Blinkit Retail Sales EDA project

Data Ingestion and Database Setup:

- Utilized SQLAlchemy to ingest CSV files into a SQLite database (Blinkit.db).
- Loaded data into tables using a custom ingest_db() function.
- Verified loaded tables and their columns using PRAGMA table_info.

Data Cleaning:

- Item Fat Content Standardization: Replaced inconsistent categories (LF, low fat, reg) with standardized labels (Low Fat, Regular).
- Item Visibility Adjustment: Replaced zero values (possibly indicating missing data or errors) with the mean visibility of non-zero records.
- Checked Dataset Health:
    - Inspected column data types, missing values, and zero values (specifically in Item Weight).
    - Summarized descriptive statistics and identified unique values in categorical columns.

Data Visualization and Insights:

Sales Distribution

- Wide Sales Range with clusters between 30–250 units.
- Multiple Peaks suggest different sales performance tiers.
- Right-Skewed distribution: a few items have very high sales.
- Business Insight: Different products might target diverse customer segments or promotional strategies.

Item Visibility Distribution

- Right-skewed: Most items have low visibility in the store.
- High Frequency at Low Visibility: Many products are placed in less prominent locations.
- Business Insight: Potential to optimize shelf space and product placement for increased visibility and sales.

Sales by Item Type (Boxplot)

- Wide Range Across Categories, with similar medians.
- Outliers in Health and Hygiene and Fruits and Vegetables indicate category-specific performance extremes.
- Business Insight: Balanced demand across item types, but scope for category-specific promotions or layout strategies.

Correlation Heatmap

- Weak correlations across numeric variables.
- Sales vs. Item Visibility: Slight negative correlation (-0.0013).
- Sales vs. Item Weight: Slight positive correlation (0.027).
- Sales vs. Outlet Establishment Year: Near-zero correlation.
- Sales vs. Rating: Weak positive correlation (0.011).

Code:

```
import pandas as pd
import os
from sqlalchemy import create_engine
import logging
import time

logging.basicConfig(
    level=logging.INFO,  # Set the logging level (DEBUG, INFO, WARNING, ERROR, CRITICAL)
    format='%(asctime)s - %(levelname)s - %(message)s',  # Log message format
    datefmt='%Y-%m-%d %H:%M:%S',  # Date/time format
    filename='app.log',  # Output log file (optional)
    filemode='a'  # Write mode: 'w' for overwrite, 'a' for append
)
engine=create_engine('sqlite:///Blinkit.db')

def ingest_db(df,table_name,engine):
    df.to_sql(table_name,con=engine,if_exists='replace',index=False)

def load_row_data():
    start=time.time()
    path = "data"
    for file in os.listdir(path):
        if '.csv' in file:
            df=pd.read_csv('data/'+file)
            logging.info(f'Ingesting {file} in db')
            ingest_db(df,file[:-4],engine)
    end=time.time()
    total_time=(end-start)/100
    logging.info('Ingestion Complete')
    logging.info(f'\n Total time taken :{total_time} minutes')
if __name__=='__main__':
    load_row_data()
```

- **Database & Table Name(columns)**

```
from sqlalchemy import create_engine, text
engine = create_engine('sqlite:///Blinkit.db')

with engine.connect() as conn:
    # List all tables
    result = conn.execute(text("SELECT name FROM sqlite_master WHERE type='table';"))
    tables = [row[0] for row in result.fetchall()]
    print(f"Tables: {tables}")

    # For each table, get column info
    for table in tables:
        print(f"\nColumns in '{table}':")
        result = conn.execute(text(f"PRAGMA table_info({table});"))
        columns = [row[1] for row in result.fetchall()]
        print(columns)
```

```
Tables: ['BKT']

Columns in 'BKT':
['Item Fat Content', 'Item Identifier', 'Item Type', 'Outlet Establishment Year', 'Outlet Identifier', 'Outlet Location Type', 'Outlet Size', 'Outlet Typ
e', 'Item Visibility', 'Item Weight', 'Sales', 'Rating']
```

- **Showing Loaded Data**

```
from sqlalchemy import create_engine, text
import pandas as pd
engine = create_engine('sqlite:///Blinkit.db')
with engine.connect() as conn:
    # connection is open here
    df = pd.read_sql_query("""select * from BKT""", conn)
# connection is closed here
df.head()
```

	Item Fat Content	Item Identifier	Item Type	Outlet Establishment Year	Outlet Identifier	Outlet Location Type	Outlet Size	Outlet Type	Item Visibility	Item Weight	Sales	Rating
0	Regular	FDX32	Fruits and Vegetables	2012	OUT049	Tier 1	Medium	Supermarket Type1	0.100014	15.10	145.4786	5.0
1	Low Fat	NCB42	Health and Hygiene	2022	OUT018	Tier 3	Medium	Supermarket Type2	0.008596	11.80	115.3492	5.0
2	Regular	FDR28	Frozen Foods	2016	OUT046	Tier 1	Small	Supermarket Type1	0.025896	13.85	165.0210	5.0
3	Regular	FDL50	Canned	2014	OUT013	Tier 3	High	Supermarket Type1	0.042278	12.15	126.5046	5.0

- **Data Cleaning**

#... I noticed some irregular values in the Item Fat Content column, such as 'Low Fat', 'Regular', 'LF', and 'reg'. However, this column should contain only two categories: 'Low Fat' and 'Regular'. Therefore, we need to replace 'LF' with 'Low Fat' and 'reg' with 'Regular'.

Below is the code for above data irregularity:

```
df['Item Fat Content'] = df['Item Fat Content'].replace({
    'LF': 'Low Fat',
    'low fat': 'Low Fat',
    'Regular': 'Regular',
    'reg': 'Regular'
})
```

#... There is "Item Visibility" (Team Visibility likely refers to how prominently an item is displayed to customers) column has zero values it can have some reasons:

1. Missing Data Representation: Zero might indicate missing or unrecorded visibility information.
2. Data Entry Error: Zero visibility is unrealistic in real-world scenarios (items are always visible to some extent if they're in the store)

I have replaced these data with mean of visibility below is the code for the data cleaning

```
mean_visibility = df[df['Item Visibility'] != 0]['Item Visibility'].mean()
df['Item Visibility'] = df['Item Visibility'].replace(0, mean_visibility)
```

- **Checking Clean data**

 # 1 info of table data:

```
print("\n Data Info")
print(df.info())
```

 # 2 Shape of table data :

```
print(f"\n Shape of Data: {df.shape}")
```

 # 3 Columns Names :

```
print(f"\n Columns: {df.columns.tolist()}")
```

 # 4 Checking Missing Values:

```
print("\n Missing Values")
print(df.isnull().sum())
```

 # 5 Summary Statistics

```
print("\n Summary Statistics")
print(df.describe(include='all'))
```

 # 6 Unique Values for Categorical Columns

```
categorical_cols = ['Item Fat Content', 'Item Identifier', 'Item Type', 'Outlet Identifier', 'Outlet Location Type', 'Outlet Size', 'Outlet Type', 'Rating']
```

```
print("\n Unique Values in Categorical Columns")
for col in categorical_cols:
    print(f"\n{col}: {df[col].nunique()} unique values")
    print(df[col].value_counts())
```

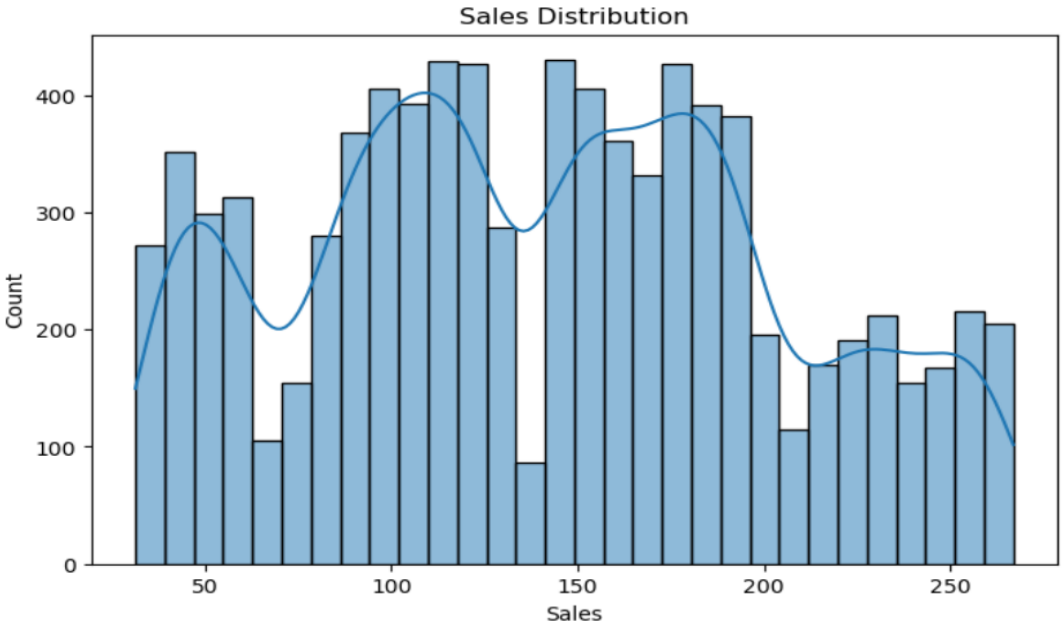
 # 7 Check Zero Values in Item Weight

```
zero_count = (df['Item Weight'] == 0).sum()
print(f"\n Zero values in Item Weight: {zero_count}")
```

- # Visualizations:

Sales Distribution:

```
plt.figure(figsize=(8,5))
sns.histplot(df['Sales'], kde=True, bins=30)
plt.title('Sales Distribution')
plt.show()
```



Insights from Sales Distribution

1. **Sales Range:** Sales values are spread over a wide range, with most data points clustered between approximately 30 and 250 units.

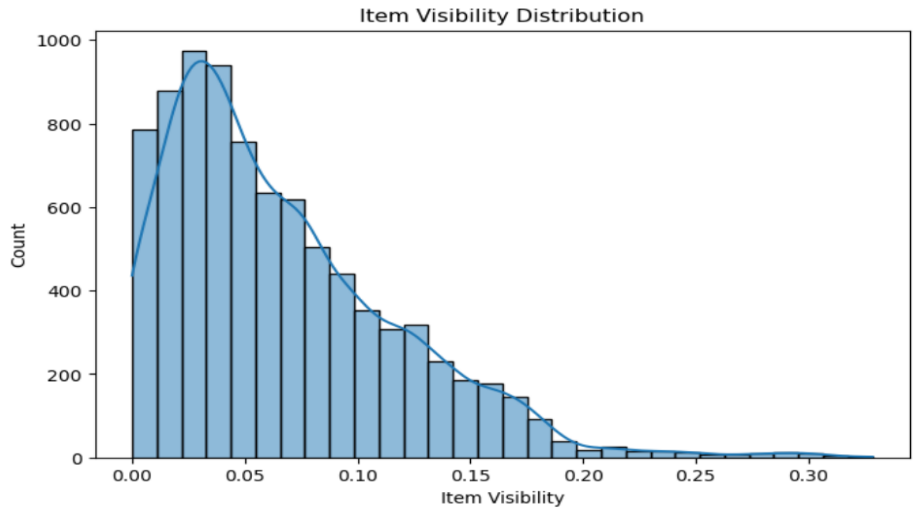
2. **Multiple Peaks**: The distribution is multi-modal (multiple peaks), suggesting several sales performance tiers. This might reflect different types of items or store-level effects.

3. **Density Trends:** High-frequency sales occur around the ranges of 50–100, 100–150, and 150–200, as shown by the density lines overlaying the histograms.

4. **Distribution Shape**: The distribution is not perfectly symmetrical and may exhibit right-skewness (tail towards higher sales), indicating a small number of items with very high sales values.

Item Visibility Distribution:

```
plt.figure(figsize=(8,5))
sns.histplot(df['Item Visibility'], kde=True, bins=30)
plt.title('Item Visibility Distribution')
plt.show()
```



1. Right-Skewed Distribution: The Item Visibility values are heavily skewed towards the left, meaning that most items have low visibility in the store.

2. High Frequency at Low Visibility: The highest count of items corresponds to visibility values close to 0, suggesting that many products are placed in less prominent or less visible locations.

3. Long Tail of Higher Visibility: While most items are in the low visibility range (0.00–0.05), there are fewer items with higher visibility (0.10–0.30).

4. Zero Values (Imputation Handled): The zero visibility values in the original data were likely imputed or replaced with the mean visibility. These zero values were likely due to missing data or data entry errors, as we previously noted.

5. Business Interpretation: This skewed distribution might indicate:

I. Majority of items placed on less prominent shelves.

II. Marketing focus may be on specific high-visibility products.

III. Opportunities to optimize store layout or product placement to increase sales.

Sales by Item Type:

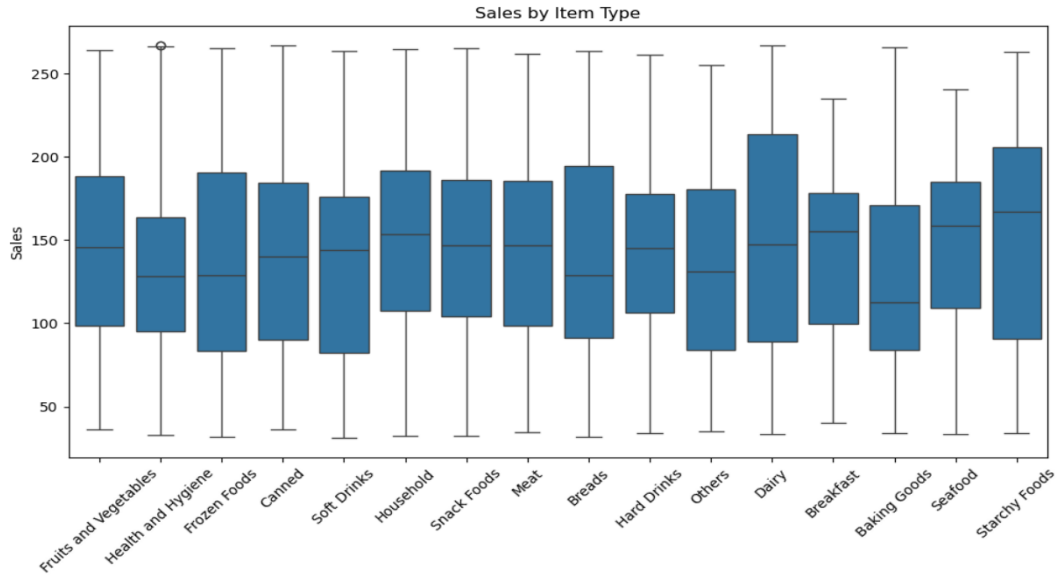
```
plt.figure(figsize=(12,6))

sns.boxplot(x='Item Type', y='Sales', data=df)

plt.xticks(rotation=45)

plt.title('Sales by Item Type')

plt.show()
```



1. **Wide Range of Sales Across Categories**: Each item type shows a wide range of sales, from low (near 0) to high (around 250), with varying distributions.

2. **Similar Medians:** The median sales values for most item types appear to be fairly similar, suggesting that, on average, item sales are comparable across categories.

3. **Outliers in Certain Categories:** Some item types (e.g., Health and Hygiene, Fruits and Vegetables) show outliers, indicating unusually high or low sales values.

4. **Interquartile Ranges (IQR):** The IQRs (spread between the 25th and 75th percentiles) vary across categories:

   - Snack Foods, Dairy, and Others show wider IQRs, indicating greater variability in sales.
   - Health and Hygiene and Canned have narrower IQRs, suggesting more consistent sales.

5. **Overall Distribution:** Despite the variability, no category clearly dominates in terms of sales. This pattern could imply:

   - Balanced customer demand across categories.
   - Potential category-specific promotions or shelf placement strategies might be needed to boost sales further.

Correlation Heatmap for Numeric Columns:

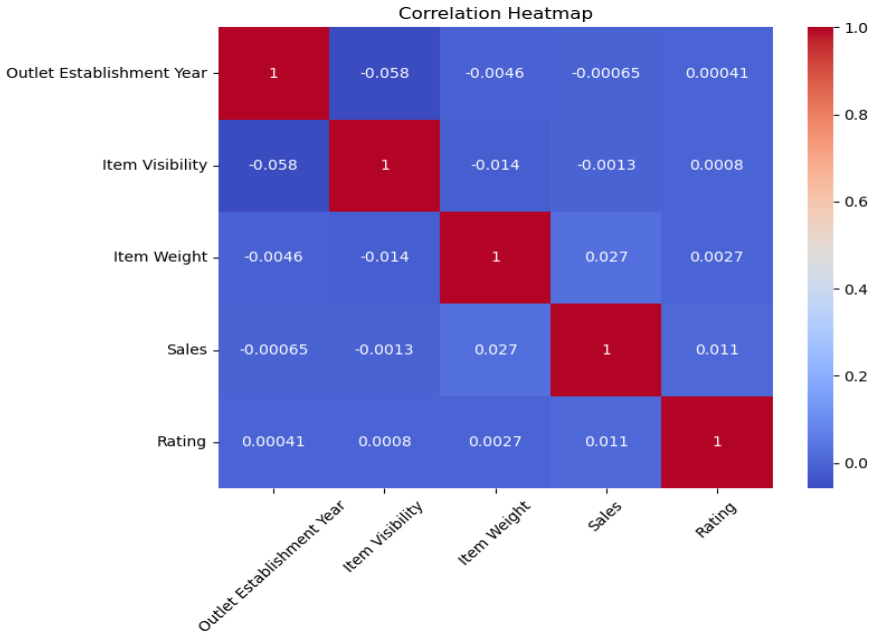
```
plt.figure(figsize=(8,6))

corr = df.corr(numeric_only=True)

sns.heatmap(corr, annot=True, cmap='coolwarm')

plt.title('Correlation Heatmap')

plt.show()
```



1. **Low Correlations Across Features:** Most variables exhibit low correlations (close to zero), suggesting:

    - Weak linear relationships between features.
    - Limited direct influence of one variable on another.

2. **Notable Observations**:
    - Item Visibility vs. Sales: Slight negative correlation (-0.0013), but negligible. This suggests item visibility doesn't strongly affect sales.
    - Item Weight vs. Sales: Slight positive correlation (0.027), also very weak.
    - Outlet Establishment Year vs. Sales: Near-zero correlation (-0.00065), indicating no meaningful relationship between the age of the outlet and sales.
    - Rating vs. Sales: Weak positive correlation (0.011), suggesting ratings might have a minimal effect.

3. **Color Gradient Interpretation**: The heatmap's colors (from red to blue) indicate the strength of correlation:

    - Red: High positive correlation (+1)
    - Blue: Low or negative correlation (-1)
    - The majority of cells are blue, reinforcing the low correlation among features.

4. Overall Insight:

- The selected features (Outlet Establishment Year, Item Visibility, Item Weight, Sales, Rating) are largely independent.
- Sales patterns are likely influenced by other variables, such as promotional efforts, store layout, or customer preferences, which are not captured in this chart.