



Students name:	Dias Nurbergenov and Zhangir Bayanov
Teacher name:	Yerasyl Amanbek
Project name:	Building Management System
Project type:	Teamwork
Number of words	2800

## ABSTRACT

Before defining project purpose, there has to be found the problems the building companies are dealing without using Building Management System (BMS). Therefore, here are the following problems we found: high energy consumption, data security of the occupants, immobility of access to data from anywhere from mobile devices (laptop, phone, tablet, etc.), less comfortable living conditions for residents, the complexity of storing and processing all large amounts of data.

During the researches, we found that the only and quality solution for above mentioned “pains” is having the Building Management System. Our interpretation consists itself creating the understandable database for building owners. It means that our database should provide all necessary informations in order to get proper interaction between the occupant and the building system.

## INTRODUCTION

### Background:

Generally, the Building Management System is a computer control system installed in buildings that manages all internal building processes such as ventilation, lighting, power systems, fire systems, security systems, and more. Usually, BMS is implemented in large projects with an extensive system for managing mechanical processes. The systems are:

HVAC (heating, ventilation, and air conditioning) system is a system that is responsible for managing all possible actions related to building ventilation. In case of accidents or any anomalies, the system raises an alarm.

Energy system can monitor the electrical power consumption and the status of the electrical switches.

Sprinkler system automatically calls the fire brigade and signals evacuation if a fire occurs inside or on the territory of the building. This system is also responsible for everything related to fire safety

Water system serves as a proper functioning of the water distribution.

Security system includes storage of data about residents and building security.

### Literature review

The Building Management System (BMS) is a comprehensive management system that is responsible to control the subsystems in a facility automatically (Joseph, 2018).

The BMS must be installed inside of the buildings in order to regulate the equipment such as electrical, water, air conditioning control systems and so on. The BMS reduces the building energy consumptions and the maintenance costs (Hossain, 2019).

This system can be considered as “Smart Home” or “Smart Building” which includes technology systems. For instance: fire alarms, HVAC (Heating Ventilation and Air-Conditioning) system (Sinopoli, 2010).

The BMS is responsible for directly receiving sensor information via the subscriptions performed to the different sensors inside the building. By using a lightweight publish/subscribe client, once a new message is stored in the middleware database, it is also forwarded to the application, allowing the BMS to act accordingly if necessary (Sembroiz, 2018)

Building management systems are typically a personal or embedded computer, operating dedicated BMS software, and communicating through an industrial control network to the HVAC equipment. (Platt, 2013)

### The importance of the work

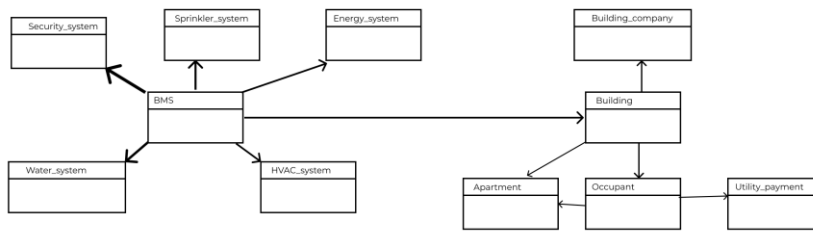
The main idea of the project is to show an example of how BMS works.

The importance of the project is to help building owners or companies calculate prices for documents and the building itself. Facilitate the management of internal building systems, as well as make it convenient to use. To improve the safety of the building. Fast access to data and management, providing reliable protection of user data and analysis as needed.

## **METHODOLOGY**

### Daily actions:

First of all, the name of the tables was invented. After they were invented, a draft version of the ERD was created.



In the beginning, it just showed which tables will be joined in the future. After this ERD was approved, work started. The project was split. The part related to people was taken by Dias, and the part related to systems was taken by Zhangir. The attribute of tables and the connection between tables were invented. Then a normal ERD was born, which in the future suffered a lot of corrections related to attributes. There were connection errors somewhere, but they were fixed. After the ERD was accepted by the development team, work began on the sql code. Tables and relationships between them were created via Primary key and Foreign key. After the successful implementation of ERD in sql, the developers began to fill in the tables with the value. Each table was filled with 10 values. After successful completion, we started working on the commands. First in line were the ALTER TABLE statement. Five such applications were successfully written. Immediately after that, 10 DML statements were created. After some time, queries and subqueries with different conditions were created. After a long discussion of the entire code, some commands were replaced with more relevant and acceptable ones. When everything was accepted, the team started working on the third task.

Overall, there are 11 tables.

#### List of entities:

'Building\_company', 'Building', 'Apartment', 'Occupant', 'Utility\_payment', 'BMS', 'HVAC\_system', 'Water\_system', 'Energy\_system', 'Security\_system', 'Sprinkler\_system'

#### Related entities:

Building\_company – Building; Building – BMS; Building – Apartment; Apartment – Occupant; Occupant – Utility\_payment; Occupant – BMS; BMS – HVAC\_system; BMS – Security\_system; BMS – Water\_system; BMS – Sprinkler\_System; BMS – Energy\_system;

#### Optionality and plurality constraints:

Building\_company → Building (mandatory, plural)

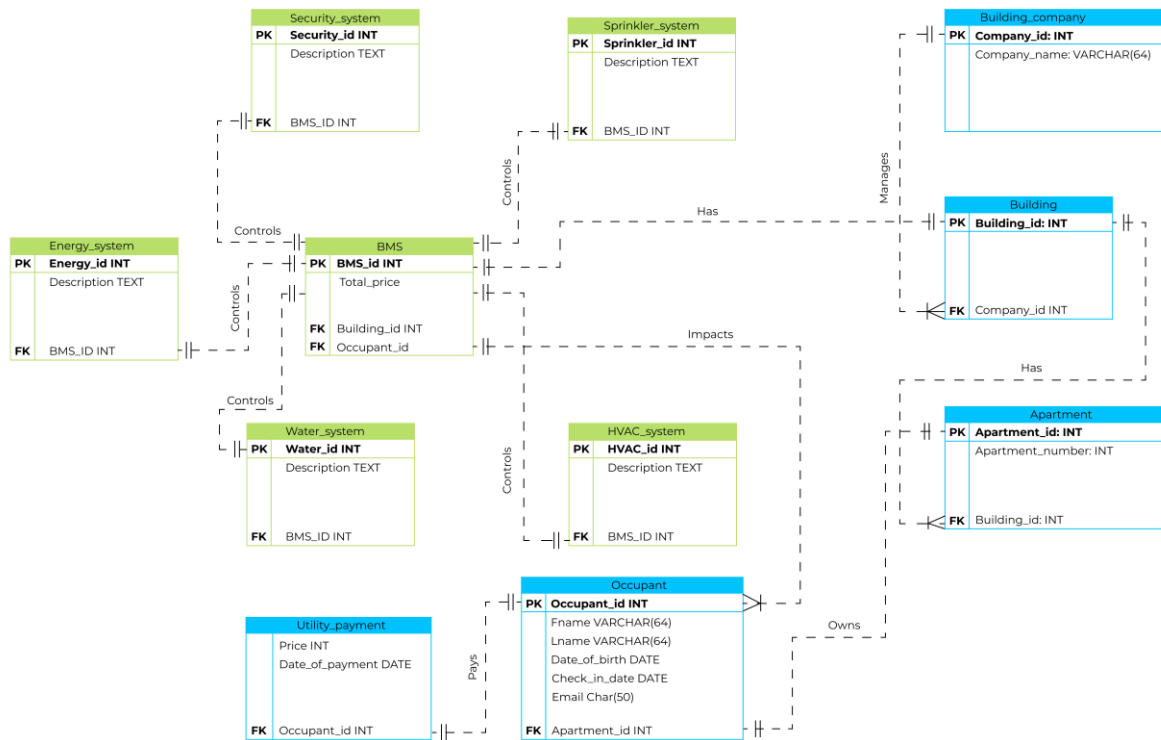
Building → Apartment (mandatory, plural)

Apartment → Occupant (optional, singular)  
Occupant → Utility\_payment (mandatory, singular)  
Building → BMS (mandatory, singular)  
Occupant → BMS (mandatory, singular)  
BMS → Security\_system (mandatory, singular)  
BMS → Energy\_system (mandatory, singular)  
BMS → Sprinkler\_system (mandatory, singular)  
BMS → Water\_system (mandatory, singular)  
BMS → HVAC\_system (mandatory, singular)

The business rules:

The company is going to build buildings and each such building has its own unique number. The building has many apartments with a specific id and number. A person lives under each such Id, before registering an apartment for himself, he provides complete information about himself, and he is given a unique id so that a request for payment of utilities comes to this Id and thanks to this id he can manage his BMS. The building has many systems such as security system, energy system, HVAC system, water system, sprinkler system. Each system is responsible for certain physical processes of the building, and all these systems are controlled by the BMS.


The Entity-Relationship Diagram (ERD) of the Building Management System:



The whole tables are filled with 10 rows.

### 1. Building\_company table.

The column 'company\_name' are filled with the building company names. The inserted company names are founded in the Internet. Primary Key is 'company\_id', because this table is related with the table 'building'

	 <b>company_id</b> [PK] integer 	<b>company_name</b> character varying (64) 
1	101	BiGroup
2	102	Trump Tower
3	103	Modern Structure
4	104	Ace & Hammer Builders
5	105	Diamond Ridge Construction
6	106	Center Circle Design-Build
7	107	Builder Gorilla
8	108	Pure Renovation Company
9	109	Mod Guys Construction
10	110	Build It Brothers



There are used **INSERT INTO** operator, in order to insert values into the table. In this case the table name is 'building\_company'. For the rest of the tables, all of them are used the same operator.

**INSERT INTO** building\_company(company\_id, company\_name)  
**VALUES**

```
(101, 'BiGroup'),
(102, 'Trump Tower'),
(103, 'Modern Structure'),
(104, 'Ace & Hammer Builders'),
(105, 'Diamond Ridge Construction'),
(106, 'Center Circle Design-Build'),
(107, 'Builder Gorilla'),
(108, 'Pure Renovation Company'),
(109, 'Mod Guys Construction'),
(110, 'Build It Brothers');
```

## 2. Building table

The columns 'building\_id' and 'company\_id' are filled with the id numbers. Primary Key is 'building\_id', because this table is related with the table 'Apartment'.

	 building_id [PK] integer	 company_id integer
1	1	101
2	2	102
3	3	103
4	4	104
5	5	105
6	6	106
7	7	107
8	8	108
9	9	109
10	10	110

The codes for above table:

```
INSERT INTO building(building_id, company_id)
VALUES
(1, 101),
(2, 102),
(3, 103),
(4, 104),
(5, 105),
(6, 106),
(7, 107),
(8, 108),
(9, 109),
(10, 110);
```

### 3. Apartment table

There is written id numbers for the 'apartment\_id' which is also Primary Key. 'Apartment\_number' is also integer number. The last one is 'building\_id' column which is Foreign Key.

	apartment_id [PK] integer	apartment_number integer	building_id integer
1	200	50	1
2	201	51	2
3	202	52	3
4	203	53	4
5	204	54	5
6	205	55	6
7	206	56	7
8	207	57	8
9	208	58	9
10	209	59	10

The codes for the above table:

```
INSERT INTO apartment(apartment_id, apartment_number, building_id)
VALUES
(200, 50, 1),
(201, 51, 2),
(202, 52, 3),
(203, 53, 4),
(204, 54, 5),
(205, 55, 6),
(206, 56, 7),
(207, 57, 8),
(208, 58, 9),
(209, 59, 10);
```

#### 4. Occupant table.

There is written 'occupant\_id' as Primary Key with the numbers. The columns 'fname' and 'lname' are used for the resident names. 'Date\_of\_birth' represents the birth date of the occupants. 'Email' column shows the email address of the person. The 'apartment\_id' is a Foreign Key.

	occupant_id [PK] integer	fname character varying (64)	lname character varying (64)	date_of_birth date	check_in_date date	email character (500)	apartment_id integer
1	300	Dias	Nurbergenov	2003-01-05	2020-09-01	diasnktteam@gmail...	200
2	301	Zhangir	Bayanov	2003-04-29	2020-10-22	bayanov@gmail.com...	201
3	302	Cristiano	Ronaldo	1985-02-05	2019-12-31	cristiano@gmail.co...	202
4	303	Lionel	Messi	1987-06-24	2015-03-21	leomessi@gmail.co...	203
5	304	Sergio	Ramos	1986-01-30	2017-10-08	sergio@gmail.com	204
6	305	Oliver	Queen	1988-01-22	2010-01-28	greenarrow@gmail.c...	205
7	306	Barry	Allen	1991-07-07	2012-12-12	flash@gmail.com	206
8	307	Jeff	Bezos	1964-01-12	2007-04-01	amazon@info.com	207
9	308	Bill	Gates	1055-10-28	2009-02-28	gates@gmail.com	208
10	309	Mark	Zuckerberg	1984-05-14	2018-08-11	facebook@support.c...	209

The codes for above table:



```
INSERT INTO occupant(occupant_id, fname, lname, date_of_birth, check_in_date, email, apartment_id)
VALUES
(300, 'Dias', 'Nurbergenov', '2003-01-05', '2020-09-01', 'diankteam@gmail.com', 200),
(301, 'Zhangir', 'Bayanov', '2003-04-29', '2020-10-22', 'bayanove@gmail.com', 201),
(302, 'Cristiano', 'Romaldo', '1985-02-09', '2019-12-31', 'cristiano@gmail.com', 202),
(303, 'Lionel', 'Messi', '1987-06-24', '2015-03-21', 'leomessi@gmail.com', 203),
(304, 'Sergio', 'Ramos', '1986-01-30', '2017-10-08', 'sergio@gmail.com', 204),
(305, 'Oliver', 'Queen', '1988-01-22', '2010-01-28', 'greenarrow@gmail.com', 205),
(306, 'Barry', 'Allen', '1991-07-07', '2012-12-12', 'flash@gmail.com', 206),
(307, 'Jeff', 'Bezos', '1964-01-12', '2007-04-01', 'amazon@info.com', 207),
(308, 'Bill', 'Gates', '1955-10-28', '2009-02-28', 'gates@gmail.com', 208),
(309, 'Mark', 'Zuckerberg', '1984-05-14', '2018-08-11', 'facebook@support.com', 209);
```

## 5. Utility\_payment table.

The 'price' column shows the amount of the price in american dollars. 'Date\_of\_price' represents the transaction day from the occupant. 'Occupant\_id' is a Foreign key.

	price integer	date_of_price date	occupant_id integer
1	2060	2020-10-01	300
2	2160	2020-11-22	301
3	2260	2020-01-31	302
4	2360	2015-04-21	303
5	2460	2017-11-08	304
6	2560	2010-02-28	305
7	2660	2013-01-12	306
8	2760	2007-05-01	307
9	2860	2009-03-28	308
10	2960	2018-09-11	309

The codes for above table:

```
INSERT INTO utility_payment (price, date_of_price, occupant_id)
VALUES
(2060, '2020-10-01', 300),
(2160, '2020-11-22', 301),
(2260, '2020-01-31', 302),
(2360, '2015-04-21', 303),
(2460, '2017-11-08', 304),
(2560, '2010-02-28', 305),
(2660, '2013-01-12', 306),
(2760, '2007-05-01', 307),
(2860, '2009-03-28', 308),
(2960, '2018-09-11', 309);
```

## 6. BMS table.

There is written the column 'bms\_id' as id number. The column 'total\_price' shows the overall price for using all systems. The Foreign keys are: 'building\_id' and 'occupant\_id' columns.

	bms_id [PK] integer	total_price integer	building_id integer	occupant_id integer
1	1001	7500	1	300
2	1002	6000	2	301
3	1003	8000	3	302
4	1004	7900	4	303
5	1005	6500	5	304
6	1006	9850	6	305
7	1007	10000	7	306
8	1008	9500	8	307
9	1009	6000	9	308
10	1010	7500	10	309

The codes for above table:

```

INSERT INTO BMS (BMS_id, total_price, building_id, occupant_id)
VALUES (1001, '7500', 1, 300 );
INSERT INTO BMS (BMS_id, total_price, building_id, occupant_id)
VALUES (1002, '6000', 2, 301);
INSERT INTO BMS (BMS_id, total_price, building_id, occupant_id)
VALUES (1003, '8000', 3, 302);
INSERT INTO BMS (BMS_id, total_price, building_id, occupant_id)
VALUES (1004, '7900', 4, 303);
INSERT INTO BMS (BMS_id, total_price, building_id, occupant_id)
VALUES (1005, '6500', 5, 304);
INSERT INTO BMS (BMS_id, total_price, building_id, occupant_id)
VALUES (1006, '9850', 6, 305);
INSERT INTO BMS (BMS_id, total_price, building_id, occupant_id)
VALUES (1007, '10000', 7, 306);
INSERT INTO BMS (BMS_id, total_price, building_id, occupant_id)
VALUES (1008, '9500', 8, 307);
INSERT INTO BMS (BMS_id, total_price, building_id, occupant_id)
VALUES (1009, '6000', 9, 308);
INSERT INTO BMS (BMS_id, total_price, building_id, occupant_id)
VALUES (1010, '7500', 10, 309);

```

## 7. HVAC\_system table.

Primary key is 'hvac\_id' is filled with the id numbers. The column 'description' shows the brief information of the HVAC system. Every resident has different types of the HVAC system. 'bms\_id' is a Foreign key.

	hvac_id [PK] integer	description text	bms_id integer
1	10001	air intake, heat recovery, preheating, cooling	1001
2	10002	filtration, air heating, preheating, extractio	1002
3	10003	recirculation, air heating, aspiration, extraction, filtration	1003
4	10004	air heating, regulation of the differential pressure between rooms, aspiration	1004
5	10005	heat recovery, aspiration, filtration, air heating, cooling	1005
6	10006	aspiration, supply to a clean room, cooling	1006
7	10007	filtering, air heating, preheating, the supply to the clean room	1007
8	10008	air heating, regulation of the differential pressure between rooms, filtration	1008
9	10009	heat recovery, aspiration, filtration, air heating, cooling	1009
10	10010	regulation of the differential pressure between rooms, aspiration, filtration, air heating, cooling	1010

The codes for the above table:

```

INSERT INTO HVAC_system (HVAC_id, Description, BMS_id)
VALUES (10001, 'air intake, heat recovery, preheating, cooling', 1001);
INSERT INTO HVAC_system (HVAC_id, Description, BMS_id)
VALUES (10002, 'filtration, air heating, preheating, extractio', 1002);
INSERT INTO HVAC_system (HVAC_id, Description, BMS_id)
VALUES (10003, 'recirculation, air heating, aspiration, extraction, filtration', 1003);
INSERT INTO HVAC_system (HVAC_id, Description, BMS_id)
VALUES (10004, 'air heating, regulation of the differential pressure between rooms, aspiration', 1004);
INSERT INTO HVAC_system (HVAC_id, Description, BMS_id)
VALUES (10005, 'heat recovery, aspiration, filtration, air heating, cooling', 1005);
INSERT INTO HVAC_system (HVAC_id, Description, BMS_id)
VALUES (10006, 'aspiration, supply to a clean room, cooling', 1006);
INSERT INTO HVAC_system (HVAC_id, Description, BMS_id)
VALUES (10007, 'filtering, air heating, preheating, the supply to the clean room', 1007);
INSERT INTO HVAC_system (HVAC_id, Description, BMS_id)
VALUES (10008, 'air heating, regulation of the differential pressure between rooms, filtration', 1008);
INSERT INTO HVAC_system (HVAC_id, Description, BMS_id)
VALUES (10009, 'heat recovery, aspiration, filtration, air heating, cooling', 1009);
INSERT INTO HVAC_system (HVAC_id, Description, BMS_id)
VALUES (10010, 'regulation of the differential pressure between rooms, aspiration, filtration, air heating, cooling', 1010);

```

## 8. Water\_system table.

There is written 'water\_id' as Primary key. 'description' column shows the informations of the Water system. Some of the occupants have same model of the water system.

'bms\_id' is a Foreign key.

	water_id [PK] integer	description text	bms_id integer
1	10001	cleaning, storage and distribution of water for apartments, yard washing	1001
2	10002	water filtration, yard flushing, autonomous watering of plants	1002
3	10003	yard washing, cleaning, swimming pool	1003
4	10004	storage and distribution of water for apartments, yard washing, cleaning, water filtration	1004
5	10005	Coagulation, water filtration, swimming pool,	1005
6	10006	flocculation, cleaning, water filtration	1006
7	10007	yard washing, cleaning, swimming pool	1007
8	10008	water filtration, yard flushing, autonomous watering of plants	1008
9	10009	storage and distribution of water for apartments, yard washing, cleaning, water filtration	1009
10	10010	flocculation, cleaning, water filtration	1010

The codes for above table:

```
INSERT INTO Water_system (Water_id, Description, BMS_id)
VALUES (10001, 'cleaning, storage and distribution of water for apartments, yard washing', 1001);
INSERT INTO Water_system (Water_id, Description, BMS_id)
VALUES (10002, 'water filtration, yard flushing, autonomous watering of plants', 1002);
INSERT INTO Water_system (Water_id, Description, BMS_id)
VALUES (10003, 'yard washing, cleaning, swimming pool', 1003);
INSERT INTO Water_system (Water_id, Description, BMS_id)
VALUES (10004, 'storage and distribution of water for apartments, yard washing, cleaning, water filtration', 1004);
INSERT INTO Water_system (Water_id, Description, BMS_id)
VALUES (10005, 'Coagulation, water filtration, swimming pool,', 1005);
INSERT INTO Water_system (Water_id, Description, BMS_id)
VALUES (10006, 'flocculation, cleaning, water filtration', 1006);
INSERT INTO Water_system (Water_id, Description, BMS_id)
VALUES (10007, 'yard washing, cleaning, swimming pool', 1007);
INSERT INTO Water_system (Water_id, Description, BMS_id)
VALUES (10008, 'water filtration, yard flushing, autonomous watering of plants', 1008);
INSERT INTO Water_system (Water_id, Description, BMS_id)
VALUES (10009, 'storage and distribution of water for apartments, yard washing, cleaning, water filtration', 1009);
INSERT INTO Water_system (Water_id, Description, BMS_id)
VALUES (10010, 'flocculation, cleaning, water filtration', 1010);
```

## 9. Energy\_system table.

There is written 'energy\_id' as Primary key. 'description' column shows the informations of the Energy system. Some of the occupants have same model of the energy system.

'bms\_id' is a Foreign key.

	energy_id [PK] integer	description text	bms_id integer
1	10001	elevators, heating, emergency lighting, turnstiles and hatches	1001
2	10002	elevators, operating costs, heating	1002
3	10003	elevators, emergency lighting, turnstiles and hatches	1003
4	10004	elevators, lighting, equipment failure Notification	1004
5	10005	elevators, turnstiles and hatches, operating costs	1005
6	10006	elevators, heating, emergency lighting, turnstiles and hatches	1006
7	10007	elevators, heating, operating costs	1007
8	10008	elevators, turnstiles and hatches, heating	1008
9	10009	elevators, heating, emergency lighting	1009
10	10010	elevators, turnstiles and hatches, operating costs	1010

The codes for above table:

```

INSERT INTO Energy_system (Energy_id, Description, BMS_id)
VALUES (10001, 'elevators, heating, emergency lighting, turnstiles and hatches', 1001);
INSERT INTO Energy_system (Energy_id, Description, BMS_id)
VALUES (10002, 'elevators, operating costs, heating', 1002);
INSERT INTO Energy_system (Energy_id, Description, BMS_id)
VALUES (10003, 'elevators, emergency lighting, turnstiles and hatches ', 1003);
INSERT INTO Energy_system (Energy_id, Description, BMS_id)
VALUES (10004, 'elevators, lighting, equipment failure Notification', 1004);
INSERT INTO Energy_system (Energy_id, Description, BMS_id)
VALUES (10005, 'elevators, turnstiles and hatches, operating costs', 1005);
INSERT INTO Energy_system (Energy_id, Description, BMS_id)
VALUES (10006, 'elevators, heating, emergency lighting, turnstiles and hatches', 1006);
INSERT INTO Energy_system (Energy_id, Description, BMS_id)
VALUES (10007, 'elevators, heating, operating costs', 1007);
INSERT INTO Energy_system (Energy_id, Description, BMS_id)
VALUES (10008, 'elevators, turnstiles and hatches, heating', 1008);
INSERT INTO Energy_system (Energy_id, Description, BMS_id)
VALUES (10009, 'elevators, heating, emergency lighting', 1009);
INSERT INTO Energy_system (Energy_id, Description, BMS_id)
VALUES (10010, 'elevators, turnstiles and hatches, operating costs', 1010);

```

## 10. Security\_system table.

There is written 'security\_id' as Primary key. 'description' column shows the informations of the security system. Some of the occupants have same model of the security system. 'bms\_id' is a Foreign key.

	security_id [PK] integer	description text	bms_id integer
1	10001	security cameras, security, alarm system	1001
2	10002	wireless security camera, face recognition, alarm system	1002
3	10003	security cameras, retinal recognition, armed guards	1003
4	10004	video surveillance systems, face recognition, security, imitation of the presence of owners, alarm system	1004
5	10005	security cameras, palm recognition, alarm, security	1005
6	10006	wireless security camera, palm recognition, alarm system, security, automatic lighting of the territory in case of penetration	1006
7	10007	video surveillance systems, voice recognition, security shutters, armed guards, alarm systems	1007
8	10008	video surveillance, automatic illumination of the territory in case of penetration, access card recognition, armed security,	1008
9	10009	video surveillance systems, face recognition, security, imitation of the presence of owners, alarm system	1009
10	10010	wireless security camera, face recognition, alarm system	1010

The codes for the above table:

```

INSERT INTO Security_system (Security_id, Description, BMS_id)
VALUES (10001, 'security cameras, security, alarm system', 1001);
INSERT INTO Security_system (Security_id, Description, BMS_id)
VALUES (10002, 'wireless security camera, face recognition, alarm system', 1002);
INSERT INTO Security_system (Security_id, Description, BMS_id)
VALUES (10003, 'security cameras, retinal recognition, armed guards', 1003);
INSERT INTO Security_system (Security_id, Description, BMS_id)
VALUES (10004, 'video surveillance systems, face recognition, security, imitation of the presence of owners, alarm system', 1004);
INSERT INTO Security_system (Security_id, Description, BMS_id)
VALUES (10005, 'security cameras, palm recognition, alarm, security', 1005);
INSERT INTO Security_system (Security_id, Description, BMS_id)
VALUES (10006, 'wireless security camera, palm recognition, alarm system, security, automatic lighting of the territory in case of penetration', 1006);
INSERT INTO Security_system (Security_id, Description, BMS_id)
VALUES (10007, 'video surveillance systems, voice recognition, security shutters, armed guards, alarm systems', 1007);
INSERT INTO Security_system (Security_id, Description, BMS_id)
VALUES (10008, 'video surveillance, automatic illumination of the territory in case of penetration, access card recognition, armed security,', 1008);
INSERT INTO Security_system (Security_id, Description, BMS_id)
VALUES (10009, 'video surveillance systems, face recognition, security, imitation of the presence of owners, alarm system', 1009);
INSERT INTO Security_system (Security_id, Description, BMS_id)
VALUES (10010, 'wireless security camera, face recognition, alarm system', 1010);

```

## 11. Sprinkler\_system table.

There is written 'sprinkler\_id' as Primary key. 'description' column shows the informations of the sprinkler system. Some of the occupants have same model of the sprinkler system. 'bms\_id' is a Foreign key.

	sprinkler_id [PK] integer	description text	bms_id integer
1	10001	fire hydrant system, layflat fire hose, booster pump set, fire sprinkler	1001
2	10002	hydrant, layflat fire hose, fire detection system, breaching inlet	1002
3	10003	pipework & valves, fire sprinkler, fire hydrant system	1003
4	10004	layflat fire hose, hydrant, breaching inlet	1004
5	10005	fire brigade booster, fire detection system, fire hydrant system	1005
6	10006	pipework & valves, layflat fire hose, fire hydrant system	1006
7	10007	layflat fire hose, fire brigade booster, breaching inlet	1007
8	10008	fire hydrant system, layflat fire hose, booster pump set, fire sprinkler	1008
9	10009	booster pump set, fire sprinkler, fire hydrant system	1009
10	10010	hydrant, layflat fire hose, fire detection system, breaching inlet	1010

The codes for the above table:

```
INSERT INTO Sprinkler_system (Sprinkler_id, Description, BMS_id)
VALUES (10001, 'fire hydrant system, layflat fire hose, booster pump set, fire sprinkler', 1001);
INSERT INTO Sprinkler_system (Sprinkler_id, Description, BMS_id)
VALUES (10002, 'hydrant, layflat fire hose, fire detection system, breaching inlet', 1002);
INSERT INTO Sprinkler_system (Sprinkler_id, Description, BMS_id)
VALUES (10003, 'pipework & valves, fire sprinkler, fire hydrant system', 1003);
INSERT INTO Sprinkler_system (Sprinkler_id, Description, BMS_id)
VALUES (10004, 'layflat fire hose, hydrant, breaching inlet', 1004);
INSERT INTO Sprinkler_system (Sprinkler_id, Description, BMS_id)
VALUES (10005, 'fire brigade booster, fire detection system, fire hydrant system', 1005);
INSERT INTO Sprinkler_system (Sprinkler_id, Description, BMS_id)
VALUES (10006, 'pipework & valves, layflat fire hose, fire hydrant system', 1006);
INSERT INTO Sprinkler_system (Sprinkler_id, Description, BMS_id)
VALUES (10007, 'layflat fire hose, fire brigade booster, breaching inlet', 1007);
INSERT INTO Sprinkler_system (Sprinkler_id, Description, BMS_id)
VALUES (10008, 'fire hydrant system, layflat fire hose, booster pump set, fire sprinkler', 1008);
INSERT INTO Sprinkler_system (Sprinkler_id, Description, BMS_id)
VALUES (10009, 'booster pump set, fire sprinkler, fire hydrant system', 1009);
INSERT INTO Sprinkler_system (Sprinkler_id, Description, BMS_id)
VALUES (10010, 'hydrant, layflat fire hose, fire detection system, breaching inlet', 1010);
```

## CASE STUDY/RESULTS

### 5 ALTER TABLES:

```
ALTER TABLE building_company ADD COLUMN establishment_year INT;
ALTER TABLE building_company DROP COLUMN establishment_year;
ALTER TABLE occupant alter fname SET NOT NULL;
ALTER TABLE utility_payment ADD PRIMARY KEY (Price);
ALTER TABLE utility_payment DROP CONSTRAINT utility_payment_pkey;
```

### 10 QUERIES:

1. Tables “Occupant” and “Utility\_payment” are combined, in order to get the following information:

“How much does the person, who has been living for more than 5 years in the apartment, pay for the utility service?”

	occupant_id integer	fname character varying (64)	lname character varying (64)	date_of_birth date	check_in_date date	email character (500)	price integer	date_of_price date	living_duration interval
1	305	Oliver	Queen	1988-01-22	2010-01-28	greenarrow@gmail.c...	2560	2010-02-28	10 years 9 mons 23 days
2	306	Barry	Allen	1991-07-07	2012-12-12	flash@gmail.com	2660	2013-01-12	7 years 11 mons 8 days
3	307	Jeff	Bezos	1964-01-12	2007-04-01	amazon@info.com	1000000	2023-12-22	13 years 7 mons 19 days
4	308	Bill	Gates	1055-10-28	2009-02-28	gates@gmail.com	2860	2009-03-28	11 years 8 mons 20 days

To get above SQL table, there are used **JOIN** operator with **DML** statements:

**FULL OUTER JOIN** operator is used to connect tables with their Foreign Keys. The living duration is calculated in the **WHERE** condition as 'the current date' - 'the date when the occupant settled to apartment'.

```
SELECT Occupant.occupant_id, Occupant.fname, Occupant.lname, Occupant.date_of_birth, Occupant.check_in_date,
Occupant.email, Utility_payment.price, Utility_payment.date_of_price,
age(Occupant.check_in_date)
AS Living_duration FROM Occupant FULL OUTER JOIN Utility_payment
ON Occupant.occupant_id = Utility_payment.occupant_id
WHERE DATE_PART('year', CURRENT_DATE::date) - DATE_PART('year', Occupant.check_in_date::date) > 5;
```

2. Finding the person who pays least amount of the price:

	fname character varying (64)	lname character varying (64)	price integer
1	Dias	Nurbergenov	2060

There is also used **JOIN** operator. To find the least price, **MIN** statement helps to output it.

```
SELECT fname, lname, price
FROM occupant
INNER JOIN utility_payment
ON occupant.occupant_id = utility_payment.occupant_id
WHERE utility_payment.price = (SELECT MIN(price) FROM utility_payment);
```

3. The average price of the utility service.

	average_price numeric
1	2510.00

There is used **AVG** operator with the **ROUND** one too. **ROUND** is used to get approximate number and there should be used the number inside of the brackets of the **ROUND** operator, in order to take n number after 0 in the approximation process. In this case, there is used number 2.

```
SELECT ROUND(AVG(price),2) AS Average price
FROM utility_payment;
```

4. Finding the sum of the utility payments and the sum of the total price of the BMS.

	sum_utility_payment_price bigint	sum_bms_total_price bigint
1	25100	78750

There is used **SUM** operator in order to get the sum of the all utility payments and all total price for BMS. They are called 'SUM\_UTILITY\_payment\_price' and 'SUM\_BMS\_total\_price' respectively. To do this, there is used AS operator. The **FULL OUTER JOIN** is used here to get the connection between utility\_payment and BMS tables via occupant table.

```
SELECT SUM(Utility_payment.price) AS SUM_Utility_payment_price,
SUM(BMS.total_price) AS SUM_BMS_total_price
FROM BMS
FULL OUTER JOIN Occupant ON BMS.occupant_id = Occupant.occupant_id
FULL OUTER JOIN Utility_payment ON Occupant.occupant_id = Utility_payment.occupant_id;
```

5. Finding the resident who pays maximum amount of the money to the BMS.

	fname character varying (64)	lname character varying (64)	total_price integer
1	Barry	Allen	10000

There is used **INNER JOIN** operator. Also, in order to get the maximum amount of the paid total price for the BMS, the **MAX** operator is used.

```
SELECT fname, lname, total_price
FROM occupant
INNER JOIN BMS
ON occupant.occupant_id = BMS.occupant_id
WHERE BMS.total_price = (SELECT MAX(total_price) FROM BMS);
```

6. Sorting the names of the residents. This is helpful for the companies orientate through the dates of the occupants by their names.

	fname character varying (64)	lname character varying (64)	date_of_birth date
1	Barry	Allen	1991-07-07
2	Bill	Gates	1055-10-28

As an example, here is used for the names where started with the 'B'. In order to get that result, there is used **LIKE** operator.



```
SELECT fname, lname, date_of_birth
FROM occupant
WHERE fname LIKE 'B%';
```

7. Showing the list of residents with their prices for utility. It is useful to answer following question: How much does this person have to pay for the utility?

	fname character varying (64)	lname character varying (64)	price integer
1	Dias	Nurbergenov	2060
2	Zhangir	Bayanov	2160
3	Cristiano	Ronaldo	2260
4	Lionel	Messi	2360
5	Sergio	Ramos	2460
6	Oliver	Queen	2560
7	Barry	Allen	2660
8	Jeff	Bezos	2760
9	Bill	Gates	2860
10	Mark	Zuckerberg	2960

There was used **INNER JOIN** operator to connect occupant table with the utility\_payment one.

```
SELECT fname, lname, price
FROM occupant
INNER JOIN utility_payment
ON occupant.occupant_id = utility_payment.occupant_id;
```

8. Presenting the expensive building companies which used higher total price for BMS.

	company_name character varying (64)	total_price integer
1	Modern Structure	8000
2	Center Circle Design-Build	9850
3	Builder Gorilla	10000
4	Pure Renovation Company	9500

There was used two conditions and **AND** operator.. One of them is about connecting the Foreign keys and other one outputs the total\_price which are higher than 7900 dollars.



```
SELECT Building_company.company_name, BMS.total_price
FROM Building_company
FULL OUTER JOIN Building ON Building_company.company_id = Building.company_id
FULL OUTER JOIN BMS ON BMS.building_id = Building.building_id
WHERE BMS.building_id= Building.building_id
AND BMS.total_price > 7999;
```

## 9. Finding the cheap paid prices.

	price integer
1	2060
2	2160
3	2260
4	2360
5	2460

There is used **OR** operator. It is used to output when one of the conditions is true. Here is there is no price which is higher than 10000 dollars, therefore the output the the prices which are less than 2500 dollars.

```
SELECT utility_payment.price
FROM utility_payment
WHERE utility_payment.price > 10000 OR utility_payment.price < 2500;
```

## 10. Outputting the descriptions of each services. It is helpful to find which models of the services occupant uses.

	security_description text	hvac_description text	energy_description text	water_description text	sprinkler_description text
1	security cameras, security alarm syst...	air intake, heat recovery, preheating, cooling	elevators, heating, emergency lighting, turnstil...	cleaning, storage and distribution of water for ...	fire hydrant system, layflat fire hose, booster pump set...
2	wireless security camera, face recog...	filtration, air heating, preheating, extractio...	elevators, operating costs, heating	water filtration, yard flushing, autonomous wat...	hydrant, layflat fire hose, fire detection system, breac...
3	security cameras, retinal recognition, ...	recirculation, air heating, aspiration, extrac...	elevators, emergency lighting, turnstiles and h...	yard washing, cleaning, swimming pool	pipework & valves, fire sprinkler, fire hydrant system
4	video surveillance systems, face reco...	air heating, regulation of the differential pr...	elevators, lighting, equipment failure notificati...	storage and distribution of water for apartmen...	layflat fire hose, hydrant, breaching inlet
5	security cameras, palm recognition, a...	heat recovery, aspiration, filtration, air heat...	elevators, turnstiles and hatches, operating co...	Coagulation, water filtration, swimming pool...	fire brigade booster, fire detection system, fire hydrant...
6	wireless security camera, palm recog...	aspiration, supply to a clean room, cooling	elevators, heating, emergency lighting, turnstil...	floculation, cleaning, water filtration	pipework & valves, layflat fire hose, fire hydrant system
7	video surveillance systems, voice rec...	filtering, air heating, preheating, the supply...	elevators, heating, operating costs	yard washing, cleaning, swimming pool	layflat fire hose, fire brigade booster, breaching inlet
8	video surveillance, automatic illumina...	air heating, regulation of the differential pr...	elevators, turnstiles and hatches, heating	water filtration, yard flushing, autonomous wist...	fire hydrant system, layflat fire hose, booster pump set...
9	video surveillance systems, face reco...	heat recovery, aspiration, filtration, air heat...	elevators, heating, emergency lighting	storage and distribution of water for apartmen...	booster pump set, fire sprinkler, fire hydrant system
10	wireless security camera, face recog...	regulation of the differential pressure bet...	elevators, turnstiles and hatches, operating co...	floculation, cleaning, water filtration	hydrant, layflat fire hose, fire detection system, breac...

There is used **FULL OUTER JOIN** to connect tables and also **AS** operator to rename the description.

```

SELECT Security_system.Description AS Security_Description,
HVAC_system.Description AS HVAC_Description,
Energy_system.Description AS Energy_Description,
Water_system.Description AS Water_Description,
Sprinkler_system.Description AS Sprinkler_Description
FROM BMS
FULL OUTER JOIN Security_system ON BMS.BMS_id = Security_system.BMS_id
FULL OUTER JOIN HVAC_system ON BMS.BMS_id = HVAC_system.BMS_id
FULL OUTER JOIN Energy_system ON BMS.BMS_id = Energy_system.BMS_id
FULL OUTER JOIN Water_system ON BMS.BMS_id = Water_system.BMS_id
FULL OUTER JOIN Sprinkler_system ON BMS.BMS_id = Sprinkler_system.BMS_id;

```

### 5 SUBQUERIES:

1. This command uses SINGLE ROW, outputs only one tuple, where the occupant's name is " Bill".

	fname character varying (64)	lname character varying (64)	email character (500)
1	Bill	Gates	gates@gmail.com ...

```

SELECT fname, lname, email
FROM occupant
WHERE occupant_id =
(SELECT occupant_id
FROM occupant
WHERE fname = 'Bill');

```

2. This code uses a SINGLE ROW, in which the command displays a occupant whose "price" is more than 2900

	price integer	date_of_price date	occupant_id integer
1	2960	2018-09-11	309

```
SELECT price, date_of_price, occupant_id
FROM utility_payment
WHERE occupant_id =
(SELECT occupant_id
FROM utility_payment
WHERE price > 2900);
```

- Here MULTIPLE ROW is used, that is, two or more tuples are output. This command displays a occupant whose apart ment\_number is greater than 56

	apartment_id [PK] integer	apartment_number integer
1	206	56
2	207	57
3	208	58
4	209	59

```
SELECT apartment.apartment_id, apartment.apartment_number
FROM apartment
WHERE apartment.apartment_id IN
(SELECT apartment.apartment_id FROM apartment
WHERE apartment.apartment_id > 205);
```

- Here MULTIPLE ROW is used. This command displays a apartment with apartment\_id is greater than 205

	fname character varying (64)	lname character varying (64)
1	Oliver	Queen
2	Barry	Allen
3	Jeff	Bezos
4	Bill	Gates
5	Mark	Zuckerberg

```

SELECT  occupant.fname, occupant.lname
FROM  occupant
WHERE  apartment_id IN
      (SELECT  apartment_id FROM  apartment
      WHERE  apartment_number > 54 );

```

5. This code uses MULTIPLE ROW, and the OR operator that helps us output the names of companies whose id is less than 3 and more than 7

	company_name character varying (64)
1	Modern Structure
2	Ace & Hammer Builders
3	Diamond Ridge Construction
4	Center Circle Design-Build
5	Builder Gorilla

```




SELECT  building_company.company_name
FROM  building_company
WHERE  company_id NOT IN
      (SELECT  company_id FROM  building
      WHERE  building_id < 3 OR building_id > 7 );

```

## DML STATEMENTS:




### 11 UPDATE statements:

1. The value of the 'company\_name' column is changed to 'NULL', where the 'company\_id' is equal to 107.

	 <b>company_id</b> [PK] integer 	<b>company_name</b> character varying (64) 
1	101	BiGroup
2	102	Trump Tower
3	103	Modern Structure
4	104	Ace & Hammer Builders
5	105	Diamond Ridge Construction
6	106	Center Circle Design-Build
7	108	Pure Renovation Company
8	109	Mod Guys Construction
9	110	Build It Brothers
10	107	[null]

```
UPDATE building_company
SET company_name = NULL
WHERE company_id = 107;
```

- The value of the 'company\_id' is changed to 'NULL', where the 'building\_id' is equal to 5.

	 <b>building_id</b> [PK] integer 	<b>company_id</b> integer 
1	1	101
2	2	102
3	3	103
4	4	104
5	6	106
6	7	107
7	8	108
8	9	109
9	10	110
10	5	[null]

```
UPDATE building
SET company_id = NULL
WHERE building_id = 5;
```

- The value of the 'apartment\_number' is changed to 100, where the 'apartment\_id' is equal to 209.

	apartment_id [PK] integer	apartment_number integer	building_id integer
1	200	50	1
2	201	51	2
3	202	52	3
4	203	53	4
5	204	54	5
6	205	55	6
7	206	56	7
8	207	57	8
9	208	58	9
10	209	100	10

```
UPDATE apartment
SET apartment_number = 100
WHERE apartment_id = 209;
```

- The values of the 'date\_of\_birth' and 'email' columns are changed to '1980-01-05' and '[TheBest@gmail.com](mailto:TheBest@gmail.com)' respectively, where the 'occupant\_id' is equal to 300.

	occupant_id [PK] integer	fname character varying (64)	lname character varying (64)	date_of_birth date	check_in_date date	email character (500)	apartment_id integer
1	301	Zhangir	Bayanov	2003-04-29	2020-10-22	bayanov@gmail.com...	201
2	302	Cristiano	Ronaldo	1985-02-05	2019-12-31	cristiano@gmail.co...	202
3	303	Lionel	Messi	1987-06-24	2015-03-21	leomessi@gmail.co...	203
4	304	Sergio	Ramos	1986-01-30	2017-10-08	sergio@gmail.com ...	204
5	305	Oliver	Queen	1988-01-22	2010-01-28	greenarrow@gmail.c...	205
6	306	Barry	Allen	1991-07-07	2012-12-12	flash@gmail.com ...	206
7	307	Jeff	Bezos	1964-01-12	2007-04-01	amazon@info.com ...	207
8	308	Bill	Gates	1055-10-28	2009-02-28	gates@gmail.com ...	208
9	309	Mark	Zuckerberg	1984-05-14	2018-08-11	facebook@support.c...	209
10	300	Dias	Nurbergenov	1980-01-05	2020-09-01	TheBest@gmail.com...	200

```
UPDATE occupant
SET date_of_birth = '1980-01-05', email = 'TheBest@gmail.com'
WHERE occupant_id = 300;
```

5. The 'price' and 'date\_of\_price' are changed to 1000000 and '2023-12-22' respectively, where the occupant\_id is equal to 307.

	price integer	date_of_price date	occupant_id integer
1	2060	2020-10-01	300
2	2160	2020-11-22	301
3	2260	2020-01-31	302
4	2360	2015-04-21	303
5	2460	2017-11-08	304
6	2560	2010-02-28	305
7	2660	2013-01-12	306
8	2860	2009-03-28	308
9	2960	2018-09-11	309
10	1000000	2023-12-22	307

```
UPDATE utility_payment
SET price = 1000000, date_of_price = '2023-12-22'
WHERE occupant_id = 307;
```

6. This code changes the value total\_price to 10000 in BMS table using the or operator, where BMS\_id is equal 1003 and 1005

	bms_id [PK] integer	total_price integer	building_id integer	occupant_id integer
1	1001	7500	1	300
2	1002	6000	2	301
3	1004	7900	4	303
4	1006	9850	6	305
5	1007	10000	7	306
6	1008	9500	8	307
7	1009	6000	9	308
8	1010	7500	10	309
9	1003	10000	3	302
10	1005	10000	5	304

```
UPDATE BMS SET total_price = 10000
WHERE BMS_id = 1003 OR BMS_id = 1005;
```

7. This code changes the value BMS\_id to 1007 in Security\_system table using the or operator, where security\_id is equal 10001 and 10009

	security_id [PK] integer	description text	bms_id integer
1	10002	wireless security camera, face recognition, alarm system	1002
2	10003	security cameras, retinal recognition, armed guards	1003
3	10004	video surveillance systems, face recognition, security, imitation of the pre...	1004
4	10005	security cameras, palm recognition, alarm, security	1005
5	10006	wireless security camera, palm recognition, alarm system, security, autom...	1006
6	10007	video surveillance systems, voice recognition, security shutters, armed gu...	1007
7	10008	video surveillance, automatic illumination of the territory in case of penetr...	1008
8	10010	wireless security camera, face recognition, alarm system	1010
9	10001	security cameras, security, alarm system	1007
10	10009	video surveillance systems, face recognition, security, imitation of the pre...	1007

```
UPDATE Security_system SET BMS_id = 1007
WHERE Security_id = 10009 OR Security_id = 10001;
```

8. This code changes the value description to NULL in HVAC\_system, where HVAC\_id is more than 10008

	hvac_id [PK] integer	description text	bms_id integer
1	10001	air intake, heat recovery, preheating, cooling	1001
2	10002	filtration, air heating, preheating, extractio	1002
3	10003	recirculation, air heating, aspiration, extraction, filtration	1003
4	10004	air heating, regulation of the differential pressure between rooms, aspirati...	1004
5	10005	heat recovery, aspiration, filtration, air heating, cooling	1005
6	10006	aspiration, supply to a clean room, cooling	1006
7	10007	filtering, air heating, preheating, the supply to the clean room	1007
8	10008	air heating, regulation of the differential pressure between rooms, filtration	1008
9	10009	heat recovery, aspiration, filtration, air heating, cooling	1009
10	10010	regulation of the differential pressure between rooms, aspiration, filtration...	[null]

```
UPDATE HVAC_system SET BMS_id = NULL
WHERE HVAC_id = 10010;
```

9. This code changes the value description to NULL in Energy\_system table using the or operator, where energy\_id is less than 10003 and more than 10008

	energy_id [PK] integer	description text	bms_id integer
1	10003	elevators, emergency lighting, turnstiles and hatches	1003
2	10004	elevators, lighting, equipment failure Notification	1004
3	10005	elevators, turnstiles and hatches, operating costs	1005
4	10006	elevators, heating, emergency lighting, turnstiles and hatches	1006
5	10007	elevators, heating, operating costs	1007
6	10008	elevators, turnstiles and hatches, heating	1008
7	10001	[null]	1001
8	10002	[null]	1002
9	10009	[null]	1009
10	10010	[null]	1010



```
UPDATE Energy_system SET Description = NULL
WHERE Energy_id > 10008 OR Energy_id < 10003 ;
```

10. This code changes the value to descriptions of 'water filtration, yard flushing, Autonomous watering of plants' in Water\_system table using the or operator, where water\_id is between 10003 and 10008

	water_id [PK] integer	description text	bms_id integer
1	10003	yard washing, cleaning, swimming pool	1003
2	10004	storage and distribution of water for apartments, yard washing, cleaning, ...	1004
3	10005	Coagulation, water filtration, swimming pool,	1005
4	10006	flocculation, cleaning, water filtration	1006
5	10007	yard washing, cleaning, swimming pool	1007
6	10008	water filtration, yard flushing, autonomous watering of plants	1008
7	10001	water filtration, yard flushing, autonomous watering of plants	1001
8	10002	water filtration, yard flushing, autonomous watering of plants	1002
9	10009	water filtration, yard flushing, autonomous watering of plants	1009
10	10010	water filtration, yard flushing, autonomous watering of plants	1010

```
UPDATE Water_system SET Description = 'water filtration, yard flushing, autonomous watering of plants'
WHERE Water_id > 10008 OR Water_id < 10003 ;
```

11. This code changes the Description value in the Sprinkler\_system table which has a sprinkler\_id equal of 10005

	sprinkler_id [PK] integer	description text	bms_id integer
1	10001	fire hydrant system, layflat fire hose, booster pump set, fire sprinkler	1001
2	10002	hydrant, layflat fire hose, fire detection system, breaching inlet	1002
3	10003	pipework & valves, fire sprinkler, fire hydrant system	1003
4	10004	layflat fire hose, hydrant, breaching inlet	1004
5	10006	pipework & valves, layflat fire hose, fire hydrant system	1006
6	10007	layflat fire hose, fire brigade booster, breaching inlet	1007
7	10008	fire hydrant system, layflat fire hose, booster pump set, fire sprinkler	1008
8	10009	booster pump set, fire sprinkler, fire hydrant system	1009
9	10010	hydrant, layflat fire hose, fire detection system, breaching inlet	1010
10	10005	hydrant, lots of hydrants	1005

```
UPDATE Sprinkler_system SET Description = 'hydrant, lots of hydrants'
WHERE Sprinkler_id = 10005 ;
```

## 6 DELETE statements:

1. This code deletes all tuples in the Splinker\_system Table that have a splinker\_id less than 10007

	<b>sprinkler_id</b> [PK] integer	<b>description</b> text	<b>bms_id</b> integer
1	10001	fire hydrant syst...	1001
2	10002	hydrant, layflat ...	1002
3	10003	pipework & valv...	1003
4	10004	layflat fire hose,...	1004
5	10006	pipework & valv...	1006
6	10007	layflat fire hose,...	1007
7	10005	hydrant, lots of ...	1005

```
DELETE FROM Sprinkler_system
WHERE Sprinkler_id > 10007;
```

- This code deletes all tuples in the Water\_system Table that have a description is to equal 'water filtration, yard flushing, autonomous watering of plants'.

	<b>water_id</b> [PK] integer	<b>description</b> text	<b>bms_id</b> integer
1	10003	yard washing, cl...	1003
2	10004	storage and dist...	1004
3	10005	Coagulation, wa...	1005
4	10006	flocculation, cle...	1006
5	10007	yard washing, cl...	1007

```
DELETE FROM Water_system
WHERE Description = 'water filtration, yard flushing, autonomous watering of plants';
```

- This code deletes all tuples in the Energy\_system Table that have a 'BMS\_id' is greater or equal than 1000.

	<b>energy_id</b> [PK] integer	<b>description</b> text	<b>bms_id</b> integer

```
DELETE FROM Energy_system
WHERE BMS_id >= 1000;
```

- This code deletes all tuples in the building\_company table that have an id is equal to 105.

	<b>company_id</b> [PK] integer	<b>company_name</b> character varying (64)
1	101	BiGroup
2	102	Trump Tower
3	103	Modern Structure
4	104	Ace & Hammer Builders
5	106	Center Circle Design-Build
6	108	Pure Renovation Company
7	109	Mod Guys Construction
8	110	Build It Brothers
9	107	[null]

```
DELETE FROM building_company
WHERE company_id = 105;
```

5. This command DELETE a tuple in the Security\_system table using OR, where Security\_id is less than or equal to 10002 and where It is equal to 10010

	<b>security_id</b> [PK] integer	<b>description</b> text	<b>bms_id</b> integer
1	10003	security cameras, retinal recognition, armed guards	1003
2	10004	video surveillance systems, face recognition, security, imitation of the pre...	1004
3	10005	security cameras, palm recognition, alarm, security	1005
4	10006	wireless security camera, palm recognition, alarm system, security, autom...	1006
5	10007	video surveillance systems, voice recognition, security shutters, armed gu...	1007
6	10008	video surveillance, automatic illumination of the territory in case of penetr...	1008
7	10009	video surveillance systems, face recognition, security, imitation of the pre...	1007

```
DELETE FROM Security_system
WHERE Security_id <= 10002 OR Security_id=10010;
```

6. This command DELETES a tuple in the HVAC\_system table where the HVAC\_id is greater than 10008

	hvac_id [PK] integer	description text	bms_id integer
1	10001	air intake, heat recovery, preheating, cooling	1001
2	10002	filtration, air heating, preheating, extractio	1002
3	10003	recirculation, air heating, aspiration, extraction, filtration	1003
4	10004	air heating, regulation of the differential pressure between rooms, aspirati...	1004
5	10005	heat recovery, aspiration, filtration, air heating, cooling	1005
6	10006	aspiration, supply to a clean room, cooling	1006
7	10007	filtering, air heating, preheating, the supply to the clean room	1007
8	10008	air heating, regulation of the differential pressure between rooms, filtration	1008

```
DELETE FROM HVAC_system
WHERE HVAC_id > 10008;
```

Here is the whole codes with Entity Relationship Diagram are on the following link on GitHub: <https://github.com/Dias007/ICT-Project>

## CONCLUSION

### Positive:

- The team gained experience in creating an example of an existing BMS.
- The team has learned a lot of interesting things. How do homes that use BMS work, and what are the disadvantages of those homes that do not have this system
- Students have learned to work as a team on a project that will undoubtedly be very useful in the future

### Minuses:

- Lack of information about professional work on the BMS (terms, videos, etc.).
- Given the weekend, there was not enough time for this project.

### Results:

The result of this project is an approximate representation of the essence and functionality of the BMS. But this project cannot be called a fully functional BMS.

### Future of the project:

If you make a few dozen adjustments to this work in the future, it may become a fully functional but simple BMS in the future. But to achieve this result, you need a lot of experience in programming and knowledge of how buildings work at an acceptable level.

Since BMS has a lot of hidden details. For example, if the programmer doesn't know how the building's ventilation system works, they can go bad, and this can lead to an accident.

## REFERENCES:

- Joseph, J. (2020). Building Management System - an overview | ScienceDirect Topics. Retrieved 20 November 2020, from <https://www.sciencedirect.com/topics/engineering/building-management-system>
- J.Knibbe, E. (2020). US5565855A - Building management system - Google Patents. Retrieved 20 November 2020, from <https://patents.google.com/patent/US5565855A/en>
- J. Thomas, R., A. Anderson, N., G. Donaldson, S., & A. Behar, M. (2020). US7567844B2 - Building management system - Google Patents. Retrieved 20 November 2020, from <https://patents.google.com/patent/US7567844B2/en>
- Suzuki, M., & Yagishita, M. (2020). US4918615A - Building management system - Google Patents. Retrieved 20 November 2020, from <https://patents.google.com/patent/US4918615A/en>
- JUNG, H., YOUN, S., & JEON, D. (2020). US20080195687A1 - Building management system and method - Google Patents. Retrieved 20 November 2020, from <https://patents.google.com/patent/US20080195687A1/en>