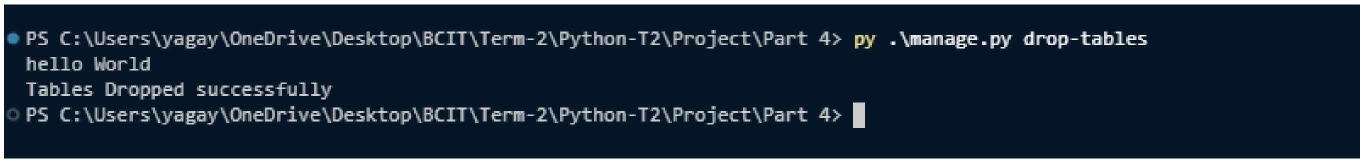
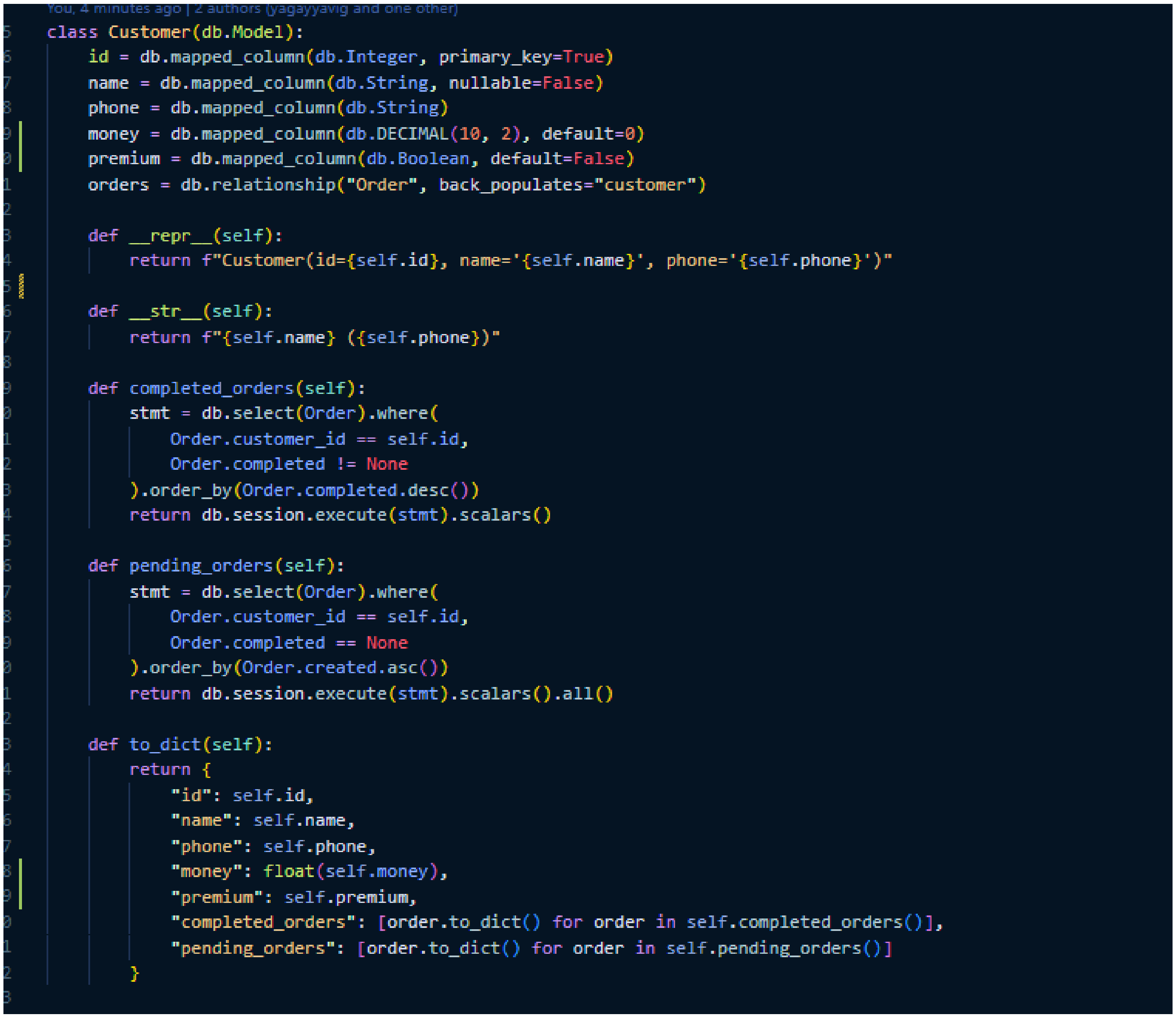
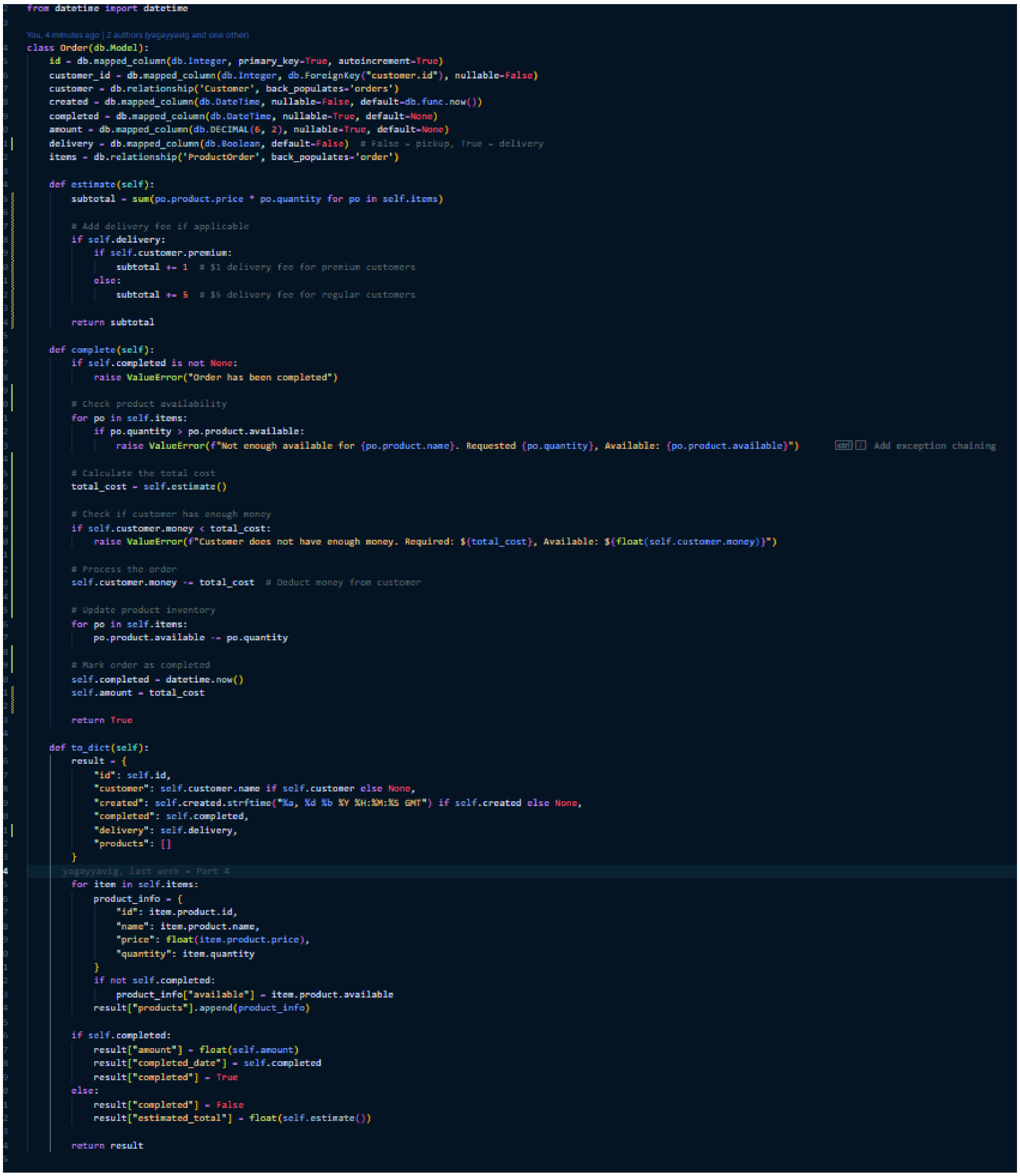
Python final submission



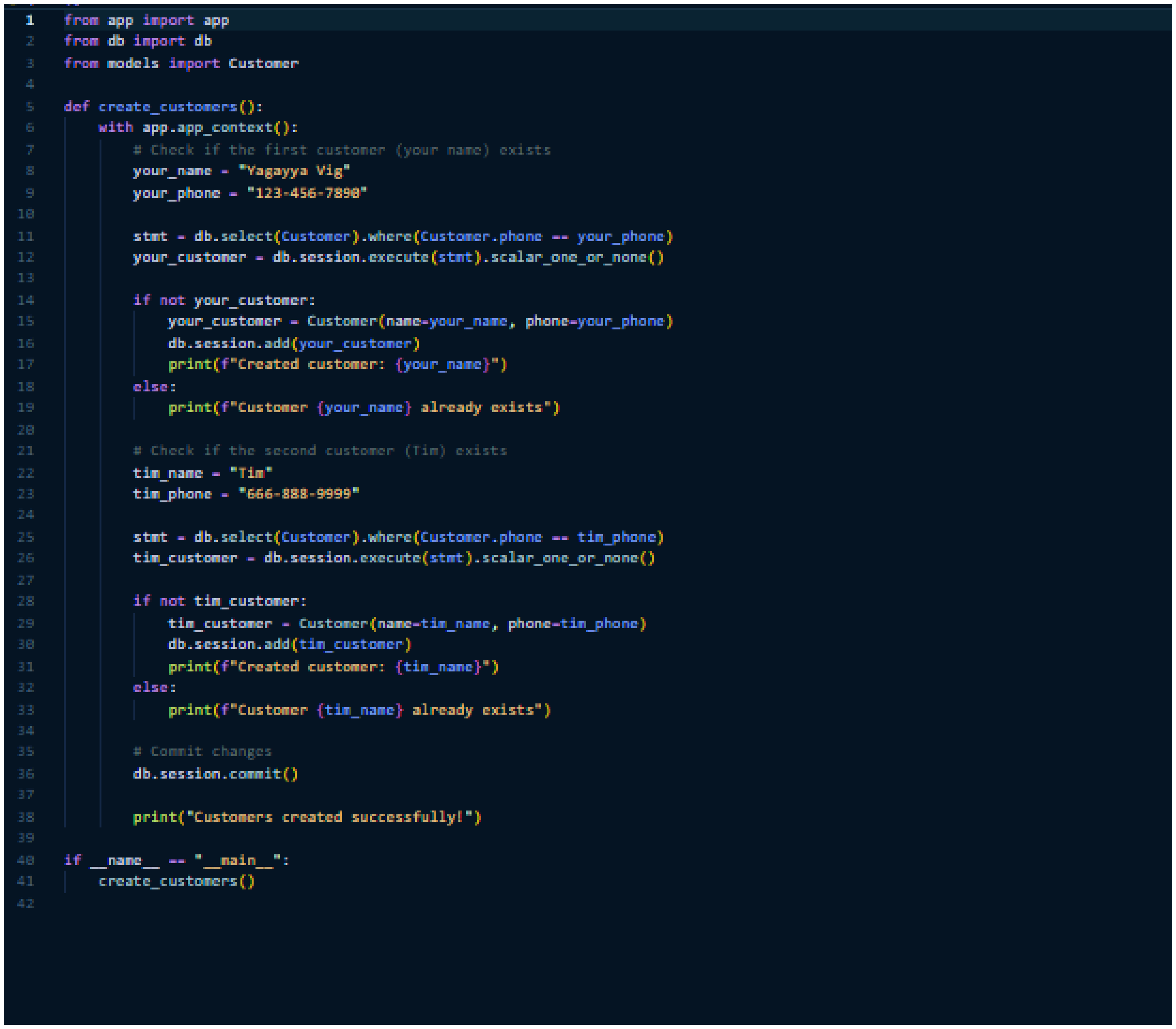
```
PS C:\Users\yagay\OneDrive\Desktop\BCIT\Term-2\Python-T2\Project\Part 4> py .\manage.py drop-tables
hello World
Tables Dropped successfully
PS C:\Users\yagay\OneDrive\Desktop\BCIT\Term-2\Python-T2\Project\Part 4>
```



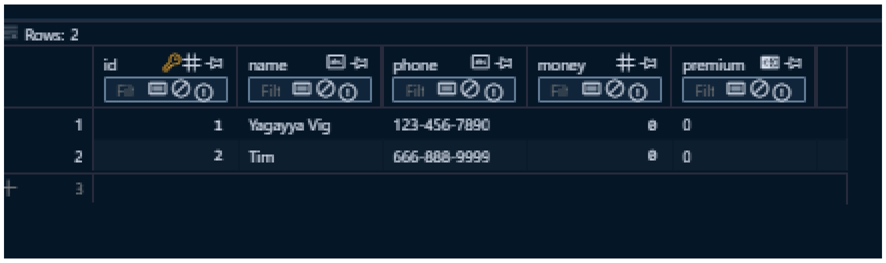
```
You, 4 minutes ago | 2 authors (yagayyavig and one other)
class Customer(db.Model):
    id = db.mapped_column(db.Integer, primary_key=True)
    name = db.mapped_column(db.String, nullable=False)
    phone = db.mapped_column(db.String)
    money = db.mapped_column(db.DECIMAL(10, 2), default=0)
    premium = db.mapped_column(db.Boolean, default=False)
    orders = db.relationship("Order", back_populates="customer")

    def __repr__(self):
        return f"Customer(id={self.id}, name='{self.name}', phone='{self.phone}')"

    def __str__(self):
        return f"{self.name} ({self.phone})"

    def completed_orders(self):
        stmt = db.select(Order).where(
            Order.customer_id == self.id,
            Order.completed != None
        ).order_by(Order.completed.desc())
        return db.session.execute(stmt).scalars()

    def pending_orders(self):
        stmt = db.select(Order).where(
            Order.customer_id == self.id,
            Order.completed == None
        ).order_by(Order.created.asc())
        return db.session.execute(stmt).scalars().all()

    def to_dict(self):
        return {
            "id": self.id,
            "name": self.name,
            "phone": self.phone,
            "money": float(self.money),
            "premium": self.premium,
            "completed_orders": [order.to_dict() for order in self.completed_orders()],
            "pending_orders": [order.to_dict() for order in self.pending_orders()]
        }
```


You, 4 minutes ago | 2 authors (yagayyavig and one other)
yagayyavig, last week • Part 4
ctrl i Add exception chaining

```python
from datetime import datetime

class Order(db.Model):
    id = db.mapped_column(db.Integer, primary_key=True, autoincrement=True)
    customer_id = db.mapped_column(db.Integer, db.ForeignKey("customer.id"), nullable=False)
    customer = db.relationship('Customer', back_populates='orders')
    created = db.mapped_column(db.DateTime, nullable=False, default=db.func.now())
    completed = db.mapped_column(db.DateTime, nullable=True, default=None)
    amount = db.mapped_column(db.DECIMAL(6, 2), nullable=True, default=None)
    delivery = db.mapped_column(db.Boolean, default=False)  # False = pickup, True = delivery
    items = db.relationship('ProductOrder', back_populates='order')

    def estimate(self):
        subtotal = sum(po.product.price * po.quantity for po in self.items)

        # Add delivery fee if applicable
        if self.delivery:
            if self.customer.premium:
                subtotal += 1  # $1 delivery fee for premium customers
            else:
                subtotal += 5  # $5 delivery fee for regular customers

        return subtotal

    def complete(self):
        if self.completed is not None:
            raise ValueError("Order has been completed")

        # Check product availability
        for po in self.items:
            if po.quantity > po.product.available:
                raise ValueError(f"Not enough available for {po.product.name}. Requested {po.quantity}, Available: {po.product.available}")

        # Calculate the total cost
        total_cost = self.estimate()

        # Check if customer has enough money
        if self.customer.money < total_cost:
            raise ValueError(f"Customer does not have enough money. Required: ${total_cost}, Available: ${float(self.customer.money)}")

        # Process the order
        self.customer.money -= total_cost  # Deduct money from customer

        # Update product inventory
        for po in self.items:
            po.product.available -= po.quantity

        # Mark order as completed
        self.completed = datetime.now()
        self.amount = total_cost

        return True

    def to_dict(self):
        result = {
            "id": self.id,
            "customer": self.customer.name if self.customer else None,
            "created": self.created.strftime("%a, %d %b %Y %H:%M:%S GMT") if self.created else None,
            "completed": self.completed,
            "delivery": self.delivery,
            "products": []
        }

        for item in self.items:
            product_info = {
                "id": item.product.id,
                "name": item.product.name,
                "price": float(item.product.price),
                "quantity": item.quantity
            }
            if not self.completed:
                product_info["available"] = item.product.available
            result["products"].append(product_info)

        if self.completed:
            result["amount"] = float(self.amount)
            result["completed_date"] = self.completed
            result["completed"] = True
        else:
            result["completed"] = False
            result["estimated_total"] = float(self.estimate())

        return result
```

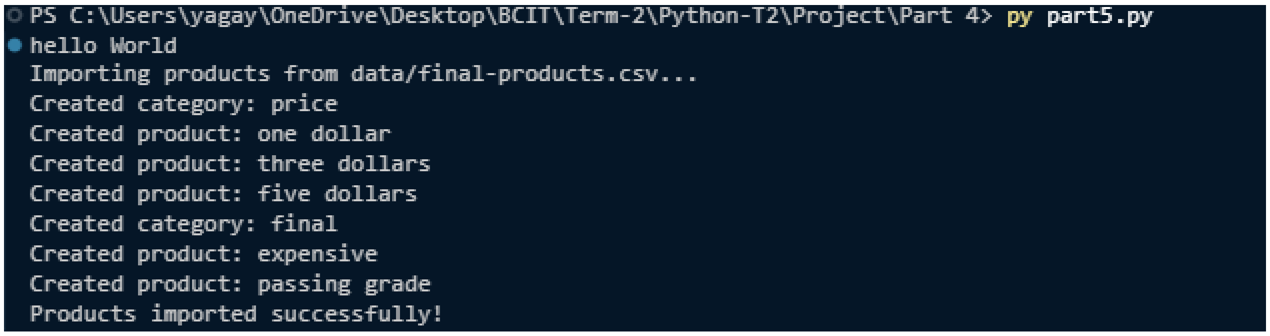


```python
from app import app
from db import db
from models import Customer

def create_customers():
    with app.app_context():
        # Check if the first customer (your name) exists
        your_name = "Yagayya Vig"
        your_phone = "123-456-7890"

        stmt = db.select(Customer).where(Customer.phone == your_phone)
        your_customer = db.session.execute(stmt).scalar_one_or_none()

        if not your_customer:
            your_customer = Customer(name=your_name, phone=your_phone)
            db.session.add(your_customer)
            print(f"Created customer: {your_name}")
        else:
            print(f"Customer {your_name} already exists")

        # Check if the second customer (Tim) exists
        tim_name = "Tim"
        tim_phone = "666-888-9999"

        stmt = db.select(Customer).where(Customer.phone == tim_phone)
        tim_customer = db.session.execute(stmt).scalar_one_or_none()

        if not tim_customer:
            tim_customer = Customer(name=tim_name, phone=tim_phone)
            db.session.add(tim_customer)
            print(f"Created customer: {tim_name}")
        else:
            print(f"Customer {tim_name} already exists")

        # Commit changes
        db.session.commit()

        print("Customers created successfully!")

if __name__ == "__main__":
    create_customers()
```



Rows: 2

	id	name	phone	money	premium
1	1	Yagayya Vig	123-456-7890	0	0
2	2	Tim	666-888-9999	0	0
3					



```
PS C:\Users\yagay\OneDrive\Desktop\BCIT\Term-2\Python-T2\Project\Part 4> py part5.py
hello World
Importing products from data/final-products.csv...
Created category: price
Created product: one dollar
Created product: three dollars
Created product: five dollars
Created category: final
Created product: expensive
Created product: passing grade
Products imported successfully!
```