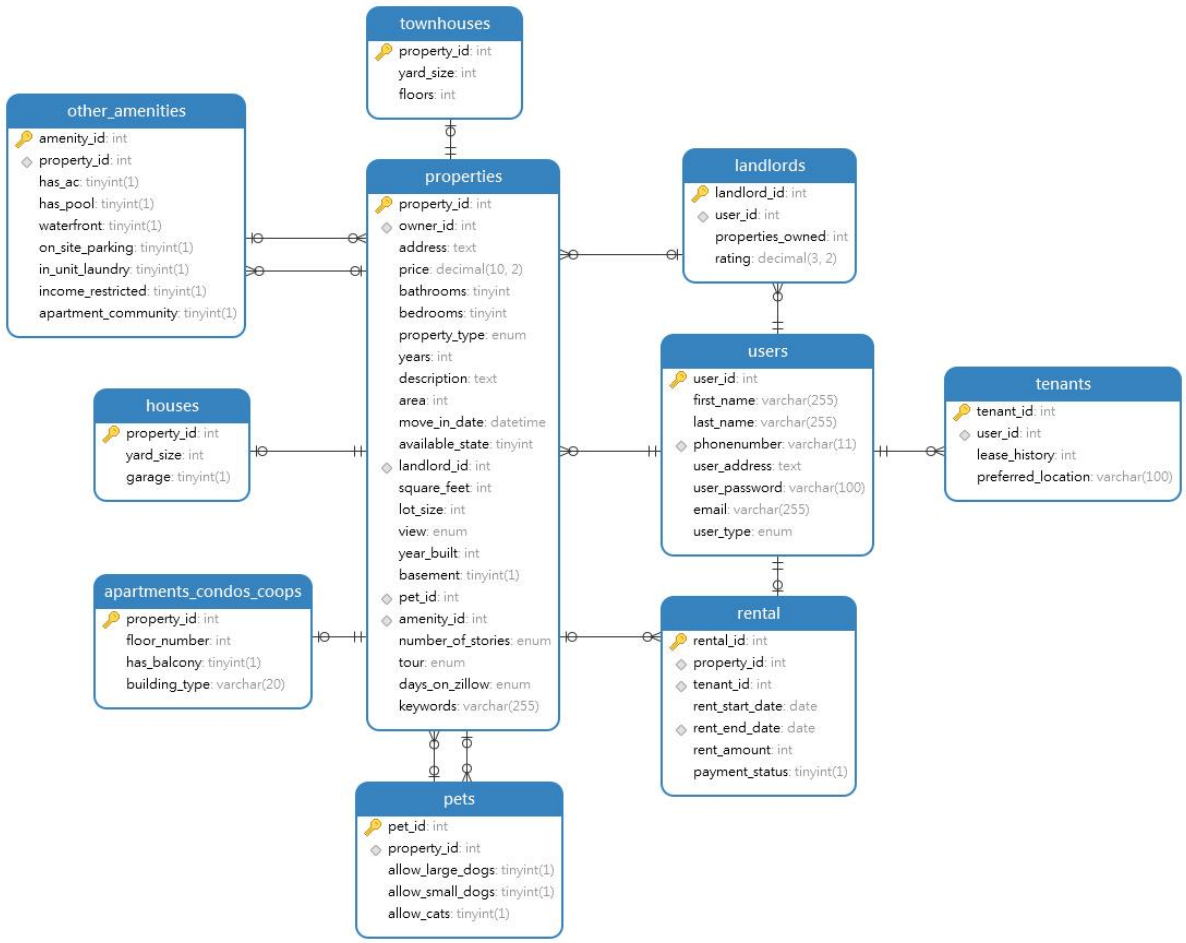
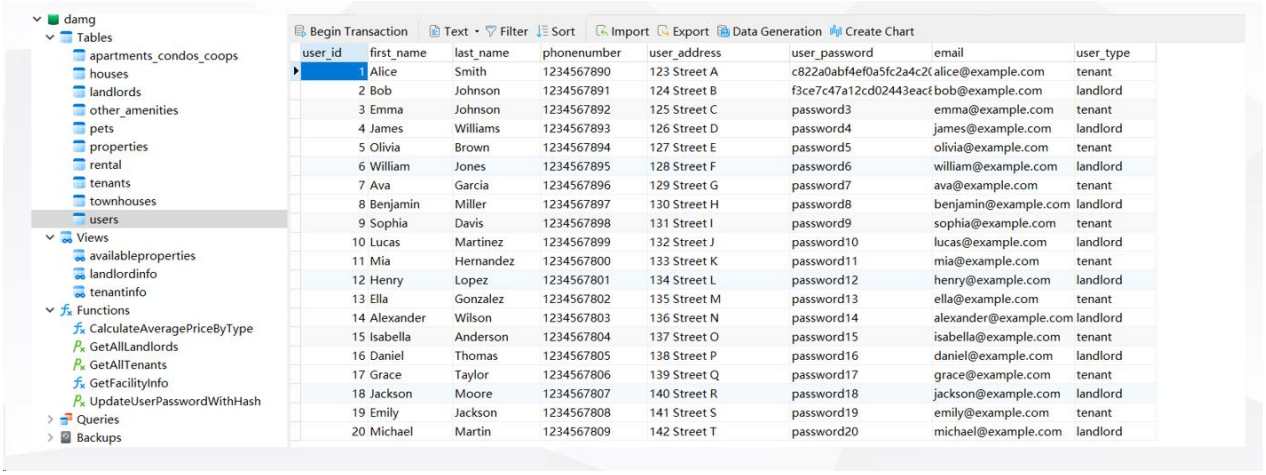
Team 6 House Rental Database Management Systems

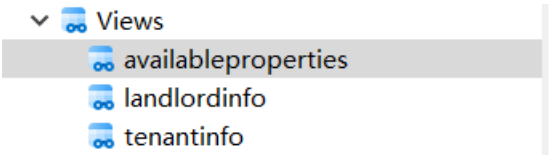
1.Final E-R Diagram



2. Database overview



3. The Views



property_id	owner_id	address	price	bathrooms	bedrooms	property_type	years	description	area	m
3	3	789 Hill Rd	350000.00	3	4	townhouses	8	Spacious townhouse with	2500	2(
5	5	202 Forest Rd	280000.00	2	2	houses	12	Charming house near the	1800	2(
7	7	404 Oak St	310000.00	3	3	townhouses	9	Townhouse in a family-frie	2100	2(
9	9	606 Beach Rd	470000.00	3	4	houses	14	Beachfront house with stu	2700	2(

first_name	last_name	landlord_id	user_id	properties_owned	rating
Bob	Johnson	2	2	3	3.00
William	Jones	4	6	6	4.20
Benjamin	Miller	5	8	2	4.00
Lucas	Martinez	6	10	8	4.60
Henry	Lopez	7	12	3	3.50
Alexander	Wilson	8	14	5	4.10
Daniel	Thomas	9	16	7	4.30
Jackson	Moore	10	18	4	3.90

first_name	last_name	tenant_id	user_id	lease_history	preferred_location
Emma	Johnson	1	3	1	Downtown
Ava	Garcia	2	7	2	Suburbs
Sophia	Davis	3	9	3	Near University
Mia	Hernandez	4	11	4	City Center
Ella	Gonzalez	5	13	1	Riverside
Isabella	Anderson	6	15	2	Uptown
Grace	Taylor	7	17	3	East Side
Emily	Jackson	8	19	1	West End

4. **Data encryption**

```
-- ----------------------------
-- Procedure structure for UpdateUserPasswordWithHash
-- ----------------------------
DROP PROCEDURE IF EXISTS `UpdateUserPasswordWithHash`;
delimiter ;;
CREATE PROCEDURE `UpdateUserPasswordWithHash`(IN p_user_id INT, IN p_new_password VARCHAR(100))
BEGIN DECLARE hashed_password VARCHAR(64);
  SET hashed_password = SHA2(p_new_password, 256);
  UPDATE Users SET user_password = hashed_password WHERE user_id = p_user_id;
  END
;;
delimiter ;
```

We use SHA-256 algorithm to encrypt users' password. And we create a trigger, after insert user' information, trigger will automatically use this store procedure to encrypt.

5. Store procedure

```
-- ----------------------------
-- Procedure structure for GetAllLandlords
-- ----------------------------
DROP PROCEDURE IF EXISTS `GetAllLandlords`;
delimiter ;;
CREATE PROCEDURE `GetAllLandlords`()
BEGIN
 SELECT Users.username, Landlords.* FROM Users JOIN Landlords ON Users.user_id = Landlords.user_id
 WHERE Users.user_type = 'landlord';

 END
;;
delimiter ;
```

```
-- ----------------------------
-- Procedure structure for UpdateUserPasswordWithHash
-- ----------------------------
DROP PROCEDURE IF EXISTS `UpdateUserPasswordWithHash`;
delimiter ;;
CREATE PROCEDURE `UpdateUserPasswordWithHash`(IN p_user_id INT, IN p_new_password VARCHAR(100))
BEGIN DECLARE hashed_password VARCHAR(64);
  SET hashed_password = SHA2(p_new_password, 256);
  UPDATE Users SET user_password = hashed_password WHERE user_id = p_user_id;
  END
;;
delimiter ;
```

```
-- ----------------------------
-- Procedure structure for GetAllTenants
-- ----------------------------
DROP PROCEDURE IF EXISTS `GetAllTenants`;
delimiter ;;
CREATE PROCEDURE `GetAllTenants`()
BEGIN
  SELECT Users.username, Tenants.* FROM Users JOIN Tenants ON Users.user_id = Tenants.user_id
  WHERE Users.user_type = 'tenant';
  END
;;
delimiter ;
```

We create 3 store procedures to help us manage this database better.

6. Triggers

```
-- ----------------------------
-- Triggers structure for table Rental
-- ----------------------------
DROP TRIGGER IF EXISTS `UpdatePropertyAvailability`;
delimiter ;;
CREATE TRIGGER `UpdatePropertyAvailability` AFTER INSERT ON `Rental` FOR EACH ROW BEGIN
UPDATE Properties SET available_state = 0 WHERE property_id = NEW.property_id;
END
;;
delimiter ;
```

We also create another trigger to manage our database.

7. Table-Level Check Constraint

```
CONSTRAINT `CheckPropertyPrice` CHECK ((`price` >= 0))
```

```
CONSTRAINT `CheckLeaseDates` CHECK ((`rent_start_date` < `rent_end_date`)),
CONSTRAINT `CheckRentAmount` CHECK ((`rent_amount` > 0))
```

We create some check constraints to help us manage this database better.

8. Non-Clustered Index

Name	Fields	Index Type	Index method	Comment
userid_UNIQUE	`rental_id` ASC	UNIQUE	BTREE	
userphone_UNIQUE	`tenant_id` ASC	UNIQUE	BTREE	
houseid_UNIQUE	`rent_end_date` ASC	UNIQUE	BTREE	
property_id	`property_id` ASC	NORMAL	BTREE	

We create many non-clustered index to improve the speed of database retrieval