UNIVERSIDAD NACIONAL DE SAN AGUSTIN

ESCUELA PROFESIONAL DE CIENCIA DE LA COMPUTACION

BASE DE DATOS II

GRUPO A

Trabajo Practico #3

Docente:
Mg. Franci Suni Lopez

Realizado por:
Pinto Medina, Brian Wilbert

1 de junio de 2020

Competencias del curso

Conoce, comprende e implementa los conceptos avanzados de base de datos referentes a estructuras de datos, concurrencia y base de datos distribuidas.

Competencias de la practica

Crea bases de datos y realiza consultas SQL para resolver problemas reales.

Conceptos Basicos

Se pide crear una base de datos bajo las siguientes condiciones, para cada tabla se presenta su nombre y las columnas de que consta entre paréntesis:

- Tabla provincias(codpro, nombre): Esta tabla almacena las provincias de España, cada una con su código de provincia(clave primaria) y su nombre.
- Tabla pueblos(codpue, nombre, codpro): Almacena los pueblos de España o, por lo menos, aquéllos donde tenemos clientes. Para cada pueblo se dispone de su código de pueblo (clave primaria), su nombre y el código de la provincia a la que pertenece (clave ajena).
- Tabla clientes(codcli, nombre, direccion, codpostal, codpue): Almacena información sobre los clientes de la empresa. Para cada cliente se dispone de su código de cliente (clave primaria), su nombre, su dirección, su código postal y el código de pueblo donde reside (clave ajena).
- Tabla vendedores(codven, nombre, direccion, codpostal, codpue, codjefe): Almacena información sobre los vendedores de la empresa. Para cada vendedor se dispone de su código de vendedor (clave primaria), su nombre, su dirección, su código postal, el código de pueblo

donde reside (clave ajena a la tabla pueblos) y el código de su jefe inmediato superior (clave ajena a la misma tabla de vendedores).

- Tabla vendedores(codven, nombre, direccion, codpostal, codpue, codjefe): Almacena información sobre los vendedores de la empresa. Para cada vendedor se dispone de su código de vendedor (clave primaria), su nombre, su dirección, su código postal, el código de pueblo donde reside (clave ajena a la tabla pueblos) y el código de su jefe inmediato superior (clave ajena a la misma tabla de vendedores).
- Tabla articulos(codart, descrip, precio, stock, stock_min): Almacena información sobre los artículos que ofrece la empresa y sus cantidades disponibles en el almacén (stocks). Para cada artículo se dispone de su código de artículo específico (clave primaria), su descripción, su precio actual, su stock y su stock mínimo, es decir, el valor umbral por debajo del cual se debe reponer.
- Tabla facturas(codfac, fecha, codcli, codven, iva, dto): Almacena toda la información sobre las facturas, excepto sus líneas. Como en cada factura el número de líneas es variable, todas las líneas de todas las facturas se almacenan juntas en otra tabla. Para cada factura en esta tabla se guarda su código de factura (clave primaria), su fecha, el código del cliente que ha realizado la compra (clave ajena), el código del vendedor que ha realizado la venta (clave ajena), el iva aplicado y el descuento global de la factura.
- Tabla lineas_fac(codfac, linea, cant, codart, precio, dto): Almacena información sobre las líneas de las facturas. Para cada línea se dispone del código de factura a la que pertenece (clave ajena), su número de línea, la cantidad de la línea, el código del artículo vendido (clave ajena), el precio al que se vende el artículo y el descuento que se debe aplicar en la línea. No hay que confundir este descuento, cuyo ámbito de aplicación es la línea, con el descuento global de la factura, el cual se halla obviamente en la tabla de facturas. La clave primaria de esta tabla va a ser la combinación del código de factura y del número de

línea pues, por ejemplo, sólo existirá una única tercera línea de la factura 15.

A continuación se muestran las tablas antes mencionadas:

Columna	¿Nulo?	Tipo de Dato
codpro	not null	VARCHAR2(2)
nombre	not null	VARCHAR2(30)

Tabla 1 – Tabla Provincias

Columna	¿Nulo?	Tipo de Dato
codpue	not null	VARCHAR2(5)
nombre	not null	VARCHAR2(40)
codpro	not null	VARCHAR2(2)

Tabla 2 – Tabla Pueblos

Columna	¿Nulo?	Tipo de Dato
codcli	not null	NUMBER(5)
nombre	not null	VARCHAR2(50)
direccion	not null	VARCHAR2(50)
codpostal		VARCHAR2(5)
codpue	not null	VARCHAR2(5)

Tabla 3 – Tabla Clientes

Columna	¿Nulo?	Tipo de Dato
codven	not null	NUMBER(5)
nombre	not null	VARCHAR2(50)
direccion	not null	VARCHAR2(50)
codpostal		VARCHAR2(6)
codpue	not null	VARCHAR2(5)
codjefe	not null	NUMBER(5)

Tabla 4 – Tabla Vendedores

Columna	¿Nulo?	Tipo de Dato
codart	not null	VARCHAR2(8)
descrip	not null	VARCHAR2(40)
precio	not null	NUMBER(7,2)
stock		NUMBER(6)
stock_min		NUMBER(6)

Tabla 5 – Tabla Artículos

Columna	¿Nulo?	Tipo de Dato
codfac	not null	NUMBER(6)
fecha	not null	DATE
codcli		NUMBER(5)
codven		NUMBER(5)
iva		NUMBER(2)
dto		NUMBER(2)

Tabla 6 – Tabla Facturas

Columna	¿Nulo?	Tipo de Dato
codfac	not null	NUMBER(6)
linea	notnull	NUMBER(2)
cant		NUMBER(5)
codart	not null	VARCHAR2(8)
precio		NUMBER(7,2)
dto		NUMBER(2)

Tabla 7 – Tabla Lineas_fac

Implementacion Base de Datos

Antes de comenzar con la resolucion de la lista de ejercicios seria bueno mostrar un poco sobre la creacion de la base de datos. Para comenzar resaltar que los items y las restricciones ya venian en las tablas mostradas anteriormente, asi que nos limitaremos a seguir el modelo.

```
1  DROP DATABASE IF EXISTS Empresa;
2  CREATE SCHEMA Empresa DEFAULT CHARACTER SET utf8;
3  USE Empresa;
4
5  DROP TABLE IF EXISTS Provincias;
6  CREATE TABLE Provincias(
7          codpro VARCHAR(2) NOT NULL PRIMARY KEY,
8          nombre VARCHAR(30) NOT NULL
9  );
10
11 DROP TABLE IF EXISTS Pueblos;
12 CREATE TABLE Pueblos(
13         codpue VARCHAR(5) NOT NULL PRIMARY KEY,
14         nombre VARCHAR(40) NOT NULL,
15         codpro VARCHAR(2) NOT NULL,
16         CONSTRAINT `fk_pueblo_prov`
17         FOREIGN KEY(codpro)
18         REFERENCES Provincias(codpro) ON UPDATE CASCADE ON DELETE
               RESTRICT
19 );
20
21 DROP TABLE IF EXISTS Clientes;
22 CREATE TABLE Clientes(
23         codcli DECIMAL(5,0) UNSIGNED NOT NULL PRIMARY KEY,
24         nombre VARCHAR(50) NOT NULL,
25         direccion VARCHAR(50) NOT NULL,
26         codpostal VARCHAR(5),
27         codpue VARCHAR(5) NOT NULL,
28         CONSTRAINT `fk_cliente_pue`
29         FOREIGN KEY(codpue)
30         REFERENCES Pueblos(codpue) ON UPDATE CASCADE ON DELETE
              RESTRICT
31 );
32
```

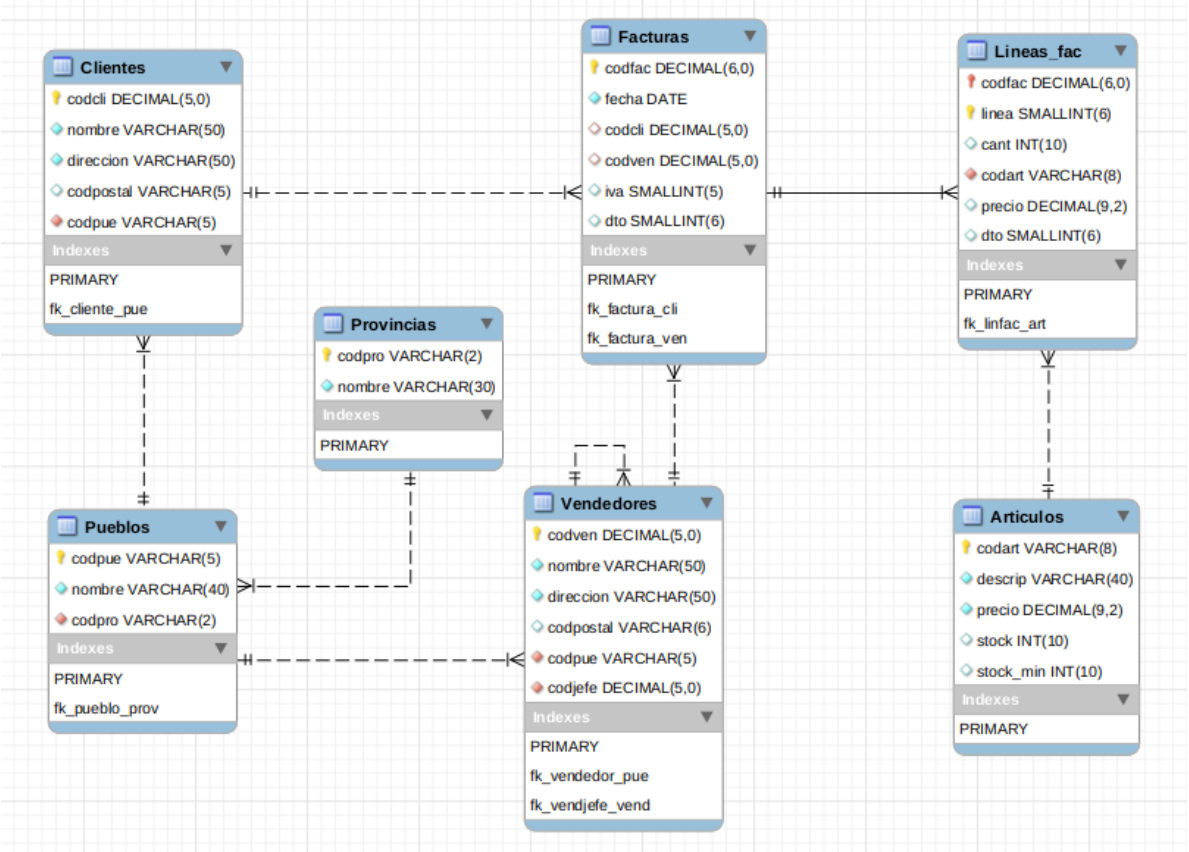
```
33 DROP TABLE IF EXISTS Vendedores;
34 CREATE TABLE Vendedores(
35         codven DECIMAL(5,0) UNSIGNED NOT NULL PRIMARY KEY,
36         nombre VARCHAR(50) NOT NULL,
37         direccion VARCHAR(50) NOT NULL,
38         codpostal VARCHAR(6),
39         codpue VARCHAR(5) NOT NULL,
40         codjefe DECIMAL(5,0) UNSIGNED NOT NULL,
41         CONSTRAINT `fk_vendedor_pue`
42         FOREIGN KEY(codpue)
43         REFERENCES Pueblos(codpue) ON UPDATE CASCADE ON DELETE
               RESTRICT,
44         CONSTRAINT `fk_vendjefe_vend`
45         FOREIGN KEY(codjefe)
46         REFERENCES Vendedores(codven) ON UPDATE CASCADE ON DELETE
               RESTRICT
47 );
48
49 DROP TABLE IF EXISTS Articulos;
50 CREATE TABLE Articulos(
51         codart VARCHAR(8) NOT NULL PRIMARY KEY,
52         descrip VARCHAR(40) NOT NULL,
53         precio DECIMAL(9,2) UNSIGNED NOT NULL,
54         stock INT UNSIGNED,
55         stock_min INT UNSIGNED
56 );
57
58 DROP TABLE IF EXISTS Facturas;
59 CREATE TABLE Facturas(
60         codfac DECIMAL(6,0) UNSIGNED NOT NULL PRIMARY KEY,
61         fecha DATE NOT NULL,
62         codcli DECIMAL(5,0) UNSIGNED,
63         codven DECIMAL(5,0) UNSIGNED,
64         iva SMALLINT UNSIGNED,
65         dto SMALLINT,
66         CONSTRAINT `fk_factura_cli`
67         FOREIGN KEY(codcli)
68         REFERENCES Clientes(codcli) ON UPDATE CASCADE ON DELETE
               RESTRICT,
69         CONSTRAINT `fk_factura_ven`
70         FOREIGN KEY(codven)
71         REFERENCES Vendedores(codven) ON UPDATE CASCADE ON DELETE
               RESTRICT
```

```
72 );
73
74 DROP TABLE IF EXISTS Lineas_fac;
75 CREATE TABLE Lineas_fac(
76         codfac DECIMAL(6,0) UNSIGNED NOT NULL,
77         linea SMALLINT UNSIGNED NOT NULL,
78         cant INT UNSIGNED,
79         codart VARCHAR(8) NOT NULL,
80         precio DECIMAL(9,2) UNSIGNED,
81         dto SMALLINT,
82         PRIMARY KEY(codfac,linea),
83         CONSTRAINT `fk_linfac_fac`
84         FOREIGN KEY(codfac)
85         REFERENCES Facturas(codfac) ON UPDATE CASCADE ON DELETE
              RESTRICT,
86         CONSTRAINT `fk_linfac_art`
87         FOREIGN KEY(codart)
88         REFERENCES Articulos(codart) ON UPDATE CASCADE ON DELETE
              RESTRICT
89 );
```

NOTA: Puede encontrar este script con el nombre de **Empresa.sql**

Diagrama Fisico de la Base de Datos

Una vez cargada la base de datos podemos ver si se realizo correctamente por medio de la herramienta Reverse Engine, lo cual nos generara el Diagrama Fisico de nuestra base de datos Empresa:



Llenando la Base de Datos

Habiendo revisado que todas las conexiones podemos proceder a llenar nuestra base de datos, al ser una cantidad de datos extensa es mejor utilizar procedimientos para no estar escribiendo codigo de mas. Para lo cual crearemos nuestros procedimientos de la siguiente manera:

```
USE Empresa;
DROP PROCEDURE IF EXISTS insertarProvincia;
DELIMITER $$
CREATE PROCEDURE insertarProvincia(
        IN _codpro VARCHAR(2),
        IN _nombre VARCHAR(30))
BEGIN
        INSERT INTO Provincias(codpro, nombre)
        VALUES(_codpro, _nombre);
END$$

DROP PROCEDURE IF EXISTS insertarPueblo;
DELIMITER $$
CREATE PROCEDURE insertarPueblo(
        IN _codpue VARCHAR(5),
        IN _nombre VARCHAR(40),
        IN _codpro VARCHAR(2))
BEGIN
        INSERT INTO Pueblos(codpue, nombre, codpro)
        VALUES(_codpue, _nombre, _codpro);
END$$

DROP PROCEDURE IF EXISTS insertarCliente;
DELIMITER $$
CREATE PROCEDURE insertarCliente(
        IN _codcli DECIMAL(5,0) UNSIGNED,
        IN _nombre VARCHAR(50),
        IN _direccion VARCHAR(50),
        IN _codpostal VARCHAR(5),
        IN _codpue VARCHAR(5))
BEGIN
        INSERT INTO Clientes(codcli, nombre, direccion, codpostal
           , codpue)
        VALUES(_codcli, _nombre, _direccion, _codpostal, _codpue)
           ;
END$$

DROP PROCEDURE IF EXISTS insertarVendedor;
DELIMITER $$
CREATE PROCEDURE insertarVendedor(
        IN _codven DECIMAL(5,0) UNSIGNED,
        IN _nombre VARCHAR(50),
```

```
41          IN _direccion VARCHAR(50),
42          IN _codpostal VARCHAR(6),
43          IN _codpue VARCHAR(5),
44          IN _codjefe DECIMAL(5,0) UNSIGNED)
45  BEGIN
46          INSERT INTO Vendedores(codven, nombre, direccion,
               codpostal, codpue, codjefe)
47          VALUES(_codven, _nombre, _direccion, _codpostal, _codpue,
                _codjefe);
48  END$$
49
50  DROP PROCEDURE IF EXISTS insertarArticulo;
51  DELIMITER $$
52  CREATE PROCEDURE insertarArticulo(
53          IN _codart VARCHAR(8),
54          IN _descrip VARCHAR(40),
55          IN _precio DECIMAL(9,2) UNSIGNED,
56          IN _stock INT UNSIGNED,
57          IN _stock_min INT UNSIGNED)
58  BEGIN
59          INSERT INTO Articulos(codart, descrip, precio, stock,
               stock_min)
60          VALUES(_codart, _descrip, _precio, _stock, _stock_min);
61  END$$
62
63  DROP PROCEDURE IF EXISTS insertarFactura;
64  DELIMITER $$
65  CREATE PROCEDURE insertarFactura(
66          IN _codfac DECIMAL(6,0) UNSIGNED,
67          IN _fecha DATE,
68          IN _codcli DECIMAL(5,0) UNSIGNED,
69          IN _codven DECIMAL(5,0) UNSIGNED,
70          IN _iva SMALLINT UNSIGNED,
71          IN _dto SMALLINT)
72  BEGIN
73          INSERT INTO Facturas(codfac, fecha, codcli, codven, iva,
               dto)
74          VALUES(_codfac, _fecha, _codcli, _codven, _iva, _dto);
75  END$$
76
77  DROP PROCEDURE IF EXISTS insertarLineaFac;
78  DELIMITER $$
79  CREATE PROCEDURE insertarLineaFac(
```

```
80          IN _codfac DECIMAL(6,0) UNSIGNED,
81          IN _linea SMALLINT UNSIGNED,
82          IN _cant INT UNSIGNED,
83          IN _codart VARCHAR(8),
84          IN _dto SMALLINT)
85  BEGIN
86          DECLARE _precio DECIMAL(9,2) UNSIGNED;
87          SET _precio = (SELECT precio FROM Articulos WHERE codart
                = _codart);
88          INSERT INTO Lineas_fac(codfac, linea, cant, codart,
                precio, dto)
89          VALUES(_codfac, _linea, _cant, _codart, _precio, _dto);
90  END$$
```

Como se pudo ver anteriormente se realizo un procedimiento por tabla. El anterior script lo puede encontrar como **InsertProceduresEmpresa.sql**

Ahora simplemente procedemos a llenar los datos, dare 1 ejemplo por cada caso debido a que es una extensa cantidad de datos.

```
1  -- INSERTAR PROVINCIAS
2  CALL insertarProvincia('AQ','Arequipa');
3  -- INSERTAR PUEBLOS
4  CALL insertarPueblo('AQP00','Arequipa','AQ');
5  -- INSERTAR CLIENTES
6  CALL insertarCliente(10000,'Nadinne','Av. Jesus #2008','40000','
      AQP00');
7  -- INSERTAR VENDEDORES
8  CALL insertarVendedor(10000,'Brian','Urb. Manco Capac F
      -34','40000','AQP00',10000);
9  -- INSERTAR ARTICULOS
10 CALL insertarArticulo('LICUAD00','Licuadora',299.99,25,10);
11 -- INSERTAR FACTURAS
12 CALL insertarFactura(100000,'2019-01-01',10000,10000,18,NULL);
13 -- INSERTAR LINEAS DE FACTURACION
14 CALL insertarLineaFac(100000,1,2,'LICUAD00',NULL);
```

NOTA: Usted puede encontrar el script completo en **FillDB.sql**.

Tabla Provincias

#	codpro	nombre
1	AL	Alicante
2	AQ	Arequipa
3	BA	Barcelona
4	CA	Castellon
5	GE	Gerona
6	VA	Valencia
*	NULL	NULL

Tabla Pueblos

#	codpue	nombre	codpro
1	ALI00	Guadalest	AL
2	ALI01	Calpe	AL
3	AQP00	Arequipa	AQ
4	BAR00	Tavertet	BA
5	BAR01	Cardona	BA
6	GER00	Monells	GE
7	GER01	Santa Pau	GE
8	VAL00	Sagunto	VA
9	VAL01	Miramar	VA
*	NULL	NULL	NULL

Tabla Articulos

#	codart	descrip	precio	stock	stock_min
1	BATIDO00	Batidora	40.50	40	15
2	CAFETE00	Cafetera	70.99	40	15
3	COCINA00	Cocina	549.99	20	5
4	EXTRAC00	Extractor	399.20	35	10
5	LAVADO00	Lavadora	849.00	20	5
6	LICUAD00	Licuadora	299.99	25	10
7	PICADO00	Picadora	169.90	35	10
8	REFRIG00	Refrigerador	979.99	20	5
9	SECADO00	Secadora	699.19	25	5
*	NULL	NULL	NULL	NULL	NULL

Tabla Clientes

#	codcli	nombre	direccion	codposta	codpue
1	10000	Nadinne	Av. Jesus #2008	40000	AQP00
2	10001	Lucia	Av. Flores #205	40001	ALI00
3	10002	Marina	Calle Roma #…	40001	ALI00
4	10003	Jose	Urb. Casa Bl…	40002	ALI01
5	10004	Pedro	Av. Paris #1562	40002	ALI01
6	10005	Gabriel	Calle Segovia…	40003	BAR00
7	10006	Sama…	Urb. Electrica…	40003	BAR00
8	10007	Carla	Av. Venecia #…	40004	BAR01
9	10008	Carmen	Calle Azul #456	40004	BAR01
10	10009	Mario	Urb. Colon C-…	40005	GER00
11	10010	David	Av. Girasoles …	40005	GER00
12	10011	Luis	Calle Libertad…	40006	GER01
13	10012	Vicente	Urb. Indepen…	40006	GER01
14	10013	Silvia	Av. Colonial #…	40007	VAL00
15	10014	Javier	Calle Tokio #8…	40007	VAL00
16	10015	Steph…	Urb. Monterri…	40008	VAL01
17	10016	Carlos	Av. Ejercito #…	40008	VAL01
*	NULL	NULL	NULL	NULL	NULL

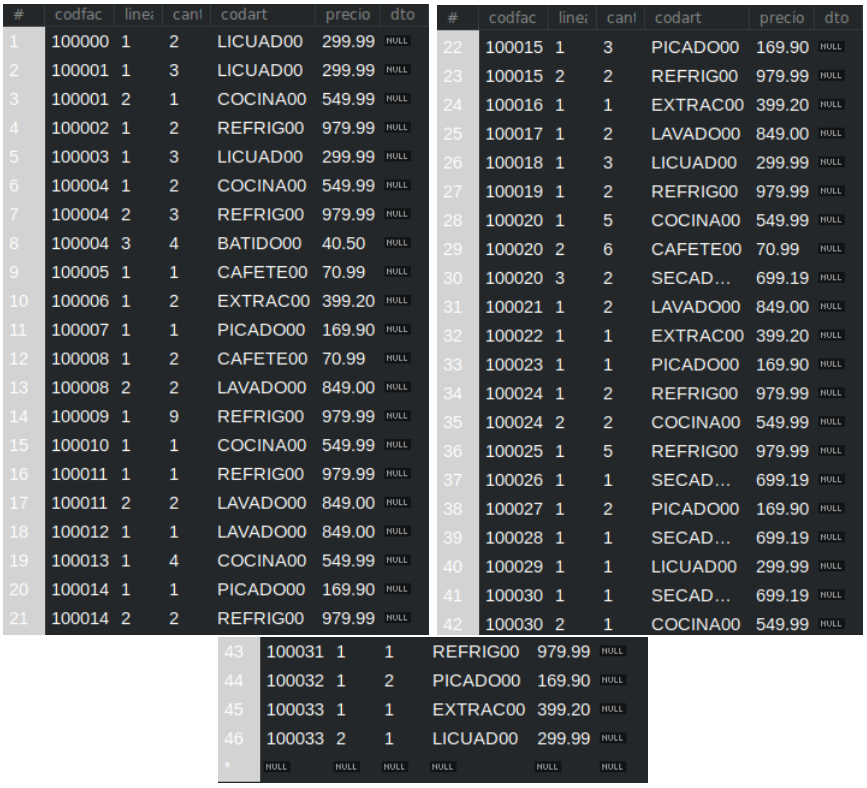
Tabla Vendedores

#	codver	nombre	direccion	codposta	codpue	codjefe
1	10000	Brian	Urb. Manco Capac F-34	40000	AQP00	10000
2	10001	Liza	Av. Albacete #1253	40001	ALI00	10000
3	10002	Charlie	Calle Alcala #152	40001	ALI00	10001
4	10003	Joel	Urb. Daganzo E-15	40002	ALI01	10001
5	10004	Juan	Av. Algeciras #2145	40002	ALI01	10001
6	10005	Roberto	Calle Larios #451	40003	BAR00	10000
7	10006	Paco	Urb. Fragua S-4	40003	BAR00	10005
8	10007	Bruno	Av. Lerida #246	40004	BAR01	10005
9	10008	Hugo	Calle Betis #748	40004	BAR01	10005
10	10009	Magda	Urb. Fuenfria K-3	40005	GER00	10000
11	10010	Rosa	Av. Sevilla #1211	40005	GER00	10009
12	10011	Flore…	Calle Rabat #321	40006	GER01	10009
13	10012	Susan	Urb. Raimundo Lulio …	40006	GER01	10009
14	10013	Cecilia	Av. Nenufares #1245	40007	VAL00	10000
15	10014	Karina	Calle Zafiro #653	40007	VAL00	10013
16	10015	Lourdes	Urb. Zaragoza P-23	40008	VAL01	10013
17	10016	Scarlet	Av. Cid #1954	40008	VAL01	10013
*	NULL	NULL	NULL	NULL	NULL	NULL

Tabla Facturas

#	codfac	fecha	codcli	codver	iva	dto
1	100000	2019-01-01	10000	10000	18	NULL
2	100001	2019-01-05	10001	10001	18	NULL
3	100002	2019-01-15	10002	10001	18	NULL
4	100003	2019-01-30	10002	10002	18	NULL
5	100004	2019-02-02	10002	10002	18	NULL
6	100005	2019-02-12	10003	10002	18	NULL
7	100006	2019-03-04	10003	10003	18	NULL
8	100007	2019-03-14	10004	10003	18	NULL
9	100008	2019-03-25	10004	10004	18	NULL
10	100009	2019-04-24	10005	10005	18	NULL
11	100010	2019-04-30	10006	10005	18	NULL
12	100011	2019-05-02	10006	10005	18	NULL
13	100012	2019-05-10	10006	10005	18	NULL
14	100013	2019-05-15	10007	10006	18	NULL
15	100014	2019-05-22	10008	10007	18	NULL
16	100015	2019-05-27	10008	10008	18	NULL
17	100016	2019-06-06	10009	10008	18	NULL
18	100017	2019-06-07	10009	10009	18	NULL
19	100018	2019-06-16	10009	10009	18	NULL
20	100019	2019-06-24	10010	10009	18	NULL
21	100020	2019-07-06	10010	10010	18	NULL
22	100021	2019-07-14	10011	10011	18	NULL
23	100022	2019-08-03	10011	10011	18	NULL
24	100023	2019-08-26	10011	10012	18	NULL
25	100024	2019-08-28	10012	10013	18	NULL
26	100025	2019-09-04	10012	10013	18	NULL
27	100026	2019-09-09	10013	10013	18	NULL
28	100027	2019-10-12	10013	10014	18	NULL
29	100028	2019-10-29	10014	10014	18	NULL
30	100029	2019-11-10	10014	10015	18	NULL
31	100030	2019-11-17	10015	10015	18	NULL
32	100031	2019-11-18	10016	10015	18	NULL
33	100032	2019-12-02	10016	10015	18	NULL
34	100033	2019-12-10	10016	10016	18	NULL
*	NULL	NULL	NULL	NULL	NULL	NULL

Tabla Lineas_fac



#	codfac	linea	cant	codart	precio	dto
1	100000	1	2	LICUAD00	299.99	NULL
2	100001	1	3	LICUAD00	299.99	NULL
3	100001	2	1	COCINA00	549.99	NULL
4	100002	1	2	REFRIG00	979.99	NULL
5	100003	1	3	LICUAD00	299.99	NULL
6	100004	1	2	COCINA00	549.99	NULL
7	100004	2	3	REFRIG00	979.99	NULL
8	100004	3	4	BATIDO00	40.50	NULL
9	100005	1	1	CAFETE00	70.99	NULL
10	100006	1	2	EXTRAC00	399.20	NULL
11	100007	1	1	PICADO00	169.90	NULL
12	100008	1	2	CAFETE00	70.99	NULL
13	100008	2	2	LAVADO00	849.00	NULL
14	100009	1	9	REFRIG00	979.99	NULL
15	100010	1	1	COCINA00	549.99	NULL
16	100011	1	1	REFRIG00	979.99	NULL
17	100011	2	2	LAVADO00	849.00	NULL
18	100012	1	1	LAVADO00	849.00	NULL
19	100013	1	4	COCINA00	549.99	NULL
20	100014	1	1	PICADO00	169.90	NULL
21	100014	2	2	REFRIG00	979.99	NULL
22	100015	1	3	PICADO00	169.90	NULL
23	100015	2	2	REFRIG00	979.99	NULL
24	100016	1	1	EXTRAC00	399.20	NULL
25	100017	1	2	LAVADO00	849.00	NULL
26	100018	1	3	LICUAD00	299.99	NULL
27	100019	1	2	REFRIG00	979.99	NULL
28	100020	1	5	COCINA00	549.99	NULL
29	100020	2	6	CAFETE00	70.99	NULL
30	100020	3	2	SECAD…	699.19	NULL
31	100021	1	2	LAVADO00	849.00	NULL
32	100022	1	1	EXTRAC00	399.20	NULL
33	100023	1	1	PICADO00	169.90	NULL
34	100024	1	2	REFRIG00	979.99	NULL
35	100024	2	2	COCINA00	549.99	NULL
36	100025	1	5	REFRIG00	979.99	NULL
37	100026	1	1	SECAD…	699.19	NULL
38	100027	1	2	PICADO00	169.90	NULL
39	100028	1	1	SECAD…	699.19	NULL
40	100029	1	1	LICUAD00	299.99	NULL
41	100030	1	1	SECAD…	699.19	NULL
42	100030	2	1	COCINA00	549.99	NULL
43	100031	1	1	REFRIG00	979.99	NULL
44	100032	1	2	PICADO00	169.90	NULL
45	100033	1	1	EXTRAC00	399.20	NULL
46	100033	2	1	LICUAD00	299.99	NULL
*	NULL	NULL	NULL	NULL	NULL	NULL

A continuacion se procedera a realizar la lista de ejercicios.

Lista de Ejercicios

1. Escribir una consulta que obtenga el código y nombre de cada cliente y el número de facturas que ha realizado durante el año pasado.

```
1  -- EJERCICIO #1
2  SELECT codcli 'Codigo_Cliente', cliente.nombre 'Nombre',
       COUNT(*) 'N Facturas' FROM Clientes cliente
3  JOIN Facturas factura USING (codcli) WHERE
4  EXTRACT( year FROM factura.fecha ) = EXTRACT( year from
      CURRENT_DATE ) - 1
5  GROUP BY codcli, cliente.nombre;
```

#	Codigo_Cliente	Nombre	Nº Facturas
1	10000	Nadinne	1
2	10001	Lucia	1
3	10002	Marina	3
4	10003	Jose	2
5	10004	Pedro	2
6	10005	Gabriel	1
7	10006	Samantha	3
8	10007	Carla	1
9	10008	Carmen	2
10	10009	Mario	3
11	10010	David	2
12	10011	Luis	3
13	10012	Vicente	2
14	10013	Silvia	2
15	10014	Javier	2
16	10015	Stephanie	1
17	10016	Carlos	3

2. Escribir una consulta que obtenga el código de factura, la fecha y el importe (sin considerar descuentos ni impuestos) de cada una de las facturas.

```
1  -- EJERCICIO #2
2  SELECT codfac 'Codigo factura', factura.fecha 'Fecha',
      SUM(linea_fac.precio * linea_fac.cant) 'Precio Total'
       FROM Facturas factura
3  JOIN Lineas_fac linea_fac USING (codfac)
4  GROUP BY codfac, factura.fecha;
```

#	Codigo factura	Fecha	Precio Total
1	100000	2019-01-01	599.98
2	100001	2019-01-05	1449.96
3	100002	2019-01-15	1959.98
4	100003	2019-01-30	899.97
5	100004	2019-02-02	4201.95
6	100005	2019-02-12	70.99
7	100006	2019-03-04	798.40
8	100007	2019-03-14	169.90
9	100008	2019-03-25	1839.98
10	100009	2019-04-24	8819.91
11	100010	2019-04-30	549.99
12	100011	2019-05-02	2677.99
13	100012	2019-05-10	849.00
14	100013	2019-05-15	2199.96
15	100014	2019-05-22	2129.88
16	100015	2019-05-27	2469.68
17	100016	2019-06-06	399.20
18	100017	2019-06-07	1698.00
19	100018	2019-06-16	899.97
20	100019	2019-06-24	1959.98
21	100020	2019-07-06	4574.27
22	100021	2019-07-14	1698.00
23	100022	2019-08-03	399.20
24	100023	2019-08-26	169.90
25	100024	2019-08-28	3059.96
26	100025	2019-09-04	4899.95
27	100026	2019-09-09	699.19
28	100027	2019-10-12	339.80
29	100028	2019-10-29	699.19
30	100029	2019-11-10	299.99
31	100030	2019-11-17	1249.18
32	100031	2019-11-18	979.99
33	100032	2019-12-02	339.80
34	100033	2019-12-10	699.19

3. Escribir una sentencia que calcule el código y nombre de cada vendedor y su facturación durante el año pasado.

```
-- EJERCICIO #3
SELECT codven 'Codigo Vendedor', vendedor.nombre 'Nombre
   ', SUM(linea_fac.precio*linea_fac.cant) 'Facturacion'
    FROM Vendedores vendedor
JOIN Facturas factura USING (codven)
JOIN Lineas_fac linea_fac USING (codfac)
WHERE EXTRACT(year FROM factura.fecha) = EXTRACT(year
   FROM CURRENT_DATE) - 1
GROUP BY codven, vendedor.nombre;
```

#	Codigo Vendedor	Nombre	Facturacion
1	10000	Brian	599.98
2	10001	Liza	3409.94
3	10002	Charlie	5172.91
4	10003	Joel	968.30
5	10004	Juan	1839.98
6	10005	Roberto	12896.89
7	10006	Paco	2199.96
8	10007	Bruno	2129.88
9	10008	Hugo	2868.88
10	10009	Magda	4557.95
11	10010	Rosa	4574.27
12	10011	Florencia	2097.20
13	10012	Susan	169.90
14	10013	Cecilia	8659.10
15	10014	Karina	1038.99
16	10015	Lourdes	2868.96
17	10016	Scarlet	699.19

4. Escribir una sentencia que calcule el número de unidades vendidas en cada provincia durante el año pasado.

```
SELECT provincia.nombre 'Provincia', SUM(linea_fac.cant)
    'Articulos Vendidos' FROM Provincias provincia
JOIN Pueblos pueblo USING (codpro)
JOIN Clientes cliente USING (codpue)
JOIN Facturas factura USING (codcli)
JOIN Lineas_fac linea_fac USING (codfac)
WHERE EXTRACT(year FROM factura.fecha) = EXTRACT(year
   FROM CURRENT_DATE) - 1
GROUP BY provincia.nombre;
```

#	Provincia	Articulos Vendidos
1	Alicante	26
2	Arequipa	2
3	Barcelona	26
4	Gerona	34
5	Valencia	12

5. Escribir una consulta que obtenga el código y nombre de aquellos clientes que han sido atendidos alguna vez por vendedores residentes en otras provincias.

```
-- EJERCICIO #5
SELECT DISTINCT codcli, cliente.nombre FROM Clientes
   cliente
JOIN Pueblos pueblo1 ON cliente.codpue = pueblo1.codpue
JOIN Facturas facturas USING (codcli)
JOIN Vendedores vendedor USING (codven)
JOIN Pueblos pueblo2 ON vendedor.codpue = pueblo2.codpue
WHERE pueblo1.codpro <> pueblo2.codpro;
```

#	Codigo Cliente	Nombre
1	10009	Mario
2	10012	Vicente

6. Escribir una consulta que obtenga el código y nombre de aquellos clientes de la provincia de Valencia que tienen alguna factura con 2 líneas o más.

   NOTA: Se cambio un poco el ejercicio debido a que en mi BD no tenia ninguna factura con mas de 10 lineas en la provincia de Valencia, por lo cual se procedio a limitarlo a 2 lineas o mas.

```
1      -- EJERCICIO #6
2      SELECT DISTINCT codcli 'Codigo Cliente', Clientes.nombre
           'Nombre' FROM Lineas_fac
3      JOIN Facturas USING (codfac)
4      JOIN Clientes USING (codcli)
5      JOIN Pueblos USING (codpue)
6      JOIN Provincias USING (codpro)
7      WHERE Provincias.nombre = 'Valencia'
8      GROUP BY codfac, codcli, Clientes.nombre
9      HAVING COUNT(*) >= 2;
```

#	Codigo Cliente	Nombre
1	10015	Stephanie
2	10016	Carlos

7. Escribir una consulta que obtenga el código y descripción de aquellos artículos que durante el año pasado se vendieron siempre en varios (más de uno) meses consecutivos. Por ejemplo, artículos vendidos en marzo, abril y mayo, pero no aquéllos vendidos en agosto y diciembre.

```
1      SELECT codart 'Articulo', Articulos.descrip 'Descripcion
          ' FROM Articulos
2      JOIN Lineas_fac USING (codart)
3      JOIN Facturas USING (codfac)
4      WHERE EXTRACT(year FROM Facturas.fecha) = EXTRACT(year
          FROM CURRENT_DATE) - 1
5      GROUP BY codart, Articulos.descrip
6      HAVING COUNT(DISTINCT EXTRACT(month FROM Facturas.fecha)
          ) > 1
```

```
7       AND MAX(EXTRACT(month FROM Facturas.fecha)) - MIN(
           EXTRACT(month FROM Facturas.fecha)) + 1 =
8       COUNT(DISTINCT EXTRACT(month FROM Facturas.fecha));
```

#	Articulo	Descripcion

8. Escribir una consulta que muestre el código y nombre de aquellos clientes de la provincia de Castellón o Barcelona que han facturado más de 6000 euros.

   NOTA: En mi base de datos no tengo ninguna persona que halla facturado mas de 6000 soles en Castellon, pero tengo una en Barcelona que lo ha hecho por lo cual se cambio la pregunta a Castellon o Barcelona.

```
1       -- EJERCICIO 8
2       SELECT codcli 'Codigo Cliente', Clientes.nombre 'Nombre'
            FROM Pueblos
3       JOIN Clientes USING (codpue)
4       JOIN Facturas USING (codcli)
5       JOIN Lineas_fac USING (codfac)
6       JOIN Provincias USING (codpro)
7       WHERE Provincias.nombre = 'Barcelona' OR Provincias.
           nombre = 'Castellon'
8       GROUP BY codcli, Clientes.nombre
9       HAVING SUM( cant * precio ) > 6000.00;
```

#	Codigo Cliente	Nombre
1	10005	Gabriel

9. Escribir una consulta que calcule la facturación máxima realizada por los clientes de la provincia de Castellón en un mes del año pasado.

   NOTA: Se cambio Castallon por Barcelona para que se puede apreciar el funcionamiento del query.

```
1      -- EJERCICIO 9
2      SELECT MAX(SUM(Lineas_fac.cant * Lineas_fac.precio)) '
          Facturacion Maxima' FROM Pueblos
3      JOIN Clientes USING (codpue)
4      JOIN Facturas USING (codcli)
5      JOIN Lineas_fac USING (codfac)
6      JOIN Provincias USING (codpro)
7      WHERE Provincias.nombre = 'Barcelona'
8      AND EXTRACT(year FROM Facturas.fecha) = EXTRACT(year
          FROM CURRENT_DATE) - 1
9      GROUP BY codcli, EXTRACT(year FROM Facturas.fecha);
```

#	Facturacion Maxima
1	8819.91

10. Escribir una consulta que obtenga el nombre de cada jefe y el número de vendedores que dependen de él (se considerará como jefe a aquel vendedor que es jefe de al menos otro vendedor).

```
1      -- EJERCICIO 10
2      SELECT jefe.nombre 'Nombre Jefe', COUNT(*) 'N Vendedores
          ' FROM Vendedores jefe
3      JOIN Vendedores vendedor ON vendedor.codjefe = jefe.
          codven
4      GROUP BY jefe.nombre;
```

#	Nombre Jefe	Nº Vendedores
1	Brian	5
2	Cecilia	3
3	Liza	3
4	Magda	3
5	Roberto	3

11. Para aquellos clientes de la Comunidad Valenciana cuyo nombre comienza por la misma letra que comienza el nombre del pueblo en el que residen, mostrar el nombre del cliente, el nombre del pueblo y el número de artículos distintos comprados durante el último trimestre del año pasado. En el listado final sólo deben aparecer aquellos clientes cuya facturación en el mismo periodo superó los 6000 euros, sin considerar impuestos ni descuentos.

```
1     -- EJERCICIO 11
2     SELECT Clientes.nombre 'Nombre Cliente', Pueblos.nombre
         'Nombre Pueblo', COUNT(DISTINCT Lineas_fac.codart) '
         Nombre Articulo' FROM Clientes
3     JOIN Pueblos USING (codpue)
4     JOIN Facturas USING (codcli)
5     JOIN Lineas_fac USING (codfac)
6     WHERE Pueblos.codpro = 'VA'
7     AND SUBSTR( Clientes.nombre, 1, 1 ) = SUBSTR( Pueblos.
         nombre, 1, 1 )
8     AND EXTRACT(month FROM Facturas.fecha) = 10 || EXTRACT(
         month FROM Facturas.fecha) = 11 || EXTRACT(month FROM
          Facturas.fecha) = 12
9     AND EXTRACT(year FROM Facturas.fecha) = EXTRACT(year
         FROM CURRENT_DATE) - 1
10    GROUP BY codcli, Clientes.nombre, Pueblos.nombre
11    HAVING SUM(Lineas_fac.cant * Lineas_fac.precio) > 6000;
```

#	Nombre Cliente	Nombre Pueblo	Nombre Articulo

No se tiene ningun cliente cuyo nombre comience con la letra del pueblo.

12. Artículos cuya descripción consta de más de 15 letras o dígitos que han sido comprados por más de 5 clientes distintos de la provincia de Castellón durante los últimos diez días del año pasado. En el listado final se debe mostrar el artículo y su descripción.

```
1     -- EJERCICIO 12
```

```
2       SELECT codart 'Codigo Articulo', Articulos.descrip '
           Descripcion' FROM Articulos
3       JOIN Lineas_fac USING (codart)
4       JOIN Facturas USING (codfac)
5       JOIN Clientes USING (codcli)
6       JOIN Pueblos USING (codpue)
7       JOIN Provincias USING (codpro)
8       WHERE LENGTH( Articulos.descrip ) > 15
9       AND Provincias.nombre = 'Castellon'
10      AND EXTRACT(year FROM Facturas.fecha) = EXTRACT(year
           FROM CURRENT_DATE) - 1
11      AND EXTRACT(month FROM Facturas.fecha) = 12
12      AND EXTRACT(day FROM Facturas.fecha) > 21
13      GROUP BY codart, Articulos.descrip
14      HAVING COUNT(DISTINCT codcli) > 5;
```

#	Codigo Articulo	Descripcion

No se tiene ningun articulo con mas de 15 caracteres

13. Código y nombre de aquellos pueblos cuya primera letra del nombre es la misma que la primera letra del nombre de la provincia, en los que residen más de 3 clientes y en los que se han facturado más de 1000 unidades en total durante el tercer trimestre del año pasado.

```
1       -- EJERCICIO 13
2       SELECT codpue 'Codigo Pueblo', Pueblos.nombre 'Nombre'
           FROM Pueblos
3       JOIN Provincias USING (codpro)
4       JOIN Clientes USING (codpue)
5       JOIN Facturas USING (codcli)
6       JOIN Lineas_fac USING (codfac)
7       WHERE UPPER(SUBSTR(Pueblos.nombre,1,1)) = UPPER(SUBSTR(
           Provincias.nombre,1,1))
8       AND EXTRACT(year FROM Facturas.fecha) = EXTRACT(year
           FROM CURRENT_DATE) - 1
9       AND EXTRACT(month FROM Facturas.fecha) = 10 || EXTRACT(
           month FROM Facturas.fecha) = 11 || EXTRACT(month FROM
            Facturas.fecha) = 12
```

```
10      GROUP BY codpue, Pueblos.nombre
11      HAVING COUNT(DISTINCT codcli) > 3
12      AND SUM(Lineas_fac.cant) > 1000;
```

#	Codigo Pueblo	Nombre

Nadie compro mas de 1000 unidades.

14. Para aquellos vendedores cuyo primer o segundo apellido terminan con ‘EZ’ (se asume que ningún nombre de pila termina con dicho sufijo), mostrar el número de clientes de su misma provincia a los que ha realizado alguna venta durante los 10 últimos días del año pasado. Mostrar el código y nombre del vendedor, además del citado número de clientes.

```
1       -- EJERCICIO 14
2       SELECT codven 'Codigo Vendedor', Vendedores.nombre '
           Nombre', COUNT(DISTINCT codcli) [NU+FFKCl]iente'FROM
           Vendedores
3       JOIN Pueblos pueblo1 ON Vendedores.codpue = pueblo1.
           codpue
4       JOIN Facturas USING (codven)
5       JOIN Clientes USING (codcli)
6       JOIN Pueblos pueblo2 ON Clientes.codpue = pueblo2.codpue
7       WHERE EXTRACT(year FROM Facturas.fecha) = EXTRACT(year
           FROM CURRENT_DATE) - 1
8       AND EXTRACT(month FROM Facturas.fecha) > 21
9       AND (UPPER(Vendedores.nombre) LIKE '%EZ %' OR UPPER(
           Vendedores.nombre) LIKE '%EZ')
10      AND pueblo1.codpro = pueblo2.codpro
11      GROUP BY codven, Vendedores.nombre;
```

#	Codigo Vendedor	Nombre	Nº Cliente

El nombre de ningun vendedor termina en EZ.

15. Escribir una consulta que obtenga el código y nombre de aquellos pueblos donde residen al menos un vendedor o al menos un cliente. No eliminar del resultado los pueblos repetidos.

```
1      -- EJERCICIO 15
2      SELECT codpue 'Codigo Pueblo', pueblo1.nombre 'Nombre'
          FROM Pueblos pueblo1
3      JOIN Vendedores USING (codpue)
4      UNION ALL
5      SELECT codpue 'Codigo Pueblo', pueblo2.nombre 'Nombre'
          FROM Pueblos pueblo2
6      JOIN Clientes USING (codpue);
```

#	Codigo Pueblo	Nombre
1	ALI00	Guadalest
2	ALI00	Guadalest
3	ALI01	Calpe
4	ALI01	Calpe
5	AQP00	Arequipa
6	BAR00	Tavertet
7	BAR00	Tavertet
8	BAR01	Cardona
9	BAR01	Cardona
10	GER00	Monells
11	GER00	Monells
12	GER01	Santa ...
13	GER01	Santa ...
14	VAL00	Sagunto
15	VAL00	Sagunto
16	VAL01	Miramar
17	VAL01	Miramar

17	VAL01	Miramar
18	ALI00	Guadalest
19	ALI00	Guadalest
20	ALI01	Calpe
21	ALI01	Calpe
22	AQP00	Arequipa
23	BAR00	Tavertet
24	BAR00	Tavertet
25	BAR01	Cardona
26	BAR01	Cardona
27	GER00	Monells
28	GER00	Monells
29	GER01	Santa ...
30	GER01	Santa ...
31	VAL00	Sagunto
32	VAL00	Sagunto
33	VAL01	Miramar
34	VAL01	Miramar

16. Escribir una consulta que obtenga el código y nombre de aquellos pueblos donde residen al menos un vendedor o al menos un cliente. Eliminar del resultado los pueblos repetidos.

```
1      -- EJERCICIO 16
2      SELECT codpue 'Codigo Pueblo', pueblo1.nombre 'Nombre'
```

```
        FROM Pueblos pueblo1
JOIN Vendedores USING (codpue)
UNION
SELECT codpue 'Codigo Pueblo', pueblo2.nombre 'Nombre'
        FROM Pueblos pueblo2
JOIN Clientes USING (codpue);
```

#	Codigo Pueblo	Nombre
1	ALI00	Guadalest
2	ALI01	Calpe
3	AQP00	Arequipa
4	BAR00	Tavertet
5	BAR01	Cardona
6	GER00	Monells
7	GER01	Santa Pau
8	VAL00	Sagunto
9	VAL01	Miramar

17. Escribir una consulta que obtenga el código y nombre de aquellos pueblos donde residen al menos un vendedor y al menos un cliente.

```
-- EJERCICIO 17
SELECT DISTINCT Pueblos.codpue 'Codigo Pueblo', Pueblos.
        nombre 'Nombre' FROM Pueblos
INNER JOIN Vendedores ON Pueblos.codpue = Vendedores.
        codpue
INNER JOIN Clientes ON Pueblos.codpue = Clientes.codpue;
```

#	Codigo Pueblo	Nombre
1	ALI00	Guadalest
2	ALI01	Calpe
3	AQP00	Arequipa
4	BAR00	Tavertet
5	BAR01	Cardona
6	GER00	Monells
7	GER01	Santa Pau
8	VAL00	Sagunto
9	VAL01	Miramar

18. Escribir una consulta que obtenga el código y nombre de aquellos pueblos donde residen al menos un vendedor pero no reside ningún cliente.

```
-- EJERCICIO 18
SELECT DISTINCT Pueblos.codpue 'Codigo Pueblo', Pueblos.
   nombre 'Nombre' FROM Pueblos
INNER JOIN Vendedores ON Pueblos.codpue = Vendedores.
   codpue
WHERE Pueblos.codpue NOT IN(
SELECT Pueblos.codpue FROM Pueblos
INNER JOIN Clientes ON Pueblos.codpue = Clientes.codpue)
   ;
```

#	Codigo Pueblo	Nombre

19. Escribir una consulta que obtenga el código y descripción de aquellos artículos que nunca han sido vendidos en el mes de enero.

```
-- EJERCICIO 19
SELECT DISTINCT A.codart 'Codigo Articulo', A.descrip '
   Descripcion' FROM Articulos A
WHERE A.codart NOT IN(
SELECT B.codart FROM Articulos B
JOIN Lineas_fac USING (codart)
JOIN Facturas USING (codfac)
WHERE EXTRACT(month FROM Facturas.fecha) = 1);
```

#	Codigo Articulo	Descrip
1	BATIDO00	Batidora
2	CAFETE00	Cafetera
3	EXTRAC00	Extractor
4	LAVADO00	Lavadora
5	PICADO00	Picadora
6	SECADO00	Secadora

20. Escribir una consulta que muestre el código de cada artículo cuyo stock supera las 20 unidades, con un precio superior a 15 euros, y de los que no hay ninguna factura en el último trimestre del año pasado.

```
-- EJERCICIO 20
SELECT DISTINCT Articulos.codart 'Codigo Articulo' FROM
   Articulos
WHERE Articulos.stock > 20 AND Articulos.precio > 15
NOT IN(
SELECT Lineas_fac.codart FROM Lineas_fac, Facturas
WHERE Facturas.codfac = Lineas_fac.codfac
AND EXTRACT(year FROM Facturas.fecha) = EXTRACT(year
   FROM CURRENT_DATE) - 1);
```

#	Codigo Articulo
1	BATIDO00
2	CAFETE00
3	EXTRAC00
4	LICUAD00
5	PICADO00
6	SECADO00

21. Vendedores y clientes cuyo nombre coincide (vendedores que a su vez han comprado algo a la empresa).

```
-- EJERCICIO 21
SELECT DISTINCT Vendedores.nombre 'Nombre Vendedor'
   FROM Vendedores
INNER JOIN Clientes ON Vendedores.nombre = Clientes.
   nombre;
```

#	Nombre Vendedor

En la base de datos ningun nombre de vendedor coincide con algun nombre de cliente.

22. Escribir una consulta que muestre los códigos de los artículos tales que su stock esté por debajo del doble de su stock mínimo, y el número total de unidades vendidas sea mayor que 100.

    NOTA: Para que se observe mejor el resultado del query cambie los valores un poco aumentanto el stock minimo al triple en lugar del doble y reduciendo el total de unidades a 10.

```
1      -- EJERCICIO 22
2      SELECT DISTINCT Articulos.codart 'Codigo Articulo' FROM
          Articulos
3      INNER JOIN Lineas_fac ON Articulos.codart = Lineas_fac.
          codart
4      WHERE Articulos.stock < Articulos.stock_min * 3
5      GROUP BY Lineas_fac.codart
6      HAVING SUM(Lineas_fac.cant) > 10
```

#	Codigo Articulo
1	LICUAD00

23. Escribir una consulta que obtenga la facturación mensual de cada mes y también la facturación anual tras el mes de diciembre de todos los años en los que la empresa está operando.

    NOTA: La base de datos esta a partir del 2019 y no se han realizado ventas en el 2020 por lo tanto solo habra un anyo de facturacion.

```
1      -- EJERCICIO 23
2      SELECT DATE_FORMAT(SYSDATE(), '%Y') || DATE_FORMAT(
          SYSDATE(), '%m') 'Codigo', DATE_FORMAT(SYSDATE(), '%Y
          ') 'Anyo', SUM( Lineas_fac.cant * Lineas_fac.precio )
           'Facturacion' FROM Facturas
3      JOIN Lineas_fac USING (codfac)
4      GROUP BY DATE_FORMAT(SYSDATE(), '%Y'), DATE_FORMAT(
          SYSDATE(), '%m')
```

#	Codigo	Anyo	Facturacion
1	1	2020	56752.28

24. Escribir una consulta que obtenga el código y nombre de aquellas provincias en las que no hubo ventas de los vendedores residentes en dichas provincias durante el año pasado.

```
-- EJERCICIO 24
SELECT A.codpro 'Codigo Provincia', A.nombre 'Nombre'
   FROM Provincias A
WHERE A.codpro NOT IN(
SELECT B.codpro FROM Provincias B
JOIN Pueblos USING (codpro)
JOIN Vendedores USING (codpue)
JOIN Facturas USING (codven)
WHERE EXTRACT(year FROM Facturas.fecha) = EXTRACT(year
   FROM CURRENT_DATE) - 1);
```

#	Codigo Provincia	Nombre
1	CA	Castellon

25. Escribir una consulta que muestre el código y descripción de aquellos artículos que se han vendido alguna vez, pero nunca en la provincia de Castellón.

```
-- EJERCICIO 25
SELECT DISTINCT Articulos.codart 'Codigo Articulo',
   Articulos.descrip 'Descripcion' FROM Articulos
JOIN Lineas_fac USING (codart)
WHERE Articulos.codart NOT IN(
SELECT Articulos.codart FROM Articulos
JOIN Lineas_fac USING (codart)
JOIN Facturas USING (codfac)
JOIN Clientes USING (codcli)
JOIN Pueblos USING (codpue)
JOIN Provincias USING (codpro)
WHERE Provincias.nombre = 'Castellon');
```

#	Codigo Articulo	Descripcion
1	BATIDO00	Batidora
2	CAFETE00	Cafetera
3	COCINA00	Cocina
4	EXTRAC00	Extractor
5	LAVADO00	Lavadora
6	LICUAD00	Licuadora
7	PICADO00	Picadora
8	REFRIG00	Refrigerador
9	SECADO00	Secadora

26. Escribir una consulta que muestre el nombre de cada provincia y el número de facturas realizadas a clientes de dicha provincia durante el año pasado. Si una provincia no tiene ninguna factura, debe aparecer con la cantidad cero.

```
1   -- EJERCICIO 26
2   SELECT Provincias.Nombre 'Nombre Provincia', COUNT(*) 'N
        Facturas' FROM Provincias
3   JOIN Pueblos USING (codpro)
4   JOIN Clientes USING (codpue)
5   JOIN Facturas USING (codcli)
6   WHERE EXTRACT(year FROM Facturas.fecha) = EXTRACT(year
       FROM CURRENT_DATE) - 1
7   GROUP BY Provincias.codpro
8   UNION(
9   SELECT Provincias.nombre, 0 FROM Provincias
10  WHERE Provincias.codpro NOT IN(
11  SELECT Provincias.codpro FROM Provincias
12  JOIN Pueblos USING (codpro)
13  JOIN Clientes USING (codpue)
14  JOIN Facturas USING (codcli)
15  WHERE EXTRACT(year FROM Facturas.fecha) = EXTRACT(year
       FROM CURRENT_DATE) - 1));
```

#	Nombre Provincia	N Facturas
1	Alicante	8
2	Arequipa	1
3	Barcelona	7
4	Gerona	10
5	Valencia	8
6	Castellon	0

Usted puede encontrar todos los queries realizados anteriormente en la practica en el archivo **Query.sql** que se anexara con el informe.