

SQL + Python Questions

Author: Ovidiu Mura

Git Repo: <https://github.com/ovimura/pacteraedge-hackathon>

Q1

- Query 1. Rank 10 stores has lowest overall average weekly sales. If there is a tie between two
- average weekly sales, they should have the same rank.

Q1

```
select st, avg_ws, rank() over( order by avg_ws asc) as RANK from  
(select st, avg_ws from  
    (select Store st, avg(Weekly_Sales) avg_ws from  
    sales_data  
    group by Store)  
order by 2 ASC limit 10);
```

Q1

```
25
26 -- Query 1
27
28 select st, avg_ws, rank() over( order by avg_ws asc) as RANK from
29 (select st, avg_ws from
30 (select Store st, avg(Weekly_Sales) avg_ws from sales_data
31 group by Store)
32 order by 2 ASC limit 10);
33
34
```

	st	avg_ws	RANK
1	5	5053.41581286811	1
2	33	5728.41405272083	2
3	44	6038.92981447899	3
4	3	6373.03398295704	4
5	38	7492.4784596577	5
6	16	7863.2241236895	6
7	29	8158.81060920147	7
8	7	8358.76614833022	8
9	36	8584.4125634844	9
10	30	8764.23771939633	10

Q2

-- Query 2: For each type of the stores, find 5 stores with highest average weekly sales.

Q2

```
create table high_avg_ws_A as
select st.Type, st.store, avg(sa.Weekly_Sales) from stores_data st inner join sales_data sa on st.store = sa.store
where st.Type = 'A'
group by 1, 2
order by 3 desc limit 5;
```

```
create table high_avg_ws_B as
select st.Type, st.store, avg(sa.Weekly_Sales) from stores_data st inner join sales_data sa on st.store = sa.store
where st.Type = 'B'
group by 1, 2
order by 3 desc limit 5;
```

```
create table high_avg_ws_C as
select st.Type, st.store, avg(sa.Weekly_Sales) from stores_data st inner join sales_data sa on st.store = sa.store
where st.Type = 'C'
group by 1, 2
order by 3 desc limit 5;
```

```
select type, Store, "avg(sa.Weekly_Sales)" as avg_weely_sales from high_avg_ws_A
UNION
select * from high_avg_ws_B
UNION
select * from high_avg_ws_C;
```

Q2

```
57
58 select type, Store, "avg(sa.Weekly_Sales)" as avg_weely_sales from high_avg_ws_A
59 UNION
60 select * from high_avg_ws_B
61 UNION
62 select * from high_avg_ws_C;
63
64
```

	Type	Store	avg_weely_sales
1	A	2	26898.0700312562
2	A	4	29161.2104147196
3	A	13	27355.13689135
4	A	14	28784.8517270916
5	A	20	29508.3015919327
6	B	10	26332.3038187106
7	B	12	14867.3086192684
8	B	18	15733.3131362207
9	B	22	15181.218886251
10	B	23	19776.180880597
11	C	30	8764.23771939631
12	C	37	10297.3550263669
13	C	38	7492.47845965771
14	C	42	11443.3701179347
15	C	43	13415.1141179085

Q3

-- Query 3: Find stores with increased total sales in December from 2010 to 2011.

Q3

```
SELECT a.Store, a.date, a.total_ws, b.total_ws, b.date from
(select Store, substr(Date, 4,7) as date, sum(Weekly_Sales) as
total_ws, substr(Date, 7,4) as year from sales_data
where (substr(Date, 7,4) = '2011' and substr(Date, 4,2) == '12')
group by 1) a
join
(select Store, substr(Date, 4,7) as date, sum(Weekly_Sales) as
total_ws, substr(Date, 7,4) as year from sales_data
where (substr(Date, 7,4) = '2010' and substr(Date, 4,2) == '12')
group by 1) b
on a.Store = b.Store
where a.total_ws > b.total_ws;
```

Q3

```

64
65 -- Query 3
66 SELECT a.Store, a.date, a.total_ws, b.total_ws, b.date from
67 (select Store, substr(Date, 4,7) as date, sum(Weekly_Sales) as total_ws, substr(Date, 7,4) as year from sales_data
68 where (substr(Date, 7,4) = '2011' and substr(Date, 4,2) == '12')
69 group by 1) a join
70 (select Store, substr(Date, 4,7) as date, sum(Weekly_Sales) as total_ws, substr(Date, 7,4) as year from sales_data
71 where (substr(Date, 7,4) = '2010' and substr(Date, 4,2) == '12')
72 group by 1) b on a.Store = b.Store
73 where a.total_ws > b.total_ws;
74

```

	Store	date	total_ws	total_ws	date
1	1	12/2011	9032594.71	8876953.18	12/2010
2	4	12/2011	13144846.51	12466674.3	12/2010
3	5	12/2011	1931376.8	1829294.03	12/2010
4	6	12/2011	9869325.58000001	9793698.97000001	12/2010
5	7	12/2011	3841626.31	3643627.46	12/2010
6	9	12/2011	3404113.5	3214648.24	12/2010
7	11	12/2011	8395491.14999999	8267278.68	12/2010
8	12	12/2011	6283683.83	6147059.47	12/2010
9	13	12/2011	12800264.37	12587690.25	12/2010
10	16	12/2011	3401326.62	3311064.31	12/2010
11	17	12/2011	5055908.87	4883511.08	12/2010
12	26	12/2011	5854303.21	5813819.77	12/2010
13	31	12/2011	7938387.07	7796083.04	12/2010
14	32	12/2011	7213433.31	7040638.37	12/2010
15	33	12/2011	1221666.15	1170963.26	12/2010
16	37	12/2011	2655367.07	2590847.49	12/2010
17	38	12/2011	2013322.26	1715347.27	12/2010
18	39	12/2011	9497869.8	8630149.33000001	12/2010
19	40	12/2011	5927494.64	5750981.56	12/2010
20	41	12/2011	8051202.84	7509735.45	12/2010
21	42	12/2011	2759543.23	2606910.56	12/2010
22	43	12/2011	3049629.72	2963362.34	12/2010
23	44	12/2011	1539906.15	1428149.1	12/2010

Q4

- 4. Analyze how sales is influenced by holiday
- a. Average weekly sales on holiday weeks and non-holiday weeks
- b. Top 20 weeks with highest weekly sales and calculate portion of holiday weeks and nonholiday weeks.

Q4

-- QUERY 4.a

```
select Date, IsHoliday, avg(Weekly_Sales) as avg_weekly_sales from sales_data
group by 2, 1
order by 2 DESC;
```

-- QUERY 4.b

```
create table h_highest_ws as
select Date, avg(Weekly_Sales) as avg_weekly_sales, IsHoliday from sales_data
where IsHoliday = 'TRUE'
group by 1
order by 2 DESC limit 10;
```

```
create table h_non_highest_ws as
select Date, avg(Weekly_Sales) as avg_weekly_sales, IsHoliday from sales_data
where IsHoliday = 'FALSE'
group by 1
order by 2 DESC limit 10;
```

```
SELECT * FROM h_highest_ws
UNION
SELECT * FROM h_non_highest_ws
order by 3 DESC;
```

Q4

```

75
76 -- QUERY 4.a
77
78 select Date, IsHoliday, avg(Weekly_Sales) as avg_weekly_sales from sales_data
79 group by 2, 1
80 order by 2 DESC;
81

```

	Date	IsHoliday	avg_weekly_sales
1	07/09/2012	TRUE	16294.6929568442
2	09/09/2011	TRUE	15809.0694827586
3	10/02/2012	TRUE	16664.2478907031
4	10/09/2010	TRUE	15537.7588832142
5	11/02/2011	TRUE	16111.7061912865
6	12/02/2010	TRUE	16352.0560317997
7	25/11/2011	TRUE	22043.5634756703
8	26/11/2010	TRUE	22403.3367052417
9	30/12/2011	TRUE	15332.1548584748
10	31/12/2010	TRUE	13738.5385660891
11	01/04/2011	FALSE	14726.8692612674
12	01/06/2012	FALSE	16405.5894393476
13	01/07/2011	FALSE	16232.8623336745
14	01/10/2010	FALSE	14391.7805349233
15	02/03/2012	FALSE	15672.5869464883
16	02/04/2010	FALSE	17098.6202984063
17	02/07/2010	FALSE	16769.7924237231
18	02/09/2011	FALSE	15387.1221668362
19	02/12/2011	FALSE	16496.5118537074
20	03/02/2012	FALSE	15480.5536076587

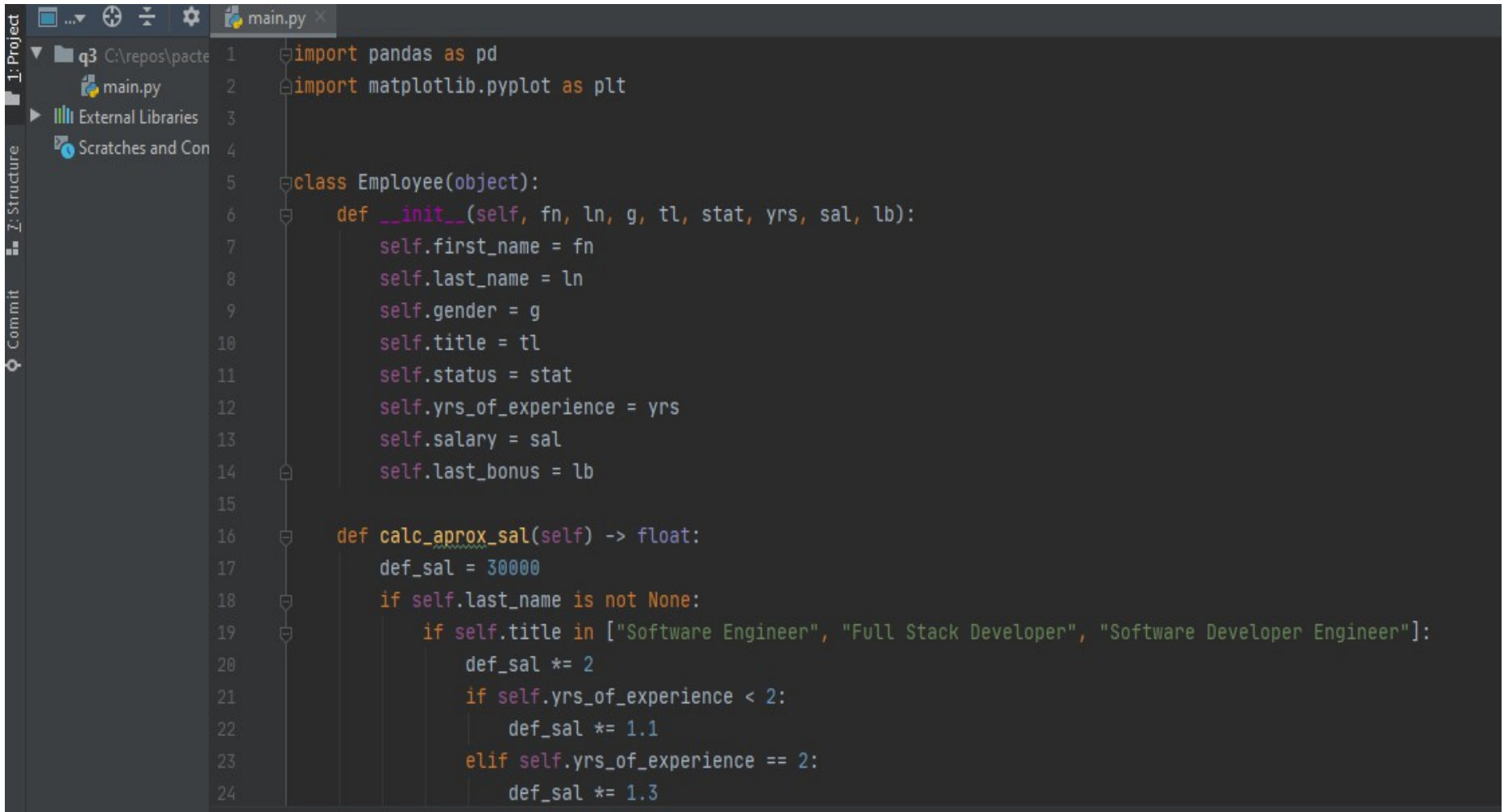
```

95
96 SELECT * FROM h_highest_ws
97 UNION
98 SELECT * FROM h_non_highest_ws
99 order by 3 DESC;
100

```

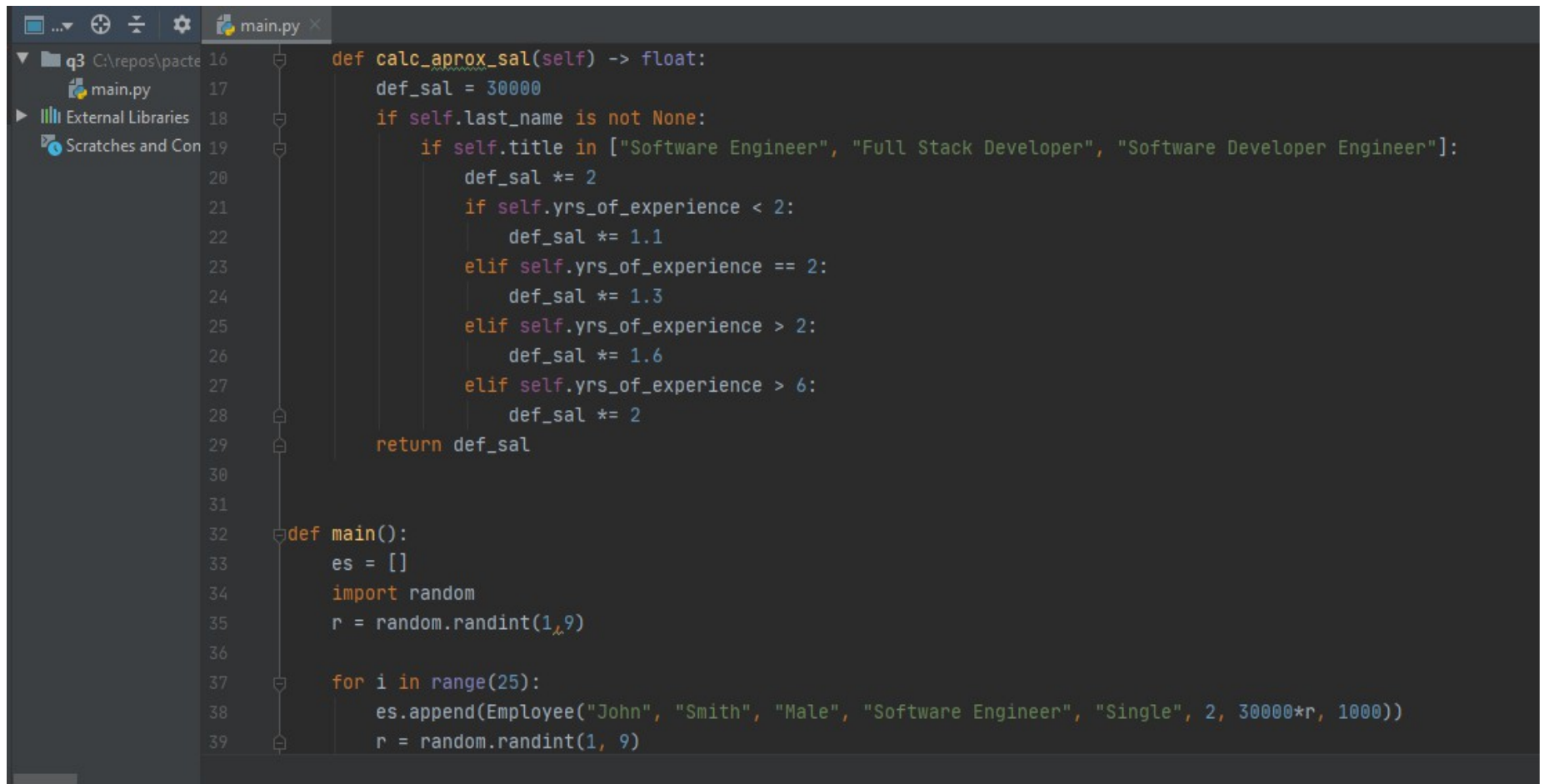
	Date	avg_weekly_sales	IsHoliday
1	07/09/2012	16294.6929568442	TRUE
2	09/09/2011	15809.0694827586	TRUE
3	10/02/2012	16664.2478907031	TRUE
4	10/09/2010	15537.7588832142	TRUE
5	11/02/2011	16111.7061912865	TRUE
6	12/02/2010	16352.0560317997	TRUE
7	25/11/2011	22043.5634756703	TRUE
8	26/11/2010	22403.3367052417	TRUE
9	30/12/2011	15332.1548584748	TRUE
10	31/12/2010	13738.5385660891	TRUE
11	02/04/2010	17098.6202984063	FALSE
12	04/06/2010	17246.9220343642	FALSE
13	06/04/2012	17935.7411565538	FALSE
14	06/07/2012	17309.3623370483	FALSE
15	09/12/2011	18458.8530564785	FALSE
16	10/12/2010	18882.8936194029	FALSE
17	16/12/2011	19942.1493328908	FALSE
18	17/12/2010	20892.463619466	FALSE
19	23/12/2011	25437.1461215725	FALSE
20	24/12/2010	27378.6926928282	FALSE

Python Question



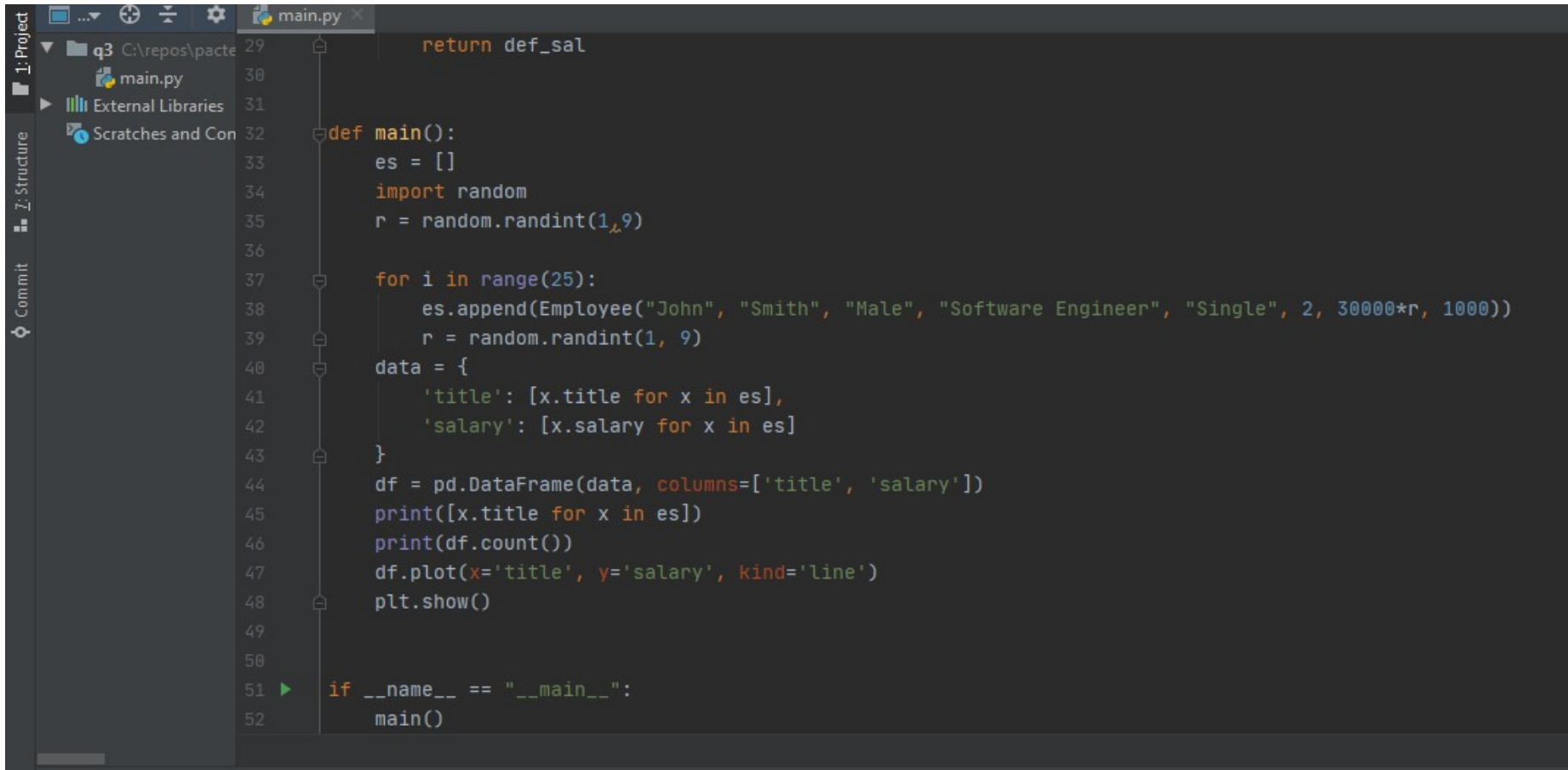
```
1  import pandas as pd
2  import matplotlib.pyplot as plt
3
4
5  class Employee(object):
6      def __init__(self, fn, ln, g, tl, stat, yrs, sal, lb):
7          self.first_name = fn
8          self.last_name = ln
9          self.gender = g
10         self.title = tl
11         self.status = stat
12         self.yrs_of_experience = yrs
13         self.salary = sal
14         self.last_bonus = lb
15
16     def calc_aprox_sal(self) -> float:
17         def_sal = 30000
18         if self.last_name is not None:
19             if self.title in ["Software Engineer", "Full Stack Developer", "Software Developer Engineer"]:
20                 def_sal *= 2
21                 if self.yrs_of_experience < 2:
22                     def_sal *= 1.1
23                 elif self.yrs_of_experience == 2:
24                     def_sal *= 1.3
```

Python Question



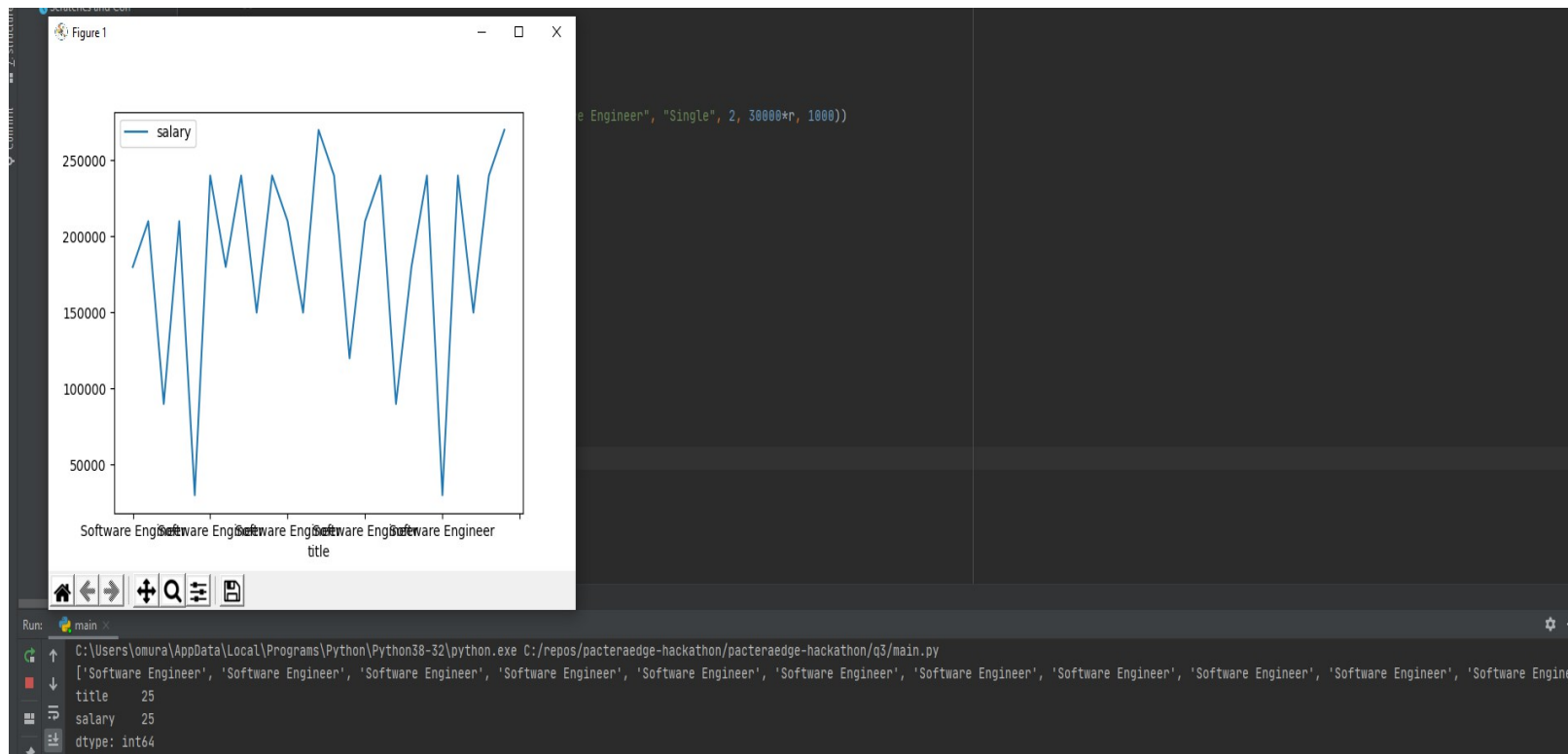
```
16 def calc_aprox_sal(self) -> float:
17     def_sal = 30000
18     if self.last_name is not None:
19         if self.title in ["Software Engineer", "Full Stack Developer", "Software Developer Engineer"]:
20             def_sal *= 2
21             if self.yrs_of_experience < 2:
22                 def_sal *= 1.1
23             elif self.yrs_of_experience == 2:
24                 def_sal *= 1.3
25             elif self.yrs_of_experience > 2:
26                 def_sal *= 1.6
27             elif self.yrs_of_experience > 6:
28                 def_sal *= 2
29     return def_sal
30
31
32 def main():
33     es = []
34     import random
35     r = random.randint(1,9)
36
37     for i in range(25):
38         es.append(Employee("John", "Smith", "Male", "Software Engineer", "Single", 2, 30000*r, 1000))
39     r = random.randint(1, 9)
```

Python Question



```
29     return def_sal
30
31
32 def main():
33     es = []
34     import random
35     r = random.randint(1,9)
36
37     for i in range(25):
38         es.append(Employee("John", "Smith", "Male", "Software Engineer", "Single", 2, 30000*r, 1000))
39         r = random.randint(1, 9)
40     data = {
41         'title': [x.title for x in es],
42         'salary': [x.salary for x in es]
43     }
44     df = pd.DataFrame(data, columns=['title', 'salary'])
45     print([x.title for x in es])
46     print(df.count())
47     df.plot(x='title', y='salary', kind='line')
48     plt.show()
49
50
51 if __name__ == "__main__":
52     main()
```


Python Question



Python Question

