For the final project, I went ahead and looked at the relationship between NBA player's points per game and their salary by analyzing a directed graph with players as nodes and a difference of at most 2 PPG (points per game) as edges between the players. The first thing I did was gather two datasets: a data set with all NBA players and their stats and a data set with all NBA players and their salaries. Both of these datasets were imported from https://www.basketball-reference.com/ .

	Rk	Player	Pos	Age	Tm	G	GS	MP	FG	FGA	...	FT%	ORB	DRB	TRB	AST	STL	BLK	TOV	PF	PTS
0	1	Precious Achiuwa	C	22	TOR	73	28	23.6	3.6	8.3	...	.595	2.0	4.5	6.5	1.1	0.5	0.6	1.2	2.1	9.1
1	2	Steven Adams	C	28	MEM	76	75	26.3	2.8	5.1	...	.543	4.6	5.4	10.0	3.4	0.9	0.8	1.5	2.0	6.9
2	3	Bam Adebayo	C	24	MIA	56	56	32.6	7.3	13.0	...	.753	2.4	7.6	10.1	3.4	1.4	0.8	2.6	3.1	19.1
3	4	Santi Aldama	PF	21	MEM	32	0	11.3	1.7	4.1	...	.625	1.0	1.7	2.7	0.7	0.2	0.3	0.5	1.1	4.1
4	5	LaMarcus Aldridge	C	36	BRK	47	12	22.3	5.4	9.7	...	.873	1.6	3.9	5.5	0.9	0.3	1.0	0.9	1.7	12.9
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
837	601	Thaddeus Young	PF	33	TOR	26	0	18.3	2.6	5.5	...	.481	1.5	2.9	4.4	1.7	1.2	0.4	0.8	1.7	6.3
838	602	Trae Young	PG	23	ATL	76	76	34.9	9.4	20.3	...	.904	0.7	3.1	3.7	9.7	0.9	0.1	4.0	1.7	28.4
839	603	Omer Yurtseven	C	23	MIA	56	12	12.6	2.3	4.4	...	.623	1.5	3.7	5.3	0.9	0.3	0.4	0.7	1.5	5.3
840	604	Cody Zeller	C	29	POR	27	0	13.1	1.9	3.3	...	.776	1.9	2.8	4.6	0.8	0.3	0.2	0.7	2.1	5.2
841	605	Ivica Zubac	C	24	LAC	76	76	24.4	4.1	6.5	...	.727	2.9	5.6	8.5	1.6	0.5	1.0	1.5	2.7	10.3

842 rows × 30 columns

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Salary	Salary.1	Salary.2	Salary.3	Salary.4	Salary.5	Unnamed: 9	Unnamed: 10
0	Rk	Player	Tm	2021-22	2022-23	2023-24	2024-25	2025-26	2026-27	Signed Using	Guaranteed
1	1	Stephen Curry	GSW	$45,780,966	$48,070,014	$51,915,615	$55,761,216	$59,606,817	NaN	Bird Rights	$261,134,628
2	2	John Wall	HOU	$44,310,840	$47,366,760	NaN	NaN	NaN	NaN	Bird Rights	$44,310,840
3	3	Russell Westbrook	LAL	$44,211,146	$47,063,478	NaN	NaN	NaN	NaN	Bird Rights	$44,211,146
4	4	James Harden	PHI	$43,848,000	$46,872,000	NaN	NaN	NaN	NaN	Bird Rights	$43,848,000
...	...	...	...	...	...	...	...	...	...	...	...
673	613	Moses Brown	CLE	$1,720,779	NaN	NaN	NaN	NaN	NaN	Minimum Salary	$211,046
674	614	Juwan Morgan	BOS	$19,186	$1,815,677	NaN	NaN	NaN	NaN	Minimum Salary	$115,116
675	615	Trent Forrest	UTA	$8,558	NaN	NaN	NaN	NaN	NaN	Minimum Salary	$8,558
676	616	Ish Wainright	PHO	$8,558	NaN	NaN	NaN	NaN	NaN	Minimum Salary	$8,558
677	617	Kessler Edwards	BRK	$5,318	$1,563,518	NaN	NaN	NaN	NaN	Minimum Salary	$5,318

The next step was to clean the data. Because basketball reference has a header every 20 rows, I went ahead and removed these headers for both tables.

```
df_2022[df_2022.Age == 'Age']
```

```
df = df_2022.drop(df_2022[df_2022.Age == 'Age'].index)
df
```

```
df.drop_duplicates(subset ="Player", keep = 'last', inplace = True)
df
```

I then turned each of the tables to csv files.

```
df.to_csv('2022players.csv')
```

I loaded up both tables and combined the two tables, taking only the rows that were in both tables, meaning that it would only take players from both tables. For the columns, I only took the "Player" and "PTS" columns from the first table and the "Salary" column from the second table. There were a couple rows with "nan" as their salary, so I also removed those rows and again turned the table into a csv file named "combtable"

Player	PTS	Salary
Aaron Gordon	15	$16,409,091
Aaron Holiday	6.8	$3,980,551
Aaron Nesmith	3.8	$3,631,200
Aaron Wiggins	8.3	$1,000,000
Abdel Nader	2.4	$2,000,000
Al Horford	10.2	$27,000,000
Alec Burks	11.7	$9,536,000
Aleksej Pokusevski	7.6	$3,113,160
Alex Caruso	7.4	$8,604,651
Alex Len	6	$3,731,707

Since nodes would have an edge if their PPG was within 2 points of another player, I coded this and put them into one table, with node being the source node and node2 being the target node and turned it into a csv file named "nbaplayer_node".

```python
edges = make_array()
edges2 = make_array()
for x in range(comb_table.num_rows):
    for y in range(comb_table.num_rows):
        if abs(comb_table[1][x] - comb_table[1][y]) <= 2 and comb_table[1][x] != comb_table[1][y]:
            edges = np.append(edges, comb_table[0][x])
            edges2 = np.append(edges2, comb_table[0][y])
edges
edges2
```

```
array(['Bobby Portis', 'Bogdan Bogdanović', 'Caris LeVert', ...,
       'Willy Hernangómez', 'Zach Collins', 'Zeke Nnaji'], dtype='<U32')
```

```python
node_graph = Table().with_columns("node", edges, "node2", edges2)
node_graph
```

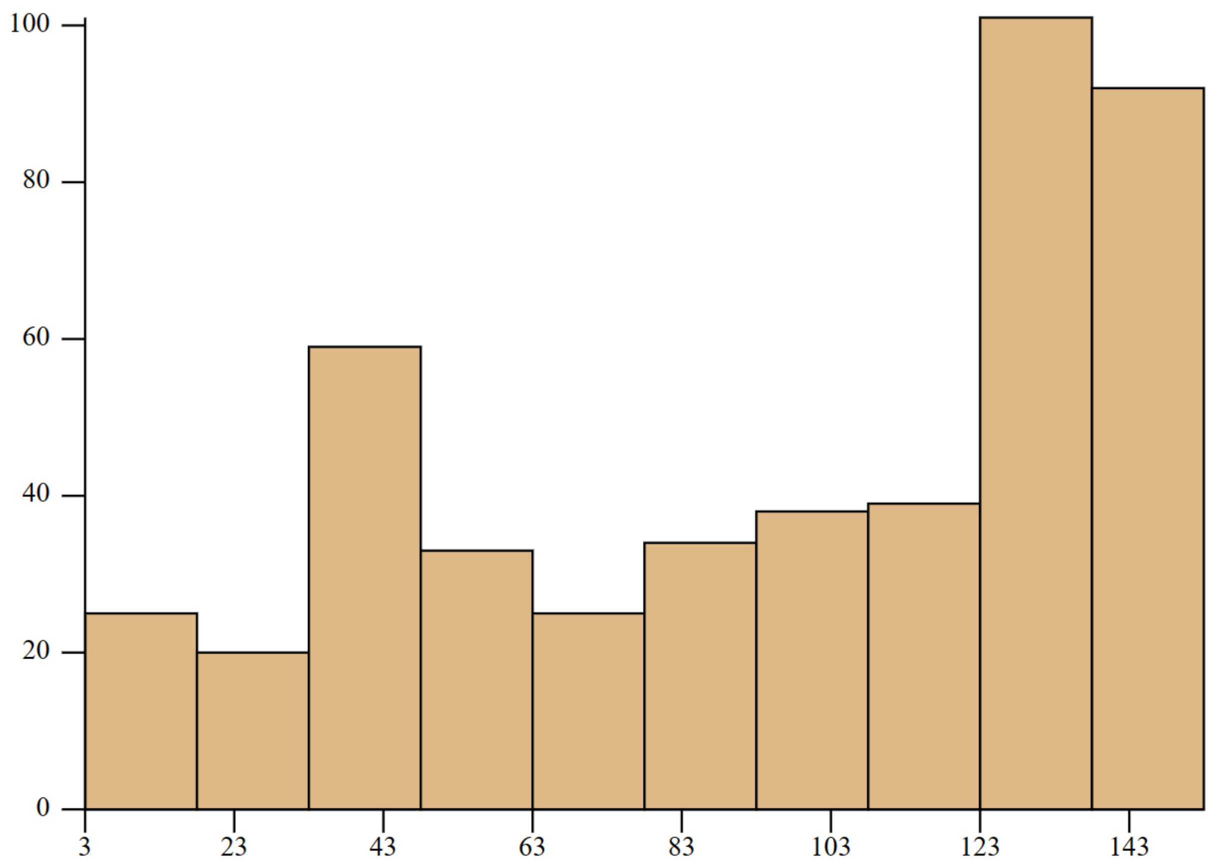
node	node2
Aaron Gordon	Bobby Portis
Aaron Gordon	Bogdan Bogdanović
Aaron Gordon	Caris LeVert
Aaron Gordon	Carmelo Anthony
Aaron Gordon	Chris Duarte
Aaron Gordon	Chris Paul
Aaron Gordon	Cole Anthony
Aaron Gordon	Collin Sexton
Aaron Gordon	De'Andre Hunter
Aaron Gordon	Drew Eubanks

Once I had the csv file that I wanted, I created a directed graph on rust by using a hash map. The keys for the hash map were the vertices of the graph, which were the players.

```rust
struct Graph {
    outedges: AdjacencyLists,
}

// functions that implies graph that helps build the network edges in HashTable
impl Graph {
    fn add_directed_edges(&mut self, edges:&ListOfEdges) {
        for (u,v) in edges {
            match self.outedges.get_mut(u){
                Some(adj) => adj.push(v.to_string()),
                None => {let _ = self.outedges.insert(u.to_string(), vec![v.to_string()]);}
            }
        }
    }
    //creates graph
    fn new_empty() -> Graph{
        Graph{
            outedges: HashMap::new()
        }
    }
}
```

```rust
//function to read in csv file and put into hashmap
fn read_edges() -> Graph{
    let input = fs::read_to_string("nbaplayer_node.csv").expect("Reading the file failed");
    let mut lines = input.trim().split("\n");
    let n = lines.next().unwrap().trim().parse::<usize>().unwrap();
    let mut graph = Graph::new_empty();
    let mut edges: ListOfEdges = Vec::new();

    for l in lines{
        let mut vertices = l.split(",");
        let a = vertices.next().unwrap().to_string();
        let b = vertices.next().unwrap().to_string();
        edges.push((a,b))
    }

    graph.add_directed_edges(&edges);
    graph
}
```

I then coded to find the degree for each of the vertices and printed it out. I then used the degrees found previously to make a histogram for the degree distribution of the graph with the x axis being the number of edges each node has.



Once I got the degrees for each vertex, I combined it to the table with each player's PPG and salary named final_table.csv.

Looking at the combined table, the league wide average for points was 9.9 ppg with an average salary of $8,045,381 and an average degree of 95.8. When sorting the table by the 10 players who had the lowest ppg, the average ppg was 0.6 ppg with an average salary of $1,683,362 and an average degree of 33.9. When sorting the table by the 10 players who had the highest ppg, the average ppg was 28.7 with an average salary of $27,255,465 and an average degree of 7.1. When sorting the table by the 10 players with the lowest degree, we get an average ppg of 28.5, average salary of $29,794,172, and an average degree of 7.

When looking at all the averages mentioned, we can see some relationships between the three factors. For one, there is a positive relationship between ppg and salary, as a player with a higher ppg will typically be paid more. We can also see that having a really low number of nodes can correspond with either being connected with the top scorers in the league or the worst scorers in the league while having a larger number of degrees corresponds with being connected to the average scorers of the league.