Viventra Property Management System

Presented by: Zeel Modi

GitHub: [https://github.com/zeel2509]

LinkedIn: [https://www.linkedin.com/in/zeel2509/]

Table of Contents

1. Introduction

Viventra is a modern digital solution designed to address the day-to-day challenges of residential property management. Built with a deep understanding of landlord and tenant pain points, it transforms a traditionally fragmented and paper-based process into a centralized, intelligent, and user-friendly experience. The system supports core tasks such as lease creation, rent collection, payment tracking, maintenance requests, and communication between stakeholders—all within a single platform.

1.1 Industry Background:

The residential real estate industry plays a foundational role in housing global populations, especially in urban and semi-urban regions. However, property management within this sector has historically lagged behind in digital adoption. Most small to mid-sized landlords and property managers rely on spreadsheets, physical documents, and manual tracking. This reliance often results in inefficiencies, delays, and errors.

With technological advancements, there is an increasing demand for smart property management systems that not only automate routine tasks but also ensure transparency, scalability, and real-time reporting. Viventra addresses this digital gap by offering a robust, scalable, and intelligent property management platform.

1.2 Business Problem:

Property managers and landlords often struggle with:

1. Managing a growing portfolio of properties and tenants using outdated systems

2. Handling maintenance requests and repairs in a timely manner

3. Tracking and reconciling rent payments manually

4. Communicating clearly and promptly with tenants

5. Generating reports and maintaining accurate records for tax and compliance purposes

These challenges are exacerbated when managing multiple tenants, properties, or staff without a centralized system. Errors in rent tracking, missed maintenance, and lack of visibility into records can lead to tenant dissatisfaction, legal risks, and financial losses. Viventra solves this by offering automation, real-time tracking, and structured data storage.

2. Mission and Vision

2.1 Mission Statement:

"A smart digital platform simplifying lease, payment, and maintenance management for tenants and landlords."

Viventra aims to bridge the gap between landlords and tenants by offering a tech-enabled, intuitive interface that simplifies common rental processes. The mission focuses on eliminating human error, reducing paperwork, and improving efficiency.

2.2 Vision Statement:

"To become a leading digital property management solution that enhances transparency, improves communication, and reduces the administrative burden in residential rentals."

Viventra envisions an industry where digital property management is the norm. It strives to create an ecosystem where tenants and landlords interact seamlessly, processes are automated, and data is leveraged to improve decision-making and satisfaction.

3. System Overview

Viventra is a relational database-driven platform composed of six core tables representing the primary elements of residential rental operations: Owners, Properties, Tenants, Lease Agreements, Payments, and Maintenance Requests. It facilitates the complete rental lifecycle from listing to leasing, payment collection, issue tracking, and communication. The backend is structured to maintain referential integrity through carefully designed primary and foreign key relationships, enabling secure and scalable property data management.

4. Final Field List Table

Preliminary Table List	Final Table List
Owners	Owners
Properties	Property
Tenants	Tenants
Lease_Agreements	Lease_Agreements
Payments	Payments
Maintenance_Request	Maintenance_Requests
Deposits	
Amenities	

4.1 Owners:

The "Owners" table captures details of individuals or organizations that own rental properties. It is critical for linking each property to its rightful owner.

Preliminary Fields	Final Fields
owner_id *(PK)*	owner_id *(PK)*
first_name	first_name
last_name	last_name
gender	contact_number
contact_number	email
email	
address	
national_id	
account_status	
registration_date	

Fields: owner_id (PK), first_name, last_name, contact_number, email

Use: Supports lease negotiation, payment distribution, and communication

4.2 Property:

This table documents all properties managed under the system. It records comprehensive property details and is central to operations.

Preliminary Fields	Final Fields
property_id *(PK)*	**property_id *(PK)***
owner_id *(FK)*	**address**
address	**city**
city	**state**
state	**zip_code**
zip_code	**property_type**
property_type	**num_bedrooms**
num_bedrooms	**num_bathrooms**
num_bathrooms	**availability_status**
num_bathrooms	**owner_id *(FK)***
availability_status	
rent_amount	
floor_number	
furnished_status	
listed_date	
last_renovated	
amenity_ids	

Fields: property_id (PK), address, city, state, zip_code, property_type, bedrooms, bathrooms, availability_status, owner_id (FK)

Use: Links with leases, maintenance, and ownership records

4.3 Tenants:

Houses all tenant-related data. This table supports lease creation and maintenance request tracking.

Preliminary Fields	Final Fields
tenant_id *(PK)*	**tenant_id *(PK)***
first_name	**first_name**
last_name	**last_name**
gender	**phone**
phone	**email**
email	**date_of_birth**
date_of_birth	**emergency_contact**
nationality	
ID_proof	
permanent_address	
current_address	
emergency_contact	
occupation	
registration_date	

Fields: tenant_id (PK), first_name, last_name, phone_number, email, date_of_birth, emergency_contact_number

Use: Manages tenant records, communication, and payment validation

4.4 Lease_Agreements:

Defines rental agreements between tenants and properties. It records lease terms and is linked to payments.

Preliminary Fields	Final Fields
lease_id *(PK)*	**lease_id *(PK)***
property_id *(FK)*	**property_id *(FK)***
tenant_id *(FK)*	**tenant_id *(FK)***
owner_id *(FK)*	**start_date**
lease_number	**end_date**
start_date	**rent_amount**
end_date	**lease_status**
rent_amount	
payment_frequency	
lease_status	
agreement_file	
termination_date	
comments	

Fields: lease_id (PK), tenant_id (FK), property_id (FK), start_date, end_date, rent_amount, lease_status

Use: Tracks contract history, lease duration, and status

4.5 Payments:

Logs all rent and deposit transactions associated with each lease.

Preliminary Fields	Final Fields
payment_id *(PK)*	payment_id *(PK)*
lease_id *(FK)*	lease_id *(FK)*
payment_date	payment_date
amount	amount
payment_method	payment_method
payment_status	payment_status
transaction_reference	
receipt_file	
late_fee	

Fields: payment_id (PK), lease_id (FK), payment_date, amount, method, status

Use: Generates billing reports, tracks overdue/missed payments

4.6 Maintenance_Requests:

Captures tenant-submitted service or repair needs for properties.

Preliminary Fields	Final Fields
request_id *(PK)*	request_id *(PK)*
property_id *(FK)*	property_id *(FK)*
tenant_id *(FK)*	tenant_id *(FK)*
request_date	request_date
issue_type	issue_description
issue_description	status
assigned_technician	resolution_date
status	
priority_level	
resolution_date	

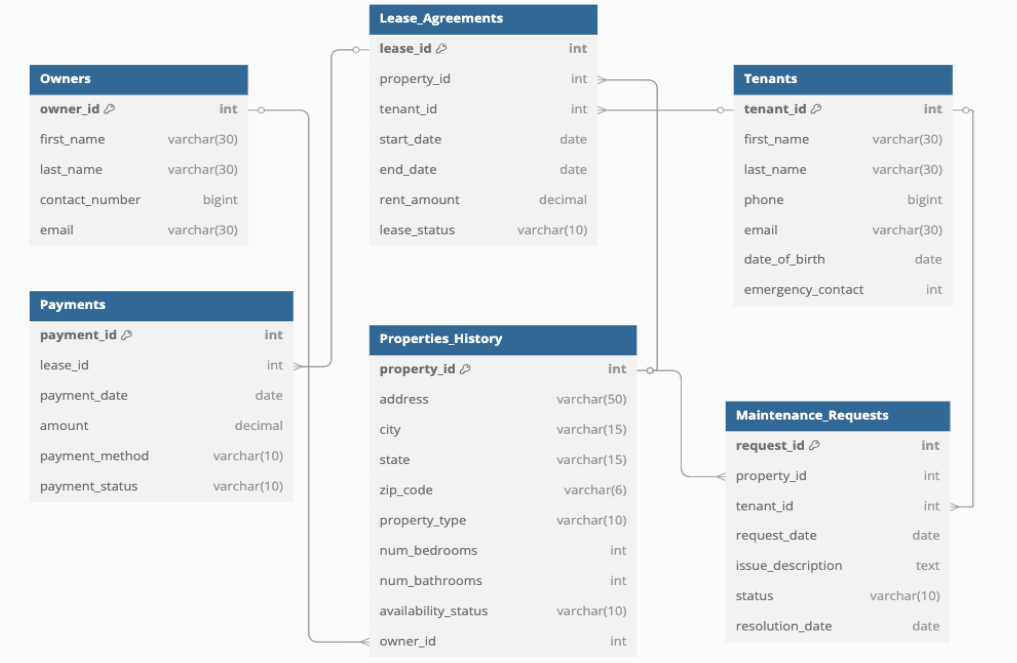
cost_estimate	
actual_cost	
feedback	
image_uploads	

Fields: request_id (PK), tenant_id (FK), property_id (FK), request_date, issue_description, status, resolution_date

Use: Manages maintenance workflows, service tracking, and response times

5. Entity Relationship Diagram

The ERD illustrates one-to-many relationships across all six primary tables. The system architecture ensures data consistency, traceability, and scalability.



5.1 Explanation of ER Diagram with Each Table:

5.1.1 Owners

Each owner is uniquely identified and can own multiple properties.

Relationship: Owners (1) → (N) Property

5.1.2 Property

Each property is tied to an owner and may have multiple lease agreements and maintenance logs.

Central hub for operations.

5.1.3 Tenants

Tenants can lease multiple properties over time and can submit maintenance requests.

Relationship: Tenants (1) → (N) Lease_Agreements and Maintenance_Requests

5.1.4 Lease_Agreements

Bridges Tenants and Properties.

Stores key financial and contractual terms.

Relationship: Lease_Agreements (1) → (N) Payments

5.1.5 Payments

Tracks recurring rent payments.

Facilitates financial oversight.

5.1.6 Maintenance_Requests

Maintains tenant-submitted requests and their resolution progress.

Links tenants to properties through requests.

5.2 Relationship Table for Viventra Property Management System:

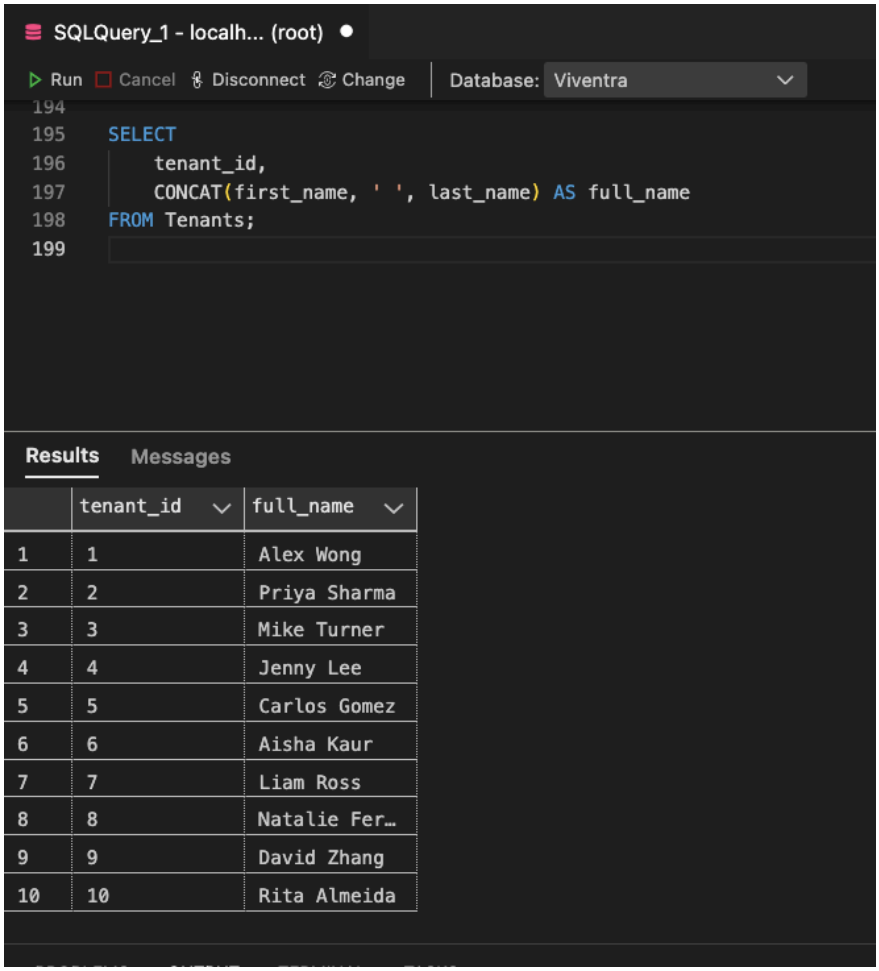
Parent Table	Child Table	Relationship Type
Owners	Property	One-to-Many
Property	Lease_Agreements	One-to-Many
Tenants	Lease_Agreements	One-to-Many
Lease_Agreements	Payments	One-to-Many
Tenants	Maintenance_Requests	One-to-Many
Property	Maintenance_Requests	One-to-Many

6. SQL Database Queries

6.1 Query: Tenant Full Name

SELECT tenant_id, CONCAT(first_name, ' ', last_name) AS full_name

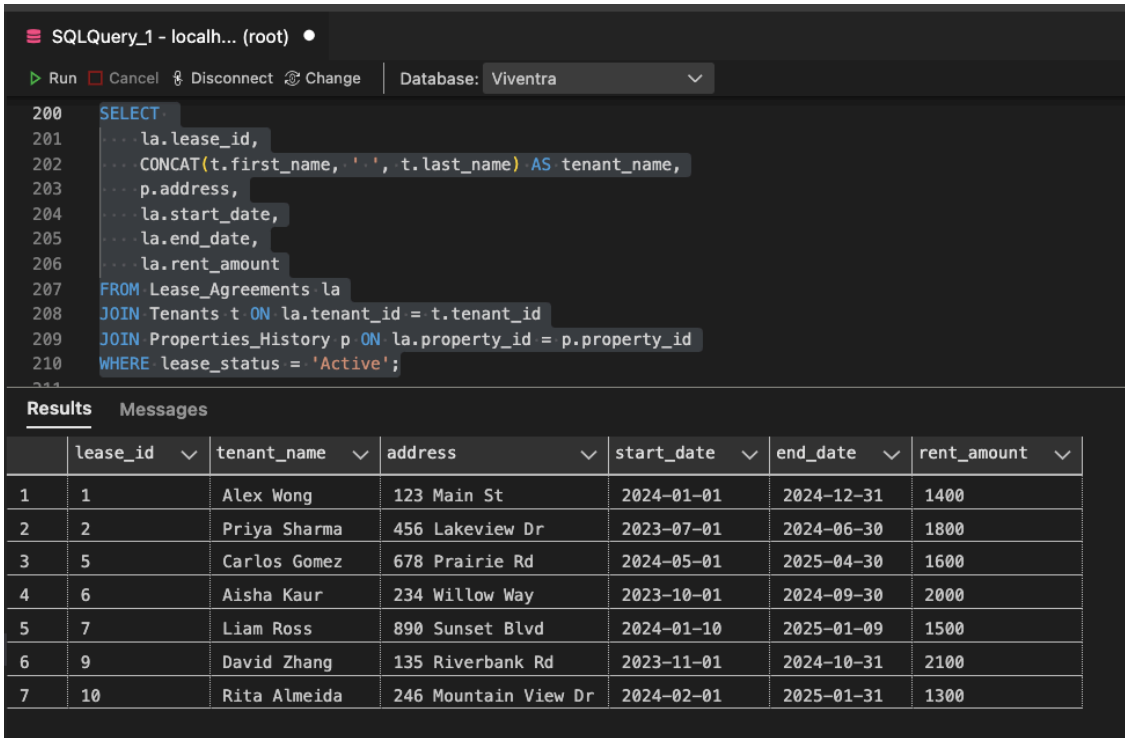
FROM Tenants;



	tenant_id	full_name
1	1	Alex Wong
2	2	Priya Sharma
3	3	Mike Turner
4	4	Jenny Lee
5	5	Carlos Gomez
6	6	Aisha Kaur
7	7	Liam Ross
8	8	Natalie Fer…
9	9	David Zhang
10	10	Rita Almeida

6.2 Query: Lease Status Active

```
SELECT la.lease_id, CONCAT(t.first_name, ' ', t.last_name) AS tenant_name,
    p.address, la.start_date, la.end_date, la.rent_amount
FROM Lease_Agreements la
JOIN Tenants t ON la.tenant_id = t.tenant_id
JOIN Properties_History p ON la.property_id = p.property_id
WHERE lease_status = 'Active';
```



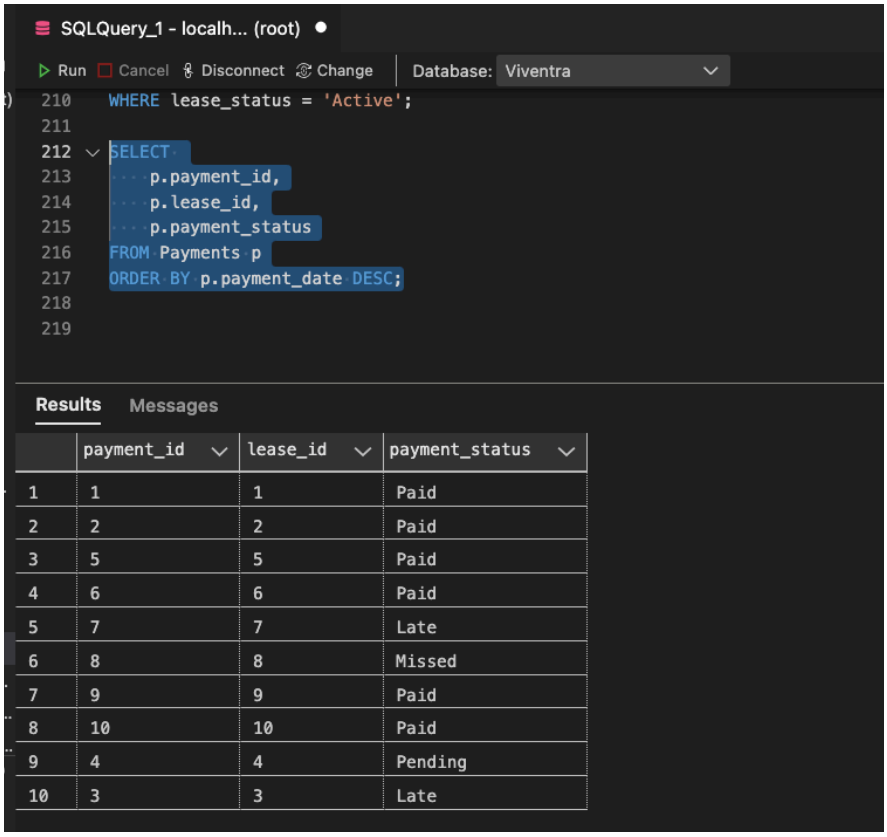
	lease_id	tenant_name	address	start_date	end_date	rent_amount
1	1	Alex Wong	123 Main St	2024-01-01	2024-12-31	1400
2	2	Priya Sharma	456 Lakeview Dr	2023-07-01	2024-06-30	1800
3	5	Carlos Gomez	678 Prairie Rd	2024-05-01	2025-04-30	1600
4	6	Aisha Kaur	234 Willow Way	2023-10-01	2024-09-30	2000
5	7	Liam Ross	890 Sunset Blvd	2024-01-10	2025-01-09	1500
6	9	David Zhang	135 Riverbank Rd	2023-11-01	2024-10-31	2100
7	10	Rita Almeida	246 Mountain View Dr	2024-02-01	2025-01-31	1300

6.3 Query: Payment Status for Each Lease

SELECT payment_id, lease_id, payment_status

FROM Payments

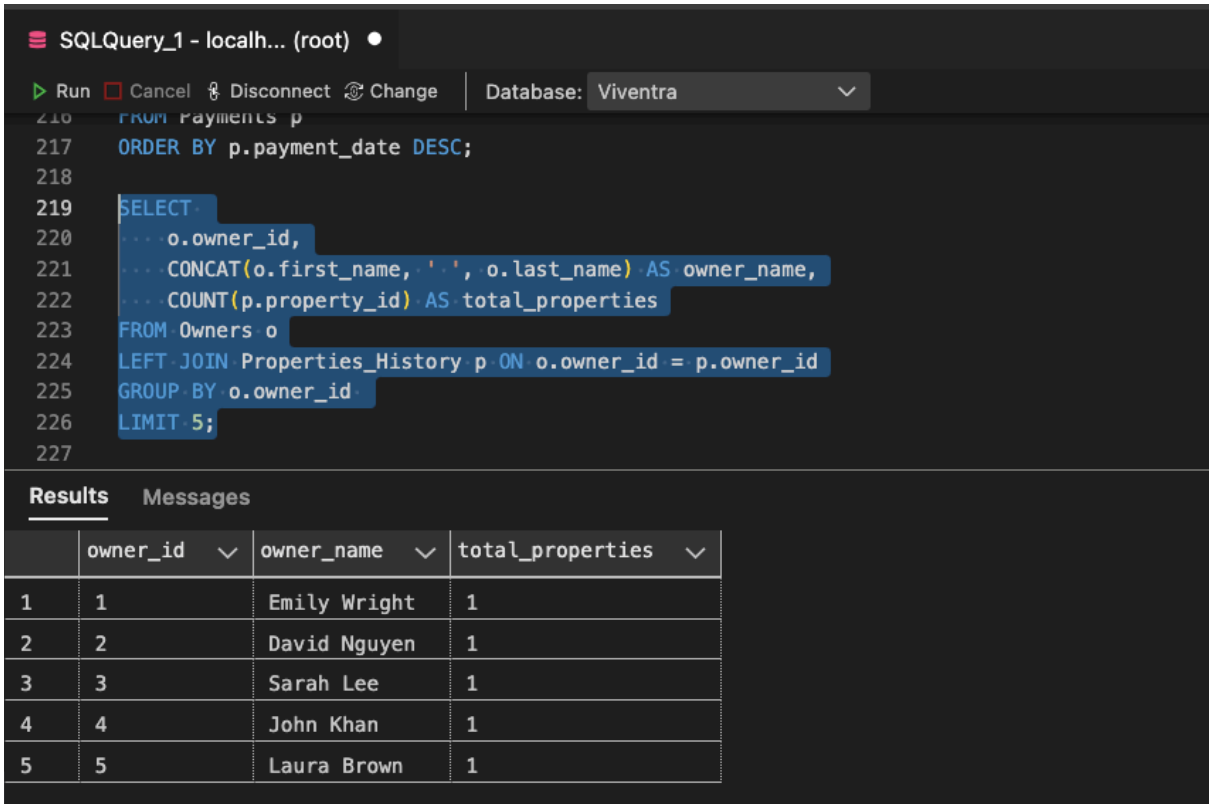
ORDER BY payment_date DESC;



	payment_id	lease_id	payment_status
1	1	1	Paid
2	2	2	Paid
3	5	5	Paid
4	6	6	Paid
5	7	7	Late
6	8	8	Missed
7	9	9	Paid
8	10	10	Paid
9	4	4	Pending
10	3	3	Late

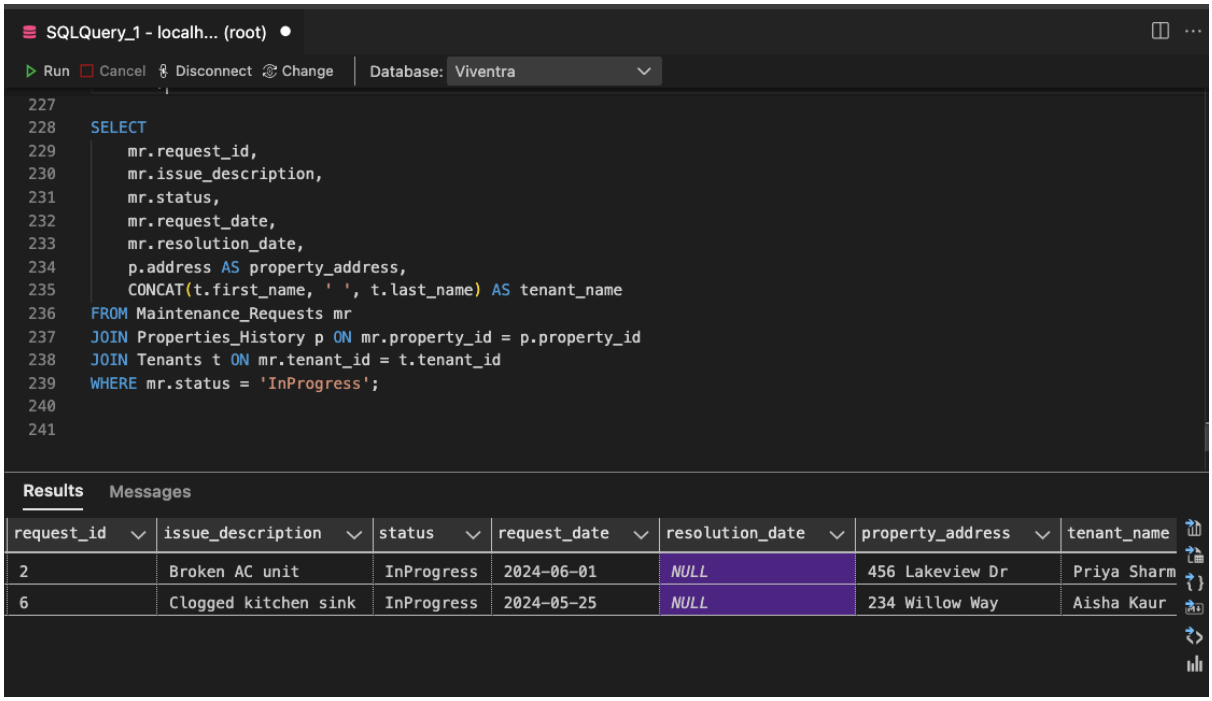
6.4 Query: Property Count per Owner

```
SELECT o.owner_id, CONCAT(o.first_name, ' ', o.last_name) AS owner_name,
    COUNT(p.property_id) AS total_properties
FROM Owners o
LEFT JOIN Properties_History p ON o.owner_id = p.owner_id
GROUP BY o.owner_id
LIMIT 5;
```

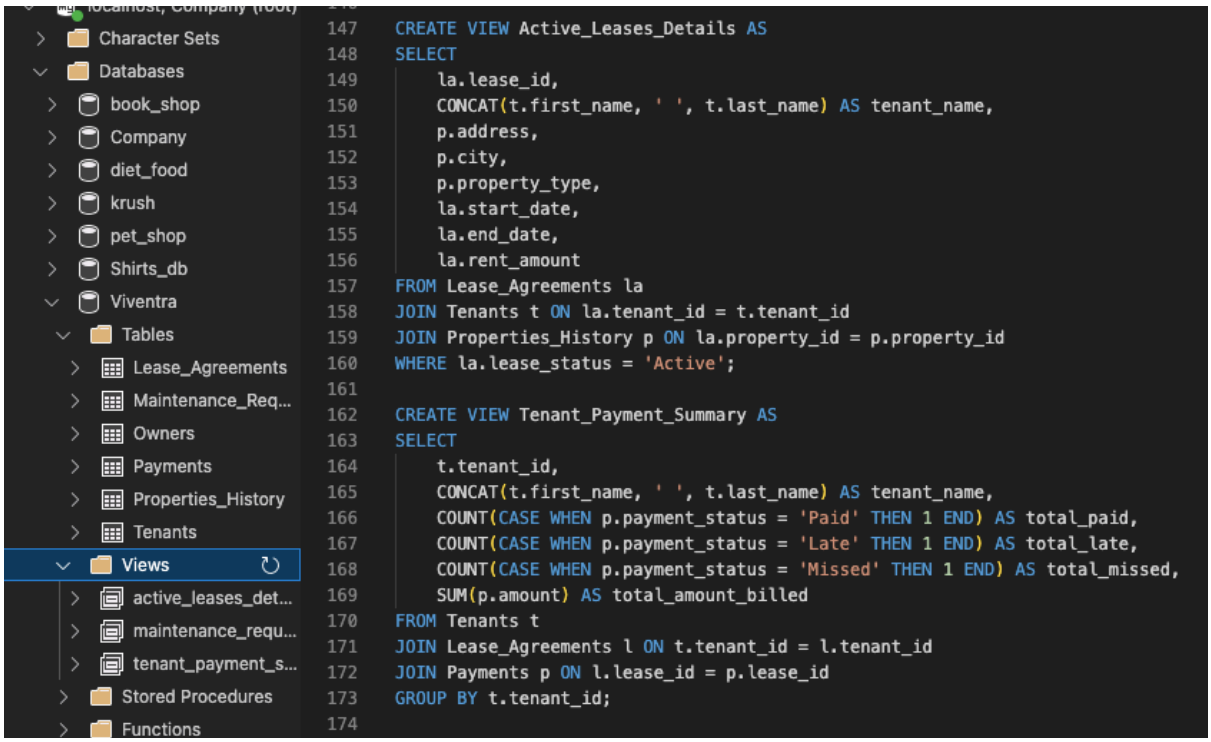


6.5 Query: In-Progress Maintenance Details

```
SELECT mr.request_id, mr.issue_description, mr.status,
    mr.request_date, mr.resolution_date,
    p.address AS property_address,
    CONCAT(t.first_name, ' ', t.last_name) AS tenant_name
FROM Maintenance_Requests mr
JOIN Properties_History p ON mr.property_id = p.property_id
JOIN Tenants t ON mr.tenant_id = t.tenant_id
WHERE mr.status = 'InProgress';
```



7. SQL Queries for Creating Views



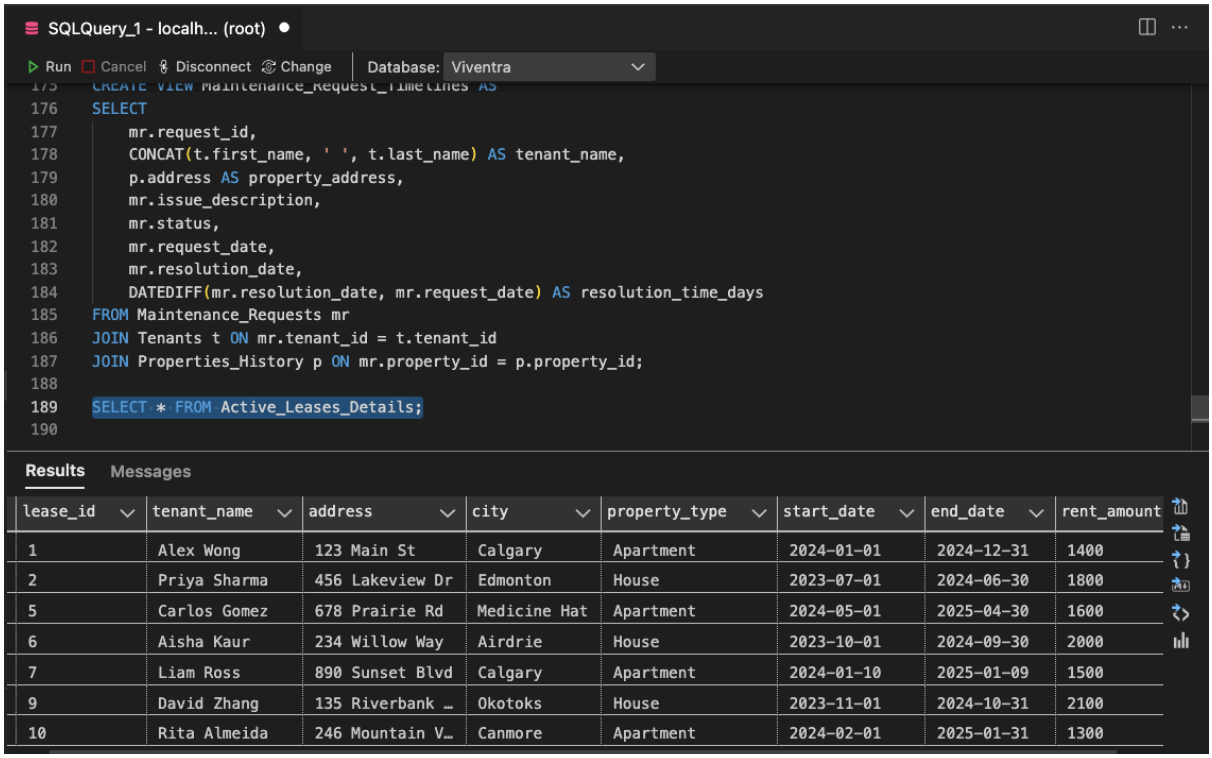

```sql
CREATE VIEW Active_Leases_Details AS
SELECT
    la.lease_id,
    CONCAT(t.first_name, ' ', t.last_name) AS tenant_name,
    p.address,
    p.city,
    p.property_type,
    la.start_date,
    la.end_date,
    la.rent_amount
FROM Lease_Agreements la
JOIN Tenants t ON la.tenant_id = t.tenant_id
JOIN Properties_History p ON la.property_id = p.property_id
WHERE la.lease_status = 'Active';

CREATE VIEW Tenant_Payment_Summary AS
SELECT
    t.tenant_id,
    CONCAT(t.first_name, ' ', t.last_name) AS tenant_name,
    COUNT(CASE WHEN p.payment_status = 'Paid' THEN 1 END) AS total_paid,
    COUNT(CASE WHEN p.payment_status = 'Late' THEN 1 END) AS total_late,
    COUNT(CASE WHEN p.payment_status = 'Missed' THEN 1 END) AS total_missed,
    SUM(p.amount) AS total_amount_billed
FROM Tenants t
JOIN Lease_Agreements l ON t.tenant_id = l.tenant_id
JOIN Payments p ON l.lease_id = p.lease_id
GROUP BY t.tenant_id;
```

```sql
CREATE VIEW Maintenance_Request_Timelines AS
SELECT
    mr.request_id,
    CONCAT(t.first_name, ' ', t.last_name) AS tenant_name,
    p.address AS property_address,
    mr.issue_description,
    mr.status,
    mr.request_date,
    mr.resolution_date,
    DATEDIFF(mr.resolution_date, mr.request_date) AS resolution_time_days
FROM Maintenance_Requests mr
JOIN Tenants t ON mr.tenant_id = t.tenant_id
JOIN Properties_History p ON mr.property_id = p.property_id;
```

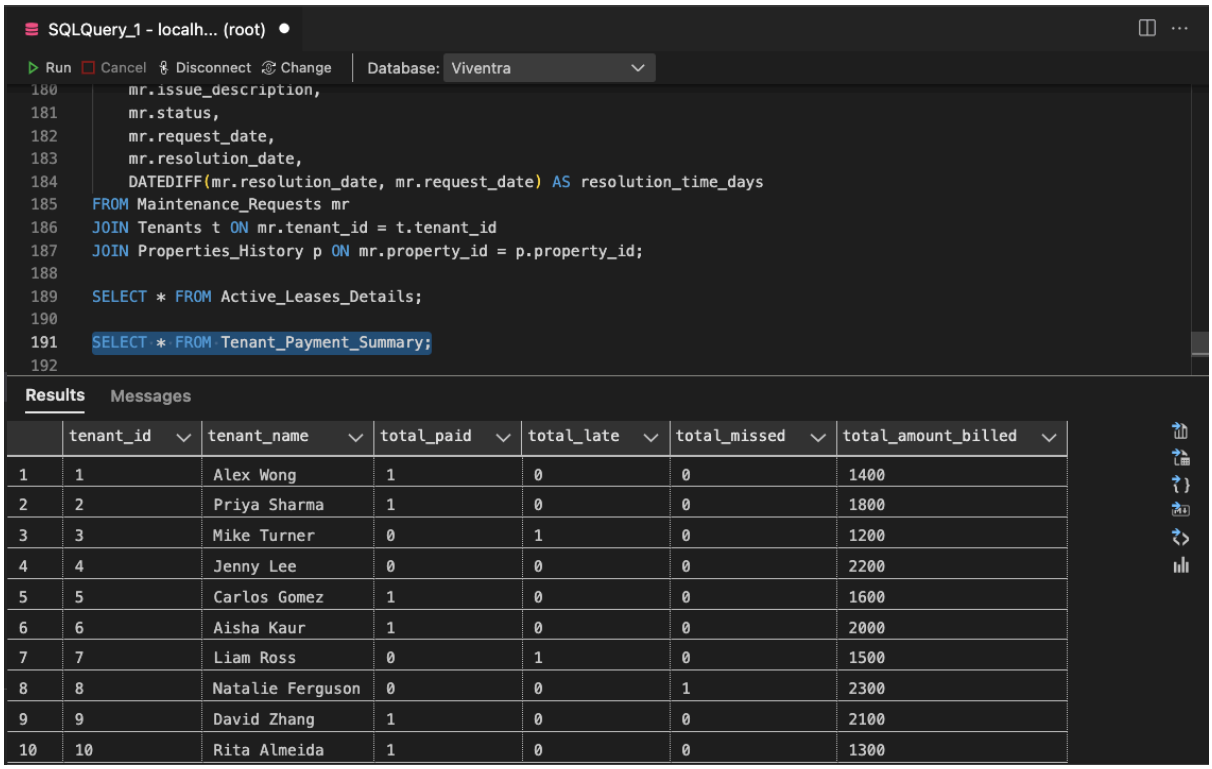
7.1 View: Active Leases Details

```
CREATE VIEW Active_Leases_Details AS
SELECT la.lease_id, CONCAT(t.first_name, ' ', t.last_name) AS tenant_name,
    p.address, p.city, p.property_type,
    la.start_date, la.end_date, la.rent_amount
FROM Lease_Agreements la
JOIN Tenants t ON la.tenant_id = t.tenant_id
JOIN Properties_History p ON la.property_id = p.property_id
WHERE la.lease_status = 'Active';
```



7.2 View: Tenant Payment Summary

```
CREATE VIEW Tenant_Payment_Summary AS
SELECT t.tenant_id, CONCAT(t.first_name, ' ', t.last_name) AS tenant_name,
    COUNT(CASE WHEN p.payment_status = 'Paid' THEN 1 END) AS total_paid,
    COUNT(CASE WHEN p.payment_status = 'Late' THEN 1 END) AS total_late,
    COUNT(CASE WHEN p.payment_status = 'Missed' THEN 1 END) AS total_missed,
    SUM(p.amount) AS total_amount_billed
FROM Tenants t
JOIN Lease_Agreements l ON t.tenant_id = l.tenant_id
JOIN Payments p ON l.lease_id = p.lease_id
GROUP BY t.tenant_id;
```



	tenant_id	tenant_name	total_paid	total_late	total_missed	total_amount_billed
1	1	Alex Wong	1	0	0	1400
2	2	Priya Sharma	1	0	0	1800
3	3	Mike Turner	0	1	0	1200
4	4	Jenny Lee	0	0	0	2200
5	5	Carlos Gomez	1	0	0	1600
6	6	Aisha Kaur	1	0	0	2000
7	7	Liam Ross	0	1	0	1500
8	8	Natalie Ferguson	0	0	1	2300
9	9	David Zhang	1	0	0	2100
10	10	Rita Almeida	1	0	0	1300

7.3 View: Maintenance Request Timelines

```
CREATE VIEW Maintenance_Request_Timelines AS
SELECT mr.request_id, CONCAT(t.first_name, ' ', t.last_name) AS tenant_name,
    p.address AS property_address,
    mr.issue_description, mr.status,
    mr.request_date, mr.resolution_date,
    DATEDIFF(mr.resolution_date, mr.request_date) AS resolution_time_days
FROM Maintenance_Requests mr
JOIN Tenants t ON mr.tenant_id = t.tenant_id
JOIN Properties_History p ON mr.property_id = p.property_id;
```