SQL PROJECT (MAVEN TOY SALES)

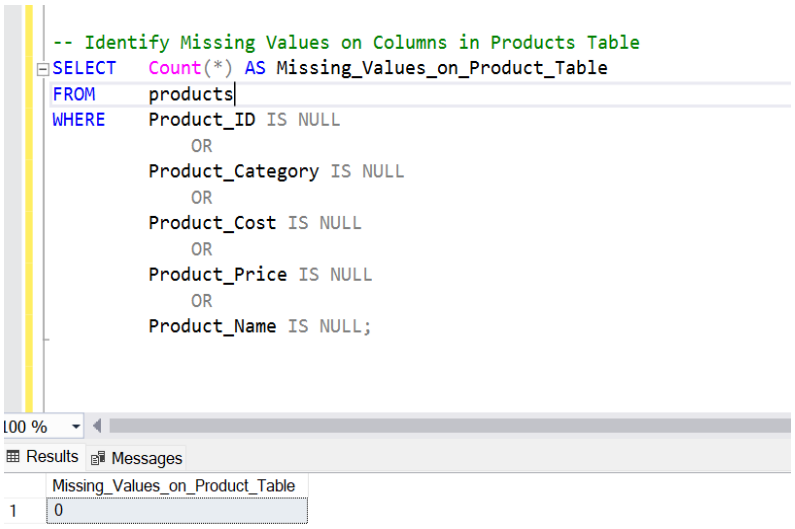
DATA CLEANING AND PROCESSING

Data cleaning is the crucial process of identifying and correcting errors, inconsistencies, and inaccuracies in datasets to enhance their quality for reliable analysis.

This entails dealing with problems such as duplicates, missing values, standardizing data types, etc. The cleaned dataset formed the solid foundation for subsequent analyses, it helps to ensure that data is accurate and well prepared for analysis.

Data Cleaning on product Table

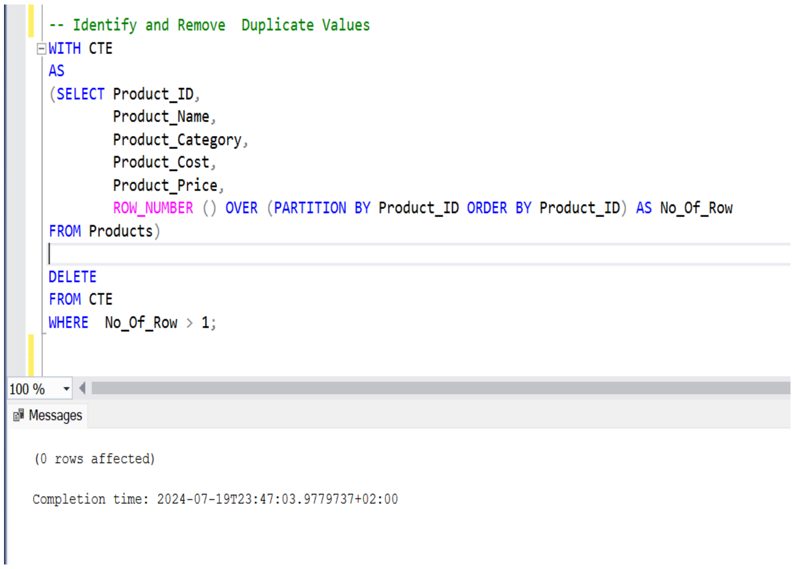
The key column in the product table was cleaned using the following data cleaning procedures.



```sql
-- Identify Missing Values on Columns in Products Table
SELECT   Count(*) AS Missing_Values_on_Product_Table
FROM     products
WHERE    Product_ID IS NULL
             OR
         Product_Category IS NULL
             OR
         Product_Cost IS NULL
             OR
         Product_Price IS NULL
             OR
         Product_Name IS NULL;
```

```
100 %
Results   Messages
   Missing_Values_on_Product_Table
1  0
```

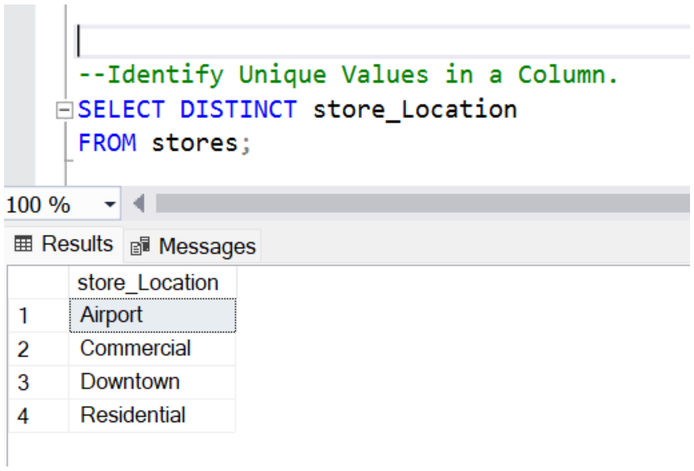
The generated result indicates that there are no missing values in the products table.



```sql
-- Identify and Remove  Duplicate Values
WITH CTE
AS
(SELECT Product_ID,
        Product_Name,
        Product_Category,
        Product_Cost,
        Product_Price,
        ROW_NUMBER () OVER (PARTITION BY Product_ID ORDER BY Product_ID) AS No_Of_Row
FROM Products)

DELETE
FROM CTE
WHERE  No_Of_Row > 1;
```

```
100 %
Messages

(0 rows affected)

Completion time: 2024-07-19T23:47:03.9779737+02:00
```

The generated result indicates that there are no duplicate values in product table



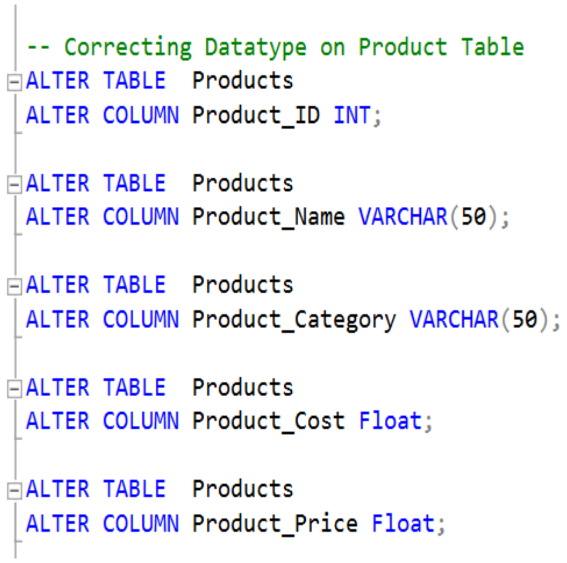
```sql
--Identify Unique Values in a Column.
SELECT DISTINCT store_Location
FROM stores;
```

100 %

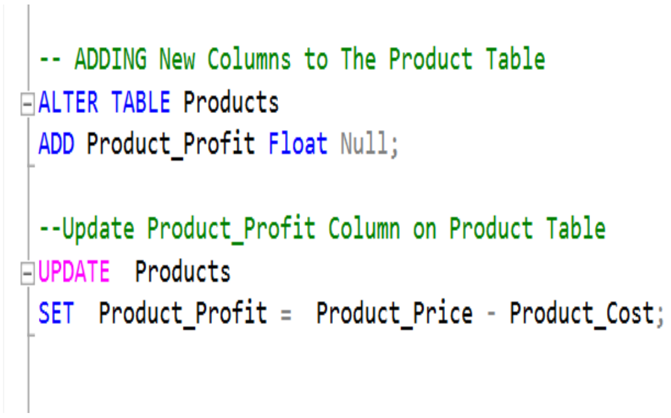
Results | Messages

```
  store_Location
1 Airport
2 Commercial
3 Downtown
4 Residential
```

The generated result affirms that all values in the Product Category column are categorized appropriately.



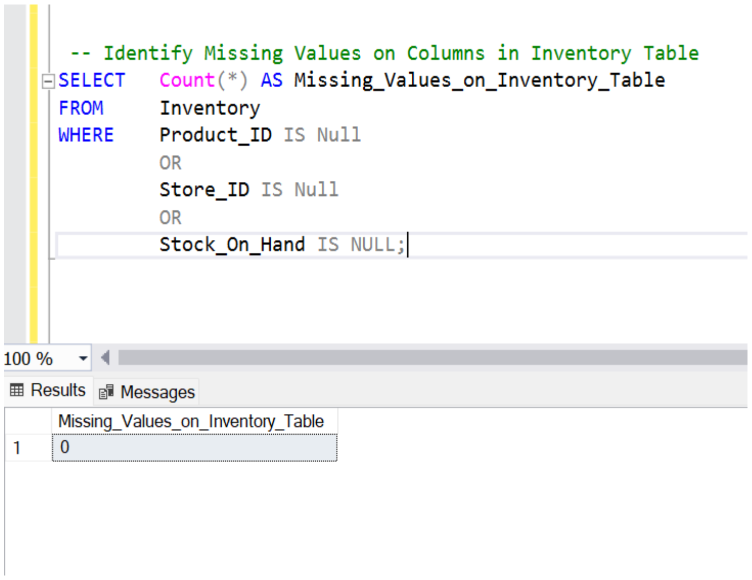
```sql
-- Correcting Datatype on Product Table
ALTER TABLE  Products
ALTER COLUMN Product_ID INT;

ALTER TABLE  Products
ALTER COLUMN Product_Name VARCHAR(50);

ALTER TABLE  Products
ALTER COLUMN Product_Category VARCHAR(50);

ALTER TABLE  Products
ALTER COLUMN Product_Cost Float;

ALTER TABLE  Products
ALTER COLUMN Product_Price Float;
```



```sql
-- ADDING New Columns to The Product Table
ALTER TABLE Products
ADD Product_Profit Float Null;

--Update Product_Profit Column on Product Table
UPDATE  Products
SET  Product_Profit =  Product_Price - Product_Cost;
```

Data Cleaning on Inventory Table

The key column in the inventory table was cleaned using the following data cleaning procedures.



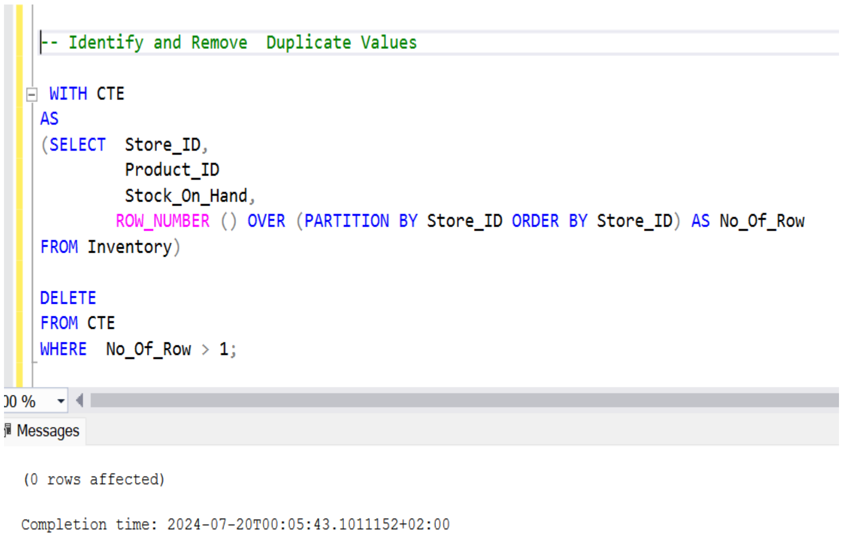
```
 -- Identify Missing Values on Columns in Inventory Table
SELECT   Count(*) AS Missing_Values_on_Inventory_Table
FROM     Inventory
WHERE    Product_ID IS Null
         OR
         Store_ID IS Null
         OR
         Stock_On_Hand IS NULL;
```

100 %

Results | Messages

	Missing_Values_on_Inventory_Table
1	0

According to the generated result no values are missing from the inventory table's.



```
-- Identify and Remove  Duplicate Values

 WITH CTE
AS
(SELECT  Store_ID,
         Product_ID
         Stock_On_Hand,
        ROW_NUMBER () OVER (PARTITION BY Store_ID ORDER BY Store_ID) AS No_Of_Row
FROM Inventory)

DELETE
FROM CTE
WHERE  No_Of_Row > 1;
```

00 %

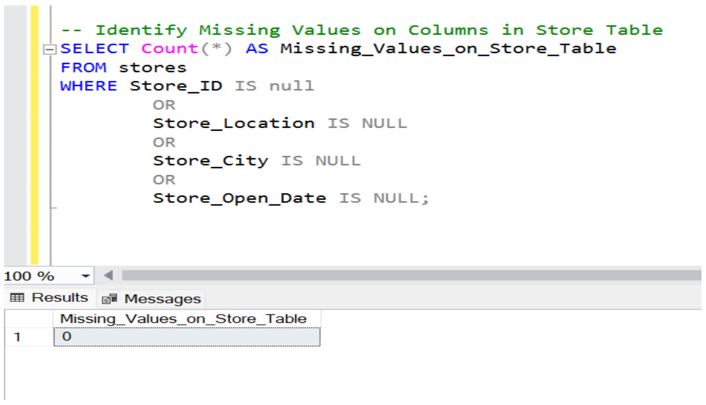
Messages

```
(0 rows affected)

Completion time: 2024-07-20T00:05:43.1011152+02:00
```

The generated result indicates that there are no duplicate values in inventory table.

Data Cleaning on Store Table

The key column in the Store table was cleaned using the following data cleaning procedures.



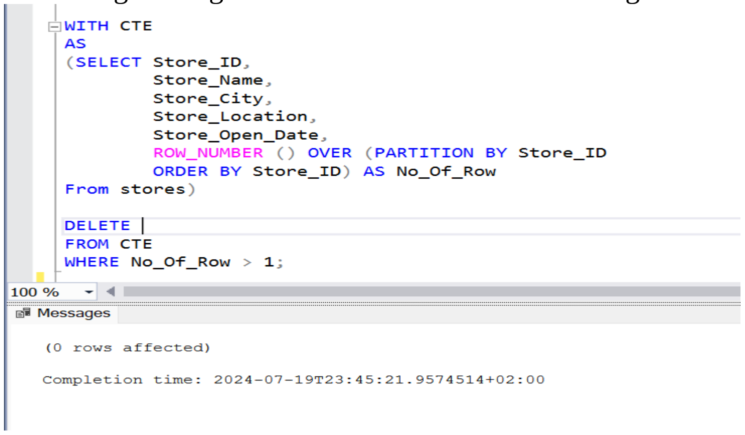
```
-- Identify Missing Values on Columns in Store Table
SELECT Count(*) AS Missing_Values_on_Store_Table
FROM stores
WHERE Store_ID IS null
        OR
        Store_Location IS NULL
        OR
        Store_City IS NULL
        OR
        Store_Open_Date IS NULL;
```

100 %

Results | Messages

	Missing_Values_on_Store_Table
1	0

According to the generated result there are no missing values in the Sales dataset.



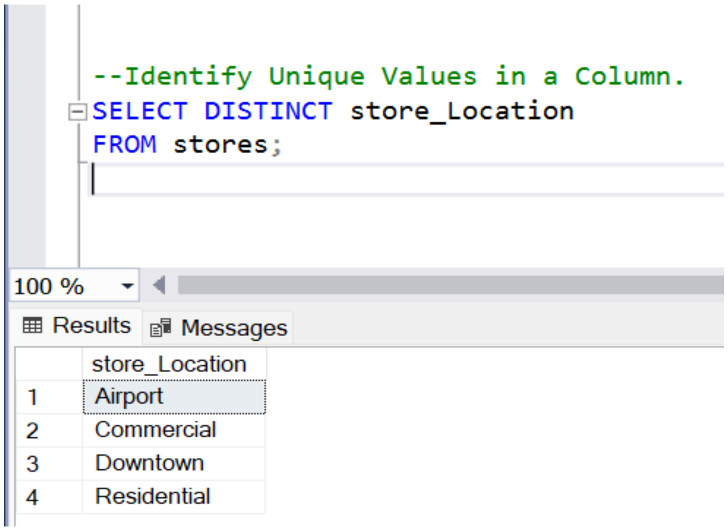
```
WITH CTE
AS
(SELECT Store_ID,
        Store_Name,
        Store_City,
        Store_Location,
        Store_Open_Date,
        ROW_NUMBER () OVER (PARTITION BY Store_ID
        ORDER BY Store_ID) AS No_Of_Row
From stores)

DELETE
FROM CTE
WHERE No_Of_Row > 1;
```

100 %

Messages

```
(0 rows affected)

Completion time: 2024-07-19T23:45:21.9574514+02:00
```

According to the generated result it indicates that there are no duplicate values in store table



```
--Identify Unique Values in a Column.
SELECT DISTINCT store_Location
FROM stores;
```

100 %

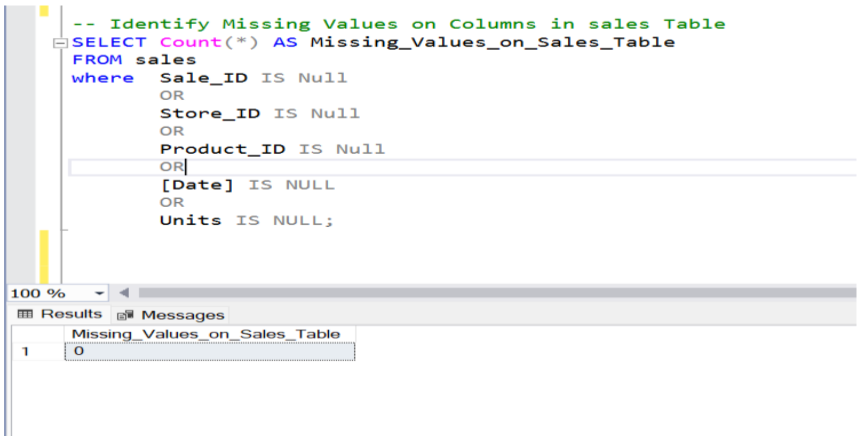
Results | Messages

	store_Location
1	Airport
2	Commercial
3	Downtown
4	Residential

The generated result indicates that all values in the Store Location column are categorized correctly.

Data Cleaning on Sales Table

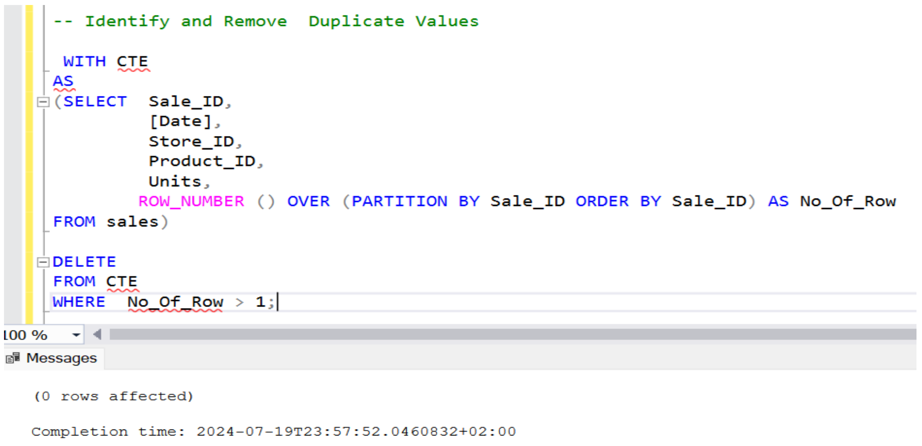
The key column in the Sales table was cleaned using the following data cleaning procedures.



```
-- Identify Missing Values on Columns in sales Table
SELECT Count(*) AS Missing_Values_on_Sales_Table
FROM sales
where   Sale_ID IS Null
        OR
        Store_ID IS Null
        OR
        Product_ID IS Null
        OR
        [Date] IS NULL
        OR
        Units IS NULL;
```

```
100 %
Results   Messages
   Missing_Values_on_Sales_Table
1  0
```

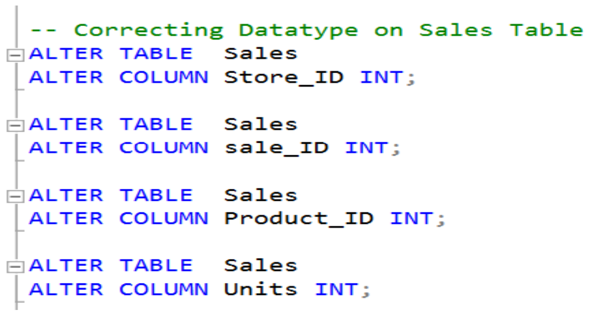
The generated result indicates that there are no missing values on the key columns in sales table



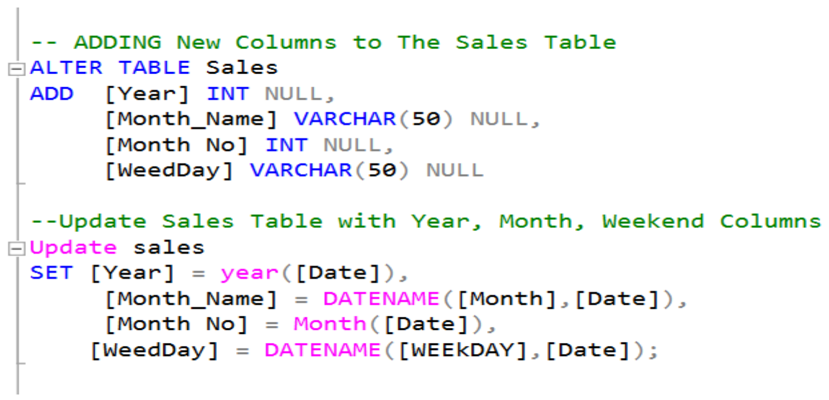
```
-- Identify and Remove  Duplicate Values

 WITH CTE
AS
(SELECT  Sale_ID,
         [Date],
         Store_ID,
         Product_ID,
         Units,
        ROW_NUMBER () OVER (PARTITION BY Sale_ID ORDER BY Sale_ID) AS No_Of_Row
FROM sales)

DELETE
FROM CTE
WHERE  No_Of_Row > 1;
```

```
100 %
Messages

(0 rows affected)

Completion time: 2024-07-19T23:57:52.0460832+02:00
```

The generated result indicates that there are no duplicate values in sales table



```
-- Correcting Datatype on Sales Table
ALTER TABLE  Sales
ALTER COLUMN Store_ID INT;

ALTER TABLE  Sales
ALTER COLUMN sale_ID INT;

ALTER TABLE  Sales
ALTER COLUMN Product_ID INT;

ALTER TABLE  Sales
ALTER COLUMN Units INT;
```



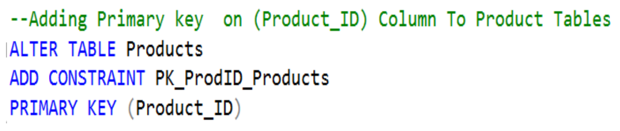
```
-- ADDING New Columns to The Sales Table
ALTER TABLE Sales
ADD   [Year] INT NULL,
      [Month_Name] VARCHAR(50) NULL,
      [Month No] INT NULL,
      [WeedDay] VARCHAR(50) NULL

--Update Sales Table with Year, Month, Weekend Columns
Update sales
SET [Year] = year([Date]),
      [Month_Name] = DATENAME([Month],[Date]),
      [Month No] = Month([Date]),
     [WeedDay] = DATENAME([WEEkDAY],[Date]);
```

Established Relationships Between Tables.

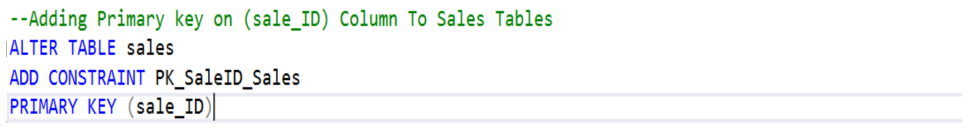
One of the most important aspects of database design in SQL is establishing relationships between tables, which is usually accomplished by using primary keys and foreign keys. A primary key functions as a unique identifier for each record in a table and is a fundamental concept in database design. Every record in the table has a unique value stored in a primary key. This uniqueness ensures that each record can be uniquely identified and distinguished from others in the same table.

A column, or group of columns, in a relational database table that creates a connection between data in two tables is called a foreign key. In addition to establishing relationships between tables, it serves to maintain referential integrity. One table's foreign key can be used to refer to another table's primary key. To guarantee that values in the foreign key column(s) match values already present in the primary key column of the referenced table, this relationship establishes a link between the data in the two tables.

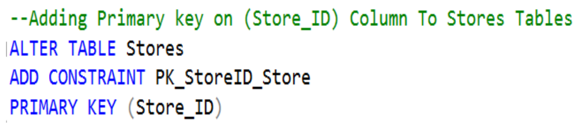
The following procedure was used to establish relationships between the tables.



```
--Adding Primary key  on (Product_ID) Column To Product Tables
ALTER TABLE Products
ADD CONSTRAINT PK_ProdID_Products
PRIMARY KEY (Product_ID)
```



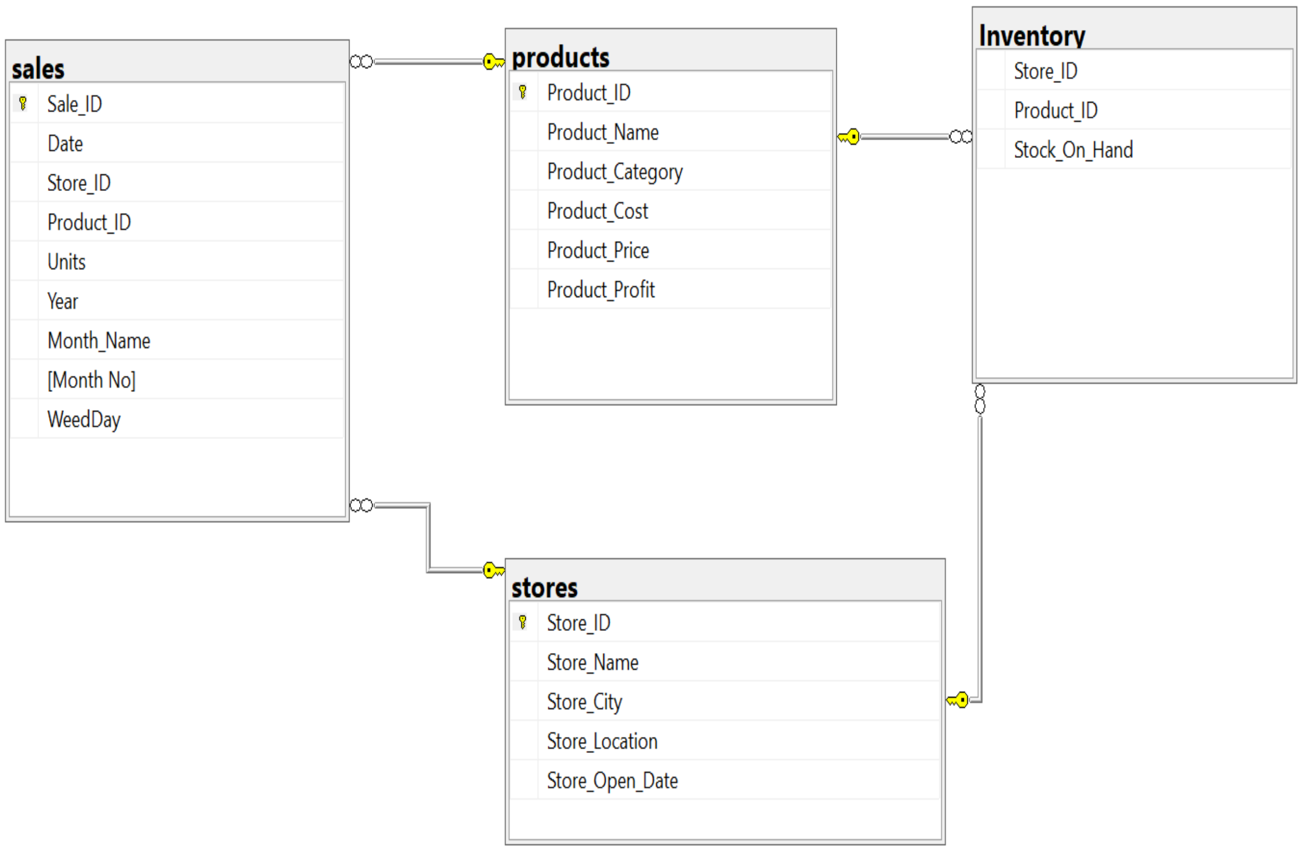
```
--Adding Primary key on (sale_ID) Column To Sales Tables
ALTER TABLE sales
ADD CONSTRAINT PK_SaleID_Sales
PRIMARY KEY (sale_ID)
```



```
--Adding Primary key on (Store_ID) Column To Stores Tables
ALTER TABLE Stores
ADD CONSTRAINT PK_StoreID_Store
PRIMARY KEY (Store_ID)
```

```
-- Creating a Foreign key on (Store_ID) Column on sales Table References (Store_ID) Column on stores Table to
--  Establish Relationship Between sales Table and stores Table

ALTER TABLE sales
ADD CONSTRAINT FK_StoreID_Store
FOREIGN KEY (Store_ID) REFERENCES stores (Store_ID)

-- Creating a Foreign key on (Store_ID) Column on Inventory Table References (Store_ID) Column on Store Table to
-- Establish Relationship Between Inventory Table and Store Table

ALTER TABLE Inventory
ADD CONSTRAINT FK_StoreID_Inventory
FOREIGN KEY (Store_ID) REFERENCES stores (Store_ID)
```

Establishing connections between the tables is essential to the analysis. In addition to maintaining data organization, it also speeds up query execution, prevents redundant data entry, and creates a flexible database structure. The outcome of the relationship that was established between the tables is shown below.



sales
Sale_ID
Date
Store_ID
Product_ID
Units
Year
Month_Name
[Month No]
WeedDay
products
Product_ID
Product_Name
Product_Category
Product_Cost
Product_Price
Product_Profit
Inventory
Store_ID
Product_ID
Stock_On_Hand
stores
Store_ID
Store_Name
Store_City
Store_Location
Store_Open_Date