Optum

Salesforce COE – ChatGPT & GitHub Copilot Study

Feb 2023

Make a promise,
keep a promise

AI Tools for Salesforce Coding

Objective:

Comparative analysis of ChatGPT vs GitHub Co-Pilot in the context of Salesforce Eco-system ensuring efficiency of our Salesforce developers.

What is ChatGPT

- It is an AI (Artificial Intelligence) chatbot or LLM (Large Language Model) that uses natural language processing (NLP) to interact through simple language
- OpenAI is the company behind ChatGPT, and they are heavily backed by Microsoft

What is GitHub Copilot

- GitHub Copilot is a cloud-based artificial intelligence tool developed by GitHub and OpenAI to assist users of Visual Studio Code, Visual Studio, Neovim, and JetBrains integrated development environments by autocompleting code

© 2022 Optum, Inc. All rights reserved.

Key Use Cases

As part of our POC to test out Open AI based ChatGPT and GitHub Copilot, more than 35 Use cases were executed outside the Optum network. The key use cases for code generation are highlighted below

Trigger/Apex Coding

Triggers

- Map contact address details into account object
- Create Contact record when an Account is created.
- Prevent recursion on Trigger
- Bulkification of Apex class for creating Accounts/Contacts in a single DML statement

Batch Class

- Create an apex batch class that will append “IamTestingCopilot” to all contacts
- apex class to fetch 1 million records using batch class
- Scheduling an Apex Batch every 4 hours

ChatGPT & GitHub Copilot.

LWC/Aura/VF/Declarative Options

LWC/AURA Component

- Create LWC/AURA component with First Name and Last Name as fields
- Write a code for creating an aura component with submit and cancel button
- Test class for LWC

VF Pages

- Create visual force page to render 2 images side by side
- Rendering VF pages as PDF

Declarative Options

- Delete related contacts whenever its associated account is deleted using Lightning Flows
- Send an email and create a task when Opportunity is closed
- Throw an error message whenever Account Number contains alphabets using Validation Rule.

© 2022 Optum, Inc. All rights reserved.

Use Case Analysis

ChatGPT

01 **Triggers**
ChatGPT writes fully functional code aligned with Business context.

02 **Batch Class**
Works well with Low, Medium and High complexity scenarios

03 **LWC/AURA**
ChatGPT generates helpful solution with component creation, JavaScript logic and required code base..

04 **VF Pages**
Gives out well defined code base with next steps.

05 **Declarative Tools**
ChatGPT provides step by step guide for creating flows & Validation Rule.

GitHub Copilot

01 **Triggers**
GitHub Copilot is more localized and meant for small snippets.

02 **Batch Class**
Works well with Low & Medium complexity scenarios

03 **LWC/AURA**
Good for adding basic skeleton. For complex build of Visualforce, Aura & LWC, copilot fails to add complex logic

04 **VF Pages**
Copilot fails to understand the context while suggesting snippets

05 **Declarative Tools**
Copilot does not support Lightning flows & Validation Rules

© 2022 Optum, Inc. All rights reserved.

Summary

Our View

ChatGPT is the winner when it comes to accuracy and agility of writing a code aligned with business requirements. Both ChatGPT and GitHub Copilot would need to be connected to one of our Salesforce org for deeper insights into quality of code generated.

ChatGPT model

Its too early to comment on the real time usage and code generation, however ChatGPT produces more accurate and Business use case related code.

GitHub Copilot model

GitHub Copilot is very localized and works seamlessly with Visual Studio Code. It Prompts and suggest various coding solutions based on existing coding and methods residing in code base.

Core Limitations

- Unable to follow design patterns
- AI Engine - Data source is not yet configurable for ChatGPT
- Compared to Java & Python, Salesforce Code generation is more secondary

Model comparison guide

The guide below compares the two models to help you determine which is best for Salesforce Developers

	ChatGPT Results	GitHub Copilot Results
Promptness	Promptness of code is governed by refined keyword inputs to ChatGPT Open AI based bot.	Gives multiple coding solutions for same keyword search.
Accuracy	Code generated using appropriate keywords, produces contextual, accurate and relatable code aligned to Business needs	More accurate on localized code and small snippets.
Quality	Writes efficient Test Classes for the Apex classes. Lesser manual intervention required to refactor the test classes	Writes Test classes, however manual intervention is needed to modify the code base
Efficiency	Developers productivity likely to go up by 20-25% in case of brand-new components Developers productivity likely to go up by 5-10% in case of enhancing an existing components	Developers productivity likely to go up by 20-25% in case of brand-new components Developers productivity likely to go up by 5-10% in case of enhancing an existing components
Licensing (Prices mentioned are from the official website, not Optum Licensing Team)	ChatGPT – Free Edition with Limited Features ChatGPT Plus – 20$ Per User/Per Month ChartGPT Professional – 42 $ Per User/Per Month	CoPilot – Free Edition – 60 days CoPilot Enterprise – 10$ Per User/Per Month CoPilot Business – 19$ Per User/Per Month

© 2022 Optum, Inc. All rights reserved.

Optum

Optum is a registered trademark of Optum, Inc. in the U.S. and other jurisdictions. All other brand or product names are the property of their respective owners. Because we are continuously improving our products and services, Optum reserves the right to change specifications without prior notice. Optum is an equal opportunity employer.

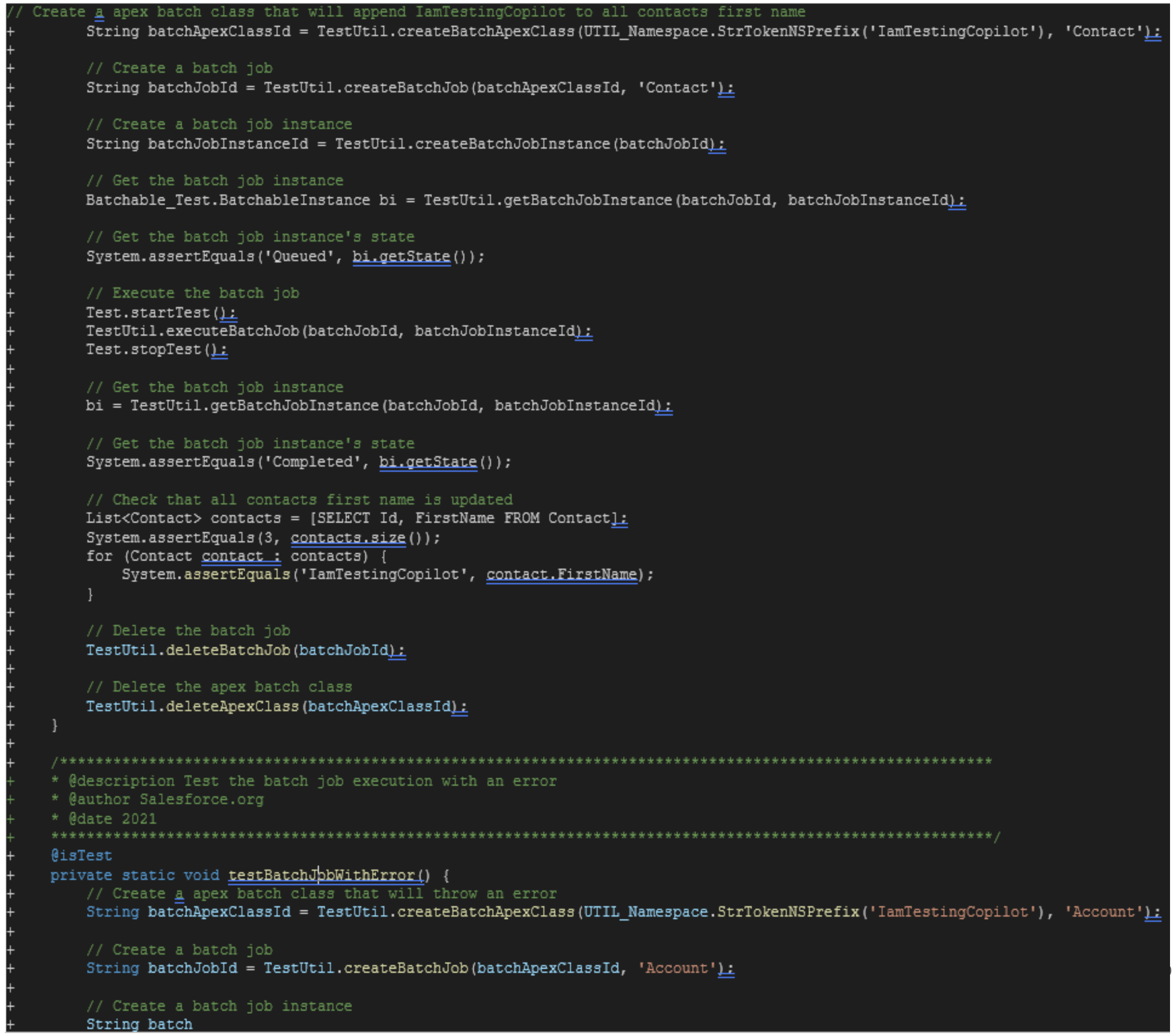
© 2022 Optum, Inc. All rights reserved.

Appendix

© 2022 Optum, Inc. All rights reserved.

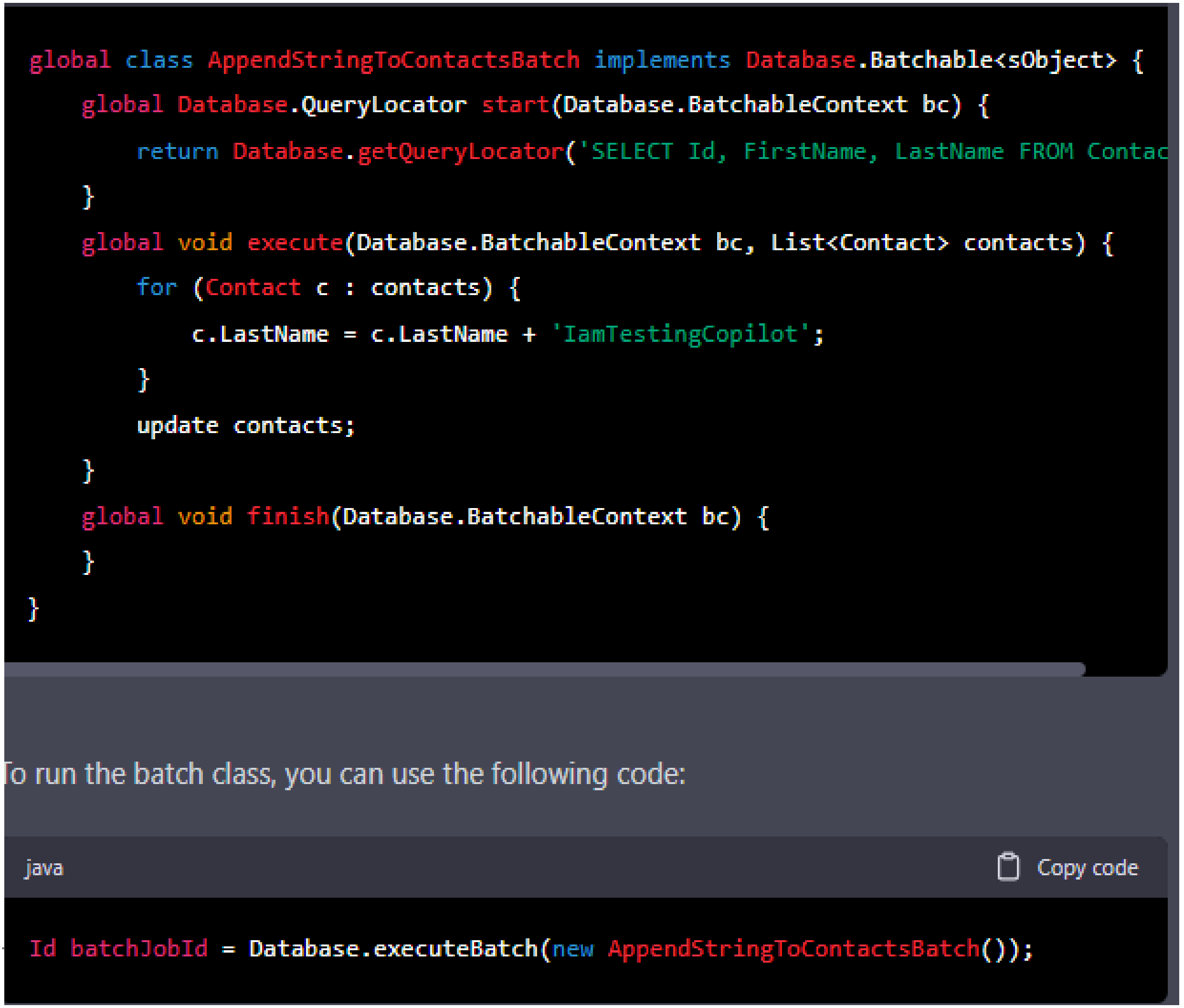
1. Create an apex batch class that will append “IamTestingCopilot” to all contacts.

Copilot



```
// Create a apex batch class that will append IamTestingCopilot to all contacts first name
+           String batchApexClassId = TestUtil.createBatchApexClass(UTIL_Namespace.StrTokenNSPrefix('IamTestingCopilot'), 'Contact');
+
+           // Create a batch job
+           String batchJobId = TestUtil.createBatchJob(batchApexClassId, 'Contact');
+
+           // Create a batch job instance
+           String batchJobInstanceId = TestUtil.createBatchJobInstance(batchJobId);
+
+           // Get the batch job instance
+           Batchable_Test.BatchableInstance bi = TestUtil.getBatchJobInstance(batchJobId, batchJobInstanceId);
+
+           // Get the batch job instance's state
+           System.assertEquals('Queued', bi.getState());
+
+           // Execute the batch job
+           Test.startTest();
+           TestUtil.executeBatchJob(batchJobId, batchJobInstanceId);
+           Test.stopTest();
+
+           // Get the batch job instance
+           bi = TestUtil.getBatchJobInstance(batchJobId, batchJobInstanceId);
+
+           // Get the batch job instance's state
+           System.assertEquals('Completed', bi.getState());
+
+           // Check that all contacts first name is updated
+           List<Contact> contacts = [SELECT Id, FirstName FROM Contact];
+           System.assertEquals(3, contacts.size());
+           for (Contact contact : contacts) {
+               System.assertEquals('IamTestingCopilot', contact.FirstName);
+           }
+
+           // Delete the batch job
+           TestUtil.deleteBatchJob(batchJobId);
+
+           // Delete the apex batch class
+           TestUtil.deleteApexClass(batchApexClassId);
+       }
+
+       /*******************************************************************************************
+       * @description Test the batch job execution with an error
+       * @author Salesforce.org
+       * @date 2021
+       ********************************************************************************************/
+       @isTest
+       private static void testBatchJobWithError() {
+           // Create a apex batch class that will throw an error
+           String batchApexClassId = TestUtil.createBatchApexClass(UTIL_Namespace.StrTokenNSPrefix('IamTestingCopilot'), 'Account');
+
+           // Create a batch job
+           String batchJobId = TestUtil.createBatchJob(batchApexClassId, 'Account');
+
+           // Create a batch job instance
+           String batch
```

ChatGPT



```
global class AppendStringToContactsBatch implements Database.Batchable<sObject> {
    global Database.QueryLocator start(Database.BatchableContext bc) {
        return Database.getQueryLocator('SELECT Id, FirstName, LastName FROM Contac
    }
    global void execute(Database.BatchableContext bc, List<Contact> contacts) {
        for (Contact c : contacts) {
            c.LastName = c.LastName + 'IamTestingCopilot';
        }
        update contacts;
    }
    global void finish(Database.BatchableContext bc) {
    }
}
```

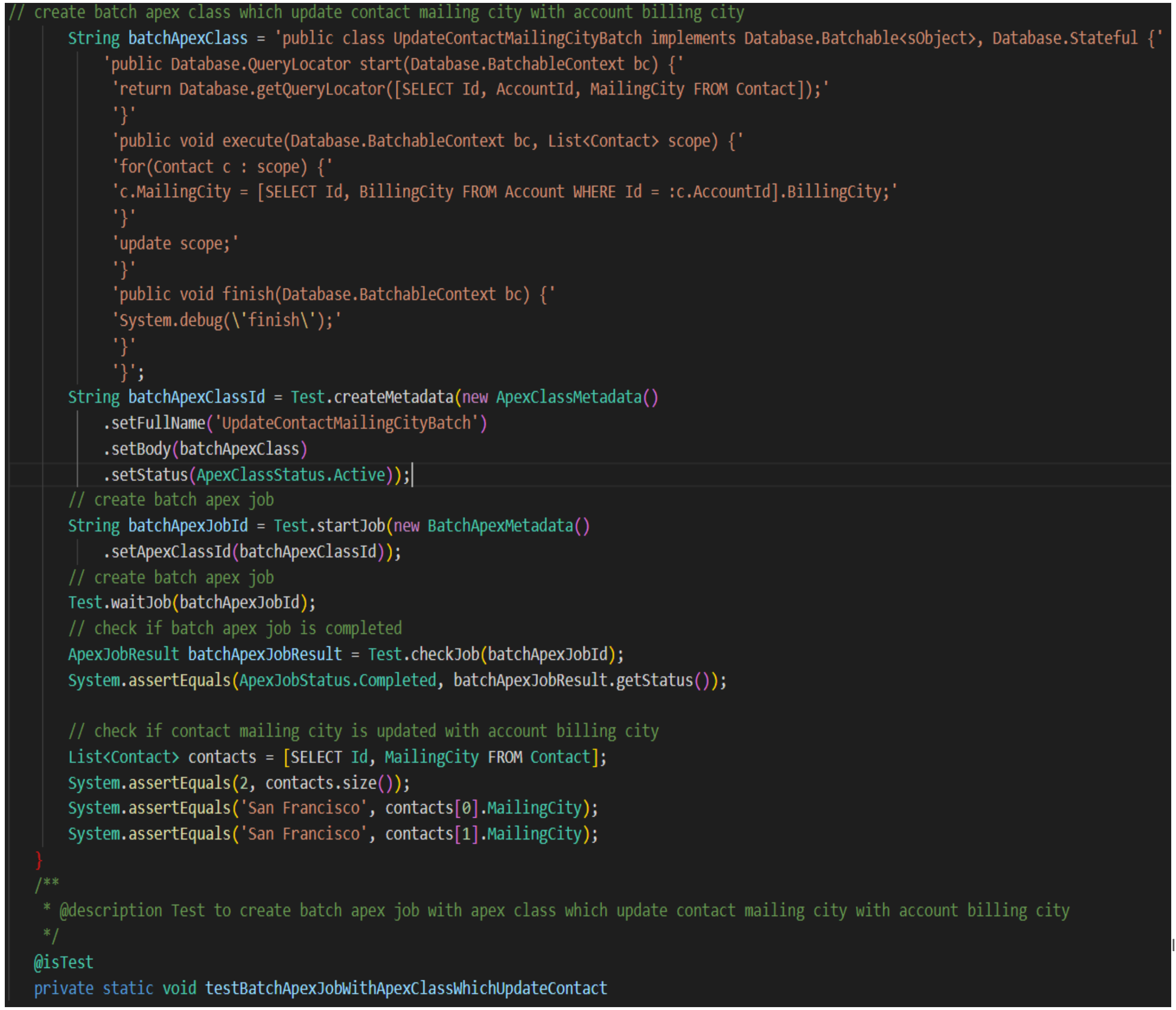
To run the batch class, you can use the following code:

```java
Id batchJobId = Database.executeBatch(new AppendStringToContactsBatch());
```

tum, Inc.

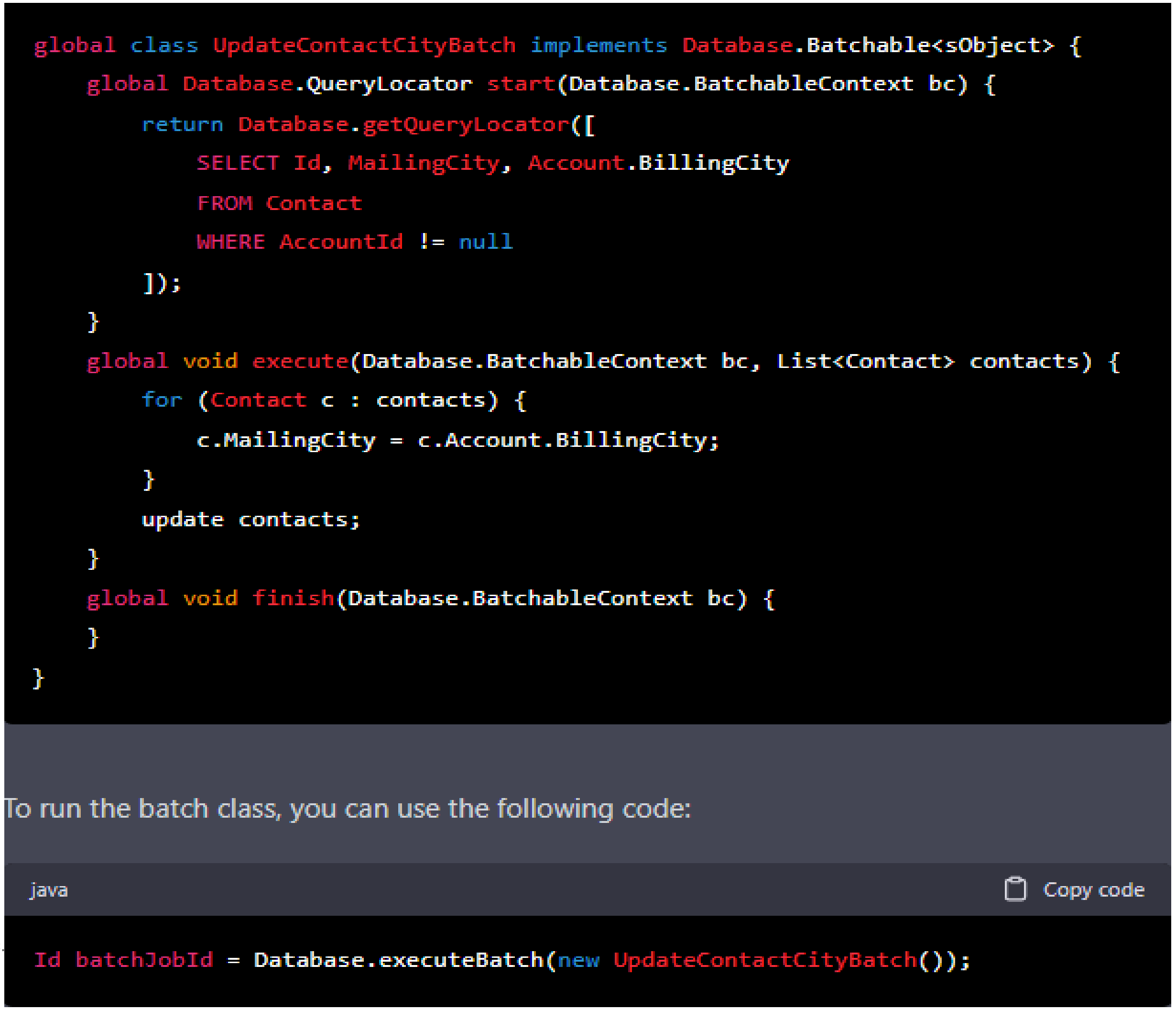
2. Create batch apex class which update contact mailing city with account billing city.

Copilot



```
// create batch apex class which update contact mailing city with account billing city
        String batchApexClass = 'public class UpdateContactMailingCityBatch implements Database.Batchable<sObject>, Database.Stateful {'
            'public Database.QueryLocator start(Database.BatchableContext bc) {'
             'return Database.getQueryLocator([SELECT Id, AccountId, MailingCity FROM Contact]);'
             '}'
             'public void execute(Database.BatchableContext bc, List<Contact> scope) {'
             'for(Contact c : scope) {'
             'c.MailingCity = [SELECT Id, BillingCity FROM Account WHERE Id = :c.AccountId].BillingCity;'
             '}'
             'update scope;'
             '}'
             'public void finish(Database.BatchableContext bc) {'
             'System.debug(\'finish\');'
             '}'
             '}';
        String batchApexClassId = Test.createMetadata(new ApexClassMetadata()
            .setFullName('UpdateContactMailingCityBatch')
            .setBody(batchApexClass)
            .setStatus(ApexClassStatus.Active));
        // create batch apex job
        String batchApexJobId = Test.startJob(new BatchApexMetadata()
            .setApexClassId(batchApexClassId));
        // create batch apex job
        Test.waitJob(batchApexJobId);
        // check if batch apex job is completed
        ApexJobResult batchApexJobResult = Test.checkJob(batchApexJobId);
        System.assertEquals(ApexJobStatus.Completed, batchApexJobResult.getStatus());

        // check if contact mailing city is updated with account billing city
        List<Contact> contacts = [SELECT Id, MailingCity FROM Contact];
        System.assertEquals(2, contacts.size());
        System.assertEquals('San Francisco', contacts[0].MailingCity);
        System.assertEquals('San Francisco', contacts[1].MailingCity);
    }
    /**
     * @description Test to create batch apex job with apex class which update contact mailing city with account billing city
     */
    @isTest
    private static void testBatchApexJobWithApexClassWhichUpdateContact
```

ChatGPT



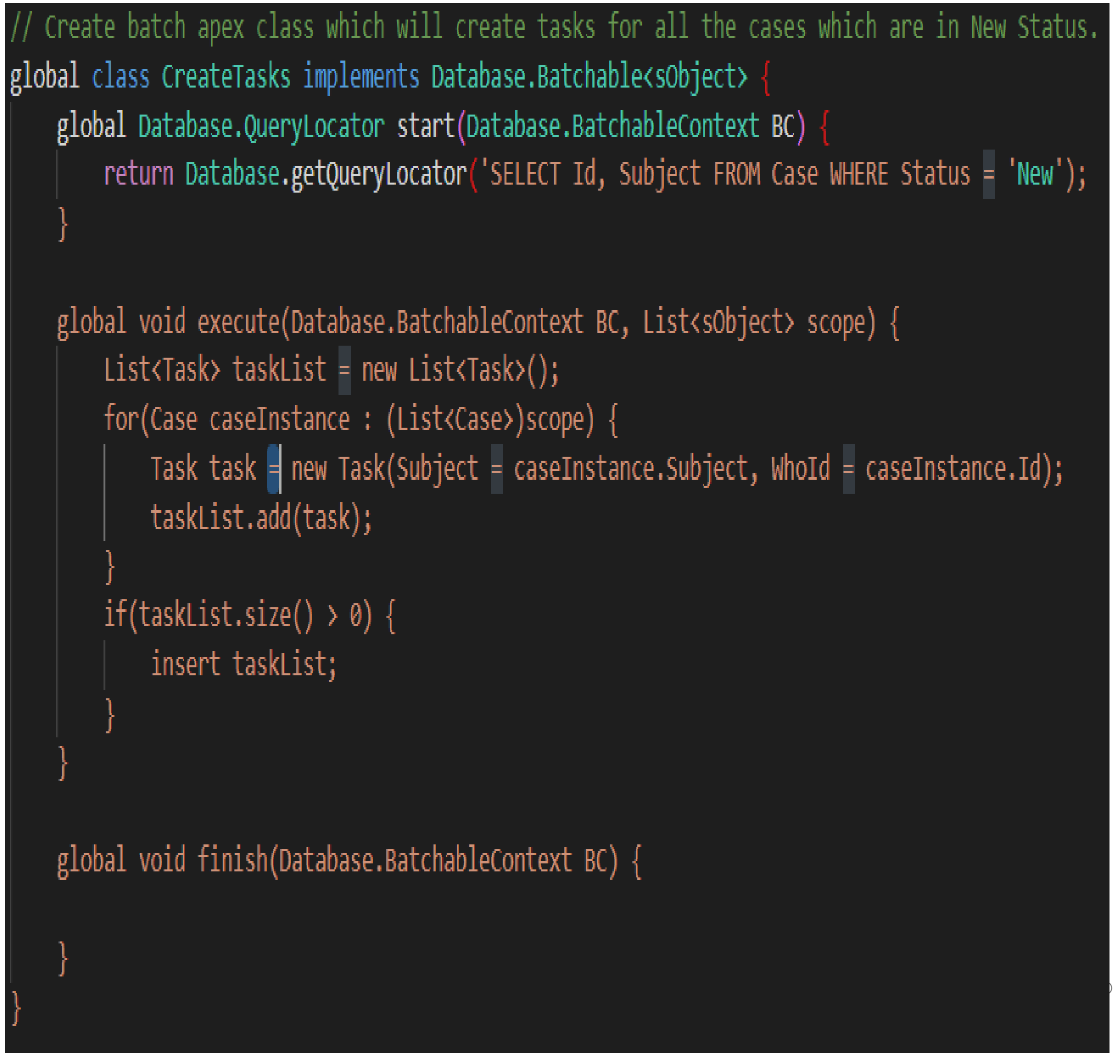
```
global class UpdateContactCityBatch implements Database.Batchable<sObject> {
    global Database.QueryLocator start(Database.BatchableContext bc) {
        return Database.getQueryLocator([
            SELECT Id, MailingCity, Account.BillingCity
            FROM Contact
            WHERE AccountId != null
        ]);
    }
    global void execute(Database.BatchableContext bc, List<Contact> contacts) {
        for (Contact c : contacts) {
            c.MailingCity = c.Account.BillingCity;
        }
        update contacts;
    }
    global void finish(Database.BatchableContext bc) {
    }
}
```

To run the batch class, you can use the following code:

```java
Id batchJobId = Database.executeBatch(new UpdateContactCityBatch());
```

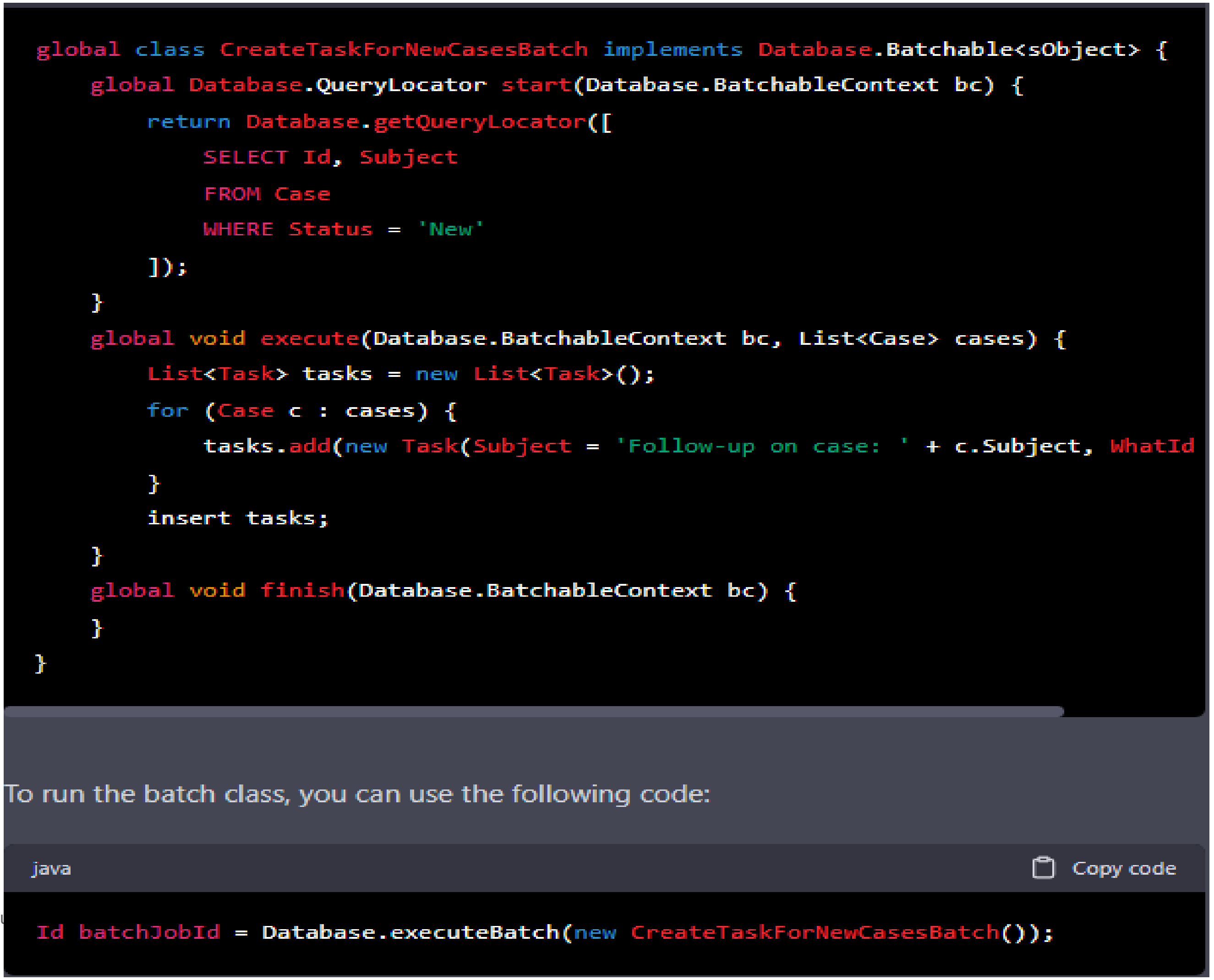
3. Create batch apex class which will create tasks for all the cases which are in New Status

Copilot



```
// Create batch apex class which will create tasks for all the cases which are in New Status.
global class CreateTasks implements Database.Batchable<sObject> {
    global Database.QueryLocator start(Database.BatchableContext BC) {
        return Database.getQueryLocator('SELECT Id, Subject FROM Case WHERE Status = 'New');
    }

    global void execute(Database.BatchableContext BC, List<sObject> scope) {
        List<Task> taskList = new List<Task>();
        for(Case caseInstance : (List<Case>)scope) {
            Task task = new Task(Subject = caseInstance.Subject, WhoId = caseInstance.Id);
            taskList.add(task);
        }
        if(taskList.size() > 0) {
            insert taskList;
        }
    }

    global void finish(Database.BatchableContext BC) {

    }
}
```

ChatGPT



```
global class CreateTaskForNewCasesBatch implements Database.Batchable<sObject> {
    global Database.QueryLocator start(Database.BatchableContext bc) {
        return Database.getQueryLocator([
            SELECT Id, Subject
            FROM Case
            WHERE Status = 'New'
        ]);
    }
    global void execute(Database.BatchableContext bc, List<Case> cases) {
        List<Task> tasks = new List<Task>();
        for (Case c : cases) {
            tasks.add(new Task(Subject = 'Follow-up on case: ' + c.Subject, WhatId
        }
        insert tasks;
    }
    global void finish(Database.BatchableContext bc) {
    }
}
```

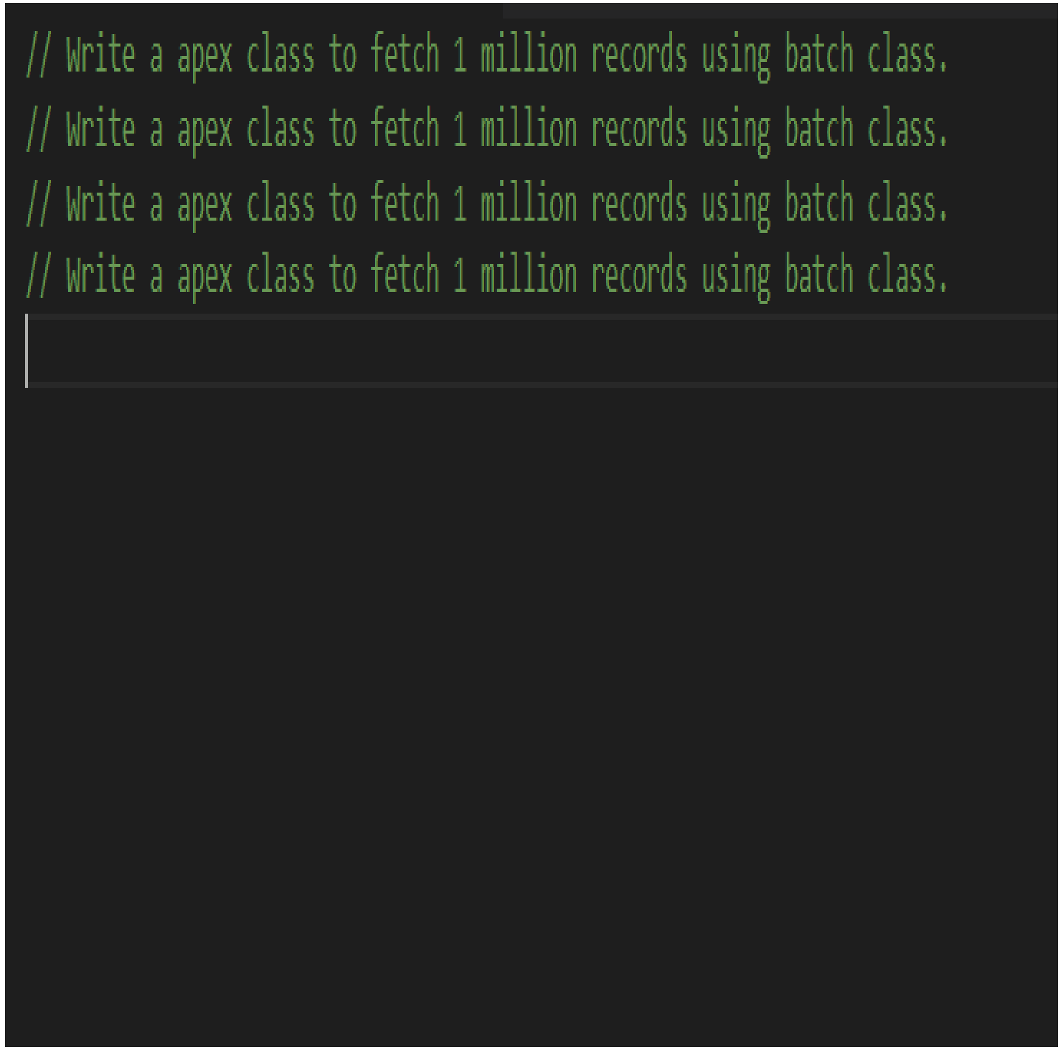
To run the batch class, you can use the following code:

java Copy code

```java
Id batchJobId = Database.executeBatch(new CreateTaskForNewCasesBatch());
```

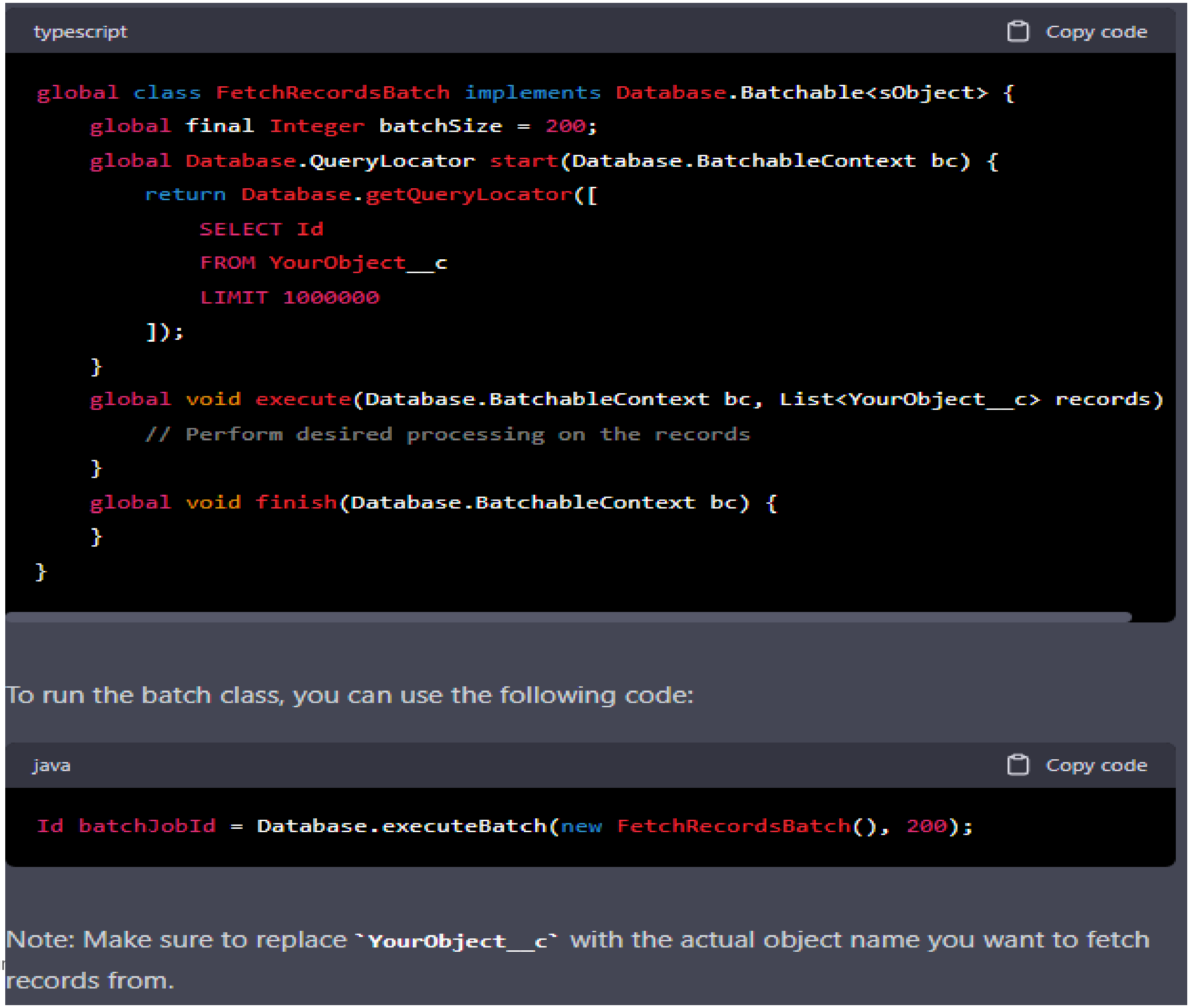
4. Write an apex class to fetch 1 million records using batch class

Copilot



```
// Write a apex class to fetch 1 million records using batch class.
// Write a apex class to fetch 1 million records using batch class.
// Write a apex class to fetch 1 million records using batch class.
// Write a apex class to fetch 1 million records using batch class.
```

ChatGPT



```
global class FetchRecordsBatch implements Database.Batchable<sObject> {
    global final Integer batchSize = 200;
    global Database.QueryLocator start(Database.BatchableContext bc) {
        return Database.getQueryLocator([
            SELECT Id
            FROM YourObject__c
            LIMIT 1000000
        ]);
    }
    global void execute(Database.BatchableContext bc, List<YourObject__c> records)
        // Perform desired processing on the records
    }
    global void finish(Database.BatchableContext bc) {
    }
}
```

To run the batch class, you can use the following code:

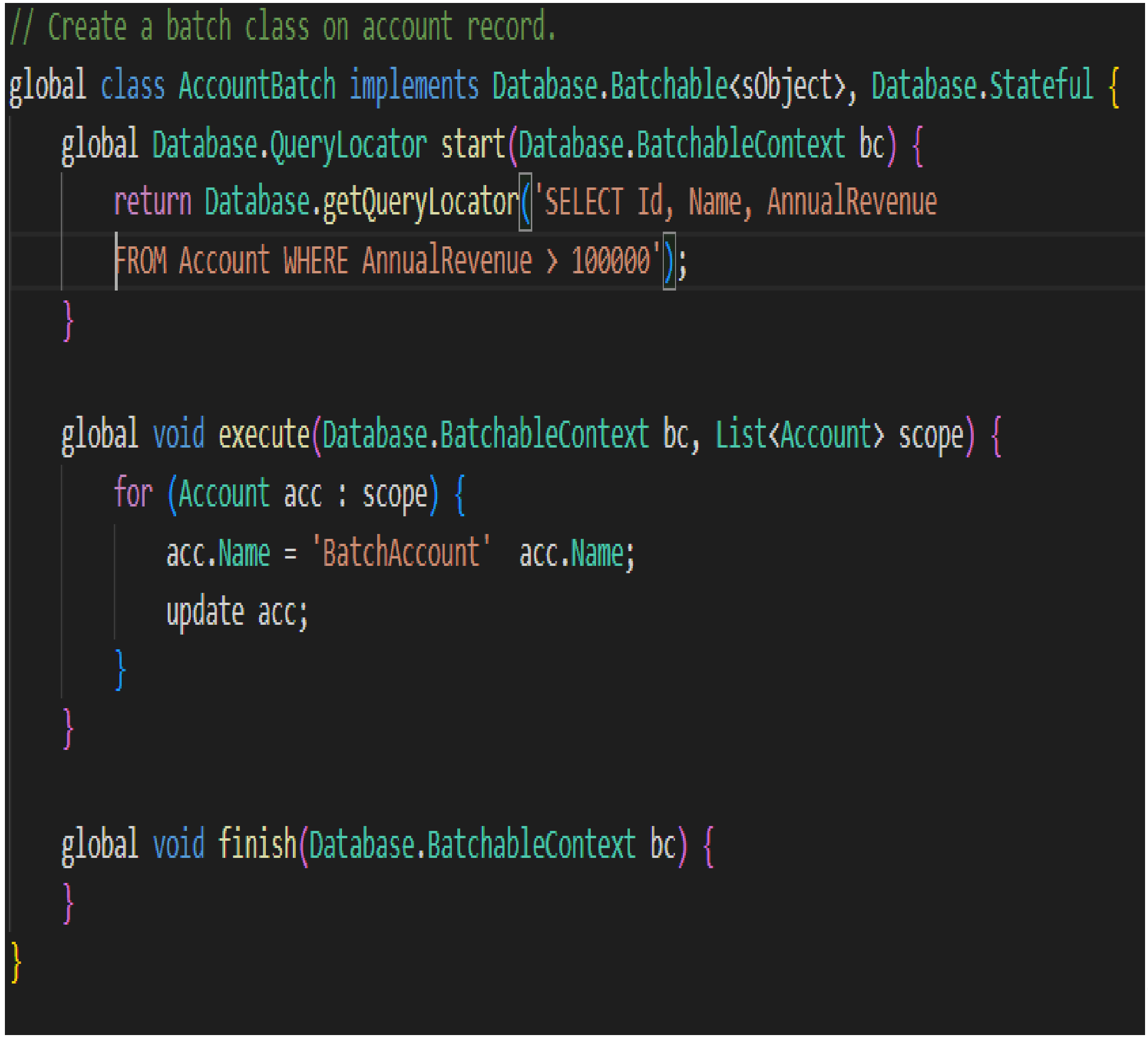
```
Id batchJobId = Database.executeBatch(new FetchRecordsBatch(), 200);
```

Note: Make sure to replace `YourObject__c` with the actual object name you want to fetch records from.

© 2022 Optu

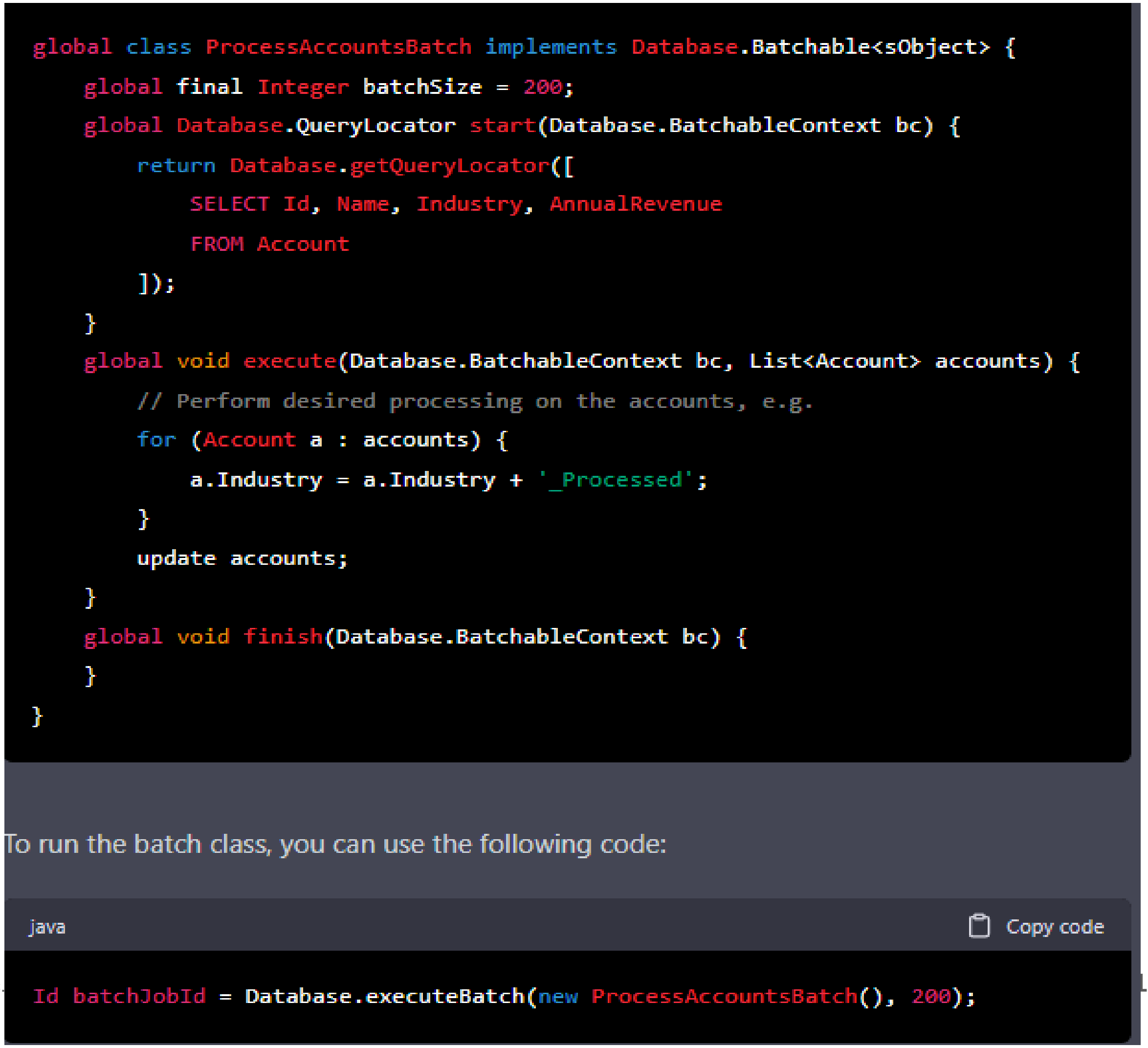
5. Write a batch apex on account record

Copilot



```
// Create a batch class on account record.
global class AccountBatch implements Database.Batchable<sObject>, Database.Stateful {
    global Database.QueryLocator start(Database.BatchableContext bc) {
        return Database.getQueryLocator('SELECT Id, Name, AnnualRevenue
        FROM Account WHERE AnnualRevenue > 100000');
    }

    global void execute(Database.BatchableContext bc, List<Account> scope) {
        for (Account acc : scope) {
            acc.Name = 'BatchAccount'  acc.Name;
            update acc;
        }
    }

    global void finish(Database.BatchableContext bc) {
    }
}
```

ChatGPT



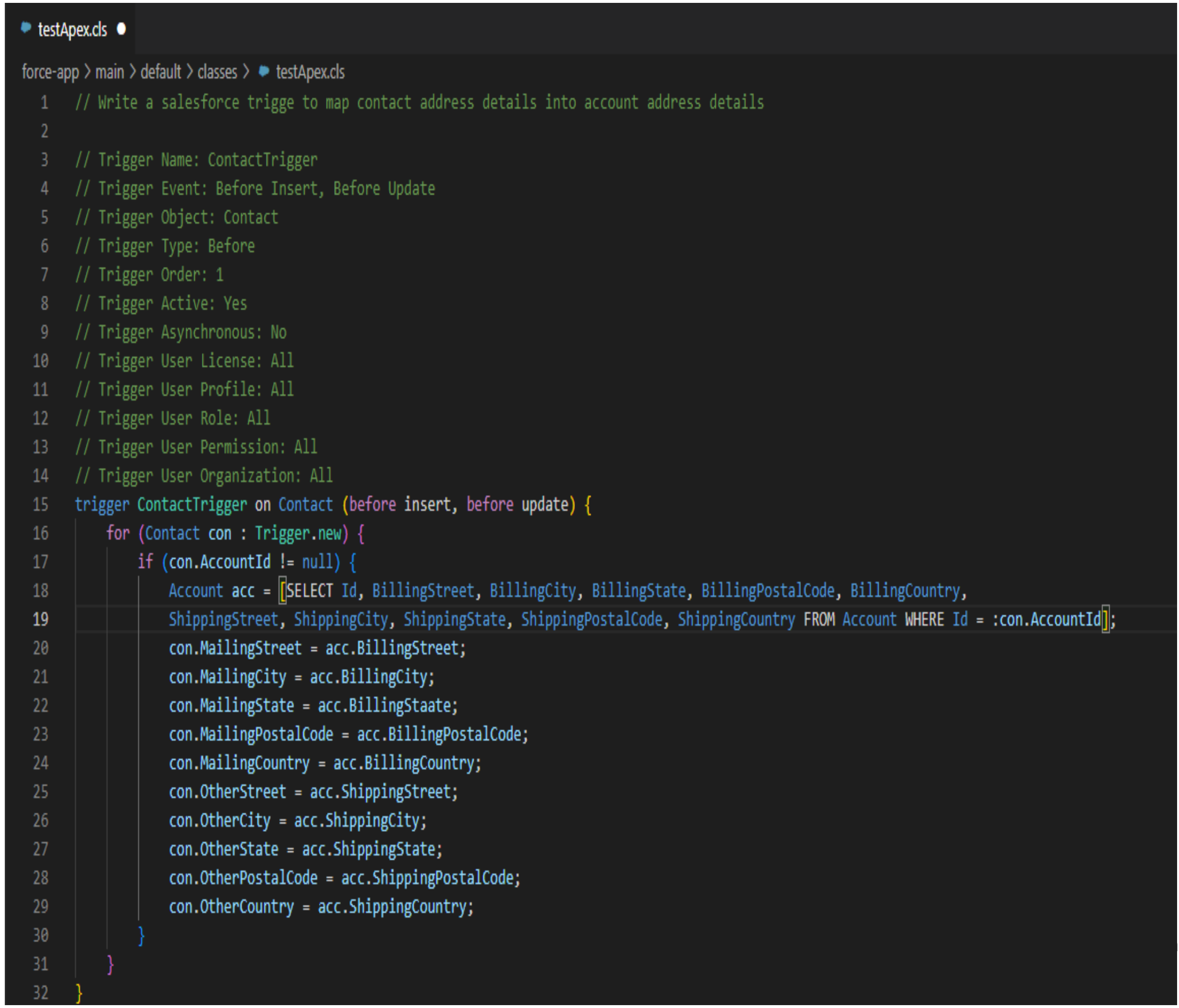
```
global class ProcessAccountsBatch implements Database.Batchable<sObject> {
    global final Integer batchSize = 200;
    global Database.QueryLocator start(Database.BatchableContext bc) {
        return Database.getQueryLocator([
            SELECT Id, Name, Industry, AnnualRevenue
            FROM Account
        ]);
    }
    global void execute(Database.BatchableContext bc, List<Account> accounts) {
        // Perform desired processing on the accounts, e.g.
        for (Account a : accounts) {
            a.Industry = a.Industry + '_Processed';
        }
        update accounts;
    }
    global void finish(Database.BatchableContext bc) {
    }
}
```

To run the batch class, you can use the following code:

```
Id batchJobId = Database.executeBatch(new ProcessAccountsBatch(), 200);
```

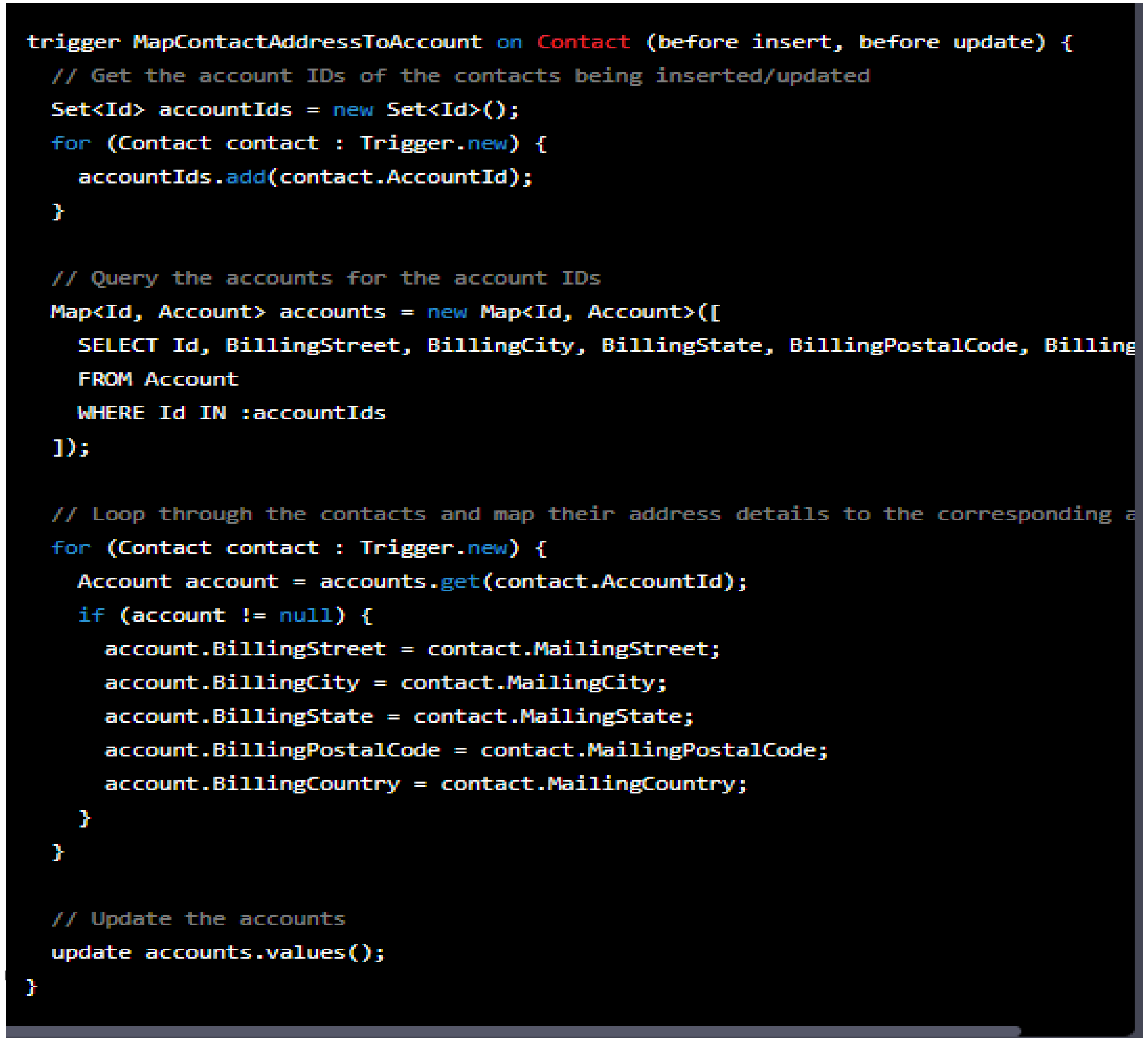
6. Write a salesforce trigger to map contact address details into account object:

Copilot



```
// Write a salesforce trigge to map contact address details into account address details

// Trigger Name: ContactTrigger
// Trigger Event: Before Insert, Before Update
// Trigger Object: Contact
// Trigger Type: Before
// Trigger Order: 1
// Trigger Active: Yes
// Trigger Asynchronous: No
// Trigger User License: All
// Trigger User Profile: All
// Trigger User Role: All
// Trigger User Permission: All
// Trigger User Organization: All
trigger ContactTrigger on Contact (before insert, before update) {
    for (Contact con : Trigger.new) {
        if (con.AccountId != null) {
            Account acc = [SELECT Id, BillingStreet, BillingCity, BillingState, BillingPostalCode, BillingCountry,
            ShippingStreet, ShippingCity, ShippingState, ShippingPostalCode, ShippingCountry FROM Account WHERE Id = :con.AccountId];
            con.MailingStreet = acc.BillingStreet;
            con.MailingCity = acc.BillingCity;
            con.MailingState = acc.BillingStaate;
            con.MailingPostalCode = acc.BillingPostalCode;
            con.MailingCountry = acc.BillingCountry;
            con.OtherStreet = acc.ShippingStreet;
            con.OtherCity = acc.ShippingCity;
            con.OtherState = acc.ShippingState;
            con.OtherPostalCode = acc.ShippingPostalCode;
            con.OtherCountry = acc.ShippingCountry;
        }
    }
}
```

ChatGPT

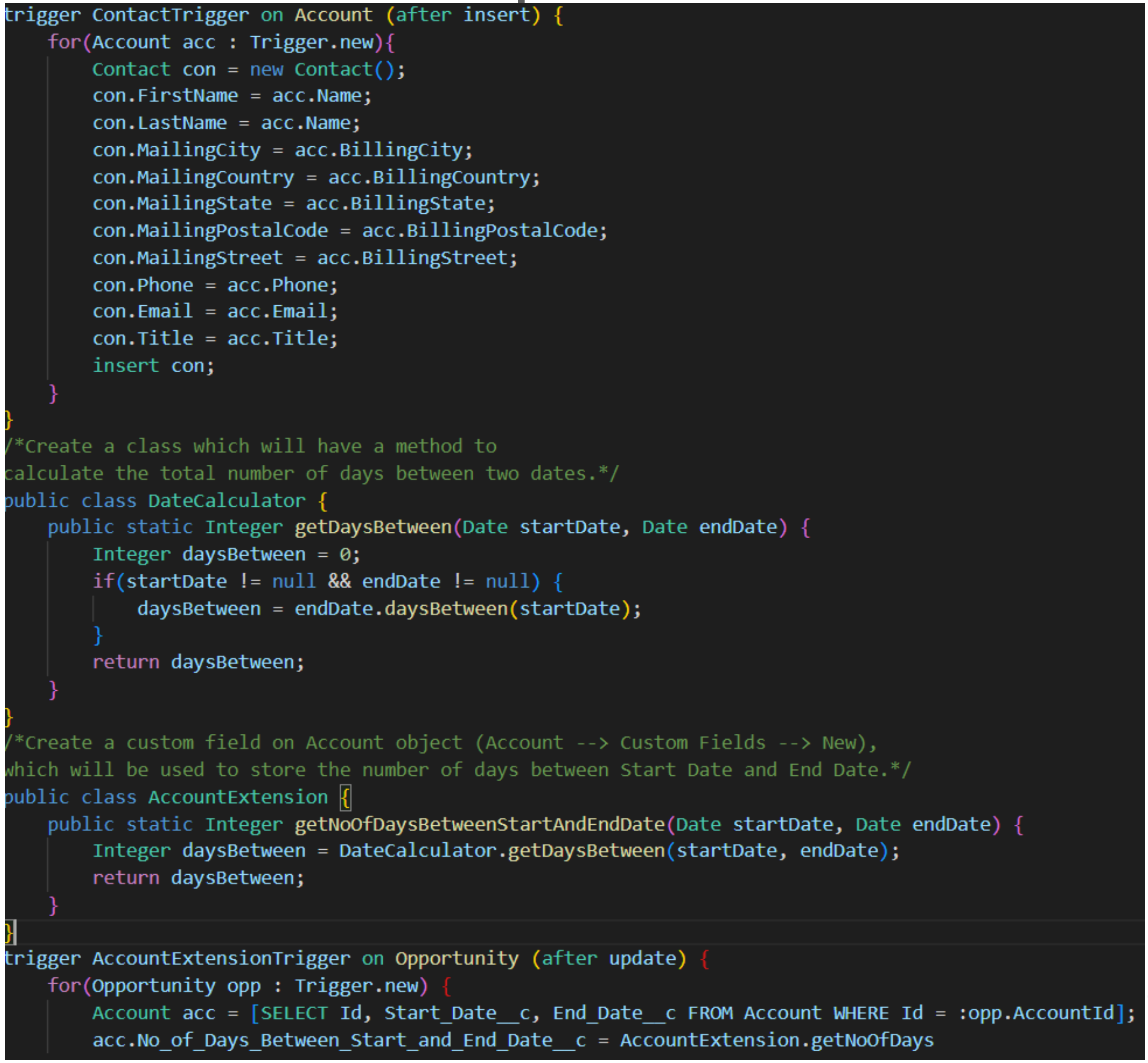


```
trigger MapContactAddressToAccount on Contact (before insert, before update) {
  // Get the account IDs of the contacts being inserted/updated
  Set<Id> accountIds = new Set<Id>();
  for (Contact contact : Trigger.new) {
    accountIds.add(contact.AccountId);
  }

  // Query the accounts for the account IDs
  Map<Id, Account> accounts = new Map<Id, Account>([
    SELECT Id, BillingStreet, BillingCity, BillingState, BillingPostalCode, Billing
    FROM Account
    WHERE Id IN :accountIds
  ]);

  // Loop through the contacts and map their address details to the corresponding a
  for (Contact contact : Trigger.new) {
    Account account = accounts.get(contact.AccountId);
    if (account != null) {
      account.BillingStreet = contact.MailingStreet;
      account.BillingCity = contact.MailingCity;
      account.BillingState = contact.MailingState;
      account.BillingPostalCode = contact.MailingPostalCode;
      account.BillingCountry = contact.MailingCountry;
    }
  }

  // Update the accounts
  update accounts.values();
}
```

c. All

7. Create apex trigger which will create whenever a new account is created. Contact name should match Account Name, contact mailing city should be equal to account mailing city, contact mailing country should be equal to account billing country

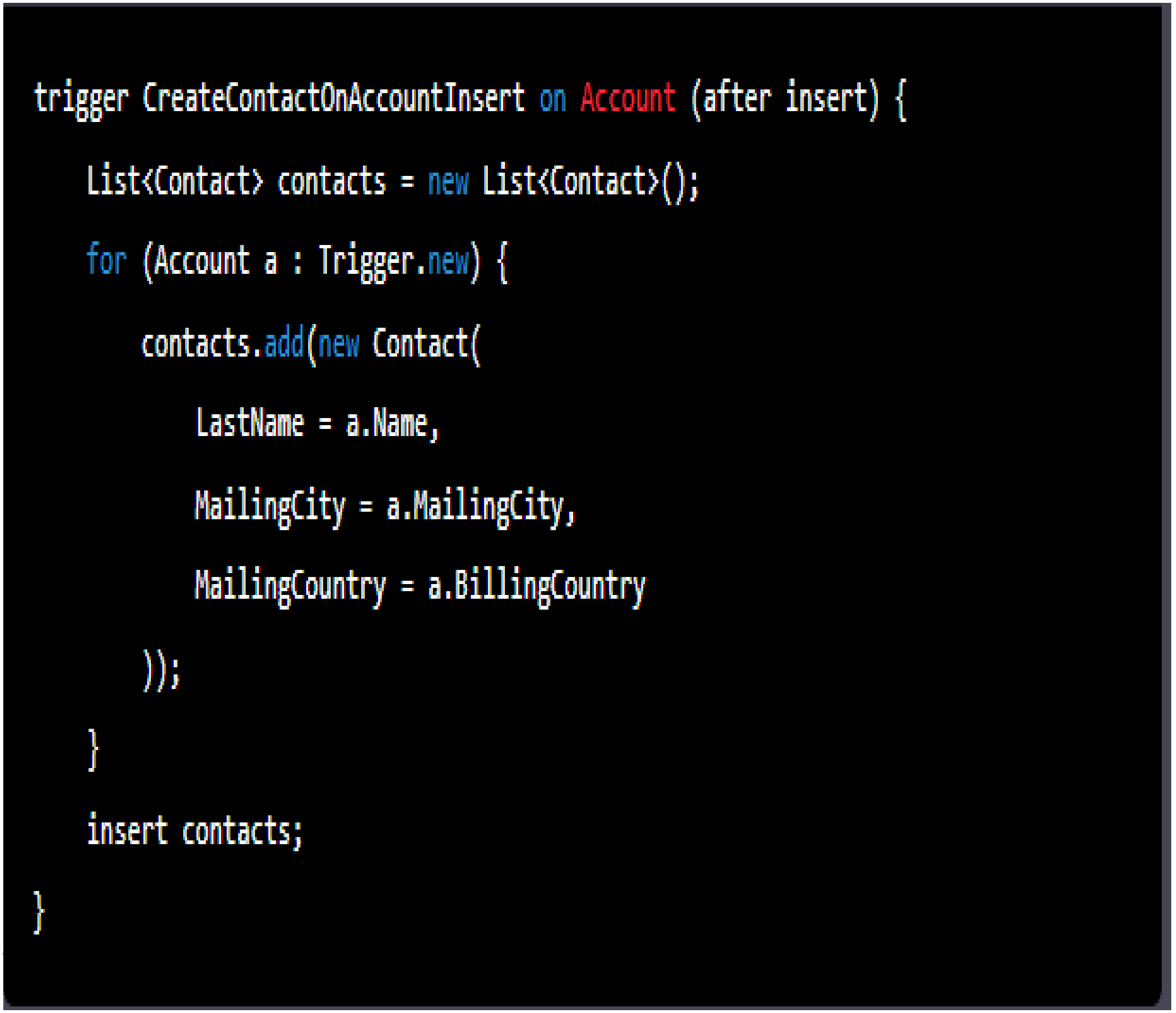
Copilot



```
trigger ContactTrigger on Account (after insert) {
    for(Account acc : Trigger.new){
        Contact con = new Contact();
        con.FirstName = acc.Name;
        con.LastName = acc.Name;
        con.MailingCity = acc.BillingCity;
        con.MailingCountry = acc.BillingCountry;
        con.MailingState = acc.BillingState;
        con.MailingPostalCode = acc.BillingPostalCode;
        con.MailingStreet = acc.BillingStreet;
        con.Phone = acc.Phone;
        con.Email = acc.Email;
        con.Title = acc.Title;
        insert con;
    }
}
/*Create a class which will have a method to
calculate the total number of days between two dates.*/
public class DateCalculator {
    public static Integer getDaysBetween(Date startDate, Date endDate) {
        Integer daysBetween = 0;
        if(startDate != null && endDate != null) {
            daysBetween = endDate.daysBetween(startDate);
        }
        return daysBetween;
    }
}
/*Create a custom field on Account object (Account --> Custom Fields --> New),
which will be used to store the number of days between Start Date and End Date.*/
public class AccountExtension {
    public static Integer getNoOfDaysBetweenStartAndEndDate(Date startDate, Date endDate) {
        Integer daysBetween = DateCalculator.getDaysBetween(startDate, endDate);
        return daysBetween;
    }
}
trigger AccountExtensionTrigger on Opportunity (after update) {
    for(Opportunity opp : Trigger.new) {
        Account acc = [SELECT Id, Start_Date__c, End_Date__c FROM Account WHERE Id = :opp.AccountId];
        acc.No_of_Days_Between_Start_and_End_Date__c = AccountExtension.getNoOfDays
```

Optum

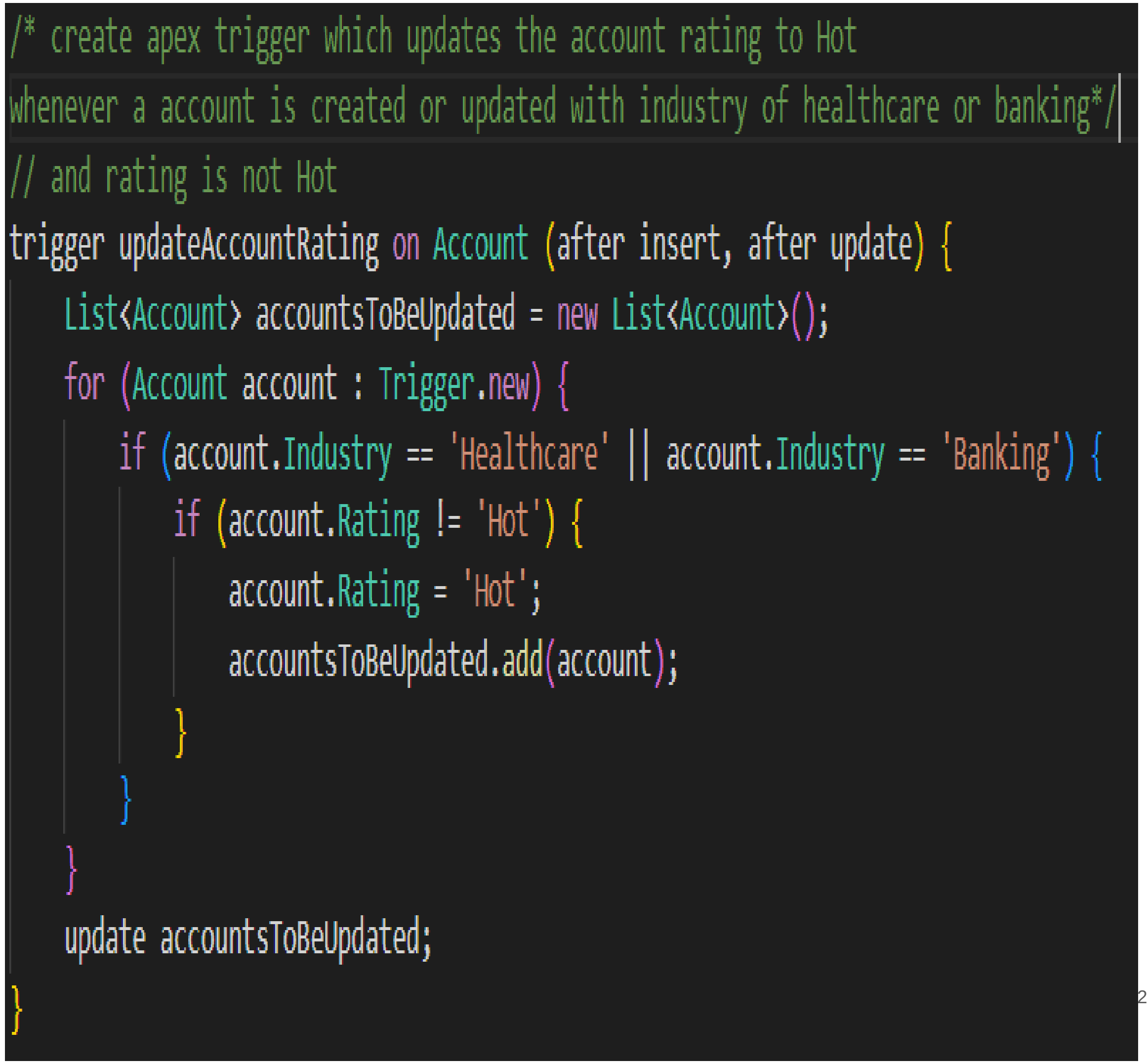
ChatGPT



```
trigger CreateContactOnAccountInsert on Account (after insert) {
    List<Contact> contacts = new List<Contact>();
    for (Account a : Trigger.new) {
        contacts.add(new Contact(
            LastName = a.Name,
            MailingCity = a.MailingCity,
            MailingCountry = a.BillingCountry
        ));
    }
    insert contacts;
}
```

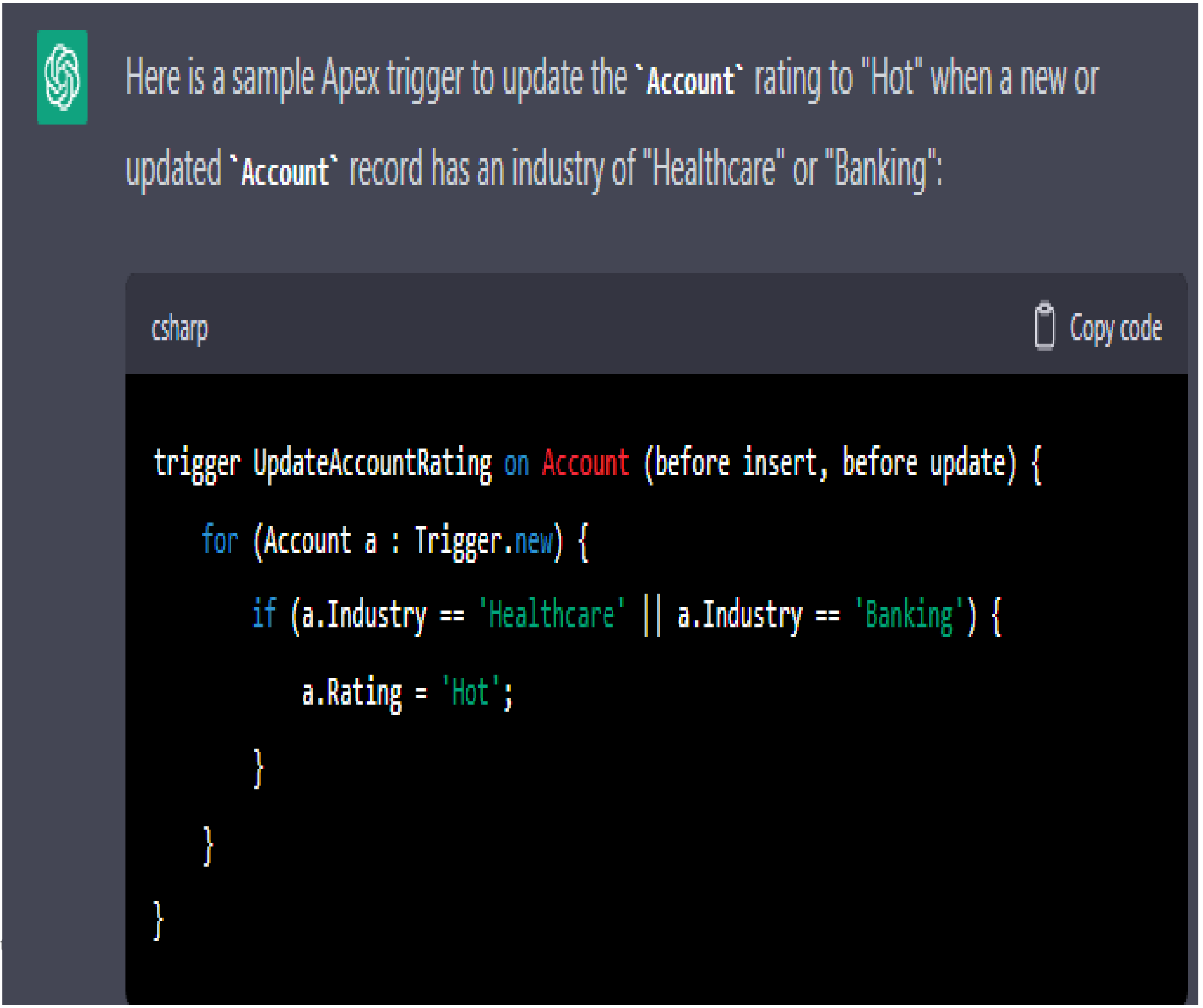
8. create apex trigger which updates the account rating to Hot whenever a account is created or updated with industry of healthcare or banking

Copilot



```
/* create apex trigger which updates the account rating to Hot
whenever a account is created or updated with industry of healthcare or banking*/
// and rating is not Hot
trigger updateAccountRating on Account (after insert, after update) {
    List<Account> accountsToBeUpdated = new List<Account>();
    for (Account account : Trigger.new) {
        if (account.Industry == 'Healthcare' || account.Industry == 'Banking') {
            if (account.Rating != 'Hot') {
                account.Rating = 'Hot';
                accountsToBeUpdated.add(account);
            }
        }
    }
    update accountsToBeUpdated;
}
```

ChatGPT

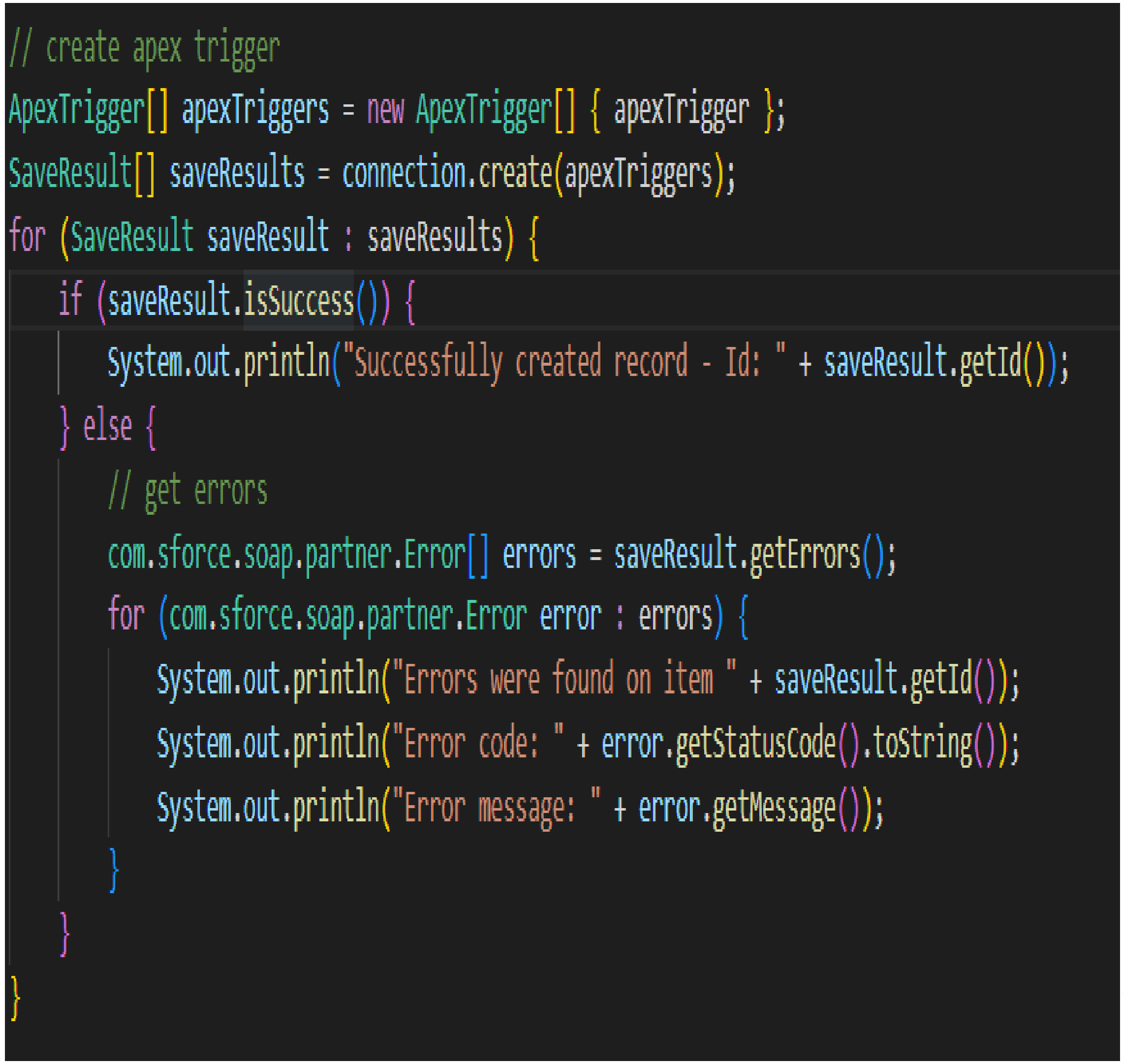


Here is a sample Apex trigger to update the `Account` rating to "Hot" when a new or updated `Account` record has an industry of "Healthcare" or "Banking":

```
trigger UpdateAccountRating on Account (before insert, before update) {
    for (Account a : Trigger.new) {
        if (a.Industry == 'Healthcare' || a.Industry == 'Banking') {
            a.Rating = 'Hot';
        }
    }
}
```

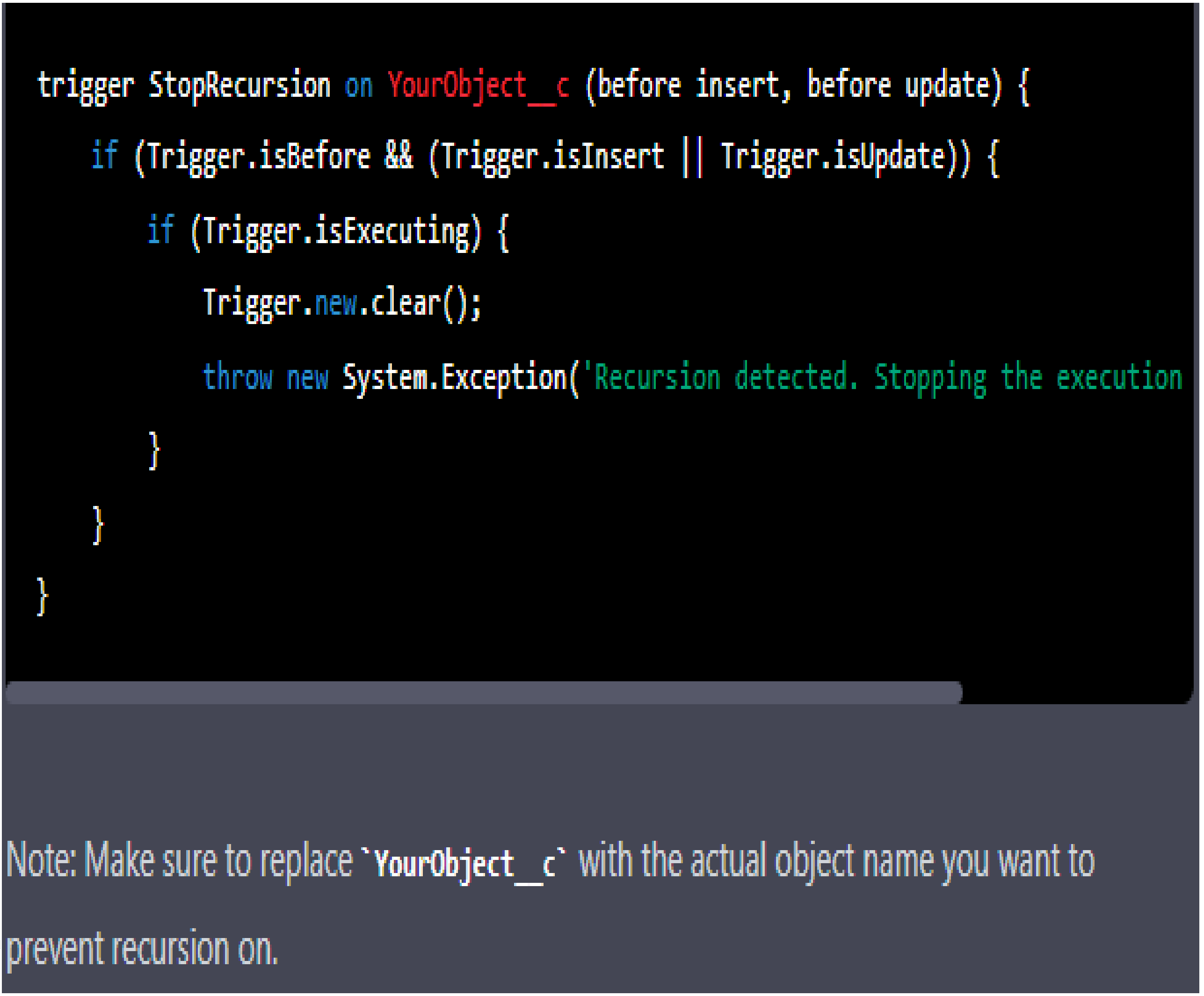
9. Write a trigger to stop recursive:

Copilot



```
// create apex trigger
ApexTrigger[] apexTriggers = new ApexTrigger[] { apexTrigger };
SaveResult[] saveResults = connection.create(apexTriggers);
for (SaveResult saveResult : saveResults) {
    if (saveResult.isSuccess()) {
        System.out.println("Successfully created record - Id: " + saveResult.getId());
    } else {
        // get errors
        com.sforce.soap.partner.Error[] errors = saveResult.getErrors();
        for (com.sforce.soap.partner.Error error : errors) {
            System.out.println("Errors were found on item " + saveResult.getId());
            System.out.println("Error code: " + error.getStatusCode().toString());
            System.out.println("Error message: " + error.getMessage());
        }
    }
}
```

ChatGPT



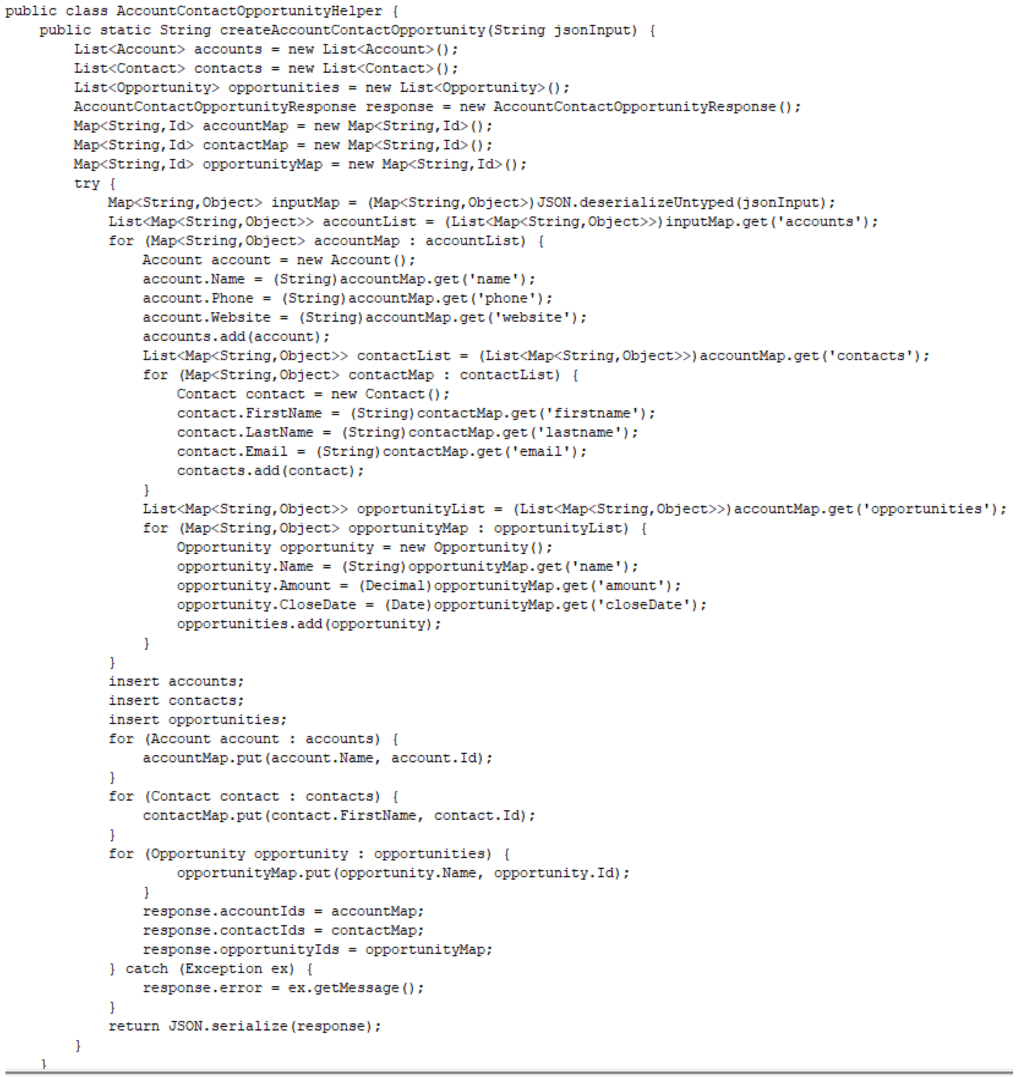
```
trigger StopRecursion on YourObject__c (before insert, before update) {
    if (Trigger.isBefore && (Trigger.isInsert || Trigger.isUpdate)) {
        if (Trigger.isExecuting) {
            Trigger.new.clear();
            throw new System.Exception('Recursion detected. Stopping the execution
        }
    }
}
```

Note: Make sure to replace `YourObject__c` with the actual object name you want to prevent recursion on.

Optu

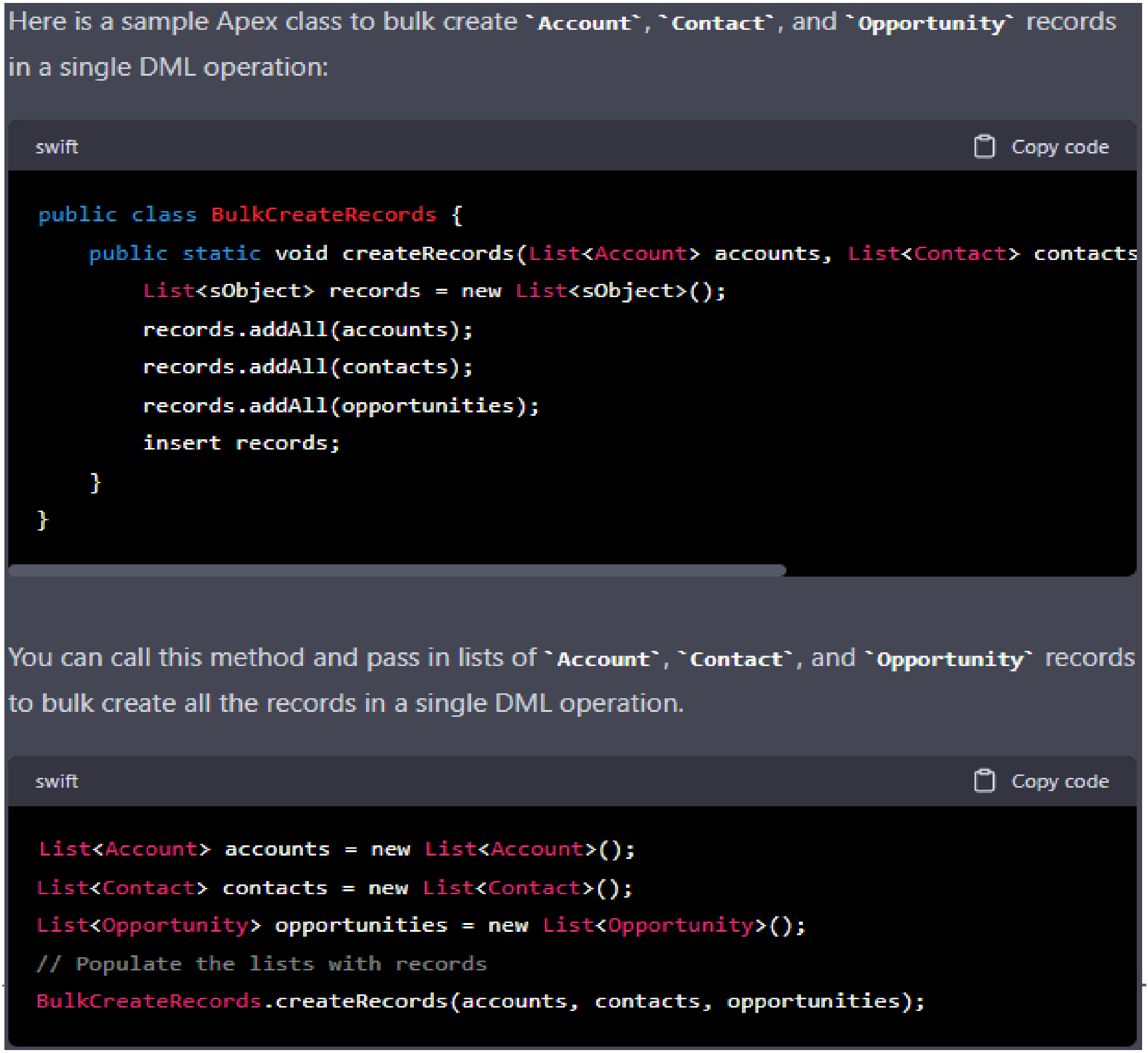
10. Write a salesforce apex class to bulk create account, contact and opportunity in single DML operation

Copilot



```
public class AccountContactOpportunityHelper {
    public static String createAccountContactOpportunity(String jsonInput) {
        List<Account> accounts = new List<Account>();
        List<Contact> contacts = new List<Contact>();
        List<Opportunity> opportunities = new List<Opportunity>();
        AccountContactOpportunityResponse response = new AccountContactOpportunityResponse();
        Map<String,Id> accountMap = new Map<String,Id>();
        Map<String,Id> contactMap = new Map<String,Id>();
        Map<String,Id> opportunityMap = new Map<String,Id>();
        try {
            Map<String,Object> inputMap = (Map<String,Object>)JSON.deserializeUntyped(jsonInput);
            List<Map<String,Object>> accountList = (List<Map<String,Object>>)inputMap.get('accounts');
            for (Map<String,Object> accountMap : accountList) {
                Account account = new Account();
                account.Name = (String)accountMap.get('name');
                account.Phone = (String)accountMap.get('phone');
                account.Website = (String)accountMap.get('website');
                accounts.add(account);
                List<Map<String,Object>> contactList = (List<Map<String,Object>>)accountMap.get('contacts');
                for (Map<String,Object> contactMap : contactList) {
                    Contact contact = new Contact();
                    contact.FirstName = (String)contactMap.get('firstname');
                    contact.LastName = (String)contactMap.get('lastname');
                    contact.Email = (String)contactMap.get('email');
                    contacts.add(contact);
                }
                List<Map<String,Object>> opportunityList = (List<Map<String,Object>>)accountMap.get('opportunities');
                for (Map<String,Object> opportunityMap : opportunityList) {
                    Opportunity opportunity = new Opportunity();
                    opportunity.Name = (String)opportunityMap.get('name');
                    opportunity.Amount = (Decimal)opportunityMap.get('amount');
                    opportunity.CloseDate = (Date)opportunityMap.get('closeDate');
                    opportunities.add(opportunity);
                }
            }
            insert accounts;
            insert contacts;
            insert opportunities;
            for (Account account : accounts) {
                accountMap.put(account.Name, account.Id);
            }
            for (Contact contact : contacts) {
                contactMap.put(contact.FirstName, contact.Id);
            }
            for (Opportunity opportunity : opportunities) {
                    opportunityMap.put(opportunity.Name, opportunity.Id);
                }
                response.accountIds = accountMap;
                response.contactIds = contactMap;
                response.opportunityIds = opportunityMap;
            } catch (Exception ex) {
                response.error = ex.getMessage();
            }
            return JSON.serialize(response);
        }
    }
```

ChatGPT



Here is a sample Apex class to bulk create `Account`, `Contact`, and `Opportunity` records in a single DML operation:

```swift
public class BulkCreateRecords {
    public static void createRecords(List<Account> accounts, List<Contact> contacts
        List<sObject> records = new List<sObject>();
        records.addAll(accounts);
        records.addAll(contacts);
        records.addAll(opportunities);
        insert records;
    }
}
```

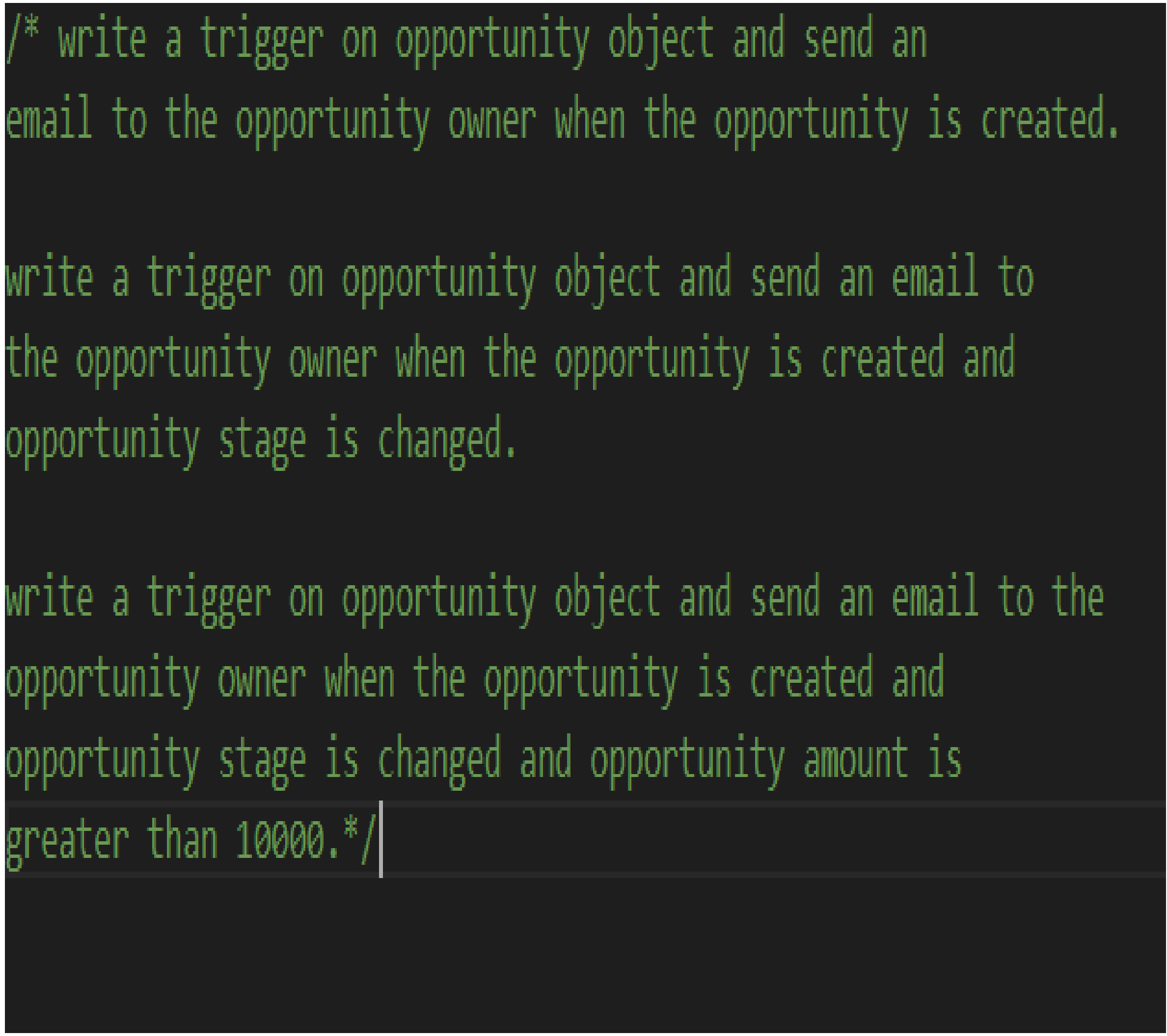
You can call this method and pass in lists of `Account`, `Contact`, and `Opportunity` records to bulk create all the records in a single DML operation.

```swift
List<Account> accounts = new List<Account>();
List<Contact> contacts = new List<Contact>();
List<Opportunity> opportunities = new List<Opportunity>();
// Populate the lists with records
BulkCreateRecords.createRecords(accounts, contacts, opportunities);
```

© 2022 Optum, Inc.

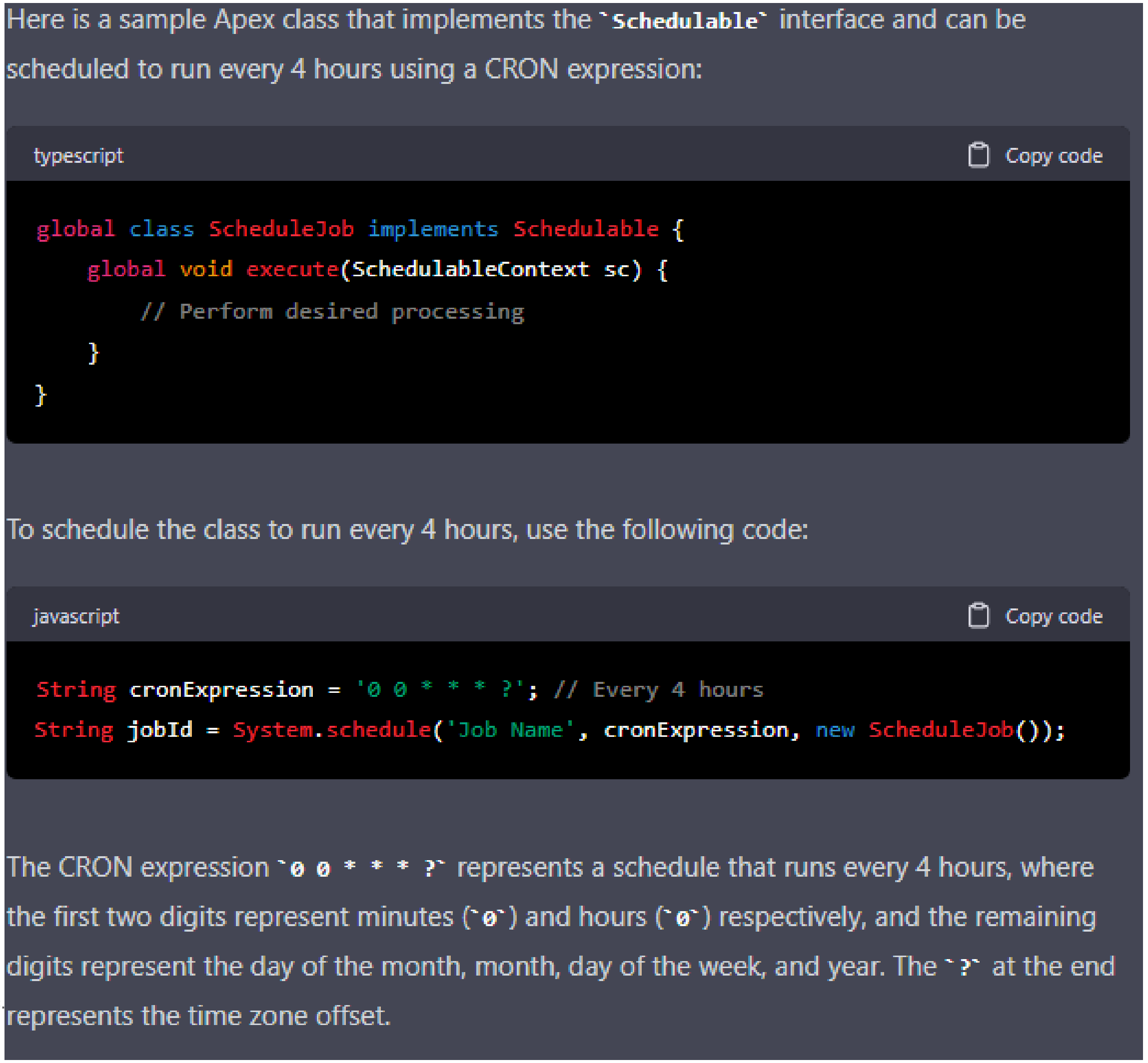
11. Write a apex class and CRON job and schedule it to execute for every 4 hours.

Copilot



```
/* write a trigger on opportunity object and send an
email to the opportunity owner when the opportunity is created.

write a trigger on opportunity object and send an email to
the opportunity owner when the opportunity is created and
opportunity stage is changed.

write a trigger on opportunity object and send an email to the
opportunity owner when the opportunity is created and
opportunity stage is changed and opportunity amount is
greater than 10000.*/
```

ChatGPT



Here is a sample Apex class that implements the `Schedulable` interface and can be scheduled to run every 4 hours using a CRON expression:

```
global class ScheduleJob implements Schedulable {
    global void execute(SchedulableContext sc) {
        // Perform desired processing
    }
}
```

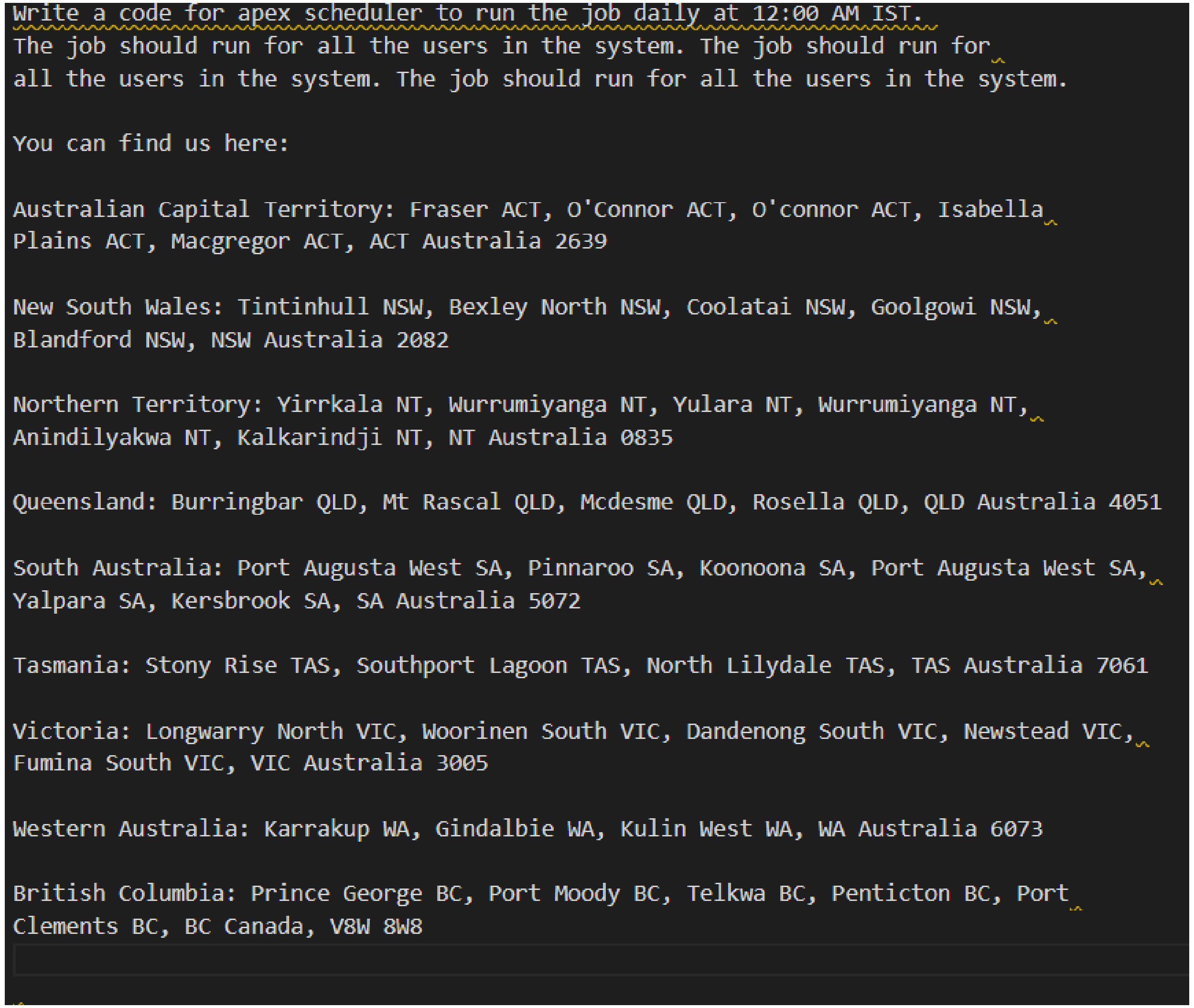
To schedule the class to run every 4 hours, use the following code:

```
String cronExpression = '0 0 * * * ?'; // Every 4 hours
String jobId = System.schedule('Job Name', cronExpression, new ScheduleJob());
```

The CRON expression `0 0 * * * ?` represents a schedule that runs every 4 hours, where the first two digits represent minutes (`0`) and hours (`0`) respectively, and the remaining digits represent the day of the month, month, day of the week, and year. The `?` at the end represents the time zone offset.

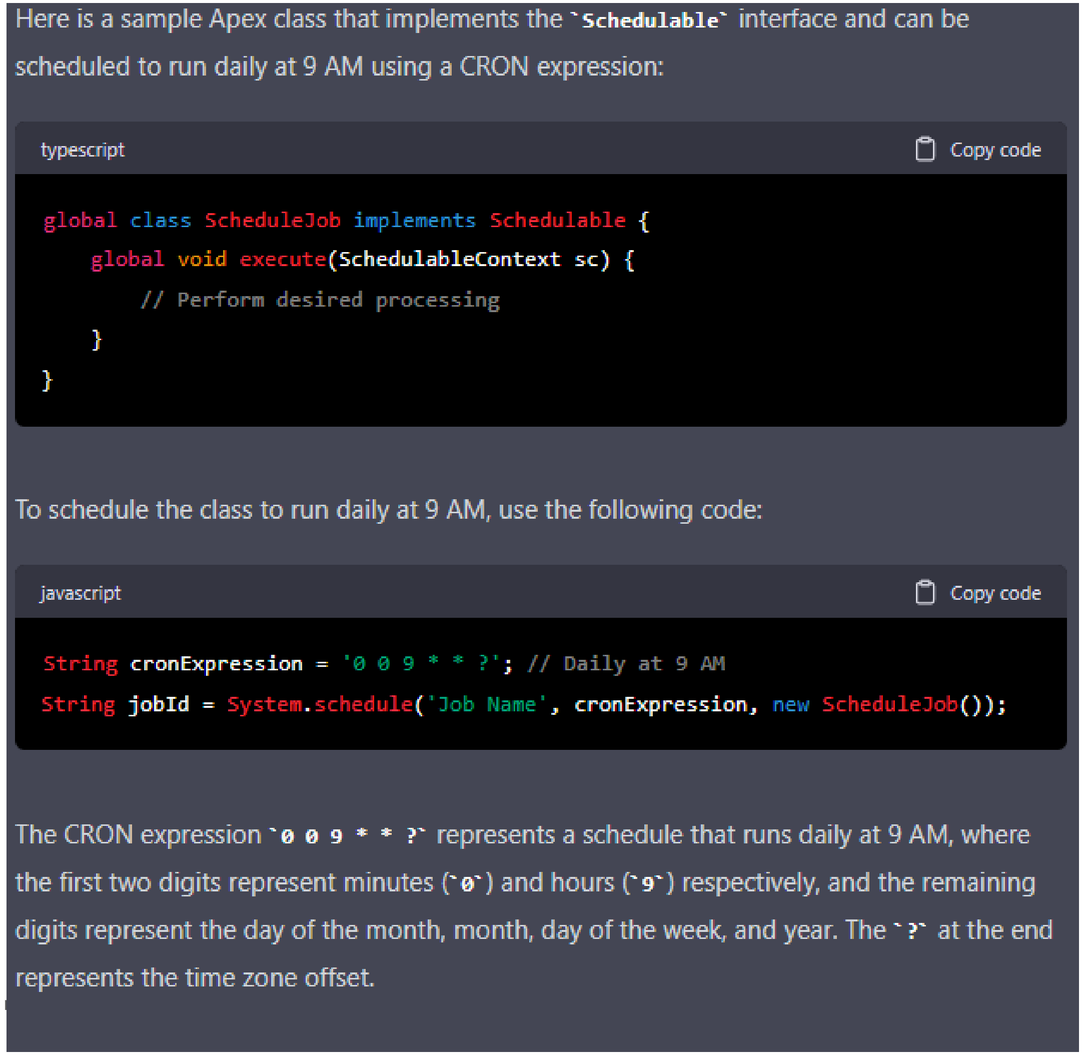
12. Write a code for scheduler class which schedule daily at 9 AM

Copilot



```
Write a code for apex scheduler to run the job daily at 12:00 AM IST.
The job should run for all the users in the system. The job should run for
all the users in the system. The job should run for all the users in the system.

You can find us here:

Australian Capital Territory: Fraser ACT, O'Connor ACT, O'connor ACT, Isabella
Plains ACT, Macgregor ACT, ACT Australia 2639

New South Wales: Tintinhull NSW, Bexley North NSW, Coolatai NSW, Goolgowi NSW,
Blandford NSW, NSW Australia 2082

Northern Territory: Yirrkala NT, Wurrumiyanga NT, Yulara NT, Wurrumiyanga NT,
Anindilyakwa NT, Kalkarindji NT, NT Australia 0835

Queensland: Burringbar QLD, Mt Rascal QLD, Mcdesme QLD, Rosella QLD, QLD Australia 4051

South Australia: Port Augusta West SA, Pinnaroo SA, Koonoona SA, Port Augusta West SA,
Yalpara SA, Kersbrook SA, SA Australia 5072

Tasmania: Stony Rise TAS, Southport Lagoon TAS, North Lilydale TAS, TAS Australia 7061

Victoria: Longwarry North VIC, Woorinen South VIC, Dandenong South VIC, Newstead VIC,
Fumina South VIC, VIC Australia 3005

Western Australia: Karrakup WA, Gindalbie WA, Kulin West WA, WA Australia 6073

British Columbia: Prince George BC, Port Moody BC, Telkwa BC, Penticton BC, Port
Clements BC, BC Canada, V8W 8W8
```

ChatGPT



Here is a sample Apex class that implements the `Schedulable` interface and can be scheduled to run daily at 9 AM using a CRON expression:

```typescript
global class ScheduleJob implements Schedulable {
    global void execute(SchedulableContext sc) {
        // Perform desired processing
    }
}
```

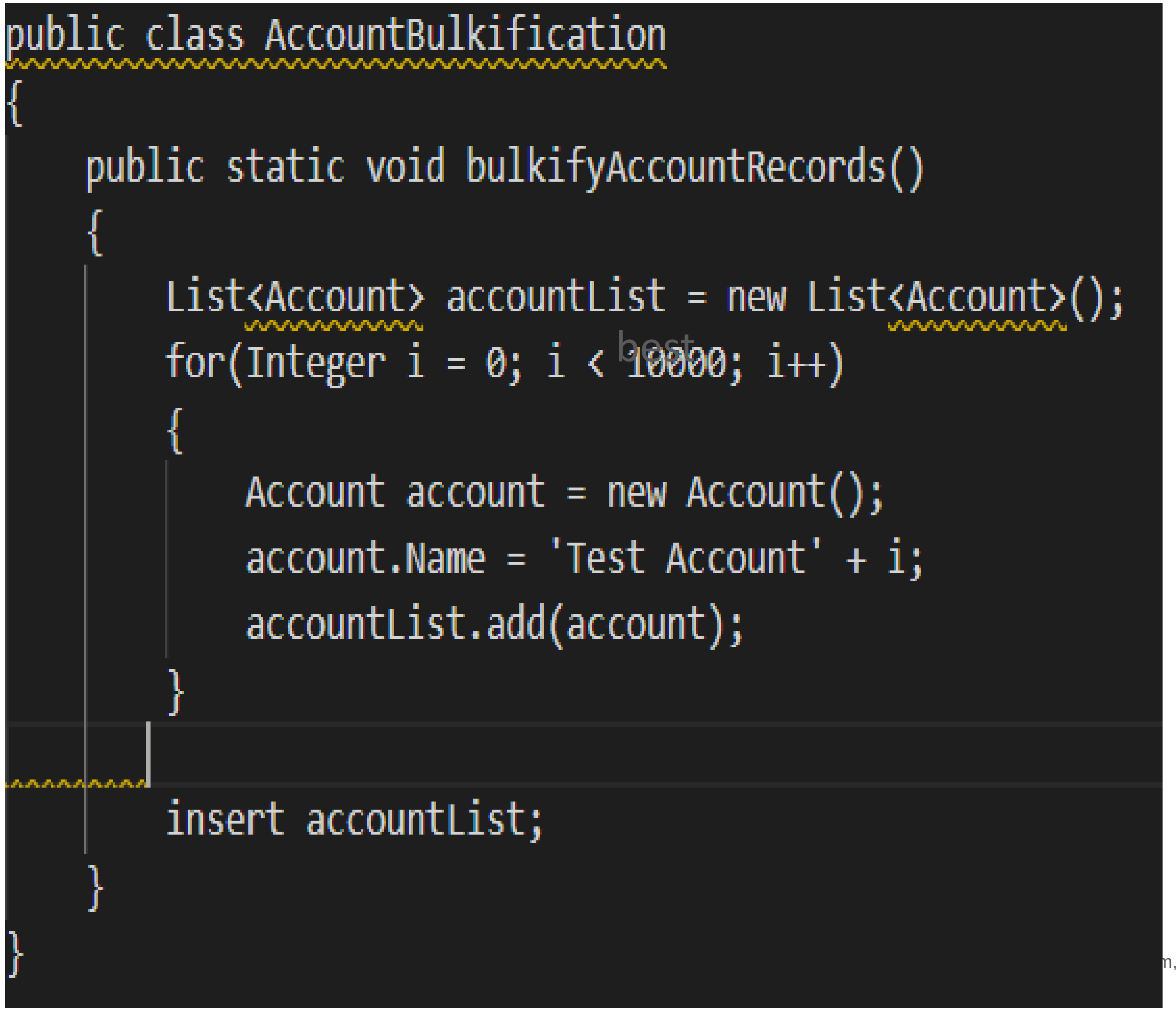
To schedule the class to run daily at 9 AM, use the following code:

```javascript
String cronExpression = '0 0 9 * * ?'; // Daily at 9 AM
String jobId = System.schedule('Job Name', cronExpression, new ScheduleJob());
```

The CRON expression `0 0 9 * * ?` represents a schedule that runs daily at 9 AM, where the first two digits represent minutes (`0`) and hours (`9`) respectively, and the remaining digits represent the day of the month, month, day of the week, and year. The `?` at the end represents the time zone offset.

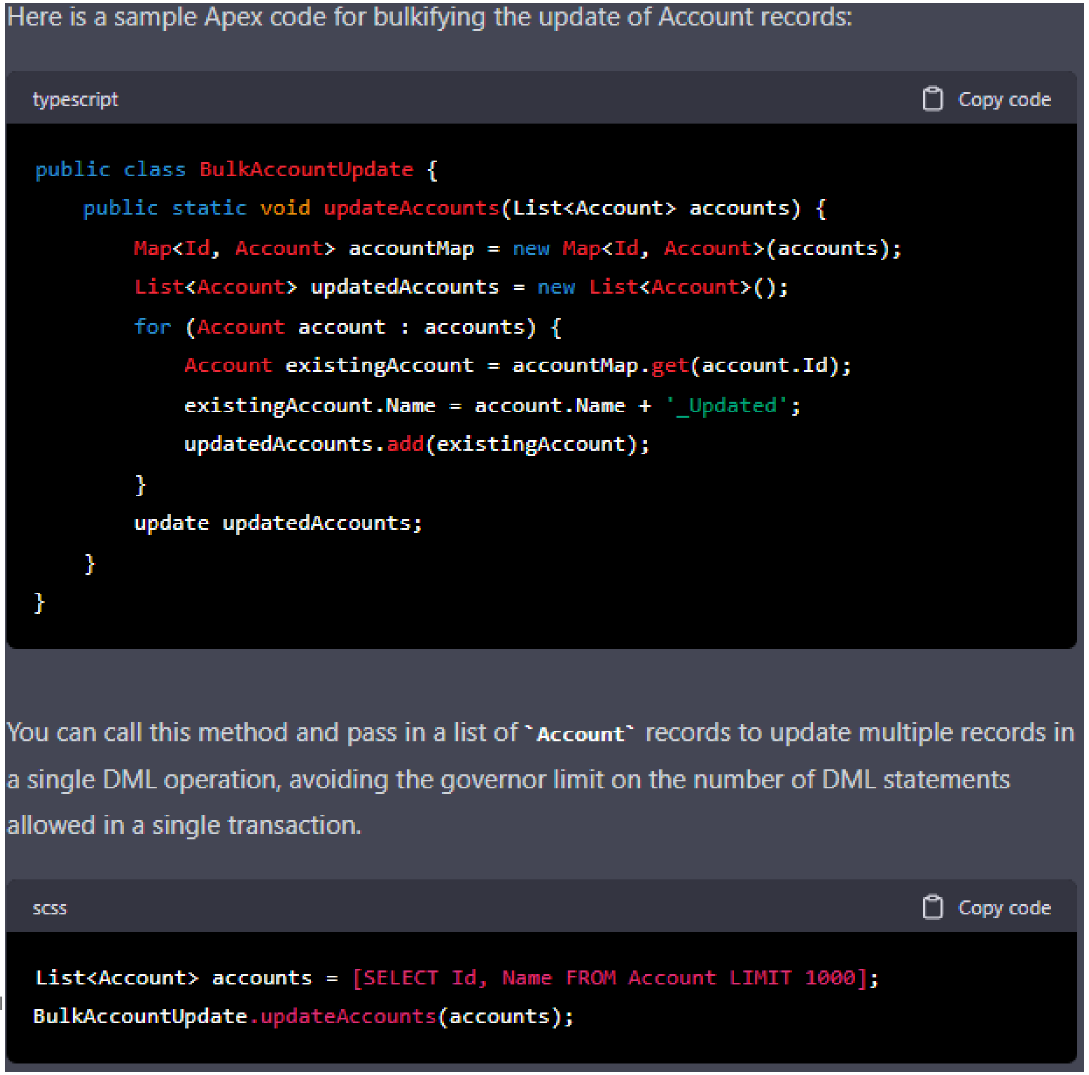
13. Write an apex code for bulkification of the Account records

Copilot



```
public class AccountBulkification
{
    public static void bulkifyAccountRecords()
    {
        List<Account> accountList = new List<Account>();
        for(Integer i = 0; i < 10000; i++)
        {
            Account account = new Account();
            account.Name = 'Test Account' + i;
            accountList.add(account);
        }

        insert accountList;
    }
}
```

ChatGPT



Here is a sample Apex code for bulkifying the update of Account records:

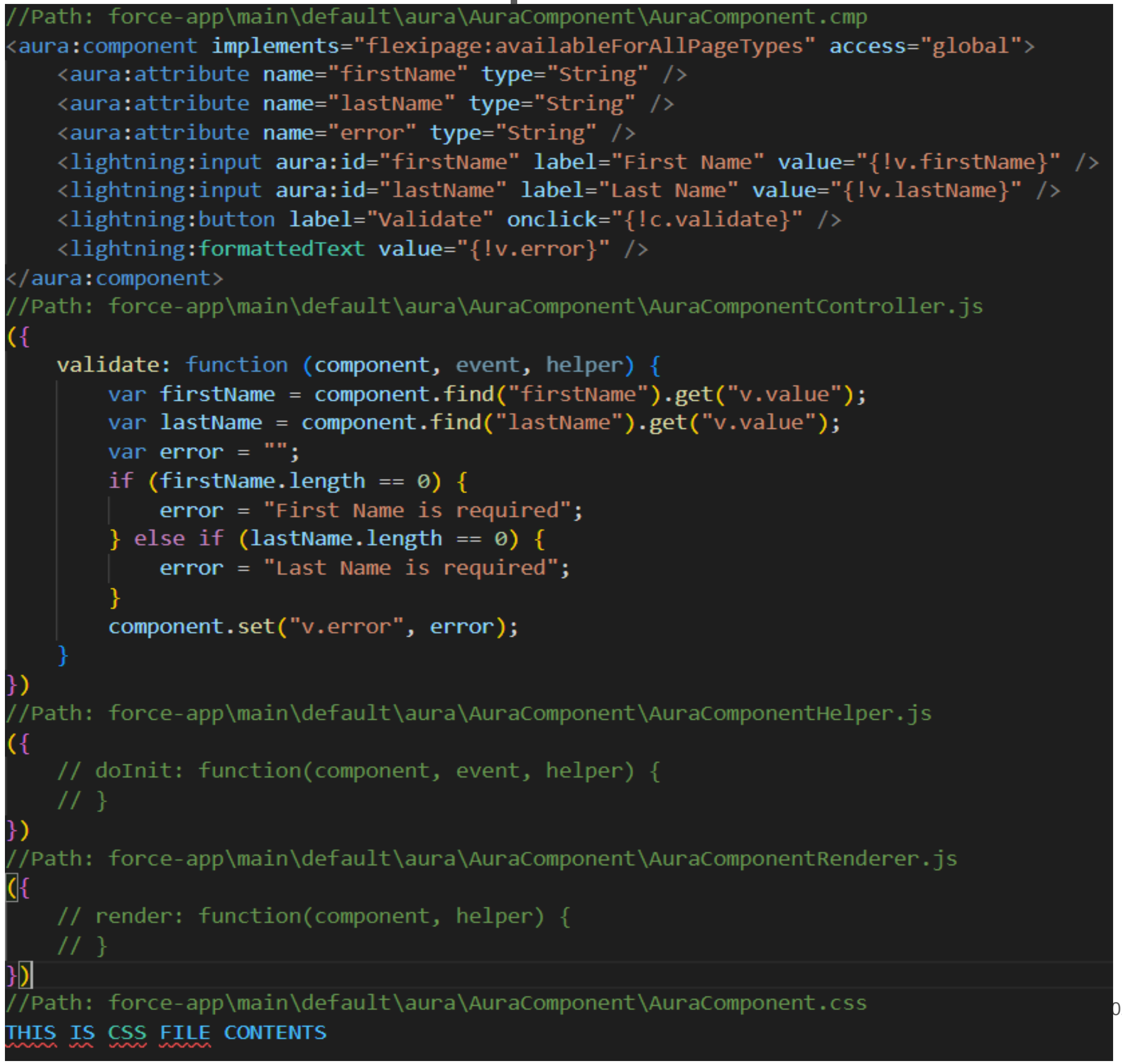
```
public class BulkAccountUpdate {
    public static void updateAccounts(List<Account> accounts) {
        Map<Id, Account> accountMap = new Map<Id, Account>(accounts);
        List<Account> updatedAccounts = new List<Account>();
        for (Account account : accounts) {
            Account existingAccount = accountMap.get(account.Id);
            existingAccount.Name = account.Name + '_Updated';
            updatedAccounts.add(existingAccount);
        }
        update updatedAccounts;
    }
}
```

You can call this method and pass in a list of `Account` records to update multiple records in a single DML operation, avoiding the governor limit on the number of DML statements allowed in a single transaction.

```
List<Account> accounts = [SELECT Id, Name FROM Account LIMIT 1000];
BulkAccountUpdate.updateAccounts(accounts);
```

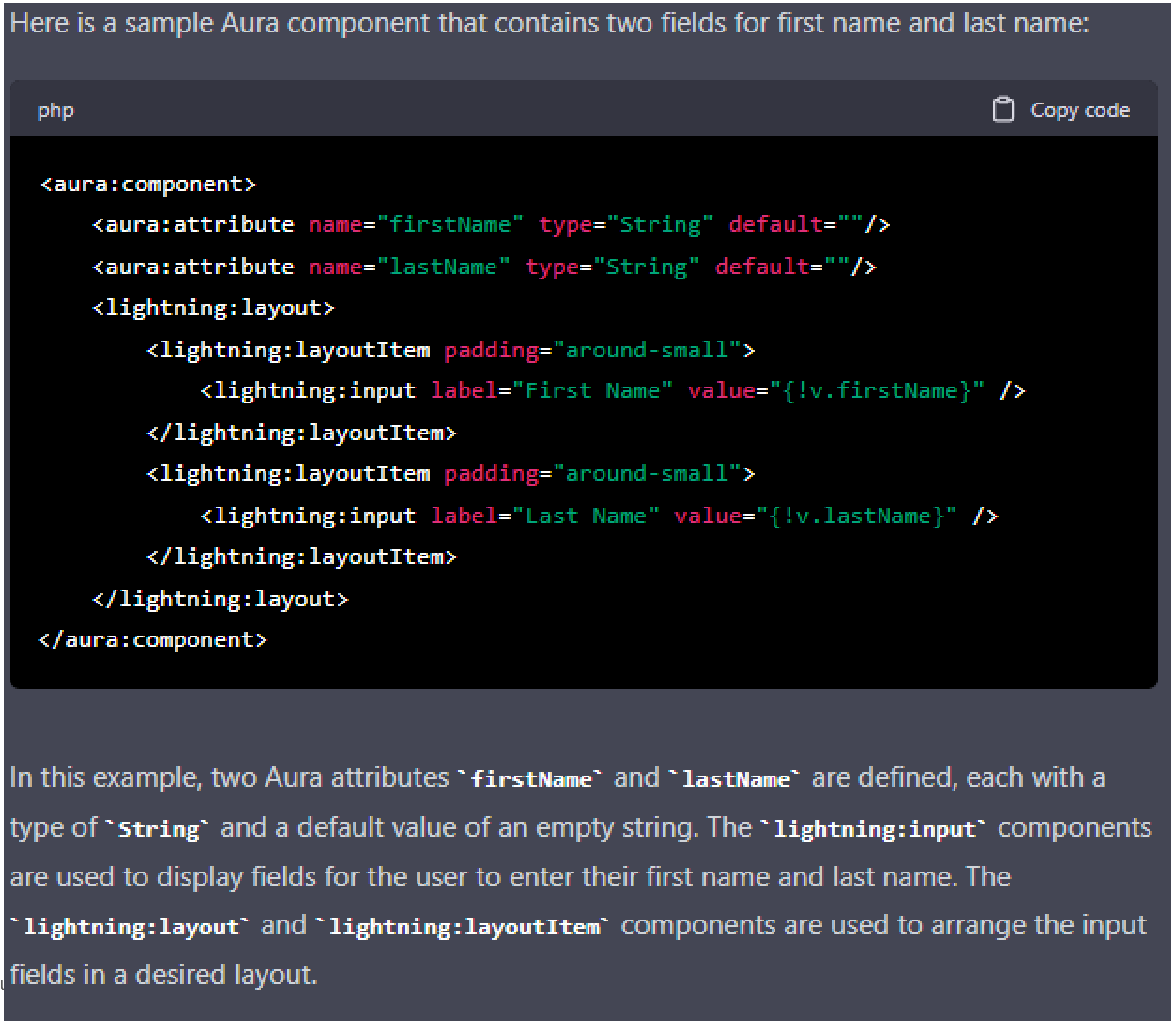
14. Create an Aura component with 2 fields firstname and lastname:

Copilot



```
//Path: force-app\main\default\aura\AuraComponent\AuraComponent.cmp
<aura:component implements="flexipage:availableForAllPageTypes" access="global">
    <aura:attribute name="firstName" type="String" />
    <aura:attribute name="lastName" type="String" />
    <aura:attribute name="error" type="String" />
    <lightning:input aura:id="firstName" label="First Name" value="{!v.firstName}" />
    <lightning:input aura:id="lastName" label="Last Name" value="{!v.lastName}" />
    <lightning:button label="Validate" onclick="{!c.validate}" />
    <lightning:formattedText value="{!v.error}" />
</aura:component>
//Path: force-app\main\default\aura\AuraComponent\AuraComponentController.js
({
    validate: function (component, event, helper) {
        var firstName = component.find("firstName").get("v.value");
        var lastName = component.find("lastName").get("v.value");
        var error = "";
        if (firstName.length == 0) {
            error = "First Name is required";
        } else if (lastName.length == 0) {
            error = "Last Name is required";
        }
        component.set("v.error", error);
    }
})
//Path: force-app\main\default\aura\AuraComponent\AuraComponentHelper.js
({
    // doInit: function(component, event, helper) {
    // }
})
//Path: force-app\main\default\aura\AuraComponent\AuraComponentRenderer.js
({
    // render: function(component, helper) {
    // }
})
//Path: force-app\main\default\aura\AuraComponent\AuraComponent.css
THIS IS CSS FILE CONTENTS
```

ChatGPT



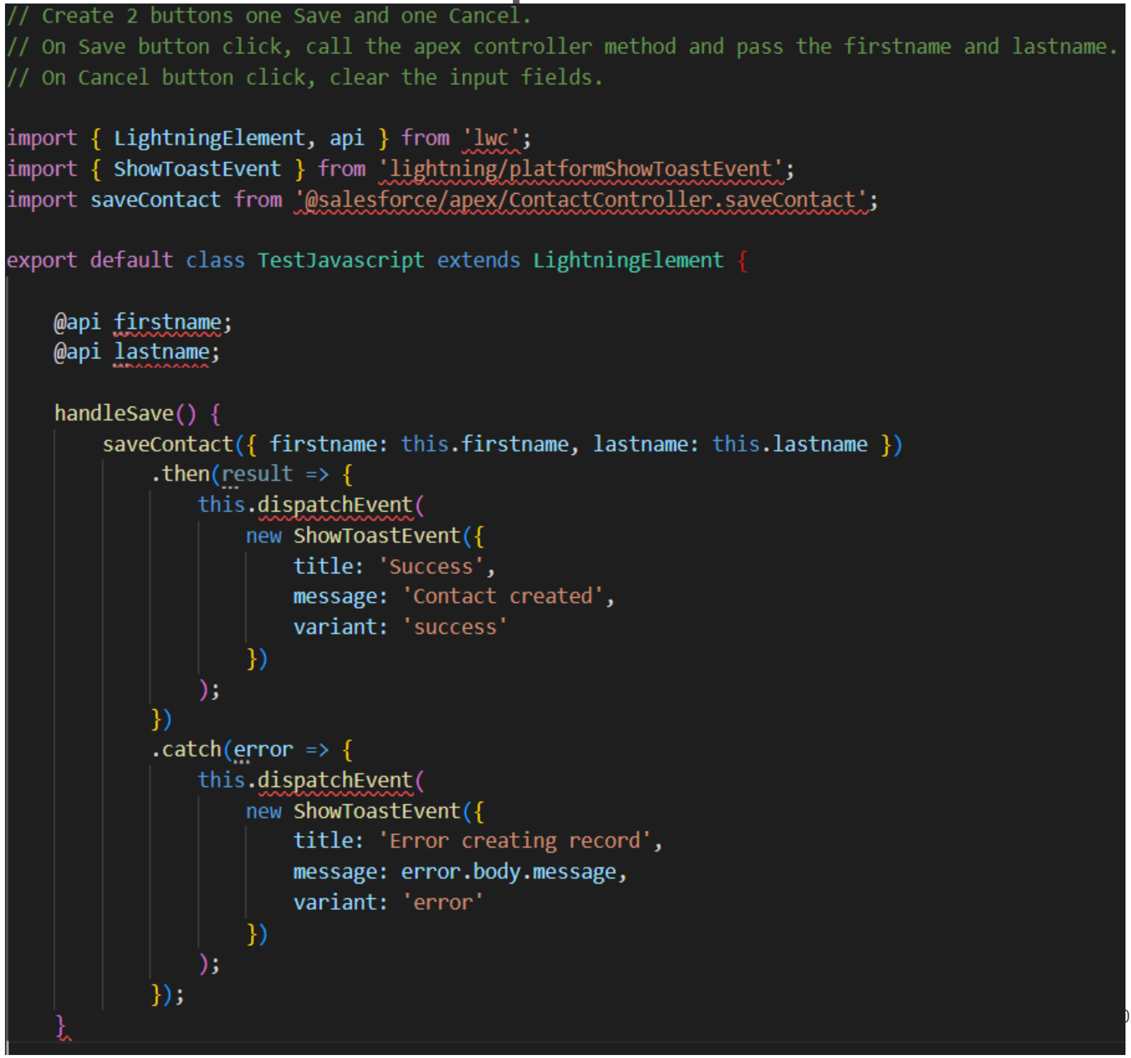
Here is a sample Aura component that contains two fields for first name and last name:

```php
<aura:component>
    <aura:attribute name="firstName" type="String" default=""/>
    <aura:attribute name="lastName" type="String" default=""/>
    <lightning:layout>
        <lightning:layoutItem padding="around-small">
            <lightning:input label="First Name" value="{!v.firstName}" />
        </lightning:layoutItem>
        <lightning:layoutItem padding="around-small">
            <lightning:input label="Last Name" value="{!v.lastName}" />
        </lightning:layoutItem>
    </lightning:layout>
</aura:component>
```

In this example, two Aura attributes `firstName` and `lastName` are defined, each with a type of `String` and a default value of an empty string. The `lightning:input` components are used to display fields for the user to enter their first name and last name. The `lightning:layout` and `lightning:layoutItem` components are used to arrange the input fields in a desired layout.

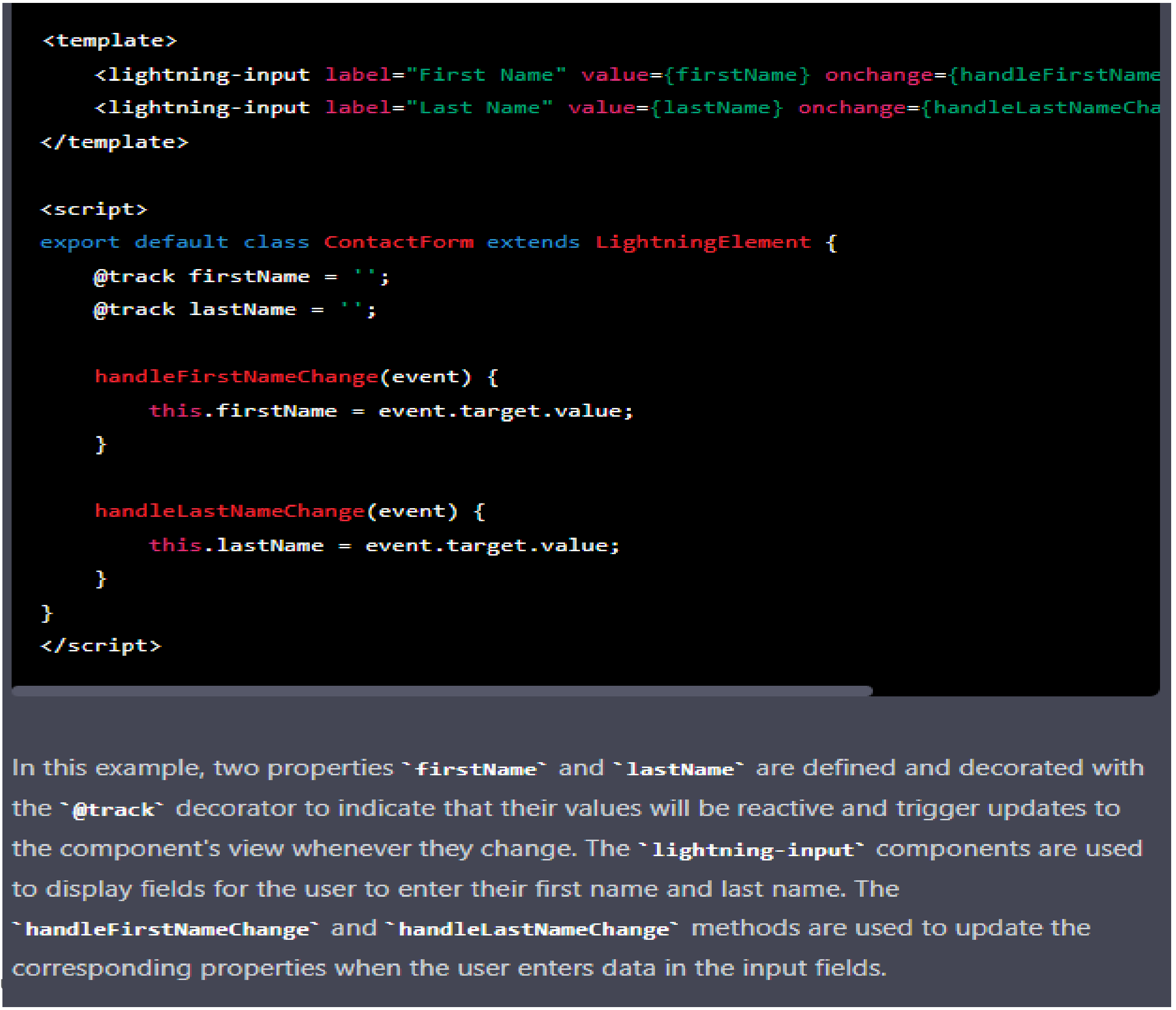
15. Create a LWC component to with firstname and lastname as fields:

Copilot



```
// Create 2 buttons one Save and one Cancel.
// On Save button click, call the apex controller method and pass the firstname and lastname.
// On Cancel button click, clear the input fields.

import { LightningElement, api } from 'lwc';
import { ShowToastEvent } from 'lightning/platformShowToastEvent';
import saveContact from '@salesforce/apex/ContactController.saveContact';

export default class TestJavascript extends LightningElement {

    @api firstname;
    @api lastname;

    handleSave() {
        saveContact({ firstname: this.firstname, lastname: this.lastname })
            .then(result => {
                this.dispatchEvent(
                    new ShowToastEvent({
                        title: 'Success',
                        message: 'Contact created',
                        variant: 'success'
                    })
                );
            })
            .catch(error => {
                this.dispatchEvent(
                    new ShowToastEvent({
                        title: 'Error creating record',
                        message: error.body.message,
                        variant: 'error'
                    })
                );
            });
    }
```

ChatGPT

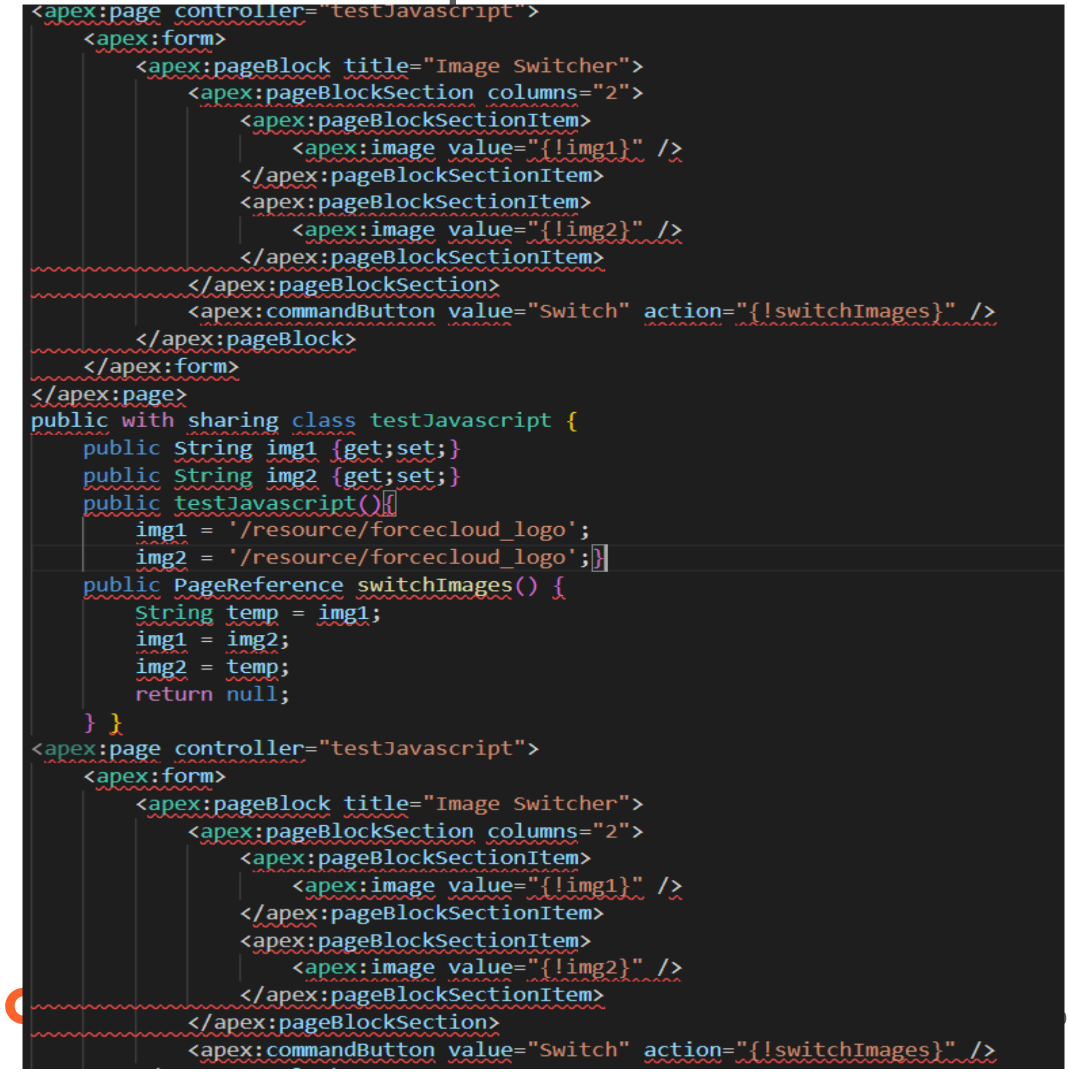


```
<template>
    <lightning-input label="First Name" value={firstName} onchange={handleFirstName
    <lightning-input label="Last Name" value={lastName} onchange={handleLastNameCha
</template>

<script>
export default class ContactForm extends LightningElement {
    @track firstName = '';
    @track lastName = '';

    handleFirstNameChange(event) {
        this.firstName = event.target.value;
    }

    handleLastNameChange(event) {
        this.lastName = event.target.value;
    }
}
</script>
```

In this example, two properties `` `firstName` `` and `` `lastName` `` are defined and decorated with the `` `@track` `` decorator to indicate that their values will be reactive and trigger updates to the component's view whenever they change. The `` `lightning-input` `` components are used to display fields for the user to enter their first name and last name. The `` `handleFirstNameChange` `` and `` `handleLastNameChange` `` methods are used to update the corresponding properties when the user enters data in the input fields.

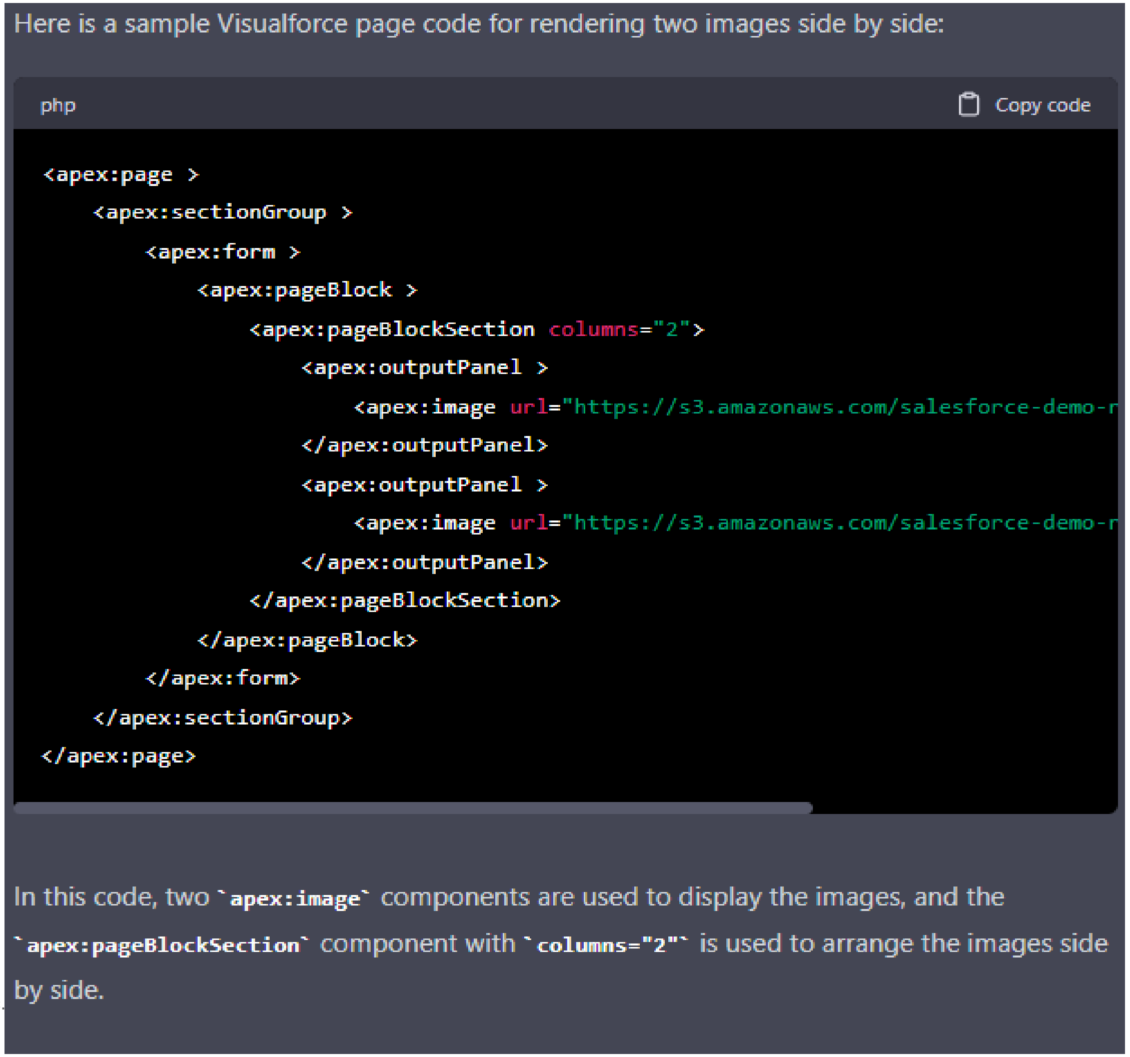
16. Write a code for creating a visual force page to render 2 images side by side

Copilot



```
<apex:page controller="testJavascript">
    <apex:form>
        <apex:pageBlock title="Image Switcher">
            <apex:pageBlockSection columns="2">
                <apex:pageBlockSectionItem>
                    <apex:image value="{!img1}" />
                </apex:pageBlockSectionItem>
                <apex:pageBlockSectionItem>
                    <apex:image value="{!img2}" />
                </apex:pageBlockSectionItem>
            </apex:pageBlockSection>
            <apex:commandButton value="Switch" action="{!switchImages}" />
        </apex:pageBlock>
    </apex:form>
</apex:page>
public with sharing class testJavascript {
    public String img1 {get;set;}
    public String img2 {get;set;}
    public testJavascript(){
        img1 = '/resource/forcecloud_logo';
        img2 = '/resource/forcecloud_logo';}
    public PageReference switchImages() {
        String temp = img1;
        img1 = img2;
        img2 = temp;
        return null;
    } }
<apex:page controller="testJavascript">
    <apex:form>
        <apex:pageBlock title="Image Switcher">
            <apex:pageBlockSection columns="2">
                <apex:pageBlockSectionItem>
                    <apex:image value="{!img1}" />
                </apex:pageBlockSectionItem>
                <apex:pageBlockSectionItem>
                    <apex:image value="{!img2}" />
                </apex:pageBlockSectionItem>
            </apex:pageBlockSection>
            <apex:commandButton value="Switch" action="{!switchImages}" />
```

ChatGPT



Here is a sample Visualforce page code for rendering two images side by side:

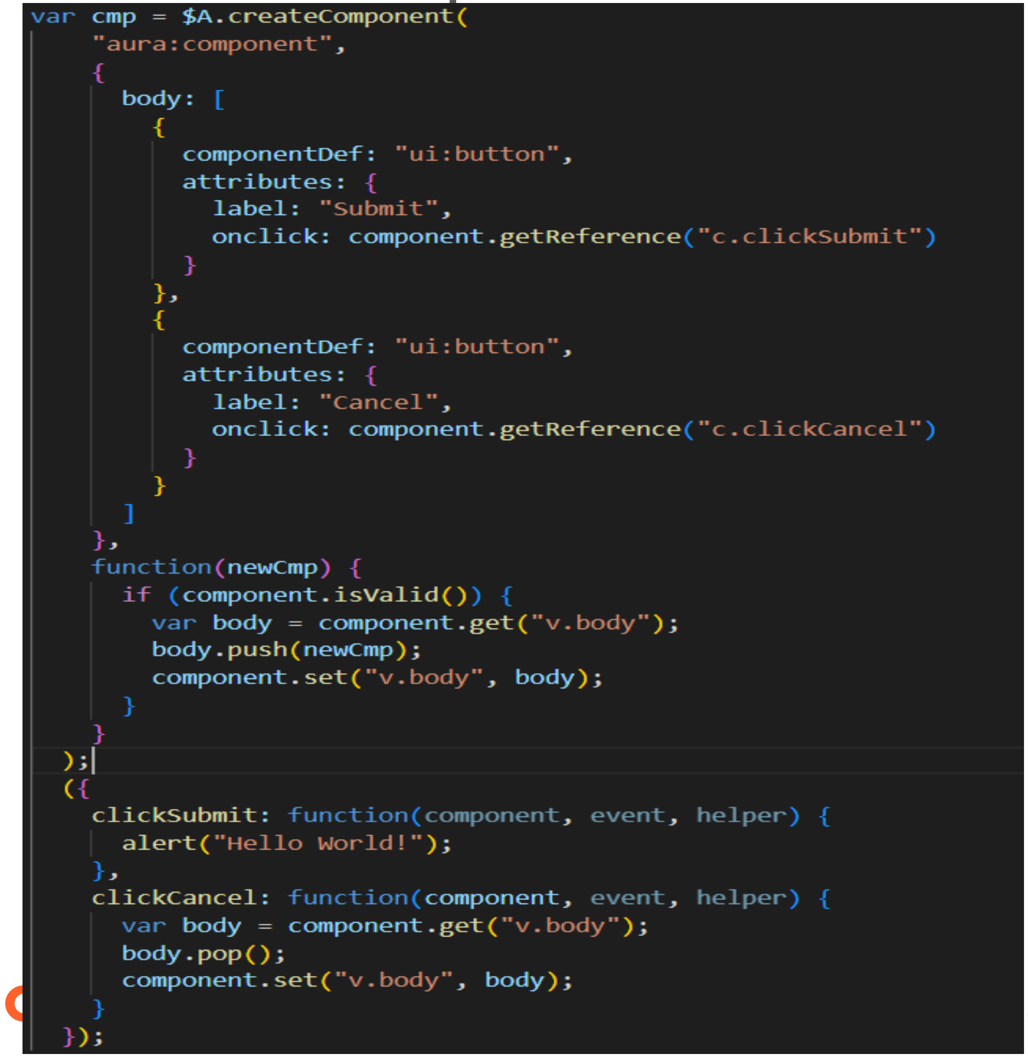
```php
<apex:page >
    <apex:sectionGroup >
        <apex:form >
            <apex:pageBlock >
                <apex:pageBlockSection columns="2">
                    <apex:outputPanel >
                        <apex:image url="https://s3.amazonaws.com/salesforce-demo-r
                    </apex:outputPanel>
                    <apex:outputPanel >
                        <apex:image url="https://s3.amazonaws.com/salesforce-demo-r
                    </apex:outputPanel>
                </apex:pageBlockSection>
            </apex:pageBlock>
        </apex:form>
    </apex:sectionGroup>
</apex:page>
```

In this code, two `apex:image` components are used to display the images, and the `apex:pageBlockSection` component with `columns="2"` is used to arrange the images side by side.

22 Optum, Inc.

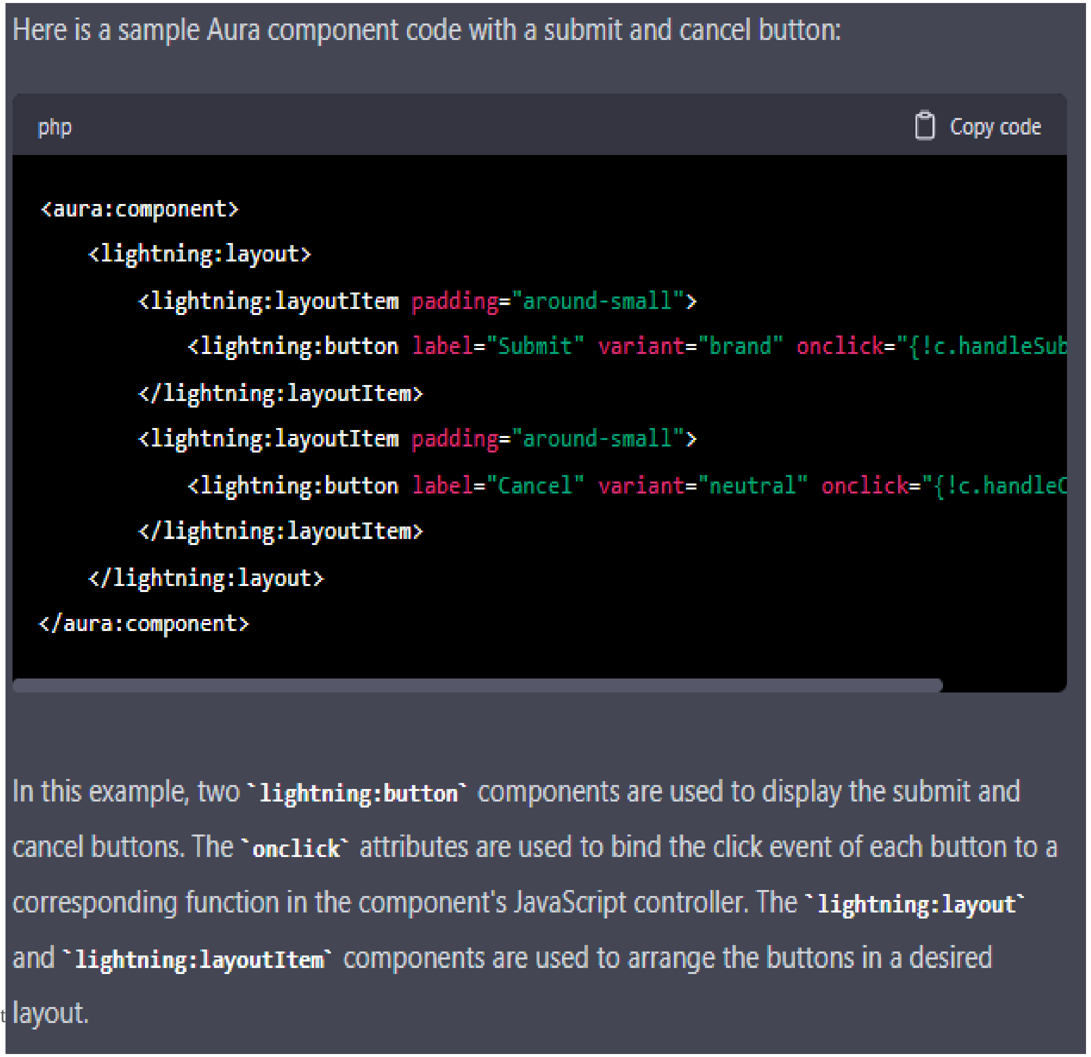
17. Write a code for creating an aura component with submit and cancel button:

Copilot



```
var cmp = $A.createComponent(
    "aura:component",
    {
      body: [
        {
          componentDef: "ui:button",
          attributes: {
            label: "Submit",
            onclick: component.getReference("c.clickSubmit")
          }
        },
        {
          componentDef: "ui:button",
          attributes: {
            label: "Cancel",
            onclick: component.getReference("c.clickCancel")
          }
        }
      ]
    },
    function(newCmp) {
      if (component.isValid()) {
        var body = component.get("v.body");
        body.push(newCmp);
        component.set("v.body", body);
      }
    }
  );
  ({
    clickSubmit: function(component, event, helper) {
      alert("Hello World!");
    },
    clickCancel: function(component, event, helper) {
      var body = component.get("v.body");
      body.pop();
      component.set("v.body", body);
    }
  });
```

ChatGPT



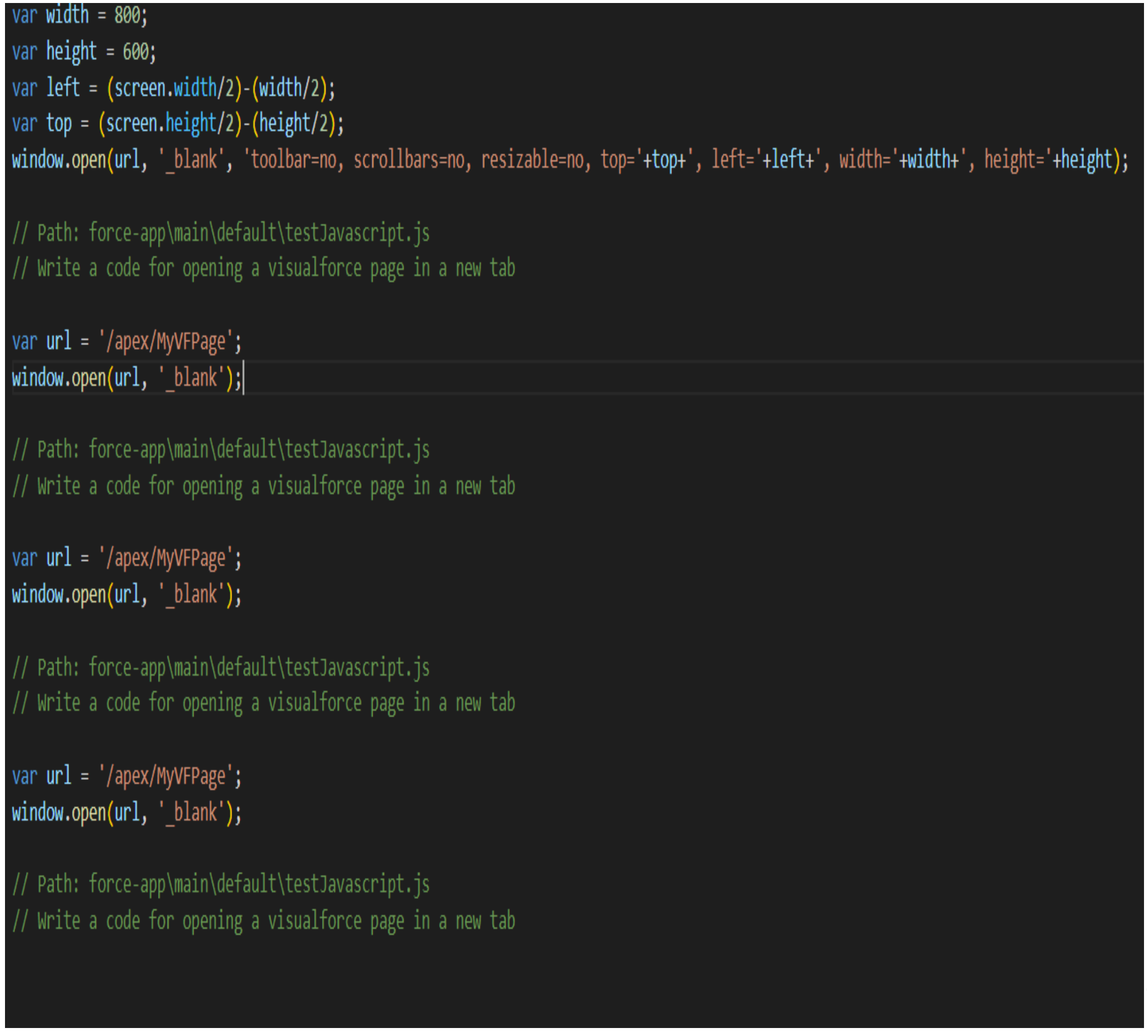
Here is a sample Aura component code with a submit and cancel button:

```php
<aura:component>
    <lightning:layout>
        <lightning:layoutItem padding="around-small">
            <lightning:button label="Submit" variant="brand" onclick="{!c.handleSub
        </lightning:layoutItem>
        <lightning:layoutItem padding="around-small">
            <lightning:button label="Cancel" variant="neutral" onclick="{!c.handleC
        </lightning:layoutItem>
    </lightning:layout>
</aura:component>
```

In this example, two `lightning:button` components are used to display the submit and cancel buttons. The `onclick` attributes are used to bind the click event of each button to a corresponding function in the component's JavaScript controller. The `lightning:layout` and `lightning:layoutItem` components are used to arrange the buttons in a desired layout.

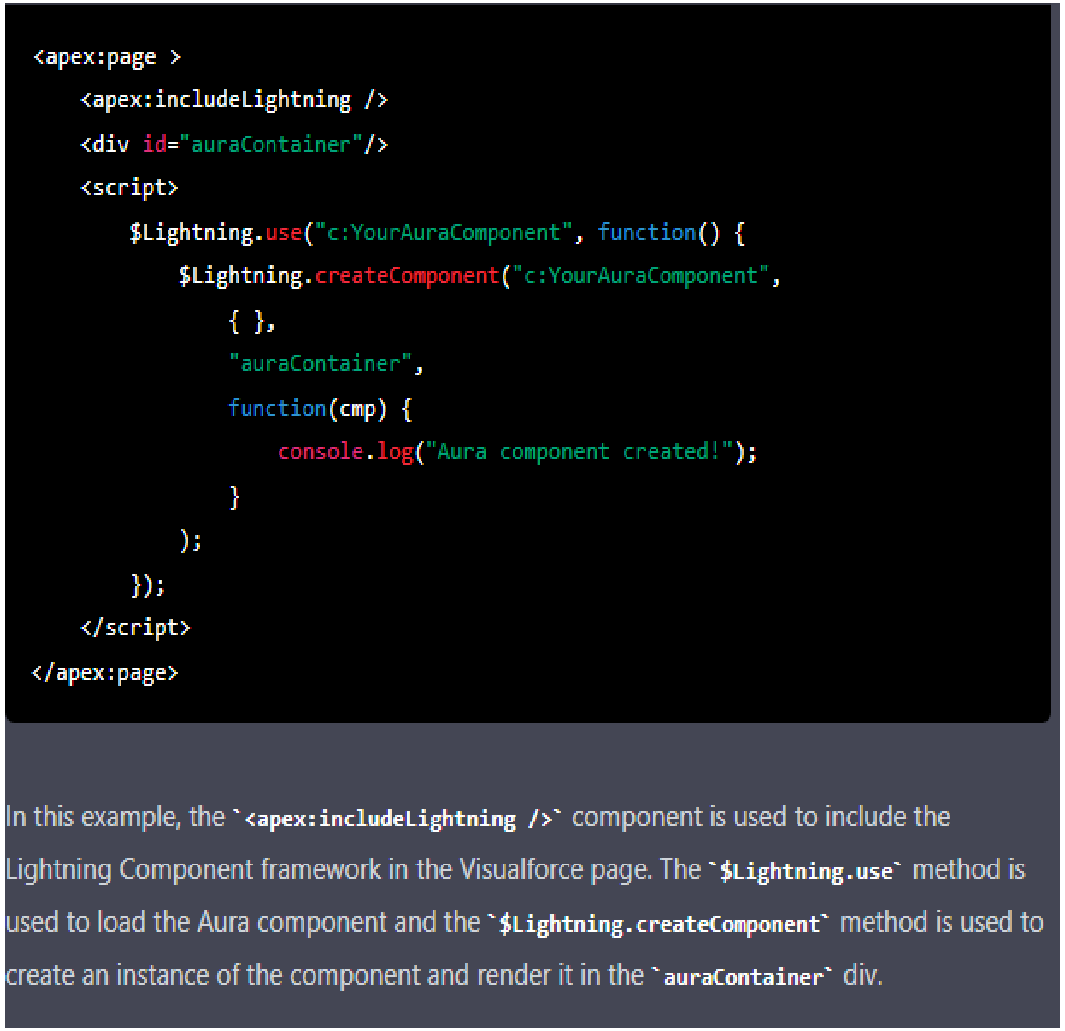
18. Write a code for opening an aura component in visual force page:

Copilot



```
var width = 800;
var height = 600;
var left = (screen.width/2)-(width/2);
var top = (screen.height/2)-(height/2);
window.open(url, '_blank', 'toolbar=no, scrollbars=no, resizable=no, top='+top+', left='+left+', width='+width+', height='+height);

// Path: force-app\main\default\testJavascript.js
// Write a code for opening a visualforce page in a new tab

var url = '/apex/MyVFPage';
window.open(url, '_blank');

// Path: force-app\main\default\testJavascript.js
// Write a code for opening a visualforce page in a new tab

var url = '/apex/MyVFPage';
window.open(url, '_blank');

// Path: force-app\main\default\testJavascript.js
// Write a code for opening a visualforce page in a new tab

var url = '/apex/MyVFPage';
window.open(url, '_blank');

// Path: force-app\main\default\testJavascript.js
// Write a code for opening a visualforce page in a new tab
```

ChatGPT

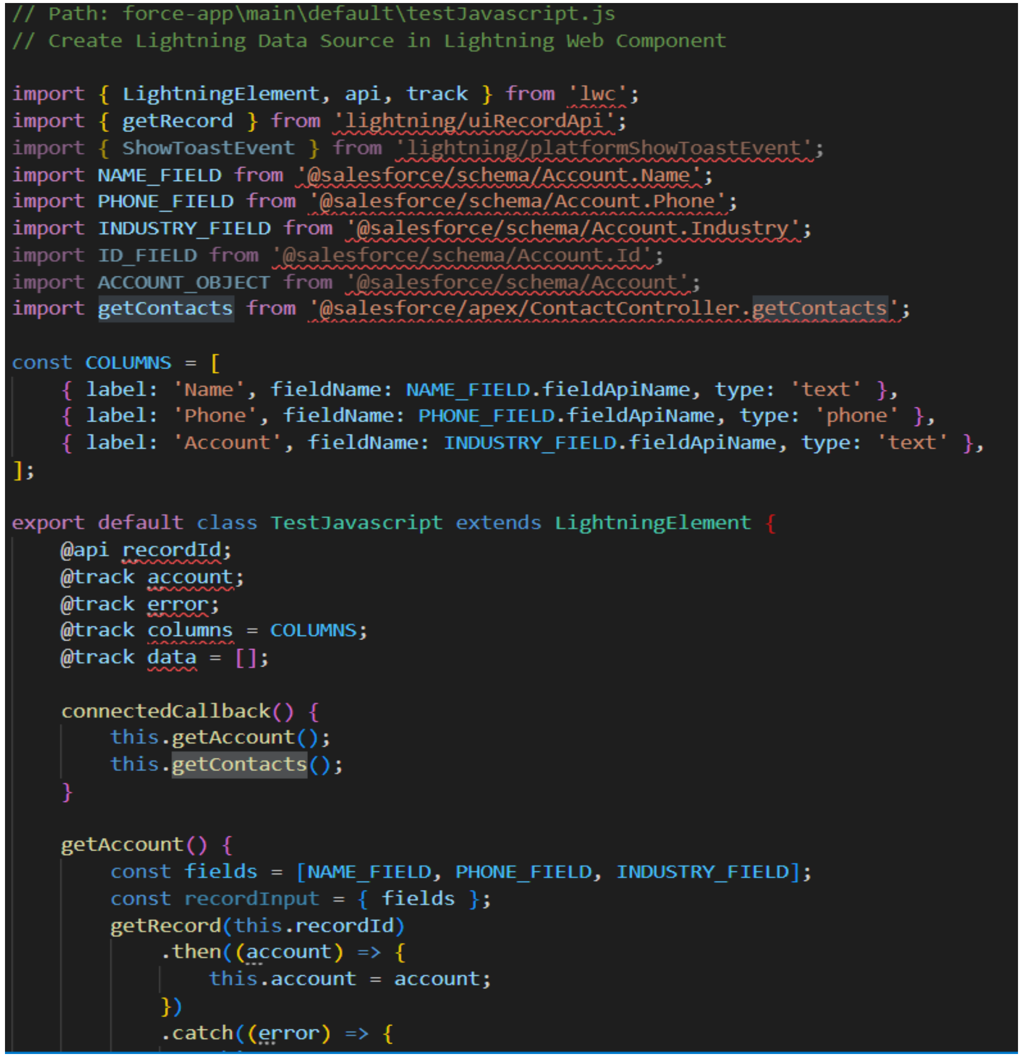


```
<apex:page >
    <apex:includeLightning />
    <div id="auraContainer"/>
    <script>
        $Lightning.use("c:YourAuraComponent", function() {
            $Lightning.createComponent("c:YourAuraComponent",
                { },
                "auraContainer",
                function(cmp) {
                    console.log("Aura component created!");
                }
            );
        });
    </script>
</apex:page>
```

In this example, the `<apex:includeLightning />` component is used to include the Lightning Component framework in the Visualforce page. The `$Lightning.use` method is used to load the Aura component and the `$Lightning.createComponent` method is used to create an instance of the component and render it in the `auraContainer` div.

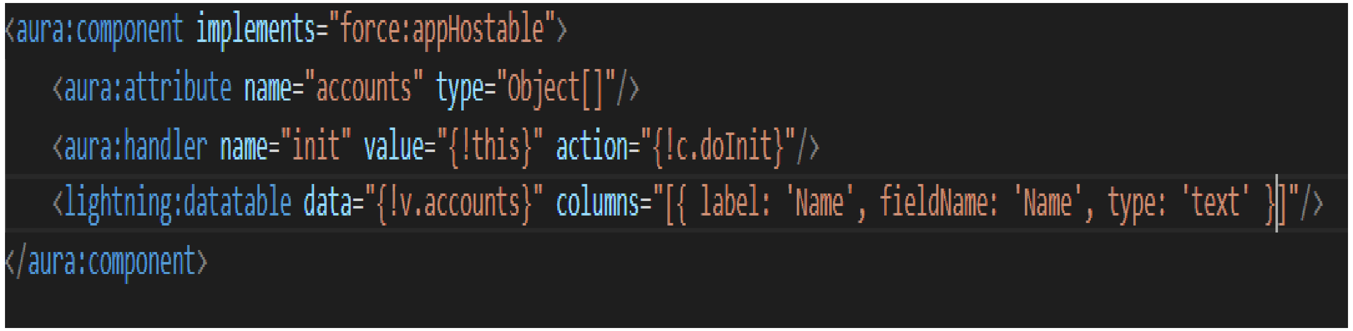
19. Create Lightning Data Source in Aura Component:

Copilot



```javascript
// Path: force-app\main\default\testJavascript.js
// Create Lightning Data Source in Lightning Web Component

import { LightningElement, api, track } from 'lwc';
import { getRecord } from 'lightning/uiRecordApi';
import { ShowToastEvent } from 'lightning/platformShowToastEvent';
import NAME_FIELD from '@salesforce/schema/Account.Name';
import PHONE_FIELD from '@salesforce/schema/Account.Phone';
import INDUSTRY_FIELD from '@salesforce/schema/Account.Industry';
import ID_FIELD from '@salesforce/schema/Account.Id';
import ACCOUNT_OBJECT from '@salesforce/schema/Account';
import getContacts from '@salesforce/apex/ContactController.getContacts';

const COLUMNS = [
    { label: 'Name', fieldName: NAME_FIELD.fieldApiName, type: 'text' },
    { label: 'Phone', fieldName: PHONE_FIELD.fieldApiName, type: 'phone' },
    { label: 'Account', fieldName: INDUSTRY_FIELD.fieldApiName, type: 'text' },
];

export default class TestJavascript extends LightningElement {
    @api recordId;
    @track account;
    @track error;
    @track columns = COLUMNS;
    @track data = [];

    connectedCallback() {
        this.getAccount();
        this.getContacts();
    }

    getAccount() {
        const fields = [NAME_FIELD, PHONE_FIELD, INDUSTRY_FIELD];
        const recordInput = { fields };
        getRecord(this.recordId)
            .then((account) => {
                this.account = account;
            })
            .catch((error) => {
```

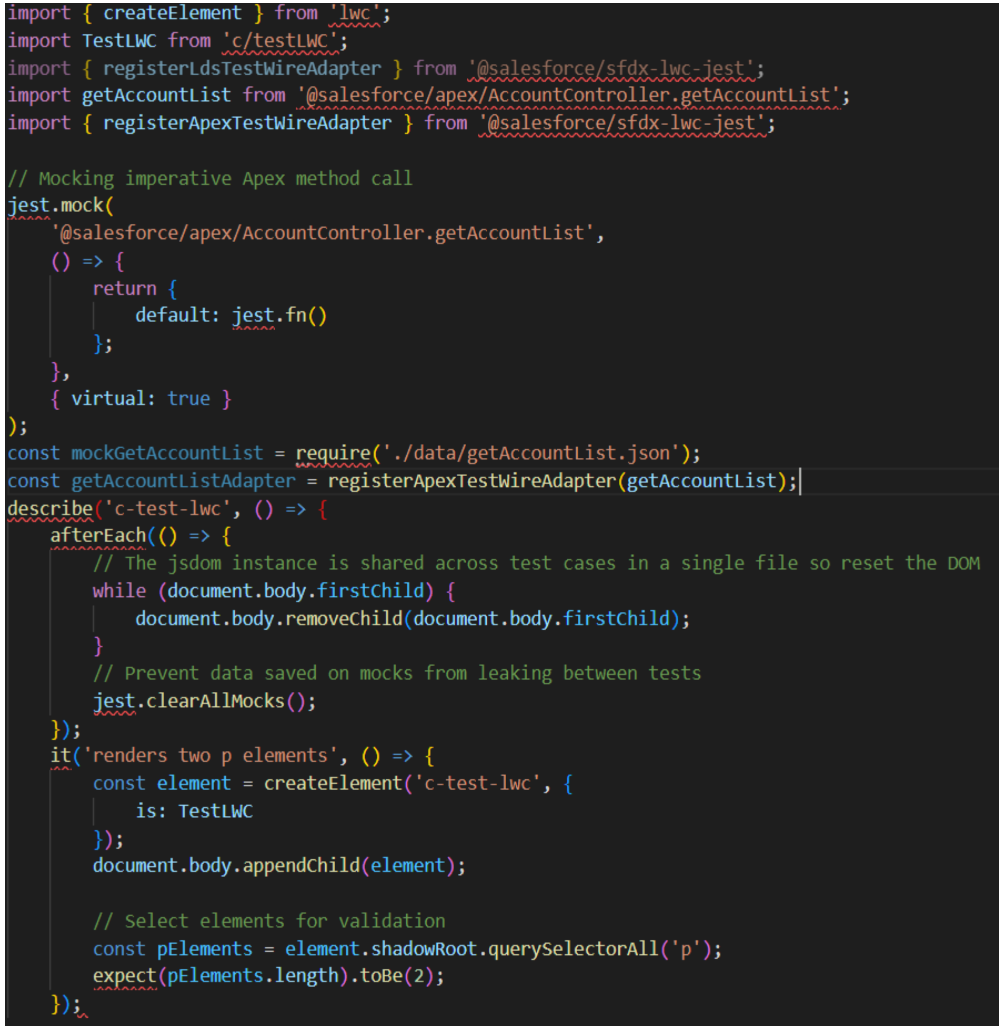
ChatGPT



```xml
<aura:component implements="force:appHostable">
    <aura:attribute name="accounts" type="Object[]"/>
    <aura:handler name="init" value="{!this}" action="{!c.doInit}"/>
    <lightning:datatable data="{!v.accounts}" columns="[{ label: 'Name', fieldName: 'Name', type: 'text' }]"/>
</aura:component>
```

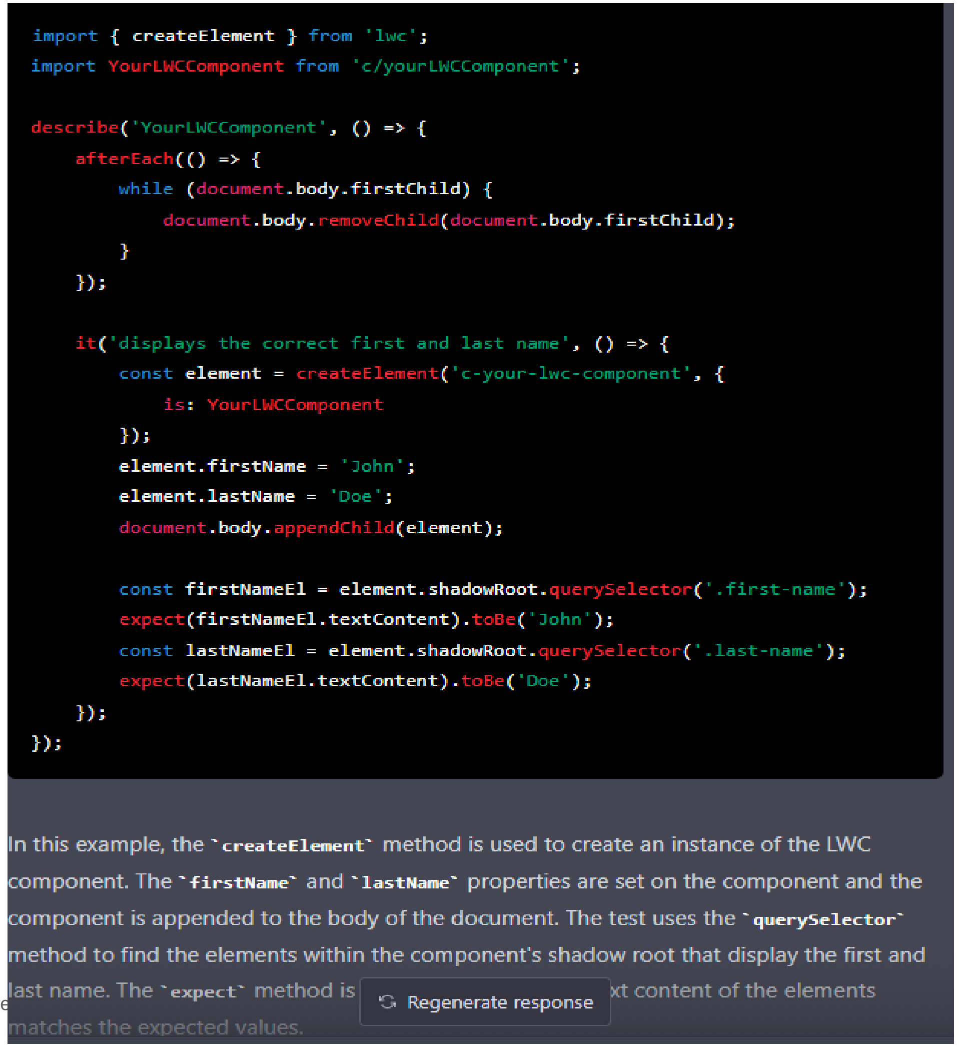
20. Write Test class for LWC:

Copilot



```
import { createElement } from 'lwc';
import TestLWC from 'c/testLWC';
import { registerLdsTestWireAdapter } from '@salesforce/sfdx-lwc-jest';
import getAccountList from '@salesforce/apex/AccountController.getAccountList';
import { registerApexTestWireAdapter } from '@salesforce/sfdx-lwc-jest';

// Mocking imperative Apex method call
jest.mock(
    '@salesforce/apex/AccountController.getAccountList',
    () => {
        return {
            default: jest.fn()
        };
    },
    { virtual: true }
);
const mockGetAccountList = require('./data/getAccountList.json');
const getAccountListAdapter = registerApexTestWireAdapter(getAccountList);
describe('c-test-lwc', () => {
    afterEach(() => {
        // The jsdom instance is shared across test cases in a single file so reset the DOM
        while (document.body.firstChild) {
            document.body.removeChild(document.body.firstChild);
        }
        // Prevent data saved on mocks from leaking between tests
        jest.clearAllMocks();
    });
    it('renders two p elements', () => {
        const element = createElement('c-test-lwc', {
            is: TestLWC
        });
        document.body.appendChild(element);

        // Select elements for validation
        const pElements = element.shadowRoot.querySelectorAll('p');
        expect(pElements.length).toBe(2);
    });
```

ChatGPT

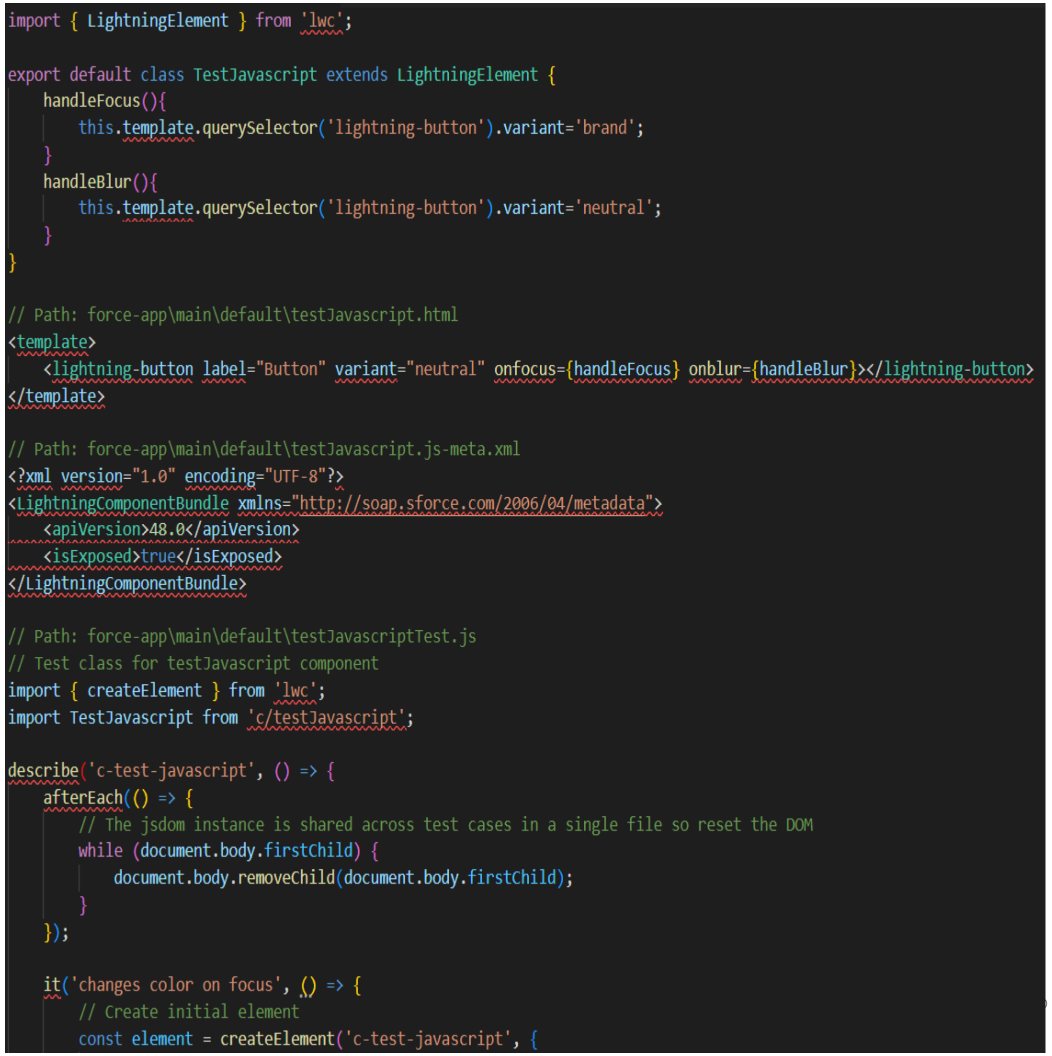


```
import { createElement } from 'lwc';
import YourLWCComponent from 'c/yourLWCComponent';

describe('YourLWCComponent', () => {
    afterEach(() => {
        while (document.body.firstChild) {
            document.body.removeChild(document.body.firstChild);
        }
    });

    it('displays the correct first and last name', () => {
        const element = createElement('c-your-lwc-component', {
            is: YourLWCComponent
        });
        element.firstName = 'John';
        element.lastName = 'Doe';
        document.body.appendChild(element);

        const firstNameEl = element.shadowRoot.querySelector('.first-name');
        expect(firstNameEl.textContent).toBe('John');
        const lastNameEl = element.shadowRoot.querySelector('.last-name');
        expect(lastNameEl.textContent).toBe('Doe');
    });
});
```

In this example, the `createElement` method is used to create an instance of the LWC component. The `firstName` and `lastName` properties are set on the component and the component is appended to the body of the document. The test uses the `querySelector` method to find the elements within the component's shadow root that display the first and last name. The `expect` method is xt content of the elements matches the expected values.

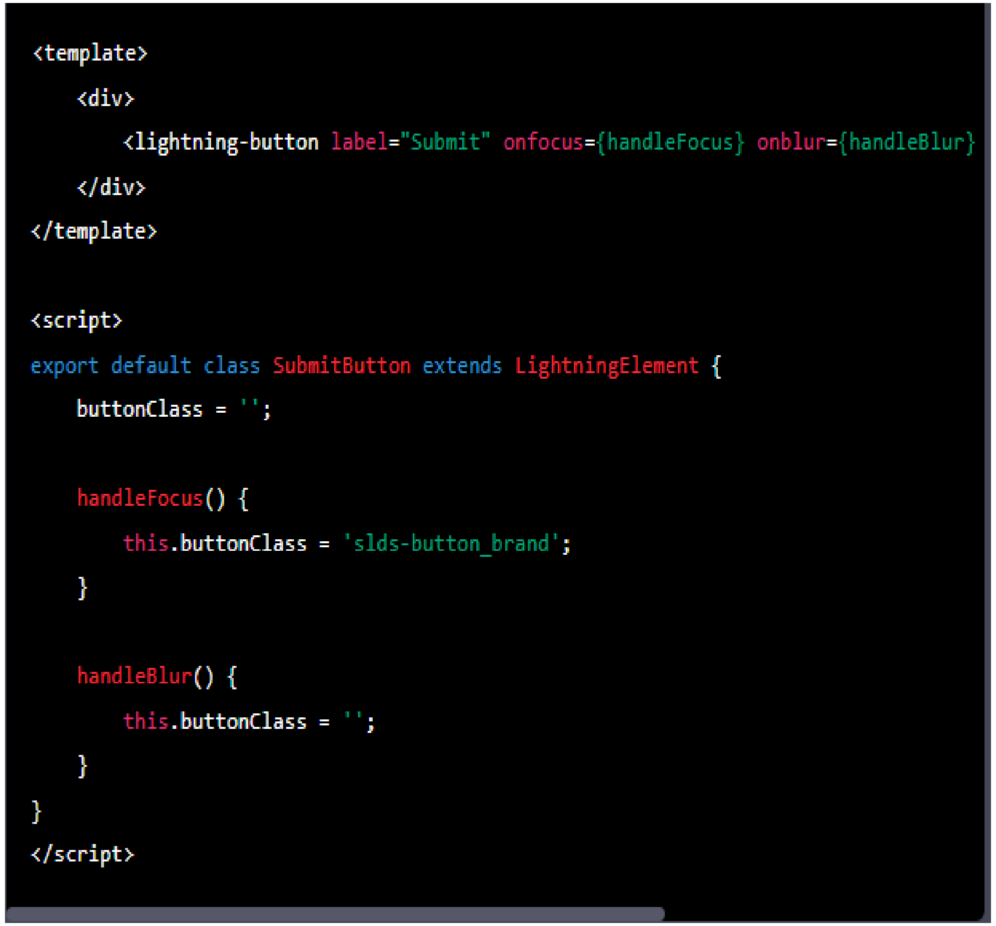
21. Create a lightning component with submit button, on focus change button colour to blue

Copilot



```
import { LightningElement } from 'lwc';

export default class TestJavascript extends LightningElement {
    handleFocus(){
        this.template.querySelector('lightning-button').variant='brand';
    }
    handleBlur(){
        this.template.querySelector('lightning-button').variant='neutral';
    }
}

// Path: force-app\main\default\testJavascript.html
<template>
    <lightning-button label="Button" variant="neutral" onfocus={handleFocus} onblur={handleBlur}></lightning-button>
</template>

// Path: force-app\main\default\testJavascript.js-meta.xml
<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
    <apiVersion>48.0</apiVersion>
    <isExposed>true</isExposed>
</LightningComponentBundle>

// Path: force-app\main\default\testJavascriptTest.js
// Test class for testJavascript component
import { createElement } from 'lwc';
import TestJavascript from 'c/testJavascript';

describe('c-test-javascript', () => {
    afterEach(() => {
        // The jsdom instance is shared across test cases in a single file so reset the DOM
        while (document.body.firstChild) {
            document.body.removeChild(document.body.firstChild);
        }
    });

    it('changes color on focus', () => {
        // Create initial element
        const element = createElement('c-test-javascript', {
```

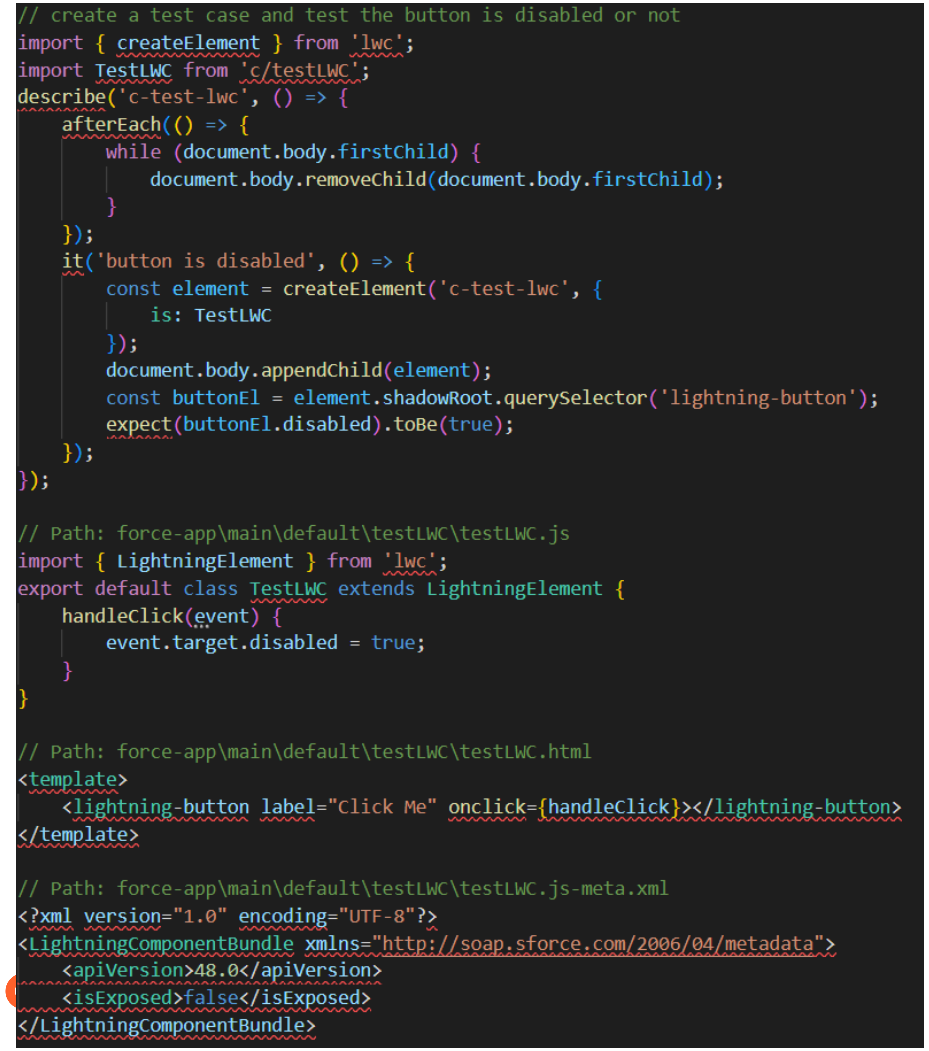
ChatGPT



```
<template>
    <div>
        <lightning-button label="Submit" onfocus={handleFocus} onblur={handleBlur}
    </div>
</template>

<script>
export default class SubmitButton extends LightningElement {
    buttonClass = '';

    handleFocus() {
        this.buttonClass = 'slds-button_brand';
    }

    handleBlur() {
        this.buttonClass = '';
    }
}
</script>
```

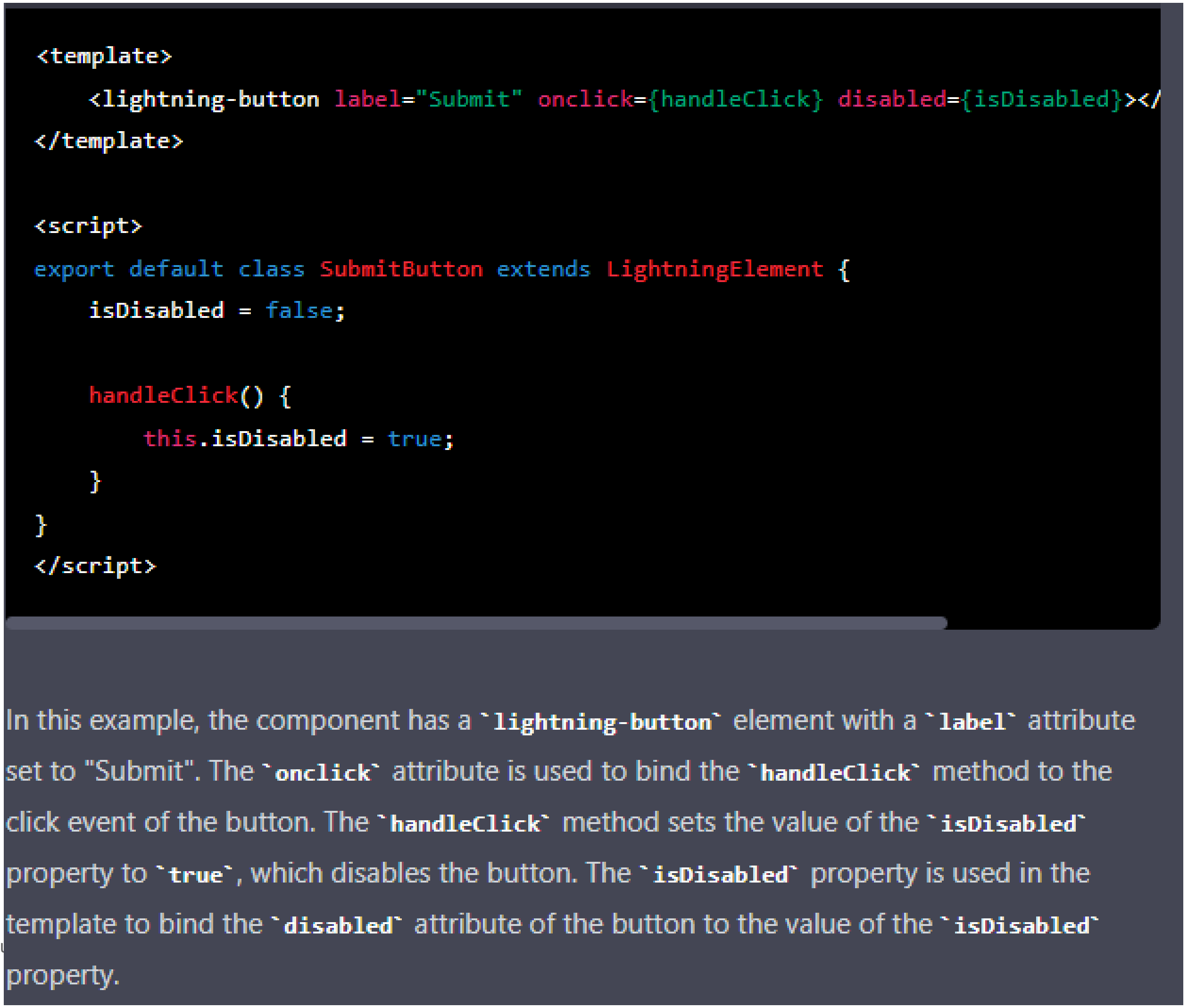
22. Create a lwc component with a button, on click disable the button:

Copilot



```
// create a test case and test the button is disabled or not
import { createElement } from 'lwc';
import TestLWC from 'c/testLWC';
describe('c-test-lwc', () => {
    afterEach(() => {
        while (document.body.firstChild) {
            document.body.removeChild(document.body.firstChild);
        }
    });
    it('button is disabled', () => {
        const element = createElement('c-test-lwc', {
            is: TestLWC
        });
        document.body.appendChild(element);
        const buttonEl = element.shadowRoot.querySelector('lightning-button');
        expect(buttonEl.disabled).toBe(true);
    });
});

// Path: force-app\main\default\testLWC\testLWC.js
import { LightningElement } from 'lwc';
export default class TestLWC extends LightningElement {
    handleClick(event) {
        event.target.disabled = true;
    }
}

// Path: force-app\main\default\testLWC\testLWC.html
<template>
    <lightning-button label="Click Me" onclick={handleClick}></lightning-button>
</template>

// Path: force-app\main\default\testLWC\testLWC.js-meta.xml
<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
    <apiVersion>48.0</apiVersion>
    <isExposed>false</isExposed>
</LightningComponentBundle>
```

ChatGPT



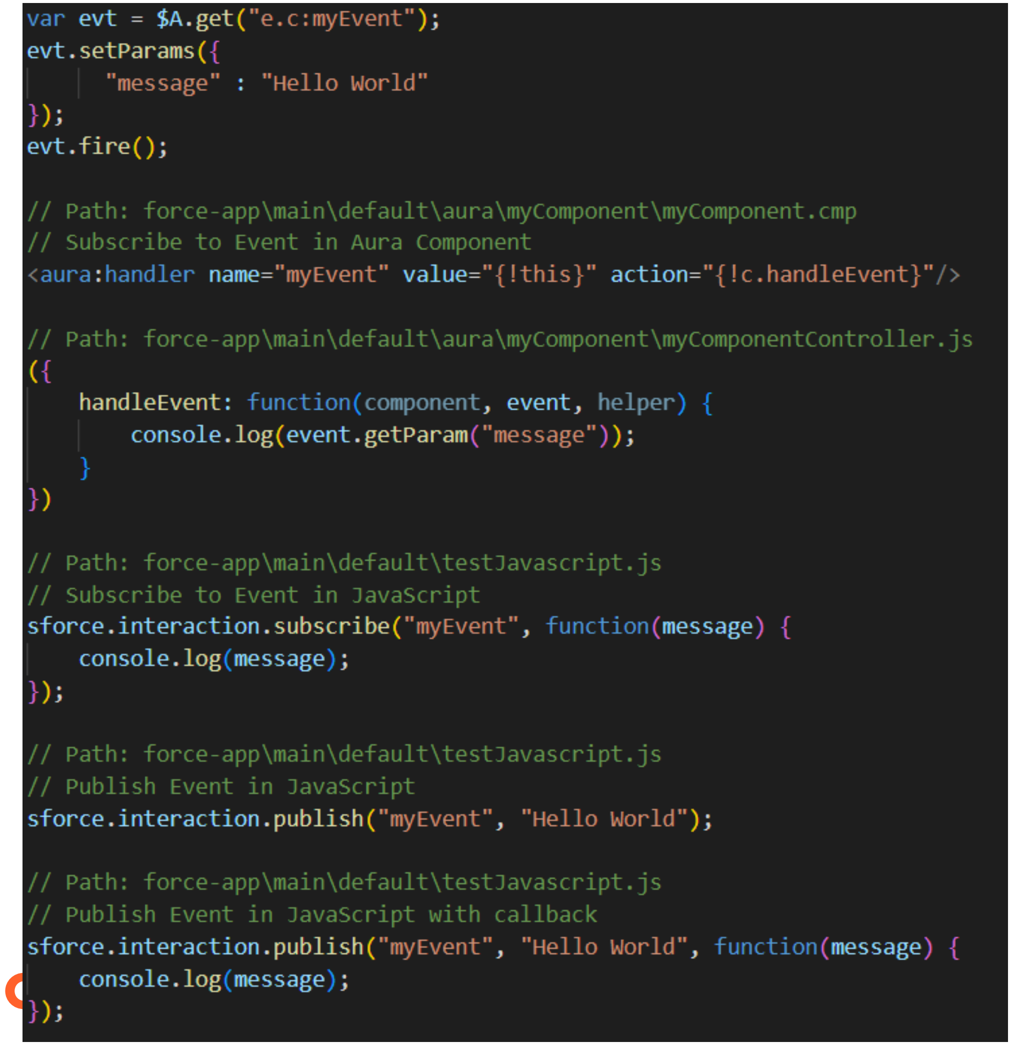
```
<template>
    <lightning-button label="Submit" onclick={handleClick} disabled={isDisabled}></
</template>

<script>
export default class SubmitButton extends LightningElement {
    isDisabled = false;

    handleClick() {
        this.isDisabled = true;
    }
}
</script>
```

In this example, the component has a `lightning-button` element with a `label` attribute set to "Submit". The `onclick` attribute is used to bind the `handleClick` method to the click event of the button. The `handleClick` method sets the value of the `isDisabled` property to `true`, which disables the button. The `isDisabled` property is used in the template to bind the `disabled` attribute of the button to the value of the `isDisabled` property.

© 2022 Opt

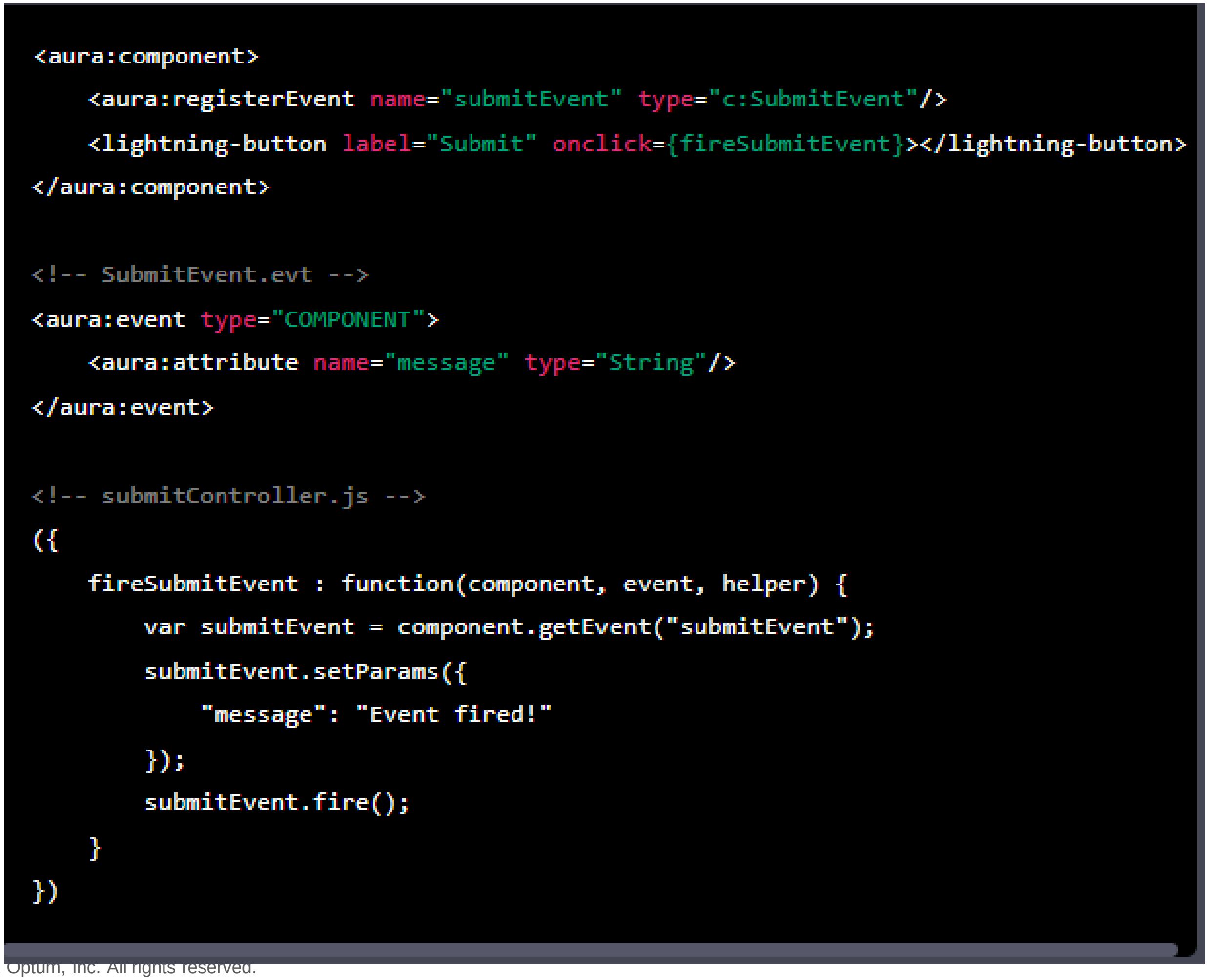
23. Create Event in Aura Component

Copilot



```
var evt = $A.get("e.c:myEvent");
evt.setParams({
    "message" : "Hello World"
});
evt.fire();

// Path: force-app\main\default\aura\myComponent\myComponent.cmp
// Subscribe to Event in Aura Component
<aura:handler name="myEvent" value="{!this}" action="{!c.handleEvent}"/>

// Path: force-app\main\default\aura\myComponent\myComponentController.js
({
    handleEvent: function(component, event, helper) {
        console.log(event.getParam("message"));
    }
})

// Path: force-app\main\default\testJavascript.js
// Subscribe to Event in JavaScript
sforce.interaction.subscribe("myEvent", function(message) {
    console.log(message);
});

// Path: force-app\main\default\testJavascript.js
// Publish Event in JavaScript
sforce.interaction.publish("myEvent", "Hello World");

// Path: force-app\main\default\testJavascript.js
// Publish Event in JavaScript with callback
sforce.interaction.publish("myEvent", "Hello World", function(message) {
    console.log(message);
});
```

ChatGPT

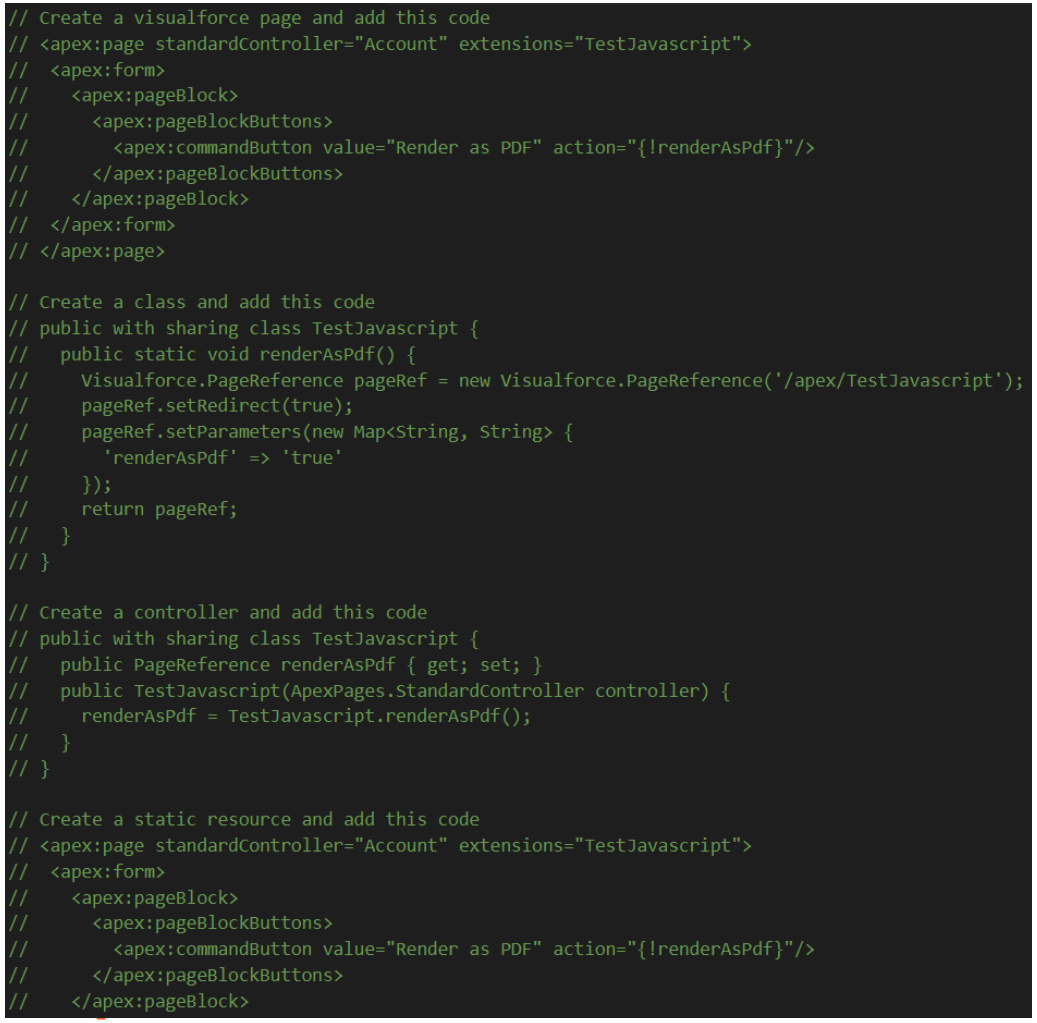


```
<aura:component>
    <aura:registerEvent name="submitEvent" type="c:SubmitEvent"/>
    <lightning-button label="Submit" onclick={fireSubmitEvent}></lightning-button>
</aura:component>

<!-- SubmitEvent.evt -->
<aura:event type="COMPONENT">
    <aura:attribute name="message" type="String"/>
</aura:event>

<!-- submitController.js -->
({
    fireSubmitEvent : function(component, event, helper) {
        var submitEvent = component.getEvent("submitEvent");
        submitEvent.setParams({
            "message": "Event fired!"
        });
        submitEvent.fire();
    }
})
```

© 2022 Optum, Inc. All rights reserved.

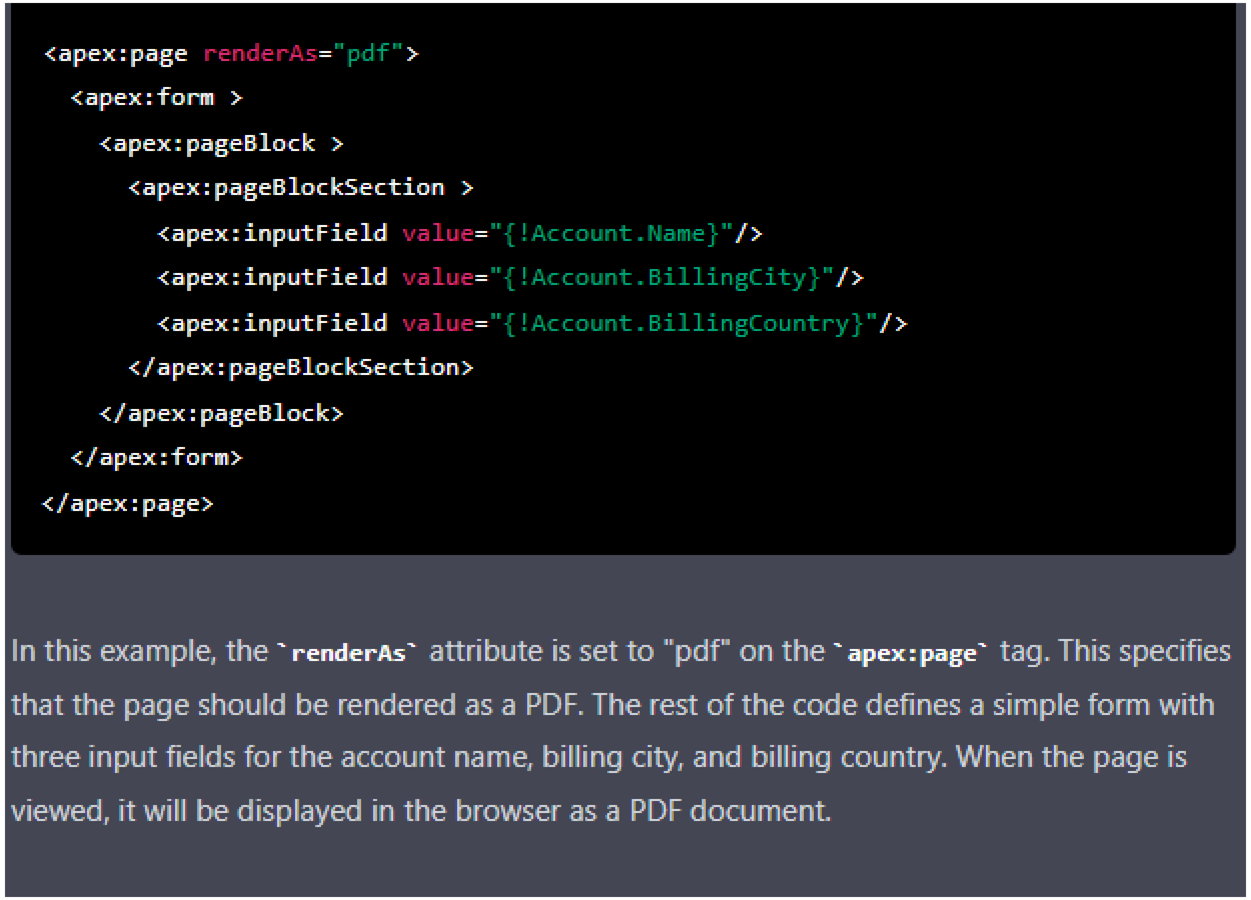
24. Create a visualforce page render as pdf

Copilot



```
// Create a visualforce page and add this code
// <apex:page standardController="Account" extensions="TestJavascript">
//  <apex:form>
//    <apex:pageBlock>
//      <apex:pageBlockButtons>
//        <apex:commandButton value="Render as PDF" action="{!renderAsPdf}"/>
//      </apex:pageBlockButtons>
//    </apex:pageBlock>
//  </apex:form>
// </apex:page>

// Create a class and add this code
// public with sharing class TestJavascript {
//   public static void renderAsPdf() {
//     Visualforce.PageReference pageRef = new Visualforce.PageReference('/apex/TestJavascript');
//     pageRef.setRedirect(true);
//     pageRef.setParameters(new Map<String, String> {
//       'renderAsPdf' => 'true'
//     });
//     return pageRef;
//   }
// }

// Create a controller and add this code
// public with sharing class TestJavascript {
//   public PageReference renderAsPdf { get; set; }
//   public TestJavascript(ApexPages.StandardController controller) {
//     renderAsPdf = TestJavascript.renderAsPdf();
//   }
// }

// Create a static resource and add this code
// <apex:page standardController="Account" extensions="TestJavascript">
//  <apex:form>
//    <apex:pageBlock>
//      <apex:pageBlockButtons>
//        <apex:commandButton value="Render as PDF" action="{!renderAsPdf}"/>
//      </apex:pageBlockButtons>
//    </apex:pageBlock>
```

ChatGPT



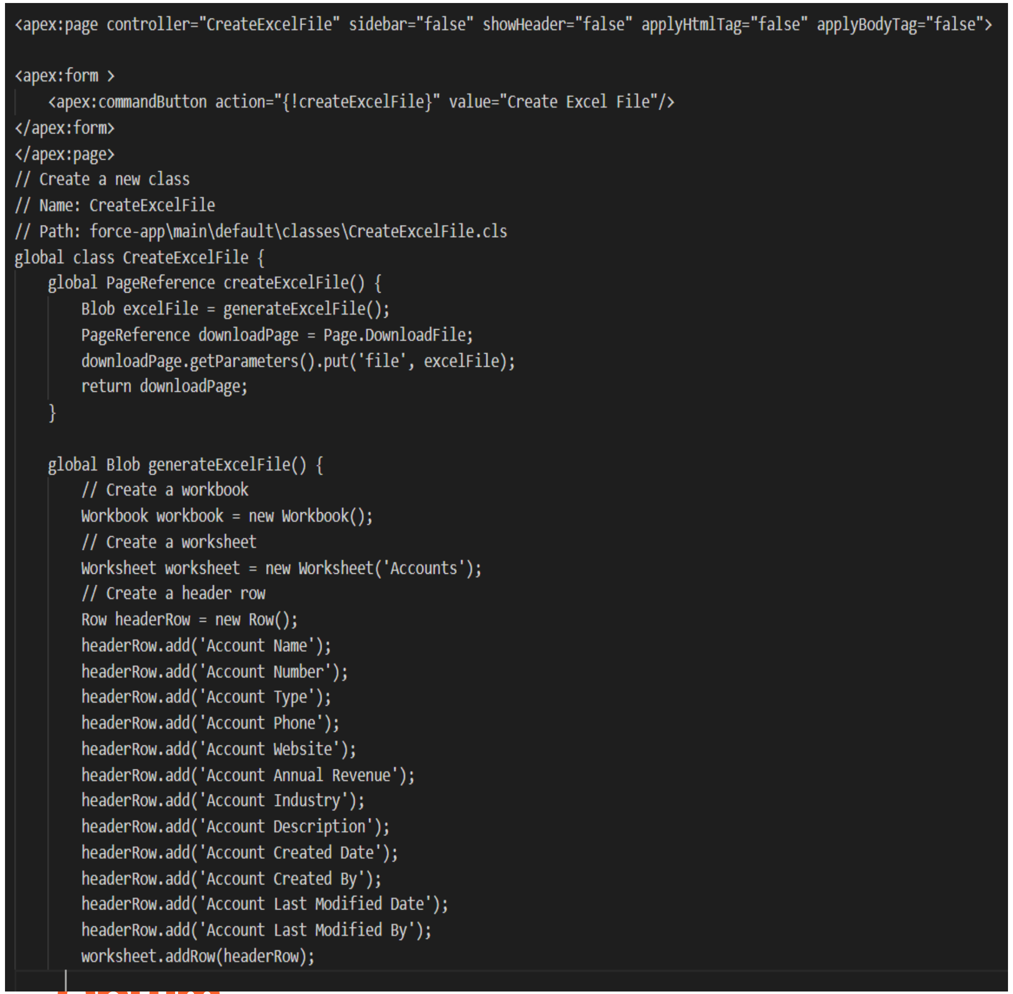
```
<apex:page renderAs="pdf">
  <apex:form >
    <apex:pageBlock >
      <apex:pageBlockSection >
        <apex:inputField value="{!Account.Name}"/>
        <apex:inputField value="{!Account.BillingCity}"/>
        <apex:inputField value="{!Account.BillingCountry}"/>
      </apex:pageBlockSection>
    </apex:pageBlock>
  </apex:form>
</apex:page>
```

In this example, the `renderAs` attribute is set to "pdf" on the `apex:page` tag. This specifies that the page should be rendered as a PDF. The rest of the code defines a simple form with three input fields for the account name, billing city, and billing country. When the page is viewed, it will be displayed in the browser as a PDF document.

© 2022 Optum, Inc. All rights reserved.

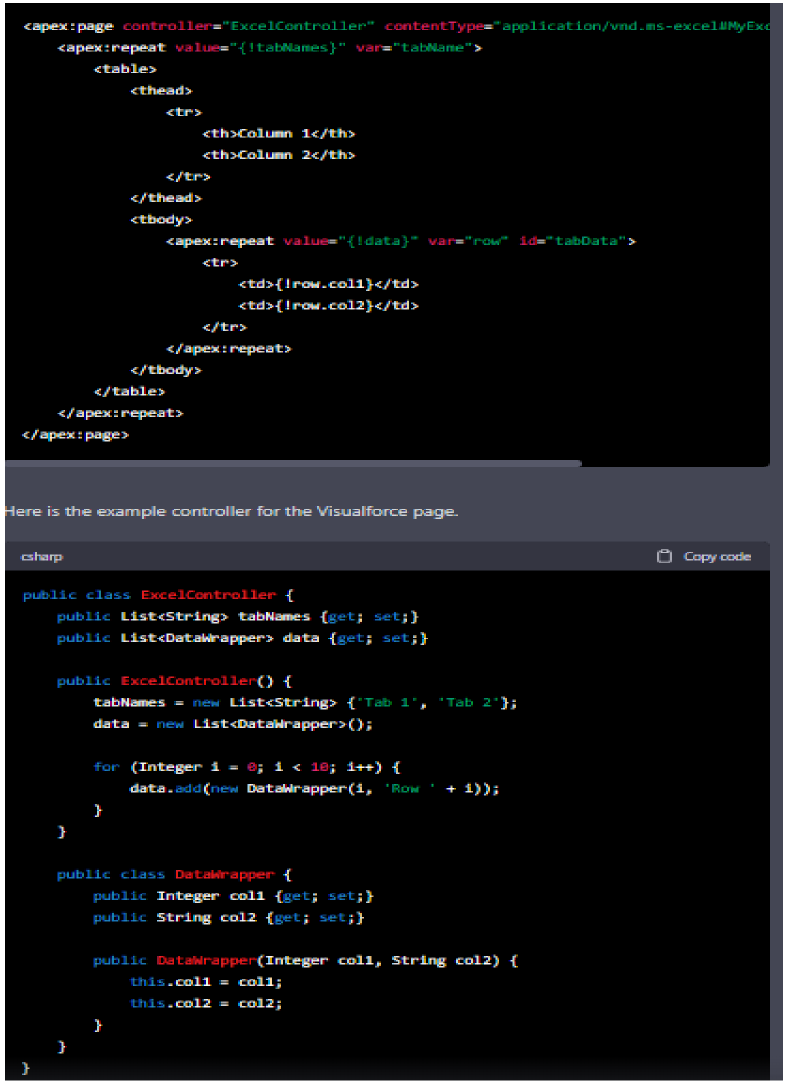
25. Create a visualforce page that can create a excel file with two tabs

Copilot



```
<apex:page controller="CreateExcelFile" sidebar="false" showHeader="false" applyHtmlTag="false" applyBodyTag="false">

<apex:form >
    <apex:commandButton action="{!createExcelFile}" value="Create Excel File"/>
</apex:form>
</apex:page>
// Create a new class
// Name: CreateExcelFile
// Path: force-app\main\default\classes\CreateExcelFile.cls
global class CreateExcelFile {
    global PageReference createExcelFile() {
        Blob excelFile = generateExcelFile();
        PageReference downloadPage = Page.DownloadFile;
        downloadPage.getParameters().put('file', excelFile);
        return downloadPage;
    }

    global Blob generateExcelFile() {
        // Create a workbook
        Workbook workbook = new Workbook();
        // Create a worksheet
        Worksheet worksheet = new Worksheet('Accounts');
        // Create a header row
        Row headerRow = new Row();
        headerRow.add('Account Name');
        headerRow.add('Account Number');
        headerRow.add('Account Type');
        headerRow.add('Account Phone');
        headerRow.add('Account Website');
        headerRow.add('Account Annual Revenue');
        headerRow.add('Account Industry');
        headerRow.add('Account Description');
        headerRow.add('Account Created Date');
        headerRow.add('Account Created By');
        headerRow.add('Account Last Modified Date');
        headerRow.add('Account Last Modified By');
        worksheet.addRow(headerRow);
```

ChatGPT



```
<apex:page controller="ExcelController" contentType="application/vnd.ms-excel#MyExc
    <apex:repeat value="{!tabNames}" var="tabName">
        <table>
            <thead>
                <tr>
                    <th>Column 1</th>
                    <th>Column 2</th>
                </tr>
            </thead>
            <tbody>
                <apex:repeat value="{!data}" var="row" id="tabData">
                    <tr>
                        <td>{!row.col1}</td>
                        <td>{!row.col2}</td>
                    </tr>
                </apex:repeat>
            </tbody>
        </table>
    </apex:repeat>
</apex:page>
```

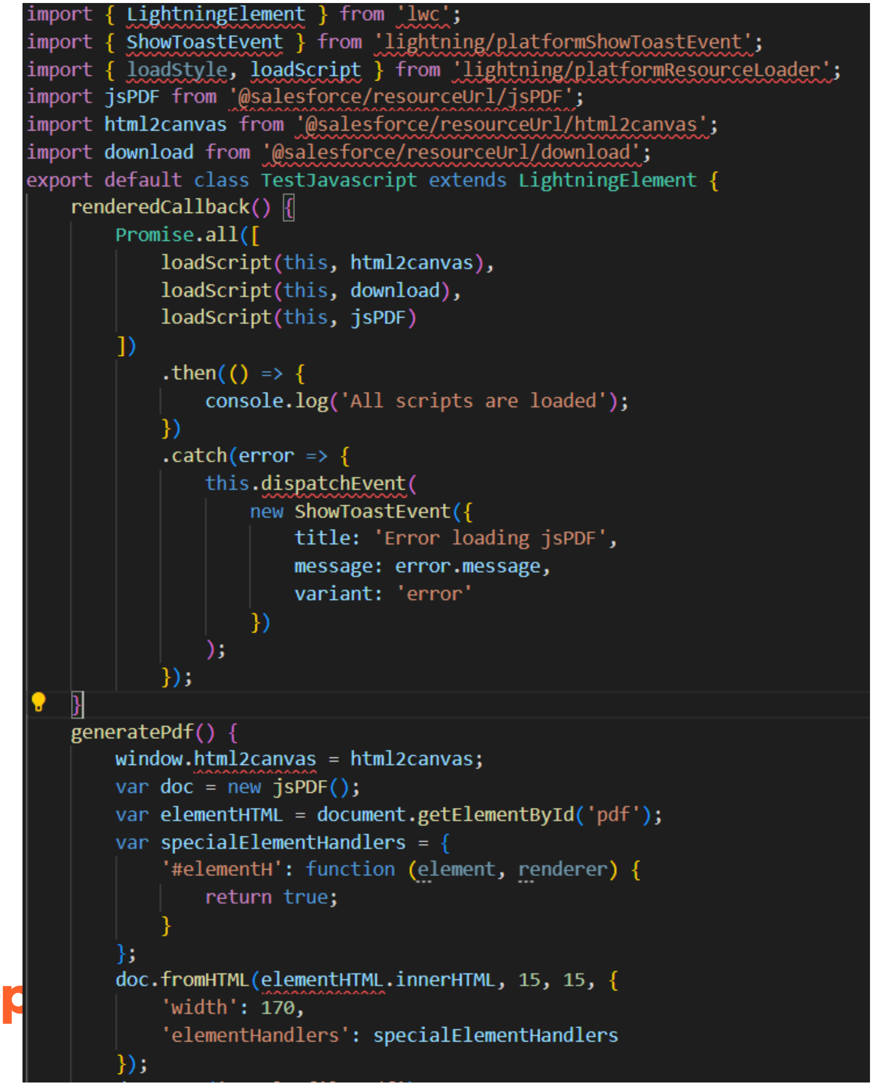
Here is the example controller for the Visualforce page.

```
public class ExcelController {
    public List<String> tabNames {get; set;}
    public List<DataWrapper> data {get; set;}

    public ExcelController() {
        tabNames = new List<String> {'Tab 1', 'Tab 2'};
        data = new List<DataWrapper>();

        for (Integer i = 0; i < 10; i++) {
            data.add(new DataWrapper(i, 'Row ' + i));
        }
    }

    public class DataWrapper {
        public Integer col1 {get; set;}
        public String col2 {get; set;}

        public DataWrapper(Integer col1, String col2) {
            this.col1 = col1;
            this.col2 = col2;
        }
    }
}
```

© 2022 Optum, Inc. All rights reserved.

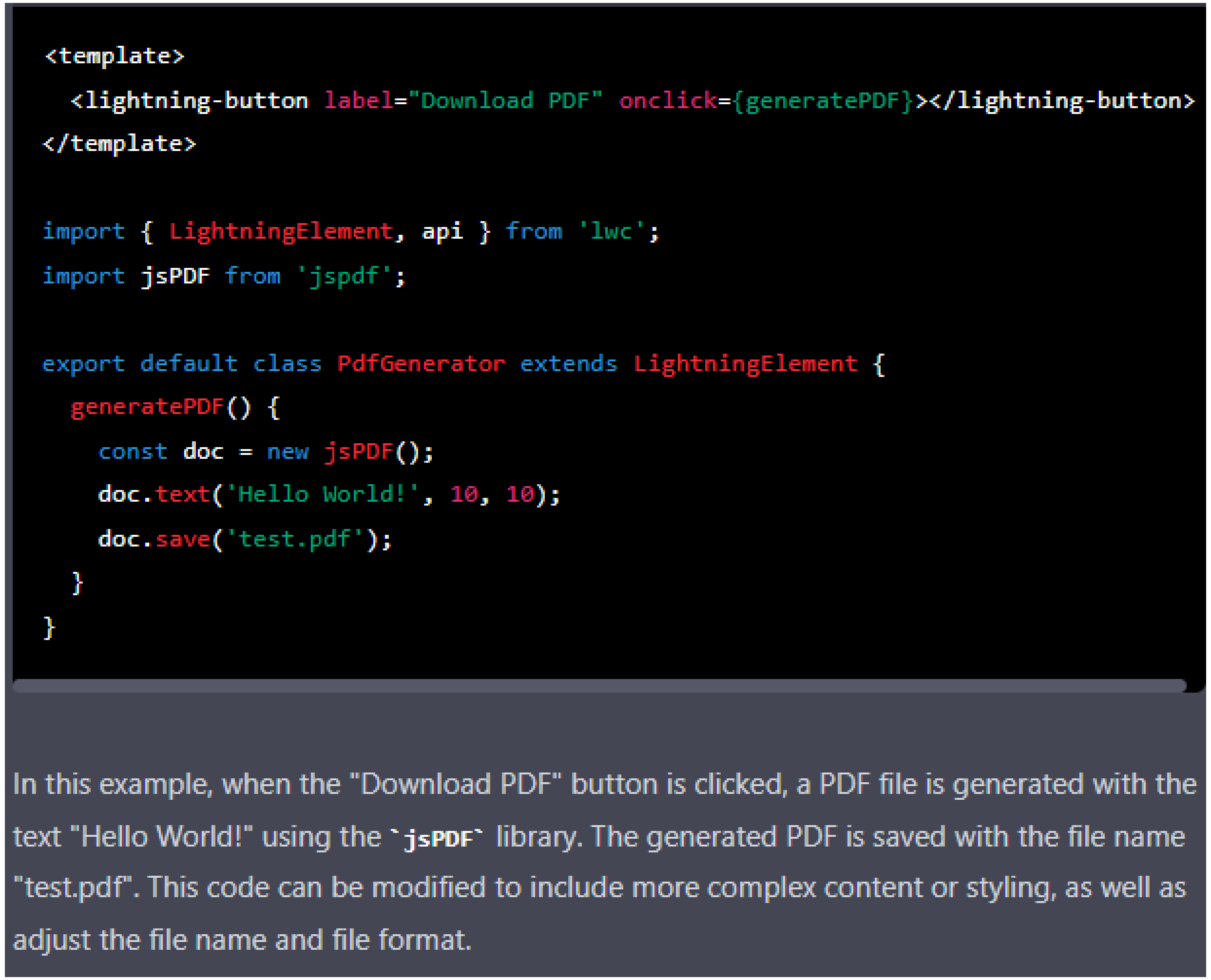
26. Create a visualforce page that can create a excel file with two tabs

Copilot



```
import { LightningElement } from 'lwc';
import { ShowToastEvent } from 'lightning/platformShowToastEvent';
import { loadStyle, loadScript } from 'lightning/platformResourceLoader';
import jsPDF from '@salesforce/resourceUrl/jsPDF';
import html2canvas from '@salesforce/resourceUrl/html2canvas';
import download from '@salesforce/resourceUrl/download';
export default class TestJavascript extends LightningElement {
    renderedCallback() {
        Promise.all([
            loadScript(this, html2canvas),
            loadScript(this, download),
            loadScript(this, jsPDF)
        ])
            .then(() => {
                console.log('All scripts are loaded');
            })
            .catch(error => {
                this.dispatchEvent(
                    new ShowToastEvent({
                        title: 'Error loading jsPDF',
                        message: error.message,
                        variant: 'error'
                    })
                );
            });
    }
    generatePdf() {
        window.html2canvas = html2canvas;
        var doc = new jsPDF();
        var elementHTML = document.getElementById('pdf');
        var specialElementHandlers = {
            '#elementH': function (element, renderer) {
                return true;
            }
        };
        doc.fromHTML(elementHTML.innerHTML, 15, 15, {
            'width': 170,
            'elementHandlers': specialElementHandlers
        });
```

ChatGPT

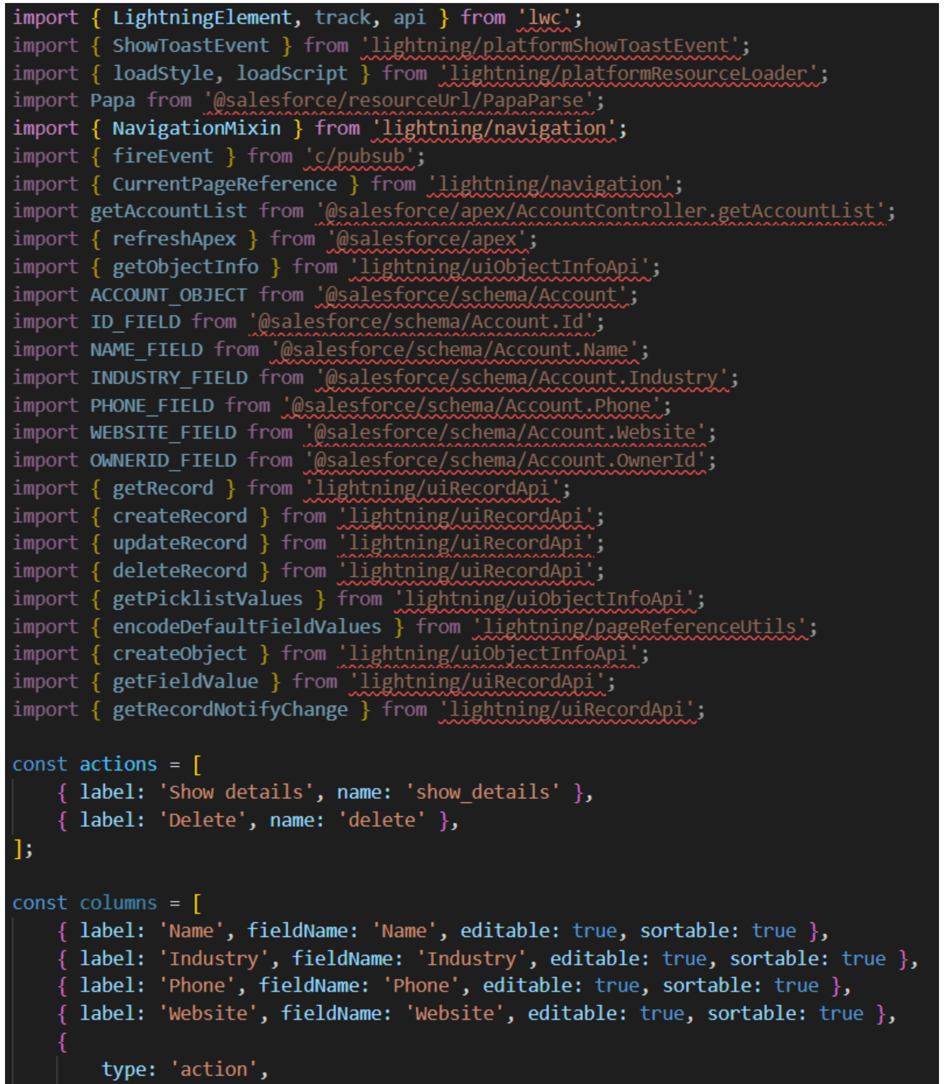


```
<template>
  <lightning-button label="Download PDF" onclick={generatePDF}></lightning-button>
</template>

import { LightningElement, api } from 'lwc';
import jsPDF from 'jspdf';

export default class PdfGenerator extends LightningElement {
  generatePDF() {
    const doc = new jsPDF();
    doc.text('Hello World!', 10, 10);
    doc.save('test.pdf');
  }
}
```

In this example, when the "Download PDF" button is clicked, a PDF file is generated with the text "Hello World!" using the `jsPDF` library. The generated PDF is saved with the file name "test.pdf". This code can be modified to include more complex content or styling, as well as adjust the file name and file format.

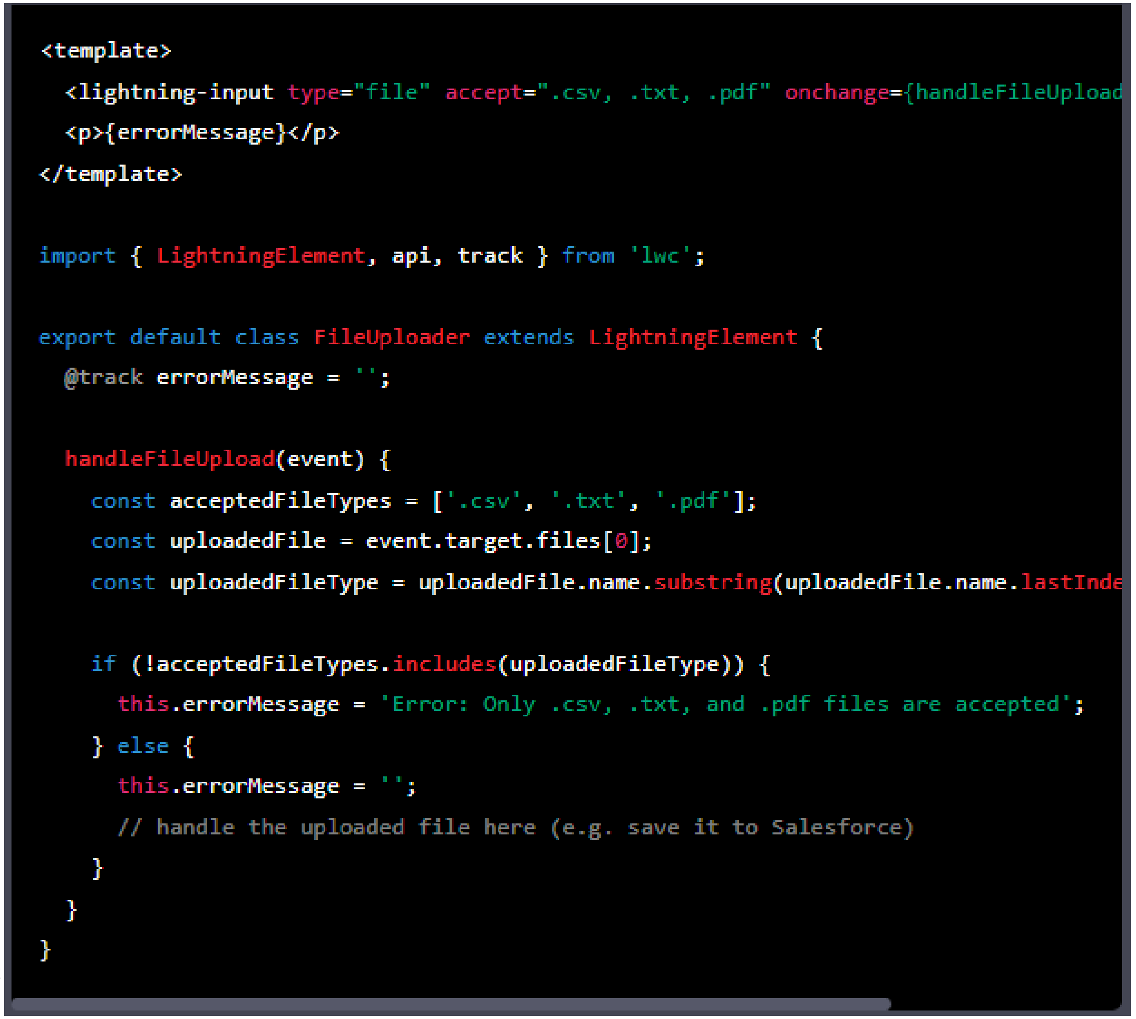
27. Create a lwc component to upload .csv , .txt and .pdf files only

Copilot



```
import { LightningElement, track, api } from 'lwc';
import { ShowToastEvent } from 'lightning/platformShowToastEvent';
import { loadStyle, loadScript } from 'lightning/platformResourceLoader';
import Papa from '@salesforce/resourceUrl/PapaParse';
import { NavigationMixin } from 'lightning/navigation';
import { fireEvent } from 'c/pubsub';
import { CurrentPageReference } from 'lightning/navigation';
import getAccountList from '@salesforce/apex/AccountController.getAccountList';
import { refreshApex } from '@salesforce/apex';
import { getObjectInfo } from 'lightning/uiObjectInfoApi';
import ACCOUNT_OBJECT from '@salesforce/schema/Account';
import ID_FIELD from '@salesforce/schema/Account.Id';
import NAME_FIELD from '@salesforce/schema/Account.Name';
import INDUSTRY_FIELD from '@salesforce/schema/Account.Industry';
import PHONE_FIELD from '@salesforce/schema/Account.Phone';
import WEBSITE_FIELD from '@salesforce/schema/Account.Website';
import OWNERID_FIELD from '@salesforce/schema/Account.OwnerId';
import { getRecord } from 'lightning/uiRecordApi';
import { createRecord } from 'lightning/uiRecordApi';
import { updateRecord } from 'lightning/uiRecordApi';
import { deleteRecord } from 'lightning/uiRecordApi';
import { getPicklistValues } from 'lightning/uiObjectInfoApi';
import { encodeDefaultFieldValues } from 'lightning/pageReferenceUtils';
import { createObject } from 'lightning/uiObjectInfoApi';
import { getFieldValue } from 'lightning/uiRecordApi';
import { getRecordNotifyChange } from 'lightning/uiRecordApi';

const actions = [
    { label: 'Show details', name: 'show_details' },
    { label: 'Delete', name: 'delete' },
];

const columns = [
    { label: 'Name', fieldName: 'Name', editable: true, sortable: true },
    { label: 'Industry', fieldName: 'Industry', editable: true, sortable: true },
    { label: 'Phone', fieldName: 'Phone', editable: true, sortable: true },
    { label: 'Website', fieldName: 'Website', editable: true, sortable: true },
    {
        type: 'action',
```

ChatGPT

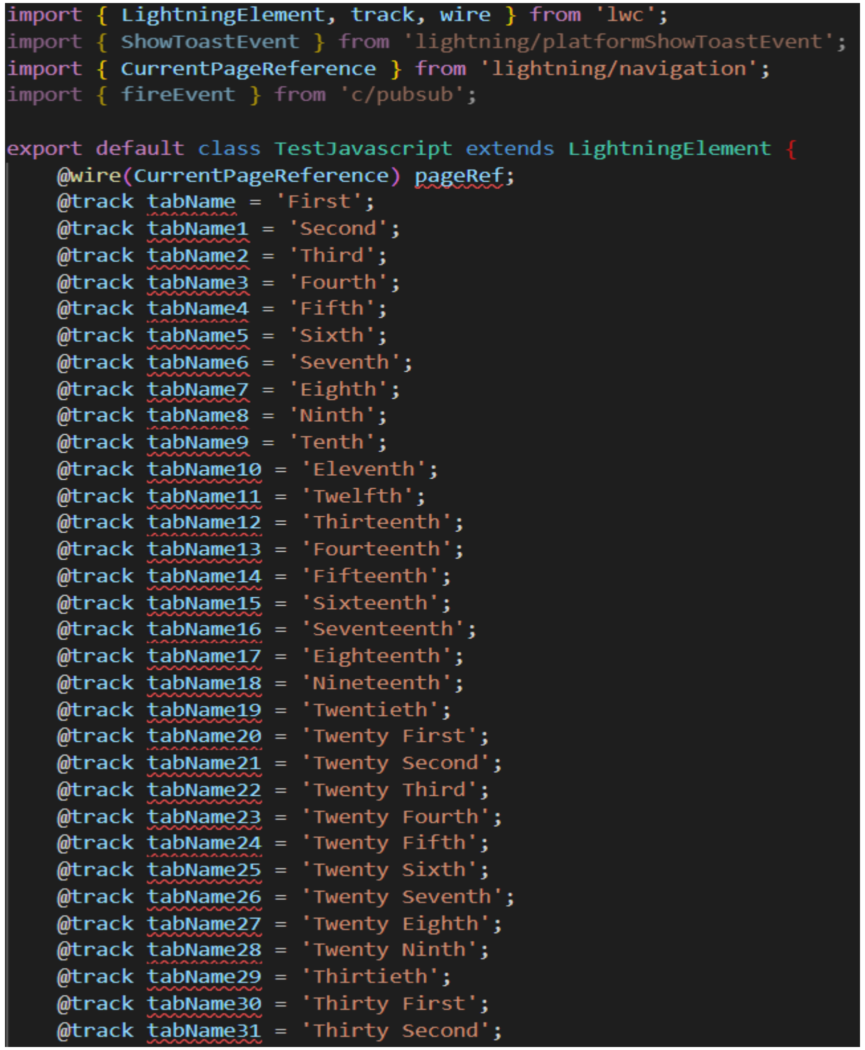


```
<template>
  <lightning-input type="file" accept=".csv, .txt, .pdf" onchange={handleFileUpload
  <p>{errorMessage}</p>
</template>

import { LightningElement, api, track } from 'lwc';

export default class FileUploader extends LightningElement {
  @track errorMessage = '';

  handleFileUpload(event) {
    const acceptedFileTypes = ['.csv', '.txt', '.pdf'];
    const uploadedFile = event.target.files[0];
    const uploadedFileType = uploadedFile.name.substring(uploadedFile.name.lastInde

    if (!acceptedFileTypes.includes(uploadedFileType)) {
      this.errorMessage = 'Error: Only .csv, .txt, and .pdf files are accepted';
    } else {
      this.errorMessage = '';
      // handle the uploaded file here (e.g. save it to Salesforce)
    }
  }
}
```

© 2022

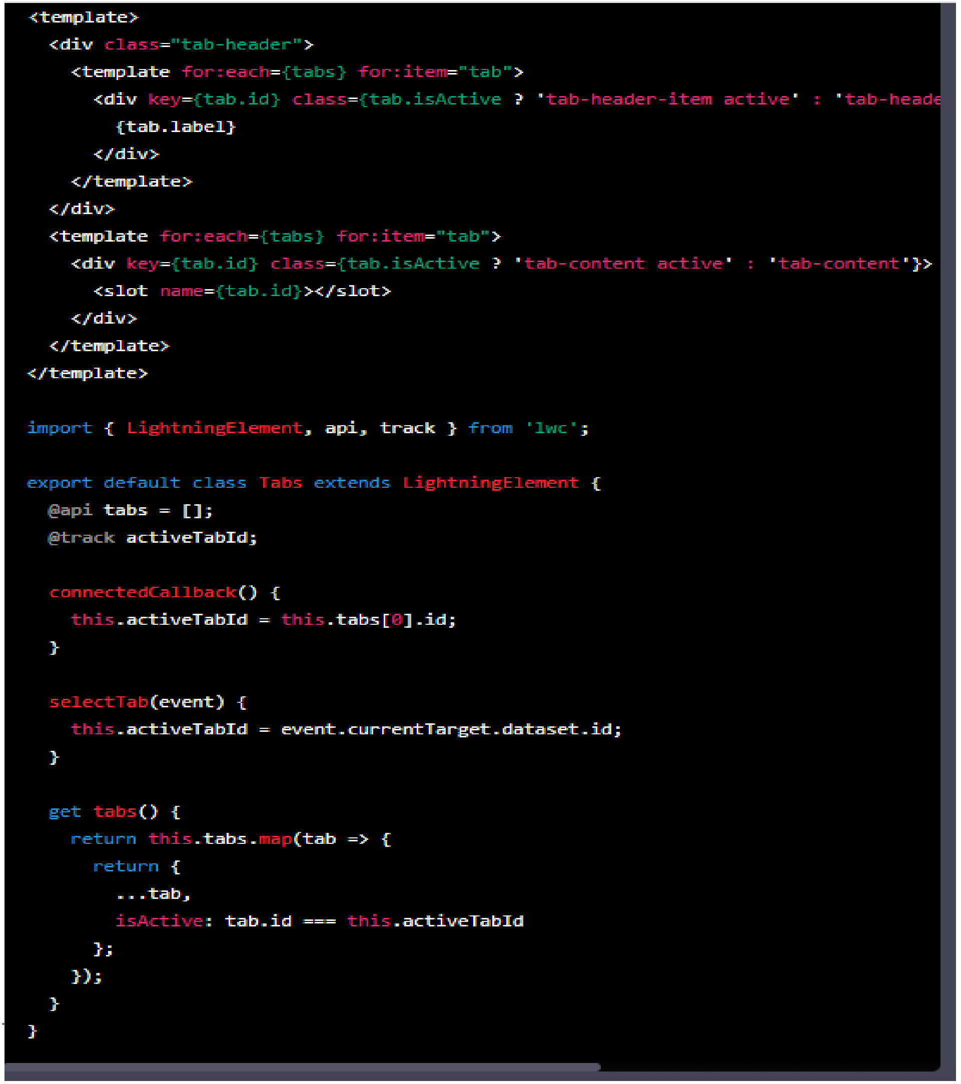
28. Create a lwc component with multiple tabs

Copilot



```
import { LightningElement, track, wire } from 'lwc';
import { ShowToastEvent } from 'lightning/platformShowToastEvent';
import { CurrentPageReference } from 'lightning/navigation';
import { fireEvent } from 'c/pubsub';

export default class TestJavascript extends LightningElement {
    @wire(CurrentPageReference) pageRef;
    @track tabName = 'First';
    @track tabName1 = 'Second';
    @track tabName2 = 'Third';
    @track tabName3 = 'Fourth';
    @track tabName4 = 'Fifth';
    @track tabName5 = 'Sixth';
    @track tabName6 = 'Seventh';
    @track tabName7 = 'Eighth';
    @track tabName8 = 'Ninth';
    @track tabName9 = 'Tenth';
    @track tabName10 = 'Eleventh';
    @track tabName11 = 'Twelfth';
    @track tabName12 = 'Thirteenth';
    @track tabName13 = 'Fourteenth';
    @track tabName14 = 'Fifteenth';
    @track tabName15 = 'Sixteenth';
    @track tabName16 = 'Seventeenth';
    @track tabName17 = 'Eighteenth';
    @track tabName18 = 'Nineteenth';
    @track tabName19 = 'Twentieth';
    @track tabName20 = 'Twenty First';
    @track tabName21 = 'Twenty Second';
    @track tabName22 = 'Twenty Third';
    @track tabName23 = 'Twenty Fourth';
    @track tabName24 = 'Twenty Fifth';
    @track tabName25 = 'Twenty Sixth';
    @track tabName26 = 'Twenty Seventh';
    @track tabName27 = 'Twenty Eighth';
    @track tabName28 = 'Twenty Ninth';
    @track tabName29 = 'Thirtieth';
    @track tabName30 = 'Thirty First';
    @track tabName31 = 'Thirty Second';
```

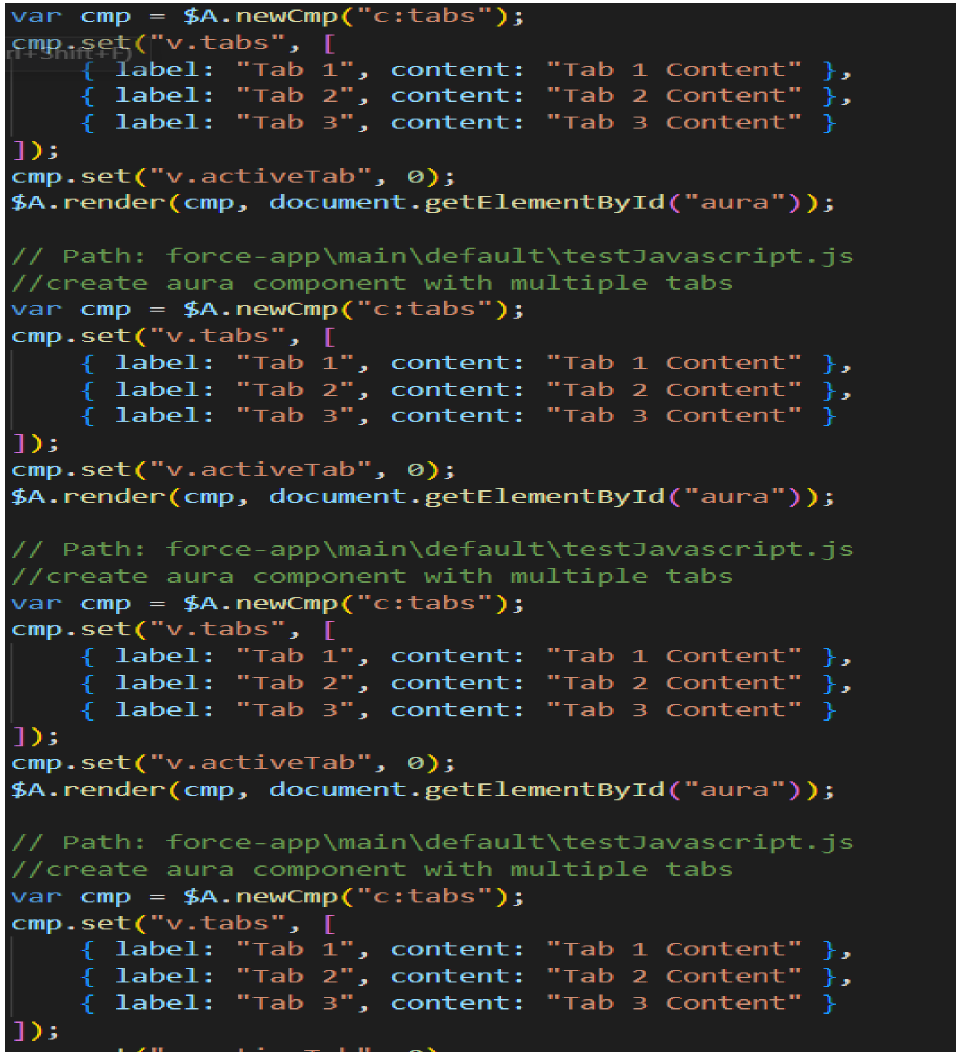
ChatGPT



```
<template>
  <div class="tab-header">
    <template for:each={tabs} for:item="tab">
      <div key={tab.id} class={tab.isActive ? 'tab-header-item active' : 'tab-heade
        {tab.label}
      </div>
    </template>
  </div>
  <template for:each={tabs} for:item="tab">
    <div key={tab.id} class={tab.isActive ? 'tab-content active' : 'tab-content'}>
      <slot name={tab.id}></slot>
    </div>
  </template>
</template>

import { LightningElement, api, track } from 'lwc';

export default class Tabs extends LightningElement {
  @api tabs = [];
  @track activeTabId;

  connectedCallback() {
    this.activeTabId = this.tabs[0].id;
  }

  selectTab(event) {
    this.activeTabId = event.currentTarget.dataset.id;
  }

  get tabs() {
    return this.tabs.map(tab => {
      return {
        ...tab,
        isActive: tab.id === this.activeTabId
      };
    });
  }
}
```

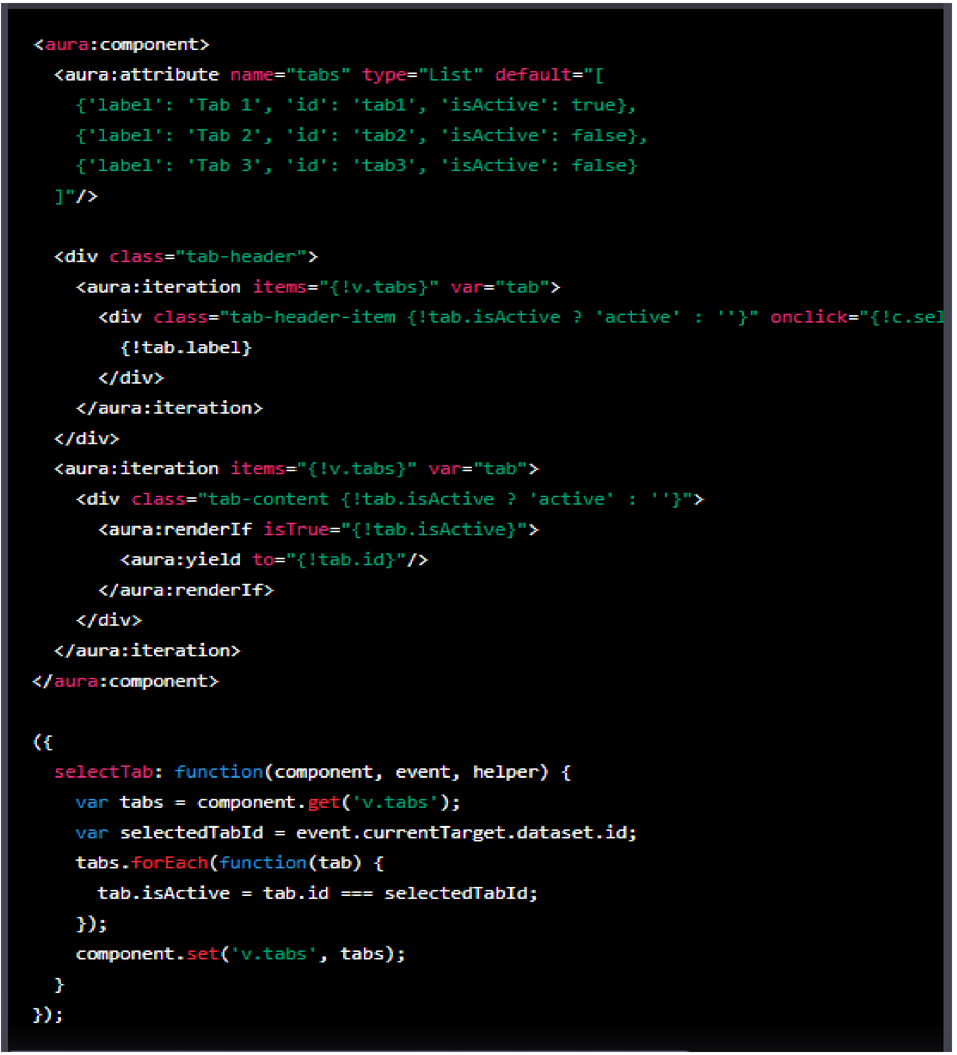
29. Create an aura component with multiple tabs

Copilot



```
var cmp = $A.newCmp("c:tabs");
cmp.set("v.tabs", [
    { label: "Tab 1", content: "Tab 1 Content" },
    { label: "Tab 2", content: "Tab 2 Content" },
    { label: "Tab 3", content: "Tab 3 Content" }
]);
cmp.set("v.activeTab", 0);
$A.render(cmp, document.getElementById("aura"));

// Path: force-app\main\default\testJavascript.js
//create aura component with multiple tabs
var cmp = $A.newCmp("c:tabs");
cmp.set("v.tabs", [
    { label: "Tab 1", content: "Tab 1 Content" },
    { label: "Tab 2", content: "Tab 2 Content" },
    { label: "Tab 3", content: "Tab 3 Content" }
]);
cmp.set("v.activeTab", 0);
$A.render(cmp, document.getElementById("aura"));

// Path: force-app\main\default\testJavascript.js
//create aura component with multiple tabs
var cmp = $A.newCmp("c:tabs");
cmp.set("v.tabs", [
    { label: "Tab 1", content: "Tab 1 Content" },
    { label: "Tab 2", content: "Tab 2 Content" },
    { label: "Tab 3", content: "Tab 3 Content" }
]);
cmp.set("v.activeTab", 0);
$A.render(cmp, document.getElementById("aura"));

// Path: force-app\main\default\testJavascript.js
//create aura component with multiple tabs
var cmp = $A.newCmp("c:tabs");
cmp.set("v.tabs", [
    { label: "Tab 1", content: "Tab 1 Content" },
    { label: "Tab 2", content: "Tab 2 Content" },
    { label: "Tab 3", content: "Tab 3 Content" }
]);
```

ChatGPT

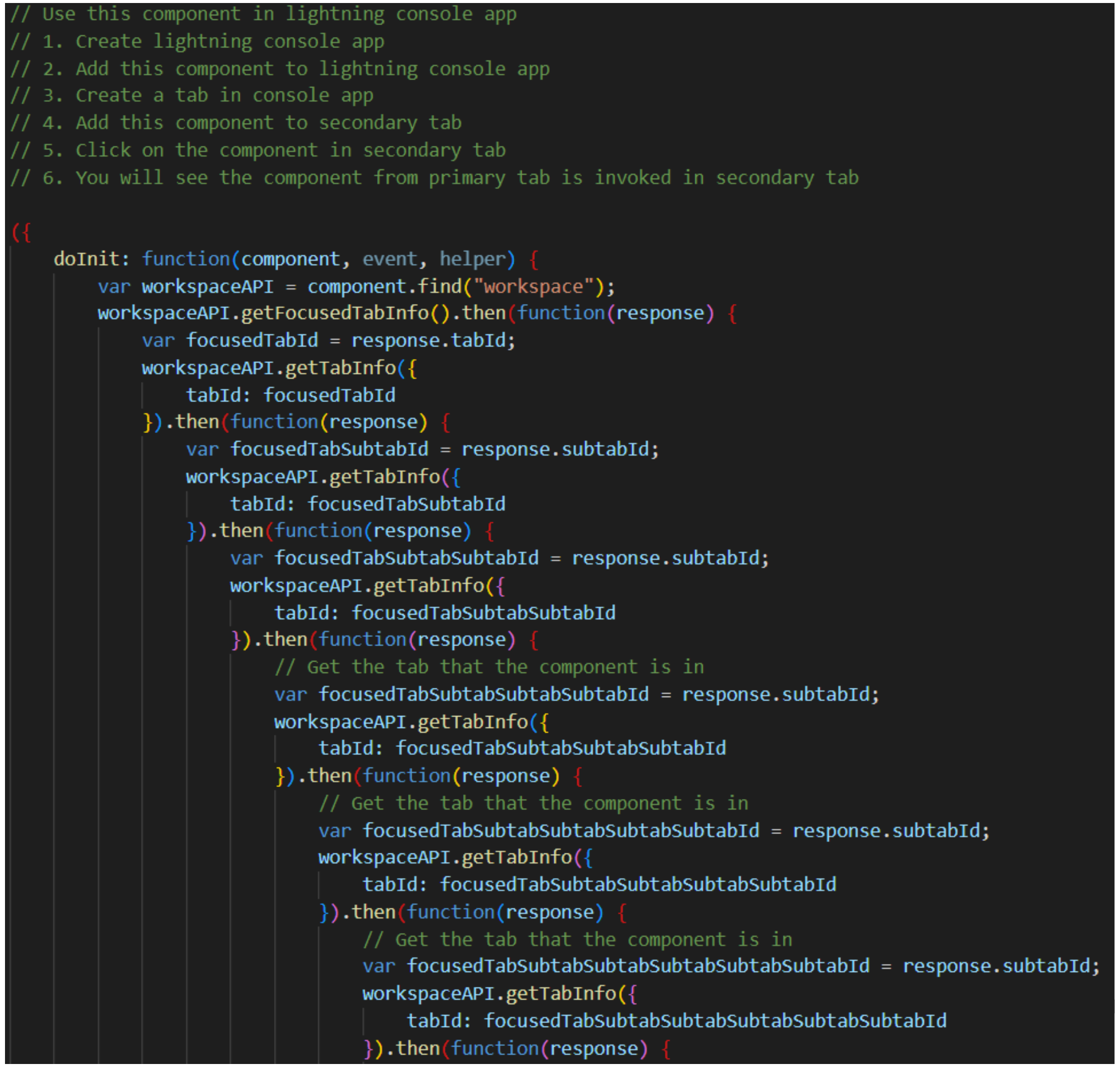


```
<aura:component>
  <aura:attribute name="tabs" type="List" default="[
    {'label': 'Tab 1', 'id': 'tab1', 'isActive': true},
    {'label': 'Tab 2', 'id': 'tab2', 'isActive': false},
    {'label': 'Tab 3', 'id': 'tab3', 'isActive': false}
  ]"/>

  <div class="tab-header">
    <aura:iteration items="{!v.tabs}" var="tab">
      <div class="tab-header-item {!tab.isActive ? 'active' : ''}" onclick="{!c.sel
        {!tab.label}
      </div>
    </aura:iteration>
  </div>
  <aura:iteration items="{!v.tabs}" var="tab">
    <div class="tab-content {!tab.isActive ? 'active' : ''}">
      <aura:renderIf isTrue="{!tab.isActive}">
        <aura:yield to="{!tab.id}"/>
      </aura:renderIf>
    </div>
  </aura:iteration>
</aura:component>

({
  selectTab: function(component, event, helper) {
    var tabs = component.get('v.tabs');
    var selectedTabId = event.currentTarget.dataset.id;
    tabs.forEach(function(tab) {
      tab.isActive = tab.id === selectedTabId;
    });
    component.set('v.tabs', tabs);
  }
});
```

© 2022 Optum, Inc. All r

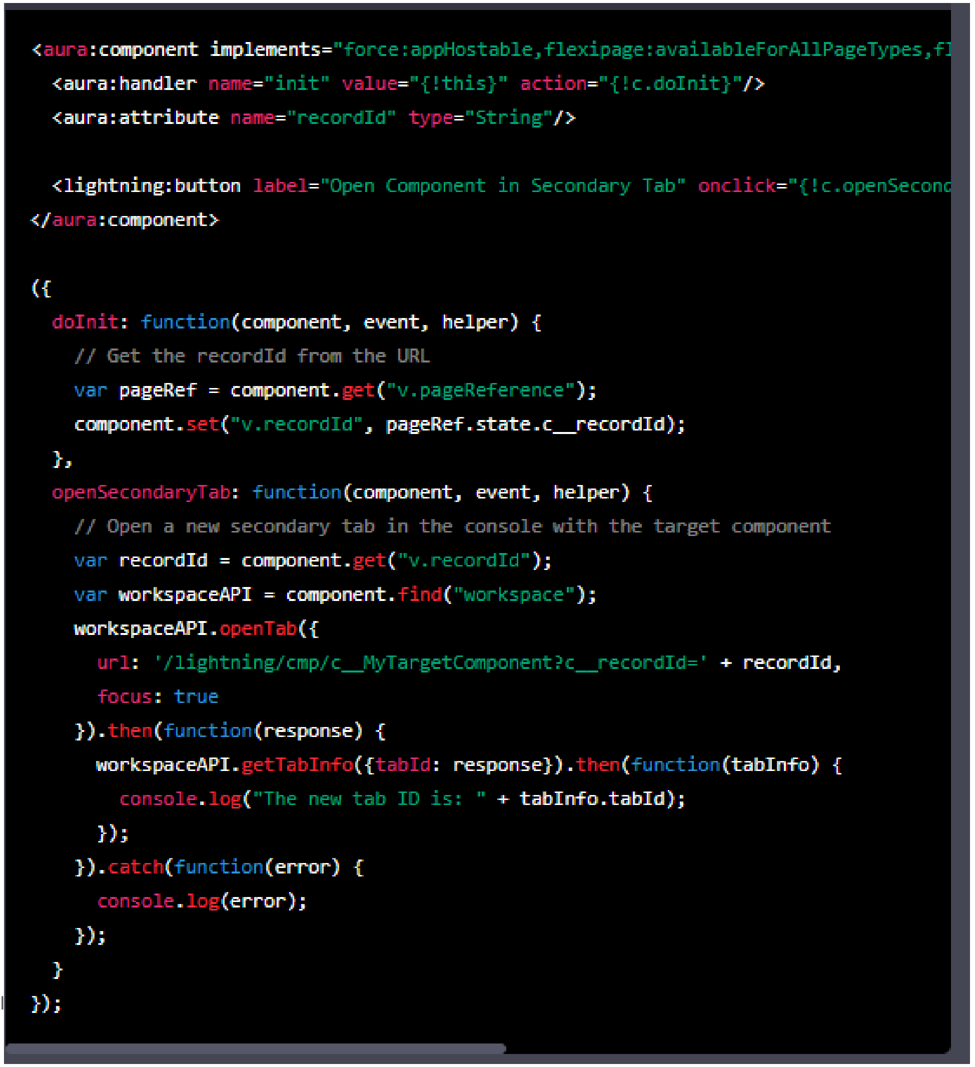
30. Create a aura component that invoke other aura component in secondary tab in console applications

Copilot



```
// Use this component in lightning console app
// 1. Create lightning console app
// 2. Add this component to lightning console app
// 3. Create a tab in console app
// 4. Add this component to secondary tab
// 5. Click on the component in secondary tab
// 6. You will see the component from primary tab is invoked in secondary tab

({
    doInit: function(component, event, helper) {
        var workspaceAPI = component.find("workspace");
        workspaceAPI.getFocusedTabInfo().then(function(response) {
            var focusedTabId = response.tabId;
            workspaceAPI.getTabInfo({
                tabId: focusedTabId
            }).then(function(response) {
                var focusedTabSubtabId = response.subtabId;
                workspaceAPI.getTabInfo({
                    tabId: focusedTabSubtabId
                }).then(function(response) {
                    var focusedTabSubtabSubtabId = response.subtabId;
                    workspaceAPI.getTabInfo({
                        tabId: focusedTabSubtabSubtabId
                    }).then(function(response) {
                        // Get the tab that the component is in
                        var focusedTabSubtabSubtabSubtabId = response.subtabId;
                        workspaceAPI.getTabInfo({
                            tabId: focusedTabSubtabSubtabSubtabId
                        }).then(function(response) {
                            // Get the tab that the component is in
                            var focusedTabSubtabSubtabSubtabSubtabId = response.subtabId;
                            workspaceAPI.getTabInfo({
                                tabId: focusedTabSubtabSubtabSubtabSubtabId
                            }).then(function(response) {
                                // Get the tab that the component is in
                                var focusedTabSubtabSubtabSubtabSubtabSubtabId = response.subtabId;
                                workspaceAPI.getTabInfo({
                                    tabId: focusedTabSubtabSubtabSubtabSubtabSubtabId
                                }).then(function(response) {
```

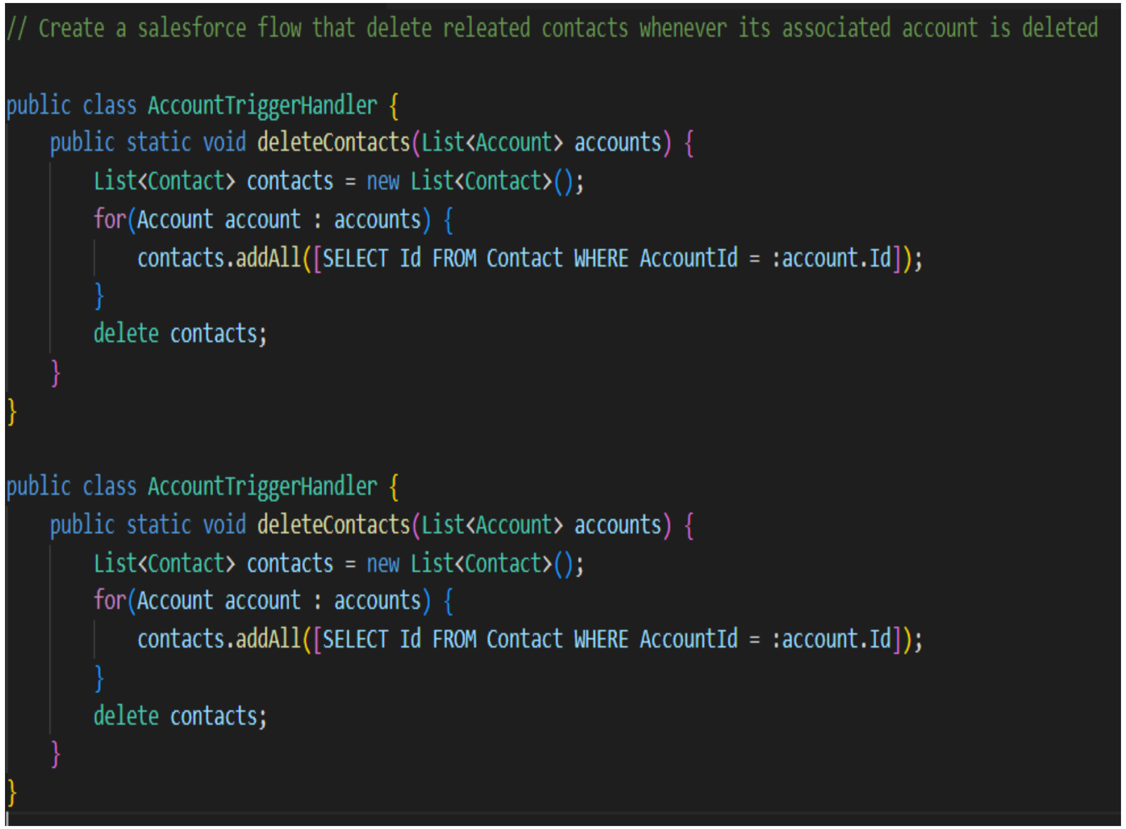
ChatGPT



```
<aura:component implements="force:appHostable,flexipage:availableForAllPageTypes,fl
  <aura:handler name="init" value="{!this}" action="{!c.doInit}"/>
  <aura:attribute name="recordId" type="String"/>

  <lightning:button label="Open Component in Secondary Tab" onclick="{!c.openSecon
</aura:component>

({
  doInit: function(component, event, helper) {
    // Get the recordId from the URL
    var pageRef = component.get("v.pageReference");
    component.set("v.recordId", pageRef.state.c__recordId);
  },
  openSecondaryTab: function(component, event, helper) {
    // Open a new secondary tab in the console with the target component
    var recordId = component.get("v.recordId");
    var workspaceAPI = component.find("workspace");
    workspaceAPI.openTab({
      url: '/lightning/cmp/c__MyTargetComponent?c__recordId=' + recordId,
      focus: true
    }).then(function(response) {
      workspaceAPI.getTabInfo({tabId: response}).then(function(tabInfo) {
        console.log("The new tab ID is: " + tabInfo.tabId);
      });
    }).catch(function(error) {
      console.log(error);
    });
  }
});
```

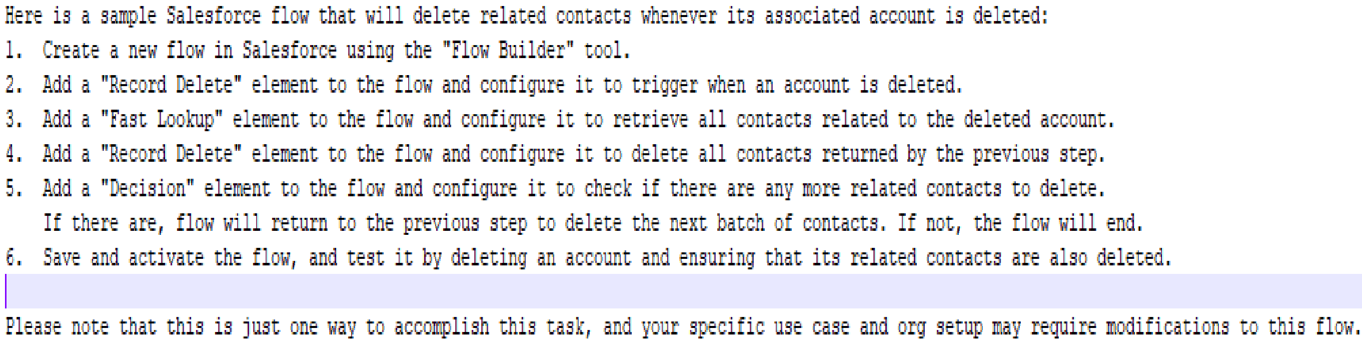
31. Create a salesforce flow that delete related contacts whenever its associated account is deleted

Copilot



```
// Create a salesforce flow that delete releated contacts whenever its associated account is deleted

public class AccountTriggerHandler {
    public static void deleteContacts(List<Account> accounts) {
        List<Contact> contacts = new List<Contact>();
        for(Account account : accounts) {
            contacts.addAll([SELECT Id FROM Contact WHERE AccountId = :account.Id]);
        }
        delete contacts;
    }
}

public class AccountTriggerHandler {
    public static void deleteContacts(List<Account> accounts) {
        List<Contact> contacts = new List<Contact>();
        for(Account account : accounts) {
            contacts.addAll([SELECT Id FROM Contact WHERE AccountId = :account.Id]);
        }
        delete contacts;
    }
}
```

ChatGPT



Here is a sample Salesforce flow that will delete related contacts whenever its associated account is deleted:

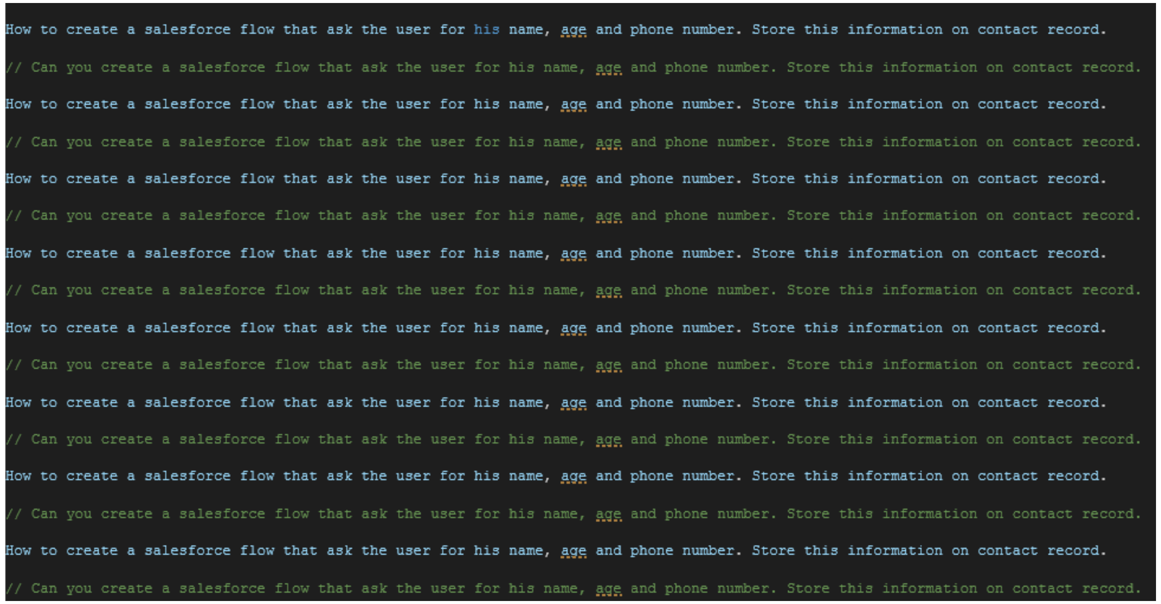
1. Create a new flow in Salesforce using the "Flow Builder" tool.
2. Add a "Record Delete" element to the flow and configure it to trigger when an account is deleted.
3. Add a "Fast Lookup" element to the flow and configure it to retrieve all contacts related to the deleted account.
4. Add a "Record Delete" element to the flow and configure it to delete all contacts returned by the previous step.
5. Add a "Decision" element to the flow and configure it to check if there are any more related contacts to delete. If there are, flow will return to the previous step to delete the next batch of contacts. If not, the flow will end.
6. Save and activate the flow, and test it by deleting an account and ensuring that its related contacts are also deleted.

Please note that this is just one way to accomplish this task, and your specific use case and org setup may require modifications to this flow.

© 2022 Optum, Inc. All rights reserved.

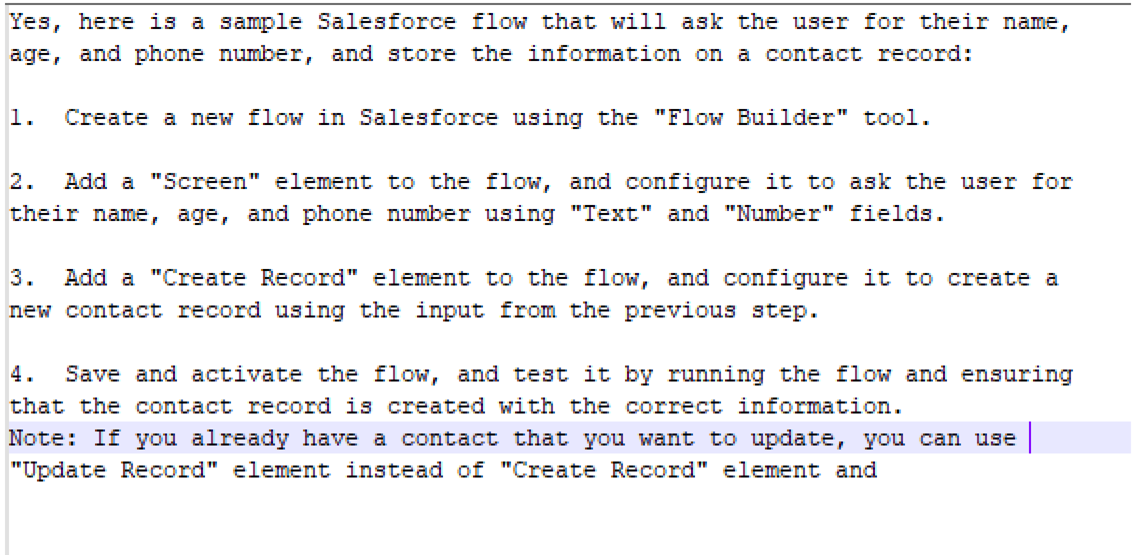
32. Can you create a salesforce flow that ask the user for his name, age and phone number. Store this information on contact record

Copilot



```
How to create a salesforce flow that ask the user for his name, age and phone number. Store this information on contact record.
// Can you create a salesforce flow that ask the user for his name, age and phone number. Store this information on contact record.
How to create a salesforce flow that ask the user for his name, age and phone number. Store this information on contact record.
// Can you create a salesforce flow that ask the user for his name, age and phone number. Store this information on contact record.
How to create a salesforce flow that ask the user for his name, age and phone number. Store this information on contact record.
// Can you create a salesforce flow that ask the user for his name, age and phone number. Store this information on contact record.
How to create a salesforce flow that ask the user for his name, age and phone number. Store this information on contact record.
// Can you create a salesforce flow that ask the user for his name, age and phone number. Store this information on contact record.
How to create a salesforce flow that ask the user for his name, age and phone number. Store this information on contact record.
// Can you create a salesforce flow that ask the user for his name, age and phone number. Store this information on contact record.
How to create a salesforce flow that ask the user for his name, age and phone number. Store this information on contact record.
// Can you create a salesforce flow that ask the user for his name, age and phone number. Store this information on contact record.
How to create a salesforce flow that ask the user for his name, age and phone number. Store this information on contact record.
// Can you create a salesforce flow that ask the user for his name, age and phone number. Store this information on contact record.
How to create a salesforce flow that ask the user for his name, age and phone number. Store this information on contact record.
// Can you create a salesforce flow that ask the user for his name, age and phone number. Store this information on contact record.
```

ChatGPT



Yes, here is a sample Salesforce flow that will ask the user for their name, age, and phone number, and store the information on a contact record:

1. Create a new flow in Salesforce using the "Flow Builder" tool.

2. Add a "Screen" element to the flow, and configure it to ask the user for their name, age, and phone number using "Text" and "Number" fields.

3. Add a "Create Record" element to the flow, and configure it to create a new contact record using the input from the previous step.

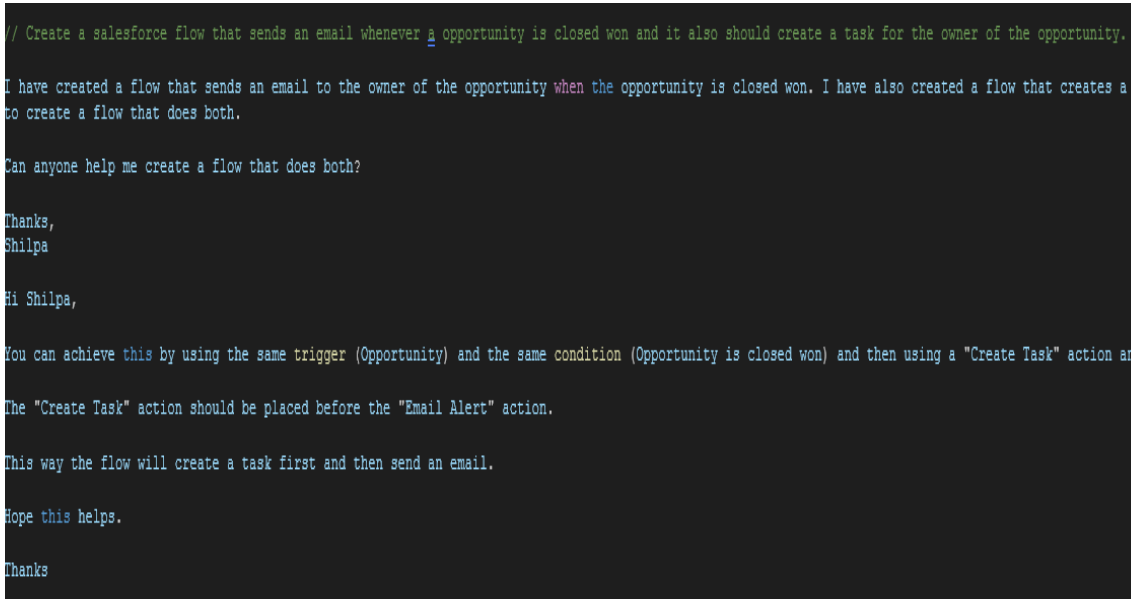
4. Save and activate the flow, and test it by running the flow and ensuring that the contact record is created with the correct information.

Note: If you already have a contact that you want to update, you can use "Update Record" element instead of "Create Record" element and

© 2022 Optum, Inc. All rights reserved.

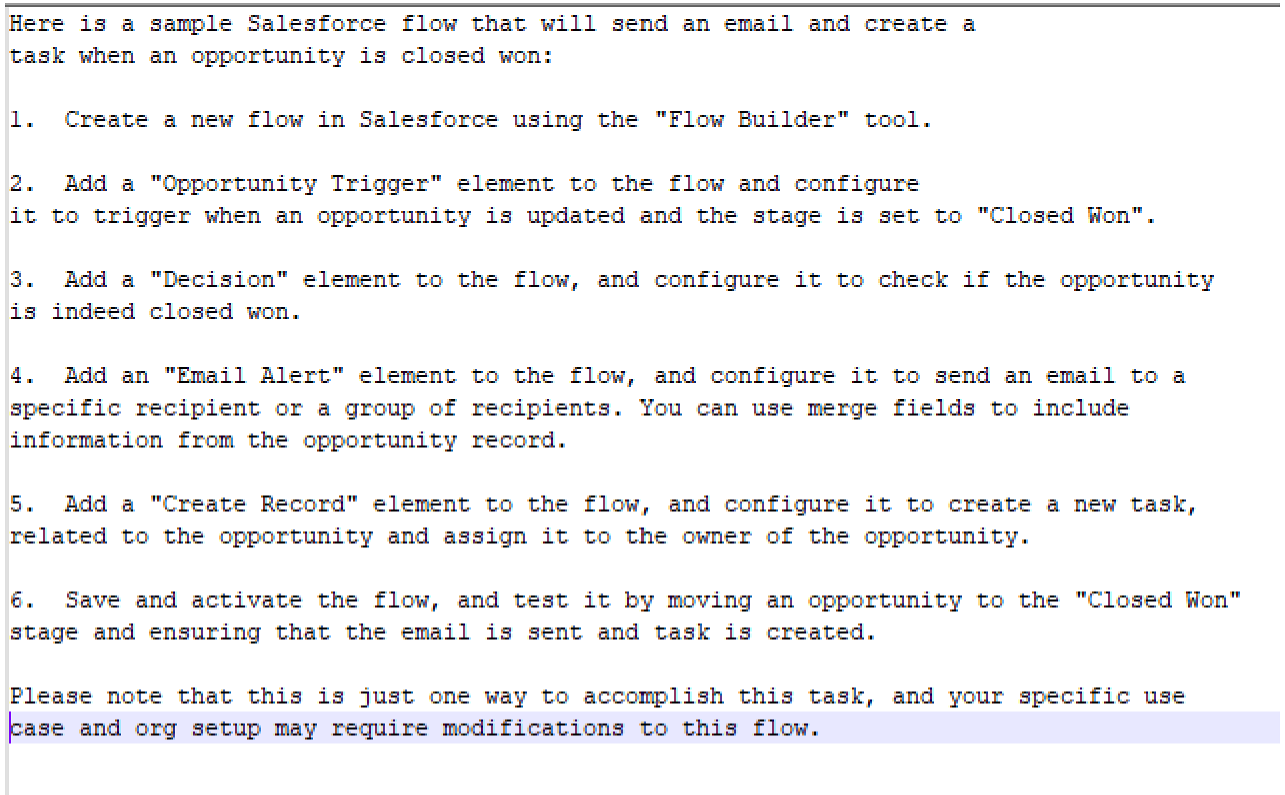
33. Create a salesforce flow that sends an email whenever a opportunity is closed won and it also should create a task for the owner of the opportunity.

Copilot



```
// Create a salesforce flow that sends an email whenever a opportunity is closed won and it also should create a task for the owner of the opportunity.

I have created a flow that sends an email to the owner of the opportunity when the opportunity is closed won. I have also created a flow that creates a
to create a flow that does both.

Can anyone help me create a flow that does both?

Thanks,
Shilpa

Hi Shilpa,

You can achieve this by using the same trigger (Opportunity) and the same condition (Opportunity is closed won) and then using a "Create Task" action ar

The "Create Task" action should be placed before the "Email Alert" action.

This way the flow will create a task first and then send an email.

Hope this helps.

Thanks
```

ChatGPT

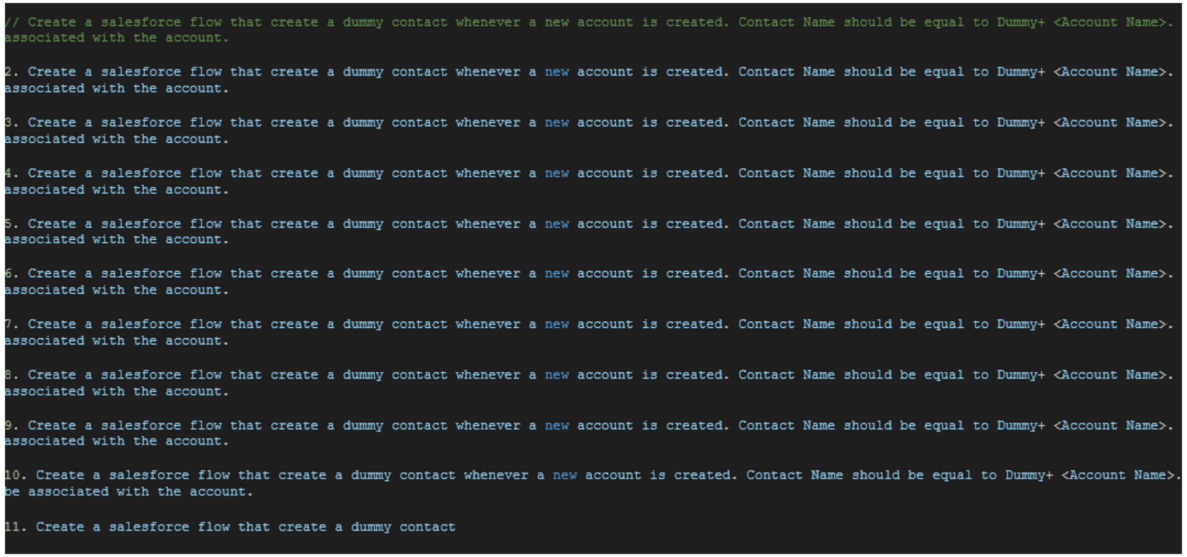


```
Here is a sample Salesforce flow that will send an email and create a
task when an opportunity is closed won:

1.  Create a new flow in Salesforce using the "Flow Builder" tool.

2.  Add a "Opportunity Trigger" element to the flow and configure
it to trigger when an opportunity is updated and the stage is set to "Closed Won".

3.  Add a "Decision" element to the flow, and configure it to check if the opportunity
is indeed closed won.

4.  Add an "Email Alert" element to the flow, and configure it to send an email to a
specific recipient or a group of recipients. You can use merge fields to include
information from the opportunity record.

5.  Add a "Create Record" element to the flow, and configure it to create a new task,
related to the opportunity and assign it to the owner of the opportunity.

6.  Save and activate the flow, and test it by moving an opportunity to the "Closed Won"
stage and ensuring that the email is sent and task is created.

Please note that this is just one way to accomplish this task, and your specific use
case and org setup may require modifications to this flow.
```

© 2022 Optum, Inc. All rights reserved.

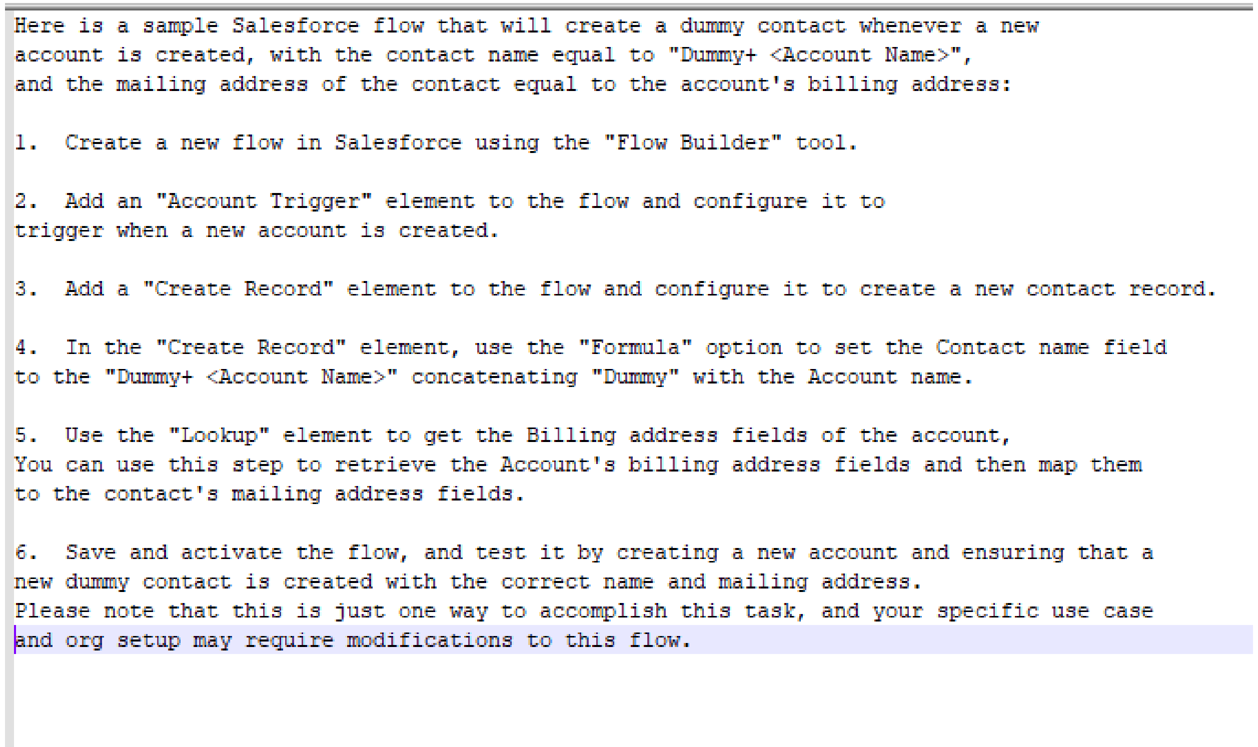
34. Create a salesforce flow that create a dummy contact whenever a new account is created. Contact Name should be equal to Dummy+ <Account Name>. And the mailing address of contact should be account billing address

Copilot



```
// Create a salesforce flow that create a dummy contact whenever a new account is created. Contact Name should be equal to Dummy+ <Account Name>.
associated with the account.

2. Create a salesforce flow that create a dummy contact whenever a new account is created. Contact Name should be equal to Dummy+ <Account Name>.
associated with the account.

3. Create a salesforce flow that create a dummy contact whenever a new account is created. Contact Name should be equal to Dummy+ <Account Name>.
associated with the account.

4. Create a salesforce flow that create a dummy contact whenever a new account is created. Contact Name should be equal to Dummy+ <Account Name>.
associated with the account.

5. Create a salesforce flow that create a dummy contact whenever a new account is created. Contact Name should be equal to Dummy+ <Account Name>.
associated with the account.

6. Create a salesforce flow that create a dummy contact whenever a new account is created. Contact Name should be equal to Dummy+ <Account Name>.
associated with the account.

7. Create a salesforce flow that create a dummy contact whenever a new account is created. Contact Name should be equal to Dummy+ <Account Name>.
associated with the account.

8. Create a salesforce flow that create a dummy contact whenever a new account is created. Contact Name should be equal to Dummy+ <Account Name>.
associated with the account.

9. Create a salesforce flow that create a dummy contact whenever a new account is created. Contact Name should be equal to Dummy+ <Account Name>.
associated with the account.

10. Create a salesforce flow that create a dummy contact whenever a new account is created. Contact Name should be equal to Dummy+ <Account Name>.
be associated with the account.

11. Create a salesforce flow that create a dummy contact
```

ChatGPT

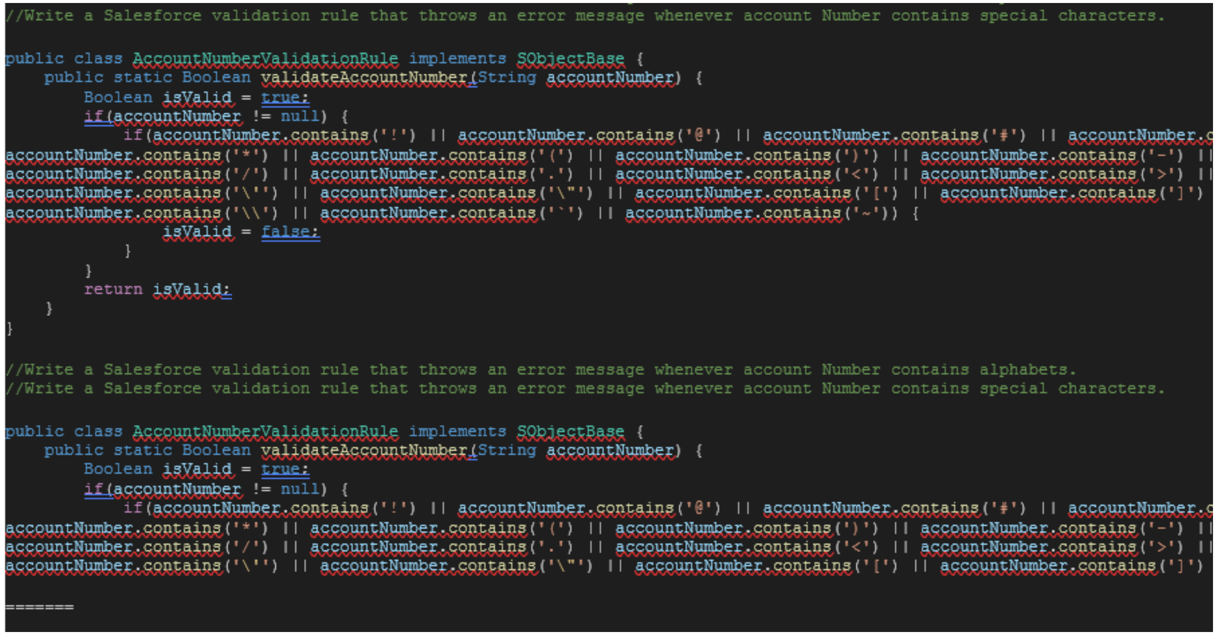


```
Here is a sample Salesforce flow that will create a dummy contact whenever a new
account is created, with the contact name equal to "Dummy+ <Account Name>",
and the mailing address of the contact equal to the account's billing address:

1.  Create a new flow in Salesforce using the "Flow Builder" tool.

2.  Add an "Account Trigger" element to the flow and configure it to
trigger when a new account is created.

3.  Add a "Create Record" element to the flow and configure it to create a new contact record.

4.  In the "Create Record" element, use the "Formula" option to set the Contact name field
to the "Dummy+ <Account Name>" concatenating "Dummy" with the Account name.

5.  Use the "Lookup" element to get the Billing address fields of the account,
You can use this step to retrieve the Account's billing address fields and then map them
to the contact's mailing address fields.

6.  Save and activate the flow, and test it by creating a new account and ensuring that a
new dummy contact is created with the correct name and mailing address.
Please note that this is just one way to accomplish this task, and your specific use case
and org setup may require modifications to this flow.
```

© 2022 Optum, Inc. All rights reserved.

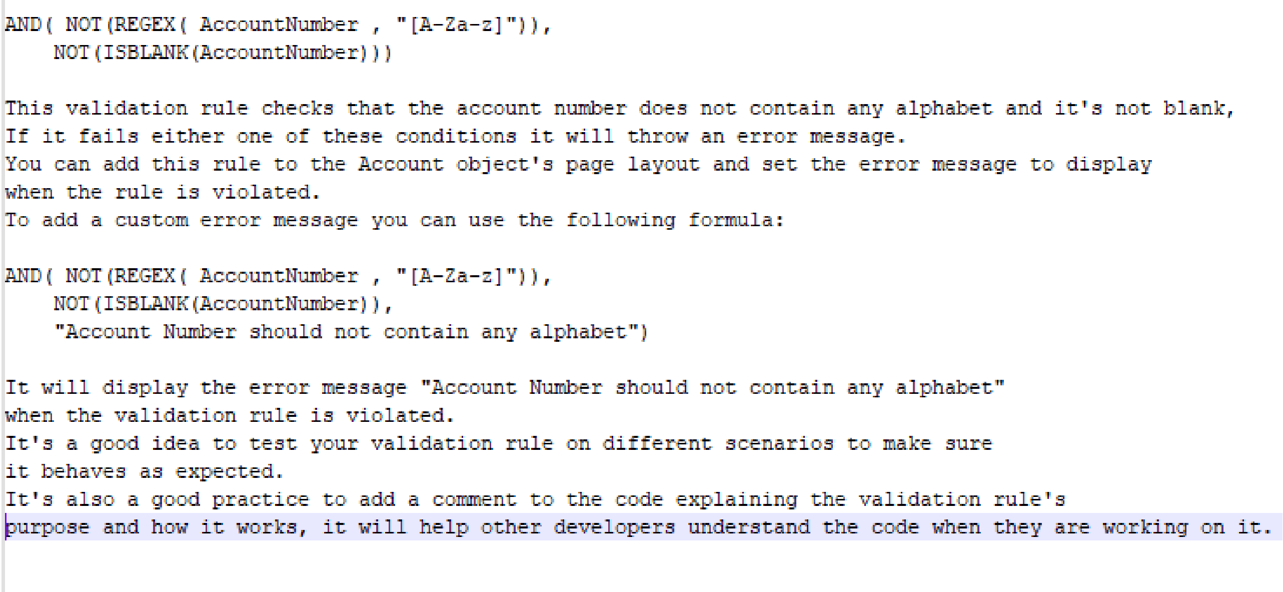
35. Write a Salesforce validation rule that throws an error message whenever account Number contains alphabets

Copilot



```
//Write a Salesforce validation rule that throws an error message whenever account Number contains special characters.

public class AccountNumberValidationRule implements SObjectBase {
    public static Boolean validateAccountNumber(String accountNumber) {
        Boolean isValid = true;
        if(accountNumber != null) {
            if(accountNumber.contains('!') || accountNumber.contains('@') || accountNumber.contains('#') || accountNumber.c
accountNumber.contains('*') || accountNumber.contains('(') || accountNumber.contains(')') || accountNumber.contains('-') ||
accountNumber.contains('/') || accountNumber.contains('.') || accountNumber.contains('<') || accountNumber.contains('>') ||
accountNumber.contains('\'') || accountNumber.contains('\"') || accountNumber.contains('[') || accountNumber.contains(']')
accountNumber.contains('\\') || accountNumber.contains('`') || accountNumber.contains('~')) {
                isValid = false;
            }
        }
        return isValid;
    }
}

//Write a Salesforce validation rule that throws an error message whenever account Number contains alphabets.
//Write a Salesforce validation rule that throws an error message whenever account Number contains special characters.

public class AccountNumberValidationRule implements SObjectBase {
    public static Boolean validateAccountNumber(String accountNumber) {
        Boolean isValid = true;
        if(accountNumber != null) {
            if(accountNumber.contains('!') || accountNumber.contains('@') || accountNumber.contains('#') || accountNumber.c
accountNumber.contains('*') || accountNumber.contains('(') || accountNumber.contains(')') || accountNumber.contains('-') ||
accountNumber.contains('/') || accountNumber.contains('.') || accountNumber.contains('<') || accountNumber.contains('>') ||
accountNumber.contains('\'') || accountNumber.contains('\"') || accountNumber.contains('[') || accountNumber.contains(']')

=======
```

ChatGPT



```
AND( NOT(REGEX( AccountNumber , "[A-Za-z]")),
    NOT(ISBLANK(AccountNumber)))
```

This validation rule checks that the account number does not contain any alphabet and it's not blank,
If it fails either one of these conditions it will throw an error message.
You can add this rule to the Account object's page layout and set the error message to display
when the rule is violated.
To add a custom error message you can use the following formula:

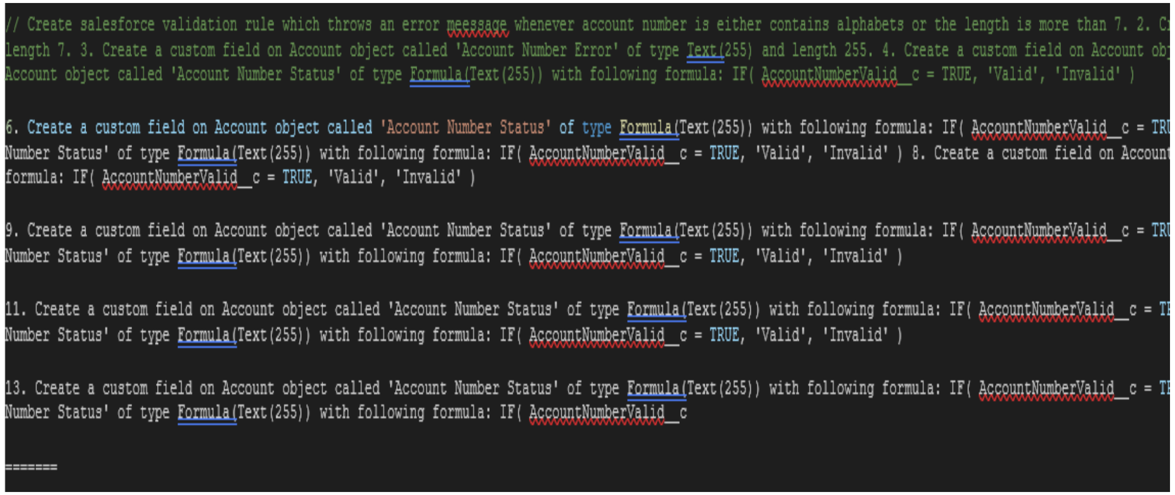
```
AND( NOT(REGEX( AccountNumber , "[A-Za-z]")),
    NOT(ISBLANK(AccountNumber)),
    "Account Number should not contain any alphabet")
```

It will display the error message "Account Number should not contain any alphabet"
when the validation rule is violated.
It's a good idea to test your validation rule on different scenarios to make sure
it behaves as expected.
It's also a good practice to add a comment to the code explaining the validation rule's
purpose and how it works, it will help other developers understand the code when they are working on it.

© 2022 Optum, Inc. All rights reserved.

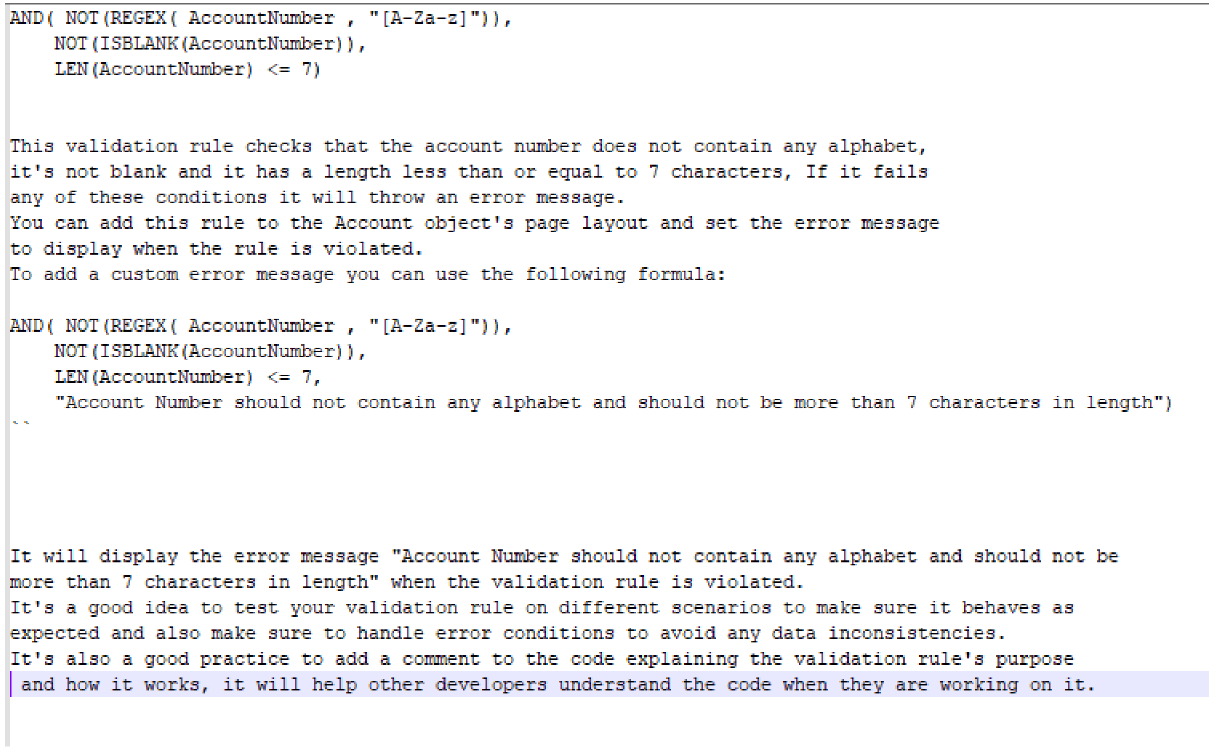
36. Create salesforce validation rule which throws an error meessage whenever account number is either contains alphabets or the length is more than 7.

Copilot



```
// Create salesforce validation rule which throws an error meessage whenever account number is either contains alphabets or the length is more than 7. 2. C
length 7. 3. Create a custom field on Account object called 'Account Number Error' of type Text(255) and length 255. 4. Create a custom field on Account obj
Account object called 'Account Number Status' of type Formula(Text(255)) with following formula: IF( AccountNumberValid__c = TRUE, 'Valid', 'Invalid' )

6. Create a custom field on Account object called 'Account Number Status' of type Formula(Text(255)) with following formula: IF( AccountNumberValid__c = TRU
Number Status' of type Formula(Text(255)) with following formula: IF( AccountNumberValid__c = TRUE, 'Valid', 'Invalid' ) 8. Create a custom field on Account
formula: IF( AccountNumberValid__c = TRUE, 'Valid', 'Invalid' )

9. Create a custom field on Account object called 'Account Number Status' of type Formula(Text(255)) with following formula: IF( AccountNumberValid__c = TRU
Number Status' of type Formula(Text(255)) with following formula: IF( AccountNumberValid__c = TRUE, 'Valid', 'Invalid' )

11. Create a custom field on Account object called 'Account Number Status' of type Formula(Text(255)) with following formula: IF( AccountNumberValid__c = T
Number Status' of type Formula(Text(255)) with following formula: IF( AccountNumberValid__c = TRUE, 'Valid', 'Invalid' )

13. Create a custom field on Account object called 'Account Number Status' of type Formula(Text(255)) with following formula: IF( AccountNumberValid__c = T
Number Status' of type Formula(Text(255)) with following formula: IF( AccountNumberValid__c

=======
```

ChatGPT



```
AND( NOT(REGEX( AccountNumber , "[A-Za-z]")),
    NOT(ISBLANK(AccountNumber)),
    LEN(AccountNumber) <= 7)


This validation rule checks that the account number does not contain any alphabet,
it's not blank and it has a length less than or equal to 7 characters, If it fails
any of these conditions it will throw an error message.
You can add this rule to the Account object's page layout and set the error message
to display when the rule is violated.
To add a custom error message you can use the following formula:

AND( NOT(REGEX( AccountNumber , "[A-Za-z]")),
    NOT(ISBLANK(AccountNumber)),
    LEN(AccountNumber) <= 7,
    "Account Number should not contain any alphabet and should not be more than 7 characters in length")
``



It will display the error message "Account Number should not contain any alphabet and should not be
more than 7 characters in length" when the validation rule is violated.
It's a good idea to test your validation rule on different scenarios to make sure it behaves as
expected and also make sure to handle error conditions to avoid any data inconsistencies.
It's also a good practice to add a comment to the code explaining the validation rule's purpose
 and how it works, it will help other developers understand the code when they are working on it.
```

© 2022 Optum, Inc. All rights reserved.