|  |
| --- |
| BUAN6320.504 Group Project Step #2 |
| Requirements Definition Document |
| Group 12 – Apartment Management System |

|  |
| --- |
| M. Dash, T. Raman, V. Sonje, T. Takatani  12-13-2023 |

Table of Contents

[Overview 2](#_Toc149978077)

[Assumptions and Special Considerations 2](#_Toc149978078)

[Business Rules 2](#_Toc149978079)

[Entity and Attribute Descriptions 3](#_Toc149978080)

[APARTMENT 3](#_Toc149978081)

[STAFF 3](#_Toc149978082)

[TENANT 3](#_Toc149978083)

[LEASE 4](#_Toc149978084)

[REQUEST 4](#_Toc149978085)

[PAYMENT 5](#_Toc149978086)

[Relationship and Cardinality Descriptions 5](#_Toc149978087)

[APARTMENT – LEASE 5](#_Toc149978088)

[TENANT – LEASE 5](#_Toc149978089)

[LEASE – PAYMENT 6](#_Toc149978090)

[APARTMENT – REQUEST 6](#_Toc149978091)

[STAFF – REQUEST 6](#_Toc149978092)

[Entity Relationship Diagrams (ERD) 7](#_Toc149978093)

# Overview

Managing an apartment complex involves addressing a variety of tasks, including dealing with maintenance requests, overseeing leases, and tracking monthly payments. Our project’s goal is to create a practical database system designed specifically for small apartment complexes. We aim to keep the system as simple as possible to empower property managers, tenants, and staff to efficiently handle their responsibilities, in an organized and error-free manner.

# Assumptions and Special Considerations

This project simplifies apartment management with these main assumptions:

1. All apartment units are within the same building with the same street address. Therefore, we do not require an entity that contains building information.
2. All maintenance requests are paid for by the apartment complex, not the tenant. Therefore, we do not need to make a relationship between tenant payments and maintenance requests.
3. There can be only one tenant connected to a lease. This person represents the primary lease holder. A future expansion for this system would be to include secondary tenants.
4. Multiple staff members may help address requests, however, only one staff member manages any specific request; they are the point of contact.
5. No lease can be terminated before the specified end date of the lease.
6. A tenant must sign another lease to renew any leases after the specified lease end date.
7. The system includes past tenant information, uniquely identified by the details within the history of leases in connection with tenant and apartment information.
8. Only one payment per day is processed for any given lease.

Special considerations to this project:

1. The lease weak entity was given a surrogate key on implementation to increase query performance.
2. ER-Assistant has hard coded widths for the entity blocks. As such, some attributes titles may be cutoff.
3. Within the Entity Relationship Diagram (ERD), foreign keys are identified with the “Attribute (FK)” notation.

# Business Rules

1. An APARTMENT may have zero, one, or multiple LEASES.
2. Each LEASE must be associated with one, and only one, APARTMENT.
3. A TENANT must sign one or multiple LEASES.
4. Each LEASE must be signed by one, and only one, TENANT.
5. A LEASE requires one or multiple PAYMENTS.
6. Each PAYMENT must be linked to one and only one LEASE.
7. An APARTMENT may raise zero, one, or multiple maintenance REQUESTS.
8. Each maintenance REQUEST must be raised from one, and only one, APARTMENT.
9. A STAFF member may manage zero, one, or multiple maintenance REQUESTS.
10. Each maintenance REQUEST must be managed by one, and only one, STAFF member.

# Entity and Attribute Descriptions

### APARTMENT

**Entity Name**: APARTMENT

**Entity Description**: Represents individual housing units within the same apartment complex.

**Main Attributes** of APARTMENT:

* Apt\_ID (Primary Key): Unique identifier for the apartment. This is a natural primary key based on the unit number.
* Apt\_Sqft: Square footage or area of the apartment unit.
* Apt\_BedNum: Total number of bedrooms in the apartment unit.
* Apt\_BathNum: Total number of bathrooms in the apartment unit.
* Apt\_GarageNum: Assigned garage number for the apartment unit.

### STAFF

**Entity Name**: STAFF

**Entity Description**: Represents individuals employed within the apartment complex to manage various aspects of its operations.

**Main Attributes** of STAFF:

* Stf\_ID (Primary Key): Unique identifier for the staff. This is a surrogate primary key.
* Stf\_FName: First name of the staff member.
* Stf\_LName: Last name of the staff member.
* Stf\_Role: Specific role or position held by the staff member.
* Stf\_Phone: Telephone number of the staff member.
* Stf\_Email: E-mail address of the staff member.
* Stf\_HireDate: Date of hiring of the staff member.

### TENANT

**Entity Name**: TENANT

**Entity Description**: Represents individuals who have leased an apartment unit within the apartment complex.

**Main Attributes** of TENANT:

* Ten\_ID (Primary Key): Unique identifier for the tenant. This is a surrogate primary key.
* Ten\_FName: First name of the tenant.
* Ten\_LName: Last name of the tenant.
* Ten\_DOB: Date of Birth of the tenant.
* Ten\_Phone: Telephone number of the tenant.
* Ten\_Email: E-mail address of the tenant.

### LEASE

**Entity Name**: LEASE

**Entity Description**: Represents a legally binding agreement between a tenant and the apartment complex owner or property management for the occupancy of an apartment unit.

**Main Attributes** of LEASE:

* Les\_ID (Primary Key): Unique identifier for the lease. This is a surrogate primary key.
* Apt\_ID (FK): Unique identifier for the apartment unit.
* Ten\_ID (FK): Unique identifier for the tenant.
* Les\_StartDate: Specific date when the lease agreement officially begins and becomes effective.
* Les\_EndDate: Specific date when the lease agreement is set to expire.
* Les\_CostPerMonth: Amount of money due per month for the right to occupy and use the apartment unit by the tenant.
* Les\_AppFee: One-time fee associated with processing the tenant’s rental application.
* Les\_Deposit: Amount of money complex holds for future damages and costs.

### REQUEST

**Entity Name**: REQUEST

**Entity Description**: Represents a record of requests made by tenants for repairs, maintenance, or service within the apartment complex for a specific apartment unit.

**Main Attributes** of REQUEST:

* Apt\_ID (PK,FK): Unique identifier for the apartment unit.
* Req\_IssueDate (Primary Key): Date on which the request is registered.
* Req\_Type (Primary Key): Type of the request raised.
* Req\_Status: Current condition of progress of the request (Pending, In-progress, Finished).
* Req\_Cost: Cost associated with the service provided for the request.
* Req\_FinishDate: Date on which the request was finished.
* Stf\_ID (FK): Unique identifier for the staff member managing the request.

### PAYMENT

**Entity Name**: PAYMENT

**Entity Description**: Represents financial transactions related to rent payments for a specific unit within the apartment complex.

**Main Attributes** of PAYMENT:

* Les\_ID (PK, FK): Unique identifier for the lease for which a payment is made.
* Pay\_Date (Primary Key): Date on which transaction was processed.
* Pay\_Amount: Total amount of money received in the transaction.
* Pay\_Fine: Fine imposed on the tenant (if any) for late rent payment.
* Pay\_Method: Mechanism of monetary transaction (Bank transfer, cash, EFT etc.).
* Pay\_Status: End result after payment is processed (Accepted, Declined, Failed).

# Relationship and Cardinality Descriptions

### APARTMENT – LEASE

**Relationship**: “Has” between APARTMENT and LEASE

**Cardinality**: 1:M between APARTMENT and LEASE

**Business Rule**: An APARTMENT may have zero, one, or multiple LEASES; Each LEASE must be associated with one, and only one, APARTMENT.

### TENANT – LEASE

**Relationship**: “Signs” between TENANT and LEASE

**Cardinality**: 1:M between TENANT and LEASE

**Business Rule**: A TENANT must sign one or multiple LEASES; Each LEASE must be signed by one, and only one, TENANT.

### LEASE – PAYMENT

**Relationship**: “Requires” between LEASE and PAYMENT

**Cardinality**: 1:M between LEASE and PAYMENT

**Business Rule**: A LEASE requires one or multiple PAYMENTS; Each PAYMENT must be linked to one and only one LEASE.

### APARTMENT – REQUEST

**Relationship**: “Raises” between APARTMENT and REQUEST

**Cardinality**: 1:M between APARTMENT and REQUEST

**Business Rule**: An APARTMENT may raise zero, one, or multiple maintenance REQUESTS; Each maintenance REQUEST must be raised from one, and only one, APARTMENT.

### STAFF – REQUEST

**Relationship**: “Manages” between STAFF and REQUEST

**Cardinality**: 1:M between STAFF and REQUEST

**Business Rule**: A STAFF member may manage to zero, one, or multiple maintenance REQUESTS; Each maintenance REQUEST must be managed by one, and only one, STAFF member.

# Entity Relationship Diagrams (ERD)

Original Concept

A diagram of a computer code

Description automatically generated with medium confidence

Implemented Concept

A diagram of a company

Description automatically generated

# DDL Script

/\* Prompt dropping Triggers \*/

set search\_path='public';

DROP TRIGGER IF EXISTS set\_apt\_defaults ON apartment;

DROP TRIGGER IF EXISTS set\_ten\_defaults ON tenant;

DROP TRIGGER IF EXISTS set\_stf\_defaults ON staff;

DROP TRIGGER IF EXISTS set\_les\_defaults ON lease;

DROP TRIGGER IF EXISTS set\_req\_defaults ON request;

DROP TRIGGER IF EXISTS set\_pay\_defaults ON payment;

DROP TRIGGER IF EXISTS set\_apt\_updates ON apartment;

DROP TRIGGER IF EXISTS set\_ten\_updates ON tenant;

DROP TRIGGER IF EXISTS set\_stf\_updates ON staff;

DROP TRIGGER IF EXISTS set\_les\_updates ON lease;

DROP TRIGGER IF EXISTS set\_req\_updates ON request;

DROP TRIGGER IF EXISTS set\_pay\_updates ON payment;

/\* Prompt dropping Functions \*/

DROP FUNCTION IF EXISTS insert\_trigger();

DROP FUNCTION IF EXISTS update\_trigger();

DROP FUNCTION IF EXISTS ten\_insert\_trigger();

DROP FUNCTION IF EXISTS stf\_insert\_trigger();

DROP FUNCTION IF EXISTS les\_insert\_trigger();

/\* Prompt dropping Sequences \*/

DROP SEQUENCE IF EXISTS ten\_id\_seq;

DROP SEQUENCE IF EXISTS stf\_id\_seq;

DROP SEQUENCE IF EXISTS les\_id\_seq;

/\* Prompt dropping Views \*/

/\* Prompt dropping Indices \*/

DROP INDEX IF EXISTS IDX\_apt\_garageid;

DROP INDEX IF EXISTS IDX\_ten\_phone;

DROP INDEX IF EXISTS IDX\_ten\_email;

DROP INDEX IF EXISTS IDX\_stf\_phone;

DROP INDEX IF EXISTS IDX\_stf\_email;

DROP INDEX IF EXISTS IDX\_les\_aptid\_FK;

DROP INDEX IF EXISTS IDX\_les\_tenid\_FK;

DROP INDEX IF EXISTS IDX\_req\_stfid\_FK;

DROP INDEX IF EXISTS IDX\_req\_status;

DROP INDEX IF EXISTS IDX\_pay\_status;

/\* Prompt dropping Table alters \*/

ALTER TABLE lease DROP CONSTRAINT IF EXISTS check\_dates;

ALTER TABLE lease DROP CONSTRAINT IF EXISTS check\_deposit;

ALTER TABLE lease DROP CONSTRAINT IF EXISTS check\_cost;

ALTER TABLE request DROP CONSTRAINT IF EXISTS check\_status;

ALTER TABLE payment DROP CONSTRAINT IF EXISTS check\_method;

ALTER TABLE payment DROP CONSTRAINT IF EXISTS check\_status;

/\* Prompt dropping Tables \*/

DROP TABLE IF EXISTS payment CASCADE;

DROP TABLE IF EXISTS request CASCADE;

DROP TABLE IF EXISTS lease CASCADE;

DROP TABLE IF EXISTS apartment CASCADE;

DROP TABLE IF EXISTS tenant CASCADE;

DROP TABLE IF EXISTS staff CASCADE;

/\* Prompt creating Tables based on entities \*/

/\*

TABLE apartment IS 'Information for individual housing units within the same apartment complex.'

COLUMN apt\_id IS 'Unique identifier for the apartment.

This is a natural primary key based on the unit number.'

COLUMN apt\_sqft IS 'Square footage or area of the apartment unit.'

COLUMN apt\_bednum IS 'Total number of bedrooms in the apartment unit.'

COLUMN apt\_bathnum IS 'Total number of bathrooms in the apartment unit.'

COLUMN apt\_garageid IS 'Assigned garage identifier for the apartment unit.'

COLUMN created\_by IS 'Audit column - indicates user who inserted data.'

COLUMN created\_date IS 'Audit column - indicates date of insert.'

COLUMN updated\_by IS 'Audit column - indicates who made last update.'

COLUMN updated\_date IS 'Audit column - date of last update.'

\*/

CREATE TABLE apartment (

apt\_id VARCHAR(5) NOT NULL, --This is a natural key, does not require a sequence

apt\_sqft INTEGER NOT NULL,

apt\_bednum INTEGER NOT NULL,

apt\_bathnum INTEGER NOT NULL,

apt\_garageid VARCHAR(5) NOT NULL,

created\_by VARCHAR(50),

created\_date TIMESTAMP,

updated\_by VARCHAR(50),

updated\_date TIMESTAMP,

CONSTRAINT PK\_apartment PRIMARY KEY (apt\_id)

);

/\*

TABLE tenant IS 'Information for individuals who have leased an apartment unit

within the apartment complex.'

COLUMN ten\_id IS 'Unique identifier for the staff. This is a surrogate primary key.'

COLUMN ten\_fname IS 'First name of the tenant.'

COLUMN ten\_lname IS 'Last name of the tenant.'

COLUMN ten\_dob IS 'Date of Birth of the tenant.'

COLUMN ten\_phone IS 'Telephone number of the tenant.'

COLUMN ten\_email IS 'E-mail address of the tenant.'

COLUMN created\_by IS 'Audit column - indicates user who inserted data.'

COLUMN created\_date IS 'Audit column - indicates date of insert.'

COLUMN updated\_by IS 'Audit column - indicates who made last update.'

COLUMN updated\_date IS 'Audit column - date of last update.'

\*/

CREATE TABLE tenant (

ten\_id INTEGER NOT NULL,

ten\_fname VARCHAR(50) NOT NULL,

ten\_lname VARCHAR(50) NOT NULL,

ten\_dob DATE NOT NULL,

ten\_phone VARCHAR(50) NOT NULL,

ten\_email VARCHAR(50) NOT NULL,

created\_by VARCHAR(50),

created\_date TIMESTAMP,

updated\_by VARCHAR(50),

updated\_date TIMESTAMP,

CONSTRAINT PK\_tenant PRIMARY KEY (ten\_id)

);

/\*

TABLE staff IS 'Information for individuals employed within the apartment complex

to manage various aspects of its operations.'

COLUMN stf\_id IS 'Unique identifier for the staff. This is a surrogate primary key.'

COLUMN stf\_fname IS 'First name of the staff member.'

COLUMN stf\_lname IS 'Last name of the staff member.'

COLUMN stf\_role IS 'Specific role or position held by the staff member.'

COLUMN stf\_phone IS 'Telephone number of the staff member.'

COLUMN stf\_email IS 'E-mail address of the staff member.'

COLUMN str\_hiredate IS 'Date of hiring of the staff member.'

COLUMN created\_by IS 'Audit column - indicates user who inserted data.'

COLUMN created\_date IS 'Audit column - indicates date of insert.'

COLUMN updated\_by IS 'Audit column - indicates who made last update.'

COLUMN updated\_date IS 'Audit column - date of last update.'

\*/

CREATE TABLE staff (

stf\_id INTEGER NOT NULL,

stf\_fname VARCHAR(50) NOT NULL,

stf\_lname VARCHAR(50) NOT NULL,

stf\_role VARCHAR(50) NOT NULL,

stf\_phone VARCHAR(50) NOT NULL,

stf\_email VARCHAR(50) NOT NULL,

stf\_hiredate DATE NOT NULL,

created\_by VARCHAR(50),

created\_date TIMESTAMP,

updated\_by VARCHAR(50),

updated\_date TIMESTAMP,

CONSTRAINT PK\_staff PRIMARY KEY (stf\_id)

);

/\*

TABLE lease IS 'Information on legally binding agreement between a tenant

and the apartment complex owner or property management

for the occupancy of an apartment unit.'

COLUMN les\_id IS 'Unique identifier for the staff. This is a surrogate primary key.'

COLUMN apt\_id IS 'Unique identifier for the apartment unit involved in the lease.

This is a foreign key.'

COLUMN ten\_id IS 'Unique identifier for the tenant involved in the lease.

This is a foreign key.'

COLUMN les\_startdate IS 'Specific date when the lease agreement

officially begins and becomes effective.'

COLUMN les\_enddate IS 'Specific date when the lease agreement is set to expire.'

COLUMN les\_costpermonth IS 'Amount of money due per month for the right to occupy

and use the apartment unit by the tenant.'

COLUMN les\_appfee IS 'One-time fee associated with processing the tenant’s rental application.'

COLUMN les\_deposit IS 'Amount of money complex holds for damages and costs at lease end date.'

COLUMN created\_by IS 'Audit column - indicates user who inserted data.'

COLUMN created\_date IS 'Audit column - indicates date of insert.'

COLUMN updated\_by IS 'Audit column - indicates who made last update.'

COLUMN updated\_date IS 'Audit column - date of last update.'

\*/

CREATE TABLE lease (

les\_id INTEGER NOT NULL,

apt\_id VARCHAR(5) NOT NULL,

ten\_id INTEGER NOT NULL,

les\_startdate DATE NOT NULL,

les\_enddate DATE NOT NULL,

les\_costpermonth INTEGER NOT NULL,

les\_appfee INTEGER NOT NULL,

les\_deposit INTEGER NOT NULL,

created\_by VARCHAR(50),

created\_date TIMESTAMP,

updated\_by VARCHAR(50),

updated\_date TIMESTAMP,

CONSTRAINT PK\_lease PRIMARY KEY (les\_id),

CONSTRAINT FK\_lease\_apt\_id FOREIGN KEY (apt\_id) REFERENCES apartment,

CONSTRAINT FK\_lease\_ten\_id FOREIGN KEY (ten\_id) REFERENCES tenant

);

/\*

TABLE request IS 'A record of requests made by tenants for repairs, maintenance,

or service within the apartment complex for a specific apartment unit.'

COLUMN apt\_id IS 'Unique identifier for the apartment involved in a request.

This is both a foreign key and part of the composite primary key.'

COLUMN req\_issuedate IS 'Date on which the request is registered.

This is part of the composite primary key.'

COLUMN req\_type IS 'The type of the request raised. This is par of the composite primary key.'

COLUMN req\_status IS 'Current condition of progress of the request (Pending, In-progress, Finished).'

COLUMN req\_cost IS 'Cost associated with the service provided for the request.'

COLUMN req\_finishdate IS 'Date on which the request was finished.'

COLUMN stf\_id IS 'Unique identifier for the staff member managing the request.'

COLUMN created\_by IS 'Audit column - indicates user who inserted data.'

COLUMN created\_date IS 'Audit column - indicates date of insert.'

COLUMN updated\_by IS 'Audit column - indicates who made last update.'

COLUMN updated\_date IS 'Audit column - date of last update.'

\*/

CREATE TABLE request (

apt\_id VARCHAR(5) NOT NULL,

req\_issuedate DATE NOT NULL,

req\_type VARCHAR(50) NOT NULL,

req\_status VARCHAR(50) NOT NULL,

req\_cost DECIMAL(5,2),

req\_finishdate DATE,

stf\_id INTEGER,

created\_by VARCHAR(50),

created\_date TIMESTAMP,

updated\_by VARCHAR(50),

updated\_date TIMESTAMP,

CONSTRAINT PK\_request PRIMARY KEY (apt\_id, req\_issuedate, req\_type),

CONSTRAINT FK\_request\_apt\_id FOREIGN KEY (apt\_id) REFERENCES apartment,

CONSTRAINT FK\_request\_stf\_id FOREIGN KEY (stf\_id) REFERENCES staff

);

/\*

TABLE request IS 'A record of requests made by tenants for repairs, maintenance,

or service within the apartment complex for a specific apartment unit.'

COLUMN les\_id IS 'Unique identifier for the lease for which a payment is made.

This is both a foreign key and part of the composite primary key.'

COLUMN pay\_date IS 'Date on which transaction was processed.

This is part of the composite primary key.'

COLUMN pay\_amount IS 'Total amount of money received in the transaction.'

COLUMN pay\_fine IS 'Fine imposed on the tenant (if any) for late rent payment.'

COLUMN pay\_method IS 'Mechanism of monetary transaction

(Bank transfer, cash, electronic fund transfer etc.).'

COLUMN pay\_status IS 'End result after payment is processed (Accepted, Declined, or Failed).'

COLUMN created\_by IS 'Audit column - indicates user who inserted data.'

COLUMN created\_date IS 'Audit column - indicates date of insert.'

COLUMN updated\_by IS 'Audit column - indicates who made last update.'

COLUMN updated\_date IS 'Audit column - date of last update.'

\*/

CREATE TABLE payment (

les\_id INTEGER NOT NULL,

pay\_date DATE NOT NULL,

pay\_amount INTEGER NOT NULL,

pay\_fine INTEGER NOT NULL,

pay\_method VARCHAR(50) NOT NULL,

pay\_status VARCHAR(50) NOT NULL,

created\_by VARCHAR(50),

created\_date TIMESTAMP,

updated\_by VARCHAR(50),

updated\_date TIMESTAMP,

CONSTRAINT PK\_payment PRIMARY KEY (les\_id, pay\_date),

CONSTRAINT FK\_payment\_les\_id FOREIGN KEY (les\_id) REFERENCES lease

);

/\* Prompt creating indices for natural and foreign keys \*/

-- Apartment

CREATE UNIQUE INDEX IDX\_apt\_garageid ON apartment (apt\_garageid);

-- Tenant

CREATE UNIQUE INDEX IDX\_ten\_phone ON tenant (ten\_phone);

CREATE UNIQUE INDEX IDX\_ten\_email ON tenant (ten\_email);

-- Staff

CREATE UNIQUE INDEX IDX\_stf\_phone ON staff (stf\_phone);

CREATE UNIQUE INDEX IDX\_stf\_email ON staff (stf\_email);

-- Lease

CREATE INDEX IDX\_les\_aptid\_FK ON lease (apt\_id);

CREATE INDEX IDX\_les\_tenid\_FK ON lease (ten\_id);

-- Request

CREATE INDEX IDX\_req\_stfid\_FK ON request (stf\_id);

CREATE INDEX IDX\_req\_status ON request (req\_status);

-- Payment

CREATE INDEX IDX\_pay\_status ON payment (pay\_status);

/\* Prompt creating Check Constraints \*/

-- Lease

ALTER TABLE lease ADD CONSTRAINT check\_dates CHECK (les\_enddate > les\_startdate);

ALTER TABLE lease ADD CONSTRAINT check\_deposit CHECK (les\_deposit >= les\_costpermonth);

ALTER TABLE lease ADD CONSTRAINT check\_cost CHECK (les\_costpermonth > 500);

-- Request

ALTER TABLE request ADD CONSTRAINT check\_status

CHECK (req\_status = 'Pending' OR req\_status = 'In-Progress' OR req\_status = 'Finished');

-- Payment

ALTER TABLE payment ADD CONSTRAINT check\_method

CHECK (pay\_method = 'Check' OR pay\_method = 'EFT' OR pay\_method = 'Cash');

ALTER TABLE payment ADD CONSTRAINT check\_status

CHECK (pay\_status = 'Accepted' OR pay\_status = 'Declined' OR pay\_status = 'Failed');

/\* Prompt creating Sequences for surrogate, primary keys \*/

-- For use to populate surrogate keys for tenant.

CREATE SEQUENCE ten\_id\_seq

START WITH 1

INCREMENT BY 1

MINVALUE 1;

-- For use to populate surrogate keys for staff.

CREATE SEQUENCE stf\_id\_seq

START WITH 1

INCREMENT BY 1

MINVALUE 1;

-- For use to populate surrogate keys for lease.

CREATE SEQUENCE les\_id\_seq

START WITH 1

INCREMENT BY 1

MINVALUE 1;

/\* Prompt creating Functions \*/

/\*

Business purpose: The general insert trigger automatically assigns

appropriate values to the created\_by, created\_date, updated\_by, and updated\_date fields.

\*/

CREATE OR REPLACE FUNCTION insert\_trigger()

RETURNS TRIGGER AS $$

BEGIN

-- Set created\_by to the current user if it's null

IF NEW.created\_by IS NULL THEN

NEW.created\_by := current\_user;

END IF;

-- Set created\_date to the current timestamp if it's null

IF NEW.created\_date IS NULL THEN

NEW.created\_date := NOW();

END IF;

-- Set updated\_by to the current user if it's null

IF NEW.updated\_by IS NULL THEN

NEW.updated\_by := current\_user;

END IF;

-- Set updated\_date to the current timestamp if it's null

IF NEW.updated\_date IS NULL THEN

NEW.updated\_date := NOW();

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

/\*

Business purpose: The general update trigger automatically assigns

appropriate values to the updated\_by and updated\_date fields.

\*/

CREATE OR REPLACE FUNCTION update\_trigger()

RETURNS TRIGGER AS $$

BEGIN

NEW.updated\_by := current\_user;

NEW.updated\_date := NOW();

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

/\*

Business purpose: The ten insert trigger automatically assigns

appropriate values to the created\_by, created\_date, updated\_by, and updated\_date fields.

Also automatically assigns a sequential ten\_id to a newly-inserted row in the tenant table.

\*/

CREATE OR REPLACE FUNCTION ten\_insert\_trigger()

RETURNS TRIGGER AS $$

BEGIN

-- Set ten\_id to the next value in the sequence

NEW.ten\_id = NEXTVAL('ten\_id\_seq');

-- Set created\_by to the current user if it's null

IF NEW.created\_by IS NULL THEN

NEW.created\_by := current\_user;

END IF;

-- Set created\_date to the current timestamp if it's null

IF NEW.created\_date IS NULL THEN

NEW.created\_date := NOW();

END IF;

-- Set updated\_by to the current user if it's null

IF NEW.updated\_by IS NULL THEN

NEW.updated\_by := current\_user;

END IF;

-- Set updated\_date to the current timestamp if it's null

IF NEW.updated\_date IS NULL THEN

NEW.updated\_date := NOW();

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

/\*

Business purpose: The stf insert trigger automatically assigns

appropriate values to the created\_by, created\_date, updated\_by, and updated\_date fields.

Also automatically assigns a sequential stf\_id to a newly-inserted row in the staff table.

\*/

CREATE OR REPLACE FUNCTION stf\_insert\_trigger()

RETURNS TRIGGER AS $$

BEGIN

-- Set ten\_id to the next value in the sequence

NEW.stf\_id = NEXTVAL('stf\_id\_seq');

-- Set created\_by to the current user if it's null

IF NEW.created\_by IS NULL THEN

NEW.created\_by := current\_user;

END IF;

-- Set created\_date to the current timestamp if it's null

IF NEW.created\_date IS NULL THEN

NEW.created\_date := NOW();

END IF;

-- Set updated\_by to the current user if it's null

IF NEW.updated\_by IS NULL THEN

NEW.updated\_by := current\_user;

END IF;

-- Set updated\_date to the current timestamp if it's null

IF NEW.updated\_date IS NULL THEN

NEW.updated\_date := NOW();

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

/\*

Business purpose: The les insert trigger automatically assigns

appropriate values to the created\_by, created\_date, updated\_by, and updated\_date fields.

Also automatically assigns a sequential les\_id to a newly-inserted row in the lease table.

\*/

CREATE OR REPLACE FUNCTION les\_insert\_trigger()

RETURNS TRIGGER AS $$

BEGIN

-- Set ten\_id to the next value in the sequence

NEW.les\_id = NEXTVAL('les\_id\_seq');

-- Set created\_by to the current user if it's null

IF NEW.created\_by IS NULL THEN

NEW.created\_by := current\_user;

END IF;

-- Set created\_date to the current timestamp if it's null

IF NEW.created\_date IS NULL THEN

NEW.created\_date := NOW();

END IF;

-- Set updated\_by to the current user if it's null

IF NEW.updated\_by IS NULL THEN

NEW.updated\_by := current\_user;

END IF;

-- Set updated\_date to the current timestamp if it's null

IF NEW.updated\_date IS NULL THEN

NEW.updated\_date := NOW();

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

/\* Prompt creating insert Triggers \*/

CREATE TRIGGER set\_apt\_defaults

BEFORE INSERT ON apartment

FOR EACH ROW

EXECUTE FUNCTION insert\_trigger();

CREATE TRIGGER set\_ten\_defaults

BEFORE INSERT ON tenant

FOR EACH ROW

EXECUTE FUNCTION ten\_insert\_trigger();

CREATE TRIGGER set\_stf\_defaults

BEFORE INSERT ON staff

FOR EACH ROW

EXECUTE FUNCTION stf\_insert\_trigger();

CREATE TRIGGER set\_les\_defaults

BEFORE INSERT ON lease

FOR EACH ROW

EXECUTE FUNCTION les\_insert\_trigger();

CREATE TRIGGER set\_req\_defaults

BEFORE INSERT ON request

FOR EACH ROW

EXECUTE FUNCTION insert\_trigger();

CREATE TRIGGER set\_pay\_defaults

BEFORE INSERT ON payment

FOR EACH ROW

EXECUTE FUNCTION insert\_trigger();

/\* Prompt creating update Triggers \*/

CREATE TRIGGER set\_apt\_updates

BEFORE UPDATE ON apartment

FOR EACH ROW

EXECUTE FUNCTION update\_trigger();

CREATE TRIGGER set\_ten\_updates

BEFORE UPDATE ON tenant

FOR EACH ROW

EXECUTE FUNCTION update\_trigger();

CREATE TRIGGER set\_stf\_updates

BEFORE UPDATE ON staff

FOR EACH ROW

EXECUTE FUNCTION update\_trigger();

CREATE TRIGGER set\_les\_updates

BEFORE UPDATE ON lease

FOR EACH ROW

EXECUTE FUNCTION update\_trigger();

CREATE TRIGGER set\_req\_updates

BEFORE UPDATE ON request

FOR EACH ROW

EXECUTE FUNCTION update\_trigger();

CREATE TRIGGER set\_pay\_updates

BEFORE UPDATE ON payment

FOR EACH ROW

EXECUTE FUNCTION update\_trigger();

-- Check DBMS data dictionary to ensure all objects created successfully

SELECT TABLE\_NAME FROM information\_schema.tables

WHERE table\_schema = 'public';

# Populating Tables

Populate all Tables \*/

-- Apartment Table

INSERT INTO apartment (apt\_id, apt\_sqft, apt\_bednum, apt\_bathnum, apt\_garageid)

VALUES ('101', 2000, 3, 2, '1A');

INSERT INTO apartment (apt\_id, apt\_sqft, apt\_bednum, apt\_bathnum, apt\_garageid)

VALUES ('102', 1000, 2, 2, '1B');

INSERT INTO apartment (apt\_id, apt\_sqft, apt\_bednum, apt\_bathnum, apt\_garageid)

VALUES ('103', 800, 1, 1, '1C');

INSERT INTO apartment (apt\_id, apt\_sqft, apt\_bednum, apt\_bathnum, apt\_garageid)

VALUES ('104', 800, 1, 1, '1D');

INSERT INTO apartment (apt\_id, apt\_sqft, apt\_bednum, apt\_bathnum, apt\_garageid)

VALUES ('105', 1000, 2, 2, '1E');

INSERT INTO apartment (apt\_id, apt\_sqft, apt\_bednum, apt\_bathnum, apt\_garageid)

VALUES ('201', 2000, 3, 2, '2A');

INSERT INTO apartment (apt\_id, apt\_sqft, apt\_bednum, apt\_bathnum, apt\_garageid)

VALUES ('202', 1000, 2, 2, '2B');

INSERT INTO apartment (apt\_id, apt\_sqft, apt\_bednum, apt\_bathnum, apt\_garageid)

VALUES ('203', 800, 1, 1, '2C');

INSERT INTO apartment (apt\_id, apt\_sqft, apt\_bednum, apt\_bathnum, apt\_garageid)

VALUES ('204', 800, 1, 1, '2D');

INSERT INTO apartment (apt\_id, apt\_sqft, apt\_bednum, apt\_bathnum, apt\_garageid)

VALUES ('205', 1000, 2, 2, '2E');

-- Tenant Table

INSERT INTO tenant (ten\_fname, ten\_lname, ten\_dob, ten\_phone, ten\_email)

VALUES ('Gasan', 'Elkhodari', '1982-05-10', '1-555-283-7298', 'gasan.elkhodari@email.com');

INSERT INTO tenant (ten\_fname, ten\_lname, ten\_dob, ten\_phone, ten\_email)

VALUES ('Vipul', 'Sonje', '1997-09-12', '1-505-283-7298', 'vxs230000@utdallas.edu');

INSERT INTO tenant (ten\_fname, ten\_lname, ten\_dob, ten\_phone, ten\_email)

VALUES ('Manmohan', 'Dash', '1995-05-10', '1-355-283-7298', 'man.dash@email.com');

INSERT INTO tenant (ten\_fname, ten\_lname, ten\_dob, ten\_phone, ten\_email)

VALUES ('Thiru', 'Raman', '1982-05-10', '1-578-283-7298', 'thiru.raman@email.com');

INSERT INTO tenant (ten\_fname, ten\_lname, ten\_dob, ten\_phone, ten\_email)

VALUES ('Tait', 'Takatani', '1995-05-10', '1-595-273-7298', 'tait.takatani@email.com');

INSERT INTO tenant (ten\_fname, ten\_lname, ten\_dob, ten\_phone, ten\_email)

VALUES ('Jane', 'Smith', '1992-05-10', '1-555-283-7398', 'jane.smith@email.com');

INSERT INTO tenant (ten\_fname, ten\_lname, ten\_dob, ten\_phone, ten\_email)

VALUES ('Michael', 'Johnson', '1985-08-15', '1-555-754-5832', 'michael.johnson@email.com');

INSERT INTO tenant (ten\_fname, ten\_lname, ten\_dob, ten\_phone, ten\_email)

VALUES ('Emily', 'Brown', '1990-03-20', '1-555-409-1467', 'emily.brown@email.com');

INSERT INTO tenant (ten\_fname, ten\_lname, ten\_dob, ten\_phone, ten\_email)

VALUES ('David', 'Wilson', '1978-01-05', '1-555-126-9205', 'david.wilson@email.com');

INSERT INTO tenant (ten\_fname, ten\_lname, ten\_dob, ten\_phone, ten\_email)

VALUES ('Sarah', 'Davis', '1982-12-08', '1-555-538-3641', 'sarah.davis@email.com');

INSERT INTO tenant (ten\_fname, ten\_lname, ten\_dob, ten\_phone, ten\_email)

VALUES ('William', 'Lee', '1993-06-14', '1-555-672-8754', 'william.lee@email.com');

INSERT INTO tenant (ten\_fname, ten\_lname, ten\_dob, ten\_phone, ten\_email)

VALUES ('Olivia', 'Taylor', '1987-10-02', '1-555-347-6320', 'olivia.taylor@email.com');

INSERT INTO tenant (ten\_fname, ten\_lname, ten\_dob, ten\_phone, ten\_email)

VALUES ('James', 'Harris', '1975-07-25', '1-555-912-4198', 'james.harris@email.com');

INSERT INTO tenant (ten\_fname, ten\_lname, ten\_dob, ten\_phone, ten\_email)

VALUES ('Ava', 'Martin', '1998-04-03', '1-555-645-2576', 'ava.martin@email.com');

INSERT INTO tenant (ten\_fname, ten\_lname, ten\_dob, ten\_phone, ten\_email)

VALUES ('Liam', 'Garcia', '1980-09-12', '1-555-371-6839', 'liam.garcia@email.com');

-- Staff Table

INSERT INTO staff (stf\_fname, stf\_lname, stf\_role, stf\_phone, stf\_email, stf\_hiredate)

VALUES ('Sophia', 'Johnson', 'Manager', '1-555-324-7321', 'sophia.johnson@email.com', '2011-03-17');

INSERT INTO staff (stf\_fname, stf\_lname, stf\_role, stf\_phone, stf\_email, stf\_hiredate)

VALUES ('Vipul', 'Sonje', 'Associate Manager', '1-555-354-7321', 'vipul.sonje@email.com', '2015-03-17');

INSERT INTO staff (stf\_fname, stf\_lname, stf\_role, stf\_phone, stf\_email, stf\_hiredate)

VALUES ('Tait', 'Takatani', 'Associate Manager', '1-535-324-7321', 'takatani.tait@email.com', '2010-03-17');

INSERT INTO staff (stf\_fname, stf\_lname, stf\_role, stf\_phone, stf\_email, stf\_hiredate)

VALUES ('Liam', 'Williams', 'Assistant Manager', '1-555-587-5984', 'liam.williams@email.com', '2019-08-29');

INSERT INTO staff (stf\_fname, stf\_lname, stf\_role, stf\_phone, stf\_email, stf\_hiredate)

VALUES ('Olivia', 'Davis', 'Maintenance Technician', '1-555-149-8274', 'olivia.davis@email.com', '2014-05-06');

INSERT INTO staff (stf\_fname, stf\_lname, stf\_role, stf\_phone, stf\_email, stf\_hiredate)

VALUES ('Noah', 'Smith', 'Receptionist', '1-555-826-4569', 'noah.smith@email.com', '2016-11-14');

INSERT INTO staff (stf\_fname, stf\_lname, stf\_role, stf\_phone, stf\_email, stf\_hiredate)

VALUES ('Ava', 'Anderson', 'Security Guard', '1-555-703-2198', 'ava.anderson@email.com', '2012-02-18');

INSERT INTO staff (stf\_fname, stf\_lname, stf\_role, stf\_phone, stf\_email, stf\_hiredate)

VALUES ('Mason', 'Martinez', 'Housekeeper', '1-555-235-6345', 'mason.martinez@email.com', '2017-09-23');

INSERT INTO staff (stf\_fname, stf\_lname, stf\_role, stf\_phone, stf\_email, stf\_hiredate)

VALUES ('Harper', 'Taylor', 'Groundskeeper', '1-555-482-7821', 'harper.taylor@email.com', '2013-07-04');

INSERT INTO staff (stf\_fname, stf\_lname, stf\_role, stf\_phone, stf\_email, stf\_hiredate)

VALUES ('Elijah', 'Moore', 'Accountant', '1-555-619-3590', 'elijah.moore@email.com', '2018-12-30');

INSERT INTO staff (stf\_fname, stf\_lname, stf\_role, stf\_phone, stf\_email, stf\_hiredate)

VALUES ('Scarlett', 'Jackson', 'Maintenance Technician', '1-555-754-1472', 'scarlett.jackson@email.com', '2015-06-11');

INSERT INTO staff (stf\_fname, stf\_lname, stf\_role, stf\_phone, stf\_email, stf\_hiredate)

VALUES ('James', 'Harris', 'Maintenance Technician', '1-555-398-5038', 'james.harris@email.com', '2011-10-08');

-- Lease Table

INSERT INTO lease (apt\_id, ten\_id, les\_startdate, les\_enddate, les\_costpermonth, les\_appfee, les\_deposit)

VALUES ('101', 1, '2022-05-20', '2023-05-19', 3000, 200, 3000);

INSERT INTO lease (apt\_id, ten\_id, les\_startdate, les\_enddate, les\_costpermonth, les\_appfee, les\_deposit)

VALUES ('102', 2, '2022-09-12', '2023-09-11', 1200, 200, 1200);

INSERT INTO lease (apt\_id, ten\_id, les\_startdate, les\_enddate, les\_costpermonth, les\_appfee, les\_deposit)

VALUES ('103', 3, '2022-07-03', '2023-07-02', 900, 200, 900);

INSERT INTO lease (apt\_id, ten\_id, les\_startdate, les\_enddate, les\_costpermonth, les\_appfee, les\_deposit)

VALUES ('105', 4, '2022-09-30', '2023-09-29', 1100, 200, 1100);

INSERT INTO lease (apt\_id, ten\_id, les\_startdate, les\_enddate, les\_costpermonth, les\_appfee, les\_deposit)

VALUES ('201', 5, '2023-03-15', '2024-03-14', 3200, 200, 3200);

INSERT INTO lease (apt\_id, ten\_id, les\_startdate, les\_enddate, les\_costpermonth, les\_appfee, les\_deposit)

VALUES ('202', 6, '2022-11-30', '2023-11-29', 1200, 200, 1200);

INSERT INTO lease (apt\_id, ten\_id, les\_startdate, les\_enddate, les\_costpermonth, les\_appfee, les\_deposit)

VALUES ('204', 7, '2023-02-07', '2024-02-06', 900, 200, 900);

INSERT INTO lease (apt\_id, ten\_id, les\_startdate, les\_enddate, les\_costpermonth, les\_appfee, les\_deposit)

VALUES ('205', 8, '2023-08-25', '2024-08-24', 1200, 200, 1200);

INSERT INTO lease (apt\_id, ten\_id, les\_startdate, les\_enddate, les\_costpermonth, les\_appfee, les\_deposit)

VALUES ('103', 9, '2023-10-28', '2024-10-27', 1000, 200, 1000);

INSERT INTO lease (apt\_id, ten\_id, les\_startdate, les\_enddate, les\_costpermonth, les\_appfee, les\_deposit)

VALUES ('101', 10, '2023-07-02', '2024-07-01', 3000, 200, 3000);

INSERT INTO lease (apt\_id, ten\_id, les\_startdate, les\_enddate, les\_costpermonth, les\_appfee, les\_deposit)

VALUES ('105', 4, '2023-09-30', '2024-09-29', 1200, 0, 1200);

-- Request Table

INSERT INTO request (apt\_id, req\_issuedate, req\_type, req\_status, req\_cost, req\_finishdate, stf\_id)

VALUES ('101', '2022-06-06', 'Appliance', 'Finished', 50, '2022-06-07', 3);

INSERT INTO request (apt\_id, req\_issuedate, req\_type, req\_status, req\_cost, req\_finishdate, stf\_id)

VALUES ('103', '2022-07-20', 'Electric', 'Finished', 154, '2022-07-25', 3);

INSERT INTO request (apt\_id, req\_issuedate, req\_type, req\_status, req\_cost, req\_finishdate, stf\_id)

VALUES ('201', '2022-11-01', 'HVAC', 'Finished', 584, '2023-08-05', 3);

INSERT INTO request (apt\_id, req\_issuedate, req\_type, req\_status, req\_cost, req\_finishdate, stf\_id)

VALUES ('105', '2022-12-14', 'Plumbing', 'Finished', 347, '2022-12-16', 9);

INSERT INTO request (apt\_id, req\_issuedate, req\_type, req\_status, req\_cost, req\_finishdate, stf\_id)

VALUES ('202', '2023-03-28', 'Electric', 'Finished', 259, '2023-04-01', 10);

INSERT INTO request (apt\_id, req\_issuedate, req\_type, req\_status, req\_cost, req\_finishdate, stf\_id)

VALUES ('102', '2023-05-06', 'Appliance', 'Finished', 634, '2023-05-10', 3);

INSERT INTO request (apt\_id, req\_issuedate, req\_type, req\_status, stf\_id)

VALUES ('101', '2023-10-29', 'Electric', 'In-Progress', 10);

INSERT INTO request (apt\_id, req\_issuedate, req\_type, req\_status, stf\_id)

VALUES ('202', '2023-10-31', 'Appliance', 'In-Progress', 3);

INSERT INTO request (apt\_id, req\_issuedate, req\_type, req\_status)

VALUES ('204', '2023-11-02', 'Electric', 'Pending');

INSERT INTO request (apt\_id, req\_issuedate, req\_type, req\_status)

VALUES ('101', '2023-11-03', 'HVAC', 'Pending');

-- Payment Table

--les\_id=1: VALUES ('101', 1, '5-20-2022', '5-19-2023', 3000, 200, 3000);

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (1, '2022-05-20', 1064, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (1, '2022-06-01', 3000, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (1, '2022-07-01', 3000, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (1, '2022-08-01', 3000, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (1, '2022-09-01', 3000, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (1, '2022-10-01', 3000, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (1, '2022-11-01', 3000, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (1, '2022-12-01', 3000, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (1, '2023-01-01', 3000, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (1, '2023-02-01', 3000, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (1, '2023-03-01', 3000, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (1, '2023-04-01', 3000, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (1, '2023-05-19', 1936, 0, 'EFT', 'Accepted');

--les\_id=2: VALUES ('102', 2, '9-12-2022', '9-11-2023', 1200, 200, 1200);

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (2, '2022-09-12', 720, 0, 'Check', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (2, '2022-10-01', 1200, 0, 'Check', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (2, '2022-11-01', 1200, 0, 'Check', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (2, '2022-12-01', 1200, 0, 'Check', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (2, '2023-01-01', 1200, 0, 'Check', 'Declined');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (2, '2023-01-04', 1212, 12, 'Check', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (2, '2023-02-01', 1200, 0, 'Check', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (2, '2023-03-01', 1200, 0, 'Check', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (2, '2023-04-01', 1200, 0, 'Check', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (2, '2023-05-01', 1200, 0, 'Check', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (2, '2023-06-01', 1200, 0, 'Check', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (2, '2023-07-01', 1200, 0, 'Check', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (2, '2023-08-01', 1200, 0, 'Check', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (2, '2023-09-11', 480, 0, 'Check', 'Accepted');

--les\_id=3: VALUES ('103', 3, '7-3-2022', '7-2-2023', 900, 200, 900);

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (3, '2022-07-03', 843, 0, 'EFT', 'Declined');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (3, '2022-07-04', 843, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (3, '2022-08-01', 900, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (3, '2022-09-01', 900, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (3, '2022-10-01', 900, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (3, '2022-11-01', 900, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (3, '2022-12-01', 900, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (3, '2023-01-01', 900, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (3, '2023-02-01', 900, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (3, '2023-03-01', 900, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (3, '2023-04-01', 900, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (3, '2023-05-01', 900, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (3, '2023-06-01', 900, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (3, '2023-07-01', 57, 0, 'EFT', 'Accepted');

--les\_id=4: VALUES ('105', 4, '9-30-2022', '9-29-2023', 1100, 200, 1100);

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (4, '2022-09-30', 37, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (4, '2022-10-01', 1100, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (4, '2022-11-01', 1100, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (4, '2022-12-01', 1100, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (4, '2023-01-01', 1100, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (4, '2023-02-01', 1100, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (4, '2023-03-01', 1100, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (4, '2023-04-01', 1100, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (4, '2023-05-01', 1100, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (4, '2023-06-01', 1100, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (4, '2023-07-01', 1100, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (4, '2023-08-01', 1100, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (4, '2023-09-01', 1063, 0, 'EFT', 'Accepted');

--les\_id=5: VALUES ('201', 5, '3-15-2023', '3-14-2024', 3200, 200, 3200);

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (5, '2023-03-15', 1756, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (5, '2023-04-01', 3200, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (5, '2023-05-01', 3200, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (5, '2023-06-01', 3200, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (5, '2023-07-01', 3200, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (5, '2023-08-01', 3200, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (5, '2023-09-01', 3200, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (5, '2023-10-01', 3200, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (5, '2023-11-01', 3200, 0, 'EFT', 'Accepted');

--les\_id=6: VALUES ('202', 6, '11-30-2022', '11-29-2023', 1200, 200, 1200);

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (6, '2022-11-30', 40, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (6, '2022-12-01', 1200, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (6, '2023-01-01', 1200, 0, 'EFT', 'Declined');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (6, '2023-01-05', 1212, 12, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (6, '2023-02-01', 1200, 0, 'EFT', 'Declined');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (6, '2023-02-06', 1212, 12, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (6, '2023-03-04', 1212, 12, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (6, '2023-04-01', 1200, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (6, '2023-05-01', 1200, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (6, '2023-06-01', 1200, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (6, '2023-07-01', 1200, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (6, '2023-08-01', 1200, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (6, '2023-09-01', 1200, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (6, '2023-10-01', 1200, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (6, '2023-11-01', 1160, 0, 'EFT', 'Accepted');

--les\_id=7: VALUES ('204', 7, '2-7-2023', '2-6-2024', 900, 200, 900);

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (7, '2023-02-07', 709, 0, 'Check', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (7, '2023-03-01', 900, 0, 'Check', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (7, '2023-04-10', 909, 9, 'Check', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (7, '2023-05-01', 900, 0, 'Check', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (7, '2023-06-01', 900, 0, 'Check', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (7, '2023-07-01', 900, 0, 'Check', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (7, '2023-08-01', 900, 0, 'Check', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (7, '2023-09-01', 900, 0, 'Check', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (7, '2023-10-01', 900, 0, 'Check', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (7, '2023-11-06', 909, 9, 'Check', 'Accepted');

--les\_id=8: VALUES ('205', 8, '2023-08-25', '2024-08-24', 1200, 200, 1200);

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (8, '2023-08-25', 271, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (8, '2023-09-01', 1200, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (8, '2023-10-01', 1200, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (8, '2023-11-01', 1200, 0, 'EFT', 'Accepted');

--les\_id=9: VALUES ('103', 9, '2023-10-28', '2024-10-27', 1000, 200, 1000);

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (9, '2023-10-28', 129, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (9, '2023-11-01', 1000, 0, 'EFT', 'Accepted');

--les\_id=10: VALUES ('101', 10, '2023-07-02', '2024-07-01', 3000, 200, 3000);

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (10, '2023-07-02', 2902, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (10, '2023-08-01', 3000, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (10, '2023-09-01', 3000, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (10, '2023-10-01', 3000, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (10, '2023-11-01', 3000, 0, 'EFT', 'Accepted');

--les\_id=11: VALUES ('105', 4, '2023-09-30', '2024-09-29', 1200, 0, 1200);

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (11, '2023-09-30', 40, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (11, '2023-10-01', 1200, 0, 'EFT', 'Accepted');

INSERT INTO payment (les\_id, pay\_date, pay\_amount, pay\_fine, pay\_method, pay\_status)

VALUES (11, '2023-11-01', 1200, 0, 'EFT', 'Accepted');

BASIC QUERIES

1. Select all columns and all rows from one table.

SELECT \* FROM tenant;

----------------------------------------------------------------------------------

1. Select five columns and all rows from one table.

SELECT apt\_id

,apt\_sqft

,apt\_bednum

,apt\_bathnum

,apt\_garageid

FROM apartment;

----------------------------------------------------------------------------------

1. Select all columns from all rows from one view.

CREATE OR REPLACE VIEW

staff\_info AS

SELECT \* FROM staff;

SELECT \* FROM staff\_info;

----------------------------------------------------------------------------------

1. Using a join on 2 tables, select all columns and all rows from the tables without the use of a Cartesian product.

SELECT \* FROM staff s

INNER JOIN request r

ON s.Stf\_ID = r.Stf\_ID

----------------------------------------------------------------------------------

1. Select and order data retrieved from one table.

SELECT \* FROM REQUEST

ORDER BY Req\_IssueDate desc

----------------------------------------------------------------------------------

1. Using a join on 3 tables, select 5 columns from the 3 tables. Use syntax that would limit the output to 10 rows.

SELECT a.Apt\_ID

,a.Apt\_Sqft

,t.Ten\_ID

,t.Ten\_FName

,l.les\_startDate

FROM apartment a

INNER JOIN lease l

ON a.apt\_id = l.apt\_id

INNER JOIN tenant t

ON t.ten\_id = l.ten\_id

LIMIT 10;

----------------------------------------------------------------------------------

1. Select distinct rows using joins on 3 tables.

SELECT DISTINCT(s.Stf\_ID)

,a.apt\_id

,r.req\_issuedate

,r.req\_type

,r.req\_status

,r.req\_finishdate

FROM apartment a

INNER JOIN request r

ON a.apt\_id = r.apt\_id

INNER JOIN staff s

ON s.Stf\_ID = r.Stf\_ID

----------------------------------------------------------------------------------

1. Use GROUP BY and HAVING in a select statement using one or more tables.

----Find Apartments that have raised multiple maintenance requests

SELECT apartment.apt\_id

,COUNT(request.req\_status) AS pending\_requests

FROM apartment

LEFT JOIN request

ON apartment.apt\_id = request.apt\_id

GROUP BY apartment.apt\_id

HAVING COUNT(request.req\_status) >1;

----------------------------------------------------------------------------------

1. Use IN clause to select data from one or more tables.

SELECT \* FROM PAYMENT p

where p.les\_id IN

(SELECT l.les\_id FROM LEASE l

where l.les\_StartDate >= '2023-01-01'

AND l.les\_StartDate <= '2023-10-31')

AND p.Pay\_Method = 'Check'

----------------------------------------------------------------------------------

1. Select length of one column from one table (use LENGTH function)

SELECT S.Stf\_ID

,S.Stf\_FName

,LENGTH(S.Stf\_FName)

FROM STAFF S

----------------------------------------------------------------------------------

1. Delete one record from one table. Use select statements to demonstrate the table contents before and after the DELETE statement. Make sure you use ROLLBACK afterwards so that the data will not be physically removed.

SELECT \* FROM PAYMENT P;

BEGIN;

DELETE FROM PAYMENT P

where P.les\_ID = 1

AND P.pay\_date = '2022-07-01';

SELECT \* FROM PAYMENT P

ROLLBACK;

END;

SELECT \* FROM PAYMENT P

----------------------------------------------------------------------------------

1. Update one record from one table. Use select statements to demonstrate the table contents before and after the UPDATE statement. Make sure you use ROLLBACK afterwards so that the data will not be physically removed.

SELECT \* FROM STAFF S order by S.stf\_id;

BEGIN;

UPDATE STAFF SET stf\_role = 'Assistant Manager'

where stf\_id = 2;

SELECT \* FROM STAFF S order by S.stf\_id;

ROLLBACK;

END;

SELECT \* FROM STAFF S order by S.stf\_id;

# Advance Queries

1. Calculate the total payments made by each tenant for the year 2023, including fines.

**Business Implication:** This query helps in understanding the financial contributions of

each tenant for the year 2023, including lease payments and fines.

SELECT

tenant.ten\_id,

ten\_fname,

ten\_lname,

SUM(pay\_amount + pay\_fine) AS total\_payments\_2023

FROM tenant

LEFT JOIN lease ON tenant.ten\_id = lease.ten\_id

LEFT JOIN payment ON lease.les\_id = payment.les\_id

WHERE EXTRACT(YEAR FROM pay\_date) = 2023

GROUP BY tenant.ten\_id

ORDER BY total\_payments\_2023 DESC;

1. Find the staff member who has completed the most requests, including In-Progress and completed requests.

**Business Implication**: This query helps identify the most productive staff member in handling service requests, which can be used for performance evaluation.

SELECT

staff.stf\_id,

stf\_fname,

stf\_lname,

COUNT(request.req\_issuedate) AS total\_requests

FROM staff

LEFT JOIN request ON staff.stf\_id = request.stf\_id

WHERE (request.req\_status ILIKE ('Finished')

OR

request.req\_status ILIKE ('In-Progress'))

GROUP BY staff.stf\_id

ORDER BY total\_requests DESC

LIMIT 1;