ETL (Extract, Transform, Load) Process

This section outlines the ETL process used to prepare the data for analysis. It involves:

1. Extracting raw datasets from various sources (Sales_January, Sales_February, ... , and Sales_December),

2. Transforming the data by:

- merging sources,
- correcting data types,
- handling missing values,
- removing duplicates to ensure consistency and accuracy,
- changing the data type to optimize memory usage,
- expanding the dataset with supplementary columns,
- organizing oata by Order date chronologically and reindexing,
- and formating float datatype to ensure consistency and accuracy,

3. And finally, Loading the cleaned and structured data into the analysis environment for further exploration and modelling.

Importing libraries

In [7]:
```
import pandas as pd
```

Combining sales data from all months into a single consolidated CSV file.

In [9]:
```
# Defining the folder path
folder_path = r"C:\Monthly_Sales"

file_names = [
    "Sales_January_2024.csv", "Sales_February_2024.csv", "Sales_March_2024.csv", "S
    "Sales_May_2024.csv", "Sales_June_2024.csv", "Sales_July_2024.csv", "Sales_Augu
    "Sales_September_2024.csv", "Sales_October_2024.csv", "Sales_November_2024.csv"
]

# Creating full paths to each file
full_paths = [f"{folder_path}\\{file}" for file in file_names]
```

```
# Reading and combining all files into one DataFrame
joined_data = pd.concat([pd.read_csv(file) for file in full_paths], ignore_index=Tr

# Saving the joined DataFrame into a new CSV file named "joined_data.csv"
output_file = f"{folder_path}\\joined_data.csv"
joined_data.to_csv(output_file, index=False)

print("All files integrated into:", output_file)
```

```
All files integrated into: C:\Monthly_Sales\joined_data.csv
```

Loading the updated DataFrame

In [11]:
```
# Skip Blank Rows if present in the dataset

df = pd.read_csv(r'C:\Monthly_Sales\joined_data.csv', skip_blank_lines=True)
df.head()
```

Out[11]:

	Order ID	Product Name	Units Purchased	Unit Price	Order Date	Delivery Address
0	141234	Dell UltraSharp Monitor	1	410.99	01/10/24 10:27	590 Washington St, New York City, NY 10001
1	141235	AAA Batteries (4-pack)	1	4.99	01/22/24 09:33	208 1st St, San Francisco, CA 94016
2	141236	AA Batteries (4-pack)	1	5.84	01/19/24 19:11	53 Meadow St, Los Angeles, CA 90001
3	141237	USB-C Charging Cable	1	11.95	01/30/24 13:34	800 Dogwood St, Austin, TX 73301
4	141238	Samsung Odyssey Monitor	1	409.99	01/16/24 08:56	746 4th St, San Francisco, CA 94016

In [12]:
```
df.shape
```

Out[12]:
```
(172531, 6)
```

Data Cleaning Process

Thoroughly clean and standardize the data to eliminate errors, ensure consistency, and build a solid foundation for meaningful insights.

Find and remove rows with NaN values

In [15]:
```
df.isna().sum()
```

Out[15]:
```
Order ID            448
Product Name        448
Units Purchased     448
Unit Price          448
Order Date          448
Delivery Address    448
dtype: int64
```

In [16]:
```
# If Nan value is present in Order ID and Unit Purchased, it will be impossible to
# Therefore, drop Nan values in Order ID and Units Purchased.

df.dropna(subset=['Order ID', 'Units Purchased'], inplace=True)
```

In [17]:
```
# Check if Nan value is present

df.isna().sum()
```

Out[17]:
```
Order ID            0
Product Name        0
Units Purchased     0
Unit Price          0
Order Date          0
Delivery Address    0
dtype: int64
```

In [18]:
```
# Further check if any NaN values or blank rows are present

blank_rows_na = df[df.isnull().any(axis=1)]
blank_rows_na
```

Out[18]:

Order ID	Product Name	Units Purchased	Unit Price	Order Date	Delivery Address

Find and remove rows with duplicate values

In [20]:
```
# Find duplicate values

df.duplicated()
```

Out[20]:
```
0         False
1         False
2         False
3         False
4         False
          ...
172526    False
172527    False
172528    False
172529    False
172530    False
Length: 172083, dtype: bool
```

Verifying and correcting incorrect data types in the dataset.

In [22]:
```
# check for data types

df.dtypes
```

Out[22]:
```
Order ID            object
Product Name        object
Units Purchased     object
Unit Price          object
Order Date          object
Delivery Address    object
dtype: object
```

Correcting incorrect data types

In [24]:
```
df['Order Date'] = pd.to_datetime(df['Order Date'], format='%m/%d/%y %H:%M', errors

df['Units Purchased'] = pd. to_numeric(df['Units Purchased'], errors='coerce')

df['Unit Price'] = pd. to_numeric(df['Unit Price'], errors='coerce')
```

In [25]:
```
# Verify the presence of NaN values remaining in the columns as a result of using e

df.isna().sum()
```

Out[25]:
```
Order ID              0
Product Name          0
Units Purchased     303
Unit Price          303
Order Date          303
Delivery Address      0
dtype: int64
```

In [26]:
```
df = df.dropna()
```

Changing the data type to optimize memory usage (Optional)

In [28]:
```
df['Order ID'] = pd.to_numeric(df['Order ID'], downcast='integer')
df['Product Name'] = df['Product Name'].astype('category')
df['Units Purchased'] = df['Units Purchased']. astype('int8')
df['Unit Price'] = pd.to_numeric(df['Unit Price'], downcast='float')
df['Delivery Address'] = df['Delivery Address'].astype('category')
```

Expanding the dataset with supplementary columns

Adding month column

In [31]:
```
df['Month'] = df['Order Date'].dt.month
df
```

Out[31]:

	Order ID	Product Name	Units Purchased	Unit Price	Order Date	Delivery Address	Month
0	141234	Dell UltraSharp Monitor	1	410.989990	2024-01-10 10:27:00	590 Washington St, New York City, NY 10001	1
1	141235	AAA Batteries (4-pack)	1	4.990000	2024-01-22 09:33:00	208 1st St, San Francisco, CA 94016	1
2	141236	AA Batteries (4-pack)	1	5.840000	2024-01-19 19:11:00	53 Meadow St, Los Angeles, CA 90001	1
3	141237	USB-C Charging Cable	1	11.950000	2024-01-30 13:34:00	800 Dogwood St, Austin, TX 73301	1
4	141238	Samsung Odyssey Monitor	1	409.989990	2024-01-16 08:56:00	746 4th St, San Francisco, CA 94016	1
...	...	...	...	...	...	...	...
172526	306090	Bose SoundSport Headphones	1	99.989998	2024-12-11 12:34:00	148 Johnson St, San Francisco, CA 94016	12
172527	306091	Lightning Charging Cable	1	14.950000	2024-12-26 22:39:00	892 7th St, New York City, NY 10001	12
172528	306092	Galaxy buds Headphones	1	120.000000	2024-12-18 12:16:00	565 Ridge St, San Francisco, CA 94016	12
172529	306093	Bose SoundSport Headphones	1	99.989998	2024-12-10 06:24:00	94 Forest St, New York City, NY 10001	12
172530	306094	USB-C Charging Cable	1	11.950000	2024-12-16 18:41:00	137 Main St, Portland, OR 97035	12

171780 rows × 7 columns

In [32]:
```
df['Month Name'] = df['Order Date'].dt.strftime('%B')
df
```

Out[32]:

	Order ID	Product Name	Units Purchased	Unit Price	Order Date	Delivery Address	Month	M
0	141234	Dell UltraSharp Monitor	1	410.989990	2024-01-10 10:27:00	590 Washington St, New York City, NY 10001	1	Ja
1	141235	AAA Batteries (4-pack)	1	4.990000	2024-01-22 09:33:00	208 1st St, San Francisco, CA 94016	1	Ja
2	141236	AA Batteries (4-pack)	1	5.840000	2024-01-19 19:11:00	53 Meadow St, Los Angeles, CA 90001	1	Ja
3	141237	USB-C Charging Cable	1	11.950000	2024-01-30 13:34:00	800 Dogwood St, Austin, TX 73301	1	Ja
4	141238	Samsung Odyssey Monitor	1	409.989990	2024-01-16 08:56:00	746 4th St, San Francisco, CA 94016	1	Ja
...	...	...	...	...	...	...	...	
172526	306090	Bose SoundSport Headphones	1	99.989998	2024-12-11 12:34:00	148 Johnson St, San Francisco, CA 94016	12	Dec
172527	306091	Lightning Charging Cable	1	14.950000	2024-12-26 22:39:00	892 7th St, New York City, NY 10001	12	Dec
172528	306092	Galaxy buds Headphones	1	120.000000	2024-12-18 12:16:00	565 Ridge St, San Francisco, CA 94016	12	Dec
172529	306093	Bose SoundSport Headphones	1	99.989998	2024-12-10 06:24:00	94 Forest St, New York City, NY 10001	12	Dec
172530	306094	USB-C Charging Cable	1	11.950000	2024-12-16 18:41:00	137 Main St, Portland, OR 97035	12	Dec

171780 rows × 8 columns

Adding Year Column

In [34]:
```
df['Order Date'] = pd.to_datetime(df['Order Date'], errors='coerce')

# Now safely extract the year
df['Year'] = df['Order Date'].dt.year
```

Adding week day column

In [36]:
```
df['Day of Week'] = df['Order Date'].dt.strftime('%a')
df
```

Out[36]:

	Order ID	Product Name	Units Purchased	Unit Price	Order Date	Delivery Address	Month	M
0	141234	Dell UltraSharp Monitor	1	410.989990	2024-01-10 10:27:00	590 Washington St, New York City, NY 10001	1	Ja
1	141235	AAA Batteries (4-pack)	1	4.990000	2024-01-22 09:33:00	208 1st St, San Francisco, CA 94016	1	Ja
2	141236	AA Batteries (4-pack)	1	5.840000	2024-01-19 19:11:00	53 Meadow St, Los Angeles, CA 90001	1	Ja
3	141237	USB-C Charging Cable	1	11.950000	2024-01-30 13:34:00	800 Dogwood St, Austin, TX 73301	1	Ja
4	141238	Samsung Odyssey Monitor	1	409.989990	2024-01-16 08:56:00	746 4th St, San Francisco, CA 94016	1	Ja
...	...	...	...	...	...	...	...	
172526	306090	Bose SoundSport Headphones	1	99.989998	2024-12-11 12:34:00	148 Johnson St, San Francisco, CA 94016	12	Dec
172527	306091	Lightning Charging Cable	1	14.950000	2024-12-26 22:39:00	892 7th St, New York City, NY 10001	12	Dec
172528	306092	Galaxy buds Headphones	1	120.000000	2024-12-18 12:16:00	565 Ridge St, San Francisco, CA 94016	12	Dec
172529	306093	Bose SoundSport Headphones	1	99.989998	2024-12-10 06:24:00	94 Forest St, New York City, NY 10001	12	Dec
172530	306094	USB-C Charging Cable	1	11.950000	2024-12-16 18:41:00	137 Main St, Portland, OR 97035	12	Dec

171780 rows × 10 columns

Adding hour column

In [38]:
```
df['Hour'] = df['Order Date'].dt.hour
df
```

Out[38]:

	Order ID	Product Name	Units Purchased	Unit Price	Order Date	Delivery Address	Month	M
0	141234	Dell UltraSharp Monitor	1	410.989990	2024-01-10 10:27:00	590 Washington St, New York City, NY 10001	1	Ja
1	141235	AAA Batteries (4-pack)	1	4.990000	2024-01-22 09:33:00	208 1st St, San Francisco, CA 94016	1	Ja
2	141236	AA Batteries (4-pack)	1	5.840000	2024-01-19 19:11:00	53 Meadow St, Los Angeles, CA 90001	1	Ja
3	141237	USB-C Charging Cable	1	11.950000	2024-01-30 13:34:00	800 Dogwood St, Austin, TX 73301	1	Ja
4	141238	Samsung Odyssey Monitor	1	409.989990	2024-01-16 08:56:00	746 4th St, San Francisco, CA 94016	1	Ja
...	...	...	...	...	...	...	...	
172526	306090	Bose SoundSport Headphones	1	99.989998	2024-12-11 12:34:00	148 Johnson St, San Francisco, CA 94016	12	Dec
172527	306091	Lightning Charging Cable	1	14.950000	2024-12-26 22:39:00	892 7th St, New York City, NY 10001	12	Dec
172528	306092	Galaxy buds Headphones	1	120.000000	2024-12-18 12:16:00	565 Ridge St, San Francisco, CA 94016	12	Dec
172529	306093	Bose SoundSport Headphones	1	99.989998	2024-12-10 06:24:00	94 Forest St, New York City, NY 10001	12	Dec
172530	306094	USB-C Charging Cable	1	11.950000	2024-12-16 18:41:00	137 Main St, Portland, OR 97035	12	Dec

171780 rows × 11 columns

Adding city column

In [40]:
```
def city(address):
    return address.split(",")[1].strip(" ")

def state_abbrev(address):
    return address.split(",")[2].split(" ")[1]

df['City'] = df['Delivery Address'].apply(lambda x: f"{city(x)}  ({state_abbrev(x)}
df.head()
```

Out[40]:

	Order ID	Product Name	Units Purchased	Unit Price	Order Date	Delivery Address	Month	Month Name	Yea
0	141234	Dell UltraSharp Monitor	1	410.98999	2024-01-10 10:27:00	590 Washington St, New York City, NY 10001	1	January	202
1	141235	AAA Batteries (4-pack)	1	4.99000	2024-01-22 09:33:00	208 1st St, San Francisco, CA 94016	1	January	202
2	141236	AA Batteries (4-pack)	1	5.84000	2024-01-19 19:11:00	53 Meadow St, Los Angeles, CA 90001	1	January	202
3	141237	USB-C Charging Cable	1	11.95000	2024-01-30 13:34:00	800 Dogwood St, Austin, TX 73301	1	January	202
4	141238	Samsung Odyssey Monitor	1	409.98999	2024-01-16 08:56:00	746 4th St, San Francisco, CA 94016	1	January	202

Organizing Data by Order Date Chronologically and Reindex

In [42]:
```
df = df.sort_values(by = 'Order Date')
df
```

Out[42]:

	Order ID	Product Name	Units Purchased	Unit Price	Order Date	Delivery Address	Month	Month Nam
19768	160155	Alienware Monitor	1	400.989990	2024-01-01 05:04:00	765 Ridge St, Portland, OR 97035	1	Januar
10247	151041	AAA Batteries (4-pack)	1	4.990000	2024-01-01 05:04:00	964 Lakeview St, Atlanta, GA 30301	1	Januar
5789	146765	AAA Batteries (4-pack)	1	4.990000	2024-01-01 05:20:00	546 10th St, San Francisco, CA 94016	1	Januar
4578	145617	Amana Washing Machine	1	600.000000	2024-01-01 05:24:00	961 Meadow St, Portland, OR 97035	1	Januar
15989	156535	Lightning Charging Cable	2	14.950000	2024-01-01 05:45:00	451 Elm St, Los Angeles, CA 90001	1	Januar
...	...	...	...	...	...	...	...	.
163805	297748	USB-C Charging Cable	2	11.950000	2025-01-01 02:37:00	258 Forest St, Los Angeles, CA 90001	1	Januar
150010	284606	Bose SoundSport Headphones	1	99.989998	2025-01-01 02:50:00	211 Johnson St, Boston, MA 02215	1	Januar
168593	302330	AA Batteries (4-pack)	1	5.840000	2025-01-01 03:03:00	665 6th St, San Francisco, CA 94016	1	Januar
150117	284711	AA Batteries (4-pack)	1	5.840000	2025-01-01 03:19:00	250 8th St, San Francisco, CA 94016	1	Januar

	Order ID	Product Name	Units Purchased	Unit Price	Order Date	Delivery Address	Month	Mont Nam
169966	303626	USB-C Charging Cable	3	11.950000	2025-01-01 04:43:00	651 Lakeview St, Dallas, TX 75001	1	Januar

171780 rows × 12 columns

In [43]:
```
df = df.reset_index(drop=True)
df
```

Out[43]:

	Order ID	Product Name	Units Purchased	Unit Price	Order Date	Delivery Address	Month	Month Nam
0	160155	Alienware Monitor	1	400.989990	2024-01-01 05:04:00	765 Ridge St, Portland, OR 97035	1	Januar
1	151041	AAA Batteries (4-pack)	1	4.990000	2024-01-01 05:04:00	964 Lakeview St, Atlanta, GA 30301	1	Januar
2	146765	AAA Batteries (4-pack)	1	4.990000	2024-01-01 05:20:00	546 10th St, San Francisco, CA 94016	1	Januar
3	145617	Amana Washing Machine	1	600.000000	2024-01-01 05:24:00	961 Meadow St, Portland, OR 97035	1	Januar
4	156535	Lightning Charging Cable	2	14.950000	2024-01-01 05:45:00	451 Elm St, Los Angeles, CA 90001	1	Januar
...	...	...	...	...	...	...	...	.
171775	297748	USB-C Charging Cable	2	11.950000	2025-01-01 02:37:00	258 Forest St, Los Angeles, CA 90001	1	Januar
171776	284606	Bose SoundSport Headphones	1	99.989998	2025-01-01 02:50:00	211 Johnson St, Boston, MA 02215	1	Januar
171777	302330	AA Batteries (4-pack)	1	5.840000	2025-01-01 03:03:00	665 6th St, San Francisco, CA 94016	1	Januar
171778	284711	AA Batteries (4-pack)	1	5.840000	2025-01-01 03:19:00	250 8th St, San Francisco, CA 94016	1	Januar

	Order ID	Product Name	Units Purchased	Unit Price	Order Date	Delivery Address	Month	Mont Nam
171779	303626	USB-C Charging Cable	3	11.950000	2025-01-01 04:43:00	651 Lakeview St, Dallas, TX 75001	1	Januar

171780 rows × 12 columns

Adding Total Sales column

In [45]:
```
df['Total Sales'] = df['Units Purchased'] * df['Unit Price']
df.head()
```

Out[45]:

	Order ID	Product Name	Units Purchased	Unit Price	Order Date	Delivery Address	Month	Month Name	Year
0	160155	Alienware Monitor	1	400.98999	2024-01-01 05:04:00	765 Ridge St, Portland, OR 97035	1	January	2024
1	151041	AAA Batteries (4-pack)	1	4.99000	2024-01-01 05:04:00	964 Lakeview St, Atlanta, GA 30301	1	January	2024
2	146765	AAA Batteries (4-pack)	1	4.99000	2024-01-01 05:20:00	546 10th St, San Francisco, CA 94016	1	January	2024
3	145617	Amana Washing Machine	1	600.00000	2024-01-01 05:24:00	961 Meadow St, Portland, OR 97035	1	January	2024
4	156535	Lightning Charging Cable	2	14.95000	2024-01-01 05:45:00	451 Elm St, Los Angeles, CA 90001	1	January	2024

Formatting Unit Price and Total Sales to two decimal places.

In [47]:
```
df['Unit Price'] = df['Unit Price'].apply(lambda x: "%.2f" % x)
```

In [48]:
```
df['Total Sales'] = df['Total Sales'].apply(lambda x: "%.2f" % x)
df.head()
```

Out[48]:

	Order ID	Product Name	Units Purchased	Unit Price	Order Date	Delivery Address	Month	Month Name	Year	D We
0	160155	Alienware Monitor	1	400.99	2024-01-01 05:04:00	765 Ridge St, Portland, OR 97035	1	January	2024	M
1	151041	AAA Batteries (4-pack)	1	4.99	2024-01-01 05:04:00	964 Lakeview St, Atlanta, GA 30301	1	January	2024	M
2	146765	AAA Batteries (4-pack)	1	4.99	2024-01-01 05:20:00	546 10th St, San Francisco, CA 94016	1	January	2024	M
3	145617	Amana Washing Machine	1	600.00	2024-01-01 05:24:00	961 Meadow St, Portland, OR 97035	1	January	2024	M
4	156535	Lightning Charging Cable	2	14.95	2024-01-01 05:45:00	451 Elm St, Los Angeles, CA 90001	1	January	2024	M

Converting to numeric

In [50]:
```
df['Unit Price'] = pd.to_numeric(df['Unit Price'])
df['Total Sales'] = pd.to_numeric(df['Total Sales'])
```

In [51]:
```
df.dtypes
```

Out[51]:
```
Order ID                    int32
Product Name             category
Units Purchased             int8
Unit Price               float64
Order Date         datetime64[ns]
Delivery Address         category
Month                       int32
Month Name                 object
Year                        int32
Day of Week                object
Hour                        int32
City                       object
Total Sales              float64
dtype: object
```

In [52]:
```
df.head()
```

Out[52]:

	Order ID	Product Name	Units Purchased	Unit Price	Order Date	Delivery Address	Month	Month Name	Year	D We
0	160155	Alienware Monitor	1	400.99	2024-01-01 05:04:00	765 Ridge St, Portland, OR 97035	1	January	2024	M
1	151041	AAA Batteries (4-pack)	1	4.99	2024-01-01 05:04:00	964 Lakeview St, Atlanta, GA 30301	1	January	2024	M
2	146765	AAA Batteries (4-pack)	1	4.99	2024-01-01 05:20:00	546 10th St, San Francisco, CA 94016	1	January	2024	M
3	145617	Amana Washing Machine	1	600.00	2024-01-01 05:24:00	961 Meadow St, Portland, OR 97035	1	January	2024	M
4	156535	Lightning Charging Cable	2	14.95	2024-01-01 05:45:00	451 Elm St, Los Angeles, CA 90001	1	January	2024	M

In [53]:
```
df.describe()
```

Out[53]:

	Order ID	Units Purchased	Unit Price	Order Date	Month	
count	171780.000000	171780.000000	171780.000000	171780	171780.000000	171
mean	223706.282122	1.072674	228.039622	2024-07-22 15:29:26.871463424	7.183147	2
min	141234.000000	1.000000	4.990000	2024-01-01 05:04:00	1.000000	2
25%	182483.750000	1.000000	11.950000	2024-04-19 09:03:45	4.000000	2
50%	223736.500000	1.000000	99.990000	2024-07-27 19:17:30	7.000000	2
75%	264937.250000	1.000000	399.990000	2024-11-12 14:04:45	11.000000	2
max	306094.000000	7.000000	1700.000000	2025-01-01 04:43:00	12.000000	2
std	47585.469217	0.308029	367.043540	NaN	3.735458	

In [54]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 171780 entries, 0 to 171779
Data columns (total 13 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   Order ID          171780 non-null  int32
 1   Product Name      171780 non-null  category
 2   Units Purchased   171780 non-null  int8
 3   Unit Price        171780 non-null  float64
 4   Order Date        171780 non-null  datetime64[ns]
 5   Delivery Address  171780 non-null  category
 6   Month             171780 non-null  int32
 7   Month Name        171780 non-null  object
 8   Year              171780 non-null  int32
 9   Day of Week       171780 non-null  object
 10  Hour              171780 non-null  int32
 11  City              171780 non-null  object
 12  Total Sales       171780 non-null  float64
dtypes: category(2), datetime64[ns](1), float64(2), int32(4), int8(1), object(3)
memory usage: 16.5+ MB
```

Exporting Cleaned Data to CSV

In [56]:
```
df.to_csv("C:\\Monthly_Sales\\cleaned_data.csv", index=False)
```