Task: to scrape normal transaction from the given link

> Link: https://bscscan.com/txs

Import required libraries

In [1]:

```
import csv
import json
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.common.by import By
# from selenium.webdriver.common.desired_capabilities import DesiredCapabilities
```

Function to launch browser and get a given link

In [2]:

```
def launch_chrome(link):
    ops = Options()
    ops.add_experimental_option("detach", True) # prevents browser from closing when
    # dc = DesiredCapabilities.CHROME
    driver = webdriver.Chrome(options=ops,
                              executable_path="C:/Users/Jay/Desktop/cODE/WebDrivers/

    driver.get(link)
    return driver
```

Scraping the data

```
XPATHs of table rows 1 to 3:

   <tr> 1 of <tbody> : "//*[@id="paywall_mask"]/table/tbody/tr[1]"

   <tr> 2 of <tbody> : "//*[@id="paywall_mask"]/table/tbody/tr[2]"

   <tr> 3 of <tbody> : "//*[@id="paywall_mask"]/table/tbody/tr[3]"

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

<td> 3 of <tr> 1 : //*[@id="paywall_mask"]/table/tbody/tr[3]/td[1]
```

Function to get headers (list)

In [3]:

```
# returns a list
def get_header(table):
    header_txt = table.find_element(By.XPATH, '//*[@id="paywall_mask"]/table/thead')
    temp = header_txt.split(sep='\n')
    # print(temp)
    temp2 = temp[0].split(' ')
    # print(temp2)
    header = [temp2[0]+' '+temp2[1]]
    header.append(temp2[2])
    header.extend(temp[1:])
    return header
```

Function to write rows into 'data.csv'

```
params:
   wr: writer object,
   table: table element found using driver.find_element()

return:
    failed_rows: list
    normal_rows: list
```

In [4]:

```
# params:
#   wr: writer object,
#   table: table element found using driver.find_element()

def write_normal_rows(wr, table):
    failed_rows = []
    normal_rows = []
    # get rows of current page
    for row_num in range(1,51):
        row_xpath = f'//*[@id="paywall_mask"]/table/tbody/tr[{row_num}]'
        #              //*[@id="paywall_mask"]/table/tbody/tr[1]
        row = table.find_element(By.XPATH, row_xpath)
        # 12 <td> tags in a row
        normal_row_content = []
        failed_row_content = []

        # check if row has span with failed icon (<i> tag)
        normal = False
        try:
            i_xpath = f'//*[@id="paywall_mask"]/table/tbody/tr[{row_num}]/td[2]/span
            i_tag = table.find_element(By.XPATH, i_xpath).tag_name

        # if failed icon or <i> tag is not found, Then exception will be raised
        # it means that transaction is normal
        except:
            normal = True

        # get text from <td> in the row:
        # <td> 1, 5, 8, 12 donot have any text
        for td_num in [2,3,4,6,7,9,10,11]:
            td_xpath = f'//*[@id="paywall_mask"]/table/tbody/tr[{row_num}]/td[{td_nu
            if normal:
                normal_row_content.append(row.find_element(By.XPATH, td_xpath).text)

            else:
                failed_row_content.append(row.find_element(By.XPATH, td_xpath).text)

        if len(failed_row_content) > 0:
            failed_rows.append(failed_row_content)

        normal_rows.append(normal_row_content)
        wr.writerow(normal_row_content)


    return normal_rows, failed_rows
```

Function to get all the transactions

```
it writes transactions into data.csv and failed_data.csv
```

In [8]:
```python
def get_transactions():

    with open('data.csv', 'w', newline='') as csvfile:
        wr = csv.writer(csvfile)

        failed_rows = []

        driver = launch_chrome(f"https://bscscan.com/txs")

        table_xpath = '//*[@id="paywall_mask"]/table'
        table = driver.find_element(By.XPATH, table_xpath)

        header = get_header(table)
        wr.writerow(header) # write the header into csv file
        print('Writing header to "data.csv" file ',header)

        for page in range(1,11):

            driver = launch_chrome(f"https://bscscan.com/txs?p={page}")

            print(f"Reading page{page}...")


            # write rows into csv file and
            # return lists of normal_rows, failed_rows
            normal_rows, failed_rows = write_normal_rows(wr, table)

            # print(failed_rows)
            # write failed rows in its csv file

            # click next to go to next page
            next = driver.find_element(By.XPATH,'//*[@id="ctl00"]/div[3]/ul/li[4]/a'
            next.click()

            driver.quit() # close driver

    print('Successfully written all normal rows into "data.csv" ')
    with open('failed_data.csv', 'w', newline='') as csvfile:
        wr = csv.writer(csvfile)
        wr.writerow(header) # write the header into csv file
        print('Writing header to \'failed_data.csv\' file ',header)

        print("Writing failed rows into the csv file...")
        [ wr.writerow(fr) for fr in failed_rows ]

    print('Successfully written all failed rows into "failed_data.csv" ')
```

get 50 transactions from a page scrape 10 such pages total 500 transactions

Note: actual number of transactions will be less than 500 as some of them will be in failed_data.csv

In [9]:
```python
get_transactions()
```

```
<ipython-input-2-58a148e52162>:5: DeprecationWarning: executable_path has been depre
cated, please pass in a Service object
  driver = webdriver.Chrome(options=ops,
Writing header to "data.csv" file  ['Txn Hash', 'Method', 'Block', 'Age', 'From', 'T
o', 'Value', 'Txn Fee']
```

```
Reading page1...
Reading page2...
Reading page3...
Reading page4...
Reading page5...
Reading page6...
Reading page7...
Reading page8...
Reading page9...
Reading page10...
Successfully written all normal rows into "data.csv"
Writing header to 'failed_data.csv' file  ['Txn Hash', 'Method', 'Block', 'Age', 'Fr
om', 'To', 'Value', 'Txn Fee']
Writing failed rows into the csv file...
Successfully written all failed rows into "failed_data.csv"
```

CSV to JSON

In [ ]:

```python
# Function to convert a CSV to JSON
# Takes the file paths as arguments
def make_json(csvFilePath, jsonFilePath):

    # create a dictionary
    data = {}

    # Open a csv reader called DictReader
    with open(csvFilePath, encoding='utf-8') as csvf:
        csvReader = csv.DictReader(csvf)

        # Convert each row into a dictionary
        # and add it to data
        for rows in csvReader:

            # Assuming a column named 'No' to
            # be the primary key
            key = rows['Txn Hash']
            data[key] = rows

    # Open a json writer, and use the json.dumps()
    # function to dump data
    with open(jsonFilePath, 'w', encoding='utf-8') as jsonf:
        jsonf.write(json.dumps(data, indent=4))
```

In [ ]:

```python
make_json('data.csv', 'data.json')
```

In [ ]:

```python
make_json('failed_data.csv', 'failed_data.json')
```