

## WhatsNext 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 through innovative technology and customer-focused solutions. This Salesforce project aims to enhance customer experience, streamline the ordering process, and strengthen operational efficiency across the organization.

### Project Overview

This Salesforce implementation improves the customer ordering process by automatically suggesting the nearest dealer based on the customer's address. This feature enhances convenience and reduces the effort required for customers when placing vehicle orders.

The system also prevents order placement for vehicles that are out of stock. This ensures that customers can only place orders for available units, reducing confusion and boosting order accuracy.

A scheduled process is included to automatically update bulk order records. If a vehicle is out of stock, the order status becomes Pending. If the vehicle is available, the order status becomes Confirmed. This automation promotes transparency and accurate communication with customers.

Overall, this project enhances efficiency, reduces manual workload, minimizes errors, and improves customer satisfaction.

### Requirements

#### 1. Salesforce CRM Implementation

- Store and manage vehicle details, stock availability, and dealer information.
- Track customer orders, test drives, and service requests.
- Automatically assign orders to the nearest dealer based on customer location.

## **2. Process Automation**

- Prevent order placement for out-of-stock vehicles.
- Auto-assign orders to the nearest dealer.
- Send automated email reminders for scheduled test drives.

## **3. Apex and Triggers**

- Implement Apex triggers for stock validation and dealer assignment.
- Use a trigger handler framework for modular and maintainable code.

## **4. Batch Jobs**

- Create a Batch Apex job to periodically check and update stock availability.
- Send scheduled email notifications for stock replenishment and order processing.

### **What You'll Learn**

- Data Modelling
- Fields and Relationships
- Lightning App Builder
- Record-Triggered Flows
- Apex and Apex Triggers
- Batch Apex
- Scheduled Apex

# Project Progress

## Salesforce Credentials Creation

The screenshot shows the Salesforce Setup Home page. On the left, there's a sidebar with links like Setup Home, Service Setup Assistant, Commerce Setup Assistant, and Platform Tools. The main area features a "Welcome, Jermaine Micah" message and three cards under "Achieve Popular Business Goals": "Connect with Sales Prospects and Customers" (3 Completed, 1 In Progress), "Track & Manage Customer Data" (2 Completed, 1 In Progress), and "Capture & Auto-Qualify Leads" (1 Completed). Below these are sections for "Recent Items" and "Vehicle Order" objects.

## Data Management-Objects

The screenshot shows the Salesforce Object Manager page. It lists six custom objects: Vehicle, Vehicle Customer, Vehicle Dealer, Vehicle Order, Vehicle Service Request, and Vehicle Test Drive. Each entry includes the Label, API Name, Type, Description, Last Modified date (all listed as 11/20/2025), and a Deployed status indicator.

Label	API Name	Type	Description	Last Modified	Deployed
Vehicle	Vehicle_c	Custom Object		11/20/2025	✓
Vehicle Customer	Vehicle_Customer_c	Custom Object		11/20/2025	✓
Vehicle Dealer	Vehicle_Dealer_c	Custom Object		11/20/2025	✓
Vehicle Order	Vehicle_Order_c	Custom Object		11/21/2025	✓
Vehicle Service Request	Vehicle_Service_Request_c	Custom Object		11/20/2025	✓
Vehicle Test Drive	Vehicle_Test_Drive_c	Custom Object		11/20/2025	✓

## Data Management-Tabs

The screenshot shows the Salesforce Setup interface with the 'Tabs' tab selected under 'User Interface'. The page title is 'Tabs'. It includes sections for 'Custom Tabs', 'Web Tabs', and 'Visualforce Tabs'. The 'Custom Tabs' section lists several custom object tabs with their labels and styles:

Action	Label	Tab Style	Description
Edit   Del	Vehicle Customers	People	
Edit   Del	Vehicle Dealers	Building	
Edit   Del	Vehicle Orders	Box	
Edit   Del	Vehicles	Car	
Edit   Del	Vehicle Service Requests	Form	
Edit   Del	Vehicle Test Drives	Gears	

The 'Web Tabs' and 'Visualforce Tabs' sections both indicate that no tabs have been defined.

## Data Management-App Manager

The screenshot shows the App Manager interface for the 'Vehicle Customers' app. The top navigation bar includes links for Vehicle Customers, Vehicle Dealers, Vehicle Service Requests, Vehicle Orders, Vehicle Test Drives, Vehicles, Reports, and Dashboards. The main area displays a list of recently viewed items under the heading 'Recently Viewed'. A single item, 'John', is listed with a checkbox next to it. A toolbar at the bottom right provides options for New, Import, Change Owner, and Assign Label.

## Data Management-Fields

Setup > Object Manager > Vehicle Customer

**Fields & Relationships**

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Address	Address__c	Text(60)		
Created By	CreatedBy	Lookup(User)		
Email	Email__c	Email		
Last Modified By	LastModifiedBy	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)		✓
Phone	Phone__c	Phone		
Preferred Vehicle Type	Preferred_Vehicle_Type__c	Picklist		
Vehicle Customer Name	Name	Text(80)		✓

Setup > Object Manager > Vehicle Dealer

**Fields & Relationships**

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedBy	Lookup(User)		
Dealer Code	Dealer_Code__c	Auto Number		
Dealer Location	Dealer_Location__c	Text(60)		
Dealer Name	Name	Text(80)		✓
Email	Email__c	Email		
Last Modified By	LastModifiedBy	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)		✓
Phone	Phone__c	Phone		

Setup Home Object Manager

SETUP > OBJECT MANAGER  
Vehicle

**Fields & Relationships**  
9 Items, Sorted by Field Label

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedById	Lookup(User)		
Last Modified By	LastModifiedById	Lookup(User)		
Owner	OwnerId	LookupUser/Group		✓
Price	Price__c	Currency(18, 0)		
Status	Status__c	Picklist		
Stock Quantity	Stock_Quantity__c	Number(18, 0)		
Vehicle Dealer	Vehicle_Dealer__c	Lookup(Vehicle Dealer)		✓
Vehicle Model	Vehicle_Model__c	Picklist		
Vehicle Name	Name	Text(80)		✓

Search Setup

Setup Home Object Manager

SETUP > OBJECT MANAGER  
Vehicle Order

**Fields & Relationships**  
9 Items, Sorted by Field Label

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Assigned Dealer	Assigned_Dealer__c	Lookup(Vehicle Dealer)		✓
Created By	CreatedById	Lookup(User)		
Last Modified By	LastModifiedById	Lookup(User)		
Order Date	Order_Date__c	Date		
Owner	OwnerId	LookupUser/Group		✓
Status	Status__c	Picklist		
Vehicle	Vehicle__c	Lookup(Vehicle)		✓
Vehicle Customer	Vehicle_Customer__c	Lookup(Vehicle Customer)		✓
Vehicle Order Number	Name	Auto Number		✓

Search Setup

The image displays two screenshots of the Salesforce Object Manager interface, one above the other.

**Vehicle Service Request**

**Fields & Relationships**  
9 Items, Sorted by Field Label

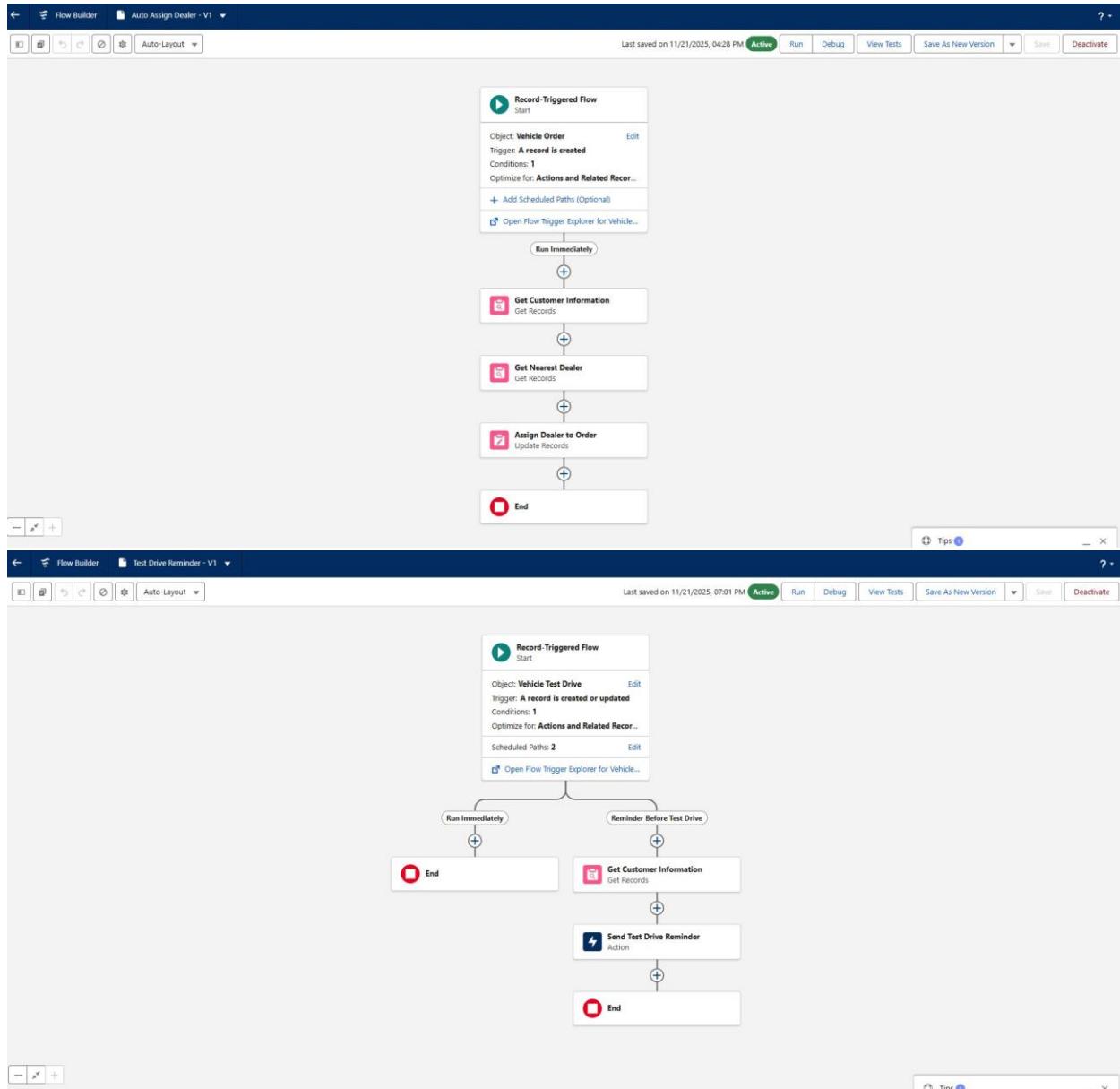
FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedBy	Lookup(User)		✓
Customer	Customer_c	Lookup(Customer)		✓
Issue Description	Issue_Description_c	Text(60)		✓
Last Modified By	LastModifiedBy	Lookup(User)		✓
Owner	OwnerId	Lookup(User.Group)		✓
Service Date	Service_Date_c	Date		✓
Status	Status_c	Picklist		✓
Vehicle	Vehicle_c	Lookup(Vehicle)		✓
Vehicle Service Request Name	Name	Text(80)		✓

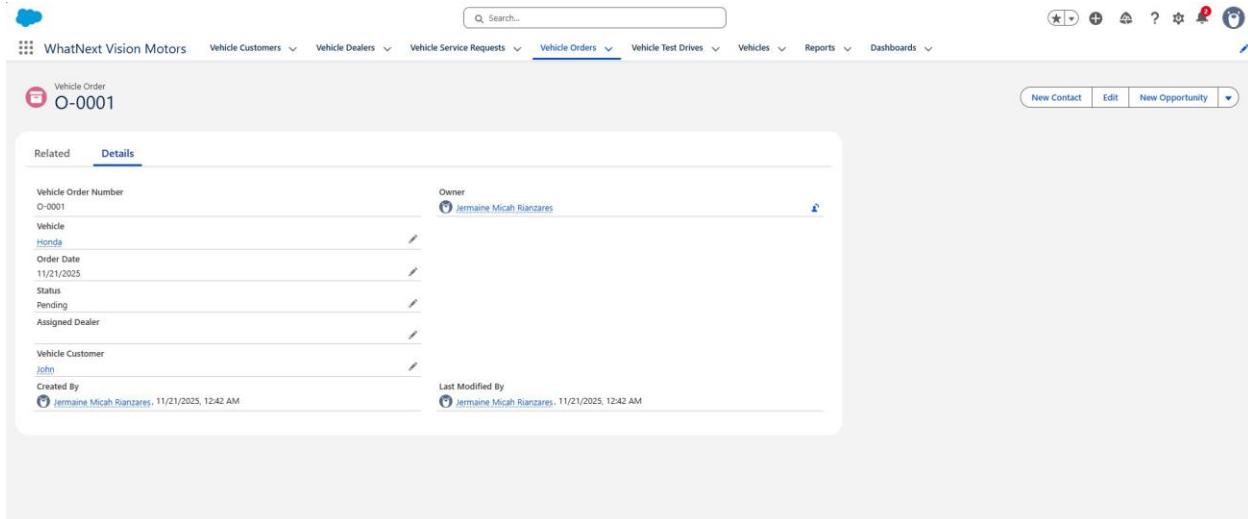
**Vehicle Test Drive**

**Fields & Relationships**  
8 Items, Sorted by Field Label

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedBy	Lookup(User)		✓
Last Modified By	LastModifiedBy	Lookup(User)		✓
Owner	OwnerId	Lookup(User.Group)		✓
Status	Status_c	Picklist		✓
Test Drive Date	Test_Drive_Date_c	Date		✓
Vehicle	Vehicle_c	Lookup(Vehicle)		✓
Vehicle Customer	Vehicle_Customer_c	Lookup(Vehicle Customer)		✓
Vehicle Test Drive Name	Name	Text(80)		✓

## Automation





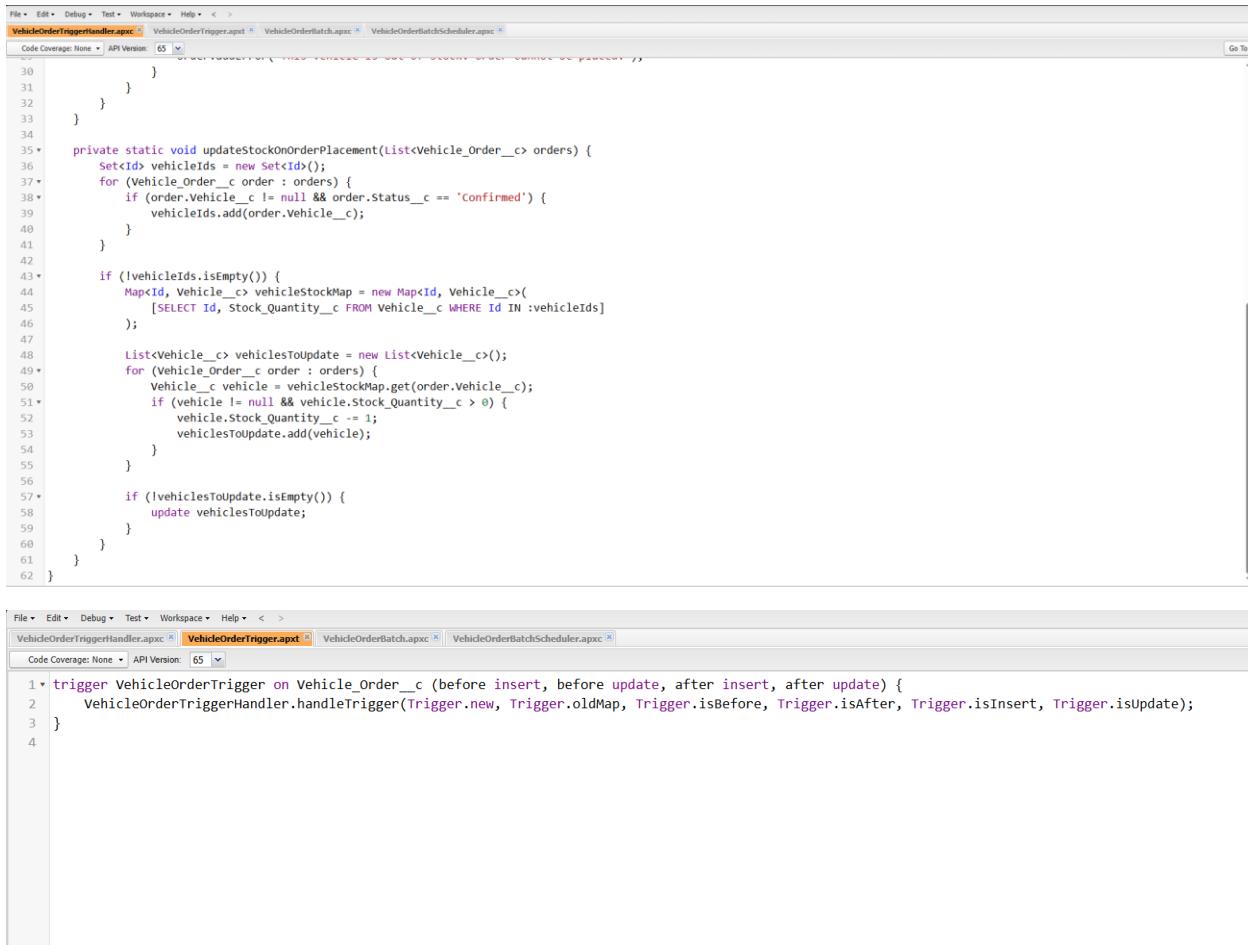
## Apex and Batch Class

```

File • Edit • Debug • Test • Workspace • Help • < >
VehicleOrderTriggerHandler.apc [ ] VehicleOrderTrigger.apc [ ] VehicleOrderBatch.apc [ ] VehicleOrderBatchScheduler.apc [ ]
Code Coverage: None • API Version: 65

1 * public class VehicleOrderTriggerHandler {
2
3     public static void handleTrigger(List<Vehicle_Order__c> newOrders, Map<Id, Vehicle_Order__c> oldOrders, Boolean isBefore, Boolean isAfter, Boolean isInsert, Boolean isUpdate) {
4         if (isBefore && (isInsert || isUpdate)) {
5             preventOrderIfOutOfStock(newOrders);
6         }
7
8         if (isAfter && (isInsert || isUpdate)) {
9             updateStockOnOrderPlacement(newOrders);
10        }
11    }
12
13    private static void preventOrderIfOutOfStock(List<Vehicle_Order__c> orders) {
14        Set<Id> vehicleIds = new Set<Id>();
15        for (Vehicle_Order__c order : orders) {
16            if (order.Vehicle__c != null) {
17                vehicleIds.add(order.Vehicle__c);
18            }
19        }
20
21        if (!vehicleIds.isEmpty()) {
22            Map<Id, Vehicle__c> vehicleStockMap = new Map<Id, Vehicle__c>(
23                [SELECT Id, Stock_Quantity__c FROM Vehicle__c WHERE Id IN :vehicleIds]
24            );
25
26            for (Vehicle_Order__c order : orders) {
27                Vehicle__c vehicle = vehicleStockMap.get(order.Vehicle__c);
28                if (vehicle != null && vehicle.Stock_Quantity__c <= 0) {
29                    orderaddError('This vehicle is out of stock. Order cannot be placed.');
30                }
31            }
32        }
33    }
}

```



The image shows a screenshot of a code editor interface with two tabs open:

- VehicleOrderTriggerHandler.apxc**: This tab contains Java code for a trigger handler. It includes methods for handling triggers on the `Vehicle_Order__c` object, specifically for insert and update operations. The code uses `Set<Id>` to store vehicle IDs and `Map<Id, Vehicle__c>` to map vehicle IDs to their corresponding records. It also performs a database query to get vehicle stock quantities.
- VehicleOrderTrigger.apxt**: This tab contains Apex trigger code. It defines a trigger named `VehicleOrderTrigger` on the `Vehicle_Order__c` object. The trigger is triggered before insert and update operations. It calls the `handleTrigger` method of the `VehicleOrderTriggerHandler` class, passing in parameters related to the trigger event.

```
VehicleOrderTriggerHandler.apxc
private static void updateStockOnOrderPlacement(List<Vehicle_Order__c> orders) {
    Set<Id> vehicleIds = new Set<Id>();
    for (Vehicle_Order__c order : orders) {
        if (order.Vehicle__c != null && order.Status__c == 'Confirmed') {
            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]
        );
        List<Vehicle__c> vehiclesToUpdate = new List<Vehicle__c>();
        for (Vehicle_Order__c order : orders) {
            Vehicle__c vehicle = vehicleStockMap.get(order.Vehicle__c);
            if (vehicle != null && vehicle.Stock_Quantity__c > 0) {
                vehicle.Stock_Quantity__c -= 1;
                vehiclesToUpdate.add(vehicle);
            }
        }
        if (!vehiclesToUpdate.isEmpty()) {
            update vehiclesToUpdate;
        }
    }
}
}

VehicleOrderTrigger.apxt
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);
}
```

File ▾ Edit ▾ Debug ▾ Test ▾ Workspace ▾ Help ▾ < >

VehicleOrderTriggerHandler.apxc VehicleOrderTrigger.apxt **VehicleOrderBatch.apxc** VehicleOrderBatchScheduler.apxc

Code Coverage: None API Version: 65

```
1 global class VehicleOrderBatch implements Database.Batchable<sObject> {
2
3     global Database.QueryLocator start(Database.BatchableContext bc) {
4         return Database.getQueryLocator([
5             SELECT Id, Status__c, Vehicle__c FROM Vehicle_Order__c WHERE Status__c = 'Pending'
6         ]);
7     }
8
9     global void execute(Database.BatchableContext bc, List<Vehicle_Order__c> orderList) {
10        Set<Id> vehicleIds = new Set<Id>();
11        for (Vehicle_Order__c order : orderList) {
12            if (order.Vehicle__c != null) {
13                vehicleIds.add(order.Vehicle__c);
14            }
15        }
16
17        if (!vehicleIds.isEmpty()) {
18            Map<Id, Vehicle__c> vehicleStockMap = new Map<Id, Vehicle__c>(
19                [SELECT Id, Stock_Quantity__c FROM Vehicle__c WHERE Id IN :vehicleIds]
20            );
21
22            List<Vehicle_Order__c> ordersToUpdate = new List<Vehicle_Order__c>();
23            List<Vehicle__c> vehiclesToUpdate = new List<Vehicle__c>();
24
25            for (Vehicle_Order__c order : orderList) {
26                Vehicle__c vehicle = vehicleStockMap.get(order.Vehicle__c);
27                if (vehicle != null && vehicle.Stock_Quantity__c > 0) {
28                    order.Status__c = 'Confirmed';
29                    vehicle.Stock_Quantity__c -= 1;
30                    ordersToUpdate.add(order);
31                    vehiclesToUpdate.add(vehicle);
32                }
33            }

```

File ▾ Edit ▾ Debug ▾ Test ▾ Workspace ▾ Help ▾ < >

VehicleOrderTriggerHandler.apxc VehicleOrderTrigger.apxt **VehicleOrderBatch.apxc** VehicleOrderBatchScheduler.apxc

Code Coverage: None API Version: 65

```
11    for (Vehicle_Order__c order : orderList) {
12        if (order.Vehicle__c != null) {
13            vehicleIds.add(order.Vehicle__c);
14        }
15    }
16
17    if (!vehicleIds.isEmpty()) {
18        Map<Id, Vehicle__c> vehicleStockMap = new Map<Id, Vehicle__c>(
19            [SELECT Id, Stock_Quantity__c FROM Vehicle__c WHERE Id IN :vehicleIds]
20        );
21
22        List<Vehicle_Order__c> ordersToUpdate = new List<Vehicle_Order__c>();
23        List<Vehicle__c> vehiclesToUpdate = new List<Vehicle__c>();
24
25        for (Vehicle_Order__c order : orderList) {
26            Vehicle__c vehicle = vehicleStockMap.get(order.Vehicle__c);
27            if (vehicle != null && vehicle.Stock_Quantity__c > 0) {
28                order.Status__c = 'Confirmed';
29                vehicle.Stock_Quantity__c -= 1;
30                ordersToUpdate.add(order);
31                vehiclesToUpdate.add(vehicle);
32            }
33        }
34
35        if (!ordersToUpdate.isEmpty()) update ordersToUpdate;
36        if (!vehiclesToUpdate.isEmpty()) update vehiclesToUpdate;
37    }
38
39
40    global void finish(Database.BatchableContext bc) {
41        System.debug('Vehicle order batch job completed.');
42    }
43}
```

File ▾ Edit ▾ Debug ▾ Test ▾ Workspace ▾ Help ▾ < >

VehicleOrderTriggerHandler.apxc VehicleOrderTrigger.apxt VehicleOrderBatch.apxc **VehicleOrderBatchScheduler.apxc**

Code Coverage: None API Version: 65

```
1  global class VehicleOrderBatchScheduler implements Schedulable {
2      global void execute(SchedulableContext sc) {
3          VehicleOrderBatch batchJob = new VehicleOrderBatch();
4          Database.executeBatch(batchJob, 50); // 50 = batch size
5      }
6  }
```