Mittal School of Business

LOVELY PROFESSIONAL UNIVERSITY

Course Code: INTM571	Course Title: DATA ANALYTICS
Course Instructor: Ms. Tanima Thakur	Section: Q2255
Academic Task No: CA-2	Academic Task Title: **EDA on assessing the impact of Age and select variables on Data Science Industry**
Max. Marks: 50	Marks Obtained:
Evaluation Parameters:	As per rubrics

Learning Outcome

We have gained the knowledge of analyzing Data using Python and interpreting results based on those outputs.

Declaration:

We declare that this CA is our individual work. We have not copied it from any other students' work orfrom any other source except where due acknowledgement is made explicitly in the text, nor has any part been written for us by any other person.

Submitted By:

Pratyaksh Mishra
Reg. No. 12209726

Simran Sharma
Reg. No. 12202231

Evaluator's Signature and Date: 05-03-2023

Table of Content

Introduction

Over time, there has been an increasing demand for data scientists around the world. Some individuals pursue university studies before entering the field, while others rely on electronic platforms to gain knowledge and seek available work opportunities. However, age is an important factor that is absent in those who do not wait to finish their university studies. The question arises whether age really is a significant factor to consider or if it is simply a fear for those who differ from their colleagues who wait to complete their studies. As someone who belongs to the second category, I can speak about this topic in a focused manner and understand their perspective.

Whether one is in the first or second category or simply interested in working in data science, they may have questions about the requirements companies have for this field. The most crucial factor to consider is often age, which is closely linked to educational degrees and the trust of customers. This raises the question of whether there is a specific age to start working as a data scientist or if experience alone is sufficient. We will attempt to answer these questions through Kaggle's 5th annual survey.

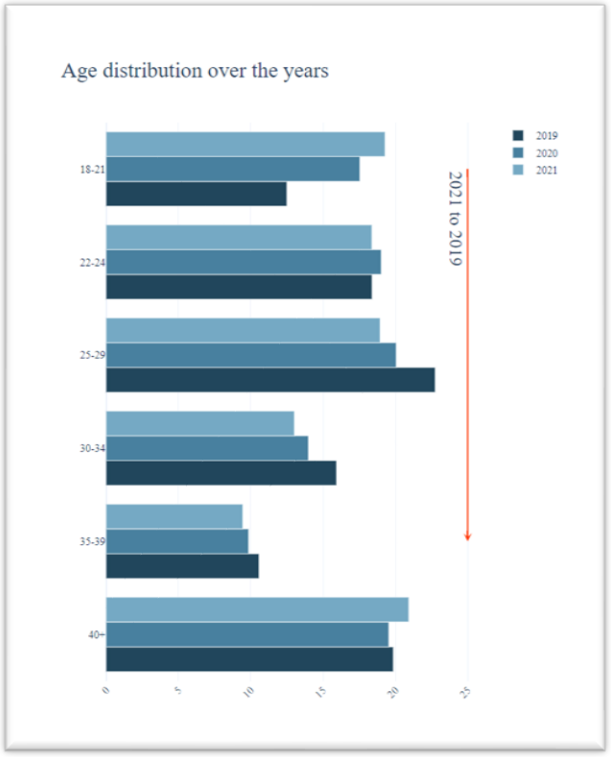
It is essential to keep in mind that when employers, company managers, or even governments set requirements that are beyond individuals' means, such as age, they are inhibiting and suppressing talents that could contribute significantly to the development of their country, company, or business. Thus, it is important to focus on individuals who are willing and able to work.

Age is typically associated with other factors such as university degrees and experience, which are all linked to annual salaries. While money is not everything in the world, it is a crucial criterion that we will use to measure the effectiveness of each of these factors to see the entire picture of the impact of age. We will analyze the distribution of each variable of interest to gain a deeper understanding and identify trends and audience demographics. We will then examine the crucial variables' connection to age and annual compensation and each other using bivariate and multivariate analyses. We will highlight all important findings and use them to provide helpful recommendations in the conclusion and recommendation section.

We will use Python for analysis along with additional libraries such as Matplotlib, Seaborn, and Plotly for data visualization, and Numpy and Pandas for data analysis and processing. We will primarily use Plotly because it is an interactive, open-source plotting library that supports many unique chart types covering a wide range of areas. All resources used in this analysis are included at the end of the report.

Some Info before we start

1. Age Distribution over the years

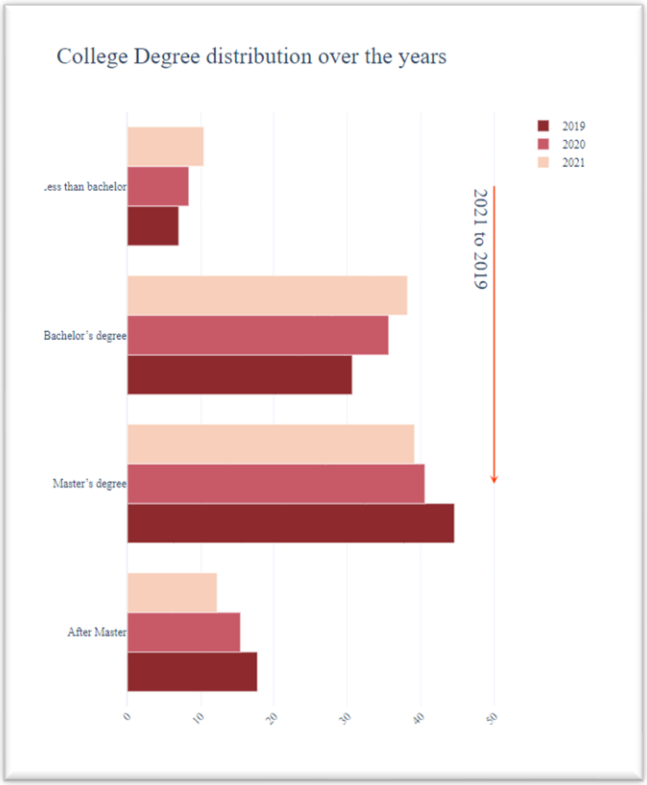


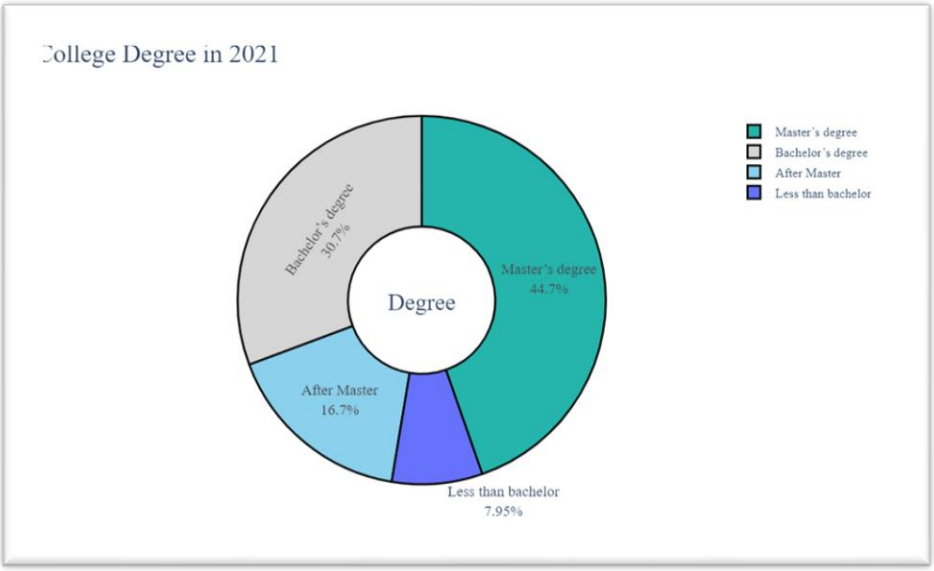
We can see some important things:

- The number of young undergraduates—or even high school students—has grown over time, indicating that adolescent awareness of data science is growing. This could intensify competition because students who show an early interest in the subject are likely to take it more seriously than those who learn and begin only in preparation for college. Of course, this isn't always the case; some people simply pick it up because it's popular among kids and they want to fit in. (College graduates and hardworking teens may challenge you!)
- Data scientists in the range of 22 to 24 have a fixed ratio over the years (Be aware that this is the age group of people who typically land their first job or even work in general), and that isn't a good thing (not to increase their ratio of contribution in the Kaggle community or even improve their skills) because that may cause for them a huge problem, which is that at the year 2024 if the ratio of the first range (18-21) still increases they will have a tough competition with those. And those people might quit their occupations, which would make them less productive and ultimately lead to unemployment.
- Data scientists between the ages of 25 and 39 have declined over time. It appears that as people get older (under 40), the proportion of data scientists contributing to Kaggle is reducing, which is excellent news for young Kagglers (Experts

contribution will not be so big maybe because they focus on their company or their own job and clients)

2. College Degree distribution over the Years





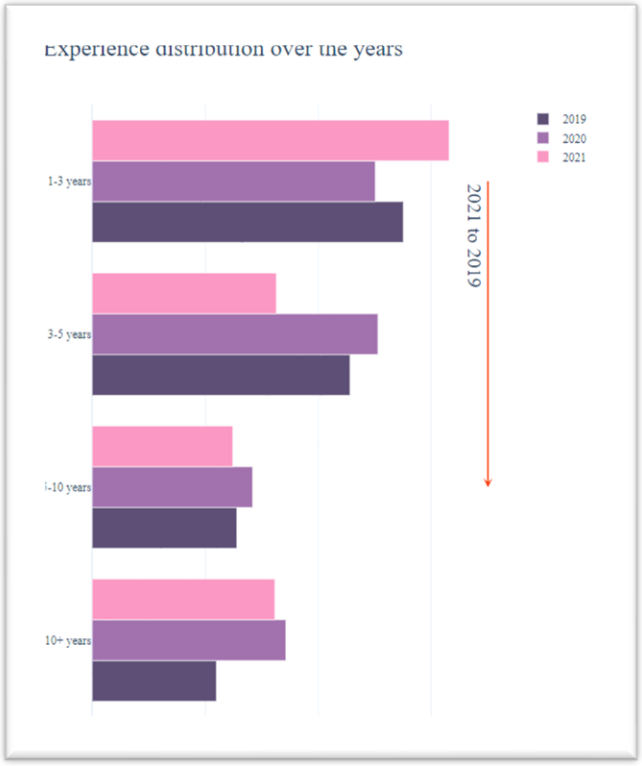
That gives us the following news:

- We don't have many people who aren't in school or attending university because about 75% of our respondents have or are enrolled in Bachelor's or

Master's degrees (or they plan to have one in the next two years). I know that university isn't the only thing in the world and that some people don't attend university and become very successful, but in general that doesn't happen for the majority of them. We will also compare experience vs. university degree and annual compensation vs. university degree.

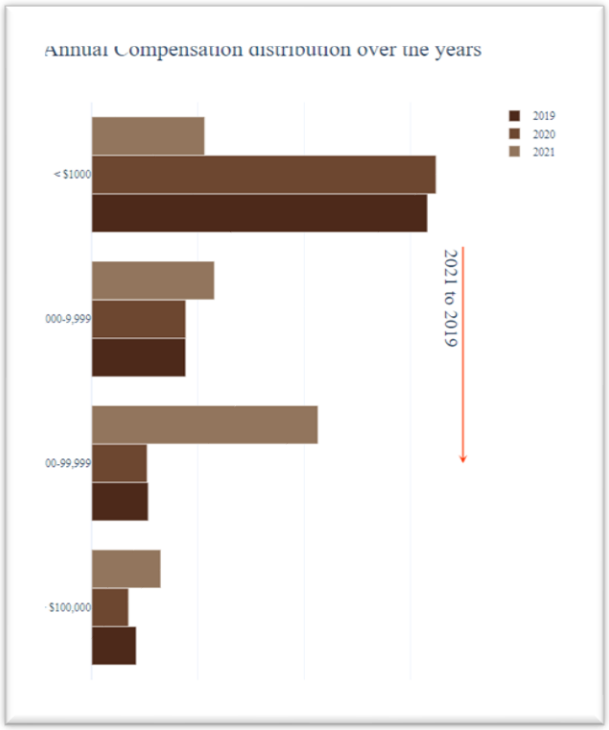
- The ratio of students pursuing bachelor's degrees has risen over time, possibly as a result of an increase in the age range of 18 to 21 years old, which is the category for bachelor's degrees.
- We've seen a significant decline in high studies (studies after a Master's degree) over the years, which is good news for students and will make the competition easier (Do students in this category not have the time to participate to Kaggle due of their studies? We shall see."

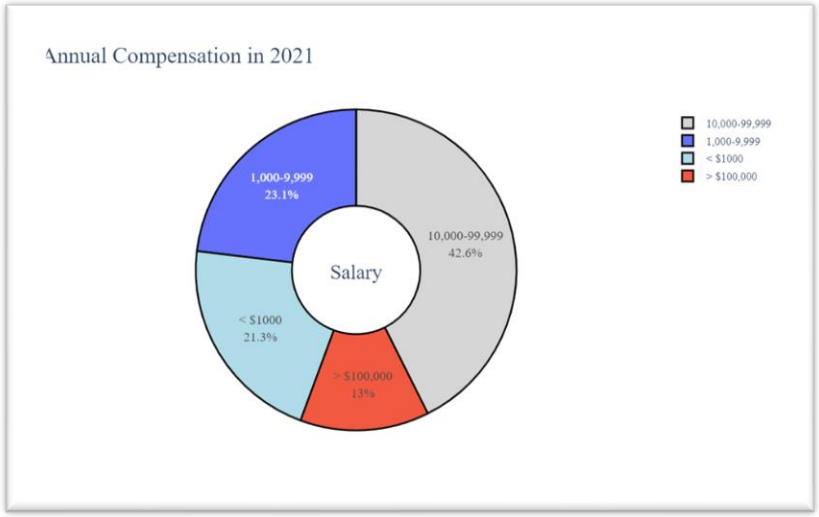
3. Experience Distribution over the Years



- Half of People in 2021 and 2019 are junior (or beginning) data scientists, according to this small note. I define "Junior" as 1-3 years of coding experience.
- This led us to conclude that, generally speaking, the more experience a data scientist has, the lower their participation rate will be in 2019 and 2021. And that provides us with the following details.

4. Annual Compensation Distribution over the years.





- We will see a significant drop in the range of $1,000 in 2021, but that's good news since we will also see a significant increase in the range of $10,000 to $99,999. Congratulations to them.
- 42.6% of People earn between $10,000 and $99,999 per year, while 55.6% earn more than $10,000.

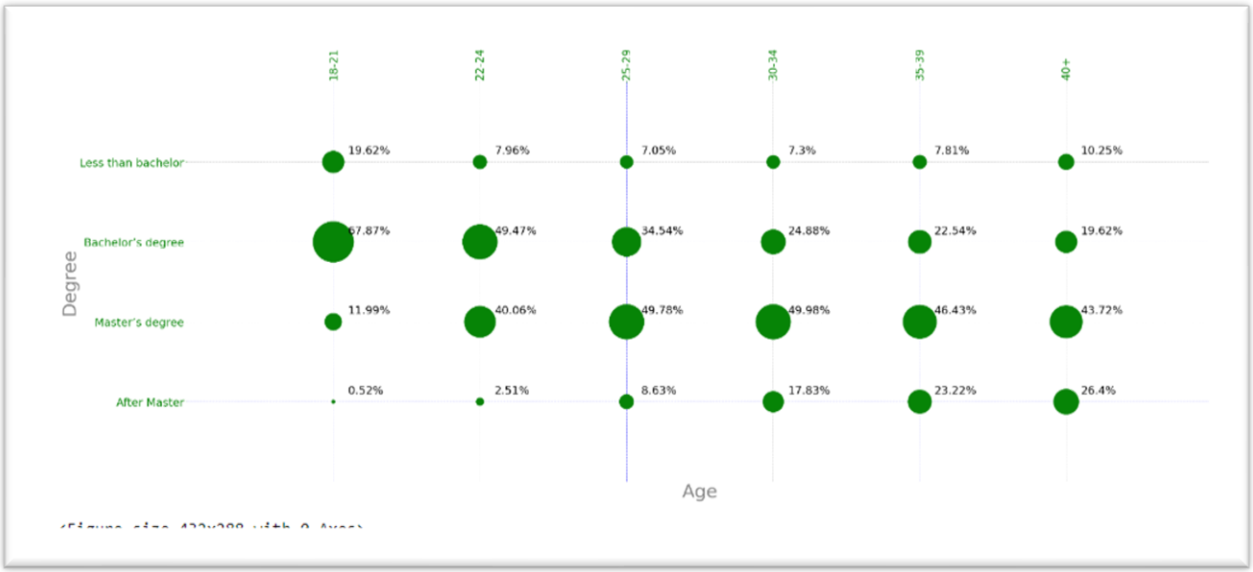
Questions about age effects

Here, we'll examine the effects of age on the relevant variables, and the key results are presented below. Some of these findings will be incorporated into the recommendations section of the article to help those who want to learn data science but are unsure of where to begin, parents who want their children to learn or work in this field (if their children want to), and the government, which is considering adding data science or programming courses for kids or providing scholarships for young people.

An overview of the categorical plot that is being used in this instance is as follows: each column's sum is 100%, and each column represents the distribution of each value on the x-axis on the y-axis values by its percentage or ratio, with the precise ratio being stated and exhibited by its size. Ex: If we have 100 master's students, the graph will look like this if 20 of them receive less than 10,000 and 80 receive higher.

	Master Student
Less than 10,000	20%
Higher than 10,000	80%

Q1. How age affects the college degree?

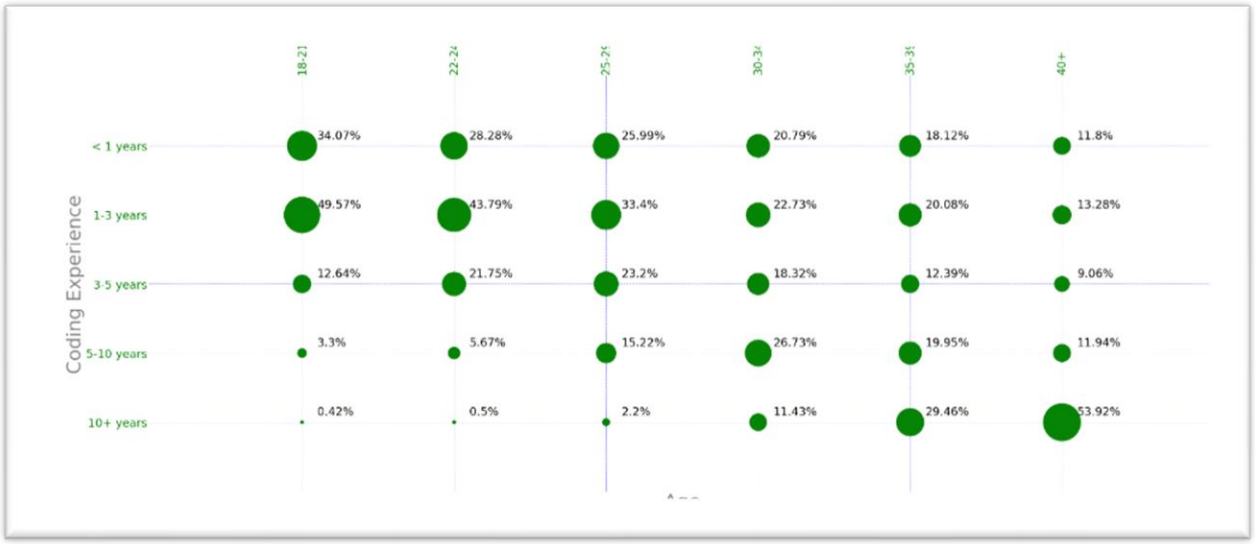


So we can see from this graph the following:

- Several individuals between the ages of 18 and 21 possess degrees that are at least equal to a master's degree. This is quite surprising because, in my home country of Egypt, attaining a master's degree at that age is unimaginable in daily life. see this: regarding this odd information.

- We can see that your chances of earning a Master's degree or above increase as you age. (And with the Bachelor or lower, vice versa)
- We can see that your chances of earning a Master's degree or above increase as you age. (And with the Bachelor or lower, vice versa)
- We can observe that around half of those with 24 years or more hold master's degrees or plan to.

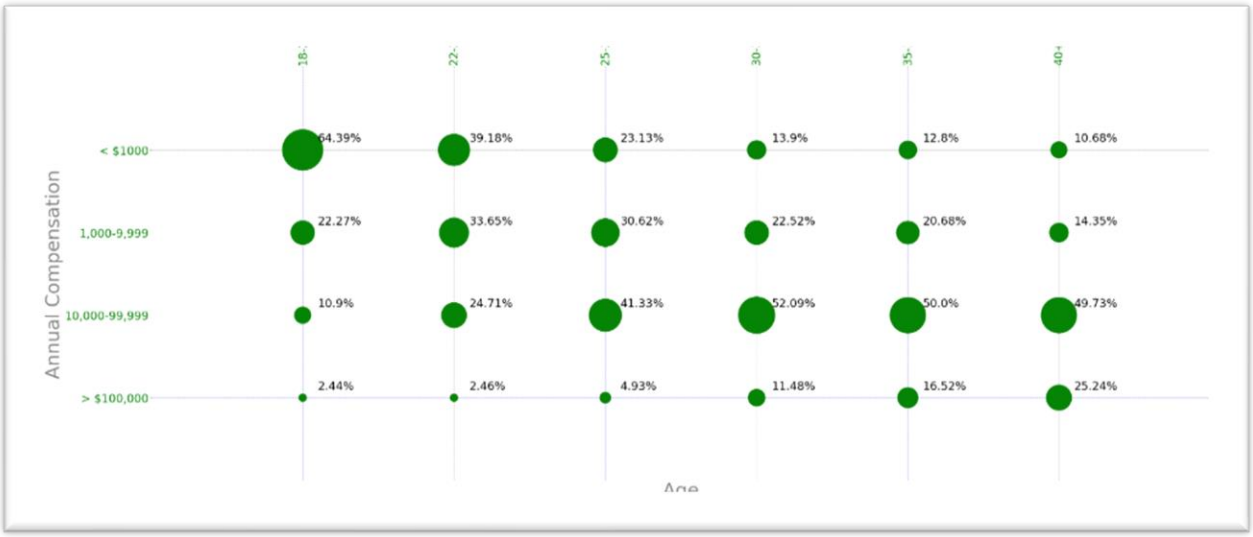
Q2. How age affects the Coding Experience?



I won't talk about the typical things like my older age, my greater experience, or anything like here. Below, I'll focus on the most crucial details:

- We have some employees in the age range of 18 to 21 who have five to ten years of experience, which indicates that some of our employees begin writing code at least at the age of twelve (I've computed the average to be between 8 and 16). We will revisit this issue in the guidance section and when we see how it affects annual compensation because it is so fantastic.
- People in the 22–24 age bracket often have a high percentage of 1-3 years of coding experience, which is common. Although they may begin that once they have registered in their university, it is still odd that this range of experience is the same for people between the ages of 18 and 21. As we showed in the previous paragraph, there are far more people than just a handful that begin programming before the age of 18.

Q3. How age affects the Annual Compensation?

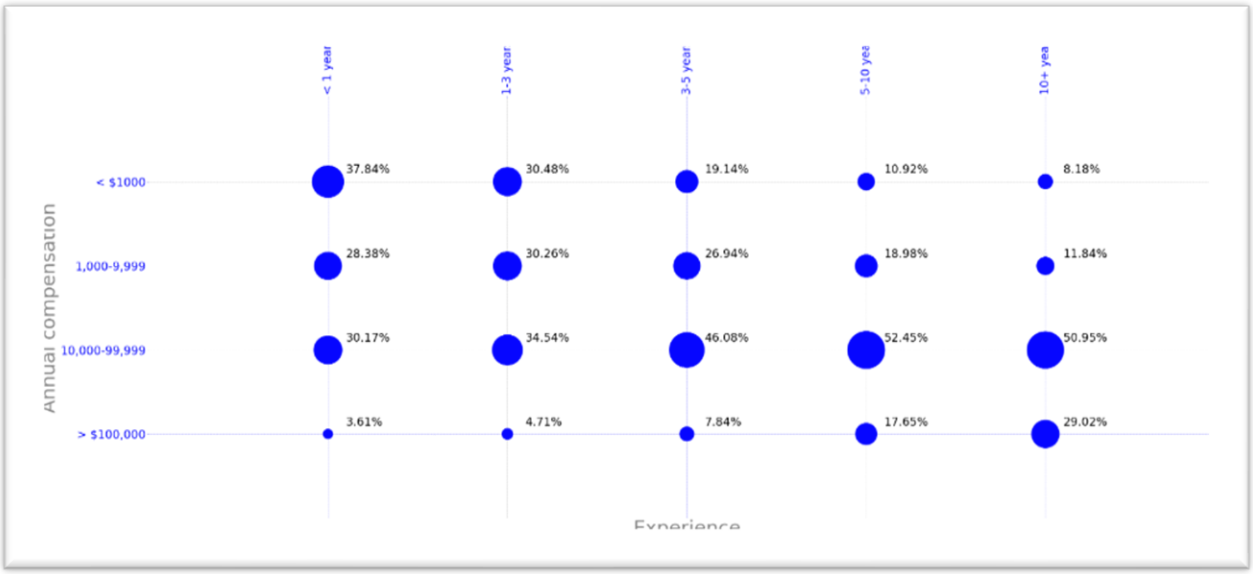


It's common to see that as you get older, your income increases, but let me show you what is very odd:

- We will see in the following sections whether any of the young people who make more than $100,000 between the ages of 18 and 21 are those who began their careers at a young age, but at least now we know that young people can succeed to such a great extent.
- We have a sizable percentage (25%) of adults over 40 who make $10,000 or less per year in salary. When talking about the range of 30-39, it's roughly 35%.

Questions about relations among other vars

Q4. Does coding experience have a huge effect on Annual Compensation?



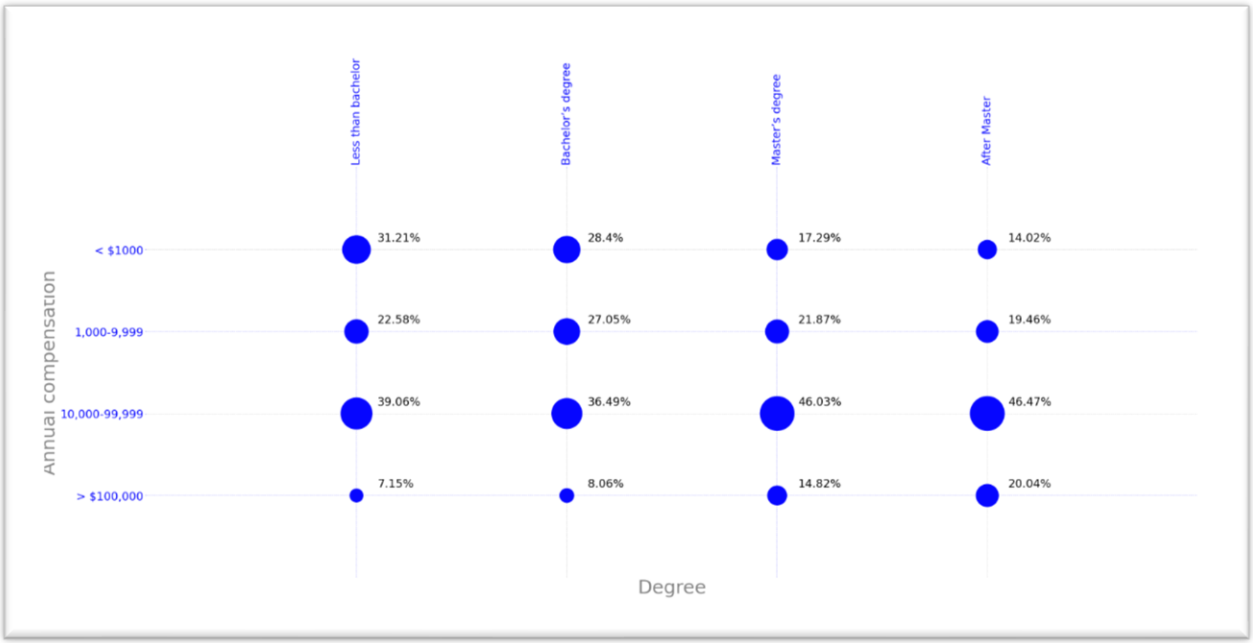
Yes, it has a positive impact, but is that a requirement for all data scientists, or even something they should particularly concentrate on?

- From the first and last ranges, it is clear that increasing Experience causes a decrease in low salary and vice versa for increasing Experience.
- We have some brilliant programmers with only a year of experience and very high salaries. This indicates that the likelihood of earning a salary of more than $10,000 with less than a year of experience is roughly 33.78%.
- With the exception of experience of five years or more, the influence of experience does not show up in the range of $1,000 to $99,999.
- Data scientists who have three or more years of experience often earn between $10,000 and $99,999 annually.
- Only after you have more than five years of experience does the likelihood that you will have more than $100,000 improve.

Summary:

With more expertise, there is a lower chance of receiving less than $1000. Except for experience longer than three years, the influence of experience will not be significant in ranges of $1,000 to 999,999 and beyond $100,000.

Q5. Does college degree have a huge effect on Annual Compensation?



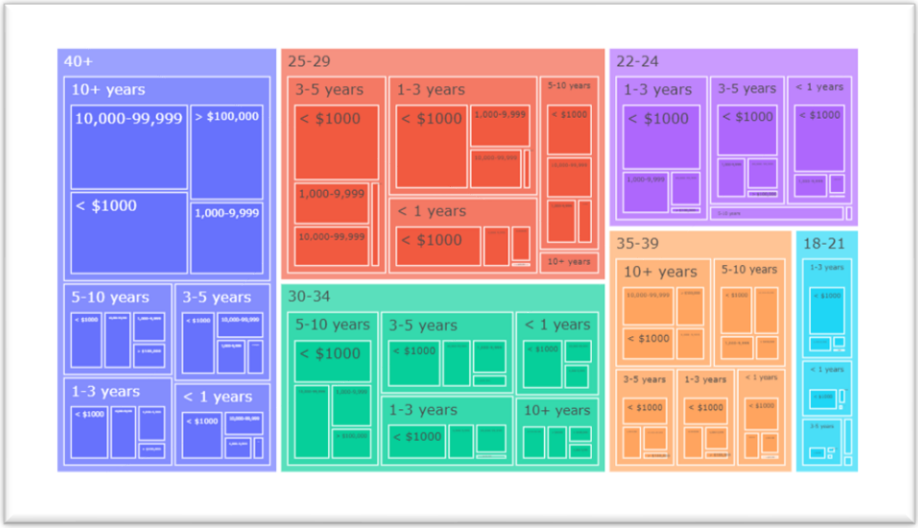
It's obvious from the few remarks that it has a positive impact, however it is significantly less than what coding experience would have. If we compare the pay range of over $100,000, we can plainly see that it starts at 7% and rises to 20% in the range of "After Master" degrees, but coding experience starts at 4% and falls to 27% in the "After Master" degrees. Moreover, a degree has a positive impact on salaries. For example, consider a wage range of $1,000 where the starting salary was 31% and ended at 14%. (From lowest degrees to highest degrees). Otherwise, we might claim that it is set between the salary ranges of $1,000 and $99,999. Understand what that means?

It indicates that your chances of landing a job with an annual salary between $10,000 and $99,999 dollars are good if someone else has a degree beyond a bachelor's and you don't. If you have 5–10 years of coding experience and his experience is theoretical, you can double this rate.

Q6. Did young people benefit financially from what they worked on in their childhood?

This part speaks about the young people who have an experience of code writing from their childhood.

Big Picture



Then let's see our group of interest which are the people who works a lot from their childhood we will consider them as following:

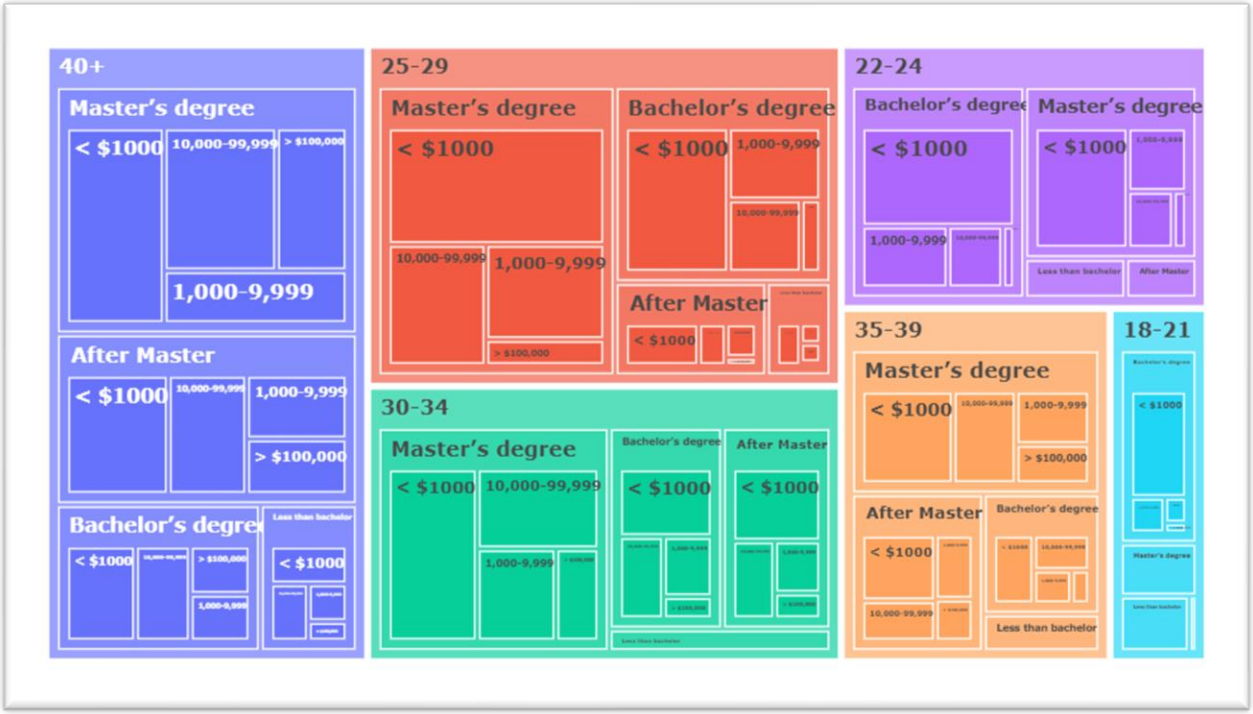
- Less than or equals 24 years old
- If they are between the ages of 18 and 21, have experience of at least 3 years; otherwise, they must have experience of at least 5 years.

Hard-working Youth

- And yes, even if they are still young, their labour has paid off and they have made money! From that success, the following cases are excluded:
  someone receives $1,000
  who receives $1,000–$9,000 and has five to ten years of experience
- We can observe that 33% of young persons (18 to 21 years old) with 3-5 years of coding expertise earn $1,000 or more, 15.7% earn more than $10,000 (this is not a typo), and 4% earn more than $100,000.
- 40% (about 50%) of young adults between the ages of 18 and 21 with 5 to 10 years of coding experience earn $1,000 or more, 15.2% earn over $10,000, and 5% earn over $100,000. Don't you see something, please? Very similar ratios exist! As a result, the following rule (based on their average) can be used: When you are 18 or 20 years old, you will have an annual income that is 36.5% greater than $1,000, 15.4% higher than $10,000, and 4.5% higher than $100,000 if you put in the effort to learn how to code while you are young (around 3+ years before you are 18).
- If we talk about the 22–24 age group, the same thing will still occur, but with better ratios. With five to ten years of experience, the ratios will be as follows:
  - 55.8% of people have more than $1,000.
  - 34.5% of people have more than $10,000.
  - 6.5% of people have more than $100,000.
- Therefore, the effect of age is seen in the 5–10 years of experience in the age range of 18–21, where you will receive $1,000 or more with a ratio of 36.5%, but 55.8% if you are in the 22–24 range; 10,000 or more with a ratio of 15.4%, but 34.5% if you are in the 22–24 range; and 100,000 or more with a ratio of 4.5%, but 6.5% if you are in the 22–24 range.

Q7. Did young people benefit financially from what they learned(from College) in their teens ?

This section addresses the young people who pursue master's degrees or above before or during the age of 24, as well as some connected issues.



These are some Questions:

Q8. Are there any age groups where you are paid more if you have a higher degree?

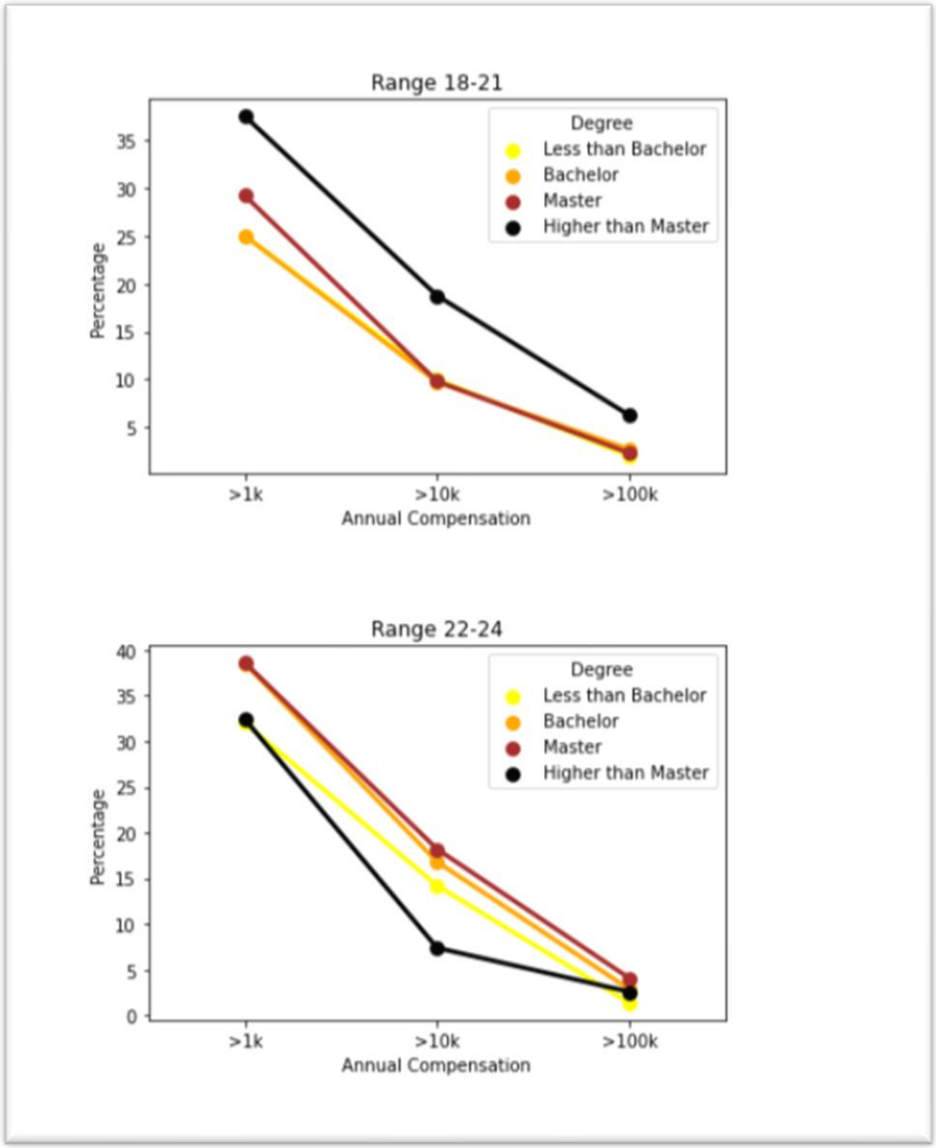
First let's read the graph to answer the questions:

Range 18-21:

- Less than Bachelor's: got over $1,000 with a ratio of 25%, over $10,000 with a ratio of 10%, and over $100,000 with a ratio of 2%.
- Bachelor's: received more than $1,000 with a ratio of 25%, more than 10,000 with a ratio of 9.7%, and more than 100,000 with a ratio of 2.7%.
- Master's: received more than $1,000 with a ratio of 29.2%, more than 10,000 with a ratio of 9.8%, and more than 100,000 with a ratio of 2.3%.
- After earning a master's degree, persons who earned more than $1,000 had a ratio of 37.5%; those who earned more than 10,000 had an 18.75% ratio; and those who earned more than $100,000 had a 6.25% ratio.

Range 22-24:

- Less than Bachelor's: received more than $1,000 with a ratio of 32%, more than 10,000 with a ratio of 14.2%, and more than 100,000 with a ratio of 1.4%.
- Bachelor's: got more than $1,000 with a ratio of 38.5%; more than $10,000 with a ratio of 16.8%; and more than $100,000 with a ratio of 2.9%.
- Master's: received more than $1,000 with a ratio of 38.6%, more than 10,000 with a ratio of 18.2%, and more than 100,000 with a ratio of 4.1%.
- After earning a master's degree, they received more than $1,000 with a ratio of 32.4%, more than $10,000 with a ratio of 7.4%, and more than $100,000 with a ratio of 2.6% (this isn't accurate because there aren't many of those people who made this claim, so we may dismiss it too).

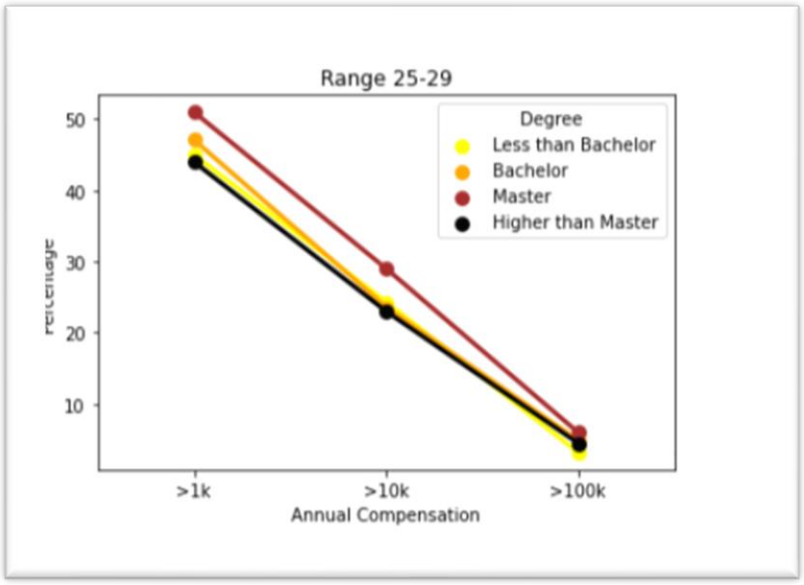


From this information, we can answer question 3 and conclude that, yes, it will benefit them directly. Having a Master in the age group of 18 to 21 will increase their likelihood of having more than $1,000 in their possession. In particular, having a bachelor's degree in this age bracket will increase your likelihood of receiving yearly remuneration in the range of $1,000

to $9.9999, and having a master's degree will increase your likelihood of receiving more than $100,000.

Range 25-29:

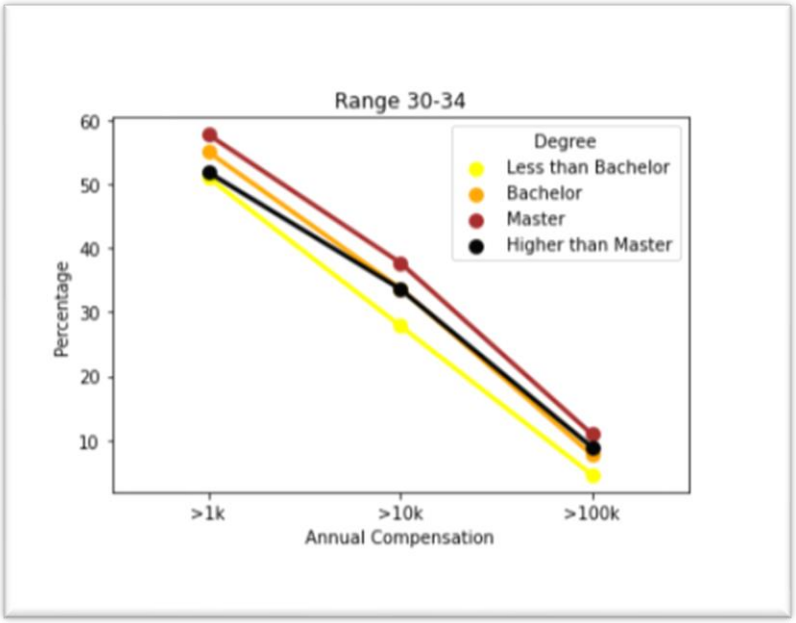
- Less than Bachelor's: received more than $1,000 with a ratio of 45%, more than 10,000 with a ratio of 24.3%, and more than 100,000 with a ratio of 3.2%
- Bachelor's: obtained more than $1,000 with a ratio of 47.1%, more than 10,000 with a ratio of 23.6%, and more than 100,000 with a ratio of 4.9%.
- Master's: received more than $1,000 with a ratio of 51%; more than 10,000 with a ratio of 29%; and more than 100,000 with a ratio of 6%.
- Following a master's degree, I earned more than $1,000 with a ratio of 44%, more than $10,000 with 23%, and more than $100,000 with 4.5%.



We only want to look at a limited sample, so we won't be doing this for everyone. We can respond to question 2 by saying that earning more than a Master's degree from a university will not increase your annual salary; on the contrary, it will reduce your likelihood of earning well. (This issue is known as "Overqualification")

Range 30-34:

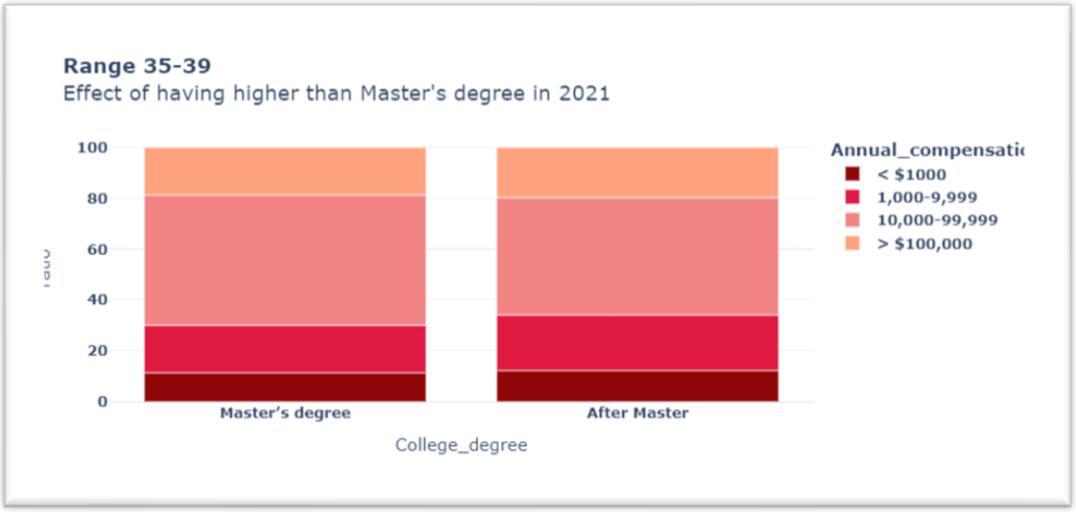
- Less than Bachelor's: received more than $1,000 with a ratio of 51%, more than 10,000 with a ratio of 27.86%, and more than 100,000 with a ratio of 4.6%.
- Bachelor's: got more than $1,000 with a ratio of 55%, more than $10,000 with a ratio of 33.7%, and more than $100,000 with a ratio of 7.7%.
- Master's: received more than $1,000 with a ratio of 57.7%; more than 10,000 with a ratio of 37.7%; and more than 100,000 with a ratio of 10.9%.
- After earning a master's degree, I received more than $1,000 with a ratio of 51.8%, more than $10,000 with a ratio of 33.6%, and more than $100,000 with a ratio of 8.9%.

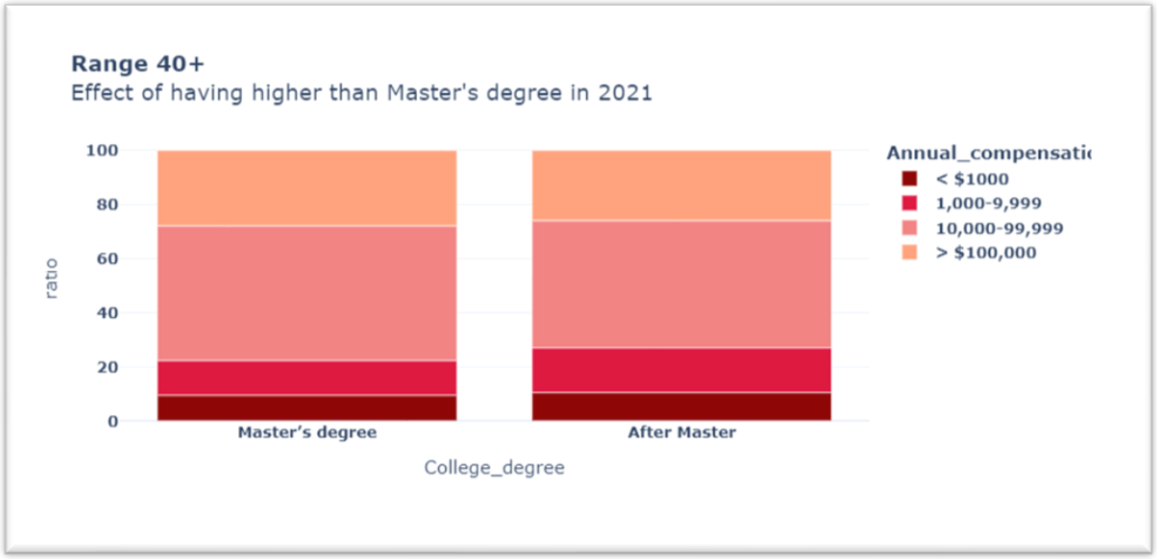


It is now obvious that earning more money will result from obtaining a higher degree (response to Question 1), but this only applies to master's degrees. This guarantees that our response to the second question, which is greater than a Master's degree for more ranges, won't produce outcomes that are superior to those of a Master's degree alone.

Q9. Do degrees earned after earning a Master's degree actually increase your annual salary in a significant way relative to the Master's degree itself?

We want to know what age range people with more education than a Master's degree receive better yearly salaries. We'll do that since we saw that it didn't improve their outcomes in the prior age groups, so we'll concentrate on the last two age groups here in the hopes that it will in these groups. We will use either the increase in receiving more than $10,000 or the decrease in receiving less than $10,000 to determine whether annual compensation has increased or decreased.

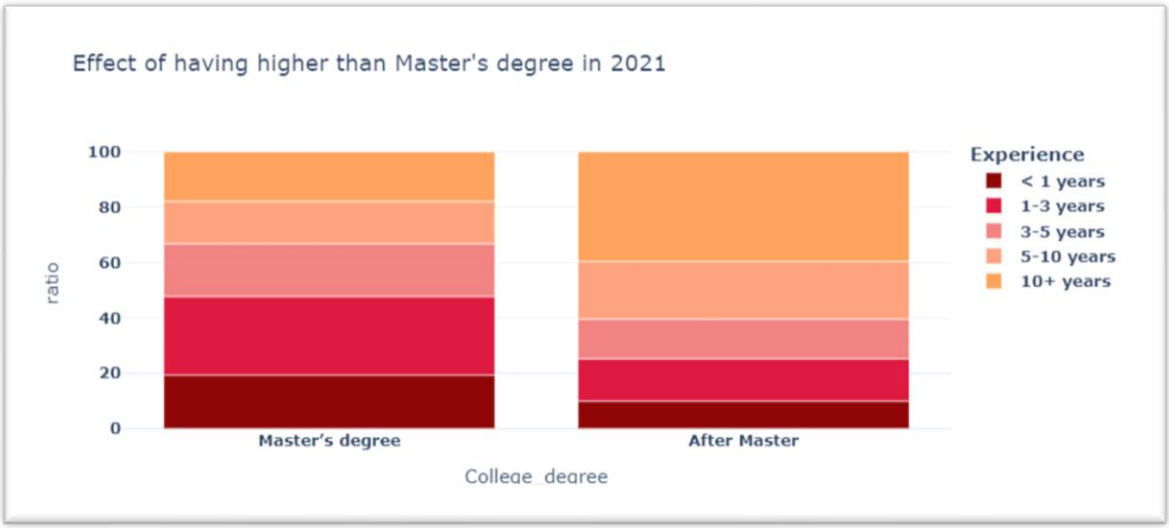




We obtained the same results in the remaining two categories as well, and we were aware that no age range would profit monetarily from holding a degree higher than a Master's degree.

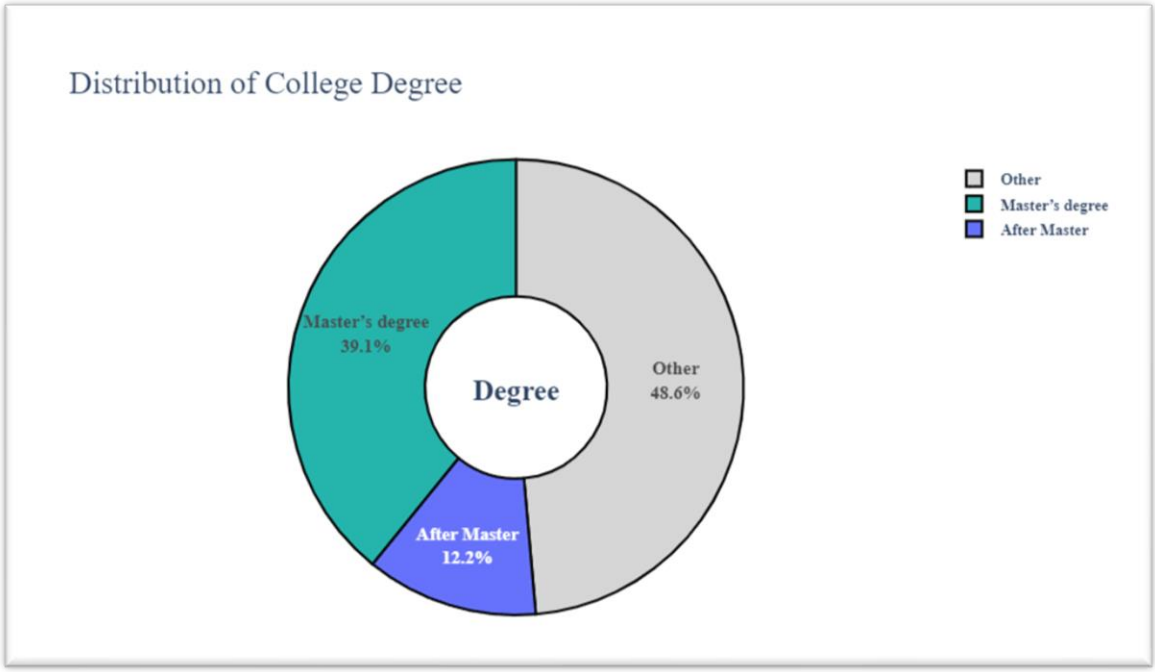
Suggestion: It would be a good idea to include a question regarding the greatest college degree you currently hold but do not intend to obtain or enrol in, as those who intend to do so will undoubtedly not receive higher annual salaries than those who do, which could be what led to the puzzling outcome. We can also ask if those with degrees higher than a master's degree have "overqualification" problems.

Q10. Does those who have Master's degree have a better coding experience than those who have or intend to have higher than Master?



The fact that those with higher education levels have greater coding experience than those with Master's degrees is not shocking, so we will disregard this factor and move on to the following one, which is participation in the Kaggle Community.

Q11. Do people with master's degrees participate at a higher rate than those who already have or want to obtain higher degrees?



Sure, they have a greater participation rate, but the sample of data scientists with a Master's degree or more isn't so small as to constitute the issue. The final two justifications are "seeking scientific achievements" and "overqualification," since it will also not be the cause. So, including it in the Kaggle 2022 yearly survey will be a welcome addition.

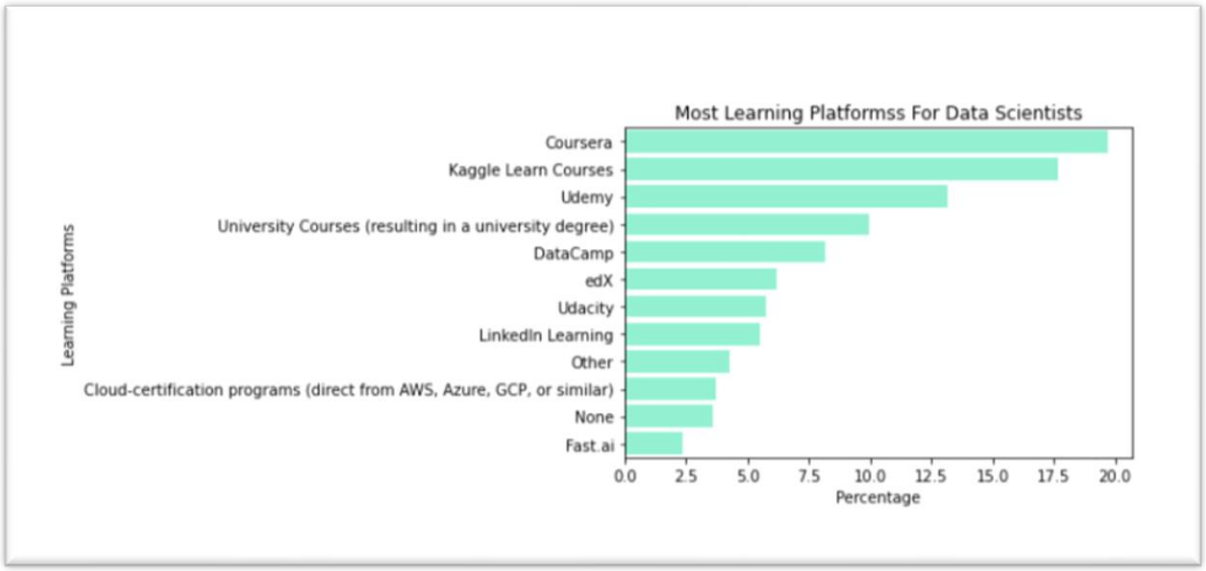
Conclusion & recommendations

As we've seen, having a college degree, being older, and having coding experience are all significant factors that have a significant impact on the annual compensation for data scientists. However, in practise, we are unable to change our age or college degree because those factors are fixed, and because college degree doesn't have a significant enough impact on annual compensation to merit our children's effort to increase it before its due.

The only variable we've identified that is adjustable is coding experience, and it also has the biggest impact on annual salary. For a few months, we might utilise our spare time to learn the fundamentals of programming, including problem-solving, before moving on to the data science requirements, which include math and statistics, analytical thinking, and machine learning techniques. I'd want to now talk about the resources you will need to use the results of this survey, such as learning platforms, tools, and the programming

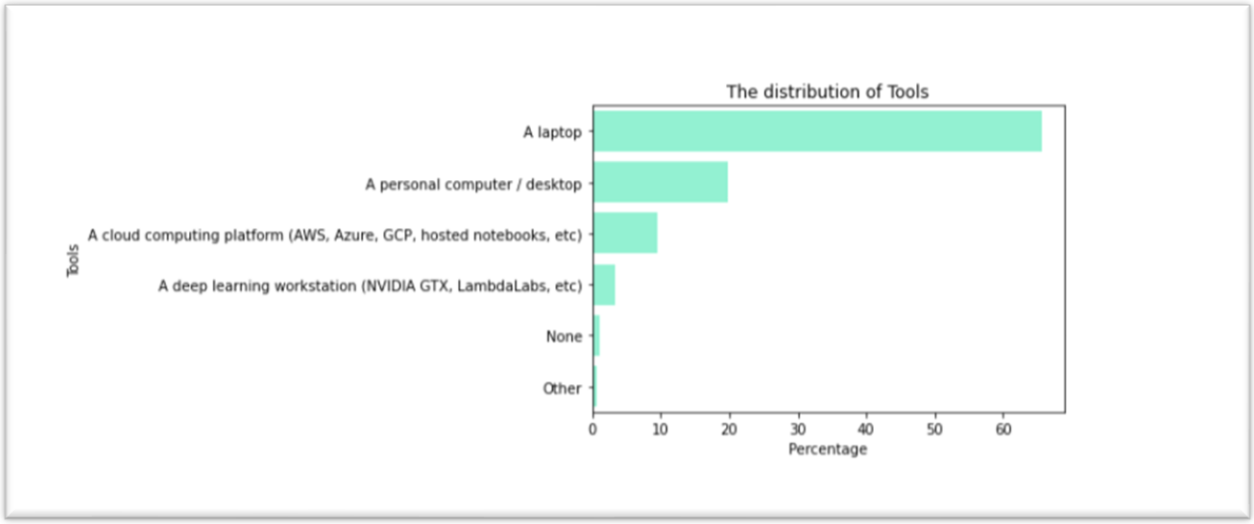
language to start with. view this Data Science Sources, Data Science Roadmap, and Programming Roadmap.

Platforms



As we can see those are the most popular platforms.

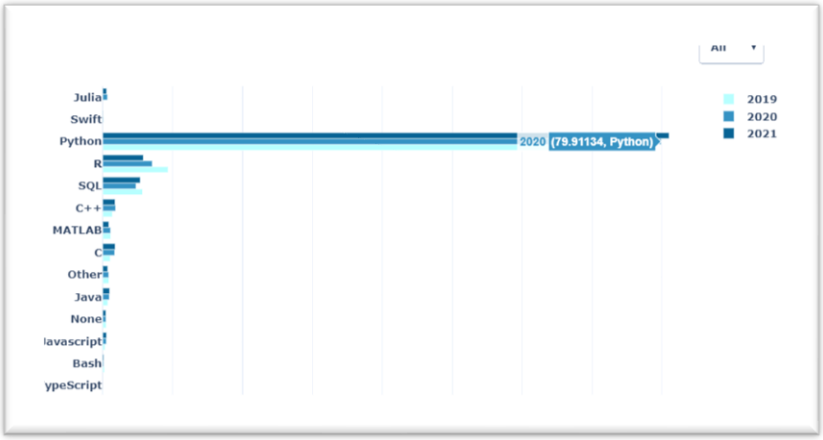
Can I Work with My Personal Computer? (Tools)



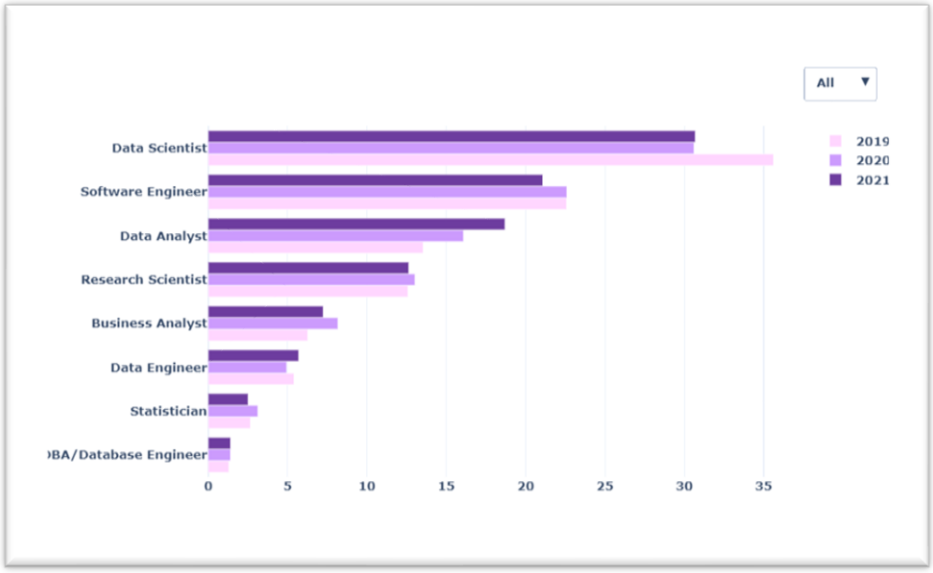
So the answer is yes, you can use your personal computer or your laptop or even buy a new laptop for yourself only for study.

Which Programming Language for Data Science?

I've made a programming language comparison. Here is a graph to see a quick overview about the popular ones.



Let's examine the most common job descriptions on Kaggle in 2021.



References

Articles about salaries:

- **High-income countries**
- **Low-income countries**

Helpful Notebooks and articles

Plotly Guide

Plotly functions explained with examples. This notebook has a very wide range of functions structured in an impressive manner.

Categorical Plots

Lots of categorical plots functions Implemented and well-documented

Waffle Plots

Implementation of waffle plots in a single smart article.

Donut Chart

Implementation of Donut plot with helpful references to learn more in depth.

Dataset

https://drive.google.com/drive/folders/18oq5GheY1WD0FtlyZ6EQfxbPw6B3IcyY?usp=sharing

ANNEXURE

Python Code from Jupyter Notebook

In [1]:
```
!pip install pywaffle
```

```
Requirement already satisfied: pywaffle in c:\users\hp\anaconda3\lib\site-packages (1.1.0)
Requirement already satisfied: matplotlib in c:\users\hp\anaconda3\lib\site-packages (from pywaffle) (3.5.1)
Requirement already satisfied: fontawesomefree in c:\users\hp\anaconda3\lib\site-packages (from pywaffle) (6.3.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\hp\anaconda3\lib\site-packages (from matplotlib->pywaffle) (4.2
5.0)
Requirement already satisfied: numpy>=1.17 in c:\users\hp\anaconda3\lib\site-packages (from matplotlib->pywaffle) (1.21.5)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\hp\anaconda3\lib\site-packages (from matplotlib->pywaffle)
(2.8.2)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\hp\anaconda3\lib\site-packages (from matplotlib->pywaffle) (1.
3.2)
Requirement already satisfied: cycler>=0.10 in c:\users\hp\anaconda3\lib\site-packages (from matplotlib->pywaffle) (0.11.0)
Requirement already satisfied: packaging>=20.0 in c:\users\hp\anaconda3\lib\site-packages (from matplotlib->pywaffle) (21.3)
Requirement already satisfied: pillow>=6.2.0 in c:\users\hp\anaconda3\lib\site-packages (from matplotlib->pywaffle) (9.0.1)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\hp\anaconda3\lib\site-packages (from matplotlib->pywaffle) (3.0.
4)
Requirement already satisfied: six>=1.5 in c:\users\hp\anaconda3\lib\site-packages (from python-dateutil>=2.7->matplotlib->p
ywaffle) (1.16.0)
```

In [2]:
```
# Data Analysis
import numpy as np
import pandas as pd

# Data Visualization
import matplotlib.pyplot as plt
from pywaffle import Waffle
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go

# Setting up
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline

# Data
survey21 = pd.read_csv("D:\INTM571\kaggle_survey_2021_responses.csv\kaggle_survey_2021_responses.csv")
survey20 = pd.read_csv("D:\INTM571\kaggle_survey_2020_responses.csv\kaggle_survey_2020_responses.csv")
survey19 = pd.read_csv("D:\INTM571\multiple_choice_responses.csv\multiple_choice_responses.csv")
```

In [3]:

```
# Here we want to select our features of interest and prepare them in this analysis which are:
# Most Important --> 'Age', 'University Degree', 'Annual compensation', 'Experience or years of writing code or dealing with
# Less Important --> 'platforms', 'country', 'Role name' and "Desktop or laptop"

# We will do that in two phases:
# First, we will select the columns and dealing with them if one of them has many columns
# Second, we will rename each column with appropriate name (putting same name in each survey dataframe through the years)


# Let's start in this cell with 2021

# 1st
# After some codes we found that and see the descriptive file:
# Q1 : 'Age'
# Q4 : 'Degree'
# Q25 : 'Annual compensation'
# Q6 : 'Experience'


# Q5 -> 'Role name' or 'Job title'
# Q3 -> 'Country'
# Q11 -> "Device" or 'Laptop or Computer'  We will speak about it in extra section, but with 2021 only
# Q40 -> 'Platforms'  it has multiple columns and we may put it in the "Extra Section" in the article

# We will call the updated dataframe srv21

features_of_interest = ['Q1', 'Q3', 'Q4', 'Q5', 'Q6', 'Q25']
srv21 = survey21[features_of_interest]
srv21 = srv21[1:]  # To exclude first row
# 2nd
srv21.columns = ['Age', 'Country', 'College_degree', 'Job_title', 'Experience', 'Annual_compensation']
```

In [4]:

```
# Now with 2020
# After checking it, it seems exactly as survey21 (in the columns' places and names so we will do the same)
# The question here and the point that want to be checked is it the same choices or not because if it is not we must fix it.

features_of_interest = ['Q1', 'Q3', 'Q4', 'Q5', 'Q6', 'Q25']
srv20 = survey20[features_of_interest]
srv20 = srv20[1:]  # To exclude first row
# 2nd
srv20.columns = ['Age', 'Country', 'College_degree', 'Job_title', 'Experience', 'Annual_compensation']
```

In [5]:

```
# Now with 2019
# After checking it, it seems somewhat as survey20 (in the columns' places and names so we will do the same except some updat
# The question here and the point that want to be checked is it the same choices or not because if it is not we must fix it a
# after updating it like what we'll do with 2020 dataframe.

features_of_interest = ['Q1', 'Q3', 'Q4', 'Q5', 'Q15', 'Q11']
srv19 = survey19[features_of_interest]
srv19 = srv19[1:]  # To exclude first row
# 2nd
srv19.columns = ['Age', 'Country', 'College_degree', 'Job_title', 'Experience', 'Annual_compensation']
```

In [6]:

```
# Ordinal vars before updating in 2021
Age_order = ['18-21', '22-24', '25-29', '30-34', '35-39', '40-44', '45-49', '50-54', '55-59', '60-69', '70+']
experience_order = ['I have never written code', '< 1 years', '1-3 years', '3-5 years', '5-10 years', '10-20 years', '20+ yea

money_order = ['$0-999', '1,000-1,999', '2,000-2,999', '3,000-3,999', '4,000-4,999', '5,000-7,499', '7,500-9,999', '10,000-14
               '15,000-19,999', '20,000-24,999', '25,000-29,999', '30,000-39,999', '40,000-49,999', '50,000-59,999', '60,000-
               '70,000-79,999', '80,000-89,999', '90,000-99,999', '100,000-124,999', '125,000-149,999', '150,000-199,999', '2
               '250,000-299,999', '300,000-499,999', '$500,000-999,999', '>$1,000,000']

degree_order = ['I prefer not to answer', 'No formal education past high school',
                "Some college/university study without earning a bachelor's degree",
                "Bachelor's degree", "Master's degree", "Doctoral degree",
                'Professional doctorate']
```

In [7]:

```
# Now we want to fix the values in the values of each feature of interest
# 'Age', 'Country', 'College_degree', 'Job_title', 'Experience', 'Annual_compensation'


# Age
# we interested in those ranges : '18-21', '22-24', '25-29', '30-34', '35-39', '40+'
replace_me = ['40-44', '45-49', '50-54', '55-59', '60-69', '70+']
s19, s20 = srv19['Age'].replace(replace_me, '40+'), srv20['Age'].replace(replace_me, '40+')
s21 = srv21['Age'].replace(replace_me, '40+')
srv19['Age'], srv20['Age'], srv21['Age'] = s19, s20, s21




# Countries
# we will deal only with the common countries which are 50 countries
countries_of_interest = pd.Series(list(set(srv20['Country']).intersection(set(srv21['Country'])).intersection(set(srv19['Cour
# Those was countries we will use for the extra section not for the main topic




# College Degree
# To make the three sets equals we must fix the 'professional doctoral' in srv21 to 'professional degree'
s = srv21['College_degree'].replace(['Professional doctorate'], 'Professional degree')
srv21['College_degree'] = s

# We want to divide degrees into three ranges: under bachelor, bachelor, master and after master
replace_me = ['I prefer not to answer', 'No formal education past high school',
              "Some college/university study without earning a bachelor's degree"]

# Less than Bachelor
s19, s20 = srv19['College_degree'].replace(replace_me, 'Less than bachelor'), srv20['College_degree'].replace(replace_me, 'Le
s21 = srv21['College_degree'].replace(replace_me, 'Less than bachelor')
srv19['College_degree'], srv20['College_degree'], srv21['College_degree'] = s19, s20, s21
```

```
# After Master
replace_me = ["Doctoral degree", 'Professional degree']
s19, s20 = srv19['College_degree'].replace(replace_me, 'After Master'), srv20['College_degree'].replace(replace_me, 'After Ma
s21 = srv21['College_degree'].replace(replace_me, 'After Master')
srv19['College_degree'], srv20['College_degree'], srv21['College_degree'] = s19, s20, s21




# Job title
# We will put any values but values of interest in the section 'Other'
values_of_interest = {'Student', 'Data Scientist', 'Data Analyst', 'Business Analyst', 'Software Engineer', 'Data Engineer',
s19 = srv19['Job_title'].replace([job for job in set(srv19['Job_title']) if job not in values_of_interest], 'Other')
s20 = srv20['Job_title'].replace([job for job in set(srv20['Job_title']) if job not in values_of_interest], 'Other')
s21 = srv21['Job_title'].replace([job for job in set(srv21['Job_title']) if job not in values_of_interest], 'Other')
srv19['Job_title'], srv20['Job_title'], srv21['Job_title'] = s19, s20, s21




# Experience
# We will be interested in some values only called 'values_of_interest'
# Years 1-3: Junior; Years 3-5: Mid-Level; Years 5-10: Senior; Years 10+: Principal/Architect;


values_of_interest = ['I have never written code', '< 1 years', '1-3 years', '3-5 years', '5-10 years', '10+ years']
# Range 1 and Range 2 is fixed itself we will start with Range 3
s19, s20 = srv19['Experience'].replace(['1-2 years'], '1-3 years'), srv20['Experience'].replace(['1-2 years'], '1-3 years')
srv19['Experience'], srv20['Experience'] = s19, s20
# srv21 fixed itself


# Range 4 and Range 5 are fixed let's go to Last Range
s19, s20 = srv19['Experience'].replace(['10-20 years', '20+ years'], '10+ years'), srv20['Experience'].replace(['10-20 years'
s21 = srv21['Experience'].replace(['10-20 years', '20+ years'], '10+ years')
srv19['Experience'], srv20['Experience'], srv21['Experience'] = s19, s20, s21
```

```
# Annual Compensation
# we will set the annual compensation into four ranges <1000, 1000-9,999, 10,000-99,999, 100,000<=

# Range 1
s19, s20 = srv19['Annual_compensation'].replace(['$0 (USD)', '$1-$99', '$100-$999'], '< $1000'), srv20['Annual_compensation']
srv19['Annual_compensation'], srv20['Annual_compensation'] = s19, s20
s21 = srv21['Annual_compensation'].replace(['$0-999'], '< $1000')
srv21['Annual_compensation'] = s21

# Range 2
s19, s20 = srv19['Annual_compensation'].replace(['$1000-$9,999'], '1,000-9,999'), srv20['Annual_compensation'].replace(['$100
srv19['Annual_compensation'], srv20['Annual_compensation'] = s19, s20
s21 = srv21['Annual_compensation'].replace(['1,000-1,999', '2,000-2,999', '3,000-3,999', '4,000-4,999', '5,000-7,499', '7,500
srv21['Annual_compensation'] = s21

# Range 3
s19, s20 = srv19['Annual_compensation'].replace(['$10,000-$99,999'], '10,000-99,999'), srv20['Annual_compensation'].replace([
srv19['Annual_compensation'], srv20['Annual_compensation'] = s19, s20
s21 = srv21['Annual_compensation'].replace(['10,000-14,999',
               '15,000-19,999', '20,000-24,999', '25,000-29,999', '30,000-39,999', '40,000-49,999', '50,000-59,999', '60,000-
               '70,000-79,999', '80,000-89,999', '90,000-99,999'], '10,000-99,999')
srv21['Annual_compensation'] = s21

# Range 4
s19, s20 = srv19['Annual_compensation'].replace(['> $100,000 ($USD)'], '> $100,000'), srv20['Annual_compensation'].replace(['
srv19['Annual_compensation'], srv20['Annual_compensation'] = s19, s20
s21 = srv21['Annual_compensation'].replace(['100,000-124,999', '125,000-149,999', '150,000-199,999', '200,000-249,999',
               '250,000-299,999', '300,000-499,999', '$500,000-999,999', '>$1,000,000'], '> $100,000')
srv21['Annual_compensation'] = s21


# So we finished fixing the values of each feature in all the wanted years
```

In [8]:
```
# Ordinal vars Agter updating in all years
age_order = ['18-21', '22-24', '25-29', '30-34', '35-39', '40+']

experience_order = ['< 1 years', '1-3 years', '3-5 years', '5-10 years', '10+ years']

money_order = ['< $1000', '1,000-9,999', '10,000-99,999', '> $100,000']

degree_order = ['Less than bachelor', "Bachelor's degree", "Master's degree", 'After Master']
```

In [9]:
```
# Merging the dataframes
srv19['year'] = 2019
srv20['year'] = 2020
srv21['year'] = 2021
srv_all = srv19.append(srv20).append(srv21)
```

In [11]:

```python
# For plotting any categorical variable (to see the distribution)
class one_cat:

    def __init__(self, data, col, title, xlabel=None, ylabel=None, width=400, height=400):
        self.data = data
        self.col = col
        self.title = title
        self.xlabel = xlabel
        self.ylabel = ylabel
        self.width = width
        self.height = height

# To plot a distribution for our catigorical variable if it has no specific order(nominal)
    def distribution_nominal(self , orientation='v',template='plotly'):

        '''
        Plotting your categorical variable if it is nominal (Has no order like Country names)
        orientation: To choose plottin the bars in a vertical representation or horizontal
        template: The template of your plotly graph
        '''

        # Figure itself

        # Vertical
        if orientation == 'v':

            fig = px.histogram(self.data, x=self.col,
                               width=600,
                               height=500,
                               histnorm='percent',
                               template=template
                               )
            # Some Updates
            fig.update_layout(title=self.title,
                        font_family="San Serif",
                        titlefont={'size': 20},
                        showlegend=True,
                        legend=dict(
                            orientation="v",
                            y=1.0,
                            yanchor="top",
                            x=1.0,
                            xanchor="right"
                        )
                       ).update_xaxes(categoryorder='total descending')
```

```python
        # Horizontal
        else:


            fig = px.histogram(self.data,
                   y=self.col,
                   orientation='h',
                   width=600,
                   height=350,
                   histnorm='percent',
                   template=template
                   )

            # Some Updates
            fig.update_layout(title=self.title,
                              font_family="San Serif",
                              titlefont={'size': 20},
                              showlegend=True,
                              legend=dict(
                                  orientation="v",
                                  y=1.0,
                                  yanchor="top",
                                  x=1.0,
                                  xanchor="right"
                              )
                             ).update_yaxes(categoryorder='total ascending')




        # Traces Updates
        fig.update_traces(marker_color=None, marker_line_color='white',
                          marker_line_width=1.5, opacity=0.99)
        fig.show()



    # To plot a distribution for our catigorical variable if it has specific order(ordinal)
    def distribution_ordinal(self, the_specific_order, orientation='v', template='plotly'):
```

```
        '''
        Plotting your categorical variable if it is ordianl (Has order eg: small, medium and large)
        the_specific_order: The right order of your categorical variable
        orientation: To choose plotting the bars in a vertical representation or horizontal
        template: The template of your plotly graph
        '''

        # Figure itself
        if orientation == 'v':

            # Vertical
            fig = px.histogram(self.data, x=self.col,
                               width=self.width, height=self.height,
                               histnorm='percent',
                               template=template,
                               category_orders={
                                   self.col: the_specific_order
                               }
                               )
            # Some Updates
            fig.update_layout(title=self.title,
                              font_family="San Serif",
                              titlefont={'size': 20},
                              showlegend=True,
                              legend=dict(
                                  orientation="v",
                                  y=1.0,
                                  yanchor="top",
                                  x=1.0,
                                  xanchor="right"
                              )
                             )


        else:

            # Horizontal
            fig = px.histogram(self.data,
                   y=self.col,
                   orientation='h',
                   width=self.width,height=self.height,
                   histnorm='percent',
                   template=template,
                   category_orders={
                               self.col: the_specific_order
                    }
                   )
```

```
            # Some Updates
            fig.update_layout(title=self.title,
                              font_family="San Serif",
                              titlefont={'size': 20},
                              showlegend=True,
                              legend=dict(
                                  orientation="v",
                                  y=1.0,
                                  yanchor="top",
                                  x=1.0,
                                  xanchor="right"
                              )
                             )



        # Traces Updates
        fig.update_traces(marker_color=None, marker_line_color='white',
                          marker_line_width=1.5, opacity=0.99)
        fig.show()




    def pie(self):

        '''
        Plotting Pie chart with the default colors
        '''

        fig = px.pie(self.data, self.col,
                     title=self.title,
                     hole=0.6)
        fig.update_traces(textposition='inside', textinfo='percent+label')
        fig.update_layout(uniformtext_minsize=10, uniformtext_mode='hide')
        fig.show()
```

```python
# As blocks
def waffle(self, title):
    '''
    Plotting your categorical variable in blocks without specific shapes
    title: The title of your chart
    '''

    df = self.data
    var = self.col

    # Take as input a dataframe and variable, and return a Pandas series with
    # approximate percentage values for filling out a waffle plot.

    # compute base quotas
    percentages = 100 * df[var].value_counts() / df.shape[0]
    counts = np.floor(percentages).astype(int) # integer part = minimum quota
    decimal = (percentages - counts).sort_values(ascending = False)

    # add in additional counts to reach 100
    rem = 100 - counts.sum()
    for cat in decimal.index[:rem]:
        counts[cat] += 1


    waffle_counts = counts
    prev_count = 0
    # for each category,
    for cat in range(waffle_counts.shape[0]):
        # get the block indices
        blocks = np.arange(prev_count, prev_count + waffle_counts[cat])
        # and put a block at each index's location
        x = blocks % 10 # use mod operation to get ones digit
        y = blocks // 10 # use floor division to get tens digit
        plt.bar(x = y, height = 0.8, width = 0.8, bottom = x)
        prev_count += waffle_counts[cat]

    # aesthetic wrangling
    plt.legend(waffle_counts.index, bbox_to_anchor = (1, 0.5), loc = 6)
    plt.axis('off')
    plt.axis('square')
    plt.title(title, size=20)
    plt.show()
```

```python
def waffle_with_shape(self, order, shape='child'):

    '''
    Plotting your categorical variable in blocks with specific shapes
    title: The title of your chart
    shape: The shape name
    '''
    series = self.data[self.col].dropna().value_counts()
    series = 100*series // series.sum()
    d = {Range:value for Range, value in zip(series.index, series.values)}
    data = {Range:d[Range] if Range in d else 0 for Range in order}
    fig = plt.figure(
        FigureClass=Waffle,
        rows=8,
        values=data,
        legend={'loc': 'upper left', 'bbox_to_anchor': (1, 1)},
        icons=shape, icon_size=20,
        icon_legend=True,
        title={'label': self.title, 'loc': 'left', 'size':20}
    )
    fig.show()




def updated_pie_chart(self, title, colors):
    '''
    Plotting your categorical variable with Pie chart having the variable name in it's center and with your specific colo
    title: The title of the graph
    colors: List of colors to each value of variable's values
    '''
    fig = go.Figure()
    fig.add_trace(
        go.Pie(
            labels=self.data[self.col],
            values=None,
            hole=.4,
            title=title,
            titlefont={'color':None, 'size':22},
            )
        )
    fig.update_traces(
        hoverinfo='label+value',
        textinfo='label+percent',
        textfont_size=14,
        marker=dict(
            colors=colors,
            line=dict(color='#000000',

                      width=2)
            )
        )
```

```
        # Updating the layout
        fig.update_layout(title=self.title,
                          font_family="San Serif",
                          bargap=0.2,
                          titlefont={'size': 24},
                          legend=dict(
                          orientation="v", y=1, yanchor="top", x=1.25, xanchor="right")
                          )
        fig.show()
```

In [12]:

```
# Two Categorical Values (color through each value)
class two_cat:

    def __init__(self, data, col, hue, title, width, height, xlabel=None, ylabel=None):
        self.data = data
        self.col = col
        self.hue = hue
        self.title = title
        self.width = width
        self.height = height
        self.xlabel = xlabel
        self.ylabel = ylabel

    def bar_group_nominal(self, orientation='v', template='plotly'):
        '''
        Plotting grouped bar graph if you have your variables nominal
        orientation: To choose plotting the bars in a vertical representation or horizontal
        template: The template of your plotly graph
        '''

        # Vertical
        if orientation == 'v':

            fig = px.histogram(self.data, x=self.col, y=None, color=self.hue,
                               width=self.width,height=self.height,
                               histnorm='percent', # By percentage not by counts
                               template=template)

            fig.update_layout(title=self.title,
                              font_family="San Serif",
                              bargap=0.2,
                              barmode='group',
                              titlefont={'size': 24},
                              legend=dict(
                              orientation="v", y=1, yanchor="top", x=1.25, xanchor="right")
                              ).update_xaxes(categoryorder='total descending')

        # Horizontal
        elif orientation == 'h':

            fig = px.histogram(self.data, y=self.col, x=None, color=self.hue,
                                orientation='h',
                                width=self.width, height=self.height,
                                histnorm='percent', # By percentage not by counts
                                template=template
                               )

            fig.update_layout(title=self.title,
                              font_family="San Serif",
                              bargap=0.2,
                              barmode='group',
                              titlefont={'size': 24},
                              legend=dict(
                              orientation="v", y=1, yanchor="top", x=1.25, xanchor="right")
                              ).update_yaxes(categoryorder='total ascending')

        fig.show()
```

```
def bar_group_ordinal(self, order, orientation='v', template='plotly'):
    '''
    Plotting grouped bar graph if you have your variables ordinal
    orientation: To choose plotting the bars in a vertical representation or horizontal
    template: The template of your plotly graph
    '''

    # The graph itself

    # Vertical
    if orientation == 'v':

        fig = px.histogram(self.data, x=self.col, y=None, color=self.hue,
                           width=self.width,height=self.height,
                           histnorm='percent', # By percentage not by counts
                           category_orders={
                               self.col:order
                           },
                           template=template)

        # Updating the layout
        fig.update_layout(title=self.title,
                          font_family="San Serif",
                          bargap=0.2,
                          barmode='group',
                          titlefont={'size': 24},
                          legend=dict(
                          orientation="v", y=1, yanchor="top", x=1.25, xanchor="right")
                          )

    # Horizontal
    elif orientation == 'h':

        fig = px.histogram(self.data, y=self.col, x=None, color=self.hue,
                           orientation='h',
                           width=self.width, height=self.height,
                           histnorm='percent', # By percentage not by counts
                           category_orders={
                               self.col:order
                           },
                           template=template
                         )

        # Updating the layout
        fig.update_layout(title=self.title,
                          font_family="San Serif",
                          bargap=0.2,
                          barmode='group',
                          titlefont={'size': 24},
                          legend=dict(
                          orientation="v", y=1, yanchor="top", x=1.25, xanchor="right")
                          )

    fig.show()
```

```python
# It's source will be in the appendix section Thanks "GEORGII VYSHNIA"
def catscatter(self, colx_order, coly_order, color=['grey','black'], ratio=180, font='Helvetica'):
    '''
    It's name abbreviation for Categorical scatter, so from it's name it's very likely to scatter that you know but for c
    The size of each point varies by the percentage of the existence of each value in variable1 in variable2
    colx_order: the order of variable1
    coly_order: the order of variable2
    color: the colors of them (Be careful it's not easy to be determined)
    ratio: the size of the points (overall not specific point)
    font: the font of your words
    '''
    agg_data = self.data.groupby([self.col, self.hue]).size().reset_index(name='respondent_count')
    # this will generate the count, but we want the ratio of each range in var1 in each range in var2 so we will use the
    agg_data['ratio'] = 0

    for var1_val in colx_order:
        summ = agg_data[agg_data[self.col] == var1_val]['respondent_count'].sum()

        for var2_val in coly_order:

            row = agg_data[ (agg_data[self.col] == var1_val) & (agg_data[self.hue] == var2_val) ]
            value = row['respondent_count'].values[0]
            agg_data.loc[(agg_data[self.col] == var1_val) & (agg_data[self.hue] == var2_val), 'ratio'] = round(value / su



    # this will prevent manifesting a little bug of catscatter
    # casting age and gender to int64 as a result of the catscatter plotting below
    agg_data_copy = agg_data.copy()
    colx = self.col
    coly = self.hue
    df = agg_data_copy
    cols = 'ratio'
    # Create a dict to encode the categeories into numbers (sorted)
    colx_codes=dict(zip(colx_order,range(len(df[colx].unique()))))
    coly_codes=dict(zip(coly_order[::-1],range(len(df[coly].unique()))))

    # Apply the encoding
    df[colx]=df[colx].apply(lambda x: colx_codes[x])
    df[coly]=df[coly].apply(lambda x: coly_codes[x])


    # Prepare the aspect of the plot
    plt.rcParams['xtick.bottom'] = plt.rcParams['xtick.labelbottom'] = False
    plt.rcParams['xtick.top'] = plt.rcParams['xtick.labeltop'] = True
    plt.rcParams['font.sans-serif']=font
    plt.rcParams['xtick.color']=color[-1]
    plt.rcParams['ytick.color']=color[-1]
    plt.box(False)


    # Plot all the lines for the background
    for num in range(len(coly_codes)):
        plt.hlines(num,-1,len(colx_codes)+1,linestyle='dashed',linewidth=1,color=color[num%2],alpha=0.5)
    for num in range(len(colx_codes)):
        plt.vlines(num,-1,len(coly_codes)+1,linestyle='dashed',linewidth=1,color=color[num%2],alpha=0.5)
```

```
        # Plot the scatter plot with the numbers
        plt.scatter(df[colx],
                    df[coly],
                    s=df[cols]*ratio,
                    zorder=2,
                    color=color[-1])

        # To annotate the scatter plot
        agg_data_copy = agg_data.copy()
        xmap = {order:val for order, val in zip(colx_order, range(len(colx_order)) ) }
        ymap = {order:val for order, val in zip(coly_order, range(len(coly_order))[::-1] ) }
        for var1_val in colx_order:
            x_place = xmap[var1_val]

            for var2_val in coly_order:
                y_place = ymap[var2_val]
                percentage = agg_data_copy.loc[ (agg_data_copy[colx] == var1_val) & (agg_data_copy[coly] == var2_val)  ,'rati
                plt.annotate(str(percentage)+'%', (x_place+.1, y_place+.1), size=30)

        # Change the ticks numbers to categories and limit them
        plt.xticks(ticks=list(colx_codes.values()),labels=colx_codes.keys(),rotation=90)
        plt.yticks(ticks=list(coly_codes.values()),labels=coly_codes.keys())
        plt.xlim(xmin=-1,xmax=len(colx_codes))
        plt.ylim(ymin=-1,ymax=len(coly_codes))

        # Some updates
        plt.xticks(fontsize=30)
        plt.yticks(fontsize=30)
        if self.xlabel:
            plt.xlabel(self.xlabel, size=50, color='gray')

        if self.ylabel:
            plt.ylabel(self.ylabel, size=50, color='gray')


        plt.savefig('cat.png')
        plt.show()




    def stacked_plot(self, colx_order, coly_order, template='plotly', colors=None):

        '''
        Plotting stacked plot for your categorical variables if they are nominal
        colx_order: The order of values of your first variable
        coly_order : The order of values of your second variable
        '''

        agg_data = self.data.groupby([self.col, self.hue]).size().reset_index(name='respondent_count')
        # This will generate the count, but we want the ratio of each range in var1 in each range in var2 so we will use the
        agg_data['ratio'] = 0

        for var1_val in colx_order:
            summ = agg_data[agg_data[self.col] == var1_val]['respondent_count'].sum()

            for var2_val in coly_order:
                row = agg_data[ (agg_data[self.col] == var1_val) & (agg_data[self.hue] == var2_val) ] # Data of interest
```

```
        # Plotting them as bar graph, but with the color argument
        fig = px.bar(
            agg_data,
            template=template,
            x=self.col,
            y='ratio',
            color=self.hue,
            title=self.title,
            height=400,
            category_orders={
                self.col: colx_order,
                self.hue: coly_order
            },
            color_discrete_map={
                value: colors[i] for value, i in zip(coly_order, range(len(colors)))
            })
        fig.show()




    def bar_group_ordinal_own_colors(self, colx_order, hue_order, colors, arrow_positions, template='plotly'):

        '''
        The grouped by graph is doing the same as the normal grouped bar graph but with your specific arrow and colors
        colx_order: The order of values of your first variable
        hue_order: The order of values of your second variable
        colors: List of colors having a color for each value in variable2 or the hue
        arrow_positions: List having four values to specify the start point (x0, y0) and the end point (x1, y1) for your arrc
        '''
        # Reverse the order for better representation
        colx_order = colx_order[::-1]

        fig = go.Figure()
        fig.update_layout(
            autosize=False,
            width=self.width,
            height=self.height,)

        # Plotting first var in each value of second var
        for val, color in zip(hue_order, colors):

            # To select values of interest and order them
            data_of_interest = self.data[self.data['year'] == val]
            y_values = data_of_interest[self.col].value_counts()
            y_values = y_values / y_values.sum()*100
            mapping = {y_values_index:y_values_value for y_values_index, y_values_value in zip(y_values.index, y_values.value
            y = [mapping[x_val] for x_val in colx_order]


            # Plot it
            fig.add_trace(go.Bar(
                y=colx_order,
                x=y,
                orientation='h',
                name=val,
                marker_color=color
            ))
```

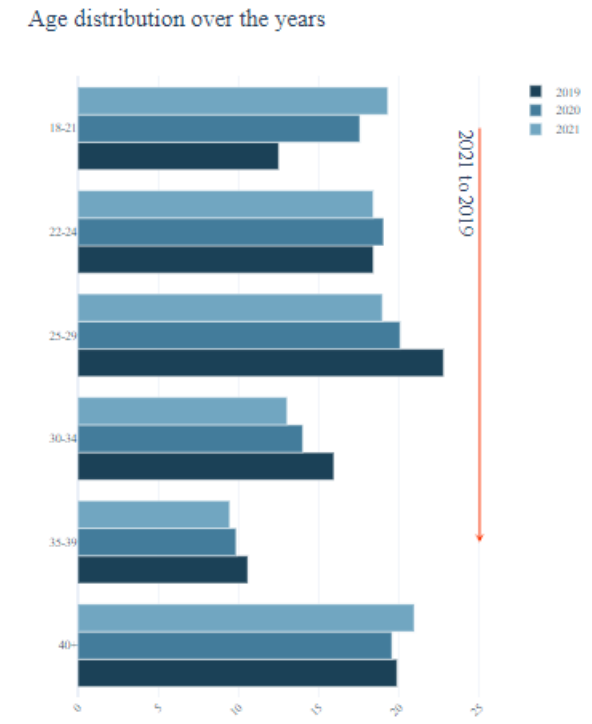
```
        # Here we modify the tickangle of the xaxis, resulting in rotated labels.
        fig.update_layout(barmode='group', xaxis_tickangle=-45)


        # Those arrows are for this case you can change them for your graph
        # Arrow
        xtail, ytail, xhead, yhead = arrow_positions
        arrow = go.layout.Annotation(dict(
                        x= xhead, y= yhead, # Head of the arrow position
                        xref="x", yref="y",

                        # Text
                        text="2021 to 2019",
                        textangle=90,
                        font_size=20,
                        xanchor='left',
                        yanchor='top',

                        showarrow=True,
                        axref = "x", ayref='y',
                        ax= xtail, ay= ytail, # Tail of the arrow position
                        arrowhead = 3,
                        arrowwidth=1.5,
                        arrowcolor='rgb(255,51,0)')
                    )

        fig.update_layout(
        annotations= [arrow],)


        # Updating the layout
        fig.update_layout(title=self.title, template=template,
                          font_family="San Serif",
                          bargap=0.2,
                          titlefont={'size': 24},
                          legend=dict(
                          orientation="v", y=1, yanchor="top", x=1.25, xanchor="right")
                          )

        fig.show()
```

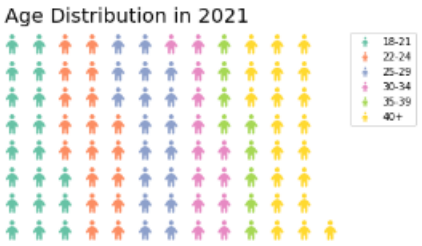
In [13]:

```
srv_all = srv_all.query('Experience != "I have never written code"')
srv21 = srv21.query('Experience != "I have never written code"')
```

In [14]:

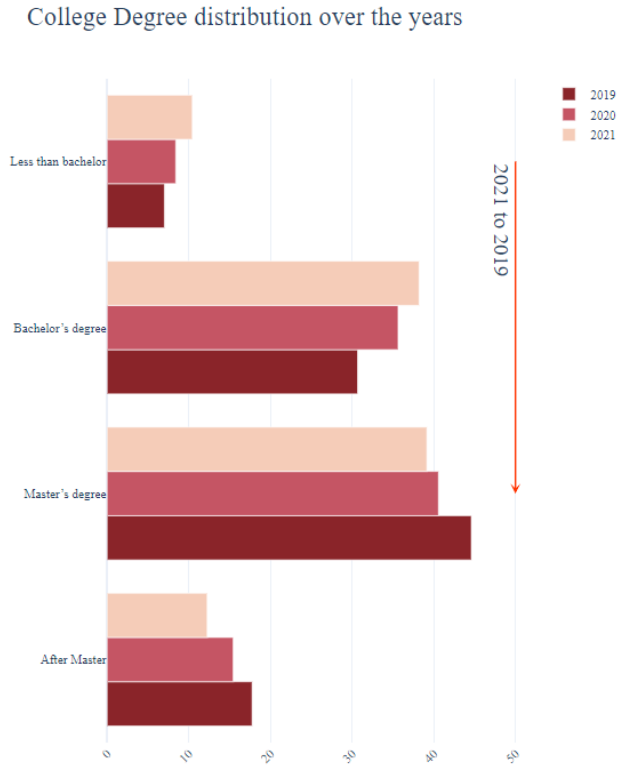
```
age = two_cat(srv_all, 'Age', 'year', 'Age distribution over the years', 600, 800)
age.bar_group_ordinal_own_colors(age_order, [2019, 2020, 2021], ['#1B4157', '#437C9B', '#71A6C1'], [25, 5, 25, 1], template=
age = one_cat(srv21, 'Age', 'Age Distribution in 2021')
age.waffle_with_shape(age_order)
```

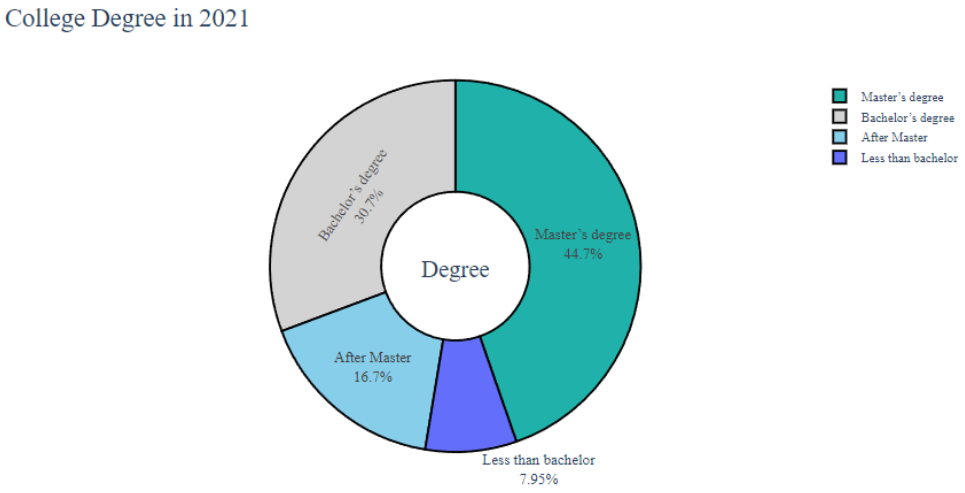




In [15]:

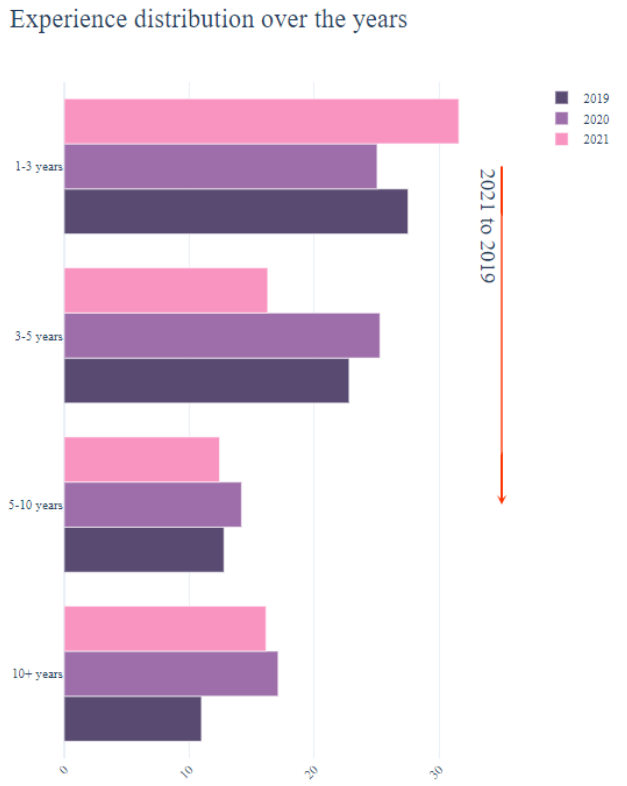
```
degree = two_cat(srv_all, 'College_degree', 'year', 'College Degree distribution over the years', 600, 800)
degree.bar_group_ordinal_own_colors(degree_order, [2019, 2020, 2021], ['#8A2429', '#C55564', '#F5CCB8'], [50, 3, 50, 1], temp
degree = one_cat(srv21.dropna(), 'College_degree', 'College Degree in 2021')
degree.updated_pie_chart('Degree', ['lightgray', 'lightseagreen', 'lightblue', 'skyblue'])
```

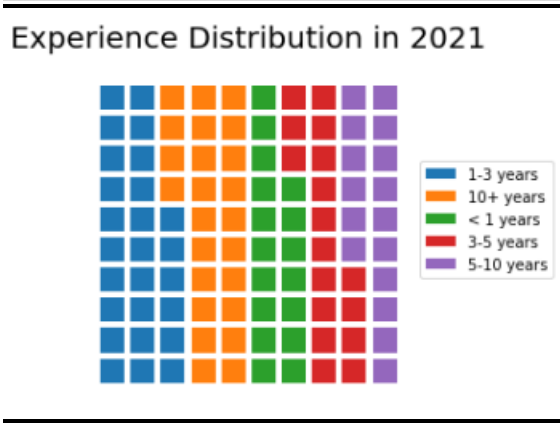




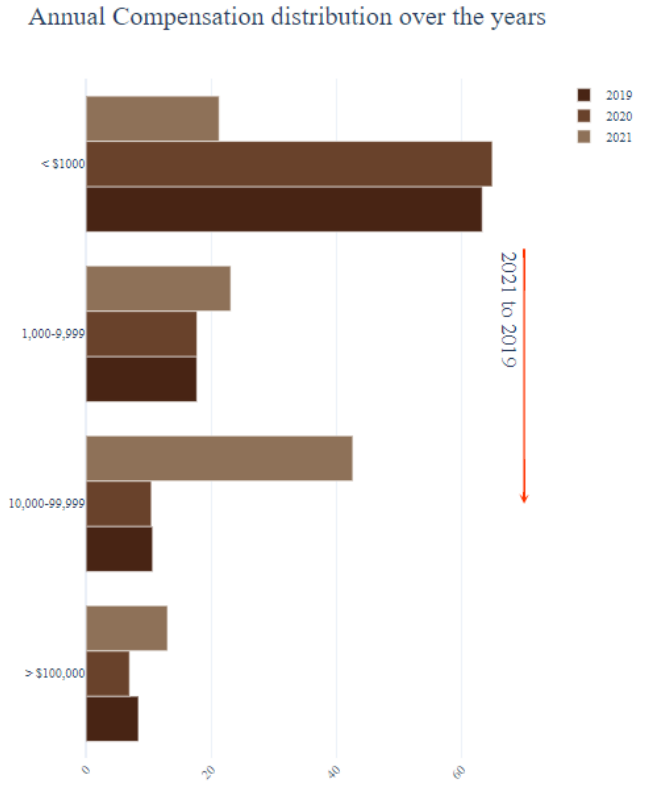
In [16]:

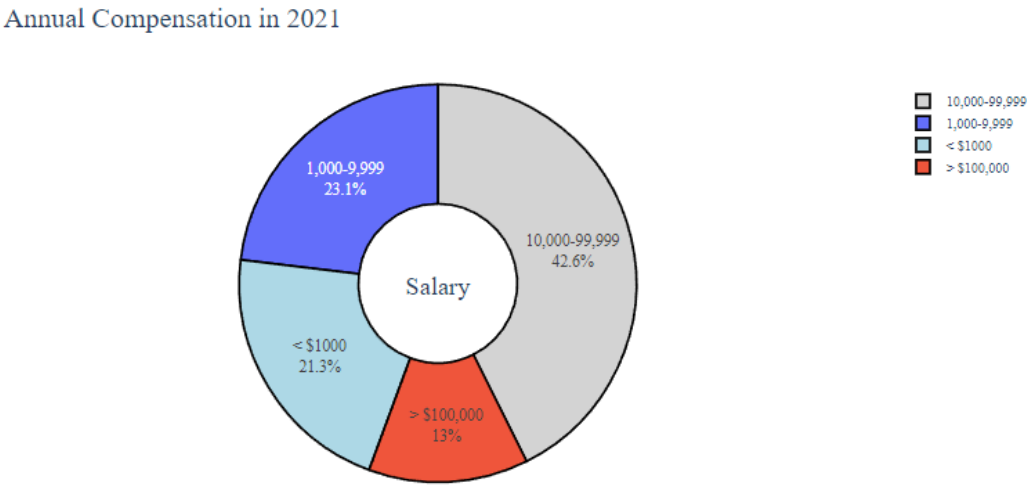
```
ex = two_cat(srv_all, 'Experience', 'year', 'Experience distribution over the years', 600, 800)
ex.bar_group_ordinal_own_colors(experience_order[1:], [2019, 2020, 2021],
                                ['#594A72', '#9E6EAA', '#F994C1'], [35, 3, 35, 1], template='plotly_white') # From one to ass
ex = one_cat(srv21.dropna(), 'Experience', 'Experience in 2021')
ex.waffle('Experience Distribution in 2021')
```





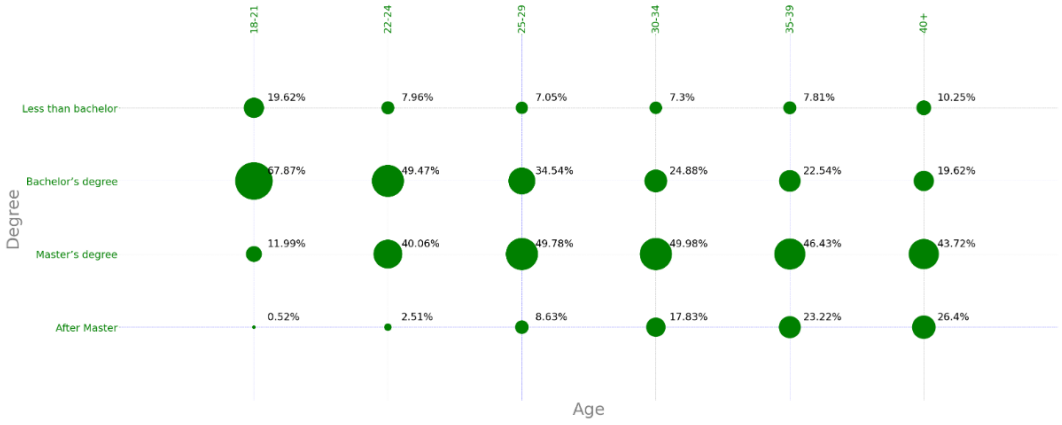
In [17]:
```
salary = two_cat(srv_all, 'Annual_compensation', 'year', 'Annual Compensation distribution over the years', 600, 800)
salary.bar_group_ordinal_own_colors(money_order, [2019, 2020, 2021], ['#482414', '#69422B', '#8E7158'], [70, 2.5, 70, 1], tem
salary = one_cat(srv21.dropna(), 'Annual_compensation', 'Annual Compensation in 2021')
salary.updated_pie_chart('Salary', ['lightgray', 'lightseagreen', 'lightblue', 'skyblue'])
```





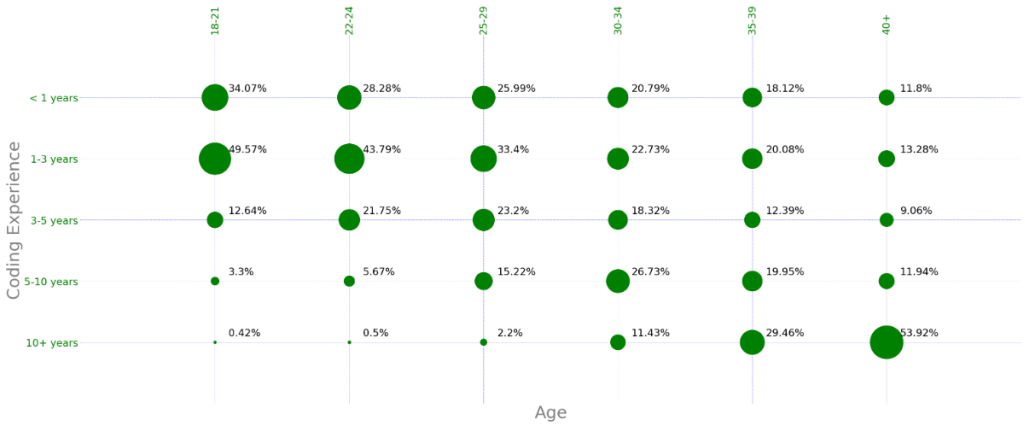
In [18]:
```
#How age affects the college degree?
colors=['blue', 'grey', 'green']
# create the plot
plt.figure(figsize=(50,20))
age_degree = two_cat(srv21, 'Age', 'College_degree', 'Age Vs Degree', 400, 400, xlabel='Age', ylabel='Degree')
age_degree.catscatter(age_order, degree_order, color=colors, ratio=180)
plt.savefig('catPlot.png')
```

```
findfont: Font family ['sans-serif'] not found. Falling back to DejaVu Sans.
findfont: Generic family 'sans-serif' not found because none of the following families were found: Helvetica
findfont: Font family ['sans-serif'] not found. Falling back to DejaVu Sans.
findfont: Generic family 'sans-serif' not found because none of the following families were found: Helvetica
```



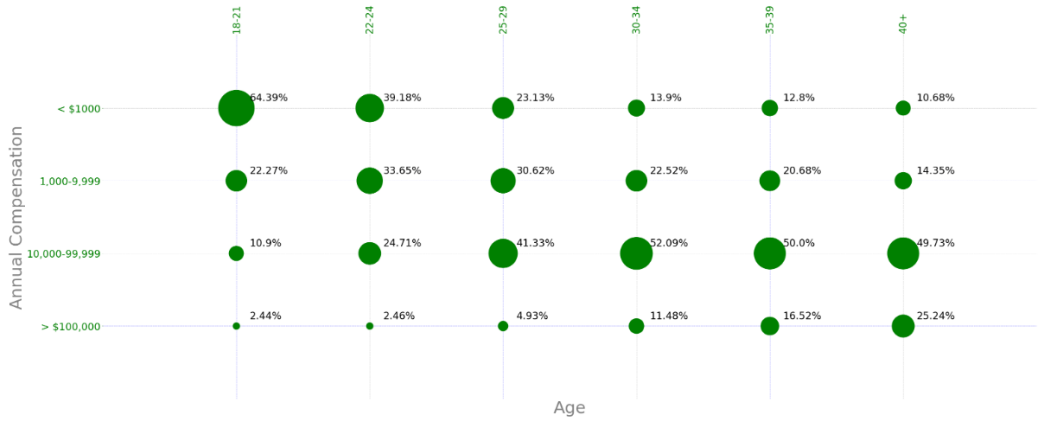

```
<Figure size 432x288 with 0 Axes>
```

In [19]:
```
#How age affects the Coding Experience?
colors=['blue', 'grey', 'green']
# create the plot
plt.figure(figsize=(50,20))
age_degree = two_cat(srv21, 'Age', 'Experience', 'Age Vs Degree', 400, 400, xlabel='Age', ylabel='Coding Experience')
age_degree.catscatter(age_order, experience_order, color=colors, ratio=180)
```



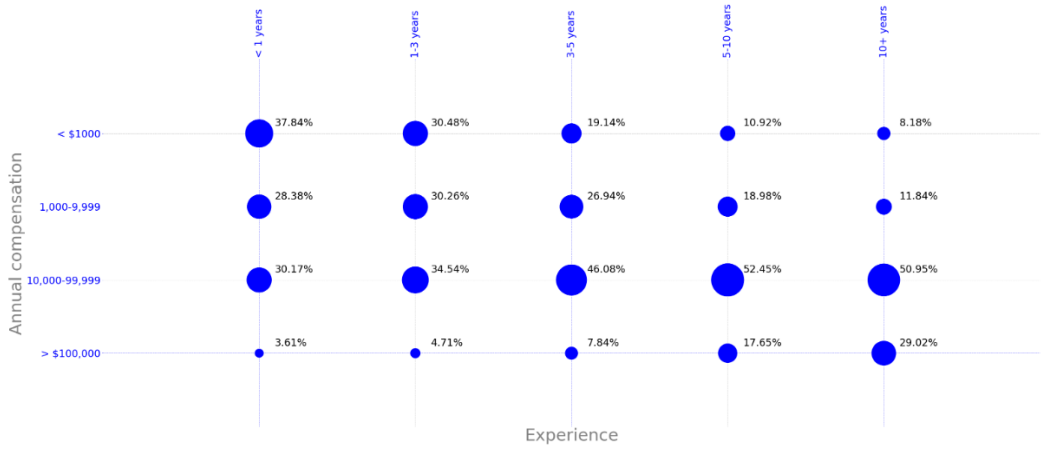
In [20]:

```
#How age affects the Annual Compensation?
colors=['blue', 'grey', 'green']
# create the plot
plt.figure(figsize=(50,20))
age_degree = two_cat(srv21, 'Age', 'Annual_compensation', 'Age Vs Degree', 400, 400, xlabel='Age', ylabel='Annual Compensatic
age_degree.catscatter(age_order, money_order, color=colors, ratio=180)
```



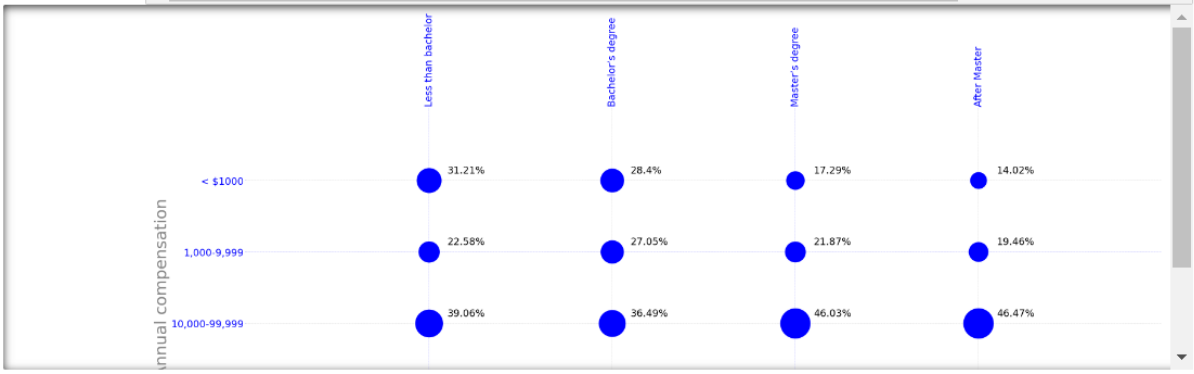
In [24]:

```
#Does coding experience have a huge effect on Annual Compensation?
colors=['blue', 'grey', 'blue']
# create the plot
plt.figure(figsize=(50,20))
age_degree = two_cat(srv21, 'Experience', 'Annual_compensation', 'Experience Vs Annual_compensation', 400, 400, xlabel='Exper
age_degree.catscatter(experience_order, money_order, color=colors)
```

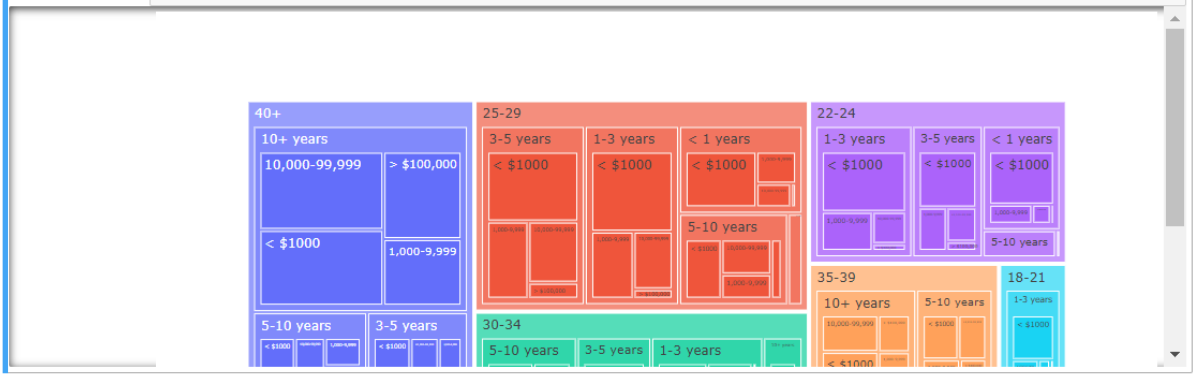


In [27]:

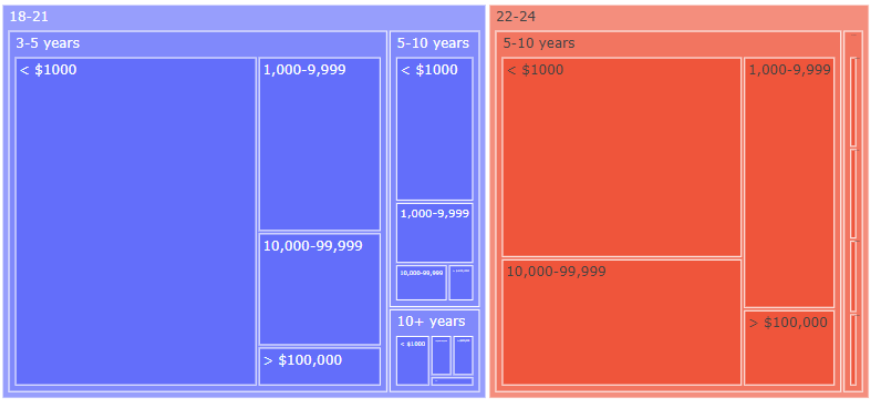
```
#Does college degree have a huge effect on Annual Compensation?
colors=['blue', 'grey', 'blue']
# create the plot
plt.figure(figsize=(50,20))
age_degree = two_cat(srv21, 'College_degree', 'Annual_compensation', 'Degree Vs Annual_compensation', 400, 400, xlabel='Degre
age_degree.catscatter(degree_order, money_order, color=colors)
```



In [22]:
```
#Did young people benefit financially from what they worked on in their childhood ?
dimension_cats = ['Age', 'Experience', 'Annual_compensation']
agg_data = srv_all.groupby(dimension_cats).size().reset_index(name='count')
fig = px.treemap(agg_data, path=dimension_cats, values='count')
fig.show()
```



In [23]:
```
#Hard-working Youth
cond1 = (srv_all['Experience'].isin(pd.Series(['5-10 years', '10+ years']))) & (srv_all['Age'] == '22-24')
cond2 = (srv_all['Experience'].isin(pd.Series(['3-5 years', '5-10 years', '10+ years']))) & (srv_all['Age'] == '18-21')

youth_worked_hard = srv_all[  (cond1)  |  (cond2)  ]
dimension_cats = ['Age', 'Experience', 'Annual_compensation']
agg_data = youth_worked_hard.groupby(dimension_cats).size().reset_index(name='count')
fig = px.treemap(agg_data, path=dimension_cats, values='count')
fig.show()
```



Thankyou