Coding Challenges: DML and SOQL

SOQL, SOSL and DML

These challenges should be done in a Trailhead Playground, not a Scratch Org! That way, you have records to work with.

SOQL

All these challenges can be completed in a single line using a cleverly structured SOQL query. Take the extra challenge to come up with the solution that uses the lowest number of lines of code!

Challenge I

Write a class called `DataLord` that has 1 method called `NumAccounts`. This method should return the number of `Accounts` that exist in org

Challenge II

Write a method called `NumHot`. This method should return the number of `Accounts` with a `Rating` of Hot

Challenge III

Write a method called `NumComplex`. This method should return the number of `Accounts` that have an ampersand (&) in the `Name`

Challenge IV

Write a method called `BigFive`. This method should return the top 5 `Accounts` in terms of `Annual Revenue`. You may only use 1 line of code for the method body

Challenge V

Write a method called `LastBigOpportunity`. This method should return the `Opportunity` with the most recent `Close Date` that had an `Amount` over $100,000.

Challenge VI

Write a method called `MajorPlayers`. This method should return all `Contacts` that are associated to an `Account` with a `Rating` of Hot.

Challenge VII

Write a method called `MostPopularAccount`. This method should return the `Account` `Name` that has the most associated `Contacts`.

Elite Challenge I

Universal Containers is interested in knowing the top 5 `Lead` `Sources` that are most likely to close and how likely those `Sources` are to close. They also wish to ignore the case where the `Lead` `Source` is blank. Write a single SOQL query to retrieve this information.

SOSL

Challenge I

Write a method to retrieve all `Contacts` and `Leads` that are named Tom.

Challenge II

Write a method to retrieve all `Accounts` that have an 'a' and an 'o' in one of their fields.

Elite Challenge I

Robin knows that there is a phone number that ends in 1000, but has no idea which field it is in. They know it is either on an `Account`, `Contact`, or `Lead` and just need the `Name`. Write a method that solves Robin's issue.

DML

Challenge I

Write a method called `SpawnContacts` that creates and inserts 200 uniquely named `Contacts` into the database.

Challenge II

Write a method called `CreateHomes` that creates and inserts 3 uniquely named `Accounts` into the database.

Challenge III

Write a method called `Relocate` that deletes the 50 most recent `Contacts` in the database.

Challenge IV

Write a method called `AssignHomes` that relates the all `Contacts` created in these challenges to the 3 `Accounts` randomly.

Note: This should only assign `Contacts` created during this challenge, and should ignore all `Contacts` created otherwise.

Elite Challenge I

Write a method called `Play`. This method should set a savepoint and rollback to the savepoint just before ending. It should then call all 4 methods in prior DML challenges then, before rolling back, print to the Debug Log some statistics about the state of the Database. Include: Number of `Contacts` and `Accounts` at the start, Number of `Contacts` and `Accounts` currently, and the number of Related `Contacts` for each `Account`.

Master Challenge

Create a class called `PlaygroundSetup`. It should have a method called `Setup` that creates starting data using the following rules:

- `Setup` should accept a parameter of type `Integer`
    - This parameter represents the number of `Accounts` and `Leads` to create
    - This parameter, divided by 2, represents the number of `Products` to create.

- Each `Account` should have:
    - a unique `Name`
    - a random `Annual Revenue` between $10,000 and $3,000,000
    - a `Rating` based on their `Annual Revenue`:
        - Cold: `Annual Revenue` < $75,000
        - Warm: $75,000 < `Annual Revenue` < $750,000
        - Hot: $750,000 < `Annual Revenue`
    - a random number of `Contacts` (between 0-4)
- Each `Contact` should have:
    - a random `Name` (use a premade list of First and Last names to pull from)
    - an `Email Address` in the form LastNameFirstName@AccountName.com
- Each `Lead` should have:
    - a random `Name` (same as Contacts)
    - a random `Company Name` that matches an `Account Name` we created
    - an `Email Address` in the same format as the `Contacts`
- Each `Product` should have:
    - a random `Price` in the Standard `Pricebook`
    - a `Name`
    - a unique `Product Code`

Use multiple helper methods in order to logically sort your code. No matter how many records are created, your solution should use the same number of DML statements and SOQL queries.

Note: Your org has limited data storage. Don't try doing this with too many records at once! Stick to low values, like 10-20.