Step-1:Import Packages

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Step-2: Read the data

In [2]:
```
path='Visadataset.csv'
visa_df=pd.read_csv(path)
visa_df
```

Out[2]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_t
0	EZYV01	Asia	High School	N	
1	EZYV02	Asia	Master's	Y	
2	EZYV03	Asia	Bachelor's	N	
3	EZYV04	Asia	Bachelor's	N	
4	EZYV05	Africa	Master's	Y	
...	...	...	...	...	
25475	EZYV25476	Asia	Bachelor's	Y	
25476	EZYV25477	Asia	High School	Y	
25477	EZYV25478	Asia	Master's	Y	
25478	EZYV25479	Asia	Master's	Y	
25479	EZYV25480	Asia	Bachelor's	Y	

25480 rows × 12 columns

In [17]:
```
path='bank.csv'
bank_df=pd.read_csv(path,sep=';')
bank_df
```

Out[17]:

	age	job	marital	education	default	balance	housing	loan	contact
0	30	unemployed	married	primary	no	1787	no	no	cellular
1	33	services	married	secondary	no	4789	yes	yes	cellular
2	35	management	single	tertiary	no	1350	yes	no	cellular
3	30	management	married	tertiary	no	1476	yes	yes	unknown
4	59	blue-collar	married	secondary	no	0	yes	no	unknown
...	...	...	...	...	...	...	...	...	...
4516	33	services	married	secondary	no	-333	yes	no	cellular
4517	57	self-employed	married	tertiary	yes	-3313	yes	yes	unknown
4518	57	technician	married	secondary	no	295	no	no	cellular
4519	28	blue-collar	married	secondary	no	1137	no	no	cellular
4520	44	entrepreneur	single	tertiary	no	1136	yes	yes	cellular

4521 rows × 17 columns

Step-3:DATA qucik checks Visadf

- columns
- shape
- dtype
- len
- head
- tail
- info
- isnull

In [20]:
```
visa_df.columns
```

Out[20]:
```
Index(['case_id', 'continent', 'education_of_employee', 'has_job_experience',
       'requires_job_training', 'no_of_employees', 'yr_of_estab',
       'region_of_employment', 'prevailing_wage', 'unit_of_wage',
       'full_time_position', 'case_status'],
      dtype='object')
```

In [24]:
```
visa_df.shape
print('The number of rows are:',visa_df.shape[0])
print('The number of columns are:',visa_df.shape[1])
```

```
The number of rows are: 25480
The number of columns are: 12
```

In [26]:
```
visa_df.dtypes
```

Out[26]:
```
case_id                 object
continent               object
education_of_employee   object
has_job_experience      object
requires_job_training   object
no_of_employees          int64
yr_of_estab              int64
region_of_employment    object
prevailing_wage        float64
unit_of_wage            object
full_time_position      object
case_status             object
dtype: object
```

In [28]:
```
types=visa_df.dtypes
types
```

Out[28]:
```
case_id                 object
continent               object
education_of_employee   object
has_job_experience      object
requires_job_training   object
no_of_employees          int64
yr_of_estab              int64
region_of_employment    object
prevailing_wage        float64
unit_of_wage            object
full_time_position      object
case_status             object
dtype: object
```

Method-1

In [45]:
```
type_df=pd.DataFrame(types).reset_index()
print(type_df.columns) # 2
type_df.columns=['Column Name','Data type']
type_df
```

```
Index(['index', 0], dtype='object')
```

Out[45]:

	Column Name	Data type
0	case_id	object
1	continent	object
2	education_of_employee	object
3	has_job_experience	object
4	requires_job_training	object
5	no_of_employees	int64
6	yr_of_estab	int64
7	region_of_employment	object
8	prevailing_wage	float64
9	unit_of_wage	object
10	full_time_position	object
11	case_status	object

In [43]:
```
types
```

Out[43]:
```
case_id                 object
continent               object
education_of_employee   object
has_job_experience      object
requires_job_training   object
no_of_employees          int64
yr_of_estab              int64
region_of_employment    object
prevailing_wage        float64
unit_of_wage            object
full_time_position      object
case_status             object
dtype: object
```

In [47]:
```
# types is a series it has both keys and values
types.keys()
```

Out[47]:
```
Index(['case_id', 'continent', 'education_of_employee', 'has_job_experience',
       'requires_job_training', 'no_of_employees', 'yr_of_estab',
       'region_of_employment', 'prevailing_wage', 'unit_of_wage',
       'full_time_position', 'case_status'],
      dtype='object')
```

In [55]:
```
types.values
```

Out[55]:
```
array([dtype('O'), dtype('O'), dtype('O'), dtype('O'), dtype('O'),
       dtype('int64'), dtype('int64'), dtype('O'), dtype('float64'),
       dtype('O'), dtype('O'), dtype('O')], dtype=object)
```

In [63]:
```
dir(types)
```

Out[63]:
```
['T',
 '_AXIS_LEN',
 '_AXIS_ORDERS',
 '_AXIS_TO_AXIS_NUMBER',
 '_HANDLED_TYPES',
 '__abs__',
 '__add__',
 '__and__',
 '__annotations__',
 '__array__',
 '__array_priority__',
 '__array_ufunc__',
 '__bool__',
 '__class__',
 '__column_consortium_standard__',
 '__contains__',
 '__copy__',
 '__deepcopy__',
 '__delattr__',
 '__delitem__',
 '__dict__',
 '__dir__',
 '__divmod__',
 '__doc__',
 '__eq__',
 '__finalize__',
 '__float__',
 '__floordiv__',
 '__format__',
 '__ge__',
 '__getattr__',
 '__getattribute__',
 '__getitem__',
 '__getstate__',
 '__gt__',
 '__hash__',
 '__iadd__',
 '__iand__',
 '__ifloordiv__',
 '__imod__',
 '__imul__',
 '__init__',
 '__init_subclass__',
 '__int__',
 '__invert__',
 '__ior__',
 '__ipow__',
 '__isub__',
 '__iter__',
 '__itruediv__',
 '__ixor__',
 '__le__',
 '__len__',
 '__lt__',
 '__matmul__',
 '__mod__',
 '__module__',
 '__mul__',
 '__ne__',
 '__neg__',
```

```
'__new__',
'__nonzero__',
'__or__',
'__pandas_priority__',
'__pos__',
'__pow__',
'__radd__',
'__rand__',
'__rdivmod__',
'__reduce__',
'__reduce_ex__',
'__repr__',
'__rfloordiv__',
'__rmatmul__',
'__rmod__',
'__rmul__',
'__ror__',
'__round__',
'__rpow__',
'__rsub__',
'__rtruediv__',
'__rxor__',
'__setattr__',
'__setitem__',
'__setstate__',
'__sizeof__',
'__str__',
'__sub__',
'__subclasshook__',
'__truediv__',
'__weakref__',
'__xor__',
'_accessors',
'_accum_func',
'_agg_examples_doc',
'_agg_see_also_doc',
'_align_for_op',
'_align_frame',
'_align_series',
'_append',
'_arith_method',
'_as_manager',
'_attrs',
'_binop',
'_can_hold_na',
'_check_inplace_and_allows_duplicate_labels',
'_check_is_chained_assignment_possible',
'_check_label_or_level_ambiguity',
'_check_setitem_copy',
'_clear_item_cache',
'_clip_with_one_bound',
'_clip_with_scalar',
'_cmp_method',
'_consolidate',
'_consolidate_inplace',
'_construct_axes_dict',
'_construct_result',
'_constructor',
'_constructor_expanddim',
'_constructor_expanddim_from_mgr',
```

```
'_constructor_from_mgr',
'_data',
'_deprecate_downcast',
'_dir_additions',
'_dir_deletions',
'_drop_axis',
'_drop_labels_or_levels',
'_duplicated',
'_find_valid_index',
'_flags',
'_flex_method',
'_from_mgr',
'_get_axis',
'_get_axis_name',
'_get_axis_number',
'_get_axis_resolvers',
'_get_block_manager_axis',
'_get_bool_data',
'_get_cacher',
'_get_cleaned_column_resolvers',
'_get_index_resolvers',
'_get_label_or_level_values',
'_get_numeric_data',
'_get_rows_with_mask',
'_get_value',
'_get_values_tuple',
'_get_with',
'_getitem_slice',
'_gotitem',
'_hidden_attrs',
'_indexed_same',
'_info_axis',
'_info_axis_name',
'_info_axis_number',
'_init_dict',
'_init_mgr',
'_inplace_method',
'_internal_names',
'_internal_names_set',
'_is_cached',
'_is_copy',
'_is_label_or_level_reference',
'_is_label_reference',
'_is_level_reference',
'_is_mixed_type',
'_is_view',
'_is_view_after_cow_rules',
'_item_cache',
'_ixs',
'_logical_func',
'_logical_method',
'_map_values',
'_maybe_update_cacher',
'_memory_usage',
'_metadata',
'_mgr',
'_min_count_stat_function',
'_name',
'_needs_reindex_multi',
'_pad_or_backfill',
```

```
'_protect_consolidate',
'_reduce',
'_references',
'_reindex_axes',
'_reindex_indexer',
'_reindex_multi',
'_reindex_with_indexers',
'_rename',
'_replace_single',
'_repr_data_resource_',
'_repr_latex_',
'_reset_cache',
'_reset_cacher',
'_set_as_cached',
'_set_axis',
'_set_axis_name',
'_set_axis_nocheck',
'_set_is_copy',
'_set_labels',
'_set_name',
'_set_value',
'_set_values',
'_set_with',
'_set_with_engine',
'_shift_with_freq',
'_slice',
'_stat_function',
'_stat_function_ddof',
'_take_with_is_copy',
'_to_latex_via_styler',
'_typ',
'_update_inplace',
'_validate_dtype',
'_values',
'_where',
'abs',
'add',
'add_prefix',
'add_suffix',
'agg',
'aggregate',
'align',
'all',
'any',
'apply',
'argmax',
'argmin',
'argsort',
'array',
'asfreq',
'asof',
'astype',
'at',
'at_time',
'attrs',
'autocorr',
'axes',
'backfill',
'between',
'between_time',
```

```
'bfill',
'bool',
'case_id',
'case_status',
'case_when',
'clip',
'combine',
'combine_first',
'compare',
'continent',
'convert_dtypes',
'copy',
'corr',
'count',
'cov',
'cummax',
'cummin',
'cumprod',
'cumsum',
'describe',
'diff',
'div',
'divide',
'divmod',
'dot',
'drop',
'drop_duplicates',
'droplevel',
'dropna',
'dtype',
'dtypes',
'duplicated',
'education_of_employee',
'empty',
'eq',
'equals',
'ewm',
'expanding',
'explode',
'factorize',
'ffill',
'fillna',
'filter',
'first',
'first_valid_index',
'flags',
'floordiv',
'full_time_position',
'ge',
'get',
'groupby',
'gt',
'has_job_experience',
'hasnans',
'head',
'hist',
'iat',
'idxmax',
'idxmin',
'iloc',
```

```
'index',
'infer_objects',
'info',
'interpolate',
'is_monotonic_decreasing',
'is_monotonic_increasing',
'is_unique',
'isin',
'isna',
'isnull',
'item',
'items',
'keys',
'kurt',
'kurtosis',
'last',
'last_valid_index',
'le',
'list',
'loc',
'lt',
'map',
'mask',
'max',
'mean',
'median',
'memory_usage',
'min',
'mod',
'mode',
'mul',
'multiply',
'name',
'nbytes',
'ndim',
'ne',
'nlargest',
'no_of_employees',
'notna',
'notnull',
'nsmallest',
'nunique',
'pad',
'pct_change',
'pipe',
'plot',
'pop',
'pow',
'prevailing_wage',
'prod',
'product',
'quantile',
'radd',
'rank',
'ravel',
'rdiv',
'rdivmod',
'region_of_employment',
'reindex',
'reindex_like',
```

```
'rename',
'rename_axis',
'reorder_levels',
'repeat',
'replace',
'requires_job_training',
'resample',
'reset_index',
'rfloordiv',
'rmod',
'rmul',
'rolling',
'round',
'rpow',
'rsub',
'rtruediv',
'sample',
'searchsorted',
'sem',
'set_axis',
'set_flags',
'shape',
'shift',
'size',
'skew',
'sort_index',
'sort_values',
'squeeze',
'std',
'str',
'struct',
'sub',
'subtract',
'sum',
'swapaxes',
'swaplevel',
'tail',
'take',
'to_clipboard',
'to_csv',
'to_dict',
'to_excel',
'to_frame',
'to_hdf',
'to_json',
'to_latex',
'to_list',
'to_markdown',
'to_numpy',
'to_period',
'to_pickle',
'to_sql',
'to_string',
'to_timestamp',
'to_xarray',
'transform',
'transpose',
'truediv',
'truncate',
'tz_convert',
```

```
 'tz_localize',
 'unique',
 'unit_of_wage',
 'unstack',
 'update',
 'value_counts',
 'values',
 'var',
 'view',
 'where',
 'xs',
 'yr_of_estab']
```

In [77]:
```
# step-1:
types=visa_df.dtypes
# step-2:
keys=types.keys().tolist()
# list(types.keys())
# step-3:
values=types.values.tolist()
# when we have two lists are avaialble
type_df=pd.DataFrame(zip(keys,values),
            columns=['Column','Type'])
type_df.to_csv('Datatype.csv',index=False)
```

In [81]:
```
keys,values
```

Out[81]:
```
(['case_id',
  'continent',
  'education_of_employee',
  'has_job_experience',
  'requires_job_training',
  'no_of_employees',
  'yr_of_estab',
  'region_of_employment',
  'prevailing_wage',
  'unit_of_wage',
  'full_time_position',
  'case_status'],
 [dtype('O'),
  dtype('O'),
  dtype('O'),
  dtype('O'),
  dtype('O'),
  dtype('int64'),
  dtype('int64'),
  dtype('O'),
  dtype('float64'),
  dtype('O'),
  dtype('O'),
  dtype('O')])
```

In [87]:
```
cat=[]
num=[]
for i,j in zip(keys,values):
    if j=='object':
        cat.append(i)
    else:
        num.append(i)
```

In [89]:
```
cat
```

Out[89]:
```
['case_id',
 'continent',
 'education_of_employee',
 'has_job_experience',
 'requires_job_training',
 'region_of_employment',
 'unit_of_wage',
 'full_time_position',
 'case_status']
```

In [91]:
```
num
```

Out[91]:
```
['no_of_employees', 'yr_of_estab', 'prevailing_wage']
```

Select_dtypes

In [102…
```
cat=visa_df.select_dtypes(include='object').columns
num=visa_df.select_dtypes(exclude='object').columns
```

Task-2:

- I want to how many memebrs are available in continent

In [117…
```
# step-1: Main dataframe
visa_df
# step-2: select continent column
visa_df['continent']
# step-3: apply condition
con=visa_df['continent']=='Asia'
# step-4: extract the True values
len(visa_df[con])
```

Out[117…
```
16861
```

In [ ]:
```
con=visa_df['continent']=='Asia'
len(visa_df[con])
##########################################
con=visa_df['continent']=='Africa'
len(visa_df[con])
```

unique

In [120…
```
visa_df['continent'].unique()
```

Out[120…
```
array(['Asia', 'Africa', 'North America', 'Europe', 'South America',
       'Oceania'], dtype=object)
```

nunique

In [123…
```
visa_df['continent'].nunique()
```

Out[123…
```
6
```

In [ ]:
```
con=visa_df['continent']=='Asia'
len(visa_df[con])
```

```
##########################################
con=visa_df['continent']=='Africa'
len(visa_df[con])
##########################################
con=visa_df['continent']=='North America'
len(visa_df[con])
con=visa_df['continent']=='Europe'
len(visa_df[con])
con=visa_df['continent']=='South America'
len(visa_df[con])
con=visa_df['continent']=='Oceania'
len(visa_df[con])

con=visa_df['continent']==i
len(visa_df[con])
```

In [127…
```
labels=visa_df['continent'].unique()
for i in labels:
    con=visa_df['continent']==i
    count=len(visa_df[con])
    print(count)
```

```
16861
551
3292
3732
852
192
```

In [7]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns


path='Visadataset.csv'
visa_df=pd.read_csv(path)
visa_df

cat=visa_df.select_dtypes(include='object').columns
num=visa_df.select_dtypes(exclude='object').columns
```

raw data

In [16]:
```
visa_df # df
visa_df['continent'] # series
visa_df['continent'].values # array of values
```

Out[16]:
```
array(['Asia', 'Asia', 'Asia', ..., 'Asia', 'Asia', 'Asia'], dtype=object)
```

frequency tabel

In [25]:
```
con=visa_df['continent']=='Asia'
len(visa_df[con])
```

Out[25]:
```
16861
```

In [27]:
```
visa_df['continent'].unique()
```

Out[27]:
```
array(['Asia', 'Africa', 'North America', 'Europe', 'South America',
       'Oceania'], dtype=object)
```

In [29]:
```
labels=visa_df['continent'].unique()
count=[]
for i in labels:
    con=visa_df['continent']==i
    count.append(len(visa_df[con]))
count
```

Out[29]:
```
[16861, 551, 3292, 3732, 852, 192]
```

In [35]:
```
# we have two lists
# one list name: labels
# count
# if you are not able to do this
# katam zindagi ==== 1week
continent_df=pd.DataFrame(zip(labels,count),columns=['Continent','No Of Applican
```

In [39]:
```
# all together
path='Visadataset.csv'
visa_df=pd.read_csv(path)
visa_df

cat=visa_df.select_dtypes(include='object').columns
num=visa_df.select_dtypes(exclude='object').columns

labels=visa_df['continent'].unique()
count=[]
for i in labels:
    con=visa_df['continent']==i
    count.append(len(visa_df[con]))
count

continent_df=pd.DataFrame(zip(labels,count),
                          columns=['Continent','No Of Applicants'])

continent_df.to_csv('continent_df.csv',index=False)
```

In [41]:
```
continent_df
```

Out[41]:

	Continent	No Of Applicants
0	Asia	16861
1	Africa	551
2	North America	3292
3	Europe	3732
4	South America	852
5	Oceania	192

value_counts

In [ ]:
```
```

In [43]:
```
visa_df['continent'].value_counts()
```

Out[43]:
```
continent
Asia             16861
Europe            3732
North America     3292
South America      852
Africa             551
Oceania            192
Name: count, dtype: int64
```

Group by

In [46]:
```
# dividing continents into a group
# then calculate size of each group
visa_df.groupby('continent').size()
```

Out[46]:
```
continent
Africa             551
Asia             16861
Europe            3732
North America     3292
Oceania            192
South America      852
dtype: int64
```

In [49]:
```
continent_keys=visa_df['continent'].value_counts().keys()
continent_values=visa_df['continent'].value_counts().values
continent_df=pd.DataFrame(zip(continent_keys,continent_values),
             columns=['Continent','No Of Applicants'])
continent_df
```

Out[49]:

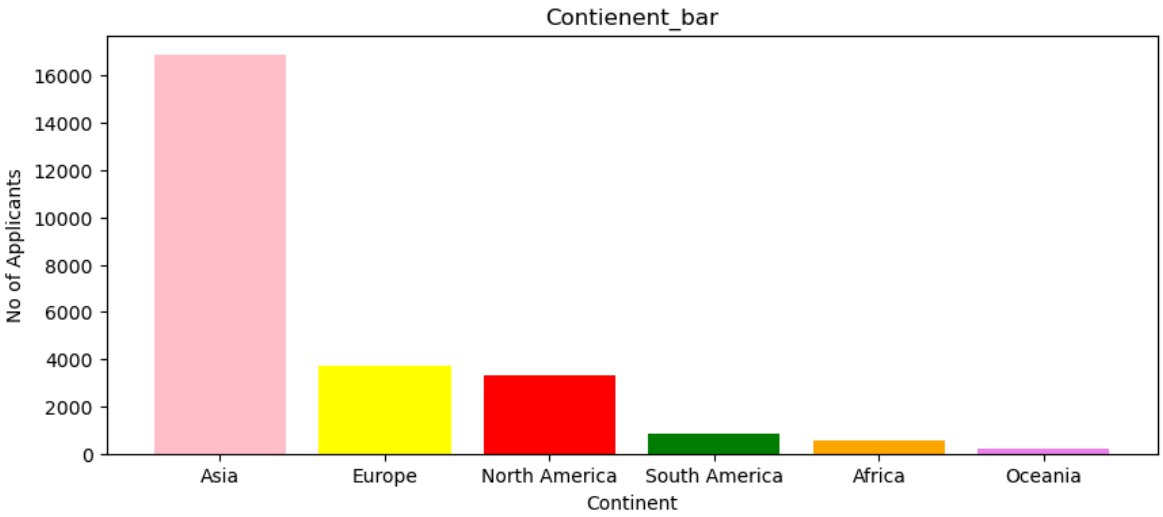
	Continent	No Of Applicants
0	Asia	16861
1	Europe	3732
2	North America	3292
3	South America	852
4	Africa	551
5	Oceania	192

Bar chart

- under matplotlib
- plt.bar

In [75]:
```
continent_keys=visa_df['continent'].value_counts().keys()
continent_values=visa_df['continent'].value_counts().values
col=['pink','yellow','red','green','orange','violet']
plt.figure(figsize=(10,4))
plt.bar(continent_keys,
        continent_values,
```

```
       color=col)
plt.xlabel('Continent')
plt.ylabel('No of Applicants')
plt.title('Contienent_bar')
plt.savefig('Contienent_bar.jpg')
plt.savefig('Contienent_bar.png')
plt.show()
```



Relative frequency

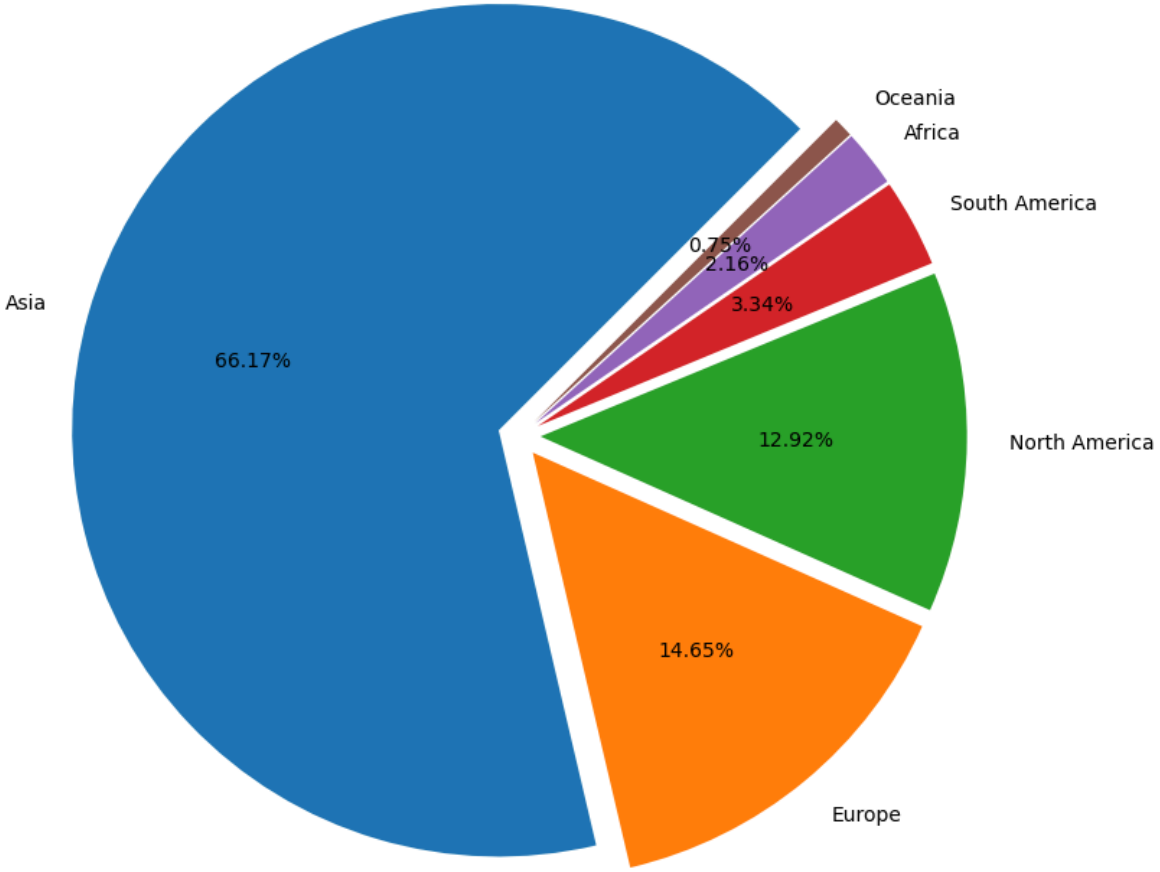
In [82]:
```
visa_df['continent'].value_counts(normalize=True)*100
```

Out[82]:
```
continent
Asia              66.173469
Europe            14.646782
North America     12.919937
South America      3.343799
Africa             2.162480
Oceania            0.753532
Name: proportion, dtype: float64
```

pie

In [113…
```
continent_keys=visa_df['continent'].value_counts().keys()
continent_values=visa_df['continent'].value_counts().values
plt.pie(continent_values,
        explode=[0.1,0.1,0.1,0.1,0.1,0.1],
        labels=continent_keys,
       autopct="%0.2f%%",
       startangle=45,
       radius=2)

plt.show()
```


Oceania
Africa
South America
0.75%
2.16%
3.34%
Asia
66.17%
12.92%
North America
14.65%
Europe

In [ ]: