- Write a program to display "Welcome " if a number entered by user is a multiple of five otherwise print "Bye“.

Code

```
num=int(input("Enter a number: "))
if num%5==0:
    print("Welcome")
else:
    print("Bye")
```

Output

```
Enter a number: 55
Welcome
```

➢Write a program to find the largest number out of three numbers excepted from user.

Code

```
num=[]
for x in range(1,4):
    a=int(input("Enter number:"))
    num.append(a)
    
for j in range(1,len(num)):
    if num[0]<num[j]:
        b=num[0]
        num[0]=num[j]
        num[j]=b
print("The greatest number is:",num[0])
```

Output

```
Enter number:45
Enter number:65
Enter number:88
The greatest number is: 88
```

- Write a program to accept a number from 1 to 7 and display the name of the day like 1 for Sunday , 2 for Monday and so on.

Code

```
week={1:"Sunday",2:"Monday",3:"Tuesday",4:"Wednesday",5:"Thrusday",6:"Friday",7:"Saturday"}
day=int(input("Enter a number between 1 to 7: "))
if day in range(1,7):
    print("It's",week[day])
else:
    print("Invalid Day!")
```

Output

```
Enter a number between 1 to 7: 6
It's Friday
```

➢Accept any city from the user and display monument of that city

City	Monument
Delhi	Red Fort
Agra	Taj Mahal
Jaipur	Jal Mahal

Code

```
display={"Delhi":"Red Fort","Agra":"Taj Mahal","Jaipur":"Jal Mahal"}
city=input("Enter the name of the city: ")
city=city.capitalize()
if city in display:
    print(display[city])
else:
    print("This city is not in our list")
```

Output

```
Enter the name of the city: aGRA
Taj Mahal
```

- Write a program to accept two numbers and mathematical operators and perform operation accordingly.
  Like: Enter First Number: 7 Enter Second Number : 9 Enter operator : + (you can use different operators as well) Your Answer is : 16

Code

```
num1=int(input("Enter your 1st number: "))
num2=int(input("Enter your 2nd number: "))
op=input("Enter your operator: ")
if op=="+":
    ans=num1+num2
    print("Your Answer is: ",ans)
elif op=="-":
    ans=num1-num2
    print("Your Answer is: ",ans)
elif op=="*":
    ans=num1*num2
    print("Your Answer is: ",ans)
elif op=="/":
    ans=num1/num2
    print("Your Answer is: ",ans)
elif op=="^":
    ans=num1^num2
    print("Your Answer is: ",ans)
else:
    print("Invalid operater")
```

Output

```
Enter your 1st number: 5
Enter your 2nd number: 2
Enter your operator: ^
Your Answer is:  7
```

- Check if the input is Leap Year , write a function We add a Leap Day on February 29, almost every four years. The leap day is an extra, or intercalary day and we add it to the shortest month of the year, February. In the Gregorian calendar three criteria must be taken into account to identify leap years: 1. The year can be evenly divided by 4, is a leap year, unless: 2. The year can be evenly divided by 100, it is NOT a leap year, unless: 3. The year is also evenly divisible by 400. Then it is a leap year. Examples : This means that in the Gregorian calendar, the years 2000 and 2400 are leap years, while 1800, 1900, 2100, 2200, 2300 and 2500 are NOT leap years. What you have to do? You are given the year, and you have to write a function to check if the year is leap or not. Note that you have to complete the function and remaining code is given as template. You can use a variable as input with a fix value of user input

Code

```
year=int(input("Enter your year: "))
if (year%4==0 and year%100!=0) or (year%400==0 and year%100==0):
    print(year,"is a leap year.")
else:
    print(year,"is not a leap year")
```

Ouput

```
Enter your year: 2012
2012 is a leap year.
```

6. Check if the input is Leap Year , write a function We add a Leap Day on February 29, almost every four years. The leap day is an extra, or intercalary day and we add it to the shortest month of the year, February. In the Gregorian calendar three criteria must be taken into account to identify leap years: 1. The year can be evenly divided by 4, is a leap year, unless: 2. The year can be evenly divided by 100, it is NOT a leap year, unless: 3. The year is also evenly divisible by 400. Then it is a leap year. Examples : This means that in the Gregorian calendar, the years 2000 and 2400 are leap years, while 1800, 1900, 2100, 2200, 2300 and 2500 are NOT leap years. What you have to do? You are given the year, and you have to write a function to check if the year is leap or not. Note that you have to complete the function and remaining code is given as template. You can use a variable as input with a fix value of user input

CODE:

```
import random
game=("rock","paper","scissors")
play1=input("Enter p1 name: ").capitalize()
play2 = input("Enter p2 name").capitalize()
while True:
    mov1=input(play1+" Enter your move: ").lower()
    mov2=input(play2+" Enter your move: ").lower()
    if mov1 and mov2 in game:
        break
    else:
        print("Invalid entery please re-enter")
print(play1," plays",mov1)
print(play2," plays",mov2)
if (mov1=="rock" and mov2=="scissors") or (mov1=="scissors" and mov2=="paper") or (mov1=="paper" and mov2=="rock"):
    print(play1,"wins")
elif (mov2=="rock" and mov1=="scissors") or (mov2=="scissors" and mov1=="paper") or (mov2=="paper" and mov1=="rock"):
    print(play2," wins")
else:
    print("it's a draw")
```

OUTPUT:

```
Enter p1 name: Sourabh
Enter p2 names
Sourabh Enter your move: Rock
S Enter your move: Rock
Sourabh  plays rock
S  plays rock
it's a draw
```

PART-2

Suppose you are a data analyst in a company & You have been provided with the HR dataset, now they want you to find certain insights in the organization. Now your task is to do data exploration, data preprocessing & find out the necessary insights they are looking for, the questions are as follows:

I) Import the data & check the head, tail for it.

Code and Output:

```
In [30]: import numpy as np

In [31]: import pandas as pd
         from datetime import datetime, date
         import matplotlib.pyplot as plt
```

Question i.

```
In [32]: df = pd.read_csv(r"C:\Users\ASUS\Desktop\ImsProSchool\DA4\excel_sheets\501+Case1+Dataset.csv")

In [33]: df.head()
```

Out[33]:

	Employee_Name	EmpID	Sex	GenderID	MaritalDesc	MarriedID	MaritalStatusID	DOB	State	Zip	...	ManagerID	RecruitmentSource	LastPerformanc
0	Le, Binh	10232	F	0	Single	0	0	06/14/87	MA	1886	...	13.0	Indeed	
1	Martin, Sandra	10110	F	0	Single	0	0	11/07/1987	MA	2135	...	10.0	Google Search	
2	Myers, Michael	10216	M	1	Single	0	0	04/18/80	MA	1550	...	20.0	LinkedIn	
3	Navathe, Kurt	10079	M	1	Single	0	0	04/25/70	MA	2056	...	13.0	Indeed	
4	Sutwell, Barbara	10209	F	0	Single	0	0	08/15/68	MA	2718	...	16.0	Indeed	

5 rows × 33 columns

```
In [34]: df.tail()
```

Out[34]:

	Employee_Name	EmpID	Sex	GenderID	MaritalDesc	MarriedID	MaritalStatusID	DOB	State	Zip	...	ManagerID	RecruitmentSource	LastPerforma
306	Wilkes, Annie	10204	F	0	Divorced	0	2	07/30/83	MA	1876	...	19.0	Google Search	
307	Demita, Carla	10100	F	0	Separated	0	3	02/25/51	MA	2343	...	18.0	Google Search	
308	Lundy, Susan	10096	F	0	Widowed	0	4	12/26/76	MA	2122	...	22.0	LinkedIn	
309	MacLennan, Samuel	10191	M	1	Widowed	0	4	11/09/1972	MA	1938	...	11.0	Indeed	
310	Thibaud, Kenneth	10268	M	1	Widowed	0	4	09/16/75	MA	2472	...	39.0	Other	

5 rows × 33 columns

II) Check the shape, size of the data.

CODE:

```
df.shape
```

```
(311, 33)
```

```
df.describe()
```

	EmpID	GenderID	MarriedID	MaritalStatusID	Zip	EmpStatusID	DeptID	Salary	PositionID	ManagerID	PerfScoreID	Enga
count	311.000000	311.000000	311.000000	311.000000	311.000000	311.000000	311.000000	311.000000	311.000000	303.000000	311.000000	
mean	10156.000000	0.434084	0.398714	0.810289	6555.482315	1.614148	4.610932	69020.684887	16.845659	14.570957	2.977492	
std	89.922189	0.496435	0.490423	0.943239	16908.396884	0.897483	1.083487	25156.636930	6.223419	8.078306	0.587072	
min	10001.000000	0.000000	0.000000	0.000000	1013.000000	1.000000	1.000000	45046.000000	1.000000	1.000000	1.000000	
25%	10078.500000	0.000000	0.000000	0.000000	1901.500000	1.000000	5.000000	55501.500000	18.000000	10.000000	3.000000	
50%	10156.000000	0.000000	0.000000	1.000000	2132.000000	1.000000	5.000000	62810.000000	19.000000	15.000000	3.000000	
75%	10233.500000	1.000000	1.000000	1.000000	2355.000000	3.000000	5.000000	72036.000000	20.000000	19.000000	3.000000	
max	10311.000000	1.000000	1.000000	4.000000	98052.000000	3.000000	6.000000	250000.000000	30.000000	39.000000	4.000000	

III) How many columns have categorical features?

CODE:

```
lista = list(df.select_dtypes(include=['object']).columns)
print(lista[1:])
```

```
['Sex', 'MaritalDesc', 'DOB', 'State', 'CitizenDesc', 'RaceDesc', 'DateofHire', 'DateofTermination', 'TermReason', 'EmploymentS
tatus', 'Department', 'Position', 'ManagerName', 'RecruitmentSource', 'LastPerformanceReview_Date', 'PerformanceScore']
```

IV) How many unique values are present in RaceDesc column?

CODE:

```
df['RaceDesc'].unique()
```

```
array(['Asian', 'Black or African American', 'Hispanic',
       'Two or more races', 'White', 'American Indian or Alaska Native'],
      dtype=object)
```

V) Check for mean, max .value, min .value, count, standard deviation of ManagerID column.

CODE:

```
df['ManagerID'].describe()
```

```
count    303.000000
mean      14.570957
std        8.078306
min        1.000000
25%       10.000000
50%       15.000000
75%       19.000000
max       39.000000
Name: ManagerID, dtype: float64
```

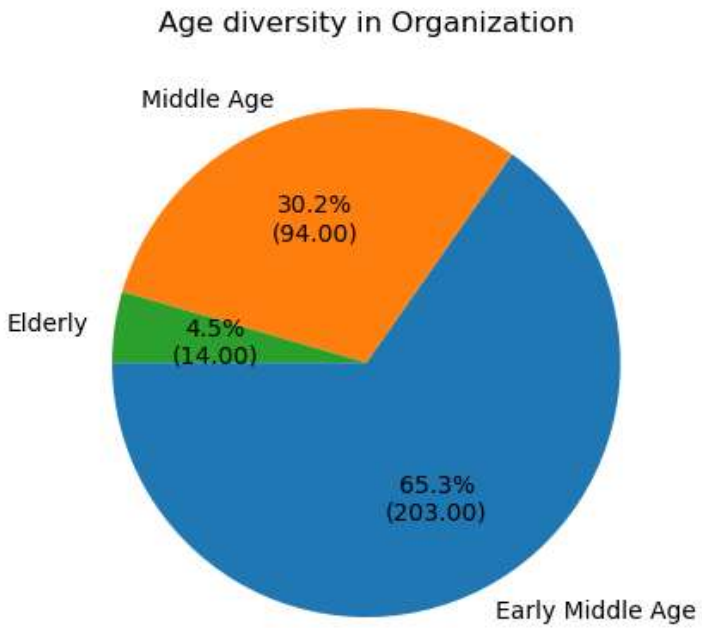
VI) Count the no of categorical , numerical columns .

CODE:

```
df.info(verbose = False)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 311 entries, 0 to 310
Columns: 33 entries, Employee_Name to Absences
dtypes: float64(2), int64(14), object(17)
memory usage: 80.3+ KB
```

VII) Make a diversity report about the dataset.




```
#Data cleaning of DOB
for dates in df['DOB']:
    if len(dates)<10:
        month =  dates[0:2]+'/'
        day = dates[3:5]+'/'
        year = '19'+dates[6:8]
        new_date = day+month+year
        df['DOB'] = df['DOB'].replace(dates,new_date)
```

```
df['DOB'] = pd.to_datetime(df['DOB'], format="%d/%m/%Y")
```

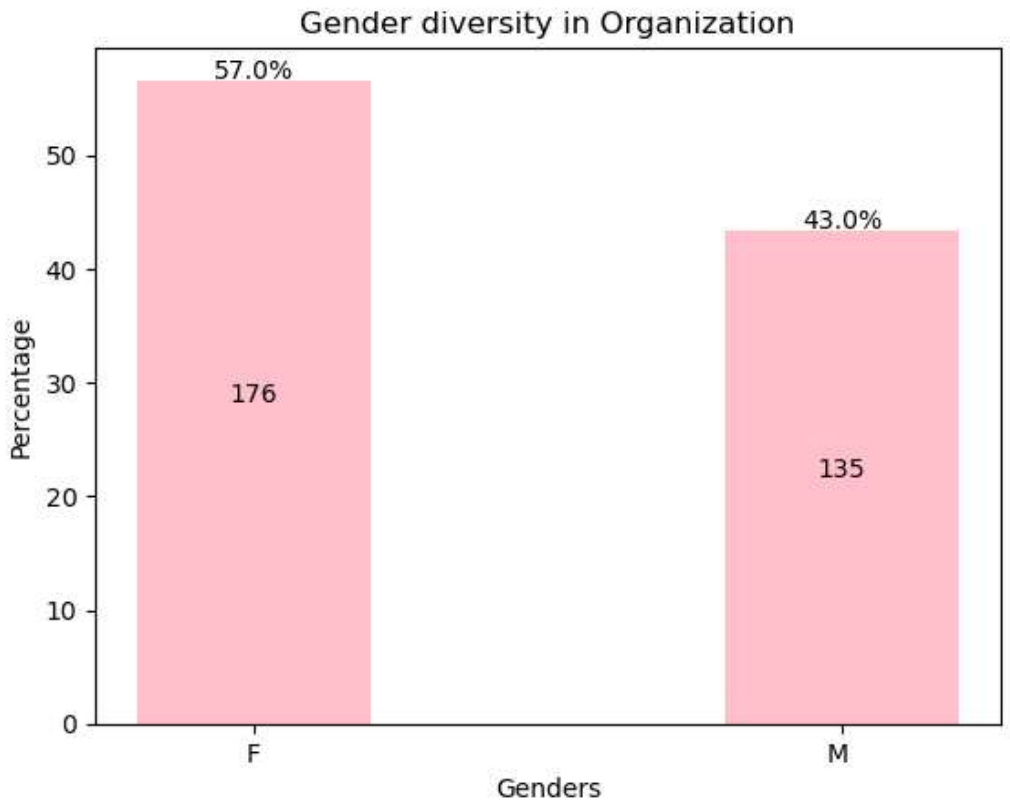
```
#Calculating age and inserting age column
ages = []
today = date.today()
for dates in df['DOB']:
    birthDate = dates
    age = today.year-birthDate.year-((today.month, today.day)<(birthDate.month, birthDate.day))
    ages.append(age)
df.insert(loc = 8,column = "Age",value = ages)
```

```
age_index=[]
for i in df['Age']:
    if 30<=i<=45:
        age_index.append(1)
    elif 45<i<=60:
        age_index.append(2)
    elif 60<i:
        age_index.append(3)
df.insert(loc=8,value=age_index,column='Age_Index')
```

```
count_unique = df['Age_Index'].value_counts().unique()
# Creating autocpt arguments
def func(pct, count_unique):
    absolute = float(pct / 100*np.sum(count_unique))
    return "{:.1f}%\n({:.2f})".format(pct, absolute)
```

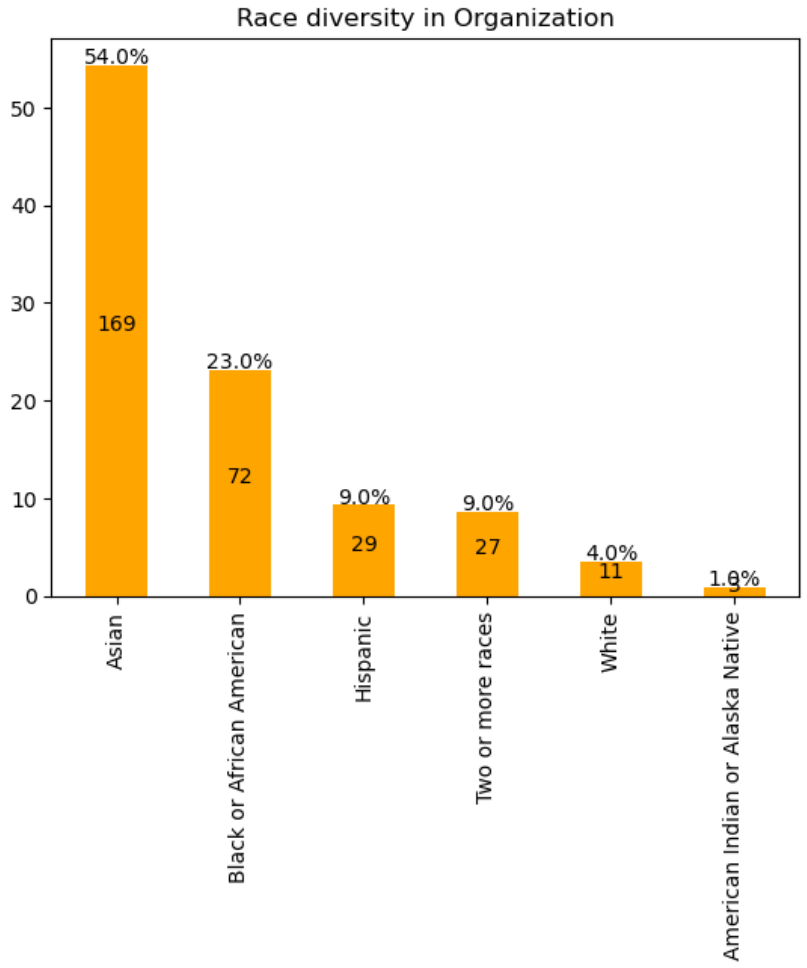
```
y = np.array(df['Age_Index'].value_counts().unique())
mylabels = ["Early Middle Age", "Middle Age", "Elderly"]

plt.pie(y, labels = mylabels,radius = 1.0, autopct = lambda pct: func(pct, y),startangle = 180)
plt.title("Age diversity in Organization")
plt.show()
```

VII) Make a diversity report about the dataset.



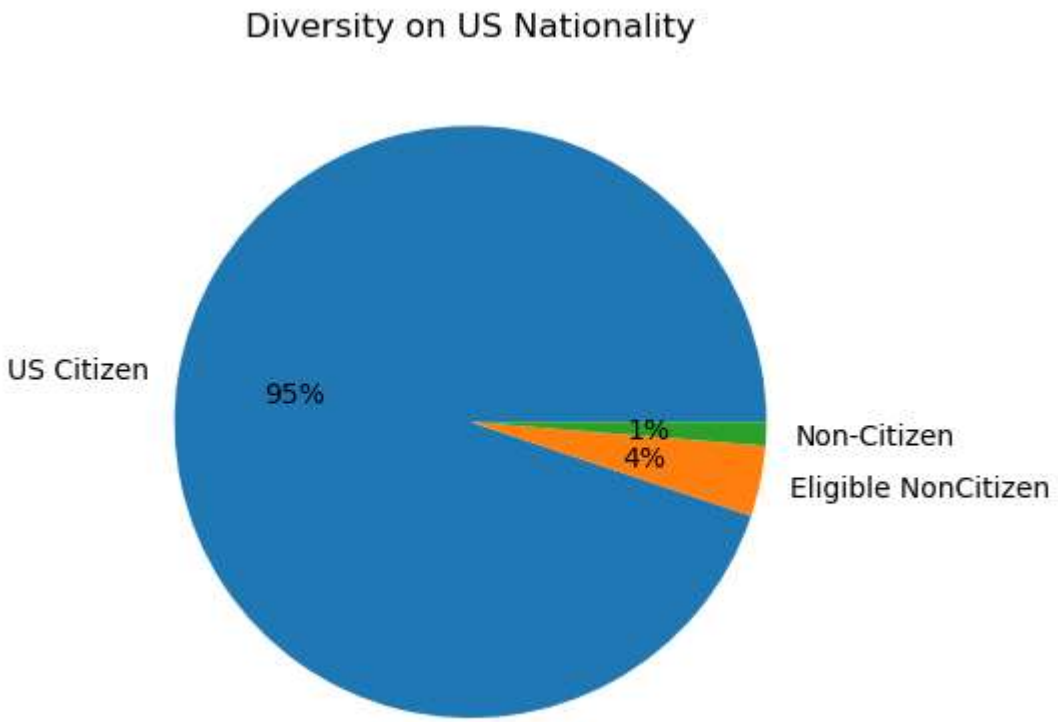

```
gender_label = df['Sex'].unique()
gender_count = np.array(df['Sex'].value_counts().unique())
suma = sum(gender_count)
gender_perct = []
for i in gender_count:
    perct = (i/suma)*100
    gender_perct.append(perct)


fig, ax = plt.subplots()

graph = plt.bar(gender_label,gender_perct,width = 0.4, color = 'pink')
plt.xlabel("Genders")
plt.ylabel("Percentage")
plt.title("Gender diversity in Organization")
for i,p in enumerate(graph):
    height = p.get_height() #height is the value of the bar
    ax.text(x=p.get_x() + p.get_width() / 2, y=height/2, #x and y are the positions where we can add the text, s is the string va
      s="{}".format(gender_count[i]),
           ha = 'center')
    ax.text(x=p.get_x() + p.get_width() / 2, y=height+0.10, #x and y are the positions where we can add the text, s is the string
      s="{}%".format(round(height, 0)),
           ha = 'center')


plt.show()
```

VII) Make a diversity report about the dataset.




```
race_label = df['RaceDesc'].unique()
race_count = np.array(df['RaceDesc'].value_counts().unique())
ax = plt.subplot()
plt.title("Race diversity in Organization")
race_perct = []
for i in race_count:
    perct = (i/sum(race_count))*100
    race_perct.append(perct)
graph = plt.bar(race_label,race_perct,width = 0.5,color = 'orange')
plt.xticks(rotation=90, horizontalalignment="center")

for i,p in enumerate(graph):
    height = p.get_height() #height is the value of the bar
    ax.text(x=p.get_x() + p.get_width() / 2, y=height/2, #x and y are the positions w
      s="{}".format(race_count[i]),
           ha = 'center')
    ax.text(x=p.get_x() + p.get_width() / 2, y=height+0.10, #x and y are the position
      s="{}%".format(round(height, 0)),
           ha = 'center')

plt.show()
```

VII) Make a diversity report about the dataset.




```
citizen_label=df.CitizenDesc.unique()
citizen_count=df.CitizenDesc.value_counts().unique()
plt.pie(citizen_count,labels=citizen_label,radius=1.0,autopct='%1.0f%%')
plt.title("Diversity on US Nationality")
plt.show()
```

VIII) Which columns have correlation with each other, what will you interpret from it.

OUTPUT:

```
df.corr()
```

	EmpID	GenderID	MarriedID	MaritalStatusID	Age	Age_Index	Zip	EmpStatusID	DeptID	Salary	PositionID	ManagerII
EmpID	1.000000	0.035914	0.048058	-0.043851	-0.068438	-0.037310	0.026858	0.056879	0.107406	-0.115319	-0.036488	0.09023
GenderID	0.035914	1.000000	-0.024199	-0.030236	-0.012011	-0.022167	0.048539	-0.021069	-0.038838	0.056097	-0.081612	-0.04321
MarriedID	0.048058	-0.024199	1.000000	0.164044	-0.020058	0.004090	-0.041147	0.072158	-0.119932	0.026165	-0.027334	-0.09400
MaritalStatusID	-0.043851	-0.030236	0.164044	1.000000	0.059134	0.060441	0.010620	0.111401	0.012768	-0.070291	0.021923	0.02306
Age	-0.068438	-0.012011	-0.020058	0.059134	1.000000	0.899088	-0.011678	0.114485	0.098657	0.094554	0.070724	0.06862
Age_Index	-0.037310	-0.022167	0.004090	0.060441	0.899088	1.000000	0.034648	0.131959	0.111340	0.063384	0.020615	0.11030
Zip	0.026858	0.048539	-0.041147	0.010620	-0.011678	0.034648	1.000000	-0.139810	0.290023	-0.037242	-0.552665	0.11922
EmpStatusID	0.056879	-0.021069	0.072158	0.111401	0.114485	0.131959	-0.139810	1.000000	0.080650	-0.113421	0.181625	0.23222
DeptID	0.107406	-0.038838	-0.119932	0.012768	0.098657	0.111340	0.290023	0.080650	1.000000	-0.448132	0.030294	0.55024
Salary	-0.115319	0.056097	0.026165	-0.070291	0.094554	0.063384	-0.037242	-0.113421	-0.448132	1.000000	-0.130563	-0.43540
PositionID	-0.036488	-0.081612	-0.027334	0.021923	0.070724	0.020615	-0.552665	0.181625	0.030294	-0.130563	1.000000	0.09800
ManagerID	0.090236	-0.043218	-0.094002	0.023065	0.068624	0.110304	0.119223	0.232223	0.550240	-0.435406	0.098005	1.00000
PerfScoreID	-0.691348	-0.054915	-0.058362	0.044693	0.079168	0.055004	-0.058350	-0.059393	-0.084811	0.130903	0.005227	-0.06055
EngagementSurvey	-0.589664	-0.036276	-0.091178	0.033249	0.063060	0.042401	-0.132848	0.028484	-0.094940	0.064966	0.074974	-0.00334
EmpSatisfaction	-0.146967	-0.044603	-0.126191	0.002068	-0.065033	-0.096805	-0.064571	-0.004423	0.031997	0.062718	-0.010402	0.10962
SpecialProjectsCount	-0.043730	0.087073	0.061278	-0.051093	-0.089476	-0.092510	-0.097027	-0.170980	-0.785101	0.508333	-0.154326	-0.52508
DaysLateLast30	0.495513	0.080329	0.002875	-0.096500	-0.052834	-0.033013	0.016150	0.060459	0.124630	-0.069443	-0.004040	0.05988
Absences	-0.025278	-0.004577	0.096086	0.018722	-0.040168	-0.020192	0.078779	0.092460	0.053308	0.082382	-0.071434	0.12718

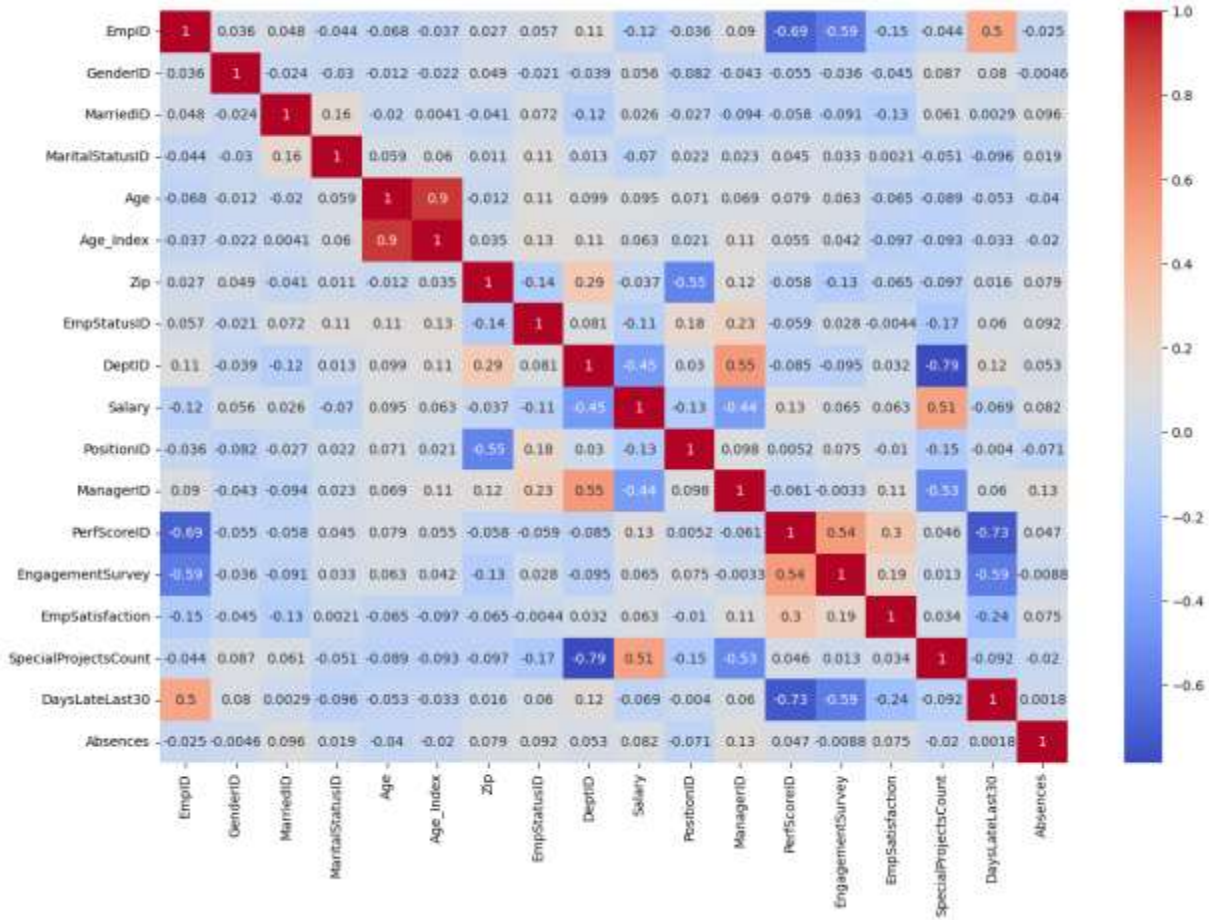
VIII) Which columns have correlation with each other, what will you interpret from it.

CODE:

```
import seaborn as sns
```

```
plt.figure(figsize=(15,10))
sns.heatmap(df.corr(),annot=True,cmap="coolwarm")
```

OUTPUT:



IX) Add a new column which specifies the number of characters in Employee_Name column.

OUTPUT:

```
df['Char_in_EName']=df['Employee_Name'].str.len()
df['Char_in_EName']
```

```
0       8
1      14
2      14
3      13
4      16
       ..
306    13
307    13
308    12
309    17
310    16
Name: Char_in_EName, Length: 311, dtype: int64
```

X) Round up the values in Engagement Survey column

OUTPUT:

```
df['EngagementSurvey']=df['EngagementSurvey'].round(0)
df['EngagementSurvey']
```

```
0      4.0
1      4.0
2      4.0
3      5.0
4      3.0
      ...
306    4.0
307    5.0
308    5.0
309    3.0
310    4.0
Name: EngagementSurvey, Length: 311, dtype: float64
```

XI) Check for a. Does marital status have any impact on salary? b. Does Race Desc have any impact on Emp Satisfaction? c. Is there any correlation between Engagement Survey and Emp Satisfaction.

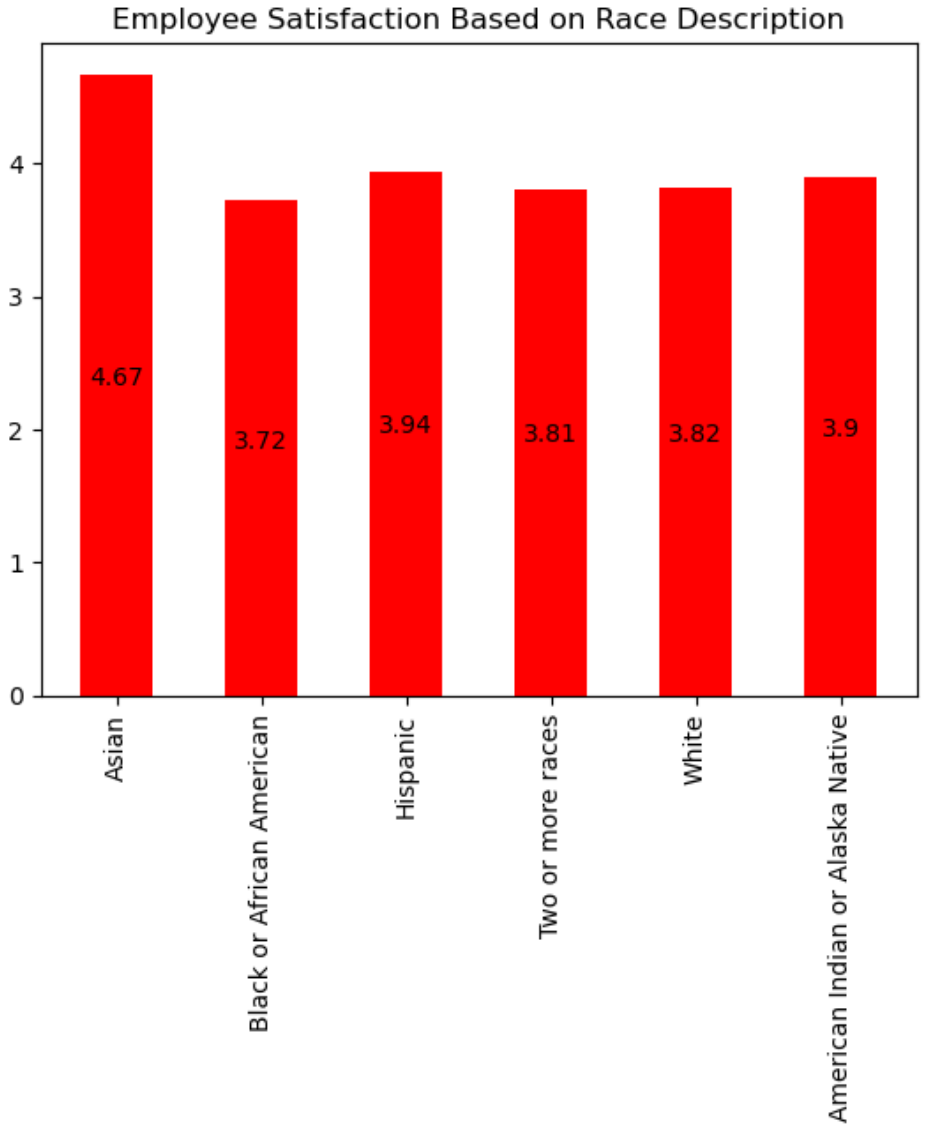
OUTPUT:

```
df['MaritalStatusID'].corr(df['Salary'])
```

```
-0.0702912246475759
```

XI) Check for a. Does marital status have any impact on salary? b. Does Race Desc have any impact on Emp Satisfaction? c. Is there any correlation between Engagement Survey and Emp Satisfaction.

OUTPUT:




```
emp_satisfaction = list(round(df.groupby(['RaceDesc'])['EmpSatisfaction'].mean(),2))
```

```
ax = plt.subplot()
plt.title("Employee Satisfaction Based on Race Description")
race_perct = []
graph = plt.bar(race_label,emp_satisfaction,width = 0.5,color = 'Red')
plt.xticks(rotation=90, horizontalalignment="center")

for i,p in enumerate(graph):
    height = p.get_height() #height is the value of the bar
    ax.text(x=p.get_x() + p.get_width() / 2, y=height/2, #x and y are the positions where we can add the text,
      s="{}".format(emp_satisfaction[i]),
           ha = 'center')


print("From the below chart we can see that RaceDescription has very minute effect on Employee Satisfaction")
```

XI) Check for a. Does marital status have any impact on salary? b. Does Race Desc have any impact on Emp Satisfaction? c. Is there any correlation between Engagement Survey and Emp Satisfaction.

OUTPUT:

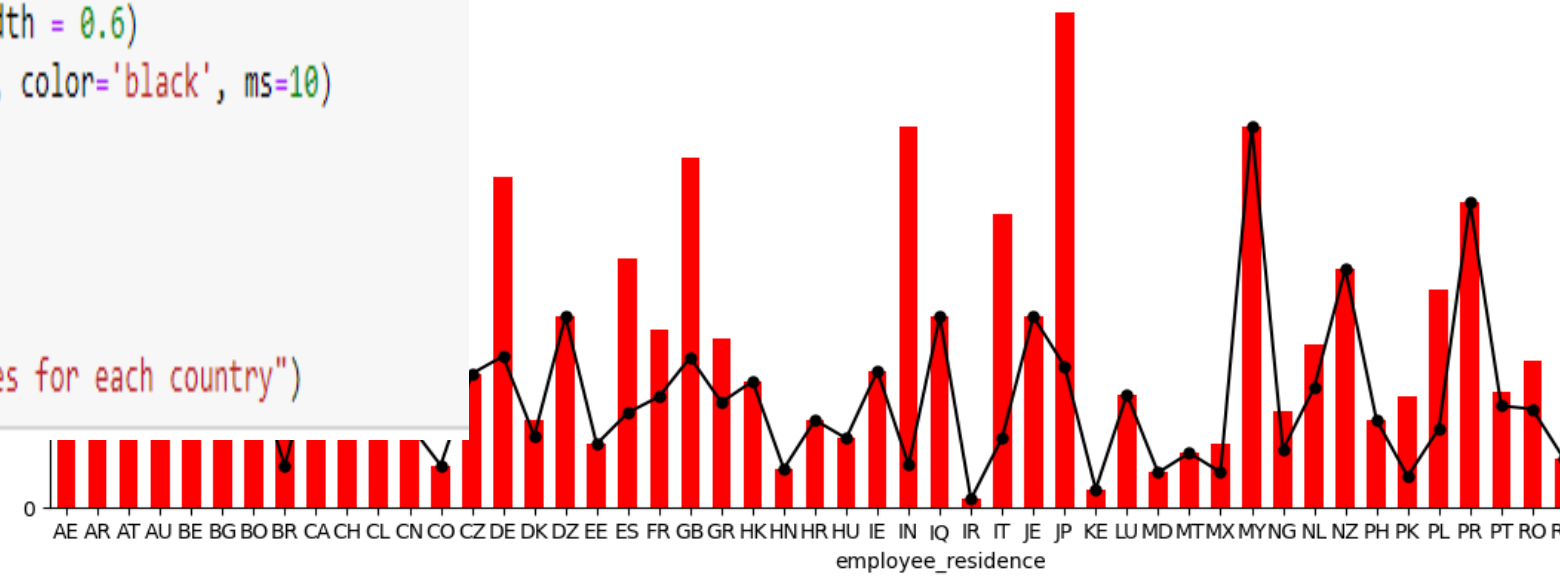
```
relate = df['EngagementSurvey'].corr(df['EmpSatisfaction'])
print("Yes there is a positive correlation between Engagement Survey and EmpSatisfaction: ",round(relate,3))
```

```
Yes there is a positive correlation between Engagement Survey and EmpSatisfaction:  0.186
```

1. H

```
result = df.groupby('employee_residence')['salary_in_usd'].aggregate(['median','max'])
result['Residence'] = df['employee_residence'].sort_values().unique()
```

```
fig, ax = plt.subplots()
ax.set_ylabel('Salary in USD')
plt.rcParams["figure.figsize"] = [16.50, 10.50]
bar_graph = result['max'].plot(kind='bar', color='red', width = 0.6)
line_graph = result['median'].plot(kind='line', marker='.', color='black', ms=10)
plt.legend(['Median Salary','Max Salary'])

plt.show()

print("From the above graph it is visible that salary varies for each country")
```



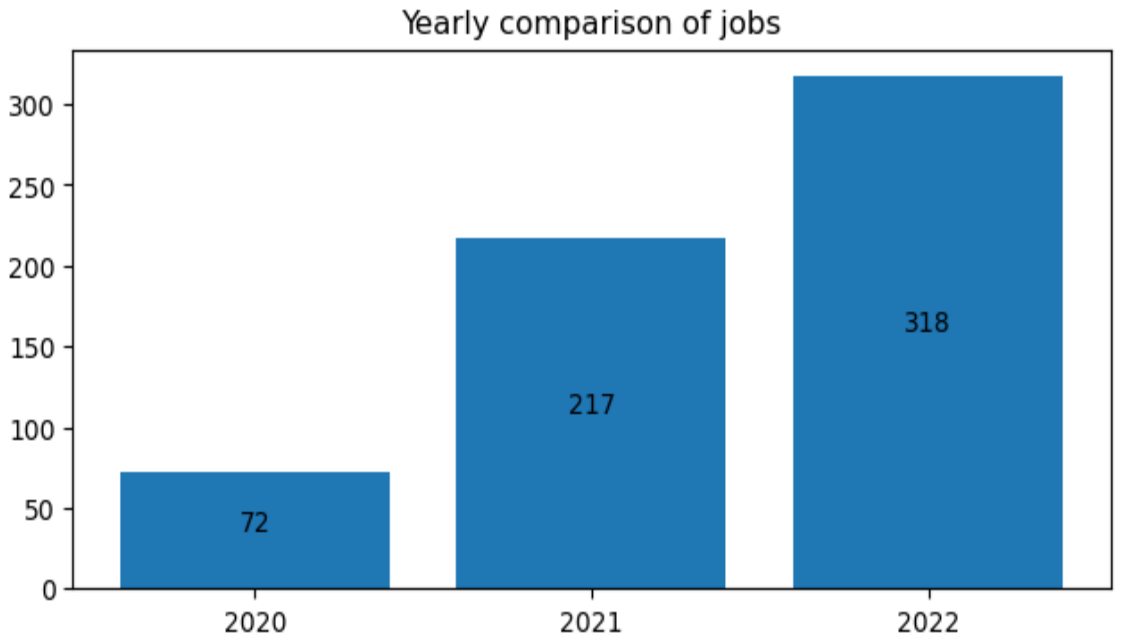
2. How has the demand of the jobs been throughout the years?

CODE:

```
demand_throughout_years = df.groupby(['work_year'])['job_title'].aggregate('count')
list_demand = list(demand_throughout_years)
```

```
ax = plt.subplot()
graph=plt.bar(['2020','2021','2022'],list_demand)
plt.xticks(rotation=0, horizontalalignment="center")
plt.rcParams["figure.figsize"] = [7.50, 3.50]
plt.title("Yearly comparison of jobs")
for g in graph:
    height = g.get_height()
    ax.text(x=g.get_x() + g.get_width() / 2, y=height/2, #x and y are the positions where we can add the
      s="{}".format(round(height, 0)),
          ha = 'center')

plt.show()
```

OUTPUT:



3. How common is to work remote?

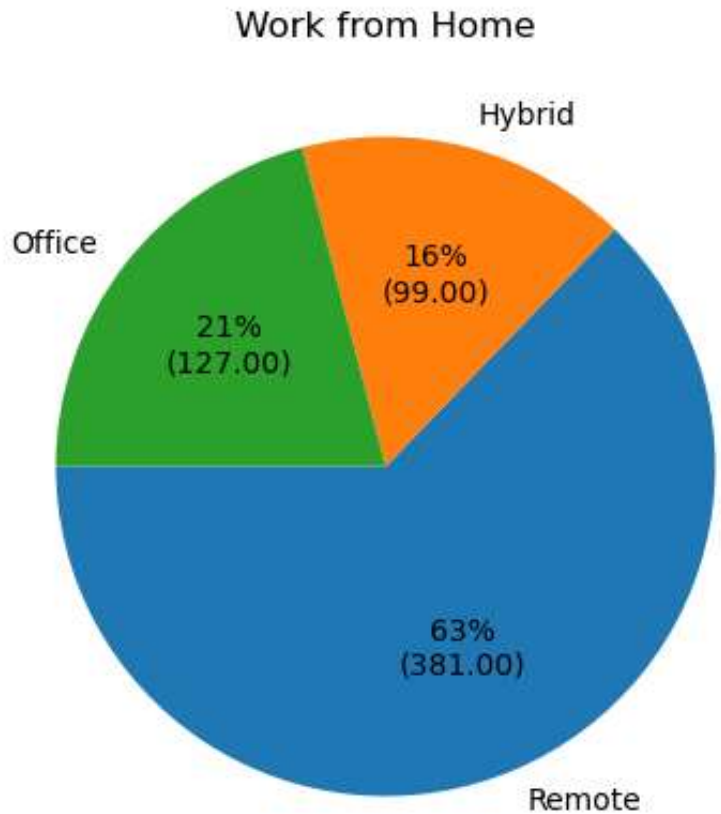
CODE:

```
remote_count = df.where(df.remote_ratio == 100)['remote_ratio'].count()
hybrid_count = df.where(df.remote_ratio == 50)['remote_ratio'].count()
office_count = df.where(df.remote_ratio == 0)['remote_ratio'].count()
total_count = df['remote_ratio'].count()
```

```
def func(pct, count_unique):
    absolute = float(pct / 100*np.sum(count_unique))
    return "{:.0f}%\n({:.2f})".format(pct, absolute)
```

```
plt.figure(figsize=(5,5))
graph = plt.pie(x=[remote_count,hybrid_count,office_count],labels=['Remote','Hybrid','Office'],
                autopct = lambda pct: func(pct,[remote_count,hybrid_count,office_count]),startangle = 180)
plt.title("Work from Home")
plt.show()
common = round((remote_count/total_count)*100,0)
print("On the given observations it can be concluded that out of 100 people {} people choose to work remotely".format(common))
```

OUTPUT:



4. What the highest paying jobs with entry level as well as for senior level experienced?

OUTPUT:

```
max_sal = list(df.groupby('experience_level')['salary_in_usd'].aggregate('max'))
max_sal
entry1 = df.salary_in_usd == max_sal[0]
entry2 = df.experience_level == 'EN'
highest_for_entry = list(df.where(entry2 & entry1)['job_title'].dropna())
cond1 = df.experience_level == 'SE'
cond2 = df.salary_in_usd == max_sal[3]
highest_for_senior = list(df.where(cond1 & cond2)['job_title'].dropna())
```

```
print("The highest paying salary for Entry level is ${} having job title {}".format(max_sal[0],highest_for_entry[0]))
print("The highest paying salary for Senior level Experienced is ${} having job title {}".format(max_sal[3],highest_for_senior[0]
```

```
The highest paying salary for Entry level is $250000 having job title Machine Learning Engineer
The highest paying salary for Senior level Experienced is $412000 having job title Data Scientist
```

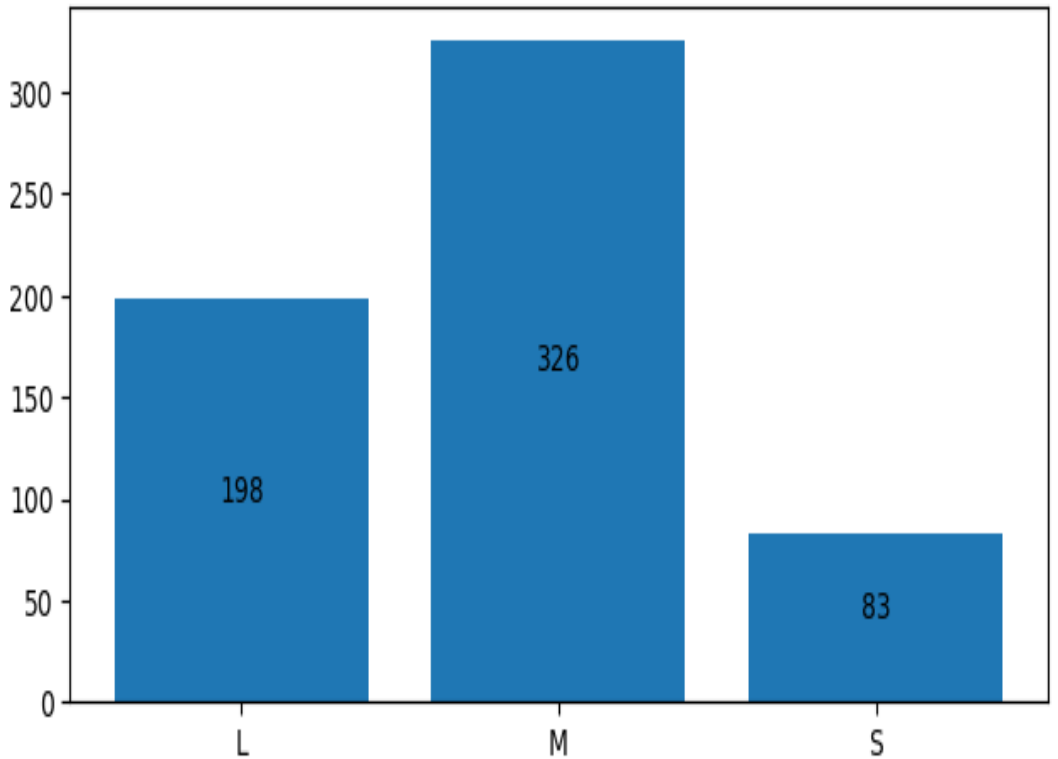
5. What company size hire the most?

Code:

```
most_hiring_by = df.groupby('company_size')['company_size'].count()
```

```
#most_hiring_by.plot(kind = 'bar')

plt.xticks(rotation=0, horizontalalignment="center")
ax = plt.subplot()
graph = plt.bar(most_hiring_by.index,list(most_hiring_by))
plt.rcParams["figure.figsize"] = [7.50, 3.50]

for g in graph:
    height = g.get_height()
    ax.text(x=g.get_x() + g.get_width() / 2, y=height/2, #x and y are the positions where we can add the text, s is the string va
      s="{}".format(round(height,2)),
          ha = 'center')
plt.show()


print("From the below data we can see that most of the hiring is done by M size companies which count to: ",most_hiring_by[1])
```

OUTPUT:



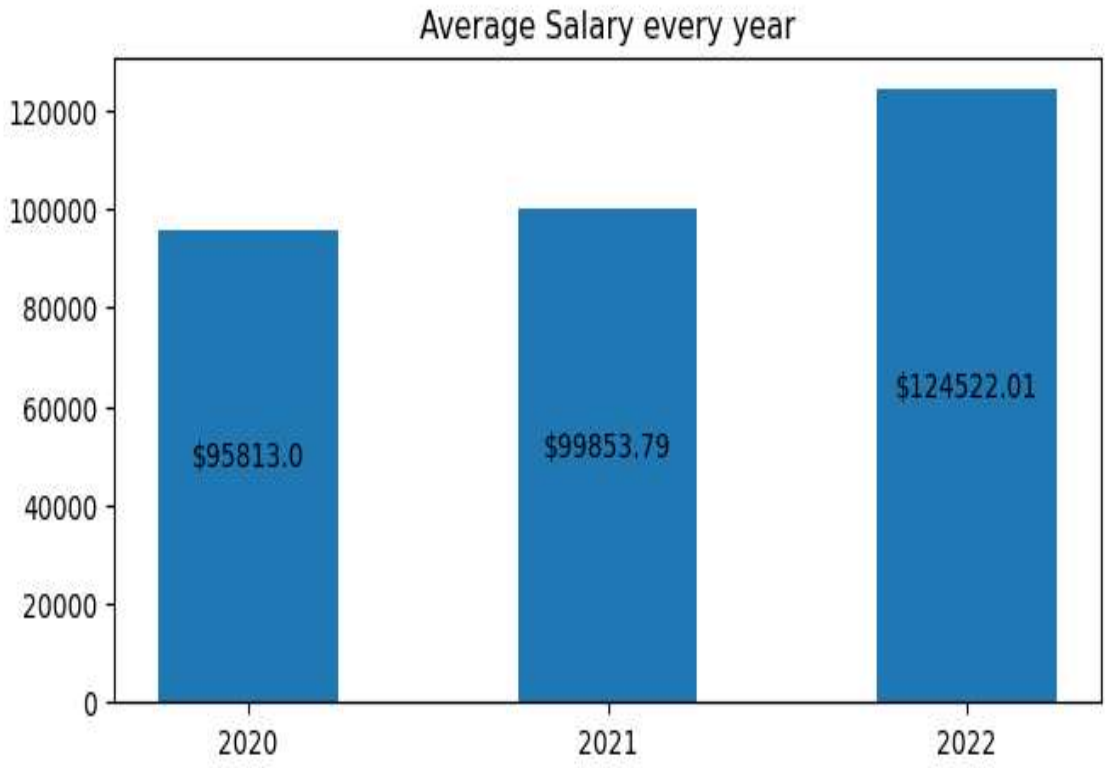
6. How has average salary changed throughout the years?

CODE:

```
avg_sal_years = df.groupby('work_year')['salary_in_usd'].mean()
```

```
list_avg = list(avg_sal_years)
ax = plt.subplot()
plt.rcParams["figure.figsize"] = [7.50, 3.50]
bar = plt.bar(['2020','2021','2022'],list_avg,width = 0.5)
for g in bar:
    height = g.get_height()
    ax.text(x=g.get_x() + g.get_width() / 2, y=height/2, #x and y are the positic
      s="${}".format(round(height,2)),
           ha = 'center')
plt.title("Average Salary every year")
plt.show()
```

OUTPUT:



7. What are most popular roles in Data Science ?

OUTPUT:

```
roles = list(df.groupby('job_title')['job_title'].filter(lambda x: True))
```

```
dict1 = {}
for role in roles:
    if "Data Science" in role or "Data Scientist" in role:
        if role in dict1.keys():
            dict1[role] += 1
        else:
            dict1[role] = 1
keys = list(dict1.keys())
values = list(dict1.values())
sorted_value_index = np.argsort(values)
sorted_dict = {keys[i]: values[i] for i in sorted_value_index}
print("The most popular roles in Data Science is: ",max(zip(dict1.values(), dict1.keys()))[1])
```

```
The most popular roles in Data Science is:  Data Scientist
```

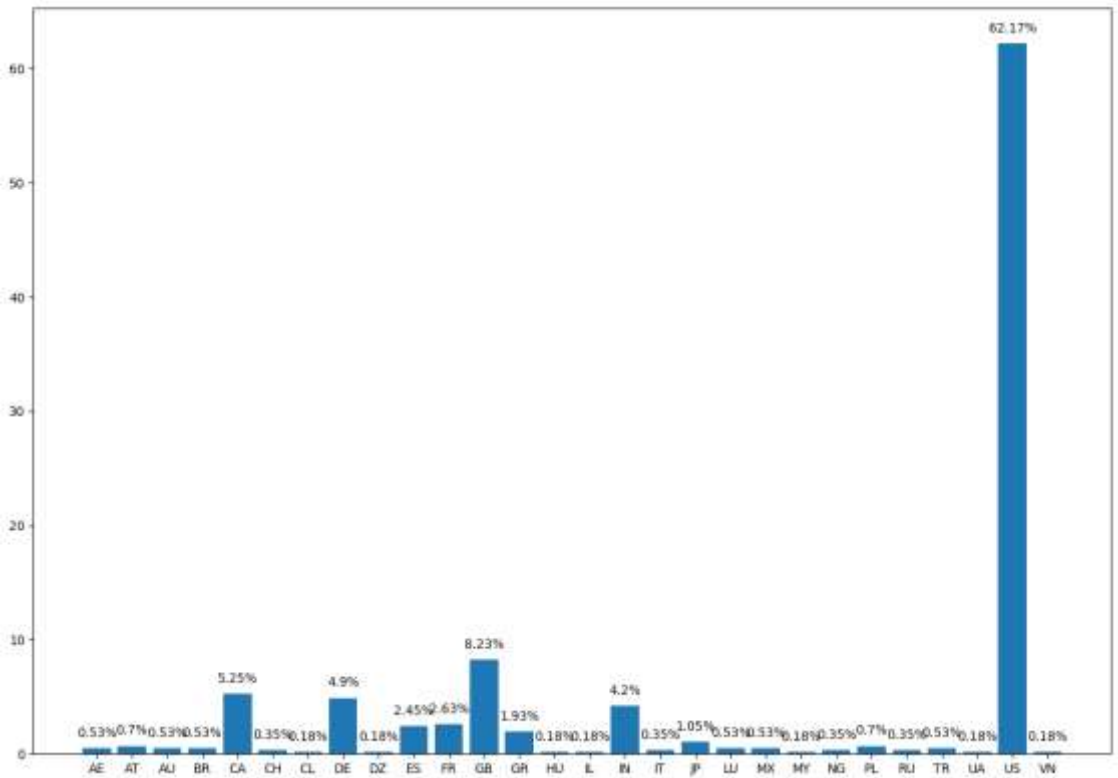
8. Which country hire the most people in Data Science?

CODE:

```
roles = pd.DataFrame(df.groupby(['company_location'])['company_location','job_title'].
                     filter(lambda x: (str(x['job_title']).find("Data Science") != -1
                                       or str(x['job_title']).find("Data Scientist") != -1)))
```

```
most_hiring = roles.groupby('company_location').aggregate('count')

#bar_chart = most_hiring.plot(kind = 'bar')
#bar_chart.set_ylabel("No of Data Science Hirings")

#plt.bar(labela,lista)

labela = most_hiring.index
lista = list(most_hiring['job_title'])
list_perct = []
for i in lista:
    list_perct.append((i/sum(lista))*100)
graph = plt.bar(labela,list_perct)
ax = plt.subplot()
for g in graph:
    height = g.get_height()
    ax.text(x=g.get_x() + g.get_width() / 2, y=height+1.0, #x and y are the positions where we can add the text, s is the string
      s="{}%".format(round(height,2)),
          ha = 'center')

plt.rcParams["figure.figsize"] = [15.50, 10.50]

plt.show()
```

OUTPUT:

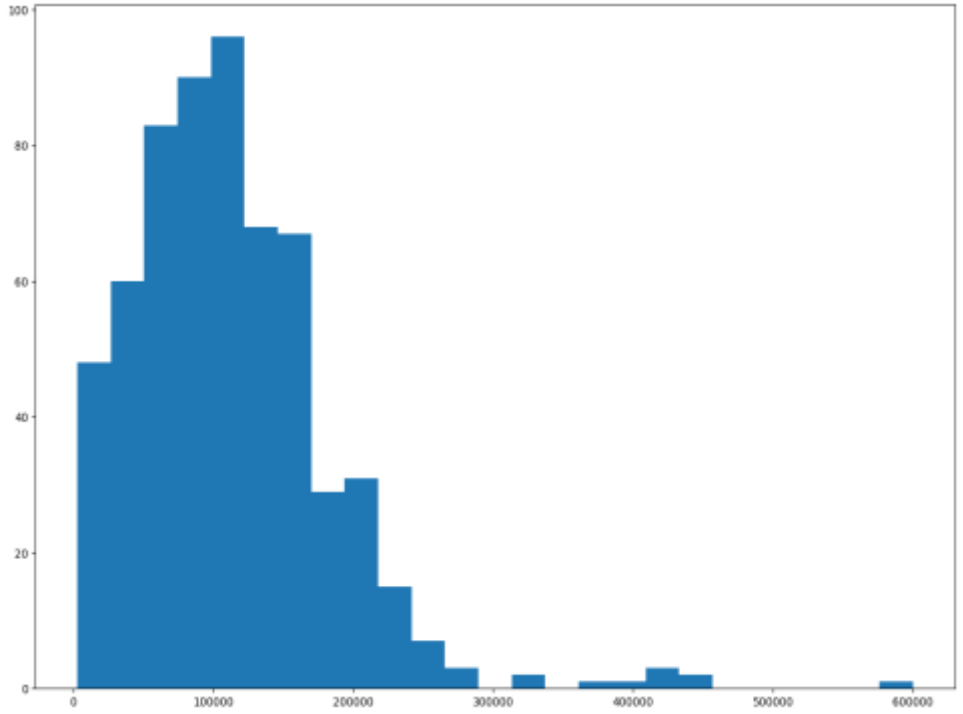


9. What is the distribution of Salaries?

CODE:

```
plt.hist(list(df['salary_in_usd']),bins = int(round(math.sqrt(df['salary_in_usd'].count()),0)))
plt.show()
```

OUTPUT:



10. How much can you expect depending on your years of experience?

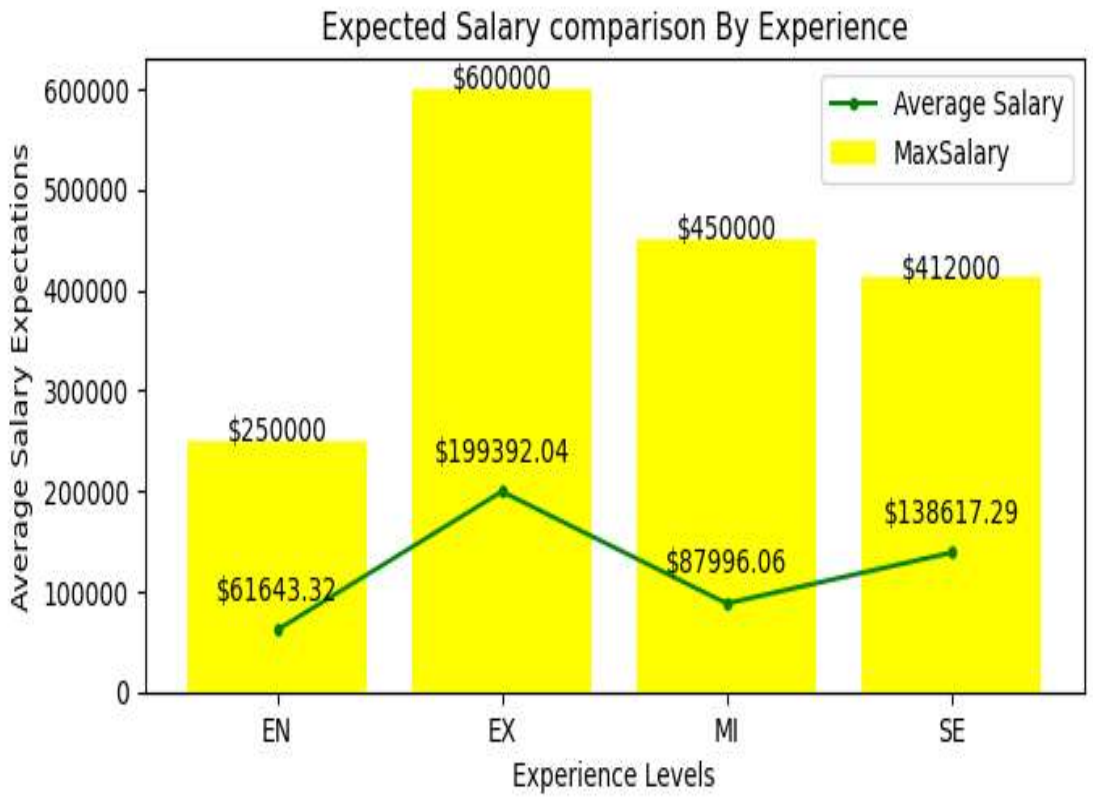
CODE:

```
salary_by_exp = list(df.groupby('experience_level')['salary_in_usd'].aggregate('mean'))
max_salary_by_exp = list(df.groupby('experience_level')['salary_in_usd'].aggregate('max'))
```

```
exp_labels = list(df['experience_level'].sort_values().unique())
```

```
graph = plt.bar(exp_labels,max_salary_by_exp,color = 'Yellow')
ax = plt.subplot()
plot2 = plt.subplot()
plot2.plot(exp_labels,salary_by_exp,marker=".",color='Green')
plt.title("Expected Salary comparison By Experience")
plt.xlabel("Experience Levels")
plt.ylabel("Average Salary Expectations")
plt.legend(['Average Salary','MaxSalary'])
for i,g in enumerate(graph):
    height = salary_by_exp[i]
    ax.text(x=g.get_x() + g.get_width() / 2, y= height+30000.0, #x and y are the positions where we can add the text, s is the st
      s="${}".format(round(salary_by_exp[i],2)),
           ha = 'center')
    ax.text(x=g.get_x() + g.get_width() / 2, y=g.get_height()+0.10, #x and y are the positions where we can add the text, s is th
      s="${}".format(round(g.get_height(),2)),
           ha = 'center')

plt.show()
```

OUTPUT:



11. Which year do people prefer to stay at home the most?

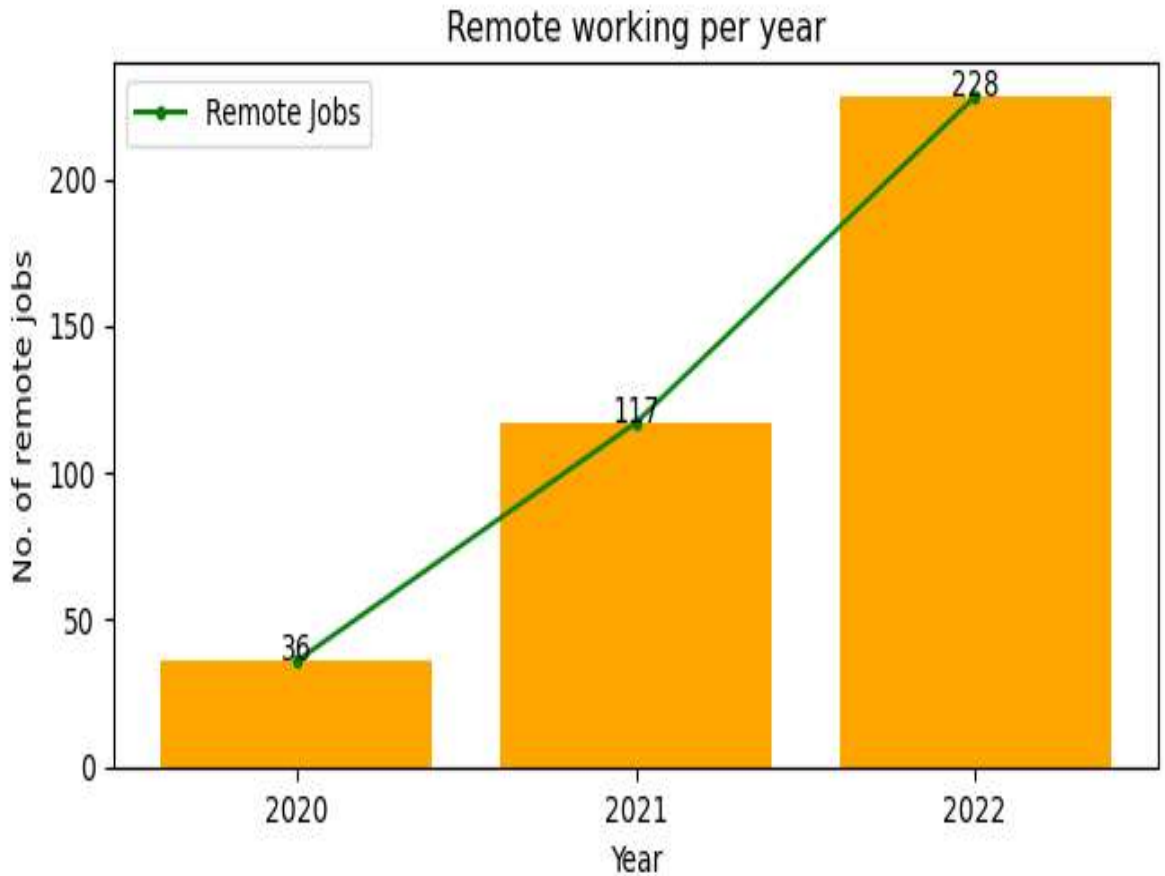
CODE:

```
remote_year = df.groupby(['remote_ratio','work_year'])['remote_ratio'].count()
```

```
lista = list(remote_year)[6:]
labela = ['2020','2021','2022']
```

```
graph = plt.bar(labela,lista,color = 'Orange')
plot2 = plt.subplot()
plot2.plot(labela,lista,marker=".",color='Green')
ax = plt.subplot()
plt.title("Remote working per year")
plt.xlabel("Year")
plt.ylabel("No. of remote jobs")
plt.legend(['Remote Jobs'])
for i,g in enumerate(graph):
    ax.text(x=g.get_x() + g.get_width() / 2, y=g.get_height()+0.10, #x and y are the positions where we can add the text, s is th
      s="{}".format(round(g.get_height(),2)),
           ha = 'center')

plt.show()
```

OUTPUT:



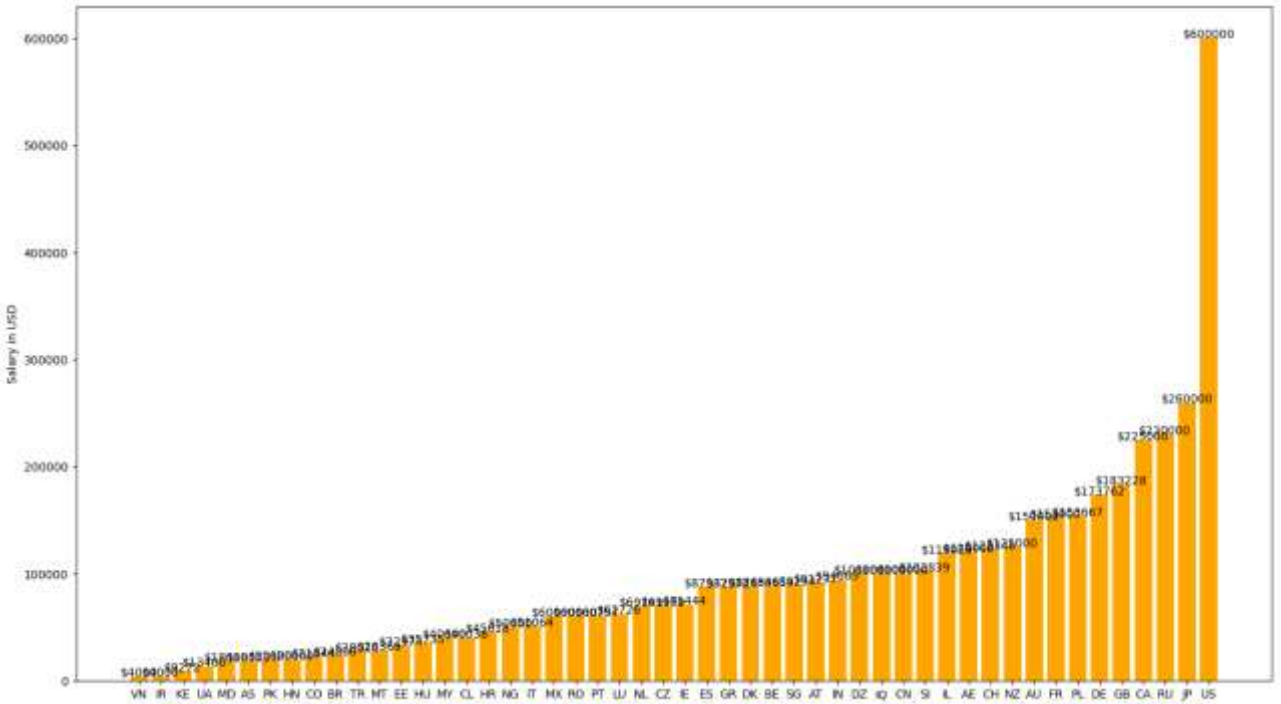
12. Which country has the highest pay?

CODE:

```
result = pd.DataFrame(df.groupby('company_location')['salary_in_usd'].aggregate(['max']))
result = result['max'].sort_values()
lista = list(result)
labela = list(result.index)
```

```
fig, ax = plt.subplots()
ax.set_ylabel('Salary in USD')
plt.rcParams["figure.figsize"] = [18.50, 10.50]
graph = plt.bar(labela,lista,color = 'Orange')
ax = plt.subplot()
for i,g in enumerate(graph):
    ax.text(x=g.get_x() + g.get_width() / 2, y=g.get_height()+0.10, #x and y are the positions where
      s="${}".format(round(g.get_height(),2)),
           ha = 'center')

plt.show()

print("As per the Given Data US has the highest pay of ${}".format(max(lista)))
```

OUTPUT:

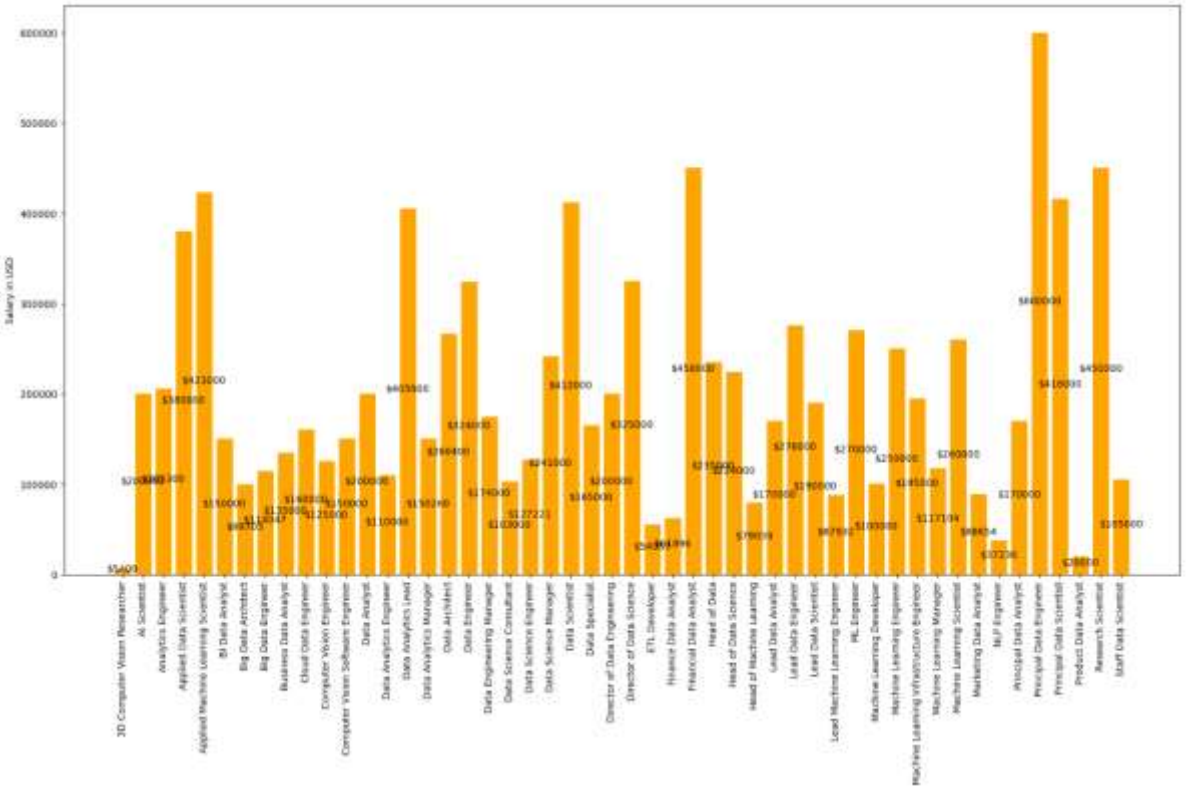


13. Which job title has the highest pay?

CODE:

```
result = df.groupby('job_title')['salary_in_usd'].aggregate(['max'])
#result = result['max'].sort_values()
lista = list(result['max'])
labela = list(result.index)
fig, ax = plt.subplots()
ax.set_ylabel('Salary in USD')
plt.rcParams["figure.figsize"] = [20.50, 10.50]
graph = plt.bar(labela,lista,color = 'Orange')
ax = plt.subplot()
for i,g in enumerate(graph):
    #x = (g.get_height()/2) if(i%2==0) else (g.get_height()/3)
    ax.text(x=g.get_x() + g.get_width() / 2, y=g.get_height()/2, #x and y are the positions where we
      s="${}".format(round(g.get_height(),2)),
          ha = 'center')
plt.xticks(rotation=90, horizontalalignment="center")

plt.show()

print("As per the Given Data Principle Data Engineer has the highest pay of ${}".format(max(lista)))
```

OUTPUT:



14. Is freelancing is worth or not?

CODE:

```
free_lance = df.where(df.employment_type == 'FL')['salary_in_usd'].median()

other_employment = df.where(df.employment_type != 'FL')['salary_in_usd'].median()
```

```
ax = plt.subplot()
graph = plt.bar(['Free Lancing','Other'],[free_lance,other_employment],width=0.4)
for i,g in enumerate(graph):
    #x = (g.get_height()/2) if(i%2==0) else (g.get_height()/3)
    ax.text(x=g.get_x() + g.get_width() / 2, y=g.get_height()/2, #x and y are the positions where we can add t
      s="${}".format(round(g.get_height(),2)),
          ha = 'center')
plt.title("Free Lancing V/S Other Employemet(Average Salary)")
plt.rcParams["figure.figsize"] = [5.50, 6.0]
#plt2.bar("Other",other_employment,color = 'Yellow',width=0.5)
plt.show()

print("As per the Data we can conclude that Free Lance Employees earn less than other employees")
```

OUTPUT: