Use of Ganache-time-traveler

Ganache time traveler is feature of ganache ethereum development blockchain that allow to manipulate the block timestamp during development and testing of smart contracts. This feature is useful for testing time dependent functionality with in the contracts such as interest rate calculations, and more. this enable to "time travel" by setting the block timestamp to a specific future or past date, allow to test smart contract actions under different time conditions.

Example:-

```
// // SPDX-License-Identifier: MIT
pragma solidity ^0.8.17;
contract TimeBasedContract {
 uint private intialTime;
  constructor(uint newIntialTime) public {
    intialTime = newIntialTime;
  }
  function isNowAfter() external view returns (bool){
 uint now = block.timestamp;
      return (now >= intialTime);
  }
}
```

```
const TimeBasedContract = artifacts.require('./TimeBasedContract');
const helper = require('ganache-time-traveler');

const Thu_Aug_10_00_00_00_UTC_2023 = 1691625600;
const Wed_Sep_20_00_00_00_UTC_2023 = 1695168000;
const secondsInOneDAY = 86400;

contract('TimeBasedContract', async (accounts) =>  {
    before('deploy TimeBasedContract', async() => {
        instance_1 = await TimeBasedContract.new(Thu_Aug_10_00_00_00_UTC_2023);
        instance_2 = await TimeBasedContract.new(Wed_Sep_20_00_00_00_UTC_2023);
    });

    beforeEach(async() => {
        snapShot = await helper.takeSnapshot();
        snapshotId = snapShot['result'];
```

```
    });
    afterEach(async() => {
        await helper.revertToSnapShot(snapshotId);
    });

    it("Thu Aug 10 00:00:00 UTC 2023 (before current time)", async() => {
        var output = await instance_1.isNowAfter.call();
        assert.equal(output, true, "output should be true");
    });

    it("Wed Sep 20 00:00:00 UTC 2023 (after current time)", async() => {
        var output = await instance_2.isNowAfter.call();
        assert.equal(output, false, "output should be false");
    });

    it("Wed Sep 20 00:00:00 UTC 2023 (after current time)", async() => {
        await helper.advanceTimeAndBlock(secondsInOneDAY * 100);
        var output = await instance_2.isNowAfter.call();
        assert.equal(output, true, "output should be true");
    });
});
```