LWC Problem statement

Dynamic Data-Driven Dashboard with Custom Filters

Requirement Description:

- Develop a dynamic dashboard in Lightning Web Components (LWC) that displays various data metrics, such as sales performance, customer feedback, and product inventory. The dashboard should allow users to apply custom filters (e.g., date range, product category, region) and dynamically update the displayed metrics without refreshing the entire page. Additionally, the dashboard should support real-time updates by listening to server-side events (using Platform Events or Streaming API).

Key Features:

- **Custom Filters:** Users can select multiple filters, and the dashboard will display the corresponding filtered data.
- **Real-Time Updates:** The dashboard should automatically update when there are changes in the underlying data, reflecting the latest metrics without manual refresh.
- **Responsive Design:** The dashboard should be responsive and adjust according to different screen sizes.

Backend Datamodel explanation

- Sales_Performance__c
- Customer_Feedback__c
- Product_Inventory__c
- Product_Category__c
- Region__c

Sales_Performance__c

- **Description:** This custom object tracks sales data.
- **Fields:**
  - **Sales_Amount__c:** Currency field representing the total sales amount.
  - **Sales_Date__c:** Date field indicating when the sales transaction occurred.
  - **Product_Category__c:** Lookup field to Product_Category__c, linking the sale to a specific product category.
  - **Region__c:** Lookup field to Region__c, indicating the geographic region where the sale occurred.

Customer_Feedback__c

- **Description:** This custom object captures feedback from customers about products or services.
- **Fields:**
  - **Feedback_Score__c:** Number field representing the customer's feedback score (e.g., 1-5).
  - **Feedback_Date__c:** Date field for when the feedback was provided.
  - **Product_Category__c:** Lookup field to Product_Category__c, linking the feedback to a specific product category.
  - **Region__c:** Lookup field to Region__c, indicating the geographic region where the feedback was collected.

Product_Inventory__c

- **Description:** This custom object manages the inventory details of products.

Fields:

- **Product_Name__c:** Text field representing the name of the product.
- **Quantity_Available__c:** Number field indicating the quantity of the product available in stock.
- **Product_Category__c:** Lookup field to Product_Category__c, categorizing the product under a specific category.
- **Region__c:** Lookup field to Region__c, indicating the geographic region where the inventory is stored or relevant.

Product_Category__c

- **Description:** This custom object categorizes products into different categories.
- **Fields:**
  - **Category_Name__c:** Text field representing the name of the product category.
  - **Category_Description__c:** Text area field describing the category.

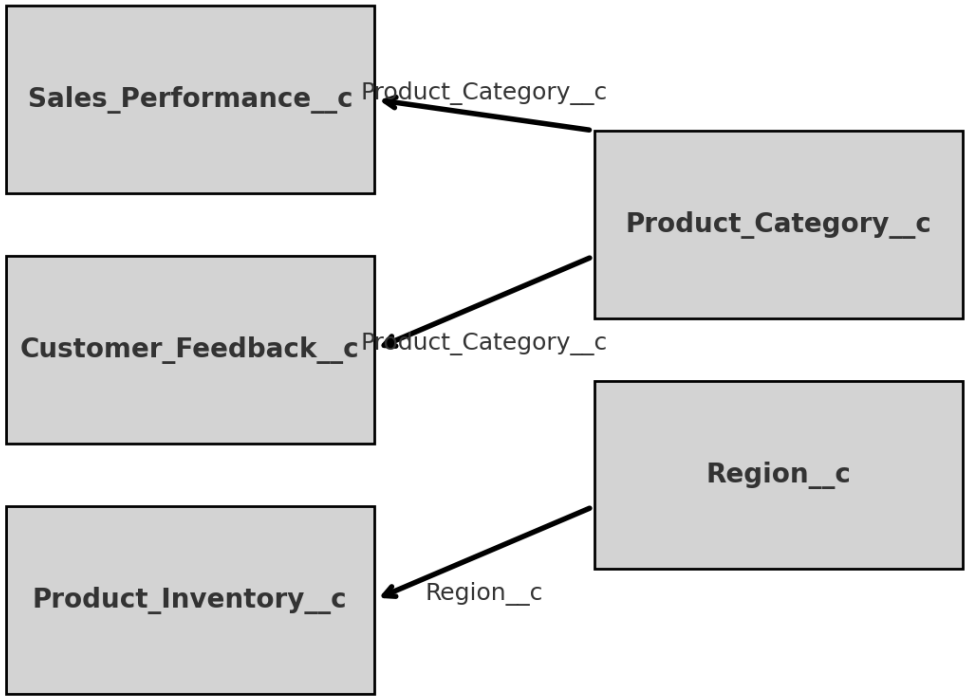
Region__c

- **Description:** This custom object represents different geographic regions.
- **Fields:**
  - **Region_Name__c:** Text field representing the name of the region.
  - **Region_Code__c:** Text field with a unique code representing the region.
  - **Region_Description__c:** Text area field describing the region.

Relationships:

- **Sales_Performance__c**, **Customer_Feedback__c**, and **Product_Inventory__c** are all linked to **Product_Category__c** and **Region__c** through lookup relationships, enabling cross-object filtering and reporting based on product categories and regions.
- **Event Object:** A custom Platform Event object (e.g., Dashboard_Update_Event__e) triggers real-time updates to the LWC dashboard when records in the related objects are updated.

Model



Note: Add a link from Product_Inventory__c to Product_Category__c

Apex Controller:

- An Apex controller class retrieves data from the Sales_Performance__c, Customer_Feedback__c, and Product_Inventory__c objects based on the selected filters.
- The controller uses SOQL queries with dynamic conditions based on the input from LWC to return filtered data.

Wireframe Description:

- **Top Section: Custom Filters**
- **Layout:** A horizontal bar across the top of the dashboard.
- **Elements:**
  - **Dropdowns:** Three dropdown menus labeled:
    - "Date Range" - Allows the user to select a specific time frame.
    - "Product Category" - Filters data by product category.
    - "Region" - Filters data by geographic region.
  - **Button:** On the far right of the bar, there is an "Apply Filters" button that users can click to refresh the data based on the selected filters.

Middle Section: Sales Performance Chart

- **Chart:** A large bar chart that takes up the majority of the width of the dashboard below the filters.
- **Title:** "Sales Performance" displayed above the chart.
- **Legend:** On the right side of the chart, there is a legend that explains the different metrics or categories represented in the bars.
- **Axes:** The X-axis represents time or categories, while the Y-axis represents sales figures.

Lower Left Section: Customer Feedback Chart

- **Chart:** A pie chart located in the lower-left quadrant of the dashboard.
- **Title:** "Customer Feedback" displayed above the chart.
- **Slices:** The chart is divided into slices representing different feedback scores (e.g., positive, neutral, negative).

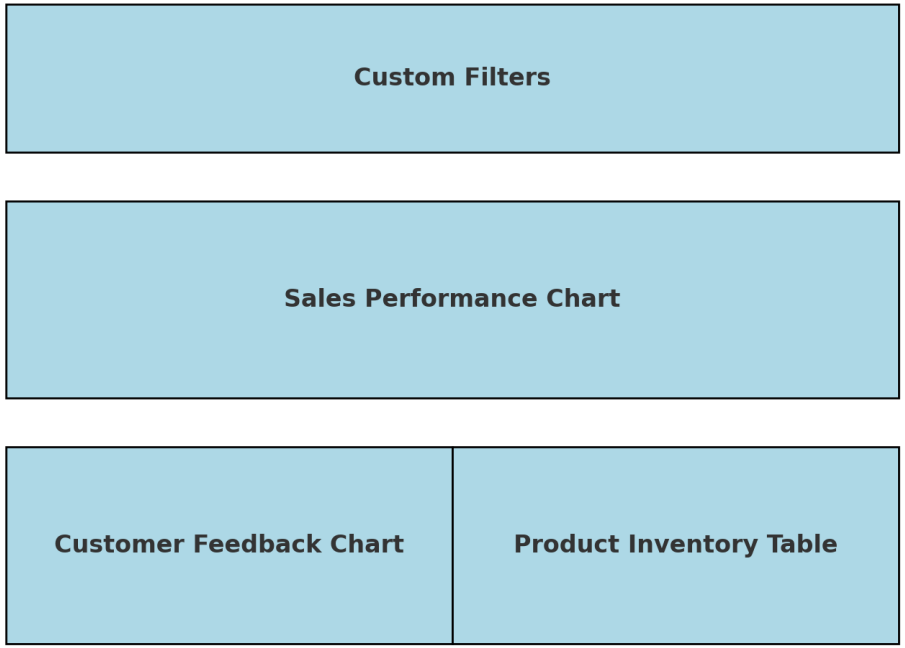
Lower Right Section: Product Inventory Table

- **Table:** A data table located in the lower-right quadrant of the dashboard.
- **Title:** "Product Inventory" displayed above the table.
- **Columns:**
  - "Product Name"
  - "Available Quantity"
  - "Region"
- **Functionality:** The table is sortable and scrollable, allowing users to view and organize the data easily.

Overall Design:

- **Clean and Modern:** The design is intended to be straightforward and user-friendly, focusing on clear data
- **Responsive Layout:** The dashboard is designed to adapt to different screen sizes, ensuring usability across devices.

End Result - UX



UX - Description

- **Custom Filters Section:** Located at the top, this section allows users to select filters like Date Range, Product Category, and Region.
- **Sales Performance Chart:** A large bar chart displayed below the filters, showing sales performance data.
- **Customer Feedback Chart:** A pie chart positioned in the lower left, displaying customer feedback scores.
- **Product Inventory Table:** A table on the lower right, listing product inventory details.

Functional Requirements:

- **Custom Filters Section (Parent Component):**
- **Filters:** Include dropdowns for filtering by Date Range, Product Category, and Region.
- **Event Handling:** When a filter is changed, fire a custom event to notify child components to refresh their data.

Sales Performance Chart (Child Component):

- **Data Binding:** Display a bar chart showing sales performance based on the selected filters.
- **Real-Time Updates:** Subscribe to a Platform Event or Streaming API to update the chart in real-time when there are changes in sales data.
- **Apex Integration:** Fetch the initial sales data through an Apex controller based on the filters passed from the parent component.

Customer Feedback Chart (Child Component):

- **Data Binding:** Display a pie chart showing customer feedback scores.
- **Lifecycle Hooks:** Use connectedCallback to fetch data when the component is first rendered.
- **Event Handling:** Listen for filter change events from the parent component and update the chart data accordingly.
- **Responsive Design:** Ensure the chart adjusts correctly on different screen sizes.

Product Inventory Table (Child Component):

1. **Data Binding:** Display a table of product inventory levels filtered by the selected criteria.
2. **Apex Integration:** Use an Apex controller to fetch the inventory data, applying the filters dynamically.
3. **Event Handling:** Update the table data when filters are applied by handling the custom event from the parent component.
4. **Real-Time Updates:** Integrate real-time updates for inventory changes using Platform Events or Streaming API.

Submission

Pre-requisite:

During submission make sure to create enough test data to show working filters and charts

Ability to build platform events and real-time dashboard refresh will have an additional score.