



### Instructions:

Apply the concept of database design by analyzing the given problem. Create a database design based on the provided business rules. Include a copy of the **Entity Relationship Diagram (ERD)**.

XYZ Electronics Store operates a retail outlet that sells various electronic devices, including new and refurbished items, and provides repair services for its customers. Based on the following business rules, design a database:

## Database Design

- A salesperson may sell many devices, but each device is sold by only one salesperson.

//For Sales Person

```
Autocommit Display 10 Save Run
CREATE TABLE Salesperson ( SalespersonID NUMBER(10,0) PRIMARY KEY, SalespersonName
VARCHAR2(100), Contact No NUMBER(10,0));

Results Explain Describe Saved SQL History
Table created.
0.05 seconds
```

//For the Electronic Devices to be Sold

```
Autocommit Display 10 Save Run
CREATE TABLE Devices ( Device ID NUMBER(10,0) PRIMARY KEY, DeviceName VARCHAR2(50),
DeviceType VARCHAR2(50), DeviceCondition VARCHAR2(50), PurchasePrice DECIMAL(10,2),
SalePrice DECIMAL(10,2), Warranty DATE );

Results Explain Describe Saved SQL History
Table created.
```



//For the Electronic Devices Sales Person Already Sold (I add the customer first before this)

```
Autocommit Display 10 Save Run
CREATE TABLE Sold Device ( Sold DeviceID NUMBER(10,0) PRIMARY KEY, Device ID
NUMBER(10,0), Purchased Date DATE, CustomerID NUMBER(10,0), SalespersonID NUMBER(10,0),
FOREIGN KEY (Device ID) REFERENCES Devices(Device ID), FOREIGN KEY (CustomerID)
REFERENCES Customer(CustomerID), FOREIGN KEY (SalespersonID) REFERENCES
Salesperson(SalespersonID) );
```

Results Explain Describe Saved SQL History

Table created.

- A customer may purchase multiple devices, but each device is purchased by only one customer.

//For Customer's

```
Autocommit Display 10 Save Run
CREATE TABLE Customer ( CustomerID NUMBER(10,0) PRIMARY KEY, CustomerName VARCHAR2(50),
Contact No NUMBER(10,0));
```

Results Explain Describe Saved SQL History

Table created.





- A customer may also visit the store solely to request repair services for their electronic devices.

//For the Request Repair details

```
Autocommit Display 10 Save Run
CREATE TABLE Repair_Requests (
  Request_ID NUMBER(10,0) PRIMARY KEY, Sold_DeviceID NUMBER(10,0),
  Request_Date DATE,
  Description VARCHAR2(50),
  FOREIGN KEY (Sold_DeviceID) REFERENCES Sold_Device(Sold_DeviceID)
);
Results Explain Describe Saved SQL History
Table created.
```

- For every device brought in for repair, the store issues a unique repair order.

//For the status

```
Autocommit Display 10 Save Run
CREATE TABLE RepairStatus ( StatusID NUMBER(10,0) PRIMARY KEY, StatusName VARCHAR2(50) );
Results Explain Describe Saved SQL History
Table created.
```







//The Repair Order

```
CREATE TABLE RepairOrder ( RepairOrderID NUMBER(10,0) PRIMARY KEY, OrderDate DATE,
Request_ID NUMBER(10,0), StatusID NUMBER(10,0), FOREIGN KEY (Request_ID) REFERENCES
Repair_Requests(Request_ID), FOREIGN KEY (StatusID) REFERENCES RepairStatus(StatusID) );
```

Table created.

- The store keeps track of all available spare parts, which can be updated as new parts are added.

//To track Spare Parts

```
CREATE TABLE SparePart (SparePartID NUMBER(10,0) PRIMARY KEY, SparePartName
VARCHAR2(50), SparePartCost DECIMAL(10,2), Stocks NUMBER(10,0)
);
```

- Each repair order may involve one or more spare parts, and the database must record which parts were used for each repair.

// More Detailed Repair Order

```
CREATE TABLE RepairOrderDetail (
RepairOrderID NUMBER(10,0), SparePartID NUMBER(10,0),
QuantityUsed NUMBER(10,0), PRIMARY KEY (RepairOrderID, SparePartID),
FOREIGN KEY (RepairOrderID) REFERENCES RepairOrder(RepairOrderID),
FOREIGN KEY (SparePartID) REFERENCES SparePart(SparePartID)
);
```

Table created.



Entity Relationship Diagram (ERD)

