Thinking Process of the Kingdom Planning Problem:

After reading the question, it gives me a few impressions:

* It’s a SQL question with two tables involve;
* The first question is fairly straight forward;
* The last two questions are asking the same thing but from two aspects;
* Overlaps? What if two fiefdoms have overlaps? : can’t calculate since there is no data support this circumstances so I decided not to consider this situation.

Solving tool: Online SQL compiler: https://sqliteonline.com/

Since I don’t have a single clue what the table is about, I tried to turn it into a graphic (table one):

A close up of a logo

Description automatically generated

For the first question:

• How many people live in each fiefdom.

The answer is:

SELECT \* FROM fiefdoms

Which should give table fiefdoms as a result:

A screenshot of a cell phone

Description automatically generated

For the second and third question:

• How many people are within 3 days travel of the fiefdom.

• How many people are greater than 3 days travel of the fiefdom.

With the help from the above graph, let’s break down the problem:

1. Get one fiefdom. (eg. Ironforge) Merge two table into one based on fiefdom\_fiefdom\_fraction.surrounding\_fiefdom

SQL:

SELECT fiefdoms.population, fiefdoms.name, fiefdom\_fiefdom\_fraction.fiefdom, fiefdom\_fiefdom\_fraction.in\_3d

FROM fiefdom\_fiefdom\_fraction

INNER JOIN fiefdoms

ON fiefdom\_fiefdom\_fraction.surrounding\_fiefdom = fiefdoms.name

WHERE fiefdom == "Ironforge";

Result:

A screenshot of a cell phone

Description automatically generated

1. Calculate surrounding fiefdom’s in\_3d\_population based on in\_3d and population and then summarise the population

SQL:

SELECT fiefdoms.population,

fiefdoms.name,

fiefdom\_fiefdom\_fraction.fiefdom,

fiefdom\_fiefdom\_fraction.in\_3d,

sum(fiefdom\_fiefdom\_fraction.in\_3d \* fiefdoms.population)

AS pop\_in\_3d

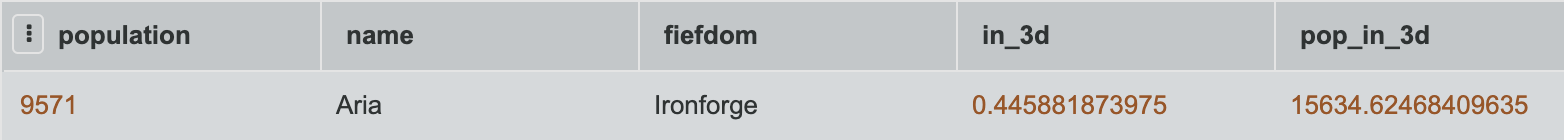
FROM fiefdom\_fiefdom\_fraction

INNER JOIN fiefdoms

on fiefdom\_fiefdom\_fraction.surrounding\_fiefdom = fiefdoms.name

WHERE fiefdom == "Ironforge";

Result: pop\_in\_3d



1. Then it needs to be populated into all fiefdoms, so we interchange WHERE clause into GROUP BY

SELECT fiefdom\_fiefdom\_fraction.fiefdom,

sum(fiefdom\_fiefdom\_fraction.in\_3d \* fiefdoms.population) AS pop\_in\_3d,

sum(fiefdom\_fiefdom\_fraction.out\_3d \* fiefdoms.population) AS pop\_out\_3d

FROM fiefdom\_fiefdom\_fraction

INNER JOIN fiefdoms

ON fiefdom\_fiefdom\_fraction.surrounding\_fiefdom = fiefdoms.name

GROUP by fiefdom

A screenshot of a cell phone

Description automatically generated

1. However, the in\_3d\_population does not include the fiefdoms themselves, so we need to add themselves to the table(At this point, I was stuck and asked my friend how to modify a SELECT statement’s result, which he introduced me “subquery”), also at this stage I realise the pop\_out\_3d is incorrect since the way we calculate it now only counts the population which surrounds the fiefdom and it does not include fiefdoms outside. (the N in the previous drawing)

SQL:

Correct pop\_in\_3d, incorrect pop\_out\_3d:

SELECT fiefdom, pop\_in\_3d + fiefdoms.population, pop\_out\_3d From (SELECT

fiefdom\_fiefdom\_fraction.fiefdom,

sum(fiefdom\_fiefdom\_fraction.in\_3d \* fiefdoms.population) AS pop\_in\_3d,

~~sum(fiefdom\_fiefdom\_fraction.out\_3d \* fiefdoms.population) AS pop\_out\_3d~~

FROM fiefdom\_fiefdom\_fraction

INNER JOIN fiefdoms

on fiefdom\_fiefdom\_fraction.surrounding\_fiefdom = fiefdoms.name

GROUP by fiefdom) INNER join fiefdoms ON fiefdom = fiefdoms.name

Result:

A screenshot of a cell phone

Description automatically generated

1. For the rest population, I decided to abandon relying on the percentage since its inaccurate, instead I decided to use Total\_Population to minus the population inside the circle of 3d and after a long testing and correction process, here is the final answer:

Final Result:

SELECT fiefdom,

(SELECT SUM(fiefdoms.population) as totalAmount FROM fiefdoms) As TotalPopulation,

pop\_in\_3d + fiefdoms.population As pop\_in\_3d\_result,

((SELECT SUM(fiefdoms.population) as totalAmount FROM fiefdoms) - pop\_in\_3d - fiefdoms.population) As pop\_out\_3d\_result

From (SELECT fiefdom\_fiefdom\_fraction.fiefdom,

sum(fiefdom\_fiefdom\_fraction.in\_3d \* fiefdoms.population) AS pop\_in\_3d

FROM fiefdom\_fiefdom\_fraction

INNER JOIN fiefdoms

on fiefdom\_fiefdom\_fraction.surrounding\_fiefdom = fiefdoms.name

GROUP by fiefdom) as fiefdom

INNER join fiefdoms ON fiefdom = fiefdoms.name

Result:

A screenshot of a cell phone

Description automatically generated

Thoughts after:

* SQL is very different from other programming language.
* The question is much harder than I thought (which proves the estimation in Agile methodology will be inaccurate from time to time).
* To some how, its hard to change thoughts into actual SQL, the thinking process of the question is quite straight forward but by using SQL is a different story.
* SQL can do a lot of stuff which could relief the load from backend.