

# EJERCICIOS-SQL-AVANZADOS-RESUELT...



**Ser\_el\_Gito**



**Introducción a la Ingeniería del Software y los Sistemas de Información I**



**2º Grado en Ingeniería Informática - Ingeniería de Computadores**



**Escuela Técnica Superior de Ingeniería Informática  
Universidad de Sevilla**

Formamos  
**talento** para un futuro  
**Sostenible**



MÁSTER EN

**Big Data &  
Business Analytics**

[saber más](#)

**EOI** Escuela de  
organización  
industrial

PROMO VERANO  
**40€**  
de descuento  
para las 10 primeras  
reservas de cada nivel

# ¡CONQUISTA LA PLAYA CON TRES SABORES!

www.couckesacademy.es



**Coucke's  
Academy**  
BY SARAH COUCKE, TEACHING SINCE 2005



**Nervión**  
Avd. San Francisco Javier 24.  
Planta Baja, Módulo 12C  
**954 65 98 99 - 605 54 50 19**  
nervion@couckesacademy.es

```
/* EJERCICIOS BD PEDIDOS - SQL AVANZADO
TABLAS AND TRIGGERS
BY SER EL GITO :) */

-- Creación de La Base de Datos.

DROP DATABASE IF EXISTS pedidos;
CREATE DATABASE pedidos;
USE pedidos;

-- Creación de Las TABLAS.

-- Usuarios:

CREATE OR REPLACE TABLE Users(
  userId      INT NOT NULL AUTO_INCREMENT,
  `name`      VARCHAR(63) NOT NULL,          -- RN-1a. La variable "name" se escribe así porque es una palabra clave.
  province    VARCHAR(63),
  startDate   DATE NOT NULL,                -- RN-1a
  PRIMARY KEY(userId),
  CONSTRAINT RN_1b_nameUnique UNIQUE(NAME)  -- RN-1b
);
-- Productos:

CREATE OR REPLACE TABLE Products(
  productId    INT NOT NULL AUTO_INCREMENT,
  description   VARCHAR(128) NOT NULL,
  price        DECIMAL(6,2) NOT NULL,       -- RN-2a
  stock        INT,
  PRIMARY KEY(productId),
  CONSTRAINT RN_2b_price_range CHECK (price >= 0), -- RN-2b
  CONSTRAINT RN_2c_positive_stock CHECK (stock >= 0) -- RN-2c
);

-- Pedidos:

CREATE OR REPLACE TABLE Orders(
  orderId      INT NOT NULL AUTO_INCREMENT,
  userId       INT NOT NULL,
  productId    INT NOT NULL,
  amount       INT DEFAULT(1),              -- RN-3a
  purchaseDate DATE DEFAULT(NOW()),         -- RN-3a
  PRIMARY KEY(orderId),
  FOREIGN KEY(userId) REFERENCES Users(userId),
  FOREIGN KEY(productId) REFERENCES Products(productId),
  CONSTRAINT RN_3b_amount_range CHECK (amount BETWEEN 0 AND 10), -- RN-3b
  CONSTRAINT RN_3c_month_Not_August CHECK ( MONTH(purchaseDate)<>8 ) -- RN-3c
);

-- Creación de Los TRIGGERS.

-- RN-4: para poder implementar un nuevo pedido,
-- es necesario que haya stock del mismo.

DELIMITER //

CREATE OR REPLACE TRIGGER RN_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 =
      'RN-4: No se puede hacer un pedido de más unidades de las disponibles';
  ELSE
    UPDATE Products SET stock=stock-new.amount
    WHERE productId=new.productId;
  END IF;
END //

DELIMITER ;
```



APÚNTATE

UNIVERSITY OF  
CAMBRIDGE

TRINITY  
COLLEGE LONDON



IELTS

WUOLAH

```

-- RN-5: Un usuario puede hacer como máximo tres pedidos diarios.

DELIMITER //

CREATE OR REPLACE TRIGGER RN_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 =
            'RN-5: No se pueden hacer más de 3 pedidos por día';
    END IF;
END //

DELIMITER ;

/* EJERCICIOS BD PEDIDOS - SQL AVANZADO
   TABLAS AND TRIGGERS
   BY SER EL GITO :) */

-- Usamos La Base de Datos creada anteriormente.

USE pedidos;

-- La base de datos se rellena mediante un procedimiento Llamado pInsertData.

DELIMITER //
CREATE OR REPLACE PROCEDURE
pInsertData()
BEGIN
    -- Limpia tablas y reinicia contadores de claves primarias.
    SET FOREIGN_KEY_CHECKS = 0; -- Desactiva la verificación de integridad referencial
    TRUNCATE Orders;
    TRUNCATE Products;
    TRUNCATE Users;
    SET FOREIGN_KEY_CHECKS = 1; -- Activa la verificación de integridad referencial

    -- Datos de Usuarios:
    INSERT INTO Users(name, province, startDate) VALUES
        ('David Ruiz', 'Sevilla', '2018-05-18'),
        ('Marta López', 'Málaga', '2018-06-12'),
        ('Raquel Lobato', 'Granada', '2018-12-01'),
        ('Antonio Gómez', 'Sevilla', '2018-03-11'),
        ('Inma Hernández', 'Málaga', '2018-04-12'),
        ('Jimena Martín', 'Granada', '2018-05-13'),
        ('Carlos Rivero', 'Huelva', '2018-09-07'),
        ('Carlos Arévalo', 'Málaga', '2018-09-07')
    ;

    -- Datos de Productos:
    INSERT INTO products(description, price, stock) VALUES
        ('Mi Band 3', 19.90, 50),
        ('Mi Band 4', 29.90, 20),
        ('Pulsera compatible con Mi Band 3 y 4', 9.90, 150),
        ('Mi Scooter', 349.90, 25),
        ('Rueda trasera de respuesto Mi Scooter', 19.90, 50),
        ('Rueda delantera de respuesto Mi Scooter', 59.90, 50);

    -- Datos de Pedidos:
    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 ;

```

# EXAMEN SORPRESA

¿Qué te pide la Cuenta NoCuenta?

- ☒ a) Nada
- ☒ b) Nada de nada

¿Qué comisiones\* tiene?

- ☒ a) 0 %
- ☒ b) 0,000000000 %

¿Cuál es su permanencia?

- ☒ a) Ninguna
- ☒ b) Si ya sabes la respuesta pa qué preguntas

\* Si has acertado todas las respuestas, esta cuenta es para ti

¡Me apunto!

\* TIN 0 % y TAE 0 %.



do your thing

## Introducción a la Ingeniería...



**Comparte estos flyers en tu clase y consigue más dinero y recompensas**



**Banco de apuntes de la**

**WUOLAH**

**1**

Imprime esta hoja

**2**

Recorta por la mitad

**3**

Coloca en un lugar visible para que tus compis puedan escanar y acceder a apuntes

**4**

Llévate dinero por cada descarga de los documentos descargados a través de tu QR





```

-- Ejecución del procedimiento.

CALL pInsertData();

/* EJERCICIOS BD PEDIDOS - SQL AVANZADO
   TABLAS AND TRIGGERS
   BY SER EL GITO :) */

-- Usamos la Base de Datos creada anteriormente.

USE pedidos;

-- Implementación de Las FUNCIONES.

-- RF-1: Importe total de pedidos de un usuario.

DELIMITER //

CREATE OR REPLACE FUNCTION fSpentMoneyUser(usId INT) RETURNS DOUBLE
BEGIN
    RETURN(
        SELECT SUM(price*amount)
        FROM users
        NATURAL JOIN orders
        NATURAL JOIN products
        WHERE userId = usId
    );
END //

DELIMITER ;

-- RF-2: Stock de un producto.

delimiter //

CREATE OR REPLACE FUNCTION fGetStock(proId INT) RETURNS INT
BEGIN
    RETURN (
        SELECT stock
        FROM products
        WHERE productId = proId
    );
END //

delimiter ;

-- RF-3: Número de unidades vendidas para un producto.

CREATE OR REPLACE FUNCTION fGetSoldUnits(proId INT) RETURNS INT
BEGIN
    RETURN (
        SELECT SUM(amount)
        FROM orders
        NATURAL JOIN products
        WHERE productId = proId
    );
END //

delimiter ;

-- RF-4: Importe total de ventas entre dos fechas.

delimiter //

CREATE OR REPLACE FUNCTION fPurchaseBetweenDates(
DATE1 DATE, DATE2 DATE) RETURNS INT
BEGIN
    RETURN (
        SELECT SUM(price*amount)
        FROM orders
        NATURAL JOIN products
        WHERE ((purchaseDate>=DATE1) && (purchaseDate<=DATE2))
    );
END //

delimiter ;

-- RF-5: ID del producto más vendido en un año.

delimiter //

```

PROMO VERANO  
**40€**  
de descuento  
para las 10 primeras  
reservas de cada nivel

## ¡CONQUISTA LA PLAYA CON TRES SABORES!



**Coucke's  
Academy**  
BY SARAH COUCKE, TEACHING SINCE 2005



**Nervión**  
Avd. San Francisco Javier 24,  
Planta Baja, Módulo 12C



**954 65 98 99 - 605 54 50 19**  
nervion@couckesacademy.es

[www.couckesacademy.es](http://www.couckesacademy.es)

```
CREATE OR REPLACE FUNCTION fBestSeller(ye YEAR) RETURNS INT
```

```
BEGIN
```

```
    RETURN (
        SELECT productId
        FROM orders
        WHERE YEAR(purchaseDate) = ye
        GROUP BY productId
        ORDER BY SUM(amount)
        DESC LIMIT 1
    );
```

```
END //
```

```
delimiter ;
```

```
-- Implementación de Los PROCEDIMIENTOS.
```

```
-- RF-6: Crear pedido con la fecha actual, cantidad y fecha suministrada.
```

```
-- Si cantidad o fecha son nulas, se deben aplicar los valores por defecto.
```

```
delimiter //
```

```
CREATE OR REPLACE PROCEDURE pInsertOrder(uId INT, pId INT, am INT, pDate DATE)
```

```
BEGIN
```

```
    INSERT INTO orders(userId, productId, amount, purchaseDate)
    SELECT uId, pId,
    IFNULL(am, 1),
    IFNULL(pDate, CURRENT_DATE);
```

```
END //
```

```
delimiter ;
```

```
-- RF-7: Modificar todos los precios de productos suministrando un porcentaje.
```

```
delimiter //
```

```
CREATE OR REPLACE PROCEDURE pChangePrices(fraction DOUBLE)
```

```
BEGIN
```

```
    UPDATE products
    SET price = price + price * fraction;
```

```
END//
```

```
delimiter ;
```

```
-- RF-8: Crear vista con pedidos realizados por un usuario (<Nombre>) en un <Año>,  
-- reemplazando espacios del nombre para evitar error SQL.
```

```
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,','  
                             );
```

```
-- Usamos EXECUTE IMMEDIATE CONCAT porque no se puede  
-- crear una vista dentro de un procedimiento.
```

```
END //
```

```
DELIMITER ;
```

```
-- RF-9: Crea la vista anterior para todos los usuarios que en un <Año> han realizado mas de <N> pedidos.
```

```
delimiter //
```

```
CREATE OR REPLACE PROCEDURE pCreateOrdersViews(minOrders INT, ordersYear YEAR)
```

```
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;
```



APÚNTATE

UNIVERSITY OF  
CAMBRIDGE

TRINITY  
COLLEGE LONDON



IELTS

WUOLAH

```

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);      -- Usamos La función anterior para crear La vista.
END LOOP;
CLOSE usersWithOrders;

END //

delimiter ;

-- Llamamos a Las funciones y procedimientos.

SELECT fSpentMoneyUser(1);
SELECT fSpentMoneyUser(2);

SELECT fGetStock(1);
SELECT fGetStock(2);

SELECT fGetSoldUnits(1);
SELECT fGetSoldUnits(2);

SELECT fPurchaseBetweenDates('2019-6-1', '2020-12-31');

SELECT fBestSeller('2019');

CALL pInsertOrder(2, 2, NULL, NULL);

CALL pChangePrices(1.2);

CALL pCreateOrdersView('David Ruiz', '2019');

CALL pCreateOrdersViews(2, 2019);

```





PROMO VERANO

**40€**  
de descuento

para las 10 primeras  
reservas de cada nivel

**¡CONQUISTA LA PLAYA  
CON TRES SABORES!**

[www.couckesacademy.es](http://www.couckesacademy.es)



**B1**



**B2**



**C1**



**Coucke's  
Academy**

BY SARAH COUCKE, TEACHING SINCE 2005



**Nervión**

Avd. San Francisco Javier 24.  
Planta Baja, Módulo 12C



**954 65 98 99 - 605 54 50 19**

[nervion@couckesacademy.es](mailto:nervion@couckesacademy.es)



**APÚNTATE**

UNIVERSITY OF  
CAMBRIDGE

TRINITY  
COLLEGE LONDON

Aptis  
Aptis

**IELTS**

Ser el Gito

**WUOLAH**