DAMG-7275 Project Database Design

Property Management Database Management System

Project Team 3

Ali Jabbarzadeh Ghandilou

Shima Bolboli

Azadeh Bakhshi

Supervisor: Prof. Mazhar

Toronto - Dec. 2023

Table of Contents:

Purpose of the Database

The purpose of the property management database is to serve as a robust and centralized platform to facilitate efficient and effective management of residential and commercial properties. It aims to streamline operations, enhance decision-making, and optimize workflows within the property management domain. The database serves as a cornerstone for various stakeholders involved in property management, including property owners, managers, tenants, vendors, and staff.

Elaboration:

The database seeks to address the challenges faced in the property management industry by offering a comprehensive solution that centralizes property-related data, automates processes, and provides analytical insights. It aims to:

Consolidate Property Information: Gather and organize property details, ownership information, lease agreements, and financial records into a single repository.

Facilitate Decision-Making: Provide tools and reports for data-driven decision-making, enabling property owners and managers to make informed strategic choices.

Enhance Tenant Experience: Offer streamlined interactions, address tenant needs promptly, and provide a platform for easy communication and issue resolution.

Ensure Regulatory Compliance: Implement robust security measures and comply with industry standards and regulations to safeguard sensitive property and tenant information.

Specific Goals or Objectives

The specific goals of the property management database are tailored to achieve targeted outcomes in the realm of property administration, financial management, tenant relations, and operational efficiency. These objectives align with the broader purpose of the database and are designed to enhance the property management experience while optimizing administrative processes.

Elaboration:

Efficient Property Management: Capture, store, and manage property details comprehensively, ensuring accurate and up-to-date records of properties under management.

Transparent Financial Management: Record and categorize financial transactions with accuracy and clarity, enabling precise financial tracking and reporting.

Streamlined Tenant Management: Maintain detailed tenant profiles, lease agreements, and payment histories, facilitating smooth tenant relations and minimizing disputes.

Automated Maintenance and Services: Manage maintenance requests, vendor assignments, and task completion status, ensuring swift and efficient property upkeep.

Robust Reporting and Analytics: Develop reporting tools and analytics for performance evaluation, allowing stakeholders to derive insights for better decision-making.

Functionalities Supported

The functionalities provided by the database encompass a wide array of capabilities, addressing different aspects of property management, financial tracking, tenant engagement, maintenance, and analytical reporting.

Elaboration:

Property Information Management: Capture property details, amenities, ownership information, and historical data to maintain a comprehensive repository of property information.

Tenant and Lease Management: Record tenant information, lease agreements, payment histories, and preferences, ensuring accurate tenant-landlord relationships.

Financial Transaction Tracking: Record and categorize financial transactions, including rents, expenses, vendor payments, and invoices for transparent financial management.

Maintenance and Service Tracking: Log maintenance requests, assign tasks to vendors, track progress, and ensure timely completion for property upkeep.

Document Management: Store and manage property-related documents, contracts, and agreements for easy access and compliance.

Reporting and Analytics: Develop reporting tools and analytics to derive insights into financial performance, occupancy rates, and property maintenance metrics.

Business Rules:

1. Each property must have a unique identifier (PropertyID) and detailed address information.

2. Tenants can have multiple lease agreements, but each lease is associated with only one tenant.

3. Financial transactions should have clear categorization (rent, expenses, vendor transactions).

4. Maintenance requests should be linked to specific properties and tenants.

5. Documents must be categorized by type (e.g., lease agreements, maintenance records).

6. A lease agreement must specify a property, its terms (start date, end date), and rental amount.

7. Financial transactions must balance, ensuring that the sum of all income and expenses matches the financial records' total.

8. Transactions should adhere to predefined categories (e.g., rent, utilities, maintenance) to maintain accurate financial reporting.

9. A resolution status must be assigned upon completion (e.g., resolved, pending) with associated completion dates.

10. Document versions must be tracked and managed to ensure that the most recent and relevant version is readily accessible.

11. Tenants should only be associated with properties designated for residential or commercial purposes based on their lease agreements.

12. Tenants' access or control over certain property areas (e.g., parking spaces) must align with lease terms and agreements.

Entity tables:

Property Table:

Attribute	Data Type
PropertyID	NUMBER
Address	VARCHAR(100)
Type	VARCHAR(50)
Size	NUMERIC
Amenities	VARCHAR(200)
OwnershipDetails	VARCHAR(200)

Insurance Table:

Attribute	Data Type
InsuranceID	NUMBER
PropertyID	NUMBER
InsuranceProvider	VARCHAR(100)
CoverageDetails	VARCHAR(500)
ExpiryDate	DATE

Calendar Table:

Attribute	Data Type
EventID	NUMBER
PropertyID	NUMBER
EventType	VARCHAR(50)
EventDate	DATE
Description	VARCHAR(200)

Vendor Table:

Attribute	Data Type
VendorID	NUMBER
Name	VARCHAR(100)
ContactInformation	VARCHAR(100)

ServicesProvided	VARCHAR(200)
ContractDetails	VARCHAR(200)

User Accounts Table:

Attribute	Data Type
UserID	NUMBER
PropertyID	NUMBER
Name	VARCHAR(100)
ContactInformation	VARCHAR(100)
Role	VARCHAR(50)
Permissions	VARCHAR(200)

Tenant Table:

Attribute	Data Type
TenantID	NUMBER
PropertyID	NUMBER
Name	VARCHAR(100)
ContactInformation	VARCHAR(100)
LeaseDetails	VARCHAR(200)
PaymentHistory	VARCHAR(200)
Preferences	VARCHAR(200)

Lease Terms Table:

Attribute	Data Type
LeaseTermID	NUMBER
TenantID	NUMBER
StartDate	DATE
EndDate	DATE
ReneentAmount	INTEGER
PaymentSchedule	VARCHAR2(50)
Deposit	NUMBER

Emergency Contacts Table:

Attribute	**Data Type**
EmergencyContactID	NUMBER
PropertyID	NUMBER
TenantID	NUMBER
ContactType	VARCHAR(50)
ContactInformation	VARCHAR(100)

Document Table:

Attribute	**Data Type**
DocumentID	NUMBER
PropertyID	NUMBER
TenantID	NUMBER
VendorID	NUMBER
DocumentType	VARCHAR(50)
Title	VARCHAR(100)
UploadDate	DATE

Parking Management Table:

Attribute	**Data Type**
ParkingID	NUMBER
PropertyID	NUMBER
TenantID	NUMBER
ParkingSpaceNumber	VARCHAR(50)
AvailabilityStatus	VARCHAR(50)

Financial Transactions Table:

Attribute	**Data Type**
TransactionID	NUMBER
PropertyID	NUMBER
TenantID	NUMBER
VendorID	NUMBER
TransactionType	VARCHAR(50)
Amount	NUMERIC

TransactionDate	DATE
PaymentStatus	VARCHAR(50)

Utility Management Table:

Attribute	**Data Type**
UtilityID	NUMBER
PropertyID	NUMBER
UtilityType	VARCHAR(50)
Provider	VARCHAR(100)
BillingDetails	VARCHAR(500)

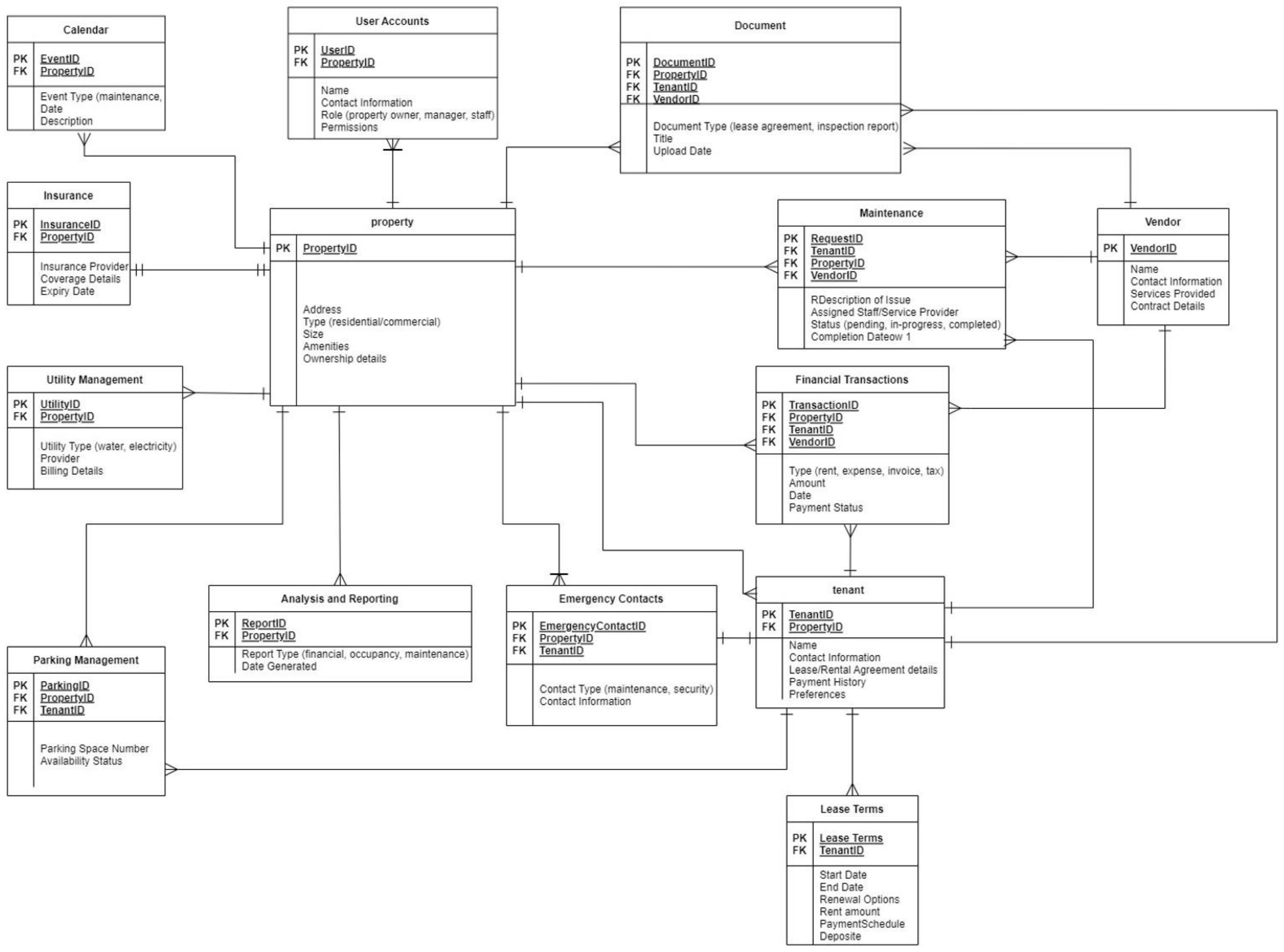
Maintenance Table:

Attribute	**Data Type**
RequestID	NUMBER
PropertyID	NUMBER
TenantID	NUMBER
VendorID	NUMBER
IssueDescription	VARCHAR(500)
AssignedStaff	VARCHAR(100)
Status	VARCHAR(50)
CompletionDate	DATE

Analysis and Reporting Table:

Attribute	**Data Type**
ReportID	NUMBER
PropertyID	NUMBER
ReportType	VARCHAR(50)
DateGenerated	DATE
Insights	VARCHAR(500)

ERD:


Calendar
PK EventID
FK PropertyID
Event Type (maintenance,
Date
Description
User Accounts
PK UserID
FK PropertyID
Name
Contact Information
Role (property owner, manager, staff)
Permissions
Document
PK DocumentID
FK PropertyID
FK TenantID
FK VendorID
Document Type (lease agreement, inspection report)
Title
Upload Date
Insurance
PK InsuranceID
FK PropertyID
Insurance Provider
Coverage Details
Expiry Date
property
PK PropertyID
Address
Type (residential/commercial)
Size
Amenities
Ownership details
Maintenance
PK RequestID
FK TenantID
FK PropertyID
FK VendorID
RDescription of Issue
Assigned Staff/Service Provider
Status (pending, in-progress, completed)
Completion Dateow 1
Vendor
PK VendorID
Name
Contact Information
Services Provided
Contract Details
Utility Management
PK UtilityID
FK PropertyID
Utility Type (water, electricity)
Provider
Billing Details
Financial Transactions
PK TransactionID
FK PropertyID
FK TenantID
FK VendorID
Type (rent, expense, invoice, tax)
Amount
Date
Payment Status
Analysis and Reporting
PK ReportID
FK PropertyID
Report Type (financial, occupancy, maintenance)
Date Generated
Emergency Contacts
PK EmergencyContactID
FK PropertyID
FK TenantID
Contact Type (maintenance, security)
Contact Information
tenant
PK TenantID
FK PropertyID
Name
Contact Information
Lease/Rental Agreement details
Payment History
Preferences
Parking Management
PK ParkingID
FK PropertyID
FK TenantID
Parking Space Number
Availability Status
Lease Terms
PK Lease Terms
FK TenantID
Start Date
End Date
Renewal Options
Rent amount
PaymentSchedule
Deposite

ERD Relationships:

User Accounts - Property (Many-to-One):

Many user accounts can be associated with one property.

Vendor - Maintenance, Financial Transactions, Documents (One-to-Many):

One vendor can be linked to many maintenance requests, financial transactions, and documents.

Property - Various Entities (One-to-Many):

One property can be associated with many tenants, maintenance requests, financial transactions, calendar events, documents, emergency contacts, parking spaces, and utility records.

Insurance - Property (One-to-One):

Each insurance record is linked to one specific property.

Tenant - Various Entities (One-to-Many):

One tenant can be associated with many lease terms, financial transactions, documents, emergency contacts, and parking spaces.

Lease Terms - Tenant (One-to-One or One-to-Many):

Each lease term is linked to one tenant.

Emergency Contacts - Tenant, Property (One-to-One or One-to-Many):

An emergency contact can be linked to one tenant or one property, and one tenant or property can have many emergency contacts.

Document - Tenant, Property, Vendor (One-to-One or One-to-Many):

One document can belong to one tenant, property, or vendor.

Parking Management - Tenant, Property (One-to-One or One-to-Many):

One parking space can belong to one tenant or one property.

Financial Transactions - Property, Tenant, Vendor (One-to-One or One-to-Many):

One financial transaction can belong to one property, tenant, or vendor.

Utility Management - Property (One-to-One or One-to-Many):

One utility record can belong to one property.

Maintenance - Property, Tenant, Vendor (One-to-One or One-to-Many):

One maintenance request can belong to one property, tenant, or vendor.

Analysis and Reporting - Property (One-to-Many):

One property can have many reports providing insights.

Physical Database Design:

Sequences:

-- Creating sequences

1. CREATE SEQUENCE P_PropertySeq START WITH 1 INCREMENT BY 1;
2. CREATE SEQUENCE P_InsuranceSeq START WITH 11 INCREMENT BY 1;
3. CREATE SEQUENCE P_CalendarSeq START WITH 101 INCREMENT BY 1;
4. CREATE SEQUENCE P_VendorSeq START WITH 201 INCREMENT BY 1;
5. CREATE SEQUENCE P_UserAccountsSeq START WITH 301 INCREMENT BY 1;
6. CREATE SEQUENCE P_TenantSeq START WITH 401 INCREMENT BY 1;
7. CREATE SEQUENCE P_LeaseTermsSeq START WITH 501 INCREMENT BY 1;
8. CREATE SEQUENCE P_EmergencyContactsSeq START WITH 601 INCREMENT BY 1;
9. CREATE SEQUENCE P_DocumentSeq START WITH 701 INCREMENT BY 1;
10. CREATE SEQUENCE P_ParkingManagementSeq START WITH 801 INCREMENT BY 1;
11. CREATE SEQUENCE P_FinancialTransactionsSeq START WITH 901 INCREMENT BY 1;
12. CREATE SEQUENCE P_UtilityManagementSeq START WITH 1001 INCREMENT BY 1;
13. CREATE SEQUENCE P_MaintenanceSeq START WITH 1101 INCREMENT BY 1;
14. CREATE SEQUENCE P_AnalysisAndReportingSeq START WITH 1201 INCREMENT BY 1;

Views:

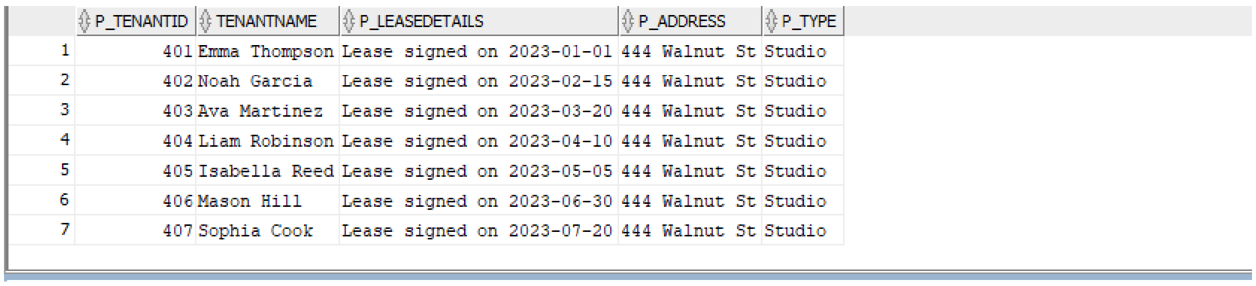
1--presents detailed lease information along with the respective property details:

CREATE VIEW TenantLeasePropertyView AS

SELECT T.P_TenantID, T.P_Name AS TenantName, T.P_LeaseDetails, P.P_Address, P.P_Type

FROM P_Tenant T

INNER JOIN P_Property P ON T.P_PropertyID = P.P_PropertyID;



	P_TENANTID	TENANTNAME	P_LEASEDETAILS	P_ADDRESS	P_TYPE
1	401	Emma Thompson	Lease signed on 2023-01-01	444 Walnut St	Studio
2	402	Noah Garcia	Lease signed on 2023-02-15	444 Walnut St	Studio
3	403	Ava Martinez	Lease signed on 2023-03-20	444 Walnut St	Studio
4	404	Liam Robinson	Lease signed on 2023-04-10	444 Walnut St	Studio
5	405	Isabella Reed	Lease signed on 2023-05-05	444 Walnut St	Studio
6	406	Mason Hill	Lease signed on 2023-06-30	444 Walnut St	Studio
7	407	Sophia Cook	Lease signed on 2023-07-20	444 Walnut St	Studio

2----calculates the total transaction amounts for each property from the P_FinancialTransactions table:

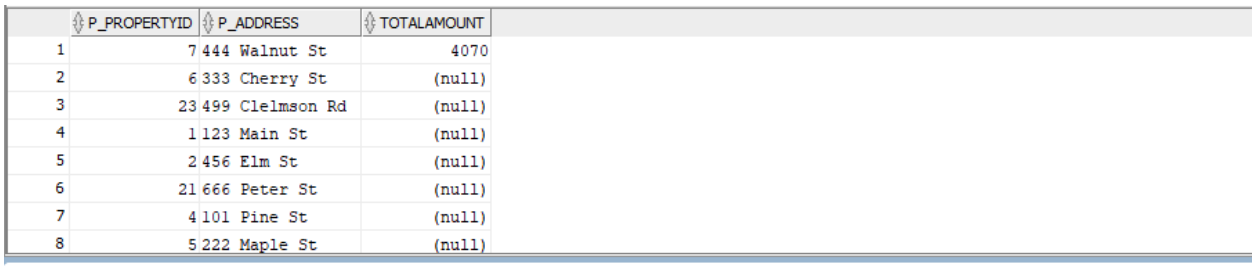
CREATE VIEW PropertyFinancialsView AS

SELECT P.P_PropertyID, P.P_Address, SUM(F.P_Amount) AS TotalAmount

FROM P_Property P

LEFT JOIN P_FinancialTransactions F ON P.P_PropertyID = F.P_PropertyID

GROUP BY P.P_PropertyID, P.P_Address;



	P_PROPERTYID	P_ADDRESS	TOTALAMOUNT
1	7	444 Walnut St	4070
2	6	333 Cherry St	(null)
3	23	499 Clelmson Rd	(null)
4	1	123 Main St	(null)
5	2	456 Elm St	(null)
6	21	666 Peter St	(null)
7	4	101 Pine St	(null)
8	5	222 Maple St	(null)

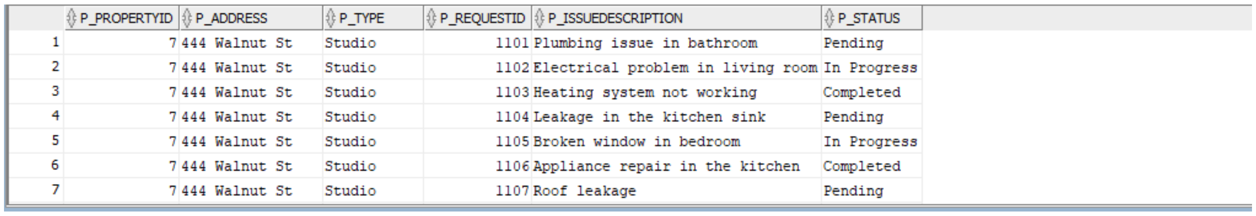
3-- Complex View involving multiple tables

CREATE VIEW PropertyMaintenanceView AS

SELECT P.P_PropertyID, P.P_Address, P.P_Type, M.P_RequestID, M.P_IssueDescription, M.P_Status

FROM P_Property P

LEFT JOIN P_Maintenance M ON P.P_PropertyID = M.P_PropertyID;



	P_PROPERTYID	P_ADDRESS	P_TYPE	P_REQUESTID	P_ISSUEDESCRIPTION	P_STATUS
1	7	444 Walnut St	Studio	1101	Plumbing issue in bathroom	Pending
2	7	444 Walnut St	Studio	1102	Electrical problem in living room	In Progress
3	7	444 Walnut St	Studio	1103	Heating system not working	Completed
4	7	444 Walnut St	Studio	1104	Leakage in the kitchen sink	Pending
5	7	444 Walnut St	Studio	1105	Broken window in bedroom	In Progress
6	7	444 Walnut St	Studio	1106	Appliance repair in the kitchen	Completed
7	7	444 Walnut St	Studio	1107	Roof leakage	Pending

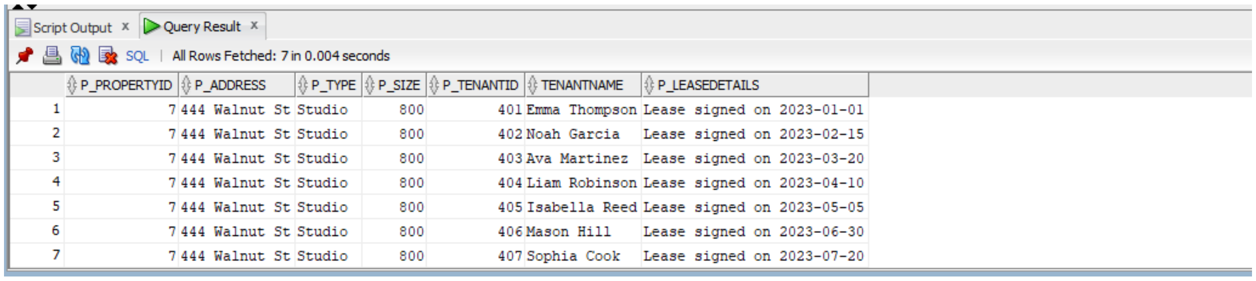
4-- Simple View combining Property and Tenant Information

CREATE VIEW PropertyTenantView AS

SELECT P.P_PropertyID, P.P_Address, P.P_Type, P.P_Size, T.P_TenantID, T.P_Name AS TenantName, T.P_LeaseDetails

FROM P_Property P

INNER JOIN P_Tenant T ON P.P_PropertyID = T.P_PropertyID;



	P_PROPERTYID	P_ADDRESS	P_TYPE	P_SIZE	P_TENANTID	TENANTNAME	P_LEASEDETAILS
1	7	444 Walnut St	Studio	800	401	Emma Thompson	Lease signed on 2023-01-01
2	7	444 Walnut St	Studio	800	402	Noah Garcia	Lease signed on 2023-02-15
3	7	444 Walnut St	Studio	800	403	Ava Martinez	Lease signed on 2023-03-20
4	7	444 Walnut St	Studio	800	404	Liam Robinson	Lease signed on 2023-04-10
5	7	444 Walnut St	Studio	800	405	Isabella Reed	Lease signed on 2023-05-05
6	7	444 Walnut St	Studio	800	406	Mason Hill	Lease signed on 2023-06-30
7	7	444 Walnut St	Studio	800	407	Sophia Cook	Lease signed on 2023-07-20

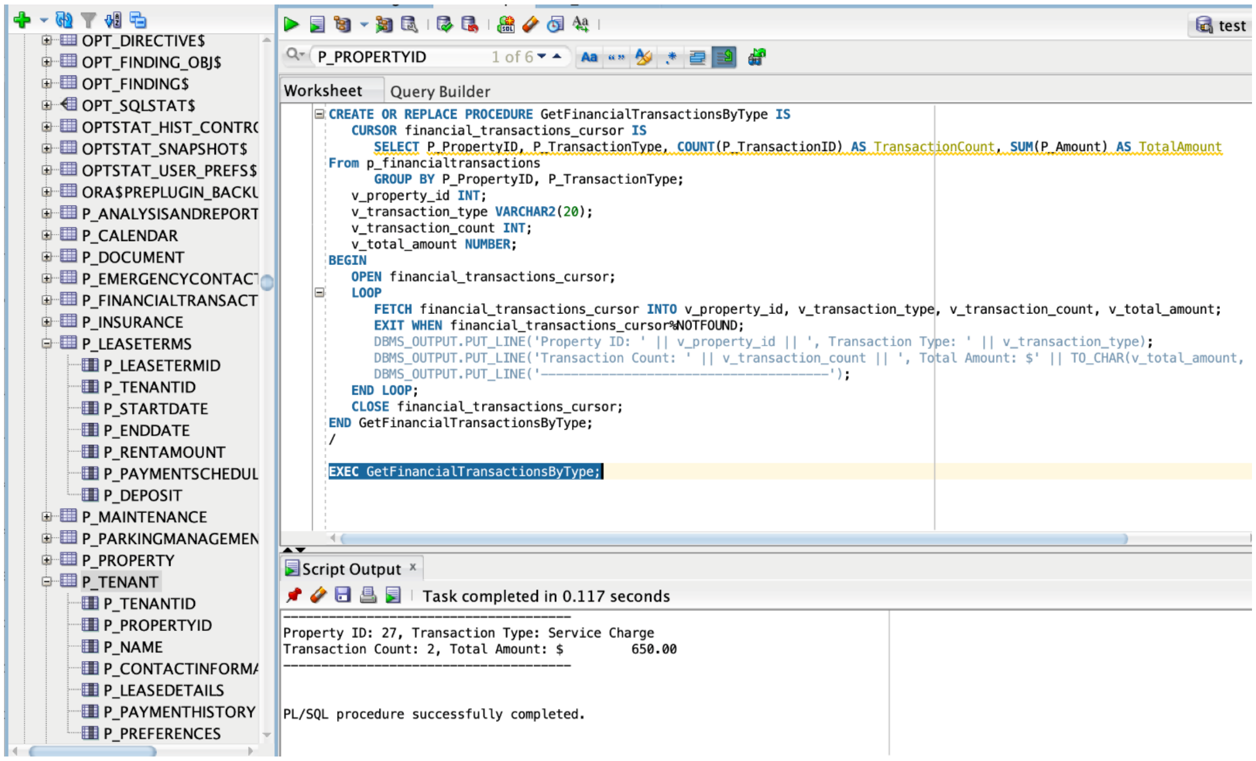
Cursor/Exception:

1- This cursor fetches financial transactions grouped by property and transaction type.

```
CREATE OR REPLACE PROCEDURE Getfinancialtransactionsbytype IS
  CURSOR Financial_Transactions_Cursor IS
    SELECT P_Propertyid, P_Transactiontype, COUNT(P_Transactionid) AS
Transactioncount, SUM(P_Amount) AS Totalamount
From P_Financialtransactions
    GROUP BY P_Propertyid, P_Transactiontype;
  V_Property_Id INT;
  V_Transaction_Type VARCHAR2(20);
  V_Transaction_Count INT;
  V_Total_Amount NUMBER;
BEGIN
  OPEN Financial_Transactions_Cursor;
  LOOP
    FETCH Financial_Transactions_Cursor INTO V_Property_Id, V_Transaction_Type,
V_Transaction_Count, V_Total_Amount;
    EXIT WHEN Financial_Transactions_Cursor%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE('Property ID: ' || V_Property_Id || ', Transaction Type: ' ||
V_Transaction_Type);
    DBMS_OUTPUT.PUT_LINE('Transaction Count: ' || V_Transaction_Count || ', Total
Amount: $' || TO_CHAR(V_Total_Amount, '999,999,999.99'));
    DBMS_OUTPUT.PUT_LINE('--------------------------------------');
  END LOOP;
  CLOSE Financial_Transactions_Cursor;
END Getfinancialtransactionsbytype;
/
```

- **Test**

```
EXEC Getfinancialtransactionsbytype;
```



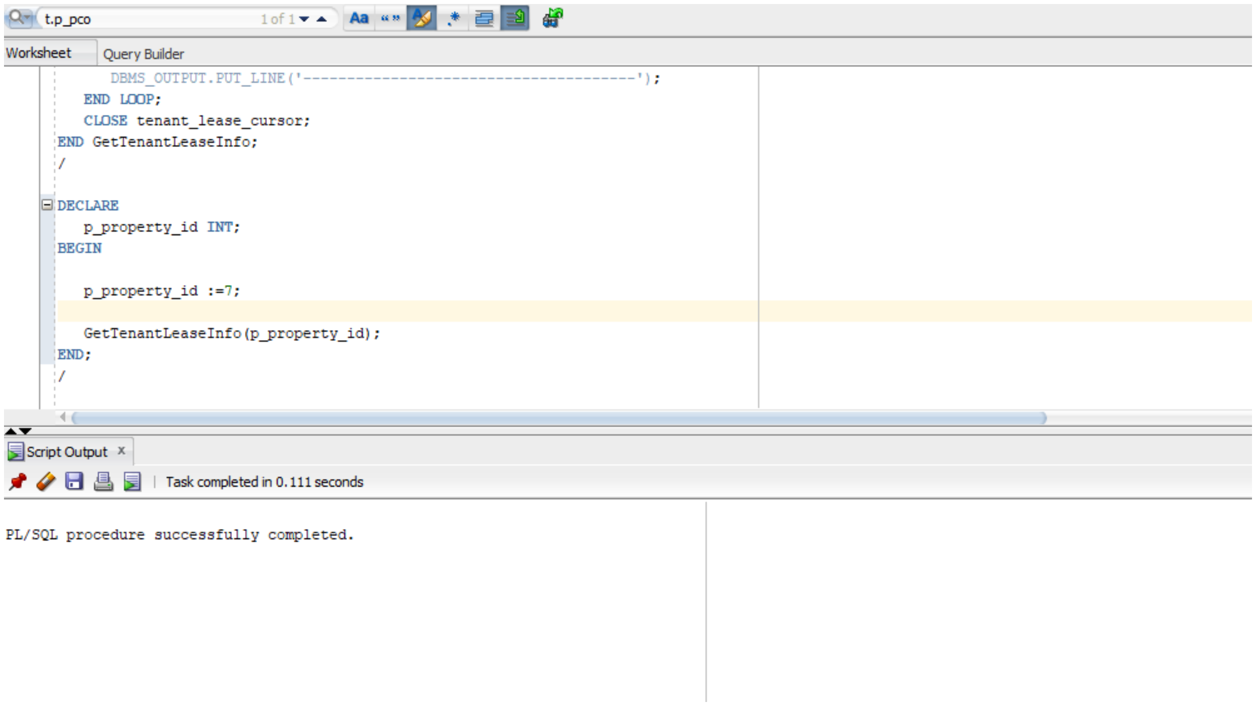
```
CREATE OR REPLACE PROCEDURE GetFinancialTransactionsByType IS
   CURSOR financial_transactions_cursor IS
      SELECT P_PropertyID, P_TransactionType, COUNT(P_TransactionID) AS TransactionCount, SUM(P_Amount) AS TotalAmount
From p_financialtransactions
      GROUP BY P_PropertyID, P_TransactionType;
   v_property_id INT;
   v_transaction_type VARCHAR2(20);
   v_transaction_count INT;
   v_total_amount NUMBER;
BEGIN
   OPEN financial_transactions_cursor;
   LOOP
      FETCH financial_transactions_cursor INTO v_property_id, v_transaction_type, v_transaction_count, v_total_amount;
      EXIT WHEN financial_transactions_cursor%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE('Property ID: ' || v_property_id || ', Transaction Type: ' || v_transaction_type);
      DBMS_OUTPUT.PUT_LINE('Transaction Count: ' || v_transaction_count || ', Total Amount: $' || TO_CHAR(v_total_amount,
      DBMS_OUTPUT.PUT_LINE('--------------------------------------');
   END LOOP;
   CLOSE financial_transactions_cursor;
END GetFinancialTransactionsByType;
/

EXEC GetFinancialTransactionsByType;
```

```
--------------------------------------
Property ID: 27, Transaction Type: Service Charge
Transaction Count: 2, Total Amount: $        650.00
--------------------------------------


PL/SQL procedure successfully completed.
```

2- This cursor fetches tenant details along with their lease terms for a specified property.

```
Create Or Replace Procedure Gettenantleaseinfo(P_Property_Id Int) Is
  Cursor Tenant_Lease_Cursor Is
    Select T.Tenantid, T.Name As Tenantname, T.Contactinformation As Tenantcontact,
         Lt.Leasetermid, Lt.Startdate, Lt.Enddate, Lt.Renewaloptions
    From Tenant T
    Join Leaseterms Lt On T.Tenantid = Lt.Tenantid
    Where T.Propertyid = P_Property_Id;

  V_Tenant_Id Int;
  V_Tenant_Name Varchar2(255);
  V_Tenant_Contact Varchar2(255);
  V_Lease_Id Int;
  V_Start_Date Date;
  V_End_Date Date;
  V_Renewal_Options Varchar2(255);
Begin
  Open Tenant_Lease_Cursor;
  Loop
```

```
      Fetch Tenant_Lease_Cursor Into V_Tenant_Id, V_Tenant_Name,
V_Tenant_Contact, V_Lease_Id, V_Start_Date, V_End_Date, V_Renewal_Options;
      Exit When Tenant_Lease_Cursor%Notfound;
      Dbms_Output.Put_Line('Tenant Id: ' || V_Tenant_Id || ', Name: ' || V_Tenant_Name ||
', Contact: ' || V_Tenant_Contact);
      Dbms_Output.Put_Line('Lease Id: ' || V_Lease_Id || ', Start Date: ' ||
To_Char(V_Start_Date, 'Dd-Mon-Yyyy') || ', End Date: ' || To_Char(V_End_Date, 'Dd-
Mon-Yyyy') || ', Renewal Options: ' || V_Renewal_Options);
      Dbms_Output.Put_Line('---------------------------------------');
   End Loop;
   Close Tenant_Lease_Cursor;
End Gettenantleaseinfo;
/
```

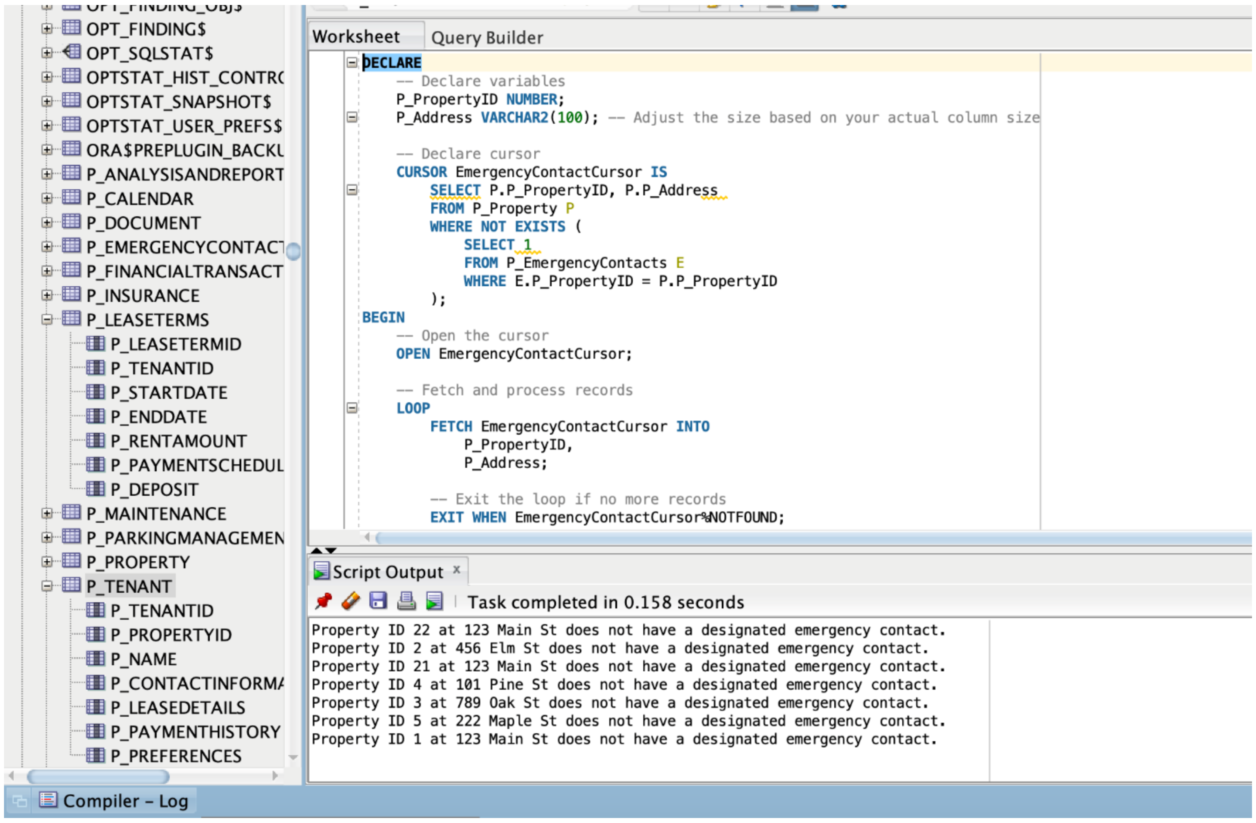


```
        DBMS_OUTPUT.PUT_LINE('---------------------------------------');
    END LOOP;
    CLOSE tenant_lease_cursor;
END GetTenantLeaseInfo;
/

DECLARE
    p_property_id INT;
BEGIN

    p_property_id :=7;

    GetTenantLeaseInfo(p_property_id);
END;
/
```

```
PL/SQL procedure successfully completed.
```

3- Each property should have designated emergency contacts. An emergency contact can be associated with multiple properties, but each property should have at least one designated emergency contact.

```
Declare
   Cursor Emergencycontactcursor Is
      Select P.P_Propertyid, P.P_Address
      From P_Property P
      Where Not Exists (
         Select 1
         From P_Emergencycontacts E
         Where E.P_Propertyid = P.P_Propertyid
      );
Begin
   -- Open The Cursor
   Open Emergencycontactcursor;

   -- Fetch And Process Records
   Loop
      Fetch Emergencycontactcursor Into
         P_Propertyid,
         P_Address;

      -- Exit The Loop If No More Records
      Exit When Emergencycontactcursor%Notfound;

      -- Print Or Handle The Properties Without Designated Emergency Contacts
      Dbms_Output.Put_Line('Property Id ' || P_Propertyid || ' At ' || P_Address || ' Does
Not Have A Designated Emergency Contact.');
   End Loop;

   -- Close The Cursor
   Close Emergencycontactcursor;
End;
/
```



```
DECLARE
    -- Declare variables
    P_PropertyID NUMBER;
    P_Address VARCHAR2(100); -- Adjust the size based on your actual column size

    -- Declare cursor
    CURSOR EmergencyContactCursor IS
        SELECT P.P_PropertyID, P.P_Address
        FROM P_Property P
        WHERE NOT EXISTS (
            SELECT 1
            FROM P_EmergencyContacts E
            WHERE E.P_PropertyID = P.P_PropertyID
        );
BEGIN
    -- Open the cursor
    OPEN EmergencyContactCursor;

    -- Fetch and process records
    LOOP
        FETCH EmergencyContactCursor INTO
            P_PropertyID,
            P_Address;

        -- Exit the loop if no more records
        EXIT WHEN EmergencyContactCursor%NOTFOUND;
```

```
Property ID 22 at 123 Main St does not have a designated emergency contact.
Property ID 2 at 456 Elm St does not have a designated emergency contact.
Property ID 21 at 123 Main St does not have a designated emergency contact.
Property ID 4 at 101 Pine St does not have a designated emergency contact.
Property ID 3 at 789 Oak St does not have a designated emergency contact.
Property ID 5 at 222 Maple St does not have a designated emergency contact.
Property ID 1 at 123 Main St does not have a designated emergency contact.
```

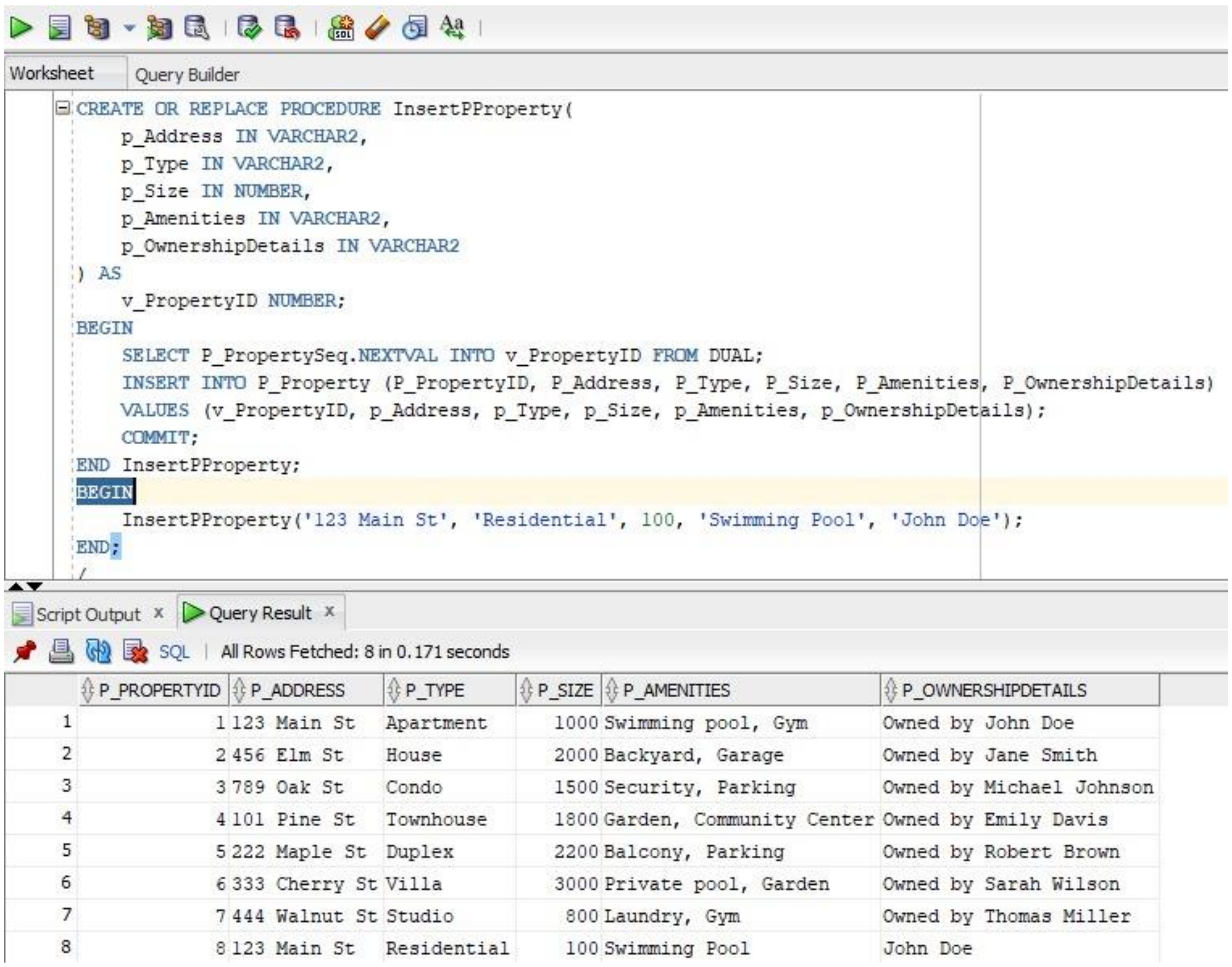
Procedure:

1- Insert Data into P_ Property

```
CREATE OR REPLACE PROCEDURE Insertpproperty(
   P_Address IN VARCHAR2,
   P_Type IN VARCHAR2,
   P_Size IN NUMBER,
   P_Amenities IN VARCHAR2,
   P_Ownershipdetails IN VARCHAR2
) AS
   V_Propertyid NUMBER;
BEGIN
   SELECT P_Propertyseq.NEXTVAL INTO V_Propertyid FROM DUAL;
   INSERT INTO P_Property (P_Propertyid, P_Address, P_Type, P_Size, P_Amenities,
P_Ownershipdetails)
   VALUES (V_Propertyid, P_Address, P_Type, P_Size, P_Amenities,
P_Ownershipdetails);
   COMMIT;
END Insertpproperty;
```

- **TEST**

```
BEGIN
   Insertpproperty('123 Main St', 'Residential', 100, 'Swimming Pool', 'John Doe');
END;
/
```



```
CREATE OR REPLACE PROCEDURE InsertPProperty(
    p_Address IN VARCHAR2,
    p_Type IN VARCHAR2,
    p_Size IN NUMBER,
    p_Amenities IN VARCHAR2,
    p_OwnershipDetails IN VARCHAR2
) AS
    v_PropertyID NUMBER;
BEGIN
    SELECT P_PropertySeq.NEXTVAL INTO v_PropertyID FROM DUAL;
    INSERT INTO P_Property (P_PropertyID, P_Address, P_Type, P_Size, P_Amenities, P_OwnershipDetails)
    VALUES (v_PropertyID, p_Address, p_Type, p_Size, p_Amenities, p_OwnershipDetails);
    COMMIT;
END InsertPProperty;
BEGIN
    InsertPProperty('123 Main St', 'Residential', 100, 'Swimming Pool', 'John Doe');
END;
/
```

	P_PROPERTYID	P_ADDRESS	P_TYPE	P_SIZE	P_AMENITIES	P_OWNERSHIPDETAILS
1	1	123 Main St	Apartment	1000	Swimming pool, Gym	Owned by John Doe
2	2	456 Elm St	House	2000	Backyard, Garage	Owned by Jane Smith
3	3	789 Oak St	Condo	1500	Security, Parking	Owned by Michael Johnson
4	4	101 Pine St	Townhouse	1800	Garden, Community Center	Owned by Emily Davis
5	5	222 Maple St	Duplex	2200	Balcony, Parking	Owned by Robert Brown
6	6	333 Cherry St	Villa	3000	Private pool, Garden	Owned by Sarah Wilson
7	7	444 Walnut St	Studio	800	Laundry, Gym	Owned by Thomas Miller
8	8	123 Main St	Residential	100	Swimming Pool	John Doe

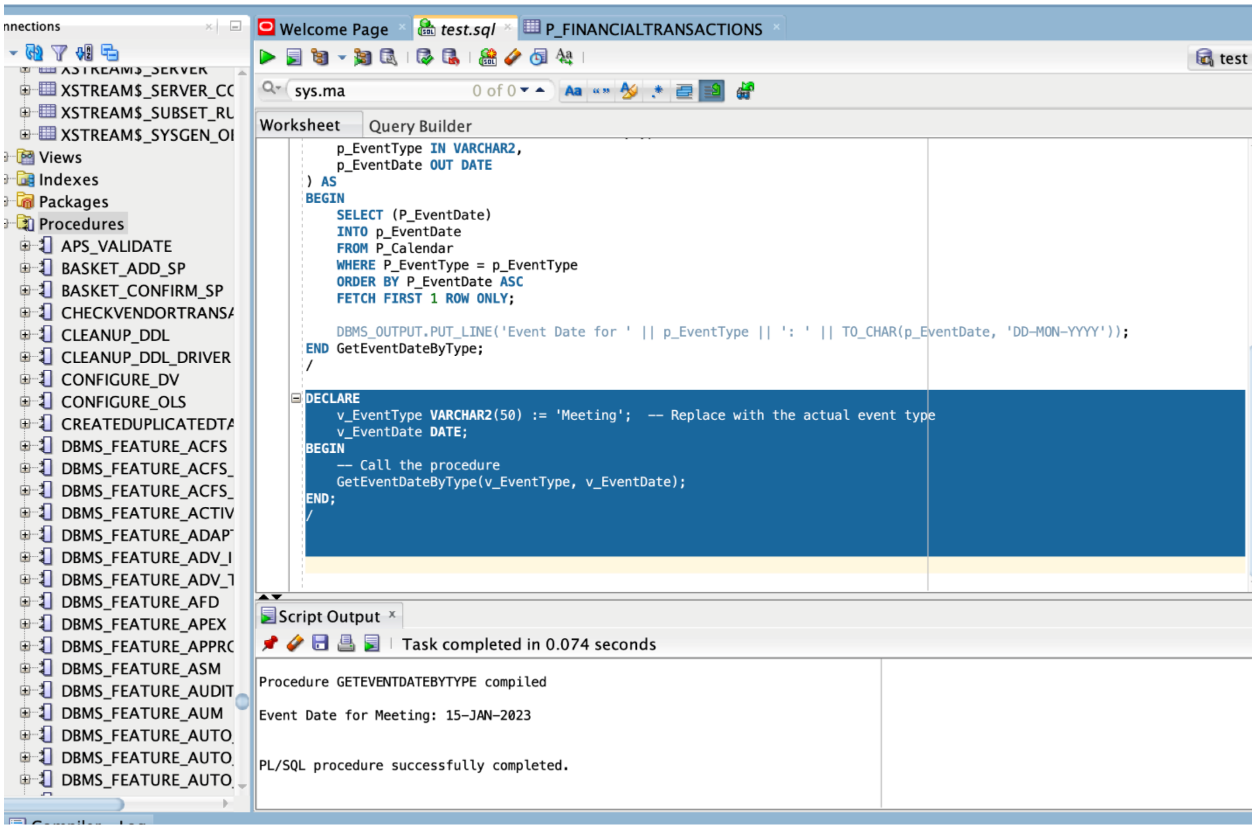
2-Retrieve Event Name by Date

```
CREATE OR REPLACE PROCEDURE Geteventdatebytype(
   P_Eventtype IN VARCHAR2,
   P_Eventdate OUT DATE
) AS
BEGIN
   SELECT (P_Eventdate)
   INTO P_Eventdate
   FROM P_Calendar
   WHERE P_Eventtype = P_Eventtype
   ORDER BY P_Eventdate ASC
   FETCH FIRST 1 ROW ONLY;

   DBMS_OUTPUT.PUT_LINE('Event Date For ' || P_Eventtype || ': ' ||
TO_CHAR(P_Eventdate, 'DD-MON-YYYY'));
END Geteventdatebytype;
```

/

- **TEST**

```
DECLARE
   V_Eventtype VARCHAR2(50) := 'Meeting';  -- Replace With The Actual Event Type
   V_Eventdate DATE;
BEGIN
   -- Call The Procedure
   Geteventdatebytype(V_Eventtype, V_Eventdate);
END;
/
```

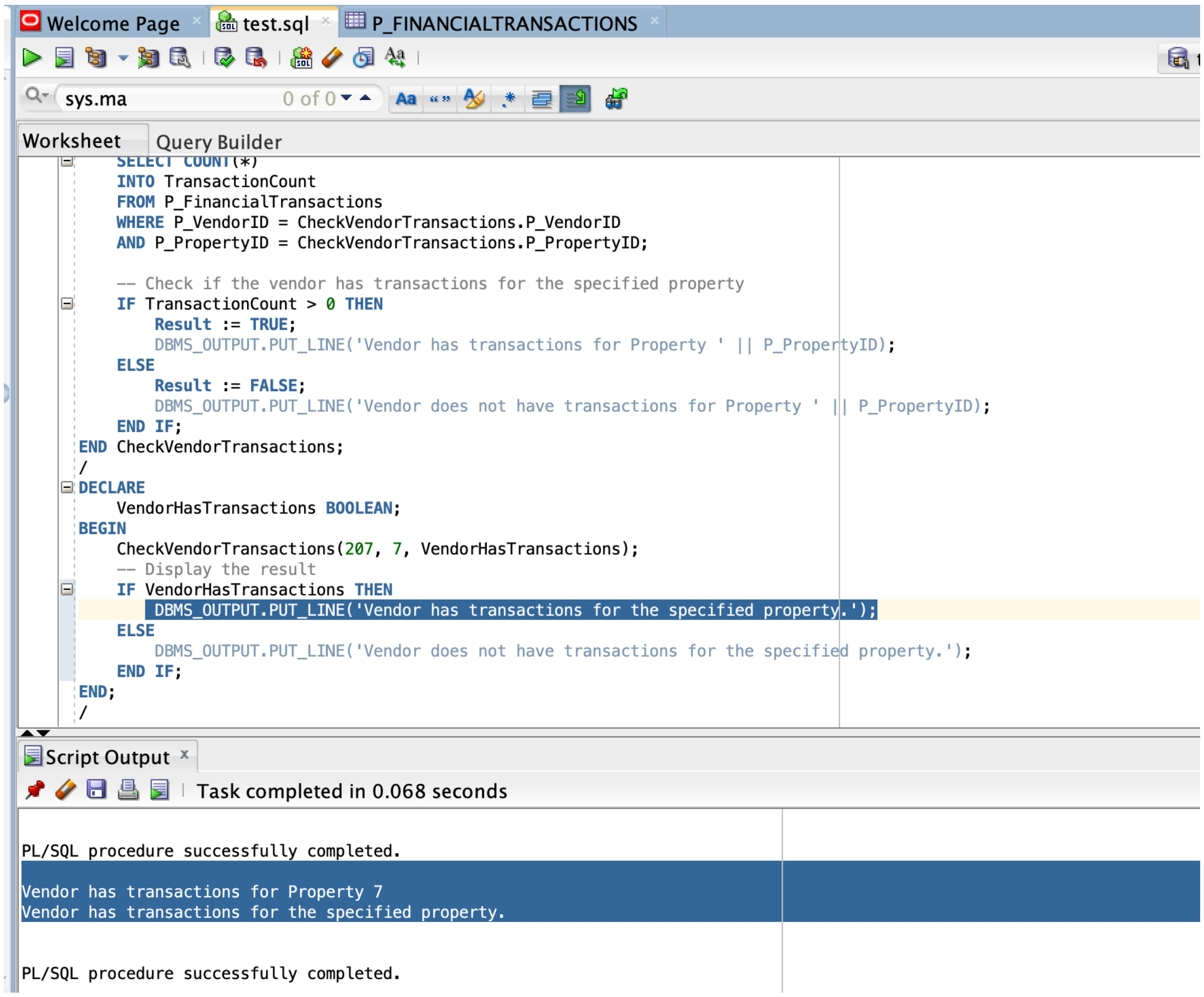


```
    p_EventType IN VARCHAR2,
    p_EventDate OUT DATE
) AS
BEGIN
    SELECT (P_EventDate)
    INTO p_EventDate
    FROM P_Calendar
    WHERE P_EventType = p_EventType
    ORDER BY P_EventDate ASC
    FETCH FIRST 1 ROW ONLY;

    DBMS_OUTPUT.PUT_LINE('Event Date for ' || p_EventType || ': ' || TO_CHAR(p_EventDate, 'DD-MON-YYYY'));
END GetEventDateByType;
/

DECLARE
    v_EventType VARCHAR2(50) := 'Meeting';  -- Replace with the actual event type
    v_EventDate DATE;
BEGIN
    -- Call the procedure
    GetEventDateByType(v_EventType, v_EventDate);
END;
/
```

```
Procedure GETEVENTDATEBYTYPE compiled

Event Date for Meeting: 15-JAN-2023


PL/SQL procedure successfully completed.
```

3-Check a vendor for a property has financial transaction or not

```
CREATE OR REPLACE PROCEDURE Checkvendortransactions (
   P_Vendorid IN NUMBER,
   P_Propertyid IN NUMBER,
   Result OUT BOOLEAN
) AS
   Transactioncount NUMBER;
BEGIN
   -- Get The Count Of Transactions Performed By The Vendor For The Specified
Property
   SELECT COUNT(*)
   INTO Transactioncount
   FROM P_Financialtransactions
   WHERE P_Vendorid = Checkvendortransactions.P_Vendorid
   AND P_Propertyid = Checkvendortransactions.P_Propertyid;

   -- Check If The Vendor Has Transactions For The Specified Property
   IF Transactioncount > 0 THEN
      Result := TRUE;
      DBMS_OUTPUT.PUT_LINE('Vendor Has Transactions For Property ' ||
P_Propertyid);
   ELSE
      Result := FALSE;
      DBMS_OUTPUT.PUT_LINE('Vendor Does Not Have Transactions For Property ' ||
P_Propertyid);
   END IF;
END Checkvendortransactions;
/
DECLARE
   Vendorhastransactions BOOLEAN;
BEGIN
   Checkvendortransactions(207, 7, Vendorhastransactions);
   -- Display The Result
   IF Vendorhastransactions THEN
      DBMS_OUTPUT.PUT_LINE('Vendor Has Transactions For The Specified
Property.');
   ELSE
      DBMS_OUTPUT.PUT_LINE('Vendor Does Not Have Transactions For The
Specified Property.');
   END IF;
END;
/
```



```
    SELECT COUNT(*)
    INTO TransactionCount
    FROM P_FinancialTransactions
    WHERE P_VendorID = CheckVendorTransactions.P_VendorID
    AND P_PropertyID = CheckVendorTransactions.P_PropertyID;

    -- Check if the vendor has transactions for the specified property
    IF TransactionCount > 0 THEN
        Result := TRUE;
        DBMS_OUTPUT.PUT_LINE('Vendor has transactions for Property ' || P_PropertyID);
    ELSE
        Result := FALSE;
        DBMS_OUTPUT.PUT_LINE('Vendor does not have transactions for Property ' || P_PropertyID);
    END IF;
END CheckVendorTransactions;
/
DECLARE
    VendorHasTransactions BOOLEAN;
BEGIN
    CheckVendorTransactions(207, 7, VendorHasTransactions);
    -- Display the result
    IF VendorHasTransactions THEN
        DBMS_OUTPUT.PUT_LINE('Vendor has transactions for the specified property.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Vendor does not have transactions for the specified property.');
    END IF;
END;
/
```

```
PL/SQL procedure successfully completed.

Vendor has transactions for Property 7
Vendor has transactions for the specified property.


PL/SQL procedure successfully completed.
```

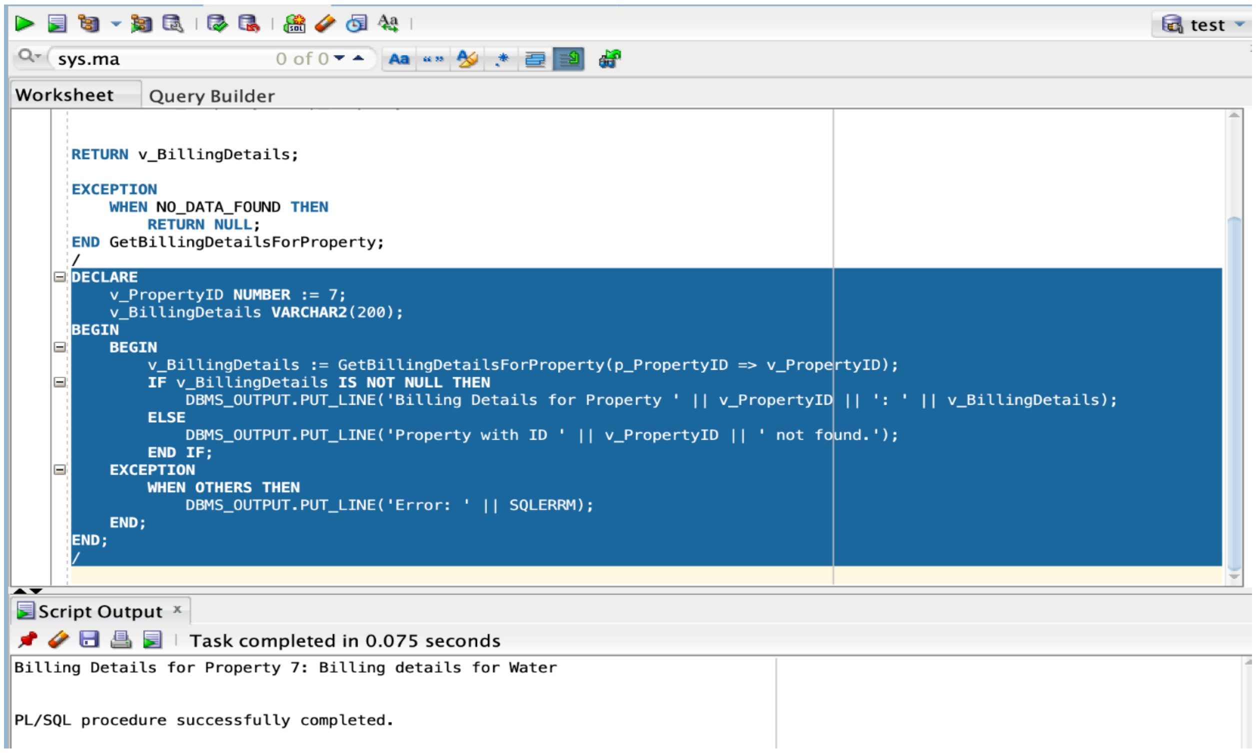
Functions

1- **This function retrieves a list of maintenance requests for a given property, including details about the request, the tenant, and the status.**

```
CREATE OR REPLACE FUNCTION Getbillingdetailsforproperty(
   P_Propertyid IN NUMBER) RETURN VARCHAR2 AS  V_Billingdetails
VARCHAR2(200);
BEGIN
   SELECT Max(P_Billingdetails)
   INTO V_Billingdetails
   FROM P_Utilitymanagement
   WHERE P_Propertyid = P_Propertyid;


RETURN V_Billingdetails;

EXCEPTION
   WHEN NO_DATA_FOUND THEN
      RETURN NULL;
END Getbillingdetailsforproperty;
/
```

- **TEST**

```
DECLARE
   v_PropertyID NUMBER := 7;
   v_BillingDetails VARCHAR2(200);
BEGIN
   BEGIN
      v_BillingDetails := GetBillingDetailsForProperty(p_PropertyID => v_PropertyID);
      IF v_BillingDetails IS NOT NULL THEN
         DBMS_OUTPUT.PUT_LINE('Billing Details for Property ' || v_PropertyID || ': ' ||
v_BillingDetails);
      ELSE
         DBMS_OUTPUT.PUT_LINE('Property with ID ' || v_PropertyID || ' not found.');
      END IF;
   EXCEPTION
      WHEN OTHERS THEN
         DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
   END;
END;
/
```

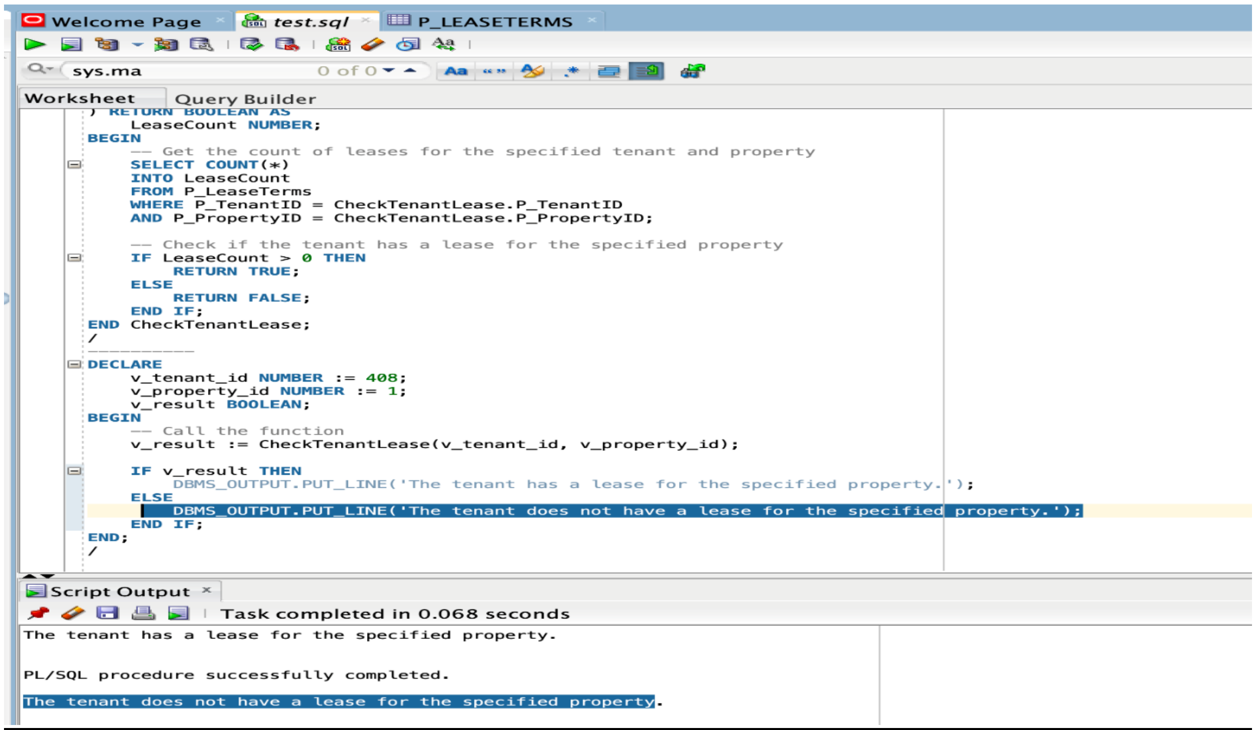


```
RETURN v_BillingDetails;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN NULL;
END GetBillingDetailsForProperty;
/
DECLARE
    v_PropertyID NUMBER := 7;
    v_BillingDetails VARCHAR2(200);
BEGIN
    BEGIN
        v_BillingDetails := GetBillingDetailsForProperty(p_PropertyID => v_PropertyID);
        IF v_BillingDetails IS NOT NULL THEN
            DBMS_OUTPUT.PUT_LINE('Billing Details for Property ' || v_PropertyID || ': ' || v_BillingDetails);
        ELSE
            DBMS_OUTPUT.PUT_LINE('Property with ID ' || v_PropertyID || ' not found.');
        END IF;
    EXCEPTION
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
    END;
END;
/
```

```
Billing Details for Property 7: Billing details for Water


PL/SQL procedure successfully completed.
```

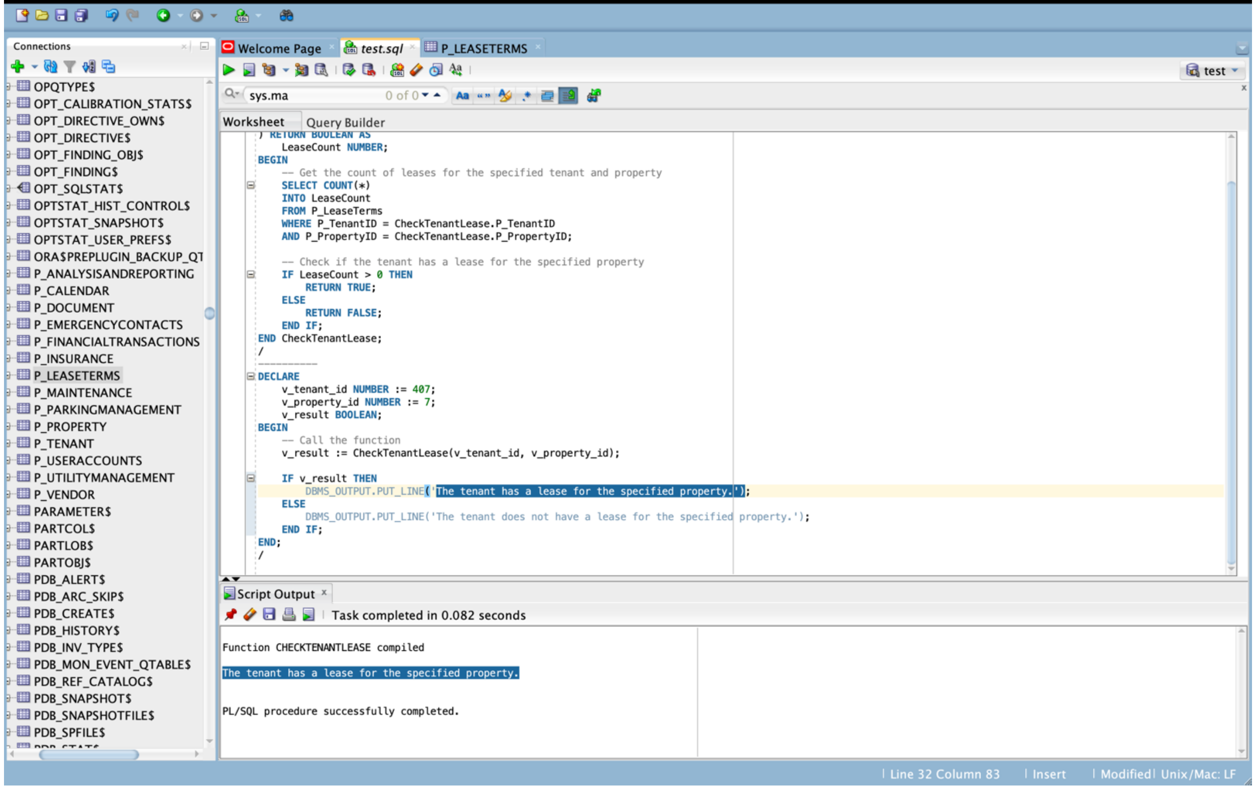
2- Function that checks if a given tenant has a lease for a specified property.

```
CREATE OR REPLACE FUNCTION CheckTenantLease (
   P_TenantID IN NUMBER,
   P_PropertyID IN NUMBER
) RETURN BOOLEAN AS
   LeaseCount NUMBER;
BEGIN
   -- Get the count of leases for the specified tenant and property
   SELECT COUNT(*)
   INTO LeaseCount
   FROM P_LeaseTerms
   WHERE P_TenantID = CheckTenantLease.P_TenantID
   AND P_PropertyID = CheckTenantLease.P_PropertyID;

   -- Check if the tenant has a lease for the specified property
   IF LeaseCount > 0 THEN
      RETURN TRUE;
   ELSE
      RETURN FALSE;
   END IF;
END CheckTenantLease;
```



```
) RETURN BOOLEAN AS
    LeaseCount NUMBER;
BEGIN
    -- Get the count of leases for the specified tenant and property
    SELECT COUNT(*)
    INTO LeaseCount
    FROM P_LeaseTerms
    WHERE P_TenantID = CheckTenantLease.P_TenantID
    AND P_PropertyID = CheckTenantLease.P_PropertyID;

    -- Check if the tenant has a lease for the specified property
    IF LeaseCount > 0 THEN
        RETURN TRUE;
    ELSE
        RETURN FALSE;
    END IF;
END CheckTenantLease;
/
-----------
DECLARE
    v_tenant_id NUMBER := 408;
    v_property_id NUMBER := 1;
    v_result BOOLEAN;
BEGIN
    -- Call the function
    v_result := CheckTenantLease(v_tenant_id, v_property_id);

    IF v_result THEN
        DBMS_OUTPUT.PUT_LINE('The tenant has a lease for the specified property.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('The tenant does not have a lease for the specified property.');
    END IF;
END;
/
```

```
Task completed in 0.068 seconds
The tenant has a lease for the specified property.

PL/SQL procedure successfully completed.

The tenant does not have a lease for the specified property.
```

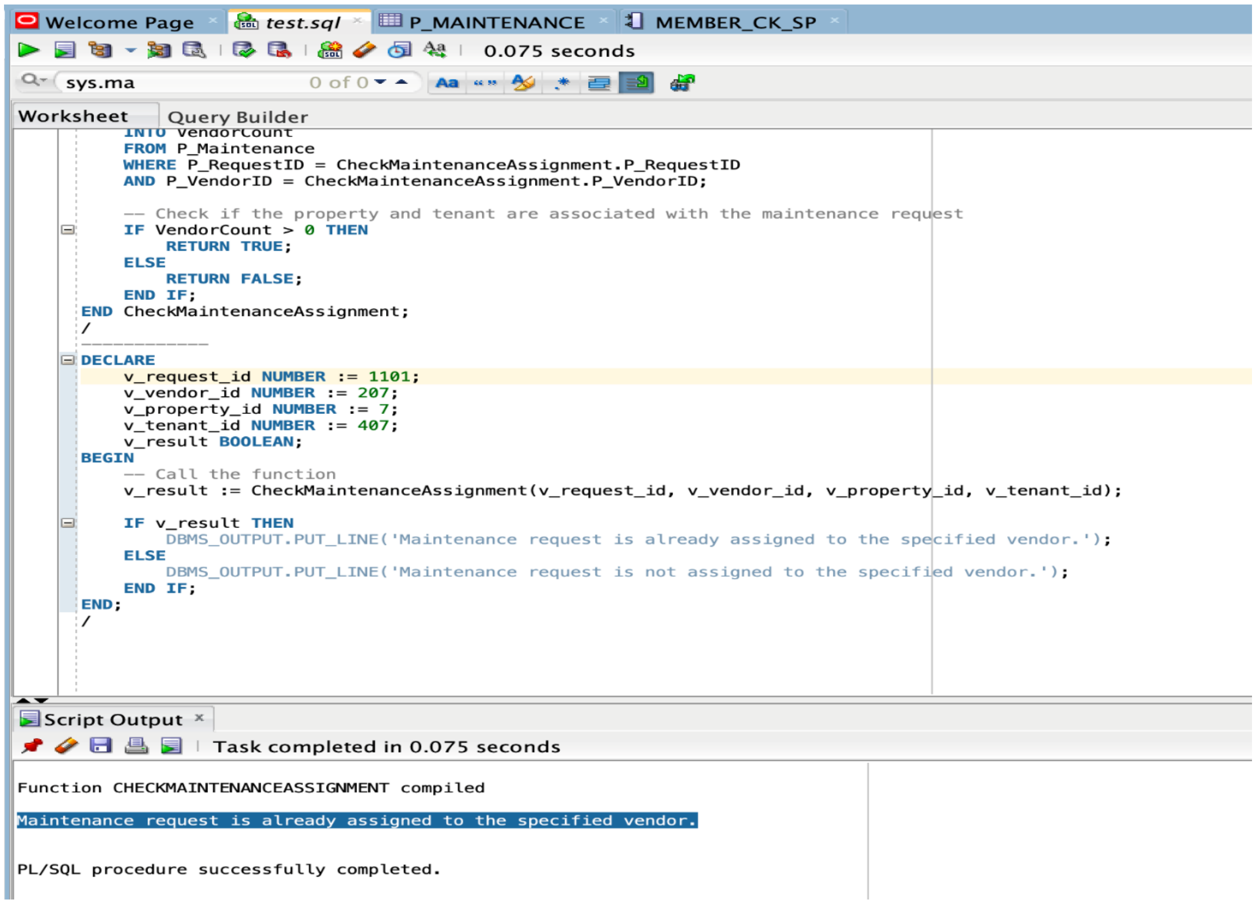


```
) RETURN BOOLEAN AS
    LeaseCount NUMBER;
BEGIN
    -- Get the count of leases for the specified tenant and property
    SELECT COUNT(*)
    INTO LeaseCount
    FROM P_LeaseTerms
    WHERE P_TenantID = CheckTenantLease.P_TenantID
    AND P_PropertyID = CheckTenantLease.P_PropertyID;

    -- Check if the tenant has a lease for the specified property
    IF LeaseCount > 0 THEN
        RETURN TRUE;
    ELSE
        RETURN FALSE;
    END IF;
END CheckTenantLease;
/
----------
DECLARE
    v_tenant_id NUMBER := 407;
    v_property_id NUMBER := 7;
    v_result BOOLEAN;
BEGIN
    -- Call the function
    v_result := CheckTenantLease(v_tenant_id, v_property_id);

    IF v_result THEN
        DBMS_OUTPUT.PUT_LINE('The tenant has a lease for the specified property.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('The tenant does not have a lease for the specified property.');
    END IF;
END;
/
```

```
Task completed in 0.082 seconds

Function CHECKTENANTLEASE compiled

The tenant has a lease for the specified property.


PL/SQL procedure successfully completed.
```

3- Maintenance requests are assigned to specific vendors/service providers. A single vendor can handle multiple maintenance requests, but each request must be assigned to only one vendor. Additionally, a maintenance request must relate to a specific property and tenant.

```
CREATE OR REPLACE FUNCTION CheckMaintenanceAssignment (
   P_RequestID IN NUMBER,
   P_VendorID IN NUMBER,
   P_PropertyID IN NUMBER,
   P_TenantID IN NUMBER
) RETURN BOOLEAN AS
   VendorCount NUMBER;
BEGIN
   -- Check if the vendor is assigned to the specified maintenance request
   SELECT COUNT(*)
   INTO VendorCount
   FROM P_Maintenance
   WHERE P_RequestID = CheckMaintenanceAssignment.P_RequestID
   AND P_VendorID = CheckMaintenanceAssignment.P_VendorID;
   IF VendorCount > 0 THEN
      RETURN TRUE;
   ELSE
      RETURN FALSE;
   END IF;
END CheckMaintenanceAssignment;
/
```



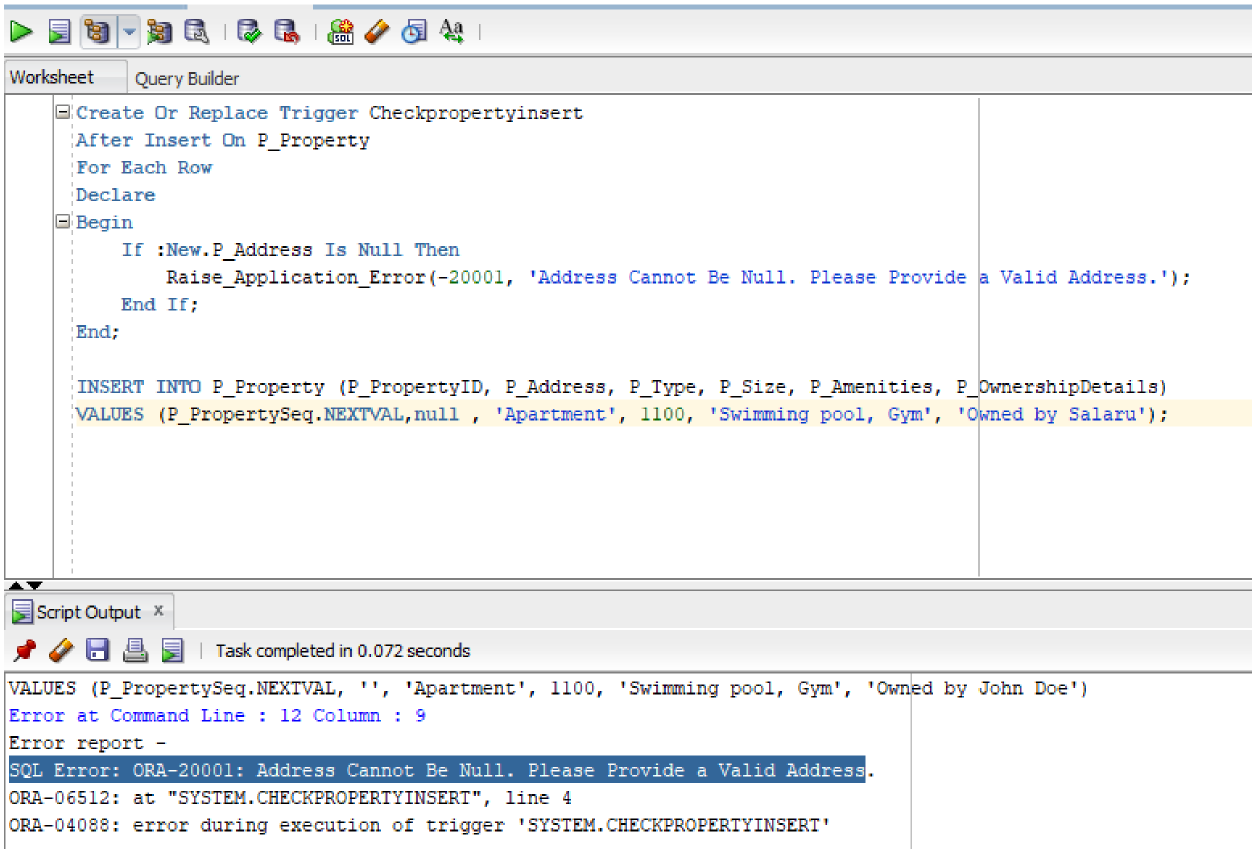
```
    INTO VendorCount
    FROM P_Maintenance
    WHERE P_RequestID = CheckMaintenanceAssignment.P_RequestID
    AND P_VendorID = CheckMaintenanceAssignment.P_VendorID;

    -- Check if the property and tenant are associated with the maintenance request
    IF VendorCount > 0 THEN
        RETURN TRUE;
    ELSE
        RETURN FALSE;
    END IF;
END CheckMaintenanceAssignment;
/
------------
DECLARE
    v_request_id NUMBER := 1101;
    v_vendor_id NUMBER := 207;
    v_property_id NUMBER := 7;
    v_tenant_id NUMBER := 407;
    v_result BOOLEAN;
BEGIN
    -- Call the function
    v_result := CheckMaintenanceAssignment(v_request_id, v_vendor_id, v_property_id, v_tenant_id);

    IF v_result THEN
        DBMS_OUTPUT.PUT_LINE('Maintenance request is already assigned to the specified vendor.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Maintenance request is not assigned to the specified vendor.');
    END IF;
END;
/
```

```
Function CHECKMAINTENANCEASSIGNMENT compiled

Maintenance request is already assigned to the specified vendor.


PL/SQL procedure successfully completed.
```

Triggers:

1-The trigger checks if the inserted property has an address and, if not, raises an exception

```
Create Or Replace Trigger Checkpropertyinsert
After Insert On P_Property
For Each Row
Declare
Begin
   If :New.P_Address Is Null Then
      Raise_Application_Error(-20001, 'Address Cannot Be Null. Please Provide a Valid
Address.');
   End If;
End;
/
```



Worksheet Query Builder

```
Create Or Replace Trigger Checkpropertyinsert
After Insert On P_Property
For Each Row
Declare
Begin
    If :New.P_Address Is Null Then
        Raise_Application_Error(-20001, 'Address Cannot Be Null. Please Provide a Valid Address.');
    End If;
End;

INSERT INTO P_Property (P_PropertyID, P_Address, P_Type, P_Size, P_Amenities, P_OwnershipDetails)
VALUES (P_PropertySeq.NEXTVAL,null , 'Apartment', 1100, 'Swimming pool, Gym', 'Owned by Salaru');
```

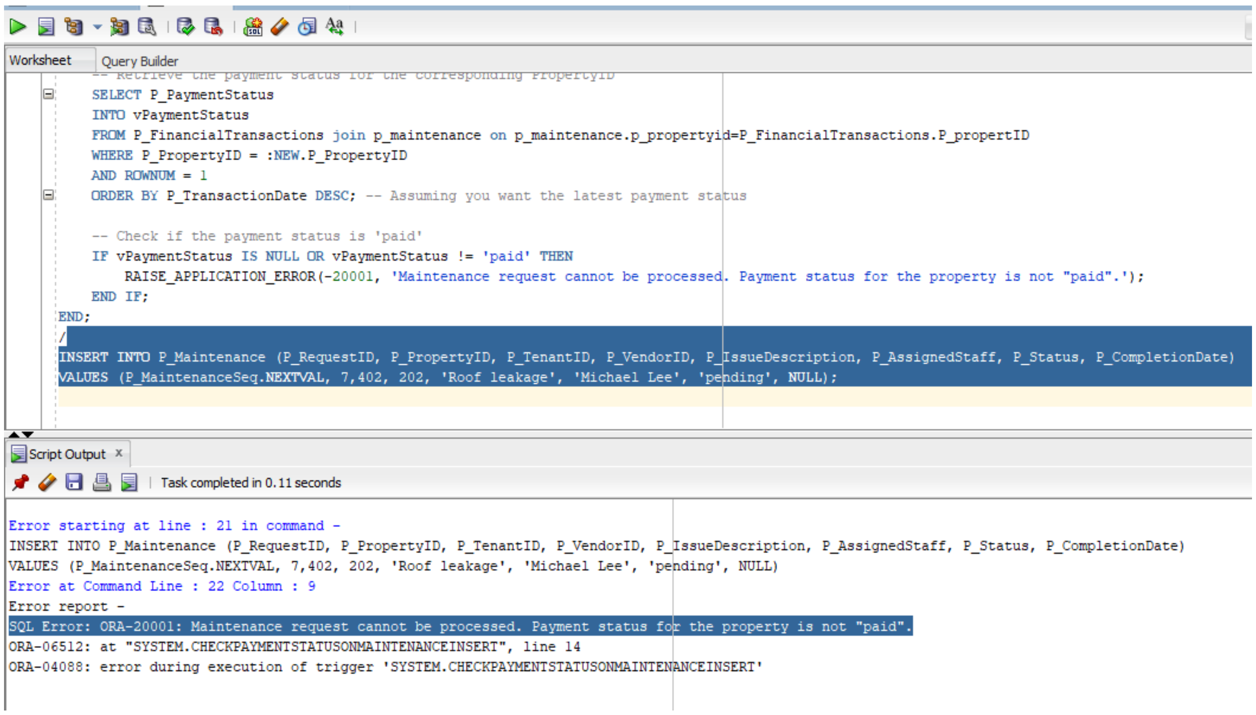
Script Output × Task completed in 0.072 seconds

```
VALUES (P_PropertySeq.NEXTVAL, '', 'Apartment', 1100, 'Swimming pool, Gym', 'Owned by John Doe')
Error at Command Line : 12 Column : 9
Error report -
SQL Error: ORA-20001: Address Cannot Be Null. Please Provide a Valid Address.
ORA-06512: at "SYSTEM.CHECKPROPERTYINSERT", line 4
ORA-04088: error during execution of trigger 'SYSTEM.CHECKPROPERTYINSERT'
```

2- This trigger checks if the P_PaymentStatus is set to 'Paid' and, if so, inserts a corresponding record into the P_Maintenance table:

```
CREATE OR REPLACE TRIGGER CheckPaymentStatusOnMaintenanceInsert
AFTER INSERT ON P_Maintenance
FOR EACH ROW
```

```
DECLARE
   vPaymentStatus VARCHAR2(50);
BEGIN
   -- Retrieve the payment status for the corresponding PropertyID
   SELECT P_PaymentStatus
   INTO vPaymentStatus
   FROM P_FinancialTransactions join p_maintenance on
p_maintenance.p_propertyid=P_FinancialTransactions.P_propertID
   WHERE P_PropertyID = :NEW.P_PropertyID
   AND ROWNUM = 1
   ORDER BY P_TransactionDate DESC; -- Assuming you want the latest payment
status

   -- Check if the payment status is 'paid'
   IF vPaymentStatus IS NULL OR vPaymentStatus != 'paid' THEN
      RAISE_APPLICATION_ERROR(-20001, 'Maintenance request cannot be
processed. Payment status for the property is not "paid".');
   END IF;
END;
/
```



```
    -- Retrieve the payment status for the corresponding PropertyID
    SELECT P_PaymentStatus
    INTO vPaymentStatus
    FROM P_FinancialTransactions join p_maintenance on p_maintenance.p_propertyid=P_FinancialTransactions.P_propertID
    WHERE P_PropertyID = :NEW.P_PropertyID
    AND ROWNUM = 1
    ORDER BY P_TransactionDate DESC; -- Assuming you want the latest payment status

    -- Check if the payment status is 'paid'
    IF vPaymentStatus IS NULL OR vPaymentStatus != 'paid' THEN
        RAISE_APPLICATION_ERROR(-20001, 'Maintenance request cannot be processed. Payment status for the property is not "paid".');
    END IF;
END;
/
INSERT INTO P_Maintenance (P_RequestID, P_PropertyID, P_TenantID, P_VendorID, P_IssueDescription, P_AssignedStaff, P_Status, P_CompletionDate)
VALUES (P_MaintenanceSeq.NEXTVAL, 7,402, 202, 'Roof leakage', 'Michael Lee', 'pending', NULL);
```

Script Output

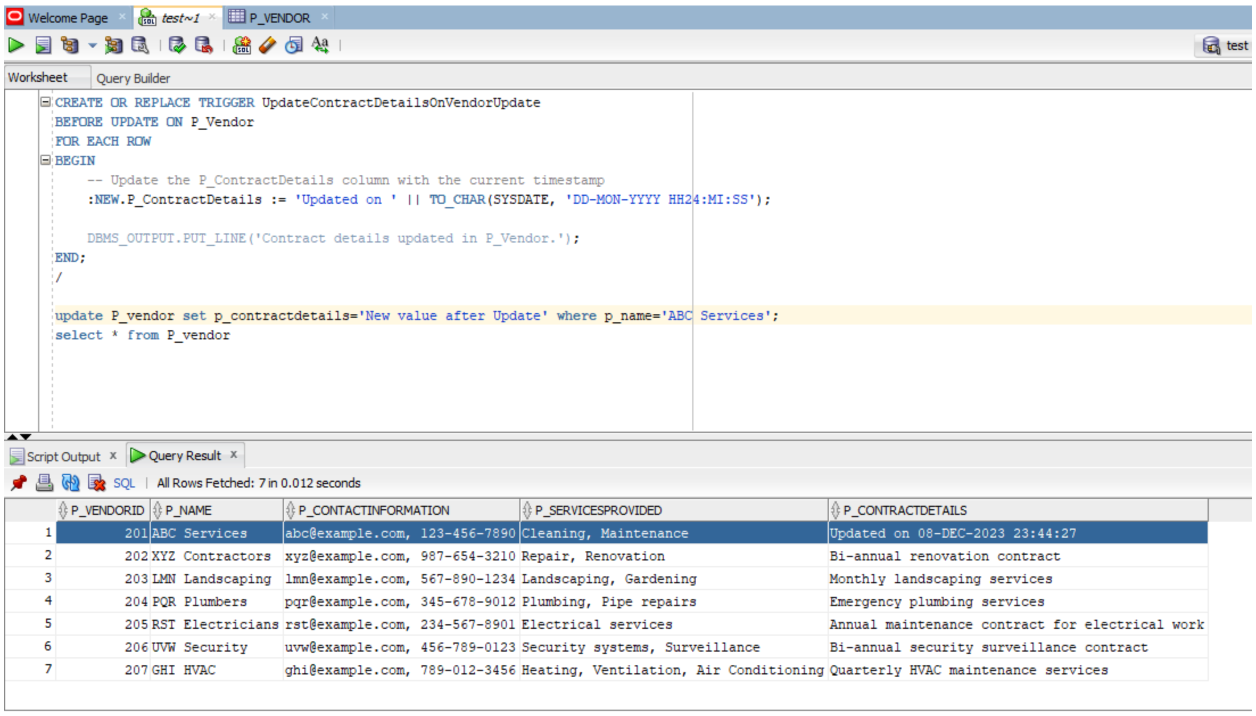
Task completed in 0.11 seconds

```
Error starting at line : 21 in command -
INSERT INTO P_Maintenance (P_RequestID, P_PropertyID, P_TenantID, P_VendorID, P_IssueDescription, P_AssignedStaff, P_Status, P_CompletionDate)
VALUES (P_MaintenanceSeq.NEXTVAL, 7,402, 202, 'Roof leakage', 'Michael Lee', 'pending', NULL)
Error at Command Line : 22 Column : 9
Error report -
SQL Error: ORA-20001: Maintenance request cannot be processed. Payment status for the property is not "paid".
ORA-06512: at "SYSTEM.CHECKPAYMENTSTATUSONMAINTENANCEINSERT", line 14
ORA-04088: error during execution of trigger 'SYSTEM.CHECKPAYMENTSTATUSONMAINTENANCEINSERT'
```

3- Update the P_ContractDetails column with the current timestamp

```
CREATE OR REPLACE TRIGGER
UPDATECONTRACTDETAILSONVENDORUPDATE
```

```
BEFORE UPDATE ON P_VENDOR
FOR EACH ROW
BEGIN
   :NEW.P_CONTRACTDETAILS := 'UPDATED ON ' || TO_CHAR(SYSDATE, 'DD-
MON-YYYY HH24:MI:SS');
   DBMS_OUTPUT.PUT_LINE('CONTRACT DETAILS UPDATED IN P_VENDOR.');
END;
/
```



```
CREATE OR REPLACE TRIGGER UpdateContractDetailsOnVendorUpdate
BEFORE UPDATE ON P_Vendor
FOR EACH ROW
BEGIN
    -- Update the P_ContractDetails column with the current timestamp
    :NEW.P_ContractDetails := 'Updated on ' || TO_CHAR(SYSDATE, 'DD-MON-YYYY HH24:MI:SS');

    DBMS_OUTPUT.PUT_LINE('Contract details updated in P_Vendor.');
END;
/

update P_vendor set p_contractdetails='New value after Update' where p_name='ABC Services';
select * from P_vendor
```

All Rows Fetched: 7 in 0.012 seconds

	P_VENDORID	P_NAME	P_CONTACTINFORMATION	P_SERVICESPROVIDED	P_CONTRACTDETAILS
1	201	ABC Services	abc@example.com, 123-456-7890	Cleaning, Maintenance	Updated on 08-DEC-2023 23:44:27
2	202	XYZ Contractors	xyz@example.com, 987-654-3210	Repair, Renovation	Bi-annual renovation contract
3	203	LMN Landscaping	lmn@example.com, 567-890-1234	Landscaping, Gardening	Monthly landscaping services
4	204	PQR Plumbers	pqr@example.com, 345-678-9012	Plumbing, Pipe repairs	Emergency plumbing services
5	205	RST Electricians	rst@example.com, 234-567-8901	Electrical services	Annual maintenance contract for electrical work
6	206	UVW Security	uvw@example.com, 456-789-0123	Security systems, Surveillance	Bi-annual security surveillance contract
7	207	GHI HVAC	ghi@example.com, 789-012-3456	Heating, Ventilation, Air Conditioning	Quarterly HVAC maintenance services

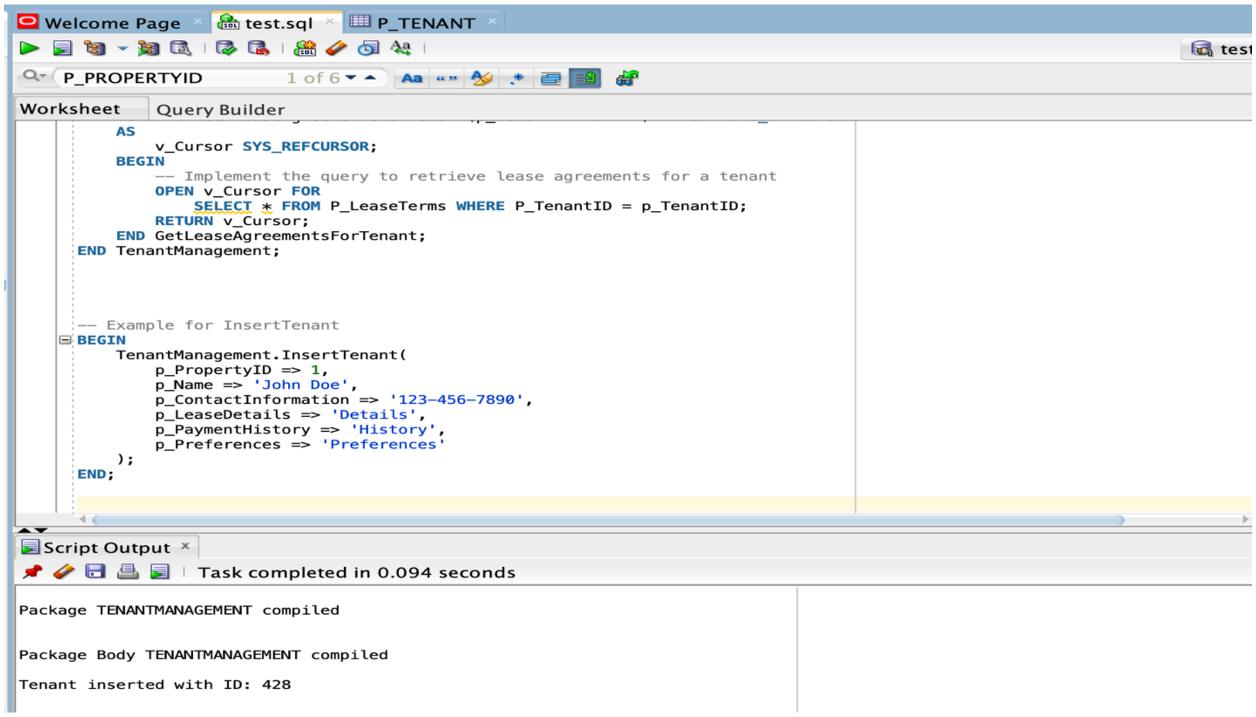
Packages:

1-Each tenant can have multiple lease agreements, but each lease agreement is associated with only one tenant. Additionally, a lease agreement must specify a property, its terms (start date, end date), and rental amount.

```
CREATE OR REPLACE PACKAGE Tenantmanagement AS
   -- Procedure To Insert A New Tenant
   PROCEDURE Inserttenant(
      P_Propertyid NUMBER,
      P_Name VARCHAR2,
      P_Contactinformation VARCHAR2,
      P_Leasedetails VARCHAR2,
      P_Paymenthistory VARCHAR2,
      P_Preferences VARCHAR2
   );
```

```
    -- Procedure To Insert A New Lease Agreement
    PROCEDURE Insertleaseagreement(
        P_Tenantid NUMBER,
        P_Propertyid NUMBER,
        P_Startdate DATE,
        P_Enddate DATE,
        P_Rentamount NUMBER,
        P_Paymentschedule VARCHAR2,
        P_Deposit NUMBER
    );

    -- Function To Retrieve Lease Agreements For A Tenant
    FUNCTION Getleaseagreementsfortenant(P_Tenantid NUMBER) RETURN
SYS_REFCURSOR;
END Tenantmanagement;
/

CREATE OR REPLACE PACKAGE BODY Tenantmanagement AS
    -- Procedure To Insert A New Tenant
    PROCEDURE Inserttenant(
        P_Propertyid NUMBER,
        P_Name VARCHAR2,
        P_Contactinformation VARCHAR2,
        P_Leasedetails VARCHAR2,
        P_Paymenthistory VARCHAR2,
        P_Preferences VARCHAR2
    )
    AS
        V_Tenantid NUMBER;
    BEGIN
        -- Insert Tenant Into P_Tenant Table
        INSERT INTO P_Tenant (P_Tenantid, P_Propertyid, P_Name,
P_Contactinformation, P_Leasedetails, P_Paymenthistory, P_Preferences)
        VALUES (P_Tenantseq.NEXTVAL, P_Propertyid, P_Name, P_Contactinformation,
P_Leasedetails, P_Paymenthistory, P_Preferences)
        RETURNING P_Tenantid INTO V_Tenantid;

        -- Print A Message Or Handle Any Additional Logic If Needed
        DBMS_OUTPUT.PUT_LINE('Tenant Inserted With ID: ' || V_Tenantid);
    END Inserttenant;

    -- Procedure To Insert A New Lease Agreement
```

```
   PROCEDURE Insertleaseagreement(
      P_Tenantid NUMBER,
      P_Propertyid NUMBER,
      P_Startdate DATE,
      P_Enddate DATE,
      P_Rentamount NUMBER,
      P_Paymentschedule VARCHAR2,
      P_Deposit NUMBER
   )
   AS
   BEGIN
      -- Insert Lease Agreement Into P_Leaseterms Table
      INSERT INTO P_Leaseterms (P_Leasetermid, P_Tenantid, P_Propertyid,
P_Startdate, P_Enddate, P_Rentamount, P_Paymentschedule, P_Deposit)
      VALUES (P_Leasetermsseq.NEXTVAL, P_Tenantid, P_Propertyid, P_Startdate,
P_Enddate, P_Rentamount, P_Paymentschedule, P_Deposit);

      -- Print A Message Or Handle Any Additional Logic If Needed
      DBMS_OUTPUT.PUT_LINE('Lease Agreement Inserted For Tenant ID: ' ||
P_Tenantid);
   END Insertleaseagreement;

   -- Function To Retrieve Lease Agreements For A Tenant
   FUNCTION Getleaseagreementsfortenant(P_Tenantid NUMBER) RETURN
SYS_REFCURSOR
   AS
      V_Cursor SYS_REFCURSOR;
   BEGIN
      -- Implement The Query To Retrieve Lease Agreements For A Tenant
      OPEN V_Cursor FOR
         SELECT * FROM P_Leaseterms WHERE P_Tenantid = P_Tenantid;
      RETURN V_Cursor;
   END Getleaseagreementsfortenant;
END Tenantmanagement;
/
-- Example For Inserttenant
BEGIN
   Tenantmanagement.Inserttenant(
      P_Propertyid => 1,
      P_Name => 'John Doe',
      P_Contactinformation => '123-456-7890',
      P_Leasedetails => 'Details',
      P_Paymenthistory => 'History',
        P_Preferences => 'Preferences'
    );
END;
```



```
    AS
        v_Cursor SYS_REFCURSOR;
    BEGIN
        -- Implement the query to retrieve lease agreements for a tenant
        OPEN v_Cursor FOR
            SELECT * FROM P_LeaseTerms WHERE P_TenantID = p_TenantID;
        RETURN v_Cursor;
    END GetLeaseAgreementsForTenant;
END TenantManagement;



-- Example for InsertTenant
BEGIN
    TenantManagement.InsertTenant(
        p_PropertyID => 1,
        p_Name => 'John Doe',
        p_ContactInformation => '123-456-7890',
        p_LeaseDetails => 'Details',
        p_PaymentHistory => 'History',
        p_Preferences => 'Preferences'
    );
END;
```

```
Package TENANTMANAGEMENT compiled


Package Body TENANTMANAGEMENT compiled

Tenant inserted with ID: 428
```

2- This package is base on the procedure for inserting in some tables. It contains Exception and RowType.

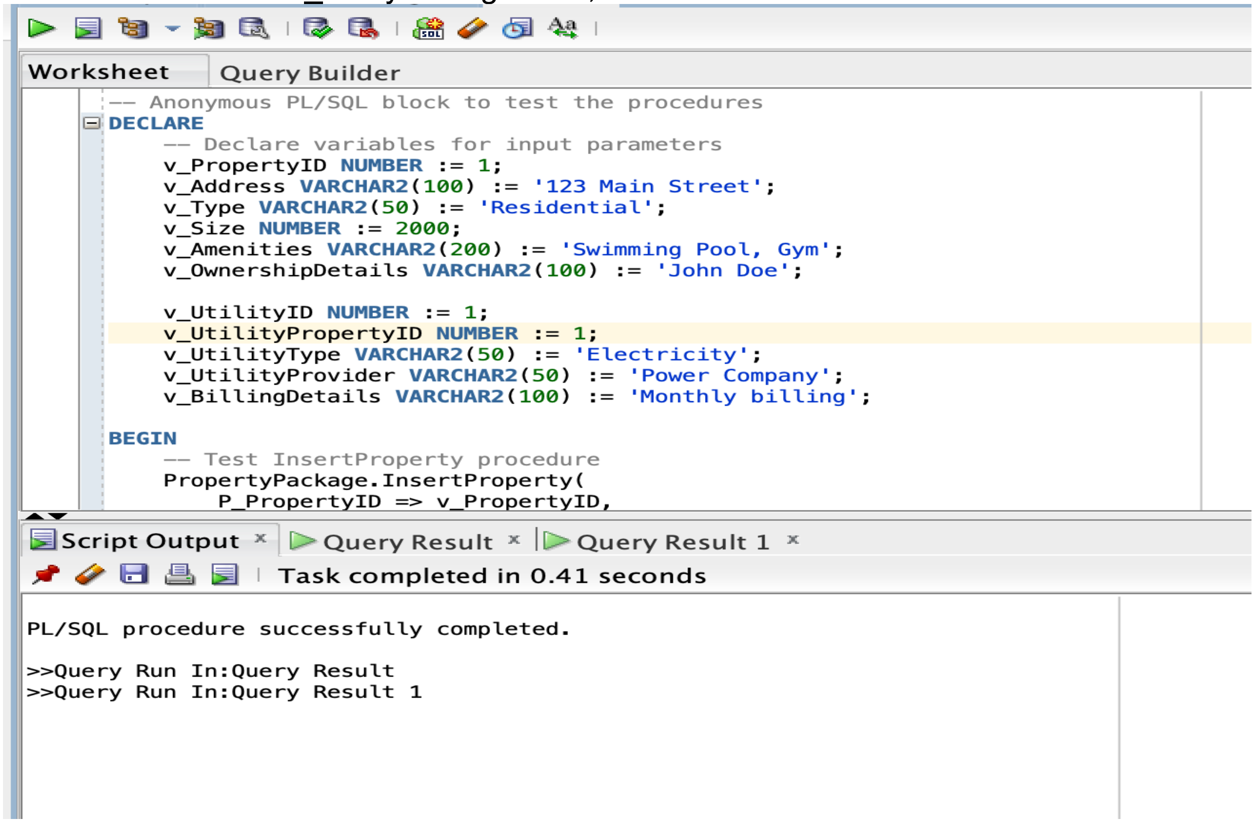
```
-- Creating the Package Specification
CREATE OR REPLACE PACKAGE PropertyPackage AS
    PROCEDURE InsertProperty(
        P_PropertyID IN NUMBER,
        P_Address IN VARCHAR2,
        P_Type IN VARCHAR2,
        P_Size IN NUMBER,
        P_Amenities IN VARCHAR2,
        P_OwnershipDetails IN VARCHAR2
    );

    PROCEDURE InsertUtilityManagement(
        P_UtilityID IN NUMBER,
        P_PropertyID IN NUMBER,
        P_UtilityType IN VARCHAR2,
        P_Provider IN VARCHAR2,
        P_BillingDetails IN VARCHAR2
    );
```

```
END PropertyPackage;
/

-- Creating the Package Body
CREATE OR REPLACE PACKAGE BODY PropertyPackage AS
   PROCEDURE InsertProperty(
      P_PropertyID IN NUMBER,
      P_Address IN VARCHAR2,
      P_Type IN VARCHAR2,
      P_Size IN NUMBER,
      P_Amenities IN VARCHAR2,
      P_OwnershipDetails IN VARCHAR2
   ) AS
      v_property P_Property%ROWTYPE;
   BEGIN
      -- Initialize the rowtype variable
      v_property.P_PropertyID := P_PropertyID;
      v_property.P_Address := P_Address;
      v_property.P_Type := P_Type;
      v_property.P_Size := P_Size;
      v_property.P_Amenities := P_Amenities;
      v_property.P_OwnershipDetails := P_OwnershipDetails;

      -- Insert data into P_Property table
      INSERT INTO P_Property VALUES v_property;
   EXCEPTION
      WHEN OTHERS THEN
         DBMS_OUTPUT.PUT_LINE('Error inserting data into P_Property: ' ||
SQLERRM);
   END InsertProperty;

   PROCEDURE InsertUtilityManagement(
      P_UtilityID IN NUMBER,
      P_PropertyID IN NUMBER,
      P_UtilityType IN VARCHAR2,
      P_Provider IN VARCHAR2,
      P_BillingDetails IN VARCHAR2
   ) AS
      v_utility P_UtilityManagement%ROWTYPE;
   BEGIN
      -- Initialize the rowtype variable
      v_utility.P_UtilityID := P_UtilityID;
      v_utility.P_PropertyID := P_PropertyID;
      v_utility.P_UtilityType := P_UtilityType;
      v_utility.P_Provider := P_Provider;
      v_utility.P_BillingDetails := P_BillingDetails;
      -- Insert data into P_UtilityManagement table
      INSERT INTO P_UtilityManagement VALUES v_utility;
   EXCEPTION
      WHEN OTHERS THEN
         DBMS_OUTPUT.PUT_LINE('Error inserting data into P_UtilityManagement: ' ||
SQLERRM);
   END InsertUtilityManagement;
END PropertyPackage;
/
```

➢ Test

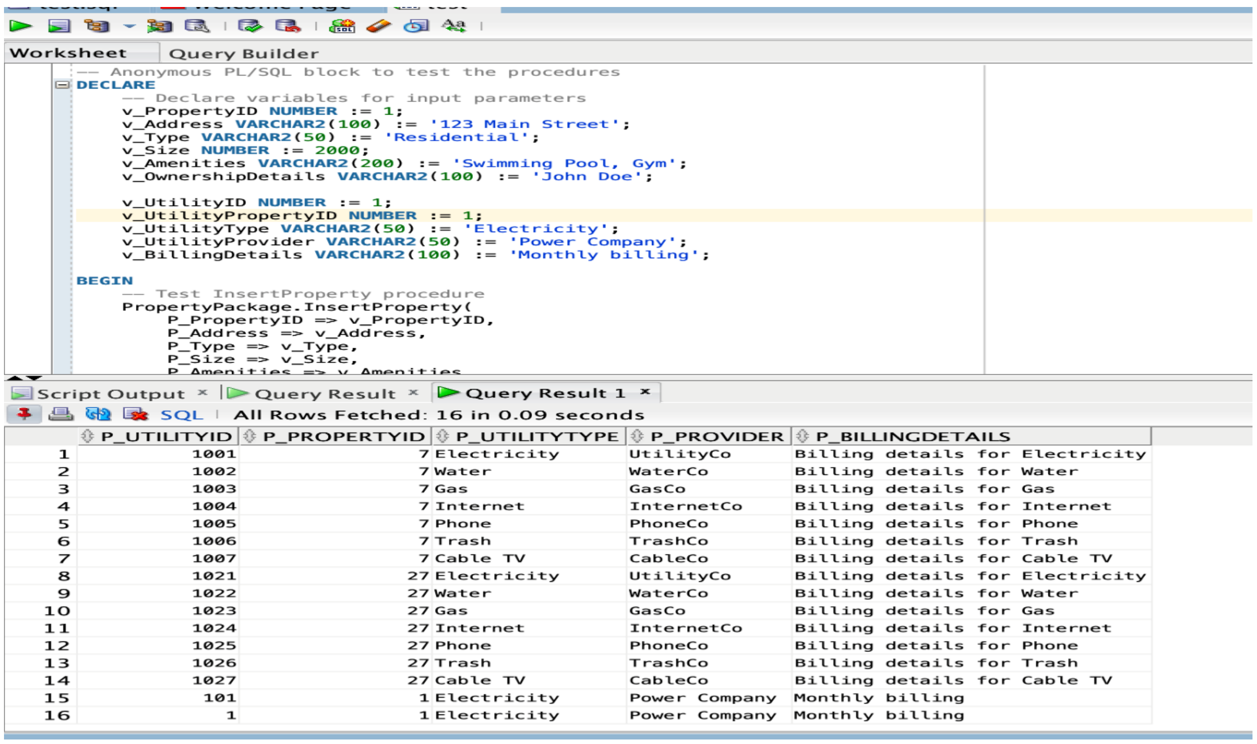
```
Declare
   -- Declare Variables For Input Parameters
   V_Propertyid Number := 1;
   V_Address Varchar2(100) := '123 Main Street';
   V_Type Varchar2(50) := 'Residential';
   V_Size Number := 2000;
   V_Amenities Varchar2(200) := 'Swimming Pool, Gym';
   V_Ownershipdetails Varchar2(100) := 'John Doe';

   V_Utilityid Number := 1;
   V_Utilitypropertyid Number := 1;
   V_Utilitytype Varchar2(50) := 'Electricity';
   V_Utilityprovider Varchar2(50) := 'Power Company';
   V_Billingdetails Varchar2(100) := 'Monthly Billing';

Begin
   -- Test Insertproperty Procedure
   Propertypackage.Insertproperty(
      P_Propertyid => V_Propertyid,
      P_Address => V_Address,
      P_Type => V_Type,
      P_Size => V_Size,
      P_Amenities => V_Amenities,
      P_Ownershipdetails => V_Ownershipdetails
   );

   -- Test Insertutilitymanagement Procedure
   Propertypackage.Insertutilitymanagement(
      P_Utilityid => V_Utilityid,
      P_Propertyid => V_Utilitypropertyid,
      P_Utilitytype => V_Utilitytype,
      P_Provider => V_Utilityprovider,
      P_Billingdetails => V_Billingdetails
```

```
    );

    -- Commit The Transaction
    Commit;
End;
/
-- Check The Data In The Tables After Testing
Select * From P_Property;
Select * From P_Utilitymanagement;
```



```
-- Anonymous PL/SQL block to test the procedures
DECLARE
    -- Declare variables for input parameters
    v_PropertyID NUMBER := 1;
    v_Address VARCHAR2(100) := '123 Main Street';
    v_Type VARCHAR2(50) := 'Residential';
    v_Size NUMBER := 2000;
    v_Amenities VARCHAR2(200) := 'Swimming Pool, Gym';
    v_OwnershipDetails VARCHAR2(100) := 'John Doe';

    v_UtilityID NUMBER := 1;
    v_UtilityPropertyID NUMBER := 1;
    v_UtilityType VARCHAR2(50) := 'Electricity';
    v_UtilityProvider VARCHAR2(50) := 'Power Company';
    v_BillingDetails VARCHAR2(100) := 'Monthly billing';

BEGIN
    -- Test InsertProperty procedure
    PropertyPackage.InsertProperty(
        P_PropertyID => v_PropertyID,
```

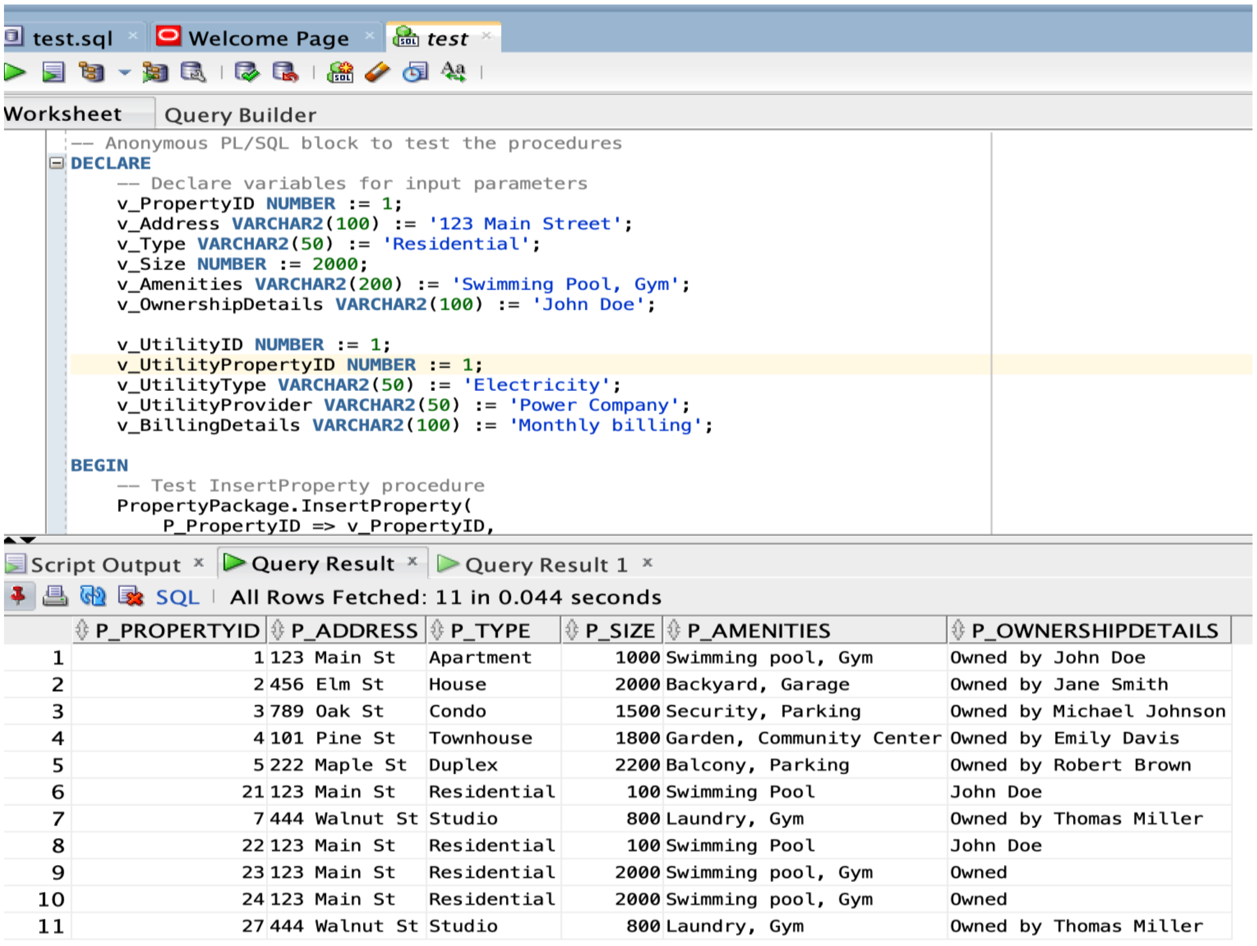
```
PL/SQL procedure successfully completed.

>>Query Run In:Query Result
>>Query Run In:Query Result 1
```



```
-- Anonymous PL/SQL block to test the procedures
DECLARE
    -- Declare variables for input parameters
    v_PropertyID NUMBER := 1;
    v_Address VARCHAR2(100) := '123 Main Street';
    v_Type VARCHAR2(50) := 'Residential';
    v_Size NUMBER := 2000;
    v_Amenities VARCHAR2(200) := 'Swimming Pool, Gym';
    v_OwnershipDetails VARCHAR2(100) := 'John Doe';

    v_UtilityID NUMBER := 1;
    v_UtilityPropertyID NUMBER := 1;
    v_UtilityType VARCHAR2(50) := 'Electricity';
    v_UtilityProvider VARCHAR2(50) := 'Power Company';
    v_BillingDetails VARCHAR2(100) := 'Monthly billing';

BEGIN
    -- Test InsertProperty procedure
    PropertyPackage.InsertProperty(
        P_PropertyID => v_PropertyID,
        P_Address => v_Address,
        P_Type => v_Type,
        P_Size => v_Size,
        P_Amenities => v_Amenities
```

All Rows Fetched: 16 in 0.09 seconds

	P_UTILITYID	P_PROPERTYID	P_UTILITYTYPE	P_PROVIDER	P_BILLINGDETAILS
1	1001	7	Electricity	UtilityCo	Billing details for Electricity
2	1002	7	Water	WaterCo	Billing details for Water
3	1003	7	Gas	GasCo	Billing details for Gas
4	1004	7	Internet	InternetCo	Billing details for Internet
5	1005	7	Phone	PhoneCo	Billing details for Phone
6	1006	7	Trash	TrashCo	Billing details for Trash
7	1007	7	Cable TV	CableCo	Billing details for Cable TV
8	1021	27	Electricity	UtilityCo	Billing details for Electricity
9	1022	27	Water	WaterCo	Billing details for Water
10	1023	27	Gas	GasCo	Billing details for Gas
11	1024	27	Internet	InternetCo	Billing details for Internet
12	1025	27	Phone	PhoneCo	Billing details for Phone
13	1026	27	Trash	TrashCo	Billing details for Trash
14	1027	27	Cable TV	CableCo	Billing details for Cable TV
15	101	1	Electricity	Power Company	Monthly billing
16	1	1	Electricity	Power Company	Monthly billing


test.sql | Welcome Page | test

Worksheet | Query Builder

```
-- Anonymous PL/SQL block to test the procedures
DECLARE
    -- Declare variables for input parameters
    v_PropertyID NUMBER := 1;
    v_Address VARCHAR2(100) := '123 Main Street';
    v_Type VARCHAR2(50) := 'Residential';
    v_Size NUMBER := 2000;
    v_Amenities VARCHAR2(200) := 'Swimming Pool, Gym';
    v_OwnershipDetails VARCHAR2(100) := 'John Doe';

    v_UtilityID NUMBER := 1;
    v_UtilityPropertyID NUMBER := 1;
    v_UtilityType VARCHAR2(50) := 'Electricity';
    v_UtilityProvider VARCHAR2(50) := 'Power Company';
    v_BillingDetails VARCHAR2(100) := 'Monthly billing';

BEGIN
    -- Test InsertProperty procedure
    PropertyPackage.InsertProperty(
        P_PropertyID => v_PropertyID,
```

Script Output | Query Result | Query Result 1

SQL | All Rows Fetched: 11 in 0.044 seconds

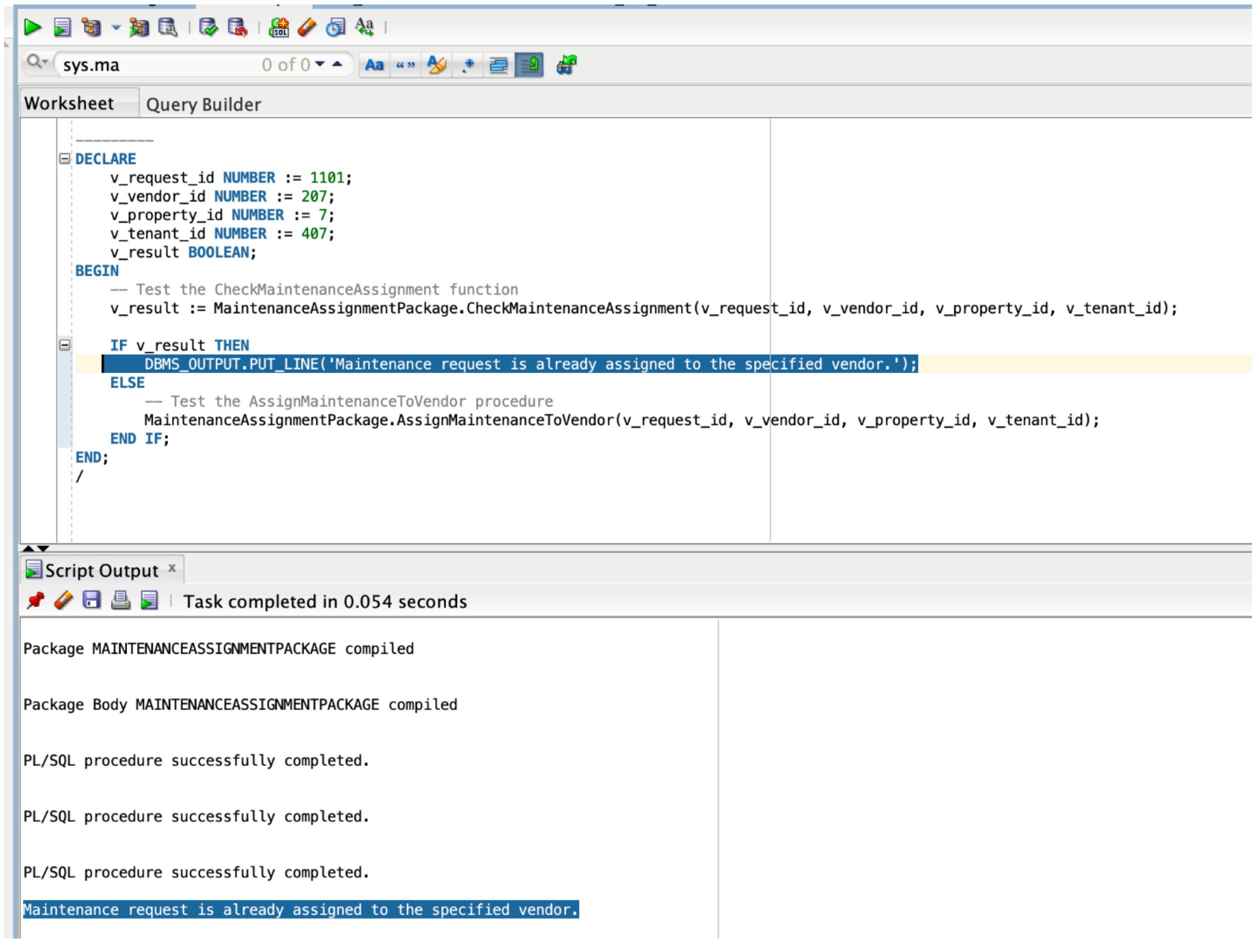
	P_PROPERTYID	P_ADDRESS	P_TYPE	P_SIZE	P_AMENITIES	P_OWNERSHIPDETAILS
1	1	123 Main St	Apartment	1000	Swimming pool, Gym	Owned by John Doe
2	2	456 Elm St	House	2000	Backyard, Garage	Owned by Jane Smith
3	3	789 Oak St	Condo	1500	Security, Parking	Owned by Michael Johnson
4	4	101 Pine St	Townhouse	1800	Garden, Community Center	Owned by Emily Davis
5	5	222 Maple St	Duplex	2200	Balcony, Parking	Owned by Robert Brown
6	21	123 Main St	Residential	100	Swimming Pool	John Doe
7	7	444 Walnut St	Studio	800	Laundry, Gym	Owned by Thomas Miller
8	22	123 Main St	Residential	100	Swimming Pool	John Doe
9	23	123 Main St	Residential	2000	Swimming pool, Gym	Owned
10	24	123 Main St	Residential	2000	Swimming pool, Gym	Owned
11	27	444 Walnut St	Studio	800	Laundry, Gym	Owned by Thomas Miller

2- based on the business rule "Maintenance requests are assigned to specific vendors/service providers. A single vendor can handle multiple maintenance requests, but each request must be assigned to only one vendor. Additionally, a maintenance request must relate to a specific property and tenant" that we wrote a function and a procedure, now we put both of them in a package to show package functionality

```
CREATE OR REPLACE PACKAGE Maintenanceassignmentpackage AS

  FUNCTION Checkmaintenanceassignment (
    P_Requestid IN NUMBER,
    P_Vendorid IN NUMBER,
    P_Propertyid IN NUMBER,
    P_Tenantid IN NUMBER
  ) RETURN BOOLEAN;
```

```
   PROCEDURE Assignmaintenancetovendor (
      P_Requestid IN NUMBER,
      P_Vendorid IN NUMBER,
      P_Propertyid IN NUMBER,
      P_Tenantid IN NUMBER
   );

END Maintenanceassignmentpackage;
/

CREATE OR REPLACE PACKAGE BODY Maintenanceassignmentpackage AS

   FUNCTION Checkmaintenanceassignment (
      P_Requestid IN NUMBER,
      P_Vendorid IN NUMBER,
      P_Propertyid IN NUMBER,
      P_Tenantid IN NUMBER
   ) RETURN BOOLEAN AS
      Vendorcount NUMBER;
      PRAGMA AUTONOMOUS_TRANSACTION; -- Use Autonomous Transaction To
Handle Exceptions
   BEGIN
      -- Check If The Vendor Is Assigned To The Specified Maintenance Request
      SELECT COUNT(*)
      INTO Vendorcount
      FROM P_Maintenance
      WHERE P_Requestid = Checkmaintenanceassignment.P_Requestid
      AND P_Vendorid = Checkmaintenanceassignment.P_Vendorid;

      -- Check If The Property And Tenant Are Associated With The Maintenance
Request
      RETURN Vendorcount > 0;
   EXCEPTION
      WHEN NO_DATA_FOUND THEN
         RETURN FALSE;
   END Checkmaintenanceassignment;

   PROCEDURE Assignmaintenancetovendor (
      P_Requestid IN NUMBER,
      P_Vendorid IN NUMBER,
      P_Propertyid IN NUMBER,
      P_Tenantid IN NUMBER
   ) AS
```

```
        Maintenancerecord P_Maintenance%ROWTYPE;
    BEGIN
        -- Check If The Vendor Is Already Assigned To The Maintenance Request
        IF Checkmaintenanceassignment(P_Requestid, P_Vendorid, P_Propertyid,
P_Tenantid) THEN
            DBMS_OUTPUT.PUT_LINE('Maintenance Request Is Already Assigned To The
Specified Vendor.');
        ELSE
            BEGIN
                -- Assign The Vendor To The Maintenance Request
                Maintenancerecord.P_Requestid := P_Requestid;
                Maintenancerecord.P_Propertyid := P_Propertyid;
                Maintenancerecord.P_Tenantid := P_Tenantid;
                Maintenancerecord.P_Vendorid := P_Vendorid;

                INSERT INTO P_Maintenance VALUES Maintenancerecord;
                COMMIT;
            EXCEPTION
                WHEN OTHERS THEN
                    DBMS_OUTPUT.PUT_LINE('Error Assigning Maintenance Request To The
Specified Vendor.');
                    ROLLBACK;
            END;

            DBMS_OUTPUT.PUT_LINE('Maintenance Request Assigned To The Specified
Vendor Successfully.');
        END IF;
    END Assignmaintenancetovendor;

END Maintenanceassignmentpackage;
/
```



```
----------
DECLARE
    v_request_id NUMBER := 1101;
    v_vendor_id NUMBER := 207;
    v_property_id NUMBER := 7;
    v_tenant_id NUMBER := 407;
    v_result BOOLEAN;
BEGIN
    -- Test the CheckMaintenanceAssignment function
    v_result := MaintenanceAssignmentPackage.CheckMaintenanceAssignment(v_request_id, v_vendor_id, v_property_id, v_tenant_id);

    IF v_result THEN
        DBMS_OUTPUT.PUT_LINE('Maintenance request is already assigned to the specified vendor.');
    ELSE
        -- Test the AssignMaintenanceToVendor procedure
        MaintenanceAssignmentPackage.AssignMaintenanceToVendor(v_request_id, v_vendor_id, v_property_id, v_tenant_id);
    END IF;
END;
/
```

```
Package MAINTENANCEASSIGNMENTPACKAGE compiled


Package Body MAINTENANCEASSIGNMENTPACKAGE compiled


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.

Maintenance request is already assigned to the specified vendor.
```

Analytical and Critical skills demonstrations:

Analyzing the database structure:

Efficiency:

- Data Retrieval: The database seems efficient in handling data retrieval due to the primary keys and foreign key constraints established. Retrieval should be optimized using primary key indexes for efficient access.

- Data Insertion and Updates: The structure appears suitable for insertion and updates. However, the efficiency might be affected if there are extensive cascading updates or if the database grows significantly in size.

Scalability:

- Future Growth: The database design can accommodate some growth. However, without additional scalability planning, it might face challenges if the data volume grows substantially.

- Changes in Requirements: It might accommodate minor changes, but substantial changes in requirements might lead to alterations in the schema.

Business Rule Compliance:

Table Relationships:

- The relationships between tables, established through primary and foreign keys, align with the business rules, ensuring data integrity and consistency.

- Challenges might arise if constraints need to be adjusted due to specific business scenarios, potentially affecting data entry or updates.

Constraint Adherence:

- The design appropriately implements constraints like primary key constraints, foreign key relationships, and data type constraints, aligning with business rules.

- In practice, there might be challenges in managing constraints, especially if modifications are needed to accommodate unique business scenarios.

Data Integrity:

- The database design, with proper foreign key references and constraints, ensures data integrity, reflecting the specified business rules.

- Challenges might arise in scenarios where data entry or updates require deviations from the established rules.

Handling Business Rule Changes:

- The design may face challenges when accommodating significant changes in business rules, necessitating schema modifications or constraint adjustments.

- Future updates or alterations to business rules might impact the database structure, possibly affecting existing data or requiring complex data migrations.

Decision-Making Process:

Normalization Levels:

- Rationale: The decision to normalize tables adhered to the normalization principles up to a certain level, aiming for data integrity and minimizing redundancy.

- Trade-offs: Achieving higher normalization might have led to more tables and complex queries, impacting performance during retrieval.

- Implications: By maintaining a balance between normalization and performance, the database promotes data consistency and minimizes redundant data, improving efficiency in updates.

Foreign Key Constraints:

- Rationale: Implementing foreign key constraints enforced relationships between tables, ensuring data consistency.

- Trade-offs: Overly strict foreign key constraints could restrict data entry and updates, impacting the flexibility of the database.

- Implications: While maintaining data integrity, there might be challenges in handling certain scenarios where constraints need to be temporarily bypassed or modified.

Critical Reflection:

Strengths:

- Comprehensive Requirement Gathering: The process began with a thorough understanding of requirements, ensuring that the database design aligned with business needs.

- Structured Approach to Database Design: The use of normalization, appropriate table structures, and relationships reflected a strong grasp of database design principles.

- Adherence to Business Rules: Efforts were made to align the database design with specified business rules, ensuring compliance and data integrity.

Areas for Improvement:

- Performance Optimization: Although some optimization techniques were applied, further exploration of indexing strategies and query optimization could enhance performance.

- Documentation and Maintenance: Enhancing documentation of the database design and maintenance procedures would streamline future modifications and troubleshooting.

- Testing and Validation: More emphasis on comprehensive testing strategies during the design phase could improve the identification and resolution of potential issues.

- Alternative Design Exploration: While the chosen design was justified, exploring more alternatives, and discussing their merits could provide a broader perspective.

- Future Planning and Adaptability: Focusing on future enhancements and adaptability to changing requirements could further refine the design for long-term efficiency.

Appendix:

Database DDL file:

```
- Creating Property table
CREATE TABLE P_Property (
   P_PropertyID NUMBER PRIMARY KEY,
   P_Address VARCHAR2(100),
   P_Type VARCHAR2(50),
   P_Size NUMBER,
   P_Amenities VARCHAR2(200),
   P_OwnershipDetails VARCHAR2(200)
);

-- Creating Insurance table
CREATE TABLE P_Insurance (
   P_InsuranceID NUMBER PRIMARY KEY,
   P_PropertyID NUMBER REFERENCES P_Property(P_PropertyID),
   P_InsuranceProvider VARCHAR2(100),
   P_CoverageDetails VARCHAR2(200),
   P_ExpiryDate DATE
);

-- Creating Calendar table
CREATE TABLE P_Calendar (
   P_EventID NUMBER PRIMARY KEY,
   P_PropertyID NUMBER REFERENCES P_Property(P_PropertyID),
   P_EventType VARCHAR2(50),
    P_EventDate DATE,
    P_Description VARCHAR2(200)
);

-- Creating Vendor table
CREATE TABLE P_Vendor (
    P_VendorID NUMBER PRIMARY KEY,
    P_Name VARCHAR2(100),
    P_ContactInformation VARCHAR2(100),
    P_ServicesProvided VARCHAR2(200),
    P_ContractDetails VARCHAR2(200)
);

-- Creating UserAccounts table
CREATE TABLE P_UserAccounts (
    P_UserID NUMBER PRIMARY KEY,
    P_PropertyID NUMBER REFERENCES P_Property(P_PropertyID),
    P_Name VARCHAR2(100),
    P_ContactInformation VARCHAR2(100),
    P_Role VARCHAR2(50),
    P_Permissions VARCHAR2(200)
);

-- Creating Tenant table
CREATE TABLE P_Tenant (
    P_TenantID NUMBER PRIMARY KEY,
    P_PropertyID NUMBER REFERENCES P_Property(P_PropertyID),
    P_Name VARCHAR2(100),
    P_ContactInformation VARCHAR2(100),
    P_LeaseDetails VARCHAR2(200),
    P_PaymentHistory VARCHAR2(200),
    P_Preferences VARCHAR2(200)
);

-- Creating LeaseTerms table
CREATE TABLE P_LeaseTerms (
    P_LeaseTermID NUMBER PRIMARY KEY,
    P_TenantID NUMBER REFERENCES P_Tenant(P_TenantID),
    P_StartDate DATE, -- Changed the name to P_StartDate
    P_EndDate DATE, -- Changed the name to P_EndDate
    P_RentAmount NUMBER, -- Added P_RentAmount column
    P_PaymentSchedule VARCHAR2(50), -- Adjusted size to 50 characters
    P_Deposit NUMBER -- Added P_Deposit column
);

--

-- Creating EmergencyContacts table
CREATE TABLE P_EmergencyContacts (
    P_EmergencyContactID NUMBER PRIMARY KEY,
    P_PropertyID NUMBER REFERENCES P_Property(P_PropertyID),
    P_TenantID NUMBER REFERENCES P_Tenant(P_TenantID),
    P_ContactType VARCHAR2(50),
    P_ContactInformation VARCHAR2(100)
);
```

```
-- Creating Document table
CREATE TABLE P_Document (
   P_DocumentID NUMBER PRIMARY KEY,
   P_PropertyID NUMBER REFERENCES P_Property(P_PropertyID),
   P_TenantID NUMBER REFERENCES P_Tenant(P_TenantID),
   P_VendorID NUMBER REFERENCES P_Vendor(P_VendorID),
   P_DocumentType VARCHAR2(50),
   P_Title VARCHAR2(100),
   P_UploadDate DATE
);

-- Creating ParkingManagement table
CREATE TABLE P_ParkingManagement (
   P_ParkingID NUMBER PRIMARY KEY,
   P_PropertyID NUMBER REFERENCES P_Property(P_PropertyID),
   P_TenantID NUMBER REFERENCES P_Tenant(P_TenantID),
   P_ParkingSpaceNumber VARCHAR2(50),
   P_AvailabilityStatus VARCHAR2(50)
);

-- Creating FinancialTransactions table
CREATE TABLE P_FinancialTransactions (
   P_TransactionID NUMBER PRIMARY KEY,
   P_PropertyID NUMBER REFERENCES P_Property(P_PropertyID),
   P_TenantID NUMBER REFERENCES P_Tenant(P_TenantID),
   P_VendorID NUMBER REFERENCES P_Vendor(P_VendorID),
   P_TransactionType VARCHAR2(50),
   P_Amount NUMBER,
    P_TransactionDate DATE,
    P_PaymentStatus VARCHAR2(50)
);

-- Creating UtilityManagement table
CREATE TABLE P_UtilityManagement (
    P_UtilityID NUMBER PRIMARY KEY,
    P_PropertyID NUMBER REFERENCES P_Property(P_PropertyID),
    P_UtilityType VARCHAR2(50),
    P_Provider VARCHAR2(100),
    P_BillingDetails VARCHAR2(200)
);

-- Creating Maintenance table
CREATE TABLE P_Maintenance (
    P_RequestID NUMBER PRIMARY KEY,
    P_PropertyID NUMBER REFERENCES P_Property(P_PropertyID),
    P_TenantID NUMBER REFERENCES P_Tenant(P_TenantID),
    P_VendorID NUMBER REFERENCES P_Vendor(P_VendorID),
    P_IssueDescription VARCHAR2(200),
    P_AssignedStaff VARCHAR2(100),
    P_Status VARCHAR2(50),
    P_CompletionDate DATE
);

-- Creating AnalysisAndReporting table
CREATE TABLE P_AnalysisAndReporting (
    P_ReportID NUMBER PRIMARY KEY,
    P_PropertyID NUMBER REFERENCES P_Property(P_PropertyID),
    P_ReportType VARCHAR2(50),
    P_DateGenerated DATE,
    P_Insights VARCHAR2(200)
);

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

-- Creating sequences
CREATE SEQUENCE P_PropertySeq START WITH 1 INCREMENT BY 1;
CREATE SEQUENCE P_InsuranceSeq START WITH 11 INCREMENT BY 1;
CREATE SEQUENCE P_CalendarSeq START WITH 101 INCREMENT BY 1;
CREATE SEQUENCE P_VendorSeq START WITH 201 INCREMENT BY 1;
CREATE SEQUENCE P_UserAccountsSeq START WITH 301 INCREMENT BY 1;
CREATE SEQUENCE P_TenantSeq START WITH 401 INCREMENT BY 1;
CREATE SEQUENCE P_LeaseTermsSeq START WITH 501 INCREMENT BY 1;
CREATE SEQUENCE P_EmergencyContactsSeq START WITH 601 INCREMENT BY
1;
CREATE SEQUENCE P_DocumentSeq START WITH 701 INCREMENT BY 1;
CREATE SEQUENCE P_ParkingManagementSeq START WITH 801 INCREMENT BY
1;
CREATE SEQUENCE P_FinancialTransactionsSeq START WITH 901 INCREMENT BY
1;
CREATE SEQUENCE P_UtilityManagementSeq START WITH 1001 INCREMENT BY
1;
CREATE SEQUENCE P_MaintenanceSeq START WITH 1101 INCREMENT BY 1;
CREATE SEQUENCE P_AnalysisAndReportingSeq START WITH 1201 INCREMENT
BY 1;
```

```
------------------------------------------------------------------

--1 Inserting data into P_Property table using sequences

INSERT INTO P_Property (P_PropertyID, P_Address, P_Type, P_Size, P_Amenities,
P_OwnershipDetails)
VALUES (P_PropertySeq.NEXTVAL, '123 Main St', 'Apartment', 1000, 'Swimming pool,
Gym', 'Owned by John Doe');

INSERT INTO P_Property (P_PropertyID, P_Address, P_Type, P_Size, P_Amenities,
P_OwnershipDetails)
VALUES (P_PropertySeq.NEXTVAL, '456 Elm St', 'House', 2000, 'Backyard, Garage',
'Owned by Jane Smith');

INSERT INTO P_Property (P_PropertyID, P_Address, P_Type, P_Size, P_Amenities,
P_OwnershipDetails)
VALUES (P_PropertySeq.NEXTVAL, '789 Oak St', 'Condo', 1500, 'Security, Parking',
'Owned by Michael Johnson');

INSERT INTO P_Property (P_PropertyID, P_Address, P_Type, P_Size, P_Amenities,
P_OwnershipDetails)
VALUES (P_PropertySeq.NEXTVAL, '101 Pine St', 'Townhouse', 1800, 'Garden,
Community Center', 'Owned by Emily Davis');

INSERT INTO P_Property (P_PropertyID, P_Address, P_Type, P_Size, P_Amenities,
P_OwnershipDetails)
VALUES (P_PropertySeq.NEXTVAL, '222 Maple St', 'Duplex', 2200, 'Balcony, Parking',
'Owned by Robert Brown');

INSERT INTO P_Property (P_PropertyID, P_Address, P_Type, P_Size, P_Amenities,
P_OwnershipDetails)
VALUES (P_PropertySeq.NEXTVAL, '333 Cherry St', 'Villa', 3000, 'Private pool,
Garden', 'Owned by Sarah Wilson');
```

```sql
INSERT INTO P_Property (P_PropertyID, P_Address, P_Type, P_Size, P_Amenities,
P_OwnershipDetails)
VALUES (P_PropertySeq.NEXTVAL, '444 Walnut St', 'Studio', 800, 'Laundry, Gym',
'Owned by Thomas Miller');

INSERT INTO P_Property (P_PropertyID, P_Address, P_Type, P_Size, P_Amenities,
P_OwnershipDetails)
VALUES (P_PropertySeq.NEXTVAL, '444 Walnut St', 'Studio', 800, 'Laundry, Gym',
'Owned by Thomas Miller');

INSERT INTO P_Property (P_PropertyID, P_Address, P_Type, P_Size, P_Amenities,
P_OwnershipDetails)
VALUES (P_PropertySeq.NEXTVAL, '123 Oak St', 'Apartment', 1200, 'Swimming Pool,
Parking', 'Owned by Jane Johnson');

INSERT INTO P_Property (P_PropertyID, P_Address, P_Type, P_Size, P_Amenities,
P_OwnershipDetails)
VALUES (P_PropertySeq.NEXTVAL, '789 Pine St', 'House', 2000, 'Garden, Fireplace',
'Owned by Robert Davis');

--------

--2 Inserting data into P_Insurance table using sequences for both P_InsuranceID and
P_PropertyID
INSERT INTO P_Insurance (P_InsuranceID, P_PropertyID, P_InsuranceProvider,
P_CoverageDetails, P_ExpiryDate)
VALUES (P_InsuranceSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'ABC Insurance',
'Fire, Theft', TO_DATE('2024-01-01', 'YYYY-MM-DD'));
```

```
INSERT INTO P_Insurance (P_InsuranceID, P_PropertyID, P_InsuranceProvider,
P_CoverageDetails, P_ExpiryDate)
VALUES (P_InsuranceSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'XYZ Insurance',
'Flood, Fire', TO_DATE('2023-12-31', 'YYYY-MM-DD'));

INSERT INTO P_Insurance (P_InsuranceID, P_PropertyID, P_InsuranceProvider,
P_CoverageDetails, P_ExpiryDate)
VALUES (P_InsuranceSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'LMN Insurance',
'Theft, Liability', TO_DATE('2024-06-30', 'YYYY-MM-DD'));

INSERT INTO P_Insurance (P_InsuranceID, P_PropertyID, P_InsuranceProvider,
P_CoverageDetails, P_ExpiryDate)
VALUES (P_InsuranceSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'PQR Insurance',
'Fire, Flood', TO_DATE('2023-11-30', 'YYYY-MM-DD'));

INSERT INTO P_Insurance (P_InsuranceID, P_PropertyID, P_InsuranceProvider,
P_CoverageDetails, P_ExpiryDate)
VALUES (P_InsuranceSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'RST Insurance',
'Theft, Vandalism', TO_DATE('2024-03-15', 'YYYY-MM-DD'));

INSERT INTO P_Insurance (P_InsuranceID, P_PropertyID, P_InsuranceProvider,
P_CoverageDetails, P_ExpiryDate)
VALUES (P_InsuranceSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'UVW Insurance',
'Fire, Liability', TO_DATE('2023-10-20', 'YYYY-MM-DD'));

INSERT INTO P_Insurance (P_InsuranceID, P_PropertyID, P_InsuranceProvider,
P_CoverageDetails, P_ExpiryDate)
VALUES (P_InsuranceSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'GHI Insurance',
'Theft, Flood', TO_DATE('2024-05-05', 'YYYY-MM-DD'));

INSERT INTO P_Insurance (P_InsuranceID, P_PropertyID, P_InsuranceProvider,
P_CoverageDetails, P_ExpiryDate)
```

```
VALUES (P_InsuranceSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'GHI Insurance',
'Theft, Flood', TO_DATE('2024-05-05', 'YYYY-MM-DD'));


INSERT INTO P_Insurance (P_InsuranceID, P_PropertyID, P_InsuranceProvider,
P_CoverageDetails, P_ExpiryDate)
VALUES (P_InsuranceSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'ABC Insurance',
'Fire, Liability', TO_DATE('2024-08-15', 'YYYY-MM-DD'));


INSERT INTO P_Insurance (P_InsuranceID, P_PropertyID, P_InsuranceProvider,
P_CoverageDetails, P_ExpiryDate)
VALUES (P_InsuranceSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'XYZ Insurance',
'Earthquake, Vandalism', TO_DATE('2025-01-10', 'YYYY-MM-DD'));


---


--3 Inserting data into P_Calendar table using sequences for both P_EventID and
P_PropertyID
INSERT INTO P_Calendar (P_EventID, P_PropertyID, P_EventType, P_EventDate,
P_Description)
VALUES (P_CalendarSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'Maintenance',
TO_DATE('2023-01-15', 'YYYY-MM-DD'), 'Routine maintenance work');


INSERT INTO P_Calendar (P_EventID, P_PropertyID, P_EventType, P_EventDate,
P_Description)
VALUES (P_CalendarSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'Meeting',
TO_DATE('2023-02-28', 'YYYY-MM-DD'), 'Property owners meeting');


INSERT INTO P_Calendar (P_EventID, P_PropertyID, P_EventType, P_EventDate,
P_Description)
VALUES (P_CalendarSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'Event',
TO_DATE('2023-03-10', 'YYYY-MM-DD'), 'Community BBQ');
```

```
INSERT INTO P_Calendar (P_EventID, P_PropertyID, P_EventType, P_EventDate,
P_Description)
VALUES (P_CalendarSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'Maintenance',
TO_DATE('2023-04-05', 'YYYY-MM-DD'), 'Repairs and servicing');

INSERT INTO P_Calendar (P_EventID, P_PropertyID, P_EventType, P_EventDate,
P_Description)
VALUES (P_CalendarSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'Event',
TO_DATE('2023-05-20', 'YYYY-MM-DD'), 'Tenant appreciation day');

INSERT INTO P_Calendar (P_EventID, P_PropertyID, P_EventType, P_EventDate,
P_Description)
VALUES (P_CalendarSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'Meeting',
TO_DATE('2023-06-08', 'YYYY-MM-DD'), 'Board meeting');

INSERT INTO P_Calendar (P_EventID, P_PropertyID, P_EventType, P_EventDate,
P_Description)
VALUES (P_CalendarSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'Maintenance',
TO_DATE('2023-07-30', 'YYYY-MM-DD'), 'Building inspections');

INSERT INTO P_Calendar (P_EventID, P_PropertyID, P_EventType, P_EventDate,
P_Description)
VALUES (P_CalendarSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'Maintenance',
TO_DATE('2023-01-15', 'YYYY-MM-DD'), 'Routine maintenance work');

INSERT INTO P_Calendar (P_EventID, P_PropertyID, P_EventType, P_EventDate,
P_Description)
VALUES (P_CalendarSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'Meeting',
TO_DATE('2023-02-28', 'YYYY-MM-DD'), 'Property owners meeting');

INSERT INTO P_Calendar (P_EventID, P_PropertyID, P_EventType, P_EventDate,
P_Description)
```

```
VALUES (P_CalendarSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'Event',
TO_DATE('2023-03-10', 'YYYY-MM-DD'), 'Community BBQ');
```

```
--4 Inserting data into P_Vendor table using sequences
INSERT INTO P_Vendor (P_VendorID, P_Name, P_ContactInformation,
P_ServicesProvided, P_ContractDetails)
VALUES (P_VendorSeq.NEXTVAL, 'ABC Services', 'abc@example.com, 123-456-
7890', 'Cleaning, Maintenance', 'Annual contract for cleaning services');

INSERT INTO P_Vendor (P_VendorID, P_Name, P_ContactInformation,
P_ServicesProvided, P_ContractDetails)
VALUES (P_VendorSeq.NEXTVAL, 'XYZ Contractors', 'xyz@example.com, 987-654-
3210', 'Repair, Renovation', 'Bi-annual renovation contract');

INSERT INTO P_Vendor (P_VendorID, P_Name, P_ContactInformation,
P_ServicesProvided, P_ContractDetails)
VALUES (P_VendorSeq.NEXTVAL, 'LMN Landscaping', 'lmn@example.com, 567-890-
1234', 'Landscaping, Gardening', 'Monthly landscaping services');

INSERT INTO P_Vendor (P_VendorID, P_Name, P_ContactInformation,
P_ServicesProvided, P_ContractDetails)
VALUES (P_VendorSeq.NEXTVAL, 'PQR Plumbers', 'pqr@example.com, 345-678-
9012', 'Plumbing, Pipe repairs', 'Emergency plumbing services');

INSERT INTO P_Vendor (P_VendorID, P_Name, P_ContactInformation,
P_ServicesProvided, P_ContractDetails)
VALUES (P_VendorSeq.NEXTVAL, 'RST Electricians', 'rst@example.com, 234-567-
8901', 'Electrical services', 'Annual maintenance contract for electrical work');
```

```
INSERT INTO P_Vendor (P_VendorID, P_Name, P_ContactInformation, P_ServicesProvided, P_ContractDetails)

VALUES (P_VendorSeq.NEXTVAL, 'UVW Security', 'uvw@example.com, 456-789-0123', 'Security systems, Surveillance', 'Bi-annual security surveillance contract');


INSERT INTO P_Vendor (P_VendorID, P_Name, P_ContactInformation, P_ServicesProvided, P_ContractDetails)

VALUES (P_VendorSeq.NEXTVAL, 'GHI HVAC', 'ghi@example.com, 789-012-3456', 'Heating, Ventilation, Air Conditioning', 'Quarterly HVAC maintenance services');


INSERT INTO P_Vendor (P_VendorID, P_Name, P_ContactInformation, P_ServicesProvided, P_ContractDetails)

VALUES (P_VendorSeq.NEXTVAL, 'MNO Architects', 'mno@example.com, 123-789-4560', 'Architectural Design, Planning', 'Design services for construction projects');


INSERT INTO P_Vendor (P_VendorID, P_Name, P_ContactInformation, P_ServicesProvided, P_ContractDetails)

VALUES (P_VendorSeq.NEXTVAL, 'JKL Flooring', 'jkl@example.com, 456-123-7890', 'Flooring Installation, Restoration', 'Annual contract for flooring maintenance');


INSERT INTO P_Vendor (P_VendorID, P_Name, P_ContactInformation, P_ServicesProvided, P_ContractDetails)

VALUES (P_VendorSeq.NEXTVAL, 'NOP Painters', 'nop@example.com, 789-456-1230', 'Painting, Wall Covering', 'Quarterly painting services for interior and exterior');


---


--5 Inserting data into P_UserAccounts table using sequences and P_PropertySeq for P_PropertyID

INSERT INTO P_UserAccounts (P_UserID, P_PropertyID, P_Name, P_ContactInformation, P_Role, P_Permissions)

VALUES (P_UserAccountsSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'John Doe', 'john@example.com, 123-456-7890', 'Manager', 'Full access');
```

```
INSERT INTO P_UserAccounts (P_UserID, P_PropertyID, P_Name,
P_ContactInformation, P_Role, P_Permissions)
VALUES (P_UserAccountsSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'Alice Smith',
'alice@example.com, 987-654-3210', 'Supervisor', 'Limited access');

INSERT INTO P_UserAccounts (P_UserID, P_PropertyID, P_Name,
P_ContactInformation, P_Role, P_Permissions)
VALUES (P_UserAccountsSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'Ethan Brown',
'ethan@example.com, 567-890-1234', 'Administrator', 'Full access');

INSERT INTO P_UserAccounts (P_UserID, P_PropertyID, P_Name,
P_ContactInformation, P_Role, P_Permissions)
VALUES (P_UserAccountsSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'Sophia
Johnson', 'sophia@example.com, 345-678-9012', 'Staff', 'Limited access');

INSERT INTO P_UserAccounts (P_UserID, P_PropertyID, P_Name,
P_ContactInformation, P_Role, P_Permissions)
VALUES (P_UserAccountsSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'William Davis',
'william@example.com, 234-567-8901', 'Analyst', 'View only');

INSERT INTO P_UserAccounts (P_UserID, P_PropertyID, P_Name,
P_ContactInformation, P_Role, P_Permissions)
VALUES (P_UserAccountsSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'Olivia Wilson',
'olivia@example.com, 456-789-0123', 'Coordinator', 'Limited access');

INSERT INTO P_UserAccounts (P_UserID, P_PropertyID, P_Name,
P_ContactInformation, P_Role, P_Permissions)
VALUES (P_UserAccountsSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'James Miller',
'james@example.com, 789-012-3456', 'Assistant', 'Limited access');

INSERT INTO P_UserAccounts (P_UserID, P_PropertyID, P_Name,
P_ContactInformation, P_Role, P_Permissions)
```

```
VALUES (P_UserAccountsSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'John Doe',
'john@example.com, 123-456-7890', 'Manager', 'Full access');


INSERT INTO P_UserAccounts (P_UserID, P_PropertyID, P_Name,
P_ContactInformation, P_Role, P_Permissions)

VALUES (P_UserAccountsSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'Alice Smith',
'alice@example.com, 987-654-3210', 'Supervisor', 'Limited access');


INSERT INTO P_UserAccounts (P_UserID, P_PropertyID, P_Name,
P_ContactInformation, P_Role, P_Permissions)

VALUES (P_UserAccountsSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'Ethan Brown',
'ethan@example.com, 567-890-1234', 'Administrator', 'Full access');


---


--6 Inserting data into P_Tenant table using sequences and P_PropertySeq for
P_PropertyID

INSERT INTO P_Tenant (P_TenantID, P_PropertyID, P_Name, P_ContactInformation,
P_LeaseDetails, P_PaymentHistory, P_Preferences)

VALUES (P_TenantSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'Emma Thompson',
'emma@example.com, 123-456-7890', 'Lease signed on 2023-01-01', 'Payment records
up to date', 'Quiet tenant, prefers email communication');


INSERT INTO P_Tenant (P_TenantID, P_PropertyID, P_Name, P_ContactInformation,
P_LeaseDetails, P_PaymentHistory, P_Preferences)

VALUES (P_TenantSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'Noah Garcia',
'noah@example.com, 987-654-3210', 'Lease signed on 2023-02-15', 'Regular
payments, no issues', 'Pet-friendly, prefers direct deposit');


INSERT INTO P_Tenant (P_TenantID, P_PropertyID, P_Name, P_ContactInformation,
P_LeaseDetails, P_PaymentHistory, P_Preferences)
```

VALUES (P_TenantSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'Ava Martinez', 'ava@example.com, 567-890-1234', 'Lease signed on 2023-03-20', 'Occasional late payments, otherwise good', 'Needs parking space, prefers text messages');

INSERT INTO P_Tenant (P_TenantID, P_PropertyID, P_Name, P_ContactInformation, P_LeaseDetails, P_PaymentHistory, P_Preferences)

VALUES (P_TenantSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'Liam Robinson', 'liam@example.com, 345-678-9012', 'Lease signed on 2023-04-10', 'Frequent delays in payments', 'Quiet tenant, prefers phone calls');

INSERT INTO P_Tenant (P_TenantID, P_PropertyID, P_Name, P_ContactInformation, P_LeaseDetails, P_PaymentHistory, P_Preferences)

VALUES (P_TenantSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'Isabella Reed', 'isabella@example.com, 234-567-8901', 'Lease signed on 2023-05-05', 'Consistently on time with payments', 'Prefers email communication, emergency contact available');

INSERT INTO P_Tenant (P_TenantID, P_PropertyID, P_Name, P_ContactInformation, P_LeaseDetails, P_PaymentHistory, P_Preferences)

VALUES (P_TenantSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'Mason Hill', 'mason@example.com, 456-789-0123', 'Lease signed on 2023-06-30', 'Payment history irregular', 'Needs maintenance for heating system, prefers email communication');

INSERT INTO P_Tenant (P_TenantID, P_PropertyID, P_Name, P_ContactInformation, P_LeaseDetails, P_PaymentHistory, P_Preferences)

VALUES (P_TenantSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'Sophia Cook', 'sophia@example.com, 789-012-3456', 'Lease signed on 2023-07-20', 'Frequent communication for payment delays', 'Prefers phone calls, emergency contact available');

INSERT INTO P_Tenant (P_TenantID, P_PropertyID, P_Name, P_ContactInformation, P_LeaseDetails, P_PaymentHistory, P_Preferences)

VALUES (P_TenantSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'Emma Thompson', 'emma@example.com, 123-456-7890', 'Lease signed on 2023-01-01', 'Payment records up to date', 'Quiet tenant, prefers email communication');

INSERT INTO P_Tenant (P_TenantID, P_PropertyID, P_Name, P_ContactInformation, P_LeaseDetails, P_PaymentHistory, P_Preferences)

VALUES (P_TenantSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'Noah Garcia', 'noah@example.com, 987-654-3210', 'Lease signed on 2023-02-15', 'Regular payments, no issues', 'Pet-friendly, prefers direct deposit');

INSERT INTO P_Tenant (P_TenantID, P_PropertyID, P_Name, P_ContactInformation, P_LeaseDetails, P_PaymentHistory, P_Preferences)

VALUES (P_TenantSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'Ava Martinez', 'ava@example.com, 567-890-1234', 'Lease signed on 2023-03-20', 'Occasional late payments, otherwise good', 'Needs parking space, prefers text messages');

--7 Inserting data into P_LeaseTerms table using sequences and P_TenantSeq for P_TenantID

INSERT INTO P_LeaseTerms (P_LeaseTermID, P_TenantID, P_StartDate, P_EndDate, P_RentAmount, P_PaymentSchedule, P_Deposit)

VALUES (P_LeaseTermsSeq.NEXTVAL, P_TenantSeq.CURRVAL, TO_DATE('2023-01-01', 'YYYY-MM-DD'), TO_DATE('2023-12-31', 'YYYY-MM-DD'), 1500, 'Monthly', 2000);

INSERT INTO P_LeaseTerms (P_LeaseTermID, P_TenantID, P_StartDate, P_EndDate, P_RentAmount, P_PaymentSchedule, P_Deposit)

VALUES (P_LeaseTermsSeq.NEXTVAL, P_TenantSeq.CURRVAL, TO_DATE('2023-02-15', 'YYYY-MM-DD'), TO_DATE('2024-02-14', 'YYYY-MM-DD'), 1800, 'Monthly', 2500);

INSERT INTO P_LeaseTerms (P_LeaseTermID, P_TenantID, P_StartDate, P_EndDate, P_RentAmount, P_PaymentSchedule, P_Deposit)

VALUES (P_LeaseTermsSeq.NEXTVAL, P_TenantSeq.CURRVAL, TO_DATE('2023-03-20', 'YYYY-MM-DD'), TO_DATE('2024-03-19', 'YYYY-MM-DD'), 1700, 'Monthly', 2200);

INSERT INTO P_LeaseTerms (P_LeaseTermID, P_TenantID, P_StartDate, P_EndDate, P_RentAmount, P_PaymentSchedule, P_Deposit)

```
VALUES (P_LeaseTermsSeq.NEXTVAL, P_TenantSeq.CURRVAL, TO_DATE('2023-04-10', 'YYYY-MM-DD'), TO_DATE('2024-04-09', 'YYYY-MM-DD'), 1600, 'Monthly', 2300);

INSERT INTO P_LeaseTerms (P_LeaseTermID, P_TenantID, P_StartDate, P_EndDate, P_RentAmount, P_PaymentSchedule, P_Deposit)
VALUES (P_LeaseTermsSeq.NEXTVAL, P_TenantSeq.CURRVAL, TO_DATE('2023-05-05', 'YYYY-MM-DD'), TO_DATE('2024-05-04', 'YYYY-MM-DD'), 1900, 'Monthly', 2400);

INSERT INTO P_LeaseTerms (P_LeaseTermID, P_TenantID, P_StartDate, P_EndDate, P_RentAmount, P_PaymentSchedule, P_Deposit)
VALUES (P_LeaseTermsSeq.NEXTVAL, P_TenantSeq.CURRVAL, TO_DATE('2023-06-30', 'YYYY-MM-DD'), TO_DATE('2024-06-29', 'YYYY-MM-DD'), 1750, 'Monthly', 2100);

INSERT INTO P_LeaseTerms (P_LeaseTermID, P_TenantID, P_StartDate, P_EndDate, P_RentAmount, P_PaymentSchedule, P_Deposit)
VALUES (P_LeaseTermsSeq.NEXTVAL, P_TenantSeq.CURRVAL, TO_DATE('2023-07-20', 'YYYY-MM-DD'), TO_DATE('2024-07-19', 'YYYY-MM-DD'), 2000, 'Monthly', 2600);

INSERT INTO P_LeaseTerms (P_LeaseTermID, P_TenantID, P_StartDate, P_EndDate, P_RentAmount, P_PaymentSchedule, P_Deposit)
VALUES (P_LeaseTermsSeq.NEXTVAL, P_TenantSeq.CURRVAL, TO_DATE('2023-08-15', 'YYYY-MM-DD'), TO_DATE('2024-08-14', 'YYYY-MM-DD'), 1600, 'Monthly', 2300);

INSERT INTO P_LeaseTerms (P_LeaseTermID, P_TenantID, P_StartDate, P_EndDate, P_RentAmount, P_PaymentSchedule, P_Deposit)
VALUES (P_LeaseTermsSeq.NEXTVAL, P_TenantSeq.CURRVAL, TO_DATE('2023-09-10', 'YYYY-MM-DD'), TO_DATE('2024-09-09', 'YYYY-MM-DD'), 1800, 'Monthly', 2500);

INSERT INTO P_LeaseTerms (P_LeaseTermID, P_TenantID, P_StartDate, P_EndDate, P_RentAmount, P_PaymentSchedule, P_Deposit)
VALUES (P_LeaseTermsSeq.NEXTVAL, P_TenantSeq.CURRVAL, TO_DATE('2023-10-05', 'YYYY-MM-DD'), TO_DATE('2024-10-04', 'YYYY-MM-DD'), 1900, 'Monthly', 2400);
```

--8 Inserting data into P_EmergencyContacts table using sequences and P_PropertySeq, P_TenantSeq for P_PropertyID, P_TenantID

INSERT INTO P_EmergencyContacts (P_EmergencyContactID, P_PropertyID, P_TenantID, P_ContactType, P_ContactInformation)

VALUES (P_EmergencyContactsSeq.NEXTVAL, P_PropertySeq.CURRVAL, P_TenantSeq.CURRVAL, 'Family', 'John Doe - 123-456-7890');

INSERT INTO P_EmergencyContacts (P_EmergencyContactID, P_PropertyID, P_TenantID, P_ContactType, P_ContactInformation)

VALUES (P_EmergencyContactsSeq.NEXTVAL, P_PropertySeq.CURRVAL, P_TenantSeq.CURRVAL, 'Neighbor', 'Alice Smith - 987-654-3210');

INSERT INTO P_EmergencyContacts (P_EmergencyContactID, P_PropertyID, P_TenantID, P_ContactType, P_ContactInformation)

VALUES (P_EmergencyContactsSeq.NEXTVAL, P_PropertySeq.CURRVAL, P_TenantSeq.CURRVAL, 'Friend', 'Ethan Brown - 567-890-1234');

INSERT INTO P_EmergencyContacts (P_EmergencyContactID, P_PropertyID, P_TenantID, P_ContactType, P_ContactInformation)

VALUES (P_EmergencyContactsSeq.NEXTVAL, P_PropertySeq.CURRVAL, P_TenantSeq.CURRVAL, 'Relative', 'Sophia Johnson - 345-678-9012');

INSERT INTO P_EmergencyContacts (P_EmergencyContactID, P_PropertyID, P_TenantID, P_ContactType, P_ContactInformation)

VALUES (P_EmergencyContactsSeq.NEXTVAL, P_PropertySeq.CURRVAL, P_TenantSeq.CURRVAL, 'Family', 'William Davis - 234-567-8901');

INSERT INTO P_EmergencyContacts (P_EmergencyContactID, P_PropertyID, P_TenantID, P_ContactType, P_ContactInformation)

VALUES (P_EmergencyContactsSeq.NEXTVAL, P_PropertySeq.CURRVAL, P_TenantSeq.CURRVAL, 'Neighbor', 'Olivia Wilson - 456-789-0123');

```
INSERT INTO P_EmergencyContacts (P_EmergencyContactID, P_PropertyID,
P_TenantID, P_ContactType, P_ContactInformation)

VALUES (P_EmergencyContactsSeq.NEXTVAL, P_PropertySeq.CURRVAL,
P_TenantSeq.CURRVAL, 'Friend', 'James Miller - 789-012-3456');


INSERT INTO P_EmergencyContacts (P_EmergencyContactID, P_PropertyID,
P_TenantID, P_ContactType, P_ContactInformation)

VALUES (P_EmergencyContactsSeq.NEXTVAL, P_PropertySeq.CURRVAL,
P_TenantSeq.CURRVAL, 'Family', 'John Doe - 123-456-7890');


INSERT INTO P_EmergencyContacts (P_EmergencyContactID, P_PropertyID,
P_TenantID, P_ContactType, P_ContactInformation)

VALUES (P_EmergencyContactsSeq.NEXTVAL, P_PropertySeq.CURRVAL,
P_TenantSeq.CURRVAL, 'Neighbor', 'Alice Smith - 987-654-3210');


INSERT INTO P_EmergencyContacts (P_EmergencyContactID, P_PropertyID,
P_TenantID, P_ContactType, P_ContactInformation)

VALUES (P_EmergencyContactsSeq.NEXTVAL, P_PropertySeq.CURRVAL,
P_TenantSeq.CURRVAL, 'Friend', 'Ethan Brown - 567-890-1234');


INSERT INTO P_Document (P_DocumentID, P_PropertyID, P_TenantID, P_VendorID,
P_DocumentType, P_Title, P_UploadDate)

VALUES (P_DocumentSeq.NEXTVAL, P_PropertySeq.CURRVAL,
P_TenantSeq.CURRVAL, P_VendorSeq.CURRVAL, 'Lease Agreement', 'Lease
Agreement - Tenant A', TO_DATE('2023-01-15', 'YYYY-MM-DD'));


INSERT INTO P_Document (P_DocumentID, P_PropertyID, P_TenantID, P_VendorID,
P_DocumentType, P_Title, P_UploadDate)

VALUES (P_DocumentSeq.NEXTVAL, P_PropertySeq.CURRVAL,
P_TenantSeq.CURRVAL, P_VendorSeq.CURRVAL, 'Utility Bill', 'Utility Bill - January
2023', TO_DATE('2023-02-20', 'YYYY-MM-DD'));
```

INSERT INTO P_Document (P_DocumentID, P_PropertyID, P_TenantID, P_VendorID, P_DocumentType, P_Title, P_UploadDate)

VALUES (P_DocumentSeq.NEXTVAL, P_PropertySeq.CURRVAL, P_TenantSeq.CURRVAL, P_VendorSeq.CURRVAL, 'Receipt', 'Receipt - Maintenance Payment', TO_DATE('2023-03-10', 'YYYY-MM-DD'));

--10 Inserting data into P_ParkingManagement table using sequences and P_PropertySeq, P_TenantSeq for P_PropertyID, P_TenantID

INSERT INTO P_ParkingManagement (P_ParkingID, P_PropertyID, P_TenantID, P_ParkingSpaceNumber, P_AvailabilityStatus)

VALUES (P_ParkingManagementSeq.NEXTVAL, P_PropertySeq.CURRVAL, P_TenantSeq.CURRVAL, 'P101', 'Available');

INSERT INTO P_ParkingManagement (P_ParkingID, P_PropertyID, P_TenantID, P_ParkingSpaceNumber, P_AvailabilityStatus)

VALUES (P_ParkingManagementSeq.NEXTVAL, P_PropertySeq.CURRVAL, P_TenantSeq.CURRVAL, 'P102', 'Occupied');

INSERT INTO P_ParkingManagement (P_ParkingID, P_PropertyID, P_TenantID, P_ParkingSpaceNumber, P_AvailabilityStatus)

VALUES (P_ParkingManagementSeq.NEXTVAL, P_PropertySeq.CURRVAL, P_TenantSeq.CURRVAL, 'P103', 'Available');

INSERT INTO P_ParkingManagement (P_ParkingID, P_PropertyID, P_TenantID, P_ParkingSpaceNumber, P_AvailabilityStatus)

VALUES (P_ParkingManagementSeq.NEXTVAL, P_PropertySeq.CURRVAL, P_TenantSeq.CURRVAL, 'P104', 'Reserved');

INSERT INTO P_ParkingManagement (P_ParkingID, P_PropertyID, P_TenantID, P_ParkingSpaceNumber, P_AvailabilityStatus)

```
VALUES (P_ParkingManagementSeq.NEXTVAL, P_PropertySeq.CURRVAL,
P_TenantSeq.CURRVAL, 'P105', 'Available');


INSERT INTO P_ParkingManagement (P_ParkingID, P_PropertyID, P_TenantID,
P_ParkingSpaceNumber, P_AvailabilityStatus)
VALUES (P_ParkingManagementSeq.NEXTVAL, P_PropertySeq.CURRVAL,
P_TenantSeq.CURRVAL, 'P106', 'Occupied');


INSERT INTO P_ParkingManagement (P_ParkingID, P_PropertyID, P_TenantID,
P_ParkingSpaceNumber, P_AvailabilityStatus)
VALUES (P_ParkingManagementSeq.NEXTVAL, P_PropertySeq.CURRVAL,
P_TenantSeq.CURRVAL, 'P107', 'Available');


INSERT INTO P_ParkingManagement (P_ParkingID, P_PropertyID, P_TenantID,
P_ParkingSpaceNumber, P_AvailabilityStatus)
VALUES (P_ParkingManagementSeq.NEXTVAL, P_PropertySeq.CURRVAL,
P_TenantSeq.CURRVAL, 'P108', 'Unavailable');


INSERT INTO P_ParkingManagement (P_ParkingID, P_PropertyID, P_TenantID,
P_ParkingSpaceNumber, P_AvailabilityStatus)
VALUES (P_ParkingManagementSeq.NEXTVAL, P_PropertySeq.CURRVAL,
P_TenantSeq.CURRVAL, 'P109', 'Occupied');


INSERT INTO P_ParkingManagement (P_ParkingID, P_PropertyID, P_TenantID,
P_ParkingSpaceNumber, P_AvailabilityStatus)
VALUES (P_ParkingManagementSeq.NEXTVAL, P_PropertySeq.CURRVAL,
P_TenantSeq.CURRVAL, 'P110', 'Reserved');
---


--11 Inserting data into P_FinancialTransactions table using sequences and
P_PropertySeq, P_TenantSeq, P_VendorSeq for P_PropertyID, P_TenantID,
P_VendorID
```

```
INSERT INTO P_FinancialTransactions (P_TransactionID, P_PropertyID, P_TenantID,
P_VendorID, P_TransactionType, P_Amount, P_TransactionDate, P_PaymentStatus)
VALUES (P_FinancialTransactionsSeq.NEXTVAL, P_PropertySeq.CURRVAL,
P_TenantSeq.CURRVAL, P_VendorSeq.CURRVAL, 'Rent', 1500, TO_DATE('2023-01-
05', 'YYYY-MM-DD'), 'Paid');

INSERT INTO P_FinancialTransactions (P_TransactionID, P_PropertyID, P_TenantID,
P_VendorID, P_TransactionType, P_Amount, P_TransactionDate, P_PaymentStatus)
VALUES (P_FinancialTransactionsSeq.NEXTVAL, P_PropertySeq.CURRVAL,
P_TenantSeq.CURRVAL, P_VendorSeq.CURRVAL, 'Maintenance Fee', 200,
TO_DATE('2023-02-10', 'YYYY-MM-DD'), 'Paid');

INSERT INTO P_FinancialTransactions (P_TransactionID, P_PropertyID, P_TenantID,
P_VendorID, P_TransactionType, P_Amount, P_TransactionDate, P_PaymentStatus)
VALUES (P_FinancialTransactionsSeq.NEXTVAL, P_PropertySeq.CURRVAL,
P_TenantSeq.CURRVAL, P_VendorSeq.CURRVAL, 'Utility Bill', 100, TO_DATE('2023-
03-15', 'YYYY-MM-DD'), 'Pending');

INSERT INTO P_FinancialTransactions (P_TransactionID, P_PropertyID, P_TenantID,
P_VendorID, P_TransactionType, P_Amount, P_TransactionDate, P_PaymentStatus)
VALUES (P_FinancialTransactionsSeq.NEXTVAL, P_PropertySeq.CURRVAL,
P_TenantSeq.CURRVAL, P_VendorSeq.CURRVAL, 'Service Charge', 300,
TO_DATE('2023-04-20', 'YYYY-MM-DD'), 'Paid');

INSERT INTO P_FinancialTransactions (P_TransactionID, P_PropertyID, P_TenantID,
P_VendorID, P_TransactionType, P_Amount, P_TransactionDate, P_PaymentStatus)
VALUES (P_FinancialTransactionsSeq.NEXTVAL, P_PropertySeq.CURRVAL,
P_TenantSeq.CURRVAL, P_VendorSeq.CURRVAL, 'Rent', 1500, TO_DATE('2023-05-
25', 'YYYY-MM-DD'), 'Paid');

INSERT INTO P_FinancialTransactions (P_TransactionID, P_PropertyID, P_TenantID,
P_VendorID, P_TransactionType, P_Amount, P_TransactionDate, P_PaymentStatus)
```

```
VALUES (P_FinancialTransactionsSeq.NEXTVAL, P_PropertySeq.CURRVAL,
P_TenantSeq.CURRVAL, P_VendorSeq.CURRVAL, 'Utility Bill', 120, TO_DATE('2023-
06-30', 'YYYY-MM-DD'), 'Pending');


INSERT INTO P_FinancialTransactions (P_TransactionID, P_PropertyID, P_TenantID,
P_VendorID, P_TransactionType, P_Amount, P_TransactionDate, P_PaymentStatus)
VALUES (P_FinancialTransactionsSeq.NEXTVAL, P_PropertySeq.CURRVAL,
P_TenantSeq.CURRVAL, P_VendorSeq.CURRVAL, 'Service Charge', 350,
TO_DATE('2023-07-10', 'YYYY-MM-DD'), 'Pending');


INSERT INTO P_FinancialTransactions (P_TransactionID, P_PropertyID, P_TenantID,
P_VendorID, P_TransactionType, P_Amount, P_TransactionDate, P_PaymentStatus)
VALUES (P_FinancialTransactionsSeq.NEXTVAL, P_PropertySeq.CURRVAL,
P_TenantSeq.CURRVAL, P_VendorSeq.CURRVAL, 'Rent', 1500, TO_DATE('2023-01-
05', 'YYYY-MM-DD'), 'Paid');


INSERT INTO P_FinancialTransactions (P_TransactionID, P_PropertyID, P_TenantID,
P_VendorID, P_TransactionType, P_Amount, P_TransactionDate, P_PaymentStatus)
VALUES (P_FinancialTransactionsSeq.NEXTVAL, P_PropertySeq.CURRVAL,
P_TenantSeq.CURRVAL, P_VendorSeq.CURRVAL, 'Utilities', 500, TO_DATE('2023-02-
10', 'YYYY-MM-DD'), 'Paid');


INSERT INTO P_FinancialTransactions (P_TransactionID, P_PropertyID, P_TenantID,
P_VendorID, P_TransactionType, P_Amount, P_TransactionDate, P_PaymentStatus)
VALUES (P_FinancialTransactionsSeq.NEXTVAL, P_PropertySeq.CURRVAL,
P_TenantSeq.CURRVAL, P_VendorSeq.CURRVAL, 'Maintenance', 200,
TO_DATE('2023-03-15', 'YYYY-MM-DD'), 'Pending');


---


--12 Inserting data into P_UtilityManagement table using sequences and
P_PropertySeq for P_PropertyID
INSERT INTO P_UtilityManagement (P_UtilityID, P_PropertyID, P_UtilityType,
P_Provider, P_BillingDetails)
```

```
VALUES (P_UtilityManagementSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'Electricity',
'UtilityCo', 'Billing details for Electricity');

INSERT INTO P_UtilityManagement (P_UtilityID, P_PropertyID, P_UtilityType,
P_Provider, P_BillingDetails)
VALUES (P_UtilityManagementSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'Water',
'WaterCo', 'Billing details for Water');

INSERT INTO P_UtilityManagement (P_UtilityID, P_PropertyID, P_UtilityType,
P_Provider, P_BillingDetails)
VALUES (P_UtilityManagementSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'Gas',
'GasCo', 'Billing details for Gas');

INSERT INTO P_UtilityManagement (P_UtilityID, P_PropertyID, P_UtilityType,
P_Provider, P_BillingDetails)
VALUES (P_UtilityManagementSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'Internet',
'InternetCo', 'Billing details for Internet');

INSERT INTO P_UtilityManagement (P_UtilityID, P_PropertyID, P_UtilityType,
P_Provider, P_BillingDetails)
VALUES (P_UtilityManagementSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'Phone',
'PhoneCo', 'Billing details for Phone');

INSERT INTO P_UtilityManagement (P_UtilityID, P_PropertyID, P_UtilityType,
P_Provider, P_BillingDetails)
VALUES (P_UtilityManagementSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'Trash',
'TrashCo', 'Billing details for Trash');

INSERT INTO P_UtilityManagement (P_UtilityID, P_PropertyID, P_UtilityType,
P_Provider, P_BillingDetails)
VALUES (P_UtilityManagementSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'Cable TV',
'CableCo', 'Billing details for Cable TV');
```

```
INSERT INTO P_UtilityManagement (P_UtilityID, P_PropertyID, P_UtilityType,
P_Provider, P_BillingDetails)

VALUES (P_UtilityManagementSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'Internet',
'InternetCo', 'Billing details for Internet');


INSERT INTO P_UtilityManagement (P_UtilityID, P_PropertyID, P_UtilityType,
P_Provider, P_BillingDetails)

VALUES (P_UtilityManagementSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'Water',
'WaterCo', 'Billing details for Water');


INSERT INTO P_UtilityManagement (P_UtilityID, P_PropertyID, P_UtilityType,
P_Provider, P_BillingDetails)

VALUES (P_UtilityManagementSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'Gas',
'GasCo', 'Billing details for Gas');


---


--13 Inserting data into P_Maintenance table using sequences and P_PropertySeq,
P_TenantSeq, P_VendorSeq for P_PropertyID, P_TenantID, P_VendorID

INSERT INTO P_Maintenance (P_RequestID, P_PropertyID, P_TenantID, P_VendorID,
P_IssueDescription, P_AssignedStaff, P_Status, P_CompletionDate)

VALUES (P_MaintenanceSeq.NEXTVAL, P_PropertySeq.CURRVAL,
P_TenantSeq.CURRVAL, P_VendorSeq.CURRVAL, 'Plumbing issue in bathroom', 'John
Doe', 'Pending', NULL);


INSERT INTO P_Maintenance (P_RequestID, P_PropertyID, P_TenantID, P_VendorID,
P_IssueDescription, P_AssignedStaff, P_Status, P_CompletionDate)

VALUES (P_MaintenanceSeq.NEXTVAL, P_PropertySeq.CURRVAL,
P_TenantSeq.CURRVAL, P_VendorSeq.CURRVAL, 'Electrical problem in living room',
'Alice Smith', 'In Progress', NULL);


INSERT INTO P_Maintenance (P_RequestID, P_PropertyID, P_TenantID, P_VendorID,
P_IssueDescription, P_AssignedStaff, P_Status, P_CompletionDate)
```

```
VALUES (P_MaintenanceSeq.NEXTVAL, P_PropertySeq.CURRVAL,
P_TenantSeq.CURRVAL, P_VendorSeq.CURRVAL, 'Heating system not working', 'Bob
Johnson', 'Completed', TO_DATE('2023-03-15', 'YYYY-MM-DD'));


INSERT INTO P_Maintenance (P_RequestID, P_PropertyID, P_TenantID, P_VendorID,
P_IssueDescription, P_AssignedStaff, P_Status, P_CompletionDate)
VALUES (P_MaintenanceSeq.NEXTVAL, P_PropertySeq.CURRVAL,
P_TenantSeq.CURRVAL, P_VendorSeq.CURRVAL, 'Leakage in the kitchen sink', 'Emily
Davis', 'Pending', NULL);


INSERT INTO P_Maintenance (P_RequestID, P_PropertyID, P_TenantID, P_VendorID,
P_IssueDescription, P_AssignedStaff, P_Status, P_CompletionDate)
VALUES (P_MaintenanceSeq.NEXTVAL, P_PropertySeq.CURRVAL,
P_TenantSeq.CURRVAL, P_VendorSeq.CURRVAL, 'Broken window in bedroom', 'Mark
Wilson', 'In Progress', NULL);


INSERT INTO P_Maintenance (P_RequestID, P_PropertyID, P_TenantID, P_VendorID,
P_IssueDescription, P_AssignedStaff, P_Status, P_CompletionDate)
VALUES (P_MaintenanceSeq.NEXTVAL, P_PropertySeq.CURRVAL,
P_TenantSeq.CURRVAL, P_VendorSeq.CURRVAL, 'Appliance repair in the kitchen',
'Sarah Brown', 'Completed', TO_DATE('2023-06-10', 'YYYY-MM-DD'));


INSERT INTO P_Maintenance (P_RequestID, P_PropertyID, P_TenantID, P_VendorID,
P_IssueDescription, P_AssignedStaff, P_Status, P_CompletionDate)
VALUES (P_MaintenanceSeq.NEXTVAL, P_PropertySeq.CURRVAL,
P_TenantSeq.CURRVAL, P_VendorSeq.CURRVAL, 'Roof leakage', 'Michael Lee',
'Pending', NULL);


INSERT INTO P_Maintenance (P_RequestID, P_PropertyID, P_TenantID, P_VendorID,
P_IssueDescription, P_AssignedStaff, P_Status, P_CompletionDate)
VALUES (P_MaintenanceSeq.NEXTVAL, P_PropertySeq.CURRVAL,
P_TenantSeq.CURRVAL, P_VendorSeq.CURRVAL, 'Appliance repair in the kitchen',
'Sarah Brown', 'Completed', TO_DATE('2023-06-10', 'YYYY-MM-DD'));
```

```
INSERT INTO P_Maintenance (P_RequestID, P_PropertyID, P_TenantID, P_VendorID, P_IssueDescription, P_AssignedStaff, P_Status, P_CompletionDate)

VALUES (P_MaintenanceSeq.NEXTVAL, P_PropertySeq.CURRVAL, P_TenantSeq.CURRVAL, P_VendorSeq.CURRVAL, 'Roof leakage', 'Michael Lee', 'Pending', NULL);


INSERT INTO P_Maintenance (P_RequestID, P_PropertyID, P_TenantID, P_VendorID, P_IssueDescription, P_AssignedStaff, P_Status, P_CompletionDate)

VALUES (P_MaintenanceSeq.NEXTVAL, P_PropertySeq.CURRVAL, P_TenantSeq.CURRVAL, P_VendorSeq.CURRVAL, 'Plumbing issue in the bathroom', 'John Smith', 'In Progress', NULL);


---


--14 Inserting data into P_AnalysisAndReporting table using sequences and P_PropertySeq for P_PropertyID

INSERT INTO P_AnalysisAndReporting (P_ReportID, P_PropertyID, P_ReportType, P_DateGenerated, P_Insights)

VALUES (P_AnalysisAndReportingSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'Financial', TO_DATE('2023-01-15', 'YYYY-MM-DD'), 'Financial analysis insights for the property');


INSERT INTO P_AnalysisAndReporting (P_ReportID, P_PropertyID, P_ReportType, P_DateGenerated, P_Insights)

VALUES (P_AnalysisAndReportingSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'Tenant Satisfaction', TO_DATE('2023-02-20', 'YYYY-MM-DD'), 'Tenant satisfaction survey results');


INSERT INTO P_AnalysisAndReporting (P_ReportID, P_PropertyID, P_ReportType, P_DateGenerated, P_Insights)

VALUES (P_AnalysisAndReportingSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'Maintenance', TO_DATE('2023-03-25', 'YYYY-MM-DD'), 'Maintenance performance analysis');
```

```
INSERT INTO P_AnalysisAndReporting (P_ReportID, P_PropertyID, P_ReportType,
P_DateGenerated, P_Insights)
VALUES (P_AnalysisAndReportingSeq.NEXTVAL, P_PropertySeq.CURRVAL,
'Occupancy Rate', TO_DATE('2023-04-30', 'YYYY-MM-DD'), 'Occupancy rate insights
for the property');


INSERT INTO P_AnalysisAndReporting (P_ReportID, P_PropertyID, P_ReportType,
P_DateGenerated, P_Insights)
VALUES (P_AnalysisAndReportingSeq.NEXTVAL, P_PropertySeq.CURRVAL,
'Revenue Forecast', TO_DATE('2023-05-05', 'YYYY-MM-DD'), 'Revenue forecast
analysis');


INSERT INTO P_AnalysisAndReporting (P_ReportID, P_PropertyID, P_ReportType,
P_DateGenerated, P_Insights)
VALUES (P_AnalysisAndReportingSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'Market
Trends', TO_DATE('2023-06-10', 'YYYY-MM-DD'), 'Real estate market trends analysis');


INSERT INTO P_AnalysisAndReporting (P_ReportID, P_PropertyID, P_ReportType,
P_DateGenerated, P_Insights)
VALUES (P_AnalysisAndReportingSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'Utilities
Usage', TO_DATE('2023-07-15', 'YYYY-MM-DD'), 'Utilities usage analysis');


INSERT INTO P_AnalysisAndReporting (P_ReportID, P_PropertyID, P_ReportType,
P_DateGenerated, P_Insights)
VALUES (P_AnalysisAndReportingSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'Market
Trends', TO_DATE('2023-06-10', 'YYYY-MM-DD'), 'Real estate market trends analysis');


INSERT INTO P_AnalysisAndReporting (P_ReportID, P_PropertyID, P_ReportType,
P_DateGenerated, P_Insights)
VALUES (P_AnalysisAndReportingSeq.NEXTVAL, P_PropertySeq.CURRVAL, 'Utilities
Usage', TO_DATE('2023-07-15', 'YYYY-MM-DD'), 'Utilities usage analysis');
```

```
INSERT INTO P_AnalysisAndReporting (P_ReportID, P_PropertyID, P_ReportType,
P_DateGenerated, P_Insights)

VALUES (P_AnalysisAndReportingSeq.NEXTVAL, P_PropertySeq.CURRVAL,
'Financial Performance', TO_DATE('2023-08-20', 'YYYY-MM-DD'), 'Financial
performance analysis');


---

--Creating SIMPLE & COMPLEX Views:

-- Simple View combining Property and Tenant Information

CREATE VIEW PropertyTenantView AS

SELECT P.P_PropertyID, P.P_Address, P.P_Type, P.P_Size, T.P_TenantID, T.P_Name
AS TenantName, T.P_LeaseDetails

FROM P_Property P

INNER JOIN P_Tenant T ON P.P_PropertyID = T.P_PropertyID;


-- Complex View involving multiple tables

CREATE VIEW PropertyMaintenanceView AS

SELECT P.P_PropertyID, P.P_Address, P.P_Type, M.P_RequestID,
M.P_IssueDescription, M.P_Status

FROM P_Property P

LEFT JOIN P_Maintenance M ON P.P_PropertyID = M.P_PropertyID;


--calculates the total transaction amounts for each property from the
P_FinancialTransactions table:

CREATE VIEW PropertyFinancialsView AS

SELECT P.P_PropertyID, P.P_Address, SUM(F.P_Amount) AS TotalAmount

FROM P_Property P

LEFT JOIN P_FinancialTransactions F ON P.P_PropertyID = F.P_PropertyID

GROUP BY P.P_PropertyID, P.P_Address;
```

```
--presents detailed lease information along with the respective property details:

CREATE VIEW TenantLeasePropertyView AS

SELECT T.P_TenantID, T.P_Name AS TenantName, T.P_LeaseDetails, P.P_Address,
P.P_Type

FROM P_Tenant T

INNER JOIN P_Property P ON T.P_PropertyID = P.P_PropertyID;


--summarizes the services provided by each vendor and their associated property
details:

CREATE VIEW VendorServiceOverview AS

SELECT V.P_VendorID, V.P_Name AS VendorName, V.P_ServicesProvided,
P.P_Address, P.P_Type

FROM P_Vendor V

INNER JOIN P_Maintenance M ON V.P_VendorID = M.P_VendorID

INNER JOIN P_Property P ON M.P_PropertyID = P.P_PropertyID;
```

The END