Informe de Consultas en Proyecto de Base de Datos

Presentado por:

Michael Alejandro Papamija Pantoja

Docente:

Brayan Arcos

Materia:

Desarrollo de Bases de Datos

Instituto Tecnológico del Putumayo

Mocoa – Putumayo

2024

Contenido

[1. INTRODUCCION 5](#_Toc184390006)

[2. Metodología 6](#_Toc184390007)

[2.1 Herramientas Utilizadas 6](#_Toc184390008)

[2.2 Procedimientos 7](#_Toc184390009)

[3. Consultas SQL 7](#_Toc184390010)

[3.1. Consultas Complejas 8](#_Toc184390011)

[3.1.1 LIKE 8](#_Toc184390012)

[3.1.2. ORDER BY 9](#_Toc184390013)

[**Explicación** 9](#_Toc184390014)

[3.1.3. LIKE, ORDER BY, ASC 10](#_Toc184390015)

[**Explicación** 10](#_Toc184390016)

[3.2. Consultas Avanzadas 11](#_Toc184390017)

[3.2.1. NOMBRE COMPLETO 11](#_Toc184390018)

[3.2.2. ROUND 11](#_Toc184390019)

[**Explicación** 11](#_Toc184390020)

[3.2.3. SALARY – JOB POSITION 12](#_Toc184390021)

[**Explicación** 12](#_Toc184390022)

[3.2.4. CANTIDAD EMPLEADOS POR PUESTO 13](#_Toc184390023)

[**Explicación** 13](#_Toc184390024)

[3.2.5. MAX – MIN PRODUCTS BY CATEGORY 13](#_Toc184390025)

[**Explicación** 14](#_Toc184390026)

[3.3. Consultas INNER JOIN 14](#_Toc184390027)

[3.3.1. Lister Product’s Category 15](#_Toc184390028)

[**Explicación** 15](#_Toc184390029)

[3.3.2. Productos más caros por categoría 15](#_Toc184390030)

[**Explicación** 16](#_Toc184390031)

[3.3.3. Nombre corto de Rol Empleado 16](#_Toc184390032)

[**Explicación** 16](#_Toc184390033)

[3.3.4. Tipo de documento de cada persona 17](#_Toc184390034)

[**Explicación** 17](#_Toc184390035)

[**Resultado** 17](#_Toc184390036)

[3.3.5. Methods de Pago mas Populares 18](#_Toc184390037)

[**Explicación** 18](#_Toc184390038)

[**Resultado** 18](#_Toc184390039)

[3.3.6. Empleados con major Salario 18](#_Toc184390040)

[**Explicación** 19](#_Toc184390041)

[**Resultado** 19](#_Toc184390042)

[3.3.7. Proveedores con mas productos 19](#_Toc184390043)

[**Explicación** 19](#_Toc184390044)

[**Resultado** 19](#_Toc184390045)

[3.3.8. Empleado con salario alto por posición 20](#_Toc184390046)

[**Explicación** 20](#_Toc184390047)

[**Resultado** 20](#_Toc184390048)

[3.4. Consultas RIGTH JOIN 20](#_Toc184390049)

[3.4.1. Obtener Roles y Usuarios 21](#_Toc184390050)

[3.4.2. Métodos de pago 22](#_Toc184390051)

[3.5. Consultas LEFT JOIN 22](#_Toc184390052)

[3.5.1. Numeros de telefono 23](#_Toc184390053)

[**Explicación** 23](#_Toc184390054)

[3.5.2. Productos con detalles 23](#_Toc184390055)

[**Explicación** 23](#_Toc184390056)

[**Resultado** 23](#_Toc184390057)

[3.6. Consultas CROSS JOIN 24](#_Toc184390058)

[3.6.1. Metodo de pago y proveedor 24](#_Toc184390059)

[**Explicación** 24](#_Toc184390060)

[**Resultado** 24](#_Toc184390061)

[3.7. Consultas ALL-ANY 24](#_Toc184390062)

[3.7.1. Productos mas caros (ALL) 25](#_Toc184390063)

[**Explicación** 25](#_Toc184390064)

[**Resultado** 25](#_Toc184390065)

[3.7.2. Factutas con monto mayor que pago individual (ANY) 25](#_Toc184390066)

[**Explicación** 26](#_Toc184390067)

[**Resultado** 26](#_Toc184390068)

[3.8. Consultas EXISTS 26](#_Toc184390069)

[3.8.1. Tiendas con al menos un producto en stock 27](#_Toc184390070)

[**Explicación** 27](#_Toc184390071)

[**Resultado** 27](#_Toc184390072)

[3.8.2. Facturas con al menos una relación con invoiceDetails 28](#_Toc184390073)

[**Explicación** 28](#_Toc184390074)

[**Resultado** 28](#_Toc184390075)

[4. Subconsultas 29](#_Toc184390076)

[4.1. Subconsultas SELECT 29](#_Toc184390077)

[4.2. Subconsultas FROM 29](#_Toc184390078)

[4.3. Subconsultas WHERE 29](#_Toc184390079)

[4.4. Subconsultas IN 29](#_Toc184390080)

[4.5. Subconsultas EXISTS 29](#_Toc184390081)

[4.6. Subconsultas ALL 29](#_Toc184390082)

[5. Vistas 29](#_Toc184390083)

[5.1. Pagos realizados por clients 29](#_Toc184390084)

[5.2. Resumen de facturas emitidas 30](#_Toc184390085)

[6. Procedimientos Almacenados 30](#_Toc184390086)

[6.1. Tienda por ID 30](#_Toc184390087)

[6.2. Productos por categoria. 31](#_Toc184390088)

[7. Triggers 31](#_Toc184390089)

[7.1. Before Insert payment 31](#_Toc184390090)

[7.2. BeforeUpdate product 32](#_Toc184390091)

[8. Indices 32](#_Toc184390092)

[8.1. Indices Agrupados 32](#_Toc184390093)

[8.1.1. Users 32](#_Toc184390094)

[8.1.2. Invoices 32](#_Toc184390095)

[8.2. Indices No Agrupados 33](#_Toc184390096)

[8.2.1. Roles por nombre 33](#_Toc184390097)

[8.2.2. Producto por codigo 33](#_Toc184390098)

[8.3. Indices Unicos 33](#_Toc184390099)

[8.3.1. Email de ususario 33](#_Toc184390100)

[8.3.2. Codigo de product 33](#_Toc184390101)

[8.4. Indices Compuestos 33](#_Toc184390102)

[8.4.1. Facturas por cliente 34](#_Toc184390103)

# INTRODUCCION

Este informe se ha elaborado con el objetivo de documentar y analizar de manera sistemática las consultas realizadas en el proyecto de bases de datos relacionales, facilitando así su comprensión y seguimiento. Las consultas se llevaron a cabo utilizando MySQL Workbench, una herramienta esencial para la gestión y manipulación de grandes volúmenes de información. Esta plataforma no solo permite ejecutar consultas SQL, sino que también proporciona un entorno visual que simplifica la interacción con la base de datos.

En el presente informe, se abordan los tipos de consultas que son consideradas fundamentales para el manejo eficiente de la información dentro de la base de datos. Estas consultas son cruciales para extraer datos relevantes y realizar análisis significativos. A través de ejemplos prácticos, se ilustrarán las diferentes operaciones que se pueden llevar a cabo, desde la selección y actualización de registros hasta la creación de nuevas tablas y la definición de relaciones entre ellas.

Además, se explorará la importancia del lenguaje SQL en el contexto de las bases de datos relacionales, destacando cómo este lenguaje permite a los usuarios interactuar con los datos almacenados, optimizando así el rendimiento y la integridad de la información. Este análisis no solo servirá como un recurso educativo para quienes deseen profundizar en el uso de SQL, sino que también proporcionará una referencia útil para el desarrollo futuro del proyecto.

# Metodología

## 2.1 Herramientas Utilizadas

Para el desarrollo de este informe se emplearon las siguientes herramientas:

* **MySQL** como sistema de gestión de bases de datos (DBMS).
* **MySQL Workbench** para la creación de la base de datos, definición de tablas y relaciones, y ejecución de consultas SQL.
* **Enlace de GitHub:** [**https://github.com/dev-Alejo24/MySQL-Michael\_Papamija-.git**](https://github.com/dev-Alejo24/MySQL-Michael_Papamija-.git)

## 2.2 Procedimientos

Los procedimientos seguidos para llevar a cabo el análisis fueron los siguientes:

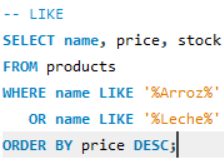
1. Creación del esquema de la base de datos en MySQL Workbench, incluyendo la definición de tablas, claves primarias, foráneas, y las relaciones entre las entidades.
2. Ejecución de consultas SQL en MySQL Workbench para extraer información relevante y validar el correcto funcionamiento de la base de datos.

# Consultas SQL

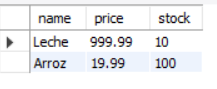
Se realizaron diversas consultas SQL para obtener datos relevantes. A continuación, se describen algunas de las consultas más importantes ejecutadas.

## Consultas Complejas

### 3.1.1 LIKE



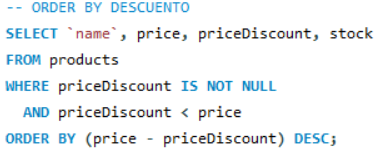
**Resultado**

****

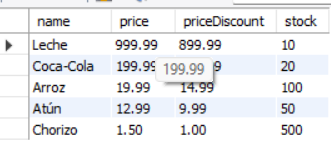
* **SELECT**: Esta cláusula selecciona las columnas name, price y stock de la tabla products.
* **FROM products**: Indica que los datos se extraen de la tabla llamada products.
* **WHERE**: Esta cláusula filtra los resultados según ciertas condiciones:
  + name LIKE '%Arroz%': Busca productos cuyo nombre contenga la palabra "Arroz" en cualquier parte del texto. El símbolo % actúa como un comodín que representa cualquier secuencia de caracteres (incluyendo ninguna).
  + OR name LIKE '%Leche%': También busca productos cuyo nombre contenga la palabra "Leche". La condición OR significa que se seleccionarán productos que cumplan al menos una de estas condiciones.
* **ORDER BY price DESC**: Los resultados se ordenan en orden descendente (DESC) según el precio, es decir, los productos más caros aparecerán primero.

Esta consulta devolverá una lista de productos que contienen "Arroz" o "Leche" en su nombre, mostrando su nombre, precio y stock, ordenados desde el más caro al más barato.

### 3.1.2. ORDER BY



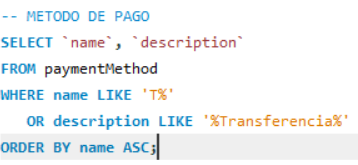
RESULTADO:



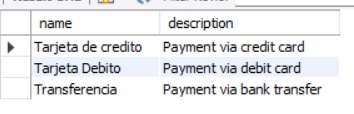
## **Explicación**

* **SELECT**: Esta cláusula selecciona las columnas name, price, priceDiscount y stock de la tabla products.
* **FROM products**: Indica que los datos provienen de la tabla llamada products.
* **WHERE**: Filtra los resultados con las siguientes condiciones:
  + priceDiscount IS NOT NULL: Asegura que solo se incluyan productos que tienen un precio de descuento definido (no nulo).
  + AND priceDiscount < price: Filtra para incluir solo aquellos productos donde el precio de descuento es menor que el precio original, lo que implica que hay un descuento aplicable.
* **ORDER BY (price - priceDiscount) DESC**: Los resultados se ordenan en orden descendente según el valor del descuento (calculado como la diferencia entre el precio original y el precio con descuento). Esto significa que los productos con mayor descuento aparecerán primero.

### 3.1.3. LIKE, ORDER BY, ASC



RESULTADO



## **Explicación**

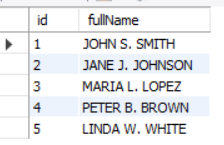
* **SELECT**: Esta cláusula selecciona las columnas name y description de la tabla paymentMethod.
* **FROM paymentMethod**: Indica que los datos provienen de la tabla llamada paymentMethod.
* **WHERE**: Filtra los resultados con las siguientes condiciones:
  + name LIKE 'T%': Busca métodos de pago cuyo nombre comience con la letra "T". El símbolo % indica que puede haber cualquier secuencia de caracteres después de "T".
  + OR description LIKE '%Transferencia%': También incluye métodos de pago cuya descripción contenga la palabra "Transferencia" en cualquier parte del texto.
* **ORDER BY name ASC**: Los resultados se ordenan en orden ascendente (ASC) según el nombre del método de pago.

## Consultas Avanzadas

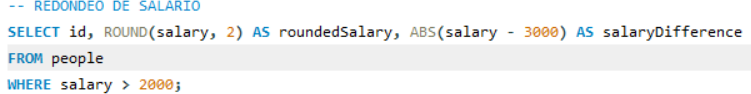
Las siguientes ocsultas son un poco mas avanzadas ya que se hace uso de funciones pertenecientes a SQL en el filtrado de datos.

### NOMBRE COMPLETO

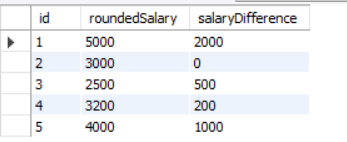
RESULTADO



### ROUND



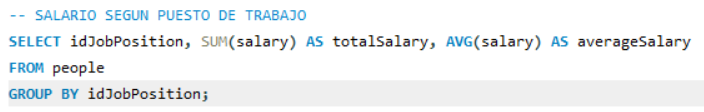
RESULTADO



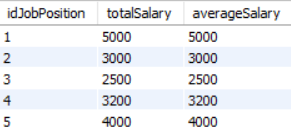
## **Explicación**

* **SELECT**: Esta cláusula selecciona el id de la tabla people y crea una nueva columna llamada fullName.
* **CONCAT**: Esta función combina varias cadenas de texto:
  + UPPER(firstName): Convierte el nombre (firstName) a mayúsculas.
  + UPPER(SUBSTRING(lastNamePaternal, 1, 1)): Toma la primera letra del apellido paterno (lastNamePaternal), la convierte a mayúscula y le añade un punto ('.').
  + UPPER(lastNamePaternal): Convierte el apellido paterno completo a mayúsculas.
* **AS fullName**: Renombra la columna resultante como fullName.

### SALARY – JOB POSITION



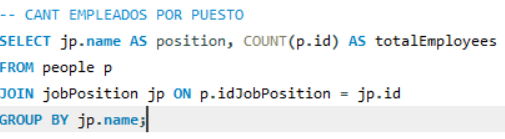
RESULTADO



## **Explicación**

* **SELECT**: Esta cláusula selecciona el identificador del puesto de trabajo (idJobPosition), suma los salarios y calcula el promedio de salarios.
* **SUM(salary)**: Calcula la suma total de los salarios para cada puesto de trabajo y lo renombra como totalSalary.
* **AVG(salary)**: Calcula el salario promedio para cada puesto de trabajo y lo renombra como averageSalary.
* **GROUP BY idJobPosition**: Agrupa los resultados por el identificador del puesto de trabajo. Esto significa que se generará una fila por cada puesto de trabajo único en la tabla.

### CANTIDAD EMPLEADOS POR PUESTO



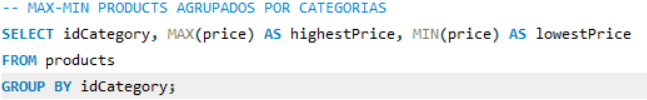
RESULTADO



## **Explicación**

* **SELECT**: Esta cláusula selecciona el nombre del puesto (jp.name) y cuenta la cantidad total de empleados (COUNT(p.id)).
* **JOIN jobPosition jp ON p.idJobPosition = jp.id**: Realiza una unión entre las tablas people (alias p) y jobPosition (alias jp) utilizando la relación entre idJobPosition en people y id en jobPosition.
* **COUNT(p.id)**: Cuenta cuántos empleados hay para cada puesto de trabajo.
* **GROUP BY jp.name**: Agrupa los resultados por el nombre del puesto. Esto genera una fila por cada tipo de puesto en la tabla

### MAX – MIN PRODUCTS BY CATEGORY



RESULTADO:



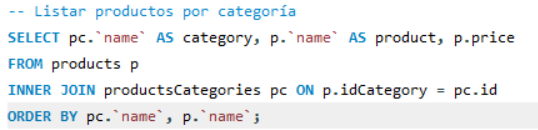
## **Explicación**

* **SELECT**: Esta cláusula selecciona el identificador de categoría (idCategory), así como el precio máximo y mínimo.
* **MAX(price)**: Calcula el precio más alto dentro de cada categoría y lo renombra como highestPrice.
* **MIN(price)**: Calcula el precio más bajo dentro de cada categoría y lo renombra como lowestPrice.
* **GROUP BY idCategory**: Agrupa los resultados por categoría. Esto significa que se generará una fila por cada categoría única en la tabla.

## Consultas INNER JOIN

En las siguientes consultas se usa lo aprendido anteriormente, esta vez se implementa o agrega el foltrado de información con INNER JOIN en las consultas SQL.

### Lister Product’s Category



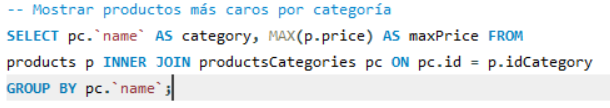
RESULTADO



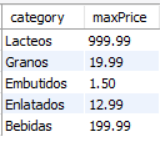
## **Explicación**

* **SELECT**: Selecciona el nombre de la categoría (pc.name), el nombre del producto (p.name) y el precio del producto (p.price).
* **FROM products p**: Indica que la tabla principal es products, usando p como alias.
* **INNER JOIN productsCategories pc ON p.idCategory = pc.id**: Realiza un INNER JOIN con la tabla productsCategories, donde se unen por el campo idCategory en products y el campo id en productsCategories.
* **ORDER BY pc.name, p.name**: Ordena los resultados primero por el nombre de la categoría y luego por el nombre del producto.

### Productos más caros por categoría



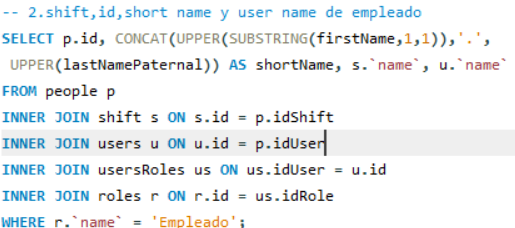
RESULTADO



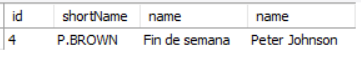
## **Explicación**

* **SELECT**: Selecciona el nombre de la categoría (pc.name) y el precio máximo de los productos (MAX(p.price)).
* **FROM products p**: Indica que la tabla principal es products.
* **INNER JOIN productsCategories pc ON pc.id = p.idCategory**: Une las tablas products y productsCategories.
* **GROUP BY pc.name**: Agrupa los resultados por nombre de categoría.

### Nombre corto de Rol Empleado



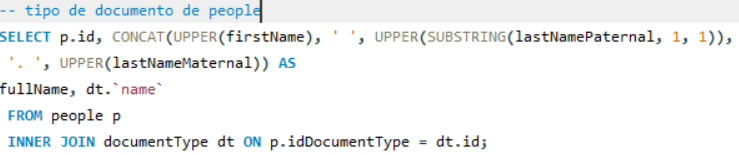
RESULTADO



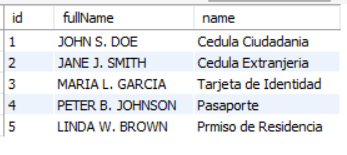
## **Explicación**

* **SELECT**: Crea un nombre corto combinando la inicial del primer nombre con el apellido paterno y selecciona el salario.
* **FROM people p**: Indica que la tabla principal es people.
* **INNER JOIN users u ON p.idUser = u.id**: Une la tabla people con users.
* **INNER JOIN usersRoles us ON us.idUser = u.id**: Une la tabla users con usersRoles.
* **INNER JOIN roles r ON r.id = us.idRole**: Une la tabla usersRoles con roles.
* **WHERE ... AND r.name = 'Empleado'**: Filtra para incluir solo empleados cuyo salario es mayor que el promedio.

### Tipo de documento de cada persona



RESULTADO



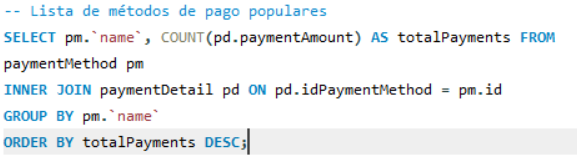
## **Explicación**

* **SELECT**: Selecciona el ID de la persona (p.id), su nombre completo y el tipo de documento.
* **FROM people p**: Indica que la tabla principal es people.
* **INNER JOIN documentType dt ON p.idDocumentType = dt.id**: Une las tablas people y documentType.

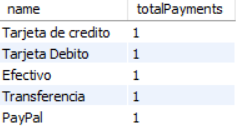
## **Resultado**

Devuelve una lista de personas junto con su nombre completo y tipo de documento.

### Methods de Pago mas Populares



**RESULTADO**



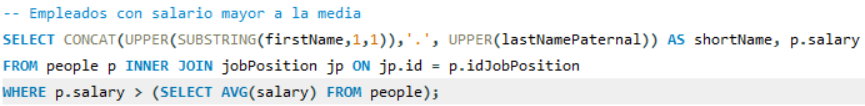
## **Explicación**

* **SELECT**: Selecciona el nombre del método de pago y cuenta los pagos realizados.
* **FROM paymentMethod pm**: Indica que la tabla principal es paymentMethod.
* **INNER JOIN paymentDetail pd ON pd.idPaymentMethod = pm.id**: Une las tablas basándose en el método de pago.
* **GROUP BY pm.name**: Agrupa los resultados por método de pago.
* **ORDER BY totalPayments DESC**: Ordena los resultados desde el método más popular al menos popular.

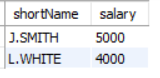
## **Resultado**

Devuelve una lista de métodos de pago junto con la cantidad total de pagos realizados para cada uno.

### Empleados con major Salario



**RESULTADO**



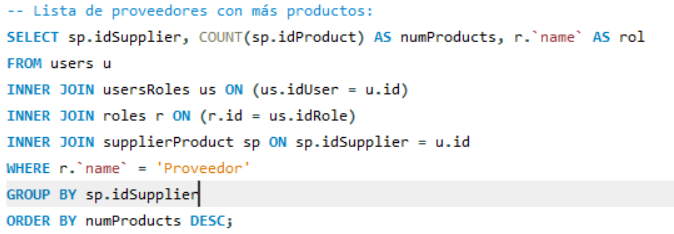
## **Explicación**

Esta consulta es similar a una anterior. Selecciona un nombre corto para empleados cuyo salario es mayor al promedio.

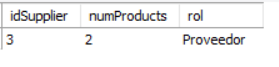
## **Resultado**

Devuelve una lista similar a la anterior pero asegurando que solo se incluyan empleados

### Proveedores con mas productos



**RESULTADO**



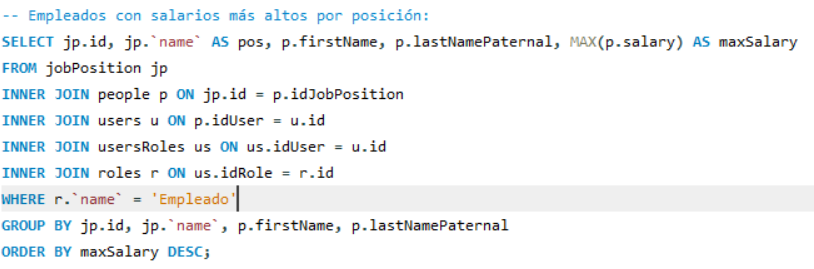
## **Explicación**

* **SELECT**: Selecciona el ID del proveedor y cuenta cuántos productos tiene.
* **FROM users u**: Indica que la tabla principal es users.
* **INNER JOIN ... WHERE ... GROUP BY ... ORDER BY ...**: Realiza varias uniones para filtrar solo proveedores y agrupar por ID del proveedor.

## **Resultado**

Devuelve una lista de proveedores junto con la cantidad total de productos que tienen, ordenada desde aquellos con más productos hasta menos.

### Empleado con salario alto por posición



**RESULTADO**



## **Explicación**

Esta consulta busca obtener los salarios más altos agrupados por puesto:

* Agrupa por puesto y selecciona el salario máximo para cada posición.

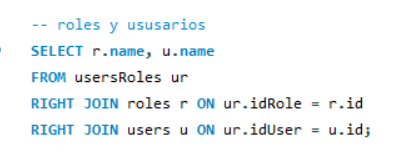
## **Resultado**

Devuelve una lista detallada mostrando cada puesto junto al empleado que tiene el salario más alto en ese puesto. En resumen, estas consultas utilizan INNER JOIN para combinar datos relevantes entre múltiples tablas en una base de datos relacional. Cada consulta está diseñada para extraer información específica sobre productos, empleados o métodos de pago basándose en relaciones definidas entre las tablas.

## Consultas RIGTH JOIN

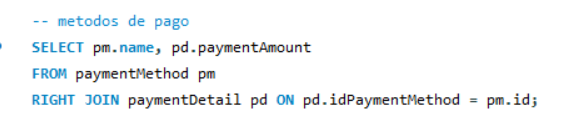
Las consultas **RIGHT JOIN** en SQL son utilizadas para combinar datos de dos o más tablas, priorizando los registros de la tabla de la derecha. Este tipo de unión asegura que todas las filas de la tabla derecha se incluyan en el resultado, incluso si no hay coincidencias en la tabla izquierda.

### Obtener Roles y Usuarios



1. **Tablas Involucradas**:
   * usersRoles (alias ur): Esta tabla probablemente relaciona usuarios con roles.
   * roles (alias r): Contiene información sobre los roles disponibles.
   * users (alias u): Contiene información sobre los usuarios.
2. **RIGHT JOIN**:
   * El primer RIGHT JOIN une la tabla roles con usersRoles. Esto significa que se seleccionarán todos los roles, incluso si no hay usuarios asociados a ellos. Si un rol no tiene usuarios, los campos correspondientes de la tabla usersRoles serán nulos.
   * El segundo RIGHT JOIN une la tabla users con el resultado anterior. Esto asegura que se incluyan todos los usuarios, incluso si no tienen un rol asignado.
3. **Resultado**:
   * La consulta devolverá una lista de nombres de roles y nombres de usuarios. Si un rol no tiene usuarios o un usuario no tiene un rol, se mostrarán valores nulos en las columnas correspondientes.

### Métodos de pago

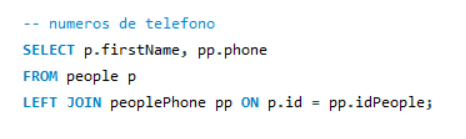


1. **Tablas Involucradas**:
   * paymentMethod (alias pm): Contiene información sobre los métodos de pago disponibles.
   * paymentDetail (alias pd): Contiene detalles sobre los pagos realizados, incluyendo el método de pago utilizado.
2. **RIGHT JOIN**:
   * Aquí, el RIGHT JOIN une la tabla paymentDetail con la tabla paymentMethod. Esto significa que se seleccionarán todos los detalles de pago, incluso si no hay un método de pago asociado. Si un detalle de pago no tiene un método correspondiente, los campos de la tabla paymentMethod serán nulos.
3. **Resultado**:
   * La consulta devolverá una lista de nombres de métodos de pago y sus montos correspondientes. Si un detalle de pago no tiene un método asociado, el nombre del método será nulo.

## Consultas LEFT JOIN

Las consultas **LEFT JOIN** en SQL son utilizadas para combinar datos de dos tablas, priorizando los registros de la tabla izquierda. Este tipo de unión asegura que todas las filas de la tabla izquierda se incluyan en el resultado, incluso si no hay coincidencias en la tabla derecha.

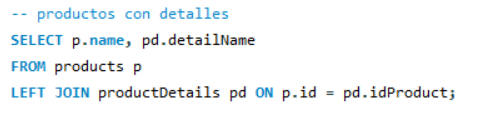
### Numeros de telefono



## **Explicación**

* **SELECT**: Selecciona el primer nombre (p.firstName) y el número de teléfono (pp.phone).
* **FROM people p**: Indica que la tabla principal es people.
* **LEFT JOIN peoplePhone pp ON p.id = pp.idPeople**: Realiza un LEFT JOIN con la tabla peoplePhone, donde se unen por el campo idPeople. Esto permite incluir todos los registros de la tabla people, incluso aquellos que no tienen números de teléfono asociados.

### Productos con detalles



## **Explicación**

* **SELECT**: Selecciona el nombre del producto (p.name) y el nombre del detalle del producto (pd.detailName).
* **FROM products p**: Indica que la tabla principal es products.
* **LEFT JOIN productDetails pd ON p.id = pd.idProduct**: Realiza un LEFT JOIN con la tabla productDetails, donde se unen por el campo idProduct. Esto asegura que todos los productos se incluyan en los resultados, incluso si no tienen detalles asociados.

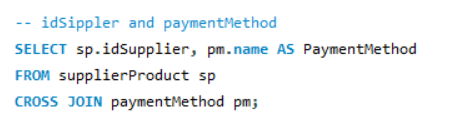
## **Resultado**

Devuelve una lista de productos junto con sus detalles. Si un producto no tiene detalles registrados, aparecerá en la lista con el campo correspondiente al detalle como NULL.

## Consultas CROSS JOIN

El **CROSS JOIN** en SQL es una operación que combina todas las filas de una tabla con todas las filas de otra tabla, generando lo que se conoce como un **producto cartesiano**. Esto significa que si la primera tabla tiene n*n* filas y la segunda tabla tiene m*m* filas, el resultado del CROSS JOIN contendrá n×*m* filas.

### Metodo de pago y proveedor



## **Explicación**

* **SELECT**: Selecciona el ID del proveedor (sp.idSupplier) y el nombre del método de pago (pm.name).
* **FROM supplierProduct sp**: Indica que la tabla principal es supplierProduct, usando sp como alias.
* **CROSS JOIN paymentMethod pm**: Realiza un CROSS JOIN con la tabla paymentMethod, usando pm como alias. Esto también generará un producto cartesiano entre las dos tablas.

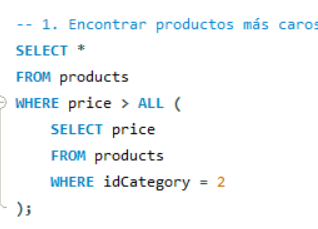
## **Resultado**

El resultado de esta consulta será una lista que muestra todas las combinaciones posibles entre los IDs de los proveedores y los métodos de pago. Si hay, por ejemplo, 4 proveedores y 6 métodos de pago, la consulta devolverá un total de 24 filas (4 x 6), donde cada combinación se muestra en una fila separada.

## Consultas ALL-ANY

### Productos mas caros (ALL)

**ALL**: Se utiliza para comparar un valor con todos los valores devueltos por una subconsulta. La condición se cumple solo si el valor es mayor (o menor) que todos esos valores.



## **Explicación**

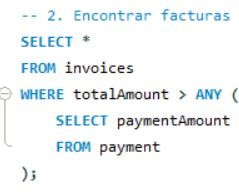
* \*\*SELECT \*\*\*: Selecciona todas las columnas de la tabla products.
* **FROM products**: Indica que los datos provienen de la tabla products.
* **WHERE price > ALL (...)**: Filtra los resultados para incluir solo aquellos productos cuyo precio es mayor que todos los precios devueltos por la subconsulta.
  + **Subconsulta**:
    - **SELECT price FROM products WHERE idCategory = 2**: Esta subconsulta selecciona los precios de todos los productos que pertenecen a la categoría con idCategory = 2.

## **Resultado**

La consulta devolverá todos los productos cuyo precio sea mayor que el precio más alto de todos los productos en la categoría especificada (idCategory = 2). Si no hay productos en esa categoría, la subconsulta no devolverá filas, y el resultado será vacío.

### Factutas con monto mayor que pago individual (ANY)

**ANY**: Se utiliza para comparar un valor con al menos uno de los valores devueltos por una subconsulta. La condición se cumple si el valor es mayor (o menor) que al menos uno de esos valores.



## **Explicación**

* \*\*SELECT \*\*\*: Selecciona todas las columnas de la tabla invoices.
* **FROM invoices**: Indica que los datos provienen de la tabla invoices.
* **WHERE totalAmount > ANY (...)**: Filtra los resultados para incluir solo aquellas facturas cuyo monto total (totalAmount) es mayor que al menos uno de los montos de pago devueltos por la subconsulta.
  + **Subconsulta**:
    - **SELECT paymentAmount FROM payment**: Esta subconsulta selecciona todos los montos de pago realizados en la tabla payment.

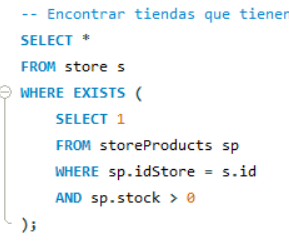
## **Resultado**

La consulta devolverá todas las facturas cuyo monto total sea mayor que al menos uno de los montos de pago registrados en la tabla payment. Si no hay pagos registrados, entonces la subconsulta no devolverá filas, y el resultado será vacío

## Consultas EXISTS

El operador **EXISTS** se utiliza para verificar si una subconsulta devuelve alguna fila. Es particularmente útil cuando se desea comprobar la existencia de registros relacionados sin necesidad de contar o recuperar datos específicos de esas filas. Si la subconsulta encuentra al menos una coincidencia, el resultado es verdadero y se incluyen los registros correspondientes de la consulta principal. Esto puede resultar en consultas más eficientes, especialmente cuando solo se necesita saber si existe alguna relación.

### Tiendas con al menos un producto en stock



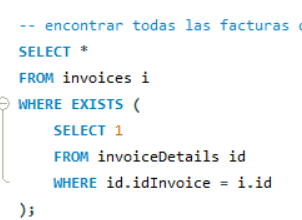
## **Explicación**

* \*\*SELECT \*\*\*: Selecciona todas las columnas de la tabla store.
* **FROM store s**: Indica que los datos provienen de la tabla store, usando s como alias.
* **WHERE EXISTS (...)**: Esta cláusula verifica si existe al menos una fila en el resultado de la subconsulta. Si la subconsulta devuelve al menos una fila, la condición se evalúa como verdadera.
  + **Subconsulta**:
    - **SELECT 1 FROM storeProducts sp**: Selecciona un valor constante (1) de la tabla storeProducts, que se utiliza para verificar la existencia de filas.
    - **WHERE sp.idStore = s.id AND sp.stock > 0**: Filtra los resultados de storeProducts para encontrar aquellos productos que pertenecen a la tienda actual (s.id) y que tienen un stock mayor que cero.

## **Resultado**

La consulta devolverá todas las tiendas que tienen al menos un producto en stock. Si no hay productos en stock para una tienda, esa tienda no aparecerá en los resultados.

### Facturas con al menos una relación con invoiceDetails



);

## **Explicación**

* \*\*SELECT \*\*\*: Selecciona todas las columnas de la tabla store.
* **FROM store s**: Indica que los datos provienen de la tabla store, usando s como alias.
* **WHERE EXISTS (...)**: Esta cláusula verifica si existe al menos una fila en el resultado de la subconsulta. Si la subconsulta devuelve al menos una fila, la condición se evalúa como verdadera.
  + **Subconsulta**:
    - **SELECT 1 FROM storeProducts sp**: Selecciona un valor constante (1) de la tabla storeProducts, que se utiliza para verificar la existencia de filas.
    - **WHERE sp.idStore = s.id AND sp.stock > 0**: Filtra los resultados de storeProducts para encontrar aquellos productos que pertenecen a la tienda actual (s.id) y que tienen un stock mayor que cero.

## **Resultado**

La consulta devolverá todas las tiendas que tienen al menos un producto en stock. Si no hay productos en stock para una tienda, esa tienda no aparecerá en los resultados.

# Subconsultas

## Subconsultas SELECT

## Subconsultas FROM

## Subconsultas WHERE

## Subconsultas IN

## Subconsultas EXISTS

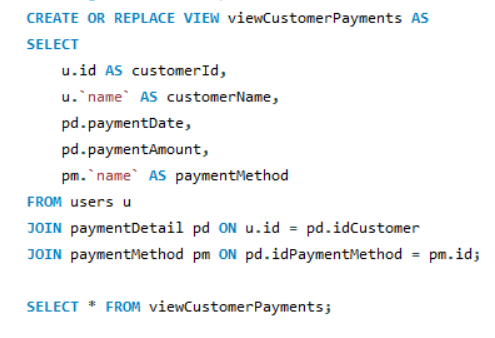
## Subconsultas ALL

# Vistas

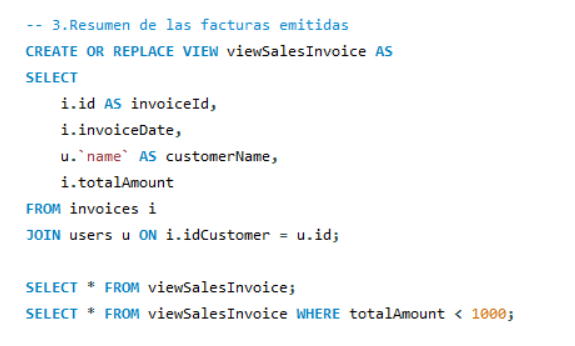
Las **vistas** son tablas virtuales que representan los resultados de una consulta SQL. Se utilizan para simplificar el acceso a datos complejos y pueden ser creadas, modificadas o eliminadas con las sentencias SQL adecuadas

* Video: <https://drive.google.com/file/d/1pxTRK7uNxr_u3RwydDvQmmFSDjtYeh_x/view?usp=sharing>

## Pagos realizados por clients



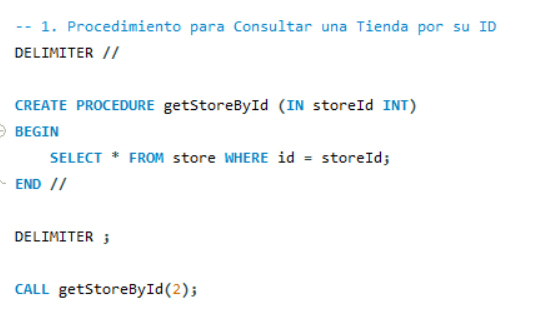
## Resumen de facturas emitidas



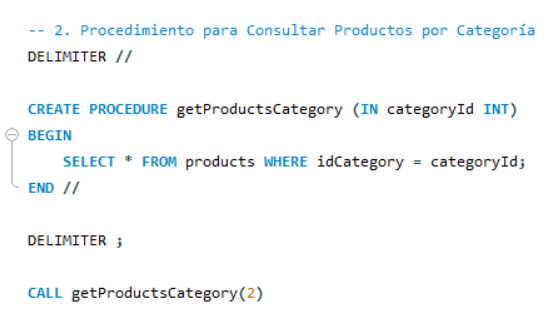
# Procedimientos Almacenados

Los **procedimientos almacenados** son conjuntos de instrucciones SQL que se almacenan en el servidor y pueden ser ejecutados repetidamente. Se crean utilizando la sentencia CREATE PROCEDURE y se invocan con CALL.

## Tienda por ID



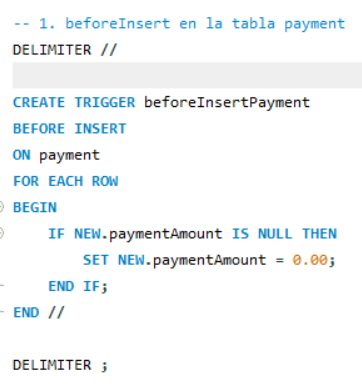
## Productos por categoria.



# Triggers

Los **triggers** son procedimientos que se ejecutan automáticamente en respuesta a eventos específicos en la base de datos, como inserciones, actualizaciones o eliminaciones. Se definen con la sentencia CREATE TRIGGER y pueden ser configurados para activarse antes o después del evento.

## Before Insert payment



## BeforeUpdate product



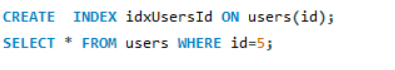
# Indices

Los **índices en bases de datos SQL** son estructuras de datos que mejoran la velocidad de las operaciones de búsqueda y recuperación de datos. Funcionan de manera similar a un índice en un libro, permitiendo localizar rápidamente registros específicos sin necesidad de escanear toda la tabla.

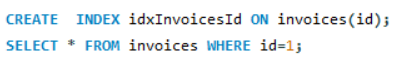
## Indices Agrupados

Un **índice agrupado** (clustered index) es una estructura de datos en bases de datos SQL que determina el orden físico en que se almacenan las filas de una tabla.

### Users



### Invoices



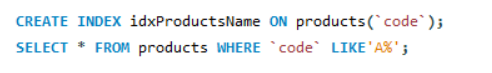
## Indices No Agrupados

Un **índice no agrupado** (non-clustered index) es una estructura de datos en bases de datos SQL que permite mejorar la velocidad de búsqueda y recuperación de datos sin modificar el orden físico en que se almacenan los registros en la tabla.

### Roles por nombre



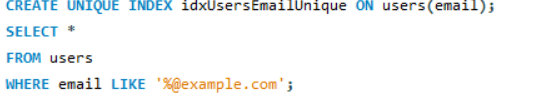
### Producto por codigo



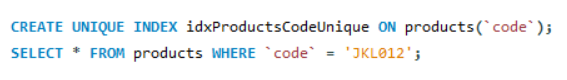
## Indices Unicos

 Asegura que todos los valores en la columna del índice sean únicos, evitando duplicados y mejorando la integridad de los datos

### Email de ususario



### Codigo de product



## Indices Compuestos

Los **índices compuestos** son estructuras de datos en bases de datos SQL que permiten indexar múltiples columnas de una tabla al mismo tiempo. Esto mejora la eficiencia de las consultas que filtran o clasifican datos basándose en varias columnas.

### Facturas por cliente

