Fastcampus Data Science Extension SCHOOL

SQL(2) - SQL(feat. jupyter, pandas)

Review

- Database
- Schema
  - 외부스키마
  - 개념스키마
  - 내부스키마
- sqlite
- SQL
  - create
  - drop
  - alter
  - select
    - where

지난 숙제

`OrderDetails에서 Quantity가 40개 이상이며, Customers의 CustomerName이 Ernst나 Stop을 포함하는 전체 데이터를 선택하세요.`

Answer

```
SELECT * FROM [Orders]
where
OrderID in (
        SELECT OrderID FROM OrderDetails
            where Quantity > 40)
            and CustomerID in (
                SELECT CustomerID FROM [Customers]
                where CustomerName like '%Ernst%'
                or CustomerName like '%Stop%');
```

import csv

```
sqlite> .mode csv
sqlite> .import {csv filepath} {table name}
```

Change schema in sqlite with temporary table

```
sqlite> begin transaction;
sqlite> create temporary table Products_backup(
   ...> ProductID, ProductName, SupplierID, CategoryID,
   Unit, Price);
sqlite> .tables
Categories            OrderDetails          Shippers
Customers             Orders                Suppliers
Employees             Products              temp.Products_backup
sqlite> insert into temp.Products_backup
select ProductID, ProductName, SupplierID, CategoryID,
Unit, Price from Products;
sqlite> drop table Products;
sqlite> create table Products(
   ...> ProductID integer,
   ...> ProductName text,
   ...> SupplierID integer,
   ...> CategoryID integer,
   ...> Unit text,
   ...> Price integer
   ...> );
sqlite> insert into Products select * from temp.Products_backup;
sqlite> drop table temp.Products_backup;
sqlite> commit;
```

sql with sqlite3, pandas

```python
import pandas as pd
import sqlite3 as lite
```

connect with sqlite

```
db = lite.connect()
```

read_sql or execute

- read_sql

```
query = "SELECT * FROM Customers;"
pd.read_sql(query, db)
```

- execute and fetchall

```
cur = db.cursor()
cur.execute(query)
cur.fetchall()
```

show table list

```
query = """
        SELECT name
    FROM sqlite_master
    WHERE
        type = 'table'
    ;
"""
```

show schema

```
query = """
        SELECT sql
    FROM sqlite_master
    WHERE
        type = 'table'
    ;
"""
```

Which is faster?

```
len(pd.read_sql())
```

```
pd.read_sql(count(*))
```

```
time.time()
# script
time.time()
```

sqlite aggregate functions

- count(*)
- count(X)
- sum(X)
- avg(X)
- group_concat(X)
- group_concat(X,Y)
- max(X)
- min(X)

filter with pandas

```
france = df["Country"] == "France"
germany = df["Country"] == "Germany"
paris = df["City"] == "Paris"
df[germany | paris]
df[germany & paris]
```

filter with sql - operator

```
query = """
    select *
    from Customers
    where
        Country = "France"
        and City = "Paris"
    ;
"""
```

sort with pandas

```
products_df = pd.read_sql('select * from Products;', db)
products_df.sort_values('ProductName', ascending=False)\
        [["ProductName", "Price"]]
```

sort with sql - ORDER BY

```
query = """
    select ProductName, Price
    from Products
    order by ProductName desc
    ;
"""
pd.read_sql(query, db)
```

text mining - like

```
# text mining
query = """
    select ProductName, Price
    from Products
    where
        ProductName like "%Ch%"
    ;
"""
pd.read_sql(query, db)
```

join in pandas - merge

```
integrated_df = orders_df.merge(df, on="CustomerID")\
        [["OrderID", "CustomerID","ContactName","Address"]]
integrated_df.head()
```

join with sql - don't

```
query = """
    select *
    from Customers, Orders
    ;
"""
pd.read_sql(query, db)
```

join with sql - better(1)

```
query = """
    select Orders.OrderID, Orders.CustomerID,
        Customers.ContactName, Customers.Address
    from Customers, Orders
    where
        Customers.CustomerID = Orders.CustomerID
    ;
"""
```

join with sql - better(2)

```
query = """
    select O.OrderID, O.CustomerID, C.ContactName, C.Address
    from Customers C, Orders O
    where
        C.CustomerID = O.CustomerID
    ;
"""
```

join with sql - best

```
query = """
    select O.OrderID, O.CustomerID, C.ContactName, C.Address
    from Customers C
        join Orders O
        on C.CustomerID = O.CustomerID
    ;
"""
```

GROUP BY in pandas

```
date_groups = orders_df.groupby("OrderDate")
date_groups.get_group("1996-07-08")

orders_df["OrderDate"].unique()
```

```
order_count_by_date = pd.DataFrame([
    {
        "OrderDate": OrderDate,
        "Count": len(date_groups.get_group(OrderDate)),
    } for OrderDate in orders_df["OrderDate"].unique()
])
order_count_by_date
```

GROUP BY

```
#sql
query = """
    select count(*), OrderDate
    from Orders
    group by OrderDate
    ;
"""
pd.read_sql(query, db)
```

Do It Yourself

`OrderDate` 를 조작하여 yyyy-mm 의 형태로 바꾼 컬럼을 추가한 뒤, 연-월 기반의 주문횟수를 pandas와 sql로 각각 구현하세요

`hint: apply, lambda, substr()`

숙제

- 앞서 배운 groupby, join을 활용하여 월간 판매량 합과 평균 구매가격을 pandas와 sql로 각각 구현하세요.