# Coding Assignments

## SQL

```
--EMPLOYEE--
    empid   empname managerid   deptid  salary  DOB
    1       emp 1       0       1       6000    1982-08-06 00:00:00.000
    2       emp 2       0       5       6000    1982-07-11 00:00:00.000
    3       emp 3       1       1       2000    1983-11-21 00:00:00.000
    13      emp 13      2       5       2000    1984-03-09 00:00:00.000
    11      emp 11      2       1       2000    1989-07-23 00:00:00.000
    9       emp 9       1       5       3000    1990-09-11 00:00:00.000
    8       emp 8       3       1       3500    1990-05-15 00:00:00.000
    7       emp 7       2       5       NULL    NULL
    3       emp 3       1       1       2000    1983-11-21 00:00:00.000

--DEPARTMENT--
    deptid  deptname
    1       IT
    2       Admin
```

1. Write a SQL query to delete duplicate records from employee table which has no primary key.

2. Suppose there is a table called REGION with columns (regid, regname). Before you could insert any data in it you realize that you no longer need this table in your database. You execute the below query to drop this table. What will be its result? Provide explanation

```
IF EXISTS (SELECT TOP 1 * FROM dbo.region)
BEGIN
        DROP TABLE dbo.region;
END
```

3. Write a query to get employee details whose department id is **not** valid or department id is **not** present in the department table. There is more than one way to write this query. Which method would you least recommend to another developer?

# React

We are looking to evaluate your proficiency with React, Redux, APIs and front-end design with a small project. There will be two parts, each one hour or less to complete.

## Setup

You can use `create-react-app` to scaffold your application. https://github.com/facebook/create-react-app

## Part 1: API Integration

Given the following API endpoints:

- [1] https://api.inquickerstaging.com/v3/winter.inquickerstaging.com/services
- [2] https://api.inquickerstaging.com/v3/winter.inquickerstaging.com/providers?include=locations%2Cschedules.location&page%5Bnumber%5D=1&page%5Bsize%5D=10

### *Assignment*

Having fetched a list of services [1] and providers [2] from the API, create an interface that has two sections:

- A 'control' section which displays each service.
- A 'results' section which displays all provider names.

Then:

- If a service name is clicked, that service gets highlighted, and
- Only providers associated with that service gets shown in the 'results' section.

### *Evaluation*

We will not be evaluating front-end UI/UX for this portion. We will be primarily looking for:

- [ ] Data successfully retrieved from API
- [ ] Data stored in Redux
- [ ] Filtering functionality using Redux state
- [ ] Data correctness

## Part 2: Design and Code Quality

### *Assignment*

Building on the previous section, we are going to enhance the results list to include more than just the providers' names.

Create a component that shows:

- a provider's image, on the left
- their name, on the right
- subspecialties, if any, on the right, below the name

Here is an ASCII representation of the component:

```
|-----------------------------------|
|   |-------|   Provider Name       |
|   | image |                       |
|   |       |   Subspecialties      |
|   |-------|                       |
|-----------------------------------|
```

Feel free to use a UI library such as https://material-ui.com/ to speed up development.

### *Evaluation*

We will not evaluate the UI/UX of the control section (that contains the service names) so that can remain relatively unstyled. We will be looking primarily at the result section, the new component, and how the code is organized/written. The criteria:

- [ ] Application is composed of multiple components.
- [ ] Component renders well, and with correct layout, on mobile/desktop viewports.
- [ ] Data correctness, when comparing frontend results to API data.

### *Submission*

- Your project should be hosted in a BitBucket or GitHub repository. Please provide the direct link to the repo.
- Your readme should include any steps necessary to run the project locally.

## .Net

Please write a program to solve the following requirements with “good practice” algorithm, coding, design and architecture.

You are working in a life insurance company. You need to create a Web API to solve to the following requirements. The Web API should be written in C# and .NET Core.

- Provide API for creating a new life insurance contract
    - The API should accept customer name, address, date of birth, gender, and sale date.
    - Based on customer country, age, gender, and sale date, the system should determine
        - coverage plan from “Coverage Plan” table
        - net rate from “Rate Chart” table

      and save that to database along with other data.

- Provide API for updating an existing life insurance contract
    - The system should update coverage plan and net rate based on latest customer information.
- Provide API for deleting an existing life insurance contract
- Provide API for getting a list of life insurance contracts
- Contract data should be saved in RDMS database (you can use any)
- No UI required
- Here are configuration data in “Coverage Plan” and “Rate Chart” tables

*Coverage Plan*

| Coverage Plan | Eligibility Date From | Eligibility Date To | Eligibility Country |
|---|---|---|---|
| Gold | 1/1/2009 | 1/1/2021 | USA |
| Platinum | 1/1/2005 | 1/1/2023 | CAN |
| Silver | 1/1/2001 | 1/1/2026 | * (any) |

**Rate Chart**

| Coverage Plan | Customer Gender | Customer Age | Net Price |
|---|---|---|---|
| Gold | M | <=40 | 1000 |
| Gold | M | >40 | 2000 |
| Gold | F | <=40 | 1200 |
| Gold | F | >40 | 2500 |
| Silver | M | <=40 | 1500 |
| Silver | M | >40 | 2600 |
| Silver | F | <=40 | 1900 |
| Silver | F | >40 | 2800 |
| Platinum | M | <=40 | 1900 |
| Platinum | M | >40 | 2900 |
| Platinum | F | <=40 | 2100 |
| Platinum | F | >40 | 3200 |

Here is example of data in “Contracts” tables

*Contracts*

| Customer Name | Customer Address | Customer Gender | Customer Country | Customer Date of birth | Sale Date | Coverage Plan | Net Price |
|---|---|---|---|---|---|---|---|
| John | Dixon | M | USA | 1/1/1975 | 1/1/2012 | Gold | 2000.00 |
| Jolie | Grande | F | CAN | 1/1/1980 | 1/1/2007 | Platinum | 2100.00 |
| Helen | Young | F | CHI | 1/1/1960 | 5/5/2005 | Silver | 2800.00 |

Good luck!