Data Analysis Dashboard using Python-Django and Flexmonster

This project is developed in Visual Code.

- **Reference:** https://morioh.com/p/88d6fc714f52

- **Bootstrap the django project:**

```
django-admin startproject [project_name]
cd [project_name]
python manage.py runserver
```

- **Create New app in project:**

```
python manage.py startapp [app_name]
```

- **Inside of [app_name]:**

__init__.py to make Python treat it as a package

admin.py - settings for the Django admin pages

apps.py - settings for app's configs

models.py - classes that will be converted to database tables by the Django's ORM

tests.py - test classes

views.py - functions & classes that define how the data is displayed in the templates

- **It is necessary to register the app in the project:**

Go to data_analysis/settings.py and append the app's name to the INSTALLED_APPS list

- **Views:** function to call and view the template
- **Templates:** html template files
- **Mapping views functions to URLs:** app URLs to project URLs
- **Model (data modelling):** a data model is a conceptual representation of the data stored in a database
- **Working with a database:**

Migration is simply a file that describes which changes must be applied to the database. Every time we need to create a database based on the model described by Python classes; we use migration.

 # create migration

```
python manage.py makemigrations analysis_dashboard
```

 # apply migrations and create database

```
python manage.py migrate analysis_dashboard
```

Data Analysis Dashboard using Python-Django and Flexmonster

 # Access the database from Django shell

```
python manage.py shell

>>  from analysis_dashboard.models import Order
    o1 = Order(product_category='Books',
    payment_method='Credit card',
    shipping_cost=39,
    unit_price=59
    )
    o1.save()
    o2 = Order(product_category='Magazines',
    payment_method='Credit card',
    shipping_cost=12,
    unit_price=50
    )
    o2.save()
```

- **Connecting data to flexomonster:**

Using an async request (AJAX) that returns the data in JSON. Within the ajax call, we make a request based on the URL contained in the data-URL property. Then we tell the ajax request that we expect a JSON object to be returned (defined by dataType).

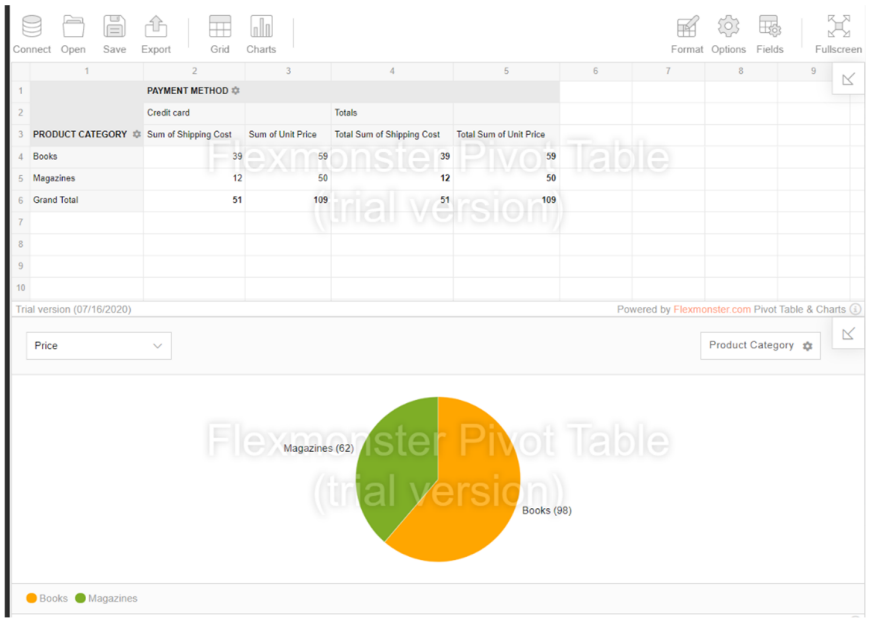
- **Customization of fields:**

Flexmonster provides a special property of the data source that allows setting field data types, custom captions, and defining multi-level hierarchies.

- **Dashboard Design and Output:**

python manage.py runserver

- **Output:**



Data Analysis Dashboard using Python-Django and Flexmonster