

Exercice Simplon :

1.IMPORT DE LA BASE DE DONNÉES VENTES SUR SQLITEONLINE

But : charger les données CSV dans une table SQL nommée `ventes` pour pouvoir interroger avec du SQL.

2.VÉRIFIER QUE LA BASE A ÉTÉ IMPORTER

```
1 SELECT *
2 FROM ventes;
3
4
```

date	produit	prix	qte	region
2022-01-01	Produit A	10	100	Nord
2022-01-02	Produit B	15	50	Nord
2022-01-02	Produit A	10	75	Sud
2022-01-03	Produit C	20	30	Nord
2022-01-03	Produit A	10	150	Sud
2022-01-04	Produit B	15	75	Nord
2022-01-04	Produit C	20	50	Sud
2022-01-05	Produit A	10	125	Nord

3. CRÉEZ LES REQUÊTES SQL POUR RÉPONDRE AUX QUESTIONS CLÉS SUR LES VENTES DE L'ENTREPRISE :

a. le chiffre d'affaires total :

```
Run SQLite
1 SELECT SUM(prix * qte) AS chiffre_affaires_total
2 FROM ventes;
3
4
```

chiffre_affaires_total
44825

Résultat : 44825

CA par produit :

```
1 SELECT produit,SUM(qte*prix) AS total_CA
2 FROM ventes
3 GROUP BY produit;
```

produit	total_CA
Produit A	17500
Produit B	15825
Produit C	11500

b. les ventes par produit :

```
1 SELECT produit,SUM(qte) AS total_ventes
2 FROM ventes
3 GROUP BY produit;
```

produit	total_ventes
Produit A	1750
Produit B	1055
Produit C	575

c. les ventes par région.

```
1 SELECT region,SUM(qte) AS ventes_region
2 FROM ventes
3 GROUP BY region;
```

region	ventes_region
Nord	1605
Sud	1775

4. RENDEZ-VOUS SUR GLITCH, UN LOGICIEL EN LIGNE PERMETTANT DE PROGRAMMER DES SCRIPTS ET DES APPLICATIONS WEB, EN DUPLIQUANT (“REMIXER”) LA BASE DE PROJET PYTHON.

5. LISEZ LES INSTRUCTIONS DU FICHIER “README.MD”.

The screenshot displays the Glitch IDE interface for a project named "stormy-abalone-sprite". On the left sidebar, there are sections for "Settings", "Assets", and "Files". The "Files" section lists "README.md", "app.py", "ventes-par-region.html", and "ventes.csv". The main area shows the "README.md" file with the following content:

Qu'est-ce que c'est ?

Ceci est un projet de visualisation de données, qui utilise le langage de programmation Python. Il utilise deux outils : [pandas](#) et [plotly](#).

- Pandas va nous permettre de télécharger un fichier de données CSV depuis une URL.
- Plotly va nous permettre de générer des graphiques puis de les exporter en page web (au format HTML).

Comment ça marche ?

← README.md : Il s'agit de ce fichier, que vous lisez en ce moment même.

← app.py : ceci est un fichier python, le cœur du projet.

Pour exécuter le fichier Python et ainsi générer un graphique sous forme de page web, cliquez sur le bouton "TERMINAL" depuis la barre d'outils en bas de page.

Puis écrivez la commande suivante : `python3 app.py`.

Cette commande se divise en deux parties :

Below the README, there is a "Terminal" section with a "Full Page Terminal" button. The terminal output is as follows:

```
For more information about this and other technical restrictions,
please see the Help Center: https://help.glitch.com/

app@stormy-abalone-sprite:~ 07:40
$ python3 app.py.
python3: can't open file 'app.py.': [Errno 2] No such file or directory

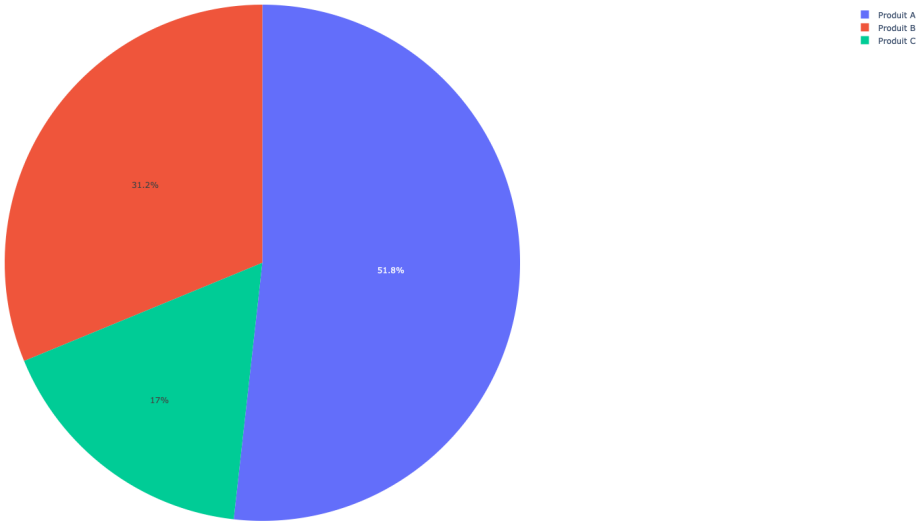
app@stormy-abalone-sprite:~ 07:40
$ python3 app.py
ventes-par-région.html généré avec succès !

app@stormy-abalone-sprite:~ 07:40
$
```

6. EN VOUS APPUYANT SUR L'EXEMPLE DONNÉ, CRÉER DEUX NOUVEAUX GRAPHIQUES :

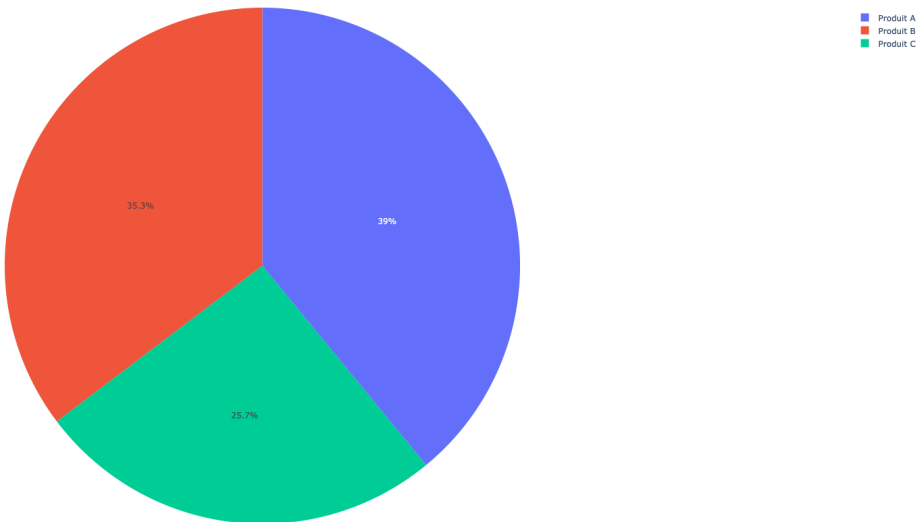
a. les ventes par produit,

quantité vendue par produit



b. le chiffre d'affaires par produit.

CA par produit



ANNEXE : CODE TAPÉ SUR GLITCH :

```
# import de plotly pour tracer les graphiques  
import plotly.express as px
```

```
# import de pandas pour manipuler les données  
import pandas as pd
```

```
#télécharger puis lecture du fichier CSV ventes depuis une URL grace à pandas.  
données = pd.read_csv('https://docs.google.com/spreadsheets/d/e/  
2PACX-1vSC4KusfFzvOsr8WJRgozssCxrELW4G4PopUkiDbvrrV2lg0S19-  
zeryp02MC9WYSVBuzGCUtn8ucZW/pub?output=csv')
```

```
#créer le graphique "vente par produit" avec les données voulues grace à plotly  
figure = px.pie(données, values='qte', names='produit', title='quantité vendue par produit')
```

```
#exporter le graphique "vente par produit" en format page web grace à plotly  
figure.write_html('ventes-par-produit.html')
```

```
#afficher le message "vente par produit" généré avec succès apres execution du code  
print('ventes-par-produit.html généré avec succès !')
```

```
#créer le graphique "vente par region" avec les données voulues grace à plotly  
figure = px.pie(données, values='qte', names='region', title='quantité vendue par region')
```

```
#exporter le graphique "vente par region" en format page web grace à plotly  
figure.write_html('ventes-par-region.html')
```

```
#afficher le message "vente par region" généré avec succès apres execution du code  
print('ventes-par-region.html généré avec succès !')
```

```
# Calcul du chiffre d'affaires par produit (qte * prix)  
données['chiffre_affaire'] = données['qte'] * données['prix']
```

```
#créer le graphique "CA par produit" avec les données voulues grace à plotly  
figure = px.pie(données, values= 'chiffre_affaire', names='produit', title='CA par produit')
```

```
#exporter le graphique "CA par produit" en format page web grace à plotly  
figure.write_html('CA-par-produit.html')
```

```
#afficher le message "CA par produit" généré avec succès apres execution du code  
print('CA-par-produit.html généré avec succès !')
```