# Data Modeling Questions

* Design a database to model a cookbook with meal types, recipes, ingredients.

# SQL Questions

* Given the following SQL SELECT statement what are the highlighted text represent?

SELECT ***CUST***.CUSTOMER AS ***NAME***, ***CUST***.CUSTOMER\_ADDRESS

FROM CUSTOMER\_V ***CUST***

WHERE ***NAME*** = ‘Home Furnishings’;

Answer: CUST is the table, NAME is an alias.

* Given the following Orders and Customers tables:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **OrderID** | | **CustomerID** | **OrderDate** | |
| 10308 | | 2 | 1996-09-18 | |
| 10309 | | 37 | 1996-09-19 | |
| 10310 | | 77 | 1996-09-20 | |
| **CustomerID** | **CustomerName** | | | **ContactName** | | **Country** |
| 1 | Alfreds Futterkiste | | | Maria Anders | | Germany |
| 2 | Ana Trujillo Emparedados y helados | | | Ana Trujillo | | Mexico |
| 3 | Antonio Moreno Taquería | | | Antonio Moreno | | Mexico |

What would the result set be given the following SQL SELECT statement?

SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate  
FROM Orders  
INNER JOIN Customers  
ON Orders.CustomerID=Customers.CustomerID;

Answer: SQL would produce only 1 result row for CustoimerID “2”.

What would the result set be given the following SQL SELECT statement?

SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate  
FROM Orders  
LEFT JOIN Customers  
ON Orders.CustomerID=Customers.CustomerID;

Answer: Full list of OrderID and OrderDate with the CustomerName associated with each order.

# Entity Framework Questions – Skipped due to lack of understanding.

* What do the following Entity Framework Object Services provide for your application in regards to data from a database?
  1. Materialization
  2. Change Tracking
  3. Object identities
* Design a code first data model which has a Project class that can contain a bunch of tasks.

namespace EFDemo.Model  
{  
 public class Project  
 {  
 // Code First infers this as the primary key column  
 public int Id { get; set; }  
 // this becomes a nullable column  
 public string Name { get; set; }  
  
 // list of tasks for a project  
 public virtual List<Task> Tasks { get; set; }  
 }  
  
 public class Task  
 {  
 // Code First infers this as the primary key column  
 public int TaskId { get; set; }  
 public string Name { get; set; }  
  
 // this is inferred as Foreign key to project table  
 public int ProjectId { get; set; }  
 }  
   
}

# Object Orientation Questions

* What are the basic concepts of OOP?

Answer: Objects can be coded and re-used by other program functionality.

* How do you program behavior into your C# class?

Answer: By using Inheritance. Base Classes standardize behavior.

* Explain method overriding.

Answer: Overriding a Base Class Method with a Method in the Derived Class.

* What is Inheritance?

Answer: Using a Base Class to standardize behavior of the Derived Classes.

* What is abstract class?

Answer: Sorry, not all that knowledgeable on Abstract Classes.

# MVC Questions

What is an example URL that would call the following controller method, assuming the default routes have been configured?

Answer: No Idea.

Part II: What HTTP Verb is used?

Answer: No Idea.

public class CatalogController : Controller

 {

     public ActionResult Specifications(int id)

     {

         var model = new SpecModel(id);

         return View();

     }

}

Examine the following View for an MVC Application:

@model IEnumerable<MVCGuidedLab.Models.Color>

@{

    ViewBag.Title = "Index";

}

<h2>Index</h2>

<p>

    @Html.ActionLink("Create New", "Create")

</p>

<table class="table">

    <tr>

        <th>

            @Html.DisplayNameFor(model => model.Name)

        </th>

        <th>

            @Html.DisplayNameFor(model => model.Value)

        </th>

        <th></th>

    </tr>

@foreach (var item in Model) {Go

    <tr>

        <td>

            @Html.DisplayFor(modelItem => item.Name)

        </td>

        <td>

            @Html.DisplayFor(modelItem => item.Value)

        </td>

        <td>

            @Html.ActionLink("Edit", "Edit", new { id=item.Id }) |

            @Html.ActionLink("Details", "Details", new { id=item.Id }) |

            @Html.ActionLink("Delete", "Delete", new { id=item.Id })

        </td>

    </tr>

}

</table>

1. What type is the Model? Create.
2. Is the model a single object, or a sequence? Single.
3. What properties are on the items in the model? Name and Value.

# Programming Exercises

To turn in this exam, you’ll create a fork of a repository, make modifications to the project in that repository, and submit a pull request with your changes.

We’ll walk you through the github workflow for those pieces. However, the code will be yours.

Go to Github.com and fork the repository <https://github.com/BillWagner/ExperienceITExam>

Then, clone your fork to your desktop. You’ll do that by clicking the “Clone in Desktop” button on the github page.

Now, you’re ready to do your work.

1. Open the program.cs file in the Loops project. The Main method has comments that describe what you should do. Make your changes.
2. Open the program.cs file in the Extension methods project. The Main method has comments that describe the code you should add. Make your changes.
3. Open the program.cs file in the Lazy Evaluation project. The Main method has comments that describe the code you should add. Make those changes as well.
4. Open the program.cs file in the Query Expresssions project. As before, the Main method has comments that describe the code you should or change.
5. Make a new ASP.NET MVC project and add it to the solution. Add a controller called DiceRoll controller. Modify the index method and the index view to display all the combinations of results from rolling 2 six-sided dice. (For example, { 1, 1}, {1, 2} etc.

Commit your changes, and then push them to github.

Now, you’re ready to submit a pull request. Navigate to your fork of the repository in github.com in a browser. Below the Code tab on the right side, you’ll see a link that says “Pull Request”. Click that. Once you’re on the Pull Request page, submit a new pull request. Add your name, and any comments you would like on your pull request, and submit the request.

We can now look at your changes, and see how you did.