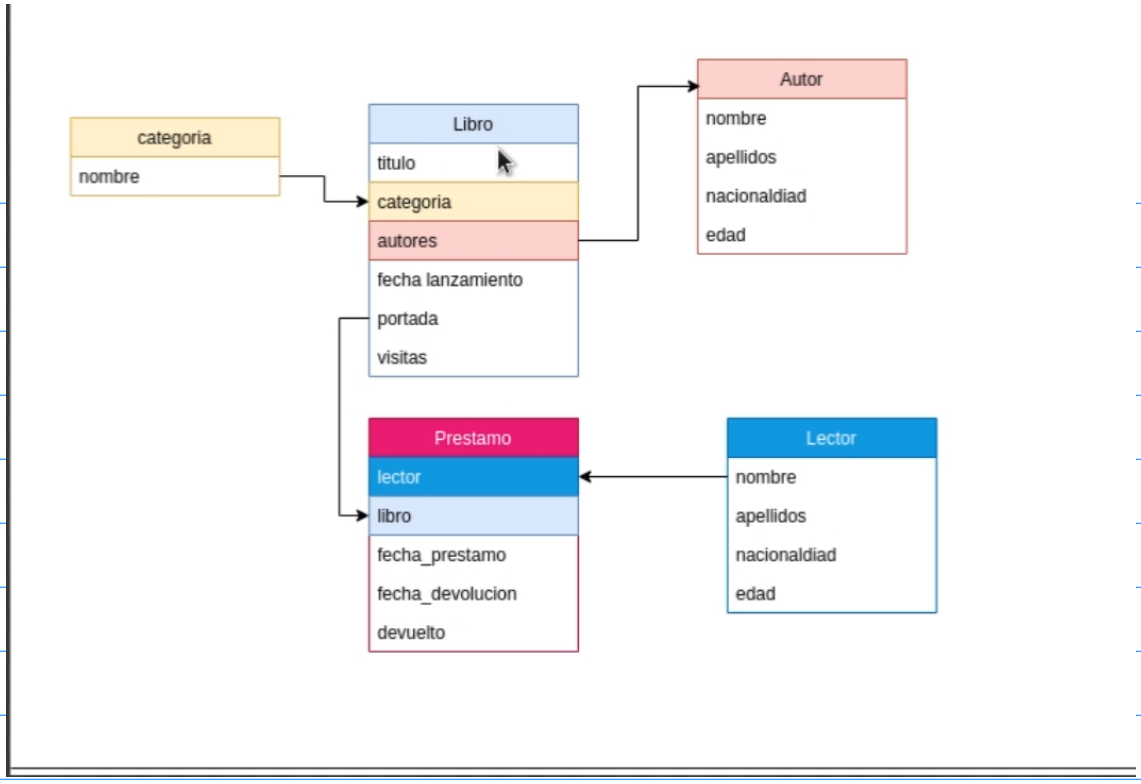
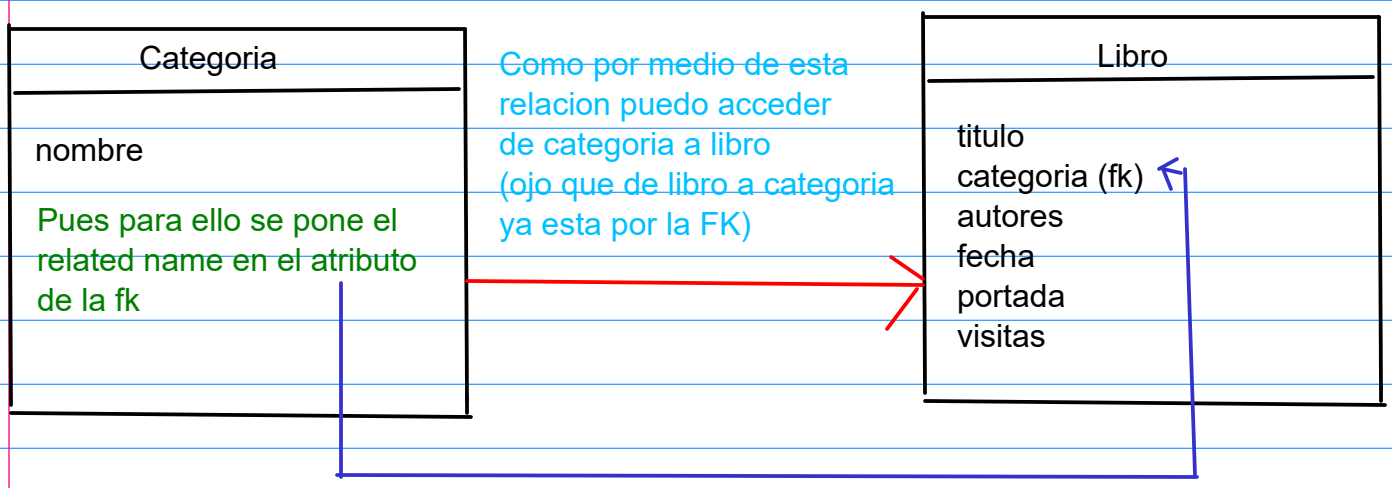


Requerimiento:
Listar todas las categorias de un autor
Los modelos de categoria y autor no estan relacionados directamente, entonces por medio de la relación que existe entre libro y categoría se pueden relacionar el autor con la categoria.

Para ello usamos el related_name que hace referencia a la relación inversa de la clave foranea, en pocas palabras nos sirve para acceder desde categoría a libro

Si se quiere acceder a libro desde categoria se puede usar este related_name



Y por medio de ese related name se puede acceder a Libro desde Categoria es decir al libro asociado a esa categoria

categoria_libro hace referencia a la relacion de libro con su fk de categoria respectiva y una vez unidos estos modelos accedo al otro atributo de la clase libro para hacer la pregunta... categoria_libro_autores_id = autor

Resumiendo el modelo libro le jala a toda la clase Categoria para al momento de hacer la pregunta retornar la categoria del libro

Clase diferencia entre Annotate y Aggregate

Aggregate: Devuelve un diccionario de valores agregados (promedios, sumas, etc.) calculados sobre el QuerySet. Cada argumento para aggregate()especifica un valor que se incluirá en el diccionario que se devuelve.

Se plantea contar cuantas veces fue prestado un libro y se intentará hacerlo usando el aggregate

De igual forma en el modelo prestamo se añade el related_name para poder acceder a la cantidad de relaciones y poder contarlas.
OjO que el Aggregate me retorna un DICCIONARIO a diferencia del Annotate que me retorna un QuerySet

¿En qué caso uso el Annotate y el Aggregate?

Para el caso del conteo que queremos en una consulta o en un queryset, obligatoriamente utilizaremos el annotate, pero si solamente necesitamos una operación aritmética para encontrar un valor, vamos a utilizar el aggregate

Se plantea que de un determinado libro del modelo libro, por ejemplo del libro A se calcule el promedio de la edad de los lectores que se prestan ese libro A

Para solucionar este problema se podria hacerlo en libro, hacer un filtro con el lado inverso y desde prestamo llegar a libro pero es mas sencillo hacer el procedimiento dentro de la tabla prestamo que ya está vinculada directamente tanto a libro como a lector, además de que es más óptimo

Entonces de igual forma se creó un manager para el modelo prestamo y ahí se hizo uso del aggregate, además al retornar un diccionario ésta funcion pues nos permite seguir añadiendo más operaciones

Values, Group_By en al ORM Django

Se plantea como requerimiento que se cuente cuantas veces se ha prestado cada libro dentro de mi tabla prestamo, por ejemplo Dracula se ha prestado 3 veces porque se repite tres veces

El annotate para que siga acumulando cada que encuentra un libro necesita algo que le indique en base a que quiere agrupar esto

En el caso del listar_categoria_libros el annotate lo agrupa automaticamente en base a un ID del mismo modelo, es por ello que aquí sí funcionaba

Pero ahora voy a listar o estoy haciendo en base a prestamos y al hacerle en base al ID pues considera diferente registros por ejemplo

Para el prestamo 1 tengo el libro 1
pero para el prestamo 2 tambien tengo el libro 1
y tambien para el prestamo 3 tengo el libro 1 entonces lo esta considerando como diferentes

*Es por ello que para que en este caso el annotate funcione pues necesita de un indicador o identificador que le indique a él en base a qué va a agrupar y contar como referencia los registros que le estamos indicando que en este caso son libros

Ojo que al usar el values ya no se devuelve un queryset sino una lista de diccionarios
¿Con qué valores?
Pues por el valor por el cual nosotros estamos agrupando y la respuesta , que serían los
(value1, valorResp) = {'libro': 9, 'num_prestados': 1} 1

Sin embargo, no siempre vamos a requerir solo de un valor para hacer el group by por ello simplemente se añade dentro del values el otro atributo

Trigam con Postgres Django

Lo que se plantea es buscar libros a partir de una palabra clave

Para ello usaremos una extensión que ya nos ofrece Django exclusiva para Postgresql que se llama Trigram similarity

* Siempre necesita de 3 caracteres
* Por defecto Postgres lo tiene desactivado entonces se debe activar, despues debemos indicar en que tabla y atributo se trabajará la triagramación
* En Windows ya viene instalado por defecto

Ya en la terminal de postgresql necesitamos indicarle a la BD que vamos a utilizar la triagramacion
CREATE EXTENSION pg_trgm;
Ahora indicaremos que se use la triagramacion en la tabla y en el atributo

CREATE INDEX (nombreAplicacion_nombreModelo_idx ON (nombreapp_nombre_modelo)
USING GIN (nombreAtributo gin_trgm_ops);

CREATE INDEX libro_titulo_idx ON libro_libro USING GIN (titulo gin_trgm_ops);