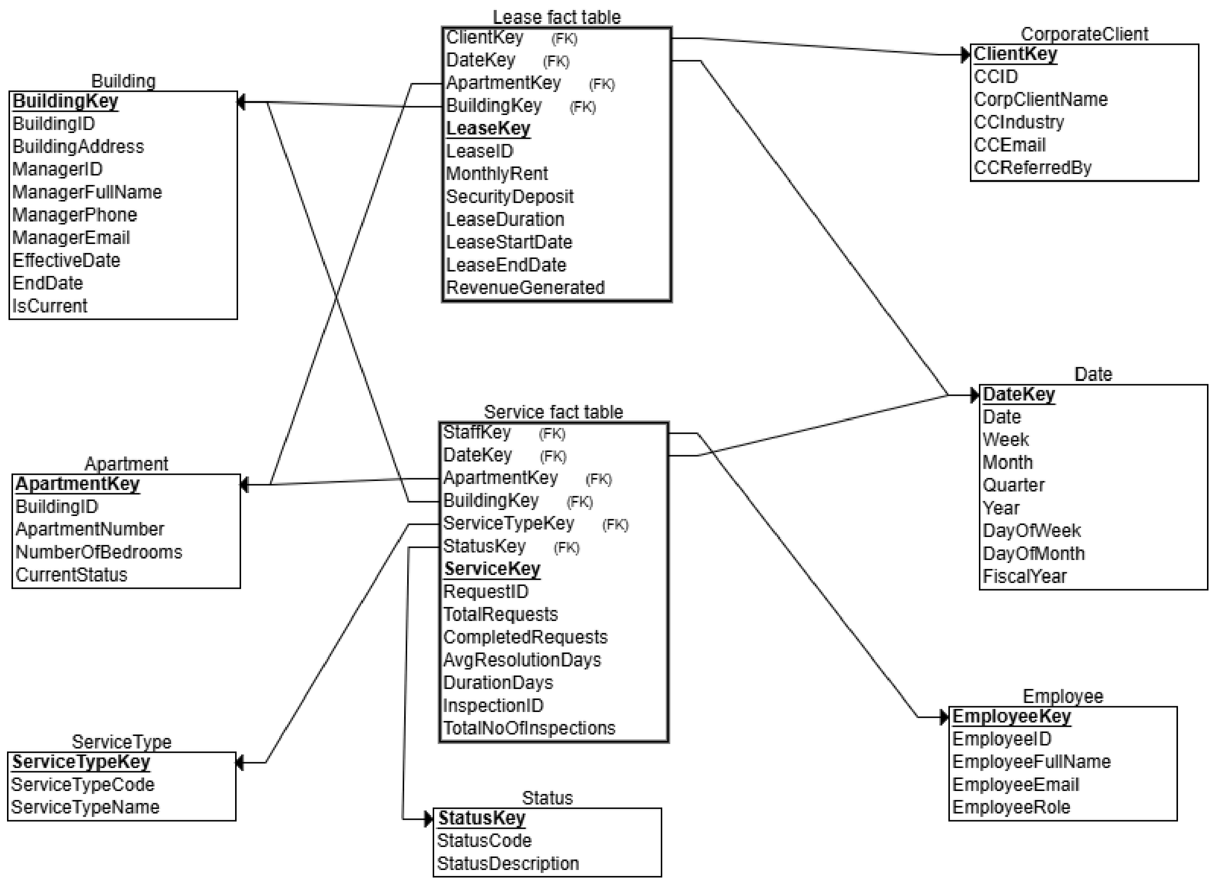


FACT AND DIMENSION TABLES

FACT TABLES:

1. **Fact_Lease:** Tracks rental analysis such as revenue, deposit balances, lease durations and monthly rent balances for each apartment.
2. **Fact_Service:** Tracks details of every inspection, cleaning, and maintenance requests as service type and durations. This allows us to answer questions about Total inspections and total maintenance requests.

DIMENSION TABLES:

1. **Dim_Building:** Stores information about each building and its managers. Uses SCD Type 2 tracking for manager changes

2. **Dim_Apartment:** Stores details about apartments, including the number of bedrooms each apartment has.
3. **Dim_ServiceType:** Stores details about the type of service event, that is whether it is inspection, cleaning, or maintenance request.
4. **Dim_Status:** Stores details of the service status (open, in-progress, completed, or cancelled).
5. **Dim_CorpClient:** Stores details of corporate clients who lease.
6. **Dim_Date:** Stores details of the date-related data.
7. **Dim_Employee:** Combined details of staff members and inspectors information handling maintenance, cleaning tasks and inspections.

<u>UPDATED SQL SCRIPT FOR PROPERTY MANAGEMENT DATABASE</u>

```
DROP DATABASE IF EXISTS Propertymanagement
CREATE DATABASE Propertymanagement;
USE Propertymanagement;

CREATE TABLE Building (
   BuildingID CHAR(5) NOT NULL,
   Street VARCHAR(100) NOT NULL,
   City VARCHAR(50) NOT NULL,
   State VARCHAR(30) NOT NULL,
   ZipCode VARCHAR(10) NOT NULL,
   ManagerIDOversees CHAR(5) NOT NULL,
   PRIMARY KEY (BuildingID)
);

CREATE TABLE Manager (
   ManagerID CHAR(5) NOT NULL,
   MFirstName VARCHAR(50) NOT NULL,
   MLastName VARCHAR(50) NOT NULL,
   MEmail VARCHAR(100) NOT NULL,
   MSalary NUMERIC(10,2) NOT NULL,
   BuildingIDResides CHAR(5),
   PRIMARY KEY (ManagerID),
   UNIQUE (MEmail)
);

-- Alter Building table to add/indicate foreign key
ALTER TABLE Building
ADD CONSTRAINT FK_Building_Manager
FOREIGN KEY (ManagerIDOversees) REFERENCES Manager(ManagerID);

-- Alter Manager table to add/indicate foreign key
ALTER TABLE Manager
ADD CONSTRAINT FK_Manager_Building
```

```sql
FOREIGN KEY (BuildingIDResides) REFERENCES Building(BuildingID);

CREATE TABLE ManagerPhone (
   MPhoneNo VARCHAR(15) NOT NULL,
   ManagerID CHAR(5) NOT NULL,
   PRIMARY KEY (MPhoneNo, ManagerID),
   FOREIGN KEY (ManagerID) REFERENCES Manager(ManagerID)
);

CREATE TABLE Apartment (
   ApartmentNo VARCHAR(10) NOT NULL,
   BuildingID CHAR(5) NOT NULL,
   NoOfBedrooms INTEGER NOT NULL,
   RentalStatus VARCHAR(20) DEFAULT 'Vacant' NOT NULL,
   PRIMARY KEY (ApartmentNo, BuildingID),
   FOREIGN KEY (BuildingID) REFERENCES Building(BuildingID) ON DELETE CASCADE
);

CREATE TABLE StaffMember (
   StaffID CHAR(4) NOT NULL,
   SFirstName VARCHAR(50) NOT NULL,
   SLastName VARCHAR(50) NOT NULL,
   SEmail VARCHAR(100) NOT NULL,
   PRIMARY KEY (StaffID),
   UNIQUE (SEmail)
);

CREATE TABLE Cleans (
   ApartmentNo VARCHAR(10) NOT NULL,
   BuildingID CHAR(5) NOT NULL,
   StaffID CHAR(4) NOT NULL,
   PRIMARY KEY (ApartmentNo, BuildingID, StaffID),
   FOREIGN KEY (ApartmentNo, BuildingID) REFERENCES Apartment(ApartmentNo, BuildingID),
   FOREIGN KEY (StaffID) REFERENCES StaffMember(StaffID)
);

CREATE TABLE Inspector (
   InspectorID CHAR(7) NOT NULL,
   IFirstName VARCHAR(50) NOT NULL,
   ILastName VARCHAR(50) NOT NULL,
   IEmail VARCHAR(100) NOT NULL,
   PRIMARY KEY (InspectorID),
   UNIQUE (IEmail)
);
```

```
CREATE TABLE Inspects (
   BuildingID CHAR(5) NOT NULL,
   InspectorID CHAR(7) NOT NULL,
   Date_Completed DATE NOT NULL,
   Next_Insp_date DATE NOT NULL,
   PRIMARY KEY (BuildingID, InspectorID),
   FOREIGN KEY (BuildingID) REFERENCES Building(BuildingID),
   FOREIGN KEY (InspectorID) REFERENCES Inspector(InspectorID)
);

CREATE TABLE CorporateClient (
   CCID CHAR(5) NOT NULL,
   CCName VARCHAR(100) NOT NULL,
   CCEmail VARCHAR(100) NOT NULL,
   CCIndustry VARCHAR(50) NOT NULL,
   CIDReferredBy CHAR(5),
   PRIMARY KEY (CCID),
   UNIQUE (CCEmail),
   FOREIGN KEY (CIDReferedBy) REFERENCES CorporateClient(CCID)
);

CREATE TABLE MaintenanceRequest (
   RequestID INTEGER NOT NULL,
   RequestDate DATETIME DEFAULT CURRENT_TIMESTAMP NOT NULL,
   Description TEXT(200) NOT NULL,
   Status VARCHAR(20) DEFAULT 'Open' NOT NULL,
   CompletedDate DATE,
   ApartmentNo VARCHAR(10) NOT NULL,
   BuildingID CHAR(5) NOT NULL,
   AssignedStaff          CHAR(4) NOT NULL,
   PRIMARY KEY (RequestID),
   FOREIGN KEY (ApartmentNo, BuildingID) REFERENCES Apartment(ApartmentNo, BuildingID),
   FOREIGN KEY (AssignedStaff) REFERENCES StaffMember(StaffID)
);

CREATE TABLE Lease (
   LeaseID INTEGER NOT NULL,
   LeaseStartDate DATE NOT NULL,
   SecurityDeposit NUMERIC(10, 2) DEFAULT 0.00 NOT NULL,
   MonthlyRent NUMERIC(10, 2) DEFAULT 0.00 NOT NULL,
   LeaseEndDate DATE NOT NULL,
   ApartmentNo VARCHAR(10) NOT NULL,
   BuildingID CHAR(5) NOT NULL,
   CCID CHAR(5) NOT NULL,
   PRIMARY KEY (LeaseID),
   FOREIGN KEY (ApartmentNo, BuildingID) REFERENCES Apartment(ApartmentNo, BuildingID),
   FOREIGN KEY (CCID) REFERENCES CorporateClient(CCID)
);
```

Populating the Database

```
-- Insert data into Manager Table with Null for BuildingIDResides
INSERT INTO Manager (ManagerID, MFirstName, MLastName, MEmail, MSalary,
BuildingIDResides) VALUES
('M0001', 'Sharon', 'Smith', 'ssmith@example.com', 85000.00, NULL),
('M0002', 'Sarah', 'Johnson', 'sjohnson@example.com', 72000.00, NULL),
('M0003', 'Michael', 'Williams', 'mwilliams@example.com', 78500.00, NULL),
('M0004', 'Emma', 'Brown', 'ebrown@example.com', 70000.00, NULL),
('M0005', 'James', 'Jones', 'jjones@example.com', 77000.00, NULL),
('M0006', 'Linda', 'Garcia', 'lgarcia@example.com', 74000.00, NULL),
('M0007', 'Robert', 'Miller', 'rmiller@example.com', 76500.00, NULL),
('M0008', 'Patricia', 'Davis', 'pdavis@example.com', 73000.00, NULL),
('M0009', 'David', 'Rodriguez', 'drodriguez@example.com', 75000.00, NULL),
('M0010', 'Jennifer', 'Martinez', 'jmartinez@example.com', 77200.00, NULL);
```

-- Insert data into Building Table

```
INSERT INTO Building (BuildingID, Street, City, State, ZipCode, ManagerIDOversees) VALUES
('B0001', '123 Sunset Blvd', 'Los Angeles', 'CA', '90028', 'M0001'),
('B0002', '456 Hollywood Ave', 'Los Angeles', 'CA', '90038', 'M0001'),
('B0003', '789 Vine St', 'Los Angeles', 'CA', '90036', 'M0001'),
('B0004', '101 Wilshire Blvd', 'Los Angeles', 'CA', '90017', 'M0010'),
('B0005', '202 Capitol Mall', 'Sacramento', 'CA', '95814', 'M0002'),
('B0006', '303 J St', 'Sacramento', 'CA', '95814', 'M0003'),
('B0007', '404 L St', 'Sacramento', 'CA', '95814', 'M0003'),
('B0008', '505 Michigan Ave', 'Chicago', 'IL', '60611', 'M0005'),
('B0009', '606 Wacker Dr', 'Chicago', 'IL', '60601', 'M0005'),
('B0010', '707 Broadway', 'New York', 'NY', '10003', 'M0004'),
('B0011', '808 5th Ave', 'New York', 'NY', '10021', 'M0006'),
('B0012', '909 Boylston St', 'Boston', 'MA', '02116', 'M0008'),
('B0013', '110 Market St', 'San Francisco', 'CA', '94103', 'M0007'),
('B0014', '211 Union St', 'San Francisco', 'CA', '94111', 'M0007'),
('B0015', '312 Pike St', 'Seattle', 'WA', '98101', 'M0009');
```

-- Update Manager records with BuildingIDResides values

```
UPDATE Manager SET BuildingIDResides = 'B0002' WHERE ManagerID = 'M0001';
UPDATE Manager SET BuildingIDResides = 'B0005' WHERE ManagerID = 'M0002';
UPDATE Manager SET BuildingIDResides = 'B0007' WHERE ManagerID = 'M0003';
UPDATE Manager SET BuildingIDResides = 'B0010' WHERE ManagerID = 'M0004';
UPDATE Manager SET BuildingIDResides = 'B0008' WHERE ManagerID = 'M0005';
UPDATE Manager SET BuildingIDResides = 'B0011' WHERE ManagerID = 'M0006';
UPDATE Manager SET BuildingIDResides = 'B0013' WHERE ManagerID = 'M0007';
UPDATE Manager SET BuildingIDResides = 'B0012' WHERE ManagerID = 'M0008';
UPDATE Manager SET BuildingIDResides = 'B0015' WHERE ManagerID = 'M0009';
UPDATE Manager SET BuildingIDResides = 'B0004' WHERE ManagerID = 'M0010';
```

-- Insert data into ManagerPhone table

```
INSERT INTO ManagerPhone (MPhoneNo, ManagerID) VALUES
('212-555-1001', 'M0001'),
('212-555-1002', 'M0001'),
('617-555-2001', 'M0002'),
('312-555-3001', 'M0003'),
('415-555-4001', 'M0004'),
('206-555-5001', 'M0005'),
('206-555-5002', 'M0005'),
('305-555-6002', 'M0006'),
('323-555-7001', 'M0007'),
('323-555-7002', 'M0007'),
('303-555-8001', 'M0008'),
('212-555-9001', 'M0009'),
('617-555-0001', 'M0010');
```

-- Insert data into Apartment Table

```
INSERT INTO Apartment (ApartmentNo, BuildingID, NoOfBedrooms, RentalStatus) VALUES
-- Los Angeles Buildings
('101', 'B0001', 2, 'Occupied'),
('102', 'B0001', 1, 'Vacant'),
('201', 'B0001', 3, 'Occupied'),
('101', 'B0002', 2, 'Occupied'),
('102', 'B0002', 2, 'Vacant'),
('201', 'B0002', 1, 'Occupied'),
('101', 'B0003', 1, 'Occupied'),
('201', 'B0003', 2, 'Occupied'),
('301', 'B0003', 3, 'Vacant'),
('101', 'B0004', 1, 'Occupied'),
('201', 'B0004', 2, 'Occupied'),
-- Sacramento Buildings
('101', 'B0005', 1, 'Vacant'),
('201', 'B0005', 2, 'Occupied'),
('101', 'B0006', 2, 'Occupied'),
('201', 'B0006', 3, 'Occupied'),
('101', 'B0007', 1, 'Occupied'),
('201', 'B0007', 2, 'Vacant'),
-- Chicago Buildings
('101', 'B0008', 2, 'Occupied'),
('201', 'B0008', 2, 'Occupied'),
('101', 'B0009', 1, 'Occupied'),
```

```sql
('201', 'B0009', 3, 'Vacant'),
-- New York Buildings
('101', 'B0010', 2, 'Occupied'),
('201', 'B0010', 1, 'Occupied'),
('101', 'B0011', 3, 'Occupied'),
('201', 'B0011', 2, 'Vacant'),
-- Boston Building
('101', 'B0012', 1, 'Occupied'),
('201', 'B0012', 2, 'Occupied'),
-- San Francisco Buildings
('101', 'B0013', 2, 'Occupied'),
('201', 'B0013', 3, 'Vacant'),
('101', 'B0014', 1, 'Occupied'),
('201', 'B0014', 2, 'Occupied'),
-- Seattle Building
('101', 'B0015', 2, 'Occupied'),
('201', 'B0015', 1, 'Vacant');
```

-- Insert data into Staff Members Table

```sql
INSERT INTO StaffMember (StaffID, SFirstName, SLastName, SEmail) VALUES
('S001', 'Davida', 'Clark', 'dclark@example.com'),
('S002', 'Laura', 'Lewis', 'llewis@example.com'),
('S003', 'Kevin', 'Lee', 'klee@example.com'),
('S004', 'Maria', 'Walker', 'mwalker@example.com'),
('S005', 'Jammy', 'Hall', 'jhall@example.com'),
('S006', 'Susan', 'Allen', 'sallen@example.com'),
('S007', 'Brown', 'Young', 'byoung@example.com'),
('S008', 'Nonye', 'King', 'nking@example.com'),
('S009', 'Markson', 'Wright', 'mwright@example.com'),
('S010', 'Karen', 'Lopez', 'klopez@example.com');
```

-- Insert data into Inspector table

```sql
INSERT INTO Inspector (InspectorID, IFirstName, ILastName, IEmail) VALUES
('INSP001', 'Jonathan', 'Peterson', 'jpeterson@inspector.gov'),
('INSP002', 'Helen', 'Baker', 'hbaker@inspector.gov'),
('INSP003', 'Richard', 'Carter', 'rcarter@inspector.gov'),
('INSP004', 'Denzel', 'Evans', 'devans@inspector.gov'),
('INSP005', 'Joseph', 'Foster', 'jfoster@inspector.gov'),
('INSP006', 'Carol', 'Green', 'cgreen@inspector.gov'),
('INSP007', 'Edward', 'Harris', 'eharris@inspector.gov'),
('INSP008', 'Logan', 'Irving', 'lirving@inspector.gov'),
('INSP009', 'Steven', 'Jackson', 'sjackson@inspector.gov'),
('INSP010', 'Sandra', 'Kelly', 'skelly@inspector.gov');
```

-- Insert data into Cleans table

```
INSERT INTO Cleans (ApartmentNo, BuildingID, StaffID) VALUES
-- Los Angeles Buildings (S001 and S002)
('101', 'B0001', 'S001'),
('102', 'B0001', 'S001'),
('201', 'B0001', 'S001'),
('101', 'B0002', 'S001'),
('102', 'B0002', 'S001'),
('201', 'B0002', 'S002'),
('101', 'B0003', 'S002'),
('201', 'B0003', 'S002'),
('301', 'B0003', 'S002'),
('101', 'B0004', 'S002'),
('201', 'B0004', 'S002'),
-- Sacramento Buildings (S003 and S004)
('101', 'B0005', 'S003'),
('201', 'B0005', 'S003'),
('101', 'B0006', 'S003'),
('201', 'B0006', 'S003'),
('101', 'B0007', 'S004'),
('201', 'B0007', 'S004'),
-- Chicago Buildings (S005)
('101', 'B0008', 'S005'),
('201', 'B0008', 'S005'),
('101', 'B0009', 'S005'),
('201', 'B0009', 'S005'),
-- New York Buildings (S006)
('101', 'B0010', 'S006'),
('201', 'B0010', 'S006'),
('101', 'B0011', 'S006'),
('201', 'B0011', 'S006'),
-- Boston Building (S007)
('101', 'B0012', 'S007'),
('201', 'B0012', 'S007'),
-- San Francisco Buildings (S008)
('101', 'B0013', 'S008'),
('201', 'B0013', 'S008'),
('101', 'B0014', 'S008'),
('201', 'B0014', 'S008'),
-- Seattle Building (S009 and S010)
('101', 'B0015', 'S009'),
('201', 'B0015', 'S010');
```

-- Insert data into Inspects table

```
INSERT INTO Inspects (BuildingID, InspectorID, Date_Completed, Next_Insp_date) VALUES
-- Los Angeles Buildings
```

```
('B0001', 'INSP001', '2024-01-15', '2025-01-15'),
('B0001', 'INSP002', '2024-06-15', '2025-06-15'),
('B0002', 'INSP001', '2024-01-20', '2025-01-20'),
('B0003', 'INSP002', '2024-02-10', '2025-02-10'),
('B0004', 'INSP002', '2024-02-15', '2025-02-15'),
-- Sacramento Buildings
('B0005', 'INSP003', '2024-02-22', '2025-02-22'),
('B0005', 'INSP004', '2024-06-27', '2025-06-27'),
('B0006', 'INSP003', '2024-03-01', '2025-03-01'),
('B0007', 'INSP004', '2024-03-10', '2025-03-10'),
-- Chicago Buildings
('B0008', 'INSP005', '2024-03-18', '2025-03-18'),
('B0009', 'INSP005', '2024-04-02', '2025-04-02'),
-- New York Buildings
('B0010', 'INSP006', '2024-04-12', '2025-04-12'),
('B0010', 'INSP007', '2024-07-10', '2025-07-10'),
('B0011', 'INSP007', '2024-04-19', '2025-04-19'),
-- Boston Building
('B0012', 'INSP008', '2024-05-03', '2025-05-03'),
-- San Francisco Buildings
('B0013', 'INSP009', '2024-05-15', '2025-05-15'),
('B0014', 'INSP009', '2024-05-22', '2025-05-22'),
-- Seattle Building
('B0015', 'INSP010', '2024-06-01', '2025-06-01');
```

-- Insert data into CorporateClient Table

```
-- First, insert clients with no referrals
INSERT INTO CorporateClient (CCID, CCName, CCEmail, CCIndustry, CIDReferedBy) VALUES
('CC001', 'TechNova Inc', 'contact@technova.com', 'Technology', NULL),
('CC002', 'Global Logistics', 'info@globallogistics.com', 'Transportation', NULL),
('CC003', 'Pacific Finance', 'corporate@pacificfinance.com', 'Finance', NULL),
('CC004', 'West Coast Media', 'business@westcoastmedia.com', 'Entertainment', NULL),
('CC005', 'Health Solutions', 'inquiries@healthsolutions.com', 'Healthcare', NULL);
-- Then, insert clients with referrals
INSERT INTO CorporateClient (CCID, CCName, CCEmail, CCIndustry, CIDReferedBy) VALUES
('CC006', 'EcoSystems Design', 'contact@ecosystems.com', 'Environment', 'CC001'),
('CC007', 'Urban Architecture', 'info@urbanarchitecture.com', 'Construction', 'CC003'),
('CC008', 'DataStream Analytics', 'sales@datastream.com', 'Data Science', 'CC001'),
('CC009', 'Creative Solutions', 'inquiries@creativesolutions.com', 'Marketing', 'CC004'),
('CC010', 'Fresh Organics', 'business@freshorganics.com', 'Food Production', 'CC002'),
('CC011', 'Medical Innovations', 'info@medicalinnovations.com', 'Biotechnology', 'CC005'),
('CC012', 'Space Systems Corp', 'contact@spacesystems.com', 'Aerospace', 'CC008'),
('CC013', 'Golden State Insurance', 'info@gsinsurance.com', 'Insurance', 'CC003'),
('CC014', 'Quantum Computing', 'business@quantumcomputing.com', 'Technology', 'CC001'),
('CC015', 'Renewable Energy Group', 'contact@renewableenergy.com', 'Energy', 'CC006');
```

-- Insert data into Lease table

```
INSERT INTO Lease (LeaseID, LeaseStartDate, SecurityDeposit, MonthlyRent, LeaseEndDate,
ApartmentNo, BuildingID, CCID) VALUES
-- Los Angeles Buildings
(5001, '2024-01-01', 3700.00, 2500.00, '2025-01-01', '101', 'B0001', 'CC001'),
(5002, '2024-01-15', 4200.00, 2800.00, '2025-01-15', '201', 'B0001', 'CC002'),
(5003, '2024-02-01', 3000.00, 2400.00, '2025-02-01', '101', 'B0002', 'CC003'),
(5004, '2024-02-15', 3500.00, 2200.00, '2025-02-15', '201', 'B0002', 'CC004'),
(5005, '2024-03-01', 2500.00, 2000.00, '2025-03-01', '101', 'B0003', 'CC005'),
(5006, '2024-03-15', 3200.00, 2600.00, '2025-03-15', '201', 'B0003', 'CC006'),
(5007, '2024-04-01', 2800.00, 2300.00, '2025-04-01', '101', 'B0004', 'CC007'),
(5008, '2024-04-15', 3200.00, 2500.00, '2025-04-15', '201', 'B0004', 'CC008'),
-- Sacramento Buildings
(5009, '2024-05-01', 3500.00, 2100.00, '2025-05-01', '201', 'B0005', 'CC009'),
(5010, '2024-05-15', 3700.00, 2300.00, '2025-05-15', '101', 'B0006', 'CC010'),
(5011, '2024-06-01', 4100.00, 2600.00, '2025-06-01', '201', 'B0006', 'CC011'),
(5012, '2024-06-15', 3600.00, 2200.00, '2025-06-15', '101', 'B0007', 'CC012'),
-- Chicago Buildings
(5013, '2024-07-01', 2900.00, 2400.00, '2025-07-01', '101', 'B0008', 'CC013'),
(5014, '2024-07-15', 3200.00, 2700.00, '2025-07-15', '201', 'B0008', 'CC014'),
(5015, '2024-08-01', 2800.00, 2300.00, '2025-08-01', '101', 'B0009', 'CC015'),
-- New York Buildings
(5016, '2024-08-15', 3500.00, 3000.00, '2025-08-15', '101', 'B0010', 'CC001'),
(5017, '2024-09-01', 3700.00, 3200.00, '2025-09-01', '201', 'B0010', 'CC002'),
(5018, '2024-09-15', 3800.00, 3300.00, '2025-09-15', '101', 'B0011', 'CC003'),
-- Boston Building
(5019, '2024-10-01', 3300.00, 2800.00, '2025-10-01', '101', 'B0012', 'CC004'),
(5020, '2024-10-15', 3500.00, 3000.00, '2025-10-15', '201', 'B0012', 'CC005'),
-- San Francisco Buildings
(5021, '2024-11-01', 3900.00, 3400.00, '2025-11-01', '101', 'B0013', 'CC006'),
(5022, '2024-11-15', 3600.00, 3100.00, '2025-11-15', '101', 'B0014', 'CC007'),
(5023, '2024-12-01', 3700.00, 3200.00, '2025-12-01', '201', 'B0014', 'CC008'),
-- Seattle Building
(5024, '2024-12-15', 3400.00, 2900.00, '2025-12-15', '101', 'B0015', 'CC009');
```

-- Insert data into Maintenance Request Table

```
INSERT INTO MaintenanceRequest (RequestID, RequestDate, Description, Status,
CompletedDate, ApartmentNo, BuildingID, AssignedStaff) VALUES
(701, '2023-01-10', 'Leaking kitchen sink', 'Completed', '2023-01-12', '101', 'B0001', 'S001'),
(702, '2023-01-20', 'Broken bathroom light fixture', 'Completed', '2023-01-23', '201', 'B0001',
'S001'),
(703, '2023-02-05', 'Thermostat not working', 'Completed', '2023-02-08', '101', 'B0002', 'S001'),
(704, '2024-02-15', 'Window won\'t close properly', 'Completed', '2024-02-16', '101', 'B0003',
'S002'),
(705, '2024-03-01', 'Dishwasher not draining', 'Completed', '2024-03-03', '201', 'B0003', 'S002'),
(706, '2024-03-15', 'Ceiling fan makes noise', 'Completed', '2024-03-19', '101', 'B0004', 'S002'),
(707, '2024-04-01', 'Smoke detector battery replacement', 'Completed', '2024-04-01', '201',
'B0005', 'S003'),
(708, '2024-04-15', 'Front door lock jammed', 'Completed', '2024-04-17', '101', 'B0006', 'S003'),
(709, '2024-05-01', 'Air conditioning not cooling', 'Completed', '2024-05-03', '201', 'B0006',
'S003'),
(710, '2024-05-15', 'Garbage disposal not working', 'Completed', '2024-05-17', '101', 'B0007',
'S004'),
(711, '2025-04-01', 'Shower drain clogged', 'In Progress', NULL, '101', 'B0008', 'S005'),
(712, '2025-04-10', 'Refrigerator temperature issues', 'Open', NULL, '201', 'B0008', 'S005'),
(713, '2025-03-20', 'Closet door off track', 'Open', NULL, '101', 'B0009', 'S005'),
(714, '2025-04-01', 'Water heater leaking', 'Open', NULL, '101', 'B0010', 'S006'),
(715, '2025-05-02', 'Stove burner not lighting', 'Open', NULL, '101', 'B0011', 'S006');

-- Update completion dates for tracking resolution times
UPDATE MaintenanceRequest
  SET Status = 'Completed',
      CompletedDate = '2025-04-03'
WHERE RequestID = 711;

UPDATE MaintenanceRequest
  SET Status = 'Completed',
      CompletedDate = '2025-04-07'
WHERE RequestID = 713;

-- Adding historical maintenance request data for trend analysis
-- Including data from multiple years to support year-to-year comparisons
INSERT INTO MaintenanceRequest (RequestID, RequestDate, Description, Status,
CompletedDate, ApartmentNo, BuildingID, AssignedStaff) VALUES
-- Historical  data for Maintenance Request from 2023
(306, '2023-06-15', 'Roof leak in attic', 'Completed', '2023-06-18', '301', 'B0003', 'S004'),
(307, '2023-07-02', 'Broken sidewalk light', 'Completed', '2023-07-04', '101', 'B0007', 'S002'),
(308, '2023-08-20', 'Pest control requested', 'Completed', '2023-08-22', '201', 'B0010', 'S005'),
(309, '2023-09-10', 'Elevator malfunction', 'Completed', '2023-09-12', '101', 'B0005', 'S001'),
(310, '2023-10-05', 'Smoke alarm false alarm', 'Completed', '2023-10-06', '101', 'B0006', 'S006'),
-- Historical data from 2024
(421, '2024-06-15', 'Garage door sensor failure', 'Completed', '2024-06-17', '201', 'B0008',
'S007'),
(422, '2024-07-01', 'Mold in bathroom corner', 'Completed', '2024-07-03', '101', 'B0009', 'S008'),
(423, '2024-08-20', 'Air vent clogged', 'Completed', '2024-08-21', '102', 'B0002', 'S009'),
(424, '2024-09-10', 'Hot water inconsistent', 'Completed', '2024-09-12', '201', 'B0004', 'S010'),
(425, '2024-11-05', 'Carpet stains cleaned', 'Completed', '2024-11-06', '101', 'B0001', 'S001');

-- Historical data added to lease to enable rental trends
```

```
 INSERT INTO Lease (LeaseID, LeaseStartDate, SecurityDeposit, MonthlyRent, LeaseEndDate,
ApartmentNo, BuildingID, CCID) VALUES
  -- 2022 leases
  (4025, '2022-02-01', 1200.00, 1800.00, '2023-01-31', '101', 'B0001', 'CC001'),
  (4026, '2022-03-15', 1500.00, 2000.00, '2023-03-14', '201', 'B0003', 'CC003'),
  (4027, '2022-06-01', 1000.00, 1600.00, '2023-05-31', '101', 'B0005', 'CC005'),
  -- 2023 leases
  (4028, '2023-01-10', 1100.00, 1700.00, '2024-01-09', '102', 'B0002', 'CC002'),
  (4029, '2023-04-01', 1300.00, 1900.00, '2024-03-31', '201', 'B0004', 'CC006'),
  (4030, '2023-07-15', 1400.00, 2000.00, '2024-07-14', '201', 'B0006', 'CC008');

-- Add a few more recent maintenance requests for trend analysis
INSERT INTO MaintenanceRequest (RequestID, RequestDate, Description, Status,
CompletedDate, ApartmentNo, BuildingID, AssignedStaff) VALUES
(716, '2025-04-15', 'Toilet constantly running', 'Open', NULL, '201', 'B0012', 'S007'),
(717, '2025-04-18', 'Broken kitchen drawer', 'Open', NULL, '101', 'B0013', 'S008'),
(718, '2025-04-20', 'Ceiling paint peeling', 'Open', NULL, '201', 'B0014', 'S008'),
(719, '2025-04-22', 'Noisy air conditioner', 'Open', NULL, '101', 'B0015', 'S009');
```

CREATING DIMENSION AND FACT TABLES

```
-- Creating Dimension and Fact Tables
-- Some data refers to original tables except surrogate keys or stated otherwise
-- Create Building dimension table
CREATE TABLE Dim_Building (
    BuildingKey INT AUTO_INCREMENT PRIMARY KEY,           -- Surrogate Key
    BuildingID CHAR(5) NOT NULL,
    BuildingAddress VARCHAR(255) NOT NULL,          -- Building street + city + state + zipcode
    ManagerID CHAR(5) NOT NULL,
    ManagerFullName VARCHAR(100) NOT NULL,       -- Manager MFirstName + MLastName
    ManagerPhone VARCHAR(20) NOT NULL,
    ManagerEmail VARCHAR(100) NOT NULL,
    EffectiveDate DATE NOT NULL,       -- Start date for SCD tracking
    EndDate DATE,        -- End date for SCD tracking
    IsCurrent BOOLEAN NOT NULL       -- Indicator for the current record
);

-- Create Apartment dimension table
CREATE TABLE Dim_Apartment (
    ApartmentKey INT AUTO_INCREMENT PRIMARY KEY,       -- Surrogate Key
    BuildingID CHAR(5) NOT NULL,
    ApartmentNumber VARCHAR(10) NOT NULL,
    NumberOfBedrooms INT,
    CurrentStatus VARCHAR(20)          -- Rental status
);
```

-- Create CorporateClient dimension table
```
CREATE TABLE Dim_CorporateClient (
   ClientKey INT AUTO_INCREMENT PRIMARY KEY,    -- Surrogate Key
   CCID CHAR(5) NOT NULL,
   CorpClientName VARCHAR(100) NOT NULL,        -- CCName on CorporateClient Table
   CCIndustry VARCHAR(50),
   CCEmail VARCHAR(100),
   CCReferredBy CHAR(5)        -- CIDReferredBy on CorporateClient Table
);
```

-- Create Date dimension table
```
CREATE TABLE Dim_Date (
   DateKey INT AUTO_INCREMENT PRIMARY KEY,      -- Surrogate Key
   Date DATE NOT NULL,         -- Actual Date
   Week INT NOT NULL,          -- Week of Year
   Month INT NOT NULL,         -- Month of Year
   Quarter INT NOT NULL,       -- Quarter of Year
   Year INT NOT NULL,-- Year
   DayOfWeek INT NOT NULL,             -- Day name (e.g., Monday)
   DayOfMonth INT NOT NULL,            -- Day of the month(e.g., 1,2,3...)
   FiscalYear INT NOT NULL             -- Fiscal Year
);
```

-- Create Employee dimension table
```
CREATE TABLE Dim_Employee (                     -- Combines staff and inspectors details
   EmployeeKey INT AUTO_INCREMENT PRIMARY KEY,              -- Surrogate Key
   EmployeeID CHAR(7) NOT NULL,    -- Staff/Inspector ID
   EmployeeFullName VARCHAR(100) NOT NULL,              -- Staff/Inspector full name
   EmployeeEmail VARCHAR(100) NOT NULL,-- Staff/Inspector email
   EmployeeRole VARCHAR(50) NOT NULL              -- Staff or Inspector
);
```

-- Create ServiceType dimension table
```
CREATE TABLE Dim_ServiceType (
   ServiceTypeKey INT AUTO_INCREMENT PRIMARY KEY,     -- Surrogate Key
   ServiceTypeCode VARCHAR(50) NOT NULL,          -- Generated for service type
   ServiceTypeName VARCHAR(100) NOT NULL          -- Inspection, Cleaning, Maintenance
);
```

-- Create Status dimension table
```
CREATE TABLE Dim_Status (
   StatusKey INT AUTO_INCREMENT PRIMARY KEY,   -- Surrogate Key
   StatusCode VARCHAR(50) NOT NULL,        -- Generated for status
   StatusDescription VARCHAR(255) NOT NULL              -- Open, In-Progress, Completed,
Cancelled
);

-- FACT TABLES
-- Create Lease fact table
CREATE TABLE Fact_Lease (
   LeaseKey INT AUTO_INCREMENT PRIMARY KEY,    -- Surrogate key
   ClientKey INT NOT NULL,
   DateKey INT NOT NULL,
   ApartmentKey INT NOT NULL,
   BuildingKey INT NOT NULL,
   LeaseID INT NOT NULL,       -- Original LeaseID
   MonthlyRent DECIMAL(10, 2) NOT NULL,
   SecurityDeposit DECIMAL(10, 2) NOT NULL,
   LeaseDuration INT NOT NULL,
   LeaseStartDate DATE NOT NULL,
   LeaseEndDate DATE NOT NULL,
   RevenueGenerated DECIMAL(10, 2) NOT NULL,     -- Added for revenue analysis
   FOREIGN KEY (ClientKey) REFERENCES Dim_CorporateClient(ClientKey),
   FOREIGN KEY (DateKey) REFERENCES Dim_Date(DateKey),
   FOREIGN KEY (ApartmentKey) REFERENCES Dim_Apartment(ApartmentKey),
   FOREIGN KEY (BuildingKey) REFERENCES Dim_Building(BuildingKey)
);

-- Create Service fact table
CREATE TABLE Fact_Service (
   ServiceKey INT AUTO_INCREMENT PRIMARY KEY,            -- Surrogate Key
   StaffKey INT NOT NULL,
   DateKey INT NOT NULL,
   ApartmentKey INT NOT NULL,
   BuildingKey INT NOT NULL,
   StatusKey INT NOT NULL,
   ServiceTypeKey INT NOT NULL,
   RequestID INT NOT NULL,
   TotalRequests INT NOT NULL,                 -- Count metric
   CompletedRequests INT NOT NULL,             -- Count metric
   AvgResolutionDays DECIMAL(5, 2) NOT NULL,       -- Calculated metric
   DurationDays INT NOT NULL,          -- Calculated metric
   Next_Insp_Date DATE NOT NULL,
   TotalNoOfInspections INT NOT NULL,                  -- Count metrics
   FOREIGN KEY (StaffKey) REFERENCES Dim_Employee(EmployeeKey),
   FOREIGN KEY (DateKey) REFERENCES Dim_Date(DateKey),
   FOREIGN KEY (ApartmentKey) REFERENCES Dim_Apartment(ApartmentKey),
   FOREIGN KEY (BuildingKey) REFERENCES Dim_Building(BuildingKey),
   FOREIGN KEY (StatusKey) REFERENCES Dim_Status(StatusKey),
   FOREIGN KEY (ServiceTypeKey) REFERENCES Dim_ServiceType(ServiceTypeKey)
);
```

1. (10 points) Use the ETL process for data extraction, transformation, and loading data into fact and dimension tables as shown below:
   Extraction from the operational database (created in Project 1) should focus on relevant data (leases, inspections, etc.).
   Transformation might include cleaning, summarizing, or aggregating data (e.g., converting date formats or combining similar records).
   Loading into the data warehouse involves populating fact and dimension tables.
   Sample data should represent multiple years of data for historical analysis, as this is a typical use case in data warehouses. Add additional data if necessary.

– ETL DIM_DATE:

```
DELIMITER $$
CREATE PROCEDURE PopulateDateDimension()
```

```
BEGIN
   DECLARE start_date DATE DEFAULT '2022-01-01';  -- Start Date can be adjusted as needed
   DECLARE end_date   DATE DEFAULT '2025-12-31'; -- End Date can be adjusted as needed
   DECLARE curr_date  DATE DEFAULT start_date;

   WHILE curr_date <= end_date DO
      INSERT IGNORE INTO `Dim_Date` (
         `Date`,`Week`,`Month`,`Quarter`,`Year`,
         `DayOfWeek`,`DayOfMonth`,`FiscalYear`
      ) VALUES (
         curr_date,
         WEEKOFYEAR(curr_date),
         MONTH(curr_date),
         QUARTER(curr_date),
         YEAR(curr_date),
         DAYOFWEEK(curr_date),
         DAYOFMONTH(curr_date),
         CASE WHEN MONTH(curr_date)>=7    -- Assuming the fiscal year starts in July
             THEN YEAR(curr_date)+1
             ELSE YEAR(curr_date)
         END
      );
      SET curr_date = DATE_ADD(curr_date, INTERVAL 1 DAY);          -- Increment by 1 day
   END WHILE;
END$$
DELIMITER ;
```

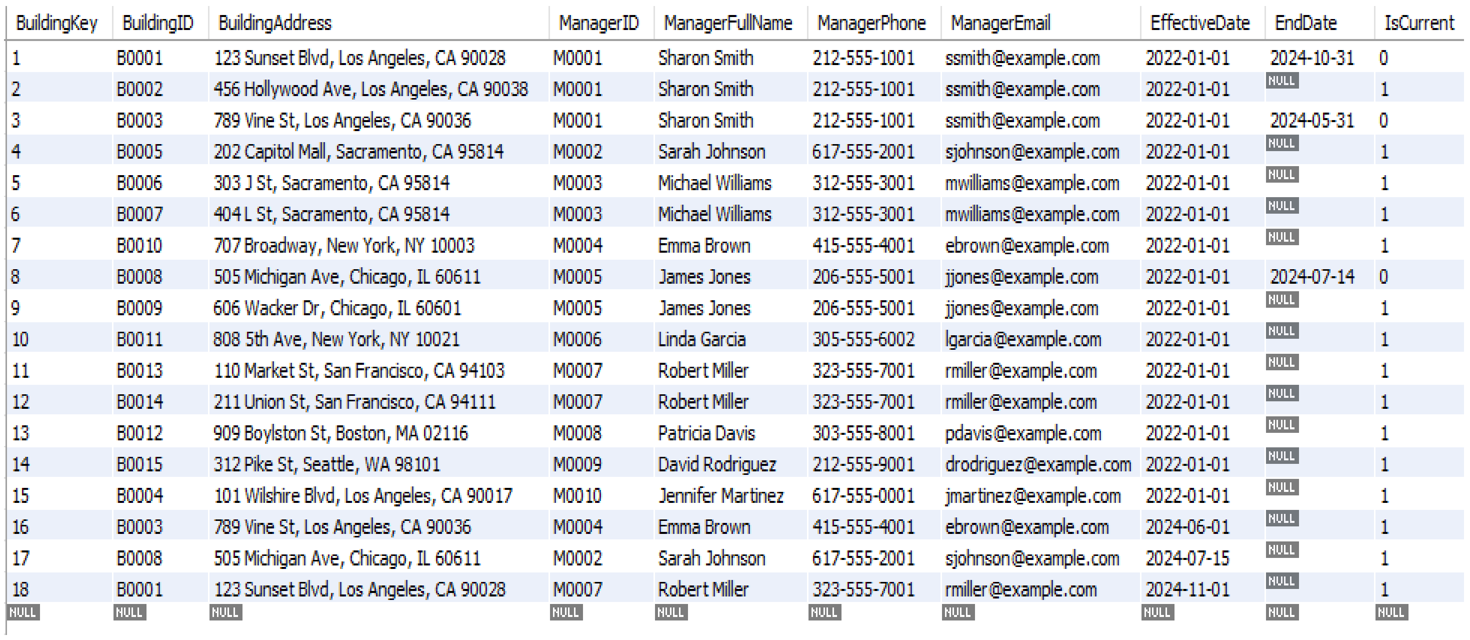
CALL PopulateDateDimension();

– ETL DIM_BUILDING AND IMPLEMENTING SCD (Slowly Changing Dimension):

```
INSERT INTO Dim_Building (BuildingID, BuildingAddress, ManagerID, ManagerFullName,
ManagerPhone, ManagerEmail, EffectiveDate, EndDate, IsCurrent)
SELECT
   b.BuildingID,
   CONCAT(b.Street, ', ', b.City, ', ', b.State, ' ', b.ZipCode) AS BuildingAddress,
   m.ManagerID,
   CONCAT(m.MFirstName, ' ', m.MLastName) AS ManagerFullName,
   -- Get the first phone number for each manager
   (SELECT mp.MPhoneNo FROM ManagerPhone mp WHERE mp.ManagerID = m.ManagerID LIMIT
1) AS ManagerPhone,
   m.MEmail,
    '2022-01-01' AS EffectiveDate, -- Starting point for our data
    NULL AS EndDate,
    TRUE AS IsCurrent
FROM Building b
JOIN Manager m ON b.ManagerIDOversees = m.ManagerID;

-- Create Stored Procedures to Simulate SCD Type 2 Changes for Building-Manager Relationship
-- ==============================================
-- Procedure to update Building-Manager relationships with SCD Type 2
DELIMITER $$
CREATE PROCEDURE UpdateBuildingManager(
  IN p_BuildingID    CHAR(5),
  IN p_NewManagerID  CHAR(5),
  IN p_EffectiveDate DATE
)
BEGIN
  DECLARE vDimKey         INT;
  DECLARE vManagerFullName VARCHAR(100);
  DECLARE vManagerPhone    VARCHAR(20);
  DECLARE vManagerEmail    VARCHAR(100);

  -- 1) grab the current row’s primary key
  SELECT BuildingKey INTO vDimKey FROM Dim_Building
  WHERE BuildingID = p_BuildingID
    AND IsCurrent  = TRUE
  LIMIT 1;

  -- 2) pull new manager info
  SELECT CONCAT(MFirstName,' ',MLastName),
    (SELECT MPhoneNo FROM ManagerPhone WHERE ManagerID = p_NewManagerID ORDER BY
MPhoneNo
     LIMIT 1),
    MEmail
  INTO vManagerFullName, vManagerPhone, vManagerEmail
  FROM Manager
  WHERE ManagerID = p_NewManagerID;

  -- 3) “close” the old dimension row by PK
  UPDATE Dim_Building
     SET EndDate   = DATE_SUB(p_EffectiveDate, INTERVAL 1 DAY),
        IsCurrent = FALSE
   WHERE BuildingKey = vDimKey;

 -- 4) insert the new current row
 INSERT INTO Dim_Building (BuildingID, BuildingAddress, ManagerID, ManagerFullName,
ManagerPhone, ManagerEmail, EffectiveDate, EndDate, IsCurrent)
 SELECT
    b.BuildingID, CONCAT(b.Street, ', ', b.City, ', ', b.State, ' ', b.ZipCode),
    m.ManagerID, vManagerFullName, vManagerPhone, vManagerEmail, p_EffectiveDate,
    NULL, TRUE
  FROM Building AS b
  JOIN Manager  AS m ON m.ManagerID = p_NewManagerID
  WHERE b.BuildingID = p_BuildingID;

  -- 5) sync the OLTP table
  UPDATE Building
     SET ManagerIDOversees = p_NewManagerID
   WHERE BuildingID = p_BuildingID;
END$$
DELIMITER ;

-- Simulate a manager change to demonstrate SCD Type 2
CALL UpdateBuildingManager('B0003','M0004','2024-06-01');
CALL UpdateBuildingManager('B0008','M0002','2024-07-15');
CALL UpdateBuildingManager('B0001','M0007','2024-11-01');
```

DIM_BUILDING TABLE

– Shows the SCD of the Building-Manager dimension table



BuildingKey	BuildingID	BuildingAddress	ManagerID	ManagerFullName	ManagerPhone	ManagerEmail	EffectiveDate	EndDate	IsCurrent
1	B0001	123 Sunset Blvd, Los Angeles, CA 90028	M0001	Sharon Smith	212-555-1001	ssmith@example.com	2022-01-01	2024-10-31	0
2	B0002	456 Hollywood Ave, Los Angeles, CA 90038	M0001	Sharon Smith	212-555-1001	ssmith@example.com	2022-01-01	NULL	1
3	B0003	789 Vine St, Los Angeles, CA 90036	M0001	Sharon Smith	212-555-1001	ssmith@example.com	2022-01-01	2024-05-31	0
4	B0005	202 Capitol Mall, Sacramento, CA 95814	M0002	Sarah Johnson	617-555-2001	sjohnson@example.com	2022-01-01	NULL	1
5	B0006	303 J St, Sacramento, CA 95814	M0003	Michael Williams	312-555-3001	mwilliams@example.com	2022-01-01	NULL	1
6	B0007	404 L St, Sacramento, CA 95814	M0003	Michael Williams	312-555-3001	mwilliams@example.com	2022-01-01	NULL	1
7	B0010	707 Broadway, New York, NY 10003	M0004	Emma Brown	415-555-4001	ebrown@example.com	2022-01-01	NULL	1
8	B0008	505 Michigan Ave, Chicago, IL 60611	M0005	James Jones	206-555-5001	jjones@example.com	2022-01-01	2024-07-14	0
9	B0009	606 Wacker Dr, Chicago, IL 60601	M0005	James Jones	206-555-5001	jjones@example.com	2022-01-01	NULL	1
10	B0011	808 5th Ave, New York, NY 10021	M0006	Linda Garcia	305-555-6002	lgarcia@example.com	2022-01-01	NULL	1
11	B0013	110 Market St, San Francisco, CA 94103	M0007	Robert Miller	323-555-7001	rmiller@example.com	2022-01-01	NULL	1
12	B0014	211 Union St, San Francisco, CA 94111	M0007	Robert Miller	323-555-7001	rmiller@example.com	2022-01-01	NULL	1
13	B0012	909 Boylston St, Boston, MA 02116	M0008	Patricia Davis	303-555-8001	pdavis@example.com	2022-01-01	NULL	1
14	B0015	312 Pike St, Seattle, WA 98101	M0009	David Rodriguez	212-555-9001	drodriguez@example.com	2022-01-01	NULL	1
15	B0004	101 Wilshire Blvd, Los Angeles, CA 90017	M0010	Jennifer Martinez	617-555-0001	jmartinez@example.com	2022-01-01	NULL	1
16	B0003	789 Vine St, Los Angeles, CA 90036	M0004	Emma Brown	415-555-4001	ebrown@example.com	2024-06-01	NULL	1
17	B0008	505 Michigan Ave, Chicago, IL 60611	M0002	Sarah Johnson	617-555-2001	sjohnson@example.com	2024-07-15	NULL	1
18	B0001	123 Sunset Blvd, Los Angeles, CA 90028	M0007	Robert Miller	323-555-7001	rmiller@example.com	2024-11-01	NULL	1
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

– POPULATE DIM_SERVICETYPE TABLE

– This is a static table and only changes if a new service is offered (e.g., move-out service)

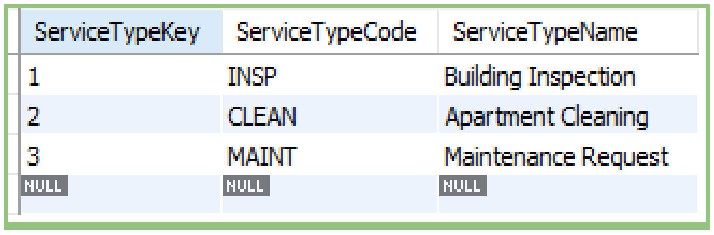
INSERT INTO Dim_ServiceType (ServiceTypeCode, ServiceTypeName) VALUES

('INSP', 'Building Inspection'),

('CLEAN', 'Apartment Cleaning'),

('MAINT', 'Maintenance Request');

DIM_SERVICETYPE TABLE



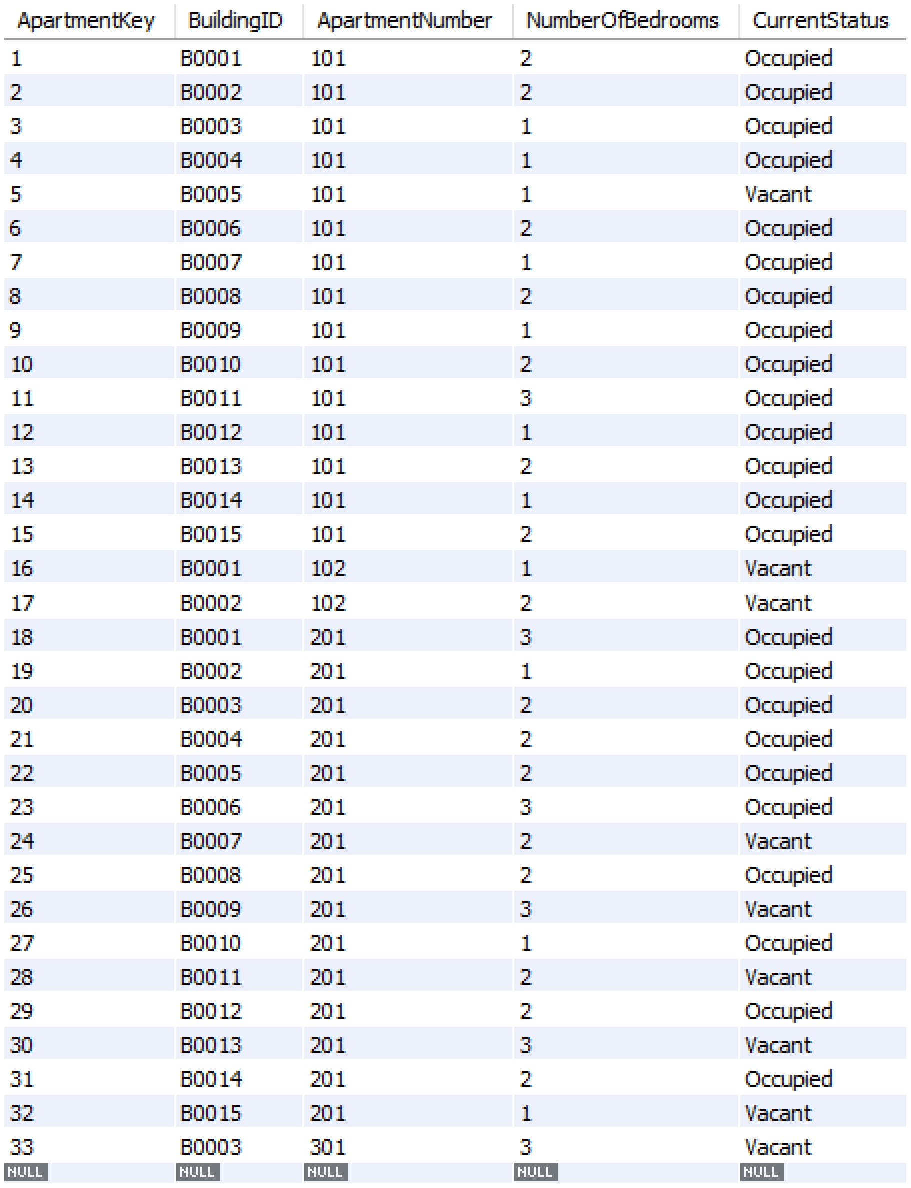
ServiceTypeKey	ServiceTypeCode	ServiceTypeName
1	INSP	Building Inspection
2	CLEAN	Apartment Cleaning
3	MAINT	Maintenance Request
NULL	NULL	NULL

– ETL DIM_APARTMENT:

INSERT INTO Dim_Apartment (BuildingID, ApartmentNumber, NumberOfBedrooms, CurrentStatus)

```
SELECT
   BuildingID, ApartmentNo, NoOfBedrooms, RentalStatus
FROM Apartment;
```

DIM_APARTMENT TABLE



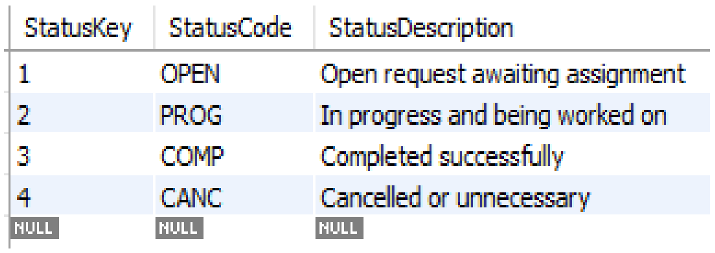
ApartmentKey	BuildingID	ApartmentNumber	NumberOfBedrooms	CurrentStatus
1	B0001	101	2	Occupied
2	B0002	101	2	Occupied
3	B0003	101	1	Occupied
4	B0004	101	1	Occupied
5	B0005	101	1	Vacant
6	B0006	101	2	Occupied
7	B0007	101	1	Occupied
8	B0008	101	2	Occupied
9	B0009	101	1	Occupied
10	B0010	101	2	Occupied
11	B0011	101	3	Occupied
12	B0012	101	1	Occupied
13	B0013	101	2	Occupied
14	B0014	101	1	Occupied
15	B0015	101	2	Occupied
16	B0001	102	1	Vacant
17	B0002	102	2	Vacant
18	B0001	201	3	Occupied
19	B0002	201	1	Occupied
20	B0003	201	2	Occupied
21	B0004	201	2	Occupied
22	B0005	201	2	Occupied
23	B0006	201	3	Occupied
24	B0007	201	2	Vacant
25	B0008	201	2	Occupied
26	B0009	201	3	Vacant
27	B0010	201	1	Occupied
28	B0011	201	2	Vacant
29	B0012	201	2	Occupied
30	B0013	201	3	Vacant
31	B0014	201	2	Occupied
32	B0015	201	1	Vacant
33	B0003	301	3	Vacant
NULL	NULL	NULL	NULL	NULL

– POPULATE DIM_STATUS TABLE

```
INSERT INTO Dim_Status (StatusCode, StatusDescription) VALUES
```

```
('OPEN', 'Open request awaiting assignment'),
('PROG', 'In progress and being worked on'),
('COMP', 'Completed successfully'),
('CANC', 'Cancelled or unnecessary');
```

DIM_STATUS TABLE

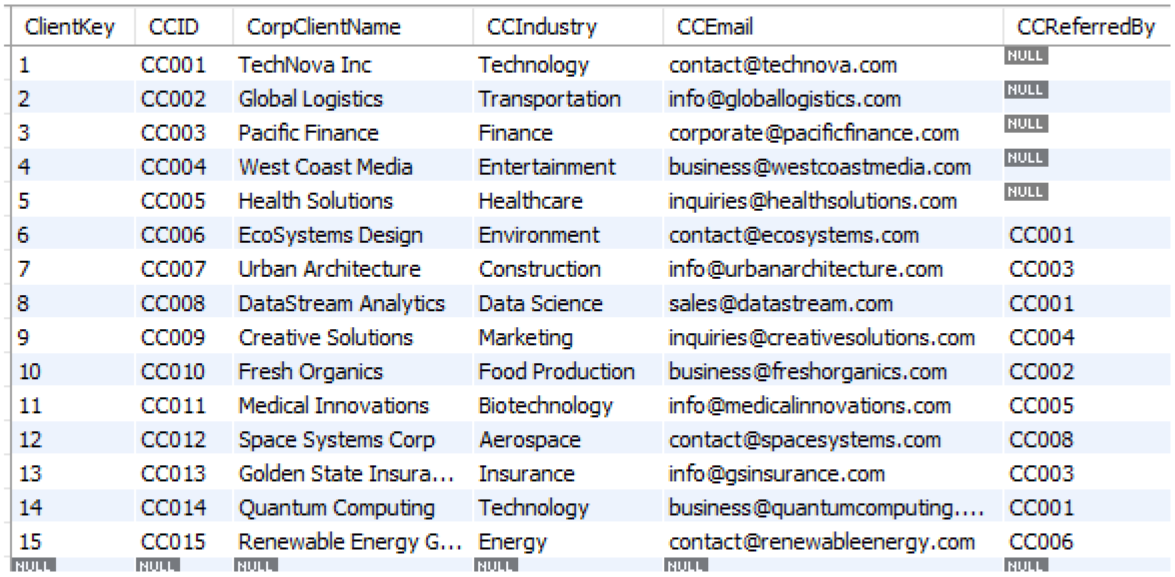


StatusKey	StatusCode	StatusDescription
1	OPEN	Open request awaiting assignment
2	PROG	In progress and being worked on
3	COMP	Completed successfully
4	CANC	Cancelled or unnecessary
NULL	NULL	NULL

– ETL DIM_CORPORATECLIENT:

```
INSERT INTO Dim_CorporateClient (CCID, CorpClientName, CCIndustry, CCEmail, CCReferredBy)
SELECT
   CCID, CCName, CCIndustry, CCEmail, CIDReferedBy
FROM CorporateClient;
```

DIM_CORPORATECLIENT TABLE



ClientKey	CCID	CorpClientName	CCIndustry	CCEmail	CCReferredBy
1	CC001	TechNova Inc	Technology	contact@technova.com	NULL
2	CC002	Global Logistics	Transportation	info@globallogistics.com	NULL
3	CC003	Pacific Finance	Finance	corporate@pacificfinance.com	NULL
4	CC004	West Coast Media	Entertainment	business@westcoastmedia.com	NULL
5	CC005	Health Solutions	Healthcare	inquiries@healthsolutions.com	NULL
6	CC006	EcoSystems Design	Environment	contact@ecosystems.com	CC001
7	CC007	Urban Architecture	Construction	info@urbanarchitecture.com	CC003
8	CC008	DataStream Analytics	Data Science	sales@datastream.com	CC001
9	CC009	Creative Solutions	Marketing	inquiries@creativesolutions.com	CC004
10	CC010	Fresh Organics	Food Production	business@freshorganics.com	CC002
11	CC011	Medical Innovations	Biotechnology	info@medicalinnovations.com	CC005
12	CC012	Space Systems Corp	Aerospace	contact@spacesystems.com	CC008
13	CC013	Golden State Insura...	Insurance	info@gsinsurance.com	CC003
14	CC014	Quantum Computing	Technology	business@quantumcomputing....	CC001
15	CC015	Renewable Energy G...	Energy	contact@renewableenergy.com	CC006
NULL	NULL	NULL	NULL	NULL	NULL

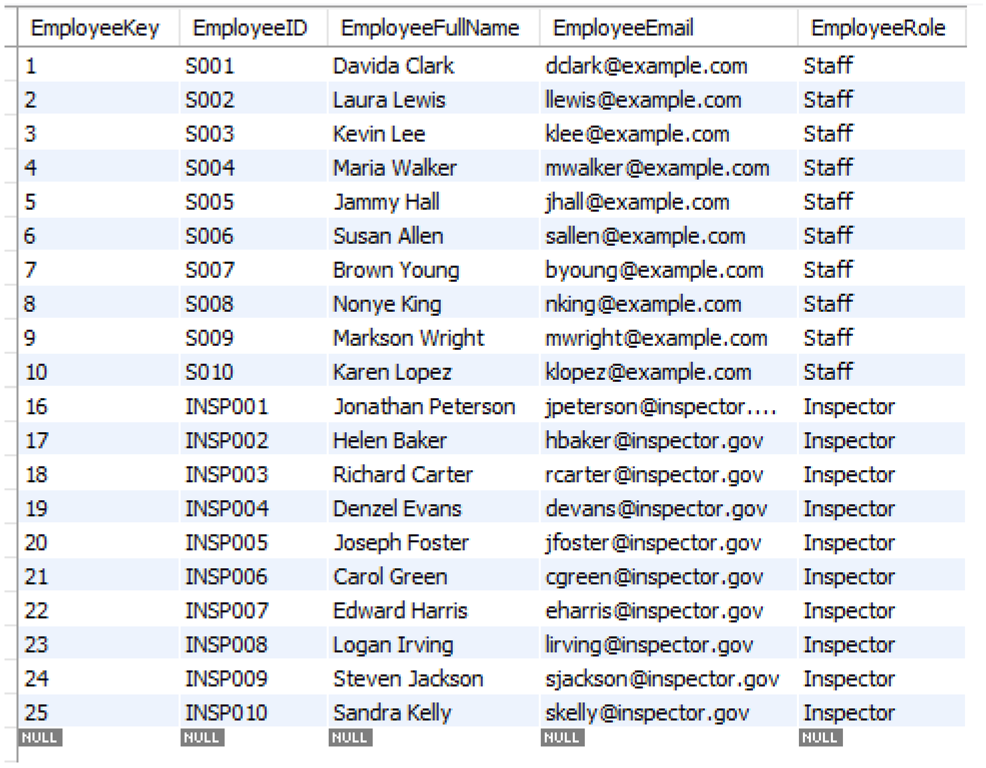
– ETL DIM_EMPLOYEE (Combining Staff and Inspectors):

```
-- Insert Staff Members
```

```
INSERT INTO Dim_Employee (EmployeeID, EmployeeFullName, EmployeeEmail, EmployeeRole)
SELECT StaffID,
   CONCAT(SFirstName, ' ', SLastName) AS EmployeeFullName,
   SEmail,
   'Staff' AS EmployeeRole
FROM StaffMember;
```

```
-- Insert Inspectors
INSERT INTO Dim_Employee (EmployeeID, EmployeeFullName, EmployeeEmail, EmployeeRole)
SELECT InspectorID,
   CONCAT(IFirstName, ' ', ILastName) AS EmployeeFullName,
   IEmail,
   'Inspector' AS EmployeeRole
FROM Inspector;
```

DIM_EMPLOYEE TABLE

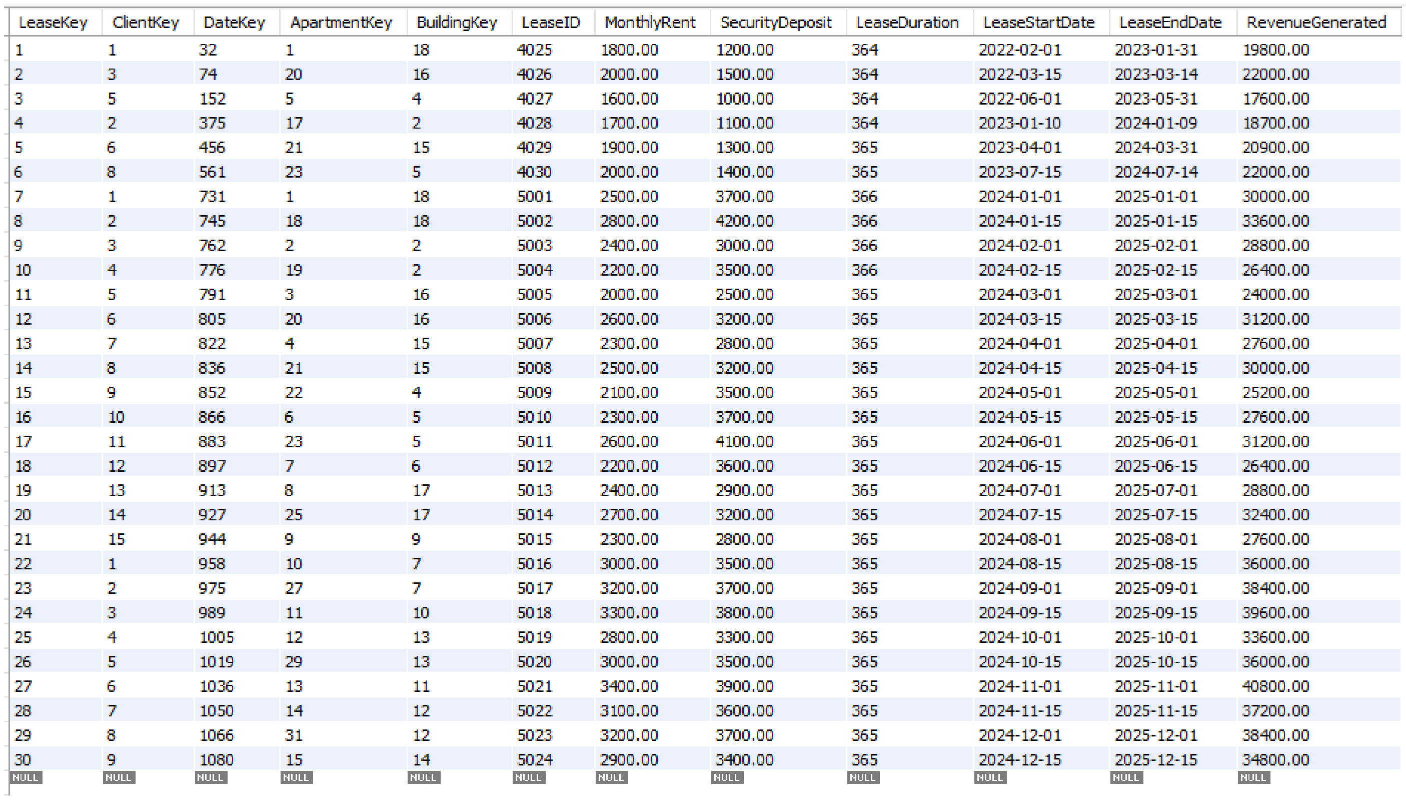


EmployeeKey	EmployeeID	EmployeeFullName	EmployeeEmail	EmployeeRole
1	S001	Davida Clark	dclark@example.com	Staff
2	S002	Laura Lewis	llewis@example.com	Staff
3	S003	Kevin Lee	klee@example.com	Staff
4	S004	Maria Walker	mwalker@example.com	Staff
5	S005	Jammy Hall	jhall@example.com	Staff
6	S006	Susan Allen	sallen@example.com	Staff
7	S007	Brown Young	byoung@example.com	Staff
8	S008	Nonye King	nking@example.com	Staff
9	S009	Markson Wright	mwright@example.com	Staff
10	S010	Karen Lopez	klopez@example.com	Staff
16	INSP001	Jonathan Peterson	jpeterson@inspector....	Inspector
17	INSP002	Helen Baker	hbaker@inspector.gov	Inspector
18	INSP003	Richard Carter	rcarter@inspector.gov	Inspector
19	INSP004	Denzel Evans	devans@inspector.gov	Inspector
20	INSP005	Joseph Foster	jfoster@inspector.gov	Inspector
21	INSP006	Carol Green	cgreen@inspector.gov	Inspector
22	INSP007	Edward Harris	eharris@inspector.gov	Inspector
23	INSP008	Logan Irving	lirving@inspector.gov	Inspector
24	INSP009	Steven Jackson	sjackson@inspector.gov	Inspector
25	INSP010	Sandra Kelly	skelly@inspector.gov	Inspector
NULL	NULL	NULL	NULL	NULL

– ETL FACT_LEASE:

– Loading data into the lease fact table from operational, dimension tables and transformed

```
metrics
INSERT INTO Fact_Lease (
   ClientKey, DateKey, ApartmentKey, BuildingKey, LeaseID, MonthlyRent, SecurityDeposit,
   LeaseDuration, LeaseStartDate, LeaseEndDate, RevenueGenerated
)
SELECT
   dc.ClientKey, dd.DateKey, da.ApartmentKey, db.BuildingKey, l.LeaseID,
   l.MonthlyRent, l.SecurityDeposit,
   DATEDIFF(l.LeaseEndDate, l.LeaseStartDate) AS LeaseDuration,
   l.LeaseStartDate, l.LeaseEndDate,
   -- Calculate total revenue as months * monthly rent
   (TIMESTAMPDIFF(MONTH, l.LeaseStartDate, l.LeaseEndDate) * l.MonthlyRent) AS
RevenueGenerated
FROM Lease l
JOIN Dim_CorporateClient dc ON l.CCID = dc.CCID
JOIN Dim_Date dd ON dd.Date = l.LeaseStartDate
JOIN Dim_Apartment da ON l.ApartmentNo = da.ApartmentNumber AND l.BuildingID =
da.BuildingID
JOIN Dim_Building db ON l.BuildingID = db.BuildingID AND db.IsCurrent = TRUE;
```



LeaseKey	ClientKey	DateKey	ApartmentKey	BuildingKey	LeaseID	MonthlyRent	SecurityDeposit	LeaseDuration	LeaseStartDate	LeaseEndDate	RevenueGenerated
1	1	32	1	18	4025	1800.00	1200.00	364	2022-02-01	2023-01-31	19800.00
2	3	74	20	16	4026	2000.00	1500.00	364	2022-03-15	2023-03-14	22000.00
3	5	152	5	4	4027	1600.00	1000.00	364	2022-06-01	2023-05-31	17600.00
4	2	375	17	2	4028	1700.00	1100.00	364	2023-01-10	2024-01-09	18700.00
5	6	456	21	15	4029	1900.00	1300.00	365	2023-04-01	2024-03-31	20900.00
6	8	561	23	5	4030	2000.00	1400.00	365	2023-07-15	2024-07-14	22000.00
7	1	731	1	18	5001	2500.00	3700.00	366	2024-01-01	2025-01-01	30000.00
8	2	745	18	18	5002	2800.00	4200.00	366	2024-01-15	2025-01-15	33600.00
9	3	762	2	2	5003	2400.00	3000.00	366	2024-02-01	2025-02-01	28800.00
10	4	776	19	2	5004	2200.00	3500.00	366	2024-02-15	2025-02-15	26400.00
11	5	791	3	16	5005	2000.00	2500.00	365	2024-03-01	2025-03-01	24000.00
12	6	805	20	16	5006	2600.00	3200.00	365	2024-03-15	2025-03-15	31200.00
13	7	822	4	15	5007	2300.00	2800.00	365	2024-04-01	2025-04-01	27600.00
14	8	836	21	15	5008	2500.00	3200.00	365	2024-04-15	2025-04-15	30000.00
15	9	852	22	4	5009	2100.00	3500.00	365	2024-05-01	2025-05-01	25200.00
16	10	866	6	5	5010	2300.00	3700.00	365	2024-05-15	2025-05-15	27600.00
17	11	883	23	5	5011	2600.00	4100.00	365	2024-06-01	2025-06-01	31200.00
18	12	897	7	6	5012	2200.00	3600.00	365	2024-06-15	2025-06-15	26400.00
19	13	913	8	17	5013	2400.00	2900.00	365	2024-07-01	2025-07-01	28800.00
20	14	927	25	17	5014	2700.00	3200.00	365	2024-07-15	2025-07-15	32400.00
21	15	944	9	9	5015	2300.00	2800.00	365	2024-08-01	2025-08-01	27600.00
22	1	958	10	7	5016	3000.00	3500.00	365	2024-08-15	2025-08-15	36000.00
23	2	975	27	7	5017	3200.00	3700.00	365	2024-09-01	2025-09-01	38400.00
24	3	989	11	10	5018	3300.00	3800.00	365	2024-09-15	2025-09-15	39600.00
25	4	1005	12	13	5019	2800.00	3300.00	365	2024-10-01	2025-10-01	33600.00
26	5	1019	29	13	5020	3000.00	3500.00	365	2024-10-15	2025-10-15	36000.00
27	6	1036	13	11	5021	3400.00	3900.00	365	2024-11-01	2025-11-01	40800.00
28	7	1050	14	12	5022	3100.00	3600.00	365	2024-11-15	2025-11-15	37200.00
29	8	1066	31	12	5023	3200.00	3700.00	365	2024-12-01	2025-12-01	38400.00
30	9	1080	15	14	5024	2900.00	3400.00	365	2024-12-15	2025-12-15	34800.00
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

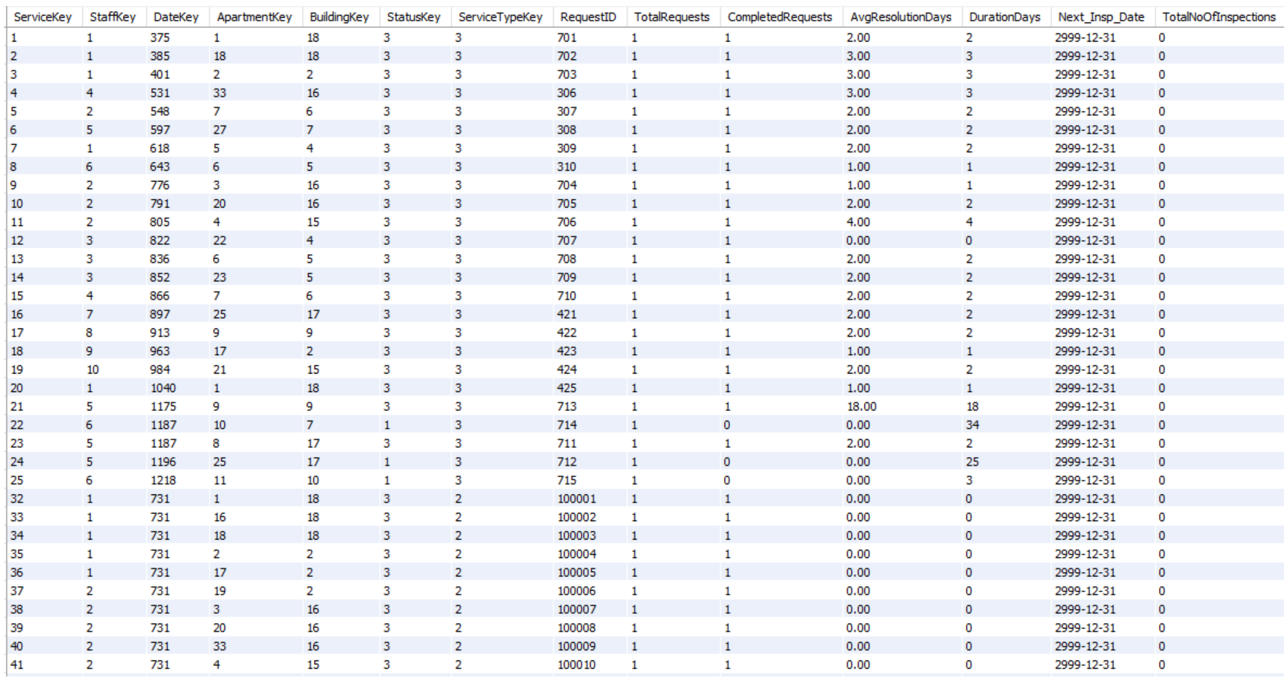
– ETL FACT_SERVICE:

– Combining Maintenance, Cleaning and Inspections

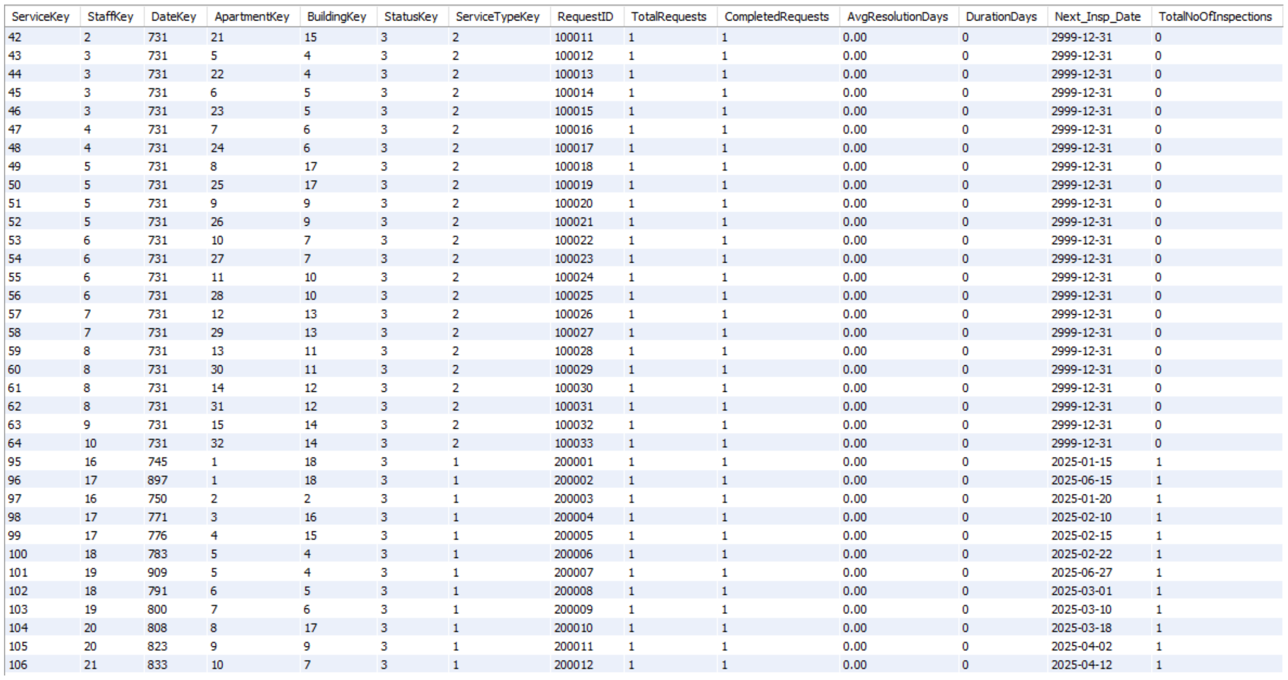
– Load the data from these operational and dimension tables and transformed metrics

```
-- Insert Maintenance Requests
INSERT INTO Fact_Service (
  StaffKey, DateKey, ApartmentKey, BuildingKey, StatusKey, ServiceTypeKey, RequestID,
  TotalRequests, CompletedRequests, AvgResolutionDays, DurationDays, Next_Insp_Date,
  TotalNoOfInspections
)
SELECT
  de.EmployeeKey, dd.DateKey, da.ApartmentKey, db.BuildingKey,
  -- Map status to status key
  (SELECT StatusKey FROM Dim_Status WHERE
    CASE
      WHEN mr.Status = 'Open' THEN StatusCode = 'OPEN'
      WHEN mr.Status = 'In Progress' THEN StatusCode = 'PROG'
      WHEN mr.Status = 'Completed' THEN StatusCode = 'COMP'
      ELSE StatusCode = 'CANC'
    END
  ) AS StatusKey,
  -- Service Type for Maintenance
  (SELECT ServiceTypeKey FROM Dim_ServiceType WHERE ServiceTypeCode = 'MAINT') AS
ServiceTypeKey,
  mr.RequestID,
  1 AS TotalRequests, -- Each record is one request
  CASE WHEN mr.Status = 'Completed' THEN 1 ELSE 0 END AS CompletedRequests,
  -- Calculate resolution days only for completed requests
  CASE
    WHEN mr.CompletedDate IS NOT NULL THEN DATEDIFF(mr.CompletedDate, mr.RequestDate)
    ELSE 0
  END AS AvgResolutionDays,
  -- Calculate duration (completed or current duration)
  CASE
    WHEN mr.CompletedDate IS NOT NULL THEN DATEDIFF(mr.CompletedDate, mr.RequestDate)
    ELSE DATEDIFF(CURRENT_DATE(), mr.RequestDate)
  END AS DurationDays,
  -- Not applicable for maintenance requests
  '2999-12-31' AS Next_Insp_Date,          -- common placeholder for non-applicable field
  0 AS TotalNoOfInspections
FROM MaintenanceRequest mr
JOIN Dim_Employee de ON mr.AssignedStaff = de.EmployeeID
```

```sql
JOIN Dim_Date dd ON dd.Date = mr.RequestDate
JOIN Dim_Apartment da ON mr.ApartmentNo = da.ApartmentNumber AND mr.BuildingID =
da.BuildingID
JOIN Dim_Building db ON mr.BuildingID = db.BuildingID AND db.IsCurrent = TRUE;

-- Insert Cleaning Records
INSERT INTO Fact_Service (
  StaffKey, DateKey, ApartmentKey, BuildingKey, StatusKey, ServiceTypeKey, RequestID,
  TotalRequests, CompletedRequests, AvgResolutionDays, DurationDays, Next_Insp_Date,
  TotalNoOfInspections
)
SELECT
  de.EmployeeKey,
  -- Use a reference date since cleaning doesn't have specific dates
  (SELECT DateKey FROM Dim_Date WHERE Date = '2024-01-01') AS DateKey,
  da.ApartmentKey,
  db.BuildingKey,
  -- Default to completed for cleaning assignments
  (SELECT StatusKey FROM Dim_Status WHERE StatusCode = 'COMP') AS StatusKey,
  -- Service Type for Cleaning
  (SELECT ServiceTypeKey FROM Dim_ServiceType WHERE ServiceTypeCode = 'CLEAN') AS
ServiceTypeKey,
  -- Generate a synthetic request ID for cleaning
  100000 + ROW_NUMBER() OVER (ORDER BY c.BuildingID, c.ApartmentNo) AS RequestID,
  1 AS TotalRequests,
  1 AS CompletedRequests, -- Assume all cleanings are completed
  0 AS AvgResolutionDays, -- Not applicable
  0 AS DurationDays, -- Not applicable
  '2999-12-31' AS Next_Insp_Date, -- Not applicable
  0 AS TotalNoOfInspections -- Not applicable
FROM Cleans c
JOIN Dim_Employee de ON c.StaffID = de.EmployeeID
JOIN Dim_Apartment da ON c.ApartmentNo = da.ApartmentNumber AND c.BuildingID =
da.BuildingID
JOIN Dim_Building db ON c.BuildingID = db.BuildingID AND db.IsCurrent = TRUE;

-- Insert Inspection Records
INSERT INTO Fact_Service (
  StaffKey, DateKey, ApartmentKey, BuildingKey, StatusKey, ServiceTypeKey, RequestID,
  TotalRequests, CompletedRequests, AvgResolutionDays, DurationDays, Next_Insp_Date,
  TotalNoOfInspections
)
SELECT
  de.EmployeeKey,
  dd.DateKey,
  -- Not tied to an apartment, use a default value
  (SELECT MIN(ApartmentKey) FROM Dim_Apartment WHERE BuildingID = i.BuildingID) AS
ApartmentKey,
  db.BuildingKey,
  -- Default to completed for past inspections
  (SELECT StatusKey FROM Dim_Status WHERE StatusCode = 'COMP') AS StatusKey,
  -- Service Type for Inspection
  (SELECT ServiceTypeKey FROM Dim_ServiceType WHERE ServiceTypeCode = 'INSP') AS
ServiceTypeKey,
  -- Generate a synthetic request ID for inspections
  200000 + ROW_NUMBER() OVER (ORDER BY i.BuildingID, i.InspectorID) AS RequestID,
  1 AS TotalRequests,
  1 AS CompletedRequests, -- All recorded inspections are completed
  0 AS AvgResolutionDays, -- Not applicable
  0 AS DurationDays, -- Not applicable
  i.Next_Insp_date,
  1 AS TotalNoOfInspections
FROM Inspects i
JOIN Dim_Employee de ON i.InspectorID = de.EmployeeID
JOIN Dim_Date dd ON dd.Date = i.Date_Completed
JOIN Dim_Building db ON i.BuildingID = db.BuildingID AND db.IsCurrent = TRUE;
```



ServiceKey	StaffKey	DateKey	ApartmentKey	BuildingKey	StatusKey	ServiceTypeKey	RequestID	TotalRequests	CompletedRequests	AvgResolutionDays	DurationDays	Next_Insp_Date	TotalNoOfInspections
1	1	375	1	18	3	3	701	1	1	2.00	2	2999-12-31	0
2	1	385	18	18	3	3	702	1	1	3.00	3	2999-12-31	0
3	1	401	2	2	3	3	703	1	1	3.00	3	2999-12-31	0
4	4	531	33	16	3	3	306	1	1	3.00	3	2999-12-31	0
5	2	548	7	6	3	3	307	1	1	2.00	2	2999-12-31	0
6	5	597	27	7	3	3	308	1	1	2.00	2	2999-12-31	0
7	1	618	5	4	3	3	309	1	1	2.00	2	2999-12-31	0
8	6	643	6	5	3	3	310	1	1	1.00	1	2999-12-31	0
9	2	776	3	16	3	3	704	1	1	1.00	1	2999-12-31	0
10	2	791	20	16	3	3	705	1	1	2.00	2	2999-12-31	0
11	2	805	4	15	3	3	706	1	1	4.00	4	2999-12-31	0
12	3	822	22	4	3	3	707	1	1	0.00	0	2999-12-31	0
13	3	836	6	5	3	3	708	1	1	2.00	2	2999-12-31	0
14	3	852	23	5	3	3	709	1	1	2.00	2	2999-12-31	0
15	4	866	7	6	3	3	710	1	1	2.00	2	2999-12-31	0
16	7	897	25	17	3	3	421	1	1	2.00	2	2999-12-31	0
17	8	913	9	9	3	3	422	1	1	2.00	2	2999-12-31	0
18	9	963	17	2	3	3	423	1	1	1.00	1	2999-12-31	0
19	10	984	21	15	3	3	424	1	1	2.00	2	2999-12-31	0
20	1	1040	1	18	3	3	425	1	1	1.00	1	2999-12-31	0
21	5	1175	9	9	3	3	713	1	1	18.00	18	2999-12-31	0
22	6	1187	10	7	1	3	714	1	0	0.00	34	2999-12-31	0
23	5	1187	8	17	3	3	711	1	1	2.00	2	2999-12-31	0
24	5	1196	25	17	1	3	712	1	0	0.00	25	2999-12-31	0
25	6	1218	11	10	1	3	715	1	0	0.00	3	2999-12-31	0
32	1	731	1	18	3	2	100001	1	1	0.00	0	2999-12-31	0
33	1	731	16	18	3	2	100002	1	1	0.00	0	2999-12-31	0
34	1	731	18	18	3	2	100003	1	1	0.00	0	2999-12-31	0
35	1	731	2	2	3	2	100004	1	1	0.00	0	2999-12-31	0
36	1	731	17	2	3	2	100005	1	1	0.00	0	2999-12-31	0
37	2	731	19	2	3	2	100006	1	1	0.00	0	2999-12-31	0
38	2	731	3	16	3	2	100007	1	1	0.00	0	2999-12-31	0
39	2	731	20	16	3	2	100008	1	1	0.00	0	2999-12-31	0
40	2	731	33	16	3	2	100009	1	1	0.00	0	2999-12-31	0
41	2	731	4	15	3	2	100010	1	1	0.00	0	2999-12-31	0



ServiceKey	StaffKey	DateKey	ApartmentKey	BuildingKey	StatusKey	ServiceTypeKey	RequestID	TotalRequests	CompletedRequests	AvgResolutionDays	DurationDays	Next_Insp_Date	TotalNoOfInspections
42	2	731	21	15	3	2	100011	1	1	0.00	0	2999-12-31	0
43	3	731	5	4	3	2	100012	1	1	0.00	0	2999-12-31	0
44	3	731	22	4	3	2	100013	1	1	0.00	0	2999-12-31	0
45	3	731	6	5	3	2	100014	1	1	0.00	0	2999-12-31	0
46	3	731	23	5	3	2	100015	1	1	0.00	0	2999-12-31	0
47	4	731	7	6	3	2	100016	1	1	0.00	0	2999-12-31	0
48	4	731	24	6	3	2	100017	1	1	0.00	0	2999-12-31	0
49	5	731	8	17	3	2	100018	1	1	0.00	0	2999-12-31	0
50	5	731	25	17	3	2	100019	1	1	0.00	0	2999-12-31	0
51	5	731	9	9	3	2	100020	1	1	0.00	0	2999-12-31	0
52	5	731	26	9	3	2	100021	1	1	0.00	0	2999-12-31	0
53	6	731	10	7	3	2	100022	1	1	0.00	0	2999-12-31	0
54	6	731	27	7	3	2	100023	1	1	0.00	0	2999-12-31	0
55	6	731	11	10	3	2	100024	1	1	0.00	0	2999-12-31	0
56	6	731	28	10	3	2	100025	1	1	0.00	0	2999-12-31	0
57	7	731	12	13	3	2	100026	1	1	0.00	0	2999-12-31	0
58	7	731	29	13	3	2	100027	1	1	0.00	0	2999-12-31	0
59	8	731	13	11	3	2	100028	1	1	0.00	0	2999-12-31	0
60	8	731	30	11	3	2	100029	1	1	0.00	0	2999-12-31	0
61	8	731	14	12	3	2	100030	1	1	0.00	0	2999-12-31	0
62	8	731	31	12	3	2	100031	1	1	0.00	0	2999-12-31	0
63	9	731	15	14	3	2	100032	1	1	0.00	0	2999-12-31	0
64	10	731	32	14	3	2	100033	1	1	0.00	0	2999-12-31	0
95	16	745	1	18	3	1	200001	1	1	0.00	0	2025-01-15	1
96	17	897	1	18	3	1	200002	1	1	0.00	0	2025-06-15	1
97	16	750	2	2	3	1	200003	1	1	0.00	0	2025-01-20	1
98	17	771	3	16	3	1	200004	1	1	0.00	0	2025-02-10	1
99	17	776	4	15	3	1	200005	1	1	0.00	0	2025-02-15	1
100	18	783	5	4	3	1	200006	1	1	0.00	0	2025-02-22	1
101	19	909	5	4	3	1	200007	1	1	0.00	0	2025-06-27	1
102	18	791	6	5	3	1	200008	1	1	0.00	0	2025-03-01	1
103	19	800	7	6	3	1	200009	1	1	0.00	0	2025-03-10	1
104	20	808	8	17	3	1	200010	1	1	0.00	0	2025-03-18	1
105	20	823	9	9	3	1	200011	1	1	0.00	0	2025-04-02	1
106	21	833	10	7	3	1	200012	1	1	0.00	0	2025-04-12	1

2. (15 points) Implement summary tables with appropriate aggregations (e.g., total rentals, inspections per building) for efficiency for querying.

   Examples of the summary tables include: "Total rental income per building, by year.", "Number of inspections per building per month.", "Staff performance based on the number of maintenance tasks completed."

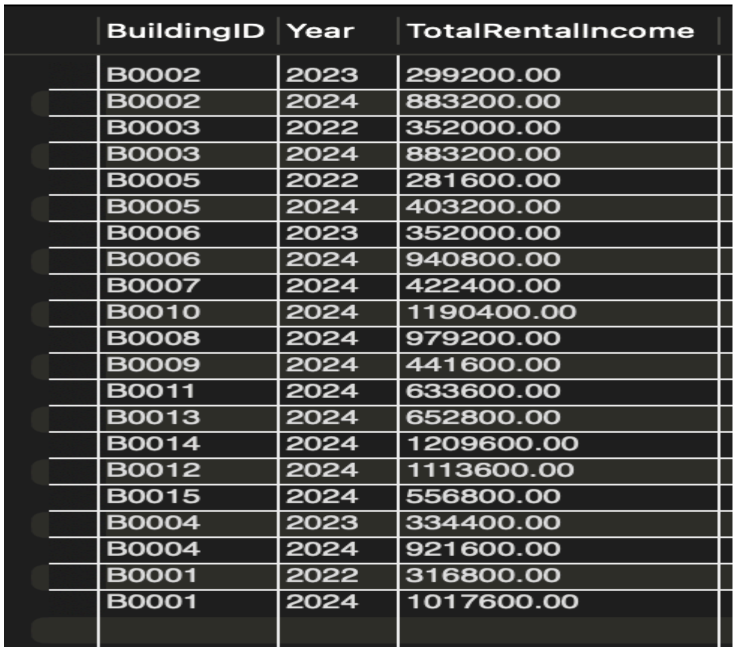
SUMMARY TABLES

Summary of Total rental income per building by year

This summary aggregates the total rental revenue generated for each building, grouped by year. It pulls data from Fact_Lease, joins with Dim_Building and Dim_Date to extract the building ID and the year of the lease start. **Use Case:** Analyze income performance across buildings and years.

– Summary table for Rental income

```
CREATE TABLE Summary_RentalIncome_ByBuilding_Year AS
SELECT
   db.BuildingID,
   dd.Year,
   SUM(fl.RevenueGenerated) AS TotalRentalIncome
FROM Fact_Lease fl
JOIN Dim_Building db ON fl.BuildingKey = db.BuildingKey
JOIN Dim_Date dd ON fl.DateKey = dd.DateKey
GROUP BY db.BuildingID, dd.Year;
```



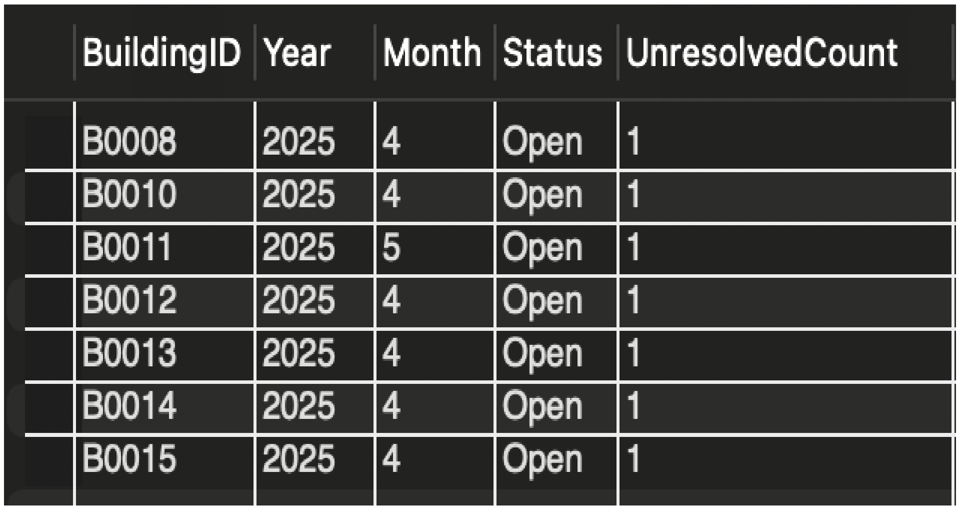
BuildingID	Year	TotalRentalIncome
B0002	2023	299200.00
B0002	2024	883200.00
B0003	2022	352000.00
B0003	2024	883200.00
B0005	2022	281600.00
B0005	2024	403200.00
B0006	2023	352000.00
B0006	2024	940800.00
B0007	2024	422400.00
B0010	2024	1190400.00
B0008	2024	979200.00
B0009	2024	441600.00
B0011	2024	633600.00
B0013	2024	652800.00
B0014	2024	1209600.00
B0012	2024	1113600.00
B0015	2024	556800.00
B0004	2023	334400.00
B0004	2024	921600.00
B0001	2022	316800.00
B0001	2024	1017600.00

Summary Table for Unsolved Maintenance Request

This summarizes all open or in-progress maintenance requests, grouped by building and time (year, month). It queries directly from the operational MaintenanceRequest table.
Use Case: Monitor operational backlog and detect problem buildings or months.

– Summary table for unresolved maintenance request

```
CREATE TABLE Summary_UnresolvedRequests AS
SELECT
   BuildingID,
   YEAR(RequestDate) AS Year,
   MONTH(RequestDate) AS Month,
   Status,
   COUNT(*) AS UnresolvedCount
FROM MaintenanceRequest
WHERE Status IN ('Open', 'In Progress')
GROUP BY BuildingID, YEAR(RequestDate), MONTH(RequestDate), Status
ORDER BY BuildingID, Year, Month;
```

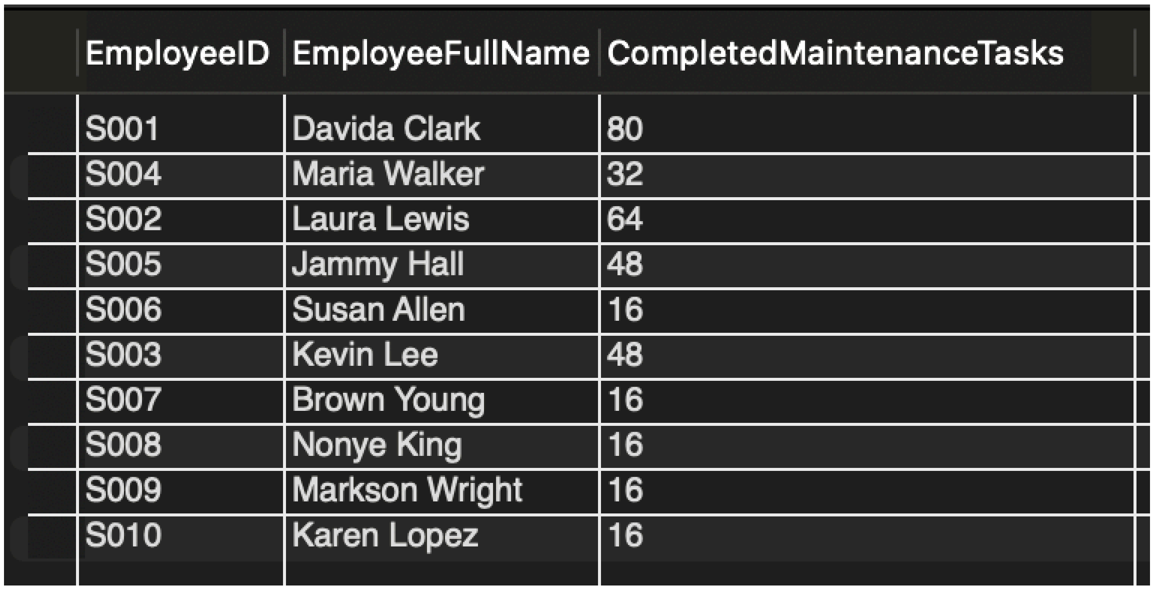


BuildingID	Year	Month	Status	UnresolvedCount
B0008	2025	4	Open	1
B0010	2025	4	Open	1
B0011	2025	5	Open	1
B0012	2025	4	Open	1
B0013	2025	4	Open	1
B0014	2025	4	Open	1
B0015	2025	4	Open	1

Summary of Staff performance based on the number of maintenance tasks completed

This aggregates the total number of completed maintenance tasks handled by each staff member. It filters the Fact_Service table for Maintenance Request and Completed successfully. **Use Case**: Evaluate which staff are completing the most maintenance jobs.

```
– Summary table of maintenance performance by staff
CREATE TABLE Summary_MaintenancePerformanceByStaff AS
SELECT
   de.EmployeeID,
   de.EmployeeFullName,
   COUNT(*) AS CompletedMaintenanceTasks
FROM Fact_Service fs
JOIN Dim_Employee de ON fs.StaffKey = de.EmployeeKey
JOIN Dim_ServiceType dst ON fs.ServiceTypeKey = dst.ServiceTypeKey
JOIN Dim_Status ds ON fs.StatusKey = ds.StatusKey
WHERE dst.ServiceTypeName = 'Maintenance Request'
  AND ds.StatusDescription = 'Completed successfully'
GROUP BY de.EmployeeID, de.EmployeeFullName;
```



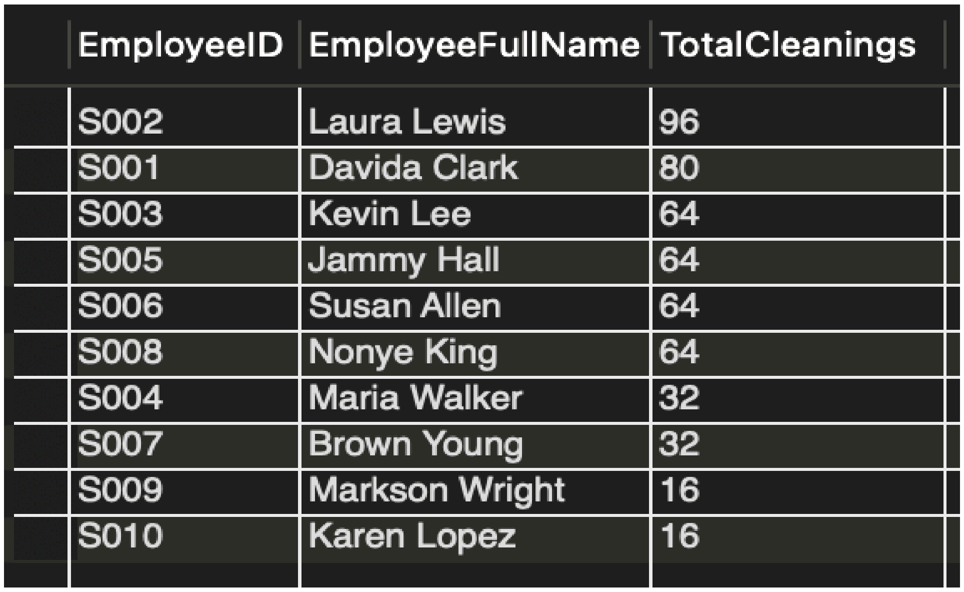
EmployeeID	EmployeeFullName	CompletedMaintenanceTasks
S001	Davida Clark	80
S004	Maria Walker	32
S002	Laura Lewis	64
S005	Jammy Hall	48
S006	Susan Allen	16
S003	Kevin Lee	48
S007	Brown Young	16
S008	Nonye King	16
S009	Markson Wright	16
S010	Karen Lopez	16

Summary of Cleaning Services Completed per Staff

This summary tracks how many cleaning services were completed by each staff member. It filters Fact_Service for Apartment Cleaning services and completed status.

Use Case: Track productivity of cleaning staff across properties.

```
– Summary of cleaning services completed per staff
CREATE TABLE Summary_CleaningPerformance_ByStaff AS
SELECT
   de.EmployeeID,
   de.EmployeeFullName,
   COUNT(*) AS TotalCleanings
FROM Fact_Service fs
JOIN Dim_Employee de ON fs.StaffKey = de.EmployeeKey
JOIN Dim_ServiceType dst ON fs.ServiceTypeKey = dst.ServiceTypeKey
JOIN Dim_Status ds ON fs.StatusKey = ds.StatusKey
WHERE dst.ServiceTypeName = 'Apartment Cleaning'
  AND ds.StatusDescription = 'Completed successfully'
GROUP BY de.EmployeeID, de.EmployeeFullName;
```

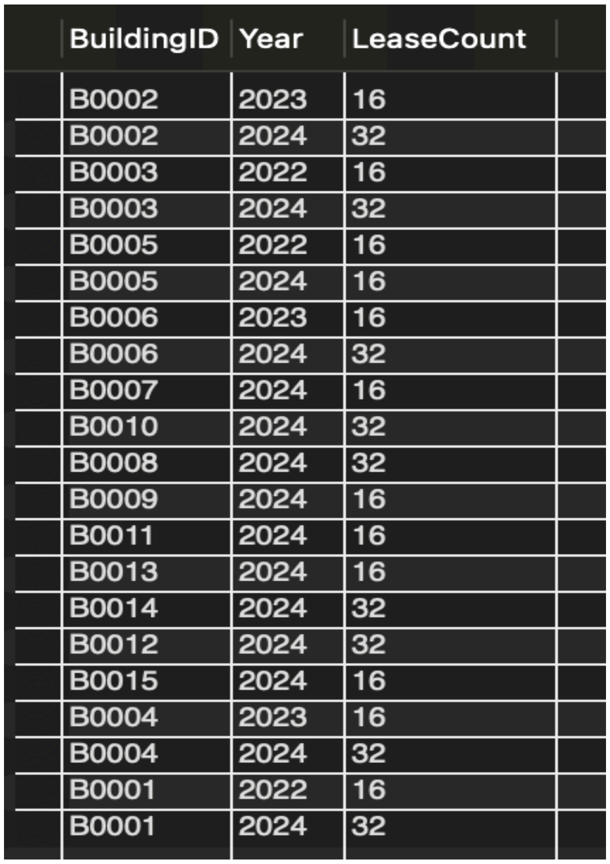


EmployeeID	EmployeeFullName	TotalCleanings
S002	Laura Lewis	96
S001	Davida Clark	80
S003	Kevin Lee	64
S005	Jammy Hall	64
S006	Susan Allen	64
S008	Nonye King	64
S004	Maria Walker	32
S007	Brown Young	32
S009	Markson Wright	16
S010	Karen Lopez	16

Summary of Lease Count per Building by Year

Counts how many leases were signed in each building per year — pulled from Fact_Lease.
Use Case: Track occupancy activity across buildings and time.

```
– Summary of lease count per building by Year
CREATE TABLE Summary_LeaseCount_ByBuilding_Year AS
SELECT
   db.BuildingID,
   dd.Year,
   COUNT(*) AS LeaseCount
FROM Fact_Lease fl
JOIN Dim_Building db ON fl.BuildingKey = db.BuildingKey
JOIN Dim_Date dd ON fl.DateKey = dd.DateKey
GROUP BY db.BuildingID, dd.Year;
```



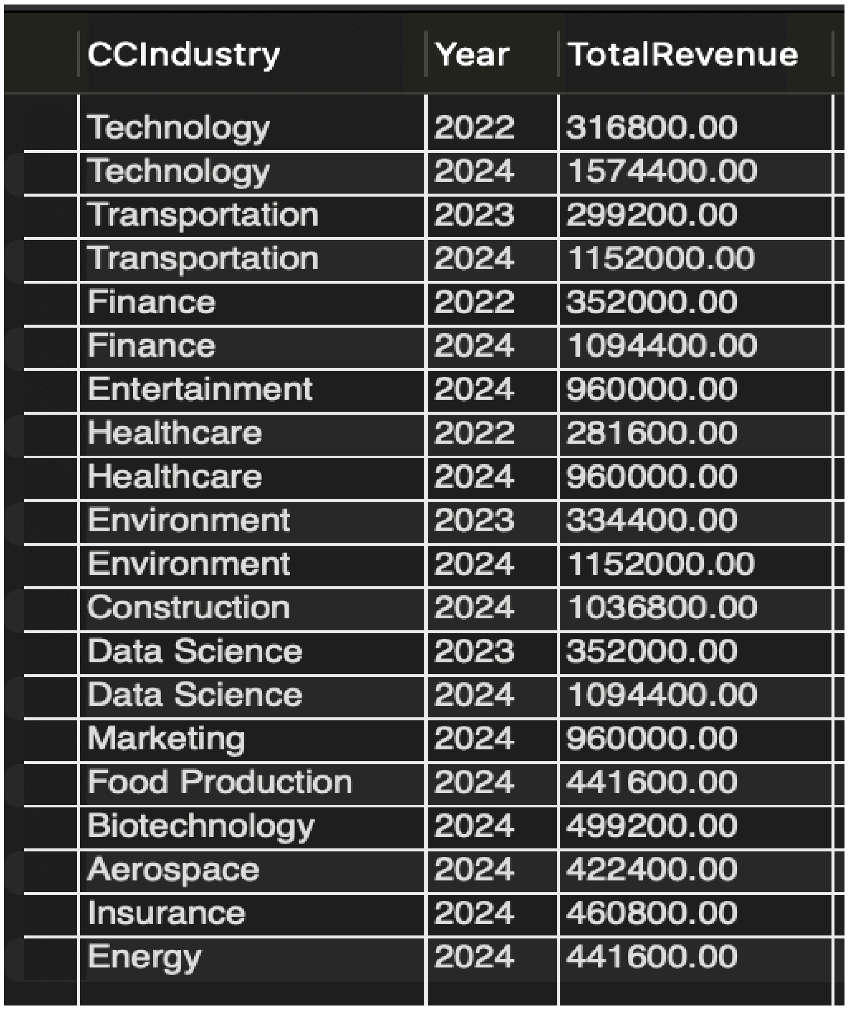
BuildingID	Year	LeaseCount
B0002	2023	16
B0002	2024	32
B0003	2022	16
B0003	2024	32
B0005	2022	16
B0005	2024	16
B0006	2023	16
B0006	2024	32
B0007	2024	16
B0010	2024	32
B0008	2024	32
B0009	2024	16
B0011	2024	16
B0013	2024	16
B0014	2024	32
B0012	2024	32
B0015	2024	16
B0004	2023	16
B0004	2024	32
B0001	2022	16
B0001	2024	32

Summary of Revenue by Client Industry per Year

Summarizes total lease revenue per client industry per year — showing which industries generate the most business.

Use Case: Identify which industries are your biggest corporate clients.

```
– Summary of revenue by client industry per year
CREATE TABLE Summary_Revenue_ByIndustry_Year AS
SELECT
    dcc.CCIndustry,
    dd.Year,
    SUM(fl.RevenueGenerated) AS TotalRevenue
FROM Fact_Lease fl
JOIN Dim_CorporateClient dcc ON fl.ClientKey = dcc.ClientKey
JOIN Dim_Date dd ON fl.DateKey = dd.DateKey
GROUP BY dcc.CCIndustry, dd.Year;
```



CCIndustry	Year	TotalRevenue
Technology	2022	316800.00
Technology	2024	1574400.00
Transportation	2023	299200.00
Transportation	2024	1152000.00
Finance	2022	352000.00
Finance	2024	1094400.00
Entertainment	2024	960000.00
Healthcare	2022	281600.00
Healthcare	2024	960000.00
Environment	2023	334400.00
Environment	2024	1152000.00
Construction	2024	1036800.00
Data Science	2023	352000.00
Data Science	2024	1094400.00
Marketing	2024	960000.00
Food Production	2024	441600.00
Biotechnology	2024	499200.00
Aerospace	2024	422400.00
Insurance	2024	460800.00
Energy	2024	441600.00

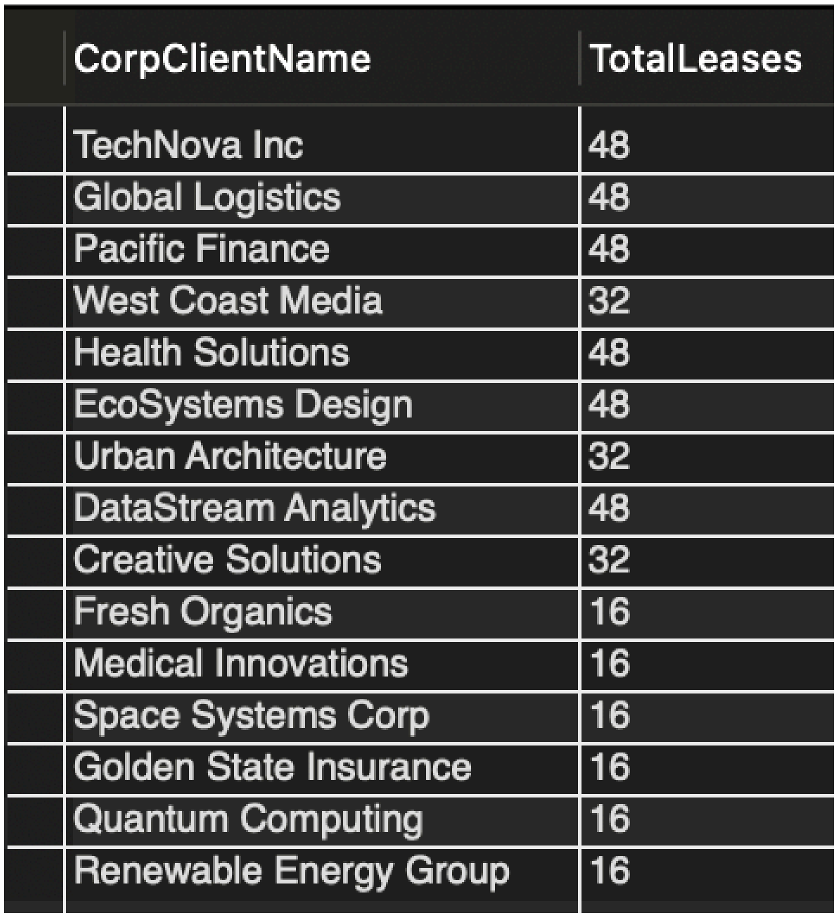
Summary of corporate clients that lease multiple units

Identifies clients who signed more than one lease, showing client retention and value.

Use Case: Spot valuable repeat clients for retention programs.

– Summary table for corporate clients that lease multiple units

```
CREATE TABLE Summary_RepeatClients AS
SELECT
   dcc.CorpClientName,
   COUNT(*) AS TotalLeases
FROM Fact_Lease fl
JOIN Dim_CorporateClient dcc ON fl.ClientKey = dcc.ClientKey
GROUP BY dcc.CorpClientName
HAVING COUNT(*) > 1;
```



CorpClientName	TotalLeases
TechNova Inc	48
Global Logistics	48
Pacific Finance	48
West Coast Media	32
Health Solutions	48
EcoSystems Design	48
Urban Architecture	32
DataStream Analytics	48
Creative Solutions	32
Fresh Organics	16
Medical Innovations	16
Space Systems Corp	16
Golden State Insurance	16
Quantum Computing	16
Renewable Energy Group	16

3. (20 points) Generate 5 business intelligence (BI) reports using SQL that answers strategic business questions. Reports should be based on meaningful queries that provide actionable insights, such as occupancy trends to analyze historical occupancy rates across buildings over the past 3 years.

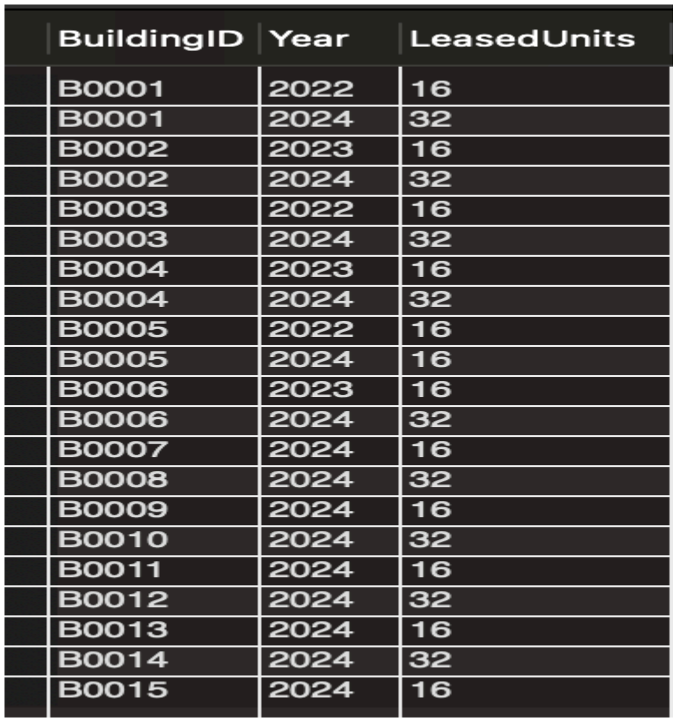
   Examples of BI reports include:

   -Occupancy Trends: Analyze occupancy rates over time (e.g., by building, by month, or by year).
   -Inspection Trends: Identify buildings with recurring inspection issues.
   -Maintenance Efficiency: Track how quickly maintenance requests are resolved over time.
   -Rental Revenue Trends: Analyze rental income per building over the past 3 years.
   -Staff Performance: Analyze which staff members handle the most or most complex maintenance requests.

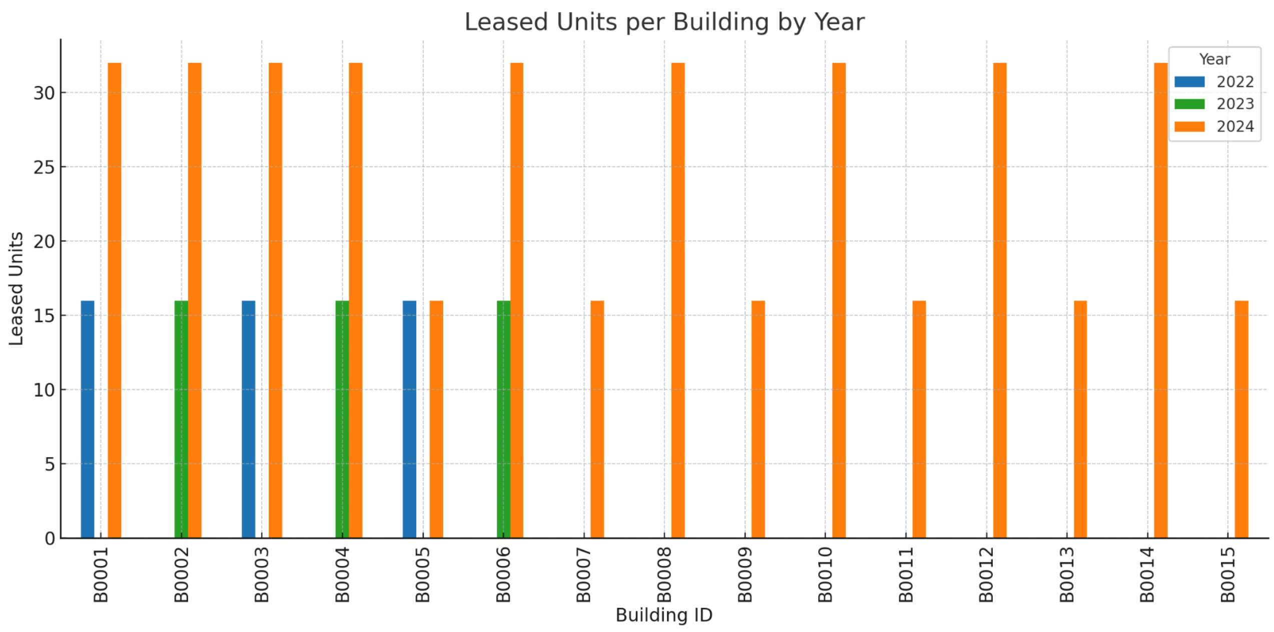
Occupancy Trends (by building and year)

Business Question: What are the occupancy rates per building for the past 3 years?

```
SELECT
  BuildingID,
  Year,
  LeaseCount AS LeasedUnits
FROM Summary_LeaseCount_ByBuilding_Year
WHERE Year >= YEAR(CURDATE()) - 2
ORDER BY BuildingID, Year;
```



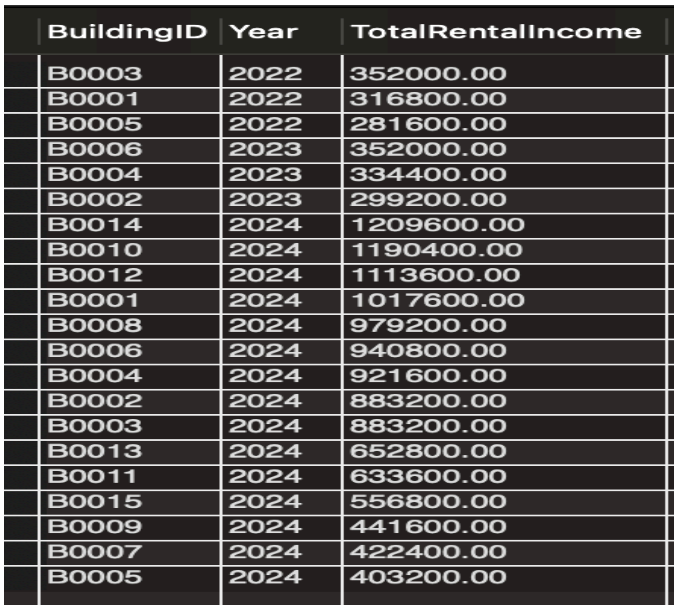
BuildingID	Year	LeasedUnits
B0001	2022	16
B0001	2024	32
B0002	2023	16
B0002	2024	32
B0003	2022	16
B0003	2024	32
B0004	2023	16
B0004	2024	32
B0005	2022	16
B0005	2024	16
B0006	2023	16
B0006	2024	32
B0007	2024	16
B0008	2024	32
B0009	2024	16
B0010	2024	32
B0011	2024	16
B0012	2024	32
B0013	2024	16
B0014	2024	32
B0015	2024	16



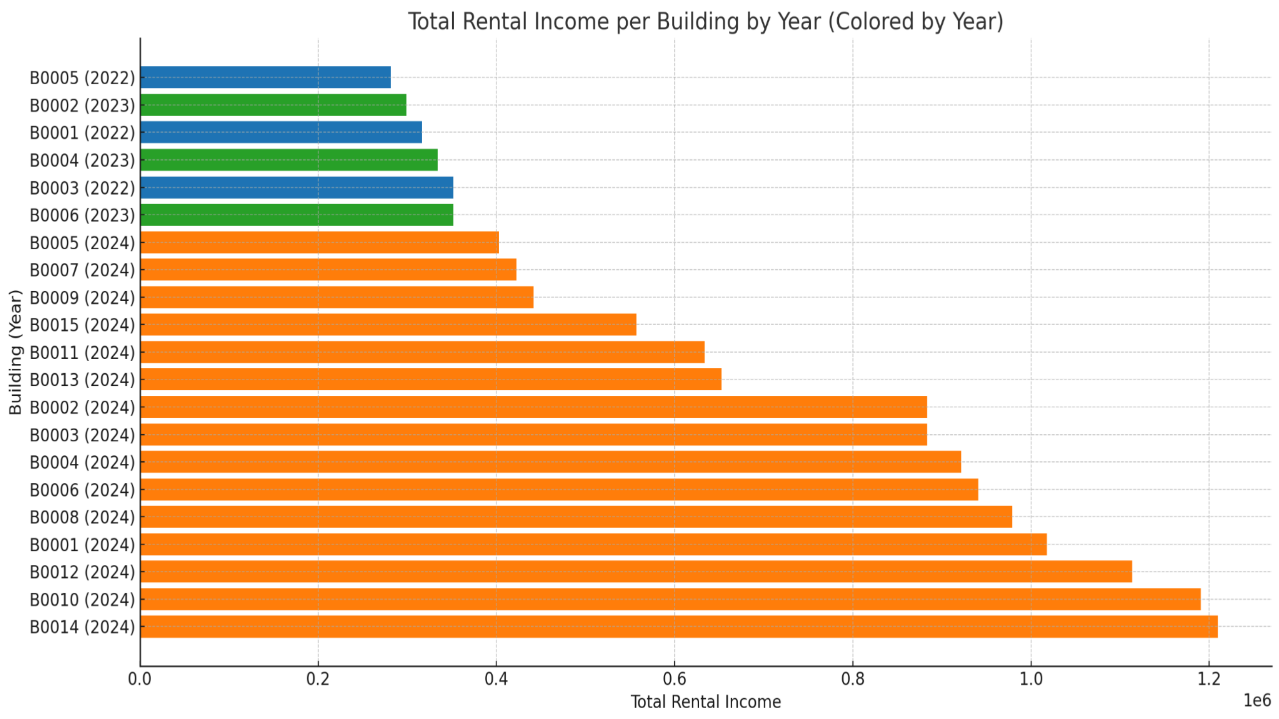
Rental Income by Building and Year

Business Question: Which buildings are generating the most rental income year over year?

```
SELECT *
FROM Summary_RentalIncome_ByBuilding_Year
ORDER BY Year, TotalRentalIncome DESC;
```



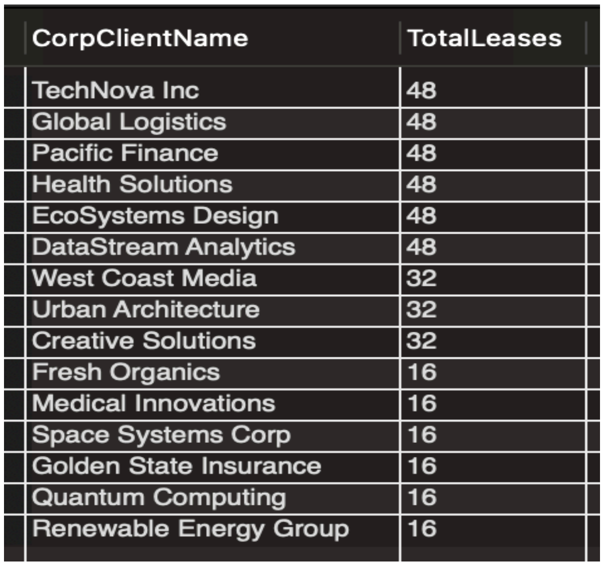
BuildingID	Year	TotalRentalIncome
B0003	2022	352000.00
B0001	2022	316800.00
B0005	2022	281600.00
B0006	2023	352000.00
B0004	2023	334400.00
B0002	2023	299200.00
B0014	2024	1209600.00
B0010	2024	1190400.00
B0012	2024	1113600.00
B0001	2024	1017600.00
B0008	2024	979200.00
B0006	2024	940800.00
B0004	2024	921600.00
B0002	2024	883200.00
B0003	2024	883200.00
B0013	2024	652800.00
B0011	2024	633600.00
B0015	2024	556800.00
B0009	2024	441600.00
B0007	2024	422400.00
B0005	2024	403200.00



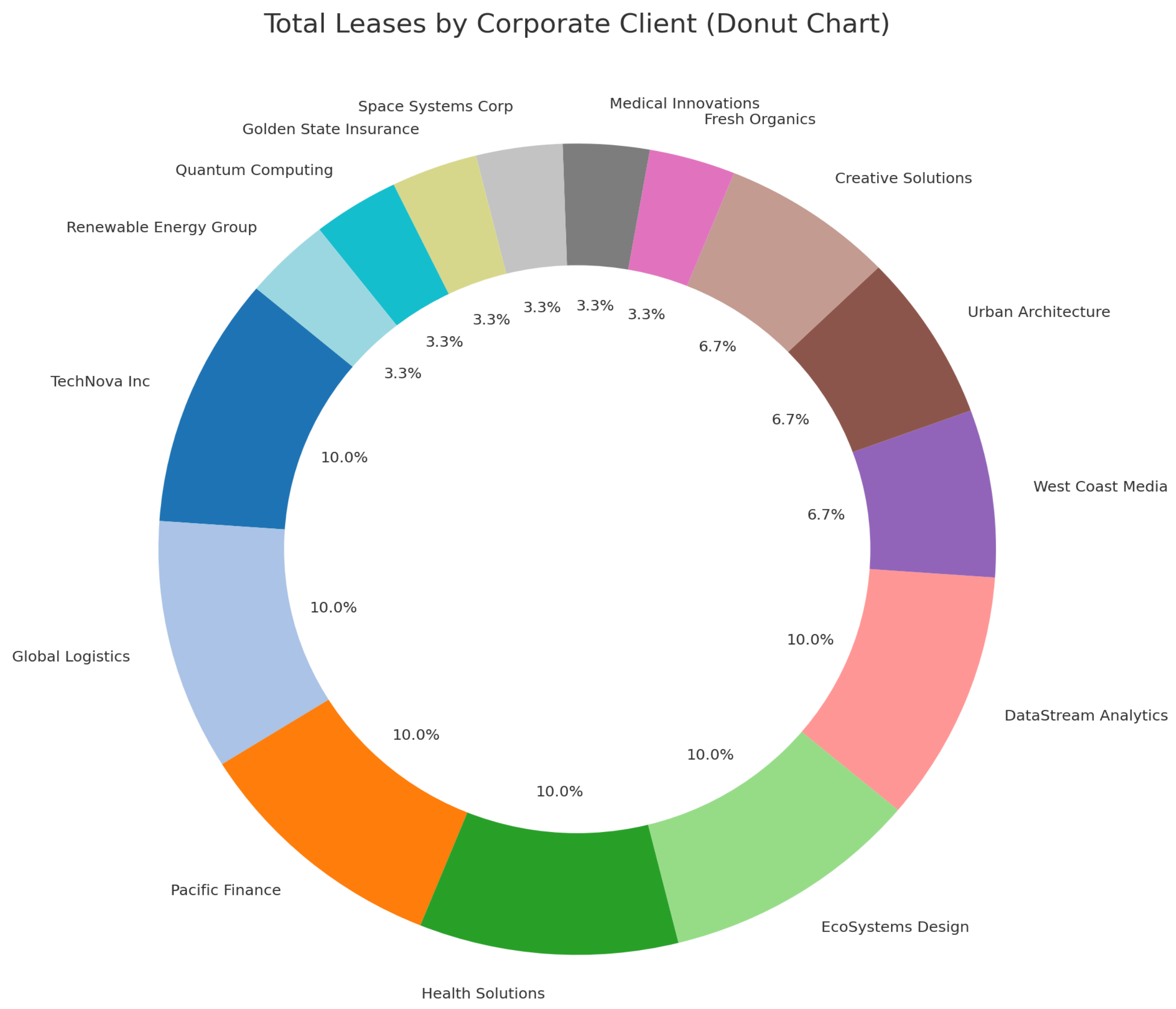
Repeated Corporate Clients

Business Question: Which corporate clients have signed multiple leases with us?

```
SELECT *
FROM Summary_RepeatClients
ORDER BY TotalLeases DESC;
```



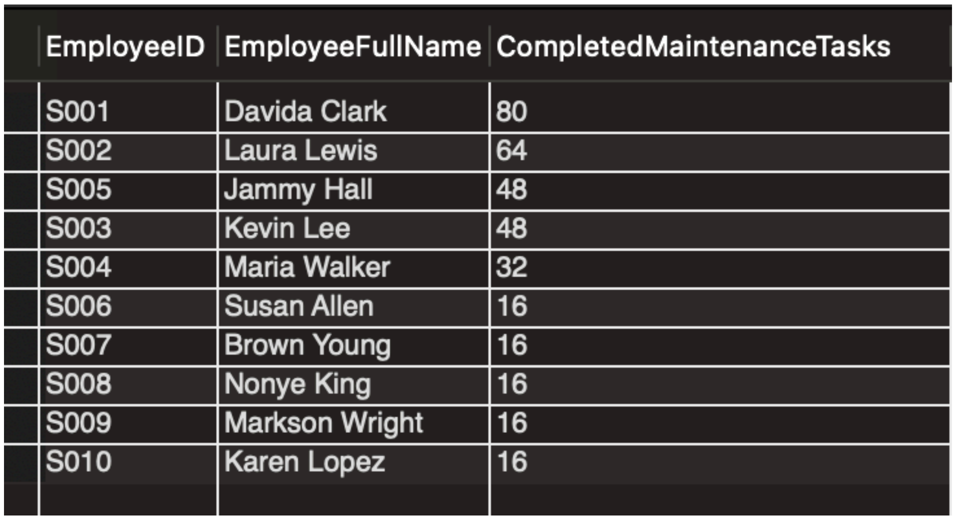
CorpClientName	TotalLeases
TechNova Inc	48
Global Logistics	48
Pacific Finance	48
Health Solutions	48
EcoSystems Design	48
DataStream Analytics	48
West Coast Media	32
Urban Architecture	32
Creative Solutions	32
Fresh Organics	16
Medical Innovations	16
Space Systems Corp	16
Golden State Insurance	16
Quantum Computing	16
Renewable Energy Group	16


Total Leases by Corporate Client (Donut Chart)
Space Systems Corp
Medical Innovations
Fresh Organics
Golden State Insurance
Quantum Computing
Creative Solutions
Renewable Energy Group
Urban Architecture
TechNova Inc
West Coast Media
Global Logistics
DataStream Analytics
Pacific Finance
EcoSystems Design
Health Solutions
3.3%
3.3%
3.3%
3.3%
3.3%
3.3%
6.7%
6.7%
6.7%
10.0%
10.0%
10.0%
10.0%
10.0%
10.0%

Staff Maintenance Performance

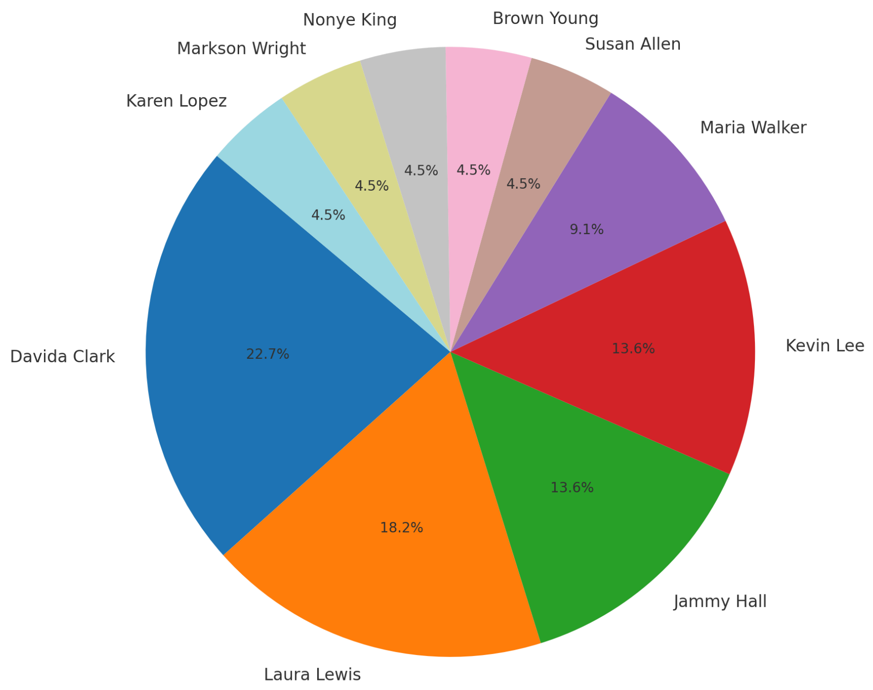
Business Question: Which staff members complete the most maintenance work?

```
SELECT *

FROM Summary_MaintenancePerformanceByStaff

ORDER BY CompletedMaintenanceTasks DESC;
```



EmployeeID	EmployeeFullName	CompletedMaintenanceTasks
S001	Davida Clark	80
S002	Laura Lewis	64
S005	Jammy Hall	48
S003	Kevin Lee	48
S004	Maria Walker	32
S006	Susan Allen	16
S007	Brown Young	16
S008	Nonye King	16
S009	Markson Wright	16
S010	Karen Lopez	16

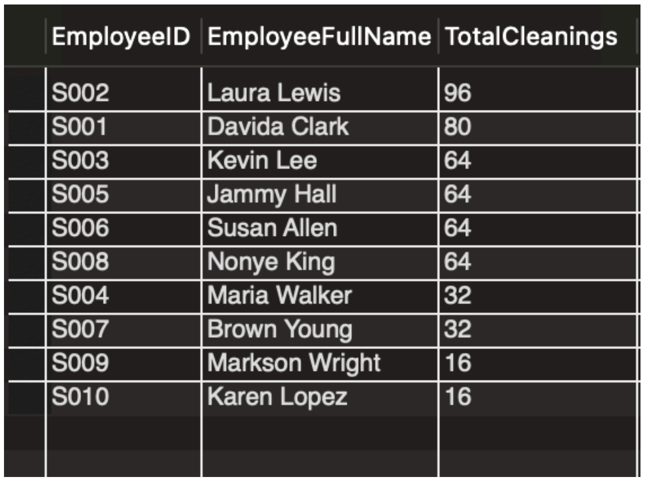
Proportion of Completed Maintenance Tasks by Employee (Unique Colors)



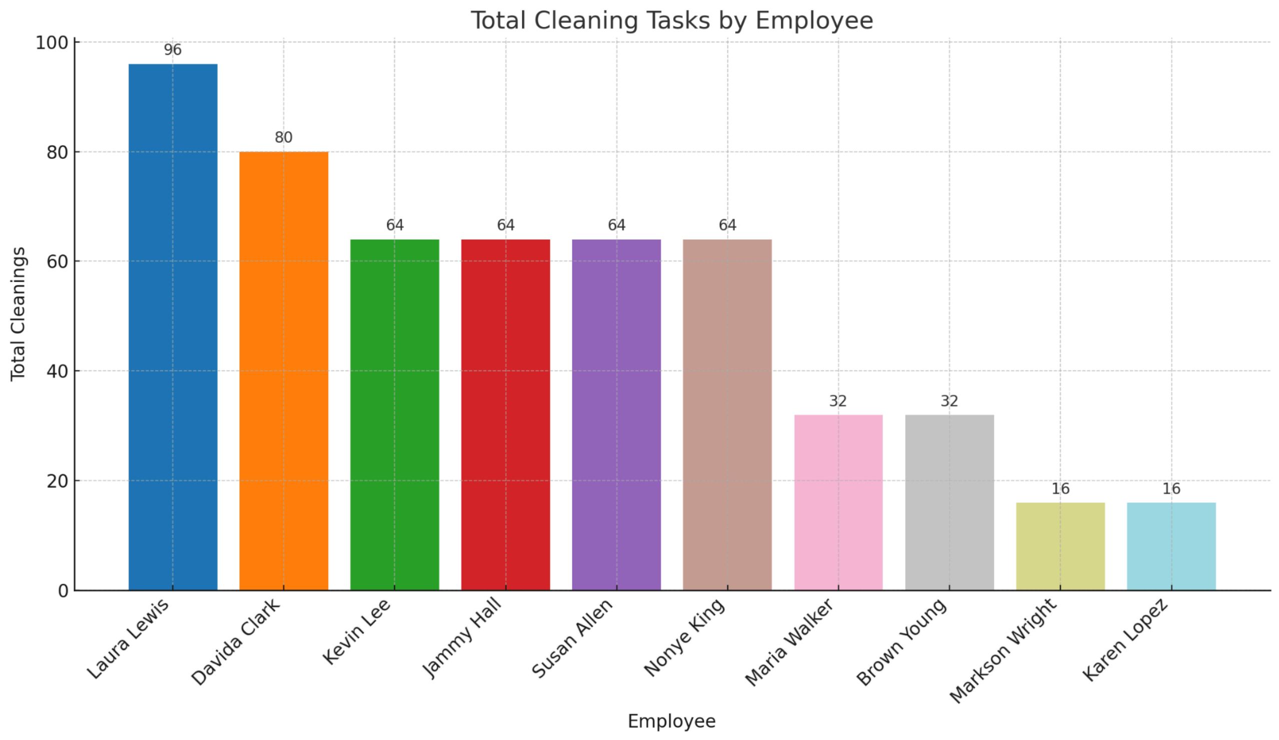
Cleaning Staff Productivity

Business Question: Which staff members are most active in cleaning services?

```
SELECT *

FROM Summary_CleaningPerformanceByStaff

ORDER BY TotalCleanings DESC;
```



EmployeeID	EmployeeFullName	TotalCleanings
S002	Laura Lewis	96
S001	Davida Clark	80
S003	Kevin Lee	64
S005	Jammy Hall	64
S006	Susan Allen	64
S008	Nonye King	64
S004	Maria Walker	32
S007	Brown Young	32
S009	Markson Wright	16
S010	Karen Lopez	16



Deliverable:

Submit a word file that contain the results (e.g., diagram and schema, SQL codes with screenshots of the results, and the tables created). The peer evaluation form is required.

[Bonus points] (5 points) Visualize the BI reports (e.g., using tools like Excel, Power BI, or Tableau).   This would help you understand the importance of visualization in decision-making.