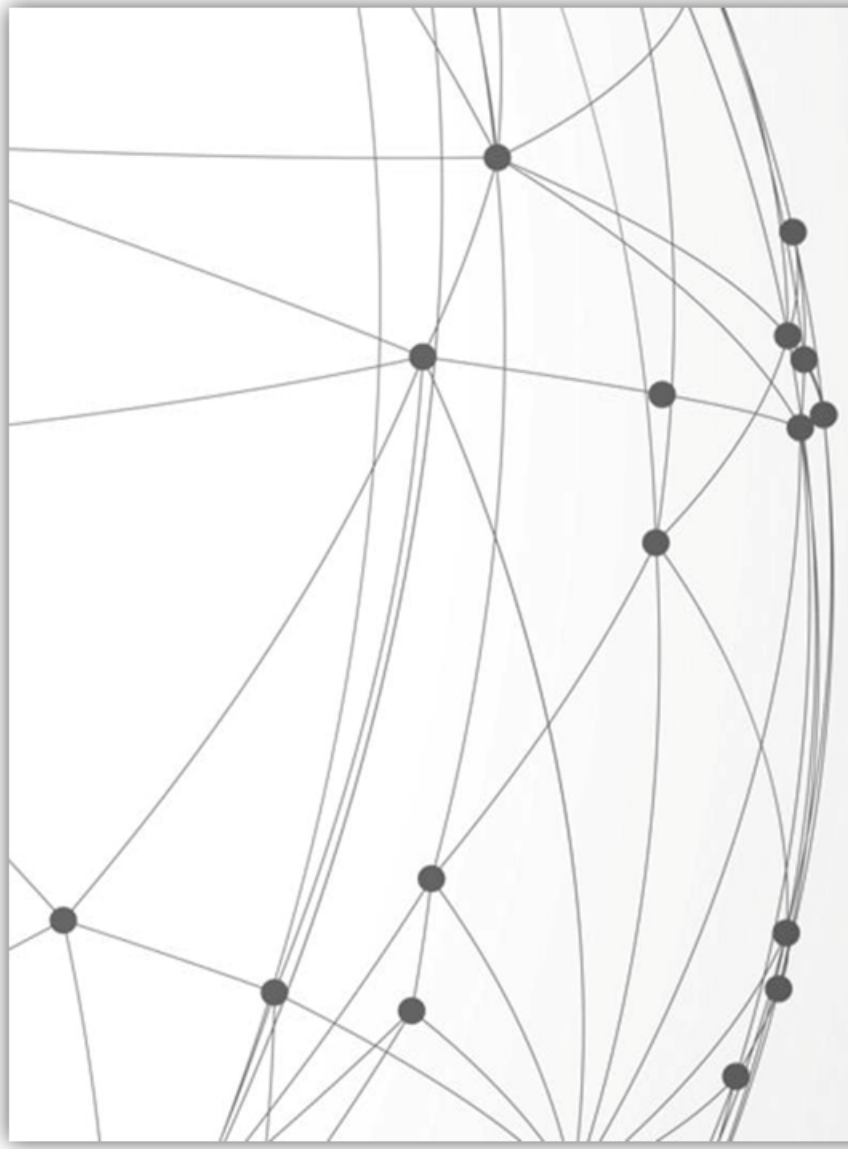


DATA-DRIVEN INSIGHTS: UNDERSTANDING SALARIES IN DATA PROFESSIONS

IT Salary Survey Dataset 2018,2019,2020

kaggle

The Story of Data Professions : Salary Survey Analysis

OVERVIEW:

This is my Independent Analysis on Salary of Data Professions using IT Salary Survey Dataset from Kaggle.The goal of this project is to analyze a Survey dataset on data related fields containing information related to salaries, Technology ,age, gender, years of experience, company sizeand employment in a various fields.

It is important to note that the analysis conducted in this project is based on survey data collected from participants within a specific field, the findings and conclusions derived from this analysis may not represent an accurate reflection of the entire population or provide definitive insights.

The sole purpose of this analysis is to gain insights into the salary trends, identify patterns, understand the relationships between different variables and showcase my skills in data analysis.

METHODOLOGY:

As part of the Vizual Intelligence Study: In this project I downloaded the dataset from kaggle and utilized Python - **Pandas and Excel** for data preparation , process and clean the data for my further analysis The dataset was then visualized using Tableau, using various charts and graphs to explore the data and derive insights.

The focus was on presenting a comprehensive exploratory analysis with storytelling elements, utilizing the power of **Tableau**'s visualizations to effectively communicate the findings.

The Story of Data Professions : Salary Survey Analysis

Top N Cities: 5

City: (All)

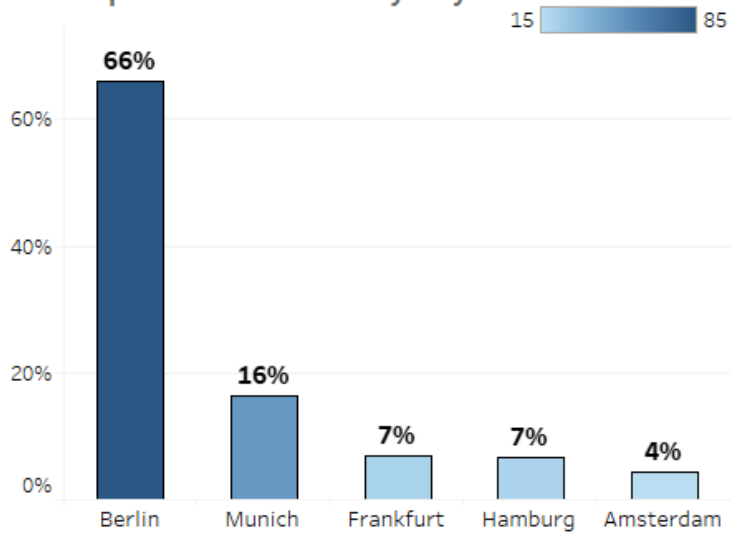
Particpants Distribution in Survey

According to this Survey data, Berlin has the highest concentration of participants with 50% of the total count. This is followed by Munich with 12% of the total count. Frankfurt and Hamburg each have 5% of the total count, while the remaining cities each have 3% or less.

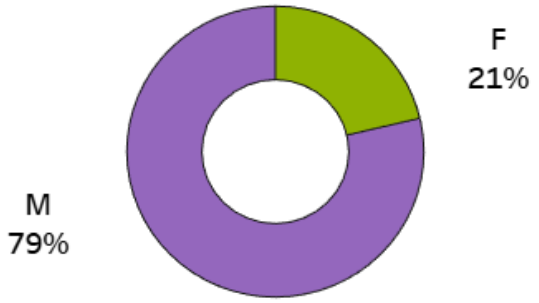
This suggests that Berlin is a major location for the Survey being analyzed in this report, with a significant concentration of participants.The other cities have much smaller concentrations of participants in comparison. Also the gender wise participants Male participants are nearly 80% and Females are 20%.

Overall, the Survey data indicates that there is a high concentration of participants in Berlin and Munich, with majorly Men in the Survey included and smaller concentrations in other cities. This provides a valuable insights into the distribution and perception of the further Analysis.

Participants Distribution by City



Participants Distribution by Gender



The Story of Data Professions : Salary Survey Analysis

Annual Salary (bins)

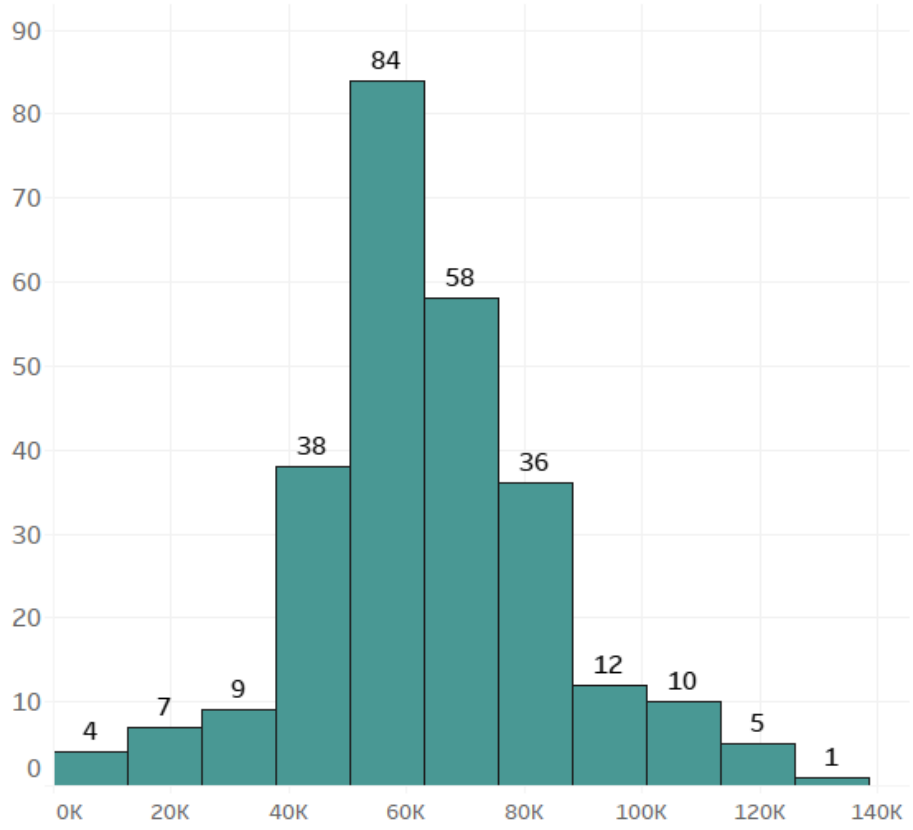
0 — 10

Based on the Histogram Analysis, it looks like most people belonging to Data Professions had a yearly salary between 38,000 and 63,000. This range had the highest number of people, with 84 people earning around 50,000 and 58 people earning around 63,000.

As we move away from this range, the number of people decreases. Also there are fewer people with lower salaries, for example, only 4 people with no salary and 7 people earning around 13,000. Similarly, there are fewer people with higher salaries, like 10 people earning around 101,000 and 5 people earning around 113,000.

Overall, it seems that the majority of participants earned between 38,000 and 63,000, while fewer people had lower or higher salaries

Distribution of Salaries in the Dataset



The Story of Data Professions : Salary Survey Analysis

The analysis reveals that salaries in data professions vary based on seniority levels. Senior positions tend to have higher salaries compared to middle positions

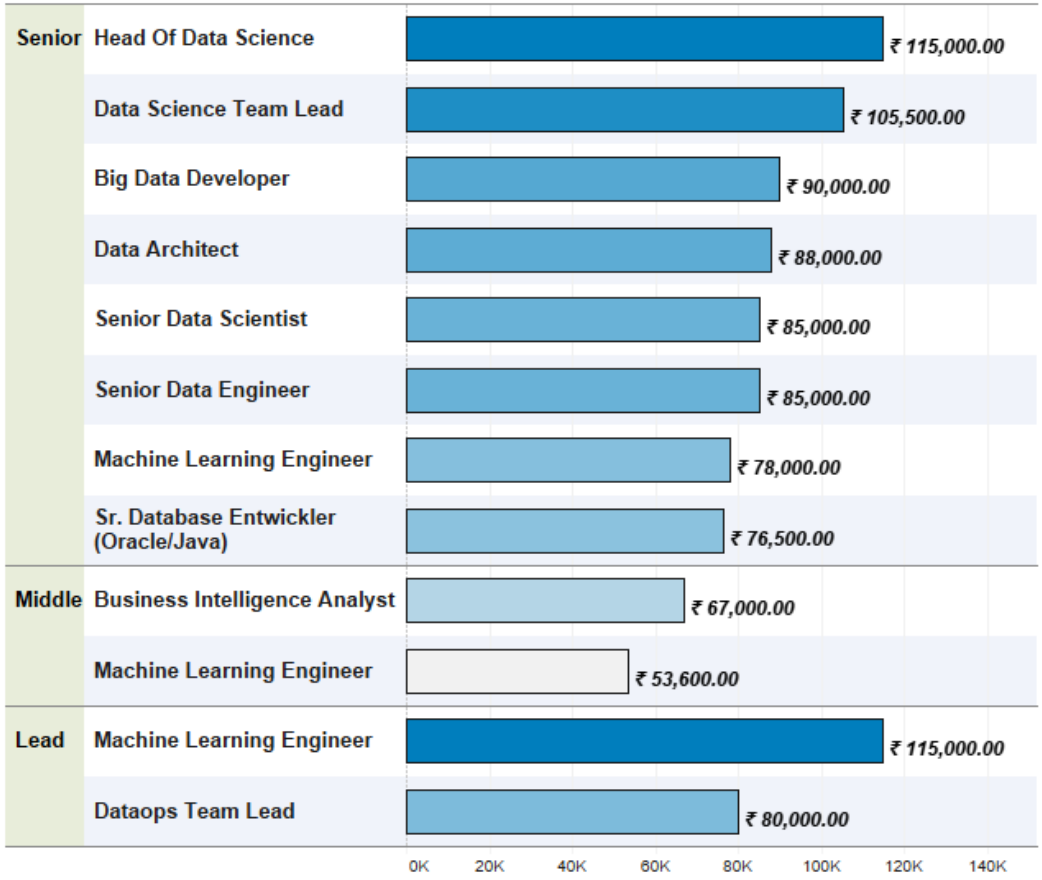
So based on the data from the analysis, it appears that the salaries for different positions in data professions vary depending on the seniority level and also the role.

For example, a Senior BI Consultant has a salary of 55,000, while a Middle BI Consultant has a salary of 46,000. Similarly, a Senior Machine Learning Engineer has a salary of 78,000 while a Middle Machine Engineer has a salary of 53,600.

Seniority level: (Multiple values)

Top N: 10

Median Salary by Seniority Level and Position



The Story of Data Professions : Salary Survey Analysis

So from the scatter plot analysis we can draw insights like,,

In the company range of 1-50, Males have higher salary compared to females in the same age group for example 21-30 age group females have average salary of 41,000.

In the 50-100 employee range, females in the 21-30 age group have lower salaries compared to males in the same age group, while females in the 31-40 and 41-50 age groups have higher salaries compared to males. Also we can see that there is an outlier in Employee range of 1-50.

In the 100-1000 employee range, females in the 21-30 and 31-40 age groups have higher salaries compared to males in the same age groups, while females in the 41-50 age group have a lower salary compared to males.

In the 1000+ employee range, females in the 21-30 age group have a lower salary compared to males, while females in the 31-40 age group have a higher salary compared to males.

So we can conclude that females tend to have higher salaries in larger companies, but Other factors, like job position, experience, and company structure may impact salary discrepancies and may influence individual salaries.

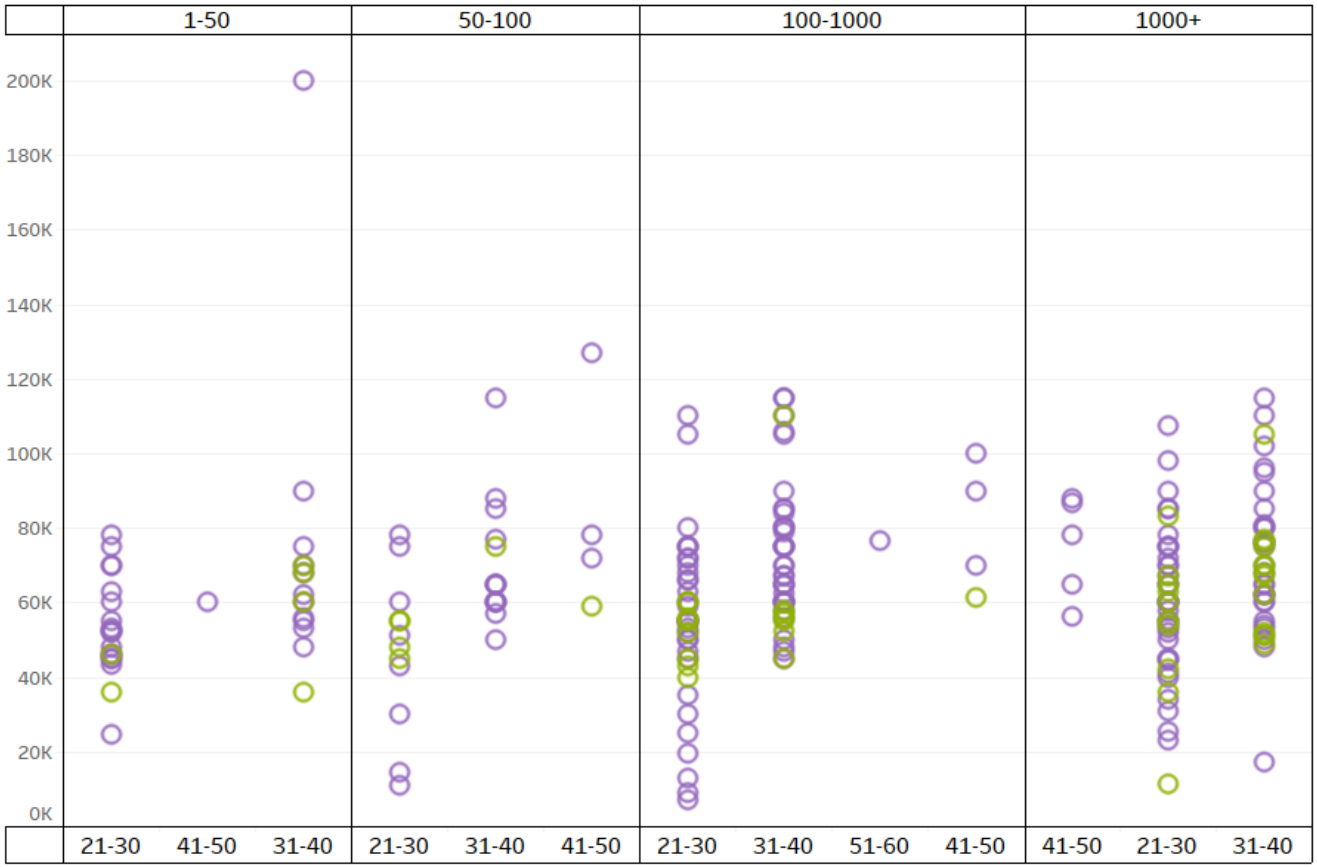
Age Group (All)

Company size group (Multiple values)

Gender (All)

Gender: F, M

How Does Salary Vary by Gender,Company Size and Age Group?



The Story of Data Professions : Salary Survey Analysis

Avg. Annual Salary: 42,650 – 66,000

Position: (All)

Total years of experience: 0 – 23

Your main technology: (All)

So from the Viz it appears that the average salaries for different positions in data professions vary depending on the technology used and the years of experience.

For example, a Data Analyst using SQL has an average salary of 51,000 with 3.25 years of experience, while a Data Analyst using Python + SQL has an average salary of 52,167 with 4.33 years of experience.

With expertise in SQL tend to have competitive average salaries while skills with Azure and Spark are associated with relatively high average salaries for Data Engineer and Data Scientist roles.

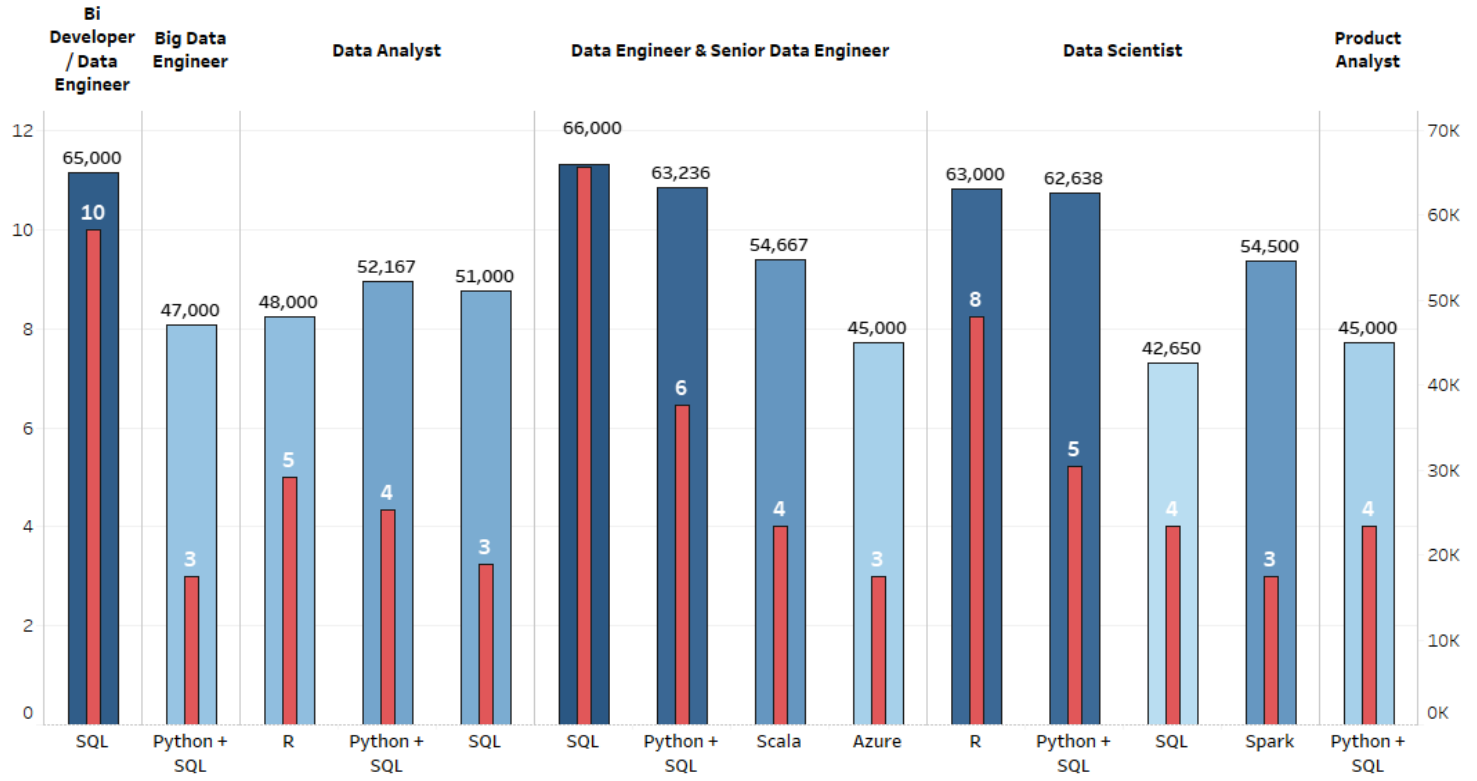
Also we can observe Data Engineer & Senior Data Engineer roles tend to have higher average salaries compared to other positions, especially when working with languages such as Scala and Python + SQL.

Also point to be noted a the average salary of 51,000 for Data Analysts using SQL with 3.25 years of experience, as well as an average salary of 52,167 for Data Analysts using Python + SQL with 4.33 years of experience, and an average salary of 48,000 for Data Analysts using R with 5 years of experience.

So both position and technology is important in determining salary levels in the data-related job market.

How Salaries Vary Acorss Different Positions with Main Technology or Pragamming Langauge and Experience

42,650 – 66,000



The Story of Data Professions : Salary Survey Analysis

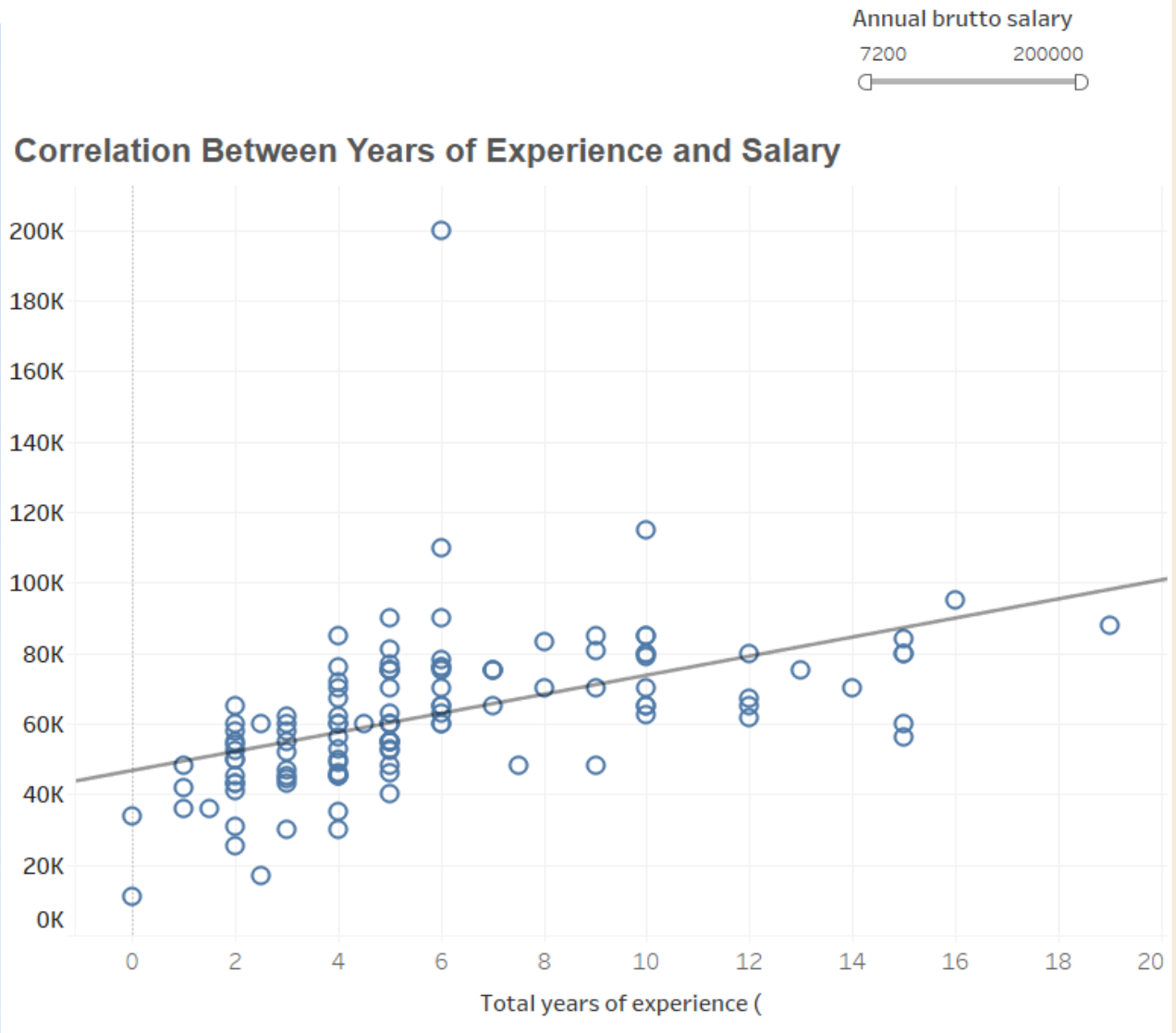
So from the scatter plot analysis, it appears that there is a positive correlation between the annual salary and the total years of experience. **As the total years of experience increases, the annual salary also tends to increase.**

For example, participants with 0 to 5 years of experience have annual salaries ranging from 11,000 to 75,000, while participants with 10 or more years of experience have annual salaries ranging from 62,500 to 115,000.

However there are some outleirs we can ignore them. For example participant with 6 years of expereince have annual salary of 200,000 which is way more than any of them.

In some instances, individuals with less experience have relatively higher salaries compared to those with more experience. For example participant with 6 years of experience have annual salary of 110,000 compared to 16 years of experience having 95,000.

This suggests that factors other than experience, such as job position, skills, or industry demand, may have influenced their salaries.



Data Professions Salary Survey Data Analysis ---- Project

@Kaggle @tableau

> ## By Mohamad Ehthesham S

Data can be found here - https://www.kaggle.com/datasets/parulpandey/2020-it-salary-survey-for-eu-region

> ## Data Processing and Data Cleaning 

Importing libraries

In [1]:
```
import pandas as pd
import numpy as np
import re
import openpyxl
```

Get Data

In [2]:
```
db_2020=pd.read_csv(r"D:\New folder (2)\Data project IT salary\Raw Files\IT Salary Survey EU  2020.csv")
db_2018=pd.read_csv(r"D:\New folder (2)\Data project IT salary\Raw Files\IT Salary Survey EU 2018.csv")
db_2019=pd.read_csv(r"D:\New folder (2)\Data project IT salary\Raw Files\T Salary Survey EU 2019.csv")


_2020=db_2020
_2018=db_2018
_2019=db_2019
```

2020

In [3]:
```
_2020.head(2)
```

Out[3]:

	Timestamp	Age	Gender	City	Position	Total years of experience	Years of experience in Germany	Seniority level	Your main technology / programming language	Other technologies/programming languages you use often	...	Annual bonus+stocks one year ago. Only answer if staying in same country
0	24/11/2020 11:14:15	26.0	Male	Munich	Software Engineer	5	3	Senior	TypeScript	Kotlin, Javascript / Typescript	...	10000
1	24/11/2020 11:14:16	26.0	Male	Berlin	Backend Developer	7	4	Senior	Ruby	NaN	...	5000

2 rows × 23 columns

Convert the 'Timestamp' column to datetime format

In [4]:
```
# seperate timestamp

_2020['Timestamp'] = pd.to_datetime(_2020['Timestamp'], format='%d/%m/%Y %H:%M:%S')

# Extract the date and time components into separate columns
_2020['Date'] = _2020['Timestamp'].dt.date
_2020['Time'] = _2020['Timestamp'].dt.time

_2020.insert(1, 'Date', _2020.pop('Date'))
_2020.insert(2, 'Time', _2020.pop('Time'))
```

In [5]:
```
_2020.head(2)
```

Out[5]:

	Timestamp	Date	Time	Age	Gender	City	Position	Total years of experience	Years of experience in Germany	Seniority level	...	Annual bonus+stocks one year ago. Only answer if staying in same country	Number of vacation days	Employment status	C c
0	2020-11-24 11:14:15	2020-11-24	11:14:15	26.0	Male	Munich	Software Engineer	5	3	Senior	...	10000	30	Full-time employee	U
1	2020-11-24 11:14:16	2020-11-24	11:14:16	26.0	Male	Berlin	Backend Developer	7	4	Senior	...	5000	28	Full-time employee	U

2 rows × 25 columns

Rename the columns

In [6]:
```
_2020.rename(columns={"Annual brutto salary (without bonus and stocks) one year ago. Only answer if staying in the same c
```

counting number of nulls

In [7]:
```
g=_2020.isnull().sum()

null_counts_df = pd.DataFrame({'Column': g.index, 'Null Count': g.values})
null_counts_df.sort_values(ascending=False,by='Null Count')
```

Out[7]:

	Column	Null Count
23	Have you been forced to have a shorter working...	880
24	Have you received additional monetary support ...	791

Rename the columns_

In [9]:
```
_2019.rename(columns={'Position (without seniority)':'Position ',"Years of experience":"Total years of experience","Yearl
```

Convert the 'Timestamp' column to datetime format

In [10]:
```
_2019['Zeitstempel'] = pd.to_datetime(_2019['Zeitstempel'], format='%d.%m.%Y %H:%M:%S')

# Extract the date and time components into separate columns
_2019['Date'] = _2019['Zeitstempel'].dt.date
_2019['Time'] = _2019['Zeitstempel'].dt.time

_2019.insert(1, 'Date', _2019.pop('Date'))
_2019.insert(2, 'Time', _2019.pop('Time'))
_2019.rename(columns={'Zeitstempel':'Timestamp'},inplace=True)
```

In [11]:
```
_2019.head(1)
```

Out[11]:

	Timestamp	Date	Time	Age	Gender	City	Seniority level	Position	Total years of experience	Your main technology / programming language	...	Yearly stocks one year ago. Only answer if staying in same country	Number of vacation days	Number of home office days per month	Main language at work
0	2019-12-02 11:18:26	2019-12-02	11:18:26	33.0	Male	Berlin	Senior	Fullstack Developer	13	PHP	...	NaN	29.0	4.0	English

1 rows × 25 columns

2018

Rename the columns

In [12]:
```
_2018.rename(columns={'Position':'Position ',"Years of experience":"Total years of experience",'Your level':'Seniority le
```

Convert the 'Timestamp' column to datetime format

In [13]:
```
_2018['Timestamp'] = pd.to_datetime(_2018['Timestamp'], format='%d/%m/%Y %H:%M:%S')

# Extract the date and time components into separate columns
_2018['Date'] = _2018['Timestamp'].dt.date
_2018['Time'] = _2018['Timestamp'].dt.time

_2018.insert(1, 'Date', _2018.pop('Date'))
_2018.insert(2, 'Time', _2018.pop('Time'))
```

In [14]:
```
_2018.head(1)
```

Out[14]:

	Timestamp	Date	Time	Age	Gender	City	Position	Total years of experience	Seniority level	Current Salary	Annual brutto salary (without bonus and stocks) one year ago	Salary two years ago	Are you getting any Stock Options?	Main language at work	Comp
0	2018-12-14 12:41:33	2018-12-14	12:41:33	43.0	M	München	QA Ingenieur	11.0	Senior	77000.0	76200.0	68000.0	No	Deutsch	100-1

Merged 2018,2019,2020

In [15]:
```
# Merge the DataFrames based on the 'Date' column while excluding duplicates
concatenated_df = pd.concat([_2018, _2019, _2020])

# Remove duplicate rows based on 'Date' column
unique_dates_df = concatenated_df.drop_duplicates(subset='Timestamp')
```

In [16]:
```
concatenated_df.head(1)
```

Out[16]:

	Timestamp	Date	Time	Age	Gender	City	Position	Total years of experience	Seniority level	Current Salary	...	Years of experience in Germany	Other technologies/programming languages you use often	(w
0	2018-12-14 12:41:33	2018-12-14	12:41:33	43.0	M	München	QA Ingenieur	11.0	Senior	77000.0	...	NaN	NaN	

1 rows × 38 columns

In [17]:
```
db=concatenated_df
```

replacing gender names to standard names

In [18]:
```
#db['Gender'].unique()
db['Gender'] = db['Gender'].replace(['Male', 'Female', 'Diverse'], ['M', 'F', 'Others'])

db['Company size']=db['Company size'].replace('up to 50-50','up to 50',regex=True)



con=db['Seniority level']=='no idea, there are no ranges in the firm '
db.loc[con,'Seniority level']='No level'

db['Seniority level'] = db['Seniority level'].apply(lambda x: str(x).capitalize())
```

cleaning non-numeric characters to numeric

In [19]:
```
db.loc[855,'Total years of experience']='23'
db.loc[1089,'Total years of experience']='6'
db.loc[1209,'Total years of experience']='0'
db.loc[805,'Total years of experience']='11'

db['Total years of experience'] = db['Total years of experience'].str.replace(',', '.')
```

conveting to numbers format all values and assigning 0 to all less than 1 year exp

In [20]:
```
db["Total years of experience"] = pd.to_numeric(db["Total years of experience"], errors='coerce')

mask=db['Total years of experience'] < 1
db.loc[mask,'Total years of experience']=0
```

Split the values in the column by comma and keep only the first part

In [23]:
```
old=['MA','GdaÅ„sk','DÃ¼sseldorf','WÃ¼rzburg','Wã¼rzburg']

new=['Massachusetts','Gdańsk','Düsseldorf','Würzburg','Würzburg']
db['City'].replace(old,new,inplace=True)
db['City'] = db['City'].str.split(',', n=1).str[0]


db['City']=db['City'].apply(lambda x:str(x).capitalize())


#removing
c=[3,4,5]
coniditon=db['Annual brutto salary (without bonus and stocks) one year ago'].astype(str).str.len().isin(c)

db.loc[coniditon,'Annual brutto salary (without bonus and stocks) one year ago']='NaN'
```

Clean the 'Main language at work' column

In [24]:
```
db['Main language at work'] = db['Main language at work'].str.lower().str.replace('[^\w\s]+', '').str.strip()

# Replace variations with standardized values
db['Main language at work'] = db['Main language at work'].replace({
```

```
    'team  russian crossteam  english': 'russian and english',
    'deutschenglisch': 'german and english',
    'englishdeutsch': 'german and english',
    'polishenglish': 'polish and english'
})

# Handle missing values
db['Main language at work'] = db['Main language at work'].replace('nan', np.nan)
db['Main language at work'] = db['Main language at work'].apply(lambda x: str(x).capitalize())

# Print unique cleaned values
unique_values = db['Main language at work'].unique()
print(unique_values)
```

```
['Deutsch' 'English' 'Russian' 'French' 'Russian and english' 'Polish'
 'Nan' 'German and english' 'Spanish' 'Italian' 'Dutch'
 'Polish and english' 'Ukrainian' 'German' 'English and german' '5050'
 'Русский' 'Czech' 'Deuglisch' 'Both' 'Russian english']
```

Cleaning Bad records in 'Number of vacation days' column

In [25]:

```
db['Number of vacation days'].replace('365','Nan',inplace=True)

# Replace non-numerical values with the mean value
db['Number of vacation days'] = pd.to_numeric(db['Number of vacation days'], errors='coerce')
mean_value = db['Number of vacation days'].mean()
db['Number of vacation days'].fillna(mean_value, inplace=True)

db.drop(columns=['0'],inplace=True)

db.rename(columns={'Ð¡ontract duration':'Contract duration'},inplace=True)
```

Create age groups

In [26]:

```
# Define the age ranges and corresponding labels for age groups
age_ranges = [(0, 20), (21, 30), (31, 40), (41, 50), (51, 60), (61, float('inf'))]
age_labels = ['0-20', '21-30', '31-40', '41-50', '51-60', '60+']

# Create a function to map age to age group
def get_age_group(age):
    for i, (start, end) in enumerate(age_ranges):
        if start <= age <= end:
            return age_labels[i]
```

Out[45]:

	Timestamp	Date	Time	Age	Gender	City	Position	Total years of experience	Seniority level	Current Salary	...	Yearly brutto salary (without bonus and stocks) in EUR	Yearly bonus + stocks in EUR	Annual bonus+stocks one year ago. Only answer if staying in same country	Employr st
7	2018-12-14 12:53:47	2018-12-14	12:53:47	36.0	F	München	Data Science Team Lead	NaN	Senior	109000.0	...	NaN	NaN	NaN	
45	2018-12-14 13:46:29	2018-12-14	13:46:29	43.0	M	Berlin	Data Engineer	NaN	Senior	84000.0	...	NaN	NaN	NaN	

2 rows × 39 columns

replacing Position names to standard names

In [30]:
```
db_filtered['Position '] = db_filtered['Position '].str.strip()

db_filtered['Position '] = db_filtered['Position '].str.title()

db_filtered['Position '] = db_filtered['Position '].str.replace(r'\bEngineet\b', 'Engineer', regex=True)

db_filtered['Position '] = db_filtered['Position '].str.replace('Databengineer', 'Data Engineer', regex=True)
```

Cleaning and Filling Null values

In [ ]:
```
cond=db_filtered['City']=='Nan'
db_filtered.loc[cond,'City']=np.nan
```

```
cond2=db_filtered['Seniority level']=='Nan'
db_filtered.loc[cond2,'Seniority level']=np.nan

cond3=db_filtered['Annual brutto salary (without bonus and stocks) one year ago']=='NaN'
db_filtered.loc[cond3,'Annual brutto salary (without bonus and stocks) one year ago']=np.nan

cond2=db_filtered['Main language at work']=='Nan'
db_filtered.loc[cond2,'Main language at work']=np.nan
```

Creating a mapping of values to be replaced

In [37]:

```
# Create a mapping of values to be replaced
replace_map = {
    'Aws': 'AWS',
    'Gcp': 'GCP',
    'Sql': 'SQL',
    'Pyrhon': 'Python'
}

# Use the pandas str methods to clean up the values in the column
db_filtered['Your main technology / programming language'] = (
    db_filtered['Your main technology / programming language']
    .str.split(',')
    .str.join(', ')
    .str.title()
    .replace(replace_map, regex=True)
)

db_filtered['Company name '] = (
    db_filtered['Company name ']
    .str.title()
    .replace(replace_map)
)
```

Create a mapping of values to be replaced

In [41]:

```
replace_map = {
    'upto 50': 'up to 50',
    '101-1000': '100-1000',
    '11-50': '10-50',
    '51-100': '50-100',
    'Oct-50':'up to 50'
}
```

```
# Use the pandas replace method to clean up the values in the column
db_filtered['Company size'] = db_filtered['Company size'].replace(replace_map)

# Define custom groupings for the company size values
group_map = {
    'up to 10': '1-50',
    'up to 50':'1-50',
    '10-50': '1-50',
    '50-100': '50-100',
    '100-1000': '100-1000',
    '1000+': '1000+'
}

# Use the pandas map method to group the company size values
db_filtered['Company size group'] = db_filtered['Company size'].map(group_map)
```

Updating to Excel File

In [42]:

```
# Load the existing Excel file into a DataFrame
existing_data = pd.read_excel(r"D:\New folder (2)\Data project IT salary\Main Data\Final Dataset4.xlsx")

# Replace the existing data with db data
updated_data = db_filtered.copy()

# Write the updated data to the Excel file, replacing the existing data
updated_data.to_excel(r"D:\New folder (2)\Data project IT salary\Main Data\Final Dataset4.xlsx", index=False)
```

> # ** Thank You *