



3. Gagnagrunnur fyrir tímataka.net

timataka.py

Í seinustu viku gerðum við forritið `timataka.py` sem að tekur inn URL, skrapar gögnin þannig að þau séu á HTML formi og vistar það svo hægt skoða þau. Út frá því áttum við að búa til reglulegar segðir sem að myndu nýta sér HTML gögnin og setja þau framm á viðeigandi hátt í `.csv` skjali. Nú höfum við uppfært þetta forrit þannig að það geti unnið með fjölbreyttari gögn.

Import og Uppsetning

```
import requests
import pandas as pd
import argparse
import re
from datetime import datetime
import os
```

- **requests:** Notað til að sækja HTML innihald vefsíðna.
- **pandas:** Notað til að vinna með og vista gögn í DataFrame og CSV formi.
- **argparse:** Notað til að lesa innskipunarlínu rök.
- **re:** Notað fyrir reglulegar segðir til að vinna úr texta.
- **datetime:** Notað til að vinna með dagsetningar og tíma.
- **os:** Notað til að vinna með skráakerfið, t.d. athuga hvort skrár eða möppur eru til.

Lesað inn Rök

```
def parse_arguments():
    parser = argparse.ArgumentParser(description='Vinna með úrslit af tímataka.net.')
    parser.add_argument('--url', help='Slóð að vefsíðu með úrslitum.')
    parser.add_argument('--output_dir', default='data',
                        help='Mappa til að vista niðurstöðurnar.')
    parser.add_argument('--debug', action='store_true',
                        help='Vistar html í skrá til að skoða.')
    return parser.parse_args()
```

- Forritið tekur inn þrjú rök:
 - `--url` : Slóð á `rimataka.net`.
 - `--output_dir` : Mappa til að vista niðurstöður (sjálgefið `data`).
 - `--debug` : Ef þetta falgg er sett, vistast HTML-ið í skrá.

Sækja HTML gögnin

```
def fetch_html(url):
    response = requests.get(url)
    if response.status_code == 200:
        return response.text
    else:
        print(f"Tókst ekki að sækja gögn af {url}")
        return None
```

- Notar `request` til að sækja HTML innihaldið frá gefinni slóð
- Athugar hvort beiðnin tókst (status code 200) og skilar innihaldinu ef svo er.

Vinna út HTML gögnunum

```
def parse_html(html):
    #...
```

Þetta er aðalfallið sem að vinnur úr HTML-inu. Það notar reglulegar segðir til að sækja:

1. Töfluna með úrslitum

2. Hausana (Dálkaheitin)
3. Gögnin (raðirnar í töflunni)
4. Upplýsingar um hlaupið

1. Finna töfluna með úrslitunum

```
# Finna töfluna sem inniheldur úrslitin
table_pattern = r"<table[^>]*>(.*?)</table>"
tables = re.findall(table_pattern, html, re.DOTALL)
```

- Reglulega segðin: `<table[^>]*>(.*?)</table>`
 - `<table[^>]*>` : Passar við `<table>` tagið með öllum eiginleikum (attributes).
 - `[^>]*` : Passar við hvaða stafi sem er nema `>`, engin eða fleiri skipti.
 - `(.*?)` : Grípur innihaldið milli `<table>` og `</table>`.
 - `.*` : Non-greedy match sem passar við sem minnstan mögulegan fjölda stafa.
 - `</table>` : Passar við lok `<table>` tagsins.
- `re.DOTALL` : Gerir `.` kleift að passa líka við línubil (newline characters).
- **Útkoma**: Listi af öllum töflum í HTML-inu.

2. Velja réttu töfluna

```
# Leita að töflu sem inniheldur bæði <thead> og <tbody>
results_table = None
for table_html in tables:
    if '<thead>' in table_html and '<tbody>' in table_html:
        results_table = table_html
        break
```

- Forritið fer í gegnum allar töflur og leitar að þeirri sem inniheldur bæði `<thead>` (hausar) og `<tbody>` (gögn).
- Fyrsta taflan sem uppfyllir þetta skilyrði er valin sem úrslitatöflan.

3. Sækja hausana (Dálkana)

```
# Finna hausana úr thead
thead_pattern = r"<thead.*?>(.*?)</thead>"
thead_match = re.search(thead_pattern, results_table, re.DOTALL)
```

- Regluleg segð: `<thead.*?>(.*?)</thead>`
 - `<thead.*?>` : Passar við `<thead>` tagið með öllum eiginleikum.
 - `(.*?)` : Grípur innihaldið milli `<thead>` og `</thead>` .
- Finnur Dálkaheitin úr töflunni.

```
if thead_match:
    thead_html = thead_match.group(1)
    th_pattern = r"<th[^>]*>(.*?)</th>"
    headers = re.findall(th_pattern, thead_html, re.DOTALL)
    # Hreinsa headers
    headers = [re.sub(r"<.*?>", "", h).strip() for h in headers]
```

- Regluleg segð fyrir hausfrumur: `<th[^>]*>(.*?)</th>`
 - `<th[^>]*>` : Passar við `<th>` tagið með eiginleikum.
 - `(.*?)`: Grípur innihaldið milli `<th>` og `</th>` .
- Hreinsar öll HTML-tags úr hausunum með `re.sub(r"<.*?>", "", h)` og notar `strip()` til að fjarlægja auka bil.

4. Sækja Gögnin (raðirnar)

```
# Finna allar raðir í tbody
tbody_pattern = r"<tbody.*?>(.*?)</tbody>"
tbody_match = re.search(tbody_pattern, results_table, re.DOTALL)
```

- Regluleg segð: `<tbody.*?>(.*?)</tbody>`
 - Sama uppbygging og áður, nú með `<tbody>` taginu.
- Finnur gögnin úr töflunni.

```
tbody_html = tbody_match.group(1)
row_pattern = r"<tr[^>]*>(.*?)</tr>"
rows = re.findall(row_pattern, tbody_html, re.DOTALL)
```

- Regluleg segð fyrir raðir: `<tr[^>]*>(.*?)</tr>`
 - Passar við `<tr>` tagið og grípur innihaldið milli `<tr>` og `</tr>`.
- Fáum lista af öllum röðum í töflunni.

Sækja gildin úr hverri röð

```
for row_html in rows:
    # Sækja gögn úr <td> elementum
    td_pattern = r"<td[^>]*>(.*?)</td>"
    tds = re.findall(td_pattern, row_html, re.DOTALL)
    if tds:
        # Hreinsa HTML tags úr gögnunum
        cells = [re.sub(r"<.*?>", "", td).strip() for td in tds]
        # Búa til orðabók með hausum sem lykla ef þeir eru til
        if headers and len(headers) == len(cells):
            result = dict(zip(headers, cells))
        else:
            # Ef hausar eru ekki til staðar eða fjöldi reita passar ekki
            result = {"Column_{idx}": cell for idx, cell in enumerate(cells)}
    data.append(result)
```

- Regluleg segð fyrir reiti í röð: `<td[^>]*>(.*?)</td>`
 - Passar við `<td>` tagið og grípur innihaldið.
- Hreinsar HTML-tags úr hverju gildi og notar `strip()` til að fjarlægja bil.
- Býr til orðabók (`dict`) fyrir hverja röð þar sem lykjar eru dálkaheitin og gildi eru gögnin úr reitunum.

5. Sækja upplýsingar um hlaupið

```
# Bæta við viðbótarupplýsingum um hlaupið
race_info = {}
```

5.1. Sækja heiti hlaupsins

Forritið reynir að sækja heiti hlaupsins úr mismunandi hlutum HTML-skjalsins því að oftast en ekki heita hlaupin það sama en flokkarnir eru mismunandi. Til þess að hafa nöfnin skýrari bætast við þau heiti flokka þegar við á.

```
# 1. Sækja heiti hlaupsins úr mismunandi hlutum
race_name_parts = []

# Úr <title> taginu
title_match = re.search(r"<title>(?:TÍMATAKA:)?(.*?)</title>", html, re.DOTALL)
if title_match:
    title_text = re.sub(r"<.*?>", "", title_match.group(1)).strip()
    race_name_parts.append(title_text)

# Úr <h2> taginu
h2_match = re.search(r"<h2>(.*?)</h2>", html, re.DOTALL)
if h2_match:
    h2_text = re.sub(r"<.*?>", "", h2_match.group(1)).strip()
    race_name_parts.append(h2_text)

# Úr valinni <option> (ef til staðar)
option_match = re.search(
    r"<option[^>]*selected[^>]*(.*?)</option>", html, re.DOTALL)
if option_match:
    option_text = re.sub(r"<.*?>", "", option_match.group(1)).strip()
    race_name_parts.append(option_text)

# Úr <h3> taginu
h3_match = re.search(r"<h3>(.*?)</h3>", html, re.DOTALL)
if h3_match:
    h3_text = re.sub(r"<.*?>", "", h3_match.group(1)).strip()
    race_name_parts.append(h3_text)

# Sameina heiti hlaupsins
race_name = ' - '.join(race_name_parts)
race_info['nafn'] = race_name if race_name else 'Óþekkt hlaup'
```

- Úr `<title>` taginu:

- Segð: `<title>(?:TÍMATAKA:)?(.*?)</title>`
- `(?:TÍMATAKA:)?` : Valfrjáls passa við "TÍMATAKA:" án þess að grípa.
- `(.*?)`: Grípur innihaldið (heiti hlaupsins).
Úr `<h2>` taginu:
- Segð: `<h2>(.*?)</h2>`
- Úr valinni `<option>` :
 - Segð: `<option[>]*selected[>]*>(.*?)</option>`
 - Leitar að `<option>` með `selected` eiginleikanum.
- Úr `<h3>` taginu:
 - Segð: `<h3>(.*?)</h3>`

Heiti hlaupsins er síðan sett saman úr þessum hlutum með `-` á milli.

5.2. Sækja viðbótarupplýsingar

```
# 2. Sækja viðbótarupplýsingar úr <div> elementum
div_pattern = r'<div class="(col-xs-4 col-md-3|hidden-xs col-md-3)">\s*' \
              r'<small class="stats-label">(.*?)</small>\s*' \
              r'<h4>(.*?)</h4>\s*</div>'
divs = re.findall(div_pattern, html, re.DOTALL)
```

- Regluleg segð:
 - Passar við `<div>` með tilteknum class:
 - `col-xs-4 col-md-3` eða `hidden-xs col-md-3`.
 - Innan þess er `<small>` með `stats-label` class sem inniheldur merki (label).
 - Síðan `<h4>` sem inniheldur gildi (value).
- Niðurstaðan er listi af tuple þar sem hvert tuple inniheldur:
 - `class`, `label`, `value`.

```

details = {}
for _, label, value in divs:
    label = label.strip()
    value = value.strip()
    details[label] = value

# Bæta upplýsingum við race_info
race_info['start_time'] = details.get('Start time')
race_info['started_finished'] = details.get('Started / Finished')
race_info['percent_completed'] = details.get('% completed')
race_info['est_finish_time'] = details.get('Est. finish time')

```

Upplýsingar eru geymdar í `details` orðabók og síðan bætt við `race_info`.

5.3. Aðskilja `Started` og `Finished`

```

# 3. Aðskilja 'started' og 'finished' úr 'Started / Finished'
if 'started_finished' in race_info:
    started_finished = race_info['started_finished']
    started_finished_match = re.match(r'(\d+)\s*/\s*(\d+)', started_finished)
    if started_finished_match:
        race_info['started'] = int(started_finished_match.group(1))
        race_info['finished'] = int(started_finished_match.group(2))
    else:
        race_info['started'] = None
        race_info['finished'] = None
del race_info['started_finished']

```

- Regluleg segð: `(\d+)\s*/\s*(\d+)`
 - Grípur tvær tölur sem eru aðskildar með `/`.
- Setur fjölda sem hófu hlaupið og fjölda sem luku því í `race_info`.

5.4. Reiknar 4pphafstíma

```
# 4. Reikna 'upphaf' með því að sameina dagsetningu og 'start_time'
if race_info.get('start_time'):
    try:
        # Reyna að lesa tíma
        time_str = race_info['start_time']
        time_obj = datetime.strptime(time_str, "%H:%M").time()
        # Finna dagsetningu úr HTML
        date_pattern = r"(\d{1,2}\.\s+\w+\s+\d{4})"
        date_match = re.search(date_pattern, html)
        if date_match:
            date_str = date_match.group(1)
            try:
                date_obj = datetime.strptime(date_str, "%d. %B %Y")
            except ValueError:
                date_obj = datetime.now()
        else:
            date_obj = datetime.now()
        # Sameina dagsetningu og tíma
        datetime_obj = datetime.combine(date_obj.date(), time_obj)
        race_info['upphaf'] = datetime_obj.strftime("%Y-%m-%d %H:%M:%S")
    except ValueError:
        race_info['upphaf'] = None
else:
    race_info['upphaf'] = None
```

- Regluleg segð til að finna dagsetningu: `(\d{1,2}\.\s+\w+\s+\d{4})`
 - Passar við dagsetningu á formi `DD. Mánuður YYYY`, t.d.
`15. ágúst 2021`.
- Reiknar upphafstíma með því að sameina dagsetningu og upphafstíma hlaupsins.

5.5. Fjöldi þátttakenda

```
# 5. Fjöldi þátttakenda
race_info['fjoldi'] = race_info.get('started') or len(data)
```

Notar fjölda sem hófu hlaupið ef hann er til, annars notar fjölda lína í data.

Setja `id` fyrir hlaupið

```
# 6. Setja 'id' fyrir hlaupið
race_info_file = os.path.join('data', 'hlaup_info.csv')
if os.path.exists(race_info_file):
    existing_races = pd.read_csv(race_info_file)
    max_id = existing_races['id'].max()
    race_info['id'] = max_id + 1
else:
    race_info['id'] = 1 # Fyrsta hlaupið
```

Setur einstakt `id` fyrir hvert hlaup með því að taka hæsta gildið í `hlaup_info.csv` og bæta við 1.

5.7 Bæta `hlaup_id` við gögnin

```
# 7. Bæta 'hlaup_id' við gögnin í 'data'
for result in data:
    result['hlaup_id'] = race_info['id']
```

Bætir `hlaup_id` við hverja færslu í `data` til að tengja hana við hlaupið.

6. Skila niðurstöðum

```
return data, race_info
```

Skilar lista af þátttakendagögnum (`data`) og upplýsingum um hlaupið (`race_info`).

Vista Niðurstöður

```
def skrifa_nidurstodur(data, race_info, output_dir):
    """
    Skrifar niðurstöður í úttaksskrár.
    """
    # ...
```

- Þetta fall vistar gögnin í CSV-skrá.

- Það tryggir að ef skrárnar eru til, þá eru gögnin **bætt við** þær, þannig að við endum með aðeins tvær skrár sem innihalda öll gögnin.

Aðalforritið

```
def main():
    args = parse_arguments()

    # Athugar hvort slóðin sé í réttu formi með reglulegri segð
    url_pattern = r"^https?://(www\.)?timataka\.net/./+urslit/\?race=\d+(&cat=\w+)?(\&age=\d+)?(\&age="
    if not re.match(url_pattern, args.url):
        print("Slóðin er ekki í réttu formi frá timataka.net")
        return

    html = fetch_html(args.url)
    if not html:
        raise Exception("Ekki tókst að sækja HTML gögn, athugið URL.")

    if args.debug:
        if not os.path.exists(args.output_dir):
            os.makedirs(args.output_dir)
        html_file = os.path.join(args.output_dir, 'debug.html')
        with open(html_file, 'w') as file:
            file.write(html)
        print(f"HTML fyrir {args.url} vistað í {html_file}")

    results, race_info = parse_html(html)
    skrifa_nidurstodur(results, race_info, args.output_dir)
```

- Regluleg segð fyrir slóðina:

```
https?://(www\.)?timataka\.net/./+urslit/\?race=\d+(&cat=\w+)?(\&age=\d+)?(\&age=
```

- Segðin tryggir að slóðin sé á réttu formi, t.d. innihaldi `race`, `cat`, og aðrar mögulegar breytur.

- Athugar og vistar HTML ef `--debug` er sett.
- Keyrir `parse_html` til að fá gögnin og vistar þau með `skrifa_nidurstodur`.

Notkun

Keyrslan á forrituna virkar einungis fyrir eitt URL og getur verið keyrt á eftirfarandi hátt:

```
python3 code/timataka.py --url "https://timataka.net/snaefellsjokulshlaupid2014/urslit/?r
```

- Þetta mun sækja gögnin frá gefnu url og vista þau í `data/hlaup.csv` og `data/hlaup_info.csv`.
- Ef `--debug` er notað, verður HTML-skjalið vistað í `data/debug.html`.

data/urls.txt

Til þess að geta tekið inn öll URL-in sem voru inn á timataka.net í ágúst mánuði árið 2021, safnaði ég öllum URL slóðum í textaskjalið `data/urls.txt`.

Þar inni eru öll þau URL-in fyrir ágúst mánuð. Ég tók URL fyrir Heildarúrslit þegar hægt var en annars tók ég URL fyrir viðeigandi flokka.

agust_url.py

Þetta forrit notum við til þess að taka inn `.txt` skrá, lesa allar línurnar í því og keyra `timataka.py` fyrir allar línurnar. Við kóðuðum `timataka.py` þannig að ef það er til skrá `hlaup.csv` og `hlaup_info.csv` þá bætir hann gögnunum við skránnar sem nú þegar er til. Þanngi með því að keyra `agust_url.py` þannig að hún taki inn `data/urls.txt`, höfum við tvær skrár CSV skrár, `hlaup.csv` og `hlaup_info.csv` með öllum upplýsingum úr hlaupunum úr ágúst mánuðu frá `timataka.net`

Import og uppsetning

```
import argparse
import subprocess
```

- **argparse**: Notað til að lesa innskipunarlínu rök.
- **subprocess**: Notað til að keyra ytri forrit (í þessu tilfelli `timataka.py`) úr Python kóða.

Lesað inn rök

```
def parse_arguments():
    parser = argparse.ArgumentParser(description='Keyrir timataka.py fyrir lista af URL-inum')
    parser.add_argument('--input_file', required=True,
                        help='Slóð að .txt skrá sem inniheldur URL-in.')
    parser.add_argument('--output_dir', default='data',
                        help='Mappa til að vista niðurstöðurnar.')
    parser.add_argument('--debug', action='store_true',
                        help='Vistar html í skrá til að skoða.')
    return parser.parse_args()
```

- Forritið tekur inn þrjú rök:
 - `--input_file` : Slóð að `.txt` skrá sem inniheldur URL-in. Hvert URL er á sér línu.
 - `--output_dir` : Mappa til að vista niðurstöðurnar (sjálfgefið `data`).
 - `--debug` : Ef þetta flagg er sett, þá verður HTML-ið vistað fyrir hvert URL (gagnlegt til villuleitar).

Aðalforrit

```
def main():
    args = parse_arguments()

    # Lesað inn URL-in úr .txt skránni
    with open(args.input_file, 'r') as f:
        urls = [line.strip() for line in f if line.strip()]

    for url in urls:
        print(f"Keyri fyrir URL: {url}")
        cmd = ['python3', 'timataka.py', '--url', url, '--output_dir', args.output_dir]
        if args.debug:
            cmd.append('--debug')
        subprocess.run(cmd)

if __name__ == "__main__":
    main()
```

- Lesa inn URL-in:

- Opnar `input_file` og les inn öll URL sem eru ekki tómar línur.
- Notar for lykkju til að búa til lista af URL-um.

- `urls = [line.strip() for line in f if line.strip()]`

- Keyra `timataka.py` fyrir hvert URL

- for lykkja fyrir hvert URL

- `urls = [line.strip() for line in f if line.strip()]`

- Prentar út hvaða URL er verið að vinna með, sem gagnlegt er til að fylgjast með framvindunni.

- `print(f"Keyri fyrir URL: {url}")`

- Býr til skipun (cmd) sem keyrir `timataka.py` með viðeigandi rökum:

- `cmd = ['python3', 'timataka.py', '--url', url, '--output_d`

- `python3`: Keyrir Python3.

- `timataka.py`: Forritið sem sækir og vinnur úr gögnunum.
- `--url`: Slóðin að vefsíðunni með úrslitunum.
- `--output_dir`: Mappan þar sem niðurstöðurnar eru vistaðar.

- Ef `--debug` flaggið er sett, er því bætt við skipunina.

- `if args.debug:`
`cmd.append('--debug')`

- Notar `subprocess.run(cmd)` til að keyra skipunina.

- `subprocess.run(cmd)`

- Þetta keyrir `timataka.py` með gefnum rökum og bíður þar til það klárast áður en haldið er áfram í næsta URL.

Notkun

Keyrsla

Við getum keyrt `agust_url.py` með því að taka inn textaskjalið `data/urls.text` svona:

```
python3 agust_url.py --input_file data/urls.txt --output_dir data --debug
```

Eftir keyrslu

- Forritið mun keyra `timataka.py` fyrir hvert URL og safna gögnunum saman í `hlaup.csv` og `hlaup_info.csv` í `data/` möppunni.
- Ef `--debug` flaggið var notað, verður HTML skráin fyrir hvert hlaup vistuð sem `debug.html` (athugaðu að hún verður yfirskrifuð í hverri keyrslu).

`sql.sql`

Skráin `sql.sql` er SQL skipanaskrá sem býr til SQLite gagnagrunn með tveimur töflum, `hlaup` og `timataka`, og les inn gögnin úr CSV skrá. Við tökum inn í hana CSV skrárna sem við höfum gert setjum þær upp í töflur svo að gögnin sé skýrari.

Skýring á kóðanum

1. Búa til töflurnar

1.1 taflan `hlaup`

```
DROP TABLE IF EXISTS hlaup;  
CREATE TABLE hlaup (  
  id INTEGER PRIMARY KEY,  
  nafn TEXT,  
  start_time TIME,  
  est_finish_time TIME,  
  "started" INTEGER,  
  "finished" INTEGER,  
  percent_completed INTEGER,  
  upphaf DATETIME,  
  fjoldi INTEGER  
);
```

- `DROP TABLE IF EXISTS hlaup;` : Eyðir töflunni hlaup ef hún er til, til að tryggja að við byrjum með hreina töflu.
- `CREATE TABLE hlaup (...);` : Býr til töfluna hlaup með eftirfarandi dálkum:
 - `id` : Auðkenni hlaupsins (Primary Key).
 - `nafn` : Heiti hlaupsins.
 - `start_time` : Upphafstími hlaupsins.
 - `est_finish_time` : Áætlaður lokatími hlaupsins.
 - `"started"` : Fjöldi sem hófu hlaupið.
 - `"finished"` : Fjöldi sem luku hlaupinu.
 - `percent_completed` : Hlutfall sem luku hlaupinu.
 - `upphaf` : Dagsetning og tími upphafs hlaupsins.
 - `fjoldi` : Heildarfjöldi þátttakenda.

Taflan `timataka`

```
DROP TABLE IF EXISTS timataka;
CREATE TABLE timataka (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  hlaup_id INTEGER,
  "Rank" INTEGER,
  BIB INTEGER,
  "Name" TEXT,
  "Year" INTEGER,
  Club TEXT,
  Split TEXT,
  "Time" TIME,
  Chiptime TIME,
  Behind TEXT,
  FOREIGN KEY (hlaup_id) REFERENCES hlaup(id)
);
```

- `DROP TABLE IF EXISTS timataka;` : Eyðir töflunni timataka ef hún er til.
- `CREATE TABLE timataka (...);` : Býr til töfluna timataka með eftirfarandi dálkum:
 - `id` : Auðkenni (Primary Key með sjálfvirkri hækkun).
 - `hlaup_id` : Auðkenni hlaupsins sem þessi tímataka tilheyrir (Foreign Key vísar í hlaup(id)).
 - `"Rank"` : Röð keppanda.
 - `BIB` : Keppnisnúmer keppanda.
 - `"Name"` : Nafn keppanda.
 - `"Year"` : Fæðingarár keppanda.
 - `Club` : Félag keppanda.
 - `Split` : Millitímar eða annað.
 - `"Time"` : Lokatími keppanda.
 - `Chiptime` : Tími mældur með flögu (ef til staðar).
 - `Behind` : Tími á eftir fyrsta manni eða öðrum viðmiðum.

Innlestrarstillingar og viðskipti

```
-- Slökkva á viðskiptum til að auka innlestrarhraða
PRAGMA synchronous = OFF;
BEGIN TRANSACTION;
```

- `PRAGMA synchronous = OFF;` : Stillir SQLite til að slökkva á vissri samstillingu til að auka hraða við innlestur. Þetta getur aukið hættu á gagnatapi ef kerfið hrynur, en er ásættanlegt í þessu samhengi þar sem við getum endurskapað gagnagrunninn.
- `BEGIN TRANSACTION;` : Byrjar viðskipti (transaction) til að tryggja að allar breytingar fari í gegn eða engar.

3. Lesa inn gögn í `hlaup` töfluna

```
-- Lesa inn gögn í 'hlaup' töfluna
.mode csv
.separator ","
.import data/hlaup_info.csv hlaup_temp
```

- `.mode csv` : Setur innlestrarhaminn í CSV.
- `.separator ","` : Tilgreinir að skilin milli dálka séu komma.
- `.import data/hlaup_info.csv hlaup_temp` : Les inn gögnin úr `data/hlaup_info.csv` í tímabundna töflu `hlaup_temp`.

Flytja gögnin yfir í `hlaup` töfluna

```
-- Færa gögnin úr hlaup_temp yfir í hlaup (til að tryggja rétta dálkaröð)
INSERT INTO hlaup (id, nafn, start_time, est_finish_time, "started", "finished", percent_completed)
SELECT id, nafn, start_time, est_finish_time, "started", "finished", percent_completed, up

DROP TABLE hlaup_temp;
```

- `INSERT INTO hlaup (...) SELECT ... FROM hlaup_temp;` : Færir gögnin úr `hlaup_temp` yfir í aðaltöfluna `hlaup`, með því að tilgreina nákvæma dálkaröð.
- `DROP TABLE hlaup_temp;` : Eyðir tímabundnu töflunni `hlaup_temp`.

4. Lesa inn gögn í timataka töfluna

```
-- Lesa inn gögn í 'timataka' töfluna
.mode csv
.separator ","
.import data/hlaup.csv timataka_temp
```

- Les inn gögnin úr `data/hlaup.csv` í tímabundna töflu `timataka_temp`.

Hreinsa hauslínuna og flytja gögnin yfir

```
-- Eyða hausalínunni úr timataka_temp
DELETE FROM timataka_temp WHERE hlaup_id = 'hlaup_id';

-- Færa gögnin úr timataka_temp yfir í timataka
INSERT INTO timataka (hlaup_id, "Rank", BIB, "Name", "Year", Club, Split, "Time", Chiptime, Behind)
SELECT hlaup_id, "Rank", BIB, "Name", "Year", Club, Split, "Time", Chiptime, Behind FROM timataka_temp;

DROP TABLE timataka_temp;
```

- `DELETE FROM timataka_temp WHERE hlaup_id = 'hlaup_id';` : Eyðir hausalínunni (fyrstu línu) úr `timataka_temp` þar sem `hlaup_id` er textinn `'hlaup_id'`.
- `INSERT INTO timataka (...) SELECT ... FROM timataka_temp;` : Færir gögnin úr `timataka_temp` yfir í aðaltöfluna `timataka`, með réttri dálkaröð.
- `DROP TABLE timataka_temp;` : Eyðir tímabundnu töflunni `timataka_temp`.

5. Ljúka viðskiptum og endurstilla samstillingu

```
COMMIT;
PRAGMA synchronous = ON;
```

- `COMMIT;` : Ljúkar viðskiptum og skrifar öll gögnin í gagnagrunninn.
- `PRAGMA synchronous = ON;` : Endurstillir samstillingu í sjálfgefið gildi til að tryggja gagnöryggi framvegis.

6. Sannreyna fjölda þátttakenda

6.1. Bera saman fjölda úr báðum töflunum

```
-- Sannreyna fjölda þátttakenda í hverju hlaupi
SELECT
  h.id,
  h.nafn,
  h.fjoldi AS Fjoldi_ut_fra_hlaup_toflu,
  COUNT(r.id) AS Fjoldi_ut_fra_timataka_toflu
FROM
  hlaup h
LEFT JOIN
  timataka r ON h.id = r.hlaup_id
GROUP BY
  h.id, h.nafn, h.fjoldi;
```

- Velur `id` og `nafn` hlaupsins.
- `h.fjoldi` er fjöldi þátttakenda samkvæmt `hlaup` töflunni.
- `COUNT(r.id)` er fjöldi tímataka úr `timataka` töflunni sem tengjast því hlaupi.
- Notar `LEFT JOIN` til að tengja töflurnar á `hlaup_id`.
- Hópar eftir `h.id`, `h.nafn`, `h.fjoldi` til að fá samantekt fyrir hvert hlaup.

6.2. Athuga hvort fjöldinn stemmi

```
-- Athuga hvort fjöldi úr 'hlaup' töflunni stemmir við fjölda úr 'timataka' töflunni
SELECT
  CASE
    WHEN h.fjoldi = COUNT(r.id) THEN 'Fjöldi stemmir fyrir hlaup ' || h.nafn
    ELSE 'Fjöldi stemmir EKKI fyrir hlaup ' || h.nafn || ' (hlaup_tafli: ' || h.fjoldi
  END AS Niðurstaða
FROM
  hlaup h
LEFT JOIN
  timataka r ON h.id = r.hlaup_id
GROUP BY
  h.id, h.nafn, h.fjoldi;
```

- Notar `CASE` til að athuga hvort fjöldinn í `hlaup` og `timataka` töflunum stemmi.

- Ef fjöldinn er sá sami, skilar skilaboðum um að fjöldinn stemmi.
- Annars skilar skilaboðum um að fjöldinn stemmi ekki og gefur upp fjöldann úr báðum töflum.
- Útkoman er dálkurinn `Niðurstaða` sem inniheldur þessi skilaboð.

Notkun

Nú loksins getum við keyrt SQL skránni með SQLite. Við höfum tryggt að dálkaheitin í CSV-skránum passi við dálkaheitin í töflunum. Ef að `data/hlaup_info.csv` sem inniheldur upplýsingar um hlaupin, og `data/hlaup.csv` sem inniheldur tímatökugögn um keppendur, getum við keyrt.

1. Keyrðu SQL-skránni með SQLite

```
sqlite3 timataka.db < sql.sql
```

Keyrum þetta í terminal. Þetta mun búa til gagnagrunninn `timataka.db`, búa til töflurnar, lesa inn gögnin og keyra fyrirspurnirnar til að sannreyna fjöldann.

2. Skoða niðurstöður

2.1 Opna gagnagrunninn

- Byrjum á að opna gagnagrunninn í terminal

```
sqlite3 timataka.db
```

- Keyrðu fyrirspurnina til að skoða gögnin

```
""bash
```

```
SELECT * FROM hlaup;
```

```
SELECT * FROM timataka LIMIT 10;
```

2.2. Keyra `table.py`

- Þú getur keyrt `table.py` sem er python kóði sem ég gerði, til þess að sjá gróflega hvernig töflurnar líta úr á aðeins skýrari hátt

```
python3 table.py
```

Svona leystum við verkefni 3.