

Impactic Odoo Test Exercise

Introduction

The test has been split into two tasks starting with a pure Python exercise, followed by a task that goes more in depth with Odoo itself. The task will be done using Odoo 17 Community Edition. A Docker Compose file and Odoo conf file will be provided for deployment. The final task is not mandatory to be completed entirely, as it dives deeper into some specifics of Odoo's development, however it is encouraged to try to get as far as possible there. Even if you are not able to finish the tasks, **asking the appropriate questions** may earn you bonus points. And even if we didn't want to (...but we do), we will also be looking at your **coding style**, so **structure and comment your code** as if someone else would have to be able to understand and modify it easily.

Please read through entire test exercise before you start dig deeper.

Throughout the exercise, various references to documentation and code will be provided. Even though the documentation covers a large number of topics, it is highly advised to also clone [Odoo's official GitHub repository](#) to your development environment, where you can find specific implementation examples of Odoo's base code, which you will be extending in Task 2. We definitely encourage you to not only look at it, but to also dive deeper into the base code itself - it will be a every-day part of your development to explore and get to know the base code better.

If you have any questions about Odoo's ORM and framework or if you have any difficulty in deploying Odoo, feel free to contact me via tonu.kaare@impactic.ee.

Prerequisites

Before you begin with the tasks, we recommend you prepare an Odoo deployment and a development environment first. We suggest using any UNIX-based OS (Ubuntu, Debian, OS X) for development, as deploying Odoo with Docker could be somewhat more difficult on Windows machines.

Install Docker

Follow [Docker](#) official installation instructions.

It is also highly recommended you enable [command-line completion](#) if it is not yet enabled for your OS user. You will most likely be operating Docker from the root user unless you've set-up other users manually, so also check if completion is enabled there.

Python and IDE

Use your favorite IDE for Python development. In your project/environment settings, make sure you are using Python 3.10+

Configuring and Running Odoo

Download the zipped (.zip) file from [here](#) and extract it in the directory you want the Docker instance to be in.

Inside extracted folder you find:


- `docker-compose.yml` file. No need to adjust it. You can change port in left hand side if port 8069 is already in use.
- `odoo.conf` file. No need to adjust it but you may. You can find other possible configuration variables and explanation to current ones from [here](#). If you run into memory or CPU time limit problems, check the parameters `limit_memory_hard`, `limit_memory_soft`, `limit_request`, `limit_request` and `limit_time_real`.
- `addons` folder. **The custom modules you will develop should be put under this directory.** If you want, you can also separate modules into subdirectories there, but you should also then modify the configuration in `odoo.conf` accordingly, by adding all the

subdirectories to the addons path separated by ';' - they are mounted to /mnt/extra-addons/* inside the Docker instance, so no extra mounting will be required.

You can start the server by running the following command in the directory the docker-compose.yml file is in:

```
1 docker compose up -d
```

Now, try to open <http://localhost:8069/> in your browser. If everything went correctly, you should be greeted with the Odoo database creation page, which should look like this (with empty inputs):



Master Password

Database Name

Email

Password

Phone Number

Language

Country

Demo Data ☐

[Create database](#) [or restore a database](#)

If you need to change the config and redeploy it, use the following command:

```
1 docker compose down && docker compose up -d
```

If you develop Python modules, you will need to regularly restart the server, which can be done with the following command (change the last parameter to reflect the container_name in your docker-compose.yml):

```
1 docker restart odoo_test_exercise
```

You will also need to keep an eye on the logs regularly, which you can do with the following command (omit the -f if you don't want to output the log continuously):

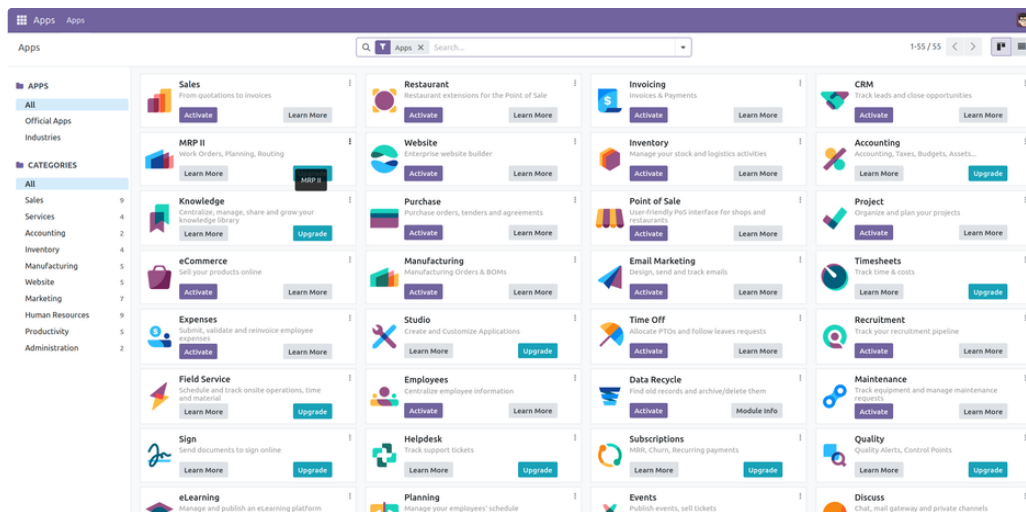
```
1 docker logs -f --tail 300 odoo_test_exercise
```

Creating a database

On your database creation page, fill out the appropriate fields to finish your first Odoo deployment.

- *Master Password* - enter the admin_passwd from you odoo.conf file.
- *Database Name* - enter a database name that matches your dbfilter regex in odoo.conf file. For example if your format was “test_exercise.*”, you could enter either “test_exercise1” or “test_exercise_1” and they will both be usable. Format .* accepts all database names.
- *Email* - enter any username (doesn't have to be an email).
- *Password* - Fill out the password field as you like.
- *Phone Number* - leave empty
- *Language* - we recommend leaving the default “English (US)”.
- *Country* - we recommend using “Estonia” however in the scope of this exercise, you can also leave it empty.
- *Demo Data* - In order to help you test your custom modules more easily later, we also recommend checking the “Demo Data” checkbox.

Once you press the “Create database” and let the backend prepare it for you (you can check the logs for progress). After login you should be greeted with the following page:



For now, your Odoo instance is deployed successfully. We will come back to Odoo itself in Task 2, where we will look at how to install various modules, including your custom one.

Task 1: Search Domains in Polish Notation

The first exercise only requires standard Python and doesn't yet touch Odoo or its framework directly. Use at least Python 3.10+, but do add a comment somewhere which exact Python version you used. In this exercise, **you will not be allowed to use any other Python libraries other than the built-in ones** (i.e. the collections library is completely fine). The only exceptions are libraries that can provide a set of test data, however it would be great if no extra installs are needed to get your code up and running.

As with any other ORM, you will come in contact with a lot of search queries across all the accessible data, that the ORM provides. In Odoo, you will be using Search Domains, which are defined as a list in [Polish Notation](#). This structure lets you define a complex query of AND's, OR's, set operations etc. as an (almost) flat list, rather than a complicated data structure or a pure SQL query.

In order for you to get an understanding of how the notation works, you will be building a parser for it that will take in the search domain and generate a valid SQL query from it (that works in PostgreSQL). You won't be having to deal with JOINS, grouping or ordering the query, but only the “WHERE” clause of the query itself without any one-to-many or many-to-many relations having to be supported. The query does not need to be valid for Odoo in this sense.

 You can use file provided [here](#), to generate test data.

Search Domain Specifications

Search Domain is a list of criteria, each criterion being a triple (either a list or a tuple) consisting of `(field_name, operator, value)`, where:

- `field_name` (str)
 - is a database field (column) name for the table you are querying from
- `operator` (str)
 - is an operator used to compare the value corresponding the `field_name` in the database to the value in the `value` parameter
 - Supported operators in this exercise are as follows:
 - `=`
 - equals to
 - `!=`
 - not equals to
 - `>`
 - greater than
 - `>=`
 - greater than or equal to
 - `<`
 - less than
 - `<=`
 - less than or equal to
- `value` (any type)
 - variable or a static value, that must be comparable to the database column type-wise through the `operator`

The Search Domain criteria can be combined using logical operators in the *prefix* form (polish notation):

- `'&'`
 - logical AND, arity 2 (uses the next 2 criteria or combinations of them)
- `'|'`
 - logical OR, arity 2



These are simplified conditions of [what is supported by Odoo's search domains](#) and don't include, for example, relational fields or set operations.

Examples

To search for news articles, that have a type "science", are not written in English and originate from the Baltic countries, you would use one of the following three domains (all equivalent):

```
1 ['&', ('type', '=', 'science'),
2   '&', ('language_code', '!=', 'en_US'),
3     '|', ('country_code', '=', 'ee'),
4     '|', ('country_code', '=', 'lv'),
5       ('country_code', '=', 'lt')]
```

```
1 ['&', '&',
2   ('type', '=', 'science'),
3   ('language_code', '!=', 'en_US'),
4   '|', '|',
5   ('country_code', '=', 'ee'),
```

```

6 ('country_code', '=', 'lv'),
7 ('country_code', '=', 'lt')]

```

```

1 ['&', '&', '|', '|',
2 ('type', '=', 'science'),
3 ('language_code', '!=', 'en_US'),
4 ('country_code', '=', 'ee'),
5 ('country_code', '=', 'lv'),
6 ('country_code', '=', 'lt')]

```

The domain itself will be interpreted as follows:

```

1 (type IS 'science') AND
2 (language IS NOT english) AND
3 (country IS Estonia OR Latvia OR Lithuania)

```

An example of a query built from any of the three domains should be the same (logic-wise) and could be as follows:

```

1 SELECT * FROM news_article
2 WHERE news_article.type = 'science'
3 AND news_article.language_code != 'en_US'
4 AND (
5     news_article.country_code = 'ee'
6 OR news_article.country_code = 'lv'
7 OR news_article.country_code = 'lt'
8 );

```

Task Completion

When submitting, combine the code of Task 1 into a single Python script. Execution will be done with either Python 3.10 or, if possible, with the version you provided in the comments. Pretty-formatting of the SQL query string is not required, only the logic itself is checked. Please also provide an PostgreSQL-compatible SQL script to generate the test data if you didn't use provided one.

- ✔ Pay attention to the description of the task and try to go through the solution on paper first. Sometimes the description could already have contradictions or errors in it (🤔) - it is the responsibility of the developer to make sure they first understand the underlying problem rather than getting stuck on any specific details within the task's description.

Bonus Points

If you want extra brownie points, you can implement a version of the function, that takes the domain in as a string representation of it (i.e.

```
domain = "[('field', '=', value)]").
```

Task 2: Developing an Odoo module

In this task we will be building a simple extension module to the CRM module already built into Odoo. You can build it inside addons folder created earlier. Custom module structure can be found [here](#).

Before you start

There are multiple resources available to get to know Odoo better:

- 📖 You can get a crash course in Odoo, the UI and all the various modules from [Odoo's 'Getting Started' tutorial](#). The UI there might look slightly different compared to what you have installed, as the tutorials are based on the Enterprise version of the software. All of the

concepts however are the same and the Enterprise edition is just a set of additional modules, extensions and features on top of the Community edition.

You can also check out the rest of the tutorials on [Odoo's eLearning platform](#) if you want to get a head start on getting to know Odoo better. We also recommend you create an account on the eLearning platform to track your progress and get rewards for completing milestones.

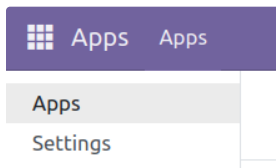
Before moving forward, we highly recommend you go through [Odoo's development tutorial](#) first and follow along with the examples. You can use the environment we have set up or you can duplicate it.

As a heads-up, most of the information in Chapter 2 can be skipped as we have set up the development environment in a slightly different way, but do read through it - there might be some useful information there, such as debugging. Additionally, any other modifications to the configuration, addons, logs or restarting the server can be found in the introduction of this test exercise.

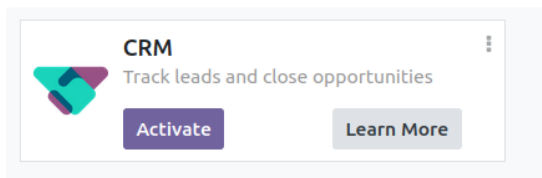
While we suggest going through the entire development tutorial, for the completion of Task 2 you should go through Chapters 1 to 14 at minimum. Without doing so, the completion of Task 2 will be much harder.

Installing CRM in Odoo

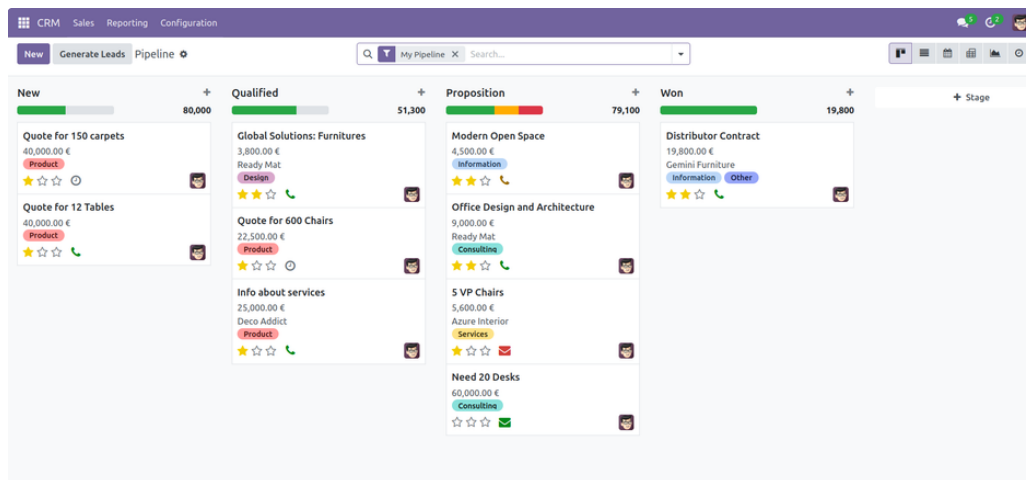
Lets start by installing a built-in Odoo module, which we will be extending in just a bit - the CRM module. Navigate to <http://localhost:8069> (or similar depending on your configuration). If you have navigated somewhere else in Odoo, click on the main menu button in the top left of the page (four squares) and select 'Apps':



Find the CRM module under the Apps page and click on 'Install'.



After the CRM module is installed, under the main menu you should see new items, such as Discuss, Calendar, CRM etc. You can check the other menu items out if you want, but we will be focusing on the following two: CRM and Contacts. If you enabled demo data, then the CRM page should look something like this:



i **Customer Relationship Management or CRM** is used to track all the possible sales opportunities your company might have and to check the progress on them, get reminders on meetings, scheduled calls, gather relevant information about the customer and their interests and, in general, to assist your company's sales team making any *possible* sales into *actual* sales.

Sales possibilities in Odoo are called both *leads* and *opportunities* - the difference of the two usually being the degree of confidence in the customer's interest. *Lead* refers to something we think the customer might be interested in and an *opportunity* being where the customer has already expressed or confirmed that interest, but no sales orders or contracts have yet been established.

The usual workflow goes from Lead → Opportunity → Sales Order. In the default configuration, only one level is used for both a *lead* and an *opportunity* which we will not be changing.

i The base code for the CRM module can be found here: <https://github.com/odoo/odoo/tree/17.0/addons/crm>

Feel free to check out the rest of the leads, create some of your own and otherwise poke around in the CRM module. Central to the CRM module is also the Contacts page under the main menu.

The Task

Lets spice the things up with a fictional scenario.

We have a Client who wishes to minimize the number of clicks they have to do to perform every-day tasks in the CRM module. Unique in the way compared to how most companies operate with their sales leads, they have an “ethos” that every person in their system should have a lead opened about them, as they consider everyone to be a potential customer (yes even their employees for some reason).

As such, they have requested that we add a new page to the CRM lead's form view which would help them navigate more efficiently. The idea behind that page is to find every person in the system (res.partner aka Contact/Company object, from here on out just “partners”), that does not yet have a CRM lead associated with them and offer them a way to open the next lead straight from that page, removing a few extra clicks from the process.

In more technical terms, we have outlined a few requirements for the task that must be completed and some optional ones, which you can also complete that will make the solution to the problem a bit more complete. Down below, the requirements in **bold** are required and the ones in *italic* are optional.

- On the CRM lead form view, add a new tab called “Next Leads” (down below where you have “Internal Notes” and “Extra Information”)
- Within that tab, add a list of partners, that do not have a CRM lead associated with them (hint: use one2many, compute and readonly)
- For each item/row on that list, add a button to the list view that would create a new lead, open it and already have the “Customer” field filled out with the partner, where the button was clicked

- *Limit the list to partners, that are not system users (aka cannot log in to backend) - you can check out various user rights under res.users or other flags on the partner, user objects*
- *Add a date filter to also include partners, that haven't had a new lead created in the past 28 days*
- *Color the lines differently if the partner has not had a lead ever vs hasn't had a lead only in the past 28 days (hint: use decoration-X, i.e. decoration-info in the XML)*
- *Add whatever ideas you might have to make this new feature even more thorough - completely up to you what else you would like to add there - list those features when submitting the task*

Task Completion

Submit the entire module of Task 2 and whatever else you created during the Odoo tutorial. Outline any additional changes you made, what features you added yourself etc.

- ✓ If you wish, you can skip the task requirements in *italic* above, however we recommend at least trying to find a solution for them - even if you are not able to complete them, you can explain what you tried and what problems you ran into, what turned out to be difficult etc.

Bonus Points

For bonus points, you can answer these questions below in your own thoughts about the problem and solution proposed to the customer. Try to reason about the proposed solution and other possible alternatives to achieve the same end result.

- Given the custom nature of this request, why do you think a development such as this should or should not be recommended to the customer?
- Are there better alternatives to this which would solve the underlying problem, but not in the exact way the customer wishes?
- Would it take less time or more time to develop the alternative and how would the extra investment in time/money/resources be more beneficial to the customer?
- What are the various pitfalls of such a development that should be taken into account and communicated to the customer before continuing with this specific development? In more simpler terms: why do you think this solution to the customer's problem is good or bad?

Conclusion

Don't worry if you are not able to complete everything. We want to see your effort and logical thinking above all. Zip your python script (and SQL script if you didn't use provided one) from task 1 and custom module from task 2 together and send it to us. Also feel free to add text file with all you comments. Thank you!