1. Order and OnlineOrder

```
class Order:
    def __init__(self, order_id):
        self.order_id = order_id

    def process_payment(self):
        print(f"Processing payment for Order #{self.order_id}")

class OnlineOrder(Order):
    def __init__(self, order_id, email):
        super().__init__(order_id)
        self.email = email

    def process_payment(self):
        super().process_payment()
        print(f"Sending confirmation email to {self.email}")

order = OnlineOrder(101, "customer@example.com")
order.process_payment()
```

Key Points :

1. There are two classes Order and OnlineOrder.
2. OnlineOrder is a child class of Order which means it inherits from it.
3. The super() keyword is used to call the process_payment method from the parent class.
4. The process_payment method is overridden in OnlineOrder to add more features like email confirmation.
5. It processes the payment using the parent class method, then sends a confirmation email.

2. Employee and Manager

```
class Employee:
    def __init__(self, name, salary):
        self.name = name
        self.salary = salary

    def display(self):
        print(f"Name: {self.name}, Salary: ₹{self.salary}")

class Manager(Employee):
    def __init__(self, name, salary, department):
        super().__init__(name, salary)
        self.department = department

    def display(self):
        super().display()
        print(f"Department: {self.department}")

m = Manager("Shaik", 90000, "IT")
m.display()
```

Key Points :

1. There are two classes Employee and Manager.
2. Manager class is inheriting from Employee.
3. The child class Manager adds an extra attribute called department.
4. The display() method is also overridden to show department along with name and salary.
5. Displays combined output from parent and child classes.

3. Vehicle and Car

```
class Vehicle:
    def start(self):
        print("Vehicle started")

class Car(Vehicle):
    def start(self):
        super().start()
        print("Car is ready to go")

c = Car()
c.start()
```

Key Points :

1. There are two classes Vehicle and Car.
2. Car is a subclass of Vehicle.
3. Method start() is overridden and extended using super().start().
4. First, it prints "Vehicle started", then "Car is ready to go".

4. User Login System

```
class User:
    def __init__(self, username):
        self.username = username

    def login(self):
        print(f"{self.username} logged in")

class Admin(User):
    def login(self):
        super().login()
        print(f"{self.username} has admin privileges")

a = Admin("admin_user")
a.login()
```

Key Points :

1. There are two classes User and Admin.
2. Admin is a child class of User.

3. Method login() is overridden to add a message.
4. Uses super().login() to reuse the login message from User.

5. Shape and Circle

```
class Shape:
    def __init__(self):
        print("This is a shape")

    def area(self):
        print("Area formula not defined")

class Circle(Shape):
    def __init__(self, radius):
        super().__init__()
        self.radius = radius

    def area(self):
        super().area()
        print("Circle Area:", 3.14 * self.radius * self.radius)

c = Circle(5)
c.area()
```

Key Points :

1. There are two classes Shape and Circle.
2. Circle inherits from Shape.
3. The area() method is overridden to display the area of circle.
4. First it prints the base message "Area formula not defined".

6. Person and Student

```
class Person:
    def __init__(self, name):
        self.name = name

    def show(self):
        print(f"Name: {self.name}")

class Student(Person):
    def __init__(self, name, grade):
        super().__init__(name)
        self.grade = grade

    def show(self):
        super().show()
        print(f"Grade: {self.grade}")

s = Student("Ali", "A")
s.show()
```

Key Points :

1. There are two classes Person and Student
2. Student class inherits from Person.
3. Student class adds a new attribute called grade.
4. The show() method is overridden to display the grade.
5. It uses super().show() to reuse code for name display.

7. BankAccount and SavingsAccount

```
class BankAccount:
    def __init__(self, balance):
        self.balance = balance

    def show_balance(self):
        print(f"Balance: ₹{self.balance}")

class SavingsAccount(BankAccount):
    def __init__(self, balance, interest):
        super().__init__(balance)
        self.interest = interest

    def show_balance(self):
        super().show_balance()
        print(f"Interest Rate: {self.interest}%")

acc = SavingsAccount(10000, 5)
acc.show_balance()
```

Key Points :

1. There are two classess BankAccount and SavingsAccount.
2. SavingsAccount is a subclass of BankAccount.
3. SavingsAccount adds an extra attribute interest.
4. SavingsAccount overrides show_balance() to display interest rate.
5. super().show_balance() is used to print the base balance.

8. Product and ElectronicProduct

```
class Product:
    def __init__(self, name):
        self.name = name

    def details(self):
        print(f"Product: {self.name}")

class ElectronicProduct(Product):
    def __init__(self, name, warranty):
        super().__init__(name)
        self.warranty = warranty
```

```
    def details(self):
        super().details()
        print(f"Warranty: {self.warranty} years")

p = ElectronicProduct("Laptop", 2)
p.details()
```

Key Points :

1. There are two classes Product and ElectronicProduct.
2. ElectronicProduct inherits from Product.
3. ElectronicProduct adds a new property called warranty.
4. The details() method is overridden to include warranty.
5. super().details() prints the product name from the parent class.

9. Animal and Dog

```
class Animal:
    def sound(self):
        print("Animal sound")

class Dog(Animal):
    def sound(self):
        super().sound()
        print("Dog barks")

d = Dog()
d.sound()
```

Key Points :

1. There are two classes Animal and Dog.
2. Dog class inherits from Animal.
3. The sound() method is overridden in Dog.
4. Dog class calls the parent sound() method first using super() and then adds its own message.

10. Book and EBook

```
class Book:
    def __init__(self, title):
        self.title = title

    def show(self):
        print(f"Title: {self.title}")

class EBook(Book):
    def __init__(self, title, file_size):
        super().__init__(title)
        self.file_size = file_size
```

```
    def show(self):
        super().show()
        print(f"File Size: {self.file_size} MB")

eb = EBook("Python Guide", 5)
eb.show()
```

Key Points :

1. There are two classes Book and EBook.
2. EBook inherits from Book.
3. EBook adds a new attribute file_size.
4. The show() method is overridden to display file size.
5. EBook calls the show() method first using super() and display file size along with Title.