OS Lab 10 – Analytics Dashboard

IULIANA MARIN

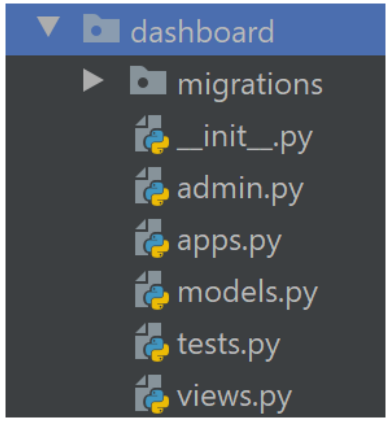
Getting started

- Create a new project called analytics_project inside the terminal of Pycharm:

```
django-admin startproject analytics_project
```

- Check that everything is alright by running the command: python manage.py runserver
- Create a new application called dashboard in your project:

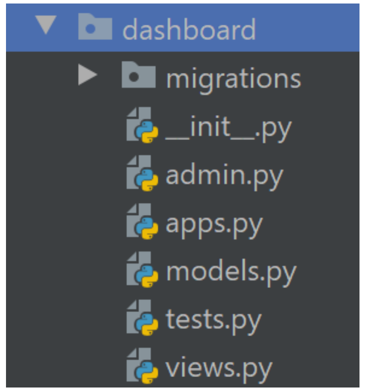
```
python manage.py startapp dashboard
```



```
dashboard
    migrations
    __init__.py
    admin.py
    apps.py
    models.py
    tests.py
    views.py
```

Information about the files

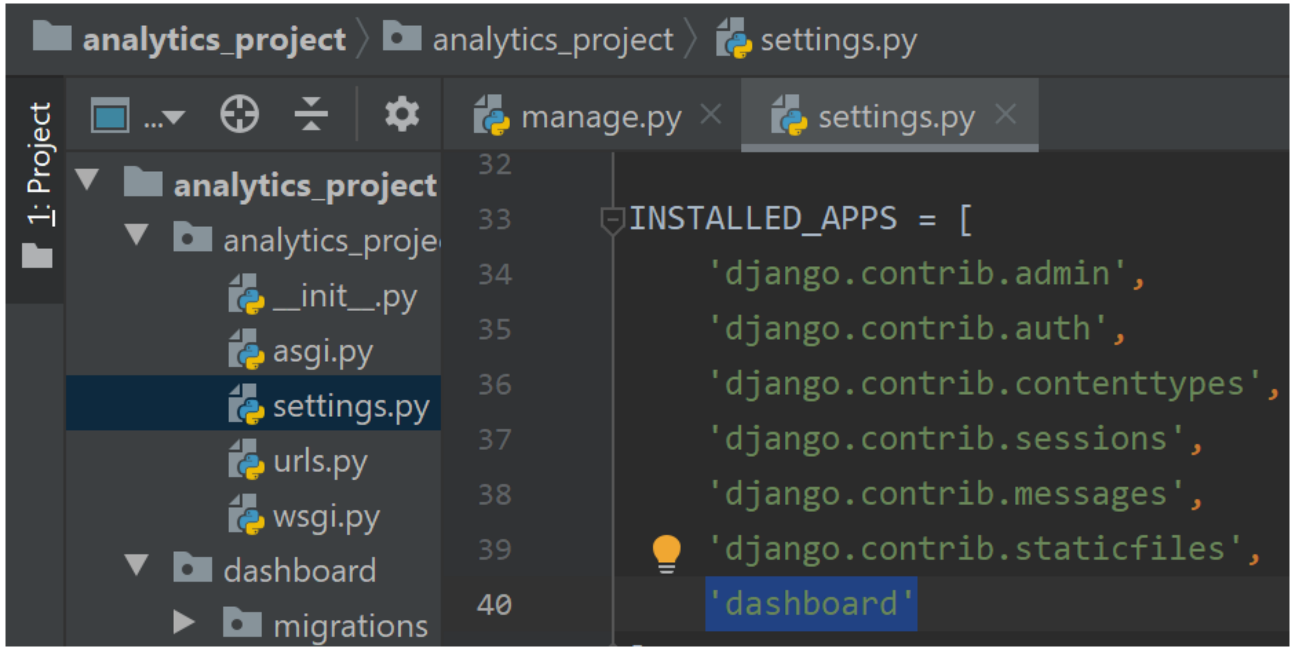
- Your project contains several files:
  - __init__.py : Python treats it as a package
  - admin.py : settings for the Django admin pages
  - apps.py : settings for app's configs
  - models.py: classes that will be converted to database tables by the Django's object-relational mapping (ORM)
  - tests.py : test classes
  - views.py: functions and classes that define how the data is displayed in templates



```
▼ dashboard
  ▶ migrations
    __init__.py
    admin.py
    apps.py
    models.py
    tests.py
    views.py
```

Registration of the dashboard application

- It is necessary to register the app in the project.
- Go to analytics_project/settings.py and append the app's name to the INSTALLED_APPS list:



```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'dashboard'
```

View

- Update the dashboard/views.py file and add a function which directs a user to the specific templates that are defined in dashboard/templates folder, as well as a method which retrieves the data of the pivot table.

```python
from django.http import JsonResponse
from django.shortcuts import render
from dashboard.models import Order
from django.core import serializers

def dashboard_with_pivot(request):
    return render(request, 'dashboard_with_pivot.html', {})

def pivot_data(request):
    dataset = Order.objects.all()
    data = serializers.serialize('json', dataset)
    return JsonResponse(data, safe=False)
```

Templates

- Create a directory named **templates** inside the **dashboard** project. Inside it create a new file called **dashboard_with_pivot.html**.

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Dashboard with Flexmonster</title>
  <script src="https://cdn.flexmonster.com/flexmonster.js"></script>
  <script src="https://code.jquery.com/jquery-3.3.1.min.js"></script>
  <link rel="stylesheet" href="https://cdn.flexmonster.com/demo.css">
</head>
<body>
<div id="pivot-table-container" data-url="{% url 'pivot_data' %}"></div>
<div id="pivot-chart-container"></div>
</body>
</html>
```

Mapping view functions to URLs

- Views are called and rendered to the user through the use of views that are mapped to corresponding URLs. Edit the **analytics_project/urls.py** file:

```
from django.contrib  import  admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('dashboard/', include('dashboard.urls')),
]
```

Mapping view functions to URLs

- Create the file **urls.py** inside the **dashboard** folder and add the list of the URL patterns which are mapped to the view functions.

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.dashboard_with_pivot, name='dashboard_with_pivot'),
    path('data', views.pivot_data, name='pivot_data'),
]
```

Creation of Models

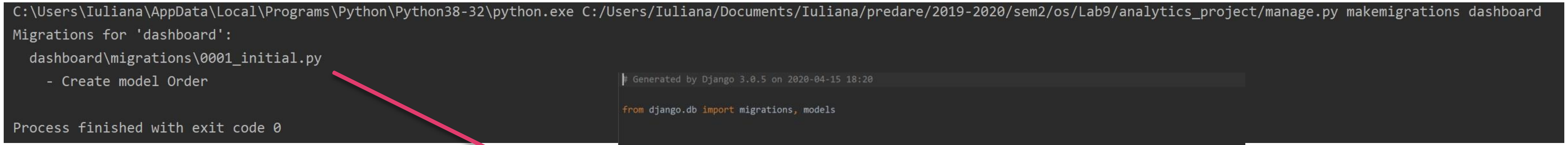
- A data model is a conceptual representation of the data which is stored in a database. The DB is SQLite which comes together with the Django web development server.
- Considering that we create a dashboard for the sales department, an **Order** class is created inside **dashboard/models.py**:

```python
from django.db import models

class Order(models.Model):
    product_category = models.CharField(max_length=20)
    payment_method = models.CharField(max_length=50)
    shipping_cost = models.CharField(max_length=50)
    unit_price = models.DecimalField(max_digits=5, decimal_places=2)
```

Translation of the model class into a database table

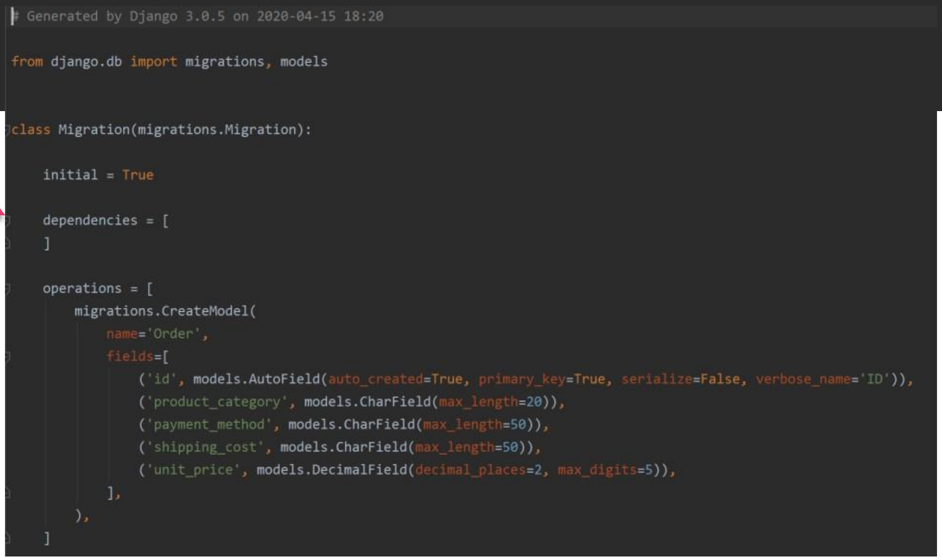
- Migration is responsible for the changes which are applied to the database. This is done using the command:

python manage.py makemigrations dashboard



```
C:\Users\Iuliana\AppData\Local\Programs\Python\Python38-32\python.exe C:/Users/Iuliana/Documents/Iuliana/predare/2019-2020/sem2/os/Lab9/analytics_project/manage.py makemigrations dashboard
Migrations for 'dashboard':
  dashboard\migrations\0001_initial.py
    - Create model Order

Process finished with exit code 0
```

Determines the creation of the following file

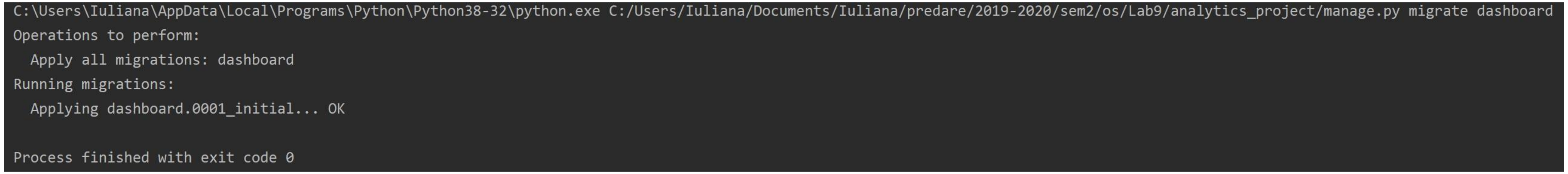


```
# Generated by Django 3.0.5 on 2020-04-15 18:20

from django.db import migrations, models


class Migration(migrations.Migration):

    initial = True

    dependencies = [
    ]

    operations = [
        migrations.CreateModel(
            name='Order',
            fields=[
                ('id', models.AutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
                ('product_category', models.CharField(max_length=20)),
                ('payment_method', models.CharField(max_length=50)),
                ('shipping_cost', models.CharField(max_length=50)),
                ('unit_price', models.DecimalField(decimal_places=2, max_digits=5)),
            ],
        ),
    ]
```

Translation of the model class into a database table

- Based on the created migration file, apply the changes and the database file (db.sqlite3) will be updated:

python manage.py migrate dashboard



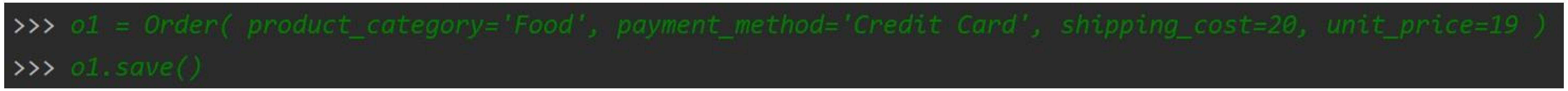
```
C:\Users\Iuliana\AppData\Local\Programs\Python\Python38-32\python.exe C:/Users/Iuliana/Documents/Iuliana/predare/2019-2020/sem2/os/Lab9/analytics_project/manage.py migrate dashboard
Operations to perform:
  Apply all migrations: dashboard
Running migrations:
  Applying dashboard.0001_initial... OK

Process finished with exit code 0
```

As well as: python manage.py migrate

Creation of database entries

- Database entries are created using the Django shell.
- Start the Django shell using the command: python manage.py shell
- Write the following code in the interactive console:

```
from dashboard.models import Order
o1 = Order( product_category='Food', payment_method='Credit Card', shipping_cost=20, unit_price=19)
o1.save()
o2 = Order( product_category='Cleaning products', payment_method= 'Cash', shipping_cost=20,
unit_price =29 )
o2.save()
o3 = Order( product_category='Movies', payment_method= 'Cash', shipping_cost=20, unit_price=25 )
o3.save()
```



```
>>> o1 = Order( product_category='Food', payment_method='Credit Card', shipping_cost=20, unit_price=19 )
>>> o1.save()
```

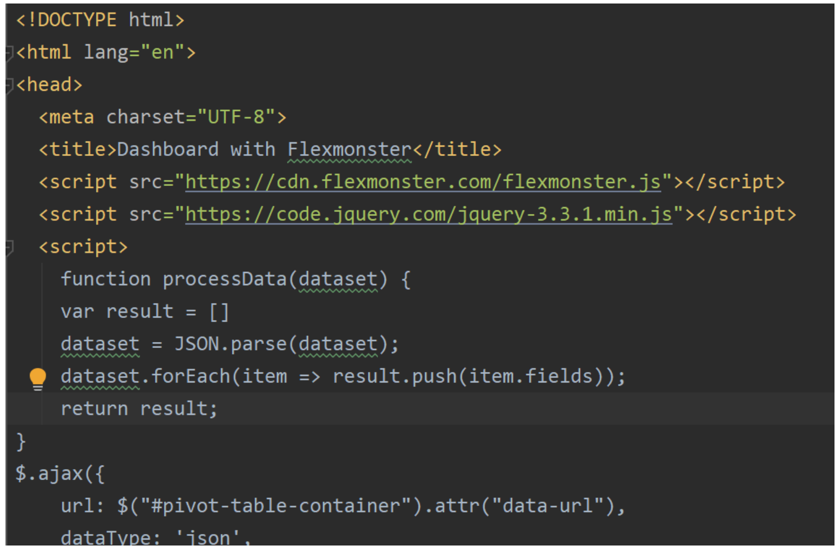
Connection to Flexmonster

- Data needs to be passed from the model to a data visualization tool on the front end. Flexmonster is used for visualizing data.
- The back end and Flexmonster communicate based on a request-response cycle using Python and the Django template engine to write JavaScript code in the template, as well as by using async request, AJAX, which return data as JSON. The second approach will be used in this lab.
- Flexmonster can interpret JSON.
- Edit the **templates/dashboard_pivot.html** file. The two div contains will render the pivot grid and pivot charts.
- In the AJAX call, the request is based on the URL contained in the data-URL property.
- The server sets the JSON response to be the data parameter.

Connection to Flexmonster

- Edit the **templates/dashboard_pivot.html** file. Add the following lines of code inside the script tag.

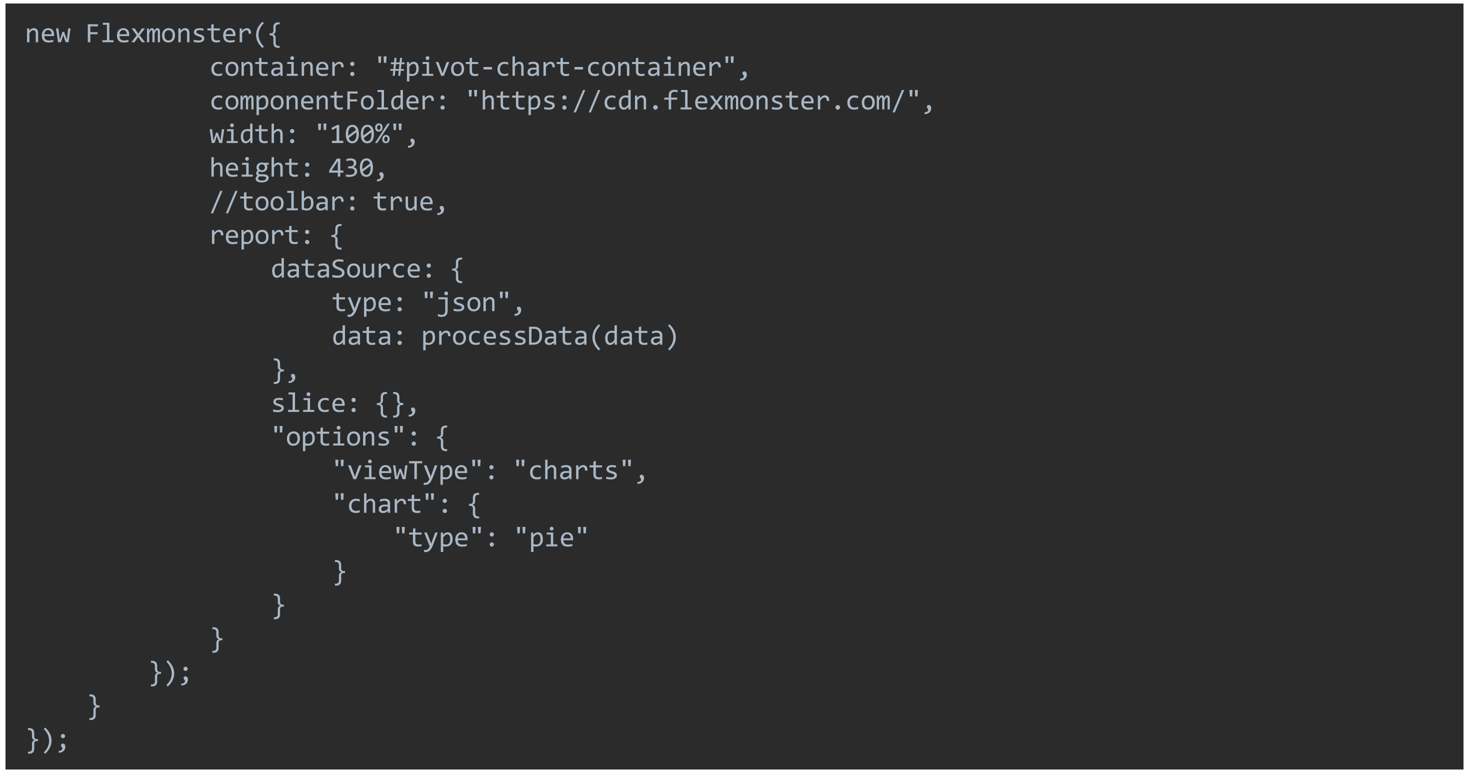
```
function processData(dataset) {
    var result = []
    dataset  = JSON.parse(dataset);
    dataset.forEach(item => result.push(item.fields));
    return result;
}
$.ajax({
    url: $("#pivot-table-container").attr("data-url"),
    dataType: 'json',
    success: function(data) {
        new Flexmonster({
            container: "#pivot-table-container",
            componentFolder: "https://cdn.flexmonster.com/",
            width: "100%",
            height: 430,
            toolbar: true,
            report: {
                dataSource: {
                    type: "json",
                    data: processData(data)
                },
                slice: {}
            }
        });
```



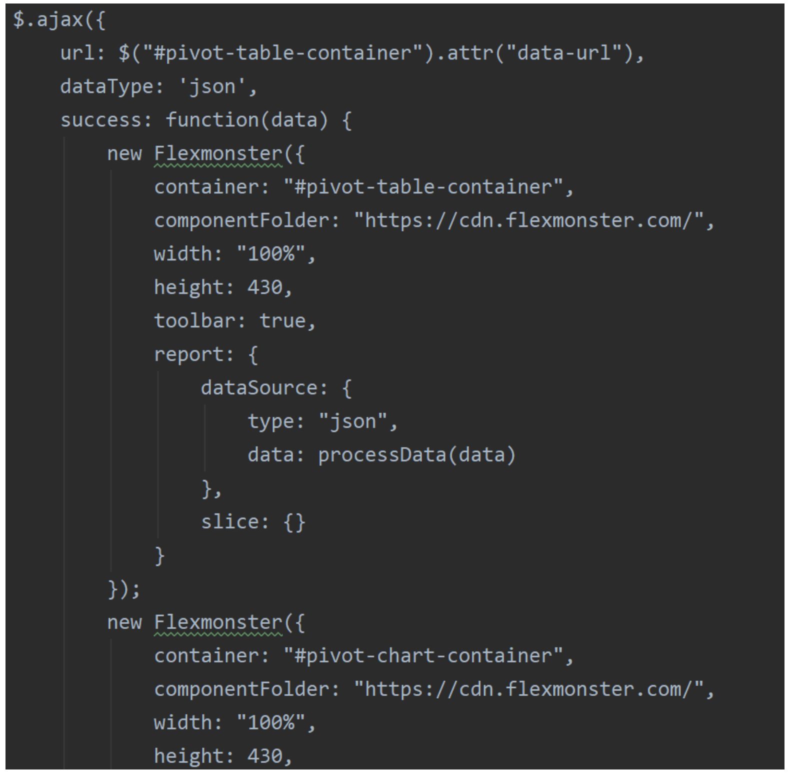
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Dashboard with Flexmonster</title>
  <script src="https://cdn.flexmonster.com/flexmonster.js"></script>
  <script src="https://code.jquery.com/jquery-3.3.1.min.js"></script>
  <script>
    function processData(dataset) {
    var result = []
    dataset = JSON.parse(dataset);
    dataset.forEach(item => result.push(item.fields));
    return result;
}
$.ajax({
    url: $("#pivot-table-container").attr("data-url"),
    dataType: 'json',
```

Connection to Flexmonster

- Edit the **templates/dashboard_pivot.html** file. Add the following lines of code inside the script tag.



```
new Flexmonster({
            container: "#pivot-chart-container",
            componentFolder: "https://cdn.flexmonster.com/",
            width: "100%",
            height: 430,
            //toolbar: true,
            report: {
                dataSource: {
                    type: "json",
                    data: processData(data)
                },
                slice: {},
                "options": {
                    "viewType": "charts",
                    "chart": {
                        "type": "pie"
                    }
                }
            }
        });
    }
});
```



```
$.ajax({
    url: $("#pivot-table-container").attr("data-url"),
    dataType: 'json',
    success: function(data) {
        new Flexmonster({
            container: "#pivot-table-container",
            componentFolder: "https://cdn.flexmonster.com/",
            width: "100%",
            height: 430,
            toolbar: true,
            report: {
                dataSource: {
                    type: "json",
                    data: processData(data)
                },
                slice: {}
            }
        });
        new Flexmonster({
            container: "#pivot-chart-container",
            componentFolder: "https://cdn.flexmonster.com/",
            width: "100%",
            height: 430,
```

Customizing fields

- Flexmonster datasource allows to set data types, custom captions and multi-level hierarchies. Edit the **templates/dashboard_pivot.html** file. Add the following lines of code for the dataSource property.

```
mapping: {
    "product_category": {
        "caption": "Product Category",
        "type": "string"
    },
    "payment_method": {
        "caption": "Payment Method",
        "type": "string"
    },
    "shipping_cost": {
        "caption": "Shipping Cost",
        "type": "number"
    },
    "unit_price": {
        "caption": "Unit Price",
        "type": "number"
    }
}
```

Dashboard design

- The final content of the **templates/dashboard_pivot.html** file, along with styling is:

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Dashboard with Flexmonster</title>
    <script src="https://cdn.flexmonster.com/flexmonster.js"></script>
    <script src="https://code.jquery.com/jquery-3.3.1.min.js"></script>
    <link rel="stylesheet" href="https://cdn.flexmonster.com/demo.css">
    <style>
/* Charts Style  */

.fm-charts-color-1 {
  fill: #ffa600 !important;
}

.fm-charts-color-2 {
  fill: #7eae26 !important;
}

.fm-charts-color-3 {
  fill: #00a45a !important;
}
```

Dashboard design

- The final content of the **templates/dashboard_pivot.html** file, along with styling is:

```
.fm-charts-color-4 {
  fill: #df3800 !important;
}

.fm-charts-color-5 {
  fill: #e95800 !important;
}

.fm-charts-color-6 {
  fill: #ffa600 !important;
}</style>
</head>
<body>
<div id="pivot-table-container" data-url="{% url 'pivot_data' %}"></div>
<div id="pivot-chart-container"></div>
<script>
function processData(dataset) {
    var result = []
    dataset = JSON.parse(dataset);
    dataset.forEach(item => result.push(item.fields));
    return result;
}
```

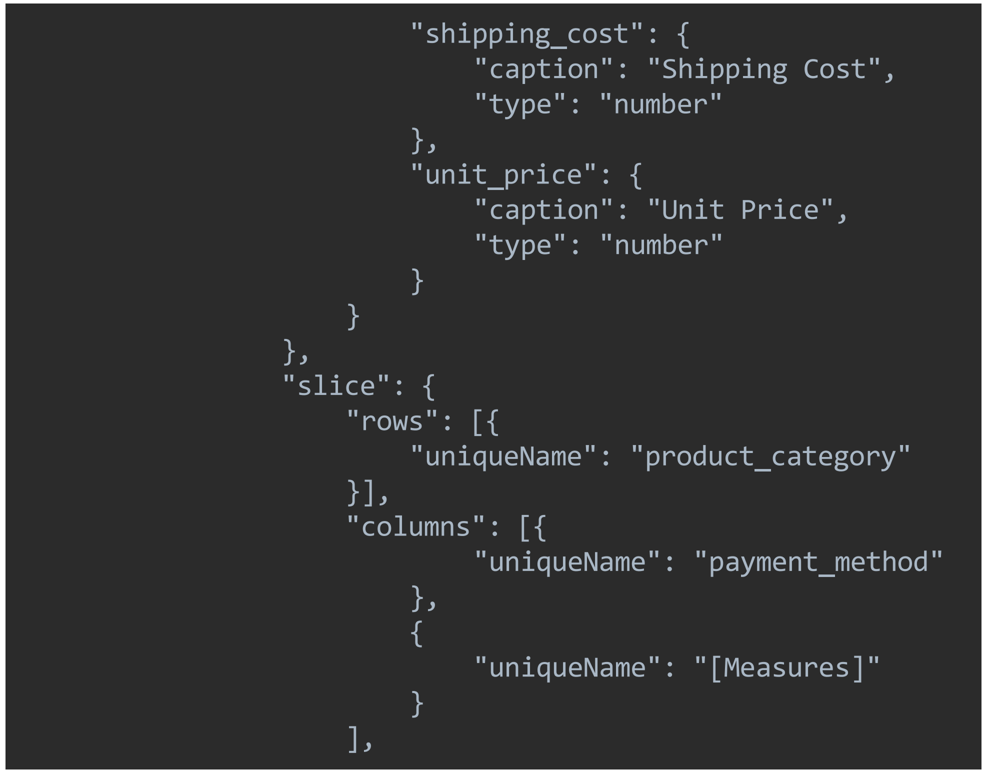
Dashboard design

- The final content of the **templates/dashboard_pivot.html** file, along with styling is:

```
$.ajax({
    url: $("#pivot-table-container").attr("data-url"),
    dataType: 'json',
    success: function(data) {
        new Flexmonster({
            container: "#pivot-table-container",
            componentFolder: "https://cdn.flexmonster.com/",
            width: "100%",
            height: 430,
            toolbar: true,
            report: {
                dataSource: {
                    type: "json",
                    data: processData(data),
                    mapping: {
                        "product_category": {
                            "caption": "Product Category"
                        },
                        "payment_method": {
                            "caption": "Payment Method"
                        },
```

Dashboard design

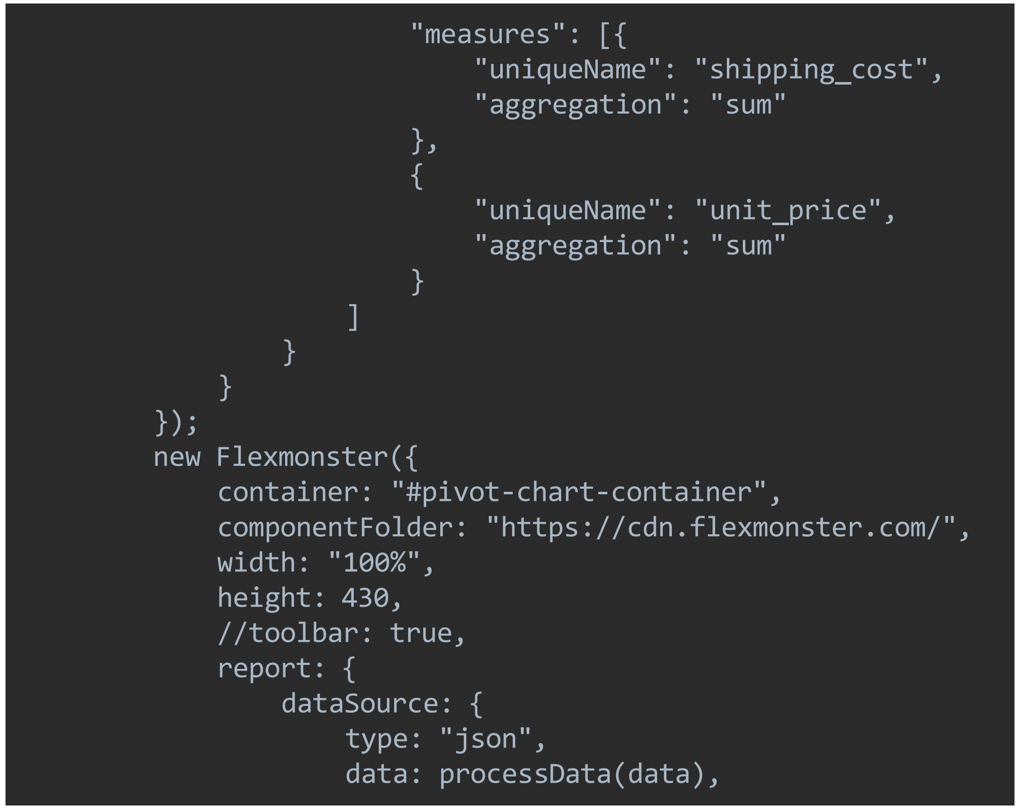
- The final content of the **templates/dashboard_pivot.html** file, along with styling is:



```
                        "shipping_cost": {
                            "caption": "Shipping Cost",
                            "type": "number"
                        },
                        "unit_price": {
                            "caption": "Unit Price",
                            "type": "number"
                        }
                    }
                },
                "slice": {
                    "rows": [{
                        "uniqueName": "product_category"
                    }],
                    "columns": [{
                            "uniqueName": "payment_method"
                        },
                        {
                            "uniqueName": "[Measures]"
                        }
                    ],
```

Dashboard design

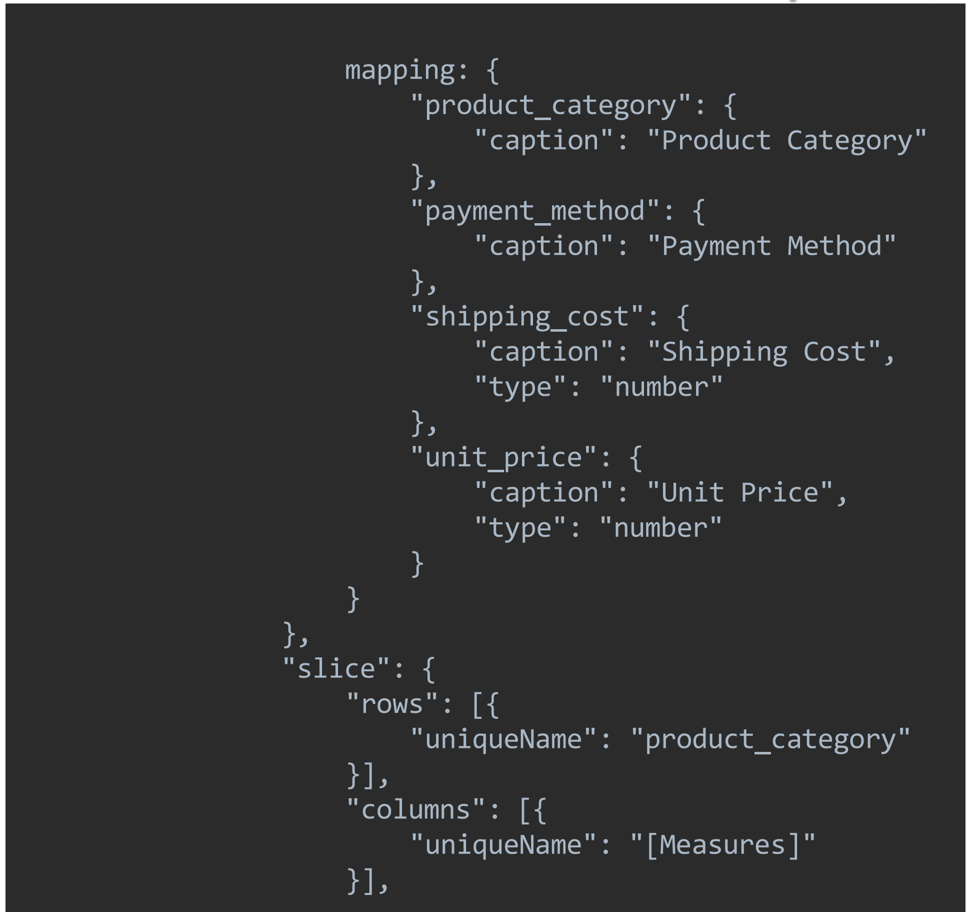
- The final content of the **templates/dashboard_pivot.html** file, along with styling is:



```
                    "measures": [{
                        "uniqueName": "shipping_cost",
                        "aggregation": "sum"
                    },
                    {
                        "uniqueName": "unit_price",
                        "aggregation": "sum"
                    }
                ]
            }
        }
    });
    new Flexmonster({
        container: "#pivot-chart-container",
        componentFolder: "https://cdn.flexmonster.com/",
        width: "100%",
        height: 430,
        //toolbar: true,
        report: {
            dataSource: {
                type: "json",
                data: processData(data),
```

Dashboard design

- The final content of the **templates/dashboard_pivot.html** file, along with styling is:



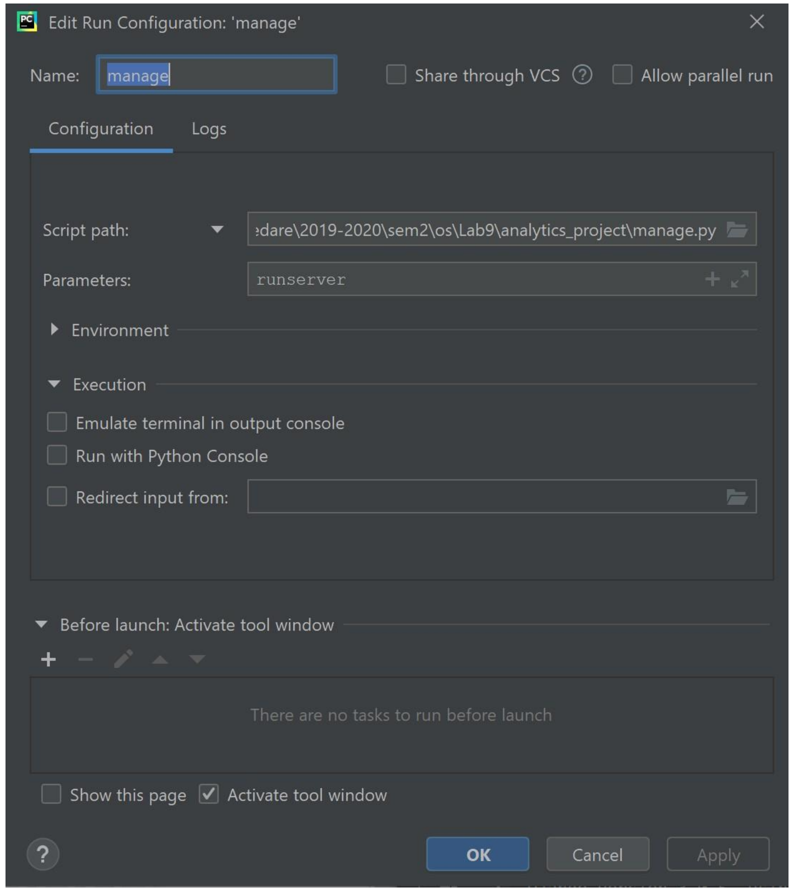
```
                    mapping: {
                        "product_category": {
                            "caption": "Product Category"
                        },
                        "payment_method": {
                            "caption": "Payment Method"
                        },
                        "shipping_cost": {
                            "caption": "Shipping Cost",
                            "type": "number"
                        },
                        "unit_price": {
                            "caption": "Unit Price",
                            "type": "number"
                        }
                    }
                },
                "slice": {
                    "rows": [{
                        "uniqueName": "product_category"
                    }],
                    "columns": [{
                        "uniqueName": "[Measures]"
                    }],
```

Dashboard design

- The final content of the **templates/dashboard_pivot.html** file, along with styling is:

```
                    "measures": [{
                        "uniqueName": "Price",
                        "formula": "sum(\"shipping_cost\") + sum(\"unit_price\")",
                        "caption": "Price"
                    }]
                },
                "options": {
                    "viewType": "charts",
                    "chart": {
                        "type": "pie"
                    }
                }
            }
        });
    }
});

</script>
</body>
</html>
```

Test the application


Edit Run Configuration: 'manage'

Name: manage

[ ] Share through VCS

[ ] Allow parallel run

Script path: :dare\2019-2020\sem2\os\Lab9\analytics_project\manage.py

Parameters: runserver

[ ] Emulate terminal in output console

[ ] Run with Python Console

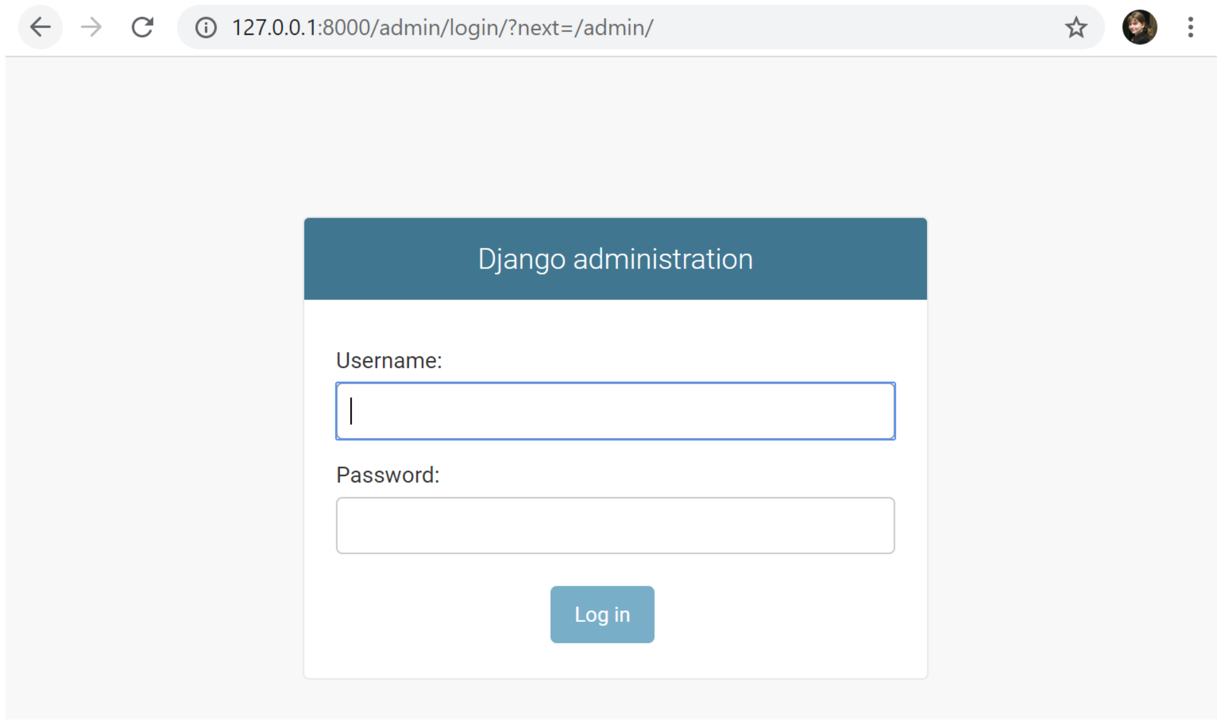
[ ] Redirect input from:

Before launch: Activate tool window

There are no tasks to run before launch

[ ] Show this page

[✓] Activate tool window

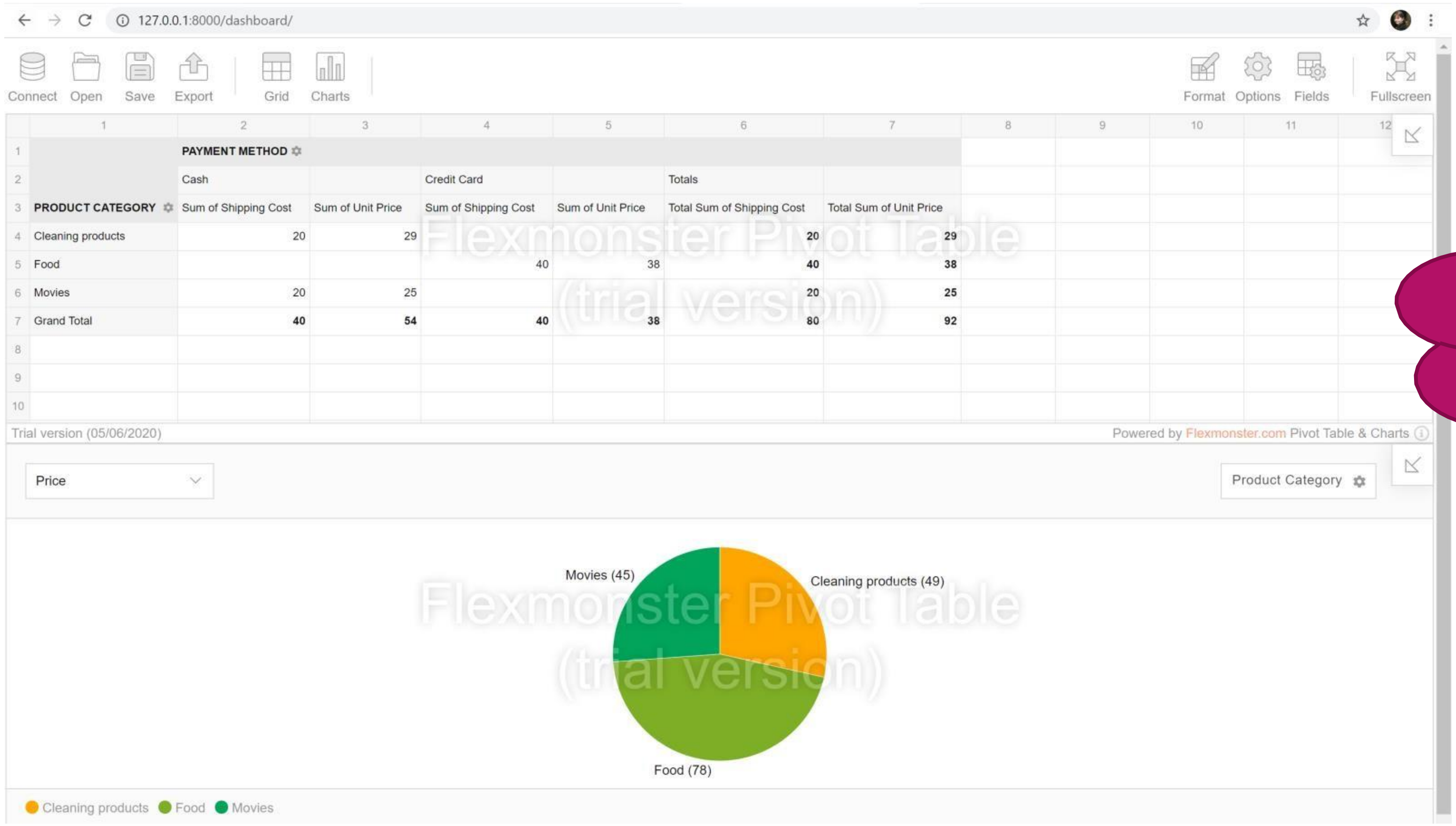

127.0.0.1:8000/admin/login/?next=/admin/

Django administration

Username:

Password:

Test the application

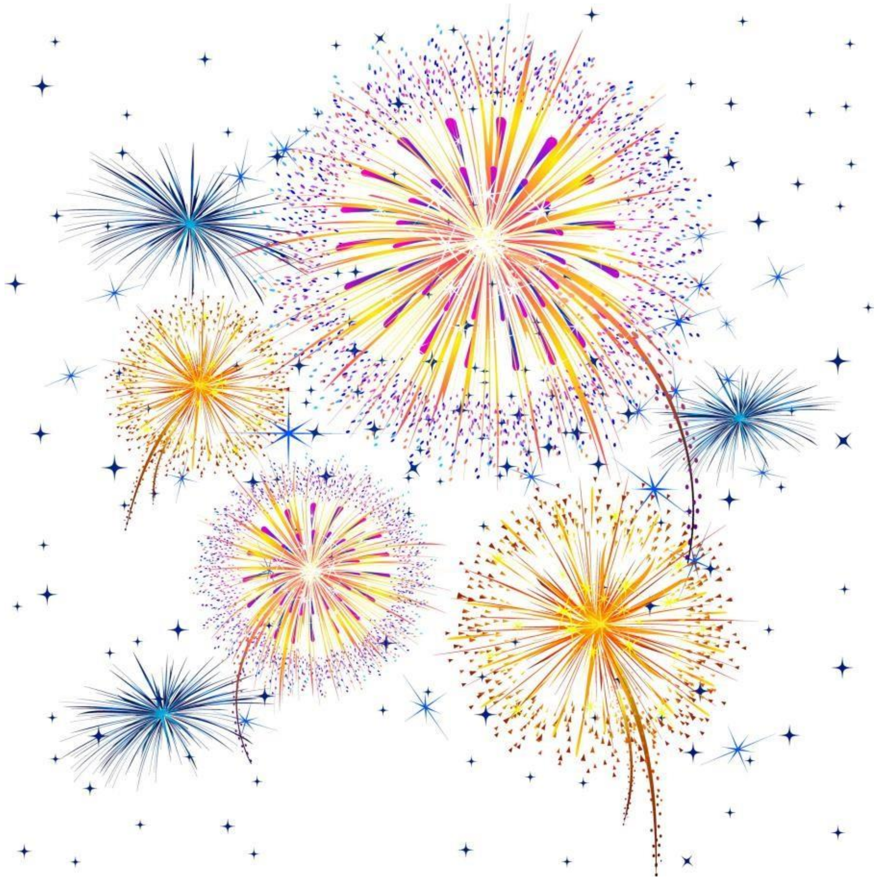


	PAYMENT METHOD					
	Cash		Credit Card		Totals	
PRODUCT CATEGORY	Sum of Shipping Cost	Sum of Unit Price	Sum of Shipping Cost	Sum of Unit Price	Total Sum of Shipping Cost	Total Sum of Unit Price
Cleaning products	20	29			**20**	**29**
Food			40	38	**40**	**38**
Movies	20	25			**20**	**25**
Grand Total	**40**	**54**	**40**	**38**	**80**	**92**

Trial version (05/06/2020)

Try the options which you have available. Psst! … you can even export data as pdf, change the chart type.

Congratulations!
You have finished your analytics dashboard app!



You can now start to work on your project!