Wanhui Citizens Database Design

ETL/Database Design/ MySQL

--Ganesh Dasari

TABLE OF CONTENTS

INTRODUCTION

We have been tasked with creating a database to track and manage information about the citizens of Wanhui, a country made up of various martial sects ruled by Supreme Ruler Chen Cheng. This task comes in response to martial artists abandoning their sects and causing violence, which raises suspicions of alliances and potential threats to the Supreme Ruler. To accomplish this, we will design a database with four tables: Citizens, Sects, Alliances, and Inventory.

Back Story

Wanhui is a land filled with martial sects, alliances, and citizens who cultivate various powers. The Supreme Ruler, Chen Cheng, has noticed a troubling trend: martial artists are abandoning their sects and engaging in random acts of violence. Suspicion has arisen that these rogue martial artists may be orchestrating these acts of violence, posing a threat to the country. Additionally, some sects are growing unusually strong, leading to concerns about potential alliances aiming to overthrow the Supreme Ruler.

In response to these concerns, we have been entrusted with the task of designing a comprehensive database to track the movements and activities of citizens. The first step involved collecting data, including sect names, cultivation powers, alliance names, and citizen names.

After extracting and transforming the data, we created a MySQL database named "WANHUI" with four essential tables:

Alliances: This table contains the names of alliances that exist in Wanhui, providing insights into the groups and relationships within the country.

Sects: The sects table includes information about the martial sects present in Wanhui, along with their affiliations to alliances. This table is essential for understanding the structure of the martial world in the country.

Inventory: This table tracks the weaponry available to each sect, providing insights into their capabilities.
Citizens: The citizens table holds information about the people living in Wanhui, including their names, age, gender, cultivation powers, power ranks, and whether they are rogue martial artists. The sect master column, not included in this initial database design, will be added later.

By creating this comprehensive database, we aim to help the Supreme Ruler better understand the dynamics within Wanhui. The data will enable the authorities to track the activities of citizens, identify potential threats, and address the issues of rogue martial artists and powerful alliances more effectively.

As we proceed, we will further analyze and explore this data in MySQL Workbench, allowing us to uncover hidden patterns, detect potential threats, and maintain peace and order in Wanhui. Our journey continues as we delve deeper into the martial world of this unique land.

The project : ETL and database design.

What did I do :

- Scrape data from different websites using Selenium.
- Generate fake data using Python's library Faker.
- Transform our data to the desired format.
- Using MySQL-Python Connector, we connected to our MySQL Workbench and designed our database.
- Load our data into the database we built.
- EDA and some cleaning using SQL.
- Run a query to get relevant data and save in a xlsx file so we can connect to Tableau and build a dashboard. (Note: Using Tableau Desktop you can connect directly to the database. But since I use the community/public version, I can only connect to csv/xlsx files).

So that's about it ! I am super excited to be sharing it with you and I hope it will reach you ~

Ok now to the project itself. We will start with the absolute basic !

WEB SCRAPING

Let's import everything we need:

```
# importing the libraries we will use throughtout this mission.

from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.common.by import By

options = Options()
options.add_argument("--profile-directory=Default")
options.add_argument("--headless")
options.add_argument('--no-sandbox')
options.add_argument('--disable-dev-shm-usage')
options.add_argument('--disable-blink-features=AutomationControlled')
options.add_argument("start-maximized")
options.add_argument("--incognito")
options.add_argument("--disable-site-isolation-trials")
options.add_argument("user-agent=Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/103.0.0.0 Safari/537.36")

driver = webdriver.Chrome(service = Service(executable_path='C:/Users/armon/Downloads/chromedriver_win32/chromedriver.exe'), options=options)

from faker import Faker
import pandas as pd
import numpy as np
import time
import csv
import random
import mysql.connector
```

Selenium

Selenium is used for scraping, faker to import fake data, pandas so we can build some dataframes, numpy while inserting our data in the database to choose some randomized selection (with random as well), time to wait some seconds while scraping our data, there is one already in the selenium library but I prefer this one, csv to read data from my csv files and finally mysql connector to, duh, connect to our MySQL Workbench.

Let's get going. We will start by scraping names of sects.

```
# Now let's start scraping. First off let's get the names of sects.

def get_sect_names():

    driver.get('https://www.fantasynamegenerators.com/wuxia-sect-names.php')
    time.sleep(3)
    final = []
    load_more = driver.find_element(By.XPATH, '//*[@id="nameGen"]/input')

    for _ in range(10):   # each loop gives us 10 names so 100 names is more than enough. We didnt choose a smaller number because sometimes there are duplicates, ew.
        time.sleep(2)
        result = driver.find_element(By.XPATH , '//*[@id="result"]').get_attribute("innerText").split('\n')
        for sect in result:
            if len(sect) > 0:
                sect_name = sect.split(' ')[0]
                final.append(sect_name)

        driver.execute_script("arguments[0].click();", load_more)
    driver.close()
    return final


sects_list = get_sect_names()
sects_list = list(dict.fromkeys(sects_list)) ## to get rid of duplicates

sects_backup = pd.DataFrame(sects_list)
sects_backup.to_csv('sects_backup.csv')
```

We will the website as shown in the code, this is a snippet of how it looks:

As you can see, it lists 10 names that we can use as suitable sect names. By clicking 'Get names' you get an extra 10 and so on. Now here I should mention that sometimes it shows a name twice or thrice, so just to meet the number of sects I need , it loops 10 times, which means a top of 100 sect names. We won't be using all of that but it's better to have a bit more than we need just in case.

We then save all of our result in a csv file. Why do I do that? Just to be on the safe side. What happened was that my other function that scrapes names takes a looong long time. And the following pic will let you on the huge amount of pain that I got:

```
sects_list = get_sect_names()
sects_list = list(dict.fromkeys(sects_list)) ## to get rid of duplicates

citizens_names_list = get_citizen_names()
citizens_names_list = list(dict.fromkeys(citizens_names_list))
```

⊗ 711m 6.6s

As you can see , the code crashed after 700+ minutes, which is more than 11 hours ! Just because my laptop got disconnected from the wifi and I didn't know. 11 hours that amounted to nothing since I couldn't even get my results. So yeah, always keep a backup as a csv or whatever.

Ok, so we got our sects names. Let's now get citizens names:

```python
global backup
backup = []


def get_citizen_names():

    final = []
    driver = webdriver.Chrome(service = Service(executable_path='C:/Users/armon/Downloads/chromedriver_win32/chromedriver.exe'), options=options)
    driver.get('https://blog.reedsy.com/character-name-generator/language/mandarin-chinese/')
    time.sleep(3)
    load_more = driver.find_element(By.XPATH, '//*[@value="Generate names"]')

    for _ in range(10000):
        time.sleep(2)
        result = driver.find_elements(By.XPATH, '//*[@id="names-container"]')
        for item in result:
            names = item.find_elements(By.TAG_NAME, "h3")
            time.sleep(2)
            for name in names:
                global backup
                time.sleep(1)
                ch_name = name.get_attribute("innerText")
                time.sleep(1)
                backup.append(ch_name)
                final.append(ch_name)
                time.sleep(1)
        driver.execute_script("arguments[0].click();", load_more)

    return final


# We used whatever we could gather in the backup list, because the function takes wayyyyyyyyyy too long. and after running for 7 hrs once, I thought it collected like 30k names
but that website has a lot of duplicates and I ended up with like 8k. so yeah, we will use the backup list.

names1 = list(dict.fromkeys(backup))
names = pd.DataFrame(names1)
names.to_csv('names_backup.csv')  # Always build a habit of saving the data you gather if it's large, just in case you need some and you offline or something happened.
```

This time, the website gives us 5 names each loop, so I made it run a 10,000 times which was also useless. Because I thought I will get around 40k to 50k of names which didn't happen. Instead, It gave me a bunch but with so many duplicates that it amounted to 8K unique names. It's ok, we can work with that. Which leads me to the next step, generating fake data.

Generate fake data

```python
# We need more names so we will use faker library to extract romanized chinese names (pinyin)

fake = Faker("zh_CN")

faker_names = []

for _ in range(1000000):
    faker_names.append(fake.romanized_name())

fakes = list(dict.fromkeys(faker_names))

fakes1 = pd.DataFrame(fakes)
fakes1.to_csv('fakes_backup.csv')
```

The number of loops might scare you off, but it actually runs in less than a minute ! Again a lot of duplicates but we will use whatever we can.

Next is something I did just because. You can skip it. I also generated a bunch of Chinese names using Chinese characters just because they are closer to my heart. But didn't use as the base since I can't guarantee that everyone who sees my projects can read Chinese. That's why I stuck with the romanized ones.

```
# something I wanted to do as well, get actual chinese names and not romanized.

chinese = []

for _ in range(1000000):
    chinese.append(fake.name())

chinese_names = list(dict.fromkeys(chinese))
chinese = pd.DataFrame(chinese_names)
chinese.to_csv('Chinese_names_backup.csv')
```

Which yielded me around 43k unique names ! sweet.

That's it for the extracting our data. Now we will transform it a bit.

The names we got from scraping the website and the names we got from faker have one big difference. The order. Names obtained through scraping are in this order : surname + given_name while the faker names are the opposite. So we want to use the one with the surname being in the front and join all these names together in one place.

```
# Let's get our faker names and switch between the surname and given name

fake_pinyin = []

with open('fakes_backup.csv') as f:
    lines = [line.split(',') for line in f]
    for line in lines:
        full_name = line[1].split('\n')[0]
        split = full_name.split(' ')
        fake_pinyin.append(split[1] + " " + split[0])
```

```python
# now extract the backup names in a list too.

backup_names = []

with open('names_backup.csv') as f:
    lines = [line.split(',') for line in f]
    for line in lines:
        full_name = line[1].split('\n')[0]
        backup_names.append(full_name)
```

```python
# join together and make sure no duplicates

pinyin_names = fake_pinyin + backup_names
complete_pinyin_names = list(dict.fromkeys(pinyin_names))
```

Now we already extracted our data and transformed it a bit. What's left is to start designing a database and load them into it. That will happen in the 3rd part coming tomorrow. For now just make sure you got how the scraping, reading from a csv files works. Run it for yourselves and explore a bit. I will be creating a python file now in addition to the jupyter notebook that is already on my GitHub so you can run it once and fast. Since the notebooks are a tiny bit of a hassle.

Connecting to our MySQL Workbench and designing the database.

We will still be using python, and to be able to apply whatever we do in python to our MySQL Workbench we will use the mysql-python connector.

```
conda install -c anaconda mysql-connector-python
```

and we should be good to go. Let's set it up:

```
# Create our database.

db = mysql.connector.connect(host = "localhost", user = "root", password = "##########")

mycursor = db.cursor()

mycursor.execute("CREATE DATABASE WANHUI")

db = mysql.connector.connect(host = "localhost", user = "root", password = "##########", database = 'WANHUI')
```

As you see in the code above, we created a connection to our RDBMS by providing it's name, and the password. You should change them according to yours. Afterwards, we created a database called 'Wanhui' so it will hold all of our tables that we will create in a bit. Then in the last line we modified the database connection so it would use the database we just created.

Let's start building our tables, starting off with the easiest: the Alliance table.

```
# Let's start with the easiest table: alliances.

mycursor.execute("CREATE TABLE alliances (ID INT PRIMARY KEY AUTO_INCREMENT, alliance_name VARCHAR(100) NOT NULL) ")

vals  = ", ".join(f"('{alliance}')" for alliance in alliance_list)
mycursor.execute(f"INSERT INTO alliances (alliance_name) VALUES {vals}")
db.commit()
```

So we created a table called alliances which has only 2 columns: ID and alliance_name. You might've noticed that we made it so that the ID will auto increment on it's own which is very convenient. Then we joined all the alliances name in tuples with commas separating them so it will be possible to insert all of the values in the table in one line instead of one by one. If we go to our Workbench, it will look like this:

ID	alliance_name
1	Iron Brotherhood Alliance
2	Beast Tamers Alliance
3	Holy Lands of Flame Allinace
4	Truth Seeking Alliance
5	Heaven Trampling Alliance
6	Godly Phoenix Alliance
7	Demon Banishment Alliance
8	Death Masters Alliance
9	Etheral light Alliance
10	Northern Wall Alliance
11	Central Heavenly Alliance
12	Crimson Dragons Alliance
13	Shadow Vengeance Alliance
14	Silent Night Alliance

Ok now that we know everything is going well let's continue with our next table: sects.

```
# Now let's continue with the sects table:

mycursor.execute("CREATE TABLE sects (ID INT PRIMARY KEY AUTO_INCREMENT, sect_name VARCHAR(100), alliance_id INT)")

vals = []
for sect in sects_list:
    vals.append((sect, random.randint(1,14)))

sql = "INSERT INTO sects (sect_name, alliance_id) VALUES (%s, %s)"

mycursor.executemany(sql, vals)
db.commit()
```

Ok so this time we created the sects table. It only has 3 columns: ID, sect name and the alliance ID which tells us to which alliance does this sect belong to. We inserted the values by creating a list containing tuples that store in them the sect name and a random number between 1 to 14 (Since that's the number of alliances we have), and since it's a list, we do it with using 'executemany'. Then we commit the changes to the database using db.commit()

Bear in mind that we still need to add some columns to this table for example: the sect master, but since we don't yet have the citizens table we can't do it yet. So these changes we will make when we move to the workbench and not in python so it's more fun.

Ok, next table: Inventory.

```
mycursor.execute("CREATE TABLE inventory (ID INT PRIMARY KEY AUTO_INCREMENT, sect_id INT, swords INT,
                                arrows INT, poison INT, daggers INT, ships INT, SS_Rated_weapons INT )")

vals = []
i = 1

for sect in sects_list:
    vals.append((i, random.randint(0,5000), random.randint(0,10000), random.randint(0,1000), random.randint(0,1000) ,
                random.randint(0,50) , random.randint(0,10)))
    i = i+1

sql = "INSERT INTO inventory (sect_id, swords, arrows, poison, daggers, ships, SS_Rated_weapons) VALUES (%s, %s, %s, %s, %s, %s, %s)"

mycursor.executemany(sql, vals)
db.commit()
```

The inventory table has the following columns: ID, sect_id so we can check later the inventory of a specific sect, and the number of swords, arrows, poison, daggers, ships and SS rated weapons that each sect has. To choose how many weapons does each sect has we used the random library. Feel free to adjust the numbers that make sense to you haha.

And now for my beloved last table: Citizens.

First let's build the table:

```
# Now for the last table, citizens.

sql = """CREATE TABLE citizens(
    ID INT PRIMARY KEY AUTO_INCREMENT,
    full_name VARCHAR(100),
    age INT,
    Gender ENUM('F', 'M'),
    cultivation VARCHAR(100),
    power_rank INT,
    isRouge ENUM('Yes', 'No'),
    sect_id INT)"""

mycursor.execute(sql)
```

So we have an ID, full_name, age, gender which can be either of what we specified, cultivation powers, their rank, if they are rouge or not and finally the sect ID of the sect they belong to. Run that and we will have the table ready. Now let's populate it:

```
vals = []

weights = [0.99, 0.01]

ranks = ['1','2','3','4','5','6','7','8','9']
ranks_weight  = [0.25, 0.15, 0.20, 0.15, 0.10, 0.05, 0.06, 0.03, 0.01 ]

for name in complete_pinyin_names:
    vals.append((name, random.randint(15, 77), random.choice(['M', 'F']), random.choice(cultivation_list),
                 np.random.choice(ranks, p=ranks_weight), np.random.choice(['No', 'Yes'], p=weights), random.randint(1,30)))

sql = "INSERT INTO citizens (full_name, age, Gender, cultivation, power_rank, isRouge, sect_id) VALUES (%s, %s, %s, %s, %s, %s, %s)"

mycursor.executemany(sql, vals)
db.commit()
```

First off we have an empty list vals which we will add to it the data the we want to load into the table. Then there is weights, with values 0.99 and 0.01. What does it mean? Before when we used the random library to generate random data in a specific range, each number has an equal chance of being the one selected. But this time here, we can't for example give to each citizen equal chances of being rouge or not since that means that almost 50% of the citizens will be rouge ! so I will use the random choice function from numpy that also takes as a parameter the weights. So we will use the weights for the rouge, and then the ranks weight for the power ranks of the citizens.

One thing that you might've noticed as well. When generating numbers to represent the sect ID we chose the range 1 to 30. why? we indeed have around a 100 sect but I chose not to use all of them in order to have more citizens in each sect. We have around 9000 citizens, even more, and by choosing 30 sects we should get around 300 people per sect. If we chose more, the sects will have way lesser people and that is not fun.

Last thing I did was also create a citizens_CH table that hold my 43k citizens with the Chinese names but we won't do that here since it's extra and just for my personal choice.

EDA and some cleaning using SQL

Now go to your Workbench and everything should be done well. Here is a snippet of mine:

To start querying our database, we need to use it first, so our queries would apply to it's tables:

```
USE WANHUI;
```

Now let's take a peek at our tables:

```
-- first off let's get a sneak peek at our tables:

SELECT * FROM citizens LIMIT 20;
SELECT * FROM sects LIMIT 20;
SELECT * FROM inventory LIMIT 20;
SELECT * FROM alliances LIMIT 20;
SELECT * FROM citizens_CH LIMIT 20;
```

How does our citizens table look like for example :

Result Grid | Filter Rows: | Edit: | Export/Import:

ID	full_name	age	Gender	cultivation	power_rank	isRouge	sect_id
1	Li Gang	39	F	Light	1	No	15
2	Xiang Yan	59	M	Minds	5	No	14
3	Fu Gang	44	F	Plants	1	No	29
4	Yin Chao	23	F	Fire	3	No	29
5	Lei Guiying	61	F	Wisdom	7	No	15
6	Luo Na	55	F	telekinesis	3	No	2
7	Hu Min	21	M	Minds	2	No	5
8	Mao Li	41	M	Physical Strength	1	No	1
9	Lin Ping	63	M	Physical Strength	1	No	20
10	Wan Xia	29	F	Array	7	No	12
11	Su Yan	66	F	Minds	1	No	17
12	Zheng Lei	23	M	Minds	4	No	24
13	Duan Li	66	M	Wind	7	No	16
14	Jiang Na	49	M	Demonic	5	No	28
15	Fan Ming	76	M	Water	4	No	17
16	Jiang Juan	59	M	Array	3	No	28
17	Zhong Qi...	57	M	Sword	3	No	19
18	Ren Qiang	73	F	Water	3	No	1
19	Shen Na	42	F	Magic	2	No	16
20	Lei Yan	42	M	Medicine	1	No	5
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Next, how many rows and columns do our tables have ?

```
-- The shape of our data:

SELECT COUNT(*) AS num_rows FROM citizens;
SELECT COUNT(*) AS num_columns FROM information_schema.columns WHERE TABLE_NAME = 'citizens';

SELECT COUNT(*) AS num_rows FROM sects;
SELECT COUNT(*) AS num_columns FROM information_schema.columns WHERE TABLE_NAME = 'sects';

SELECT COUNT(*) AS num_rows FROM inventory;
SELECT COUNT(*) AS num_columns FROM information_schema.columns WHERE TABLE_NAME = 'inventory';

SELECT COUNT(*) AS num_rows FROM alliances;
SELECT COUNT(*) AS num_columns FROM information_schema.columns WHERE TABLE_NAME = 'alliances';

SELECT COUNT(*) AS num_rows FROM citizens_CH;
SELECT COUNT(*) AS num_columns FROM information_schema.columns WHERE TABLE_NAME = 'citizens_CH';
```

What is the result for the citizens_CH ? let's run it:

Result Grid

num_rows
43882

heesh we have 43k+ rows in it, and how many columns? if you run it it should be 8.

Now generally we would also check if we have null values or duplicates but since we gathered our own data, we know for a fact we don't have them so :)

I want to change something, our tables citizens and citizens_CH have the full name in one column, let's split them into surname and given name. Before doing unnecessary changes, we need to make sure we can do it:

```sql
SELECT SUBSTRING_INDEX(SUBSTRING_INDEX(full_name, ' ', 1), ' ', -1) AS surname,
       SUBSTRING_INDEX(SUBSTRING_INDEX(full_name, ' ', 2), ' ', -1) AS given_name
FROM   citizens;
```

drum rolls , does it work?

```sql
SELECT full_name, SUBSTRING_INDEX(SUBSTRING_INDEX(full_name, ' ', 1), ' ', -1) AS surname,
       SUBSTRING_INDEX(SUBSTRING_INDEX(full_name, ' ', 2), ' ', -1) AS given_name
FROM   citizens;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

full_name	surname	given_name
Li Gang	Li	Gang
Xiang Yan	Xiang	Yan
Fu Gang	Fu	Gang
Yin Chao	Yin	Chao
Lei Guiying	Lei	Guiying
Luo Na	Luo	Na
Hu Min	Hu	Min
Mao Li	Mao	Li
Lin Ping	Lin	Ping
Wan Xia	Wan	Xia
Su Yan	Su	Yan
Zheng Lei	Zheng	Lei
Duan Li	Duan	Li
Jiang Na	Jiang	Na
Fan Ming	Fan	Ming
Jiang Juan	Jiang	Juan
Zhong Qi...	Zhong	Qiang
Ren Qiang	Ren	Qiang
Shen Na	Shen	Na
Lei Yan	Lei	Yan
Liang Jun	Liang	Jun
Wan Qiang	Wan	Qiang
Li Wei	Li	Wei
Chen Juan	Chen	Juan
Duan Xiulan	Duan	Xiulan
Deng Gang	Deng	Gang

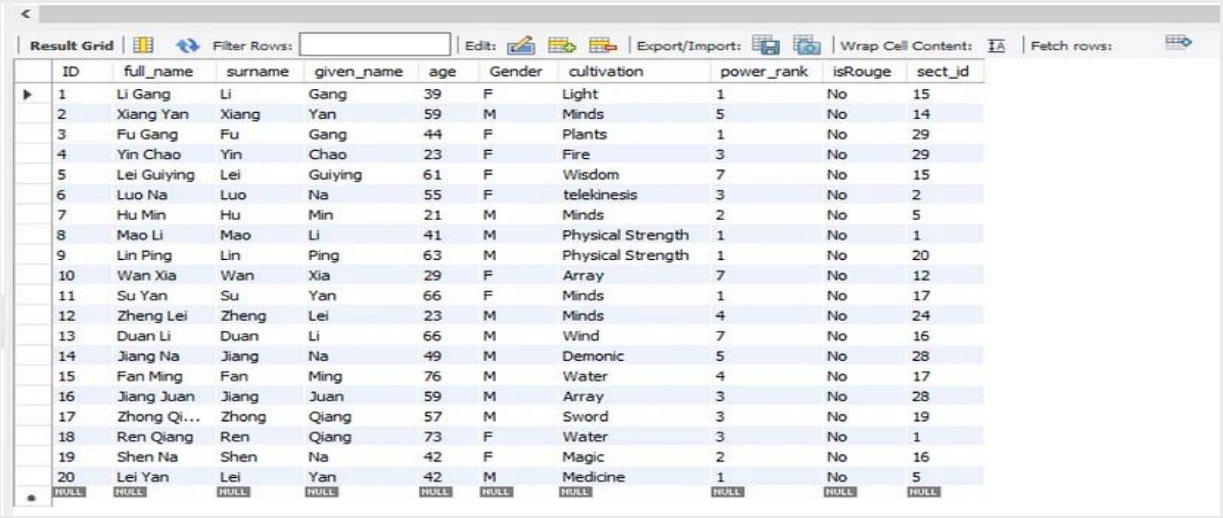
YESSS IT DOESS, noice. Now let's add this to our table.

```sql
-- after we verified that we can split the column successfully, let's assign them to their respective tables.

-- add the new columns we need to the tables:

ALTER TABLE citizens ADD surname VARCHAR(250) DEFAULT '' AFTER full_name;
ALTER TABLE citizens ADD given_name VARCHAR(250) DEFAULT '' AFTER surname;
```

So in the lines above, we added columns surname and given_name directly after the full_name column so that it looks pleasant for the eye and the default value is an empty string. It's time we populate them:

```
-- populate them with the data :

UPDATE citizens SET
surname = SUBSTRING_INDEX(SUBSTRING_INDEX(full_name, ' ', 1), ' ', -1),
given_name = SUBSTRING_INDEX(SUBSTRING_INDEX(full_name, ' ', 2), ' ', -1)
WHERE ID > 0;  # to bypass the safe mode error.
```

We will get this:



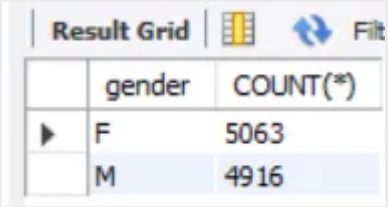
ID	full_name	surname	given_name	age	Gender	cultivation	power_rank	isRouge	sect_id
1	Li Gang	Li	Gang	39	F	Light	1	No	15
2	Xiang Yan	Xiang	Yan	59	M	Minds	5	No	14
3	Fu Gang	Fu	Gang	44	F	Plants	1	No	29
4	Yin Chao	Yin	Chao	23	F	Fire	3	No	29
5	Lei Guiying	Lei	Guiying	61	F	Wisdom	7	No	15
6	Luo Na	Luo	Na	55	F	telekinesis	3	No	2
7	Hu Min	Hu	Min	21	M	Minds	2	No	5
8	Mao Li	Mao	Li	41	M	Physical Strength	1	No	1
9	Lin Ping	Lin	Ping	63	M	Physical Strength	1	No	20
10	Wan Xia	Wan	Xia	29	F	Array	7	No	12
11	Su Yan	Su	Yan	66	F	Minds	1	No	17
12	Zheng Lei	Zheng	Lei	23	M	Minds	4	No	24
13	Duan Li	Duan	Li	66	M	Wind	7	No	16
14	Jiang Na	Jiang	Na	49	M	Demonic	5	No	28
15	Fan Ming	Fan	Ming	76	M	Water	4	No	17
16	Jiang Juan	Jiang	Juan	59	M	Array	3	No	28
17	Zhong Qi...	Zhong	Qiang	57	M	Sword	3	No	19
18	Ren Qiang	Ren	Qiang	73	F	Water	3	No	1
19	Shen Na	Shen	Na	42	F	Magic	2	No	16
20	Lei Yan	Lei	Yan	42	M	Medicine	1	No	5

Almost like a scene from the dreams.

We will continue from now on in the form of questions so that it's more fun.

```
-- Q1: What is the count of male and female martial artists ?

SELECT gender , COUNT(*) FROM citizens GROUP BY 1;
```

and this is what we get:



gender	COUNT(*)
F	5063
M	4916

```sql
-- Q2: How many alliances and sects we have?

SELECT COUNT(DISTINCT alliance_name) AS alliances_number FROM alliances;
SELECT COUNT(DISTINCT alliance_id) AS unique_alliances FROM sects WHERE ID < 31;
```

First off let's run the first query:

	alliances_number
▶	14

So we have 14 alliances, the second line is to see how many alliances is in our sects table. But why only where the sect ID is less than 31 ? if you remember when we designed the database using Python, we didn't want to use all of the 90+ sects we scraped, we only partitioned our citizens into 30 sects, proof is here:

```python
# Now for the last table, citizens.

sql = """CREATE TABLE citizens(
    ID INT PRIMARY KEY AUTO_INCREMENT,
    full_name VARCHAR(100),
    age INT,
    Gender ENUM('F', 'M'),
    cultivation VARCHAR(100),
    power_rank INT,
    isRouge ENUM('Yes', 'No'),
    sect_id INT)"""

mycursor.execute(sql)

vals = []

weights = [0.99, 0.01]

ranks = ['1','2','3','4','5','6','7','8','9']
ranks_weight  = [0.25, 0.15, 0.20, 0.15, 0.10, 0.05, 0.06, 0.03, 0.01 ]

for name in complete_pinyin_names:
    vals.append((name, random.randint(15, 77), random.choice(['M', 'F']), random.choice(cultivation_list), np.random.choice(ranks, p=ranks_weight), np.random.choice(['No',
    'Yes'], p=weights), random.randint(1,30)))

sql = "INSERT INTO citizens (full_name, age, Gender, cultivation, power_rank, isRouge, sect_id) VALUES (%s, %s, %s, %s, %s, %s, %s)"

mycursor.executemany(sql, vals)
db.commit()
```

0.9s Python

So that's why we want to know to how many alliances do our sects belong?

	unique_alliances
▶	12

You see? out of all alliances, when we chose random sects to ally with random alliances only 12 were used. There are other 2 which no sect belong to basically. Yeah, we only have 30 sects :

```
88 •  SELECT COUNT(DISTINCT sect_id) AS sect_number FROM citizens;
89
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

sect_number
30

question number 3:

```
-- Q3: What is the average age in different sects ?

SELECT sect_name, AVG(age) AS average_age
FROM citizens
JOIN secta ON citizens.sect_id = secta.ID
GROUP BY 1
ORDER BY 2 DESC;
```

and that would be :

```
93
94 •  SELECT sect_name, AVG(age) AS average_age
95    FROM citizens
96    JOIN secta ON citizens.sect_id = secta.ID
97    GROUP BY 1
98    ORDER BY 2 DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Co

sect_name	average_age
Huangyongqigong	48.7147
Hourongyumen	48.0769
Yanghehui	47.6436
Wangong	47.5548
Xiaomen	47.5217
Yunmenbang	47.3870
Liuhaihui	46.9732
Dibang	46.9601
Hepai	46.9288
Anangong	46.7313
Liuhebang	46.7234
Tengjiao	46.6717
Zuigaozhengyijiao	46.6264
Xiongweihaipai	46.5661
Hexiesenhui	46.4224
Lulianmen	46.3976
Shuoxiongjiao	46.3150
Zhonghui	46.1600
Qingiehugong	46.1012
Yuhegong	46.0363
Kaiminghougong	45.9544
Jingshajiao	45.8818
Yingdugong	45.8620
Huasenjiao	45.8171
Qingiehaijiao	45.6761
Rongyusenpai	45.5200

Wow, pretty uniformed. NEXTTT:

```sql
-- Q4: Same as question three but check average age also across genders:

SELECT sect_name, gender, AVG(age) AS average_age
FROM citizens
JOIN sects ON citizens.sect_id = sects.ID
GROUP BY 1, 2
ORDER BY 1,2,3 DESC;
```

and that would be:

```sql
108 • SELECT sect_name, gender, AVG(age) AS average_age
109   FROM citizens
110   JOIN sects ON citizens.sect_id = sects.ID
111   GROUP BY 1, 2
112   ORDER BY 1,2,3 DESC;
113
```

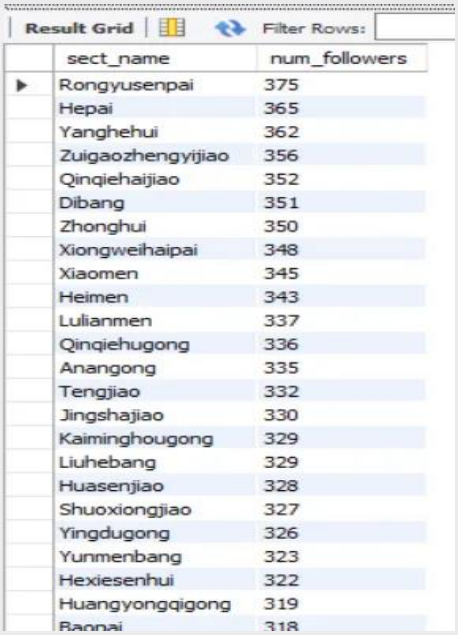
Result Grid | Filter Rows: | Export: | Wrap Cell Conte

sect_name	gender	average_age
Anangong	F	46.0059
Anangong	M	47.4699
Baopai	F	43.6447
Baopai	M	46.9759
Dibang	F	46.2696
Dibang	M	47.9184
Feidaopai	F	44.7938
Feidaopai	M	45.8387
Heimen	F	45.9076
Heimen	M	44.7547
Hepai	F	47.6099
Hepai	M	46.2514
Hexiesenhui	F	46.7099
Hexiesenhui	M	46.1313
Hourongy...	F	48.2895
Hourongy...	M	47.8571
Huangyon...	F	48.0596
Huangyon...	M	49.3036
Huasenjiao	F	45.9266
Huasenjiao	M	45.6887
Jingshajiao	F	43.8683
Jingshajiao	M	47.9448
Kaimingho...	F	46.2197
Kaimingho...	M	45.6603
Liuhaihui	F	46.4416
Liuhaihui	M	47.5379

Let's see question number 5:

```sql
-- Q5 : The count of martial artists by sects:

SELECT sect_name, COUNT(*) AS num_followers
FROM citizens
JOIN sects ON citizens.sect_id = sects.ID
GROUP BY 1 ORDER BY 2 DESC;
```

How many ?



Result Grid | Filter Rows:

sect_name	num_followers
Rongyusenpai	375
Hepai	365
Yanghehui	362
Zuigaozhengyijiao	356
Qingiehaijiao	352
Dibang	351
Zhonghui	350
Xiongweihaipai	348
Xiaomen	345
Heimen	343
Lulianmen	337
Qingiehugong	336
Anangong	335
Tengjiao	332
Jingshajiao	330
Kaiminghougong	329
Liuhebang	329
Huasenjiao	328
Shuoxiongjiao	327
Yingdugong	326
Yunmenbang	323
Hexiesenhui	322
Huangyongqigong	319
Baopai	318

Rongyue senpai ??? kinda sus.

```sql
-- Q6: which alliance has the most number of people in it and who has the least ?

SELECT alliance_name, COUNT(full_name) AS total_followers
FROM citizens
JOIN sects ON citizens.sect_id = sects.ID
JOIN alliances ON sects.alliance_id = alliances.ID
GROUP BY 1 ORDER BY 2 DESC;
```

Ok brothers and sisters which is the strongest alliance ?

alliance_name	total_followers
Beast Tamers Alliance	1676
Central Heavenly Alliance	1664
Northern Wall Alliance	1326
Silent Night Alliance	1326
Truth Seeking Alliance	678
Godly Phoenix Alliance	665
Iron Brotherhood Alliance	651
Holy Lands of Flame Allinace	651
Death Masters Alliance	365
Shadow Vengeance Alliance	329
Heaven Trampling Alliance	326
Demon Banishment Alliance	322

Nice, I knew I can count on the beast tamers. Which leads us to :

```sql
-- Q7: Which sect has the highest number of SS Rated weapons?

SELECT sect_name, SS_Rated_weapons
FROM inventory
JOIN sects ON inventory.sect_id  = sects.ID
ORDER BY 2 DESC;
```

Yeah, that's some info Chen Cheng wants to know:

Result Grid | Filter Rows:

sect_name	SS_Rated_weapons
Dadanwangguanhui	10
Zimen	10
Laogong	10
Laohui	10
Yingdugong	10
Baimen	10
Huangrongyuhui	10
Panlaobang	10
Xiaomen	9
Meilianmen	9
Yinganpai	9
Zihoamen	9
Ruancaopai	9
Tengxijiao	9
Xiongweihaipai	9
Dasenmen	9
Nianhui	9
Baopai	8
Zhonghui	8
Yanbang	8
Ruomen	8
Shuoxiongjiao	8
Yunmenbang	8
Liuhebang	8

didn't expect any less with that kinda name.

```sql
-- Q8: Which surname is the most common in all of Wanhui ?

SELECT surname, COUNT(*) FROM citizens GROUP BY 1 ORDER BY 2 DESC LIMIT 1;
```

Ok don't leave us hanging, what is it?

surname	COUNT(*)
Qin	156

```
159     -- Q9: What cultivation is the most prevalent ?
160
161 •   SELECT cultivation, COUNT(*) FROM citizens GROUP BY 1 ORDER BY 2 DESC;
162
```

….

	cultivation	COUNT(*)
▶	Earth	513
	telekinesis	511
	Array	504
	Fire	503
	Demonic	497
	Souls	495
	forest	485
	Sword	482
	Plants	478
	Darkness	474
	Wind	473
	Minds	467
	Poison	467
	Wisdom	464
	Physical ...	464
	Water	463
	Light	462
	Medicine	454
	Shadow	451
	Magic	443
	Lightning	429

Now let's continue with the pressing issue:

```
-- Q10: Which sect has the most number of rouge martial artists:

SELECT sect_name, COUNT(*) num_rouge
FROM citizens
JOIN sects ON citizens.sect_id = sects.ID
WHERE isRouge = 'Yes'
GROUP BY 1 ORDER BY 2 DESC;
```

Yeah that's the sauce we need.

	sect_name	num_rouge
▶	Qingiehaijiao	12
	Huangyongqigong	9
	Xiongweihaipai	6
	Zhonghui	6
	Xiaomen	6
	Jingshajiao	6
	Liuhebang	5
	Qiuxiaopai	5
	Liuhaihui	5
	Rongyusenpai	5
	Shuoxiongjiao	4
	Huasenjiao	4
	Wangong	4
	Feidaopai	4
	Yunmenbang	4
	Kaiminghougong	4
	Yingdugong	4
	Hexiesenhui	4
	Hepai	4
	Tengjiao	3
	Heimen	3
	Yuhegong	3
	Yanghehui	2
	Houronavumen	2

All our eyes are on you Qinqiehaijiao… I'm watching..

```
-- Q11: What is the cultivation of most of who are rouge ?

SELECT cultivation, COUNT(*) AS rouge_practicioners
FROM citizens WHERE isRouge='Yes' GROUP BY 1 ORDER BY 2 DESC;
```

	cultivation	rouge_practicioners
▶	Physical Strength	14
	Sword	10
	Wind	8
	Darkness	7
	Magic	7
	forest	7
	Array	6
	Fire	6
	Earth	6
	telekinesis	6
	Plants	5
	Water	5
	Demonic	5
	Minds	5
	Shadow	5
	Medicine	4
	Poison	4
	Light	4
	Lightning	4
	Souls	3
	Wisdom	2

whew shiver me timbers..

```
-- Q12: which sect has the most number of advanced martial artists (ranks 7-9):

SELECT sect_name , COUNT(*) AS advanced_ranks
FROM citizens
JOIN sects ON citizens.sect_id = sects.ID
WHERE citizens.power_rank BETWEEN 7 AND 9
GROUP BY 1 ORDER BY 2 DESC;
```

	sect_name	advanced_ranks
▶	Dibang	43
	Hepai	41
	Qinqiehaijiao	40
	Qinqiehugong	40
	Zhonghui	40
	Yunmenbang	40
	Hourongyumen	39
	Qiuxiaopai	38
	Xiaomen	36
	Anangong	35

Number 3 here me meet again.. too many coincidences..

Some window function should do it.

```
WITH candidates AS (
  SELECT c.sect_id, c.full_name, c.age, ROW_NUMBER() OVER (PARTITION BY sect_id ORDER BY age DESC) AS rn
  FROM citizens AS c WHERE power_rank = 9
)
SELECT * FROM candidates WHERE rn = 1;
```

does it work?

	sect_id	full_name	age	rn
▶	1	Bai Zedong	59	1
	2	Luo Shun	56	1
	3	Sun Chen	77	1
	4	He Yan	21	1
	5	Ma Guangli	75	1
	6	Ren Fan	65	1
	7	Ding Yongrui	37	1
	8	Zhou Qingge	25	1
	10	Wei Pengfei	61	1
	11	Fu Zhilan	69	1
	12	Zhu Baozhai	66	1
	13	Cao Xiaodan	67	1
	14	Ding He	73	1
	15	Chang Hualing	51	1
	16	Huang Xiaobo	39	1
	17	Lin Mei	57	1
	18	Kang Jinghua	69	1
	19	Han He	62	1
	20	Tao Jian	74	1
	21	He Xiaowen	32	1
	22	Feng Yan	71	1
	23	Yuan Shuny...	15	1
	24	Dai Meirong	72	1
	25	Su Hu	61	1

Looks like it. Let's change the tables and assign them:

```
-- Now let's add the columns to our tables:

ALTER TABLE secta ADD sect_master VARCHAR(250) DEFAULT '' AFTER sect_name;
```

```sql
-- We also want the sect_master ID to be included:

ALTER TABLE sects ADD sect_master_id INT DEFAULT 0 AFTER sect_master;
```

```sql
-- Now let's start populating these columns.

WITH candidates AS (
  SELECT c.sect_id, c.full_name, c.age,c.ID, ROW_NUMBER() OVER (PARTITION BY sect_id ORDER BY age DESC) AS rn
  FROM citizens AS c WHERE power_rank = 9
)

UPDATE sects INNER JOIN( SELECT full_name, sect_id FROM candidates  WHERE rn=1) AS result ON sects.ID = result.sect_id SET sects.sect_master = result.full_name;
UPDATE sects INNER JOIN(SELECT sect_id, ID FROM candidates  WHERE rn=1) AS result ON sects.ID = result.sect_id SET sects.sect_master_id = result.ID;
```

Ok now that we did that, how does our sects table look?

ID	sect_name	sect_master	sect_master_id	alliance_id
1	Rongyusenpai	Bai Zedong	8226	11
2	Yanghehui	Luo Shun	4146	2
3	Dibang	Sun Chen	2656	4
4	Feidaopai	He Yan	766	14
5	Hourongyumen	Ma Guangli	8751	2
6	Hexiesenhui	Ren Fan	6549	7
7	Wangong	Ding Yongrui	5381	14
8	Huasenjiao	Zhou Qingge	6125	10

Nice looking.

```sql
-- Let's see if some of the sect_masters are rouge (either left the sect or are basically traitors) since we forogt to mention that earlier :/
-- Should've added that condition to the candidates CTE.

SELECT COUNT(*) FROM citizens
JOIN sects ON citizens.ID = sects.sect_master_id
WHERE isRouge = 'Yes';
```

COUNT(*)
0

Perfect. No rouge sect master. Now let's change their ranks in the citizens table to 10:

```sql
-- What's left now is to change their ranks to 10 !

UPDATE citizens
SET power_rank = 10 WHERE ID IN (SELECT sect_master_ID FROM sects);
```

We are nearing the end, or so we thought !

```
-- !!!! BREAKING NEWS !!!!

-- All rouge martial artists joined hands and chose to follow a mysterious man who is from another country
-- The Supreme Ruler we work for asked us to make the necessary adjustments to our database.
-- It's rumored that this man's name is 'Armonia', 23 years old with 'Demonic' cultivation probably.
-- He is announcing that this new sect shall be called 'Necro' because of his ability of controlling the dead
-- So we should add that sect to our tables, and also add this suspicious man as a "citizen".
```

Ok, let's add him as a citizen first:

```
INSERT INTO citizens(full_name, surname, given_name, age, Gender, cultivation, power_rank, isRouge, sect_id)
VALUES('Armonia', '', '', 23, 'M', 'Necromancy', 10, 'No', 100); # we already have 99 sects so his will be the 100th.
```

```
INSERT INTO sects(sect_name, sect_master, sect_master_id)
VALUES('Necro', 'Armonia', 9980);
```

We know that we have 9979 citizens so his ID will be 9980. Now that we did add him as a citizen and a sect master, let's make all rouge people follow his sect.

```
UPDATE citizens SET sect_id = 100 WHERE isRouge='Yes';
```

Ok, let's give him some weapons:

```
INSERT INTO inventory(sect_id, swords, arrows, poison, daggers, ships, SS_Rated_weapons)
VALUES(100, 500, 723, 2, 125, 1, 2);
```

let's also create an alliance that his only sect belongs to, since he seeks to expand :

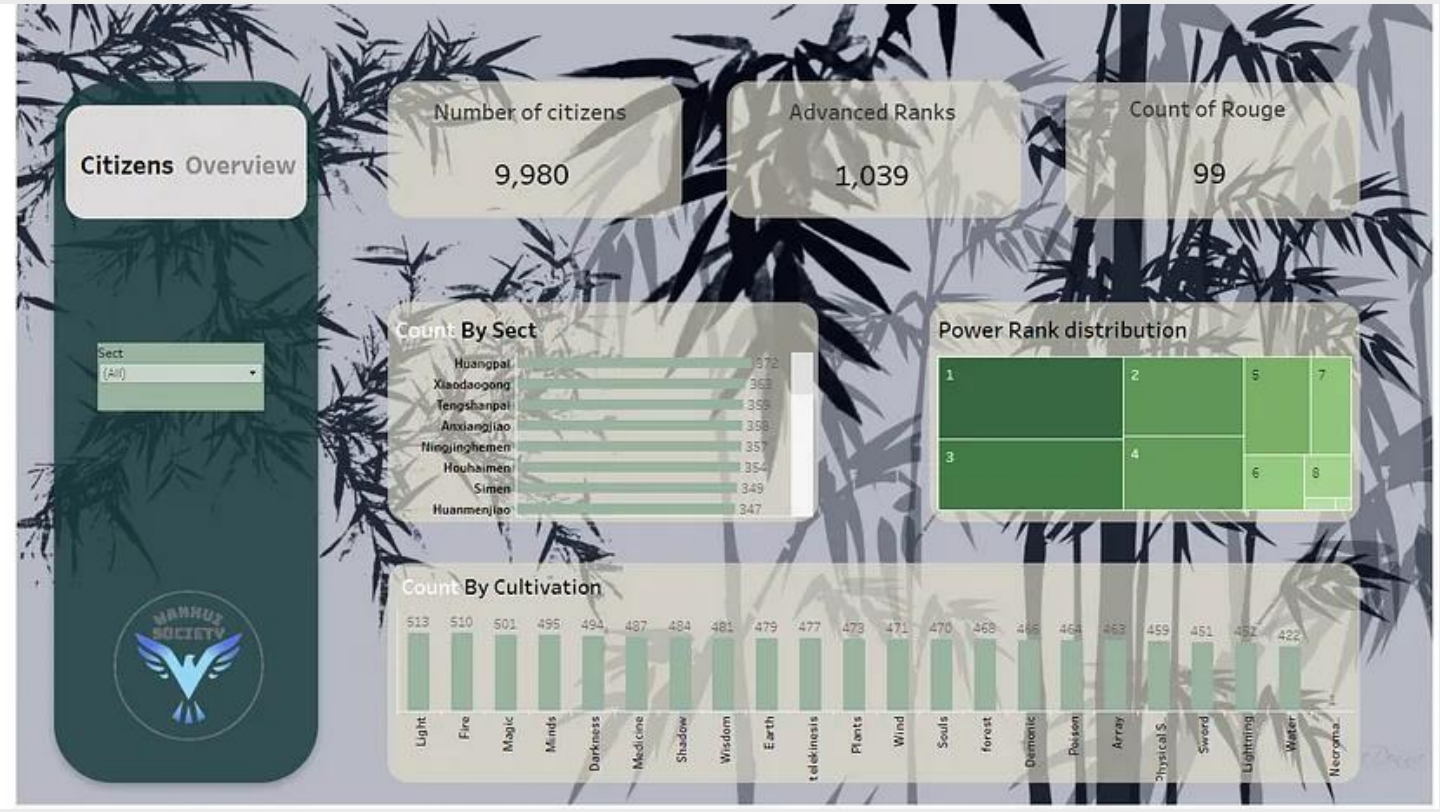
```
-- Let's make an alliance and add only his sect to it.

INSERT INTO alliances(alliance_name) VALUES ('SOLO');

UPDATE sects SET alliance_id = 15 WHERE ID = 100;
```

That's it for today. Maybe next time we can conduct wars between sects and make it even more fun but not today. Last thing I want to do is choose the data I want and export it an external txt file and from there load into excel and build a dashboard. That's how we do it:

```
SELECT c.ID, c.full_name, c.surname,c.sect_id, c.age, c.gender, c.cultivation, c.power_rank, c.isRouge, s.sect_name, a.alliance_name
FROM citizens c
JOIN sects s ON c.sect_id = s.ID
JOIN alliances a ON s.alliance_id = a.ID
INTO OUTFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/Wanhui.txt'
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n';
```

Connecting to Tableau and build a dashboard

The final file is already on my Github and you can go straight to building a dashboard.



THANK YOU