DATABASE MANAGEMENT SYSTEM

REAL ESTATE MANAGEMENT SYSTEM

SYED AHMED ALI	DT-22301
MUHAMMAD KHUZAIMA	DT-22302
EBAAD KHAN	DT-22045
EZAAN KHAN	DT-22046

INTRODUCTION

This project involves designing and implementing a database for a real estate management system. The database facilitates the management of property listings, transactions, tenants, and related activities. The objective is to create a robust, normalized database schema up to the Third Normal Form (3NF) to ensure data integrity and reduce redundancy. The front end of the application, developed using Python, interacts with this database to provide a user-friendly interface for managing real estate operations.

NORMALIZATION

Normalization involves organizing the attributes and tables of a relational database to minimize redundancy and dependency. The process typically involves the following steps:

1NF(First Normal Form): Ensuring that each table has atomic (indivisible) values and each column contains values of a single type.

2NF(Second Normal Form): Ensuring that each non primary key attribute is fully functionally dependent on the primary key.

3NF(Third Normal Form): Ensuring that there are no transitive dependencies, i.e., non primary key attributes are not dependent on other non primary key
attributes.

NORMALIZATION

OWNER TABLE

1NF: ALL VALUES ARE ATOMIC AND THERE IS A PRIMARY KEY.

2NF ALL NON KEY ATTRIBUTES DEPEND ON OWNER_ ID.

3NF THERE ARE NO TRANSITIVE DEPENDENCIES.

OwnerID	Name	ContactInfo	Email
1	John Doe	123-456-7890	john.doe@example.com
2	Jane Smith	234-567-8901	jane.smith@example.com
3	Michael Johnson	345-678-9012	michael.johnson@example.com
4	Patricia Brown	456-789-0123	patricia.brown@example.com
5	Robert Davis	567-890-1234	robert.davis@example.com
6	Linda Martinez	678-901-2345	linda.martinez@example.com
7	William Garcia	789-012-3456	william.garcia@example.com
8	Elizabeth Wilson	890-123-4567	elizabeth.wilson@example.com
9	Charles Anderson	901-234-5678	charles.anderson@example.com
10	Barbara Taylor	012-345-6789	barbara.taylor@example.com
NULL	NULL	NULL	NULL

NORMALIZATION

LOCATION TABLE

1NF: ALL VALUES ARE ATOMIC AND THERE IS A PRIMARY KEY.

2NF ALL NON KEY ATTRIBUTES DEPEND ON LOCATION_ ID.

3NF THERE ARE NO TRANSITIVE DEPENDENCIES.

LocationID	Address	City	State	ZipCode
1	123 Elm St	Springfield	IL	62701
2	456 Oak St	Chicago	IL	60601
3	789 Pine St	Los Angeles	CA	90001
4	101 Maple St	New York	NY	10001
5	202 Birch St	Houston	TX	77001
6	303 Cedar St	Phoenix	AZ	85001
7	404 Walnut St	Philadelphia	PA	19019
8	505 Ash St	San Antonio	TX	78201
9	606 Willow St	San Diego	CA	92101
10	707 Hickory St	Dallas	TX	75201
NULL	NULL	NULL	NULL	NULL

NORMALIZATION

PROPERTY TABLE

1NF: ALL VALUES ARE ATOMIC AND THERE IS A PRIMARY KEY.

2NF ALL NON KEY ATTRIBUTES DEPEND ON PROPERTY_ ID.

3NF THERE ARE NO TRANSITIVE DEPENDENCIES.

PropertyID	Address	Price	LocationID	OwnerID
1	123 Elm St	250000.00	1	1
2	456 Oak St	300000.00	2	2
3	789 Pine St	150000.00	3	3
4	101 Maple St	350000.00	4	4
5	202 Birch St	275000.00	5	5
6	303 Cedar St	325000.00	6	6
7	404 Walnut St	200000.00	7	7
8	505 Ash St	225000.00	8	8
9	606 Willow St	180000.00	9	9
10	707 Hickory St	400000.00	10	10
11	123 Elm St	250000.00	1	1
NULL	NULL	NULL	NULL	NULL

DATA DEFINITION LANGUAGE

```sql
CREATE TABLE Owner (
    OwnerID INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    ContactInfo VARCHAR(150),
    Email VARCHAR(100) UNIQUE
);
```

OwnerID	Name	ContactInfo	Email
1	John Doe	123-456-7890	john.doe@example.com
2	Jane Smith	234-567-8901	jane.smith@example.com
3	Michael Johnson	345-678-9012	michael.johnson@example.com
4	Patricia Brown	456-789-0123	patricia.brown@example.com
5	Robert Davis	567-890-1234	robert.davis@example.com
6	Linda Martinez	678-901-2345	linda.martinez@example.com
7	William Garcia	789-012-3456	william.garcia@example.com
8	Elizabeth Wilson	890-123-4567	elizabeth.wilson@example.com
9	Charles Anderson	901-234-5678	charles.anderson@example.com
10	Barbara Taylor	012-345-6789	barbara.taylor@example.com
NULL	NULL	NULL	NULL

```sql
CREATE TABLE Location (
    LocationID INT AUTO_INCREMENT PRIMARY KEY,
    Address VARCHAR(150) NOT NULL,
    City VARCHAR(100) NOT NULL,
    State VARCHAR(100) NOT NULL,
    ZipCode VARCHAR(10) NOT NULL
);
```

LocationID	Address	City	State	ZipCode
1	123 Elm St	Springfield	IL	62701
2	456 Oak St	Chicago	IL	60601
3	789 Pine St	Los Angeles	CA	90001
4	101 Maple St	New York	NY	10001
5	202 Birch St	Houston	TX	77001
6	303 Cedar St	Phoenix	AZ	85001
7	404 Walnut St	Philadelphia	PA	19019
8	505 Ash St	San Antonio	TX	78201
9	606 Willow St	San Diego	CA	92101
10	707 Hickory St	Dallas	TX	75201
NULL	NULL	NULL	NULL	NULL

DATA DEFINITION LANGUAGE

```sql
CREATE TABLE Property (
    PropertyID INT AUTO_INCREMENT PRIMARY KEY,
    Address VARCHAR(150) NOT NULL,
    Price DECIMAL(15, 2) NOT NULL,
    LocationID INT,
    OwnerID INT,
    FOREIGN KEY (OwnerID) REFERENCES Owner(OwnerID) ON DELETE CASCADE,
    FOREIGN KEY (LocationID) REFERENCES Location(LocationID) ON DELETE CASCADE
);
```

PropertyID	Address	Price	LocationID	OwnerID
1	123 Elm St	250000.00	1	1
2	456 Oak St	300000.00	2	2
3	789 Pine St	150000.00	3	3
4	101 Maple St	350000.00	4	4
5	202 Birch St	275000.00	5	5
6	303 Cedar St	325000.00	6	6
7	404 Walnut St	200000.00	7	7
8	505 Ash St	225000.00	8	8
9	606 Willow St	180000.00	9	9
10	707 Hickory St	400000.00	10	10
11	123 Elm St	250000.00	1	1
NULL	NULL	NULL	NULL	NULL

DATA MANIPULATION LANGUAGE

```sql
-- Inserting sample data into Owner table
INSERT INTO Owner (Name, ContactInfo, Email) VALUES
('John Doe', '123-456-7890', 'john.doe@example.com'),
('Jane Smith', '234-567-8901', 'jane.smith@example.com'),
('Michael Johnson', '345-678-9012', 'michael.johnson@example.com'),
('Patricia Brown', '456-789-0123', 'patricia.brown@example.com'),
('Robert Davis', '567-890-1234', 'robert.davis@example.com'),
('Linda Martinez', '678-901-2345', 'linda.martinez@example.com'),
('William Garcia', '789-012-3456', 'william.garcia@example.com'),
('Elizabeth Wilson', '890-123-4567', 'elizabeth.wilson@example.com'),
('Charles Anderson', '901-234-5678', 'charles.anderson@example.com'),
('Barbara Taylor', '012-345-6789', 'barbara.taylor@example.com');
```

```sql
-- Inserting sample data into Property table
INSERT INTO Property (Address, Price, OwnerID, LocationID) VALUES
('123 Elm St', 250000.00, 1, 1),
('456 Oak St', 300000.00, 2, 2),
('789 Pine St', 150000.00, 3, 3),
('101 Maple St', 350000.00, 4, 4),
('202 Birch St', 275000.00, 5, 5),
('303 Cedar St', 325000.00, 6, 6),
('404 Walnut St', 200000.00, 7, 7),
('505 Ash St', 225000.00, 8, 8),
('606 Willow St', 180000.00, 9, 9),
('707 Hickory St', 400000.00, 10, 10);
```

```sql
-- Inserting sample data into Location table
INSERT INTO Location (Address, City, State, ZipCode) VALUES
('123 Elm St', 'Springfield', 'IL', '62701'),
('456 Oak St', 'Chicago', 'IL', '60601'),
('789 Pine St', 'Los Angeles', 'CA', '90001'),
('101 Maple St', 'New York', 'NY', '10001'),
('202 Birch St', 'Houston', 'TX', '77001'),
('303 Cedar St', 'Phoenix', 'AZ', '85001'),
('404 Walnut St', 'Philadelphia', 'PA', '19019'),
('505 Ash St', 'San Antonio', 'TX', '78201'),
('606 Willow St', 'San Diego', 'CA', '92101'),
('707 Hickory St', 'Dallas', 'TX', '75201');
```

DATA MANIPULATION LANGUAGE

```
354    -- UPDATE OPERATIONS
355
356 •  UPDATE Owner
357    SET ContactInfo = '999-999-9999'
358    WHERE OwnerID = 1;
359
360 •  UPDATE Property SET Price = 500000 WHERE PropertyID = 1;
361


367    -- Delete OPERATIONS
368
369 •  DELETE FROM Tenant WHERE TenantID = 1;
370
371    -- Select OPERATIONS
372 •  SELECT * FROM Agent WHERE City = 'New York';
373
```

JOINS

```
235
236  -- Retrieve all details for properties, including owner, location, and agent details for current listings:
237
238  SELECT
239      p.PropertyID,
240      p.Address,
241      p.Price,
242      o.Name AS OwnerName,
243      l.City,
244      l.State,
245      a.Name AS AgentName,
246      lg.ListingDate
247  FROM
248      Property p
249  JOIN
250      Owner o ON p.OwnerID = o.OwnerID
251  JOIN
252      Location l ON p.LocationID = l.LocationID
253  JOIN
254      Listing lg ON p.PropertyID = lg.PropertyID
255  JOIN
```

PropertyID	Address	Price	OwnerName	City	State	AgentName	ListingDate
1	123 Elm St	250000.00	John Doe	Springfield	IL	Alice Thompson	2023-01-01
2	456 Oak St	300000.00	Jane Smith	Chicago	IL	George White	2023-01-02
3	789 Pine St	150000.00	Michael Johnson	Los Angeles	CA	Sophia Harris	2023-01-03
4	101 Maple St	350000.00	Patricia Brown	New York	NY	Henry Clark	2023-01-04
5	202 Birch St	275000.00	ali	Houston	TX	Isabella Lewis	2023-01-05
6	303 Cedar St	325000.00	Linda Martinez	Phoenix	AZ	Lucas Young	2023-01-06
7	404 Walnut St	200000.00	William Garcia	Philadelphia	PA	Mia Allen	2023-01-07
8	505 Ash St	225000.00	Elizabeth Wilson	San Antonio	TX	James Scott	2023-01-08
9	606 Willow St	180000.00	Charles Anderson	San Diego	CA	Olivia King	2023-01-09
10	707 Hickory St	400000.00	Barbara Taylor	Dallas	TX	Benjamin Wright	2023-01-10

JOINS

```
259    -- Retrieve listing information along with agent and property details:
260
261 •  SELECT
262        l.ListingID,
263        a.Name AS AgentName,
264        p.Address,
265        l.ListingDate
266    FROM
267        Listing l
268    JOIN
269        Agent a ON l.AgentID = a.AgentID
270    JOIN
271        Property p ON l.PropertyID = p.PropertyID;
```

ListingID	AgentName	Address	ListingDate
1	Alice Thompson	123 Elm St	2023-01-01
2	George White	456 Oak St	2023-01-02
3	Sophia Harris	789 Pine St	2023-01-03
4	Henry Clark	101 Maple St	2023-01-04
5	Isabella Lewis	202 Birch St	2023-01-05
6	Lucas Young	303 Cedar St	2023-01-06
7	Mia Allen	404 Walnut St	2023-01-07
8	James Scott	505 Ash St	2023-01-08
9	Olivia King	606 Willow St	2023-01-09
10	Benjamin Wright	707 Hickory St	2023-01-10

JOINS

```sql
-- Retrieve all leases including tenant and property details, and include lease payments:

SELECT
    l.LeaseID,
    t.Name AS TenantName,
    p.Address,
    l.LeaseStartDate,
    l.LeaseEndDate,
    pm.PaymentID,
    pm.Amount AS PaymentAmount,
    pm.PaymentDate
FROM
    Lease l
JOIN
    Tenant t ON l.TenantID = t.TenantID
JOIN
    Property p ON l.PropertyID = p.PropertyID
LEFT JOIN
    Payment pm ON l.LeaseID = pm.LeaseID;

```

LeaseID	TenantName	Address	LeaseStartDate	LeaseEndDate	PaymentID	PaymentAmount	PaymentDate
1	Olivia Johnson	123 Elm St	2023-04-01	2024-03-31	1	1000.00	2023-05-01
2	Lucas White	456 Oak St	2023-04-02	2024-04-01	2	1500.00	2023-05-02
3	Sophia Harris	789 Pine St	2023-04-03	2024-04-02	3	2000.00	2023-05-03
4	Henry Clark	101 Maple St	2023-04-04	2024-04-03	4	2500.00	2023-05-04
5	Isabella Lewis	202 Birch St	2023-04-05	2024-04-04	5	3000.00	2023-05-05
6	James Scott	303 Cedar St	2023-04-06	2024-04-05	6	3500.00	2023-05-06
7	Mia Allen	404 Walnut St	2023-04-07	2024-04-06	7	4000.00	2023-05-07
8	George White	505 Ash St	2023-04-08	2024-04-07	8	4500.00	2023-05-08
9	Alice Thompson	606 Willow St	2023-04-09	2024-04-08	9	5000.00	2023-05-09
10	Benjamin Wright	707 Hickory St	2023-04-10	2024-04-09	10	5500.00	2023-05-10

JOINS

```
309     -- Retrieve inspection details along with property information:
310
311 •   SELECT
312         i.InspectionID,
313         p.Address,
314         i.InspectionDate,
315         i.Report
316     FROM
317         Inspection i
318     JOIN
319         Property p ON i.PropertyID = p.PropertyID;
320
```

InspectionID	Address	InspectionDate	Report
1	123 Elm St	2023-03-01	Inspection report for property 1
2	456 Oak St	2023-03-02	Inspection report for property 2
3	789 Pine St	2023-03-03	Inspection report for property 3
4	101 Maple St	2023-03-04	Inspection report for property 4
5	202 Birch St	2023-03-05	Inspection report for property 5
6	303 Cedar St	2023-03-06	Inspection report for property 6
7	404 Walnut St	2023-03-07	Inspection report for property 7
8	505 Ash St	2023-03-08	Inspection report for property 8
9	606 Willow St	2023-03-09	Inspection report for property 9
10	707 Hickory St	2023-03-10	Inspection report for property 10

JOINS

```
321    --  Retrieve all properties, including those without a listing (use RIGHT JOIN to include all properties, even those not listed).
322
323 •  SELECT
324        p.PropertyID,
325        p.Address,
326        l.ListingID,
327        a.Name AS AgentName
328    FROM
329        Property p
330    RIGHT JOIN
331        Listing l ON p.PropertyID = l.PropertyID
332    LEFT JOIN
333        Agent a ON l.AgentID = a.AgentID
334    ORDER BY
335        p.PropertyID;
336
```

	PropertyID	Address	ListingID	AgentName
▶	1	123 Elm St	1	Alice Thompson
	2	456 Oak St	2	George White
	3	789 Pine St	3	Sophia Harris
	4	101 Maple St	4	Henry Clark
	5	202 Birch St	5	Isabella Lewis
	6	303 Cedar St	6	Lucas Young
	7	404 Walnut St	7	Mia Allen
	8	505 Ash St	8	James Scott
	9	606 Willow St	9	Olivia King
	10	707 Hickory St	10	Benjamin Wright

JOINS

```
337     --   Retrieve all tenants and their corresponding lease details, including tenants without a lease.
338
339 •   SELECT
340         t.TenantID,
341         t.Name AS TenantName,
342         l.LeaseID,
343         l.PropertyID,
344         l.LeaseStartDate,
345         l.LeaseEndDate
346     FROM
347         Tenant t
348     LEFT JOIN
349         Lease l ON t.TenantID = l.TenantID
350     ORDER BY
351         t.TenantID;
352
```

TenantID	TenantName	LeaseID	PropertyID	LeaseStartDate	LeaseEndDate
1	Olivia Johnson	1	1	2023-04-01	2024-03-31
2	Lucas White	2	2	2023-04-02	2024-04-01
3	Sophia Harris	3	3	2023-04-03	2024-04-02
4	Henry Clark	4	4	2023-04-04	2024-04-03
5	Isabella Lewis	5	5	2023-04-05	2024-04-04
6	James Scott	6	6	2023-04-06	2024-04-05
7	Mia Allen	7	7	2023-04-07	2024-04-06
8	George White	8	8	2023-04-08	2024-04-07
9	Alice Thompson	9	9	2023-04-09	2024-04-08
10	Benjamin Wright	10	10	2023-04-10	2024-04-09

BENEFITS:

- Data Integrity: Minimizing redundancy and preventing anomalies.
- Efficiency:Improved query performance and database management.
- Scalability: A robust design that can handle additional data and relationships.

FUTURE ENHANCEMENTS:

- Additional Functionalities: Implementing advanced features like stored procedures and triggers.
- Optimization: Continuous performance monitoring and optimization.
- Security: Implementing robust security measures.

CONCLUSION

In this project, we successfully designed and implemented a database management system for a real estate transaction application. Starting with a denormalized table, we ensured data integrity and minimized redundancy through normalization.

Key Accomplishments:

- Database Design: Created a denormalized table combining properties, owners, and agents, and normalized it to 3NF.
- Normalization:
  - 1NF: Ensured atomic values and eliminated repeating groups.
  - 2NF: Removed partial dependencies by decomposing into Property, Owner, and Agent tables.
  - 3NF: Removed transitive dependencies, ensuring non-key attributes fully depended on the primary key.
- SQL Queries: Demonstrated DDL and DML commands for creating, altering, and manipulating tables.
- Joins: Used various joins to retrieve related data from multiple tables.

The final database schema is robust, scalable, and ready to support real-world applications.