Sales Insights and Analysis Project

Using Pandas to Drive Data-Driven Business Decisions

Presented By: Vaibhav Patel

Introduction

To analyze sales data from multiple tables and generate actionable insights.

Data Sources:

- transactions: Details on each sale transaction
- customers: Customer profiles and types
- products: Product categories and codes
- markets: Market regions and zones
- date: Calendar information

Key Areas:

1. Data cleaning and preparation

2.Exploratory data analysis (EDA)

3 .Insight generation with advanced analysis techniques

Data Import and Initial Setup

- Imported Pandas and Numpy libraries.
- Loaded sales_data.csv
- Previewed the data structure.

Load the CSV Data

- Reading CSV FILE

```
import pandas as pd
```

```
transactions=pd.read_csv('transactions.csv')
products=pd.read_csv('products.csv')
markets=pd.read_csv('markets.csv')
date=pd.read_csv('date.csv')
customers=pd.read_csv('customers.csv')
```

```
print(transactions)
print(customers)
print(markets)
print(date)
print(products)
```

- Transactions

```
transactions.head(5)
```

	product_code	customer_code	market_code	order_date	sales_qty	sales_amount	currency	profit_margin_percentage	profit_margin
0	Prod279	Cus020	Mark011	2017-10-18	1	102	INR	0	12
1	Prod279	Cus020	Mark011	2017-10-19	1	102	INR	0	29
2	Prod279	Cus020	Mark011	2017-11-08	1	102	INR	0	36
3	Prod279	Cus020	Mark011	2018-03-09	1	102	INR	0	35
4	Prod279	Cus020	Mark011	2018-03-20	1	102	INR	0	26

- Customers

```
customers.head(5)
```

	customer_code	custmer_name	customer_type	customer_code.1	customers	customers.1	customers.2	customers.3
0	Cus002	Nomad Stores	Brick & Mortar	NaN	NaN	NaN	NaN	NaN
1	Cus003	Excel Stores	Brick & Mortar	NaN	NaN	NaN	NaN	NaN
2	Cus004	Surface Stores	Brick & Mortar	NaN	NaN	NaN	NaN	NaN
3	Cus005	Premium Stores	Brick & Mortar	NaN	NaN	NaN	NaN	NaN
4	Cus006	Electricalsara Stores	Brick & Mortar	NaN	NaN	NaN	NaN	NaN

- Date

```
]: date.head(5)
```

]:

	date	cy_date	year	month_name	date_yy_mmm	date.1	date.2	date.3	date.4	date.5
0	2017-06-02	2017-06-01	2017	June	17-Jun\r	NaN	NaN	NaN	NaN	NaN
1	2017-06-03	2017-06-01	2017	June	17-Jun\r	NaN	NaN	NaN	NaN	NaN
2	2017-06-04	2017-06-01	2017	June	17-Jun\r	NaN	NaN	NaN	NaN	NaN
3	2017-06-05	2017-06-01	2017	June	17-Jun\r	NaN	NaN	NaN	NaN	NaN
4	2017-06-06	2017-06-01	2017	June	17-Jun\r	NaN	NaN	NaN	NaN	NaN

- Markets

```
markets.head(5)
```

	markets_code	markets_name	zone	markets_code.1	markets	markets.1	markets.2	markets.3
0	Mark002	Mumbai	Central	NaN	NaN	NaN	NaN	NaN
1	Mark003	Ahmedabad	North	NaN	NaN	NaN	NaN	NaN
2	Mark004	Delhi NCR	North	NaN	NaN	NaN	NaN	NaN
3	Mark005	Kanpur	North	NaN	NaN	NaN	NaN	NaN
4	Mark006	Bengaluru	South	NaN	NaN	NaN	NaN	NaN

- Products

```
products.head(5)
```

	product_code	product_type	product_code.1	products	products.1	products.2	products.3
0	Prod002	Own Brand\r	NaN	NaN	NaN	NaN	NaN
1	Prod003	Own Brand\r	NaN	NaN	NaN	NaN	NaN
2	Prod004	Own Brand\r	NaN	NaN	NaN	NaN	NaN
3	Prod005	Own Brand\r	NaN	NaN	NaN	NaN	NaN
4	Prod006	Own Brand\r	NaN	NaN	NaN	NaN	NaN

Data Cleaning and Preparation

- Dropped unnecessary columns (e.g., columns with suffixes) to reduce clutter.

- Removed rows with missing values in critical columns to maintain data integrity

Outcome: Clean and concise datasets ready for analysis

Check for Missing Values

- Used isnull().sum() to find the total missing values in each column.
- Highlights columns with missing values, guiding decisions on handling them.

```
products.isnull().sum()
```

```
product_code          0
product_type          0
product_code.1      277
products            277
products.1          277
products.2          277
products.3          277
dtype: int64
```

```
transactions.isnull().sum()
```

```
product_code                0
customer_code               0
market_code                 0
order_date                  0
sales_qty                   0
sales_amount                0
currency                    0
profit_margin_percentage    0
profit_margin               0
cost_price                  0
```

```
customers.isnull().sum()
```

```
customer_code         0
custmer_name          0
customer_type         0
customer_code.1      36
customers            36
customers.1          36
customers.2          36
customers.3          36
dtype: int64
```

Check for Missing Values

- Ensures the dataset is complete and ready for accurate analysis

```
date.isnull().sum()
```

```
date              0
cy_date           0
year              0
month_name        0
date_yy_mmm       0
date.1         1124
date.2         1124
date.3         1124
date.4         1124
date.5         1124
dtype: int64
```

```
markets.isnull().sum()
```

```
markets_code       0
markets_name       0
zone               1
markets_code.1    15
markets           15
markets.1         15
markets.2         15
markets.3         15
dtype: int64
```

Handle Missing Values:

- Ensure high data quality by removing incomplete rows, especially in key columns

- Used dropna() in key columns (e.g., product_code, customer_code, sales_amount) to remove rows with missing data.

Dropping Null Values in Customers Table

- Ensure data integrity by removing incomplete entries in the customers table.
- Cleaned customers table, ensuring accurate customer-related insights.

```
print(customers.columns)
```

```
Index(['customer_code', 'custmer_name', 'customer_type', 'customer_code.1',
       'customers', 'customers.1', 'customers.2', 'customers.3'],
      dtype='object')
```

```
# Drop extra columns in 'customers' DataFrame
customers = customers.drop(columns=['customer_code.1', 'customers', 'customers.1', 'customers.2', 'customers.3'])
```

```
# Verify remaining columns
print(customers.columns)
```

```
Index(['customer_code', 'custmer_name', 'customer_type'], dtype='object')
```

Dropping Null Values in Products Table

- Remove rows with missing data in the products table to improve data reliability.

- Ensures only complete product data remains, ready for analysis.

```
print(products.columns)
```

```
Index(['product_code', 'product_type', 'product_code.1', 'products',
       'products.1', 'products.2', 'products.3'],
      dtype='object')
```

```
# Drop extra columns in 'products' DataFrame
products = products.drop(columns=['product_code.1', 'products', 'products.1', 'products.2', 'products.3'])
```

```
print(products.columns)
```

```
Index(['product_code', 'product_type'], dtype='object')
```

Dropping Null Values in Markets Table

- Eliminate incomplete rows in the markets table to maintain accurate regional insights.
- Cleaned markets table, enhancing analysis accuracy by retaining only valid entries.

```
print(markets.columns)
```

```
Index(['markets_code', 'markets_name', 'zone', 'markets_code.1', 'markets',
       'markets.1', 'markets.2', 'markets.3'],
      dtype='object')
```

```
markets = markets.drop(columns=[ 'markets_code.1', 'markets',
       'markets.1', 'markets.2', 'markets.3'])
```

```
print(markets.columns)
```

```
Index(['markets_code', 'markets_name', 'zone'], dtype='object')
```

Dropping Null Values in Date Table

- Remove entries with missing dates for accurate time-based analysis.

- Ensures completeness in the date table, critical for reliable time-based metrics and trends.

```
7]: print(date.columns)

Index(['date', 'cy_date', 'year', 'month_name', 'date_yy_mmm', 'date.1',
       'date.2', 'date.3', 'date.4', 'date.5'],
      dtype='object')

3]: date = date.drop(columns=[ 'date.1',
       'date.2', 'date.3', 'date.4', 'date.5' ])

9]: print(date.columns)

Index(['date', 'cy_date', 'year', 'month_name', 'date_yy_mmm'], dtype='object')
```

Data-Driven Sales Insights Using Pandas

- Sales Trends: Sales distribution across product types, customers, and regions.
- Customer Segmentation: Top customer insights and purchase behavior analysis
- Product Metrics: Popularity, profitability, and performance trends.
- Temporal Analysis: Monthly and yearly performance tracking.
- Profit Analysis: Profit margins by market, product type, and region.

Total Sales by Product Type

- Understand the distribution of sales by different product types.

- **Insight**: Identifies top-performing product categories, guiding product strategy.

```
sales_by_product_type = transactions.merge(products, on='product_code') \
    .groupby('product_type')['sales_amount'].sum().reset_index()
print(sales_by_product_type)
```

```
     product_type  sales_amount
0  Distribution\r     145259554
1     Own Brand\r     369728277
```

Sales Quantity by Customer Type

- Measure sales volume by customer type to understand customer segments.
- **Insight**: Reveals which customer segments contribute most to sales, informing targeted marketing efforts.

```
sales_qty_by_customer_type = transactions.merge(customers, on='customer_code') \
    .groupby('customer_type')['sales_qty'].sum().reset_index()
print(sales_qty_by_customer_type)
```

```
    customer_type  sales_qty
0  Brick & Mortar    1677681
1      E-Commerce     549020
```

Profit Margin Analysis by Market Zone

- Evaluate profitability by region to prioritize zones with high potential.
- **Insight**: Helps in resource allocation and regional strategy by identifying high-margin zones.

```
]: profit_by_zone = transactions.merge(markets, left_on='market_code', right_on='markets_code') \
       .groupby('zone')['profit_margin'].sum().reset_index()
   print(profit_by_zone)
```

```
      zone  profit_margin
0  Central        1237386
1    North        1163034
2    South         121779
```

Monthly Sales Trends

- Track monthly sales to identify seasonal trends.
- **Insight**: Highlights peak and off-peak months, assisting in inventory and marketing planning.

```
monthly_sales = transactions.merge(date, left_on='order_date', right_on='date') \
    .groupby('month_name')['sales_amount'].sum().reset_index()
print(monthly_sales)
```

```
   month_name  sales_amount
0       April      88800967
1      August      71656522
2    December      84758732
3    February      89313340
4     January      99705087
5        July      71086419
6        June      74758578
7       March      92747572
8         May      83453779
9    November      92885925
10    October      80503337
11  September      54967624
```

Average Sales per Product in Each Market

- Determine average sales for products across different markets.

- **Insight**: Identifies products with high demand in specific markets, guiding localized product strategies.

```
avg_sales_per_product_market = transactions.merge(products, on='product_code') \
    .merge(markets, left_on='market_code', right_on='markets_code') \
    .groupby(['product_type', 'markets_name'])['sales_amount'].mean().reset_index()
print(avg_sales_per_product_market)
```

```
      product_type  markets_name  sales_amount
0   Distribution\r     Ahmedabad   3504.512692
1   Distribution\r     Bengaluru  42634.285714
2   Distribution\r        Bhopal   4666.758710
3   Distribution\r  Bhubaneshwar   6334.690476
4   Distribution\r     Delhi NCR   7808.621100
5   Distribution\r     Hyderabad   2582.326708
6   Distribution\r        Kanpur   2665.159715
7   Distribution\r         Kochi   3515.908764
8   Distribution\r       Lucknow   6017.833333
9   Distribution\r        Mumbai   5745.010949
10  Distribution\r        Nagpur    501.973782
11  Distribution\r         Patna   6276.206897
12  Distribution\r         Surat   7330.032258
13     Own Brand\r     Ahmedabad   6875.927697
14     Own Brand\r     Bengaluru   6260.000000
15     Own Brand\r        Bhopal   3038.426903
16     Own Brand\r  Bhubaneshwar   9362.597015
17     Own Brand\r     Delhi NCR   9440.226471
18     Own Brand\r     Hyderabad   4365.985330
19     Own Brand\r        Kanpur   4441.912716
20     Own Brand\r         Kochi   4811.933753
21     Own Brand\r       Lucknow  20438.733333
22     Own Brand\r        Mumbai  15989.813316
23     Own Brand\r        Nagpur   2262.376326
24     Own Brand\r         Patna   6489.495495
25     Own Brand\r         Surat   4372.650704
```

Yearly Sales Growth

- Examine annual sales growth to measure performance over time.
- **Insight:** Tracks growth, indicating business expansion or contraction year-over-year.

```
[]: yearly_sales = transactions.merge(date, left_on='order_date', right_on='date') \
        .groupby('year')['sales_amount'].sum().reset_index()
    print(yearly_sales)
```

```
   year  sales_amount
0  2017      92882236
1  2018     413667773
2  2019     335875806
3  2020     142212067
```

High Revenue Products

- Identify top products by revenue.
- **Insight**: Provides focus on high-revenue products, optimizing product line strategy.

```
top_revenue_products = transactions.merge(products, on='product_code')
    .groupby('product_type')['sales_amount'].sum().reset_index() \
    .sort_values(by='sales_amount', ascending=False).head(10)
print(top_revenue_products)
```

```
     product_type  sales_amount
1     Own Brand\r     369728277
0  Distribution\r     145259554
```

Key Insights and Outcomes

- Top Product Types: Key revenue-driving products.
- High-Margin Zones: Profitable zones ideal for expansion.
- Customer Loyalty: Strong repeat customer base.
- Monthly Trends: Seasonal patterns for inventory planning.

Summary

- Completed end-to-end data handling, from loading to cleaning and analysis
- Key insights derived from sales data can inform business strategy


THANKS