DigitalHouse > Coding School

DATA SCIENCE

Clase 46

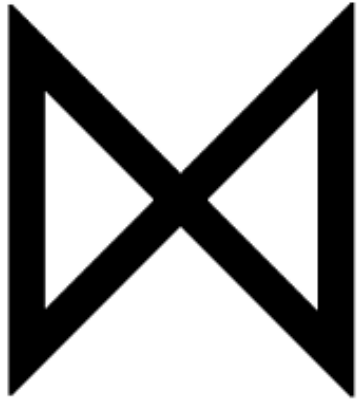
JOINS SQL

2017

JOINS SQL

OBJETIVOS DE LA CLASE

1. **Explicar qué es una operación JOIN**
2. **Visualizar el JOIN como una operación entre conjuntos**
3. **Distinguir diferentes tipos de JOIN**
4. **Realizar JOINS en SQL**



COMBINANDO TABLAS

Datos Normalizados vs Denormalizados

Hay varias maneras de organizar los datos en una base de datos relacional. Dos definiciones comunes para las configuraciones de datos son: normalizados y denormalizados.

- Datos **Normalizados**: Hay una única tabla por entidad y se usan claves foráneas para conectar las entidades.

- Datos **Denormalizados**: tienen menos tablas y pueden, por ejemplo, poner todos los tweets e información de los usuarios en la misma tabla. Osea, mezclar más de una entidad por tabla y repetir datos de una entidad en distintos registros.

Cada estilo tiene ventajas y desventajas. Las tablas Denormalizadas tienden a duplicar una gran cantidad de información. Por ejemplo, en la tabla de tweets / usuarios combinados, podemos almacenar la dirección de cada usuario muchas veces en distintos registros. En lugar de almacenarla una vez por usuario, la estamos almacenando una vez por cada tweet.
Esto facilita el acceso a los datos si necesitamos encontrar el tweet junto con la ubicación del usuario. Pero ¿qué pasa si tenemos que actualizar los datos de un usuario?

Las tablas Normalizadas ahorran espacio de almacenamiento y simplifican las operaciones de creación, modificación y eliminación (evitando errores e inconsistencias). Sin embargo, si alguna vez necesitamos acceder a esas dos piezas de información, necesitaríamos hacer un join de las dos tablas, que puede ser una operación bastante lenta.

JOINS

- Los Joins SQL se utilizan cuando los datos se distribuyen en diferentes tablas. Una operación join permite combinar filas de dos o más tablas en una sola tabla nueva. Para que esto sea posible, debe existir un campo común entre las tablas.

- Las operaciones de JOIN se pueden considerar como operaciones entre dos conjuntos, donde los registros con la misma clave se combinan y los registros que faltan en un conjunto se descartan o se incluyen como valores NULL.

- Imaginemos que contamos con un dataset que contiene las siguientes tablas: Personas y Logros
- Veamos el contenido de las tablas:

```
SELECT * FROM Personas;
SELECT * FROM Logros;
```

Personas

PersonalID	Apellido
1	Fleming
2	Eratóstenes
3	Newton
4	Fernandez

Logros

LogroID	PersonalID	Logro
1	1	Descubrimiento de la Penicilina
2	2	Cálculo del perímetro terrestre
3	3	Ley de gravitación universal
4	3	Desarrollo del Cálculo
5		Cura del cancer

INNER JOIN

- El tipo más común de combinación es el: INNER JOIN (join simple). El INNER JOIN devuelve todas las filas de ambas tablas donde se cumple la condición de combinación.
- Observen en las tablas de Personas y Logros que la columna PersonaID en la tabla Logros hace referencia a PersonaID en la tabla Personas.

Personas

PersonaID	Apellido
1	Fleming

Logros

LogroID	PersonaID	Logro
1	1	Descubrimiento de la Penicilina

- La relación entre las dos tablas es la columna PersonaID. Por lo tanto, podemos combinar las dos tablas para obtener una tabla como la siguiente:

Apellido	Logro
Fleming	Descubrimiento de la Penicilina
Eratóstenes	Cálculo del perímetro terrestre
Newton	Ley de gravitación universal
Newton	Desarrollo del Cálculo

- Donde la información contenida en las dos tablas est
combinada en una sola tabla,
usando la clave común PersonaID.

- A partir de las tablas anteriores, para producir la tabla combinada, donde Apellido provenga de la tabla Personas y Logro provenga de la tabla Logros, se puede utilizar la siguiente sentencia SQL:

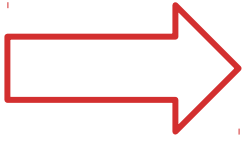
```
SELECT Personas.Apellido, Logros.Logro
FROM Personas
INNER JOIN Logros
ON Personas.PersonaID = Logros.PersonaID;
```

- El INNER JOIN hace la intersección de dos tablas, excluyendo los registros donde PersonaID es null en cualquiera de las dos tablas.

PersonaID	Apellido
1	Fleming
2	Eratóstenes
3	Newton
4	Fernandez

⋈

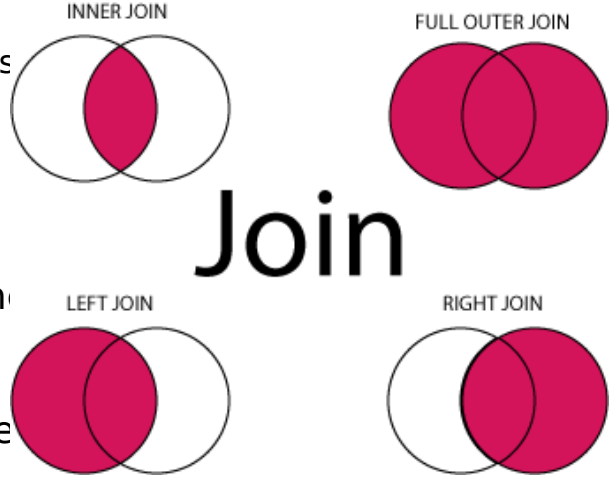
LogroID	PersonaID	Logro
1	1	Descubrimiento de la Penicilina
2	2	Cálculo del perímetro terrestre
3	3	Ley de gravitación universal
4	3	Desarrollo del Cálculo
5		Cura del cancer



Apellido	Logro
Fleming	Descubrimiento de la Penicilina
Eratóstenes	Cálculo del perímetro terrestre
Newton	Ley de gravitación universal
Newton	Desarrollo del Cálculo

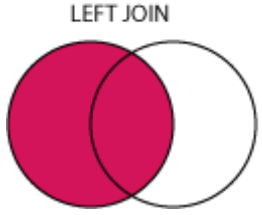
DEMO: DISTINTOS TIPOS DE JOIN

- Hay varios tipos de operaciones de JOIN.
  - **INNER JOIN**: Retorna todos los registros donde haya al menos una coincidencia en ambas tablas
  - **LEFT JOIN**: Retorna todos los registros de la tabla izquierda, y los registros que coincidan de la tabla derecha
  - **RIGHT JOIN**: Retorna todos los registros de la tabla derecha, y los registros que coincidan de la tabla izquierda
  - **FULL OUTER JOIN**: Retorna todos los registros de ambas correspondencia



- Es mucho más fácil entender los JOIN como operacion de intersección de conjuntos.
- Existe una teoría matemáticamente sólida detrás de e operaciones llamada Álgebra Relacional.

LEFT JOIN



El LEFT JOIN retorna todas las filas de la tabla izquierda (tabla1), con las filas coincidentes en la tabla derecha (tabla2). El resultado es NULL en el lado derecho cuando no hay coincidencia.

Sintaxis del Left Join:

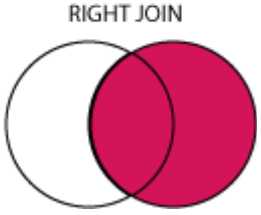
```
SELECT nombres_columnas
FROM tabla1
LEFT JOIN tabla2
ON tabla1.columna_relacion=tabla2.columna_relacion;
```

Ejemplo para Personas y Logros:

```
SELECT Personas.Apellido,Logros.Logro
FROM Personas
LEFT JOIN Logros
ON Personas.PersonaID = Logros.PersonaID;
```

Apellido	Logro
Fleming	Descubrimiento de la Penicilina
Eratóstenes	Cálculo del perímetro terrestre
Newton	Ley de gravitación universal
Newton	Desarrollo del Cálculo
Fernandez	

RIGHT JOIN



De forma similar, el RIGHT JOIN devuelve todas las filas de la tabla derecha (tabla2), con las filas coincidentes en la tabla de la izquierda (tabla1).
El resultado es NULL del lado izquierdo cuando no hay coincidencia.

Sintaxis del Right Join:

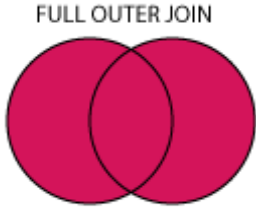
```
SELECT nombres_columnas
FROM tabla1
RIGHT JOIN tabla2
ON tabla1.columna_relacion= tabla2.columna_relacion;
```

Ejemplo para Personas y Logros:

```
SELECT Personas.Apellido, Logros.Logro
FROM Personas
RIGHT JOIN Logros
ON Personas.PersonaID = Logros.PersonaID;
```

Apellido	Logro
Fleming	Descubrimiento de la Penicilina
Eratóstenes	Cálculo del perímetro terrestre
Newton	Ley de gravitación universal
Newton	Desarrollo del Cálculo
	Cura del cancer

FULL OUTER JOIN



El FULL OUTER JOIN retorna todas las filas de la tabla de la izquierda (tabla1) y de la tabla de la derecha (tabla2).
El FULL OUTER JOIN combina el resultado de LEFT y RIGHT JOIN. En este caso podemos tener valores NULL de ambos lados.

Sintaxis del FULL OUTER JOIN:

```
SELECT nombres_columnas
FROM tabla1
FULL OUTER JOIN tabla2
ON tabla1.columna_relacion= tabla2.columna_relacion;
```

Ejemplo para Personas y Logros:

```
SELECT Personas.Apellido, Logros.Logro
FROM Personas
FULL OUTER JOIN Logros
ON Personas.PersonaID = Logros.PersonaID;
```

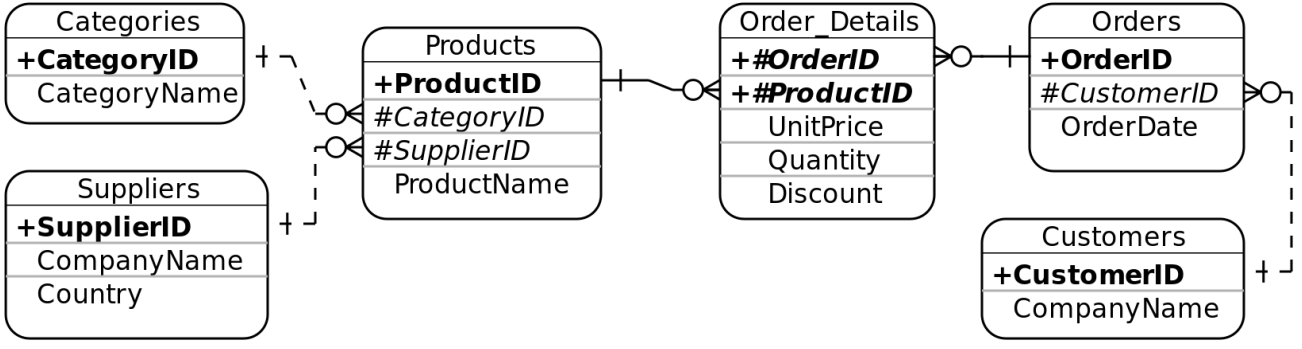
Apellido	Logro
Fleming	Descubrimiento de la Penicilina
Eratóstenes	Cálculo del perímetro terrestre
Newton	Ley de gravitación universal
Newton	Desarrollo del Cálculo
	Cura del cancer
Fernandez	

PRÁCTICA GUIADA: OTROS JOINS

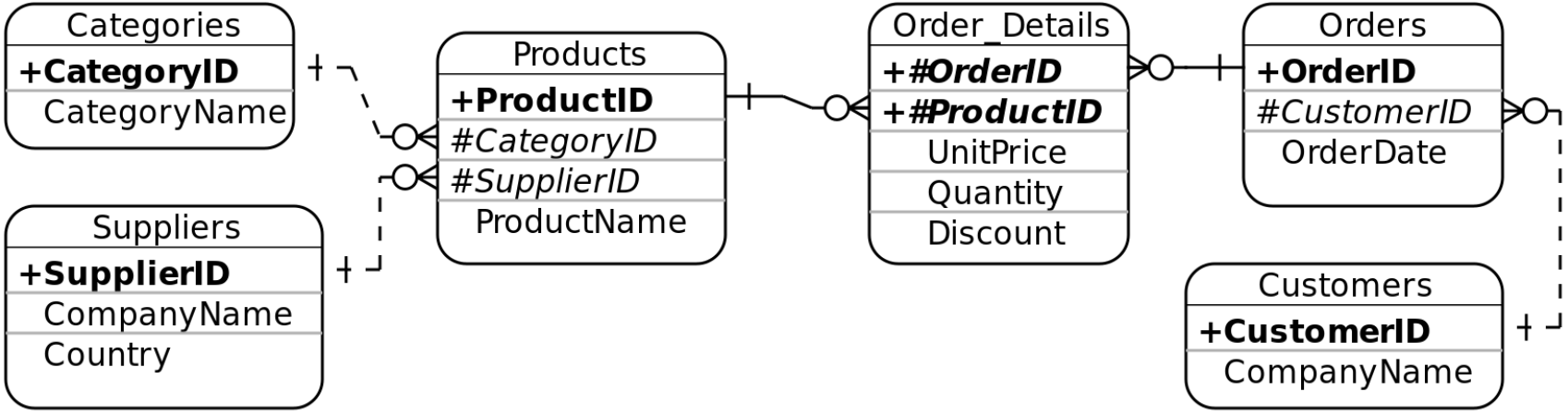
- Para esta práctica utilizaremos la base de datos Northwind. En el notebook tienen las instrucciones para la conexión son:

  Anteriormente se utilizó *pandas.merge* para responder preguntas que contenían datos de múltiples tablas. Aquí intentaremos responder ese tipo de preguntas utilizando SQL puro.

- Visualicemos el DER de las tablas que vamos a utilizar a continuación.



- Se desea visualizar las órdenes con su respectivo cliente, imprimiendo en la respuesta OrderID, CompanyName y OrderDate.



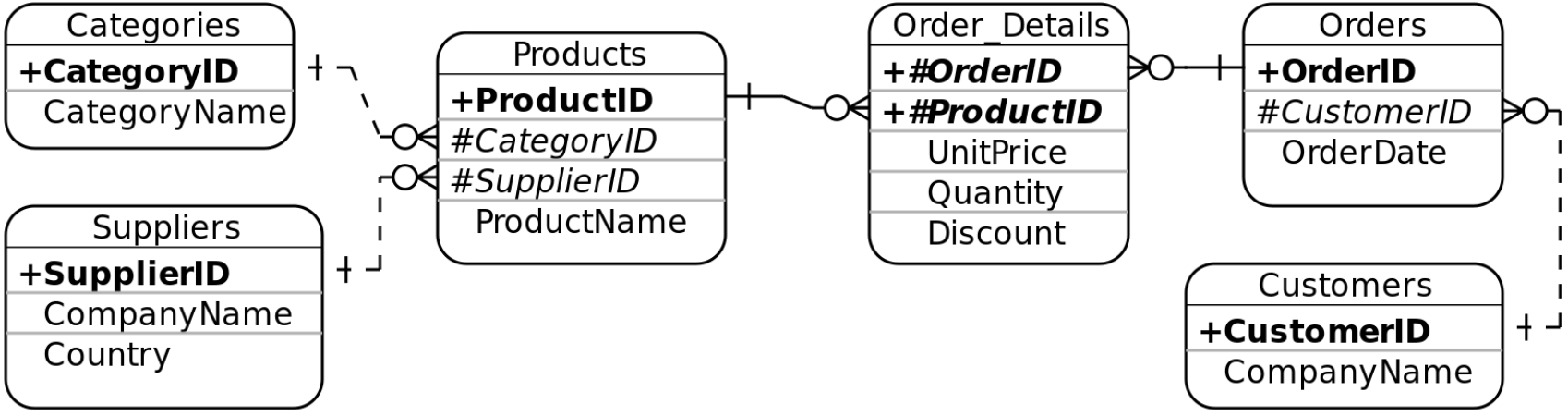
- Se desea visualizar las órdenes con su respectivo cliente, imprimiendo en la respuesta OrderID, CompanyName y OrderDate.

Solución:

```
SELECT orders."OrderID", customers."CompanyName", orders."OrderDate"
FROM orders
INNER JOIN customers ON orders."CustomerID" = customers."CustomerID";
```

OrderID	CompanyName	OrderDate
10248	Vins et alcools Chevalier	1996-07-04
10249	Toms Spezialitäten	1996-07-05
10250	Hanari Carnes	1996-07-08
10251	Victuailles en stock	1996-07-08
10252	Suprêmes délices	1996-07-09

- Se desea visualizar los clientes que no hayan realizado órdenes, imprimiendo en la respuesta CustomerID y CompanyName.



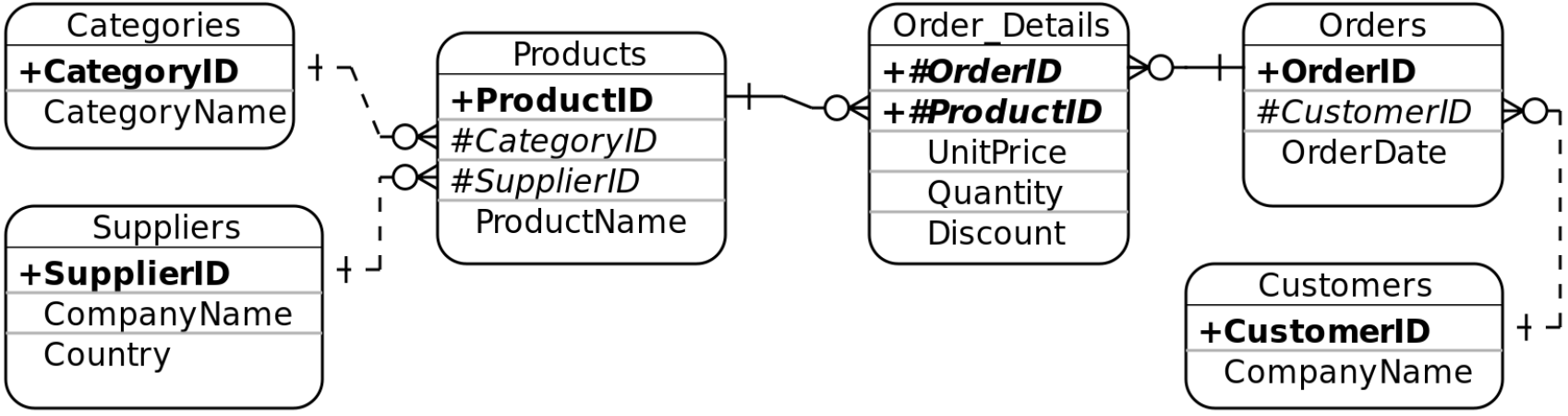
- Se desea visualizar los clientes que no hayan realizado órdenes, imprimiendo en la respuesta CustomerID y CompanyName.

Solución:

```
SELECT customers."CustomerID", customers."CompanyName"
FROM customers
LEFT JOIN orders ON orders."CustomerID" = customers."CustomerID"
WHERE orders."CustomerID" IS NULL;
```

CustomerID	CompanyName
FISSA	FISSA Fabrica Inter. Salchichas S.A.
PARIS	Paris spécialités

- ¿Cuántos productos por categoría contiene el catálogo? Imprima la respuesta con CategoryName y Cantidad.



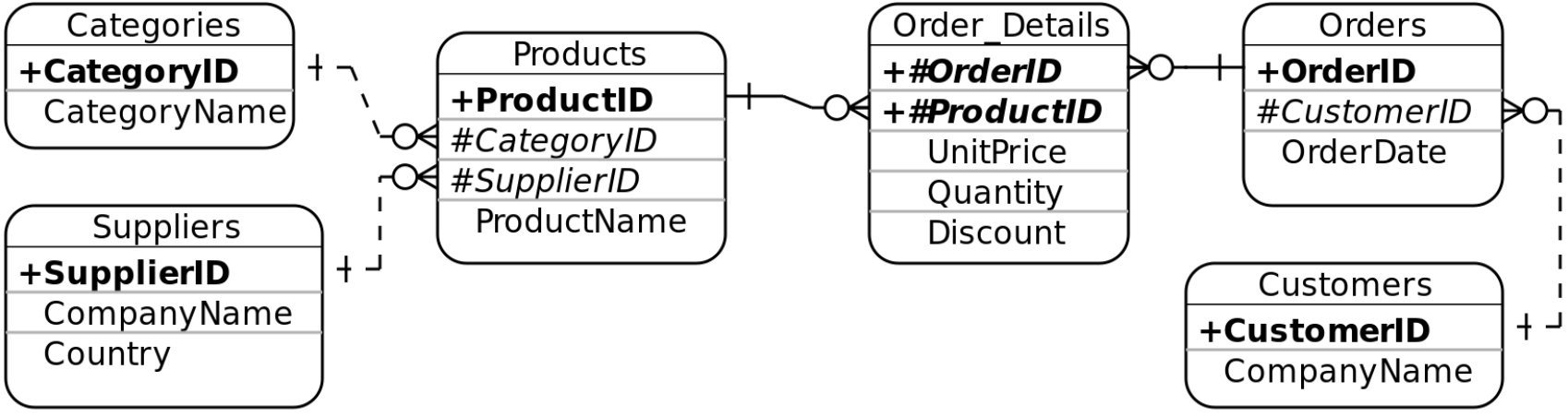
- ¿Cuántos productos por categoría contiene el catálogo? Imprima la respuesta con CategoryName y Cantidad.

Solución:

```
SELECT "CategoryName",count("ProductID") AS "Cantidad"
FROM categories AS c
LEFT JOIN products AS p ON p."CategoryID" = c."CategoryID"
GROUP BY c."CategoryName"
ORDER BY "Cantidad" DESC;
```

CategoryName	Cantidad
Confections	13
Condiments	12
Beverages	12
Seafood	12
Dairy Products	10
Grains/Cereals	7
Meat/Poultry	6
Produce	5
Magic	0

- ¿Cuáles son los 5 clientes están generando los mayores ingresos? Imprima una tabla con CustomerID e Ingresos. Tendrá que utilizar datos de 3 tablas.



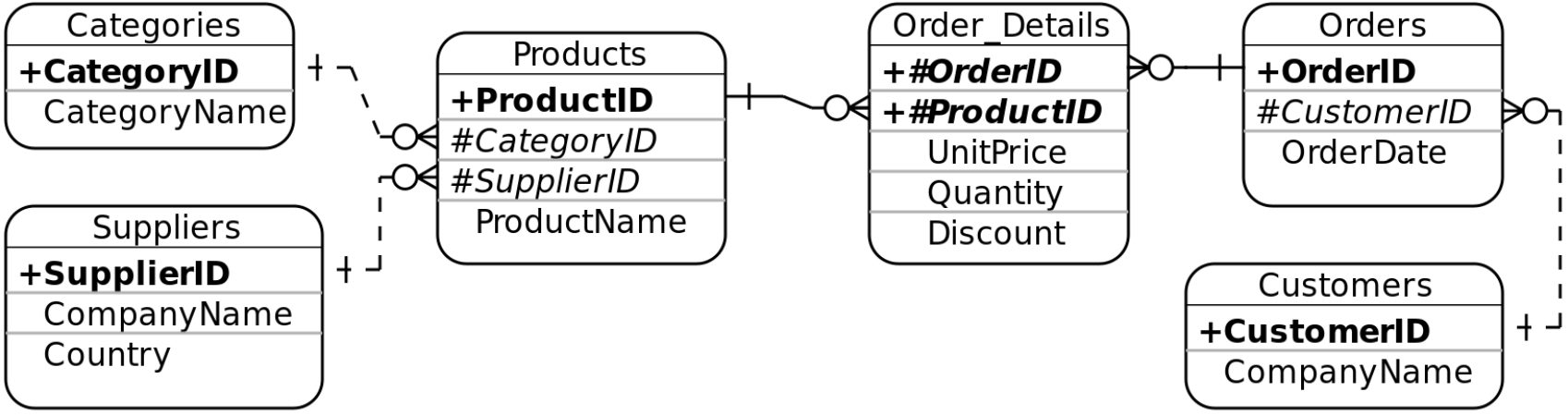
- ¿Cuáles son los 5 clientes que están generando los mayores ingresos? Imprima una tabla con CustomerID e Ingresos. Tendrá que utilizar datos de 3 tablas.

Solución:

```
SELECT c."CustomerID",
   SUM ("UnitPrice" *
      "Quantity" *
      (1.0 - "Discount"))
  AS "Ingresos"
FROM customers AS c
JOIN orders AS o ON c."CustomerID" = o."CustomerID"
JOIN order_details AS od ON o."OrderID" = od."OrderID"
GROUP BY c."CustomerID"
ORDER BY "Ingresos" DESC
LIMIT 5;
```

CustomerID	Ingresos
QUICK	110277.305030394
ERNSH	104874.978143677
SAVEA	104361.949540394
RATTC	51097.8008282682
HUNGO	49979.9050814955

- ¿En qué países se encuentran los 5 principales proveedores por número de unidades suministradas que hayan sido vendidas? Imprima una tabla con el CompanyName del proveedor, Country y el número total de Unidades Vendidas.



- ¿En qué países se encuentran los 5 principales proveedores por número de unidades suministradas que hayan sido vendidas? Imprima una tabla con el CompanyName del proveedor, Country y el número total de Unidades Vendidas.

Solución:

```
SELECT s."CompanyName",s."Country",
   SUM(od."Quantity") AS "Unidades"
FROM suppliers AS s
JOIN products AS p
   ON s."SupplierID" = p."SupplierID"
JOIN order_details AS od
   ON p."ProductID" = od."ProductID"
GROUP BY s."SupplierID"
ORDER BY "Unidades" DESC
LIMIT 5;
```

CompanyName	Country	Unidades
Plutzer Lebensmittelgroßmärkte AG	Germany	4072
Pavlova, Ltd.	Australia	3937
Specialty Biscuits, Ltd.	UK	3679
Gai pâturage	France	3073
Norske Meierier	Norway	2526

DEMO: SUBQUERY

- El lenguaje SQL es muy versátil y se pueden hacer cosas más allá de JOIN entre dos tablas diferentes.
- Un **SUBQUERY** o query interno o query anidado, es un query dentro de otro query SQL.
- Se pueden utilizar por ejemplo para establecer condiciones sobre el query principal y así restringir los resultados del query principal (externo), según los resultados del query secundario (interno).
- Un subquery puede ser usado dentro de sentencias SELECT, INSERT, UPDATE y DELETE, con operadores como =, <, >, >=, <=, IN, BETWEEN etc.

Sintaxis

Aquí hay un ejemplo de un subquery. La tabla resultante del subquery es usada como condición en el WHERE del query principal.

```
SELECT columna1
  FROM tabla1
  WHERE columna2 [Operador de comparación]
    (SELECT columna3
     FROM tabla2
     WHERE condición);
```

- Por ejemplo, extraigamos todas las órdenes de clientes de Francia.

```
SELECT "OrderID" FROM Orders
WHERE "CustomerID" IN
      (SELECT "CustomerID"
       FROM Customers
       WHERE "Country" = 'France');
```

OrderID
10248
10251
10265
10274
10295
10297
10311
10331

- Vean que en este caso es equivalente a hacer un JOIN de la siguiente for

```
SELECT "OrderID" FROM Orders
JOIN Customers ON Orders."CustomerID" = Customers."CustomerID"
WHERE Customers."Country" = 'France';
```

- Veamos otro ejemplo, aquí vamos a mostrar los 5 productos más baratos cuyos precios sean mayores a la media e imprimiremos para ellos su categoría, nombre y precio.

```
SELECT "CategoryName","ProductName","UnitPrice"
FROM products AS p
JOIN categories AS c ON c."CategoryID" = p."CategoryID"
WHERE p."UnitPrice" > (SELECT AVG("UnitPrice") FROM Products)
ORDER BY p."UnitPrice" ASC
LIMIT 5;
```

CategoryName	ProductName	UnitPrice
Produce	Uncle Bob's Organic Dried Pears	30
Seafood	Ikura	31
Confections	Gumbär Gummibärchen	31.23
Dairy Products	Mascarpone Fabioli	32
Meat/Poultry	Perth Pasties	32.8

OTROS COMANDOS SQL

- Trabajo en equipos: vayan a http://www.w3schools.com/sql y elijan una orden SQL de la que no hayan oído hablar todavía.
- Lean acerca de ella durante 5 minutos, luego explíquele a su equipo
- Utilizaremos los últimos 5 minutos para compartir algunos hallazgos interesantes con el resto de la clase.

CONCLUSIÓN

- En esta clase hemos comenzado a descubrir todo el potencial de las bases de datos relacionales mediante JOINs y subquerys. Estos nos permiten mezclar y combinar datos de varias tablas, con el fin de extraer resultados útiles.