Collibra

# Django coding challenge

## The definitions

Let us have the following Django model definitions that represent relational database structures:

```
from django.db import models


class NodeTypes(models.Model):
    node_type_name = models.CharField(max_length=100)


class Nodes(models.Model):
    node_type = models.ForeignKey(NodeTypes, on_delete=models.CASCADE, related_name='nodes')
    parent = models.ForeignKey('self', on_delete=models.CASCADE, related_name='children', null=True)
    node_name = models.CharField(max_length=100)


class Tables(models.Model):
    node = models.ForeignKey(Nodes, on_delete=models.CASCADE, related_name='tables')
    table_name = models.CharField(max_length=100)
    description = models.TextField()


class Columns(models.Model):
    table = models.ForeignKey(Tables, on_delete=models.CASCADE, related_name='columns')
    column_name = models.CharField(max_length=100)
    description = models.TextField()
```

Where:

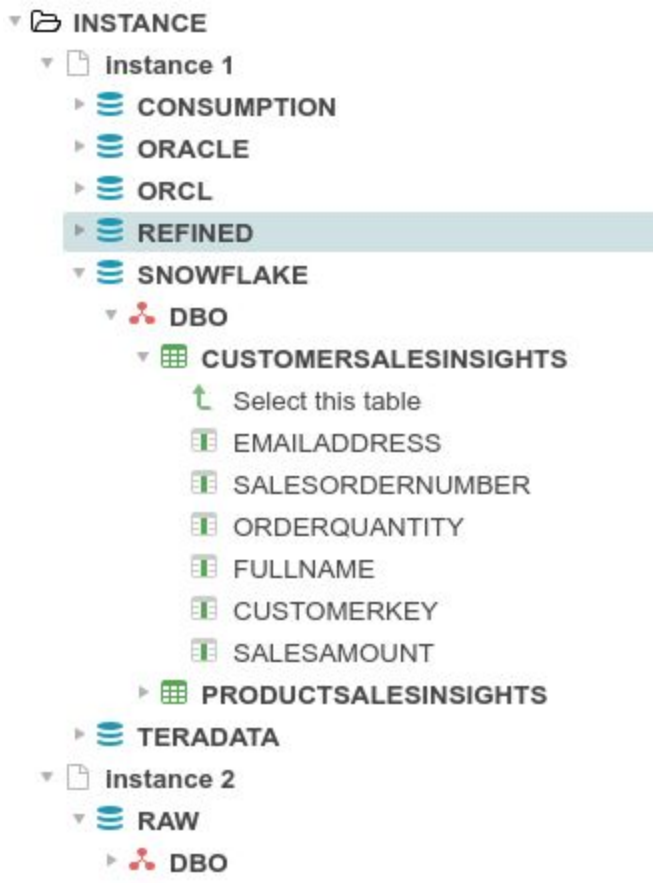
- `NodeTypes` are types of components of a relational database system, such as instance, database, schema;
- `Nodes` are parts of a tree hierarchy that store such components of a relational database system;
- `Tables` are relational database tables;
- `Columns` are relational database columns.

For example, suppose that we have a database and a schema named "foo" and "bar" respectively with the following data definition:

```
CREATE TABLE IF NOT EXISTS "baz" (
    "quux" integer,
    "quuz" integer
);
```

The Django representation would create two `NodeTypes` model instances named "database" and "schema" together with two `Nodes` model instances named "foo" and "bar". The former would have the `node_type` attribute set to the "database" `NodeTypes` model instance and the `parent` attribute to `None`. The latter would have the `node_type` attribute set to the "schema" `NodeTypes` model instance and the `parent` attribute to the "database" `Nodes` model instance. Furthermore, it would create a `Tables` model instance named "baz" pointing to the "schema" `Nodes` model instance. Finally, it would create two Columns model instances named "quux" and "quuz", that would point to the "baz" `Tables` model instance.

# The assignment




1. Create a new Django project that would contain the above-mentioned model definitions. You can expand the models in any way you like, but do not change their relations.
2. In the Django project, implement a tree browser that would show the `NodeTypes` → `Nodes` → `Tables` → `Columns` hierarchy and navigate through it. On the right side, there is an example of such a browser. However, your implementation does not need to look the same.
3. The tree browser should have the closing/opening functionality by means of clicking on its branches. You are allowed to use a JavaScript library of your choice to provide this functionality. The default state of the browser is to have all the branches closed.
4. After clicking on the table/column item in the tree browser, the user should be redirected to a new page where there is shown the name of the selected item together with its description. If possible, include the tree browser that would contain the selected item in an opened state as well.
5. Optionally, you can write a test suite, create a Dockerfile, document your project, put it into a git repository, style it properly etc. Do the same as you would do in case of working on a live project.
6. Pack your Django project directory into an archive file and send it to us via email.