**235229119**

**Nivashini J**

**Rollno.: 235229119**

**MODERN DATABASE SYSTEM**

**I-m.Sc – Data science**

**Bishop Heber college**

**Online shopping Management**

**ER DIAGRAM**

orderitems

customer

Added as

Placed

orders

contains

product

**My SQL: SQL stands for Structured Query Language**andis a computer language that we use to interact with a relational database. SQL is a tool for organizing, managing, and retrieving archived data from a computer database

**TABLES AND ITS ENTITIES**:

**Product Table:**

ProductID, Name Description Price StockQuantity Category

**Customer Table:**

Customer , FirstName LastName Email Phone Address

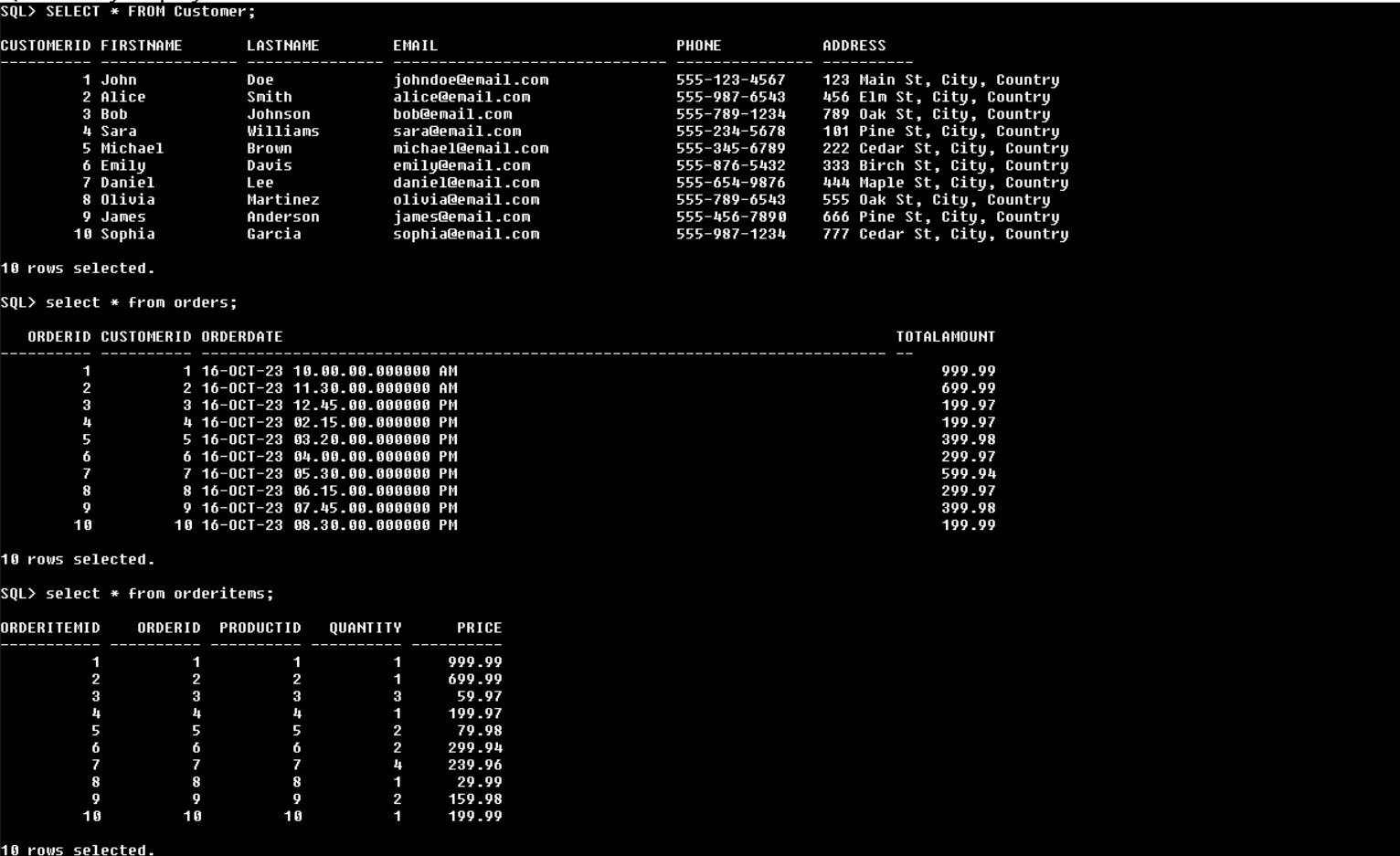
**Orders Table:**

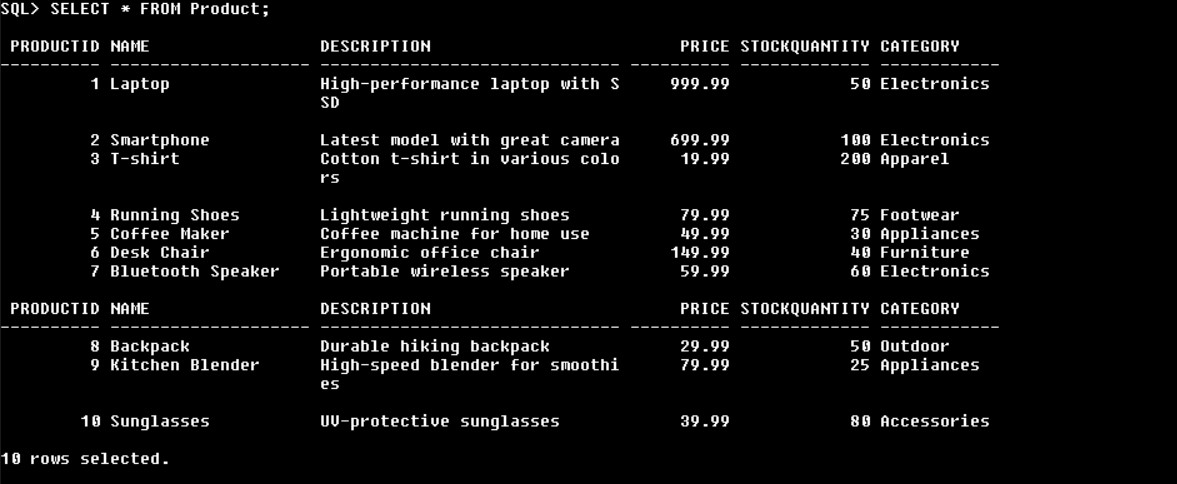
OrderID , CustomerID OrderDate TotalAmount

**OrderItem Table:**

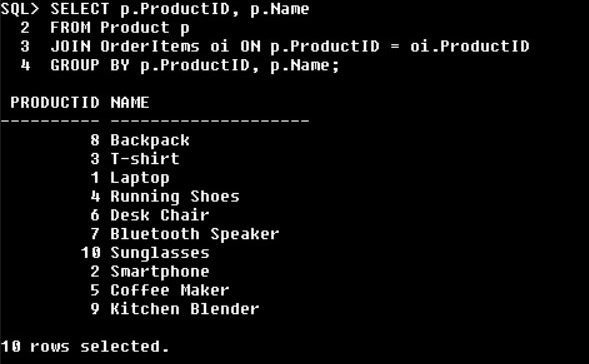
OrderItemID OrderID ProductID Quantity Price

**TABLES:**

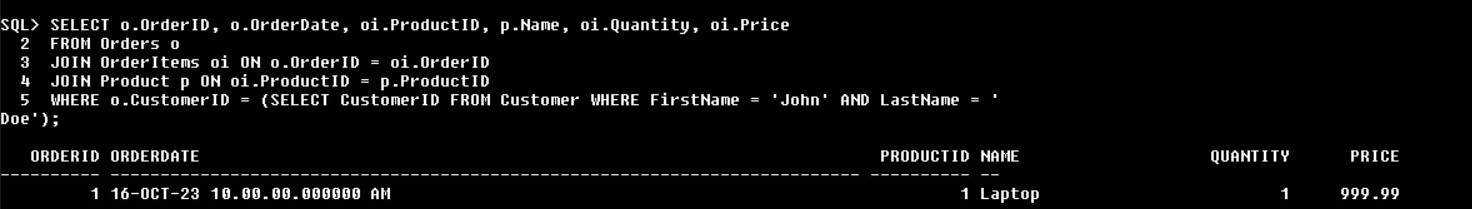




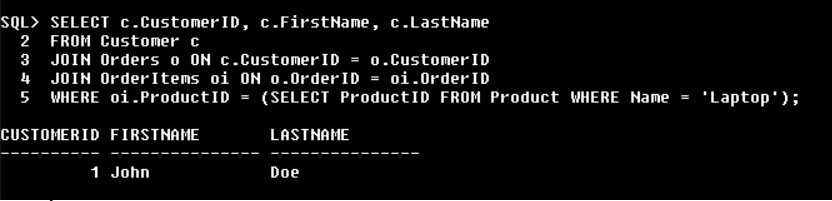
1] List products that have been ordered at least once:



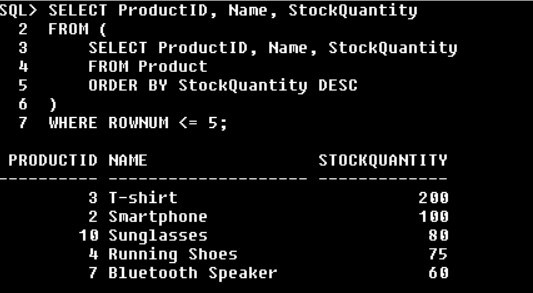
2] Retrieve all orders made by a specific customer:



3] List customers who have purchased a specific product:



4] Retrieve the 5 most recent orders:



5] Find the customer who spent the most:



MONGODB : MongoDB is an open-source document-oriented database that is designed to store a large scale of data and also allows you to work with that data very efficiently. It is categorized under the NoSQL (Not only SQL) database because the storage and retrieval of data in the MongoDB are not in the form of tables.

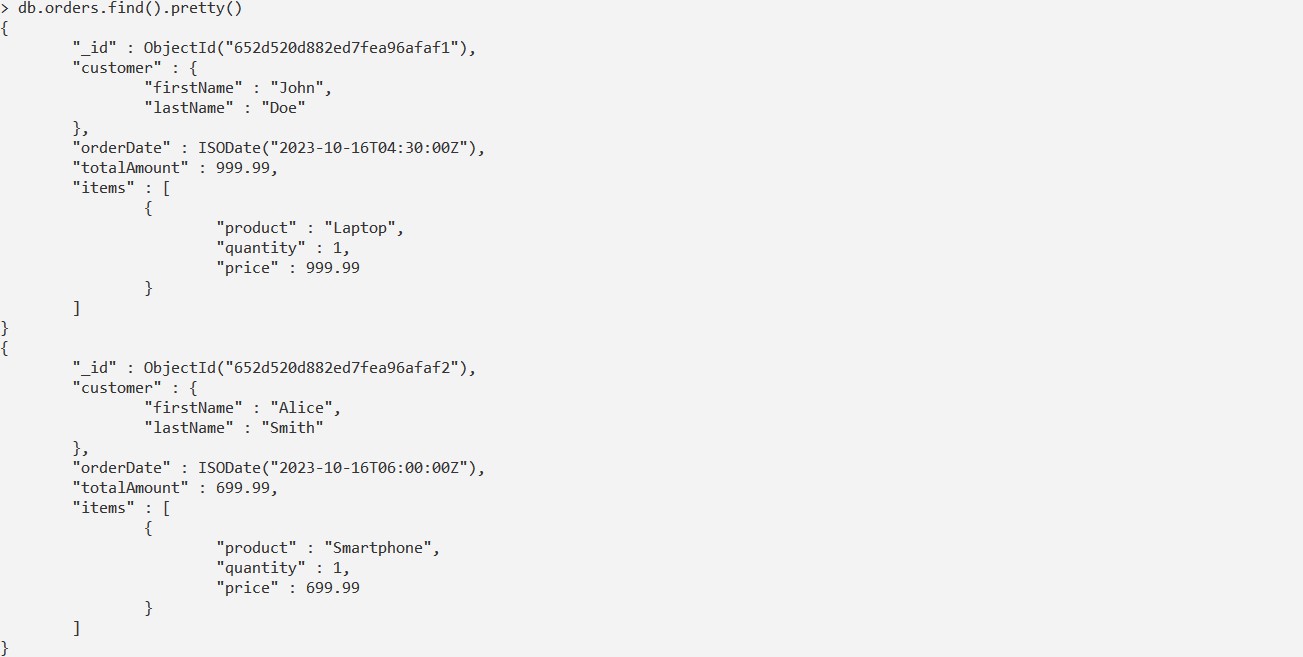
**DATABASE NAME:**

* online\_shopping

**COLLECTION NAMES:**

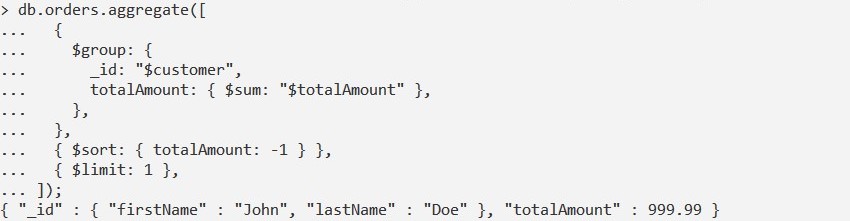
* customers
* orders
* products

**COLLECTIONS:** 

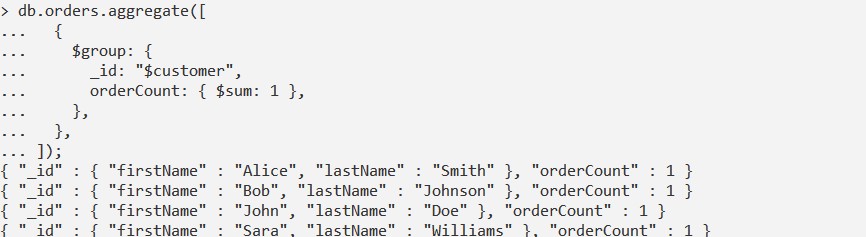




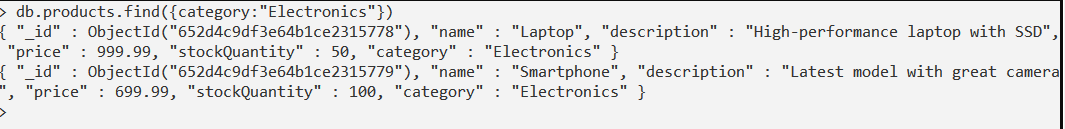
**1] Find the customer with the highest total amount spent:**



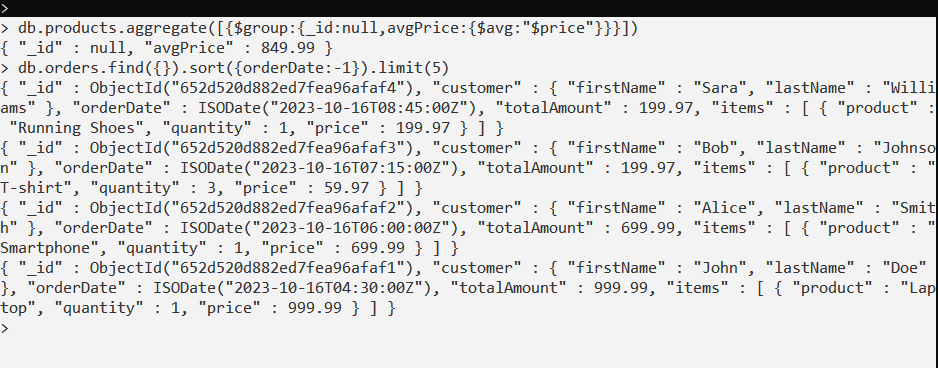
**2] Retrieve the number of orders placed by each customer**:



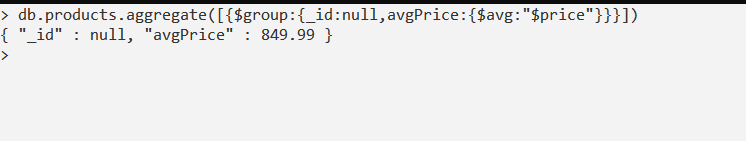
**3] List all products in a specific category (e.g., Electronics):**



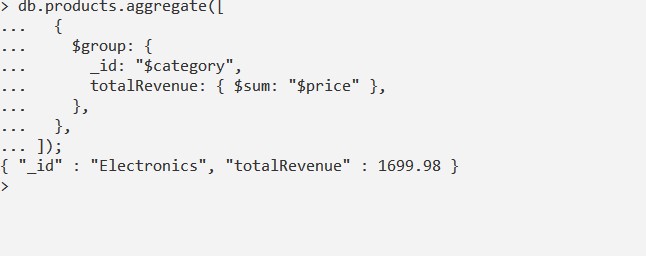
4] Find the 5 most recent orders:



**5] Calculate the average price of products:**



**6] Calculate the total revenue for each category of products:**



**Neo4j:** Neo4j is the most famous database management system and it is also a NoSQL database system. Neo4j is different from Mysql or MongoDB it has its own features that’s makes it special compared to other Database Management System.

**Lables:**

* Products
* Orders
* Orderitems
* Customer

**Queries for Creating Nodes for lables and Relationships:**

1]

CREATE (laptop:Product {name: "Laptop", description: "High-performance laptop with SSD", price: 999.99, stockQuantity: 50, category: "Electronics"}),

       (smartphone:Product {name: "Smartphone", description: "Latest model with great camera", price: 699.99, stockQuantity: 100, category: "Electronics"})

2]

CREATE (john:Customer {firstName: "John", lastName: "Doe", email: "johndoe@email.com", phone: "555-123-4567", address: "123 Main St, City, Country"}),

       (alice:Customer {firstName: "Alice", lastName: "Smith", email: "alice@email.com", phone: "555-987-6543", address: "456 Elm St, City, Country"})

3]

CREATE (order1:Order {orderDate: datetime('2023-10-16T10:00:00'), totalAmount: 999.99}),

       (order2:Order {orderDate: datetime('2023-10-16T11:30:00'), totalAmount: 699.99})

* creating Relationship:

CREATE (john)-[:PLACED]->(order1),

       (alice)-[:PLACED]->(order2)

4]

CREATE (orderItem1:OrderItem {quantity: 1, price: 999.99}),(orderItem2:OrderItem {quantity: 1, price: 699.99})

* creating relationship:

CREATE (order1)-[:CONTAINS]->(orderItem1),

       (order2)-[:CONTAINS]->(orderItem2)

**Node lables:**

1]

MATCH (n:Order) RETURN n LIMIT 25

|  |
| --- |
|  |

.

{

"identity": 179,

"labels": [

"Order"

],

"properties": {

"totalAmount": 999.99,

"orderDate": "2023-10-16T10:00:00Z"

},

"elementId": "179"

}

{

"identity": 180,

"labels": [

"Order"

],

"properties": {

"totalAmount": 699.99,

"orderDate": "2023-10-16T11:30:00Z"

},

"elementId": "180"

}

2]

MATCH (n:OrderItem) RETURN n LIMIT 25

{

"identity": 181,

"labels": [

"OrderItem"

],

"properties": {

"quantity": 1,

"price": 999.99

},

"elementId": "181"

}

{

"identity": 182,

"labels": [

"OrderItem"

],

"properties": {

"quantity": 1,

"price": 699.99

},

"elementId": "182"

}

3]

MATCH (n:Product) RETURN n LIMIT 25

{

"identity": 169,

"labels": [

"Product"

],

"properties": {

"price": 999.99,

"name": "Laptop",

"stockQuantity": 50,

"description": "High-performance laptop with SSD",

"category": "Electronics"

},

"elementId": "169"

}

{

"identity": 171,

"labels": [

"Product"

],

"properties": {

"price": 999.99,

"name": "Laptop",

"stockQuantity": 50,

"description": "High-performance laptop with SSD",

"category": "Electronics"

},

"elementId": "171"

}{

"identity": 171,

"labels": [

"Product"

],

"properties": {

"price": 999.99,

"name": "Laptop",

"stockQuantity": 50,

"description": "High-performance laptop with SSD",

"category": "Electronics"

},

"elementId": "171"

}{

"identity": 172,

"labels": [

"Product"

],

"properties": {

"price": 699.99,

"name": "Smartphone",

"stockQuantity": 100,

"description": "Latest model with great camera",

"category": "Electronics"

},

"elementId": "172"

}

4]

MATCH (n:Customer) RETURN n LIMIT 25



**Cyber Queries for Retrieve Nodes and Relationships:**

**1] Find the customer with the highest total amount spent:**

MATCH (c:Customer)-[:PLACED]->(o:Order)

WITH c, SUM(o.totalAmount) AS totalSpent

ORDER BY totalSpent DESC

LIMIT 1

RETURN c, totalSpent;



**2] Retrieve all orders made by a specific customer (e.g., John Doe):**

MATCH (c:Customer {firstName: "John", lastName: "Doe"})-[:PLACED]->(o:Order)

RETURN o;



**3] Find orders placed within a specific date range (e.g., between two dates):**

MATCH (o:Order)

WHERE o.orderDate >= datetime('2023-10-15T00:00:00') AND o.orderDate < datetime('2023-10-17T00:00:00')

RETURN o;

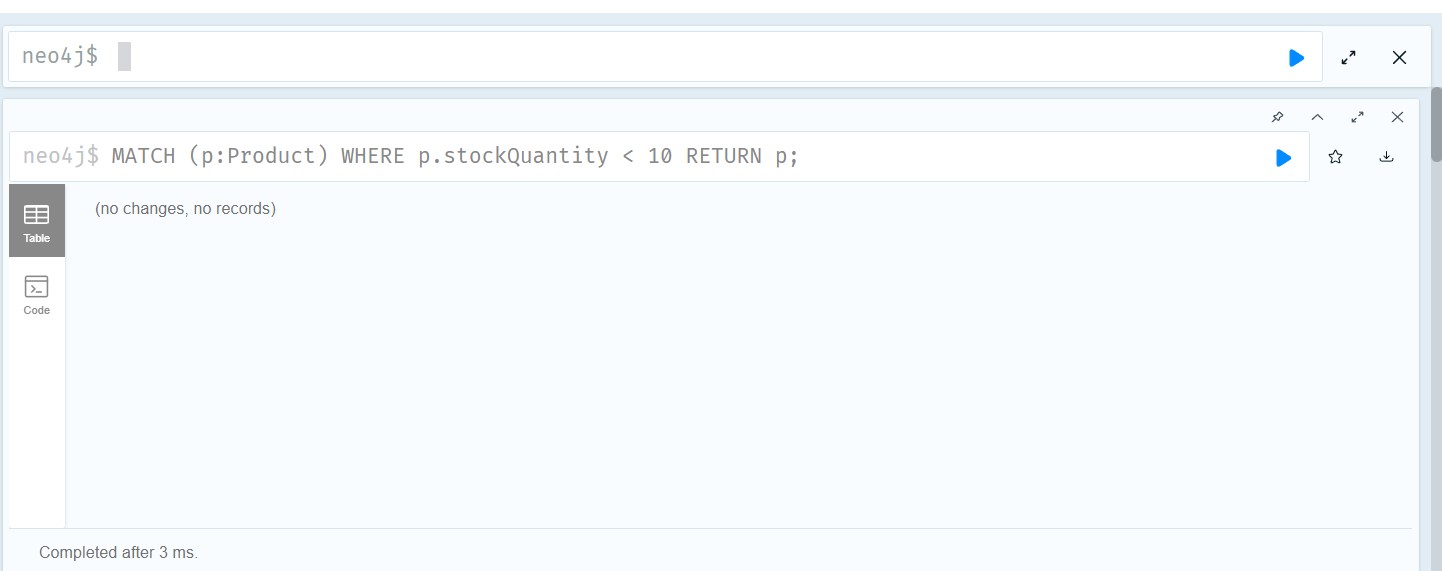


**4] List products with low stock quantities (e.g., below 10):**

MATCH (p:Product)

WHERE p.stockQuantity < 10

RETURN p;



**5]List orders with a total amount over a specific value (e.g., $500):**

MATCH (o:Order)

WHERE o.totalAmount > 500

RETURN o;

