Performance Report ETL

This notebook outlines the process of extracting, transforming, and loading (ETL) performance and outage data. The goal is to clean and prepare the data for further analysis and visualization.

Libraries and Setup

We start by importing the necessary libraries for data manipulation and analysis.

In [27]:
```
import pandas as pd
import numpy as np
```

Extract Performance Data

In this section, we extract the performance data from an Excel file. The data is initially previewed to understand its structure and contents.

In [28]:
```
# Extracting performance data from Excel
performance_file_path = 'C:/Users/mayow/OneDrive/My Projects [data analytics]/Telecommunications Site Outages Analysis and Rep
performance_df = pd.read_excel(performance_file_path, sheet_name='2023 - 2024 PA Trends')

# Displaying the first few rows of the dataframe
performance_df.head()
```

Out[28]:

	Site	Anchor Tenant	ATC ID	Anchor Tenant.1	New Region	State	Airtel ID	Airtel Region	New Vendor	Cluster	...	2024-07-17 00:00:00	2024-07-18 00:00:00	2024-07-19 00:00:00	2024-07-20 00:00:00
0	NI0223	NI0223	406684	NI0223	Abuja	Niger	NI0223	NORTH WEST	I-ENG	Minna 1	...	100.0	100.000000	100.0	100.0
1	KG0391	KG0391	408209	KG0391	Abuja	Kogi	KG0391	NORTH WEST	I-ENG	Okene	...	100.0	89.791667	100.0	100.0
2	ZM0008	ZM0008	404767	ZM0008	Kano	Zamfara	ZM0008	NORTH WEST	I-ENG	Gusau	...	100.0	100.000000	100.0	100.0
3	NaN	NI0228	627978	NI0228	Abuja	Niger	NI0228	NORTH WEST	I-ENG	Suleja	...	100.0	100.000000	100.0	100.0
4	KG0084	KG0084	402368	KG0084	Abuja	Kogi	KG0084	NORTH WEST	I-ENG	Okene	...	100.0	100.000000	100.0	100.0

5 rows × 589 columns

Data Cleaning and Transformation for Performance Data

Here we clean and transform the performance data. This involves removing redundant columns, reshaping the data, and ensuring that it is in a usable format for analysis.

Dropping Unnecessary Columns

We drop columns that are not needed for the analysis, focusing on the 'Anchor Tenant' and date-related columns.

In [29]:
```
# Function to check if a column name is a date
def is_date(string):
    try:
        # Attempt to convert the string to a datetime object
        pd.to_datetime(string)
        return True
    except ValueError:
        # If conversion fails, it's not a date
        return False

# Identify columns that represent dates
date_columns = [col for col in performance_df.columns if is_date(col)]

# Specify columns to keep, including 'Anchor Tenant' and all date columns
columns_to_keep = ['Anchor Tenant'] + date_columns

# Subset the DataFrame to include only the specified columns
performance_df = performance_df[columns_to_keep]

# Unpivot the DataFrame from wide to long format, with 'Anchor Tenant' and 'Date' as identifier variables
performance_df = pd.melt(performance_df, id_vars=['Anchor Tenant'], var_name='Date', value_name='Performance')
```

```
# Drop rows where 'Performance' column has NaN values
performance_df = performance_df.dropna(subset=['Performance'])

# Display the last 2 rows of the transformed DataFrame
performance_df.tail(2)
```

Out[29]:

	Anchor Tenant	Date	Performance
1074361	NI4584	2024-07-26 00:00:00	100.0
1074362	NI0710	2024-07-26 00:00:00	100.0

Export Cleaned Performance Data

The cleaned and transformed performance data is saved to a CSV file for further analysis and integration with other datasets.

In [30]:
```
# Saving the cleaned data to a CSV file
performance_cleaned_file_path = 'C:/Users/mayow/OneDrive/My Projects [data analytics]/Telecommunications Site Outages Analysis
performance_df.to_csv(performance_cleaned_file_path, index=False)
```

Extract Outage Data

Similar to the performance data, we extract the outage data from another sheet in the same Excel file. This data will also be cleaned and prepared for analysis.

In [31]:
```
# Extracting outage data from Excel
outage_file_path = 'C:/Users/mayow/OneDrive/My Projects [data analytics]/Telecommunications Site Outages Analysis and Reportin
outage_df = pd.read_excel(outage_file_path, sheet_name='Outage')
outage_df = outage_df.drop(columns=['Sub Category'])

# Displaying the first few rows of the dataframe
outage_df.head(2)
```

Out[31]:

	Issue Number	Alarm Severity	Status	Event Title	Airtel Site ID	ISM	ATC ID	Alarm Start Time	Alarm Cleared Time	DURATION	MTTR	MTTR RANGE	Root Cause	Specific Cause	Underlyi Cau
0	28350667	Closed	Major	SITE DOWN AT KD1070	KD1070	I-ENG	627855	2024-07-01 22:02:00	2024-07-01 22:56:00	54.0	0.900000	<2hrs	Power	DG Fault	DG In st Mo
1	28350336	Closed	Major	SITE DOWN AT KB0040	KB0040	I-ENG	402100	2024-07-01 21:59:00	2024-07-01 22:13:00	14.0	0.233333	<2hrs	Power	Fuel Outage	Oth Plann Wor

Data Formatting and Key Creation

In this section, we format the date columns and create a unique key for each outage entry. This key helps in identifying unique records and prevents data duplication.

Date Formatting

Ensure the date columns are in the correct datetime format for consistent analysis.

In [32]:
```
# Converting date columns to datetime format
outage_df['Alarm Start Time'] = pd.to_datetime(outage_df['Alarm Start Time'])
outage_df['Alarm Cleared Time'] = pd.to_datetime(outage_df['Alarm Cleared Time'])

# Creating a unique key for each entry
outage_df['key'] = outage_df['Issue Number'].astype(str) + '_' + outage_df['Alarm Start Time'].dt.strftime('%Y-%m-%d %H:%M:%S'
outage_df.head(1)
```

Out[32]:

	Issue Number	Alarm Severity	Status	Event Title	Airtel Site ID	ISM	ATC ID	Alarm Start Time	Alarm Cleared Time	DURATION	...	MTTR RANGE	Root Cause	Specific Cause	Underlying Cause	(
0	28350667	Closed	Major	SITE DOWN AT KD1070	KD1070	I-ENG	627855	2024-07-01 22:02:00	2024-07-01 22:56:00	54.0	...	<2hrs	Power	DG Fault	DG In stop Mode	[TC I I SY

1 rows × 21 columns

Comparing New and Existing Data

We compare the newly extracted data with existing cleaned data to identify and append only the unique entries. This approach ensures that our data repository is comprehensive and up-to-date without any data loss.

Reading Existing Cleaned Data and Appending New Data

In [33]:

```
outages_path = 'C:/Users/mayow/OneDrive/My Projects [data analytics]/Telecommunications Site Outages Analysis and Reporting/up
existing_outages = pd.read_csv(outages_path)

existing_keys = set(existing_outages['key'])

new_entries = outage_df[~outage_df['key'].isin(existing_keys)]

# Append new entries to existing outages
updated_outages = pd.concat([existing_outages, new_entries], ignore_index=True)

updated_outages.tail(2)
```

Out[33]:

	Issue Number	Alarm Severity	Status	Event Title	Airtel Site ID	ISM	ATC ID	Cluster	Regional Manager	Alarm Start Time	...	MTTR RANGE	Root Cause	Specific Cause	Underlying Cause	C
798	29025391	Closed	Major	SITE DOWN AT AB0036	AB0036	I-ENG	400026	NaN	NaN	2024-07-26 20:24:00	...	2hrs - 4hrs	Power	DG Fault	Water Pump Fault	FAU SEF
799	29028679	Closed	Major	SITE DOWN AT KB0461	KB0461	I-ENG	627736	NaN	NaN	2024-07-26 23:18:00	...	<2hrs	Power	DG Fault	DG Oil	EN ADI

2 rows × 23 columns

Saving Updated Data

Finally, we save our cleaned files

In [34]:

```
# Save the updated dataset
updated_outages.to_csv(outages_path, index=False)
```