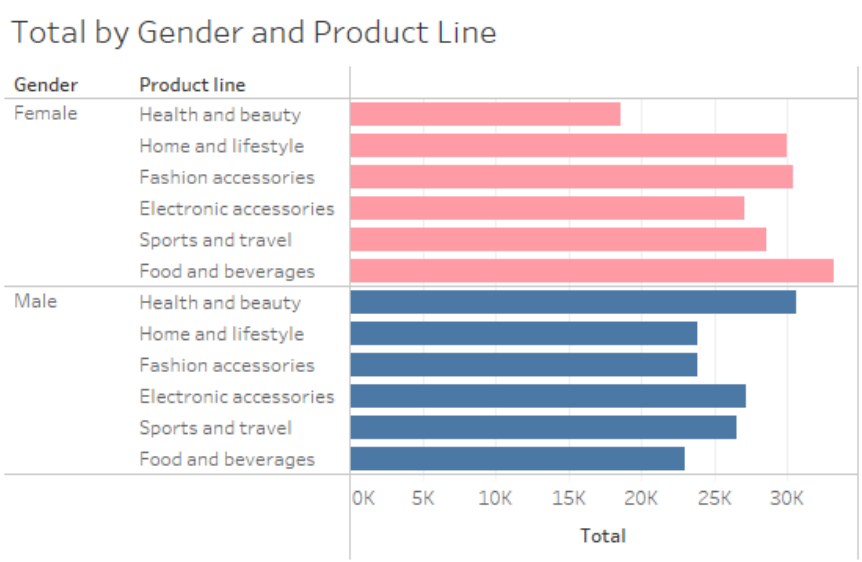
Real Time Sales Dashboard

Showcases real-time sales data monitoring and interactive analytics using Tableau and MySQL integrated with Apache Kafka refresh data.

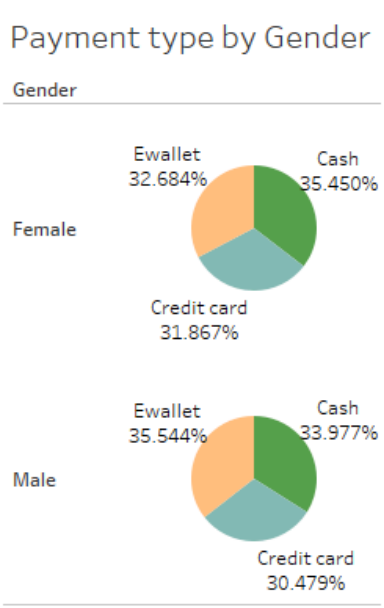
Features:

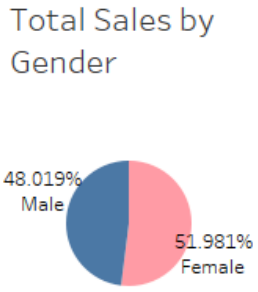
- Real-time updates from a streaming pipeline
- Interactive filters for data columns such as regions, products, and time frames.
- Highlights sales trends, top performing products, and regional breakdown.

Part 1: For this project, I wanted to create a visual dashboard to demonstrate data trends and analyses to business people. First I took a data set called “supermarket sales” from Kaggle.com, which includes information on the region, customers gender, total sale cost, etc. I used Python to load, clean, and analyze the data, I created a MySQL database to store the data, and I used Tableau to make a dashboard based on the data. Below is an image of the dashboard:

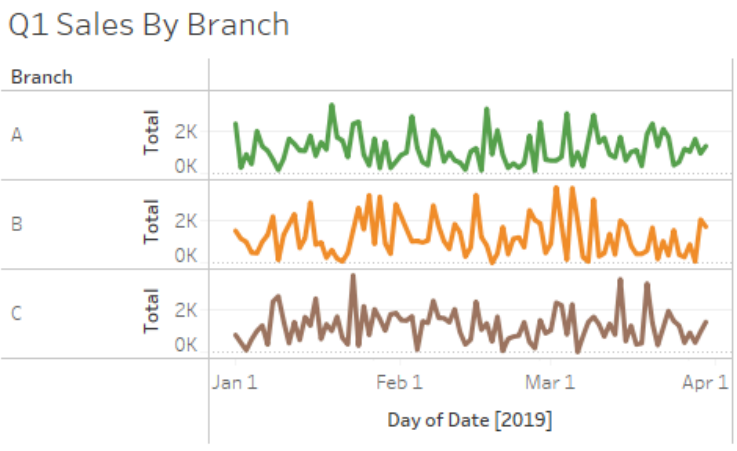


Men tend to spend significantly more than women on Health and beauty products and women spend considerably more on Food and Beverages.





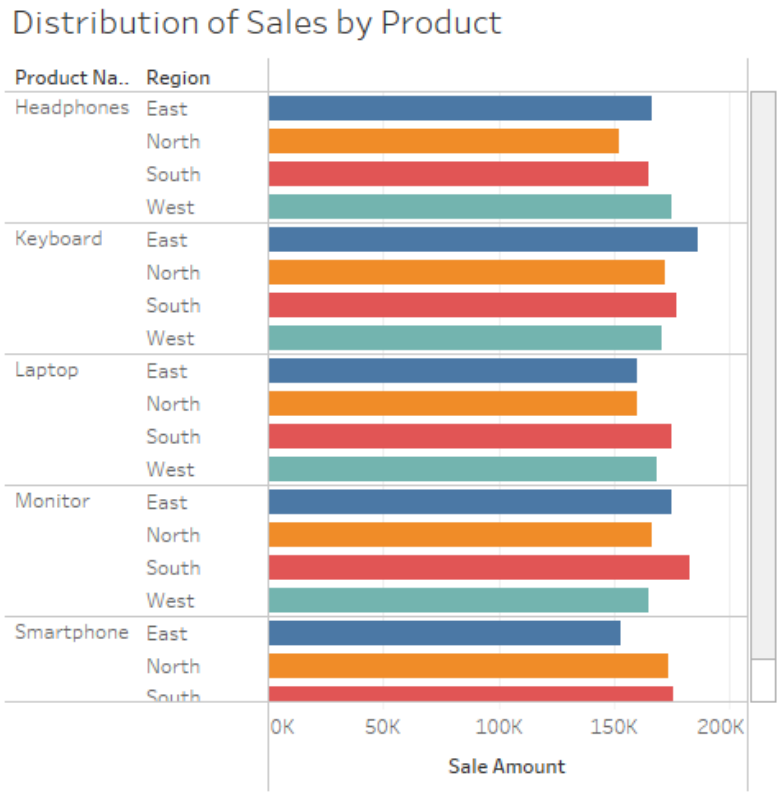
Females spend slightly more money than males. Males prefer to pay with ewallets while females prefer to pay in cash.

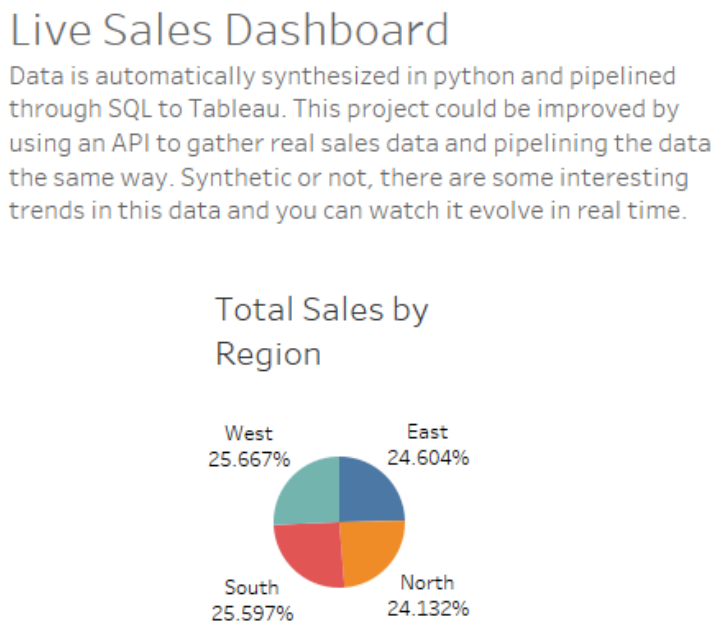


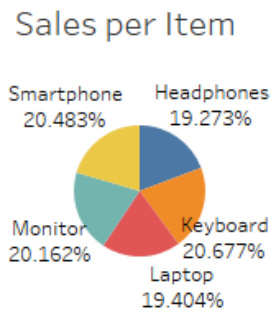
The sales trends are roughly the same for each branch.

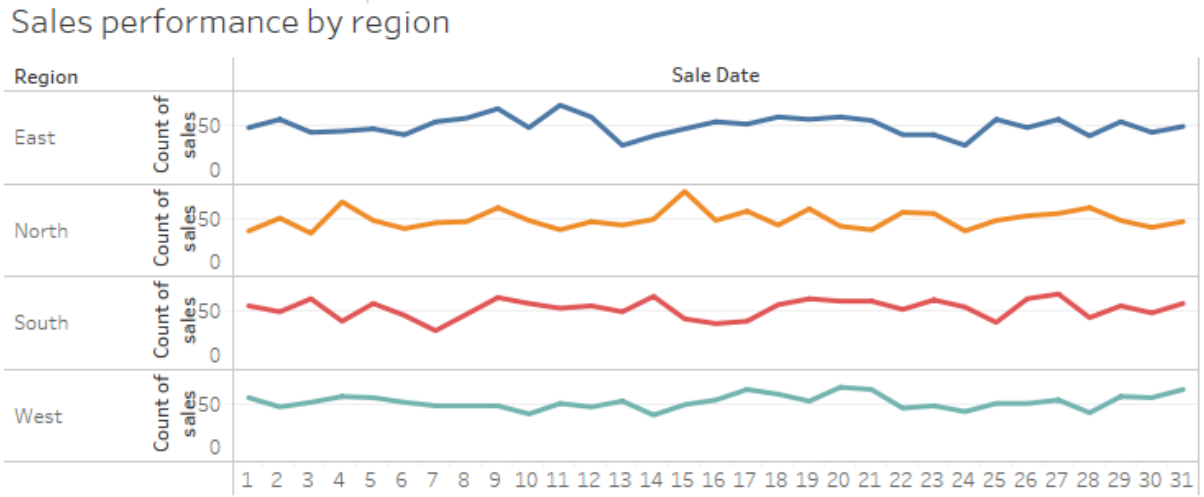
Mandalay could sell more sports and travel merchandise due to it’s location seriving as a major transportation hub in Myanmar. Naypyitaw is the political center of Myanmar and attracts many tourists which could explain why food and beverages were most popular. Yangon is wealthier than the other two areas, which can explain why home and lifestyle is most popular.

Part 2: After completing the first dashboard, I wanted to improve the project by having the dashboard update in real time as new information becomes available. There are many APIs that can export sales data from popular websites like ebay and amazon, but they are not freely accessible. Instead, I wrote a python script which will randomly generate data continuously every second. The script sends the newly generated data to SQL through Docker to Tableau where it automatically updates. Below are two screenshots of the dashboard, taken seconds after one another:

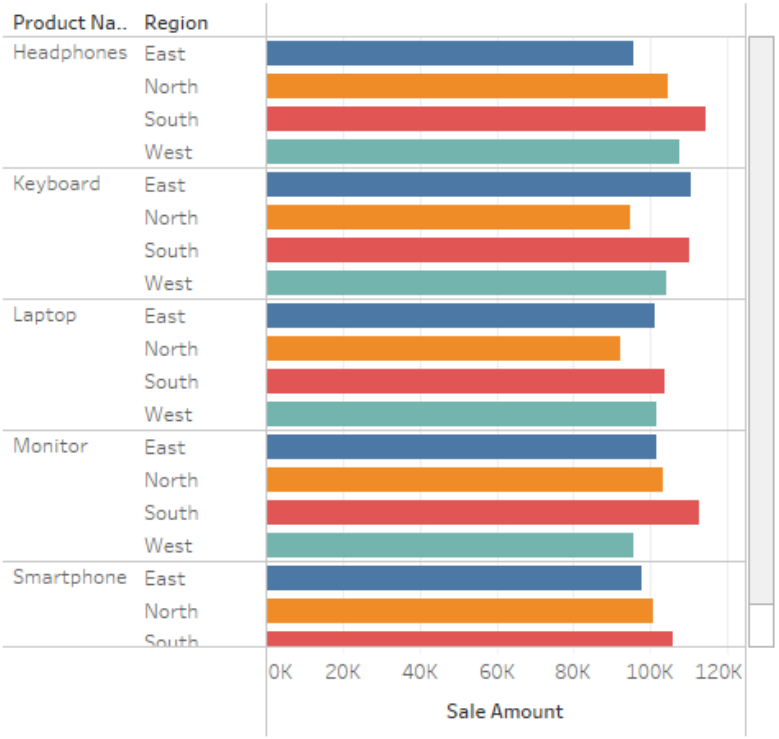








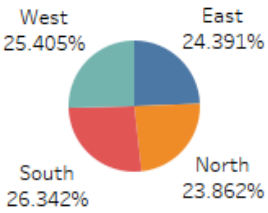
Distribution of Sales by Product



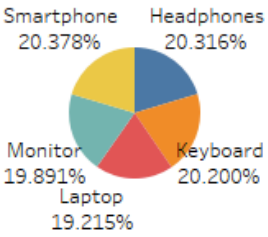
Live Sales Dashboard

Data is automatically synthesized in python and pipelined through SQL to Tableau. This project could be improved by using an API to gather real sales data and pipelining the data the same way. Synthetic or not, there are some interesting trends in this data and you can watch it evolve in real time.

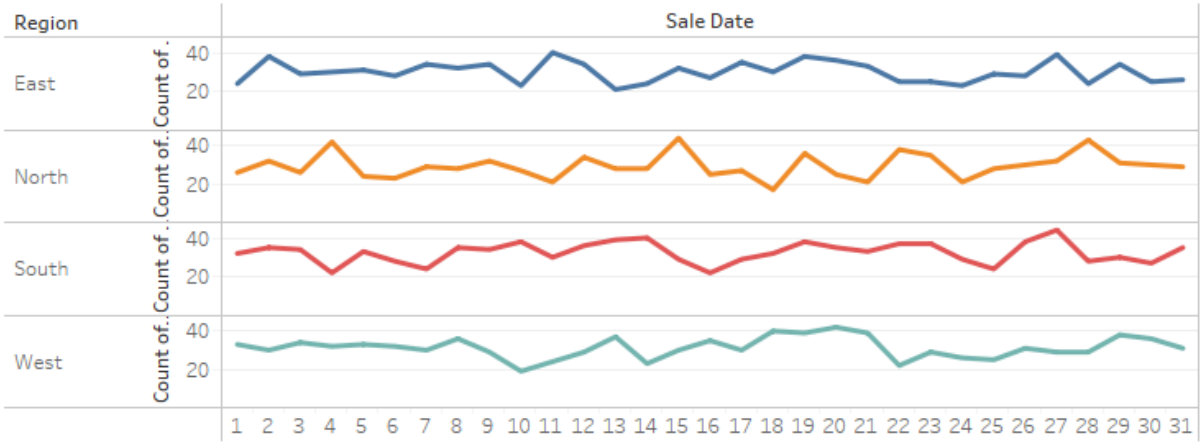
Total Sales by Region



Sales per Item



Sales performance by region



Technology used: Python, Jupyter notebook, MySQL, Docker virtual desktop, Tableau
Techniques: Data analysis, data cleaning, data simulation, data pipelines
Python Code:

```
from confluent_kafka import Producer, KafkaError
import json
import random
import string
from datetime import datetime, timedelta
import time
import mysql.connector

# Function to generate synthetic sales data
def generate_sales_data(num_records):
    products = ['Laptop', 'Smartphone', 'Headphones', 'Monitor', 'Keyboard']
    regions = ['North', 'South', 'East', 'West']

    sales_data = []
    for _ in range(num_records):
        transaction_id = ''.join(random.choices(string.ascii_uppercase +
string.digits, k=10))
        customer_id = random.randint(1000, 9999)
        product_id = random.randint(1, len(products))
        product_name = products[product_id - 1]
        sale_amount = round(random.uniform(100, 1000), 2)
        sale_date = datetime.now() - timedelta(days=random.randint(0, 30))
        region = random.choice(regions)

        sales_data.append({
            'transaction_id': transaction_id,
            'customer_id': customer_id,
            'product_id': product_id,
            'product_name': product_name,
            'sale_amount': sale_amount,
            'sale_date': sale_date.strftime('%Y-%m-%d %H:%M:%S'),
            'region': region
        })
    return sales_data

# Function to send data to Kafka
```

```
def send_to_kafka(producer, topic, data):
    for sale in data:
        try:
            producer.produce(topic, key=sale['transaction_id'],
value=json.dumps(sale))
            print(f"Successfully sent record with transaction_id:
{sale['transaction_id']} to Kafka.")
        except KafkaError as e:
            print(f"Failed to send record {sale['transaction_id']} to Kafka:
{e}")
    producer.flush()  # Ensure all data is sent

# Function to save data to MySQL
def save_sales_data_to_mysql(sales_data):
    try:
        conn = mysql.connector.connect(
            host='localhost',
            user='root',
            password='nice try hacker',
            database='sales_dashboard',
            port=3307
        )
        cursor = conn.cursor()
        print("Connected to MySQL successfully!")
    except mysql.connector.Error as e:
        print(f"Error connecting to MySQL: {e}")
        return

    for record in sales_data:
        print(f"Attempting to insert: {record['transaction_id']}")
        try:
            cursor.execute("""
                INSERT INTO sales (transaction_id, customer_id, product_id,
product_name, sale_amount, sale_date, region)
                VALUES (%s, %s, %s, %s, %s, %s, %s)
            """, (record['transaction_id'], record['customer_id'],
record['product_id'],
                  record['product_name'], record['sale_amount'],
record['sale_date'], record['region']))
```

```
            print(f"Successfully inserted record {record['transaction_id']}")
        except mysql.connector.Error as e:
            print(f"Error inserting record {record['transaction_id']}: {e}")

    conn.commit()  # Commit changes to the database
    print(f"{len(sales_data)} records committed to MySQL.")
    cursor.close()
    conn.close()

# Kafka producer setup
producer = Producer({
    'bootstrap.servers': 'localhost:9092'
})

# Topic to publish to
topic = 'sales_topic'

# Simulate sending data
def simulate_sales():
    while True:
        sales_data = generate_sales_data(10)
        send_to_kafka(producer, topic, sales_data)
        save_sales_data_to_mysql(sales_data)
        print(f"{len(sales_data)} records inserted into MySQL and sent to
Kafka.")
        time.sleep(5)

# Start generating and sending sales data
simulate_sales()
```

Improvements:

- This product can be made better by using APIs to gather real sales data. You would only need to change a bit of code at the beginning as the data would then be pipelined the same way once processed in python.
- We could change the synthesized data to be more resemblant of real sales data instead of just random.