



**Making the world's
decentralised data more
accessible.**

Lab Exercise Guide

Table of Contents

Introduction	2
Pre-requisites	2
Exercise 1: Staking Reward with Aggregation	2
High level steps	2
Detailed steps	3
Step 1: Initialize your project	3
Step 2: Update the graphql schema	3
Step 3: Update the manifest file (aka project.yaml)	4
Step 4: Update the mappings file	5
Step 5: Build the project	7
Step 6: Run a query	8

Introduction

In this exercise, we will take the starter project and look at how we can aggregate data. Specifically, we will index staking rewards and then aggregate them over for a particular account. In effect we are determining how much reward an account has accumulated over time.

Pre-requisites

Completion of Module 3.

Exercise 1: Staking Reward with Aggregation

High level steps

1. Initialise the starter project
2. Update your mappings, manifest file and graphql schema file.
3. Generate, build and deploy your code
4. Deploy your code in Docker
5. Query for address balances in the playground

Detailed steps

Step 1: Initialize your project

The first step in creating a SubQuery project is to create a project with the following command:

```
~/Code/subQuery$ subql init
Project name [subql-starter]: staking-rewards
Git repository:
RPC endpoint [wss://polkadot.api.onfinality.io/public-ws]:
Authors: sa
Description:
Version: [1.0.0]:
License: [Apache-2.0]:
Init the starter package... staking-rewards is ready
```

Step 2: Update the graphql schema

Add an entity called SumStakingRewards. This has 2 simple fields that allows us to record the account reward along with the block height which will allow us to do a cross check.

```
type SumReward @entity{
  id: ID! # AccountId
  accountReward: BigInt!
  blockheight: Int!
}
```

Step 3: Update the manifest file (aka project.yaml)

Update the manifest file to only include a handleReward handler and update the filter method to Rewarded. This is the only event we want to capture for now.

```
- handler: handleReward
  kind: substrate/EventHandler
  filter:
    module: staking
    method: Rewarded
```

Note: Previously the method was “Reward” and not “Rewarded”. Here the new version is used, but if you want to capture earlier events of the chain an **additional** handler must be included (as shown below).

```
- handler: handleReward
  kind: substrate/EventHandler
  filter:
    module: staking
    method: Reward
```

The full manifest file should look like this:

```
specVersion: 0.0.1
description: ''
repository: ''
schema: ./schema.graphql
network:
  endpoint: wss://polkadot.api.onfinality.io/public-ws
  dictionary:
    https://api.subquery.network/sq/subquery/dictionary-polkadot
dataSources:
  - name: main
    kind: substrate/Runtime
    startBlock: 7000000
    mapping:
      handlers:
        - handler: handleReward
          kind: substrate/EventHandler
          filter:
            module: staking
            method: Rewarded
```

Note: We are starting at block height 7 million in this example.

Step 4: Update the mappings file

The initialisation command pre-creates a sample mappings file with 3 functions, handleBlock, handleEvent and handleCall. Delete all of them as we will create our own.

```
export async function handleReward(event: SubstrateEvent): Promise<void>
{
}
}
```

Next inside `handleReward()` we declare an event object as follows:

```
const {event: {data: [account, newReward]}} = event;
```

Then we declare a variable called `entity` and attempt to get the `account.toString()` value.

```
let entity = await SumReward.get(account.toString());
```

We then check if this is undefined. If it is, we then create a new instance of our `SumReward` entity, passing the `account` into the constructor as an ID field.

```
if (entity === undefined){  
    entity = createSumReward(account.toString());  
}
```

This is what our “`createSumReward`” helper function should look like:

```
function createSumReward(accountId: string): SumReward {  
    const entity = new SumReward(accountId);  
    entity.accountReward = BigInt(0);  
    return entity;  
}
```

Finally, we retrieve the block height, like we’ve done in previous lessons, and then get the `newReward`, adding it to the current value. We then save the entity.

```
entity.blockheight = event.block.block.header.number.toNumber();  
entity.accountReward += (newReward as Balance).toBigInt();  
  
await entity.save();
```

The full mapping file should look like this:

```
import {SubstrateExtrinsic,SubstrateEvent,SubstrateBlock} from
"@subql/types";
import {SumReward} from "../types";
import {Balance} from "@polkadot/types/interfaces";

function createSumReward(accountId: string): SumReward {
  const entity = new SumReward(accountId);
  entity.accountReward = BigInt(0);
  return entity;
}

export async function handleReward(event: SubstrateEvent): Promise<void>
{
  const {event: {data: [account, newReward]}} = event;
  let entity = await SumReward.get(account.toString());
  if(entity === undefined){
    entity = createSumReward(account.toString());
  }

  entity.blockheight = event.block.block.header.number.toNumber();
  entity.accountReward += (newReward as Balance).toBigInt();

  await entity.save();
}
```

Step 5: Build the project

Run the standard codegen, build and docker commands.

```
yarn codegen
OR
npm run-script codegen
```

```
yarn build
OR
npm run-script build
```

```
docker-compose pull && docker-compose up
```

Step 6: Run a query

Once the docker container is up and running, which could take a few minutes, open up your browser and navigate to www.localhost:3000.

This will open up a “playground” where you can create your query. Copy the example below.

```
query{
  sumRewards(first:10){
    nodes{
      id
      accountReward
      createdAt
      blockheight
    }
  }
}
```

This should return:

```
{
  "data": {
    "sumRewards": {
      "nodes": [
        {
          "id": "1122Swicp7esoyo1h2jjzKbSiQ4j18XP6fd98DSFgVjERQyC",
          "accountReward": "821480629",
          "createdAt": "2021-11-19T03:16:00.203+00:00",
          "blockheight": 7001820
        },
        {
          "id": "11233qXh9wqztw1H1FDMb6Ku3AAACnT4iVHdwcfasyKwSwrf",
          "accountReward": "7112727441",
          "createdAt": "2021-11-19T03:16:53.66+00:00",
          "blockheight": 7013965
        },
        {
          "id": "112439qDrECV8fZ5YWbQet4bx6RJJ2qfj7zX2gzyxYbr19q3",
          "accountReward": "3149346184",
          "createdAt": "2021-11-19T03:16:37.325+00:00",
          "blockheight": 7013737
        },
        {
          "id": "1124RsEfEgJEZvEq4HbtGFcpqoxnqSy79EjNZY9tzPct3AB6o",
          "accountReward": "55897389622125",
          "createdAt": "2021-11-19T03:16:33.981+00:00",
          "blockheight": 7013737
        }
      ]
    }
  }
}
```



```
    "blockheight": 7012725
  },
  {
    "id": "1124hemLbsff2pMjxLtxZmSZRfDJepFnHvy11JGgcxJowqVA",
    "accountReward": "870384601",
    "createdAt": "2021-11-19T03:16:58.421+00:00",
    "blockheight": 7013965
  }
]
}
}
```


Let's pick a specific account to cross check. Run the query below:

```
query{
  sumRewards(first:10 filter:{id: {equalTo:
    "1122SWicp7esoyo1h2jjzKbSiQ4j18XP6fd98DSFgVjERQyC"}}){
    nodes{
      id
      accountReward
      createdAt
      blockheight
    }
  }
}
```

This should return:

```
{
  "data": {
    "sumRewards": {
      "nodes": [
        {
          "id": "1122SWicp7esoyo1h2jjzKbSiQ4j18XP6fd98DSFgVjERQyC",
          "accountReward": "821480629",
          "createdAt": "2021-11-19T03:16:00.203+00:00",
          "blockheight": 7001820
        }
      ]
    }
  }
}
```

Cross checking this in Polkascan, we can see that at blockheight 7001820, the reward was 0.0821480629 DOT. This number matches the result obtained above but weren't we supposed to aggregate all the rewards?

For  1122SWicp7esoyo1h2jjzKbSiQ4j18XP6fd98DSFgVjERQyC (163)				
Block	Event ID	Time	Action	Value
7028395	7028395-181	51 days 11 hrs ago	staking(Reward)	0.1087877635 DOT
7022804	7022804-66	51 days 20 hrs ago	staking(Reward)	0.1157687632 DOT
7001820	7001820-189	53 days 7 hrs ago	staking(Reward)	0.0821480629 DOT
6985166	6985166-183	54 days 11 hrs ago	staking(Reward)	0.0753597257 DOT
6971485	6971485-171	55 days 10 hrs ago	staking(Reward)	0.0978062443 DOT

Firstly, we started indexing at blockheight 7,000,000 just for demonstration purposes, so we will ignore all records prior. Secondly, because indexing takes a bit of time, it picks up the reward at blockheight 7001820 relatively quickly, but after a few minutes, it will pick up 7022804 and so on.

Below shows the total aggregated result of the 3 records after block 7M. Summing up the values, $0.0821480629 + 0.1157687632 + 0.1087877635 = 0.3067045896$ which is what we expect.

```
{
  "data": {
    "sumRewards": {
      "nodes": [
        {
          "id": "1122SWicp7esoyo1h2jjzKbSiQ4j18XP6fd98DSFgVjERQyC",
          "accountReward": "3067045896",
          "createdAt": "2021-11-19T03:23:32.029+00:00",
          "blockheight": 7028395
        }
      ]
    }
  }
}
```