Q1

Consider the two DataFrames shown below, both of which have `Name` as the index. Which of the following expressions can be used to get the data of all students (from student_df) including their roles as staff, where `nan` denotes no role?

`student_df`

	School
Name	
James	Business
Mike	Law
Sally	Engineering

`staff_df`

	Role
Name	
Kelly	Director of HR
Sally	Course liasion
James	Grader

- ○ `pd.merge(student_df, staff_df, how='right', left_index=True, right_index=True)`
- ○ `pd.merge(staff_df, student_df, how='right', left_index=False, right_index=True)`
- [✓ `pd.merge(student_df, staff_df, how='left', left_index=True, right_index=True)`]
- ○ `pd.merge(staff_df, student_df, how='left', left_index=True, right_index=True)`

Q2

Consider a DataFrame named `df` with columns named `P2010`, `P2011`, `P2012`, `P2013`, `P2014` and `P2015` containing float values. We want to use the apply method to get a new DataFrame named `result_df` with a new column `AVG`. The `AVG` column should average the float values across `P2010` to `P2015`. The apply method should also remove the 6 original columns (`P2010` to `P2015`). For that, what should be the value of `x` and `y` in the given code?

```
frames = ['P2010', 'P2011', 'P2012', 'P2013','P2014', 'P2015']
df['AVG'] = df[frames].apply(lambda z: np.mean(z), axis=x)
result_df = df.drop(frames,axis=y)
```

- [✓ x = 1
  y = 1]
- ○ x = 0
  y = 1
- ○ x = 1
  y = 0
- ○ x = 0
  y = 0

Q3

Consider the Dataframe `df` below, instatiated with a list of grades, ordered from best grade to worst. Which of the following options can be used to substitute **X** in the code given below, if we want to get all the grades **between** 'A' and 'B' where 'A' is better than 'B'?

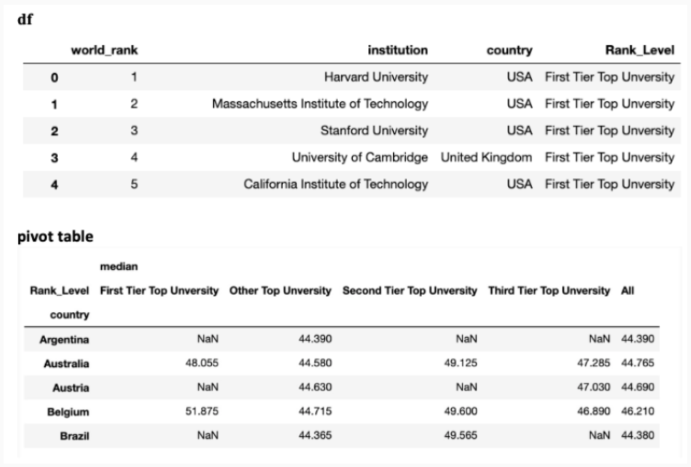
```
import pandas as pd

df = pd.DataFrame(['A+', 'A', 'A-', 'B+', 'B', 'B-', 'C+', 'C', 'C-', 'D+', 'D'], index=['excellent', 'excellent', 'excellent', 'good', 'good', 'good',
'ok', 'ok', 'ok', 'poor', 'poor'], columns = ['Grades'])
my_categories= X
grades = df['Grades'].astype(my_categories)
result = grades[(grades>'B') & (grades<'A')]
```

- ○ `my_categories = pd.CategoricalDtype(categories=['D', 'D+', 'C-', 'C', 'C+', 'B-', 'B', 'B+', 'A-', 'A', 'A+'])`
- ○ `(my_categories=['A+', 'A', 'A-', 'B+', 'B', 'B-', 'C+', 'C', 'C-', 'D+', 'D'], ordered=True)`
- ○ `my_categories = pd.CategoricalDtype(categories=['A+', 'A', 'A-', 'B+', 'B', 'B-', 'C+', 'C', 'C-', 'D+', 'D'])`
- [✓ `my_categories = pd.CategoricalDtype(categories=['D', 'D+', 'C-', 'C', 'C+', 'B-', 'B', 'B+', 'A-', 'A', 'A+'], ordered=True)`]

Q4

Consider the DataFrame `df` shown in the image below. Which of the following can return the head of the pivot table as shown in the image below `df`?



df

	world_rank	institution	country	Rank_Level
0	1	Harvard University	USA	First Tier Top Unversity
1	2	Massachusetts Institute of Technology	USA	First Tier Top Unversity
2	3	Stanford University	USA	First Tier Top Unversity
3	4	University of Cambridge	United Kingdom	First Tier Top Unversity
4	5	California Institute of Technology	USA	First Tier Top Unversity

pivot table

	median				
Rank_Level	**First Tier Top Unversity**	**Other Top Unversity**	**Second Tier Top Unversity**	**Third Tier Top Unversity**	**All**
country					
Argentina	NaN	44.390	NaN	NaN	44.390
Australia	48.055	44.580	49.125	47.285	44.765
Austria	NaN	44.630	NaN	47.030	44.690
Belgium	51.875	44.715	49.600	46.890	46.210
Brazil	NaN	44.365	49.565	NaN	44.380

- ○ `df.pivot_table(values='score', index='Rank_Level', columns='country', aggfunc=[np.median], margins=True)`
- ○ `df.pivot_table(values='score', index='Rank_Level', columns='country', aggfunc=[np.median])`
- [✓ `df.pivot_table(values='score', index='country', columns='Rank_Level', aggfunc=[np.median])`]
- ○ `df.pivot_table(values='score', index='country', columns='Rank_Level', aggfunc=[np.median], margins=True)`

Q5

Assume that the date '11/29/2019' in MM/DD/YYYY format is the 4th day of the week, what will be the result of the following?

```
import pandas as pd
(pd.Timestamp('11/29/2019') + pd.offsets.MonthEnd()).weekday()
```

- [✓ 5]
- ○ 4
- ○ 6
- ○ 7

Q6

Consider a DataFrame `df`. We want to create groups based on the column `group_key` in the DataFrame and fill the `nan` values with group means using:

```
filling_mean = lambda g: g.fillna(g.mean())
```

Which of the following is correct for performing this task?

- ○ `df.groupby(group_key).transform(filling_mean)`
- [✓ `df.groupby(group_key).aggregate(filling_mean)`]
- ○ `df.groupby(group_key).apply(filling_mean)`
- ○ `df.groupby(group_key).filling_mean()`

Q7

`student_df`

	First Name	Last Name	School
0	James	Hammond	Business
1	Mike	Smith	Law
2	Sally	Brooks	Engineering

`staff_df`

	First Name	Last Name	Role
0	Kelly	Desjardins	Director of HR
1	Sally	Brooks	Course liasion
2	James	Wilde	Grader

Consider the DataFrames above, both of which have a standard integer based index. Which of the following can be used to get the data of all students (from `student_df`) and merge it with their staff roles where `nan` denotes no role?

- ○ `result_df = pd.merge(student_df, staff_df, how='right', on=['First Name', 'Last Name'])`
- [✓ `result_df = pd.merge(staff_df, student_df, how='right', on=['First Name', 'Last Name'])`]
- ○ `result_df = pd.merge(staff_df, student_df, how='outer', on=['First Name', 'Last Name'])`
- ○ `result_df = pd.merge(student_df, staff_df, how='inner', on=['First Name', 'Last Name'])`

Q8

Consider a DataFrame `df` with columns `name`, `reviews_per_month`, and `review_scores_value`. This DataFrame also consists of several missing values. Which of the following can be used to:
i) calculate the number of entries in the `name` column, and
ii) calculate the mean and standard deviation of the `reviews_per_month`, grouping by different `review_scores_value`?

- ○ `df.groupby('review_scores_value').agg({'name': len, 'reviews_per_month': (np.mean, np.std)})`
- ○ `df.agg({'name': len, 'reviews_per_month': (np.nanmean, np.nanstd)}`
- [✓ `df.groupby('review_scores_value').agg({'name': len, 'reviews_per_month': (np.nanmean, np.nanstd)})`]
- ○ `df.agg({'name': len, 'reviews_per_month': (np.mean, np.std)}`

Q9

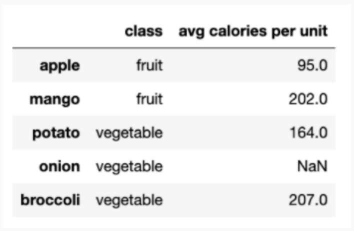
What will be the result of the following code?:

```
import pandas as pd
pd.Period('01/12/2019', 'M') + 5
```

- ○ `Period('2019-12', 'M')`
- ○ `Period('2019-12-01', 'D')`
- [✓ `Period('2019-06', 'M')`]
- ○ `Period('2019-12-06', 'D')`

Q10

Which of the following is **not** a valid expression to create a Pandas GroupBy object from the DataFrame shown below?



	class	avg calories per unit
apple	fruit	95.0
mango	fruit	202.0
potato	vegetable	164.0
onion	vegetable	NaN
broccoli	vegetable	207.0

- ○ `df.groupby('class', axis = 0)`
- [✓ `df.groupby('vegetable')`]
- ○ `df.groupby('class')`
- ○ `grouped = df.groupby(['class', 'avg calories per unit'])`