

nir

December 6, 2024

1 -

1.1

```
[1691]: import numpy as np
import pandas as pd

from matplotlib import pyplot as plt
import seaborn as sns

import json
from sklearn.feature_extraction.text import CountVectorizer

from scipy import stats

import plotly.express as px
```

1.2

```
[1651]: # JSON
with open('vacancies_data_with_salary.json', 'r', encoding='utf-8') as file:
    data = json.load(file)

#
vacancies = pd.json_normalize(data)
df = vacancies.copy()
```

## 2 EDA

```
[1652]: df
```

```
[1652]:
```

	id	name	area
0	112620411	\	
1	112595285	React-	
2	112214994	Frontend-	
3	112543205	Backend (Java)	
4	112524085	/frontend	

```

...
7923 110849150 Engineering & Manufacturing Intern / ...
7924 109079028 Engineering & Manufacturing Intern / ...
7925 112419562
7926 111423693 Senior ML Engineer
7927 108659245 Engineering & Manufacturing Intern / ...

```

```

published_at      schedule
0      2024-12-05T16:19:49+0300      \
1      2024-12-05T13:04:08+0300
2      2024-11-30T09:52:47+0300
3      2024-12-04T17:40:31+0300
4      2024-12-04T14:47:08+0300

```

```

...
7923 2024-12-03T10:56:39+0300
7924 2024-11-27T16:20:29+0300
7925 2024-12-03T13:07:49+0300
7926 2024-12-05T14:04:15+0300
7927 2024-12-05T15:58:21+0300

```

```

professional_roles      experience
0      [ ]      3 6 \
1      [ , ]      1 3
2      [ , ]      1 3
3      [ , ]      1 3
4      [ , ]

```

```

...
7923      [ ]
7924      [ ]
7925 [ , ]
7926      [ - ]      3 6
7927      [ ]

```

```

employment      key_skills
0      [ , ] \
1      [React, React Native, CSS, HTML, HTML5, Node.j...
2      [JavaScript, HTML, CSS, VueJS, TypeScript, RES...
3      [Java, Spring Framework, SQL, NoSQL, Git, CI/C...
4      []
...
7923      []
7924      [ ]
7925      []
7926      [Python, SQL, Numpy, pandas, PyTorch, PySpark,...
7927      []

```

```

salary.from salary.to salary.currency salary.gross

```

0	350000.0	500000.0		RUR	False
1	600.0	950.0		USD	False
2	150000.0	200000.0		KZT	True
3	1000000.0	NaN		KZT	True
4	NaN	200.0		USD	False
...	...	...	...	...	...
7923	100000.0	NaN		RUR	True
7924	100000.0	NaN		RUR	True
7925	45000.0	NaN		RUR	True
7926	NaN	330000.0		RUR	False
7927	100000.0	NaN		RUR	True

[7928 rows x 13 columns]

## 2.1

```
[1653]: df.isnull().sum()
```

```
[1653]: id                0
name                  0
area                 0
published_at         0
schedule             0
professional_roles   0
experience           0
employment           0
key_skills           0
salary.from          890
salary.to            3425
salary.currency       0
salary.gross         1
dtype: int64
```

```
[1654]: df = df.drop(['salary.gross'], axis=1)
```

## 2.2

```
[1655]: df = df.rename(columns={'salary.from': 'salary_from',
                                'salary.to': 'salary_to',
                                'salary.currency': 'salary_currency',
                                'salary.gross': 'salary_gross'})
```

## 2.3

```
[ ]:
```

```
[1656]: df[['salary_from', 'salary_to']] = df[['salary_from', 'salary_to']].fillna(0)

#           'salary'
def calculate_salary(row):
    if row['salary_from'] != 0 and row['salary_to'] != 0:
        return (row['salary_from'] + row['salary_to']) / 2
    else:
        return row['salary_from'] + row['salary_to']

df['salary'] = df.apply(calculate_salary, axis=1)
```

## 2.4

```
[1657]: df['published_date'] = pd.to_datetime(df['published_at']).dt.strftime('%Y/%m/
        ↵ %d')
```

## 2.5

```
[1658]: df['job'] = df['professional_roles'].apply(lambda x: ', '.join(x))
```

```
[1659]: it_positions = [' ', ' ', ' ', 'Data', 'Developer', ' ',  
    ↪ ' ', 'UI/UX ', 'DevOps', 'Machine Learning Engineer', '  
    ↪ ' ', '']  
  
df = df[df['job'].str.contains('|'.join(it_positions), regex=True)]
```

```
[1660]: df = df.drop(['published_at', 'professional_roles', 'salary_from', 'salary_to'], axis=1)
```

## 2.6

```
[1661]: #
currency_rates = {
    'USD': 94.5,      #
    'KZT': 0.2,      #
    'BYR': 0.037,    #
    'UZS': 0.0078,   #
    'EUR': 103.2,     #
    'KGS': 1.08,      #
    'AZN': 55.0       #
}

#
df['salary_rub'] = df.apply(
    lambda row: row['salary'] * currency_rates.get(row['salary_currency'], 1),
    axis=1
)
```

## 2.7

```
[1662]: df["key_skills"] = df["key_skills"].apply(lambda x: ",".join(x if x else ' ')  
↪      ')
```

```
[1663]: df = df.drop('salary', axis=1)
```

```
[1664]: df = df.drop(['salary_currency', 'id'], axis=1)
```

## 2.8

```
[1665]: df['salary_rub'] = df['salary_rub'].astype(int)  
  
df['published_date'] = pd.to_datetime(df['published_date'])
```

## 2.9

```
[1666]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
Index: 4944 entries, 1 to 7927  
Data columns (total 9 columns):  
#   Column                Non-Null Count  Dtype    
---  ---  
0   name                   4944 non-null   object   
1   area                   4944 non-null   object   
2   schedule               4944 non-null   object   
3   experience              4944 non-null   object   
4   employment              4944 non-null   object   
5   key_skills              4944 non-null   object   
6   published_date          4944 non-null   datetime64[ns]  
7   job                     4944 non-null   object   
8   salary_rub              4944 non-null   int64    
dtypes: datetime64[ns](1), int64(1), object(7)  
memory usage: 386.2+ KB
```

```
[1667]: df.isnull().sum()
```

```
[1667]: name                0  
area                    0  
schedule                0  
experience               0  
employment              0  
key_skills               0  
published_date          0  
job                     0  
salary_rub              0  
dtype: int64
```

## 3 analysis\_df

```
[1668]: analysis_df = df.copy()
```

### 3.1

```
[1669]: def plot_percent(df, column_name, title, xlabel, flip_axes=False):
#         -10
    top_10 = df[column_name].value_counts(normalize=True).multiply(100).
    ↪round(2).head(10).reset_index(name='percent')

    #
    plt.figure(figsize=(12, 6), dpi=200)

    if flip_axes:
        plt.figure(figsize=(16, 8), dpi=200)
        #         flip_axes True,           X   Y
        sns.barplot(data=top_10, x='percent', y=column_name)
        plt.xlabel('      , %')
        plt.ylabel(xlabel)
    else:
        plt.figure(figsize=(12, 6), dpi=200)
        #         ,
        sns.barplot(data=top_10, x=column_name, y='percent')
        plt.xlabel(xlabel)
        plt.ylabel('      , %')
        plt.xticks(rotation=45)

    plt.title(title)

    #
    for index, value in enumerate(top_10['percent']):
        if flip_axes:
            plt.text(value + 0.5, index, f'{value}%', va='center', ha='left', ↵
            ↪fontsize=10)
        else:
            plt.text(index, value + 0.5, f'{value}%', ha='center', va='bottom', ↵
            ↪fontsize=10)

    plt.show()
```

### 3.2 area

#### 3.2.1

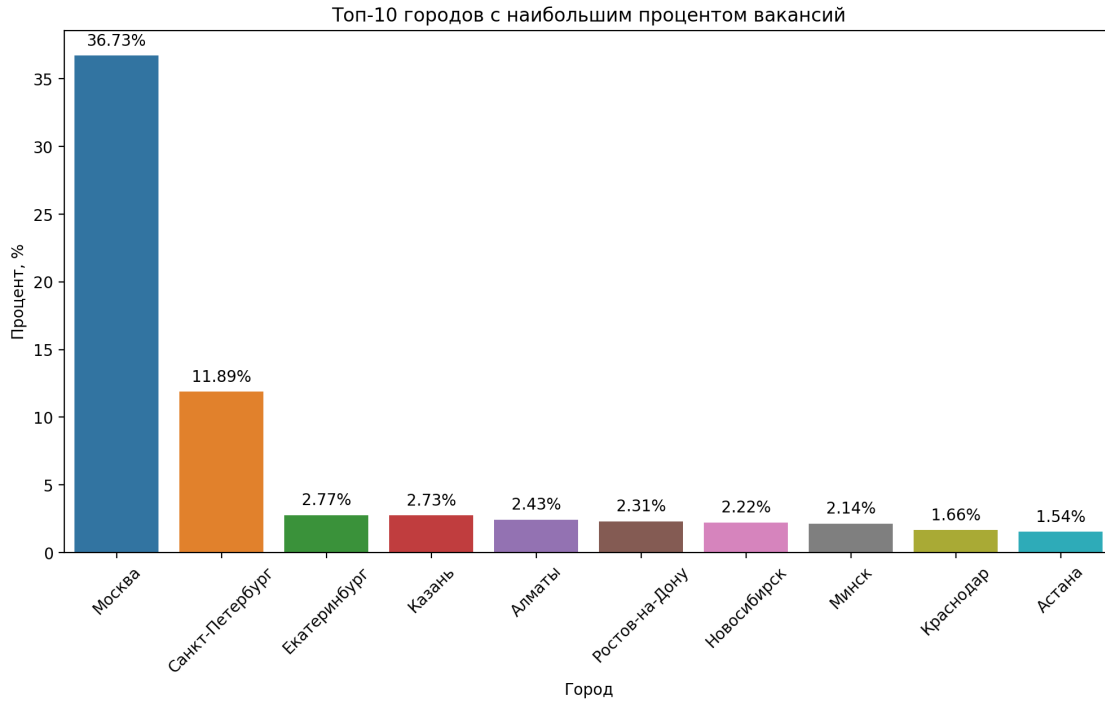
```
[1670]: print(f'                - {analysis_df["area"].nunique()}')
```

### 3.2.2

```
[1671]: plot_percent(analysis_df, 'area', ' -10', ' ', ' ')

```

```
<Figure size 2400x1200 with 0 Axes>
```

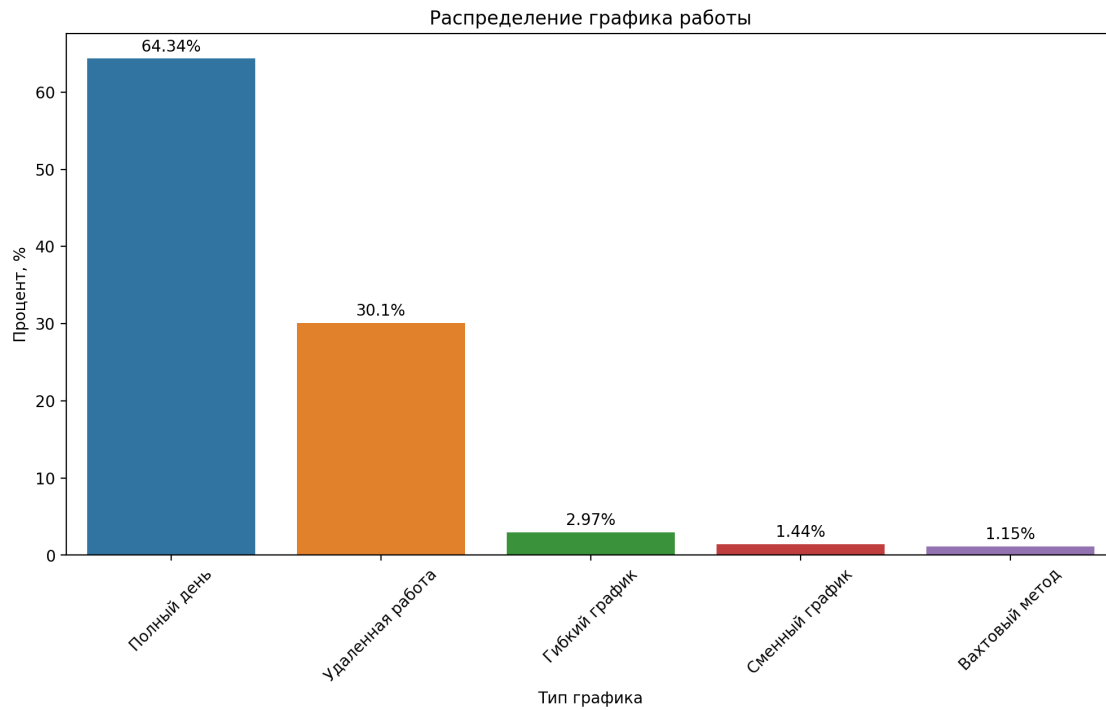


### 3.3 schedule

```
[1672]: plot_percent(analysis_df, 'schedule', ' ', ' ', xlabel=' ')

```

<Figure size 2400x1200 with 0 Axes>

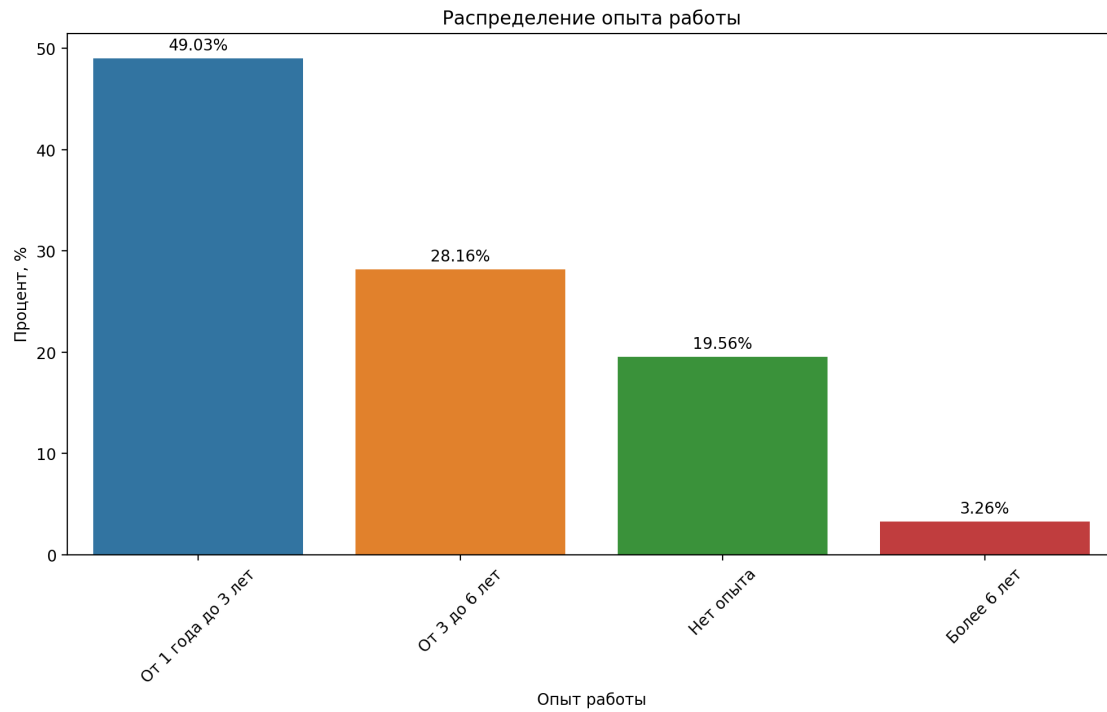


### 3.4 experience

```
[1673]: plot_percent(analysis_df, 'experience', title=' ', xlabel=' ')
↳ ')
```

<Figure size 2400x1200 with 0 Axes>

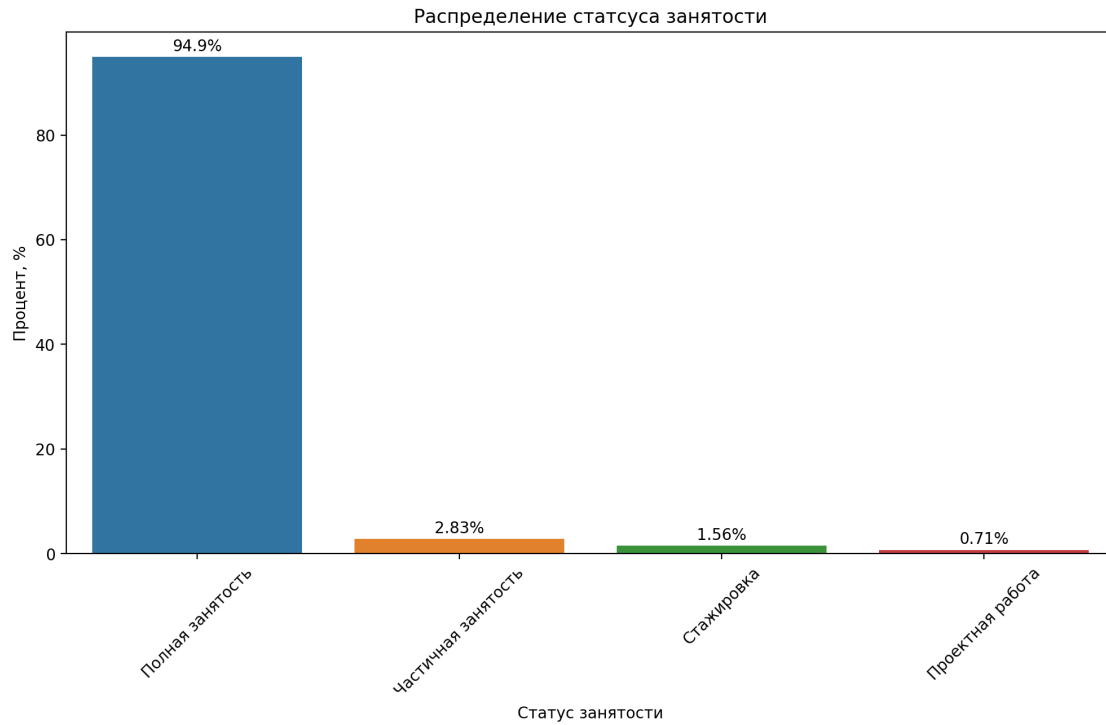




### 3.5 employment

```
[1674]: plot_percent(analysis_df, 'employment', ' ', 'C', '')
```

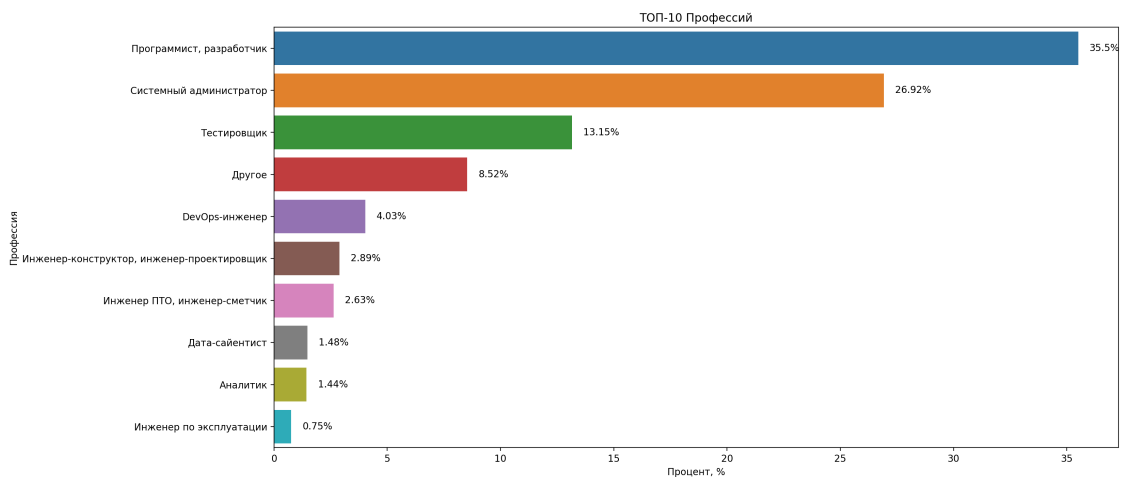
<Figure size 2400x1200 with 0 Axes>



### 3.6 job

```
[1675]: plot_percent(analysis_df, 'job', '-10', ' ', True)
```

<Figure size 2400x1200 with 0 Axes>



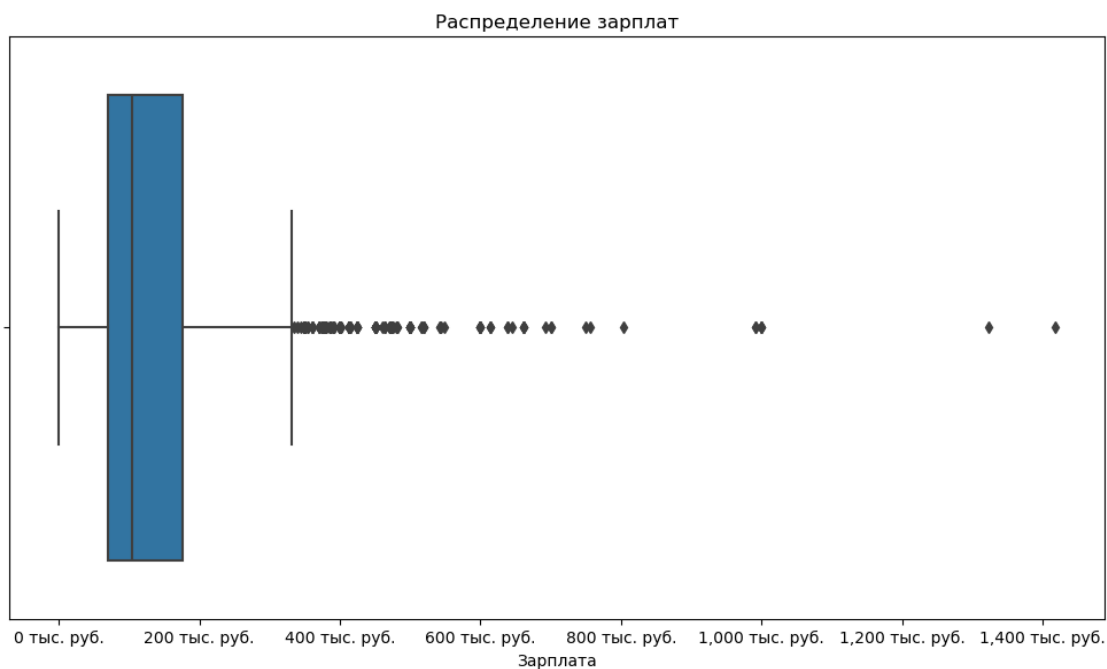
### 3.7 salary

```
[1676]: from matplotlib.ticker import FuncFormatter
```

```
#
plt.figure(figsize=(10, 6))
sns.boxplot(x=analysis_df['salary_rub'])

#           X
formatter = FuncFormatter(lambda x, pos: f'{int(x / 1000):,} . .')
plt.gca().xaxis.set_major_formatter(formatter)

plt.xlabel('')
plt.title('')
plt.tight_layout()
plt.show()
```



```
[1677]: plt.figure(figsize=(15,8), dpi=200)
```

```
#
mode_salary = analysis_df['salary_rub'].mode()[0]

#           histplot   KDE
```

```

sns.histplot(analysis_df['salary_rub'], kde=True, color='skyblue', bins=30)

#           X,
plt.xlabel('           ( . .)')
plt.ylabel('           ')
plt.title('           KDE')

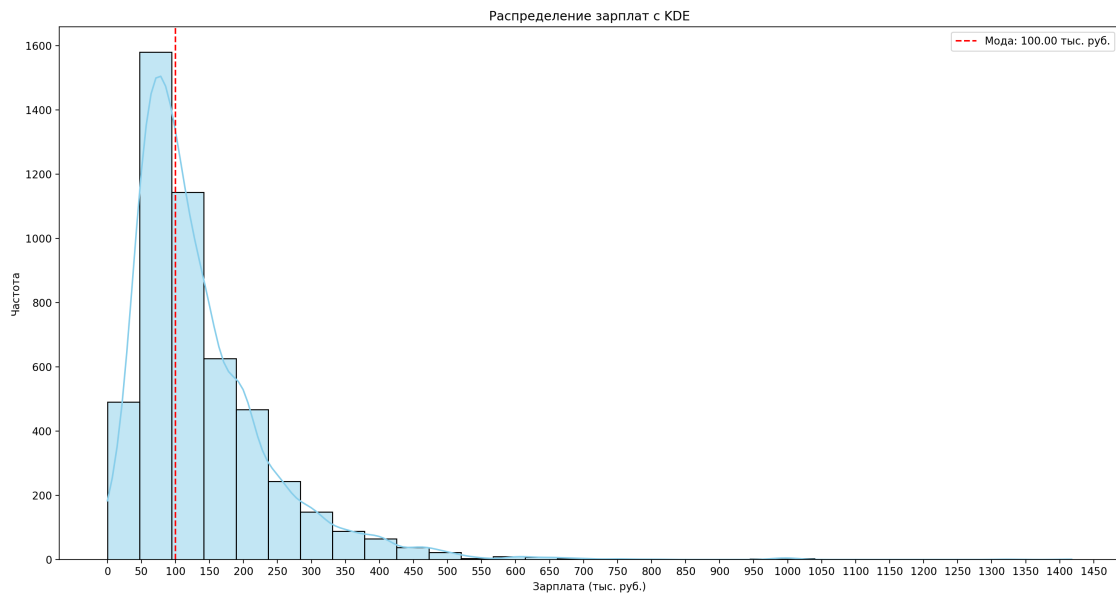
#
plt.axvline(mode_salary, color='red', linestyle='--', label=f' : {mode_salary/
↪1000:.2f} . .')

#
plt.legend()

#           X,           50 .
xticks = np.arange(0, analysis_df['salary_rub'].max() + 50000, step=50000) #_
↪           50 .
plt.xticks(xticks, labels=[f'{int(x/1000)}' for x in xticks]) # " . ." _
↪

plt.tight_layout()
plt.show()

```



4

4.1 : .

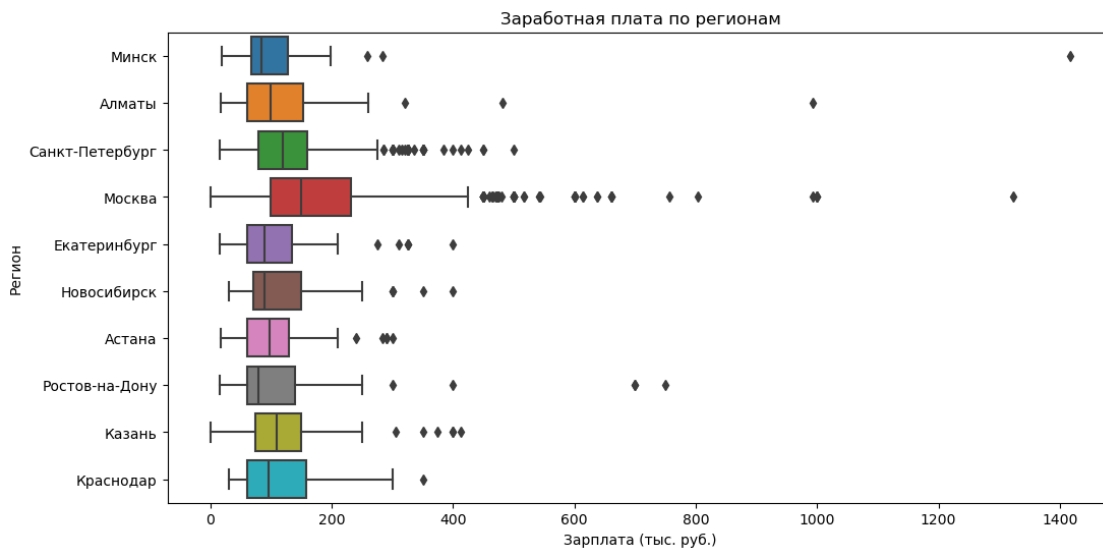
4.1.1 -

```
[1678]: # DataFrame ' ', 1000
analysis_df.loc[(analysis_df['area'] == ' ') & (analysis_df['salary_rub'] <
↪1000), 'salary_rub'] *= 1000
```

```
[1679]: top_10_area = analysis_df[analysis_df['area'].isin([' ', ' - ',
↪' ', ' ', ' ', ' - - ', ' ', ' ', ' ', ' '))]

# .loc
top_10_area.loc[:, 'salary_rub'] = top_10_area['salary_rub'] / 1000 #
↪

plt.figure(figsize=(12, 6))
sns.boxplot(data=top_10_area, y='area', x='salary_rub')
plt.ylabel(' ')
plt.xlabel(' ( . .)')
plt.title(' ')
plt.show()
```



4.2 : , .

[ ]:

```
[1680]: experience_salary = analysis_df.groupby('experience')['salary_rub'].mean().
        ↪round().reset_index(name='mean_salary').sort_values(by='mean_salary')
experience_salary
```

```
[1680]:
```

	experience	mean_salary	
1		71308.0	
2	1	3	113581.0
3	3	6	207756.0
0		6	285242.0

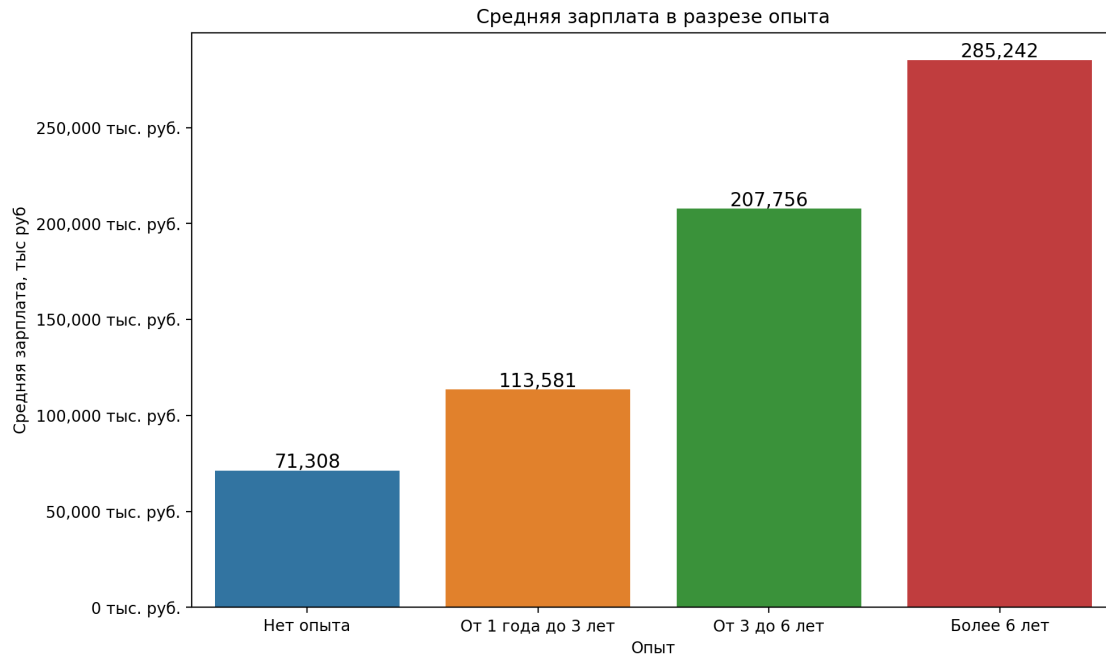
```
[1681]: #
plt.figure(figsize=(10, 6), dpi=200)
ax = sns.barplot(data=experience_salary, x='experience', y='mean_salary')

#
for p in ax.patches:
    ax.annotate(f'{p.get_height():,.0f}',
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center',
                fontsize=12, color='black',
                xytext=(0, 5), textcoords='offset points')

#
formatter = FuncFormatter(lambda x, pos: f'{int(x):,} . .')
ax.yaxis.set_major_formatter(formatter)

#
plt.xlabel('')
plt.ylabel('')
plt.title('')

#
plt.tight_layout()
plt.show()
```



4.3 : .

[1682]: analysis\_df

[1682]:

		name	area
1	React-		\
2	Frontend-		
3	Backend (Java)		
4	/frontend		
5	Junior Frontend-	-	
...		...	...
7922	Engineering & Manufacturing Intern /	...	
7923	Engineering & Manufacturing Intern /	...	
7924	Engineering & Manufacturing Intern /	...	
7926	Senior ML Engineer		
7927	Engineering & Manufacturing Intern /	...	

	schedule	experience	employment
1	1	3	\
2	1	3	
3	1	3	
4			
5			
...	...	...	...
7922			

7923  
7924  
7926  
7927

3 6

```
key_skills published_date
1 React,React Native,CSS,HTML,HTML5,Node.js,JSON 2024-12-05 \
2 JavaScript,HTML,CSS,VueJS,TypeScript,REST API,ES6 2024-11-30
3 Java,Spring Framework,SQL,NoSQL,Git,CI/CD,Dock... 2024-12-04
4 2024-12-04
5 JavaScript,REST API,Git,TypeScript,React Nativ... 2024-12-04
...
7922 2024-12-04
7923 2024-12-03
7924 2024-11-27
7926 Python,SQL,Numpy,pandas,PyTorch,PySpark,Hadoop... 2024-12-05
7927 2024-12-05
```

```
job salary_rub
1 , 73237
2 , 35000
3 , 200000
4 , 18900
5 , 125000
...
7922 100000
7923 100000
7924 100000
7926 - 330000
7927 100000
```

[4944 rows x 9 columns]

```
[1683]: employment_salary =analysis_df.groupby('employment')['salary_rub'].mean().
        ↪round().reset_index(name='mean_salary').sort_values(by='mean_salary')
employment_salary
```

```
[1683]: employment mean_salary
2 59641.0
3 78492.0
0 140350.0
1 151180.0
```

```
[1684]: #
plt.figure(figsize=(10, 6), dpi=200)
ax = sns.barplot(data=employment_salary, x='employment', y='mean_salary')
```



```

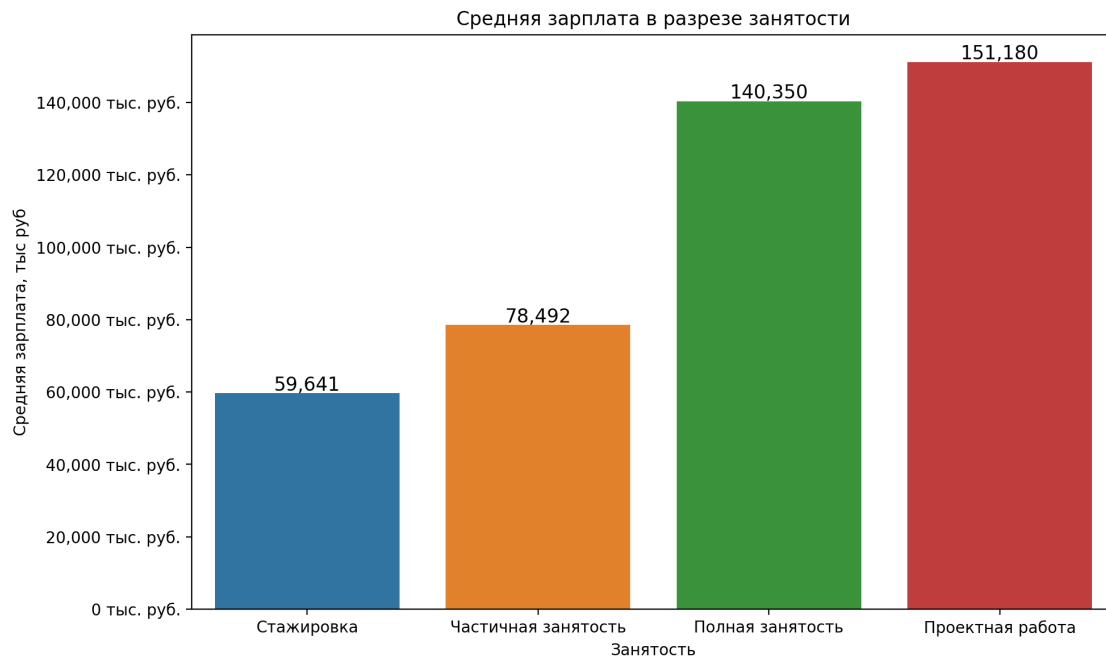
#
for p in ax.patches:
    ax.annotate(f'{p.get_height():,.0f}',
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center',
                fontsize=12, color='black',
                xytext=(0, 5), textcoords='offset points')

#
Y
formatter = FuncFormatter(lambda x, pos: f'{int(x):,} . .')
ax.yaxis.set_major_formatter(formatter)

#
plt.xlabel('')
plt.ylabel(' , ')
plt.title(' ')

#
plt.tight_layout()
plt.show()

```



4.4 : “ ” .

```
[1685]: # , " " key_skills
analysis_df['has_training'] = analysis_df['key_skills'].str.contains(' ',
↪ case=False, na=False)

# 'has_training'
analysis_df['has_training'] = analysis_df['has_training'].apply(lambda x: ' '
↪ ' if x else ' ')
```

```
[1686]: has_training_salary = analysis_df.groupby('has_training')['salary_rub'].mean()
has_training_salary
```

```
[1686]: has_training
132159.620213
139918.316025
Name: salary_rub, dtype: float64
```

```
[1687]: # 'has_training'
group_with_training = analysis_df[analysis_df['has_training'] == ' '
↪ '']['salary_rub']
group_without_training = analysis_df[analysis_df['has_training'] == ' '
↪ '']['salary_rub']

# -
shapiro_with_training = stats.shapiro(group_with_training.dropna())
shapiro_without_training = stats.shapiro(group_without_training.dropna())

# p-value 0.05,
if shapiro_with_training.pvalue < 0.05 or shapiro_without_training.pvalue < 0.
↪ 05:
    print(' , - .')
    # -
    u_stat, p_value = stats.mannwhitneyu(group_with_training,
↪ group_without_training, alternative='two-sided')
else:
    print(' , t- .')
    # t-
    t_stat, p_value = stats.ttest_ind(group_with_training,
↪ group_without_training, nan_policy='omit')

#
if p_value < 0.05:
    print(f' (p-value = {p_value})')
else:
    # p-value 2
    print(f' (p-value = {p_value})')
```

```
analysis_df = analysis_df.drop(['has_training'], axis=1)
```

(p-value = 0.04068661721120872)

## 4.5 -10 IT

```
[1688]: # " "
skills_by_job = (
    analysis_df
    .assign(key_skills=analysis_df['key_skills'].str.split(',')) #
    ↪
    .explode('key_skills') #
    .query("key_skills != ' '") # " "
    .groupby(['job', 'key_skills']) #
    .size() #
    .reset_index(name='count') # DataFrame
)

# -10
top_skills_by_job = (
    skills_by_job
    .sort_values(['job', 'count'], ascending=[True, False]) #
    .groupby('job') #
    .head(10) # -10
    .groupby('job')['key_skills'] #
    .apply(lambda x: ', '.join(x)) #
    .reset_index(name='top_skills') # DataFrame
)
```

```
[1689]: #
top_skills_by_job['job'] = top_skills_by_job['job'].str.wrap(30) #
↪
top_skills_by_job['top_skills'] = top_skills_by_job['top_skills'].str.wrap(100)
↪ #
top_skills_by_job = top_skills_by_job.sort_values('job', ascending=True) #
↪

#
plt.figure(figsize=(18, 14)) #

#
bars = plt.barh(top_skills_by_job['job'], [1] * len(top_skills_by_job),
↪ color='skyblue', height=0.8) #

# ( )
```

```

for bar, skills in zip(bars, top_skills_by_job['top_skills']):
    plt.text(
        0.02, #
        bar.get_y() + bar.get_height() / 2,
        skills,
        ha='left',
        va='center',
        fontsize=10,
        color='black',
    )

#
plt.xlabel(' ', fontsize=14, labelpad=15) # X
plt.ylabel(' ', fontsize=14, labelpad=15) # Y
plt.title(' -10 ', fontsize=16, pad=20) #
plt.xticks([]) # X
plt.tight_layout(pad=2.5) #

#
plt.show()

```

Топ-10 навыков по профессиям

Профессии	Тестировщик	Тестирование, Функциональное тестирование, SQL, QA, Postman, Ручное тестирование, API, Atlassian Jira, Тестирование пользовательского интерфейса, Git
	Системный администратор	Администрирование сетевого оборудования, Настройка ПК, Настройка сетевых подключений, Администрирование серверов Windows, Linux, Настройка ПО, Техническая поддержка пользователей, Администрирование серверов, Установка ПО, Администрирование серверов Linux
	Программист, разработчик	Git, JavaScript, PostgreSQL, SQL, TypeScript, React, Python, HTML, PHP, Docker
	Инженер-энергетик, инженер-электрик	Пуско-наладочные работы, Работа в команде, Разработка электросхем, 4 группа по электробезопасности, MS Dos, Выполнение технического надзора, Грамотность, Знание компьютера, Исполнительность, Компьютер
	Инженер-электроник, инженер-электронщик	Схемотехника электронного оборудования, Диагностика неисправностей, Подбор оборудования, Altium Designer, БПЛА, Ввод оборудования в эксплуатацию, Контроль состояния оборудования, Модульный ремонт техники, Монтаж электрооборудования, Обращение с технической документацией
	Инженер-конструктор, инженер-проектировщик	AutoCAD, ESKD, Компас-3D, SolidWorks, Проектно-конструкторская деятельность, Разработка проектной документации, Проектирование, Чтение чертежей, 3D Моделирование, Проектная документация
	Инженер по эксплуатации	Инженерные системы, Водоснабжение и канализация, Организаторские навыки, Вентиляция и кондиционирование, Контроль исправности оборудования, Мобильность, Строительство, Техническая эксплуатация, Техническое обслуживание зданий, Техническое обслуживание оборудования
	Инженер по охране труда и технике безопасности, инженер-эколог	Охрана труда и техника безопасности, Охрана труда, Пожарная безопасность, Вводный инструктаж по охране труда и технике безопасности, Обеспечение техники безопасности, Проведение инструктажей, Взаимодействие с органами государственного контроля, разработка инструкций по охране труда, Аттестация по ОТ и ТБ, Знание законодательства РФ
	Инженер по качеству	Контроль качества, Linux, TCP/IP, Наличие сертификации ISTQB или аналогичные сертификаты, Настройка ПК, Пользователь ПК, Продвинутый пользователь ПК, Сборка ПК, Сертификация продукции, Тестирование
	Инженер ПТО, инженер-сметчик	AutoCAD, Проектная документация, Строительство, КС-2, Строительная документация, КС-3, Строительный контроль, Документальное сопровождение, Исполнительная документация, Деловая переписка
Навыки	Инженер ПНР	Автоматизация технологических процессов, Пуско-наладочные работы, АСУ ТП, Знание технологических процессов, Инженерные системы, Монтаж оборудования, AutoCAD, Linux, MS Office, Modbus
	Другое	Английский язык, Python, Работа с большим объемом информации, Деловая переписка, Ответственность, MS Excel, Работа в команде, Внимательность, SQL, Деловое общение
	Дата-сайентист	Python, SQL, Big Data, pandas, PyTorch, PostgreSQL, Machine Learning, Математическая статистика, Scikit-learn, Apache Airflow
	Аналитик	SQL, Анализ данных, Аналитическое мышление, Сбор и анализ информации, Python, MS Excel, Бизнес-анализ, Математическая статистика, Работа с базами данных, Разработка технических заданий
	DevOps-инженер	Linux, Docker, DevOps, Kubernetes, CI/CD, PostgreSQL, Ansible, Bash, Nginx, Python

```
[1690]: analysis_df
```

```
[1690]:
1          React-
2      Frontend-
3      Backend      (Java)
4      /frontend
5      Junior Frontend-      -
...
7922 Engineering & Manufacturing Intern /      ...
7923 Engineering & Manufacturing Intern /      ...
7924 Engineering & Manufacturing Intern /      ...
7926      Senior ML Engineer
7927 Engineering & Manufacturing Intern /      ...
```

```

schedule      experience      employment
1      1      3      \
2      1      3
3      1      3
4
5
...
7922
7923
7924
7926      3      6
7927
```

```

key_skills published_date
1      React,React Native,CSS,HTML,HTML5,Node.js,JSON      2024-12-05 \
2      JavaScript,HTML,CSS,VueJS,TypeScript,REST API,ES6      2024-11-30
3      Java,Spring Framework,SQL,NoSQL,Git,CI/CD,Dock...      2024-12-04
4      2024-12-04
5      JavaScript,REST API,Git,TypeScript,React Nativ...      2024-12-04
...
7922      2024-12-04
7923      2024-12-03
7924      2024-11-27
7926 Python,SQL,Numpy,pandas,PyTorch,PySpark,Hadoop...      2024-12-05
7927      2024-12-05
```

```

job salary_rub
1      ,      73237
2      ,      35000
3      ,      200000
4      ,      18900
5      ,      125000
```

```

...
7922          100000
7923          100000
7924          100000
7926          -      330000
7927          100000

```

[4944 rows x 9 columns]

[ ]:

```

[1693]: #           ,           6           2024
start_date = pd.to_datetime("2024-11-06")
df_filtered = df[df['published_date'] >= start_date]

#
df_grouped = df_filtered.groupby([df_filtered['published_date'].dt.date, ↵
                                ↵ 'job']).size().reset_index(name='count')

#
fig = px.line(df_grouped, x='published_date', y='count', color='job',
              title=' ',
              labels={'published_date': ' ', 'count': ' '})
fig.show()

```

4.6 : , ( 10 17 , 17 24 )

.

.