[[1]](#footnote-1)

Primer Examen Parcial

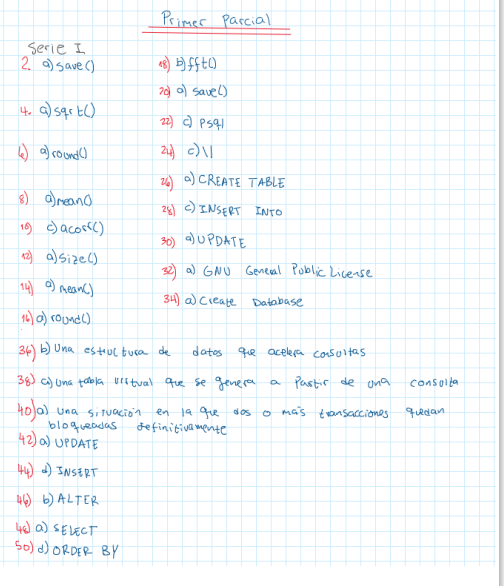
Francisco Alberto, Navarro Orozco, 202004752

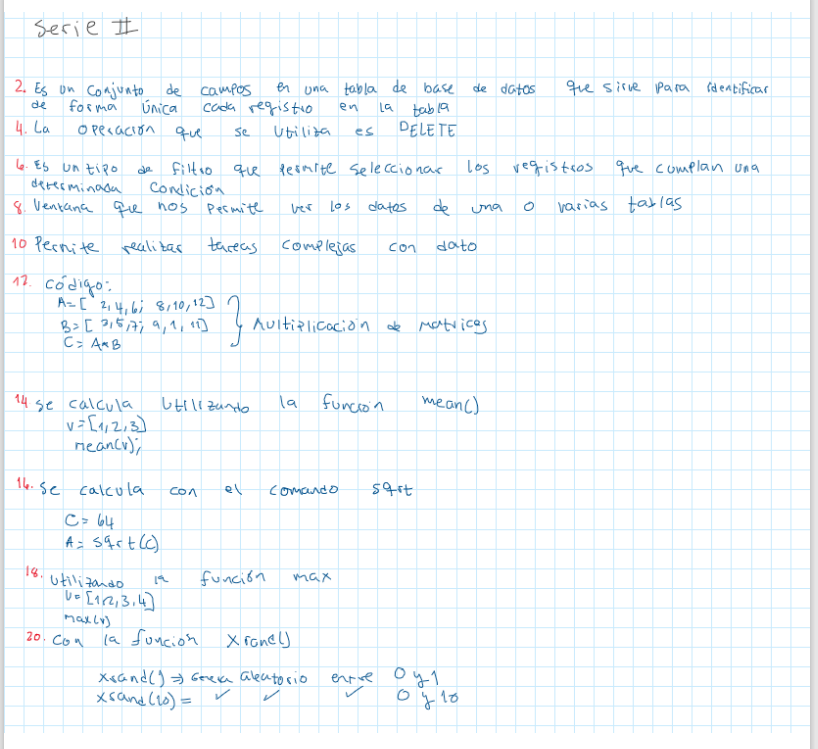
*Escuela de Mecánica Eléctrica, Facultad de Ingeniería, Universidad de San Carlos de Guatemala*

*Para el primer examen Parcial se realizaron un conjunto de programas en Python y Octave conectándolos con la base de datos PostegreSQL*

# Código utilizado

* Primera Serie



* Segunda Serie
* Tercera Serie
* Primer Programa
* import psycopg2
* def connect\_to\_database():
* try:
* connection = psycopg2.connect(
* user="postgres",
* password="2405",
* host="localhost",
* port="5433",
* database="0980 Proyectos"
* )
* return connection
* except (Exception, psycopg2.Error) as error:
* print("Error al conectar a la base de datos:", error)
* return None
* def insert\_student(connection, nombre, edad, genero, direccion):
* try:
* cursor = connection.cursor()
* insert\_query = '''INSERT INTO estudiantes (nombre, edad, genero, direccion)
* VALUES (%s, %s, %s, %s);'''
* data = (nombre, edad, genero, direccion)
* cursor.execute(insert\_query, data)
* connection.commit()
* cursor.close()
* print("Estudiante agregado con éxito.")
* except (Exception, psycopg2.Error) as error:
* print("Error al agregar estudiante:", error)
* def update\_student(connection, estudiante\_id, campo, nuevo\_valor):
* try:
* cursor = connection.cursor()
* update\_query = f'''UPDATE estudiantes
* SET {campo} = %s
* WHERE id = %s;'''
* data = (nuevo\_valor, estudiante\_id)
* cursor.execute(update\_query, data)
* connection.commit()
* cursor.close()
* print("Información del estudiante actualizada con éxito.")
* except (Exception, psycopg2.Error) as error:
* print("Error al actualizar información del estudiante:", error)
* def delete\_student(connection, estudiante\_id):
* try:
* cursor = connection.cursor()
* delete\_query = '''DELETE FROM estudiantes WHERE id = %s;'''
* cursor.execute(delete\_query, (estudiante\_id,))
* connection.commit()
* cursor.close()
* print("Estudiante eliminado de la base de datos.")
* except (Exception, psycopg2.Error) as error:
* print("Error al eliminar estudiante:", error)
* def main():
* connection = connect\_to\_database()
* if connection:
* while True:
* print("\nOpciones:")
* print("1. Agregar estudiante")
* print("2. Editar información de estudiante")
* print("3. Eliminar estudiante")
* print("4. Salir")
* opcion = input("Selecciona una opción: ")
* if opcion == "1":
* nombre = input("Nombre del estudiante: ")
* edad = int(input("Edad: "))
* genero = input("Género: ")
* direccion = input("Dirección: ")
* insert\_student(connection, nombre, edad, genero, direccion)
* elif opcion == "2":
* estudiante\_id = int(input("ID del estudiante a editar: "))
* campo = input("Campo a editar (nombre/edad/genero/direccion): ")
* nuevo\_valor = input(f"Nuevo valor para {campo}: ")
* update\_student(connection, estudiante\_id, campo, nuevo\_valor)
* elif opcion == "3":
* estudiante\_id = int(input("ID del estudiante a eliminar: "))
* delete\_student(connection, estudiante\_id)
* elif opcion == "4":
* print("Saliendo del programa.")
* break
* else:
* print("Opción no válida. Por favor, selecciona una opción válida.")
* connection.close()
* if \_\_name\_\_ == "\_\_main\_\_":
* main()
* Segundo Programa
* import psycopg2
* def connect\_to\_database():
* try:
* connection = psycopg2.connect(
* user="postgres",
* password="2405",
* host="localhost",
* port="5433",
* database="0980 Proyectos"
* )
* return connection
* except (Exception, psycopg2.Error) as error:
* print("Error al conectar a la base de datos:", error)
* return None
* def insert\_expense(connection, fecha, categoria, descripcion, monto):
* try:
* cursor = connection.cursor()
* insert\_query = '''INSERT INTO gastos (fecha, categoria, descripcion, monto)
* VALUES (%s, %s, %s, %s);'''
* data = (fecha, categoria, descripcion, monto)
* cursor.execute(insert\_query, data)
* connection.commit()
* cursor.close()
* print("Gasto agregado con éxito.")
* except (Exception, psycopg2.Error) as error:
* print("Error al agregar gasto:", error)
* def generate\_report(connection):
* try:
* cursor = connection.cursor()
* report\_query = '''SELECT fecha, categoria, descripcion, monto
* FROM gastos
* ORDER BY fecha DESC;'''
* cursor.execute(report\_query)
* results = cursor.fetchall()
* cursor.close()
* print("Reporte de gastos:")
* for row in results:
* print(f"Fecha: {row[0]}, Categoría: {row[1]}, Descripción: {row[2]}, Monto: {row[3]}")
* except (Exception, psycopg2.Error) as error:
* print("Error al generar reporte:", error)
* def adjust\_budget(connection, categoria, nuevo\_monto):
* try:
* cursor = connection.cursor()
* update\_query = f'''UPDATE presupuestos
* SET monto = %s
* WHERE categoria = %s;'''
* data = (nuevo\_monto, categoria)
* cursor.execute(update\_query, data)
* connection.commit()
* cursor.close()
* print("Presupuesto ajustado con éxito.")
* except (Exception, psycopg2.Error) as error:
* print("Error al ajustar presupuesto:", error)
* def main():
* connection = connect\_to\_database()
* if connection:
* while True:
* print("\nOpciones:")
* print("1. Agregar gasto")
* print("2. Ver informe de gastos")
* print("3. Ajustar presupuesto")
* print("4. Salir")
* opcion = input("Selecciona una opción: ")
* if opcion == "1":
* fecha = input("Fecha del gasto (YYYY-MM-DD): ")
* categoria = input("Categoría del gasto: ")
* descripcion = input("Descripción del gasto: ")
* monto = float(input("Monto del gasto: "))
* insert\_expense(connection, fecha, categoria, descripcion, monto)
* elif opcion == "2":
* generate\_report(connection)
* elif opcion == "3":
* categoria = input("Categoría del presupuesto a ajustar: ")
* nuevo\_monto = float(input("Nuevo monto del presupuesto: "))
* adjust\_budget(connection, categoria, nuevo\_monto)
* elif opcion == "4":
* print("Saliendo del programa.")
* break
* else:
* print("Opción no válida. Por favor, selecciona una opción válida.")
* connection.close()
* if \_\_name\_\_ == "\_\_main\_\_":
* main()
* Tercero Programa
* import psycopg2
* def connect\_to\_database():
* try:
* connection = psycopg2.connect(
* user="postgres",
* password="2405",
* host="localhost",
* port="5433",
* database="0980 Proyectos"
* )
* return connection
* except (Exception, psycopg2.Error) as error:
* print("Error al conectar a la base de datos:", error)
* return None
* def insert\_product(connection, nombre, descripcion, cantidad, precio):
* try:
* cursor = connection.cursor()
* insert\_query = '''INSERT INTO inventarios (nombre, descripcion, cantidad, precio)
* VALUES (%s, %s, %s, %s);'''
* data = (nombre, descripcion, cantidad, precio)
* cursor.execute(insert\_query, data)
* connection.commit()
* cursor.close()
* print("Producto agregado con éxito.")
* except (Exception, psycopg2.Error) as error:
* print("Error al agregar producto:", error)
* def main():
* connection = connect\_to\_database()
* if connection:
* while True:
* print("\nOpciones:")
* print("1. Agregar producto")
* print("2. Salir")
* opcion = input("Selecciona una opción: ")
* if opcion == "1":
* nombre = input("Nombre del producto: ")
* descripcion = input("Descripción del producto: ")
* cantidad = int(input("Cantidad del producto: "))
* precio = float(input("Precio del producto: "))
* insert\_product(connection, nombre, descripcion, cantidad, precio)
* elif opcion == "2":
* print("Saliendo del programa.")
* break
* else:
* print("Opción no válida. Por favor, selecciona una opción válida.")
* connection.close()
* if \_\_name\_\_ == "\_\_main\_\_":
* main()
* Cuarto Programa
* import psycopg2
* def connect\_to\_database():
* try:
* connection = psycopg2.connect(
* user="postgres",
* password="2405",
* host="localhost",
* port="5433",
* database="0980 Proyectos"
* )
* return connection
* except (Exception, psycopg2.Error) as error:
* print("Error al conectar a la base de datos:", error)
* return None
* def insert\_sale(connection, fecha, producto, cantidad, precio):
* try:
* cursor = connection.cursor()
* insert\_query = '''INSERT INTO pedidos (fecha, producto, cantidad, precio)
* VALUES (%s, %s, %s, %s);'''
* data = (fecha, producto, cantidad, precio)
* cursor.execute(insert\_query, data)
* connection.commit()
* cursor.close()
* print("Pedido agregado con éxito.")
* except (Exception, psycopg2.Error) as error:
* print("Error al agregar pedido:", error)
* def edit\_sale(connection, id, fecha, producto, cantidad, precio):
* try:
* cursor = connection.cursor()
* update\_query = '''UPDATE pedidos
* SET fecha = %s,
* producto = %s,
* cantidad = %s,
* precio = %s
* WHERE id = %s;'''
* data = (fecha, producto, cantidad, precio, id)
* cursor.execute(update\_query, data)
* connection.commit()
* cursor.close()
* print("Pedido editado con éxito.")
* except (Exception, psycopg2.Error) as error:
* print("Error al editar pedido:", error)
* def delete\_sale(connection, id):
* try:
* cursor = connection.cursor()
* delete\_query = '''DELETE FROM pedidos
* WHERE id = %s;'''
* data = (id,)
* cursor.execute(delete\_query, data)
* connection.commit()
* cursor.close()
* print("Pedido eliminado con éxito.")
* except (Exception, psycopg2.Error) as error:
* print("Error al eliminar pedido:", error)
* def main():
* connection = connect\_to\_database()
* if connection:
* while True:
* print("\nOpciones:")
* print("1. Agregar pedido")
* print("2. Editar pedido")
* print("3. Eliminar pedido")
* print("4. Salir")
* opcion = input("Selecciona una opción: ")
* if opcion == "1":
* fecha = input("Fecha del pedido (YYYY-MM-DD): ")
* producto = input("Producto: ")
* cantidad = int(input("Cantidad: "))
* precio = float(input("Precio: "))
* insert\_sale(connection, fecha, producto, cantidad, precio)
* elif opcion == "2":
* id = input("ID del pedido a editar: ")
* fecha = input("Fecha nueva del pedido (YYYY-MM-DD): ")
* producto = input("Producto nuevo: ")
* cantidad = int(input("Cantidad nueva: "))
* precio = float(input("Precio nuevo: "))
* edit\_sale(connection, id, fecha, producto, cantidad, precio)
* elif opcion == "3":
* id = input("ID del pedido a eliminar: ")
* delete\_sale(connection, id)
* elif opcion == "4":
* print("Saliendo del programa.")
* break
* else:
* print("Opción no válida. Por favor, selecciona una opción válida.")
* connection.close()
* if \_\_name\_\_ == "\_\_main\_\_":
* main()
* Quinto Programa
* def connect\_to\_database():
* try:
* connection = psycopg2.connect(
* user="postgres",
* password="2405",
* host="localhost",
* port="5433",
* database="0980 Proyectos"
* )
* return connection
* except (Exception, psycopg2.Error) as error:
* print("Error al conectar a la base de datos:", error)
* return None
* def insert\_sale(connection, fecha, producto, cantidad, precio):
* try:
* cursor = connection.cursor()
* insert\_query = '''INSERT INTO ventas (fecha, producto, cantidad, precio)
* VALUES (%s, %s, %s, %s);'''
* data = (fecha, producto, cantidad, precio)
* cursor.execute(insert\_query, data)
* connection.commit()
* cursor.close()
* print("Venta agregada con éxito.")
* except (Exception, psycopg2.Error) as error:
* print("Error al agregar venta:", error)
* def generate\_report(connection):
* try:
* cursor = connection.cursor()
* report\_query = '''SELECT fecha, producto, cantidad, precio
* FROM ventas
* ORDER BY fecha DESC;'''
* cursor.execute(report\_query)
* results = cursor.fetchall()
* cursor.close()
* print("Reporte de ventas:")
* for row in results:
* print(f"Fecha: {row[0]}, Producto: {row[1]}, Cantidad: {row[2]}, Precio: {row[3]}")
* except (Exception, psycopg2.Error) as error:
* print("Error al generar reporte:", error)
* def analyze\_data(connection):
* try:
* cursor = connection.cursor()
* analysis\_query = '''SELECT producto, SUM(cantidad) AS cantidad,
* SUM(precio) AS precio
* FROM ventas
* GROUP BY producto;'''
* cursor.execute(analysis\_query)
* results = cursor.fetchall()
* cursor.close()
* for row in results:
* print(f"Producto: {row[0]}, Cantidad: {row[1]}, Precio: {row[2]}")
* print("Patrones y tendencias:")
* # Aquí puedes agregar tu código para analizar los datos y encontrar patrones y tendencias.
* # Por ejemplo, puedes calcular el crecimiento de las ventas, el producto más vendido, etc.
* except (Exception, psycopg2.Error) as error:
* print("Error al analizar datos:", error)
* def main():
* connection = connect\_to\_database()
* if connection:
* while True:
* print("\nOpciones:")
* print("1. Agregar venta")
* print("2. Generar reporte")
* print("3. Analizar datos")
* print("4. Salir")
* opcion = input("Selecciona una opción: ")
* if opcion == "1":
* fecha = input("Fecha de la venta (YYYY-MM-DD): ")
* producto = input("Producto: ")
* cantidad = int(input("Cantidad: "))
* precio = float(input("Precio: "))
* insert\_sale(connection, fecha, producto, cantidad, precio)
* elif opcion == "2":
* generate\_report(connection)
* elif opcion == "3":
* analyze\_data(connection)
* elif opcion == "4":
* print("Saliendo del programa.")
* break
* else:
* print("Opción no válida. Por favor, selecciona una opción válida.")
* connection.close()
* if \_\_name\_\_ == "\_\_main\_\_":
* main()
* Sexto Programa
* import psycopg2
* def connect\_to\_database():
* try:
* connection = psycopg2.connect(
* user="postgres",
* password="2405",
* host="localhost",
* port="5433",
* database="0980 Proyectos"
* )
* return connection
* except (Exception, psycopg2.Error) as error:
* print("Error al conectar a la base de datos:", error)
* return None
* def close\_connection(connection):
* try:
* connection.close()
* return None
* except (Exception, psycopg2.Error) as error:
* print("Error al cerrar la conexión:", error)
* return None
* def view\_sensor\_results\_graph(connection, sensor):
* import matplotlib.pyplot as plt
* try:
* cursor = connection.cursor()
* select\_query = '''SELECT calidad, precio
* FROM sensores
* WHERE sensor = %s;'''
* data = (sensor,)
* cursor.execute(select\_query, data)
* results = cursor.fetchall()
* cursor.close()
* if results:
* calidad = [row[0] for row in results]
* precio = [row[1] for row in results]
* plt.plot(calidad, precio, linewidth=2)
* plt.xlabel("Calidad")
* plt.ylabel("Precio")
* plt.title("Gráfica de calidad-precio de los sensores")
* plt.show()
* else:
* print("El sensor no existe.")
* except (Exception, psycopg2.Error) as error:
* print("Error al ver resultados del sensor:", error)
* def main():
* connection = connect\_to\_database()
* if connection:
* while True:
* print("\nOpciones:")
* print("1. Agregar sensor")
* print("2. Editar sensor")
* print("3. Ver resultados del sensor")
* print("4. Ver resultados del sensor en gráfica")
* print("5. Salir")
* opcion = input("Selecciona una opción: ")
* if opcion == "1":
* sensor = input("Nombre del sensor: ")
* calidad = input("Calidad del sensor: ")
* precio = float(input("Precio del sensor: "))
* add\_sensor(connection, sensor, calidad, precio)
* elif opcion == "2":
* sensor = input("Nombre del sensor: ")
* calidad = input("Calidad del sensor: ")
* precio = float(input("Precio del sensor: "))
* edit\_sensor(connection, sensor, calidad, precio)
* elif opcion == "3":
* sensor = input("Nombre del sensor: ")
* view\_sensor\_results(connection, sensor)
* elif opcion == "4":
* sensor = input("Nombre del sensor: ")
* view\_sensor\_results\_graph(connection, sensor)
* elif opcion == "5":
* print("Saliendo del programa.")
* close\_connection(connection)
* break
* else:
* print("Opción no válida. Por favor, selecciona una opción válida.")
* else:
* print("Error al conectar a la base de datos.")
* if \_\_name\_\_ == "\_\_main\_\_":
* main()
* Séptimo Programa
* def main():
* connection = connect\_to\_database()
* if connection:
* while True:
* print("\nOpciones:")
* print("1. Ingresar género")
* print("2. Ingresar calificación")
* print("3. Salir")
* opcion = input("Selecciona una opción: ")
* if opcion == "1":
* genre = input("Ingresa el género que deseas ver: ")
* elif opcion == "2":
* global rating
* rating = input("Ingresa la calificación que deseas (de 1 a 5): ")
* elif opcion == "3":
* print("Saliendo del programa.")
* close\_connection(connection)
* break
* else:
* print("Opción no válida. Por favor, selecciona una opción válida.")
* recommended\_movies = view\_recommended\_movie(connection, genre, rating)
* if recommended\_movies:
* print("Las películas recomendadas son:")
* for movie in recommended\_movies:
* if movie:
* print(movie)
* else:
* print("No se encontraron resultados que coincidan con los criterios.")
* else:
* print("No se encontraron resultados que coincidan con los criterios.")
* else:
* print("Error al conectar a la base de datos.")
* if \_\_name\_\_ == "\_\_main\_\_":
* main()
* Octavo Programa
* import psycopg2
* def connect\_to\_database():
* try:
* connection = psycopg2.connect(
* user="postgres",
* password="2405",
* host="localhost",
* port="5433",
* database="0980 Proyectos"
* )
* return connection
* except (Exception, psycopg2.Error) as error:
* print("Error al conectar a la base de datos:", error)
* return None
* def close\_connection(connection):
* try:
* connection.close()
* return None
* except (Exception, psycopg2.Error) as error:
* print("Error al cerrar la conexión:", error)
* return None
* def add\_company(connection, nombre, direccion, telefono, ingresos\_anuales, egresos\_anuales):
* try:
* cursor = connection.cursor()
* insert\_query = '''INSERT INTO empresas (nombre, direccion, telefono, ingresos\_anuales, egresos\_anuales)
* VALUES (%s, %s, %s, %s, %s);'''
* data = (nombre, direccion, telefono, ingresos\_anuales, egresos\_anuales)
* cursor.execute(insert\_query, data)
* connection.commit()
* cursor.close()
* print("Empresa agregada con éxito.")
* except (Exception, psycopg2.Error) as error:
* print("Error al agregar empresa:", error)
* def select\_company(connection, nombre):
* try:
* cursor = connection.cursor()
* select\_query = '''SELECT nombre, direccion, telefono, ingresos\_anuales, egresos\_anuales
* FROM empresas
* WHERE nombre = %s;'''
* data = (nombre,)
* cursor.execute(select\_query, data)
* results = cursor.fetchall()
* cursor.close()
* if results:
* print("Empresa seleccionada:")
* for row in results:
* print(f"Nombre: {row[0]}, Dirección: {row[1]}, Teléfono: {row[2]}, Ingresos anuales: {row[3]}, Egresos anuales: {row[4]}")
* else:
* print("La empresa no existe.")
* except (Exception, psycopg2.Error) as error:
* print("Error al seleccionar empresa:", error)
* def view\_results(connection, nombre):
* try:
* cursor = connection.cursor()
* select\_query = '''SELECT nombre, ingresos\_anuales - egresos\_anuales AS utilidad
* FROM empresas
* WHERE nombre = %s;'''
* data = (nombre,)
* cursor.execute(select\_query, data)
* results = cursor.fetchall()
* cursor.close()
* if results:
* print("Resultados:")
* for row in results:
* print(f"Empresa: {row[0]}, Utilidad: {row[1]}")
* else:
* print("La empresa no existe.")
* return results
* except (Exception, psycopg2.Error) as error:
* print("Error al ver resultados:", error)
* def close\_connection(connection):
* try:
* connection.close()
* return None
* except (Exception, psycopg2.Error) as error:
* print("Error al cerrar la conexión:", error)
* return None
* def main():
* connection = connect\_to\_database()
* if connection:
* while True:
* print("\nOpciones:")
* print("1. Agregar empresa")
* print("2. Seleccionar empresa")
* print("2.1. Ver resultados")
* print("3. Salir")
* opcion = input("Selecciona una opción: ")
* if opcion == "1":
* nombre = input("Nombre de la empresa: ")
* direccion = input("Dirección de la empresa: ")
* telefono = input("Teléfono de la empresa: ")
* ingresos\_anuales = float(input("Ingresos anuales: "))
* egresos\_anuales = float(input("Egresos anuales: "))
* add\_company(connection, nombre, direccion, telefono, ingresos\_anuales, egresos\_anuales)
* elif opcion == "2":
* nombre = input("Nombre de la empresa: ")
* select\_company(connection, nombre)
* elif opcion == "2.1":
* nombre = input("Nombre de la empresa: ")
* results = view\_results(connection, nombre)
* for row in results:
* print(f"Empresa: {row[0]}, Utilidad: {row[1]}")
* elif opcion == "3":
* print("Saliendo del programa.")
* connection.close()
* break
* else:
* print("Opción no válida. Por favor, selecciona una opción válida.")
* else:
* print("Error al conectar a la base de datos.")
* if \_\_name\_\_ == "\_\_main\_\_":
* main()
* Noveno Programa
* import psycopg2
* def connect\_to\_database():
* try:
* connection = psycopg2.connect(
* user="postgres",
* password="2405",
* host="localhost",
* port="5433",
* database="0980 Proyectos"
* )
* return connection
* except (Exception, psycopg2.Error) as error:
* print("Error al conectar a la base de datos:", error)
* return None
* def add\_product(connection, producto, cantidad, precio):
* try:
* cursor = connection.cursor()
* insert\_query = '''INSERT INTO inventario (producto, cantidad, precio)
* VALUES (%s, %s, %s);'''
* data = (producto, cantidad, precio)
* cursor.execute(insert\_query, data)
* connection.commit()
* cursor.close()
* print("Producto agregado con éxito.")
* except (Exception, psycopg2.Error) as error:
* print("Error al agregar producto:", error)
* def update\_product\_quantity(connection, producto, cantidad):
* try:
* cursor = connection.cursor()
* update\_query = '''UPDATE inventario
* SET cantidad = %s
* WHERE producto = %s;'''
* data = (cantidad, producto)
* cursor.execute(update\_query, data)
* connection.commit()
* cursor.close()
* print("Cantidad de producto actualizada con éxito.")
* except (Exception, psycopg2.Error) as error:
* print("Error al actualizar cantidad de producto:", error)
* def delete\_product(connection, producto):
* try:
* cursor = connection.cursor()
* delete\_query = '''DELETE FROM inventario
* WHERE producto = %s;'''
* data = (producto,)
* cursor.execute(delete\_query, data)
* connection.commit()
* cursor.close()
* print("Producto eliminado con éxito.")
* except (Exception, psycopg2.Error) as error:
* print("Error al eliminar producto:", error)
* def generate\_sales\_report(connection):
* try:
* cursor = connection.cursor()
* report\_query = '''SELECT  producto, cantidad, precio
* FROM inventario
* ORDER BY fecha DESC;'''
* cursor.execute(report\_query)
* results = cursor.fetchall()
* cursor.close()
* print("Reporte de ventas:")
* for row in results:
* print(f" Producto: {row[1]}, Cantidad: {row[2]}, Precio: {row[3]}")
* except (Exception, psycopg2.Error) as error:
* print("Error al generar reporte:", error)
* def main():
* connection = connect\_to\_database()
* if connection:
* while True:
* print("\nOpciones:")
* print("1. Agregar producto")
* print("2. Actualizar cantidad de producto")
* print("3. Eliminar producto")
* print("4. Generar informe de ventas")
* print("5. Salir")
* opcion = input("Selecciona una opción: ")
* if opcion == "1":
* producto = input("Nombre del producto: ")
* cantidad = int(input("Cantidad: "))
* precio = float(input("Precio: "))
* add\_product(connection, producto, cantidad, precio)
* elif opcion == "2":
* producto = input("Nombre del producto: ")
* cantidad = int(input("Cantidad nueva: "))
* update\_product\_quantity(connection, producto, cantidad)
* elif opcion == "3":
* producto = input("Nombre del producto: ")
* delete\_product(connection, producto)
* elif opcion == "4":
* generate\_sales\_report(connection)
* elif opcion == "5":
* print("Saliendo del programa.")
* break
* else:
* print("Opción no válida. Por favor, selecciona una opción válida.")
* connection.close()
* if \_\_name\_\_ == "\_\_main\_\_":
* main()
* Décimo Programa
* Onceavo Programa

import psycopg2

def connect\_to\_database():

    try:

        connection = psycopg2.connect(

            user="postgres",

            password="2405",

            host="localhost",

            port="5433",

            database="0980 Proyectos"

        )

        return connection

    except (Exception, psycopg2.Error) as error:

        print("Error al conectar a la base de datos:", error)

        return None

def show\_all\_songs(connection):

    try:

        cursor = connection.cursor()

        query = "SELECT \* FROM canciones"

        cursor.execute(query)

        results = cursor.fetchall()

        cursor.close()

        print("Listado de canciones:")

        for row in results:

            print(f"Artista: {row[0]}, Canción: {row[1]}")

    except (Exception, psycopg2.Error) as error:

        print("Error al obtener canciones:", error)

def search\_by\_artist(connection, artist):

    try:

        cursor = connection.cursor()

        query = f"SELECT \* FROM canciones WHERE artista = '{artist}'"

        cursor.execute(query)

        results = cursor.fetchall()

        cursor.close()

        if results:

            print(f"Canciones de {artist}:")

            for row in results:

                print(f"Canción: {row[1]}")

        else:

            print(f"No se encontraron canciones de {artist}")

    except (Exception, psycopg2.Error) as error:

        print("Error al buscar canciones por artista:", error)

def search\_by\_song(connection, song):

    try:

        cursor = connection.cursor()

        query = f"SELECT \* FROM canciones WHERE cancion = '{song}'"

        cursor.execute(query)

        results = cursor.fetchall()

        cursor.close()

        if results:

            print(f"Canción {song}:")

            for row in results:

                print(f"Artista: {row[0]}")

        else:

            print(f"No se encontró la canción {song}")

    except (Exception, psycopg2.Error) as error:

        print("Error al buscar canciones por canción:", error)

def main():

    connection = connect\_to\_database()

    if connection:

        while True:

            print("\nOpciones:")

            print("1. Desplegar el listado de canciones")

            print("2. Buscar por artista")

            print("3. Buscar por canción")

            print("4. Salir")

            opcion = input("Selecciona una opción: ")

            if opcion == "1":

                show\_all\_songs(connection)

            elif opcion == "2":

                artista = input("Introduce el nombre del artista: ")

                search\_by\_artist(connection, artista)

            elif opcion == "3":

                cancion = input("Introduce el título de la canción: ")

                search\_by\_song(connection, cancion)

            elif opcion == "4":

                print("Saliendo del programa.")

                break

            else:

                print("Opción no válida. Por favor, selecciona una opción válida.")

        connection.close()

if \_\_name\_\_ == "\_\_main\_\_":

    main()

* Doceavo Programa
* import psycopg2
* def connect\_to\_database():
* try:
* connection = psycopg2.connect(
* user="postgres",
* password="2405",
* host="localhost",
* port="5433",
* database="0980 Proyectos"
* )
* return connection
* except (Exception, psycopg2.Error) as error:
* print("Error al conectar a la base de datos:", error)
* return None
* def show\_all\_questions(connection):
* try:
* cursor = connection.cursor()
* query = "SELECT pregunta, respuesta FROM futbol"
* cursor.execute(query)
* results = cursor.fetchall()
* cursor.close()
* print("Listado de preguntas:")
* for row in results:
* print(f"Pregunta: {row[0]}, Respuesta: {row[1]}")
* except (Exception, psycopg2.Error) as error:
* print("Error al obtener preguntas:", error)
* def generate\_random\_question(connection):
* try:
* cursor = connection.cursor()
* query = "SELECT pregunta, respuesta FROM futbol ORDER BY random() LIMIT 1"
* cursor.execute(query)
* results = cursor.fetchall()
* cursor.close()
* if results:
* return results[0]
* else:
* return None
* except (Exception, psycopg2.Error) as error:
* print("Error al generar pregunta aleatoria:", error)
* return None
* def play\_game(connection):
* lives = 3
* points = 0
* while lives > 0:
* question = generate\_random\_question(connection)
* print(question[0])
* answer = input("Tu respuesta: ")
* if answer == question[1]:
* points += 1
* print("¡Correcto! Ganaste 1 punto.")
* else:
* lives -= 1
* print("¡Incorrecto! Perdiste una vida.")
* if lives == 0:
* print("Has perdido el juego.")
* break
* print("Tu puntuación actual es:", points)
* print("Has ganado", points, "puntos.")
* def main():
* connection = connect\_to\_database()
* if connection:
* while True:
* print("\nOpciones:")
* print("1. Jugar")
* print("2. Ver preguntas")
* print("3. Salir")
* opcion = input("Selecciona una opción: ")
* if opcion == "1":
* play\_game(connection)
* elif opcion == "2":
* show\_all\_questions(connection)
* elif opcion == "3":
* print("Saliendo del programa.")
* break
* else:
* print("Opción no válida. Por favor, selecciona una opción válida.")
* connection.close()
* if \_\_name\_\_ == "\_\_main\_\_":
* main()
* Cuarta Serie

if(exist('OCTAVE\_VERSION','builtin')~=0)

pkg load signal; end

%MENU

opcion = 0;

while opcion ~=5 disp('elija una opcion') disp('1.Grabacion') disp('2.Reproduccion') disp('3.Graficar') disp('4.Graficar densidad')

disp('5. Salir')

opcion = input('Ingrese la opcion:'); switch opcion

case 1 try

duracion = input('Ingrese la duración en segundos:'); disp('Iniciando la grabación');

recObj = audiorecorder; recordblocking(recObj, duracion); disp('Grabacion terminada'); data= getaudiodata(recObj);

audiowrite('audio.wav', data, recObj.SampleRate); disp('Archivo de audio grabado ');

catch

disp('Error al grabar audio'); end\_try\_catch

case 2 try

[data, fs] = audioread('audio.wav'); sound(data, fs);

catch

disp('Error al reproducir el audio'); end\_try\_catch

case 3 try

[data, fs]=audioread('audio.wav');

tiempo = linspace(0, length(data)/fs, length(data)); plot(tiempo, data);

xlabel('Tiempo(s)');

ylabel('Amplitud'); title('Audio');

catch

disp('Error al graficar '); end\_try\_catch

case 4 try

disp('Graficando espectro de frecuencia'); [audio, Fs] = audioread('audio.wav');

L = length(audio);

r = linspace(0, Fs/2, L/2+1); ventana = hann(L);

Sxx= pwelch(audio, ventana, 0, L, Fs); plot(r, 10\*log10(Sxx(1:L/2+1))); xlabel('Frecuencia (Hz)');

ylabel('Densidad espectral de potencia(dB/Hz)'); title('Espectro de freuencia de la señal grabada'); catch

disp('Error al graficar audio'); end\_try\_catch

case 5

disp('Saliendo del programa'); break

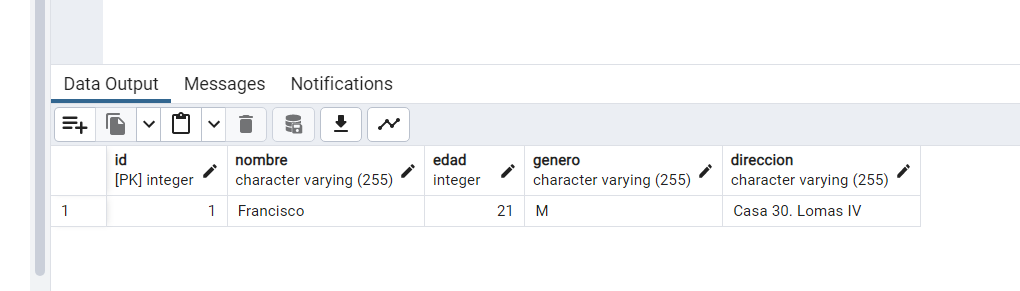
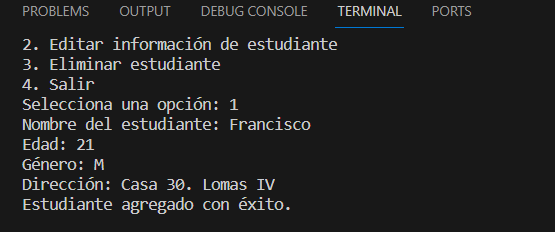
otherwise

disp('Opción inválida'); end

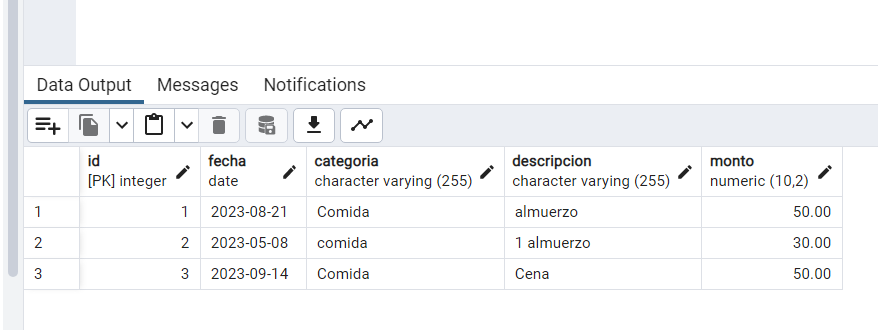
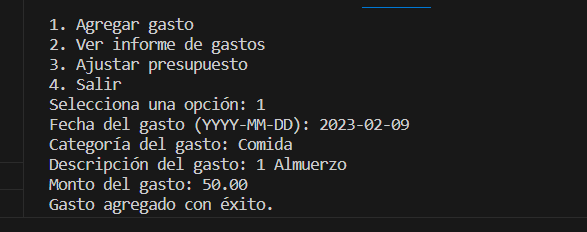
end

# Resultados

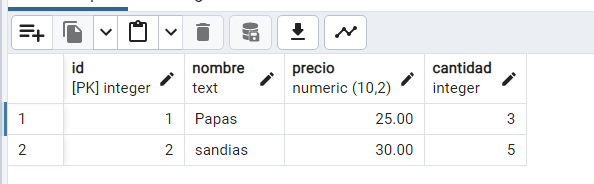
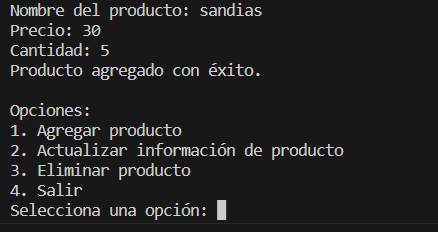
* Primer Programa



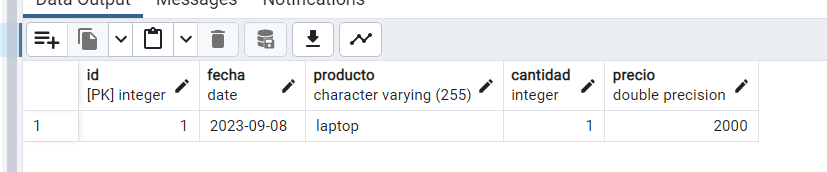
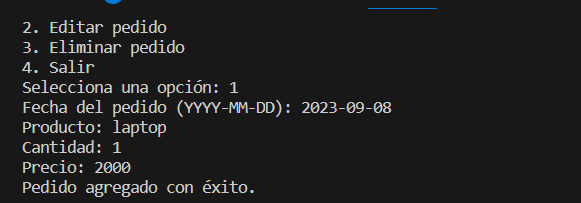
* Segundo Programa



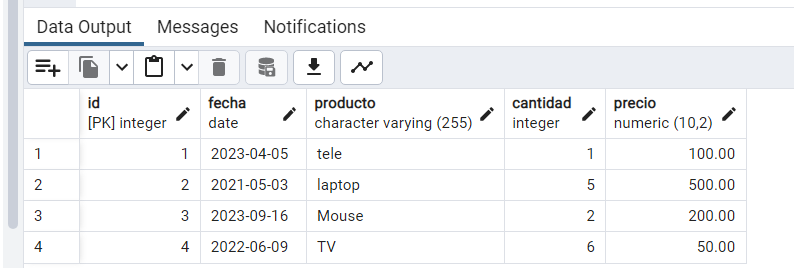
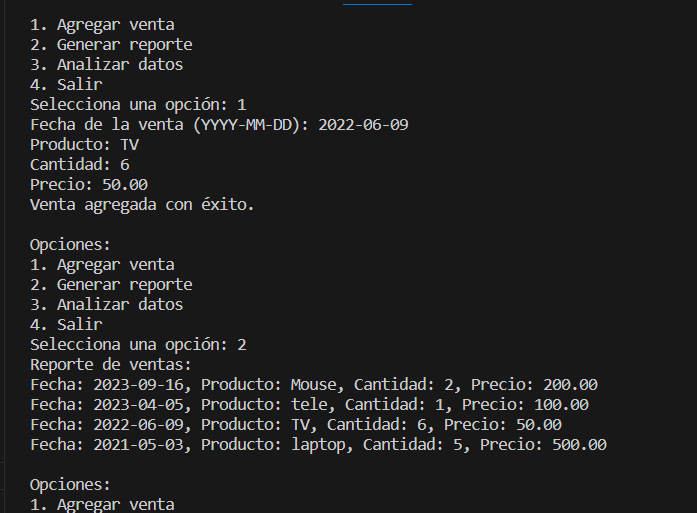
* Tercero Programa

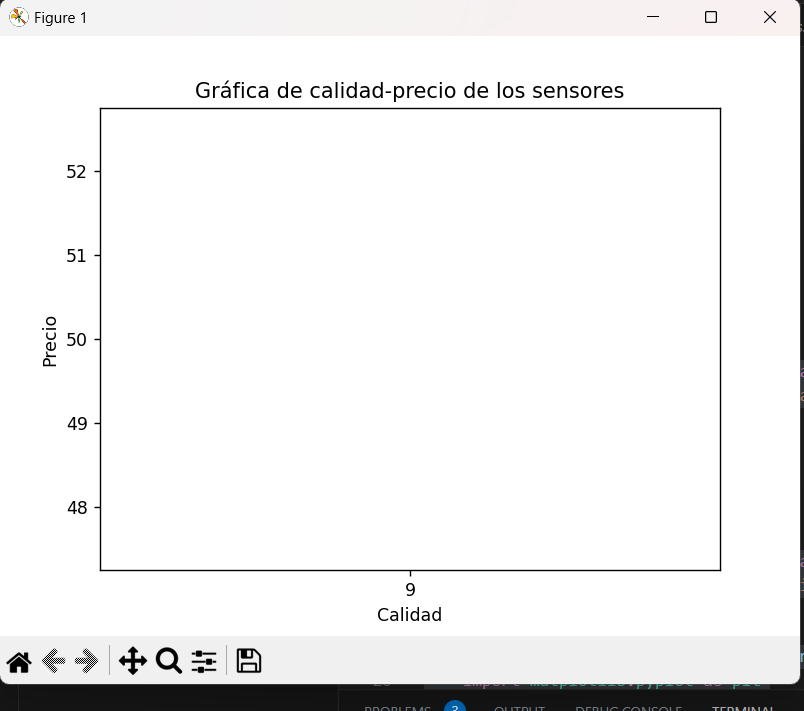
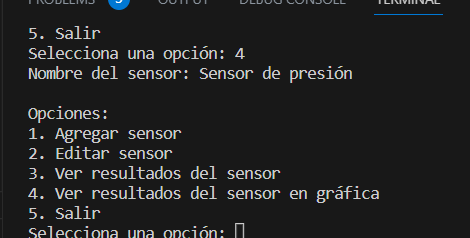
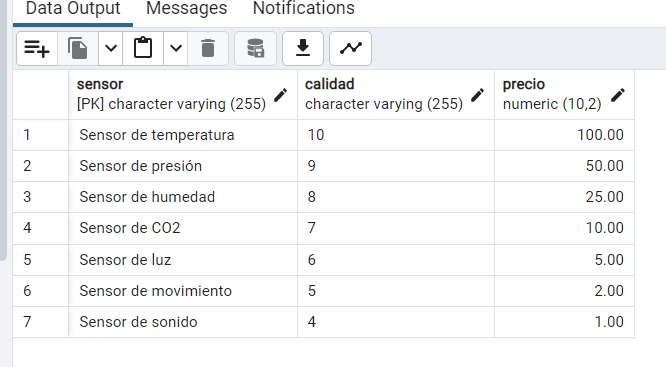


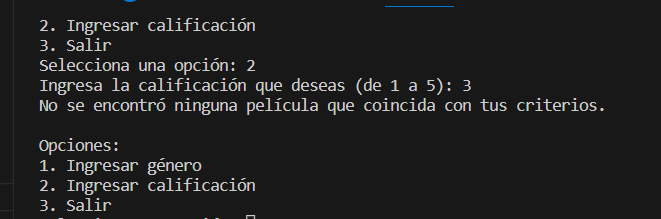
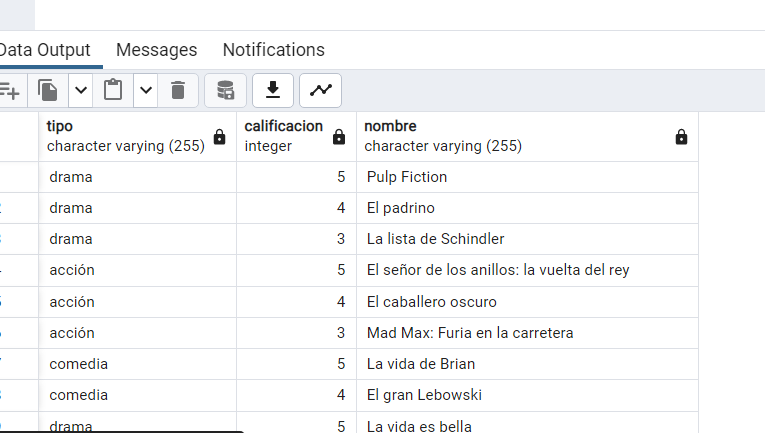
* Cuarto Programa

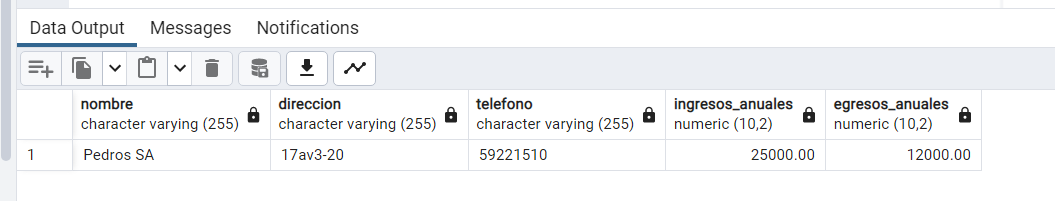
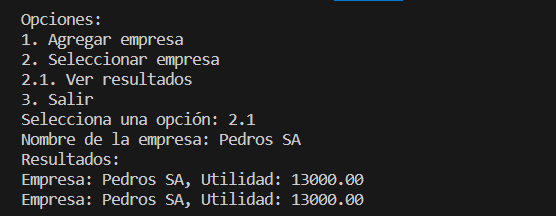


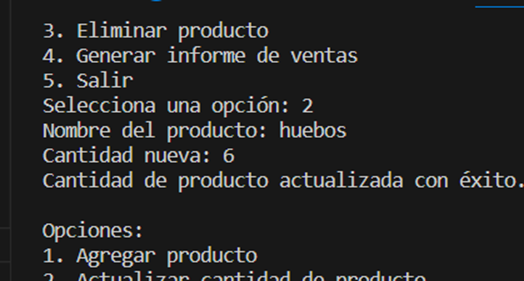
* Quinto Programa

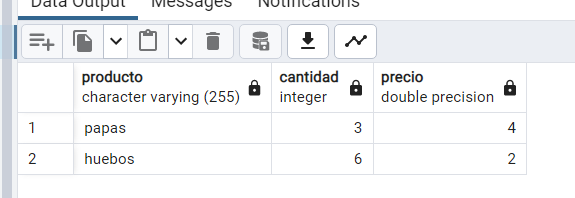


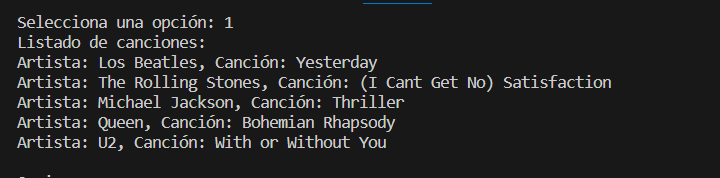
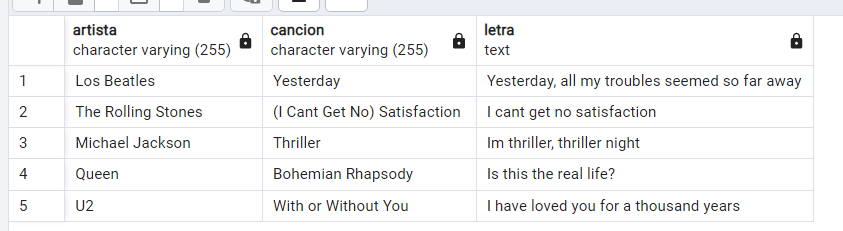
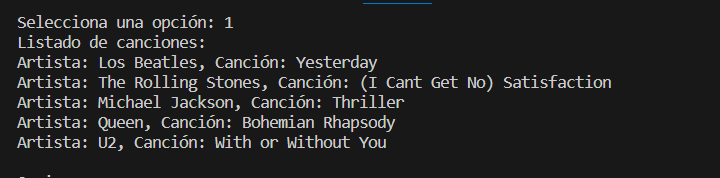
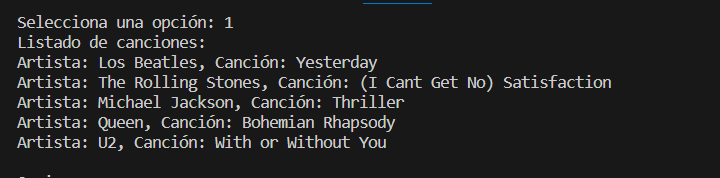
* Sexto Programa
* ****
* Séptimo Programa

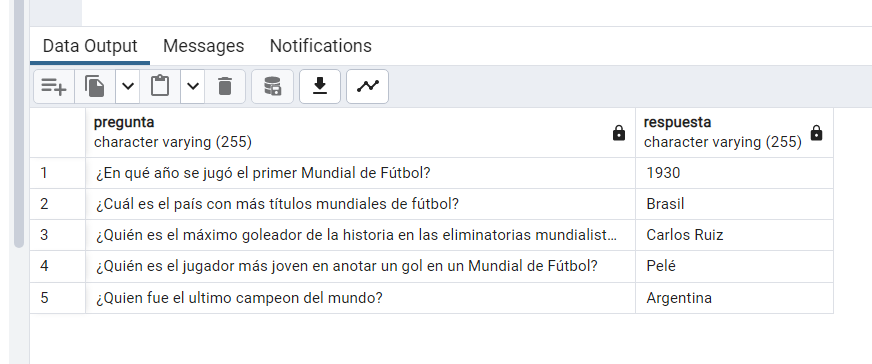
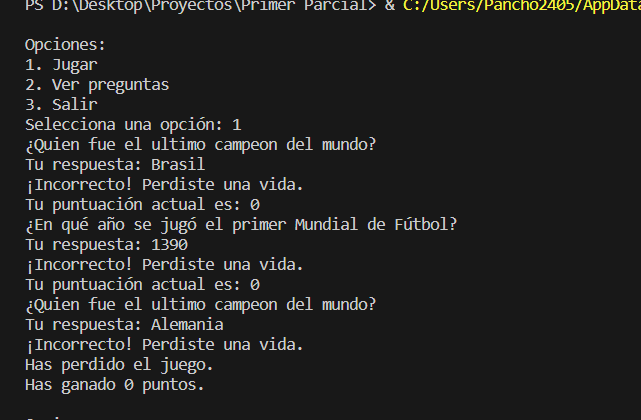
****

* Octavo Programa
* ****
* Noveno Programa

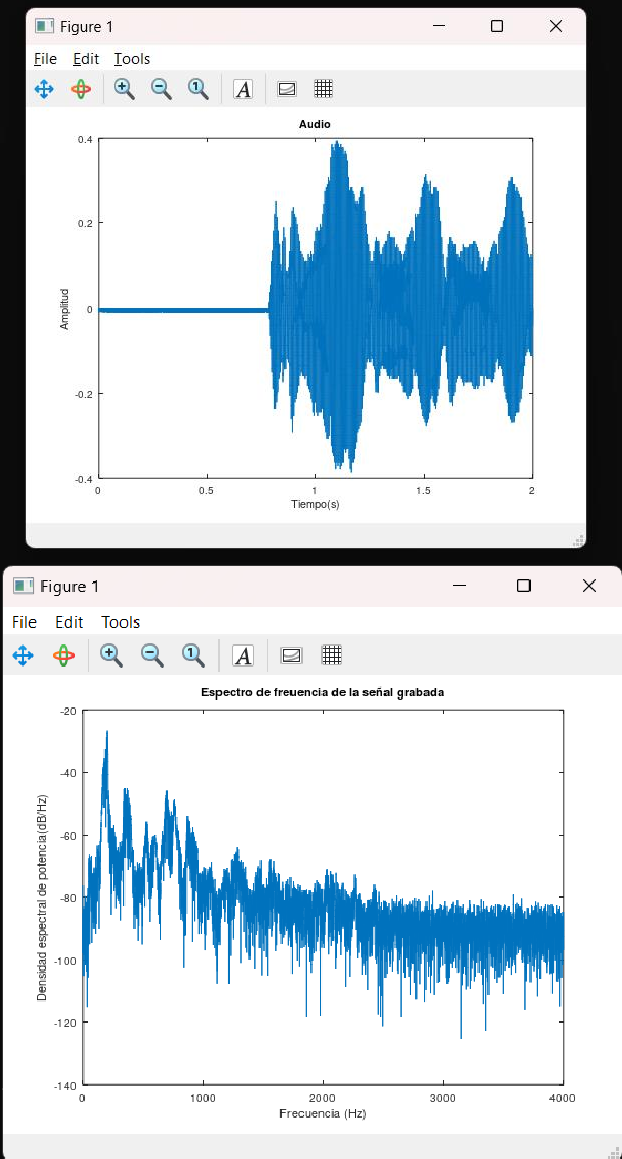


****

* Décimo Programa
* ****
* Onceavo Programa
* 
* 
* Doceavo Programa



SERIE IV



1. [↑](#footnote-ref-1)