TripAdvisor E-Management

User Story: The TripAdvisor E-Management app with the salesforce works with your all-in-one travel companion, empowering you to plan, book, and make the most of the trips. Discover millions of reviews and insights shared by fellow travelers, helping you make informed decisions for every aspect of your journey. Whether you're seeking the perfect hotel, top-rated restaurants, must-visit attractions, or the best travel deals, TripAdvisor has you covered

Acceptance Criteria:

- As the Salesforce User we have to manage the data for the Hotels, Flights, and Food Options for this we have to create some automation for simplification.

- To ensure that when a new Food Option is added or updated, the corresponding Hotel's information is updated accordingly. For example, you might want to maintain a total count of food options for each hotel.

- Also there is automation for the customer benefits if the there buying amount is with respect to some amount then they will get some discounts on their bill

- For the flights there schedule process being involved where the customer who has booked the flight will get the reminder mail alert for knowing proper timing of the flight before 24 hrs it's important to manage the in a good way.

- The system should provide confirmation or notification to the user upon successful sending of the email.

Solution: For the Above requirements of TripAdvisor we have created the solutions by creating the custom objects and Fields the Custom Objects that are created are Hotels, Food Options, Customer & Flights. For the Automation we have used here a flow and triggers and for scheduling the email alerts we have created the Apex Schedulable class so email alerts will be created.

Activity 1 : Create Hotel Object

Use Case: Hotel Object is created to ensure that when a new Food Option is added or updated with the necessary information

1. Enter label : Hotel
2. Plural Name : Hotels
3. Data Type : (text)
4. Field Name : Hotel Name
5. Click Allow Reports
6. Allow Search → Save

With Above References Create following Object

Food Option → Data Type ⇒ Auto Number → Format→ FO - {0000}

Flight → Data Type ⇒ Auto Number → Format→ FL- {0000}

Customer → Text → Field Name → Customer Name

Activity 2: Create Fields in the Hotel Object

Sr. No.	**Field Name**	**Data Type**
1	**TotalFoodOptions**	**Number**
2	**Date**	**Date**

Activity 3 : Create Fields for Food Option

Sr. No.	**Field Name**	**Data Type**
1	**Name**	**Text**
2	**Hotel**	**Hotel(Lookup)**

3	**Food Amount**	**Currency**

Activity 4: Create Fields in the Flight Object

Sr. No.	**Field Name**	**Data Type**
1	**DepartureDateTime**	**Date/Time**
2	**Hotel Name**	**Hotel(Lookup)**

Activity 5: Create Fields in the Customer Object

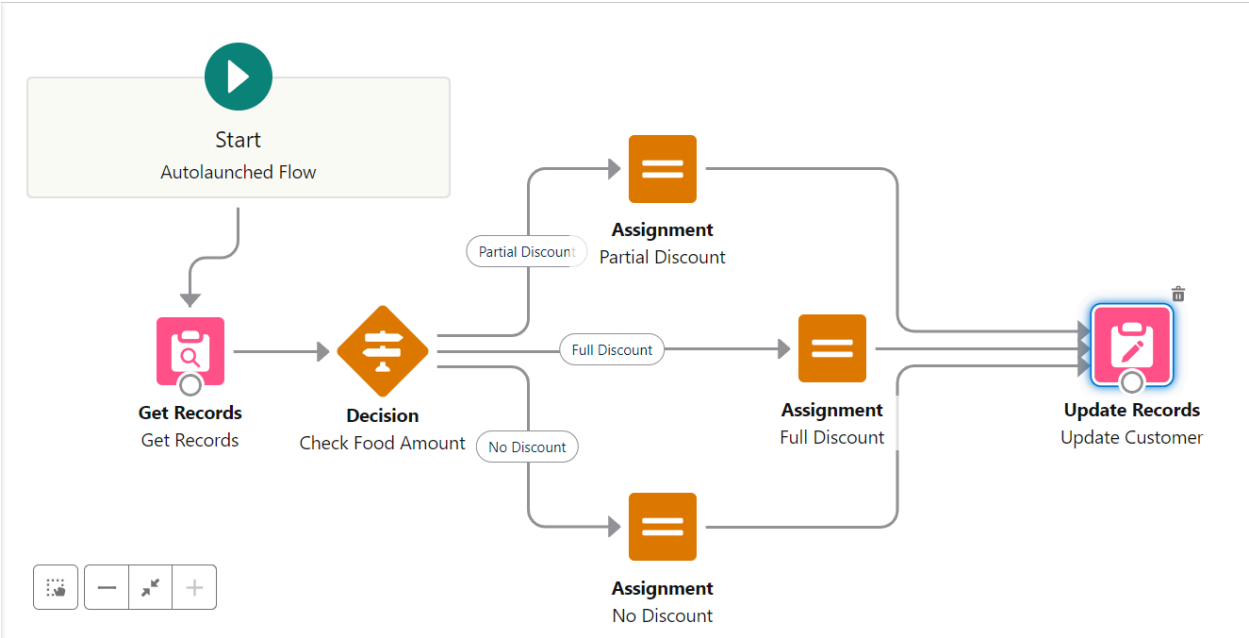
Sr. No.	**Field Name**	**Data Type**
1	**Customer Name (Standard Field)**	**Text**
2	**Discount Amount**	**Formula (Currency)**
3	**Discount Percent**	**Percentage**

Activity 5: Create Flow

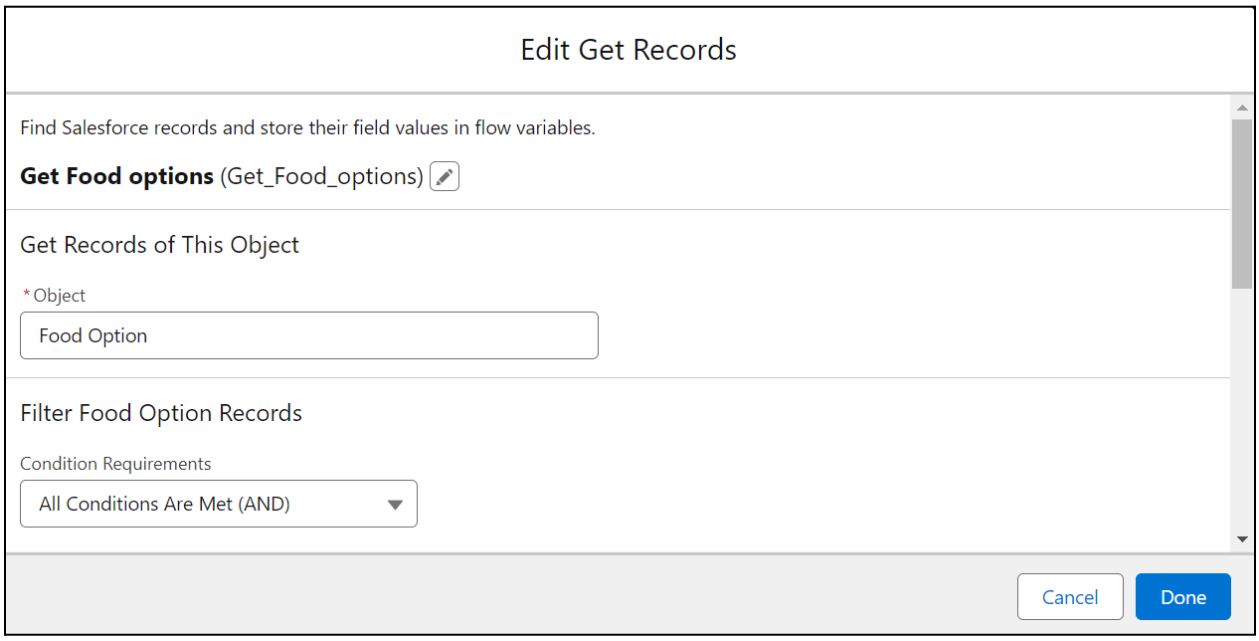
Create the Flow for the discount for customer when the Amount is greater than 3000 some some Amount of Discounts will be there if the Amount is between 1500 to 3000 so Some Amount of Discount will be there for them

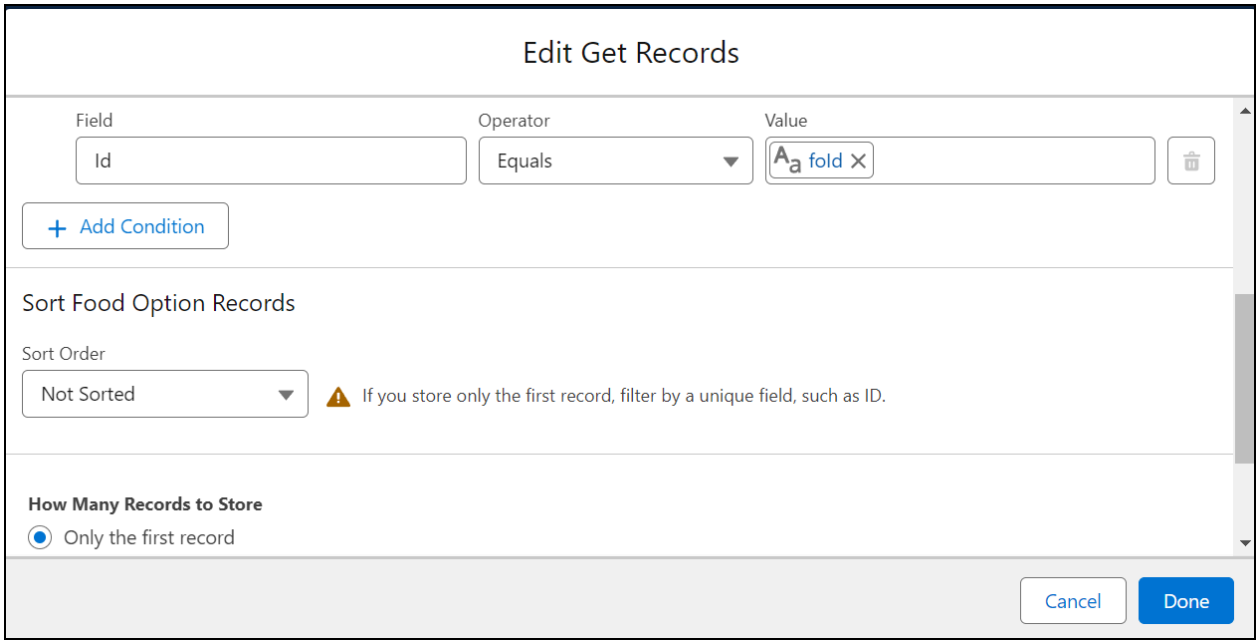
Work for It:

Create 3 variable →
Variable→Api name → foId → text → Available for Input
Variable→Api name → csId → text → Available for Input
Variable→Api name → discount → Number

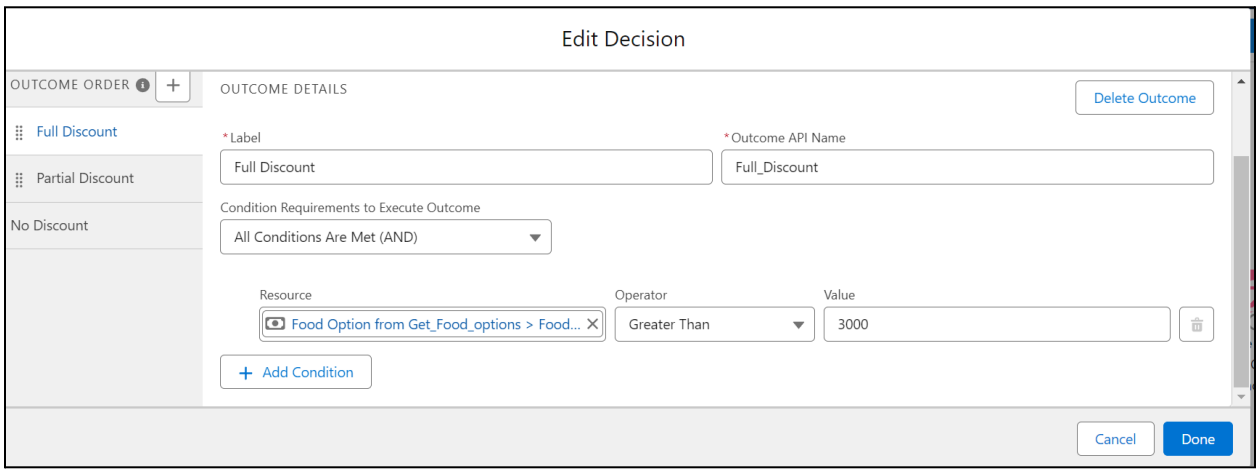


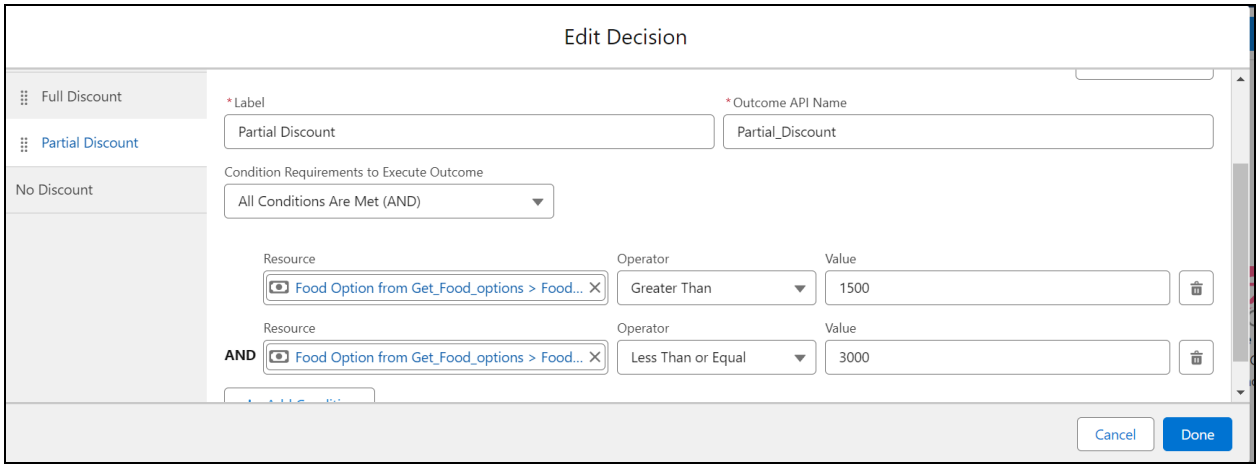
Flow Steps : Get Records



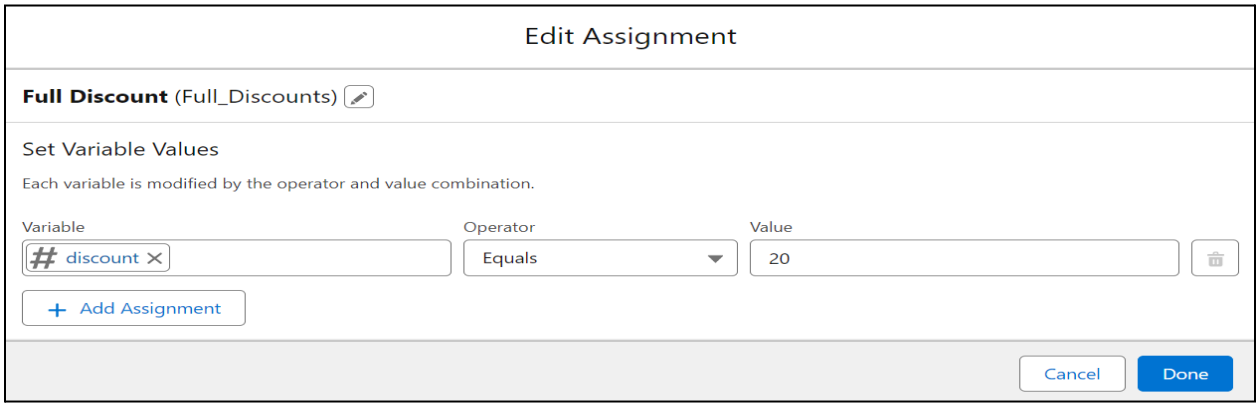


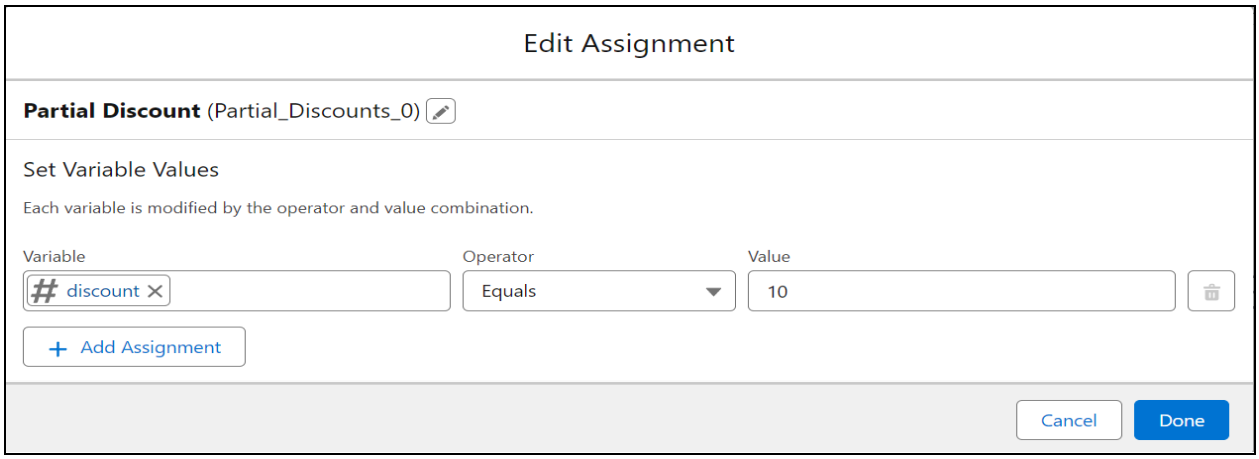
2. Decision Element: Create 2 Outcomes

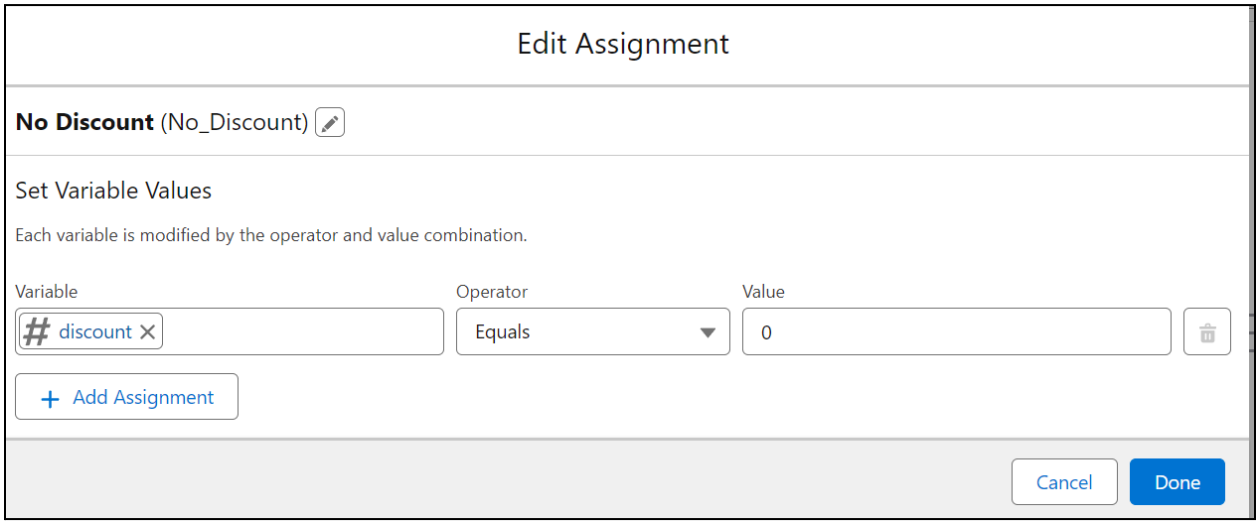




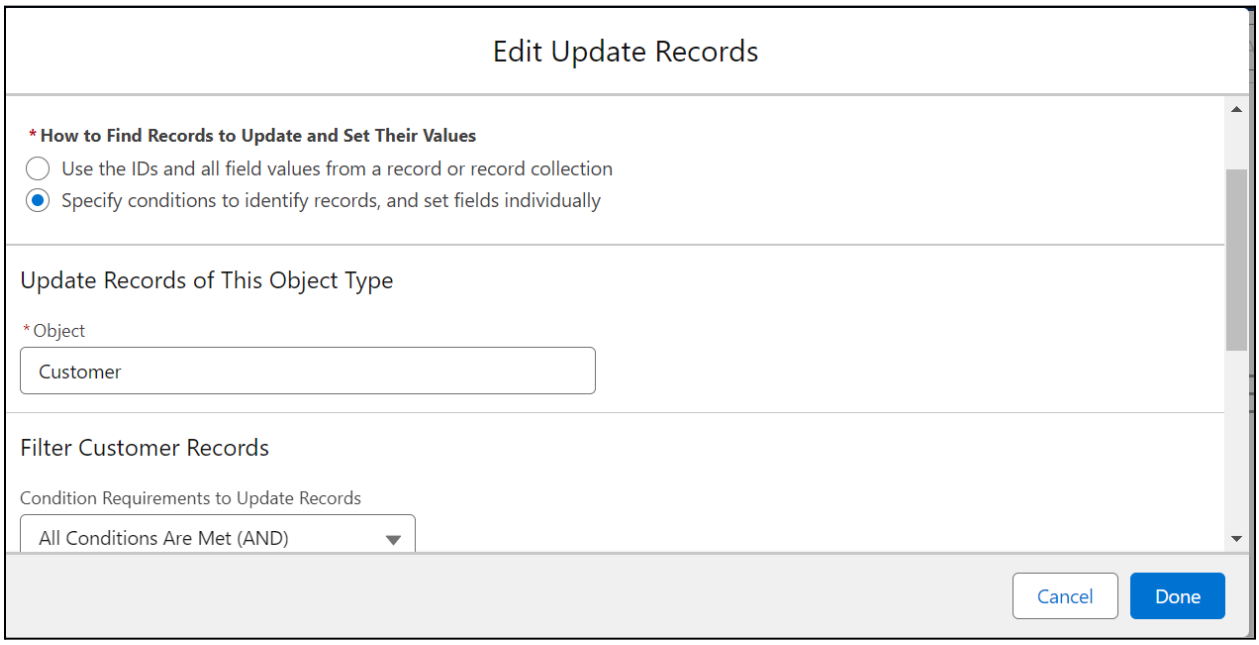
Take the 3 Assignments → Full Discount, Partial Discount & No Discount

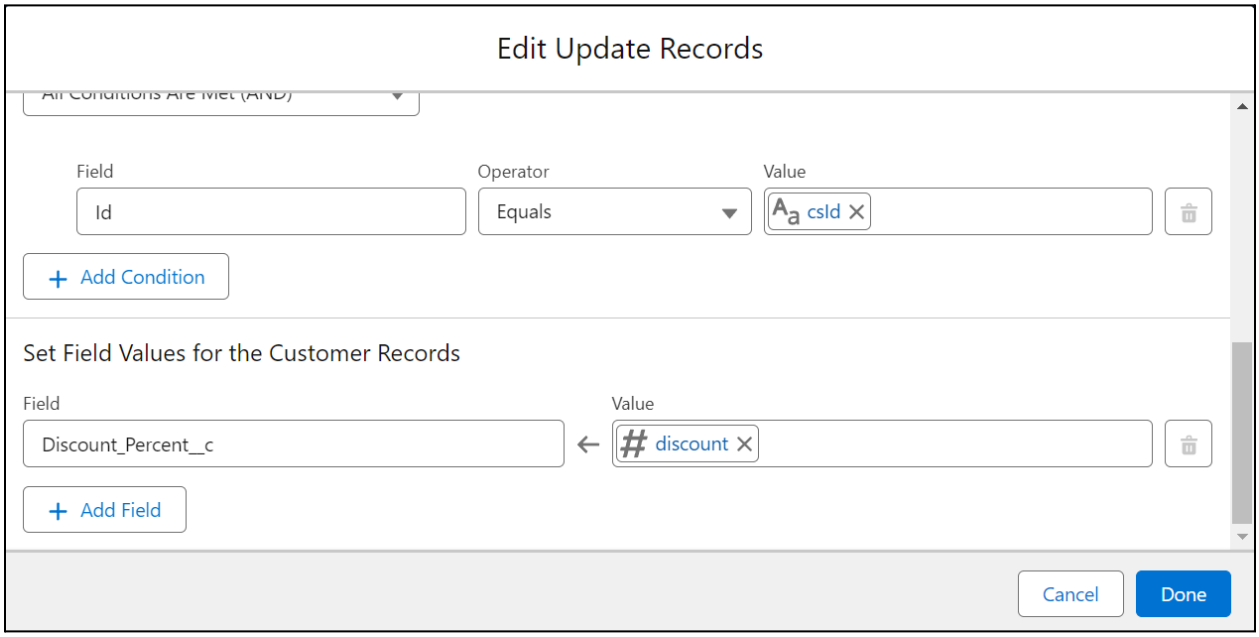



Edit Assignment
Partial Discount (Partial_Discounts_0)
Set Variable Values
Each variable is modified by the operator and value combination.
Variable
Operator
Value
discount
Equals
10
Add Assignment
Cancel
Done


Edit Assignment
No Discount (No_Discount)
Set Variable Values
Each variable is modified by the operator and value combination.
Variable
Operator
Value
discount
Equals
0
Add Assignment
Cancel
Done

Update Record Element





Apex Triggers:

Scenario: In the Hotel you have to ensure that when a new Food Option is added or updated, the corresponding Hotel's information is updated accordingly. For example, you might want to maintain a total count of food options for each hotel. To manage the things properly with perspective to the Hotel things should be clearly manageable for making the food options available with respect to hotels

Apex Trigger With Handler

```
public class FoodOptionTriggerHandler {

   // Method to update Hotel Information Based on Food Options
   public static void updateHotelInformation(List<Food_Option__c> newFoodOptions) {
      Set<Id> hotelIdsToUpdate = new Set<Id>();

      // Collect unique Hotel Ids affected by food options changes
      for (Food_Option__c foodOptions : newFoodOptions) {
         hotelIdsToUpdate.add(foodOptions.Hotel__c);
      }

      // Update hotel information based on food options
            List<Hotel__c> hotelsToUpdate = [SELECT Id, Name, TotalFoodOptions__c FROM
Hotel__c WHERE ID IN :hotelIdsToUpdate];
      for (Hotel__c hotel : hotelsToUpdate) {
         // Recalculate total food options count
               Integer totalFoodOptions = [SELECT COUNT() FROM Food_Option__c WHERE
Hotel__c = :hotel.Id];
         hotel.TotalFoodOptions__c = totalFoodOptions;
      }
      // Update hotels with new total food options count
      update hotelsToUpdate;
   }

   // Method to handle when the Hotel field is updated
    public static void handleHotelFieldChange(Map<Id, Food_Option__c> oldFoodOptionsMap,
List<Food_Option__c> newFoodOptions) {
      Set<Id> oldHotelIds = new Set<Id>();
```

```
        Set<Id> newHotelIds = new Set<Id>();

        // Collect old and new Hotel IDs where the field has changed
        for (Food_Option__c newFoodOption : newFoodOptions) {
            Food_Option__c oldFoodOption = oldFoodOptionsMap.get(newFoodOption.Id);

            // Check if the Hotel__c field has changed
            if (newFoodOption.Hotel__c != oldFoodOption.Hotel__c) {
                oldHotelIds.add(oldFoodOption.Hotel__c);
                newHotelIds.add(newFoodOption.Hotel__c);
            }
        }

        // Combine old and new Hotel IDs for updates
        Set<Id> hotelIdsToUpdate = new Set<Id>();
        hotelIdsToUpdate.addAll(oldHotelIds);
        hotelIdsToUpdate.addAll(newHotelIds);

        // Update hotel information for affected hotels
        if (!hotelIdsToUpdate.isEmpty()) {
            List<Hotel__c> hotelsToUpdate = [SELECT Id, TotalFoodOptions__c FROM Hotel__c
WHERE Id IN :hotelIdsToUpdate];
            for (Hotel__c hotel : hotelsToUpdate) {
                Integer totalFoodOptions = [SELECT COUNT() FROM Food_Option__c WHERE
Hotel__c = :hotel.Id];
                hotel.TotalFoodOptions__c = totalFoodOptions;
            }
            update hotelsToUpdate;
        }
    }
}
```

```apex
public class FoodOptionTriggerHandler {

    // Method to update Hotel Information Based on Food Options
    public static void updateHotelInformation(List<Food_Option__c> newFoodOptions) {
        Set<Id> hotelIdsToUpdate = new Set<Id>();

        // Collect unique Hotel Ids affected by food options changes
        for (Food_Option__c foodOptions : newFoodOptions) {
            hotelIdsToUpdate.add(foodOptions.Hotel__c);
        }

        // Update hotel information based on food options
        List<Hotel__c> hotelsToUpdate = [SELECT Id, Name, TotalFoodOptions__c FROM Hotel__c WHERE ID IN :hotelIdsToUpdate];
        for (Hotel__c hotel : hotelsToUpdate) {
            // Recalculate total food options count
            Integer totalFoodOptions = [SELECT COUNT() FROM Food_Option__c WHERE Hotel__c = :hotel.Id];
            hotel.TotalFoodOptions__c = totalFoodOptions;
        }
        // Update hotels with new total food options count
        update hotelsToUpdate;
    }

    // Method to handle when the Hotel field is updated
    public static void handleHotelFieldChange(Map<Id, Food_Option__c> oldFoodOptionsMap, List<Food_Option__c> newFoodOptions) {
        Set<Id> oldHotelIds = new Set<Id>();
        Set<Id> newHotelIds = new Set<Id>();

        // Collect old and new Hotel IDs where the field has changed
        for (Food_Option__c newFoodOption : newFoodOptions) {
            Food_Option__c oldFoodOption = oldFoodOptionsMap.get(newFoodOption.Id);

            // Check if the Hotel__c field has changed
            if (newFoodOption.Hotel__c != oldFoodOption.Hotel__c) {
                oldHotelIds.add(oldFoodOption.Hotel__c);
                newHotelIds.add(newFoodOption.Hotel__c);
            }
        }

        // Combine old and new Hotel IDs for updates
        Set<Id> hotelIdsToUpdate = new Set<Id>();
        hotelIdsToUpdate.addAll(oldHotelIds);
        hotelIdsToUpdate.addAll(newHotelIds);

        // Update hotel information for affected hotels
        if (!hotelIdsToUpdate.isEmpty()) {
            List<Hotel__c> hotelsToUpdate = [SELECT Id, TotalFoodOptions__c FROM Hotel__c WHERE Id IN :hotelIdsToUpdate];
            for (Hotel__c hotel : hotelsToUpdate) {
                Integer totalFoodOptions = [SELECT COUNT() FROM Food_Option__c WHERE Hotel__c = :hotel.Id];
                hotel.TotalFoodOptions__c = totalFoodOptions;
            }
            update hotelsToUpdate;
        }
    }
}
```

Trigger

```apex
trigger FoodOptionTrigger on Food_Option__c (after insert,after update) {
   If(trigger.isInsert  && trigger.isAfter){
    FoodOptionTriggerHandler.updateHotelInformation(trigger.new);
   }
   Else If(trigger.isAfter && trigger.isUpdate){
    FoodOptionTriggerHandler.handleHotelFieldChange(trigger.oldMap, trigger.new);
   }
}
```

```
trigger FoodOptionTrigger on Food_Option__c (after insert,after update) {
    If(trigger.isInsert  && trigger.isAfter){
     FoodOptionTriggerHandler.updateHotelInformation(trigger.new);
    }
    Else If(trigger.isAfter && trigger.isUpdate){
     FoodOptionTriggerHandler.handleHotelFieldChange(trigger.oldMap, trigger.new);
    }
}
```

Scenario With Apex Schedule Class

Create the Reminder mail for the customer who has booked the flight according to that booking set the Apex schedule so mail will be sent prior to 24hrs.

Note: Please create the required field for Scheduled Apex Code

```
public class FlightReminderScheduledJob implements Schedulable {

   public void execute(SchedulableContext sc) {
      sendFlightReminders();
   }

   private void sendFlightReminders() {
      // Query for flights departing within the next 24 hours
         List<Flight__c> upcomingFlights = [SELECT Id, Name, DepartureDateTime__c FROM
Flight__c
                                    WHERE DepartureDateTime__c >= :DateTime.now()
                                    AND DepartureDateTime__c <= :DateTime.now().addDays(1)];

      for (Flight__c flight : upcomingFlights) {
```

```
        // Customize the logic to send reminder emails
        // For this example, we'll print a log message; replace this with your email sending logic.
                System.debug('Sending reminder email for Flight ' + flight.Name + ' to ' +
flight.ContactEmail__c);

        // Example: Send email using Messaging.SingleEmailMessage
        Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();
        email.setToAddresses(new List<String>{ flight.ContactEmail__c });
        email.setSubject('Flight Reminder: ' + flight.Name);
        email.setPlainTextBody('This is a reminder for your upcoming flight ' + flight.Name +
                        ' departing on ' + flight.DepartureDateTime__c);
        Messaging.sendEmail(new List<Messaging.SingleEmailMessage>{ email });
    }
  }
}
```

The FlightReminderScheduledJob class implements the Schedulable interface, and the execute method is where you put the logic to send reminder emails.

The sendFlightReminders method queries for flights departing within the next 24 hours. You can customize the query based on your specific requirements.

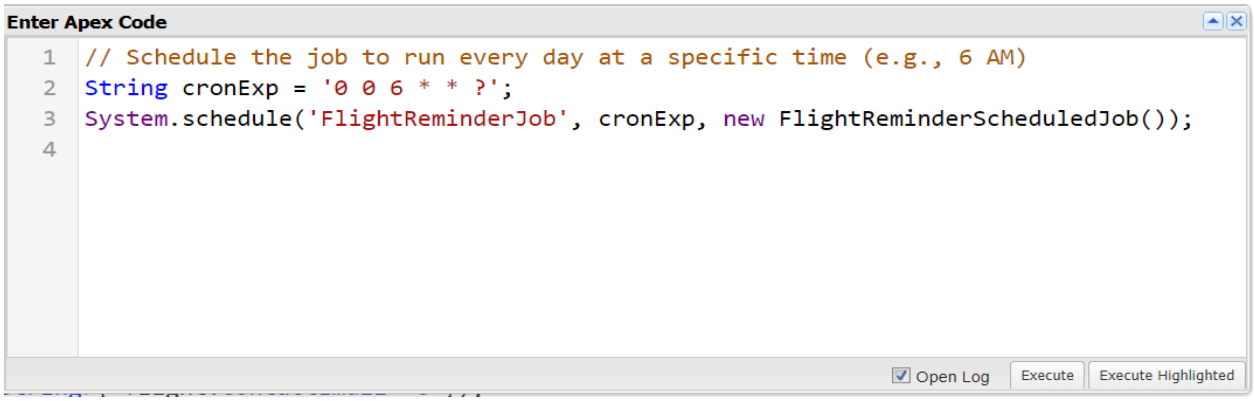
Create the Apex code in an anonymous Window to execute the Apex Code

FlightReminderScheduledJob.apxc

Code Coverage: None API Version: 59

```
1  public class FlightReminderScheduledJob implements Schedulable {
2
3      public void execute(SchedulableContext sc) {
4          sendFlightReminders();
5      }
6
7      private void sendFlightReminders() {
8          // Query for flights departing within the next 24 hours
9          List<Flight__c> upcomingFlights = [SELECT Id, Name, DepartureDateTime__c FROM Flight__c
10                                             WHERE DepartureDateTime__c >= :DateTime.now()
11                                             AND DepartureDateTime__c <= :DateTime.now().addDays(1)];
12
13         for (Flight__c flight : upcomingFlights) {
14             // Customize the logic to send reminder emails
15             // For this example, we'll print a log message; replace this with your email sending logic.
16             System.debug('Sending reminder email for Flight ' + flight.Name + ' to ' + flight.ContactEmail__c);
17
18             // Example: Send email using Messaging.SingleEmailMessage
19             Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();
20             email.setToAddresses(new List<String>{ flight.ContactEmail__c });
21             email.setSubject('Flight Reminder: ' + flight.Name);
22             email.setPlainTextBody('This is a reminder for your upcoming flight ' + flight.Name +
23                                    ' departing on ' + flight.DepartureDateTime__c);
24             Messaging.sendEmail(new List<Messaging.SingleEmailMessage>{ email });
25         }
26     }
27 }
```

```
// Schedule the job to run every day at a specific time (e.g., 6 AM)
String cronExp = '0 0 6 * * ?';
System.schedule('FlightReminderJob', cronExp, new FlightReminderScheduledJob());
```



Conclusion: We have Created this Customization process for the proper flow of the business if tripAdvisor where they can easily access the Hotel requirement then food options and also the ease for the customers with the preferable discount with there Amount limits this process helps to save time from multiple manual processes.

This Project of TripAdvisor is used for managing this Requirements for the trip with the Automation and Asynchronous Process