Explain Video Link:

- https://drive.google.com/file/d/1zu-Yk6b57rPoFyw8g3H165IiiBgAwN-S/view (https://drive.google.com/file/d/1zu-Yk6b57rPoFyw8g3H165IiiBgAwN-S/view)

Name:

- LIU,HONGYANG
- 17201091/1

Q1:

1. You are required to make a user-agent that will crawl the WWW (your familiar domain) to produce dataset of a particular website.
    - the web site can be as simple as a list of webpages and what other pages they link to
    - the output does not need to be in XHTML (or HTML) form a multi-stage approach (e.g. produce the xhtml or html in csv format)

import libraries

In [1]:

```
#import libraries
import requests
import pandas as pd
import time
```

web crawling

In [2]:

```
# user-agent
url_agent = 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_3) AppleWebKit/537.36
 (KHTML, like Gecko) Chrome/80.0.3987.122 Safari/537.36'

headers = {'User-Agent':url_agent}

Name=["customer","transaction","product"]

# WWW websites
url="https://beijing01.github.io/DataWarehouse/"

content=[]

for i in Name:

    #execute the crawlling task every 1 seconds to avoid banning by the server

    time.sleep(1)

    try:
        # build url
        webpage =url+i+".html"

        resp = requests.get(webpage,headers=headers)

        df = pd.read_html(resp.text)

        content.append(df[0])

    except:
        print("The website doesn't work!",url+i)
```

data preprocessing

In [3]:

```
data=content[0]
customer = {'customer_id':data.iloc[:,1],'DOB':data.iloc[:,2],'Gender':data.iloc
[:,3],
            'City_id':data.iloc[:,4],'City':data.iloc[:,5]}
customer = pd.DataFrame(customer)
```

In [4]:

```
#customer
customer.head()
```

Out[4]:

	customer_id	DOB	Gender	City_id	City
0	268408	02-01-1970	M	4.0	Tianjin
1	269696	07-01-1970	F	8.0	Chaohu
2	268159	08-01-1970	F	8.0	Chaohu
3	270181	10-01-1970	F	2.0	Chongqing
4	268073	11-01-1970	M	1.0	Beijing

In [5]:

```
data=content[1]

transactions = {'transaction_id':data.iloc[:,1],'customer_id':data.iloc[:,2],'transaction_date':data.iloc[:,3],
                'prod_cat_code':data.iloc[:,4],'prod_subcat_code':data.iloc[:,5],
                'Quantity':data.iloc[:,6],'Price':data.iloc[:,7],'Tax':data.iloc[:,8],
                'Total_amount':data.iloc[:,9],'Store_type':data.iloc[:,10]}
transactions = pd.DataFrame(transactions)
```

In [6]:

```
transactions.head()
```

Out[6]:

	transaction_id	customer_id	transaction_date	prod_cat_code	prod_subcat_code	Quantity
0	80712190438	270351	28-02-2014	1	1	-5
1	29258453508	270384	27-02-2014	5	3	-5
2	51750724947	273420	24-02-2014	6	5	-2
3	93274880719	271509	24-02-2014	11	6	-3
4	51750724947	273420	23-02-2014	6	5	-2

In [7]:

```
#transaction
transactions.head()
```

Out[7]:

	transaction_id	customer_id	transaction_date	prod_cat_code	prod_subcat_code	Quantity
0	80712190438	270351	28-02-2014	1	1	-5
1	29258453508	270384	27-02-2014	5	3	-5
2	51750724947	273420	24-02-2014	6	5	-2
3	93274880719	271509	24-02-2014	11	6	-3
4	51750724947	273420	23-02-2014	6	5	-2

In [8]:

```
data=content[2]

product={'pro_cat_code':data.iloc[:,1],'pro_cat':data.iloc[:,2],
         'prod_subcat_code':data.iloc[:,3],'prod_subcat':data.iloc[:,4]}

product = pd.DataFrame(product)
```

In [9]:

```
#product
product.head()
```

Out[9]:

	pro_cat_code	pro_cat	prod_subcat_code	prod_subcat
0	1	Clothing	4	Mens
1	1	Clothing	1	Women
2	1	Clothing	3	Kids
3	2	Footwear	1	Mens
4	2	Footwear	3	Women

save to csv file

In [10]:

```
# store the data to csv format
customer.to_csv("customer.csv")
transactions.to_csv("transactions.csv")
product.to_csv("product.csv")
```

Q2:

1. Draw snowflake schema diagram for the above dataset. Justify your attributes to be selected in the respective dimensions.

In [11]:

```
# check the data sets generated in first step
#customer
customer.head()
```

Out[11]:

	customer_id	DOB	Gender	City_id	City
0	268408	02-01-1970	M	4.0	Tianjin
1	269696	07-01-1970	F	8.0	Chaohu
2	268159	08-01-1970	F	8.0	Chaohu
3	270181	10-01-1970	F	2.0	Chongqing
4	268073	11-01-1970	M	1.0	Beijing

In [12]:

```
#transaction
transactions.head()
```

Out[12]:

	transaction_id	customer_id	transaction_date	prod_cat_code	prod_subcat_code	Quantity
0	80712190438	270351	28-02-2014	1	1	-5
1	29258453508	270384	27-02-2014	5	3	-5
2	51750724947	273420	24-02-2014	6	5	-2
3	93274880719	271509	24-02-2014	11	6	-3
4	51750724947	273420	23-02-2014	6	5	-2

In [13]:

```
#product
product.head()
```

Out[13]:

	pro_cat_code	pro_cat	prod_subcat_code	prod_subcat
0	1	Clothing	4	Mens
1	1	Clothing	1	Women
2	1	Clothing	3	Kids
3	2	Footwear	1	Mens
4	2	Footwear	3	Women

The snowflake schema should meet the three requirements

- 1 build **fact table** surrended by **dimension tables**
- 2 reduce data redundancy
- 3 Normalized data sturcture

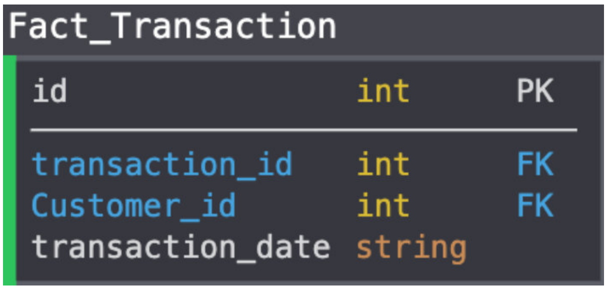
We will draw the the entity relationship diagram to represent the snowflake schema

In [14]:

```
#draw the fact table to represent the transaction and customer information
Fact_Transaction=transactions[["transaction_id","customer_id","transaction_date"
]]
Fact_Transaction.head(3)
```

Out[14]:

	transaction_id	customer_id	transaction_date
0	80712190438	270351	28-02-2014
1	29258453508	270384	27-02-2014
2	51750724947	273420	24-02-2014



Fact_Transaction

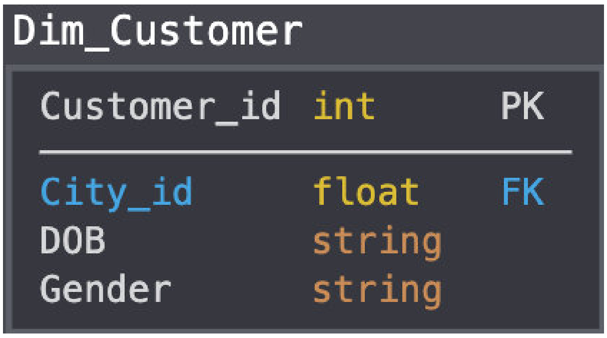
id	int	PK
transaction_id	int	FK
Customer_id	int	FK
transaction_date	string	

In [15]:

```
#draw customer dim table to represent customer information
Dim_Customer=customer[["customer_id","DOB","Gender","City_id"]]
Dim_Customer.head(3)
```

Out[15]:

	customer_id	DOB	Gender	City_id
0	268408	02-01-1970	M	4.0
1	269696	07-01-1970	F	8.0
2	268159	08-01-1970	F	8.0



Dim_Customer

Customer_id	int	PK
City_id	float	FK
DOB	string	
Gender	string	

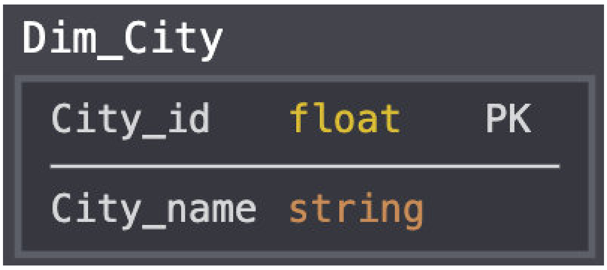
In [16]:

```
#draw city dim table

Dim_City=customer[["City_id","City"]]
Dim_City.head(3)
```

Out[16]:

	City_id	City
0	4.0	Tianjin
1	8.0	Chaohu
2	8.0	Chaohu



Dim_City

City_id	float	PK
City_name	string	

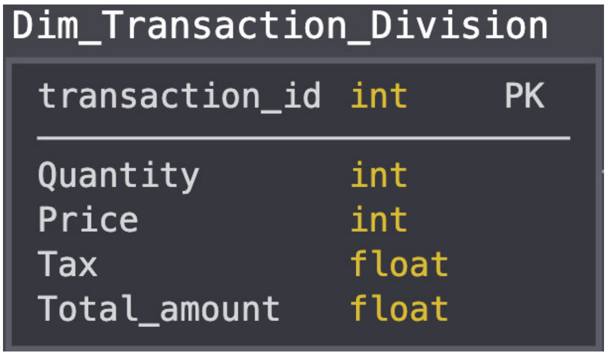
In [17]:

```
#draw Transaction_Division dim table to represent the goods, number, price  in e
very transaction

Dim_Transaction_Division=transactions[["transaction_id","Quantity","Price","Tax"
,"Total_amount"]]
Dim_Transaction_Division.head(3)
```

Out[17]:

	transaction_id	Quantity	Price	Tax	Total_amount
0	80712190438	-5	-772	405.300	-4265.300
1	29258453508	-5	-1497	785.925	-8270.925
2	51750724947	-2	-791	166.110	-1748.110



Dim_Transaction_Division

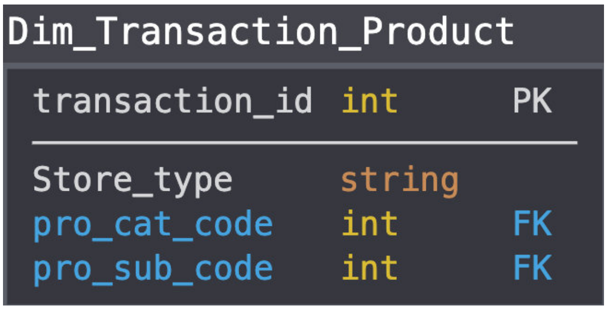
transaction_id	int	PK
Quantity	int	
Price	int	
Tax	float	
Total_amount	float	

In [18]:

```
# draw the transaction product table
Dim_Transaction_Product=transactions[["transaction_id","Store_type","prod_cat_co
de","prod_subcat_code"]]
Dim_Transaction_Product.head(3)
```

Out[18]:

	transaction_id	Store_type	prod_cat_code	prod_subcat_code
0	80712190438	e-Shop	1	1
1	29258453508	e-Shop	5	3
2	51750724947	TeleShop	6	5



Dim_Transaction_Product

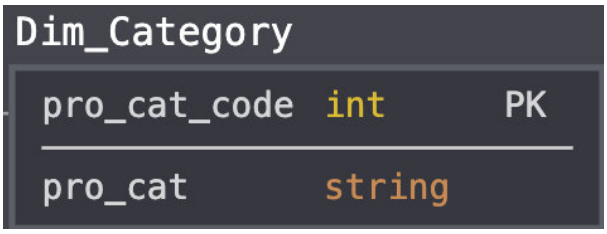
transaction_id	int	PK
Store_type	string	
pro_cat_code	int	FK
pro_sub_code	int	FK

In [19]:

```
# draw the product category table
Dim_Category=product[["pro_cat_code","pro_cat"]]
Dim_Category.head(3)
```

Out[19]:

	pro_cat_code	pro_cat
0	1	Clothing
1	1	Clothing
2	1	Clothing



Dim_Category

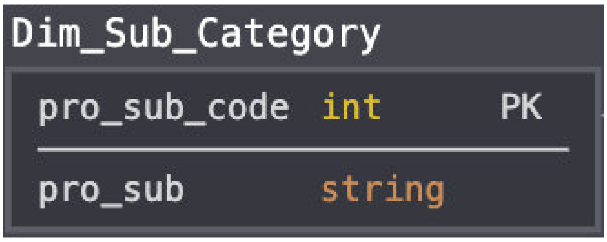
pro_cat_code	int	PK
pro_cat	string	

In [20]:

```
#draw the product subcategory table
Dim_Sub_Category=product[["prod_subcat_code","prod_subcat"]]
Dim_Sub_Category.head(3)
```

Out[20]:

	prod_subcat_code	prod_subcat
0	4	Mens
1	1	Women
2	3	Kids

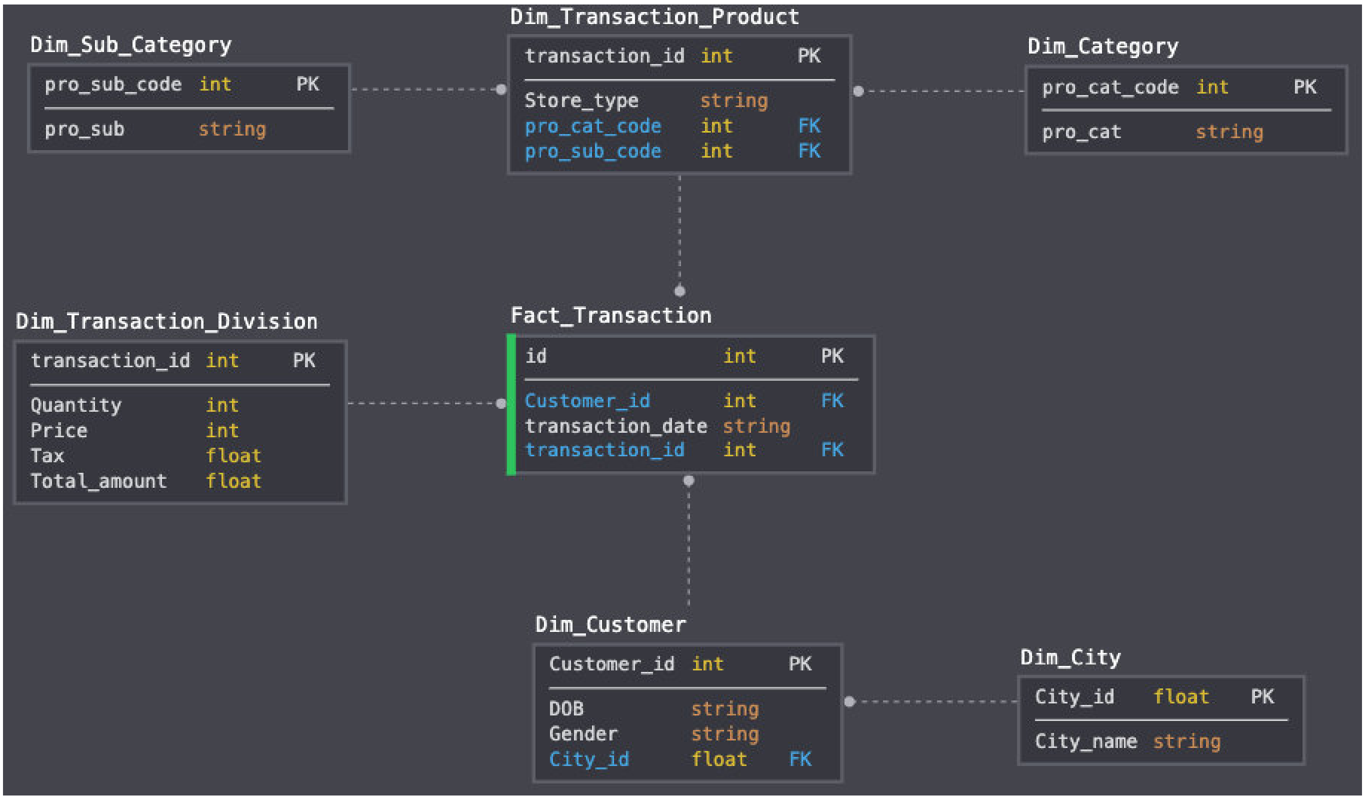


Dim_Sub_Category

pro_sub_code	int	PK
pro_sub	string	

Draw snowflake schema diagram for the above dataset

- PK: Primary Key
- FK: Foreign Key



Dim_Sub_Category

pro_sub_code	int	PK
pro_sub	string	

Dim_Transaction_Product

transaction_id	int	PK
Store_type	string	
pro_cat_code	int	FK
pro_sub_code	int	FK

Dim_Category

pro_cat_code	int	PK
pro_cat	string	

Dim_Transaction_Division

transaction_id	int	PK
Quantity	int	
Price	int	
Tax	float	
Total_amount	float	

Fact_Transaction

id	int	PK
Customer_id	int	FK
transaction_date	string	
transaction_id	int	FK

Dim_Customer

Customer_id	int	PK
DOB	string	
Gender	string	
City_id	float	FK

Dim_City

City_id	float	PK
City_name	string	

In [ ]: