MASTERING SQL ZERO TO HERO ORACLE SQL DEVELOPER

PUBLIC SAFETY DATABASE SQL PROJECT FROM ZERO

RUSLAN BORISEVICH

Project Overview

This guide outlines the development of a comprehensive Public Safety Database using SQL. The database is designed to assist law enforcement agencies in managing and accessing critical information about individuals of interest, criminal records, and ongoing investigations.

Key Features:

- Central repository for person-related data
- Tracking of criminal records and wanted notices
- Management of aliases and reported sightings
- Scalable design to accommodate future expansion

Database Structure:

1.Persons
2.CriminalRecords
3.WantedNotices
4.Aliases
5.Sightings

These tables are interconnected using foreign key relationships, allowing for complex queries and data integrity.

RUSLAN BORISEVICH

Purpose

This project demonstrates advanced SQL database design and query capabilities while addressing real-world challenges in public safety. It showcases skills in:

- Database schema design
- SQL query writing
- Data relationship modeling
- Consideration of privacy and security concerns

Note: Implementation of such a system in a real-world scenario would require strict adherence to legal standards, privacy regulations, and ethical considerations.

RUSLAN BORISEVICH

Implementation of the Public Safety Database: Creating the Database and Tables

Step 1: Create the Database:

```
CREATE DATABASE public_safety_db;
```

Step 2: USE: Select the Public Safety Database to Work With:

```
USE public_safety;
```

Next steps we will create the 5 tables in our Public Safety Database :

- 1.Persons: Stores personal details of individuals.
- 2.CriminalRecords: Stores criminal records associated with persons.
- 3.WantedNotices: Contains notices for individuals wanted by authorities.
- 4.Aliases: Stores alternative names (aliases) for individuals.
- 5.Sightings: Stores reports or tips about the sightings of individuals.

WITH EXPLNATION FOR EACH COLUMN IN EACH TABLE

RUSLAN BORISEVICH

Implementation of the Public Safety Database: Creating the Database and Tables

Step 3: Create the Persons Table:

```
CREATE TABLE Persons (
    person_id INT PRIMARY KEY,
    first_name VARCHAR2(50),
    last_name VARCHAR2(50),
    birth_date DATE,
    gender VARCHAR2(10),
    nationality VARCHAR2(50),
    last_known_address VARCHAR2(255),
    biometric_id VARCHAR2(100),
    is_wanted NUMBER(1) DEFAULT 0,
    date_added DATE,
    last updated TIMESTAMP
);
```

Explanation:

- PersonID: A unique identifier for each person.
- FirstName, LastName: The individual's first and last names.
- DateOfBirth: The individual's date of birth.
- Gender, Nationality: Gender and nationality details.
- LastKnownAddress: The last known address of the individual.
- BiometricID: A unique identifier for biometric information (e.g., fingerprint ID).
- IsWanted: A boolean flag indicating if the person is wanted by authorities.
- DateAdded: The date the record was added.
- LastUpdated: A timestamp of the last update to the record.

NOTE :ISWANTED IS DEFINED AS A NUMBER(1) BECAUSE ORACLE DOESN'T HAVE A BOOLEAN TYPE. 0 IS USED FOR FALSE, AND 1 FOR TRUE.

RUSLAN BORISEVICH

Implementation of the Public Safety Database: Creating the Database and Tables

Step 4: Create the CriminalRecords Table:

```
CREATE TABLE CriminalRecords (
    record_id INT PRIMARY KEY,
    person_id INT,
    offense_type VARCHAR2(50),
    offense_date DATE,
    conviction VARCHAR2(10),
    sentence_details CLOB,
    FOREIGN KEY (person_id) REFERENCES Persons(person_id)
);
```

Explanation:

- RecordID: A unique identifier for each criminal record.
- PersonID: Links the criminal record to the person in the Persons table.
- OffenseType: The type of offense (e.g., theft, assault).
- OffenseDate: The date the offense occurred.
- Conviction: The conviction status or outcome (e.g., guilty, acquitted).
- SentenceDetails: Details about the sentence, if applicable.

Implementation of the Public Safety Database: Creating the Database and Tables

Step 5: Create the WantedNotices Table:

```
CREATE TABLE WantedNotices (
    notice_id INT PRIMARY KEY,
    person_id INT,
    issuing_authority VARCHAR2(100),
    date_issued DATE,
    reason CLOB,
    severity VARCHAR2(20),
    reward_amount NUMBER(10, 2),
    FOREIGN KEY (person_id) REFERENCES Persons(person_id)
);
```

Explanation:

- **NoticeID:** A unique identifier for each wanted notice.
- **PersonID:** Links the wanted notice to the person in the Persons table.
- **IssuingAuthority:** The authority issuing the wanted notice (e.g., police department).
- **DateIssued:** The date the notice was issued.
- **Reason:** The reason why the person is wanted.
- **Severity:** The severity of the notice (e.g., high, medium, low).
- **RewardAmount:** The reward offered for information leading to the person's capture.

Implementation of the Public Safety Database: Creating the Database and Tables

Step 6: Create the Aliases Table:

```
CREATE TABLE Aliases (
    alias_id INT PRIMARY KEY,
    person_id INT,
    alias_first_name VARCHAR2(50),
    alias_last_name VARCHAR2(50),
    FOREIGN KEY (person_id) REFERENCES Persons(person_id)
);
```

Explanation:

- AliasID: A unique identifier for each alias.
- PersonID: Links the alias to the person in the Persons table.
- AliasFirstName, AliasLastName: The alternative first and last names used by the person.

RUSLAN BORISEVICH

Implementation of the Public Safety Database: Creating the Database and Tables

Step 7: Create the Sightings Table:

```
CREATE TABLE Sightings (
    sighting_id INT PRIMARY KEY,
    person_id INT,
    date_reported TIMESTAMP,
    location VARCHAR2(255),
    description CLOB,
    reporter_contact VARCHAR2(100),
    FOREIGN KEY (person_id) REFERENCES Persons(person_id)
);
```

Explanation:

- SightingID: A unique identifier for each sighting report.
- PersonID: Links the sighting to the person in the Persons table.
- DateReported: The date and time the sighting was reported.
- Location: The location where the sighting occurred.
- Description: A description of the sighting or the circumstances.
- ReporterContact: Contact information of the person who reported the sighting.

Lets fill all the tables with data:

```
-- Inserting data into Persons table
INSERT INTO Persons (PersonID, FirstName, LastName, DateOfBirth, Gender, Nationality, LastKnownAddress, BiometricID, IsWanted, DateAdded, LastUpdated) VALUES
(1, 'John', 'Doe', '1985-03-15', 'Male', 'USA', '123 Main St, Anytown, AN 12345', 'BIO123456', TRUE, '2023-01-01', CURRENT_TIMESTAMP),
(2, 'Jane', 'Smith', '1990-07-22', 'Female', 'Canada', '456 Elm St, Othertown, OT 67890', 'BIO789012', FALSE, '2023-02-15', CURRENT_TIMESTAMP),
(3, 'Michael', 'Johnson', '1988-11-30', 'Male', 'UK', '789 Oak Rd, Somewhere, SW 54321', 'BIO345678', TRUE, '2023-03-10', CURRENT_TIMESTAMP),
(4, 'Emily', 'Brown', '1992-05-18', 'Female', 'Australia', '321 Pine Ave, Nowhere, NW 98765', 'BIO901234', FALSE, '2023-04-05', CURRENT_TIMESTAMP),
(5, 'David', 'Lee', '1983-09-07', 'Male', 'South Korea', '654 Cedar Ln, Anywhere, AW 13579', 'BIO567890', TRUE, '2023-05-20', CURRENT_TIMESTAMP);

-- Inserting data into CriminalRecords table
INSERT INTO CriminalRecords (RecordID, PersonID, OffenseType, OffenseDate, Conviction, SentenceDetails) VALUES
(1, 1, 'Theft', '2022-12-10', 'Guilty', '6 months probation'),
(2, 1, 'Assault', '2023-02-05', 'Guilty', '1 year imprisonment'),
(3, 3, 'Fraud', '2023-01-20', 'Guilty', '2 years imprisonment, $10,000 fine'),
(4, 5, 'Cybercrime', '2023-04-15', 'Pending Trial', 'N/A');

-- Inserting data into WantedNotices table
INSERT INTO WantedNotices (NoticeID, PersonID, IssuingAuthority, DateIssued, Reason, Severity, RewardAmount) VALUES
(1, 1, 'Anytown Police Department', '2023-06-01', 'Parole Violation', 'Moderate', 1000.00),
(2, 3, 'Interpol', '2023-05-15', 'International Fraud', 'High', 5000.00),
(3, 5, 'FBI', '2023-07-01', 'Cybercrime', 'High', 10000.00);

-- Inserting data into Aliases table
INSERT INTO Aliases (AliasID, PersonID, AliasFirstName, AliasLastName) VALUES
(1, 1, 'Johnny', 'Smith'),
(2, 1, 'Jack', 'Davis'),
(3, 3, 'Mike', 'Jackson'),
(4, 5, 'Daniel', 'Kim');

-- Inserting data into Sightings table
INSERT INTO Sightings (SightingID, PersonID, DateReported, Location, Description, ReporterContact) VALUES
(1, 1, '2023-06-15 14:30:00', 'Central Park, Anytown', 'Seen jogging in the park', 'anonymous'),
(2, 1, '2023-06-20 09:45:00', 'Main Street Cafe, Anytown', 'Spotted having coffee', '555-0123'),
(3, 3, '2023-06-18 18:20:00', 'Heathrow Airport, London', 'Attempting to board a flight', 'airport security'),
(4, 3, '2023-06-25 11:10:00', 'Paris, France', 'Seen near the Eiffel Tower', '555-0456'),
(5, 5, '2023-07-05 22:05:00', 'Cybercafe, Seoul', 'Using a computer', 'cafe owner');
```

Lets add a few more persons to the database, including some based on known historical criminal figures:

```
-- Adding more persons to the Persons table
INSERT INTO Persons (PersonID, FirstName, LastName, DateOfBirth, Gender, Nationality, LastKnownAddress, BiometricID, IsWanted, DateAdded, LastUpdated) VALUES
(6, 'Pablo', 'Escobar', '1949-12-01', 'Male', 'Colombia', 'Last known: Medellín, Colombia', 'BIO123ESC', TRUE, '1989-08-01', CURRENT_TIMESTAMP),
(7, 'Alphonse', 'Capone', '1899-01-17', 'Male', 'USA', 'Alcatraz Island, San Francisco, CA', 'BIO456CAP', FALSE, '1929-03-15', CURRENT_TIMESTAMP),
(8, 'Bonnie', 'Parker', '1910-10-01', 'Female', 'USA', 'Last seen: Bienville Parish, LA', 'BIO789BON', TRUE, '1933-05-20', CURRENT_TIMESTAMP),
(9, 'Clyde', 'Barrow', '1909-03-24', 'Male', 'USA', 'Last seen: Bienville Parish, LA', 'BIO101CLY', TRUE, '1933-05-20', CURRENT_TIMESTAMP);

-- Adding criminal records
INSERT INTO CriminalRecords (RecordID, PersonID, OffenseType, OffenseDate, Conviction, SentenceDetails) VALUES
(5, 6, 'Drug Trafficking', '1976-03-15', 'Convicted in absentia', 'Multiple life sentences'),
(6, 7, 'Tax Evasion', '1931-10-17', 'Guilty', '11 years in federal prison'),
(7, 8, 'Armed Robbery', '1932-08-05', 'Not convicted', 'Died before trial'),
(8, 9, 'Murder', '1932-08-05', 'Not convicted', 'Died before trial');

-- Adding wanted notices
INSERT INTO WantedNotices (NoticeID, PersonID, IssuingAuthority, DateIssued, Reason, Severity, RewardAmount) VALUES
(4, 6, 'DEA', '1984-07-01', 'International Drug Trafficking', 'Extreme', 2000000.00),
(5, 8, 'FBI', '1933-05-20', 'Interstate Bank Robbery', 'High', 10000.00),
(6, 9, 'FBI', '1933-05-20', 'Interstate Bank Robbery and Murder', 'High', 10000.00);

-- Adding aliases
INSERT INTO Aliases (AliasID, PersonID, AliasFirstName, AliasLastName) VALUES
(5, 6, 'El Padrino', ''),
(6, 7, 'Scarface', ''),
(7, 8, 'Bonnie', 'Parker'),
(8, 9, 'Clyde', 'Champion');

-- Adding sightings
INSERT INTO Sightings (SightingID, PersonID, DateReported, Location, Description, ReporterContact) VALUES
(6, 6, '1984-11-30 10:15:00', 'Medellín, Colombia', 'Seen entering a known safe house', 'confidential informant'),
(7, 8, '1934-05-23 14:30:00', 'Bienville Parish, Louisiana', 'Driving a stolen Ford V8', 'local sheriff'),
(8, 9, '1934-05-23 14:30:00', 'Bienville Parish, Louisiana', 'Driving a stolen Ford V8', 'local sheriff');
```

5 EASY QUERIES YOU CAN USE WITH THE PUBLIC SAFETY DATABASE:

1. List all persons who are wanted:

```
SELECT first_name, last_name
FROM Persons
WHERE is_wanted = 1;
```

Explanation:
This query retrieves the first and last names of all individuals who are marked as "wanted" in the persons table.
The is_wanted column is a boolean (or number) where 1 indicates that the person is wanted by authorities.

2. Count the total number of records in the `Persons` table:

```
SELECT COUNT(*) AS total_persons
FROM Persons;
```

Explanation:
This query counts the total number of records (rows) in the persons table, which gives the total number of individuals stored in the database.

3. Find the birth dates of all individuals named "John":

```
SELECT first_name, last_name, birth_date
FROM Persons
WHERE  first_name =  'John' ;
```

Explanation:
This query retrieves the first name, last name, and birth date of all individuals whose first name is "John."
It filters the records in the persons table to find matches based on the first_name column.

4.Get the last known address of a person by their person_id:

```
SELECT last_known_address
FROM Persons
WHERE person_id =  3 ;
```

Explanation:
This query retrieves the last known address of a specific individual based on their unique person_id.

5.List all sightings reported in the last 7 days:

```
SELECT *
FROM sightings
WHERE DATE_REPORTED >= CURRENT_DATE - INTERVAL '7' DAY;
```

EXPLANATION:
This query retrieves all columns (*) from the sightings table where the date_reported is within the last 7 days.
This helps track recent sightings or tips reported in the system.

RUSLAN BORISEVICH

5 MEDIUM QUERIES YOU CAN USE WITH THE PUBLIC SAFETY DATABASE:

1. Find all persons with criminal records:

```
SELECT p.first_name, p.last_name, cr.offense_type, cr.offense_date
FROM persons p
JOIN criminal_records cr ON p.person_id = cr.person_id;
```

Explanation:
TThis query retrieves the first name, last name, offense type, and offense date for all persons who have a criminal record. It joins the persons table with the criminal_records table based on the person_id.

2. List all wanted persons along with the date their wanted notice was issued:

```
SELECT p.first_name, p.last_name, wn.date_issued
FROM persons p
JOIN wanted_notices wn ON p.person_id = wn.person_id
WHERE p.is_wanted = 1;
```

Explanation:
TThis query retrieves the first name, last name, and the date the wanted notice was issued for all individuals who are marked as "wanted." It joins the persons table with the wanted_notices table based on the person_id.

3.Find all persons who have been reported in more than one location:

```
SELECT person_id, COUNT(DISTINCT location) AS location_count
FROM sightings
GROUP BY person_id
HAVING COUNT (DISTINCT location) > 1;
```

Explanation:
This query identifies individuals who have been sighted in more than one distinct location. It groups the records by person_id and counts the distinct locations. The HAVING clause filters those with more than one location.

4.List the full details of all persons who have aliases:

```
SELECT p.*, a.alias_first_name, a.alias_last_name
FROM Persons p
JOIN aliases a ON p.person_id = a.person_id;
```

Explanation:
This query retrieves all columns from the persons table along with the alias first and last names for those individuals who have aliases. It joins the persons table with the aliases table on the person_id.

5.Find all sightings of a specific person by their person_id:

```
SELECT s.*
FROM sightings s
WHERE PERSON_ID = 3 ;
```

EXPLANATION:
This query retrieves all sightings related to a specific person by their person_id. Replace 123 with the actual person_id you want to look up.

RUSLAN BORISEVICH

5 HARD QUERIES YOU CAN USE WITH THE PUBLIC SAFETY DATABASE:

1. Find all persons with criminal records:

```
SELECT
    p.first_name,
    p.last_name,
    COUNT(cr.record_id) AS offense_count,
    AVG(LENGTH(cr.sentence_details)) AS average_sentence_length
FROM persons p
JOIN criminal_records cr ON p.person_id = cr.person_id
GROUP BY p.person_id, p.first_name, p.last_name
HAVING COUNT(cr.record_id) > 1;
```

Explanation:
This query finds individuals who have committed more than one offense. It counts the number of offenses and calculates the average length of the sentence details (if available). The HAVING clause ensures only those with multiple offenses are included.

2. Identify persons who have been sighted in more than two locations within the last 30 days:

```
WITH recent_sightings AS (
    SELECT person_id, COUNT(DISTINCT location) AS location_count
    FROM sightings
    WHERE date_reported >=  CURRENT_DATE - INTERVAL '30'  DAY
    GROUP BY person_id
)
SELECT p.first_name, p.last_name, rs.location_count
FROM recent_sightings rs
JOIN persons p ON rs.person_id = p.person_id
WHERE rs.location_count > 2;
```

Explanation:
This query identifies individuals who have been sighted in more than two distinct locations within the last 30 days. It uses a Common Table Expression (CTE) to first calculate the number of locations for each person and then joins it with the persons table.

RUSLAN BORISEVICH

5 HARD QUERIES YOU CAN USE WITH THE PUBLIC SAFETY DATABASE:

3. List all wanted persons along with their most recent sighting location and date:

```
SELECT p.first_name, p.last_name, s.location, s.date_reported
FROM persons p
JOIN sightings s ON p.person_id = s.person_id
WHERE p.is_wanted = 1
AND s.date_reported = (
    SELECT MAX(date_reported)
    FROM sightings
    WHERE person_id = p.person_id
);
```

```
SELECT p.first_name, p.last_name, rs.location_count
FROM recent_sightings rs
JOIN persons p ON rs.person_id = p.person_id
WHERE rs.location_count > 2;
```

Explanation:
This query retrieves the most recent sighting details (location and date) for all wanted individuals. It joins the persons table with the sightings table and uses a subquery to find the latest sighting date for each person.

4.Calculate the total reward amount for all wanted persons by nationality:

```
SELECT p.nationality, SUM(wn.reward_amount) AS total_reward
FROM persons p
JOIN wanted_notices wn ON p.person_id = wn.person_id
GROUP BY p.nationality
ORDER BY total_reward DESC;
```

Explanation:
This query calculates the total reward amount for all wanted individuals, grouped by their nationality. It sums up the reward amounts from the wanted_notices table and groups the results by the nationality column.

RUSLAN BORISEVICH

5 HARD QUERIES YOU CAN USE WITH THE PUBLIC SAFETY DATABASE:

5. Find persons who have been both sighted and have a criminal record, listing the sighting location and offense type:

```
SELECT
    p.first_name,
    p.last_name,
    s.location,
    cr.offense_type
FROM persons p
JOIN sightings s ON p.person_id = s.person_id
JOIN criminal_records cr ON p.person_id = cr.person_id;
```

Explanation:
This query retrieves the first name, last name, sighting location, and offense type for individuals who have both been sighted and have a criminal record. It joins the persons table with both the sightings and criminal_records tables on the person_id.n.

RUSLAN BORISEVICH

Conclusion

This guide has walked you through the creation and implementation of a Public Safety Database using SQL. From initial setup to complex queries.

Found this guide helpful? I'd love to hear from you!

RUSLAN BORISEVICH