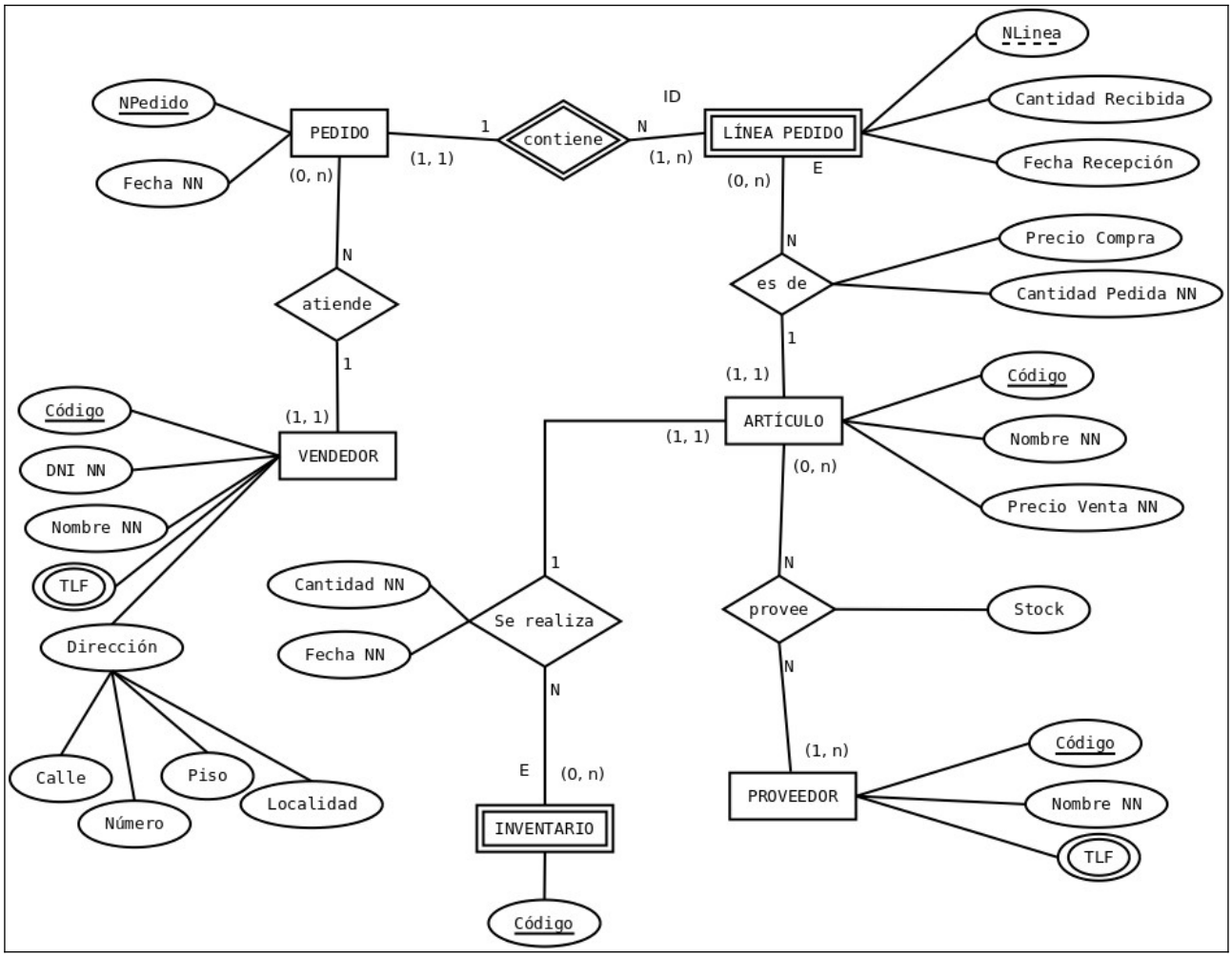
SQLAlchemy trabajo I

# Table of Contents

## DER de partida




## Paso a tablas (parcial)

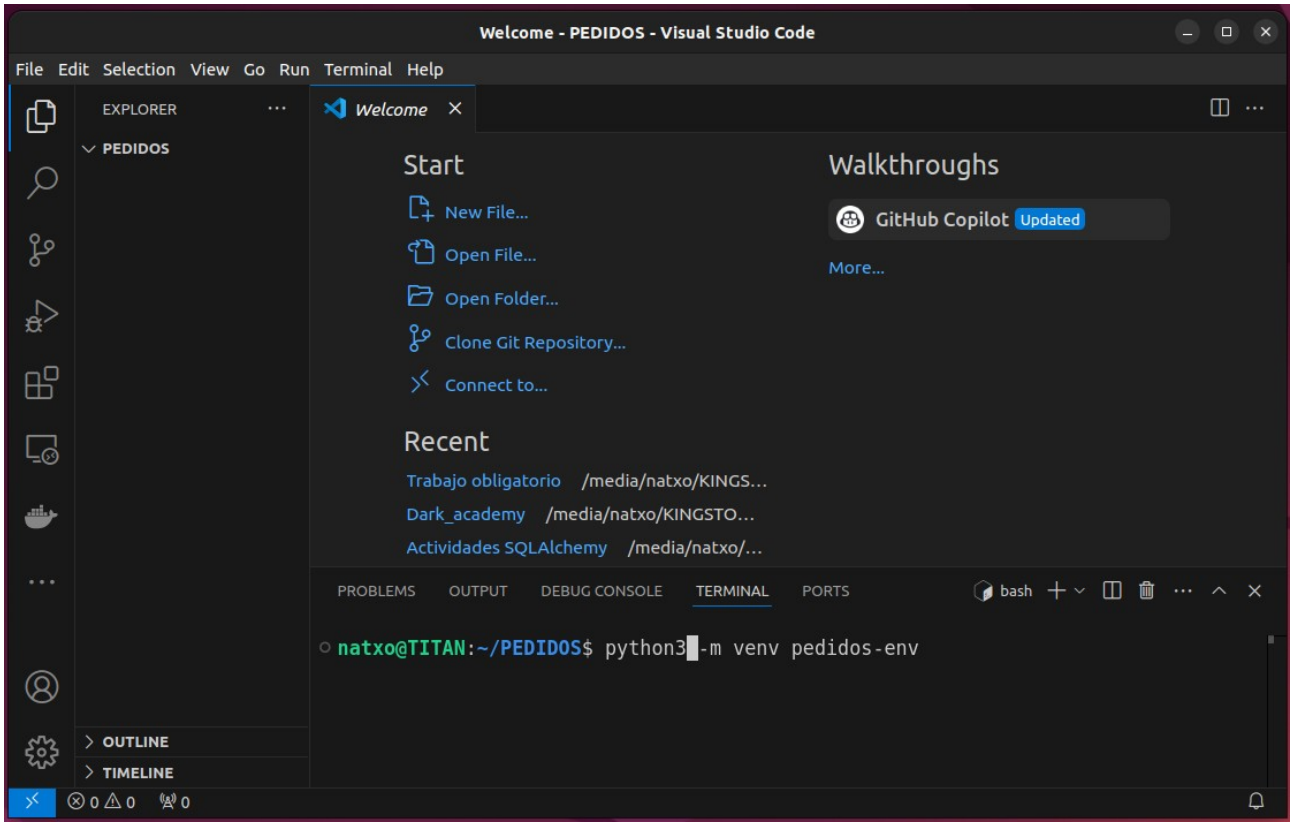
**VENDEDORES(Código, DNI, Nombre, Dcalle, Dnúmero, Dpiso, Dlocalidad)**
PK: {Código}
AK: {DNI}
VNN: {DNI}, {Nombre}

**TLFS(TLF, Código_vendedor) #normalización multivaluados**
PK: {TLF}
FK: {Código_vendedor} -> VENDEDORES(Código) BC/MC
VNN: {Código_vendedor} DE

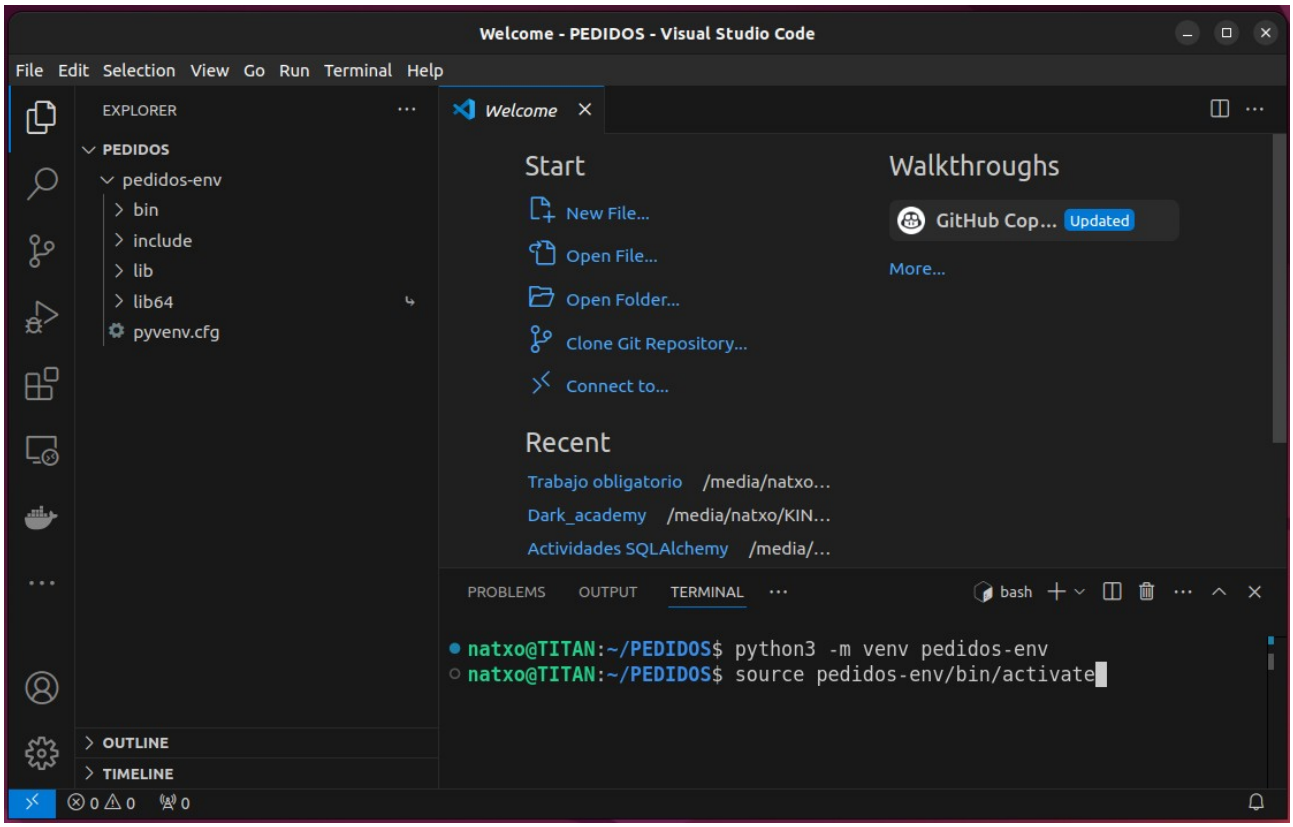
**PEDIDOS(NPedido, Fecha, Código_vendedor)**
PK: {NPedido}
FK: {Código_vendedor} -> VENDEDORES(Código) BR/MC
VNN: {Código_vendedor} DE

## Creación VENV para el proyecto PEDIDOS

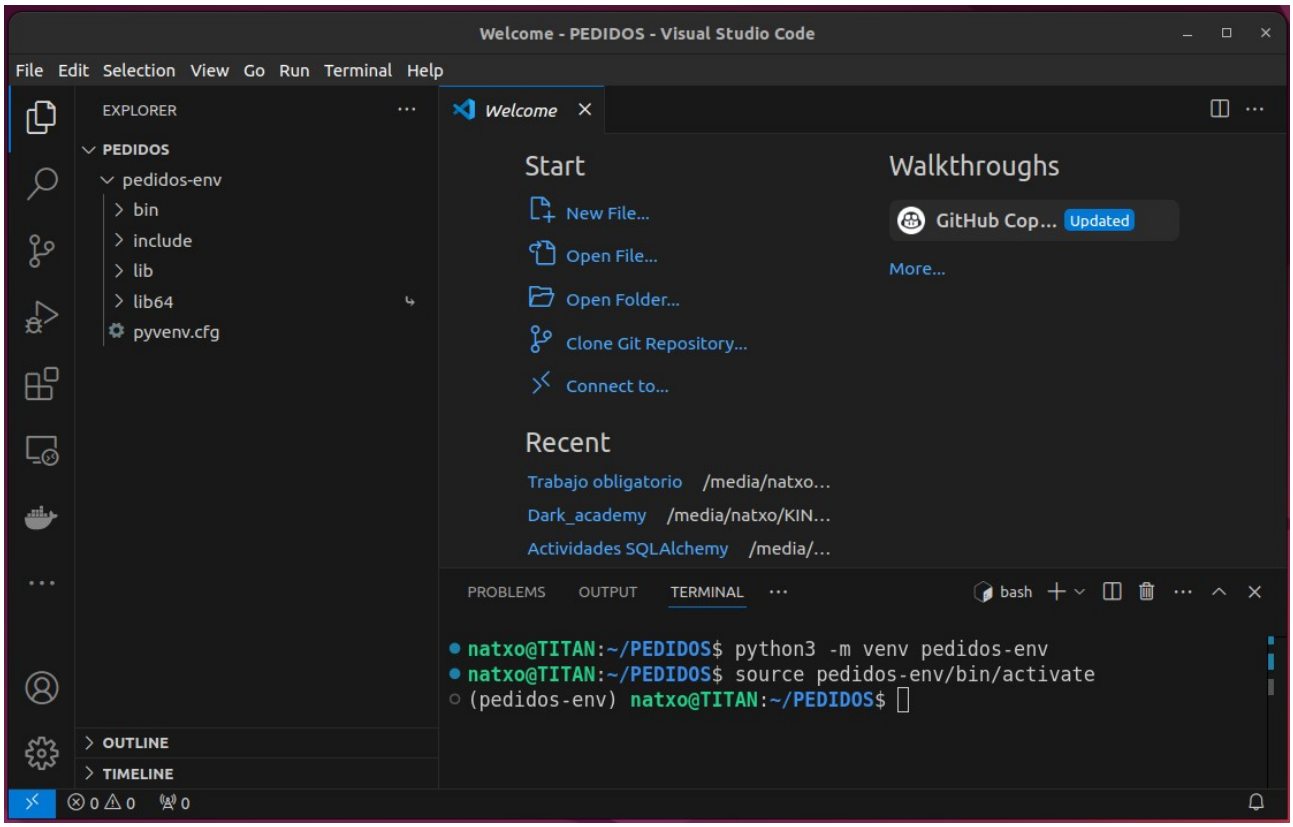
1. Creamos carpeta en almacenamiento POSIX y dentro de ella ejecutamos:



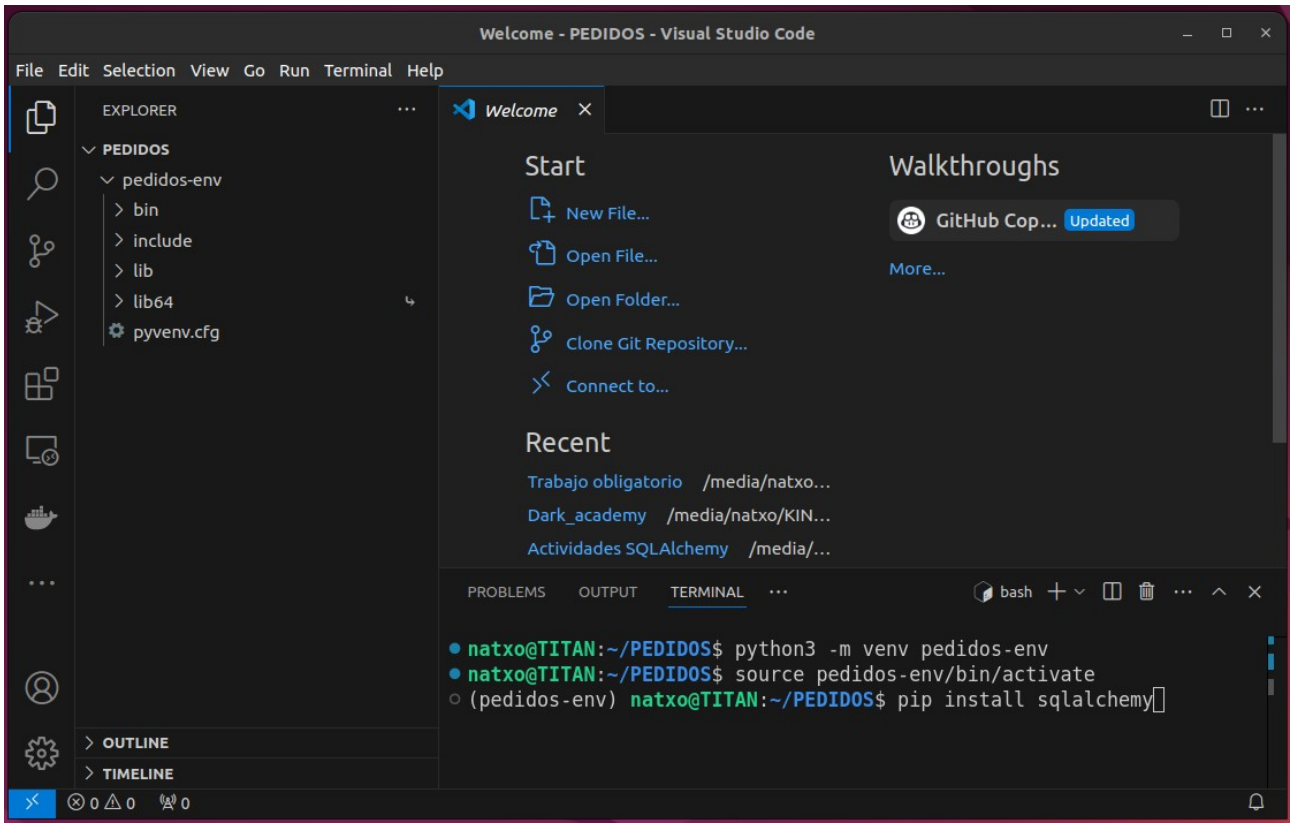

Esto crea la carpeta con el entorno congelado con la versión actual de Python para la aplicación que vamos a crear. Ahora activamos el entorno virtual:



Vemos que el prompt cambia:




Ahora instalamos SQLAlchemy para que esté en el entorno virtual:



Cuando acabemos la sesión de programación teclemos **deactivate** y nos saca del entorno.

## CREACIÓN DE TABLAS (Vendedor, Telefono y Pedido)

Se proporciona el código. Explóralo y tómalo como base para las tablas que tengas que crear más adelante.
**NOTA:** los ficheros proporcionados son:

- conectar.py: aquí solo tenéis que adaptar el url_object para conectar a vuestro mysql server
- crear_BD.py: aquí no tenéis que hacer nada
- crear_tablas.py: crea las tablas (borrando contenidos anteriores). Aquí tenéis que añadir las tablas mapeadas.

En resumen, en conectar.py adaptáis el url_object y en crear_tablas.py añadís nuevas tablas.

## INSERCION DE DATOS (Vendedor, Telefono y Pedido)

Se proporciona el código. Explóralo y tómalo como base para inserciones en otras tablas que tengas que hacer más adelante.

**NOTA:** una vez creadas las tablas. Para insertar datos se usa la reflexión de tablas (para que veáis como se hace), podéis observarlo en los ficheros que se adjuntan. Esto es porque en crear_tablas.py se usa crear_BD.py que borra la BD. Más adelante, para consultas.py lo haremos de otra manera.

Se proporciona el fichero datos_vendedores.py

En la tabla vendedores se insertan los siguientes datos:
Vendedor('1111111A', 'Pepe')
Vendedor'22222222B', 'Pepa')
Vendedor('33333333C', 'Pepito')

En la tabla telefonos se insertan los siguientes datos:
Telefono('111111111', 1)
Telefono('121212121', 1)
Telefono('222222222', 2)
Telefono('232323232', 2)
Telefono('333333333', 3)
Telefono('343434343', 3)

Se proporciona el fichero datos_pedidos.py

En la tabla pedidos se insertan los siguientes datos:

Pedido(fecha=date(2025, 1, 7), codigo_vendedor=1)
Pedido(fecha=date(2025, 1, 8), codigo_vendedor=1)
Pedido(fecha=date(2025, 1, 9), codigo_vendedor=1)
Pedido(fecha=date(2025, 1, 6), codigo_vendedor=2)
Pedido(fecha=date(2025, 1, 7), codigo_vendedor=2)

De momento esto es lo que se tiene:

```
mysql> use PEDIDOS
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-------------------+
| Tables_in_PEDIDOS |
+-------------------+
| pedidos           |
| telefonos         |
| vendedores        |
+-------------------+
3 rows in set (0,00 sec)

mysql> select * from vendedores;
+--------+-----------+--------+--------+---------+-------+------------+
| codigo | DNI       | nombre | dcalle | dnumero | dpiso | dlocalidad |
+--------+-----------+--------+--------+---------+-------+------------+
|      1 | 11111111A | Pepe   | NULL   | NULL    | NULL  | NULL       |
|      2 | 22222222B | Pepa   | NULL   | NULL    | NULL  | NULL       |
|      3 | 33333333C | Pepito | NULL   | NULL    | NULL  | NULL       |
+--------+-----------+--------+--------+---------+-------+------------+
3 rows in set (0,00 sec)

mysql> select * from telefonos;
+-----------+-----------------+
| TLF       | codigo_vendedor |
+-----------+-----------------+
| 111111111 |               1 |
| 121212121 |               1 |
| 222222222 |               2 |
| 232323232 |               2 |
| 333333333 |               3 |
| 343434343 |               3 |
+-----------+-----------------+
6 rows in set (0,00 sec)

mysql> select * from pedidos;
+---------+------------+-----------------+
| npedido | fecha      | codigo_vendedor |
+---------+------------+-----------------+
|       1 | 2025-01-07 |               1 |
|       2 | 2025-01-08 |               1 |
|       3 | 2025-01-09 |               1 |
|       4 | 2025-01-06 |               2 |
|       5 | 2025-01-07 |               2 |
+---------+------------+-----------------+
5 rows in set (0,00 sec)

mysql>
```

## Es tu turno:

Ya tenemos estas tablas:

**VENDEDORES(Código, DNI, Nombre, Dcalle, Dnúmero, Dpiso, Dlocalidad)**
PK: {Código}
AK: {DNI}
VNN: {DNI}, {Nombre}

**TLFS(TLF, Código_vendedor) #normalización multivaluados**
PK: {TLF}
FK: {Código_vendedor} -> VENDEDORES(Código) BC/MC
VNN: {Código_vendedor} DE

**PEDIDOS(NPedido, Fecha, Código_vendedor)**
PK: {NPedido}
FK: {Código_vendedor} -> VENDEDORES(Código) BR/MC
VNN: {Código_vendedor} DE

Vamos a generar/normalizar ARTICULOS y LINEAS_PEDIDO:

**ARTICULOS(Código, Nombre, Precio_venta)**
PK: {código}
VNN: {Nombre}, {Precio_venta}

**LINEAS_PEDIDO(NLínea, NPedido, Código_artículo, Precio_compra, Cantidad_pedida, Cantidad_recibida, Fecha_recepción)**
PK: {NLínea, NPedido} # fortalecimento clave débil
FK: {NPedido} -> PEDIDOS(NPedido) BC/MC
FK: {Código_artículo} -> ARTICULOS(Código) BR/MC
VNN: {Cantidad_pedida}, {Código_artículo} DE

**SE PIDE:**
1. Amplia crear_tablas.py para dar cabida a estas dos nuevas tablas.
2. Genera los ficheros: datos_articulos.py y datos_lineas_pedido.py para poner datos coherentes con los que ya existen en la BDA.

El SQL emitido debe ser:

```
2025-01-09 10:03:35,474 INFO sqlalchemy.engine.Engine [raw sql] ()
2025-01-09 10:03:35,474 INFO sqlalchemy.engine.Engine DESCRIBE `PEDIDOS`.`articulos`
2025-01-09 10:03:35,474 INFO sqlalchemy.engine.Engine [raw sql] ()
2025-01-09 10:03:35,474 INFO sqlalchemy.engine.Engine DESCRIBE `PEDIDOS`.`lineas_pedido`
2025-01-09 10:03:35,474 INFO sqlalchemy.engine.Engine [raw sql] ()
2025-01-09 10:03:35,475 INFO sqlalchemy.engine.Engine
CREATE TABLE vendedores (
	codigo INTEGER NOT NULL AUTO_INCREMENT,
	`DNI` VARCHAR(9) NOT NULL,
	nombre VARCHAR(75) NOT NULL,
	dcalle VARCHAR(50),
	dnumero VARCHAR(5),
	dpiso VARCHAR(5),
	dlocalidad VARCHAR(50),
	PRIMARY KEY (codigo),
	UNIQUE (`DNI`)
)

2025-01-09 10:03:35,475 INFO sqlalchemy.engine.Engine [no key 0.00006s] ()
2025-01-09 10:03:35,519 INFO sqlalchemy.engine.Engine
CREATE TABLE articulos (
	codigo INTEGER NOT NULL AUTO_INCREMENT,
	nombre VARCHAR(50) NOT NULL,
	precio_venta NUMERIC(8, 2) NOT NULL,
	PRIMARY KEY (codigo)
)

2025-01-09 10:03:35,519 INFO sqlalchemy.engine.Engine [no key 0.00016s] ()
2025-01-09 10:03:35,549 INFO sqlalchemy.engine.Engine
CREATE TABLE telefonos (
	`TLF` VARCHAR(9) NOT NULL,
	codigo_vendedor INTEGER NOT NULL,
	PRIMARY KEY (`TLF`),
	FOREIGN KEY(codigo_vendedor) REFERENCES vendedores (codigo) ON DELETE CASCADE ON UPDATE
CASCADE
)

2025-01-09 10:03:35,549 INFO sqlalchemy.engine.Engine [no key 0.00018s] ()
2025-01-09 10:03:35,589 INFO sqlalchemy.engine.Engine
CREATE TABLE pedidos (
	npedido INTEGER NOT NULL AUTO_INCREMENT,
	fecha DATE NOT NULL,
	codigo_vendedor INTEGER NOT NULL,
	PRIMARY KEY (npedido),
	FOREIGN KEY(codigo_vendedor) REFERENCES vendedores (codigo) ON DELETE RESTRICT ON UPDATE
CASCADE
)
```

2025-01-09 10:03:35,590 INFO sqlalchemy.engine.Engine [no key 0.00020s] ()
2025-01-09 10:03:35,631 INFO sqlalchemy.engine.Engine
**CREATE TABLE lineas_pedido (**
**nlinea INTEGER NOT NULL,**
**npedido INTEGER NOT NULL,**
**codigo_articulo INTEGER NOT NULL,**
**precio_compra NUMERIC(8, 2),**
**cantidad_pedida INTEGER NOT NULL,**
**cantidad_recibida INTEGER,**
**fecha_recepcion DATE,**
**PRIMARY KEY (nlinea, npedido),**
**FOREIGN KEY(npedido) REFERENCES pedidos (npedido) ON DELETE CASCADE ON UPDATE CASCADE,**
**FOREIGN KEY(codigo_articulo) REFERENCES articulos (codigo) ON DELETE RESTRICT ON UPDATE CASCADE**
**)**

2025-01-09 10:03:35,631 INFO sqlalchemy.engine.Engine [no key 0.00015s] ()
2025-01-09 10:03:35,675 INFO sqlalchemy.engine.Engine COMMIT

Un ejemplo de datos insertados puede ser:

```
mysql> use PEDIDOS
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from articulos;
+--------+------------------------------+--------------+
| codigo | nombre                       | precio_venta |
+--------+------------------------------+--------------+
|      1 | Raspberry PI 4 4GB           |        94.90 |
|      2 | Raspberry PI 5 8GB Starter Kit |       144.90 |
|      3 | Raspberry PI Zero 8GB        |        29.09 |
|      4 | Raspberry PI 5 8GB           |        94.92 |
|      5 | Placa desarrollo RP          |        15.79 |
+--------+------------------------------+--------------+
5 rows in set (0,00 sec)

mysql> select * from lineas_pedido;
+--------+---------+-----------------+---------------+-----------------+-------------------+-----------------+
| nlinea | npedido | codigo_articulo | precio_compra | cantidad_pedida | cantidad_recibida | fecha_recepcion |
+--------+---------+-----------------+---------------+-----------------+-------------------+-----------------+
|      1 |       1 |               1 |          NULL |               2 |              NULL | NULL            |
|      1 |       2 |               1 |          NULL |               2 |              NULL | NULL            |
|      1 |       3 |               4 |          NULL |               3 |              NULL | NULL            |
|      1 |       4 |               5 |          NULL |               3 |              NULL | NULL            |
|      1 |       5 |               4 |          NULL |               3 |              NULL | NULL            |
|      2 |       1 |               2 |          NULL |               4 |              NULL | NULL            |
|      2 |       2 |               2 |          NULL |               3 |              NULL | NULL            |
|      2 |       3 |               1 |          NULL |               2 |              NULL | NULL            |
|      2 |       5 |               5 |          NULL |               6 |              NULL | NULL            |
|      3 |       3 |               3 |          NULL |               2 |              NULL | NULL            |
+--------+---------+-----------------+---------------+-----------------+-------------------+-----------------+
10 rows in set (0,00 sec)

mysql>
```

## Es tu turno II:

Ya tenemos estas tablas:

**VENDEDORES(Código, DNI, Nombre, Dcalle, Dnúmero, Dpiso, Dlocalidad)**
PK: {Código}
AK: {DNI}
VNN: {DNI}, {Nombre}

**TLFS(TLF, Código_vendedor) #normalización multivaluados**
PK: {TLF}
FK: {Código_vendedor} -> VENDEDORES(Código) BC/MC
VNN: {Código_vendedor} DE

**PEDIDOS(NPedido, Fecha, Código_vendedor)**

PK: {NPedido}
FK: {Código_vendedor} -> VENDEDORES(Código) BR/MC
VNN: {Código_vendedor} DE

**ARTICULOS(Código, Nombre, Precio_venta)**
PK: {código}
VNN: {Nombre}, {Precio_venta}

**LINEAS_PEDIDO(NLínea, NPedido, Código_artículo, Precio_compra, Cantidad_pedida, Cantidad_recibida, Fecha_recepción)**
PK: {NLínea, NPedido} # fortalecimento clave débil
FK: {NPedido} -> PEDIDOS(NPedido) BC/MC
FK: {Código_artículo} -> ARTICULOS(Código) BR/MC
VNN: {Cantidad_pedida}, {Código_artículo} (DE)

Vamos a generar/normalizar PROVEEDORES y provee:

**PROVEEDORES(Código, Nombre)**
PK: {Código)
VNN: {Nombre}

**TLFS_P(TLF, Código_proveedor) # normalización**
PK: {TLF}
FK: {Código_proveedor} → Proveedores(Código) BR/MC
VNN: {Código_proveedor}

**provee(Código_artículo, Código_proveedor, stock)**
PK: {Código_artículo, Código_proveedor}
FK: {Código_artículo} → ARTICULOS(Código) BR/MC
FK: {Código_proveedor} → PROVEEDORES(Código) BR/MC
VNN: Stock

**SE PIDE:**
1. Amplia crear_tablas.py para dar cabida a estas tres nuevas tablas.
2. Genera los ficheros: datos_proveedores.py (datos de proveedores y sus teléfonos) y datos_provee.py para poner datos coherentes con los que ya existen en la BDA.

El SQL emitido debe ser:

```
2025-01-10 09:25:21,558 INFO sqlalchemy.engine.Engine SELECT DATABASE()
2025-01-10 09:25:21,558 INFO sqlalchemy.engine.Engine [raw sql] ()
2025-01-10 09:25:21,558 INFO sqlalchemy.engine.Engine SELECT @@sql_mode
2025-01-10 09:25:21,558 INFO sqlalchemy.engine.Engine [raw sql] ()
2025-01-10 09:25:21,559 INFO sqlalchemy.engine.Engine SELECT @@lower_case_table_names
2025-01-10 09:25:21,559 INFO sqlalchemy.engine.Engine [raw sql] ()
2025-01-10 09:25:21,559 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2025-01-10 09:25:21,559 INFO sqlalchemy.engine.Engine DESCRIBE `PEDIDOS`.`vendedores`
2025-01-10 09:25:21,559 INFO sqlalchemy.engine.Engine [raw sql] ()
2025-01-10 09:25:21,560 INFO sqlalchemy.engine.Engine DESCRIBE `PEDIDOS`.`telefonos`
2025-01-10 09:25:21,560 INFO sqlalchemy.engine.Engine [raw sql] ()
2025-01-10 09:25:21,561 INFO sqlalchemy.engine.Engine DESCRIBE `PEDIDOS`.`pedidos`
2025-01-10 09:25:21,561 INFO sqlalchemy.engine.Engine [raw sql] ()
2025-01-10 09:25:21,561 INFO sqlalchemy.engine.Engine DESCRIBE `PEDIDOS`.`articulos`
2025-01-10 09:25:21,561 INFO sqlalchemy.engine.Engine [raw sql] ()
2025-01-10 09:25:21,561 INFO sqlalchemy.engine.Engine DESCRIBE `PEDIDOS`.`lineas_pedido`
2025-01-10 09:25:21,561 INFO sqlalchemy.engine.Engine [raw sql] ()
2025-01-10 09:25:21,562 INFO sqlalchemy.engine.Engine DESCRIBE `PEDIDOS`.`proveedores`
2025-01-10 09:25:21,562 INFO sqlalchemy.engine.Engine [raw sql] ()
2025-01-10 09:25:21,562 INFO sqlalchemy.engine.Engine DESCRIBE `PEDIDOS`.`telefonos_p`
```

2025-01-10 09:25:21,562 INFO sqlalchemy.engine.Engine [raw sql] ()
2025-01-10 09:25:21,563 INFO sqlalchemy.engine.Engine DESCRIBE `PEDIDOS`.`provee`
2025-01-10 09:25:21,563 INFO sqlalchemy.engine.Engine [raw sql] ()
2025-01-10 09:25:21,563 INFO sqlalchemy.engine.Engine
**CREATE TABLE vendedores (**
**codigo INTEGER NOT NULL AUTO_INCREMENT,**
**`DNI` VARCHAR(9) NOT NULL,**
**nombre VARCHAR(75) NOT NULL,**
**dcalle VARCHAR(50),**
**dnumero VARCHAR(5),**
**dpiso VARCHAR(5),**
**dlocalidad VARCHAR(50),**
**PRIMARY KEY (codigo),**
**UNIQUE (`DNI`)**
**)**

2025-01-10 09:25:21,564 INFO sqlalchemy.engine.Engine [no key 0.00006s] ()
2025-01-10 09:25:21,662 INFO sqlalchemy.engine.Engine
**CREATE TABLE articulos (**
**codigo INTEGER NOT NULL AUTO_INCREMENT,**
**nombre VARCHAR(50) NOT NULL,**
**precio_venta NUMERIC(8, 2) NOT NULL,**
**PRIMARY KEY (codigo)**
**)**

2025-01-10 09:25:21,662 INFO sqlalchemy.engine.Engine [no key 0.00017s] ()
2025-01-10 09:25:21,699 INFO sqlalchemy.engine.Engine
**CREATE TABLE proveedores (**
**codigo INTEGER NOT NULL AUTO_INCREMENT,**
**nombre VARCHAR(50) NOT NULL,**
**PRIMARY KEY (codigo)**
**)**

2025-01-10 09:25:21,699 INFO sqlalchemy.engine.Engine [no key 0.00020s] ()
2025-01-10 09:25:21,731 INFO sqlalchemy.engine.Engine
**CREATE TABLE telefonos (**
**`TLF` VARCHAR(9) NOT NULL,**
**codigo_vendedor INTEGER NOT NULL,**
**PRIMARY KEY (`TLF`),**
**FOREIGN KEY(codigo_vendedor) REFERENCES vendedores (codigo) ON DELETE CASCADE ON UPDATE CASCADE**
**)**

2025-01-10 09:25:21,731 INFO sqlalchemy.engine.Engine [no key 0.00016s] ()
2025-01-10 09:25:21,774 INFO sqlalchemy.engine.Engine
**CREATE TABLE pedidos (**
**npedido INTEGER NOT NULL AUTO_INCREMENT,**
**fecha DATE NOT NULL,**
**codigo_vendedor INTEGER NOT NULL,**
**PRIMARY KEY (npedido),**
**FOREIGN KEY(codigo_vendedor) REFERENCES vendedores (codigo) ON DELETE RESTRICT ON UPDATE CASCADE**
**)**

2025-01-10 09:25:21,774 INFO sqlalchemy.engine.Engine [no key 0.00016s] ()
2025-01-10 09:25:21,817 INFO sqlalchemy.engine.Engine
**CREATE TABLE telefonos_p (**
**`TLF` VARCHAR(9) NOT NULL,**
**codigo_proveedor INTEGER NOT NULL,**
**PRIMARY KEY (`TLF`),**
**FOREIGN KEY(codigo_proveedor) REFERENCES proveedores (codigo) ON DELETE RESTRICT ON UPDATE CASCADE**
**)**

2025-01-10 09:25:21,817 INFO sqlalchemy.engine.Engine [no key 0.00016s] ()
2025-01-10 09:25:21,865 INFO sqlalchemy.engine.Engine
**CREATE TABLE provee (**
**codigo_articulo INTEGER NOT NULL,**
**codigo_proveedor INTEGER NOT NULL,**
**stock INTEGER NOT NULL,**
**PRIMARY KEY (codigo_articulo, codigo_proveedor),**
**FOREIGN KEY(codigo_articulo) REFERENCES articulos (codigo) ON DELETE RESTRICT ON UPDATE CASCADE,**
**FOREIGN KEY(codigo_proveedor) REFERENCES proveedores (codigo) ON DELETE RESTRICT ON UPDATE CASCADE**
**)**

2025-01-10 09:25:21,865 INFO sqlalchemy.engine.Engine [no key 0.00016s] ()

```
2025-01-10 09:25:21,914 INFO sqlalchemy.engine.Engine
CREATE TABLE lineas_pedido (
	nlinea INTEGER NOT NULL,
	npedido INTEGER NOT NULL,
	codigo_articulo INTEGER NOT NULL,
	precio_compra NUMERIC(8, 2),
	cantidad_pedida INTEGER NOT NULL,
	cantidad_recibida INTEGER,
	fecha_recepcion DATE,
	PRIMARY KEY (nlinea, npedido),
	FOREIGN KEY(npedido) REFERENCES pedidos (npedido) ON DELETE CASCADE ON UPDATE CASCADE,
	FOREIGN KEY(codigo_articulo) REFERENCES articulos (codigo) ON DELETE RESTRICT ON UPDATE
CASCADE
)

2025-01-10 09:25:21,914 INFO sqlalchemy.engine.Engine [no key 0.00017s] ()
2025-01-10 09:25:21,971 INFO sqlalchemy.engine.Engine COMMIT
```

Un ejemplo de datos insertados puede ser:

```
mysql> select * from proveedores;
+--------+------------+
| codigo | nombre     |
+--------+------------+
|      1 | Amazon     |
|      2 | AliExpress |
+--------+------------+
2 rows in set (0,00 sec)

mysql> select * from provee;
+-----------------+------------------+-------+
| codigo_articulo | codigo_proveedor | stock |
+-----------------+------------------+-------+
|               1 |                1 |    40 |
|               1 |                2 |    20 |
|               2 |                1 |    30 |
|               2 |                2 |    20 |
|               3 |                1 |    50 |
|               4 |                1 |    60 |
|               5 |                2 |    40 |
+-----------------+------------------+-------+
7 rows in set (0,00 sec)

mysql>
```

## Es tu turno III:

Se proporcionan los siguientes ficheros:

- esquema.py: donde se ponen los datos de conexión. Además, contiene las declaraciones de las tablas mapeadas (solo pongo las tres primeras ya que de otra forma os estaría dando la solución del apartado anterior).
- esqueleto de consultas.py: aquí van las consultas ... os pongo la cabecera que importa de esquema.py lo necesario y alguna que otra cosa para las consultas.

**SE PIDE:**

Realiza las siguientes consultas utilizando el estilo ORM (NO daré por buenas consultas realizadas con text()):

1. Nombre de artículos y precio de aquellos que valgan más de 100 euros. El listado debe mostrarse de mayor a menor precio.

2. Listado de vendedores (nombre) ordenado por el número de pedidos (también se muestra) que han hecho. La ordenación es de mayor a menor.

3. Nombre y cantidad vendida del artículo que más se ha vendido en unidades.

4. Línea de pedido más cara de todas (cantidad_pedida * precio_venta). Debe mostrarse nlinea, npedido y el total de la línea.

5. Realiza una consulta que sirva para rellenar la tabla inventario. Es decir, que obtenga para cada artículo (código) en la fecha a actual la cantidad que hay en stock.

Se debe realizar un fichero consultas.py que importe las cosas necesarias y que ejecute las consultas imprimiendo el resultado de éstas.

El resultado de ejecutar consultas.py da muchas pistas:

(pedidos-env) natxo@TITAN:~/PEDIDOS$ /home/natxo/PEDIDOS/pedidos-env/bin/python /home/natxo/PEDIDOS/consultas/consultas.py

2025-01-11 16:21:01,876 INFO sqlalchemy.engine.Engine SELECT DATABASE()

2025-01-11 16:21:01,876 INFO sqlalchemy.engine.Engine [raw sql] ()

2025-01-11 16:21:01,878 INFO sqlalchemy.engine.Engine SELECT @@sql_mode

2025-01-11 16:21:01,878 INFO sqlalchemy.engine.Engine [raw sql] ()

2025-01-11 16:21:01,879 INFO sqlalchemy.engine.Engine SELECT @@lower_case_table_names

2025-01-11 16:21:01,879 INFO sqlalchemy.engine.Engine [raw sql] ()

2025-01-11 16:21:01,880 INFO sqlalchemy.engine.Engine BEGIN (implicit)

2025-01-11 16:21:01,901 INFO sqlalchemy.engine.Engine

**SELECT articulos.nombre, articulos.precio_venta**

**FROM articulos**

**WHERE articulos.precio_venta > %s ORDER BY articulos.precio_venta DESC**

2025-01-11 16:21:01,901 INFO sqlalchemy.engine.Engine [generated in 0.00015s] (100,)

**('Raspberry PI 5 8GB Starter Kit', Decimal('144.90'))**

2025-01-11 16:21:01,904 INFO sqlalchemy.engine.Engine

**SELECT vendedores.nombre, count(pedidos.codigo_vendedor) AS count_1**

**FROM vendedores INNER JOIN pedidos ON vendedores.codigo = pedidos.codigo_vendedor GROUP BY vendedores.nombre**

2025-01-11 16:21:01,904 INFO sqlalchemy.engine.Engine [generated in 0.00008s] ()

**('Pepe', 3)**

**('Pepa', 2)**

**SELECT lineas_pedido.codigo_articulo, sum(lineas_pedido.cantidad_pedida) AS cantidad_total**

**FROM lineas_pedido GROUP BY lineas_pedido.codigo_articulo**

2025-01-11 16:21:01,908 INFO sqlalchemy.engine.Engine

**WITH anon_1 AS**

**(SELECT lineas_pedido.codigo_articulo AS codigo_articulo, sum(lineas_pedido.cantidad_pedida) AS cantidad_total**

**FROM lineas_pedido GROUP BY lineas_pedido.codigo_articulo)**

**SELECT articulos.nombre, cantidad_total**

**FROM articulos INNER JOIN anon_1 ON articulos.codigo = anon_1.codigo_articulo ORDER BY cantidad_total DESC**

2025-01-11 16:21:01,908 INFO sqlalchemy.engine.Engine [generated in 0.00010s] ()

(aquí simplemente obtengo la primera línea con first())

**Placa desarrollo RP**

**9**

2025-01-11 16:21:01,909 INFO sqlalchemy.engine.Engine

**SELECT lineas_pedido.nlinea, lineas_pedido.npedido, lineas_pedido.cantidad_pedida * articulos.precio_venta AS total_linea**

**FROM lineas_pedido INNER JOIN articulos ON articulos.codigo = lineas_pedido.codigo_articulo ORDER BY total_linea DESC**

2025-01-11 16:21:01,909 INFO sqlalchemy.engine.Engine [generated in 0.00008s] ()

(aquí simplemente obtengo la primera línea con first())

**2**

**1**

**579.60**

2025-01-11 16:21:01,912 INFO sqlalchemy.engine.Engine

**SELECT provee.codigo_articulo, provee.codigo_proveedor, provee.stock AS cantidad_en_stock, CURRENT_DATE AS fecha**

**FROM provee**

2025-01-11 16:21:01,912 INFO sqlalchemy.engine.Engine [generated in 0.00013s] ()

(**1, 1, 40, datetime.date(2025, 1, 11))**

**(1, 2, 20, datetime.date(2025, 1, 11))**

**(2, 1, 30, datetime.date(2025, 1, 11))**

**(2, 2, 20, datetime.date(2025, 1, 11))**

**(3, 1, 50, datetime.date(2025, 1, 11))**

**(4, 1, 60, datetime.date(2025, 1, 11))**

**(5, 2, 40, datetime.date(2025, 1, 11))**

(pedidos-env) natxo@TITAN:~/PEDIDOS$

Al final de todo el proyecto queda organizado así: