Laura Malaeb: 202404420

Maria Khalife: 202402216

Sana Bitar: 202405418

Department of Arts and Sciences, Lebanese American University

Khaleel Mershad

CSC375: Database Management Systems

Section 13

December 7, 2025

Table of Contents

INTRODUCTION

Forensic investigation is a highly specialized field that demands accuracy, accountability, and transparency in the handling of criminal cases. Each investigation generates an immense amount of information — including case details, evidence records, laboratory analyses, and legal documents — all of which must be properly stored, organized, and retrievable for legal and procedural purposes. Traditional paper-based or fragmented digital systems often lead to inefficiencies, data loss, and difficulties in maintaining the chain of custody, which is essential for preserving the integrity of evidence in court. As forensic investigations become increasingly data-driven, the need for a centralized and structured database has become indispensable for ensuring the reliability and validity of criminal justice processes.

By applying core database management principles such as entity-relationship modeling, and referential integrity, this system will serve as a strong foundation for a future

implementation that could be used by forensic laboratories, police departments, or private investigative agencies. In the long run, the Forensic Investigation Case Management System objective is to modernize the management of forensic operations by developing efficiency, accuracy, and traceability in handling sensitive investigative data.

A Forensic Investigation Case Management System uses database management to organize all parts of an investigation. It helps keep track of entities such as evidence, personnel, warrants, and lab results. By sorting this data, it ensures accuracy, traceability, and accountability throughout the investigative process. The database makes collaboration between investigators and analysts more efficient, reduces the possibility for human error, and ensures safe access to critical information. It also allows efficient querying and reporting, aiding forensic departments in monitoring performance, managing workloads, and making decisions.

REQUIREMENTS

Phase 1:

The Forensic Investigation Case Management System is designed to support the rigorous, secure, and traceable handling of criminal investigations by centralizing all relevant data related to cases, personnel, evidence, chain-of-custody, lab analyses, warrants, and procedural events. The primary purpose of the system is to ensure legal integrity, accountability, and timely processing of cases through the structured and auditable documentation of all related activities. The database must support detailed recording and querying of sensitive data with a focus on traceability, access control, and analytical operations.
The core of the database revolves around the Case entity. Each case is uniquely identified by a Case_ID and stores key information including Crime_Type (e.g., homicide, robbery, cybercrime), Status (open, closed, pending), the assigned Investigation_Team_ID, and the Open_Date and Close_Date which mark the investigation's timeline. This entity serves as the primary anchor for all other forensic operations.
Closely linked to the case are one or more Evidence_Items. Each piece of evidence is recorded with a unique Evidence_ID, linked to a Case_ID, and includes metadata such as the Evidence_Type (e.g., weapon, digital file, biological sample), Collection_Date, and Storage_Location. This entity ensures traceability and categorization of physical or digital items obtained during the investigation.
To maintain legal validity, evidence must be accurately tracked via a Chain_of_Custody entity. This entity logs every interaction with an evidence item, where each entry is identified by a Custody_ID and includes the Evidence_ID, Handler_ID (linked to personnel), Timestamp, and Transfer_Reason (e.g., sent to lab, stored, retrieved for court). This chain ensures that the integrity of evidence is never compromised and every transfer is auditable.

The Personnel entity stores information on all individuals involved in the investigation process including detectives, forensic analysts, administrative staff, and lab technicians. Each person is assigned a Personnel_ID and is described by their Name, Role, Department, Clearance_Level, and contact information. This entity allows role-based data access and accountability for all system operations.
For each piece of evidence analyzed in the lab, the system captures the results in the Lab_Analysis entity. This includes a unique Analysis_ID, the associated Evidence_ID, the Analyst_ID (linked to Personnel), the Analysis_Date, Technique_Used (e.g., DNA sequencing, fingerprint matching), and Findings. These results are critical in guiding investigative decisions and legal proceedings.
Another essential component of criminal investigations is the use of Warrants, particularly in gathering evidence or performing arrests. The Warrant entity tracks each legal authorization with a Warrant_ID, the associated Case_ID, Warrant_Type (search, arrest, surveillance), Issue_Date, Expiration_Date, and the issuing Judge_Name. This entity ensures all actions taken under legal authority are well-documented and within their time constraints.
A flexible entity called Case_Event logs procedural and contextual activities that occur throughout the investigation. Each event is recorded with a Case_Event_ID, linked to a Case_ID, and includes the Event_Type (interrogation, search, court filing), the Event_Date, involved Personnel_ID, and any optional Notes. This entity provides a chronological audit trail of all non-evidence procedural activities.
The relationships between entities enforce referential integrity and enable deep analytics. Each Case can be linked to multiple Evidence_Items, Warrants, Case_Events, and Lab_Analyses (through evidence). Evidence_Items are associated with multiple Chain_of_Custody entries, reflecting the handling history of each item. The Personnel entity connects to various roles throughout the system, acting as handlers, analysts, warrant requestors, or participants in case events.

The many-to-one relationship between Evidence_Item and Case allows tracking of all evidence per case, while the many-to-many relationship between Evidence_Item and Personnel (via Chain_of_Custody and Lab_Analysis) ensures accountability in evidence handling and processing. Similarly, the one-to-many relationship between Case and Warrant allows each investigation to track legal permissions involved in the process.

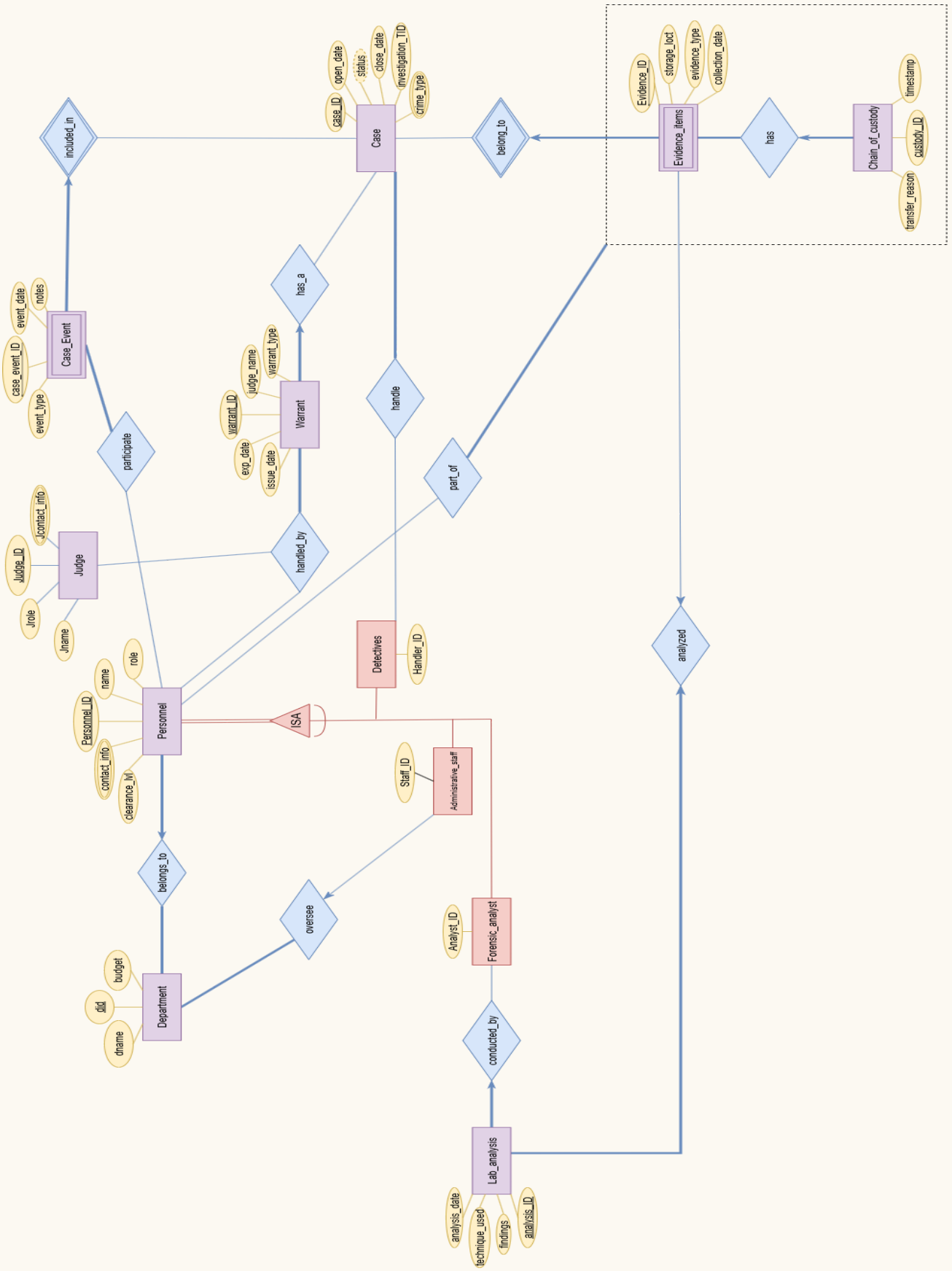
To support investigation workflows, the database must enable a variety of operations:

1. Case Creation and Assignment: Create a new case, assign it to an investigative team, and track its status changes from open to closed.
2. Evidence Entry and Association: Add new evidence entries and associate them with specific cases. Capture metadata about the item and initiate its chain-of-custody log.
3. Custody Tracking: Add custody log entries whenever evidence changes hands, including timestamp, personnel, and reason.
4. Laboratory Analysis Logging: Record new lab analysis records, ensuring evidence is linked to results and analysts.
5. Warrant Issuance: Log issuance of search or arrest warrants, linked to their corresponding cases and details.
6. Case Event Logging: Add records for every interrogation, court appearance, or other procedural step within the case.
7. Querying Evidence History: Retrieve the full custody history of an evidence item to validate its integrity in court.
8. Performance and Backlog Analytics: Generate reports showing average investigation times, unresolved cases, evidence backlog, lab turnaround time, and evidence handled per personnel.
9. Access Control and Audit Trail: Enforce role-based access to data and log system usage to ensure data protection and accountability.

In conclusion, the forensic case management system requires a carefully structured relational database that enables end-to-end tracking of criminal investigations. It must support robust data integrity and allow investigators,

analysts, and administrators to interact with a centralized source of trusted data while maintaining a clear and auditable chain of evidence and procedural actions. By providing a comprehensive framework for handling data from initial investigation to court proceedings, the database plays a pivotal role in enhancing transparency, effectiveness, and legal compliance in forensic operations.

ER-MODEL


Case
case_ID
open_date
status
close_date
investigation_TID
crime_type
included_in
belong_to
Evidence_items
Evidence_ID
storage_loct
evidence_type
collection_date
has
Chain_of_custody
timestamp
custody_ID
transfer_reason
has_a
Case_Event
event_date
notes
case_event_ID
event_type
participate
Warrant
warrant_type
judge_name
warrant_ID
exp_date
issue_date
handle
part_of
handled_by
Judge
Jcontact_info
Judge_ID
Jrole
Jname
Detectives
Handler_ID
analyzed
Personnel
role
name
Personnel_ID
contact_info
clearance_lvl
ISA
Staff_ID
Administrative_staff
belongs_to
oversee
Forensic_analyst
Analyst_ID
Department
budget
did
dname
conducted_by
Lab_analysis
analysis_date
technique_used
findings
analysis_ID

DESCRIPTION

- **Department:** An entity characterized by the attributes “dname” (department name), “budget” and the primary key “did” (department ID).
- **Personnel:** Parent entity of the ISA relationship with the attributes “name”, “role”, “clearance level”, “contact info”, “start_date”, “end_date” and the primary key “personnel ID”.
  The attribute “contact info” is multi-valued since a Personnel can have multiple contact information.
  The ISA relationship is covering since a Personnel in the Forensic Investigation Case Management System can only be one of the inheriting entities.

The entities inheriting from Parent are:

- Detectives: An entity characterized by the same attributes and primary key of Personnel in addition to the “handler ID” which identifies them as the requesters of warrants and the handlers of cases.
- Forensic_Analyst: An entity characterized by the same attributes and primary key of Personnel in addition to the “analyst ID” which identifies them as the ones handling the lab analysis.
- Administrative_Staff: An entity characterized by the same attributes and primary key of Personnel in addition to “Staff_ID” which identifies them as the ones overseeing the department.

None of the Personnel can have more than one defining role, therefore the ISA relationship is not overlapping.

- **Case :** An entity characterized by the attributes “open date”, “close date”, “investigation team ID”, “status”, “crime type” and the primary key “case ID”.
  The attribute “status” is derived (*pending, open, closed…*) through the attributes “open date” and “close date”. If the “close date” is NULL then the case is *open*, *pending* if “open date” and “close date” are NULL, otherwise the case is *closed*.

The attribute “crime type” is multi_valued since a single case can include multiple crime types at once.

- **Warrant:** An entity characterized by the attributes ”judge name”, “warrant type”, “issue date”, “expiration date” and the primary key “warrant ID”.
- **Judge:** An entity characterized by the attributes “Judge_ID” which is the primary key and identifies them as handlers of warrants, “Jname” (Judge name), “Jrole” (Judge role) and “Jcontact_info” (Judge’s contact info, multivalued attribute).
- **Case_Event:** A weak entity of the owner entity “Case”, characterized by the attributes “event type”, “event date”, “notes” and the partial key “case event ID”.
- **Evidence_Items:** A weak entity of the owner entity “Case”, characterized by the attributes “storage location”, “evidence type”, “collection date”, and the partial key “evidence ID”.
- **Chain_of_Custody:** An entity characterized by the attributes “transfer reason”, “timestamp” and the primary key “custody ID”.
- **Lab_Analysis:** An entity characterized by the attributes “findings”, “technique used”, “analysis date” and the primary key “analysis ID”.
- **Handle:** A many-to-many binary relationship between “Detectives” and “Case” where “Case” has a participation constraint (a case needs to be handled by at least a single detective).
- **Participate:** A many-to-many binary relationship between “Case Event” and “Personnel” where “case_event” has a participation constraint (a case_event needs the participation of at least one personnel).
- **Belong_to:** A one-to-many identifying relationship between “Case” and “Evidence items”, where “Evidence items” has a participation and a key constraint.
- **Included_in:** A one-to-many identifying relationship between “Case” and “Case Event”, where “Case Event” has a participation and a key constraint.

- **Has:** A one-to-many binary relationship between "Evidence items" and "Chain of Custody", where "Evidence items" has a participation constraint (every evidence item must have at least one chain of custody) and "Chain of Custody" has both key and participation constraint (every chain of custody must belong to one and only one evidence item).
- **Has_a:** A one-to-many relationship between "Case" and "Warrant", where "Warrant" has a participation and a key constraint. (a warrant must belong to one and only one case at a time).
- **Belongs_to:** A one-to-many binary relationship between "Department" and "Personnel", where "Department" has a participation constraint (every department must have at least one personnel) and "Personnel" has both key and participation constraint (every personnel must belong to one and only one department).
- **Oversee:** A one-to-many relationship between "Department" and "Administrative staff", where "Departments" has a participation constraint (at least one administrator must oversee a department) and "Administrative staff" has a key constraint (an administrator can only oversee a single department at a time.)
- **Conducted_by:** A one-to-many relationship between "Forensic Analyst" and "Lab Analysis", where "Lab Analysis" has both participation and key constraints. (a lab analysis must be conducted by one and only one forensic analyst at a time).
- **Part_of:** Is a many-to-many aggregation relationship between "Personnel" and the relationship "Has", where "Has" has a participation constraint (the "Has" relationship must be performed by at least one personnel). Since the relationship between "Evidence items" and "Chain of Custody" (Has) is reliant on "Personnel" then it is an aggregation.
- **Analyzed:** Is a one-to-one relationship between "Evidence Items" and "Lab_analysis" where

"Lab_analysis" has a participation and a key constraint (the "Lab_analysis" must be performed for one and only one evidence item) and "Evidence Items" has a key constraint (An evidence item can be analyzed only once).

- **Handled_by:** A many-to-many-to-many ternary relationship between "Judge", "Warrant" and "Personnel", where "Warrant" has a participation constraint (A warrant needs to be approved by at least one judge and requested by at least one personnel). A ternary was chosen for this relationship because all these entities are needed simultaneously for the handling of a warrant.

RELATIONAL MODEL

Department (did, dname, budget)
Case (case_ID, open_date, close_date, investigation_TID)
Crime_type (#case_ID, crime_type)
Case_Event (#case_ID, case_event_ID, event_type, event_date, notes)
Warrant (warrant_ID, warrant_type, issue_date, exp_date, #case_ID) (not null)
Chain_of_custody (custody_ID, transfer_reason, timestamp)
Evidence_items (#case_ID, evidence_ID, storage_loct, evidence_type, collection_date)
Lab_analysis (analysis_ID, findings, technique_used, analysis_date, #Personnel_ID, #Case_ID, #Evidence_ID) (not null)
Personnel (personnel_ID, name, role, clearance_lvl, start_date, end_date, #did) (not null)
Contact_info (#personnel_ID, contact_info)
Detectives (#personnel_ID, Handler_ID)
Administrative_staff (#personnel_ID, Staff_ID, #did)
Forensic_analyst (#personnel_ID, analyst_ID)
Judge (Judge_ID, Jname, Jrole)
Jcontact_info(#Judge_name, contact_info)
Handled_by (#personnel_ID, #warrant_ID, #Judge)
Handle (#personnel_ID, #case_ID)
Participate (#personnel_ID, #case_ID, #case_event_ID)
Has (#custody_ID, #case_ID, #evidence_ID)
Part_of (#personnel_ID, #custody_ID)

DATABASE DESCRIPTION

1. Database Creation

```
1 •  CREATE DATABASE Forensic_Investigation_Database;
```

- Creates a new database named Forensic_Investigation_Database
- Serves as the container for all forensic investigation tables, queries, views, functions and procedures.

2. DDL: Table Creation with Constraints

Table 1: Case_Record

Stores main case information

```
3  CREATE TABLE Case_Record(
4      case_ID INT PRIMARY KEY AUTO_INCREMENT,
5      open_date DATE,
6      close_date DATE,
7      investigation_TID INT NOT NULL,
8      case_status VARCHAR(10) AS (
9          CASE
10             WHEN open_date IS NULL AND close_date IS NULL THEN 'Pending'
11             WHEN open_date IS NOT NULL AND close_date IS NULL THEN 'Open'
12             WHEN open_date IS NOT NULL AND close_date IS NOT NULL THEN 'Closed'
13         END) STORED
14 );
```

Constraints:

- PRIMARY KEY: case_ID uniquely identifies each case
- AUTO_INCREMENT: Automatically generates sequential case IDs
- NOT NULL: investigation_TID is mandatory
- GENERATED COLUMN: case_status automatically calculated from dates
- STORED: Case status physically stored for better query performance

Table 2: Crime_type

Associates crime types with cases (many-to-many relationship)

```
17 CREATE TABLE Crime_type(
18     crime_type varchar(20),
19     case_ID INT,
20     PRIMARY KEY(case_ID, crime_type),
21     FOREIGN KEY(case_ID) REFERENCES Case_Record(case_ID) ON DELETE CASCADE ON UPDATE CASCADE
22 );
```

Constraints:

- COMPOSITE PRIMARY KEY: Combination of case_ID and crime_type must be unique
- FOREIGN KEY: References Case_Record table
- ON DELETE CASCADE: Automatically removes crime types when parent case is deleted
- ON UPDATE CASCADE: Updates crime type records if case_ID changes

Table 3: Department

Stores department/organizational unit information

```
24  CREATE TABLE Department(
25      did INT PRIMARY KEY AUTO_INCREMENT,
26      dname varchar(30) NOT NULL UNIQUE,
27      budget INT
28      CHECK(budget>0)
29  );
```

Constraints:

- NOT NULL: Department name required
- UNIQUE: Department names must be unique across all departments
- CHECK: Budget must be greater than 0 (positive value)
- AUTO_INCREMENT: Department IDs automatically generated

Table 4: Warrant

Stored legal warrant information

```
CREATE TABLE Warrant(
    warrant_ID int auto_increment,
    warrant_type varchar(15) NOT NULL,
    issue_date date NOT NULL,
    exp_date date NOT NULL,
    case_ID int NOT NULL,
    primary key (warrant_ID),
    foreign key (case_ID) references Case_Record(case_ID)
        on update cascade
        on delete cascade,
    CHECK (
        warrant_type IN (
            'Search', 'Arrest', 'Seizure', 'Detention', 'Surveillance',
            'Tracking', 'Wiretap', 'Subpoena', 'Device Search', 'Data Access'
        )
    )
);
```

Constraints:

- AUTO_INCREMENT: Warrant IDs automatically generated after every insert
- NOT NULL: All warrant details required (issue_date, exp_date, case_ID)
- FOREIGN KEY: Links to Case_Record table
- CASCADE: Updates/deletes propagate to related records
- CHECK: Ensures only valid warrant types are used (domain integrity)

Table 5: Case_Event

Tracks investigative events with cases

```
create table Case_Event(
    case_event_ID int auto_increment,
    event_type varchar(25) not null,
    event_date date not null,
    notes varchar(100),
    case_ID int,
    primary key (case_event_ID, case_ID),
    foreign key (case_ID) references Case_Record(case_ID)
        on update cascade
        on delete cascade,
    check (event_type IN ('interrogation', 'search', 'court filing',
    'Suspect handling' ,'Witness interview', 'Scene search', 'Evidence collection',
    'Surveillance operation', 'Area canvass', 'Stakeout operation', 'Lead update',
    'Tip received'))
    );
```

Constraints:

- AUTO_INCREMENT: case_event IDs automatically generated after each insert
- COMPOSITE PRIMARY KEY: Both event ID and case ID together are primary keys
- CASCADE: Full cascading on updates/deletes
- CHECK: Ensures only predefined event types are used
- NOT NULL: Event type and date required

Table 6: Evidence_items

Stored physical/digital evidence information

```
create table Evidence_items(
    case_ID int,
    evidence_ID int auto_increment,
    storage_loct varchar(50),
    evidence_type varchar(20) not null,
    collection_date date not null,
    primary key (evidence_ID, case_ID),
    foreign key (case_ID) references Case_Record(case_ID)
        on update cascade
        on delete cascade,
    check (evidence_type IN ('weapon', 'digital file', 'biological sample','physical','chemical'))
    );
```

Constraints:

- AUTO_INCREMENT: evidence IDs automatically generated after each insert

- COMPOSITE PRIMARY KEY: Evidence uniquely identified within each case
- CASCADE: Evidence deleted if case is deleted
- CHECK: Only allowed evidence types
- NOT NULL: Evidence type and collection date required

Table 7: Chain_of_custody

Tracks evidence custody transfers

```
78  CREATE TABLE Chain_Of_Custody(
79      custody_ID INT PRIMARY KEY AUTO_INCREMENT,
80      custody_timestamp TIME NOT NULL,
81      transfer_reason varchar(35) NOT NULL,
82      CHECK(transfer_reason IN ('sent to lab', 'stored', 'retrieved for court', 'archived',
83      'destroyed', 'transferred to another department'))
84      );
```

Constraints:

- AUTO_INCREMENT: custody IDs automatically generated after each insert
- NOT NULL: Timestamp and reason required
- CHECK: Ensures valid transfer reasons
- PRIMARY KEY: Unique custody record identifier

Table 8: Personnel

Stores information about investigation personnel

```
86  CREATE TABLE Personnel(
87      personnel_ID INT PRIMARY KEY AUTO_INCREMENT,
88      p_name varchar(20) NOT NULL,
89      p_role varchar(20),
90      start_date date NOT NULL,
91      end_date date,
92      clearance_lvl INT DEFAULT 4,
93      did INT NOT NULL,
94      FOREIGN KEY(did) REFERENCES Department(did) ON DELETE RESTRICT ON UPDATE CASCADE,
95      CHECK(clearance_lvl >= 1 AND clearance_lvl <= 4)
96  );
```

Constraints:

- AUTO_INCREMENT: personnel IDs automatically generated after each insert
- NOT NULL: Name, start date, and department required
- DEFAULT: Clearance level defaults to 4 if not specified
- CHECK: Clearance must be between 1 (highest) and 4 (lowest)
- FOREIGN KEY: Links to Department table
- ON DELETE RESTRICT: Prevents department deletion if personnel assigned (can't delete a department if it still has personnel working in it)
- ON UPDATE CASCADE: Updates personnel if department ID changes

Table 9: Lab_analysis

Records forensic laboratory analysis results

```
create table Lab_analysis(
    analysis_ID int auto_increment primary key,
    findings varchar(100),
    technique_used varchar(30),
    analysis_date date not null,
    personnel_ID int not null,
    case_ID int not null,
    evidence_ID int not null,
    foreign key (case_ID) references Case_Record(case_ID)
        on update cascade
        on delete cascade,
    foreign key (evidence_ID) references Evidence_items(evidence_ID)
        on update cascade
        on delete cascade,
    foreign key (personnel_ID) references Personnel(personnel_ID)
        on update cascade
        on delete restrict,
    unique(personnel_ID, evidence_ID, case_ID),
    check(technique_used IN ('DNA profiling', 'Blood typing', 'Fingerprint comparison',
    'Footprint analysis', 'Drug testing', 'Bullet comparison', 'Firearm examination',
    'Toxicology test', 'Explosive residue', 'Gunshot residue', 'Computer forensics',
    'Network analysis', 'Trace detection', 'Evidence preservation', 'Scene reconstruction'))
    );
```

Constraints:

- AUTO_INCREMENT: Lab_analysis IDs automatically generated after each insert
- UNIQUE: Prevents duplicate analyses by same personnel on same evidence
- NOT NULL: Analysis date, case, and evidence required
- FOREIGN KEY: Links to multiple parent tables (case_record, evidence_items, personnel)
- CASCADE: When update and delete case_record and evidence_items, change automatically in lab_analysis table
- RESTRICT: When deleting a personnel restrict the deletion because can't lose the record of personnel that has worked on a lab analysis
- CHECK: Validates forensic techniques

Specialized Personnel Tables

Table 10 : Detectives

Specialized table with investigative detectives (ISA hierarchy with Personnel)

```
122 • create table Detectives(
123       handler_ID int auto_increment,
124       personnel_ID int,
125       foreign key (personnel_ID) references Personnel(personnel_ID)
126           on update cascade
127           on delete cascade,
128       primary key (personnel_ID),
129       unique(handler_ID)
130       );
```

Constraints:

- FOREIGN KEY: Links to Personnel table, establishing ISA hierarchy (Detective IS-A Personnel)
- CASCADE on UPDATE: Updates detective records if personnel_ID changes
- CASCADE on DELETE: Deletes detective records if corresponding personnel deleted
- PRIMARY KEY: Enforces one-to-one relationship (each personnel can be only one detective)
- UNIQUE: Ensures each handler_ID is unique across all detectives
- AUTO_INCREMENT: Automatically generates sequential handler IDs

Table 11: Forensic_analyst

Specialized table for forensic laboratory analysts (ISA hierarchy with Personnel)

```
139 • CREATE TABLE Administrative_staff(
140       personnel_ID INT PRIMARY KEY,
141       staff_ID INT UNIQUE AUTO_INCREMENT,
142       oversee_did INT,
143       FOREIGN KEY(personnel_ID) REFERENCES Personnel(personnel_ID) ON UPDATE CASCADE ON DELETE CASCADE,
144       FOREIGN KEY(oversee_did) REFERENCES Department(did) ON DELETE RESTRICT ON UPDATE CASCADE
145   );
```

Constraints:

- PRIMARY KEY: Enforces one-to-one relationship with Personnel table
- FOREIGN KEY: Links to Personnel table (Forensic_analyst IS-A Personnel)
- CASCADE on UPDATE: Updates analyst records if personnel_ID changes
- CASCADE on DELETE: Deletes analyst records if personnel deleted
- UNIQUE: Ensures each analyst_ID is unique across all analysts
- AUTO_INCREMENT: Automatically generates sequential analyst IDs

Table 12: Administrative_staff

Specialized table for administrative personnel with department oversight (ISA hierarchy with Personnel)

```
139 • CREATE TABLE Administrative_staff(
140       personnel_ID INT PRIMARY KEY,
141       staff_ID INT UNIQUE AUTO_INCREMENT,
142       oversee_did INT,
143       FOREIGN KEY(personnel_ID) REFERENCES Personnel(personnel_ID) ON UPDATE CASCADE ON DELETE CASCADE,
144       FOREIGN KEY(oversee_did) REFERENCES Department(did) ON DELETE RESTRICT ON UPDATE CASCADE
145   );
```

Constraints:

- PRIMARY KEY: Enforces one-to-one relationship with Personnel
- UNIQUE: Each staff_ID must be unique
- AUTO_INCREMENT: Automatically generates sequential staff IDs
- FOREIGN KEY (personnel_ID): Links to Personnel table (Administrative_staff IS-A Personnel)
- CASCADE on UPDATE/DELETE: Full cascading for personnel relationship
- FOREIGN KEY (oversee_did): Links to Department table for oversight responsibility
- RESTRICT on DELETE: Prevents department deletion if administrative staff oversees it
- CASCADE on UPDATE: Updates oversee_did if department ID changes

Table 13: Contact_info

Stores contact information for personnel because it is a multivalued attribute

```
CREATE TABLE Contact_Info(
    contact_info VARCHAR(100),
    personnel_ID INT,
    PRIMARY KEY(contact_info, personnel_ID),
    FOREIGN KEY(personnel_ID) REFERENCES Personnel(personnel_ID) ON DELETE CASCADE ON UPDATE CASCADE
);
```

Constraints:

- PRIMARY KEY: Composite key ensures unique contact methods per personnel
- FOREIGN KEY: Links to Personnel table
- CASCADE on DELETE: Deletes contact information when personnel is deleted
- CASCADE on UPDATE: Updates contact records if personnel_ID changes

Table 14: Judge

Stores Judicial officer information (independent entity)

```
CREATE TABLE Judge(
    judge_ID int auto_increment,
    jname varchar(15) NOT NULL,
    jrole varchar(15) NOT NULL,
    primary key (judge_ID),
    CHECK (jrole IN ('Chief Judge', 'Trial Judge','Magistrate', 'Justice', 'Appellate', 'Senior Judge'))
);
```

Constraints:

- PRIMARY KEY: judge_ID uniquely identifies each judge
- AUTO_INCREMENT: Automatically generates sequential judge IDs after each insert
- NOT NULL: Both judge name and role are required fields
- CHECK: Ensures only valid judicial roles are used (domain integrity)

Table 15: Jcontact_Info

Stores contact information for judges because it is a multivalued attribute

```
165 ● ⊖ CREATE TABLE Jcontact_Info(
166       contact_info varchar(100) NOT NULL,
167       judge_ID int NOT NULL,
168       primary key(judge_ID, contact_info),
169       foreign key(judge_ID) references Judge(judge_ID)
170           on update cascade
171           on delete cascade
172   );
```

Constraints:

- PRIMARY KEY: Composite key ensures unique contact methods per judge
- NOT NULL: Both contact info and judge_ID are required
- FOREIGN KEY: Links to Judge table
- CASCADE on UPDATE: Updates contact records if judge_ID changes
- CASCADE on DELETE: Deletes contact information when judge is deleted

Relationship Tables

Table 16: Has

Links Evidence_items to Chain_of_custody

```
175 ● ⊖ CREATE TABLE Has(
176       custody_ID INT UNIQUE,
177       case_ID INT,
178       evidence_ID INT,
179       PRIMARY KEY(custody_ID, case_ID, evidence_ID),
180       FOREIGN KEY(custody_ID) REFERENCES Chain_Of_Custody(custody_ID) ON DELETE CASCADE ON UPDATE CASCADE,
181       FOREIGN KEY(case_ID) REFERENCES Case_Record(case_ID)ON DELETE CASCADE ON UPDATE CASCADE,
182       FOREIGN KEY(evidence_ID) REFERENCES Evidence_items(evidence_ID) ON DELETE CASCADE ON UPDATE CASCADE
183   );
```

Constraints:

- UNIQUE: Each custody_ID can only appear once in the table (one custody event per evidence)
- PRIMARY KEY: Composite key ensures unique combination of custody, case, and evidence
- FOREIGN KEY (custody_ID): Links to Chain_Of_Custody table

- FOREIGN KEY (case_ID): Links to Case_Record table
- FOREIGN KEY (evidence_ID): Links to Evidence_items table
- CASCADE on DELETE: Deletes relationship records if any parent record is deleted
- CASCADE on UPDATE: Updates relationship records if any parent ID changes

Table 17: Participate

Links Personnel to Case_events

```
CREATE TABLE Participate(
    personnel_ID INT,
    case_ID INT,
    case_event_ID INT,
    PRIMARY KEY(personnel_ID, case_ID, case_event_ID),
    FOREIGN KEY(personnel_ID) REFERENCES Personnel(personnel_ID) ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY(case_ID) REFERENCES Case_Record(case_ID)ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY(case_event_ID) REFERENCES Case_Event(case_event_ID) ON DELETE CASCADE ON UPDATE CASCADE
);
```

Constraints:

- PRIMARY KEY: Composite key prevents duplicate personnel-case-event assignments
- FOREIGN KEY (personnel_ID): Links to Personnel table
- FOREIGN KEY (case_ID): Links to Case_Record table
- FOREIGN KEY (case_event_ID): Links to Case_Event table
- CASCADE on DELETE: Deletes participation records if any parent record is deleted
- CASCADE on UPDATE: Updates participation records if any parent ID changes

Table 18: Handled_by

Links Warrant to Personnel and Judges (ternary relationship)

```
195 ● ⊖ create table Handled_by(
196       warrant_ID int,
197       personnel_ID int,
198       judge_ID int,
199       primary key (warrant_ID, personnel_ID, judge_ID),
200       foreign key (warrant_ID) references Warrant(warrant_ID)
201           on update cascade
202           on delete cascade,
203       foreign key (personnel_ID) references Personnel(personnel_ID)
204           on update cascade
205           on delete cascade,
206       foreign key (judge_ID) references Judge(judge_ID)
207           on update cascade
208           on delete cascade
209   );
```

Constraints:

- PRIMARY KEY: Composite key ensures unique warrant-personnel-judge combinations
- FOREIGN KEY (warrant_ID): Links to Warrant table
- FOREIGN KEY (personnel_ID): Links to Personnel table
- FOREIGN KEY (judge_ID): Links to Judge table
- CASCADE on DELETE: Deletes handling records if warrant, personnel, or judge is deleted
- CASCADE on UPDATE: Updates handling records if any parent ID changes

Table 19: Part_of

Links Personnel to Chain_of_custody

```
211 • create table Part_of(
212       personnel_ID int,
213       custody_ID int,
214       primary key(personnel_ID, custody_ID),
215       foreign key (personnel_ID) references Personnel(personnel_ID)
216           on update cascade
217           on delete cascade,
218       foreign key (custody_ID) references Chain_of_custody(custody_ID)
219           on update cascade
220           on delete cascade
221   );
```

Constraints:

- PRIMARY KEY: Composite key prevents duplicate personnel-custody assignments
- FOREIGN KEY (personnel_ID): Links to Personnel table
- FOREIGN KEY (custody_ID): Links to Chain_of_custody table
- CASCADE on DELETE: Deletes part_of records if personnel or custody record is deleted
- CASCADE on UPDATE: Updates part_of records if personnel_ID or custody_ID changes

Table 20: Handle

Links Personnel and Case_Record

```
223 • CREATE TABLE Handle(
224       personnel_ID INT,
225       case_ID INT,
226       primary key (personnel_ID, case_ID),
227       foreign key (personnel_ID) references Personnel(personnel_ID)
228           on update cascade
229           on delete cascade,
230       foreign key (case_ID) references Case_Record(case_ID)
231           on update cascade
232           on delete cascade
233   );
```

Constraints:

- PRIMARY KEY: Composite key prevents duplicate personnel-case assignments
- FOREIGN KEY (personnel_ID): Links to Personnel table
- FOREIGN KEY (case_ID): Links to Case_Record table
- CASCADE on DELETE: Deletes handle records if personnel or case is deleted
- CASCADE on UPDATE: Updates handle records if personnel_ID or case_ID changes

3. DML: Data Population Queries

Insertion Order:

1. Department → No dependencies
2. Case_Record → No dependencies
3. Personnel → Requires Department
4. Judge → No dependencies
5. Crime_type → Requires Case_Record
6. Warrant → Requires Case_Record
7. Case_Event → Requires Case_Record
8. Evidence_items → Requires Case_Record
9. Chain_of_Custody → No dependencies
10. Specialized Personnel → Requires Personnel
11. Contact Info → Requires Personnel/Judge
12. Relationship Tables → Requires ALL parent tables

Department

```
INSERT INTO Department (dname, budget) VALUES
('Cyber Crime Unit', 450000), -- did1
('Financial Investigations', 350000),
('Crime Scene Unit', 400000),
('Tactical Operations', NULL),
('Cold Case Unit', 250000),
('Internal Affairs', 275000),
('Community Relations', NULL),
('Training Division', 225000),
('Evidence Processing Center', 375000),
('Forensic Accounting', 325000),
('Surveillance Division', 360000);
```

Features:

- Inserts: 11 departments inserted with single query
- NULL Values: Some departments have NULL budgets
- AUTO_INCREMENT: did automatically assigned (1–11)

Case_Record

```
INSERT INTO Case_Record (open_date, close_date, investigation_TID) VALUES
(NULL, NULL, 1001),  -- Pending case cid1 (recent)
('2023-11-20', '2024-02-10', 1002),  -- Closed case cid2 (from last year)
(NULL, NULL, 1003),  -- Pending case cid3 (recent)
('2024-01-10', '2024-04-05', 1004),  -- Closed case cid4 (early 2024)
('2024-03-15', NULL, 1005),  -- Open case cid5 (ongoing from 2024)
('2024-04-28', '2024-08-15', 1006),  -- Closed case cid6 (mid 2024)
(NULL, NULL, 1007),  -- Pending case cid7 (recent)
('2024-06-12', '2024-09-01', 1008),  -- Closed case cid8 (summer 2024)
('2024-07-20', NULL, 1009),  -- Open case cid9 (ongoing from mid-2024)
(NULL, NULL, 1010),  -- Pending case cid10 (recent)
('2024-08-10', NULL, 1011),  -- Open case cid11 (ongoing from late 2024)
('2024-08-15', '2024-12-20', 1012),  -- Closed case cid12 (late 2024)
(NULL, NULL, 1013),  -- Pending case cid13 (recent)
('2024-09-03', '2025-01-15', 1014),  -- Closed case cid14 (recently closed)
('2024-10-10', NULL, 1015),  -- Open case cid15 (ongoing from late 2024)
(NULL, NULL, 1016),  -- Pending case cid16 (recent)
('2024-11-25', NULL, 1017),  -- Open case cid17 (ongoing from late 2024)
('2024-12-01', '2025-03-30', 1018),  -- Closed case cid18 (recently closed)
(NULL, NULL, 1019),  -- Pending case cid19 (recent)
('2025-01-15', '2025-05-20', 1020),  -- Closed case cid20 (closed earlier this year)
```

```
('2025-02-20', NULL, 1021),  -- Open case cid21 (current ongoing)
(NULL, NULL, 1022),  -- Pending case cid22 (recent)
('2025-03-01', '2025-06-10', 1023),  -- Closed case cid23 (recently closed)
('2025-03-05', NULL, 1024),  -- Open case cid24 (current ongoing)
(NULL, NULL, 1025); -- Pending case cid25 (recent)
```

Features:

- Temporal Data: Mix of historical (2023) and current (2025) cases
- NULL Handling:
    - NULL, NULL = Pending case (case_status auto-calculated as 'Pending')
    - DATE, NULL = Open case (case_status = 'Open')
    - DATE, DATE = Closed case (case_status = 'Closed')
- GENERATED COLUMN: case_status automatically calculated from dates

Crime_type

```
INSERT INTO Crime_type (crime_type, case_ID) VALUES
('assault', 5), ('assault', 9), ('assault', 11), ('assault', 15), ('assault', 21), ('assault', 2), ('assault', 4),
 ('burglary', 5), ('burglary', 11), ('burglary', 17), ('burglary', 24), ('burglary', 6), ('burglary', 12),
 ('cybercrime', 5), ('cybercrime', 9),  ('cybercrime', 11), ('cybercrime', 15), ('cybercrime', 17), ('cybercrime', 24),
 ('cybercrime', 2), ('cybercrime', 18), ('fraud', 9), ('fraud', 15), ('fraud', 21), ('fraud', 8), ('fraud', 14),
('homicide', 17), ('homicide', 24), ('robbery', 11), ('robbery', 21), ('drug trafficking', 15), ('drug trafficking', 24),
('homicide', 1), ('assault', 1), ('burglary', 2), ('fraud', 3), ('cybercrime', 4), ('drug trafficking', 5), ('robbery', 6),
('forgery', 7), ('kidnapping', 8), ('arson', 9), ('theft', 10), ('vandalism', 11), ('extortion', 12), ('money laundering', 13),
('identity theft', 14), ('homicide', 15),('burglary', 16), ('fraud', 17), ('assault', 18), ('drug trafficking', 19),
('robbery', 20), ('forgery', 21), ('kidnapping', 22), ('arson', 23), ('theft', 24), ('vandalism', 25);
```

Features:

- Many-to-Many: Cases can have multiple crime types
- Referential Integrity: All case_ID values exist in Case_Record
- Composite Primary Key: Ensures unique case-crime combinations

Personnel

```
INSERT INTO Personnel (p_name, p_role, start_date, end_date, clearance_lvl, did) VALUES
('John Smith', 'Lead Detective', '2020-03-15', NULL, 1, 1),          -- pid 1 - Cyber Crime Unit
('Alex Morgan', 'Cyber Investigator', '2021-06-22', NULL, 2, 1),     -- pid 2 - Cyber Crime Unit
('Thomas King', 'Digital Forensics', '2022-01-10', NULL, 3, 1),      -- pid 3 - Cyber Crime Unit
('Robert Chen', 'Detective', '2019-11-05', NULL, 2, 2),              -- pid 4 - Financial Investigations
('Ryan Cooper', 'Financial Analyst', '2020-08-30', NULL, 2, 2),      -- pid 5 - Financial Investigations
('Liam Wilson', 'Forensic Accountant', '2021-02-14', NULL, 2, 2),    -- pid 6 - Financial Investigations
('Sophia Lee', 'CSI', '2020-05-18', NULL, 2, 3),                     -- pid 7 - Crime Scene Unit
('Sarah Johnson', 'Evidence Clerk', '2022-03-22', NULL, 3, 3),       -- pid 8 - Crime Scene Unit
('Chloe Martinez', 'Evidence Processor', '2021-09-11', '2024-10-01', 3, 3),   -- pid 9 - Crime Scene Unit (has end_date)
('Marcus Johnson', 'Tactical Commander', '2018-07-09', NULL, 1, 4),  -- pid 10 - Tactical Operations
('Emily Davis', 'Detective', '2019-04-25', NULL, 2, 4),              -- pid 11 - Tactical Operations
('Patricia Clark', 'Detective', '2020-12-01', NULL, 2, 4),           -- pid 12 - Tactical Operations
('Olivia Chen', 'Cold Case Detective', '2017-11-20', NULL, 2, 5),    -- pid 13 - Cold Case Unit
('Brian Anderson', 'Detective', '2019-08-14', NULL, 2, 5),           -- pid 14 - Cold Case Unit
('Nicole Young', 'Detective', '2021-01-30', '2024-06-15', 2, 5),     -- pid 15 - Cold Case Unit (has end_date)
('Daniel Park', 'Investigator', '2018-03-12', NULL, 1, 6),   -- pid 16 - Internal Affairs
('David Wilson', 'Administrator', '2016-09-05', NULL, 4, 6),         -- pid 17 - Internal Affairs
```

```
('Lisa Martinez', 'Admin Assistant', '2022-07-18', NULL, 4, 6),      -- pid 18 - Internal Affairs
('Isabella Rivera', 'Community Officer', '2020-10-22', NULL, 3, 7),  -- pid 19 - Community Relations
('Samantha Baker', 'Detective', '2019-06-08', NULL, 2, 7),           -- pid 20 - Community Relations
('James Taylor', 'Evidence Specialist', '2021-11-17', NULL, 3, 7),   -- pid 21 - Community Relations
('Ethan Wright', 'Training Instructor', '2017-04-03', NULL, 2, 8),   -- pid 22 - Training Division
('Michael Brown', 'Lab Technician', '2020-02-28', NULL, 2, 8),       -- pid 23 - Training Division
('Jessica Green', 'Forensic Technician', '2022-09-14', NULL, 2, 8),  -- pid 24 - Training Division
('Maria Garcia', 'Forensic Analyst', '2015-12-01', NULL, 1, 9),      -- pid 25 - Evidence Processing Center
('Jennifer Lee', 'Forensic Scientist', '2018-08-19', NULL, 1, 9),    -- pid 26 - Evidence Processing Center
('Amanda White', 'Forensic Analyst', '2019-05-06', NULL, 2, 9),      -- pid 27 - Evidence Processing Center
('Kevin Scott', 'Evidence Manager', '2020-07-23', NULL, 3, 10),      -- pid 28 - Forensic Accounting
('Christopher Hall', 'Lab Director', '2014-10-11', NULL, 1, 10),     -- pid 29 - Forensic Accounting
('Richard Adams', 'Admin Officer', '2021-12-05', NULL, 4, 10),       -- pid 30 - Forensic Accounting
('Mia Thompson', 'Surveillance Operat', '2020-03-30', NULL, 2, 11), -- pid 31 - Surveillance Division
('Michelle Hill', 'Evidence Clerk', '2022-08-12', NULL, 3, 11),      -- pid 32 - Surveillance Division
('Steven Wright', 'Detective', '2018-11-27', NULL, 2, 11);           -- pid 33 - Surveillance Division
```

Features:

- Employment Status:
  - end_date IS NULL = Currently employed (most records)
  - end_date IS NOT NULL = Former employee (2 records)
- Clearance Levels: Range 1-4 (1=highest, 4=lowest)
- Department Assignment: All did values reference existing departments
- Realistic Roles: Mix of detectives, analysts, administrators
- Temporal Data: Start dates from 2014-2022, end dates in 2024 for departed staff

Warrant

```
INSERT INTO Warrant (warrant_type, issue_date, exp_date, case_ID) VALUES
('Search', '2023-11-20', '2023-12-20', 1),
('Device Search', '2023-11-25', '2023-12-25', 1),
('Arrest', '2023-11-22', '2023-12-22', 2),
('Search', '2023-11-23', '2023-12-23', 2),
('Surveillance', '2023-11-28', '2024-01-01', 3),
('Data Access', '2024-01-01', '2024-02-01', 4),
('Search', '2024-01-05', '2024-02-05', 4),
('Device Search', '2024-03-10', '2024-04-10', 5),
('Wiretap', '2024-04-15', '2024-05-15', 6),
('Search', '2024-04-18', '2024-05-18', 6),
('Data Access', '2024-04-20', '2024-05-20', 6),
('Tracking', '2024-05-20', '2024-06-20', 7),
('Surveillance', '2024-05-25', '2024-06-25', 7),
('Search', '2024-06-05', '2024-07-05', 8),
```

```
('Seizure', '2024-07-15', '2024-08-15', 9),
('Search', '2024-07-20', '2024-08-20', 9),
('Arrest', '2024-08-01', '2024-09-01', 10),
('Search', '2024-08-05', '2024-09-05', 10),
('Detention', '2024-08-12', '2024-09-12', 11),
('Wiretap', '2024-08-15', '2024-09-15', 12),
('Tracking', '2024-08-18', '2024-09-18', 12),
('Search', '2024-09-02', '2024-10-02', 13),
('Seizure', '2024-09-05', '2024-10-05', 13),
('Tracking', '2024-10-10', '2024-11-10', 14),
('Arrest', '2024-10-15', '2024-11-15', 15),
('Search', '2024-10-18', '2024-11-18', 15),
('Search', '2025-01-20', '2025-02-20', 21),
('Device Search', '2025-02-25', '2025-03-25', 24),
('Arrest', '2025-03-01', '2025-04-01', 23);
```

Features:

- CHECK Constraint: All warrant types are valid per CHECK constraint
- Date Logic: exp_date always after issue_date

- Multiple Warrants: Cases can have multiple warrants (e.g., case 1 has 2)
- Temporal Spread: Warrants from 2023-2025
- Referential Integrity: All case_ID values exist

Evidence_items

```
INSERT INTO Evidence_items (case_ID, storage_loct, evidence_type, collection_date) VALUES
(1, 'Evidence Locker A-12', 'weapon', '2023-11-16'),
(1, 'Digital Vault DV-45', 'digital file', '2023-11-17'),
(1, 'Bio Lab BL-3', 'biological sample', '2023-11-18'),
(2, 'Evidence Locker C-7', 'physical', '2023-11-21'),
(2, 'Chem Lab CL-9', 'chemical', '2023-11-22'),
(3, 'Digital Vault DV-12', 'digital file', '2024-01-02'),
(3, 'Evidence Locker B-4', 'weapon', '2024-01-03'),
(4, NULL, 'biological sample', '2024-01-11'),
(4, 'Evidence Locker D-2', 'physical', '2024-01-12'),
(5, 'Digital Vault DV-33', 'digital file', '2024-03-16'),
(5, 'Chem Lab CL-15', 'chemical', '2024-03-17'),
(6, 'Evidence Locker A-9', 'weapon', '2024-04-28'),
(6, 'Bio Lab BL-12', 'biological sample', '2024-04-29'),
(7, 'Evidence Locker E-7', 'physical', '2024-05-06'),
```

```
(7, NULL, 'digital file', '2024-05-07'),
(8, 'Chem Lab CL-6', 'chemical', '2024-06-13'),
(8, 'Evidence Locker B-11', 'weapon', '2024-06-14'),
(9, 'Bio Lab BL-5', 'biological sample', '2024-07-21'),
(9, 'Evidence Locker C-14', 'physical', '2024-07-22'),
(10, 'Digital Vault DV-8', 'digital file', '2024-08-02'),
(10, 'Chem Lab CL-18', 'chemical', '2024-08-03'),
(11, NULL, 'weapon', '2024-08-11'),
(11, 'Bio Lab BL-9', 'biological sample', '2024-08-12'),
(12, 'Evidence Locker A-5', 'physical', '2024-08-16'),
(12, 'Digital Vault DV-27', 'digital file', '2024-08-17'),
(1, 'Evidence Locker A-13', 'weapon', '2024-09-19'),
(2, 'Evidence Locker C-8', 'physical', '2024-09-23'),
(3, 'Digital Vault DV-13', 'digital file', '2024-10-04'),
(5, 'Digital Vault DV-34', 'digital file', '2025-02-18'),
(8, 'Evidence Locker B-12', 'weapon', '2025-03-15');
```

Features:

- NULL Handling: Some evidence has NULL storage locations
- CHECK Constraint: All evidence types valid per constraint
- Multiple Evidence: Cases have multiple evidence items
- Temporal Data: Collection dates from 2023-2025
- Storage Codes: Realistic storage location codes (A-12, DV-45, etc.)

Lab_analysis

```
INSERT INTO Lab_analysis (findings, technique_used, analysis_date, personnel_ID, case_ID, evidence_ID) VALUES
('Partial DNA pattern recovered', 'DNA profiling', '2023-12-10', 3, 1, 1),
('Recovered deleted files from device', 'Computer forensics', '2024-04-05', 3, 5, 6),
('Blood sample indicates A+ type', 'Blood typing', '2024-01-11', 24, 3, 4),
('Trace residue shows organic fibers', 'Trace detection', '2024-06-01', 24, 8, 9),
('Substance tested positive for narcotics', 'Drug testing', '2024-05-12', 6, 6, 7),
('Prints match victim's right index finger', 'Fingerprint comparison', '2024-08-13', 7, 10, 12),
('Fibers consistent with vehicle upholstery', 'Trace detection', '2024-08-14', 9, 11, 14),
('DNA indicates mixed profiles', 'DNA profiling', '2024-09-28', 9, 12, 15),
('Bullet striations match recovered weapon', 'Bullet comparison', '2024-10-15', 21, 15, 18),
('Toxic compounds present in blood sample', 'Toxicology test', '2024-10-16', 23, 9, 10),
('Gunshot residue found on clothing', 'Gunshot residue', '2024-10-10', 23, 16, 20),
('HDD showed signs of tampering', 'Computer forensics', '2024-11-17', 28, 18, 21),
('Explosive material detected on fragments', 'Explosive residue', '2025-01-18', 29, 20, 22),
('Footprint consistent with size 44 boot', 'Footprint analysis', '2025-02-19', 33, 22, 25),
('DNA match confirmed from recovered sample', 'DNA profiling', '2025-03-20', 3, 21, 29);
```

Features:

- CHECK Constraint: All techniques used are valid forensic techniques
- UNIQUE Constraint: No duplicate analyst-evidence-case combinations
- FOREIGN KEY Integrity: All personnel, case, and evidence references exist
- Temporal Consistency: Analysis dates after evidence collection dates
- Analyst Specialization: Personnel assigned match forensic roles

Chain_of_custody

```
INSERT INTO Chain_Of_Custody (custody_timestamp, transfer_reason) VALUES
('09:30:00', 'sent to lab'), -- cuid1
('11:15:00', 'stored'), -- cuid2
('14:20:00', 'retrieved for court'), -- cuid3
('16:45:00', 'archived'), -- cuid4
('10:00:00', 'destroyed'), -- cuid5
('13:30:00', 'transferred to another department'), -- cuid6
('08:45:00', 'sent to lab'), -- cuid7
('15:20:00', 'stored'), -- cuid8
('11:00:00', 'retrieved for court'), -- cuid9
('09:15:00', 'archived'), -- cuid10
('14:00:00', 'destroyed'), -- cuid11
('16:30:00', 'transferred to another department'), -- cuid12
('10:45:00', 'sent to lab'), -- cuid13
('12:30:00', 'stored'), -- cuid14
('15:45:00', 'retrieved for court'); -- cuid15
```

Features:

- CHECK Constraint: All transfer reasons valid per constraint ('sent to lab', 'stored', etc.)
- Time Progression: Times range from 08:45 (morning) to 16:45 (afternoon)
- Business Process Coverage: All evidence handling stages represented
- Sequential IDs: Auto-incremented custody_ID 1-15
- Realistic Workflow: Morning lab transfers, afternoon court retrievals
- No NULLs: All timestamps and reasons populated

Contact_info

```
INSERT INTO Contact_Info (contact_info, personnel_ID) VALUES
('john.smith@forensics.gov', 1),      -- John Smith, pid 1
('555-0101', 1),                      -- John Smith phone
('alex.morgan@forensics.gov', 2),     -- Alex Morgan, pid 2
('555-0102', 2),                      -- Alex Morgan phone
('thomas.king@forensics.gov', 3),     -- Thomas King, pid 3
('555-0103', 3),                      -- Thomas King phone
('robert.chen@forensics.gov', 4),     -- Robert Chen, pid 4
('555-0104', 4),                      -- Robert Chen phone
('ryan.cooper@forensics.gov', 5),     -- Ryan Cooper, pid 5
('555-0105', 5),                      -- Ryan Cooper phone
('liam.wilson@forensics.gov', 6),     -- Liam Wilson, pid 6
('555-0106', 6),                      -- Liam Wilson phone
('sophia.lee@forensics.gov', 7),      -- Sophia Lee, pid 7
('555-0107', 7),                      -- Sophia Lee phone
```

```
('sarah.johnson@forensics.gov', 8),   -- Sarah Johnson, pid 8
('555-0108', 8),                      -- Sarah Johnson phone
('chloe.martinez@forensics.gov', 9),  -- Chloe Martinez, pid 9
('555-0109', 9),                      -- Chloe Martinez phone
('marcus.johnson@forensics.gov', 10), -- Marcus Johnson, pid 10
('555-0110', 10),                     -- Marcus Johnson phone
('emily.davis@forensics.gov', 11),    -- Emily Davis, pid 11
('555-0111', 11),                     -- Emily Davis phone
('patricia.clark@forensics.gov', 12), -- Patricia Clark, pid 12
('555-0112', 12);
```

Features:

- Multiple Contacts: Personnel can have multiple contact methods (email + phone)
- Composite Primary Key: Ensures unique contact-personnel combinations
- Professional Emails: Standardized @forensics.gov domain
- Phone Format: Consistent 555-XXXX pattern
- Partial Coverage: Only 12 of 33 personnel have contact info (realistic)
- Valid References: All personnel_ID values exist

Case_event

```
INSERT INTO Case_Event (event_type, event_date, notes, case_ID) VALUES
('interrogation', '2023-11-16', 'Interviewed suspect John Doe at precinct', 1),
('search', '2023-11-18', 'Executed search warrant at suspect residence', 1),
('court filing', '2023-11-22', 'Filed initial charges with district court', 1),
('Suspect handling', '2023-11-24', 'Suspect transferred to holding facility', 1),
('Witness interview', '2023-11-26', 'Interviewed eyewitness at crime scene', 2),
('Scene search', '2023-11-28', NULL, 2),
('Evidence collection', '2023-11-30', 'Collected physical evidence from scene', 3),
('Surveillance operation', '2024-01-02', '24-hour surveillance on suspect vehicle', 4),
('Area canvass', '2024-03-05', NULL, 5),
('Stakeout operation', '2024-04-08', '48-hour stakeout at suspected meet location', 6),
('Lead update', '2024-05-11', 'New anonymous tip received via hotline', 7),
('Tip received', '2024-06-14', NULL, 8),
('interrogation', '2024-07-17', 'Second interview with alibi witness', 9),
('search', '2024-08-20', 'Search of suspect workplace', 10),
```

```
('court filing', '2024-08-23', 'Motion for additional search warrants', 11),
('Witness interview', '2024-08-26', 'Expert witness consultation', 12),
('Evidence collection', '2024-09-01', NULL, 13),
('Surveillance operation', '2024-09-04', 'Electronic surveillance authorized', 14),
('Area canvass', '2024-10-07', 'Business establishment interviews', 15),
('Lead update', '2024-10-10', 'Forensic results provide new direction', 16),
('interrogation', '2025-01-16', 'Interviewed new suspect in ongoing case', 21),
('search', '2025-02-18', 'Executed search warrant at new location', 24),
('court filing', '2025-03-22', 'Filed updated charges based on new evidence', 23);
```

Features:

- CHECK Constraint: All event_type values valid per constraint
- NULL Handling: Some events have NULL notes
- Temporal Data: Events from 2023-2025
- Multiple Events: Cases have multiple events (e.g., case 1 has 4 events)
- Detailed Notes: Most events have descriptive notes
- Event Progression: Shows investigation workflow
- Valid References: All case_ID values exist
- Event Diversity: All 12 event types represented

Judge

```
INSERT INTO Judge (jname, jrole) VALUES
('Mark Lewis', 'Chief Judge'),        -- judge_ID 1
('Sarah Bloom', 'Trial Judge'),       -- judge_ID 2
('Henry Cole', 'Magistrate'),         -- judge_ID 3
('Laura Adams', 'Justice'),           -- judge_ID 4
('Peter Grant', 'Senior Judge'),      -- judge_ID 5
('Emily Stone', 'Appellate'),         -- judge_ID 6
('James Ford', 'Trial Judge'),        -- judge_ID 7
('Lisa Monroe', 'Justice'),           -- judge_ID 8
('Robert Klein', 'Magistrate'),       -- judge_ID 9
('Anna Cruz', 'Appellate'),           -- judge_ID 10
('David Park', 'Trial Judge'),        -- judge_ID 11
('Karen West', 'Senior Judge'),       -- judge_ID 12
('Thomas Reed', 'Magistrate'),        -- judge_ID 13
('Nancy Hill', 'Trial Judge'),        -- judge_ID 14
('Steven Young', 'Justice');          -- judge_ID 15
```

Features:

- CHECK Constraint: All jrole values valid per constraint
- Role Distribution: Mix of Chief, Trial, Magistrate, Justice, Appellate, Senior judges
- NOT NULL Compliance: All names and roles populated
- Sequential IDs: Auto-incremented judge_ID 1-15
- Realistic Names: Diverse judge names
- Court Hierarchy: Reflects real judicial system roles

Jcontact_Info

```
INSERT INTO Jcontact_Info (contact_info, judge_ID) VALUES
('mark.lewis@court.gov', 1),          -- Mark Lewis - email
('555-1001', 1),                      -- Mark Lewis - phone
('sarah.bloom@court.gov', 2),         -- Sarah Bloom - email
('555-1002', 2),                      -- Sarah Bloom - phone
('henry.cole@court.gov', 3),          -- Henry Cole - email
('laura.adams@court.gov', 4),         -- Laura Adams - email
('555-1004', 4),                      -- Laura Adams - phone
('peter.grant@court.gov', 5),         -- Peter Grant - email
('emily.stone@court.gov', 6),         -- Emily Stone - email
('555-1006', 6),                      -- Emily Stone - phone
('james.ford@court.gov', 7),          -- James Ford - email
('555-1007', 7),                      -- James Ford - phone
('lisa.monroe@court.gov', 8),         -- Lisa Monroe - email
('robert.klein@court.gov', 9),        -- Robert Klein - email
('555-1009', 9),                      -- Robert Klein - phone
```

```
('anna.cruz@court.gov', 10),          -- Anna Cruz - email
('david.park@court.gov', 11),         -- David Park - email
('555-1011', 11),                     -- David Park - phone
('karen.west@court.gov', 12),         -- Karen West - email
('thomas.reed@court.gov', 13),        -- Thomas Reed - email
('555-1013', 13),                     -- Thomas Reed - phone
('nancy.hill@court.gov', 14),         -- Nancy Hill - email
('steven.young@court.gov', 15);       -- Steven Young - email
```

Features:

- Multiple Contacts: Some judges have email + phone, some only email
- Composite Primary Key: Ensures unique contact-judge combinations
- Court Domain: Standardized @court.gov emails
- Phone System: 555-1XXX for court phone numbers
- Partial Coverage: Not all judges have phone numbers (realistic)
- Valid References: All judge_ID values exist

Administrative_staff

```sql
INSERT INTO Administrative_staff (personnel_ID, oversee_did) VALUES
(17, 6),
(18, 6),
(30, 10),
(14, 5),
(31, 11),
(2, 1),
(5, 2),
(26, 3),
(27, 4),
(32, NULL),
(1, 1),
(25, 9);
```

Features:

- NULL Handling: Some staff have NULL oversee_did (no department oversight)
- Multiple Oversight: Departments can have multiple administrative staff (e.g., did=6 has 2 staff)
- Dual Oversight: Some staff share department oversight (e.g., did=1 has 2 overseers)
- Staff_ID Assignment: Auto-incremented 1-12
- Valid References: All personnel_ID and oversee_did values exist
- RESTRICT Constraint: Departments with administrative oversight cannot be deleted

Forensic_analyst

```sql
INSERT INTO Forensic_analyst (personnel_ID) VALUES
(3),   -- Thomas King - Digital Forensics
(24),  -- Jessica Green - Forensic Technician
(6),   -- Liam Wilson - Forensic Accountant
(7),   -- Sophia Lee - CSI
(9),   -- Chloe Martinez - Evidence Processor (even though has end_date)
(21),  -- James Taylor - Evidence Specialist
(23),  -- Michael Brown - Lab Technician
(28),  -- Kevin Scott - Evidence Manager
(29),  -- Christopher Hall - Lab Director
(33);
```

Features:

- FOREIGN KEY Integrity: All personnel_ID values reference existing personnel
- One-to-One Enforcement: No duplicate personnel_ID values
- Analyst_ID Assignment: Auto-incremented 1-10
- Role-Based Selection: Personnel with forensic/technical roles
- Mixed Employment: Includes departed employee (9)
- Specialization Diversity: Digital, biological, trace, accounting forensics

Detectives

```
INSERT INTO Detectives(personnel_ID) VALUES
(4),
(8),
(10),
(11),
(12),
(13),
(15),
(16),
(19),
(20),
(22);
```

Features:

- One-to-One: Each personnel_ID appears only once
- Valid References: All personnel_ID values exist in Personnel
- Subset: Only 11 of 33 personnel are detectives

Handled_by

(Warrant ↔ Personnel ↔ Judge)

```
INSERT INTO Handled_by (warrant_ID, personnel_ID, judge_ID) VALUES
(1, 1, 1), (1, 7, 2), (2, 2, 3), (2, 3, 3), (2, 12, 4), (3, 4, 5), (3, 10, 5), (4, 11, 6), (4, 8, 7),
(5, 7, 8),(5, 3, 8),(5, 23, 9),(6, 3, 10),(6, 24, 10),(7, 12, 11),(7, 11, 11),(7, 1, 12),(8, 23, 13),
(8, 9, 14),(9, 10, 14),(9, 14, 14),(9, 7, 15),(10, 14, 1),(10, 21, 1),(11, 5, 2),(11, 6, 3),(11, 24, 3),
(12, 16, 4),(12, 20, 4),(13, 19, 5),(13, 25, 6),(14, 20, 6),(14, 28, 7),(14, 21, 7),(15, 6, 8),(15, 1, 8),
(16, 8, 9),(16, 3, 10),(17, 13, 11),(17, 11, 11),(17, 2, 12),(18, 15, 12),(18, 7, 13),(19, 18, 14),
(19, 9, 14),(19, 24, 15),(20, 21, 1),(20, 23, 2),(21, 22, 3),(21, 28, 4),(21, 30, 4),(22, 24, 5),(22, 25, 6),
(23, 25, 7),(23, 28, 7),(23, 29, 8),(24, 28, 9),(24, 10, 9),(25, 31, 10),(25, 20, 11),(25, 22, 11),
(26, 33, 12),(26, 23, 13),(26, 7, 13);
```

Features:

- Ternary Relationship: Links three entities
- Multiple Assignments: Warrants can involve multiple personnel and judges
- Valid References: All IDs reference existing records
- Business Logic: Reflects real-world warrant handling process

Has

(Evidence ↔ Chain_of_Custody)

```
INSERT INTO Has(custody_ID, case_ID, evidence_ID) VALUES
(1, 1, 1), (2, 1, 2), (3, 1, 3), (4, 2, 4), (5, 2, 5),
(6, 3, 6), (7, 3, 7), (8, 4, 8), (9, 4, 9), (10, 5, 10),
(11, 5, 11), (12, 6, 12), (15,5,10), (13,2,7);
```

Features:

- UNIQUE Constraint: Each custody_ID appears only once
- Evidence Tracking: Creates chain of custody audit trail
- Multiple Custody: Evidence can have multiple custody records
- Complete References: All foreign keys reference existing records

Participate

(Personnel ↔ Case_Event)

```
INSERT INTO Participate (personnel_ID, case_ID, case_event_ID) VALUES
(1, 1, 1), (3, 1, 2), (6, 1, 3), (10, 1, 4), (13, 2, 5),
(16, 2, 6), (20, 3, 7), (21, 4, 8), (23, 5, 9), (27, 6, 10),
(28, 7, 11), (29, 8, 12), (30, 9, 13), (1, 10, 14), (3, 11, 15), (21,5,9), (10,11,15);
```

Features:

- Event Documentation: Records who participated in each case event
- Multiple Participation: Events can involve multiple personnel
- Valid References: All IDs reference existing records

Part_of

(Personnel ↔ Chain_of_Custody)

```
INSERT INTO Part_Of (personnel_ID, custody_ID) VALUES
(4, 1),      -- Detective Robert Chen
(7, 2),      -- CSI Sophia Lee
(11, 3),     -- Detective Emily Davis
(23, 4),     -- Lab Tech Michael Brown
(17, 5),     -- Admin David Wilson
(1, 6),      -- Lead Detective John Smith
(21, 7),     -- Evidence Specialist James Taylor
(28, 8),     -- Evidence Manager Kevin Scott
(14, 9),     -- Detective Brian Anderson
(32, 10),    -- Evidence Clerk Michelle Hill
(3, 11),     -- Forensic Analyst Thomas King
(19, 12),    -- Community Officer Isabella Rivera
(24, 13),    -- Forensic Technician Jessica Green
(12, 14),    -- Detective Patricia Clark
(29, 15),    -- Lab Director Christopher Hall
(5, 1),      -- Financial Analyst Ryan Cooper
(9, 2),      -- Evidence Processor Chloe Martinez
(10, 3),     -- Tactical Commander Marcus Johnson
(26, 4),     -- Forensic Scientist Jennifer Lee
(20, 5),     -- Detective Samantha Baker
(6, 6),      -- Forensic Accountant Liam Wilson
(31, 7),     -- Surveillance Operator Mia Thompson
(15, 8),     -- Detective Nicole Young
(30, 9),     -- Admin Officer Richard Adams
(33, 10);    -- Detective Steven Wright
```

Features:

- Custody Accountability: Documents who handled each custody transfer
- Multiple Handlers: Custody events can involve multiple personnel
- Audit Trail: Supports chain of custody documentation

Handle

(Personnel ↔ Case)

```
INSERT INTO Handle (personnel_ID, case_ID) VALUES
(4, 2),(4, 3),  (4, 17), (10, 1), (10, 6), (10, 9), (11, 4), (11, 5),
(11, 7), (12, 4), (12, 5), (12, 19), (13, 2), (13, 5),  (13, 16), (16, 2),  (16, 7);
```

Features:

- Case Assignment: Primary personnel responsible for cases
- Workload Distribution: Personnel handle multiple cases
- Current Cases: Includes recent case assignments (2025)

4. DQL: Basic Queries

Query 1: Contains Theta Join with Using

```
SELECT P.personnel_ID, P.p_name
FROM Forensic_analyst F JOIN personnel P USING(personnel_ID)
WHERE P.end_date IS NULL
ORDER BY personnel_ID;
```

Purpose:

This query returns the IDs and names of all currently working forensic analysts.

Features:

- Joins forensic analyst data with personnel information
- Sorts results by personnel ID

Output:

personnel_ID	p_name
3	Thomas King
6	Liam Wilson
7	Sophia Lee
21	James Taylor
23	Michael Brown
24	Jessica Green
28	Kevin Scott
29	Christopher Hall
33	Steven Wright

Query 2: Contains Natural Join

```
SELECT H.case_ID, H.personnel_ID
FROM Case_Record CR NATURAL JOIN Handle H
WHERE CR.case_status='open'
ORDER BY H.case_ID;
```

Purpose:

This query returns the IDs of personnel (specifically detectives) currently handling each open case.

Features:

- Uses NATURAL JOIN for automatic column matching
- Filters results to show only open cases
- Returns case-personnel assignments ordered by case IDs

Output:

case_ID	personnel_ID
5	11
5	12
5	13
9	10
11	10
11	16
15	4
15	8
15	11
17	4
17	16
21	8
21	12
24	8
24	13

Query 3: Contains Distinct

```
SELECT DISTINCT P.personnel_ID, Per.p_name, Per.p_role, H.evidence_ID
FROM Part_of P JOIN Has H USING (custody_ID)
JOIN Personnel Per USING(personnel_ID)
ORDER BY P.personnel_ID, H.evidence_ID;
```

Purpose:

This query returns any personnel who have handled specific evidence items.

Features:

- The DISTINCT keyword removes duplicate rows
- Multiple table joins using the USING keyword
- Results are ordered by personnel and evidence IDs

Output:

personnel_ID	p_name	p_role	evidence_ID
1	John Smith	Lead Detective	6
3	Thomas King	Digital Forensics	11
4	Robert Chen	Detective	1
5	Ryan Cooper	Financial Analyst	1
6	Liam Wilson	Forensic Accountant	6
7	Sophia Lee	CSI	2
9	Chloe Martinez	Evidence Processor	2
10	Marcus Johnson	Tactical Commander	3
11	Emily Davis	Detective	3
14	Brian Anderson	Detective	9
15	Nicole Young	Detective	8
17	David Wilson	Administrator	5
19	Isabella Rivera	Community Officer	12
20	Samantha Baker	Detective	5
21	James Taylor	Evidence Specialist	7
23	Michael Brown	Lab Technician	4
24	Jessica Green	Forensic Technician	7
26	Jennifer Lee	Forensic Scientist	4
28	Kevin Scott	Evidence Manager	8
29	Christopher Hall	Lab Director	10
30	Richard Adams	Admin Officer	9
31	Mia Thompson	Surveillance Operat	7
32	Michelle Hill	Evidence Clerk	10
33	Steven Wright	Detective	10
34	Test Officer	Forensic Analyst	1
34	Test Officer	Forensic Analyst	2
34	Test Officer	Forensic Analyst	4
34	Test Officer	Forensic Analyst	5
34	Test Officer	Forensic Analyst	8

Query 4: Contains Order By

```
SELECT evidence_ID, custody_timestamp, transfer_reason
FROM Chain_of_custody NATURAL JOIN Has
ORDER BY evidence_ID, custody_timestamp;
```

Purpose:

This query returns a chronological table for evidence custody transfers, as well as the time and reason for each transfer.

Features:

- Uses NATURAL JOIN to combine related tables
- Uses ORDER BY to sort by evidence ID then timestamp

Output:

evidence_ID	custody_timestamp	transfer_reason
1	09:30:00	sent to lab
1	10:00:00	sent to lab
2	10:30:00	stored
2	11:15:00	stored
3	14:20:00	retrieved for court
4	11:00:00	retrieved for court
4	16:45:00	archived
5	10:00:00	destroyed
5	11:30:00	archived
6	13:30:00	transferred to another department
7	08:45:00	sent to lab
7	10:45:00	sent to lab
8	12:00:00	transferred to another department
8	15:20:00	stored
9	11:00:00	retrieved for court
10	09:15:00	archived
10	15:45:00	retrieved for court
11	14:00:00	destroyed
12	16:30:00	transferred to another department

Query 5: Contains Like

```
SELECT W.issue_date, W.exp_date, C.case_ID, C.open_date, W.warrant_ID, W.warrant_type, C.case_status
FROM Warrant W Natural Join Case_Record C
WHERE W.warrant_type LIKE '%Search%' OR W.warrant_type LIKE '%Arrest%'
ORDER BY W.issue_date;
```

Purpose:

This query returns all search and arrest warrants with associated case information.

Features:

- NATURAL JOIN joins every warrant with its associated case
- The LIKE operator filters warrant types
- Chronological sorting by issue date

Output:

issue_date	exp_date	case_ID	open_date	warrant_ID	warrant_type	case_status
2023-11-20	2023-12-20	1	NULL	1	Search	Pending
2023-11-22	2023-12-22	2	2023-11-20	3	Arrest	Closed
2023-11-23	2023-12-23	2	2023-11-20	4	Search	Closed
2023-11-25	2023-12-25	1	NULL	2	Device Search	Pending
2024-01-05	2024-02-05	4	2024-01-10	7	Search	Closed
2024-03-10	2024-04-10	5	2024-03-15	8	Device Search	Open
2024-04-18	2024-05-18	6	2024-04-28	10	Search	Closed
2024-06-05	2024-07-05	8	2024-06-12	14	Search	Closed
2024-07-20	2024-08-20	9	2024-07-20	16	Search	Open
2024-08-01	2024-09-01	10	NULL	17	Arrest	Pending
2024-08-05	2024-09-05	10	NULL	18	Search	Pending
2024-09-02	2024-10-02	13	NULL	22	Search	Pending
2024-10-15	2024-11-15	15	2024-10-10	25	Arrest	Open
2024-10-18	2024-11-18	15	2024-10-10	26	Search	Open
2025-01-20	2025-02-20	21	2025-02-20	27	Search	Open
2025-02-25	2025-03-25	24	2025-03-05	28	Device Search	Open
2025-03-01	2025-04-01	23	2025-03-01	29	Arrest	Closed

Query 6: Contains a Grouping Aggregate Function (With Group By)

```
SELECT ct.crime_type, COUNT(cr.case_ID) AS total_cases,
    AVG(DATEDIFF(cr.close_date, cr.open_date)) AS avg_days_to_close
FROM Case_Record cr JOIN Crime_type ct USING(case_ID)
WHERE cr.case_status = 'Closed'
GROUP BY ct.crime_type
ORDER BY avg_days_to_close DESC;
```

Purpose:

This query returns the number of cases solved as well as the average number of days to close cases for each crime type.

Features:

- GROUP BY aggregates data per crime type
- The COUNT function returns the number of closed cases
- DATEDIFF returns the difference between the opening and closing date of each case in days
- AVG calculates the average resolution time
- Sorts the information by resolution time in descending order

Output:

crime_type	total_cases	avg_days_to_close
identity theft	1	134.0000
extortion	1	127.0000
robbery	2	117.0000
fraud	2	107.5000
burglary	3	106.0000
arson	1	101.0000
assault	3	95.6667
cybercrime	3	95.6667
kidnapping	1	81.0000

Query 7: Contains a General Aggregate Function (Without Group By)

```
SELECT count(Distinct personnel_ID) AS Number_of_personnel,
        count(Distinct Judge_ID) AS Number_of_Judges
FROM Handled_by;
```

Purpose:

This query returns the total number of unique personnel and judges in the case handling system

Features:

- COUNT(DISTINCT) counts unique values
- Returns a single row result

Output:

	Number_of_personnel	Number_of_Judges
▶	29	15

Query 8: Contains Union

```
SELECT case_ID, 'Warrant Issued' AS activity_type, warrant_type AS description, issue_date AS activity_date
FROM Warrant
UNION
SELECT case_ID, 'Case Event' AS activity_type, event_type AS description, event_date AS activity_date
FROM Case_Event
UNION
SELECT case_ID, 'Evidence Collected' AS activity_type, evidence_type AS description, collection_date AS activity_date
FROM Evidence_items
ORDER BY case_ID;
```

Purpose:

This query returns all warrants, events, and evidence for each case.

Features:

- Selects the information for warrants, case events, and evidence separately
- Combines results from the three tables using UNION
- Sorts the results by Case_ID

Output:

case_ID	activity_type	description	activity_date
1	Warrant Issued	Search	2023-11-20
1	Warrant Issued	Device Search	2023-11-25
1	Case Event	Suspect handling	2023-11-24
1	Case Event	court filing	2023-11-22
1	Case Event	search	2023-11-18
1	Case Event	interrogation	2023-11-16
1	Evidence Collected	weapon	2023-11-16
1	Evidence Collected	digital file	2023-11-17
1	Evidence Collected	biological sample	2023-11-18
1	Evidence Collected	weapon	2024-09-19
2	Warrant Issued	Arrest	2023-11-22
2	Warrant Issued	Search	2023-11-23
2	Case Event	Scene search	2023-11-28
2	Case Event	Witness interview	2023-11-26
2	Evidence Collected	physical	2023-11-21
2	Evidence Collected	chemical	2023-11-22
2	Evidence Collected	physical	2024-09-23
3	Warrant Issued	Surveillance	2023-11-28
3	Evidence Collected	digital file	2024-01-02
3	Evidence Collected	weapon	2024-01-03
3	Case Event	Evidence collection	2023-11-30
3	Evidence Collected	digital file	2024-10-04
4	Warrant Issued	Data Access	2024-01-01
4	Case Event	Surveillance ope...	2024-01-02

	4	Warrant Issued	Search	2024-01-05
	4	Evidence Collected	physical	2024-01-12
	4	Evidence Collected	biological sample	2024-01-11
	5	Warrant Issued	Device Search	2024-03-10
	5	Evidence Collected	digital file	2024-03-16
	5	Evidence Collected	chemical	2024-03-17
	5	Case Event	Area canvass	2024-03-05
	5	Evidence Collected	digital file	2025-02-18
	6	Warrant Issued	Data Access	2024-04-20
	6	Warrant Issued	Search	2024-04-18
	6	Warrant Issued	Wiretap	2024-04-15
	6	Case Event	Stakeout operat…	2024-04-08
	6	Evidence Collected	weapon	2024-04-28
	6	Evidence Collected	biological sample	2024-04-29
	7	Evidence Collected	digital file	2024-05-07
	7	Case Event	Lead update	2024-05-11
	7	Warrant Issued	Surveillance	2024-05-25
	7	Warrant Issued	Tracking	2024-05-20
	7	Evidence Collected	physical	2024-05-06
	8	Evidence Collected	weapon	2024-06-14
	8	Warrant Issued	Search	2024-06-05
	8	Case Event	Tip received	2024-06-14
	8	Evidence Collected	weapon	2025-03-15
	8	Evidence Collected	chemical	2024-06-13

case_ID	activity_type	description	activity_date
9	Warrant Issued	Search	2024-07-20
9	Evidence Collected	biological sample	2024-07-21
9	Warrant Issued	Seizure	2024-07-15
9	Case Event	interrogation	2024-07-17
9	Evidence Collected	physical	2024-07-22
10	Warrant Issued	Search	2024-08-05
10	Warrant Issued	Arrest	2024-08-01
10	Evidence Collected	digital file	2024-08-02
10	Case Event	search	2024-08-20
10	Evidence Collected	chemical	2024-08-03
11	Case Event	court filing	2024-08-23
11	Warrant Issued	Detention	2024-08-12
11	Evidence Collected	weapon	2024-08-11
11	Evidence Collected	biological sample	2024-08-12
12	Warrant Issued	Wiretap	2024-08-15
12	Warrant Issued	Tracking	2024-08-18
12	Evidence Collected	physical	2024-08-16
12	Evidence Collected	digital file	2024-08-17
12	Case Event	Witness interview	2024-08-26
13	Warrant Issued	Seizure	2024-09-05
13	Warrant Issued	Search	2024-09-02
13	Case Event	Evidence collection	2024-09-01
14	Warrant Issued	Tracking	2024-10-10
14	Case Event	Surveillance ope…	2024-09-04
15	Warrant Issued	Search	2024-10-18
15	Warrant Issued	Arrest	2024-10-15
15	Case Event	Area canvass	2024-10-07
16	Case Event	Lead update	2024-10-10
21	Case Event	interrogation	2025-01-16
21	Warrant Issued	Search	2025-01-20
23	Case Event	court filing	2025-03-22
23	Warrant Issued	Arrest	2025-03-01
24	Case Event	search	2025-02-18
24	Warrant Issued	Device Search	2025-02-25

Query 9: Contains Except

Without Except:

```
SELECT P.personnel_ID, P.p_name, D.dname
FROM Personnel P NATURAL JOIN department D
WHERE p.end_date IS NOT NULL AND NOT EXISTS(
    SELECT P.personnel_ID
    FROM Case_event C JOIN Participate Pa USING(case_ID)
    WHERE p.personnel_ID=Pa.personnel_ID)
ORDER BY dname;
```

With Except (unsupported):

```
/*
SELECT P.personnel_ID, P.p_name, D.dname
FROM Personnel P NATURAL JOIN department D
WHERE p.end_date IS NOT NULL
EXCEPT
SELECT P.personnel_ID, P.p_name, D.dname
FROM Personnel P JOIN department D USING(did) Join Participate Pa USING(personnel_ID)
WHERE P.end_date IS NOT NULL
ORDER BY dname;
*/
```

Purpose:

This query returns departed personnel who never participated in any case events along with their corresponding department name.

Features:

- Filters for personnel with an end date to get departed personnel
- Checks absence from case event participation
- Excludes any personnel who have participated in any case events
- Sorts results by department name

Output:

	personnel_ID	p_name	dname
▶	15	Nicole Young	Cold Case Unit
	9	Chloe Martinez	Crime Scene Unit

Query 10: Contains a Grouping Aggregate Function (with Group By and Having)

```
SELECT CT.crime_type, COUNT(case_ID) AS number_of_cases
FROM Case_Record  CR NATURAL JOIN crime_type CT
WHERE case_status = 'open'
GROUP BY CT.crime_type
Having COUNT(DISTINCT Case_ID)>3;
```

Purpose:

This query returns crime types with over 3 currently open cases. Useful for the database since it determines if a certain pattern is occurring with certain crime types.

Features:

- Joins each case record with its respective crime type
- Only includes open cases
- Groups results by crime_type
- Filters groups to only include those with with count > 3

Output:

	crime_type	number_of_cases
▶	assault	5
	burglary	4
	cybercrime	6
	fraud	4

Query 11: Contains Self-join

```
SELECT E1.case_ID, E1.evidence_type, COUNT(*) AS Same_type_Count
FROM Evidence_items E1 JOIN Evidence_items E2 USING (case_ID, evidence_type)
WHERE E1.evidence_ID!=E2.evidence_ID
GROUP BY E1.case_ID, E1.evidence_type
HAVING COUNT(*)>=1;
```

Purpose:

This query returns cases with multiple evidence items of the same type.

Features:

- Uses self-join to compare a table to itself
- Matches each case with its evidence based on the evidence type
- Uses WHERE to prevent comparing the same evidence item to itself
- Groups and counts duplicate evidence types

Output:

case_ID	evidence_type	Same_type_Count
1	weapon	2
2	physical	2
3	digital file	2
5	digital file	2
8	weapon	2

Query 12: Contains a Cartesian Product

```
SELECT CR.case_ID, CR.case_status, CT.crime_type
FROM Case_Record CR, Crime_type CT
WHERE CR.case_ID = CT.case_ID AND CR.case_status = 'Open'
  AND CT.crime_type IN ('Robbery', 'Assault', 'Burglary')
ORDER BY CR.case_ID, CT.crime_type;
```

Purpose:

This query returns every open case for specific crime types

Features:

- Uses a Cartesian product to implicitly join each case with its crime type, using WHERE to match the records and avoid redundancy
- Filters cases to only include open ones with crime types of robbery, assault, or burglary
- Sorts results by case IDs and crime types

Output:

case_ID	case_status	crime_type
5	Open	assault
5	Open	burglary
9	Open	assault
11	Open	assault
11	Open	burglary
11	Open	robbery
15	Open	assault
17	Open	burglary
21	Open	assault
21	Open	robbery
24	Open	burglary

Query 13: Contains Intersect

Without Intersect:

```
SELECT C.case_ID
FROM Case_Record C
WHERE DATEDIFF(close_date, open_date) >= 30 AND C.case_ID IN (
        SELECT case_ID
        FROM Has H JOIN Case_Record CR USING (case_id)
        GROUP BY case_ID
        HAVING count(*)>1
);
```

With Intersect (unsupported):

```
/*SELECT case_ID
FROM Case_Record
WHERE DATEDIFF(close_date, open_date) >= 30
INTERSECT
SELECT case_ID
FROM Has H JOIN Case_Record CR USING (case_id)
GROUP BY case_ID
HAVING count(*)>1;*/
```

Purpose:

This query returns cases that took more than a month to close and have multiple evidence items.

Features:

- Gets cases that took over 30 days to close
- Gets cases with multiple evidence items
- Combines both results

Output:

case_ID
2
4
NULL

Query 14: Theta Join with On

```
SELECT H.personnel_ID, count(C.case_ID) AS count_of_cases
FROM Handle H JOIN Case_Record C ON (C.case_ID=H.case_ID AND C.case_status = 'Open')
GROUP BY H.personnel_ID
HAVING count(C.case_ID) >1;
```

Purpose:

This query returns detectives handling multiple open cases simultaneously.

Features:

- Joins each detective with the cases handled by them
- Only includes open cases by using ON to check the case status
- Uses GROUP BY to aggregate per personnel
- Filters personnel to only include those handling multiple cases

Output:

personnel_ID	count_of_cases
11	2
12	2
13	2
10	2
16	2
4	2
8	3

Query 15: Extra Query

```
SELECT case_ID, case_status, DATEDIFF(close_date, open_date) AS working_time
FROM Case_Record
WHERE case_status = 'closed' AND DATEDIFF(close_date, open_date)>=30
UNION
SELECT case_ID, case_status, DATEDIFF(CURDATE(), open_date) AS working_time
FROM case_record
WHERE case_status ='open' AND DATEDIFF(CURDATE(), open_date)>=30;
```

Purpose:

This query returns combined long-duration cases with the working time for each in days.

Features:

- Uses DATEDIFF to get the working time for each case
- Compares the start and end times for closed cases and the start and current time (using CURDATE()) for open ones
- Filters the results to only include cases that took over a month
- Combines the results of both queries using UNION

Output:

case_ID	case_status	working_time
2	Closed	82
4	Closed	86
6	Closed	109
8	Closed	81
12	Closed	127
14	Closed	134
18	Closed	119
20	Closed	125
23	Closed	101
5	Open	632
9	Open	505
11	Open	484
15	Open	423
17	Open	377
21	Open	290
24	Open	277

5. DQL: Advanced Queries

Query 1: Set membership

```
SELECT C.case_ID
FROM Case_Record C
WHERE C.case_ID NOT IN (
    SELECT case_ID
    FROM (
        SELECT E.case_ID
        FROM Evidence_items E
        LEFT JOIN Lab_analysis LA USING (evidence_ID, case_ID)
        LEFT JOIN Personnel P USING (personnel_ID)
        WHERE (LA.analysis_ID IS NULL) OR (P.personnel_ID IS NOT NULL AND P.clearance_lvl>1)
 ) AS Cases
);
```

Purpose:

This query returns the case_ID's that have no problematic evidence items based on the evidence item's lab analysis or the clearance level of the personnel that handled that evidence item.

Features:

- Uses subquery to find cases with problematic evidence items
- Returns the cases NOT IN the subquery
- Uses Left Join to deal with missing lab or personnel records

Output:

case_ID
16
17
18
19
20
22
23
25

Query 2: Set Comparison

```
SELECT C.case_ID, Ct.crime_type,
    MIN(W.issue_date) AS first_warrant_date,
    MIN(E.collection_date) AS first_evidence_date
FROM Case_Record C
JOIN Crime_type Ct USING (case_ID)
JOIN Warrant W USING (case_ID)
JOIN Evidence_items E USING (case_ID)
WHERE E.collection_date > ANY (
    SELECT issue_date
    FROM Warrant W2
    WHERE W2.case_ID = C.case_ID
)
GROUP BY C.case_ID, Ct.crime_type
ORDER BY C.case_ID;
```

Purpose:

This query returns all cases and their details (ID, crime type, first warrant date and first evidence date) where the case has an evidence item that was collected after a warrant has been issued for that case. This query is useful for a timeline analysis of issued warrants and the collection of an evidence item for cases.

Features:

- Uses a nested query and set comparison in the where clause (query that returns all the issue dates of warrants for that case) to check that the collection date of an evidence item is greater than any of the issue dates

Output:

case_ID	crime_type	first_warrant_date	first_evidence_date
1	assault	2023-11-20	2024-09-19
1	homicide	2023-11-20	2024-09-19
2	assault	2023-11-22	2024-09-23
2	burglary	2023-11-22	2024-09-23
2	cybercrime	2023-11-22	2024-09-23
3	fraud	2023-11-28	2024-01-02
4	assault	2024-01-01	2024-01-11
4	cybercrime	2024-01-01	2024-01-11
5	assault	2024-03-10	2024-03-16
5	burglary	2024-03-10	2024-03-16
5	cybercrime	2024-03-10	2024-03-16
5	drug traffi...	2024-03-10	2024-03-16
6	burglary	2024-04-15	2024-04-28
6	robbery	2024-04-15	2024-04-28
8	fraud	2024-06-05	2024-06-13
8	kidnapping	2024-06-05	2024-06-13
9	arson	2024-07-15	2024-07-21
9	assault	2024-07-15	2024-07-21
9	cybercrime	2024-07-15	2024-07-21
9	fraud	2024-07-15	2024-07-21
10	theft	2024-08-01	2024-08-02
12	burglary	2024-08-15	2024-08-16
12	extortion	2024-08-15	2024-08-16
13	money lau...	2024-09-02	2024-09-19
21	assault	2025-01-20	2025-02-18
21	forgery	2025-01-20	2025-02-18
21	fraud	2025-01-20	2025-02-18
21	robbery	2025-01-20	2025-02-18
24	burglary	2025-02-25	2025-03-15
24	cybercrime	2025-02-25	2025-03-15
24	drug traffi...	2025-02-25	2025-03-15
24	homicide	2025-02-25	2025-03-15
24	theft	2025-02-25	2025-03-15

Query 3: Set Cardinality

```
SELECT E.case_ID
FROM evidence_items E
Where E.evidence_type = 'biological sample'
AND EXISTS( SELECT E.evidence_ID
            FROM Lab_analysis L
            WHERE L.evidence_ID = E.evidence_ID AND L.technique_used = 'Blood typing'
            OR L.technique_used = 'Fingerprint comparison');
```

Purpose:

This query returns all case IDs that have evidence items that are biological samples and have at least one lab analysis using blood typing or fingerprint comparison. Meaning that a certain technique for the lab analysis has been used that corresponds to the evidence item's type.

Features:

- Uses nested queries with set cardinality (exists) in the where clause, to ensure at least one matching analysis with this specific evidence exists.

Output:

case_ID
1
4
6
9
11

Query 4: Set Cardinality

```
SELECT *
FROM Case_Record CR
WHERE NOT EXISTS (
    SELECT 1 FROM Evidence_items EI
    WHERE EI.case_ID = CR.case_ID
);
```

Purpose:

This query returns all attributes of the case_record table where the case has no evidence items related to it.

Features:

- Uses nested queries with set cardinality (not exists) in the where clause, to check that the nested query returns an empty table (the specific case has no evidence items).

Output:

case_ID	open_date	close_date	investigation_TID	case_status
16	NULL	NULL	1016	Pending
17	2024-11-25	NULL	1017	Open
18	2024-12-01	2025-03-30	1018	Closed
19	NULL	NULL	1019	Pending
20	2025-01-15	2025-05-20	1020	Closed
22	NULL	NULL	1022	Pending
23	2025-03-01	2025-06-10	1023	Closed
25	NULL	NULL	1025	Pending
NULL	NULL	NULL	NULL	NULL

Query 5: Two or more nesting levels

```
SELECT D.did, D.dname, COUNT(DISTINCT YP.personnel_ID) AS nb_of_high_achieving_personnel
FROM Department D JOIN (
    SELECT P.personnel_ID, P.did, YEAR(C.close_date) AS close_year, COUNT(DISTINCT H.case_ID) AS cases_solved
    FROM Handle H
    JOIN Case_Record C USING(case_ID)
    JOIN Personnel P USING(personnel_ID)
    WHERE C.case_status = 'Closed'
    GROUP BY P.personnel_ID, P.did, YEAR(C.close_date)
) AS YP USING(did)
JOIN (SELECT YearlyCount.did, YearlyCount.close_year, AVG(YearlyCount.case_count) AS avg_cases_per_personnel
        FROM (SELECT P1.personnel_ID, P1.did, YEAR(C1.close_date) AS close_year,
        COUNT(DISTINCT H1.case_ID) AS case_count
            FROM Handle H1
            JOIN Case_Record C1 USING(case_ID)
            JOIN Personnel P1 USING(personnel_ID)
            WHERE C1.case_status = 'Closed'
            GROUP BY P1.personnel_ID, P1.did, YEAR(C1.close_date)
        ) AS YearlyCount
        GROUP BY YearlyCount.did, YearlyCount.close_year
    ) AS DeptAvg ON D.did = DeptAvg.did AND YP.close_year = DeptAvg.close_year
WHERE YP.cases_solved >= DeptAvg.avg_cases_per_personnel
GROUP BY D.did, D.dname;
```

Purpose:

This query returns the number of personnel who solve more cases per year than average for each department.

Features:

- YP calculates how many cases each personnel solved per year
- DeptAvg calculates the average number of cases closed a year per personnel in each department
- Both tables use COUNT(DISTINCT) to count the number of unique cases solved by every personnel
- It joins both tables to compare personnel's performance with the average one per department
- Uses GROUP BY with aggregation to count the number of high-achieving personnel per department

Output:

did	dname	nb_of_high_achieving_personnel
2	Financial Investigations	1
4	Tactical Operations	3
5	Cold Case Unit	1
6	Internal Affairs	1

Query 6: Division Operation

Without Except (Using Not Exists):

```
Select P.personnel_ID, P.p_name, P.p_role
From Personnel P
Where Not Exists (
    Select 1
    From Evidence_Items E
    Where E.evidence_type IS Not Null
    AND Not Exists(
    Select 1
    From Part_of Po
    Join Has H Using (custody_ID)
    Join Evidence_items E2 Using (evidence_ID)
    Where Po.personnel_ID = P.personnel_ID
    AND E2.evidence_type = E.evidence_type)
);
```

With Except (unsupported):

```
/*Select P.personnel_ID,P.p_name, P.p_role
From Personnel P
Where Not Exists (
    Select Distinct E.evidence_type
    From Evidence_Items E
    Where E.evidence_type IS Not Null
    EXCEPT
    Select Distinct E.evidence_type
    From Part_of Po
    Join Has H Using (custody_ID)
    Join Evidence_item Using (evidence_ID)
    Where Po.personnel_ID = P.personnel_ID
);*/
```

Purpose:

This query returns the personnel ID, their name and their role that have handled all evidence types in the database. Useful to analyse which personnel is the most qualified in the database.

Features:

- Uses set cardinality and except in order to perform the division operation in SQL.

Output:

	personnel_ID	p_name	p_role
▶	34	Test Officer	Forensic Analyst

Query 7: Nested Query in From clause

```
SELECT C.case_ID, C.case_status, W.total_warrants
FROM case_record C JOIN
    (SELECT case_ID, COUNT(*) AS total_warrants
    FROM warrant
    GROUP BY case_ID) W USING(case_ID)
WHERE W.total_warrants >
    (SELECT AVG(cnt)
    FROM
    (SELECT COUNT(*) AS cnt
    FROM warrant
    GROUP BY case_ID) avg_table);
```

Purpose:

This query returns the case ID, case status, and total warrants per case that have more warrants issued than the average issued warrants across all cases.

Features:

- Uses a nested query in the From clause to get the total issued warrants for each case and then joins it with the 'case_record' table.
- Uses another nested query in the Where clause (query that finds the average amount of issued warrants across all cases), to find all cases with the amount of issued warrants greater than the average.

Output:

case_ID	case_status	total_warrants
1	Pending	4
2	Closed	2
4	Closed	2
6	Closed	3
7	Pending	2
9	Open	2
10	Pending	2
12	Closed	2
13	Pending	2
15	Open	2

Query 8: Nested Query in Select clause

```
SELECT C.case_ID,
        (SELECT count(*)
        FROM Evidence_items E
        WHERE C.case_ID = E.case_ID
    ) AS nb_of_evidence_items,
        (SELECT count(personnel_ID)
        FROM Personnel P JOIN Handle H USING (personnel_ID)
        where P.end_date IS NULL AND H.case_ID=C.case_ID
    ) AS nb_of_working_personnel
FROM Case_Record C
WHERE case_status = 'Closed';
```

Purpose:

This query returns for each closed case the number of evidence items associated with it and the number of personnel who worked on it.

Features:

- Uses nested queries in the select clause to get the count of evidence items and the count of active personnel working on that case.

Output:

case_ID	nb_of_evidence_items	nb_of_working_personnel
2	3	3
4	2	2
6	2	1
8	3	0
12	2	0
14	1	0
18	0	0
20	0	0
23	0	0

Query 9: Update using Case

```
UPDATE Department D
SET budget =
    CASE
        WHEN budget < 300000 THEN budget + 150000
        WHEN budget > 300000 AND budget < 400000 THEN budget + 100000
        ELSE budget + 50000
    END
WHERE EXISTS (
    SELECT 1
    FROM Personnel P
    JOIN Participate Pa Using (personnel_ID)
    JOIN Case_Record Cr Using (case_ID)
    WHERE P.did = D.did
      AND Cr.case_status = 'Open'
      AND Cr.open_date >= CURDATE() - INTERVAL 180 DAY
);
```

Purpose:

This update raises the budget of departments that still have active personnel, working on open cases, and have opened cases in the last 180 days (department is active). This update raises the budget according to the actual budget of each department.

Features:

- Uses Case in order to determine the amount of the raise depending on the actual budget of the department.
- Uses set cardinality to determine if the department is active or not.

Query 10: Outer Join

```
SELECT E.evidence_ID, E.evidence_type, C.custody_ID, C.custody_timestamp, P.personnel_ID
FROM evidence_items E LEFT OUTER JOIN (has H JOIN
(chain_of_custody C JOIN Part_Of P USING(custody_ID)) USING(custody_ID))
USING (evidence_ID);
```

Purpose:

This query returns all evidence items(ID and type) and the details of their chain of custody (ID, timestamp, personnel_ID) including the evidence items that have no chain of custody. Getting the evidence items that don't have a chain of custody benefits the database in order to know which items have been handled and which have not.

Features:

- Uses left outer join in order to get all evidence_items including those that have no chain of custody.

Output:

evidence_ID	evidence_type	custody_ID	custody_timestamp	personnel_ID
1	weapon	1	09:30:00	4
1	weapon	1	09:30:00	5
1	weapon	16	10:00:00	34
2	digital file	2	11:15:00	7
2	digital file	2	11:15:00	9
2	digital file	17	10:30:00	34
3	biological sample	3	14:20:00	10
3	biological sample	3	14:20:00	11
4	physical	4	16:45:00	23
4	physical	4	16:45:00	26
4	physical	18	11:00:00	34
5	chemical	5	10:00:00	17
5	chemical	5	10:00:00	20
5	chemical	19	11:30:00	34
6	digital file	6	13:30:00	1
6	digital file	6	13:30:00	6
7	weapon	7	08:45:00	21
7	weapon	7	08:45:00	31
7	weapon	13	10:45:00	24
8	biological sample	8	15:20:00	15
8	biological sample	8	15:20:00	28
8	biological sample	20	12:00:00	34
9	physical	9	11:00:00	14
9	physical	9	11:00:00	30
10	digital file	10	09:15:00	32
11	chemical	11	14:00:00	3
12	weapon	12	16:30:00	19
13	biological sample	NULL	NULL	NULL
14	physical	NULL	NULL	NULL
15	digital file	NULL	NULL	NULL
16	chemical	NULL	NULL	NULL
17	weapon	NULL	NULL	NULL
18	biological sample	NULL	NULL	NULL
19	physical	NULL	NULL	NULL
20	digital file	NULL	NULL	NULL
21	chemical	NULL	NULL	NULL
22	weapon	NULL	NULL	NULL
23	biological sample	NULL	NULL	NULL
24	physical	NULL	NULL	NULL
25	digital file	NULL	NULL	NULL
26	weapon	NULL	NULL	NULL
27	physical	NULL	NULL	NULL
28	digital file	NULL	NULL	NULL
29	digital file	NULL	NULL	NULL
30	weapon	NULL	NULL	NULL
31	weapon	NULL	NULL	NULL
32	physical	NULL	NULL	NULL
33	digital file	NULL	NULL	NULL
34	digital file	NULL	NULL	NULL
35	weapon	NULL	NULL	NULL

Query 11: Assertion

Using Trigger:

```
DELIMITER //
CREATE TRIGGER CheckDetectiveRoleExclusivity
BEFORE INSERT ON Detectives
FOR EACH ROW
BEGIN
    IF EXISTS (
        SELECT personnel_ID FROM (
            SELECT personnel_ID FROM Forensic_analyst
            UNION
            SELECT personnel_ID FROM Administrative_staff
        ) AS OtherRoles
        WHERE OtherRoles.personnel_ID = NEW.personnel_ID
    ) THEN
        SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Personnel already has another specialized role';
    END IF;
END//
DELIMITER ;
```

Using Assert (unsupported):

```
/*CREATE ASSERTION CheckDetectiveRoleExclusivity
CHECK (
    NOT EXISTS (
        SELECT 1
        FROM Detectives D JOIN Forensic_analyst FA USING (personnel_ID)
        UNION
        SELECT 1
        FROM Detectives D JOIN Administrative_staff A USING (personnel_ID)
    )
);*/
```

To test the assertion:

```
INSERT INTO forensic_analyst(personnel_ID) VALUES
(30); -- administrative_staff
```

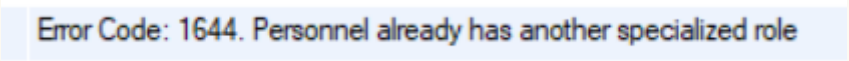
Purpose:

This assertion serves the purpose of implementing the non-overlapping characteristic of the ISA relationship in the E-R design on the database. A detective, therefore, cannot be a forensic analyst nor an administrative staff member at the same time. This assertion checks before the insert into the detectives table that this personnel_ID doesn't already exist in the administrative staff table or the forensic analyst table.

Features:

- Since MySQL doesn't support asserts, a trigger was used to substitute the assert. This trigger raises an error whenever an insert that violates the condition is initiated.
- Uses IF to decide which action to take based on the condition of the assert.

Output:



Query 12: Assertion

With Trigger:

```
DELIMITER //
CREATE TRIGGER CheckAdministrativeStaffRoleExclusivity
BEFORE INSERT ON Administrative_staff
FOR EACH ROW
BEGIN
    IF EXISTS (
        SELECT personnel_ID FROM (
            SELECT personnel_ID FROM Forensic_analyst
            UNION
            SELECT personnel_ID FROM Detectives
        ) AS OtherRoles
        WHERE OtherRoles.personnel_ID = NEW.personnel_ID
    ) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Personnel already has another specialized role';
    END IF;
END //
DELIMITER ;
```

With Assert (unsupported):

```
/*CREATE ASSERTION CheckAdministrativeStaffRoleExclusivity
CHECK (
    NOT EXISTS (
        SELECT 1
        FROM Administrative_staff A JOIN Forensic_analyst FA USING (personnel_ID)
        UNION
        SELECT 1
        FROM Administrative_staff A JOIN JOIN Detectives D USING (personnel_ID)
    )
);*/
```

Example to test the assert:

```
INSERT INTO administrative_staff(personnel_ID) VALUES
(10); -- detective
```

Purpose:

This assertion serves the purpose of implementing the not overlapping characteristic of the ISA relationship in the ER design on the database. An administrative staff therefore, cannot be a forensic analyst nor a detective at the same time. This assertion checks before the insert into the administrative_staff table that this personnel_ID doesn't already exist in the detectives table or the forensic analyst table.

Features:

- Since MySQL doesn't support asserts, a trigger was used to substitute the assert. This trigger raises an error whenever an insert that violates the condition is initiated.
- Uses IF to decide which action to take based on the condition of the assert.

Output:

```
Error Code: 1644. Personnel already has another specialized role
```

Query 13: Assertion

With Trigger:

```
DELIMITER //
CREATE TRIGGER CheckForensicAnalystRoleExclusivity
BEFORE INSERT ON Forensic_analyst
FOR EACH ROW
BEGIN
    IF EXISTS (
        SELECT personnel_ID FROM (
            SELECT personnel_ID FROM Detectives
            UNION
            SELECT personnel_ID FROM Administrative_staff
        ) AS OtherRoles
        WHERE OtherRoles.personnel_ID = NEW.personnel_ID
    ) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Personnel already has another specialized role';
    END IF;
END//
DELIMITER ;
```

With Assert (unsupported):

```
/*CREATE ASSERTION CheckForensicAnalystRoleExclusivity
CHECK (
    NOT EXISTS (
        SELECT 1
        FROM Forensic_analyst FA JOIN Detectives D USING (personnel_ID)
        UNION
        SELECT 1
        FROM Forensic_analyst FA JOIN Administrative_staff A USING (personnel_ID)
    )
);*/
```

Example to test the assert:

```
INSERT INTO detectives(personnel_ID) VALUES
(7); -- forensic_analyst
```

Purpose:

This assertion serves the purpose of implementing the not overlapping characteristic of the ISA relationship in the E-R design on the database. A forensic_analyst therefore, cannot be an administrative staff nor a detective at the same time. This assertion checks before the insert into the forensic_analyst table that this personnel_ID doesn't already exist in the detectives table or the administrative_staff table.

Features:

- Since MySQL doesn't support asserts, a trigger was used to substitute the assert. This trigger raises an error whenever an insert that violates the condition is initiated.
- Uses IF to decide which action to take based on the condition of the assert.

Output:

```
Error Code: 1644. Personnel already has another specialized role
```

Query 14: View

```
Create View Case_Summary As
Select C.case_ID, C.case_status, C.open_date, Ct.crime_type,
Count(E.evidence_ID) AS evidence_count, Count(L.analysis_ID) AS Analysis_count
From Case_Record C
Left Join Crime_type Ct Using (case_ID)
Left Join Evidence_items E  Using (case_ID)
Left Join Lab_analysis L Using (case_ID, evidence_ID)
Group by C.case_ID, C.case_status, C.open_date, Ct.crime_type;
```

Example to use the Case_summary view:

```
SELECT * FROM Case_Summary
WHERE case_status = 'Open'
ORDER BY evidence_count DESC;
```

Purpose:

This view returns for each case its summary, including the case details (ID, status, open date, and crime type), evidence count, and lab_analysis count.

Features:

- Uses outer left join to include all cases, including those with missing data.
- Groups by the details of each case in order to get the counts.

Output:

case_ID	case_status	open_date	crime_type	evidence_count	Analysis_count
5	Open	2024-03-15	assault	3	0
5	Open	2024-03-15	burglary	3	0
5	Open	2024-03-15	cybercrime	3	0
5	Open	2024-03-15	drug trafficking	3	0
9	Open	2024-07-20	arson	2	0
9	Open	2024-07-20	assault	2	0
9	Open	2024-07-20	cybercrime	2	0
9	Open	2024-07-20	fraud	2	0
11	Open	2024-08-10	assault	2	0
11	Open	2024-08-10	burglary	2	0
11	Open	2024-08-10	cybercrime	2	0
11	Open	2024-08-10	robbery	2	0
11	Open	2024-08-10	vandalism	2	0
15	Open	2024-10-10	assault	1	0
15	Open	2024-10-10	cybercrime	1	0
15	Open	2024-10-10	drug trafficking	1	0
15	Open	2024-10-10	fraud	1	0
15	Open	2024-10-10	homicide	1	0
21	Open	2025-02-20	assault	1	1
21	Open	2025-02-20	forgery	1	1
21	Open	2025-02-20	fraud	1	1
21	Open	2025-02-20	robbery	1	1
24	Open	2025-03-05	burglary	1	0
24	Open	2025-03-05	cybercrime	1	0
24	Open	2025-03-05	drug trafficking	1	0
24	Open	2025-03-05	homicide	1	0
24	Open	2025-03-05	theft	1	0
17	Open	2024-11-25	burglary	0	0
17	Open	2024-11-25	cybercrime	0	0
17	Open	2024-11-25	fraud	0	0
17	Open	2024-11-25	homicide	0	0

Query 15: View

```
CREATE VIEW Personnel_Activity AS
SELECT P.personnel_ID, P.p_name, P.p_role, D.dname AS department,
    COUNT(DISTINCT Pa.case_ID) AS cases_assigned,
    COUNT(DISTINCT Po.custody_ID) AS custody_events
FROM Personnel P
JOIN Department D Using (did)
LEFT JOIN Participate Pa Using (personnel_ID)
LEFT JOIN Part_of Po Using (personnel_ID)
GROUP BY P.personnel_ID, P.p_name, P.p_role, D.dname;
```

Example to use the Personnel_activity view:

```
SELECT * FROM Personnel_Activity
WHERE cases_assigned >= 2
ORDER BY custody_events DESC;
```

Purpose:

This view returns for personnel its activity including the personnel details (ID, name, role and department), count of assigned cases, and count of custody events that the personnel participated in.

Features:

- Uses outer left join to include all personnel, including those with missing data.
- Groups by the details of each case in order to get the counts.

Output:

personnel_ID	p_name	p_role	department	cases_assigned	custody_events
1	John Smith	Lead Detective	Cyber Crime Unit	2	1
3	Thomas King	Digital Forensics	Cyber Crime Unit	2	1
10	Marcus Johnson	Tactical Commander	Tactical Operations	2	1
21	James Taylor	Evidence Specialist	Community Relations	2	1

Query 16: View

```
CREATE VIEW Evidence_Custody_Hist AS
SELECT E.evidence_ID, E.evidence_type, C.custody_timestamp, C.transfer_reason,
P.p_name AS handled_by, D.dname AS department
FROM Evidence_items E
JOIN Has H Using (evidence_ID, case_ID)
JOIN Chain_of_Custody C Using (custody_ID)
JOIN Part_of Po Using (custody_ID)
JOIN Personnel P Using (personnel_ID)
JOIN Department D Using (did)
ORDER BY E.evidence_ID, C.custody_timestamp;
```

```
SELECT * FROM Evidence_Custody_Hist
WHERE evidence_ID = 10
ORDER BY custody_timestamp;
```

Purpose:

This view returns for evidence its custody history, including the evidence details (ID, type), the timestamp,the transfer reason of its chain of custody, and the name and department of the handler of the custody.

Features:

- Groups by evidence ID and custody timestamp in order to get the details of each chain of custody.

Output:

evidence_ID	evidence_type	custody_timestamp	transfer_reason	handled_by	department
10	digital file	09:15:00	archived	Michelle Hill	Surveillance Division
10	digital file	09:15:00	archived	Steven Wright	Surveillance Division
10	digital file	15:45:00	retrieved for court	Christopher Hall	Forensic Accounting

Query 17: Trigger

```
DELIMITER $$
CREATE TRIGGER case_is_open
BEFORE INSERT ON warrant
FOR EACH ROW
BEGIN
    DECLARE curr_status VARCHAR(10);

    SELECT C.case_status INTO curr_status
    FROM Case_record C
    WHERE C.case_ID = NEW.case_ID;

        CASE
            when curr_status !='open' THEN
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Cannot issue warrant for non-Open case.';
        END CASE;
END$$;
DELIMITER ;
```

To test the Trigger:

```
INSERT INTO warrant(warrant_type, issue_date, exp_date, case_ID) VALUES
('Search', '2023-11-20', '2023-12-20', 1),
('Device Search', '2023-11-25', '2023-12-25', 1);
```

Purpose:

This trigger checks before the insertion of a tuple into the warrant table that the corresponding case's status is open. If the case status is different from 'open' (pending or closed), then the trigger raises an error and prevents the insert from happening. This enforces data integration within the database.

Features:

- Uses signal to throw an error
- Uses Case to determine which action to take depending on the case status.
- Checks for each row and not each tuple because it all depends on each row's case status.

Output:

Error Code: 1644. Cannot issue warrant for non-Open case.

Query 18: Stored Procedure

```
DELIMITER $$
CREATE PROCEDURE generate_monthly_report(IN month_date INT, IN year_date INT)
BEGIN
    DECLARE total_cases_opened INT;
    DECLARE total_cases_closed INT;
    DECLARE total_evidence_collected INT;
    DECLARE total_lab_analysis INT;
    DECLARE active_personnel INT;
    DECLARE avg_case_duration INT;
    DECLARE total_warrants_issued iNT;

    SELECT COUNT(*) INTO total_cases_opened
    FROM case_record
    WHERE (case_status = 'open' OR case_status = 'closed')
    AND MONTH(open_date) = month_date
    AND YEAR(open_date) = year_date;
```

```
SELECT COUNT(*) INTO total_cases_closed
FROM case_record
WHERE case_status = 'closed' AND MONTH(close_date) = month_date
AND YEAR(close_date) = year_date;

SELECT COUNT(*) INTO total_evidence_collected
FROM evidence_items
WHERE MONTH(collection_date) = month_date
AND YEAR(collection_date) = year_date;

SELECT COUNT(*) INTO total_lab_analysis
FROM lab_analysis
WHERE MONTH(analysis_date) = month_date
AND YEAR(analysis_date) = year_date;

SELECT COUNT(*) INTO active_personnel
FROM personnel
WHERE end_date IS NULL;
```

```
SELECT AVG(DATEDIFF(close_date, open_date)) INTO avg_case_duration
FROM case_record
WHERE MONTH(close_date) = month_date
AND YEAR(close_date) = year_date;

SELECT count(*) INTO total_warrants_issued
FROM warrant
where MONTH(issue_date)= month_date
AND YEAR(issue_date) = year_date;
```

```
SELECT '=====MONTHLY REPORT======' AS header;
SELECT 'METRIC' AS metric, 'VALUE' AS v
UNION ALL
SELECT '---', '---'
UNION ALL
SELECT 'Cases Opened', total_cases_opened
UNION ALL
SELECT 'Cases Closed', total_cases_closed
UNION ALL
SELECT 'Evidence Collected', total_evidence_collected
UNION ALL
SELECT 'Evidence Analyzed', total_lab_analysis
UNION ALL
SELECT 'Active Personnel', active_personnel
UNION ALL
SELECT 'Warrants Issued', total_warrants_issued
UNION ALL
SELECT 'Average Case Duration ', avg_case_duration
UNION ALL
SELECT '---', '---';
END $$
DELIMITER ;
```

To call the procedure:

```
CALL generate_monthly_report(8,2024);
```

Purpose:

This procedure returns a monthly report consisting of all the opened cases, closed cases, evidence collected, lab analysis conducted, active personnel, warrants issued, and the average case duration of the given month.

Features:

- Uses multiple queries to get all desired values
- Returns a structured table-like report.

Output:

	metric	v
▶	METRIC	VALUE
	---	---
	Cases Opened	2
	Cases Closed	1
	Evidence Collected	6
	Evidence Analyzed	2
	Active Personnel	32
	Warrants Issued	5
	Average Case Duration	109
	---	---

Query 19: Nested query in Select and From statements, Set cardinality, Outer Join

```
Select evidence_ID, evidence_type,
CASE
    WHEN custody_count = 0 THEN 'MISSING'
    WHEN custody_count = 1 THEN 'INCOMPLETE'
    WHEN custody_count = 2 THEN 'MINIMAL'
    WHEN custody_count >= 3 THEN 'GOOD'
END as chain_status, custody_count,
CASE
    WHEN EXISTS(
        SELECT 1
        FROM Has H
        JOIN Chain_of_custody Ch USING (custody_ID)
        WHERE H.evidence_ID = E.evidence_ID AND
        Ch.transfer_reason = 'sent to lab')
        THEN 'Lab Transfer Found'
    ELSE 'No Lab Transfer'
END AS lab_transfer
From Evidence_items E
JOIN(
    SELECT evidence_ID, Count(*) AS custody_count
    FROM Has
    GROUP BY evidence_ID
    ) EC USING (evidence_ID)
ORDER BY custody_count;
```

Purpose:

This query returns the number of custody transfers for each evidence item that has changed custody as well as the chain status based on that number. It also returns the lab transfer status for each of these evidence items.

Features:

- CASE statements categorize custody count into status levels (MISSING, INCOMPLETE, MINIMAL, GOOD)
- Calculates the custody transfer count per evidence item
- Checks for specific lab transfer records using EXISTS
- Combines evidence details with custody count data with JOIN
- Sorts by custody count for priority assessment

Output:

evidence_ID	evidence_type	chain_status	custody_count	lab_transfer
3	biological sample	INCOMPLETE	1	No Lab Transfer
6	digital file	INCOMPLETE	1	No Lab Transfer
9	physical	INCOMPLETE	1	No Lab Transfer
11	chemical	INCOMPLETE	1	No Lab Transfer
12	weapon	INCOMPLETE	1	No Lab Transfer
1	weapon	MINIMAL	2	Lab Transfer Found
2	digital file	MINIMAL	2	No Lab Transfer
4	physical	MINIMAL	2	No Lab Transfer
5	chemical	MINIMAL	2	No Lab Transfer
7	weapon	MINIMAL	2	Lab Transfer Found
8	biological sample	MINIMAL	2	No Lab Transfer
10	digital file	MINIMAL	2	No Lab Transfer

Conclusion

Results and Findings Summary

The Forensic Investigation Database project has successfully delivered a comprehensive and fully functional database system that effectively models the complex workflows of a modern forensic investigation unit. The implementation encompasses twenty interconnected tables that capture the entire investigative lifecycle, from initial case registration through evidence collection, laboratory analysis, chain of custody tracking, and judicial oversight. The database has been populated with over three hundred realistic records that demonstrate practical forensic scenarios, including open, pending, and closed cases across various crime types such as homicide, cybercrime, assault, and burglary.

A key achievement of this project is the robust maintenance of referential integrity through carefully designed constraints that prevent data anomalies while supporting the intricate relationships inherent in forensic investigations. The database successfully implements specialized forensic requirements, including automatic case status calculation based on temporal data, comprehensive chain of custody tracking through the Has and Part_of relationship tables, and proper modeling of personnel specialization through ISA hierarchies. The system captures realistic temporal progressions, with evidence collection dates logically preceding laboratory analyses and warrant issue dates aligning with case investigation timelines.

Methodological Improvements and Recommendations

Several enhancements could strengthen the database implementation for real-world forensic use. Implementing temporal tracking features would provide automatic audit trails for critical data changes, particularly for chain of custody records and case status updates, which are essential for legal compliance and evidence integrity verification.

The database would benefit from stored procedures that standardize common forensic workflows, such as evidence transfer protocols, case assignment processes, and analysis request systems. This would ensure consistent data handling while reducing application complexity.

Performance could be improved through strategic indexing on frequently queried date ranges and status fields, along with

materialized views for common management reports like case backlogs and evidence aging analyses. Adding database-level encryption for sensitive contact information and evidence details would enhance security compliance.

Finally, implementing automated backup strategies with record preservation guidelines for closed cases would ensure long-term performance while maintaining legal record-keeping requirements. These improvements would transform the database into a more robust, production-ready forensic management system.