💻 Question No: 01

⚙️ Setup

- Ensure the Python kernel has the necessary libraries: `pandas` , `matplotlib` and `lets-plot` , os
- Ensure the `bank-additional-full.csv` file is in the `data` folder.

In [2]:
```
import matplotlib.pyplot as plt
import pandas as pd
import os
os.getcwd()
import numpy as np

from lets_plot import * # This imports all of ggplot2's functions
LetsPlot.setup_html()
```

In [3]:
```
df = pd.read_csv('D:/Data Science for Marketing-I/data/bank-additional-full.csv',se
```

In [4]:
```
df
```

Out[4]:

	age	job	marital	education	default	housing	loan	contact	m
0	56	housemaid	married	basic.4y	no	no	no	telephone	
1	57	services	married	high.school	unknown	no	no	telephone	
2	37	services	married	high.school	no	yes	no	telephone	
3	40	admin.	married	basic.6y	no	no	no	telephone	
4	56	services	married	high.school	no	no	yes	telephone	
...	...	...	...	...	...	...	...	...	
41183	73	retired	married	professional.course	no	yes	no	cellular	
41184	46	blue-collar	married	professional.course	no	no	no	cellular	
41185	56	retired	married	university.degree	no	yes	no	cellular	
41186	44	technician	married	professional.course	no	no	no	cellular	
41187	74	retired	married	professional.course	no	yes	no	cellular	

41188 rows × 21 columns

i. Perform the basic analysis. What kind of insights do they provide?

In [5]:
```
df.shape
```

Out[5]: (41188, 21)

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   age             41188 non-null  int64  
 1   job             41188 non-null  object 
 2   marital         41188 non-null  object 
 3   education       41188 non-null  object 
 4   default         41188 non-null  object 
 5   housing         41188 non-null  object 
 6   loan            41188 non-null  object 
 7   contact         41188 non-null  object 
 8   month           41188 non-null  object 
 9   day_of_week     41188 non-null  object 
 10  duration        41188 non-null  int64  
 11  campaign        41188 non-null  int64  
 12  pdays           41188 non-null  int64  
 13  previous        41188 non-null  int64  
 14  poutcome        41188 non-null  object 
 15  emp.var.rate    41188 non-null  float64
 16  cons.price.idx  41188 non-null  float64
 17  cons.conf.idx   41188 non-null  float64
 18  euribor3m       41188 non-null  float64
 19  nr.employed     41188 non-null  float64
 20  y               41188 non-null  object 
dtypes: float64(5), int64(5), object(11)
memory usage: 6.6+ MB
```

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

Out[7]:

	age	job	marital	education	default	housing	loan	contact	month	day_c
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	
1	57	services	married	high.school	unknown	no	no	telephone	may	
2	37	services	married	high.school	no	yes	no	telephone	may	
3	40	admin.	married	basic.6y	no	no	no	telephone	may	
4	56	services	married	high.school	no	no	yes	telephone	may	

5 rows × 21 columns

ii. Create a new column named “Conversion” by transforming categorical values in the variable “y” into numerical representations, and why is this transformation important in data analysis?

In [8]:
```
df['conversion']=df['y'].apply(lambda x:1 if x=='yes' else 0)
df
```

Out[8]:

	age	job	marital	education	default	housing	loan	contact	m
0	56	housemaid	married	basic.4y	no	no	no	telephone	
1	57	services	married	high.school	unknown	no	no	telephone	
2	37	services	married	high.school	no	yes	no	telephone	
3	40	admin.	married	basic.6y	no	no	no	telephone	
4	56	services	married	high.school	no	no	yes	telephone	
...	...	...	...	...	...	...	...	...	
41183	73	retired	married	professional.course	no	yes	no	cellular	
41184	46	blue-collar	married	professional.course	no	no	no	cellular	
41185	56	retired	married	university.degree	no	yes	no	cellular	
41186	44	technician	married	professional.course	no	no	no	cellular	
41187	74	retired	married	professional.course	no	yes	no	cellular	

41188 rows × 22 columns

💡 The code creates a new binary column 'conversion' where 1 represents 'yes' and 0 represents any other value in the 'y' column. This is a common technique for encoding categorical data into a numerical format.

iii. Describe how the Aggregate Conversion Rate is calculated and interpret its significance in the context of the dataset.

Aggregate Conversion Rate

In [9]:
```
df['conversion'].value_counts()
```

Out[9]:
```
conversion
0    36548
1     4640
Name: count, dtype: int64
```

In [10]:
```
df['conversion'].sum()/df['conversion'].count()*100
```

Out[10]: np.float64(11.265417111780131)

💡 The Aggregate Conversion Rate is the percentage of "yes" responses (conversions) out of the total entries. It helps measure the overall success rate of the campaign.

In [11]:
```
df.groupby('conversion')['age'].count()/(df['conversion'].count())*100
```

Out[11]:
```
conversion
0    88.734583
1    11.265417
Name: age, dtype: float64
```

💡 This code calculates percentage distribution of the 'age' column across the unique values of the 'conversion' column. In this case:

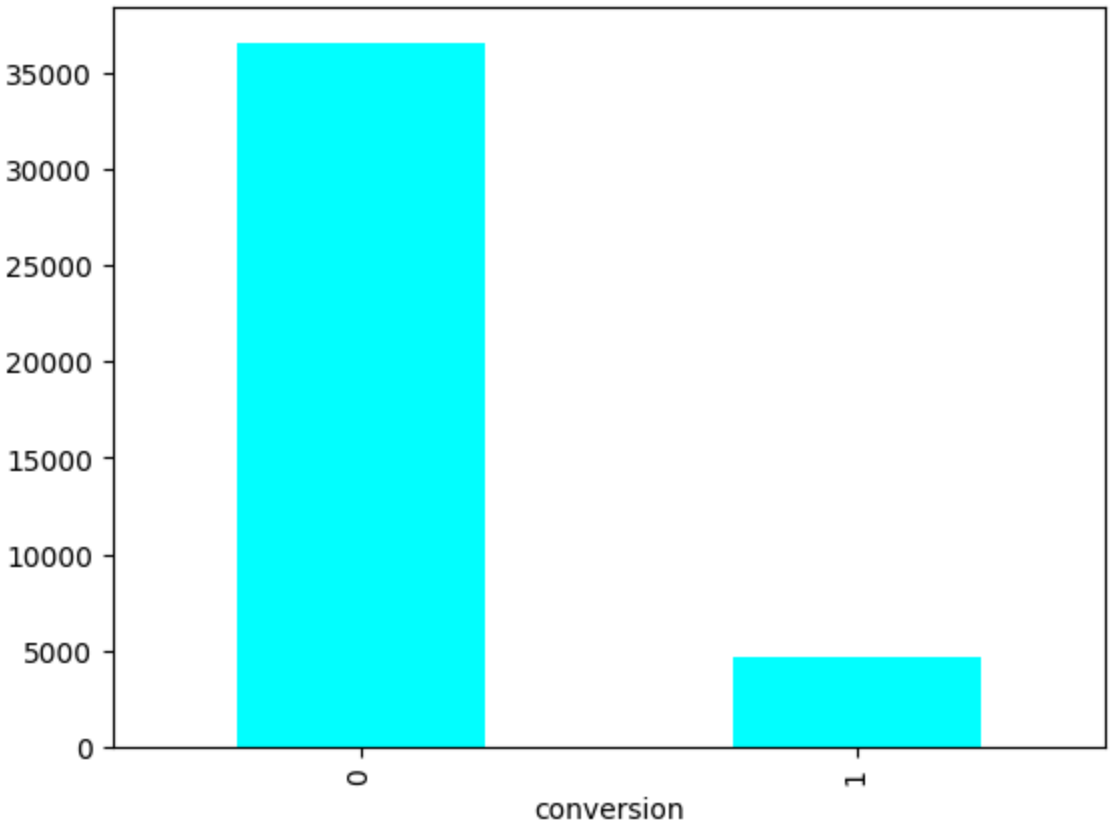
88.73% of the data corresponds to conversion = 0.

11.27% of the data corresponds to conversion = 1.

iv. What is the purpose of plotting the conversion data using a bar chart, and how does the code achieve this visualization?

In [12]:
```
df.groupby('conversion').count()['age'].plot(kind='bar',
    color='cyan',)
```

Out[12]: <Axes: xlabel='conversion'>



💡 The graph shows no.of.users or items in two categories: non-converted and converted.The taller bar (around 30,000) represents the non-converted group, meaning a large majority did not convert.The shorter bar (around 5,000) represents the converted group, meaning a smaller portion did convert.

This indicates that conversion rates are low, with only a small fraction of the total achieving conversion.

v. How can conversion rates by the number of contacts be calculated and visualized in a dataset

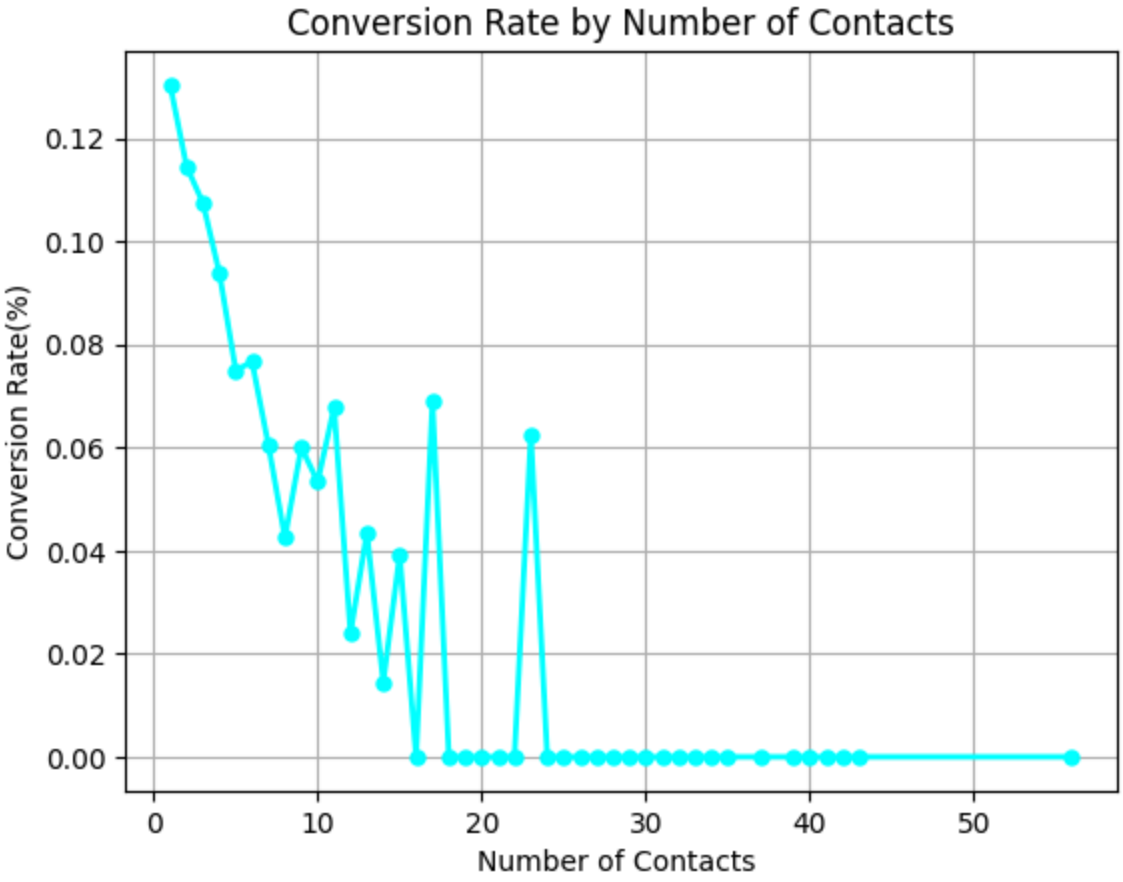
In [13]:
```
conversions_by_contacts = df.groupby(
    by='campaign'
)['conversion'].sum() / df.groupby(
    by='campaign'
)['conversion'].count() * 100
pd.DataFrame(conversions_by_contacts)
```

Out[13]:

campaign	conversion
1	13.037071
2	11.456954
3	10.747051
4	9.392682
5	7.504690
6	7.660878
7	6.041335
8	4.250000
9	6.007067
10	5.333333
11	6.779661
12	2.400000
13	4.347826
14	1.449275
15	3.921569
16	0.000000
17	6.896552
18	0.000000
19	0.000000
20	0.000000
21	0.000000
22	0.000000
23	6.250000
24	0.000000
25	0.000000
26	0.000000
27	0.000000
28	0.000000
29	0.000000

campaign	conversion
30	0.000000
31	0.000000
32	0.000000
33	0.000000
34	0.000000
35	0.000000
37	0.000000
39	0.000000
40	0.000000
41	0.000000
42	0.000000
43	0.000000
56	0.000000

In [14]:
```
conversion_by_contacts = df.groupby('campaign')['conversion'].mean()
conversion_by_contacts.plot(kind='line', marker='o', color='cyan', markersize=5, li
plt.title("Conversion Rate by Number of Contacts")
plt.xlabel("Number of Contacts ")
plt.ylabel("Conversion Rate(%)")
plt.grid(True)
plt.show()
```



💡 The data shows a positive correlation between the number of contacts and conversion rates. While increasing contacts improves conversions, it's important to find the optimal number of contacts to balance effort and results.Beyond a certain point (e.g., 50 contacts),the conversion rate may not increase significantly, so further efforts might not be cost-effective.

vi. How are age groups created using a lambda function in a dataset, and why is grouping data into age ranges beneficial for analysis?

In [15]:
```
df['age_group']=df['age'].apply(lambda x:'-30' if x<=30 else '31-40' if x<=40 else
df['age_group'].value_counts()
```

Out[15]:
```
age_group
31-40    16385
41-50    10240
-30       7383
51-60     6270
60+        910
Name: count, dtype: int64
```

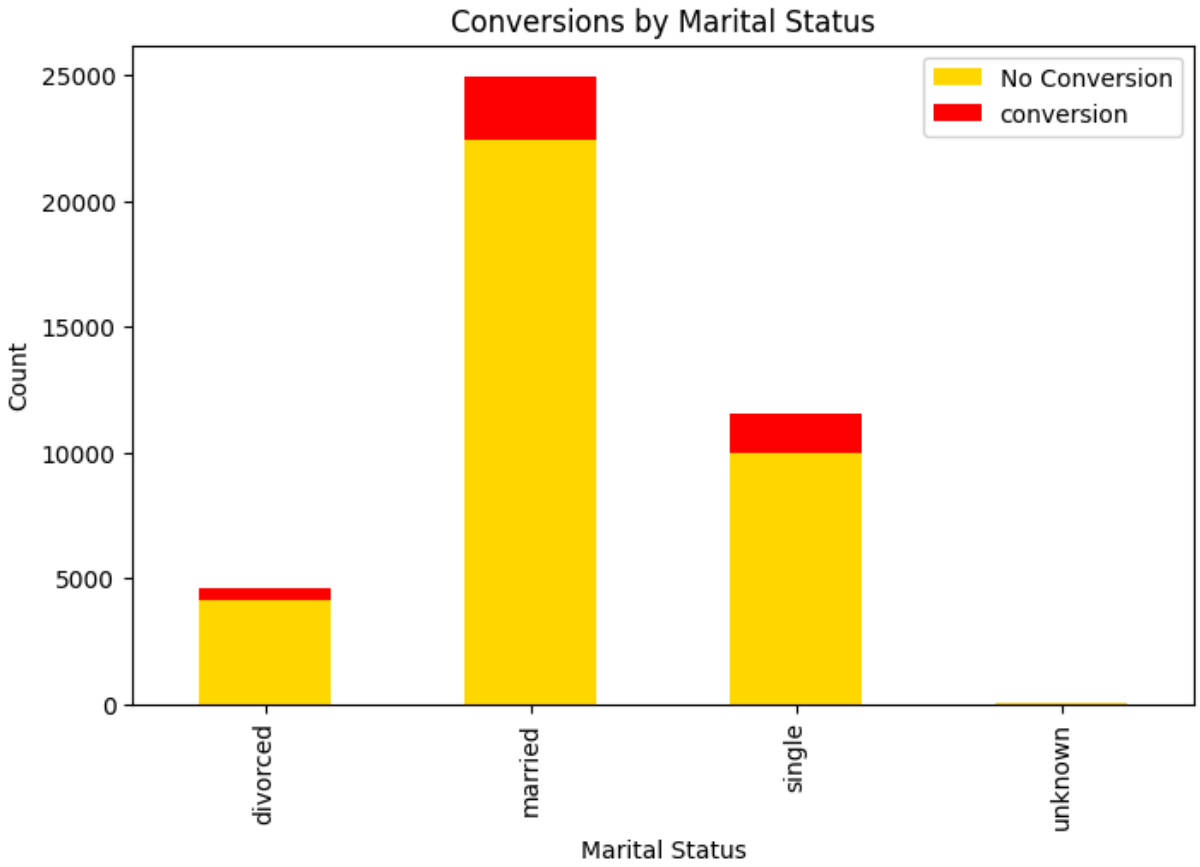
In [16]:
```
df.groupby('age_group')['conversion'].sum()/df.groupby('age_group')['conversion'].c
```

Out[16]:
```
age_group
-30      15.224164
31-40     9.746720
41-50     8.173828
51-60    10.653907
60+      45.494505
Name: conversion, dtype: float64
```

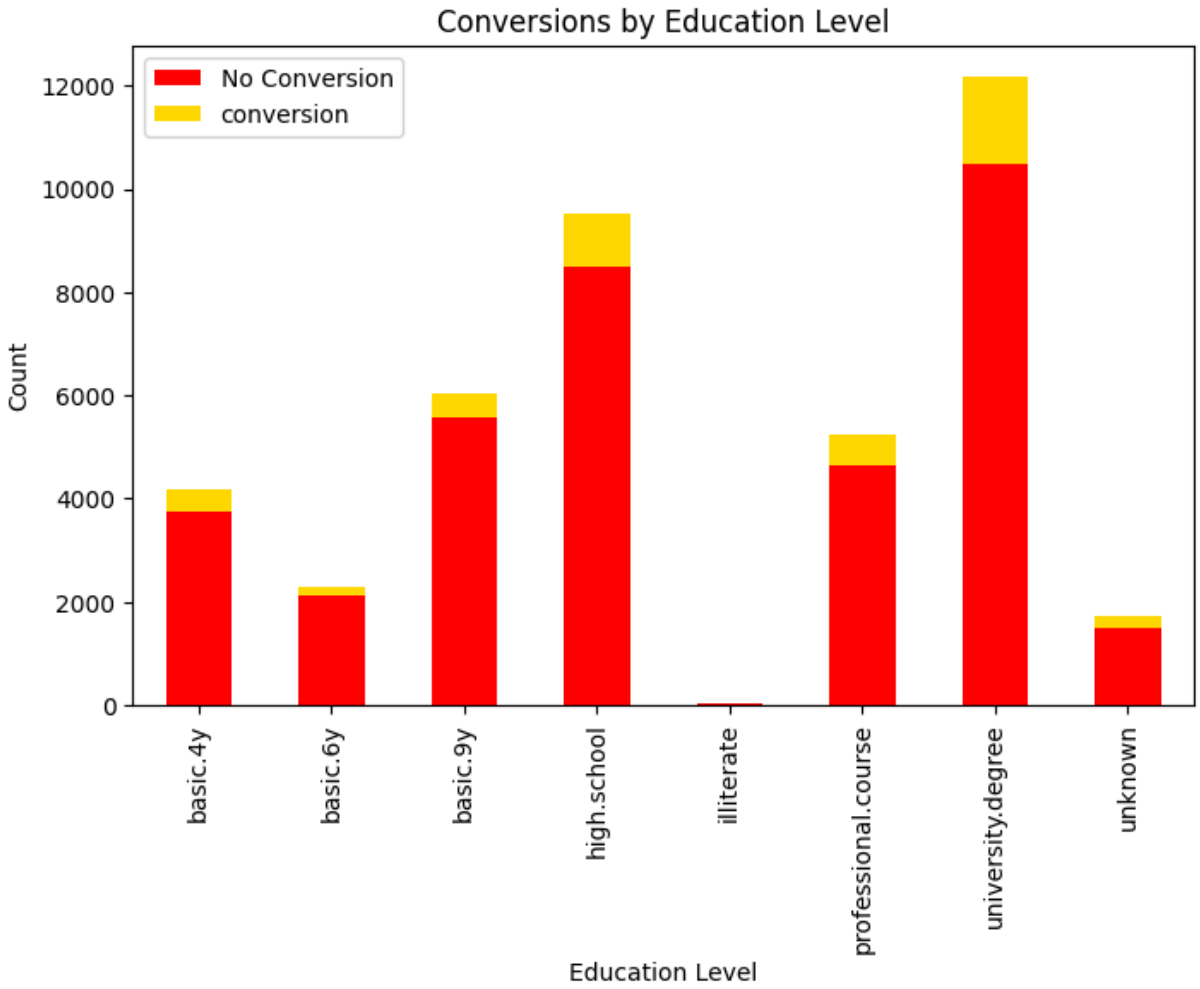
💡 The age group 31-40 has the highest count (16,385 individuals), making it the largest group.The age group 41-50 is the second-largest, with 10,240 individuals.The age group -30 (likely representing ages 0-30) has 7,383 individuals.The age group 51-60 has 6,270 individuals.The age group 60+ has the smallest count, with only 910 individuals. Grouping data into age ranges is beneficial for analysis because it helps identify trends, patterns, or behaviors within specific age groups, simplifying insights and decision-making.

vii. In an analysis comparing conversions and non-conversions by marital status, what additional insights could be explored and how would you extend the code to perform this analysis with the variable Education

In [17]:
```
marital_status_conversion = df.groupby(['marital', 'conversion']).size().unstack()
education_conversion = df.groupby(['education', 'conversion']).size().unstack()
marital_status_conversion.plot(kind='bar', stacked=True, figsize=(8, 5), color=['go
plt.title("Conversions by Marital Status")
plt.xlabel("Marital Status")
plt.ylabel("Count")
plt.legend(["No Conversion", "conversion"])
plt.show()
```



In [18]:
```
education_conversion.plot(kind='bar', stacked=True, figsize=(8, 5), color=['red', '
plt.title("Conversions by Education Level")
plt.xlabel("Education Level")
plt.ylabel("Count")
plt.legend(["No Conversion", "conversion"])
plt.show()
```



In [19]:
```
# Group by marital status and education, and calculate conversion rates
conversion_analysis = df.groupby(['marital', 'education'])['conversion'].agg(['sum'
conversion_analysis['conversion_rate'] = conversion_analysis['sum'] / conversion_an
conversion_analysis = conversion_analysis.reset_index()

# Display the results
print(conversion_analysis)
```

	marital	education	sum	count	conversion_rate
0	divorced	basic.4y	83	489	16.973415
1	divorced	basic.6y	13	182	7.142857
2	divorced	basic.9y	31	565	5.486726
3	divorced	high.school	107	1193	8.968986
4	divorced	illiterate	1	2	50.000000
5	divorced	professional.course	61	657	9.284627
6	divorced	university.degree	160	1337	11.967091
7	divorced	unknown	20	187	10.695187
8	married	basic.4y	313	3228	9.696406
9	married	basic.6y	139	1767	7.866440
10	married	basic.9y	298	4156	7.170356
11	married	high.school	475	5158	9.208996
12	married	illiterate	3	15	20.000000
13	married	professional.course	357	3156	11.311787
14	married	university.degree	821	6394	12.840163
15	married	unknown	126	1054	11.954459
16	single	basic.4y	31	453	6.843267
17	single	basic.6y	36	337	10.682493
18	single	basic.9y	142	1316	10.790274
19	single	high.school	448	3150	14.222222
20	single	illiterate	0	1	0.000000
21	single	professional.course	177	1424	12.429775
22	single	university.degree	683	4406	15.501589
23	single	unknown	103	481	21.413721
24	unknown	basic.4y	1	6	16.666667
25	unknown	basic.6y	0	6	0.000000
26	unknown	basic.9y	2	8	25.000000
27	unknown	high.school	1	14	7.142857
28	unknown	professional.course	0	6	0.000000
29	unknown	university.degree	6	31	19.354839
30	unknown	unknown	2	9	22.222222

In [20]:
```
df.groupby(['marital','education'])['conversion'].mean().unstack()
```

Out[20]:

education	basic.4y	basic.6y	basic.9y	high.school	illiterate	professional.course	univers
marital							
divorced	0.169734	0.071429	0.054867	0.089690	0.5	0.092846	
married	0.096964	0.078664	0.071704	0.092090	0.2	0.113118	
single	0.068433	0.106825	0.107903	0.142222	0.0	0.124298	
unknown	0.166667	0.000000	0.250000	0.071429	NaN	0.000000	

In [21]:
```
import seaborn as sns
```

In [22]:
```
# Create a pivot table for conversion rates
pivot_table = conversion_analysis.pivot(index='marital', columns='education', value

# Plot a heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(pivot_table, annot=True, fmt=".2f", cmap="YlGnBu", cbar_kws={'label': '
```

```
plt.title('Conversion Rates by Marital Status and Education Level')
plt.xlabel('Education Level')
plt.ylabel('Marital Status')
plt.show()
```