Term Project 1 - Peaks

A. BOGNAR

Introduction, File Structure

Tidy Tuesdays is a great practice project that was designed mainly for R. However, it contains several relational databases, one of which is the **Himalayan Climbing Expeditions**. The data for it was collected from 1905 to Spring 2019. To see further details check the original source.

In my project, I focused on building a data workflow similar to the one learned during the course.

To run the project all one has to do is:

1. Download the 3 **Clean csv files** and save them in the MySQL Upload Folder.
2. Run the SQL files in the following order:
    - Loading. Note that this is the only file where codes have to be changed.
    - Cleaning
    - Analytical Layer and Triggers
    - Views

Within the *Loading* file, the path of the downloaded csv files have to be changed to the local computer directory. Afterwards, the other 3 SQL files can be executed without changes. They contain the necessary and some supplementary code and comments for clarification. For *Cleaning* a separate SQL file explains the data exploration and the testing of the code. Additionally, the EER diagram as well as the raw files can be downloaded. Lastly, there are two files related to cleaning: Multi_leads_id.csv and multi_leads.txt. These are not necessary and there purpose is explained below.

Data and Analytics Questions

The data contains 3 tables:

1. Peaks, with 468 peak points of the Himalayas.
2. Expeditions, 10364 lines about individual groups attempting to climb the peaks.
3. Members, 76519 individuals taking part in the expeditions.

My Attempt is to answer the following questions at the end of the project:

- Within the first successful expeditions with known leaders that reached the heights:
    - What were the highest peaks ever reached?
    - What were the most recent achievements?
    - How many casualties were suffered?
    - Which countries lead the successful expeditions?

Clean files, Loading and Stored Procedures

Data Loading

To answer these questions, first data has to be loaded and prepared. Accordingly I confirmed the requirements of the data columns by declaring the data types of each variable. These can be seen in the below Model:

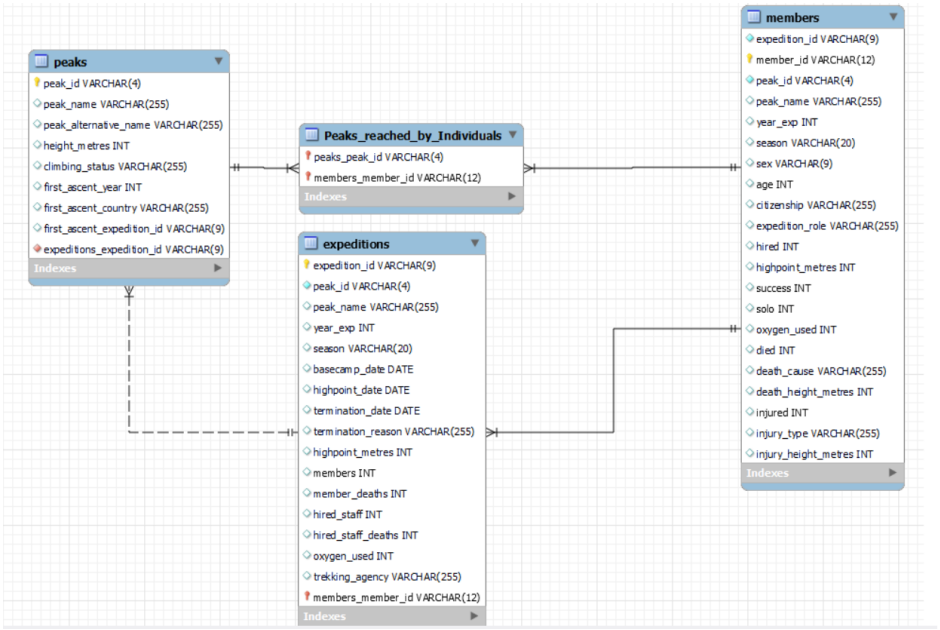

Figure 1: EER Diagram

Notably:

- Each table has its own unique ID which serves as primary key.
- Each table can be connected to another through the IDs.
- By design there are two 1:n relationships: 1 peak can only be reached first by 1 expedition; 1 member_id can only lead to 1 expedition and thus to 1 peak.

Unfortunately some observations violated these data type requirements. Therefore, the following changes were made between Raw and Clean csv files to allow loading:

- Date format changed to SQL readable.
- Commas removed from text fields for column separation.
- 2 Duplicate Primary keys were found: one in *Expeditions* (KANG10101), one in *Members* (KANG10101). In both cases the more recent ID was kept since the old ones were incomplete.

Only the ID columns were set up to not allow null values. In many other cases NA values were provided. These had to standardized during the loading process. Based on tidy standards: for numbers missing value were changed to 9999999, text values to *Unknown*, dates to 1900-01-01. Additionally binary values were changed from TRUE/FALSE to 1/0.

No NULL values were included in the tables. All missing values were originally NA.

```
SET
basecamp_date = IF(@basecamp_date = 'NA' , '1900-01-01' , @basecamp_date),
highpoint_date = IF(@highpoint_date = 'NA' , '1900-01-01' , @highpoint_date),
termination_date = IF(@termination_date = 'NA' , '1900-01-01' , @termination_date),
highpoint_metres = IF(@highpoint_metres = 'NA' , 9999999 , @highpoint_metres),
oxygen_used = IF(@oxygen_used = 'TRUE' , 1 , 0),
trekking_agency = IF(@trekking_agency = 'NA' , 'Unknown' , @trekking_agency);
```

Figure 2: Standardizing missing values

Data Cleaning

After the data was loaded, I started to explore it and found the following issues regarding my analytic goals:

1. Expeditions often had multiple countries working together to reach a height. It can't easily be seen how many countries are in 1 field.
2. Member roles were not standardized. In order to select the leader of successful groups, I needed to reduce number of leaders to 1 wherever possible.

Accordingly, I created stored procedures to solve the two issues. For country names I had to make 2 steps:

1. Select countries that have space in their names (e.g. S Korea) and replace them with underscore.
2. Wherever multiple countries are involved separate them by slash (e.g. Japan/S_Korea).

After execution, one can easily tell how many countries are involved in any expedition.

For member roles, the selection of leaders had more layers:

1. Find roles including the word 'Leader'.
2. Check which of them are primary leaders in their groups. In these cases change the role to just Leader.
3. Find groups that have multiple leaders. In these cases change all to 'Co-Leaders'.

In point 3. I decided to only include expeditions where one single leader can be assigned. I could have decided to choose one of the Co-Leaders and declare them Leader or First Lead and this would have given me slightly more results in my final table. However, this is a common occurrence, with already hundreds of Co-leaders and I had no tangible information to choose a main leader.

One technical detail: to select the member_ids of Co-Leaders under *Leader* role, I used a nested select query. However, this could not be added to stored procedure due to performance reasons. Instead, I copied and transpose pasted the values into this csv file. Then with the below Bash code, I saved them to this .txt and copied back to the query.

```
cat Multi_leads_id.csv > multi_leads.txt
```

After these changes the tables are sufficiently cleared for operations. Next, is the design of analytic layer.

Analytical Layer, ETL, Trigger

As mentioned for my analysis I chose Peaks that:

1. Were already climbed.

2. We know the expedition that climbed them.
3. We know the main leader of the successful expedition.

Therefore, I created the below Analytic Layer including all relevant variables.

Figure 3: Analytic Layer

ETL

Here, I also made the same corrections to Leader Origin countries as previously to expedition countries. In addition, I designed a trigger to complete my ETL pipeline.

The trigger activates whenever a peak is updated. Specifically, if a previously unclimbed peak is now reached and an expedition with a known leader is provided, the record will be moved to the analytical layer. As a test exercise, I found a peak *Dhaulagiri*, which according to this source was already climbed in 1960 but was unclimbed in the database. I updated it with the relevant information and it is now included in the analytics.

Therefore, the ETL parts are:

- **Extract** by table joins.
- **Transform** by the leader country
- **Loading** by updating the layer.

Data Marts, Views

Finally the dynamically updating layer can be used to answer my questions. Therefore, I created 4 views, each for 1 question and I also looked at some supplementary information such as grouping heights of the peaks or the average age of successful leaders by country of origin.

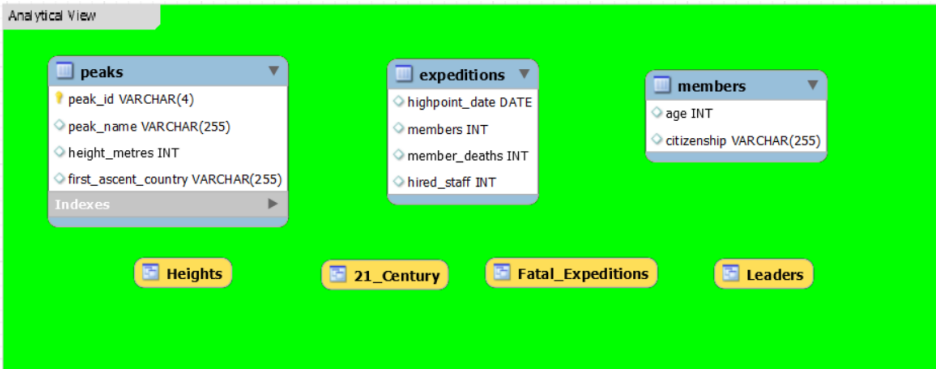

Figure 4: Views related to the Analytic Layer

Conclusions

In conclusion, my answers to the research questions are:

- Everest with 8850 meters is the highest peak reached with a known leader.
- On 2019-04-28, Sano Kailash (6452 m) was reached by an Austria-Nepalese collaboration.
- A total of 11 expedition deaths were suffered to reach these heights.
- Exactly 100 successful missions were lead by Japan solely.