



SACRAMENTO STATE  
COLLEGE OF CONTINUING EDUCATION

# MSBA202

## Data Management for Business Analytics

Section - 70

Prepared by – Group 1

**Bindu Nithyananda – 304429021**

**Md Tarbiar Swat - 304454332**

**Hashim Gilani - 304439369**

**Suleman Ahmad - 304467046**

**Group Project - Part 1**

Submission Date  
**April 10, 2025**

Prepared for  
**Beom-Jin Choi**



## 1. 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.

## 2. 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.

## 3. 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)

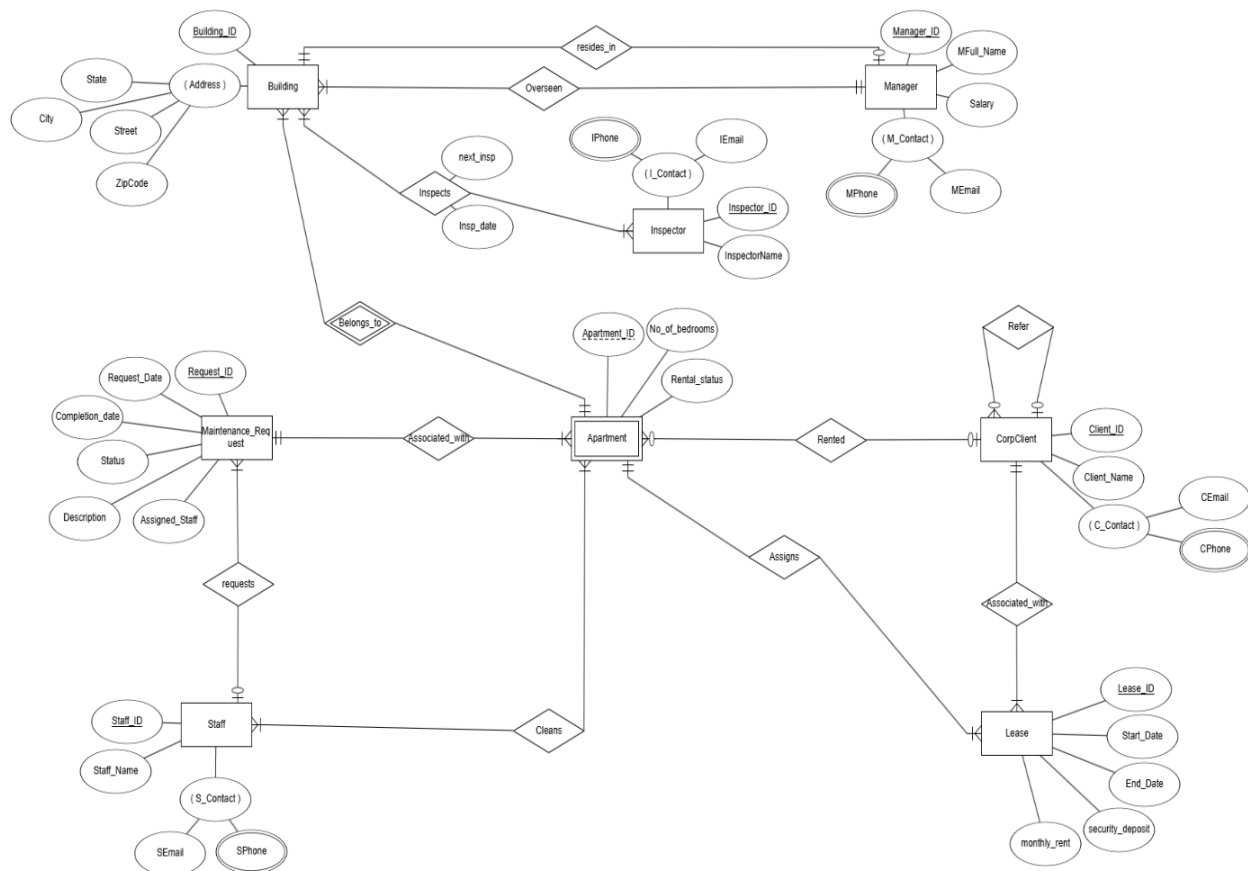
- **Resident\_Feedback** (Feedback\_ID, Feedback\_Date, Category, Description, Status, Apartment\_ID)
- **CorpClient** (Client\_ID, Client\_Name, CEmail,CPhone)
- **Lease** (Lease\_ID, Building\_ID, AptNo, Client\_ID, LeaseStartDate, LeaseEndDate, MonthlyRent, SecurityDeposit)

## 4. ER Diagram

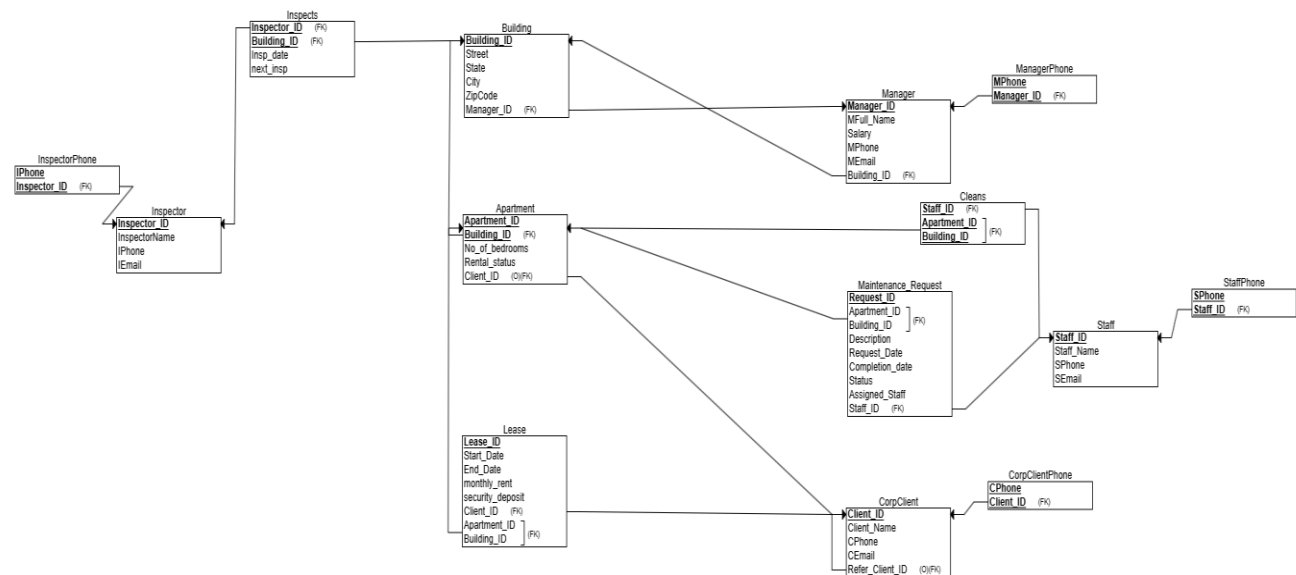
The ER Diagram visually represents the conceptual design of the property management system. It illustrates all major entities—such as Building, Manager, Apartment, MaintenanceRequest, Lease, CorpClient, Inspector, Staff, and related associative entities like Overseen, Inspection, and Cleans—and the relationships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

## 5. (20 Points) Create an ER diagram and the relational schema for the database

### 5.1 Entity-Relationship Diagram: PROPERTY MANAGEMENT SYSTEM



## 5.2 Relational-Schema: PROPERTY MANAGEMENT SYSTEM



6. (15 points) Create the data dictionary (an example is shown below) that documents metadata about tables and related constraints.

### 6.1 Data-Dictionary: PROPERTY MANAGEMENT SYSTEM

<b>Entity:</b>	<b>MANAGER TABLE</b>	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	-
EMail	Manager's email address	VARCHAR	50 alpha numeric characters	-	YES	YES	-
Building_ID	Building where the manager resides	INT	10 digits	-	YES	NO	FK
<b>MANAGER PHONE TABLE</b>							
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
<b>Entity:</b>	<b>BUILDING TABLE</b>	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
<b>Entity:</b>	<b>APARTMENT TABLE</b>	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
<b>Entity:</b>	<b>CORPCLIENT TABLE</b>	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 malINtenance 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	MalINtenance, 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	-

## 7. (10 points) Populate the database with sample data using SQL DDL codes (at least 10 rows).

### 7.1 SQL Database creation and table population

The screenshot displays the SQL Enterprise Manager interface. On the left, the 'SCHEMAS' pane shows a tree view with 'pms' (Tables, Views, Stored Procedures, Functions), 'sakila', 'sys', and 'world'. The 'Administration' tab is selected, and the 'Schemas' sub-tab is active. The main area shows the 'CreateDB\_Table\_Query\_Group...' window with the following SQL code:

```
1  /* Enable foreign key checks */
2  • create database PMS;
3  • use PMS;
4  • SET FOREIGN_KEY_CHECKS = 0;
5
6  /* Manager and Building Tables */
7  • CREATE TABLE MANAGER (
8      Manager_ID INT PRIMARY KEY,
9      MFull_Name CHAR(25) NOT NULL,
10     Salary INT NOT NULL,
11     M_Contact VARCHAR(15) NOT NULL,
12     MEmail VARCHAR(50) NOT NULL UNIQUE,
13     Building_ID INT NOT NULL,
14     FOREIGN KEY (Building_ID) REFERENCES BUILDING(Building_ID)
15 );
16
17 • CREATE TABLE BUILDING (
18     Building_ID INT PRIMARY KEY,
19     Street VARCHAR(30),
20     City VARCHAR(20),
21     State VARCHAR(20),
22     ZipCode INT,
23     Manager_ID INT NOT NULL UNIQUE,
24     FOREIGN KEY (Manager_ID) REFERENCES MANAGER(Manager_ID)
25 );
26
27 /* Manager Phone Table */
28 • CREATE TABLE MANAGER_PHONE (
29     Mphone VARCHAR(15) NOT NULL,
30     Manager_ID INT NOT NULL,
31     FOREIGN KEY (Manager_ID) REFERENCES MANAGER(Manager_ID)
32 );
33
34 /* Apartment Table */
35 • CREATE TABLE APARTMENT (
36     Apartment_ID INT,
37     Building_ID INT,
38     No_of_Bedrooms INT NOT NULL CHECK (No_of_Bedrooms BETWEEN 1 AND 5),
39     Rental_status VARCHAR(10) DEFAULT 'Vacant' NOT NULL,
40     Client_ID INT,
41     PRIMARY KEY (Apartment_ID, Building_ID),
42     FOREIGN KEY (Building_ID) REFERENCES BUILDING(Building_ID),
43     FOREIGN KEY (Client_ID) REFERENCES CORPCLIENT(Client_ID)
44 );
45
```

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

```

45
46  /* Corporate Client Tables */
47  CREATE TABLE CORPCLIENT (
48      Client_ID INT PRIMARY KEY,
49      Client_Name VARCHAR(30) NOT NULL,
50      CEmail VARCHAR(20) NOT NULL UNIQUE,
51      Refer_Client_ID INT,
52      FOREIGN KEY (Refer_Client_ID) REFERENCES CORPCLIENT(Client_ID)
53  );
54
55  CREATE TABLE CORPCLIENT_PHONE (
56      CPhone VARCHAR(15) NOT NULL,
57      Client_ID INT NOT NULL,
58      FOREIGN KEY (Client_ID) REFERENCES CORPCLIENT(Client_ID)
59  );
60
61  /* Lease Table */
62  CREATE TABLE LEASE (
63      Lease_ID INT(10) PRIMARY KEY,
64      Apartment_ID INT(10) NOT NULL,
65      Building_ID INT(10) NOT NULL,
66      Start_Date DATE NOT NULL,
67      End_Date DATE NOT NULL,
68      monthly_rent DECIMAL(10,2) NOT NULL,
69      security_deposit DECIMAL(10,2) NOT NULL,
70      Client_ID INT(10) NOT NULL,
71      FOREIGN KEY (Apartment_ID, Building_ID) REFERENCES APARTMENT(Apartment_ID, Building_ID),
72      FOREIGN KEY (Client_ID) REFERENCES CORPCLIENT(Client_ID)
73  );
74
75  -- Staff Tables
76  CREATE TABLE STAFF (
77      Staff_ID INT PRIMARY KEY,
78      Staff_Name VARCHAR(30) NOT NULL,
79      SEmail VARCHAR(20) NOT NULL UNIQUE
80  );
81
82  CREATE TABLE STAFF_PHONE (
83      SPhone VARCHAR(15) NOT NULL,
84      Staff_ID INT NOT NULL,
85      FOREIGN KEY (Staff_ID) REFERENCES STAFF(Staff_ID)
86  );

```

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

```

133
134  /* Resident Feedback Table */
135  CREATE TABLE RESIDENT_FEEDBACK (
136      FeedbackID INT AUTO_INCREMENT PRIMARY KEY,
137      Building_ID INT NOT NULL,
138      Apartment_ID INT NOT NULL,
139      FeedbackDate DATE NOT NULL,
140      Category ENUM('Maintenance', 'Noise', 'Facility') NOT NULL,
141      Description TEXT NOT NULL,
142      Status VARCHAR(10) NOT NULL CHECK (Status IN ('Open', 'Resolved')),
143      FOREIGN KEY (Apartment_ID, Building_ID) REFERENCES APARTMENT(Apartment_ID, Building_ID)
144  );
145
146  /* Utility Billing Table */
147  CREATE TABLE UTILITY_BILLING (
148      UtilityID INT AUTO_INCREMENT PRIMARY KEY,
149      Building_ID INT NOT NULL,
150      Apartment_ID INT NOT NULL,
151      UtilityType ENUM('Electricity', 'Water', 'Gas') NOT NULL,
152      MonthlyCost DECIMAL(10,2) NOT NULL CHECK (MonthlyCost > 0),
153      DueDate DATE NOT NULL,
154      FOREIGN KEY (Apartment_ID, Building_ID) REFERENCES APARTMENT(Apartment_ID, Building_ID)
155  );

```

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  );
```



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

```

1
2 SET FOREIGN_KEY_CHECKS = 0;
3
4 /* Insert Managers with temporary Building_IDs */
5 INSERT INTO MANAGER (Manager_ID, MFull_Name, Salary, M_Contact, MEmail, Building_ID) VALUES
6 (103, 'Saanvi Desai', 68000, '555-0103', 'sdesai@gmail.com', 3),
7 (104, 'Arjun Kumar', 72000, '555-0203', 'akumar@gmail.com', 4),
8 (105, 'Ishaan Rao', 70000, '555-0301', 'irao@gmail.com', 5),
9 (106, 'Meera Nair', 75000, '555-0401', 'mnair@gmail.com', 6),
10 (107, 'Reyansh Singh', 69000, '555-0501', 'rsingh@gmail.com', 7),
11 (108, 'Aarohi Patel', 71000, '555-0601', 'apatel@gmail.com', 8),
12 (109, 'Krishna Verma', 73000, '555-0701', 'kverma@gmail.com', 9),
13 (110, 'Navya Reddy', 72000, '555-0801', 'nreddy@gmail.com', 10),
14 (111, 'Sai Prakash', 74000, '555-0901', 'sprakash@gmail.com', 11),
15 (112, 'Diya Sharma', 76000, '555-1001', 'dsharma@gmail.com', 12);
16
17 /* Insert Buildings with actual Manager_IDs */
18 INSERT INTO BUILDING (Building_ID, Street, City, State, ZipCode, Manager_ID) VALUES
19 (3, '789 Pine Rd', 'Greenwich', 'CT', 06830, 103),
20 (4, '101 Birch Blvd', 'Fairfield', 'CT', 06824, 104),
21 (5, '202 Cedar Ln', 'Stamford', 'CT', 06901, 105),
22 (6, '303 Elm St', 'Norwalk', 'CT', 06851, 106),
23 (7, '404 Maple Dr', 'Danbury', 'CT', 06810, 107),
24 (8, '505 Oak Ct', 'Bridgeport', 'CT', 06604, 108),
25 (9, '606 Pine Ave', 'New Haven', 'CT', 06510, 109),
26 (10, '707 Birch Rd', 'Hartford', 'CT', 06103, 110),
27 (11, '808 Cedar Blvd', 'Waterbury', 'CT', 06702, 111),
28 (12, '909 Elm Ave', 'New London', 'CT', 06320, 112);
29

```

Output

Action Output

#	Time	Action
✓ 24	08:48:54	INSERT INTO BUILDING (Building_ID, Street, City, State, ZipCode, Manager_ID) VALUES (3, '789 Pine Rd', 'Greenwich', 'CT', 06830, 103)
✓ 25	08:48:54	UPDATE MANAGER SET Building_ID = 3 WHERE Manager_ID = 103
✓ 26	08:48:54	UPDATE MANAGER SET Building_ID = 4 WHERE Manager_ID = 104
✓ 27	08:48:54	UPDATE MANAGER SET Building_ID = 5 WHERE Manager_ID = 105
✓ 28	08:48:54	UPDATE MANAGER SET Building_ID = 6 WHERE Manager_ID = 106
✓ 29	08:48:54	UPDATE MANAGER SET Building_ID = 7 WHERE Manager_ID = 107
✓ 30	08:48:54	UPDATE MANAGER SET Building_ID = 8 WHERE Manager_ID = 108
✓ 31	08:48:54	UPDATE MANAGER SET Building_ID = 9 WHERE Manager_ID = 109
✓ 32	08:48:54	UPDATE MANAGER SET Building_ID = 10 WHERE Manager_ID = 110
✓ 33	08:48:54	UPDATE MANAGER SET Building_ID = 11 WHERE Manager_ID = 111
✓ 34	08:48:54	UPDATE MANAGER SET Building_ID = 12 WHERE Manager_ID = 112
✓ 35	08:48:54	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)
✓ 36	08:48:54	INSERT INTO CORPCLIENT (Client_ID, Client_Name, CEmail, Refer_Client_ID) VALUES (504, 'Innovative Solutions Ltd', 'innovate@gmail.com', 103)
✓ 37	08:48:54	INSERT INTO CORPCLIENT_PHONE (CPhone, Client_ID) VALUES ('555-1101', 504), ('555-1102', 504), ('555-2001', 505), ('555-2002', 506), ('555-2003', 507), ('555-2004', 508), ('555-2005', 509), ('555-2006', 510), ('555-2007', 511), ('555-2008', 512)
✓ 38	08:48:54	INSERT INTO APARTMENT (Apartment_ID, Building_ID, No_of_Bedrooms, Rental_status, Client_ID) VALUES (203, 3, 2, 'Vacant', NULL), (204, 4, 3, 'Occupied', 103), (205, 5, 2, 'Vacant', NULL), (206, 6, 3, 'Occupied', 104), (207, 7, 2, 'Vacant', NULL), (208, 8, 3, 'Occupied', 105), (209, 9, 2, 'Vacant', NULL), (210, 10, 3, 'Occupied', 106), (211, 11, 2, 'Vacant', NULL), (212, 12, 3, 'Occupied', 107)
✓ 39	08:48:54	INSERT INTO LEASE (Lease_ID, Apartment_ID, Building_ID, Start_Date, End_Date, monthly_rent, security_deposit, Client_ID) VALUES (1001, 203, 3, '2023-05-15', '2024-05-15', 1200, 500, 103), (1002, 204, 4, '2023-06-01', '2024-06-01', 1500, 600, 104), (1003, 205, 5, '2023-06-15', '2024-06-15', 1000, 400, 105), (1004, 206, 6, '2023-07-01', '2024-07-01', 1800, 700, 106), (1005, 207, 7, '2023-07-15', '2024-07-15', 1100, 450, 107), (1006, 208, 8, '2023-08-01', '2024-08-01', 1600, 650, 108), (1007, 209, 9, '2023-08-15', '2024-08-15', 1300, 550, 109), (1008, 210, 10, '2023-09-01', '2024-09-01', 1400, 500, 110), (1009, 211, 11, '2023-09-15', '2024-09-15', 1700, 600, 111), (1010, 212, 12, '2023-10-01', '2024-10-01', 1200, 400, 112)
✓ 40	08:48:54	INSERT INTO STAFF (Staff_ID, Staff_Name, SEmail) VALUES (203, 'Ravi Mehta', 'rmehta@gmail.com'), (204, 'Priya Sharma', 'psharma@gmail.com'), (205, 'Amit Singh', 'asingh@gmail.com'), (206, 'Neha Gupta', 'ngupta@gmail.com'), (207, 'Vikram Reddy', 'vreddy@gmail.com'), (208, 'Sneha Nair', 'snair@gmail.com'), (209, 'Adarsh Kumar', 'adkumar@gmail.com'), (210, 'Ishika Verma', 'iverma@gmail.com'), (211, 'Rishabh Patel', 'rpatel@gmail.com'), (212, 'Ananya Sharma', 'anasharma@gmail.com')
✓ 41	08:48:54	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)
✓ 42	08:48:54	INSERT INTO INSPECTOR (Inspector_ID, InspectorName, IEmail) VALUES (303, 'Ritika Das', 'rdas@exam.com'), (304, 'Kabir Malhotra', 'kmalhotra@exam.com'), (305, 'Ananya Singh', 'asingh@exam.com'), (306, 'Vishal Kumar', 'vkumar@exam.com'), (307, 'Ishika Verma', 'iverma@exam.com'), (308, 'Rishabh Patel', 'rpatel@exam.com'), (309, 'Ananya Sharma', 'anasharma@exam.com'), (310, 'Ritika Das', 'rdas@exam.com'), (311, 'Kabir Malhotra', 'kmalhotra@exam.com'), (312, 'Ananya Singh', 'asingh@exam.com')
✓ 43	08:48:54	INSERT INTO INSPECTOR_PHONE (IPhone, Inspector_ID) VALUES ('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)
✓ 44	08:48:54	INSERT INTO INSPECTS (Inspector_ID, Building_ID, Insp_date, next_insp) VALUES (303, 3, '2023-05-15', '2024-05-15'), (304, 4, '2023-06-01', '2024-06-01'), (305, 5, '2023-06-15', '2024-06-15'), (306, 6, '2023-07-01', '2024-07-01'), (307, 7, '2023-07-15', '2024-07-15'), (308, 8, '2023-08-01', '2024-08-01'), (309, 9, '2023-08-15', '2024-08-15'), (310, 10, '2023-09-01', '2024-09-01'), (311, 11, '2023-09-15', '2024-09-15'), (312, 12, '2023-10-01', '2024-10-01')
✓ 45	08:48:54	INSERT INTO CLEANS (Staff_ID, Apartment_ID, Building_ID) VALUES (203, 203, 3), (204, 204, 3), (205, 205, 4), (206, 206, 4), (207, 207, 4), (208, 208, 5), (209, 209, 5), (210, 210, 5), (211, 211, 6), (212, 212, 6)
✓ 46	08:48:54	INSERT INTO MAINTENANCE_REQUEST (Request_ID, Building_ID, Apartment_ID, Request_Date, Completion_date, Status, Assigned_Staff_ID) VALUES (1001, 3, 203, '2023-05-15', '2024-05-15', 'Open', 203), (1002, 4, 204, '2023-06-01', '2024-06-01', 'Open', 204), (1003, 5, 205, '2023-06-15', '2024-06-15', 'Open', 205), (1004, 6, 206, '2023-07-01', '2024-07-01', 'Open', 206), (1005, 7, 207, '2023-07-15', '2024-07-15', 'Open', 207), (1006, 8, 208, '2023-08-01', '2024-08-01', 'Open', 208), (1007, 9, 209, '2023-08-15', '2024-08-15', 'Open', 209), (1008, 10, 210, '2023-09-01', '2024-09-01', 'Open', 210), (1009, 11, 211, '2023-09-15', '2024-09-15', 'Open', 211), (1010, 12, 212, '2023-10-01', '2024-10-01', 'Open', 212)

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\_ x

Limit to 1000 rows

29

30 /\* Update Managers to reference the correct Building\_IDs \*/

31 UPDATE MANAGER SET Building\_ID = 3 WHERE Manager\_ID = 103;

32 UPDATE MANAGER SET Building\_ID = 4 WHERE Manager\_ID = 104;

33 UPDATE MANAGER SET Building\_ID = 5 WHERE Manager\_ID = 105;

34 UPDATE MANAGER SET Building\_ID = 6 WHERE Manager\_ID = 106;

35 UPDATE MANAGER SET Building\_ID = 7 WHERE Manager\_ID = 107;

36 UPDATE MANAGER SET Building\_ID = 8 WHERE Manager\_ID = 108;

37 UPDATE MANAGER SET Building\_ID = 9 WHERE Manager\_ID = 109;

38 UPDATE MANAGER SET Building\_ID = 10 WHERE Manager\_ID = 110;

39 UPDATE MANAGER SET Building\_ID = 11 WHERE Manager\_ID = 111;

40 UPDATE MANAGER SET Building\_ID = 12 WHERE Manager\_ID = 112;

41

42 /\* Insert Manager Phone Numbers \*/

43 INSERT INTO MANAGER\_PHONE (Mphone, Manager\_ID) VALUES

44 ('555-0103', 103),

45 ('555-0104', 104),

46 ('555-0301', 105),

47 ('555-0401', 106),

48 ('555-0501', 107),

49 ('555-0601', 108),

50 ('555-0701', 109),

51 ('555-0801', 110),

52 ('555-0901', 111),

53 ('555-1001', 112);

54

55 /\* Insert Corporate Clients with referrals \*/

56 INSERT INTO CORPCLIENT (Client\_ID, Client\_Name, CEmail, Refer\_Client\_ID) VALUES

57 (504, 'Innovative Solutions Ltd', 'innovate@gmail.com', NULL),

58 (505, 'Eco Friendly Enterprises', 'eco@gmail.com', 504),

59 (506, 'Smart Tech Corp', 'smart@gmail.com', 505),

60 (507, 'Global Logistics Inc', 'log@gmail.com', 506),

61 (508, 'Creative Designs Studio', 'desi@gmail.com', 507),

62 (509, 'Health Plus Clinic', 'health@gmail.com', 508),

63 (510, 'EduTech Solutions', 'edutech@gmail.com', 509),

64 (511, 'Green Energy Co', 'green@gmail.com', 510),

65 (512, 'Fast Delivery Services', 'del@gmail.com', 511),

66 (513, 'Bright Future School', 'sc@gmail.com', 512);

67

68 /\* Insert Corporate Client Phone Numbers \*/

69 INSERT INTO CORPCLIENT\_PHONE (CPhone, Client\_ID) VALUES

70 ('555-1101', 504),

71 ('555-1102', 504),

72 ('555-2001', 505),

73 ('555-2002', 505),

74 ('555-3001', 506),

75 ('555-3002', 506),

76 ('555-4001', 507),

77 ('555-4002', 507),

78 ('555-5001', 508),

79 ('555-5002', 508);

80

Navigator

SCHEMAS

Filter objects

pmis

Tables

Views

Stored Procedures

Functions

sakila

sys

world

Administration

Schemas

Information

No object selected

InsertValues\_Query\_GroupProj...

Limit to 1000 rows

```
81  /* Insert Apartments */
82  INSERT INTO APARTMENT (Apartment_ID, Building_ID, No_of_Bedrooms, Rental_status, Client_ID) VALUES
83  (203, 3, 2, 'Vacant', NULL),
84  (204, 3, 3, 'Occupied', 504),
85  (205, 4, 1, 'Vacant', NULL),
86  (206, 4, 2, 'Occupied', 505),
87  (207, 5, 2, 'Vacant', NULL),
88  (208, 5, 3, 'Occupied', 506),
89  (209, 6, 1, 'Vacant', NULL),
90  (210, 6, 2, 'Occupied', 507),
91  (211, 7, 2, 'Vacant', NULL),
92  (212, 7, 3, 'Occupied', 508);
93
94  /* Insert Leases */
95  INSERT INTO LEASE (Lease_ID, Apartment_ID, Building_ID, Start_Date, End_Date, monthly_rent, security_deposit, Client_ID) VALUES
96  (1003, 203, 3, '2023-05-01', '2024-04-30', 2200.00, 3300.00, 504),
97  (1004, 204, 3, '2023-06-15', '2024-06-14', 2400.00, 3600.00, 505),
98  -- Continue Lease Inserts
99  (1005, 205, 4, '2023-07-01', '2024-06-30', 1900.00, 2850.00, 506),
100  (1006, 206, 4, '2023-08-01', '2024-07-31', 2100.00, 3150.00, 507),
101  (1007, 207, 5, '2023-09-01', '2024-08-31', 1700.00, 2550.00, 508),
102  (1008, 208, 5, '2023-10-01', '2024-09-30', 1800.00, 2700.00, 509),
103  (1009, 209, 6, '2023-11-01', '2024-10-31', 1600.00, 2400.00, 510),
104  (1010, 210, 6, '2023-12-01', '2024-11-30', 2000.00, 3000.00, 511),
105  (1011, 211, 7, '2024-01-01', '2024-12-31', 2300.00, 3450.00, 512),
106  (1012, 212, 7, '2024-02-01', '2025-01-31', 2500.00, 3750.00, 513);
107
108  /* Insert Staff Members */
109  INSERT INTO STAFF (Staff_ID, Staff_Name, SEmail) VALUES
110  (203, 'Ravi Mehta', 'rmehta@gmail.com'),
111  (204, 'Priya Sharma', 'psharma@gmail.com'),
112  (205, 'Kunal Joshi', 'kjoshi@gmail.com'),
113  (206, 'Nikita Rao', 'nrao@gmail.com'),
114  (207, 'Amit Kapoor', 'akapoor@gmail.com'),
115  (208, 'Sneha Iyer', 'siyer@gmail.com'),
116  (209, 'Rahul Das', 'rdas@gmail.com'),
117  (210, 'Pooja Sen', 'psen@gmail.com'),
118  (211, 'Manoj Varma', 'mvarma@gmail.com'),
119  (212, 'Tanya Roy', 'troymail.com');
120
121  /* Insert Staff Phone Numbers */
122  INSERT INTO STAFF_PHONE (SPhone, Staff_ID) VALUES
123  ('555-0602', 203),
124  ('555-0603', 204),
125  ('555-0604', 205),
126  ('555-0605', 206),
127  ('555-0606', 207),
128  ('555-0607', 208),
129  ('555-0608', 209),
130  ('555-0609', 210),
131  ('555-0610', 211),
132  ('555-0611', 212);
133
```

The screenshot shows a database management tool interface. On the left, the 'Navigator' pane displays a tree view of the database schema for the 'pms' database, including Tables, Views, Stored Procedures, and Functions. Below the Navigator is the 'Schemas' pane, which shows 'Administration' and 'Information' tabs. The main editor area displays a series of SQL queries for inserting data into the 'pms' database. The queries are numbered 133 through 185. The queries are as follows:

```

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 Anora', '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

pmis

Tables

Views

Stored Procedures

Functions

sakila

sys

world

Administration Schemas

Information

No object selected

InsertValues\_Query\_GroupProj... x

Limit to 1000 rows

```
185
186  /* Insert Maintenance Requests */
187  INSERT INTO MAINTENANCE_REQUEST (Request_ID, Building_ID, Apartment_ID, Request_Date, Completion_date, Status, Assigned_Staff) VALUES
188  (403, 3, 203, '2023-05-20', '2023-05-22', 'Completed', 203),
189  (404, 3, 204, '2023-06-05', NULL, 'Pending', 204),
190  (405, 4, 205, '2023-06-15', '2023-06-17', 'Completed', 205),
191  (406, 4, 206, '2023-07-10', NULL, 'Pending', 206),
192  (407, 5, 207, '2023-08-01', NULL, 'Pending', 207),
193  (408, 5, 208, '2023-08-15', '2023-08-16', 'Completed', 208),
194  (409, 6, 209, '2023-09-01', NULL, 'Pending', 209),
195  (410, 6, 210, '2023-10-01', '2023-10-02', 'Completed', 210),
196  (411, 7, 211, '2023-11-01', NULL, 'Pending', 211),
197  (412, 7, 212, '2023-12-01', NULL, 'Pending', 212);
198
199  /* Insert Resident Feedback */
200  INSERT INTO RESIDENT_FEEDBACK (Building_ID, Apartment_ID, FeedbackDate, Category, Description, Status) VALUES
201  (3, 203, '2023-06-10', 'Facility', 'Gym equipment needs maintenance.', 'Open'),
202  (3, 204, '2023-06-12', 'Noise', 'Parties late at night.', 'Open'),
203  (4, 205, '2023-07-01', 'Maintenance', 'Air conditioning not working.', 'Resolved'),
204  (4, 206, '2023-07-10', 'Noise', 'Construction noise in the morning.', 'Open'),
205  (5, 207, '2023-08-01', 'Facility', 'Pool water is not clean.', 'Resolved'),
206  (5, 208, '2023-08-05', 'Noise', 'Elevator noise.', 'Open'),
207  (6, 209, '2023-09-01', 'Maintenance', 'Plumbing issue in bathroom.', 'Resolved'),
208  (6, 210, '2023-09-10', 'Facility', 'Lights in hallway flickering.', 'Resolved'),
209  (7, 211, '2023-10-01', 'Noise', 'Neighbors playing loud music.', 'Open'),
210  (7, 212, '2023-10-05', 'Maintenance', 'Broken window in living room.', 'Resolved');
211
212  /* Insert Utility Bills */
213  INSERT INTO UTILITY_BILLING (Building_ID, Apartment_ID, UtilityType, MonthlyCost, DueDate) VALUES
214  (3, 203, 'Electricity', 160.00, '2023-06-01'),
215  (3, 204, 'Water', 80.00, '2023-06-01'),
216  (4, 205, 'Gas', 100.00, '2023-07-01'),
217  (4, 206, 'Electricity', 170.00, '2023-07-01'),
218  (5, 207, 'Water', 85.00, '2023-08-01'),
219  (5, 208, 'Gas', 95.00, '2023-08-01'),
220  (6, 209, 'Electricity', 150.00, '2023-09-01'),
221  (6, 210, 'Water', 90.00, '2023-09-01'),
222  (7, 211, 'Gas', 110.00, '2023-10-01'),
223  (7, 212, 'Electricity', 175.00, '2023-10-01');
224
225  /* Re-enable foreign key checks */
226  SET FOREIGN_KEY_CHECKS = 1;
```



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

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

The screenshot shows a database management tool interface. On the left, the 'Navigator' pane displays a tree view of the 'pms' database schema, including tables like 'apartment', 'building', 'cleans', etc. The 'Information' pane shows the structure of the 'apartment' table:

**Table: apartment**

**Columns:**

- Apartment\_ID** int PK
- Building\_ID** int PK
- No\_of\_Bedrooms** int
- Rental\_status** varchar(10)
- Client\_ID** int

The 'Result Grid' pane displays the data for the 'apartment' table, showing columns: Apartment\_ID, Building\_ID, No\_of\_Bedrooms, Rental\_status, and Client\_ID. The data is as follows:

Apartment_ID	Building_ID	No_of_Bedrooms	Rental_status	Client_ID
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
NULL	NULL	NULL	NULL	NULL

The screenshot shows the same database management tool interface, but with the 'cleans' table selected. The 'Information' pane shows the structure of the 'cleans' table:

**Table: cleans**

**Columns:**

- Staff\_ID** int
- Apartment\_ID** int
- Building\_ID** int

The 'Result Grid' pane displays the data for the 'cleans' table, showing columns: Staff\_ID, Apartment\_ID, and Building\_ID. The data is as follows:

Staff_ID	Apartment_ID	Building_ID
203	203	3
204	204	3
205	205	4
206	206	4
207	207	5
208	208	5
209	209	6
210	210	6
211	211	7
212	212	7

Navigator

SCHEMAS

Filter objects

pms

Tables

- apartment
- building
- cleans
- corpclient
- corpclient\_phone
- inspector
- inspector\_phone
- inspects
- lease
- maintenance\_request
- manager
- manager\_phone
- resident\_feedback
- staff
- staff\_phone
- utility\_billing

Views

Administration Schemas

Information

Table: corpclient

Columns:

- Client\_ID int PK
- Client\_Name varchar(30)
- CEmail varchar(255)
- Refer\_Client\_ID int

apartment x

Limit to 1000 rows

1 SELECT \* FROM corpclient;

Result Grid

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\_request
- manager
- manager\_phone
- resident\_feedback
- staff
- staff\_phone
- utility\_billing

Views

Administration Schemas

Information

Table: corpclient

Columns:

- Client\_ID int PK
- Client\_Name varchar(30)
- CEmail varchar(255)
- Refer\_Client\_ID int

apartment x

Limit to 1000 rows

1 SELECT \* FROM corpclient\_phone;

Result Grid

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\_request
  - 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:**

Inspector_ID	InspectorName	IEmail
303	Ritka Das	rdas@exam.com
304	Kabir Malhotra	kmalh@exam.com
305	Leena Bhatt	lbhatt@exam.com
306	Aditya Singh	asingh@exam.com
307	Tarvi 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
NULL	NULL	NULL

**Navigator**

**SCHEMAS**

Filter objects

**pms**

- Tables
  - apartment
  - building
  - cleans
  - corpclient
  - corpclient\_phone
  - inspector
  - inspector\_phone
  - inspects
  - lease
  - maintenance\_request
  - 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:**

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\_request
- 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 x

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
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\_request
- 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)
- EMail** varchar(50)
- Building\_ID** int

apartment x

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	EMail	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
NULL	NULL	NULL	NULL	NULL	NULL

Navigator

SCHEMAS

Filter objects

pms

Tables

- apartment
- building
- cleans
- corpdient
- corpdient\_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 x

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
- corpdient
- corpdient\_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 x

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
NULL	NULL	NULL

The screenshot shows a database management interface. On the left, the 'SCHEMAS' pane displays a tree view of tables under the 'pms' schema, including 'apartment', 'building', 'cleans', 'corpclient', 'corpclient\_phone', 'inspector', 'inspector\_phone', 'inspects', 'lease', 'maintenance\_reques', 'manager', 'manager\_phone', 'resident\_feedback', 'staff', 'staff\_phone', and 'utility\_billing'. The 'Information' pane shows the structure of the 'staff\_phone' table: 'SPhone' (varchar(15)) and 'Staff\_ID' (int). The main query editor shows the SQL statement: `SELECT * FROM staff_phone;`. The 'Result Grid' displays the following data:

SPhone	Staff_ID
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

8. (30 points) 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.

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

The screenshot shows a database management interface. The 'SCHEMAS' pane shows the 'pms' schema with tables including 'apartment', 'building', 'cleans', 'corpclient', 'corpclient\_phone', 'inspector', 'inspector\_phone', 'inspects', 'lease', 'maintenance\_reques', 'manager', 'manager\_phone', 'resident\_feedback', 'staff', 'staff\_phone', and 'utility\_billing'. The 'Information' pane shows the structure of the 'utility\_billing' table: 'UtilityID' (int AI PK), 'Building\_ID' (int), 'Apartment\_ID' (int), 'UtilityType' (enum('Elec', 'Gas', 'Water')), 'MonthlyCost' (decimal(10, 2)), and 'DueDate' (date). The main query editor shows the SQL statement: `SELECT b.Building_ID, COUNT(a.Apartment_ID) AS Total_Apartments, SUM(CASE WHEN a.Rental_status = 'Occupied' THEN 1 ELSE 0 END) AS Occupied, ROUND((SUM(CASE WHEN a.Rental_status = 'Occupied' THEN 1 ELSE 0 END) / COUNT(a.Apartment_ID)) * 100, 2) AS Occupancy_Rate FROM BUILDING b INNER JOIN APARTMENT a ON b.Building_ID = a.Building_ID GROUP BY b.Building_ID;`. The 'Result Grid' displays the following data:

Building_ID	Total_Apartments	Occupied	Occupancy_Rate
3	2	1	50.00
4	2	1	50.00
5	2	1	50.00
6	2	1	50.00
7	2	2	100.00

## 8.2 List managers earning above average salary with their building addresses.

The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the 'pms' database schema with tables: apartment, building, cleans, corpdclient, corpdclient\_phone, inspector, and inspector\_phone. The right pane shows a query window with the following SQL code:

```
9
10 /* List managers earning above average salary with their building addresses. */
11 SELECT m.MFull_Name AS Manager_Name, m.Salary, CONCAT(b.Street, ' ', b.City, ' ', b.State, ' ', b.ZipCode) AS Building_Address
12 FROM MANAGER m
13 LEFT JOIN BUILDING b ON m.Building_ID = b.Building_ID
14 WHERE m.Salary > (SELECT AVG(Salary) FROM MANAGER);
15
16 /* Find leases with durations longer than 1 year. */
```

The 'Result Grid' shows the following data:

Manager_Name	Salary	Building_Address
Meera Nair	75000	303 Elm St, Norwalk, CT 6851
Krishna Verma	73000	606 Pine Ave, New Haven, CT 6510
Sai Prakash	74000	808 Cedar Blvd, Waterbury, CT 6702
Diya Sharma	76000	909 Elm Ave, New London, CT 6320

The 'Information' pane shows the structure of the 'utility\_billing' table:

**Table: utility\_billing**

**Columns:**

- UtilityID: int AI PK
- Building\_ID: int
- Apartment\_ID: int
- UtilityType: enum('Elec
- MonthlyCost: decimal(10
- DueDate: date

## 8.3 Count maintenance requests by status per building.

The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the 'pms' database schema with tables: apartment, building, cleans, corpdclient, corpdclient\_phone, inspector, and inspector\_phone. The right pane shows a query window with the following SQL code:

```
23 /* Count maintenance requests by status per building. */
24 SELECT b.Building_ID,
25        mr.Status,
26        COUNT(mr.Request_ID) AS Total_Requests
27 FROM BUILDING b
28 INNER JOIN MAINTENANCE_REQUEST mr ON b.Building_ID = mr.Building_ID
29 GROUP BY b.Building_ID, mr.Status;
30
```

The 'Result Grid' shows the following data:

Building_ID	Status	Total_Requests
3	Completed	1
3	Pending	1
4	Completed	1
4	Pending	1
5	Pending	1
5	Completed	1
6	Pending	1
6	Completed	1
7	Pending	2

The 'Information' pane shows the structure of the 'utility\_billing' table:

**Table: utility\_billing**

**Columns:**

- UtilityID: int AI PK
- Building\_ID: int
- Apartment\_ID: int
- UtilityType: enum('Elec
- MonthlyCost: decimal(10
- DueDate: date

## 8.4 Show unresolved feedback with apartment details.

The screenshot shows a database management tool interface. On the left, the 'SCHEMAS' pane displays a tree view of the 'pms' database, including tables like 'apartment', 'building', 'cleans', 'corpdient', 'corpdient\_phone', 'inspector', and 'inspector\_phone'. Below this, the 'Information' pane shows details for the 'utility\_billing' table, including its columns and data types.

The main area displays a SQL query in the 'infoQuery\_GroupProjectPart1' window:

```
31 /* Show unresolved feedback with apartment details. */
32 SELECT rf.FeedbackID, rf.Description, a.Building_ID, a.Apartment_ID
33 FROM RESIDENT_FEEDBACK rf
34 LEFT JOIN APARTMENT a ON rf.Building_ID = a.Building_ID AND rf.Apartment_ID = a.Apartment_ID
35 WHERE rf.Status = 'Open';
36
```

Below the query, the 'Result Grid' shows the following data:

FeedbackID	Description	Building_ID	Apartment_ID
1	Gym equipment needs maintenance.	3	203
2	Parties late at night.	3	204
4	Construction noise in the morning.	4	206
6	Elevator noise.	5	208
9	Neighbors playing loud music.	7	211

## 8.5 Calculate total utility costs per building.

The screenshot shows the same database management tool interface. The 'SCHEMAS' and 'Information' panes are the same as in the previous screenshot.

The main area displays a SQL query in the 'infoQuery\_GroupProjectPart1' window:

```
37 /* Calculate total utility costs per building. */
38 SELECT b.Building_ID,
39 SUM(ub.MonthlyCost) AS Total_Utility_Cost
40 FROM BUILDING b
41 INNER JOIN UTILITY_BILLING ub ON b.Building_ID = ub.Building_ID
42 GROUP BY b.Building_ID;
43
44 /* List corporate clients who didn't refer anyone. */

```

Below the query, the 'Result Grid' shows the following data:

Building_ID	Total_Utility_Cost
3	240.00
4	270.00
5	180.00
6	240.00
7	285.00



## 8.6 List corporate clients who didn't refer anyone.

The screenshot shows a SQL Studio interface. On the left, the 'Navigator' pane displays the 'pms' schema with tables: apartment, building, cleans, corpcient, corpcient\_phone, inspector, and inspector\_phone. The 'Information' pane shows details for the 'utility\_billing' table, including columns: UtilityID (int AI PK), Building\_ID (int), Apartment\_ID (int), UtilityType (enum('Elec', 'Water', 'Gas')), MonthlyCost (decimal(10, 2)), and DueDate (date). The main query editor shows the following SQL code:

```
43
44  /* List corporate clients who didn't refer anyone. */
45  SELECT Client_ID AS CCID, Client_Name AS CCName
46  FROM CORPCLIENT
47  WHERE Client_ID NOT IN (SELECT Refer_Client_ID FROM CORPCLIENT WHERE Refer_Client_ID IS NOT NULL);
48
```

The 'Result Grid' at the bottom shows the following data:

CCID	CCName
513	Bright Future School

## 8.7 Count inspections per inspector.

The screenshot shows a SQL Studio interface. On the left, the 'Navigator' pane displays the 'pms' schema with tables: apartment, building, cleans, corpcient, corpcient\_phone, inspector, and inspector\_phone. The 'Information' pane shows details for the 'utility\_billing' table, including columns: UtilityID (int AI PK), Building\_ID (int), Apartment\_ID (int), UtilityType (enum('Elec', 'Water', 'Gas')), MonthlyCost (decimal(10, 2)), and DueDate (date). The main query editor shows the following SQL code:

```
49  /* Count inspections per inspector. */
50  SELECT i.InspectorName AS Inspector_Name, COUNT(ins.Inspector_ID) AS Total_Inspections
51  FROM INSPECTOR i
52  LEFT JOIN INSPECTS ins ON i.Inspector_ID = ins.Inspector_ID
53  GROUP BY i.InspectorName;
54
```

The 'Result Grid' at the bottom shows the following data:

Inspector_Name	Total_Inspections
Ritika Das	1
Kabir Malhotra	1
Leena Bhatt	1
Aditya Singh	1
Tanvi Saxena	1
Mohit Jain	1
Nidhi Kapoor	1
Ajay Deshmukh	1
Simran Kohli	1
Dev Arora	1

## 8.8 Create a trigger to mark apartments as occupied when leased.

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'SCHEMAS' pane displays a tree view for the 'pms' database, including tables like 'apartment', 'building', 'cleans', 'corpclient', 'corpclient\_phone', and 'inspector'. The main pane, titled 'infoQuery\_GroupProjectPart1', displays a SQL script for creating a trigger. The script is as follows:

```
65 AFTER INSERT ON LEASE
66 FOR EACH ROW
67 BEGIN
68     UPDATE APARTMENT
69     SET Rental_status = 'Occupied'
70     WHERE Building_ID = NEW.Building_ID AND Apartment_ID = NEW.Apartment_ID;
71 END;
72 //
```

Below the script, a list of SQL queries is visible, including a CREATE TRIGGER statement for 'UpdateApartmentStatus'.

## 8.9 Automatically log completed maintenance requests

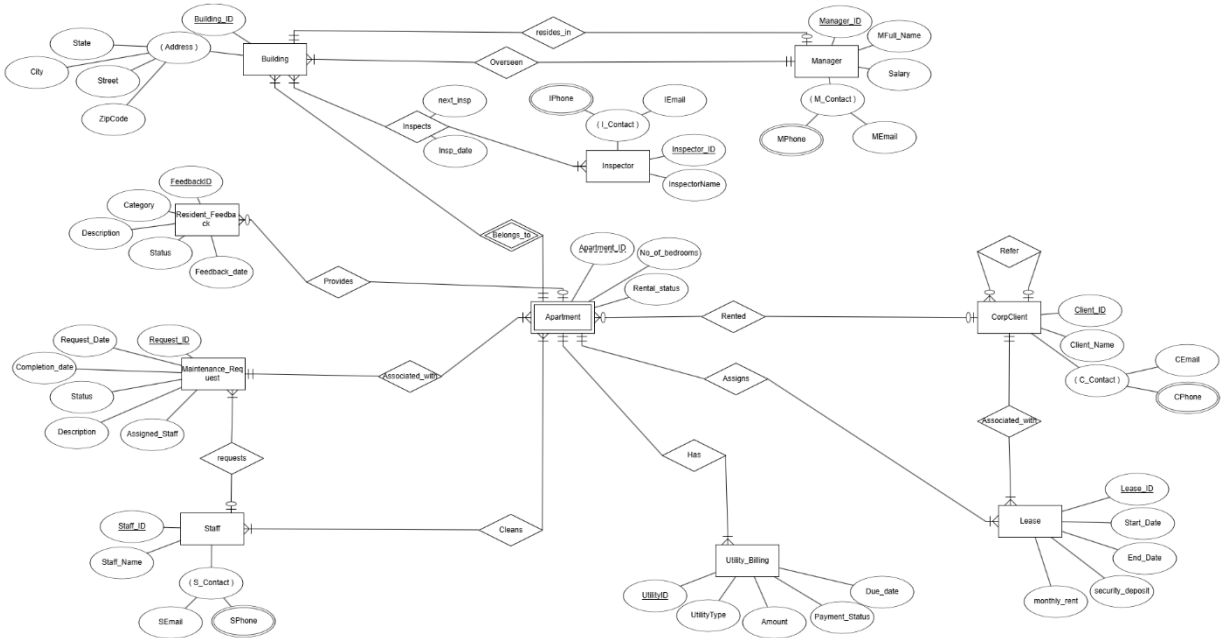
The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'SCHEMAS' pane displays a tree view for the 'pms' database, including tables like 'apartment', 'building', 'cleans', 'corpclient', 'corpclient\_phone', and 'inspector'. The main pane, titled 'infoQuery\_GroupProjectPart1', displays a SQL script for creating a trigger. The script is as follows:

```
104 DELIMITER //
105 CREATE TRIGGER LogCompletedMaintenance
106 AFTER UPDATE ON MAINTENANCE_REQUEST
107 FOR EACH ROW
108 BEGIN
109     IF NEW.Status = 'Completed' AND OLD.Status != 'Completed' THEN
110         INSERT INTO MAINTENANCE_LOG (Request_ID, CompletionDate)
111         VALUES (NEW.Request_ID, NOW());
112     END IF;
113 END;
```

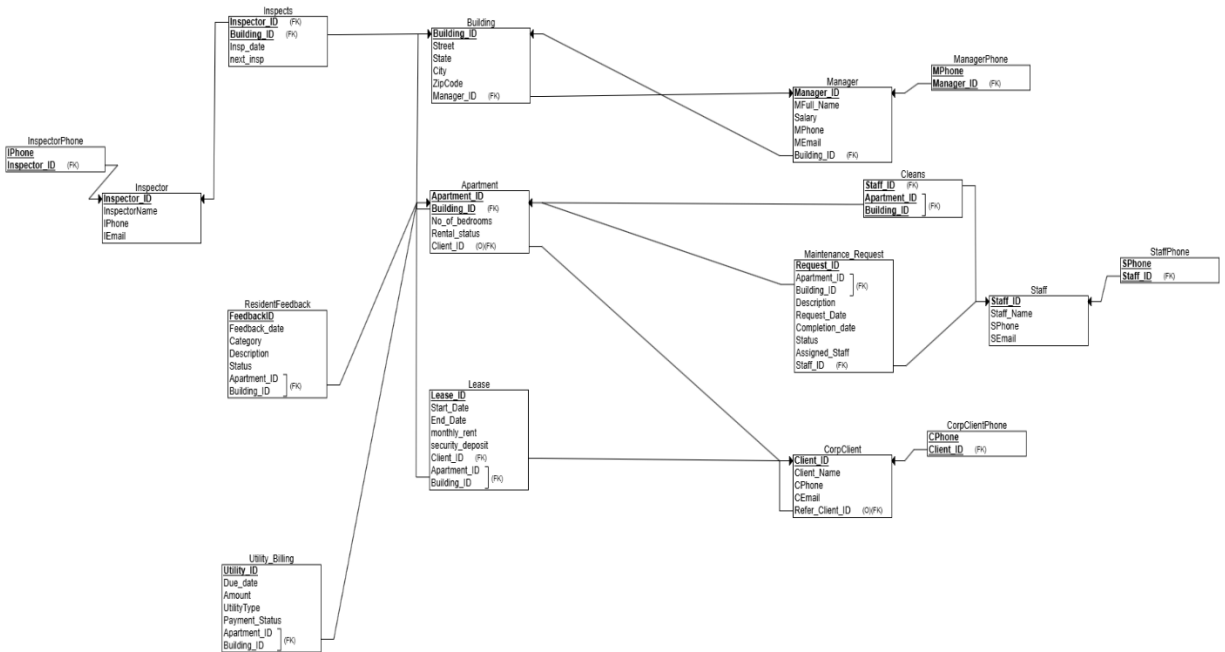
Below the script, a list of SQL queries is visible, including a CREATE TRIGGER statement for 'LogCompletedMaintenance'.



## 9. Entity-Relationship: PROPERTY MANAGEMENT SYSTEM: 2 NEW ENTITIES



### 9.1 Relational-Schema: PROPERTY MANAGEMENT SYSTEM: 2 NEW ENTITIES



## 9.2 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	-

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

### 9.3.1 High Utility Cost Apartments with Pending Feedback

Navigator

SCHEMAS

Filter objects

- pms
  - Tables
    - apartment
    - building
    - cleans
    - corpcient
    - corpcient\_phone
    - inspector
    - inspector\_phone
    - inspects
    - lease
    - maintenance\_request
    - manager
    - manager\_phone
    - resident\_feedback
    - staff
    - staff\_phone
    - utility\_billing
  - Views
  - Stored Procedures
  - Functions
- sakila
- sys
- world

BusinessQuery\_1\_GroupProject...

Limit to 1000 rows

```

1  SELECT
2      ub.Apartment_ID,
3      ub.Building_ID,
4      FORMAT(ub.MonthlyCost, 2) AS Current_UtilityCost,
5      (
6          SELECT FORMAT(AVG(MonthlyCost), 2)
7          FROM UTILITY_BILLING
8          WHERE UtilityType = ub.UtilityType
9      ) AS Type_Avg,
10     (
11         SELECT GROUP_CONCAT(Description SEPARATOR ' | ')
12         FROM RESIDENT_FEEDBACK
13         WHERE Apartment_ID = ub.Apartment_ID
14             AND Building_ID = ub.Building_ID
15             AND Status = 'Open'
16     ) AS Open_Issues
17 FROM UTILITY_BILLING ub
18 WHERE ub.MonthlyCost > (
19     SELECT AVG(MonthlyCost)
20     FROM UTILITY_BILLING
21     WHERE UtilityType = ub.UtilityType
22 )
23 AND EXISTS (
24     SELECT 1
25     FROM RESIDENT_FEEDBACK
26     WHERE Apartment_ID = ub.Apartment_ID
27         AND Building_ID = ub.Building_ID
28         AND Status = 'Open'
29 )
30 ORDER BY ub.MonthlyCost DESC;
31

```

Administration Schemas

Information

Schema: pms

Result Grid

Filter Rows:

Export: Wrap Cell Content:

Apartment_ID	Building_ID	Current_UtilityCost	Type_Avg	Open_Issues
206	4	170.00	163.75	Construction noise in the morning.
211	7	110.00	101.67	Neighbors playing loud music.

### 9.3.2 Utility payment compliance check

The screenshot displays a database management interface with a left-hand sidebar and a main workspace. The sidebar, titled 'Navigator', shows a tree view of the 'pms' schema, including tables like 'apartment', 'building', 'cleans', 'corpclient', 'corpclient\_phone', 'inspector', 'inspector\_phone', 'inspects', 'lease', 'maintenance\_request', 'manager', 'manager\_phone', 'resident\_feedback', 'staff', 'staff\_phone', and 'utility\_billing'. The main workspace, titled 'BusinessQuery\_2\_GroupProject...', contains a SQL query and a 'Result Grid' at the bottom.

**SQL Query:**

```
1 SELECT
2     ub.Building_ID,
3     ub.Apartment_ID,
4     ub.UtilityType,
5     FORMAT(ub.MonthlyCost, 2) AS Due_Amount,
6     ub.DueDate,
7     CASE
8         WHEN DATEDIFF(CURDATE(), ub.DueDate) > 30 THEN 'Delinquent'
9         WHEN DATEDIFF(CURDATE(), ub.DueDate) > 15 THEN 'Overdue'
10        ELSE 'Current'
11    END AS Payment_Status,
12    (
13        SELECT COUNT(*)
14        FROM RESIDENT_FEEDBACK
15        WHERE Apartment_ID = ub.Apartment_ID
16            AND Building_ID = ub.Building_ID
17            AND Category = 'Facility'
18    ) AS Facility_Complaints
19 FROM UTILITY_BILLING ub
20 ORDER BY Payment_Status, DueDate DESC;
21
```

**Result Grid:**

	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

### 9.3.3 Apartments with Unresolved Feedback & High Utility Costs

The screenshot shows a SQL query in the BusinessQuery\_3\_GroupProject... window. The query selects apartment details and utility costs for unresolved feedback issues where the monthly cost is above the building average. The result grid shows two rows of data.

```
1 SELECT
2     rf.Apartment_ID,
3     rf.Building_ID,
4     rf.Description AS Open_Issue,
5     FORMAT(ub.MonthlyCost, 2) AS Utility_Cost
6 FROM RESIDENT_FEEDBACK rf
7 JOIN UTILITY_BILLING ub
8     ON rf.Apartment_ID = ub.Apartment_ID
9     AND rf.Building_ID = ub.Building_ID
10 WHERE rf.Status = 'Open'
11     AND ub.MonthlyCost > (
12         SELECT AVG(MonthlyCost)
13         FROM UTILITY_BILLING
14     );
```

Apartment_ID	Building_ID	Open_Issue	Utility_Cost
203	3	Gym equipment needs maintenance.	160.00
206	4	Construction noise in the morning.	170.00

### 9.3.4 Utility Costs vs Building Average

The screenshot shows a SQL query in the BusinessQuery\_4\_GroupProject... window. The query compares the monthly utility cost of each apartment to the average monthly cost of its building. The result grid shows five rows of data.

```
1 SELECT
2     ub.Apartment_ID,
3     ub.Building_ID,
4     FORMAT(ub.MonthlyCost, 2) AS Current_Cost,
5     FORMAT(
6         (
7             SELECT AVG(MonthlyCost)
8             FROM UTILITY_BILLING
9             WHERE Building_ID = ub.Building_ID
10        ), 2
11     ) AS Building_Avg_Cost
12 FROM UTILITY_BILLING ub
13 WHERE ub.MonthlyCost > (
14     SELECT AVG(MonthlyCost)
15     FROM UTILITY_BILLING
16     WHERE Building_ID = ub.Building_ID
17 );
```

Apartment_ID	Building_ID	Current_Cost	Building_Avg_Cost
203	3	160.00	120.00
206	4	170.00	135.00
208	5	95.00	90.00
209	6	150.00	120.00
212	7	175.00	142.50

### 9.3.5 Open Feedback Count with Total Utility Cost

The screenshot displays a database management interface. On the left, a 'Navigator' pane shows the 'pms' schema with various tables. The main area shows a SQL query in a text editor, and the bottom pane shows the 'Result Grid' with the query's output.

**SQL Query:**

```
1 SELECT
2     ub.Apartment_ID,
3     ub.Building_ID,
4     FORMAT(SUM(ub.MonthlyCost), 2) AS Total_Uilities,
5     (
6         SELECT COUNT(*)
7         FROM RESIDENT_FEEDBACK
8         WHERE Apartment_ID = ub.Apartment_ID
9             AND Building_ID = ub.Building_ID
10            AND Status = 'Open'
11     ) AS Open_Feedback_Count
12 FROM UTILITY_BILLING ub
13 GROUP BY ub.Apartment_ID, ub.Building_ID
14 HAVING Open_Feedback_Count > 0;
```

**Result Grid:**

Apartment_ID	Building_ID	Total_Uilities	Open_Feedback_Count
203	3	160.00	1
204	3	80.00	1
206	4	170.00	1
208	5	95.00	1
211	7	110.00	1

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 Insights:

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

## **2. 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.