Data Cleaning using SQL on Housing Sales

Joel Choión

1 Introduction

This paper is a documentation on the data cleaning steps taken to clean the data in a housing sales data set. The cleaning steps were performed using SQL in the Microsoft SQL Server Management Studio. First, I inspected the data for anomalies or oddities. I was able to spot some wrong entries, some unforunate column design choices, and some duplicates. These are all going to be addressed in the following. The query is located in the same folder and can be inspected, if one wishes to do so.

2 Date and Time

First we see that in the SaleDate column the time is displayed by 00:00:00.000 in all rows. This is unnecessary information which we can remove. I have tried changing the column using UPDATE and SET, however somehow it is not changed after checking. Therefore we add a new column first, with the ALTER TABLE and ADD command, and then change that column to our liking.

```sql
ALTER TABLE Housing
   ADD SaleDateConverted Date;

UPDATE Housing
   SET SaleDateConverted = CONVERT(Date, SaleDate)
```

Now the column `SaleDateConverted` displays only the date of the sale without the time.

3 Null Values in Property Address Column

Next we are looking at the Property Address column. I have seen some NULL values when going through the data set initially. Therefore we need to decide what to do with these values. Since the property address does not usually change, if we were to somehow be able to retrieve that data, we would not need to remove the rows, which still include a lot of information. If we look at the ParcelID we can observe that for pretty much all the rows with the same ParcelID, the same Property Address exists. Hence, we can populate empty Property Addresses if we have other entries with the same ParcelID and a Property Address. To do this, we Join the same table on the ParcelID and look for entries with non matching UniqueIDs, and where the Property address in the first table is NULL. If in the second table the property address has a value, we can populate the NULL values.

```
SELECT a.ParcelID, a.PropertyAddress, b.ParcelID, b.PropertyAddress,
    ISNULL(a.PropertyAddress, b.PropertyAddress)
FROM PortfolioProject.dbo.Housing a
JOIN PortfolioProject.dbo.Housing b
   ON a.ParcelID = b.ParcelID
   AND a.[UniqueID ] <> b.[UniqueID ]
WHERE a.PropertyAddress is null


UPDATE a
SET PropertyAddress = ISNULL(a.PropertyAddress, b.PropertyAddress)
FROM PortfolioProject.dbo.Housing a
JOIN PortfolioProject.dbo.Housing b
   ON a.ParcelID = b.ParcelID
   AND a.[UniqueID ] <> b.[UniqueID ]
WHERE a.PropertyAddress is NULL
```

Populating Null values in the Property Address column

After updating the table, we no longer have any empty property addresses.

4 Splitting Address Column to Individual Columns

Currently, the Property Address entries are structured as such:

`Number␣Street,␣City`

So here, we want to split the address into two columns by splitting on the comma. By making use of the SUBSTRING function, we can split the column and create two new ones. The PropertyStreet and the PropertyCity columns.

The Owner Address is another address column, which if split, would provide a better overview and would be easier to use. Here, we have three delimiters, which would be annoying to deal with using Substring commands. Therefore we use the PARSENAME command. However, it only splits the column on periods, therefore we need to replace the commas to periods prior to using the command.

```
ALTER TABLE Housing
   ADD PropertyStreet Nvarchar(255)

ALTER TABLE Housing
   ADD PropertyCity Nvarchar(255)
-- Minus one to get rid of comma
UPDATE Housing
```

```
   SET PropertyStreet = SUBSTRING(PropertyAddress, 1, CHARINDEX(',', PropertyAddress)-1)
-- Plus two to get rid of comma and space
UPDATE Housing
   SET PropertyCity = SUBSTRING(PropertyAddress, CHARINDEX(',', PropertyAddress)+2,
       LEN(PropertyAddress))

SELECT OwnerAddress
FROM PortfolioProject.dbo.Housing

ALTER TABLE Housing
   Add OwnerStreet Nvarchar(255)

ALTER TABLE Housing
   Add OwnerCity Nvarchar(255)

ALTER TABLE Housing
   Add OwnerState Nvarchar(255)

UPDATE Housing
   SET OwnerStreet = PARSENAME(REPLACE(OwnerAddress,',','.'),3)

UPDATE Housing
   SET OwnerCity = PARSENAME(REPLACE(OwnerAddress,',','.'),2)

UPDATE Housing
   SET OwnerState = PARSENAME(REPLACE(OwnerAddress,',','.'),1)
```

Splitting Owner and Property Address into separate columns to make later use easier

5 Unifying SoldasVacant Column Entries

When looking through the data set, I saw some entries in the SoldasVacant column, where the values where N instead of No. After running a quick query, we see that Y, N, Yes, and No exist. We want to have a unified notation and since Yes and No is used more often, I decided to go with that convention. I used the CASE function to find and change instances, in which the value needed to be converted.

```
SELECT DISTINCT(SoldAsVacant), COUNT(SoldAsVacant)
FROM PortfolioProject.dbo.Housing
GROUP BY SoldAsVacant
ORDER BY 2

SELECT SoldAsVacant,
   CASE WHEN SoldAsVacant = 'Y' THEN 'Yes'
```

```
       WHEN SoldAsVacant = 'N' THEN 'No'
       ELSE SoldAsVacant
       END
FROM PortfolioProject.dbo.Housing

UPDATE Housing
   SET SoldAsVacant = CASE WHEN SoldAsVacant = 'Y' THEN 'Yes'
                    WHEN SoldAsVacant = 'N' THEN 'No'
                    ELSE SoldAsVacant
                    END
```

Replacing Y to Yes and N to No

6 Removing Duplicate Rows

If duplicate Rows exist, in this context, they can be considered as a false entry and can be removed. We look for duplicate rows using a CTE and finding rows which appear more often than once. Then we use the DELETE command to remove the duplicates, after having checked, that they actually are duplicates.

```
WITH RowNumCTE AS (
SELECT *,
   ROW_NUMBER() OVER (
   PARTITION BY ParcelID,
            PropertyStreet,
            SalePrice,
            SaleDateConverted,
            LegalReference
            ORDER BY
              ParcelID
              ) Row_Num
FROM PortfolioProject.dbo.Housing
)

DELETE
FROM RowNumCTE
WHERE Row_Num > 1
```

Removing duplicate rows using a CTE

7 Removing Unused Columns

Lastly, we remove columns which are not as important any longer, because we altered the columns by creating new ones. Generally it is not a good habit to delete columns permanently in the data set. Nonetheless, as this is a personal exercise and the data is not used by others after my cleaning, I just removed the deprecated columns using DROP.

```sql
SELECT *
FROM PortfolioProject.dbo.Housing

ALTER TABLE Housing
   DROP COLUMN OwnerAddress, PropertyAddress, SaleDate
```

8 Conclusion

Conclusively, I have cleaned data regarding house sales using SQL queries. I have split columns to make further usage easier, I have removed unnecessary information from columns, I have populated NULL values through inspection and deduction on the current existing values, I have removed inconsistencies, and removed duplicate entries.

Thank you for reading this document.