Requirement - 1

Create new module res\_localization\_ksc

* Model
  + res.country.ksc
* Fields
  + Name of the country - Char
  + Short Code of the country - Char
  + States - One2many - res.state.ksc
* Action
* Menu
  + Countries
  + Parent Menu - Localization -> Localization
* View
  + Tree, Form, Search
* Search View
  + One should be able to search by country name
  + One should be able to search by country code
* Security - Create appropriate security group

--------------------

* Model
  + res.state.ksc
* Fields
  + Name of the state - Char
  + Short Code of the state - Char
  + Country - M2O (res.country.ksc)
  + Cities - One2many (res.city.ksc)
* Action
* Menu
  + States
  + Parent Menu - Localization -> Localization
* View
  + Tree, Form, Search
* Search View
  + One should be able to search by state name
  + One should be able to search by state code
  + One should be able to search by country
  + Group by Country

---------------------

* Model
  + res.city.ksc
* Fields
  + City Name
  + State - M2O (res.state.ksc)
* Action
* Menu
  + City
  + Parent Menu - Localization -> Localization
* View
  + Tree, Search
* Tree - Make tree editable, we don’t need form view
* Search View
  + One should be able to search by city name
  + One should be able to search by state
  + Group by state

Requirement - 2

Create new module name employee\_mgmt\_ksc

* Model
  + employee.department.ksc
* Fields
  + Department Name - Char - required - True
  + Employees - One2many
  + Department Manager - M2O - res.users
* Action
* Menu
  + Department
  + Parent Menu - HR -> Employees
* View
  + Tree, Form, Search
* Search View
  + One should be able to search by Department name
* Security
  + Create a group employee

------------------

* Model
  + employee.department.shift.ksc
* Fields
  + Shift - Selection - ( Morning, Afternoon, Evening, Night)
  + employee\_ids - O2M -
* \_rec\_name = Shift
* Action
* Menu
  + Employee Shifts
  + Parent Menu - HR -> Employees
* View
  + Tree, Search
  + Tree - Make tree editable, we don’t need form view
* Search View
  + One should be able to search by shift name

-----------------------

* Model
  + employee.ksc
* Fields
  + Name of the employee - Char - required - True
  + Department Name of the employee - Many2one(employee.department.ksc) - required - True
  + Shift - Many2one(employee.department.shift.ksc) - required - True
  + Job Position of employee - Char
  + Salary - Float, digits(6,2)
  + Hire Date - Date
  + Gender - (Male, Female, Transgender)- Selection
  + Job Type - (Permanent, Ad Hoc) - selection
  + Is\_manager - Boolean
  + Manager - M2O - employee.ksc - nocreate - no open
  + Related User - M2O - res.users
  + employee\_ids - O2M - employee.ksc - based on manager\_id - readonly
  + increment\_percentage - Float - digits - 2 - can only be visible to manager group
* Action
* Menu
  + Employees
  + Parent Menu - HR -> Employees
* View
  + Tree, Form, Search
* Search View
  + Employee Name, Job Position, Department Name
  + Filter - Job Type (Show example of domain)
  + Group By - Department Name, Job Type, Gender

------

* Model
  + employee.leave.ksc
* Fields
  + Employee - M2O - employee.ksc
  + Department - M2O - employee.department.ksc
  + Start date - Date
  + End date - Date
  + Status - Draft, Approved, Refused, Cancelled - default - Draft
  + Leave Description - Char - required - True
* Action
* Menu
  + Leaves
  + Parent Menu HR -> Employees
* View
  + Tree, Form, Search
* Search view
  + Employee, Department
  + Group by - Employee, Department
* Security -
  + The Department Manager should see the leaves of all the employees including managers and employees of his department.
  + Manager should be able to see the leaves of its employees
  + Employee should be able to see his leave only

Requirement - 3

Create new module named sale\_ksc

* Model
  + product.category.ksc
* Fields
  + name - char - required
  + parent\_id - M2O product.category.ksc
* Action
* Menu
  + Product Category
  + Parent Menu - My Sales -> Category

---------------

* Model
  + product.ksc
* Fields
  + name - char - required
  + sku - char - required
  + weight - float - 2 decimals
  + length - float - 2 decimals
  + volume - Float - 2 decimals
  + width - float - 2 decimals
  + barcode - char
  + product\_type - selection - (Storable, Consumable, Service)
  + sale\_price - Float - 2 decimals - default 1.00
  + cost\_price - Float - 2 decimals - default 1.00
  + Product Category - M2O - product.category.ksc
  + UOM - M2O - product.uom.ksc
* Action
* Menu
  + Products
  + Parent Menu - My Sales -> Sales
* View
  + Tree, Form, Search
* Search View -
  + Name,sku,barcode

------------

* Model
  + product.uom.category.ksc
* Fields
  + name - char
  + uom\_ids - O2M - product.uom.ksc
* Menu
* Action
* View - tree, form, search
* Search - name

-----------------

* Model
  + product.uom.ksc
* Fields
  + name - char
  + uom category - M2O product.uom.category.ksc
* Menu
* Action
* View - tree, form, search
* Search - name

----------

* Model
  + res.partner.ksc
* Fields
  + Name - Char
  + Street 1 - Char
  + Street 2 - Char
  + Country - M2O - res.country.ksc
  + State - M2O - res.state.ksc
  + City - M2O - res.city.ksc
  + Zip Code - Char
  + Email - Char
  + Mobile - Char
  + Phone - Char
  + Photo - Image
  + Website - Char
  + Active - boolean
  + parent\_id - M2O - res.partner.ksc
  + child\_ids - O2M - res.partner.ksc - show kanban view - based on parent\_id
  + Address Type - Selection(Invoice, Shipping, Contact) - show this on child form only
* Action
* Define Action
* Menu
  + Customer
  + Parent Menu - My Sales -> Sales
* View
  + Tree, Form, Search
* Search View -
  + On tree view list only main customer not the child customer, the default view should be like that, do appropriate changes in 'action'
  + Name, Country, State, Email
  + Group By - Country, State, City
  + Filter - Active, Inactive

----------

* Model
  + sale.order.ksc
* Fields
  + Order No (name) - Char - required
  + Customer - M2O - res.partner.ksc - domain - only main customer should be listed - required
  + Invoice Customer - Many2one(res.partner.ksc), domain - Address Type should be Invoice - required and should be of selected main customer only
  + Shipping Customer - Many2one(res.partner.ksc), domain - Address Type should be Shipping - required and should be of selected main customer only
  + Sale Order Date - Date
  + Order Line - One2many - sale.order.line.ksc
  + Salesperson - M2O - res.users
  + State - draft, confirmed, done, cancelled
* Action
* Menu
  + Orders
  + Parent Menu - My Sales -> Sales
* View
  + Tree, Form, Search
* Search View -
  + Order No, Customer, Sale Order Date
  + Group by - Customer
* Security
  + Groups - Salesperson - salesperson should only be able to view their own orders
  + Manager - should see all salesperson orders

---------

* Model
  + sale.order.line.ksc
* Fields
  + Order No - M2O - sale.order.ksc
  + Product - Many2one(product.ksc)
  + Name - label - ‘Description’ - Text
  + Quantity - Float, digits(6,2)
  + Unit Price - Float, digits(6,2)
  + State - selection - draft, confirmed, cancelled
  + uom\_id - M2O - product.uom.ksc

Requirement - 4

Considering Requirement - 3

* Add unique \_sql\_constraint on country\_code field for res.country.ksc
* Add field state\_code in res.state.ksc model. Add ‘api.constrain’ to restrict unique values for state\_code field
* When product is selected on sale.order.line.ksc, fetch the unit price from product model, also set the quantity as 1 - use onchange method
* subtotal\_without\_tax - Float - computed field which is stored in database, which depends on order line qty and order line unit price
* Add field total\_weight in sale.order.ksc model, which should be computed field with 2 decimals and should not be stored in database - it will calculate the weight from sale.order.line.ksc -> product\_id field
* Add a field total\_volume with 2 decimals in sale.order.ksc, which should be computed field and should not be stored in the database - it will calculate the weight from sale.order.line.ksc -> product\_id field
* Add total\_weight and total\_volume field in sale.order tree view
* Add order\_total in sale.order.ksc - Float - compute, store true field - depends on subtotal\_without\_tax field of sale.order.line.ksc model
* Add on change on customer field, it should bring invoice and shipping address automatically, if multiple invoice or shipping address are associated with customer then bring the first ones from the list

Requirement - 5

Consider requirement 3 & 4 and code it in same module

* Model - crm.team.ksc
* Fields
  + Name - Char
  + Team leader - M2O - res.users
* Action
* View
* Menu -
  + Sales Team
  + Parent - My Sales -> CRM

-----------------

* Bring crm.lead.ksc model and its view to this module
* Model - crm.lead.ksc
* Fields
  + partner\_id - M2O - res.partner.ksc
  + order\_ids - O2M - keep it readonly, record will not be generated from here, it is just for the display purpose
  + lead\_line\_ids - O2M - crm.lead.line.ksc - inverse name - lead\_id - Use inline tree view

------------------

* Model - crm.lead.line.ksc
* Fields
  + product\_id - M2O - product.ksc
  + name - Description - product\_id.name - use onchange of product
  + expected\_sell\_qty - Float
  + uom\_id - M2O - product.uom.ksc
  + lead\_id - M2O - crm.lead.ksc

--------

* Model - sale.order.ksc
* Fields
  + lead\_id - M2O - crm.lead.ksc
* Add buttons to mark the lead as won or lost
* When the lead is in the won state provide the button to generate the Sales quotation with partner details
* When the Generate Sales Quotation button is clicked make sure partner\_id is already set/created. If that is not created, raise a warning and ask to create a customer first.
* If customer is already created, then create new sales order with that partner and also find the appropriate invoice partner and shipping partner
* From the button of “Generate Sales Quotation” in sales order generation, take products from crm.lead.line for sale.order.line

Requirement - 6

* Model
  + stock.warehouse.ksc
* Fields
  + Name - Char - required
  + Short Code - Char - required
  + address - M2O - res.partner.ksc
  + stock\_location\_id - M2O - stock.location.ksc - allow only “internal” type of location - label - Stock Location
  + view\_location\_id - M2O - stock.location.ksc - allow only “view” type of location
* Menu
* Action
* Views - Tree, Form, Search

------------

* Model
  + stock.location.ksc
* Fields
  + name - Char - required
  + parent\_id - M2O - stock.location.ksc
  + location\_type - Selection - Vendor, Customer, Internal, Inventory Loss, Production, Transit, View
  + is\_scrap\_location - Boolean
* Menu
* Action
* View - Tree, Form, Search

------------------

Model - stock.picking.ksc

* name - Char - Generate the sequence and should automatically be generated when record is created
* partner\_id - M2O - res.partner.ksc - here the shipping partner of sales order should be set
* back\_order\_id - M2O - stock.picking.ksc
* state - Selection(draft, validate, cancelled) - default - draft
* sale\_order\_id - M2O - sale.order.ksc
* purchase\_order\_id - M2O - purchase.order.ksc
* transaction\_type - Selection - [In, Out]
* move\_ids - O2M - stock.move.ksc - inverse\_name - picking\_id
* transaction\_date - Date - default - Today

------------

Model - stock.move.ksc

* name - Char - label - “Description” - “Product Name : Source Location -> Destination Location”
* product\_id - M2O - product.ksc - required
* uom\_id - M2O - product.uom.ksc - required
* source\_location\_id - M2O - stock.location.ksc
* destination\_location\_id - M2O - stock.location.ksc
* qty\_to\_deliver - Float
* qty\_delivered - Float
* state - Selection - (draft, done, cancelled) - default - draft
* sale\_line\_id - M2O - sale.order.line.ksc
* purchase\_line\_id - M2O - purchase.order.line.ksc
* stock\_inventory\_id - M2O - stock.inventory.ksc
* picking\_id - M2O - stock.picking.ksc

-------------

Model - sale.order.ksc

Fields

* warehouse\_id - M2O - stock.warehouse.ksc

Model - purchase.order.ksc

Fields

* warehouse\_id - M2O - stock.warehouse.ksc

Add a button in purchase order to confirm the order. When the purchase order is confirmed, create Incoming Shipment (stock.picking.ksc & stock.move.ksc) records. Data from purchase.order will be set in stock.picking(partner\_id, purchase\_order\_id). Data from purchase.order.line.ksc will be set to stock.move(product\_id, uom\_id,qty, purchase\_line\_id) For stock.picking, set the transaction\_type as “In”. For stock.move, find the source location from stock.location of type vendor. The destination location will be stock\_location\_id from warehouse set in purchase.order.ksc

Add a button in sales order to confirm the order, when the order is confirmed, change the state of each sales order line to confirm, state of order to confirm, create delivery order(stock.picking.ksc) and create stock move from sales order line

You should create a customer location with location type as “Customer”. For delivery order, the system should find the location of type “Customer” and set it as “destination location” in stock.move.ksc. The source location will be the “stock location” set in the warehouse.

--------------

Model - product.ksc

product\_stock - Float - compute - store False

It should go through all the warehouses available in the system and then take the stock location associated with it and then scan the stock moves and calculate the stock and show it on the screen.

When a context is passed with a key as ‘location’, then search for stock moves with that location only. If context with key ‘location’ is not found then scan all the warehouse and its location and sum up the data

---------

Model - stock.inventory.ksc

name - Char

state - Selection(draft, in-progress, done, cancelled)

location\_id - M2O - stock.location.ksc - location where the adjustment is to done and qty will be calculated for this location

inventory\_date - Date - default value should be today’s date inventory\_line\_ids - O2M - stock.inventory.line.ksc

Add a button to “Start Inventory”, which will set the available quantities in all the products in stock.inventory.line. It should be visible only in draft state. Change the state to in-progress Add a button to validate the inventory, which will generate the stock moves in ‘done’ state directly. This button should be visible in in-progress state only Once the inventory is validated, it cannot be cancelled

----------------

Model - stock.inventory.line.ksc

product\_id - M2O - product.ksc

available\_qty - Float

counted\_product\_qty - Float

difference - Float - compute (counted\_product\_qty - available\_qty)

Requirement - 7

Model - stock.picking.ksc

Fields

parent\_picking\_id - M2O - stock.picking

When delivery order is validated, for whichever products it is partially delivered, transfer it to backorder, with all those products with required products to be delivered.

Make sure all these delivery orders are associated with the same sales order Make sure all the stock moves of all associated delivery orders are associated with appropriate sale order lines

Model - sale.order.ksc

Fields -

picking\_ids - O2M - stock.picking.ksc - readonly

Model - sale.order.line.ksc

stock\_move\_ids - O2M - stock.move.ksc

delivered\_qty - Float - compute store - It should scan through all the validated delivery orders and calculate the the total delivered quantities

Requirement - 8

* We need to update the stock of the specific product from form view using wizard(TransientModel).
* Model - product.stock.update.ksc
* Fields -
  + location\_id - M2O - stock.location.ksc - add onchange to find the current stock
  + current\_stock - Float
  + counted\_qty - Float
  + difference\_qty - Float - compute (counted - current) - store - False
* Menu
* Action
  + Check for binding\_model\_id, binding\_type, binding\_view\_types, target, view\_id
* View
  + Form View
  + On the form view add two buttons in footer “Update Stock” and “Cancel”
  + When Update Stock is clicked, it should generate the record for stock.inventory.ksc and stock.inventory.line.ksc model with appropriate values in it.
  + Thereafter it should validate the stock.inventory.ksc record which got generated.
  + Cross check the stock.inventory.ksc record and its generated stock moves.
* Security
  + Make sure you assign the security to defined transient model of update product stock

Requirement - 9

* Model
  + product.ksc
* Add a button of type ‘object’ on the form view of the product.
* Clicking on that button shall open the wizard for updating the stock of that product
* Use the same wizard model which was used in Requirement - 8
* Use the same form view which was used in Requirement - 8
* Refer the core code to return “Action” from button click

Requirement - 10

* Model
  + product.ksc
* Add a button of type ‘action’ on the form view of the product
* Clicking on that button shall open the wizard for updating the stock of that product
* Use the same wizard model which was used in Requirement - 8
* Use the same form view which was used in Requirement - 8

Requirement - 11

* Model
  + sale.order.ksc
* Add button-box at the top right corner of the screen for Delivery order
* If there are more than one delivery order, open list view of the delivery order with listing only delivery orders which are associated with that sales order
* If there is only one delivery order, open the form view for that delivery order
* Add button-box at the top right corner of the screen for Stock move
* If there are more than one stock move, open list view of the stock move with listing only stock moves which are associated with that sales order
* If there is only one stock move, open the form view for that stock move

Requirement - 12

* Model
  + sale.order.line.ksc
* Fields
  + warehouse\_id - M2O - stock.warehouse.ksc
* For each sale order line set the warehouse and based on that create/split the delivery order
* There will be separate delivery orders for each warehouse set in the sales order line. Order lines having same warehouse should be grouped together

Requirement - 13

* Model
  + account.tax.ksc
* Fields
  + name - Char
  + tax\_use - Selection - (None, Sales, Purchase)
  + tax\_value - Float - label - Amount
  + tax\_amount\_type - Selection - (Percentage, Fixed) - default - Percentage
* Action
* Menu
* View
  + Tree, Form, Search
  + Group by tax\_use, tax\_amount\_type

-------------

* Model
  + product.ksc
* Fields
  + tax\_ids - Many2many - label - Customer Taxes - domain - load only Sales Type of tax
  + Add a notebook - add page with string “Accounting” - add tax\_ids field over there as many2many\_tags widget

-----------

* Model
  + sale.order.line.ksc
* Fields
  + tax\_ids - Many2many - use many2many\_tags widget
  + subtotal\_tax - Float - decimal 2 - compute - store - true - depends on - qty, unit price
* When product is selected, bring the taxes from product and set it in the sales order line - use onchange of product\_id

---------------

* Model
  + sale.order.ksc
* Fields
* total\_tax - Float - decimal - 2 - computed - depends on subtotal\_tax field of order\_line\_ids - store - true
* total\_amount - Float - decimal - 2 - formula addition of subtotal\_tax from sale.order.line.ksc, compute store true, depends on order\_line\_ids

===================

Requirement - 14

* Model
  + Sale.order
* Module - sale\_order\_extended
* In sale order, add M2O field for crm.lead (there is already a M2O field of crm.lead, so make sure to give different name of the field)
* When new sales quotation gets created from crm.lead perform following operations
  + Set M2O field of lead in sale.order
  + There is already a field named ‘tag\_ids’ in sale.order, set tag "From Lead” in that field (create this tag record from xml)

Requirement - 15

* Model
  + Sale.order
* Install Delivery module
* Create a product using xml
* On confirm button on sale.order form, the shipping product should be added automatically to sale.order.line model
* Inherit the appropriate method of order confirmation of sale.order model and add the logic to add the shipping line automatically
* Make sure Shipping line should be added only once - try to use “is\_delivery” field in sale.order.line model

Requirement - 16

* Model
  + product.product
* Fields
  + deposit\_product\_id - M2O - product.product
  + deposit\_product\_qty - Integer
* On the product form view (main product), add deposit\_product\_id field and deposit\_product\_qty field.
* Sale.order and sale.order.line
  + When the main product is added on the sale.order.line
  + Add a button named “Manage Deposits”
  + When “Manage Deposits” button is clicked, scan all the products and check if there is any deposit product associated with the product. If there is any, then that product should be added to the sale.order.line. If once the deposit product is added already, then it should not be added more than once.
  + If the quantity of main product is updated, make sure to update the quantity of deposit product also
  + When main product order line is deleted, make sure the deposit line is also deleted

Requirement - 17

* Model
  + sale.order
* Add button of type object on the sale order form view.
* When the button is clicked, scan all the sale.order.line and get products from them.
* Search for other sale.order.line from other sale orders which are reserved and yet not done or cancelled and show the list view of sale.order.line

Requirement - 18

* Model
  + sale.order
* Add a button of type object in sale.order
* When that button is clicked, try to perform following operations
* Confirm the order
* Validate the delivery order

Requirement - 19

* Model
  + sale.order
* Fields
  + is\_all\_picking\_completed - Boolean - Compute - Store - False
* This field should be able to search whether all the related pickings(Delivery orders) are validated or not. If any one of the picking is not validated then this field should not be set to True, in case all the pickings are validated then it should be marked as True. Any picking is cancelled, that should also be considered as completed. In summary any picking in draft, waiting, partial available state should return False
* View
  + Inherit search view and perform the above operation
  + Add is\_all\_picking\_completed Filter to search view

Requirement - 20

Generate a Qweb report for showing the country name and country code

Requirement - 21

Generate a Qweb report for sale.order model, which should be similar to the existing “Quotation / Order” report.

Requirement - 22

Trace the process of managing Tax in sale.order model

Requirement - 23

Refer the picking operation report from stock.picking print menu for delivery orders. Create appropriate class and apply appropriate logic

Users will select the delivery orders from the delivery order list view and based on that data the report should be generated.

Based on this create a new report, where data shown product wise as follows:

Large Cabinet

**Picking** **Quantity**

---------------------------------------------------------------------------------------------------------------------------

WH/OUT/0001 2

WH/OUT/0002 5

WH/OUT/0007 5

---------------------------------------------------------------------------------------------------------------------------

**Total** **12**

---------------------------------------------------------------------------------------------------------------------------

Pen

**Picking** **Quantity**

---------------------------------------------------------------------------------------------------------------------------

WH/OUT/0002 2

WH/OUT/0003 2

WH/OUT/0007 3

---------------------------------------------------------------------------------------------------------------------------

**Total** **7**

---------------------------------------------------------------------------------------------------------------------------

Pencil

**Picking** **Quantity**

---------------------------------------------------------------------------------------------------------------------------

WH/OUT/00013 3

WH/OUT/0001 8

WH/OUT/0007 3

---------------------------------------------------------------------------------------------------------------------------

**Total** **14**

---------------------------------------------------------------------------------------------------------------------------

Requirement - 24

**Stock**

Large Cabinet

**Picking** **Quantity**

---------------------------------------------------------------------------------------------------------------------------

WH/OUT/0001 2

WH/OUT/0002 5

WH/OUT/0007 5

---------------------------------------------------------------------------------------------------------------------------

**Total** **12**

---------------------------------------------------------------------------------------------------------------------------

Pencil

**Picking** **Quantity**

---------------------------------------------------------------------------------------------------------------------------

WH/OUT/0002 2

WH/OUT/0003 2

WH/OUT/0007 3

---------------------------------------------------------------------------------------------------------------------------

**Total** **7**

---------------------------------------------------------------------------------------------------------------------------

Shelf 1

Pencil

**Picking** **Quantity**

---------------------------------------------------------------------------------------------------------------------------

WH/OUT/00013 3

WH/OUT/0001 8

WH/OUT/0007 3

---------------------------------------------------------------------------------------------------------------------------

**Total** **14**

---------------------------------------------------------------------------------------------------------------------------

Large Cabinet

**Picking** **Quantity**

---------------------------------------------------------------------------------------------------------------------------

WH/OUT/00013 3

WH/OUT/0001 8

WH/OUT/0007 3

---------------------------------------------------------------------------------------------------------------------------

**Total** **14**

---------------------------------------------------------------------------------------------------------------------------

Requirement - 25

Calculate Sales Profit Margin

In the sale.order model considering the cost price(standard price) field of product, calculate the profit margin percentage and profit value for each sale order line.

Also calculate the total profit margin percentage and profit value at sales order level.

Find the formula to calculate the profit margin percentage on your own and apply it.

Profit value = sale price - cost price

Cost price and sales price keep changing, so make sure for those sales order where the profit margin and profit value are calculated once, it should not be changed in future when cost price or sales prices are changed.

Requirement - 26

Create a fresh product with its variants

Add stock to those variants

In sale.order model, add a field product\_tmpl\_ids - m2m

Apply on change method to product\_tmpl\_ids field. When onchange is fired add all the variants of that template into sale.order.line

While adding the variants to the sale.order.line, check if stock of those variants is available or not, skip the variants whose stock is not available and don’t add it to the sale.order.line Stock should be checked for warehouse set in the sale.order

Requirement - 27

* Install Project module
* It contains the model of project.tag
* Try to add a M2M compute store false field in project.tag based on project.task model should show the users who worked on specific tag
  + Example :
    - Bug - John, Adam, Michael
    - New Feature - Adam, Michael
    - Support - John

Requirement - 28

Model - sale.order.line

Odoo generates delivery orders based on the warehouse. It creates separate delivery orders for separate warehouses.

We need to add a M2O field warehouse\_id in sale.order.line model and when order is confirmed, need to make sure that separate delivery orders are created as per the warehouses configured in sale.order.line model

If the warehouse is not set in the sale.order.line model, then consider the warehouse from sale.order model

Requirement - 29

Generate the following report

Sales By Salesperson

Create a wizard which should have

From Date - Date

To Date - Date

Salesperson - M2M

Create a new menu in the sales app.

Based on the above data it should be able to generate the following report

Report By Salesperson

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Order Dates** : <from date> To <to date> | | |  |
|  |  | **<name of salesperson>** |  |  |
| **Order No** | **Customer** | **Order Date** | **Total Tax** | **Total Amount** |
| 001 | AAA | 02-02-2021 | 230 | 570 |
| 002 | BBB | 02-02-2021 | 235 | 670 |
| 003 | CCC | 02-02-2021 | 237 | 470 |

------------------------------------------------------------------------------------------------------------------------

|  |  |
| --- | --- |
| **(show currency of the order) 702** | **1710** |

------------------------------------------------------------------------------------------------------------------------

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | **<name of salesperson>** |  |  |
| **Order No** | **Customer** | **Order Date** | **Total Tax** | **Total Amount** |
| 004 | DDD | 02-02-2021 | 130 | 770 |
| 005 | EEE | 02-02-2021 | 135 | 470 |
| 006 | FFF | 02-02-2021 | 137 | 870 |

------------------------------------------------------------------------------------------------------------------------

|  |  |
| --- | --- |
| **(show currency of the order) 402** | **2110** |

------------------------------------------------------------------------------------------------------------------------

Requirement - 30

Model - sale.order.line

Add a feld history\_unit\_price in sale.order.line

When someone try to add the product to sale.order.line, try to bring the unit price from last confrmed sale order of same customer and set it the new feld mentioned above Model - purchase.order.line

Add a feld history\_unit\_price in purchase.order.line

When someone try to add the product to purchase.order.line, try to bring the unit price from last confrmed purchase order of same vendor and set it the new feld mentioned above

Requirement - 31

1. Import Sales Order fle in a user friendly way - fnd the attached fle.
2. Import customers based "Customer Code" - Customers must be unique
3. Maintain logs for the operations, make sure your logs are easily understood
4. Confgure default sales tax as 10%
5. A confguration should be provided to add "Shipping Product" which will be added during fle import, under Sales confguration screen
6. Calculate proft at order level based on the "Cost per unit" values
7. The shipping charge line must include the SKU of actual product that is being shipped - [<SKU>], along the shipping product description
8. Whole fle must be processed, any errors must be handled wisely, the operations should not crash
9. The order line containing zero quantities must be skipped, remember order must not be skipped
10. Don't use any existing solution, available from any source
11. The screen(for Import fle) you develop should be as follows:
    * Master Operations at top area of the screen for uploading a fle
    * Create a Job Log which contains the log lines of the operations, such as order creation for success or error.
    * Those Log lines are also to be shown in the same screen below the operational area as a grid.
    * Provide a navigation button on the top right corner for the sales order created while importing the fle.
    * The sales import fle must be attached to Job of Log created.
    * Provide a navigation button at the top to navigate to the Job associated with this import of fle.
    * One import contains one Job for log.

**Ref: How to read csv fle?**

**https://realpython.com/python-csv/#reading-csv-fles-with-csv**