訂單管理資料庫

11011142曾鈺涵、11011209謝佳璇

專題目的

建立一個訂單管理的資料庫，用於管理產品、客戶、訂單、產品評價和客戶反饋。此系統旨在提供方便的商品管理和訂單處理功能，並支持多種數據查詢和報告功能。

主要功能包括：

- 產品管理：記錄產品的基本信息、價格和庫存等。
- 客戶管理：記錄客戶的基本信息、聯絡方式等。
- 訂單管理：管理客戶的訂單詳情。
- 產品評價：紀錄客戶對產品的評價，幫助改進產品質量。
- 客戶反饋：收集和處理客戶的意見和建議，提升客戶滿意度和服務品質。

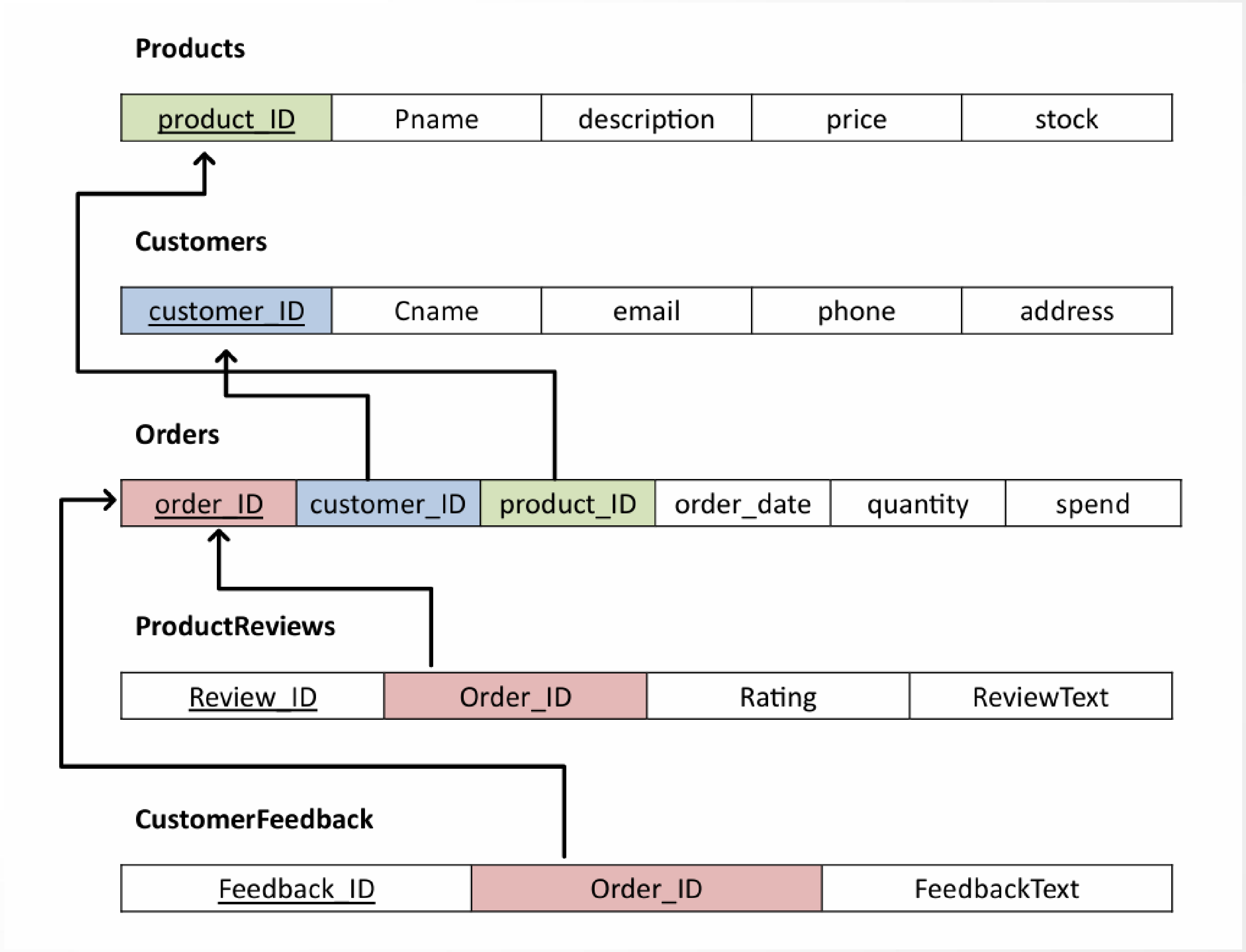
系統架構

- 設計數據庫的實體-關係模型和關聯模型，定義各個表格及其關係。

➢ 實體和關係：

- 產品（Products）
- 客戶（Customers）
- 訂單（Orders）
- 產品評價（ProductReviews）
- 客戶反饋（CustomerFeedback）

- **Relational Model**



Products

product_ID	Pname	description	price	stock

Customers

customer_ID	Cname	email	phone	address

Orders

order_ID	customer_ID	product_ID	order_date	quantity	spend

ProductReviews

Review_ID	Order_ID	Rating	ReviewText

CustomerFeedback

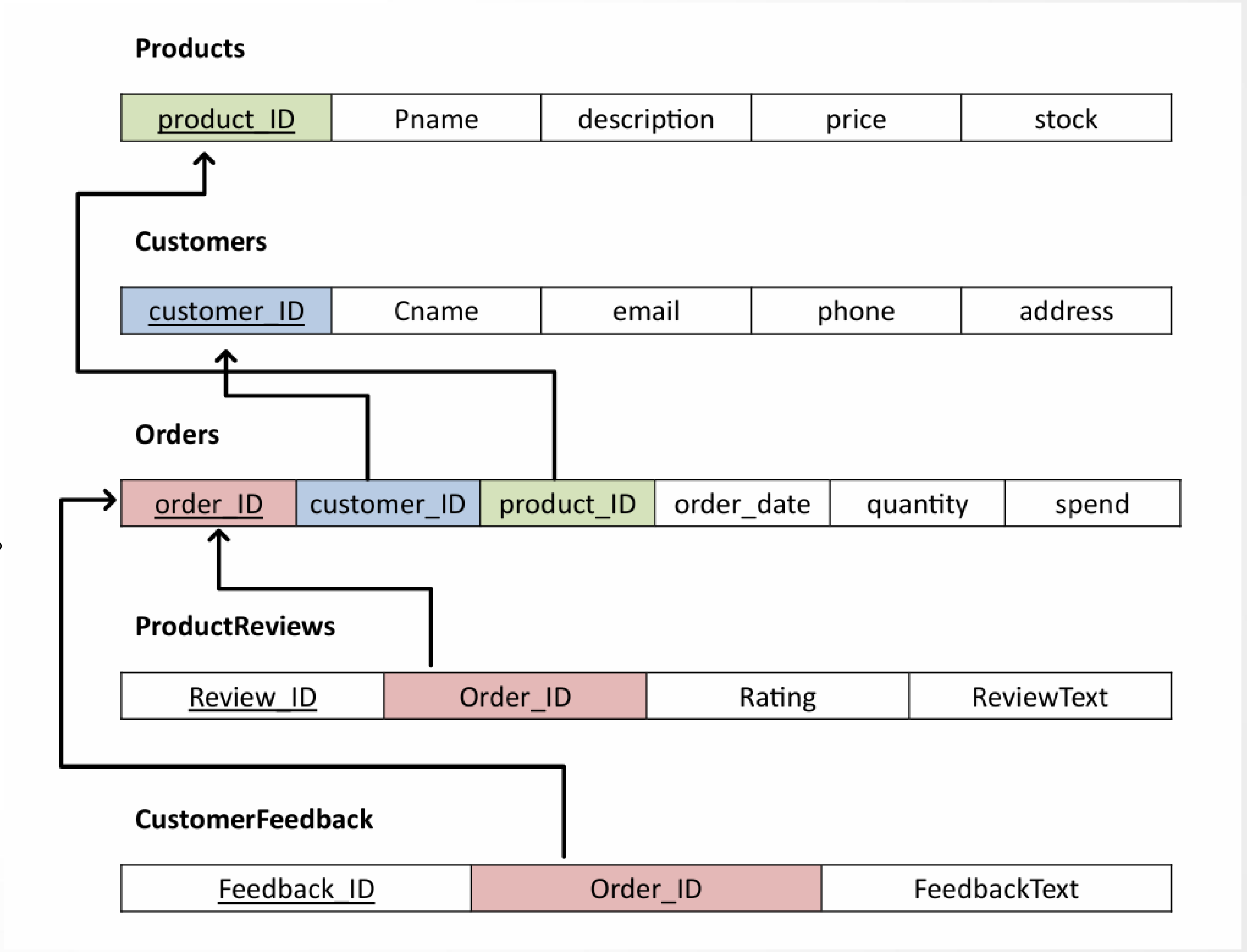
Feedback_ID	Order_ID	FeedbackText

系統架構

➢ 關係

- 一位客戶可以有多個訂單（1對多）。
- 一個訂單只會有一位客戶（1對1）。
- 一個訂單只會有一個產品（1對1）。
- 一個產品評價只會有一個訂單ID。（1對1）。
- 一筆訂單只會有一份產品評價（1對1）。
- 一筆訂單只會給一份客戶反饋（1對1）。

- **Relational Model**



創建Tables

首先我們連接到我們的資料庫，並創建需要的table，且所有表格皆不能為空。

- **Products**

product_ID ：整數，ID為主鍵，並設為自動遞增。

Pname：設定在50個字元以內。

description ：文字描述。

price ：設定10位數含小數點後兩位。

stock ：整數。

- **Customers**

customer_ID ：整數，ID為主鍵，並設為自動遞增。

Cname ：設定在50個字元以內。

email ：設定在50個字元以內。

phone ：設定在10個字元以內。

address ：文字描述。

創建Tables

- **Orders (此表格中皆不能為空)**

order_ID： 整數，ID為主鍵，並設為自動遞增。

customer_ID：整數，引用Customers表格中的customer_ID，當客戶被刪除時，相關訂單也會刪除。

product_ID：整數，引用Products表格中的product_ID ，當產品被刪除時，相關訂單也會刪除。

order_date： 訂單日期和時間。

quantity：整數，訂單產品數量。

spend：此筆訂單總花費，共十位數含小數點後兩位。

創建Tables

- **ProductReviews**

Review_ID ：整數，ID為主鍵，並設為自動遞增。

Order_ID：整數，引用Orders表格中的order_ID 。

Rating ：整數並評價1~5顆星。

ReviewText：文字描述產品品質。

- **CustomerFeedback**

Feedback_ID ：整數，ID為主鍵，並設為自動遞增。

Order_ID：整數，引用Orders表格中的order_ID 。

FeedbackText ： 文字描述服務品質。

```
import mysql.connector as mycont
from mysql.connector import errorcode
# set connection parameters
config={
    'user':'11011209',
    'password':'11011209',
    'host':'140.135.65.53',
    'database':'DB11011209',
    'buffered':True
    }

# connect to MySQL server
try:
  con=mycont.connect(**config,use_pure=True)
except mycont.Error as err:
  print(str(err))

# get cursor to perform query
cursor=con.cursor()

# 建立表格
tables = {}

tables['Products'] = (
    "CREATE TABLE Products ("
    "  product_ID INT AUTO_INCREMENT PRIMARY KEY,"
    "  Pname VARCHAR(50) NOT NULL,"
    "  description TEXT,"
    "  price DECIMAL(10, 2) NOT NULL,"
    "  stock INT NOT NULL"
    ") ENGINE=InnoDB"
)

tables['Customers'] = (
    "CREATE TABLE Customers ("
    "  customer_ID INT AUTO_INCREMENT PRIMARY KEY,"
    "  Cname VARCHAR(50) NOT NULL,"
    "  email VARCHAR(50) NOT NULL,"
    "  phone VARCHAR(10),"
    "  address TEXT"
    ") ENGINE=InnoDB"
)
```

```
tables['Orders'] = (
    "CREATE TABLE Orders ("
    "  order_ID INT AUTO_INCREMENT PRIMARY KEY,"
    "  customer_ID INT NOT NULL,"
    "  product_ID INT NOT NULL,"
    "  order_date DATETIME NOT NULL,"
    "  quantity INT NOT NULL,"
    "  spend DECIMAL(10, 2) NOT NULL,"
    "  FOREIGN KEY (customer_ID) REFERENCES Customers (customer_ID) ON DELETE CASCADE,"
    "  FOREIGN KEY (product_ID) REFERENCES Products (product_ID) ON DELETE CASCADE"
    ") ENGINE=InnoDB"
)
tables['ProductReviews'] = (
    "CREATE TABLE ProductReviews ("
    "  Review_ID INT AUTO_INCREMENT PRIMARY KEY,"
    "  Order_ID INT NOT NULL,"
    "  Rating INT CHECK (Rating >= 1 AND Rating <= 5),"
    "  ReviewText TEXT,"
    "  FOREIGN KEY (Order_ID) REFERENCES Orders(order_ID)  ON DELETE CASCADE"
    ") ENGINE=InnoDB"
)
tables['CustomerFeedback'] = (
    "CREATE TABLE CustomerFeedback ("
    "  Feedback_ID INT AUTO_INCREMENT PRIMARY KEY,"
    "  Order_ID INT NOT NULL,"
    "  FeedbackText TEXT NOT NULL,"
    "  FOREIGN KEY (Order_ID) REFERENCES Orders(order_ID) ON DELETE CASCADE"
    ") ENGINE=InnoDB"
)

# 建立每一個表格
for table_name, ddl in tables.items():
    try:
        print(f"Creating table {table_name}: ", end='')
        cursor.execute(ddl)
        print("OK")
    except mysql.connector.Error as err:
        print(f"Error: {err.msg}")

# Walrus operator(:=) is used here
# Read and execute SQL commands
while (stmt:=input())!=None:
  first_cmd=stmt.split()[0].lower() # get the first word and convert to lower case chars
  if first_cmd=="q": # end if get `q`
    break
  cursor.execute(stmt) # execute the SQL statement
  if first_cmd=="select":  # is a select command
    cols=cursor.column_names # get column names
    for col in cols: # print column names
      print(col)
  else: con.commit() # is not select command
  print(cursor.rowcount) # print how many rows are involved in this command
con.close() # close connection
```

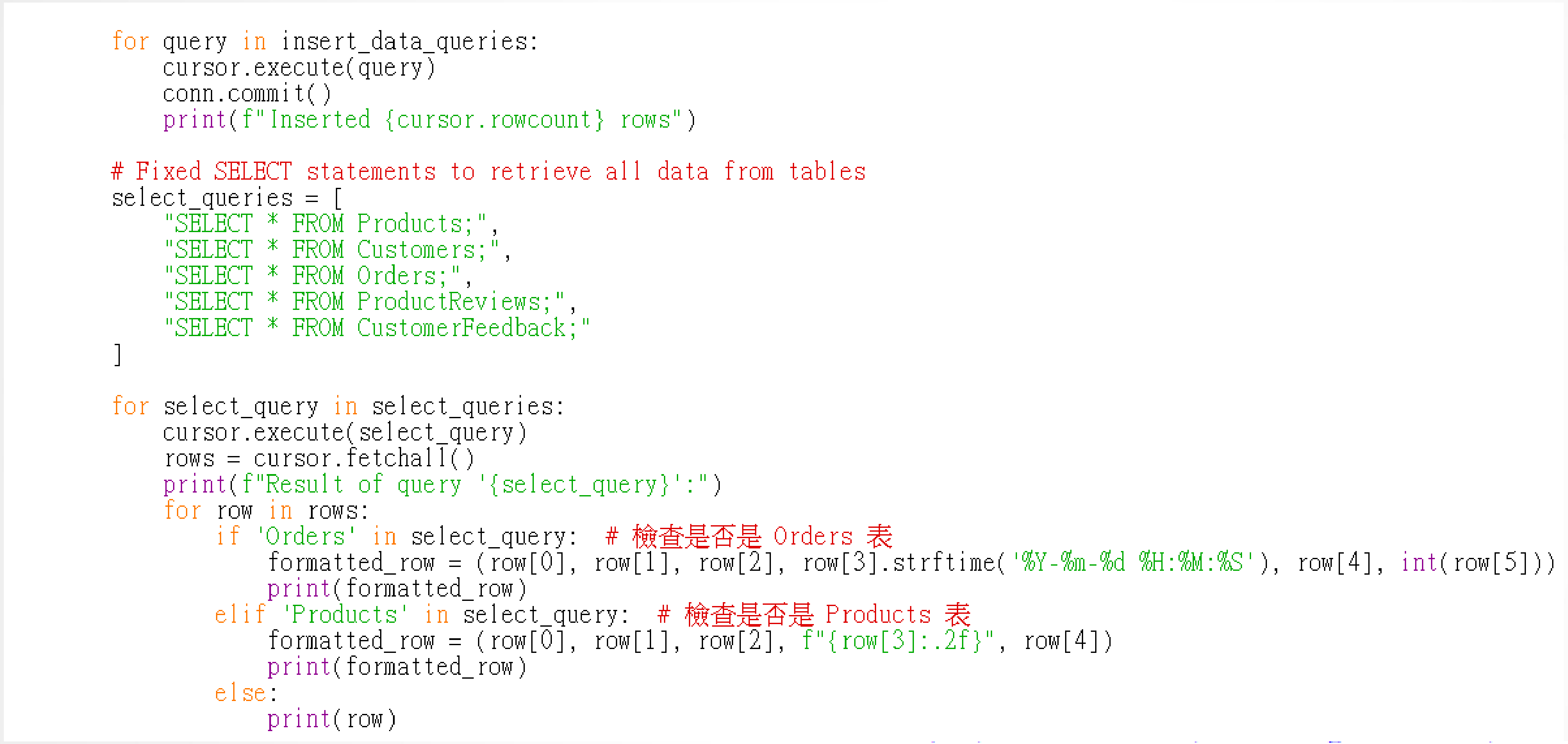
輸入數據

我們總共新增了20筆資料。我們設定10項產品、10位客戶，每人皆購買2項產品，所以總共有20筆訂單資料，以下是輸入的資料欄位:

- Products（產品名稱,顏色尺寸,價格,庫存量）
- Customers （客戶名稱,信箱,電話,地址）
- Orders （客戶ID,產品ID,訂單日期時間,購買數量,訂單總花費）
- ProductReviews （訂單ID, 1~5星,產品評價）
- CustomerFeedback （訂單ID, 服務反饋）

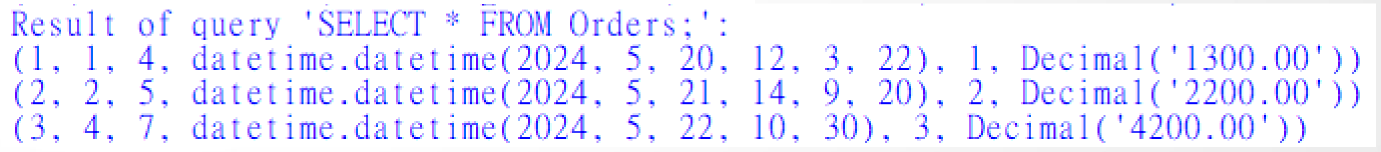
```
# 插入資料
insert_data_queries = [
    """
    INSERT INTO Products (Pname, description, price, stock) VALUES
    ('棒球帽', '藍F', 1200, 50),
    ('棒球帽', '紅F', 1200, 60),
    ('平沿帽', '黑F', 1300, 30),
    ('平沿帽', '灰F', 1300, 20),
    ('老帽', '藍F', 1100, 70),
    ('老帽', '綠F', 1100, 80),
    ('運動帽', '黃F', 1400, 90),
    ('運動帽', '紫F', 1400, 100),
    ('紳士帽', '黑F', 1600, 10),
    ('紳士帽', '白F', 1600, 15);
    """,
    """
    INSERT INTO Customers (Cname, email, phone, address) VALUES
   ('Mike Lin', 'mike@gmail.com', '0912345679', '101 Pine St'),
   ('Sarah Wang', 'sarah@gmail.com', '0987654322', '102 Maple St'),
   ('James Lee', 'james@gmail.com', '0912223334', '103 Cedar St'),
   ('Eva Liu', 'eva@gmail.com', '0912345680', '104 Birch St'),
   ('Kevin Wu', 'kevin@gmail.com', '0987654323', '105 Walnut St'),
   ('Nancy Chang', 'nancy@gmail.com', '0912223335', '106 Willow St'),
   ('Tony Huang', 'tony@gmail.com', '0912345681', '107 Poplar St'),
   ('Betty Ho', 'betty@gmail.com', '0987654324', '108 Cypress St'),
   ('Gary Chiu', 'gary@gmail.com', '0912223336', '109 Spruce St'),
   ('Alice Chen', 'alice@gmail.com', '0912345682', '110 Fir St');
    """,
    """
    INSERT INTO Orders (customer_ID, product_ID, order_date, quantity, spend) VALUES
    (1, 4, '2024-05-20 12:03:22', 1, 1300),
    (2, 5, '2024-05-21 14:09:20', 2, 2200),
    (4, 7, '2024-05-22 10:30:00', 3, 4200),
    (3, 6, '2024-05-22 10:30:00', 1, 1100),
    (5, 8, '2024-05-24 12:45:00', 2, 2800),
    (6, 9, '2024-05-25 13:15:00', 1, 1600),
    (7, 10, '2024-05-26 14:00:00', 1, 1600),
    (8, 1, '2024-05-27 15:30:00', 2, 2400),
    (9, 2, '2024-05-28 16:00:00', 3, 3600),
    (10, 3, '2024-05-29 17:45:00', 2, 2600),
    (1, 3, '2024-11-20 01:03:02', 1, 1300),
    (2, 4, '2024-02-21 13:09:56', 2, 2600),
    (3, 8, '2024-03-01 17:38:23', 1, 1400),
    (4, 10, '2024-01-23 18:27:40', 3, 4800),
    (5, 4, '2024-02-24 21:45:00', 2, 2600),
    (6, 5, '2024-10-25 11:15:00', 1, 1100),
    (7, 1, '2024-08-16 14:07:00', 1, 1200),
    (8, 9, '2024-06-27 23:30:23', 2, 3200),
    (9, 3, '2024-09-09 09:00:13', 3, 3900),
    (10, 2, '2024-05-01 10:45:00', 2, 2400);
    """,
    """
    INSERT INTO ProductReviews (Order_ID,Rating, ReviewText) VALUES
    (1,5,'非常滿意!'),
    (2,4,'好商品!'),
    (3,5,'質量很好!'),
    (4,3,'不錯的選擇'),
    (5,4,'挺好的!'),
    (6,5,'很喜歡!'),
    (7,4,'物超所值!'),
    (8,3,'可以接受'),
    (9,4,'物有所值!'),
    (10,5,'非常棒!'),
    (11,1,'差評!'),
    (12,1,'不再回購!'),
    (13,5,'質量很好!'),
    (14,3,'不錯的選擇'),
    (15,2,'還可以!'),
    (16,5,'很喜歡!'),
    (17,4,'cp值高!'),
    (18,3,'可以接受'),
    (19,4,'物有所值!'),
    (20,5,'good!');
    """,
    """
    INSERT INTO CustomerFeedback (Order_ID, FeedbackText) VALUES
    (1,'價格合理'),
    (2,'物流效率高'),
    (3,'服務周到'),
    (4,'客服態度好!'),
    (5,'物流很快!'),
    (6,'產品包裝不錯'),
    (7,'商品描述詳細'),
    (8, '服務態度很好'),
    (9,'出貨迅速'),
    (10,'商品品質好'),
    (11,'不要!'),
    (12,'物流很快!'),
    (13,'產品包裝不錯'),
    (14,'商品描述詳細'),
    (15,'服務態度很好'),
    (16,'出貨迅速'),
    (17,'商品品質好'),
    (18,'價格合理'),
    (19,'物流效率高'),
    (20,'服務周到');
    """
]
```

輸出固定select 且格式化產品的價格、訂單的日期時間和訂單總花費



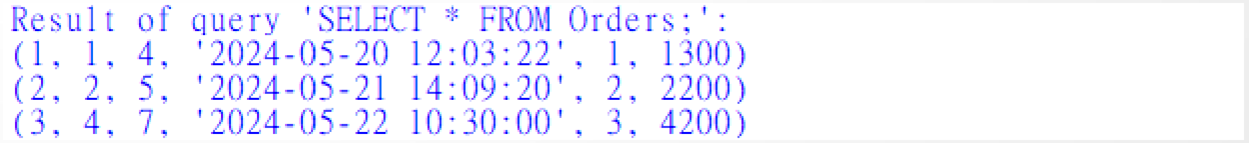
```python
for query in insert_data_queries:
    cursor.execute(query)
    conn.commit()
    print(f"Inserted {cursor.rowcount} rows")

# Fixed SELECT statements to retrieve all data from tables
select_queries = [
    "SELECT * FROM Products;",
    "SELECT * FROM Customers;",
    "SELECT * FROM Orders;",
    "SELECT * FROM ProductReviews;",
    "SELECT * FROM CustomerFeedback;"
]

for select_query in select_queries:
    cursor.execute(select_query)
    rows = cursor.fetchall()
    print(f"Result of query '{select_query}':")
    for row in rows:
        if 'Orders' in select_query:  # 檢查是否是 Orders 表
            formatted_row = (row[0], row[1], row[2], row[3].strftime('%Y-%m-%d %H:%M:%S'), row[4], int(row[5]))
            print(formatted_row)
        elif 'Products' in select_query:  # 檢查是否是 Products 表
            formatted_row = (row[0], row[1], row[2], f"{row[3]:.2f}", row[4])
            print(formatted_row)
        else:
            print(row)
```

格式化前:



```
Result of query 'SELECT * FROM Orders;':
(1, 1, 4, datetime.datetime(2024, 5, 20, 12, 3, 22), 1, Decimal('1300.00'))
(2, 2, 5, datetime.datetime(2024, 5, 21, 14, 9, 20), 2, Decimal('2200.00'))
(3, 4, 7, datetime.datetime(2024, 5, 22, 10, 30), 3, Decimal('4200.00'))
```

格式化後:



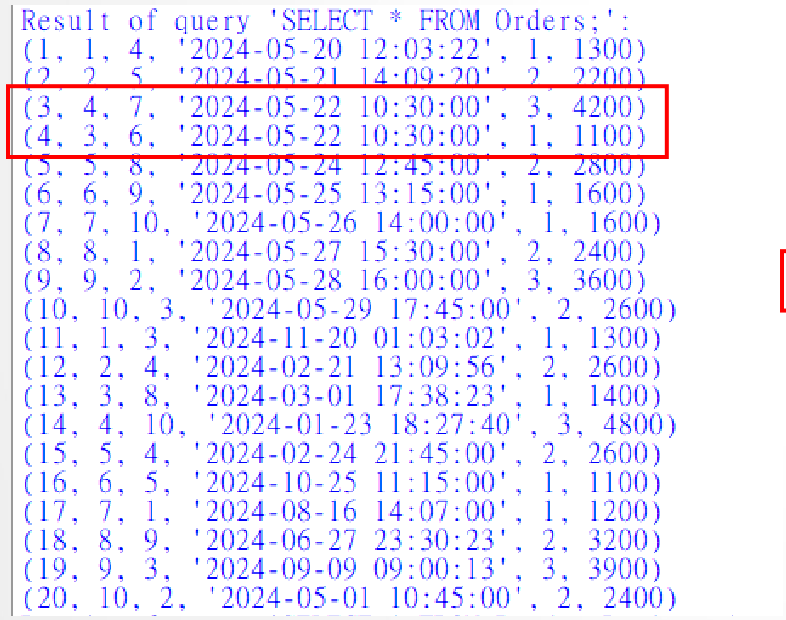
```
Result of query 'SELECT * FROM Orders;':
(1, 1, 4, '2024-05-20 12:03:22', 1, 1300)
(2, 2, 5, '2024-05-21 14:09:20', 2, 2200)
(3, 4, 7, '2024-05-22 10:30:00', 3, 4200)
```

執行不特定select指令

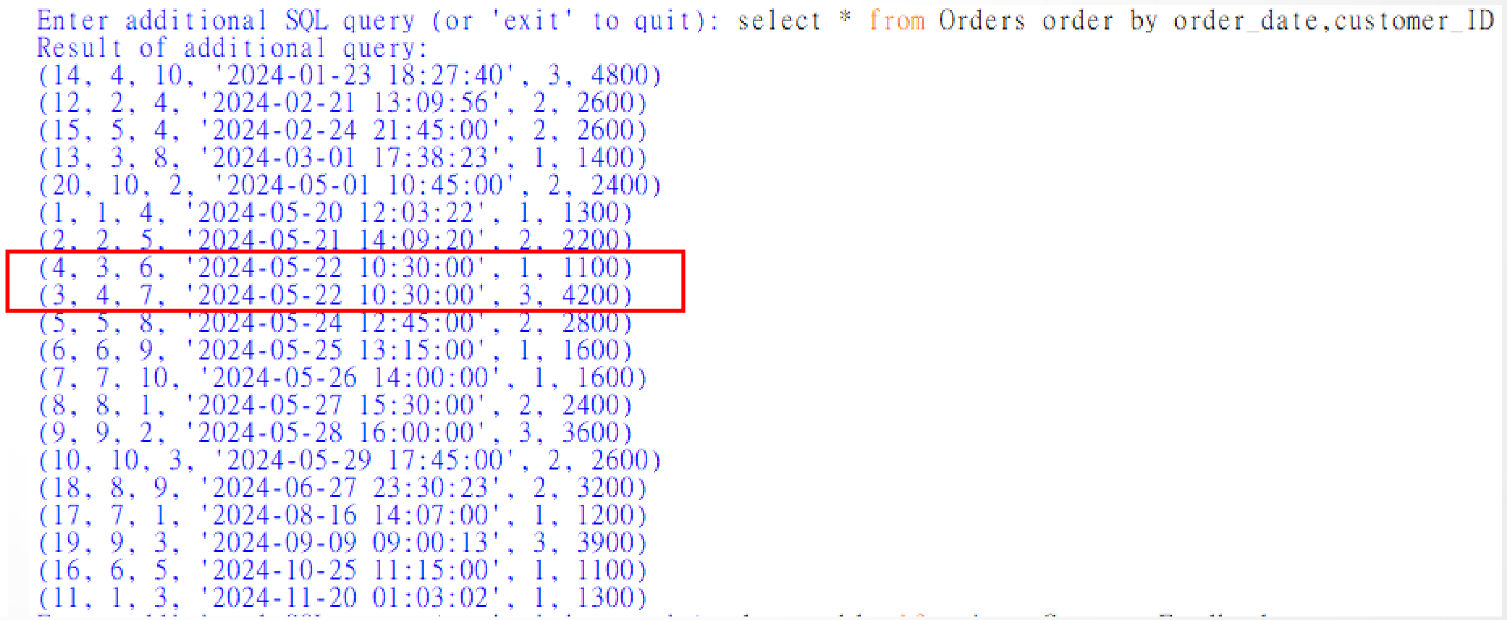
我們使用**order by** 先將Orders 的訂單時間由遠到近排列，再將Orders 的客戶ID由小排到大。

在相同時間的訂單(2024-05-22 10:30:00)會依客戶ID 進行排序。

格式 : Orders（客戶ID,產品ID,訂單日期時間,購買數量,訂單總花費）



```
Result of query 'SELECT * FROM Orders;':
(1, 1, 4, '2024-05-20 12:03:22', 1, 1300)
(2, 2, 5, '2024-05-21 14:09:20', 2, 2200)
(3, 4, 7, '2024-05-22 10:30:00', 3, 4200)
(4, 3, 6, '2024-05-22 10:30:00', 1, 1100)
(5, 5, 8, '2024-05-24 12:45:00', 2, 2800)
(6, 6, 9, '2024-05-25 13:15:00', 1, 1600)
(7, 7, 10, '2024-05-26 14:00:00', 1, 1600)
(8, 8, 1, '2024-05-27 15:30:00', 2, 2400)
(9, 9, 2, '2024-05-28 16:00:00', 3, 3600)
(10, 10, 3, '2024-05-29 17:45:00', 2, 2600)
(11, 1, 3, '2024-11-20 01:03:02', 1, 1300)
(12, 2, 4, '2024-02-21 13:09:56', 2, 2600)
(13, 3, 8, '2024-03-01 17:38:23', 1, 1400)
(14, 4, 10, '2024-01-23 18:27:40', 3, 4800)
(15, 5, 4, '2024-02-24 21:45:00', 2, 2600)
(16, 6, 5, '2024-10-25 11:15:00', 1, 1100)
(17, 7, 1, '2024-08-16 14:07:00', 1, 1200)
(18, 8, 9, '2024-06-27 23:30:23', 2, 3200)
(19, 9, 3, '2024-09-09 09:00:13', 3, 3900)
(20, 10, 2, '2024-05-01 10:45:00', 2, 2400)
```

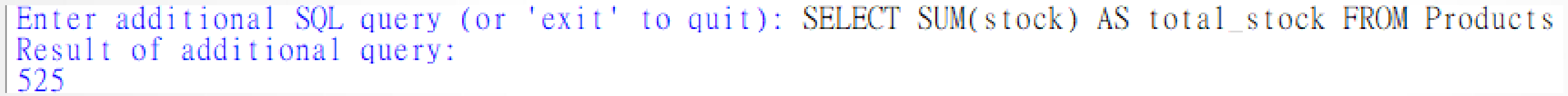


```
Enter additional SQL query (or 'exit' to quit): select * from Orders order by order_date,customer_ID
Result of additional query:
(14, 4, 10, '2024-01-23 18:27:40', 3, 4800)
(12, 2, 4, '2024-02-21 13:09:56', 2, 2600)
(15, 5, 4, '2024-02-24 21:45:00', 2, 2600)
(13, 3, 8, '2024-03-01 17:38:23', 1, 1400)
(20, 10, 2, '2024-05-01 10:45:00', 2, 2400)
(1, 1, 4, '2024-05-20 12:03:22', 1, 1300)
(2, 2, 5, '2024-05-21 14:09:20', 2, 2200)
(4, 3, 6, '2024-05-22 10:30:00', 1, 1100)
(3, 4, 7, '2024-05-22 10:30:00', 3, 4200)
(5, 5, 8, '2024-05-24 12:45:00', 2, 2800)
(6, 6, 9, '2024-05-25 13:15:00', 1, 1600)
(7, 7, 10, '2024-05-26 14:00:00', 1, 1600)
(8, 8, 1, '2024-05-27 15:30:00', 2, 2400)
(9, 9, 2, '2024-05-28 16:00:00', 3, 3600)
(10, 10, 3, '2024-05-29 17:45:00', 2, 2600)
(18, 8, 9, '2024-06-27 23:30:23', 2, 3200)
(17, 7, 1, '2024-08-16 14:07:00', 1, 1200)
(19, 9, 3, '2024-09-09 09:00:13', 3, 3900)
(16, 6, 5, '2024-10-25 11:15:00', 1, 1100)
(11, 1, 3, '2024-11-20 01:03:02', 1, 1300)
```

額外指令

- **SELECT SUM**

我們先用select sum 統計產品(Product)的總庫存量，



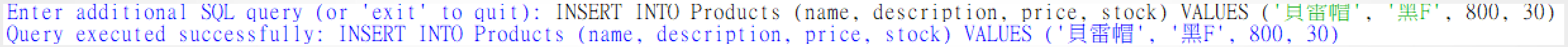
```
Enter additional SQL query (or 'exit' to quit): SELECT SUM(stock) AS total_stock FROM Products
Result of additional query:
525
```

➔得知現在總庫存量為525個。

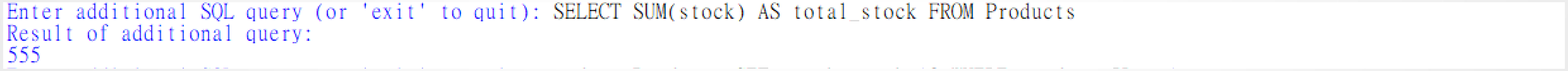
- **INSERT**

用insert插入一個新產品'貝雷帽、黑色F、售價800、30個'，插入後再次使用select sum統計總庫存量，確定是否有將貝雷帽的資訊插入Products。

➔原先的總庫存量從525 ⟹ 555，增加了30個。



```
Enter additional SQL query (or 'exit' to quit): INSERT INTO Products (name, description, price, stock) VALUES ('貝雷帽', '黑F', 800, 30)
Query executed successfully: INSERT INTO Products (name, description, price, stock) VALUES ('貝雷帽', '黑F', 800, 30)
```



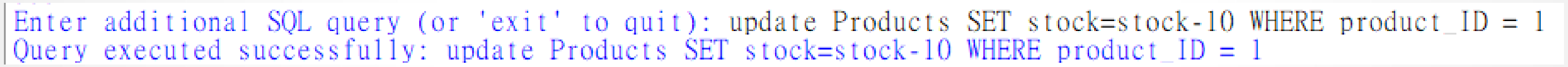
```
Enter additional SQL query (or 'exit' to quit): SELECT SUM(stock) AS total_stock FROM Products
Result of additional query:
555
```

額外指令

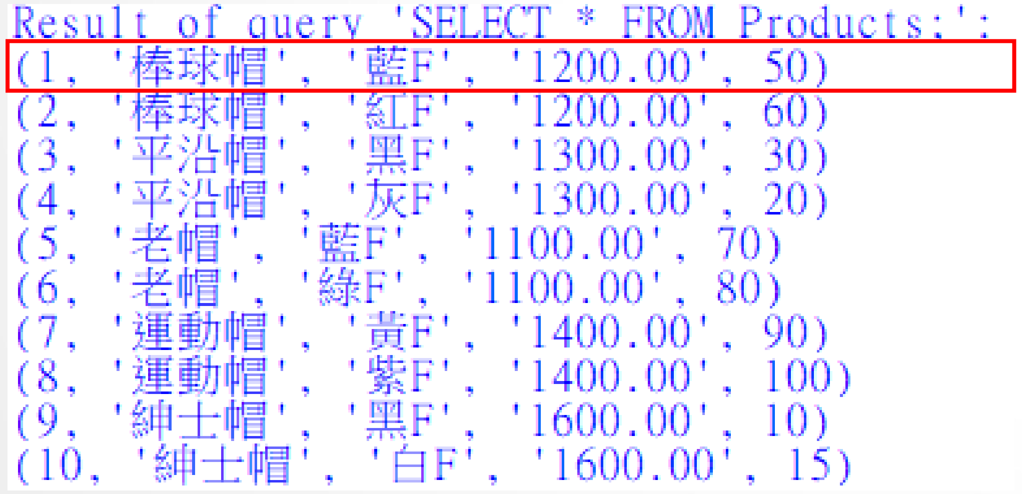
- **UPDATE**

我們輸入 update 將產品 Product_ID 為 1 的庫存量 減少 10個。

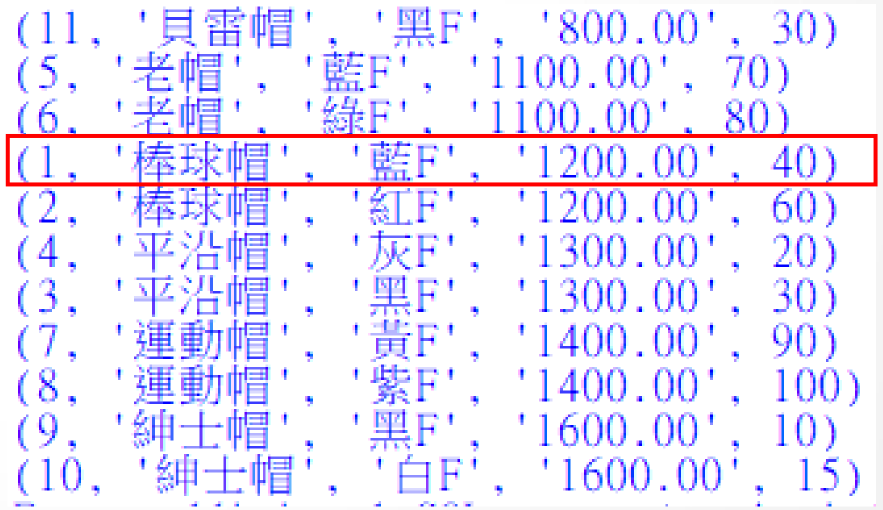
所以Product_ID為 1的產品原庫存為 50 個，減 10 個後更新為 40 個。



```
Enter additional SQL query (or 'exit' to quit): update Products SET stock=stock-10 WHERE product_ID = 1
Query executed successfully: update Products SET stock=stock-10 WHERE product_ID = 1
```



```
Result of query 'SELECT * FROM Products;':
(1, '棒球帽', '藍F', '1200.00', 50)
(2, '棒球帽', '紅F', '1200.00', 60)
(3, '平沿帽', '黑F', '1300.00', 30)
(4, '平沿帽', '灰F', '1300.00', 20)
(5, '老帽', '藍F', '1100.00', 70)
(6, '老帽', '綠F', '1100.00', 80)
(7, '運動帽', '黃F', '1400.00', 90)
(8, '運動帽', '紫F', '1400.00', 100)
(9, '紳士帽', '黑F', '1600.00', 10)
(10, '紳士帽', '白F', '1600.00', 15)
```



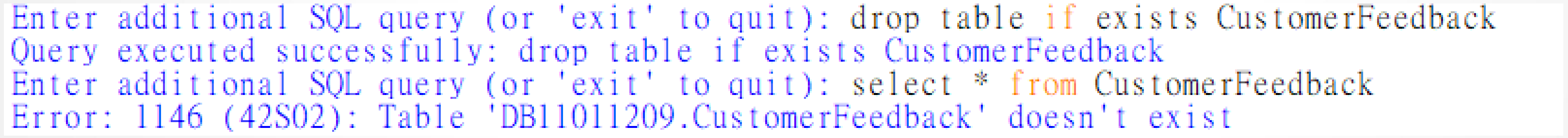
```
(11, '貝雷帽', '黑F', '800.00', 30)
(5, '老帽', '藍F', '1100.00', 70)
(6, '老帽', '綠F', '1100.00', 80)
(1, '棒球帽', '藍F', '1200.00', 40)
(2, '棒球帽', '紅F', '1200.00', 60)
(4, '平沿帽', '灰F', '1300.00', 20)
(3, '平沿帽', '黑F', '1300.00', 30)
(7, '運動帽', '黃F', '1400.00', 90)
(8, '運動帽', '紫F', '1400.00', 100)
(9, '紳士帽', '黑F', '1600.00', 10)
(10, '紳士帽', '白F', '1600.00', 15)
```

額外指令

- **DROP TABLE**

我們使用drop table先把 CustomerFeedback 此表格刪除(drop)，

再利用 select 確認是否已經將 CustomerFeedback 表格刪除。



```
Enter additional SQL query (or 'exit' to quit): drop table if exists CustomerFeedback
Query executed successfully: drop table if exists CustomerFeedback
Enter additional SQL query (or 'exit' to quit): select * from CustomerFeedback
Error: 1146 (42S02): Table 'DB11011209.CustomerFeedback' doesn't exist
```

謝謝