With the dataset columns you've provided, there are various interesting graphs and visualizations you can use to extract useful information for the store owner. Here are some ideas:

1. **Age Group Distribution:** Create a bar chart or pie chart to show the distribution of customers across different age groups. This can help the store owner understand the primary age group of their customers and tailor their products and marketing accordingly.
2. **Gender Distribution:** Visualize the gender distribution of customers using a pie chart or bar chart. This can provide insights into whether the store attracts more male or female customers, helping the owner plan targeted marketing campaigns.
3. **Marital Status Distribution:** Present a bar chart to show the distribution of customers based on their marital status. Understanding the marital status of customers can help in promoting relevant products or festival offers.
4. **Occupation Distribution:** Create a bar chart or pie chart to visualize the distribution of customers across different occupations. This information can help the store owner understand the preferences and needs of various customer segments.
5. **Product Category Sales:** Plot a bar chart showing the sales or orders for each product category. This can help the owner identify the best-selling categories and potentially consider expanding or focusing on those product lines.
6. **Top Products Sold:** Display a bar chart or a table showing the top-selling products and their corresponding sales amounts. This can help the store owner identify the most popular products during the festival season.
7. **State-wise Sales:** Use a map visualization or bar chart to showcase the sales amounts or number of orders from different states. This can help the store owner understand which regions have higher demand and which ones may need more attention.
8. **Zone-wise Sales:** If applicable, visualize sales data based on different zones. This can help the store owner identify high-performing zones and understand regional trends.
9. **Customer Loyalty:** Analyze the number of orders placed by each user and create a histogram or bar chart to visualize customer loyalty. This can help the store owner identify their most loyal customers and potentially offer them loyalty rewards.
10. **Average Order Amount:** Plot a histogram or box plot to show the distribution of order amounts. This can help the store owner understand the typical spending behavior of customers during the festival season.
11. **Correlation Heatmap:** Create a correlation heatmap to explore relationships between various factors like age, gender, occupation, and order amounts. This can reveal interesting insights into customer behavior and preferences.

Remember that the choice of graphs and visualizations will depend on the nature of the data and the specific questions the store owner wants to address. Always ensure that the visualizations are clear, concise, and tailored to the audience's needs.

I got the following code:

<!DOCTYPE html>

<html>

<head>

<title>Festival Store Information</title>

<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

</head>

<body>

<h1>Festival Store Information</h1>

<canvas id="productCategoryChart" width="400" height="200"></canvas>

<div>

<label for="ageGroupSelect">Select Age Group:</label>

<select id="ageGroupSelect">

<option value="all">All Age Groups</option>

</select>

</div>

<canvas id="ageGroupProductCategoryChart" width="400" height="200"></canvas>

<div>

<label for="genderSelect">Select Gender:</label>

<select id="genderSelect">

<option value="all">All Genders</option>

</select>

</div>

<canvas id="genderProductCategoryChart" width="400" height="200"></canvas>

<script>

let data; // Store the fetched data globally

let ageGroupsData; // Store the unique age groups globally

let genderData; // Store the unique gender data globally

let ageGroupProductCategoryChart; // Store the second chart globally

let genderProductCategoryChart; // Store the third chart globally

fetch("http://127.0.0.1:5000/tom")

.then(response => response.json())

.then(data => {

this.data = data; // Store the fetched data in the global variable

// Extract product categories and their counts for the first chart

const productCategories = data.reduce((acc, row) => {

const category = row["Product\_Category"];

acc[category] = (acc[category] || 0) + 1;

return acc;

}, {});

// Get an array of 25 different colors for the bars

const colors = [

'red', 'blue', 'green', 'orange', 'purple', 'yellow', 'cyan', 'magenta', 'brown',

'teal', 'pink', 'lime', 'indigo', 'maroon', 'gold', 'navy', 'coral', 'violet',

'olive', 'steelblue', 'darkorange', 'lightgreen', 'orchid', 'sienna', 'lightblue'

];

// Get the canvas element and create the first chart

const ctx = document.getElementById('productCategoryChart').getContext('2d');

const productCategoryChart = new Chart(ctx, {

type: 'bar',

data: {

labels: Object.keys(productCategories),

datasets: [{

label: 'Product Category Distribution',

data: Object.values(productCategories),

backgroundColor: colors.slice(0, Object.keys(productCategories).length),

borderColor: colors.slice(0, Object.keys(productCategories).length),

borderWidth: 1

}]

},

options: {

scales: {

y: {

beginAtZero: true

}

}

}

});

// Create an array of unique age groups

ageGroupsData = Array.from(new Set(data.map(row => row["Age Group"])));

// Populate the age group dropdown menu with options

const ageGroupSelect = document.getElementById('ageGroupSelect');

ageGroupsData.forEach(ageGroup => {

const option = document.createElement('option');

option.value = ageGroup;

option.textContent = ageGroup;

ageGroupSelect.appendChild(option);

});

// Update the age group graph based on the selected age group

function updateAgeGroupGraph(selectedAgeGroup) {

// Extract age groups and their counts for the second chart

const ageGroups = this.data.reduce((acc, row) => {

const ageGroup = row["Age Group"];

const category = row["Product\_Category"];

if (selectedAgeGroup === 'all' || ageGroup === selectedAgeGroup) {

if (!acc[ageGroup]) {

acc[ageGroup] = {};

}

acc[ageGroup][category] = (acc[ageGroup][category] || 0) + 1;

}

return acc;

}, {});

// Get the canvas element and create/update the second chart

const ctx2 = document.getElementById('ageGroupProductCategoryChart').getContext('2d');

if (ageGroupProductCategoryChart) {

// If the chart exists, update its data

ageGroupProductCategoryChart.data.labels = Object.keys(ageGroups);

ageGroupProductCategoryChart.data.datasets = Object.keys(productCategories).map((category, index) => {

return {

label: category,

data: Object.values(ageGroups).map(group => group[category] || 0),

backgroundColor: colors[index % colors.length],

borderColor: colors[index % colors.length],

borderWidth: 1

};

});

ageGroupProductCategoryChart.update();

} else {

// If the chart does not exist, create it

ageGroupProductCategoryChart = new Chart(ctx2, {

type: 'bar',

data: {

labels: Object.keys(ageGroups),

datasets: Object.keys(productCategories).map((category, index) => {

return {

label: category,

data: Object.values(ageGroups).map(group => group[category] || 0),

backgroundColor: colors[index % colors.length],

borderColor: colors[index % colors.length],

borderWidth: 1

};

})

},

options: {

scales: {

y: {

beginAtZero: true

}

}

}

});

}

}

// Add an event listener to the age group dropdown menu

ageGroupSelect.addEventListener('change', (event) => {

const selectedAgeGroup = event.target.value;

updateAgeGroupGraph(selectedAgeGroup);

});

// Initialize the age group graph with all age groups

updateAgeGroupGraph('all');

// Create an array of unique genders

genderData = Array.from(new Set(data.map(row => row["Gender"])));

// Populate the gender dropdown menu with options

const genderSelect = document.getElementById('genderSelect');

genderData.forEach(gender => {

const option = document.createElement('option');

option.value = gender;

option.textContent = gender;

genderSelect.appendChild(option);

});

// Update the gender graph based on the selected gender

function updateGenderGraph(selectedGender) {

// Extract genders and their counts for the third chart

const genders = this.data.reduce((acc, row) => {

const gender = row["Gender"];

const category = row["Product\_Category"];

if (selectedGender === 'all' || gender === selectedGender) {

if (!acc[gender]) {

acc[gender] = {};

}

acc[gender][category] = (acc[gender][category] || 0) + 1;

}

return acc;

}, {});

// Get the canvas element and create/update the third chart

const ctx3 = document.getElementById('genderProductCategoryChart').getContext('2d');

if (genderProductCategoryChart) {

// If the chart exists, update its data

genderProductCategoryChart.data.labels = Object.keys(genders);

genderProductCategoryChart.data.datasets = Object.keys(productCategories).map((category, index) => {

return {

label: category,

data: Object.values(genders).map(gender => gender[category] || 0),

backgroundColor: colors[index % colors.length],

borderColor: colors[index % colors.length],

borderWidth: 1

};

});

genderProductCategoryChart.update();

} else {

// If the chart does not exist, create it

genderProductCategoryChart = new Chart(ctx3, {

type: 'bar',

data: {

labels: Object.keys(genders),

datasets: Object.keys(productCategories).map((category, index) => {

return {

label: category,

data: Object.values(genders).map(gender => gender[category] || 0),

backgroundColor: colors[index % colors.length],

borderColor: colors[index % colors.length],

borderWidth: 1

};

})

},

options: {

scales: {

y: {

beginAtZero: true

}

}

}

});

}

}

// Add an event listener to the gender dropdown menu

genderSelect.addEventListener('change', (event) => {

const selectedGender = event.target.value;

updateGenderGraph(selectedGender);

});

// Initialize the gender graph with all genders

updateGenderGraph('all');

})

.catch(error => {

console.error('Error fetching data:', error);

});

</script>

</body>

</html>

With the following variables:

Age Age Group Amount Cust\_name Gender Marital\_Status Occupation Orders Product\_Category Product\_ID State User\_ID Zone