

## **Programming Lab**

Summer Term 2024

### **Portfolio Review– Data Analysis Portal\_ OMAN\_PIZZA**

#### **Project Manager:**

Omar Abd Alwahed 1407819

Artur Gubarkov 1305830

Nasratullah Ahmadzai 1443103

Meles Lemma Tegegne Demoz 1387667

#### **Lecturer:**

Sebastian Bremm

#### **Submission:**

21.07.2024

#### **Document version**

Version	Datum	Autor	GitHub Repository
---------	-------	-------	-------------------

1.0	21.07.2024	OMAN	<a href="https://github.com/khan2999/OMAN">https://github.com/khan2999/OMAN</a>
-----	------------	------	---

## 1 Table of Contents

### 2 Project Description ..... 5

### 3 Main Page..... 6

#### 3.1 Dashboard.....7

#### 3.2 User controls.....7

##### 3.2.1 Time period selection ..... 7

##### 3.2.2 Other more detailed analysis selection..... 8

#### 3.3 Business Analysis Container .....8

##### 3.3.1 Total revenue ..... 8

##### 3.3.2 Average order value..... 9

##### 3.3.3 Total orders ..... 9

##### 3.3.4 Total products sold ..... 9

#### 3.4 Business Analysis Charts .....9

##### 3.4.1 Income timeline chart ..... 10

##### 3.4.2 Top 5 stores by income..... 10

##### 3.4.3 Top 5 products ..... 10

##### 3.4.4 Customer types ..... 11

#### 3.5 Problems and solutions..... 11

##### 3.5.1 Overlapping charts..... 11

##### 3.5.2 Bad visibility of charts ..... 11

##### 3.5.3 Data integration and visualization..... 11

### 4 Location Based Analysis ..... 12

#### 4.1 Interactive Map ..... 12

##### 4.1.1 Revenue Markers ..... 12

##### 4.1.2 Marker Size Variation..... 12

##### 4.1.3 Dynamic Interaction..... 13

##### 4.1.4 Legend ..... 13

#### 4.2 User Controls ..... 13

##### 4.2.1 Data Selection ..... 13

4.2.2	Action Buttons .....	13
4.2.3	Select Year .....	14
4.3	<i>Business Analysis Charts</i> .....	14
4.3.1	Total Sales Chart.....	14
4.3.2	Customer Count Chart .....	14
4.3.3	Order Count Chart .....	15
4.3.4	Average Spending Chart.....	15
4.3.5	Customer Distance Chart .....	15
4.3.6	Cross-Sell Chart.....	16
4.3.7	Summary Table .....	16
4.4	<i>Challenges and Issues</i> .....	16
4.4.1	Authentication with GitHub .....	16
4.4.2	Handling Data and Server Responses .....	17
4.4.3	Data Integration and Visualization.....	17
4.4.4	Interactive Map Markers .....	17
4.4.5	Creating and Positioning the Legend .....	17
<b>5</b>	<b><i>Chart Based Analysis</i></b> .....	<b>21</b>
5.1	<i>Implementation of the Main Page</i> .....	22
5.1.1	Frontend Implementation .....	22
5.1.2	Backend Implementation.....	22
5.1.3	Charts on the Main page and their Insights : .....	23
5.2	<i>Insight Sales Analysis</i> .....	27
	Sales by Day of the Week: .....	27
	Product Size Sales: .....	28
	Store Performance Comparison: .....	29
	Distance Analysis:.....	29
	Store Performance by Category: .....	30
	Customer Segmentation: .....	30
	Inventory Turnover Rate: .....	31
5.3	<i>KPI Reports</i> .....	32
	Revenue Growth: .....	32
	New vs. Returning Customers: .....	33
	Popular Products: .....	33

Conversion Rate: .....	34
Order Frequency:.....	34
<b>6    Number based analysis .....</b>	<b>35</b>
6.1 <i>Implementation of the "Show Revenue" Option .....</i>	35
6.2 <i>Rationale Behind the Implementation .....</i>	35
6.3 <i>Benefits for Managerial Decision-Making.....</i>	36
6.4 <i>Implementation of the "Show Product Portfolio" Option .....</i>	37
Backend Development.....	37
6.5 <i>Rationale Behind the Implementation .....</i>	37
6.6 <i>How This Option Makes Decision-Making Easier for Managers .....</i>	38
6.7 <i>Implementation of the "Show Sales Trend" Option .....</i>	40
6.8 <i>Rationale Behind the Implementation .....</i>	40
6.9 <i>How This Option Makes Decision-Making Easier for Managers .....</i>	40
6.10 <i>Implementation of the "Show Product Popularity" Option .....</i>	41
6.11 <i>Rationale Behind the Implementation .....</i>	42
6.12 <i>How This Option Makes Decision-Making Easier for Managers .....</i>	42
6.13 <i>Implementation of the "Show Sales per Cost" Option.....</i>	43
6.14 <i>Rationale Behind the Implementation .....</i>	44
6.15 <i>How This Option Makes Decision-Making Easier for Managers .....</i>	44
6.16 <i>Challenges and Solutions.....</i>	45
6.16.1 <i>Data Quality Issues.....</i>	45
6.16.2 <i>Technical Challenges.....</i>	45
6.17 <i>Comprehensive Solutions .....</i>	46
6.18 <i>Lessons Learned .....</i>	46
6.19 <i>Improvements for Next Time .....</i>	46
Scalable System Architecture .....	47
User Experience (UX) Enhancements:.....	47
Advanced Analytics Integration: .....	47
6.20 <i>Reflections on Frameworks and Tools .....</i>	47
6.21 <i>However, there were also some challenges: .....</i>	47
Data Inconsistencies and Cleaning:.....	47
6.22 <i>Summary .....</i>	48
<b>7 Order Based Analysis .....</b>	<b>48</b>

7.1 Order Based Analysis overview .....	48
7.2 User Controls .....	49
7.2.1 Data period selection .....	49
7.2.2 Chart type selection for customer order analysis + Orders per customer category .....	49
7.3 Business analysis charts .....	49
7.3.1 Customer order analysis.....	49
7.3.2 Stores by category chart.....	50
7.3.3 Orders per Customer Category .....	51
7.3.4 Products per customer category .....	52
7.4 Problems and solutions .....	52
7.4.1 Getting all the correct data from the database .....	52
7.4.2 Correct data input.....	52
7.4.3 Make only one chart interactive instead of two .....	53

## 2 Project Description

The Pizza Data Analysis and Visualization Tool is a comprehensive platform designed to empower managers with critical insights into their business operations. This tool enhances decision-making and strategic planning by providing a range of analytical features.

The platform allows managers to track revenue trends over time, offering insights into financial performance by year and quarter. It also includes product portfolio analysis, which gives an in-depth look at the sales performance of each product, highlighting both top sellers and underperformers. This information is crucial for effective inventory management and marketing strategies.

Sales trend monitoring is another key feature, providing detailed analyses of sales data across various periods. This helps managers understand seasonal fluctuations and the impact of marketing campaigns, aiding in future planning.

Additionally, the tool offers product popularity insights, showing which products are most favored by customers based on sales figures and ratings. This enables managers to focus their marketing efforts more effectively and ensure high-demand items are well-stocked.

The customer distribution mapping feature uses an interactive map to display customer locations, allowing managers to see which customers are within a 10-mile radius of each store. This visual representation aids in resource allocation and delivery planning. The tool also includes location-based analysis to identify regional trends and customer distributions, supporting targeted marketing initiatives.

Furthermore, the platform provides chart-based analysis for visualizing data trends and patterns, and number-based analysis for gaining precise insights into key metrics. Order-based analysis examines customer purchasing behavior, helping managers plan inventory and optimize sales strategies. A welcome page offers an overview of the tool's features and guides users through the interface, ensuring ease of use.

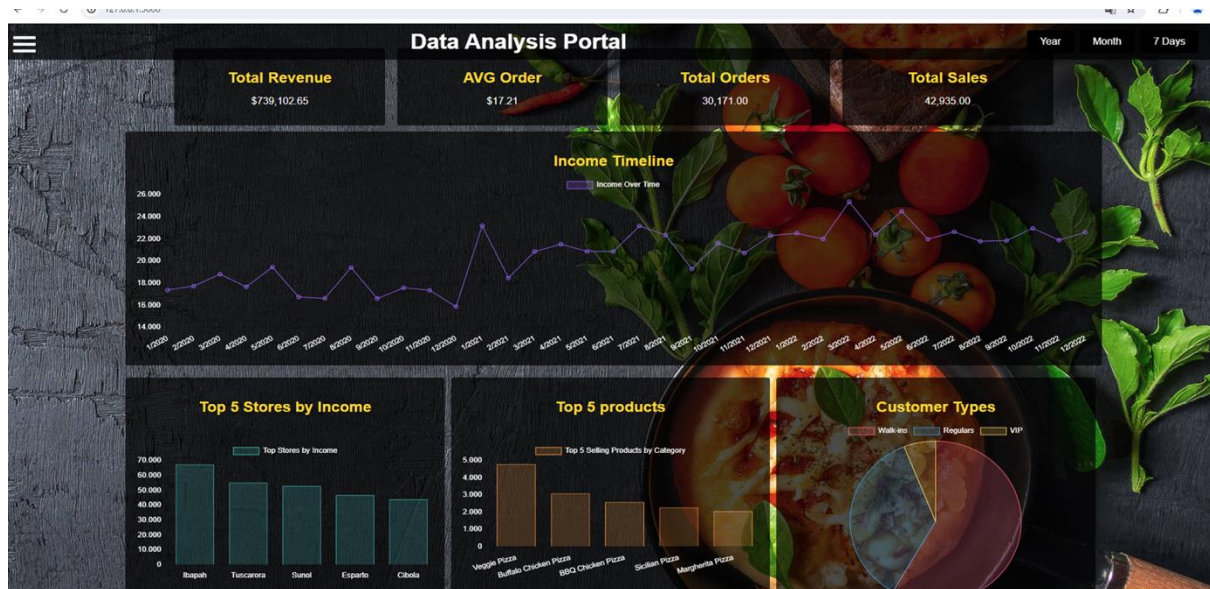
Built with robust technologies like Flask, MySQL, Python, Pandas, Chart.js, and Leaflet.js, the tool ensures efficient data handling, real-time access, and interactive visualizations. The user-friendly interface makes it easy for managers to interact with the data and gain valuable insights.

In summary, the Pizza Data Analysis and Visualization Tool equips managers with the necessary tools to optimize business performance. By leveraging detailed visualizations and actionable data, the tool supports informed decision-making and strategic planning, ultimately driving business success.

### **3 Main Page**

The welcome page is the main page of our web-based application, which focuses on showing overall important information for the pizza chain and getting a quick overview of the most important data.

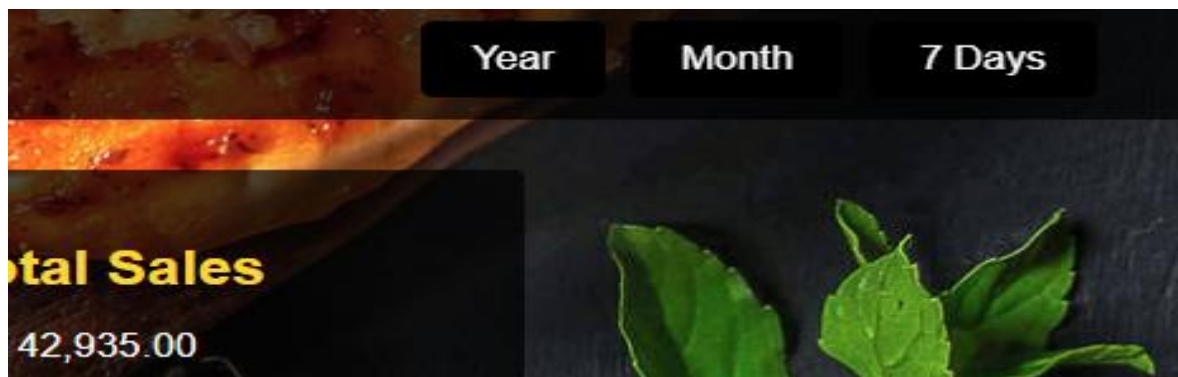
## 3.1 Dashboard



The dashboard on the main page is designed to provide a comprehensive overview of key business metrics for our pizza store chain. The welcome page displays total revenue, average order value, total orders, and total sales in numerical form. Additionally, it features several charts including a timeline chart for income, a bar chart showcasing the top 5 stores by income, the top 5 most ordered products, and a pie chart displaying customer types and their order frequencies.

## 3.2 User controls

### 3.2.1 Time period selection

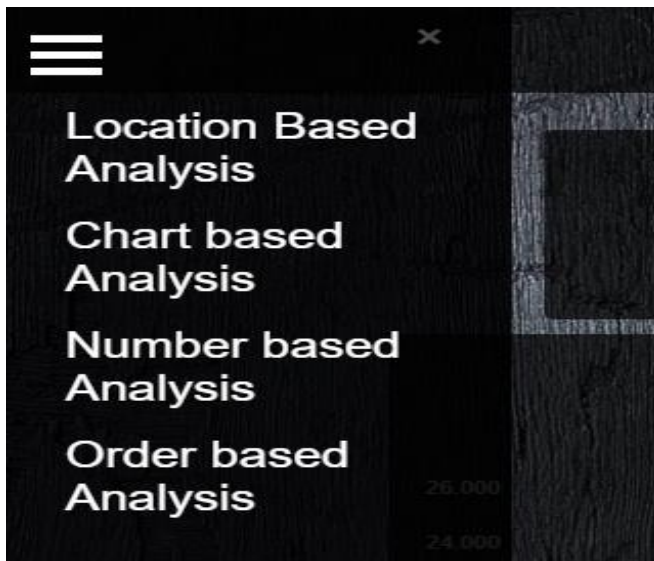


At the top right corner of the welcome page are 3 buttons, to choose which time period the data should be displayed.

- Year button = Time period between 01.01.2022 - 31.12.2022

- Month button = Time period between 01.12.2022 - 31.12.2022
- 7 Days button = Time period between 24.12.2022 - 31.12.2022

### 3.2.2 Other more detailed analysis selection

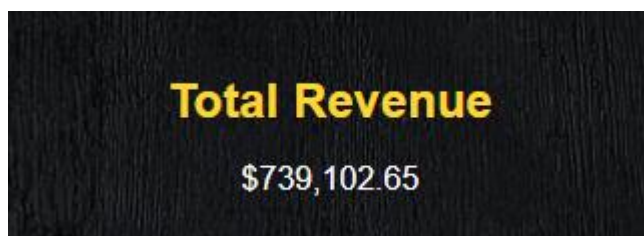


On the top left corner is a navigation bar, which opens up by clicking on the 3 lines. It contains 4 buttons “Location based Analysis”, “Chart based Analysis”, “Number based Analysis” and “Order based Analysis” which when clicking on them redirect to the chosen more specific analysis page.

## 3.3 Business Analysis Container

The container/boxes on the welcome page contain important numeric information for all pizza stores.

### 3.3.1 Total revenue



Shows the total revenue which the pizza company made depending on the time period selected.



### 3.3.2 Average order value



Shows the average order value which the customers had depending on the time period selected.

### 3.3.3 Total orders



Shows the total order count of the company depending on the time period selected.

### 3.3.4 Total products sold

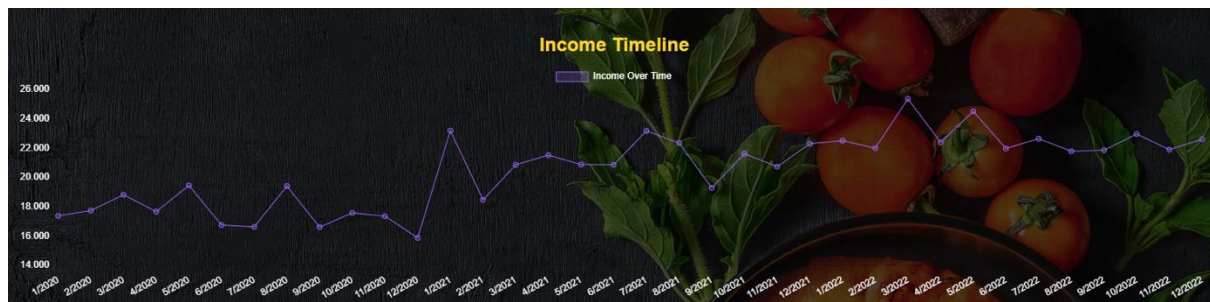


Shows the total number of products sold by the company depending on the time period selected.

## 3.4 Business Analysis Charts

The charts on the page provide a quick visual representation of various business metrics. These charts update accordingly to the button which gets clicked and help to get a quick insight into the overall business data.

### 3.4.1 Income timeline chart



The income timeline chart show the income of the company made with monthly or daily data, depending on the time period chosen.

### 3.4.2 Top 5 stores by income



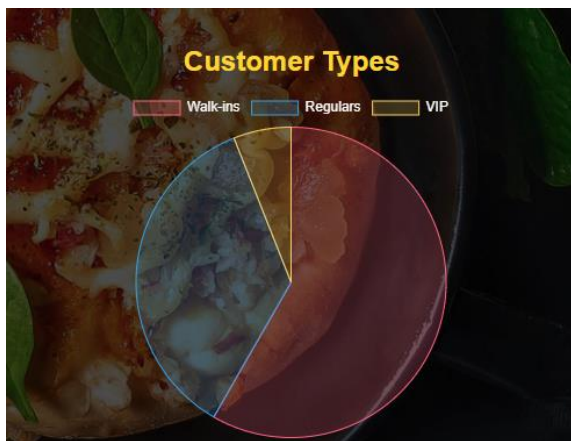
The top 5 stores by income bar chart shows the top 5 stores of the company and how much profit they made.

### 3.4.3 Top 5 products



The top 5 product bar chart shows the top 5 products of the company and how often each pizza got sold.

### 3.4.4 Customer types



The customer types of pie chart show which customer category ordered how often on the selected time period. Customers who ordered only 1-2 times at our stores are called Walk-Ins, customers who ordered 3-9 times Regulars and customers who ordered more than 9 times are called VIPs.

## 3.5 Problems and solutions

During the coding and implementation of new functions, several challenges and problems were encountered and had to be addressed:

### 3.5.1 Overlapping charts

- **Problem:** Some charts overlapped with each other.
- **Solutions:** Edit CSS part so every chart has its own place and is clearly to see.

### 3.5.2 Bad visibility of charts

- **Problem:** Because of the background and the transparency of some charts, they were hard to read.
- **Solution:** Edit transparency value and edit colors of charts to be easy to read on the background.

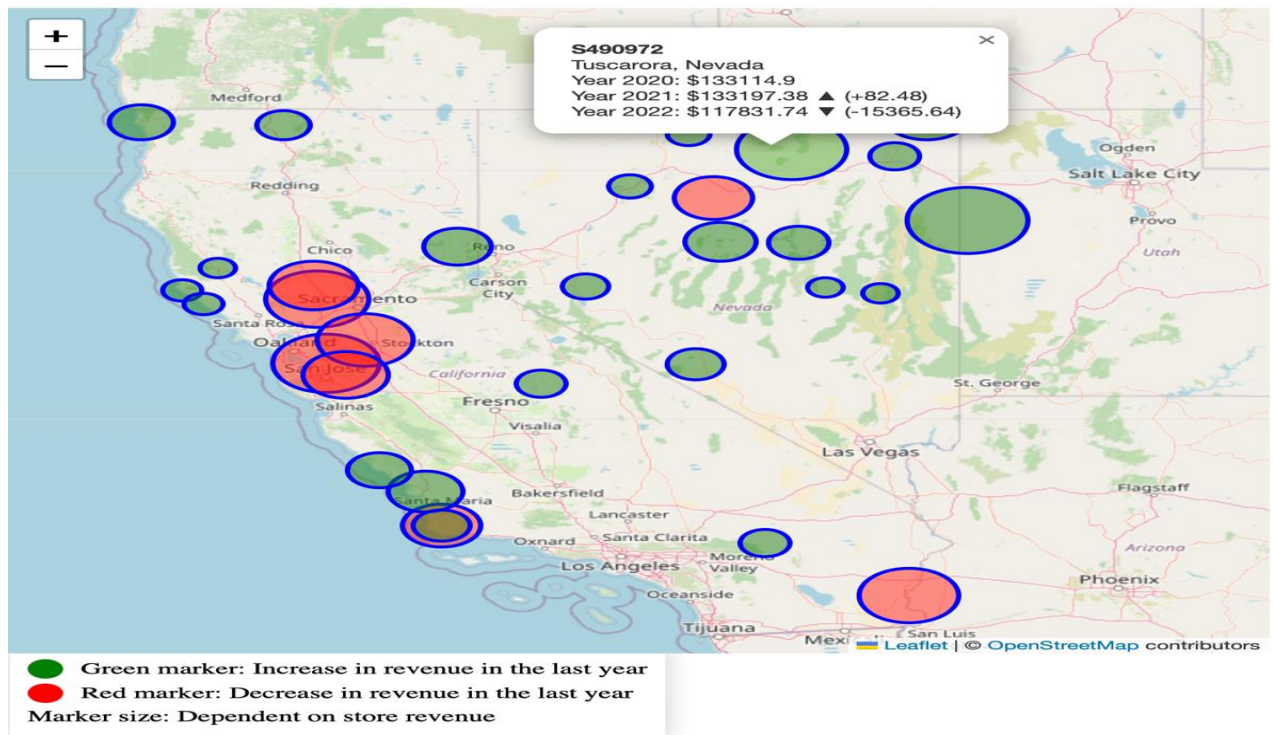
### 3.5.3 Data integration and visualization

- **Problem:** Correct integration and display of dynamic data in the charts.
- **Solution:** Used Chart.js to create interactive charts and implemented functions for dynamically updating the charts based on the selected data.

## 4 Location Based Analysis

This page is part of the comprehensive business analysis web application. It focuses on location-based analysis by visualizing store locations on an interactive map. The map provides a clear overview of store performance based on revenue data.

### 4.1 Interactive Map



The interactive map displays the geographic distribution of stores and offers the following features:

#### 4.1.1 Revenue Markers

- **Green Markers:** Represent stores that have shown an increase in revenue over the past year.
- **Red Markers:** Represent stores that have shown a decrease in revenue over the past year.

#### 4.1.2 Marker Size Variation

The size of the markers is proportional to the revenue of the respective store. A larger symbol indicates higher revenue, while a smaller symbol indicates lower revenue.

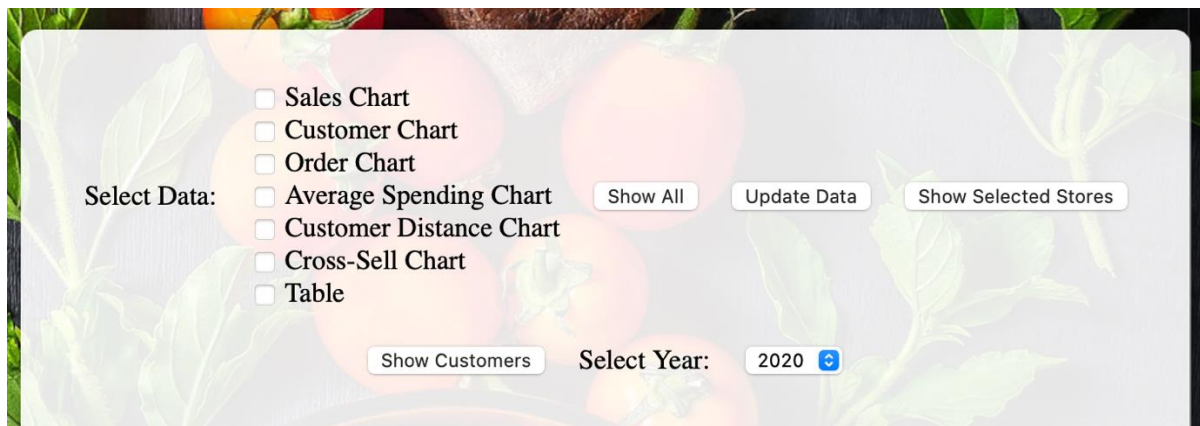
### 4.1.3 Dynamic Interaction

Users can click on the markers to get more information about the respective store. This allows for a detailed analysis of revenue trends and the geographic distribution of stores.

### 4.1.4 Legend

In the bottom left corner of the map, there is a legend that explains the meaning of the different marker colors and sizes.

## 4.2 User Controls



The top section of the page offers comprehensive control options for selecting and displaying various data. Users can display specific charts, update data, and switch between different years.

### 4.2.1 Data Selection

Users can select the following data to display:

- Sales Chart: Chart showing sales figures.
- Customer Chart: Chart showing customer numbers.
- Order Chart: Chart showing orders.
- Average Spending Chart: Chart showing average customer spending.
- Customer Distance Chart: Chart showing customer numbers based on their distance to the store.
- Cross-Sell Chart: Chart showing cross-selling data.
- Table: Table summarizing the selected data.

### 4.2.2 Action Buttons

- Show All: Displays all available charts and data.
- Update Data: Updates the displayed data based on the current selection criteria.
- Show Selected Stores: Displays only the selected stores on the map.
- Show Customers: Displays the locations of customers on the map.



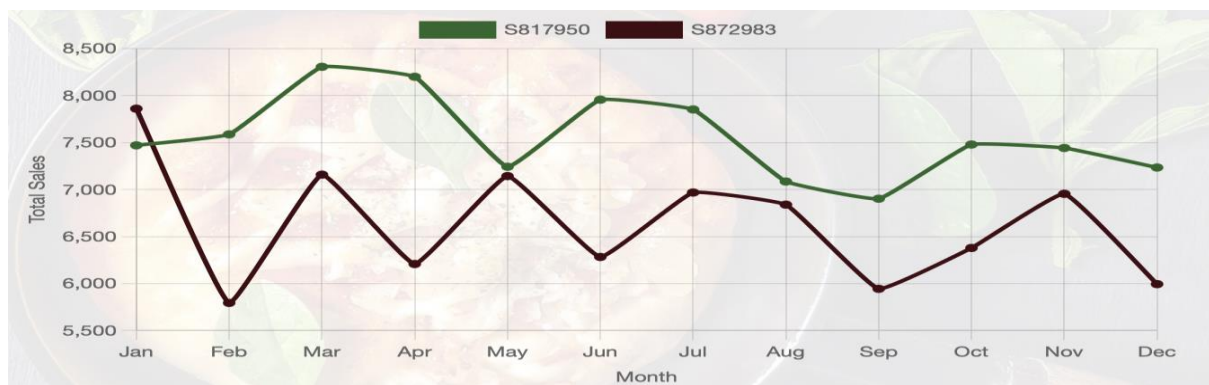
### 4.2.3 Select Year

A dropdown menu allows the user to select the year for which the data should be displayed. This is especially useful for analyzing trends and changes over multiple years.

## 4.3 Business Analysis Charts

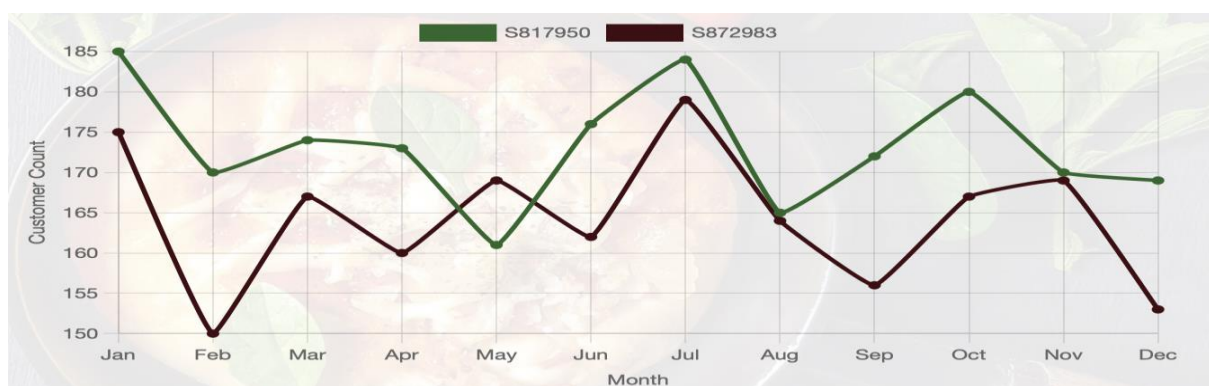
The charts on the page provide a detailed visual representation of various business metrics. These charts are interactive and help gain deep insights into the business data.

### 4.3.1 Total Sales Chart



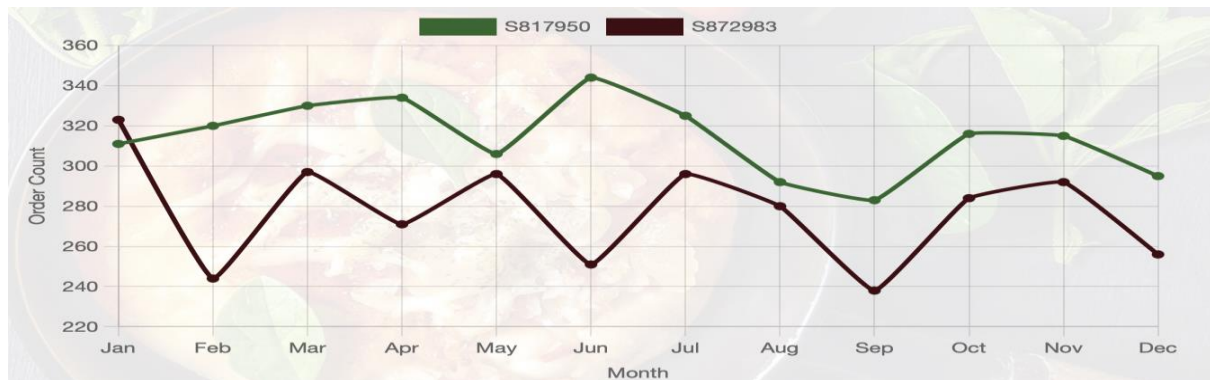
The sales chart shows the monthly total sales for various stores in the selected year. The lines in different colors represent different stores.

### 4.3.2 Customer Count Chart



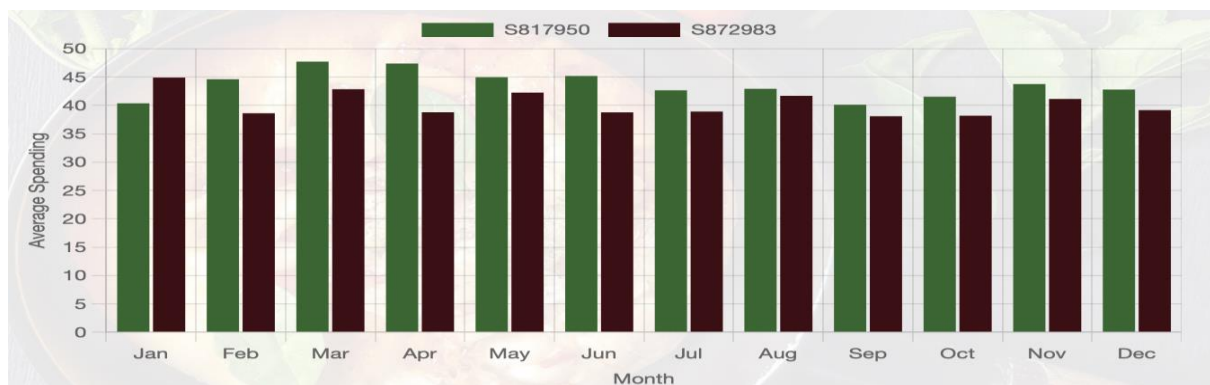
The customer chart shows the monthly customer count for various stores in the selected year. Similar to the sales chart, the lines in different colors represent different stores.

### 4.3.3 Order Count Chart



The order chart shows the monthly order count for various stores in the selected year. The lines in different colors represent different stores.

### 4.3.4 Average Spending Chart



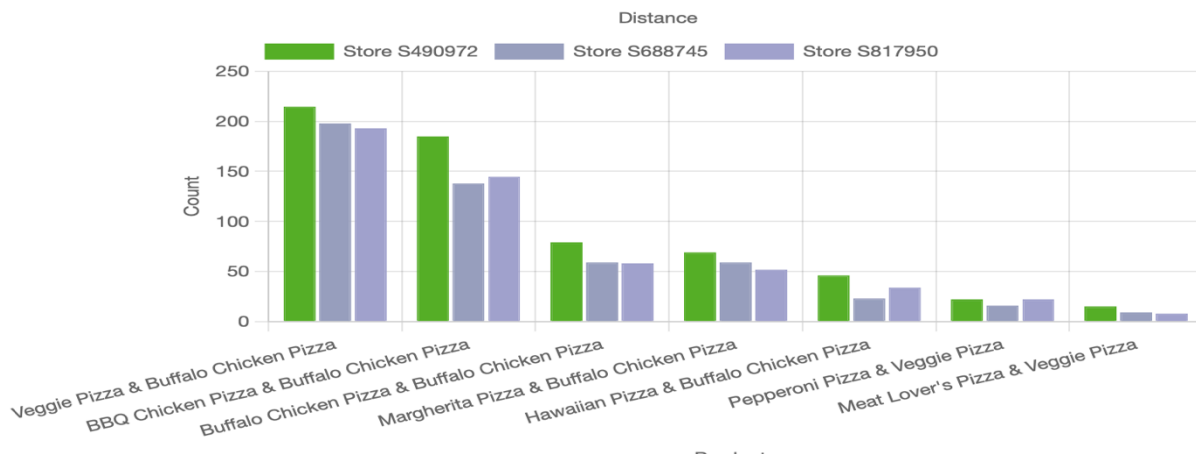
The average spending chart shows the average spending per month for various stores in the selected year. The bars in different colors represent different stores.

### 4.3.5 Customer Distance Chart



The customer distance chart shows the number of customers based on their distance to the stores. The bars in different colors represent different stores.

### 4.3.6 Cross-Sell Chart



The cross-sell chart shows the number of sales of product combinations that are frequently bought together. The bars in different colors represent different stores.

### 4.3.7 Summary Table

Store	Delta from Previous Year	Revenue	Delta from Previous Year	Customers	Delta from Previous Year	Orders
S490972	N/A	\$133114.90	N/A	N/A	N/A	N/A
S688745	N/A	\$105304.74	N/A	N/A	N/A	N/A
S817950	N/A	\$94295.17	N/A	N/A	N/A	N/A

The table summarizes the key metrics of the selected stores. It provides a quick overview of revenue trends, customer counts, and orders.

Columns: Store: Store identification number Delta from Previous Year: Change compared to the previous year Revenue: Total revenue Customers: Number of customers Orders: Number of orders.

## 4.4 Challenges and Issues

During the coding and implementation of new functions, several challenges and problems were encountered and had to be addressed:

### 4.4.1 Authentication with GitHub

- **Problem:** GitHub removed support for password authentication for Git operations over HTTPS, leading to authentication errors.
- **Solution:** Introduced Personal Access Tokens (PAT) or SSH keys for authentication.



#### 4.4.2 Handling Data and Server Responses

- **Problem:** Ensuring that data is correctly fetched and processed from the server.
- **Solution:** Used Promises and the Fetch API for asynchronous data requests and error handling to ensure the application is robust against network errors.

#### 4.4.3 Data Integration and Visualization

- **Problem:** Integration and display of dynamic data in the charts.
- **Solution:** Used Chart.js to create interactive charts and implemented functions for dynamically updating the charts based on the selected data.

#### 4.4.4 Interactive Map Markers

- **Problem:** Display and interactivity of map markers, including dynamic sizing and color coding.
- **Solution:** Used Leaflet.js to create and manage maps and markers, and implemented event handlers for user interactions.

#### 4.4.5 Creating and Positioning the Legend

- **Problem:** The legend must be clear and informative to explain the significance of the markers on the map to users.
- **Solution:** Designed and styled the legend using CSS to display it in the correct position and ensure it is intuitive and informative.

##### 4.4.5.1 Implementation of the "View Customers on the Map" Feature

To enhance the functionality of the Pizza Data Analysis project, a new feature was added that allows users to visualize customers on an interactive map. This feature lets users click on specific store locations to see a pop-up message that shows customers within a 10-mile radius of the store.

On the backend, the system was extended to provide geolocation data for both customers and stores. This involved creating endpoints that retrieve and deliver accurate coordinates. This geolocation data is essential for plotting customers and stores correctly on the map.

On the frontend, an interactive map was created using a mapping library that enables dynamic and interactive displays. When users click the "View Customers on the Map" button, the system fetches customer and store data from the backend and displays them as markers on the map. Each store marker is interactive, and clicking on it triggers a pop-up message showing customers within a 10-mile radius.

#### 4.4.5.2 Rationale Behind the Implementation

The "View Customers on the Map" feature provides managers with a powerful visual tool to understand customer distribution in relation to store locations. This visualization aids in several key areas:

**1. Visualize Customer Distribution:**

- Managers can easily see where their customers are located geographically.
- This helps identify areas with high customer density, which can be targeted for marketing campaigns or considered for new store locations.

**2. Improve Customer Service:**

- By knowing which stores serve the most customers within a specific radius, managers can allocate resources more effectively.
- Understanding customer locations relative to stores aids in planning efficient delivery routes, improving delivery times and customer satisfaction.

**3. Enhance Strategic Planning:**

- The visual representation supports data-driven decisions about store placements and expansions.
- Managers can evaluate the effectiveness of marketing strategies in different geographical areas and adjust their approach based on customer concentration.

#### 4.4.5.3 How This Feature Enhances Decision-Making

The "View Customers on the Map" feature significantly enhances decision-making by providing clear and interactive visual representation of customer and store locations. This immediate access to geographical data supports several strategic initiatives:

- **Targeted Marketing:**
  - Managers can direct marketing efforts to areas with high customer density, increasing the effectiveness of campaigns.
- **Resource Allocation:**
  - By understanding which stores have the highest customer concentration, managers can allocate resources more efficiently to meet demand.
- **Store Optimization:**
  - Data-driven insights into customer proximity support strategic decisions about where to place new stores or expand existing ones.
- **Efficient Deliveries:**
  - Visualizing customer locations helps plan more efficient delivery routes, reducing costs and improving delivery times.

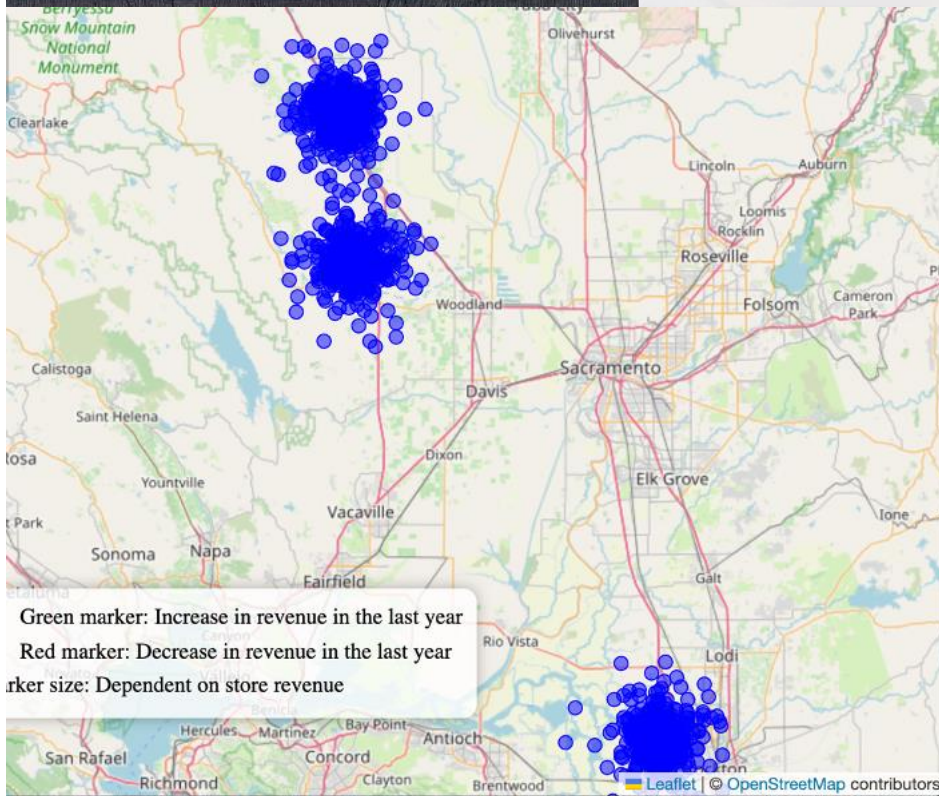
## Store Locations and Data Selection



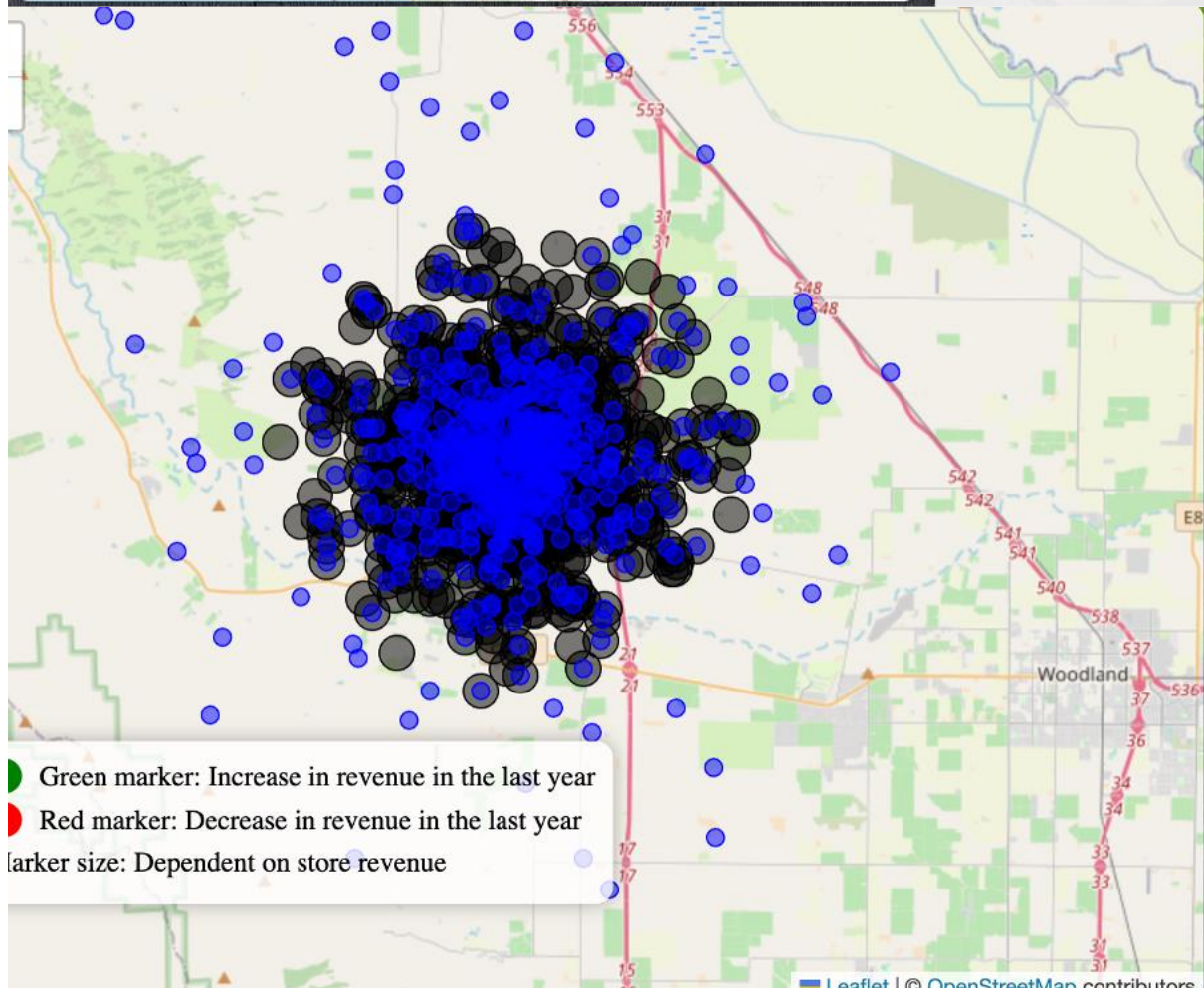
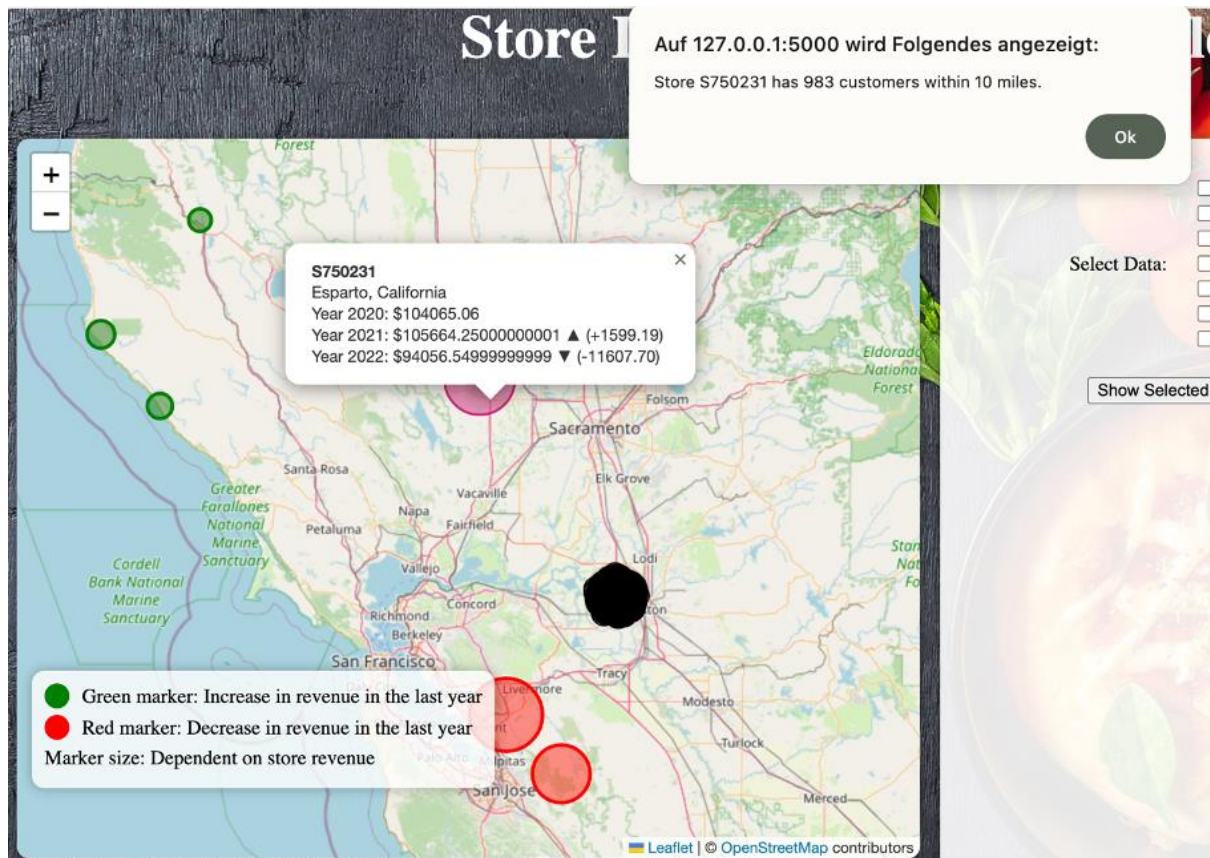
☐ Sales Chart  
☐ Customer Chart  
☐ Order Chart  
☐ Average Spending Chart  
☐ Customer Distance Chart  
☐ Cross-Sell Chart  
☐ Table

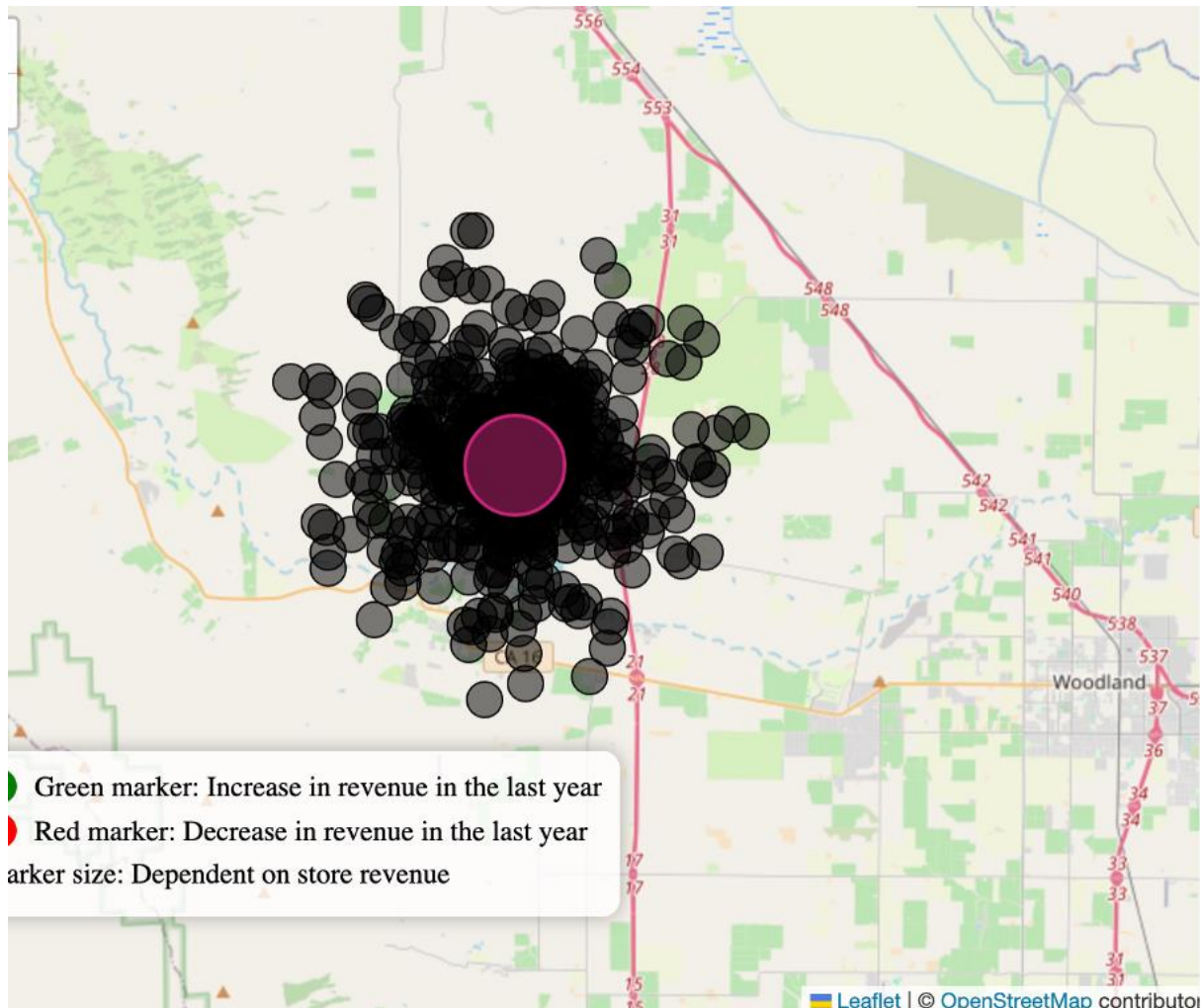
Select Data:

Select Year: 2020









## 5 Chart Based Analysis

### Introduction:

The Charts Based Analysis Dashboard is designed to provide managers and stakeholders with actionable insights into pizza sales performance and customer behaviour. It helps answer key business questions such as:

**Sales Performance:** What are our overall sales trends? How do sales vary by category, time, and location? Which products are most popular?

**Customer Behaviour:** Where are our customers located? How frequently do they order? Are we acquiring new customers over time? What is the average order value and frequency?

**Operational Efficiency:** How can we optimize inventory, staffing, or marketing efforts based on sales and customer data?

## 5.1 Implementation of the Main Page

This part of the documentation provides an overview of the implementation of seven different charts used for visualizing sales data. The backend uses Python, Flask, MySQL, and Plotly, while the frontend uses HTML, JavaScript, and the ECharts library. On the first page, the manager/user can perform an overall analysis of the data. The Manager/user can choose between seven different charts and could filter the data according to Date, category (in this case, pizza), and states. Some charts also allow for drill-down to more detailed insights. This ability to drill down feature will appear, when the user hover over the charts and the charts colour changes to green. The user/Manager has the possibility to either drilldown by clicking on the chart itself or clicking on the “Drilldown” Button.

### 5.1.1 Frontend Implementation

The frontend is responsible for rendering the charts and providing interactivity. The main components are:

#### 1. HTML Elements:

- chartSelector: Dropdown to select the type of chart.
- dateFilters: Date range selectors.
- categoryFilters: Category filter dropdown.
- stateSelector: Dropdown for selecting the state.
- filterButton: Button to apply selected filters.
- chartContainer: Container where charts are rendered.
- loadingSpinner: Spinner displayed during data loading.

#### 2. JavaScript Functions:

- showLoading(): Displays the loading spinner.
- hideLoading(): Hides the loading spinner.
- showTooltip(x, y, value): Displays tooltips on the chart.
- hideTooltip(): Hides tooltips on the chart.
- renderChart(chartData, chartLabels, chartType, title): Renders a chart based on the given data and type.
- updateChart(): Fetches data based on selected filters and updates the chart.
- updateFilters(): Updates the visibility of filters based on the selected chart type.

#### 3. Event Listeners:

- Listeners for chart selection changes, filter button clicks, and chart resize events.

### 5.1.2 Backend Implementation

The backend is built using Python, Flask, and MySQL, providing API endpoints to fetch data for each chart. The key steps are:

**Database Connection:** Establish a connection to the MySQL database.

#### API Endpoints:

- /filter: Fetches data based on chart type and applied filters.

- /states: Retrieves the list of states for the state selector.
- /drilldown: Fetches detailed data for drill-down functionality.

Data Retrieval and Processing:

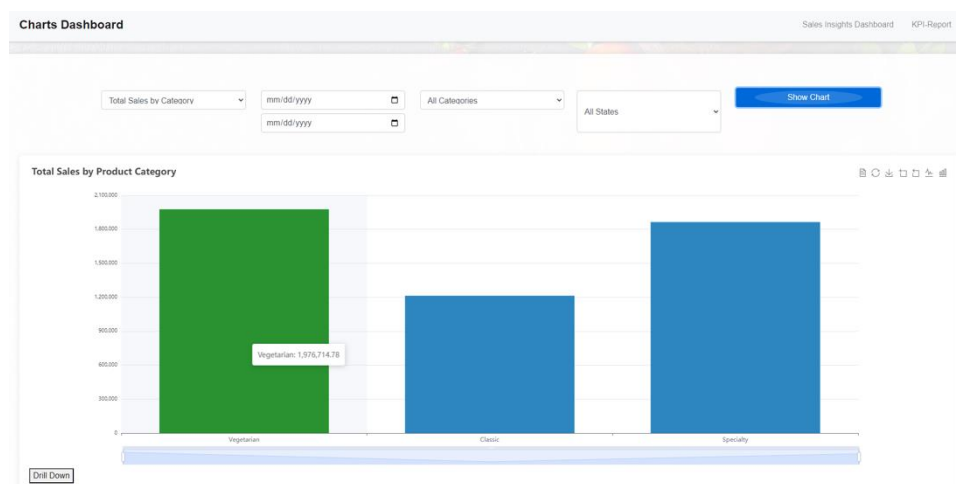
- Execute SQL queries based on chart type and filters.
- Convert decimal values to float for compatibility.
- Return data in JSON format suitable for visualization.

Error Handling: Manage database connection issues and query execution errors

Logging: Track query execution and errors for monitoring and troubleshooting.

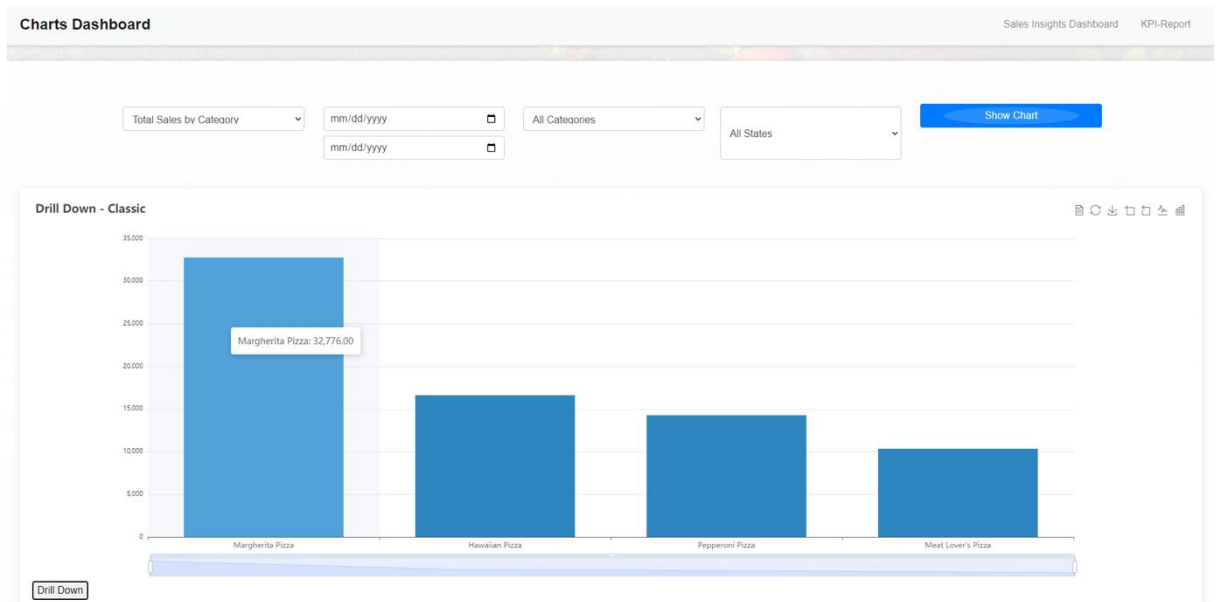
### 5.1.3 Charts on the Main page and their Insights :

**Total Sales by Category:** Displays the total sales revenue generated by each product category, helping identify the most profitable categories.



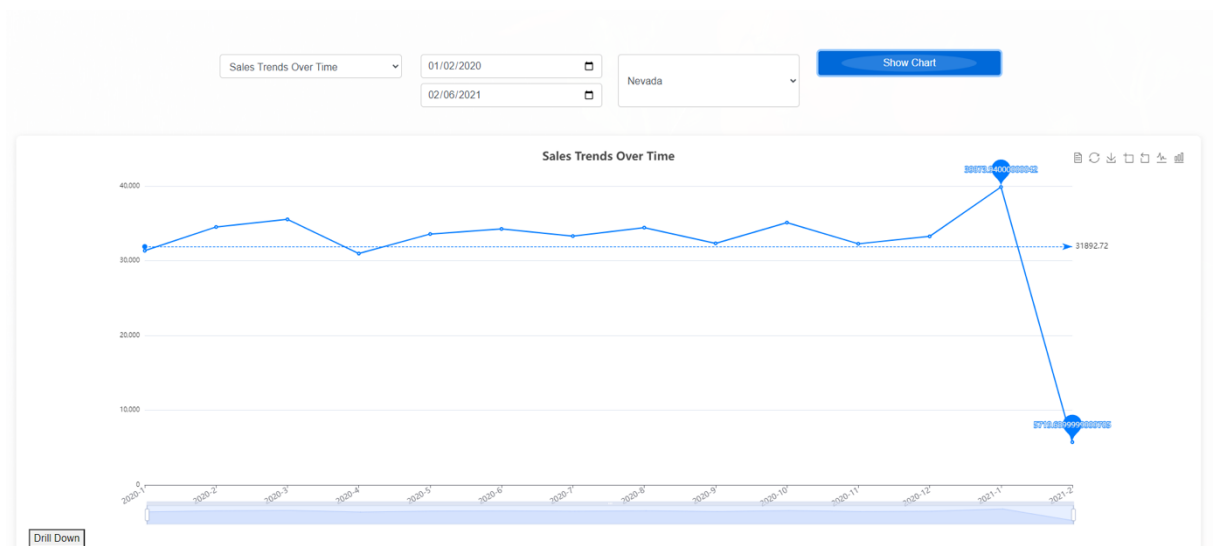
**Drill-down:** Clicking on a category segment can show detailed sales data for the types of pizza within that category. Clicking on a category segment can show detailed sales data for the types of pizza within that category. For example clicking on a category segment (classic) the following details of the Category will show up.





**Sales Trends Over Time:** Visualizes monthly sales trends, helping to identify seasonal patterns, growth or decline trends, and the impact of marketing campaigns or other events.

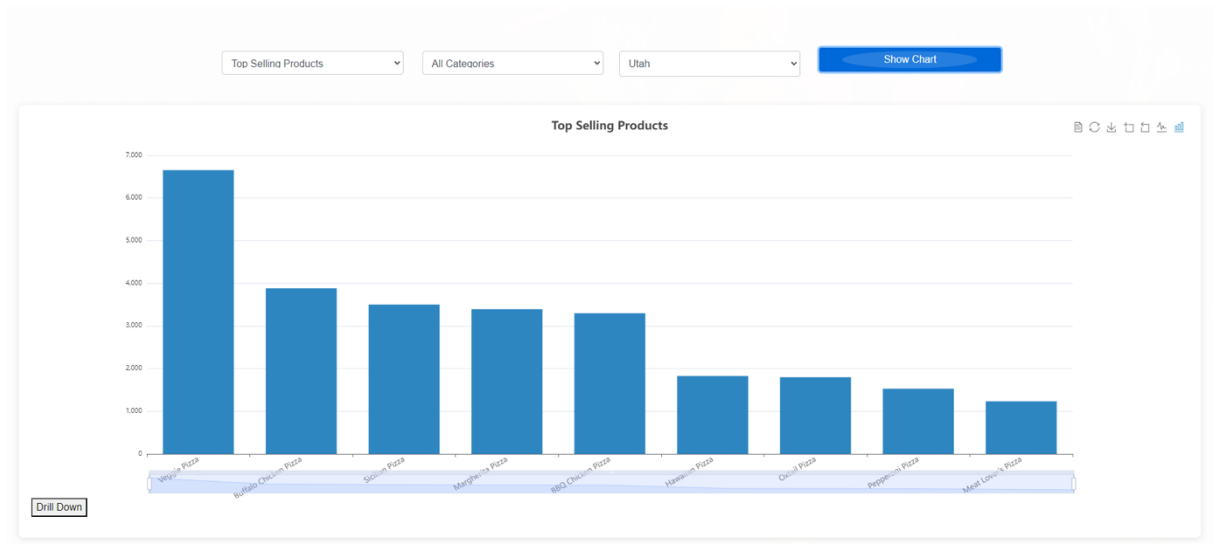
**Example:** Manager/ User wants to see the trends of sale from 01.02.2020 till 02.06.2021 just for the stores that are in Nevada :



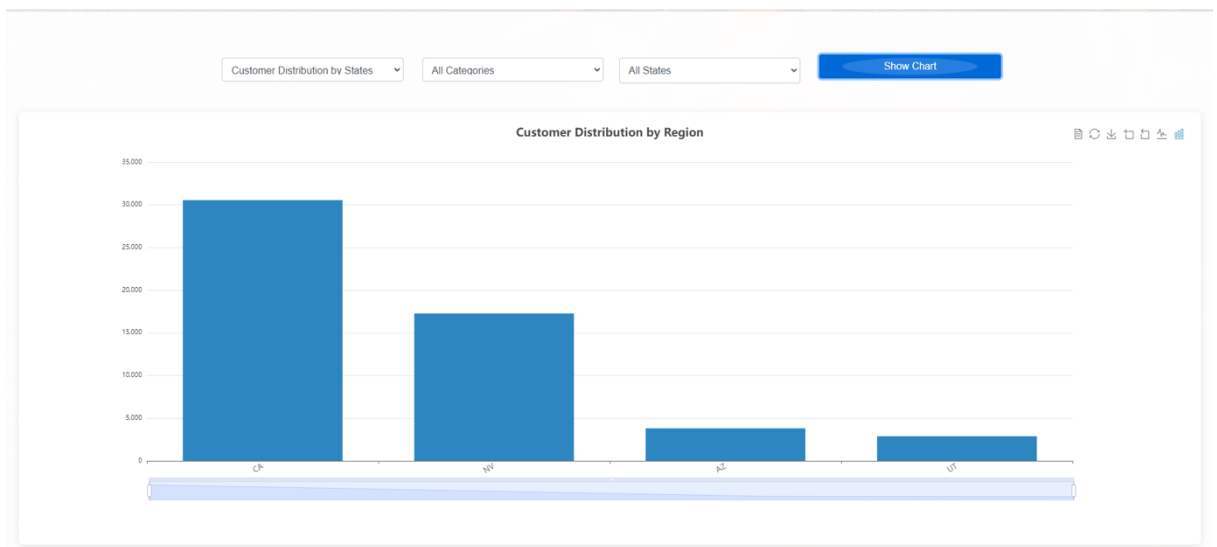
**Top Selling Products:** Identifies the best-selling pizza products, either by quantity sold or total revenue, highlighting the most popular items.

**Example** Top selling products in Utah:

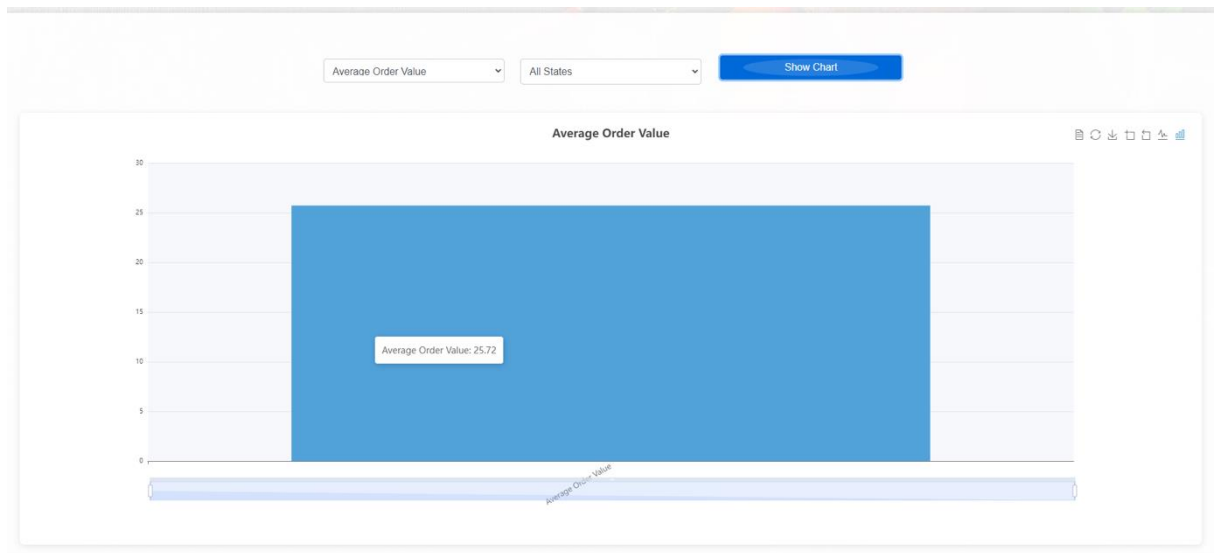




**Customer Distribution by States:** Shows the geographic distribution of customers based on their locations (using latitude and longitude from the customer table). This can inform targeted marketing campaigns or reveal areas with high customer density.

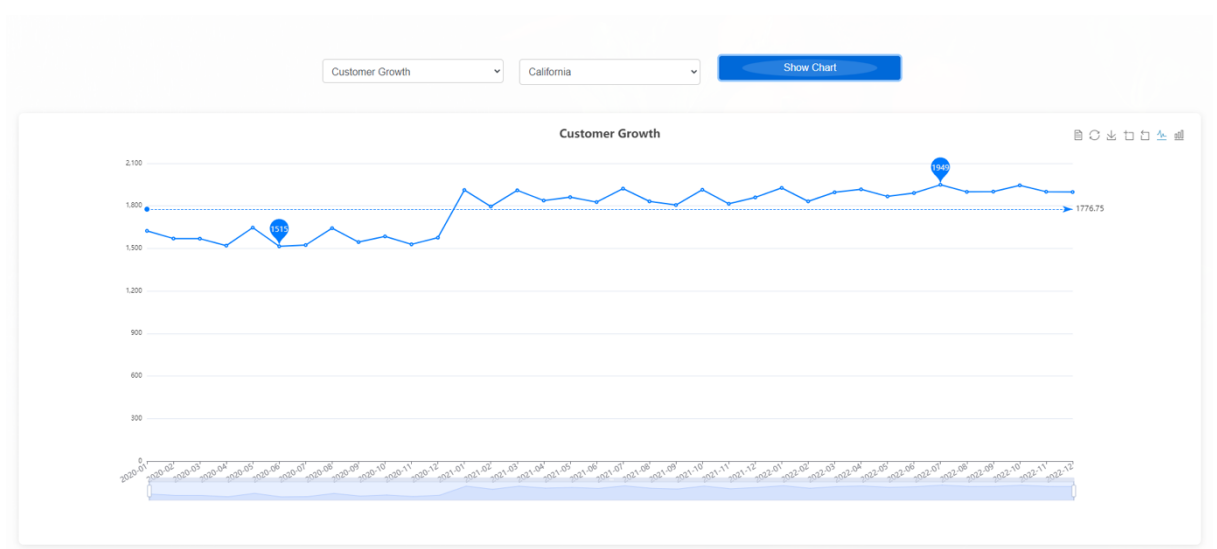


**Average Order Value (AOV) :** Calculates the average amount spent per order, providing insights into customer spending patterns and helping to shape pricing strategies and product bundling.

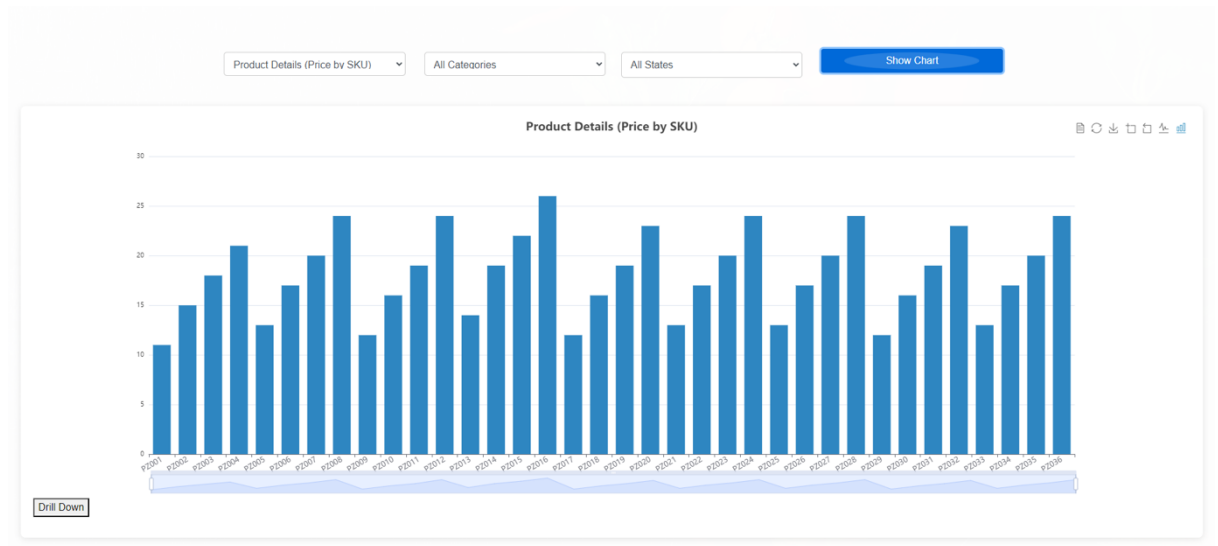


**Customer Growth:** Tracks the number of new customers acquired over time, which can be used to evaluate the effectiveness of marketing and customer acquisition efforts.

**Example:** The Manager/user would like to see the number of new Customers for the stores that are in California:



**Product Details:** Provides a comprehensive view of individual pizza products, including their SKU (unique identifier). This information is essential for menu management, inventory tracking, and understanding product performance.



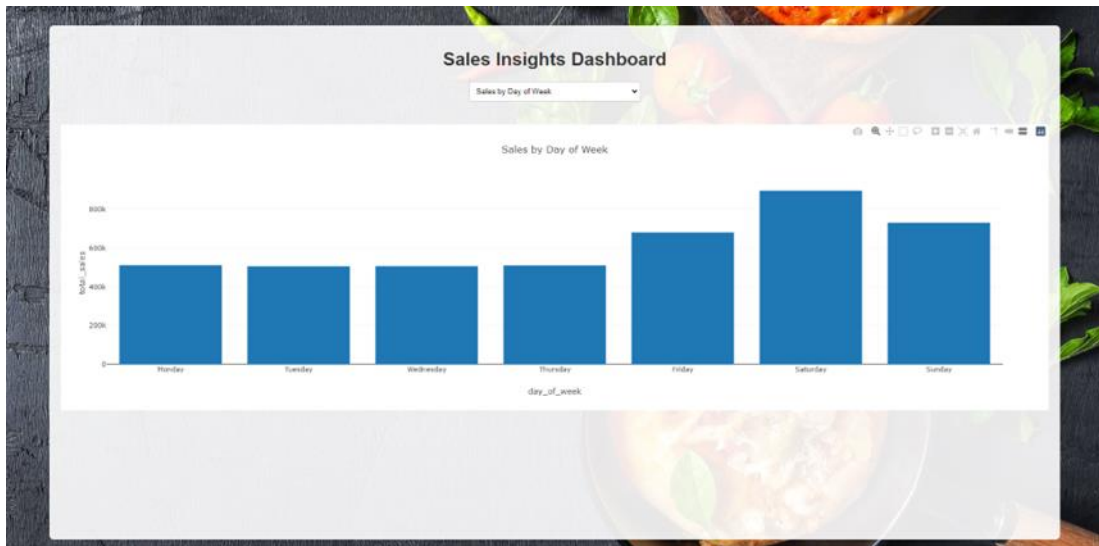
## 5.2 Insight Sales Analysis

When The User/Manager clicks on the Right top Button “The Insight Sales Analysis” a new Page with the chart for analysing sales will appear which provides a deeper look into sales performance and customer behaviour, helping managers and stakeholders make informed decisions. The dashboard includes several detailed analysis charts that address various aspects of sales and operations.

**Sales by Day of the Week** Displays the total sales generated on each day of the week. This helps identify the best and worst-performing days, which can inform staffing and promotional strategies.

**Endpoint:** /api/sales\_by\_day\_of\_week

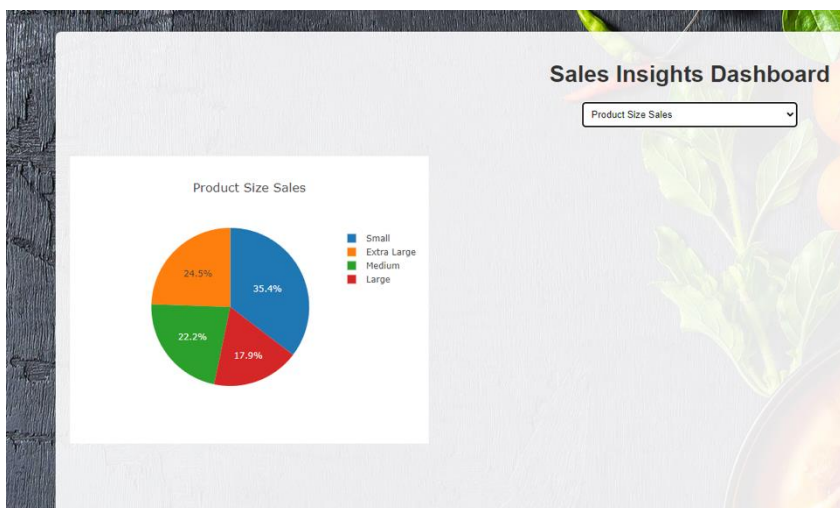
**Chart Type:** Bar Chart



**Product Size Sales:** Shows the total quantity of products sold by size. This information is useful for inventory management and understanding customer preferences regarding product sizes.

**Endpoint:** /api/product\_size\_sales

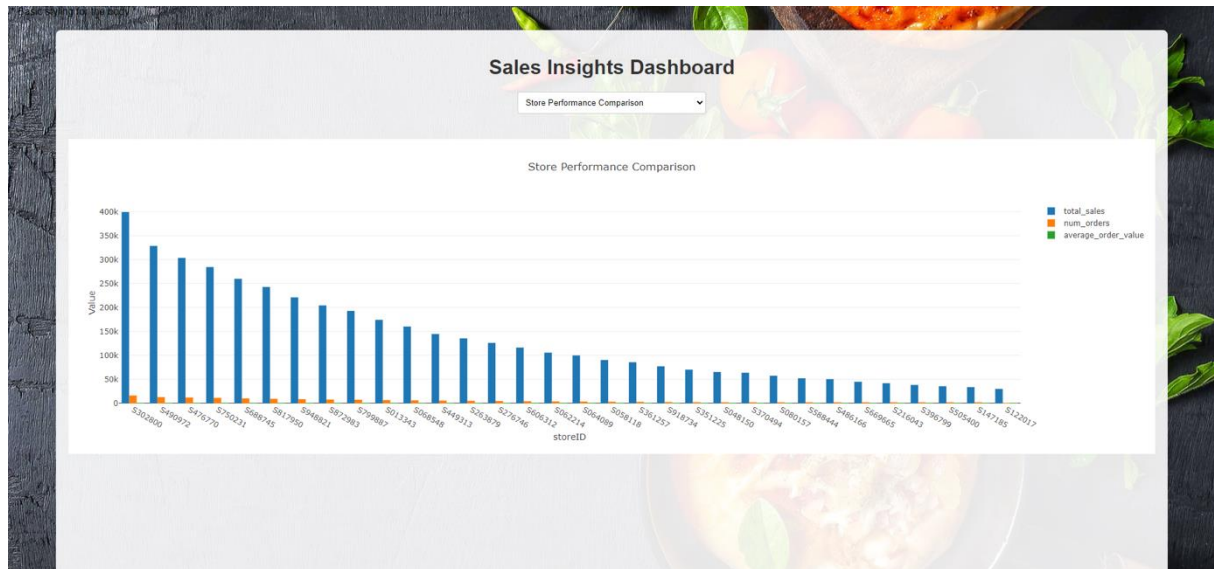
**Chart Type:** Pie Chart



**Store Performance Comparison:** Compares the performance of different stores based on total sales, number of orders, and average order value. This helps in benchmarking and identifying top and bottom-performing stores.

**Endpoint:** /api/filter\_store\_performance\_by\_category

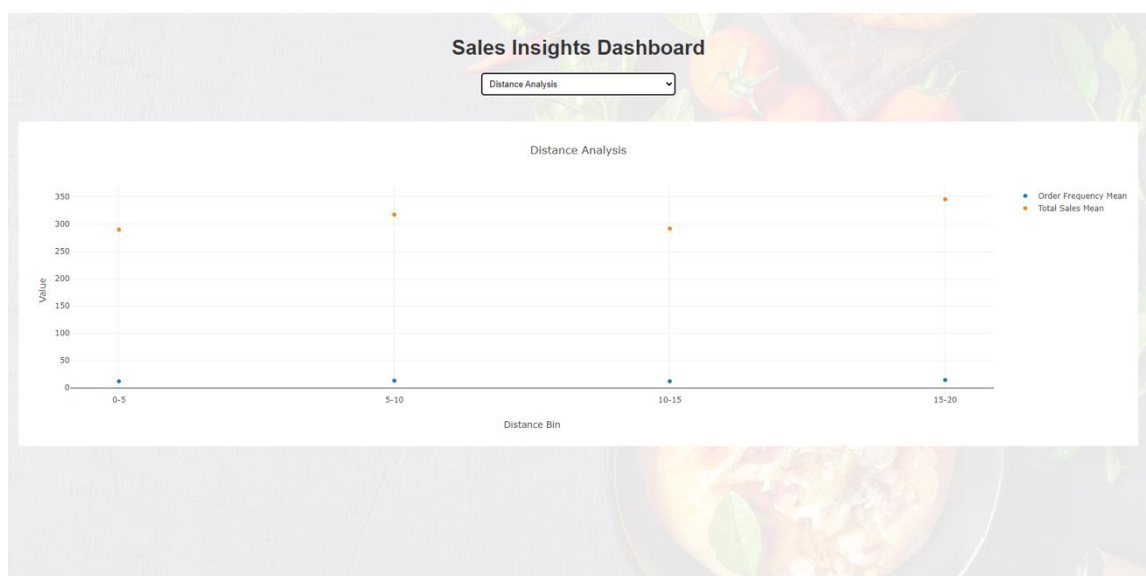
**Chart Type:** Stacked Bar Chart



**Distance Analysis:** Analyses the relationship between the distance of customers from the store and their order frequency and total sales. This can inform decisions about delivery zones and targeted marketing efforts.

**Endpoint:** /api/distance\_analysis

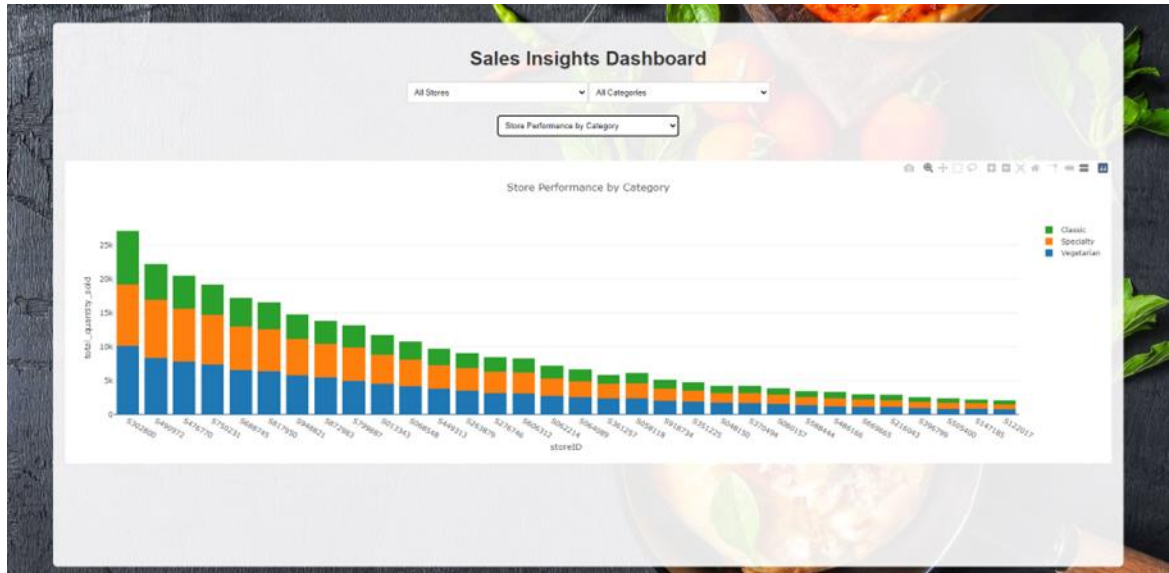
**Chart Type:** Scatter Plot



**Store Performance by Category:** Displays the total quantity of products sold by category for each store. This helps identify which product categories are performing well in various locations.

**Endpoint:** /api/store\_performance\_by\_category

**Chart Type:** Stacked Bar Chart

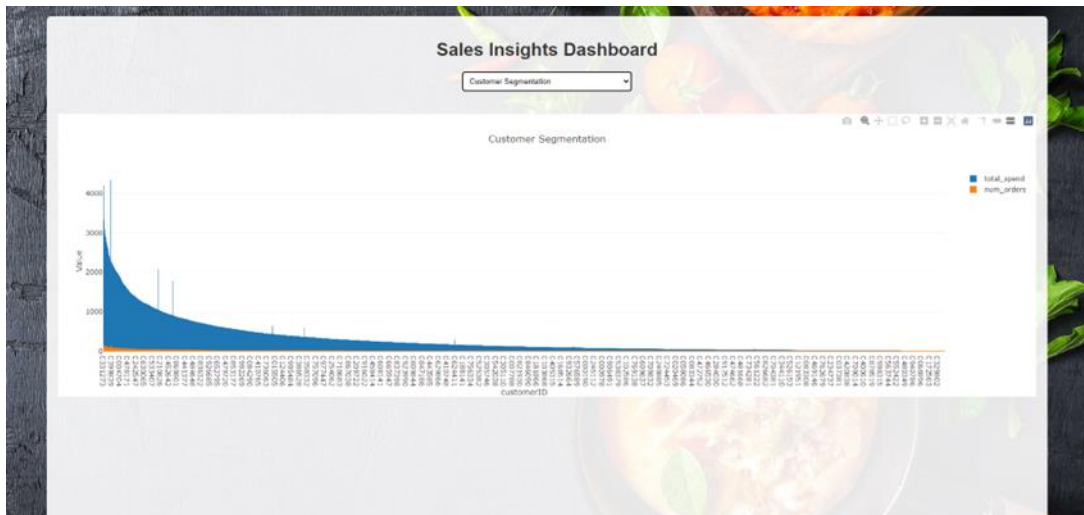


**Filter Ability for Store Performance by Category:** Allows users/managers to filter stores by their Store ID and see their performance by each category separately. It Provides a detailed and tailored view of store performance. The Endpoint of the Filter is : /api/filter\_store\_performance\_by\_category

**Customer Segmentation:** Segments customers based on their total spend, number of orders, and location. This helps in understanding customer demographics and tailoring marketing strategies.

**Endpoint:** /api/customer\_segmentation

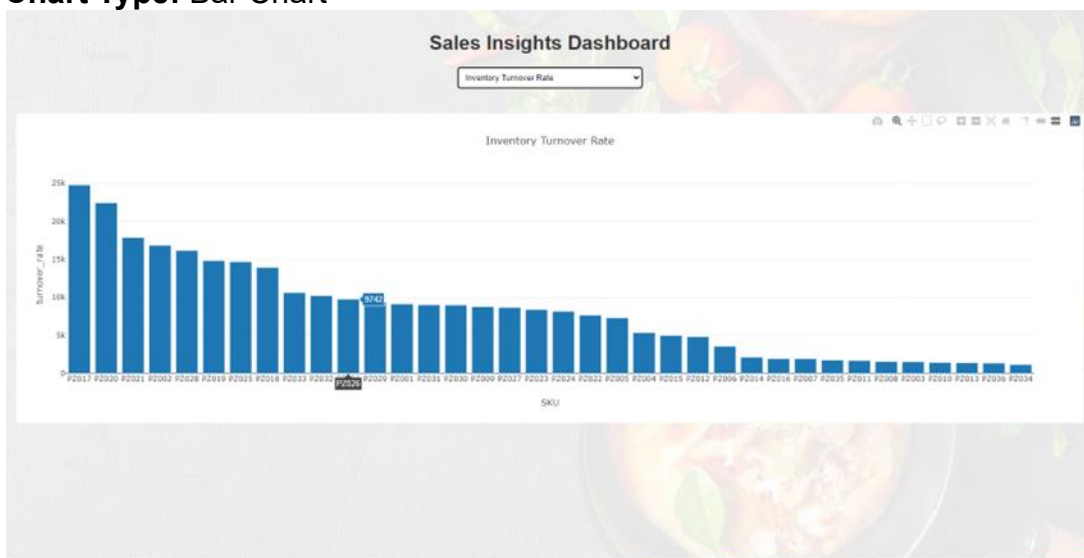
**Chart Type:** Clustered Bar Chart



**Inventory Turnover Rate:** Calculates the turnover rate of inventory by analysing the total quantity of products sold. This is crucial for efficient inventory management and identifying fast and slow-moving products.

**Endpoint:** /api/inventory\_turnover\_rate

**Chart Type:** Bar Chart



## 5.3 KPI Reports

The KPI Reports dashboard is designed to assist managers in tracking and analyzing essential business metrics. It provides a comprehensive view of critical Key Performance Indicators (KPIs).

### Frontend Implementation

The front-end interface for KPI Reports is designed to be user-friendly and interactive. Key features include:

- **Dropdown Menu:** Allows users to select the KPI they wish to view. The options include Revenue Growth, New vs. Returning Customers, Popular Products, Conversion Rate, and Order Frequency.
- **Plot Area:** Displays the selected KPI chart. The plot area dynamically updates based on the user's selection from the dropdown menu.
- **Loading Indicator:** A spinner is shown while data is being fetched from the server. This provides visual feedback that data is being processed.
- **Responsive Design:** The layout is designed to be responsive and visually appealing across various devices and screen sizes.

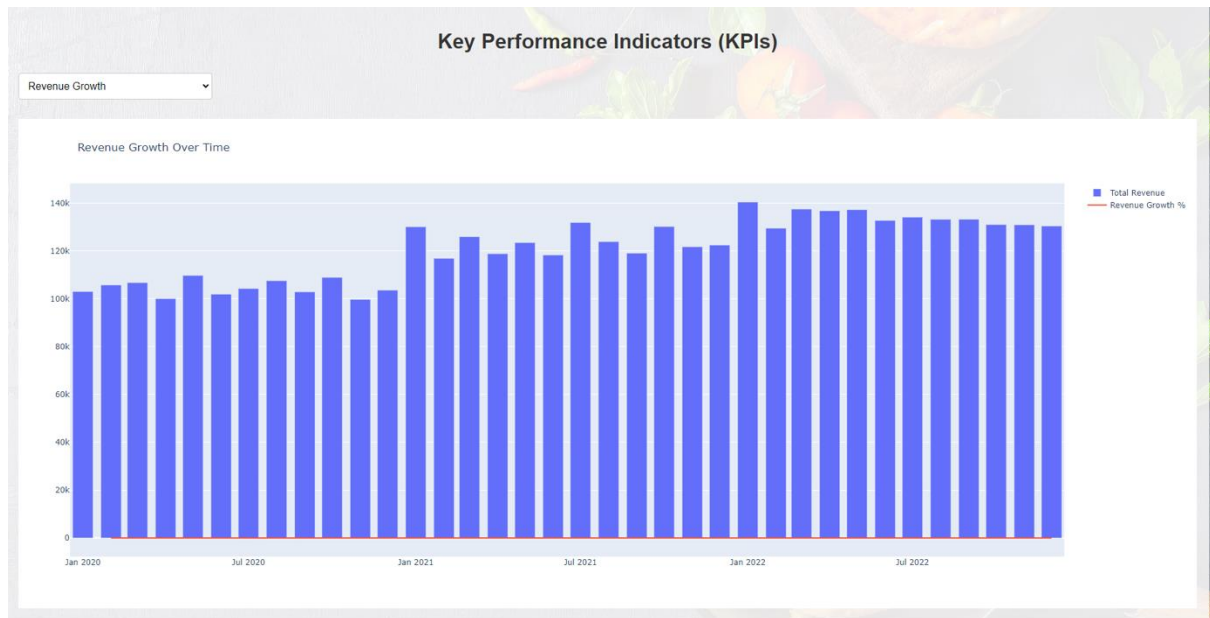
The explanation of the **backend side** of the code for each chart will be described alongside the charts and these are:

**Revenue Growth:** Shows how much revenue the business is making and how fast it's growing over time. This helps The Manager to understand if the business is doing well financially and how quickly it's expanding.

**Endpoint:** /api/revenue\_growth

**Chart Type:** Line and Bar chart





**New vs. Returning Customers:** Shows the proportion of new customers compared to returning customers. This metric helps in understanding customer loyalty and the effectiveness of acquisition strategies.

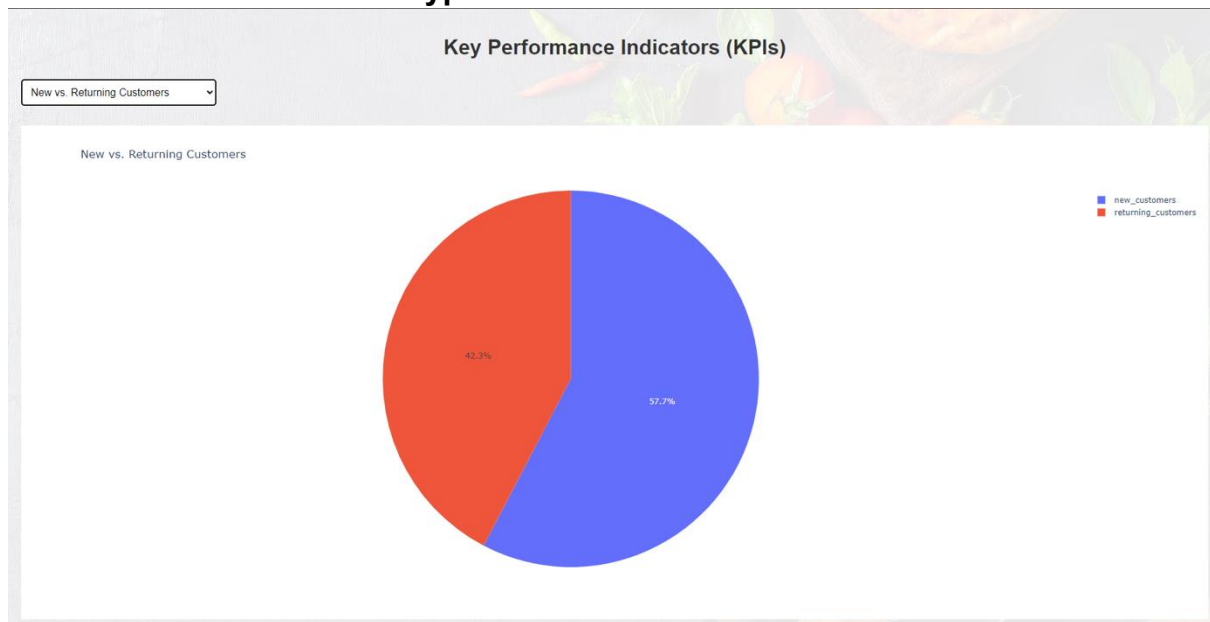
**Endpoint:** /api/new\_vs\_returning\_customers

**Chart**

**Type:**

Pie

cha



**Popular Products:** Displays the most popular products based on order frequency. This information is useful for inventory management and marketing strategies.

**Endpoint:** /api/popular\_products

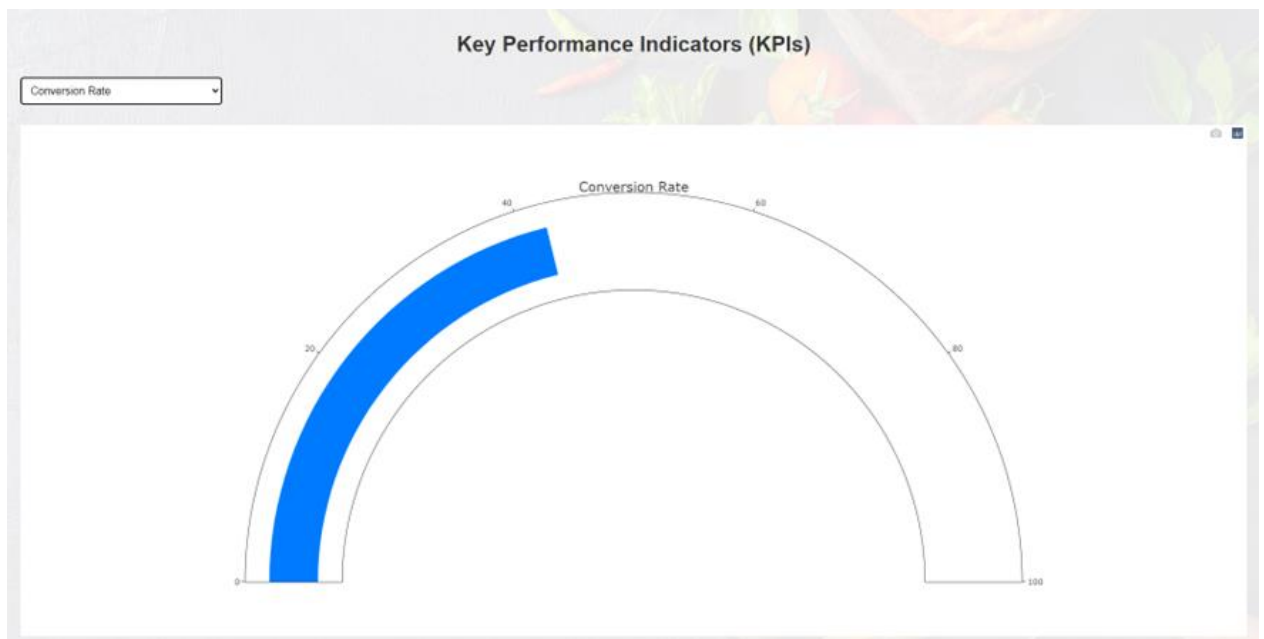
**Chart Type:** Bar chart



**Conversion Rate:** Measures the percentage of customers who have made a purchase out of the total number of visitors. This KPI is crucial for evaluating the effectiveness of sales and marketing efforts.

**Endpoint:** /api/conversion\_rate

**Chart Type:** Gauge chart



**Order Frequency:** Identifies the customers who order most frequently. This metric helps in understanding customer loyalty and can inform targeted marketing efforts.

**Endpoint:** /api/order\_frequency

**Chart Type:** Bar chart



## 6 Number based analysis

### 6.1 Implementation of the "Show Revenue" Option

To implement the "Show Revenue" option, I undertook a comprehensive approach that involved both backend and frontend development. On the backend, I utilized Flask, a Python web framework, to create an API endpoint dedicated to fetching revenue data from the database. This endpoint executes a query on the `revenue_data` table, extracting essential information such as the year, quarter, and total revenue. The data is then formatted into JSON, a format that is easily consumable by the frontend.

For the frontend, I designed a user interface component that includes a button labeled "Show Revenue" and a table structure for displaying the retrieved data. When the button is clicked, it triggers an API call to the backend endpoint. Upon receiving the data, the frontend dynamically populates the table, presenting the revenue information in a clear and organized manner. This interaction ensures that managers can view the latest revenue data with just a single click. As a user one can chose the year and the quarter of the two different years in order to show it on the chart. [Chart 1-2]

### 6.2 Rationale Behind the Implementation

The primary reason for implementing the "Show Revenue" option was to enhance data visibility and accessibility for managers. By providing a straightforward way to access revenue data, managers can monitor the company's financial performance with ease. Breaking down the revenue by year and quarter allows for a detailed analysis of financial trends over time, enabling managers to make informed decisions.

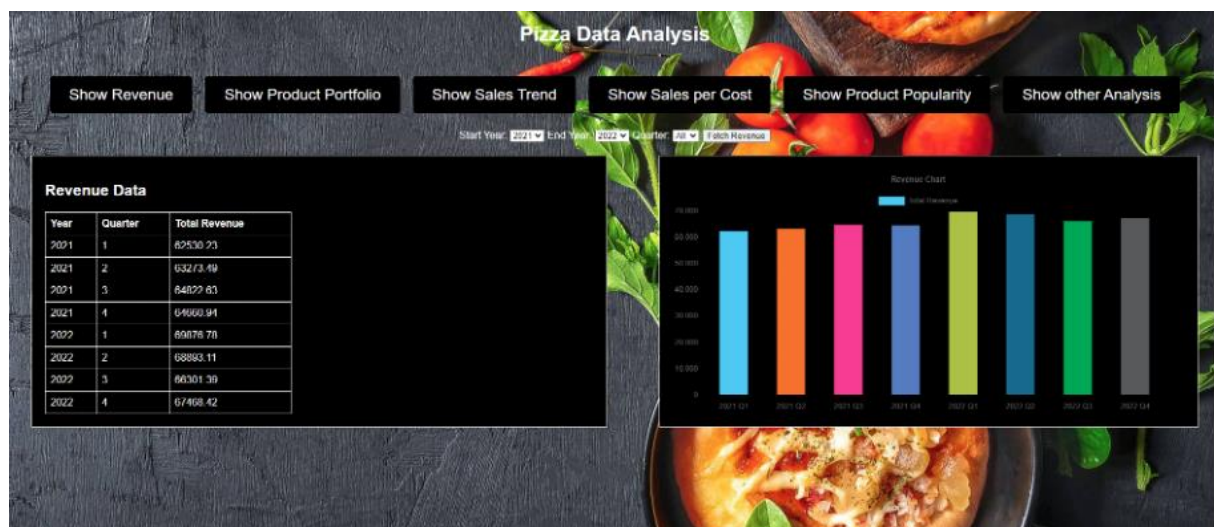
This feature also supports strategic decision-making by giving managers the tools to identify successful strategies and replicate them. Conversely, it helps in recognizing periods of lower performance, prompting further analysis and necessary corrective actions. The automated nature of this feature saves valuable time, reducing the need for manual data gathering and processing, thus allowing managers to focus more on analysis and strategy.

### 6.3 Benefits for Managerial Decision-Making

The "Show Revenue" option significantly simplifies decision-making for managers in several ways. Firstly, it provides immediate access to crucial revenue data without the need for complex database queries or extensive reports. This immediacy ensures that decisions are based on the most current data available.

Secondly, the revenue data is displayed in a clear and concise table format, making it easy to interpret and analyze. This visual presentation helps managers quickly identify trends, anomalies, and key performance indicators. By having the revenue data segmented by year and quarter, managers can perform trend analysis and identify seasonal patterns, supporting more accurate forecasting and strategic planning.

Lastly, the automation of data retrieval and display processes saves time, allowing managers to focus on critical business activities. This efficiency enhances overall productivity and enables quicker responses to emerging business needs. In summary, the "Show Revenue" option provides managers with a powerful, efficient, and user-friendly tool to monitor and analyze the company's financial performance, facilitating more informed, timely, and effective decision-making.



[Chart

1]



[Chart 2]

## 6.4 Implementation of the "Show Product Portfolio" Option

To implement the "Show Product Portfolio" option, I designed a solution that integrates both backend and frontend components seamlessly.

**Backend Development:** I started by creating an API endpoint using Flask, a popular Python web framework. This endpoint is designed to fetch product portfolio data from the database. Specifically, it queries the `product_data` table to retrieve information about each product, including the product name and the total quantity sold. The data is then formatted into JSON, ensuring it can be easily consumed by the frontend.

**Frontend Development:** On the frontend, I added a button labeled "Show Product Portfolio" to the user interface. When clicked, this button initiates a request to the backend to retrieve the product portfolio data. I also designed a table and a chart to display this data. The table lists each product alongside its total quantity sold, while the chart provides a visual representation of the same data. In addition one can have the options of choosing between Pie chart, Line chart and Bar chart on the option button. Each provided chart could identify a different view of the product by which one can easily understand the analyzed data. [Chart 3-5]

As one clicked on the one of the bars on the bar chart, the chart will direct create another chart with the specific information "sale distributions" of that specific product. [Chart 6]

When the "Show Product Portfolio" button is clicked, it triggers an API call to the backend endpoint. The frontend then dynamically populates the table and chart with the received data, presenting the product portfolio in both text and graphical formats. This dual representation ensures that managers can easily interpret and analyze the information.

## 6.5 Rationale Behind the Implementation

The "Show Product Portfolio" option was implemented to provide managers with a comprehensive overview of the sales performance of different products. By detailing the total quantity sold for each product, this feature helps managers understand which



products are performing well and which are not. This information is crucial for making informed decisions about inventory management and marketing strategies.

Understanding product performance allows managers to identify top-selling products that drive revenue, as well as underperforming products that may need additional marketing efforts or strategic adjustments. This data is also vital for inventory management, ensuring that high-demand products are adequately stocked to meet customer needs.

## 6.6 How This Option Makes Decision-Making Easier for Managers

The "Show Product Portfolio" option simplifies decision-making for managers by providing comprehensive data access and clear visualization. Managers can immediately access detailed sales data for each product without the need for manual data extraction and compilation. This immediate access to information ensures that decisions are based on the most current and relevant data.

The combination of a table and a chart offers both detailed and visual representations of the product sales data. This dual approach caters to different analytical preferences and helps managers grasp the information quickly. Visual trends and patterns in the chart highlight the relative performance of products, aiding in quick comparative analysis.

With the total quantity sold for each product displayed clearly, managers can easily identify top-selling and underperforming products. This insight supports targeted decision-making and strategic adjustments, allowing managers to tailor marketing strategies, optimize product offerings, and improve overall sales performance.

Moreover, the automated process of fetching and displaying product portfolio data saves significant time and effort for managers. This efficiency allows them to focus more on strategic planning and less on data management tasks, enhancing overall productivity and effectiveness.



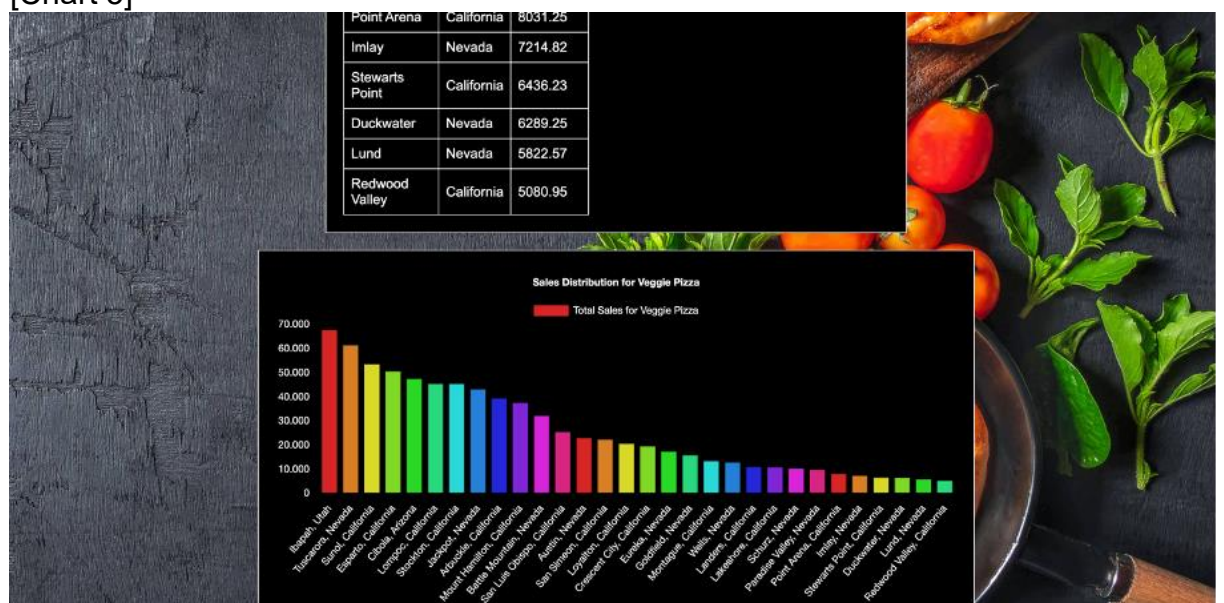
[Chart 3]



[Chart 4]



[Chart 5]



[Chart 6]

## 6.7 Implementation of the "Show Sales Trend" Option

To implement the "Show Sales Trend" option, I developed a seamless integration of both backend and frontend components.

**Backend Development:** The process began by creating an API endpoint using Flask, a robust Python web framework. This endpoint is designed to fetch sales trend data from the database. It queries the `sales_data` table to retrieve monthly sales figures for each product, formatting the data into JSON for easy consumption by the frontend.

**Frontend Development:** On the frontend, I added a button labeled "Show Sales Trend" to the user interface. This button, when clicked, sends a request to the backend to retrieve the sales trend data. To display this data, I designed a dynamic chart that visually represents the sales trends over time. [Chart 7]

When the "Show Sales Trend" button is clicked, it triggers an API call to the backend endpoint. The frontend then dynamically populates the chart with the received data, showcasing the sales trends for each product over time. This visual representation allows managers to easily interpret and analyze the data.

## 6.8 Rationale Behind the Implementation

The "Show Sales Trend" option was implemented to provide managers with a detailed analysis of sales trends over time. By displaying monthly sales data for each product, this feature helps managers monitor sales performance, evaluate marketing effectiveness, and plan inventory and production efficiently.

Understanding sales trends is crucial for identifying how product sales fluctuate throughout the year. This information allows managers to make informed strategic decisions. It also helps in assessing the impact of marketing campaigns, determining the best times to launch promotions, and ensuring that inventory levels meet future demand forecasts.

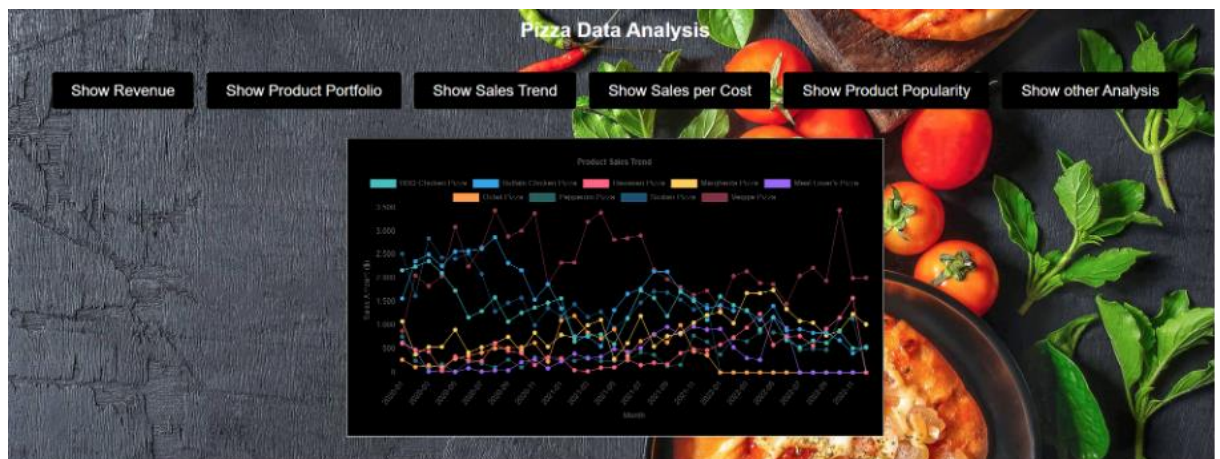
## 6.9 How This Option Makes Decision-Making Easier for Managers

The "Show Sales Trend" option significantly simplifies decision-making for managers by providing a comprehensive view of sales performance over time:

1. **Immediate Data Access:** Managers can quickly access up-to-date sales trend data without the need to navigate through complex reports or databases. This immediate access ensures that decisions are based on the latest available information.
2. **Clear Visualization:** The sales trend chart provides a visual representation of sales data, making it easy to identify trends, spikes, and declines in sales. Visualizing data over time helps managers quickly grasp changes in sales performance and react accordingly.



3. **Data-Driven Insights:** By analyzing sales trends, managers can identify the most and least popular products over different periods. This analysis supports targeted decision-making, such as adjusting marketing strategies, planning inventory, and scheduling promotions.
4. **Strategic Planning:** Historical sales trends provide valuable insights that inform future business strategies. Managers can use this data to anticipate market demands, optimize product offerings, and improve overall business performance.
5. **Operational Efficiency:** Automating the retrieval and display of sales trend data saves significant time and effort for managers. This efficiency allows them to focus more on strategic planning and less on data management tasks, enhancing overall productivity and effectiveness.



[Chart 7]

## 6.10 Implementation of the "Show Product Popularity" Option

To implement the "Show Product Popularity" option, I meticulously integrated both backend and frontend components to ensure a seamless user experience.

**Backend Development:** I started by creating an API endpoint using Flask, a powerful Python web framework. This endpoint is designed to fetch product popularity data from the database. It queries the `product_popularity` table to retrieve the product name, total quantity sold, and popularity rating. The data is then formatted into JSON, making it easy for the frontend to process and display.

**Frontend Development:** On the frontend, I added a button labeled "Show Product Popularity" to the user interface. When clicked, this button initiates a request to the backend to retrieve the product popularity data. I also designed a table to display this data. The table lists each product alongside its total quantity sold and its popularity rating.

When the "Show Product Popularity" button is clicked, it triggers an API call to the backend endpoint. The frontend then dynamically populates the table with the received data, presenting the product popularity in a clear and organized manner. This

presentation allows managers to easily interpret and analyze the popularity of each product. [Chart 8]

## **6.11 Rationale Behind the Implementation**

The "Show Product Popularity" option was implemented to provide managers with a comprehensive understanding of which products are most popular among customers. By displaying the total quantity sold and popularity rating for each product, this feature helps managers:

### **1. Identify Top-Selling Products:**

- Recognize the products that are driving the most sales.
- Understand customer preferences and trends.

### **2. Enhance Marketing Strategies:**

- Focus marketing efforts on popular products to maximize sales.
- Develop targeted promotions to boost the popularity of underperforming products.

### **3. Improve Inventory Management:**

- Ensure that popular products are adequately stocked to meet customer demand.
- Avoid overstocking products with lower popularity, reducing inventory costs.

## **6.12 How This Option Makes Decision-Making Easier for Managers**

The "Show Product Popularity" option significantly simplifies decision-making for managers by providing clear insights into customer preferences and product performance:

### **1. Immediate Data Access:**

- Managers can quickly access up-to-date popularity data without needing to navigate through complex reports or databases.
- This immediate access ensures that decisions are based on the latest available information.

### **2. Clear Visualization:**

- The product popularity table provides a straightforward representation of sales data, making it easy to identify top-performing and underperforming products.
- Visualizing data with a popularity rating helps managers quickly grasp customer preferences.

### **3. Data-Driven Insights:**

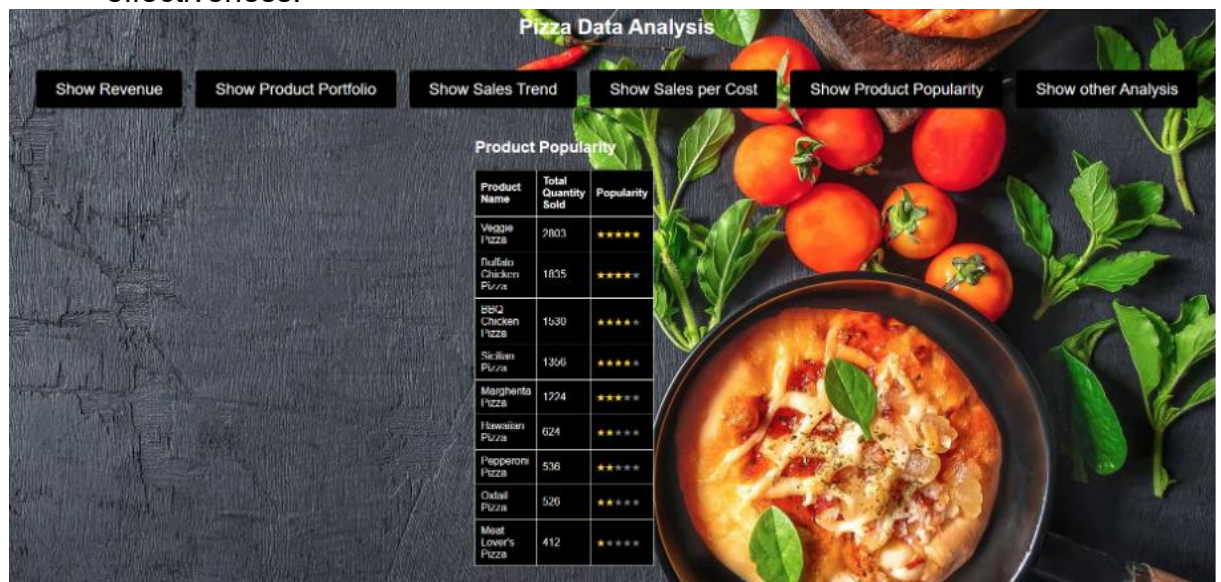
- By analyzing product popularity, managers can identify trends and patterns in customer behavior.
- This analysis supports targeted decision-making, such as adjusting marketing strategies, optimizing product offerings, and planning inventory levels.

### **4. Strategic Planning:**

- Understanding which products are most popular allows managers to plan future product launches and promotional campaigns more effectively.
- Managers can use this data to align product offerings with customer preferences and market trends.

#### 5. Operational Efficiency:

- Automating the retrieval and display of product popularity data saves significant time and effort for managers.
- This efficiency allows them to focus more on strategic planning and less on data management tasks, enhancing overall productivity and effectiveness.



[Chart

8]

### 6.13 Implementation of the "Show Sales per Cost" Option

To implement the "Show Sales per Cost" option, I meticulously integrated both backend and frontend components to provide a seamless and efficient user experience.

**Backend Development:** I began by creating an API endpoint using Flask, a versatile Python web framework. This endpoint is designed to fetch sales data categorized by product cost from the database. It queries the `sales_data` table to retrieve information about each product's cost and corresponding sales figures. The data is then formatted into JSON to ensure easy consumption by the frontend.

**Frontend Development:** On the frontend, I added a button labeled "Show Sales per Cost" to the user interface. When clicked, this button sends a request to the backend to retrieve the sales data. Additionally, I designed a dynamic chart to display this data, categorizing sales figures by product cost for clear and easy analysis.

When the "Show Sales per Cost" button is clicked, it triggers an API call to the backend endpoint. The frontend then dynamically populates the chart with the received data, presenting the sales figures categorized by product cost in a clear and organized manner. This visual representation allows managers to easily interpret and analyze the data related to product costs and sales. [Chart 9]

## 6.14 Rationale Behind the Implementation

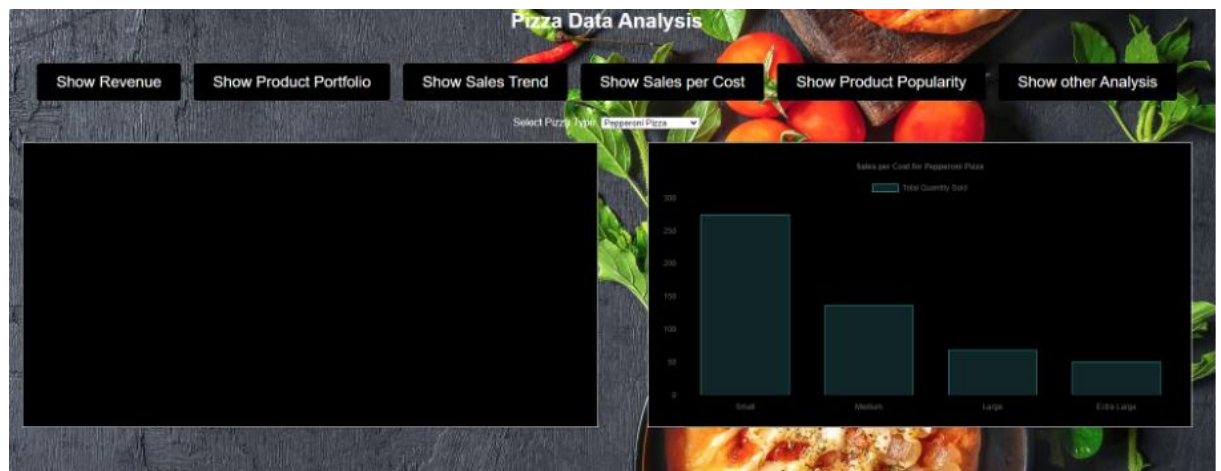
The "Show Sales per Cost" option was implemented to provide managers with detailed insights into how sales figures relate to product costs. By displaying sales data categorized by cost, this feature helps managers analyze their pricing strategy, optimize revenue, and plan inventory and production more effectively.

Understanding the relationship between product pricing and sales performance is crucial for identifying which price points are driving the most sales and which are underperforming. This information allows managers to adjust their pricing strategies based on actual sales data to maximize revenue. Additionally, having insights into the optimal price points helps in balancing profitability and customer demand.

## 6.15 How This Option Makes Decision-Making Easier for Managers

The "Show Sales per Cost" option simplifies decision-making for managers by providing clear insights into the financial performance of products at different price points:

1. **Immediate Data Access:** Managers can quickly access up-to-date sales data categorized by product cost without the need for complex reports or database queries. This immediate access ensures that pricing decisions are based on the most current and relevant data.
2. **Clear Visualization:** The sales per cost chart provides a visual representation of sales data, making it easy to identify trends, spikes, and declines in sales performance across different price points. Visualizing data in this manner helps managers quickly understand the impact of pricing on sales.
3. **Data-Driven Insights:** By analyzing sales data categorized by cost, managers can identify which price points are most effective in driving sales. This analysis supports targeted decision-making, such as adjusting pricing strategies, optimizing product offerings, and planning inventory levels.
4. **Strategic Planning:** Historical sales data categorized by cost provides valuable insights that inform future pricing strategies. Managers can use this data to anticipate market demands, set competitive prices, and improve overall business performance.
5. **Operational Efficiency:** Automating the retrieval and display of sales per cost data saves significant time and effort for managers. This efficiency allows them to focus more on strategic planning and less on data management tasks, enhancing overall productivity and effectiveness.



[Chart 9]

## 6.16 Challenges and Solutions

During the development of the Pizza Data Analysis project, we encountered several challenges, particularly related to data quality and technical integration. Here's a summary of these challenges and how we addressed them.

### 6.16.1 Data Quality Issues

One of the primary challenges we faced was ensuring the accuracy and completeness of the data.

1. **Customers on the Sea:** We discovered that some customer geolocation data was inaccurate, resulting in customers being plotted on the sea.
  - **Solution:** for this problem we couldn't find any solutions because we could not manipulate the data which we have got from our professor.
2. **Two Restaurants on Each Other on the Map:** Another issue was the overlapping geolocation data for restaurants, which caused two restaurants to appear at the same location on the map.
  - **Solution:** for this problem we couldn't find any solutions because we could not manipulate the data which we have got from our professor.
3. **Inconsistent Column Naming:** We found inconsistencies in the data schema, such as some column names being in all capital letters, which could lead to errors in data processing.
  - **Solution:** for this problem we couldn't find any solutions because we could not manipulate the data which we have got from our professor.

### 6.16.2 Technical Challenges

Integrating multiple tools and technologies presented another set of challenges.

1. **Integrating Pandas:** Pandas was used for data manipulation and analysis, but integrating it with the Flask backend and ensuring efficient data processing was challenging due to the large datasets.
  - **Solution:** We optimized the data processing workflows by leveraging Pandas' powerful data manipulation capabilities. This included using vectorized operations and avoiding loops where possible, ensuring efficient handling of large datasets.
2. **Creating Dynamic Charts with Chart.js and D3.js:** Developing dynamic and interactive charts using Chart.js and D3.js was essential, but it posed integration challenges, especially ensuring that the charts updated correctly with real-time data.
  - **Solution:** We used asynchronous JavaScript (AJAX) to fetch data dynamically from the Flask API endpoints. This approach allowed the charts to update in real-time without requiring a full page reload, providing a smooth and responsive user experience.

## 6.17 Comprehensive Solutions

To address these challenges effectively, we implemented several key solutions:

1. **Data Validation and Cleaning:** We established a robust data validation pipeline to identify and correct inaccuracies, standardizing data formats such as geolocation coordinates and column names to ensure consistency and reliability.
2. **Efficient Data Processing:** We leveraged Pandas for efficient data manipulation and analysis, implementing performance optimizations to handle large datasets smoothly.
3. **Real-Time Data Visualization:** We integrated Chart.js and D3.js for creating dynamic and interactive charts, using AJAX for real-time data fetching and updating visualizations seamlessly.
4. **Standardization and Best Practices:** We applied consistent naming conventions and data schemas across the project, following best practices in coding and data management to enhance maintainability and scalability.

By addressing these challenges with targeted solutions, we successfully delivered a reliable and user-friendly system for data analysis and visualization. This ensured that managers and stakeholders were equipped with accurate, real-time insights to drive informed decision-making and strategic planning.

## 6.18 Lessons Learned

During the development of the Pizza Data Analysis project, several lessons were learned that will inform future projects. Here are the key takeaways:

## 6.19 Improvements for Next Time

**Enhanced Data Validation and Cleaning Processes:** In future projects, I would implement more rigorous data validation and cleaning processes from the beginning. Automated scripts to detect and correct inaccuracies, such as geolocation errors or inconsistent data formats, would be put in place early on. This would prevent issues



like customers plotted on the sea or overlapping restaurant locations, ensuring more accurate and reliable analysis.

**Scalable System Architecture:** Designing the system architecture with scalability in mind from the outset would be another priority. Utilizing a microservices architecture and containerization technologies like Docker would help manage increased loads and simplify maintenance. A scalable architecture would better accommodate future growth, ensuring the system remains responsive and efficient as the volume of data and number of users increase.

**User Experience (UX) Enhancements:** A greater emphasis on user experience design would also be a focus. Incorporating more user feedback into the development process through iterative testing and refinement of the interface would make the tool more intuitive and easier to use, improving overall user satisfaction and productivity.

**Advanced Analytics Integration:** Integrating more advanced analytics capabilities, such as machine learning models for predictive analytics, right from the start would provide deeper insights and more powerful predictive capabilities. This would help managers make even more informed decisions by incorporating automated forecasting and anomaly detection.

## 6.20 Reflections on Frameworks and Tools

**Python and Pandas:** I found Python, combined with Pandas, to be extremely powerful for data manipulation and analysis. The extensive libraries and ease of use allowed for efficient processing of large datasets. The flexibility and functionality provided by these tools facilitated rapid development and robust data analysis.

**Flask:** Flask was an excellent choice for backend development. Its lightweight nature and simplicity made it easy to set up and integrate with other tools, while still being powerful enough to handle complex tasks. I appreciated Flask's straightforward and modular approach, which streamlined API development and made the codebase easy to manage.

**JavaScript and Chart Libraries (Chart.js, D3.js):** JavaScript, along with Chart.js and D3.js, enabled the creation of dynamic and interactive visualizations. These tools offered extensive customization options and were effective for real-time data updates. I found these tools highly effective for building responsive and visually appealing charts that significantly enhanced data visualization.

## 6.21 However, there were also some challenges:

**Data Inconsistencies and Cleaning:** Handling data inconsistencies was a significant challenge. The effort required to clean and standardize data was time-consuming and sometimes frustrating. While Pandas was effective for data cleaning, the initial inconsistency in data formats posed a substantial hurdle, highlighting the need for better initial data collection practices.

**Complexity of D3.js:** D3.js, while powerful, has a steep learning curve and can be complex to implement for more intricate visualizations. Although D3.js offered extensive customization, its complexity sometimes slowed down development and required a significant investment of time to achieve the desired results.

**Scalability Limitations of Flask:** Flask, being lightweight, has limitations in scalability for larger applications without additional configurations and enhancements. While Flask was excellent for rapid development, scaling the application required additional tools and configurations, which added to the complexity.

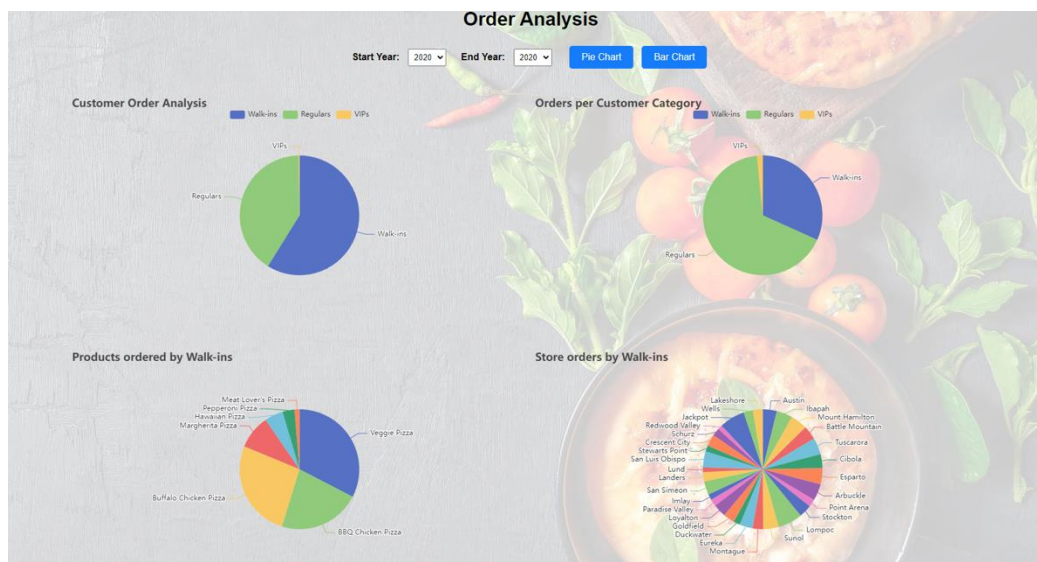
## 6.22 Summary

The development of the Pizza Data Analysis project provided valuable insights into both the strengths and challenges of various frameworks and tools. Python, Pandas, Flask, and JavaScript libraries facilitated robust data analysis and dynamic visualizations. However, the project also highlighted areas for improvement, such as data validation processes, system scalability, and user experience design. These lessons learned will inform future projects, ensuring more efficient development and enhanced functionality.

## 7 Order Based Analysis

This page is part of the comprehensive business analysis web application. It focuses on the different types of customers and their orders with which an analysis of their behavior is possible.

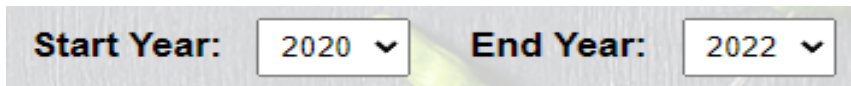
### 7.1 Order Based Analysis overview



The Order Based Analysis page is designed to provide detailed information about different types of customers the stores have. The charts show the different types of customers, their orders, the products they ordered and where they ordered.

## 7.2 User Controls

### 7.2.1 Data period selection



Start Year: 2020 End Year: 2022

The manager can select the time period for which he wants to see the data, the years which are possible to choose from are 2020, 2021 and 2022 although the start year can not be smaller than the end year.

### 7.2.2 Chart type selection for customer order analysis + Orders per customer category

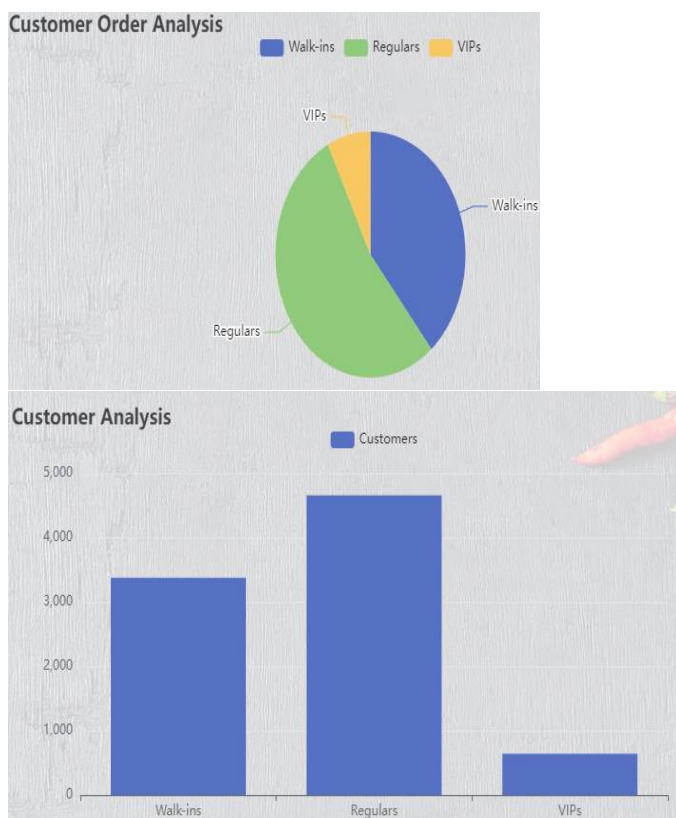


Pie Chart Bar Chart

Depending on which kind of chart is needed, it is possible after selecting the time period to get a bar chart or a pie chart.

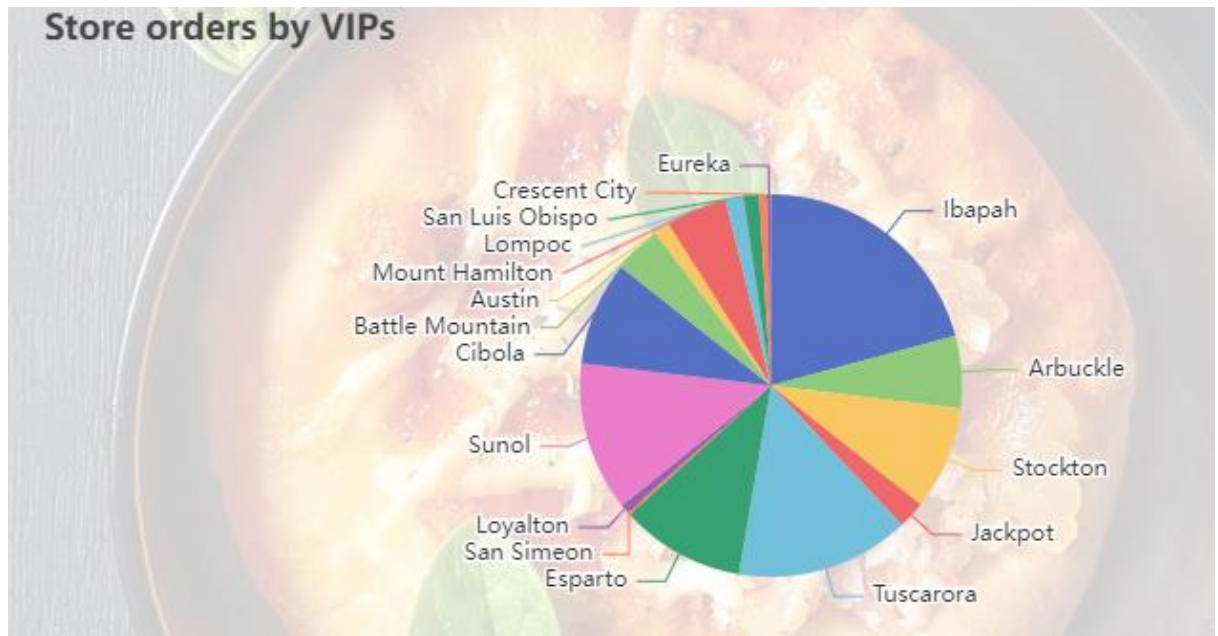
## 7.3 Business analysis charts

### 7.3.1 Customer order analysis



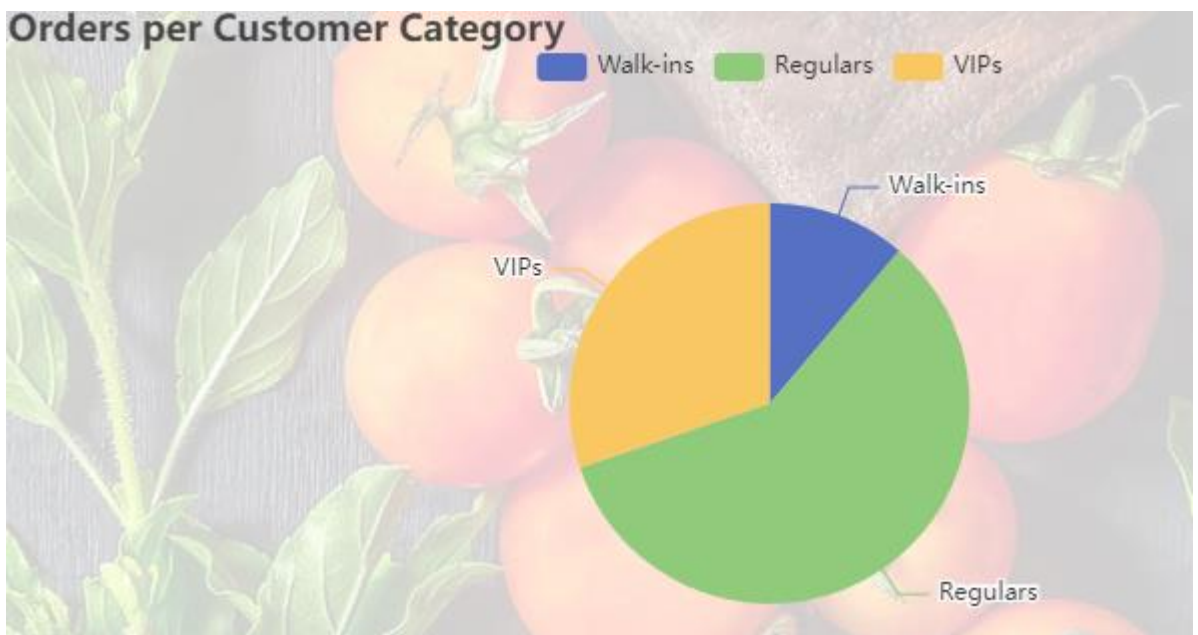
The Customer Order Analysis chart shows how many customers of each category there are and the percentage they have. It is possible to select what kind of chart should be shown like a bar chart or a pie chart. This chart is interactive which means that clicking on the for-example VIP category will create a new pie chart containing information about where and how much those customers ordered.

### 7.3.2 Stores by category chart



The stores by category chart reacts to the clicked category on the customer analysis chart. It shows the exact stores and how many orders were made in those specific stores. With this data it is possible to see which stores attract which types of customers and to find out why those customers visit these stores and why not others.

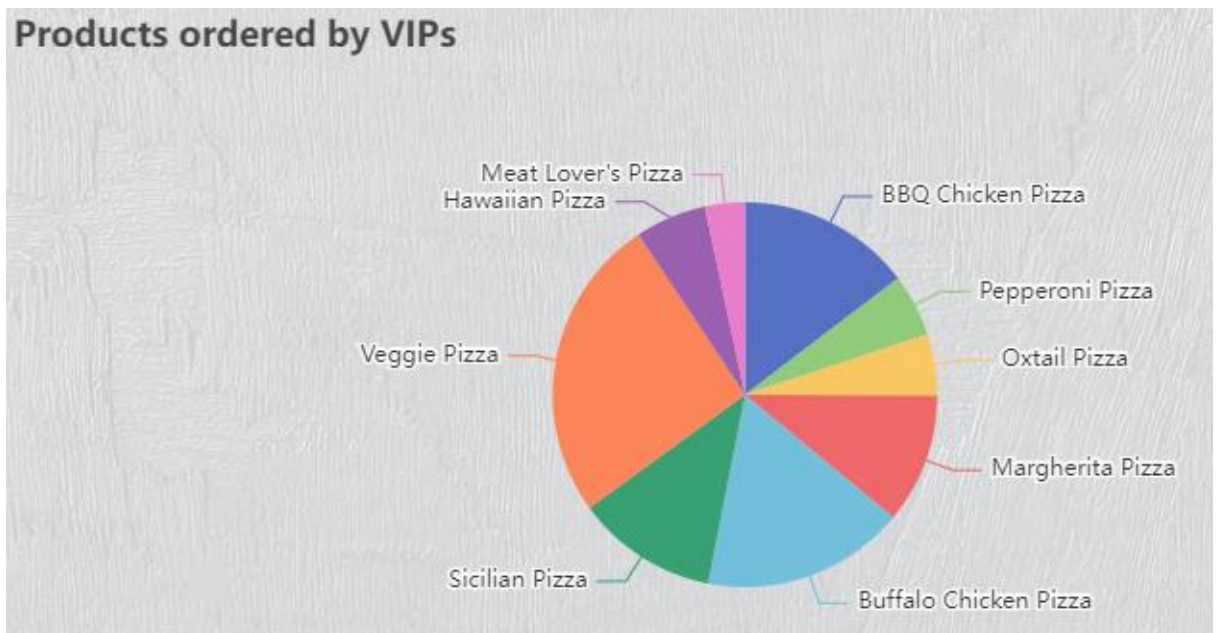
### 7.3.3 Orders per Customer Category



The orders per customer category chart shows the percentage and exact count of orders each category did. It is also possible to select a bar chart or pie chart. This chart is important for the analysis because the manager can see that he has for example 3 times less VIPs than regulars but the VIPs have nearly 3 times more orders than the walk-ins so it would be better to focus on the VIP customers. This chart is also interactive, meaning clicking for example on the VIP section will open a new pie chart containing all the products the VIPs ordered in the selected time period.



### 7.3.4 Products per customer category



The products per category charts show which kind of pizza a selected category ordered, how many of them and the percentage of all the pizzas that were ordered. With this information it is possible for the manager to make different campaigns for each category depending on the pizzas they prefer.

## 7.4 Problems and solutions

### 7.4.1 Getting all the correct data from the database

- **Problem:** More orders for a customer category than products ordered.
- **Solution:** Delete tables from database with incomplete data and create them again with correct and complete data.

### 7.4.2 Correct data input

- **Problem:** If the start year is smaller than the end year selected the site would stop to work.
- **Solution:** Fetch error when end year is smaller than start year and show an error on the site, so the user selects an end year that is like the start year or bigger.



### 7.4.3 Make only one chart interactive instead of two

- **Problem:** Only the orders per customers category chart was interactive and should create both store/products per category chart which led to collapsing the whole application.
- **Solution:** Made the customer order analysis interactive to open the stores per category chart.