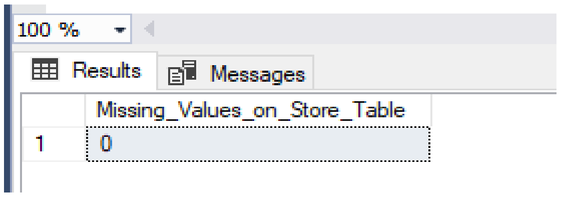
SQL PROJECT (MAVEN TOY SALES)

DATA CLEANING AND PROCESSING

Data cleaning is the crucial process of detecting and rectifying errors, discrepancies, and inaccuracies in datasets to enhance their quality for reliable analysis. This involves addressing issues like missing values, duplicates, standardizing data types, etc. The cleaned dataset formed the solid foundation for subsequent analyses, helps to ensure that data is accurate and well-prepared for analysis.

Data Cleaning on Store Table: The following data cleaning procedures were applied to the key column in the store table

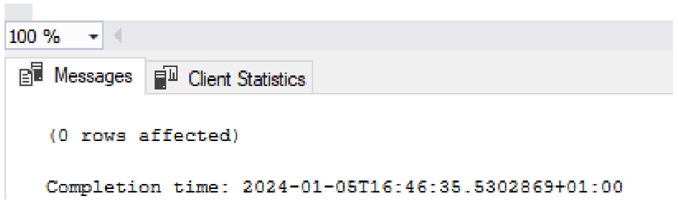
```
--Identify Missing Values on Key Column
SELECT COUNT (*) AS Missing_Values_on_Store_Table
FROM   Tbl_Stores
WHERE  Store_ID IS NULL
       OR
       Store_Location IS NULL
```



	Missing_Values_on_Store_Table
1	0

The generated result indicates that there are no missing values in store_ID and store_Location.

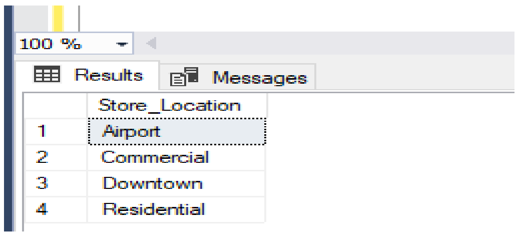
```
--Identify and Remove Duplicate Values
WITH CTE
AS
(SELECT Store_ID,
        Store_Name,
        Store_City,
        Store_Location,
        Store_Open_Date,
        ROW_NUMBER () OVER (PARTITION BY Store_ID ORDER BY Store_ID) AS ROW_N
FROM Tbl_Stores)

DELETE
FROM   CTE
WHERE  ROW_N > 1
```



```
(0 rows affected)

Completion time: 2024-01-05T16:46:35.5302869+01:00
```

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

```sql
-- Identify Inconsistencies in Data such as Variation in Spelling or
Categorization
SELECT DISTINCT Store_Location
FROM   Tbl_Stores
```



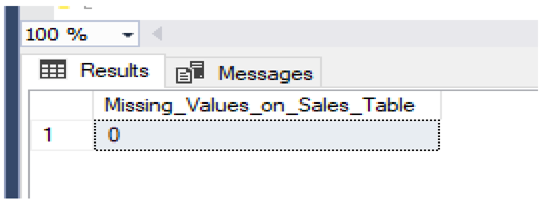
```
100 %
Results   Messages
   Store_Location
1  Airport
2  Commercial
3  Downtown
4  Residential
```

The generated result affirms the accurate spelling and proper categorization of all values in the Store Location column.

```sql
--Correcting Datatype on Key Column
ALTER TABLE Tbl_Stores
ALTER COLUMN Store_ID
INT

ALTER TABLE Tbl_Stores
ALTER COLUMN Store_Location
NVARCHAR (50)
```

Data Cleaning on Sales Table: The following data cleaning procedures were applied to the key column in the sales table.

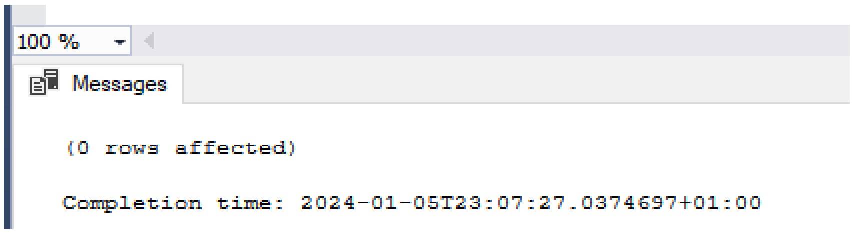
```sql
--Identify Missing Values on Key Column
SELECT COUNT (*) AS Missing_Values_on_Sales_Table
FROM   Tbl_Sales
WHERE  Sale_ID IS NULL
       OR
       [Date] IS NULL
       OR
       Store_ID IS NULL
       OR
       Product_ID IS NULL
       OR
       Units IS NULL
```



```
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 ROW_N
FROM Tbl_Sales)

DELETE
FROM CTE
WHERE ROW_N > 1
```



```
100 %
Messages

(0 rows affected)

Completion time: 2024-01-05T23:07:27.0374697+01:00
```

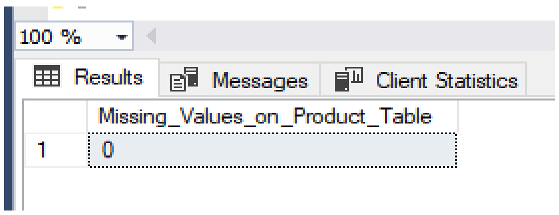
The generated result indicates that there are no duplicate values in sales table

```
--Correcting Datatype on Key Column
ALTER TABLE  Tbl_Sales
ALTER COLUMN Sale_ID
INT

ALTER TABLE  Tbl_Sales
ALTER COLUMN [Date]
DATE

ALTER TABLE  Tbl_Sales
ALTER COLUMN Store_ID
INT

ALTER TABLE  Tbl_Sales
ALTER COLUMN Product_ID
INT

ALTER TABLE  Tbl_Sales
ALTER COLUMN Units
INT

--Adding New Columns to the Sales Table
ALTER TABLE Tbl_Sales
ADD [Week_Day] NVARCHAR (50) NULL,
    [Month_Number] INT NULL,
    [Month_Name] NVARCHAR (50) NULL,
    [Year] INT NULL
```

```
--Update New Columns added to the Sales Table
UPDATE Tbl_Sales
SET    Week_Day = DATENAME([WEEKDAY], [Date]),
       Month_Number = MONTH([Date]),
       Month_Name = DATENAME([Month], [Date]),
       [Year] = YEAR([Date])
```

Data Cleaning on Product Table: The following data cleaning procedures were applied to the key column in the product table.

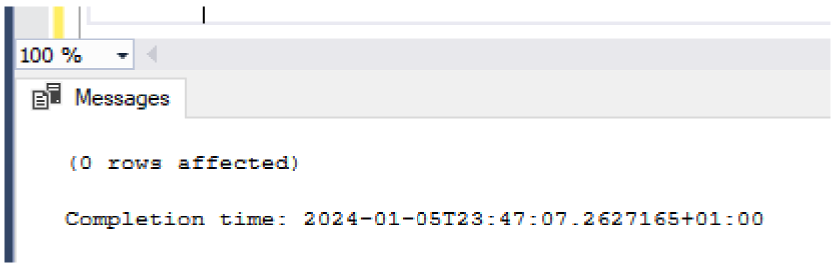
```
--Identify Missing Values on Key Column
SELECT COUNT (*) AS Missing_Values_on_Product_Table
FROM   Tbl_Products
WHERE  Product_ID IS NULL
       OR
       Product_Name IS NULL
       OR
       Product_Category IS NULL
       OR
       Product_Cost IS Null
       OR
       Product_Price IS NULL
```



	Missing_Values_on_Product_Table
1	0

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

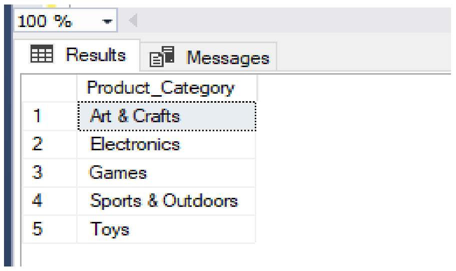
```
--Identify and Remove Duplicate Values
WITH CTE
AS
(SELECT Product_ID,
        Product_Name,
        Product_Category,
        Product_Cost,
        Product_Price,
        ROW_NUMBER () OVER (PARTITION BY Product_ID ORDER BY Product_ID) AS ROW_N
FROM Tbl_Products)

DELETE
FROM CTE
WHERE ROW_N > 1
```



```
(0 rows affected)

Completion time: 2024-01-05T23:47:07.2627165+01:00
```

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

```
-- Identify Inconsistencies in Data such as Variation in Spelling or
Categorization
SELECT DISTINCT Product_Category
FROM Tbl_Products
```



	Product_Category
1	Art & Crafts
2	Electronics
3	Games
4	Sports & Outdoors
5	Toys

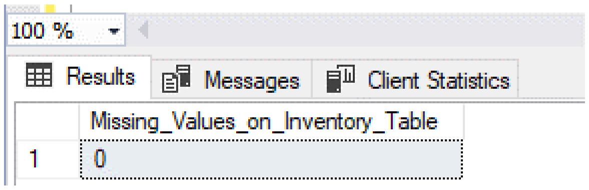
The generated result affirms the accurate spelling and proper categorization of all values in the Product Category column.

```
--Correcting Datatype on Key Column
ALTER TABLE  Tbl_Products
ALTER COLUMN Product_ID
INT

ALTER TABLE  Tbl_Products
ALTER COLUMN Product_Name
NVARCHAR (50)

ALTER TABLE  Tbl_Products
ALTER COLUMN Product_Category
NVARCHAR (50)

ALTER TABLE  Tbl_Products
ALTER COLUMN Product_Cost
FLOAT

ALTER TABLE  Tbl_Products
ALTER COLUMN Product_Price
FLOAT

--Adding New Column to the Product Table
ALTER TABLE Tbl_Products
ADD
Product_Profit
FLOAT NULL

--Update Product_Profit column on Product Table
UPDATE Tbl_Products
```

```
SET Product_Profit = Product_Price - Product_Cost
```

Data Cleaning on Inventory Table: The following data cleaning procedures were applied to the key column in the inventory table.

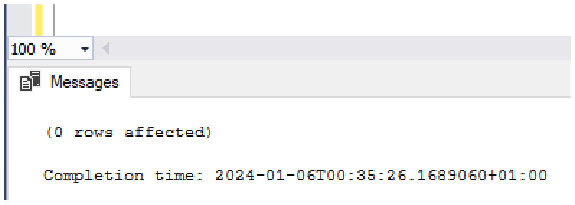
```
--Identify Missing Values on Key Columns
SELECT COUNT (*) AS Missing_Values_on_Inventory_Table
FROM    Tbl_Inventory
WHERE   Store_ID IS NULL
        OR
        Product_ID IS NULL
        OR
        Stock_On_Hand IS Null
```



	Missing_Values_on_Inventory_Table
1	0

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

```
--Identify and Remove Duplicate Values
WITH CTE
AS
(SELECT Store_ID,
        Product_ID,
        Stock_on_Hand,
        ROW_NUMBER () OVER (PARTITION BY Store_ID, Product_ID, Stock_on_Hand
        ORDER BY Store_ID) AS ROW_N
FROM Tbl_Inventory)

DELETE
FROM CTE
WHERE ROW_N > 1
```



```
(0 rows affected)

Completion time: 2024-01-06T00:35:26.1689060+01:00
```

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

```
--Correcting Datatype
ALTER TABLE  Tbl_Inventory
ALTER COLUMN Store_ID
INT

ALTER TABLE  Tbl_Inventory
ALTER COLUMN Product_ID
INT

ALTER TABLE  Tbl_Inventory
ALTER COLUMN Stock_on_Hand
INT
```

Established Relationships Between Tables

Establishing relationships between tables in SQL is a crucial aspect of database design, typically achieved through the use of primary key and foreign keys.

A primary key is a fundamental concept in database design, and it serves as a unique identifier for each record in a table. A primary key contains unique values for each record within the table. This uniqueness ensures that each record can be uniquely identified and distinguished from others in the same table.

A foreign key is a column or a set of columns in a relational database table that establishes a link between data in two tables. It serves to enforce referential integrity and create relationships between tables. A foreign key in one table refers to the primary key in another table. This relationship creates a link between the data in the two tables, ensuring that values in the foreign key column(s) correspond to existing values in the primary key column of the referenced table.

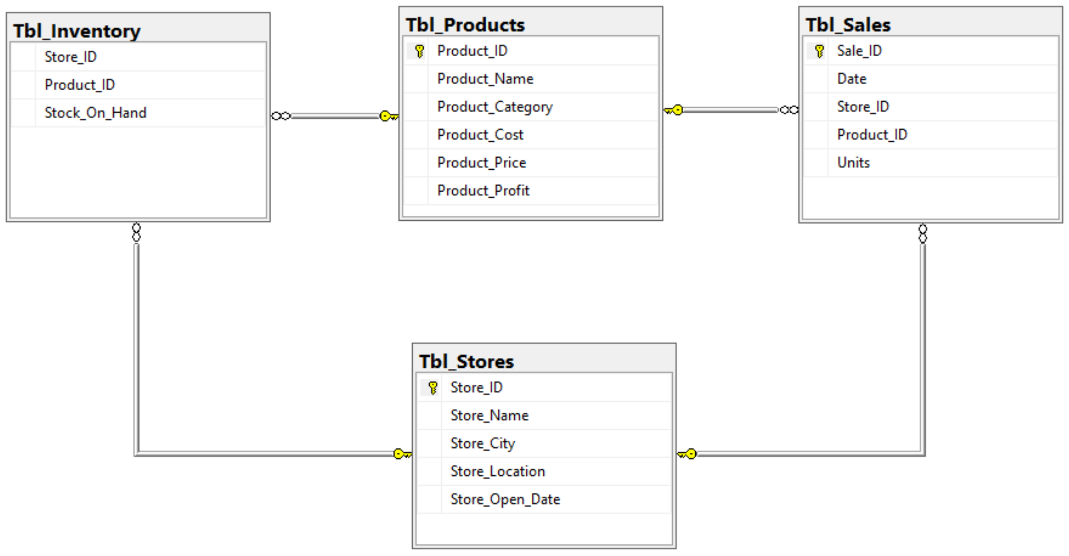
Relationship were established between tables by carrying the following process.

```
--Adding Primary Key on Product_ID on Product Table
ALTER TABLE Tbl_Products
ADD CONSTRAINT PK_Product_ID
PRIMARY KEY (Product_ID)

--Adding Primary Key on Sales_ID on Sales Table
ALTER TABLE Tbl_Sales
ADD CONSTRAINT PK_Sales_ID
PRIMARY KEY (Sale_ID)

--Adding Primary Key on Store_ID on Store Table
ALTER TABLE Tbl_Stores
ADD CONSTRAINT PK_Store_ID
PRIMARY KEY (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 Stores
Table
ALTER TABLE Tbl_Inventory
ADD CONSTRAINT FK_Store_ID_Tbl_Inventory_Store_ID_Tbl_Stores
FOREIGN KEY(Store_ID) REFERENCES Tbl_Stores (Store_ID)
```

```sql
--Creating a Foreign Key on Product_ID Column on Inventory Table References
Product_ID Column on Product Table to Establish Relationship Between Inventory
Table and Product Table
ALTER TABLE Tbl_Inventory
ADD CONSTRAINT FK_Product_ID_Product_ID_Tbl_Products
FOREIGN KEY(Product_ID) REFERENCES Tbl_Products (Product_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 Store
Table
ALTER TABLE Tbl_Sales
ADD CONSTRAINT FK_Store_ID_on_Tbl_Sales_Store_ID_Tbl_Stores
FOREIGN KEY(Store_ID) REFERENCES Tbl_Stores (Store_ID)

--Creating a Foreign Key on Product_ID Column on Sales Table References Product_ID
Column on Products Table to Establish Relationship Between Sales Table and
Products Table
ALTER TABLE Tbl_Sales
ADD CONSTRAINT FK_Product_ID_on_Tbl_Sales_Product_ID_Tbl_Products
FOREIGN KEY(Product_ID) REFERENCES Tbl_Products (Product_ID)
```

Creating relationships between tables is crucial for this analysis. It not only keeps the data organized but also helps in making queries faster, avoiding unnecessary duplication of data, and building a flexible database structure.

Below is the result of the relationship established between tables


Tbl_Inventory
Store_ID
Product_ID
Stock_On_Hand
Tbl_Products
Product_ID
Product_Name
Product_Category
Product_Cost
Product_Price
Product_Profit
Tbl_Sales
Sale_ID
Date
Store_ID
Product_ID
Units
Tbl_Stores
Store_ID
Store_Name
Store_City
Store_Location
Store_Open_Date