```
#1.
class Author(models.Model):
    name = models.CharField(
        max_length=40,
    )


class Book(models.Model):
    title = models.CharField(
        max_length=40,
    )
    price = models.DecimalField(
        max_digits=5,
        decimal_places=2,
    )
    author = models.ForeignKey(
        to=Author,
        on_delete=models.CASCADE,
    )


def show_all_authors_with_their_books():
    authors_with_books = []
    authors = Author.objects.all().order_by('id')
    for author in authors:
        # books = Book.objects.filter(author=author) - below is the same
        books = author.book_set.all()
        if not books:
            continue
        titles = ', '.join(book.title for book in books)
        authors_with_books.append(f'{author.name} has written - {titles}!')
    return '\n'.join(authors_with_books)


def delete_all_authors_without_books():
    Author.objects.filter(book__isnull=True).delete()


#2.
class Song(models.Model):
    title = models.CharField(
        max_length=100,
        unique=True,
    )


class Artist(models.Model):
    name = models.CharField(
        max_length=100,
        unique=True,
    )
    songs = models.ManyToManyField(
        to=Song,
        related_name='artists',
    )


def add_song_to_artist(artist_name, song_title):
    artist = Artist.objects.get(name=artist_name)
    song = Song.objects.get(title=song_title)
    artist.songs.add(song)


def get_songs_by_artist(artist_name):
    artist = Artist.objects.get(name=artist_name)
    return artist.songs.all().order_by('-id')


def remove_song_from_artist(artist_name, song_title):
    artist = Artist.objects.get(name=artist_name)
    song = Song.objects.get(title=song_title)
    artist.songs.remove(song)


#3.
class Product(models.Model):
    name = models.CharField(
        max_length=100,
        unique=True,
    )


class Review(models.Model):
    description = models.TextField(
        max_length=200,
    )
    rating = models.PositiveIntegerField()
    product = models.ForeignKey(
        to=Product,
        on_delete=models.CASCADE,
        null=True,
        blank=True,
            related_name='reviews',
    )


def calculate_average_rating_for_product_by_name(product_name):
    product = Product.objects.get(name=product_name)
    reviews = product.reviews.all()
    total_rating = sum(r.rating for r in reviews)
    average_rating = total_rating / len(reviews)

    return average_rating

    # product = Product.objects.annotate(
    #     total_ratings=Sum('review__rating'),
    #     num_reviews=Count('review'),
    # ).get(name=product_name)
    #
    # average_rating = product.total_ratings / product.num_reviews
    #
    # return average_rating


def get_reviews_with_high_ratings(treshold):
    return Review.objects.filter(rating__gte=treshold)
```

```
def get_products_with_no_reviews():
    return Product.objects.filter(reviews__isnull=True).order_by('-name')


def delete_products_without_reviews():
    Product.objects.filter(reviews__isnull=True).delete()



#4.
class Driver(models.Model):
    first_name = models.CharField(
        max_length=50,
    )
    last_name = models.CharField(
        max_length=50,
    )


class DrivingLicense(models.Model):
    license_number = models.CharField(
        max_length=10,
        unique=True,
    )
    issue_date = models.DateField()
    driver = models.OneToOneField(
        to=Driver,
        on_delete=models.CASCADE,
    )

    def __str__(self):
        expiration_date = self.issue_date + timedelta(days=365)
        return f'License with id: {self.license_number} expires on
{expiration_date}'


def calculate_licenses_expiration_dates():
    licenses = DrivingLicense.objects.all().order_by('-license_number')
    return '\n'.join(str(l) for l in licenses)


def get_drivers_with_expired_licenses(due_date):
    expiration_cutoff_date = due_date - timedelta(days=365)
    expired_drivers =
Driver.objects.filter(drivinglicense__issue_date__gt=expiration_cutoff_date)
    return expired_drivers

#5.
class Owner(models.Model):
    name = models.CharField(
        max_length=50,
    )


class Car(models.Model):
    model = models.CharField(
        max_length=50,
    )
    year = models.PositiveIntegerField()
    owner = models.ForeignKey(
        to=Owner,
        on_delete=models.CASCADE,
        null=True,
        blank=True,
        related_name='cars',
    )


class Registration(models.Model):
    registration_number = models.CharField(
        max_length=10,
        unique=True,
    )
    registration_date = models.DateField(
        null=True,
        blank=True,
    )
    car = models.OneToOneField(
        to=Car,
        on_delete=models.CASCADE,
        null=True,
        blank=True,
    )


def register_car_by_owner(owner):
    registration = Registration.objects.filter(car__isnull=True).first()
    car = Car.objects.filter(registration__isnull=True).first()

    car.owner = owner
    car.registration = registration
    car.save()

    registration.registration_date = date.today()
    registration.car = car
    registration.save()

    return f'Successfully registered {car.model} to {owner.name} with registration
number {registration.registration_number}.'



#6.
@admin.register(Car)
class CarAdmin(admin.ModelAdmin):
    list_display = ['model', 'year', 'owner', 'car_details']

    @staticmethod
    def car_details(obj):
        try:
            owner_name = obj.owner.name
        except AttributeError:
            owner_name = "No owner"

        try:
            registration_number = obj.registration.registration_number
        except AttributeError:
            registration_number = "No registration number"
        return f'Owner: {owner_name}, Registration: {registration_number}'

    car_details.short_description = "Car Details"
```