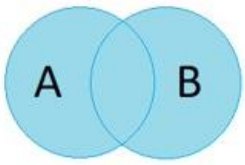
EXERCISES QUE TRABAJAREMOS EN EL CUE:

- EXERCISE 1: RELACIONES DE TABLAS.
- EXERCISE 2: REALIZANDO CONSULTAS MULTITABLAS.

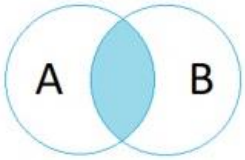
EXERCISE 1: RELACIONES DE TABLAS.

En la mayoría de las consultas intervienen más de una tabla, las cuales deben relacionarse para poder obtener las columnas requeridas, y así lograr el diseño esperado. Estas relaciones se logran gracias a las claves primarias (*PK*) y foráneas (*FK*).

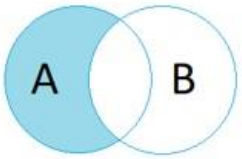
Las operaciones más comunes que relacionan dos, o más tablas, son:



UNION
A U B



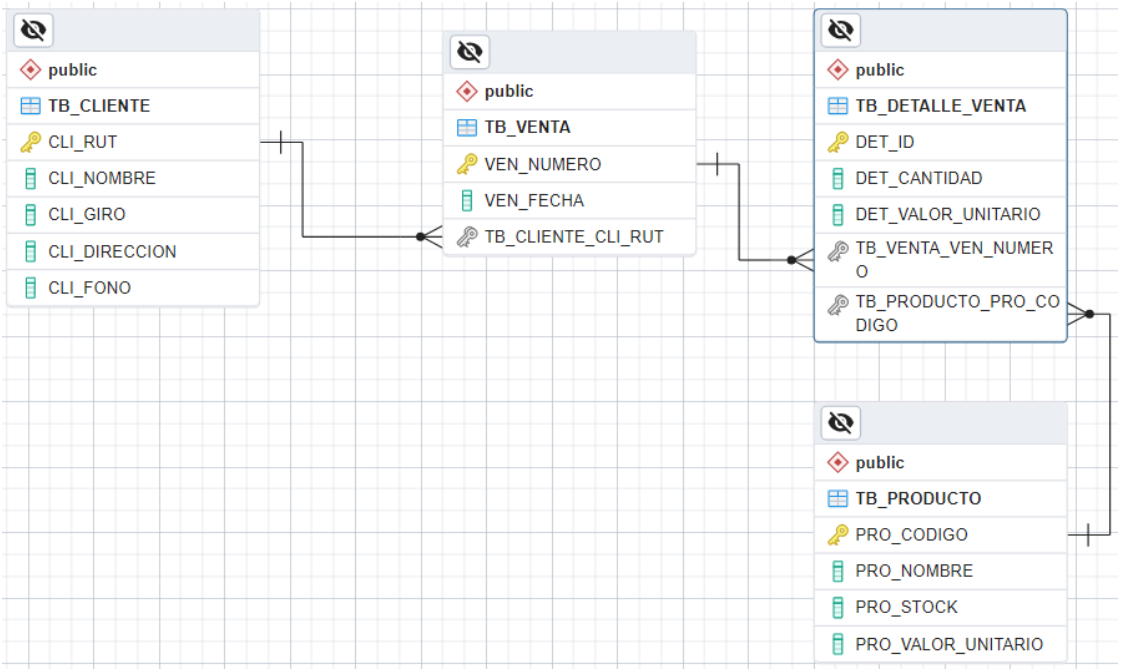
INTERSECCION
A ∩ B



DIFERENCIA
A - B

INNER JOIN ON

Supongamos el siguiente Modelo Relacional:



Donde se nos pide el siguiente listado del detalle de venta por año, ordenadas por fecha de venta.

Fecha de Venta	N° de Venta	Total Venta

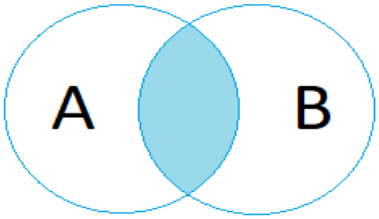
Para poder construir este listado o informe, debemos contar con el Modelo Relacional, y así diseñar o pensar nuestra consulta. Si revisamos dicho Modelo, se observará que los datos solicitados pueden ser obtenidos desde las tablas **TB_VENTA** y **TB_DETALLE_VENTA**, por lo que debemos relacionar ambas tablas mediante la relación indicada en el modelo. Por otra parte, las columnas del informe “Fecha de Venta” y “N° de Venta”, se pueden obtener de forma directa, pero “Total Venta” no es directa, por lo que deben realizarse operaciones matemáticas para calcularla.

En primer lugar, construiremos una consulta que traiga todas las columnas que se requieren en el informe.

```
SELECT VEN_NUMERO, VEN_FECHA, DET_CANTIDAD, DET_VALOR_UNITARIO
FROM TB_VENTA AS V
INNER JOIN TB_DETALLE_VENTA AS DV ON V.VEN_NUMERO = DV.VEN_NUMERO
```

La sentencia `JOIN` relaciona `TB_VENTA` con `TB_DETALLE_VENTA`, a través del campo `VEN_NUMERO (PK)` en `TB_VENTA`, y el campo `VEN_NUMERO (FK)` en `TB_DETALLE_VENTA`. Esta relación significa que, para cada registro de `TB_VENTA`, se buscarán los registros de `TB_DETALLE_VENTA` que cumplan con la condición: `V.VEN_NUMERO = DV.VEN_NUMERO.`

Gráficamente, estamos obteniendo la intersección entre las dos tablas.



Luego, el subtotal de cada detalle de una venta lo podemos obtener de la siguiente forma:

```
SELECT VEN_NUMERO,
VEN_FECHA,
DET_CANTIDAD * DET_VALOR_UNITARIO AS SubTotal
FROM TB_VENTA AS V
INNER JOIN TB_DETALLE_VENTA AS DV ON V.VEN_NUMERO = DV.VEN_NUMERO
```

Ahora, debemos agrupar por los campos que se repiten, para poder obtener la suma de todos los subtotales con la función de agregado SUM, lo que representa el total de la venta. La consulta sería la siguiente:

```
SELECT VEN_NUMERO AS NumeroVenta,
```

```
VEN_FECHA AS FechaVenta,
SUM(DET_CANTIDAD * DET_VALOR_UNITARIO) AS TotalVenta
FROM TB_VENTA AS V
INNER JOIN TB_DETALLE_VENTA AS DV ON V.VEN_NUMERO = DV.VEN_NUMERO
GROUP BY VEN_NUMERO, VEN_FECHA
```

Para finalizar, agregamos el filtro para clasificar y listar solo las ventas de un año.

```
SELECT VEN_NUMERO AS NumeroVenta,
VEN_FECHA AS FechaVenta,
SUM(DET_CANTIDAD * DET_VALOR_UNITARIO) AS TotalVenta
FROM TB_VENTA AS V
INNER JOIN TB_DETALLE_VENTA AS DV ON V.VEN_NUMERO = DV.VEN_NUMERO
WHERE to_number(to_char(VEN_FECHA,'YYYY')) = 2020
GROUP BY VEN_NUMERO, VEN_FECHA
```

Al relacionar mediante la sentencia **JOIN**, se puede hacer por más de un campo, e incluso, con cualquier operador, no necesariamente con uno igual.

```
…
FROM A
INNER JOIN B ON A.C1 = B.C2 AND A.C2 = B.C3
…
```

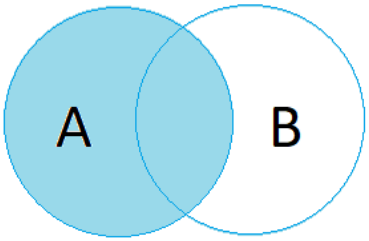
```
…
FROM A
INNER JOIN B ON A.C1 = B.C2 OR A.C2 > B.C3
…
```

```
…
FROM A
INNER JOIN B ON expresión
…
```

LEFT JOIN ON

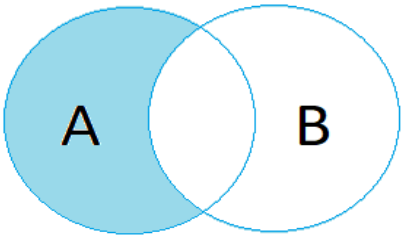
La sintaxis es idéntica a **INNER JOIN**, con la diferencia que se incluirán todos los elementos de la tabla, a la izquierda de la sentencia.

```
SELECT *
FROM A
LEFT JOIN B ON A.ColumnaN = B.ColumnaM
```



Agregando un criterio, podemos obtener la diferencia A – B:

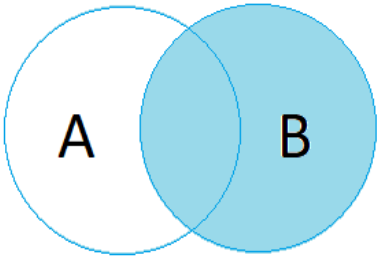
```
SELECT *
FROM A
LEFT JOIN B ON A.ColumnaN = B.ColumnaM
WHERE B.ColumnaM IS NULL
```



RIGHT JOIN ON

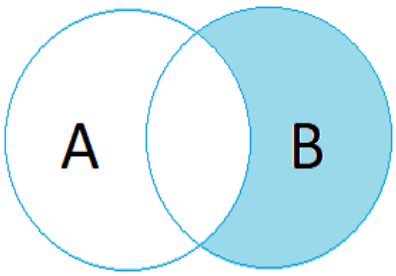
La sintaxis es idéntica a **INNER JOIN**, con la diferencia que se incluirán todos los elementos de la tabla, a la derecha de la sentencia.

```
SELECT *
FROM A
RIGHT JOIN B ON A.ColumnaN = B.ColumnaM
```

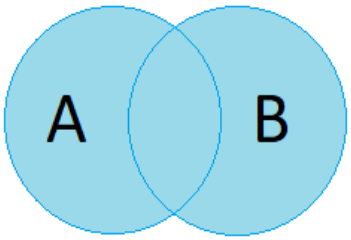


Agregando un criterio, podemos obtener la diferencia A – B:

```
SELECT *
FROM A
RIGHT JOIN B ON A.ColumnaN = B.ColumnaM
WHERE B.ColumnaM IS NULL
```



UNIÓN



La palabra UNION permite añadir el resultado de un SELECT, a otro SELECT. Para ello, ambas instrucciones tienen que utilizar el mismo número y tipo de columnas.

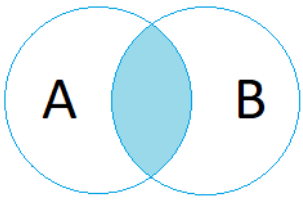
Ejemplo:

```
1 SELECT nombre FROM clientes
2 UNION
3 SELECT nombre FROM proveedores
```

El resultado serán los nombres de los clientes y proveedores que no se encuentren duplicados. La UNION retorna el resultado de las filas de ambas consultas que no se encuentren duplicados. Para el caso de UNION ALL, el funcionamiento es idéntico, pero éste no elimina los datos repetidos.

```
1 SELECT nombre FROM clientes
2 UNION ALL
3 SELECT nombre FROM proveedores
```

INTERSECCIÓN

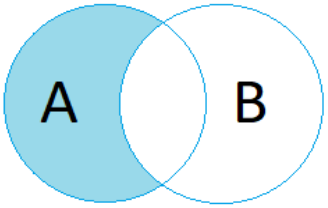


La sentencia INTERSECT permite unir dos consultas SELECT, de modo que el resultado serán las filas que estén presentes en ambas consultas, es decir, la intercepción de ambas.

Ejemplo:

```
SELECT nombre FROM clientes
INTERSECT
SELECT nombre FROM proveedores
```

DIFERENCIA



Con MINUS también se combinan dos consultas SELECT, de forma que aparecerán los registros del primer SELECT que no estén presentes en el segundo.

Ejemplo:

```
SELECT nombre, marca
FROM productos
WHERE tipo=1
MINUS
SELECT nombre, marca
FROM productos
WHERE tipo=2
```

PRODUCTO CARTESIANO

Opera sobre dos o más tablas, efectuando un producto cartesiano del contenido de éstas, no siendo necesario que ambas posean la misma estructura, y devolviendo una nueva, cuyo contenido es todas las posibles combinaciones de las filas de una de las tablas.

```
SELECT …. FROM T1, T2, …, Tn
```

Supongamos que tenemos las tablas **Trabajador** y **Comuna,** con los datos que se indican a continuación:

Trabajador	
Nombre	**CodigoComuna**
Juan	2
Pedro	1
Carlos	1

Comuna	
Codigo	**NombreComuna**
1	Santiago
2	Concepción

Si queremos realizar el producto cartesiano entre ambas tablas, podríamos desarrollar la siguiente operación:

```
SELECT Nombre, NombreComuna
FROM Trabajador, Comuna
ORDER BY Trabajador, Comuna
```

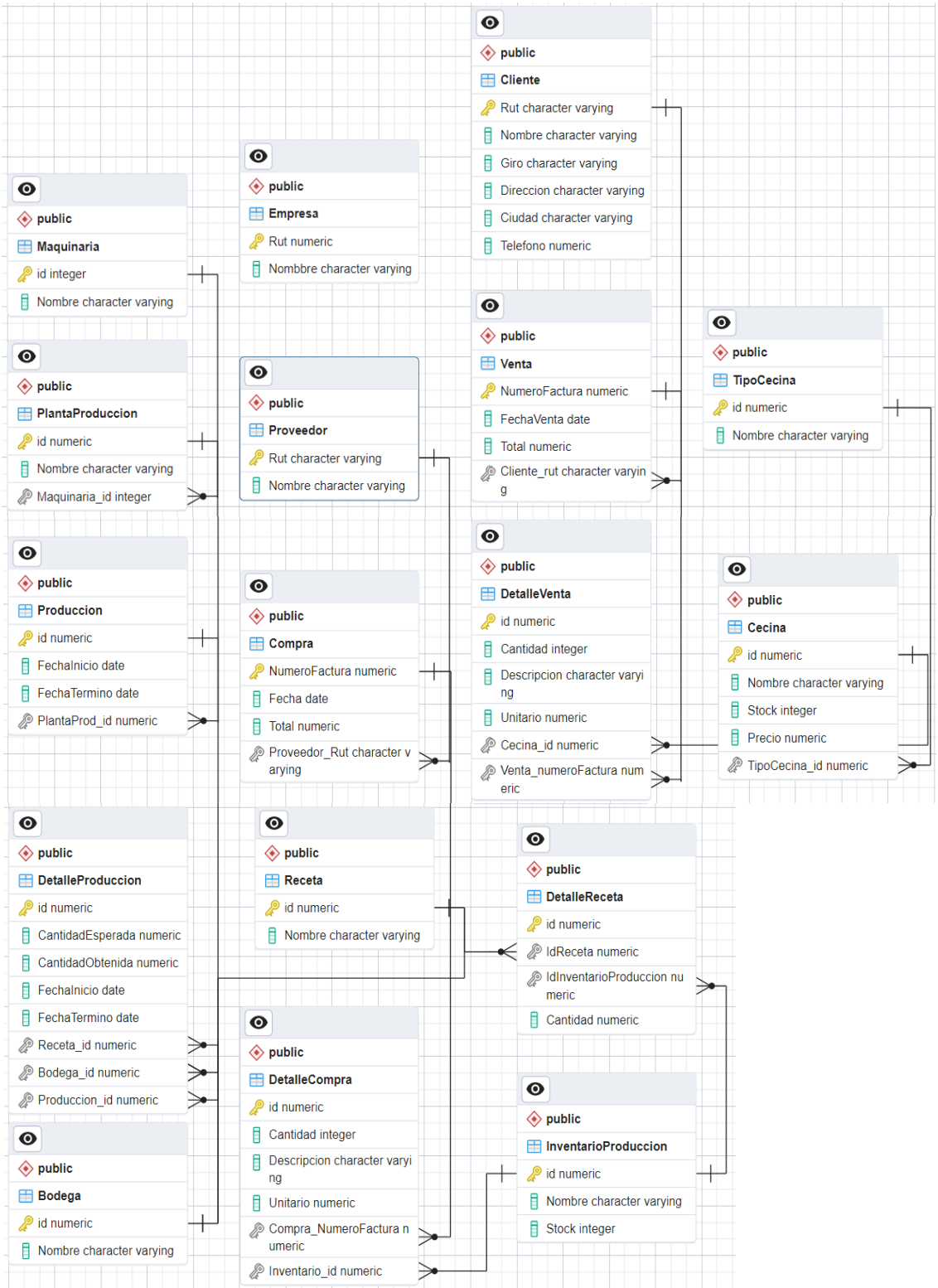
El resultado sería:

Nombre	NombreComuna
Carlos	Concepción
Carlos	Santiago
Juan	Concepción
Juan	Santiago
Pedro	Concepción
Pedro	Santiago

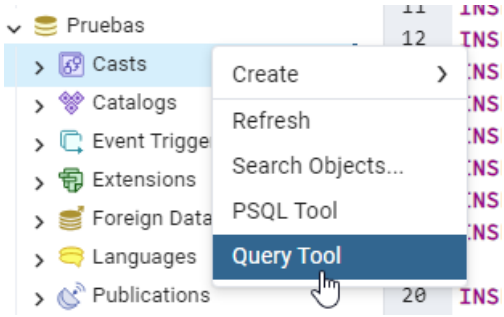
Esto representa como si los trabajadores ejercen en todas las comunas, y lo que se realizaron fueron todas las combinaciones posibles entre ambas tablas.

EXERCISE 2: REALIZANDO CONSULTAS MULTITABLAS.

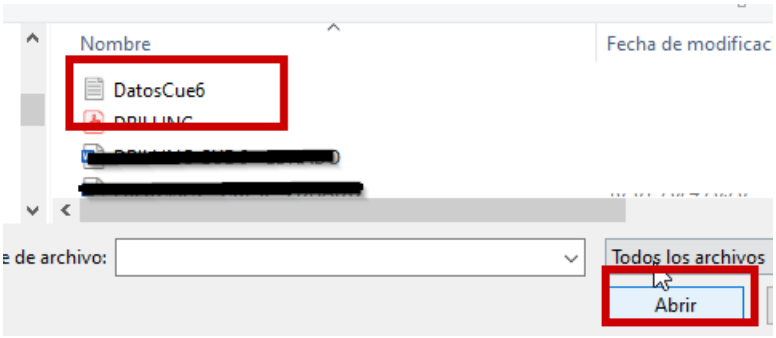
Realizaremos consultas multitablas, basándonos en el siguiente Modelo Entidad - Relación:



Lo primero que haremos será insertar algunos datos en nuestra tabla, para eso nos dirigimos a “query tools”



Aquí podremos abrir el archivo con los datos que vamos a insertar




```
INSERT INTO "Venta" ("NumeroFactura", "FechaVenta", "Total", "Cliente_Rut") VALUES (1, '2020-01-15', 10000, '789123-8');
INSERT INTO "Venta" ("NumeroFactura", "FechaVenta", "Total", "Cliente_Rut") VALUES (2, '2020-06-18', 15000, '7456789-0');
INSERT INTO "Venta" ("NumeroFactura", "FechaVenta", "Total", "Cliente_Rut") VALUES (3, '2020-07-05', 20000, '456789-0');
INSERT INTO "Venta" ("NumeroFactura", "FechaVenta", "Total", "Cliente_Rut") VALUES (4, '2020-09-20', 18000, '123456-k');
INSERT INTO "Venta" ("NumeroFactura", "FechaVenta", "Total", "Cliente_Rut") VALUES (5, '2020-12-29', 8000, '123456-k');
```

Una vez insertados los datos lo primero que haremos será listar las compras del año 2020, con los atributos: Fecha, Total, NombreProveedor, RutProveedor, y Cantidad de ítems comprados.

Para esto, debemos relacionar las tablas “Compra” y “DetalleCompra”. Éstas se relacionan mediante el atributo NumeroFactura. Por este motivo, escribiremos el comando:

```
1  SELECT * FROM "Compra"
2  INNER JOIN "DetalleCompra"
3  ON "NumeroFactura" = "Compra_NumeroFactura"
4
```

Obteniendo como resultado las columnas de ambas tablas:

	NumeroFactu numeric	Fecha date	Total numeric	Proveedor_Ru character var	id numeric	Cantidad integer	Descripcion character varying	Unitario numeric	Compra_Num numeric	Inventario_id numeric
1	1	2020-01-12	50000	987654-9	1	2	Cerdo Entero	10000	1	[null]
2	1	2020-01-12	50000	987654-9	2	1	Vaca entera	30000	1	[null]
3	2	2020-02-12	80000	123456-1	3	1	Surtido de cecinas	80000	2	[null]
4	3	2020-08-12	70000	987654-9	4	1	Surtido de cecinas	70000	3	[null]
5	4	2020-09-18	50000	123456-1	5	3	Cerdo Entero	9000	4	[null]
6	4	2020-09-18	50000	123456-1	6	1	Vasa entera	32000	4	[null]

Filtraremos los atributos correspondientes. Para eso, debemos crear la relación de las tablas con la entidad proveedora.

Agregamos en la siguiente línea: `INNER JOIN PROVEEDOR ON PROVEEDOR_RUT = RUT`, quedando el comando total así:

```
SELECT * FROM "Compra"
INNER JOIN "DetalleCompra"
ON "NumeroFactura" = "Compra_NumeroFactura"
INNER JOIN "Proveedor" ON "Proveedor_Rut" = "Rut"

```

Como resultado tendremos las columnas correspondientes a la tabla proveedor

NumeroFactu numeric	Fecha date	Total numeric	Proveedor_Ru character var	id numeric	Cantidad integer	Descripcion character varying	Unitario numeric	Compra_Num numeric	Inventario_id numeric	Rut character var	Nombre character varying
1	2020-01-12	50000	987654-9	1	2	Cerdo Entero	10000	1	[null]	987654-9	Faena San Nicolas
1	2020-01-12	50000	987654-9	2	1	Vaca entera	30000	1	[null]	987654-9	Faena San Nicolas
2	2020-02-12	80000	123456-1	3	1	Surtido de cecinas	80000	2	[null]	123456-1	Cecinas Mustafá
3	2020-08-12	70000	987654-9	4	1	Surtido de cecinas	70000	3	[null]	987654-9	Faena San Nicolas
4	2020-09-18	50000	123456-1	5	3	Cerdo Entero	9000	4	[null]	123456-1	Cecinas Mustafá
4	2020-09-18	50000	123456-1	6	1	Vasa entera	32000	4	[null]	123456-1	Cecinas Mustafá

Para filtrar la lista de las compras del año 2020, usaremos el comando:

```
SELECT * FROM "Compra"
INNER JOIN "DetalleCompra"
ON "NumeroFactura" = "Compra_NumeroFactura"
INNER JOIN "Proveedor" ON "Proveedor_Rut" = "Rut"
WHERE TO_CHAR("Fecha", 'YYYY') = '2020'
```

Utilizamos `TO_CHAR` para convertir el formato de la Fecha de `DATE`, a `CHARACTER`. Al ejecutar la query con la fecha, el resultado será el mismo, dado que todos los datos ingresados corresponden al año 2020.

NumeroFactu numeric	Fecha date	Total numeric	Proveedor_Ru character var	id numeric	Cantidad integer	Descripcion character varying	Unitario numeric	Compra_Num numeric	Inventario_id numeric	Rut character var	Nombre character varying
1	2020-01-12	50000	987654-9	1	2	Cerdo Entero	10000	1	[null]	987654-9	Faena San Nicolas
1	2020-01-12	50000	987654-9	2	1	Vaca entera	30000	1	[null]	987654-9	Faena San Nicolas
2	2020-02-12	80000	123456-1	3	1	Surtido de cecinas	80000	2	[null]	123456-1	Cecinas Mustafá
3	2020-08-12	70000	987654-9	4	1	Surtido de cecinas	70000	3	[null]	987654-9	Faena San Nicolas
4	2020-09-18	50000	123456-1	5	3	Cerdo Entero	9000	4	[null]	123456-1	Cecinas Mustafá
4	2020-09-18	50000	123456-1	6	1	Vasa entera	32000	4	[null]	123456-1	Cecinas Mustafá

Continuamos reemplazando la primera parte del comando, por: `SELECT FECHA, TOTAL, NOMBRE AS NOMBREPROVEEEDOR, PROVEEDOR_RUT AS RUTPROVEEDOR, 0 AS CANTIDADITEMS FROM COMPRA`, quedando finalmente:

```
SELECT "Fecha", "Total", "Nombre" AS "NombreProveedor", "Proveedor_Rut" AS "RutProveedor",
0 AS "CantidadItems" FROM "Compra"

INNER JOIN "DetalleCompra"
ON "NumeroFactura" = "Compra_NumeroFactura"
INNER JOIN "Proveedor" ON "Proveedor_Rut" = "Rut"
WHERE TO_CHAR("Fecha", 'YYYY') = '2020'
```

Obteniendo así el siguiente resultado:

	Fecha date	Total numeric	NombreProveedor character varying	RutProveedor character varying	CantidadItems integer
1	2020-01-12	50000	Faena San Nicolas	987654-9	0
2	2020-01-12	50000	Faena San Nicolas	987654-9	0
3	2020-02-12	80000	Cecinas Mustafá	123456-1	0
4	2020-08-12	70000	Faena San Nicolas	987654-9	0
5	2020-09-18	50000	Cecinas Mustafá	123456-1	0
6	2020-09-18	50000	Cecinas Mustafá	123456-1	0

Tal como está definido, el atributo “CantidadItems” nos deja un problema. Al efectuar una consulta, se duplicarían las compras por cada uno de los detalles de éstas. Para evitarlo, agregaremos una línea al final, con la instrucción: `GROUP BY FECHA, TOTAL, NOMBRE, PROVEEDOR_RUT`. Usamos la función “Count”, que devuelve el número de filas de la consulta.

Quedará finalmente como:

```
SELECT "Fecha", "Total", "Nombre" AS "NombreProveedor", "Proveedor_Rut" AS "RutProveedor",
COUNT (*) AS "CantidadItems" FROM "Compra"
INNER JOIN "DetalleCompra"
ON "NumeroFactura" = "Compra_NumeroFactura"
INNER JOIN "Proveedor" ON "Proveedor_Rut" = "Rut"
WHERE TO_CHAR("Fecha", 'YYYY') = '2020'
GROUP BY "Fecha", "Total", "Nombre", "Proveedor_Rut"
```

Quedando de la siguiente manera:

	Fecha date	Total numeric	NombreProveedor character varying	RutProveedor character var	CantidadItem bigint
1	2020-01-12	50000	Faena San Nicolas	987654-9	2
2	2020-02-12	80000	Cecinas Mustafá	123456-1	1
3	2020-08-12	70000	Faena San Nicolas	987654-9	1
4	2020-09-18	50000	Cecinas Mustafá	123456-1	2

Continuaremos realizando un listado de todas las ventas del año 2020, pero con las columnas: Fecha, Total, Nombre, el monto más alto del artículo vendido, y el promedio de los subtotales de los vendidos.

Para realizar esta consulta, comenzamos escribiendo:

```
SELECT "FechaVenta" AS "Fecha", "Total", "Nombre" AS "NombreCliente",
0 AS "MontoMayor",
0 AS "Promedio"
FROM "Venta"
INNER JOIN "DetalleVenta" ON "NumeroFactura" = "Venta_numeroFactura"
INNER JOIN "Cliente" ON "Cliente_rut" = "Rut"
```

De esta forma, obtenemos el siguiente resultado:

	Fecha date	Total numeric	NombreCliente character varyi	MontoMayor integer	Promedio integer
1	2020-01-15	10000	Dafne	0	0
2	2020-01-15	10000	Dafne	0	0
3	2020-06-18	15000	Salim	0	0
4	2020-07-05	20000	Salim	0	0
5	2020-07-05	20000	Salim	0	0
6	2020-07-05	20000	Salim	0	0
7	2020-09-20	18000	Cecil	0	0
8	2020-12-29	8000	Cecil	0	0

Continuaremos con la obtención de las ventas del año 2020 y agruparemos la información con “FechaVenta”, “Total” y “Nombre”

```
SELECT "FechaVenta" AS "Fecha", "Total", "Nombre" AS "NombreCliente",
0 AS "MontoMayor",
0 AS "Promedio"
FROM "Venta"
INNER JOIN "DetalleVenta" ON "NumeroFactura" = "Venta_numeroFactura"
INNER JOIN "Cliente" ON "Cliente_rut" = "Rut"
WHERE TO_CHAR ("FechaVenta", 'YYYY') = '2020'
GROUP BY "FechaVenta", "Total", "Nombre"
```

El resultado quedará:

	Fecha date	Total numeric	NombreCliente character varying	MontoMayor integer	Promedio integer
1	2020-01-15	10000	Dafne	0	0
2	2020-06-18	15000	Salim	0	0
3	2020-07-05	20000	Salim	0	0
4	2020-09-20	18000	Cecil	0	0
5	2020-12-29	8000	Cecil	0	0

Para mostrar el monto más alto de una venta, vamos a desplegar la información. En la tabla DetalleVenta, podemos multiplicar el atributo Cantidad por Unitario, y se los entregamos como parámetros a una función de agregación llamada “max”. Ésta sirve para obtener el mayor valor de una columna determinada.

La consulta quedará de la siguiente forma:

```
SELECT "FechaVenta" AS "Fecha", "Total", "Nombre" AS "NombreCliente",
MAX("Cantidad"*"Unitario") AS "MontoMayor",
FROM "Venta"
INNER JOIN "DetalleVenta" ON "NumeroFactura" = "Venta_numeroFactura"
INNER JOIN "Cliente" ON "Cliente_rut" = "Rut"
WHERE TO_CHAR ("FechaVenta", 'YYYY') = '2020'
GROUP BY "FechaVenta", "Total", "Nombre"
```

Finalmente, para terminar de obtener los datos de esta consulta, nos centraremos en el promedio de los subtotales.

Vamos a utilizar la función AVG, y le entregaremos como parámetro los atributos Cantidad y Unitario. La consulta quedará de la siguiente forma:

```
SELECT "FechaVenta" AS "Fecha", "Total", "Nombre" AS "NombreCliente",
MAX("Cantidad"*"Unitario") AS "MontoMayor",
AVG ("Cantidad"*"Unitario") AS "Promedio"
FROM "Venta"
INNER JOIN "DetalleVenta" ON "NumeroFactura" = "Venta_numeroFactura"
INNER JOIN "Cliente" ON "Cliente_rut" = "Rut"
WHERE TO_CHAR ("FechaVenta", 'YYYY') = '2020'
GROUP BY "FechaVenta", "Total", "Nombre"
```

Al ejecutarla, no encontramos errores, y continuamos obteniendo la impresión de las columnas solicitadas.

	Fecha date	Total numeric	NombreCliente character varyin	MontoMayor numeric	Promedio numeric
1	2020-01-15	10000	Dafne	9000	5000.0000000000000000
2	2020-06-18	15000	Salim	15000	15000.0000000000000000
3	2020-07-05	20000	Salim	14000	6666.6666666666666667
4	2020-09-20	18000	Cecil	1800	1800.0000000000000000
5	2020-12-29	8000	Cecil	8000	8000.0000000000000000

Continuaremos mostrando las compras y ventas realizadas en el mes de enero del año 2020, con las columnas: Nombre, Total y Tipo.

Para poder obtener el resultado esperado, escribiremos la siguiente consulta para la tabla Compra:

```
SELECT "Nombre", "Total", 'Compra' AS "Tipo" FROM "Compra"
INNER JOIN "Proveedor" ON "Proveedor_Rut" = "Rut"
WHERE TO_CHAR("Fecha", 'YYYY-MM') = '2020-01'
```

	Nombre character varying	Total numeric	Tipo text
1	Faena San Nicolas	50000	Compra

Y para la tabla Venta escribiremos:

```
SELECT "Nombre", "Total", 'Venta' AS "Tipo" FROM "Venta"
INNER JOIN "Cliente" ON "Cliente_rut" = "Rut"
WHERE TO_CHAR ("FechaVenta", 'YYYY-MM') = '2020-01'
```

	Nombre character varying	Total numeric	Tipo text
1	Dafne	10000	Venta

Luego, para unir ambos resultados, agregaremos la palabra “unión”, que realizará la relación entre las tablas, quedando la consulta de la siguiente manera:

```
SELECT "Nombre", "Total", 'Compra' AS "Tipo" FROM "Compra"
INNER JOIN "Proveedor" ON "Proveedor_Rut" = "Rut"
WHERE TO_CHAR("Fecha", 'YYYY-MM') = '2020-01'
UNION
SELECT "Nombre", "Total", 'Venta' AS "Tipo" FROM "Venta"
INNER JOIN "Cliente" ON "Cliente_rut" = "Rut"
WHERE TO_CHAR ("FechaVenta", 'YYYY-MM') = '2020-01'
```

	Nombre character varying	Total numeric	Tipo text
1	Dafne	10000	Venta
2	Faena San Nicolas	50000	Compra

Finalmente, listaremos las ventas mensuales para el año 2020, incluyendo los atributos: Mes, TotalMes y Promedio.

```
SELECT TO_CHAR("FechaVenta", 'YYYY-MM') "Mes",
SUM ("Cantidad"*"Unitario") "TotalMes",
AVG ("Cantidad"*"Unitario") "Promedio"
FROM "Venta"
INNER JOIN "DetalleVenta" ON "NumeroFactura" = "Venta_numeroFactura"
WHERE TO_CHAR ("FechaVenta", 'YYYY') = '2020'
GROUP BY TO_CHAR ("FechaVenta", 'YYYY-MM')
```

La consulta resultante será la siguiente:

	Mes text	TotalMes numeric	Promedio numeric
1	2020-01	10000	5000.0000000000000000
2	2020-06	15000	15000.0000000000000000
3	2020-07	20000	6666.6666666666666667
4	2020-09	1800	1800.0000000000000000
5	2020-12	8000	8000.0000000000000000