04-12-2020

LECTURE - 20

200

KANAV BANSAL

MERGING AND GROUPBY:

MERGING: JOINING

↳ Pandas has full-featured, high performance in-memory join operations idiomatically very similar to relational databases like SQL.

↳ Pandas provides a single function, merge, as the entry point for all the standard database join operations between DataFrame Objects.

SYNTAX:

```
Pd.merge(left, right, how='inner', on=None,
    left_on=None, right_on=None, left_index=False,
    right_index=False, sort=True)
```

→ left - A DataFrame object

→ right - Another DataFrame object

→ on - Columns (names) to join on. Must be found in both the left and right DataFrame objects.

→ left_on - columns from the left DataFrame to use as keys. Can either be column names or arrays with length equal to the length of the DataFrame.

→ right_on - columns from the right DataFrame to use as keys. Can either be column names or arrays with length equal to the length of the DataFrame.

→ left_index - If True, use the index (row labels) from the left DataFrame as its join key(s). In case of a DataFrame with a MultiIndex (hierarchical), the number of levels must match the number of join keys from the right DataFrame.

→ right_index - Same usage as left_index for right DataFrame.

→ how - One of 'left', 'right', 'outer', 'inner'. Default to 'inner'. Each method

→ Sort - Sort the result DataFrame by the join keys in lexicographical order. Defaults to True, setting to False will improve the performance substantially in many cases.

GROUPBY:

↳ This function is used to split the data into groups based on some criteria.

↳ Pandas objects can be split on any of their axes.

↳ The abstract definition of grouping is to Provide a mapping of label to group names.

SYNTAX:

DataFrame.groupby(by=None, axis=0, level=None, as_index=True, sort=True, group_keys=True, squeeze=False, **kwargs)

→ by - mapping, function, str or iterable.

→ axis - int, default 0.

→ level - If the axis is a multiIndex (hierarchical), group by a particular level or levels.

→ as_index: For aggregated output, return object with group labels as the index. Only relevant for DataFrame input. as_index=False is effectively 'SQL-style' grouped output.

→ sort - sort group keys. Get better performance by turning this off.

Note: This doesn't influence the order of observations within each group.

Groupby preserves the order of rows within each group.

→ group_keys: when calling apply, add group keys to index to identify pieces.

→ squeeze - Reduce the dimensionality of the return type if possible, otherwise return a consistent type.

→ Returns - Groupby object.

I/p - cust_df = pd.read_csv('data/customer_data.csv')

Prod_df = pd.read_csv('data/product_data.csv')

pur_df = pd.read_csv('data/purchase_data.csv)

→ pur_df.head()

O/p -

	cust_id	prod_id
0	cust_12345	P_001
1	cust_12346	P_003
2	cust_12347	P_002
3	cust_12348	P_004
4	cust_12349	P_001

→ cust_df.head()

O/p -

	cust_id	prod_id
0	cust_12345	Sally
1	cust_12346	Jenna
2	cust_12347	Eliana
3	cust_12348	Cristen
4	cust_12349	Steve

→ prod_df.head()

O/p -

	prod_id	Prod_name
0	P_001	Machine learning (ML)
1	P_002	Data structurs (DS)
2	P_003	Full stack development (FSD)
3	P_004	competitive Programming (CP)

→ df = pd.merge(pur_df, cust_df, how='inner', on='cust_id')

df.head()

O/p -

	cust_id	prod_id	cust_name
0	cust_12345	P_001	Sally
1	cust_12346	P_003	Jenna
2	cust_12347	P_002	Eliana
3	cust_12348	P_004	Cristen
4	cust_12349	P_001	Steve

inner - joins means commondata in the DF.

→ df = pd.merge(df, prod_df, how='inner', on='prod_id')

df.head()

O/p -

	cust_id	prod_id	Cust_name	Prod_name
0	cust_12345	P_001	Sally	ML
1	cust_12349	P_003	Steve Jenna	ML
2	cust_12346	P_002	Kelli Eliana	ML
3	cust_12348	P_004	Aretha Cristen	ML
4	cust_12345	P_001	Carla Steve	ML

→ df.prod_name.value_counts()

O/p - Data structures 9

Machine learning 9

Competitive Programming 6

Full stack Development 5

Name : prod_name, dtype : int64

→ new_df = df.groupby('prod_name')
new_df.first()

O/p -

Prod_name	cust_id	Prod_id	cust_name
CP	cust_12348	P_004	Cristen
DS	cust_12347	P_002	Eliana
FSD	cust_12346	P_003	Jenna
ML	cust_12345	P_001	Sally

→ new_df.group.keys() → Group names

O/p - dict_keys(['CP', 'DS', 'FSD', 'ML'])

→ new_df.get_group('ML') → required group

→ new_df.describe()

→ new_df = df.groupby('species')

→ new_df['sepallengthcm'].mean()

O/p -

species	
Iris-setosa	5.006
Iris-versicolor	5.936
Iris-virginica	6.588

Name: Sepallength, dtype: float64