finis
Universidad Finis Terrae
Facultad de Ingeniería

Laboratorio 2 - Base de Datos
“Implementación de Base de Datos en SQL Server y MySQL: Corredor de propiedades”

Asignatura:
Base de Datos

Profesores:
Carolina Del Carmen Zambrano Matamala

Autores:
Miguel Cornejo
Nicolás Arellano
Benjamín Sepúlveda Labbe
Guillermo Guiñez

Fecha de entrega:
27/10/2024

Grupo:
5

Índice

1. Introducción

En este laboratorio se tiene como objetivo presentar el desarrollo, diseño e implementación de una base de datos relacional específicamente hecha para completar las necesidades de un corredor de propiedades. El sistema busca optimizar la gestión de información relacionada con las propiedades, los arrendamientos, los propietarios, las reparaciones necesarias en las propiedades, los pagos efectuados y pendientes, así como otros elementos esenciales en el trabajo diario de un negocio inmobiliario. La base de datos fue inicialmente diseñada en Microsoft Access, para luego ser integrada y optimizada en motores de bases de datos como SQL Server y MySQL, con el fin de ofrecer mayor escalabilidad y mejor manejo de consultas complejas, aspectos necesarios para un negocio que maneja un gran volumen creciente de datos y que requiere una estructura flexible y sólida para su crecimiento.

El sistema desarrollado no solo tiene la necesidad de gestionar información sobre las propiedades disponibles para arrendar o vender, sino que también permite un seguimiento detallado de los propietarios y arrendatarios asociados a cada inmueble. Por otro lado, incluye la gestión de los pagos, permitiendo al corredor tener un control preciso sobre los ingresos recibidos por concepto de arrendamientos, así como los pagos pendientes. Este aspecto es fundamental para evitar errores financieros y mejorar la previsión de ingresos. Asimismo, el sistema permite gestionar las reparaciones necesarias en cada propiedad, haciendo fácil la comunicación entre propietarios, arrendatarios y proveedores de servicios, y asegurando que las solicitudes de mantenimiento se gestionan de manera eficiente y en el menor tiempo posible.

El problema que se busca resolver con el diseño de la base de datos es la gestión más eficiente de propiedades, propietarios, reparaciones, visitas y arriendos dentro de una empresa de bienes raíces. El objetivo es organizar y almacenar de manera estructurada la información importante de cada propiedad, su historial de reparaciones, los propietarios que los nuevos que han llegado, y el registro de visitas realizadas a cada propiedad. Esto logrará que se pueda acceder a la información rápidamente, realizar consultas y obtener información sobre el estado de las propiedades y sus propietarios.

2. Código de SQL Server y MySQL

El desarrollo de los código se basaron en la implementación de la solución de Microsoft Access, gracias a la ventaja de tener sus modelos de esquema conceptual entidad relación y su modelo relacional, el cual su diagrama esquema Relacional es la siguiente:

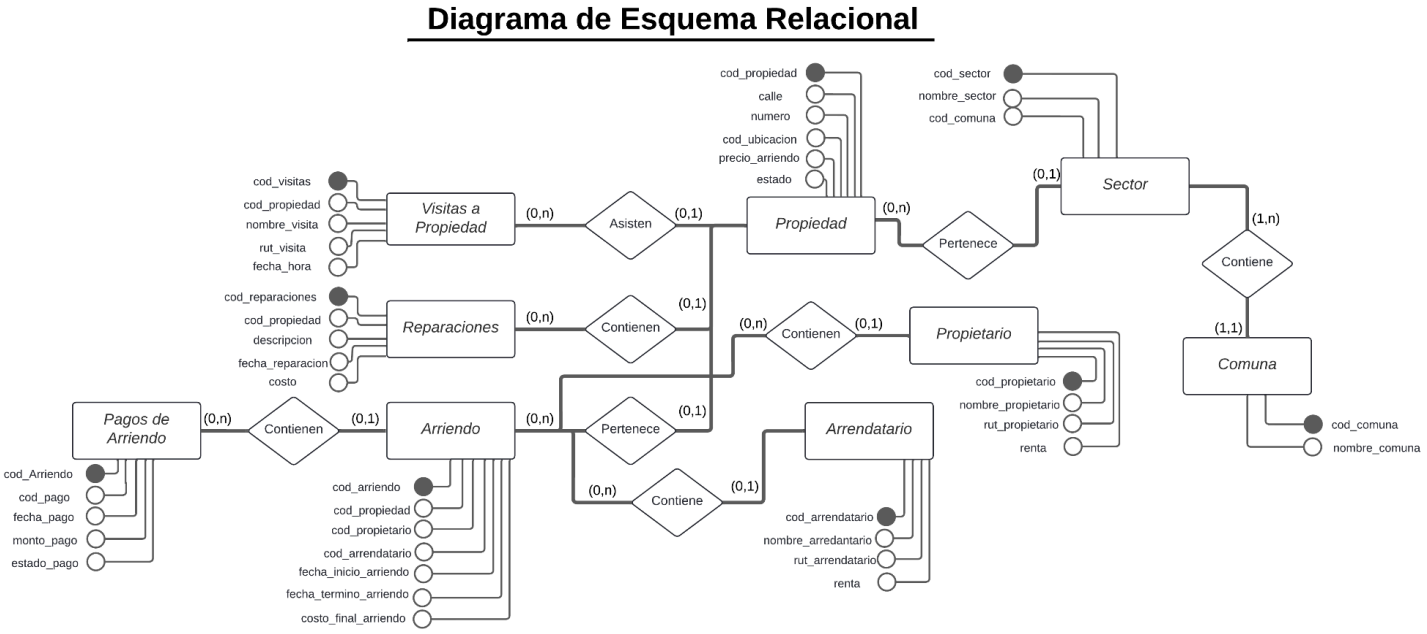


Figura 1: "Esquema Conceptual Entidades Relación"

2.1 SQL Server Microsoft

2.1.1 Creación de base de datos:

Primero conectarse a una instancia del motor de base de datos de SQL Server. Luego en el Explorador de objetos, crea un *New Query*, luego escribir el siguiente código para crear y usar la nueva base de datos *Corredor_de_Propiedades*:

```
1  CREATE DATABASE Corredor_de_Propiedades
2  Use Corredor_de_Propiedades;
```

Figura 2: Código de SQL Server "Creación de base de datos"

2.1.2 Creación de Tablas:

A continuación se muestra el siguiente código el cual se crean las tablas de la base de datos con sus respectivas caracterización de **atributos,** los cuales no pueden quedarse con valores nulos o

vacíos, además de con una restricción de *CHECK* en cada tabla y ya definidas las variables crear las **llaves primarias** y **foráneas**.

```
1  -- Comuna --
2  CREATE TABLE COMUNA(
3     cod_comuna INT NOT NULL,
4     nombre_comuna VARCHAR(25) NOT NULL,
5     PRIMARY KEY(cod_comuna),
6     CHECK (LEN(nombre_comuna) > 9) -- Nombre_comuna debe tener mayor a 10 caracteres de largo (Cerro Navia)
7  );
```

Figura 3: Código de SQL Server "Creación de tabla Comuna"

La tabla *COMUNA* se define con dos campos principales: *cod_comuna*, que es un entero no nulo y funciona como clave primaria, y *nombre_comuna*, una cadena de texto de hasta 25 caracteres que también es no nula. Adicionalmente, se establece una restricción de validación mediante *CHECK*, que asegura que el nombre de la comuna debe tener más de 9 caracteres, garantizando que los nombres cortos no sean permitidos. Este diseño asegura la integridad de los datos relacionados con las comunas, evitando duplicados y errores en los nombres.

```
1  -- Sector --
2  CREATE TABLE SECTOR(
3     cod_sector INT NOT NULL,
4     nombre_sector VARCHAR(30) NOT NULL,
5     cod_comuna INT NOT NULL,
6     PRIMARY KEY(cod_sector),
7           FOREIGN KEY(cod_comuna) REFERENCES COMUNA(cod_comuna),
8     CHECK (LEN(nombre_sector) > 5) -- Nombre_sector debe ser mayor a 5 caracteres de largo (La Vega)
9  );
```

Figura 4: Código de SQL Server "Creación de tabla Sector*"*

La tabla *SECTOR* está compuesta por tres campos principales: *cod_sector*, un entero no nulo que actúa como clave primaria, *nombre_sector*, una cadena de texto de hasta 30 caracteres que es obligatoria, y *cod_comuna*, que es una referencia externa a la tabla *COMUNA* mediante la clave foránea *cod_comuna*. Además, se implementa una restricción *CHECK* que asegura que el nombre del sector tenga más de 5 caracteres, garantizando que los nombres cortos no sean aceptados. Esta estructura asegura la relación entre los sectores y las comunas, manteniendo la integridad referencial y la validez de los datos ingresados.

```
1  -- Propiedad --
2  CREATE TABLE PROPIEDAD(
3     cod_propiedad INT NOT NULL,
4     calle VARCHAR(50) NOT NULL,
5     numero INT NOT NULL,
6     precio_arriendo DECIMAL(10,2) NOT NULL CHECK (precio_arriendo > 50000), -- Precio debe ser mayor a 0 ej:
7  5,900,000,000.00
8     estado VARCHAR(20) NOT NULL CHECK (estado IN ('Disponible', 'Arrendada')), -- Estado debe ser 'Disponible' o
9  'Arrendada'
10    cod_ubicacion INT NOT NULL,
11    PRIMARY KEY(cod_propiedad),
```

```
12    FOREIGN KEY(cod_ubicacion) REFERENCES SECTOR(cod_sector)
13 );
```

Figura 5: Código de SQL Server "Creación de tabla Propiedad"

La tabla ***PROPIEDAD*** contiene información sobre propiedades con los siguientes campos: *cod_propiedad*, un entero no nulo que actúa como clave primaria; *calle*, una cadena de texto de hasta 50 caracteres, y *número*, un valor entero, ambos campos son obligatorios. El campo *precio_arriendo* almacena el precio de arriendo en formato decimal, con una restricción que asegura que sea mayor a 50000, lo que garantiza que no se ingresen valores bajos. Además, el campo *estado* define si la propiedad está 'Disponible' o 'Arrendada', restringido a estos dos valores mediante *CHECK*. Finalmente, el campo *cod_ubicacion* es una clave foránea que hace referencia a la tabla ***SECTOR***, vinculando cada propiedad con su sector correspondiente. Este diseño asegura la integridad de los datos relacionados con las propiedades, su ubicación, y el estado de arriendo.

```
1  -- Arrendatario --
2  CREATE TABLE ARRENDATARIO (
3     cod_arrendatario INT NOT NULL,
4     nombre_arrendatario VARCHAR(30) NOT NULL,
5     rut_arrendatario VARCHAR(12) NOT NULL,
6     renta DECIMAL(10,2) NOT NULL CHECK (renta > 50000), -- La renta debe ser mayor a 50000
7     PRIMARY KEY(cod_arrendatario),
8     CHECK (LEN(rut_arrendatario) = 12) -- Asegurando que el RUT tenga 12 caracteres
9  );
```

Figura 6: Código de SQL Server "Creación de tabla Arrendatario *"*

La tabla ***ARRENDATARIO*** almacena información sobre los arrendatarios con los siguientes campos: *cod_arrendatario*, un entero no nulo que actúa como clave primaria; *nombre_arrendatario*, que es una cadena de texto de hasta 30 caracteres y es obligatoria; *rut_arrendatario*, un valor de 12 caracteres que identifica de manera única al arrendatario, con una restricción que garantiza su longitud exacta. Además, el campo *renta* almacena el ingreso del arrendatario en formato decimal, con una restricción que asegura que la renta sea superior a 50000. Estas restricciones y claves aseguran que los datos de los arrendatarios sean precisos y consistentes, especialmente en lo que respecta a la identificación y el ingreso mínimo requerido.

```
1  -- Propietario --
2  CREATE TABLE PROPIETARIO(
3     cod_propietario INT NOT NULL,
4     nombre_propietario VARCHAR(30) NOT NULL,
5     rut_propietario VARCHAR(12) UNIQUE NOT NULL, -- RUT único
6     renta DECIMAL(10,2) NOT NULL CHECK (renta > 50000), -- La renta debe ser mayor a 50000
7     PRIMARY KEY(cod_propietario),
8     CHECK (LEN(rut_propietario) = 12) -- Verificar RUT tenga 12 caracteres
9  );
```

Figura 7: Código de SQL Server "Creación de tabla Propietario"

La tabla ***PROPIETARIO*** está diseñada para almacenar información de los propietarios con los siguientes campos: *cod_propietario*, un entero no nulo que actúa como clave primaria; *nombre_propietario*, una cadena de texto de hasta 30 caracteres y obligatoria; y *rut_propietario*,

una cadena única de 12 caracteres que identifica al propietario de manera exclusiva, con una restricción que asegura la longitud exacta de este campo. Además, el campo *renta* almacena el ingreso del propietario en formato decimal, con una restricción que garantiza que sea superior a 50000. Este diseño asegura la consistencia y unicidad en los datos de los propietarios, validando tanto el formato del rut como el ingreso mínimo requerido.

```
1  -- Visitas de propiedad --
2  CREATE TABLE VISITAS_PROPIEDAD(
3     cod_visitas INT NOT NULL,
4     cod_propiedad INT NOT NULL,
5     nombre_visita VARCHAR(25) NOT NULL,
6     rut_visita VARCHAR(12) NOT NULL, -- 12.345.678-9
7     fecha_hora DATETIME NOT NULL, -- Fechas y horas combinadas
8     PRIMARY KEY(cod_visitas),
9     FOREIGN KEY (cod_propiedad) REFERENCES PROPIEDAD(cod_propiedad),
10    CHECK (LEN(rut_visita) = 12) -- Verificar RUT tenga 12 caracteres
11 );
```

Figura 8: Código de SQL Server "Creación de tabla Visitas de propiedad"

La tabla *VISITAS_PROPIEDAD* registra las visitas realizadas a las propiedades. Contiene los siguientes campos: *cod_visitas*, un entero no nulo que actúa como clave primaria; *cod_propiedad*, un campo entero que referencia la propiedad visitada mediante una clave foránea a la tabla PROPIEDAD; *nombre_visita*, una cadena de hasta 25 caracteres que almacena el nombre de la persona que realiza la visita; y *rut_visita*, una cadena de 12 caracteres que contiene el RUT del visitante, con una restricción que asegura su longitud. Además, el campo *fecha_hora* registra la fecha y hora de la visita de manera combinada. Las restricciones aseguran la integridad de los datos, validando tanto la referencia a la propiedad como la correcta estructura del rut.

```
1  -- Reparaciones --
2  CREATE TABLE REPARACIONES(
3     cod_reparaciones INT NOT NULL,
4     descripciones VARCHAR(100) NOT NULL,
5     fecha_reparacion DATE NOT NULL,
6     costo DECIMAL(10,2) NOT NULL CHECK (costo >= 50000),
7     cod_propiedad INT NOT NULL,
8     PRIMARY KEY(cod_reparaciones),
9     FOREIGN KEY(cod_propiedad) REFERENCES PROPIEDAD(cod_propiedad)
10 );
```

Figura 9: Código de SQL Server "Creación de tabla Reparaciones "

La tabla *REPARACIONES* almacena los detalles de las reparaciones realizadas en las propiedades. Está compuesta por los siguientes campos: *cod_reparaciones*, un entero no nulo que actúa como clave primaria; *descripciones*, una cadena de hasta 100 caracteres que detalla la naturaleza de la reparación; *fecha_reparacion*, que registra la fecha en que se realizó la reparación; y *costo*, que es un valor decimal que debe ser igual o superior a 50000, garantizado por una restricción *CHECK*. El campo *cod_propiedad* es una clave foránea que vincula la reparación con la tabla *PROPIEDAD*, asegurando que cada reparación esté asociada a una propiedad específica. Esta estructura asegura la integridad y exactitud de los datos, permitiendo registrar costos, descripciones y fechas de reparación de manera precisa.

```
1   -- Arriendo --
2   CREATE TABLE ARRIENDO(
3      cod_arriendo INT NOT NULL,
4      fecha_inicio_arriendo DATE NOT NULL,
5      fecha_termino_arriendo DATE NOT NULL,
6      costo_final_arriendo DECIMAL(10,2) NOT NULL CHECK (costo_final_arriendo > 50000),
7      cod_propiedad INT NOT NULL,
8      cod_propietario INT NOT NULL,
9      cod_arrendatario INT NOT NULL,
10     PRIMARY KEY(cod_arriendo),
11     FOREIGN KEY (cod_propiedad) REFERENCES PROPIEDAD(cod_propiedad),
12     FOREIGN KEY (cod_propietario) REFERENCES PROPIETARIO(cod_propietario),
13     FOREIGN KEY (cod_arrendatario) REFERENCES ARRENDATARIO(cod_arrendatario)
14  );
```

Figura 10: Código de SQL Server "Creación de tabla Arriendo *"*

La tabla *ARRIENDO* almacena la información sobre los contratos de arriendo. Está compuesta por los siguientes campos: *cod_arriendo*, un entero que actúa como clave primaria; *fecha_inicio_arriendo* y *fecha_termino_arriendo*, que registran las fechas de inicio y término del arriendo respectivamente; *costo_final_arriendo*, un valor decimal que debe ser mayor a 50000, asegurado mediante una restricción *CHECK*. Además, la tabla incluye tres claves foráneas: *cod_propiedad*, que referencia la tabla *PROPIEDAD*; *cod_propietario*, que se refiere a la tabla *PROPIETARIO*; y *cod_arrendatario*, que vincula al arrendatario de la tabla *ARRENDATARIO*. Esta estructura asegura que cada arriendo esté relacionado correctamente con una propiedad, un propietario y un arrendatario, manteniendo la integridad de los datos en el sistema.

```
1   -- Pagos de arriendo --
2   CREATE TABLE PAGOS_ARRIENDO (
3      cod_pago INT NOT NULL,
4      cod_arriendo INT NOT NULL,
5      fecha_pago DATE NOT NULL,
6      monto_pago DECIMAL(10,2) NOT NULL CHECK (monto_pago > 50000),
7      estado_pago VARCHAR(20) NOT NULL CHECK (estado_pago IN ('Pagado', 'Pendiente')),
8      PRIMARY KEY(cod_pago),
9      FOREIGN KEY (cod_arriendo) REFERENCES ARRIENDO(cod_arriendo)
10  );
```

Figura 11: Código de SQL Server "Creación de tabla Pagos de arriendo"

La tabla *PAGOS_ARRIENDO* registra los pagos realizados por los arriendos. Sus campos incluyen *cod_pago*, un entero no nulo que actúa como clave primaria; *cod_arriendo*, un entero que referencia la tabla *ARRIENDO* mediante una clave foránea; *fecha_pago*, que registra la fecha en la que se realizó el pago; y *monto_pago*, un valor decimal que debe ser superior a 50000, validado mediante una restricción *CHECK*. El campo *estado_pago* permite definir si el pago está 'Pagado' o 'Pendiente', con una restricción que garantiza que solo se utilicen estos dos valores. Esta estructura asegura la correcta relación entre los pagos y los arriendos, proporcionando un control detallado sobre los montos y el estado de los pagos.

2.1.3 Inserción de datos:

Una vez obtenida las tablas, se requiere rellenar de información relevante para la complementación del funcionamiento de la base de datos, con sus respectivos datos ingresados a través del código SQL, de la siguiente manera:

```
1  --COMUNA--
2  INSERT INTO COMUNA (cod_comuna, nombre_comuna)
3  VALUES
4  (11, 'Cerro Navia'),
5  (22, 'La Florida'),
6  (33, 'Las Condes'),
7  (44, 'Lo Barnechea'),
8  (55, 'Providencia');
9
10 --SECTOR--
11 INSERT INTO SECTOR (cod_sector, nombre_sector, cod_comuna)
12 VALUES
13 (10, 'Sector Central', 11),
14 (20, 'Sector Norte', 22),
15 (30, 'Sector Sur', 33),
16 (40, 'Sector Oriente', 44),
17 (50, 'Sector Poniente', 55),
18 (60, 'La Vega Chica', 11),
19 (70, 'Los Jardines', 33),
20 (80, 'Villa Hermosa', 44),
21 (90, 'El Mirador', 55);
22
23 --PROPIEDAD--
24 INSERT INTO PROPIEDAD (cod_propiedad, calle, numero, precio_arriendo, estado, cod_ubicacion)
25 VALUES
26 (12, 'Av. Las Torres', 101, 800000.00, 'Disponible', 10),
27 (23, 'Calle Independencia', 202, 950000.00, 'Arrendada', 20),
28 (34, 'Av. Providencia', 303, 1200000.00, 'Disponible', 30),
29 (45, 'Calle Los Pinos', 404, 650000.00, 'Arrendada', 40),
30 (56, 'Calle Irarrazabal', 505, 750000.00, 'Disponible', 50),
31 (67, 'Calle Bellavista', 606, 820000.00, 'Disponible', 60);
32
33 --ARRENDATARIO--
34 INSERT INTO ARRENDATARIO (cod_arrendatario, nombre_arrendatario, rut_arrendatario, renta)
35 VALUES
36 (13, 'Juan Perez', '12.345.678-9', 1500000.00),
37 (24, 'Carla Reyes', '98.765.432-1', 1300000.00),
38 (35, 'Pedro Gomez', '11.111.111-1', 1400000.00),
39 (46, 'Ana Diaz', '22.222.222-2', 1700000.00),
40 (57, 'Luis Lopez', '33.333.333-3', 1600000.00),
41 (68, 'Miguel Sanchez', '44.444.444-4', 1550000.00);
42
43 --PROPIETARIO--
44 INSERT INTO PROPIETARIO (cod_propietario, nombre_propietario, rut_propietario, renta)
45 VALUES
46 (14, 'Carlos Gonzalez', '44.444.444-4', 2000000.00),
47 (25, 'Maria Silva', '55.555.555-5', 2100000.00),
48 (36, 'Jorge Salinas', '66.666.666-6', 1900000.00),
```

```
(47, 'Sofia Mendez', '77.777.777-7', 2200000.00),
(58, 'Patricia Vega', '88.888.888-8', 1800000.00),
(69, 'Alberto Vargas', '99.999.999-9', 2300000.00);

--VISITAS_PROPIEDAD--
INSERT INTO VISITAS_PROPIEDAD (cod_visitas, cod_propiedad, nombre_visita, rut_visita, fecha_hora)
VALUES
(15, 12, 'Miguel Fernandez', '99.999.999-9', '2024-10-10 14:30:00'),
(26, 23, 'Jose Gonzalez', '12.312.312-3', '2024-10-11 09:45:00'),
(37, 34, 'Paula Nuñez', '98.798.798-7', '2024-10-12 11:15:00'),
(48, 45, 'Ricardo Fuentes', '32.132.132-1', '2024-10-13 15:00:00'),
(59, 56, 'Lucia Morales', '21.321.321-3', '2024-10-14 13:00:00'),
(60, 67, 'Felipe Cruz', '33.344.455-6', '2024-10-15 16:00:00');

--REPARACIONES--
INSERT INTO REPARACIONES (cod_reparaciones, descripciones, fecha_reparacion, costo, cod_propiedad)
VALUES
(16, 'Falla en las tuberías del baño', '2024-09-10', 100000.00, 12),
(27, 'Reparación del portón frontal', '2024-09-15', 150000.00, 23),
(38, 'Obstrucción en las tuberías del gas', '2024-09-20', 120000.00, 34),
(49, 'Arreglo eléctrico del medidor', '2024-09-25', 200000.00, 45),
(60, 'Reparacion de ventanas', '2024-09-30', 80000.00, 56),
(61, 'Cambio de pisos', '2024-10-05', 50000.00, 67);

--ARRIENDO--
INSERT INTO ARRIENDO (cod_arriendo, fecha_inicio_arriendo, fecha_termino_arriendo, costo_final_arriendo,
cod_propiedad, cod_propietario, cod_arrendatario)
VALUES
(17, '2024-10-01', '2025-10-01', 950000.00, 12, 14, 13),
(28, '2024-09-01', '2025-09-01', 900000.00, 23, 25, 24),
(39, '2024-08-01', '2025-08-01', 1000000.00, 34, 36, 35),
(50, '2024-07-01', '2025-07-01', 850000.00, 45, 47, 46),
(61, '2024-06-01', '2025-06-01', 1200000.00, 56, 58, 57),
(62, '2024-05-01', '2025-05-01', 1150000.00, 67, 69, 68);

--PAGOS_ARRIENDO--
INSERT INTO PAGOS_ARRIENDO (cod_pago, cod_arriendo, fecha_pago, monto_pago, estado_pago)
VALUES
(18, 17, '2024-10-05', 950000.00, 'Pagado'),
(29, 28, '2024-09-05', 900000.00, 'Pendiente'),
(40, 39, '2024-08-05', 1000000.00, 'Pagado'),
(51, 50, '2024-07-05', 850000.00, 'Pagado'),
(62, 61, '2024-06-05', 1200000.00, 'Pendiente'),
(63, 62, '2024-05-05', 1150000.00, 'Pagado');
```

Figura 12: Código de SQL Server "Ingresos de datos para las tablas de la base de datos"

2.2 MySQL

El desarrollo del código en MySQL sigue un proceso similar al de SQL Server, pero se deben hacer algunas adaptaciones y cambios debido a las diferencias de sintaxis y manejo de tipos de datos. A continuación se detalla el proceso de creación y manejo de la base de datos "*Corredor_de_Propiedades*" en MySQL.

2.2.1 Creación de base de datos:

Para comenzar, se debe de conectarse a MySQL y se deben ejecutar los siguientes comandos para crear y usar la base de datos "*Corredor_de_Propiedades*"

```
1  CREATE DATABASE Corredor_de_Propiedades;
2  USE Corredor_de_Propiedades;
```

Figura 13: Código de MySQL "Creación de la Base de Datos"

2.2.2 Creación de Tablas:

A continuación se presenta el código para la creación de tablas, donde se destacan los cambios necesarios en MySQL, como el uso de *CHAR_LENGTH* para medir la longitud de cadenas en lugar de LEN (de SQL Server) y el uso de *ENUM* para campos con valores específicos.

- **Tabla Comuna**

```
1  CREATE TABLE COMUNA(
2          cod_comuna INT NOT NULL,
3          nombre_comuna VARCHAR(25) NOT NULL,
4          PRIMARY KEY(cod_comuna),
5          CHECK (CHAR_LENGTH(nombre_comuna) > 9) -- Verificación de longitud en MySQL
6  );
```

Figura 14: Código de MySQL "Creación de tabla Comuna"

- **Tabla Sector**

```
1  CREATE TABLE SECTOR(
2          cod_sector INT NOT NULL,
3          nombre_sector VARCHAR(30) NOT NULL,
4          cod_comuna INT NOT NULL,
5          PRIMARY KEY(cod_sector),
6          FOREIGN KEY (cod_comuna) REFERENCES COMUNA(cod_comuna),
7          CHECK (CHAR_LENGTH(nombre_sector) > 5)
8  );
```

Figura 15: Código de MySQL "Creación de tabla Sector"

- **Tabla Propiedad**

```
1  CREATE TABLE PROPIEDAD(
2          cod_propiedad INT NOT NULL,
3          calle VARCHAR(50) NOT NULL,
4          numero INT NOT NULL,
5          precio_arriendo DECIMAL(10,2) NOT NULL CHECK (precio_arriendo > 50000),
6          estado ENUM('Disponible', 'Arrendada') NOT NULL,
7          cod_ubicacion INT NOT NULL,
8          PRIMARY KEY(cod_propiedad),
9          FOREIGN KEY (cod_ubicacion) REFERENCES SECTOR(cod_sector)
10 );
```

Figura 16: Código de MySQL "Creación de tabla Propiedad"

- **Tabla Arrendatario**

```
1  CREATE TABLE ARRENDATARIO (
2          cod_arrendatario INT NOT NULL,
3          nombre_arrendatario VARCHAR(30) NOT NULL,
4          rut_arrendatario VARCHAR(12) NOT NULL,
5          renta DECIMAL(10,2) NOT NULL CHECK (renta > 50000),
6          PRIMARY KEY(cod_arrendatario),
7          CHECK (CHAR_LENGTH(rut_arrendatario) = 12)
8  );
```

Figura 17: Código de MySQL "Creación de tabla Arrendatario"

- **Tabla Propietario**

```
1  CREATE TABLE PROPIETARIO(
2          cod_propietario INT NOT NULL,
3          nombre_propietario VARCHAR(30) NOT NULL,
4          rut_propietario VARCHAR(12) UNIQUE NOT NULL,
5          renta DECIMAL(10,2) NOT NULL CHECK (renta > 50000),
6          PRIMARY KEY(cod_propietario),
7          CHECK (CHAR_LENGTH(rut_propietario) = 12)
8  );
```

Figura 18: Código de MySQL "Creación de tabla Propietario"

- **Tabla Visitas de Propiedad**

```
1  CREATE TABLE VISITAS_PROPIEDAD(
2          cod_visitas INT NOT NULL,
3          cod_propiedad INT NOT NULL,
4          nombre_visita VARCHAR(25) NOT NULL,
5          rut_visita VARCHAR(12) NOT NULL,
6          fecha_hora DATETIME NOT NULL,
7          PRIMARY KEY(cod_visitas),
8          FOREIGN KEY (cod_propiedad) REFERENCES PROPIEDAD(cod_propiedad),
9          CHECK (CHAR_LENGTH(rut_visita) = 12)
10 );
```

Figura 19: Código de MySQL "Creación de tabla Visitas de Propiedad"

- **Tabla Reparaciones**

```
1  CREATE TABLE REPARACIONES(
2          cod_reparaciones INT NOT NULL,
3          descripciones VARCHAR(100) NOT NULL,
4          fecha_reparacion DATE NOT NULL,
5          costo DECIMAL(10,2) NOT NULL CHECK (costo >= 50000),
6          cod_propiedad INT NOT NULL,
7          PRIMARY KEY(cod_reparaciones),
8          FOREIGN KEY (cod_propiedad) REFERENCES PROPIEDAD(cod_propiedad)
9  );
```

Figura 20: Código de MySQL "Creación de tabla Reparaciones"

- **Tabla Arriendo**

```
1  CREATE TABLE ARRIENDO(
2          cod_arriendo INT NOT NULL,
3          fecha_inicio_arriendo DATE NOT NULL,
4          fecha_termino_arriendo DATE NOT NULL,
5          costo_final_arriendo DECIMAL(10,2) NOT NULL CHECK (costo_final_arriendo > 50000),
6          cod_propiedad INT NOT NULL,
7          cod_propietario INT NOT NULL,
8          cod_arrendatario INT NOT NULL,
9          PRIMARY KEY(cod_arriendo),
10         FOREIGN KEY (cod_propiedad) REFERENCES PROPIEDAD(cod_propiedad),
11         FOREIGN KEY (cod_propietario) REFERENCES PROPIETARIO(cod_propietario),
12         FOREIGN KEY (cod_arrendatario) REFERENCES ARRENDATARIO(cod_arrendatario)
13 );
```

Figura 21: Código de MySQL "Creación de tabla Arriendo"

- **Tabla Pagos Arriendo**

```
1  CREATE TABLE PAGOS_ARRIENDO (
2          cod_pago INT NOT NULL,
3          cod_arriendo INT NOT NULL,
4          fecha_pago DATE NOT NULL,
5          monto_pago DECIMAL(10,2) NOT NULL CHECK (monto_pago > 50000),
6          estado_pago ENUM('Pagado', 'Pendiente') NOT NULL,
7          PRIMARY KEY(cod_pago),
8          FOREIGN KEY (cod_arriendo) REFERENCES ARRIENDO(cod_arriendo)
9  );
```

Figura 22: Código de MySQL "Creación de tabla Pagos Arriendo"

2.2.3 Inserción de datos:

Al igual que en SQL network, luego de crear las tablas estas deben de ser rellenadas con datos para que las tablas tengan un propósito y cumplan con su función

```
1  -- Inserción de datos --
2  -- Comuna --
3  INSERT INTO COMUNA (cod_comuna, nombre_comuna)
4  VALUES
5  (11, 'Cerro Navia'),
6  (22, 'La Florida'),
7  (33, 'Las Condes'),
8  (44, 'Lo Barnechea'),
9  (55, 'Providencia');
10
11 -- Sector --
12 INSERT INTO SECTOR (cod_sector, nombre_sector, cod_comuna)
13 VALUES
14 (10, 'Sector Central', 11),
15 (20, 'Sector Norte', 22),
16 (30, 'Sector Sur', 33),
17 (40, 'Sector Oriente', 44),
18 (50, 'Sector Poniente', 55),
19 (60, 'La Vega Chica', 11),
20 (70, 'Los Jardines', 33),
21 (80, 'Villa Hermosa', 44),
22 (90, 'El Mirador', 55);
23
24 -- Propiedad --
25 INSERT INTO PROPIEDAD (cod_propiedad, calle, numero, precio_arriendo, estado, cod_ubicacion)
26 VALUES
27 (12, 'Av. Las Torres', 101, 800000.00, 'Disponible', 10),
28 (23, 'Calle Independencia', 202, 950000.00, 'Arrendada', 20),
29 (34, 'Av. Providencia', 303, 1200000.00, 'Disponible', 30),
30 (45, 'Calle Los Pinos', 404, 650000.00, 'Arrendada', 40),
31 (56, 'Calle Irarrazabal', 505, 750000.00, 'Disponible', 50),
32 (67, 'Calle Bellavista', 606, 820000.00, 'Disponible', 60);
33
34 -- Arrendatario --
35 INSERT INTO ARRENDATARIO (cod_arrendatario, nombre_arrendatario, rut_arrendatario, renta)
36 VALUES
37 (13, 'Juan Perez', '12.345.678-9', 1500000.00),
38 (24, 'Carla Reyes', '98.765.432-1', 1300000.00),
39 (35, 'Pedro Gomez', '11.111.111-1', 1400000.00),
40 (46, 'Ana Diaz', '22.222.222-2', 1700000.00),
41 (57, 'Luis Lopez', '33.333.333-3', 1600000.00),
42 (68, 'Miguel Sanchez', '44.444.444-4', 1550000.00);
43
44 -- Propietario --
45 INSERT INTO PROPIETARIO (cod_propietario, nombre_propietario, rut_propietario, renta)
46 VALUES
47 (14, 'Carlos Gonzalez', '44.444.444-4', 2000000.00),
48 (25, 'Maria Silva', '55.555.555-5', 2100000.00),
49 (36, 'Jorge Salinas', '66.666.666-6', 1900000.00),
```

```
(47, 'Sofia Mendez', '77.777.777-7', 2200000.00),
(58, 'Patricia Vega', '88.888.888-8', 1800000.00),
(69, 'Alberto Vargas', '99.999.999-9', 2300000.00);

-- Visitas de propiedad --
INSERT INTO VISITAS_PROPIEDAD (cod_visitas, cod_propiedad, nombre_visita, rut_visita, fecha_hora)
VALUES
(15, 12, 'Miguel Fernandez', '99.999.999-9', '2024-10-10 14:30:00'),
(26, 23, 'Jose Gonzalez', '12.312.312-3', '2024-10-11 09:45:00'),
(37, 34, 'Paula Nuñez', '98.798.798-7', '2024-10-12 11:15:00'),
(48, 45, 'Ricardo Fuentes', '32.132.132-1', '2024-10-13 15:00:00'),
(59, 56, 'Lucia Morales', '21.321.321-3', '2024-10-14 13:00:00'),
(60, 67, 'Felipe Cruz', '33.344.455-6', '2024-10-15 16:00:00');

-- Reparaciones --
INSERT INTO REPARACIONES (cod_reparaciones, descripciones, fecha_reparacion, costo, cod_propiedad)
VALUES
(16, 'Falla en las tuberias del baño', '2024-09-10', 100000.00, 12),
(27, 'Reparación del Portón Frontal', '2024-09-15', 150000.00, 23),
(38, 'Obstrucción en las Tuberias del Gas', '2024-09-20', 120000.00, 34),
(49, 'Arreglo eléctrico del medidor', '2024-09-25', 200000.00, 45),
(60, 'Reparación de ventanas', '2024-09-30', 80000.00, 56),
(61, 'Cambio de pisos', '2024-10-05', 50000.00, 67);

-- Arriendo --
INSERT INTO ARRIENDO (cod_arriendo, fecha_inicio_arriendo, fecha_termino_arriendo, costo_final_arriendo,
cod_propiedad, cod_propietario, cod_arrendatario)
VALUES
(17, '2024-10-01', '2025-10-01', 950000.00, 12, 14, 13),
(28, '2024-09-01', '2025-09-01', 900000.00, 23, 25, 24),
(39, '2024-08-01', '2025-08-01', 1000000.00, 34, 36, 35),
(50, '2024-07-01', '2025-07-01', 850000.00, 45, 47, 46),
(61, '2024-06-01', '2025-06-01', 1200000.00, 56, 58, 57),
(62, '2024-05-01', '2025-05-01', 1150000.00, 67, 69, 68);

-- Pagos de arriendo --
INSERT INTO PAGOS_ARRIENDO (cod_pago, cod_arriendo, fecha_pago, monto_pago, estado_pago)
VALUES
(18, 17, '2024-10-05', 950000.00, 'Pagado'),
(29, 28, '2024-09-05', 900000.00, 'Pendiente'),
(40, 39, '2024-08-05', 1000000.00, 'Pagado'),
(51, 50, '2024-07-05', 850000.00, 'Pagado'),
(62, 61, '2024-06-05', 1200000.00, 'Pendiente'),
(63, 62, '2024-05-05', 1150000.00, 'Pagado');
```

Figura 23: Código de MySQL "Inserción de datos en las tablas ya creadas"

3. Modelo Esquema Relacional

A Través de los softwares usados, donde se complementaban las relaciones entre las llaves primarias y foráneas, se crea el esquema modelo relacional, el cual, se transforman las entidades y sus relaciones en tablas para un sistema de gestión de bases de datos. Identificando las claves primarias y foráneas, y se organiza la información de las tablas el cual los mismos software tienen la funcionalidad automática. A continuación, se presentan los siguientes esquemas:

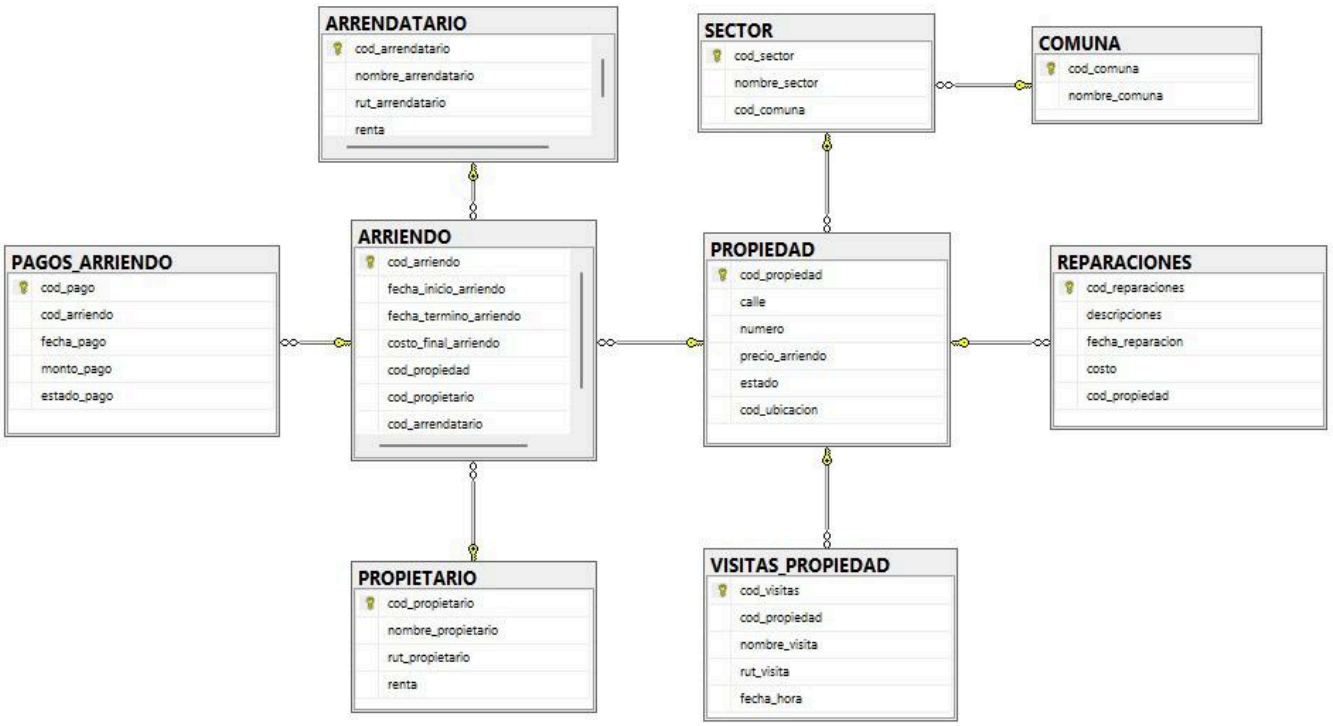


Figura 24: DataBase diagrams, SQL Server "Modelo de Esquema Relacional de tablas"

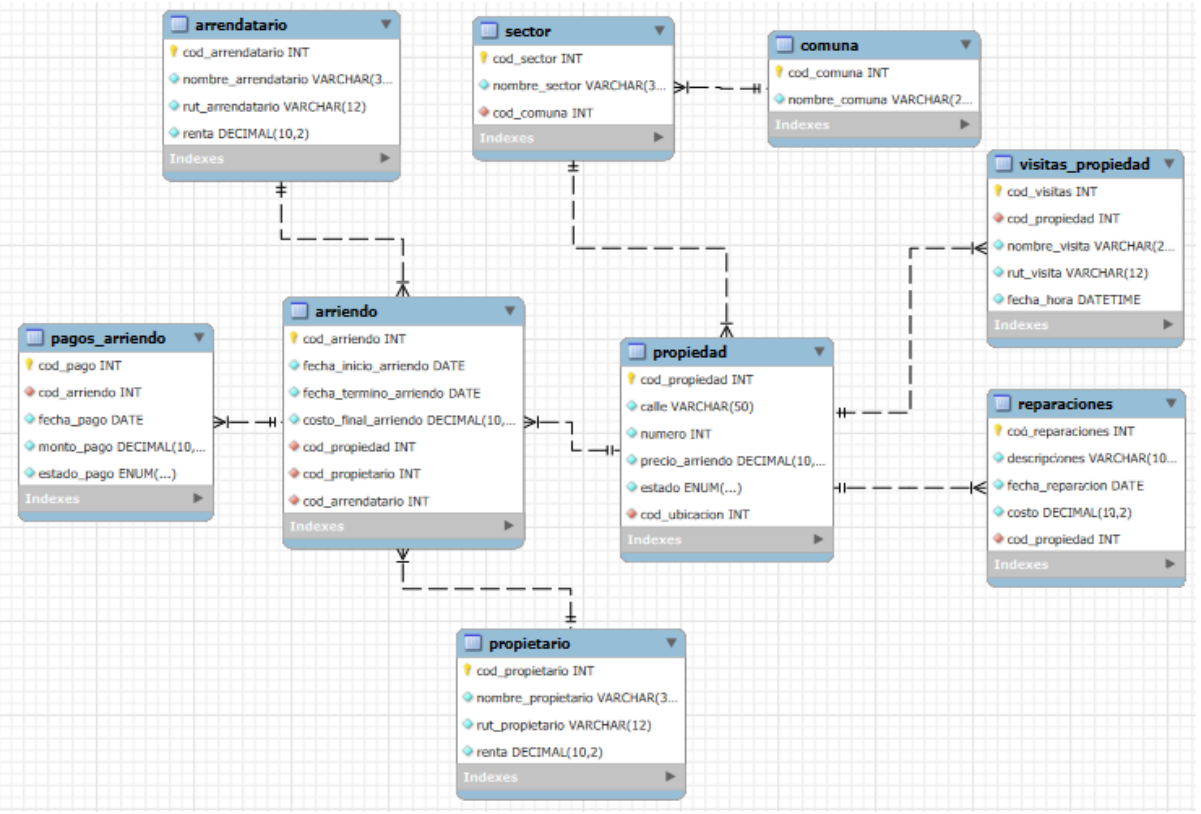


Figura 25: EER Model, MySQL "Modelo de Esquema Relacional de tablas"

4. Consultas SQL Simples

A continuación se presentan 20 consultas simples las cuales permiten seleccionar, filtrar, ordenar y realizar cálculos sencillos sobre los datos almacenados en la base de datos en las tablas del Corredor de propiedades, presentando las siguientes consultas de cada software de base de datos:

4.1 Consultas de SQL Server Microsoft

```
1  --|1|Consulta para listar todas las tablas--
2  SELECT * FROM ARRENDATARIO;
3  SELECT * FROM ARRIENDO;
4  SELECT * FROM COMUNA;
5  SELECT * FROM PAGOS_ARRIENDO;
6  SELECT * FROM PROPIEDAD
7  SELECT * FROM PROPIETARIO;
8  SELECT * FROM REPARACIONES;
9  SELECT * FROM SECTOR;
10 SELECT * FROM VISITAS_PROPIEDAD;
```

Figura 26: Código de SQL Server "Consultas simple"

Las consultas *SELECT * FROM Tablas* listan todos los datos almacenados en cada una de las tablas mencionadas (*ARRENDATARIO*, *ARRIENDO*, *COMUNA*, *PAGOS_ARRIENDO*, *PROPIEDAD*, *PROPIETARIO*, *REPARACIONES*, *SECTOR*, *VISITAS_PROPIEDAD*). Estas consultas permiten obtener una visión completa de los datos en cada tabla.

El objetivo de esta consulta es visualizar todas las columnas y registros presentes en cada tabla de la base de datos. Es útil para revisar si se han ingresado correctamente los datos en cada tabla.

	cod_arrendatario	nombre_arrendatario	rut_arrendatario	renta
1	13	Juan Perez	12.345.678-9	1500000.00
2	24	Carla Reyes	98.765.432-1	1300000.00
3	35	Pedro Gomez	11.111.111-1	1400000.00
4	46	Ana Diaz	22.222.222-2	1700000.00
5	57	Luis Lopez	33.333.333-3	1600000.00
6	68	Miguel Sanchez	44.444.444-4	1550000.00

Tabla 1: Código de SQL Server "Consulta simple 1, Tabla ARRENDATARIO"

	cod_arriendo	fecha_inicio_arriendo	fecha_termino_arriendo	costo_final_arriendo	cod_propiedad	cod_propietario	cod_arrendatario
1	17	2024-10-01	2025-10-01	950000.00	12	14	13
2	28	2024-09-01	2025-09-01	900000.00	23	25	24
3	39	2024-08-01	2025-08-01	1000000.00	34	36	35
4	50	2024-07-01	2025-07-01	850000.00	45	47	46
5	61	2024-06-01	2025-06-01	1200000.00	56	58	57
6	62	2024-05-01	2025-05-01	1150000.00	67	69	68

Tabla 2: Código de SQL Server "Consulta simple 1, Tabla ARRIENDO"

	cod_comuna	nombre_comuna
1	11	Cerro Navia
2	22	La Florida
3	33	Las Condes
4	44	Lo Barnechea
5	55	Providencia

Tabla 3: Código de SQL Server "Consulta simple 1, Tabla COMUNA"

	cod_pago	cod_arriendo	fecha_pago	monto_pago	estado_pago
1	18	17	2024-10-05	950000.00	Pagado
2	29	28	2024-09-05	900000.00	Pendiente
3	40	39	2024-08-05	1000000.00	Pagado
4	51	50	2024-07-05	850000.00	Pagado
5	62	61	2024-06-05	1200000.00	Pendiente
6	63	62	2024-05-05	1150000.00	Pagado

Tabla 4: Código de SQL Server "Consulta simple 1, Tabla PAGOS_ARRIENDO"

	cod_propiedad	calle	numero	precio_arriendo	estado	cod_ubicacion
1	12	Av. Las Torres	101	800000.00	Disponible	10
2	23	Calle Independencia	202	950000.00	Arrendada	20
3	34	Av. Providencia	303	1200000.00	Disponible	30
4	45	Calle Los Pinos	404	650000.00	Arrendada	40
5	56	Calle Irarrazabal	505	750000.00	Disponible	50
6	67	Calle Bellavista	606	820000.00	Disponible	60

Tabla 5: Código de SQL Server "Consulta simple 1, Tabla PROPIEDAD"

	cod_propietario	nombre_propietario	rut_propietario	renta
1	14	Carlos Gonzalez	44.444.444-4	2000000.00
2	25	Maria Silva	55.555.555-5	2100000.00
3	36	Jorge Salinas	66.666.666-6	1900000.00
4	47	Sofia Mendez	77.777.777-7	2200000.00
5	58	Patricia Vega	88.888.888-8	1800000.00
6	69	Alberto Vargas	99.999.999-9	2300000.00

Tabla 6: Código de SQL Server "Consulta simple 1, Tabla PROPIETARIO"

	cod_reparaciones	descripciones	fecha_reparacion	costo	cod_propiedad
1	16	Falla en las tuberias del baño	2024-09-10	100000.00	12
2	27	Reparación del Portón Frontal	2024-09-15	150000.00	23
3	38	Obstrucción en las Tuberias...	2024-09-20	120000.00	34
4	49	Arreglo eléctrico del medidor	2024-09-25	200000.00	45
5	60	Reparación de ventanas	2024-09-30	80000.00	56
6	61	Cambio de pisos	2024-10-05	50000.00	67

Tabla 7: Código de SQL Server "Consulta simple 1, Tabla REPARACIONES"

	cod_sector	nombre_sector	cod_comuna
1	10	Sector Central	11
2	20	Sector Norte	22
3	30	Sector Sur	33
4	40	Sector Oriente	44
5	50	Sector Ponie...	55
6	60	La Vega Chica	11
7	70	Los Jardines	33
8	80	Villa Hermosa	44

Tabla 8: Código de SQL Server "Consulta simple 1, Tabla SECTOR"

	cod_visitas	cod_propiedad	nombre_visita	rut_visita	fecha_hora
1	15	12	Miguel Fernandez	99.999.999-9	2024-10-10 14:30:00.000
2	26	23	Jose Gonzalez	12.312.312-3	2024-10-11 09:45:00.000
3	37	34	Paula Nuñez	98.798.798-7	2024-10-12 11:15:00.000
4	48	45	Ricardo Fuentes	32.132.132-1	2024-10-13 15:00:00.000
5	59	56	Lucia Morales	21.321.321-3	2024-10-14 13:00:00.000
6	60	67	Felipe Cruz	33.344.455-6	2024-10-15 16:00:00.000

Tabla 9: Código de SQL Server "Consulta 1, Tabla VISITAS_PROPIEDAD"

```
1  --|2|Obtener todas las comunas que tienen más de 10 caracteres en su nombre--
2  SELECT * FROM COMUNA
3  WHERE LEN(nombre_comuna) > 10;
```

Figura 27: Código de SQL Server "Consulta simple 2"

La parte de "*SELECT * FROM COMUNA*" y "*WHERE LEN(nombre_comuna) > 10;*" se encarga de obtener todas las comunas que contienen un nombre de más de 10 caracteres. Se utiliza el asterisco para seleccionar todas las columnas de la tabla.

	cod_comuna	nombre_comuna
1	11	Cerro Navia
2	44	Lo Barnechea
3	55	Providencia

Tabla 10: Código de SQL Server "Consulta simple 2"

```
1  --|3|Obtener todas las propiedades que están disponibles
2  SELECT * FROM PROPIEDAD
3  WHERE estado = 'Disponible';
```

Figura 28: Código de SQL Server "Consulta simple 3"

Esta consulta selecciona todas las columnas de la tabla PROPIEDAD donde el estado de la propiedad es "Disponible". El propósito es listar todas las propiedades que están actualmente disponibles.

	cod_propiedad	calle	numero	precio_arriendo	estado	cod_ubicacion
1	12	Av. Las Torres	101	800000.00	Disponible	10
2	34	Av. Providencia	303	1200000.00	Disponible	30
3	56	Calle Irarrazabal	505	750000.00	Disponible	50
4	67	Calle Bellavista	606	820000.00	Disponible	60

Tabla 11: Código de SQL Server "Consulta simple 3"

```
1  --|4|Listar los arrendatarios con renta mayor a $1 000 000--
2  SELECT nombre_arrendatario, renta
3  FROM ARRENDATARIO WHERE renta > 1000000.00;
```

Figura 29: Código de SQL Server "Consulta simple 4"

Esta consulta selecciona los nombres y las rentas de todos los arrendatarios que tienen una renta superior a 1.000.000. La consulta se ejecuta en la tabla "*ARRENDATARIO*", y filtra los registros cuyo valor en la columna "*RENTA*" excede dicho umbral.

	nombre_arrendatario	renta
1	Juan Perez	1500000.00
2	Carla Reyes	1300000.00
3	Pedro Gomez	1400000.00
4	Ana Diaz	1700000.00
5	Luis Lopez	1600000.00
6	Miguel Sanchez	1550000.00

Tabla 12: Código de SQL Server "Consulta simple 4"

```
1  --|5|Listar los propietarios con renta mayor a $2 000 000--
2  SELECT nombre_propietario, renta
3  FROM PROPIETARIO WHERE renta > 2000000.00;
```

Figura 30: Código de SQL Server "Consulta simple 5"

Esta consulta da los nombres de los propietarios y sus rentas, pero sólo para aquellos que cuenten con ingreso mayor a 2.000.000. La búsqueda se realiza en la tabla "*PROPIETARIO*", filtrando los registros con una renta superior a ese valor.

	nombre_propietario	renta
1	Maria Silva	2100000.00
2	Sofia Mendez	2200000.00
3	Alberto Vargas	2300000.00

Tabla 13: Código de SQL Server "Consulta simple 5"

```
1  --|6|Obtener todas las propiedades cuyo precio de arriendo sea mayor a 900000
2  SELECT * FROM PROPIEDAD
3  WHERE precio_arriendo > 900000;
```

Figura 31: Código de SQL Server "Consulta simple 6"

Esta consulta selecciona todas las columnas de la tabla "*PROPIEDAD*", pero únicamente para las propiedades cuyo precio de inicio es superior a 900 000. Este filtro se aplica sobre el campo "*precio_arriendo*".

	cod_propiedad	calle	numero	precio_arriendo	estado	cod_ubicacion
1	23	Calle Independencia	202	950000.00	Arrendada	20
2	34	Av. Providencia	303	1200000.00	Disponible	30

Tabla 14: Código de SQL Server "Consulta simple 6"

```
1  --|7|Mostrar el costo total de reparaciones en una propiedad específica--
2  SELECT SUM(costo) AS costo_total_reparaciones
3  FROM REPARACIONES
4  WHERE cod_propiedad = (SELECT cod_propiedad FROM PROPIEDAD WHERE calle = 'Av. Providencia');
```

Figura 32: Código de SQL Server "Consulta simple 7"

En este caso, la consulta suma el costo total de las reparaciones para las propiedades ubicadas en la calle "Av. Providencia". Primero, a través de una subconsulta, se obtiene el "*cod_propiedad*" de aquellas propiedades ubicadas en la calle mencionada. Luego, la consulta principal calcula la suma de los costos ligados a esas reparaciones.

	costo_total_reparaciones
1	120000.00

Tabla 15: Código de SQL Server "Consulta simple 7"

```
1  --|8|Obtener los sectores que contienen la palabra 'Vega'
2  SELECT * FROM SECTOR
3  WHERE nombre_sector LIKE '%Vega%';
```

Figura 33: Código de SQL Server "Consulta simple 8"

Ahora la consulta selecciona todas las columnas de la tabla "*SECTOR*" donde el nombre del sector contiene la palabra "Vega". El operador "*LIKE*" con el comodín "*%*" se utiliza para buscar coincidencias momentáneas en el campo de "*nombre_sector*".

	cod_sector	nombre_sector	cod_comuna
1	60	La Vega Chica	11

Tabla 16: Código de SQL Server "Consulta simple 8"

```
1  --|9|Listar todos los pagos de arriendo pendientes--
2  SELECT PA.cod_pago, PA.fecha_pago, PA.monto_pago FROM PAGOS_ARRIENDO PA
3  WHERE PA.estado_pago = 'Pendiente';
```

Figura 34: Código de SQL Server "Consulta simple 9"

La consulta selecciona el código, la fecha y el monto de los pagos de la tabla "*PAGOS_ARRIENDO*" donde el estado del pago es "Pendiente". Sólo se mostrarán los registros de pagos que aún no se han completado o salvado.

	cod_pago	fecha_pago	monto_pago
1	29	2024-09-05	900000.00
2	62	2024-06-05	1200000.00

Tabla 17: Código de SQL Server "Consulta simple 9"

```
1  --|10|Contar cuántas propiedades tienen un precio de arriendo mayor a 600000
2  SELECT COUNT(*) AS TotalPropiedades
3  FROM PROPIEDAD
4  WHERE precio_arriendo > 600000;
```

Figura 35: Código de SQL Server "Consulta simple 10"

En la consulta se cuenta el número total de propiedades en la tabla "*PROPIEDAD*" cuyo precio a llegar es mayor a 600 000. El resultado será un único valor que indica cuántas propiedades cumplen con dicho criterio.

	TotalPropiedades
1	6

Tabla 18: Código de SQL Server "Consulta simple 10"

```
1  --|11|Listar los propietarios que tienen una renta mayor a 2000000 o cuyo RUT comienza con '88'
2  SELECT * FROM PROPIETARIO
3  WHERE renta > 2000000 OR rut_propietario LIKE '88%';
```

Figura 36: Código de SQL Server "Consulta simple 11"

En este caso la consulta selecciona todos los registros de la tabla "*PROPIETARIO*" donde la renta es mayor a 2.000.000 o el "*rut_propietario*" comienza con "88". El resultado incluirá todos los propietarios que cumplan al menos una de estas condiciones.

	cod_propietario	nombre_propietario	rut_propietario	renta
1	25	Maria Silva	55.555.555-5	2100000.00
2	47	Sofia Mendez	77.777.777-7	2200000.00
3	58	Patricia Vega	88.888.888-8	1800000.00
4	69	Alberto Vargas	99.999.999-9	2300000.00

Tabla 19: Código de SQL Server "Consulta simple 11"

```
1  --|12|Obtener las reparaciones cuyo costo es mayor o igual a 150000
2  SELECT * FROM REPARACIONES
3  WHERE costo >= 150000;
```

Figura 37: Código de SQL Server "Consulta simple 12"

Dentro de esta consulta se seleccionan todos los registros de la tabla "*REPARACIONES*" donde el costo es mayor o igual a 150 000. El resultado incluirá todos los detalles de las reparaciones que tienen un costo significativo, lo que permite identificar reparaciones más costosas.

	cod_reparaciones	descripciones	fecha_reparacion	costo	cod_propiedad
1	27	Reparación del Portón Frontal	2024-09-15	150000.00	23
2	49	Arreglo eléctrico del medidor	2024-09-25	200000.00	45

Tabla 20: Código de SQL Server "Consulta simple 12"

```
1  --|13|Listar las propiedades arrendadas que tienen un costo final de arriendo menor a 1000000
2  SELECT cod_arriendo,costo_final_arriendo,cod_propietario,cod_propietario FROM ARRIENDO
3  WHERE costo_final_arriendo < 1000000;
```

Figura 38: Código de SQL Server "Consulta simple 13"

Esta consulta selecciona los registros de la tabla "*ARRIENDO*" donde el "*costo_final_arriendo*" es menor a 1.000.000. Los campos que se recuperan incluyen el código de arriendo "(*cod_arriendo*)", el costo final del arriendo "(*costo_final_arriendo*)", y el código del propietario "(*cod_propietario*)". Esto permite identificar los arrendamientos con un costo menor al millón.

	cod_arriendo	costo_final_arriendo	cod_propietario	cod_propietario
1	17	950000.00	14	14
2	28	900000.00	25	25
3	50	850000.00	47	47

Tabla 21: Código de SQL Server "Consulta simple 12"

```
1  --|14|Obtener todos los arrendatarios cuyo nombre contiene la letra 'i'
2  SELECT * FROM ARRENDATARIO
3  WHERE nombre_arrendatario LIKE '%i%';
```

Figura 39: Código de SQL Server "Consulta simple 14"

En esta consulta se selecciona todos los registros de la tabla "*ARRENDATARIO*" donde el "*nombre_arrendatario*" contiene la letra "i". El uso del operador "*LIKE*" junto con los caracteres "*%*" permite encontrar nombres que tengan "i" en cualquier parte de la cadena. Esto es útil para filtrar arrendatarios según un criterio específico en su nombre.

	cod_arrendatario	nombre_arrendatario	rut_arrendatario	renta
1	46	Ana Diaz	22.222.222-2	1700000.00
2	57	Luis Lopez	33.333.333-3	1600000.00
3	68	Miguel Sanchez	44.444.444-4	1550000.00

Tabla 22: Código de SQL Server "Consulta simple 14"

```
1  --|15|Obtener la información de las propiedades que están arrendadas y cuyo precio de arriendo es
2  mayor a 700000
3  SELECT * FROM PROPIEDAD
   WHERE estado = 'Arrendada' AND precio_arriendo > 700000;
```

Figura 40: Código de SQL Server "Consulta simple 15"

Dentro de esta consulta se selecciona todos los registros de la tabla "*PROPIEDAD*" que cumplen dos condiciones: el estado de la propiedad debe ser "*Arrendada*" y el "*precio_arriendo*" debe ser superior a 700 000. Esto permite identificar propiedades que están actualmente arrendadas y que tienen un precio de arriendo alto, lo cual puede ser útil para la gestión de arrendamientos.

	cod_propiedad	calle	numero	precio_arriendo	estado	cod_ubicacion
1	23	Calle Independencia	202	950000.00	Arrendada	20

Tabla 23: Código de SQL Server "Consulta simple 15"

```
1  --|16|Obtener la suma de todas las rentas de propietarios--
2  SELECT SUM(renta) AS total_renta_propietarios FROM PROPIETARIO;
```

Figura 41: Código de SQL Server "Consulta simple 16"

Esta consulta calcula la suma total de las rentas de todos los propietarios en la tabla "*PROPIETARIO*". El resultado se presenta como "*total_renta_propietarios*". Esto sirve para evaluar el ingreso total generado por los propietarios en un periodo determinado de tiempo, además para análisis financieros o de gestión de propiedades.

	total_renta_propietarios
1	12300000.00

Tabla 24: Código de SQL Server "Consulta simple 16"

```
1  --|17|Obtener la suma de todas las rentas de arrendatarios--
2  SELECT SUM(renta) AS total_renta_arrendatarios FROM ARRENDATARIO;
```

Figura 42: Código de SQL Server "Consulta simple 17"

Acá en la consulta se calcula la suma total de las rentas pagadas por todos los arrendatarios en la tabla "*ARRENDATARIO*". El resultado se presenta con el nombre "*total_renta_arrendatarios*". Así se logra entender el total de ingresos generados por los arrendatarios, permitiendo realizar análisis financieros sobre el flujo de caja y la rentabilidad del arrendamiento.

	total_renta_arrendatarios
1	9050000.00

Tabla 25: Código de SQL Server "Consulta simple 17"

```
1  --|18|Obtener todas las propiedades en un sector específico con código 10
2  SELECT * FROM PROPIEDAD
3  WHERE cod_ubicacion = 10;
```

Figura 43: Código de SQL Server "Consulta simple 18"

Esta consulta selecciona todas las columnas de la tabla "*PROPIEDAD*" para las propiedades que tienen un código de ubicación más específico, en este caso, 10. Esto para obtener información sobre todas las propiedades ubicadas en esa ubicación particular.

	cod_propiedad	calle	numero	precio_arriendo	estado	cod_ubicacion
1	12	Av. Las Torres	101	800000.00	Disponible	10

Tabla 26: Código de SQL Server "Consulta simple 18"

```
1  --|19|Obtener los pagos realizados después de una fecha 2024-08-01--
2  SELECT PA.cod_pago, PA.fecha_pago, PA.monto_pago
3  FROM PAGOS_ARRIENDO PA
4  WHERE PA.fecha_pago > '2024-08-01';
```

Figura 44: Código de SQL Server "Consulta simple 19"

En este caso, la consulta selecciona el código de pago, la fecha de pago y el monto del pago de la tabla "*PAGOS_ARRIENDO*" para todos los registros en los que la fecha de pago es después del 1 de agosto de 2024. Esto permite obtener información sobre los pagos de arriendo realizados después de esa fecha.

	cod_pago	fecha_pago	monto_pago
1	18	2024-10-05	950000.00
2	29	2024-09-05	900000.00
3	40	2024-08-05	1000000.00

Tabla 27: Código de SQL Server "Consulta simple 19"

```
1  --|20|Obtener el nombre de arrendatarios cuya renta está entre 1500000 y 1800000
2  SELECT * FROM ARRENDATARIO
3  WHERE renta BETWEEN 1500000 AND 1800000;
```

Figura 45: Código de SQL Server "Consulta simple 20"

Por último, en esta consulta se seleccionan todos los registros de la tabla "*ARRENDATARIO*" donde la renta se encuentra en el rango de 1.500.000 a 1.800.000. Esto para identificar a los arrendatarios que tienen una renta dentro de ese intervalo específico, lo que puede ayudar en la gestión de arrendamientos o análisis de precios.

	cod_arrendatario	nombre_arrendatario	rut_arrendatario	renta
1	13	Juan Perez	12.345.678-9	1500000.00
2	46	Ana Diaz	22.222.222-2	1700000.00
3	57	Luis Lopez	33.333.333-3	1600000.00
4	68	Miguel Sanchez	44.444.444-4	1550000.00

Tabla 28: Código de SQL Server "Consulta simple 19"

4.2 Consultas de MySQL

Por el lado de MySQL, las consultas son las mismas y funcionan de la misma manera con la excepción de los ajustes del código para que se adapte a la sintaxis de MySQL:

```
1  -- Listar todas las tablas --
2  SELECT * FROM ARRENDATARIO;
3  SELECT * FROM ARRIENDO;
4  SELECT * FROM COMUNA;
5  SELECT * FROM PAGOS_ARRIENDO;
6  SELECT * FROM PROPIEDAD;
7  SELECT * FROM PROPIETARIO;
8  SELECT * FROM REPARACIONES;
9  SELECT * FROM SECTOR;
10 SELECT * FROM VISITAS_PROPIEDAD;
```

Figura 46: Código de MySQL "Consultas simples"

En este caso, el código queda igual al que se tenía en SQL Server.

cod_arrendatario	nombre_arrendatario	rut_arrendatario	renta
13	Juan Perez	12.345.678-9	1500000.00
24	Carla Reyes	98.765.432-1	1300000.00
35	Pedro Gomez	11.111.111-1	1400000.00
46	Ana Diaz	22.222.222-2	1700000.00
57	Luis Lopez	33.333.333-3	1600000.00
68	Miguel Sanchez	44.444.444-4	1550000.00
NULL	NULL	NULL	NULL

Tabla 29: Código de MySQL "Consulta simple 1, Tabla ARRENDATARIO"

cod_arriendo	fecha_inicio_arriendo	fecha_termino_arriendo	costo_final_arriendo	cod_propiedad	cod_propietario	cod_arrendatario
17	2024-10-01	2025-10-01	950000.00	12	14	13
28	2024-09-01	2025-09-01	900000.00	23	25	24
39	2024-08-01	2025-08-01	1000000.00	34	36	35
50	2024-07-01	2025-07-01	850000.00	45	47	46
61	2024-06-01	2025-06-01	1200000.00	56	58	57
62	2024-05-01	2025-05-01	1150000.00	67	69	68
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Tabla 30: Código de MySQL "Consulta simple 1, Tabla ARRIENDO"

cod_comuna	nombre_comuna
11	Cerro Navia
22	La Florida
33	Las Condes
44	Lo Barnechea
55	Providencia
NULL	NULL

Tabla 31: Código de MySQL "Consulta simple 1, Tabla COMUNA"

cod_pago	cod_arriendo	fecha_pago	monto_pago	estado_pago
18	17	2024-10-05	950000.00	Pagado
29	28	2024-09-05	900000.00	Pendiente
40	39	2024-08-05	1000000.00	Pagado
51	50	2024-07-05	850000.00	Pagado
62	61	2024-06-05	1200000.00	Pendiente
63	62	2024-05-05	1150000.00	Pagado
NULL	NULL	NULL	NULL	NULL

Tabla 32: Código de MySQL "Consulta simple 1, Tabla PAGOS ARRIENDO"

cod_propiedad	calle	numero	precio_arriendo	estado	cod_ubicacion
12	Av. Las Torres	101	800000.00	Disponible	10
23	Calle Independencia	202	950000.00	Arrendada	20
34	Av. Providencia	303	1200000.00	Disponible	30
45	Calle Los Pinos	404	650000.00	Arrendada	40
56	Calle Irarrazabal	505	750000.00	Disponible	50
67	Calle Bellavista	606	820000.00	Disponible	60
NULL	NULL	NULL	NULL	NULL	NULL

Tabla 33: Código de MySQL "Consulta simple 1, Tabla PROPIEDAD"

cod_propietario	nombre_propietario	rut_propietario	renta
14	Carlos Gonzalez	44.444.444-4	2000000.00
25	Maria Silva	55.555.555-5	2100000.00
36	Jorge Salinas	66.666.666-6	1900000.00
47	Sofia Mendez	77.777.777-7	2200000.00
58	Patricia Vega	88.888.888-8	1800000.00
69	Alberto Vargas	99.999.999-9	2300000.00
NULL	NULL	NULL	NULL

Tabla 34: Código de MySQL "Consulta simple 1, Tabla PROPIETARIO"

cod_reparaciones	descripciones	fecha_reparacion	costo	cod_propiedad
16	Falla en las tuberias del baño	2024-09-10	100000.00	12
27	Reparación del Portón Frontal	2024-09-15	150000.00	23
38	Obstrucción en las Tuberias del Gas	2024-09-20	120000.00	34
49	Arreglo eléctrico del medidor	2024-09-25	200000.00	45
60	Reparación de ventanas	2024-09-30	80000.00	56
61	Cambio de pisos	2024-10-05	50000.00	67
NULL	NULL	NULL	NULL	NULL

Tabla 35: Código de MySQL "Consulta simple 1, Tabla REPARACIONES"

cod_sector	nombre_sector	cod_comuna
10	Sector Central	11
20	Sector Norte	22
30	Sector Sur	33
40	Sector Oriente	44
50	Sector Poniente	55
60	La Vega Chica	11
70	Los Jardines	33
80	Villa Hermosa	44
90	El Mirador	55
NULL	NULL	NULL

Tabla 36: Código de MySQL "Consulta simple 1, Tabla COMUNA"

cod_visitas	cod_propiedad	nombre_visita	rut_visita	fecha_hora
15	12	Miguel Fernandez	99.999.999-9	2024-10-10 14:30:00
26	23	Jose Gonzalez	12.312.312-3	2024-10-11 09:45:00
37	34	Paula Nuñez	98.798.798-7	2024-10-12 11:15:00
48	45	Ricardo Fuentes	32.132.132-1	2024-10-13 15:00:00
59	56	Lucia Morales	21.321.321-3	2024-10-14 13:00:00
60	67	Felipe Cruz	33.344.455-6	2024-10-15 16:00:00
NULL	NULL	NULL	NULL	NULL

Tabla 37: Código de MySQL "Consulta simple 1, Tabla VISITAS PROPIEDAD"

```
1  -- Comunas con nombre mayor a 10 caracteres --
2  SELECT * FROM COMUNA
3  WHERE CHAR_LENGTH(nombre_comuna) > 10;
```

Figura 47: Código de MySQL "Consultas simples, Comunas con nombre mayor a 10 carácter"

Diferencia: En SQL Server se usa *LEN*, mientras que en MySQL se emplea *CHAR_LENGTH* para medir la longitud de las cadenas.

cod_comuna	nombre_comuna
11	Cerro Navia
44	Lo Barnechea
55	Providencia
NULL	NULL

Tabla 38: Código de MySQL "Consulta simple 2"

```
1  -- Propiedades disponibles --
2  SELECT * FROM PROPIEDAD
3  WHERE estado = 'Disponible';
```

Figura 48: Código de MySQL "Consultas simples, Comunas con nombre mayor a 10 carácter"

Diferencias: No se realizaron cambios en esta consulta, ya que la sintaxis es compatible en ambos motores.

cod_propiedad	calle	numero	precio_arriendo	estado	cod_ubicacion
12	Av. Las Torres	101	800000.00	Disponible	10
34	Av. Providencia	303	1200000.00	Disponible	30
56	Calle Irarrazabal	505	750000.00	Disponible	50
67	Calle Bellavista	606	820000.00	Disponible	60
NULL	NULL	NULL	NULL	NULL	NULL

Tabla 39: Código de MySQL "Consulta simple 3"

```
1  -- Arrendatarios con renta mayor a $1,000,000 --
2  SELECT nombre_arrendatario, renta
3  FROM ARRENDATARIO
4  WHERE renta > 1000000.00;
```

Figura 49: Código de MySQL "Consultas simples, Comunas con nombre mayor a 10 caracter"

Diferencias: No se realizaron cambios en esta consulta, ya que la sintaxis es compatible en ambos motores.

nombre_arrendatario	renta
Juan Perez	1500000.00
Carla Reyes	1300000.00
Pedro Gomez	1400000.00
Ana Diaz	1700000.00
Luis Lopez	1600000.00
Miguel Sanchez	1550000.00

Tabla 40: Código de MySQL "Consulta simple 4"

```
1  -- Propietarios con renta mayor a $2,000,000 --
2  SELECT nombre_propietario, renta
3  FROM PROPIETARIO
4  WHERE renta > 2000000.00;
```

Figura 50: Código de MySQL "Consultas simples, Comunas con nombre mayor a 10 caracter"

Diferencias: No se realizaron cambios en esta consulta, ya que la sintaxis es compatible en ambos motores.

nombre_propietario	renta
Maria Silva	2100000.00
Sofia Mendez	2200000.00
Alberto Vargas	2300000.00

Tabla 41: Código de MySQL "Consulta simple5"

```
1  --- Propiedades cuyo precio de arriendo es mayor a 900,000 --
2  SELECT * FROM PROPIEDAD
3  WHERE precio_arriendo > 900000;
```

Figura 51: Código de MySQL "Consultas simples, Comunas con nombre mayor a 10 caracter"

Diferencias: No se realizaron cambios en esta consulta, ya que la sintaxis es compatible en ambos motores.

cod_propiedad	calle	numero	precio_arriendo	estado	cod_ubicacion
23	Calle Independencia	202	950000.00	Arrendada	20
34	Av. Providencia	303	1200000.00	Disponible	30
NULL	NULL	NULL	NULL	NULL	NULL

Tabla 42: Código de MySQL "Consulta simple 6"

```
1  -- Costo total de reparaciones en una propiedad específica --
2  SELECT SUM(costo) AS costo_total_reparaciones
3  FROM REPARACIONES
4  WHERE cod_propiedad = (SELECT cod_propiedad FROM PROPIEDAD WHERE calle = 'Av. Providencia');
```

Figura 52: Código de MySQL "Consultas simples, Comunas con nombre mayor a 10 caracter"

Diferencias: No se realizaron cambios en esta consulta, ya que la sintaxis es compatible en ambos motores.

costo_total_reparaciones
120000.00

Tabla 43: Código de MySQL "Consulta simple 7"

```
1  -- Sectores que contienen 'Vega' en su nombre --
2  SELECT * FROM SECTOR
3  WHERE nombre_sector LIKE '%Vega%';
```

Figura 53: Código de MySQL "Consultas simples, Comunas con nombre mayor a 10 caracter"

Diferencias: No se realizaron cambios en esta consulta, ya que la sintaxis es compatible en ambos motores.

cod_sector	nombre_sector	cod_comuna
60	La Vega Chica	11
NULL	NULL	NULL

Tabla 44: Código de MySQL "Consulta simple 8"

```
1  -- Pagos de arriendo pendientes --
2  SELECT PA.cod_pago, PA.fecha_pago, PA.monto_pago
3  FROM PAGOS_ARRIENDO PA
4  WHERE PA.estado_pago = 'Pendiente';
```

Figura 54: Código de MySQL "Consultas simples, Comunas con nombre mayor a 10 caracter"

Diferencias: No se realizaron cambios en esta consulta, ya que la sintaxis es compatible en ambos motores.

cod_pago	fecha_pago	monto_pago
29	2024-09-05	900000.00
62	2024-06-05	1200000.00
NULL	NULL	NULL

Tabla 45: Código de MySQL "Consulta simple 9"

```
1  - Número de propiedades con precio de arriendo mayor a 600,000 --
2  SELECT COUNT(*) AS TotalPropiedades
3  FROM PROPIEDAD
4  WHERE precio_arriendo > 600000;
```

Figura 55: Código de MySQL "Consultas simples, Comunas con nombre mayor a 10 caracter"

Diferencias: No se realizaron cambios en esta consulta, ya que la sintaxis es compatible en ambos motores.

TotalPropiedades
6

Tabla 46: Código de MySQL "Consulta simple 10"

```
1  -- Propietarios con renta mayor a $2,000,000 o cuyo RUT comienza con '88' --
2  SELECT * FROM PROPIETARIO
3  WHERE renta > 2000000 OR rut_propietario LIKE '88%';
```

Figura 56: Código de MySQL "Consultas simples, Comunas con nombre mayor a 10 caracter"

Diferencias: No se realizaron cambios en esta consulta, ya que la sintaxis es compatible en ambos motores.

cod_propietario	nombre_propietario	rut_propietario	renta
25	Maria Silva	55.555.555-5	2100000.00
47	Sofia Mendez	77.777.777-7	2200000.00
58	Patricia Vega	88.888.888-8	1800000.00
69	Alberto Vargas	99.999.999-9	2300000.00
NULL	NULL	NULL	NULL

Tabla 47: Código de MySQL "Consulta simple 11"

```
1  -- Reparaciones con costo mayor o igual a 150,000 --
2  SELECT * FROM REPARACIONES
3  WHERE costo >= 150000;
```

Figura 57: Código de MySQL "Consultas simples, Comunas con nombre mayor a 10 caracter"

Diferencias: No se realizaron cambios en esta consulta, ya que la sintaxis es compatible en ambos motores.

cod_reparaciones	descripciones	fecha_reparacion	costo	cod_propiedad
27	Reparación del Portón Frontal	2024-09-15	150000.00	23
49	Arreglo eléctrico del medidor	2024-09-25	200000.00	45
NULL	NULL	NULL	NULL	NULL

Tabla 48: Código de MySQL "Consulta simple 12"

```
1  -- Propiedades arrendadas con costo de arriendo menor a 1,000,000 --
2  SELECT cod_arriendo, costo_final_arriendo, cod_propietario
3  FROM ARRIENDO
4  WHERE costo_final_arriendo < 1000000;
```

Figura 58: Código de MySQL "Consultas simples, Comunas con nombre mayor a 10 caracter"

Diferencias: No se realizaron cambios en esta consulta, ya que la sintaxis es compatible en ambos motores.

cod_arriendo	costo_final_arriendo	cod_propietario
17	950000.00	14
28	900000.00	25
50	850000.00	47
NULL	NULL	NULL

Tabla 49: Código de MySQL "Consulta simple 13"

```
1  -- Arrendatarios cuyo nombre contiene la letra 'i' --
2  SELECT * FROM ARRENDATARIO
3  WHERE nombre_arrendatario LIKE '%i%';
```

Figura 59: Código de MySQL "Consultas simples, Comunas con nombre mayor a 10 caracter"

Diferencias: No se realizaron cambios en esta consulta, ya que la sintaxis es compatible en ambos motores.

cod_arrendatario	nombre_arrendatario	rut_arrendatario	renta
46	Ana Diaz	22.222.222-2	1700000.00
57	Luis Lopez	33.333.333-3	1600000.00
68	Miguel Sanchez	44.444.444-4	1550000.00
NULL	NULL	NULL	NULL

Tabla 50: Código de MySQL "Consulta simple 14"

```
1  -- Propiedades arrendadas y con precio de arriendo mayor a 700,000 --
2  SELECT * FROM PROPIEDAD
3  WHERE estado = 'Arrendada' AND precio_arriendo > 700000;
```

Figura 60: Código de MySQL "Consultas simples, Comunas con nombre mayor a 10 caracter"

Diferencias: No se realizaron cambios en esta consulta, ya que la sintaxis es compatible en ambos motores.

cod_propiedad	calle	numero	precio_arriendo	estado	cod_ubicacion
23	Calle Independencia	202	950000.00	Arrendada	20
NULL	NULL	NULL	NULL	NULL	NULL

Tabla 51: Código de MySQL "Consulta simple 15"

```
1  --- Suma de todas las rentas de propietarios --
2  SELECT SUM(renta) AS total_renta_propietarios
3  FROM PROPIETARIO;
```

Figura 61: Código de MySQL "Consultas simples, Comunas con nombre mayor a 10 caracter"

Diferencias: No se realizaron cambios en esta consulta, ya que la sintaxis es compatible en ambos motores.

total_renta_propietarios
12300000.00

Tabla 52: Código de MySQL "Consulta simple 16"

```
1  -- Suma de todas las rentas de arrendatarios --
2  SELECT SUM(renta) AS total_renta_arrendatarios
3  FROM ARRENDATARIO;
```

Figura 62: Código de MySQL "Consultas simples, Comunas con nombre mayor a 10 caracter"

Diferencias: No se realizaron cambios en esta consulta, ya que la sintaxis es compatible en ambos motores.

	total_renta_arrendatarios
▶	9050000.00

Tabla 53: Código de MySQL "Consulta simple 17"

```
1  -- Propiedades en el sector específico con código 10 --
2  SELECT * FROM PROPIEDAD
3  WHERE cod_ubicacion = 10;
```

Figura 63: Código de MySQL "Consultas simples, Comunas con nombre mayor a 10 caracter"

Diferencias: No se realizaron cambios en esta consulta, ya que la sintaxis es compatible en ambos motores.

	cod_propiedad	calle	numero	precio_arriendo	estado	cod_ubicacion
▶	12	Av. Las Torres	101	800000.00	Disponible	10
*	NULL	NULL	NULL	NULL	NULL	NULL

Tabla 54: Código de MySQL "Consulta simple 18"

```
1  -- Pagos realizados después de una fecha específica --
2  SELECT PA.cod_pago, PA.fecha_pago, PA.monto_pago
3  FROM PAGOS_ARRIENDO PA
4  WHERE PA.fecha_pago > '2024-08-01';
```

Figura 64: Código de MySQL "Consultas simples, Comunas con nombre mayor a 10 caracter"

Diferencias: No se realizaron cambios en esta consulta, ya que la sintaxis es compatible en ambos motores.

	cod_pago	fecha_pago	monto_pago
▶	18	2024-10-05	950000.00
	29	2024-09-05	900000.00
	40	2024-08-05	1000000.00
*	NULL	NULL	NULL

Tabla 55: Código de MySQL "Consulta simple 19"

```
1  --- Arrendatarios con renta entre 1,500,000 y 1,800,000 --
2  SELECT * FROM ARRENDATARIO
3  WHERE renta BETWEEN 1500000 AND 1800000;
```

Figura 65: Código de MySQL "Consultas simples, Comunas con nombre mayor a 10 caracter"

Diferencias: No se realizaron cambios en esta consulta, ya que la sintaxis es compatible en ambos motores.

	cod_arrendatario	nombre_arrendatario	rut_arrendatario	renta
▶	13	Juan Perez	12.345.678-9	1500000.00
	46	Ana Diaz	22.222.222-2	1700000.00
	57	Luis Lopez	33.333.333-3	1600000.00
	68	Miguel Sanchez	44.444.444-4	1550000.00
*	NULL	NULL	NULL	NULL

Tabla 56: Código de MySQL "Consulta simple 20"

5. Consultas SQL Anidadas

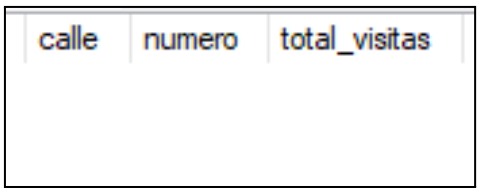
A continuación se presentan 15 consultas anidadas, las cuales también conocidas como sub consultas, son consultas que se incluyen dentro de otra consulta. Estas sub consultas permiten realizar operaciones más complejas y específicas al utilizar el resultado de una consulta interna como parte de la consulta externa, presentando las siguientes consultas:

5.1 Consultas de SQL Server Microsoft

```
1  --|1|Propiedades con más de 2 visitas realizadas--
2  SELECT P.calle, P.numero, COUNT(VP.cod_visitas) AS total_visitas
3  FROM PROPIEDAD P
4  LEFT JOIN VISITAS_PROPIEDAD VP ON VP.cod_propiedad = P.cod_propiedad
5  GROUP BY P.calle, P.numero
6  HAVING COUNT(VP.cod_visitas) > 2;
```

Figura 66: Código de SQL Server "Consulta anidada 1"

Esta consulta obtiene las propiedades que han recibido más de dos visitas. Se realiza una unión externa izquierda *(LEFT JOIN*) entre las tablas *PROPIEDAD* y *VISITAS_PROPIEDAD* para incluir todas las propiedades, incluso aquellas sin visitas registradas. Luego, se agrupan los resultados por dirección y número de cada propiedad, contando las visitas realizadas a cada una. La cláusula *HAVING* filtra los resultados para mostrar únicamente aquellas propiedades con un conteo de visitas mayor a dos, permitiendo identificar propiedades con alta actividad de visitas, la cual ninguna no hay ninguna visita mayor a dos, porque son todas por una visita en propiedad.



calle	numero	total_visitas

Tabla 57: Código de SQL Server "Consulta anidada 1"

```
1  --|2|Lista de propietarios que tienen más de una propiedad arrendada--
2  SELECT PR.nombre_propietario, COUNT(DISTINCT P.cod_propiedad) AS total_propiedades
3  FROM PROPIETARIO PR
4  LEFT JOIN ARRIENDO AR ON PR.cod_propietario = AR.cod_propietario
5  LEFT JOIN PROPIEDAD P ON AR.cod_propiedad = P.cod_propiedad
6  WHERE P.estado = 'Arrendada'
7  GROUP BY PR.nombre_propietario
8  HAVING COUNT(DISTINCT P.cod_propiedad) > 1;
```

Figura 67: Código de SQL Server "Consulta anidada 2"

Esta consulta genera una lista de propietarios que poseen más de una propiedad en estado de arriendo. Utiliza una unión externa izquierda (*LEFT JOIN)* entre las tablas *PROPIETARIO* y *ARRIENDO* para incluir a todos los propietarios y las propiedades que tengan arrendadas. Luego, se aplica otra unión con la tabla *PROPIEDAD* para obtener detalles de las propiedades arrendadas, filtrando por aquellas cuyo estado es 'Arrendada'. Finalmente, la consulta agrupa los resultados por el nombre de cada propietario y cuenta el número distinto de propiedades arrendadas por cada uno, mostrando sólo aquellos que tienen más de una propiedad arrendada mediante la cláusula *HAVING*. La cual muestra solo muestra ninguna, debido a que solo dos propietarios coinciden pero no tienen más de dos propiedades.

nombre_propietario	total_propiedades

Tabla 58: Código de SQL Server "Consulta anidada 2"

```
1  --|3|Calcular el promedio de precios de arriendo por comuna
2  SELECT C.nombre_comuna, AVG(P.precio_arriendo) AS promedio_precio
3  FROM COMUNA C
4  LEFT JOIN SECTOR S ON C.cod_comuna = S.cod_comuna
5  LEFT JOIN PROPIEDAD P ON S.cod_sector = P.cod_ubicacion
6  GROUP BY C.nombre_comuna;
```

Figura 68: Código de SQL Server "Consulta anidada 3"

Esta consulta calcula el promedio de los precios de arriendo de propiedades por cada comuna. Utiliza una unión externa izquierda (*LEFT JOIN*) entre las tablas *COMUNA* y *SECTOR* para asociar cada comuna con sus sectores, y luego otra unión izquierda con la tabla *PROPIEDAD* para relacionar los sectores con las propiedades ubicadas en ellos. Esto permite incluir todas las comunas, incluso aquellas sin propiedades registradas. Finalmente, la consulta agrupa los resultados por el nombre de la comuna y calcula el promedio del precio de arriendo de las propiedades en cada una, devolviendo el promedio de arriendo por comuna.

	nombre_comuna	promedio_precio
1	Cerro Navia	810000.000000
2	La Florida	950000.000000
3	Las Condes	1200000.000000
4	Lo Barnechea	650000.000000
5	Providencia	750000.000000

Tabla 59: Código de SQL Server "Consulta anidada 3"

```
1  --|4|Propiedades arrendadas por arrendatarios con renta mayor a $1 500 000--
2  SELECT P.calle, P.numero, A.nombre_arrendatario, A.renta
3  FROM PROPIEDAD P
4  INNER JOIN ARRIENDO AR ON P.cod_propiedad = AR.cod_propiedad
5  INNER JOIN ARRENDATARIO A ON AR.cod_arrendatario = A.cod_arrendatario
6  WHERE A.renta > 1500000.00;
```

Figura 69: Código de SQL Server "Consulta anidada 4"

Esta consulta obtiene las propiedades arrendadas cuyos arrendatarios tienen una renta superior a $1500000. Utiliza uniones internas (*INNER JOIN*) entre las tablas *PROPIEDAD* y *ARRIENDO* para identificar las propiedades en arriendo, y entre *ARRIENDO* y *ARRENDATARIO* para asociar cada arriendo con su arrendatario correspondiente. Al aplicar la condición en la cláusula *WHERE*, se filtran los resultados para incluir solo aquellos arrendatarios con una renta mayor a $1.500.000. La consulta devuelve la dirección y número de cada propiedad, junto con el nombre y renta del arrendatario.

	calle	numero	nombre_arrendatario	renta
1	Calle Los Pinos	404	Ana Diaz	1700000.00
2	Calle Irarrazabal	505	Luis Lopez	1600000.00
3	Calle Bellavista	606	Miguel Sanchez	1550000.00

Tabla 60: Código de SQL Server "Consulta anidada 4"

```
1  --|5|Obtener las propiedades arrendadas en 'Las Condes' con reparaciones realizadas--
2  SELECT P.calle, P.numero, R.descripciones, R.fecha_reparacion
3  FROM COMUNA C
4  LEFT JOIN SECTOR S ON C.cod_comuna = S.cod_comuna
5  LEFT JOIN PROPIEDAD P ON S.cod_sector = P.cod_ubicacion
6  LEFT JOIN REPARACIONES R ON P.cod_propiedad = R.cod_propiedad
7  WHERE C.nombre_comuna = 'Las Condes' AND P.estado = 'Arrendada';
```

Figura 70: Código de SQL Server "Consulta anidada 5"

Esta consulta recupera las propiedades en estado de arriendo ubicadas en la comuna de 'Las Condes' y que tienen reparaciones registradas. Utiliza uniones externas izquierdas (*LEFT JOIN*) para conectar las tablas *COMUNA*, *SECTOR*, *PROPIEDAD* y *REPARACIONES*, lo que asegura que se incluyan todas las propiedades y sus reparaciones, incluso si algunas propiedades no tienen reparaciones. La cláusula *WHERE* filtra los resultados para seleccionar solo aquellas propiedades que están arrendadas y se ubican en 'Las Condes'. La consulta devuelve la dirección

y número de cada propiedad, junto con las descripciones y fechas de las reparaciones realizadas. La cual no hay ninguna, porque solo coincide con una propiedad de 'Las Condes' pero no en *estado* ,ya que la propiedad está en un estado 'Disponible'

calle	numero	descripciones	fecha_reparacion

Tabla 61: Código de SQL Server "Consulta anidada 5"

```
1  --|6|Obtener los arrendatarios con pagos pendientes en propiedades de 'Sector Poniente'--
2  SELECT A.nombre_arrendatario, PA.monto_pago
3  FROM ARRENDATARIO A
4  RIGHT JOIN ARRIENDO AR ON A.cod_arrendatario = AR.cod_arrendatario
5  RIGHT JOIN PAGOS_ARRIENDO PA ON AR.cod_arriendo = PA.cod_arriendo
6  RIGHT JOIN PROPIEDAD P ON AR.cod_propiedad = P.cod_propiedad
7  RIGHT JOIN SECTOR S ON P.cod_ubicacion = S.cod_sector
8  WHERE S.nombre_sector = 'Sector Poniente' AND PA.estado_pago = 'Pendiente';
```

Figura 71: Código de SQL Server "Consulta anidada 6"

Esta consulta obtiene a los arrendatarios que tienen pagos pendientes en propiedades situadas en el 'Sector Poniente'. Se utilizan uniones externas derechas (*RIGHT JOIN*) para conectar las tablas *ARRENDATARIO*, *ARRIENDO*, *PAGOS_ARRIENDO*, *PROPIEDAD* y *SECTOR*, asegurando que se incluyan todas las propiedades del sector, incluso si algunos arrendatarios o pagos no tienen coincidencias en otras tablas. La cláusula *WHERE* filtra los resultados para incluir solo las propiedades ubicadas en 'Sector Poniente' y cuyos pagos están en estado 'Pendiente'. La consulta devuelve el nombre del arrendatario y el monto del pago pendiente.

	nombre_arrendatario	monto_pago
1	Luis Lopez	1200000.00

Tabla 62: Código de SQL Server "Consulta anidada 6"

```
1  --|7|Obtener las propiedades disponibles que han tenido reparaciones recientes--
2  SELECT P.calle, P.numero, R.descripciones, R.fecha_reparacion
3  FROM PROPIEDAD P
4  FULL OUTER JOIN REPARACIONES R ON P.cod_propiedad = R.cod_propiedad
5  WHERE P.estado = 'Disponible' AND R.fecha_reparacion >= '2024-09-01';
```

Figura 72: Código de SQL Server "Consulta anidada 7"

Esta consulta busca las propiedades disponibles que han tenido reparaciones recientes, realizadas desde el 1 de septiembre de 2024. Utiliza una unión externa completa (*FULL OUTER JOIN*) entre las tablas *PROPIEDAD* y *REPARACIONES*, lo que permite incluir todas las propiedades, independientemente de si tienen reparaciones registradas, y todas las reparaciones, incluso si no están asociadas a propiedades específicas. La cláusula *WHERE* filtra los resultados para seleccionar únicamente aquellas propiedades que están en estado 'Disponible' y que tienen

reparaciones cuya fecha es igual o posterior a la indicada. La consulta devuelve la calle, el número de la propiedad, la descripción de la reparación y su fecha.

	calle	numero	descripciones	fecha_reparacion
1	Av. Las Torres	101	Falla en las tuberias del baño	2024-09-10
2	Av. Providencia	303	Obstrucción en las Tuberias del Gas	2024-09-20
3	Calle Irarrazabal	505	Reparación de ventanas	2024-09-30
4	Calle Bellavista	606	Cambio de pisos	2024-10-05

Tabla 63: Código de SQL Server "Consulta anidada 7"

```
1  --|8|Obtener los propietarios con propiedades que tienen reparaciones mayores a $100 000--
2  SELECT P.cod_propiedad, PR.nombre_propietario, R.costo
3  FROM PROPIETARIO PR
4  LEFT JOIN ARRIENDO AR ON PR.cod_propietario = AR.cod_propietario
5  LEFT JOIN PROPIEDAD P ON AR.cod_propiedad = P.cod_propiedad
6  LEFT JOIN REPARACIONES R ON P.cod_propiedad = R.cod_propiedad
7  WHERE R.costo > 100000.00;
```

Figura 73: Código de SQL Server "Consulta anidada 8"

Esta consulta recupera la lista de propietarios que poseen propiedades con reparaciones cuyo costo supera los $100.000. Utiliza uniones externas izquierdas (*LEFT JOIN*) para conectar las tablas *PROPIETARIO*, *ARRIENDO*, *PROPIEDAD* y *REPARACIONES*, garantizando que todos los propietarios y propiedades se incluyan, incluso si algunos registros no tienen coincidencias en las tablas relacionadas. La cláusula *WHERE* filtra los resultados para mostrar únicamente las reparaciones con un costo superior a $100.000. La consulta devuelve el código de propiedad, el nombre del propietario y el costo de la reparación, identificando propiedades con reparaciones significativas asociadas a cada propietario.

	cod_propiedad	nombre_propietario	costo
1	23	Maria Silva	150000.00
2	34	Jorge Salinas	120000.00
3	45	Sofia Mendez	200000.00

Tabla 64: Código de SQL Server "Consulta anidada 8"

```
1  --|9|Listar los arrendatarios con propiedades en 'Lo Barnechea' y pagos pendientes--
2  SELECT A.nombre_arrendatario, P.calle, PA.monto_pago
3  FROM ARRENDATARIO A
4  LEFT JOIN ARRIENDO AR ON A.cod_arrendatario = AR.cod_arrendatario
5  LEFT JOIN PAGOS_ARRIENDO PA ON AR.cod_arriendo = PA.cod_arriendo
6  LEFT JOIN PROPIEDAD P ON AR.cod_propiedad = P.cod_propiedad
7  LEFT JOIN SECTOR S ON P.cod_ubicacion = S.cod_sector
8  LEFT JOIN COMUNA C ON S.cod_comuna = C.cod_comuna
9  WHERE C.nombre_comuna = 'Lo Barnechea' AND PA.estado_pago = 'Pendiente';--Esta vacia porque no coincide con
   Pendiente, solo Pagado
```

Figura 74: Código de SQL Server "Consulta anidada 9"

Esta consulta busca arrendatarios que tienen propiedades en la comuna de 'Lo Barnechea' y cuyos pagos están pendientes. Utiliza uniones externas izquierdas (*LEFT JOIN*) para conectar las tablas *ARRENDATARIO*, *ARRIENDO*, *PAGOS_ARRIENDO*, *PROPIEDAD*, *SECTOR* y *COMUNA*, asegurando que se incluyan todos los arrendatarios y sus propiedades, incluso si algunos registros no tienen coincidencias en otras tablas. La cláusula *WHERE* filtra los resultados para seleccionar solo las propiedades ubicadas en 'Lo Barnechea' y con pagos en estado 'Pendiente'. La consulta devuelve el nombre del arrendatario, la calle de la propiedad y el monto del pago pendiente, destacando arrendatarios con pagos pendientes en esa comuna. La tabla saldra ningun dato, debido a que coincide con una propiedad de l;a comuna de 'Lo Barnechea', pero en el *estado_pago* no, debido a que se busca 'Pendiente', el cual solo se encuentra el 'Pagado'.

nombre_arrendatario	calle	monto_pago

Tabla 65: Código de SQL Server "Consulta anidada 9"

```
1  --|10|Obtener el n mero de propiedades en 'La Florida' con reparaciones mayores a $100 000--
2  SELECT COUNT(P.cod_propiedad) AS total_propiedades
3  FROM COMUNA C
4  LEFT JOIN SECTOR S ON C.cod_comuna = S.cod_comuna
5  LEFT JOIN PROPIEDAD P ON S.cod_sector = P.cod_ubicacion
6  LEFT JOIN REPARACIONES R ON P.cod_propiedad = R.cod_propiedad
7  WHERE C.nombre_comuna = 'La Florida' AND R.costo > 100000.00;
```

Figura 75: Código de SQL Server "Consulta anidada 10"

Esta consulta cuenta el número de propiedades en la comuna de 'La Florida' que tienen reparaciones cuyo costo supera los $100.000. Para lograrlo, utiliza uniones externas izquierdas (*LEFT JOIN*) para conectar las tablas *COMUNA*, *SECTOR*, *PROPIEDAD* y *REPARACIONES*, asegurando que todas las comunas y sus sectores se incluyan, incluso si no todas las propiedades tienen reparaciones registradas. La cláusula *WHERE* filtra los resultados para seleccionar solo las propiedades ubicadas en 'La Florida' con reparaciones que superan el costo indicado. La consulta devuelve el conteo de tales propiedades, identificado en la salida como *total_propiedades*. La cual solo hay un total de uno en las propiedades ubicadas en 'La Florida'.

	total_propiedades
1	1

Tabla 66: Código de SQL Server "Consulta anidada 10"

```
1  --|11|Obtener (solo una) la propiedad m s cara en 'Las Condes' que est  disponible--
2  SELECT TOP 1 P.calle, P.numero, P.precio_arriendo
```

```
3  FROM COMUNA C
4  LEFT JOIN SECTOR S ON C.cod_comuna = S.cod_comuna
5  LEFT JOIN PROPIEDAD P ON S.cod_sector = P.cod_ubicacion
6  WHERE C.nombre_comuna = 'Las Condes' AND P.estado = 'Disponible'
7  ORDER BY P.precio_arriendo DESC;
```

Figura 76: Código de SQL Server "Consulta anidada 11"

Esta consulta busca obtener la propiedad disponible más cara ubicada en la comuna de 'Las Condes'. Utiliza uniones externas izquierdas (*LEFT JOIN*) entre las tablas *COMUNA*, *SECTOR* y *PROPIEDAD* para conectar cada comuna con sus respectivos sectores y propiedades, permitiendo que todas las comunas se incluyan, incluso si no todas las propiedades tienen coincidencias en cada nivel. La instrucción *TOP 1* selecciona solo la propiedad con el precio de arriendo más alto, mientras que la cláusula *WHERE* (que falta en el código proporcionado) debería incluirse para filtrar las propiedades de la comuna 'Las Condes' y aquellas que están en estado 'Disponible'.

	calle	numero	precio_arriendo
1	Av. Providencia	303	1200000.00

Tabla 67: Código de SQL Server "Consulta anidada 11"

```
1  --|12|Obtener el pago mas reciente realizado por el arrendatario 'Luis Lopez'--
2  SELECT TOP 1 PA.fecha_pago, PA.monto_pago
3  FROM ARRENDATARIO A
4  LEFT JOIN ARRIENDO AR ON A.cod_arrendatario = AR.cod_arrendatario
5  LEFT JOIN PAGOS_ARRIENDO PA ON AR.cod_arriendo = PA.cod_arriendo
6  WHERE A.nombre_arrendatario = 'Luis Lopez'
7  ORDER BY PA.fecha_pago DESC;
```

Figura 77: Código de SQL Server "Consulta anidada 12"

Esta consulta recupera el pago más reciente realizado por el arrendatario 'Luis Lopez'. Utiliza uniones externas izquierdas (*LEFT JOIN*) para conectar las tablas *ARRENDATARIO*, *ARRIENDO* y *PAGOS_ARRIENDO*, lo que permite incluir todos los arrendatarios y sus pagos, incluso si algunos no tienen coincidencias en todas las tablas. La cláusula *WHERE* filtra los resultados para seleccionar únicamente los registros relacionados con 'Luis Lopez'. Luego, la instrucción *TOP 1* junto con *ORDER BY PA.fecha_pago DESC* ordena los pagos en orden descendente por fecha, devolviendo solo el pago más reciente del arrendatario. La consulta muestra la fecha y el monto del último pago realizado por Luis Lopez.

	fecha_pago	monto_pago
1	2024-06-05	1200000.00

Tabla 68: Código de SQL Server "Consulta anidada 12"

```
1  --|13|Obtener el total de rentas generadas por propiedades en 'Sector Sur'--
2  SELECT SUM(AR.costo_final_arriendo) AS total_renta
3  FROM SECTOR S
4  LEFT JOIN PROPIEDAD P ON S.cod_sector = P.cod_ubicacion
5  LEFT JOIN ARRIENDO AR ON P.cod_propiedad = AR.cod_propiedad
6  WHERE S.nombre_sector = 'Sector Sur';
```

Figura 78: Código de SQL Server "Consulta anidada 13"

Esta consulta calcula el total de rentas generadas por propiedades en el 'Sector Sur'. Utiliza uniones externas izquierdas (*LEFT JOIN*) entre las tablas *SECTOR*, *PROPIEDAD* y *ARRIENDO*, permitiendo que todos los sectores se incluyan en el resultado, incluso si algunas propiedades no tienen coincidencias en otras tablas. La cláusula *WHERE* filtra los resultados para incluir únicamente aquellas propiedades ubicadas en el 'Sector Sur'. Luego, la función *SUM* se aplica sobre el campo *costo_final_arriendo* de la tabla *ARRIENDO* para calcular el total de las rentas generadas en ese sector específico. La consulta devuelve este valor como *total_renta*.

	total_renta
1	1000000.00

Tabla 69: Código de SQL Server "Consulta anidada 13"

```
1  --|14|Obtener el n mero de visitas realizadas por personas con Rut que empiecen con '21'--
2  SELECT COUNT(VP.cod_visitas) AS total_visitas
3  FROM VISITAS_PROPIEDAD VP
4  WHERE VP.rut_visita LIKE '21%';
```

Figura 79: Código de SQL Server "Consulta anidada 14"

Esta consulta cuenta el número de visitas realizadas por personas cuyo rut comienza con '21'. Para ello, selecciona la tabla *VISITAS_PROPIEDAD* y utiliza la función *COUNT* sobre el campo cod_visitas para calcular el total de registros que cumplen con la condición especificada. La cláusula *WHERE* emplea el operador *LIKE* con el patrón '21%', que busca todos los ruts que comiencen con '21'. El resultado de la consulta se devuelve en la columna *total_visitas*, mostrando el número de visitas que cumplen con este criterio.

	total_visitas
1	1

Tabla 70: Código de SQL Server "Consulta anidada 14"

```
1  --|15|Obtener los arrendatarios con m s de una propiedad arrendada--
2  SELECT A.nombre_arrendatario, COUNT(AR.cod_arriendo) AS total_arriendos
3  FROM ARRENDATARIO A
4  LEFT JOIN ARRIENDO AR ON A.cod_arrendatario = AR.cod_arrendatario
```

```
5  GROUP BY A.nombre_arrendatario
6  HAVING COUNT(AR.cod_arriendo) > 1;--si es >0 todos tienen un solo arrendatario, lo cual >1 no hay ninguno
```

Figura 80: Código de SQL Server "Consulta anidada 15"

Esta consulta identifica a los arrendatarios que tienen más de una propiedad arrendada. Utiliza una unión externa izquierda (*LEFT JOIN*) entre las tablas *ARRENDATARIO* y *ARRIENDO* para incluir todos los arrendatarios, incluso si algunos no tienen registros de arriendo asociados. La consulta agrupa los resultados por el nombre del arrendatario y utiliza la función *COUNT* sobre *cod_arriendo* para contar el número de propiedades arrendadas por cada arrendatario. La cláusula *HAVING* filtra los resultados, mostrando sólo aquellos arrendatarios con un conteo superior a uno en el campo *cod_arriendo*. Esto devuelve los nombres de los arrendatarios que tienen más de una propiedad arrendada y el total de arriendos de cada uno. El cual no hay ningún dato, debido a que todos los arrendatarios solo arriendan una propiedad con su código de arriendo, el cual ninguno tiene más de dos arriendos.

nombre_arrendatario	total_arriendos

Tabla 71: Código de SQL Server "Consulta anidada 15"

5.2 Consultas de MySQL Anidadas

A continuación, se mostrarán las versiones de las consultas anidadas en MySQL con sus respectivos cambios:

```
1  -- Propiedades con más de 2 visitas realizadas --
2  SELECT P.calle, P.numero, COUNT(VP.cod_visitas) AS total_visitas
3  FROM PROPIEDAD P
4  LEFT JOIN VISITAS_PROPIEDAD VP ON VP.cod_propiedad = P.cod_propiedad
5  GROUP BY P.calle, P.numero
6  HAVING COUNT(VP.cod_visitas) > 2;
```

Figura 81: Código de MySQL Server "Consulta anidada 1"

Diferencias: No se realizaron cambios en esta consulta, ya que la sintaxis es compatible en ambos motores.

	calle	numero	total_visitas

Tabla 72: Código de MySQL Server "Consulta anidada 1"

```
1  -- |2| Lista de propietarios que tienen más de una propiedad arrendada
2  SELECT PR.nombre_propietario, COUNT(DISTINCT P.cod_propiedad) AS total_propiedades
3  FROM PROPIETARIO PR
4  LEFT JOIN ARRIENDO AR ON PR.cod_propietario = AR.cod_propietario
5  LEFT JOIN PROPIEDAD P ON AR.cod_propiedad = P.cod_propiedad
6  WHERE P.estado = 'Arrendada'
7  GROUP BY PR.nombre_propietario
8  HAVING COUNT(DISTINCT P.cod_propiedad) > 1;
```

Figura 82: Código de MySQL Server "Consulta anidada 2"

Diferencias: No se realizaron cambios en esta consulta, ya que la sintaxis es compatible en ambos motores.

	nombre_propietario	total_propiedades

Tabla 73: Código de MySQL Server "Consulta anidada 2"

```
1  -- |3| Calcular el promedio de precios de arriendo por comuna
2  SELECT C.nombre_comuna, AVG(P.precio_arriendo) AS promedio_precio
3  FROM COMUNA C
4  LEFT JOIN SECTOR S ON C.cod_comuna = S.cod_comuna
5  LEFT JOIN PROPIEDAD P ON S.cod_sector = P.cod_ubicacion
6  GROUP BY C.nombre_comuna;
```

Figura 83: Código de MySQL Server "Consulta anidada 3"

Diferencias: No se realizaron cambios en esta consulta, ya que la sintaxis es compatible en ambos motores.

	nombre_comuna	promedio_precio
▶	Cerro Navia	810000.000000
	La Florida	950000.000000
	Las Condes	1200000.000000
	Lo Barnechea	650000.000000
	Providencia	750000.000000

Tabla 74: Código de MySQL Server "Consulta anidada 3"

```
1  -- |4| Propiedades arrendadas por arrendatarios con renta mayor a $1,500,000
2  SELECT P.calle, P.numero, A.nombre_arrendatario, A.renta
3  FROM PROPIEDAD P
4  INNER JOIN ARRIENDO AR ON P.cod_propiedad = AR.cod_propiedad
```

```
5  INNER JOIN ARRENDATARIO A ON AR.cod_arrendatario = A.cod_arrendatario
6  WHERE A.renta > 1500000.00;
```

Figura 84: Código de MySQL Server "Consulta anidada 4"

Diferencias: No se realizaron cambios en esta consulta, ya que la sintaxis es compatible en ambos motores.

	calle	numero	nombre_arrendatario	renta
	Calle Los Pinos	404	Ana Diaz	1700000.00
	Calle Irarrazabal	505	Luis Lopez	1600000.00
▶	Calle Bellavista	606	Miguel Sanchez	1550000.00

Tabla 75: Código de MySQL Server "Consulta anidada 4"

```
1  -- |5| Obtener las propiedades arrendadas en 'Las Condes' con reparaciones realizadas
2  SELECT P.calle, P.numero, R.descripciones, R.fecha_reparacion
3  FROM COMUNA C
4  LEFT JOIN SECTOR S ON C.cod_comuna = S.cod_comuna
5  LEFT JOIN PROPIEDAD P ON S.cod_sector = P.cod_ubicacion
6  LEFT JOIN REPARACIONES R ON P.cod_propiedad = R.cod_propiedad
7  WHERE C.nombre_comuna = 'Las Condes' AND P.estado = 'Arrendada';
```

Figura 85: Código de MySQL Server "Consulta anidada 5"

Diferencias: No se realizaron cambios en esta consulta, ya que la sintaxis es compatible en ambos motores.

	calle	numero	descripciones	fecha_reparacion

Tabla 76: Código de MySQL Server "Consulta anidada 5"

```
1  -- |6| Obtener los arrendatarios con pagos pendientes en propiedades de 'Sector Poniente'
2  SELECT A.nombre_arrendatario, PA.monto_pago
3  FROM ARRENDATARIO A
4  RIGHT JOIN ARRIENDO AR ON A.cod_arrendatario = AR.cod_arrendatario
5  RIGHT JOIN PAGOS_ARRIENDO PA ON AR.cod_arriendo = PA.cod_arriendo
6  RIGHT JOIN PROPIEDAD P ON AR.cod_propiedad = P.cod_propiedad
7  RIGHT JOIN SECTOR S ON P.cod_ubicacion = S.cod_sector
8  WHERE S.nombre_sect6or = 'Sector Poniente' AND PA.estado_pago = 'Pendiente';
```

Figura 86: Código de MySQL Server "Consulta anidada 6"

Diferencias: No se realizaron cambios en esta consulta, ya que la sintaxis es compatible en ambos motores.

	nombre_arrendatario	monto_pago
▶	Luis Lopez	1200000.00

Tabla 77: Código de MySQL Server "Consulta anidada 6"

```
1  -- |7| Obtener las propiedades disponibles que han tenido reparaciones recientes
2  SELECT P.calle, P.numero, R.descripciones, R.fecha_reparacion
3  FROM PROPIEDAD P
4  LEFT JOIN REPARACIONES R ON P.cod_propiedad = R.cod_propiedad
5  WHERE P.estado = 'Disponible' AND R.fecha_reparacion >= '2024-09-01';
```

Figura 87: Código de MySQL Server "Consulta anidada 7"

Cambio: MySQL no admite FULL OUTER JOIN, por lo que se reemplaza con LEFT JOIN. Este cambio limita los resultados a registros con propiedades únicamente en estado 'Disponible' que cumplen con la condición de reparación.

	calle	numero	descripciones	fecha_reparacion
▶	Av. Las Torres	101	Falla en las tuberias del baño	2024-09-10
	Av. Providencia	303	Obstrucción en las Tuberias del Gas	2024-09-20
	Calle Irarrazabal	505	Reparación de ventanas	2024-09-30
	Calle Bellavista	606	Cambio de pisos	2024-10-05

Tabla 78: Código de MySQL Server "Consulta anidada 7"

```
1  -- |8| Obtener los propietarios con propiedades que tienen reparaciones mayores a $100,000
2  SELECT P.cod_propiedad, PR.nombre_propietario, R.costo
3  FROM PROPIETARIO PR
4  LEFT JOIN ARRIENDO AR ON PR.cod_propietario = AR.cod_propietario
5  LEFT JOIN PROPIEDAD P ON AR.cod_propiedad = P.cod_propiedad
6  LEFT JOIN REPARACIONES R ON P.cod_propiedad = R.cod_propiedad
7  WHERE R.costo > 100000.00;
```

Figura 88: Código de MySQL Server "Consulta anidada 8"

Diferencias: No se realizaron cambios en esta consulta, ya que la sintaxis es compatible en ambos motores.

	cod_propiedad	nombre_propietario	costo
▶	23	Maria Silva	150000.00
	34	Jorge Salinas	120000.00
	45	Sofia Mendez	200000.00

Tabla 79: Código de MySQL Server "Consulta anidada 8"

```
1  -- |9| Listar los arrendatarios con propiedades en 'Lo Barnechea' y pagos pendientes
2  SELECT A.nombre_arrendatario, P.calle, PA.monto_pago
3  FROM ARRENDATARIO A
4  JOIN ARRIENDO AR ON A.cod_arrendatario = AR.cod_arrendatario
5  JOIN PAGOS_ARRIENDO PA ON AR.cod_arriendo = PA.cod_arriendo
6  JOIN PROPIEDAD P ON AR.cod_propiedad = P.cod_propiedad
7  JOIN SECTOR S ON P.cod_ubicacion = S.cod_sector
8  JOIN COMUNA C ON S.cod_comuna = C.cod_comuna
9  WHERE C.nombre_comuna = 'Lo Barnechea' AND PA.estado_pago = 'Pendiente';
```

Figura 89: Código de MySQL Server "Consulta anidada 9"

Diferencias: No se realizaron cambios en esta consulta, ya que la sintaxis es compatible en ambos motores.

	nombre_arrendatario	calle	monto_pago

Tabla 80: Código de MySQL Server "Consulta anidada 9"

```
1  -- |10| Obtener el número de propiedades en 'La Florida' con reparaciones mayores a $100,000
2  SELECT COUNT(P.cod_propiedad) AS total_propiedades
3  FROM COMUNA C
4  LEFT JOIN SECTOR S ON C.cod_comuna = S.cod_comuna
5  LEFT JOIN PROPIEDAD P ON S.cod_sector = P.cod_ubicacion
6  LEFT JOIN REPARACIONES R ON P.cod_propiedad = R.cod_propiedad
7  WHERE C.nombre_comuna = 'La Florida' AND R.costo > 100000.00;
```

Figura 90: Código de MySQL Server "Consulta anidada 10"

Diferencias: No se realizaron cambios en esta consulta, ya que la sintaxis es compatible en ambos motores.

	total_propiedades
▶	1

Tabla 81: Código de MySQL Server "Consulta anidada 10"

```
1  -- |11| Obtener la propiedad más cara en 'Las Condes' que esté disponible
2  SELECT P.calle, P.numero, P.precio_arriendo
3  FROM COMUNA C
```

```
4  LEFT JOIN SECTOR S ON C.cod_comuna = S.cod_comuna
5  LEFT JOIN PROPIEDAD P ON S.cod_sector = P.cod_ubicacion
6  WHERE C.nombre_comuna = 'Las Condes' AND P.estado = 'Disponible'
7  ORDER BY P.precio_arriendo DESC
8  LIMIT 1;
```

Figura 91: Código de MySQL Server "Consulta anidada 11"

Cambio: En MySQL, *TOP 1* se reemplaza con *LIMIT 1* para limitar la cantidad de resultados.

	calle	numero	precio_arriendo
▶	Av. Providencia	303	1200000.00

Tabla 82: Código de MySQL Server "Consulta anidada 11"

```
1  -- |12| Obtener el pago más reciente realizado por el arrendatario 'Luis Lopez'
2  SELECT PA.fecha_pago, PA.monto_pago
3  FROM ARRENDATARIO A
4  LEFT JOIN ARRIENDO AR ON A.cod_arrendatario = AR.cod_arrendatario
5  LEFT JOIN PAGOS_ARRIENDO PA ON AR.cod_arriendo = PA.cod_arriendo
6  WHERE A.nombre_arrendatario = 'Luis Lopez'
7  ORDER BY PA.fecha_pago DESC
8  LIMIT 1;
```

Figura 92: Código de MySQL Server "Consulta anidada 12"

Cambio: En MySQL, *LIMIT 1* sustituye *TOP 1*.

	fecha_pago	monto_pago
▶	2024-06-05	1200000.00

Tabla 83: Código de MySQL Server "Consulta anidada 12"

```
1  -- |13| Obtener el total de rentas generadas por propiedades en 'Sector Sur'
2  SELECT SUM(AR.costo_final_arriendo) AS total_renta
3  FROM SECTOR S
4  LEFT JOIN PROPIEDAD P ON S.cod_sector = P.cod_ubicacion
5  LEFT JOIN ARRIENDO AR ON P.cod_propiedad = AR.cod_propiedad
6  WHERE S.nombre_sector = 'Sector Sur';
```

Figura 93: Código de MySQL Server "Consulta anidada 13"

Diferencias: No se realizaron cambios en esta consulta, ya que la sintaxis es compatible en ambos motores.

	total_renta
▶	1000000.00

Tabla 84: Código de MySQL Server "Consulta anidada 13"

```
1  -- |14| Obtener el número de visitas realizadas por personas con RUT que empiecen con '21'
2  SELECT COUNT(VP.cod_visitas) AS total_visitas
3  FROM VISITAS_PROPIEDAD VP
4  WHERE VP.rut_visita LIKE '21%';
```

Figura 94: Código de MySQL Server "Consulta anidada 14"

Diferencias: No se realizaron cambios en esta consulta, ya que la sintaxis es compatible en ambos motores.

	total_visitas
▶	1

Tabla 85: Código de MySQL Server "Consulta anidada 14"

```
1  -- |15| Obtener los arrendatarios con más de una propiedad arrendada
2  SELECT A.nombre_arrendatario, COUNT(AR.cod_arriendo) AS total_arriendos
3  FROM ARRENDATARIO A
4  LEFT JOIN ARRIENDO AR ON A.cod_arrendatario = AR.cod_arrendatario
5  GROUP BY A.nombre_arrendatario
6  HAVING COUNT(AR.cod_arriendo) > 1;
```

Figura 95: Código de MySQL Server "Consulta anidada 15"

Diferencias: No se realizaron cambios en esta consulta, ya que la sintaxis es compatible en ambos motores.

	nombre_arrendatario	total_arriendos

Tabla 86: Código de MySQL Server "Consulta anidada 15"

6. Diferencias entre SQL Network y MySQL

La migración de código de SQL Server a MySQL Workbench implica realizar varias adaptaciones debido a diferencias de sintaxis y características entre ambos sistemas de gestión de bases de datos (SGBD). A continuación, se detallan las modificaciones realizadas para asegurar la compatibilidad del código en MySQL Workbench, manteniendo la funcionalidad original y optimizando el rendimiento en este nuevo entorno.

6.1 Adaptaciones Realizadas:

1. Funciones de Cadena para Longitud de Texto:

- SQL Server: Utiliza *LEN()* para obtener la longitud de una cadena de texto.
- MySQL: Utiliza *CHAR_LENGTH()* en lugar de *LEN()* para medir la longitud de una cadena.

2. Tipos de Datos ENUM:

- SQL Server: No soporta el tipo de dato *ENUM*. En su lugar, se usan restricciones *CHECK* o columnas de tipo *VARCHAR* con validación adicional.
- MySQL: Permite el tipo *ENUM*, simplificando la implementación de valores predeterminados en campos que requieren opciones limitadas.

3. Conversión de TOP a LIMIT:

- SQL Server: Usa *TOP* para limitar el número de registros en los resultados.
- MySQL: Emplea *LIMIT* para la misma funcionalidad.

4. Uniones Externas Completas (FULL OUTER JOIN)

- SQL Server: Admite *FULL OUTER JOIN* para retornar todos los registros cuando hay una coincidencia en cualquiera de las tablas.
- MySQL: No soporta directamente *FULL OUTER JOIN*. En su lugar, se combinan *LEFT JOIN* y *RIGHT JOIN* junto con la cláusula *UNION* para obtener resultados equivalentes.

5. Conversión de DATETIME a Formatos Compatibles

- SQL Server: Utiliza *DATETIME* para almacenar fechas y horas.
- MySQL: También admite *DATETIME*, pero se realizaron verificaciones para asegurar compatibilidad con el formato *YYYY-MM-DD HH:MM:SS* que MySQL interpreta directamente.

6. Operador de Concatenación de Cadenas

- SQL Server: Emplea más para concatenar cadenas.
- MySQL: Usa CONCAT() para concatenar múltiples cadenas.

7. Cláusula *RIGHT JOIN* en Consultas Específicas

- Algunas consultas en SQL Server utilizaban *RIGHT JOIN*, que en MySQL se mantuvo igual, pero se revisaron las condiciones de *JOIN* para asegurar que los datos se extraen correctamente y no haya ambigüedades.

8. Cambio en la Sintaxis de Comentarios

- **SQL Server**: Utiliza -- para comentarios de una sola línea.
- **MySQL**: También permite --, pero se verifica que no se incluyan espacios adicionales que podrían generar errores de sintaxis en MySQL Workbench.

7. Conclusión

La base de datos implementada en SQL Server y MySQL ofrece una mayor robustez y capacidad para manejar grandes volúmenes de datos en comparación con su versión original en Access. Esto permite al sistema crecer junto con el negocio, proporcionando la escalabilidad necesaria para soportar un incremento en el número de propiedades gestionadas, arrendamientos y usuarios del sistema. Además, los motores SQL Server y MySQL ofrecen capacidades avanzadas para la ejecución de consultas complejas, lo que facilita el análisis de datos a gran escala y la generación de informes detallados sobre el rendimiento del negocio, los flujos de ingresos y otros indicadores clave. Este aspecto es especialmente importante para la toma de decisiones estratégicas, ya que proporciona una visión completa y precisa del estado actual del negocio, permitiendo al corredor anticipar problemas, identificar oportunidades de mejora y optimizar sus operaciones.

El proceso de migración de la base de datos de Access a SQL Server y MySQL también es abordado, destacando las ventajas y desafíos de este tipo de migración. Se comparan los dos motores de base de datos en términos de rendimiento, escalabilidad, facilidad de uso y costo, proporcionando un análisis detallado que permite al lector entender las razones detrás de la elección de estas tecnologías. Esta comparativa es especialmente relevante para aquellos corredores de propiedades que buscan implementar una solución de base de datos similar, ya que les permite tomar decisiones informadas sobre la tecnología más adecuada para sus necesidades específicas.

Para terminar, este laboratorio de desarrollo de base de datos proporciona una solución completa para la gestión eficiente de la información en el negocio inmobiliario. Al organizar los datos de manera estructurada, garantizar la integridad de la información y ofrecer capacidades avanzadas de consulta y análisis, el sistema mejora significativamente la eficiencia operativa del corredor de propiedades, facilitando la toma de decisiones estratégicas y asegurándose que la información esté siempre disponible y actualizada. Además, el traspaso a motores de bases de datos más avanzados como SQL Server y MySQL asegura que el crezca junto con el negocio, proporcionando una solución óptima para el manejo de grandes volúmenes de datos.