

Sistema de control de stock: 'Nutrición Animal'



Institución: Instituto Superior Politécnico de Córdoba

Integrantes: Fabricio Campillay

Índice:

1. DESCRIPCIÓN (¿Qué?)
2. JUSTIFICACIÓN (¿Por qué?)
3. OBJETIVOS (¿Para qué?)
4. METODOLOGÍA (¿Cómo?)
5. CRONOGRAMA
6. PRESENTACION DEL PROYECTO
7. CONCLUSIONES

1. Descripción del proyecto

El proyecto consiste en desarrollar una aplicación en Python para la gestión de usuarios y accesos, la conexión con una base de datos para realizar consultas de inventario y proveedores, y el análisis de datos de precipitaciones con visualización gráfica. Esta aplicación incluye un sistema de CRUD (Crear, Leer, Actualizar y Eliminar) de usuarios con autenticación de acceso, registro de accesos exitosos y fallidos, y almacenamiento de datos en archivos binarios. Además, permite ejecutar consultas SQL en la base de datos 'nutricion_animal' para obtener información relevante de productos, proveedores y órdenes de despacho y realizar un análisis de datos pluviales con almacenamiento y visualización de dichos datos en gráficos específicos.

La aplicación fue desarrollada en varias fases:

- a) Definición de clases y estructura: Se diseñaron las clases Usuario y Acceso, junto con los módulos de gestión 'gestionUsuario', 'gestionAcceso', 'gestionBusqueda' y 'analisisDatos'.
- b) Desarrollo del CRUD de Usuarios y Accesos: Se implementaron métodos de creación, modificación, eliminación y visualización de usuarios y accesos en archivos binarios.
- c) Integración de base de datos: Se integró el módulo de consultas SQL mediante 'mysql.connector', permitiendo ejecutar consultas específicas de inventario y proveedores.
- d) Análisis de datos pluviales: Se implementaron funciones para crear y analizar registros de precipitaciones mediante Pandas, Numpy y Matplotlib, generando archivos CSV y gráficos de visualización.
- e) Desarrollo de interfaz de menú en consola: Se implementó un menú principal y submenús para acceder a cada funcionalidad de la aplicación, permitiendo al usuario navegar fácilmente por las opciones.

2. Justificación

Este proyecto es importante porque responde a la necesidad de gestionar usuarios y accesos de forma eficiente y segura, además de la importancia de registrar y analizar datos específicos, como la información de inventario y registros de precipitaciones. Para el proponente, esta aplicación representa una oportunidad de poner en práctica conocimientos en Python, gestión de bases de datos y análisis de datos, integrando varias áreas del desarrollo de software.

Desde un contexto práctico, la aplicación permite a organizaciones que manejan inventarios y datos almacenar, consultar y analizar información relevante de manera accesible. El sistema de gestión de usuarios, con autenticación y registro de accesos, asegura un control de seguridad adecuado para sistemas de datos sensibles. Además, al incluir consultas SQL a una base

de datos de inventario y análisis de datos pluviales, el proyecto resulta útil para áreas como la gestión de inventarios en empresas agrícolas o ganaderas y la supervisión de datos climatológicos.

3. Objetivos

Objetivo general:

Desarrollar una aplicación en Python para la gestión de usuarios, accesos, inventarios y análisis de datos pluviales, integrando consultas SQL y visualización gráfica para brindar una solución de administración de información completa.

Objetivos específicos:

- a) Implementar un sistema de CRUD para usuarios y accesos que permita gestionar datos de usuarios y registros de autenticación en archivos binarios.
- b) Conectar la aplicación a una base de datos 'nutricion_animal' para realizar consultas SQL que faciliten el acceso a datos de inventario y proveedores.
- c) Crear una funcionalidad de análisis de datos pluviales utilizando Pandas y Numpy, generando gráficos con Matplotlib para visualizar el comportamiento anual y mensual de las precipitaciones.

4. Metodología

Organización y Gestión

La aplicación fue organizada en módulos, lo cual permitió un desarrollo ordenado y modularizado. Se gestionó mediante la creación de clases específicas para usuarios y accesos, cada una con sus métodos y módulos independientes para la conexión a la base de datos, el análisis de datos y la ejecución de consultas. La aplicación fue desarrollada por un equipo técnico compuesto por un desarrollador con conocimientos en Python, bases de datos SQL y visualización de datos mediante el uso de librerías de Python.

Actividades

- a) Planeación: Se definieron los requisitos y la estructura general de la aplicación, especificando las funcionalidades y la organización en módulos.
- b) Desarrollo: El proyecto se implementó siguiendo las fases propuestas, comenzando por el CRUD de usuarios, seguido de la conexión a la base de datos y la funcionalidad de análisis de datos pluviales.

- c) Pruebas y Validación: Se realizaron pruebas en cada módulo para asegurar el correcto funcionamiento de las funcionalidades de gestión de usuarios, consultas a la base de datos y análisis de datos.

Logística y Producción

La aplicación se desarrolló y ejecutó en un entorno local con Python y la base de datos MySQL. Se utilizaron herramientas de análisis de datos (Pandas, Numpy) y visualización (Matplotlib). Los recursos económicos incluyeron el tiempo de desarrollo y herramientas de software libre, por lo que no hubo costos adicionales en infraestructura o licencias.

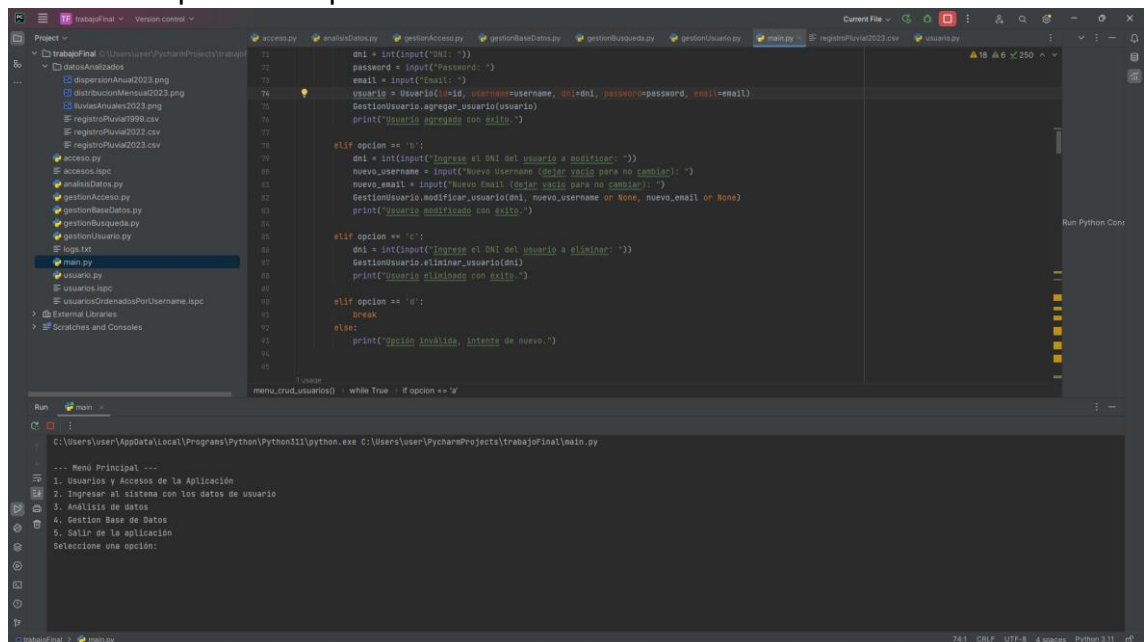
5. Cronograma

Fase	Actividad	Fecha de Inicio	Fecha de finalización
Planeación	Definición de requisitos y estructura	1 de Octubre	3 de Octubre
Implementación CRUD	Desarrollo del CRUD de usuarios y accesos	4 de Octubre	10 de Octubre
Conexión a la base de datos	Configuración y consultas SQL	11 de Octubre	15 de Octubre
Análisis de datos pluviales	Creación y análisis de registros de precipitaciones	29 de Octubre	31 de Octubre
Pruebas y validación	Pruebas unitarias y de integración	21 de Octubre	31 de Octubre
Documentación	Creación de README.md y documentación final	28 de Octubre	31 de Octubre

6. Presentación del proyecto

A continuación, se presentan algunas capturas de pantalla de las principales funcionalidades de la aplicación:

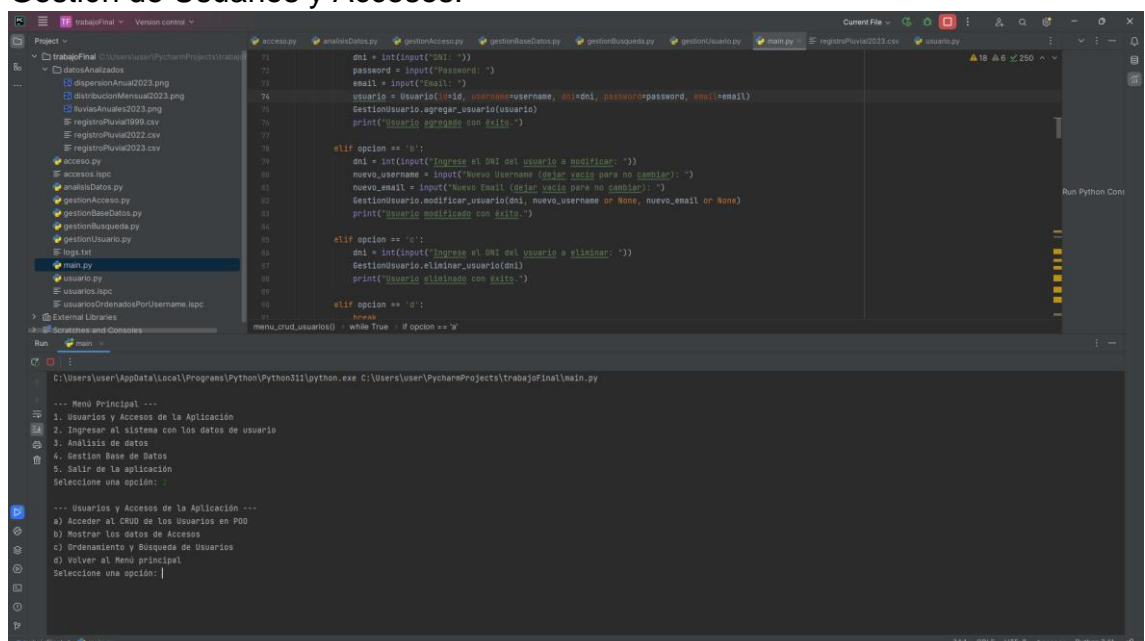
a) Menú Principal de la Aplicación:



```
71 dni = int(input("DNI: "))
72 password = input("Password: ")
73 email = input("Email: ")
74 usuario = Usuario(dni, username=username, dni=dni, password=password, email=email)
75 GestionUsuario.agregar_usuario(usuario)
76 print("Usuario agregado con éxito.")
77
78 elif opcion == 'b':
79 dni = int(input("Ingresar el DNI del usuario a modificar: "))
80 nuevo_username = input("Nuevo Username (dejar vacío para no cambiar): ")
81 nuevo_email = input("Nuevo Email (dejar vacío para no cambiar): ")
82 GestionUsuario.modificar_usuario(dni, nuevo_username or None, nuevo_email or None)
83 print("Usuario modificado con éxito.")
84
85 elif opcion == 'c':
86 dni = int(input("Ingresar el DNI del usuario a eliminar: "))
87 GestionUsuario.eliminar_usuario(dni)
88 print("Usuario eliminado con éxito.")
89
90 elif opcion == 'd':
91 break
92 else:
93 print("Opción inválida, intenta de nuevo.")
94
95
96 menu_crud_usuarios() while True: if opcion == 'q'
```

```
C:\Users\User\AppData\Local\Programs\Python\Python311\python.exe C:\Users\User\PycharmProjects\trabajoFinal\main.py
--- Menú Principal ---
1. Usuarios y Accesos de la Aplicación
2. Ingresar al sistema con los datos de usuario
3. Análisis de datos
4. Gestión Base de Datos
5. Salir de la aplicación
Seleccione una opción: 1
```

b) Gestión de Usuarios y Accesos:



```
71 dni = int(input("DNI: "))
72 password = input("Password: ")
73 email = input("Email: ")
74 usuario = Usuario(dni, username=username, dni=dni, password=password, email=email)
75 GestionUsuario.agregar_usuario(usuario)
76 print("Usuario agregado con éxito.")
77
78 elif opcion == 'b':
79 dni = int(input("Ingresar el DNI del usuario a modificar: "))
80 nuevo_username = input("Nuevo Username (dejar vacío para no cambiar): ")
81 nuevo_email = input("Nuevo Email (dejar vacío para no cambiar): ")
82 GestionUsuario.modificar_usuario(dni, nuevo_username or None, nuevo_email or None)
83 print("Usuario modificado con éxito.")
84
85 elif opcion == 'c':
86 dni = int(input("Ingresar el DNI del usuario a eliminar: "))
87 GestionUsuario.eliminar_usuario(dni)
88 print("Usuario eliminado con éxito.")
89
90 elif opcion == 'd':
91 break
92 else:
93 print("Opción inválida, intenta de nuevo.")
94
95
96 menu_crud_usuarios() while True: if opcion == 'q'
```

```
C:\Users\User\AppData\Local\Programs\Python\Python311\python.exe C:\Users\User\PycharmProjects\trabajoFinal\main.py
--- Menú Principal ---
1. Usuarios y Accesos de la Aplicación
2. Ingresar al sistema con los datos de usuario
3. Análisis de datos
4. Gestión Base de Datos
5. Salir de la aplicación
Seleccione una opción: 1
--- Usuarios y Accesos de la Aplicación ---
a) Acceder al CRUD de los Usuarios en PGO
b) Mostrar los datos de Accesos
c) Ordenamiento y Búsqueda de Usuarios
d) Volver al Menú principal
e) Salir de la aplicación
Seleccione una opción: a
```

- CRUD de usuarios:

```

71 dni = int(input("DNI: "))
72 password = input("Password: ")
73 email = input("Email: ")
74 usuario = Usuario(dni, username=username, dni=dni, password=password, email=email)
75 GestionUsuario.agregar_usuario(usuario)
76 print("Usuario agregado con éxito.")
77
78 elif opcion == 'b':
79     dni = int(input("Ingrese el DNI del usuario a modificar: "))
80     nuevo_username = input("Nuevo Username (dejar vacío para no cambiar): ")
81     nuevo_email = input("Nuevo Email (dejar vacío para no cambiar): ")
82     GestionUsuario.modificar_usuario(dni, nuevo_username or None, nuevo_email or None)
83     print("Usuario modificado con éxito.")
84
85 elif opcion == 'c':
86     menu_crud_usuarios() while True: if opcion == 'q'

```

```

--- Menu Principal ---
1. Usuarios y Accesos de la Aplicación
2. Ingresar al sistema con los datos de usuario
3. Analisis de datos
4. Gestión Base de Datos
5. Salir de la aplicación
Seleccione una opción:

--- Usuarios y Accesos de la Aplicación ---
a) Acceder al CRUD de los Usuarios en PBD
b) Mostrar los datos de Accesos
c) Ordenamiento y Búsqueda de Usuarios
d) Volver al Menu Principal
Seleccione una opción:

--- CRUD de Usuarios ---
a) Agregar un nuevo usuario
b) Modificar un usuario
c) Eliminar un usuario (debe su DNI)
d) Volver al menu anterior
Seleccione una opción:

```

c) Gestión de Base de Datos:

```

71 dni = int(input("DNI: "))
72 password = input("Password: ")
73 email = input("Email: ")
74 usuario = Usuario(dni, username=username, dni=dni, password=password, email=email)
75 GestionUsuario.agregar_usuario(usuario)
76 print("Usuario agregado con éxito.")
77
78 elif opcion == 'b':
79     dni = int(input("Ingrese el DNI del usuario a modificar: "))
80     nuevo_username = input("Nuevo Username (dejar vacío para no cambiar): ")
81     nuevo_email = input("Nuevo Email (dejar vacío para no cambiar): ")
82     GestionUsuario.modificar_usuario(dni, nuevo_username or None, nuevo_email or None)
83     print("Usuario modificado con éxito.")
84
85 elif opcion == 'c':
86     dni = int(input("Ingrese el DNI del usuario a eliminar: "))
87     GestionUsuario.eliminar_usuario(dni)
88     print("Usuario eliminado con éxito.")
89
90 elif opcion == 'd':
91     break
92 else:
93     print("Opción inválida, intento de nuevo.")
94
95 menu_crud_usuarios() while True: if opcion == 'q'

```

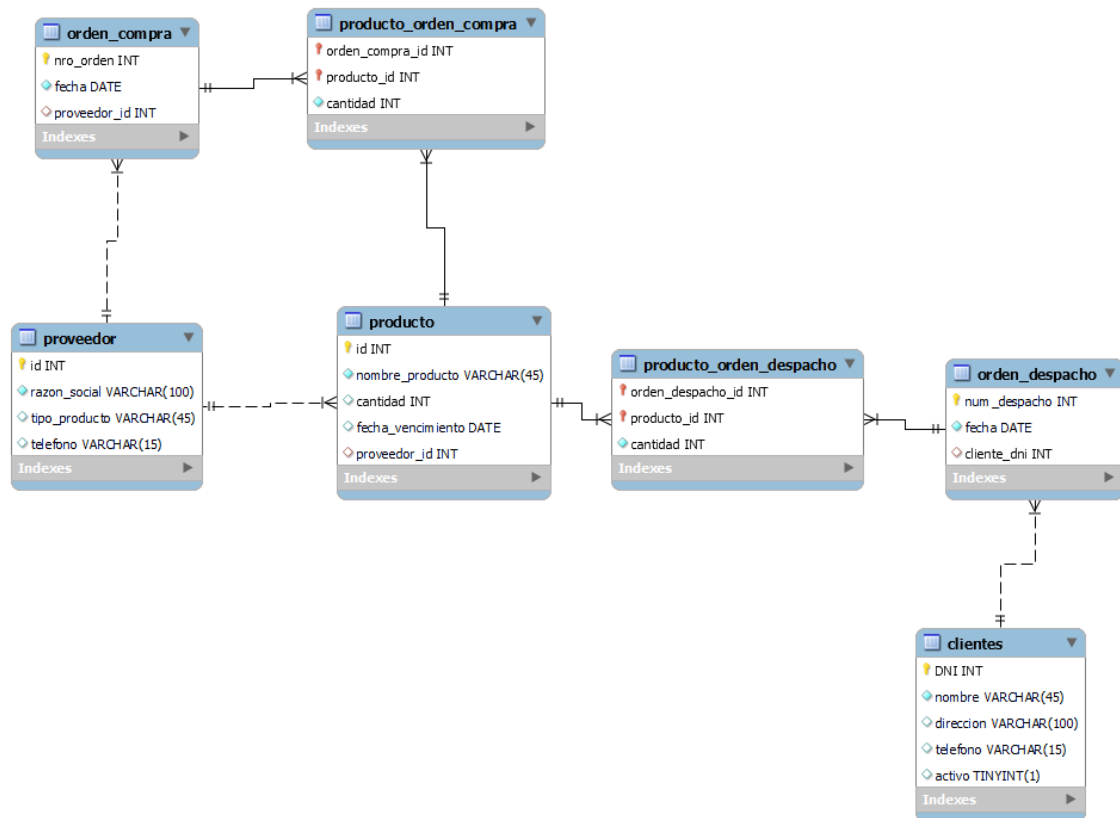
```

1. Usuarios y Accesos de la Aplicación
2. Ingresar al sistema con los datos de usuario
3. Analisis de datos
4. Gestión Base de Datos
5. Salir de la aplicación
Seleccione una opción:
[Info] Conexión a la base de datos exitosa.

--- Gestión de Base de Datos ---
a) Cantidad despachada por Juan Perez
b) Productos cuya cantidad está por debajo del promedio
c) Productos con total de cantidad superior a 200
d) Productos con fecha de vencimiento posterior a una fecha específica
e) Productos con cantidades menores a 100 y fecha de vencimiento anterior a una fecha específica
f) Clasificación de productos en niveles de stock
g) Volver al menu principal
Seleccione una opción:

```

- Diagrama ER.



- Ejecución de consultas SQL.
 - 1. cantidad despachada por 'juan perez'.

```

1  USE nutricion_animal;
2  --- Consultas de Join
3  SELECT c.nombre AS nombre_cliente, o.num_despacho, p.nombre_producto, pod.cantidad
4  FROM orden_despacho o
5  JOIN clientes c ON o.cliente_dni = c.DNI
6  JOIN producto_orden_despacho pod ON o.num_despacho = pod.orden_despacho_id
7  JOIN producto p ON pod.producto_id = p.id
8  WHERE c.DNI = 12345678;
9
10
11
12
  
```

nombre_cliente	num_despacho	nombre_producto	cantidad
Juan Pérez	1	Alimento para terneros	20

- 2. Productos cuya cantidad está por debajo del promedio.

The screenshot shows a SQL IDE window with a query editor and a result grid. The query editor contains the following SQL code:

```
1 • USE nutricion_animal;
2 --- Consultas con Subconsultas
3 • SELECT nombre_producto, cantidad
4 FROM producto
5 WHERE cantidad < (SELECT AVG(cantidad) FROM producto);
6
7
8
9
```

The result grid displays the following data:

nombre_producto	cantidad
Alimento para terneros	100
Alimento para ñandúes	150

- 3. Proveedores con total de cantidad superior a 200.

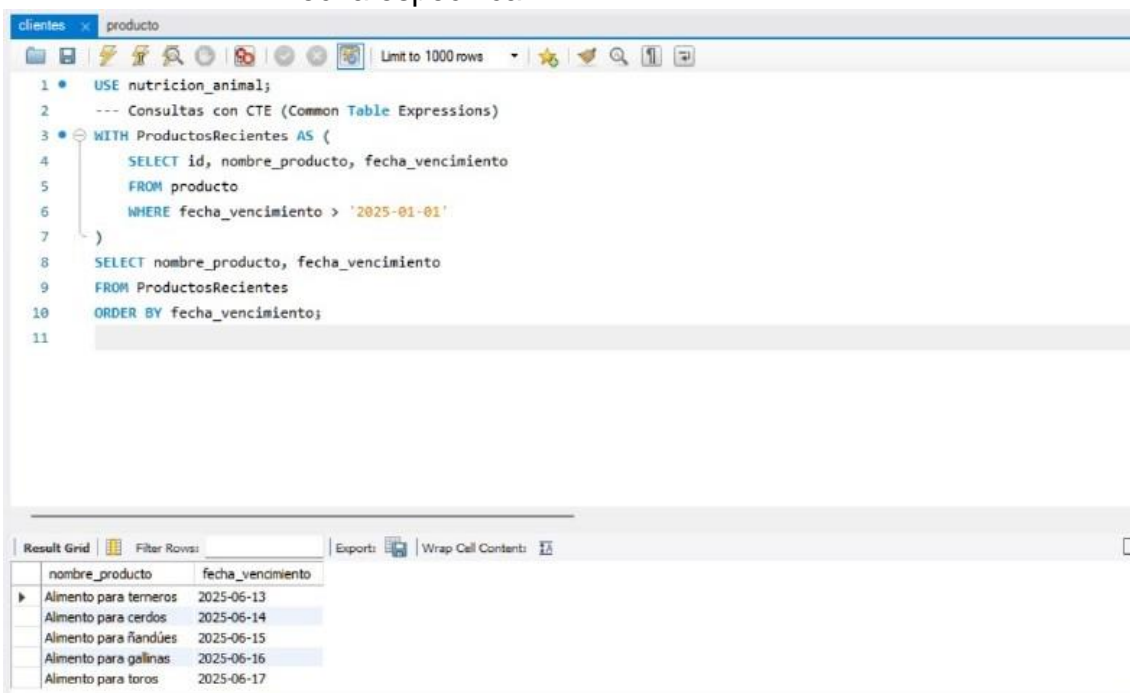
The screenshot shows a SQL IDE window with a query editor and a result grid. The query editor contains the following SQL code:

```
1 • USE nutricion_animal;
2 --- Consultas con Funciones de Agregación
3 • SELECT proveedor_id, SUM(cantidad) AS total_cantidad
4 FROM producto
5 GROUP BY proveedor_id
6 HAVING SUM(cantidad) > 200;
7
```

The result grid displays the following data:

proveedor_id	total_cantidad
4	250

- 4. Productos con fecha de vencimiento posterior a una fecha específica.



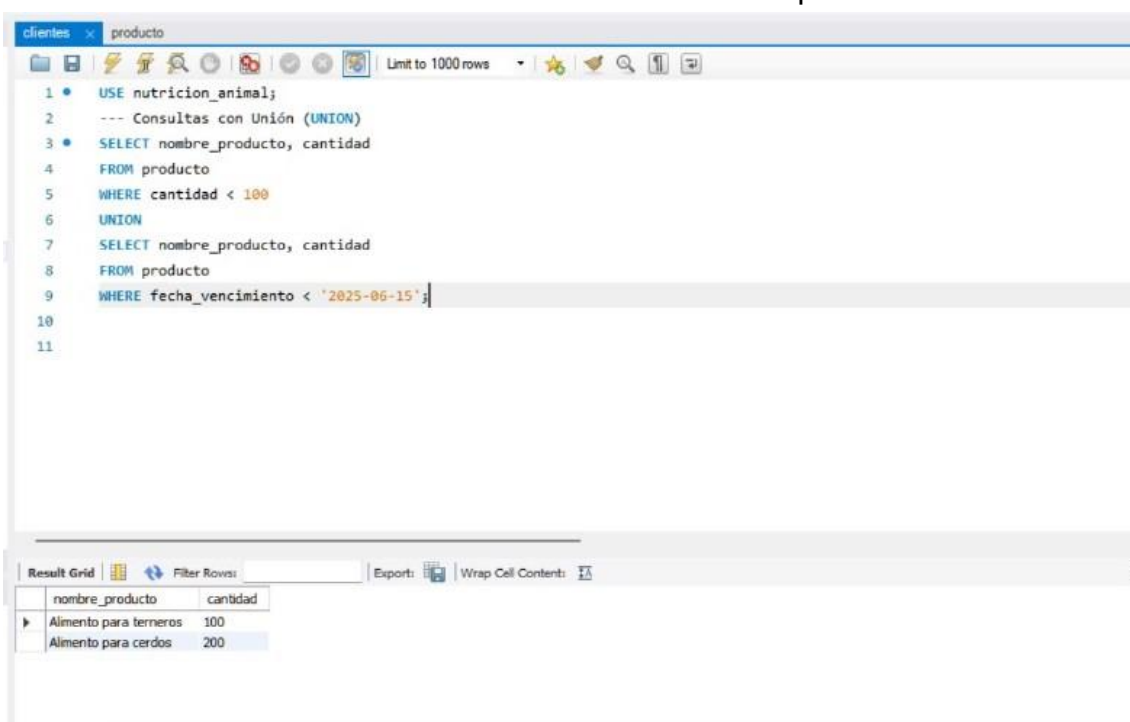
```

1 • USE nutricion_animal;
2 --- Consultas con CTE (Common Table Expressions)
3 • WITH ProductosRecientes AS (
4     SELECT id, nombre_producto, fecha_vencimiento
5     FROM producto
6     WHERE fecha_vencimiento > '2025-01-01'
7 )
8 SELECT nombre_producto, fecha_vencimiento
9 FROM ProductosRecientes
10 ORDER BY fecha_vencimiento;
11

```

nombre_producto	fecha_vencimiento
Alimento para terneros	2025-06-13
Alimento para cerdos	2025-06-14
Alimento para ñandúes	2025-06-15
Alimento para gallinas	2025-06-16
Alimento para toros	2025-06-17

- 5. Productos con cantidades menores a 100 y fecha de vencimiento anterior a una fecha específica.



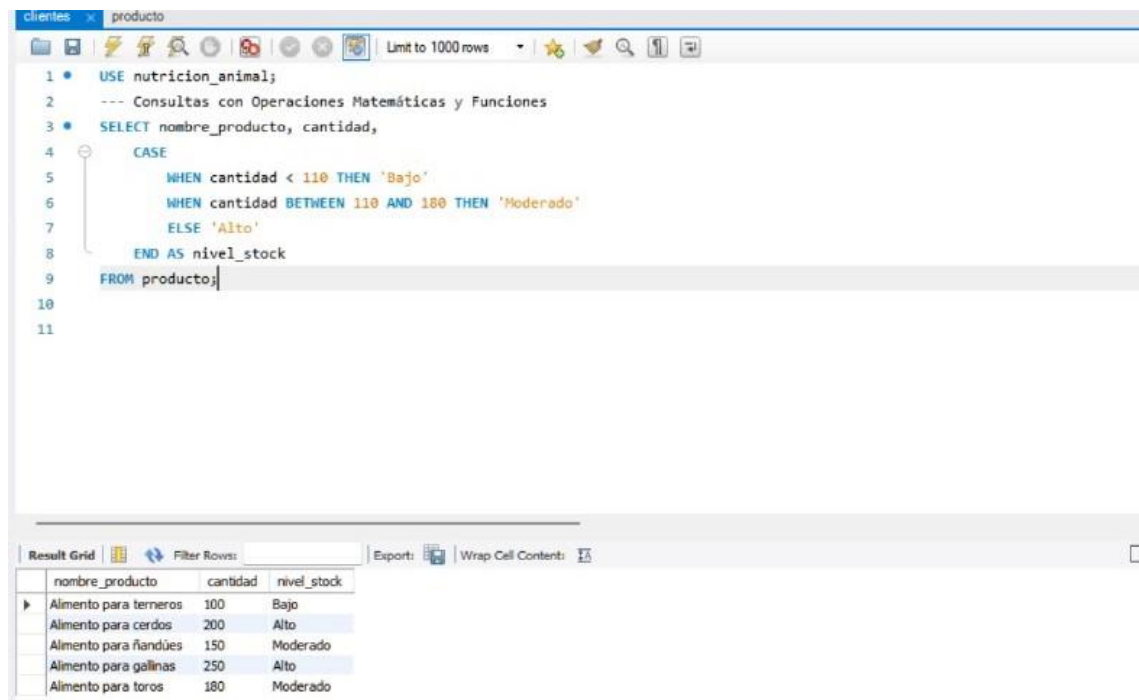
```

1 • USE nutricion_animal;
2 --- Consultas con Unión (UNION)
3 • SELECT nombre_producto, cantidad
4 FROM producto
5 WHERE cantidad < 100
6 UNION
7 SELECT nombre_producto, cantidad
8 FROM producto
9 WHERE fecha_vencimiento < '2025-06-15';
10
11

```

nombre_producto	cantidad
Alimento para terneros	100
Alimento para cerdos	200

○ 6. Clasificación de productos en niveles de stock.



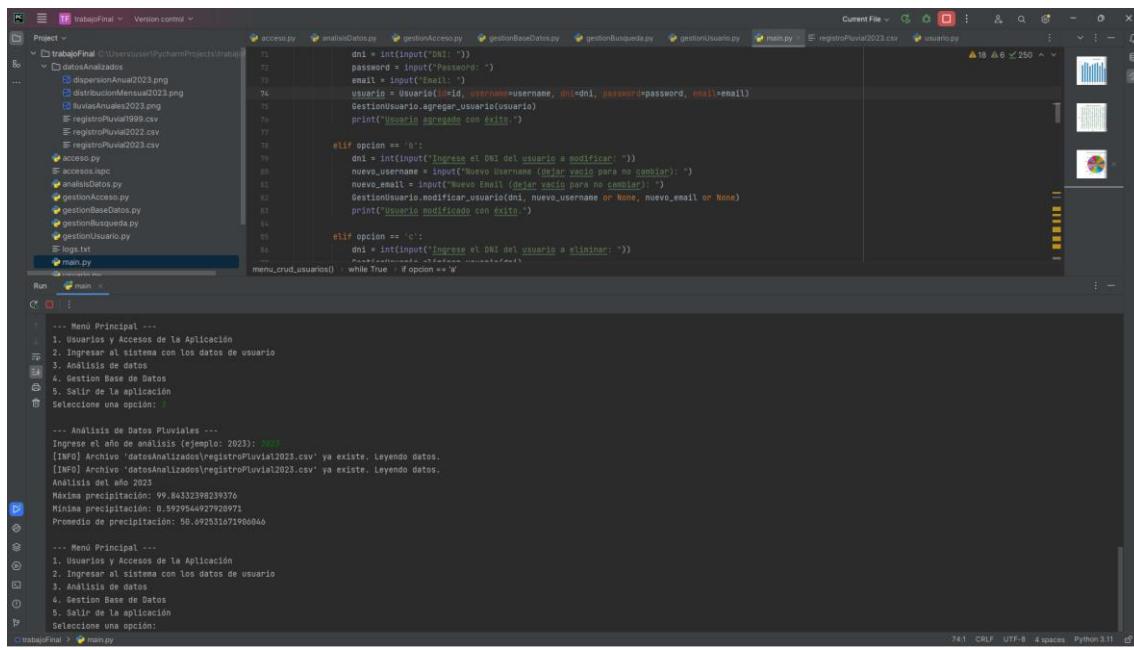
```

1 • USE nutricion_animal;
2 --- Consultas con Operaciones Matemáticas y Funciones
3 • SELECT nombre_producto, cantidad,
4     CASE
5     WHEN cantidad < 110 THEN 'Bajo'
6     WHEN cantidad BETWEEN 110 AND 180 THEN 'Moderado'
7     ELSE 'Alto'
8     END AS nivel_stock
9 FROM producto;
10
11

```

nombre_producto	cantidad	nivel_stock
Alimento para terneros	100	Bajo
Alimento para cerdos	200	Alto
Alimento para ñandúes	150	Moderado
Alimento para gallinas	250	Alto
Alimento para toros	180	Moderado

d) Análisis de Datos Pluviales:



```

--- Menú Principal ---
1. Usuarios y Accesos de la Aplicación
2. Ingresar al sistema con los datos de usuario
3. Análisis de datos
4. Gestión Base de Datos
5. Salir de la aplicación
Seleccione una opción: 3

--- Análisis de Datos Pluviales ---
Ingrese el año de análisis (ejemplo: 2023): 2023
[INFO] Archivo 'datosAnálisis(registroPluvial2023.csv)' ya existe. Leyendo datos.
Análisis del año 2023
Máxima precipitación: 99.8633239829376
Mínima precipitación: 0.592954492792971
Promedio de precipitación: 56.09253167190846

--- Menú Principal ---
1. Usuarios y Accesos de la Aplicación
2. Ingresar al sistema con los datos de usuario
3. Análisis de datos
4. Gestión Base de Datos
5. Salir de la aplicación
Seleccione una opción:

```

7. Conclusiones

El desarrollo de esta aplicación ha demostrado ser una solución efectiva para gestionar usuarios, controlar accesos y realizar análisis de datos, integrando varias tecnologías y metodologías de programación. El sistema modularizado permite una fácil adaptación y escalabilidad y el uso de Python y MySQL garantiza la flexibilidad y potencia necesarias para manejar grandes volúmenes de datos.

La implementación de consultas SQL ha permitido el acceso a datos complejos de inventario y proveedores, beneficiando sectores que dependen de la gestión de inventarios y de datos de clientes. Asimismo, la integración de un análisis de datos pluviales con gráficos proporciona una herramienta útil para visualizar datos climatológicos, lo cual podría ser de interés en sectores agrícolas como es el caso de nuestro proyecto o de investigación ambiental.