SQL Database design

Wednesday, 16 August 2023 7:40 pm

Objective: Design a functional Database

Outline:
-Design a database for storing product details, sales details, vendor details, and inventory details
-Populate and update database
-Backup database
-Stored procedures
-Create views

```
CREATE TABLE VENDOR_TABLE
(
VendorID int PRIMARY KEY IDENTITY (1,1),
Vendor_name nvarchar(50),
Contact_email nvarchar(100) UNIQUE,
Contact_phone int
);

CREATE TABLE PRODUCT_TABLE
(
ProductID int PRIMARY KEY IDENTITY (1,1),
Product_name nvarchar(50),
Category nvarchar(30),
Cost_Price float,
Selling_Price float,
VendorID nvarchar(50),
Quantity_in_stock int
FOREIGN KEY (VendorID) REFERENCES VENDOR_TABLE(VendorID)
);

CREATE TABLE SALES_TABLE
(
SaleID int PRIMARY KEY IDENTITY (1,1),
ProductID int,
Sale_Date Date,
Quantity_sold int,
Revenue float
FOREIGN KEY (ProductID) REFERENCES PRODUCT_TABLE(ProductID)
);

CREATE TABLE INVENTORY_TABLE
(
InventoryID INT PRIMARY KEY IDENTITY (1,1),
ProductID int,
Product_name nvarchar(50),
Stock_quantity int,
Reorder_threshold int
FOREIGN KEY (ProductID) REFERENCES PRODUCT_TABLE(ProductID)
);
```

Using the block of SQL codes above , I was able to create the following tables:

Product Table:
- ProductID (Primary Key)
- Name
- Category
- Cost Price
- Selling PriceI
- VendorID (Foreign Key referencing Vendor Table)

- QuantityInStock

Sales Table:
- SaleID (Primary Key)
- ProductID (Foreign Key referencing Product Table)
- SaleDate
- QuantitySold
- Revenue

Vendor Table:
- VendorID (Primary Key)
- VendorName
- ContactEmail
- ContactPhone

Inventory Table:
- InventoryID (Primary Key)
- ProductID (Foreign Key referencing Product Table)
- Product name
- StockQuantity
- ReorderThreshold

Using python, I was able to populate the database with dummy data generated by Utilizing the PYODBC, RANDOM libraries in library

After creating the database, I ran some queries to make sure it ran running properly.
I also corrected some errors that occurred while populating the database.

Next, I went ahead to create a backup of the database. There are 2 ways I know to do this but I resorted to using the following block of code:

```
BACKUP DATABASE [LOFTY VENTURES ]
TO DISK = 'C:\Program Files\Microsoft SQL Server\MSSQL16.SQLEXPRESS\MSSQL\Backup\LOFTY_VENTURES.bak'
WITH FORMAT, NAME = 'Full Backup';
```

My next objective was to create a stored procedure for updating product price and updating inventory data
This is used for updating product selling price:

```
CREATE PROCEDURE UpdatesellingtPrice
   @ProductID INT,
   @NewPrice FLOAT
AS
BEGIN
   UPDATE  PRODUCT_TABLE
   SET Selling_Price = @NewPrice
   WHERE ProductID = @ProductID;

   SELECT 'Selling_Price updated to' +' '+ CAST(@NewPrice AS NVARCHAR(20)) AS Result;
END;
```

This is used for updating stock quantity in inventory table:

```
CREATE PROCEDURE update_stocks_quantity
        @ProductID INT
       ,@NewQty INT
AS
BEGIN
       UPDATE INVENTORY_TABLE
       SET Stock_quantity = @NewQty
       WHERE ProductID = @ProductID;

       SELECT 'New stock Qty is now' +' '+ CAST(@NewQty AS NVARCHAR(20))+ '!' AS Result ;
```

My next object is to create views in SQL
A view is a virtual table or a saved query result that acts as a table but is not physically stored as a separate table in the database. Instead, it's a dynamically generated representation of data from one or more tables.

For instance, I have created a sales report view:

```
CREATE VIEW SalesReport AS
SELECT
   Sales.SaleID
   ,PRDCT.ProductID
   ,PRDCT.Product_name
   ,PRDCT.Category
   ,SALES.Sale_Date
   ,SALES.Quantity_sold
   ,SALES.Revenue
   ,VNDR.Vendor_name
FROM SALES_TABLE AS Sales
INNER JOIN PRODUCT_TABLE AS PRDCT ON SALES.ProductID = PRDCT.ProductID
INNER JOIN VENDOR_TABLE AS VNDR ON PRDCT.VendorID = Vndr.VendorID;

SELECT *
FROM SalesReport
```