SACRAMENTO STATE
COLLEGE OF CONTINUING EDUCATION

MSBA202

Data Management for Business Analytics

Section - 70

Prepared by – Group 1

Md Tarbiar Swat - 304454332
Bindu Nithyananda – 304429021
Hashim Gilani - 304439369
Suleman Ahmad - 304467046

Group Project - Part 1 & 2

Submission Date
May 8, 2025

Prepared for
Beom-Jin Choi

PART 1

1.0 Business Objective

To upgrade the existing operational database for a property management company by enhancing data integrity, reducing redundancy, and providing improved support for complex queries. The upgraded system will effectively manage detailed information on buildings, apartments, managers, staff, inspections, leases, maintenance requests, and corporate clients. In addition, the solution will support an analytical database for historical analysis and decision-making, enabling the company to monitor leasing trends, maintenance costs, and overall property performance.

1.1 Database & Business Requirement

- **Operational Database Requirements:**
  - **Manage Buildings & Managers:**
    Record details of buildings and the managers who oversee them, including tracking which building a manager resides in.
  - **Apartment & Leasing Management:**
    Capture apartment details (number of bedrooms, occupancy status) and manage leasing information for corporate clients. This includes tracking lease start/end dates, monthly rent, and security deposits.
  - **Maintenance Requests:**
    Record maintenance requests from tenants with details such as request date, description, and status.
  - **Inspections:**
    Manage and track building inspections by inspectors, including inspection dates and next scheduled inspection dates.
  - **Cleaning Operations:**
    Track cleaning assignments by linking staff members to the apartments they clean.

1.2 List of Entities and Attributes

- **Building** (Building_ID,Street,State,City ZipCode,Manager_ID)
- **Manager** (Manager ID,MFull_Name,Salary, MPhone,MEmail,Building_ID)
- **Apartment** (Apartment_ID, No_of_bedrooms, Rental_status, Building_ID)
- **MaintenanceRequest** (Request_ID, Building_ID, AptNo, Request_Date, Description, Status)
- **Inspector** (Inspector_ID,InspectorName,IPhone,IEmail)
- **Inspects**(Inspector_ID,Insp_date,next_insp,Building_ID)
- **Staff** (Staff_ID, S_Name, SPhone,SEmail)
- **Cleans** (Apartment_ID, Staff_ID, Building_ID)

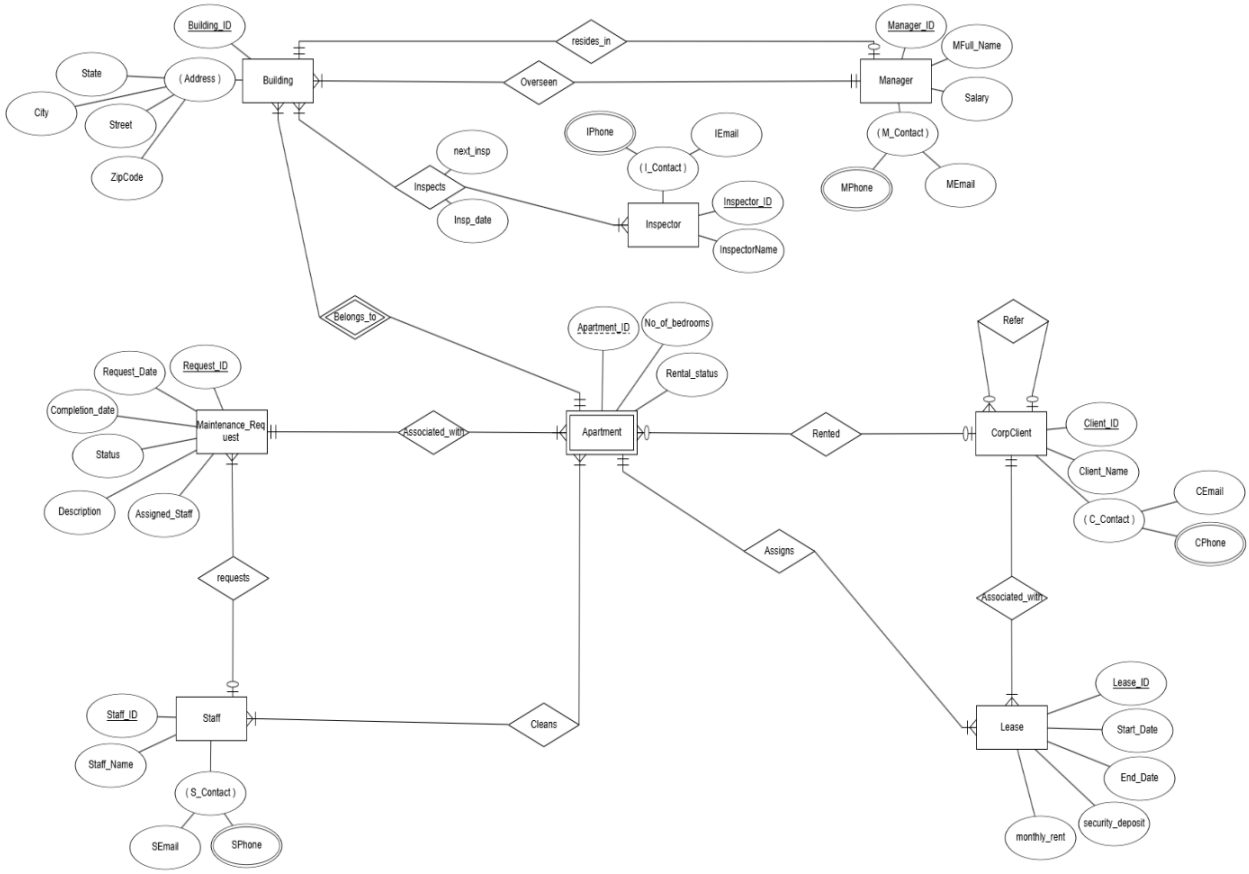
- **Resident_Feedback** (Feedback_ID, Feedback_Date, Category, Description, Status, Apartment_ID)
- **CorpClient** (Client_ID, Client_Name, CEmail,CPhone)
- **Lease** (Lease_ID, Building_ID, AptNo, Client_ID, LeaseStartDate, LeaseEndDate, MonthlyRent, SecurityDeposit)

2.0 ER Diagram

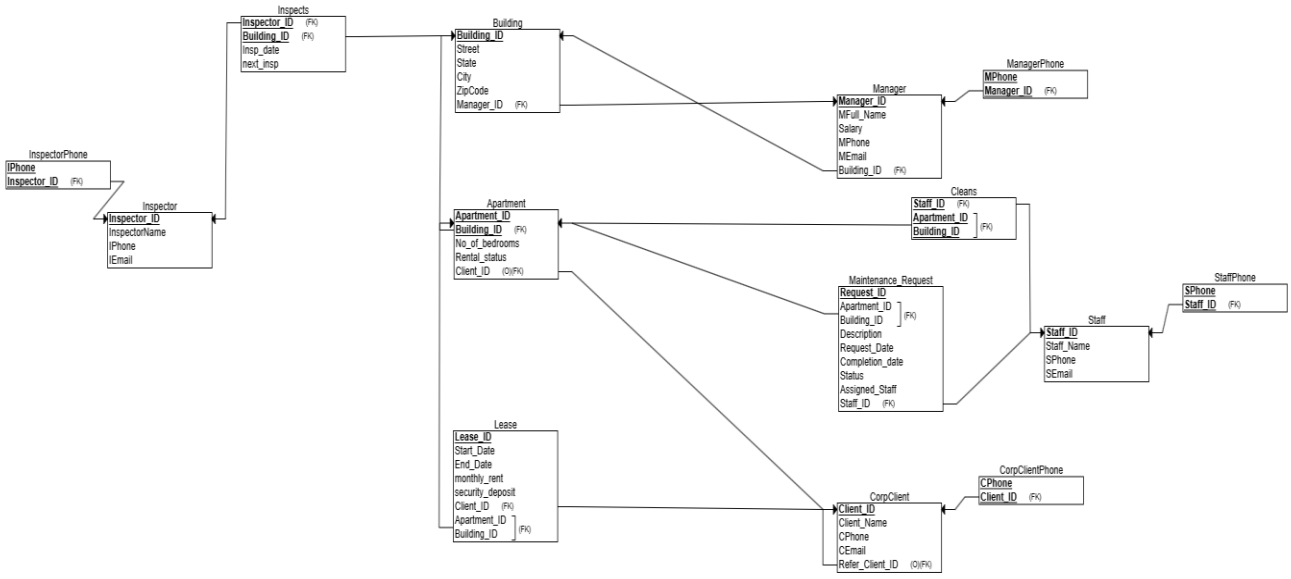
The ER Diagram visually represents the conceptual design of the property management system. It illustrates all major entities—such as Building, Manager, Apartment, MaintenanceRequest, Lease, CorpClient, Inspector, Staff, and related associative entities like Overseen, Inspection, and Cleans—and the relationships between them.

2.1 Create an ER diagram and the relational schema for the database (20 Points)

2.1.1 Entity-Relationship Diagram: PROPERTY MANAGEMENT SYSTEM



2.1.2 Relational-Schema: PROPERTY MANAGEMENT SYSTEM



2.2 Create the data dictionary (an example is shown below) that documents metadata about tables and related constraints. (15 points)

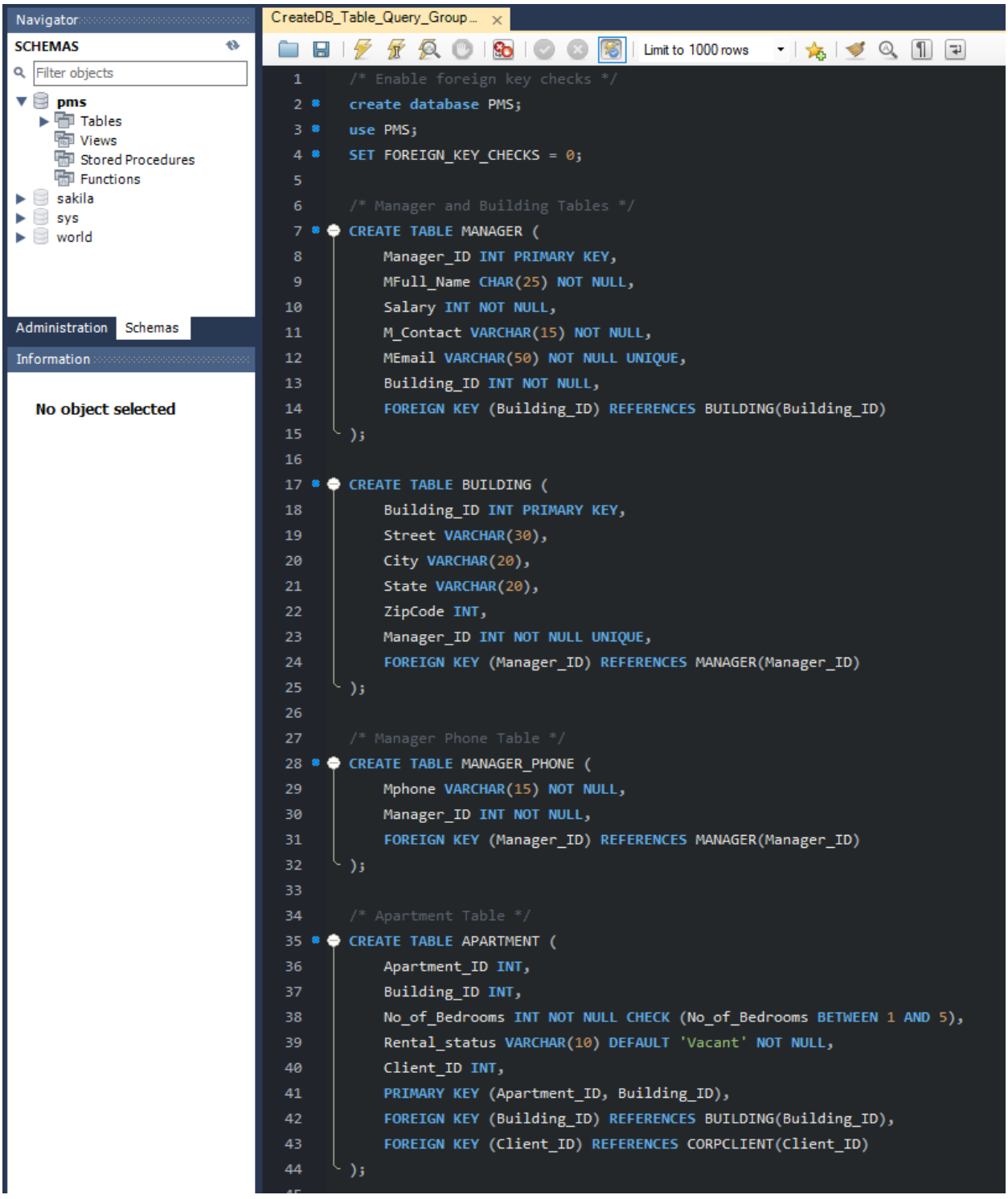
2.2.1 Data-Dictionary: PROPERTY MANAGEMENT SYSTEM

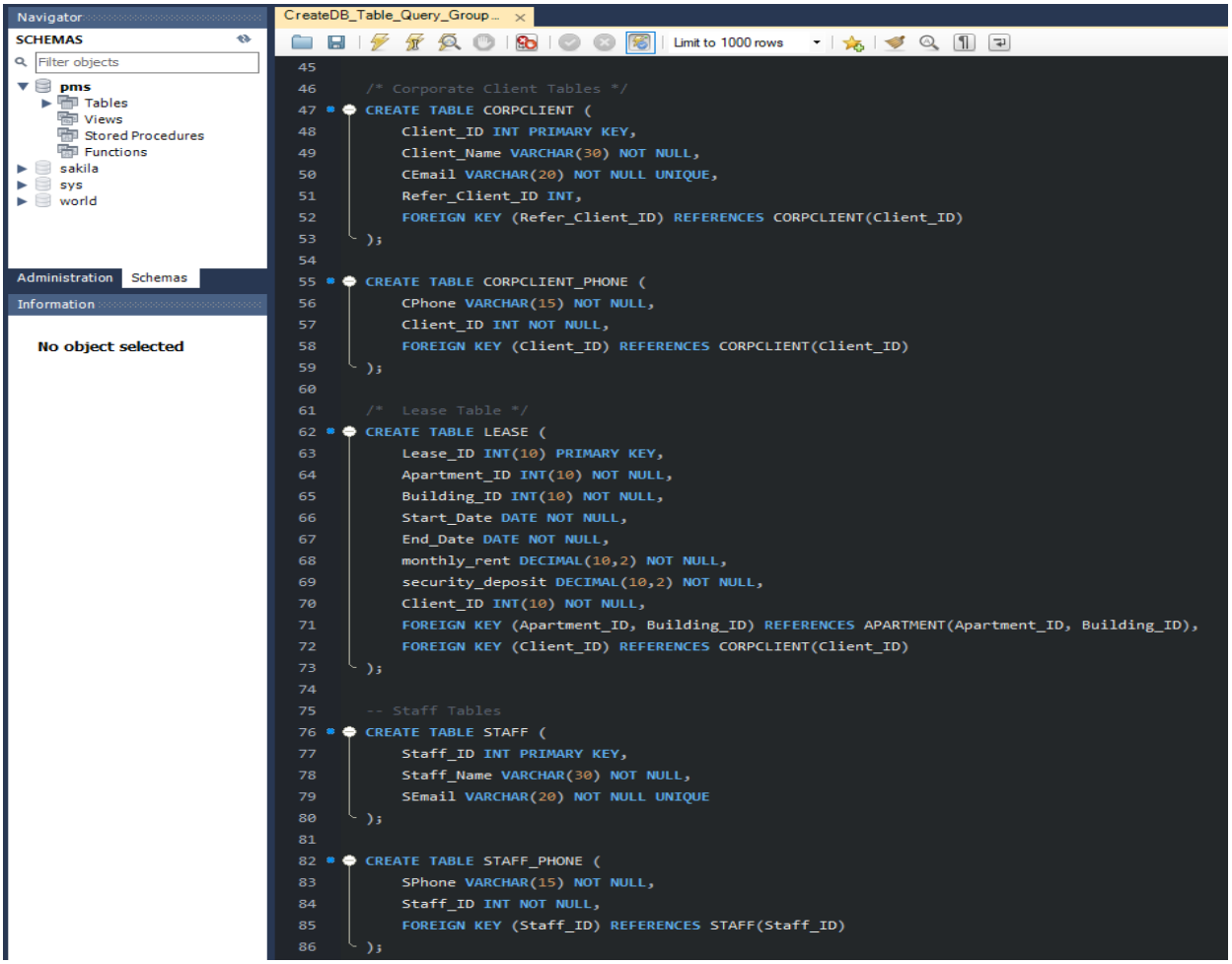
Entity:	**MANAGER TABLE**	Contains all data about property managers.					
Field Name	**Description**	**Type**	**Specifications**	**Default**	**Required**	**Unique**	**Key(s)**
Manager_ID	Unique manager ID	INT	10 digits, unique ID	-	YES	YES	PK
MFull_Name	Manager's full name	CHAR	25 characters	-	YES	NO	-
Salary	Manager's annual salary	INT	20 digits	-	YES	NO	-
M_Contact	Manager's contact number	VARCHAR	15 alpha numeric characters	-	YES	NO	-
MEmail	Manager's email address	VARCHAR	50 alpha numeric characters		YES	YES	
Building_ID	Building where the manager resides	INT	10 digits	-	YES	NO	FK
	MANAGER PHONE TABLE						
Mphone	Phone number of Manager	INT	10 digits	-	YES	NO	Null Allowed
Manager_ID	Unique ID of Manager	INT	10 digits, unique ID	-	YES	YES	FK
Entity:	**BUILIDING TABLE**	Stores building details and assigned manager.					
Field Name	**Description**	**Type**	**Specifications**	**Default**	**Required**	**Unique**	**Key(s)**
Building_ID	Building ID	INT	10 digits, unique ID	-	YES	YES	PK
Street	Building street address	VARCHAR	30 alpha numeric characters				
City	Building in which city	VARCHAR	20 alpha numeric characters				
State	Building in which state	VARCHAR	20 alpha numeric characters				
ZipCode	Building postal code	INT	10 digits				
Manager_ID	Assigned Manager ID	INT	10 digits, unique ID	-	YES	YES	FK
Entity:	**APARTMENT TABLE**	Tracks apartments within buildings and their status.					
Field Name	**Description**	**Type**	**Specifications**	**Default**	**Required**	**Unique**	**Key(s)**
Apartment_ID	Apartment Number	INT	10 digits, unique ID	-	YES	YES	PK (Composite)
Building_ID	Building ID	INT	10 digits, unique ID	-	YES	NO	PK(Composite)
No_of_Bedrooms	Number of Bedrooms	INT	2 digits, (1 - 5 Bedrooms)	-	YES	NO	-
Rental_status	Occupancy Status	VARCHAR	Occupied or Vacant	Vacant	YES	NO	-
Client_ID	Corporate Client ID	INT	10 digits (if occupied)	NULL	NO	NO	FK
Entity:	**CORPCLIENT TABLE**	Tracks apartments within buildings and their status.					
Field Name	**Description**	**Type**	**Specifications**	**Default**	**Required**	**Unique**	**Key(s)**
Client_ID	Corporate Client ID	INT	10 digits, unique ID	-	YES	YES	PK
Client_Name	Client Name	VARCHAR	30 alpha numeric characters	-	YES	NO	-
CEmail	Client Phone Number	VARCHAR	20 alpha numeric characters	-	YES	YES	-
Refer_Client_ID	Referrer Client ID	INT	10 digits (if referred)	NULL	NO	NO	FK

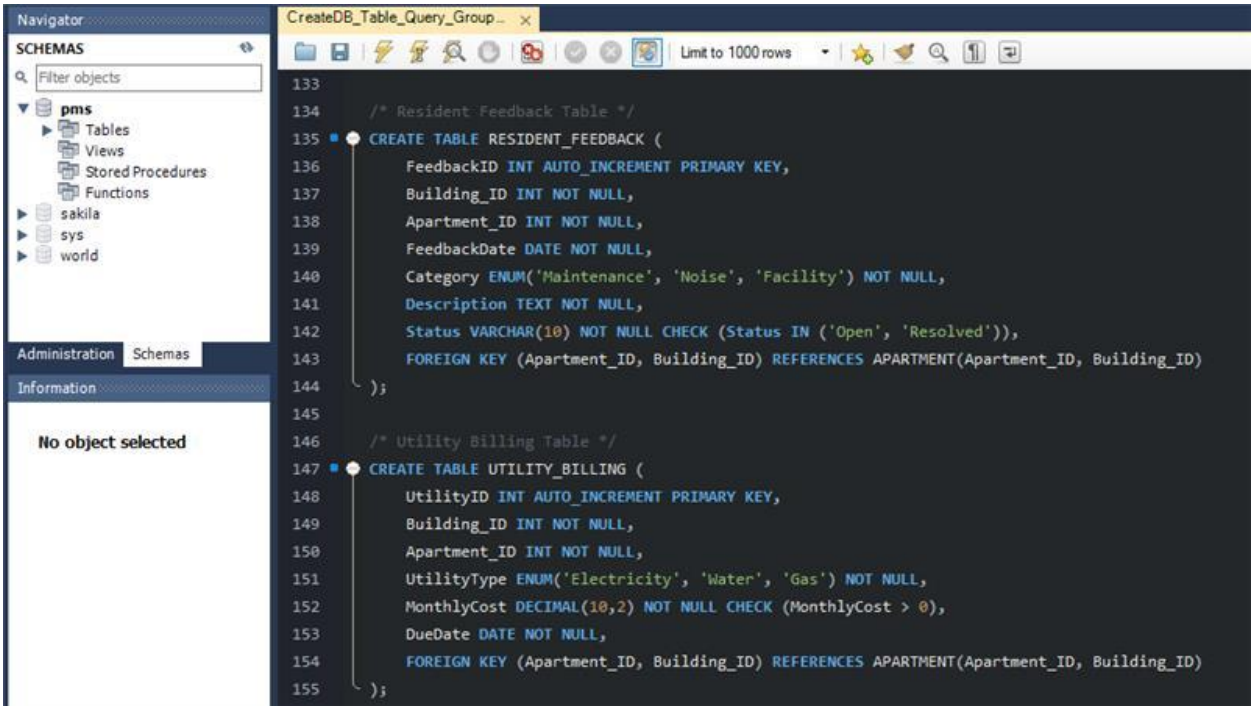
	CORPCLIENT PHONE TABLE						
CPhone	Phone number of corporate client	INT	10 digits	-	YES	NO	Null Allowed
Client_ID	Unique ID of corporate client	INT	10 digits, unique ID	-	YES	YES	FK
Entity:	**LEASE TABLE**	Tracks apartment lease agreements.					
Field Name	**Description**	**Type**	**Specifications**	**Default**	**Required**	**Unique**	**Key(s)**
Lease_ID	Lease ID	INT	10 digits, unique ID	-	YES	YES	PK
Apartment_ID	Apartment Number	VARCHAR	10 digits	-	YES	NO	FK (Composite)
BuildingID	Building ID	INT	10 digits	-	YES	NO	FK (Composite)
Start_Date	Lease Start Date	DATE	Format: YYYY-MM-DD	-	YES	NO	-
End_Date	Lease End Date	DATE	Format: YYYY-MM-DD	-	YES	NO	-
monthly_rent	Monthly Rent	DECIMAL(10,2)	Currency	-	YES	NO	-
security_deposit	Security Deposit	DECIMAL(10,2)	Currency	-	YES	NO	-
Client_ID	Corporate Client ID	INT	10 digits	-	YES	NO	FK
Entity:	**STAFF TABLE**	Records staff members responsible for cleaning.					
Field Name	**Description**	**Type**	**Specifications**	**Default**	**Required**	**Unique**	**Key(s)**
Staff_ID	Staff ID	INT	10 digits, unique ID	-	YES	YES	PK
Staff_Name	Staff Name	VARCHAR	30 alpha numeric characters	-	YES	NO	-
SEmail	Staff email address	VARCHAR	20 alpha numeric characters	-	YES	YES	-
	STAFF PHONE TABLE						
SPhone	Phone number of staff	INT	10 digits	-	YES	NO	Null Allowed
Staff_ID	Unique ID of staff	INT	10 digits, unique ID	-	YES	YES	FK
Entity:	**INSPECTOR TABLE**	Tracks building inspectors.					
Field Name	**Description**	**Type**	**Specifications**	**Default**	**Required**	**Unique**	**Key(s)**
Inspector_ID	Inspector ID	INT	10 digits, unique ID	-	YES	YES	PK
InspectorName	Inspector Name	VARCHAR	30 alpha numeric characters	-	YES	NO	-
IEmail	Inspector email address	VARCHAR	20 alpha numeric characters	-	YES	YES	-
	INSPECTOR PHONE TABLE						
IPhone	Phone number of inspector	INT	10 digits	-	YES	NO	Null Allowed
Inspector_ID	Unique ID of inspector	INT	10 digits, unique ID	-	YES	YES	FK
Entity:	**INSPECTS TABLE**	Logs building inspections and schedules.					
Field Name	**Description**	**Type**	**Specifications**	**Default**	**Required**	**Unique**	**Key(s)**
Inspector_ID	Inspector ID	INT	10 digits, unique ID	-	YES	YES	PK
Building_ID	Building ID	INT	10 digits	-	YES	NO	FK
Insp_date	Inspection date	DATE	Format: YYYY-MM-DD	-	YES	NO	-
Insp_date	Inspection date	DATE	Format: YYYY-MM-DD	-	YES	NO	-
next_insp	Next inspection date	DATE	Format: YYYY-MM-DD	-	YES	NO	-
Entity:	**CLEANS TABLE**	Assigns staff to clean specific apartments.					
Field Name	**Description**	**Type**	**Specifications**	**Default**	**Required**	**Unique**	**Key(s)**
Staff_ID	Identifier of the staff member	INT	10 digits	-	YES	NO	FK
Apartment_ID	Identifier of the apartment	INT	10 digits	-	YES	NO	FK (Composite)
Building_ID	Building ID	INT	10 digits	-	YES	NO	FK (Composite)
Entity:	**MAINTENANCE REQUEST TABLE**	Records tenant maINTenance requests.					
Field Name	**Description**	**Type**	**Specifications**	**Default**	**Required**	**Unique**	**Key(s)**
Request_ID	Unique id for each request	INT	10 digits, unique ID	-	YES	YES	PK
Building_ID	Identifier of the building	INT	10 digits	-	YES	NO	FK (Composite)
Apartment_ID	Apartment Number	INT	10 digits	-	YES	NO	FK (Composite)
Request_Date	Date of Request	DATE	Format: YYYY-MM-DD	-	YES	NO	-
Completion_date	Date of completion	DATE	Format: YYYY-MM-DD	-	NO	NO	
Status	Maintenance status	VARCHAR	20 alpha numeric characters	-	YES	NO	
Assigned_Staff	Staff Member Assigned	VARCHAR	30 alpha numeric characters	-	YES	NO	FK
Staff_ID	Request Status	INT	10 digits	-	NO	NO	-
Entity:	**RESIDENT FEEDBACK TABLE**	Track utility billing of apartments					
Field Name	**Description**	**Type**	**Specifications**	**Default**	**Required**	**Unique**	**Key(s)**
FeedbackID	Unique feedback ID	INT	Auto-increment	-	YES	YES	PK
BuildingID	Building ID	INT	Links to Apartment	-	YES	NO	-
AptNo	Apartment number	INT	Links to Apartment	-	YES	NO	-
FeedbackDate	Date of feedback submission	DATE	Format: YYYY-MM-DD	-	YES	NO	-
Category	Feedback category	ENUM	MaINTenance, Noise, Facility	-	YES	NO	-
Description	Feedback details	TEXT		-	YES	NO	-
Status	Feedback resolution status	VARCHAR(10)	CHECK: Open/Resolved	-	YES	NO	-
Entity:	**UTILITY BILLING**	Track feedbacks from residents					
Field Name	**Description**	**Type**	**Specifications**	**Default**	**Required**	**Unique**	**Key(s)**
UtilityID	Unique utility bill ID	INT	Auto-increment	-	YES	YES	PK
BuildingID	Building ID	INT	Links to Apartment	-	YES	NO	-
AptNo	Apartment number	INT	Links to Apartment	-	YES	NO	-
UtilityType	Type of utility	ENUM	Electricity, Water, Gas	-	YES	NO	-
MonthlyCost	Monthly utility cost	DECIMAL(10,2)	Positive value (e.g., 150.00)	-	YES	NO	-
DueDate	Pavment due date	DATE	Format: YYYY-MM-DD	-	YES	NO	-

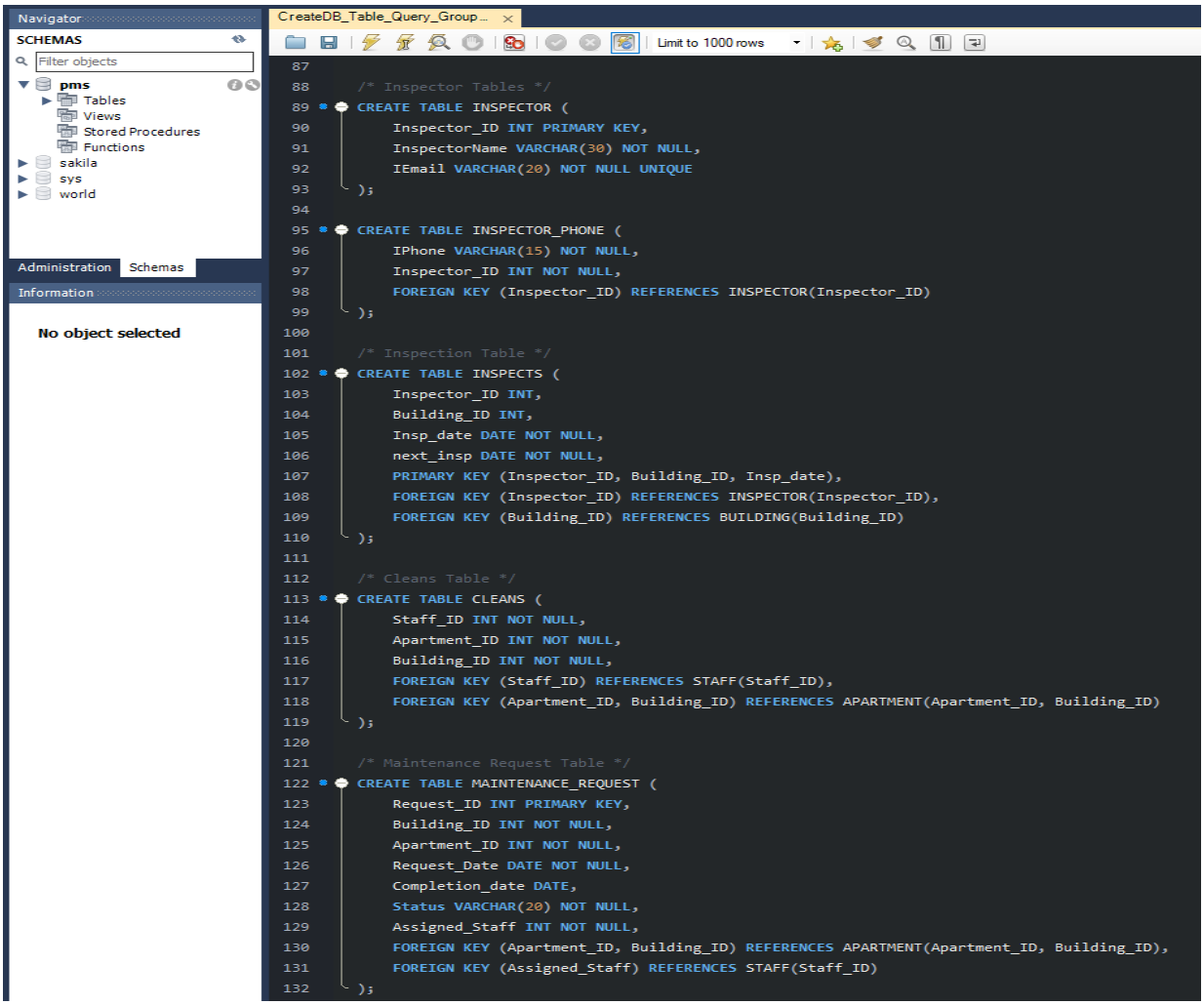
2.3 Populate the database with sample data using SQL DDL codes (at least 10 rows). (10 points)

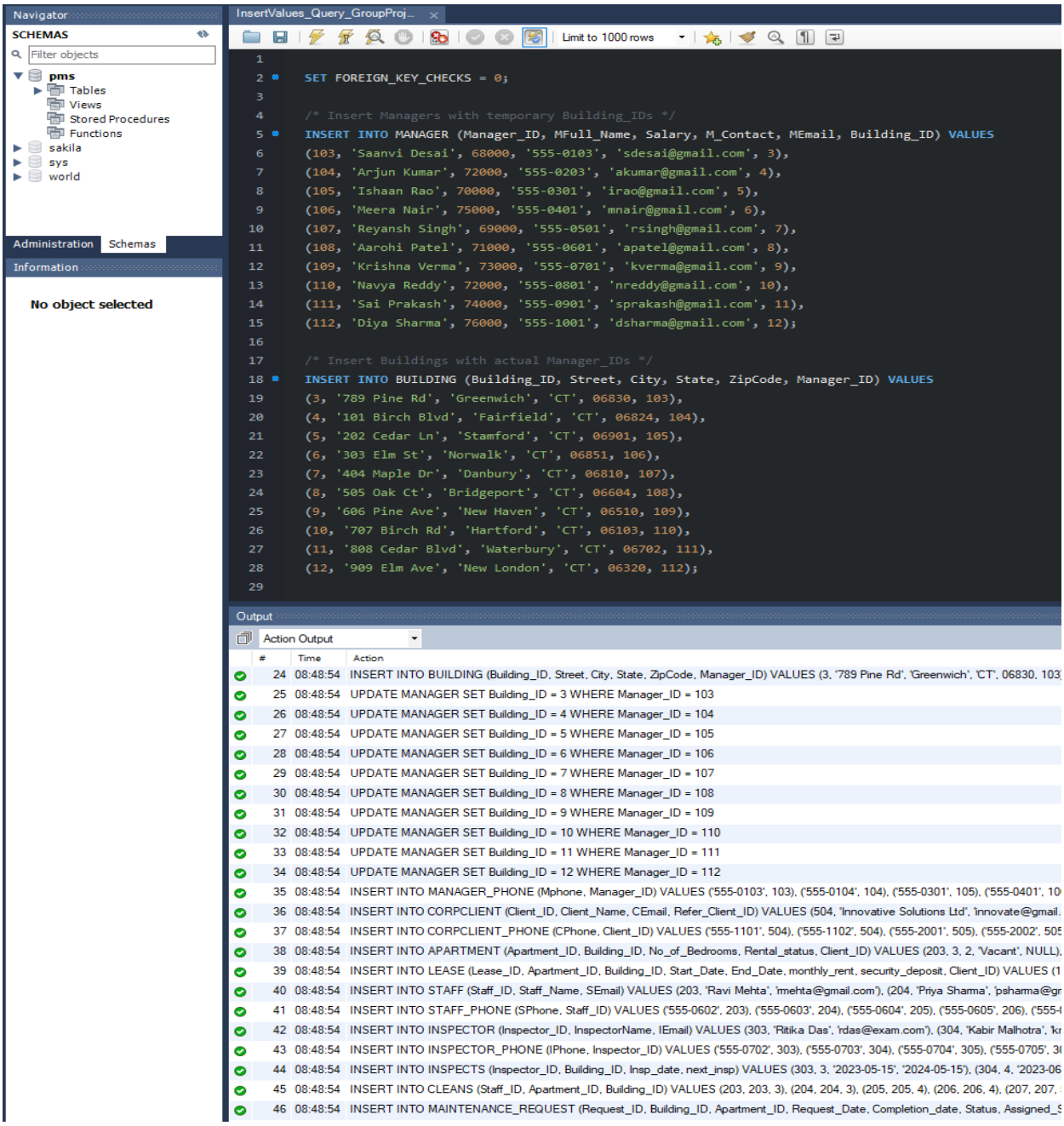
2.3.1 SQL Database creation and table population



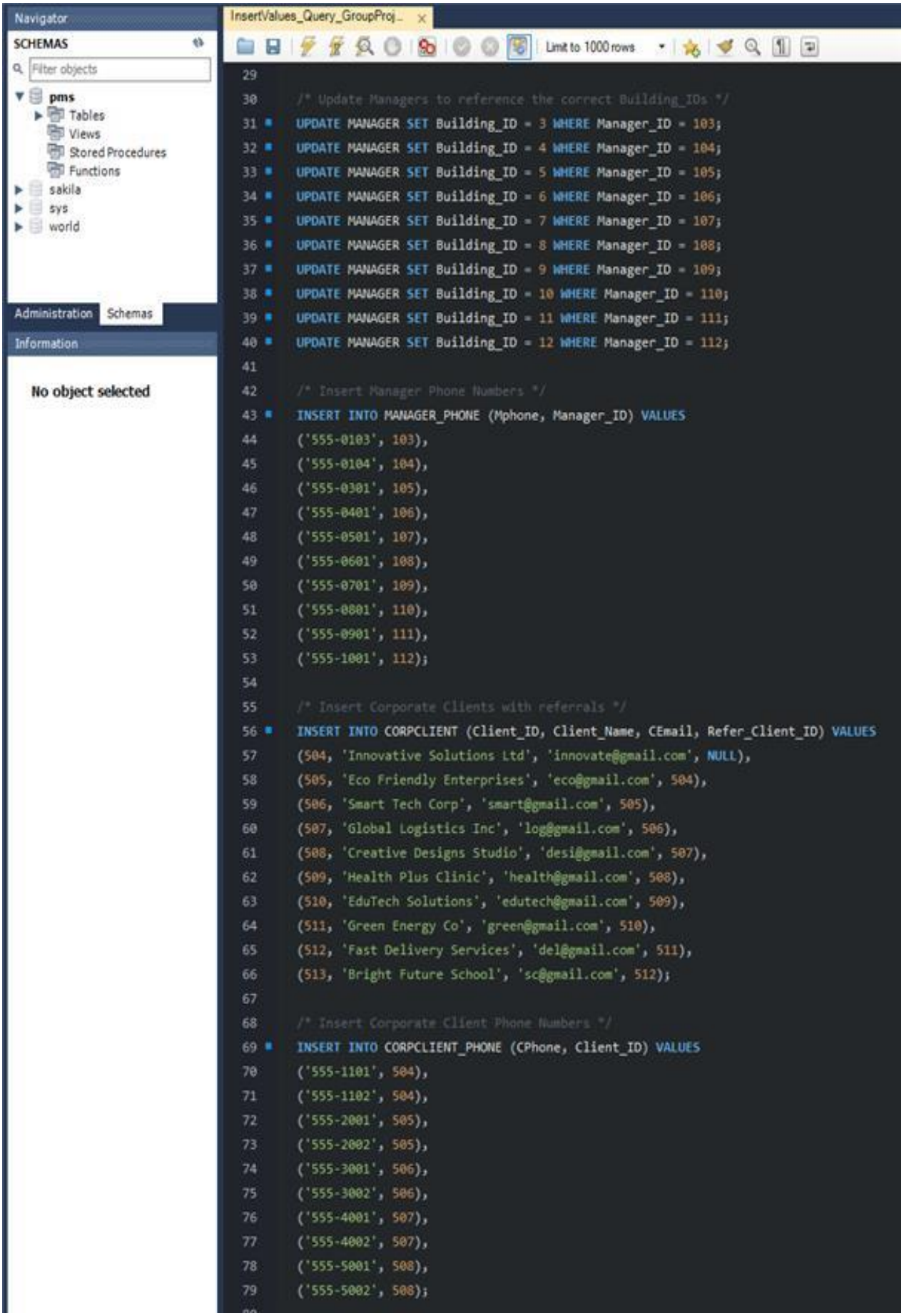

Navigator
SCHEMAS
Filter objects
pms
Tables
Views
Stored Procedures
Functions
sakila
sys
world
Administration
Schemas
Information
No object selected
CreateDB_Table_Query_Group...
Limit to 1000 rows
/* Corporate Client Tables */
CREATE TABLE CORPCLIENT (
Client_ID INT PRIMARY KEY,
Client_Name VARCHAR(30) NOT NULL,
CEmail VARCHAR(20) NOT NULL UNIQUE,
Refer_Client_ID INT,
FOREIGN KEY (Refer_Client_ID) REFERENCES CORPCLIENT(Client_ID)
);
CREATE TABLE CORPCLIENT_PHONE (
CPhone VARCHAR(15) NOT NULL,
Client_ID INT NOT NULL,
FOREIGN KEY (Client_ID) REFERENCES CORPCLIENT(Client_ID)
);
/* Lease Table */
CREATE TABLE LEASE (
Lease_ID INT(10) PRIMARY KEY,
Apartment_ID INT(10) NOT NULL,
Building_ID INT(10) NOT NULL,
Start_Date DATE NOT NULL,
End_Date DATE NOT NULL,
monthly_rent DECIMAL(10,2) NOT NULL,
security_deposit DECIMAL(10,2) NOT NULL,
Client_ID INT(10) NOT NULL,
FOREIGN KEY (Apartment_ID, Building_ID) REFERENCES APARTMENT(Apartment_ID, Building_ID),
FOREIGN KEY (Client_ID) REFERENCES CORPCLIENT(Client_ID)
);
-- Staff Tables
CREATE TABLE STAFF (
Staff_ID INT PRIMARY KEY,
Staff_Name VARCHAR(30) NOT NULL,
SEmail VARCHAR(20) NOT NULL UNIQUE
);
CREATE TABLE STAFF_PHONE (
SPhone VARCHAR(15) NOT NULL,
Staff_ID INT NOT NULL,
FOREIGN KEY (Staff_ID) REFERENCES STAFF(Staff_ID)
);

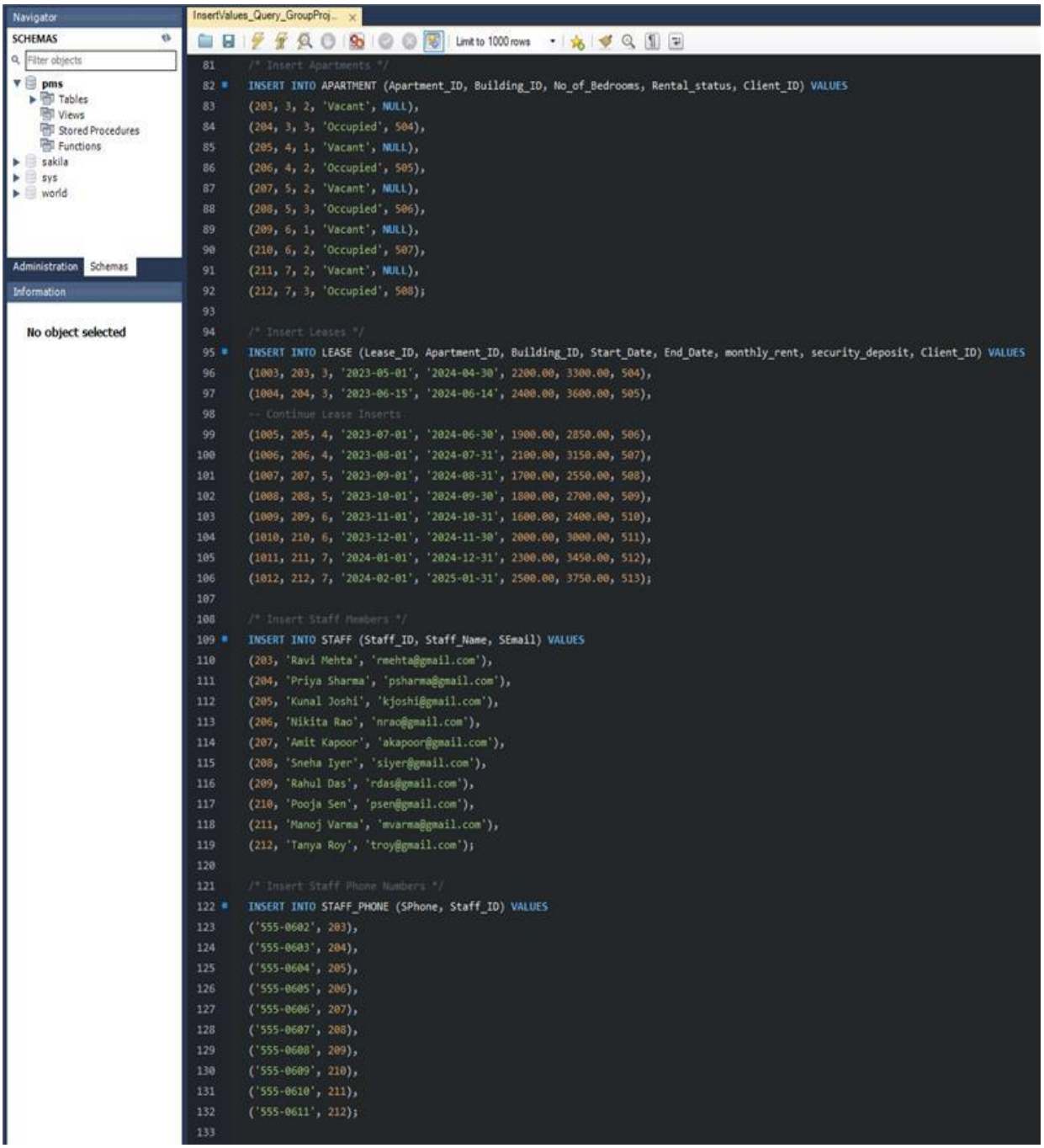

Navigator
SCHEMAS
Filter objects
pms
Tables
Views
Stored Procedures
Functions
sakila
sys
world
Administration
Schemas
Information
No object selected
CreateDB_Table_Query_Group...
Limit to 1000 rows
/* Resident Feedback Table */
CREATE TABLE RESIDENT_FEEDBACK (
FeedbackID INT AUTO_INCREMENT PRIMARY KEY,
Building_ID INT NOT NULL,
Apartment_ID INT NOT NULL,
FeedbackDate DATE NOT NULL,
Category ENUM('Maintenance', 'Noise', 'Facility') NOT NULL,
Description TEXT NOT NULL,
Status VARCHAR(10) NOT NULL CHECK (Status IN ('Open', 'Resolved')),
FOREIGN KEY (Apartment_ID, Building_ID) REFERENCES APARTMENT(Apartment_ID, Building_ID)
);
/* Utility Billing Table */
CREATE TABLE UTILITY_BILLING (
UtilityID INT AUTO_INCREMENT PRIMARY KEY,
Building_ID INT NOT NULL,
Apartment_ID INT NOT NULL,
UtilityType ENUM('Electricity', 'Water', 'Gas') NOT NULL,
MonthlyCost DECIMAL(10,2) NOT NULL CHECK (MonthlyCost > 0),
DueDate DATE NOT NULL,
FOREIGN KEY (Apartment_ID, Building_ID) REFERENCES APARTMENT(Apartment_ID, Building_ID)
);

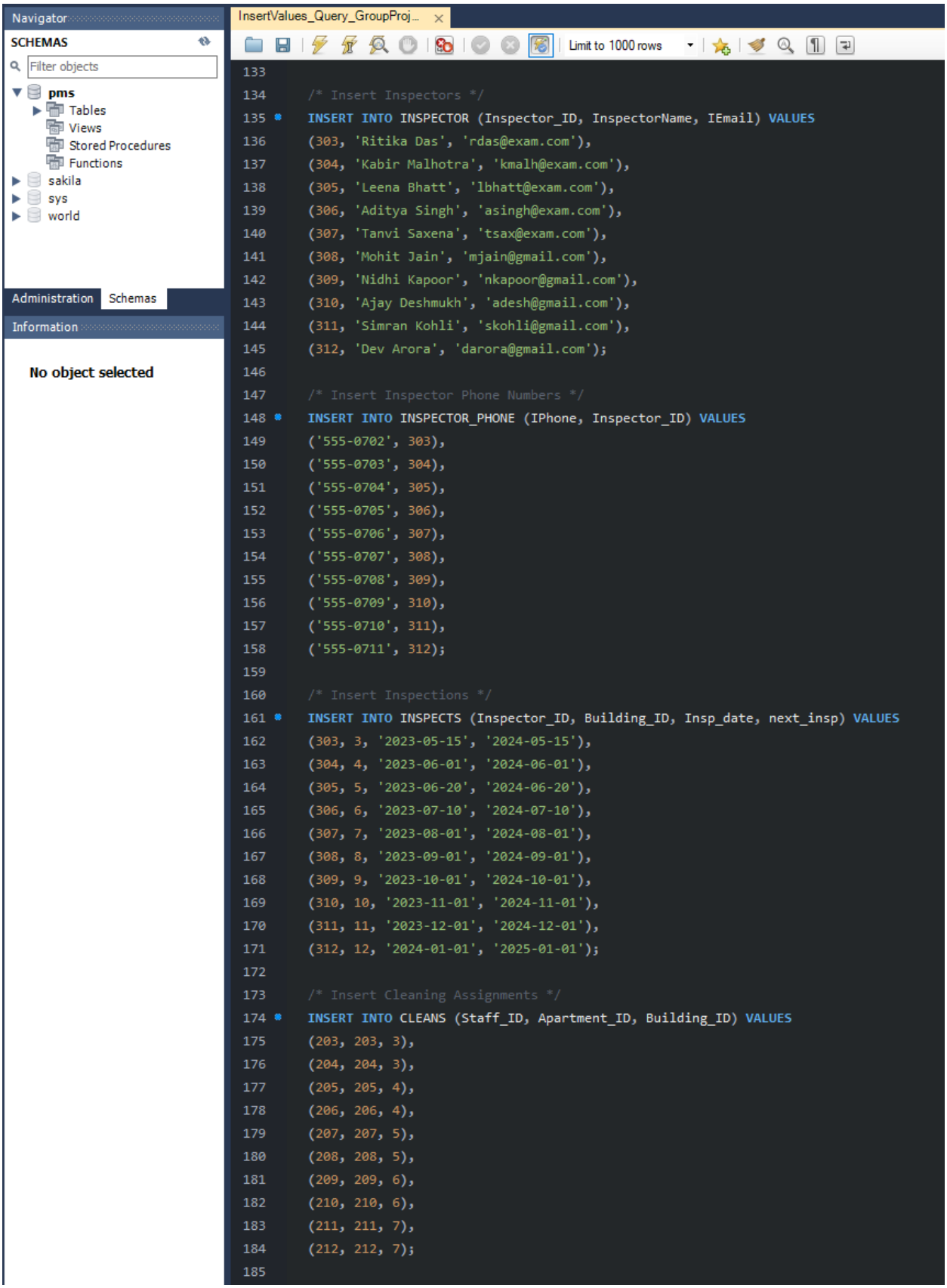

Navigator
SCHEMAS
Filter objects
pms
Tables
Views
Stored Procedures
Functions
sakila
sys
world
Administration
Schemas
Information
No object selected
CreateDB_Table_Query_Group...
Limit to 1000 rows
87
88 /* Inspector Tables */
89 CREATE TABLE INSPECTOR (
90 Inspector_ID INT PRIMARY KEY,
91 InspectorName VARCHAR(30) NOT NULL,
92 IEmail VARCHAR(20) NOT NULL UNIQUE
93);
94
95 CREATE TABLE INSPECTOR_PHONE (
96 IPhone VARCHAR(15) NOT NULL,
97 Inspector_ID INT NOT NULL,
98 FOREIGN KEY (Inspector_ID) REFERENCES INSPECTOR(Inspector_ID)
99);
100
101 /* Inspection Table */
102 CREATE TABLE INSPECTS (
103 Inspector_ID INT,
104 Building_ID INT,
105 Insp_date DATE NOT NULL,
106 next_insp DATE NOT NULL,
107 PRIMARY KEY (Inspector_ID, Building_ID, Insp_date),
108 FOREIGN KEY (Inspector_ID) REFERENCES INSPECTOR(Inspector_ID),
109 FOREIGN KEY (Building_ID) REFERENCES BUILDING(Building_ID)
110);
111
112 /* Cleans Table */
113 CREATE TABLE CLEANS (
114 Staff_ID INT NOT NULL,
115 Apartment_ID INT NOT NULL,
116 Building_ID INT NOT NULL,
117 FOREIGN KEY (Staff_ID) REFERENCES STAFF(Staff_ID),
118 FOREIGN KEY (Apartment_ID, Building_ID) REFERENCES APARTMENT(Apartment_ID, Building_ID)
119);
120
121 /* Maintenance Request Table */
122 CREATE TABLE MAINTENANCE_REQUEST (
123 Request_ID INT PRIMARY KEY,
124 Building_ID INT NOT NULL,
125 Apartment_ID INT NOT NULL,
126 Request_Date DATE NOT NULL,
127 Completion_date DATE,
128 Status VARCHAR(20) NOT NULL,
129 Assigned_Staff INT NOT NULL,
130 FOREIGN KEY (Apartment_ID, Building_ID) REFERENCES APARTMENT(Apartment_ID, Building_ID),
131 FOREIGN KEY (Assigned_Staff) REFERENCES STAFF(Staff_ID)
132);

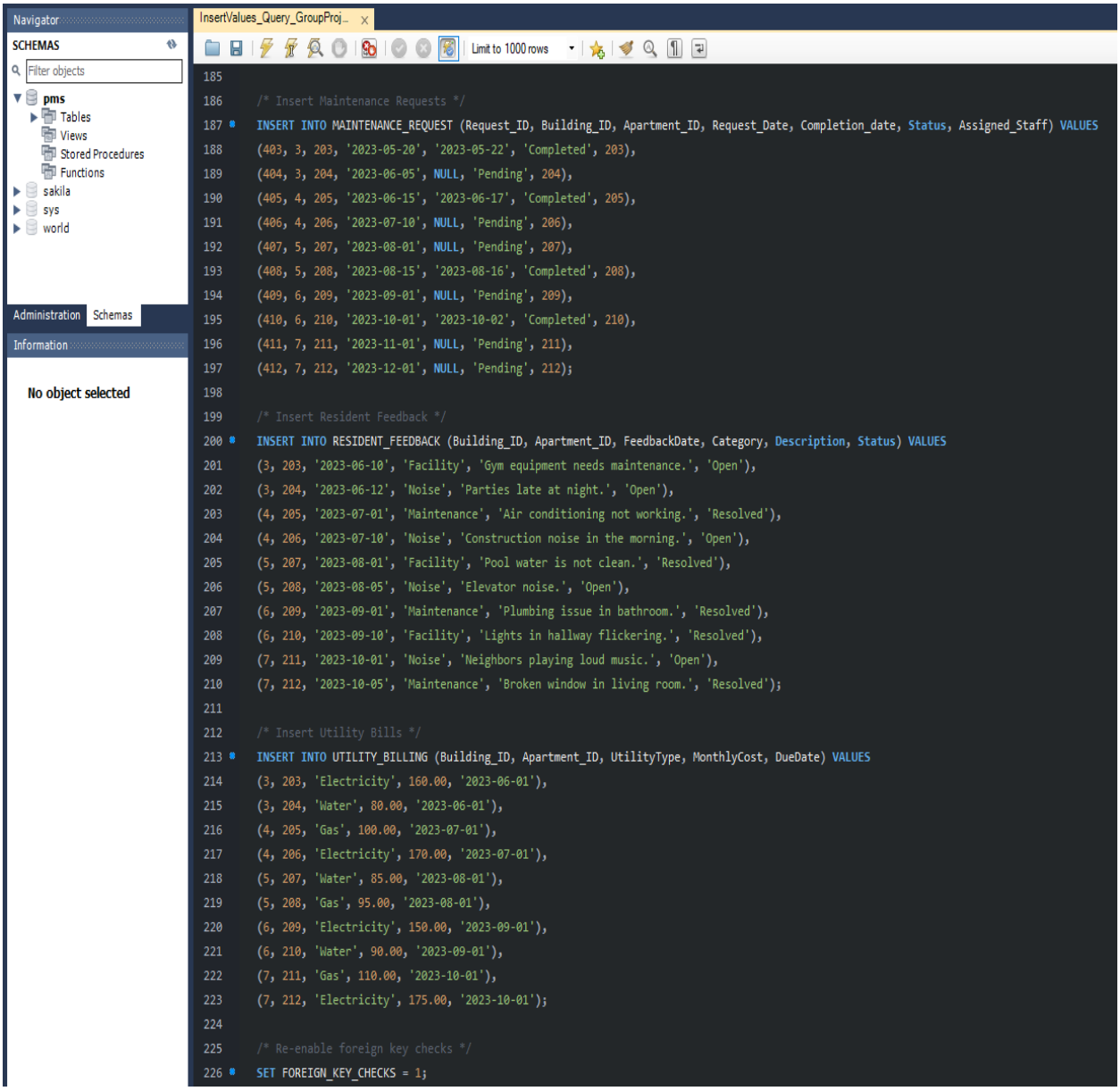


ACTION OUTPUT: SUCCESSFUL QUERY EXECUTION


Navigator
SCHEMAS
Filter objects
pms
Tables
Views
Stored Procedures
Functions
sakila
sys
world
Administration
Schemas
Information
No object selected
InsertValues_Query_GroupProj...
Limit to 1000 rows
/* Update Managers to reference the correct Building_IDs */
UPDATE MANAGER SET Building_ID = 3 WHERE Manager_ID = 103;
UPDATE MANAGER SET Building_ID = 4 WHERE Manager_ID = 104;
UPDATE MANAGER SET Building_ID = 5 WHERE Manager_ID = 105;
UPDATE MANAGER SET Building_ID = 6 WHERE Manager_ID = 106;
UPDATE MANAGER SET Building_ID = 7 WHERE Manager_ID = 107;
UPDATE MANAGER SET Building_ID = 8 WHERE Manager_ID = 108;
UPDATE MANAGER SET Building_ID = 9 WHERE Manager_ID = 109;
UPDATE MANAGER SET Building_ID = 10 WHERE Manager_ID = 110;
UPDATE MANAGER SET Building_ID = 11 WHERE Manager_ID = 111;
UPDATE MANAGER SET Building_ID = 12 WHERE Manager_ID = 112;
/* Insert Manager Phone Numbers */
INSERT INTO MANAGER_PHONE (Mphone, Manager_ID) VALUES
('555-0103', 103),
('555-0104', 104),
('555-0301', 105),
('555-0401', 106),
('555-0501', 107),
('555-0601', 108),
('555-0701', 109),
('555-0801', 110),
('555-0901', 111),
('555-1001', 112);
/* Insert Corporate Clients with referrals */
INSERT INTO CORPCLIENT (Client_ID, Client_Name, CEmail, Refer_Client_ID) VALUES
(504, 'Innovative Solutions Ltd', 'innovate@gmail.com', NULL),
(505, 'Eco Friendly Enterprises', 'eco@gmail.com', 504),
(506, 'Smart Tech Corp', 'smart@gmail.com', 505),
(507, 'Global Logistics Inc', 'log@gmail.com', 506),
(508, 'Creative Designs Studio', 'desi@gmail.com', 507),
(509, 'Health Plus Clinic', 'health@gmail.com', 508),
(510, 'EduTech Solutions', 'edutech@gmail.com', 509),
(511, 'Green Energy Co', 'green@gmail.com', 510),
(512, 'Fast Delivery Services', 'del@gmail.com', 511),
(513, 'Bright Future School', 'sc@gmail.com', 512);
/* Insert Corporate Client Phone Numbers */
INSERT INTO CORPCLIENT_PHONE (CPhone, Client_ID) VALUES
('555-1101', 504),
('555-1102', 504),
('555-2001', 505),
('555-2002', 505),
('555-3001', 506),
('555-3002', 506),
('555-4001', 507),
('555-4002', 507),
('555-5001', 508),
('555-5002', 508);

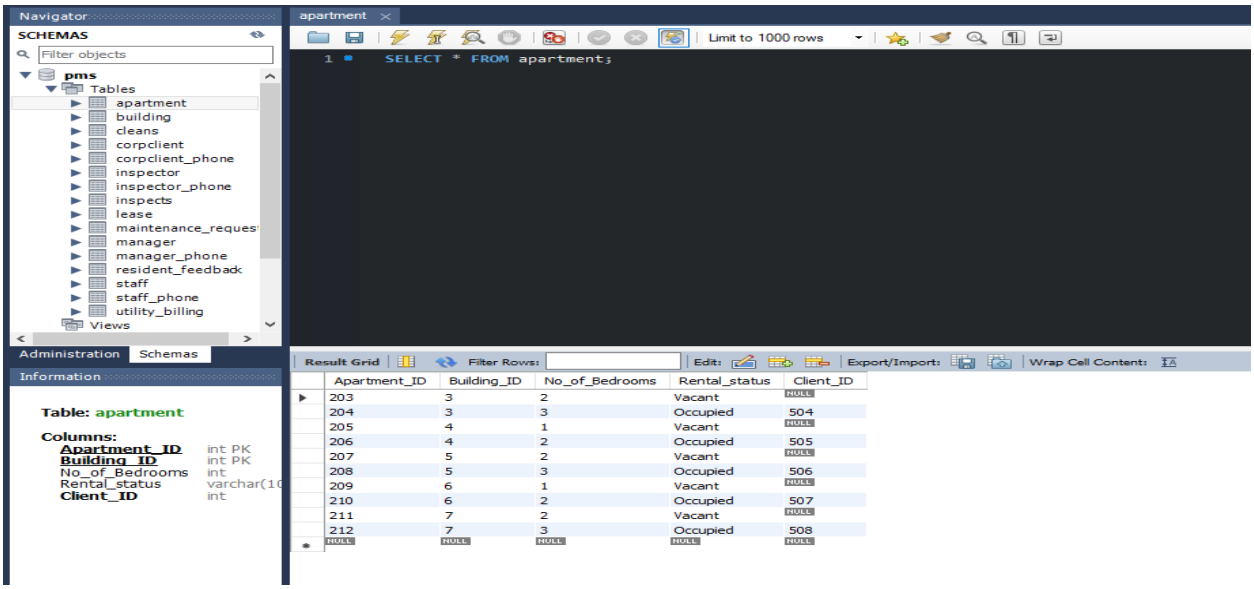

Navigator
SCHEMAS
Filter objects
pms
Tables
Views
Stored Procedures
Functions
sakila
sys
world
Administration
Schemas
Information
No object selected
InsertValues_Query_GroupProj...
Limit to 1000 rows
/* Insert Apartments */
INSERT INTO APARTMENT (Apartment_ID, Building_ID, No_of_Bedrooms, Rental_status, Client_ID) VALUES
(203, 3, 2, 'Vacant', NULL),
(204, 3, 3, 'Occupied', 504),
(205, 4, 1, 'Vacant', NULL),
(206, 4, 2, 'Occupied', 505),
(207, 5, 2, 'Vacant', NULL),
(208, 5, 3, 'Occupied', 506),
(209, 6, 1, 'Vacant', NULL),
(210, 6, 2, 'Occupied', 507),
(211, 7, 2, 'Vacant', NULL),
(212, 7, 3, 'Occupied', 508);
/* Insert Leases */
INSERT INTO LEASE (Lease_ID, Apartment_ID, Building_ID, Start_Date, End_Date, monthly_rent, security_deposit, Client_ID) VALUES
(1003, 203, 3, '2023-05-01', '2024-04-30', 2200.00, 3300.00, 504),
(1004, 204, 3, '2023-06-15', '2024-06-14', 2400.00, 3600.00, 505),
-- Continue Lease Inserts
(1005, 205, 4, '2023-07-01', '2024-06-30', 1900.00, 2850.00, 506),
(1006, 206, 4, '2023-08-01', '2024-07-31', 2100.00, 3150.00, 507),
(1007, 207, 5, '2023-09-01', '2024-08-31', 1700.00, 2550.00, 508),
(1008, 208, 5, '2023-10-01', '2024-09-30', 1800.00, 2700.00, 509),
(1009, 209, 6, '2023-11-01', '2024-10-31', 1600.00, 2400.00, 510),
(1010, 210, 6, '2023-12-01', '2024-11-30', 2000.00, 3000.00, 511),
(1011, 211, 7, '2024-01-01', '2024-12-31', 2300.00, 3450.00, 512),
(1012, 212, 7, '2024-02-01', '2025-01-31', 2500.00, 3750.00, 513);
/* Insert Staff Members */
INSERT INTO STAFF (Staff_ID, Staff_Name, SEmail) VALUES
(203, 'Ravi Mehta', 'rmehta@gmail.com'),
(204, 'Priya Sharma', 'psharma@gmail.com'),
(205, 'Kunal Joshi', 'kjoshi@gmail.com'),
(206, 'Nikita Rao', 'nrao@gmail.com'),
(207, 'Amit Kapoor', 'akapoor@gmail.com'),
(208, 'Sneha Iyer', 'siyer@gmail.com'),
(209, 'Rahul Das', 'rdas@gmail.com'),
(210, 'Pooja Sen', 'psen@gmail.com'),
(211, 'Manoj Varma', 'mvarma@gmail.com'),
(212, 'Tanya Roy', 'troy@gmail.com');
/* Insert Staff Phone Numbers */
INSERT INTO STAFF_PHONE (SPhone, Staff_ID) VALUES
('555-0602', 203),
('555-0603', 204),
('555-0604', 205),
('555-0605', 206),
('555-0606', 207),
('555-0607', 208),
('555-0608', 209),
('555-0609', 210),
('555-0610', 211),
('555-0611', 212);

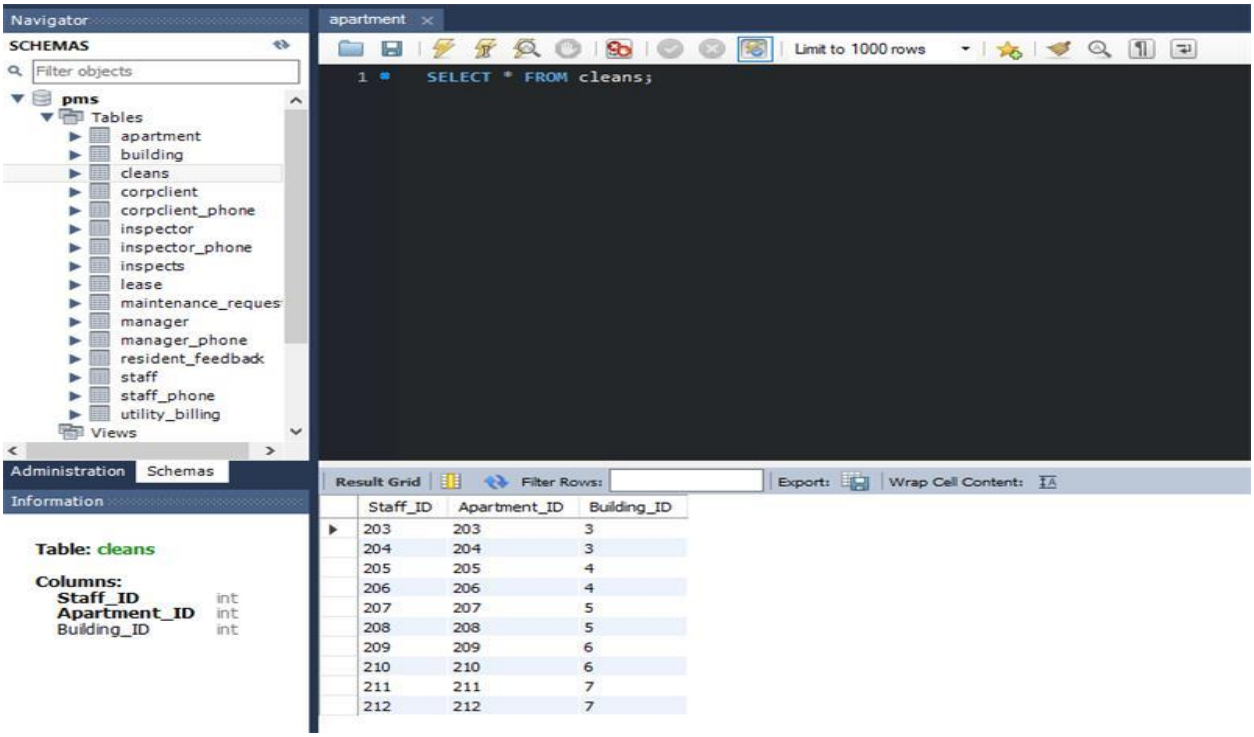

Navigator
SCHEMAS
Filter objects
pms
Tables
Views
Stored Procedures
Functions
sakila
sys
world
Administration
Schemas
Information
No object selected
InsertValues_Query_GroupProj...
Limit to 1000 rows
133
134 /* Insert Inspectors */
135 INSERT INTO INSPECTOR (Inspector_ID, InspectorName, IEmail) VALUES
136 (303, 'Ritika Das', 'rdas@exam.com'),
137 (304, 'Kabir Malhotra', 'kmalh@exam.com'),
138 (305, 'Leena Bhatt', 'lbhatt@exam.com'),
139 (306, 'Aditya Singh', 'asingh@exam.com'),
140 (307, 'Tanvi Saxena', 'tsax@exam.com'),
141 (308, 'Mohit Jain', 'mjain@gmail.com'),
142 (309, 'Nidhi Kapoor', 'nkapoor@gmail.com'),
143 (310, 'Ajay Deshmukh', 'adesh@gmail.com'),
144 (311, 'Simran Kohli', 'skohli@gmail.com'),
145 (312, 'Dev Arora', 'darora@gmail.com');
146
147 /* Insert Inspector Phone Numbers */
148 INSERT INTO INSPECTOR_PHONE (IPhone, Inspector_ID) VALUES
149 ('555-0702', 303),
150 ('555-0703', 304),
151 ('555-0704', 305),
152 ('555-0705', 306),
153 ('555-0706', 307),
154 ('555-0707', 308),
155 ('555-0708', 309),
156 ('555-0709', 310),
157 ('555-0710', 311),
158 ('555-0711', 312);
159
160 /* Insert Inspections */
161 INSERT INTO INSPECTS (Inspector_ID, Building_ID, Insp_date, next_insp) VALUES
162 (303, 3, '2023-05-15', '2024-05-15'),
163 (304, 4, '2023-06-01', '2024-06-01'),
164 (305, 5, '2023-06-20', '2024-06-20'),
165 (306, 6, '2023-07-10', '2024-07-10'),
166 (307, 7, '2023-08-01', '2024-08-01'),
167 (308, 8, '2023-09-01', '2024-09-01'),
168 (309, 9, '2023-10-01', '2024-10-01'),
169 (310, 10, '2023-11-01', '2024-11-01'),
170 (311, 11, '2023-12-01', '2024-12-01'),
171 (312, 12, '2024-01-01', '2025-01-01');
172
173 /* Insert Cleaning Assignments */
174 INSERT INTO CLEANS (Staff_ID, Apartment_ID, Building_ID) VALUES
175 (203, 203, 3),
176 (204, 204, 3),
177 (205, 205, 4),
178 (206, 206, 4),
179 (207, 207, 5),
180 (208, 208, 5),
181 (209, 209, 6),
182 (210, 210, 6),
183 (211, 211, 7),
184 (212, 212, 7);
185

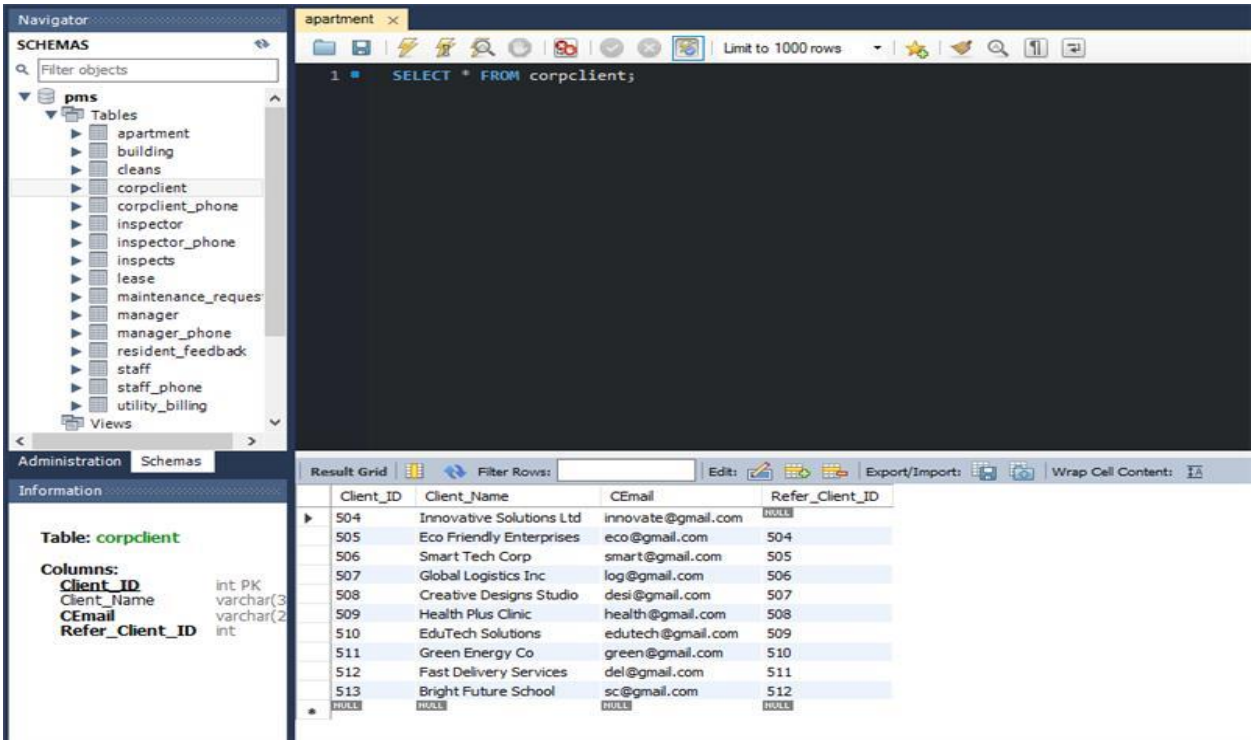

Navigator
SCHEMAS
Filter objects
pms
Tables
Views
Stored Procedures
Functions
sakila
sys
world
Administration
Schemas
Information
No object selected
InsertValues_Query_GroupProj...
Limit to 1000 rows
/* Insert Maintenance Requests */
INSERT INTO MAINTENANCE_REQUEST (Request_ID, Building_ID, Apartment_ID, Request_Date, Completion_date, Status, Assigned_Staff) VALUES
(403, 3, 203, '2023-05-20', '2023-05-22', 'Completed', 203),
(404, 3, 204, '2023-06-05', NULL, 'Pending', 204),
(405, 4, 205, '2023-06-15', '2023-06-17', 'Completed', 205),
(406, 4, 206, '2023-07-10', NULL, 'Pending', 206),
(407, 5, 207, '2023-08-01', NULL, 'Pending', 207),
(408, 5, 208, '2023-08-15', '2023-08-16', 'Completed', 208),
(409, 6, 209, '2023-09-01', NULL, 'Pending', 209),
(410, 6, 210, '2023-10-01', '2023-10-02', 'Completed', 210),
(411, 7, 211, '2023-11-01', NULL, 'Pending', 211),
(412, 7, 212, '2023-12-01', NULL, 'Pending', 212);
/* Insert Resident Feedback */
INSERT INTO RESIDENT_FEEDBACK (Building_ID, Apartment_ID, FeedbackDate, Category, Description, Status) VALUES
(3, 203, '2023-06-10', 'Facility', 'Gym equipment needs maintenance.', 'Open'),
(3, 204, '2023-06-12', 'Noise', 'Parties late at night.', 'Open'),
(4, 205, '2023-07-01', 'Maintenance', 'Air conditioning not working.', 'Resolved'),
(4, 206, '2023-07-10', 'Noise', 'Construction noise in the morning.', 'Open'),
(5, 207, '2023-08-01', 'Facility', 'Pool water is not clean.', 'Resolved'),
(5, 208, '2023-08-05', 'Noise', 'Elevator noise.', 'Open'),
(6, 209, '2023-09-01', 'Maintenance', 'Plumbing issue in bathroom.', 'Resolved'),
(6, 210, '2023-09-10', 'Facility', 'Lights in hallway flickering.', 'Resolved'),
(7, 211, '2023-10-01', 'Noise', 'Neighbors playing loud music.', 'Open'),
(7, 212, '2023-10-05', 'Maintenance', 'Broken window in living room.', 'Resolved');
/* Insert Utility Bills */
INSERT INTO UTILITY_BILLING (Building_ID, Apartment_ID, UtilityType, MonthlyCost, DueDate) VALUES
(3, 203, 'Electricity', 160.00, '2023-06-01'),
(3, 204, 'Water', 80.00, '2023-06-01'),
(4, 205, 'Gas', 100.00, '2023-07-01'),
(4, 206, 'Electricity', 170.00, '2023-07-01'),
(5, 207, 'Water', 85.00, '2023-08-01'),
(5, 208, 'Gas', 95.00, '2023-08-01'),
(6, 209, 'Electricity', 150.00, '2023-09-01'),
(6, 210, 'Water', 90.00, '2023-09-01'),
(7, 211, 'Gas', 110.00, '2023-10-01'),
(7, 212, 'Electricity', 175.00, '2023-10-01');
/* Re-enable foreign key checks */
SET FOREIGN_KEY_CHECKS = 1;

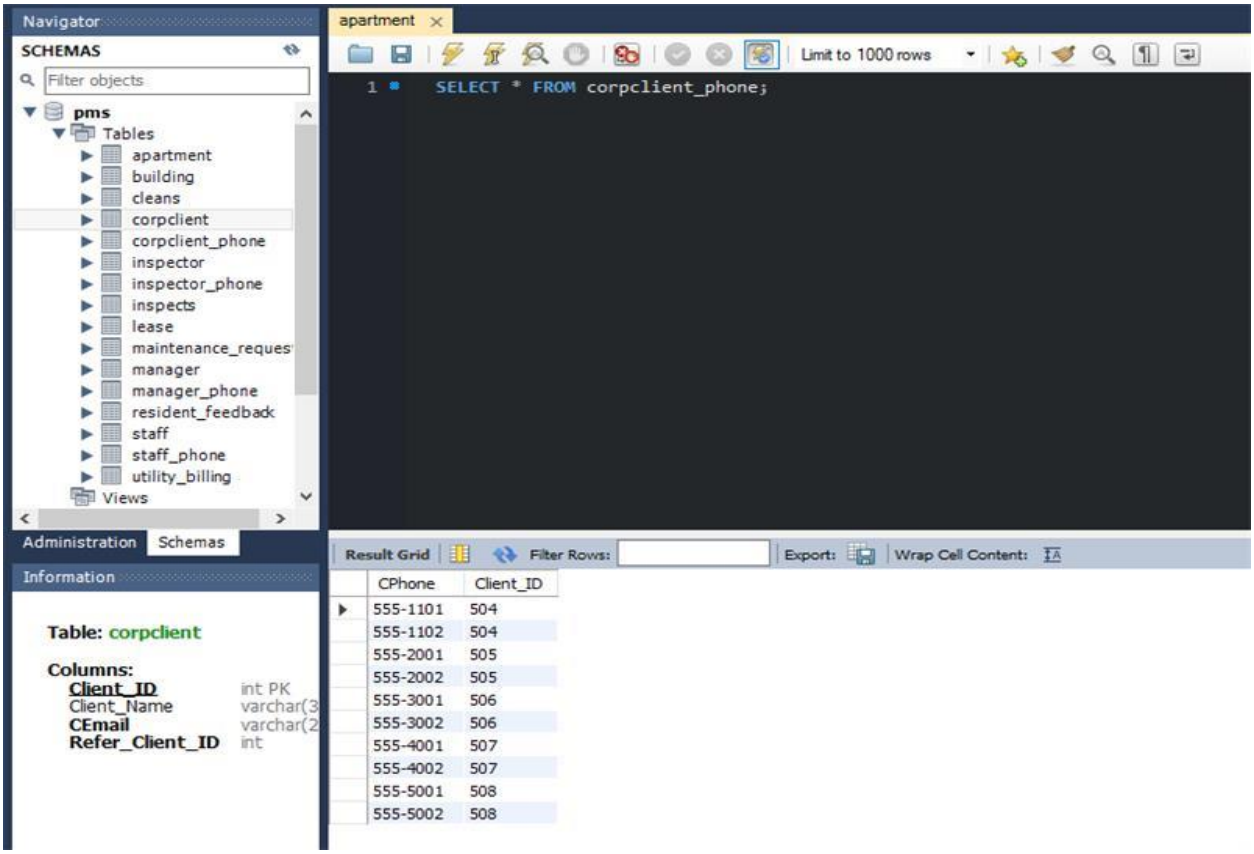
2.3.2 The SELECT * FROM TABLE statement in SQL would return all columns and all rows from the specified table.

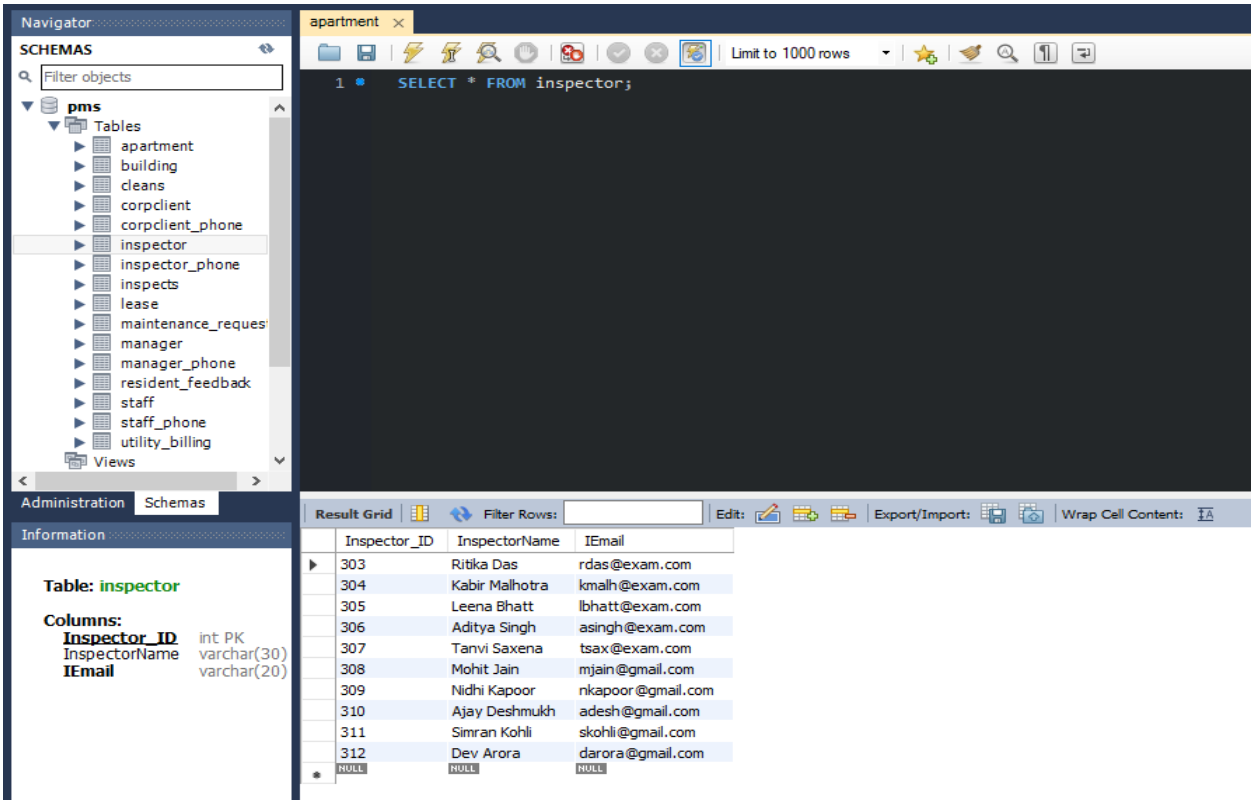
This following snippets show the details of all the tables from the query:

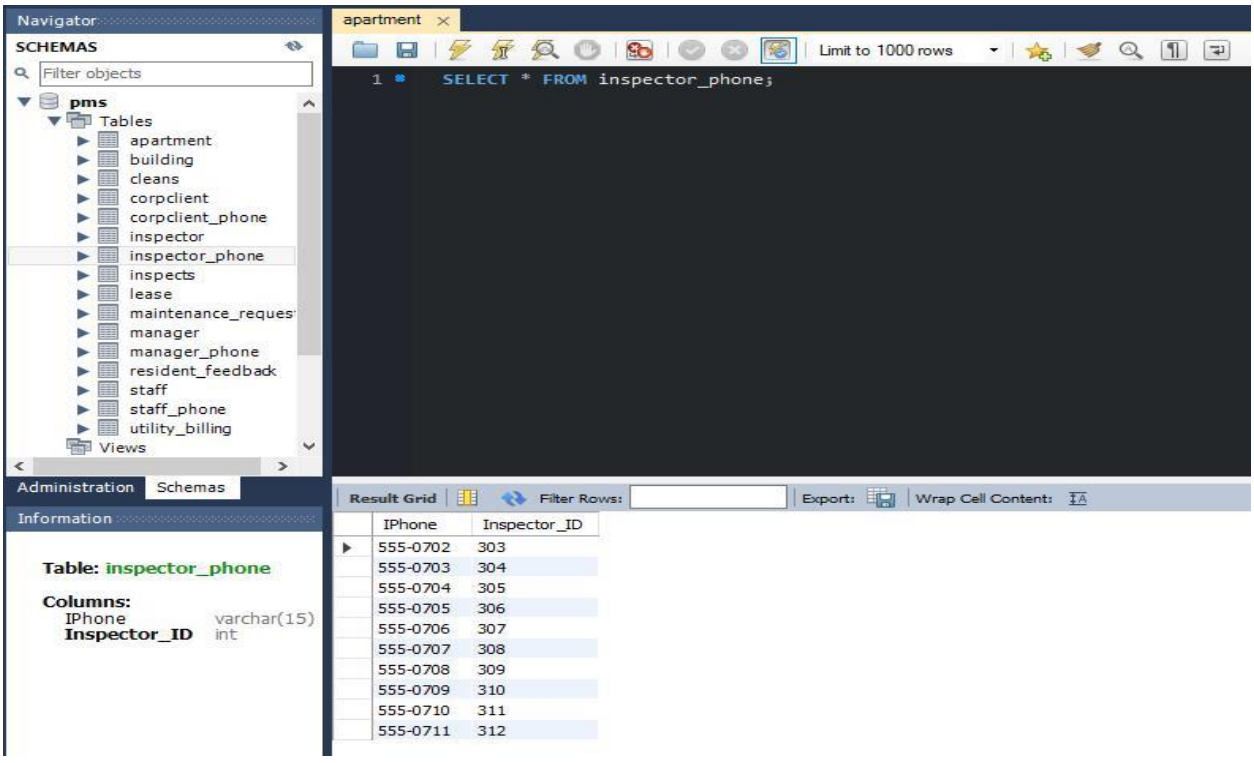


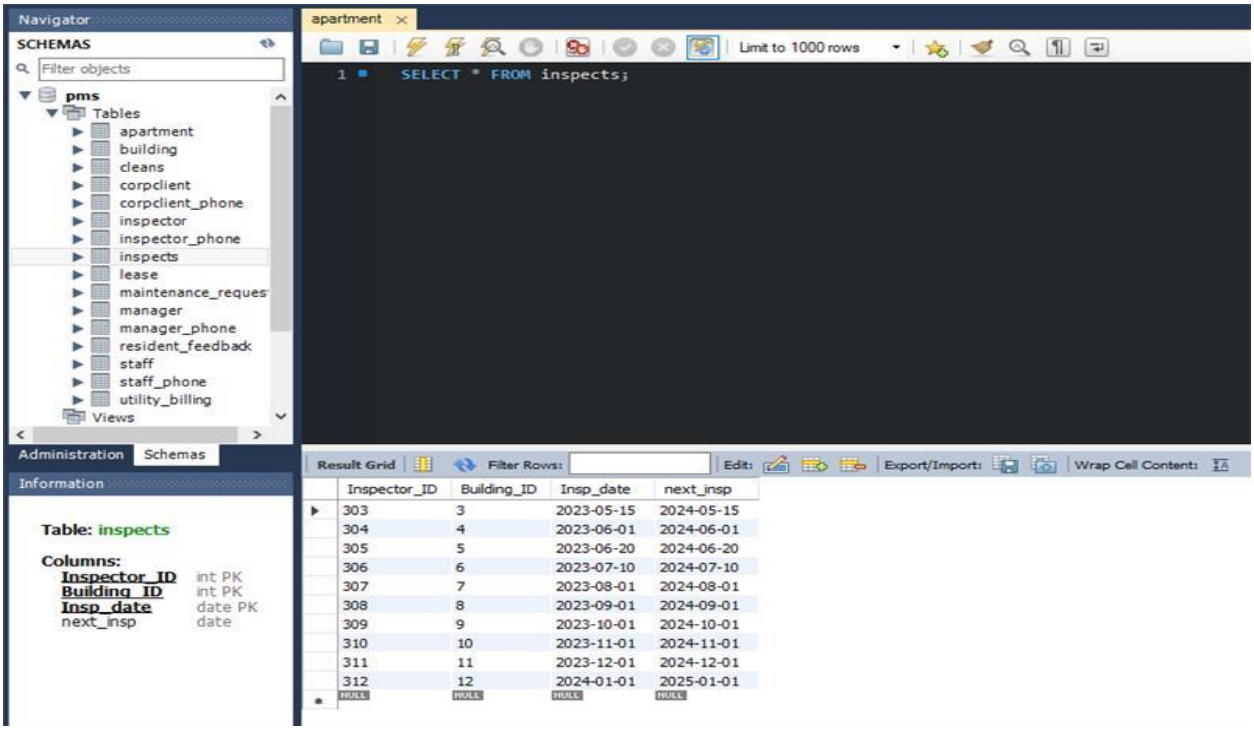


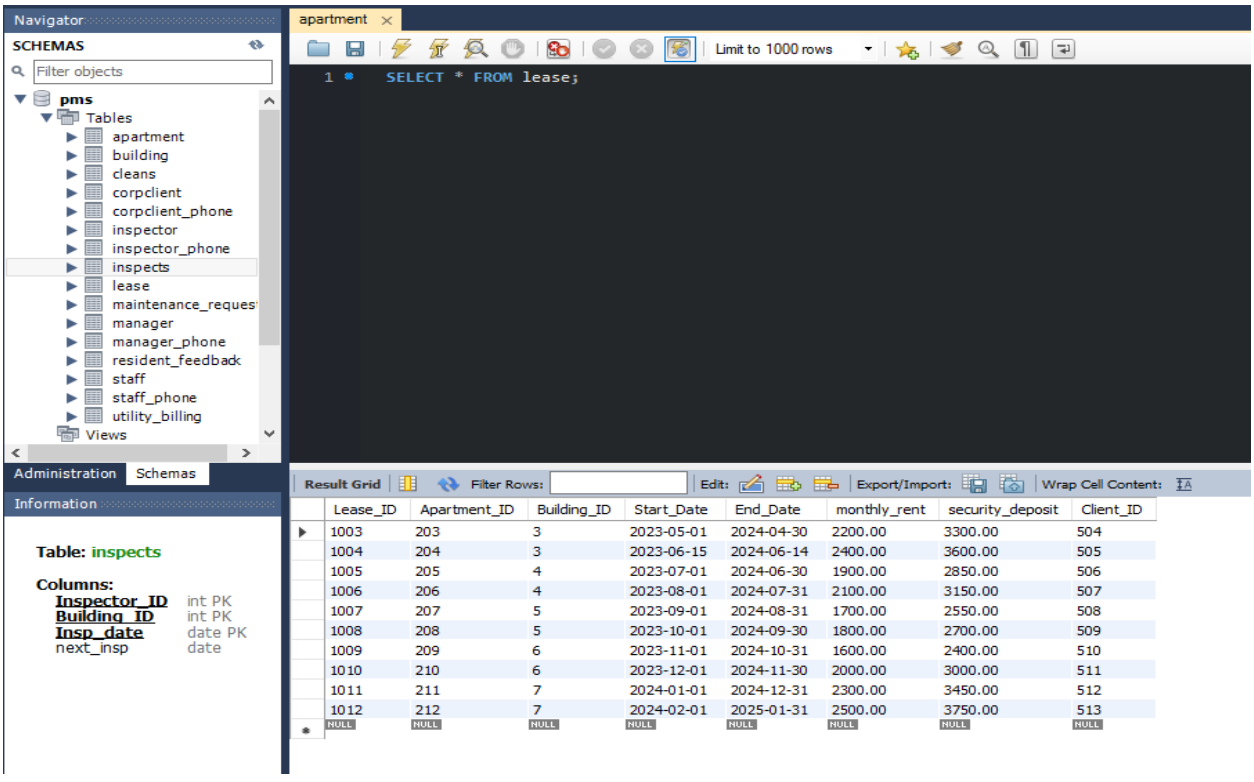

Navigator
SCHEMAS
Filter objects
pms
Tables
apartment
building
cleans
corpclient
corpclient_phone
inspector
inspector_phone
inspects
lease
maintenance_reques
manager
manager_phone
resident_feedback
staff
staff_phone
utility_billing
Views
Administration
Schemas
Information
Table: corpclient
Columns:
Client_ID int PK
Client_Name varchar(3
CEmail varchar(2
Refer_Client_ID int
apartment
Limit to 1000 rows
1 SELECT * FROM corpclient;
Result Grid
Filter Rows:
Edit:
Export/Import:
Wrap Cell Content:
Client_ID | Client_Name | CEmail | Refer_Client_ID
504 | Innovative Solutions Ltd | innovate@gmail.com | NULL
505 | Eco Friendly Enterprises | eco@gmail.com | 504
506 | Smart Tech Corp | smart@gmail.com | 505
507 | Global Logistics Inc | log@gmail.com | 506
508 | Creative Designs Studio | desi@gmail.com | 507
509 | Health Plus Clinic | health@gmail.com | 508
510 | EduTech Solutions | edutech@gmail.com | 509
511 | Green Energy Co | green@gmail.com | 510
512 | Fast Delivery Services | del@gmail.com | 511
513 | Bright Future School | sc@gmail.com | 512

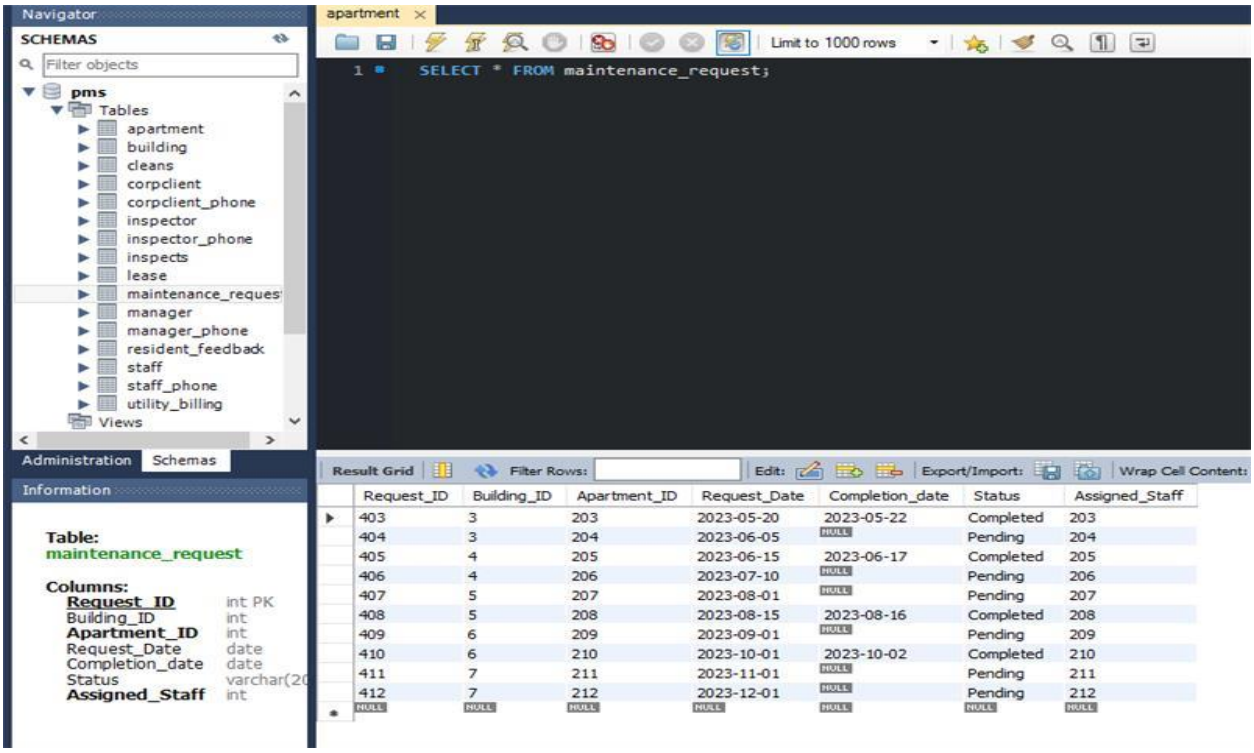

Navigator
SCHEMAS
Filter objects
pms
Tables
apartment
building
cleans
corpclient
corpclient_phone
inspector
inspector_phone
inspects
lease
maintenance_reques
manager
manager_phone
resident_feedback
staff
staff_phone
utility_billing
Views
Administration
Schemas
Information
Table: corpclient
Columns:
Client_ID int PK
Client_Name varchar(3
CEmail varchar(2
Refer_Client_ID int
apartment
Limit to 1000 rows
1 SELECT * FROM corpclient_phone;
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
CPhone | Client_ID
555-1101 | 504
555-1102 | 504
555-2001 | 505
555-2002 | 505
555-3001 | 506
555-3002 | 506
555-4001 | 507
555-4002 | 507
555-5001 | 508
555-5002 | 508

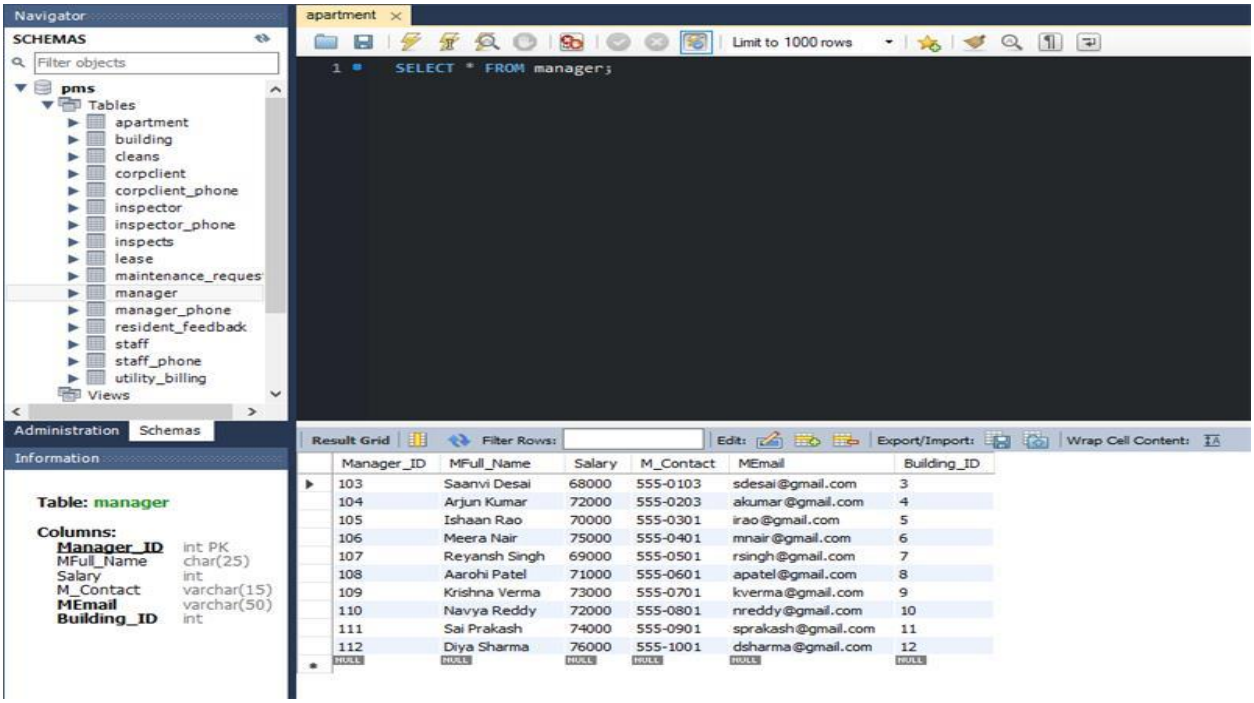

Navigator
SCHEMAS
Filter objects
pms
Tables
apartment
building
cleans
corpclient
corpclient_phone
inspector
inspector_phone
inspects
lease
maintenance_reques
manager
manager_phone
resident_feedback
staff
staff_phone
utility_billing
Views
Administration
Schemas
Information
Table: inspector
Columns:
Inspector_ID int PK
InspectorName varchar(30)
IEmail varchar(20)
apartment
Limit to 1000 rows
1 SELECT * FROM inspector;
Result Grid
Filter Rows:
Edit:
Export/Import:
Wrap Cell Content:
Inspector_ID InspectorName IEmail
303 Ritika Das rdas@exam.com
304 Kabir Malhotra kmalh@exam.com
305 Leena Bhatt lbhatt@exam.com
306 Aditya Singh asingh@exam.com
307 Tanvi Saxena tsax@exam.com
308 Mohit Jain mjain@gmail.com
309 Nidhi Kapoor nkapoor@gmail.com
310 Ajay Deshmukh adesh@gmail.com
311 Simran Kohli skohli@gmail.com
312 Dev Arora darora@gmail.com

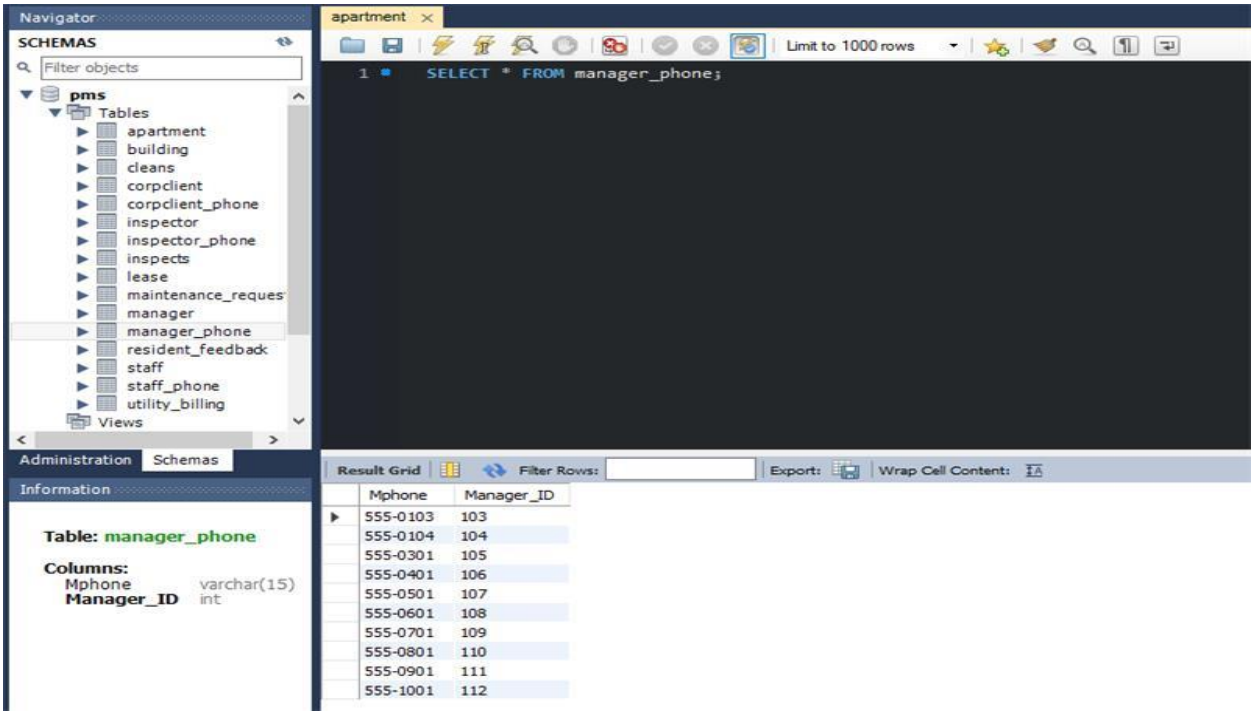

Navigator
SCHEMAS
Filter objects
pms
Tables
apartment
building
cleans
corpclient
corpclient_phone
inspector
inspector_phone
inspects
lease
maintenance_reques
manager
manager_phone
resident_feedback
staff
staff_phone
utility_billing
Views
Administration
Schemas
Information
Table: inspector_phone
Columns:
IPhone varchar(15)
Inspector_ID int
apartment
Limit to 1000 rows
1 SELECT * FROM inspector_phone;
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
IPhone Inspector_ID
555-0702 303
555-0703 304
555-0704 305
555-0705 306
555-0706 307
555-0707 308
555-0708 309
555-0709 310
555-0710 311
555-0711 312

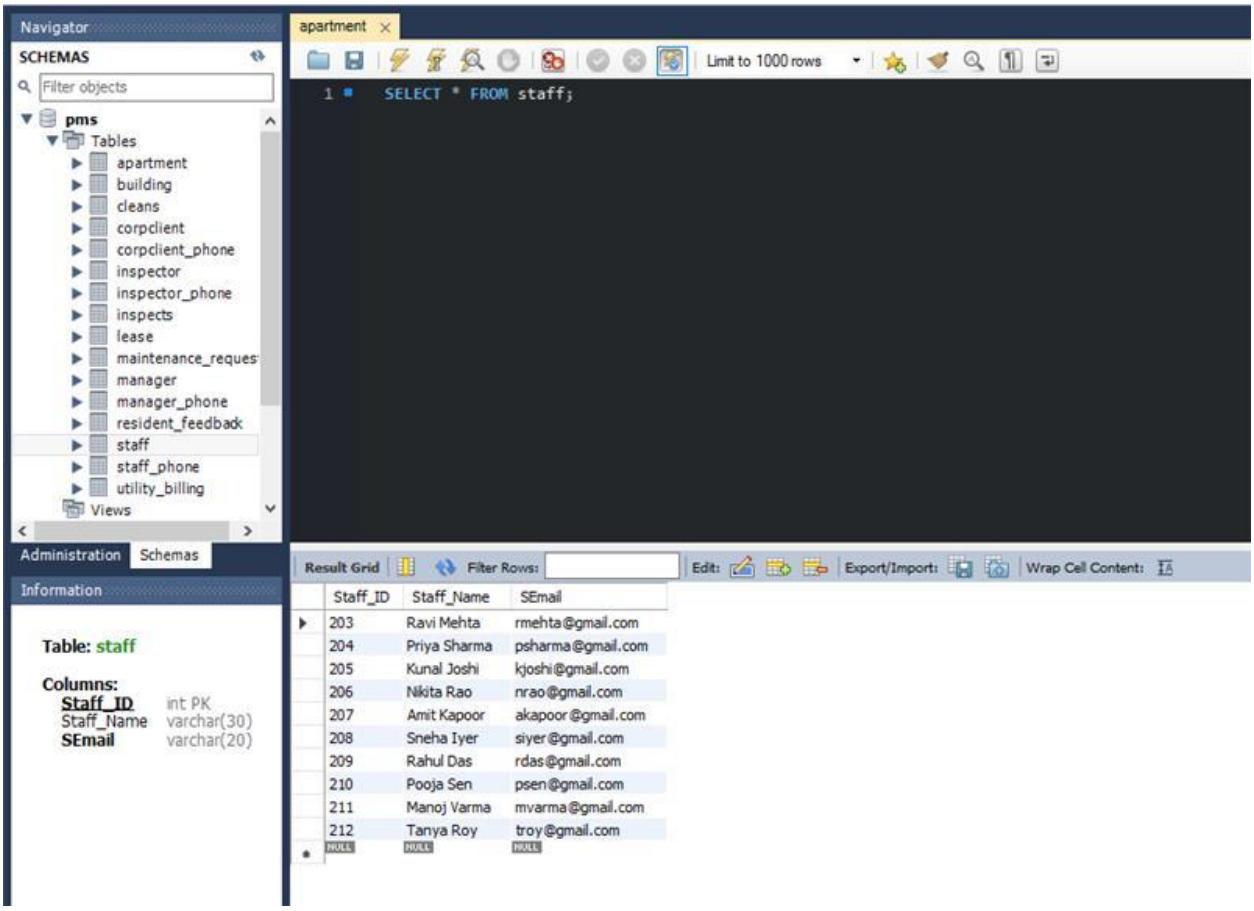

Navigator
SCHEMAS
Filter objects
pms
Tables
apartment
building
cleans
corpclient
corpclient_phone
inspector
inspector_phone
inspects
lease
maintenance_reques
manager
manager_phone
resident_feedback
staff
staff_phone
utility_billing
Views
Administration
Schemas
Information
Table: inspects
Columns:
Inspector_ID int PK
Building_ID int PK
Insp_date date PK
next_insp date
apartment
Limit to 1000 rows
1 SELECT * FROM inspects;
Result Grid
Filter Rows:
Edit:
Export/Import:
Wrap Cell Content:
Inspector_ID Building_ID Insp_date next_insp
303 3 2023-05-15 2024-05-15
304 4 2023-06-01 2024-06-01
305 5 2023-06-20 2024-06-20
306 6 2023-07-10 2024-07-10
307 7 2023-08-01 2024-08-01
308 8 2023-09-01 2024-09-01
309 9 2023-10-01 2024-10-01
310 10 2023-11-01 2024-11-01
311 11 2023-12-01 2024-12-01
312 12 2024-01-01 2025-01-01
NULL NULL NULL NULL

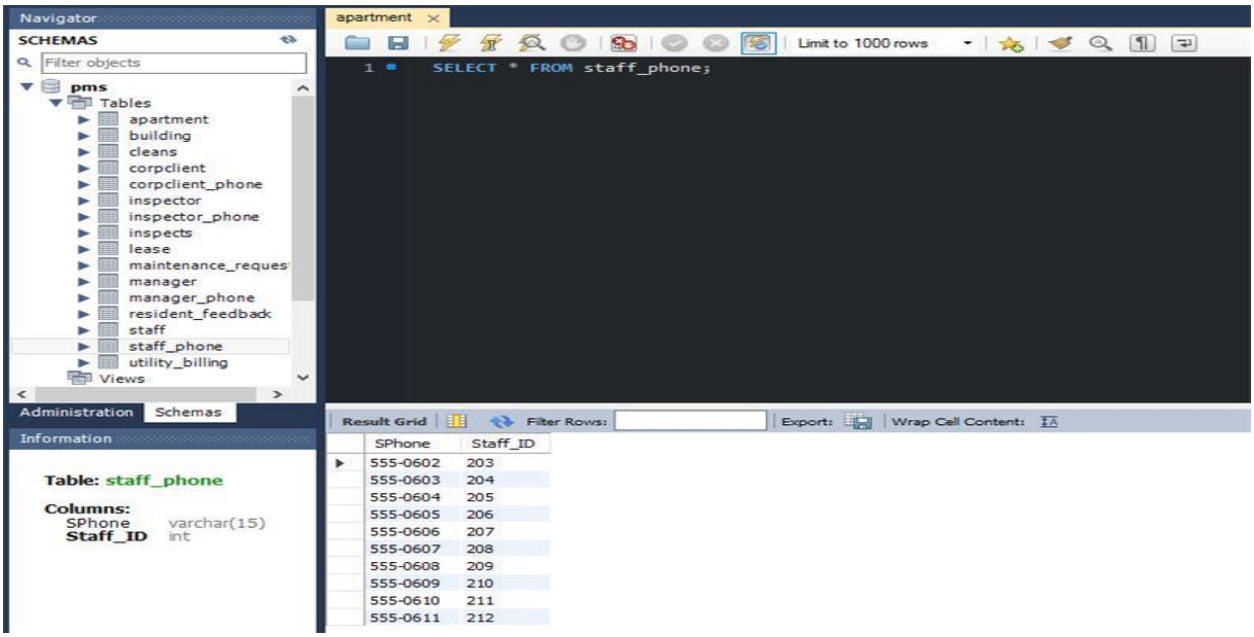

Navigator
SCHEMAS
Filter objects
pms
Tables
apartment
building
cleans
corpclient
corpclient_phone
inspector
inspector_phone
inspects
lease
maintenance_reques
manager
manager_phone
resident_feedback
staff
staff_phone
utility_billing
Views
Administration
Schemas
Information
Table: inspects
Columns:
Inspector_ID int PK
Building_ID int PK
Insp_date date PK
next_insp date
apartment
Limit to 1000 rows
1 SELECT * FROM lease;
Result Grid
Filter Rows:
Edit:
Export/Import:
Wrap Cell Content:
Lease_ID Apartment_ID Building_ID Start_Date End_Date monthly_rent security_deposit Client_ID
1003 203 3 2023-05-01 2024-04-30 2200.00 3300.00 504
1004 204 3 2023-06-15 2024-06-14 2400.00 3600.00 505
1005 205 4 2023-07-01 2024-06-30 1900.00 2850.00 506
1006 206 4 2023-08-01 2024-07-31 2100.00 3150.00 507
1007 207 5 2023-09-01 2024-08-31 1700.00 2550.00 508
1008 208 5 2023-10-01 2024-09-30 1800.00 2700.00 509
1009 209 6 2023-11-01 2024-10-31 1600.00 2400.00 510
1010 210 6 2023-12-01 2024-11-30 2000.00 3000.00 511
1011 211 7 2024-01-01 2024-12-31 2300.00 3450.00 512
1012 212 7 2024-02-01 2025-01-31 2500.00 3750.00 513
NULL NULL NULL NULL NULL NULL NULL NULL


Navigator
SCHEMAS
Filter objects
pms
Tables
apartment
building
cleans
corpclient
corpclient_phone
inspector
inspector_phone
inspects
lease
maintenance_reques
manager
manager_phone
resident_feedback
staff
staff_phone
utility_billing
Views
Administration
Schemas
Information
Table:
maintenance_request
Columns:
Request_ID int PK
Building_ID int
Apartment_ID int
Request_Date date
Completion_date date
Status varchar(20
Assigned_Staff int
apartment
Limit to 1000 rows
1 SELECT * FROM maintenance_request;
Result Grid
Filter Rows:
Edit:
Export/Import:
Wrap Cell Content:
Request_ID Building_ID Apartment_ID Request_Date Completion_date Status Assigned_Staff
403 3 203 2023-05-20 2023-05-22 Completed 203
404 3 204 2023-06-05 NULL Pending 204
405 4 205 2023-06-15 2023-06-17 Completed 205
406 4 206 2023-07-10 NULL Pending 206
407 5 207 2023-08-01 NULL Pending 207
408 5 208 2023-08-15 2023-08-16 Completed 208
409 6 209 2023-09-01 NULL Pending 209
410 6 210 2023-10-01 2023-10-02 Completed 210
411 7 211 2023-11-01 NULL Pending 211
412 7 212 2023-12-01 NULL Pending 212


Navigator
SCHEMAS
Filter objects
pms
Tables
apartment
building
cleans
corpclient
corpclient_phone
inspector
inspector_phone
inspects
lease
maintenance_reques
manager
manager_phone
resident_feedback
staff
staff_phone
utility_billing
Views
Administration
Schemas
Information
Table: manager
Columns:
Manager_ID int PK
MFull_Name char(25)
Salary int
M_Contact varchar(15)
MEmail varchar(50)
Building_ID int
apartment
Limit to 1000 rows
1 SELECT * FROM manager;
Result Grid
Filter Rows:
Edit:
Export/Import:
Wrap Cell Content:
Manager_ID MFull_Name Salary M_Contact MEmail Building_ID
103 Saanvi Desai 68000 555-0103 sdesai@gmail.com 3
104 Arjun Kumar 72000 555-0203 akumar@gmail.com 4
105 Ishaan Rao 70000 555-0301 irao@gmail.com 5
106 Meera Nair 75000 555-0401 mnair@gmail.com 6
107 Reyansh Singh 69000 555-0501 rsingh@gmail.com 7
108 Aarohi Patel 71000 555-0601 apatel@gmail.com 8
109 Krishna Verma 73000 555-0701 kverma@gmail.com 9
110 Navya Reddy 72000 555-0801 nreddy@gmail.com 10
111 Sai Prakash 74000 555-0901 sprakash@gmail.com 11
112 Diya Sharma 76000 555-1001 dsharma@gmail.com 12

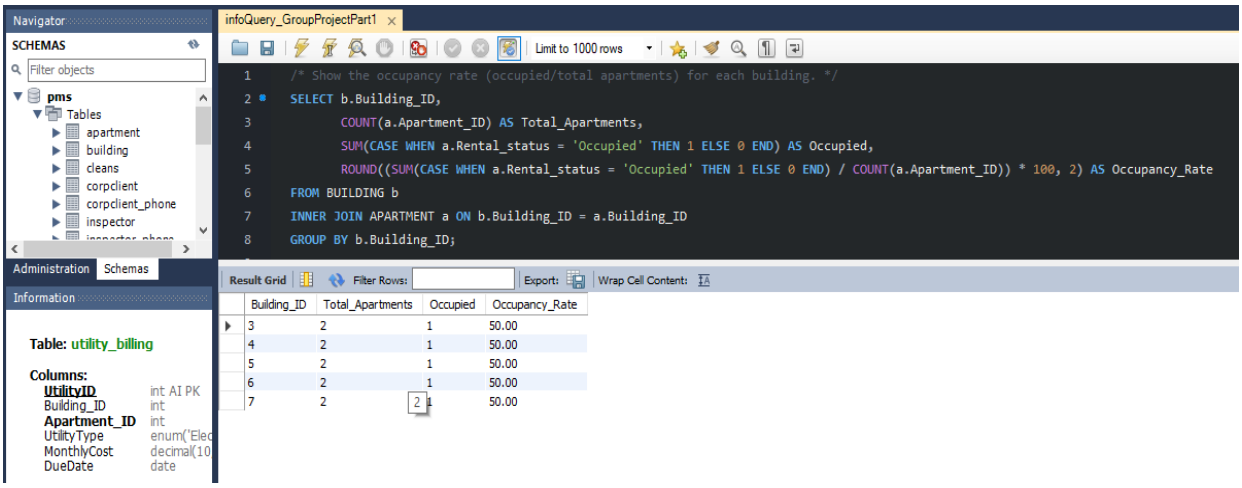

Navigator
SCHEMAS
Filter objects
pms
Tables
apartment
building
cleans
corpclient
corpclient_phone
inspector
inspector_phone
inspects
lease
maintenance_reques
manager
manager_phone
resident_feedback
staff
staff_phone
utility_billing
Views
Administration
Schemas
Information
Table: manager_phone
Columns:
Mphone varchar(15)
Manager_ID int
apartment
Limit to 1000 rows
1 • SELECT * FROM manager_phone;
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
Mphone
Manager_ID
555-0103 103
555-0104 104
555-0301 105
555-0401 106
555-0501 107
555-0601 108
555-0701 109
555-0801 110
555-0901 111
555-1001 112


Navigator
SCHEMAS
Filter objects
pms
Tables
apartment
building
cleans
corpclient
corpclient_phone
inspector
inspector_phone
inspects
lease
maintenance_reques
manager
manager_phone
resident_feedback
staff
staff_phone
utility_billing
Views
Administration
Schemas
Information
Table: staff
Columns:
Staff_ID int PK
Staff_Name varchar(30)
SEmail varchar(20)
apartment
Limit to 1000 rows
1 • SELECT * FROM staff;
Result Grid
Filter Rows:
Edit:
Export/Import:
Wrap Cell Content:
Staff_ID
Staff_Name
SEmail
203 Ravi Mehta rmehta@gmail.com
204 Priya Sharma psharma@gmail.com
205 Kunal Joshi kjoshi@gmail.com
206 Nikita Rao nrao@gmail.com
207 Amit Kapoor akapoor@gmail.com
208 Sneha Iyer siyer@gmail.com
209 Rahul Das rdas@gmail.com
210 Pooja Sen psen@gmail.com
211 Manoj Varma mvarma@gmail.com
212 Tanya Roy troy@gmail.com

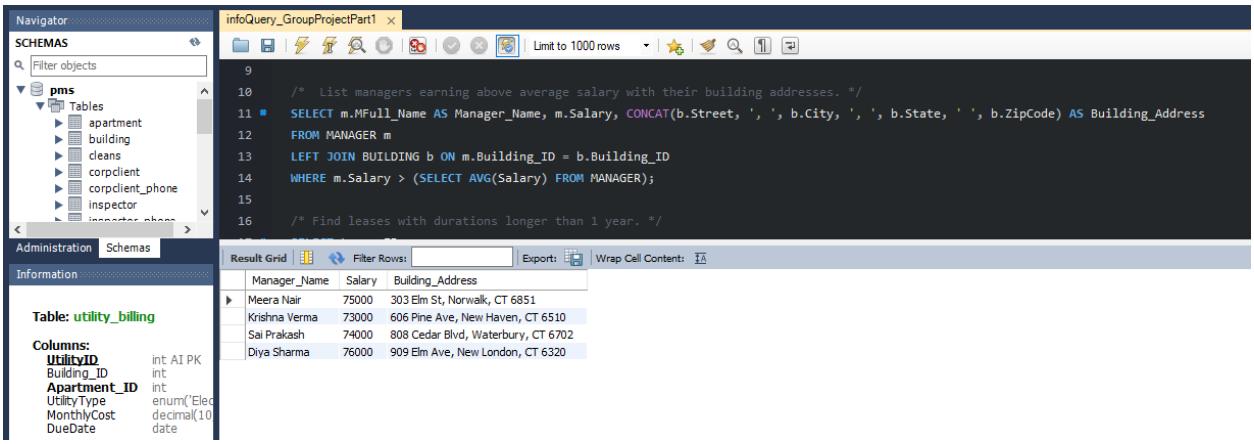


2.4 Create 10 business questions and provide the SQL codes and results. Submit the questions, SQL codes and the screenshots of the results. Use different types of joins, aggregation function, sub query, CHECK, stored procedure, trigger at least once. (30 points)

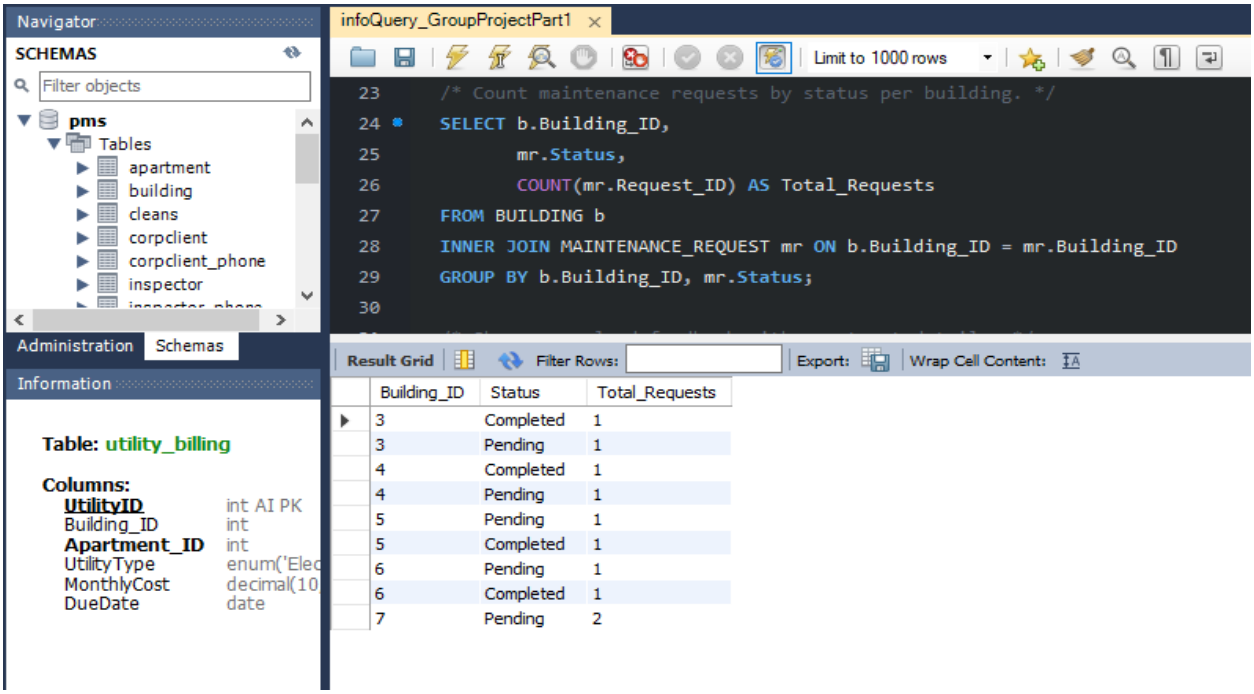
2.4.1 Show the occupancy rate (occupied/total apartments) for each building.



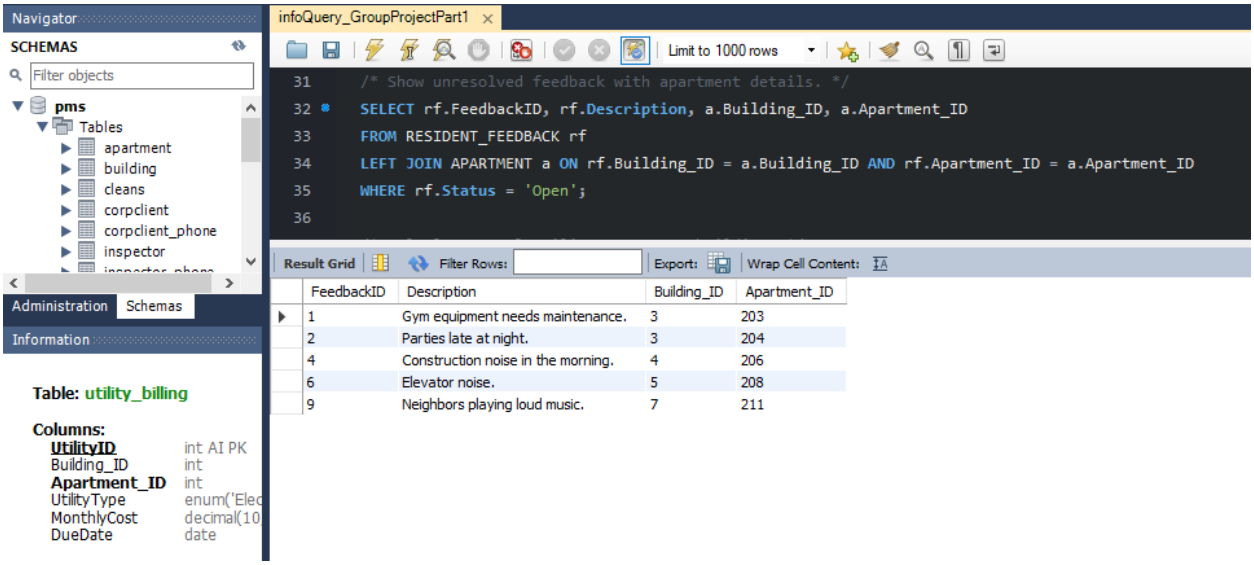
2.4.2 List managers earning above average salary with their building addresses.



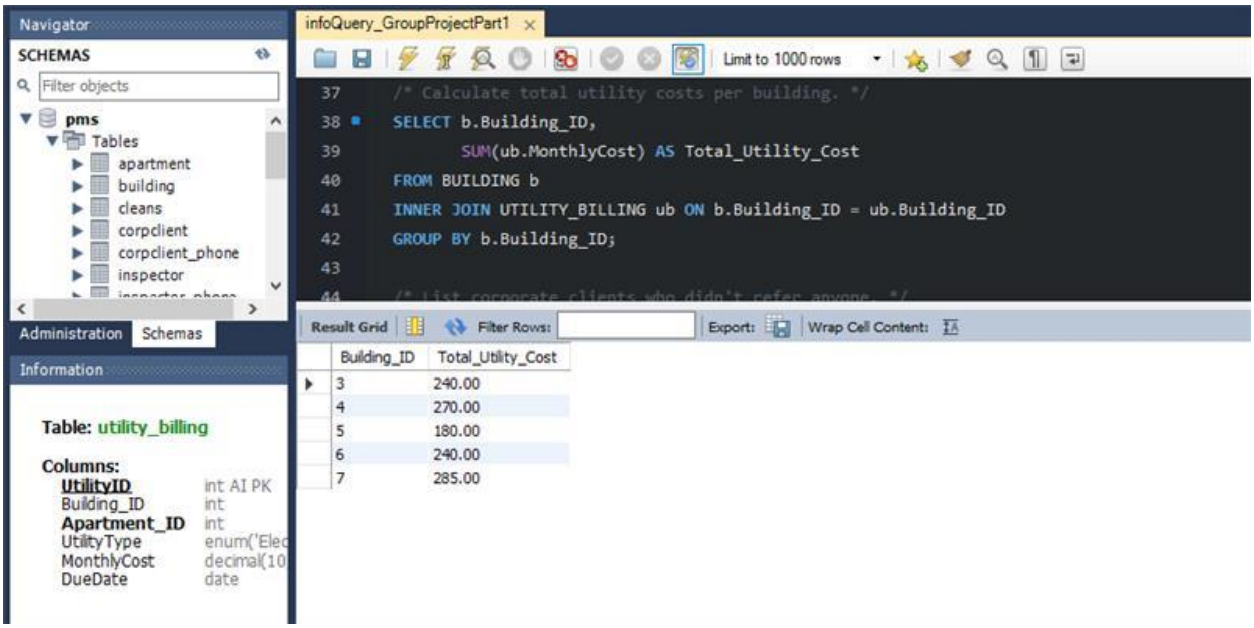
2.4.3 Count maintenance requests by status per building.



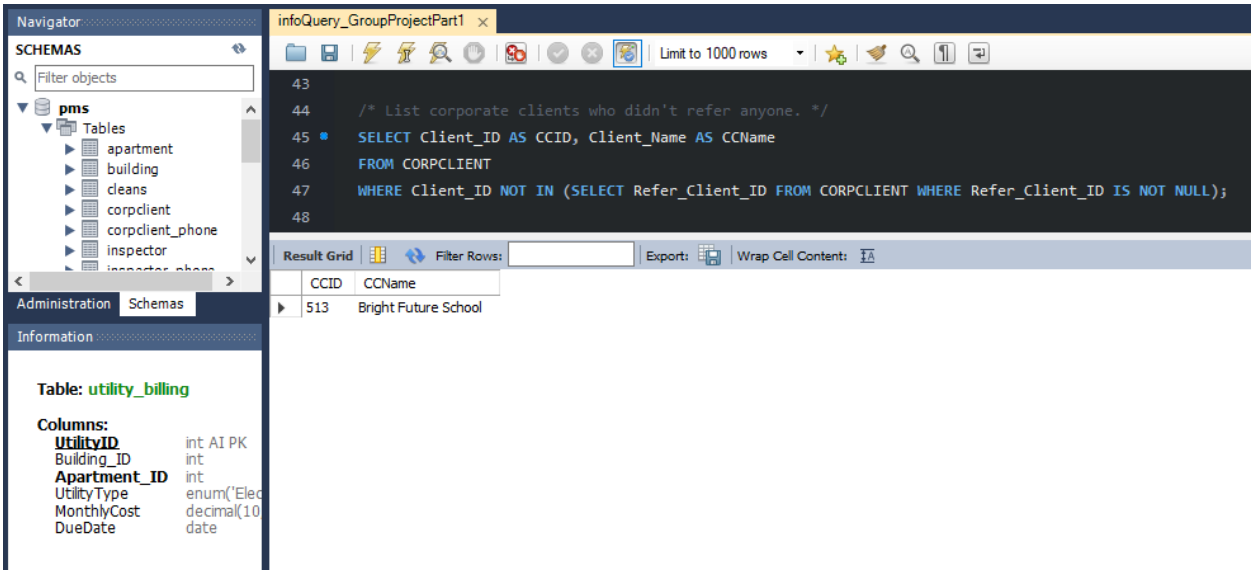
2.4.4 Show unresolved feedback with apartment details.



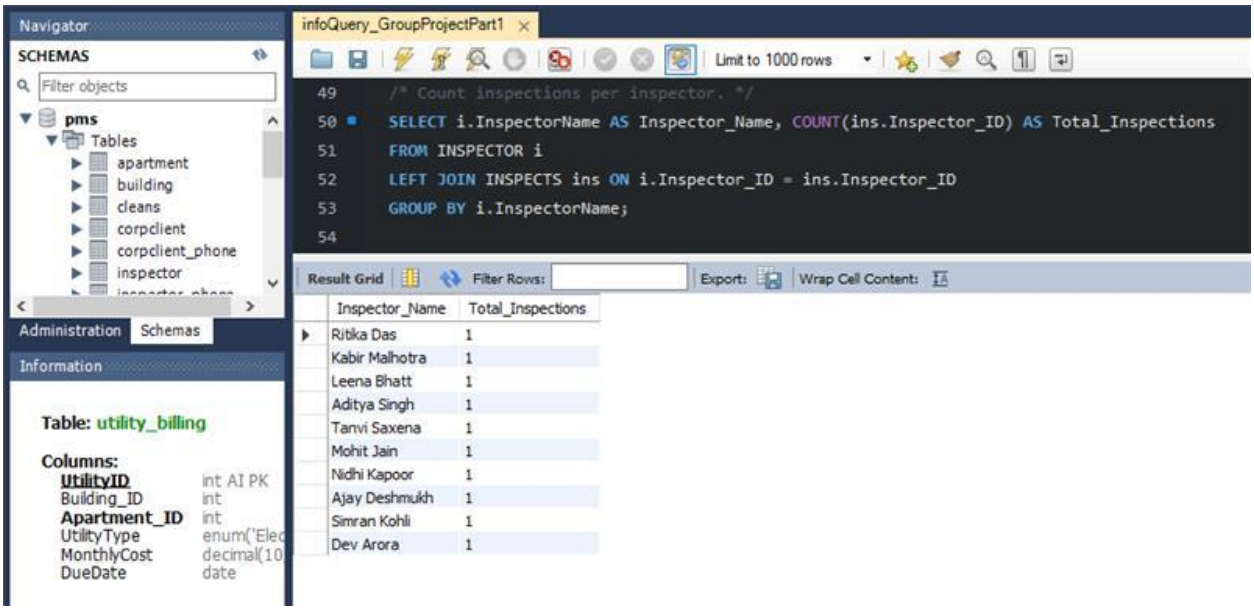
2.4.5 Calculate total utility costs per building.



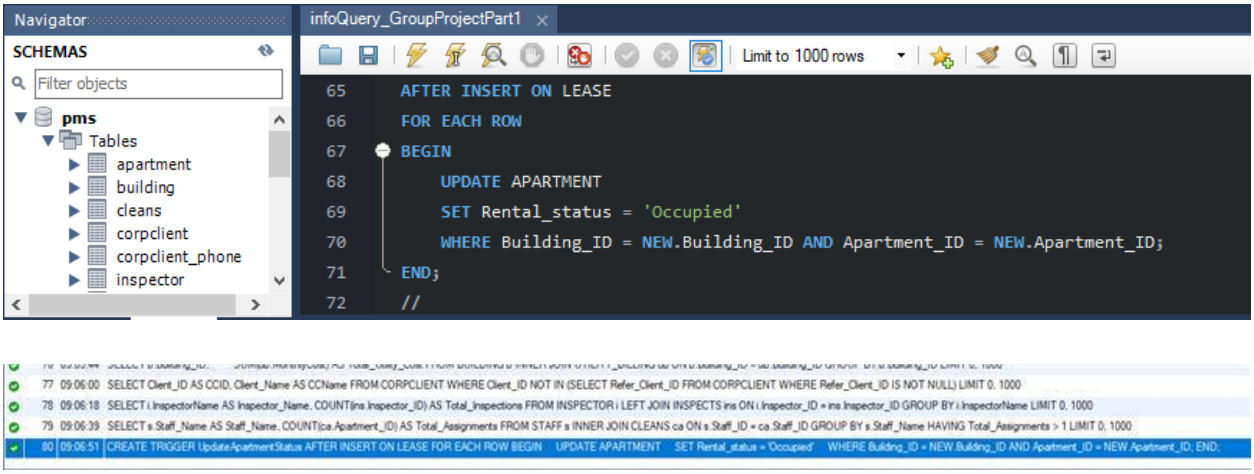
2.4.6 List corporate clients who didn't refer anyone.



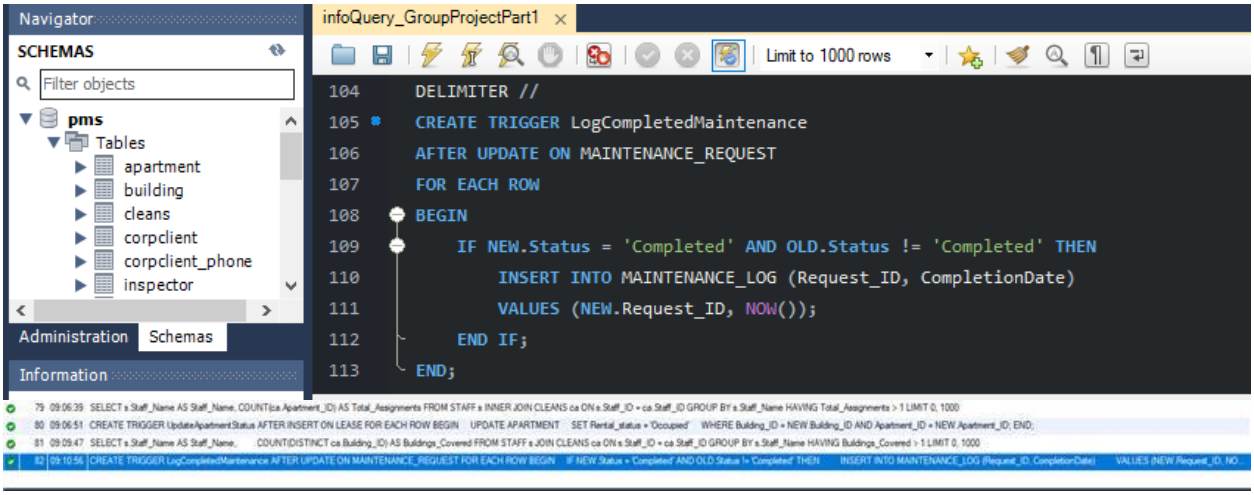
2.4.7 Count inspections per inspector.



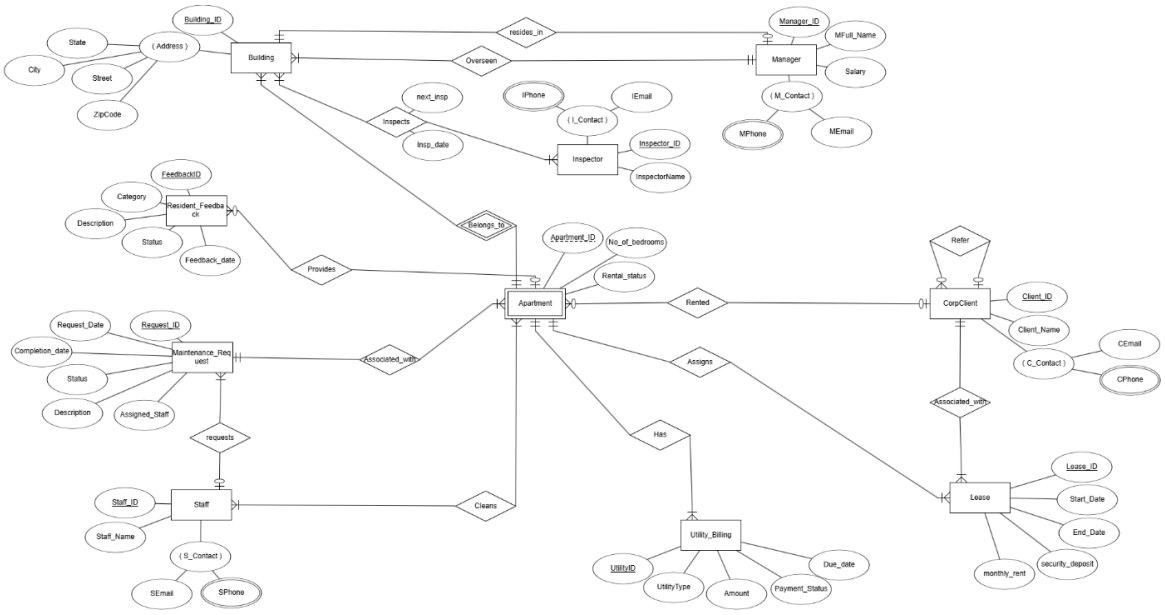
2.4.8 Create a trigger to mark apartments as occupied when leased.



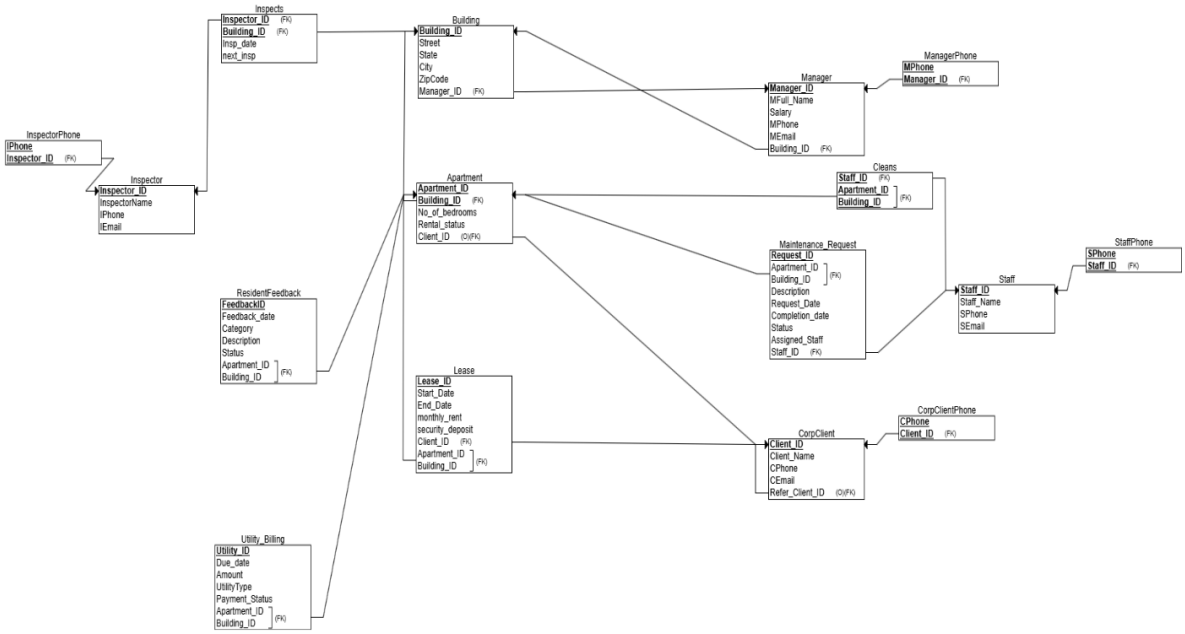
2.4.9 Automatically log completed maintenance requests



3.0 Entity-Relationship: Property Management System: 2 New Entities



3.1 Relational-Schema: PROPERTY MANAGEMENT SYSTEM: 2 NEW ENTITIES

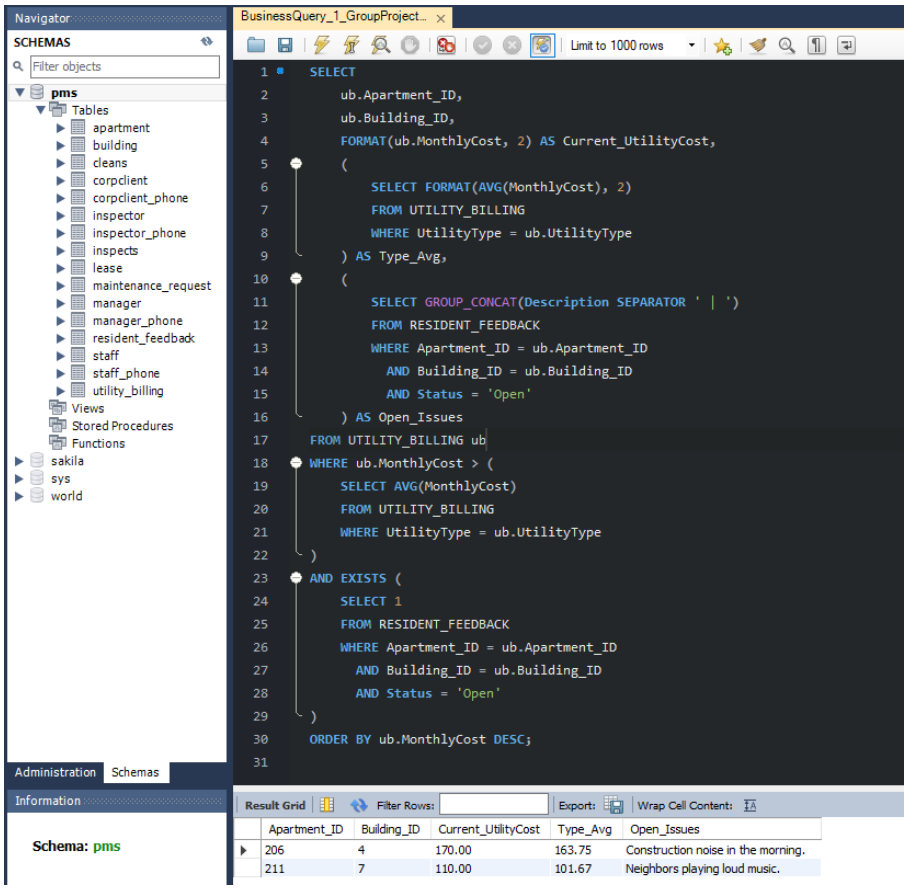


3.2 DATA-DICTIONARY: PROPERTY MANAGEMENT SYSTEM: 2 NEW ENTITIES

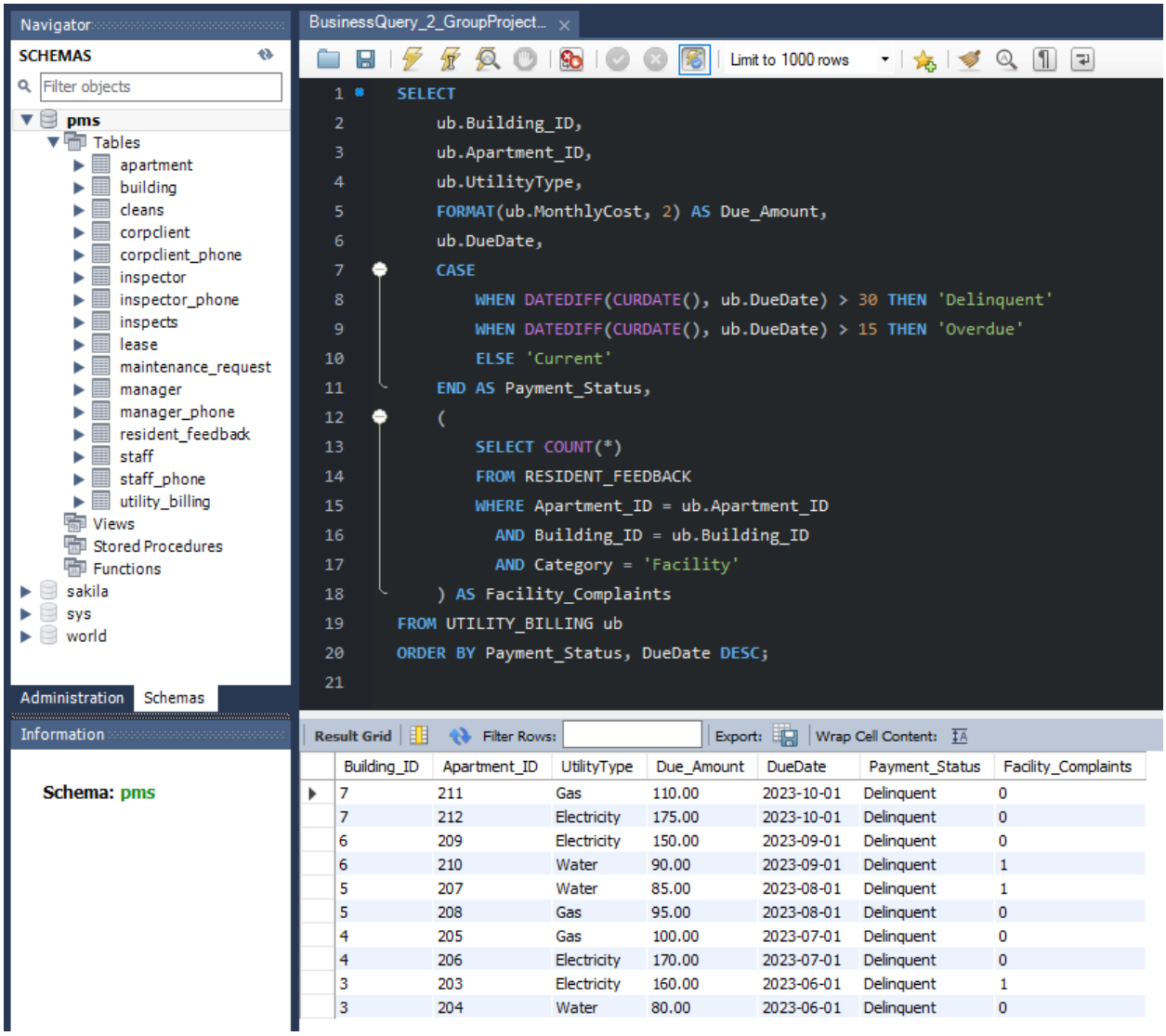
Entity:	RESIDENT FEEDBACK TABLE	Track utility billing of apartments					
Field Name	**Description**	**Type**	**Specifications**	**Default**	**Required**	**Unique**	**Key(s)**
FeedbackID	Unique feedback ID	INT	Auto-increment	-	YES	YES	PK
BuildingID	Building ID	INT	Links to Apartment	-	YES	NO	-
AptNo	Apartment number	INT	Links to Apartment	-	YES	NO	-
FeedbackDate	Date of feedback submission	DATE	Format: YYYY-MM-DD	-	YES	NO	-
Category	Feedback category	ENUM	MaINTenance, Noise, Facility	-	YES	NO	-
Description	Feedback details	TEXT		-	YES	NO	-
Status	Feedback resolution status	VARCHAR(10)	CHECK: Open/Resolved	-	YES	NO	-
Entity:	**UTILITY BILLING**	Track feedbacks from residents					
Field Name	**Description**	**Type**	**Specifications**	**Default**	**Required**	**Unique**	**Key(s)**
UtilityID	Unique utility bill ID	INT	Auto-increment	-	YES	YES	PK
BuildingID	Building ID	INT	Links to Apartment	-	YES	NO	-
AptNo	Apartment number	INT	Links to Apartment	-	YES	NO	-
UtilityType	Type of utility	ENUM	Electricity, Water, Gas	-	YES	NO	-
MonthlyCost	Monthly utility cost	DECIMAL(10,2)	Positive value (e.g., 150.00)	-	YES	NO	-
DueDate	Payment due date	DATE	Format: YYYY-MM-DD	-	YES	NO	-

3.3 BUSINESS QUERY: 2 NEW ENTITIES – RESIDENT FEEDBACK AND UTILITY BILLING

3.3.1 High Utility Cost Apartments with Pending Feedback

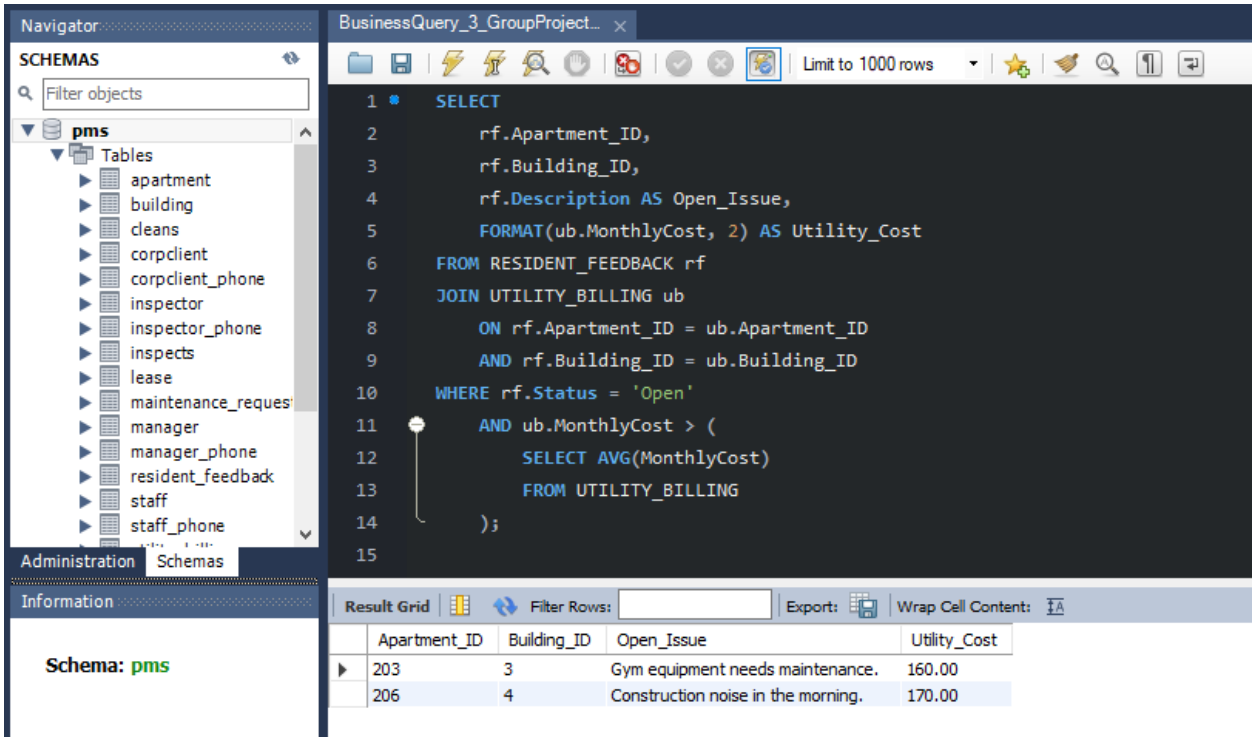


3.3.2 Utility payment compliance check

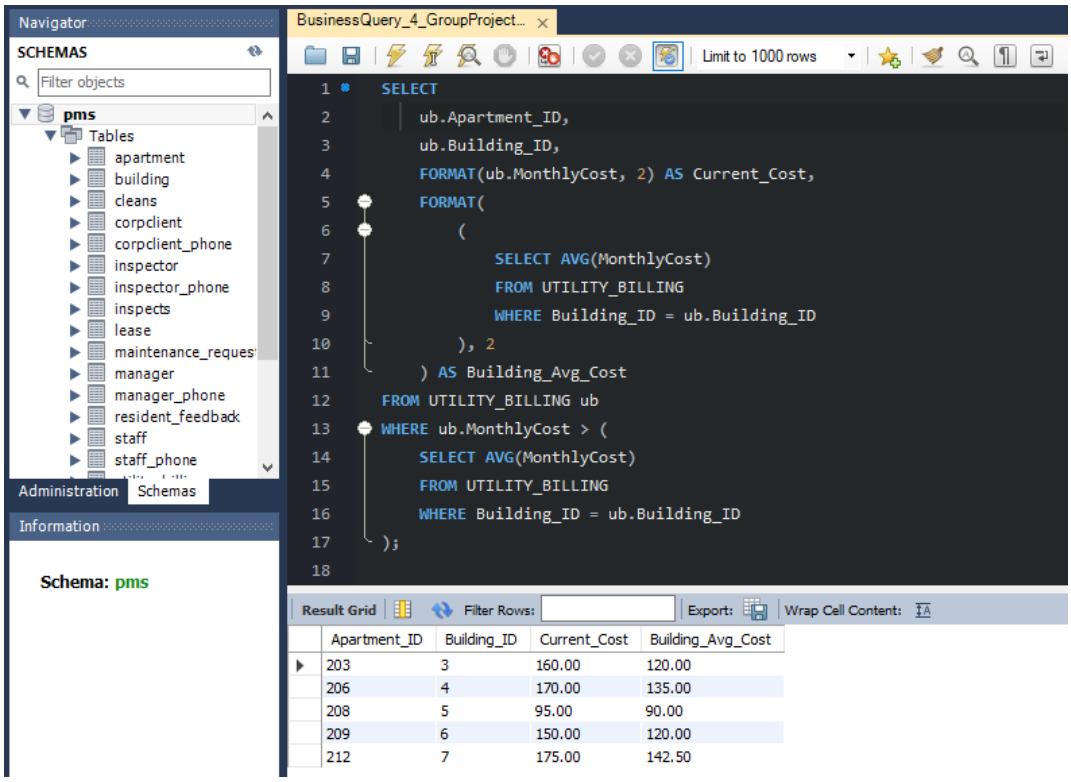


Building_ID	Apartment_ID	UtilityType	Due_Amount	DueDate	Payment_Status	Facility_Complaints
7	211	Gas	110.00	2023-10-01	Delinquent	0
7	212	Electricity	175.00	2023-10-01	Delinquent	0
6	209	Electricity	150.00	2023-09-01	Delinquent	0
6	210	Water	90.00	2023-09-01	Delinquent	1
5	207	Water	85.00	2023-08-01	Delinquent	1
5	208	Gas	95.00	2023-08-01	Delinquent	0
4	205	Gas	100.00	2023-07-01	Delinquent	0
4	206	Electricity	170.00	2023-07-01	Delinquent	0
3	203	Electricity	160.00	2023-06-01	Delinquent	1
3	204	Water	80.00	2023-06-01	Delinquent	0

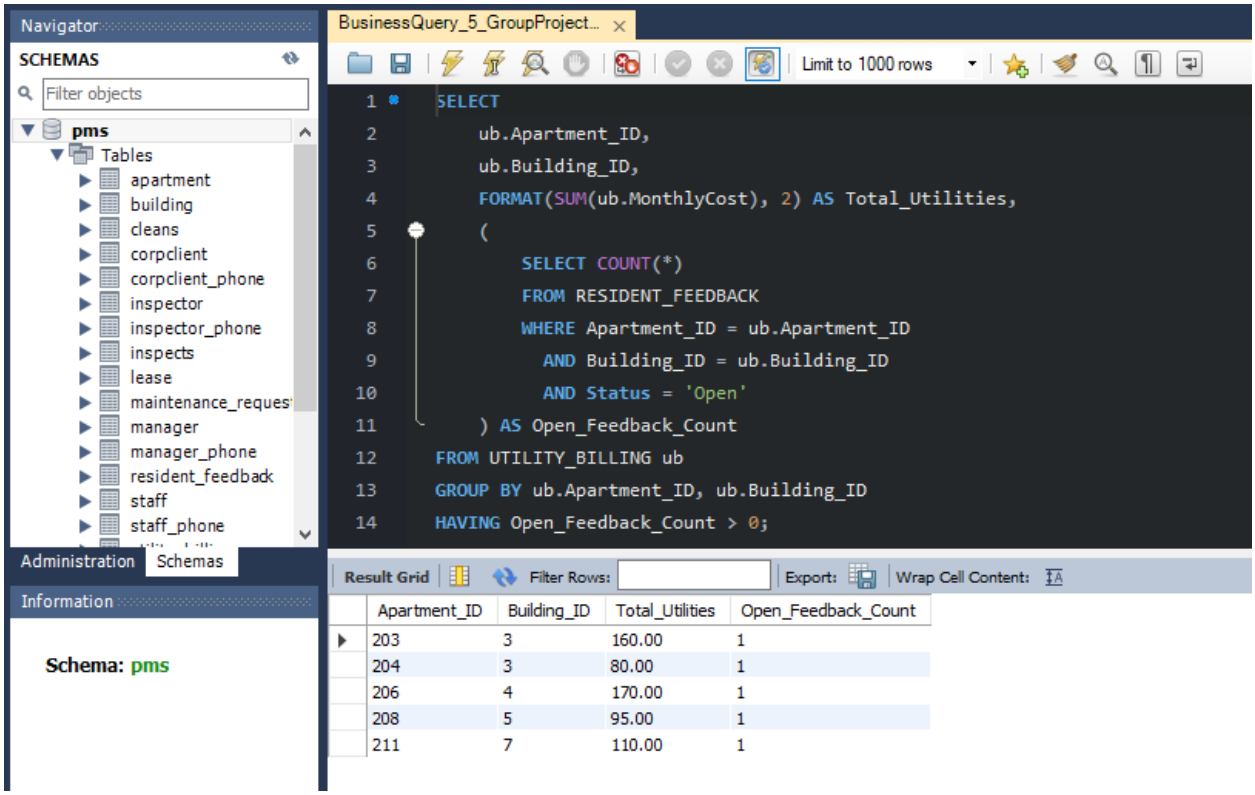
3.3.3 Apartments with Unresolved Feedback & High Utility Costs



3.3.4 Utility Costs vs Building Average



3.3.5 Open Feedback Count with Total Utility Cost



This project successfully addressed the property management company's operational and analytical needs by modernizing its database infrastructure.

Part 1: Operational Database Upgrade

1. **Enhanced Functionality:**

    - The upgraded database now efficiently tracks leases and maintenance requests, streamlining rental management and repair workflows.
    - New entities like Resident Feedback and Utility Billing were added to improve tenant satisfaction monitoring and cost tracking.

2. **Data Integrity & Efficiency:**

    - Triggers (e.g., auto-updating apartment vacancy status) and stored procedures (e.g., rent revenue calculation) reduced manual effort.
    - Referential integrity was enforced through foreign keys and constraints (e.g., CHECK clauses for valid statuses).

3. **Business Insight**

    - Queries using joins, aggregations, and sub queries answered critical questions, such as identifying unoccupied apartments or analyzing referral trends.

1. **Scalability:**

    - The relational schema and ER diagram ensure flexibility to accommodate future expansions, such as integrating IoT devices for smart buildings.

Impact - The upgraded system will:

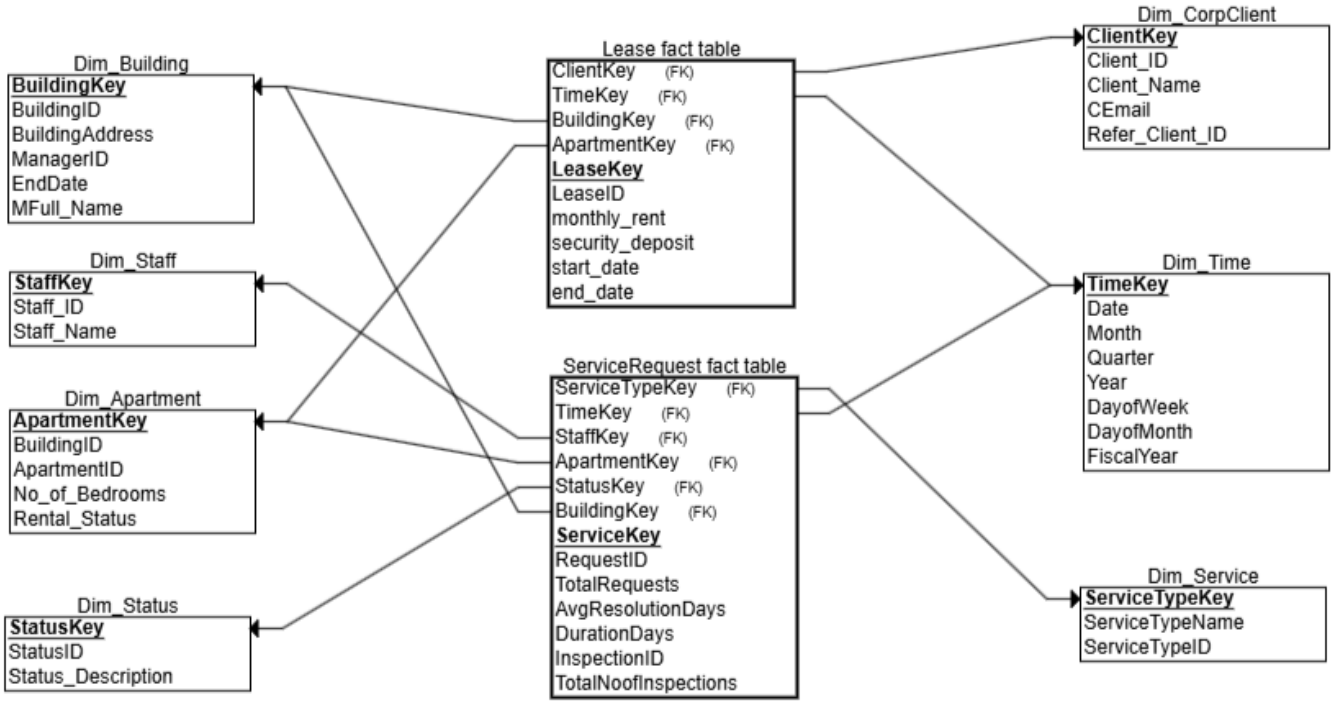
- Reduce vacancies by identifying leasing trends and high-demand apartment features.
- Lower operational costs through predictive maintenance alerts and utility usage optimization.
- Improve tenant retention by resolving feedback faster and ensuring compliance with safety inspections.
- Support strategic growth by analyzing referral patterns and corporate client preferences.

By bridging operational efficiency with analytical depth, this database redesign positions the company to thrive in a competitive real estate market while maintaining scalability for future innovations.

PART 2

4.0 Fact and Dimension Schema Design

This schema is designed using a star structure to support efficient analytical processing within a property management data warehouse. At the center are two fact tables that track key business processes—rental activity and service requests—each surrounded by multiple descriptive dimensions. The rental-related table records lease transactions across time, units, and tenants, capturing metrics such as rent, deposits, and lease durations. The service-related table tracks maintenance activities and operational performance indicators including request volume, resolution times, and inspection counts. Dimensions enrich the analysis by providing attributes for time periods, property structures, units, personnel, client relationships, and service categories. This model enables flexible slicing and historical trend analysis across various perspectives such as time, location, client, or operational category, supporting informed decision-making in real estate operations.



4.1 Executive Summary

To support strategic decision-making and historical analysis, a data warehouse was designed and implemented for a property management company. This warehouse enables executives, managers, and analysts to monitor performance indicators such as rental income, occupancy trends, maintenance efficiency, and inspection patterns over time. The solution leverages

dimensional modeling and ETL processes to transform operational data into a structured, analytics-ready format that supports business intelligence and reporting needs.

5.0 Fact Tables:

The data warehouse includes two primary fact tables that capture the core transactional processes of the property management system:

5.1 Lease Transaction Fact Table

The **Lease Transaction Fact** table serves as the central hub for tracking all leasing activities within the property management system. Each record in this table represents a unique lease agreement between a corporate tenant and a property unit (apartment).

- **Structure**:
    - The primary key, LeaseKey, uniquely identifies each lease transaction.
    - Foreign keys such as ClientKey, TimeKey, BuildingKey, and ApartmentKey link this fact table to dimension tables like Corporate Client, Time, Building, and Apartment, respectively. These connections allow analysts to contextualize leases by tenant, location, and time.
    - Key attributes include LeaseID (the original lease identifier from the source system), monthly_rent (the recurring payment amount), security_deposit (the upfront deposit held for the lease), and start_date/end_date (defining the lease term).
- **Business Purpose**:
  This table enables stakeholders to analyze rental income trends, occupancy rates, and tenant retention. For example, by joining with the Time Dimension, managers can compare quarterly revenue across buildings or identify seasonal fluctuations in lease signings. It also supports compliance reporting by tracking deposit amounts and lease durations.

5.2 Service Request Management Fact Table

The Service Request Management Fact table captures all maintenance and service requests raised by tenants or property managers. It provides insights into operational efficiency and resource allocation.

- **Structure**:
    - The primary key, ServiceKey, uniquely identifies each service request.
    - Foreign keys like ServiceTypeKey, TimeKey, StaffKey, ApartmentKey, StatusKey, and BuildingKey connect to dimensions such as Service Type, Time, Staff, Apartment, Status, and Building. These relationships allow for granular analysis of request types, resolution times, and staff performance.
    - Metrics such as TotalRequests (aggregating requests by category), AvgResolutionDays (average time to resolve issues), and DurationDays (time taken for a specific request) quantify operational efficiency.
- **Business Purpose**:
  This table helps identify recurring maintenance issues (e.g., plumbing failures in older buildings) and evaluate staff productivity. For instance, by linking to the Staff Dimension, managers can determine which employees resolve requests fastest or handle the most complex tasks. It also supports root-cause analysis by correlating service types (e.g., electrical repairs) with apartment attributes (e.g., building age).

6.0 Database Schema Recap: Property Management System

This schema supports a comprehensive property management system, covering building administration, leasing, maintenance, staff, inspections, and tenant interactions.

Manager & Building: The `MANAGER` table stores manager details, each linked to a building via `BUILDING`, which captures location and structural attributes. Managers can also be contacted through multiple numbers stored in `MANAGER_PHONE`.

Corporate Clients: `CORPCLIENT` holds tenant company information, including referral links to other clients. Their contact numbers are maintained in `CORPCLIENT_PHONE`.

Apartments & Leasing: The `APARTMENT` table defines each unit by its building, bedroom count, and rental status. Leasing agreements are tracked in the `LEASE` table with terms, rent, deposits, and associated clients.

Staff & Cleaning: Staff records are maintained in `STAFF` with contact data in `STAFF_PHONE`. Cleaning responsibilities are mapped through the `CLEANS` table, linking staff to specific apartments.

Inspection Management: The `INSPECTOR` and `INSPECTS` tables manage inspection personnel and their assignments to buildings, including inspection and follow-up dates. Inspector contacts are stored in `INSPECTOR_PHONE`.

Maintenance Requests: `MAINTENANCE_REQUEST` logs service requests, resolution durations, and staff assignments, supporting operational tracking and performance analysis.

Resident Feedback & Utilities: Tenant-submitted issues are captured in `RESIDENT_FEEDBACK`, categorized and tracked by status. The `UTILITY_BILLING` table logs apartment-level utility costs, types, and due dates for billing purposes.

This relational design ensures robust data integrity, traceability across operations, and strong analytical potential for building management and service delivery.

7.0 Data Warehouse Schema

The schema is designed using a dimensional star model to support analytical reporting for lease agreements and service request management within the housing system. It includes clearly defined dimension and fact tables to enable time-based, client-centric, and operational analysis.

Dim_CorpClient: Captures corporate tenant details including IDs, names, contact emails, and referral links. Acts as a key entity for lease and client-based analysis.

Dim_Time: Stores temporal breakdowns such as date, month, quarter, year, day of the week, and fiscal year. Supports time-based slicing of metrics in both fact tables.

Dim_Building: Represents building entities with address, manager details, and end-of-use dates. Enables performance and usage tracking by property location and assigned managers.

Dim_Staff: Stores maintenance or service staff identities for productivity and assignment analysis across requests.

Dim_Apartment: Defines each apartment by building ID, apartment number, bedroom count, and rental status. Serves as the property unit reference in both lease and service transactions.

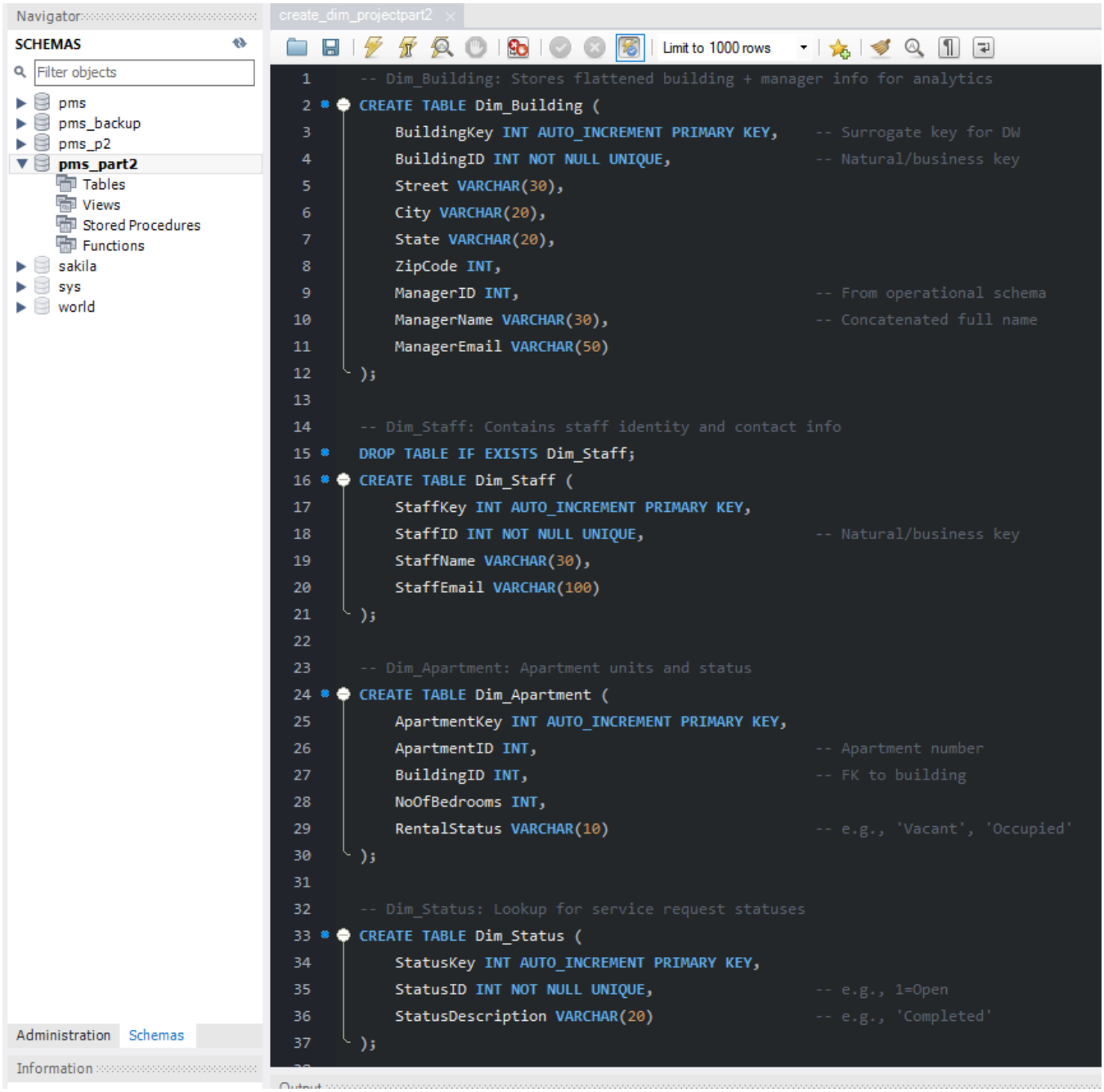
Dim_Status: Classifies service request states (e.g., open, closed, pending), useful for monitoring request pipelines and backlogs.

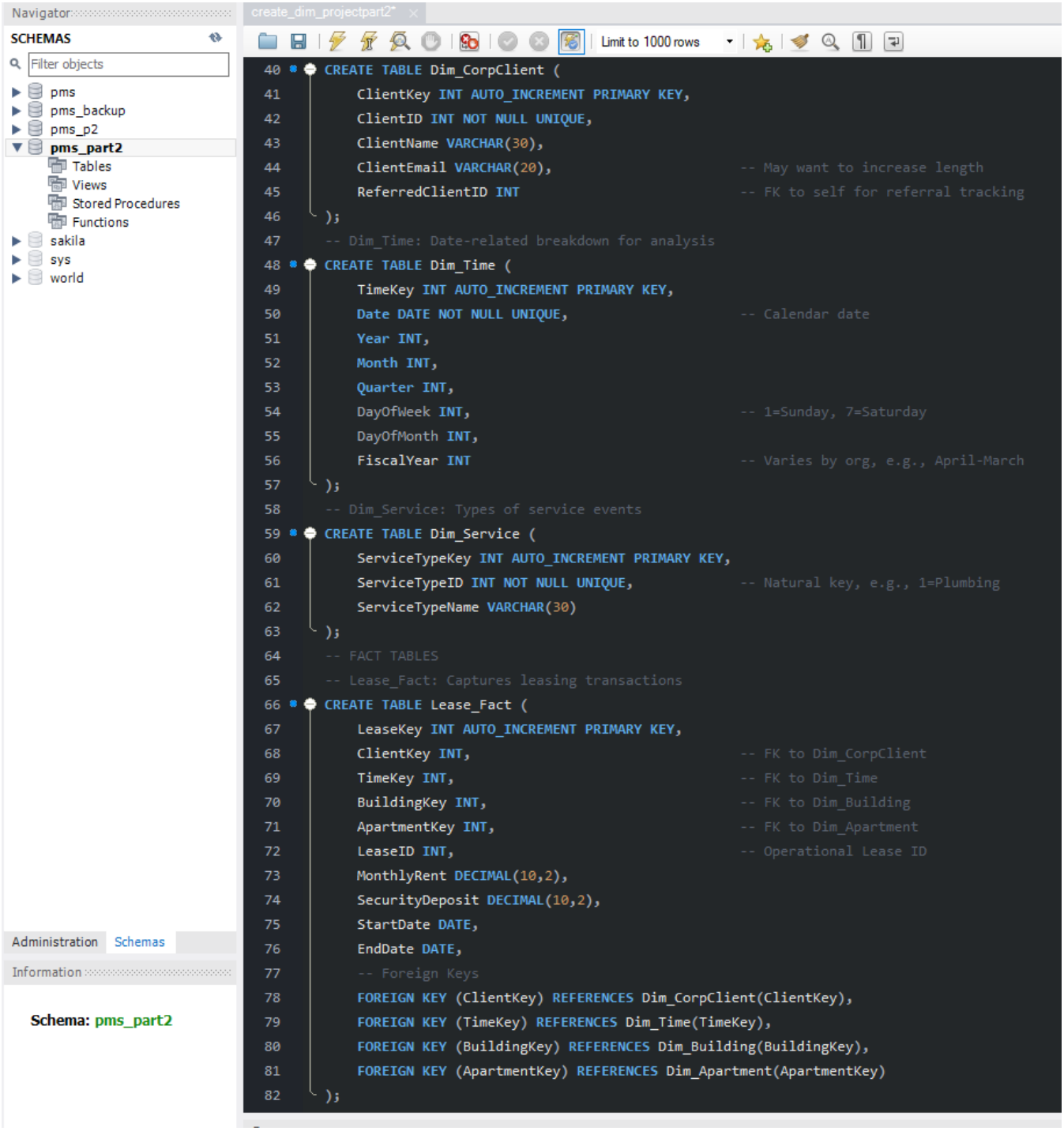
Dim_Service: Describes types of services or maintenance issues (e.g., electrical, plumbing), enabling breakdowns of operational demand.

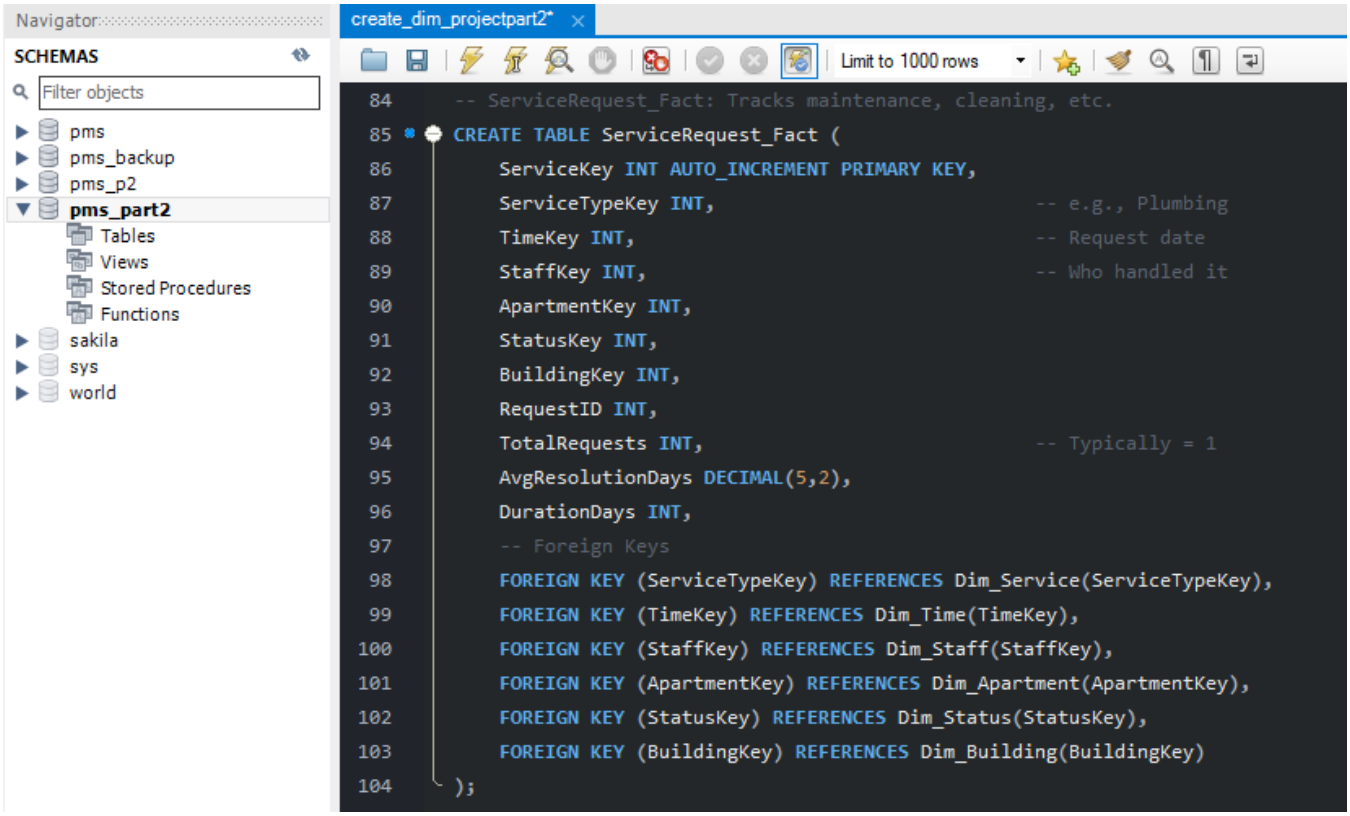
LeaseFact: Central fact table for lease agreements, linking corporate clients, properties, and lease periods. Tracks lease IDs, monthly rent, deposits, and lease duration, enabling analysis of revenue and occupancy trends.

ServiceRequestFact: Logs all tenant or manager-raised service requests with metrics such as total requests, resolution times, and inspections. Links to staff, apartments, buildings, time, status, and service types for end-to-end performance insights.

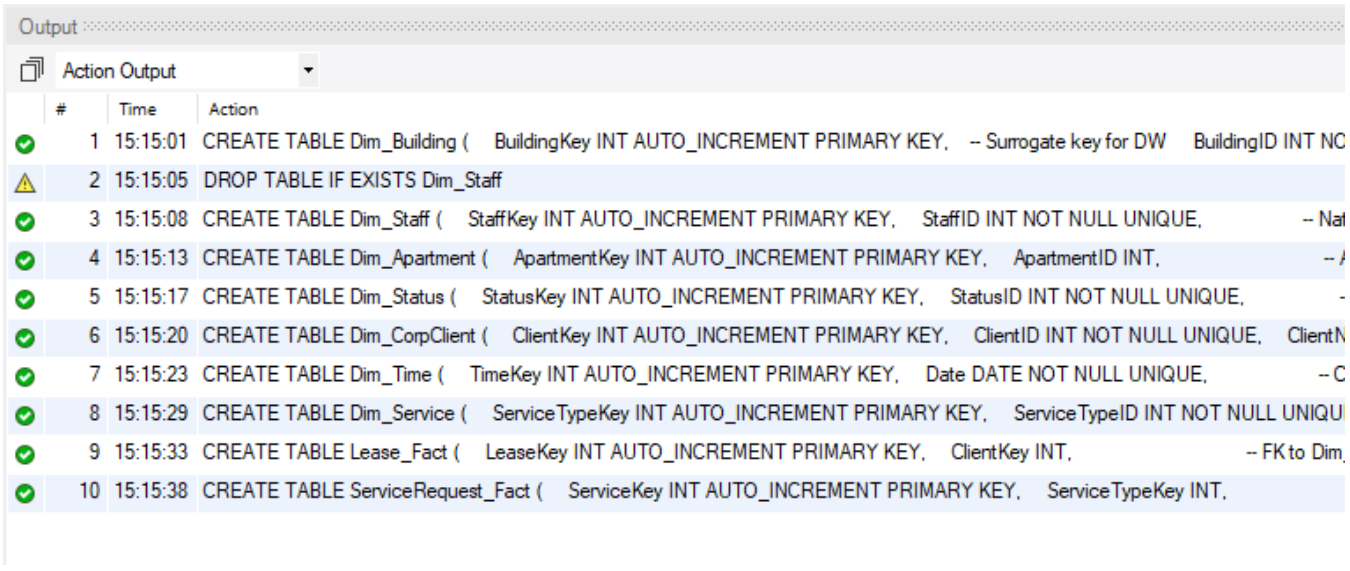
7.1 Creating Dimension & Fact Tables



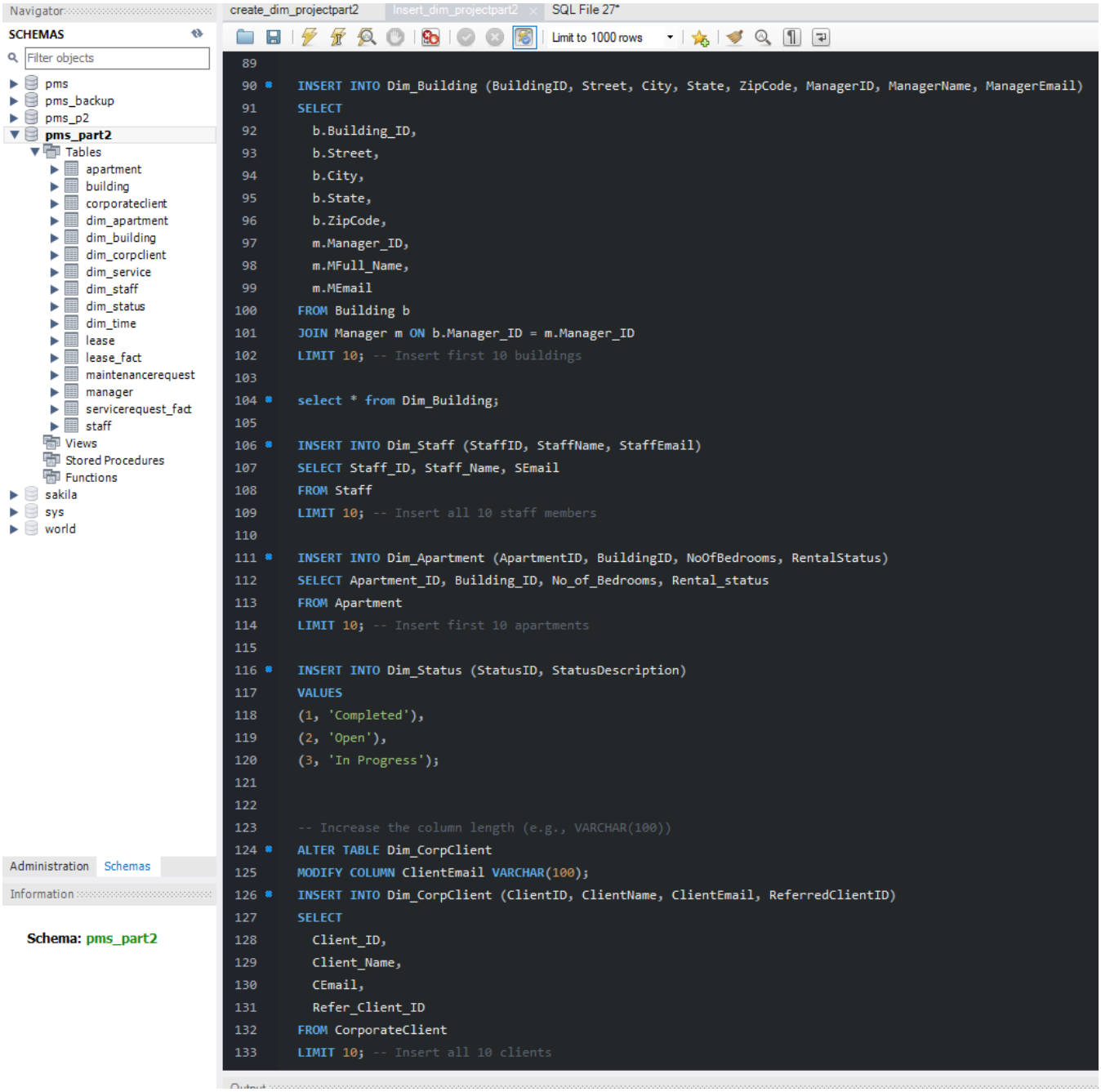

Navigator
SCHEMAS
Filter objects
pms
pms_backup
pms_p2
pms_part2
Tables
Views
Stored Procedures
Functions
sakila
sys
world
Administration
Schemas
Information
Schema: pms_part2
create_dim_projectpart2*
Limit to 1000 rows
CREATE TABLE Dim_CorpClient (
ClientKey INT AUTO_INCREMENT PRIMARY KEY,
ClientID INT NOT NULL UNIQUE,
ClientName VARCHAR(30),
ClientEmail VARCHAR(20), -- May want to increase length
ReferredClientID INT -- FK to self for referral tracking
);
-- Dim_Time: Date-related breakdown for analysis
CREATE TABLE Dim_Time (
TimeKey INT AUTO_INCREMENT PRIMARY KEY,
Date DATE NOT NULL UNIQUE, -- Calendar date
Year INT,
Month INT,
Quarter INT,
DayOfWeek INT, -- 1=Sunday, 7=Saturday
DayOfMonth INT,
FiscalYear INT -- Varies by org, e.g., April-March
);
-- Dim_Service: Types of service events
CREATE TABLE Dim_Service (
ServiceTypeKey INT AUTO_INCREMENT PRIMARY KEY,
ServiceTypeID INT NOT NULL UNIQUE, -- Natural key, e.g., 1=Plumbing
ServiceTypeName VARCHAR(30)
);
-- FACT TABLES
-- Lease_Fact: Captures leasing transactions
CREATE TABLE Lease_Fact (
LeaseKey INT AUTO_INCREMENT PRIMARY KEY,
ClientKey INT, -- FK to Dim_CorpClient
TimeKey INT, -- FK to Dim_Time
BuildingKey INT, -- FK to Dim_Building
ApartmentKey INT, -- FK to Dim_Apartment
LeaseID INT, -- Operational Lease ID
MonthlyRent DECIMAL(10,2),
SecurityDeposit DECIMAL(10,2),
StartDate DATE,
EndDate DATE,
-- Foreign Keys
FOREIGN KEY (ClientKey) REFERENCES Dim_CorpClient(ClientKey),
FOREIGN KEY (TimeKey) REFERENCES Dim_Time(TimeKey),
FOREIGN KEY (BuildingKey) REFERENCES Dim_Building(BuildingKey),
FOREIGN KEY (ApartmentKey) REFERENCES Dim_Apartment(ApartmentKey)
);

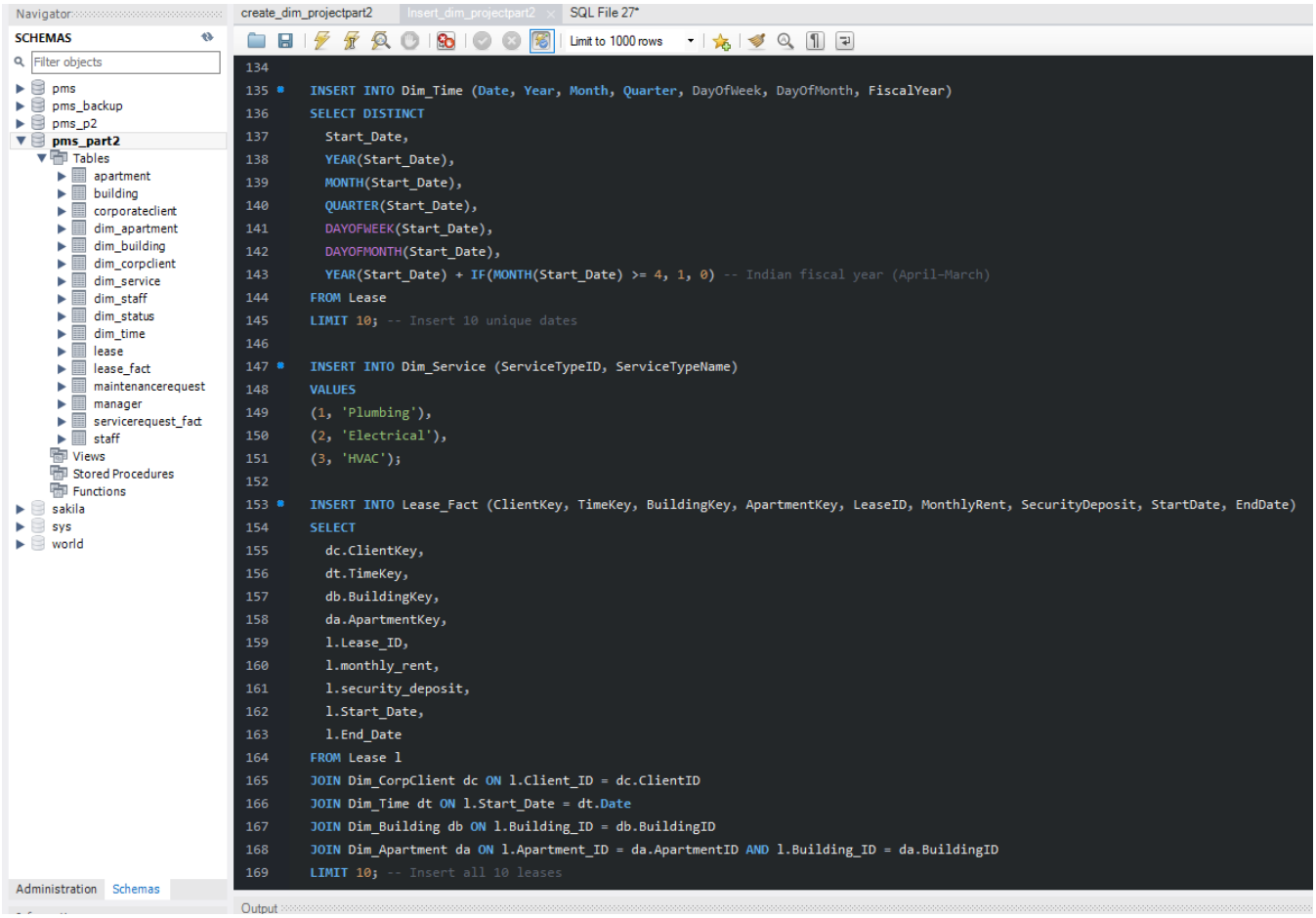


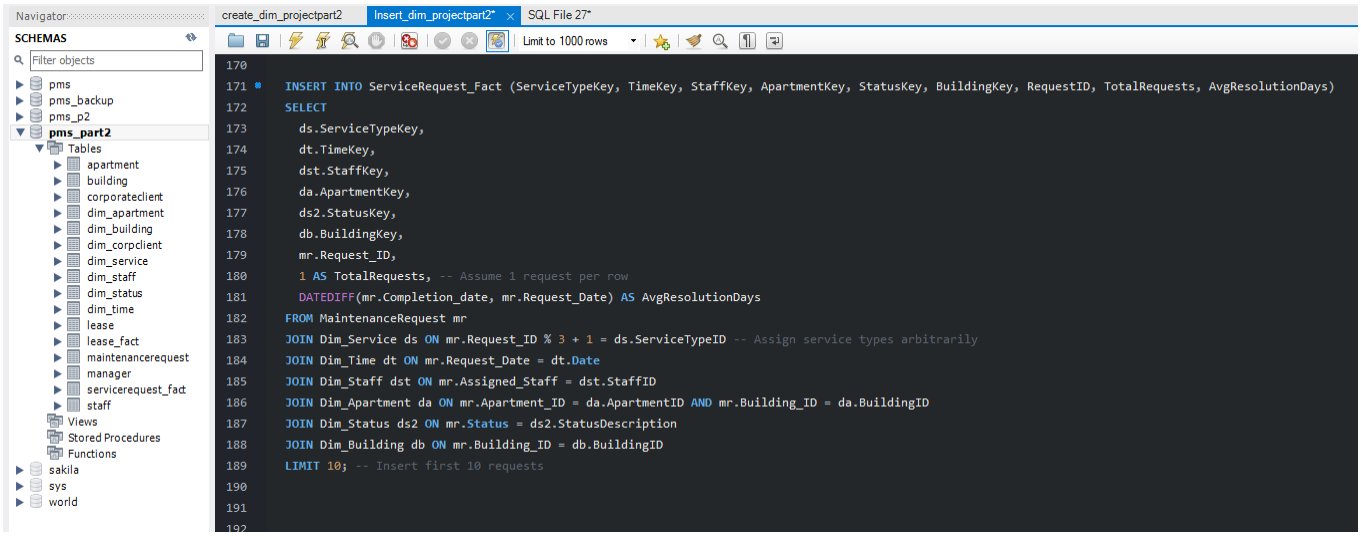
7.1.1 Success Log



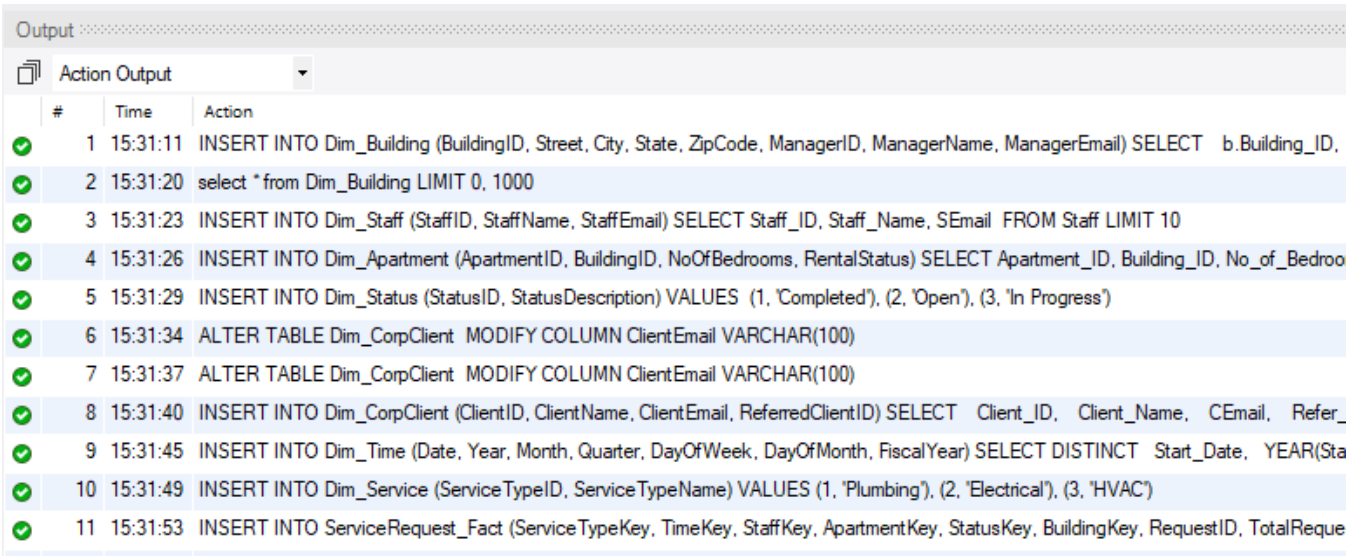
7.2 Populating the warehouse




Navigator
SCHEMAS
Filter objects
pms
pms_backup
pms_p2
pms_part2
Tables
apartment
building
corporateclient
dim_apartment
dim_building
dim_corpclient
dim_service
dim_staff
dim_status
dim_time
lease
lease_fact
maintenancerequest
manager
servicerequest_fact
staff
Views
Stored Procedures
Functions
sakila
sys
world
Administration
Schemas
create_dim_projectpart2
Insert_dim_projectpart2
SQL File 27*
Limit to 1000 rows
134
135 INSERT INTO Dim_Time (Date, Year, Month, Quarter, DayOfWeek, DayOfMonth, FiscalYear)
136 SELECT DISTINCT
137 Start_Date,
138 YEAR(Start_Date),
139 MONTH(Start_Date),
140 QUARTER(Start_Date),
141 DAYOFWEEK(Start_Date),
142 DAYOFMONTH(Start_Date),
143 YEAR(Start_Date) + IF(MONTH(Start_Date) >= 4, 1, 0) -- Indian fiscal year (April-March)
144 FROM Lease
145 LIMIT 10; -- Insert 10 unique dates
146
147 INSERT INTO Dim_Service (ServiceTypeID, ServiceTypeName)
148 VALUES
149 (1, 'Plumbing'),
150 (2, 'Electrical'),
151 (3, 'HVAC');
152
153 INSERT INTO Lease_Fact (ClientKey, TimeKey, BuildingKey, ApartmentKey, LeaseID, MonthlyRent, SecurityDeposit, StartDate, EndDate)
154 SELECT
155 dc.ClientKey,
156 dt.TimeKey,
157 db.BuildingKey,
158 da.ApartmentKey,
159 l.Lease_ID,
160 l.monthly_rent,
161 l.security_deposit,
162 l.Start_Date,
163 l.End_Date
164 FROM Lease l
165 JOIN Dim_CorpClient dc ON l.Client_ID = dc.ClientID
166 JOIN Dim_Time dt ON l.Start_Date = dt.Date
167 JOIN Dim_Building db ON l.Building_ID = db.BuildingID
168 JOIN Dim_Apartment da ON l.Apartment_ID = da.ApartmentID AND l.Building_ID = da.BuildingID
169 LIMIT 10; -- Insert all 10 leases
Output


Navigator
SCHEMAS
Filter objects
pms
pms_backup
pms_p2
pms_part2
Tables
apartment
building
corporateclient
dim_apartment
dim_building
dim_corpclient
dim_service
dim_staff
dim_status
dim_time
lease
lease_fact
maintenancerequest
manager
servicerequest_fact
staff
Views
Stored Procedures
Functions
sakila
sys
world
create_dim_projectpart2
Insert_dim_projectpart2*
SQL File 27*
Limit to 1000 rows
170
171 INSERT INTO ServiceRequest_Fact (ServiceTypeKey, TimeKey, StaffKey, ApartmentKey, StatusKey, BuildingKey, RequestID, TotalRequests, AvgResolutionDays)
172 SELECT
173 ds.ServiceTypeKey,
174 dt.TimeKey,
175 dst.StaffKey,
176 da.ApartmentKey,
177 ds2.StatusKey,
178 db.BuildingKey,
179 mr.Request_ID,
180 1 AS TotalRequests, -- Assume 1 request per row
181 DATEDIFF(mr.Completion_date, mr.Request_Date) AS AvgResolutionDays
182 FROM MaintenanceRequest mr
183 JOIN Dim_Service ds ON mr.Request_ID % 3 + 1 = ds.ServiceTypeID -- Assign service types arbitrarily
184 JOIN Dim_Time dt ON mr.Request_Date = dt.Date
185 JOIN Dim_Staff dst ON mr.Assigned_Staff = dst.StaffID
186 JOIN Dim_Apartment da ON mr.Apartment_ID = da.ApartmentID AND mr.Building_ID = da.BuildingID
187 JOIN Dim_Status ds2 ON mr.Status = ds2.StatusDescription
188 JOIN Dim_Building db ON mr.Building_ID = db.BuildingID
189 LIMIT 10; -- Insert first 10 requests
190
191
192

7.2.1 Success Log

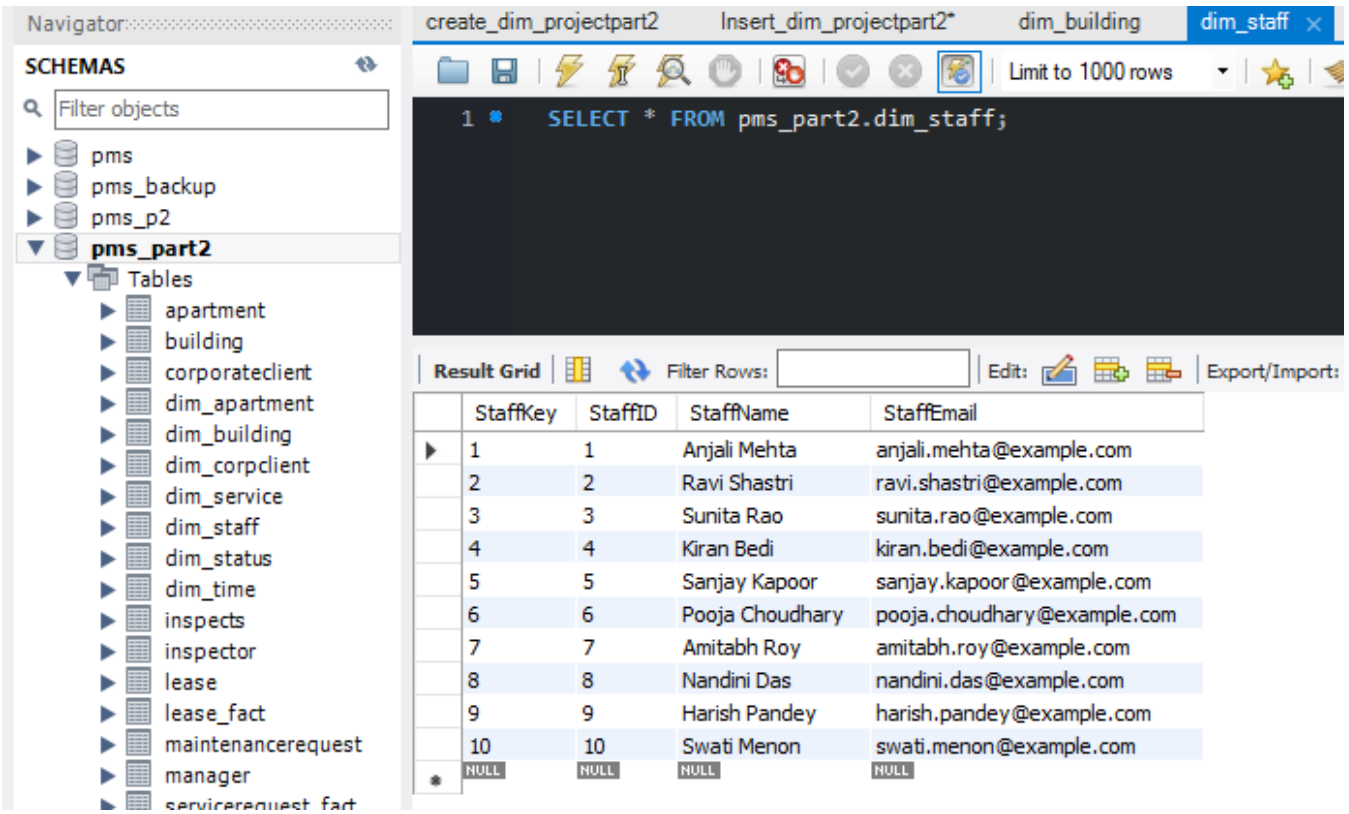


Output

Action Output

#	Time	Action
1	15:31:11	INSERT INTO Dim_Building (BuildingID, Street, City, State, ZipCode, ManagerID, ManagerName, ManagerEmail) SELECT b.Building_ID,
2	15:31:20	select * from Dim_Building LIMIT 0, 1000
3	15:31:23	INSERT INTO Dim_Staff (StaffID, StaffName, StaffEmail) SELECT Staff_ID, Staff_Name, SEmail FROM Staff LIMIT 10
4	15:31:26	INSERT INTO Dim_Apartment (ApartmentID, BuildingID, NoOfBedrooms, RentalStatus) SELECT Apartment_ID, Building_ID, No_of_Bedroo
5	15:31:29	INSERT INTO Dim_Status (StatusID, StatusDescription) VALUES (1, 'Completed'), (2, 'Open'), (3, 'In Progress')
6	15:31:34	ALTER TABLE Dim_CorpClient MODIFY COLUMN ClientEmail VARCHAR(100)
7	15:31:37	ALTER TABLE Dim_CorpClient MODIFY COLUMN ClientEmail VARCHAR(100)
8	15:31:40	INSERT INTO Dim_CorpClient (ClientID, ClientName, ClientEmail, ReferredClientID) SELECT Client_ID, Client_Name, CEmail, Refer_
9	15:31:45	INSERT INTO Dim_Time (Date, Year, Month, Quarter, DayOfWeek, DayOfMonth, FiscalYear) SELECT DISTINCT Start_Date, YEAR(Sta
10	15:31:49	INSERT INTO Dim_Service (ServiceTypeID, ServiceTypeName) VALUES (1, 'Plumbing'), (2, 'Electrical'), (3, 'HVAC')
11	15:31:53	INSERT INTO ServiceRequest_Fact (ServiceTypeKey, TimeKey, StaffKey, ApartmentKey, StatusKey, BuildingKey, RequestID, TotalReque

7.3 Retrieving the Data



Navigator

SCHEMAS

Filter objects

- pms
- pms_backup
- pms_p2
- **pms_part2**
  - Tables
    - apartment
    - building
    - corporateclient
    - dim_apartment
    - dim_building
    - dim_corpclient
    - dim_service
    - dim_staff
    - dim_status
    - dim_time
    - inspects
    - inspector
    - lease
    - lease_fact
    - maintenancerequest
    - manager
    - servicerequest_fact

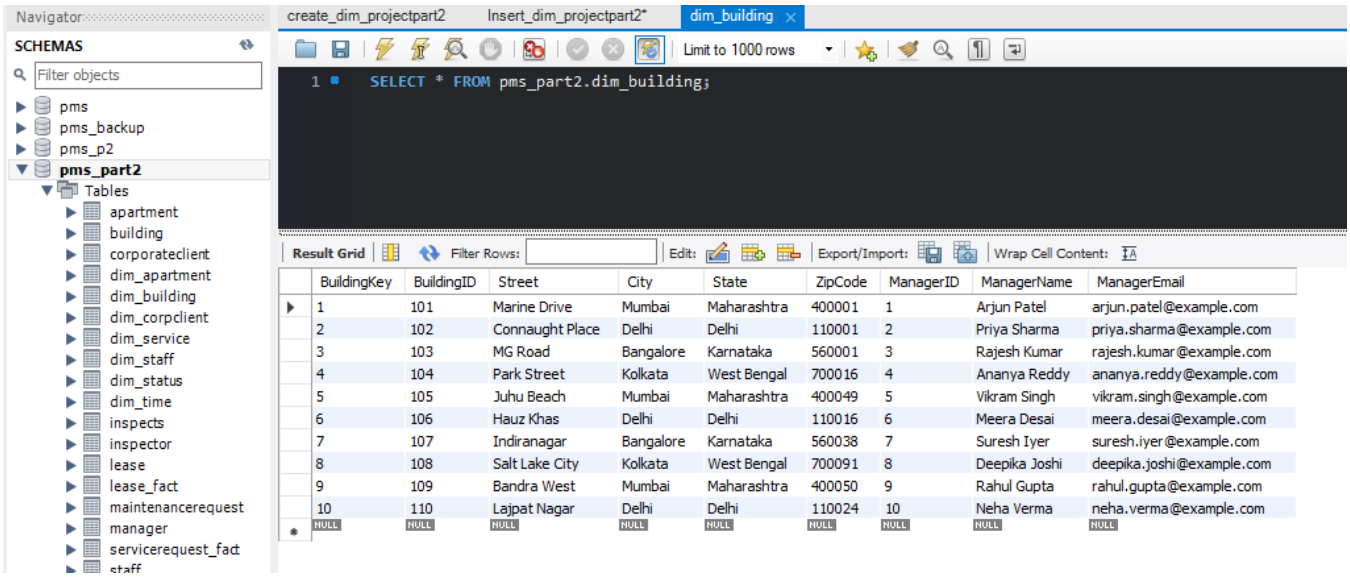
create_dim_projectpart2 | Insert_dim_projectpart2* | dim_building | dim_staff

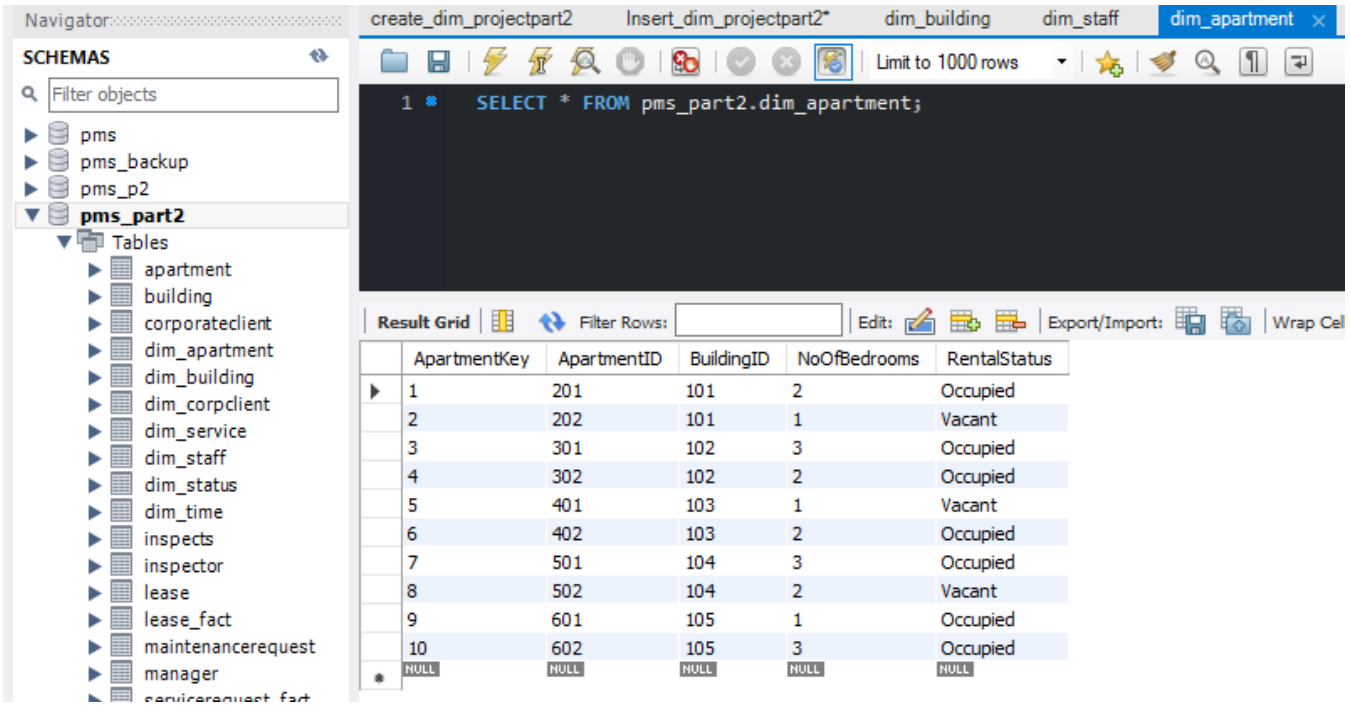
Limit to 1000 rows

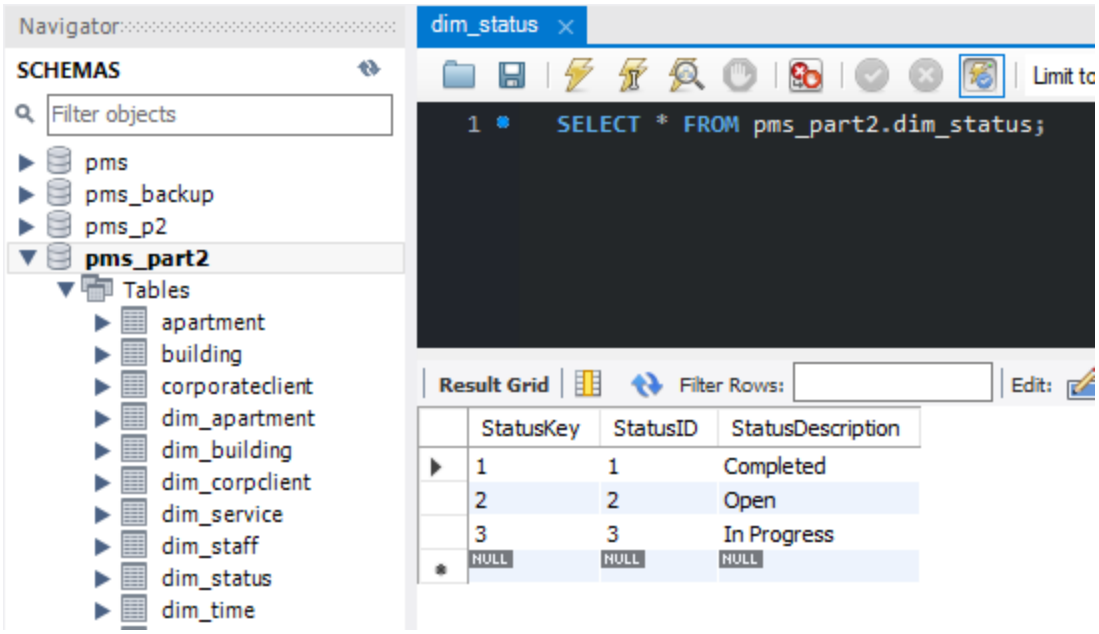
```
1 •   SELECT * FROM pms_part2.dim_staff;
```

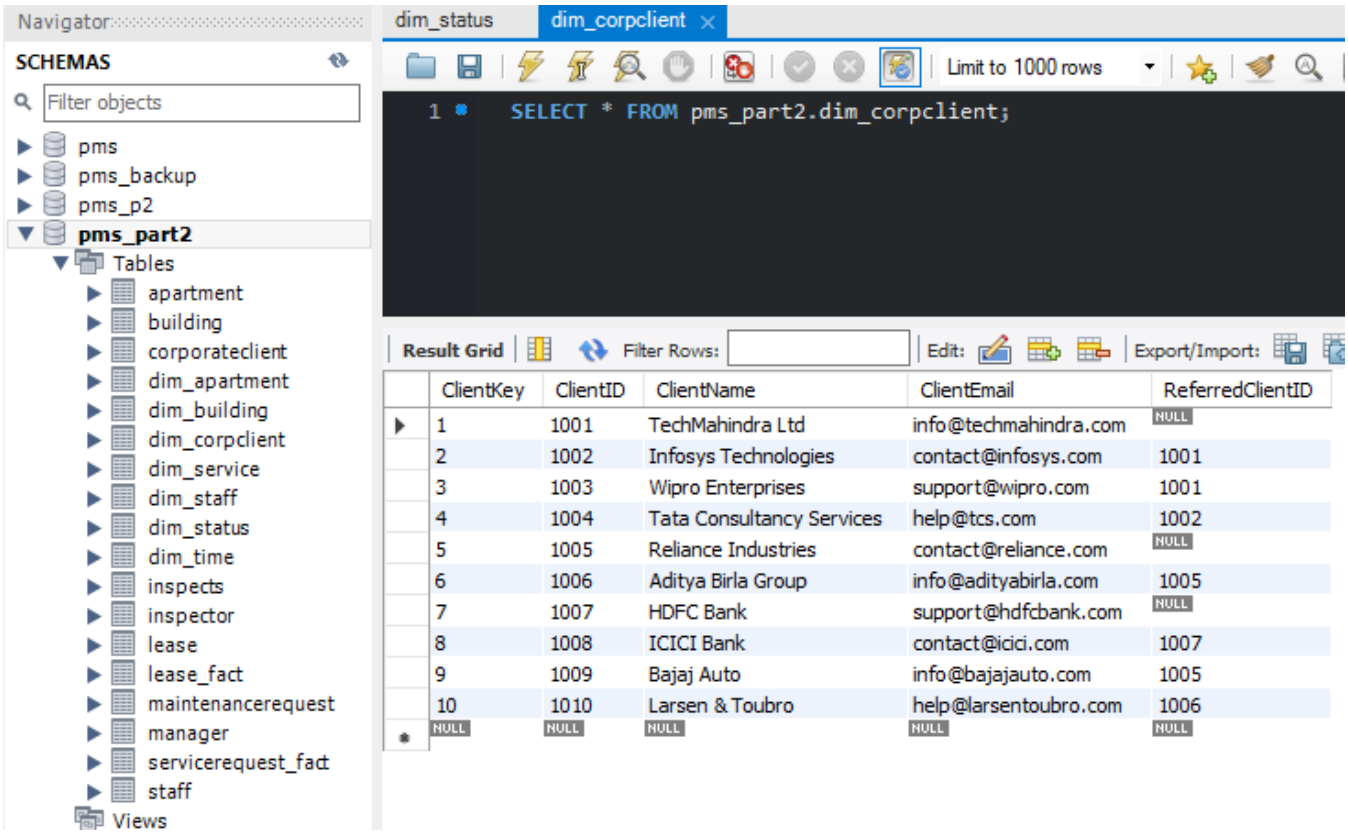
Result Grid | Filter Rows: | Edit: | Export/Import:

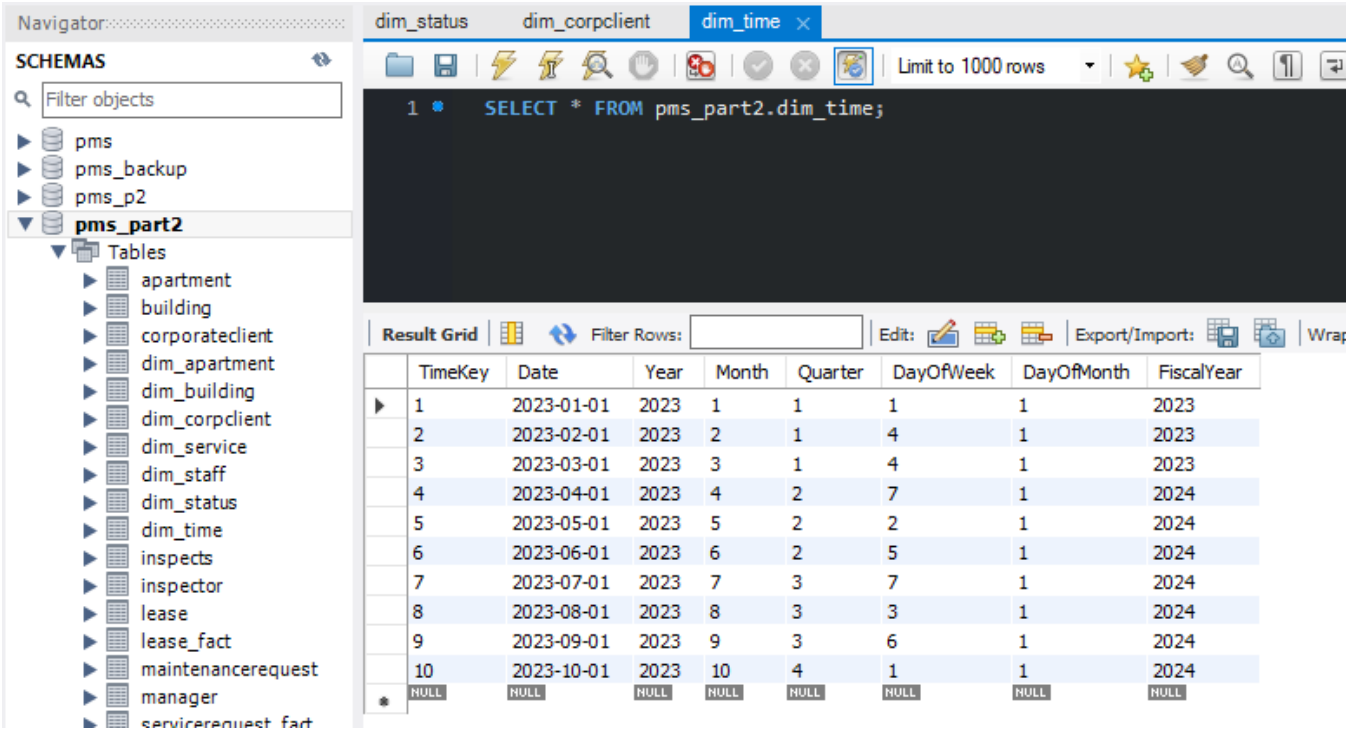
StaffKey	StaffID	StaffName	StaffEmail
1	1	Anjali Mehta	anjali.mehta@example.com
2	2	Ravi Shastri	ravi.shastri@example.com
3	3	Sunita Rao	sunita.rao@example.com
4	4	Kiran Bedi	kiran.bedi@example.com
5	5	Sanjay Kapoor	sanjay.kapoor@example.com
6	6	Pooja Choudhary	pooja.choudhary@example.com
7	7	Amitabh Roy	amitabh.roy@example.com
8	8	Nandini Das	nandini.das@example.com
9	9	Harish Pandey	harish.pandey@example.com
10	10	Swati Menon	swati.menon@example.com
NULL	NULL	NULL	NULL

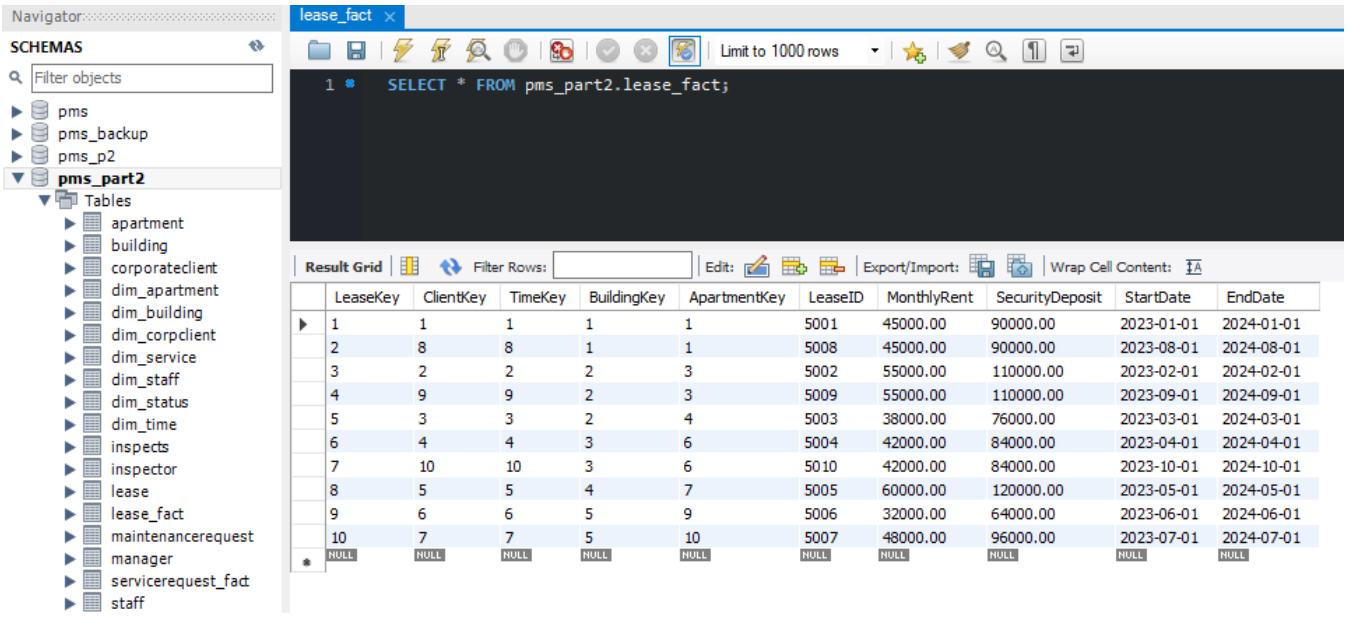

Navigator
SCHEMAS
Filter objects
pms
pms_backup
pms_p2
pms_part2
Tables
apartment
building
corporateclient
dim_apartment
dim_building
dim_corpclient
dim_service
dim_staff
dim_status
dim_time
inspects
inspector
lease
lease_fact
maintenancerequest
manager
servicerequest_fact
staff
create_dim_projectpart2
Insert_dim_projectpart2*
dim_building
Limit to 1000 rows
1 SELECT * FROM pms_part2.dim_building;
Result Grid
Filter Rows:
Edit:
Export/Import:
Wrap Cell Content:
BuildingKey BuildingID Street City State ZipCode ManagerID ManagerName ManagerEmail
1 101 Marine Drive Mumbai Maharashtra 400001 1 Arjun Patel arjun.patel@example.com
2 102 Connaught Place Delhi Delhi 110001 2 Priya Sharma priya.sharma@example.com
3 103 MG Road Bangalore Karnataka 560001 3 Rajesh Kumar rajesh.kumar@example.com
4 104 Park Street Kolkata West Bengal 700016 4 Ananya Reddy ananya.reddy@example.com
5 105 Juhu Beach Mumbai Maharashtra 400049 5 Vikram Singh vikram.singh@example.com
6 106 Hauz Khas Delhi Delhi 110016 6 Meera Desai meera.desai@example.com
7 107 Indiranagar Bangalore Karnataka 560038 7 Suresh Iyer suresh.iyer@example.com
8 108 Salt Lake City Kolkata West Bengal 700091 8 Deepika Joshi deepika.joshi@example.com
9 109 Bandra West Mumbai Maharashtra 400050 9 Rahul Gupta rahul.gupta@example.com
10 110 Lajpat Nagar Delhi Delhi 110024 10 Neha Verma neha.verma@example.com
NULL

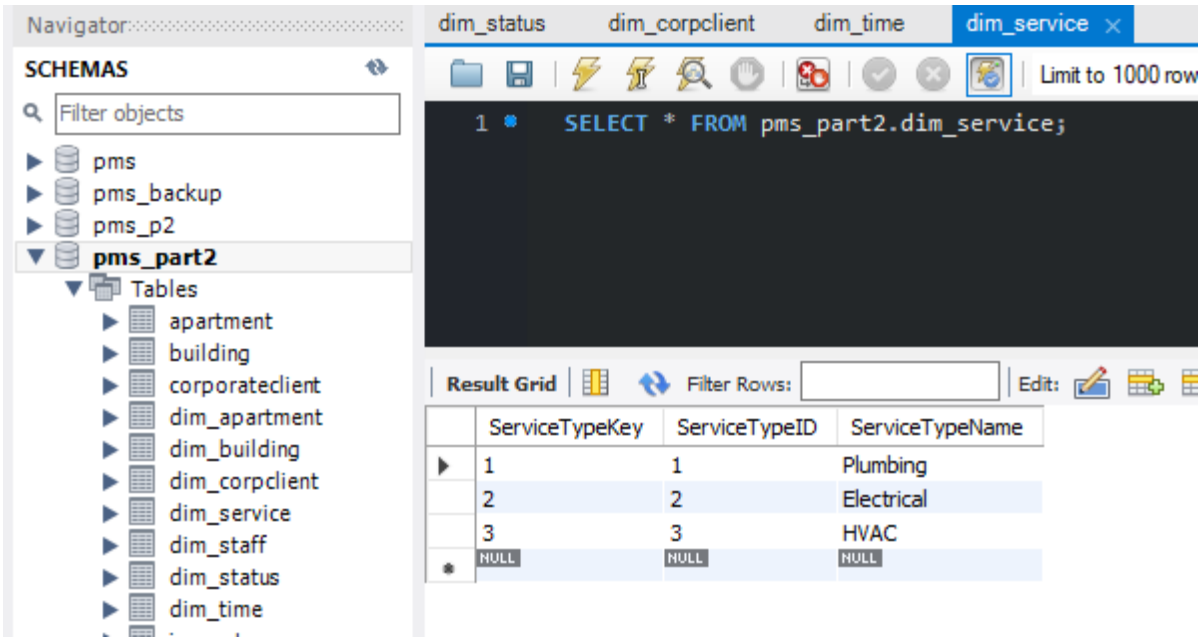

Navigator
SCHEMAS
Filter objects
pms
pms_backup
pms_p2
pms_part2
Tables
apartment
building
corporateclient
dim_apartment
dim_building
dim_corpclient
dim_service
dim_staff
dim_status
dim_time
inspects
inspector
lease
lease_fact
maintenancerequest
manager
create_dim_projectpart2
Insert_dim_projectpart2*
dim_building
dim_staff
dim_apartment
Limit to 1000 rows
1 SELECT * FROM pms_part2.dim_apartment;
Result Grid
Filter Rows:
Edit:
Export/Import:
Wrap Cel
ApartmentKey ApartmentID BuildingID NoOfBedrooms RentalStatus
1 201 101 2 Occupied
2 202 101 1 Vacant
3 301 102 3 Occupied
4 302 102 2 Occupied
5 401 103 1 Vacant
6 402 103 2 Occupied
7 501 104 3 Occupied
8 502 104 2 Vacant
9 601 105 1 Occupied
10 602 105 3 Occupied
NULL

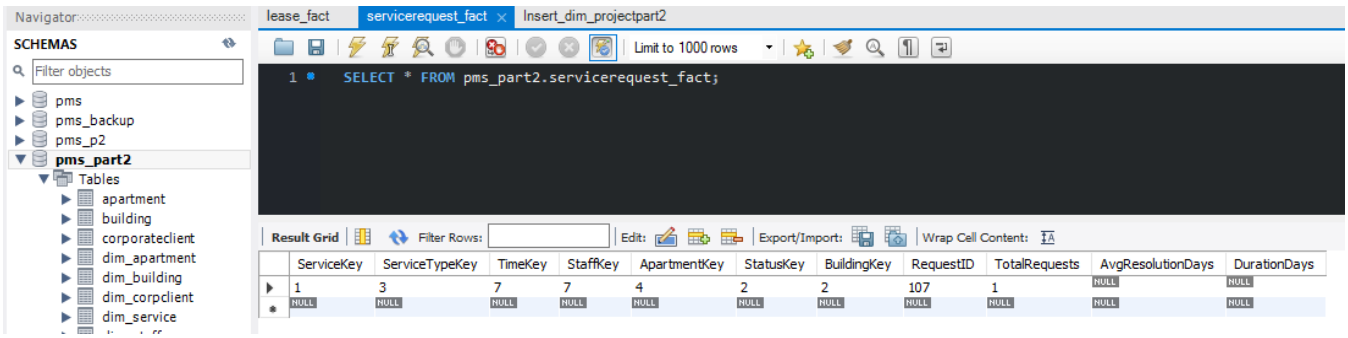

Navigator
SCHEMAS
Filter objects
pms
pms_backup
pms_p2
pms_part2
Tables
apartment
building
corporateclient
dim_apartment
dim_building
dim_corpclient
dim_service
dim_staff
dim_status
dim_time
dim_status
Limit to
1 SELECT * FROM pms_part2.dim_status;
Result Grid
Filter Rows:
Edit:
StatusKey StatusID StatusDescription
1 1 Completed
2 2 Open
3 3 In Progress
NULL NULL NULL


Navigator
SCHEMAS
Filter objects
pms
pms_backup
pms_p2
pms_part2
Tables
apartment
building
corporateclient
dim_apartment
dim_building
dim_corpclient
dim_service
dim_staff
dim_status
dim_time
inspects
inspector
lease
lease_fact
maintenancerequest
manager
servicerequest_fact
staff
Views
dim_status
dim_corpclient
Limit to 1000 rows
1 SELECT * FROM pms_part2.dim_corpclient;
Result Grid
Filter Rows:
Edit:
Export/Import:
ClientKey ClientID ClientName ClientEmail ReferredClientID
1 1001 TechMahindra Ltd info@techmahindra.com NULL
2 1002 Infosys Technologies contact@infosys.com 1001
3 1003 Wipro Enterprises support@wipro.com 1001
4 1004 Tata Consultancy Services help@tcs.com 1002
5 1005 Reliance Industries contact@reliance.com NULL
6 1006 Aditya Birla Group info@adityabirla.com 1005
7 1007 HDFC Bank support@hdfcbank.com NULL
8 1008 ICICI Bank contact@icici.com 1007
9 1009 Bajaj Auto info@bajajauto.com 1005
10 1010 Larsen & Toubro help@larsentoubro.com 1006
NULL NULL NULL NULL NULL


Navigator
SCHEMAS
Filter objects
pms
pms_backup
pms_p2
pms_part2
Tables
apartment
building
corporateclient
dim_apartment
dim_building
dim_corpclient
dim_service
dim_staff
dim_status
dim_time
inspects
inspector
lease
lease_fact
maintenancerequest
manager
dim_status
dim_corpclient
dim_time
Limit to 1000 rows
1 SELECT * FROM pms_part2.dim_time;
Result Grid
Filter Rows:
Edit:
Export/Import:
TimeKey Date Year Month Quarter DayOfWeek DayOfMonth FiscalYear
1 2023-01-01 2023 1 1 1 1 2023
2 2023-02-01 2023 2 1 4 1 2023
3 2023-03-01 2023 3 1 4 1 2023
4 2023-04-01 2023 4 2 7 1 2024
5 2023-05-01 2023 5 2 2 1 2024
6 2023-06-01 2023 6 2 5 1 2024
7 2023-07-01 2023 7 3 7 1 2024
8 2023-08-01 2023 8 3 3 1 2024
9 2023-09-01 2023 9 3 6 1 2024
10 2023-10-01 2023 10 4 1 1 2024


Navigator
SCHEMAS
Filter objects
pms
pms_backup
pms_p2
pms_part2
Tables
apartment
building
corporateclient
dim_apartment
dim_building
dim_corpclient
dim_service
dim_staff
dim_status
dim_time
inspects
inspector
lease
lease_fact
maintenancerequest
manager
servicerequest_fact
staff
lease_fact
Limit to 1000 rows
1 SELECT * FROM pms_part2.lease_fact;
Result Grid
Filter Rows:
Edit:
Export/Import:
Wrap Cell Content:
LeaseKey ClientKey TimeKey BuildingKey ApartmentKey LeaseID MonthlyRent SecurityDeposit StartDate EndDate
1 1 1 1 1 5001 45000.00 90000.00 2023-01-01 2024-01-01
2 8 8 1 1 5008 45000.00 90000.00 2023-08-01 2024-08-01
3 2 2 2 3 5002 55000.00 110000.00 2023-02-01 2024-02-01
4 9 9 2 3 5009 55000.00 110000.00 2023-09-01 2024-09-01
5 3 3 2 4 5003 38000.00 76000.00 2023-03-01 2024-03-01
6 4 4 3 6 5004 42000.00 84000.00 2023-04-01 2024-04-01
7 10 10 3 6 5010 42000.00 84000.00 2023-10-01 2024-10-01
8 5 5 4 7 5005 60000.00 120000.00 2023-05-01 2024-05-01
9 6 6 5 9 5006 32000.00 64000.00 2023-06-01 2024-06-01
10 7 7 5 10 5007 48000.00 96000.00 2023-07-01 2024-07-01





8.0 ETL Process:

8.1 Purpose of the ETL Script

The script transforms raw transactional data from an OLTP database into a structured star schema for analytical reporting. It populates dimension tables (descriptive entities) and fact tables (transactional metrics) to enable business intelligence queries.

8.1.1 Key Components

Stored Procedure: PopulateDimTime

- **Objective**: Precompute time-related attributes for a date range (2020–2030) to support time-based aggregations.
- **Logic**:
    - Uses a loop to generate a row for every date between 2020-01-01 and 2030-12-31.
    - Extracts temporal attributes:
        - Year, Quarter, Month, DayOfWeek, DayOfMonth.
        - Derives FiscalYear as YEAR + 1 if the month is July or later (assumes fiscal year starts in July).
- **Importance**: Serves as a universal time reference for all date-based operations in fact tables.

8.1.2 Dimension Table Population

- Dim_Building:
    - **Source**: BUILDING (address details) and MANAGER (manager name).
    - **Transformation**: Combines street, city, state, and ZIP code into a single BuildingAddress field.
    - Links buildings to their assigned managers via Manager_ID.
- Dim_Apartment:
    - **Source**: APARTMENT table.
    - Maps raw apartment attributes (Apartment_ID, No_of_Bedrooms, Rental_Status) directly to the dimension.
- Dim_CorpClient:
    - **Source**: CORPCLIENT table.
    - Uses NOT EXISTS to ensure uniqueness, preventing duplicate client entries during incremental loads.
    - Preserves referral relationships via Refer_Client_ID.
- Dim_Staff:
    - **Source**: STAFF table.

- Directly maps staff IDs and names for tracking personnel assignments.

8.1.3 Fact Table Population

- LeaseFact:
  - **Source**: LEASE table (transactional lease records).
  - **Joins**:
    - Dim_CorpClient to link tenants to leases.
    - Dim_Time to associate lease start dates with temporal attributes.
    - Dim_Building and Dim_Apartment to tie leases to physical units.
  - **Measures**: Captures financial metrics (monthly_rent, security_deposit) and lease duration (start_date, end_date).
- ServiceRequestFact:
  - **Source**: MAINTENANCE_REQUEST table.
  - **Joins**:
    - Dim_Time to track request dates.
    - Dim_Staff to identify assigned personnel.
    - Dim_Apartment and Dim_Building for unit/location context.
  - **Calculations**:
    - AvgResolutionDays: Computed as DATEDIFF(Completion_Date, Request_Date) for resolved requests.
    - DurationDays: Uses CURRENT_DATE for unresolved requests to calculate ongoing resolution time.
  - **Metrics**: Aggregates request counts (TotalRequests) and ties inspections to resolution efficiency.

8.1.4 Critical Design Choices

1. **Time Dimension Prepopulation**:
   - A static Dim_Time table is created upfront to avoid runtime date calculations during fact table loads. This optimizes query performance.

2. **Referential Integrity**:
    - Foreign keys (e.g., BuildingKey, ClientKey) enforce relationships between fact and dimension tables, ensuring data consistency.
3. **Handling Unresolved Requests**:
    - For open maintenance requests (Completion_Date IS NULL), the script uses CURRENT_DATE to compute DurationDays, providing real-time visibility into unresolved issues.
4. **Fiscal Year Logic**:
    - Assumes a July–June fiscal calendar. This logic is hardcoded but can be modified if the fiscal year aligns with the calendar year.
5. **Incremental Loads**:
    - The NOT EXISTS clause in Dim_CorpClient prevents duplicate client entries, making the script idempotent (safe for reruns).

8.1.5 Workflow Summary

1. **Dimensions First**:
    - Static dimensions (Dim_Time, Dim_Building, etc.) are populated first to serve as lookup tables for fact data.
2. **Facts Second**:
    - Transactional data from LEASE and MAINTENANCE_REQUEST is loaded into fact tables, enriched with dimension keys.
3. **Dependency Management**:
    - The order of operations ensures foreign key constraints are respected (e.g., Dim_Time must exist before LeaseFact is populated).

8.1.6 Business Value

- **Unified Reporting**: Combines scattered OLTP data into a single schema for cross-functional analysis (e.g., linking lease revenue to maintenance costs).
- **Performance**: Star schema optimizes query speed for dashboards and reports.
- **Scalability**: Designed to handle incremental data loads without reprocessing historical data.

This ETL script transforms raw operational data into an analytical goldmine, enabling stakeholders to derive insights on occupancy trends, maintenance efficiency, and financial performance.

8.2 Why Truncating

8.2.1 Full Refresh Strategy

- Our ETL process is designed as a full refresh — every run:
- Deletes old data (via TRUNCATE)
- Reloads everything cleanly from the source
- This avoids duplication, inconsistencies, or partial updates

8.2.2 Prevent Orphaned/Outdated Records

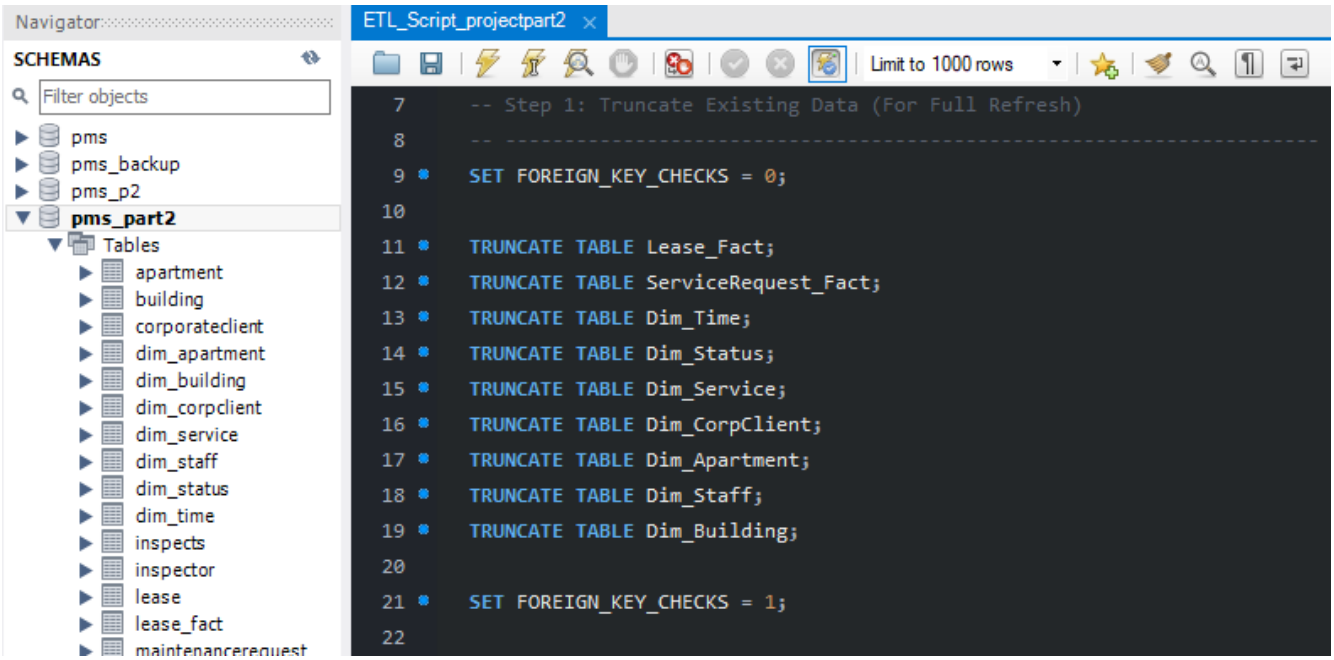
- If dimension keys or mappings change (e.g., building IDs updated), keeping old fact rows might break foreign key relationships
- Truncating ensures the fact and dimension tables are synchronized

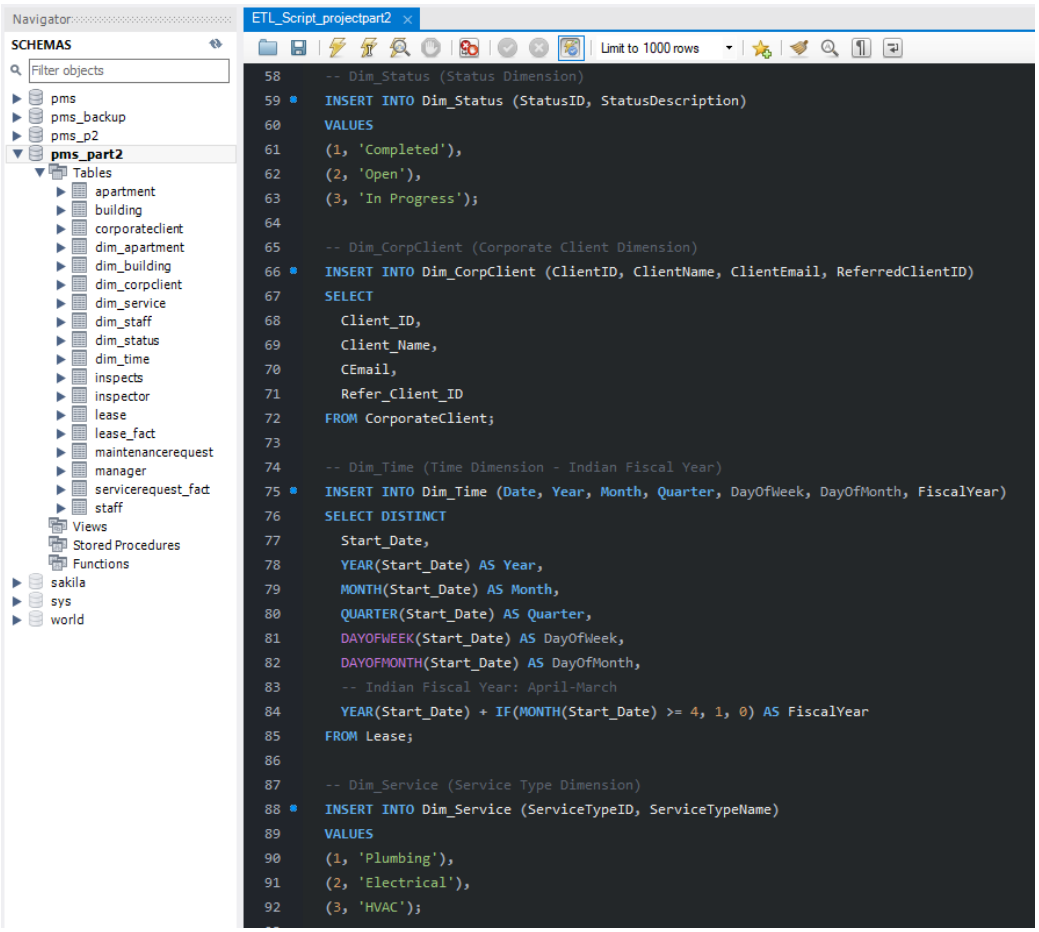
8.2.3 Clean Staging for Testing or Development

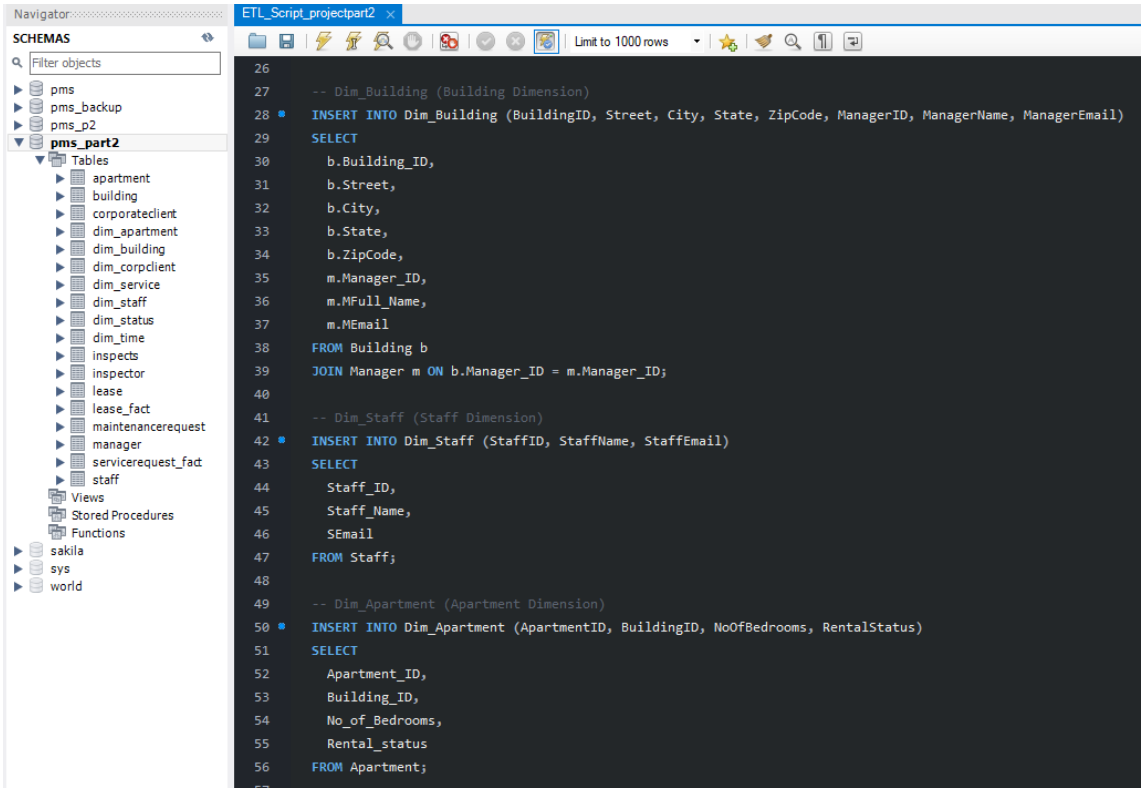
- In academic or controlled environments, you often test multiple ETL cycles
- Truncation ensures that each cycle starts from scratch, so results are repeatable

8.2.4 Better Performance

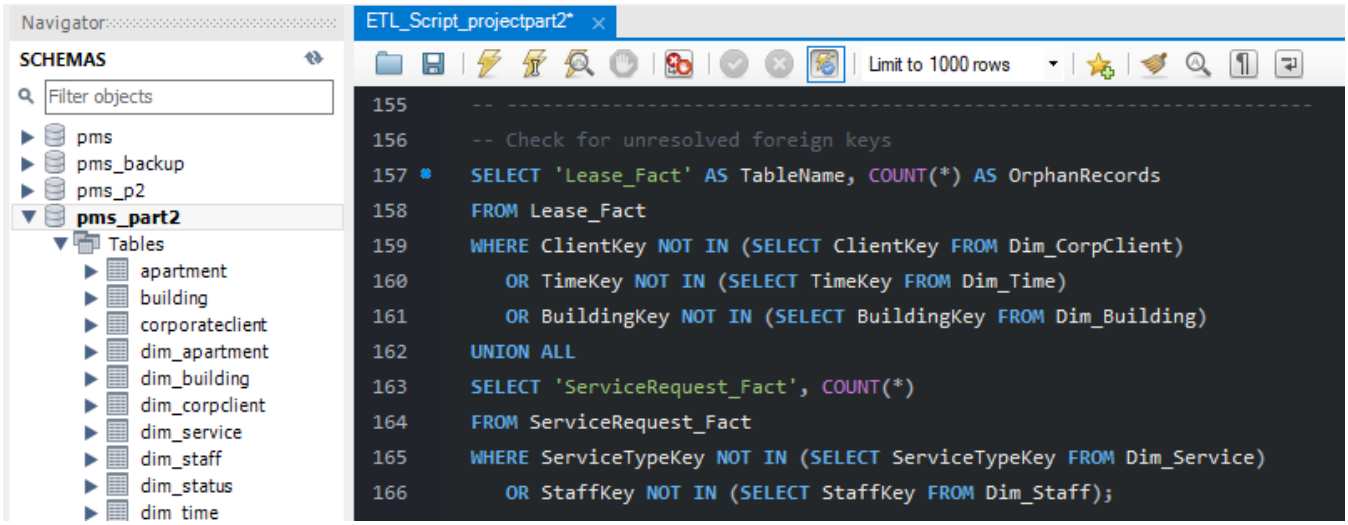
- TRUNCATE is faster than DELETE and resets auto-increment IDs
- Ideal when you don't need to preserve old data between loads




Navigator
SCHEMAS
Filter objects
pms
pms_backup
pms_p2
pms_part2
Tables
apartment
building
corporateclient
dim_apartment
dim_building
dim_corpclient
dim_service
dim_staff
dim_status
dim_time
inspects
inspector
lease
lease_fact
maintenancerequest
manager
servicerequest_fact
staff
Views
Stored Procedures
Functions
sakila
sys
world
ETL_Script_projectpart2
Limit to 1000 rows
-- Dim_Status (Status Dimension)
INSERT INTO Dim_Status (StatusID, StatusDescription)
VALUES
(1, 'Completed'),
(2, 'Open'),
(3, 'In Progress');
-- Dim_CorpClient (Corporate Client Dimension)
INSERT INTO Dim_CorpClient (ClientID, ClientName, ClientEmail, ReferredClientID)
SELECT
Client_ID,
Client_Name,
CEmail,
Refer_Client_ID
FROM CorporateClient;
-- Dim_Time (Time Dimension - Indian Fiscal Year)
INSERT INTO Dim_Time (Date, Year, Month, Quarter, DayOfWeek, DayOfMonth, FiscalYear)
SELECT DISTINCT
Start_Date,
YEAR(Start_Date) AS Year,
MONTH(Start_Date) AS Month,
QUARTER(Start_Date) AS Quarter,
DAYOFWEEK(Start_Date) AS DayOfWeek,
DAYOFMONTH(Start_Date) AS DayOfMonth,
-- Indian Fiscal Year: April-March
YEAR(Start_Date) + IF(MONTH(Start_Date) >= 4, 1, 0) AS FiscalYear
FROM Lease;
-- Dim_Service (Service Type Dimension)
INSERT INTO Dim_Service (ServiceTypeID, ServiceTypeName)
VALUES
(1, 'Plumbing'),
(2, 'Electrical'),
(3, 'HVAC');


Navigator
SCHEMAS
Filter objects
pms
pms_backup
pms_p2
pms_part2
Tables
apartment
building
corporateclient
dim_apartment
dim_building
dim_corpclient
dim_service
dim_staff
dim_status
dim_time
inspects
inspector
lease
lease_fact
maintenancerequest
manager
servicerequest_fact
staff
Views
Stored Procedures
Functions
sakila
sys
world
ETL_Script_projectpart2
Limit to 1000 rows
-- Dim_Building (Building Dimension)
INSERT INTO Dim_Building (BuildingID, Street, City, State, ZipCode, ManagerID, ManagerName, ManagerEmail)
SELECT
b.Building_ID,
b.Street,
b.City,
b.State,
b.ZipCode,
m.Manager_ID,
m.MFull_Name,
m.MEmail
FROM Building b
JOIN Manager m ON b.Manager_ID = m.Manager_ID;
-- Dim_Staff (Staff Dimension)
INSERT INTO Dim_Staff (StaffID, StaffName, StaffEmail)
SELECT
Staff_ID,
Staff_Name,
SEmail
FROM Staff;
-- Dim_Apartment (Apartment Dimension)
INSERT INTO Dim_Apartment (ApartmentID, BuildingID, NoOfBedrooms, RentalStatus)
SELECT
Apartment_ID,
Building_ID,
No_of_Bedrooms,
Rental_status
FROM Apartment;


Navigator
SCHEMAS
Filter objects
pms
pms_backup
pms_p2
pms_part2
Tables
apartment
building
corporateclient
dim_apartment
dim_building
dim_corpclient
dim_service
dim_staff
dim_status
dim_time
inspects
inspector
lease
lease_fact
maintenancerequest
manager
servicerequest_fact
staff
Views
Stored Procedures
Functions
sakila
sys
world
Administration
Schemas
Information
Schema: pms_part2
ETL_Script_projectpart2
Limit to 1000 rows
INSERT INTO Lease_Fact (
ClientKey, TimeKey, BuildingKey, ApartmentKey, LeaseID,
MonthlyRent, SecurityDeposit, StartDate, EndDate
)
SELECT
dc.ClientKey,
dt.TimeKey,
db.BuildingKey,
da.ApartmentKey,
l.Lease_ID,
l.Monthly_Rent,
l.Security_Deposit,
l.Start_Date,
l.End_Date
FROM Lease l
JOIN Dim_CorpClient dc ON l.Client_ID = dc.ClientID
JOIN Dim_Time dt ON l.Start_Date = dt.Date
JOIN Dim_Building db ON l.Building_ID = db.BuildingID
JOIN Dim_Apartment da ON l.Apartment_ID = da.ApartmentID AND l.Building_ID = da.BuildingID;
-- ServiceRequest_Fact (Maintenance Metrics)
INSERT INTO ServiceRequest_Fact (
ServiceTypeKey, TimeKey, StaffKey, ApartmentKey,
StatusKey, BuildingKey, RequestID, TotalRequests,
AvgResolutionDays, DurationDays
)
SELECT
ds.ServiceTypeKey,
dt.TimeKey,
dst.StaffKey,
da.ApartmentKey,
ds2.StatusKey,
db.BuildingKey,
mr.Request_ID,
1 AS TotalRequests,
DATEDIFF(mr.Completion_date, mr.Request_Date) AS AvgResolutionDays,
DATEDIFF(COALESCE(mr.Completion_date, CURDATE()), mr.Request_Date) AS DurationDays
FROM MaintenanceRequest mr
JOIN Dim_Service ds ON
CASE
WHEN mr.Description LIKE '%plumb%' THEN 1
WHEN mr.Description LIKE '%electr%' THEN 2
ELSE 3
END = ds.ServiceTypeID
JOIN Dim_Time dt ON mr.Request_Date = dt.Date
JOIN Dim_Staff dst ON mr.Assigned_Staff = dst.StaffID
JOIN Dim_Apartment da ON mr.Apartment_ID = da.ApartmentID
AND mr.Building_ID = da.BuildingID
JOIN Dim_Status ds2 ON mr.Status = ds2.StatusDescription
JOIN Dim_Building db ON mr.Building_ID = db.BuildingID;



Output

Action Output

#	Time	Action
1	15:49:51	SET FOREIGN_KEY_CHECKS = 0
2	15:49:54	TRUNCATE TABLE Lease_Fact
3	15:49:56	TRUNCATE TABLE ServiceRequest_Fact
4	15:49:58	TRUNCATE TABLE Dim_Time
5	15:49:59	TRUNCATE TABLE Dim_Status
6	15:50:00	TRUNCATE TABLE Dim_Service
7	15:50:01	TRUNCATE TABLE Dim_CorpClient
8	15:50:02	TRUNCATE TABLE Dim_Apartment
9	15:50:03	TRUNCATE TABLE Dim_Staff
10	15:50:04	TRUNCATE TABLE Dim_Building
11	15:50:05	SET FOREIGN_KEY_CHECKS = 1
12	15:50:14	INSERT INTO Dim_Building (BuildingID, Street, City, State, ZipCode, ManagerID, ManagerName, ManagerEmail) SELECT b.Building_ID, b.Street, b.City, b.S
13	15:50:18	INSERT INTO Dim_Staff (StaffID, StaffName, StaffEmail) SELECT Staff_ID, Staff_Name, SEmail FROM Staff
14	15:50:21	INSERT INTO Dim_Apartment (ApartmentID, BuildingID, NoOfBedrooms, RentalStatus) SELECT Apartment_ID, Building_ID, No_of_Bedrooms, Rental_statu
15	15:50:24	INSERT INTO Dim_Status (StatusID, StatusDescription) VALUES (1, 'Completed'), (2, 'Open'), (3, 'In Progress')
16	15:50:31	INSERT INTO Dim_CorpClient (ClientID, ClientName, ClientEmail, ReferredClientID) SELECT Client_ID, Client_Name, CEmail, Refer_Client_ID FROM Corpor
17	15:50:35	INSERT INTO Dim_Time (Date, Year, Month, Quarter, DayOfWeek, DayOfMonth, FiscalYear) SELECT DISTINCT Start_Date, YEAR(Start_Date) AS Year, M
18	15:50:40	INSERT INTO Dim_Service (ServiceTypeID, ServiceTypeName) VALUES (1, 'Plumbing'), (2, 'Electrical'), (3, 'HVAC')
19	15:50:45	INSERT INTO Lease_Fact (ClientKey, TimeKey, BuildingKey, ApartmentKey, LeaseID, MonthlyRent, SecurityDeposit, StartDate, EndDate) SELECT dc.Clie
20	15:50:54	INSERT INTO ServiceRequest_Fact (ServiceTypeKey, TimeKey, StaffKey, ApartmentKey, StatusKey, BuildingKey, RequestID, TotalRequests, AvgResolutio
21	15:51:01	SELECT 'Lease_Fact' AS TableName, COUNT(*) AS OrphanRecords FROM Lease_Fact WHERE ClientKey NOT IN (SELECT ClientKey FROM Dim_CorpClient)
22	15:51:13	SELECT 'Lease_Fact' AS TableName, COUNT(*) AS OrphanRecords FROM Lease_Fact WHERE ClientKey NOT IN (SELECT ClientKey FROM Dim_CorpClient)

9.0 Analytical Summary Tables

The property management data warehouse is supported by six analytical summary tables that deliver fast, actionable insights across operations and finance. These include staff efficiency (tracking request resolution speed and workload), occupancy trends (measuring lease-based apartment usage across buildings), yearly rental income (estimating revenue per building), unresolved maintenance requests (highlighting pending issues by location), request resolution details (analyzing completion times by month/year), and rental income by bedroom count (assessing profitability by unit size). Designed for precomputed use in dashboards, these summaries enable quick, data-driven decisions in staffing, pricing, leasing, and maintenance, with periodic refreshes and consistent key relationships ensuring reliable performance.

9.1 Summary_StaffEfficiency

Purpose: Measures staff performance in resolving maintenance requests, focusing on speed and workload.
Key Metrics:

- TotalRequestsHandled: Number of completed requests per staff member per year.
- AvgResolutionDays: Average days taken to resolve requests.

SQL Logic:

- Joins Dim_Staff with MAINTENANCE_REQUEST to link staff members to their assigned requests.
- Filters by Status = 'completed' to exclude open/pending requests.
- Groups by staff and year to track annual performance.

Business Value:

- Identifies top-performing staff (e.g., those with the lowest AvgResolutionDays).
- Highlights training needs for staff with slower resolution times.

9.2 Summary_OccupancyTrends

Purpose: Tracks apartment occupancy rates across buildings.
Key Metrics:

- TotalOccupiedApartments: Count of leased apartments.
- TotalApartments: Total available apartments.
- AverageOccupancyRate: Occupied apartments as a percentage of total.

SQL Logic:

- Joins Dim_Building with Dim_Apartment to map apartments to their buildings.
- Uses a LEFT JOIN with LeaseFact to count apartments with active leases.
- Calculates occupancy rate using:

(Occupied / Total) * 100

Business Value:

- Identifies underperforming buildings (low occupancy).
- Supports decisions on rent adjustments or marketing campaigns.

9.3 Summary_YearlyRentalIncome

Purpose: Calculates annual rental income for each building over the last two years.
Key Metrics:

- TotalRentalIncome: Estimated income based on lease duration and monthly rent.

SQL Logic:

- Joins LeaseFact with Dim_Building and Dim_Time to link leases to buildings and years.
- Estimates income by multiplying monthly_rent by the lease duration (in months):

monthly_rent * (DATEDIFF(end_date, start_date) / 30)

- Filters data for the last two years using YEAR(CURDATE()).

Business Value:

- Tracks revenue growth/decline over time.
- Compares building profitability.

9.4 Summary_UnresolvedRequests

Purpose: Identifies unresolved maintenance requests per building.
Key Metric:

- UnresolvedRequests: Count of open/pending requests.

SQL Logic:

- Directly queries MAINTENANCE_REQUEST and joins with Dim_Building.
- Filters requests where Status is not completed.

Business Value:

- Highlights buildings with operational inefficiencies.
- Prioritizes resource allocation for urgent repairs.

9.5 Summary_RequestResolutionDetails

Purpose: Analyzes resolution times for completed requests by month/year.
Key Metrics:

- ResolutionDays: Days taken to resolve a request.

SQL Logic:

- Links MAINTENANCE_REQUEST to Dim_Time to extract year/month from Request_Date.
- Filters by Status = 'completed'.

Business Value:

- Identifies seasonal trends in maintenance demand.
- Monitors SLA compliance (e.g., 95% of requests resolved within 5 days).

9.6 Summary_RentalIncomeByBedrooms

Purpose: Analyzes rental income and pricing by apartment size (bedrooms).
Key Metrics:

- TotalRentalIncome: Total revenue per bedroom category.
- AverageMonthlyRent: Average rent for apartments of a specific size.

SQL Logic:

- Joins LeaseFact with Dim_Apartment to group leases by bedroom count.
- Estimates income using lease duration and monthly rent.

Business Value:

- Determines optimal pricing for apartment sizes (e.g., 3-bedroom units generate the most income).
- Guides future construction or renovations (e.g., build more 2-bedroom units).

9.7 Implementation Notes

9.7.1 Time-Based Approximations:

- Rental income calculations assume 30 days/month for simplicity.
- Adjust if precise calendar-month billing is required.

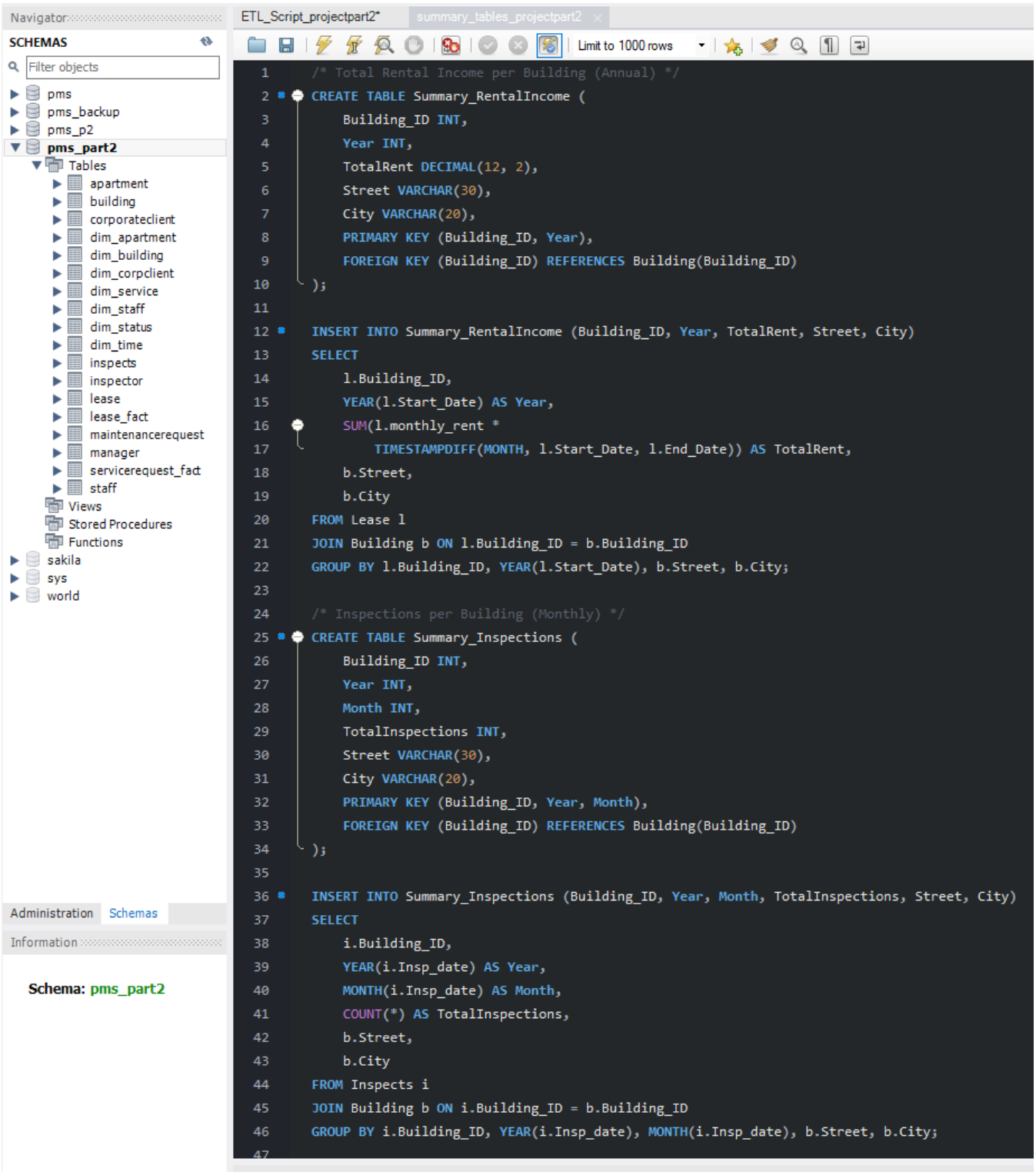
9.7.2 Performance:

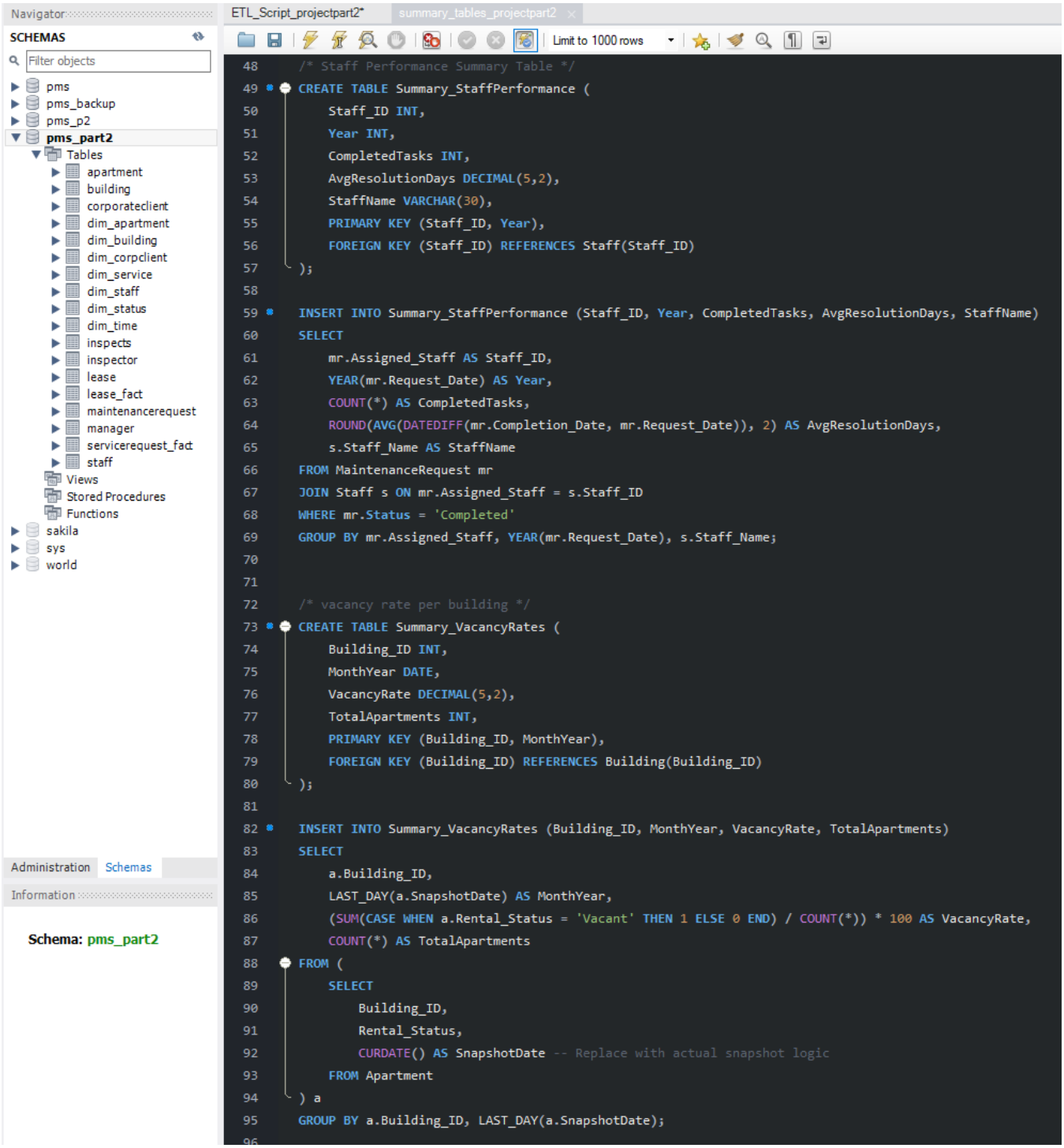
- Summary tables are precompiled to speed up dashboards/reports.
- Refresh them periodically (e.g., nightly) for up-to-date insights.

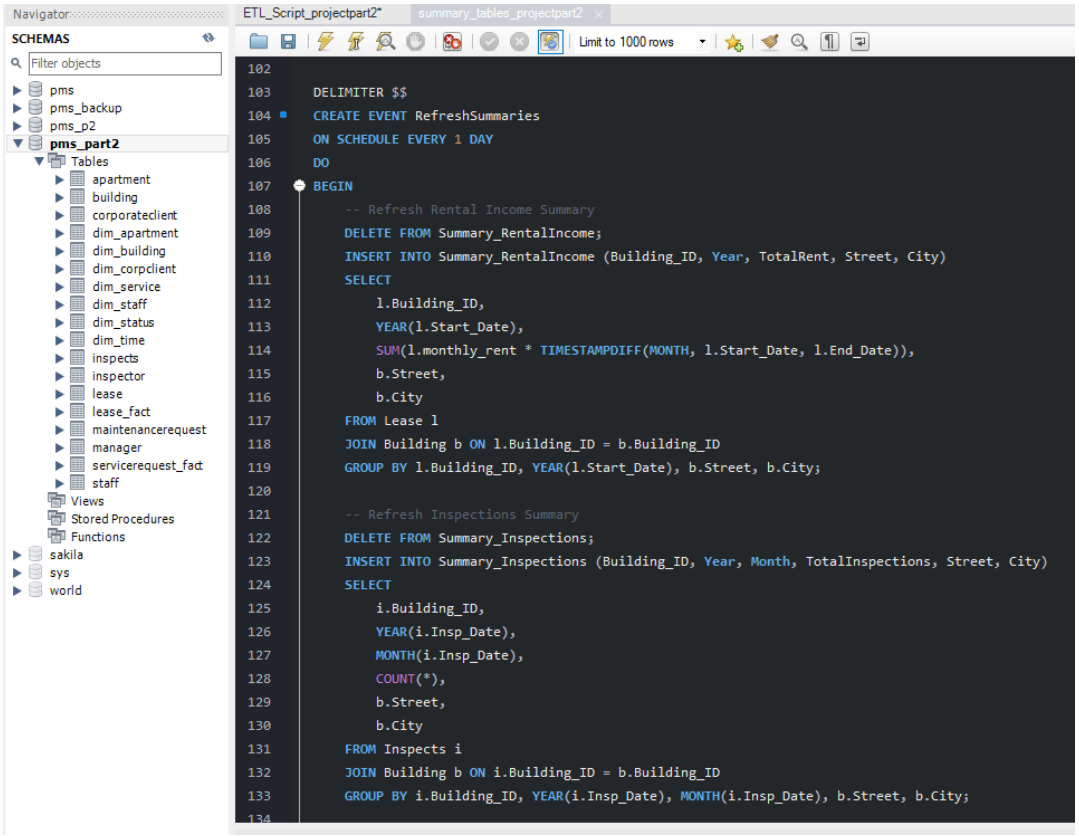
9.7.3 Data Consistency:

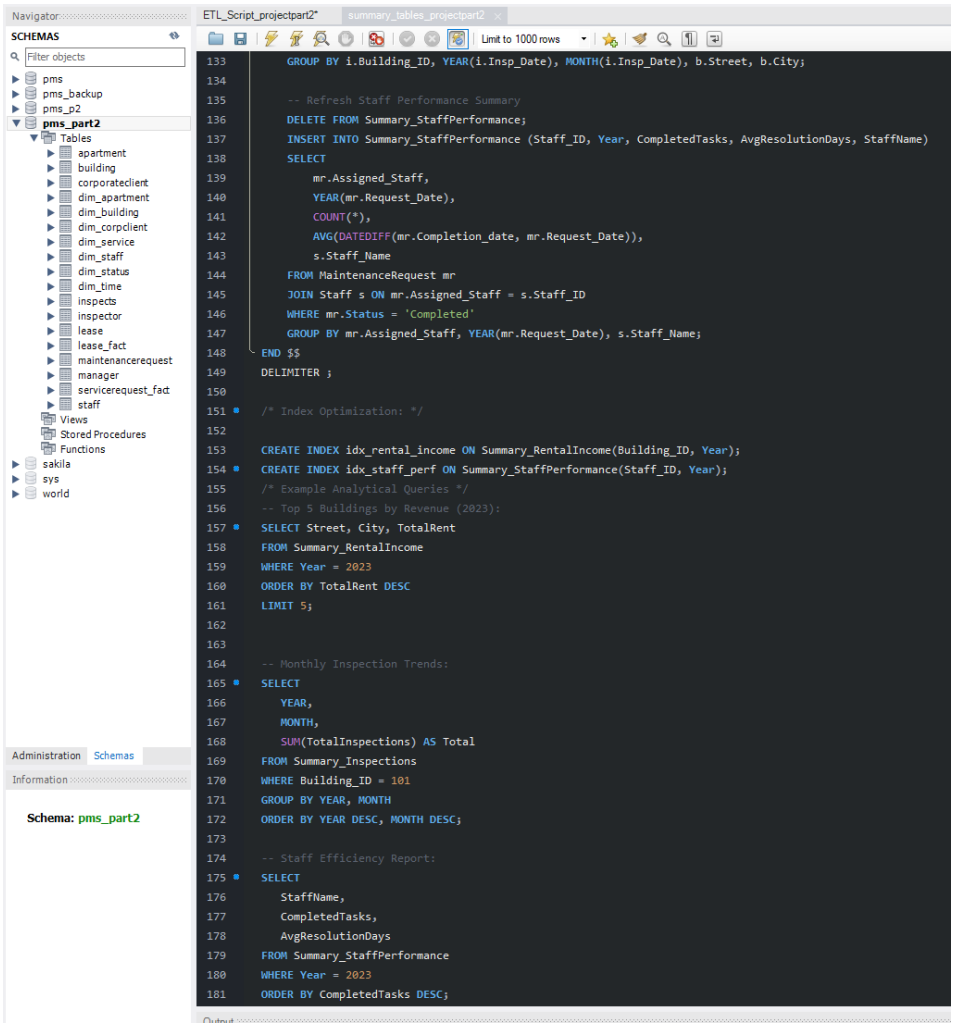
- Ensure foreign keys (e.g., BuildingKey) match between fact/dimension tables.

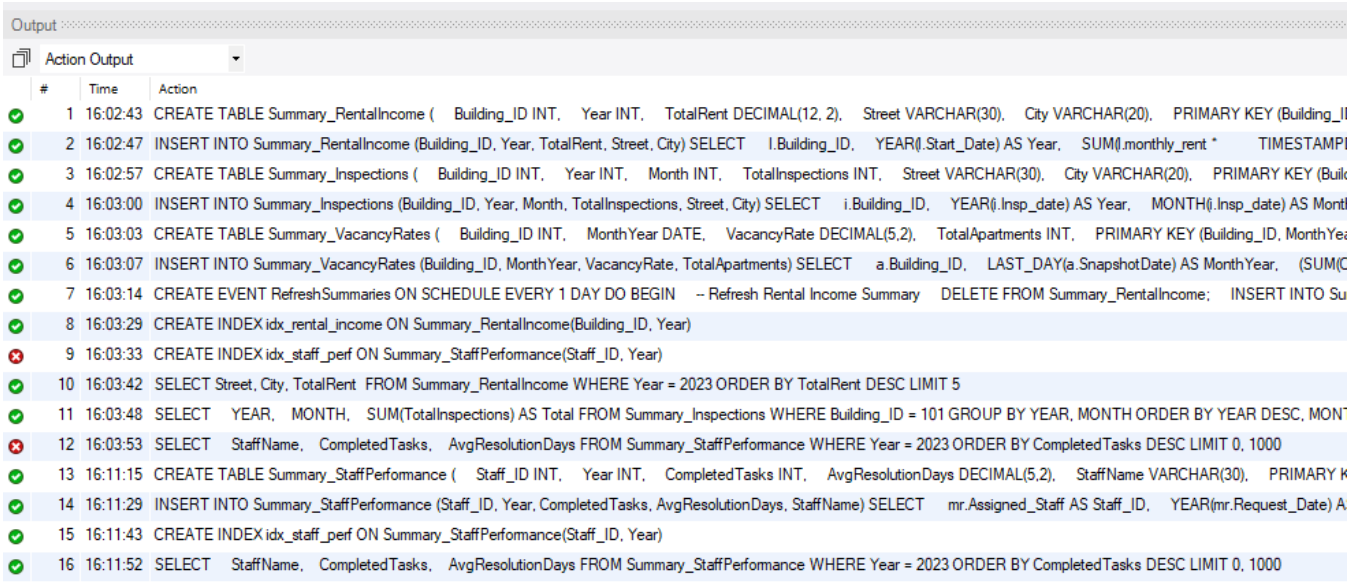
These tables provide actionable insights into **revenue**, **occupancy**, and **operations**, enabling data-driven decisions in property management.


Navigator
SCHEMAS
Filter objects
pms
pms_backup
pms_p2
pms_part2
Tables
apartment
building
corporateclient
dim_apartment
dim_building
dim_corpclient
dim_service
dim_staff
dim_status
dim_time
inspects
inspector
lease
lease_fact
maintenancerequest
manager
servicerequest_fact
staff
Views
Stored Procedures
Functions
sakila
sys
world
Administration
Schemas
Information
Schema: pms_part2
ETL_Script_projectpart2*
summary_tables_projectpart2
Limit to 1000 rows
/* Total Rental Income per Building (Annual) */
CREATE TABLE Summary_RentalIncome (
Building_ID INT,
Year INT,
TotalRent DECIMAL(12, 2),
Street VARCHAR(30),
City VARCHAR(20),
PRIMARY KEY (Building_ID, Year),
FOREIGN KEY (Building_ID) REFERENCES Building(Building_ID)
);
INSERT INTO Summary_RentalIncome (Building_ID, Year, TotalRent, Street, City)
SELECT
l.Building_ID,
YEAR(l.Start_Date) AS Year,
SUM(l.monthly_rent *
TIMESTAMPDIFF(MONTH, l.Start_Date, l.End_Date)) AS TotalRent,
b.Street,
b.City
FROM Lease l
JOIN Building b ON l.Building_ID = b.Building_ID
GROUP BY l.Building_ID, YEAR(l.Start_Date), b.Street, b.City;
/* Inspections per Building (Monthly) */
CREATE TABLE Summary_Inspections (
Building_ID INT,
Year INT,
Month INT,
TotalInspections INT,
Street VARCHAR(30),
City VARCHAR(20),
PRIMARY KEY (Building_ID, Year, Month),
FOREIGN KEY (Building_ID) REFERENCES Building(Building_ID)
);
INSERT INTO Summary_Inspections (Building_ID, Year, Month, TotalInspections, Street, City)
SELECT
i.Building_ID,
YEAR(i.Insp_date) AS Year,
MONTH(i.Insp_date) AS Month,
COUNT(*) AS TotalInspections,
b.Street,
b.City
FROM Inspects i
JOIN Building b ON i.Building_ID = b.Building_ID
GROUP BY i.Building_ID, YEAR(i.Insp_date), MONTH(i.Insp_date), b.Street, b.City;


Navigator
SCHEMAS
Filter objects
pms
pms_backup
pms_p2
pms_part2
Tables
apartment
building
corporateclient
dim_apartment
dim_building
dim_corpclient
dim_service
dim_staff
dim_status
dim_time
inspects
inspector
lease
lease_fact
maintenancerequest
manager
servicerequest_fact
staff
Views
Stored Procedures
Functions
sakila
sys
world
Administration
Schemas
Information
Schema: pms_part2
ETL_Script_projectpart2*
summary_tables_projectpart2
Limit to 1000 rows
/* Staff Performance Summary Table */
CREATE TABLE Summary_StaffPerformance (
Staff_ID INT,
Year INT,
CompletedTasks INT,
AvgResolutionDays DECIMAL(5,2),
StaffName VARCHAR(30),
PRIMARY KEY (Staff_ID, Year),
FOREIGN KEY (Staff_ID) REFERENCES Staff(Staff_ID)
);
INSERT INTO Summary_StaffPerformance (Staff_ID, Year, CompletedTasks, AvgResolutionDays, StaffName)
SELECT
mr.Assigned_Staff AS Staff_ID,
YEAR(mr.Request_Date) AS Year,
COUNT(*) AS CompletedTasks,
ROUND(AVG(DATEDIFF(mr.Completion_Date, mr.Request_Date)), 2) AS AvgResolutionDays,
s.Staff_Name AS StaffName
FROM MaintenanceRequest mr
JOIN Staff s ON mr.Assigned_Staff = s.Staff_ID
WHERE mr.Status = 'Completed'
GROUP BY mr.Assigned_Staff, YEAR(mr.Request_Date), s.Staff_Name;
/* vacancy rate per building */
CREATE TABLE Summary_VacancyRates (
Building_ID INT,
MonthYear DATE,
VacancyRate DECIMAL(5,2),
TotalApartments INT,
PRIMARY KEY (Building_ID, MonthYear),
FOREIGN KEY (Building_ID) REFERENCES Building(Building_ID)
);
INSERT INTO Summary_VacancyRates (Building_ID, MonthYear, VacancyRate, TotalApartments)
SELECT
a.Building_ID,
LAST_DAY(a.SnapshotDate) AS MonthYear,
(SUM(CASE WHEN a.Rental_Status = 'Vacant' THEN 1 ELSE 0 END) / COUNT(*)) * 100 AS VacancyRate,
COUNT(*) AS TotalApartments
FROM (
SELECT
Building_ID,
Rental_Status,
CURDATE() AS SnapshotDate -- Replace with actual snapshot logic
FROM Apartment
) a
GROUP BY a.Building_ID, LAST_DAY(a.SnapshotDate);

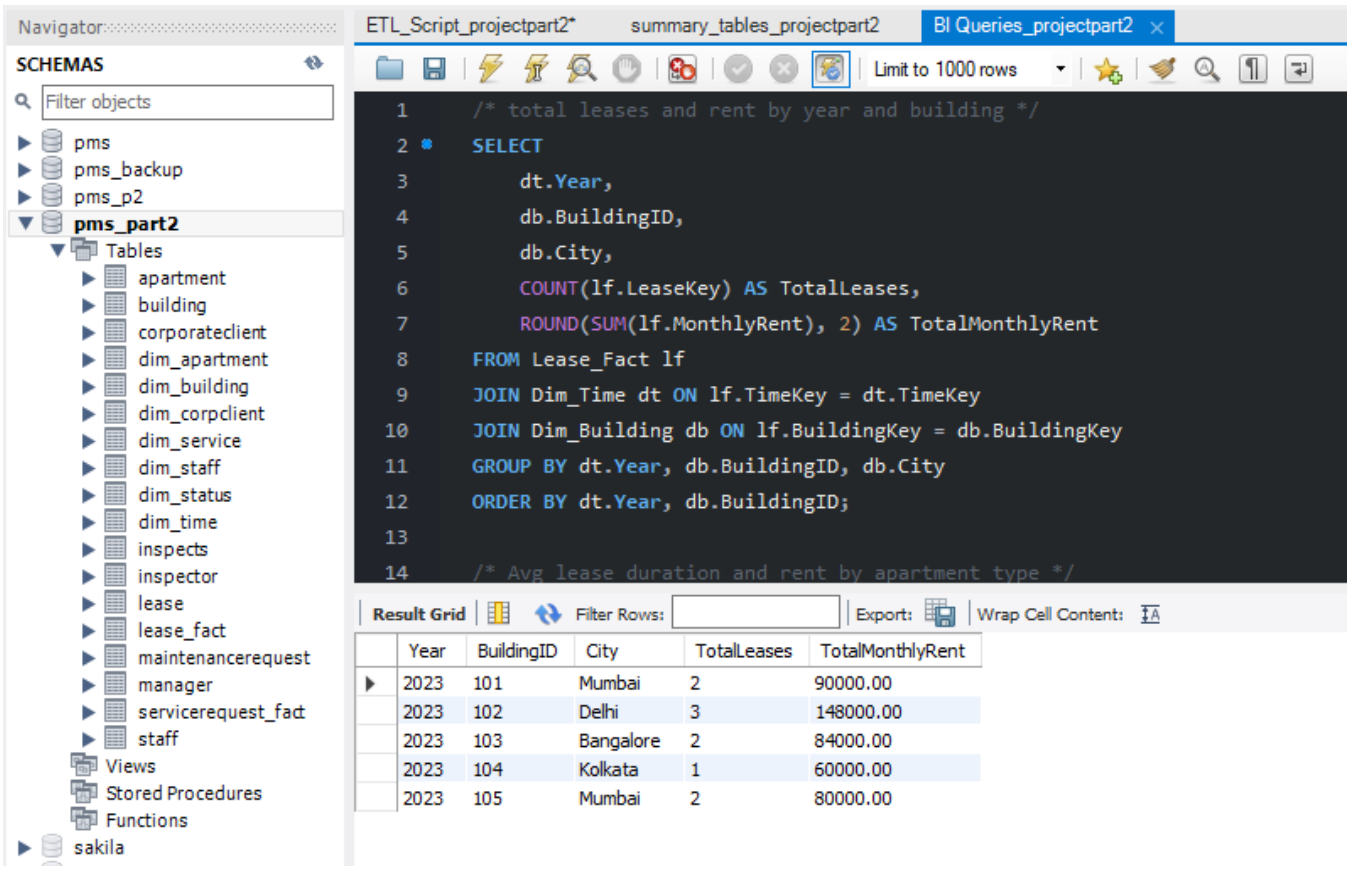

Navigator
SCHEMAS
Filter objects
pms
pms_backup
pms_p2
pms_part2
Tables
apartment
building
corporateclient
dim_apartment
dim_building
dim_corpclient
dim_service
dim_staff
dim_status
dim_time
inspects
inspector
lease
lease_fact
maintenancerequest
manager
servicerequest_fact
staff
Views
Stored Procedures
Functions
sakila
sys
world
ETL_Script_projectpart2*
summary_tables_projectpart2
Limit to 1000 rows
102
103 DELIMITER $$
104 CREATE EVENT RefreshSummaries
105 ON SCHEDULE EVERY 1 DAY
106 DO
107 BEGIN
108 -- Refresh Rental Income Summary
109 DELETE FROM Summary_RentalIncome;
110 INSERT INTO Summary_RentalIncome (Building_ID, Year, TotalRent, Street, City)
111 SELECT
112 l.Building_ID,
113 YEAR(l.Start_Date),
114 SUM(l.monthly_rent * TIMESTAMPDIFF(MONTH, l.Start_Date, l.End_Date)),
115 b.Street,
116 b.City
117 FROM Lease l
118 JOIN Building b ON l.Building_ID = b.Building_ID
119 GROUP BY l.Building_ID, YEAR(l.Start_Date), b.Street, b.City;
120
121 -- Refresh Inspections Summary
122 DELETE FROM Summary_Inspections;
123 INSERT INTO Summary_Inspections (Building_ID, Year, Month, TotalInspections, Street, City)
124 SELECT
125 i.Building_ID,
126 YEAR(i.Insp_Date),
127 MONTH(i.Insp_Date),
128 COUNT(*),
129 b.Street,
130 b.City
131 FROM Inspects i
132 JOIN Building b ON i.Building_ID = b.Building_ID
133 GROUP BY i.Building_ID, YEAR(i.Insp_Date), MONTH(i.Insp_Date), b.Street, b.City;
134


Navigator
SCHEMAS
Filter objects
pms
pms_backup
pms_p2
pms_part2
Tables
apartment
building
corporateclient
dim_apartment
dim_building
dim_corpclient
dim_service
dim_staff
dim_status
dim_time
inspects
inspector
lease
lease_fact
maintenancerequest
manager
servicerequest_fact
staff
Views
Stored Procedures
Functions
sakila
sys
world
Administration
Schemas
Information
Schema: pms_part2
ETL_Script_projectpart2*
summary_tables_projectpart2
Limit to 1000 rows
133 GROUP BY i.Building_ID, YEAR(i.Insp_Date), MONTH(i.Insp_Date), b.Street, b.City;
134
135 -- Refresh Staff Performance Summary
136 DELETE FROM Summary_StaffPerformance;
137 INSERT INTO Summary_StaffPerformance (Staff_ID, Year, CompletedTasks, AvgResolutionDays, StaffName)
138 SELECT
139 mr.Assigned_Staff,
140 YEAR(mr.Request_Date),
141 COUNT(*),
142 AVG(DATEDIFF(mr.Completion_date, mr.Request_Date)),
143 s.Staff_Name
144 FROM MaintenanceRequest mr
145 JOIN Staff s ON mr.Assigned_Staff = s.Staff_ID
146 WHERE mr.Status = 'Completed'
147 GROUP BY mr.Assigned_Staff, YEAR(mr.Request_Date), s.Staff_Name;
148 END $$
149 DELIMITER ;
150
151 /* Index Optimization: */
152
153 CREATE INDEX idx_rental_income ON Summary_RentalIncome(Building_ID, Year);
154 CREATE INDEX idx_staff_perf ON Summary_StaffPerformance(Staff_ID, Year);
155 /* Example Analytical Queries */
156 -- Top 5 Buildings by Revenue (2023):
157 SELECT Street, City, TotalRent
158 FROM Summary_RentalIncome
159 WHERE Year = 2023
160 ORDER BY TotalRent DESC
161 LIMIT 5;
162
163
164 -- Monthly Inspection Trends:
165 SELECT
166 YEAR,
167 MONTH,
168 SUM(TotalInspections) AS Total
169 FROM Summary_Inspections
170 WHERE Building_ID = 101
171 GROUP BY YEAR, MONTH
172 ORDER BY YEAR DESC, MONTH DESC;
173
174 -- Staff Efficiency Report:
175 SELECT
176 StaffName,
177 CompletedTasks,
178 AvgResolutionDays
179 FROM Summary_StaffPerformance
180 WHERE Year = 2023
181 ORDER BY CompletedTasks DESC;



10.0 Advanced BI Reports: 5 complex SQL queries with strategic value and explanations

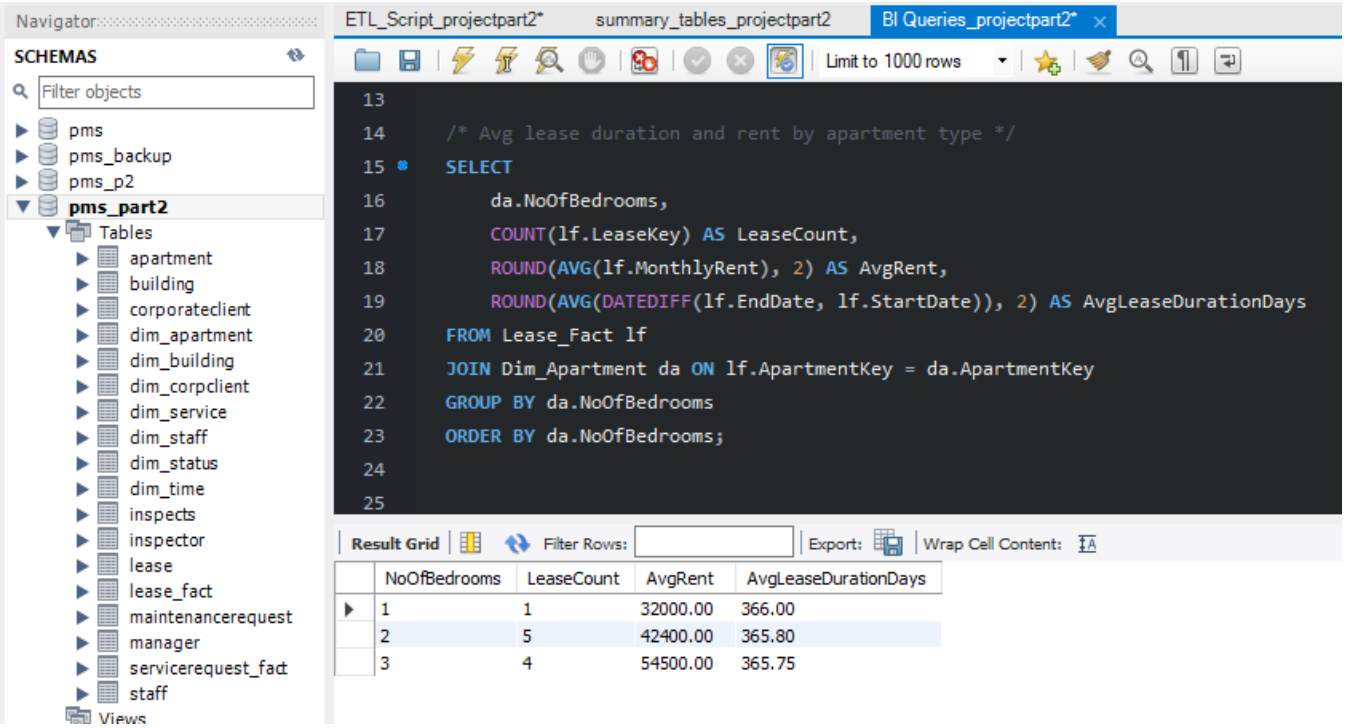
10.1 Total Leases and Rent by Year and Building



What it shows:
Summarizes how many leases were signed and how much rental income was generated per building each year.

Why it matters:
This helps management monitor financial performance and occupancy levels over time, identify top-performing buildings, and make investment or pricing decisions.

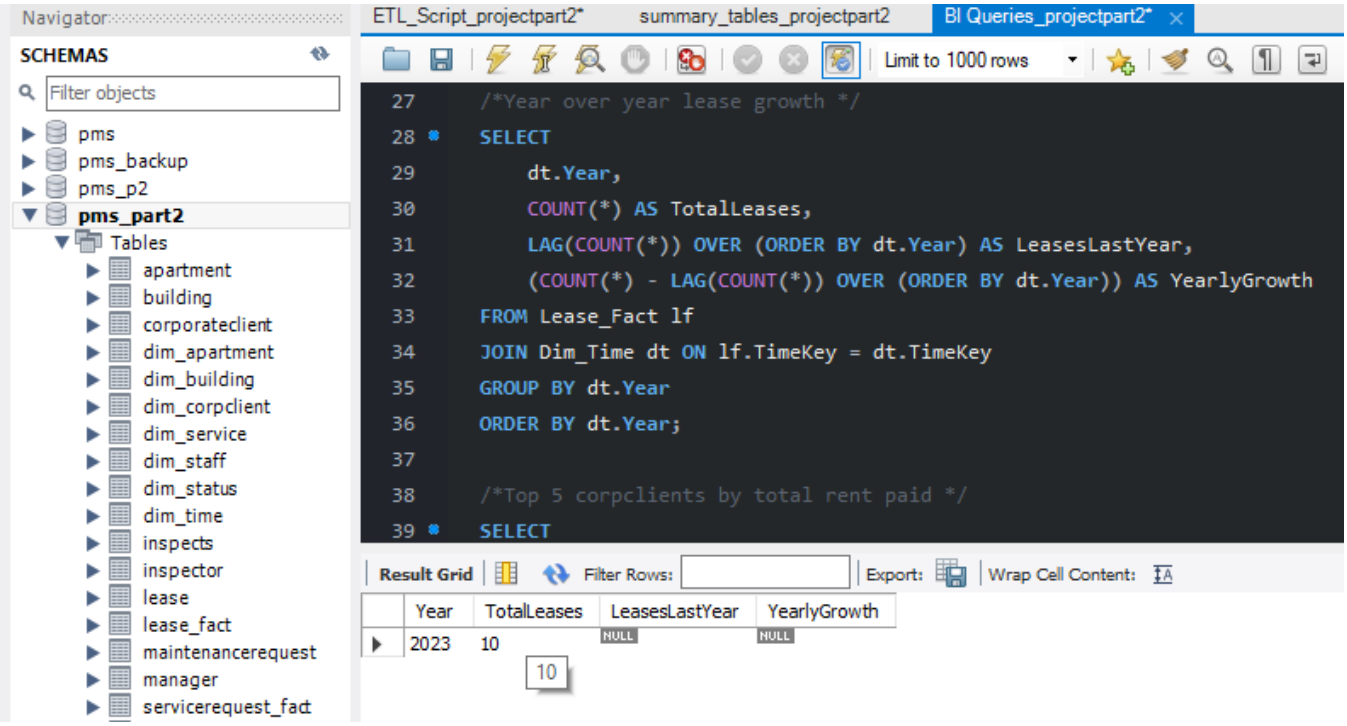
10.2 Average Lease Duration and Rent by Apartment Type



What it shows:
Breaks down lease count, average rent, and lease length by apartment size (e.g., 1-bedroom, 2-bedroom).

Why it matters:
Helps identify which apartment types are most in demand or most profitable, aiding decisions on property design, pricing strategies, and marketing focus.

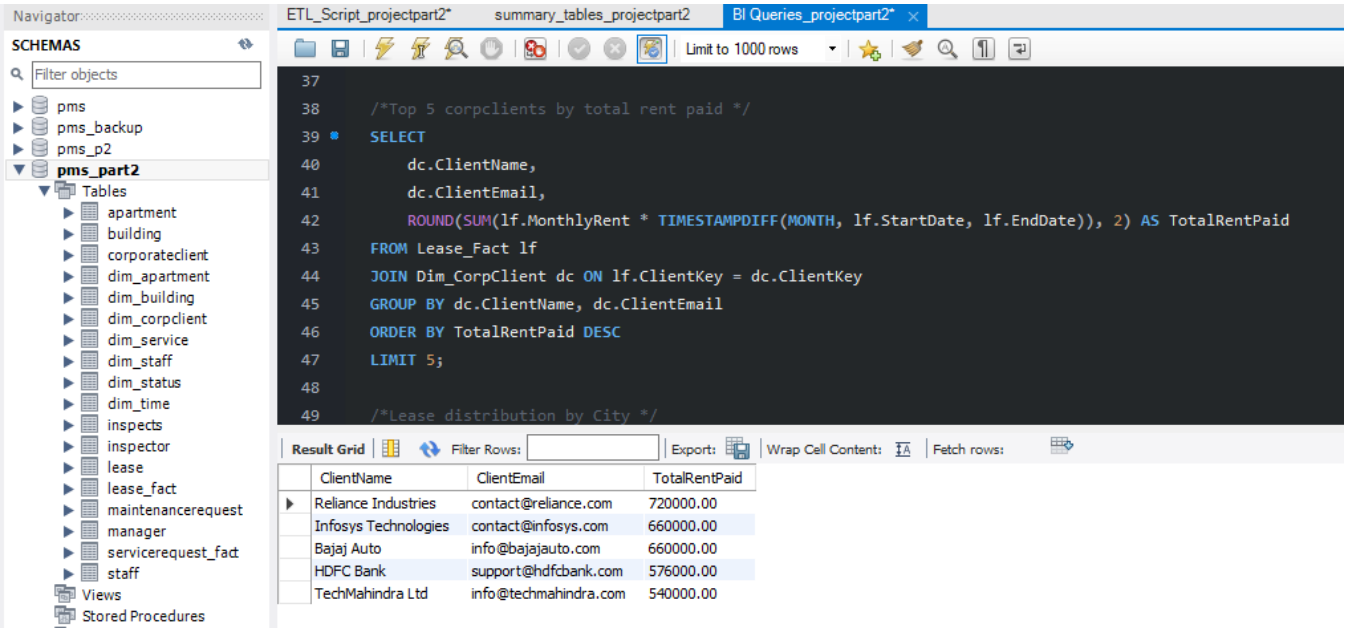
10.3 Year-Over-Year Lease Growth



What it shows:
Tracks how lease volume has changed each year and calculates the difference compared to the previous year.

Why it matters:
Reveals business growth trends or downturns, allowing leadership to react to market demand shifts or operational issues.

10.4 Top 5 Corporate Clients by Total Rent Paid



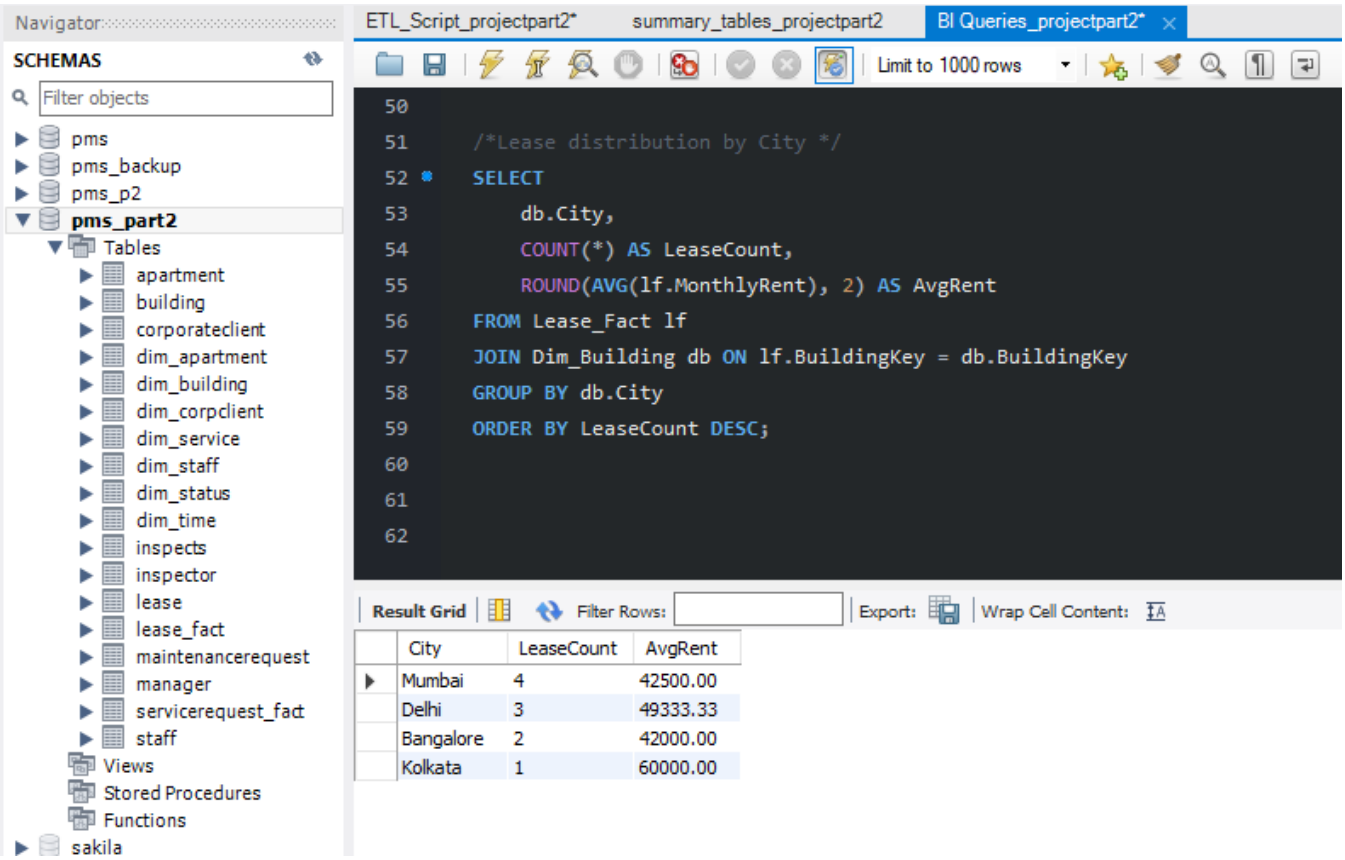
What it shows:

Ranks corporate clients based on the total rent they've paid across their leases.

Why it matters:

Identifies high-value clients for retention efforts, loyalty programs, or personalized service. It can also inform risk assessment if a top client ends their lease.

10.5 Lease Distribution by City



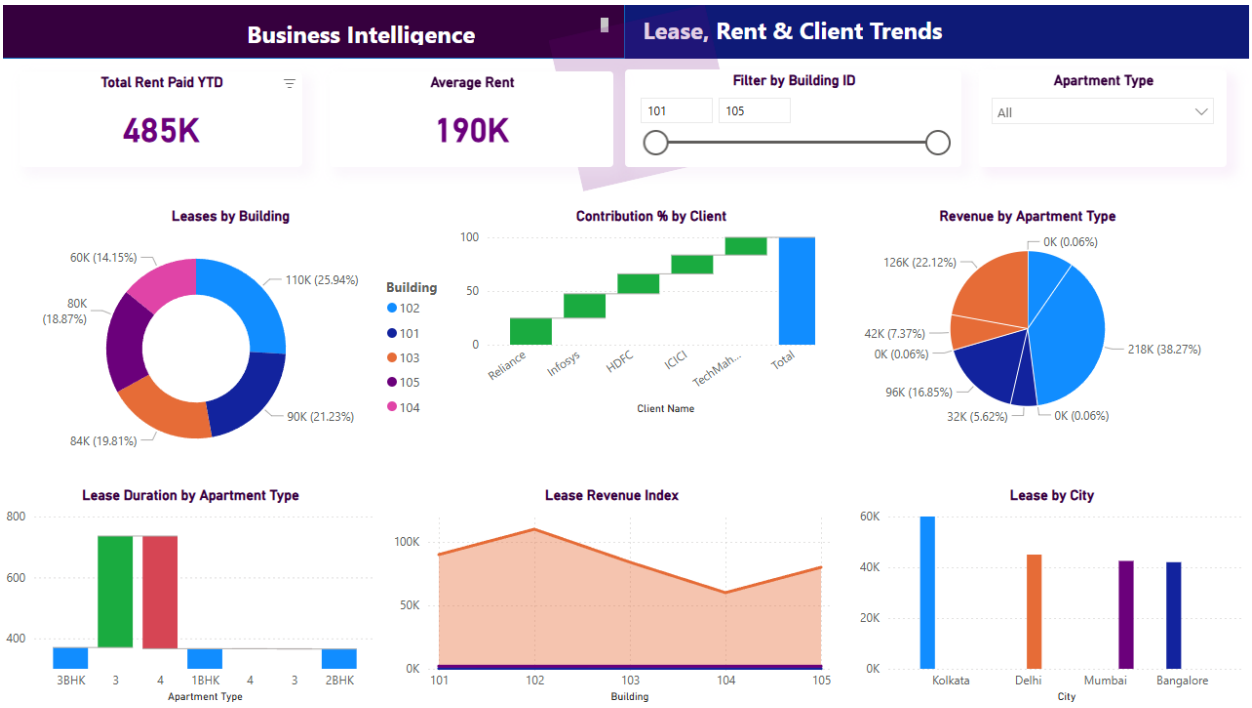
What it shows:

Counts how many leases exist in each city and what the average rent is.

Why it matters:

Enables geographic performance comparison, supports regional pricing strategies, and informs decisions on property expansion or divestment.

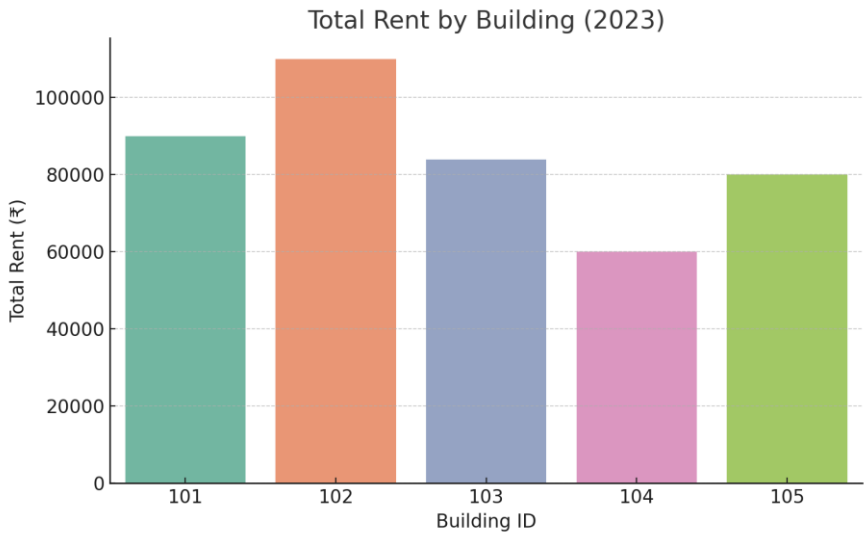
11.0 Bonus Section – Visualization



Built with PowerBI > Dashboard Link: Click here

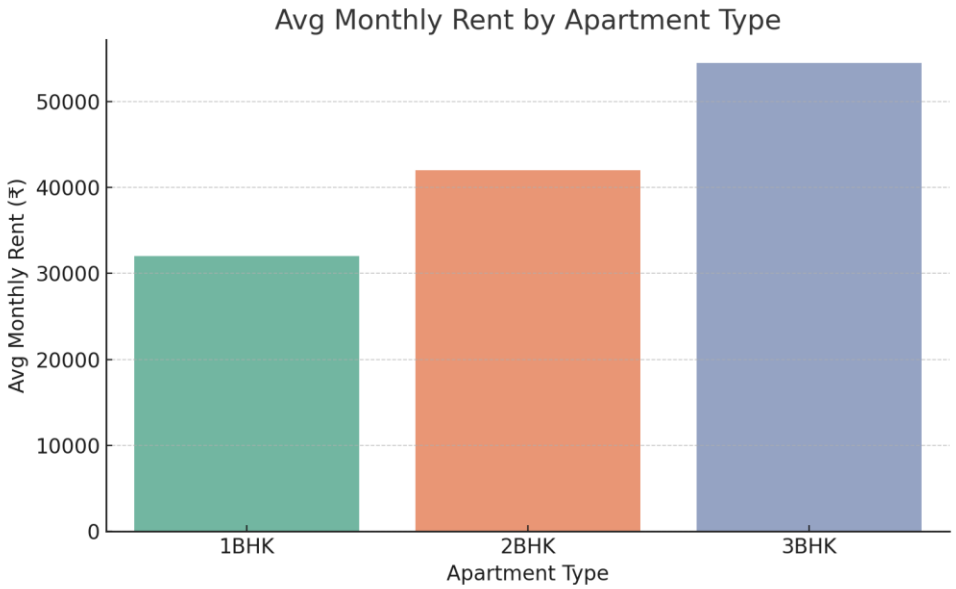
11.1 Total Leases and Rent by Year and Building

This chart displays the total number of leases and corresponding rental income generated by each building in 2023. It helps management identify high-performing buildings and analyze occupancy trends, enabling strategic rent setting and property-level decisions.



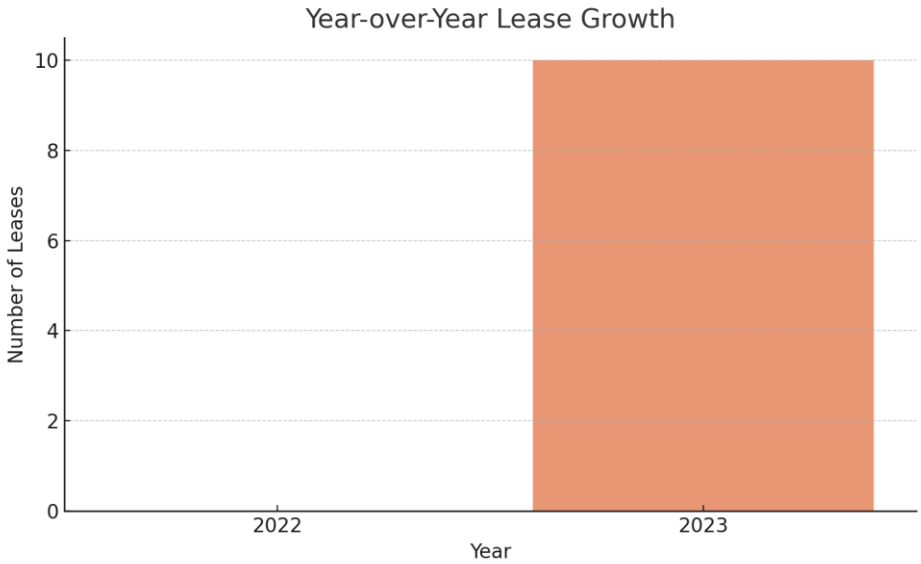
11.2 Average Lease Duration and Rent by Apartment Type

Visualizing the average monthly rent alongside lease durations for different apartment types, this chart identifies which unit sizes (1BHK, 2BHK, 3BHK) are more popular or profitable. It supports layout planning and targeted marketing.



11.3 Year-Over-Year Lease Growth

This visual highlights annual changes in lease volume. For example, a jump from 0 to 10 leases in 2023 signals business growth. Managers can use this to understand market traction and plan expansion or operational improvements.



11.4 Top 5 Corporate Clients by Total Rent Paid

Ranking the highest-paying corporate clients, this chart helps prioritize client engagement, evaluate risk from large accounts, and personalize service offerings. It's essential for revenue assurance and loyalty strategies.

11.5 Lease Distribution by City

This scatter plot compares how many leases were signed in each city and their average rent levels. It informs geographic investment strategies and identifies high-potential or underperforming regions.