**IS Developer Interview Scenarios**

**Scenario 1**

One of the junior team members has a regular task to provide the details of new supporter signups on request so that they can be sent a welcome email via the Pure360 email platform. The instructions are as follows:

* [fundraising manager] will email you to ask for the previous week’s new signups
* Open QRY\_WELCOME\_REPORT\_20181101.sql in SSMS
* Amend the date range to the previous week
* Run the script on the live database
* Copy the output to an Excel spreadsheet
* Email the output to [fundraising manager]
* [fundraising manager] will then manually upload the output to Pure360 so as to send the welcome email

How would you summarise the current approach?

It is manual, time consuming and prone to error. It is also not good practice in terms of GDPR because personal data is being emailed and probably copied to various places along the way (email download folders etc.)

How would you do this differently:

1. if you were asked to use a ‘quick win’ approach

Create an SSIS package does the following:

* Uses the SQL script as a source. The script automatically adjusts the date criteria to the previous week. This is connected to live SQL database.
* Outputs the data to an Excel destination in a file location that the fundraising manager can access (or probably better, as a csv.)
* SSIS package automatically sends an email message to manager when done specifying where file can be found

This could just be run by the developer when required or released and run automatically on a schedule as a SQL Server agent job.

1. as part of a long term data strategy

Long term aim would be to automate this so end user doesn’t need to load the data themselves. They are just responsible for sending the email out. They might also not even want to do that. Could just be a standardised message in which case the whole thing could be automated.

* Use an SSIS package similar to the above. Again, it will output to a file location. Will run in a SQL Agent job. Schedule could be weekly, daily or even more frequently. Log the export of the file somewhere in the database with the date criteria used (so, however frequently it is done, it will always pick up new signees only.)
* Create an API application that will load the data from the file into Pure360. Call this application from the SQL agent job. (I’m assuming that Pure360’s API has the capability to load data. If not some other automated upload should be possible.)
* In Pure360, create an automated email that uses this data. If that is the requirement. Or fundraising manager just sends the email out themselves.

The best possible solution is really to send the welcome email out from the same platform. So, *if* you can collect the details in Pure360 then send it out from there.

I have created welcome feeds before from data warehouse into ExactTarget (now called Salesforce Marketing Cloud)

**Scenario 2**

We use the Engaging Networks (EN) platform for online supporter acquisition and donations, via various EN hosted landing pages that are part of our main website. We currently have a daily scheduled SQL Server agent job that runs a scripted API call to extract new records from EN; these records are then processed into our Blackbaud CRM system via a series of SQL stored procedures.

This is currently a 1 way integration, but there is a proposal to use EN to deliver real time personalised engagement with existing supporters via triggered email journeys. These consist of a series of tailored email communications that are triggered when a supporter takes an action, and can be varied according to further actions taken by a supporter.

This will require a 2 way integration, since ~~although~~ some supporter actions such as giving or revoking consent to receiving email and making donations will be captured and stored on either system (depending on the source), both systems need to be updated so that they are correct.

1. What are the additional considerations when building a 2 way integration between systems?

The challenge with 2 way integrations between systems can be to keep track of what piece of data was recorded in what system and when. This is a challenge if the same fields are being updated by the system itself and by the integration with the other system. And if the integration is incremental ie only includes new/updated records.

I have dealt with 2 way integration between Eloqua and the data warehouse where the same set of marketing consent fields were being updated by Eloqua and also by the integration from the data warehouse. Because the integration was bi-directional the same set of records would loop round the integration potentially endlessly. And much worse, under certain circumstance, the incorrect marketing consent would be derived.

1. How would you approach this?

The solution depends on how flexible the data architecture and update process is in the two platforms. And how the data is being used. You could have a different set of fields being updated from the integration with its own update date. This would mean it is possible to identify what has been captured by the system and what has been updated from the integration. This may cause issues with usability though (you would have two consent fields for example.) Or you could create separate rows from the integration (again there could be usability issue though.) The solution we used in Eloqua was for consent fields to be updated in two tables within Eloqua (main contact table and another table). The second table was used by the integration into the data warehouse. The contact table was used by the integration from the data warehouse. Something like this could also work.

**Scenario 3**

The following two scripts are syntactically correct but not working as expected. Please describe what is wrong with them:

Script 1:

IF OBJECT\_ID('tempdb..#SOURCE') IS NOT NULL DROP TABLE #SOURCE

CREATE TABLE #SOURCE

(

S\_ID UNIQUEIDENTIFIER NOT NULL

, IMD NVARCHAR(10)

)

CREATE INDEX CIDX ON #SOURCE (S\_ID)

GO

INSERT INTO #SOURCE(S\_ID, IMD)

VALUES('175fbac6-407f-4a00-8d8d-9479a92ce868',3),

('14d29432-92e5-4b2b-8e11-fedcdadbf7c4',4)

DECLARE @C\_ID UNIQUEIDENTIFIER

, @RecTotal int -- Total number of records in the trusts temp table

, @Count int -- Counter to monitor how many records have been processed via the loop

, @IMD\_V NVARCHAR(10)

, @Update int

, @Insert int

, @Add\_C\_by uniqueidentifier

if @Add\_C\_by is null

exec crm.dbo.USP\_CHANGEAGENT\_GETORCREATECHANGEAGENT @Add\_C\_by output

SET @Count = 0

SET @Update = 1

SET @Insert = 1

SET @RecTotal = (SELECT count(\*) FROM #SOURCE)

WHILE @Count < @RecTotal

BEGIN

SELECT @C\_ID = S\_id

, @IMD\_V = IMD

FROM #SOURCE

ORDER BY S\_ID

OFFSET @COUNT ROWS

FETCH NEXT 1 ROWS ONLY

--Check to see if the constituent exists and then update the values.

IF EXISTS(select \* from CRM.dbo.ATTRIBUTEC7A5E4CE0FCA44948AACEE1766ED901B where CONSTITUENTID = @C\_ID)

PRINT 'EXISTS'

UPDATE CRM.dbo.ATTRIBUTEC7A5E4CE0FCA44948AACEE1766ED901B

SET value =

( CASE --Depending on whether the Postcode is valid or not (-99 not valid) it will either update the

WHEN (@IMD\_V <> -99) THEN @IMD\_V --constituent record Decile score with the latest IMD Value, however if the Postcode isn't valid

ELSE value --it retains the existing value. This will be the same for the Comment and Start date.

END

),

comment =

( CASE

WHEN (@IMD\_V <> -99) THEN 'has a post code'

ELSE 'Does not have a post code'

END

),

STARTDATE =

( CASE

WHEN (@IMD\_V <> -99) THEN GETDATE()

ELSE STARTDATE

END

)

PRINT @UPDATE

SET @Update = @Update + 1

-- increment the counter & output results

NEXT:

PRINT @C\_ID

PRINT @IMD\_V

PRINT @RecTotal

PRINT @Count

SET @Count = @Count + 1

END

There is no where clause in the update statement. It is checking to see if there are rows with the CONSTITUENTID but is not using it to update the rows. It is actually updating all rows each loop. It should be something like the below.

IF EXISTS(select \* from CRM.dbo.ATTRIBUTEC7A5E4CE0FCA44948AACEE1766ED901B where CONSTITUENTID = @C\_ID)

PRINT 'EXISTS'

UPDATE CRM.dbo.ATTRIBUTEC7A5E4CE0FCA44948AACEE1766ED901B

SET value =

( CASE --Depending on whether the Postcode is valid or not (-99 not valid) it will either update the

WHEN (@IMD\_V <> -99) THEN @IMD\_V --constituent record Decile score with the latest IMD Value, however if the Postcode isn't valid

ELSE value --it retains the existing value. This will be the same for the Comment and Start date.

END

),

comment =

( CASE

WHEN (@IMD\_V <> -99) THEN 'has a post code'

ELSE 'Does not have a post code'

END

),

STARTDATE =

( CASE

WHEN (@IMD\_V <> -99) THEN GETDATE()

ELSE STARTDATE

END

)

WHERE CONSTITUENTID = @C\_ID

PRINT @UPDATE

SET @Update = @Update + 1

**Scenario 3**

Script 2:

DROP TABLE NDCS\_ETL.dbo.EN\_temp\_phone\_exclusions;

GO

SELECT e.tableid,

e.constituentid,

e.constituentname,

e.campaign\_date,

e.phone\_consent\_to\_contact,

sc1.STARTDATE AS [VIKI phone solicit code startdate],

sc1.[DESCRIPTION] AS [VIKI phone solicit code]

INTO NDCS\_ETL.dbo.EN\_temp\_phone\_exclusions

FROM ##temp\_EN\_consent\_errors e

-- solicit code set by this staging record

LEFT JOIN (

-- distinct used to tidy up output where more than 1 donation made on same day

SELECT DISTINCT csc1.CONSTITUENTID, sc1.[DESCRIPTION], STARTDATE

FROM CRM.dbo.CONSTITUENTSOLICITCODE csc1

INNER JOIN CRM.dbo.SOLICITCODE sc1 ON csc1.SOLICITCODEID =sc1.ID

AND sc1.[DESCRIPTION] LIKE '%phone%'

WHERE csc1.COMMENTS LIKE 'EN%'

) sc1 ON e.CONSTITUENTID = e.CONSTITUENTID

WHERE [Interpreted phone consent] <> [NEW Interpreted phone consent]

-- no subsequent opt in

AND NOT EXISTS (

SELECT CONSTITUENTID

FROM CRM.dbo.CONSTITUENTSOLICITCODE csc2

INNER JOIN CRM.dbo.SOLICITCODE sc2 ON csc2.SOLICITCODEID =sc2.ID

AND sc2.[DESCRIPTION] = 'Phone opt-in'

WHERE csc2.CONSTITUENTID = e.constituentid

AND DATEDIFF(DAY, e.campaign\_date, csc2.STARTDATE) > 0

)

There is an error in the join from the ##temp\_EN\_consent\_errors table to the subquery. So, this…

FROM ##temp\_EN\_consent\_errors e

-- solicit code set by this staging record

LEFT JOIN (

-- distinct used to tidy up output where more than 1 donation made on same day

SELECT DISTINCT csc1.CONSTITUENTID, sc1.[DESCRIPTION], STARTDATE

FROM CRM.dbo.CONSTITUENTSOLICITCODE csc1

INNER JOIN CRM.dbo.SOLICITCODE sc1 ON csc1.SOLICITCODEID =sc1.ID

AND sc1.[DESCRIPTION] LIKE '%phone%'

WHERE csc1.COMMENTS LIKE 'EN%'

) sc1 ON e.CONSTITUENTID = e.CONSTITUENTID

Should be this…

FROM ##temp\_EN\_consent\_errors e

-- solicit code set by this staging record

LEFT JOIN (

-- distinct used to tidy up output where more than 1 donation made on same day

SELECT DISTINCT csc1.CONSTITUENTID, sc1.[DESCRIPTION], STARTDATE

FROM CRM.dbo.CONSTITUENTSOLICITCODE csc1

INNER JOIN CRM.dbo.SOLICITCODE sc1 ON csc1.SOLICITCODEID =sc1.ID

AND sc1.[DESCRIPTION] LIKE '%phone%'

WHERE csc1.COMMENTS LIKE 'EN%'

) sc1 ON e.CONSTITUENTID = sc1.CONSTITUENTID