Mari kita analisis logika kode Anda dalam model CourseBooking dan hubungannya dengan model yang diinherit (sale.order dan sale.order.line) yang baru.

**1. Metode create**

@api.model

def create(self, vals):

if vals.get('name', \_('New')) == \_('New'):

vals['name'] = self.env['ir.sequence'].next\_by\_code('course.booking') or \_('New')

return super(CourseBooking, self).create(vals)

* **Logika**: Ketika membuat entri baru untuk CourseBooking, jika name tidak diberikan, sistem akan menghasilkan nama baru menggunakan sequence course.booking.

* **Kaitannya**: Setelah entri baru dibuat, name yang dihasilkan dapat digunakan di dalam sale.order untuk melacak asal usul pesanan tersebut.

**2. One2many Field sale\_order\_line\_ids**

sale\_order\_line\_ids = fields.One2many('sale.order', 'course\_booking\_id', string='Sale Order Line')

* **Logika**: Ini mendefinisikan relasi satu ke banyak antara CourseBooking dan sale.order. Dengan ini, Anda dapat mengakses semua order penjualan yang terkait dengan booking ini.

* **Kaitannya**: Saat sale.order baru dibuat, course\_booking\_id pada sale.order akan diisi dengan ID CourseBooking, sehingga Anda bisa mengaitkan antara keduanya.

**3. Metode action\_confirm**

def action\_confirm(self):

for rec in self:

sequence\_name = self.env['ir.sequence'].next\_by\_code('course.booking.number.sequence')

rec.write({'name': sequence\_name, 'state': 'confirm'})

rec.\_create\_sale\_order()

return True

* **Logika**: Mengonfirmasi booking dengan mengubah status menjadi 'confirm', memberikan nama baru berdasarkan sequence, dan membuat order penjualan baru.

* **Kaitannya**: Di dalam \_create\_sale\_order, Anda menambahkan field course\_booking\_id, yang memungkinkan Anda menghubungkan booking dengan pesanan penjualan yang dihasilkan.

**4. Metode \_compute\_sale\_order\_count**

@api.depends('sale\_order\_line\_ids')

def \_compute\_sale\_order\_count(self):

for rec in self:

rec.sale\_order\_count = len(rec.sale\_order\_line\_ids)

* **Logika**: Menghitung jumlah order penjualan yang terkait dengan booking ini.

* **Kaitannya**: Memungkinkan pengguna untuk melihat berapa banyak pesanan yang terkait langsung dengan booking tertentu.

**5. Metode \_create\_sale\_order**

def \_create\_sale\_order(self):

sale\_order\_obj = self.env['sale.order']

sale\_order\_values = self.\_prepare\_sale\_order\_values()

sale\_order\_id = sale\_order\_obj.create(sale\_order\_values)

return sale\_order\_id

* **Logika**: Membuat pesanan penjualan baru dengan nilai yang disiapkan oleh \_prepare\_sale\_order\_values.

* **Kaitannya**: Pesanan yang dibuat di sini akan memiliki referensi ke CourseBooking, yang memungkinkan pengelolaan dan pelacakan yang lebih baik.

**6. Metode \_prepare\_sale\_order\_values**

def \_prepare\_sale\_order\_values(self):

return {

'partner\_id': self.partner\_id.id,

'user\_id': self.user\_id.id,

'origin': self.name,

'validity\_date': self.expiration\_date,

'date\_order': fields.Datetime.now(),

'course\_booking\_id': self.id,

'order\_line': self.\_prepare\_order\_line()

}

* **Logika**: Menyusun nilai untuk pesanan penjualan baru, termasuk mengaitkan course\_booking\_id.

* **Kaitannya**: Menyediakan informasi yang relevan untuk pesanan penjualan yang dihasilkan, termasuk lini pesanan yang akan disiapkan.

**7. Metode \_prepare\_order\_line**

def \_prepare\_order\_line(self):

values = [(0, 0, {

'course\_booking\_line\_id': record.id,

'product\_id': record.product\_id.id,

'name': '%s - %s' % (record.product\_id.name, record.course\_subject\_id.name),

'product\_uom': record.product\_id.uom\_id.id,

'product\_uom\_qty': 1,

'price\_unit': record.sale\_price

}) for record in self.booking\_line\_ids]

return values

* **Logika**: Menyusun lini pesanan untuk pesanan penjualan berdasarkan lini booking.

* **Kaitannya**: Setiap lini pesanan akan mengaitkan course\_booking\_line\_id, yang menghubungkan kembali ke detail booking kursus.

**8. Metode action\_sale\_order\_view**

def action\_sale\_order\_view(self):

action = self.env['ir.actions.act\_window'].\_for\_xml\_id('sale.act\_res\_partner\_2\_sale\_order')

action["domain"] = [("course\_booking\_id", "in", self.ids)]

action["context"] = {'create': 0, 'edit': 0}

return action

* **Logika**: Membuka tampilan semua pesanan penjualan yang terkait dengan booking.

* **Kaitannya**: Memungkinkan pengguna untuk dengan mudah mengakses dan melihat semua pesanan penjualan yang terhubung dengan booking tertentu.

**Kesimpulan**

Secara keseluruhan, model Anda terstruktur dengan baik untuk menangani pengelolaan booking kursus dan integrasinya dengan sistem pemesanan penjualan Odoo. Dengan menambahkan field yang relevan dan metode yang sesuai, Anda telah menciptakan relasi yang logis antara kursus dan penjualan, memungkinkan pelacakan dan pengelolaan yang lebih efisien. Jika Anda memiliki pertanyaan lebih lanjut atau ingin membahas aspek tertentu lebih mendalam, silakan beri tahu!