**all ORM fields are inside a fields.py**

D:\Projects\v18\odoo\odoo\fields.py

**Q 5: Model fields and its properties**

**1. Char**

* **Description**: A string field for short text (e.g., a name).
* **Properties**:
  + string: Label for the field.
  + size: Maximum length of the string (default is 256).
  + required: Boolean to make the field mandatory.
  + default: Default value.
  + readonly: Boolean to make the field readonly.
  + index: Boolean to create an index in the database.
* Example:
  + name = fields.Char(string="Name", size=128, required=True)

**2. Text**

* **Description**: A field for longer text.
* **Properties**:
  + string: Label for the field.
  + required: Boolean to make the field mandatory.
  + readonly: Boolean to make the field readonly.
* Example:
  + description = fields.Text(string="Description")

**3. Boolean**

* **Description**: A field that stores True/False values.
* **Properties**:
  + string: Label for the field.
  + default: Default value (True/False).
  + required: Boolean to make the field mandatory.
* Example:
  + is\_active = fields.Boolean(string="Is Active", default=True)

**4. Integer**

* **Description**: A field for storing integer numbers.
* **Properties**:
  + string: Label for the field.
  + required: Boolean to make the field mandatory.
  + default: Default integer value.
* Example:
  + age = fields.Integer(string="Age", required=True)

**5. Float**

* **Description**: A field for storing floating point numbers.
* **Properties**:
  + string: Label for the field.
  + digits: Defines the number of decimal places (e.g., (12, 2) for a maximum of 12 digits with 2 decimals).
  + required: Boolean to make the field mandatory.
  + default: Default floating point value.
* Example:
  + price = fields.Float(string="Price", digits=(12, 2))

**6. Selection**

* **Description**: A field for selecting one value from a predefined list of options.
* **Properties**:
  + string: Label for the field.
  + selection: A list of tuples with the options (e.g., [(1, 'Option 1'), (2, 'Option 2')]).
  + default: Default selected value.
  + required: Boolean to make the field mandatory.
* Example:
  + status = fields.Selection(
  + selection=[('draft', 'Draft'), ('confirmed', 'Confirmed')],
  + string="Status", default='draft'
  + )

**10. Date**

* **Description**: A field for storing a date.
* **Properties**:
  + string: Label for the field.
  + required: Boolean to make the field mandatory.
  + default: Default date (either a fixed date or computed).
* Example:
  + start\_date = fields.Date(string="Start Date", default=fields.Date.today)

**11. Datetime**

* **Description**: A field for storing a timestamp (date and time).
* **Properties**:
  + string: Label for the field.
  + required: Boolean to make the field mandatory.
  + default: Default datetime value.
* Example:
  + start\_datetime = fields.Datetime(string="Start Date & Time")

**12. Binary**

* **Description**: A field for storing binary data (e.g., files, images).
* **Properties**:
  + string: Label for the field.
  + attachment: Boolean to store the binary data as an attachment.
* Example:
  + image = fields.Binary(string="Image", attachment=True)

**13. HTML**

* **Description**: A field for storing HTML content.
* **Properties**:
  + string: Label for the field.
  + required: Boolean to make the field mandatory.
* Example:
  + description\_html = fields.Html(string="Description HTML")

**14. Computed Fields**

* **Description**: A field whose value is computed dynamically based on other fields.
* **Properties**:
  + string: Label for the field.
  + compute: The method that calculates the field's value.
  + store: Boolean to store the computed value in the database.
  + inverse: Method to set the value back if the computed field is editable.
* **Example:**

total\_amount = fields.Float(string="Total Amount",compute="\_compute\_total\_amount", store=True)

@api.depends('order\_lines')

def \_compute\_total\_amount(self):

for record in self:

record.total\_amount = sum(line.price\_subtotal for line in record.order\_lines)

**15. Related Field**

* **Description**: The **related** field is often used to access data from a related model by following a chain of relationships, and it works similarly to a computed field in that the value is automatically updated when the related model's field is changed.
* **Properties**:
* **related**: This specifies the field in the related model (through a chain of relationships). It defines how to get the value of the related field.
* **string**: This is the label that will appear on the form view or list view.
* **type**: This specifies the field type (e.g., Char, Integer, Float, etc.).
* **store**: Whether or not the value should be stored in the database. This is useful when you want the related field to behave like a regular field that is stored persistently. If set to False, it behaves like a computed field that is recalculated when needed.
* **readonly**: If set to True, the field becomes read-only in the UI, meaning the value cannot be edited directly.
* Example:

from odoo import models, fields

class SaleOrder(models.Model):

\_name = 'sale.order'

name = fields.Char("Order Name")

partner\_id = fields.Many2one('res.partner', string="Customer")

# Related field to access partner's phone

partner\_phone = fields.Related(

'partner\_id.phone', # Field 'phone' in the related model 'res.partner'

type='Char', # The type of the field is Char (phone number)

string="Customer Phone", # Label to display in the UI

store=False # Do not store the value in the database

)

**16. Monetary Field**

* **Description**:
  + When you need to store **financial data** like prices, amounts, or totals and associate them with a **specific currency**.
* When you want **automatic currency formatting** and **precision management** based on the current user's company settings or a specific currency.
* When you need to **ensure accurate rounding** and **currency-specific behavior** for monetary values.
* Properties:
  + **currency\_field**:
  + This specifies the field that holds the related currency. The currency\_field should be a Many2one field linking to the res.currency model, which contains the currency information.
  + **string**: The label to display for the field in views (e.g., "Amount" or "Price").
  + **required**: Boolean value (True or False) that determines whether the field must be filled in before saving the record.
  + **store**: Boolean value (True or False) indicating whether the value should be stored in the database. By default, it's set to False to calculate it dynamically.
  + **precision**: Defines the number of decimal places for the monetary value. It is controlled by the currency and is usually set automatically.
  + **readonly**: Boolean value (True or False) that indicates whether the field is read-only.
* Example:
  + from odoo import models, fields
  + class SaleOrder(models.Model):
  + \_name = 'sale.order'
  + name = fields.Char("Order Name")
  + amount\_total = fields.Monetary(
  + string="Total Amount",
  + currency\_field='currency\_id', # Field that links to the currency
  + required=True
  + )
  + currency\_id = fields.Many2one('res.currency', string="Currency", default=lambda self: self.env.company.currency\_id)

**17. Reference Field**

* Description: the **Reference** field type is used to create a flexible link to any record in any model. Unlike standard relational fields (e.g., Many2one, One2many, Many2many), the **Reference** field does not point to a specific model but instead allows the user to choose from a list of possible models at runtime. This provides flexibility when you need to create links to multiple models without explicitly defining the relationship.
* Properties:
  + **selection**: This is a method that returns a list of tuples representing the models that can be referenced. Each tuple consists of the model name and a label that will be shown in the UI.
  + **string**: The label for the field in the UI (e.g., "Reference").
* Example:
  + from odoo import models, fields
  + class MyModel(models.Model):
  + \_name = 'my.model'
  + reference\_field = fields.Reference(
  + selection='\_get\_reference\_models', # Method to return available models
  + string='Reference'
  + )
  + def \_get\_reference\_models(self):
  + return [
  + ('res.partner', 'Partner'), # Allows selecting a Partner
  + ('product.product', 'Product'), # Allows selecting a Product
  + ]
* Accessing the Referenced Record
* class MyModel(models.Model):
* \_name = 'my.model'
* reference\_field = fields.Reference(
* selection='\_get\_reference\_models',
* string='Reference'
* )
* def get\_reference\_record(self):
* for record in self:
* if record.reference\_field:
* model\_name, record\_id = record.reference\_field
* related\_record = self.env[model\_name].browse(record\_id)
* # Do something with the related record
* return related\_record