## Scenario

‘Sysarmy’ is an internet group and community aimed at IT workers and employees, with focus on news, tips, advice, and general support. This group often makes a survey about the community job, salary (both in local currency and in USD) years of experience, and much more

The results of the survey are made public for everyone. It can be accessed from this link: 2022.1 - sysarmy - Encuesta de remuneración salarial Argentina - Hojas de cálculo de Google

As a personal project, I decided to analyze the DB. The first step is to gather general insights about all types of jobs, and then take a specific on ‘Developers’ and ‘Data Analyst’ salary increase based on job experience

## Analytical Process

1) Defining the problem:

Since this is a personal analysis, I chose to focus on giving insights to any new person or candidate willing to seek an IT job, so they know how much they can expect in terms of raw salary. In the second part of the analysis, I focus on the ‘Developer’ and ‘Data Analyst’ jobs to analyze growth and experience of the community members. For this specific exercise, we consider only “Full-Time”, “Part-Time” and “Freelance” employees

General insights can be obtained with an analysis of statistical mean, mode, max and min for each job. Specific insights can be obtained with the mean, per years of experience, of both jobs for a Full-Time contract

2) Data Source and Tools:

The survey is available on csv. Since it is a relatively small dataset, I did not consider setting up a SQL DB. Instead, I preferred to load the entire csv into memory with Python and work with is as a Pandas DataFrame

I choose Python over R because I have more experience with it

```
path = PATH
base_df = pd.read_csv(path)
```

3) Cleaning data

First we will detail the cleaning process for the general insights. The first step is to know what jobs are listed. For this case, I used a pivot table with the count() function. There were a lot of 1 or 2 jobs listed. So I applied a filter to only consider the count is greater than 10

```
temp = pd.DataFrame(base_df["Trabajo de"])

d = temp.groupby("Trabajo de").filter(lambda x: len(x) > 10)

c = d.groupby("Trabajo de").count()
```

I will calculate the statistical values with the raw salary. So I must make sure that the entire column is on INT type variable

```
df_FT_AVG = df_FT2["Salario mensual o retiro BRUTO (en tu moneda local)"]
AVG_list = df_FT_AVG.values.tolist()
AVG_list_clean = []

for elem in AVG_list:
    try:
        AVG_list_clean.append(int(elem))
    except:
        pass
```

There have been some cases where the value inserted in the column was not an INT (for example: 96800 ARS ). These kinds of values will be ignored.

A total of 149 rows are ignored, from a total of 4838. This represents nearly 3.07%

Another problem found is that the data in the salary may not be accurate. At this moment, anything below 5 digits (10.000) is not a living wage, even for a Part-Time. There are some rows with 2 or 3 digits. I think they wanted to mean thousands, but since I can't be sure, and it would mess with the average, I decided to take them out.

I also sort the list to be able to obtain the max and the min with simple list comprehension

```
AVG_list_sort = [v for v in AVG_list_clean if v > 10000]
AVG_list_sort.sort(reverse=True)
print(len(AVG_list_clean)-len(AVG_list_sort))
```

A total of 166 rows are ignored with this filter, from a total of 4689. This represents nearly 3.54%

It can happen that a certain contract and job does not produce any results (For example: VP / Freelance). In this case, all the statistical calculations would fail. Also, if there is more than one mode, it would print an error.

To solve these issues, I use Try/Except to assign a '0' value

For the specific 'Developer' and 'Data Analyst' cases, besides the general steps, I used the similar method to get the most popular 'years of experience'. I ignore 'half-years' (for example: 1.2 and 1.5)

A total of 35 rows are ignored, from a total of 1897 rows. This represents nearly 1.85%

4) Analysis

For General Insights, I use two 'for' loops, for contract type and for job type. Then I put them into graphs. I use a horizontal bar chart for simple visualization

```
contratos = ["Full-Time",
             "Part-Time",
             "Freelance"
            ]
profesiones = ["Architect",
               "BI Analyst / Data Analyst",
               "Business Analyst",
               "Consultant",
               "DBA",
               "Data Engineer",
               "Data Scientist",
               "Designer",
               "Developer",
               "HelpDesk",
               "Infosec",
               "Manager / Director",
               "Middleware",
               "Networking",
               "Product Manager",
               "Project Manager",
               "QA / Tester",
               "Recruiter / HR",
               "Sales / Pre-Sales",
               "Scrum Master",
               "SysAdmin / DevOps / SRE",
               "Technical Leader",
               "UX",
               "VP / C-Level"
              ]



for cont in contratos:
    graph_mean = []
    graph_mode = []
    graph_max = []
    graph_min = []
    df_FT = pd.DataFrame(base_df.loc[base_df["Tipo de contrato"] == cont])
    for prof in profesiones:
        df_FT2 = pd.DataFrame(df_FT.loc[df_FT["Trabajo de"] == prof])
        df_FT_AVG = df_FT2["Salario mensual o retiro BRUTO (en tu moneda local)"]
        AVG_list = df_FT_AVG.values.tolist()
        AVG_list_clean = []
```

```
        for elem in AVG_list:
            try:
                AVG_list_clean.append(int(elem))
            except:
                pass



        AVG_list_sort = [v for v in AVG_list_clean if v > 10000]
        AVG_list_sort.sort(reverse=True)

#       print(len(AVG_list_clean)-len(AVG_list_sort))

        try:
            mean_FT = statistics.mean(AVG_list_sort)
        except:
            mean_FT = 0
        try:
            mode_FT = statistics.mode(AVG_list_sort)
        except:
            mode_FT = 0
        try:
            max_FT = AVG_list_sort[0]
        except:
            max_FT = 0
        try:
            min_FT = AVG_list_sort[-1]
        except:
            min_FT = 0

        graph_mean.append(mean_FT)
        graph_mode.append(mode_FT)
        graph_max.append(max_FT)
        graph_min.append(min_FT)

    fig_1 = plt.figure(figsize = (20,30))
    plt.barh(profesiones,graph_mean)
    plt.title(cont+": Mean by Profession")

    fig_2 = plt.figure(figsize = (20,30))
    plt.barh(profesiones,graph_mode)
    plt.title(cont+": Mode by Profession")

    fig_3 = plt.figure(figsize = (20,30))
    plt.barh(profesiones,graph_max)
    plt.title(cont+": Max by Profession")
```

```
    fig_4 = plt.figure(figsize = (20,30))
    plt.barh(profesiones,graph_min)
    plt.title(cont+": Min by Profession")

    plt.show()
```

For the specific insights, I get the statistical mean for each year, and put them into a dictionary. Then I graph the dictionary.

```
#Full-Time Dev

df_dev = pd.DataFrame(base_df.loc[(base_df["Tipo de contrato"] == "Full-Time")
                                  & (base_df["Trabajo de"] == "Developer")
                                  ])

#print(df_dev)

df_dev_salary = pd.DataFrame(df_dev["Salario mensual o retiro BRUTO (en tu moneda
local)"])
dev_list = df_dev_salary.values.tolist()
dev_list2 = []

for elem in dev_list:
    try:
        dev_list2.append(int(elem[0]))
    except:
        pass



dev_list_clean = [v for v in dev_list2 if v > 10000]
dev_list_clean.sort(reverse=True)


dev_AVG = statistics.mean(dev_list_clean)
dev_mode = statistics.mode(dev_list_clean)
dev_max = dev_list_clean[0]
dev_min = dev_list_clean[-1]

print(dev_AVG)
print(dev_mode)
print(dev_max)
print(dev_min)


df_dev_exp = pd.DataFrame(df_dev[[
    "Salario mensual o retiro BRUTO (en tu moneda local)",
    "Años en el puesto actual"]])
```

```
years_dict_dev = {}

for key in years:
    dev_df_years = pd.DataFrame(df_dev_exp.loc[df_dev_exp["Años en el puesto actual"] ==
key])
    dev_df_years_AVG = dev_df_years["Salario mensual o retiro BRUTO (en tu moneda
local)"]
    dev_years_AVG_list = dev_df_years_AVG.values.tolist()
    dev_years_AVG_list_clean = []

    for elem in dev_years_AVG_list:
        try:
            dev_years_AVG_list_clean.append(int(elem))
        except:
            pass

    try:
        dev_years_AVG = statistics.mean(dev_years_AVG_list_clean)
    except:
        dev_years_AVG = 0


    years_dict_dev[key] = dev_years_AVG

 fig_dev = plt.figure(figsize = (15,5))
plt.bar(*zip(*years_dict_dev.items()))
plt.title("Full-Time Dev: AVG by years of experience")
plt.show()


#Full-Time Data Analyst



df_da = pd.DataFrame(base_df.loc[(base_df["Tipo de contrato"] == "Full-Time")
                                 & (base_df["Trabajo de"] == "BI Analyst / Data Analyst")
                                 ])


df_da_salary = pd.DataFrame(df_da["Salario mensual o retiro BRUTO (en tu moneda
local)"])
da_list = df_da_salary.values.tolist()
da_list2 = []

for elem in da_list:
    try:
        da_list2.append(int(elem[0]))
```

```
    except:
        pass



da_list_clean = [v for v in da_list2 if v > 10000]
da_list_clean.sort(reverse=True)


da_AVG = statistics.mean(da_list_clean)
da_mode = statistics.mode(da_list_clean)
da_max = da_list_clean[0]
da_min = da_list_clean[-1]

print(da_AVG)
print(da_mode)
print(da_max)
print(da_min)


df_da_exp = pd.DataFrame(df_da[[
    "Salario mensual o retiro BRUTO (en tu moneda local)",
    "Años en el puesto actual"]])


years_dict_da = {}

for key in years:
    da_df_years = pd.DataFrame(df_da_exp.loc[df_da_exp["Años en el puesto actual"] ==
key])
    da_df_years_AVG = da_df_years["Salario mensual o retiro BRUTO (en tu moneda local)"]
    da_years_AVG_list = da_df_years_AVG.values.tolist()
    da_years_AVG_list_clean = []

    for elem in da_years_AVG_list:
        try:
            da_years_AVG_list_clean.append(int(elem))
        except:
            pass

    try:
        da_years_AVG = statistics.mean(da_years_AVG_list_clean)
    except:
        da_years_AVG = 0

#    print(years_AVG)
```

```
    years_dict_da[key] = da_years_AVG

print(years_dict_da)

fig_da = plt.figure(figsize = (15,5))
plt.bar(*zip(*years_dict_da.items()))
plt.title("Full-Time Data Analyst: AVG by years of experience")
plt.show()
```

Results:

There are 12 graphs annexed in .zip format. These represent the general insights.

I will detail general insights only for Full-Time contracts. These are the top 5 from highest to lowest

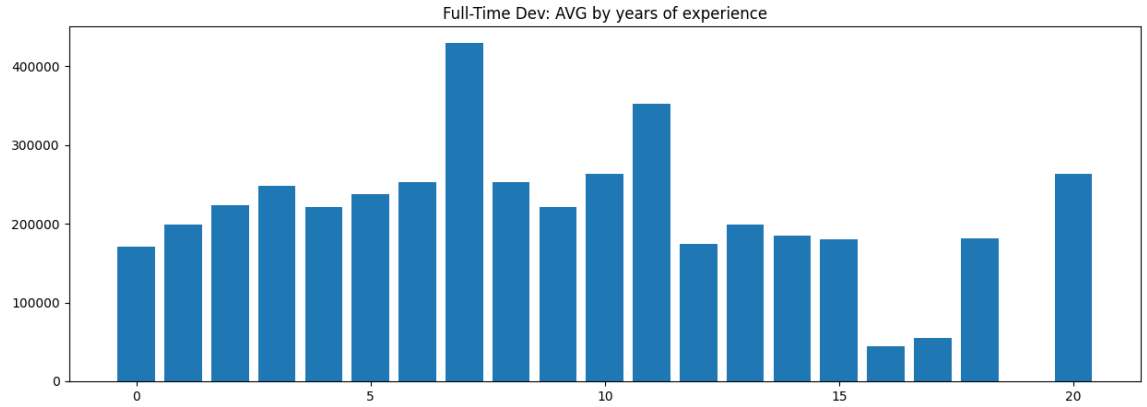
VP, Manager, Architect, Tech Leader, Product Manager are the professions with highest average
Manager, VP, Recruiter, Architect and Project Manager are the professions with highest mode
Developer, Sysadmin, Tech Leader, VP and Architect are the professions with highest max value
Scrum Master, Middleware, DBA, Sales and Infosec are the professions with highest min value

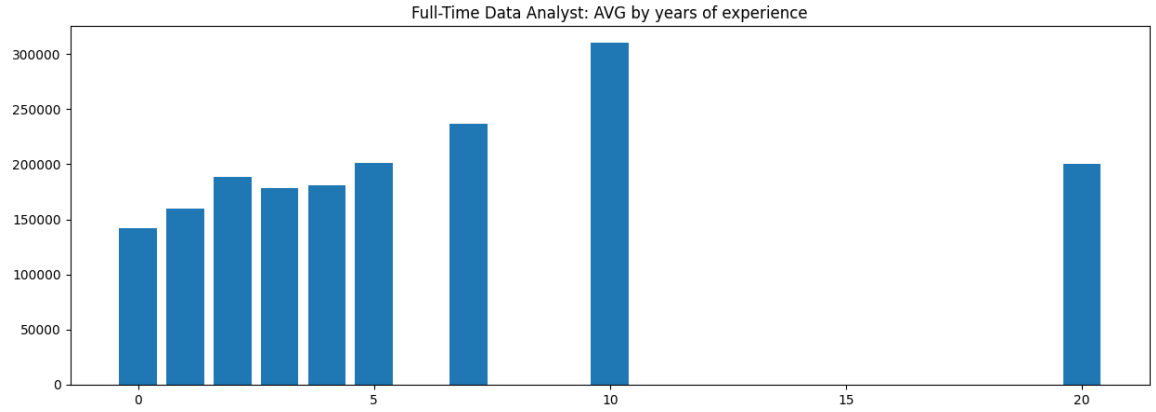
The highest average and mode belongs to the executive and leadership roles, like VP, Managers and Tech Leaders.

An interesting insight is that technical roles, like Developer and Sysadmin, can achieve the highest individual salary, but the average is lower. Which means that only a few of hundred developers and sysadmin can reach this level

Some ‘specialist’ roles like Scrum Master and Middleware get the best minimum salary. It can be because there are not many employees with these skills

As for the specific insights:




From the Developer graph, it can be seen that the average grows as they have more experience until around 10 years of experience. Then it decreases. I do not know the reason behind this phenomenon.

We can see that the Data Analyst graph has less data on higher years of experience. The reason behind this anomaly is that Data Analysis is a relatively new job

Anomalies:

- Dev: Years 7 and 11 show a much higher average than the rest, but it is not a problem of the amount of data used: these numbers are shown in the appendix
- Dev: For years 17 and 18, only one data entry was taken. The low value probably represents a bad data entry from the survey
- DA : Year 10 has a bigger growth than expected. There are only two data entries, so it may be the cause

5: Conclusions and next steps:

The IT sector is big, and it grows more every year. A new person now has insights on what salary to expect for each sector. This may help someone who is between two, or more, professions. As a personal recommendation, I would say to not follow what gives more money, but what they like instead.

A good first step for the analysis would be to collect more data and investigate the anomalies found

Specific insights were made for Full-Time Developers and Data Analysts because of my personal preference. Specifics insights for other jobs can be done

For this analysis, I did not consider company size, company reviews and happiness of the employees. A more in-depth analysis can be made to gather insights about what companies are the best to work

6: Appendix:

Años en el puesto actual DEV

| | |
|---|---|
| 0.0 | 695 |
| 0.5 | 2 |
| 1.0 | 417 |
| 1.2 | 1 |
| 1.3 | 4 |
| 1.5 | 20 |
| 2.0 | 308 |
| 2.5 | 2 |
| 3.0 | 144 |
| 4.0 | 83 |
| 5.0 | 65 |
| 6.0 | 25 |
| 7.0 | 29 |
| 8.0 | 15 |
| 9.0 | 6 |
| 10.0 | 31 |
| 11.0 | 7 |
| 12.0 | 7 |
| 13.0 | 7 |
| 14.0 | 2 |
| 15.0 | 10 |
| 16.0 | 1 |
| 17.0 | 1 |
| 18.0 | 3 |
| 20.0 | 6 |
| 22.0 | 1 |
| 23.0 | 1 |
| 24.0 | 1 |
| 25.0 | 2 |
| 33.0 | 1 |

Años en el puesto actual DA

| | |
|---|---|
| 0.0 | 59 |
| 1.0 | 30 |
| 1.5 | 1 |
| 2.0 | 27 |
| 2.5 | 1 |
| 3.0 | 7 |
| 4.0 | 5 |
| 5.0 | 5 |
| 7.0 | 1 |
| 10.0 | 2 |
| 20.0 | 1 |
| 60.0 | 1 |