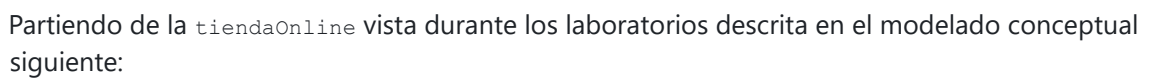


¿Cómo consigo coins?  Plan Turbo: barato  
 Planes pro: más coins

## Enunciado Evaluación Individual de Laboratorio Modelo C

**Si usted entrega sin haber sido verificada su identidad no podrá ser evaluado. El procedimiento de entrega se encuentra al final de este documento.**

## Tienda Online



The diagram illustrates the system architecture with the following components and relationships:

- «enumeration» TipoProducto**:
  - Físico
  - Digital
- «Entity» Producto**:
  - nombre
  - descripcion[0..1]
  - precioUnitario
  - puedeVenderseAMenores = true
  - tipo: TipoProducto
- «Entity» Pedido**:
  - fechaRealizacion
  - fechaEnvio[0..1]
  - direccionEntrega
  - comentarios[0..1]
  - /precio
- «Entity» Empleado**:
  - salario
- «Entity» Cliente**:
  - direccionEnvio
  - codigoPostal
  - fechaNacimiento
- «Entity» LineaPedido**:
  - unidades
  - precioUnitario
- «Entity» Usuario**:
  - email
  - contraseña
  - nombre

**Relationships:**

- Producto** \* **LineaPedido** (contiene): Multiplicity \* on both ends.
- Empleado** 0..1 **Pedido** (gestiona): Multiplicity 0..1 on Empleado, \* on Pedido.
- Empleado** \* **Pedido** (gestiona): Multiplicity \* on both ends.
- Empleado** \* **Cliente** (realiza): Multiplicity \* on Empleado, 1 on Cliente.
- Producto** 1 **Usuario** (generalization): Multiplicity 1 on Producto, \* on Usuario.
- Cliente** \* **Usuario** (generalization): Multiplicity \* on Cliente, 1 on Usuario.

**Requirements (RN):**

- RN-001**: La contraseña debe ser de al menos 8 caracteres (linked to Usuario).
- RN-002**: El cliente debe ser mayor de 14 años (linked to Cliente).
- RN-003**: Los clientes menores de 18 años no podrán pedir productos que !puedeVenderseAMenores (linked to Pedido).
- RN-004**: El precio unitario no puede ser negativo (linked to LineaPedido).
- RN-005**: La cantidad tiene que ser mayor que 0 y no puede superar las 100 unidades (linked to LineaPedido).

Las tablas y datos de prueba iniciales se encuentran en los ficheros `0.creacionTablas.sql` y `0.poblarBd.sql`.

Realice los siguientes ejercicios:

### 1. Creación de tabla. (1,5 puntos)

Incluya su solución en el fichero `1.solucionCreacionTabla.sql`.

Necesitamos conocer la opinión de nuestros clientes sobre nuestros productos. Para ello se propone la creación de una nueva tabla llamada `valoraciones`. Cada valoración versará sobre un producto y será realizada por un solo cliente. Cada producto podrá ser valorado por muchos clientes. Cada cliente podrá realizar muchas valoraciones. Un cliente no puede valorar más de una vez un mismo producto.

Para cada valoración necesitamos conocer la puntuación de 1 a 5 (sólo se permiten enteros) y la fecha en que se realiza la valoración.

## 2. Consultas SQL (DQL). 3 puntos

Incluya su solución en el fichero `2.solucionConsultas.sql`.

**2.1.** Devuelva el nombre del producto, el precio unitario y las unidades compradas para las 5 líneas de pedido con más unidades. **(1 punto)**

**2.3.** Devuelva el nombre del empleado, la fecha de realización del pedido, el precio total del pedido y las unidades totales del pedido para todos los pedidos que de más 7 días de antigüedad desde que se realizaron. Si un pedido no tiene asignado empleado, también debe aparecer en el listado devuelto. **(2 puntos)**

## 3. Procedimiento. Bonificar pedido retrasado. 3,5 puntos

Incluya su solución en el fichero `3.solucionProcedimiento.sql`.

Cree un procedimiento que permita bonificar un pedido que se ha retrasado debido a la mala gestión del empleado a cargo. Recibirá un identificador de pedido, asignará a otro empleado como gestor y reducirá un 20% el precio unitario de cada línea de pedido asociada a ese pedido. **(1,5 puntos)**

Asegure que el pedido estaba asociado a un empleado y en caso contrario lance excepción con el siguiente mensaje: **(1 punto)**

`El pedido no tiene gestor.`

Garantice que o bien se realizan todas las operaciones o bien no se realice ninguna. **(1 punto)**

## 4. Trigger. 2 puntos

Incluya su solución en el fichero `4.solucionTrigger.sql`.

Cree un trigger llamado `p_limitar_unidades_mensuales_de_productos_fisicos` que, a partir de este momento, impida la venta de más de 1000 unidades al mes de cualquier producto físico.

# SOLUCIONES

## 1. Creación de tabla.

```
DROP TABLE if EXISTS valoraciones;

CREATE TABLE valoraciones (
  id INT PRIMARY KEY AUTO_INCREMENT,
  productoId INT NOT NULL,
  clienteId INT NOT NULL,
  puntuacion INT NOT NULL CHECK (puntuacion>1 AND puntuacion <5),
  fecha DATE NOT NULL,
  UNIQUE(productoId,clienteId),
  FOREIGN KEY (clienteId) REFERENCES clientes(id),
  FOREIGN KEY (productoId) REFERENCES productos(id)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);
```

## 2. Consultas SQL (DQL).

```
-- Apt1
SELECT productos.nombre, productos.precio,lineaspedido.unidades
FROM productos
JOIN lineaspedido ON productos.id=lineaspedido.productoId
ORDER BY lineaspedido.unidades DESC
LIMIT 5;

-- Apt3
SELECT
  u.nombre AS empleado,
  p.fechaRealizacion,
  SUM(lp.unidades * lp.precio) AS precio_total,
  lp.unidades AS unidades_totales
FROM Pedidos p
LEFT JOIN Empleados e ON p.empleadoId = e.id
LEFT JOIN Usuarios u ON e.usuarioId = u.id
JOIN LineasPedido lp ON p.id = lp.pedidoId
WHERE p.fechaRealizacion < CURDATE() - INTERVAL 7 DAY
GROUP BY p.id, u.nombre, p.fechaRealizacion;
```



Importante

Puedo eliminar la publi de este documento con 1 coin

¿Cómo consigo coins? → Plan Turbo: barato  
→ Planes pro: más coins

### 3. Procedimiento. Bonificar pedido retrasado

```
DELIMITER //
CREATE OR REPLACE PROCEDURE bonificar_por_retraso(IN p_id INT)
BEGIN
    DECLARE empleado INT;
    DECLARE pedido ROW TYPE OF pedidos;
    DECLARE nuevoEmpleado INT;

    -- Manejo de errores
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        ROLLBACK;
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Error al bonificar el pedido';
    END;

    -- Iniciar transacción
    START TRANSACTION;

    SET pedido = (SELECT *
                  FROM pedidos p
                  WHERE p.id=p_id);

    SET empleado = (SELECT e.id
                   FROM empleados e
                   WHERE e.id = pedido.empleadoId);

    IF empleado IS NULL THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'El pedido no tiene gestor.';
    END IF;

    -- Contar cuántos clientes premium se han registrado este mes
    SELECT e.id
    INTO nuevoEmpleado
    FROM empleados e
    WHERE e.id !=empleado
    LIMIT 1;

    UPDATE pedidos
    SET empleadoId = nuevoEmpleado
    WHERE id = p_id;

    UPDATE lineaspedido
    SET precio = precio * 0.8
    WHERE pedidoId = p_id;

    -- Confirmar transacción
    COMMIT;
END//
DELIMITER ;
```

perdo  
espacio



Necesito  
concentración

ali ali ooh  
esto con 1 coin me  
lo quito yo...

WUOLAH



## 4. Trigger.

```
DELIMITER //
```

```
CREATE OR REPLACE TRIGGER p_limitar_unidades_mensuales_de_productos_fisicos
BEFORE INSERT ON LineasPedido
FOR EACH ROW
BEGIN
    DECLARE tipo_producto INT;
    DECLARE totalUnidades INT;
    DECLARE fecha DATE;

    -- Obtener el tipo de producto
    SELECT prd.tipoProductoId INTO tipo_producto
    FROM Productos prd
    WHERE prd.id = NEW.productoId;

    -- Solo aplicar la regla si es producto físico (tipo = 1)
    IF (tipo_producto = 1) THEN

        -- Obtener la fecha de realización del pedido
        SELECT p.fechaRealizacion INTO fecha
        FROM Pedidos p
        WHERE p.id = NEW.pedidoId;

        -- Calcular las unidades acumuladas de ese producto en el mismo mes/año
        SELECT SUM(lp.unidades) INTO totalUnidades
        FROM LineasPedido lp
        JOIN Pedidos p ON lp.pedidoId = p.id
        WHERE lp.productoId = NEW.productoId AND MONTH(CURDATE()) = MONTH(fecha) AND YEAR(CURDATE()) = YEAR(fecha);

        -- Verificar si supera el límite con la nueva inserción
        IF (NEW.unidades + totalUnidades) > 1000 THEN
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'No se permiten más de 1000 unidades de este producto físico en el mismo mes.';
        END IF;

    END IF;
END //
```

```
DELIMITER ;
```

IISSI-1 Prueba de Laboratorio. Sesión 0	Curso 2019-20 Enero de 2020
Apellidos, Nombre:	Grupo:

**Calificación Final:** \_\_\_\_\_

**Pregunta 0. (1 punto)**

Utilice HediSQL para establecer una conexión local con usuario iissi\_user y clave iissi\$user. Cree una nueva base de datos llamada <<UVUS>>\_sesion0. Ejecute el script create\_db. Para asegurar que todo es correcto ejecute la consulta SELECT count(\*) FROM Students; y compruebe que el resultado que devuelve es 21.

**SELECT count(\*) FROM students;**

**Calificación:** \_\_\_\_\_

IISSI-1 Prueba de Laboratorio. Sesión 0	Curso 2019-20 Enero de 2020
Apellidos, Nombre:	Grupo:

**Pregunta 1. (2 puntos)**

Añada el requisito de información **Alumno Interno**. Un alumno interno es un estudiante que colabora con un Departamento en actividades docentes o de investigación. Sus atributos son: el departamento en el que el estudiante participa como alumno interno, el estudiante involucrado, el año académico en el que se hace la colaboración y el número de meses que dura la colaboración. Hay que tener en cuenta las siguientes restricciones:

- Los estudiantes sólo pueden ser alumnos internos una vez en un único curso académico.
- El número de meses de la colaboración debe ser como máximo de 9 meses y como mínimo de 3.
- Todos los atributos son obligatorios, menos el número de meses de la colaboración.

```
CREATE TABLE InternalStudents(
InternalStudentId INT NOT NULL AUTO_INCREMENT,
departmentId INT NOT NULL,
studentId INT NOT NULL,
academicYear INT NOT NULL,
months INT,
PRIMARY KEY(InternalStudentId),
FOREIGN KEY(studentId) REFERENCES Students(studentId),
FOREIGN KEY(departmentId) REFERENCES
Departments(departmentId),
CONSTRAINT invalidNumberOfMonths CHECK (months >= 3 AND
months <=9),
UNIQUE(studentId, academicYear)
);
```

Calificación: \_\_\_\_\_

IISSI-1 Prueba de Laboratorio. Sesión 0	Curso 2019-20 Enero de 2020
Apellidos, Nombre:	Grupo:

**Pregunta 2. (1 punto)**

Cree un procedimiento almacenado llamado pInsertInterns () que cree los siguientes alumnos internos:

- Alumno interno del estudiante con ID=1, en el departamento con ID=1, en el año académico 2019, con una duración de 3 meses.
- Alumno interno del estudiante con ID=1, en el departamento con ID=1, en el año académico 2020, con una duración de 6 meses.
- Alumno interno del estudiante con ID=2, en el departamento con ID=1, en el año académico 2019.

```

DELIMITER //
CREATE OR REPLACE PROCEDURE pInsertInterns()
BEGIN
INSERT INTO internalStudents(departmentId, studentId, academicYear,
months) VALUES (1,1,2019,3),
(1,1,2020,6),
(1,2,2019,NULL);
END//
DELIMITER ;

```



IISSI-1 Prueba de Laboratorio. Sesión 0	Curso 2019-20 Enero de 2020
Apellidos, Nombre:	Grupo:

**Pregunta 3. (1 punto)**

Cree un procedimiento almacenado llamado pUpdateInterns(s, d) que actualiza la duración de los alumnos internos correspondientes al estudiante con ID=s con el valor d. Ejecute la llamada a pUpdateInterns(1,9)

Cree un procedimiento almacenado llamado pDeleteInterns(s) que elimina los alumnos internos correspondientes al estudiante con ID=s. Ejecute la llamada pDeleteInterns(2)

```

DELIMITER //
CREATE OR REPLACE PROCEDURE pInsertInterns()
BEGIN
INSERT INTO internalStudents(departmentId, studentId, academicYear,
months) VALUES (1,1,2019,3),
(1,1,2020,6),
(1,2,2019,NULL);
END//
DELIMITER ;

```

```

DELIMITER //
CREATE OR REPLACE PROCEDURE pUpdateInterns(s INT, d INT)
BEGIN
UPDATE InternalStudents SET months=d WHERE internalStudentId=s;
END//
CREATE OR REPLACE PROCEDURE pDeleteInterns(s INT)
BEGIN
DELETE FROM InternalStudents WHERE studentId = s;
END//
DELIMITER ;


```

Calificación: \_\_\_\_\_

IISSI-1 Prueba de Laboratorio. Sesión 0	Curso 2019-20 Enero de 2020
Apellidos, Nombre:	Grupo:

**Pregunta 4. (1 punto)**

Cree una consulta que devuelva el nombre del profesor, el nombre del grupo, y los créditos que imparte en él para todas las imparticiones de asignaturas por profesores. Un ejemplo de resultado de esta consulta es el siguiente:

Resultado #1 (3×4)		
professorName	 group	credits
Fernando	T1	6
David	T1	12
Fernando	T2	6
Fernando	L1	12

```
SELECT p.firstName, g.name, tl.credits FROM professors P, groups G,
teachingLoads tl WHERE p.professorId = tl.professorId AND g.groupId =
tl.groupId;
```

Calificación:

IISSI-1 Prueba de Laboratorio. Sesión 0	Curso 2019-20 Enero de 2020
Apellidos, Nombre:	Grupo:

**Pregunta 5. (1 punto)**

Cree una consulta que devuelva las tutorías con al menos una cita. Un ejemplo de resultado de la consulta anterior es el siguiente:

tutoringhours (1×3)	
tutoringHoursId	
1	
2	
4	

SELECT tutoringHoursId FROM appointments;

Calificación: \_\_\_\_\_

IISSI-1 Prueba de Laboratorio. Sesión 0	Curso 2019-20 Enero de 2020
Apellidos, Nombre:	Grupo:

**Pregunta 6. (1 punto)**

Cree una consulta que devuelva el nombre y apellidos de los profesores con un despacho en la planta 0. Un ejemplo de resultado de la consulta anterior es el siguiente:

professors (2×1)	
firstName	surname
Inma	Hernández

```
SELECT firstName, surname FROM professors p, offices o WHERE
p.officeld=o.officeld AND o.floor = 0;
```

Calificación: \_\_\_\_\_

IISSI-1 Prueba de Laboratorio. Sesión 0	Curso 2019-20 Enero de 2020
Apellidos, Nombre:	Grupo:

**Pregunta 7. (1 punto)**

Cree una consulta que devuelva, por cada método de acceso, la media de las notas suspensas, ordenados por esta última de menor a mayor (no tienen que aparecer los métodos de acceso que no se den en ningún alumno o nota). Un ejemplo de resultado de la consulta anterior es el siguiente:

students (2x2)	
accessMethod	avgGrade
Titulado Extranjero	0,583333
Selectividad	3,416667

```
CREATE OR REPLACE VIEW notasAlumnoMetodo AS (SELECT accessMethod,
VALUE FROM Students s, grades g WHERE g.studentId = s.studentId);
SELECT * FROM notasAlumnoMetodo;
SELECT accessMethod, AVG(VALUE) FROM notasAlumnoMetodo GROUP
BY(accessMethod);
```

Calificación: \_\_\_\_\_



IISSI-1 Prueba de Laboratorio. Sesión 0	Curso 2019-20 Enero de 2020
Apellidos, Nombre:	Grupo:

**Pregunta 8. (1 punto)**

Cree una consulta que devuelva el nombre y los apellidos de los dos estudiantes con mayor nota media, sus notas medias, y sus notas más baja. Un ejemplo de resultado de la consulta anterior es el siguiente:

students (4×2)			
firstName	surname	avgGrade	minGrade
Daniel	Pérez	6,300000	3,25
Rafael	Ramírez	5,833333	2,50

```
CREATE OR REPLACE VIEW gradesPerStudents AS (SELECT s.studentId,
firstName, surname, VALUE FROM Students s, Grades g WHERE g.studentId
= s.studentId);
SELECT firstName, surname, AVG(VALUE), MIN(VALUE) FROM
gradesPerStudents GROUP BY studentId ORDER BY AVG(VALUE) DESC LIMIT 2;
```

Calificación: \_\_\_\_\_

Importante

Puedo eliminar la publi de este documento con 1 coin

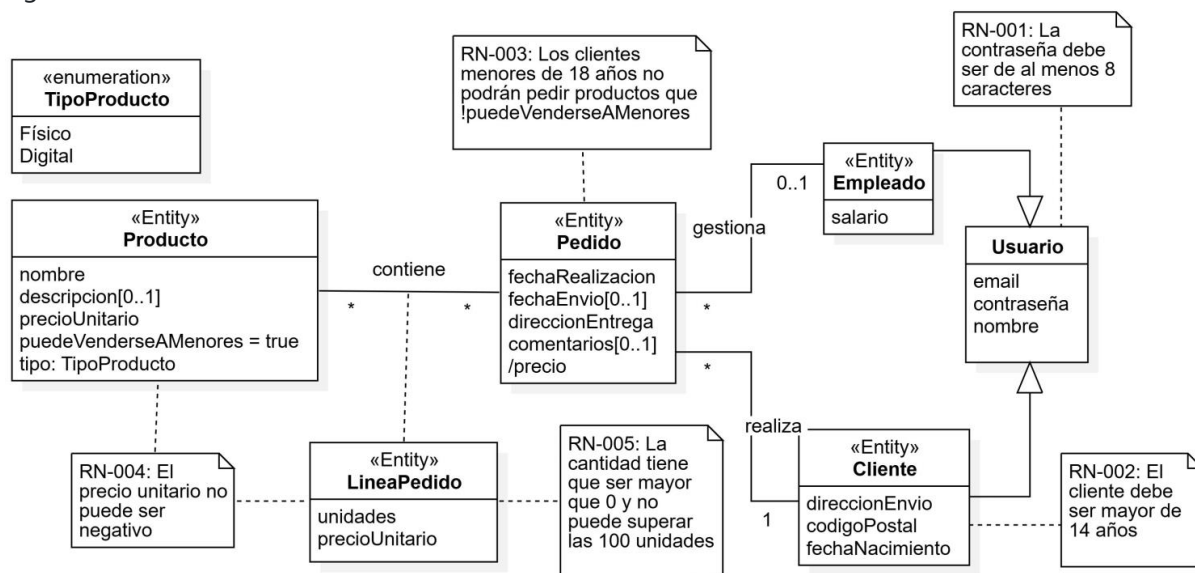
¿Cómo consigo coins? → Plan Turbo: barato  
→ Planes pro: más coins

# Enunciado Evaluación Individual de Laboratorio Modelo A

Si usted entrega sin haber sido verificada su identidad no podrá ser evaluado. El procedimiento de entrega se encuentra al final de este documento.

## Tienda Online

Partiendo de la tiendaOnline vista durante los laboratorios descrita en el modelado conceptual siguiente:



Las tablas y datos de prueba iniciales se encuentran en los ficheros 0.creacionTablas.sql y 0.poblarBd.sql.

Realice los siguientes ejercicios:

### 1. Creación de tabla. (1,5 puntos)

Incluya su solución en el fichero 1.solucionCreacionTabla.sql.

Necesitamos conocer la garantía de nuestros productos. Para ello se propone la creación de una nueva tabla llamada `Garantias`. Cada producto tendrá como máximo una garantía (no todos los productos tienen garantía), y cada garantía estará relacionada con un producto.

Para cada garantía necesitamos conocer la fecha de inicio de la garantía, la fecha de fin de la garantía, si tiene garantía extendida o no.

Asegure que la fecha de fin de la garantía es posterior a la fecha de inicio.

WUOLAH

## 2. Consultas SQL (DQL). (3 puntos)

Incluya su solución en el fichero `2.solucionConsultas.sql`.

**2.1.** Devuelva el nombre del producto, nombre del tipo de producto, y precio unitario al que se vendieron los productos digitales **(1 punto)**

**2.2.** Consulta que devuelva el nombre del empleado, el número de pedidos de más de 500 euros gestionados en este año y el importe total de cada uno de ellos, ordenados de mayor a menor importe gestionado. Los empleados que no hayan gestionado ningún pedido, también deben aparecer. **(2 puntos)**

## 3. Procedimiento. Actualizar precio de un producto y líneas de pedido no enviadas. (3,5 puntos)

Incluya su solución en el fichero `3.solucionProcedimiento.sql`.

Cree un procedimiento que permita actualizar el precio de un producto dado y que modifique los precios de las líneas de pedido asociadas al producto dado solo en aquellos pedidos que aún no hayan sido enviados. **(1,5 puntos)**

Asegure que el nuevo precio no sea un 50% menor que el precio actual y lance excepción si se da el caso con el siguiente mensaje: **(1 punto)**

No se permite rebajar el precio más del 50%.

Garantice que o bien se realizan todas las operaciones o bien no se realice ninguna. **(1 punto)**

## 4. Trigger. 2 puntos.

Incluya su solución en el fichero `4.solucionTrigger.sql`.

Cree un trigger llamado `t_asegurar_mismo_tipo_producto_en_pedidos` que impida que, a partir de ahora, un mismo pedido incluya productos físicos y digitales.

```

CREATE OR REPLACE TABLE garantias(
    id INT AUTO_INCREMENT PRIMARY KEY,
    productoId INT,
    fechaInicio DATE NOT NULL,
    fechaFin DATE NOT NULL,
    extendida BOOLEAN,
    CONSTRAINT RN CHECK(fechaFin > fechaInicio),
    UNIQUE(productoId),
    FOREIGN KEY (productoId) REFERENCES productos (id)
);

```

-- Ej 2.1

```

SELECT pr.nombre, tp.nombre, lp.precio
FROM productos pr
JOIN tiposproducto tp ON pr.tipoProductoId = tp.id
JOIN lineaspedido lp ON lp.productoId = pr.id
WHERE tp.nombre = 'Digitales';

```

-- Ej 2.2

```

SELECT u.nombre, COUNT(p.id) AS numPedidos, SUM(lp.precio * lp.unidades) AS importe
FROM usuarios u
JOIN empleados e ON e.usuarioId = u.id
LEFT JOIN pedidos p ON p.empleadoId = e.id AND YEAR(p.fechaRealizacion) = YEAR(CURDATE()) -1
JOIN lineaspedido lp ON lp.pedidoId = p.id
GROUP BY e.id
HAVING importe > 500
ORDER BY importe DESC;

```

```

DELIMITER //
CREATE OR REPLACE PROCEDURE actualizaPrecio(IN idProducto INT, IN nuevoPrecio DECIMAL(10,2))
BEGIN
    DECLARE precActual DECIMAL(10,2);
    -- Manejo de errores
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        ROLLBACK;
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Error al modificar precio';
    END;

    START TRANSACTION;

    SET precActual = (SELECT pr.precio FROM productos pr WHERE pr.id = idProducto);

    IF nuevoPrecio < 0.5*precActual THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'El no se puede rebajar menos de un 50% el precio del producto';
    END IF;

    UPDATE productos SET precio = nuevoPrecio WHERE id = idProducto;
    UPDATE lineaspedido lp JOIN pedidos p ON lp.pedidoId = p.id SET lp.precio = nuevoPrecio WHERE lp.productoId = idProducto;

    COMMIT;
END //
DELIMITER ;

```

-- Ej 4

```

DELIMITER //
CREATE OR REPLACE TRIGGER t_asegurar_mismo_tipo_producto_en_pedidos
BEFORE INSERT ON lineaspedido
FOR EACH ROW
BEGIN
    DECLARE tipoNuevoProducto INT;
    DECLARE tipoYaExistente INT;

    SELECT pr.tipoProductoId INTO tipoNuevoProducto FROM productos pr WHERE pr.id = NEW.productoId);

    SELECT pr.tipoProductoId INTO tipoYaExistente FROM productos pr JOIN lineaspedido lp ON lp.productoId = pr.id WHERE lp.pedidoId = NEW.pedidoId

    IF tipoNuevoProducto != tipoYaExistente THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'No se puede mezclar productos físicos y digitales en un mismo pedido.';
    END IF;
END IF;
END //
DELIMITER ;

```

Importante

Puedo eliminar la publi de este documento con 1 coin

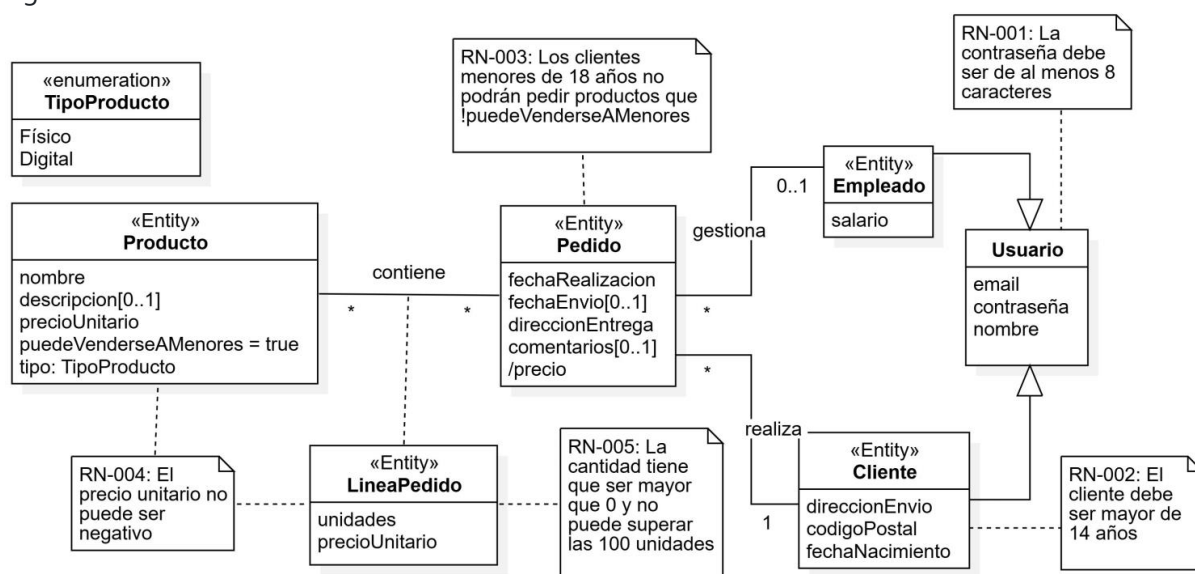
¿Cómo consigo coins? → Plan Turbo: barato  
→ Planes pro: más coins

## Enunciado Evaluación Individual de Laboratorio Modelo B

Si usted entrega sin haber sido verificada su identidad no podrá ser evaluado. El procedimiento de entrega se encuentra al final de este documento.

### Tienda Online

Partiendo de la tiendaOnline vista durante los laboratorios descrita en el modelado conceptual siguiente:



Las tablas y datos de prueba iniciales se encuentran en los ficheros 0.creacionTablas.sql y 0.poblarBd.sql.

Realice los siguientes ejercicios:

### 1. Creación de tabla. (1,5 puntos)

Incluya su solución en el fichero 1.solucionCreacionTabla.sql.

Necesitamos conocer los pagos que se realicen sobre los pedidos. Para ello se propone la creación de una nueva tabla llamada Pagos. Cada pedido podrá tener asociado varios pagos y cada pago solo corresponde con un pedido en concreto.

Para cada pago necesitamos conocer la fecha de pago, la cantidad pagada (que no puede ser negativa) y si el pago ha sido revisado o no (por defecto no estará revisado).

WUOLAH



## 2. Consultas SQL (DQL). 3 puntos

Incluya su solución en el fichero `2.solucionConsultas.sql`.

**2.1.** Devuelva el nombre del empleado, la fecha de realización del pedido y el nombre del cliente de todos los pedidos realizados este mes. **(1 puntos)**

**2.2.** Devuelva el nombre, las unidades totales pedidas y el importe total gastado de aquellos clientes que han realizado más de 5 pedidos en el último año. **(2 puntos)**

## 3. Procedimiento. 3,5 puntos

Incluya su solución en el fichero `3.solucionProcedimiento.sql`.

Cree un procedimiento que permita crear un nuevo producto con posibilidad de que sea para regalo. Si el producto está destinado a regalo se creará un pedido con ese producto y costes 0€ para el cliente más antiguo. **(1,5 puntos)**

Asegure que el precio del producto para regalo no debe superar los 50 euros y lance excepción si se da el caso con el siguiente mensaje: **(1 punto)**

No se permite crear un producto para regalo de más de 50€.

Garantice que o bien se realizan todas las operaciones o bien no se realice ninguna. **(1 punto)**

## 4. Trigger. 2 puntos

Incluya su solución en el fichero `4.solucionTrigger.sql`.

Cree un trigger llamado `t_limitar_importe_pedidos_de_menores` que impida que, a partir de ahora, los pedidos realizados por menores superen los 500€.

```
CREATE OR REPLACE TABLE pagos(
    id INT AUTO_INCREMENT PRIMARY KEY,
    pedidoId INT NOT NULL,
    fecha DATE NOT NULL,
    cantidad DECIMAL(10,2) NOT NULL,
    revisado BOOLEAN NOT NULL DEFAULT FALSE,
    CONSTRAINT RN CHECK(cantidad >= 0),
    CONSTRAINT RN_1 UNIQUE(pedidoId),
    FOREIGN KEY (pedidoId) REFERENCES pedidos (id)
);
```

```
-- Ejercicio 2
```

```
SELECT
    ue.nombre AS nombreEmpleado,
    p.fechaRealizacion,
    uc.nombre AS nombreCliente
FROM Pedidos p
JOIN Empleados e ON p.empleadoId = e.id
JOIN Usuarios ue ON e.usuarioId = ue.id -- Usuario del empleado
JOIN Clientes c ON p.clienteId = c.id
JOIN Usuarios uc ON c.usuarioId = uc.id -- Usuario del cliente
WHERE YEAR(p.fechaRealizacion) = YEAR(CURDATE())
    AND MONTH(p.fechaRealizacion) = MONTH(CURDATE());
```

```
SELECT u.nombre AS nombre, SUM(lp.unidades) AS unidades,
SUM(lp.unidades * lp.precio) AS importe
FROM usuarios u
JOIN clientes c ON c.usuarioId = u.id
JOIN pedidos p ON p.clienteId = c.id
JOIN lineaspedido lp ON lp.pedidoId = p.id
WHERE (
    SELECT COUNT(*)
    FROM pedidos p1
    WHERE p1.clienteId = c.id AND TIMESTAMPDIFF(YEAR, p.fechaRealizacion,
CURDATE()) <= 1
) > 5
GROUP BY c.id;
```

```

DELIMITER //
CREATE PROCEDURE crearProducto(
    IN p_nombre VARCHAR(255),
    IN p_descripcion VARCHAR(255),
    IN p_precio DECIMAL(10,2),
    IN p_tipoProducto VARCHAR(255),
    IN p_esRegalo BOOLEAN)
BEGIN
    DECLARE clienteMasAntiguo INT;
    DECLARE direccionClienteMasAntiguo VARCHAR(255);
    DECLARE nuevoProductoId INT;
    DECLARE nuevoPedidoId INT;
    DECLARE clienteIdAntiguo INT;

    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        ROLLBACK;
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Error al registrar
producto y regalos';
    END;

    START TRANSACTION;

    INSERT INTO productos(nombre, descripcion, precio, tipoProductoId)
    VALUES(p_nombre, p_descripcion, p_precio, p_tipoProducto);

    SET nuevoProductoId = LAST_INSERT_ID();

    IF p_esRegalo THEN
        IF p_precio > 50 THEN
            ROLLBACK;
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'No se permite crear un producto para
regalo de más de 50€';
        END IF;

        SET clienteMasAntiguo = (SELECT * FROM clientes c WHERE c.id =
MIN(c.id));
        SET direccionClienteMasAntiguo = (SELECT c.direccionEnvio FROM
clientes c WHERE c.id = clienteMasAntiguo.id);
        SET clienteIdAntiguo = (SELECT c.id FROM clientes c WHERE c.id
= clienteMasAntiguo.id);
        INSERT INTO pedidos(fechaRealizacion, direccionEntrega,
clienteId)
        VALUES(CURDATE(),direccionClienteMasAntiguo, clienteIdAntiguo);

        SET nuevoPedidoId = LAST_INSERT_ID();
        INSERT INTO lineaspedido(pedidoId, productoId, unidades,
precio)
        VALUES(nuevoPedidoId, nuevoProductoId, 1, 0);
    END IF;
    COMMIT;
END //
DELIMITER ;

```

```
-- Ej 4
DELIMITER //
CREATE OR REPLACE TRIGGER t_limitar_importe_pedidos_de_menores
BEFORE INSERT ON lineaspedido
FOR EACH ROW
BEGIN
    DECLARE importeCliente DECIMAL(10,2);
    DECLARE edadCliente INT;
```

```
        SET importeCliente = (SELECT SUM (lp.unidades * lp.precio) FROM
lineaspedido lp WHERE lp.pedidoId = NEW.pedidoId);
        SET edadCliente = (SELECT TIMESTAMPDIFF(YEAR, c.fechaNacimiento,
CURDATE()) FROM clientes c JOIN pedidos p ON p.clienteId = c.id WHERE
p.id = NEW.pedidoId);

        IF edadCliente < 18 AND importeCliente > 500 THEN
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Los pedidos realizados por menores no
pueden superar los 500€';
        END IF;
END //
DELIMITER ;
```