**COMSATS UNIVERSITY ISLAMABAD, ABBOTABAD**

**Assignment: Software Quality Engineering**

**Submitted To:**  **Nighat Usman**

**Task: "Group Project"**

**Submitted By:** **JALAL KHAN**

**Haider Rehman**

**REG NO: FA22-BSE-093**

**FA22-BSE-064**

**Department of Software Engineering**

**BSE5B**

**Police Station Management System** **project:**

**PURPOSE, ATTRIBUTES, AND DATA TYPES:**

**1. PoliceStation**

**Purpose**: Stores details about police stations. Each station is uniquely identified by its StationID.

**Attributes**:

* StationID (INT, Primary Key): Unique identifier for each station.
* StationName (VARCHAR(100)): Name of the police station.
* StationAddress (TEXT): Address of the police station.
* City (VARCHAR(50)): City where the station is located.
* StationPhone (VARCHAR(15)): Contact number of the station.
* StationEmail (VARCHAR(100)): Email address of the station.

**2. PoliceIncharge**

**Purpose**: Tracks the officer in charge of each police station.

**Attributes**:

* InchargeID (INT, Primary Key): Unique identifier for the in-charge officer.
* InchargeName (VARCHAR(100)): Name of the in-charge officer.
* InchargeContact (VARCHAR(15)): Contact number of the in-charge officer.
* StationID (INT, Foreign Key): ID of the station they are assigned to.

**3. Officer**

**Purpose**: Stores personal details of all officers in the system.

**Attributes**:

* OfficerID (INT, Primary Key): Unique identifier for each officer.
* OfficerName (VARCHAR(100)): Full name of the officer.
* OfficerRank (VARCHAR(50)): Rank of the officer (e.g., Inspector, Sub-Inspector).
* OfficerContact (VARCHAR(15)): Contact number of the officer.
* DateOfJoining (DATE): Date the officer joined the force.
* OfficerAddress (TEXT): Residential address of the officer.
* OfficerCity (VARCHAR(50)): City where the officer resides.

**4. OfficerAssignment**

**Purpose**: Maps officers to the police stations where they are assigned.

**Attributes**:

* OfficerID (INT, Foreign Key): ID of the officer.
* StationID (INT, Foreign Key): ID of the station.
* AssignedDate (DATE): Date the officer was assigned to the station.

**5. Criminal**

**Purpose**: Stores personal details of criminals.

**Attributes**:

* CriminalID (INT, Primary Key): Unique identifier for each criminal.
* CriminalName (VARCHAR(100)): Full name of the criminal.
* CriminalDOB (DATE): Date of birth of the criminal.

**6. Crime**

**Purpose**: Tracks crimes committed by criminals, including details of the crime and arresting officers.

**Attributes**:

* CrimeID (INT, Primary Key): Unique identifier for each crime.
* CrimeDetails (TEXT): Description of the crime.
* CrimeDate (DATE): Date the crime was committed.
* StationID (INT, Foreign Key): ID of the station handling the crime.
* ArrestingOfficerID (INT, Foreign Key): ID of the officer who made the arrest.
* CriminalID (INT, Foreign Key): ID of the criminal involved.

**7. Case**

**Purpose**: Maintains details of all cases registered in the system.

**Attributes**:

* CaseID (INT, Primary Key): Unique identifier for each case.
* CaseTitle (VARCHAR(100)): Brief title of the case.
* CaseType (VARCHAR(50)): Type of the case (e.g., Theft, Murder).
* CaseDetails (TEXT): Detailed description of the case.
* CaseStatus (VARCHAR(50)): Current status (e.g., Open, Closed).
* DateReported (DATE): Date the case was reported.

**8. CaseAssignment**

**Purpose**: Tracks which officer is assigned to investigate a case.

**Attributes**:

* CaseID (INT, Foreign Key): ID of the case.
* InvestigatingOfficerID (INT, Foreign Key): ID of the investigating officer.

**9. FIR**

**Purpose**: Records all First Information Reports (FIRs) filed by complainants.

**Attributes**:

* FIRID (INT, Primary Key): Unique identifier for each FIR.
* ComplainantID (INT, Foreign Key): ID of the complainant filing the FIR.
* FIRDetails (TEXT): Detailed description of the FIR.
* FIRDate (DATE): Date the FIR was filed.
* StationID (INT, Foreign Key): ID of the police station where the FIR was filed.

**10. Complainant**

**Purpose**: Stores information about complainants.

**Attributes**:

* ComplainantID (INT, Primary Key): Unique identifier for each complainant.
* ComplainantName (VARCHAR(100)): Full name of the complainant.
* ComplainantContact (VARCHAR(15)): Contact number of the complainant.

**11. Complaint**

**Purpose**: Tracks complaints filed by citizens at the police station.

**Attributes**:

* ComplaintID (INT, Primary Key): Unique identifier for each complaint.
* ComplaintTitle (VARCHAR(100)): Brief title of the complaint.
* ComplaintDetails (TEXT): Detailed description of the complaint.
* ComplainantID (INT, Foreign Key): ID of the complainant who filed the complaint.
* StationID (INT, Foreign Key): ID of the station where the complaint was filed.

**12. Evidence**

**Purpose**: Tracks evidence collected for cases.

**Attributes**:

* EvidenceID (INT, Primary Key): Unique identifier for each piece of evidence.
* EvidenceType (VARCHAR(50)): Type of evidence (e.g., Weapon, Document).
* EvidenceDetails (TEXT): Description of the evidence.
* CaseID (INT, Foreign Key): ID of the case the evidence belongs to.
* OfficerID (INT, Foreign Key): ID of the officer handling the evidence.

**13. StationOfficer**

**Purpose**: Maps police stations to their officers.

**Attributes**:

* StationID (INT, Foreign Key): ID of the station.
* OfficerID (INT, Foreign Key): ID of the officer assigned to the station.

**14. ArrestRecord**

**Purpose**: Logs arrests made by officers.

**Attributes**:

* ArrestID (INT, Primary Key): Unique identifier for each arrest.
* CriminalID (INT, Foreign Key): ID of the criminal arrested.
* ArrestingOfficerID (INT, Foreign Key): ID of the arresting officer.
* CrimeID (INT, Foreign Key): ID of the crime associated with the arrest.
* ArrestDate (DATE): Date the arrest was made.

**15. CriminalCaseMapping**

**Purpose**: Maps criminals to cases they are involved in.

**Attributes**:

* CaseID (INT, Foreign Key): ID of the case.
* CriminalID (INT, Foreign Key): ID of the criminal.

**16. CaseStationMapping**

**Purpose**: Tracks which police station is handling a particular case.

**Attributes**:

* CaseID (INT, Foreign Key): ID of the case.
* StationID (INT, Foreign Key): ID of the police station.

**Step-by-step process of normalization** TO **3NF (Third Normal Form):**

**Step 1: Unnormalized Data (UNF)**

**Initial Combined Entities**

The unnormalized table combined attributes related to the following entities in one table:

* **Police Stations**
* **Officers**
* **Criminals**
* **Crimes**
* **FIRs**
* **Cases**
* **Complaints**

The resulting table had redundant and repeated data. For instance:

1. Multiple officers assigned to the same station would result in repeating station details.
2. Multiple FIRs or crimes for the same complainant would duplicate complainant information.

**Step 2: First Normal Form (1NF)**

**Objective: Remove repeating groups and ensure each column contains atomic values.**

**Splitting Process in 1NF:**

1. Each data group (e.g., police stations, officers, criminals, FIRs, etc.) was separated into distinct tables to eliminate redundancy and ensure atomicity.
2. The attributes were grouped logically, ensuring each table contains data related to a **single entity or topic**.

**Tables Created After 1NF:**

1. **PoliceStation**
2. **Officer**
3. **Criminal**
4. **FIR**
5. **Case**
6. **Complaint**

**Step 3: Second Normal Form (2NF)**

**Objective: Eliminate partial dependencies (attributes depending on part of the primary key).**

**Splitting Process in 2NF:**

1. Composite keys (if any) were identified, and attributes dependent on part of the key were moved to separate tables.
   * For example, attributes like StationName and StationAddress were dependent only on StationID in the combined table, so they were split into the **PoliceStation** table.
2. Relationship tables were introduced to handle **many-to-many** relationships.
   * For example, the assignment of officers to stations led to a new **OfficerAssignment** table.
   * Similarly, the connection between cases and investigating officers resulted in the creation of a **CaseAssignment** table.

**Tables Created After 2NF:**

1. **OfficerAssignment** (Mapping Officers to Stations)
2. **CaseAssignment** (Mapping Cases to Investigating Officers)
3. **ArrestRecord** (Tracking arrests of criminals by officers)
4. **StationOfficer** (Mapping Officers to Stations)

**Step 4: Third Normal Form (3NF)**

**Objective: Eliminate transitive dependencies (attributes dependent on non-primary attributes).**

**Splitting Process in 3NF:**

1. Attributes that depended on non-primary attributes were moved to their respective tables.

* For example, attributes like OfficerRank or OfficerContact, which were dependent on OfficerID but unrelated to StationID, were moved to the **Officer** table.

1. Entities were further split to ensure **each table represented a single topic**:

* A dedicated **Evidence** table was created to store evidence details linked to cases.
* A **Complainant** table was created to handle complainant details separately.

**Tables Created After 3NF:**

1. **Evidence** (Handles evidence details for cases)
2. **Complainant** (Handles complainant information separately)
3. **Crime** (Details of crimes linked to criminals)
4. **ArrestRecord** (Links arrests with officers and criminals)

**Step-by-Step Explanation of Splits**

**1. Police Station and Officer Data**

* **Problem**: Redundant station details (e.g., address, phone) were repeated for every officer assigned to the station.
* **Solution**:
  + - Created **PoliceStation** table for station-specific data.
    - Created **Officer** table for officer-specific data.
    - Introduced **StationOfficer** table to map officers to stations.

**2. Criminal and Crime Data**

* **Problem**: Multiple crimes committed by the same criminal caused duplication of criminal details.
* **Solution**:
  + Created **Criminal** table for criminal-specific data.
    - Created **Crime** table for crime details.
    - Introduced **ArrestRecord** table to link criminals, crimes, and arresting officers.

**3. Case and Officer Data**

* **Problem**: Each case was linked to multiple officers, causing duplication of officer and case details.
* **Solution**:
  + - Created **Case** table for case-specific data.
    - Introduced **CaseAssignment** table to map cases to officers.

**4. FIR and Complainant Data**

* **Problem**: FIRs filed by the same complainant resulted in repeated complainant information.
* **Solution**:
  + - Created **Complainant** table for complainant-specific data.
    - Created **FIR** table for FIR details.

**5. Evidence for Cases**

* **Problem**: Evidence details were embedded in cases, causing redundancy when multiple pieces of evidence were linked to a single case.
* **Solution**:
  + - Created **Evidence** table for evidence details linked to cases.

This normalization process ensures:

1. Elimination of redundancy.
2. Logical separation of entities.
3. Consistency and integrity of data.

**Buttons and Related Query:**

**1. Button Name: HighestRankOfficerButton**

**Purpose**: This query finds the name and rank of the officer with the highest rank.

**SQL Query**:

SELECT OfficerName, OfficerRank

FROM Officer

WHERE OfficerRank = (SELECT MAX(OfficerRank) FROM Officer)

**2. Button Name: Cases Solved Query**

**Purpose**: This query calculates the number of cases solved by each officer.

**SQL Query**:

SELECT OfficerName, COUNT(Crime.CrimeID) AS CasesSolved

FROM Officer

LEFT JOIN Crime ON Officer.OfficerID = Crime.ArrestingOfficerID

GROUP BY OfficerName;

**3. Button Name: Officers Assigned Query**

**Purpose**: This query retrieves the names of officers assigned to station with StationID = 1.

**SQL Query**:

SELECT OfficerName

FROM Officer

JOIN OfficerAssignment ON Officer.OfficerID = OfficerAssignment.OfficerID

WHERE OfficerAssignment.StationID = 1  
**4. Button Name: Officers Rank Query**

**Purpose**: This query selects the names of officers whose rank is ‘inspector’.

**SQL Query**:

SELECT OfficerName

FROM Officer

WHERE OfficerRank LIKE 'inspector'

**5. Button Name: Criminal Arrests Query**

**Purpose**: This query is similar to the "Cases Solved Query" but focuses on criminals.

**SQL Query**:

SELECT CriminalName, COUNT(ArrestRecord.ArrestID) AS ArrestCount

FROM Criminal

LEFT JOIN ArrestRecord ON Criminal.CriminalID = ArrestRecord.CriminalID

GROUP BY CriminalName

**6. Button Name: Date Range Case Query**

**Purpose**: This query selects the title and status of cases reported within a specific date range (the year 2023).

**SQL Query**:

SELECT CaseTitle, CaseStatus

FROM Cases

WHERE DateReported BETWEEN '2023-01-01' AND ‘2023-12-31’.

**7. Button Name: Multiple Stations Query**

**Purpose**: This query finds officers assigned to more than one police station.

**SQL Query**:

SELECT OfficerName, COUNT(DISTINCT StationID) AS StationsAssigned

FROM Officer

JOIN StationOfficer ON Officer.OfficerID = StationOfficer.OfficerID

GROUP BY OfficerName

HAVING COUNT(DISTINCT StationID) > 1

**8. Button Name: Longest Service Query**

**Purpose**: This query retrieves the name and joining date of the officer with the longest service.**SQL Query**:

SELECT TOP 1 OfficerName, DateOfJoining

FROM Officer

ORDER BY DateOfJoining ASC

**9. Button Name: Crimes per Station Query**

**Purpose**: This query determines the total number of crimes recorded at each police station.

**SQL Query**:

SELECT StationName, COUNT(CrimeID) AS TotalCrimes

FROM PoliceStation

LEFT JOIN Crime ON PoliceStation.StationID = Crime.StationID

GROUP BY StationName;

**10. Button Name: Cases Assigned to Officer**

**Purpose**: This query retrieves the title and type of cases assigned to the officer with ‘InvestigatingOfficerID = 2’.

**SQL Query**:

SELECT c.CaseTitle, c.CaseType FROM Cases c JOIN CaseAssignment ca ON c.CaseID = ca.CaseID WHERE ca.InvestigatingOfficerID = 2.

**11. Button Name: Officer and Criminal Union**

**Purpose**: This query combines the names of officers and criminals into a single result set. ‘UNION’ combines the results of two ‘SELECT’ statements, removing duplicate entries.

**SQL Query**:

SELECT OfficerName, 'Officer' AS Source FROM Officer UNION SELECT CriminalName, 'Criminal' AS Source FROM Criminal;

**12. Button Name: Multiple Arrests Query**

**Purpose**: This query identifies officers who have made more than one arrest.

**SQL Query**:

SELECT o.OfficerName, COUNT(a.ArrestID) AS ArrestCount FROM Officer o JOIN ArrestRecord a ON o.OfficerID = a.ArrestingOfficerID GROUP BY o.OfficerName HAVING COUNT(a.ArrestID) > 1;

**13. Button Name: Solved Case Query**

**Purpose**: This query retrieves details of solved cases, including the investigating officer and the criminal involved (if known), for cases reported after 01, 1, 2012.

**SQL Query**:

SELECT c.CaseTitle, c.CaseStatus, o.OfficerName AS InvestigatingOfficer, cr.CriminalName AS InvolvedCriminal FROM Cases c JOIN CaseAssignment ca ON c.CaseID = ca.CaseID JOIN Officer o ON ca.InvestigatingOfficerID = o.OfficerID LEFT JOIN Crime crime ON c.CaseID = crime.CrimeID LEFT JOIN Criminal cr ON crime.CriminalID = cr.CriminalID WHERE c.CaseStatus = 'Solved' AND c.DateReported > '2012-01-01'

**14. Button Name: Top 5 Officers Query**

**Purpose**: This query finds the top 5 officers with the most arrests.

**SQL Query**:

SELECT TOP 5 OfficerName, COUNT(ArrestRecord.ArrestID) AS ArrestCount FROM Officer LEFT JOIN ArrestRecord ON Officer.OfficerID = ArrestRecord.ArrestingOfficerID GROUP BY OfficerName ORDER BY ArrestCount DESC

**15. Button Name: Total Crimes and Cases Query**

**Purpose**: This query calculates the total number of crimes and cases associated with each police station.

**SQL Query**:

SELECT ps.StationName, COUNT(DISTINCT c.CrimeID) AS TotalCrimes, COUNT(DISTINCT cs.CaseID) AS TotalCases FROM PoliceStation ps LEFT JOIN Crime c ON ps.StationID = c.StationID LEFT JOIN CriminalCaseMapping ccm ON c.CriminalID = ccm.CriminalID LEFT JOIN Cases cs ON ccm.CaseID = cs.CaseID GROUP BY ps.StationName;

**16. Button Name: Assigned Multiple Times Query**

**Purpose**: This query finds officers who have been assigned to the same station multiple times.

**SQL Query**:

SELECT OfficerName, StationID FROM Officer JOIN OfficerAssignment ON Officer.OfficerID = OfficerAssignment.OfficerID GROUP BY OfficerName, StationID HAVING COUNT(StationID) > 1

**17. Button Name: Complaints per Station Query**

**Purpose**: This query counts the number of complaints filed at each police station.

**SQL Query**:

SELECT PoliceStation.StationName, COUNT(Complaint.ComplaintID) AS TotalComplaints FROM PoliceStation LEFT JOIN Complaint ON PoliceStation.StationID = Complaint.StationID GROUP BY PoliceStation.StationName **18. Button Name: Multiple Crimes Query**

**Purpose** This query identifies criminals involved in more than one crime.

**SQL Query**:

SELECT CriminalName, COUNT(DISTINCT CrimeID) AS CrimesInvolved FROM Criminal LEFT JOIN Crime ON Criminal.CriminalID = Crime.CriminalID GROUP BY CriminalName HAVING COUNT(DISTINCT CrimeID) > 1

**19. Button Name: No Assignment Query**

**Purpose**: This query finds officers who have not been assigned to any station.

**SQL Query**:

SELECT OfficerName FROM Officer LEFT JOIN OfficerAssignment ON Officer.OfficerID = OfficerAssignment.OfficerID WHERE OfficerAssignment.StationID IS NULL

**20. Button Name: Cases in Last 30 Days**

**Purpose**: This query retrieves the title and reporting date of cases reported in the last 30 days.

**SQL Query**:

SELECT CaseTitle, DateReported FROM Cases WHERE DateReported >= DATEADD(day, -30, GETDATE())SELECT CaseTitle, DateReported FROM Cases WHERE DateReported >= DATEADD(day, -30, GETDATE())