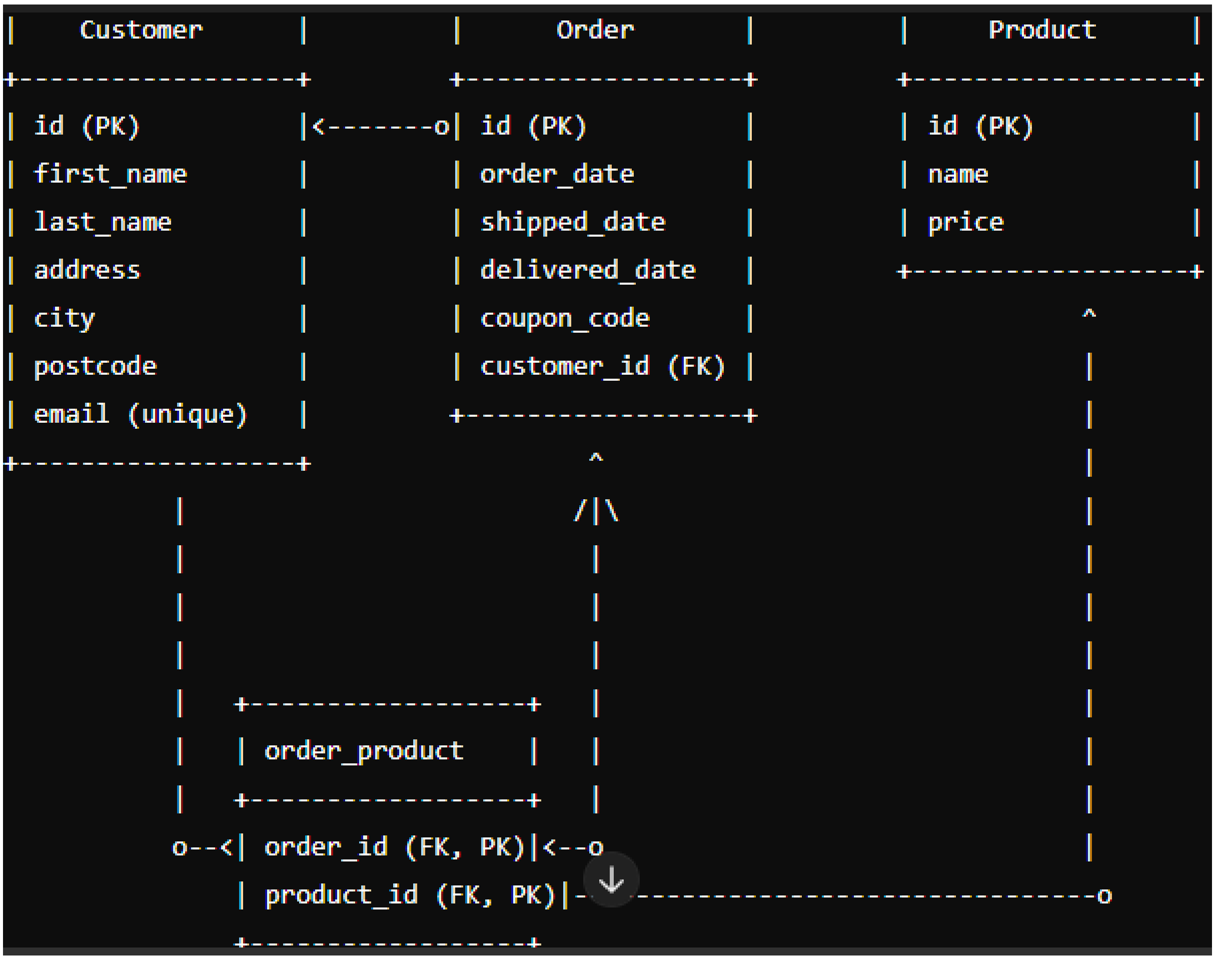
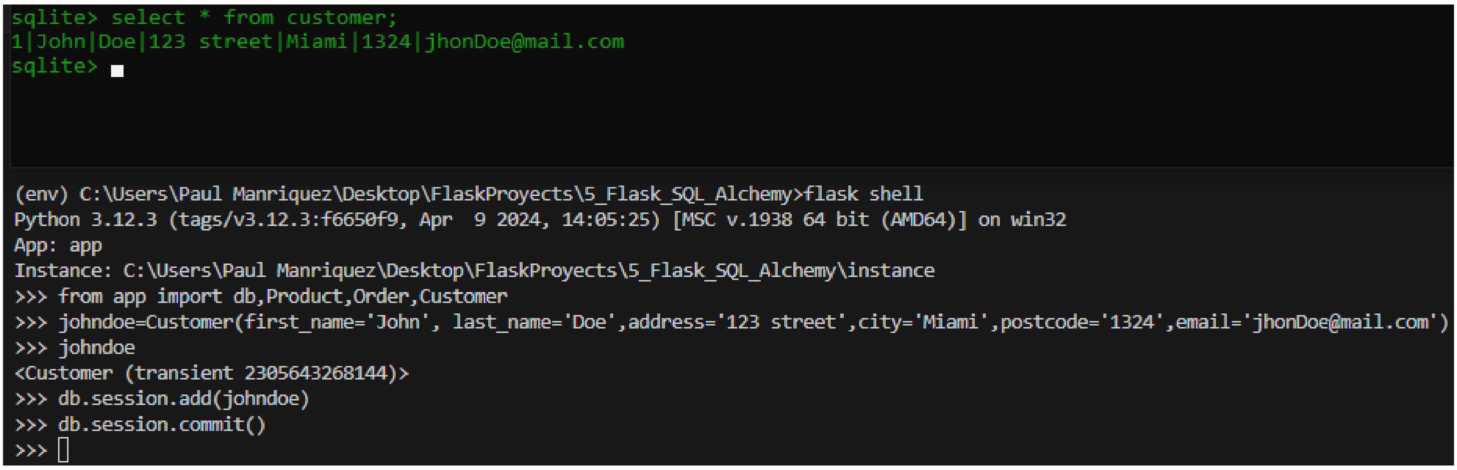
# Flask and FLASK SQL-Alchemy (NOTES)

DataBase worked in this example / simulating the process of simple data base store.



```
|    Customer       |         |      Order        |         |     Product       |
+-------------------+         +-------------------+         +-------------------+
| id (PK)           |<-------o| id (PK)           |         | id (PK)           |
| first_name        |         | order_date        |         | name              |
| last_name         |         | shipped_date      |         | price             |
| address           |         | delivered_date    |         +-------------------+
| city              |         | coupon_code       |                    ^
| postcode          |         | customer_id (FK)  |                    |
| email (unique)    |         +-------------------+                    |
+-------------------+                  ^                               |
           |                          /|\                              |
           |                           |                               |
           |                           |                               |
           |                           |                               |
           |   +------------------+    |                               |
           |   | order_product    |    |                               |
           |   +------------------+    |                               |
           o--<| order_id (FK, PK)|<--o                                |
               | product_id (FK, PK)|-----------------------------------o
               +------------------+
```

Creation of a new customer using flaskshell and visualize data through linux sqlite3

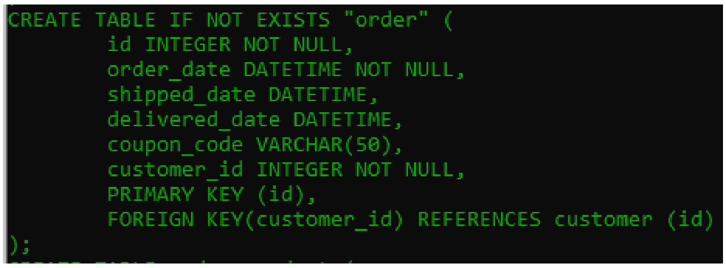


```
sqlite> select * from customer;
1|John|Doe|123 street|Miami|1324|jhonDoe@mail.com
sqlite>
```

```
(env) C:\Users\Paul Manriquez\Desktop\FlaskProyects\5_Flask_SQL_Alchemy>flask shell
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
App: app
Instance: C:\Users\Paul Manriquez\Desktop\FlaskProyects\5_Flask_SQL_Alchemy\instance
>>> from app import db,Product,Order,Customer
>>> johndoe=Customer(first_name='John', last_name='Doe',address='123 street',city='Miami',postcode='1324',email='jhonDoe@mail.com')
>>> johndoe
<Customer (transient 2305643268144)>
>>> db.session.add(johndoe)
>>> db.session.commit()
>>>
```

In SQL, certain words are considered reserved keywords because they are part of the SQL syntax. The word **"order"** is one such reserved keyword. These keywords are used for various SQL operations and functions, like ORDER BY, which is used to sort query results.When a table name or a column name is the same as a reserved keyword, it needs to be enclosed in double quotes or backticks (depending on the SQL dialect) to distinguish it as an identifier and not as a part of the SQL syntax.

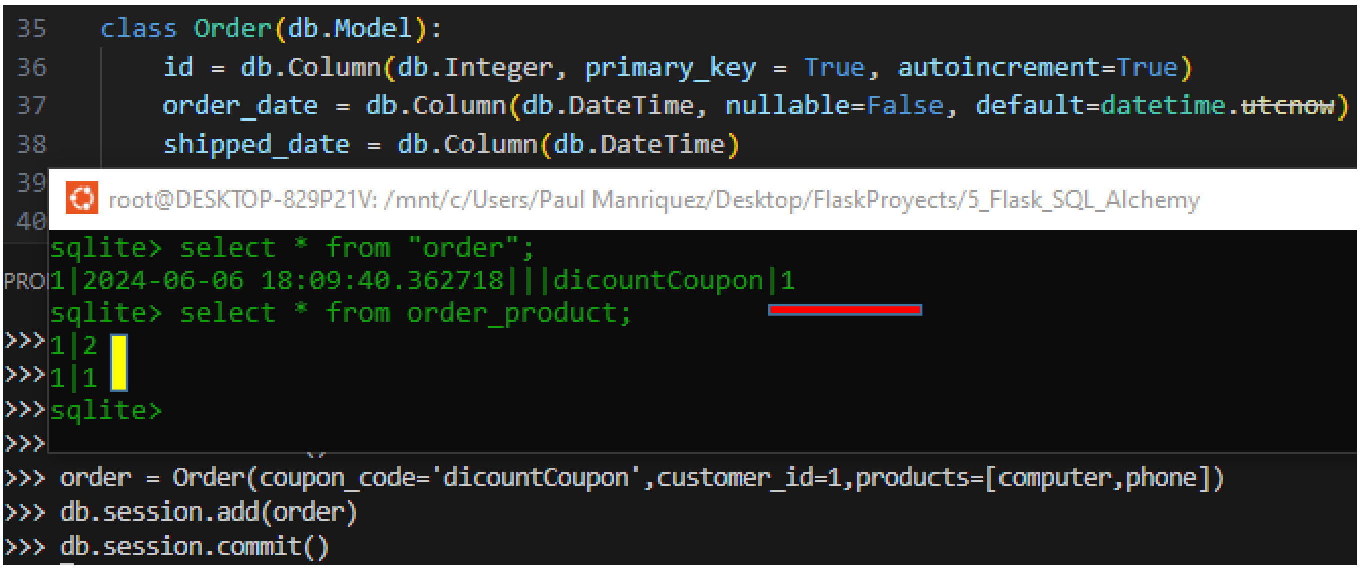
# Flask and FLASK SQL-Alchemy (NOTES)



```
CREATE TABLE IF NOT EXISTS "order" (
        id INTEGER NOT NULL,
        order_date DATETIME NOT NULL,
        shipped_date DATETIME,
        delivered_date DATETIME,
        coupon_code VARCHAR(50),
        customer_id INTEGER NOT NULL,
        PRIMARY KEY (id),
        FOREIGN KEY(customer_id) REFERENCES customer (id)
);
```

Creating a new order that is addressed to the custumer with the id=1 🟥

The costumer 1 has ordered the products with the id 2 and 1 (computer and phone) 🟨



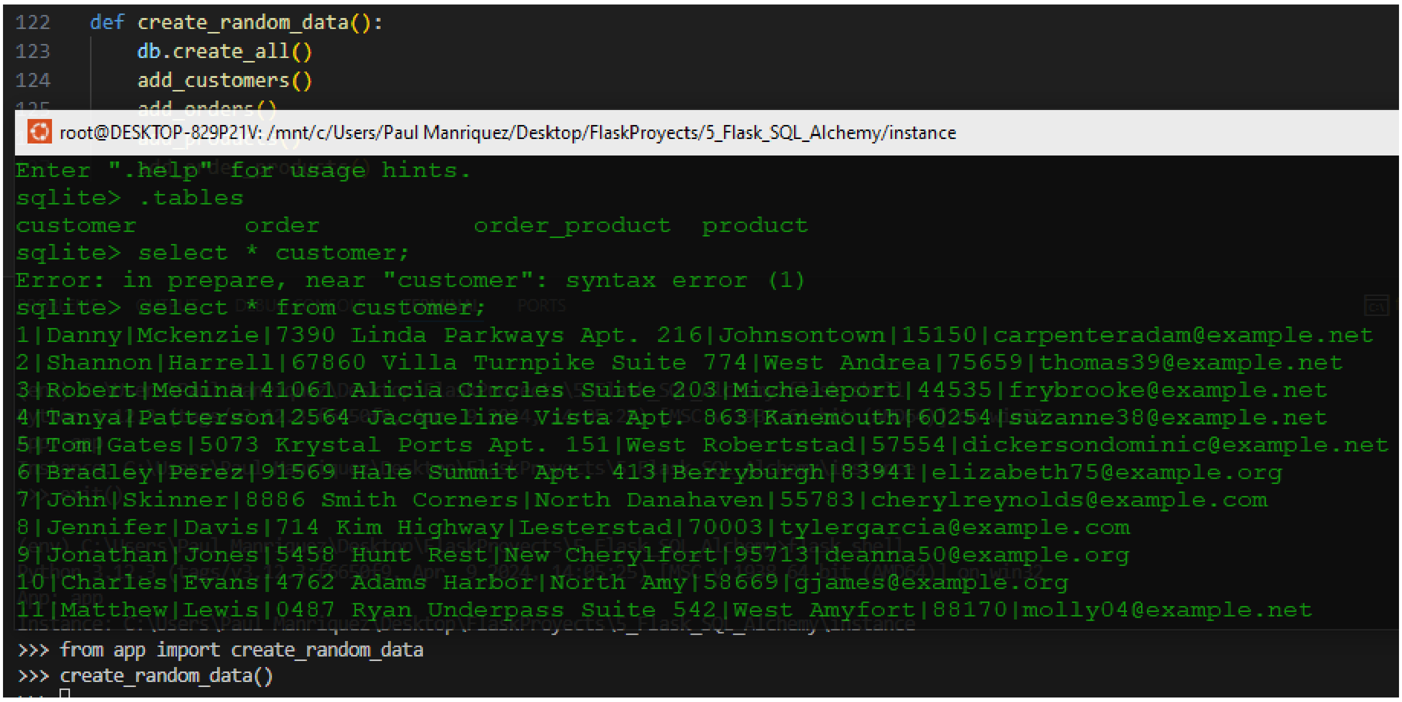
```
class Order(db.Model):
    id = db.Column(db.Integer, primary_key = True, autoincrement=True)
    order_date = db.Column(db.DateTime, nullable=False, default=datetime.~~utcnow~~)
    shipped_date = db.Column(db.DateTime)
```

root@DESKTOP-829P21V: /mnt/c/Users/Paul Manriquez/Desktop/FlaskProyects/5_Flask_SQL_Alchemy

```
sqlite> select * from "order";
1|2024-06-06 18:09:40.362718|||dicountCoupon|1
sqlite> select * from order_product;
1|2
1|1
sqlite>
```

```
>>> order = Order(coupon_code='dicountCoupon',customer_id=1,products=[computer,phone])
>>> db.session.add(order)
>>> db.session.commit()
```

Populating data through using fake module and a custom function



```
def create_random_data():
    db.create_all()
    add_customers()
    add_orders()
```

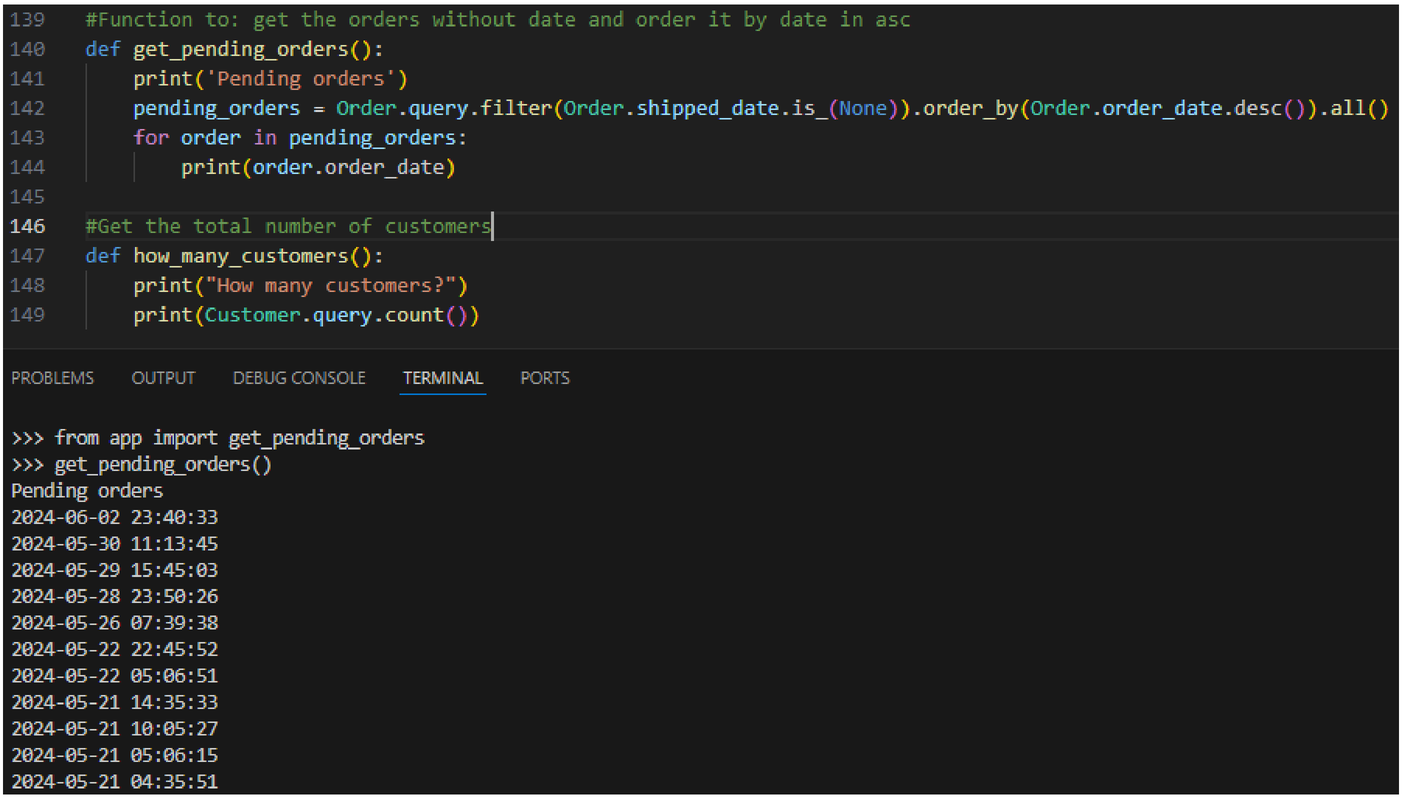
root@DESKTOP-829P21V: /mnt/c/Users/Paul Manriquez/Desktop/FlaskProyects/5_Flask_SQL_Alchemy/instance

```
Enter ".help" for usage hints.
sqlite> .tables
customer       order          order_product  product
sqlite> select * customer;
Error: in prepare, near "customer": syntax error (1)
sqlite> select * from customer;
1|Danny|Mckenzie|7390 Linda Parkways Apt. 216|Johnsontown|15150|carpenteradam@example.net
2|Shannon|Harrell|67860 Villa Turnpike Suite 774|West Andrea|75659|thomas39@example.net
3|Robert|Medina|41061 Alicia Circles Suite 203|Micheleport|44535|frybrooke@example.net
4|Tanya|Patterson|2564 Jacqueline Vista Apt. 863|Kanemouth|60254|suzanne38@example.net
5|Tom|Gates|5073 Krystal Ports Apt. 151|West Robertstad|57554|dickersondominic@example.net
6|Bradley|Perez|91569 Hale Summit Apt. 413|Berryburgh|83941|elizabeth75@example.org
7|John|Skinner|8886 Smith Corners|North Danahaven|55783|cherylreynolds@example.com
8|Jennifer|Davis|714 Kim Highway|Lesterstad|70003|tylergarcia@example.com
9|Jonathan|Jones|5458 Hunt Rest|New Cherylfort|95713|deanna50@example.org
10|Charles|Evans|4762 Adams Harbor|North Amy|58669|gjames@example.org
11|Matthew|Lewis|0487 Ryan Underpass Suite 542|West Amyfort|88170|molly04@example.net
```

```
>>> from app import create_random_data
>>> create_random_data()
```

# Flask and FLASK SQL-Alchemy (NOTES)

Making queries a functions on flask-alchemy



```python
#Function to: get the orders without date and order it by date in asc
def get_pending_orders():
    print('Pending orders')
    pending_orders = Order.query.filter(Order.shipped_date.is_(None)).order_by(Order.order_date.desc()).all()
    for order in pending_orders:
        print(order.order_date)

#Get the total number of customers
def how_many_customers():
    print("How many customers?")
    print(Customer.query.count())
```

```
>>> from app import get_pending_orders
>>> get_pending_orders()
Pending orders
2024-06-02 23:40:33
2024-05-30 11:13:45
2024-05-29 15:45:03
2024-05-28 23:50:26
2024-05-26 07:39:38
2024-05-22 22:45:52
2024-05-22 05:06:51
2024-05-21 14:35:33
2024-05-21 10:05:27
2024-05-21 05:06:15
2024-05-21 04:35:51
```