# **INFO 6210**

# **Data Management and Database Design**

# **Physical Data Model and Normalization**

# **Assignment 2**

# **Report**

## Abstract

In this assignment, we used the data received from the API of thesneakerdatabase.com. We recollected the data from the API and use json to form the database and audited those data, delete all null one and then normalized the database into Third Norm Form. Besides, using we drew the UML model graph, conceptual model graph and physical model graph based on the interrelationship of entities and finally answered all given questions.

## 1.Data Sources

### Web API

In this assignment, we choose to use the API called **Sneaker Database API.** This API is provided by thesneakerdatabase.com. It is one of the most popular websites that provides information for sneaker lovers with its well-covered sneakers data. The latest dataset we received has been stored as a Jason dataset, and will be used in further auditing process.

Here follows a screenshot of the dataset we got:



Figure 1-1-1 Screenshot of Data Gathered from Web API

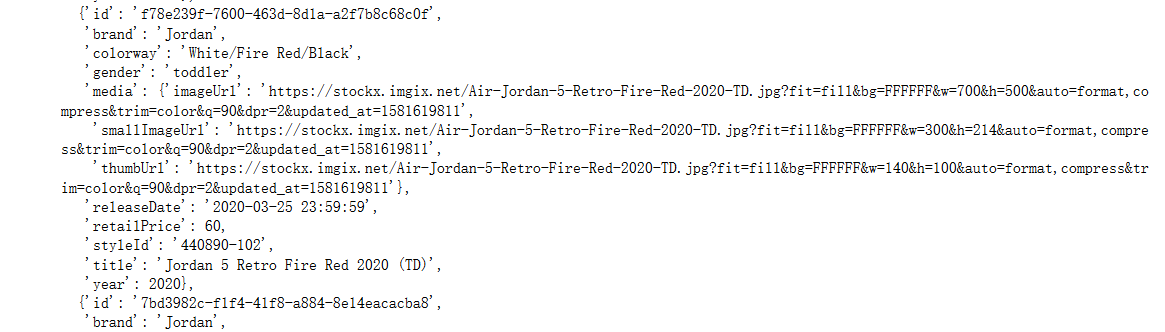


Figure 1-1-2 Screenshot of Data Gathered from Web API

Then we set up name for columns, reformed the json data into a more readable version.

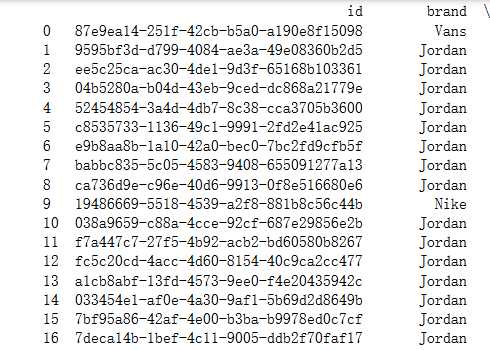


Figure 1-1-3 Screenshot of Data Gathered from Web API

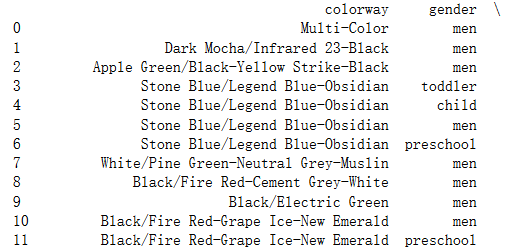


Figure 1-1-4 Screenshot of Data Gathered from Web API

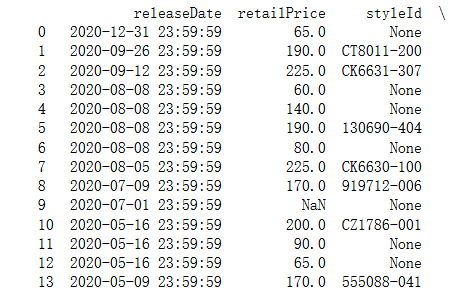


Figure 1-1-5 Screenshot of Data Gathered from Web API

## 2.Data Auditing

### Audit validity/ accuracy

In this part, we will audit the validity of our data. The main process is as below:

1.Check whether there are any null or duplicates

2.Delete the null or duplicated data

In the first version of the rawSneakerJsonData table, using *.isnull* command, we spotted several null values. Among those 49 rows of data.

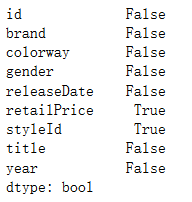


Figure 2-1-1 screenshot of Data validity auditing process and result

We then cleaned all those data with any null values and at last, 25 rows of data remained. Because currently there are no duplicates and null values, the database is already cleaned.

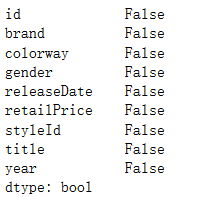


Figure 2-1-2 screenshot of Data validity auditing process and result

And here is the screenshot of those remaining data.

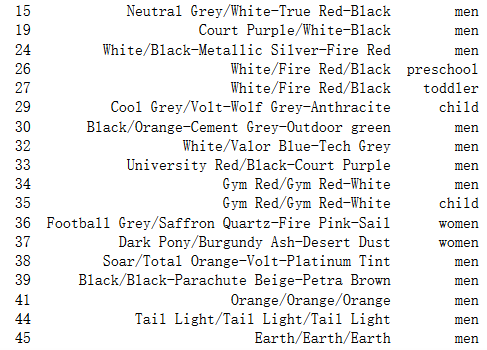


Figure 2-1-3 screenshot of Data validity auditing process and result

### Audit Completeness

There are 7 columns contained in the dataset, which are id, brand, colorway, gender, releasedate, retailPrice, styleId, title, year. These data cover all the important entities of sneakers which should have, and thus satisfy the completeness of database.

### Audit Consistency and Uniformity

The datasets are inner related by a common entity, title, which relates all the datasets. Thus, the final combined database is considered uniformed and consistent.

## 3.Reformatting the database

In this part, we used .loc command to reformate the database for the following normalizing process. After the reformatting process, the data has been divided and filled several different tables.

The Color\_gender table contains columns of 'id','colorway','gender', the Release\_time table contains columns of 'id','releaseDate','year', whereas the Brand\_Retailprice table is consisted of columns named 'id','title','brand' and 'retailPrice'. Here follows the screenshot of results of the reformatting process.



Figure 3-1-1 screenshot of database reformatting process and result



Figure 3-1-2 screenshot of database reformatting process and result

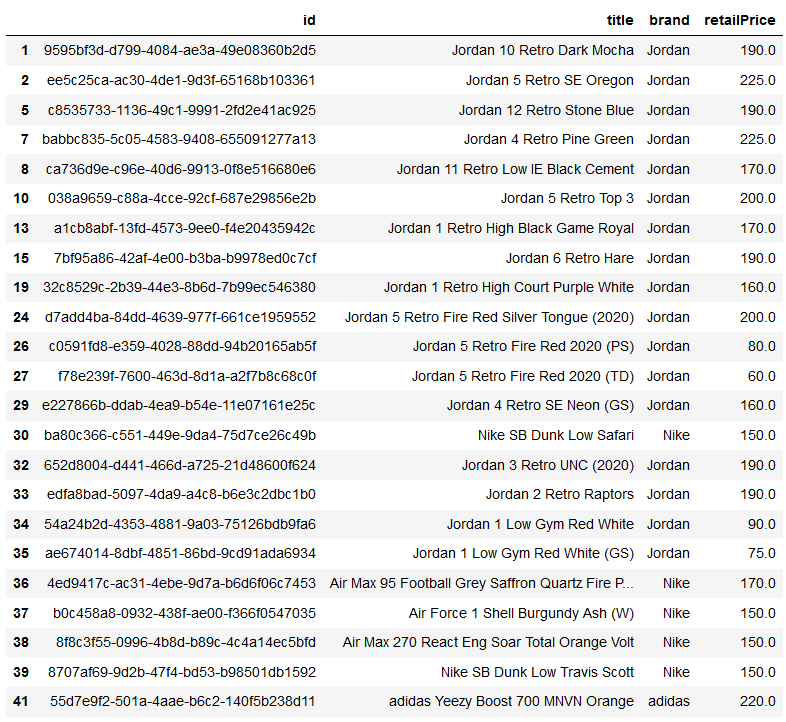


Figure 3-1-3 screenshot of database reformatting process and result

## 4.Normalization Analysis

For our sneaker database, we generally have three database: 1.Color\_Gender table, 2.Release\_time table, 3.Brand\_Retailprice table.

Since the id of the sneaker data is the primary key of each table and is already unique. There are no multi-values in each column, in other words, the value in each column is atomic. Also, no two columns store same type of values. Thus, the above database is already in First-normal-form.

However, the release year is dependent on the release data. So, the database is not in Third norm. Similarly, the title attribute in the Brand\_Retail price is dependent on the id, and the brand and retailPrice attribute is dependent on the title. Thus, the Brand\_Retail price also has transitive dependency.

Thus, we would separate the Release\_time table and Brand\_RetailPrice table to satisfy the database into second normal form and third normal form.

After the normalization, the sneaker database is now in Third Norm Form. The sneaker database consists of 5 tables, which are:

1.Sneaker\_Retailprice table(primary key is title)

2.Sneaker\_Title table(primary key is id, foreign key is title)

3.Release\_Date table(Primary key is id, foreign key is releaseDate)

4.Release\_Year table(Primary key is releaseDate)

5.Color\_Gender table(Primary key is id)

Here follows the screenshot of results of the normalization process.

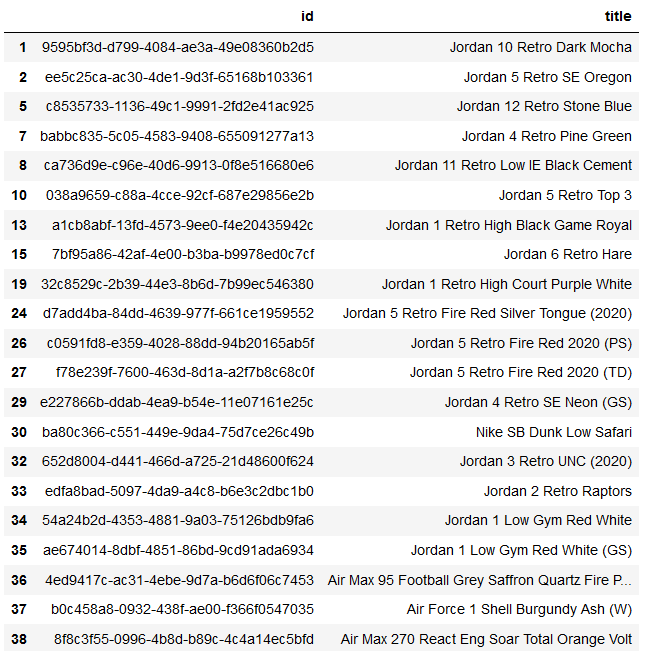


Figure 4-1-1 screenshot of database normalization process and result

## 5. Graph

### 5.1 UML model graph

From the UML model graph below we could see that the Release\_Year 'IS A' Release\_Date. Release\_Year inherited from the Release\_Date, and thus is a Release\_Date. For a single Sneaker, it has the attributes of Color\_Gender, Release\_Date and Sneaker\_Retailprice. Thus, the Sneaker\_Title 'Has A' Sneaker\_Retailprice, Release\_Date and Color\_gender.

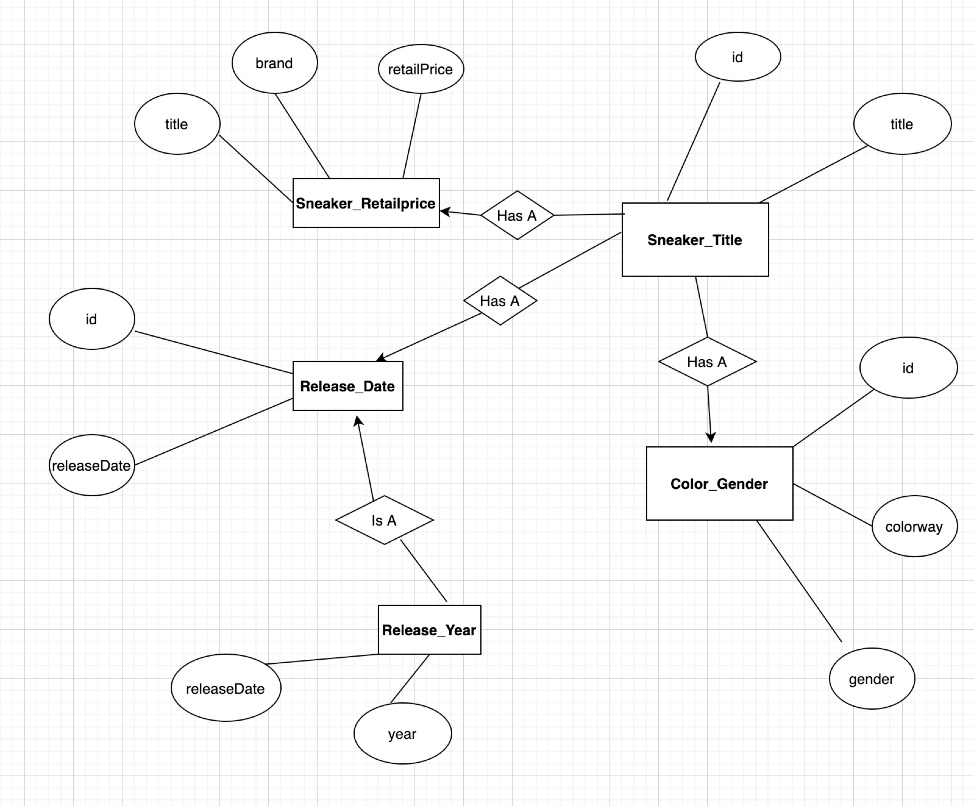


Figure 5-1-1 UML model graph

### 5.2 Conceptual Data Modeling

From the graph below we could see that in the sneaker database, there are 5 tables, namely, Sneaker\_Retailprice, Sneaker\_Title, Release\_Date, Release\_Year, Color\_Gender. The Release\_Date and Sneaker\_title has a Many-to-Many relationship. The Color\_Gender and Sneaker\_Title has a Many-to-Many relationship. The Release\_Date table and Release\_Year table has a One-to-Many relationship, since in one year there could be many detailed release date.The Sneaker\_Retailprice and Sneaker\_title is also One-to-Many relationship, since one sneaker could only have one retail price, but multiple sneakers could be sold at one price.

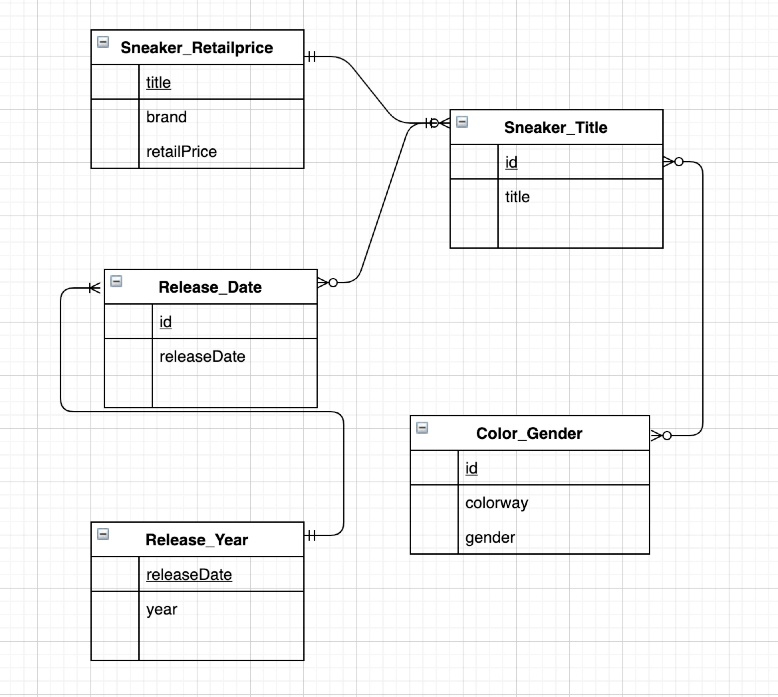


Figure 5-2-1 Conceptual Data Modeling Graph

### 5.3 Physical Model

In the following Physical model graph, we can see the detailed primary keys, foreign keys, table names, column names and data type of the database. The foreign key of Sneaker\_Title title is the primary key of the Sneaker\_Retailprice. The foreign key releaseDate is the primary key of table Release\_Year. The primary key of table Sneaker\_Title, Release\_Date and Color\_Gender are the same id.

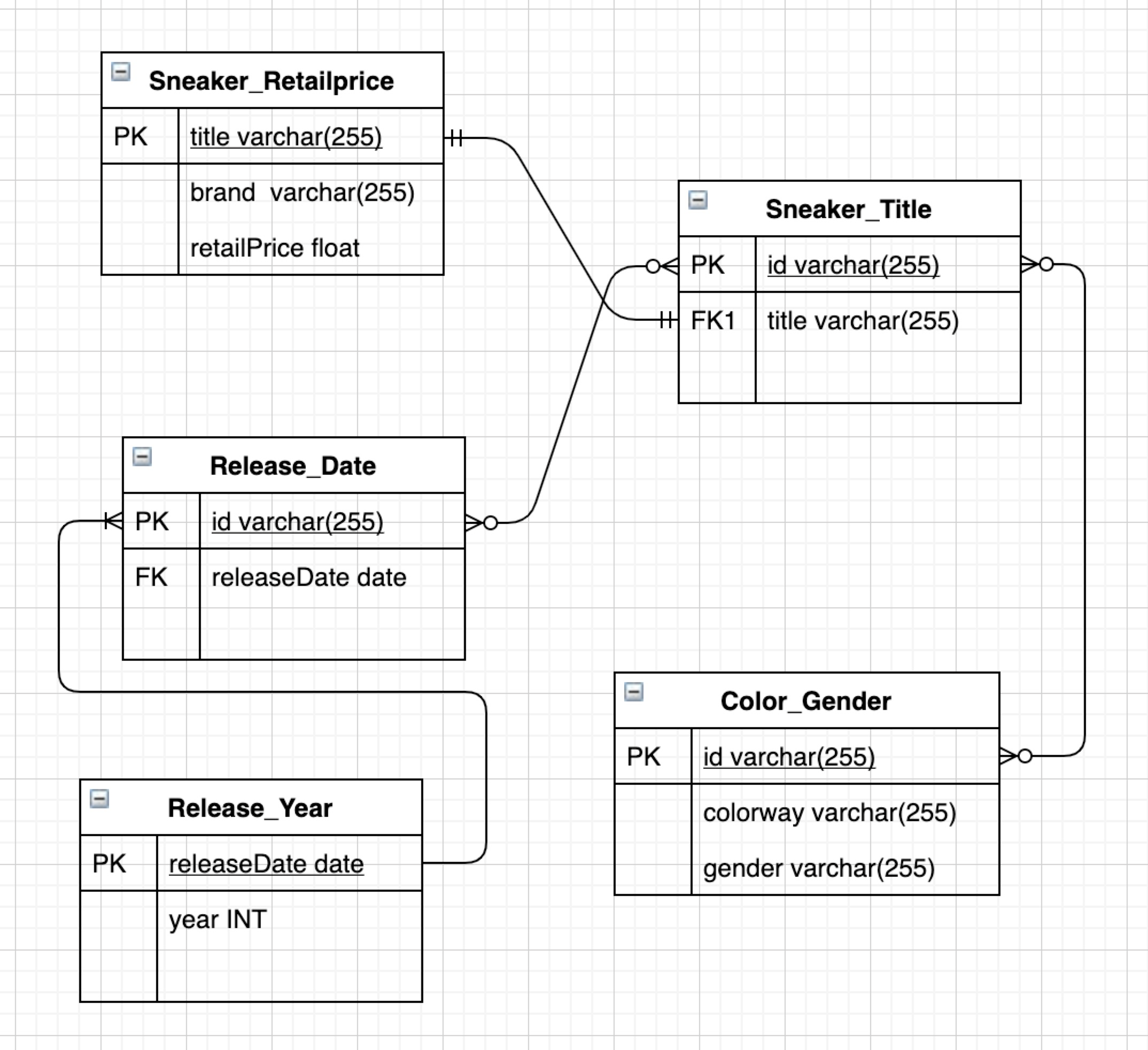


Figure 5-3-1 Physical Model Graph

## 6 Answering Questions

Question 1:

What are the ranges, data types and format of all of the attributes in your entities?

Answers: Entity1: Sneaker\_Title Attritutes: id varchar(255) title varchar (255)

Entity2: Sneaker\_Retailprice Attritutes: title varchar(255) brand varchar (255) retailPrice Float

Entity3: Release\_Date Attritutes: Id varchar(255) releaseDate date

Entity4: Color\_Gender Attritutes: colorway varchar(255) gender varchar(255)

Entity5: Release\_Year Attritutes: releaseDate date year INT

Question2:

When should you use an entity versus attribute? (Example: address of a person could be modeled as either)

Answers:

The original meaning of one entity is a subject, which refers to a table in the database. This entity must be distinguishable from other entities. Entities could be abstract or specific. The attribute is owned by entities and are used to describe each entity. The attribute could be described as a function that maps the entity that it belongs to the corresponding database.

While we first form a database, we need to establish the relationship between entities. When we dive into the entities, we need to be specific on each attribute, and thus, we need to add value to attributes, datatype and ranges to attributes.

The address of a person could indeed be modeled either as an attribute and an entity. If we define the address as an entity, the attribute belongs to it could be city, country, zip code, etc. If the address is defined as a person, the entity it belongs to could be a person. The person entity could have entities like name address, sex, age, etc.

### Question3:

When should you use an entity or relationship, and placement of attributes? (Example: a manager could be modeled as either)

A manager could be modeled as either. If a manager is modeled as an entity, a manager is a one category in the database, in other word, a table in the database. In this manager entity, attributes could be manageer\_id, name, age, sex, phone number, etc. If a manager is modeled as an attribute, it must belong to a certain entity, and manager is one of the attributes of this entity. This time, the entity that manager belongs to could be a department in a company, could be a job. The other attributes in the entity Manager belongs to could be normal staff, CEOs, etc. If a manager is modeled as a relationship, manager should be the connection of two entities. It is responsible for establishing relation between two entities. For example, there could be a managing relationship table describing the managing hierarchy of the company, like certain employees are under the leadership of certain managers.

Question4:

How did you choose your keys? Which are unique?

In my database, I used id of the sneaker as the primary key for Sneaker\_Title, Release\_Date and Color\_Gender table. Since the Id of each sneaker is the unique identifier of all these three tables, Id could be the primary key of all these three tables. When normalizing the database, we seperated the Sneaker\_Retailprice from the Sneaker\_Title table, the primary key of the new table is the title of the sneaker, which is also unique. We also separated the Release\_Year from the Release\_data database. The primary key of the Release\_year table is the releaseDate.

### Question5:

Did you model hierarchies using the “ISA” design element? Why or why not

ISA basically represents the inheritance relationship of different entities. In our UML graph model, we used ISA relationship to represent the relationship of Release\_Date table and Release\_Year table. Basically, Release\_Year is another form of Release\_Date, just being more general. This represents the inheritance relationship, and thus, is an ISA relationship.

Question6:

Were there design alternatives? What are their tradeoffs: entity vs. attribute, entity vs. relationship, binary vs. ternary relationships?

There are indeed alternatives in my database design.When forming the Color\_Gender table, we first decided to combine the Color\_Gender table into the Sneaker\_Title table. But it could be a little reduntant for the sneaker\_title database, since the entity of Colorway and gender does not have a direct strong relationship with the Sneaker\_Title. Thus, separting the Color\_Gender table from the Sneaker\_Title database would make more sense. Since the Sneaker\_Title database is connected to Color\_Gender table, Release\_Date table and Sneaker\_Retailprice table, it forms a ternary relationship.

Question7:

Where are you going find real-world data populate your model?

Currently all of my database data are from API:

<https://api>.thesneakerdatabase.com/v1/sneakers?limit=50&page=.

### Questions you must answer about your physical model:

**Are all the tables in 1NF?**

The original data is already in 1NF, since the primary key could uniquely identify a record, the values in each column of a table are atomic and there are no repeating groups.

**Are all the tables in 2NF?**

The original data is already in 2NF, since there are no composite keys, and thus, no partial dependencies. I would argue that the Release\_Year are not calculated from the Release\_Date, since they have different data type, and thus has nothing to do with calculation(the Release\_Date is string and Release\_Year is INT).

**Are all the tables in 3NF?**

The original data is not in 3NF, since the Release\_Year is dependent on the Release\_Date, and Release\_Date is dependent on the Id of the sneaker. Also, the Retail\_Price and Brand is dependent on the title of the sneaker, and the title is dependent on the Id. So, there are transitive dependencies in original Release\_Time table and Sneaker\_title table. So, we separated the above two tables to normalize the database into 3NF.

## Final Report

In this assignment, the following files are generated and used: sneaker.csv, gender.csv, Sneaker\_title.csv, colorway.csv, retailPrice.csv,test1.csv. These data are gathered from three different source, web API, web scraping and raw data, which are then merged to form the final conceptual model.

### Conclusion

In this assignment, firstly, we gathered data from different source, which are web API, web scraper and raw data. Then the data is cleaned, reformatted and combined. During the process, the null values are cleaned, and the structure and relationship of the data is much clearer. Finally, we formed the conceptual model and draw the Entity Relationship graph to further clarify the model.

### Contribution

Original contribution: 40% By External source: 20% Provided by the TA documents : 40%.

### Citations

https://app.swaggerhub.com/apis-docs/tg4solutions/the-sneaker-database/1.0.0#/

sneakers/getSneakers

http://www.thesneakerdatabase.com/

https://towardsdatascience.com/web-scraping-using-selenium-and-beautifulsoup-

99195cd70a58

http://unclechen.github.io/2016/12/11/python%E5%88%A9%E7%94%A8beautifulsoup+sele

nium%E8%87%AA%E5%8A%A8%E7%BF%BB%E9%A1%B5%E6%8A%93%E5%8F%96%E7%BD%91%E9%A1%B5%E5%86%85%E5%AE%B9/

## License

Copyright 2019 Weibo Dai Chuhong Yu

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.