Fastcampus Data Science Extension SCHOOL

SQL(3)

join again

JOIN

- INNER JOIN: 양쪽의 값 비교 후 조건에 맞는 데이터 병합
- LEFT JOIN: JOIN 왼쪽을 기준으로 오른쪽의 일치하는 데이터 병합
- RIGHT JOIN: JOIN 오른쪽을 기준으로 왼쪽의 일치하는 데이터 병합
- FULL OUTER JOIN : 일치하지 않는 값까지 모두 병합
- SELF JOIN: 자기 자신을 병합

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()`

SUM, AVG로 합과 평균 매출 구하기

sqlite aggregate functions

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

Having vs where

- 공통점: filter
- where
  - 항상 from 뒤에 위치
  - 모든 필드에 대해 필터링 가능
- having
  - group by 뒤에 위치
  - group by 후 생성된 새로운 테이블에 조건을 줄때

how to use having?

```
query = """
    SELECT
        SUM(d.Quantity) "Count",
        SUM(d.Quantity * p.Price) "Sales",
        ROUND(AVG(d.Quantity * p.Price), 2) "avg",
        SUBSTR(o.OrderDate, 0, 8) "month"
    FROM
        OrderDetails d
        JOIN
            Products p
            ON p.ProductID = d.ProductID
        JOIN
            Orders o
            ON d.OrderID = o.OrderID
    GROUP BY
        substr(o.OrderDate, 0, 8)
    HAVING
        d.Quantity >= 20
    ;
"""
pd.read_sql(query, db)
```

SQL 작성 팁

- FROM 을 먼저 채운 뒤, 다른 필드들을 채워나가면 편해요
- FROM SELECT WHERE -> GROUP BY -> HAVING -> ORDER BY

지난 숙제

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

Product 별 매출과 구매횟수 구하기

Google Analytics with SQL

Cohort Analysis

Cohort

- 특정 기간에 특정 경험을 공유한 집단간의 행동패턴을 비교/분석하는 것
- 기간 별 분석이 가능해 서비스의 기간별 추이 파악 가능
- 하나의 표에 유저 행동과 서비스의 퍼포먼스 분석이 가능

How to?

- 유저를 방문일 기준으로 나눔(Cohort)
- 각 cohort에 포함된 유저 수를 구함(Cohort Size)