Изменение типов данных

Работа с датафреймами

Чтение таблицы из файла Excel

```
# считываем данные из листа 'Лист 1' файла 'file.xlsx'
df = pd.read_excel('file.xlsx', sheet_name='Лист 1')

# считываем данные из первого по порядку листа файла 'file.xlsx'
df = pd.read_excel('file.xlsx')

# считываем данные из третьего по порядку листа файла 'file.xlsx'
df = pd.read_excel('file.xlsx', sheet_name=2)
```

Объединение двух датафреймов

```
# объединяем данные о посещениях и названия категорий товаров

# посещения из data и категории из subcategory_dict, для которых не нашлось
# соответствия в другой таблице, не попадут в результат
data.merge(subcategory_dict, on='subcategory_id', how='inner')

# Все посещения из data гарантированно останутся. Если для товара не найдётся
# подходящей категории из subcategory_dict, в её названии будет пропуск
data.merge(subcategory_dict, on='subcategory_id', how='left')

# Все категории из subcategory_dict гарантированно останутся. Если для категории
# не найдётся подходящих товаров из data, в названии товара будет пропуск
data.merge(subcategory_dict, on='subcategory_id', how='right')

# все посещения из data и все категории из subcategory_dict гарантированно останутся
data.merge(subcategory_dict, on='subcategory_id', how='outer')
```

Изменение типов данных

Формирование сводной таблицы

```
# index – столбец датафрейма, значения которого будут в строках сводной
# columns – столбец датафрейма, значения которого будут в столбцах сводной
# values – столбец, для значений которого мы считаем сводную
# aggfunc – функция, которая будет применяться к значениям
data_pivot = data.pivot_table(index='column_1', columns='column_2',
values = 'values_column', aggfunc='function')

# В датафрейме data_final содержатся данные об источнике трафика, количестве
# посещений страницы и категории товара. Следующий код строит таблицу, в которой
# в строках будут категории товаров, в столбцах – тип трафика, а в ячейках – сумма
# посещений для определённой категории и типа трафика
data_final.pivot_table(
      index='category_name',
      columns='source',
      values='visits',
      aggfunc='sum')
```

Из результата видно, что в разделе «Дача, сад и огород» было суммарно 26 315 посещений из Директа.

```
source              direct  organic
category_name
Авто                105420   310115
Бытовая техника     318385   789537
Дача, сад и огород   26315    58309
Детские товары      129915   297781
...
```

Работа с Series (столбцами датафрейма)

Перевод значений столбца из строкового типа в числовой (float или int)

```
# errors='raise' (по умолчанию) – при встрече с некорректным значением
# выдается ошибка, операция перевода в числа прерывается;
# errors='coerce' – некорректные значения принудительно заменяются на NaN;
# errors='ignore' – некорректные значения игнорируются, но остаются
data['column'] = pd.to_numeric(data['column'], errors='raise')
```

Изменение типов данных

Перевод значений столбца в произвольный тип данных

```
# приводим столбец 'id' к целочисленному типу
# при неудачной попытке приведения будет ошибка
transactions['id'] = transactions['id'].astype('int', errors='raise')

# приводим столбец 'id' к строковому типу
# при неудаче возвращает исходный столбец без изменений, ошибки не происходит
transactions['id'] = transactions['id'].astype('str', errors='ignore')
```

Перевод из строки в дату и время

```
# обязательный второй аргумент — строка формата даты
# в примере — строки вида "01.04.2019Z11:03:00"
arrivals['date_datetime'] = pd.to_datetime(
    arrivals['date'], format='%d.%m.%YZ%H:%M:%S'
)

'''
Формат строится с использованием следующих обозначений для элементов даты и времени:
• %d — день месяца (от 01 до 31)
• %m — номер месяца (от 01 до 12)
• %Y — год с указанием столетия (например, 2019)
• %H — номер часа в 24-часовом формате
• %I — номер часа в 12-часовом формате
• %M — минуты (от 00 до 59)
• %S — секунды (от 00 до 59)
'''
```

```
# получение из столбца с датой и временем...
pd.DatetimeIndex(data['date_time_column']).year # года
pd.DatetimeIndex(data['date_time_column']).month # месяца
pd.DatetimeIndex(data['date_time_column']).day # дня
pd.DatetimeIndex(data['date_time_column']).hour # часа
pd.DatetimeIndex(data['date_time_column']).minute # минуты
pd.DatetimeIndex(data['date_time_column']).second # секунды
```

Изменение типов данных

Обработка ошибок

```
 # обработка исключений
try:
  # код, где может быть ошибка
except:
  # действия, если возникла ошибка
```

Глоссарий

Unix time — формат времени в виде количества секунд, которые прошли с 00:00:00 1 января 1970 года.

Сводная таблица — инструмент обработки данных, который позволяет обобщить данные по какому-либо признаку.