This exercise is comprised of four parts:

* CSS and HTML
* JavaScript
* Database Design
* Salesforce Development

**CSS and HTML**

In the ZIP file, you will find a css.html file. You should modify this file with the editor of your choice. When viewing the source you will see inline comments that list changes that should be made to this html file. Make the necessary changes and return the file.

**JavaScript**

In the zip file, you find a javascript.html file. You should modify this file with the editor of your choice. When viewing the source you will see inline comments that list questions to answer and JavaScript code to write. Answer all the questions inline, complete all the required code, and return this file.

**Database Design**

In the ZIP file, you will find a database.html file. You should modify this file with the editor of your choice. When viewing the source you will see inline comments that list questions. Answer all of the questions inline and return this file.

Answers are uploaded to misc-questions section in the project and can be located with same filenames as in the questions. Also contains screen shots related to the exercises.

**Salesforce Development**

**Deliverables**

* Salesforce sandbox access

Sign up for a Salesforce Developer account here and perform the following work in this org:<https://developer.salesforce.com/signup>

Edit the Organization name to: Affirm Interview - {Your Name}

Once the work is complete, create a new user with the following and provide the credential to log in for review

* Github Repo

Provide an accessible link to the repo/branch that holds the components and documentations, for this assignment.

https://github.com/saratmaha/Affirm

* Technical Documentation

Write-up on design considerations and assumptions, for this assignment and store in the Github repo.

*Part 1: Trigger*

Create a custom text field on the Contact object, Account\_Industry\_\_c. Create a Contact trigger that pulls the Industry field from the related Account and populates the Account\_Industry\_\_c field on the Contact every time a Contact record is created or updated.

Also, create a test class and unit tests for this trigger.

My observations and thoughts:

1. This requirement can be achieved using standard salesforce functionality by creating a formula field because the text field value is populated based on its single parent field value. So instead of creating a text field and populating the value, formula field reference to Industry on parent object will populate the value on contact record after record save. However, there is a catch with this approach and writing custom logic makes sense depending on what we are trying to achieve. If the use case requires to use the value on contact before record save, then getting the value to use in automation makes sense because the formula field value is not available until the record is saved else if we are trying to display it on layout after the record is saved then formula field makes more sense. Also, using the formula field way should be avoided if the use case is to use in using list views, as the formula field is always calculated on the fly and have severe performance impact if the list view has large dataset.
2. This requirement can also achieve using Flows without writing Apex logic (I am not considering Workflows/Process Builders as Salesforce is planning on deprecating these features in the future and avoiding these is a best practice and would eliminate the need for any re-work in the future. <https://admin.salesforce.com/blog/2021/go-with-the-flow-whats-happening-with-workflow-rules-and-process-builder> ). There are other reasons like they will trigger after record save and it invokes unnecessary recursion, but SF announcement of retiring is one good reason to eliminate as the choice.
3. Since the requirement explicitly says to use Trigger (Apex) I will now follow the instructions to achieve the functionality. I observed the parent Account Industry field is a picklist value and the character count on any values is no greater than 30, so while creating the Account\_Industry\_\_c custom field and assuming the length of same size 30.
4. Security and Field Visibility on this field is not defined in the requirement. I assume this field has FLS only for System Admin and I created a permission set “Affirm Assesment” permission set that will have FLS and adding users to this permission set is even more scalable approach with permission set as we have more control now and add only users who want the security. The only drawback with this approach is maintaining users but I assume this is only one time activity when setting up. I assume that field should be present on all layouts and adding the field on all layouts since FLS is already handled in Permission set.
5. There are many trigger architecture frameworks that take care of some problem patterns and are addressed in their implementation. For this use case I installed Light weight trigger architecture framework from open source github project by Chris Aldridge. (<https://github.com/ChrisAldridge/Lightweight-Trigger-Framework> ). There are great advantages of using this framework and will suffice need for small – medium sized companies.
6. The core logic resides in ContactTriggerHandler class file. Added with sharing keywork to the class to make sure only the records that the user has access to will be retrieved. ( If we want the trigger to run I system mode and retrieve all records irrespective of record sharing, then with sharing key word should be removed)
7. Test class is run with 91% code coverage. Main logic is covered rest trigger context is not covered as we don’t have logic to enter those methods. (**Screenshots are added to misc-questions directory. Name of screen shot – ContactTriggerTestClassRun.png**)

*Part 2: Batch Apex*

On the Account object, create a number field called Number\_Of\_Contacts\_\_c. Create a batch job that processes all Accounts, counts the number of contacts on the account, and populates this field.

Schedule this batch job so it runs once every day.

Also, create a test class and unit tests for this batch job.

**My thoughts**

1. **This requirement can also be achieved with Flow – Scheduled trigger flow and does not require to write batch apex.**
2. **There is an app exchange package “Roll up Helper” through which we can achieve the functionality, and this is a scalable to other objects and can handle large data sets.**

**Class Names – BatchUpdateContactCount, BatchUpdateContactCountTest and scheduled in Saleforce**

**Screen shots are added to misc-questions folder**

**BatchApex\_1.png, BatchApex\_2.png, BatchApex\_Test\_1.png, ScheduleJob\_1.png, ScheduleJob\_2.png**

*Part 3: Lightning Web Component*

Create a Visualforce Page called: LWCDemo

Embed a Lightning Web Component in this Visualforce page that meets the following requirements:

* When the component loads, run the following query to fetch Accounts:

SELECT Id, Name FROM Account LIMIT 5

* Build a nice UI that displays the Accounts and lets the user select a single Account.
* When an Account is selected there should be CSS applied to indicate what Account is selected.
* There should be a Next button, when this button is clicked:
  + Fetch Id, Name, Industry, Type, Owner Name, Owner profile picture from the server.
  + This must be a separate query and not part of the original query that fetches Account Id and Name
  + Show a spinner/loading icon while this query/server call is executing
  + When data is received, display a nice UI showing the details of the selected record
  + A Back button should take the UI back to the list of Accounts

1. Created three components (one Parent - accountListDemo and two children – accoutList, accountDetail). Ideally one child will do the job and either accountDetail or accountTile can be deleted. For this exercise, I left both of them considering time factors.
2. accountListDemo will fetch the account records and display on UI and sets public property on accountTile component
3. accountTile dispatches event and is captured in the parent, then parent calls the public method on the second child accountDetail component
4. Created wrapper aura app(Lwc App) and injected web component into the aura app
5. Added the aura app on to visual force page LWC Demo (using Lightning out). One of my test works for my current company is still active. (<https://sherwin-site.herokuapp.com/> )
6. Created app builder page and dragged lwc component as well as the vfpage wrapper

Navigate to Sales Console App -> LWC Demo tab

(<https://affirminterview-saratmahavr-dev-ed.lightning.force.com/lightning/n/LWC_Demo> )

LWC Page - <https://affirminterview-saratmahavr-dev-ed--c.visualforce.com/apex/LwcDemo>