**Тема занятия № 12: Модуль 7. Выборка данных**

1. Агрегатные вычисления

Агрегатные вычисления затрагивают значения определенного поля всех записей, имеющихся в модели, или групп записей, удовлетворяющих какому-либо условию. К такого рода действиям относится вычисление числа объявлений, среднего арифметического цены, наименьшего и наибольшего значения цены и т.п.

Каждое из возможных действий, выполняемых при агрегатных вычислениях, представляется определенной агрегатной функцией. Так, существуют агрегатные функции для подсчета числа записей, среднего арифметического, минимума, максимума и др.

**Вычисления по всем записям модели**

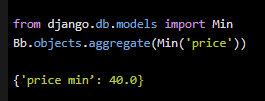
Если нужно провести агрегатное вычисление по всем записям модели, нет ничего лучше метода aggregate ():

Aggregate(<агрегатная функция 1>, <агрегатная функция 2> . . .<агрегатная функция п>)

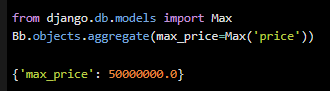
Сразу отметим два момента. Во-первых, агрегатные функции представляются экземплярами особых классов, которые объявлены В модуле django.db.models и которые мы рассмотрим чуть позже. Во-вторых, возвращаемый методом результат — словарь Python, в котором отдельные элементы представляют результаты выполнения соответствующих им агрегатных функций.

Агрегатные функции можно указать в виде как позиционных, так и именованных параметров:

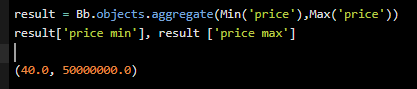
□ если агрегатная функция указана в виде позиционного параметра, то в результирующем словаре будет создан элемент с ключом вида <имя поля, по которому выполняется вычисление>\_\_<имя класса агрегатной функции^ Хранящий результат выполнения этой функции. Для примера определим наименьшее значение цены, указанное в объявлениях:



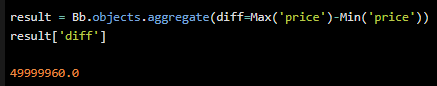
□ если агрегатная функция указана в виде именованного параметра, то ключ элемента, создаваемого в словаре, будет совпадать с именем этого параметра. Выясним наибольшее значение цены в объявлениях:

****

В вызове метода aggregate () допускается задавать произвольное число агрегатных функций:

****

А использовав для указания именованный параметр (с позиционным такой номер не пройдет), — выполнять вычисления над результатами агрегатных функций:

****

**Вычисления по группам записей**

Если нужно провести агрегатное вычисление по группам записей, сформированным согласно определенному критерию (например, узнать, сколько объявлений находится в каждой рубрике), то следует применить метод annotate ():

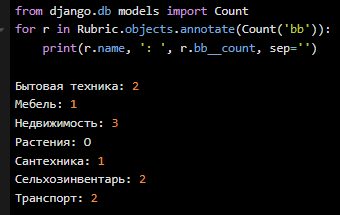
Annotate (<агрегатная функция 1>, <агрегатная функция 2> . . .<агрегатная функция п>)

Вызывается он так же, как и aggregate () , но с двумя отличиями:

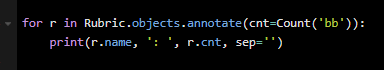
□ в качестве результата возвращается новый набор записей;

□ каждая запись из возвращенного набора содержит атрибут, имя которого генерируется по тем же правилам, что и ключ элемента в словаре, возвращенном методом aggregate о. Этот атрибут хранит результат выполнения агрегатной функции.

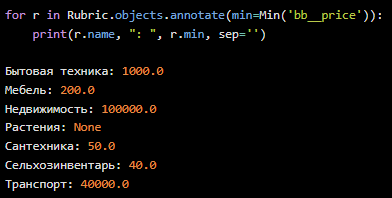
Пример подсчета числа объявлений, оставленных в каждой из рубрик (агрегатная функция указана в позиционном параметре):

****

То же самое, но теперь агрегатная функция указана в именованном параметре:

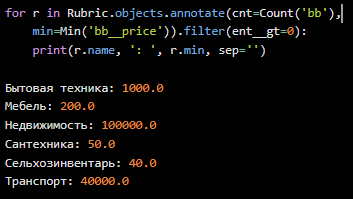
****

Посчитаем для каждой из рубрик минимальную цену, указанную в объявлении:

****

У рубрики ’’Растения”, не содержащей объявлений, значение минимальной цены равно None.

Использовав именованный параметр, мы фактически создаем в наборе записей новое поле (более подробно об этом приеме разговор пойдет позже). Следовательно, мы можем фильтровать записи по значению этого поля. Давайте же уберем из полученного ранее результата рубрики, в которых нет объявлений:

****

**Агрегатные функции**

Все агрегатные функции, поддерживаемые Django, представляются классами из модуля django.db.models.

Конструкторы этих классов принимают ряд необязательных параметров. Указывать их можно лишь в том случае, если сама агрегатная функция задана с помощью именованного параметра, — иначе мы получим сообщение об ошибке.

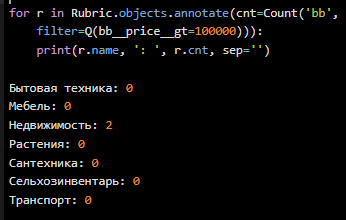
□ Count —вычисляет число записей. Формат конструктора:

Count(<имя поля>[, distinct=False][, filter=None])

Первым параметром указывается имя поля, имеющегося в записях, число которых должно быть подсчитано. Если нужно узнать число записей вторичной модели, связанных с записью первичной модели, то следует указать имя вторичной модели.

Если значение параметра distinct равно True, то будут подсчитываться только уникальные записи, если False — все записи (поведение по умолчанию). Параметр filter указывает условие для фильтрации записей в виде экземпляра класса q; если он не задан, фильтрация не выполняется.

Пример подсчета объявлений, в которых указана цена более 100 000 руб., по рубрикам:

****

□ Sum—вычисляет сумму значений, хранящихся в поле. Формат конструктора:

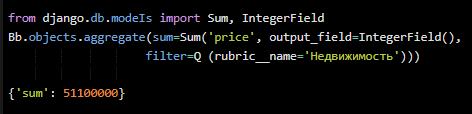
Sum (<имя поля или выражение>[, output\_field=None] [, filter=None] [,Distinct=False])

Первым параметром указывается имя поля, сумма значений которых будет вычисляться, или выражение, представленное экземпляром класса f .

Параметр output field задает тип результирующего значения в виде экземпляра класса, представляющего поле нужного типа. По умолчанию тип результата совпадает с типом поля. Параметр filter указывает условие фильтрации записей в виде экземпляра класса Q; если он не задан, то фильтрация не выполняется.

Параметр distinct поддерживается, начиная с Django 3.0. Если он равен False (по умолчанию), то будут суммироваться все значения заданного поля или выражения. Если True — только уникальные значения.

Пример подсчета суммарной цены всех объектов недвижимости и возврата ее в виде целого числа:

****

□ Min —вычисляет наименьшее значение из хранящихся в заданном поле. Формат конструктора:

М1п(<имя поля или выражение>[г output\_field=None][, filter=None])

Первым параметром указывается имя поля или выражение, представленное экземпляром класса F. Параметр output\_fieid задает тип результирующего значения в виде экземпляра класса, представляющего поле нужного типа. По умолчанию тип результата совпадает с типом поля. Параметр filter указывает условие фильтрации записей в виде экземпляра класса q; если он не указан, то фильтрация не выполняется.

□ мах — вычисляет наибольшее значение из хранящихся в заданном поле. Формат конструктора:

Мах (<имя поля или выражение>[, output\_field=None] [, filter=None])

Первым параметром указывается имя поля или выражение в виде экземпляра класса F. Параметр output field задает тип результирующего значения в виде экземпляра класса, представляющего поле нужного типа. По умолчанию тип результата совпадает с типом поля. Параметр filter указывает условие фильтрации записей в виде экземпляра класса q; если он не задан, то фильтрация не выполняется.

□ Avg —вычисляет среднее арифметическое. Формат конструктора:

Ачд(<имя поля или выражение>[, output\_field=None][, filter=None][,Distinct=False])

Первым параметром указывается имя поля, по содержимому которого будет вычисляться среднее арифметическое, или выражение в виде экземпляра класса F.

Параметр output fieid задает тип результирующего значения в виде экземпляра класса, представляющего поле нужного типа. По умолчанию это объект типа Decimal из модуля decimal Python, если заданное поле принадлежит типу Decimaifieid, и величина вещественного типа— в противном случае. Параметр filter указывает условие фильтрации записей в виде экземпляра класса Q. По умолчанию фильтрация не выполняется.

Параметр distinct поддерживается, начиная с Django 3.0. Если ему дано значение False (по умолчанию), то среднее арифметическое будет рассчитано на основе всех значений заданного поля или выражения, если True — только уникальных значений.

□ stddev —вычисляет стандартное отклонение:

Stddev(<mmff поля или выражение>[, sample=False][, output\_field=None] [,Filter=None])

Первым параметром указывается имя поля, по содержимому которого будет вычисляться стандартное отклонение, или выражение в виде экземпляра класса F.

Если значение параметра sample равно True, то вычисляется стандартное отклонение выборки, если False — собственно стандартное отклонение (поведение по умолчанию). Параметр output fieid задает тип результирующего значения в виде экземпляра класса, представляющего поле нужного типа. По умолчанию это объект типа Decimal из модуля decimal Python, если заданное поле принадлежит типу decimaifieid, и величина вещественного типа— в противном случае. Параметр filter указывает условие фильтрации записей в виде экземпляра класса Q. По умолчанию фильтрация не выполняется.

□ variance — вычисляет дисперсию:

Variance(<имя поля или выражение>[, sample=False][,Output\_field=None][, filter=None])

Первым параметром указывается имя поля, по содержимому которого будет вычисляться дисперсия, или выражение в виде экземпляра класса F. Если значение параметра sample равно True, то вычисляется стандартная дисперсия образца, если False— собственно дисперсия (поведение по умолчанию). Параметр Output field задает тип результирующего значения в виде экземпляра класса, представляющего поле нужного типа. По умолчанию это объект типа Decimal из модуля decimal Python, если заданное поле Принадлежит типу decimaifieid, и величина вещественного типа— в противном случае. Параметр filter указывает условие фильтрации записей в виде экземпляра класса Q.

По умолчанию фильтрация не выполняется.

1. Вычисляемые поля

**Вычисляемые поля**

Значения вычисляемых полей не берутся из базы данных, а вычисляются СУБД на основе других данных. В этом вычисляемые поля подобны функциональным, с тем исключением, что значения последних рассчитываются самим Django.

**Простейшие вычисляемые поля**

В самом простом виде вычисляемое поле создается с помощью метода annotate (), который нам уже знаком. В нем, с применением именованного параметра, указывается выражение, вычисляющее значение поля.

□ Для указания значения какого-либо поля применяется экземпляр класса f.

□ Константы целочисленного, вещественного и логического типов можно указывать как есть.

□ Для записи константы строкового типа используется экземпляр класса value из модуля django.db.models, конструктор которого вызывается в формате:

Value (<значение константы> [ , output\_field=None] )

Само значение константы записывается в первом параметре конструктора этого класса.

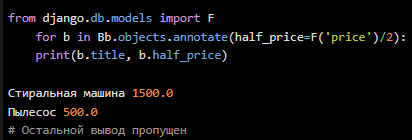
В необязательном параметре output\_fieid можно задать тип константы в виде экземпляра класса, представляющего поле нужного типа. Если параметр отсутствует, тип значения будет определен автоматически.

□ Для вычислений применяются операторы +, -, \*, / и //, записываемые как есть.

Они выполняют соответствующие действия над переданными им экземплярами классов f и value и возвращают результат этих действий также в виде экземпляра класса F.

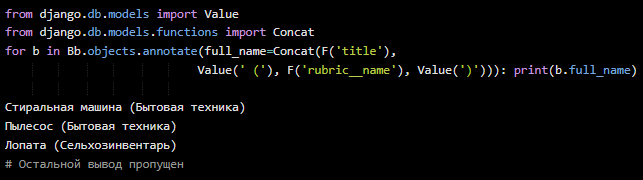
Оператор - позволяет изменить знак значения, представленного экземпляром класса f или value.

В качестве примера вычислим для каждого объявления половину указанной в нем цены:

****

Константа 2 здесь указана как есть, поскольку она принадлежит целочисленному типу, и Django благополучно обработает ее.

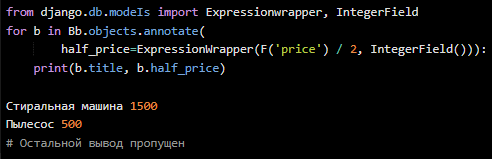
Выведем записанные в объявлениях названия товаров и, в скобках, названия рубрик:

****

В некоторых случаях может понадобиться указать для выражения, записанного виде экземпляра класса F, тип возвращаемого им результата. Непосредственно в конструкторе класса f это сделать не получится. Положение спасет класс Expressionwrapper ИЗ модуля django.db.models. Вот формат его конструктора:

Expressionwrapper(<выражение>, <тип результата^

Выражение представляется экземпляром класса F, а тип данных— экземпляром класса поля нужного типа. Пример:

****

**Функции СУБД**

Функция СУБД обрабатывается не Django и не Python, а СУБД. Django просто предоставляет для этих функций удобный объектный интерфейс.

Функции СУБД используются в вызовах метода annotate и представляются следующими классами ИЗ модуля django.db.models.functions:

□ Coalesce — возвращает первое переданное ему значение, отличное от null (даже если это пустая строка или 0). Конструктор класса вызывается в формате:

Coalesce (Оначение 1>, Оначение 2> . . . Оначение п>)

Значения представляются строковыми именами полей, экземплярами классов f и Value. Все они должны иметь одинаковый тип (например, только строковый или только числовой), в противном случае мы получим ошибку.

Пример:



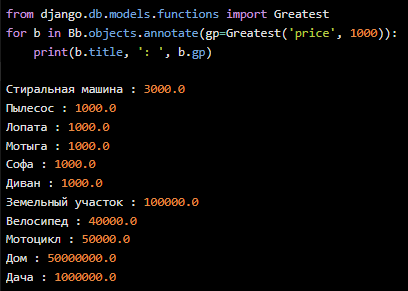
Если значение поля content отлично от null, то будет возвращено оно. В противном случае будет проверено значение поля addendum и, если оно не равно Null, функция вернет его. Если же и значение поля addendum равно null, то будет возвращена константа ' —пусто—';

□ Greatest —возвращает наибольшее значение из переданных ему:

Greatest (Оначение 1>, Оначение 2> . . . Оначение п>)

Значения представляются строковыми именами полей, экземплярами классов f и Value. Все они должны иметь одинаковый тип (например, только строковый или только числовой), в противном случае мы получим ошибку.

Пример:

****

□ Least — возвращает наименьшее значение из переданных ему:

Least(<значение 1>, <значение 2> . . . <значение п>)

Значения представляются строковыми именами полей, экземплярами классов f и value. Все они должны иметь одинаковый тип;

□ cast — принудительно преобразует заданное значение к указанному типу и возвращает результат преобразования:

Cast(<значение>, <тип>)

Значение представляется строкой с именем поля или экземпляром класса F. Тип должен указываться в виде экземпляра класса, представляющего поле соответствующего типа;

□ Concat — объединяет переданные ему значения в одну строку, которая и возвращается в качестве результата:

Concat (<значение 1>, <значение 2> . . . <значение п>)

Значения представляются строковыми именами полей, экземплярами классов f и value. Все они должны иметь строковый или текстовый тип;

□ Lower — преобразует символы строки к нижнему регистру и возвращает преобразованную строку в качестве результата:

Lower(<значение>)

Значение представляется строкой с именем поля или экземпляром класса F. Оно должно иметь строковый или текстовый тип;

□ upper — преобразует символы строки к верхнему регистру и возвращает преобразованную строку в качестве результата:

Upper(<значение>)

Значение представляется строкой с именем поля или экземпляром класса F. Оно должно иметь строковый или текстовый тип;

□ Length — возвращает длину полученного значения в символах:

Length (<значение>)

Значение представляется строкой с именем поля или экземпляром класса F. Оно должно иметь строковый или текстовый тип. Если значение равно null, то возвращается None;

□ strindex— возвращает номер вхождения указанной подстроки в строковое значение. Нумерация символов в строке начинается с 1. Если подстрока отсутствует в значении, возвращается 0. Формат конструктора:

Strindex(<значение>, <подстрока>)

Значение и подстрока представляются строками с именами поля или экземплярами классов F или value. Они должны иметь строковый или текстовый тип;

□ Substr— извлекает из значения подстроку с указанными позицией первого символа и длиной и возвращает в качестве результата:

Substr(<значение>, <позиция>[, <длина>])

Значение представляется строкой с именем поля или экземпляром класса F. Оно должно иметь строковый или текстовый тип.

При указании позиции и длины извлекаемой подстроки нужно иметь в виду, что нумерация символов в строке начинается с 1. Если длина не задана, извлекается ВСЯ Оставшаяся часть значения;

□ Left— возвращает подстроку, начинающуюся с начала заданного значения и имеющую заданную длину:

Left(<значение>, <длина>)

□ Right — возвращает подстроку, заканчивающуюся в конце заданного значения и имеющую заданную длину:

Right(<значение>, <длина>)

□ Replace— возвращает значение, в котором вместо всех вхождений заменяемой подстроки подставлена заменяющая подстрока:

Replace (<значение>, <заменяемая подстрока>[, <заменяющая подстрока>])

Если заменяющая подстрока не указана, используется пустая строка (т.е. Функция фактически будет удалять все вхождения заменяемой подстроки). Поиск заменяемой подстроки выполняется с учетом регистра символов;

□ Repeat —возвращает заданное значение, повторенное заданное число раз:

Repeat(<значение>, <число повторов>)

□ lpad— возвращает заданное значение, дополненное слева символами-заполнителями таким образом, чтобы достичь указанной длины:

Lpad (<значение>, <длина>[, <символ-заполнитель>])

Если символ-заполнитель не указан, то используется пробел;

□ rpad— возвращает заданное значение, дополненное справа символами-заполнителями таким образом, чтобы достичь указанной длины:

Rpad (<значение>, <длина> [, <символ-заполнитель>])

Если символ-заполнитель не указан, то используется пробел;

□ Trim— возвращает указанное значение с удаленными начальными и конечными пробелами:

Тгim (<значение>)

□ ltrim—возвращает указанное значение с удаленными начальными пробелами:

Ltrim (<значение>)

□ rtrim— возвращает указанное значение с удаленными конечными пробелами:

Rtrim (<значение>)

□ chr —возвращает символ с указанным целочисленным кодом.

Chr(<код символа>)

□ Ord — возвращает целочисленный код первого символа заданного значения’.

Ord(<значение>)

□ Now () — возвращает текущие дату и время;

□ Extract— извлекает часть значения даты и времени и возвращает его в виде числа:

Extract (<значение даты и (или) времени>,

Извлекаемая часть значения>[, tzinfo=None] )

Значение представляется экземпляром класса f и должно принадлежать типу даты, времени или временной отметки.

Извлекаемая часть даты и времени представляется в виде строки "year" (год),

"quarter" (номер Квартала), "month" (номер месяца), "day" (число), "week" (порядковый номер недели), "week day" (номер дня недели), "hour" (часы), "minute" (минуты) ИЛИ "second" (секунды).

Параметр tzinfo задает временную зону. На практике он указывается крайне редко.

Пример:

Extract('published', 'year')

Класс Extract можно заменить более простыми в применении классами:

Extractyear (извлекает год), extractisoyear (год в формате ISO-8601, начиная с Django 2.2), extractquarter (номер квартала), extractmonth (номер месяца), Extractday (ЧИСЛО), extractweek (порядковый номер недели), extractweekday (номер ДНЯ недели), extracthour (часы), extractminute (минуты) И extractsecond (секунды). Формат вызова конструкторов этих классов:

<класс>(<значение даты и (или) времени>[, tzinfo=None] )

Пример:

Extractyear(’published’)

□ Trunc — сбрасывает В НОЛЬ значение даты и (или) времени ДО указанной конечной части, если считать справа:

Trunc (<значение даты и (или) времени>, конечная часть>[, Output\_field=None][, tzinfo=None][, is\_dst=None])

Значение представляется экземпляром класса f и должно принадлежать типу даты, времени или временной отметки.

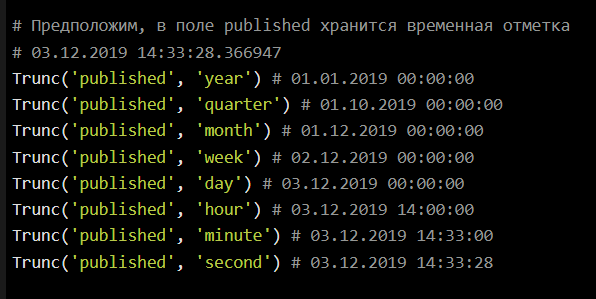
Конечная часть Представляется В виде строки "year” (год), "quarter" (номер квартала), "month" (номер месяца), "week" (неделя), "day" (число), "hour" (часы), "minute" (минуты) ИЛИ "second" (секунды).

Параметр output field указывает тип возвращаемого значения даты и (или) времени. Его значение должно представлять собой экземпляр класса datefieid, Timefieid или datetimefield. Если он не указан, то тип возвращаемого результата будет совпадать с типом изначального значения.

Параметр tzinfo задает временную зону. На практике он указывается крайне редко.

Параметр is\_dst поддерживается, начиная с Django 3.0. Если ему задать значение None (по умолчанию) или False, то коррекция летнего времени проводиться не будет, что в некоторых случаях может вызвать возникновение ошибок и возбуждение исключений ambiguoustimeerror ИЗ МОДУЛЯ pytz. Exceptions. ЕСЛИ задано значение True, то коррекция летнего времени будет проводиться.

Примеры:

****

Вместо класса Trunc можно использовать более простые в применении классы:

Truncyear (сбрасывает ДО года), truncquarter (ДО квартала), truncmonth (до месяца), truncweek (ДО ПОЛУНОЧИ понедельника текущей недели), truncday (до числа), Trunchour (ДО часов), truncminute (ДО минут), truncsecond (ДО Секунд), truncdate (извлекает значение даты) и trunctime (извлекает значение времени). Формат вызова конструкторов этих классов:

<класс>(<значение даты и (или) времени>[, output\_field=None] [, tzinfo=None])

Пример:



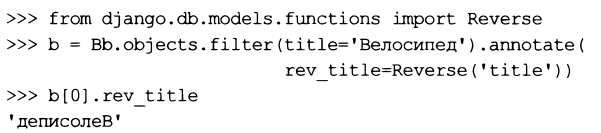
Поддержка следующих функций появилась в Django 2.2:

□ Reverse--- возвращает заданное значение. В котором символы выстроены в обратном порядке:

Reverse(<значение>)

Значение представляется строковым именем поля или экземпляром класса F. Оно должно быть строкового или текстового типа.

Пример:

****

□ Nullif —возвращает None, если заданные значения равны, и значение 1 — в противном случае:

Nulllf (Оначение 1>, Оначение 2>)

Значения представляются строковыми именами полей, экземплярами классов f ИЛИ Value;

О sqrt —возвращает квадратный корень значения:

Sqrt (Оначение>)

Значение представляется строковым именем поля или экземпляром класса F. Оно должно быть целочисленного или вещественного типа;

□ Mod —возвращает остаток от целочисленного деления значения 1 на значение 2:

Mod (Оначение 1>, Оначение 2>)

Значения представляются строковыми именами полей или экземплярами класса F. Они должны быть целочисленного или вещественного типа;

□ Power —возвращает результат возведения в степень:

Power(<основание>, <показатель>)

Основание и показатель представляются строковыми именами полей или экземплярами класса f. Они должны быть целочисленного или вещественного типа;

□ Round— округляет заданное значение до ближайшего целого и возвращает результат:

Round (Оначение>)

Значение представляется строковым именем поля или экземпляром класса F. Оно должно быть целочисленного или вещественного типа;

□ Floor— округляет заданное значение до ближайшего меньшего целого и возвращает результат:

F1оог(<значение>)

Значение представляется строковым именем поля или экземпляром класса F. Оно должно быть целочисленного или вещественного типа;

□ ceil — округляет заданное значение до ближайшего большего целого и возвращает результат:

Сеi 1 (<значение>)

Значение представляется строковым именем поля или экземпляром класса F. Оно должно быть целочисленного или вещественного типа;

□ Pi () — возвращает значение числа я. Вызывается без параметров;

□ Abs —возвращает абсолютное значение, вычисленное от заданного значения:

Abs (<значение>)

Значение представляется строковым именем поля или экземпляром класса F. Оно должно быть целочисленного или вещественного типа;

□ sin — возвращает синус значения. Заданного в радианах:

Sin(<значение>)

Значение представляется строковым именем поля или экземпляром класса F. Оно должно быть целочисленного или вещественного типа;

□ cos — возвращает косинус значения, заданного в радианах:

Cos(<значение>)

Значение представляется строковым именем поля или экземпляром класса F. Оно должно быть целочисленного или вещественного типа;

□ тап —возвращает тангенс значения, заданного в радианах:

Tan(<значение>)

Значение представляется строковым именем поля или экземпляром класса F. Оно должно быть целочисленного или вещественного типа;

□ cot —возвращает котангенс значения, заданного в радианах:

Cot(<значение>)

Значение представляется строковым именем поля или экземпляром класса F. Оно должно быть целочисленного или вещественного типа;

□ asin —возвращает арксинус значения:

Asin(<значение>)

Значение представляется строковым именем поля или экземпляром класса F. Оно должно быть целочисленного или вещественного типа и находиться в диапазоне от-1 до 1;

□ acos — возвращает арккосинус значения:

Acos(<значение>)

Значение представляется строковым именем поля или экземпляром класса F. Оно должно быть целочисленного или вещественного типа и находиться в диапазоне от-1 до 1;

□ атап —возвращает арктангенс значения:

Атап(<значение>)

Значение представляется строковым именем поля или экземпляром класса F. Оно должно быть целочисленного или вещественного типа;

□ дтап2 —возвращает арктангенс от частного от деления значения 1 на значение 2:

Атап2 (<значение 1>, <значение 2>)

Значения представляются строковыми именами полей или экземплярами класса F. Они должны быть целочисленного или вещественного типа;

□ Radians — преобразует заданное значение из градусов в радианы и возвращает результат:

Padians(<значение>)

Значение представляется строковым именем поля или экземпляром класса F. Оно должно быть целочисленного или вещественного типа;

□ Degrees — преобразует заданное значение из радианов в градусы и возвращает результат:

Degrees(<значение>)

Значение представляется строковым именем поля или экземпляром класса F. Оно должно быть целочисленного или вещественного типа;

□ Ехр —возвращает результат вычисления экспоненты от значения:

Ехр(<значение>)

Значение представляется строковым именем поля или экземпляром класса F. Оно должно быть целочисленного или вещественного типа;

□ Log —возвращает логарифм от значения по заданному основанию:

Log(<основание>, <значение>)

Основание и значение представляются строковыми именами полей или экземплярами класса f. Они должны быть целочисленного или вещественного типа;

□ Ln —возвращает натуральный логарифм от значения:

Ln(<значение>)

Значение представляется строковым именем поля или экземпляром класса f.Oho должно быть целочисленного или вещественного типа.

Поддержка следующих функций появилась в Django 3.0:

□ sign— возвращает -1, если значение отрицательное, 0, если равно нулю, и 1, если положительное:

S ign(<значение>)

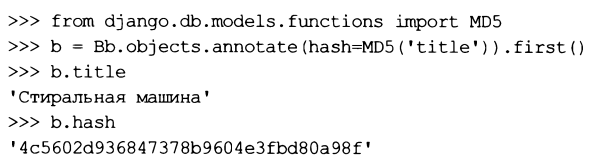
Значение представляется строковым именем поля или экземпляром класса F. Оно должно быть целочисленного или вещественного типа;

□ MD5 —возвращает хэш значения, вычисленный по алгоритму MD5:

. MD5(<значение>)

Значение представляется строковым именем поля или экземпляром класса F. Оно должно быть строкового или текстового типа.

Пример:

****

□ shai — возвращает хэш значения, вычисленный по алгоритму SHA1:

SHA1 (<значение>)

Значение представляется строковым именем поля или экземпляром класса F. Оно должно быть строкового или текстового типа;

□ SHA244 —возвращает хэш значения, вычисленный по алгоритму SHA244:

SHA244(<значение>)

Значение представляется строковым именем поля или экземпляром класса F. Оно должно быть строкового или текстового типа.

Внимание!

Oracle не поддерживает функцию SHA244.

□ SHA256 —возвращает хэш значения, вычисленный по алгоритму SHA256:

SHA25 6(<значение>)

Значение представляется строковым именем поля или экземпляром класса F. Оно должно быть строкового или текстового типа;

□ SHA384 —возвращает хэш значения, вычисленный по алгоритму SHA384:

SHA384 (<значение>)

Значение представляется строковым именем поля или экземпляром класса F. Оно должно быть строкового или текстового типа;

□ SHA512 —возвращает хэш значения, вычисленный по алгоритму SHA512:

SHA512(<значение>)

Значение представляется строковым именем поля или экземпляром класса F. Оно должно быть строкового или текстового типа.

Внимание!

Для использования функций shai, SHA244, SHA256, SHA384 и SHA512 в postgresql следует установить расширение pgcrypto. Процесс его установки описан в документации по этой СУБД.

1. Объединение наборов записей

Для объединения нескольких наборов записей в один применяется метод union ():

Union(<набор записей 1>, <набор записей 2> . . . <набор записей п>[,All=False])

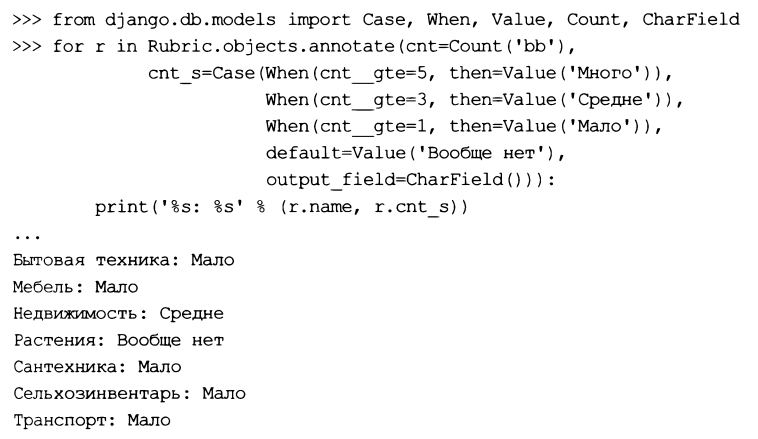
Все заданные наборы записей будут объединены с текущим, и получившийся набор будет возвращен в качестве результата.

Если значение необязательного параметра ail равно True, то в результирующем наборе будут присутствовать все записи, в том числе и одинаковые. Если же для параметра указать значение False, результирующий набор будет содержать только уникальные записи (поведение по умолчанию).

Внимание!

У наборов записей, предназначенных к объединению, не следует задавать сортировку. Если же таковая все же была указана, нужно убрать ее, вызвав метод order byo без параметров.

Для примера сформируем набор из объявлений с заявленной ценой более 100 000 руб. И объявлений по продаже бытовой техники:

****

Django поддерживает два более специализированных метода для объединения наборов записей:

□ intersection(<набор записей 1>, <набор записей 2> . . . <набор записей п>) — возвращает набор, содержащий только записи, которые имеются во всех объединяемых наборах;

□ difference (<набор записей 1>, Снабор записей 2> . . . <набор записей п>) — возвращает набор, содержащий только записи, которые имеются лишь в каком-либо одном из объединяемых наборов, но не в двух или более сразу.

1. Извлечение значений только из заданных полей

**Извлечение значений**

Только из заданных полей каждый из ранее описанных методов возвращает в качестве результата набор записей— последовательность объектов, представляющих записи. Такие структуры данных удобны в работе над целыми записями, однако отнимают много системных ресурсов.

Если необходимо извлечь из модели только значения определенного поля (полей) хранящихся там записей, удобнее применить следующие методы:

□ values ([ <поле 1>, <поле 2> . . . <поле п>])—извлекает из модели значения только указанных полей. Возвращает набор записей (экземпляр класса Queryset), элементами которого являются словари. Ключи элементов таких словарей совпадают с именами заданных полей, а значения элементов — это и есть значения полей.

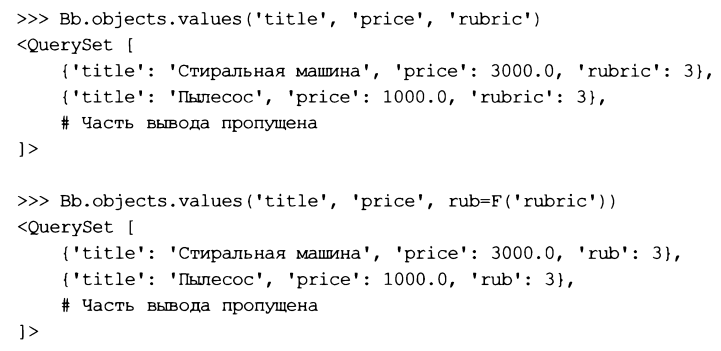
Поле может быть задано:

• позиционным параметром — в виде строки со своим именем;

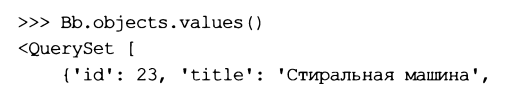
• именованным параметром — в виде экземпляра класса F. Имя параметра станет именем поля.

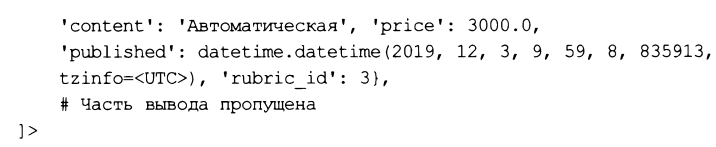
Если в числе полей, указанных в вызове метода, присутствует поле внешнего ключа, то элемент результирующего словаря, соответствующий этому полю, будет хранить значение ключа связанной записи, а не саму запись.

Примеры:

****

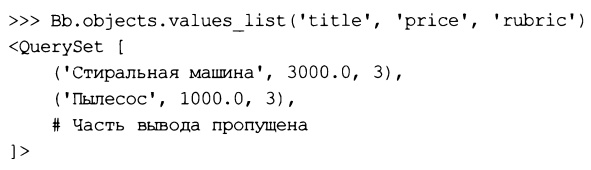
Если метод values о вызван без параметров, то он вернет набор словарей со всеми полями модели. При этом вместо полей модели он будет содержать поля таблицы из базы данных, обрабатываемой этой моделью. Вот пример (обратим внимание на имена полей — это поля таблицы, а не модели):



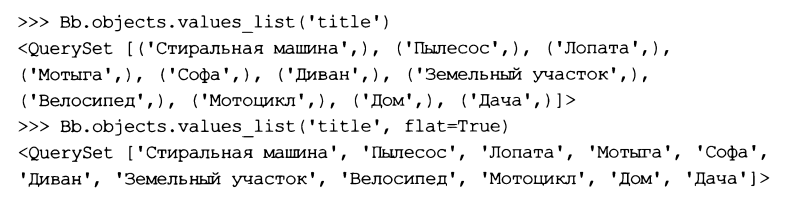


□ values\_list ( [<поле 1>, <поле 2> . . . <поле n>] [,] [flat=False] [,]

[named=Faise]) — то же самое, что values (), но возвращенный им набор записей будет содержать кортежи:

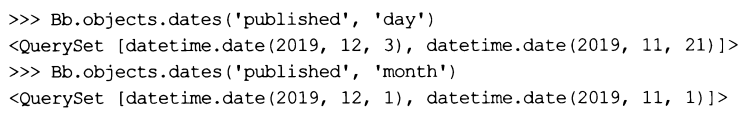
****

Параметр flat имеет смысл указывать только в случае, если возвращается значение одного поля. Если его значение— False, то значения этого поля будут оформлены как кортежи из одного элемента (поведение по умолчанию). Если же задать для него значение True, возвращенный набор записей будет содержать значения поля непосредственно. Пример:

****

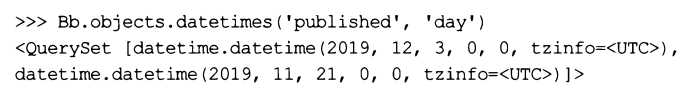
Если параметру named дать значение True, то набор записей будет содержать не обычные кортежи, а именованные;

□ dates (<имя поля>, <часть даты>[, order='ASC']) — возвращает набор записей (экземпляр класса Queryset) с уникальными значениями даты, которые присутствуют в поле с заданным именем и урезаны до заданной части, В качестве части латы можно указать "year" (год), "month" (месяц) ИЛИ "day" (число, т. Е. Дата не будет урезаться). Если параметру order дать значение "asc", то значения в наборе записей будут отсортированы по возрастанию (поведение по умолчанию), если "desc" — по убыванию. Примеры:



□ datetimes (<имя поля>, <часть даты и времени>[, order=’ASC' ] [, tzinfo=None]) — то же самое, что dates о, но манипулирует значениями временных отметок. В качестве части даты и времени МОЖНО указать "year" (год), "month" (месяц), "day" (число), "hour" (часы), "minute” (минуты) или "second" (секунды, т.е. Значение не будет урезаться). Параметр tzinfo указывает временную зону.

Пример:

****

□ in\_bulk^последовательность значении>[, field\_name=’pk’ ]) — ищет в модели записи, у которых поле с именем, заданным параметром field name, хранит значения из указанной последовательности. Возвращает словарь, ключами элементов которого станут значения из последовательности. А значениями элементов — объекты модели, представляющие найденные записи. Заданное поле должно хранить уникальные значения (параметру unique конструктора поля модели нужно задать значение True). Примеры:

