

# WhatNext Vision Motors: Shaping the Future of Mobility with Innovation and Excellence

WhatsNext Vision Motors, a pioneering force in the automotive industry, is dedicated to transforming the mobility sector with innovative technology and solutions that prioritize customer needs. The company has embarked on an ambitious Salesforce project with the core objective of enhancing the customer experience and streamlining its operational processes.

At the heart of this project is the improvement of the customer ordering process. The system is designed to automatically suggest the nearest dealer location to customers based on their address. This feature is intended to significantly enhance the convenience and efficiency of the ordering experience, making it more customer-friendly and reducing the time and effort required from the customer's end.

The project also addresses a common issue in the automotive industry: stock availability. The system includes a mechanism that prevents customers from placing orders for vehicles that are out of stock. This proactive approach ensures that customers can only create orders for vehicles that are currently available, thus avoiding potential confusion and disappointment that may arise from stock unavailability. This feature not only enhances customer satisfaction but also improves the accuracy of the company's order fulfillment process.

Furthermore, the project incorporates a scheduled process for updating the status of bulk order records. This automated process is designed to update the order status based on stock availability. If a vehicle is out of stock at the time of order placement, the system will update the order status to 'Pending.' On the other hand, if the vehicle is in stock, the system will update the status to 'Confirmed.' This ensures that all orders are accurately reflected in terms of their fulfillment status, providing clear and transparent communication to customers regarding the status of their orders.

The implementation of this Salesforce project at WhatsNext Vision Motors is expected to yield several benefits. It aims to create a more efficient ordering system that reduces the potential for errors and improves the overall service provided to

# WhatNext Vision Motors: Shaping the Future of Mobility with Innovation and Excellence

## 1. Project Overview

This document outlines the data management and object creation phase for the WhatNext Vision Motors project. The core of this phase is to establish a robust data structure within Salesforce to manage vehicles, customers, dealers, and related activities.

## 2. Data Management: Objects & Relationships

The following custom objects need to be created to form the foundational data model for the application.

Object API Name	Purpose	Relationships
Vehicle__c	Stores vehicle details	Related to Dealer & Orders
Vehicle_Dealer__c	Stores authorized dealer info	Related to Orders
Vehicle_Customer__c	Stores customer details	Related to Orders & Test Drives
Vehicle_Order__c	Tracks vehicle purchases	Related to Customer & Vehicle
Vehicle_Test_Drive__c	Tracks test drive bookings	Related to Customer & Vehicle
Vehicle_Service_Request__c	Tracks vehicle servicing requests	Related to Customer & Vehicle

## 3. Object Creation Guide: Vehicle\_\_c

The following steps detail the process for creating a new custom object, using the **Vehicle** object as an example. This same process should be followed for all objects listed in the table above, with their respective details.

### Steps:

1. From the Salesforce **Setup** page, navigate to **Object Manager** → Click on **Create** → Select **Custom Object**.
2. **Enter the Custom Object Definition:**
  - **Label:** Vehicle

- **Plural Label:** Vehicles
  - **Object Name:** Vehicle
3. **Enter Record Name Label and Format:**
- **Record Name:** Vehicle Name
  - **Data Type:** Text
4. **Enable Optional Features:**
- Check the box for **Allow Reports.**
  - Check the box for **Allow Search.**
5. **Save** the new object.

## 4. Task: Create Remaining Objects

Following the guide in Section 3, the next task is to create all the remaining objects as specified in the "Objects & Relationships" table (Section 2).


- **Story:** More Object Creation
- **Assigned to:** Pratham Shinde
- **Description:** Create the remaining objects mentioned in the Data Management-Objects table.

### Description

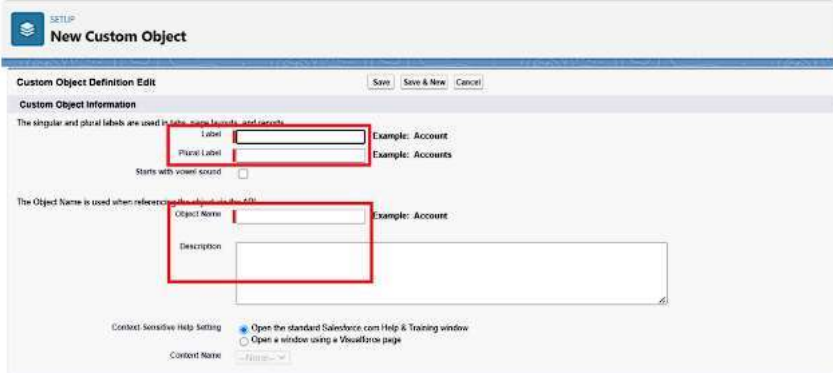
## Create Vehicle Object

The purpose of creating a Vehicle custom object is to have a clear picture of the vehicle details  
To create an object:

- From the setup page → Click on Object Manager → Click on Create → Click on Custom Object



- Enter the label name → Vehicle
- Plural label name → Vehicles
- Enter Record Name Label and Format
  - Record Name → Vehicle Name
  - Data Type → Text



# WhatNext Vision Motors: Data Management - Tabs

## 1. Task Overview: Tabs Creation

This document outlines the process for creating custom tabs for each of the objects built in the previous phase. These tabs will provide users with direct access to the object records from the Salesforce UI.

- **Epic:** Data Management-Tabs
- **Complexity:** Easy
- **Description:** Tabs creation

## 2. Prerequisite: Objects & Relationships

Before creating the tabs, it's important to reference the list of custom objects that have been created. A tab should be created for each of these objects.

Object API Name	Purpose	Relationships
Vehicle__c	Stores vehicle details	Related to Dealer & Orders
Vehicle_Dealer__c	Stores authorized dealer info	Related to Orders
Vehicle_Customer__c	Stores customer details	Related to Orders & Test Drives
Vehicle_Order__c	Tracks vehicle purchases	Related to Customer & Vehicle
Vehicle_Test_Drive__c	Tracks test drive bookings	Related to Customer & Vehicle
Vehicle_Service_Request__c	Tracks vehicle servicing requests	Related to Customer & Vehicle

## 3. Guide: Creating a Custom Tab (Vehicle)

The following steps detail the process for creating a new custom tab, using the **Vehicle** object as the primary example.

- **Assigned to:** Pratham Shinde

### Steps:

1. Navigate to **Setup**. In the Quick Find box, type "**Tabs**" and select it under User Interface.
2. In the **Custom Object Tabs** section, click **New**.

3. For **Step 1: Enter the Details**, select the **Vehicle** object from the dropdown list.
4. Select a **Tab Style**.
5. Click **Next** through the "Add to Profiles" page (keeping defaults).
6. Click **Next** through the "Add to Custom Apps" page (keeping defaults).
7. Click **Save**.

## 4. Next Steps: Create Remaining Tabs

Following the guide in Section 3, the next task is to create tabs for all the other objects that were created previously.

**Description:** Now create tabs for Other objects as well which we have created above Objects.

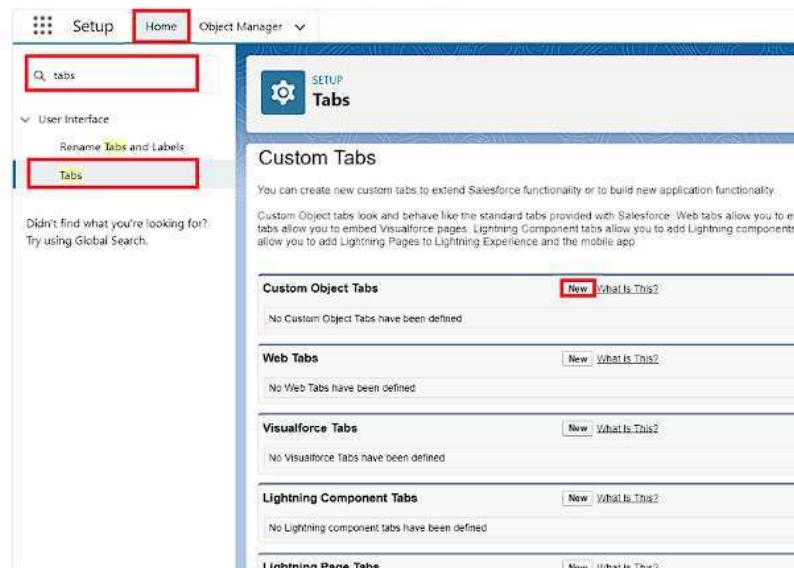
### Audio Overview

Would you like me to generate an audio overview for this new document about creating tabs?

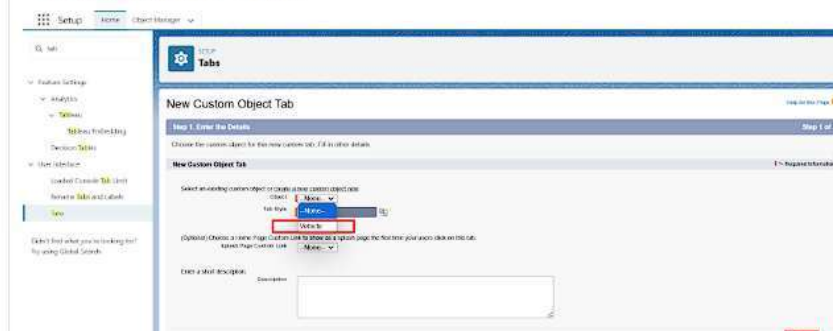
#### Description

## Creating a Custom Tab(Vehicle)

1. Go to setup page → type Tabs in Quick Find bar → click on tabs → New (under custom object tab)



2. Select Object(Vehicle) → Select any tab style → Next (Add to profiles page) keep it as default → Next (Add to Custom App) keep it as default → Save.



# WhatNext Vision Motors: Create a Lightning App

## 1. Overview

This document provides a step-by-step guide to creating the "WhatNext Vision Motors" Lightning App. This app will serve as the primary user interface, bringing together all the custom objects and tabs created in the previous phases.

## 2. App Creation and Configuration

### Steps:

1. Go to the **Setup** page. In the Quick Find box, search for "**App Manager**".
2. Select **App Manager** from the results and then click on **New Lightning App**.
3. **App Details and Branding:**
  - **App Name:** WhatNext Vision Motors
  - **Developer Name:** This will be auto-populated.
  - **Description:** Provide a meaningful description for the app.
  - **Image:** (Optional) You can upload a logo or image for branding.
  - **Primary Color Hex Value:** Keep the default.
4. Click **Next**.
5. **App Options:** Keep the default settings and click **Next**.
6. **Utility Items:** Keep the default settings and click **Next**.

## 3. Adding Navigation and User Profiles

### Steps:

1. **Add Navigation Items:**
  - In the "Available Items" list, search for the following items one by one:
    - Vehicle
    - Dealer
    - Customer
    - Order
    - Test Drive
    - Service Request
    - Reports
    - Dashboard
  - Use the arrow button to move each item to the "Selected Items" list.
  - Click **Next**.
2. **Add User Profiles:**
  - In the "Available Profiles" list, search for **System Administrator**.



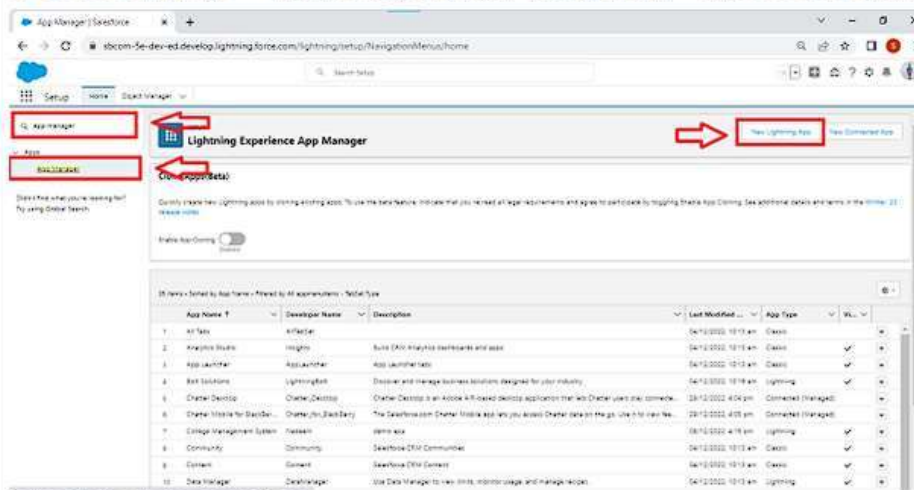
- Use the arrow button to move it to the "Selected Profiles" list.
3. **Save and Finish:**
- Click the **Save & Finish** button to complete the app creation.

## Description

# Create a Lightning App

To create a lightning app page:

1. Go to setup page → search "app manager" in quick find → select "app manager" → click on New lightning App.



1. Fill the app name in app details and branding as follow

App Name : WhatNext Vision Motors

Developer Name : this will auto populated

Description : Give a meaningful description

Image : optional (if you want to give any image you can otherwise not mandatory)

Primary color hex value : keep this default

2. Then click Next → (App option page) keep it as default → Next → (Utility Items) keep it as default → Next.



# WhatNext Vision Motors: Fields & Relationships

## 1. Key Fields for Each Object

This section outlines the necessary fields and their data types for each custom object in the project.

### 1. Vehicle\_\_c (Custom Object)

- **Vehicle\_Name\_\_c** (Text)
- **Vehicle\_Model\_\_c** (Picklist: Sedan, SUV, EV, etc.)
- **Stock\_Quantity\_\_c** (Number)
- **Price\_\_c** (Currency)
- **Dealer\_\_c** (Lookup to Dealer\_\_c)
- **Status\_\_c** (Picklist: Available, Out of Stock, Discontinued)

### 2. Vehicle\_Dealer\_\_c (Custom Object)

- **Dealer\_Name\_\_c** (Text)
- **Dealer\_Location\_\_c** (Text)
- **Dealer\_Code\_\_c** (Auto Number)
- **Phone\_\_c** (Phone)
- **Email\_\_c** (Email)

### 3. Vehicle\_Order\_\_c (Custom Object)

- **Customer\_\_c** (Lookup to Customer\_\_c)
- **Vehicle\_\_c** (Lookup to Vehicle\_\_c)
- **Order\_Date\_\_c** (Date)
- **Status\_\_c** (Picklist: Pending, Confirmed, Delivered, Canceled)

### 4. Vehicle\_Customer\_\_c (Custom Object)

- **Customer\_Name\_\_c** (Text)
- **Email\_\_c** (Email)
- **Phone\_\_c** (Phone)
- **Address\_\_c** (Text)
- **Preferred\_Vehicle\_Type\_\_c** (Picklist: Sedan, SUV, EV, etc.)

### 5. Vehicle\_Test\_Drive\_\_c (Custom Object)

- **Customer\_\_c** (Lookup to Customer\_\_c)
- **Vehicle\_\_c** (Lookup to Vehicle\_\_c)
- **Test\_Drive\_Date\_\_c** (Date)
- **Status\_\_c** (Picklist: Scheduled, Completed, Canceled)



## 6. Vehicle\_Service\_Request\_\_c (Custom Object)

- **Customer\_\_c** (Lookup to Customer\_\_c)
- **Vehicle\_\_c** (Lookup to Vehicle\_\_c)
- **Service\_Date\_\_c** (Date)
- **Issue\_Description\_\_c** (Text)
- **Status\_\_c** (Picklist: Requested, In Progress, Completed)

## 2. Guide: Creating a Field in the Vehicle Object

This example shows how to create the Vehicle\_Model\_\_c picklist field.

### Steps:

1. Go to **Setup** → **Object Manager** → search for and click on **Vehicle**.
2. Click on **Fields & Relationships** → **New**.
3. Select **Picklist** as the Data type and click **Next**.
4. Fill in the details:
  - **Field Label:** Vehicle Model
  - **Values:** Select "Enter values, with each value separated by a new line" and enter the values (e.g., Sedan, SUV, EV).
5. Click **Next** → **Next** → **Save and new**.

## 3. Guide: Creating a Lookup Relationship

This example shows how to create the lookup relationship from the Vehicle object to the Dealer object.

### Steps:

1. In the Vehicle object's **Fields & Relationships** section, click **New**.
2. Select **Lookup Relationship** as the Data type and click **Next**.
3. In the "Related To" dropdown, select **Dealer**. Click **Next**.
4. Enter the Field Label and Name. Click **Next**.
5. Set the field-level security for the profiles. Click **Next**.
6. Add the related list to the page layouts. Click **Next**.
7. Click **Save**.

## 4. Next Task: Create Remaining Fields

The next step is to create all the remaining fields for each object as listed in Section 1, using the guides above as a reference for creating different field types.

**Description:** Create remaining fields as mentioned in the Data Management-Fields.

# WhatNext Vision Motors: Flow Creation

## 1. Task: Auto-Assign Nearest Dealer

This document outlines the process for creating a record-triggered flow. The purpose of this flow is to automatically find the nearest dealer based on the customer's location and assign them to a new Vehicle Order.

- **Story:** Flow Creation
- **Assigned to:** Pratham Shinde

## 2. Flow Setup and Trigger Configuration

### Steps:

1. In **Setup**, use the Quick Find box to search for and select **Flows**. Click **New Flow**.
2. Select **Start From Scratch** and click **Next**.
3. For the flow type, select **Record-Triggered Flow** and click **Create**.
4. **Configure the Trigger:**
  - **Object:** Vehicle Order
  - **Trigger the Flow When:** A record is created.
  - **Set Entry Conditions:**
    - **Condition Requirements:** All Conditions Are Met (AND)
    - **Field:** Status\_\_c
    - **Operator:** Equals
    - **Value:** Pending

## 3. Building the Flow Logic

### Step 5: Get Customer Information

- Click the + icon and select the **Get Records** element.
- **Label:** Get Customer Information
- **Object:** Vehicle Customer
- **Filter Records:**
  - **Field:** Id
  - **Operator:** Equals
  - **Value:** {!\$Record.Vehicle\_Customer\_\_c}

### Step 6: Get Nearest Dealer

- Click the + icon and select **Get Records**.
- **Label:** Get Nearest Dealer
- **Object:** Vehicle Dealer
- **Filter Records:**

- **Field:** Dealer\_Location\_\_c
- **Operator:** Equals
- **Value:** {!Get\_Customer\_Information.Address\_\_c}

## Step 7: Assign Dealer to Order

- Click the + icon and select **Update Records**.
- **Label:** Assign Dealer to Order
- **How to Find Records...:** Select "Use the IDs and all field values from a record or record collection".
- **Select Record(s) to Update:** {!Get\_Nearest\_Dealer}

## 4. Save and Activate the Flow

### Step 8: Save the Flow

- Click **Save**.
- **Flow Label:** Auto Assign Dealer
- **Flow API Name:** Auto\_Assign\_Dealer

### Step 9: Activate Flow

- After saving, click the **Activate** button on the flow canvas.

#### Description

### Creating a record triggered flow to assign nearest dealer to the customer's location

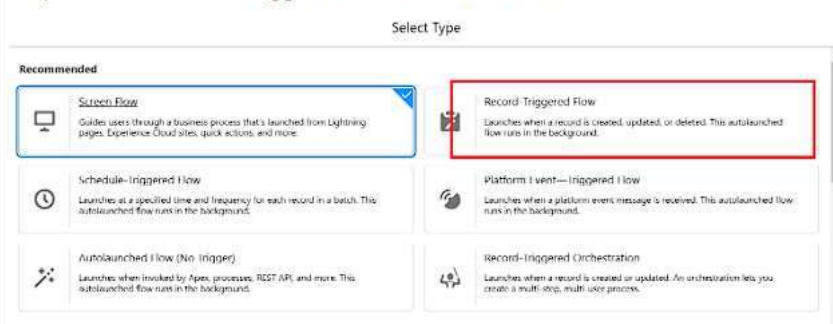
Step 1 : In Quick Find, type Flows and click on Flows. Click New Flow.



Step 2 : Select Start From Scratch and click Next.



Step 3 : Select Record-Triggered Flow and click Create



# WhatNext Vision Motors: Test Drive Reminder Flow

## 1. Task: Create Test Drive Reminder Email

This document provides instructions for creating a record-triggered flow that automatically sends an email to a customer one day before their scheduled test drive.

## 2. Flow Trigger and Scheduled Path Setup

### Step 1: Create a Record-Triggered Flow

- In **Setup**, navigate to **Flows** and create a new **Record-Triggered Flow**.
- **Object:** Vehicle Test Drive
- **Trigger the Flow When:** A record is created or updated.
- **Set Entry Conditions:**
  - **Condition Requirements:** All Conditions Are Met (AND)
  - **Field:** Status\_\_c
  - **Operator:** Equals
  - **Value:** Scheduled

### Step 2: Add a Scheduled Path

- Below the start element, click **Add Scheduled Paths (Optional)**.
- **Path Label:** Reminder Before Test Drive
- **Time Source:** Vehicle\_Test\_Drive\_\_c: Test Drive Date
- **Offset Number:** 1
- **Offset Options:** Days Before
- Click **Done**.

## 3. Building the Flow Logic

Now, build the logic within the "Reminder Before Test Drive" path.

### Step 3: Get Customer Information

- Click the **+** icon and add a **Get Records** element.
- **Label:** Get Customer Information
- **Object:** Vehicle\_Customer\_\_c
- **Filter Records:**
  - **Field:** Id
  - **Operator:** Equals
  - **Value:** {!\$Record.Customer\_\_c}

### Step 4: Send Reminder Email

- Click the **+** icon and add an **Action** element.
- **Action:** Send Email
- **Label:** Send Test Drive Reminder
- **Input Values:**
  - **Body:** Create a new text template variable for the email body.
  - **Subject:** Reminder: Your Test Drive is Tomorrow!
  - **Recipient Address List:** {!Get\_Customer\_Information.Email\_\_c}
  - **Rich-Text-Formatted Body:** {!\$GlobalConstant.True}

## 4. Save, Activate, and Test

### Step 5: Save and Activate

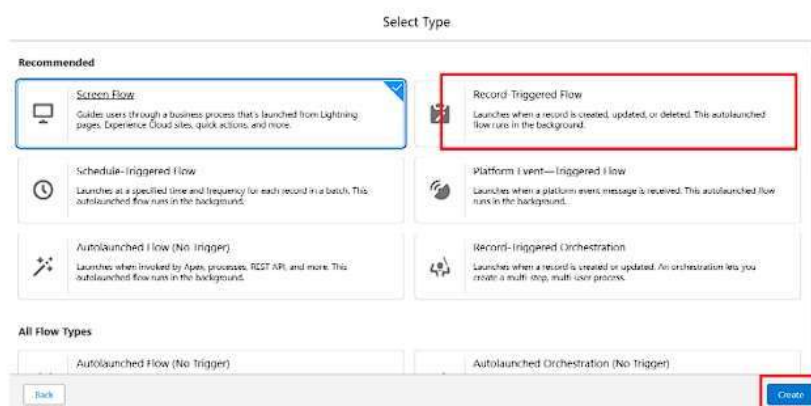
- Click **Save**.
- **Flow Label:** Test Drive Reminder
- Click **Save**, then click **Activate**.

### Step 6: Test Your Flow

- After activation, you can test the flow by creating or updating a Test Drive record to meet the criteria. The flow will schedule and send the email as configured.

## Creating record triggered flow to send an email to the customer reminding about the test drive.

### Step 1 : Select Record-Triggered Flow and click Create



### Step 2 : Select Vehicle Test Drive Object

Trigger the Flow When : A record is created or updated

#### Set Entry Condition :

- All Conditions Are Met (AND)
- Filed : Status\_\_c
- Operator : Equals
- Value : Scheduled

### Step 3 : Click + Add Scheduled Paths (below the trigger).

- Label: Reminder Before Test Drive.
- Time Source: Test\_Drive\_Date\_\_c
- Offset Number: 1.
- Offset Options: Days Before.
- Click Done.

# WhatNext Vision Motors: Apex and Batch Jobs

## 1. Task: Create Apex Trigger and Batch Jobs

This document outlines the development of Apex code to handle advanced business logic for the WhatNext Vision Motors application. This includes a trigger to manage vehicle stock during ordering and a batch job to process pending orders.

- **Assigned to:** Pratham Shinde

## 2. Apex Trigger for Vehicle Orders

This trigger prevents new orders for out-of-stock vehicles and updates the stock quantity when an order is confirmed. It uses a handler class to contain the logic.

### Step 1: Create the Trigger Handler Apex Class

1. Open the **Developer Console**.
2. Go to **File > New > Apex Class**.
3. Name the class VehicleOrderTriggerHandler and click **OK**.
4. **Paste the following code** into the class:

```
public class VehicleOrderTriggerHandler {
    public static void handleTrigger(List<Vehicle_Order__c> newOrders, Map<Id,
Vehicle_Order__c> oldOrders, Boolean isBefore, Boolean isAfter, Boolean isInsert,
Boolean isUpdate) {
        if (isBefore) {
            if (isInsert || isUpdate) {
                preventOrderIfOutOfStock(newOrders);
            }
        }

        if (isAfter) {
            if (isInsert || isUpdate) {
                updateStockOnOrderPlacement(newOrders);
            }
        }
    }

    // Method to prevent orders when the vehicle is out of stock
    private static void preventOrderIfOutOfStock(List<Vehicle_Order__c> orders) {
        Set<Id> vehicleIds = new Set<Id>();
        for (Vehicle_Order__c order : orders) {
```

```

        if (order.Vehicle__c != null) {
            vehicleIds.add(order.Vehicle__c);
        }
    }

    if (!vehicleIds.isEmpty()) {
        Map<Id, Vehicle__c> vehicleStockMap = new Map<Id, Vehicle__c>(
            [SELECT Id, Stock_Quantity__c FROM Vehicle__c WHERE Id IN
:vehicleIds]
        );

        for (Vehicle_Order__c order : orders) {
            if (vehicleStockMap.containsKey(order.Vehicle__c)) {
                Vehicle__c vehicle = vehicleStockMap.get(order.Vehicle__c);
                if (vehicle.Stock_Quantity__c <= 0) {
                    order.addError('This vehicle is out of stock. Order cannot be
placed.');
```



```

        for (Vehicle_Order__c order : orders) {
            if (vehicleStockMap.containsKey(order.Vehicle__c)) {
                Vehicle__c vehicle = vehicleStockMap.get(order.Vehicle__c);
                if (vehicle.Stock_Quantity__c > 0) {
                    vehicle.Stock_Quantity__c -= 1;
                    vehiclesToUpdate.add(vehicle);
                }
            }
        }
        if (!vehiclesToUpdate.isEmpty()) {
            update vehiclesToUpdate;
        }
    }
}

```

## Step 2: Create the Apex Trigger

1. In the **Developer Console**, go to **File > New > Apex Trigger**.
2. **Name:** VehicleOrderTrigger, **sObject:** Vehicle\_Order\_\_c. Click **Submit**.
3. **Paste the following code** to call the handler:  
 trigger VehicleOrderTrigger on Vehicle\_Order\_\_c (before insert, before update, after insert, after update) {  
     VehicleOrderTriggerHandler.handleTrigger(trigger.new, trigger.oldMap,  
     trigger.isBefore, trigger.isAfter, trigger.isInsert, trigger.isUpdate);  
 }

## 3. Apex Batch Job for Pending Orders

This batch job runs daily to find pending orders and confirm them if the vehicle is back in stock.

### Step 3: Create the Batch Apex Class

1. In the **Developer Console**, go to **File > New > Apex Class**.
2. Name the class VehicleOrderBatch and click **OK**.
3. **Paste the following code:**  

```

global class VehicleOrderBatch implements Database.Batchable<sObject> {
    global Database.QueryLocator start(Database.BatchableContext bc) {
        return Database.getQueryLocator(

```