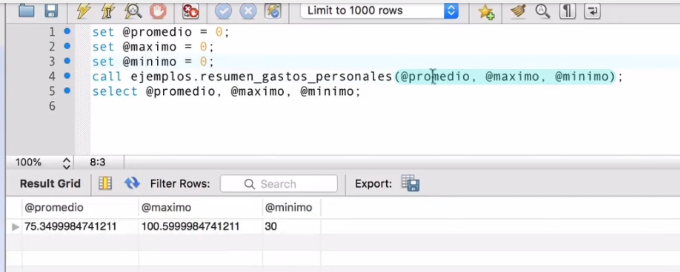
END IF;

END

* 1. [Stored Procedures](https://learn.nextu.com/mod/lesson/view.php?id=10255&pageid=46622&pid=P_WEB_DATABASE)

Ejemplo:



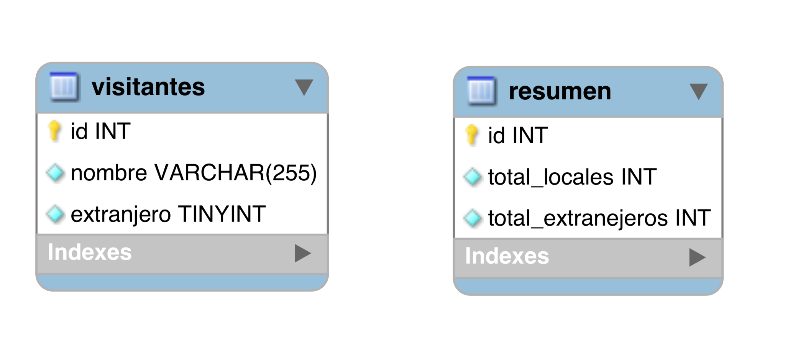


* 1. Taller 2

**Aprende haciendo**

**Taller de Stored Procedures: Visitantes de una atracción turística**

En este taller vamos a seguir utilizando nuestro ejemplo del sistema ABC que guarda el registro de las personas que visitan una atracción turística. Las tablas que usa el sistema son:



Supongamos que, por alguna razón desconocida, los totales se desincronizan y no reflejan el valor real de las visitas. Entonces podemos resolver esto creando un **Stored Procedure** que recalcule estos totales en cualquier momento.

**Meta**

Crea un **Stored Procedure** para actualizar la tabla resumen, dados los registros de la tabla visitantes. Para esto debes crear las tablas e ir llenándola con valores. El **Stored Procedure** debe calcular los totales haciendo un Group by de la tabla visitantes.

**Puntos de verificación**

Para confirmar que tu experiencia ha sido llevada a cabo con éxito, revisa los siguientes puntos:

1. Verifica las tablas: puedes usar la consulta SHOW TABLES para ver las tablas creadas usando el terminal
2. Crear registros: Agrega varios visitantes
3. Verifica el resumen: Coloca los totales en 0, ejecuta el stored procedure, cuenta los visitantes y compara con la tabla resumen.

En este taller has podido comprobar las ventajas de usar **Stored Procedure** en la definición de ciertas acciones que se pueden ejecutar en cualquier momento en el sistema.

**Solución:**

CREATE DEFINER=`root`@`localhost` PROCEDURE `recalcular\_totales`()

BEGIN

DECLARE extranjeros, locales INT;

SET extranjeros = (SELECT COUNT(\*) FROM abc.visitantes WHERE extranjero = 1);

SET locales = (SELECT COUNT(\*) FROM abc.visitantes WHERE extranjero = 0);

UPDATE resumen SET total\_extranjeros = extranjeros, total\_locales = locales WHERE id=1;

END

* 1. ¿Sabías qué?

**Cápsula de conocimiento**

**Resumen de Comandos**

En esta lectura encontrarás un resumen de la sintaxis usada para crear Triggers, que son los mecanismos de MySQL para realizar un conjunto de acciones, luego o antes de una inserción, actualización o borrado de un registro en alguna de nuestras tablas. También te proporcionamos un resumen de la sintaxis usada para crear los Stored Procedures, procedimientos, para la ejecución de ciertas acciones necesarias para automatizar ciertas tareas en nuestra base de datos.

* **Triggers:**

CREATE TRIGGER [nombre\_trigger] [AFTER|BEFORE] [UPDATE|INSERT|DELETE] ON [nombre\_tabla]FOR EACH ROW BEGIN …[cuerpo trigger]END

* **Procedures:**

CREATE PROCEDURE [nombre\_procedure] (parametros) BEGIN …[cuerpo procedure]END

Recuerda que los parámetros se clasifican en parámetros de entrada (IN) y parámetros de salida (OUT), y cada uno de ellos tiene un tipo asociado.

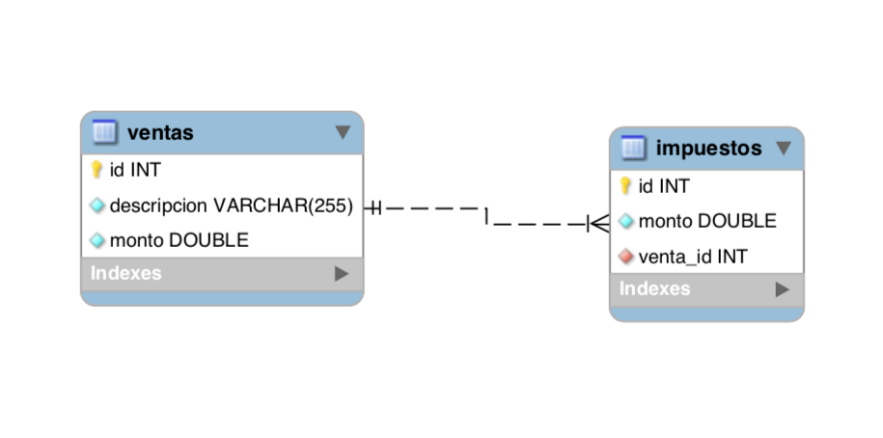
Esperamos que esta guía sea útil para reforzar tu aprendizaje de estos comandos.

* 1. Ejercicio Práctico

**Desafío práctico**

**Registrando Impuestos**

La idea de este ejercicio es que practiques con el uso de Triggers en MySQL. En esta oportunidad queremos que crees una forma automática de calcular y extraer los impuestos de una tabla de ventas al mayor. Para ello debes crear las tablas ventas e impuestos de la siguiente estructura:



Luego de crear la tabla debes definir un trigger, que agregue un registro en la tabla impuestos con el 10% del monto de cada venta nueva, y por último debes probar que tu trigger funciona.

**Solución:**



1. Lección 3: Funciones Predefinidas
   1. [Funciones Predefinidas de String](https://learn.nextu.com/mod/lesson/view.php?id=10256&pid=P_WEB_DATABASE)

CONCAT()

UCASE() 🡪 Upper Case

TRIM()

IF( Condicion, lo que se devuelve , lo que no es verdad)

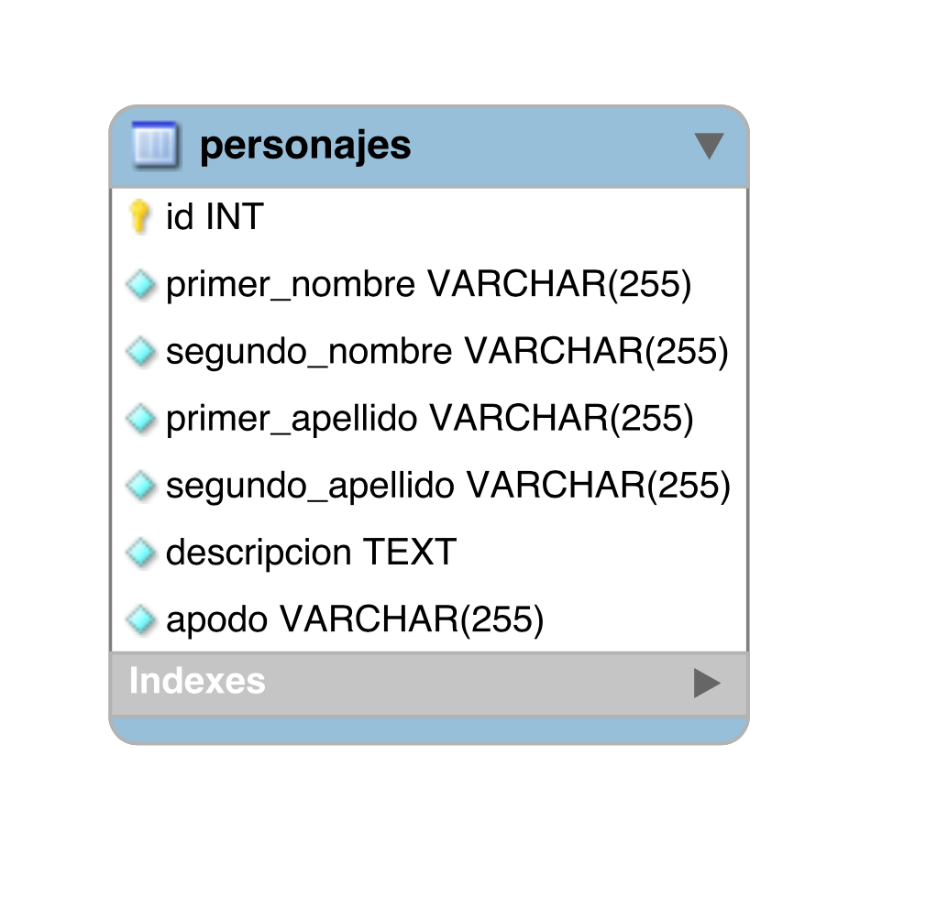
MD5() 🡪 Criptográficos

* 1. Taller 1

**Aprende haciendo**

**Taller: Funciones de Manipulación de String**

¡Bienvenido! En este taller seguimos practicando con las funciones predefinidas de MySQL. En este caso tenemos la tabla personajes del guión de una nueva película. A continuación te mostramos la tabla personajes:



**Meta**

Para practicar con las funciones de String debes crear una tabla con la estructura anterior e insertar varios registros para tu control. Luego realiza las siguientes consultas:

1. Consulta un campo nombre\_completo que contenga (primer\_nombre, segundo\_nombre, primer\_apellido y segundo apellido).
2. Consulta un campo llamado descripcion\_buena que sea la descripción, pero de su contenido reemplaza todas las ocurrencias de la palabra “Yes” por la palabra “Sí”.
3. Consulta los apodos, pero con las siguientes condiciones:
   * El apodo debe devolverse en minúscula.
   * Si el apodo es NULL devolver “N/A”. Nota que esto lo puedes hacer con un condicional IF.

**Puntos de verificación**

Para confirmar que tu experiencia ha sido llevada a cabo con éxito, revisa los siguientes puntos:

1. Verifica la tabla: puedes usar la consulta SHOW TABLES para ver la tabla personajes, usando el terminal
2. Ingresa datos: crea registros con valores diferentes para cada uno de los campos
3. Nombre completo: Verifica que tu consulta coincida con la concatenación de los nombres
4. Descripción buena: Asegúrate que las descripciones tienen la palabra “Yes” y verifica que cuando haces la consulta efectivamente se reemplazan.
5. Apodos: Crea apodos con algunas letras en mayúsculas y otros vacíos, luego al hacer la consulta revisa que se devuelvan en minúsculas o “N/A” cuando aplique.

Esperamos hayas disfrutado de la actividad, pero sobre todo que hayas aprendido a usar las correspondientes funciones predefinidas de MySQL.

Solución:

INSERT INTO taller.personajes VALUES (null,'Jose','Rodrigo','Sancho','Castro','Presente siempre en la cocina','Igo');

INSERT INTO taller.personajes VALUES (null,'Fabiola','María','Sanchez','Rodríguez','Es una sirvienta','Machi');

INSERT INTO taller.personajes VALUES (null,'Raul','Fabricio','Solórzano','Tarantino','Le gusta el Futbol','Ichio');

INSERT INTO taller.personajes VALUES (null,'María','Jose','Miranda','Salas','Es una persona intelectual','Mira');

SELECT CONCAT(primer\_nombre,' ',segundo\_nombre,' ',primer\_appelido,' ',segundo\_apellido) nombre\_completo FROM taller.personajes;

SELECT REPLACE(descripcion,'Yes','Sí') FROM taller.personajes;

SELECT IF(apodo IS NULL,'N/A',LCASE(apodo)) FROM taller.personajes;

* 1. Funciones Matemáticas y Fecha

ROUND()

WHERE fecha BETWEEN ‘fecha\_ini’ AND ‘fecha\_fin’

YEAR()

MONTH()

MONTHNAME()

WEEK()

* 1. Taller 2

**Aprende haciendo**

**Taller de Funciones Matemáticas: Atletas**

¡Bienvenido! En este taller podrás practicar con las funciones predefinidas de MySQL. En este caso tenemos la tabla atletas:



**Meta**

Crear una tabla siguiendo la estructura anterior e insertar varios registros para su control. Adicionalmente, queremos que crees las siguientes consultas:

1. Calcular el promedio de peso de los atletas
2. Calcular la edad del atleta más joven
3. Calcular el atleta más alto.
4. Listar todos los atletas con el porcentaje de carreras ganadas. Este porcentaje debe ser referente a las carreras competidas y debe estar redondeado a dos decimales.

**Puntos de verificación**

Para confirmar que tu experiencia ha sido llevada a cabo con éxito, revisa los siguientes puntos:

1. Verifica la tabla: puedes usar la consulta SHOW TABLES para ver la tabla atletas usando el terminal
2. Ingresa datos: crea registros con valores diferentes para cada uno de los campos
3. Peso promedio de los atletas: Haz el cálculo manualmente y verifica que efectivamente coincide con tu consulta
4. Edad del atleta más joven: Ubica el atleta manualmente y verifica que efectivamente coincide con tu consulta
5. Atleta más alto: Ubica el atleta manualmente y verifica que efectivamente coincide con tu consulta
6. Porcentaje de carreras ganadas: Haz el cálculo manualmente (carreras\_ganadas/carreras\_competidas\*100.0) y verifica que efectivamente coincide con tu consulta

Esperamos hayas disfrutado de la actividad, pero sobre todo que hayas aprendido a usar las correspondientes funciones predefinidas de MySQL.

**Solución:**

SELECT \* from deporte.atletas;

SELECT AVG(peso) Promedio\_peso FROM deporte.atletas;

SELECT MIN(edad) Mas\_joven FROM deporte.atletas;

SELECT ROUND(MAX(estatura),2) Maxima\_estatura FROM deporte.atletas;

SELECT nombre,ROUND(((carreras\_ganadas / carreras\_competidas) \* 100),2) Porcentaje\_carreras\_ganadas FROM deporte.atletas;

* 1. ¿Sabías qué?

**Cápsula de conocimiento**

**Funciones Predefinidas de MySQL**

MySQL proporciona una amplia variedad de funciones predefinidas que permiten manejar los datos de nuestra base de datos. En esta lectura encontrarás un conjunto de las funciones que son consideradas como las más usadas. Esta guía busca ayudarte a familiarizarte con las funciones en MySQL.



**Operadores**

| **Función** | **Descripción** |
| --- | --- |
| **+** | SUMA |
| **-** | RESTA |
| **/** | DIVISIÓN |
| **\*** | MULTIPLICACIÓN |



**Strings**

| **Función** | **Descripción** |
| --- | --- |
| **CONCAT** | Concatenar |
| **REPLACE** | Reemplazar |
| **LCASE** | Cambiar a minúsculas |
| **UCASE** | Cambiar a mayúsculas |
| **TRIM** | Quitar espacios al comienzo y final |



**Matemáticas**

| **Función** | **Descripción** |
| --- | --- |
| **EXP** | Función exponencial |
| **FLOOR** | Piso de un número |
| **POW** | Potencia |
| **ROUND** | Redondeo |
| **SQRT** | Raíz cuadrada |



**Fecha y Hora**

| **Función** | **Descripción** |
| --- | --- |
| **DATE\_FORMAT** | Especificar un formato para fecha / hora |
| **DAY** | Día |
| **MONTH** | Mes |
| **YEAR** | Año |
| **MONTHNAME** | Nombre del mes |

Recuerda que existen más funciones en MySQL y te invitamos a indagar sobre ellas en el sitio oficial de MySQL. Por el momento, esperamos que encuentres beneficiosa esta lectura para tu proceso de aprendizaje de las bases de datos relacionales.

1. Lección 4: Subqueries y Alter table
   1. [Subqueries](https://learn.nextu.com/mod/lesson/view.php?id=10257&pid=P_WEB_DATABASE)

IN()

NOT IN()

* 1. Taller 1

**Aprende haciendo**

**Taller de Subqueries: Noticias**

Una revista digital tiene un puesto importante en el mercado y desea otorgar diferentes premios. Para lo cual queremos que hagas consultas sobre la tabla **noticias**, que tiene la siguiente estructura:



**Primero recuerda, crear la tabla y luego insertar los siguientes 10 registros:**

| **#** | **Registro** |
| --- | --- |
| **1** | INSERT INTO noticias(id, titulo, visitas, fecha) VALUES (1, 'La vida del futbol', 100, '2019-01-01'); |
| **2** | INSERT INTO noticias(id, titulo, visitas, fecha) VALUES (2, 'Los beneficios del té', 50, '2019-01-10'); |
| **3** | INSERT INTO noticias(id, titulo, visitas, fecha) VALUES (3, 'Firman la paz', 70, '2019-01-02'); |
| **4** | INSERT INTO noticias(id, titulo, visitas, fecha) VALUES (4, 'Nuevo libro', 30, '2019-02-03'); |
| **5** | INSERT INTO noticias(id, titulo, visitas, fecha) VALUES (5, 'Los acuerdos', 80, '2019-02-01'); |
| **6** | INSERT INTO noticias(id, titulo, visitas, fecha) VALUES (6, 'Viva la paz', 90, '2019-02-01'); |
| **7** | INSERT INTO noticias(id, titulo, visitas, fecha) VALUES (7, 'Reencuentro', 100, '2019-02-01'); |
| **8** | INSERT INTO noticias(id, titulo, visitas, fecha) VALUES (8, 'No más', 10, '2019-02-01'); |
| **9** | INSERT INTO noticias(id, titulo, visitas, fecha) VALUES (9, 'Gran concierto', 10, '2019-02-01'); |
| **10** | INSERT INTO noticias(id, titulo, visitas, fecha) VALUES (10, 'Super receta', 11, '2019-01-01'); |

Tu misión es proveer los queries y subqueries de MySQL que permiten responder a las siguientes interrogantes del grupo directivo:

1. **¿Cuál es el promedio de las visitas?**
2. **¿Cuáles noticias están por debajo del promedio de las visitas que están en el punto (1)?**
3. **¿Cuál es el número mínimo de visitas de las noticias que no están en (2)?**

**Puntos de verificación**

Para confirmar que tu experiencia ha sido llevada a cabo con éxito, revisa los siguientes puntos:

1. Verifica la tabla: puedes usar la consulta SHOW TABLES para ver la tabla noticias usando el terminal
2. Promedio de visitas: Debe ser 55.1
3. Noticias por debajo: Deben ser:
   * Los beneficios del té
   * Nuevo libro
   * No más
   * Gran concierto
   * Súper receta
4. Número mínimo de visitas: Debe ser “firma la paz” con 70 visitas.

En este taller has tenido la oportunidad de incrementar tus habilidades en la realización de queries y subqueries en MySQL. Ahora trata de realizar tus propias consultas.

**Solución:**

CREATE TABLE `revista\_digital`.`noticias` (

`id` INT NOT NULL AUTO\_INCREMENT,

`titulo` VARCHAR(255) NOT NULL,

`visitas` INT NOT NULL,

`fecha` DATE NOT NULL,

PRIMARY KEY (`id`));

SELECT AVG(visitas) FROM noticias;

SELECT titulo FROM noticias WHERE visitas < (SELECT AVG(visitas) FROM noticias);

SELECT titulo, MIN(visitas) minimo\_numero\_visitas FROM noticias

WHERE id NOT IN (

SELECT id FROM noticias WHERE visitas < (SELECT AVG(visitas) FROM noticias));

* 1. Cambiando campos en tablas

ALTER TABLE

* 1. Taller 2

**Aprende haciendo**

**Taller: Cambiando la estructura de un producto**

¡Bienvenido!   
  
Supongamos que estás trabajando para una empresa de software que quiere vender partes de autos. Ellos han decidido que necesitan una tabla para almacenar la información de sus productos. Debes crear dicha tabla productos con los siguientes campos:

1. id: INT y clave primaria
2. nombre: VARCHAR(255)

Luego de eso, el equipo ha decidido que también debes modificar semanalmente la tabla según las necesidades del mercado:

| **Semana** | **Modificación** |
| --- | --- |
| **1** | Hacer un SQL que agregue el campo descripción a la tabla |
| **2** | Hacer un SQL que agregue un campo color |
| **3** | Hacer un SQL que modifique el campo color y lo renombre como características |

**Puntos de verificación**

Para confirmar que tu experiencia ha sido llevada a cabo con éxito, revisa los siguientes puntos:

1. Verifica la tabla: puedes usar la consulta SHOW TABLES para ver la tabla personajes, usando el terminal
2. Luego de Semana 1: Verifica si se agregó la descripción con la consulta DESCRIBE productos
3. Luego de Semana 2: Verifica si se agregó el color con la consulta DESCRIBE productos
4. Luego de Semana 3: Verifica el cambio de nombre con la consulta DESCRIBE productos

Recuerda hacer todas las instrucciones de manera separada.

**Solución:**

CREATE TABLE `taller\_vehiculos`.`new\_table` (

`id` INT NOT NULL AUTO\_INCREMENT,

`nombre` VARCHAR(255) NOT NULL,

PRIMARY KEY (`id`));

ALTER TABLE `taller\_vehiculos`.`new\_table`

ADD COLUMN `descripcion` VARCHAR(255) NOT NULL AFTER `nombre`;

ALTER TABLE `taller\_vehiculos`.`new\_table`

ADD COLUMN `color` VARCHAR(255) NOT NULL AFTER `descripcion`;

ALTER TABLE `taller\_vehiculos`.`new\_table`

CHANGE COLUMN `color` `caracteristicas` VARCHAR(255) NOT NULL ;

* 1. ¿Sabías qué? – 1

**Cápsula de conocimiento**

**Soporte de Otros Idiomas en MySQL: Español / Portugués**

MySQL tiene soporte para tu idioma ¿Qué significa esto?



**Usar tus caracteres**

Definiendo charset como UTF8 puedes escribir caracteres especiales como letras con tilde, etc.



**Orden**

Cuando en tu base de datos deseas ordenar campos del tipo string, debes definir el parámetro “collation”, el cual se usa en conjunto con el charset. MySQL de forma predeterminada ordena las palabras de acuerdo a su forma en el inglés, lo que implica que aquellas palabras que tengan tilde u otros caracteres especiales se colocan al final. Por ejemplo, si tenemos tres palabras: “dual”, “día”, “diario”, el sistema ordena estas palabras de la siguiente forma: “diario”, “dual”, “día”, ya que la tilde se toma como un caracter que va después de la “u”. La razón es que se está usando un “collation”, definido como “ISO88591”, con su valor predeterminado en Windows, que es el inglés. Esto no es deseable cuando estamos trabajando en otros idiomas, como el español o el portugués, porque nuestras palabras no quedarían ordenadas en la forma apropiada. Para estos casos debemos definir un collation “UTF8”. En este caso, las palabra de nuestro ejemplo anterior quedarían ordenadas de la siguiente manera: “día”, “diario”, “dual”, lo cual es lo correcto.



**Comparación con tildes**

Supongamos que estás haciendo una búsqueda y estás buscando la palabra “árbol"; si comparas en la sentencia WHERE con la palabra “arbol”, todas las ocurrencias que tenga “árbol” también vendrán como resultado si defines el collation y charset adecuado. Es decir MySQL permite realiza la búsqueda indistintamente si tiene la tilde o no.



**Mensajes de error en tu idioma**

Es posible configurar el servidor para que devuelva los mensajes de error en tu idioma, aunque esto no es lo recomendado ya que la documentación y comentarios en foros están mayormente en inglés.

Esperamos que esta información te sea útil y recuerda configurar correctamente tu idioma.

* 1. ¿Sabías qué? – 2

**Cápsula de conocimiento**

**Resumen de Comandos**

En esta lectura encontrarás un resumen de los comandos usados para realizar subqueries y los usados para modificar tablas en nuestra base de datos.

* **Subqueries:** Los subqueries pueden ser usados en distintas partes de una consulta y deben estar entre paréntesis (). Usualmente se usan en los campos a seleccionar y las condiciones.
* **ALTER TABLE:** Esta instrucción va seguida de la operación. Pueden colocarse una seguida de la otra.
* **Agregando campo:** En las opciones se puede especificar el valor predeterminado, el orden, si permite null, entre otros.

ALTER TABLE [nombre\_tabla] ADD COLUMN

[nombre\_campo] [tipo\_campo] [opciones\_campo]

* **Modificando campo:**

ALTER TABLE [nombre\_tabla] CHANGE COLUMN

[nombre\_campo\_original] [nombre\_campo] [tipo\_campo] [opciones\_campo]

* **Eliminando campo:**

ALTER TABLE [nombre\_tabla] DROP COLUMN [nombre\_campo]

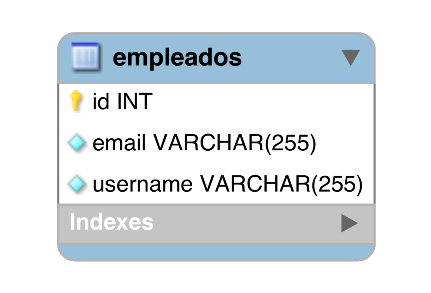
Recuerda que la mejor forma de familiarizarte con los comandos de MySQL es practicar los mismos a través de su aplicación.

* 1. Ejercicio Práctico

**Desafío práctico**

**Evolucionando la Estructura de los Empleados**

En esta oportunidad queremos que crees y modifiques los datos de un cliente y veas su evolución como estructura de datos, a partir de la siguiente tabla:



**Debes escribir instrucciones para cada uno de los siguientes pasos:**

1. Crear la tabla empleados.
2. Crear la tabla géneros con los campos id, descripción.
3. Agregar un campo genero\_id a la tabla empleados.
4. Definir la clave foránea genero\_id desde la tabla empleados a géneros.
5. Agregar un índice al campo genero\_id.
6. Renombrar el campo email por correo\_electrónico.
7. Eliminar el campo username.

**¡Manos a la obra!**

**Solución:**

CREATE TABLE `informacion`.`empleados ` (

`id` INT NOT NULL AUTO\_INCREMENT,

`email` VARCHAR(255) NOT NULL,

`username` VARCHAR(255) NOT NULL,

PRIMARY KEY (`id`));

CREATE TABLE `informacion`.`generos` (

`id` INT NOT NULL AUTO\_INCREMENT,

`descripcion` VARCHAR(255) NOT NULL,

PRIMARY KEY (`id`));

ALTER TABLE `informacion`.`empleados`

ADD COLUMN `genero\_id` INT NOT NULL AFTER `username`;

ALTER TABLE `informacion`.`empleados`

ADD INDEX `fk\_genero\_id\_idx` (`genero\_id` ASC);

;

ALTER TABLE `informacion`.`empleados`

ADD CONSTRAINT `fk\_genero\_id`

FOREIGN KEY (`genero\_id`)

REFERENCES `informacion`.`generos` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION;

ALTER TABLE `informacion`.`empleados`

CHANGE COLUMN `email` `correo\_electronico` VARCHAR(255) CHARACTER SET 'latin1' NOT NULL ;

ALTER TABLE `informacion`.`empleados`

DROP COLUMN `username`;

6. UNIDAD 2: Prueba

|  |  |
| --- | --- |
| **Comenzado en** | Wednesday, 20 de January de 2021, 13:55 |
| **Estado** | Terminados |
| **Finalizado en** | Wednesday, 20 de January de 2021, 14:09 |
| **Tiempo empleado** | 13 mins 53 segundos |
| **Puntos** | 9/10 |
| **Calificación** | **90** de un total de 100 |

Top of Form

### Pregunta 1

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### Texto de la pregunta

Supongamos que tenemos una base de datos que incluye una tabla llamada atletas, que tiene los detalles relacionados a los atletas elites de diversas disciplinas deportivas del país. En particular, esta tabla tiene un campo llamado peso, que indica el peso del atleta. Complete la siguiente definición de un Stored Procedure para que obtenga las estadísticas de peso de los atletas élites de este país, es decir, el promedio de pesos de los atletas, que debe ser retornado en la variables promedio.

CREATE DEFINER = ‘root’ @ ‘%’ PROCEDURE promedio\_peso\_atletas’ (IN limite FLOAT, \_\_\_ promedio FLOAT)

BEGIN

SELECT AVG (peso) INTO \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ FROM atletas \_\_\_\_\_\_\_\_\_\_\_\_\_ peso > limite;

END

Seleccione una:

a. OUT; promedio; WHERE

b. IN; promedio; WHERE

c. OUT; atletas; WHERE

d. OUT; promedio; FROM

#### Retroalimentación

Muy bien, con estos sentencias y variables has completado el Store Procedure correctamente

La respuesta correcta es: OUT; promedio; WHERE

### Pregunta 2

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### Texto de la pregunta

¿Qué retornara la siguiente sentencia MySQL al ejecutarla?

SELECT nombre, apellido FROM estudiantes

WHERE promedio\_general > (SELECT AVG (calificacion\_asignatura) FROM calificaciones WHERE fecha\_aprobada BETWEEN ‘2013-01-1’ AND ‘2018-12-31’)

Seleccione una:

a. El nombre y el apellido de todos los estudiantes cuyo promedio general de notas es mayor al promedio general de notas de todos los estudiantes que aprobaron asignaturas entre el 1 de enero de 2018 y el 31 de diciembre de 2018

b. El promedio general de notas de todos los estudiantes que sea  mayor al promedio general de notas de todos los estudiantes que aprobaron asignaturas entre el 1 de enero de 2018 y el 31 de diciembre de 2018.

c. Todos los detalles de los estudiantes cuyo promedio general de notas es mayor al promedio general de notas de todos los estudiantes que aprobaron asignaturas entre el 1 de enero de 2018 y el 31 de diciembre de 2018.

d. Promedio general de notas de todos los estudiantes que aprobaron asignaturas entre el 1 de enero de 2018 y el 31 de diciembre de 2018.

#### Retroalimentación

**La sentencia se interpreta como sigue selecciona el nombre y apellido de los estudiantes (tabla estudiantes) cuyo promedio general sea mayor que el promedio de la calificaciones de las asignaturas,  que se encuentra la tabla calificaciones entre el periodo especificado por el Where y el Between.**

La respuesta correcta es: El nombre y el apellido de todos los estudiantes cuyo promedio general de notas es mayor al promedio general de notas de todos los estudiantes que aprobaron asignaturas entre el 1 de enero de 2018 y el 31 de diciembre de 2018

### Pregunta 3

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### Texto de la pregunta

¿Cuál de las siguientes modificaciones de una tabla llamada canciones se pueden realizar de forma conjunta usando una sola sentencia ALTER TABLE con sus correspondientes opciones? Señale todas las que apliquen.

Seleccione una o más de una:

a. Agregar un campo a la tabla

b. Cambiar el nombre de un campo de la tabla

c. Cambiar el orden de un campo en la tabla

d. Unir la tabla canciones con la tabla cantantes

#### Retroalimentación

**Las  acciones de agregar un campo a la tabla y cambiar el nombre de un campo de la tabla se pueden realizar de forma simultánea usando la sentencia ALTER TABLE.**

Las respuestas correctas son: Agregar un campo a la tabla, Cambiar el nombre de un campo de la tabla, Cambiar el orden de un campo en la tabla

### Pregunta 4

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### Texto de la pregunta

Un Store Procedure se puede crear en \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Seleccione una:

a. Una base de datos definida por el usuario.

b. Una tabla definida por el usuario.

c. Una condición definida por el usuario (tal como, después de borrar un registro)

d. En un campo de una tabla definida por el usuario.

#### Retroalimentación

**Los Stored Procedures se crean a nivel de  base de datos.**

La respuesta correcta es: Una base de datos definida por el usuario.

### Pregunta 5

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### Texto de la pregunta

Haga coincidir los nombres de las funciones en la columna de la izquierda con su respectiva  definición en la columna de la derecha.

|  |  |
| --- | --- |
| LIKE | Answer 1 |

Bottom of Form

|  |
| --- |
|  |
| IF | Answer 2 |

|  |
| --- |
|  |
| CONCAT | Answer 3 |

|  |
| --- |
|  |
| MD5 | Answer 4 |

|  |
| --- |
|  |

#### Retroalimentación

The correct answer is: LIKE → comparación, IF → condición, CONCAT → Unión de dos o más strings, MD5 → Función criptográfica

### Pregunta 6

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### Texto de la pregunta

Mary desea obtener las ganancias obtenidas por mes entre el  1/1/2018 al 31/12/2018. Hay una tabla ventas que incluye el monto de las ventas y el monto de los gastos por mes. Ayude a Mary con estos cálculos indicando cual de las siguientes sentencias en MySQL es la correcta.

Seleccione una:

a. SELECT mes, (monto\_ventas - monto\_gastos) FROM ventas WHERE fecha\_venta BETWEEN ‘2018-01-01’ AND ‘2018-12-31’

b. SELECT mes, (monto\_ventas - monto\_gastos) WHERE fecha\_venta BETWEEN ‘2018-01-10’ AND ‘2018-12-31’

c. SELECT mes, monto\_ventas,monto\_gastos FROM ventas WHERE fecha\_venta BETWEEN ‘2018-01-01’ AND ‘2018-12-31’

d. SELECT mes, (monto\_ventas - monto\_gastos) FROM ventas WHERE fecha\_venta

#### Retroalimentación

**Para que  Mary pueda obtener sus ventas por mes debemos usar un Select seguido por el campo mes, para que aparezcan las ventas por mes, y la operación matemática que estima la ganancias; seguidamente debemos indicar la tabla de donde estamos haciendo los cálculos usando el FROM; finalmente indicamos el periodo de tiempo usando la sentencia Where con el campo fecha de venta y Between para señalar dicho periodo.**

La respuesta correcta es: SELECT mes, (monto\_ventas - monto\_gastos) FROM ventas WHERE fecha\_venta BETWEEN ‘2018-01-01’ AND ‘2018-12-31’

### Pregunta 7

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### Texto de la pregunta

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ son las opciones de consistencia que proporciona MySQL para cuando se borra o actualiza un registro que es usado en otras tablas (Seleccione todas las que apliquen).

Seleccione una o más de una:

a. CASCADE

b. RESTRICT.

c. NO ACTION.

d. UTF8.

#### Retroalimentación

**Las opciones para mantener la consistencia cuando se borran o actualizan registros usados en otras tablas son: CASCADE, RESTRICT y NO ACTION.**

Las respuestas correctas son: CASCADE, RESTRICT., NO ACTION.

### Pregunta 8

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### Texto de la pregunta

Supongamos que tenemos dos tablas: A y B y queremos obtener los nombres de las personas que están en la tabla A y en la tabla B, ¿Cuál de las siguientes sentencias es la correcta?

Seleccione una o más de una:

a. SELECT \* FROM A LEFT JOIN B ON A.nombre = B.nombre

b. SELECT \* FROM A RIGHT JOIN B ON A.nombre = B.nombre

c. SELECT \* FROM A JOIN B ON A.nombre = B.nombre

d. SELECT \* FROM A,B  WHERE A.nombre = B.nombre

#### Retroalimentación

**Cuando deseamos interceptar  dos tablas para obtener los valores que están en ambas Join o el Where**

Las respuestas correctas son: SELECT \* FROM A JOIN B ON A.nombre = B.nombre, SELECT \* FROM A,B  WHERE A.nombre = B.nombre

### Pregunta 9

Incorrecta

Puntúa 0 sobre 1

Señalar con bandera la pregunta

#### Texto de la pregunta

Supongamos que deseamos alterar nuestra base de datos mibasededatos, modificando la estructura de la tabla mitabla  para agregar un nuevo campo llamado nuevocampo, del tipo entero, que se obligatorio (es decir no nulo) y con valor predeterminado mivalorpordefecto y que estará luego del campo micampoanterior. Se desea que complete la siguiente sentencia MySQL que se utiliza para ello: ALTER TABLE \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ADD COLUMN \_\_\_\_\_\_\_\_\_\_  INT \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ DEFAULT \_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_ ‘micampoanterior’.

Seleccione una:

a. ’mitabla’; ‘minuevocampo’; NOT NULL ‘mivalorpordefecto’; AFTER

b. ’SELECT’; ‘minuevocampo’; NOT NULL ‘mivalorpordefecto’; AFTER

c. ’mibasededatos. mitabla’; ‘minuevocampo’; NO NULO ‘mivalorpordefecto’; AFTER

d. ’mibasededatos. mitabla’; ‘minuevocampo’; BEFORE ‘mivalorpordefecto’; NOT NULO

#### Retroalimentación

**Luego de un ALTER TABLE viene el nombre de la tabla que se desea alterar; luego del ADD COLUMN viene el nuevo campo o columna; debemos colocar NOT NULL para que el campo sea único; luego de DEFAULT se debe colocar el valor predeterminado; y luego del AFTER viene el nombre del campo después del cual se desea agregar el nuevo campo..**

La respuesta correcta es: ’mitabla’; ‘minuevocampo’; NOT NULL ‘mivalorpordefecto’; AFTER

### Pregunta 10

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### Texto de la pregunta

Acerca de las operaciones realizadas vía la interfaz gráfica de MySQL Workbench. ¿Cuáles de las siguientes sentencias son verdaderas?

Seleccione una:

a. Las operaciones se traducen en sentencias SQL, que podemos cambiar

b. Ninguna de  las operaciones se traduce en sentencias SQL

c. Las operaciones se traducen en sentencias SQL, que no podemos cambiar

d. MySQL Workbench no permite ejecutar SQL

#### Retroalimentación

**Uno de los atractivos de la herramienta MySQL Workbench es que las operaciones se traducen en sentencias SQL, que podemos customizar de acuerdo a nuestras necesidades**

La respuesta correcta es: Las operaciones se traducen en sentencias SQL, que podemos cambiar