Francisco Javier Ventura Segura

Reporte final
Proyecto 1

GitHub - PacoV3
https://github.com/PacoV3

Repo del proyecto
https://github.com/PacoV3/emtech-p1

Septiembre 13 del 2021

Introducción

La problemática que se presentó fue el desarrollar un programa en Python que permita analizar el historial de ventas de una empresa dedicada a la venta de tecnología, problema que puede encontrarse común mente en muchos comercios de la actualidad, los cuales buscan una mejor estrategia de ventas conociendo que productos son los que vende mejor.

Tomando en cuenta lo anterior Python cumple con las capacidades para poder analizar los datos que tiene el cliente, a través del uso de funciones, condiciones y ciclos se pueden analizar los registros para poder entregar al cliente de manera correcta los datos que necesita para analizar sus ventas a lo largo del tiempo. Cabe recalcar que Python al ser un lenguaje de uso general puede también brindar la aplicación de un sistema “Login” para que el cliente siempre pueda guardar su información de forma privada.

Definición del código

La primera parte de código consta de la implementación de una “base de datos” de usuarios, la cual por el momento fue representada con un diccionario dentro del programa, Seguida de una función que se encarga de manejar la lógica del “Login”.

```
# Create some users for the program
users = [{'username': 'paco123', 'password': 'pass123'},
         {'username': 'paco456', 'password': 'pass456'},
         {'username': 'javier123', 'password': 'javier123'},
         {'username': 'javier456', 'password': 'javier456'}]
# Go through the login process
login(users)
```

Lo que continúa después del proceso de “Login” es el análisis de los datos, separados por qué variable dentro de los datos se decide analizar el historial de ventas. Cada uno tiene por lo menos una función para calcular y otra para imprimir el resultado.

```
print('------------------- BEST SELLERS ----------------------------------------------------')
sellers = organize_products(lifestore_sales)
print_point11(sellers[0:50], lifestore_products, 'ventas.')
print('------------------ MOST SEARCHED ----------------------------------------------------')
most_search = organize_products(lifestore_searches)
print_point11(most_search[0:100], lifestore_products, 'búsquedas.')
print('------------- WORST SELLERS BY CATEGORY ----------------------------------------------')
category_sellers = organize_categories(lifestore_sales, lifestore_products)
print_point12(category_sellers, 'ventas.')
print('------------- LEAST SEARCHED BY CATEGORY ---------------------------------------------')
category_least_search = organize_categories(lifestore_searches, lifestore_products)
print_point12(category_least_search, 'búsquedas.')
print('--------------- BEST REVIEWS BY PRODUCT ----------------------------------------------')
reviews = organize_product_reviews(lifestore_sales)
print_point2(reviews[:-20:-1], lifestore_products, 'en calificación.')
print('--------------- WORST REVIEWS BY PRODUCT ----------------------------------------------')
print_point2(reviews[:20], lifestore_products, 'en calificación.')
print('------------------------- REVENUE ----------------------------------------------------')
year_n_month_sells = revenue_by_month_n_year(lifestore_sales, lifestore_products)
total_revenue = calc_total_revenue(year_n_month_sells)
print_total_revenue(total_revenue)
year_revenue = calc_revenue_by_year(year_n_month_sells)
print_revenue_by_year(year_revenue)
print_revenue_by_month(year_n_month_sells)
```

Solución del problema

La solución del problema constó de importar primero los datos del archivo donde se registraron las ventas y productos asimilando un proceso común de consultar una base de datos para luego ir creando funciones que los interpretarán a través de ciclos, debido a que puede interpretarse que estaban de una forma tabulada, en donde cada nuevo elemento del ciclo sería un registro de la base de datos.

La estructura más útil a lo largo de la solución fueron los diccionarios, debido a que permitían organizar los registros en distintos grupos que luego podían ser ordenados al pasarlos por un algoritmo de ordenamiento con base en uno de estos grupos ya mencionados.

Un ejemplo claro es el siguiente código:

El cual comienza por recibir toda la información sobre las ventas y los productos actuales para después recorrer cada venta e ir guardando en diferentes años junto con meses, el total en ventas para ese conjunto de datos. Algoritmo que con otra estructura hubiera sido más difícil de analizar.

```
def revenue_by_month_n_year(lifestore_sales, lifestore_products):
    years_n_months = {}
    for _, product_id, _, sale_date, refund in lifestore_sales:
        product_price = lifestore_products[product_id-1][2]
        formatted_date = datetime.strptime(sale_date, '%d/%m/%Y')
        year = formatted_date.year
        month = formatted_date.month
        if not refund:
            if year in years_n_months:
                if month in years_n_months[year]:
                    years_n_months[year][month] += product_price
                else:
                    years_n_months[year][month] = product_price
            else:
                years_n_months[year] = {month: product_price}
    return years_n_months
```

El output puede ser encontrado en el archivo output.txt.

Conclusión

En conclusión el proyecto funciono como una introducción a un problema común en el área del análisis de datos, debido a que el manejar variables relacionadas con la venta de uno o varios productos se viven en el día a día de la mayoría de las empresas.

Por lo dicho anteriormente es importante conocer como realizar este tipo de análisis de manera inmediata, además de constantemente practicar como se puede llevar a cabo con distintos problemas y proyectos como el presente.