CIS 556 Final Project

Shang Shi

As I described in the proposal, this is a type 2 project from a personal project. My personal project has no backend nor database function yet. So I designed DB and come up few interesting queries that is useful for my personal website which is <https://powder-for-less.herokuapp.com/>.

There is not anything similar to my idea about ski traveling website on the market. But if I break down the problem that I’m trying to resolve, there are many services out there. So my main purpose is combining something that already exist and derive a brand new service out of it.

So, This DB design helps user to find the best valued ski trip by gathering real time real data from live APIs. Also help (recommend) user to determine which season pass to buy if they want to travel with their colleagues (it could be any criteria, this is just an example).

There are a lot of unused attributes in the user\_table, because When I was designing I thought I could make sure of those attributes but I didn’t end up using them. I still think keeping them in the design is better for the long run.

The information about resorts and season passes are all real data from the IKON, EPIC, MTC (these are the names of the season passes) official websites. These three passes are the most popular multiple resorts season passes for skiers who do not live near mountains. Each pass includes multiple days of access at many different resorts which is perfect for people like me who only ski couple trips a year. The list of resorts might have overlap among different passes. The distance of the resorts to the airports is an estimate from google map. Rest of the data is fake data but similar to reality. Just from demo purposes.

There are a lot of interesting problems I can derive from this Database Design. but due to the time limit I only think of two main problems so far. I will keep working on this website as well as the DB in the future to resolve more problems related to skiing because it is my passion of life.

So my DB design for this project is shown in figure 1. The link is <https://lucid.app/lucidchart/ea50845c-d97c-4377-8dac-c22570560f33/edit?page=0_0#?folder_id=home&browser=icon>.

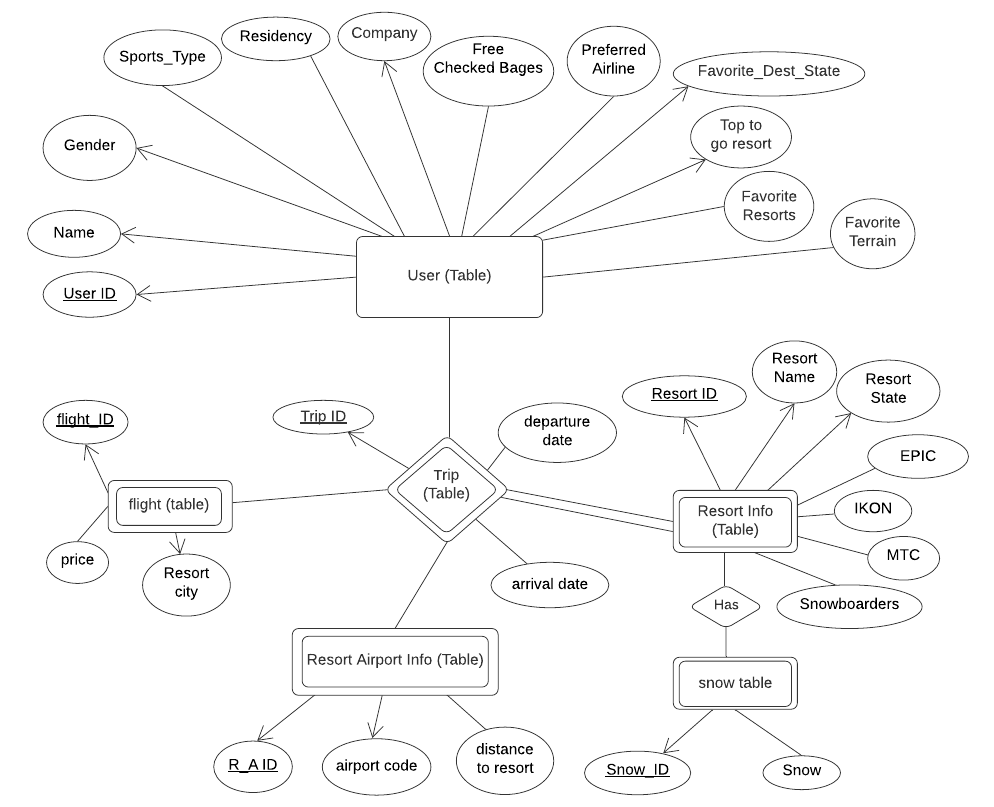


Figure 1. ERD Design for this DB

There were three major redesigns for my DB design and ERD, many small changes for my SQL. Besides the files on canvas, they can all be found on the Github repo. Everything is in the V2 branch, v2 folder. Please check for the latest commits if you want to see it. <https://github.com/Stoneshishang/CIS566-DB-Final-Class-DB-design.git>

After the DB design is final, I generated the fake data with excel. Some of the data I manually typed in, but most of the data I used RANDOMBETWEEN=() and VLOOKUP() in excel for the data generation, in order to make them realistic even though they are fake. The xlsx file in Github repo has some of the trivial data function implementation. Most of the implementation is over wrote by the pure data without function records because the RANDOMBETWEEN=() will change the number whenever I click on other cells. It disrupts the integrity of the fake data.

After converting all the xlsx to CSV. Please download them for the Table Creation. Initially I used INSERT for every single record from an online excel to sql conversion tool. I soon find it very tedious and unrealistic. Then I find out that I should use PostgresSQL COPY function. So, please replace the path to the local computer when validating the query. I have tested my queries and tables thoroughly; they should all work as expected.

So the main problem I’m trying to solve for my personal project are (they are a bit ideal but I can always extend more realistic problem out of these ideas):

1. Help user to identify the best value trip destination for skiing when snowstorm arrives.
2. Possibility helping user to determine which season pass to buy if they do not own one.

All the queries are as generic as possible without hardcoded conditions and variables so I can easily scale them up and make them work for a different data sets, as long as the data sets follow the given DB schema. Detailed query requirements and implementation please see the attached txt file. I have some comments in the txt file for future references, the comments are also helpful to understand my thought process.

Below figures are the query results from pgAdmin.

--Query 1: Assume the trip generated by the website is only to the west side of US in the trip table at this time,

--Find the top 5 resorts that has the highest snow/price ratio, (meaning heavy snow storm is coming but the airfare is still cheap so skier can ski powder without paying a lot on the flight)

--meanwhile the travel time from airport to resorts are as short as possible (for the easy of transportation from airport to ski resorts).

--Assume snow is the first priority, price is the second, distance is the third.

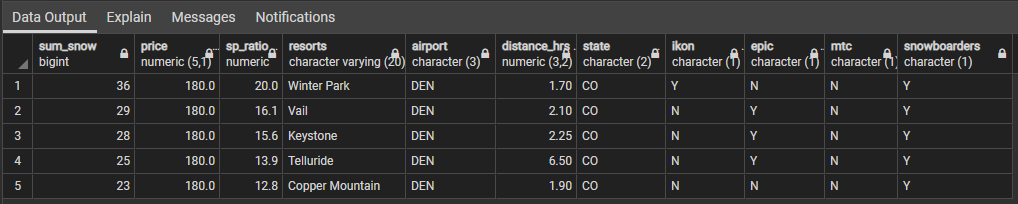


Figure . Query 1 result

--Query 2: Find the skiers/snowboarders' info who could go to the resort that has max sp\_ratio, and the resort info.



Figure . Query 2 Result

--Query 3: Find out who doesn't own a pass, call them no\_pass\_people.

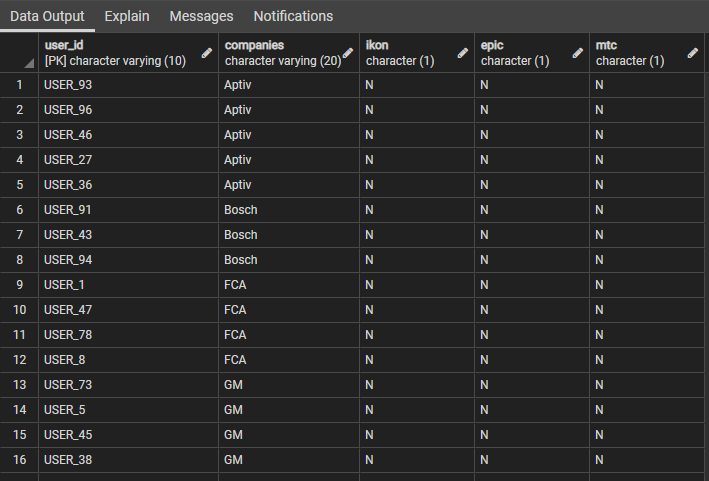


Figure . Query 3 Result

--Query 4: Recommand a pass to the no pass owner, based on the no\_pass\_people's colleague's most owned pass.

--In another word, find which pass is the most owned in no\_pass\_people's colleague, recommend them the most owned pass.

The result doesn’t include anyone who works at ford because in the DB all the ford employee owns a season.

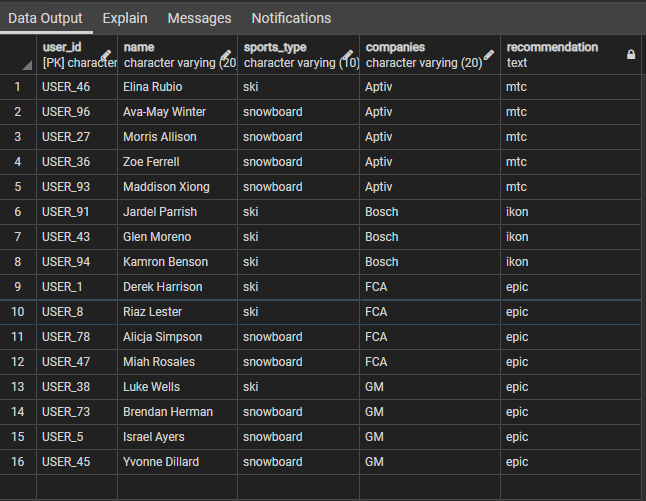


Figure . Query 4 Result

Conclusion:

I spent many hours into this project because I want to make it more realistic so that I can implement it on my own project in the future. This is the first time I experience the entire DB design to implementation process. Countless valuable lessons are learned from this project. I have a solid fundamental on DB now.