

Ejercicios de Teoría

◆◆◆
Introducción a la Ingeniería del
Software y los Sistemas de Información

Carlos Arévalo
Daniel Ayala
Margarita Cruz
Bedilia Estrada
Inma Hernández
Alfonso Márquez
David Ruiz



Escuela Técnica Superior de
Ingeniería Informática

Índice general

A. Exámenes de años anteriores	1
A.1. Curso 2021-22	1
A.1.1. Base de datos de referencia	1
A.1.2. Modelo 1	4
A.1.3. Modelo 2	13
A.1.4. Modelo 3	22
A.1.5. Modelo 4	31
B. Ejercicios de SQL avanzado	41
B.1. Ejercicios sobre base de datos de pedido	41
B.1.1. Modelo conceptual	41
B.1.2. Base de datos de referencia	41
B.1.3. Enunciado	43
B.1.4. Soluciones	45

Ejercicio A

Exámenes de años anteriores

A.1. Curso 2021-22

A.1.1. Base de datos de referencia

Listing A.1: src/examenes/21-22/createDB.sql

```
-- IISSI-1. Exámenes de laboratorio Curso 2021-22
-- Autores: David Ruiz y Daniel Ayala
-- Septiembre de 2021

-- Base de Datos
CREATE DATABASE IF NOT EXISTS ModeloExamen;
USE ModeloExamen;

-- Tablas
DROP TABLE IF EXISTS CategoriesPhotos;

DROP TABLE IF EXISTS Categories;

DROP TABLE IF EXISTS Photos;

DROP TABLE IF EXISTS Users;

-- Usuarios
CREATE TABLE IF NOT EXISTS Users (
  idUser int(11) NOT NULL AUTO_INCREMENT,
  name varchar(64) NOT NULL,
  email varchar(64) NOT NULL,
  passwd VARCHAR(256) NOT NULL,
  age tinyint(4) NOT NULL DEFAULT 18,
  PRIMARY KEY (idUser),
  UNIQUE (email)
);
```

```
-- Fotos
CREATE TABLE IF NOT EXISTS Photos (
  idPhoto int(11) NOT NULL AUTO_INCREMENT,
  idUser int(11) NOT NULL,
  url varchar(128),
  PRIMARY KEY (idPhoto),
  FOREIGN KEY (idUser) REFERENCES Users(idUser)
);

CREATE TABLE IF NOT EXISTS Categories (
  idCategory int(11) NOT NULL AUTO_INCREMENT,
  description varchar(64) NOT NULL,
  PRIMARY KEY (idCategory)
);

CREATE TABLE IF NOT EXISTS CategoriesPhotos(
  idCategoryPhoto int(11) NOT NULL AUTO_INCREMENT,
  idPhoto INT(11) NOT NULL,
  idCategory INT(11) NOT NULL,
  PRIMARY KEY (idCategoryPhoto),
  FOREIGN KEY (idPhoto) REFERENCES Photos(idPhoto) ON DELETE CASCADE,
  FOREIGN KEY (idCategory) REFERENCES Categories(idCategory) ON DELETE CASCADE,
  UNIQUE (idPhoto,idCategory)
);

-- Triggers

DELIMITER //
CREATE OR REPLACE TRIGGER tMaximoFotosUsuario
  BEFORE INSERT ON Photos
  FOR EACH ROW
  BEGIN
    DECLARE fotosActuales INT;
    SET fotosActuales = (SELECT COUNT(*) FROM Photos P WHERE P.idUser = new.idUser);
    IF (fotosActuales >= 5) THEN
      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Un usuario no puede tener mas de 5 fotos.';
    END IF;
  END//
DELIMITER ;

DELETE FROM CategoriesPhotos;
DELETE FROM Photos;
DELETE FROM Users;
DELETE FROM Categories;

INSERT INTO Categories (idCategory, description) VALUES
  (1, 'Animales'),
  (2, 'Paisajes'),
  (3, 'Personas'),
  (4, 'Comidas'),
  (5, 'Ciudades'),
  (6, 'Vacaciones'),
  (7, 'Playa');

INSERT INTO Users (idUser, name, email, passwd, age) VALUES
```

```
(1, 'David_Ruiz', 'druiz@us.es', 'pbkdf2:sha256:150000
$rFgsCpnI$884d47b3e5848d88bbf43eb22e1152af7f885e10f6f3f2ec31fd4fe184f20be3', 45),
(2, 'Daniel_Ayala', 'dayala1@us.es', 'pbkdf2:sha256:150000
$F0d4gXgB$e8d2e6ec3e15dae111cc275afffebcd7ecd4f34faad0d60abefca63ba4c2204', 28),
(3, 'Carlos_Arévalo', 'carevalo@us.es', 'pbkdf2:sha256:150000
$HtZ8oG60$3ca0720ca83d0ec35e4ec1e579d1b0fec94c2857d64dd32f08dc527ba97eb383', 55),
(4, 'Alfonso_Márquez', 'amarquez@us.es', 'pbkdf2:sha256:150000
$R1c5MR7y$e2dc906159bec8c79aa0b5b3ac84236b3b2513cbc70741be87592a05999e545c', 35);
```

```
INSERT INTO Photos (idPhoto, idUser, url) VALUES
```

```
(1, 1, 'https://picsum.photos/200/300?random=1'),
(2, 1, 'https://picsum.photos/200/300?random=2'),
(3, 1, 'https://picsum.photos/200/300?random=3'),
(4, 2, 'https://picsum.photos/200/300?random=4'),
(5, 2, 'https://picsum.photos/200/300?random=5'),
(6, 2, 'https://picsum.photos/200/300?random=6'),
(7, 3, 'https://picsum.photos/200/300?random=7'),
(8, 3, 'https://picsum.photos/200/300?random=8'),
(9, 4, 'https://picsum.photos/200/300?random=9');
```

```
INSERT INTO CategoriesPhotos (idCategoryPhoto, idPhoto, idCategory) VALUES
```

```
(1, 1, 1),
(2, 1, 7),
(3, 1, 6),
(4, 4, 5),
(5, 4, 3),
(6, 2, 2),
(7, 2, 3),
(8, 2, 5),
(9, 3, 3),
(10, 3, 4),
(11, 3, 5),
(12, 4, 7),
(13, 5, 5),
(14, 6, 4),
(15, 7, 1),
(16, 7, 2),
(17, 7, 3),
(18, 7, 5),
(19, 7, 6),
(20, 8, 1);
```

A.1.2. Modelo 1

IISI-1 Prueba de Laboratorio.	Curso 2021-22 Enero 2022
Apellidos, Nombre:	Grupo:

Calificación Final: _____

- Creen un nuevo proyecto Silence llamado UVUS y modifique settings.py con los valores adecuados para DB_CONN, SQL_SCRIPTS y USER_AUTH_DATA. Para la autenticación se mapea sobre la tabla de Usuarios siendo el identificador el atributo "email" y el password el atributo "passwd" de la tabla de Usuarios.
- Asegúrese de que todo funciona ejecutando "silence createdb", "silence createapi", "silence createtests" y finalmente "silence run".

Puerto de la base de datos (DB_CONN → port): **3306**

Puerto de despliegue de la API (HTTP_PORT):

- Laboratorios módulo F: **8081**
- En otro caso: **8080**

IISII-1 Prueba de Laboratorio.	Curso 2021-22 Enero 2022
Apellidos, Nombre:	Grupo:

Pregunta 1 (1punto)

Escriba y realice una petición HTTP para registrar un nuevo usuario en el sistema con los siguientes datos: nombre="Alumno Haciendo Examen", email="alumno@us.es", edad="20", passwd="alumno".

Escriba y realice una petición HTTP para obtener el nuevo listado de usuarios en el que deberá aparecer el usuario que acaba de dar de alta.

[Incluya capturas mostrando las peticiones realizadas y el resultado obtenido]

Calificación:

IISSI-1 Prueba de Laboratorio.	Curso 2021-22 Enero 2022
Apellidos, Nombre:	Grupo:

Pregunta 2. (1punto)

Escriba y realice una petición HTTP que cree una nueva Categoría con el valor "Examen de iissi". Tenga en cuenta para crear dicha Categoría se requiere estar autenticado.

[Incluya capturas mostrando las peticiones realizadas y el resultado obtenido]

Calificación: _____

IISSI-1 Prueba de Laboratorio.	Curso 2021-22 Enero 2022
Apellidos, Nombre:	Grupo:

Pregunta 3 (2 puntos).

Añada el requisito de información **Concurso**. Un concurso es una competición de fotos en la que una es escogida como ganadora. Sus atributos son: la foto ganadora, el nombre del concurso, la cuantía del premio, y la fecha en la que tiene lugar. Hay que tener en cuenta las siguientes restricciones:

- Una misma foto no puede ser ganadora de varios concursos un mismo día.
- La cuantía del premio debe ser mayor o igual a 10€, y menor o igual que 500€.
- La fecha del concurso debe ser anterior al año 2050.
- Todos los atributos son obligatorios.

[Incluya el código realizado](#)

Calificación:

Pregunta 4 (2 puntos).

Cree y ejecute un procedimiento almacenado llamado pAddConest(...) para crear un concurso a partir de los parámetros de entrada. Realice llamadas que creen los siguientes concursos:

- Concurso titulado "Concurso 1" ganado por la foto con ID=2, con premio de 20€, en el 01/01/2020. → OK
- Concurso titulado "Concurso 2" ganado por la foto con ID=2, con premio de 100€, en el 15/04/2019. → OK
- Concurso titulado "Concurso 3" ganado por la foto con ID=1, con premio de 70€, en el 10/10/2021. → OK
- Concurso titulado "Concurso 4" ganado por la foto con ID=1, con premio de 70€, en el 01/01/2100. → ERROR (el año del proyecto debe ser anterior al año 2050)
- Concurso titulado "Concurso 5" ganado por la foto con ID=15, con premio de 70€, del año 2021. → ERROR (por clave ajena no existente)
- Concurso titulado "Concurso 6" ganado por la foto con ID=2, con premio de 70€, en el 01/01/2020. → ERROR (una foto no puede ganar más de un concurso el mismo día)

[Incluya el código realizado y capturas mostrando el estado de la tabla tras las llamadas](#)

Cree un procedimiento almacenado llamado pDelContests(p) que elimina los concursos ganados por la Foto con ID=p. Ejecute la llamada pDelContests(1).

[Incluya el código realizado y capturas mostrando el estado de la tabla tras la llamada](#)

Cree un procedimiento almacenado llamado pUpdContests(p, a) que actualiza el premio de los proyectos ganados por la Foto p con el valor a.

Ejecute la llamada pUpdContests(2, 300).

[Incluya el código realizado y capturas mostrando el estado de la tabla tras la llamada](#)

IISI-1 Prueba de Laboratorio.	Curso 2021-22 Enero 2022
Apellidos, Nombre:	Grupo:

Calificación: _____

IISI-1 Prueba de Laboratorio.	Curso 2021-22 Enero 2022
Apellidos, Nombre:	Grupo:

Pregunta 5 (1 punto).

Cree un disparador llamado tCorrectPrize que, al actualizarse un concurso, si el premio fuera mayor a 500€, lo establezca como 500€ en su lugar. Ejecute la llamada a pUpdContests(2, 700) para comprobar que funciona el disparador. Tenga en cuenta que este procedimiento debe hacer saltar al disparador y que el valor final actualizado sea correcto.

[Incluya el código realizado y capturas mostrando el estado de la tabla tras la llamada]

Calificación: _____

IISSI-1 Prueba de Laboratorio.	Curso 2021-22 Enero 2022
Apellidos, Nombre:	Grupo:

Pregunta 6 (1 punto).

Cree un procedimiento **pAddTwoContests(...)** que, dentro de una transacción, inserte dos concursos a partir de los datos suministrados de los dos concursos.

Realice dos llamadas: una que inserte dos concursos correctamente, y una en la que el segundo rompa alguna restricción y aborte la transacción.

Incluya el código realizado y capturas mostrando el estado de la tabla tras las llamadas>>

Calificación:

IISSI-1 Prueba de Laboratorio.	Curso 2021-22 Enero 2022
Apellidos, Nombre:	Grupo:

Pregunta 7 (2 puntos).

Realice las siguientes consultas:

1. Devolver un listado de las fotos incluyendo la descripción de sus categorías (cada foto aparece varias veces si tiene varias cate. El listado debe aparecer ordenado por el nombre de la categoría.
2. Devolver un listado con el número de categorías que tiene cada foto.
3. Devolver un listado con la cantidad de fotos de cada categoría con una descripción que empiece por 'P'.

idPhoto	idUser	url	description
1	1	https://picsum.photos/200/30...	Animales
7	3	https://picsum.photos/200/30...	Animales
8	3	https://picsum.photos/200/30...	Animales
4	2	https://picsum.photos/200/30...	Ciudades
2	1	https://picsum.photos/200/30...	Ciudades
3	1	https://picsum.photos/200/30...	Ciudades
5	2	https://picsum.photos/200/30...	Ciudades
7	3	https://picsum.photos/200/30...	Ciudades
3	1	https://picsum.photos/200/30...	Comidas
6	2	https://picsum.photos/200/30...	Comidas
2	1	https://picsum.photos/200/30...	Paisajes
7	3	https://picsum.photos/200/30...	Paisajes
4	2	https://picsum.photos/200/30...	Personas
2	1	https://picsum.photos/200/30...	Personas
3	1	https://picsum.photos/200/30...	Personas
7	3	https://picsum.photos/200/30...	Personas
1	1	https://picsum.photos/200/30...	Playa
4	2	https://picsum.photos/200/30...	Playa
1	1	https://picsum.photos/200/30...	Vacaciones
7	3	https://picsum.photos/200/30...	Vacaciones

idPhoto	numberCat
1	3
2	3
3	3
4	3
5	1
6	1
7	5
8	1

idCategory	description	numberPhotos
2	Paisajes	2
3	Personas	4
7	Playa	2

Incluya el código realizado y capturas mostrando el resultado de cada consulta

Calificación: _____

Listing A.2: src/examenes/21-22/soluciones_queries_1.sql

```
-- 1. Devolver un listado de las fotos incluyendo la descripción de sus categorías (
    cada foto aparece varias veces si tiene varias cate. El listado debe aparecer
    ordenado por el nombre de la categoría.

SELECT P.*, C.description
FROM Photos P JOIN CategoriesPhotos CP ON (P.idPhoto = CP.idPhoto)
JOIN Categories C ON (CP.idCategory = C.idCategory)
ORDER BY description;

-- 2. Devolver un listado con el número de categorías que tiene cada foto.

SELECT P.idPhoto, COUNT(*) AS numberCat
FROM Photos P JOIN CategoriesPhotos CP ON (P.idPhoto = CP.idPhoto)
JOIN Categories C ON (CP.idCategory = C.idCategory)
GROUP BY CP.idPhoto;

-- 3. Devolver un listado con la cantidad de fotos de cada categoría con una descripci
    ón que empiece por 'P'.

SELECT C.idCategory, C.description, COUNT(*) AS numberPhotos
FROM Photos P JOIN CategoriesPhotos CP ON (P.idPhoto = CP.idPhoto)
JOIN Categories C ON (CP.idCategory = C.idCategory)
GROUP BY C.idCategory
HAVING C.description LIKE 'P%';
```


A.1.3. Modelo 2

IISSI-1 Prueba de Laboratorio.	Curso 2021-22 Enero 2022
Apellidos, Nombre:	Grupo:

Calificación Final: _____

- Creen un nuevo proyecto Silence llamado UVUS y modifique settings.py con los valores adecuados para DB_CONN, SQL_SCRIPTS y USER_AUTH_DATA. Para la autenticación se mapea sobre la tabla de Usuarios siendo el identificador el atributo "email" y el password el atributo "passwd" de la tabla de Usuarios.
- Asegúrese de que todo funciona ejecutando "silence createdb", "silence createapi", "silence createtests" y finalmente "silence run".

Puerto de la base de datos (DB_CONN → port): **3306**

Puerto de despliegue de la API (HTTP_PORT):

- Laboratorios módulo F: **8081**
- En otro caso: **8080**

IISSI-1 Prueba de Laboratorio.	Curso 2021-22 Enero 2022
Apellidos, Nombre:	Grupo:

Pregunta 1 (1punto)

Escriba y realice una petición HTTP para registrar un nuevo usuario en el sistema con los siguientes datos: nombre="Alumno Haciendo Control", email="alumno@us.es", edad="21", passwd="alumno".

Escriba y realice una petición HTTP para obtener el nuevo listado de usuarios en el que deberá aparecer el usuario que acaba de dar de alta.

Incluya capturas mostrando las peticiones realizadas y el resultado obtenido

Calificación:

IISSI-1 Prueba de Laboratorio.	Curso 2021-22 Enero 2022
Apellidos, Nombre:	Grupo:

Pregunta 2. (1punto)

Escriba y realice una petición HTTP que actualice la Categoría con ID=1 para que su descripción sea "Examen IISSI". Tenga en cuenta para actualizar dicha Categoría se requiere estar autenticado.

Incluya capturas mostrando las peticiones realizadas y el resultado obtenido

Calificación:

IISSI-1 Prueba de Laboratorio.	Curso 2021-22 Enero 2022
Apellidos, Nombre:	Grupo:

Pregunta 3 (2 puntos).

Añada el requisito de información **Producto**. Un producto es un objeto vendido por un usuario. Sus atributos son: el usuario que lo vende, el nombre del producto, el precio, y la cantidad de unidades disponibles. Hay que tener en cuenta las siguientes restricciones:

- Un mismo usuario no puede vender varios productos con el mismo nombre.
- El precio debe ser mayor o igual a 1€, y menor o igual que 600€.
- La cantidad de unidades disponibles debe ser mayor o igual a 0.
- Todos los atributos son obligatorios.

Incluya el código realizado

Calificación:

IISSI-1 Prueba de Laboratorio.	Curso 2021-22 Enero 2022
Apellidos, Nombre:	Grupo:

Pregunta 4 (2 puntos).

Cree y ejecute un procedimiento almacenado llamado pAddProduct(...) para crear un producto a partir de los parámetros de entrada. Realice llamadas que creen los siguientes productos:

- Producto titulado "Producto 1" vendido por el usuario con ID=2, con precio de 20€, con 10 unidades. → OK
- Producto titulado "Producto 2" vendido por el usuario con ID=2, con precio de 100€, con 1 unidad. → OK
- Producto titulado "Producto 3" vendido por el usuario con ID=1, con precio de 70€, con 5 unidades. → OK
- Producto titulado "Producto 4" vendido por el usuario con ID=1, con precio de 700€, con 5 unidades. → ERROR (el precio debe estar entre 1 y 600 euros)
- Producto titulado "Producto 5" vendido por el usuario con ID=15, con precio de 300€, con 5 unidades. → ERROR (por clave ajena no existente)
- Producto titulado "Producto 1" vendido por el usuario con ID=2, con precio de 300€, con 5 unidades. → ERROR (un mismo usuario no puede vender varios productos con el mismo nombre)

Incluya el código realizado y capturas mostrando el estado de la tabla tras las llamadas

Cree un procedimiento almacenado llamado pDelProducts(u) que elimina los Productos vendidos por el Usuario con ID=u. Ejecute la llamada pDelProducts(1).

Incluya el código realizado y capturas mostrando el estado de la tabla tras la llamada

Cree un procedimiento almacenado llamado pUpdProducts(u, a) que actualiza las unidades de los productos vendidos por el Usuario u con el valor a.

Ejecute la llamada pUpdProducts(2,1).

Incluya el código realizado y capturas mostrando el estado de la tabla tras la llamada

Calificación: _____

IISSI-1 Prueba de Laboratorio.	Curso 2021-22 Enero 2022
Apellidos, Nombre:	Grupo:

Pregunta 5 (1 punto).

Cree un disparador llamado tCorrectUnits que, al actualizarse un Producto, si la cantidad de unidades fuera menor a 0, la establezca como 0 en su lugar. Ejecute la llamada a pUpdProducts(2,-5) para comprobar que funciona el disparador. Tenga en cuenta que este procedimiento debe hacer saltar al disparador y que el valor final actualizado sea correcto.

Incluya el código realizado y capturas mostrando el estado de la tabla tras la llamada

Calificación: _____

IISI-1 Prueba de Laboratorio.	Curso 2021-22 Enero 2022
Apellidos, Nombre:	Grupo:

Pregunta 6 (1 punto).

Cree un procedimiento **pAddTwoProducts(...)** que, dentro de una transacción, inserte dos Productos a partir de los datos suministrados de los dos Productos.

Realice dos llamadas: una que inserte dos Productos correctamente, y una en la que el segundo rompa alguna restricción y aborte la transacción.

Incluya el código realizado y capturas mostrando el estado de la tabla tras las llamadas>>

Calificación:

IISI-1 Prueba de Laboratorio.	Curso 2021-22 Enero 2022
Apellidos, Nombre:	Grupo:

Pregunta 7 (2 puntos).


Reinicie la base de datos.

Realice las siguientes consultas:

1. Devolver los 4 usuarios con mayor edad.
2. Devolver un listado con la edad media de los usuarios de las fotos de cada categoría de fotos.
3. Devolver un listado con las categorías con más de 2 fotos.

 idUser	name	 email	passwd	age
3	Carlos Arévalo	carevalo@us.es	pbkdf2:sha256:150000\$HtZ8o...	55
1	David Ruiz	druiz@us.es	pbkdf2:sha256:150000\$rFgsC...	45
4	Alfonso Márquez	amarquez@us.es	pbkdf2:sha256:150000\$R1c5...	35
2	Daniel Ayala	dayale1@us.es	pbkdf2:sha256:150000\$F0d4g...	28

idCategory	ageAverage
1	51,6667
2	50,0000
3	43,2500
4	36,5000
5	40,2000
6	50,0000
7	36,5000

 idCategory	description
1	Animales
3	Personas
5	Ciudades

Incluya el código realizado y capturas mostrando el resultado de cada consulta

Calificación: _____

Listing A.3: src/examenes/21-22/soluciones_queries_2.sql

```
-- 1. Devolver los 4 usuarios con mayor edad.

SELECT * FROM Users
ORDER BY age DESC
LIMIT 4;

-- 2. Devolver un listado con la edad media de los usuarios de las fotos de cada
    categoría de fotos.

SELECT C.idCategory, AVG(U.AGE) AS ageAverage
FROM Categories C JOIN CategoriesPhotos CP ON (C.idCategory = CP.idCategory)
JOIN Photos P ON (P.idPhoto = CP.idPhoto)
JOIN Users U ON (P.idUser = U.idUser)
GROUP BY C.idCategory;

-- 3. Devolver un listado con las categorías con más de 2 fotos.

SELECT C.* FROM
Categories C JOIN CategoriesPhotos CP ON (C.idCategory = CP.idCategory)
GROUP BY C.idCategory
HAVING COUNT(*) > 2;
```

A.1.4. Modelo 3

IISSE-1 Prueba de Laboratorio.	Curso 2021-22 Enero 2022
Apellidos, Nombre:	Grupo:

Calificación Final: _____

- Creen un nuevo proyecto Silence llamado UVUS y modifique settings.py con los valores adecuados para DB_CONN, SQL_SCRIPTS y USER_AUTH_DATA. Para la autenticación se mapea sobre la tabla de Usuarios siendo el identificador el atributo "email" y el password el atributo "passwd" de la tabla de Usuarios.
- Asegúrese de que todo funciona ejecutando "silence createdb", "silence createapi", "silence createtests" y finalmente "silence run".

Puerto de la base de datos (DB_CONN → port): **3306**

Puerto de despliegue de la API (HTTP_PORT):

- Laboratorios módulo F: **8081**
- En otro caso: **8080**

IISII-1 Prueba de Laboratorio.	Curso 2021-22 Enero 2022
Apellidos, Nombre:	Grupo:

Pregunta 1 (1punto)

Escriba y realice una petición HTTP para registrar un nuevo usuario en el sistema con los siguientes datos: nombre="Alumno Empezando Examen", email="alumno@us.es", edad="19", passwd="alumno".

Escriba y realice una petición HTTP para obtener el nuevo listado de usuarios en el que deberá aparecer el usuario que acaba de dar de alta.

Incluya capturas mostrando las peticiones realizadas y el resultado obtenido

Calificación:

IISSI-1 Prueba de Laboratorio.	Curso 2021-22 Enero 2022
Apellidos, Nombre:	Grupo:

Pregunta 2. (1punto)

Escriba y realice una petición HTTP que borre la Categoría con ID=7. Tenga en cuenta para borrar dicha Categoría se requiere estar autenticado.

Incluya capturas mostrando las peticiones realizadas y el resultado obtenido

Calificación: _____

IISSI-1 Prueba de Laboratorio.	Curso 2021-22 Enero 2022
Apellidos, Nombre:	Grupo:

Pregunta 3 (2 puntos).

Añada el requisito de información **Vídeo**. Un vídeo tiene una foto usada como vista en miniatura. Sus atributos son: la foto usada como vista en miniatura, el título del vídeo, la duración en segundos, y la fecha en la que fue subido. Hay que tener en cuenta las siguientes restricciones:

- La misma foto no puede usarse en varios vídeos subidos el mismo día.
- La duración debe ser superior o igual a 5s e inferior o igual a 1000s.
- El año de la fecha debe ser posterior a 1990.
- Todos los atributos son obligatorios.

Incluya el código realizado

Calificación:

IISI-1 Prueba de Laboratorio.	Curso 2021-22 Enero 2022
Apellidos, Nombre:	Grupo:

Pregunta 4 (2 puntos).

Cree y ejecute un procedimiento almacenado llamado pAddVideo(...) para crear un Vídeo a partir de los parámetros de entrada. Realice llamadas que creen los siguientes Vídeos:

- Vídeo titulado "Vídeo 1" con la foto con ID=2, con duración de 5s, subido el 01/01/2020. → OK
- Vídeo titulado "Vídeo 2" con la foto con ID=2, con duración de 10s, subido el 02/01/2020. → OK
- Vídeo titulado "Vídeo 3" con la foto con ID=1, con duración de 40s, subido el 02/01/2020. → OK
- Vídeo titulado "Vídeo 4" con la foto con ID=1, con duración de 1700s subido el 03/01/2020. → ERROR (la duración debe estar entre 5s y 1000s)
- Vídeo titulado "Vídeo 5" con la foto con ID=15, con duración de 300s, subido el 04/01/2020. → ERROR (por clave ajena no existente)
- Vídeo titulado "Vídeo 6" con la foto con ID=2, con duración de 300s, subido el 01/01/2020. → ERROR (la misma foto no puede usarse en varios vídeos el mismo día)

Incluya el código realizado y capturas mostrando el estado de la tabla tras las llamadas

Cree un procedimiento almacenado llamado pDelVideos(p) que elimina los Vídeos con la Foto con ID=p. Ejecute la llamada pDelVideos(1).

Incluya el código realizado y capturas mostrando el estado de la tabla tras la llamada

Cree un procedimiento almacenado llamado pUpdVideos(p, d) que actualiza la duración de los Vídeos con la Foto p con el valor d.

Ejecute la llamada pUpdVideos(2,15).

Incluya el código realizado y capturas mostrando el estado de la tabla tras la llamada

Calificación: _____

IISI-1 Prueba de Laboratorio.	Curso 2021-22 Enero 2022
Apellidos, Nombre:	Grupo:

Pregunta 5 (1 punto).

Cree un disparador llamado `tCorrectDuration` que, al actualizarse un `Video`, si la duración fuera menor a 5s, la establezca como 5s en su lugar. Ejecute la llamada a `pUpdVideos(2,0)` para comprobar que funciona el disparador. Tenga en cuenta que este procedimiento debe hacer saltar al disparador y que el valor final actualizado sea correcto.

Incluya el código realizado y capturas mostrando el estado de la tabla tras la llamada

Calificación: _____

IISI-1 Prueba de Laboratorio.	Curso 2021-22 Enero 2022
Apellidos, Nombre:	Grupo:

Pregunta 6 (1 punto).

Cree un procedimiento **pAddTwoVideos(...)** que, dentro de una transacción, inserte dos Videos a partir de los datos suministrados de los dos Videos.

Realice dos llamadas: una que inserte dos Videos correctamente, y una en la que el segundo rompa alguna restricción y aborte la transacción.

Incluya el código realizado y capturas mostrando el estado de la tabla tras las llamadas>>

Calificación:

IISI-1 Prueba de Laboratorio.	Curso 2021-22 Enero 2022
Apellidos, Nombre:	Grupo:

Pregunta 7 (2 puntos).

Reinicie la base de datos.

Realice las siguientes consultas:

1. Devolver los usuarios con al menos una foto.
2. Devolver las fotos del usuario con mayor edad.
3. Devolver un listado con los usuarios con una cantidad de fotos que, multiplicada por 10, sea mayor a su edad.

idUser
4
3
2
1

idPhoto	idUser	url
7	3	https://picsum.photos/200/300/...
8	3	https://picsum.photos/200/300/...

idUser	name	email	passwd	age	numPhotos
2	Daniel Ayala	dayala1@us.es	pbkdf2:sha256:150000\$F0d4g...	28	3

Incluya el código realizado y capturas mostrando el resultado de cada consulta

Calificación: _____

Listing A.4: src/examenes/21-22/soluciones_queries_3.sql

```
-- 1. Devolver los usuarios con al menos una foto.

SELECT UNIQUE(U.idUser)
FROM Users U JOIN Photos P ON (U.idUser = P.idUser);

-- 2. Devolver las fotos del usuario con mayor edad.

SELECT U.*
FROM Users U JOIN Photos P ON (U.idUser = P.idUser)
WHERE U.age >= ALL (SELECT age FROM Users);

-- 3. Devolver un listado con los usuarios con una cantidad de fotos que, multiplicada
    por 10, sea mayor a su edad.

SELECT U.*, COUNT(*) as numPhotos
FROM Users U JOIN Photos P ON (U.idUser = P.idUser)
GROUP BY U.idUser
HAVING numPhotos*10 > U.age;
```

A.1.5. Modelo 4

IISSI-1 Prueba de Laboratorio.	Curso 2021-22 Enero 2022
Apellidos, Nombre:	Grupo:

Calificación Final: _____

- Creen un nuevo proyecto Silence llamado UVUS y modifique settings.py con los valores adecuados para DB_CONN, SQL_SCRIPTS y USER_AUTH_DATA. Para la autenticación se mapea sobre la tabla de Usuarios siendo el identificador el atributo "email" y el password el atributo "passwd" de la tabla de Usuarios.
- Asegúrese de que todo funciona ejecutando "silence createdb", "silence createapi", "silence createtests" y finalmente "silence run".

Puerto de la base de datos (DB_CONN → port): **3306**

Puerto de despliegue de la API (HTTP_PORT):

- Laboratorios módulo F: **8081**
- En otro caso: **8080**

IISSI-1 Prueba de Laboratorio.	Curso 2021-22 Enero 2022
Apellidos, Nombre:	Grupo:

Pregunta 1 (1punto)

Escriba y realice una petición HTTP para registrar un nuevo usuario en el sistema con los siguientes datos: nombre="Alumno Haciendo Examen", email="alumno@us.es", edad="19", passwd="alumno".

Escriba y realice una petición HTTP para obtener el nuevo listado de usuarios en el que deberá aparecer el usuario que acaba de dar de alta.

Incluya capturas mostrando las peticiones realizadas y el resultado obtenido

Calificación:

IISSI-1 Prueba de Laboratorio.	Curso 2021-22 Enero 2022
Apellidos, Nombre:	Grupo:

Pregunta 2. (1punto)

Escriba y realice una petición HTTP que cree una nueva Foto con la url "http:examen". Tenga en cuenta que para crear dicha Foto se requiere estar autenticado.

Incluya capturas mostrando las peticiones realizadas y el resultado obtenido

Calificación:

IISSI-1 Prueba de Laboratorio.	Curso 2021-22 Enero 2022
Apellidos, Nombre:	Grupo:

Pregunta 3 (2 puntos).

Añada el requisito de información **Evento**. Un evento es organizado por un usuario para que asista una cantidad máxima de personas. Sus atributos son: el usuario que lo organiza, el título del evento, la capacidad del evento, y la fecha en la que tendrá lugar. Hay que tener en cuenta las siguientes restricciones:

- Un mismo usuario no puede organizar varios eventos el mismo día.
- La capacidad debe ser superior o igual a 2 e inferior o igual a 1000.
- El año de la fecha debe ser posterior al 2000.
- Todos los atributos son obligatorios.

Incluya el código realizado

Calificación: _____

IISSI-1 Prueba de Laboratorio.	Curso 2021-22 Enero 2022
Apellidos, Nombre:	Grupo:

Pregunta 4 (2 puntos).

Cree y ejecute un procedimiento almacenado llamado `pAddEvent(...)` para crear un Evento a partir de los parámetros de entrada. Realice llamadas que creen los siguientes Vídeos:

- Evento titulado "Evento 1" organizado por el usuario con ID=2, con capacidad para 10, que tiene lugar el 01/04/2021. → OK
- Evento titulado "Evento 2" organizado por el usuario con ID=2, con capacidad para 10, que tiene lugar el 05/06/2018. → OK
- Evento titulado "Evento 3" organizado por el usuario con ID=1, con capacidad para 100, que tiene lugar el 04/02/2020. → OK
- Evento titulado "Evento 4" organizado por el usuario con ID=1, con capacidad para 70 que tiene lugar el 05/05/1990. → ERROR (el año de la fecha debe ser posterior al 2000)
- Evento titulado "Evento 5" organizado por el usuario con ID=15, con capacidad para 200, que tiene lugar el 04/01/2020. → ERROR (por clave ajena no existente)
- Evento titulado "Evento 6" organizado por el usuario con ID=2, con capacidad para 200, que tiene lugar el 01/04/2021. → ERROR (un usuario no puede organizar varios eventos para el mismo día)

Incluya el código realizado y capturas mostrando el estado de la tabla tras las llamadas

Cree un procedimiento almacenado llamado `pDelEvents(u)` que elimina los Eventos organizados por el usuario con ID=u.

Ejecute la llamada `pDelEvents(1)`.

Incluya el código realizado y capturas mostrando el estado de la tabla tras la llamada

Cree un procedimiento almacenado llamado `pUpdEvents (u, c)` que actualiza la capacidad de los Eventos organizados por el Usuario u con el valor c.

Ejecute la llamada `pUpdEvents(2,30)`.

Incluya el código realizado y capturas mostrando el estado de la tabla tras la llamada

Calificación: _____

IISSI-1 Prueba de Laboratorio.	Curso 2021-22 Enero 2022
Apellidos, Nombre:	Grupo:

Pregunta 5 (1 punto).

Cree un disparador llamado `tCorrectCapacity` que, al actualizarse un Evento, si la duración fuera mayor a 1000, la establezca como 1000 en su lugar. Ejecute la llamada a `pUpdEvents(2,5000)` para comprobar que funciona el disparador. Tenga en cuenta que este procedimiento debe hacer saltar al disparador y que el valor final actualizado sea correcto.

Incluya el código realizado y capturas mostrando el estado de la tabla tras la llamada

Calificación: _____

IISI-1 Prueba de Laboratorio.	Curso 2021-22 Enero 2022
Apellidos, Nombre:	Grupo:

Pregunta 6 (1 punto).

Cree un procedimiento **pAddTwoEvents(...)** que, dentro de una transacción, inserte dos Eventos a partir de los datos suministrados de los dos Eventos.

Realice dos llamadas: una que inserte dos Eventos correctamente, y una en la que el segundo rompa alguna restricción y aborte la transacción.

Incluya el código realizado y capturas mostrando el estado de la tabla tras las llamadas>>

Calificación:

IISSI-1 Prueba de Laboratorio.	Curso 2021-22 Enero 2022
Apellidos, Nombre:	Grupo:

Pregunta 7 (2 puntos).

Realice las siguientes consultas:

1. Devolver un listado de las fotos junto con la edad del usuario que la subió, ordenado por edad de menor a mayor.
2. Devolver los usuarios mayores de 30 años junto con la cantidad de fotos que tienen.
3. Devolver un listado de las categorías con exactamente dos fotos.

idPhoto	idUser	url	age
6	2	https://picsum.photos/200/30...	28
5	2	https://picsum.photos/200/30...	28
4	2	https://picsum.photos/200/30...	28
9	4	https://picsum.photos/200/30...	35
1	1	https://picsum.photos/200/30...	45
3	1	https://picsum.photos/200/30...	45
2	1	https://picsum.photos/200/30...	45
8	3	https://picsum.photos/200/30...	55
7	3	https://picsum.photos/200/30...	55

idUser	name	email	passwd	age	numPhotos
1	David Ruiz	druiz@us.es	pbkdf2:sha256:150000\$rFgsC...	45	3
3	Carlos Arévalo	carevalo@us.es	pbkdf2:sha256:150000\$HtZ8o...	55	2
4	Alfonso Márquez	amarquez@us.es	pbkdf2:sha256:150000\$R1c5...	35	1

idCategory	description
2	Paisajes
4	Comidas
6	Vacaciones
7	Playa

Incluya el código realizado y capturas mostrando el resultado de cada consulta

Calificación: _____

Listing A.5: src/examenes/21-22/soluciones_queries_4.sql

-- 1. Devolver un listado de las fotos junto con la edad del usuario que la subió, ordenado por edad de menor a mayor.

```
SELECT P.*, age
FROM Photos P JOIN Users U ON (P.idUser = U.idUser)
ORDER BY age;
```

-- 2. Devolver los usuarios mayores de 30 años junto con la cantidad de fotos que tienen.

```
SELECT U.*, COUNT(*) as numPhotos
FROM Photos P JOIN Users U ON (P.idUser = U.idUser)
WHERE U.age > 30
GROUP BY U.idUser;
```

-- 3. Devolver un listado de las categorías con exactamente dos fotos.

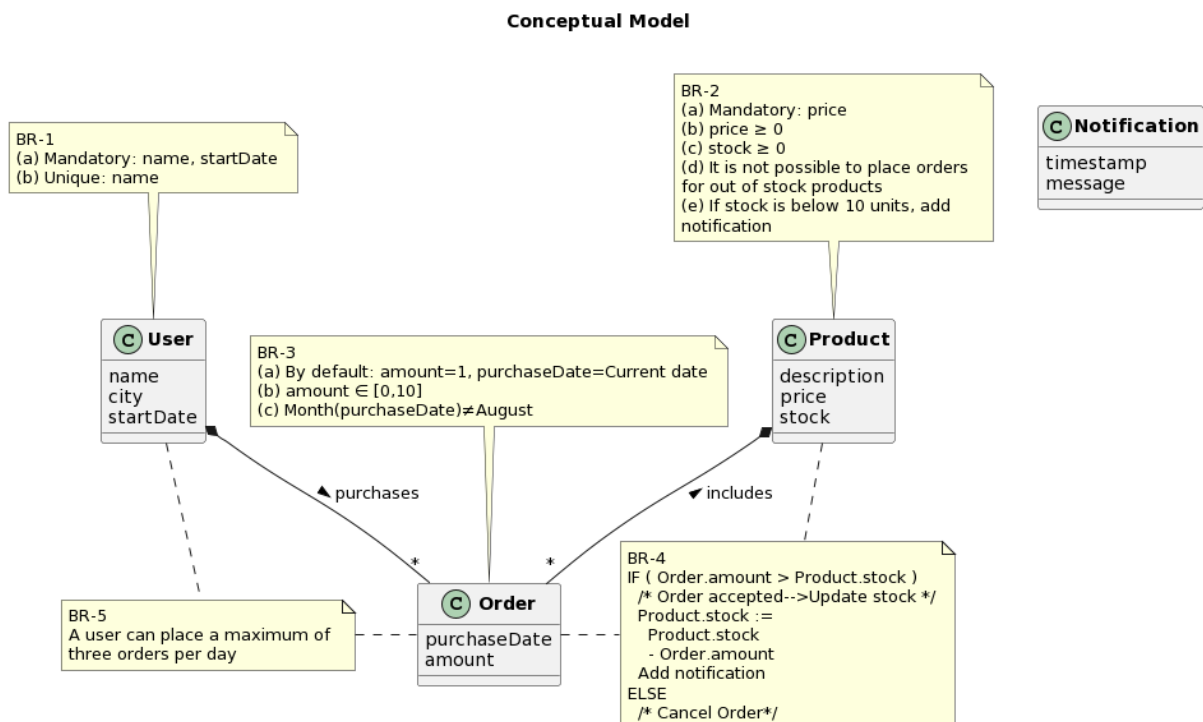
```
SELECT C.*
FROM Categories C JOIN CategoriesPhotos CP ON (C.idCategory = CP.idCategory)
GROUP BY C.idCategory
HAVING COUNT(*) = 2;
```


Ejercicio B

Ejercicios de SQL avanzado

B.1. Ejercicios sobre base de datos de pedido

B.1.1. Modelo conceptual



B.1.2. Base de datos de referencia

Listing B.1: src/BaseDatosPedidos/create.sql

```

/* Support Scripts for T11 exercises
Authors: Inma Hernández, David Ruiz, Daniel Ayala, and Carlos Arévalo
Creation date: November 2019
V1.1: November 2020 (Carlos Arévalo)
v1.2: November 2022 (Inma Hernández)
*/
--
-- Script for table and data creation
--
--
-- Database creation
--

CREATE DATABASE IF NOT EXISTS upon;
USE upon;
-- Table and trigger drop
--
DROP TABLE IF EXISTS orders;
DROP TABLE IF EXISTS products;
DROP TABLE IF EXISTS users;
DROP TABLE IF EXISTS notifications;

-- Table creation
--
/* Relation: Users */
CREATE OR REPLACE TABLE Users(
  userId      INT NOT NULL AUTO_INCREMENT,
  `name`      VARCHAR(63) NOT NULL, /*BR-1a*/
  province    VARCHAR(63),
  startDate   DATE NOT NULL, /*BR-1a*/
  PRIMARY KEY(userId),
  CONSTRAINT BR_1b_nameUnique UNIQUE(NAME) /*BR-1b*/
);

/* Relation: Products */
CREATE OR REPLACE TABLE Products(
  productId   INT NOT NULL AUTO_INCREMENT,
  description  VARCHAR(128),
  price       DECIMAL(6,2) NOT NULL, /*BR-2a*/
  stock       INT,
  PRIMARY KEY(productId),
  CONSTRAINT BR_2b_price_range CHECK (price >= 0), /*BR-2b*/
  CONSTRAINT BR_2c_positive_stock CHECK (stock >= 0 ) /*BR-2c*/
);

/* Relation: Orders*/
CREATE OR REPLACE TABLE Orders(
  orderId     INT NOT NULL AUTO_INCREMENT,
  userId      INT NOT NULL,
  productId   INT NOT NULL,
  amount      INT DEFAULT(1), /*BR-3a*/
  purchaseDate DATE DEFAULT(NOW()), /*BR-3a*/
  PRIMARY KEY(orderId),
  FOREIGN KEY(userId) REFERENCES Users(userId),
  FOREIGN KEY(productId) REFERENCES Products(productId),
  CONSTRAINT BR_3b_amount_range CHECK (amount BETWEEN 0 AND 10), /*BR-3b*/
  CONSTRAINT BR_3c_month_Not_August CHECK ( MONTH(purchaseDate)<>8 ) /*BR-3c*/

```

```

);

/*Relation: Notifications*/
CREATE TABLE Notifications(
    notificationId INT NOT NULL AUTO_INCREMENT,
    timestamp DATETIME NOT NULL,
    message VARCHAR(300) NOT NULL,
    PRIMARY KEY (notificationId)
);

/* User Data */
INSERT INTO Users(name, province, startDate) VALUES
    ('David_Ruiz', 'Sevilla', '2018-05-18'),
    ('Marta_López', 'London', '2018-06-12'),
    ('Raquel_Lobato', 'Granada', '2018-12-01'),
    ('Antonio_Gómez', 'Sevilla', '2018-03-11'),
    ('Inma_Hernández', 'London', '2018-04-12'),
    ('Jimena_Martín', 'Helsinki', '2018-05-13'),
    ('Carlos_Rivero', 'Rochester', '2018-09-07'),
    ('Carlos_Arévalo', 'London', '2018-09-07')
;

/* Product data */
INSERT INTO Products(description, price, stock) VALUES
    ('Motorola_Razr_5G', 190.90, 50),
    ('GoPro_Hero_10_Black', 396.80, 20),
    ('American_Soft_Linen_Bath_Towels', 15.90, 150),
    ('Kasa_Smart_Plug_HS103P4', 20.23, 25),
    ('LEVOIT_Air_Purifiers_for_Home', 174.90, 50),
    ('BISSELL_Steam_Shot', 27.40, 50);

/* Order data */
INSERT INTO Orders(userId, productId, amount, purchaseDate) VALUES
    (1,1,2,'2019-05-13'),
    (1,3,2,'2019-05-13'),
    (2,2,3,'2019-06-11'),
    (2,3,1,'2019-06-11'),
    (3,4,2,'2019-06-15'),
    (4,5,1,'2019-06-18'),
    (4,6,1,'2019-06-18'),
    (5,4,2,'2019-12-15'),
    (7,1,1,'2019-12-15'),
    (7,2,1,'2019-12-16'),
    (7,3,1,'2019-12-17'),
    (7,4,1,'2019-12-18'),
    (7,5,1,'2019-12-19'),
    (7,6,1,'2019-12-20'),
    (8,1,1,'2019-12-15');

/*Initially, notifications table will be empty */

```

B.1.3. Enunciado

1. Cree un procedimiento que, para un usuario, producto y cantidad dados, realice un pedido para ese usuario, producto y cantidad, usando la fecha actual como fecha de

compra.

2. Cree un procedimiento que, dado un porcentaje de aumento/disminución (por ejemplo, -0.2), modifique los precios de todos los productos en el porcentaje dado (por ejemplo, para un valor de -0.2, un producto que cuesta 10€ pasaría a costar 8€).
3. Cree un procedimiento que, a partir de un mensaje, inserte una nueva modificación con ese mensaje.
4. Cree una función que, a partir de un ID de producto, devuelva la cantidad de unidades vendidas por ese producto.
5. Cree una función que devuelva el total del beneficio de las ventas entre dos fechas dadas.
6. Cree una función que, a partir de un año, devuelva el ID del producto que más ha vendido ese año.
7. Cree una función que, a partir de un ID de usuario, devuelva la cantidad de dinero gastada en todos sus pedidos.
8. Cree una función que, a partir de un producto, devuelva su stock actual.
9. Cree un disparador que, cuando se realiza un pedido, actualice el stock del producto pedido (BR-4).
10. Cree un disparador que muestre un mensaje de error si un usuario intenta realizar más de 3 pedidos el mismo día, y añada una nueva notificación incluyendo el ID del producto y del usuario (BR-5).
11. Cree un disparador que inserta una nueva notificación si el stock de un producto pasa a estar por debajo de las 10 unidades. El mensaje debe incluir el ID del producto y las unidades que quedan en el stock.
12. Cree un disparador que inserta una notificación si la cantidad de un pedido es mayor que tres veces la media de los pedidos de ese producto. El mensaje de notificación debería incluir el ID del producto, la cantidad pedida, y la media de los pedidos. El disparador será disparado antes de insertar cualquier pedidos. Por ejemplo, si la media de unidades pedidas de un producto es 2, se debería de insertar una notificación si alguien pide más de 6 unidades.
13. Cree un procedimiento que, para un nombre de usuario y año dados, cree una vista `vOrders<nombre><año>` que contenga los pedidos realizados por ese usuario en ese año. Recuerde que un nombre puede contener espacios en blanco que deben ser eliminados.
14. Cree un procedimiento que, para un entero N y un año, cree una vista a partir del ejercicio anterior para todos los usuarios que han realizado más de N pedidos.
15. Cree un procedimiento que pruebe el procedimiento del ejercicio #1 (inserción de pedidos). El test consiste en los siguientes pasos:
 - a) Re-crear la base de datos (crear tablas e insertar datos).

- b) Llamar al procedimiento con datos correctos seleccionados de forma automática de la base de datos.
 - c) Por cada escenario de test, comprobar que los resultados de ejecutar el procedimiento son los esperados, y si no lanzar un error.
16. Cree una colección de procedimientos que prueben el procedimiento del ejercicio #1, pero esta vez bajo diferentes escenarios que involucren algún tipo de dato incorrecto (por ejemplo, hacer un pedido en agosto, con un número incorrecto de objetos, o con un ID de producto incorrecto).
17. Cree un procedimiento que pruebe la función del ejercicio #3 (que devuelve el número de unidades vendidas de un producto en particular). De nuevo, el test debe tener los siguientes pasos:
- a) Re-crear la base de datos.
 - b) Invocar la función.
 - c) Comprobar que el resultado es correcto (el esperado).

B.1.4. Soluciones

Procedimientos y funciones:

Listing B.2: src/BaseDatosPedidos/Exercises SQL-2(RFs. Functions and procedures).sql

```

/* Procedures, Functions and Triggers
Authors: Daniel Ayala, Inma Hernández y David Ruiz
Creation Date: November 2019
v1.1: November 2020 (Carlos Arévalo)
v1.2: November 2022 (Inma Hernández)
*/

-- Exercise 1: Procedure that, given a user, a product, and an amount
-- , creates an order with the current date.
DELIMITER //
CREATE OR REPLACE PROCEDURE
  pInsertOrder(userId INT, productId INT, amount INT)
BEGIN
  INSERT INTO Orders(userId, productId, amount, purchaseDate)
    VALUES (userId, productId, amount, NOW());
END //
DELIMITER ;

-- Exercise 2: Create a procedure that, given an increase / decrease
-- percentage value, modifies the prices of all products, in the given
-- percentage.

DELIMITER //
CREATE OR REPLACE PROCEDURE
  pChangePrices(fraction DOUBLE)
BEGIN
  UPDATE Products SET price = price * (1 + fraction);
END //
DELIMITER ;

```

```

-- Exercise 3: Create a procedure that, given a message,
-- inserts a new notification with that message
DELIMITER //
CREATE OR REPLACE PROCEDURE pAddNotification(IN msg VARCHAR(300))
BEGIN
    INSERT INTO notifications(TIMESTAMP, message)
    VALUES (NOW(), msg);

END //
DELIMITER ;

-- Exercise 4: Create a function that, for a given product ID, returns the amount of
-- sold units of that product.

DELIMITER //
CREATE OR REPLACE FUNCTION
    fGetSoldUnits(productId INT) RETURNS INT
BEGIN
    RETURN (
        SELECT SUM(amount)
        FROM Orders
        WHERE Orders.productId = productId
    );
END //
DELIMITER ;

-- Exercise 5: Function that returns the total sales revenue between two
-- given dates.
DELIMITER //
CREATE OR REPLACE FUNCTION
    fPurchaseBetweenDates(date1 DATE, date2 DATE) RETURNS DECIMAL
BEGIN
    RETURN (
        SELECT SUM(amount*price)
        FROM Orders NATURAL JOIN Products
        WHERE (purchaseDate >= date1 AND purchaseDate <= date2)
    );
END //
DELIMITER ;

-- Exercise 6: Function that, given a year,
-- returns the ID of the best-selling product that year
DELIMITER //
CREATE OR REPLACE FUNCTION
    fBestSeller(year INT) RETURNS INT
BEGIN
    RETURN (
        SELECT productId
        FROM Orders NATURAL JOIN Products
        WHERE YEAR(purchaseDate) = year
        GROUP BY productId
        ORDER BY SUM(amount) DESC
        LIMIT 1
    );
END //
DELIMITER ;

-- Exercise 7: Function that, given a user ID, returns the total amount spent
-- on orders

```

```

DELIMITER //
CREATE OR REPLACE FUNCTION
  fSpentMoneyUser(userId INT) RETURNS DOUBLE
BEGIN
  RETURN (
    SELECT SUM(amount*price)
    FROM Orders NATURAL JOIN Products
    WHERE (Orders.userId=userId)
  );
END //
DELIMITER ;

-- Exercise 8: Function that, given a product ID, returns its current stock.
DELIMITER //
CREATE OR REPLACE FUNCTION
  fGetStock(productId INT) RETURNS INT
BEGIN
  RETURN (
    SELECT stock
    FROM Products
    WHERE (Products.productId = productId)
  );
END //
DELIMITER ;

-- Exercise 13: Create a procedure that, for a given user name and year,
-- creates a view vOrders<Name><Year> that contains the orders placed
-- by that user that year.
DELIMITER //
CREATE OR REPLACE PROCEDURE
  pCreateOrdersView(name VARCHAR(64), ordersYear INT)
BEGIN
  DECLARE userId INT;
  SET userId = (SELECT Users.userId FROM Users WHERE Users.name = NAME);
  EXECUTE IMMEDIATE CONCAT('CREATE OR REPLACE VIEW vOrders', REPLACE(NAME, ' ', ''),
    ordersYear, ' AS
  SELECT * FROM ORDERS WHERE userId=', userId, ' AND YEAR(purchaseDate)=', ordersYear, '
  ');
END //
DELIMITER ;

-- Exercise 14: Create a procedure that, for a given integer N and a year,
-- creates the view from the former exercise for all the users who have
-- placed more than N orders.
DELIMITER //
CREATE OR REPLACE PROCEDURE
  pCreateOrdersViews(minOrders INT, ordersYear INT)
BEGIN
  DECLARE userName VARCHAR(64);
  DECLARE done BOOLEAN DEFAULT FALSE;
  DECLARE usersWithOrders CURSOR FOR
    SELECT name
    FROM Users NATURAL JOIN Orders
    WHERE YEAR(purchaseDate) = ordersYear
    GROUP BY name
    HAVING COUNT(*) >= minOrders;
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done := TRUE;
  OPEN usersWithOrders;
  readLoop: LOOP

```

```

    FETCH usersWithOrders INTO userName;
    IF done THEN
        LEAVE readLoop;
    END IF;
    CALL pCreateOrdersView(userName, ordersYear);
END LOOP;
CLOSE usersWithOrders;
END //
DELIMITER ;

```

Disparadores:

Listing B.3: src/BaseDatosPedidos/Exercises SQL-2(Triggers).sql

```

/* Exercise 9
BR-4: Admit order only if there is enough stock
  IF ( Order.amount > Product.stock ) ** Update stock
    Product.stock = Product.stock- Order.amount
    Add notification
  ELSE **Cancel order
*/
DELIMITER //
CREATE OR REPLACE TRIGGER
BR_4_tCheckEnoughStock BEFORE INSERT ON Orders FOR EACH ROW
BEGIN
    DECLARE currentStock INT;
    SET currentStock = ( SELECT stock FROM Products WHERE productId = new.productId);
    IF(new.amount > currentStock) THEN
        SIGNAL SQLSTATE '45000' SET message_text =
            'BR-4: It is not possible to place the order due to low stock';
    ELSE
        UPDATE Products SET stock=stock-new.amount
        WHERE productId=new.productId;

        call pAddNotification(CONCAT('User:', NEW.userId, ' placed pending order for '
            product:',
            NEW.productId,' with stock:', currentStock));
    END IF;
END //
DELIMITER ;

/* Exercise 10. BR-5: A user can place a maximum of three orders per day
*/
DELIMITER //
CREATE OR REPLACE TRIGGER
BR_5_tMaxOrdersDay BEFORE INSERT ON Orders FOR EACH ROW
BEGIN
    DECLARE ordersToday INT;
    SET ordersToday = (
        SELECT COUNT(*) FROM Orders
        WHERE (userId=new.userId AND purchaseDate=new.purchaseDate)
    );
    IF(ordersToday >= 3) THEN
        SIGNAL SQLSTATE '45000' SET message_text =
            'BR-5: It is not possible to place more than 3 orders per day';
    END IF;
END //
DELIMITER ;

```

```

/* Exercise 11: Create a trigger that inserts a new notification if the stock of a
product is below 10 units. */

DELIMITER //
CREATE OR REPLACE TRIGGER
  tLowStockNotification AFTER UPDATE ON Products FOR EACH ROW
BEGIN
  DECLARE message VARCHAR(300);
  IF(new.stock < 10) THEN
    SET message = CONCAT(
      'There are ',
      new.stock,
      ' units remaining for product ',
      new.productId, '.'
    );
    call pAddNotification(message);
  END IF;
END //
DELIMITER ;

/* Exercise 12: Create a trigger that inserts a notification if the amount of an order
is higher than
three times the average ordered amount for that product. */
DELIMITER //
CREATE OR REPLACE TRIGGER
  tBigOrderNotification BEFORE INSERT ON Orders FOR EACH ROW
BEGIN
  DECLARE message VARCHAR(300);
  DECLARE avgOrder DOUBLE;
  SET avgOrder = (
    SELECT AVG(amount)
    FROM Orders
    WHERE productId = new.productId
  );
  IF(new.amount > 3*avgOrder) THEN
    SET message = CONCAT(
      'A new order has been placed for ',
      new.amount,
      ' units of product ',
      new.productId,
      ', whereas the average amount ordered is ',
      avgOrder,
      '.'
    );
    call pAddNotification (message);
  END IF;
END //
DELIMITER ;

```

Tests:

Listing B.4: src/BaseDatosPedidos/Exercises SQL-2(Tests).sql

```

-- Exercises - Procedures, Functions and Triggers
-- Authors: (Carlos Arévalo, Inma Hernández, Daniel Ayala, David Ruiz)
-- Creation date: November 2020
-- v1.1: November 2022 (Inma Hernández)

-- Exercise 15: Cree un procedimiento que pruebe el procedimiento que

```

```

-- inserta un pedido.

DELIMITER //
CREATE OR REPLACE PROCEDURE
  pTestPInsertOrderSuccess()
BEGIN
  DECLARE uId INT;
  DECLARE uidCheck INT;
  DECLARE pId INT;
  DECLARE pIdcheck INT;
  DECLARE pDate DATE;
  CALL pCreateTables();
  CALL pInsertData();

  -- Find a existing user
  SELECT userId INTO uId
  FROM users
  LIMIT 1;

  -- Find a existing product
  SELECT productId INTO pId
  FROM products
  LIMIT 1;

  -- place order with minimum amount (1)
  -- could we automatically retrieve the id of the new order placed?
  CALL pInsertOrder(uId, pId, 1);

  -- check result (Assuming new order id is 16)
  SELECT Orders.userId, Orders.productId, Orders.purchaseDate
  INTO uidCheck, pIdcheck, pDate
  FROM ORDERS
  WHERE orderId = 16;

  IF ( (uId != uidCheck) OR
        (pId != pIdcheck) OR
        (YEAR(pDate) != YEAR(SYSDATE())) OR
        (MONTH(pDate) != MONTH(SYSDATE())) OR
        (DAY(pDate) != DAY(SYSDATE()))) THEN
    SIGNAL SQLSTATE '45000' SET message_text =
      'Test_for_pInsertOrder_failed';
  END IF;
END //
DELIMITER ;

-- Exercise 17: Create a procedure that tests the function in exercise 3
DELIMITER //
CREATE OR REPLACE PROCEDURE
  pTestFGetSoldUnitsSuccess()
BEGIN
  DECLARE soldUnits INT;
  CALL pCreateTables();
  CALL pinserData();
  SET soldUnits = fGetSoldUnits(1);
  IF (soldUnits != 4) THEN
    SIGNAL SQLSTATE '45000' SET message_text =
      'Test_for_fGetSoldUnits_failed';
  END IF;
END //

```

```

DELIMITER ;

-- Auxiliary procedures to re-create the DB schema
DELIMITER //
CREATE OR REPLACE PROCEDURE
  pCreateTables()
BEGIN
  DROP TABLE IF EXISTS Orders;
  DROP TABLE IF EXISTS Products;
  DROP TABLE IF EXISTS Users;
  DROP TABLE IF EXISTS Notifications;

  /* Relation: Users */
  CREATE OR REPLACE TABLE Users(
    userId      INT NOT NULL AUTO_INCREMENT,
    `name`      VARCHAR(63) NOT NULL, /*BR-1a*/
    province    VARCHAR(63),
    startDate   DATE NOT NULL, /*BR-1a*/
    PRIMARY KEY(userId),
    CONSTRAINT BR_1b_nameUnique UNIQUE(NAME) /*BR-1b*/
  );

  /* Relation: Products */
  CREATE OR REPLACE TABLE Products(
    productId   INT NOT NULL AUTO_INCREMENT,
    description  VARCHAR(128),
    price       DECIMAL(6,2) NOT NULL, /*BR-2a*/
    stock       INT,
    PRIMARY KEY(productId),
    CONSTRAINT BR_2b_price_range CHECK (price >= 0), /*BR-2b*/
    CONSTRAINT BR_2c_positive_stock CHECK (stock >= 0 ) /*BR-2c*/
  );

  /* Relation: Orders*/
  CREATE OR REPLACE TABLE Orders(
    orderId     INT NOT NULL AUTO_INCREMENT,
    userId      INT NOT NULL,
    productId   INT NOT NULL,
    amount      INT DEFAULT(1), /*BR-3a*/
    purchaseDate DATE DEFAULT(NOW()), /*BR-3a*/
    PRIMARY KEY(orderId),
    FOREIGN KEY(userId) REFERENCES Users(userId),
    FOREIGN KEY(productId) REFERENCES Products(productId),
    CONSTRAINT BR_3b_amount_range CHECK (amount BETWEEN 0 AND 10), /*BR-3b*/
    CONSTRAINT BR_3c_month_Not_August CHECK ( MONTH(purchaseDate)<>8 ) /*BR-3c*/
  );

  /*Relation: Notifications*/
  CREATE TABLE Notifications(
    notificationId INT NOT NULL AUTO_INCREMENT,
    timestamp      DATETIME NOT NULL,
    message        VARCHAR(300) NOT NULL,
    PRIMARY KEY (notificationId)
  );
END //
DELIMITER ;

```

```
-- Cree un procedimiento que inserte datos de ejemplo en las tablas.
DELIMITER //
CREATE OR REPLACE PROCEDURE
  pInsertData()
BEGIN
  INSERT INTO Users(name, province, startDate) VALUES
    ('David_Ruiz', 'Sevilla', '2018-05-18'),
    ('Marta_López', 'London', '2018-06-12'),
    ('Raquel_Lobato', 'Granada', '2018-12-01'),
    ('Antonio_Gómez', 'Sevilla', '2018-03-11'),
    ('Inma_Hernández', 'London', '2018-04-12'),
    ('Jimena_Martín', 'Helsinki', '2018-05-13'),
    ('Carlos_Rivero', 'Rochester', '2018-09-07'),
    ('Carlos_Arévalo', 'London', '2018-09-07')
  ;

  /* Product data */
  INSERT INTO Products(description, price, stock) VALUES
    ('Motorola_Razr_5G', 190.90, 50),
    ('GoPro_Hero_10_Black', 396.80, 20),
    ('American_Soft_Linen_Bath_Towels', 15.90, 150),
    ('Kasa_Smart_Plug_HS103P4', 20.23, 25),
    ('LEVOIT_Air_Purifiers_for_Home', 174.90, 50),
    ('BISSELL_Steam_Shot', 27.40, 50);

  /* Order data */
  INSERT INTO Orders(userId, productId, amount, purchaseDate) VALUES
    (1,1,2,'2019-05-13'),
    (1,3,2,'2019-05-13'),
    (2,2,3,'2019-06-11'),
    (2,3,1,'2019-06-11'),
    (3,4,2,'2019-06-15'),
    (4,5,1,'2019-06-18'),
    (4,6,1,'2019-06-18'),
    (5,4,2,'2019-12-15'),
    (7,1,1,'2019-12-15'),
    (7,2,1,'2019-12-16'),
    (7,3,1,'2019-12-17'),
    (7,4,1,'2019-12-18'),
    (7,5,1,'2019-12-19'),
    (7,6,1,'2019-12-20'),
    (8,1,1,'2019-12-15');
END //
DELIMITER ;
```