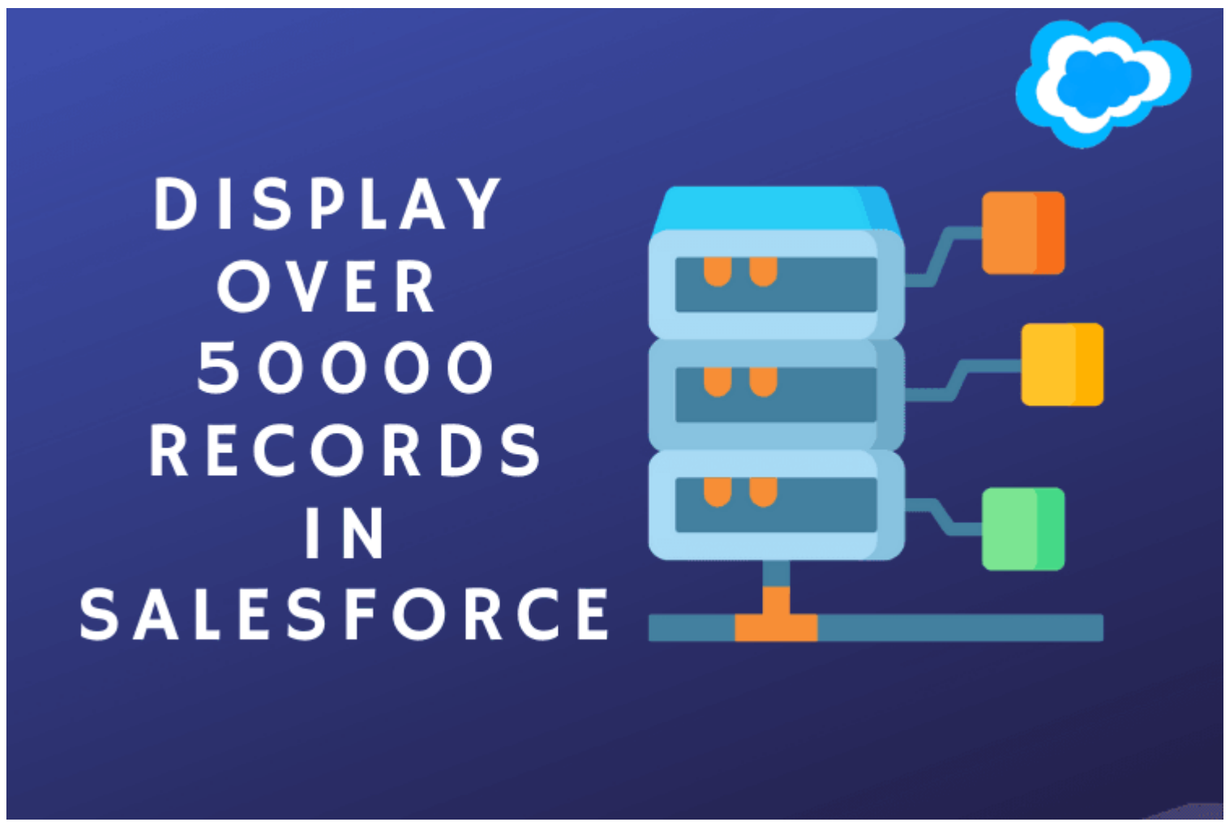
Handle Bulk data in Lightning Component



Handle and display bulk(huge) data in Lightning component. Display more than 50000 records in lightning component.

Summary

Retrieve Data

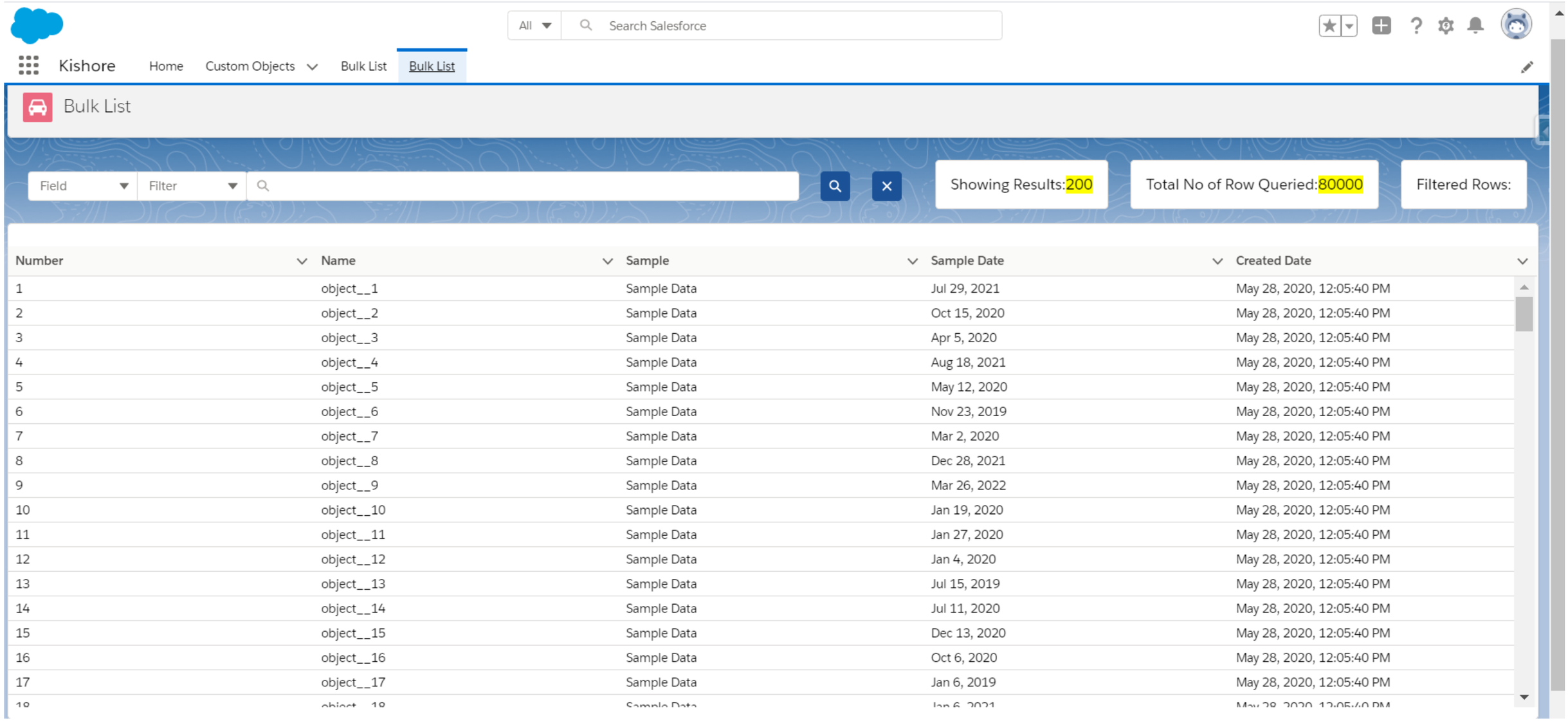
There are 3 ways we can retrieve bulk data in Lightning.

1. VF remoting with @readonly annotation (nearly 1 million records)
2. VF remoting with @readonly annotation (multiple small chunks)
3. Lightning controller (multiple small chunks)

we can retrieve data using any of the above 3 ways.

Salesforce says,

> The `@ReadOnly` annotation allows you to perform unrestricted queries against the Lightning Platform database. All other limits still apply. It's important to note that this annotation, while removing the limit of the number of returned rows for a request, blocks you from performing the following operations within the request: DML operations, calls to `System.schedule`, calls to methods annotated with `@future`, and sending emails.
>
> The `@ReadOnly` annotation is available for Web services and the `Schedulable` interface. To use the `@ReadOnly` annotation, the top level request must be in the schedule execution or the Web service invocation. For example, if a Visualforce page calls a Web service that contains the `@ReadOnly` annotation, the request fails because Visualforce is the top level request, not the Web service.



Number	Name	Sample	Sample Date	Created Date
1	object__1	Sample Data	Jul 29, 2021	May 28, 2020, 12:05:40 PM
2	object__2	Sample Data	Oct 15, 2020	May 28, 2020, 12:05:40 PM
3	object__3	Sample Data	Apr 5, 2020	May 28, 2020, 12:05:40 PM
4	object__4	Sample Data	Aug 18, 2021	May 28, 2020, 12:05:40 PM
5	object__5	Sample Data	May 12, 2020	May 28, 2020, 12:05:40 PM
6	object__6	Sample Data	Nov 23, 2019	May 28, 2020, 12:05:40 PM
7	object__7	Sample Data	Mar 2, 2020	May 28, 2020, 12:05:40 PM
8	object__8	Sample Data	Dec 28, 2021	May 28, 2020, 12:05:40 PM
9	object__9	Sample Data	Mar 26, 2022	May 28, 2020, 12:05:40 PM
10	object__10	Sample Data	Jan 19, 2020	May 28, 2020, 12:05:40 PM
11	object__11	Sample Data	Jan 27, 2020	May 28, 2020, 12:05:40 PM
12	object__12	Sample Data	Jan 4, 2020	May 28, 2020, 12:05:40 PM
13	object__13	Sample Data	Jul 15, 2019	May 28, 2020, 12:05:40 PM
14	object__14	Sample Data	Jul 11, 2020	May 28, 2020, 12:05:40 PM
15	object__15	Sample Data	Dec 13, 2020	May 28, 2020, 12:05:40 PM
16	object__16	Sample Data	Oct 6, 2020	May 28, 2020, 12:05:40 PM
17	object__17	Sample Data	Jan 6, 2019	May 28, 2020, 12:05:40 PM

Video Recording:

VF remoting with @readonly annotation (nearly 1 million records)

We can query nearly 1 million records (Salesforce says unlimited records) by annotation an apex method with @readyonly. However, there are certain limitations. @readonly annotation in this context can only be called via vf remoting to serve its purpose to perform unrestricted queries. The @RemoteAction annotation provides support for Apex methods used in Visualforce to be called via JavaScript.

Execution Log

Timestamp	Event	Details
19:54:26:001	HEAP_ALLOCATE	[520]|Bytes:48
19:54:26:001	HEAP_ALLOCATE	[139]|Bytes:6
19:54:26:001	HEAP_ALLOCATE	[EXTERNAL]|Bytes:7
19:54:26:001	METHOD_ENTRY	[1]|01p2x000005RP2e|RealOnlyApex.RealOnlyApex()
19:54:26:001	STATEMENT_EX...	[1]|
19:54:26:001	METHOD_EXIT	[1]|RealOnlyApex
19:54:26:001	STATEMENT_EX...	[5]|
19:54:26:001	STATEMENT_EX...	[7]|
19:54:26:001	HEAP_ALLOCATE	[7]|Bytes:46
19:54:26:001	HEAP_ALLOCATE	[7]|Bytes:4
19:54:26:001	HEAP_ALLOCATE	[52]|Bytes:5
19:54:26:002	HEAP_ALLOCATE	[58]|Bytes:5
19:54:26:002	HEAP_ALLOCATE	[66]|Bytes:7
19:54:26:003	SOQL_EXECUTE...	[7]|Aggregations:0|SELECT Id, Name, Name__c FROM Custom_Object__c
19:54:26:620	SOQL_EXECUTE...	[7]|Rows:80000
19:54:26:621	HEAP_ALLOCATE	[7]|Bytes:320004
19:54:26:814	HEAP_ALLOCATE	[7]|Bytes:5257788
19:54:26:814	HEAP_ALLOCATE	[7]|Bytes:320004
19:54:26:814	CODE_UNIT_FI...	VFRemote: RealOnlyApex invoke(getAuraRecords)
19:54:26:814	EXECUTION_FI...	

LIMITS

1. Visualforce Remoting Exception: Remoting response size exceeded maximum of 15 MB.

This error is not because of apex or SOQL limit, all records are successfully queried and returned, but vf remoting cannot handle data beyond 15 MB and this error is thrown in javaScript. So, to avoid this error fields queried must be limited to 2-3 which doesn't involve text area or any type of field which consumes more storage (ex: text area, rich text, images, etc).

So, if you want to get all records with just one query without keeping offset in play, this approach best serves your need.

VF remoting with @readonly annotation (multiple small chunks)

If you want to query more fields then the above mentioned methods doesn't serve your purpose. You need to divide the records that you query into small chucks. As you are calling apex from javascript it is very easy to call apex method multiple times to serve your need.

In order to query data in chunks you need to have a hook (`offset`) value, to query more records and again you are hit by one more Salesforce limit `offset` cannot be more than 2000. So what next? We need to add two fields to overcome this situation.

Metadata required: Fields:

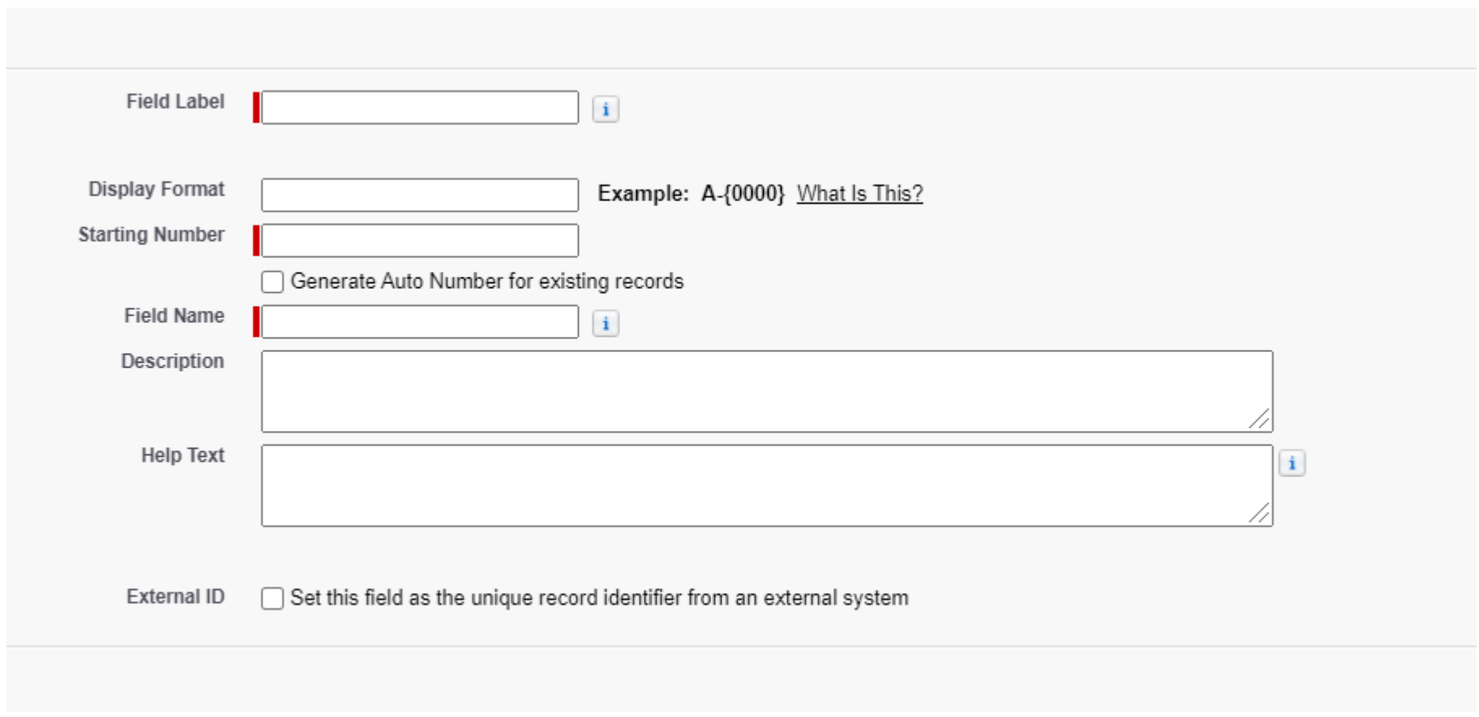
1. Auto Number
2. Formula(number)

you might have already understood what is happening, if not no worries just read ahead.

1. Create an auto number field with `{0}` this formatting and value starting from `1`.
2. Create a formula field converting auto number into its respective number as shown below.

```
VALUE(Auto number field API name)
```

Wait! what if you already have millions of existing records? That doesn't matter because when we create new Auto number fields we an option to generate auto number for existing records by selecting this checkbox `Generate Auto Number for existing records`



That's it our setup is ready.

Sample code:

```
var records = [];
    function apexCaller(varOffSet, totalRecords) {
        Visualforce.remoting.Manager.invokeAction(
            '{!$RemoteAction.RealOnlyApex.getRecords}',
            varOffSet,
            function(result){
                if(result && result.length > 0) {
                    records = records.concat(result);
                    console.log('Total Records: '+records.length);
                    publishMC(JSON.stringify(result));//using LMS to send message to Ligh
                    if(records.length < totalRecords) {
                        apexCaller(result[result.length - 1].Sequence__c, totalRecords);
                    }
                }
            }
        );
    }

    apexCaller(0, 80000);
```

SOQL Query:

```
[SELECT Id  FROM Custom_Object__c WHERE formula__c >: offSet ORDER BY formula__c ASC NULLS
```

LIMITS

I don't see and immediate limits, just try to limit the each chunk size so it does't hit 15 MB remoting limit.

Lightning controller (multiple small chunks)

All Lightning developers are very familiar on how to call apex from Lightning controller to get required data. We are going to do exactly same, but here instead of calling apex once we will be calling multiple times. Here trick is calling apex multiple times and getting data in small batches so we don't hit any limits enforced by Salesforce and get our work done, this step is same as we have done in above point `VF remoting with @readonly annotation (multiple small chunks)`.

You need to make sure you never try to retrieve more than 50000 records in single batch chunk.

Sample code:

```
auraAction : function(component, helper, offSet, totalRecords) {
        console.log("In auraAction Method");

        var action = component.get("c.getRecs");
        action.setParams({
            offSet : offSet
        });
        action.setCallback(this, function(response) {
            var state = response.getState();
            console.log("State", state);
            if(state === 'SUCCESS') {
                var result = response.getReturnValue();
                if(result && result.length > 0) {

                    var records = result;
                    helper.paginateData(component, helper, records);

                    //console.log('existingRecords: '+component.get("v.receivedRecords").l
                    var existingRecsLength = component.get("v.receivedRecords").length;

                    if(existingRecsLength < totalRecords) {
                        helper.auraAction(component, helper, result[result.length - 1].Sec
                    }
                    else {
                        component.set("v.queryCompleted", true);
                    }
                    component.set("v.totalQueryRowsReturned", component.get("v.receivedRec
                }

            }
            else {
                console.log("Error In auraAction");
            }
        });

        $A.enqueueAction(action);
```

Transfer Data to Lightning Component

If you have noticed in first two approaches we have use Visualforce to get data, so we need to transfer that data from Visualforce to Lightning Component using Lightning Message Service(LMS). Lightning Message Service lets you publish and subscribe to messages across the DOM and between Aura, Visualforce, and Lightning Web Components.

In this context you need to publish message from Visualforce and Subscribe to messages from Lightning component.

Lightning Message Channel is like any other metadata but as of now it can only be created from metadata API or using VS code. Create a file with Lightning Message Channel name and append the file name with `.messageChannel-meta.xml`.

Sample xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<LightningMessageChannel xmlns="http://soap.sforce.com/2006/04/metadata">
    <masterLabel>demoChannel</masterLabel>
    <isExposed>true</isExposed>
    <description>This Lightning Message Channel sends information from VF to LWC</descripti

    <lightningMessageFields>
        <fieldName>messageToSend</fieldName>
        <description>message To Send</description>
    </lightningMessageFields>

    <lightningMessageFields>
        <fieldName>sourceSystem</fieldName>
        <description>My source</description>
    </lightningMessageFields>

</LightningMessageChannel>
```

Publish Message:

```
// Load the MessageChannel token in a variable
    var SAMPLEMC = "{!$MessageChannel.demoChannel__c}";
    var records = 'Queried data';

    function publishMC() {
        const message = {
            messageToSend: records,
            sourceSystem: "VisualForce"
        };
        sforce.one.publish(SAMPLEMC, message);
    }
    publishMC();
```

Subscribe in AURA:

```
<lightning:messageChannel type="demoChannel__c" aura:id="sampleMessageChannel" onMessage="
```

this is how you need to transfer data from VF to AURA component.

Display Data

Now it’s up to you how you display the data. Implement pagination for better performance and user experience.

Considerations while handling Huge Data

1. Consider to stringify array before transferring.

Find full implementation of above mentioned approaches in my Github Repository.

Github Repository link