Mesut ÖNCEL Software Developer

mstoncel

@mesutoncel

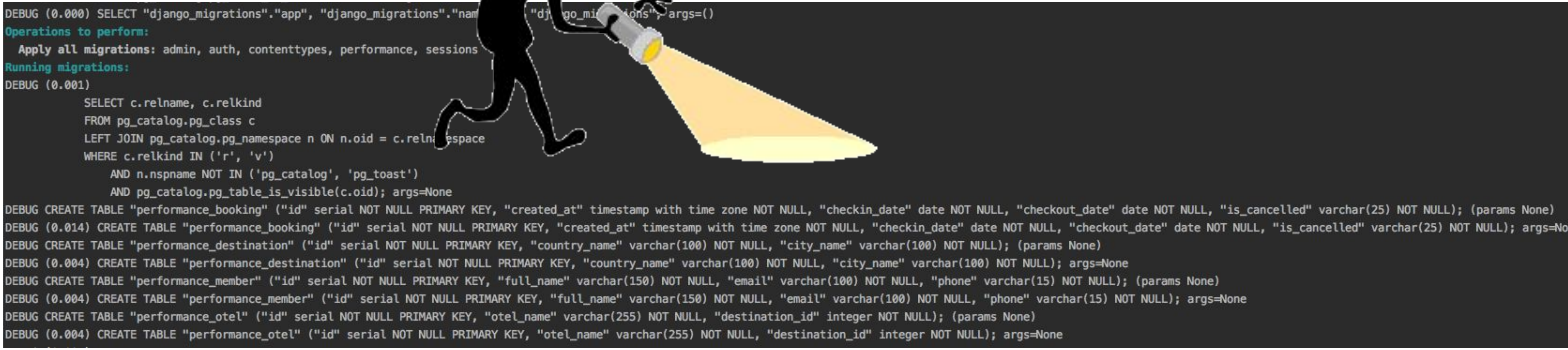
mesutoncel

mesutoncel91@gmail.com

@mesutoncel

metglobal

```
LOGGING = {
    'version': 1,
    'disable_existing_loggers' : False,
    'formatters': {
        'simple': {
            'format': '%(levelname)s %(message)s '
        },
    },
    'handlers': {
        'log_to_stdout': {
            'level': 'DEBUG',
            'class': 'logging.StreamHandler' ,
            'formatter': 'simple',
            },
        },
    'loggers': {
        'main': {
            'handlers': ['log_to_stdout'],
            'level': 'DEBUG',
            'propagate': True,
        },
        'django.db': {
            'handlers': ['log_to_stdout'],
            'level': 'DEBUG',
            'propagate': True,
        }
    }
}
```



```
DEBUG (0.000) SELECT "django_migrations"."app", "django_migrations"."nam[illegible] "dj[illegible]go_mi[illegible]ions"; args=()
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, performance, sessions
Running migrations:
DEBUG (0.001)
            SELECT c.relname, c.relkind
            FROM pg_catalog.pg_class c
            LEFT JOIN pg_catalog.pg_namespace n ON n.oid = c.relna[illegible]espace
            WHERE c.relkind IN ('r', 'v')
                AND n.nspname NOT IN ('pg_catalog', 'pg_toast')
                AND pg_catalog.pg_table_is_visible(c.oid); args=None
DEBUG CREATE TABLE "performance_booking" ("id" serial NOT NULL PRIMARY KEY, "created_at" timestamp with time zone NOT NULL, "checkin_date" date NOT NULL, "checkout_date" date NOT NULL, "is_cancelled" varchar(25) NOT NULL); (params None)
DEBUG (0.014) CREATE TABLE "performance_booking" ("id" serial NOT NULL PRIMARY KEY, "created_at" timestamp with time zone NOT NULL, "checkin_date" date NOT NULL, "checkout_date" date NOT NULL, "is_cancelled" varchar(25) NOT NULL); args=No
DEBUG CREATE TABLE "performance_destination" ("id" serial NOT NULL PRIMARY KEY, "country_name" varchar(100) NOT NULL, "city_name" varchar(100) NOT NULL); (params None)
DEBUG (0.004) CREATE TABLE "performance_destination" ("id" serial NOT NULL PRIMARY KEY, "country_name" varchar(100) NOT NULL, "city_name" varchar(100) NOT NULL); args=None
DEBUG CREATE TABLE "performance_member" ("id" serial NOT NULL PRIMARY KEY, "full_name" varchar(150) NOT NULL, "email" varchar(100) NOT NULL, "phone" varchar(15) NOT NULL); (params None)
DEBUG (0.004) CREATE TABLE "performance_member" ("id" serial NOT NULL PRIMARY KEY, "full_name" varchar(150) NOT NULL, "email" varchar(100) NOT NULL, "phone" varchar(15) NOT NULL); args=None
DEBUG CREATE TABLE "performance_otel" ("id" serial NOT NULL PRIMARY KEY, "otel_name" varchar(255) NOT NULL, "destination_id" integer NOT NULL); (params None)
DEBUG (0.004) CREATE TABLE "performance_otel" ("id" serial NOT NULL PRIMARY KEY, "otel_name" varchar(255) NOT NULL, "destination_id" integer NOT NULL); args=None
```

django

Django Model Mimarisi Nasıl Olmalı?

```
class Member(models.Model):
    full_name = models.CharField(max_length=150)
    email = models.CharField(max_length=100, db_index=True)
    phone = models.CharField(max_length=15)

class Destination(models.Model):
    country_name = models.CharField(max_length=100)
    city_name = models.CharField(max_length=100)

class Otel(models.Model):
    destination = models.ForeignKey(Destination, on_delete=models.CASCADE)
    otel_name = models.CharField(max_length=255, db_index=True)

class Booking(models.Model):
    otel = models.ForeignKey(Otel, on_delete=models.CASCADE)
    member = models.ForeignKey(Member, on_delete=models.CASCADE)
    created_at = models.DateTimeField()
    checkin_date = models.DateField()
    checkout_date = models.DateField()
    is_cancelled = models.CharField(max_length=25, db_index=True)
```

Uygun data tipi ve boyutu seçilmeli.

Gereksiz index kullanımı engellenmeli.

.save()

```python
batch_size = 5000
for batch in tqdm(range(1, 100000, batch_size)):
    request_url = 'https://randomuser.me/api/?results={}'.format(batch_size)
    results = requests.request('GET', request_url)
    results = json.loads(results.text).get('results')
    if not isinstance(results, list):
        results = [results]
    for result in results:
        if not result:
            continue
        login = result.get('login')
        name = login.get('username')
        email = result.get('email')
        phone = '+9093408902384'
        Member(full_name=name, email=email, phone=phone).save()
```

Bulk Create

```
datas =[]
batch_size = 5000
for batch in tqdm(range(1, 100000, 5000)):
    request_url = 'https://randomuser.me/api/?results={}'.format(batch_size)
    results = requests.request('GET', request_url)
    results = json.loads(results.text).get('results')
    if not isinstance(results, list):
        results = [results]
    for result in results:
        if not result:
            continue
        login = result.get('login')
        name = login.get('username')
        email = result.get('email')
        phone = '+9093408902384'
        datas.append(Member(
            full_name=name,
            email=email,
            phone=phone))
batch_size = 600
Member.objects.bulk_create(datas, batch_size)
```

Sorgu Optimizasyonu

Lazy-loaded

```
In [2]: members = Member.objects.filter(id__range=(1,100))

In [3]: members=members.exclude(email__range=('peter.fleming@example.com','teresa.lozano@example.com'))

In [4]: members.values('phone').first()
    DEBUG (0.004) SELECT "performance_member"."phone" FROM "performance_member"
                   WHERE ("performance_member"."id" BETWEEN 1 AND 100
                          AND NOT ("performance_member"."email" BETWEEN 'peter.fleming@example.com'
                          AND 'teresa.lozano@example.com')) ORDER BY "performance_member"."id"
                 ASC LIMIT 1
Out[4]: {'phone': '+9093408902384'}
```

.get()

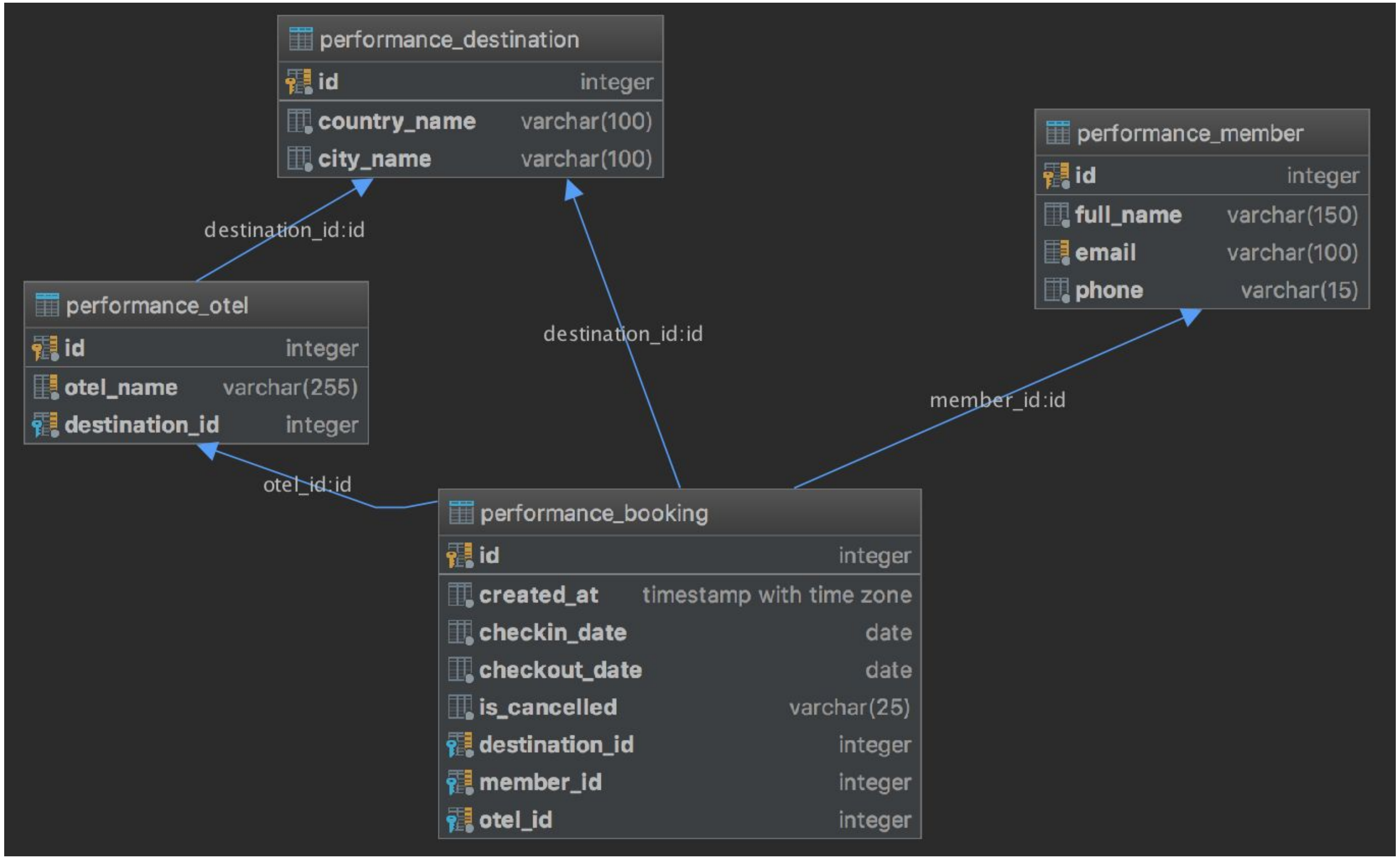
```
def get(self, *args, **kwargs):
        """
        Perform the query and return a single object matching the given
        keyword arguments.
        """
        clone = self.filter(*args, **kwargs)
        if self.query.can_filter() and not self.query.distinct_fields:
            clone = clone.order_by()
        num = len(clone)
        if num == 1:
            return clone._result_cache[0]
        if not num:
            raise self.model.DoesNotExist(
                "%s matching query does not exist." %
                self.model._meta.object_name
            )
        raise self.model.MultipleObjectsReturned(
            "get() returned more than one %s -- it returned %s!" %
            (self.model._meta.object_name, num)
```

.get() vs .first()

```
Member.objects.filter(email='amber.hall@example.com').first()

DEBUG (0.005) SELECT "performance_member"."id", "performance_member"."full_name",
                     "performance_member"."email", "performance_member"."phone"
  FROM "performance_member"
    WHERE "performance_member"."email" = 'amber.hall@example.com'
    ORDER BY "performance_member"."id" ASC LIMIT 1;



Member.objects.get(email='amber.hall@example.com')

DEBUG (0.003) SELECT "performance_member"."id", "performance_member"."full_name",
                    "performance_member"."email", "performance_member"."phone"
  FROM "performance_member"
  WHERE "performance_member"."email" = 'amber.hall@example.com';
```

Extra Foreign Key Eklenmesi

```
class Member(models.Model):
    full_name = models.CharField(max_length=150)
    email = models.CharField(max_length=100, db_index=True)
    phone = models.CharField(max_length=15)

class Destination(models.Model):
    country_name = models.CharField(max_length=100)
    city_name = models.CharField(max_length=100)

class Otel(models.Model):
    destination = models.ForeignKey(Destination,
on_delete=models.CASCADE)
    otel_name = models.CharField(max_length=255, db_index=True)

class Booking(models.Model):
    otel = models.ForeignKey(Otel, on_delete=models.CASCADE)
    member = models.ForeignKey(Member, on_delete=models.CASCADE)
    created_at = models.DateTimeField()
    checkin_date = models.DateField()
    checkout_date = models.DateField()
    is_cancelled = models.CharField(max_length=25)
    destination = models.ForeignKey(Otel, on_delete=models.CASCADE)
```



performance_destination	
id	integer
country_name	varchar(100)
city_name	varchar(100)

performance_member	
id	integer
full_name	varchar(150)
email	varchar(100)
phone	varchar(15)

performance_otel	
id	integer
otel_name	varchar(255)
destination_id	integer

performance_booking	
id	integer
created_at	timestamp with time zone
checkin_date	date
checkout_date	date
is_cancelled	varchar(25)
destination_id	integer
member_id	integer
otel_id	integer

.exists()

```
In [3]: members = Member.objects.exclude(email='amber.hall@example.com').filter(full_name='sadleopard927')

In [4]: if members:
   ...:     print(True)
   ...:
DEBUG (0.077) SELECT "performance_member"."id", "performance_member"."full_name", "performance_member"."email",
                     "performance_member"."phone"
                     FROM "performance_member"
                     WHERE (NOT ("performance_member"."email" = 'amber.hall@example.com')
                     AND "performance_member"."full_name" = 'sadleopard927');
True
```

```
In [2]: members = Member.objects.exclude(email='sofia.king@example.com').filter(full_name='sadleopard927')

In [3]: members.exists()
DEBUG (0.016) SELECT (1) AS "a"
                     FROM "performance_member"
                     WHERE (NOT ("performance_member"."email" = 'sofia.king@example.com')
                     AND "performance_member"."full_name" = 'sadleopard927') LIMIT 1;
Out[3]: True
```

.count()

```
In [4]: len(members)
DEBUG (0.105) SELECT "performance_member"."id", "performance_member"."full_name",
                        "performance_member"."email", "performance_member"."phone"
                   FROM "performance_member" WHERE (NOT ("performance_member"."email" = 'sofia.king@example.com')
                   AND "performance_member"."full_name" = 'sadleopard927');
```

```
In [2]: members = Member.objects.exclude(email='sofia.king@example.com').filter(full_name='sadleopard927')

In [3]: members.count()
DEBUG (0.078) SELECT COUNT(*) AS "__count" FROM "performance_member" WHERE (NOT ("performance_member"."email" =
'sofia.king@example.com') AND "performance_member"."full_name" = 'sadleopard927');
```

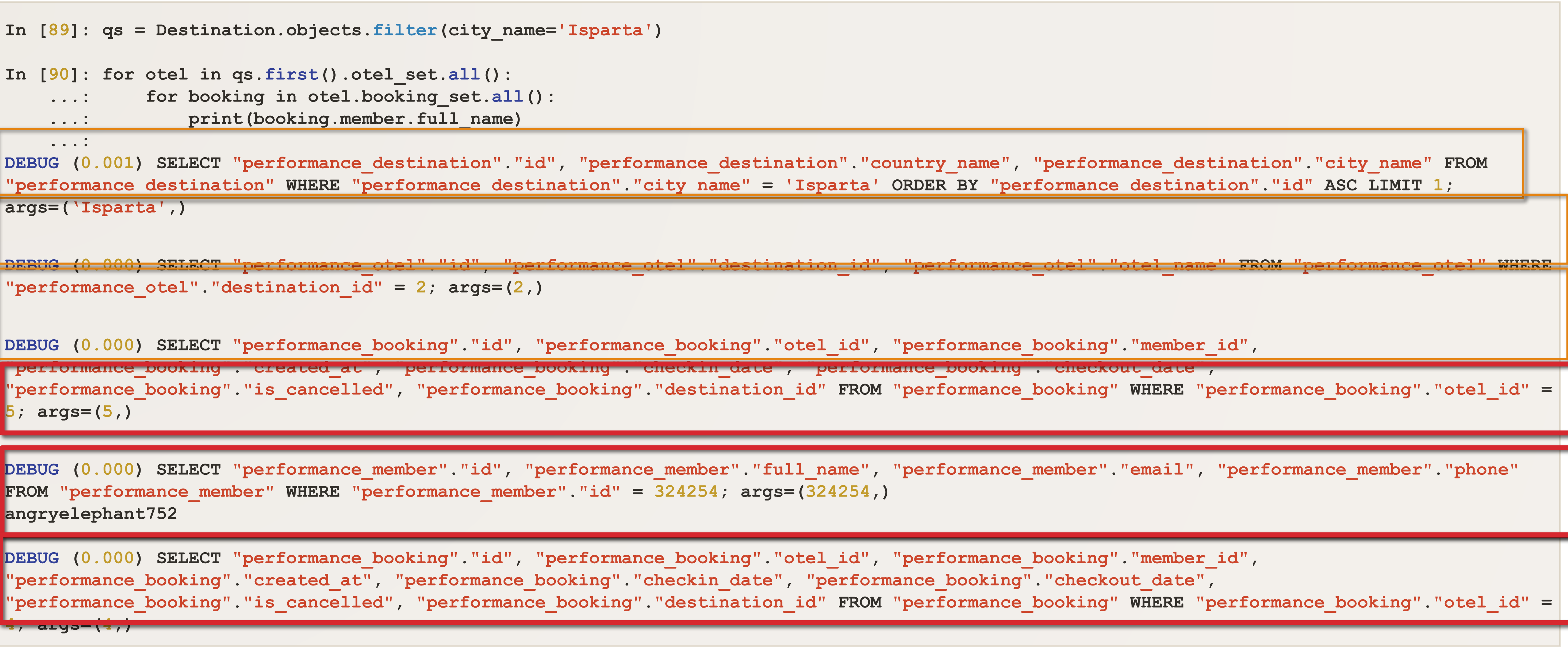
.prefect_related()

```
In [89]: Destination.objects.prefetch_related('booking_set')

Out[89]: DEBUG (0.002) SELECT "performance_destination"."id", "performance_destination"."country_name",
                        "performance_destination"."city_name" FROM "performance_destination" LIMIT 21; args=()
DEBUG (0.001) SELECT "performance_booking"."id", "performance_booking"."otel_id", "performance_booking"."member_id",
                     "performance_booking"."created_at", "performance_booking"."checkin_date",
"performance_booking"."checkout_date",
                     "performance_booking"."is_cancelled", "performance_booking"."destination_id"
                     FROM "performance_booking"
                     WHERE "performance_booking"."destination_id" IN (1, 2, 3); args=(1, 2, 3)
<QuerySet [<Destination: Destination object (1)>, <Destination: Destination object (2)>, <Destination: Destination object
(3)>]>
```

.select_related()

```
In [90]: Booking.objects.select_related('member')

Out[90]: DEBUG (0.005) SELECT "performance_booking"."id", "performance_booking"."otel_id", "performance_booking"."member_id",
                          "performance_booking"."created_at", "performance_booking"."checkin_date", "performance_booking"."checkout_date",
                          "performance_booking"."is_cancelled", "performance_booking"."destination_id", "performance_member"."id",
                          "performance_member"."full_name", "performance_member"."email", "performance_member"."phone"
                          FROM "performance_booking"
                          INNER JOIN "performance_member"
                          ON ("performance_booking"."member_id" = "performance_member"."id") LIMIT 21; args=()
<QuerySet [<Booking: Booking object (2)>, <Booking: Booking object (3)>, <Booking: Booking object (4)>, <Booking: Booking object (5)>, <Booking:
Booking object (6)>]>
```

django



```
In [89]: qs = Destination.objects.filter(city_name='Isparta')

In [90]: for otel in qs.first().otel_set.all():
    ...:     for booking in otel.booking_set.all():
    ...:         print(booking.member.full_name)
    ...:
DEBUG (0.001) SELECT "performance_destination"."id", "performance_destination"."country_name", "performance_destination"."city_name" FROM
"performance_destination" WHERE "performance_destination"."city_name" = 'Isparta' ORDER BY "performance_destination"."id" ASC LIMIT 1;
args=('Isparta',)


DEBUG (0.000) SELECT "performance_otel"."id", "performance_otel"."destination_id", "performance_otel"."otel_name" FROM "performance_otel" WHERE
"performance_otel"."destination_id" = 2; args=(2,)


DEBUG (0.000) SELECT "performance_booking"."id", "performance_booking"."otel_id", "performance_booking"."member_id",
"performance_booking"."created_at", "performance_booking"."checkin_date", "performance_booking"."checkout_date",
"performance_booking"."is_cancelled", "performance_booking"."destination_id" FROM "performance_booking" WHERE "performance_booking"."otel_id" =
5; args=(5,)

DEBUG (0.000) SELECT "performance_member"."id", "performance_member"."full_name", "performance_member"."email", "performance_member"."phone"
FROM "performance_member" WHERE "performance_member"."id" = 324254; args=(324254,)
angryelephant752

DEBUG (0.000) SELECT "performance_booking"."id", "performance_booking"."otel_id", "performance_booking"."member_id",
"performance_booking"."created_at", "performance_booking"."checkin_date", "performance_booking"."checkout_date",
"performance_booking"."is_cancelled", "performance_booking"."destination_id" FROM "performance_booking" WHERE "performance_booking"."otel_id" =
4; args=(4,)

DEBUG (0.000) SELECT "performance_member"."id", "performance_member"."full_name", "performance_member"."email", "performance_member"."phone"
FROM "performance_member" WHERE "performance_member"."id" = 324254; args=(324254,)
angryelephant752
```

```
In [87]: qs = Destination.objects.filter(city_name=‘Isparta').prefetch_related(Prefetch('otel_set__booking_set',
                                         queryset=Booking.objects.select_related('member')))

In [88]: for otel in qs.first().otel_set.all():
    ...:     for booking in otel.booking_set.all():
    ...:         print(booking.member.full_name)
    ...:
DEBUG (0.001) SELECT "performance_destination"."id", "performance_destination"."country_name", "performance_destination"."city_name" FROM
"performance_destination" WHERE "performance_destination"."city_name" = 'Isparta' ORDER BY "performance_destination"."id" ASC LIMIT 1;
args=(‘Isparta',)

DEBUG (0.000) SELECT "performance_otel"."id", "performance_otel"."destination_id", "performance_otel"."otel_name" FROM "performance_otel" WHERE
"performance_otel"."destination_id" IN (2); args=(2,)

DEBUG (0.001) SELECT "performance_booking"."id", "performance_booking"."otel_id", "performance_booking"."member_id",
"performance_booking"."created_at", "performance_booking"."checkin_date", "performance_booking"."checkout_date",
"performance_booking"."is_cancelled", "performance_booking"."destination_id", "performance_member"."id", "performance_member"."full_name",
"performance_member"."email", "performance_member"."phone" FROM "performance_booking" INNER JOIN "performance_member" ON
("performance_booking"."member_id" = "performance_member"."id") WHERE "performance_booking"."otel_id" IN (4, 5); args=(4, 5)
angryelephant752
angryelephant752
```

django

.delete()

```
members=Member.objects.all()
members.delete()

DEBUG (0.001) DELE                                            ce_member"."id"
IN (101, 102, 103,                                            113, 114, 115,
 116, 117, 118, 11                                           28, 129, 130, 131,
 132, 133, 134, 13                                           4, 145, 146, 147,
 148, 149, 150, 15                                           0, 161, 162, 163,
 164, 165, 166, 16                                           6, 177, 178, 179,
 180, 181, 182, 18                                           2, 193, 194, 195,
 196, 197, 198, 19
DEBUG (0.001) DELE                                            ce_member"."id" IN
(1, 2, 3, 4, 5, 6,                                            19, 20, 21, 22, 23,
24, 25, 26, 27, 28                                            40, 41, 42, 43, 44,
45, 46, 47, 48, 49                                            61, 62, 63, 64, 65,
66, 67, 68, 69, 70                                            82, 83, 84, 85, 86,
87, 88, 89, 90, 91
Out[3]: (57000, {'performance.Booking': 0, 'performance.Member': 57000})
```

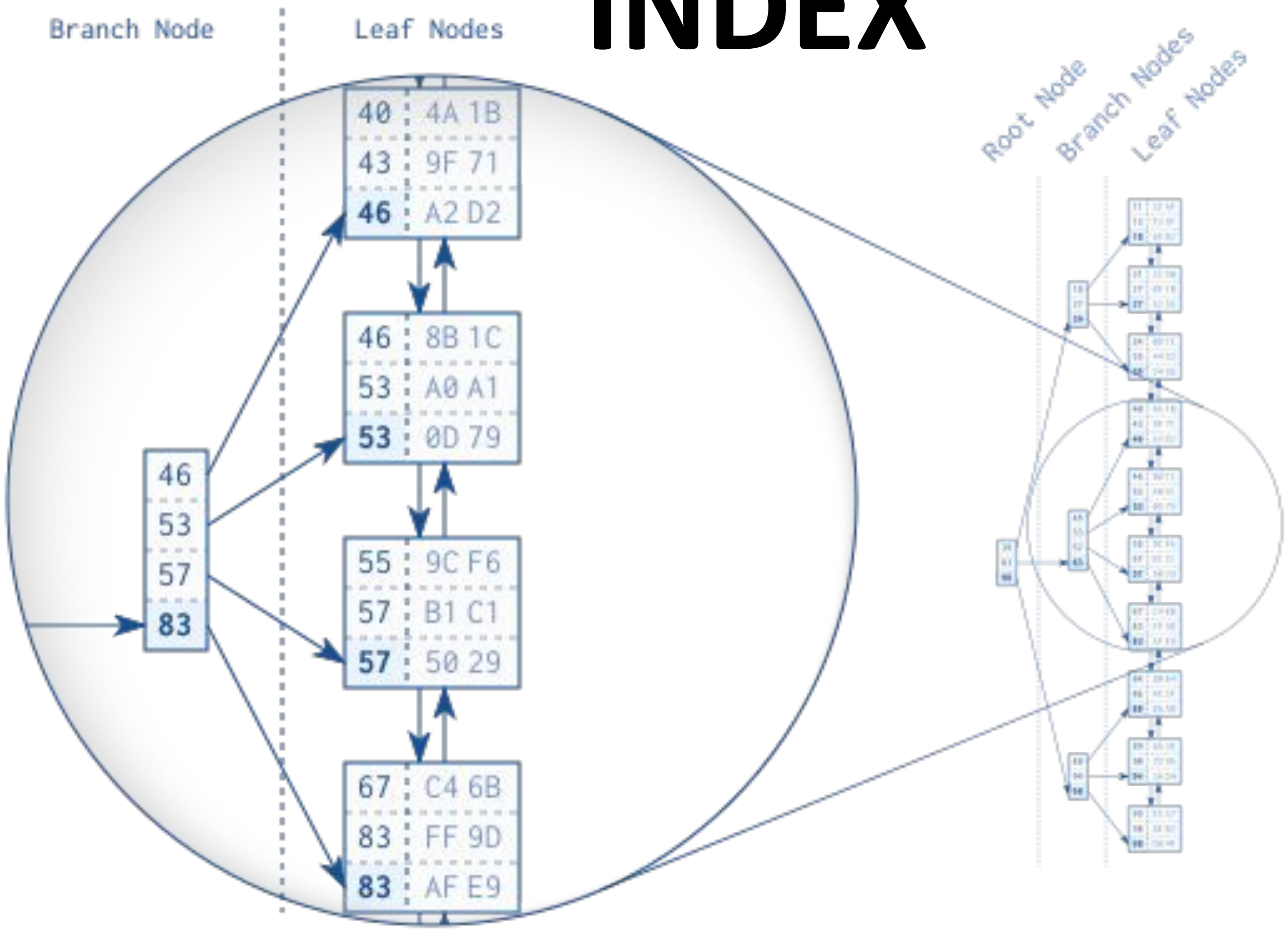
django

Truncate Table

```
class Member(models.Model):
    full_name = models.CharField(max_length=150)
    email = models.CharField(max_length=100, db_index=True)
    phone = models.CharField(max_length=15)

    @classmethod
    def truncate(cls):
        with connection.cursor() as cursor:
            cursor.execute('TRUNCATE TABLE "{0}" CASCADE'.format(cls._meta.db_table))
```

```
members = Member.truncate()
DEBUG (0.012) TRUNCATE TABLE "performance_member" CASCADE;
```

INDEX


Branch Node
Leaf Nodes
46
53
57
83
40 4A 1B
43 9F 71
46 A2 D2
46 8B 1C
53 A0 A1
53 0D 79
55 9C F6
57 B1 C1
57 50 29
67 C4 6B
83 FF 9D
83 AF E9
Root Node
Branch Nodes
Leaf Nodes

Index Operation

Concurrent Index

```
class Migration(migrations.Migration):
    atomic = False # disable transaction
    dependencies = [('performance', `0018_xxx_idx_booking')]
    operations = [
                migrations.RunSQL('CREATE INDEX CONCURRENTLY email_idx ON booking (email)’)
        ]
```

Partial Index

```
class Migration(migrations.Migration):

    dependencies = [
        ('performance', ‘0019_email_idx_booking’),
    ]

    operations = [
        migrations.RunSQL(
            "CREATE INDEX idx_booking ON performance_booking (created_at) where is_cancelled='booked';",
        )

    ]
```

TEŞEKKÜRLER