Informe de proyecto [Databases Oracle Delincuencia]

dcayo@espe.edu.ec-jetoapanta10@espe.edu.ec

Universidad de las Fuerzas Armadas
dcayo@espe.edu.ec-jetoapanta10@espe.edu.ec

Tema: Databases Oracle Delincuencia y Consultas

Abstract

Para el actual proyecto se creará una base de datos de como es un proceso que tiene una delincuente al momento que delinque a una persona civil utilizando una base de datos popular en la actualidad que es llamada Oracle Xpress con una interfaz gráfica de la misma compañía denominada Oracle Developer. Continuamente se representará los diagramas en sus 3 formas utilizando Gliffy que es un software especializado en realizar diagramas entidad relación. Y por último punto se insertará datos sintéticos realizados en el lenguaje de programación Python a la base de datos que será la cantidad de 5K para poder realizar consultas específicas de nuestra databases.

1 Introducción

Las tecnologías crecen a pasos agigantados día a día y son una de las áreas más populares y en constante crecimiento en la actualidad y el mundo laboral. Estas tecnologías incluyen una variedad de áreas como el desarrollo de software, las bases de datos relacionales e irracionales, la seguridad informática, la inteligencia artificial, la nube y el análisis de datos, entre otras. Por esta razón, el objetivo principal de este proyecto de la unidad del segundo parcial es enfocarnos en el tema de base de datos relacionales para tener el conocimiento adecuado y correcto para no tener problemas en un futuro en el mundo laboral. Para este presente trabajo se ha decidido trabajar en Oracle Developer donde se creará tablas con la sintaxis que pertenece a dicha plataforma y se llenará con datos sintéticos a través del lenguaje de Python, recordando que es uno de los lenguajes más populares en el mundo de la programación. Continuamente se realizará consultas o también conocidas como querys para recuperar, extraer o seleccionar información específica de nuestra base de datos.

Oracle es una plataforma o herramienta de base de datos relacionales desarrollada por Oracle Corporación [1]. Esta base de datos relacionales es una de las más conocidas y utilizadas en el mundo empresarial y financiero, también ofrece una excelente compatibilidad con varios sistemas operativos, especialmente en grandes sistemas de información. Oracle ofrece gran escalabilidad y rendimiento, y puede manejar grandes volúmenes de datos. Además, existen diversas herramientas de gestión y desarrollo que facilitan el trabajo con bases de datos. Oracle tiene su uso en entornos donde se manejen inmensas cantidades de datos.

Contexto y motivación El databases relacional que se creará en Oracle tendrá el nombre de delincuencia debido a que el motivo del proyecto está enfocado a la delincuencia de Ecuador [2].

Lastimosamente en los últimos años nuestro país presenta unas grandes cantidades de porcentaje de delincuencia que va aumentando cada día más. En este proyecto se recolectará y almacenará información sobre los diferentes tipos de delitos ocurridos en Ecuador, así como información relacionada, como la coordenada, el nombre de la víctima, o al grupo que pertenece el delincuente, entre otros, con el objetivo de analizar y entender mejor la delincuencia en el país. Además, se buscará identificar patrones y tendencias en la delincuencia, lo que permitirá colaborar con información para brindar a las autoridades para que puedan tomar medidas preventivas y diseñar estrategias para reducir la delincuencia en el futuro.

Problema de estudio Crear un databases con datos sintéticos 5K correctas mediante un análisis adecuado para poder obtener buenos resultados con la simulación de registros que se acerquen lo más real que pueda a este fenómeno social llamado delincuencia. Se realizará consultas donde podremos observar en que se puede aportar para detener este fenómeno que invade nuestro paìs. El enfoque de este proyecto está en la provincia de Pichincha ciudad Quito en el barrio de San Diego debido a que es una de las zonas más peligrosas que tiene Quito.

Trabajo relacionado Para esta base de datos se trabajo con con estudios realizados de delincuencia .El primer estudio es realizado por la Universidad Central del Ecuador donde el tema general es el barrio de San Diego [3] donde nos indica varios puntos importantes como la historia, el comercio, el transporte público y mapas del sector con puntos estratégicos. Para la realización de este estudio se escogió el punto de la delincuencia realizado en dicho estudio donde nos indica puntos claves donde se ejerce este fenómeno social con los nombres de las calles y las zonas más peligrosas del sector. El segundo documento para la realización de este proyecto fueron las leyes de Seguridad Pública y Estado realizadas por la Asamblea Nacional del Ecuador en el año 2009 con estado de vigentes hasta la actualidad. Se enfocará en artículos donde nos permita estudiar este fenómeno social y combatir de una forma segura este problema que afecta a la mayoría de población ecuatoriana.Por ultimo, este trabajo esta basado en un estudio general del sector centro de Quito espcificamente el barrio de San Diego y sus aledaños. El cual, fue realizados por la Universidad Central del Ecuador la facultad de Arquitectura y Urbanismo.[2]

2 Antecedentes

Para poder realizar este proyecto de la segunda unidad se comenzó analizando varios estudios que nos indicaba que la delincuencia era un fenómeno que cada vez crecía más con el pasar de los días. Por eso se comenzó con los conocimientos repartidos de parte de nuestra ingeniera partiendo de un diagrama entidad relación, identificando que relaciones y entidades pueden participar en nuestro diagrama[2]. Existieron entidades que al momento de analizar la cardinalidad se convertían en entidades. Seguidamente, se puso en práctica en crear una base de datos con sus respectivas tablas, atributos y llave primarias y secundarias en SQLite, llenándolas con 10 datos cada una de ellas para realizar consultas básicas. Al momento de crear la database se tenía la opción de poder reflejar en un software DbSchema donde al momento de cargar la bd nos mostraba el diagrama entidad-relación. Para este periodo se comenzó estudiando otra distinta base de datos llamada Oracle, que es uno de los más populares a nivel global, creando una bd llamada delincuencia con sus respectivas tablas y datos sintéticos para poder realizar consultas de nuestra base de datos. Para todo este proceso se tuvo que aplicar normalización para que no exista redundancia o datos basura en nuestra base de datos.[4]

3 Método

Para crear nuestra base de datos se trabajó en una base diferente en la que se trabajó en los anteriores laboratorios, esta vez se escogió Oracle Express Edición debido a que nos facilita con la versión gratuita. Esta edición es principalmente para personas como estudiantes que recién estamos adaptándonos o estudiando el tema de gestión de base de datos o para pequeñas empresas que necesitad realizar bases de datos no tan complejas. Para poder acceder a los paquetes de Oracle es de suma importancia registrarse con un correo institucional y tener una clave estable que al usuario no se le olvide y al momento de acceder nuevamente salga claves incorrectas debido a que es muy complejo poder recuperar dicha clave. También se instaló Oracle Developer que es una interfaz gráfica igualmente de forma gratuita que permite a las personas realizar base de datos con menos complejidad. Developer tiene su función principal es ayudar a los desarrolladores de base de datos a interactuar con Oracle de una formas más sencilla y amigable.

3.1 Diseño

Para el diseño se realiza tres formas de normalización la primera que corresponde a identificar si existen entidades juntas que no son propios de cierta entidad y lo representamos en un diagrama entidad relación . Para la segunda forma normal tenemos una tabla mucho más formal donde constan de igual forma entidades y atributos solo que en modo de columnas y presentando ya una cardinalidad. Para la tercera forma se realiza la unión de las entidades con respectiva cardinalidad entre relaciones con sus respectivas llaves primarias y foráneas.

3.2 Diagramas

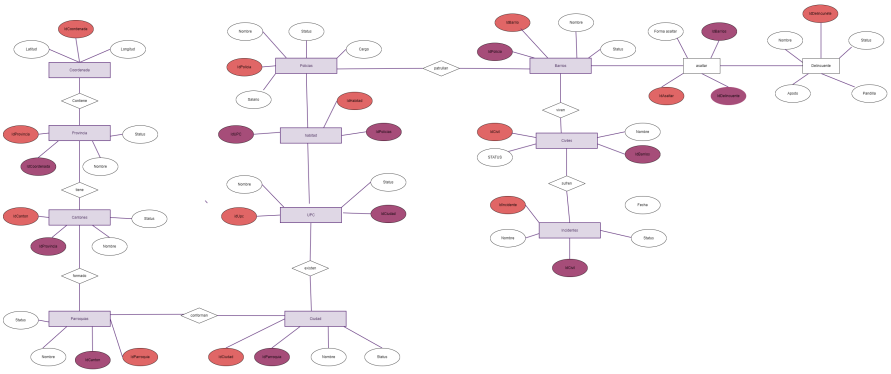


Figure 1: Diagrama Entidad Relaciòn Primera Forma Normal.

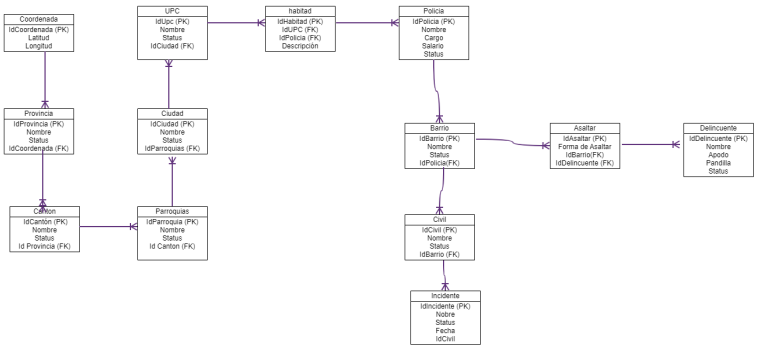


Figure 2: Diagrama Entidad Relaciòn Segunda y Tercera forma.

3.3 Base de Datos ORACLE Developer

Una vez realizado nuestro diagrama entidad relación de las 3 formas seguimos a la creación de la base de datos en el software SQLdeveloper. Abrimos y creamos el nombre de nuestra base de datos con el nombre de delincuencia y seguidamente colocaremos la clave que colocamos al momento de registrarse en Oracle. Seguidamente creamos las tablas de nuestro diagrama que son 13 entidades con sus respectivos atributos. Al momento de la creación deber tener en cuenta la sintaxis que utiliza developer que es similar a Sqlite algo que debemos tener en cuenta es como las llaves primarias van heredando a cada entidad. Podremos observar en una captura de txt el código de la creación de las tablas del proyecto llamado Delincuencia.

*oracle: Bloc de notas

Archivo Editar Ver

```
CREATE TABLE Coordenada(

coor_IdCoordenada    NUMBER PRIMARY KEY ,
coor_Latitud         VARCHAR2(50) NOT NULL  ,
coor_Longitud        VARCHAR2(50) NOT NULL,
coor_Status          VARCHAR2(50) NOT NULL
);
CREATE TABLE Provincia (
pro_IdProvincia      NUMBER PRIMARY KEY ,
pro_Nombre           VARCHAR2(50) NOT NULL,
pro_Status           VARCHAR2(50) NOT NULL,
pro_IdCoordenada INTEGER REFERENCES Coordenada(coor_IdCoordenada)
);
CREATE TABLE Canton(
can_IdCanton         NUMBER PRIMARY KEY ,
can_Nombre           VARCHAR2(50) NOT NULL,
can_Status           VARCHAR(250) NOT NULL,
can_IdProvincia INTEGER REFERENCES Provincia(pro_IdProvincia)
 );
CREATE TABLE Parroquia(
parro_IdParroquia    NUMBER PRIMARY KEY ,
parro_Nombre         VARCHAR2(50) NOT NULL,
parro_Status         VARCHAR2(50) NOT NULL,
parro_IdCanton INTEGER REFERENCES Canton(can_IdCanton)
 );
CREATE TABLE Ciudad(
ciu_IdCiudad       NUMBER PRIMARY KEY ,
ciu_Nombre     VARCHAR2(50) NOT NULL,
ciu_Status     VARCHAR2(50) NOT NULL,
ciu_IdParroquia INTEGER REFERENCES Parroquia(parro_IdParroquia)
 );
CREATE TABLE Upc(
upc_IdUpc       NUMBER PRIMARY KEY ,
upc_Nombre     VARCHAR2(50) NOT NULL,
upc_Status     VARCHAR2(50) NOT NULL,
upc_IdCiudad INTEGER REFERENCES Ciudad(ciu_IdCiudad)
```

Figure 3: Creaciòn de la Tablas.

3.4 Análisis

Actualmente la delincuencia en nuestro paìs cada vez es más grave. Existe estudios que somos uno de los paìses más peligrosos de Latinoamérica. Para este proyecto se realizó un análisis profundo que entidades pueden participar en nuestro diagrama entidad relaciòn para poder reducir este fenómeno que ataca al Ecuador. Nuestro sistema tiene varios puntos como el nombre del delincuente el apodo o la pandilla que pertenece.Seguidamente se registra el nombre de la persona civil con una entidad que nos indica la fecha del incidente y la forma de como fue el robo. Y por último nuestra base de datos registra la UPC que existen y los policías que están dentro y al barrio que pertenecen.

3.5 Optimizaciòn

Nuestro diseño presenta varias formas de actualización como de optimización según los conocimientos adquirido en cada clase de Gestión de Base Datos el modelo se va rediseñando una parte importante es tener normalizado nuestra base de datos.Al momento de normalizar nuestro sistema evitamos que existan datos vacios y redundantes. Es muy importante tener claro como va heredando las llaves primarias y secundarias. Una de las dificultades que se presentó fue al momento de querer conectar nuestra base de datos con python debido que toca tener varios conocimientos de programación. Por último en la creación de nuestra databases en oracle developer tenemos tener claro la sintaxis que maneja developer.

4 Evaluación Experimental

Para llevar a cabo la evulaciòn experimental es necesario seguir un proceso de pasos primeramente se generó datos sintéticos para nuestras 13 entidades ,seguidamente se descargó los datos en formato csv y por último se importa los datos. Las herramientas que se utilizó es Google Colab es una plataforma de programación con lenguaje de Python. Se utilizó también SQL Developer para la creación de la base de datos. Para la conexión entre el csv y la base de datos se utilizó Visual Studio que nos permite instalar extensiones que necesitemos según en el trascurso del proyecto. Una de las extensiones que utilizo para editar los datos es CSV edición que nos permitió editar las columnas y filas. Otra de las extensiones fue Python debido a que es un lenguaje de los populares en el mundo de la programación.

4.1 Descarga de los DATAFRAMES

Para la creación de la base de datos llamada delincuencia se necesita los dataframes que se realizó en los laboratorios de la primera unidad con los 5k sintéticos que se creó. También se agregó más dataframes debido a que nuestra ingeniera nos supo mencionar algunas entidades que se necesitaban para la creación de esta base de datos completa. Nuestros dataframes se compone por un, id por cada entidad un nombre y un status. Una vez que la información este de una forma correcta generamos el código para la descarga en formato csv, se necesitó eliminar la columna y fila que colab nos arroja automaticamente, ya que esto puede presentar en el orden de las columnas al momento de cargar esta información mediante las herramientas antes mencionadas que era EDIT CSV que contiene Visual studio.

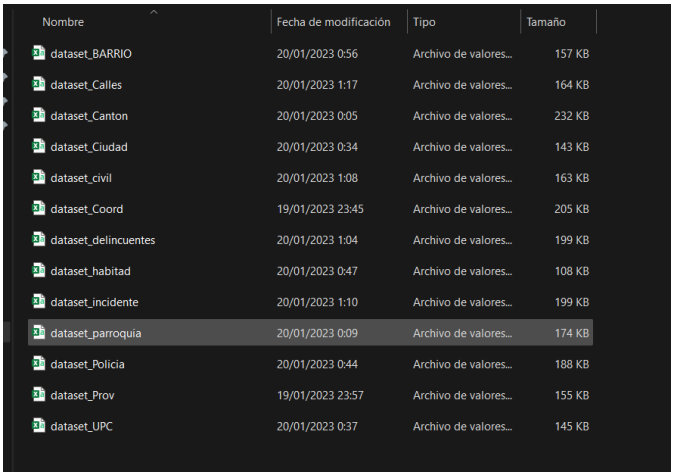


Figure 4: DataFrames.

4.2 Conexiòn a la base de datos

Este código está utilizando la biblioteca Pandas y SQLAlchemy para trabajar con una base de datos Oracle XE. El código está realizando las siguientes acciones. Crear una tabla llamada "tb-coordenada" en Oracle XE utilizando una conexión establecida mediante la función createengine de SQLAlchemy.Cargar un archivo CSV llamado "datasetCoord.csv" utilizando la función de Pandas readsv.Insertar los datos del archivo CSV en la tabla "tbcoordenada" recién creada utilizando la función tosql de Pandas.Utilizar una consulta SQL para seleccionar todos los datos de la tabla "tbcoordenada" y imprimirlos en consola.

```
'''Realizacion de coenxion a la base de datos'''
import pandas as pd
from sqlalchemy import create_engine

# Crear tabla en Oracle XE
engine = create_engine('oracle+cx_oracle://SYSTEM:Superat1234@localhost:1521/
    xe')
with engine.connect() as conn:
    conn.execute("""
        CREATE TABLE tb_coordenada(
nbr_coor_id               NUMBER PRIMARY KEY ,
var_coor_latitud          VARCHAR2(50) NOT NULL  ,
var_coor_longitud         VARCHAR2(50) NOT NULL
)
    """)

# Cargar dataset desde archivo CSV
data = pd.read_csv('dataset_Coord.csv')

# Insertar datos en tabla
data.to_sql('tb_coordenada', engine, if_exists='append', index=False)

# Unir tablas
with engine.connect() as conn:
    sql = """
        SELECT nbr_coor_id
        FROM tb_coordenada
```

```
    """
    result = conn.execute(sql)
    for row in result:
        print(row)
```

Listing 1: Conexión a la base de datos

4.3 Conexiòn de la entidad Coordenada

Este código está generando un conjunto de datos de coordenadas, utilizando varias bibliotecas. Importando las bibliotecas necesarias, incluyendo pandas para trabajar con dataframes, uuid para generar identificadores únicos, random para generar números aleatorios y faker para generar datos falsos.Se establece una variable numusers para indicar el número de datos a generar.Creamos un dataframe vacío llamado df, con las columnas "nbr coorid"var oorlatitud" y "var corlongitud".Implementamos una función coorid() que genera una lista de identificadores únicos mediante uuid y se asigna como valor para la columna "nbr coorid" del dataframe df Generamos una lista de números aleatorios entre -0 y 0.9 para la columna "var oorlatitud" y una lista de números aleatorios entre 79 y 78 para la columna "var corlongitud".Y se guarda el dataframe df como un archivo CSV llamado "dataset. A continuacion se dejara un link que redireccione a github en donde se encuantran todos las conexiones realizadas entre python y oracle Xe https://github.com/jonathan-elian-toapanta/Proyecto-DB.

```
'''Declaramos las librerias que usaremos, para poder generar los datos'''
'''Primero importamos todas las librerias
que vamos a utilizar'''
import uuid
import pandas as pd
import random
from faker import Faker
import datetime

#Numero de datos a obtener
num_users = 5000

# Generar 3 atributos de la entidad Coordenandas
features = [
    "nbr_coor_id",
    "var_coor_latitud",
    "var_coor_longitud "
]# Creating a DF for these features
df = pd.DataFrame(columns=features)

'''Mediante la libreria uuid se implementa un ID a Coordenadas,
el cual va a ser unico para cada dato'''
def coorid():
  i=1
  L=[]
  while i<=num_users:
    L.append(i)
    i=i+1
  return L

df['nbr_coor_id'] = coorid()
print(df['nbr_coor_id'].nunique()==num_users)
```

Listing 2: Conexión entidad coordenada

4.4 Consultas

Para esta secciòn se realizara estas consultas.Hallar el numero de UPC´s que existen en el sector de San Diego en la ciudad de Quito, en conjunto se debera conocer cuanto policias de rango cabo forman parte de un UPC.Conocer el numero de delincuentes en el sector de san diego y comparar con el numero de UPCs que habitan en el secto.Conocer las formas de aslasto que ocurren en el barrio san diego e identificar el numero de casos resueltos.Relacionar las coordenadas de latitud y longitud del en el barrio san Diego junto con las fechas de asaltos ocurrdos en el año 2022.Conocer el numero de pandillas que existen en el bario de san Diego .conocer a las personas que formaban parte de las pandillas en el barrio de SAN DIEGO. Identificar el estado civil del delincuente de los casos de robos que ya han sido resueltos ene el año 2022.Comparar el numero de upcs de tipo barrial que existe en el barrio de san diego y en el barrio de la magdalena en conjunto con el numero de casos resueltos en cada barrio.Hallar el nombre del barrrio en las que asaltan el delincuente apodado hunter y suforma de asaltar.Conocer el nombre del barrrio en las que asaltan la pandilla llamada los West y su forma de asaltar.

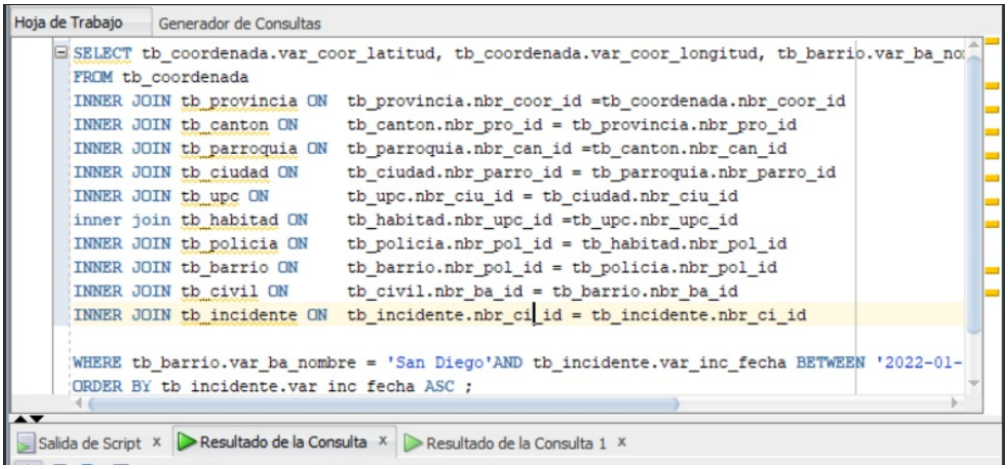


Figure 5: Còdigo para una consulta .

4.5 Python y Flask

Primeramente debemos tomar en cuenta que debemos instalar algunas librerias como flask para poder realizar la conexón con la base de datos la cual de igual manera us una libreria de cx Oracle para poder conectarse a la base de datos Oracle y mostrar los resultados de una consulta en una página web. El código define dos rutas, una para la página de inicio y otra para procesar un formulario de inicio de sesión. Cuando el usuario envía el formulario, la función procesa los datos ingresados y crea una cadena de conexión para establecer una conexión con la base de datos. Si la conexión es exitosa, se ejecuta una consulta y se muestran los resultados en una plantilla HTML. Si ocurre algún error durante el proceso, se muestra un mensaje de error al usuario en lugar de los resultados de la consulta. En resumen, este código muestra un ejemplo simple de cómo crear una aplicación web que interactúa con una base de datos Oracle.

```
'''Declaramos las librerias que usaremos, para poder generar los datos'''
from flask import Flask, render_template, request

import cx_Oracle

app = Flask(__name__)

# Ruta de la página de inicio
@app.route('/')
def index():
    return render_template('index.html')

# Ruta para procesar el formulario de inicio de sesión
@app.route('/login', methods=['POST'])
def login():
    username = request.form['username']
    password = request.form['password']

    # Combinar el nombre de usuario y la contraseña en la cadena de conexión
    con_string = username + '/' + password + '@localhost:1521/Db_Delincuencia'
    try:
        # Establecer una conexión con Oracle
        connection = cx_Oracle.connect(con_string)
        cursor = connection.cursor()

        # Ejecutar una consulta y obtener los resultados
        query = 'SELECT * FROM tb_coordenada'
        cursor.execute(query)
        rows = cursor.fetchall()

        # Cerrar la conexión y mostrar los resultados al usuario
        cursor.close()
        connection.close()
        return render_template('menu.html', rows=rows)

    except cx_Oracle.Error as error:
        return 'Error connecting to database: {}'.format(error)
```

Listing 3: Libreria Flask

Este código es una aplicación web construida con Flask que permite realizar una consulta a una base de datos Oracle. La aplicación tiene dos rutas: una para mostrar un formulario de consulta y otra para procesar la consulta. En la primera ruta, se devuelve la plantilla 'ejemplo.html', que contiene un formulario con tres campos: nombre de usuario, contraseña y consulta. En la segunda ruta, se capturan los valores de nombre de usuario, contraseña y consulta que se enviaron desde el formulario de la ruta anterior. Estos valores se combinan en una cadena de conexión con el formato requerido por cx Oracle. Luego se intenta establecer una conexión con la base de datos utilizando la cadena de conexión y se crea un cursor para ejecutar consultas. Se ejecuta la consulta obtenida del formulario y se obtienen los resultados. Luego se cierra el cursor y la conexión y se devuelve la plantilla 'resultados.html', que muestra los resultados en forma de tabla. Si se produce algún error al conectarse a la base de datos, se captura el error y se muestra al usuario un mensaje de error en lugar de los resultados de la consulta.

```python
'''Creamos una ruta para la conexion de la base de datos'''
# Ruta para mostrar el formulario de consulta
@app.route('/ejemplo')
def ejemplo():
    # Lógica de la vista
    return render_template('ejemplo.html')

# Ruta para procesar la consulta de la base de datos
@app.route('/consultar_db', methods=['POST'])
def consultar_db():
    username = request.form['username']
    password = request.form['password']
    consulta = request.form['consulta']

    # Combinar el nombre de usuario y la contraseña en la cadena de conexión
    con_string = username + '/' + password + '@localhost:1521/Db_Delincuencia'
    try:
        # Establecer una conexión con Oracle
        connection = cx_Oracle.connect(con_string)
        cursor = connection.cursor()

        # Ejecutar la consulta y obtener los resultados
        cursor.execute(consulta)
        rows = cursor.fetchall()

        # Cerrar la conexión y mostrar los resultados al usuario
        cursor.close()
        connection.close()
        return render_template('resultados.html', rows=rows)

    except cx_Oracle.Error as error:
        return 'Error connecting to database: {}'.format(error)
```

Listing 4: Conexion Base de Datos

En la siguiente parte del codigo se puede definir dos rutas definidas con la función @app.route. La primera ruta, /formulario contar upc, muestra un formulario para que el usuario ingrese los parámetros de una función. La segunda ruta, /inicio, establece una conexión con una base de datos Oracle, ejecuta una consulta para seleccionar todos los datos de la tabla tb coordenada y muestra los resultados en una plantilla HTML. Si ocurre algún error al conectarse a la base de datos, se muestra un mensaje de error en lugar de los resultados de la consulta.

```
'''Creacion de funciones y realizacion de querys'''
@app.route('/formulario_contar_upc')
def formulario_contar_upc():
    # Mostrar el formulario para ingresar los parámetros de la función
    return render_template('formulario_contar_upc.html')

@app.route('/inicio')
def inicio():
    # Establecer una conexión con Oracle
    con_string = 'delincuente/Oracle61076@localhost:1521/Db_Delincuencia'
    try:
        connection = cx_Oracle.connect(con_string)
        cursor = connection.cursor()

        # Ejecutar una consulta y obtener los resultados
        query = 'SELECT * FROM tb_coordenada'
        cursor.execute(query)
        rows = cursor.fetchall()

        # Cerrar la conexión y mostrar los resultados al usuario
        cursor.close()
        connection.close()
        return render_template('menu.html', rows=rows)

    except cx_Oracle.Error as error:
        # Mostrar un mensaje de error al usuario
        return render_template('error.html', error=error)
```

Listing 5: Funciones

El siguiente código es un archivo HTML que representa una página de inicio de sesión. El archivo contiene un formulario con dos campos, "Username" y "Password", y un botón de "Login" para enviar los datos al servidor. El formulario está conectado a la ruta "login" en la aplicación Flask a través del atributo "action" del formulario. El archivo HTML también importa un archivo CSS llamado "styles.css" y una imagen de un pájaro llamada "pajaro.png" a través de la función "url for" de Flask, que genera la ruta de acceso correcta a los archivos estáticos en el servidor. La imagen se muestra en la página junto con una tabla vacía. . La acción del formulario se establece en la ruta "/login" y se utiliza el método POST para enviar los datos del formulario al servidor.

```
'''Creacion de archivo index.html'''
<!DOCTYPE html>
<html>

<head>
          <title>Login</title>
        <link rel="stylesheet" type="text/css" href="{{ url_for('static',
    filename='styles.css', _external=True) }}">
</head>

<body>
    <center>
        <h1>Login</h1>
    </center>
    <form action="{{ url_for('login') }}" method="POST">
        <div>
            <label for="username">Username:</label>
            <input type="text" id="username" name="username" required>
        </div>
        <div>
            <label for="password">Password:</label>
            <input type="password" id="password" name="password" required>
        </div>
        <br>
        <div>
            <button type="submit">Login</button>
        </div>
    </form>
    <div class="center">
        <img id="logo" src="{{ url_for('static', filename='pajaro.png',
    _external=True) }}" alt="Logo">

        <table border="2">


</body>

</html>
```

Listing 6: Index.html

El siguiente código HTML define la estructura de una página web para consultar una base de datos de Unidades de Policía Comunitaria. El encabezado incluye un título y un menú de navegación con enlaces a diferentes secciones del sitio web, incluyendo la página de inicio, la sección "Sobre nosotros", los servicios ofrecidos, una consulta general y dos formularios para contar el número de UPC's por barrio y para contar pandillas. La sección de menú utiliza la función url for de Flask para generar enlaces dinámicos a las rutas correspondientes en el servidor. Además, se incluye una referencia a un archivo CSS externo para aplicar estilos al sitio web.

```
'''Creacion de archivo ejemplo.html que se usa para hacer el llamado a las
    otras paginas web'''
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Consultar base de datos</title>
    <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename
    ='menu.css', _external=True) }}">
  </head>
  <body>

    <header>
      <h1>Unidades de Policia Comunitaria</h1>

      <nav>
        <ul>
          <li><a href="{{ url_for('inicio')  }}">Inicio</a></li>
          <li><a href="#">Sobre nosotros</a></li>
          <li><a href="#">Servicios</a></li>
          <li><a href="{{ url_for('ejemplo') }}">Consulta General</a> </li>
          <li><a href="{{ url_for('formulario_contar_upc')  }}"method="GET">
    Numero de UPC's por barrio</a></li>
          <li><a href="{{ url_for('formulario_contar_pandillas') }}">Contar
    pandillas</a></li>
        </ul>
      </nav>
    </header>
```

Listing 7: Ejemplo.html

Este código es una página web que permite contar el número de Unidades de Policía Comunitaria (UPC) en un barrio específico. La página cuenta con un formulario que pide al usuario que ingrese el nombre de la ciudad y el nombre del barrio que desea consultar. Al hacer clic en el botón "Contar UPC", la página envía los datos ingresados por el usuario al servidor para procesar la información y devolver el número de UPC en el barrio seleccionado. Además, la página también cuenta con un menú de navegación que permite a los usuarios acceder a otras secciones de la página, como la página de inicio, la página de consulta general y la página de contar pandillas. La estructura del código está basada en HTML y utiliza la librería Flask para conectarse al servidor y procesar los datos del formulario. En resumen, este código es una herramienta útil para obtener información sobre la cantidad de UPC en un barrio específico de una ciudad.

```
'''Formulario de llamado a la pagina principal y hacer uso de la funcion
    contar pandillas'''
<!DOCTYPE html>
<html>
  <head>
    <title> Funci n Contar Pandillas</title>
    <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename
    ='menu.css', _external=True) }}">
  </head>
  <body>

    <header>
      <h1> Unidades de Polic a Comunitaria</h1>

      <nav>
        <ul>
          <li><a href="{{ url_for('inicio')  }}">Inicio</a></li>
          <li><a href="#">Sobre nosotros</a></li>
          <li><a href="#">Servicios</a></li>
          <li><a href="{{ url_for('ejemplo') }}">Consulta General</a> </li>
          <li><a href="{{ url_for('formulario_contar_upc')  }}"method="GET">
    Numero de UPC's por barrio</a></li>
          <li><a href="{{ url_for('formulario_contar_pandillas') }}">Contar
    pandillas</a></li>
        </ul>
      </nav>
    </header>

    <h1>Averigua el numero de pandillas que existen en tu barrio</h1>
    <form action="{{ url_for('contar_pandillas') }}" method="post">
      <label for="nombre_barrio">Nombre del barrio:</label>
      <input type="text" id="nombre_barrio" name="nombre_barrio"><br><br>
      <input type="submit" value="Contar Pandillas">
    </form>
  </body>
</html>
```

Listing 8: formulario$_c$*ontar*$_p$*andillas.html.html*

Este código HTML muestra un formulario que permite a los usuarios ingresar el nombre de un barrio y enviarlo al servidor para que se ejecute la función "contar pandillas" y se muestre el número de pandillas que existen en ese barrio. También incluye una sección de encabezado que contiene un menú de navegación que permite al usuario acceder a otras secciones de la página web. Además, el código utiliza una hoja de estilo CSS para dar formato al diseño visual de la página. En general, este código proporciona una funcionalidad útil para que los usuarios puedan obtener información sobre la seguridad en su comunidad.

```
'''Formulario de llamado a la pagina principal y hacer uso de la funcion
    contar UPC'''
<!DOCTYPE html>
<html>
  <head>
    <title>Funci n Contar Pandillas</title>
    <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename
    ='menu.css', _external=True) }}">
  </head>
  <body>

    <header>
      <h1>Unidades de Polic a Comunitaria</h1>

      <nav>
        <ul>
          <li><a href="{{ url_for('inicio')  }}">Inicio</a></li>
          <li><a href="#">Sobre nosotros</a></li>
          <li><a href="#">Servicios</a></li>
          <li><a href="{{ url_for('ejemplo') }}">Consulta General</a> </li>
          <li><a href="{{ url_for('formulario_contar_upc')  }}"method="GET">
    Numero de UPC's por barrio</a></li>
          <li><a href="{{ url_for('formulario_contar_pandillas') }}">Contar
    pandillas</a></li>
        </ul>
      </nav>
    </header>

    <h1>Averigua el numero de pandillas que existen en tu barrio</h1>
    <form action="{{ url_for('contar_pandillas') }}" method="post">
      <label for="nombre_barrio">Nombre del barrio:</label>
      <input type="text" id="nombre_barrio" name="nombre_barrio"><br><br>
      <input type="submit" value="Contar Pandillas">
    </form>
  </body>
</html>
```

Listing 9: formulario$_{c}ontar_{u}pc.html.html$

Los códigos presentados son un ejemplo de cómo se puede utilizar Flask para crear un sitio web dinámico con funcionalidades de base de datos. A través de los diferentes archivos HTML, se puede observar cómo se manejan las diferentes rutas y vistas de la aplicación, así como cómo se realizan consultas a la base de datos y se muestran los resultados en las páginas web. Además, se pueden apreciar algunas características útiles de Flask, como la incorporación de estilos CSS y la integración de formularios. En resumen, estos códigos proporcionan una idea general de cómo se puede implementar una aplicación web en Flask con funcionalidades de base de datos y cómo se pueden organizar las diferentes secciones de la página para ofrecer una experiencia de usuario intuitiva.

4.6 Gráfica

Como primer resultado a partir de la creación de las tablas en Oracle Developer nos presenta un diagrama realizado correctamente por lo que se estableció 11 entidades iniciales, y mediante las relaciones establecidas entre dichas entidades se generaron nuevas entidades, obteniendo un total de 13. El diagrama inicia desde las coordenadas obtenidas por Google GPS. Esas 2 coordenadas existen en las provincias del Ecuador, las cuales tienen cantones y a su vez estos contienen las parroquias en donde se encuentran las ciudades. En cada ciudad, se encuentra en diferentes unidades de policía comunitaria más conocidas como UPC, en las cuales habitan Policías que patrullan a los barrios en las cuales asaltan delincuentes y civiles que sufren incidentes. Para poder ejecutar es diagrama se estableció un proceso la cual es a archivos opción datamodel e ir la opción de importar y escoger la sección de diccionario de datos.

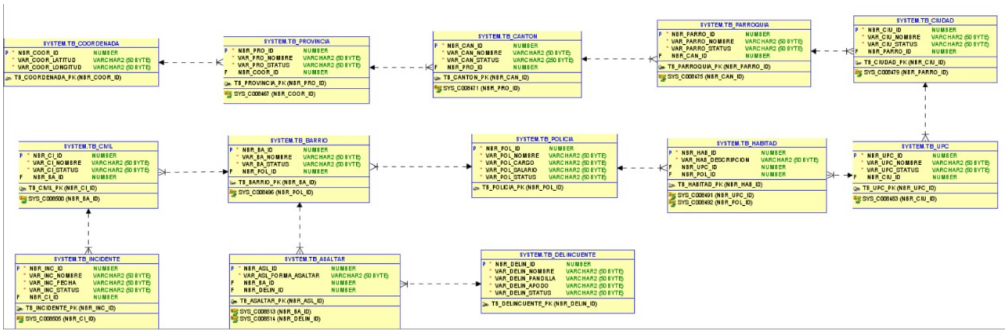


Figure 6: Diaagrama SQl Developer.

Se aprecia el modelo lógico de la base de datos creada donde describe la estructura de la base de datos de manera independiente del sistema de gestión de bases de datos (DBMS). Es decir, es una representación abstracta de la base de datos que se enfoca en la organización lógica de los datos y en cómo se relacionan entre sí. El modelo lógico se basa en un conjunto de reglas y notaciones para describir los datos y las relaciones entre ellos, y permite que los desarrolladores de bases de datos y los usuarios finales comprendan la estructura de la base de datos sin tener que preocuparse por la implementación física en un sistema de gestión de bases de datos en particular. El modelo lógico puede ser utilizado como una herramienta para la planificación, el diseño y la implementación de la base de datos.

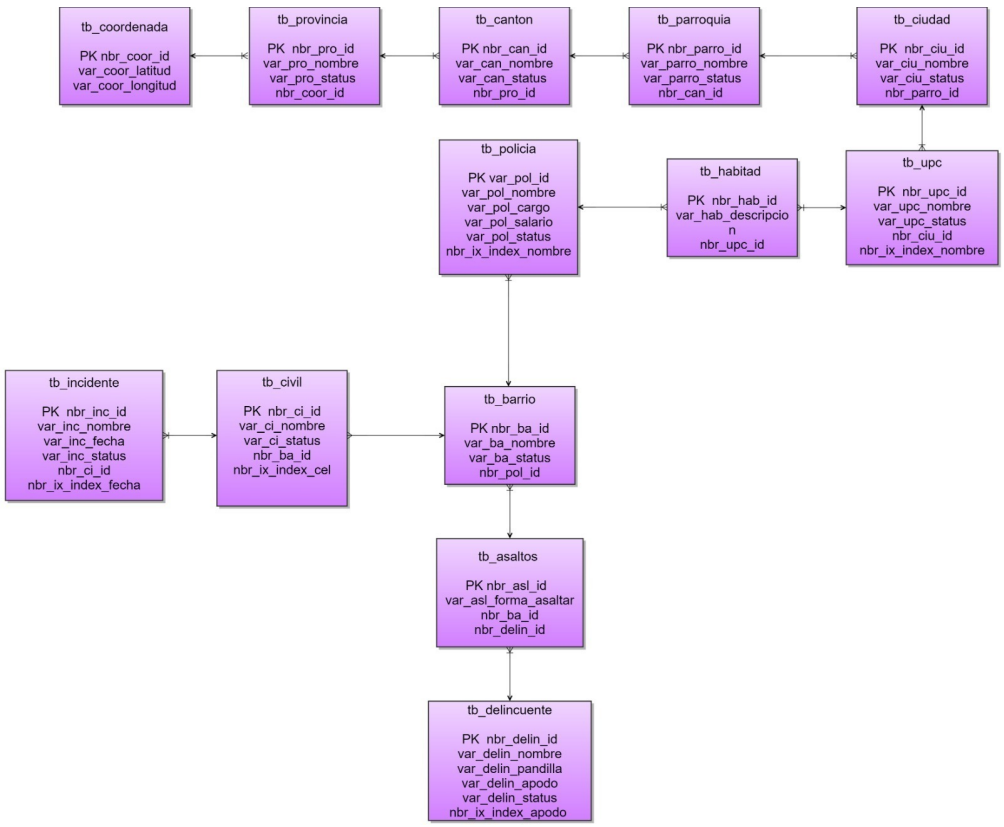


Figure 7: Modelo lógico.

Como podemos observar se muestra el modelo físico de la base de datos realizado en una aplcación Gliffy, donde se muestra a la implementación real de la estructura de la base de datos en un sistema de gestión de bases de datos (DBMS). Dicho modelo físico se construye a partir del modelo lógico y describe cómo se almacenan los datos en disco y cómo se acceden a ellos en el nivel físico. El modelo físico incluye detalles como el tamaño de los campos, el tipo de datos, las restricciones de integridad, las claves primarias y secundarias, los índices, las particiones y la asignación de almacenamiento. El objetivo del modelo físico es optimizar el rendimiento de la base de datos en términos de velocidad de acceso y capacidad de almacenamiento. El modelo físico es importante para los administradores de bases de datos y los desarrolladores de aplicaciones, ya que influye en el diseño de la base de datos, el rendimiento del sistema y la capacidad de almacenamiento.

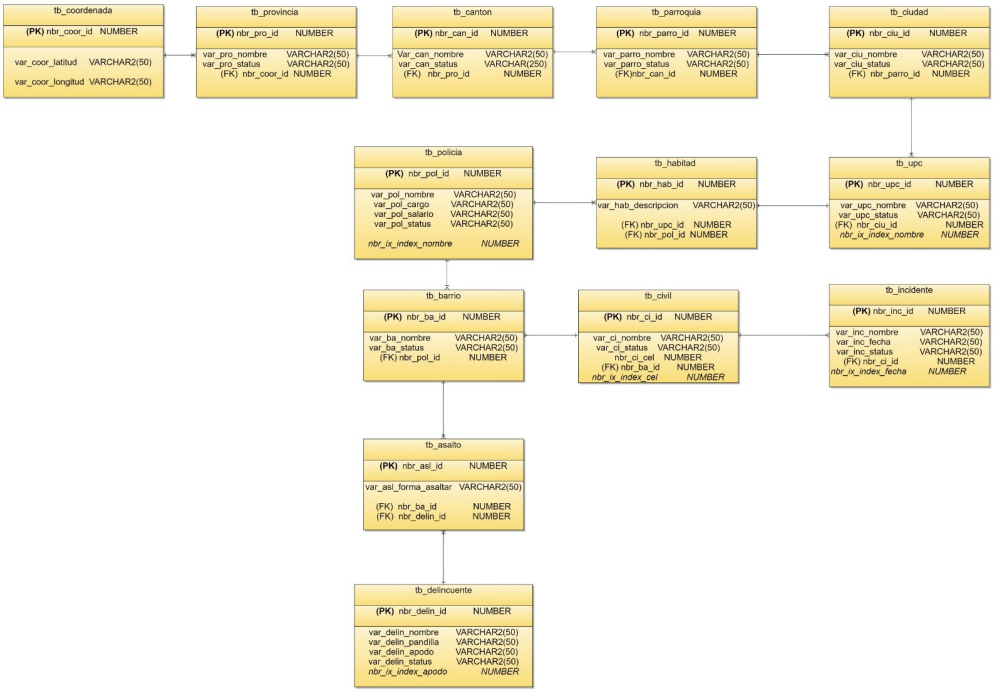
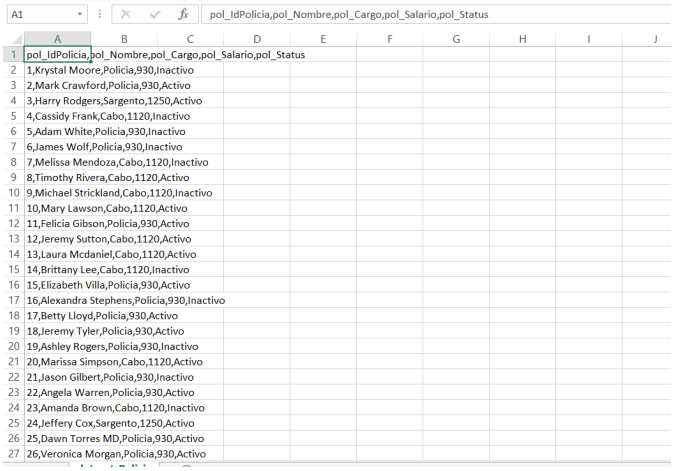


Figure 8: Modelo Físico.

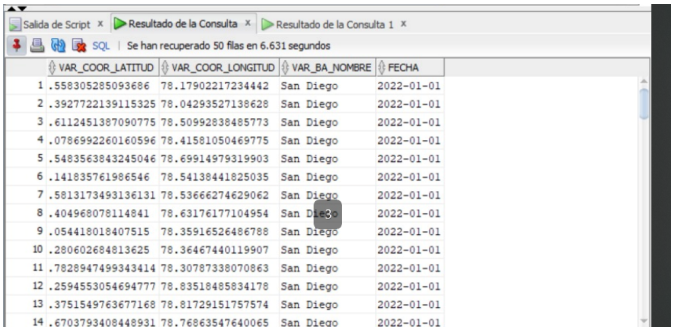
En el archivo csv que se generó al momento de crear los datos sintéticos de 5 K debe tener un formato específico. Como observamos en el gráficotenemos la entidad de Policía donde tiene un ID, nombre, status, salario, cargo la mayoría de las entidades conserva el mismo formato. Para observar como está compuesto las entidades en las secciones de antes se explicó como está formado nuestro diagrama entidad relación.



	pol_IdPolicia	pol_Nombre	pol_Cargo	pol_Salario	pol_Status
1	pol_IdPolicia	pol_Nombre	pol_Cargo	pol_Salario	pol_Status
2	1	Krystal Moore	Policia	930	Inactivo
3	2	Mark Crawford	Policia	930	Activo
4	3	Harry Rodgers	Sargento	1250	Activo
5	4	Cassidy Frank	Cabo	1120	Inactivo
6	5	Adam White	Policia	930	Inactivo
7	6	James Wolf	Policia	930	Inactivo
8	7	Melissa Mendoza	Cabo	1120	Inactivo
9	8	Timothy Rivera	Cabo	1120	Activo
10	9	Michael Strickland	Cabo	1120	Inactivo
11	10	Mary Lawson	Cabo	1120	Activo
12	11	Felicia Gibson	Policia	930	Activo
13	12	Jeremy Sutton	Cabo	1120	Activo
14	13	Laura Mcdaniel	Cabo	1120	Activo
15	14	Brittany Lee	Cabo	1120	Inactivo
16	15	Elizabeth Villa	Policia	930	Activo
17	16	Alexandra Stephens	Policia	930	Inactivo
18	17	Betty Lloyd	Policia	930	Activo
19	18	Jeremy Tyler	Policia	930	Activo
20	19	Ashley Rogers	Policia	930	Inactivo
21	20	Marissa Simpson	Cabo	1120	Activo
22	21	Jason Gilbert	Policia	930	Inactivo
23	22	Angela Warren	Policia	930	Activo
24	23	Amanda Brown	Cabo	1120	Inactivo
25	24	Jeffery Cox	Sargento	1250	Activo
26	25	Dawn Torres MD	Policia	930	Activo
27	26	Veronica Morgan	Policia	930	Activo

Figure 9: Formato CSV.

Esta consulta SQL selecciona ciertas columnas de varias tablas y las une mediante INNER JOIN.Las columnas que se seleccionan son "var coor latitud" y "var coorlongitud" de la tabla "coordenada", y "var banombre" de la tabla "tbbarrio".Las tablas se están uniendo en un orden específico: tb coordenada, tb provincia, tb canton, tb parroquia, tb ciudad, tb upc, tb habitat, tb policia, tb barrio, tb civil, tb incidente. Las condiciones de combinación se basan en hacer coincidir la clave principal de una tabla con la clave externa de otra tabla. La consulta también está filtrando los resultados para mostrar solo las filas donde la columna "var ba nombre" en la tabla "tb barrio" es igual a 'San Diego'.



Salida de Script | Resultado de la Consulta | Resultado de la Consulta 1

Se han recuperado 50 filas en 6.631 segundos

	VAR_COOR_LATITUD	VAR_COOR_LONGITUD	VAR_BA_NOMBRE	FECHA
1	.558305285093686	78.17902217234442	San Diego	2022-01-01
2	.3927722139115325	78.04293527138628	San Diego	2022-01-01
3	.6112451387090775	78.50992838485773	San Diego	2022-01-01
4	.0786992260160596	78.41581050469775	San Diego	2022-01-01
5	.5483563843245046	78.69914979319903	San Diego	2022-01-01
6	.141835761986546	78.54138441825035	San Diego	2022-01-01
7	.5813173493136131	78.53666274629062	San Diego	2022-01-01
8	.404968078114841	78.63176177104954	San Diego	2022-01-01
9	.0544180184075 15	78.35916526486788	San Diego	2022-01-01
10	.280602684813625	78.36467440119907	San Diego	2022-01-01
11	.7828947499343414	78.30787338070863	San Diego	2022-01-01
12	.2594553054694777	78.83518485834178	San Diego	2022-01-01
13	.3751549763677168	78.81729151757574	San Diego	2022-01-01
14	.6703793408448931	78.76863547640065	San Diego	2022-01-01

Figure 10: Resultado de la Consulta.

4.7 Vistas y Triggers

La vista "vw upc" es una consulta que combina información de varias tablas relacionadas de una base de datos. En este caso, la vista utiliza las tablas "tb upc", "tb ciudad", "tb parroquia", "tb canton" y "tb provincia" para obtener información sobre unidades de producción (UPC) y su ubicación geográfica en Ecuador. De igual manera en la vista selecciona el ID y nombre de cada UPC junto con el nombre de la ciudad, parroquia, cantón y provincia donde se encuentra ubicada. Para hacer que la información sea más fácil de leer, se han establecido formatos para cada columna de la vista. Al ejecutar la consulta "SELECT * FROM vw upc;", se mostrará una tabla con información sobre todas las UPC y su ubicación geográfica en la base de datos.

```
SQL> CREATE VIEW vw_upc AS
  2  SELECT u.nbr_upc_id, u.var_upc_nombre,
  3  ci.var_ciu_nombre, pa.var_parro_nombre, ca.var_can_nombre, p.var_pro_nombre
  4  FROM tb_upc u
  5  JOIN tb_ciudad ci ON u.nbr_ciu_id = ci.nbr_ciu_id
  6  JOIN tb_parroquia pa ON ci.nbr_parro_id = pa.nbr_parro_id
  7  JOIN tb_canton ca ON pa.nbr_can_id = ca.nbr_can_id
  8  JOIN tb_provincia p ON ca.nbr_pro_id = p.nbr_pro_id;

Vista creada.

SQL>
```

Figure 11: Vista 1.

```
NBR_UPC_ID VAR_UPC_NOMBRE       VAR_CIU_NOMBRE       VAR_PARRO_NOMBRE
---------- -------------------- -------------------- --------------------
VAR_CAN_NOMBRE       VAR_PRO_NOMBRE
-------------------- --------------------
Pedro Moncayo        Orellana

      4998 Lindsayberg          Quito                La Libertad
Distrito Metropolita Esmeraldas
no de Quito

      4999 Lesliefort           Aloag                Centro Hist├│rico
Cayambe              Pichincha


NBR_UPC_ID VAR_UPC_NOMBRE       VAR_CIU_NOMBRE       VAR_PARRO_NOMBRE
---------- -------------------- -------------------- --------------------
VAR_CAN_NOMBRE       VAR_PRO_NOMBRE
-------------------- --------------------
      5000 New Patrick          Quito                Centro Hist├│rico
Distrito Metropolita Azuay
no de Quito


5000 filas seleccionadas.
```

Figure 12: Vista 1 Resultado.

En la siguiente Fig: 13, la vista vw delincuente es creada a partir de la tabla tb delincuente y la tabla tb asalto. La vista se utiliza para mostrar información sobre los delincuentes y los asaltos a los que están asociados. La vista selecciona el ID, nombre, apodo y pandilla de cada delincuente de la tabla tb delincuente y el ID y la forma de asaltar de cada asalto de la tabla tb asalto. Se utiliza una combinación externa izquierda en la vista para garantizar que se incluyan todos los delincuentes, independientemente de si han participado en un asalto o no. Los resultados se muestran en un formato organizado y fácil de leer, con columnas para el ID del delincuente, el nombre, el apodo, la pandilla, el ID del asalto y la forma de asaltar.

```
SQL> CREATE VIEW vw_delincuente AS
  2  SELECT d.nbr_delin_id, d.var_delin_nombre, d.var_delin_apodo, d.var_delin_pandilla,
  3      a.nbr_asl_id AS asalto_id, a.var_asl_forma_asaltar AS asalto_forma
  4  FROM tb_delincuente d
  5  LEFT JOIN tb_asalto a ON d.nbr_delin_id = a.nbr_delin_id;

Vista creada.

SQL>
```

Figure 13: Vista 2.

```
NBR_DELIN_ID VAR_DELIN_NOMBRE     VAR_DELIN_APODO      VAR_DELIN_PANDILLA
------------ -------------------- -------------------- --------------------
 ASALTO_ID ASALTO_FORMA
---------- --------------------------------------------------
        4999 Gary Foster          perez                East
      4999 Con armas de fuego

        5000 Michaela Johnson     friedman             Lake
      5000 Con armas de fuego


5003 filas seleccionadas.
```

Figure 14: Vista 2 Resultado.

Como se muestra en la Fig: 15, la vista vw policia info combina información de las tablas tb policia, tb habitad y tb barrio. En ella se muestra el ID de los policías, su nombre, cargo, salario, y la descripción del hábitat y nombre del barrio donde trabajan. La vista utiliza un LEFT JOIN para asegurarse de que se muestren todos los policías, incluso aquellos que no tienen información en las tablas de hábitat y barrio. También se han definido los formatos de las columnas para mejorar la legibilidad de los resultados.

```
SQL>
SQL> CREATE VIEW vw_policia_info AS
  2  SELECT p.nbr_pol_id, p.var_pol_nombre, p.var_pol_cargo, p.var_pol_salario,
  3      h.var_hab_descripcion AS habitad, b.var_ba_nombre AS barrio
  4  FROM tb_policia p
  5  LEFT JOIN tb_habitad h ON p.nbr_pol_id = h.nbr_pol_id
  6  LEFT JOIN tb_barrio b ON p.nbr_pol_id = b.nbr_pol_id;

Vista creada.

SQL>
```

Figure 15: Vista 3.

```
ID_POL POL_NOMBRE          CARGO_POL
------ ------------------- -------------------
SALARIO                                          HABITAD
------------------------------------------------ --------------------
BARRIO
--------------------

  4999 Debra Davis          Cabo
1120                                             Movil
San Diego

  5000 Kelly French         Cabo
1120                                             Barrial

ID_POL POL_NOMBRE          CARGO_POL
------ ------------------- -------------------
SALARIO                                          HABITAD
------------------------------------------------ --------------------
BARRIO
--------------------
Guaman├¡


5002 filas seleccionadas.
```

Figure 16: Vista 3 Resultado.

Obeservando la La vista de la Fig: 17, "vw num incidentes barrio" está diseñada para mostrar el número de incidentes registrados en cada barrio. La vista utiliza la cláusula "JOIN" para unir las tablas "tb barrio", "tb civil" y "tb incidente" mediante sus claves correspondientes. Luego, la vista utiliza la función "COUNT" para contar el número de incidentes en cada barrio y agrupar los resultados por el nombre del barrio utilizando la cláusula "GROUP BY". La vista tiene dos columnas: "nombre barrio" que muestra el nombre del barrio y "num incidentes" que muestra el número de incidentes registrados en el barrio. Para mejorar la visualización de los resultados, se definen los formatos de columna para "nombre barrio" y "num incidentes" utilizando la cláusula "COLUMN". Finalmente, la vista se muestra utilizando la sentencia "SELECT" junto con el nombre de la vista "vw num incidentes barrio".

```
SQL> CREATE VIEW vw_num_incidentes_barrio AS
  2  SELECT b.var_ba_nombre as nombre_barrio, COUNT(i.nbr_inc_id) AS num_incidentes
  3  FROM tb_barrio b
  4  JOIN tb_civil c ON b.nbr_ba_id = c.nbr_ba_id
  5  JOIN tb_incidente i ON c.nbr_ci_id = i.nbr_ci_id
  6  GROUP BY b.var_ba_nombre;

Vista creada.

SQL>
```

Figure 17: Vista 4.

```
NOMBRE_BARRIO        NUM_INCIDENTES
-------------------- --------------
San Bartolo                      63
La Mena                          58
La Ferroviaria                   52
Comit├- del Pueblo               54
El Condado                       46
Centro Hist├|rico                57
Rumipamba                        47
San Juan                         48
Cochapamba                       63
Chimbacalle                      42
Calder├|n                        47

NOMBRE_BARRIO        NUM_INCIDENTES
-------------------- --------------
La Magdalena                     35

34 filas seleccionadas.

SQL>
```

Figure 18: Vista 4 Resultado.

El trigger en figure 15 se encarga de validar el estado de un delincuente antes de su inserción o actualización en la tabla "delincuente". Se dejecuta antes de la operación de inserción o actualización en cada fila de la tabla, gracias a la cláusula "FOR EACH ROW". El cuerpo del trigger contiene una condición que evalúa si el valor del campo "var delin status" de la fila que se está insertando o actualizando es válido. Si el valor no está en la lista de estados permitidos, el trigger arrojará un error con el mensaje "El estado del delincuente debe ser 'activo', 'inactivo' o 'capturado'". Esta validación garantiza la integridad de los datos y evita que se inserten o actualicen registros con estados inválidos en la tabla "tb delincuente", siendo una medida de seguridad para asegurar que los datos almacenados en la tabla sean consistentes y precisos.

```
SQL> CREATE OR REPLACE TRIGGER tg_validar_status_delincuente
  2  BEFORE INSERT OR UPDATE ON tb_delincuente
  3  FOR EACH ROW
  4  BEGIN
  5      IF :NEW.var_delin_status NOT IN ('activo', 'inactivo', 'capturado') THEN
  6          RAISE_APPLICATION_ERROR(-20001, 'El estado del delincuente debe ser "activo", "inactivo" o
"capturado"');
  7      END IF;
  8  END;
  9  /

Disparador creado.
```

Figure 19: Trigger 1.

El siguiente trigger se encarga de validar si un barrio existe antes de su inserción o actualización en la tabla "tb asalto". El trigger se dispara antes de la operación de inserción o actualización en cada fila de la tabla, gracias a la cláusula "FOR EACH ROW". El cuerpo del trigger contiene una consulta que cuenta el número de registros en la tabla que tienen el mismo valor en el campo "nbr ba id" que el valor que se está intentando insertar o actualizar en la tabla "tb asalto". Si la cuenta es igual a cero, el trigger arrojará un error con el mensaje "El barrio especificado no existe en la tabla tb barrio". La validación garantiza una medida de seguridad para asegurar la integridad referencial de los datos y evitar la inserción de registros con valores de barrio inválidos en la tabla "tb asalto".

```
SQL> CREATE OR REPLACE TRIGGER tg_validar_barrio_asalto
  2  BEFORE INSERT OR UPDATE ON tb_asalto
  3  FOR EACH ROW
  4  DECLARE
  5      v_count NUMBER;
  6  BEGIN
  7      SELECT COUNT(*) INTO v_count FROM tb_barrio WHERE nbr_ba_id = :NEW.nbr_ba_id;
  8      IF v_count = 0 THEN
  9          RAISE_APPLICATION_ERROR(-20001, 'El barrio especificado no existe en la tabla tb_barrio');
 10      END IF;
 11  END;
 12  /
```

Figure 20: Trigger 2.

El disparador tg registrar date incidente se utiliza para insertar la fecha y hora actual en el atributo var inc fecha de la tabla tb incidente cada vez que se realiza una nueva inserción en dicha tabla. Este disparador se ejecuta antes de la inserción de una nueva fila en la tabla tb incidente y utiliza la función TO CHAR para formatear la fecha y hora actual en un formato legible para el usuario. Proporcionando asi información valiosa para la gestión y el seguimiento de los incidentes reportados, y permitiendo una mejor comprensión de la evolución temporal de los sucesos. Además, la automatización de esta tarea a través del uso de un disparador garantiza la precisión y coherencia de la información registrada, lo que resulta en una mayor confiabilidad de los datos almacenados.

```
SQL> CREATE OR REPLACE TRIGGER tg_registrar_date_incidente
  2  BEFORE INSERT ON tb_incidente
  3  FOR EACH ROW
  4  BEGIN
  5      :NEW.var_inc_fecha := TO_CHAR(SYSDATE, 'DD-MON-YYYY HH24:MI:SS');
  6  END;
  7  /

Disparador creado.
```

Figure 21: Trigger 3.

El objetivo de este trigger es validar que la ciudad especificada en la inserción o actualización de un registro exista en la tabla tb ciudad. El trigger es creado con la cláusula BEFORE INSERT OR UPDATE, lo que permite que se active antes de la inserción o actualización de un registro. En su cuerpo, se declara una variable v count y se realiza una consulta a la tabla tb ciudad para verificar si el valor especificado en la columna nbr ciu id existe. Si el valor no existe, se levanta un error de aplicación que informa al usuario que la ciudad especificada no existe en la tabla tb ciudad. La implementación de este trigger ayuda a garantizar la integridad de los datos en la tabla, asegurando que los valores ingresados en la columna nbr ciu id sean válidos.

```
SQL> CREATE OR REPLACE TRIGGER tg_validar_ciu_upc
  2  BEFORE INSERT OR UPDATE ON tb_upc
  3  FOR EACH ROW
  4  DECLARE
  5      v_count NUMBER;
  6  BEGIN
  7      SELECT COUNT(*) INTO v_count FROM tb_ciudad WHERE nbr_ciu_id = :NEW.nbr_ciu_id;
  8      IF v_count = 0 THEN
  9          RAISE_APPLICATION_ERROR(-20001, 'La ciudad especificada no existe en la tabla tb_ciudad');
 10      END IF;
 11  END;
 12  /
```

Figure 22: Trigger 4.

El siguiente trigger se utiliza para validar la inserción o actualización de datos en la tabla tb civil. El trigger se activa antes de la operación y su función principal es verificar si el valor que se desea insertar o actualizar en la columna nbr ba id de la tabla tb civil existe en la columna nbr ba id de la tabla tb barrio. Si el valor no existe, se produce un error y se lanza una excepción. De esta manera, se garantiza la integridad de los datos y se evita la inserción de valores incorrectos o inválidos. Esto es de gran utilidad en entornos donde se requiere un alto nivel de control y seguridad en la manipulación de datos, como en sistemas de gestión de bases de datos críticos

para la policia nacional del Ecuador.

```
SQL> CREATE OR REPLACE TRIGGER tg_validar_ba_id
  2  BEFORE INSERT OR UPDATE ON tb_civil
  3  FOR EACH ROW
  4  DECLARE
  5      v_count NUMBER;
  6  BEGIN
  7      SELECT COUNT(*) INTO v_count FROM tb_barrio WHERE nbr_ba_id = :NEW.nbr_ba_id;
  8      IF v_count = 0 THEN
  9          RAISE_APPLICATION_ERROR(-20001, 'El barrio especificado no existe en la tabla tb_barrio');
 10      END IF;
 11  END;
 12  /

Disparador creado.
```

Figure 23: Trigger 5.

La figure 23 muestra a la función "fn contar pandillas", que busca contar el número de pandillas que operan en un barrio específico en la ciudad de Quito. El código utiliza INNER JOIN para unir las tablas tb ciudad, tb upc, tb habitad, tb policia, tb barrio, tb asalto y tb delincuente. A través de esta consulta, se selecciona el número de pandillas únicas que se han identificado a través de incidentes de asalto en el barrio especificado. La función recibe el nombre del barrio como parámetro y devuelve el número total de pandillas encontradas. Este tipo de función es útil en la recopilación de información sobre la actividad criminal en un área geográfica específica y puede ayudar a las autoridades a diseñar estrategias de seguridad y prevención de delitos en consecuencia.

```
SQL> CREATE OR REPLACE FUNCTION fn_contar_pandillas(
  2      nombre_barrio IN VARCHAR2
  3  )
  4  RETURN NUMBER
  5  IS
  6      v_total NUMBER;
  7  BEGIN
  8      SELECT COUNT(DISTINCT tb_delincuente.var_delin_pandilla)
  9      INTO v_total
 10      FROM tb_ciudad
 11      INNER JOIN tb_upc ON tb_upc.nbr_ciu_id = tb_ciudad.nbr_ciu_id
 12      INNER JOIN tb_habitad ON tb_habitad.nbr_upc_id = tb_upc.nbr_upc_id
 13      INNER JOIN tb_policia ON tb_policia.nbr_pol_id = tb_habitad.nbr_pol_id
 14      INNER JOIN tb_barrio ON tb_barrio.nbr_pol_id = tb_policia.nbr_pol_id
 15      INNER JOIN tb_asalto ON tb_asalto.nbr_ba_id = tb_barrio.nbr_ba_id
 16      INNER JOIN tb_delincuente ON tb_delincuente.nbr_delin_id = tb_asalto.nbr_delin_id
 17      WHERE tb_barrio.var_ba_nombre = nombre_barrio
 18        AND tb_ciudad.var_ciu_nombre = 'Quito';
 19
 20      RETURN v_total;
 21  END;
 22  /

Funci¤n creada.
```

Figure 24: Funcion.

La función "fn contar upc" permite contar el número de unidades de policía comunitaria (UPC) disponibles en un barrio de una ciudad específica. Mediante un INNER JOIN se combina las tablas "tb ciudad", "tb upc", "tb habitad", "tb policia" y "tb barrio" para obtener la información necesaria. Los parámetros de entrada de la función son el nombre de la ciudad y el nombre del barrio. La función utiliza estos parámetros para filtrar los resultados de la consulta SQL y devolver el número total de UPC en el barrio y la ciudad especificados. Este tipo de función es útil para obtener información estadística sobre la distribución de la policía comunitaria en una ciudad específica y puede ser utilizada para fines de planificación y toma de decisiones en el ámbito de la seguridad ciudadana.

```
SQL> CREATE OR REPLACE TRIGGER tb_policia_audit_trg
  2  AFTER INSERT OR UPDATE OR DELETE ON tb_policia
  3  FOR EACH ROW
  4  DECLARE
  5      v_audit_type VARCHAR2(6);
  6  BEGIN
  7      IF INSERTING THEN
  8          v_audit_type := 'INSERT';
  9      ELSIF UPDATING THEN
 10          v_audit_type := 'UPDATE';
 11      ELSE
 12          v_audit_type := 'DELETE';
 13      END IF;
 14
 15  END;
 16  /
```

Figure 25: Funcion

5 Resultados

En conclusión, el código presentado en este trabajo demuestra una solución eficaz para el cálculo del número de pandillas en un barrio específico de la ciudad de Quito, Ecuador. La función fn contar pandillas utiliza una serie de tablas y relaciones para obtener los datos necesarios y aplicar una función de conteo que devuelve un valor numérico. Esta solución puede ser utilizada por autoridades locales y fuerzas de seguridad para comprender mejor la situación delictiva en una zona específica y tomar medidas preventivas y correctivas en consecuencia. Además, el código presentado también puede servir como base para futuras investigaciones y mejoras en el campo de análisis de datos criminales. Ademas, la función fn contar upc utiliza una combinación de cláusulas JOIN y condiciones WHERE para recuperar los datos necesarios de varias tablas relacionales. El resultado se devuelve en forma de un valor numérico que indica la cantidad total de UPC en la ciudad y el barrio especificados. Este código puede ser utilizado por las autoridades locales y los investigadores para entender mejor la distribución y la cobertura de las UPC en una zona específica y tomar decisiones informadas en consecuencia. Además, este trabajo también puede servir como punto de partida para futuras investigaciones y mejoras en el análisis de datos criminales y la toma de decisiones. Finalmente,La creación de la tabla tb policia audit permite el seguimiento de los cambios realizados en la información de la policía, y el trigger tb policia audit trg registra automáticamente estos cambios en la tabla tb policia audit. Esto proporciona una forma efectiva de mantener un registro preciso de los cambios realizados en la información de la policía, lo que puede ser útil para fines de transparencia, responsabilidad y seguridad. Además, la solución presentada se puede personalizar y adaptar según las necesidades específicas de la aplicación y los usuarios finales. En resumen, este código proporciona una solución simple y efectiva para mantener un registro de auditoría de los cambios realizados en la información de la policía en una base de datos Oracle.

6 Conclusión

En conclusion, la implementación de una base de datos y página web para la Policía Nacional del Ecuador utilizando Python, Flask, SQLPlus y HTML, representa una solución tecnológica eficiente y accesible al público para la seguridad ciudadana en el país. La creación de relaciones entre las tablas y la representación en los diagramas, junto con la automatización de tareas a través de triggers y funciones, garantizan la integridad y confiabilidad de los datos almacenados en la base de datos.La recolección, almacenamiento y análisis de información relevante sobre la delincuencia en Ecuador permitirá identificar patrones y tendencias en la delincuencia, lo que puede colaborar con las autoridades en la toma de decisiones en el ámbito de la seguridad ciudadana. La combinación de esta base de datos con una página web accesible al público brinda una herramienta adicional para la comunidad en la prevención y denuncia de delitos, contribuyendo así a la creación de una cultura de seguridad ciudadana.

Por lo tanto, esta implementación tecnológica es un paso importante en la lucha contra la delincuencia en Ecuador y puede ser utilizado como modelo para otros países en la implementación de sistemas de gestión de datos eficientes y accesibles al público para la seguridad ciudadana. Es necesario seguir investigando y mejorando estos sistemas para lograr un impacto aún mayor en la seguridad ciudadana a nivel global.

Regenerate resp

References

[1] R. P. Camargo Escorcia, J. I. Ibarra Orozco, and A. M. Morales Floraison, "Interprete para analizar y ejecutar las sentencias del lenguaje estructurado de consulta sql plus (oracle) que permita mejorar las prácticas en la asignatura de laboratorio de informática v de la facultad de ingeniería de sistemas en la universidad simón bolívar," 2003.

[2] F. C. Jara, "Ecuador consta entre los tres países con más robos y asaltos durante los primeros cuatro meses del 2022, según encuesta de cid gallup.," 2022.

[3] "Inseguridad quito: escuela de formación de policías se inauguró en el barrio san diego." `https://www.ecuavisa.com/noticias/ecuador/inseguridadquitoescueladeformaciondepoliciasseinauguroenelbarriosandiegoBN3539131`, Oct. 2022. Accessed: 2023-3-4.

[4] D. A. Toapanta Jonathan, "$[\mathrm{NRC}_8 393]_L ab2Unidad2_S CRIPT_T OAPANTA_A YO.''$.

H. S., "Fundamentos de los sistemas de información," *Bases de datos relacionales: definición de relaciones entre tablas de bases de datos. República Tecnológica*, 2003.

M. Seltzer, "El acceso a los datos es más que oracle: más allá de las bases de datos relacionales," *Comunicaciones de la ACM*, vol. 7, no. 51, pp. 52–58, 2008.