Dear All.

I was proposing, that we could pimp the website with more graphics.
The WHOLE CODE IS 100% based on our dataset. So it would only be a copy and paste job.
Enclosed you would see the resulting screenshots and the according code.

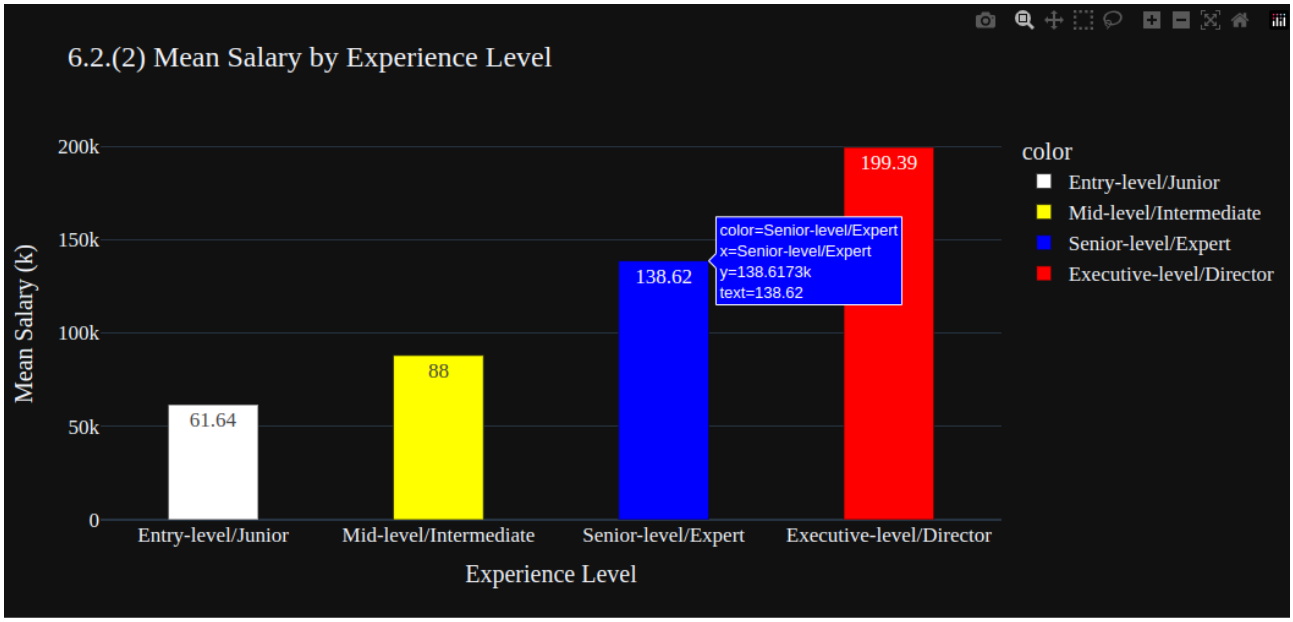
// So, first we generally prep some stuph

```
#install
!pip install country_converter

# data
import pandas as pd
import numpy as np
import country_converter as coco

# visualization
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msno
import plotly.express as px
import plotly.figure_factory as ff
import plotly.graph_objects as go
from wordcloud import WordCloud

# nltk
import nltk

# styling
%matplotlib inline
sns.set_theme(style="dark")
mpl.rcParams['axes.unicode_minus'] = False
pd.set_option('display.max_columns',None)
# plt.style.use('seaborn-dark-palette')
plt.style.use('dark_background')


# read dataframe (drop 3 columns)
df = pd.read_csv('/content/drive/MyDrive/Coding/Colab
Notebooks/data/ds_salaries.csv')
df.drop(df[['salary','salary_currency']],axis=1, inplace=True)
```

And than we go!!

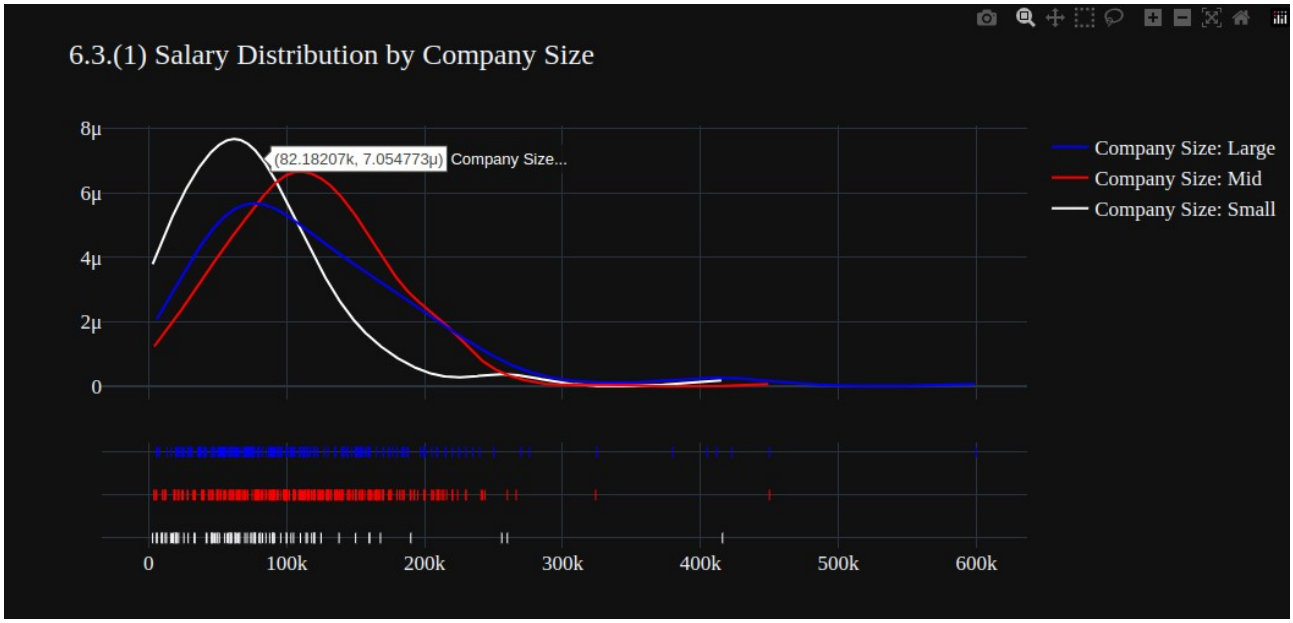

```
exlevel_salary = df[['experience_level','salary_in_usd']]

entry_salary =
exlevel_salary.loc[exlevel_salary['experience_level']=='Entry-level/Junior']
executive_salary =
exlevel_salary.loc[exlevel_salary['experience_level']=='Executive-level/
Director']
mid_salary = exlevel_salary.loc[exlevel_salary['experience_level']=='Mid-
level/Intermediate']
senior_salary =
exlevel_salary.loc[exlevel_salary['experience_level']=='Senior-level/
Expert']

hist_data =
[entry_salary['salary_in_usd'],mid_salary['salary_in_usd'],senior_salary['sa
lary_in_usd'],executive_salary['salary_in_usd']]
group_labels = ['Entry-level/Junior','Mid-level/Intermediate','Senior-
level/Expert','Executive-level/Director']
colors = ['white','yellow','blue','red']

lst = [entry_salary['salary_in_usd'].mean(),
mid_salary['salary_in_usd'].mean(),
senior_salary['salary_in_usd'].mean(),
executive_salary['salary_in_usd'].mean(),]

fig1 = ff.create_distplot(hist_data, group_labels, show_hist=False,
colors=colors)
fig2 = go.Figure(data=px.bar(x= group_labels,
y=lst,
color = group_labels,
color_discrete_sequence= colors,
title='6.2.(2) Mean Salary by Experience Level',
text = np.round([num/1000 for num in lst],2),
```

```
template = 'plotly_dark',
height=500))

fig1.layout.template = 'plotly_dark'
fig1.update_layout(title='6.2.(1) Salary Distribution by Experience
Level',font = dict(size=17,family="Franklin Gothic"))
fig2.update_traces(width=0.4)
fig2.update_layout(
xaxis_title="Experience Level",
yaxis_title="Mean Salary (k) ",
font = dict(size=17,family="Franklin Gothic"))
fig1.show()
```



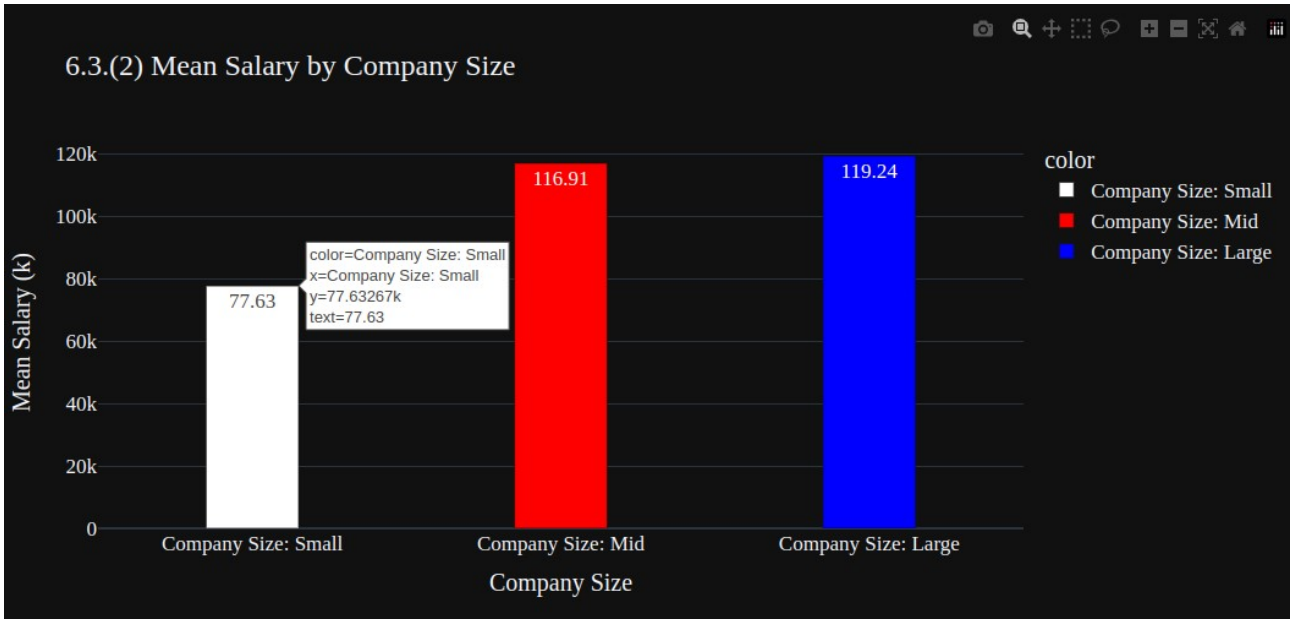
same code as above, but now:

`fig2.show()`



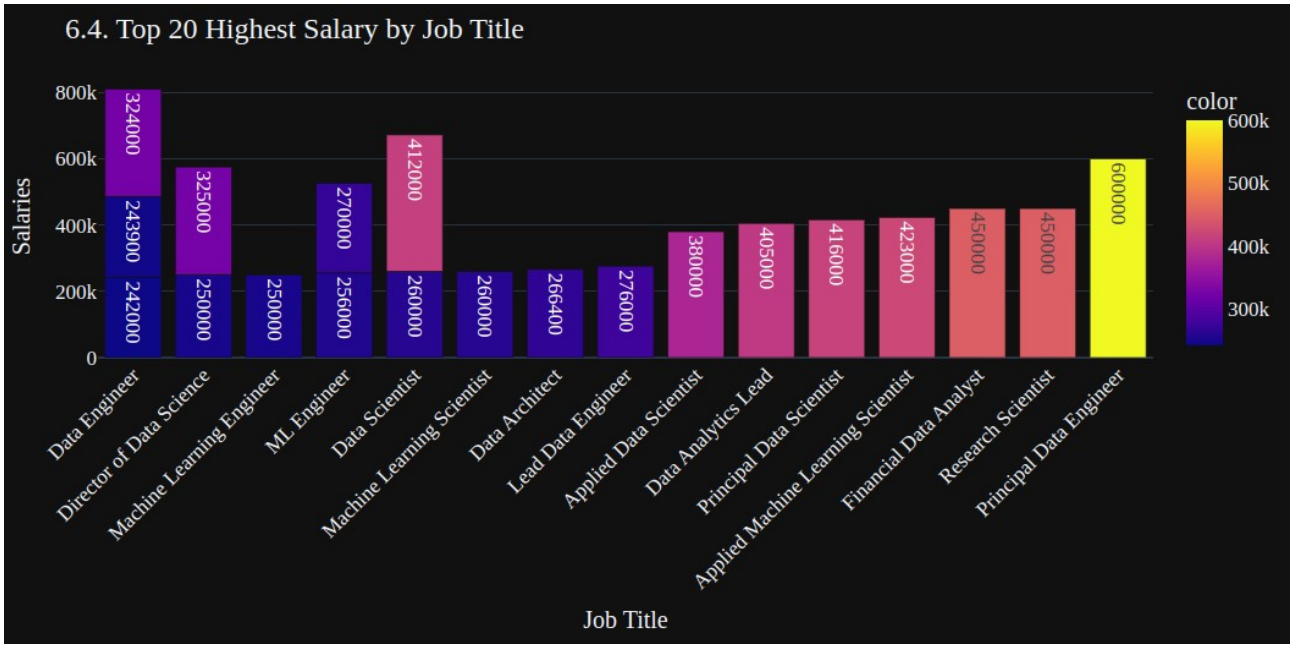

```
c_size = df[['company_size','salary_in_usd']]
small = exlevel_salary.loc[c_size['company_size']=='S']
mid = exlevel_salary.loc[c_size['company_size']=='M']
large = exlevel_salary.loc[c_size['company_size']=='L']
hist_data =
[small['salary_in_usd'],mid['salary_in_usd'],large['salary_in_usd']]
group_labels = ['Company Size: Small','Company Size: Mid','Company Size:
Large']
colors = ['white','red','blue']

lst = [small['salary_in_usd'].mean(),
mid['salary_in_usd'].mean(),
large['salary_in_usd'].mean()]

plt.figure(figsize=(20,5))
fig1 = ff.create_distplot(hist_data, group_labels, show_hist=False,
colors=colors)

fig2 = go.Figure(data=px.bar(x= group_labels,
y=lst,
color = group_labels,
color_discrete_sequence= colors,
title='6.3.(2) Mean Salary by Company Size',
text = np.round([num/1000 for num in lst],2),
template = 'plotly_dark',
height=500))

fig1.layout.template = 'plotly_dark'
```

```
fig1.update_layout(title='6.3.(1) Salary Distribution by Company Size',font
= dict(size=17,family="Franklin Gothic"))
fig2.update_traces(width=0.3)
fig2.update_layout(
xaxis_title="Company Size",
yaxis_title="Mean Salary (k)",
font = dict(size=17,family="Franklin Gothic"))
fig1.show()
```

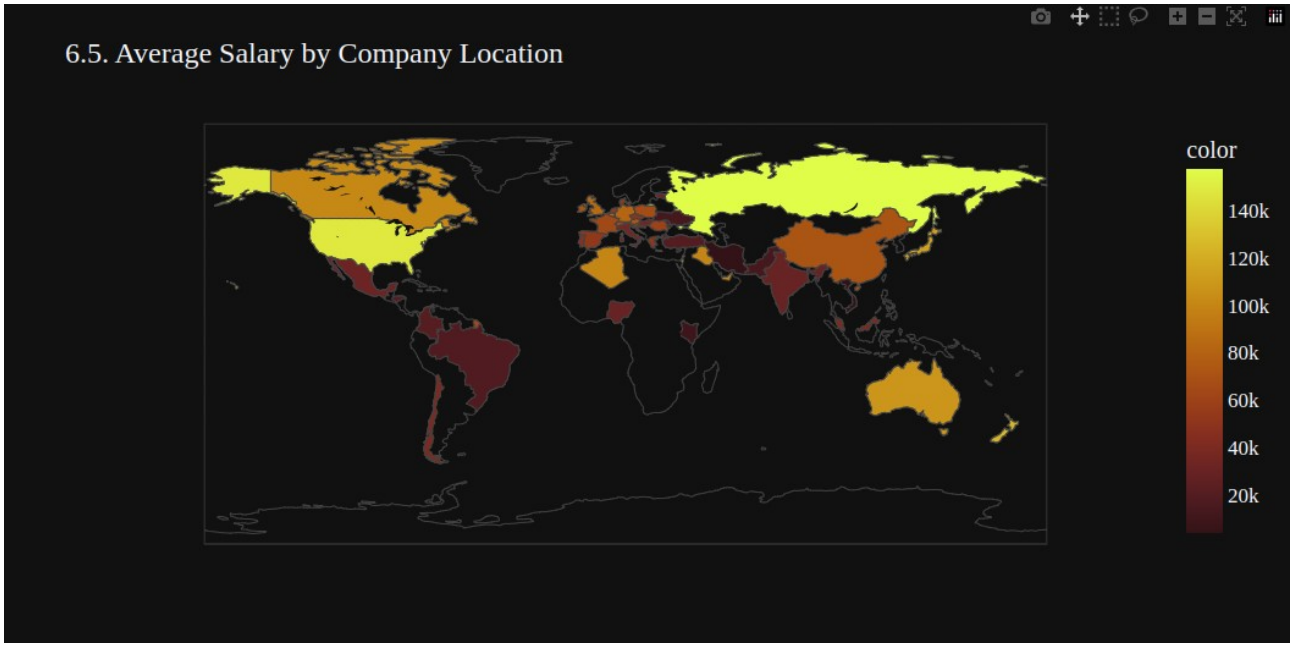


same code as above, but now:

`fig2.show()`

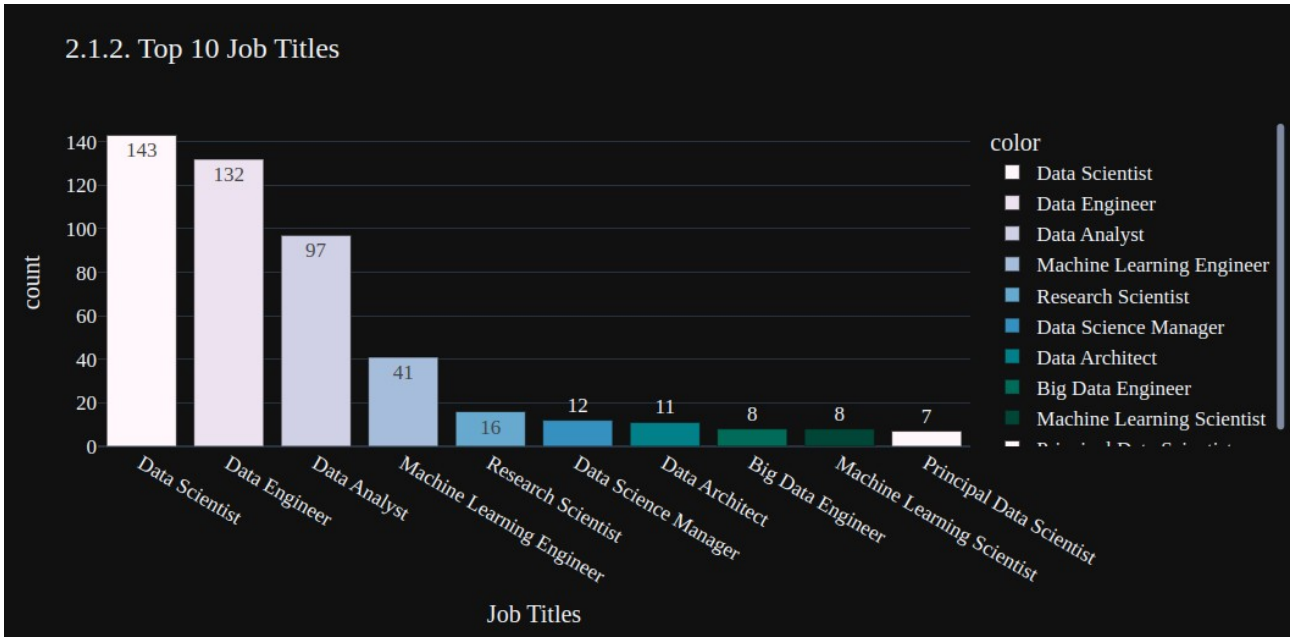



```
alary_job = df.groupby(['salary_in_usd','job_title']).size().reset_index()
salary_job = salary_job[-20:]
fig = px.bar(x=salary_job['job_title'],y=salary_job['salary_in_usd'],text =
salary_job['salary_in_usd'],
color = salary_job['salary_in_usd'],
color_discrete_sequence=px.colors.sequential.PuBu)

fig.update_layout(
xaxis_title="Job Title",
yaxis_title="Salaries ")
# fig.update_traces(width=0.9)
fig.update_layout(barmode = 'relative',xaxis_tickangle=-45,
title='6.4. Top 20 Highest Salary by Job Title', template='plotly_dark',font
= dict(size=17,family="Franklin Gothic"))
```

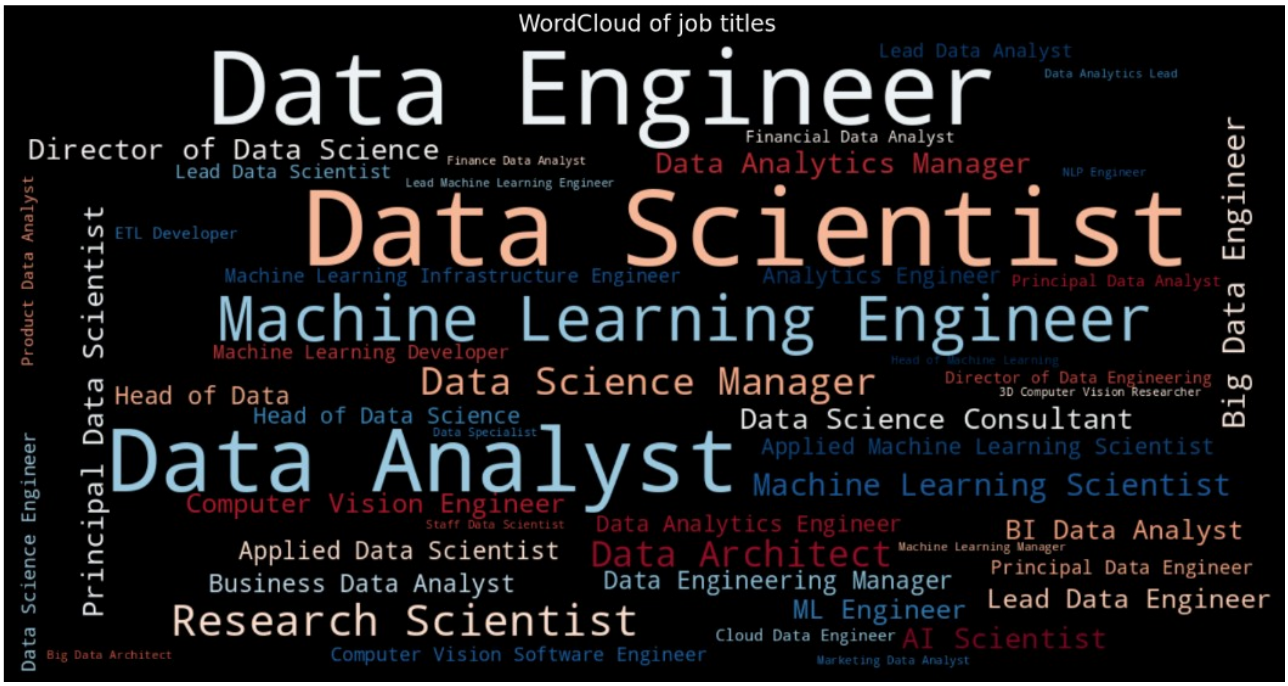



```
salary_location =
df.groupby(['salary_in_usd','company_location']).size().reset_index()
average = salary_location.groupby('company_location').mean().reset_index()

fig = px.choropleth(locations=average['company_location'],
color=average['salary_in_usd'],
color_continuous_scale=px.colors.sequential.solar,
template='plotly_dark',
title = '6.5. Average Salary by Company Location')
fig.update_layout(font = dict(size=17,family="Franklin Gothic"))
fig.show()
```

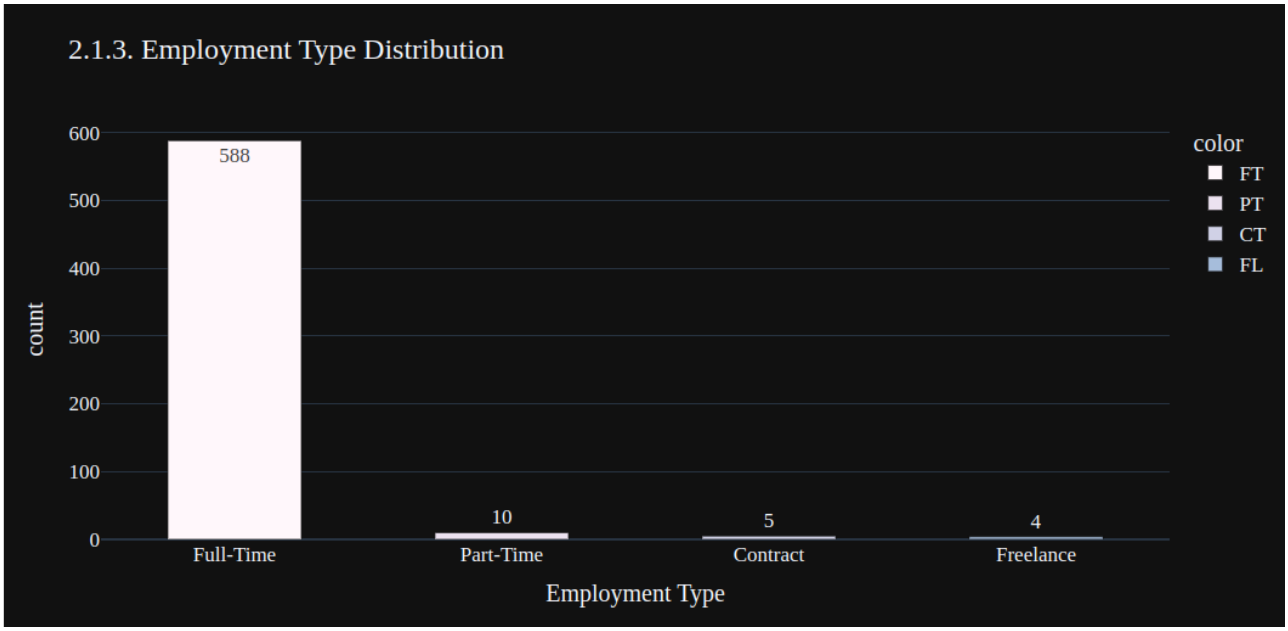

```
df['experience_level'] =
df['experience_level'].replace('EN','Entry-level/Junior')
df['experience_level'] =
df['experience_level'].replace('MI','Mid-level/Intermediate')
df['experience_level'] =
df['experience_level'].replace('SE','Senior-level/Expert')
df['experience_level'] = df['experience_level'].replace('EX','Executive-
level/Director')

ex_level = df['experience_level'].value_counts()
fig = px.treemap(ex_level,
path=[ex_level.index],
values=ex_level.values,
title = '2.1.1. Experience Level',
color=ex_level.index,
color_discrete_sequence=px.colors.sequential.PuBuGn,
template='plotly_dark',
# textinfo = "label+value+percent parent+percent entry+percent root",
width=1000, height=500)

percents = np.round((100*ex_level.values / sum(ex_level.values)).tolist(),2)
fig.data[0].customdata = [35.09, 46.13, 4.28 , 14.5]
fig.data[0].texttemplate = '%{label}<br>%{value}<br>%{customdata}%'

fig.update_layout(
font=dict(size=19,family="Franklin Gothic"))

fig.show()
```



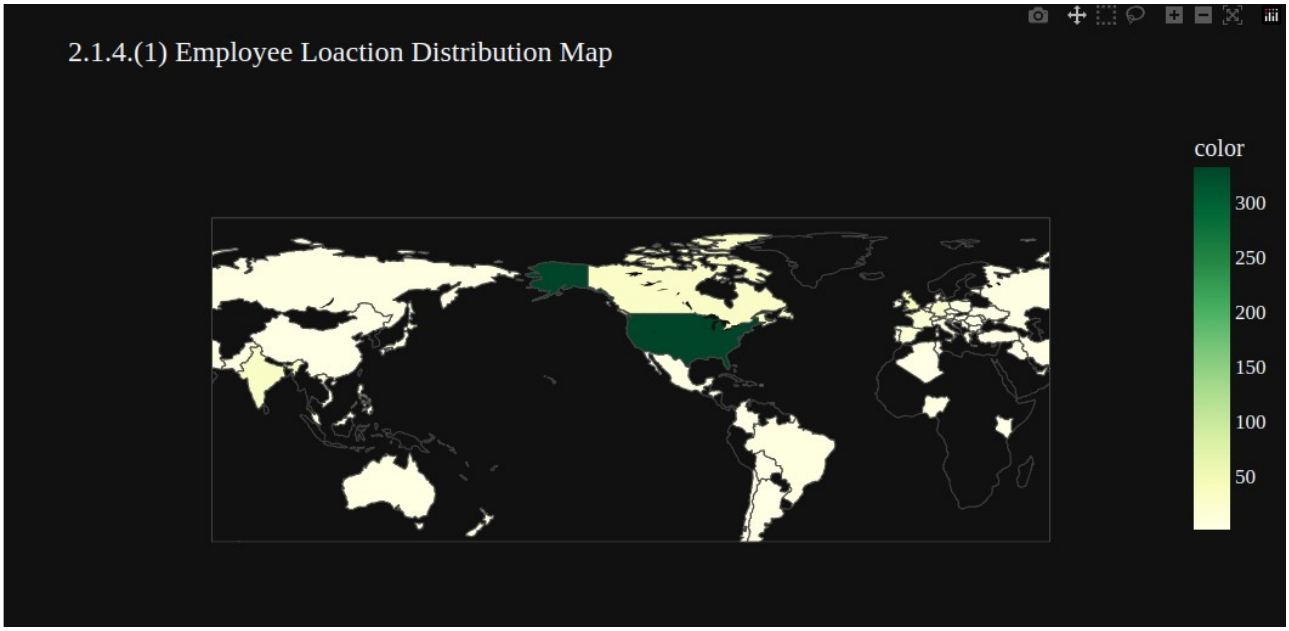

```
top10_job_title = df['job_title'].value_counts()[:10]
fig = px.bar(y=top10_job_title.values,
x=top10_job_title.index,
color = top10_job_title.index,
color_discrete_sequence=px.colors.sequential.PuBuGn,
text=top10_job_title.values,
title= '2.1.2. Top 10 Job Titles',
template= 'plotly_dark')
fig.update_layout(
xaxis_title="Job Titles",
yaxis_title="count",
font = dict(size=17,family="Franklin Gothic"))
fig.show()
```



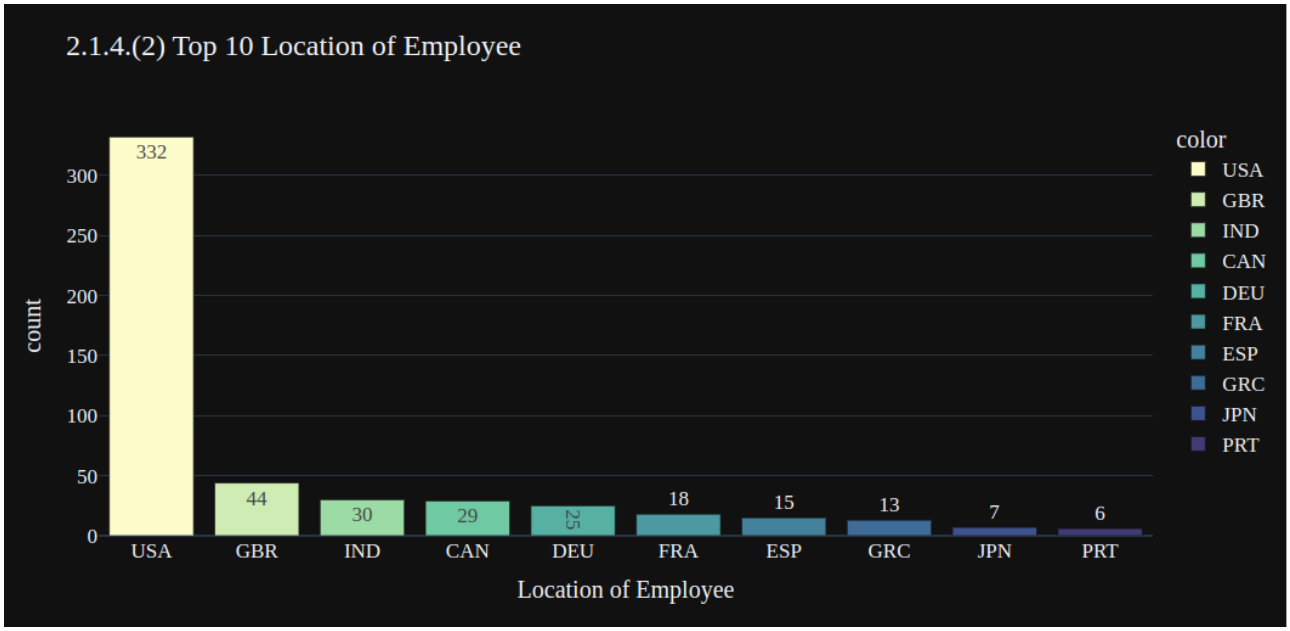

```
def Freq_df(cleanwordlist):
Freq_dist_nltk = nltk.FreqDist(cleanwordlist)
df_freq = pd.DataFrame.from_dict(Freq_dist_nltk, orient='index')
df_freq.columns = ['Frequency']
df_freq.index.name = 'Term'
df_freq = df_freq.sort_values(by=['Frequency'],ascending=False)
df_freq = df_freq.reset_index()
return df_freq

def Word_Cloud(data, color_background, colormap, title):
plt.figure(figsize = (20,15))
wc = WordCloud(width=1200,
height=600,
max_words=50,
colormap= colormap,
max_font_size = 100,
random_state=88,
background_color=color_background).generate_from_frequencies(data)
plt.imshow(wc, interpolation='bilinear')
plt.title(title, fontsize=20)
plt.axis('off')
plt.show()

freq_df = Freq_df(df['job_title'].values.tolist())
data = dict(zip(freq_df['Term'].tolist(), freq_df['Frequency'].tolist()))
data = freq_df.set_index('Term').to_dict()['Frequency']

Word_Cloud(data ,'black','RdBu', 'WordCloud of job titles')
```



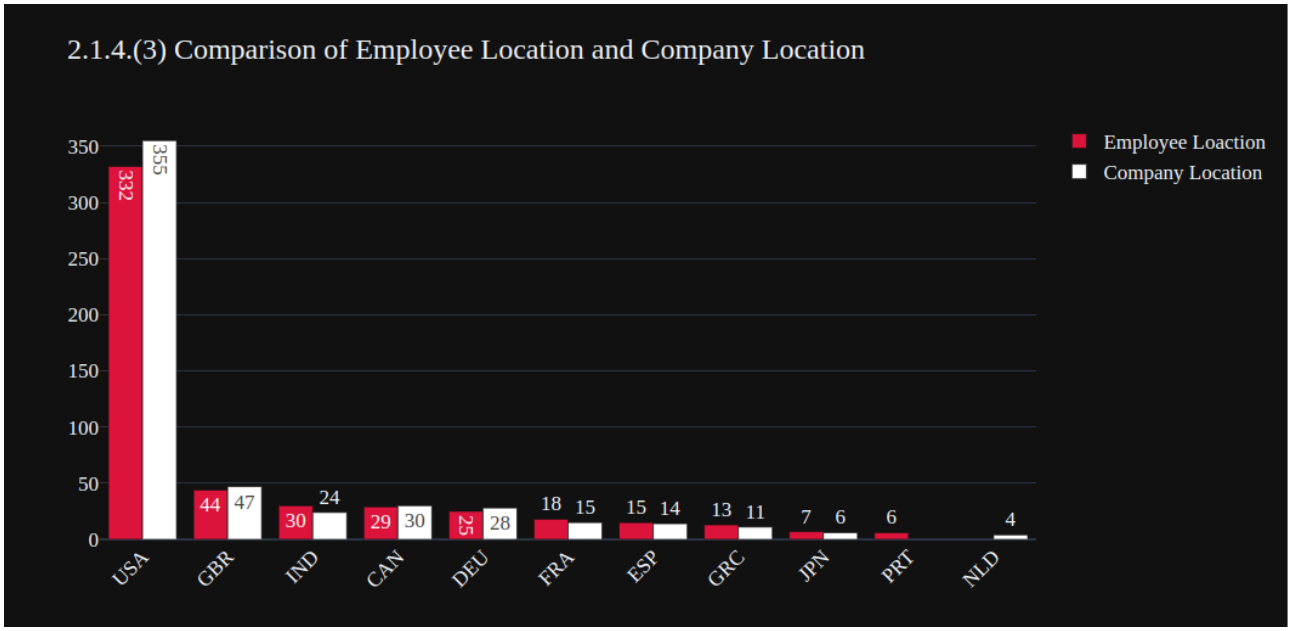

```
type_grouped = df['employment_type'].value_counts()
e_type = ['Full-Time','Part-Time','Contract','Freelance']
fig = px.bar(x = e_type, y = type_grouped.values,
color = type_grouped.index,
color_discrete_sequence=px.colors.sequential.PuBuGn,
template = 'plotly_dark',
text = type_grouped.values, title = '2.1.3. Employment Type Distribution')
fig.update_layout(
xaxis_title="Employment Type",
yaxis_title="count",
font = dict(size=17,family="Franklin Gothic"))
fig.update_traces(width=0.5)
fig.show()
```



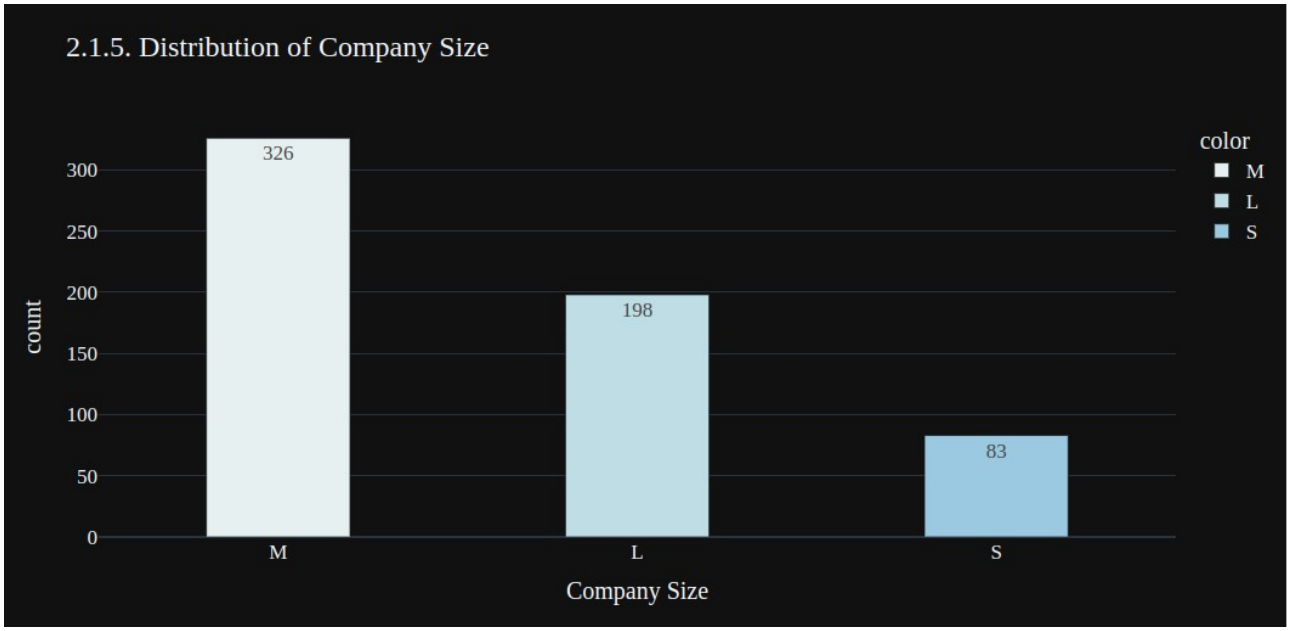

```
converted_country = coco.convert(names=df['employee_residence'], to="ISO3")
df['employee_residence'] = converted_country
residence = df['employee_residence'].value_counts()
fig = px.choropleth(locations=residence.index,
color=residence.values,
color_continuous_scale=px.colors.sequential.YlGn,
template='plotly_dark',
title = '2.1.4.(1) Employee Loaction Distribution Map')

fig.update_layout(font = dict(size= 17, family="Franklin Gothic"))
fig.show()
```



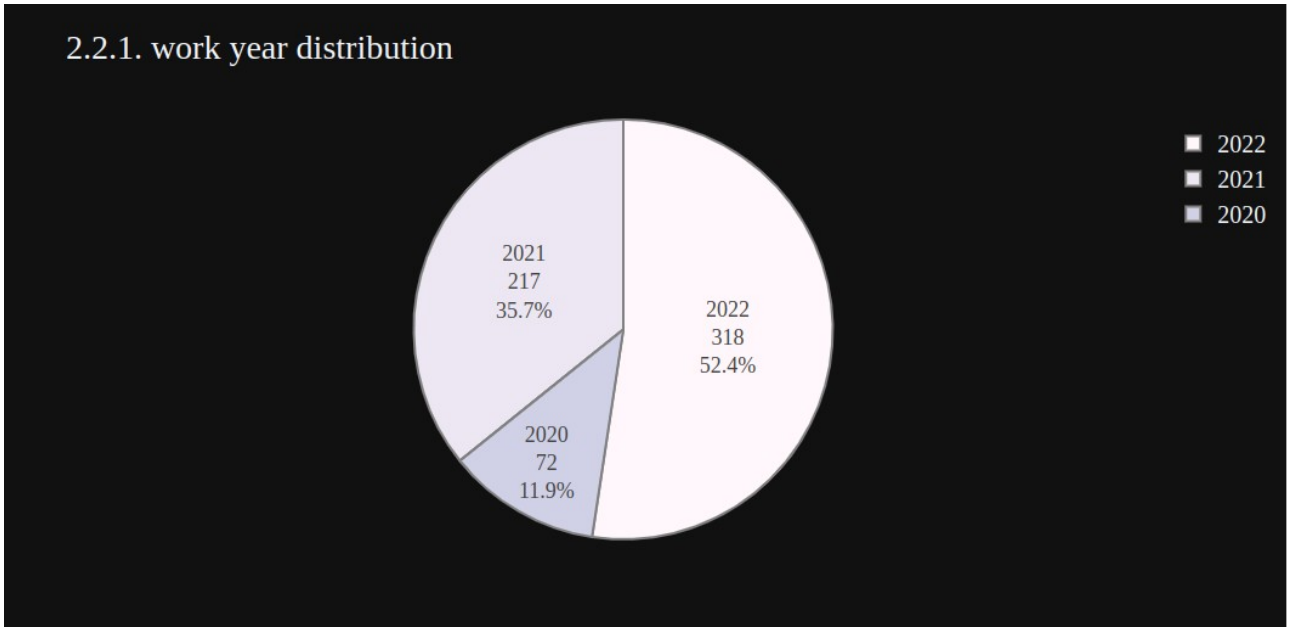

```
top10_employee_location = residence[:10]
fig = px.bar(y=top10_employee_location.values,
x=top10_employee_location.index,
color = top10_employee_location.index,
color_discrete_sequence=px.colors.sequential.deep,
text=top10_employee_location.values,
title= '2.1.4.(2) Top 10 Location of Employee',
template= 'plotly_dark')
fig.update_layout(
xaxis_title="Location of Employee",
yaxis_title="count",
font = dict(size=17,family="Franklin Gothic"))
fig.show()
```



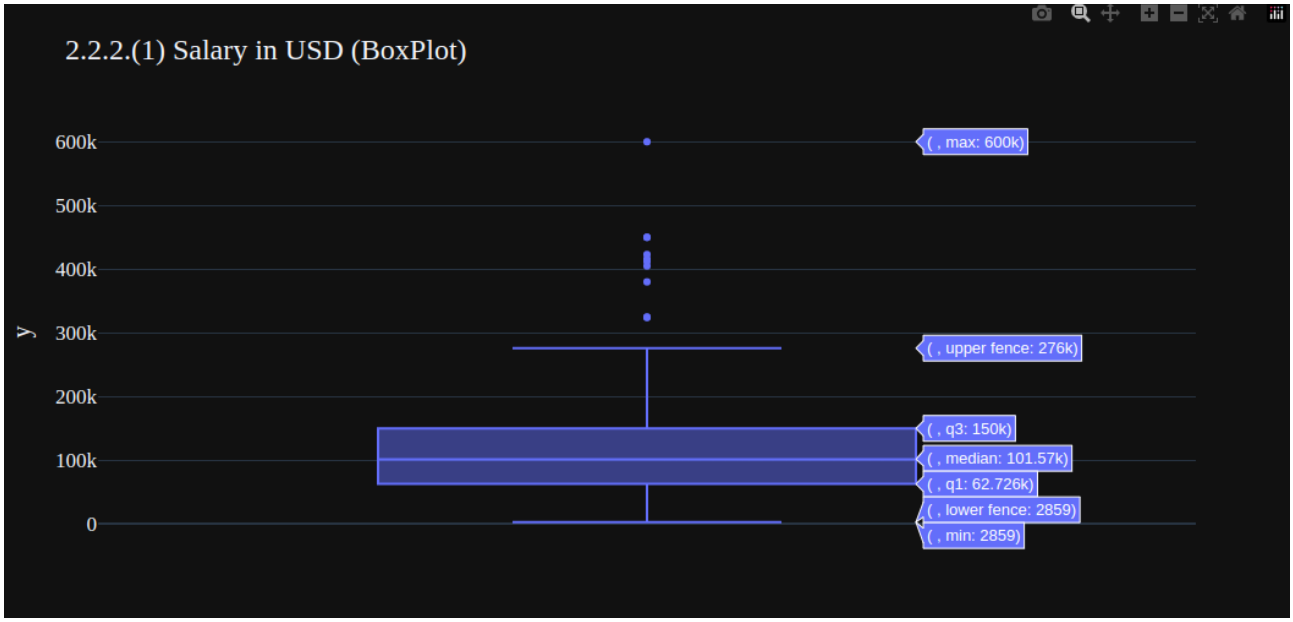

```
converted_country = coco.convert(names=df['company_location'], to="ISO3")
df['company_location'] = converted_country
c_location = df['company_location'].value_counts()
top_10_company_location = c_location[:10]
fig = go.Figure(data=[
go.Bar(name='Employee Loaction',
x=top10_employee_location.index, y=top10_employee_location.values,
text=top10_employee_location.values,marker_color='crimson'),
go.Bar(name='Company Location', x=top_10_company_location.index,
y=top_10_company_location.values,text=top_10_company_location.values,marker_
color='white')
])
fig.update_layout(barmode='group', xaxis_tickangle=-45,
title='2.1.4.(3) Comparison of Employee Location and Company
Location',template='plotly_dark',
font = dict(size=17,family="Franklin Gothic"))

fig.show()
```



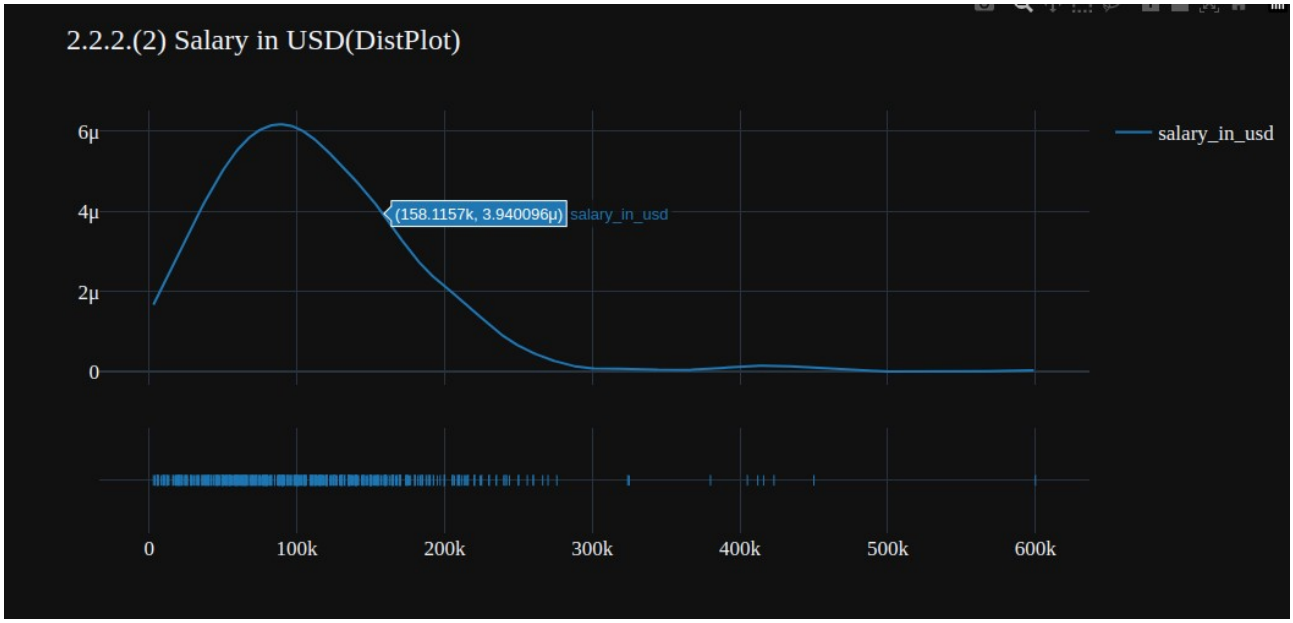

```
grouped_size = df['company_size'].value_counts()

fig = px.bar(y=grouped_size.values,
x=grouped_size.index,
color = grouped_size.index,
color_discrete_sequence=px.colors.sequential.dense,
text=grouped_size.values,
title= '2.1.5. Distribution of Company Size',
template= 'plotly_dark')

fig.update_traces(width=0.4)
fig.update_layout(
xaxis_title="Company Size",
yaxis_title="count",
font = dict(size=17,family="Franklin Gothic"))
fig.show()
```



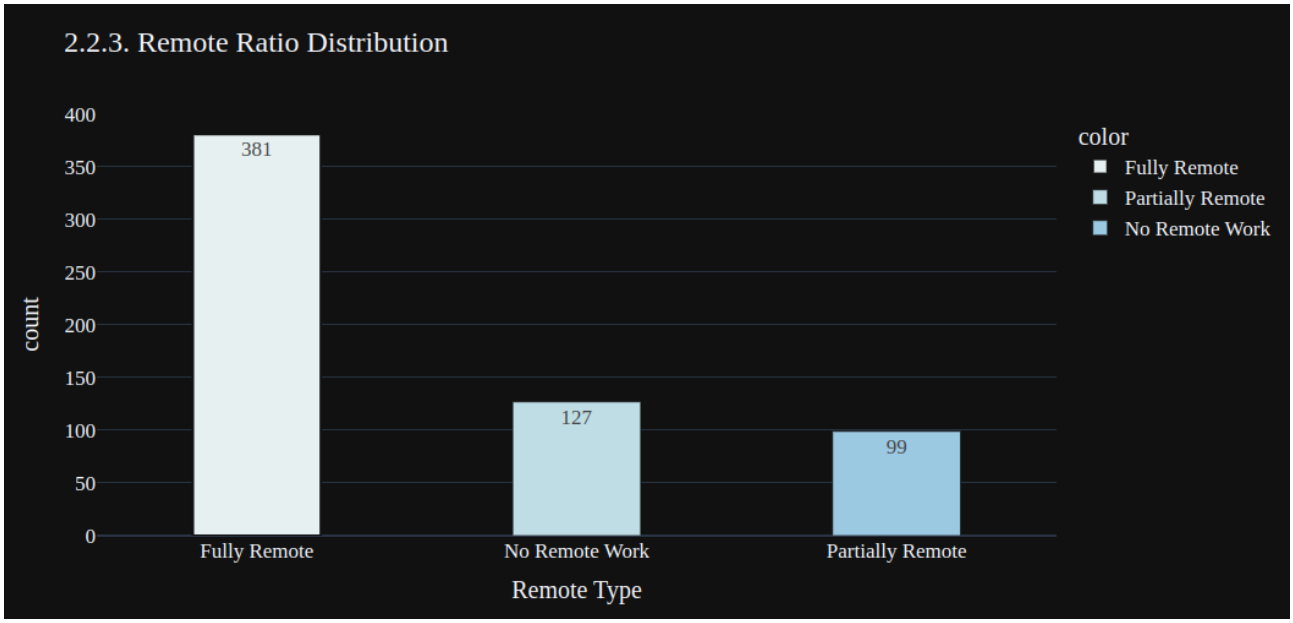

```
wkyear = df['work_year'].value_counts()
fig = px.pie(values=wkyear.values,
names=wkyear.index,
color_discrete_sequence=px.colors.sequential.PuBu,
title= '2.2.1. work year distribution',template='plotly_dark')
fig.update_traces(textinfo='label+percent+value', textfont_size=18,
marker=dict(line=dict(color='#100000', width=0.2)))

fig.data[0].marker.line.width = 2
fig.data[0].marker.line.color='gray'
fig.update_layout(
font=dict(size=20,family="Franklin Gothic"))
fig.show()
```




```
hist_data = [df['salary_in_usd']]
group_labels = ['salary_in_usd']
fig1 = px.box(y=df['salary_in_usd'],template= 'plotly_dark', title = '2.2.2.
(1) Salary in USD (BoxPlot)')
fig2 = ff.create_distplot(hist_data, group_labels, show_hist=False)
fig2.layout.template = 'plotly_dark'
fig1.update_layout(font = dict(size=17,family="Franklin Gothic"))
fig2.update_layout(title='2.2.2.(2) Salary in USD(DistPlot)', font =
dict(size=17, family="Franklin Gothic"))
fig1.show()
```



\\ Same as before, but now:
`fig2.show()`




```
remote_type = ['Fully Remote','Partially Remote','No Remote Work']

plt.figure(figsize=(20,5))
fig = px.bar(x = ['Fully Remote','No Remote Work','Partially Remote'],
y = df['remote_ratio'].value_counts().values,
color = remote_type,
color_discrete_sequence=px.colors.sequential.dense,
text=df['remote_ratio'].value_counts().values,
title = '2.2.3. Remote Ratio Distribution',
template='plotly_dark')

fig.update_traces(width=0.4)

fig.data[0].marker.line.width = 2


fig.update_layout(
xaxis_title="Remote Type",
yaxis_title="count",
font = dict(size=17,family="Franklin Gothic"))
fig.show()
```