# SQL Scripts Documentation

## 1. Database Schema Definition Script

### Description

**The Create Table script sets up the database schema for a pet adoption and foster care system. It defines five tables:**

1. **FosterHomes - Stores information about foster homes.**
2. **Pets - Stores details of pets available for adoption.**
3. **Adopters - Contains information on individuals adopting pets.**
4. **MedicalRecords - Maintains health and medical records for pets.**
5. **Rescuers - Tracks individuals who rescue pets.**

### Schema Details

* **FosterHomes:**
  + **FosterHomeID: Unique identifier (Primary Key)**
  + **Name: Name of the foster home**
  + **Address: Address details**
  + **ContactPerson: Name of the contact person**
  + **ContactPhone: Contact number**
  + **Capacity: Maximum number of pets allowed**
  + **CurrentOccupancy: Current number of pets housed.**
* **Pets:**
  + **PetID: Unique identifier (Primary Key)**
  + **Name: Pet's name**
  + **Species: Type of pet (Dog, Cat etc.)**
  + **Breed: Breed of the pet**
  + **Age: Age of the pet**
  + **Gender: Gender (Male/Female)**
  + **Status: Availability for adoption**
  + **RescueDate: Date the pet was rescued.**
  + **AdoptionDate: Date the pet was adopted.**
  + **FosterHomeID: Foreign Key linking to FosterHomes.**
* **Adopters:**
  + **AdopterID: Unique identifier (Primary Key)**
  + **Name: Name of the adopter**
  + **Email: Unique email of the adopter**
  + **Phone: Contact number**
  + **Address: Address details**
* **MedicalRecords:**
  + **MedicalRecordID: Unique identifier (Primary Key)**
  + **PetID: Foreign Key linking to Pets.**
  + **VaccinationDate: Last vaccination date**
  + **DiseaseHistory: Record of past illnesses**
  + **IsVaccinated: Boolean indicating vaccination status.**
  + **Notes: Additional medical notes**
* **Rescuers:**
  + **RescuerID: Unique identifier (Primary Key)**
  + **Name: Name of the rescuer**
  + **Phone: Contact details**
  + **Organization: Organization (if applicable)**
  + **AdopterID: Foreign Key linking to Adopters.**
  + **PetID: Foreign Key linking to Pets.**
  + **RescueDate: Date of rescue**

## 2. Database Automation & Integrity

### Description

**The Insert Records script populates the tables with sample data, providing realistic entries for demonstration.**

### Data Entries

* **FosterHomes Table: 20 records with real-world Indian locations and contacts.**
* **Pets Table: 20 records linking pets to foster homes.**
* **Adopters Table: 20 records with realistic adopter information.**
* **MedicalRecords Table: 20 medical records covering health details.**
* **Rescuers Table: 20 rescuers assigned to various pets.**

## 3. Initial Data Insertion Script

### Description

**The Trigger Script ensures database consistency by implementing automatic updates and constraints.**

### Triggers Implemented

**Adoption Tracking**

* **Trigger: after\_status\_update**
* **Action: Automatically sets AdoptionDate when the pet's status changes to "Adopted."**

**Foster Home Occupancy Management**

* **Trigger: after\_pet\_insert**
* **Action: Increments CurrentOccupancy when a pet is assigned to a foster home.**
* **Trigger: after\_pet\_update**
* **Action: Updates CurrentOccupancy when a pet moves between foster homes.**
* **Trigger: after\_pet\_delete**
* **Action: Decreases CurrentOccupancy when a pet is removed.**
* **Trigger: before\_pet\_insert**
* **Action: Prevents over-occupancy by rejecting pets if CurrentOccupancy >= Capacity.**

**Cascading Deletions**

* **Trigger: after\_pet\_delete\_medical**
* **Action: Deletes associated MedicalRecords when a pet is removed.**
* **Trigger: after\_pet\_delete\_rescuer**
* **Action: Deletes associated Rescuers records when a pet is removed.**
* **Trigger: before\_foster\_delete**
* **Action: Prevents deletion of foster homes if CurrentOccupancy > 0.**

**Adopter Deletion Handling**

* **Trigger: after\_adopter\_delete**
* **Action: Sets AdopterID to NULL in Rescuers table if an adopter is removed to prevent orphan records.**

## 4. Optimized Data Retrieval & Querying

### Description

**The Data Retrieval Script contains optimized queries to fetch relevant information from the database efficiently. These queries use JOINs, WHERE, LIKE, ORDER BY, GROUP BY, HAVING, and indexing for enhanced performance.**

### Queries Implemented

* **Retrieve all pets and their details.**
* **Retrieve only available pets.**
* **Retrieve pets younger than a specified age.**
* **Retrieve pets whose names match specific patterns using LIKE.**
* **Retrieve pets sorted by name, age, or species.**
* **Retrieve all available pets with their foster home details.**
* **Retrieve pets and their adopter details (if adopted) by correctly joining the Rescuers table.**
* **Count the number of pets per foster home.**
* **Count pets by species and filter species with more than 5 pets.**
* **Find adopters who adopted pets within the last 6 months.**
* **Retrieve medical history of unvaccinated pets.**
* **Optimize queries using indexes on commonly searched fields (e.g., Pet Name, Adoption Date).**

## 5. Database Normalization & Constraints Enforcement

### Description

The **Normalization Script** ensures that the database follows **1NF, 2NF, and 3NF** while enforcing data integrity through constraints.

### 1NF (First Normal Form) – Ensuring Atomicity

**What Happens?**

* In **1NF**, we ensure that each column contains **atomic values** (single values, no lists or repeated data).
* **Example Fix in Pets Table**:
  + Previously, Species was stored **as a string** in the Pets table.
  + **Problem**: If multiple pets belong to the same species, storing the full species name repeatedly wastes space.
  + **Solution**: We created a **new table Species** and replaced Species in Pets with a SpeciesID foreign key.

**Tables Affected:**

* **Pets table is modified**: Species VARCHAR(30) → SpeciesID INT (FK)
* **Species table is created**: SpeciesID INT (PK), SpeciesName VARCHAR(30) UNIQUE NOT NULL

### 2NF (Second Normal Form) – Removing Partial Dependencies

**What Happens?**

* In **2NF**, we remove **partial dependencies**, meaning every non-key attribute must depend **entirely on the primary key**.
* **Example Fix in FosterHomes Table**:
  + The FosterHomes table contained ContactPerson and ContactPhone.
  + **Problem**: The contact person **isn’t dependent on the FosterHomeID**, as multiple foster homes could have the same contact.
  + **Solution**: We **moved ContactPerson and ContactPhone into a new ContactDetails table** and linked it with a ContactID.

**Tables Affected:**

* **FosterHomes table is modified**:
  + ContactPerson VARCHAR(100) NOT NULL → Moved to a new table.
  + ContactPhone VARCHAR(15) NOT NULL → Moved to a new table.
  + **Added ContactID** as a **foreign key**.
* **ContactDetails table is created**:
  + ContactID INT PRIMARY KEY
  + ContactPerson VARCHAR(100) NOT NULL
  + ContactPhone VARCHAR(15) UNIQUE NOT NULL

### Summary of What Changed

| **Normalization Step** | **What Changed?** | **Which Table is Affected?** |
| --- | --- | --- |
| **1NF (Atomicity)** | Species moved to a separate table | Pets, Species |
| **2NF (No Partial Dependencies)** | ContactPerson & ContactPhone moved to ContactDetails | FosterHomes, ContactDetails |

**Key Improvements**

* **1NF:** Ensures atomicity of values and removes redundant data.
* **2NF:** Eliminates partial dependencies by moving species data into a separate table.
* **3NF:** Removes transitive dependencies by creating a separate ContactDetails table.
* **Foreign Key Constraints:** Prevent orphan records and maintain relationships.
* **CHECK Constraints:** Validate data (e.g., pet age must be ≥ 0, capacity must be ≥ current occupancy).
* **Indexing:** Improves performance on frequently queried columns.

## 6. Comprehensive CRUD Operations

### Updating Records

* **Update pet's adoption status and set adoption date**
* **UPDATE Pets**
* **SET Status = 'Adopted', AdoptionDate = CURDATE()**
* **WHERE PetID = 101;**
* **Update foster home occupancy after a pet is moved**
* **UPDATE FosterHomes**
* **SET CurrentOccupancy = CurrentOccupancy + 1**
* **WHERE FosterHomeID = (SELECT FosterHomeID FROM Pets WHERE PetID = 101);**
* **Update adopter’s contact details**
* **UPDATE Adopters**
* **SET Phone = '9876543210', Email = 'new\_email@example.com'**
* **WHERE AdopterID = 10;**
* **Update rescuer’s organization details**
* **UPDATE Rescuers**
* **SET Organization = 'Animal Care Foundation'**
* **WHERE RescuerID = 5;**

### Deleting Records

* **Delete old medical records of pets that have been adopted for over a year**
* **DELETE FROM MedicalRecords**
* **WHERE PetID IN (SELECT PetID FROM Pets WHERE Status = 'Adopted' AND AdoptionDate < DATE\_SUB(CURDATE(), INTERVAL 1 YEAR));**
* **Delete pets marked as 'Deceased' from the database**
* **DELETE FROM Pets**
* **WHERE Status = 'Deceased';**

### Inserting New Records

* **Insert a new pet into the database**
* **INSERT INTO Pets (PetID, Name, Species, Breed, Age, Gender, Status, RescueDate, FosterHomeID)**
* **VALUES (201, 'Buddy', 'Dog', 'Labrador', 2, 'Male', 'Available', CURDATE(), 3);**
* **Insert a new adopter**
* **INSERT INTO Adopters (AdopterID, Name, Email, Phone, Address)**
* **VALUES (25, 'John Doe', 'john.doe@example.com', '9988776655', '123 Main Street, Cityville');**
* **Insert a new foster home**
* **INSERT INTO FosterHomes (FosterHomeID, Name, Address, ContactPerson, ContactPhone, Capacity, CurrentOccupancy)**
* **VALUES (10, 'Happy Paws Shelter', '45 Elm Street, PetTown', 'Alice Johnson', '9090909090', 15, 5);**
* **Insert a new rescuer record**
* **INSERT INTO Rescuers (RescuerID, Name, Phone, Organization, AdopterID, PetID, RescueDate)**
* **VALUES (15, 'Sarah Connor', '9234567890', 'Pet Rescue Initiative', 25, 201, CURDATE());**

## 7. Database Schema Modification & Optimization

### Purpose of DDL ALTER Script

**The DDL ALTER script was created to:**

* **Modify and optimize the database structure without recreating tables.**
* **Add constraints and enforce data integrity using foreign keys and check constraints.**
* **Improve performance by adding indexes.**
* **Ensure referential integrity by linking related tables.**

### Key Modifications in DDL ALTER Script

1. **Adding New Columns**
   * **Added AdoptionFee to Adopters table.**
   * **Added ContactID to FosterHomes to link contact details.**
2. **Modifying Data Types**
   * **Changed ContactPhone in FosterHomes to allow longer international formats.**
3. **Adding Constraints**
   * **Ensured SpeciesID in Pets cannot be NULL.**
   * **Implemented CHECK constraints for Age in Pets and Capacity in FosterHomes.**
4. **Renaming Tables & Columns**
   * **Renamed ContactPerson to ManagerName in FosterHomes.**
   * **Renamed Species table to PetSpecies for clarity.**
5. **Foreign Key Enhancements**
   * **Linked CustomerFeedback with Adopters using a foreign key.**
   * **Ensured Rescuers.AdopterID and Rescuers.PetID set to NULL if referenced records are deleted.**
6. **Indexing for Performance**
   * **Added indexes to Pets(Name), Adopters(Email), and FosterHomes(Capacity).**

## 8. Stress Testing Script

### 1. Overview

**This document provides a detailed breakdown of the stress testing script used for evaluating database performance, integrity, and efficiency. The script is designed to stress test core functionalities, including bulk data insertion, updates, constraints enforcement, and referential integrity in the database schema.**

### 2. Components of the Stress Testing Script

**Step 1: Bulk Insert for FosterHomes**

* **Purpose: Populate the FosterHomes table with randomized but structured data.**
* **Key Operations:** 
  + **Generates FosterHomeID using ROW\_NUMBER().**
  + **Assigns random values for Name, Address, Capacity, CurrentOccupancy, and ManagerName.**
  + **Ensures all foster homes start with zero occupancy.**
* **SQL Snippet:**
* **INSERT INTO FosterHomes (FosterHomeID, Name, Address, Capacity, CurrentOccupancy, ContactID, ManagerName)**
* **SELECT ROW\_NUMBER() OVER () + 10000, CONCAT('FosterHome\_', ROW\_NUMBER() OVER ()), CONCAT('Address\_', ROW\_NUMBER() OVER ()),**
* **FLOOR(RAND() \* 500) + 500, 0, FLOOR(RAND() \* 100) + 1, CONCAT('Manager\_', ROW\_NUMBER() OVER ())**
* **FROM (SELECT \* FROM information\_schema.tables LIMIT 5000) AS T;**

**Step 2: Adjust Foster Home Capacities and Update Occupancy**

* **Purpose: Randomly adjust foster home capacities and update current occupancy counts.**
* **Key Operations:** 
  + **Uses a temporary table to manage capacity updates.**
  + **Ensures that foster home occupancy does not exceed its capacity.**
* **SQL Snippet:**
* **UPDATE FosterHomes FH**
* **JOIN (SELECT FosterHomeID FROM FosterHomes ORDER BY RAND() LIMIT 2500) AS Temp**
* **ON FH.FosterHomeID = Temp.FosterHomeID**
* **SET FH.CurrentOccupancy = FLOOR(RAND() \* FH.Capacity);**

**Step 3: Create a Temporary Table for Available Foster Homes**

* **Purpose: Identify foster homes with available space before inserting pets.**
* **Key Operations:** 
  + **Filters FosterHomes where CurrentOccupancy < Capacity.**
  + **Creates a temporary table TempAvailableFosterHomes.**

**Step 4: Bulk Insert for Pets**

* **Purpose: Populate the Pets table with randomized pet data linked to foster homes.**
* **Key Operations:** 
  + **Ensures pets are only assigned to available foster homes.**
  + **Randomly generates attributes such as SpeciesID, Breed, Age, and Status.**
  + **Assigns valid RescueDate and AdoptionDate values.**
* **SQL Snippet:**
* **INSERT INTO Pets (PetID, Name, SpeciesID, Breed, Age, Gender, Status, RescueDate, AdoptionDate, Species, FosterHomeID)**
* **SELECT ROW\_NUMBER() OVER () + 10000, CONCAT('Pet\_', ROW\_NUMBER() OVER ()), FLOOR(RAND() \* 10) + 1,**
* **CONCAT('Breed\_', FLOOR(RAND() \* 10)), FLOOR(RAND() \* 15),**
* **CASE WHEN RAND() > 0.5 THEN 'Male' ELSE 'Female' END,**
* **CASE WHEN RAND() > 0.5 THEN 'Available' ELSE 'Adopted' END,**
* **CAST(DATE\_SUB(CURDATE(), INTERVAL FLOOR(RAND() \* 1000) DAY) AS DATE),**
* **NULL, 'Unknown', (SELECT FosterHomeID FROM TempAvailableFosterHomes ORDER BY RAND() LIMIT 1)**
* **FROM Pets ORDER BY RAND() LIMIT 10000;**

**Step 5: Drop Temporary Tables**

* **Purpose: Clean up temporary tables after use to free up memory.**
* **Key Operations:** 
  + **Drops TempAvailableFosterHomes.**

**Step 6: Bulk Insert for Adopters**

* **Purpose: Populate the Adopters table with sample adopter data.**
* **Key Operations:** 
  + **Uses CONCAT() to generate random names and addresses.**
  + **Ensures unique AdopterID values.**
* **SQL Snippet:**
* **INSERT INTO Adopters (AdopterID, Name, Email, Phone, Address, AdoptionFee)**
* **SELECT ROW\_NUMBER() OVER () + 10000, CONCAT('Adopter\_', ROW\_NUMBER() OVER ()),**
* **CONCAT('adopter', ROW\_NUMBER() OVER (), '@example.com'),**
* **CONCAT('+91', FLOOR(RAND() \* 10000000000)),**
* **CONCAT('Address\_', ROW\_NUMBER() OVER ()), ROUND(RAND() \* 5000, 2)**
* **FROM Pets ORDER BY RAND() LIMIT 10000;**

**Step 7: Bulk Insert for MedicalRecords**

* **Purpose: Assign medical records to pets while ensuring uniqueness.**
* **Key Operations:** 
  + **Ensures that PetID values do not duplicate existing records.**
  + **Generates random VaccinationDate, DiseaseHistory, and IsVaccinated values.**
* **SQL Snippet:**
* **INSERT INTO MedicalRecords (MedicalRecordID, PetID, VaccinationDate, DiseaseHistory, IsVaccinated, Notes)**
* **SELECT ROW\_NUMBER() OVER () + 10000, PetID,**
* **CAST(DATE\_SUB(CURDATE(), INTERVAL FLOOR(RAND() \* 500) DAY) AS DATE),**
* **CONCAT('History\_', FLOOR(RAND() \* 10)),**
* **FLOOR(RAND() \* 2), CONCAT('Notes\_', FLOOR(RAND() \* 10))**
* **FROM Pets WHERE PetID NOT IN (SELECT PetID FROM MedicalRecords) ORDER BY RAND() LIMIT 10000;**

### 3. Expected Behavior and Results

* **Performance Testing: Ensures that bulk inserts, updates, and integrity constraints function efficiently under load.**
* **Referential Integrity: Ensures that all foreign key constraints are maintained.**
* **Trigger Testing: Tests the functionality of triggers related to adoption tracking and foster home capacity management.**
* **Error Handling: The script prevents duplicate entries and invalid data insertions.**

### 4. Conclusion

**The stress testing script effectively simulates a high-load scenario to validate the efficiency and reliability of the database. It ensures that all tables maintain referential integrity while performing large-scale data operations.**

**End of Documentation**