In [2]:
```
import numpy as np
import pandas as pd
```

In [3]:
```
customers = pd.read_csv('customers.csv',encoding = 'cp949')
orders = pd.read_csv('orders.csv',encoding = 'cp949')
products = pd.read_csv('products.csv',encoding = 'cp949')
```

In [4]:
```
customers.head()
```

Out[4]:

	customer_id	name	gender	age	join_date
0	101	고객_1	남	38	2023-01-31
1	102	고객_2	여	42	2023-02-28
2	103	고객_3	남	30	2023-03-31
3	104	고객_4	남	30	2023-04-30
4	105	고객_5	남	43	2023-05-31

In [6]:
```
orders.head()
```

Out[6]:

	order_id	customer_id	product_name	quantity	price	order_date	total_price
0	1001	101	키보드	1	1452107	2024-01-01	1452107
1	1002	108	노트북	2	1361736	2024-01-02	2723472
2	1003	110	스마트폰	2	1057293	2024-01-03	2114586
3	1004	101	헤드폰	2	629879	2024-01-04	1259758
4	1005	110	노트북	2	1319460	2024-01-05	2638920

In [7]:
```
products.head()
```

Out[7]:

	product_name	category	stock
0	노트북	전자제품	18
1	스마트폰	전자제품	97
2	태블릿	전자제품	10
3	헤드폰	액세서리	17
4	키보드	액세서리	97

데이터프레임 병합 (관계형 데이터베이스의 JOIN과 유사)

In [ ]:
```
print("\n--- merge 예제: 주문 데이터에 고객 정보 추가 ---")
# 'customer_id'를 기준으로 orders와 customers를 병합
# how='inner'는 양쪽에 모두 존재하는 customer_id만 포함
```

데이터프레임 연결 (행 또는 열 방향으로 단순히 붙이기)

In [ ]:
```
print("\n--- concat 예제 1: 신규 회원 데이터 추가 (행 방향) ---")
new_customers_data = {
    'customer_id': [111, 112],
    'name': ['고객_11', '고객_12'],
    'gender': ['남', '여'],
    'age': [25, 35],
    'join_date': [pd.to_datetime('2024-05-01'), pd.to_datetime('2024-06-15')]
}
new_customers_df = pd.DataFrame(new_customers_data)

# axis=0 (기본값): 행 방향으로 연결(customers)
```

특정 기간의 주문 데이터 분리 후 합치기 (행 방향)

In [ ]:
```
print("\n--- concat 예제 2: 주문 데이터를 분리 후 합치기 (행 방향) ---")
# 2024년 1월 10일 이전 주문
orders_part1 = orders[orders['order_date'] < '2024-01-10']
# 2024년 1월 10일 이후 주문
orders_part2 = orders[orders['order_date'] >= '2024-01-10']

# 두 개의 부분 데이터프레임을 다시 합치기
```

특정 열을 기준으로 데이터를 그룹화

In [ ]:
```
print("\n--- groupby 예제 1: 고객별 총 구매 금액 및 주문 건수 ---")
# 'customer_id'를 기준으로 그룹화하고 'total_price'의 합계와 'order_id'의 개수 계산


print("\n--- groupby 예제 2: 상품 카테고리별 평균 가격 및 최대 재고 ---")
# 상품 데이터와 주문 데이터를 병합하여 카테고리 정보 추가
orders_with_category = pd.merge(orders, products[['product_name', 'category']], on='product_name', how='left')

# 'category'를 기준으로 그룹화하고 'price'의 평균과 'stock'의 최대값 계산
```

Series의 고유 값과 그 빈도를 계산

In [ ]:
```
print("\n--- value_counts 예제 1: 가장 많이 팔린 상품 ---")
# 'product_name'별 판매량 (주문 건수 기준)


print("\n--- value_counts 예제 2: 고객 성별 분포 ---")
# 고객 데이터에서 'gender'별 분포


print("\n--- value_counts 예제 3: 가장 많은 주문을 한 고객 (고객 ID 기준) ---")
# 'customer_id'별 주문 건수
```

(aggregate): 그룹화된 데이터에 여러 집계 함수 적용

In [ ]:
```
print("\n--- agg 예제: 고객별 총 구매 금액, 평균 구매 금액, 최대 구매 금액 ---")
# 'customer_id'를 기준으로 그룹화하고 여러 집계 함수 적용


print("\n--- agg 예제 (사용자 정의 함수): 주문일의 범위 ---")
# 주문 데이터에서 월별 주문 건수 및 주문일 범위 (최소/최대 날짜)
```

연속형 데이터를 구간으로 나누어 범주형 데이터로 변환

In [ ]:
```
print("\n--- cut 예제 1: 고객 연령대를 구간으로 나누기 ---")
# 고객 연령대를 20대, 30대, 40대, 50대 등으로 구분
bins = [10, 20, 30, 40, 50, 60]
labels = ['10대', '20대', '30대', '40대', '50대']



print("\n--- cut 예제 2: 상품 가격대를 구간으로 나누기 ---")
# 주문 상품의 가격을 저가, 중가, 고가로 구분
# np.linspace를 사용하여 자동으로 4개의 동일한 간격의 구간 생성






print("\n--- cut 예제 3: 가격대별 판매량 분석 ---")
# price_group 별 판매량 집계
```