Close

```
# Data Engineering Internship – Task Completion

Name: Catherina Jercy
Internship Role: Data Engineer Intern
Company: Cognify Technologies

Dataset: Railway_info.csv

Description:
This notebook contains data exploration, cleaning, transformatio
analysis, and visualization tasks performed as part of the Data
Engineering Internship.
```

Data Engineering Internship – Task Completion

Name: Catherina Jercy Internship Role: Data Engineer Intern Company: Cognify Technologies

Dataset: Railway_info.csv

Description: This notebook contains data exploration, cleaning, transformation, analysis, and visualization tasks performed as part of the Data Engineering Internship.

```python
import pandas as pd

# Load the dataset
df = pd.read_csv("Railway_info.csv")

# Display first 10 rows
print(" 🔹 First 10 Rows:")
display(df.head(10))

# Display info / structure
print("\n 🔹 Dataset Info:")
df.info()

# Display missing values
print("\n 🔹 Missing Values:")
print(df.isnull().sum())
```

🔹 First 10 Rows:

	Train_No	Train_Name	Source_Station_Name	Destination_Station_Name	days
0	107	SWV-MAO-VLNK	SAWANTWADI ROAD	MADGOAN JN.	Saturday
1	108	VLNK-MAO-SWV	MADGOAN JN.	SAWANTWADI ROAD	Friday
2	128	MAO-KOP SPEC	MADGOAN JN.	CHHATRAPATI SHAHU MAHARAJ TERMINUS	Friday
3	290	PALACE ON WH	DELHI-SAFDAR JANG	DELHI-SAFDAR JANG	Wednesday
4	401	BSB BHARATDA	AURANGABAD	VARANASI JN.	Saturday
5	421	LKO-SVDK FTR	LUCKNOW JN.	SHRI MATA VAISHNO DEVI KATRA	Tuesday
6	422	SVDK-LKO FTR	SHRI MATA VAISHNO DEVI KATRA	LUCKNOW JN.	Monday
7	477	FTR TRAIN NO	SIRSA	SIRSA	Sunday
8	502	RJPB-UMB FTR	RAJENDRANAGAR TERMINAL	AMBALA CANTT JN	Monday
9	504	PNBE-BTI FTR	PATNA JN.	BATHINDA JN	Wednesday

```
 🔹 Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11113 entries, 0 to 11112
Data columns (total 5 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Train_No                  11113 non-null  int64
 1   Train_Name                11113 non-null  object
 2   Source_Station_Name       11113 non-null  object
 3   Destination_Station_Name  11113 non-null  object
 4   days                      11113 non-null  object
dtypes: int64(1), object(4)
memory usage: 434.2+ KB

 🔹 Missing Values:
Train_No                   0
Train_Name                 0
Source_Station_Name        0
Destination_Station_Name   0
days                       0
dtype: int64
```

```python
# Summary statistics for numerical columns
print(" 🔹 Numerical Summary:")
display(df.describe())
```

```
# Summary stats for categorical columns
print("\n 🔹 Categorical Summary:")
for col in df.select_dtypes(include='object').columns:
    print(f"\nColumn: {col}")
    print("Unique values:", df[col].nunique())
    print("Most frequent:", df[col].mode()[0])
    print("Sample values:", df[col].unique()[:5])
```

```
🔹 Numerical Summary:
```

	Train_No
count	11113.000000
mean	49190.570413
std	28515.986645
min	107.000000
25%	22607.000000
50%	47174.000000
75%	68012.000000
max	99908.000000

```
🔹 Categorical Summary:

Column: Train_Name
Unique values: 7580
Most frequent: TBM-MSB EMU
Sample values: ['SWV-MAO-VLNK' 'VLNK-MAO-SWV' 'MAO-KOP SPEC' 'PALACE ON WH'
 'BSB BHARATDA']

Column: Source_Station_Name
Unique values: 921
Most frequent: CST-MUMBAI
Sample values: ['SAWANTWADI ROAD' 'MADGOAN JN.' 'DELHI-SAFDAR JANG' 'AURANGABAD'
 'LUCKNOW JN.']

Column: Destination_Station_Name
Unique values: 924
Most frequent: CST-MUMBAI
Sample values: ['MADGOAN JN.' 'SAWANTWADI ROAD' 'CHHATRAPATI SHAHU MAHARAJ TERMINUS'
 'DELHI-SAFDAR JANG' 'VARANASI JN.']

Column: days
Unique values: 14
Most frequent: Friday
Sample values: ['Saturday' 'Friday' 'Wednesday' 'Tuesday' 'Monday']
```

```
print(" 🔹 Trains from CST-MUMBAI:")
display(df[df['Source_Station_Name'] == 'CST-MUMBAI'].head())

print("\n 🔹 Trains going to DELHI:")
display(df[df['Destination_Station_Name'] == 'DELHI'].head())

print("\n 🔹 Top 10 trains with highest Train_No:")
display(df.sort_values(by='Train_No', ascending=False).head(10))

print("\n 🔹 Top 10 trains with lowest Train_No:")
display(df.sort_values(by='Train_No', ascending=True).head(10))
```

🔹 Trains from CST-MUMBAI:

	Train_No	Train_Name	Source_Station_Name	Destination_Station_Name	days
14	1011	CSMT-NGP SF	CST-MUMBAI	NAGPUR JN.(CR)	Thursday
31	2025	CSMT-KRMI SF	CST-MUMBAI	KARMALI	Tuesday
273	10103	MANDOVI EXPR	CST-MUMBAI	MADGOAN JN.	Thursday
275	10111	KONKAN KANYA	CST-MUMBAI	MADGOAN JN.	Friday
285	11007	DECCAN EXPRE	CST-MUMBAI	PUNE JN.	Wednesday

🔹 Trains going to DELHI:

Train_No	Train_Name	Source_Station_Name	Destination_Station_Name	days

🔹 Top 10 trains with highest Train_No:

	Train_No	Train_Name	Source_Station_Name	Destination_Station_Name	days
11112	99908	EMU	PUNE JN.	TALEGAON	Sunday
11111	99907	EMU	TALEGAON	PUNE JN.	Thursday
11110	99906	EMU	PUNE JN.	TALEGAON	Wednesday
11109	99905	EMU	TALEGAON	SHIVAJINAGAR	Monday
11108	99904	PUNE-TGN EMU	PUNE JN.	TALEGAON	Tuesday
11107	99903	TGN-PUNE EMU	TALEGAON	PUNE JN.	Wednesday
11106	99902	PUNE-TGN EMU	PUNE JN.	TALEGAON	Sunday
11105	99901	TGN-PUNE EMU	TALEGAON	PUNE JN.	Tuesday
11104	99836	PUNE-TGN EMU	PUNE JN.	LONAVLA	Saturday
11103	99835	LNL-PUNE EMU	LONAVLA	PUNE JN.	Tuesday

🔹 Top 10 trains with lowest Train_No:

	Train_No	Train_Name	Source_Station_Name	Destination_Station_Name	days
0	107	SWV-MAO-VLNK	SAWANTWADI ROAD	MADGOAN JN.	Saturday
1	108	VLNK-MAO-SWV	MADGOAN JN.	SAWANTWADI ROAD	Friday
2	128	MAO-KOP SPEC	MADGOAN JN.	CHHATRAPATI SHAHU MAHARAJ TERMINUS	Friday
3	290	PALACE ON WH	DELHI-SAFDAR JANG	DELHI-SAFDAR JANG	Wednesday
4	401	BSB BHARATDA	AURANGABAD	VARANASI JN.	Saturday
5	421	LKO-SVDK FTR	LUCKNOW JN.	SHRI MATA VAISHNO DEVI KATRA	Tuesday
6	422	SVDK-LKO FTR	SHRI MATA VAISHNO DEVI KATRA	LUCKNOW JN.	Monday
7	477	FTR TRAIN NO	SIRSA	SIRSA	Sunday
8	502	RJPB-UMB FTR	RAJENDRANAGAR TERMINAL	AMBALA CANTT JN	Monday
9	504	PNBE-BTI FTR	PATNA JN.	BATHINDA JN	Wednesday

```
print("🔹 Trains going to any DELHI station:")
delhi_trains = df[df['Destination_Station_Name'].str.contains('DELHI', case=False, na=False)]
display(delhi_trains.head(10))

print("\nTotal trains going to DELHI region:", len(delhi_trains))
```

```
🔹 Trains going to any DELHI station:
```

	Train_No	Train_Name	Source_Station_Name	Destination_Station_Name	days
3	290	PALACE ON WH	DELHI-SAFDAR JANG	DELHI-SAFDAR JANG	Wednesday
63	4410	SVDK-DLI SPL	SHRI MATA VAISHNO DEVI KATRA	DELHI JN.	Wednesday
260	9005	BCT NDLS BI	MUMBAI CENTRAL	NEW DELHI	Wednesday
432	12001	BPL - NDLS S	HABIBGANJ	NEW DELHI	Monday
434	12003	LKO-NDLS SHA	LUCKNOW JN.	NEW DELHI	Sunday
437	12006	KLK-NDLS SHA	KALKA	NEW DELHI	Sunday
443	12012	KLK-NDLS SHA	KALKA	NEW DELHI	Monday
445	12014	ASR-NDLS SHA	AMRITSAR JN.	NEW DELHI	Friday
447	12016	AII-NDLS SHA	AJMER JN.	NEW DELHI	Monday
449	12018	DDN-NDLS SHA	DEHRA DUN	NEW DELHI	Saturday

```
Total trains going to DELHI region: 219
```

```python
# 1. Count trains by source station
print(" 🔹 Top 10 source stations by number of trains:")
src_count = df['Source_Station_Name'].value_counts().head(10)
display(src_count)

# 2. Count trains by destination station
print("\n 🔹 Top 10 destination stations by number of trains:")
dest_count = df['Destination_Station_Name'].value_counts().head(10)
display(dest_count)

# 3. Popular routes (Source → Destination pairs)
print("\n 🔹 Top 10 most frequent routes:")
routes = df.groupby(['Source_Station_Name', 'Destination_Station_Name']).size().reset_index(name='Count')
routes = routes.sort_values(by='Count', ascending=False).head(10)
display(routes)
```

```
🔹 Top 10 source stations by number of trains:
```

Source_Station_Name	count
CST-MUMBAI	513
SEALDAH	372
CHENNAI BEACH	339
HOWRAH JN.	338
KALYAN JN	285
THANE	186
PANVEL	141
TAMBARAM	140
MOOR MARKET	135
VELACHEERY	115

dtype: int64

```
🔹 Top 10 destination stations by number of trains:
```

Destination_Station_Name	count
CST-MUMBAI	514
SEALDAH	373
CHENNAI BEACH	342
HOWRAH JN.	337
KALYAN JN	284
THANE	194
PANVEL	144
TAMBARAM	140
MOOR MARKET	132
VELACHEERY	118

dtype: int64

```
🔹 Top 10 most frequent routes:
```

	Source_Station_Name	Destination_Station_Name	Count
928	CHENNAI BEACH	TAMBARAM	137
4400	TAMBARAM	CHENNAI BEACH	137
1127	CST-MUMBAI	PANVEL	94
3490	PANVEL	CST-MUMBAI	93
3892	RAVLI JN	CST-MUMBAI	90
1129	CST-MUMBAI	RAVLI JN	90
4716	VELACHEERY	CHENNAI BEACH	89
932	CHENNAI BEACH	VELACHEERY	87
1132	CST-MUMBAI	THANE	77
4443	THANE	CST-MUMBAI	72

Next steps: Generate code with routes New interactive sheet

```
# 1. Check duplicate rows
print("🔹 Duplicate rows count:", df.duplicated().sum())

# 2. Check duplicate train numbers
print("🔹 Duplicate train numbers:", df['Train_No'].duplicated().sum())

# 3. Check leading/trailing spaces
print("\n🔹 Checking whitespace issues:")
for col in df.columns:
    if df[col].dtype == 'object':
        whitespace = df[col].str.startswith(' ') | df[col].str.endswith(' ')
        print(col, " → whitespace entries:", whitespace.sum())
```

```
# 4. Check empty strings instead of NaN
print("\n🔹 Empty string entries:")
for col in df.select_dtypes(include='object').columns:
    print(col, ":", (df[col] == '').sum())
```

```
🔹 Duplicate rows count: 0
🔹 Duplicate train numbers: 0

🔹 Checking whitespace issues:
Train_Name  → whitespace entries: 0
Source_Station_Name  → whitespace entries: 0
Destination_Station_Name  → whitespace entries: 0
days  → whitespace entries: 0

🔹 Empty string entries:
Train_Name : 0
Source_Station_Name : 0
Destination_Station_Name : 0
days : 0
```

```
# 1. Train frequency by days
print("🔹 Train frequency by operating days:")
day_counts = df['days'].value_counts()
display(day_counts)

# 2. Top 10 busiest source stations (from previous result)
print("\n🔹 Top 10 busiest source stations:")
display(df['Source_Station_Name'].value_counts().head(10))

# 3. Top 10 busiest destination stations (from previous result)
print("\n🔹 Top 10 busiest destination stations:")
display(df['Destination_Station_Name'].value_counts().head(10))

# 4. Compare source vs destination for imbalance
print("\n🔹 Source vs Destination station imbalance:")
src = df['Source_Station_Name'].value_counts()
dest = df['Destination_Station_Name'].value_counts()
imbalance = (src - dest).sort_values(ascending=False).head(10)
display(imbalance)
```

◆ Train frequency by operating days:

days	count
Friday	1471
Tuesday	1454
Wednesday	1448
Saturday	1441
Sunday	1432
Thursday	1372
Monday	1342

```
import matplotlib.pyplot as plt

# 1. Train frequency by day
plt.figure(figsize=(10,5))
day_counts.plot(kind='bar')
plt.title("Train Frequency by Operating Days")
plt.xlabel("Days of Week")
plt.ylabel("Number of Trains")
plt.show()

# 2. Top 10 source stations
plt.figure(figsize=(10,5))
df['Source_Station_Name'].value_counts().head(10).plot(kind='bar')
plt.title("Top 10 Source Stations")
plt.xlabel("Station Name")
plt.ylabel("Number of Trains Originating")
plt.show()

# 3. Top 10 destination stations
plt.figure(figsize=(10,5))
df['Destination_Station_Name'].value_counts().head(10).plot(kind='bar')
plt.title("Top 10 Destination Stations")
plt.xlabel("Station Name")
plt.ylabel("Number of Trains Terminating")
plt.show()
```

THANE	186
PANVEL	141
TAMBARAM	140
MOOR MARKET	135
VELACHEERY	115

dtype: int64

◆ Top 10 busiest destination stations:

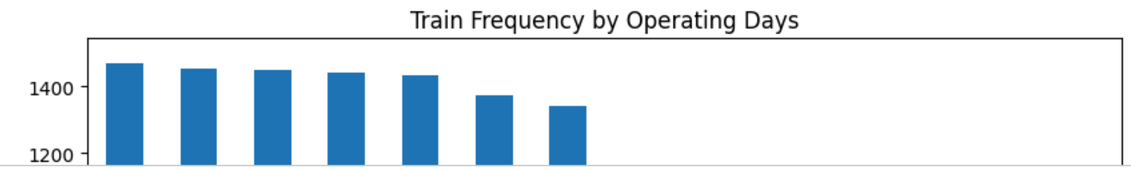
Destination_Station_Name	count
CST-MUMBAI	514
SEALDAH	373
CHENNAI BEACH	342
HOWRAH JN.	337
KALYAN JN	284
THANE	194
PANVEL	144
TAMBARAM	140
MOOR MARKET	132
VELACHEERY	118

dtype: int64

◆ Source vs Destination station imbalance:

	count
RAVLI JN	6.0
AVADI	4.0

```
     TIRUPATI    4.0
      MECHEDA    4.0
    PATNA JN.    4.0
    DOMBIVLI     3.0
  MOOR MARKET    3.0
  BHUBANESWAR    3.0
 BELAPUR C.B.D   3.0
 ARAKKONAM JN    2.0

dtype: float64
```



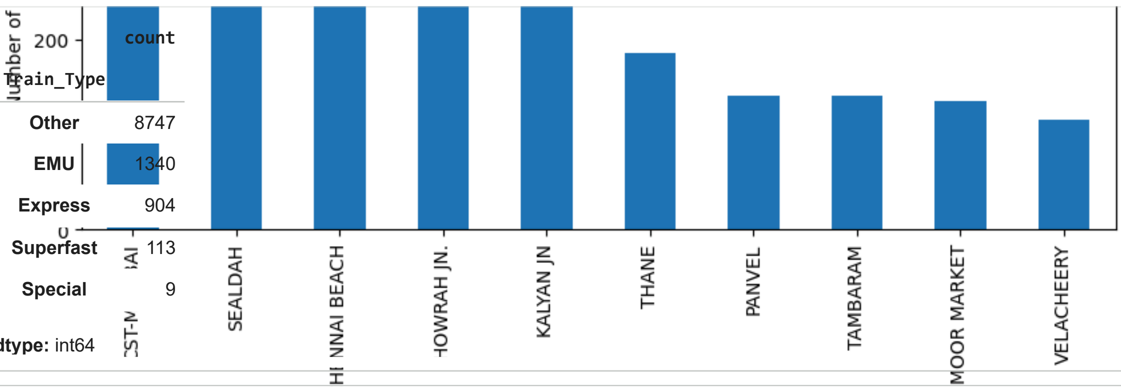
```
# Fix days like "Fridayd" => "Friday (Daily)"
df['days'] = df['days'].apply(
    lambda x: x.replace('d', '') + " (Daily)" if x.endswith('d') else x + " (Weekly)"
)

# Preview the updated values
df['days'].unique()[:15]
```

```
array(['Saturday (Weekly)', 'Friday (Weekly)', 'Wednesday (Weekly)',
       'Tuesday (Weekly)', 'Monday (Weekly)', 'Sunday (Weekly)',
       'Thursday (Weekly)', 'Monay (Daily)', 'Thursay (Daily)',
       'Tuesay (Daily)', 'Friay (Daily)', 'Wenesay (Daily)',
       'Saturay (Daily)', 'Sunay (Daily)'], dtype=object)
```

```
import re

def extract_type(name):
    name = name.upper()
    if "EMU" in name:
        return "EMU"
    elif "EXP" in name or "EXPRESS" in name:
        return "Express"
    elif "SF" in name:
        return "Superfast"
    elif "SPECIAL" in name or "SPEC" in name:
        return "Special"
    else:
        return "Other"

df['Train_Type'] = df['Train_Name'].apply(extract_type)

df['Train_Type'].value_counts()
```

Train_Type	count
Other	8747
EMU	1340
Express	904
Superfast	113
Special	9

dtype: int64



```
df['Route'] = df['Source_Station_Name'] + " → " + df['Destination_Station_Name']
df[['Train_No','Route','days','Train_Type']].head()
```

	Train_No	Route	days	Train_Type
0	107	SAWANTWADI ROAD → MADGOAN JN.	Saturday (Weekly)	Other
1	108	MADGOAN JN. → SAWANTWADI ROAD	Friday (Weekly)	Other
2	128	MADGOAN JN. → CHHATRAPATI SHAHU MAHARAJ TERMINUS	Friday (Weekly)	Special
3	290	DELHI-SAFDAR JANG → DELHI-SAFDAR JANG	Wednesday (Weekly)	Other
4	401	AURANGABAD → VARANASI JN.	Saturday (Weekly)	Other

```
day_fix = {
    "Monay": "Monday", "Tuesay": "Tuesday", "Wenesay": "Wednesday",
    "Thursay": "Thursday", "Friay": "Friday", "Saturay": "Saturday",
    "Sunay": "Sunday"
}

df['days'] = df['days'].apply(lambda x: x.replace("(Daily)", "").replace("(Weekly)", "").strip())
df['days'] = df['days'] + " (Weekly or Daily)"
```

```
import pandas as pd
import matplotlib.pyplot as plt

# Reload dataset (safe practice)
df = pd.read_csv("Railway_info.csv")

# Standardize station names
df['Source_Station_Name'] = df['Source_Station_Name'].str.upper().str.strip()
df['Destination_Station_Name'] = df['Destination_Station_Name'].str.upper().str.strip()
df['Train_Name'] = df['Train_Name'].str.upper().str.strip()
```

```
from IPython.display import display
import matplotlib.pyplot as plt

%matplotlib inline

print("Dataset shape:", df.shape)

display(df.head())
display(df['days'].value_counts())
display(df['Train_Type'].value_counts())

plt.figure(figsize=(8,4))
df['Train_Type'].value_counts().plot(kind='bar')
plt.title("Train Type Distribution")
plt.show()
```

```
Dataset shape: (11113, 5)
```

	Train_No	Train_Name	Source_Station_Name	Destination_Station_Name	days
0	107	SWV-MAO-VLNK	SAWANTWADI ROAD	MADGOAN JN.	Saturday
1	108	VLNK-MAO-SWV	MADGOAN JN.	SAWANTWADI ROAD	Friday
2	128	MAO-KOP SPEC	MADGOAN JN.	CHHATRAPATI SHAHU MAHARAJ TERMINUS	Friday
3	290	PALACE ON WH	DELHI-SAFDAR JANG	DELHI-SAFDAR JANG	Wednesday
4	401	BSB BHARATDA	AURANGABAD	VARANASI JN.	Saturday

	count
days	
Friday	1471
Tuesday	1454
Wednesday	1448
Saturday	1441
Sunday	1432
Thursday	1372
Monday	1342
Fridayd	178
Tuesdayd	174
Sundayd	170
Wednesdayd	164
Mondayd	161
Thursdayd	154
Saturdayd	152

dtype: int64

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/usr/local/lib/python3.12/dist-packages/pandas/core/indexes/base.py in get_loc(self, key)
   3804         try:
-> 3805             return self._engine.get_loc(casted_key)
   3806         except KeyError as err:

index.pyx in pandas._libs.index.IndexEngine.get_loc()

index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'Train_Type'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
                                  2 frames
/usr/local/lib/python3.12/dist-packages/pandas/core/indexes/base.py in get_loc(self, key)
   3810             ):
   3811                 raise InvalidIndexError(key)
-> 3812             raise KeyError(key) from err
   3813         except TypeError:
   3814             # If we have a listlike key, _check_indexing_error will raise

KeyError: 'Train_Type'
```

Next steps: Explain error

```
def extract_type(name):
    name = name.upper()
    if "EMU" in name:
        return "EMU"
    elif "EXP" in name or "EXPRESS" in name:
        return "EXPRESS"
    elif "SF" in name:
        return "SUPERFAST"
    elif "SPEC" in name or "SPECIAL" in name:
        return "SPECIAL"
    else:
        return "OTHER"

df['Train_Type'] = df['Train_Name'].apply(extract_type)
```

```
print(df.columns)
```

```
Index(['Train_No', 'Train_Name', 'Source_Station_Name',
       'Destination_Station_Name', 'days', 'Train_Type'],
      dtype='object')
```

```
if 'Train_Type' in df.columns:
    display(df['Train_Type'].value_counts())
else:
    print("Train_Type column not created yet.")
```

Train_Type	count
OTHER	8747
EMU	1340
EXPRESS	904
SUPERFAST	113
SPECIAL	9

dtype: int64

```
print("Columns available:")
print(df.columns)

display(df[['Train_No','Train_Name','Train_Type']].head())
```

```
Columns available:
Index(['Train_No', 'Train_Name', 'Source_Station_Name',
       'Destination_Station_Name', 'days', 'Train_Type'],
      dtype='object')
```

	Train_No	Train_Name	Train_Type
0	107	SWV-MAO-VLNK	OTHER
1	108	VLNK-MAO-SWV	OTHER
2	128	MAO-KOP SPEC	SPECIAL
3	290	PALACE ON WH	OTHER
4	401	BSB BHARATDA	OTHER

```
import pandas as pd
import matplotlib.pyplot as plt

# Reload dataset (safe practice)
df = pd.read_csv("Railway_info.csv")

# Standardize station names
df['Source_Station_Name'] = df['Source_Station_Name'].str.upper().str.strip()
df['Destination_Station_Name'] = df['Destination_Station_Name'].str.upper().str.strip()
df['Train_Name'] = df['Train_Name'].str.upper().str.strip()
```

```
# Fix day name typos
day_fix = {
    "MONAY": "MONDAY", "TUESAY": "TUESDAY", "WENESAY": "WEDNESDAY",
    "THURSAY": "THURSDAY", "FRIAY": "FRIDAY",
    "SATURAY": "SATURDAY", "SUNAY": "SUNDAY"
}

def clean_days(x):
    x = x.upper()
    if x.endswith('D'):
        day = x[:-1]
        day = day_fix.get(day, day)
        return f"{day} (DAILY)"
    else:
        return f"{x} (WEEKLY)"

df['days'] = df['days'].apply(clean_days)

df['days'].value_counts()
```

```
                        count
              days
   FRIDAY (WEEKLY)       1471
   TUESDAY (WEEKLY)      1454
 WEDNESDAY (WEEKLY)      1448
  SATURDAY (WEEKLY)      1441
   SUNDAY (WEEKLY)       1432
  THURSDAY (WEEKLY)      1372
   MONDAY (WEEKLY)       1342
    FRIDAY (DAILY)        178
   TUESDAY (DAILY)        174
    SUNDAY (DAILY)        170
  WEDNESDAY (DAILY)       164
    MONDAY (DAILY)        161
   THURSDAY (DAILY)       154
   SATURDAY (DAILY)       152

dtype: int64
```

```
def extract_type(name):
    if "EMU" in name:
        return "EMU"
    elif "EXP" in name or "EXPRESS" in name:
        return "EXPRESS"
    elif "SF" in name:
        return "SUPERFAST"
    elif "SPEC" in name or "SPECIAL" in name:
        return "SPECIAL"
    else:
```