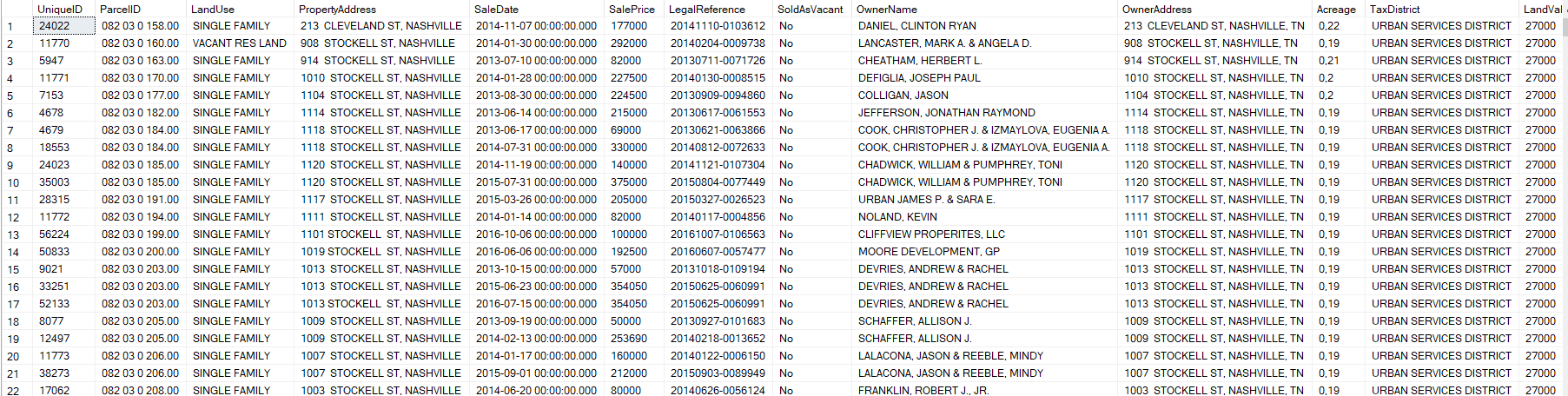
**Data Analyst   
Cleaning Project**  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
 **Facundo Cortez**  
  
  
**Introducción:**Aplicaremos Técnicas de Limpieza y Procesado de Datos  
  
  
Nuestra base de datos se llama:  
“**NASHVILLE HOUISING DATA”**  
  
Descripción:  
  
UniqueID -> ID Único por Fila  
ParcelID -> ID de la Parcela del Inmueble  
LandUse -> Uso que se le da. Ej: FAMILIAR  
PropertyAddress -> Dirección del Inmueble  
SaleDate -> Fecha en la que se generó la venta del Inmueble  
SalePrice -> Precio al cual se vendió el Inmueble  
LegalReference -> Referencia Legal  
SoldAsVacant -> Vendido como vacante  
OwnerName -> Nombre del Propietario  
OwnerAddress -> Dirección del Propietario  
Acreage -> Valor de Acres  
TaxDistrict -> Impuesto del Distrito  
LandValue -> Valor de la tierra  
BuildingValue -> Valor de la construcción  
TotalValue .> Valor Total  
YearBuilt -> Año de construcción  
Bedrooms -> Habitaciones  
Fullbath -> Baño completo  
Halfbath -> Medio baño  
  
  
  
  
  
  
 **Limpieza de Datos en SQL**

SELECT \*

FROM [dbo].[NashvilleHousing]



--------------------------------------------------------------------------------------

/\*

**-- Estandarizacion de formato de Fechas**

\*/

SELECT \*

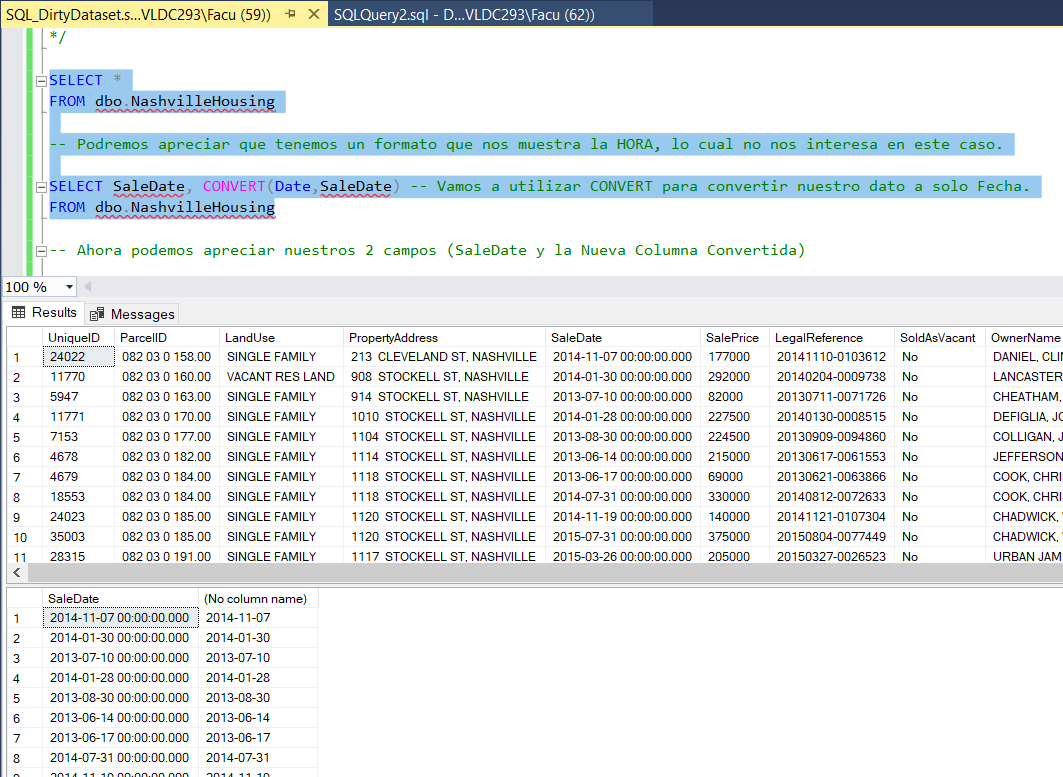
FROM dbo.NashvilleHousing

-- Podremos apreciar que tenemos un formato que nos muestra la HORA, lo cual no nos interesa en este caso.

SELECT SaleDate, CONVERT(Date,SaleDate) -- Vamos a utilizar CONVERT para convertir nuestro dato a solo Fecha.

FROM dbo.NashvilleHousing

-- Ahora podemos apreciar nuestros 2 campos (SaleDate y la Nueva Columna Convertida)



-- Adherimos una nueva Columna llamada SaleDate2

ALTER TABLE NashvilleHousing

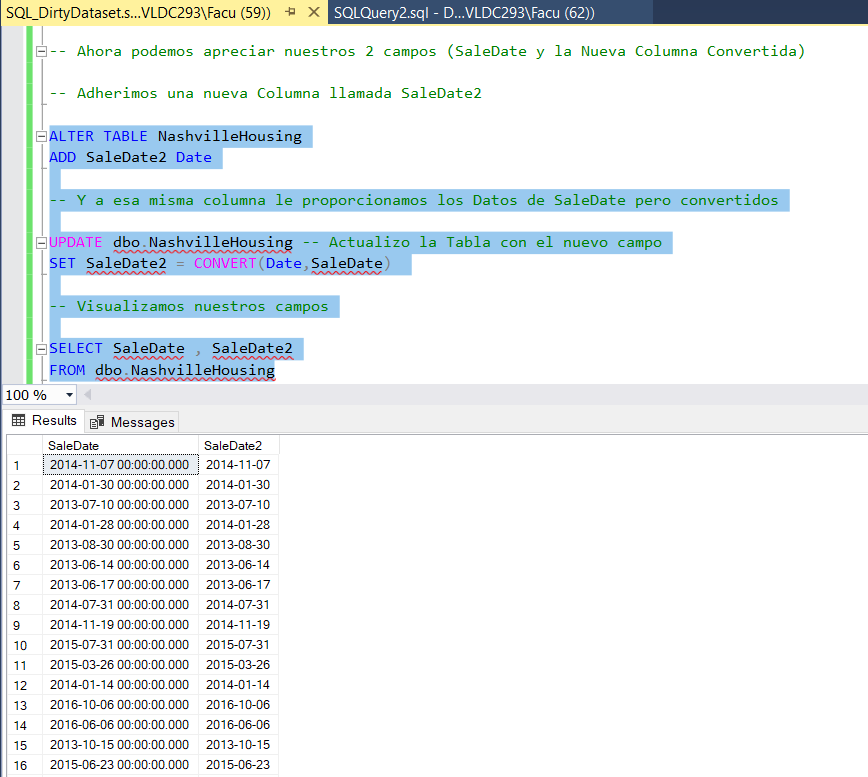
ADD SaleDate2 Date

-- Y a esa misma columna le proporcionamos los Datos de SaleDate pero convertidos

UPDATE dbo.NashvilleHousing -- Actualizo la Tabla con el nuevo campo

SET SaleDate2 = CONVERT(Date,SaleDate)

-- Visualizamos nuestros campos



SELECT SaleDate , SaleDate2

FROM dbo.NashvilleHousing

**-- DROP COLUMN**

/\*

\*/

--------------------------------------------------------------------------------------

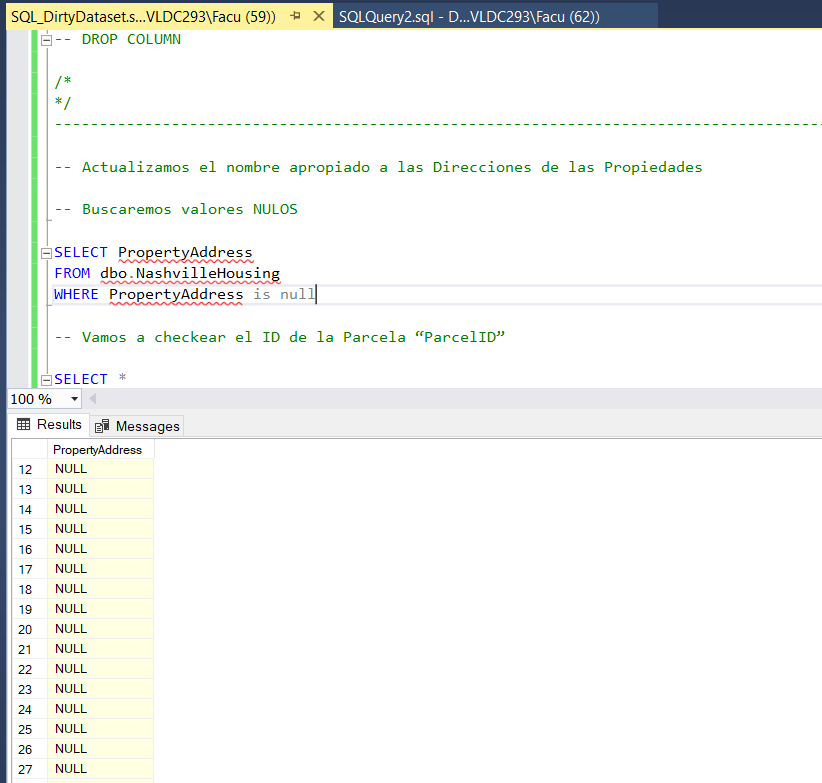
-- Actualizamos el nombre apropiado a las Direcciones de las Propiedades

-- Buscaremos valores NULOS

SELECT PropertyAddress

FROM dbo.NashvilleHousing

WHERE PropertyAddress is null



-- Vamos a checkear el ID de la Parcela “ParcelID”

SELECT \*

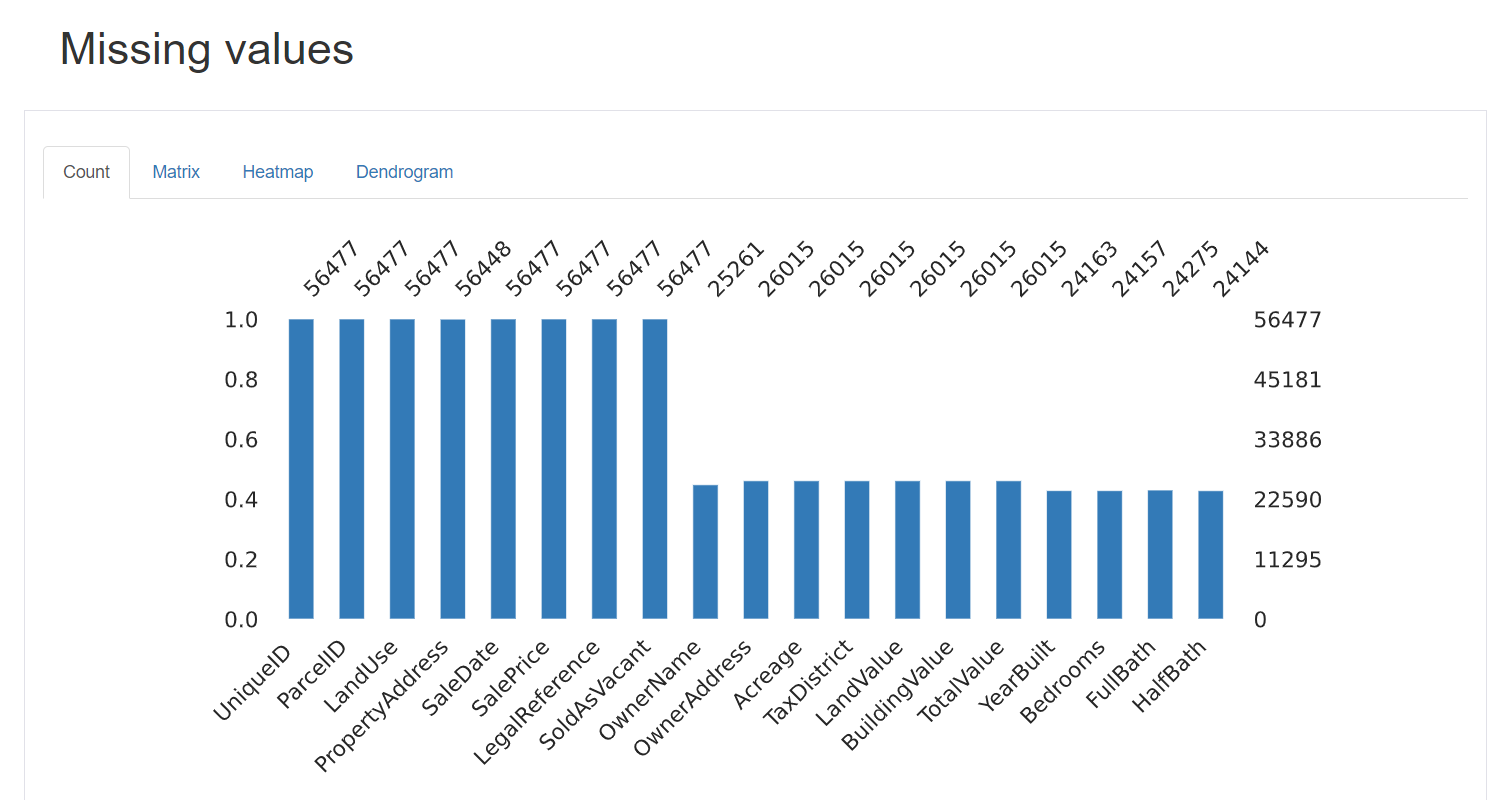
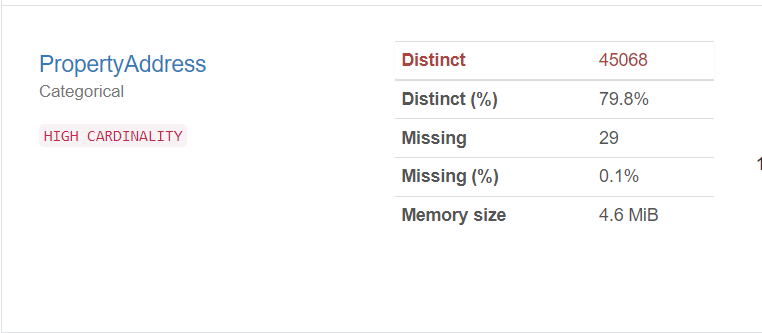
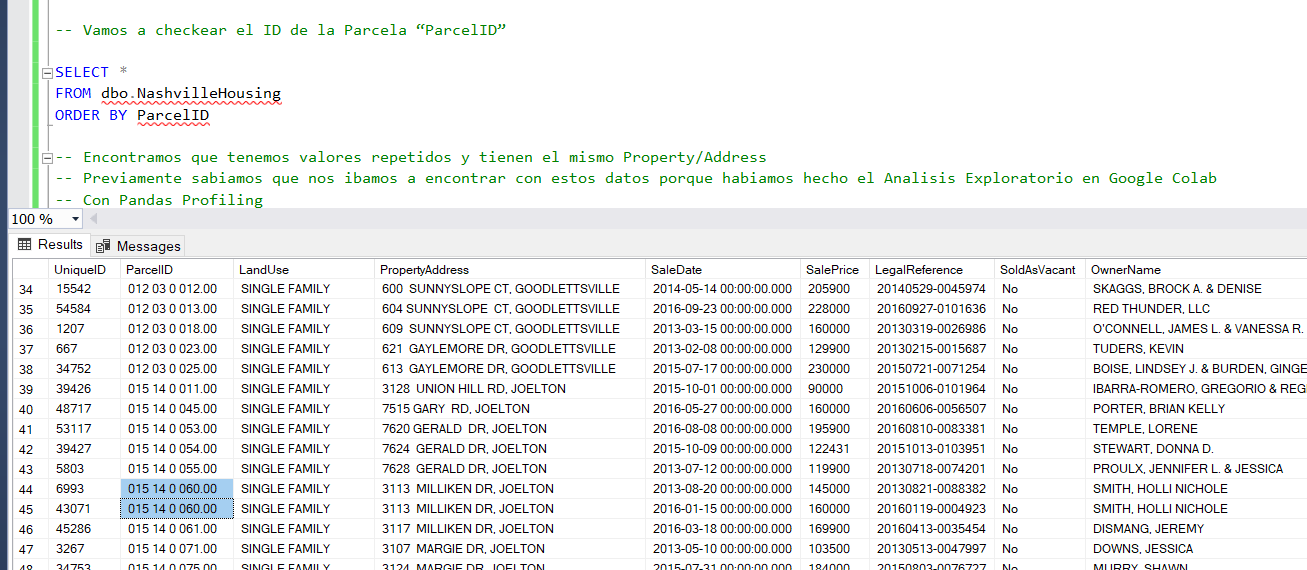
FROM dbo.NashvilleHousing

ORDER BY ParcelID

-- Encontramos que tenemos valores repetidos y tienen el mismo Property/Address

-- Previamente sabiamos que nos ibamos a encontrar con estos datos porque habiamos hecho el Analisis Exploratorio en Google Colab

-- Con Pandas Profiling



-- Entonces aplicando un JOIN en la misma Tabla

-- Estamos trayendo los mismos campos pero con la distincion del UniqueID

SELECT A.ParcelID , A.PropertyAddress , B.ParcelID, B.PropertyAddress

FROM dbo.NashvilleHousing A

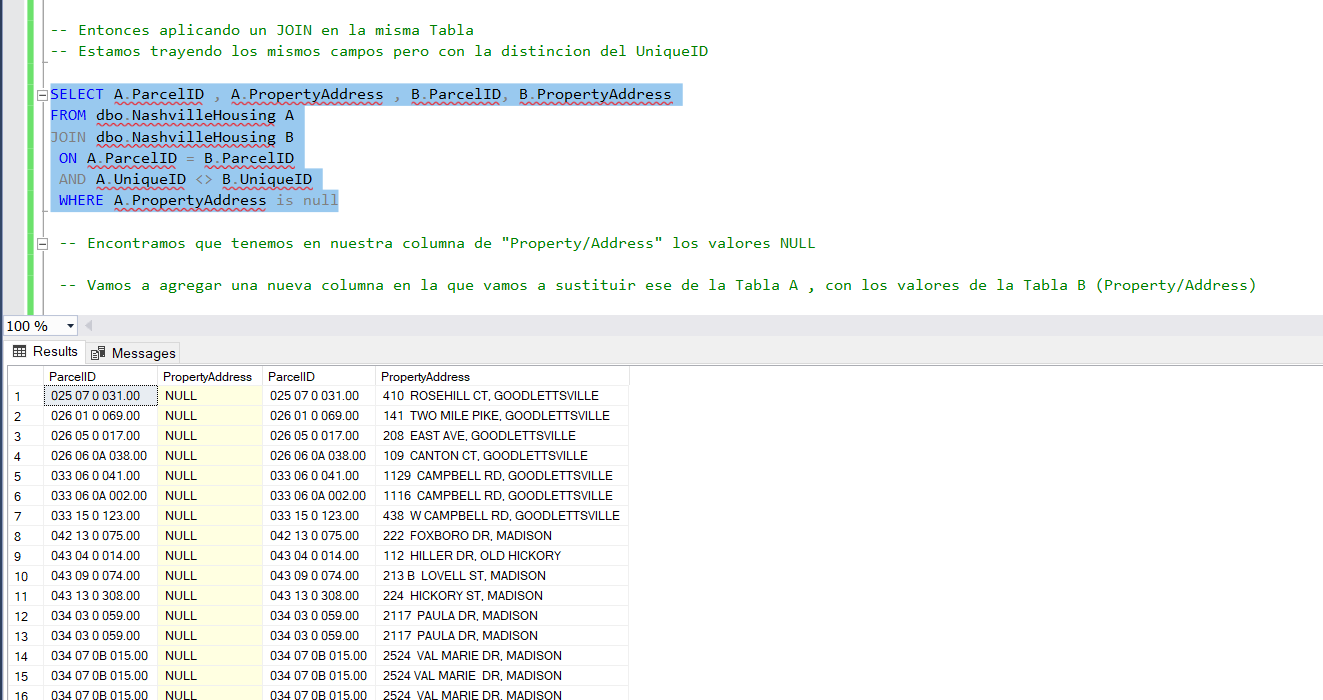
JOIN dbo.NashvilleHousing B

ON A.ParcelID = B.ParcelID

AND A.UniqueID <> B.UniqueID

WHERE A.PropertyAddress is null

-- Encontramos que tenemos en nuestra columna de "Property/Address" los valores NULL



-- Vamos a agregar una nueva columna en la que vamos a sustituir ese de la Tabla A , con los valores de la Tabla B (Property/Address)

-- CODIGO -> "PropertyAddress is NULL"

SELECT A.ParcelID , A.PropertyAddress , B.ParcelID, B.PropertyAddress, ISNULL (A.PropertyAddress , B.PropertyAddress)

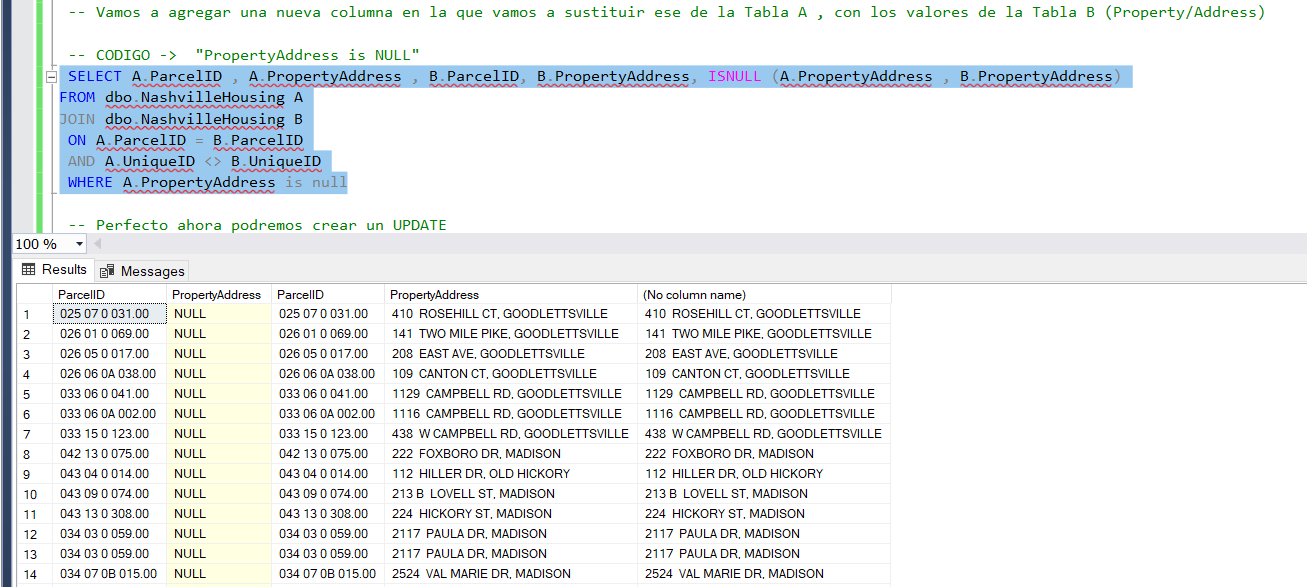
FROM dbo.NashvilleHousing A

JOIN dbo.NashvilleHousing B

ON A.ParcelID = B.ParcelID

AND A.UniqueID <> B.UniqueID

WHERE A.PropertyAddress is null



-- Perfecto ahora podremos crear un UPDATE

UPDATE A

SET PropertyAddress = ISNULL (A.PropertyAddress , B.PropertyAddress)

FROM dbo.NashvilleHousing A

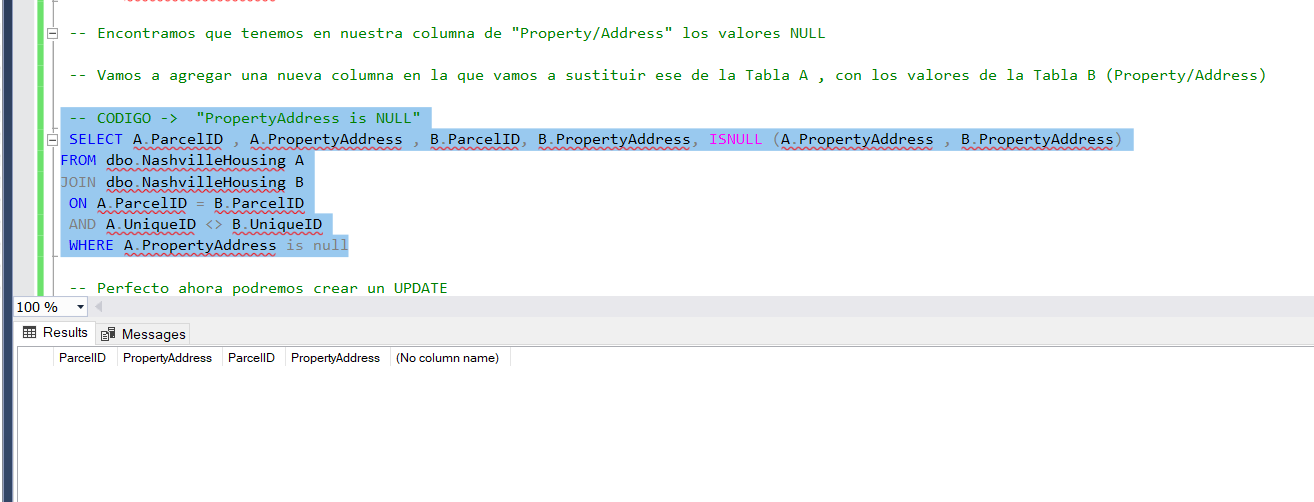
JOIN dbo.NashvilleHousing B

ON A.ParcelID = B.ParcelID

AND A.UniqueID <> B.UniqueID

WHERE A.PropertyAddress is null

-- Si volvemos a correr el CODIGO -> "PropertyAddress is NULL" . Veremos que no nos aparece ningun Valor Null ( Tabla vacía)

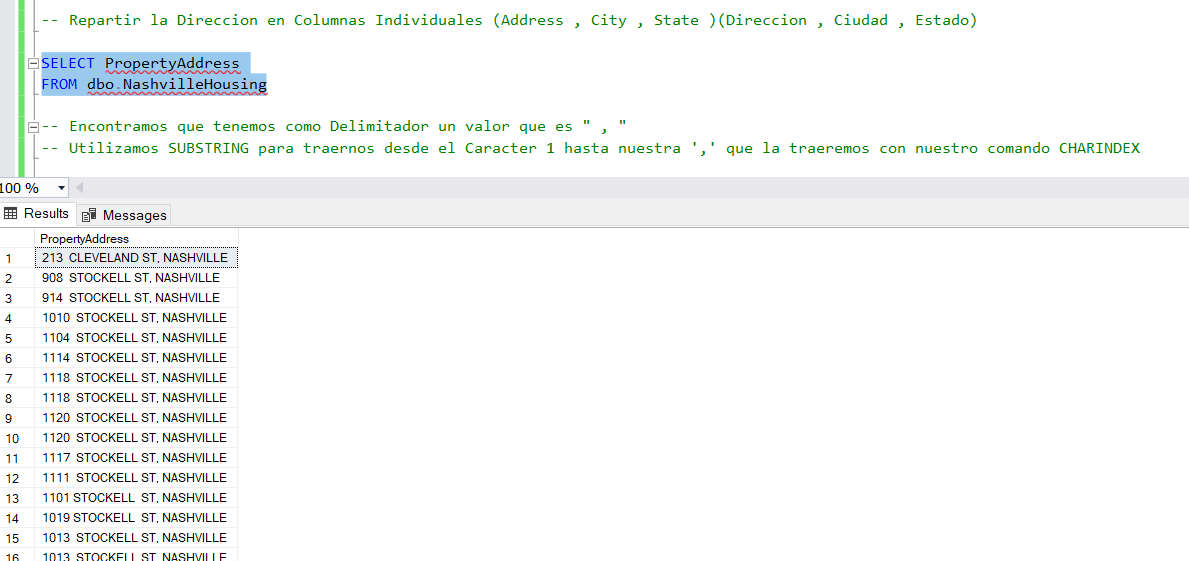


-------------------------------------------------------------------------------

-- Repartir la Direccion en Columnas Individuales (Address , City , State )(Direccion , Ciudad , Estado)

SELECT PropertyAddress

FROM dbo.NashvilleHousing



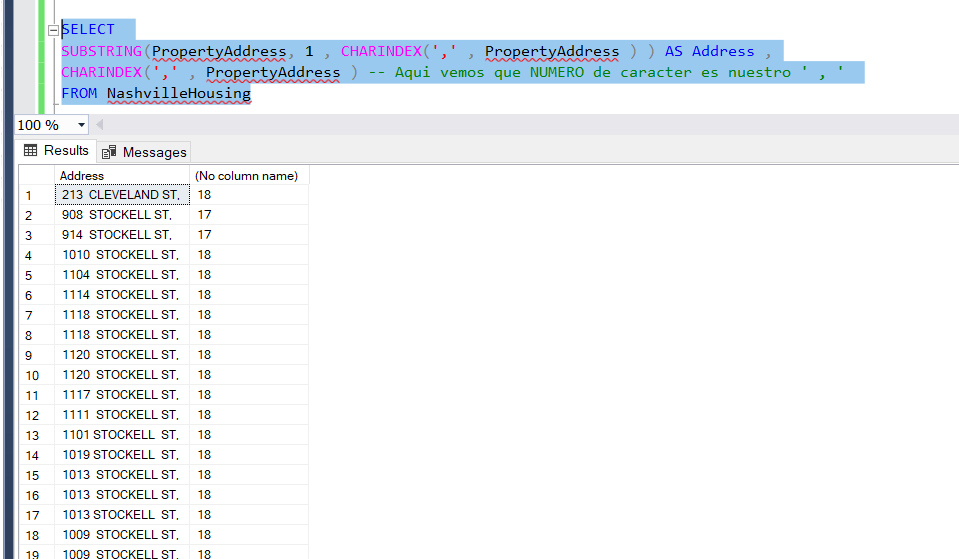
-- Encontramos que tenemos como Delimitador un valor que es " , "

-- Utilizamos SUBSTRING para traernos desde el Caracter 1 hasta nuestra ',' que la traeremos con nuestro comando CHARINDEX

SELECT

SUBSTRING(PropertyAddress, 1 , CHARINDEX(',' , PropertyAddress ) ) AS Address ,

CHARINDEX(',' , PropertyAddress ) -- Aqui vemos que NUMERO de caracter es nuestro ' , '

FROM NashvilleHousing  
  


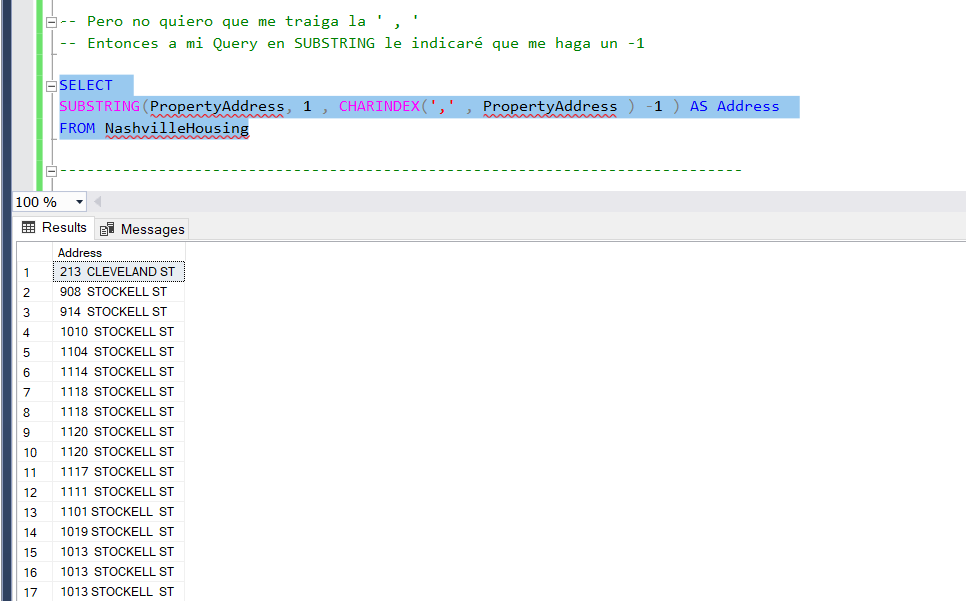
-- Pero no quiero que me traiga la ' , '

-- Entonces a mi Query en SUBSTRING le indicaré que me haga un -1

SELECT

SUBSTRING(PropertyAddress, 1 , CHARINDEX(',' , PropertyAddress ) -1 ) AS Address

FROM NashvilleHousing



----------------------------------------------------------------------------

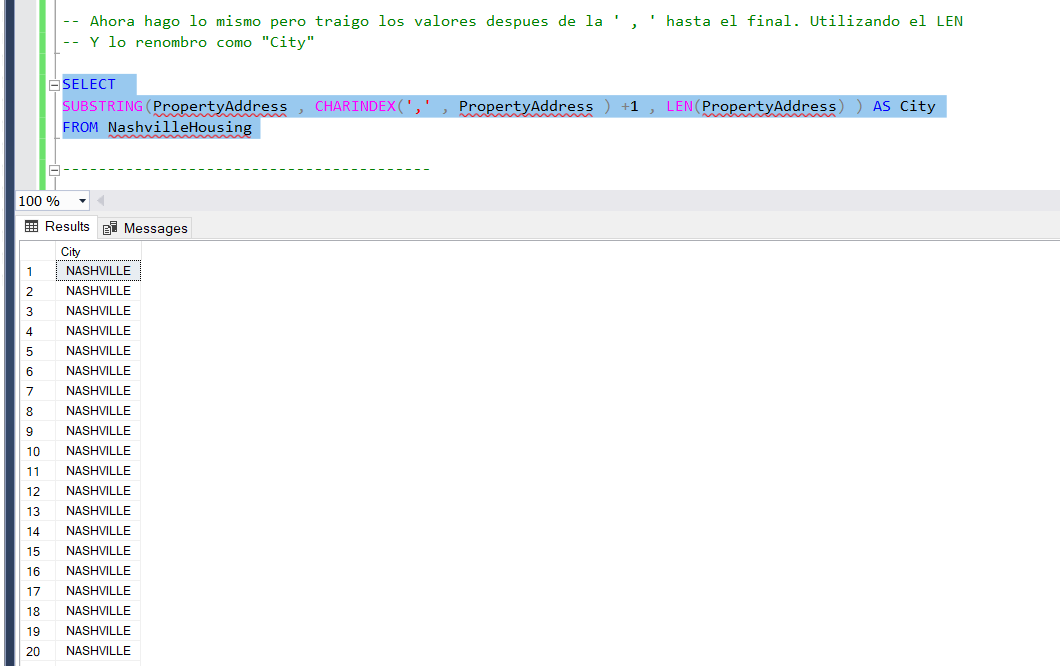
-- Ahora hago lo mismo pero traigo los valores despues de la ' , ' hasta el final. Utilizando el LEN

-- Y lo renombro como "City"

SELECT

SUBSTRING(PropertyAddress , CHARINDEX(',' , PropertyAddress ) +1 , LEN(PropertyAddress) ) AS City

FROM NashvilleHousing



-----------------------------------------

-- Voy a crear nuevas columnas

ALTER TABLE NashvilleHousing

ADD PropertyAdrees2 nvarchar(255)

ALTER TABLE NashvilleHousing

ADD City nvarchar (255)

----------------------------------------

-- Actualizo Address

UPDATE dbo.NashvilleHousing -- Actualizo la Tabla con el nuevo campo

SET PropertyAdrees2 = SUBSTRING(PropertyAddress, 1 , CHARINDEX(',' , PropertyAddress ) -1 )

-- Actualizo City

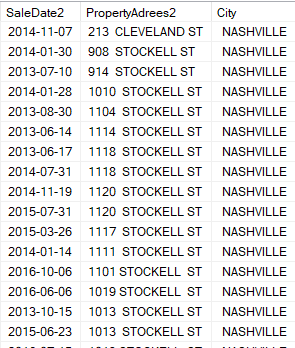
UPDATE dbo.NashvilleHousing -- Actualizo la Tabla con el nuevo campo

SET City = SUBSTRING(PropertyAddress , CHARINDEX(',' , PropertyAddress ) +1 , LEN(PropertyAddress) )

-- Vemos que hemos podido adherir nuestros 2 nuevos campos al Final de la Tabla

SELECT \*

FROM NashvilleHousing



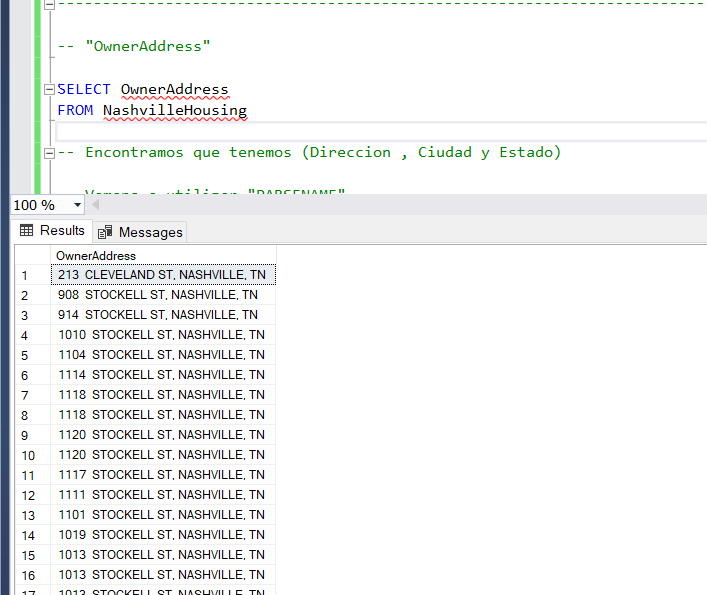
--------------------------------------------------------------------------------

-- "OwnerAddress"

SELECT OwnerAddress

FROM NashvilleHousing

-- Encontramos que tenemos (Direccion , Ciudad y Estado)



-- Vamos a utilizar "PARSENAME"

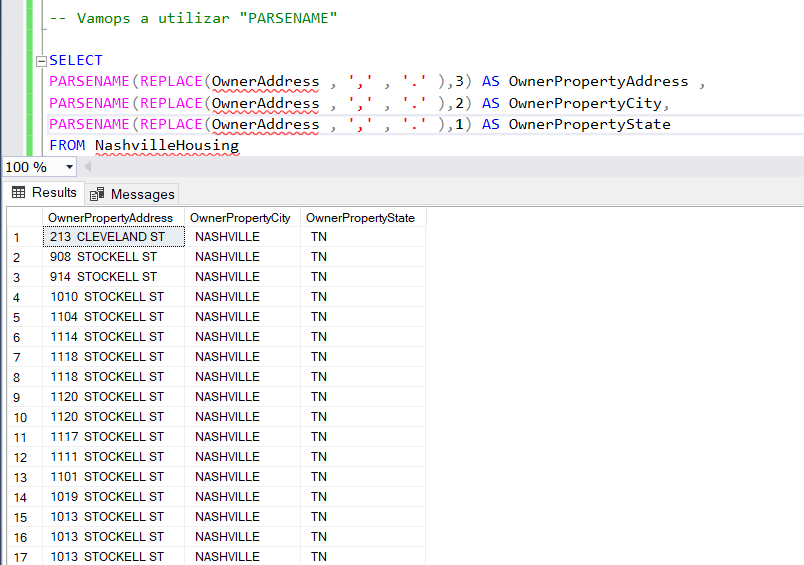
SELECT

PARSENAME(REPLACE(OwnerAddress , ',' , '.' ),3) AS OwnerPropertyAddress ,

PARSENAME(REPLACE(OwnerAddress , ',' , '.' ),2) AS OwnerPropertyCity,

PARSENAME(REPLACE(OwnerAddress , ',' , '.' ),1) AS OwnerPropertyState

FROM NashvilleHousing



-- Creamos Nuevas Columnas

ALTER TABLE NashvilleHousing

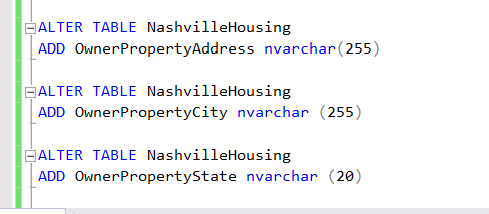
ADD OwnerPropertyAddress nvarchar(255)

ALTER TABLE NashvilleHousing

ADD OwnerPropertyCity nvarchar (255)

ALTER TABLE NashvilleHousing

ADD OwnerPropertyState nvarchar (20)



UPDATE dbo.NashvilleHousing -- Actualizo la Tabla con el nuevo campo

SET OwnerPropertyAddress = PARSENAME(REPLACE(OwnerAddress , ',' , '.'), 3)

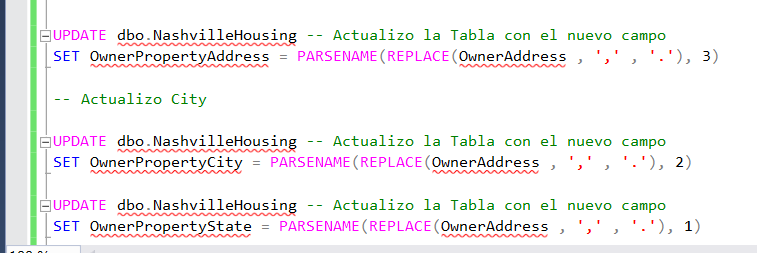
-- Actualizo City

UPDATE dbo.NashvilleHousing -- Actualizo la Tabla con el nuevo campo

SET OwnerPropertyCity = PARSENAME(REPLACE(OwnerAddress , ',' , '.'), 2)

UPDATE dbo.NashvilleHousing -- Actualizo la Tabla con el nuevo campo

SET OwnerPropertyState = PARSENAME(REPLACE(OwnerAddress , ',' , '.'), 1)



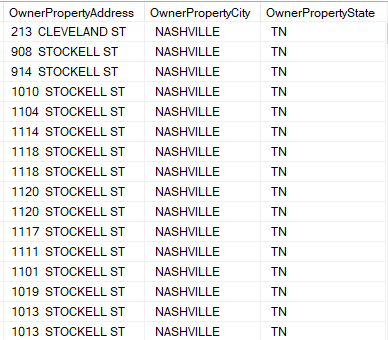
-- Vemos que hemos podido adherir nuestros 2 nuevos campos al Final de la Tabla

SELECT \*

FROM NashvilleHousing

-- Chekeamos y hemos generado de forma adecuada el PARSEADO

-- Y es mucho más útil



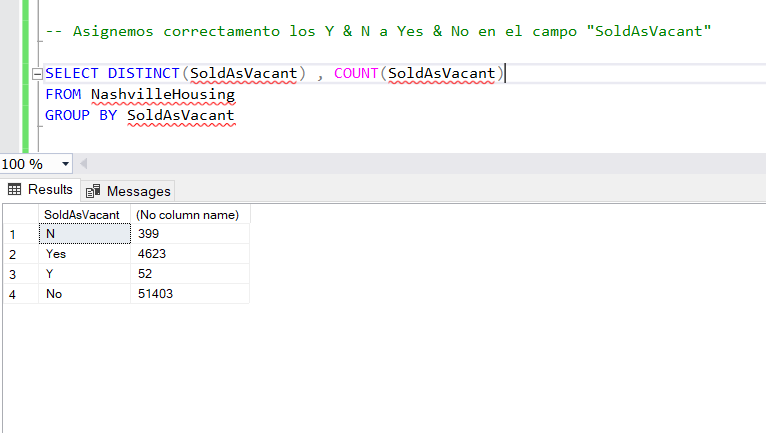
---------------------------------------------------------------------------------

-- Asignemos correctamento los Y & N a Yes & No en el campo "SoldAsVacant"

SELECT DISTINCT(SoldAsVacant) , COUNT(SoldAsVacant)

FROM NashvilleHousing

GROUP BY SoldAsVacant



Utilizamos el CASE WHEN para poder Asignar en una Nueva Columna los valores Y & N

Re-nombramos como NewSoldAsVacant

SELECT SoldAsVacant ,

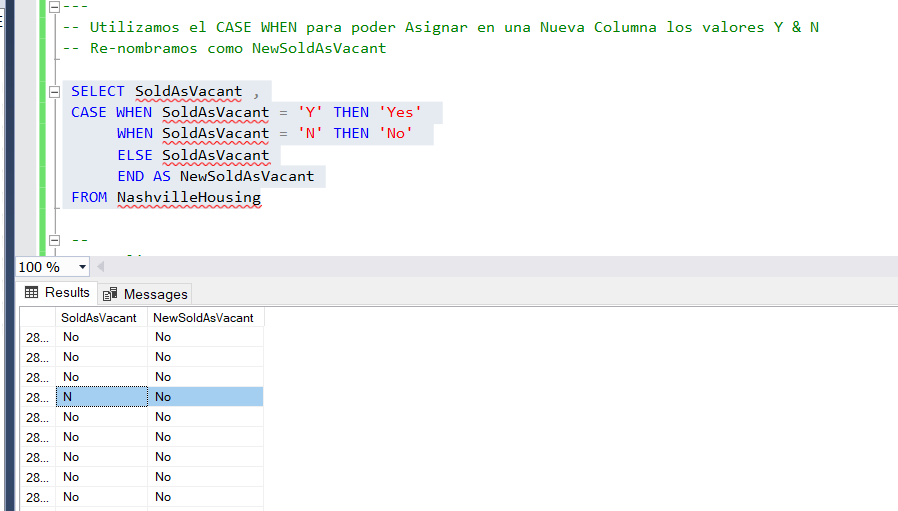
CASE WHEN SoldAsVacant = 'Y' THEN 'Yes'

WHEN SoldAsVacant = 'N' THEN 'No'

ELSE SoldAsVacant

END AS NewSoldAsVacant

FROM NashvilleHousing



--Actualizamos

UPDATE NashvilleHousing

SET SoldAsVacant =

CASE WHEN SoldAsVacant = 'Y' THEN 'Yes'

WHEN SoldAsVacant = 'N' THEN 'No'

ELSE SoldAsVacant

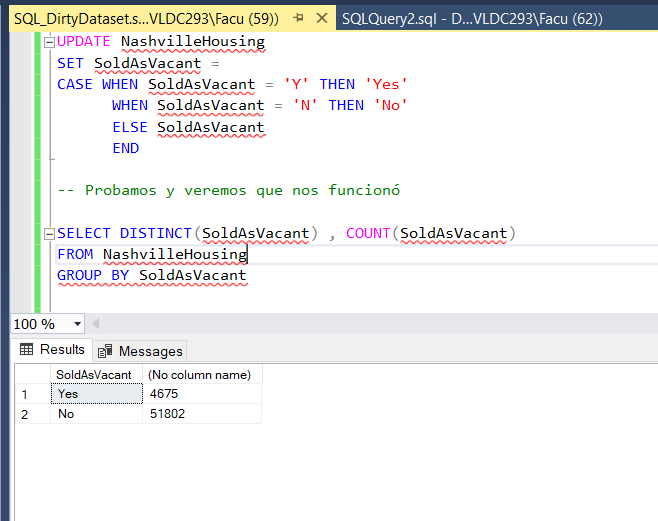
END

-- Probamos y veremos que nos funcionó

SELECT DISTINCT(SoldAsVacant) , COUNT(SoldAsVacant)

FROM NashvilleHousing

GROUP BY SoldAsVacant



-- Removamos Duplicados

-- Para identificarlos podemos utilizar RANKE, ROW\_NUMBER (La forma más simple)

-- Tambien Utilzaremos CTE (CommonTableExpression) WITH Querys

WITH NumeroFilaCTE AS (

SELECT \* ,

ROW\_NUMBER() OVER (

PARTITION BY ParcelID ,

PropertyAddress,

SalePrice,

SaleDate,

LegalReference

ORDER BY UniqueID

)

numero\_fila

FROM NashvilleHousing

)

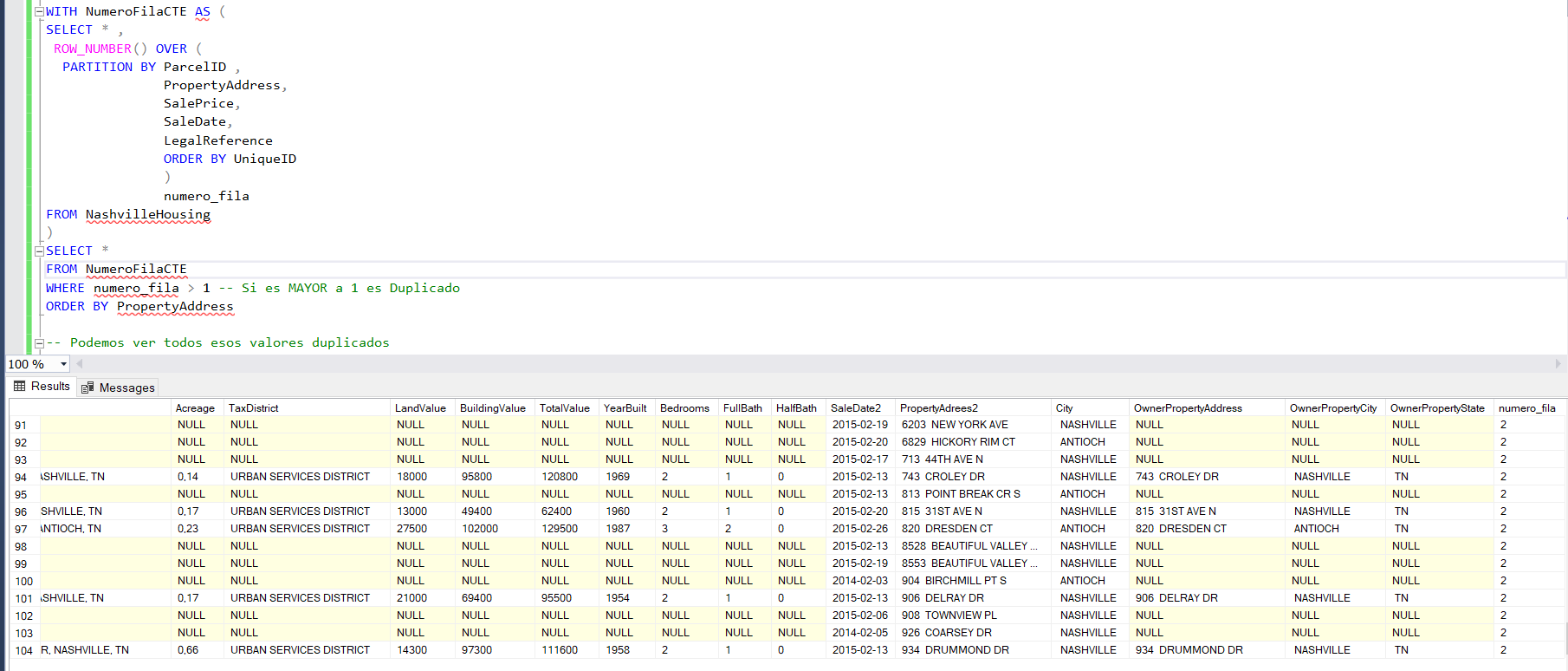
SELECT \*

FROM NumeroFilaCTE

WHERE numero\_fila > 1 -- Si es MAYOR a 1 es Duplicado

ORDER BY PropertyAddress

-- Podemos ver todos esos valores duplicados



-- Borramos los duplicados

WITH NumeroFilaCTE AS (

SELECT \* ,

ROW\_NUMBER() OVER (

PARTITION BY ParcelID ,

PropertyAddress,

SalePrice,

SaleDate,

LegalReference

ORDER BY

UniqueID

)

numero\_fila

FROM NashvilleHousing

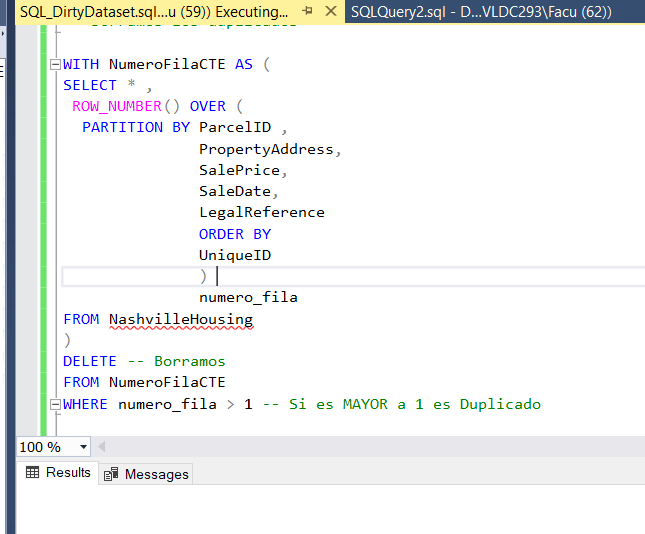
)

DELETE -- Borramos

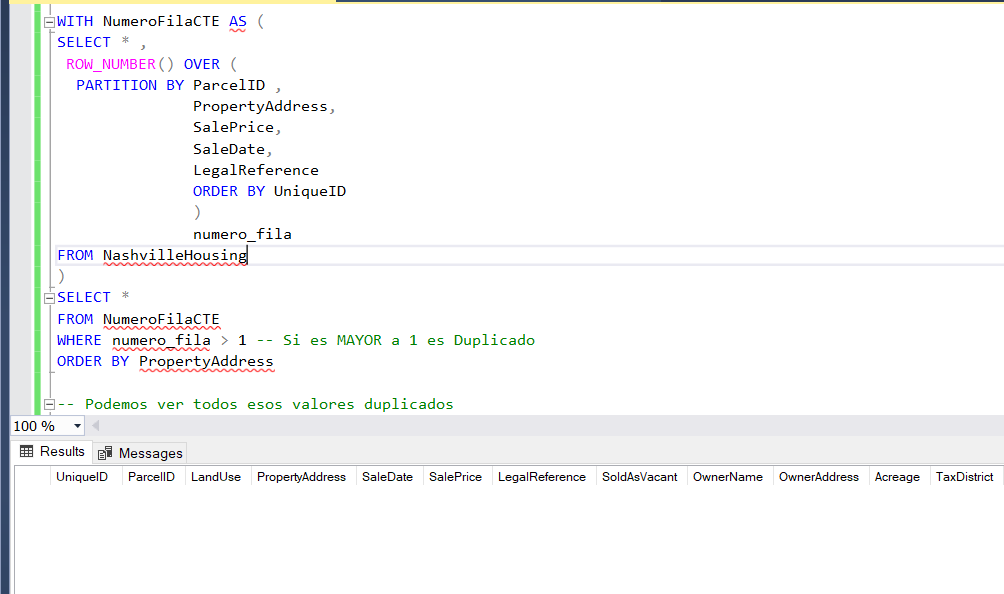
FROM NumeroFilaCTE

WHERE numero\_fila > 1 -- Si es MAYOR a 1 es Duplicado

-- Pudimos borrar los Duplicados



-- Si volvemos a mirar al Codigo del SELECT , veremos que ya no están ahi y nos devuelve una Tabla Vacía



----------------------------

-- Borraremos Columnas Inusuales y las que ya no nos sirven

-- Lo estamos haciendo solo para poder aprovechar el Dataset y jugar :)

SELECT \*

FROM NashvilleHousing

ALTER TABLE NashvilleHousing

DROP COLUMN OwnerAddress , PropertyAddress, SaleDate , TaxDistrict

