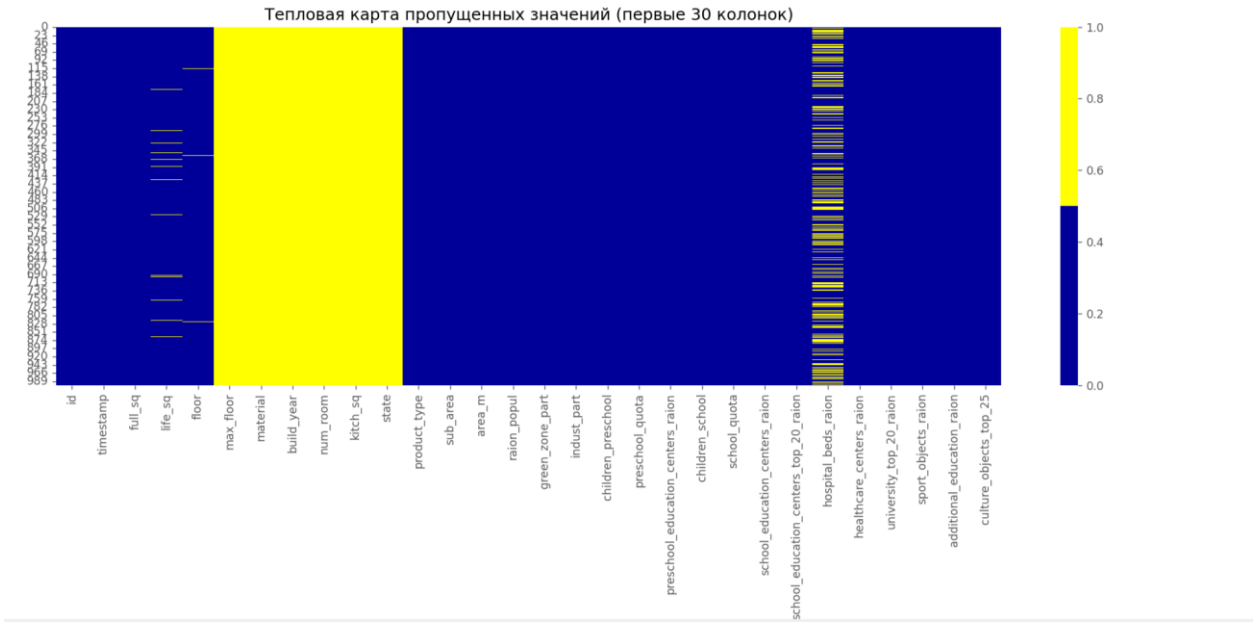
Технология ETL

Вариант 2

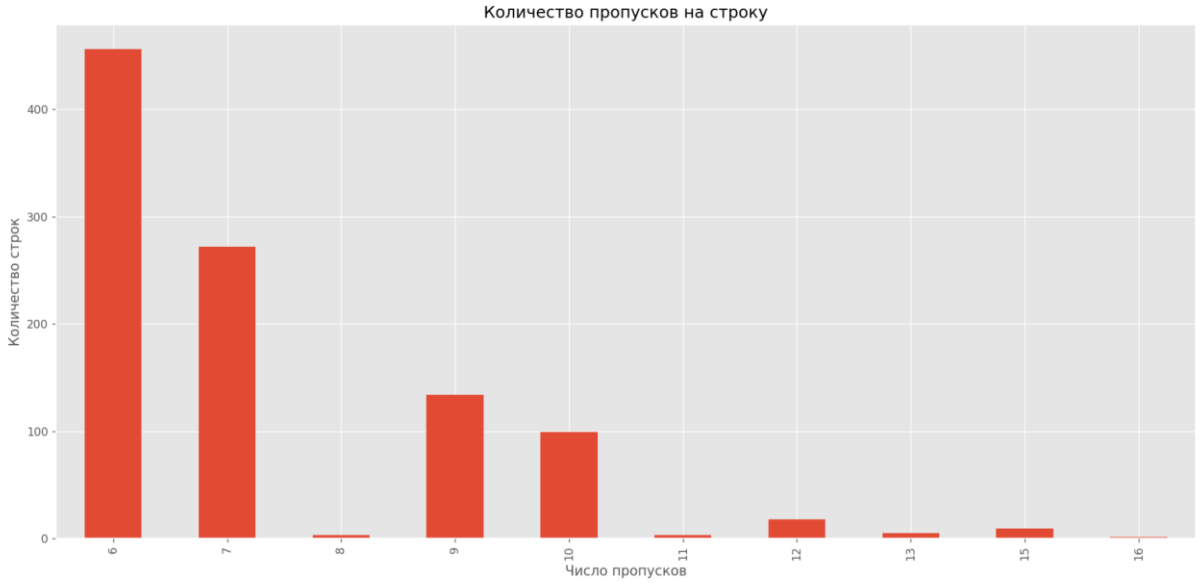
а) определить отсутствующие данные (тепловая карта, процентный список, гистограмма с отсутствующими данными) и принять обоснованное решение о дальнейшей работе с ними (отбросить наблюдения, отбросить параметр, заменить отсутствующие данные)

```
# Тепловая карта пропущенных значений (первые 30 колонок)
cols = df.columns[:30]
colours = ['#000099', '#ffff00']  # синий = данные, желтый = пропуски
sns.heatmap(df[cols].isnull(), cmap=sns.color_palette(colours))
plt.title('Тепловая карта пропущенных значений (первые 30 колонок)')
plt.tight_layout()
plt.show()

# Процент пропусков по каждому признаку
print("Процент пропусков по признакам:")
dict_missing_percentages = {}
for col in df.columns:
    pct_missing = np.mean(df[col].isnull())
    if pct_missing > 0:  # Показываем только колонки с пропусками
        print(f'{col} - {round(pct_missing*100)}%')
        dict_missing_percentages[col] = pct_missing

# Создаем индикаторы пропусков
for col in df.columns:
    if df[col].isnull().any():
        df[f'{col}_ismissing'] = df[col].isnull()

# Гистограмма количества пропусков на строку
ismissing_cols = [col for col in df.columns if 'ismissing' in col]
df['num_missing'] = df[ismissing_cols].sum(axis=1)

plt.figure(figsize=(10, 6))
df['num_missing'].value_counts().sort_index().plot.bar()
plt.title('Количество пропусков на строку')
plt.xlabel('Число пропусков')
plt.ylabel('Количество строк')
plt.tight_layout()
plt.show()

# Гистограмма количества пропусков по признакам
missing_counts = df[ismissing_cols].sum(axis=0)
missing_counts.index = missing_counts.index.str.replace('_ismissing', '')
plt.figure(figsize=(12, 6))
missing_counts.sort_values().plot.bar()
plt.ylabel('Количество пропусков')
```

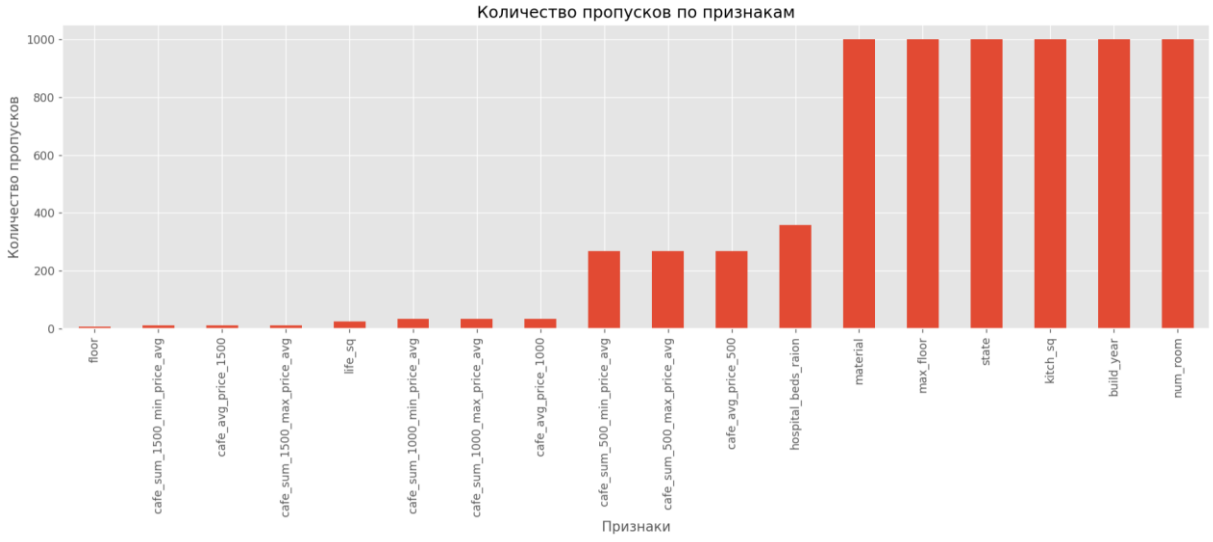
```
plt.xlabel('Признаки')
plt.title('Количество пропусков по признакам')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()

# Принятие решения о пропусках
print(f"Исходное количество строк: {len(df)}")

# Отбрасываем строки, где пропусков слишком много (>2)
ind_missing = df[df['num_missing'] > 2].index
df = df.drop(ind_missing, axis=0)
print(f"Осталось строк после удаления строк с >2 пропусками: {len(df)}")

# Удаляем признак с наибольшим числом пропусков (если процент пропусков > 50%)
if dict_missing_percentages:
    col_with_max_missing = max(dict_missing_percentages,
key=dict_missing_percentages.get)
    if dict_missing_percentages[col_with_max_missing] > 0.5:
        df = df.drop(columns=[col_with_max_missing])
        print(f"Удален столбец с наибольшим количеством пропусков:
{col_with_max_missing}")

# Замена пропусков (числовые - медианой, категориальные - модой)
for col in df.select_dtypes(include=[np.number]):
    if df[col].isnull().any():
        df[col] = df[col].fillna(df[col].median())

for col in df.select_dtypes(exclude=[np.number]):
    if df[col].isnull().any():
        df[col] = df[col].fillna(df[col].mode()[0])
```



Тепловая карта показывает распределение пропусков. Она двухцветная: синий(данные присутствуют) и желтый (отсутствуют). По оси x – названия признаков, по оси y – номера строк наблюдения. Есть столбцы, где для второй тысячи строк из таблицы данные отсутствуют.



По оси x – количество пропущенных значений в одной строке, по оси y – количество строк с соответствующим числом пропусков.



По оси x – названия признаков, по оси y – количество пропущенных значений в каждом признаке. Помогает понять, какие признаки не требуют вмешательства.

```
life_sq - 2%
floor - 1%
```

```
max_floor - 100%
material - 100%
build_year - 100%
num_room - 100%
kitch_sq - 100%
state - 100%
hospital_beds_raion - 36%
cafe_sum_500_min_price_avg - 27%
cafe_sum_500_max_price_avg - 27%
cafe_avg_price_500 - 27%
cafe_sum_1000_min_price_avg - 3%
cafe_sum_1000_max_price_avg - 3%
cafe_avg_price_1000 - 3%
cafe_sum_1500_min_price_avg - 1%
cafe_sum_1500_max_price_avg - 1%
cafe_avg_price_1500 - 1%
```

Осталось строк после удаления: 600

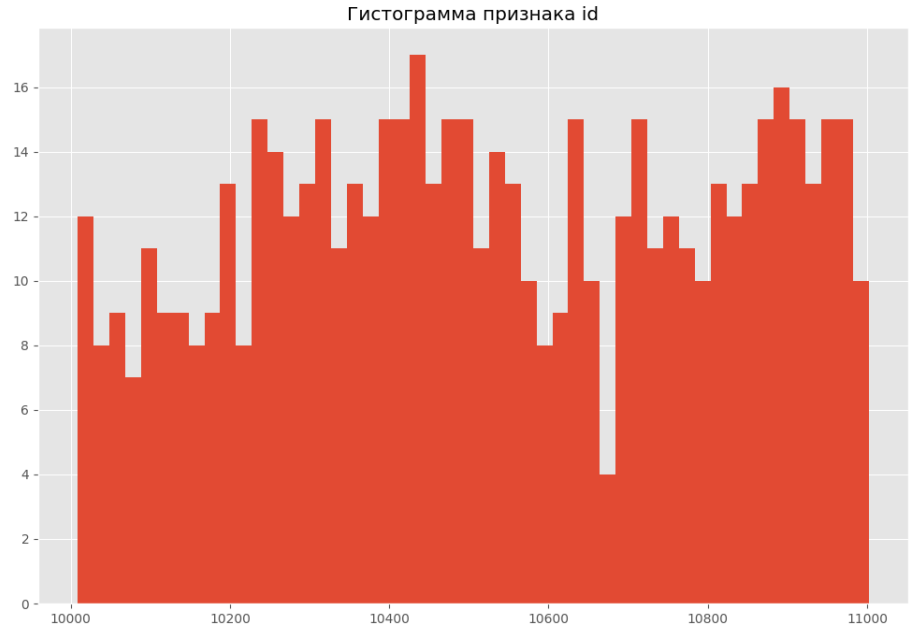
Удален столбец с наибольшим количеством пропусков: hospital_beds_raion

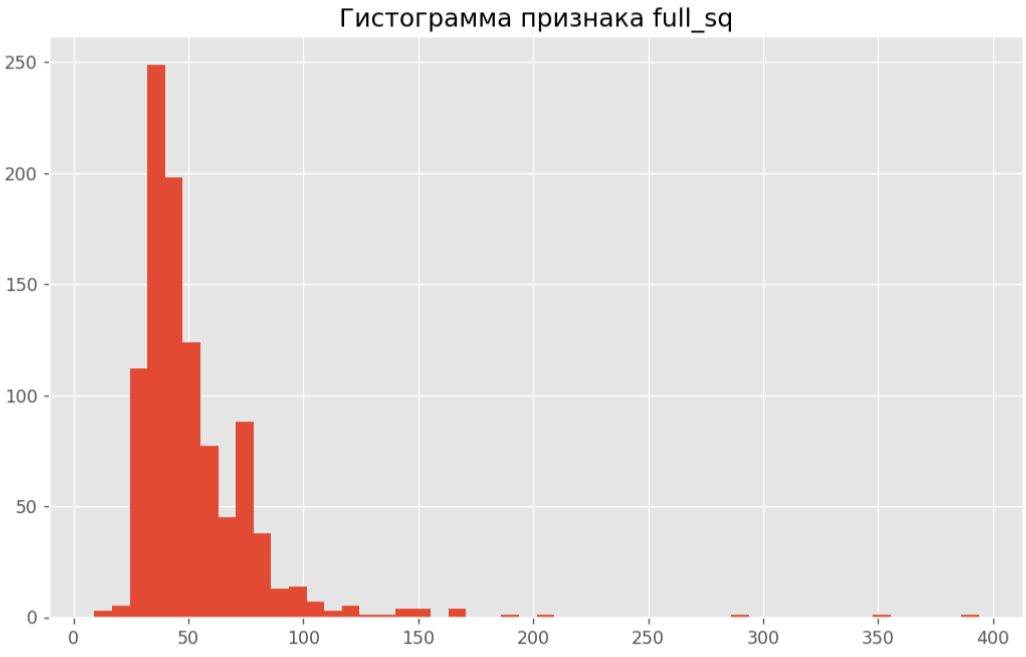
б) определить наличие выбросов в данных (построение гистограммы, использование описательной статистики) и принять обоснованное решение о дальнейшей работе с ними (отбрасываем, корректируем, оставляем)

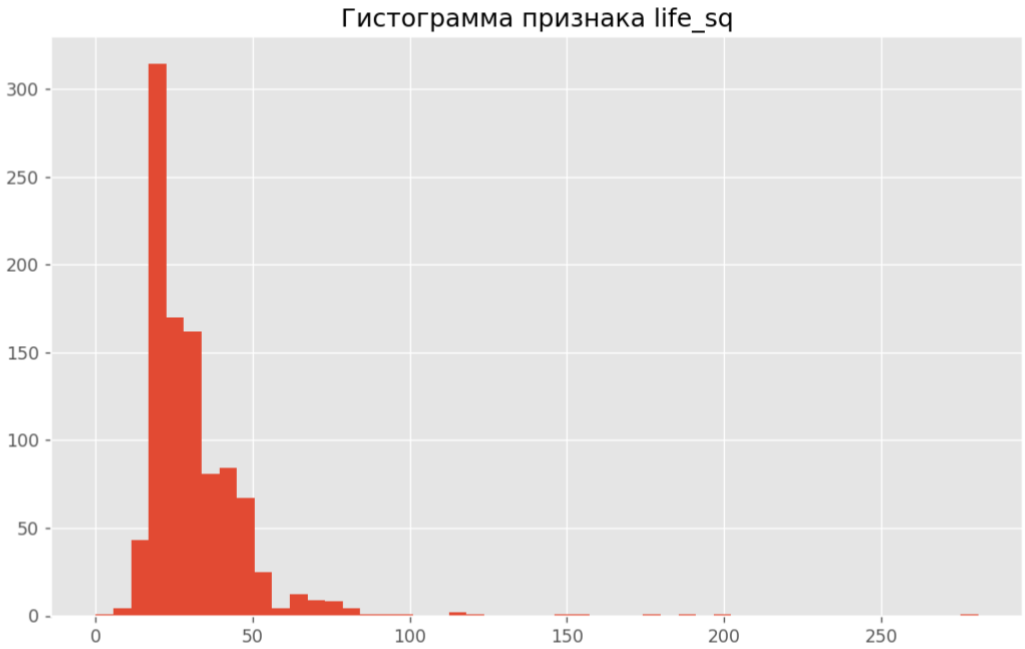
```python
numeric_cols = df.select_dtypes(include=[np.number]).columns

# Гистограммы для первых 5 числовых признаков
for col in numeric_cols[:5]:
    plt.figure(figsize=(10, 6))
    df[col].hist(bins=50)
    plt.title(f'Гистограмма признака {col}')
    plt.xlabel(col)
    plt.ylabel('Частота')
    plt.tight_layout()
    plt.show()

# Коробчатые диаграммы
for col in numeric_cols[:5]:
    plt.figure(figsize=(10, 6))
    df.boxplot(column=[col])
    plt.title(f'Boxplot признака {col}')
    plt.tight_layout()
    plt.show()

# Описательная статистика
print("Описательная статистика числовых признаков:")
print(df[numeric_cols].describe())
```

Гистограммы (анализы распределения) показывают форму распределения, плотность данных в различных диапазонах, наличие разрывов данных.

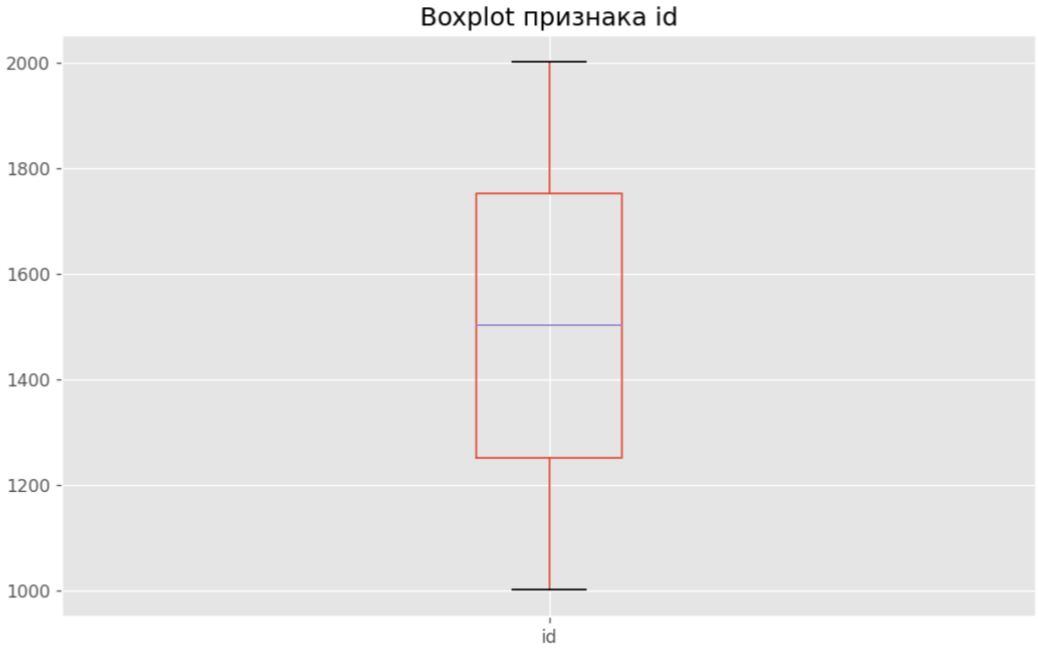
Приведем в пример первые 3 признака и их гистограммы:

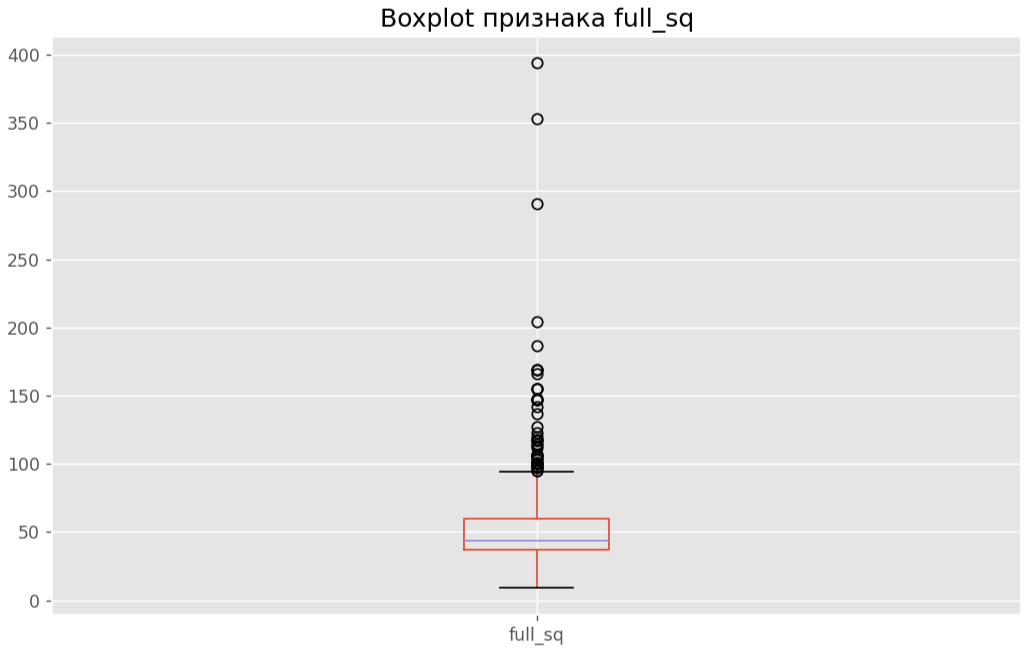






Детекция выбросов показывает медиану и квартили распределения – 3 значения, которые делят упорядоченный набор на равные части. Выбросы – точки за пределами усов.





в) определение ненужных данных (не добавляют ценности для решения данной задачи) - повторяющиеся данные, не имеют отношения к поставленной задаче, слишком много строк с одинаковыми значениями

```
# Поиск неинформативных признаков (более 95% одинаковых значений)
num_rows = len(df)
low_info_cols = []

for col in df.columns:
    cnts = df[col].value_counts(dropna=False, normalize=True)
```

```
    if len(cnts) > 0:
        top_pct = cnts.iloc[0]
        if top_pct > 0.95:
            low_info_cols.append(col)
            print(f'{col}: {top_pct:.2%} одинаковых значений')

# Удаляем неинформативные столбцы
if low_info_cols:
    df = df.drop(columns=low_info_cols)
    print(f"Удалены неинформативные столбцы: {low_info_cols}")

# Удаляем дубликаты строк
initial_shape = df.shape
df = df.drop_duplicates()
final_shape = df.shape

if initial_shape[0] != final_shape[0]:
    print(f"Удалено дубликатов: {initial_shape[0] - final_shape[0]}")
print(f"После удаления дубликатов: {df.shape}")
```

```
max_floor: 100.00% одинаковых значений
material: 100.00% одинаковых значений
build_year: 100.00% одинаковых значений
num_room: 100.00% одинаковых значений
kitch_sq: 100.00% одинаковых значений
state: 100.00% одинаковых значений
product_type: 95.88% одинаковых значений
incineration_raion: 96.28% одинаковых значений
oil_chemistry_raion: 98.99% одинаковых значений
railroad_terminal_raion: 97.08% одинаковых значений
nuclear_reactor_raion: 95.77% одинаковых значений
water_1line: 95.37% одинаковых значений
big_road1_1line: 96.48% одинаковых значений
railroad_1line: 97.99% одинаковых значений
cafe_count_500_price_4000: 95.27% одинаковых значений
cafe_count_500_price_high: 98.49% одинаковых значений
mosque_count_500: 99.40% одинаковых значений
mosque_count_1000: 97.08% одинаковых значений
life_sq_ismissing: 97.99% одинаковых значений
floor_ismissing: 99.50% одинаковых значений
```

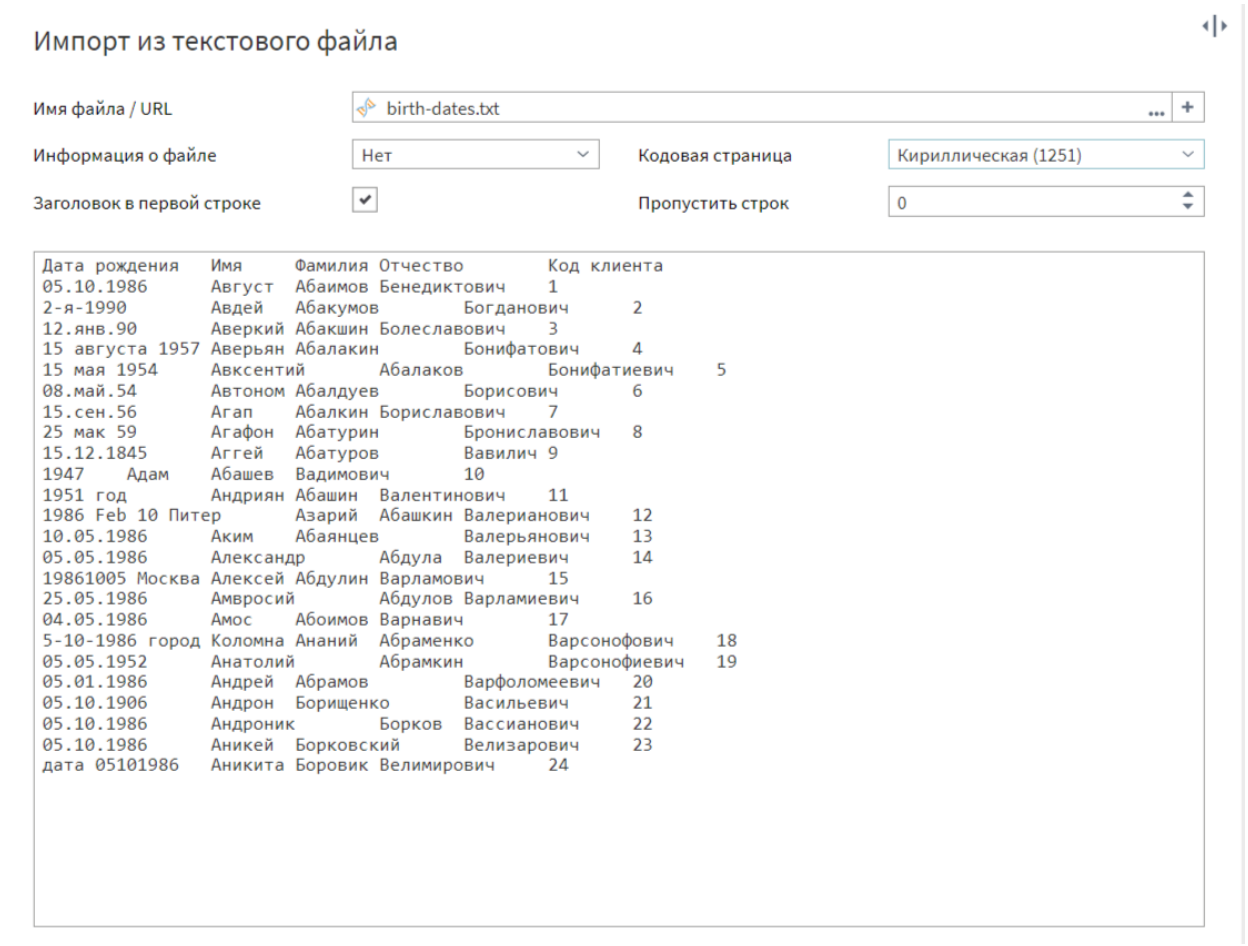
г) несогласованные - данные использование заглавных букв, форматы (время), адреса

```
# Нормализация строковых признаков (нижний регистр)
string_cols = df.select_dtypes(include=['object']).columns
for col in string_cols:
    df[col] = df[col].astype(str).str.lower().str.strip()
```

```python
# Преобразование даты (если есть столбец timestamp)
if 'timestamp' in df.columns:
    df['timestamp'] = pd.to_datetime(df['timestamp'], errors='coerce')
    df['year'] = df['timestamp'].dt.year
    df['month'] = df['timestamp'].dt.month
    df['weekday'] = df['timestamp'].dt.weekday
```

Loginom

Текстовый файл:



Разбор даты рождения:

```
if (day1 <> "", day1,
   RegExMatchedSubExp("\b(19|20)?[0-9]{2}[- /.,]([A-ZA-ЯЁ]{3,})[- /.,](0?[1-9]|[12][0-9]|3[01])\b",
   Trim(Upper(birthdate)), 1)
)
```

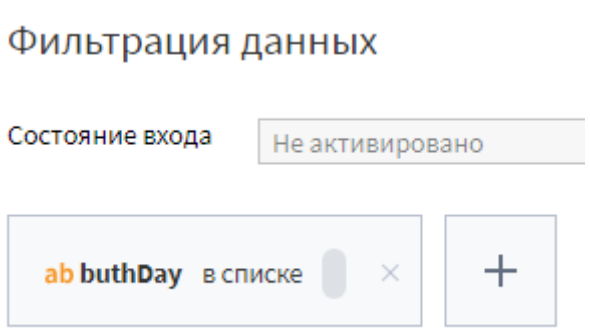
Выходные данные:

Разбор даты рождения — месяц задан строкой • Быстрый просмотр

Выходной набор данных

#	ab d1	ab buthDay	ab m1	ab birthMonth	ab y1	ab birthYear	ab Дата рождения
1							05.10.1986
2							2-я-1990
3	12	12	ЯНВ	ЯНВ	90	90	12.янв.90
4	15	15	АВГУСТА	АВГУСТА	1957	1957	15 августа 1957
5	15	15	МАЯ	МАЯ	1954	1954	15 мая 1954
6	08	08					08.май.54
7	15	15	СЕН	СЕН	56	56	15.сен.56
8	25	25	МАК	МАК	59	59	25 мак 59
9							15.12.1845
10							1947
11							1951 год
12		10		FEB		19	1986 Feb 10 Питер
13							10.05.1986
14							05.05.1986
15							19861005 Москва
16							25.05.1986
17							04.05.1986
18							5-10-1986 город Коломна
19							05.05.1952
20							05.01.1986
21							05.10.1906

Фильтр строк:



Кодировка месяца:

Импорт из текстового файла

Имя файла / URL: /month-code.txt

Информация о файле: Нет

Кодовая страница: Кириллическая (1251)

Заголовок в первой строке: ✔

Пропустить строк: 0

```
Месяц строковый Первые 3 буквы  Номер месяца
ЯНВАРЬ  ЯНВ     01
ФЕВРАЛЬ ФЕВ     02
МАРТ    МАР     03
АПРЕЛЬ  АПР     04
МАЙ     МАЙ     05
ИЮНЬ    ИЮН     06
ИЮЛЬ    ИЮЛ     07
АВГУСТ  АВГ     08
СЕНТЯБРЬ        СЕН     09
ОКТЯБРЬ ОКТ     10
НОЯБРЬ  НОЯ     11
ДЕКАБРЬ ДЕК     12
JANUARY JAN     01
FEBRUARY        FEB     02
MARCH   MAR     03
APRIL   APR     04
MAY     MAY     05
JUNE    JUN     06
JULY    JUL     07
AUGUST  AUG     08
SEPTEMBER       SEP     09
OCTOBER OCT     10
NOVEMBER        NOV     11
DECEMBER        DEC     12
```

Подсоединение из кодировки месяца

Настройка слияния данных

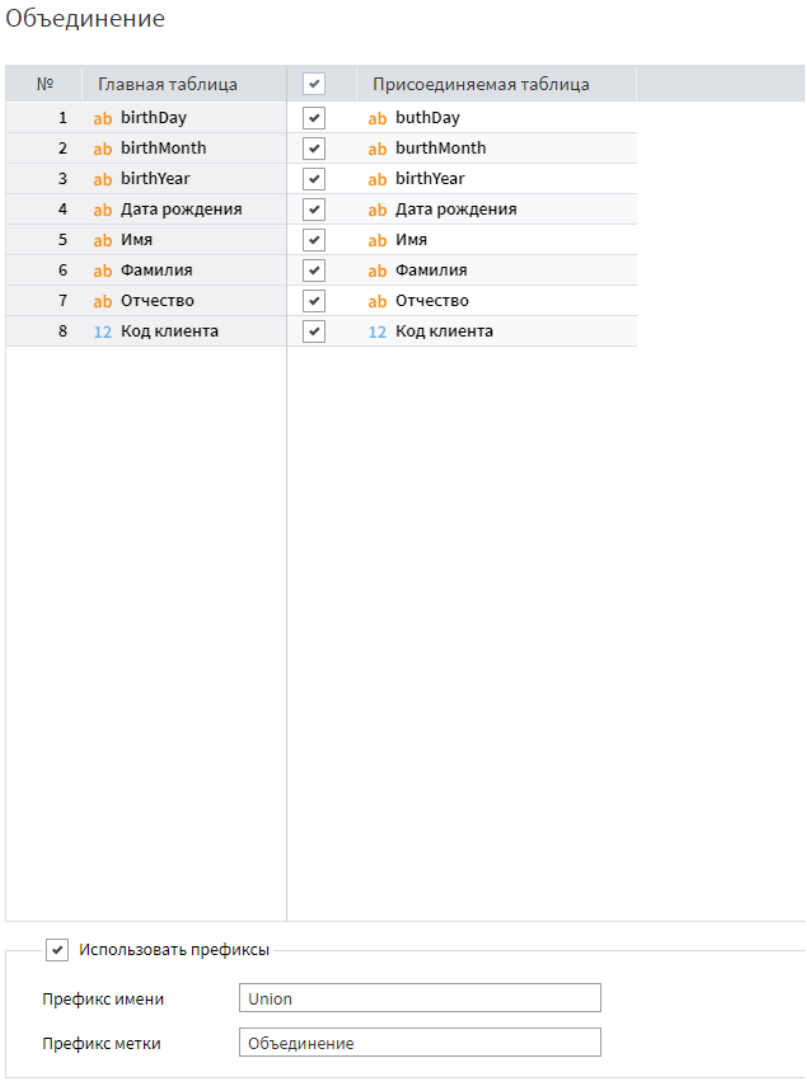
Тип операции: Левое соединение

Фильтрация	Фильтрация
Столбцы основного набора данных	Столбцы присоединяемого набора данных
ab birthMonth	ab Первые 3 буквы
ab birthYear	ab Месяц строковый
ab Дата рождения	12 Номер месяца
ab Имя	
ab Фамилия	
ab Отчество	
12 Код клиента	
ab buthDay	

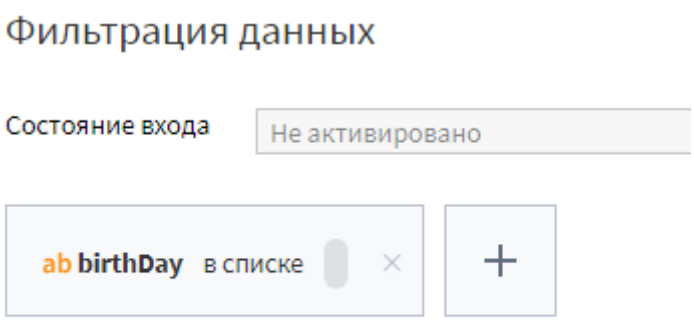
Разбор даты рождения

```
If(
    RegExMatchedSubExp("^\s*([0-3]?[0-9])[- /.]([0-1]?[0-9])[- /.](19|20)?[0-9]{2}\s*$", Trim(Data_rozhdeniya), 1) <> "",
    RegExMatchedSubExp("^\s*([0-3]?[0-9])[- /.]([0-1]?[0-9])[- /.](19|20)?[0-9]{2}\s*$", Trim(Data_rozhdeniya), 1),
    If(
        RegExMatchedSubExp("^\s*([0-1]?[0-9])[- /.]([0-3]?[0-9])[- /.](19|20)?[0-9]{2}\s*$", Trim(Data_rozhdeniya), 2) <> ""
        RegExMatchedSubExp("^\s*([0-1]?[0-9])[- /.]([0-3]?[0-9])[- /.](19|20)?[0-9]{2}\s*$", Trim(Data_rozhdeniya), 2),
        RegExMatchedSubExp("^\s*((19|20)?[0-9]{2})[- /.]([0-1]?[0-9])[- /.]([0-3]?[0-9])\s*$", Trim(Data_rozhdeniya), 4)
    )
)
```

Объединение



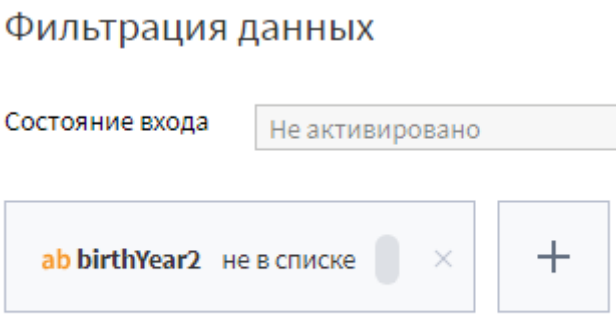
Поделим на разобранные и некорректные



Формат дня и года

```
birthYear > Year(Now())
```

Фильтр строк



Дата в формате

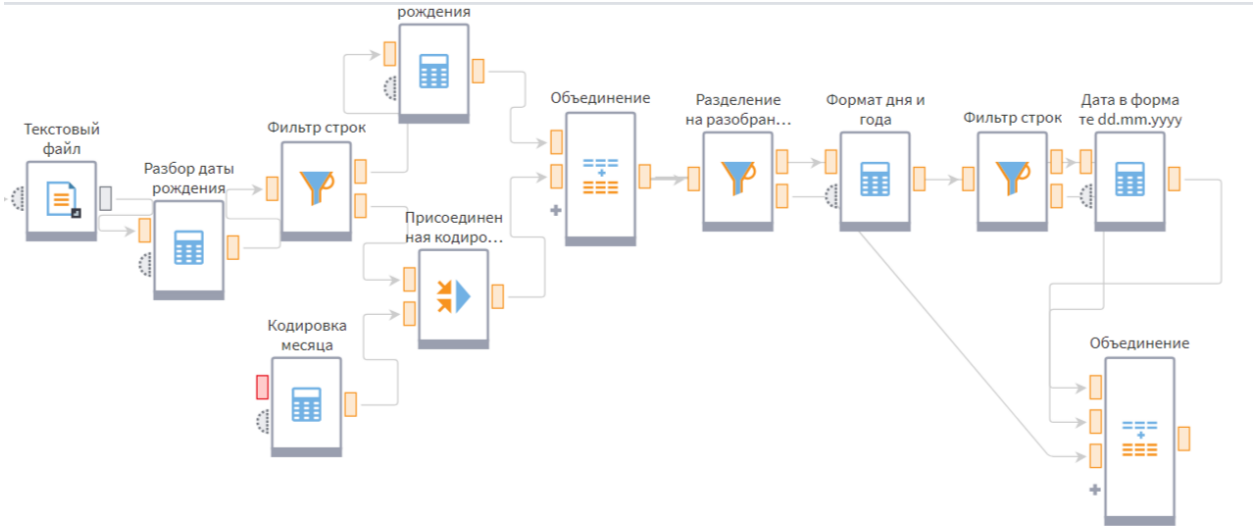
```
Concat(
    birthDay, ".",
    If(
        RegExMatch("^[0-9]{1,2}$", Trim(birthMonth)) = 1,
        Trim(birthMonth),
        IF(Nomer_mesyatsa > 10, Str(Nomer_mesyatsa), CONCAT("0", Str(Nomer_mesyatsa)))
    ),
    ".",
    birthYear
)
```

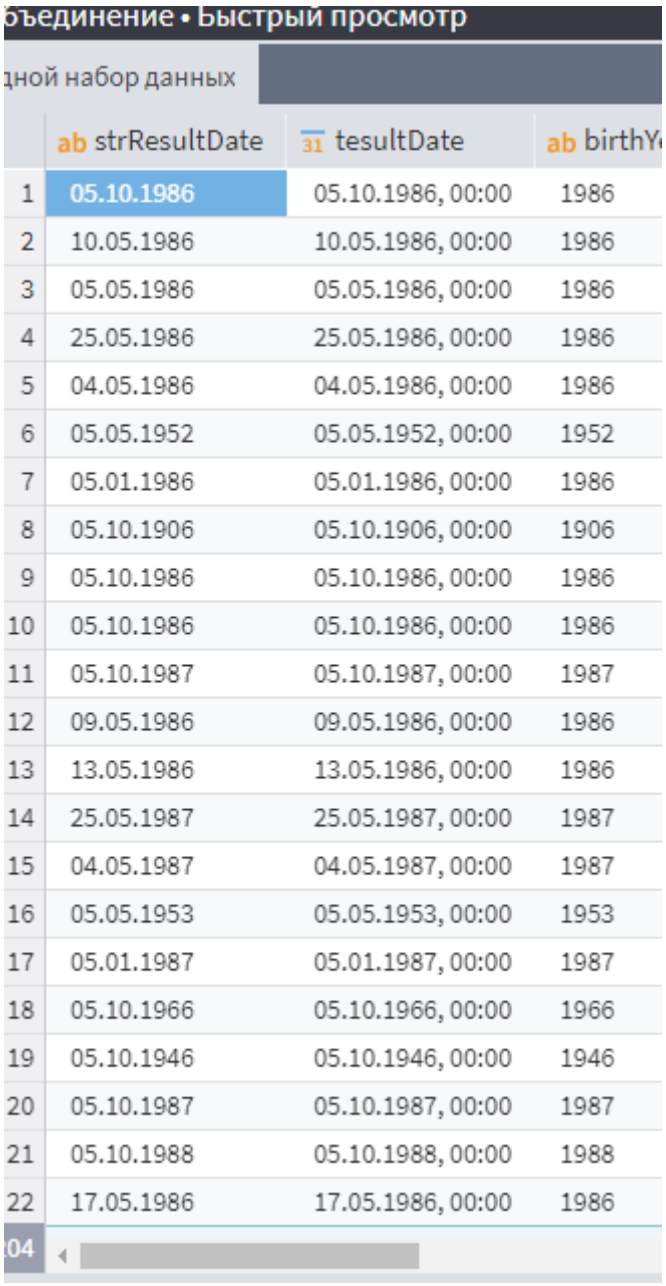
Объединение

Объединение

№	Главная таблица		Присоединяемая таблица		Присоединяемая табли
1	ab strResultDate	☐	Не выбрано	☐	Не выбрано
2	31 tesultDate	☐	Не выбрано	☐	Не выбрано
3	ab birthYear	☐	Не выбрано	☐	Не выбрано
4	ab birthMonth	☐	Не выбрано	☐	Не выбрано
5	ab birthDay	☐	Не выбрано	☐	Не выбрано
6	12 Номер месяца	☐	Не выбрано	☐	Не выбрано
7	ab birthYear2	☐	Не выбрано	☐	Не выбрано
8	0/1 invalidYear	☐	Не выбрано	☐	Не выбрано
9	ab birthDay2	☐	Не выбрано	☐	Не выбрано
10	ab Дата рождения	☐	Не выбрано	☐	Не выбрано
11	ab Имя	☐	Не выбрано	☐	Не выбрано
12	ab Фамилия	☐	Не выбрано	☐	Не выбрано
13	ab Отчество	☐	Не выбрано	☐	Не выбрано
14	12 Код клиента	☐	Не выбрано	☐	Не выбрано

Результат





	ab strResultDate	31 tesultDate	ab birthY
1	05.10.1986	05.10.1986, 00:00	1986
2	10.05.1986	10.05.1986, 00:00	1986
3	05.05.1986	05.05.1986, 00:00	1986
4	25.05.1986	25.05.1986, 00:00	1986
5	04.05.1986	04.05.1986, 00:00	1986
6	05.05.1952	05.05.1952, 00:00	1952
7	05.01.1986	05.01.1986, 00:00	1986
8	05.10.1906	05.10.1906, 00:00	1906
9	05.10.1986	05.10.1986, 00:00	1986
10	05.10.1986	05.10.1986, 00:00	1986
11	05.10.1987	05.10.1987, 00:00	1987
12	09.05.1986	09.05.1986, 00:00	1986
13	13.05.1986	13.05.1986, 00:00	1986
14	25.05.1987	25.05.1987, 00:00	1987
15	04.05.1987	04.05.1987, 00:00	1987
16	05.05.1953	05.05.1953, 00:00	1953
17	05.01.1987	05.01.1987, 00:00	1987
18	05.10.1966	05.10.1966, 00:00	1966
19	05.10.1946	05.10.1946, 00:00	1946
20	05.10.1987	05.10.1987, 00:00	1987
21	05.10.1988	05.10.1988, 00:00	1988
22	17.05.1986	17.05.1986, 00:00	1986
04			