```
import UIKit
import ArcGIS


let Satellite_url = AGSArcGISTiledLayer(url: URL(string:"https://
services.gisqatar.org.qa/server/rest/services/Imagery/QatarSatelitte/
MapServer")!)

class SubLayerViewController: UIViewController, AGSGeoViewTouchDelegate {

    @IBOutlet private var mapView: AGSMapView!
    @IBOutlet private var featureLabel:UILabel!

    private var distFeatureLayer: AGSFeatureLayer!
    private var distFeatureTable: AGSServiceFeatureTable!
    private var selectedDist: AGSArcGISFeature!
    private var screenPoint: CGPoint!

    private var results = [AGSRelatedFeatureQueryResult]()

    var map: AGSMap!

    override func viewDidLoad() {
        super.viewDidLoad()

        self.map = AGSMap(basemap: AGSBasemap(baseLayer: Satellite_url))
        self.SetMapExtent()
        self.mapView.map = self.map

        // add self as the touch delegate for map view
        self.mapView.touchDelegate = self

        // create feature table for the district layer, the origin layer in the
relationship
        self.distFeatureTable = AGSServiceFeatureTable(url: .distService)

        // feature layer for district
        let distFeatureLayer = AGSFeatureLayer(featureTable:
self.distFeatureTable)

        // add district feature layer to the map
        map.operationalLayers.add(distFeatureLayer)

        // Feature table for related Census layer
        let censusFeatureTable = AGSServiceFeatureTable(url: URL(string:
"https://khazna.gisqatar.org.qa/fed/rest/services/GISUDBAP/
QSA_Census_District_2020/FeatureServer/234")!)

        // add these to the tables on the map
        // to query related features in a layer, the layer must either be added
as a feature
        // layer in operational layers or as a feature table in tables on map
        map.tables.addObjects(from: [censusFeatureTable])

        // set selection color
        mapView.selectionProperties.color = .yellow

        // store the feature layer for later use
        self.distFeatureLayer = distFeatureLayer

        AGSAuthenticationManager.shared().delegate = self
//        findRelationships()

    }
```

```
    func SetMapExtent () {

        let envelope = AGSEnvelope (xMin: 155000.602394999, yMin: 297009.112383,
xMax: 246023.663305001, yMax: 495960.811646, spatialReference:
self.mapView?.spatialReference)
        self.map.initialViewpoint = AGSViewpoint(targetExtent: envelope)

    }

    // MARK: - AGSGeoViewTouchDelegate

    func geoView(_ geoView: AGSGeoView, didTapAtScreenPoint screenPoint:
CGPoint, mapPoint: AGSPoint) {
        // unselect previously selected park
        if let previousSelection = self.selectedDist {
            self.distFeatureLayer.unselectFeature(previousSelection)
        }

        // identify features at the tapped location
        self.mapView.identifyLayer(self.distFeatureLayer, screenPoint:
screenPoint, tolerance: 12, returnPopupsOnly: false) { [weak self] (result:
AGSIdentifyLayerResult) in

            if let error = result.error {
                // dismiss progress hud
                self?.presentAlert(error: error)
            }
            // Check if a feature is identified
            else if let feature = result.geoElements.first as? AGSArcGISFeature
{
                // store as selected park to use for querying
                self?.selectedDist = feature

                let dist = feature.attributes.value(forKey: "DIST_NO") as! Int
                self?.featureLabel.text = String(dist)

                // select feature on layer
                self?.distFeatureLayer.select(feature)

                // store the screen point for the tapped location to show
popover at that location
                self?.screenPoint = screenPoint

                // query for related features
                self?.queryRelatedFeatures()
            }
        }
    }



    func findRelationships() {
        Task {
            print("Loading MapServer...")
            let mapImageLayer = AGSArcGISMapImageLayer(url: .censusService)
            try? await mapImageLayer.load()
            print("Loaded")
            for item in mapImageLayer.mapImageSublayers {
                guard let sublayer = item as? AGSArcGISMapImageSublayer else
{ return }
                try? await sublayer.load()
                let relationshipInfos =
sublayer.mapServiceSublayerInfo?.relationshipInfos
                relationshipInfos?.forEach { relationshipInfo in
```

```
//                    if relationshipInfo.name == "Pop_Sex_2020_DL" {
//                        self.relationshipInfo = relationshipInfo
//                    }

                    print(relationshipInfo.name)
                }
            }
        }
    }


    @IBAction func QueryMaleFemale(_ sender: UIBarButtonItem) {
        queryRelatedFeatures()

    }


    // query for related features given the origin feature
    private func queryRelatedFeatures() {

        // show progress hud
//        UIApplication.shared.showProgressHUD(message: "Querying related
features")

        // query for related features
        self.distFeatureTable.queryRelatedFeatures(for: self.selectedDist)
{ [weak self] (results: [AGSRelatedFeatureQueryResult]?, error: Error?) in
            // dismiss progress hud
//            UIApplication.shared.hideProgressHUD()

            guard let self = self else { return }

            if let error = error {
                // display error
                self.presentAlert(error: error)
            } else if let results = results {
                // Show the related features found in popover
                if !results.isEmpty {
                    self.results = results
                    self.showRelatedFeatures()
                } else {  // else notify user
                    self.presentAlert(message: "No related features found")
                }
            }
        }

    }

    // show related features in a table view as popover
    private func showRelatedFeatures() {
        // perform popover segue
        self.performSegue(withIdentifier: "RelatedFeaturesSegue", sender: self)
    }

    // MARK: - Navigation
    override func prepare(for segue: UIStoryboardSegue, sender: Any?) {

//        if segue.identifier == "RelatedFeaturesSegue",
//            let navController = segue.destination as? UINavigationController,
//            let controller = navController.viewControllers.first as?
RelatedFeaturesListViewController {
//            // set results from related features query
//            controller.results = results
//        }

        if segue.identifier == "RelatedFeaturesSegue"{
```

```
            let controller = segue.destination as?
RelatedFeaturesListViewController
            controller?.results = self.results

        }


    }


}

extension SubLayerViewController: AGSAuthenticationManagerDelegate {
    func authenticationManager(
        _ authenticationManager: AGSAuthenticationManager,
        didReceive challenge: AGSAuthenticationChallenge
    ) {
        let credentials = AGSCredential(
            user: "cgis_user",
            password: "cgis_user123"
        )
        challenge.continue(with: credentials)
    }
}

//extension SubLayerViewController: AGSAuthenticationManagerDelegate {
//    func authenticationManager(_ authenticationManager:
AGSAuthenticationManager, didReceive challenge: AGSAuthenticationChallenge) {
//        // NOTE: Never hardcode login information in a production application.
This is done solely for the sake of the sample.
//        let credentials = AGSCredential(user: "cgis_user", password:
"cgis_user123")
//        challenge.continue(with: credentials)
//    }
//}


extension URL {
    static var censusService: URL {
        URL(string: "https://khazna.gisqatar.org.qa/fed/rest/services/GISUDBAP/
QSA_Census_District_2020/MapServer")!
    }
}
```