ORM module 5 - part 3

Annotating objects.

* Annotating object help me add new field to objects while quiring them.
* If I try to add primitive data I will get error non-expression error.
  + It means the value of new field should be expression or derived from expression class.
    - These classes are func, Value, F, Aggregate etc…
  + How To use it:
    - Call method from object manager and it will return me all objects with new field added.
    - Then I pass new\_field\_name= Expressesion(val).
      * Value(‘value’), will add string value to all objects in new\_field\_name
      * Or I can use reference to refer to other field values.
        + I can do some calculations on it.
      * Model.objects.annotate(new\_field\_value = F(‘referenced\_field’))

Calling Database functions

* I can use database functions to obtain desired results, for example perform some math or combine strings.
* To do that lets say I want to create a new value e.g., full name of a customer.
  + To do that I need to use concat function.
* So, example of calling a db function:
  + Annotate (full\_name = fun( ‘field\_1’, Value(‘ ’ ),‘field\_2’, function=’FUNCTIONNAME’ ) )
  + Annotate (full\_name = fun( ‘first\_name, Value(‘ ’ ),‘last\_name’, function=’CONCAT’ ) )
* Other method of calling them:
  + Annotate( full\_name = Concat (( ‘first\_name, Value(‘ ’ ),‘last\_name’, ))
  + I don’t need function keyword because I used concat.

Grouping data.

* I can use annotate() to group data.
* For example I can get promotions are made on product.
* To do that I use .annotate(product\_promotions= count(‘related\_field’))

ExpressionWrapper:

* I use this wrapper to perform complex queries.
* For example, I want to multiply product price with value to make a discount.
  + If I try to multiply a decimal with float I get mixed type error.
  + To solve this kind of problems I do the following.
* First I import ExpressionWrapper() and initate an object.
  + Inside the object I add the reference field and perform a calculation.
  + Separate it with comma and define output field as error tell us.
  + The value of output\_field should be an object call e.g., DecimalField() from database models.
  + Then I store it inside a variable.
  + Give the new field inside annotate the value of expression generated.

CONTENT TYPE MODELS

* Earlier in course, I was introduced to content type where I can decouple some option models and optionally, I use them inside the application.
* In order to do that I made another as child model and
  + added a foreign key for the parent class Tag was parent and TaggedItem was child.
  + Then I added a generic content type to refer to the model I want it to work with content type.
  + A foreign key to the object id (the other model).
  + Content object that contain the object data.
* How to use?
* In views file where I want to use the generic content type with any model I want to use it with.
  + 1st I import contenttype, other model, and the model that contains these relations:
    - From Django.contrib.contenttype.models import ContentType
    - From storefront.models import product.
    - From tags.models import taggedItem.
  + Then I make a query to get the id of the other class using the contenttype class.
    - Query = ContentType.objects.get\_for\_model(Product)
  + Then I query inside the child model that contains all these relation then I filter by Product id(inside the contenttype table can be fetched by modelName) and the id of the object I want from that class
    - TaggedItem.objects.filter( content\_type= Query, object\_id=1 )
    - This will filter me first based on content model id which is 11 inside db then from that 11 will get me the object\_id that is 1.
  + And to not end up querying over and over to parent class I can query the parent class of Taggeditem which is tag using .select\_related(‘parent’) before filtering.

Building custom object managers.

* To do that I first need to create a class that inherit from models.Manager
  + Inside of it I define method I want it to perform a query, in my case I wanted to add custom manager for content type.
  + I define a method take 3 parameters self, Model and object id.
    - Inside method I get the id of the model inside contenttype table.
    - Then I query the taggedItem and filter it by content type and object id.
  + Inside taggeditem I create objects variable and initiate an instance from my custom manager class.
  + Then I can use that object with tagged items.

Inserting objects into db got 3 methods.

1. Initiate Model constructor and pass parameters to it.
2. Call create from objects and pass key = pair values to it.
3. Initiate an object from Model then access each field with dot notation.
   1. In case of foreign key, I need the parent id, to do it I get field\_name\_id
4. Object.save(). // will execute the method for me.

Update objects

There are 3 methods to update my object.

1. To initate and object and provide it a primary key Model(pk=11)
   1. Then using object notation to access field I want to update it’s value.
      1. Problem with this method is that the Django will store the entire object in memory.
      2. This will result to losing other fields values.
      3. Collection = Collection(pk=11)

Collection.featured\_product = null.

* 1. Update object using query first to get pk then update it using get method and filter.
     1. Collection.objects.get(pk=1)
     2. Collection.featured\_product = null.
     3. This way will keep my object values.
  2. Using update from object manager, but I need to use a filter before updating.
     1. Colelction.objects.filter(pk=1).update(featured\_product=none).

1. Then call save () on object

Delete object

1. To delete one object I initiate an object and provide it a primary key then call method.delete()
2. To delete list of items I call object manager and filter then delete.

Transaction

* In some cases, I need to execute a block of code all at once and if one step fail, none of the code should be executed.
  + For example, I want to create cart and add cart items to it.
  + If something wrong occurs, I don’t want to end up having cart item without item.
* Transaction solves the problem for me by telling Django this block of code should be executed all together.
* How to do it?
  + I either create a transaction decorator on the method that receive the request or by creating a method with with keyword.
  + Steps:

1. Import transaction from Django.db
   1. From Django.db import transactions.
2. Decorate the method:
   1. @transaction.atomic()
3. Method two:
   1. Above the code I use keyword with then I add the transaction.atomic() as if it’s a method, then I indent the code I want it to be executed as block.
      1. With transaction.atomic()

Code …..

Code 2…..

Raw SQL Query

* I can use rows to execute an sql query directly to db.
* Every Model.object manager have row method that allow me to run sql query directly to db.