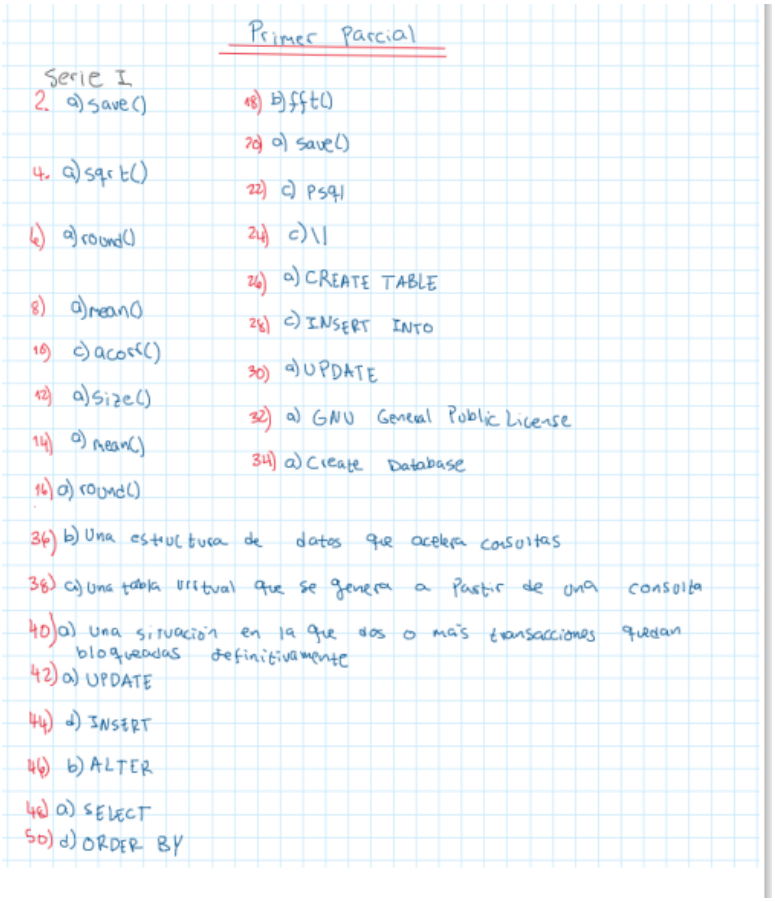
Primer Examen Parcial

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

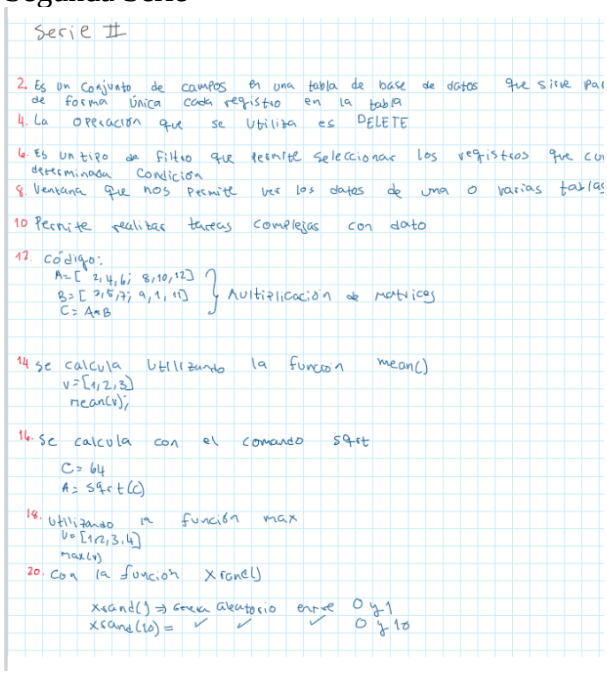


Primer Parcial

Serie I

2. a) save()
4. a) sqrt()
6) a) round()
8) a) mean()
10) c) acos()
12) a) size()
14) a) mean()
16) a) round()
18) b) fft()
20) a) save()
22) c) psql
24) c) \\
26) a) CREATE TABLE
28) c) INSERT INTO
30) a) UPDATE
32) a) GNU General Public License
34) a) Create Database
36) b) Una estructura de datos que acelera consultas
38) c) Una tabla virtual que se genera a partir de una consulta
40) a) Una situación en la que dos o más transacciones quedan bloqueadas definitivamente
42) a) UPDATE
44) d) INSERT
46) b) ALTER
48) a) SELECT
50) d) ORDER BY

- 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(connectio
n, 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(connectio
n, 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(connectio
n, sensor, calidad, precio)
            elif opcion == "3":
```

```
                sensor =
input("Nombre del sensor: ")
                view_sensor_results(c
onnection, sensor)
            elif opcion == "4":
                sensor =
input("Nombre del sensor: ")
                view_sensor_results_g
raph(connection, sensor)
            elif opcion == "5":
                print("Saliendo del
programa.")
                close_connection(conn
ection)
                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(conn
ection)
                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(connectio
n, producto, cantidad, precio)
            elif opcion == "2":
                producto =
input("Nombre del producto: ")
                cantidad =
int(input("Cantidad nueva: "))
                update_product_quanti
ty(connection, producto, cantidad)
            elif opcion == "3":
                producto =
input("Nombre del producto: ")
                delete_product(connec
tion, 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(co
    nnection)
                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('Ampl itud');
   title('Audio')
   ;
  catch
   disp('Error al graficar ');
  end_try_catch

 c
  a
  s
  e
  4
  t
  r
  y
   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

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

2. Editar información de estudiante
3. Eliminar estudiante
4. Salir
Selecciona una opción: 1
Nombre del estudiante: Francisco
Edad: 21
Género: M
Dirección: Casa 30. Lomas IV
Estudiante agregado con éxito.
```

Data Output | Messages | Notifications

	id [PK] integer	nombre character varying (255)	edad integer	genero character varying (255)	direccion character varying (255)
1	1	Francisco	21	M	Casa 30. Lomas IV

- Segundo Programa

```
1. Agregar gasto
2. Ver informe de gastos
3. Ajustar presupuesto
4. Salir
Selecciona una opción: 1
Fecha del gasto (YYYY-MM-DD): 2023-02-09
Categoría del gasto: Comida
Descripción del gasto: 1 Almuerzo
Monto del gasto: 50.00
Gasto agregado con éxito.
```

Data Output | Messages | Notifications

	id [PK] integer	fecha date	categoria character varying (255)	descripcion character varying (255)	monto numeric (
1	1	2023-08-21	Comida	almuerzo	
2	2	2023-05-08	comida	1 almuerzo	
3	3	2023-09-14	Comida	Cena	

- Tercero Programa

```
Nombre del producto: sandias
Precio: 30
Cantidad: 5
Producto agregado con éxito.

Opciones:
1. Agregar producto
2. Actualizar información de producto
3. Eliminar producto
4. Salir
Selecciona una opción:
```

	id [PK] integer	nombre text	precio numeric (10,2)	cantidad integer
1	1	Papas	25.00	3
2	2	sandias	30.00	5

- Cuarto Programa

```
2. Editar pedido
3. Eliminar pedido
4. Salir
Selecciona una opción: 1
Fecha del pedido (YYYY-MM-DD): 2023-09-08
Producto: laptop
Cantidad: 1
Precio: 2000
Pedido agregado con éxito.
```

	id [PK] integer	fecha date	producto character varying (255)	cantidad integer	precio double precision
1	1	2023-09-08	laptop	1	2000

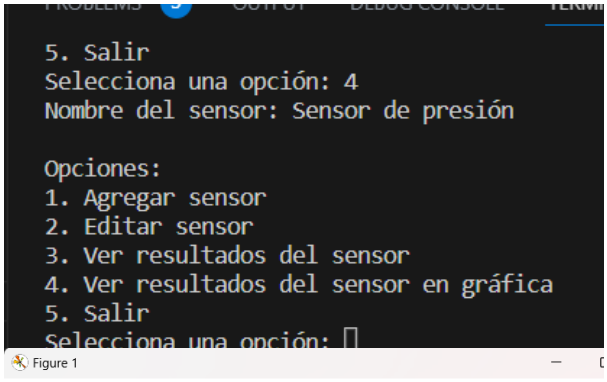
- Quinto Programa

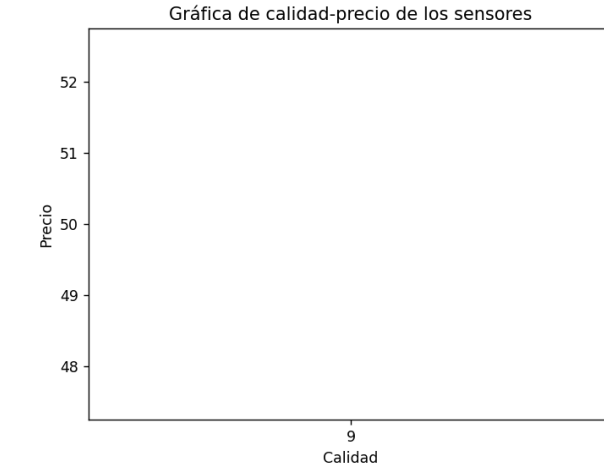
```
1. Agregar venta
2. Generar reporte
3. Analizar datos
4. Salir
Selecciona una opción: 1
Fecha de la venta (YYYY-MM-DD): 2022-06-09
Producto: TV
Cantidad: 6
Precio: 50.00
Venta agregada con éxito.

Opciones:
1. Agregar venta
2. Generar reporte
3. Analizar datos
4. Salir
Selecciona una opción: 2
Reporte de ventas:
Fecha: 2023-09-16, Producto: Mouse, Cantidad: 2, Precio: 200.00
Fecha: 2023-04-05, Producto: tele, Cantidad: 1, Precio: 100.00
Fecha: 2022-06-09, Producto: TV, Cantidad: 6, Precio: 50.00
Fecha: 2021-05-03, Producto: laptop, Cantidad: 5, Precio: 500.00

Opciones:
1. Agregar venta
```

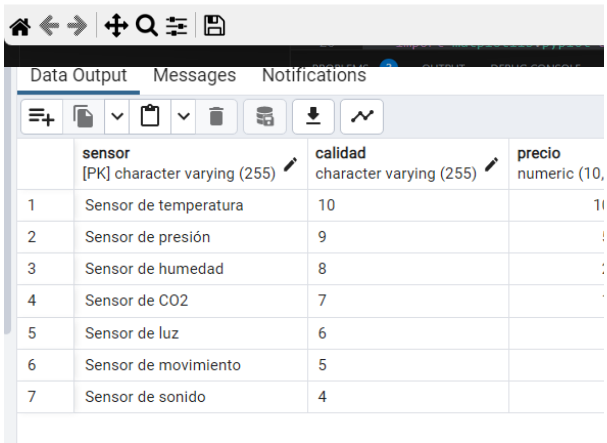
Data Output | Messages | Notifications

	id [PK] integer	fecha date	producto character varying (255)	cantidad integer	precio numeric (10,2)
1	1	2023-04-05	tele	1	100.00
2	2	2021-05-03	laptop	5	500.00
3	3	2023-09-16	Mouse	2	200.00
4	4	2022-06-09	TV	6	50.00

- Sexto Programa

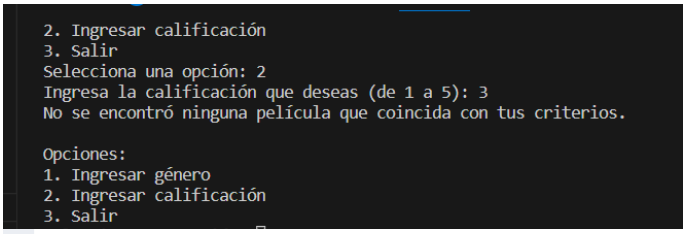
- 

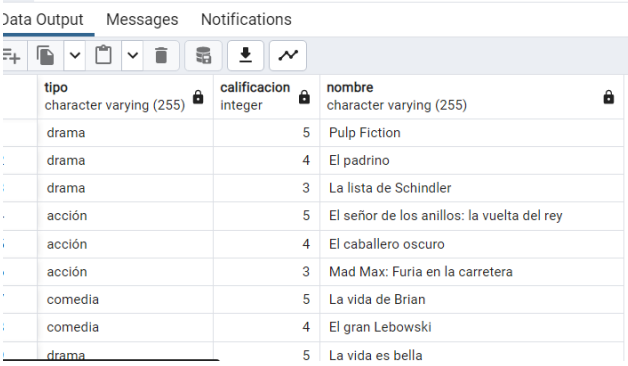




	sensor [PK] character varying (255)	calidad character varying (255)	precio numeric (10,
1	Sensor de temperatura	10	1(
2	Sensor de presión	9	
3	Sensor de humedad	8	
4	Sensor de CO2	7	
5	Sensor de luz	6	
6	Sensor de movimiento	5	
7	Sensor de sonido	4	

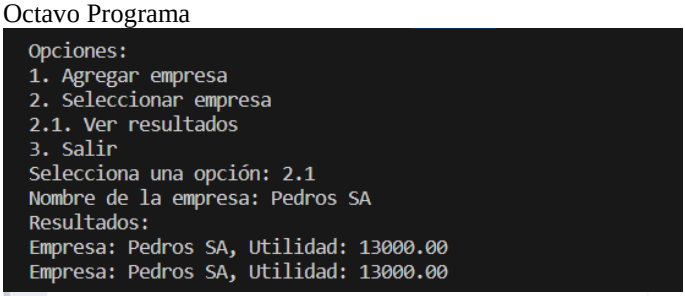
- Séptimo Programa

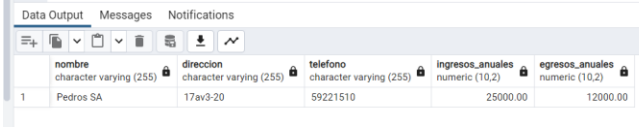
- 



	tipo character varying (255)	calificacion integer	nombre character varying (255)
	drama	5	Pulp Fiction
	drama	4	El padrino
	drama	3	La lista de Schindler
	acción	5	El señor de los anillos: la vuelta del rey
	acción	4	El caballero oscuro
	acción	3	Mad Max: Furia en la carretera
	comedia	5	La vida de Brian
	comedia	4	El gran Lebowski
	drama	5	La vida es bella

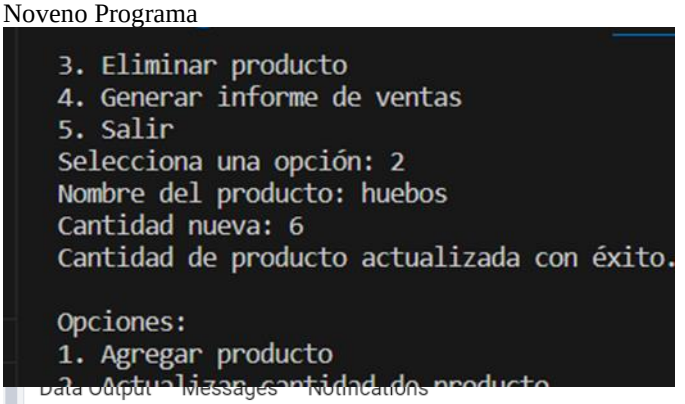
- Octavo Programa

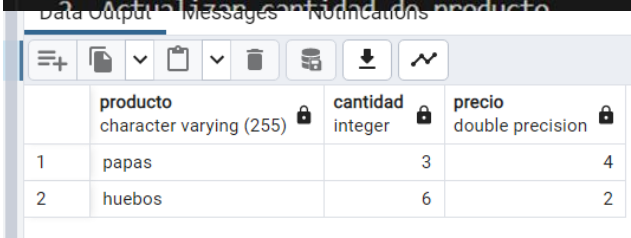
- 



	nombre character varying (255)	direccion character varying (255)	telefono character varying (255)	ingresos_anuales numeric (10,2)	egresos_anuales numeric (10,2)
1	Pedros SA	17av3-20	59221510	25000.00	12000.00

- Noveno Programa

- 



	producto character varying (255)	cantidad integer	precio double precision
1	papas	3	4
2	huebos	6	2

- Décimo Programa

```
Selecciona una opción: 1
Listado de canciones:
Artista: Los Beatles, Canción: Yesterday
Artista: The Rolling Stones, Canción: (I Cant Get No) Satisfaction
Artista: Michael Jackson, Canción: Thriller
Artista: Queen, Canción: Bohemian Rhapsody
Artista: U2, Canción: With or Without You
```

	artista character varying (255)	cancion character varying (255)	letra text
1	Los Beatles	Yesterday	Yesterday, all my troubles seemed so
2	The Rolling Stones	(I Cant Get No) Satisfaction	I cant get no satisfaction
3	Michael Jackson	Thriller	Im thriller, thriller night
4	Queen	Bohemian Rhapsody	Is this the real life?
5	U2	With or Without You	I have loved you for a thousand years

- Onceavo Programa

```
Selecciona una opción: 1
Listado de canciones:
Artista: Los Beatles, Canción: Yesterday
Artista: The Rolling Stones, Canción: (I Cant Get No) Satisfaction
Artista: Michael Jackson, Canción: Thriller
Artista: Queen, Canción: Bohemian Rhapsody
Artista: U2, Canción: With or Without You
```

```
Selecciona una opción: 1
Listado de canciones:
Artista: Los Beatles, Canción: Yesterday
Artista: The Rolling Stones, Canción: (I Cant Get No) Satisfaction
Artista: Michael Jackson, Canción: Thriller
Artista: Queen, Canción: Bohemian Rhapsody
Artista: U2, Canción: With or Without You
```

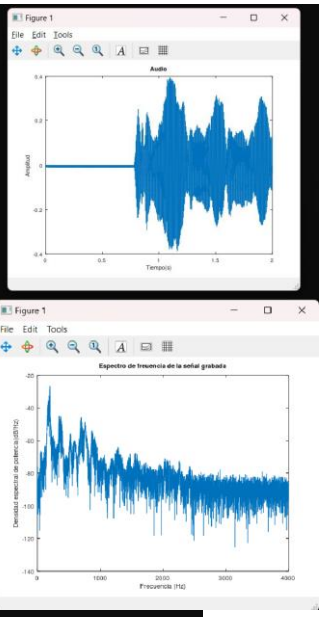
- Doceavo Programa

```
PS D:\Desktop\Proyectos\PrimerParcial> & C:/Users/Pancho2405/AppDa

Opciones:
1. Jugar
2. Ver preguntas
3. Salir
Selecciona una opción: 1
¿Quien fue el ultimo campeon del mundo?
Tu respuesta: Brasil
¡Incorrecto! Perdiste una vida.
Tu puntuación actual es: 0
¿En qué año se jugó el primer Mundial de Fútbol?
Tu respuesta: 1390
¡Incorrecto! Perdiste una vida.
Tu puntuación actual es: 0
¿Quien fue el ultimo campeon del mundo?
Tu respuesta: Alemania
¡Incorrecto! Perdiste una vida.
Has perdido el juego.
Has ganado 0 puntos.
```

Data Output | Messages | Notifications

	pregunta character varying (255)	respuesta character varying (255)
1	¿En qué año se jugó el primer Mundial de Fútbol?	1930
2	¿Cuál es el país con más títulos mundiales de fútbol?	Brasil
3	¿Quién es el máximo goleador de la historia en las eliminatorias mundialist...	Carlos Ruiz
4	¿Quién es el jugador más joven en anotar un gol en un Mundial de Fútbol?	Pelé
5	¿Quien fue el ultimo campeon del mundo?	Argentina

```
Seleccione una opcion
1.Grabar
2.Reproducir
3.Graficar
4.Graficar densidad
5. Salir
Ingrese su elección:1
Ingrese la duración de la grabación en segur
Comenzando la grabación
Grabacion finalizada
Archivo de audio grabado correctamente
Seleccione una opcion
1.Grabar
2.Reproducir
3.Graficar
4.Graficar densidad
5. Salir
Ingrese su elección:4
Graficando espectro de frecuencia
Seleccione una opcion
1.Grabar
2.Reproducir
3.Graficar
4.Graficar densidad
5. Salir
Ingrese su elección:3
Seleccione una opcion
1.Grabar
2.Reproducir
3.Graficar
4.Graficar densidad
5. Salir
Ingrese su elección:2
Seleccione una opcion
1.Grabar
2.Reproducir
3.Graficar
4.Graficar densidad
5. Salir
Ingrese su elección:
```



SERIE IV