Question1:

Login to GitHub

Create new repo- '+' on top right corner of home, give repo name, initialise with readme, click create repo eg demorepo

Create new folder on PC eg- gitdemo

open cmd from inside gitdemo folder or Navigate to that folder

give command- git init

Go to your new repo, click on clone or download on top right,copy the URL given for Clone with HTTPS

Give command- git clone "paste your copied url"

Browse to gitdemo, you will have demorepo folder inside already, enter that folder

minimize and open VS. Create new VS project in demorepo folder(browse to it inside when creating new project) eg demoproject

Go to demorepo folder and check if you have your VS project there

From inside demorepo, open cmd

Give command- git branch (checks if on correct path, it will say you are on Master if things are ok)

Write a test output in VS project's main for eg Hello Browser, save changes, close it.

In cmd give command- git add .

git commit -m "file added etc msg"

git push -u origin master

Refresh your Git repo page. Now it will be updated in git with your msg. Enter the demoproject file. Enter into subfolder to find program.cs file. Check if it has "hello browser'

Open your console app to edit more, write code according to question

save it, close it, go to cmd from inside demorepo folder

give command- git add .

git commit -m "msgprompt"

git push -u origin master

Your changes are saved. Check in GitHub repo.

Question 2:

Login to VS using same account as azure

open cloud explorer

For code as per lab with hard coded values -

Go to GitHub.com/zazzy1 - lab5- lab 5 - program.cs (Case 1)

For code where we have switch cases for entry/updation delete-

Goto GitHub.com/mohithadira- M4Stuff- program.cs

Copy paste this code into your new vs program.cs file

Go to tools- nuget package manager- manage nuget packages for solution- browse- follow step 5 of lab to add the 2 packages

go to app.config - follow step 7 in lab5

for value parameter in appSettings- go to azure- your storage account- click on access keys on left panel- copy either of the 2 connection strings given- paste in the value parameter

build and run solution

Case 1- To see your code in action : comment delete table and del entity functions, run again, open cloud explorer, navigate to you storage account- open your table - refresh- you should find the hardcoded value in the table

To run program again, change the hard coded value for name (it's hardcoded as IND) OR del the entry from the table manually

Case 2-

Use the switch cases to test all CRUD. Alter definitions as per question!

QUESTION 3-

create resource group

go to resource group- add(+)- storage- storage account- give a name- select locally redundant db- create

open VS and login using same account as Azure

view- open cloud explorer- open your resource group- expand your storage account

right click on queue- create queue- give name eg myqueue - enter

repeat prev step with table to create mytable

go to azure- add resource- Integration- logic app - give name- select your resource group- create

go to resource for your logic app- scroll down- blank logic app template

follow steps 8 to 15 from lab 6 carefully.

Notes- be very careful with json key , copy exactly as step 12 or it will give error.

once done with all 3 steps, save from top left corner

click on run

in cloud explorer, go to your queue, click on message icon on top left to add msg, write msg first- enter

wait for 1 min, go to table, refresh from top right- you should have your value added in queue moved to table

go to queue, refresh, value should have been del

add more Val if needed, check table to confirm if values are being recorded

QUESTION 4-

Login to azure

create resource group

go to resource group

compute- add function app (CLASSIC)- give name- choose .Net core for run time stack- create

go to resource- on left panel- click + in front of functions

Case 1- Using Visual Studio, Case 2- Using In-portal

Case 1- Using VS- Direct publish- stay on this page and don't press finish. This page give all steps as a guide. These same steps are below

Open VS- new project- cloud - azure function- give name- select HTTP trigger- ok

press f5 to run , it will open cmd in some time, keep waiting till it gives you an http address

copy that address, make sure vs execution is still running

paste that string in browser , suffix with- ?& name=pranshu

you should see hello, pranshu when you press enter

To add multiple parameters- duplicate every place "name" is there on your code. Eg to add city-

string city=req.Query[city]; city= city ?? data?.city; okObjectResult ($"Hello, {name} from {city})....

Run your code in vs

go to browser , paste same url

this time add suffix: ?& name=pranshu&city=Bangalore

it should show Hello Pranshu from Bangalore

Case 2: In portal- using webhook + api- create

it will have same code as before in VS that we got from http triggered azure function

click on get function url, copy

click on run

in browser paste url- suffix with &name=pranshu

same process to add second parameter as Case 1

QUESTION 5-

create resource

add - databases- sql database- give name- create new server- give credentials- create

go to resource- firewall settings- + Add client IP- toggle on for " Allow azure services and resources to access this - save

go to query editor from left panel- enter password- write sql to create a table WITH PRIMARY KEY

go to Vs- create new- cloud- asp.net web app- ok- mvc

model right click- add new - ado.net entity data model- ef designer from dbs- new connection

go to your sql azure db- copy "server name"

paste server name

change authentication to Sql server authentication and give your credentials

see drop down menu for "select or enter a database name"- you should be able to see name of db you created on azure- select it

test connection- if succeeded- ok

select yes, include sensitive data.... Next-next- Cascade down in tables to find your table- check it- finish

build solution

add controller- 3rd option (using entity framework)- select your table from drop down- select data context class from drop down- Add

Go to App start, replace controller="Home" with your controller's name

Save all in VS

Go to azure- add- web- web app- select your resource group- select Asp.net V 4.7 as Runtime Stack- select Windows- create

go to resource- get publish profile- it downloads

Go to VS- right click on project- publish- import profile- select file which u downloaded right now- it should open in browser automatically

Enjoy life with Crud operations