

Psycopg2

- Create a connection with PostgreSQL database from Python.
- Create a cursor from the connection.
- Create a table product with fields as id(primary key) automatically fetching value from sequence, name, cost price, sale price. Here name must be null and sale price must be greater than cost price.
- Create a table category with id(primary key), code and name where code must be unique and name must not be null.
- Add a foreign key for category in product table.
- Insert 20 different realistic products.
- Fetch all the products.
- Fetch first product in the list.
- Fetch all the products and order by cost price.
 - Fetch the product with the highest sale price.
- Update cost price of at least 3 products.
- Update cost price of at least 2 products.
- Fetch all the records with updated values.
- Fetch id, name, cost price, sale price, category code, category name for all the products.
- Close a cursor

- Close a connection

Code:

```
import psycopg2
```

```
#1
```

```
try:
```

```
    conn =  
    psycopg2.connect(dbname="exercise3",user="postgres",password="123",host="localhost",port="5432")
```

```
    print("Connection with database established successfully.")
```

```
except:
```

```
    print("Connection with database has not been established")
```

```
#2
```

```
try:
```

```
    cur = conn.cursor()  
    print("\nCursor created successfully")
```

```
except:
```

```
    print("\nCursor has not been created")
```

```
#3
```

```
try:
```

```
    cur.execute("create table if not exists product(id serial primary key,name  
varchar,cost_price numeric,sale_price numeric)")
```

```
    conn.commit()
```

```
    print("\nProduct table created successfully")
```

```
except:
```

```
    print("\nProduct table has not been created")
```

```
#4
```

try:

```
cur.execute("create table if not exists category(id serial primary key,code  
varchar unique,name varchar not null)")
```

```
conn.commit()
```

```
print("\nCategory table created successfully")
```

except:

```
print("\nCategory table has not been created")
```

#5

try:

```
cur.execute("alter table product add column category_id integer references  
category(id)")
```

```
conn.commit()
```

```
print("\nCategory ID column added to Product table successfully with foreign  
key constraint")
```

except:

```
print("\nError occurred while adding Category ID column to Product table")
```

#6

try:

```
categories = [  
    ("C001", "Electronics"),  
    ("C002", "Clothing"),  
    ("C003", "Home Appliances"),  
    ("C004", "Books"),  
    ("C005", "Toys")  
]
```

```
cur.executemany("insert into category (code,name) values(%s,%s)",categories)
```

```
conn.commit()
```

```
print("\nData inserted to category table")
```

```

products = [
    ("Smartphone", 399.99, 499.99, 1),
    ("Laptop", 799.99, 999.99, 2),
    ("Headphones", 49.99, 69.99, 3),
    ("Bluetooth Speaker", 29.99, 39.99, 4),
    ("Tablet", 299.99, 399.99, 5),
    ("Smartwatch", 199.99, 249.99, 1),
    ("Gaming Console", 299.99, 399.99, 2),
    ("Digital Camera", 249.99, 349.99, 3),
    ("Fitness Tracker", 79.99, 99.99, 4),
    ("Wireless Earbuds", 69.99, 89.99, 5),
    ("External Hard Drive", 79.99, 119.99, 1),
    ("Monitor", 149.99, 199.99, 2),
    ("Printer", 99.99, 149.99, 3),
    ("Keyboard", 29.99, 49.99, 4),
    ("Mouse", 19.99, 29.99, 5),
    ("Router", 59.99, 79.99, 1),
    ("Webcam", 39.99, 59.99, 2),
    ("Graphics Tablet", 149.99, 199.99, 3),
    ("Microphone", 49.99, 69.99, 4),
    ("Portable Charger", 19.99, 29.99, 5)
]

```

```

    cur.executemany("Insert into product(name,cost_price,sale_price,category_id)
values(%s,%s,%s,%s)",products)
    conn.commit()
    print("\nData inserted to Product table")
except:
    print("\nRecords failed to be inserted")

```

#7

```
cur.execute("select * from product")
```

```
print("\n",cur.fetchall())
```

#8

```
cur.execute("select * from product limit 1")
```

```
print("\n",cur.fetchone())
```

#9

```
cur.execute("select * from product order by cost_price")
```

```
print("\n",cur.fetchall())
```

#10

```
cur.execute("select * from product order by sale_price desc limit 1")
```

```
print("\n",cur.fetchone())
```

#11

```
cur.execute("SELECT id FROM product LIMIT 3")
```

```
product_ids_to_update = cur.fetchall()
```

```
for product_id in product_ids_to_update:
```

```
    cur.execute("UPDATE product SET cost_price = cost_price + 5 WHERE id =  
    %s", (product_id,))
```

```
print("\nRecord updated successfully")
```

#12

```
cur.execute("SELECT id FROM product LIMIT 2")
```

```
product_ids_to_update = cur.fetchall()
```

```
for product_id in product_ids_to_update:
```

```
    cur.execute("UPDATE product SET cost_price = cost_price + 10 WHERE id =  
    %s", (product_id,))
```

```
print("\nRecord updated successfully")
```

#13

```
cur.execute("select * from product")
```

```
print("\n",cur.fetchall())
```

#14

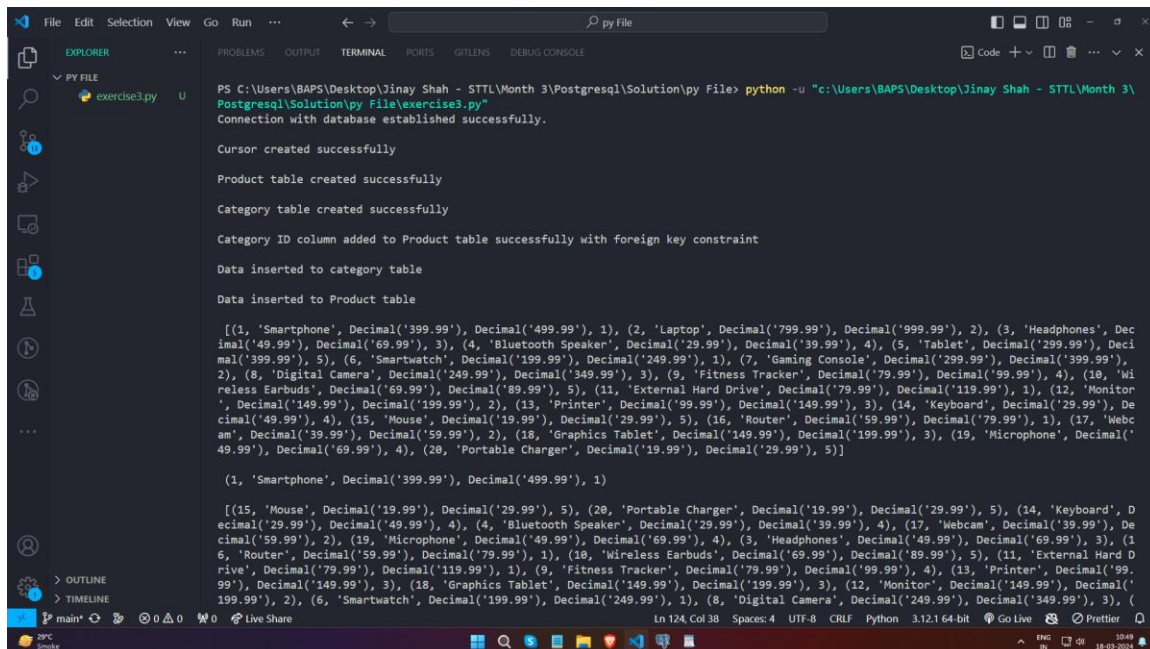
```
cur.execute("SELECT p.id, p.name, p.cost_price, p.sale_price, c.code, c.name
FROM product p JOIN category c ON p.category_id = c.id")
print(cur.fetchall())
```

#15

```
cur.close()
print("\nCursor closed successfully")
```

```
conn.close()
print("\nConnection closed successfully")
```

Output:



```
PS C:\Users\BAPS\Desktop\Jinay Shah - STTL\Month 3\Postgresql\Solution\py File> python -u "c:\Users\BAPS\Desktop\Jinay Shah - STTL\Month 3\Postgresql\Solution\py File\exercise3.py"
Connection with database established successfully.

Cursor created successfully

Product table created successfully

Category table created successfully

Category ID column added to Product table successfully with foreign key constraint

Data inserted to category table

Data inserted to Product table

[(1, 'Smartphone', Decimal('399.99'), Decimal('499.99'), 1), (2, 'Laptop', Decimal('799.99'), Decimal('999.99'), 2), (3, 'Headphones', Decimal('49.99'), Decimal('69.99'), 3), (4, 'Bluetooth Speaker', Decimal('29.99'), Decimal('39.99'), 4), (5, 'Tablet', Decimal('299.99'), Decimal('399.99'), 5), (6, 'Smartwatch', Decimal('199.99'), Decimal('249.99'), 1), (7, 'Gaming Console', Decimal('299.99'), Decimal('399.99'), 2), (8, 'Digital Camera', Decimal('249.99'), Decimal('349.99'), 3), (9, 'Fitness Tracker', Decimal('79.99'), Decimal('99.99'), 4), (10, 'Wireless Earbuds', Decimal('69.99'), Decimal('89.99'), 5), (11, 'External Hard Drive', Decimal('79.99'), Decimal('119.99'), 1), (12, 'Monitor', Decimal('149.99'), Decimal('199.99'), 2), (13, 'Printer', Decimal('99.99'), Decimal('149.99'), 3), (14, 'Keyboard', Decimal('29.99'), Decimal('49.99'), 4), (15, 'Mouse', Decimal('19.99'), Decimal('29.99'), 5), (16, 'Router', Decimal('59.99'), Decimal('79.99'), 1), (17, 'Webcam', Decimal('39.99'), Decimal('59.99'), 2), (18, 'Graphics Tablet', Decimal('149.99'), Decimal('199.99'), 3), (19, 'Microphone', Decimal('49.99'), Decimal('69.99'), 4), (20, 'Portable Charger', Decimal('19.99'), Decimal('29.99'), 5)]

(1, 'Smartphone', Decimal('399.99'), Decimal('499.99'), 1)

[(15, 'Mouse', Decimal('19.99'), Decimal('29.99'), 5), (20, 'Portable Charger', Decimal('19.99'), Decimal('29.99'), 5), (14, 'Keyboard', Decimal('29.99'), Decimal('49.99'), 4), (4, 'Bluetooth Speaker', Decimal('29.99'), Decimal('39.99'), 4), (17, 'Webcam', Decimal('39.99'), Decimal('59.99'), 2), (19, 'Microphone', Decimal('49.99'), Decimal('69.99'), 4), (3, 'Headphones', Decimal('49.99'), Decimal('69.99'), 3), (16, 'Router', Decimal('59.99'), Decimal('79.99'), 1), (10, 'Wireless Earbuds', Decimal('69.99'), Decimal('89.99'), 5), (11, 'External Hard Drive', Decimal('79.99'), Decimal('119.99'), 1), (9, 'Fitness Tracker', Decimal('79.99'), Decimal('99.99'), 4), (13, 'Printer', Decimal('99.99'), Decimal('149.99'), 3), (18, 'Graphics Tablet', Decimal('149.99'), Decimal('199.99'), 3), (12, 'Monitor', Decimal('149.99'), Decimal('199.99'), 2), (6, 'Smartwatch', Decimal('199.99'), Decimal('249.99'), 1), (8, 'Digital Camera', Decimal('249.99'), Decimal('349.99'), 3), (5, 'Tablet', Decimal('299.99'), Decimal('399.99'), 5), (7, 'Gaming Console', Decimal('299.99'), Decimal('399.99'), 2)]
```