```
import hashlib

class Blockchain:
    def __init__(self):
        self.chain = []
        self.transactions = []  # Initialize transactions list
        self.create_block(prev_hash='0', proof=1)

    def create_block(self, proof, prev_hash):
        block = {
            'index': len(self.chain) + 1,
            'proof': proof,
            'prev_hash': prev_hash,
            'transactions': self.transactions,
        }
        self.transactions = []  # Clear the transactions list after adding to the block
        self.chain.append(block)
        return block

    def add_transaction(self, drug_id, manufacturer, batch_number, expiration_date):
        self.transactions.append({
            'drug_id': drug_id,
            'manufacturer': manufacturer,
            'batch_number': batch_number,
            'expiration_date': expiration_date,
        })
        return self.get_prev_block()['index'] + 1

    def get_prev_block(self):
        return self.chain[-1]

    def proof_of_work(self, prev_proof):
        new_proof = 1
        while True:
            hash_value = hashlib.sha256(str(new_proof**2 - prev_proof**2).encode()).hexdigest()
            if hash_value[:4] == '0000':
                break
            new_proof += 1
        return new_proof

    def hash(self, block):
        encoded_block = str(block).encode()
        return hashlib.sha256(encoded_block).hexdigest()

    def is_chain_valid(self):
        for i in range(1, len(self.chain)):
            prev_block = self.chain[i - 1]
            current_block = self.chain[i]
            if current_block['prev_hash'] != self.hash(prev_block):
                return False
            prev_proof = prev_block['proof']
            current_proof = current_block['proof']
            hash_value = hashlib.sha256(str(current_proof**2 - prev_proof**2).encode()).hexdigest()
            if hash_value[:4] != '0000':
                return False
        return True

# Simulated usage
blockchain = Blockchain()

# Get the number of transactions from the user
num_of_entries = int(input("Enter the number of drug entries you want to add: "))

# List to hold all the transactions
transactions_table = []

# Collect user inputs for each drug and save them in the table
for _ in range(num_of_entries):
    drug_id = input("Enter the Drug ID: ")
    manufacturer = input("Enter the Manufacturer: ")
    batch_number = input("Enter the Batch Number: ")
    expiration_date = input("Enter the Expiration Date (YYYY-MM-DD): ")

    # Save the inputs in the transactions table
    transactions_table.append({
        'drug_id': drug_id,
```

```
            'manufacturer': manufacturer,
            'batch_number': batch_number,
            'expiration_date': expiration_date
        })

    # Add each transaction to the blockchain
    for transaction in transactions_table:
        blockchain.add_transaction(
            drug_id=transaction['drug_id'],
            manufacturer=transaction['manufacturer'],
            batch_number=transaction['batch_number'],
            expiration_date=transaction['expiration_date']
        )

    # Mine the block to add it to the chain
    prev_block = blockchain.get_prev_block()
    proof = blockchain.proof_of_work(prev_block['proof'])
    blockchain.create_block(proof, blockchain.hash(prev_block))

    # Verify the blockchain
    print("Is the blockchain valid?", blockchain.is_chain_valid())

    # Print the blockchain
    print("Blockchain:", blockchain.chain)
```

```
Enter the number of drug entries you want to add: 3
Enter the Drug ID: Drug001
Enter the Manufacturer: PharmaCom
Enter the Batch Number: Batch123
Enter the Expiration Date (YYYY-MM-DD): 2020-09-14
Enter the Drug ID: Drug002
Enter the Manufacturer: Apollo
Enter the Batch Number: 2022-05-19
Enter the Expiration Date (YYYY-MM-DD): 2025-12-06
Enter the Drug ID: Drug003
Enter the Manufacturer: Safemeds
Enter the Batch Number: Batch567
Enter the Expiration Date (YYYY-MM-DD): 2028-01-17
Is the blockchain valid? True
Blockchain: [{'index': 1, 'proof': 1, 'prev_hash': '0', 'transactions': []}, {'index': 2, 'proof': 533, 'prev_hash': '12424df85dbbe60e76
```