applying domain rules

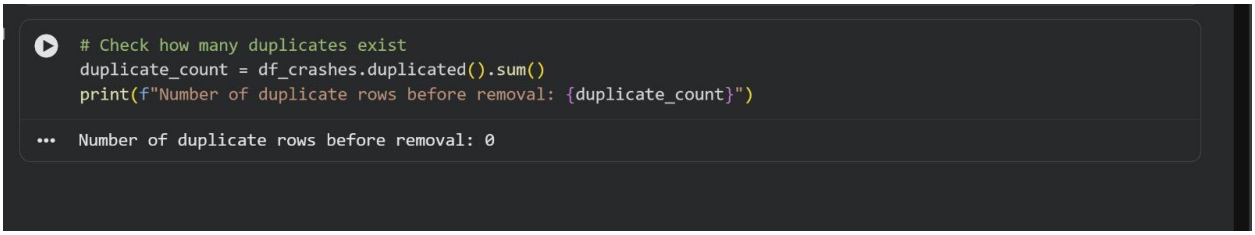
```
valid_boroughs = ['MANHATTAN', 'BROOKLYN', 'QUEENS', 'BRONX', 'STATEN ISLAND']
valid_zip_range = range(10001, 11698)

# Apply rules
df_crashes['BOROUGH'] = df_crashes['BOROUGH'].where(df_crashes['BOROUGH'].isin(valid_boroughs), "Unknown")
df_crashes['ZIP CODE'] = df_crashes['ZIP CODE'].where(df_crashes['ZIP CODE'].astype(str).astype(int).isin(va

# Latitude/Longitude bounds
df_crashes['LATITUDE'] = df_crashes['LATITUDE'].where((df_crashes['LATITUDE'] >= 40.49) & (df_crashes['LATIT
df_crashes['LONGITUDE'] = df_crashes['LONGITUDE'].where((df_crashes['LONGITUDE'] >= -74.27) & (df_crashes['L

print("Domain rules applied: categorical/geographic fields validated and cleaned.")
```

```
Domain rules applied: categorical/geographic fields validated and cleaned.
```

Standardize

```
[49]
✓ 8s
import pandas as pd

# --- Standardize Dates ---
# Convert CRASH DATE and CRASH TIME to proper datetime formats
df_crashes['CRASH DATE'] = pd.to_datetime(df_crashes['CRASH DATE'], errors='coerce')
df_crashes['CRASH TIME'] = pd.to_datetime(df_crashes['CRASH TIME'], format='%H:%M', errors='coerce').dt.time

# --- Standardize Strings ---
# Strip whitespace, convert to uppercase for consistency
string_cols = ['BOROUGH', 'VEHICLE TYPE CODE 1', 'VEHICLE TYPE CODE 2',
               'CONTRIBUTING FACTOR VEHICLE 1', 'CONTRIBUTING FACTOR VEHICLE 2']

for col in string_cols:
    df_crashes[col] = df_crashes[col].astype(str).str.strip().str.upper()

# --- Standardize Categories ---
# Define valid categories for BOROUGH
valid_boroughs = ['MANHATTAN', 'BROOKLYN', 'QUEENS', 'BRONX', 'STATEN ISLAND']
df_crashes['BOROUGH'] = df_crashes['BOROUGH'].where(df_crashes['BOROUGH'].isin(valid_boroughs), "UNKNOWN")

# Convert BOROUGH to categorical type
df_crashes['BOROUGH'] = df_crashes['BOROUGH'].astype('category')

print("Standardization complete: Dates, strings, and categories are now consistent.")

Standardization complete: Dates, strings, and categories are now consistent.
```

Tried to eliminate duplicates and found no duplicates



```
# Check how many duplicates exist
duplicate_count = df_crashes.duplicated().sum()
print(f"Number of duplicate rows before removal: {duplicate_count}")
```

```
Number of duplicate rows before removal: 0
```