```
CTRL + D = paste the current line on the lines below

#1.
class ArtworkGallery(models.Model):
    artist_name = models.CharField(
        max_length=100,
    )
    art_name = models.CharField(
        max_length=100,
    )
    rating = models.IntegerField()
    price = models.DecimalField(
        max_digits=10,
        decimal_places=2,
    )


def show_highest_rated_art():
    best_artwork = ArtworkGallery.objects.order_by('-rating', 'id').first()

    return f"{best_artwork.art_name} is the highest-rated art with a
{best_artwork.rating} rating!"


artwork1 = ArtworkGallery(artist_name="Vincent van Gogh", art_name="Starry Night",
rating=4, price=1200000.0)
artwork1.save()
artwork2 = ArtworkGallery(artist_name="Leonardo da Vinci", art_name="Mona Lisa",
rating=5, price=1500000.0)
artwork2.save()

print(show_highest_rated_art())


def bulk_create_art(first_art, second_art):
    ArtworkGallery.objects.bulk_create([
        first_art,
        second_art
    ])


def delete_negative_rated_arts():
    ArtworkGallery.objects.filter(rating__lt=0).delete()


#2.
class Laptop(models.Model):
    BRAND_CHOICES = (
        ('Asus', 'Asus'),
        ('Acer', 'Acer'),
        ('Apple', 'Apple'),
        ('Lenovo', 'Leonvo'),
        ('Dell', 'Dell'),
    )
    OS_CHOICES = (
        ('Windows', 'Windows'),
        ('MacOS', 'MacOS'),
        ('Linux', 'Linux'),
        ('Chrome OS', 'Chrome OS'),
    )

    brand = models.CharField(
        max_length=20,
        choices=BRAND_CHOICES,
    )
    processor = models.CharField(
        max_length=100,
    )
    memory = models.PositiveIntegerField(
        help_text='Memory in GB',
    )
    storage = models.PositiveIntegerField(
        help_text='Storage in GB',
    )
    operating_system = models.CharField(
        max_length=20,
        choices=OS_CHOICES,
    )
    price = models.DecimalField(
        max_digits=10,
        decimal_places=2,
    )


def show_the_most_expensive_laptop():
    laptop = Laptop.objects.order_by('-price', '-id').first()

    return f"{laptop.brand} is the most expensive laptop available for
{laptop.price}$!"


def bulk_create_laptops(*args: List[Laptop]):
    Laptop.objects.bulk_create(*args)


def update_to_512_GB_storage():
    # Laptop.objects.filter(brand__in=['Asus', 'Lenovo']).update(storage=512)
    Laptop.objects.filter(Q(brand='Lenovo') | Q(brand='Asus')).update(storage=512)


def update_to_16_GB_memory():
    Laptop.objects.filter(brand__in=['Apple', 'Dell', 'Acer']).update(memory=16)


def update_operation_systems():
    Laptop.objects.filter(brand='Asus').update(operation_system='Windows')
    Laptop.objects.filter(brand='Apple').update(operation_system='MacOS')
    Laptop.objects.filter(brand__in=['Dell',
'Acer']).update(operation_system='Linux')
    Laptop.objects.filter(brand='Lenovo').update(operation_system='Chrome OS')

    # case2
    Laptop.objects.update(
        operation_system=Case(
            When(brand='Asus', then=Value('Windows')),
            When(brand='Apple', then=Value('MacOS')),
            When(brand__in=['Dell', 'Acer'], then=Value('Linux')),
            When(brand='Lenovo', then=Value('Chrome OS')),
            default=F('operation_system'),
        )
    )


def delete_inexpensive_laptops():
    Laptop.objects.filter(price__lt=1200).delete()

#3.

def bulk_create_chess_players(*args):
    ChessPlayer.objets.bulk_create(*args)


def delete_chess_players():
    ChessPlayer.objets.filter(title='no title').delete()


def change_chess_games_won():
    ChessPlayer.objets.filter(title='GM').update(games_won=30)


def change_chess_games_lost():
    ChessPlayer.objets.filter(title='no title').update(games_lost=25)


def change_chess_games_drawn():
    ChessPlayer.objets.update(games_drawn=10)


def grand_chess_title_GM():
    ChessPlayer.objets.filter(rating__gte=2400).update(title='GM')


def grand_chess_title_IM():
    # ChessPlayer.objets.filter(Q(rating__gte=2300) & Q(rating__lt=2400))
    ChessPlayer.objets.filter(rating__range=[2300, 2399]).update(title='IM')


def grand_chess_title_FM():
    ChessPlayer.objets.filter(rating__range=[2200, 2299]).update(title='FM')


def grand_chess_title_regular_player():
    ChessPlayer.object.filter(rating__range=[0, 2199]).update(title='regular
player')



#4.
def set_new_chefs():
    Meal.objects.filter(meal_type='Breakfast').update(chef='Gordon Ramsay')
    Meal.objects.filter(meal_type='Lunch').update(chef='Julia Child')
    Meal.objects.filter(meal_type='Dinner').update(chef='Jamie Oliver')
    Meal.objects.filter(meal_type='Snack').update(chef='Thomas Keller')


def set_new_preparation_times():
    Meal.objects.filter(meal_type='Breakfast').update(preparation_time='10
minutes')
    Meal.objects.filter(meal_type='Lunch').update(preparation_time='12 minutes')
    Meal.objects.filter(meal_type='Dinner').update(preparation_time='15 minutes')
    Meal.objects.filter(meal_type='Snack').update(preparation_time='5 minutes')


def update_low_calorie_meals():
    Meal.objects.filter(meal_type__in=['Breakfast', 'Dinner']).update(calories=400)


def update_high_calorie_meals():
    Meal.objects.filter(meal_type__in=['Lunch', 'Snack']).update(calories=700)


def delete_lunch_and_snack_meals():
    Meal.objects.filter(meal_type__in=['Lunch', 'Snack']).delete()


#5.

def __str__(self):
    return f'{self.name} is guarded by {self.boss_name} who has {self.boss_health}
health points!'

def show_hard_dungeons():
    hard_dungeons = Dungeon.objects.filter(difficulty='Hard').order_by('-location')
    return  '\n'.join(str(x) for x in hard_dungeons)

def bulk_create_dungeons(*args):
    Dungeon.objects.bulk_create(*args)

def update_dungeon_names():
    Dungeon.objects.filter(difficulty='Easy').update(name='The Erased Thombs')
    Dungeon.objects.filter(difficulty='Medium').update(name='The Coral Labyrinth')
    Dungeon.objects.filter(difficulty='Hard').update(name='The Lost Haunt')


def update_dungeon_bosses_health():
    Dungeon.objects.exclude(difficulty='Easy').update(boss_health=500)


def update_dungeon_recommended_levels():
    Dungeon.objects.filter(difficulty='Easy').update(recommended_level=25)
    Dungeon.objects.filter(difficulty='Medium').update(recommended_level=50)
    Dungeon.objects.filter(difficulty='Hard').update(recommended_level=75)


def update_dungeon_rewards():
    Dungeon.objects.filter(boss_health=500).update(reward='1000 Gold')
    Dungeon.objects.filter(location__startswith='E').update(reward='New dungeon
unlocked')
    Dungeon.objects.filter(location__endswith='s').update(reward='Dragonheart
Amulet')


def set_new_locations():
    Dungeon.objects.filter(recommended_level=25).update(location='Enchanted Maze')
    Dungeon.objects.filter(recommended_level=50).update(location='Grimstone Mines')
    Dungeon.objects.filter(recommended_level=75).update(location='Shadowed Abyss')
```

```
#6.
def __str__(self):
    return f'{self.name} from {self.workout_type} type has {self.difficulty}
difficulty!'


def show_workout():
    calisthenics_crossfit_workouts =
Workout.objects.filter(workout_tyope__in=['Calisthenics', 'Crossfit'])
    return '\n'.join(str(x) for x in calisthenics_crossfit_workouts)


def get_high_difficulty_cardio_workouts():
    return Workout.objects.filter(workout_type='Cardio',
difficulty='High').order_by('instructor')


def set_new_instructors():
    Workout.objects.filter(workout_type='Cardio').update(instructor='John Smith')
    Workout.objects.filter(workout_type='Strength').update(instructor='Michael
Williams')
    Workout.objects.filter(workout_type='Yoga').update(instructor='Emily Johnson')
    Workout.objects.filter(workout_type='CrossFit').update(instructor='Sarah
Davis')
    Workout.objects.filter(workout_type='Calistheics').update(instructor='Chris
Heria')


def set_duration_times():
    Workout.objects.filter(instructor='John Smith').update(duration='15 minutes')
    Workout.objects.filter(instructor='Sarah Davis').update(duration='30 minutes')
    Workout.objects.filter(instructor='Chris Heria').update(duration='45 minutes')
    Workout.objects.filter(instructor='Michael Williams').update(duration='1 hour')
    Workout.objects.filter(instructor='Emily Johnson').update(duration='1 hour and
30 minutes')

def delete_workouts():
    Workout.objects.exclude(workout_type__in=['Calisthenics', 'Strength']).delete()
```