Polytechnic University of Puerto Rico

San Juan, Puerto Rico

Electrical and Computer Engineering Department

CECS 4202-81 Database Systems

FA 2023

Assignment 2

Due Date: October 20, 2023

Total Points: 100

Student Name: Carlos Surillo

Student ID: 138098

Prof. Dr. Alfredo Cruz

<u>Part 1: Entity Relationship Diagrams (ERD) (40 points)</u>

Chapter 5

Checkpoint 5.1

1. **How would you identify a strong entity?**
   - → Strong entities almost always have a unique identifier that is a subset of all the attributes.
2. **How would you identify a weak entity?**
   - → Weak entities may not have a key attribute of their own, as they are dependent on a strong or regular entity for their existence.
3. **What kind of a relationship line (single or double) would be leading up to the weak entity in a Chen-like diagram?**
   - → Weak entities always have full or mandatory participation from the weak side toward the owner, so it has a double line from the weak side to the owner and a single line from the owner to the weak side.
4. **What kind of relationship does a weak entity have in a Chen-like model?**
   - → In the Chen-like model weak entities are enclosed in a double box, and the corresponding relationship to the owner in a double diamond.
5. **What is a partial key?**
   - → It identifies dependents, but not uniquely.

Chapter 6

Checkpoint 6.2

1. **What is a recursive relationship?**
   - → A recursive relationship is where the same entity participates more than once in different roles. Recursive relationships are also sometimes called unary relationships.
2. **How is the recursive relationship denoted diagrammatically in the Chenlike ER model?**
   - → Recursive relationships can only have partial participation in relationships, but the cardinality can be one-to-one, one-to-many, or many-to-many.

Chapter 7

Checkpoint 7.2

1. **Can all ternary relationships be expressed in the form of binary relationships? Explain.**
   - → No because there are situations in which a relationship inherently associates more than two entities, having a relationship that cannot adequately be captured by binary relationship
2. **Come up with some attributes and entities of a relationship that you think could be a ternary relationship. Can this relationship be expressed in the form of a binary relationship?**
   - → A ternary relationship could be one between a teacher, student and class subject. It could not be shown in the form of a binary relationship because it is dependent on these three attributes.

Chapter 8

Checkpoint 8.1

1. **What is a specialization? Come up with another example of a specialization.**
   - → Specialization is intended as a process whereby the subclass inherits all the properties of the superclass. An example would be and employee superclass and the type pf employee subclass, like a developer for example.
2. **What is a generalization? Come up with another example of a specialization.**
   - → A generalization relationship specifies that several types of entities with certain common attributes can be generalized into a higher-level entity class, a generic or superclass entity. An example would be lions, dogs, foxes generalized as mammals.
3. **What is a disjoint constraint? What symbol shows the disjoint constraint in EER diagrams?**
   - → It prevents overlapping between database elements. A “d” is used as its symbol.
4. **What is an overlap constraint? What symbol shows the overlap constraint in EER diagrams?**
   - → This means that the subclass entities may overlap, a superclass entity may be a member of more than one subclass of a specialization. An “o” is used as its symbol.

Chapter 9

Checkpoint 9.2

1. **What is the first mapping rule?**
    - Map the strong entities along with their atomic attributes.

Chapter 10

Checkpoint 10.2

1. **How is the "optional" relationship shown diagrammatically in the Barker/Oracle-like model?**
    - In the Barker/Oraclelike ER model, the optional attribute is shown without the "not null" depiction. Not null means that on *no* occasion would an instance of the entity exist without knowing the value of this mandatory attribute.

Part 2: SQL

A. Review Questions

Chapter 5

1. **How do you join tables in SQL?**
    - You join tables in SQL by including a condition in the WHERE clause to ensure that matching columns contain equal values.
2. **When must you qualify names in SQL commands? How do you qualify a column name?**
    - When there is potential ambiguity in listing column names, you must qualify the columns involved in the query. You qualify a column name by writing the table name followed by a period and column name.

3. **List two operators that you can use with subqueries as an alternate way of performing joins.**
    - The IN operator or the EXISTS operator.
4. **What is a nested subquery? In which order does SQL evaluate nested subqueries?**
    - A subquery within a subquery; SQL evaluates the queries from the innermost query to the outermost query.
5. **What is an alias? How do you specify an alias in SQL? Why would you use an alias?**
    - An alias is an alternate name that you can use in the rest of the statement. You create an alias by typing the name of the table, pressing the Spacebar, and then typing the name of the alias. One reason for using an alias is simplicity.

6. **How do you join a table to itself in SQL?**
    - You can treat the table as if it were two tables in the query by creating an alias, then joining them via the where clause.
7. **What command would you use to show all rows of two tables? How would you use it? What command would you use to show only common rows between two tables? How would you use it?**
    - The union of two tables uses the UNION operator to create a temporary table containing every row that is in either the first table, the second table, or both tables. The intersection of two tables uses the INTERSECT operator to create a temporary table containing all rows that are in both tables
8. **What does it mean for two tables to be union compatible?**
    - Two tables are union compatible when they have the same number of columns and their corresponding columns have identical data types and lengths.
9. **How do you use the ALL operator with a subquery?**
    - You can use the ALL operator with subqueries to produce a single column of numbers. When you precede the subquery by the ALL operator, the condition is true only if it satisfies all values produced by the subquery.
10. **How do you use the ANY operator with a subquery?**
    - You can use the ANY operator with subqueries to produce a single column of numbers. When you precede the subquery by the ANY operator, the condition is true if it satisfies any value produced by the subquery.
11. **Which rows are included in an inner join? What clause can you use to perform an inner join in SQL?**
    - Only the rows that satisfy the condition in the WHERE clause. In the FROM clause, list the first table, and then include an INNER JOIN clause that includes the name of the second table. Instead of a WHERE clause, use an ON clause containing the same condition that you would have included in the WHERE clause.
12. **Which rows are included in a left outer join? What clause can you use to perform a left outer join in SQL?**
    - In a left outer join, all rows from the table on the left (the table listed first in the query) are included regardless of whether they match rows from the table on the right (the table listed second in the query). Use Left Join and an ON clause containing the same condition that you would have included in the WHERE clause.
13. **Which rows are included in a right outer join? What clause can you use to perform a right outer join in SQL?**
    - In a right outer join, all rows from the table on the right are included regardless of whether they match rows from the table on the left. Use Right Join and an ON clause containing the same condition that you would have included in the WHERE clause.
14. **What is the formal name for the product of two tables? How do you form a product in SQL?**

- The product, formally called the Cartesian product, of two tables is the combination of all rows in the first table and all rows in the second table. To create a product you omit the WHERE clause.

Chapter 6

1. **Which command creates a new table?**
   - You describe the new table by using the CREATE TABLE command.
2. **Which command and clause adds an individual row to a table?**
   - The INSERT command.
3. **How do you add data from an existing table to another table?**
   - You can create a SELECT command to select the desired data from the table. By placing this SELECT command in an INSERT command, you can add the query results to a table.
4. **Which command changes data in a table?**
   - You can use the UPDATE command to change rows for which a specific condition is true.
5. **Which command removes rows from a table?**
   - You use the DELETE command to remove rows from a table.
6. **Which command makes updates permanent?**
   - You use COMMIT command to make permanent changes.
7. **Which command reverses updates? Which updates are reversed?**
   - You reverse the changes by executing the ROLLBACK command.
8. **How do you use the COMMIT and ROLLBACK commands to support transactions?**
   - Before beginning the updates for a transaction, commit any previous updates by executing the COMMIT command. Complete the updates for the transaction. If any update cannot be completed, execute the ROLLBACK command and discontinue the updates for the current transaction. If you can complete all updates successfully, execute the COMMIT command after completing the final update.

9. **What is the format of the SET clause that changes the value in a column to null in an UPDATE command?**
   - The command for changing a value in a column to null is exactly what it would be for changing any other value. You simply use the value NULL as the replacement value
10. **Which command and clause adds a column to an existing table?**
    - To add a new column, use the ADD clause of the ALTER TABLE command
11. **Which command and clause changes the characteristics of an existing column in a table?**

  - You can change the characteristics of existing columns by using the MODIFY clause of the ALTER TABLE command

12. **Which command deletes a table and all its data?**
    - You can delete a table by executing the DROP TABLE command.

Chapter 7

1. **What is a view?**
    - A view is a program's or an individual user's picture of the database. A view is a derived table because the data in it comes from one or more base tables
2. **Which command creates a view?**
    - To create a view, use the CREATE VIEW command, followed by the name of the view, the word AS, and then a query.
3. **What is a defining query?**
    - A defining query indicates the rows and columns to include in the view.
4. **What happens when a user retrieves data from a view?**
    - The data does not actually exist in this form, nor will it ever exist in this form, the query acts as a sort of window into the database.
5. **What are three advantages of using views?**
    - First, views provide data independence. The second benefit of using views is that different users can see the same data in different ways through their own views. The final benefit of using views is that a view can contain only those columns required by a given user.
6. **Which types of views cannot be updated?**
    - You cannot update views that involve statistics and calculations
7. **Which command deletes a view?**
    - The DROP VIEW command.
8. **Which command gives users access privileges to various portions of the database?**
    - The main mechanism for providing access to a database, is the GRANT command. The basic idea of the GRANT command is that the database administrator can grant different types of privileges to users and then revoke them later.
9. **Which command terminates previously granted privileges?**
    - The database administrator uses the REVOKE command to revoke privileges from users
10. **What is the purpose of an index?**
    - The main mechanism for increasing the efficiency with which data is retrieved from the database is the index, you can create and use an index to speed up the searching process significantly
11. **How do you create an index? How do you create a unique index? What is the difference between an index and a unique index?**

- The command used to create an index is CREATE INDEX. The command lists the name of the index and the table name on which the index is to be created. The column on which to create the index is listed in parentheses. To ensure the uniqueness of values in a nonprimary key column, you can create a unique index by using the CREATE UNIQUE INDEX command. The unique index has all the properties of indexes already discussed, along with one additional property: The DBMS rejects any update that causes a duplicate value

12. **Which command deletes an index?**
    - The command used to drop an index is DROP INDEX, which consists of the words DROP INDEX, followed by the name of the index, the keyword ON, and the table name
13. **Does the DBMS or the user make the choice of which index to use to accomplish a given task?**
    - The DBMS chooses which index to use to accomplish a given task
14. **Describe the information the DBMS maintains in the system catalog. What are the generic names for three tables in the catalog?**
    - Information about the tables in the database kept in the system catalog or the data dictionary is known as the INFORMATION_SCHEMA. INFORMATION_SCHEMA provides access to database metadata, information about the MySQL server such as the name of a database or table, the data type of a column, or access privileges. This section describes the types of items kept in the catalog and the way in which you can query it to access information about the database structure. The generic names for the three tables are SYSTABLES, SYSCOLUMNS, and SYSVIEWS.
15. **The CUSTOMER table contains a foreign key, REP_ID, that must match the primary key of the SALES_REP table. What type of update(s) to the CUSTOMER table would violate the foreign key constraint?**
    - An update that would violate the foreign key constraint is changing the values in the REP_ID to one that is not in the original primary key of the SALES_REP table.
16. **What is the INFORMATION_SCHEMA in MySQL?**
    - Information about the tables in the database kept in the system catalog or the data dictionary
17. **How is the system catalog updated?**
    - When users create, alter, or drop tables or create or drop indexes, the DBMS updates the sys- tem catalog automatically to reflect these changes. Users should not execute SQL queries to update the catalog directly because this might produce inconsistent information.
18. **What are integrity constraints?**
    - An integrity constraint is a rule for the data in the database. If a user enters data in the database that violates any of these integrity constraints, the database develops serious problems.
19. **How do you specify a general integrity constraint?**

-

20. **When would you usually specify primary key constraints? List two alternative methods to create a primary key?**
    -
21. **How do you specify a foreign key in MySQL?**
    - The general form for assigning a foreign key is FOREIGN KEY, the column name or names of the foreign key, the REFERENCES clause, and then the table name and column that the foreign key must match.

Chapter 8

1. **How do you display letters in uppercase in MySQL, Oracle, and SQL Server? How do you display letters in lowercase in MySQL, Oracle, and SQL Server?**
    - The UPPER function displays a value in uppercase letters, and to display a value in lowercase letters, you can use the LOWER function
2. **How do you round a number to a specific number of decimal places in MySQL, Oracle, and SQL Server? How do you remove everything to the right of the decimal place in MySQL, Oracle, and SQL Server?**
    - The ROUND function, which rounds a numeric value to a desired number of decimal places, and to remove everything to the right of the decimal point use the FLOOR function
3. **How do you add months to a date in MySQL, Oracle, and SQL Server? How do you add days to a date? How would you find the number of days between two dates?**
    - Oracle uses the ADD_MONTHS function, MySQL uses DATE_ADD and SQL Server usesDATEADD. To add a specific number of days to a date, you do not need a function. DATEDIFF() is used to determine the number of days between dates.
4. **How do you obtain today's date in MySQL, Oracle, and SQL Server?**
    - MySQL uses CURDATE() function, Oracle uses SYSDATE function, and SQl Server uses GETDATE() function.
5. **How do you concatenate values in character columns in MySQL, Oracle, and SQL Server?**
    - In MySQL to concatenate columns, you use the CONCAT() function. Oracle provides the CONCAT function; however, this function accepts only two string arguments to be concatenated. Similar to MySQL, the CONCAT function exists, and it can take more than two string arguments.
6. **Which function deletes extra spaces at the end of a value?**
    - To remove the extra spaces following the first name value, you use the RTRIM function.

7. **What are stored procedures? What purpose do they serve?**

    - A stored procedure is a query saved in a file, which is placed on the server. The DBMS compiles the stored procedure and creates an execution plan. It is useful for when you anticipate running a particular query often. Another reason for saving a query as a stored procedure is convenience.

8. **In which portion of a MySQL and PL/SQL procedure do you embed SQL commands?**
    - The procedural code, which contains the commands that specify the procedure's function, appears between the BEGIN and END commands
9. **Where do you declare variables in MySQL and PL/SQL procedures?**
    - The DECLARE statement within the procedure code defines a variable to store values to be used at a later stage. When defining variable names in MySQL, the name may consist of alphanumeric, dol-
lar signs, underscores, and number signs, but cannot exceed 64 characters. In addition, part of the declaration of a variable you must assign a data type.
10. **In PL/SQL, how do you assign variables the same type as a column in the database?**
    - You can ensure that a variable has the same data type as a particular column in a table by using the %TYPE attribute. To do so, you include the name of the table, followed by a period and the name of the column, and then %TYPE.
11. **How do you place the results of a SELECT command into variables in MySQL and PL/SQL?**
    - Use the INTO clause to place the results of a SELECT statement in variable.
12. **Can you use the INSERT, UPDATE, or DELETE commands to affect more than one row in MySQL and PL/SQL procedures?**
    - You can use an UPDATE or a DELETE command in MySQL to update or delete multiple rows. Using UPDATE and DELETE statements in PL/SQL procedure is similar to MySQL with exception of differences in syntax.
13. **How do you use a SELECT command that retrieves more than one row in a procedure?**
    - A cursor is a pointer to a row in the collection of rows retrieved by an SQL command. The cursor advances one row at a time to provide sequential, one-record-at-a-time access to the retrieved rows so MySQL can process the rows. By using a cursor, MySQL can process the set of retrieved rows as though they were records in a sequential file.
14. **Which command activates a cursor?**
    - The OPEN command opens the cursor and causes the query to be executed, making the results avail- able to the procedure
15. **Which command selects the next row in a cursor?**
    - Executing a FETCH command advances the cursor to the next
row in the set of rows retrieved by the query and places the contents of the row in the indicated variables
16. **Which command deactivates a cursor?**
    - The CLOSE command closes a cursor and deactivates it.
17. **What are triggers? What purpose do they serve?**

- A trigger is a procedure that is executed automatically in response to an associated data- base operation, such as an INSERT, UPDATE, or DELETE command. Unlike a stored pro- cedure, which is executed in response to a user request, a trigger is executed in response to a command that causes the associated database operation to occur.

18. What is the purpose of the INSERTED and DELETED tables in SQL Server?

- The INSERTED table is a temporary system table that contains a copy of the values that the last SQL command inserted. The DELETED table is a temporary system table that contains a copy of the values before they were affected

B. Using MySQL

I will show how I have configured the StayWell Student Accommodation Database using MySQL:

CREATE DATABASE StayWell_Students;

CREATE TABLE OFFICE(OFFICE_NUM DECIMAL(2,0) NOT NULL PRIMARY KEY, OFFICE_NAME char(25), ADDRESS char(25), AREA char(25), CITY char(25) ,STATE CHAR(2), ZIPCODE char(5));

INSERT INTO OFFICE VALUES('1','StayWell-Columbia City', '1135 N. Wells Avenue', 'Columbia City', 'Seattle', 'WA', '98118');

INSERT INTO OFFICE VALUES('2', 'StayWell-Georgetown', '986 S. Madison Rd', 'Georgetown', 'Seattle','WA', '98108');

CREATE TABLE OWNER(OWNER_NUM CHAR(5) NOT NULL PRIMARY KEY, LAST_NAME CHAR(25), FIRST_NAME CHAR(25), ADDRESS CHAR(25), CITY VARCHAR(25), STATE CHAR(2), ZIPCODE CHAR(5));

INSERT INTO OWNER VALUES('MO100','Moore','Elle-May','8006 W. Newport Ave.','Reno',

```
'NV','89508');

INSERT INTO OWNER VALUES('PA101','Patel','Makesh','7337 Sheffield St.','Seattle',
'WA','98119');

INSERT INTO OWNER VALUES('AK102','Aksoy','Ceyda','411 Griffin Rd.','Seattle',
'WA','98131');

INSERT INTO OWNER VALUES('CO103','Cole','Meerab','9486 Circle Ave.','Olympia',
'WA','98506');

INSERT INTO OWNER VALUES('KO104','Kowalczyk','Jakub','7431 S. Bishop
St.','Bellingham',  'WA','98226');

INSERT INTO OWNER VALUES('SI105','Sims','Haydon','527 Primrose Rd.','Portland',
'OR','97203');

INSERT INTO OWNER VALUES('BU106','Burke','Ernest','613 Old Pleasant St.','Twin Falls',
'ID','83303');

INSERT INTO OWNER VALUES('RE107','Redman','Seth','7681 Fordham St.','Seattle',
'WA','98119');

INSERT INTO OWNER VALUES('LO108','Lopez','Janine','9856 Pumpkin Hill Ln.','Everett',
'WA','98213');

INSERT INTO OWNER VALUES('BI109','Bianchi','Nicole','7990 Willow Dr.','New York',
'NY','10005');
```

```
INSERT INTO OWNER VALUES('JO110','Jones','Ammarah','730 Military Ave.','Seattle',

'WA','98126');


CREATE TABLE PROPERTY(PROPERTY_ID DECIMAL(2,0) NOT NULL PRIMARY
KEY, OFFICE_NUM DECIMAL(2,0), ADDRESS CHAR(25), SQR_FT
DECIMAL(5,0),BDRMS DECIMAL(2,0),FLOORS DECIMAL(2,0), MONTHLY_RENT
DECIMAL(6,2), OWNER_NUM CHAR(5));


INSERT INTO PROPERTY VALUES(1,'1','30 West Thomas Rd.',1600,3,1,1400,'BU106');
INSERT INTO PROPERTY VALUES(2,'1','782 Queen Ln.',2100,4,2,1900,'AK102');
INSERT INTO PROPERTY VALUES(3,'1','9800 Sunbeam Ave.',1005,2,1,1200,'BI109');
INSERT INTO PROPERTY VALUES(4,'1','105 North Illinois Rd.',1750,3,1,1650,'KO104');
INSERT INTO PROPERTY VALUES(5,'1','887 Vine Rd.',1125,2,1,1160,'SI105');
INSERT INTO PROPERTY VALUES(6,'1','8 Laurel Dr.',2125,4,2,2050,'MO100');
INSERT INTO PROPERTY VALUES(7,'2','447 Goldfield St.',1675,3,2,1700,'CO103');
INSERT INTO PROPERTY VALUES(8,'2','594 Leatherwood Dr.',2700,5,2,2750,'KO104');
INSERT INTO PROPERTY VALUES(9,'2','504 Windsor Ave.',700,2,1,1050,'PA101');
INSERT INTO PROPERTY VALUES(10,'2','891 Alton Dr.',1300,3,1,1600,'LO108');
INSERT INTO PROPERTY VALUES(11,'2','9531 Sherwood Rd.',1075,2,1,1100,'JO110');
INSERT INTO PROPERTY VALUES(12,'2','2 Bow Ridge Ave.',1400,3,2,1700,'RE107');


CREATE TABLE SERVICE_CATEGORY(CATEGORY_NUM DECIMAL(2,0) NOT NULL
PRIMARY KEY, CATEGORY_DESCRIPTION CHAR(35));


INSERT INTO SERVICE_CATEGORY VALUES('1', 'Plumbing');
INSERT INTO SERVICE_CATEGORY VALUES('2', 'Heating');
INSERT INTO SERVICE_CATEGORY VALUES('3', 'Painting');
INSERT INTO SERVICE_CATEGORY VALUES('4', 'Electrical Systems');
INSERT INTO SERVICE_CATEGORY VALUES('5', 'Carpentry');
INSERT INTO SERVICE_CATEGORY VALUES('6', 'Furniture Replacement');
```

```
CREATE TABLE SERVICE_REQUEST(SERVICE_ID DECIMAL(2,0) NOT NULL
PRIMARY KEY, PROPERTY_ID DECIMAL(2,0), CATEGORY_NUM DECIMAL(2,0),
OFFICE_NUM DECIMAL (2,0), DESCRIPTION CHAR(255), STATUS CHAR(255),
EST_HOURS DECIMAL(4,0), SPENT_HOURS DECIMAL(4,0), NEXT_SERVICE_DATE
DATE);

INSERT INTO SERVICE_REQUEST VALUES(1,11,2,2,'The second bedroom upstairs is not
heating up at night.', 'Problem has been confirmed. Central heating engineer has been
scheduled',2,1,'2019-11-01');

INSERT INTO SERVICE_REQUEST VALUES(2,1,4,1,'A new strip light is needed for the
kitchen', 'Scheduled',1,0,'2019-10-02');

INSERT INTO SERVICE_REQUEST VALUES(3,6,5,1,'The bathroom door does not close
properly.', 'Service rep has confirmed issue. Scheduled to be refitted',3,1,'2019-11-09');

INSERT INTO SERVICE_REQUEST VALUES(4,2,4,1,'New outlet has been requested for
upstairs bedroom.', 'Scheduled',1,0,'2019-10-02');

INSERT INTO SERVICE_REQUEST VALUES(5,8,3,2,'New paint job requested for common
area.', 'Open',10,0, null);

INSERT INTO SERVICE_REQUEST VALUES(6,4,1,1,'Shower is dripping when not in use.',

'Problem has been confirmed. Plumber scheduled',4,2,'2019-10-07');

INSERT INTO SERVICE_REQUEST VALUES(7,2,2,1,'Heating unit in entrance smells like its
burning', 'Service rep confirmed the issue to be dust in heating unit.To be cleaned',1,0,'2019-10-
09');

INSERT INTO SERVICE_REQUEST VALUES(8,9,1,2,'Kitchen sink does not drain properly.',
'Problem has been confirmed. Plumber scheduled',6,2,'2019-11-12');
```

```
INSERT INTO SERVICE_REQUEST VALUES(9,12,6,2,'New sofa requested.', 'Open',2,0,
null);

CREATE TABLE RESIDENTS(RESIDENT_ID DECIMAL(2,0) NOT NULL PRIMARY
KEY, FIRST_NAME CHAR(25),  SURNAME CHAR(25),PROPERTY_ID DECIMAL(2,0));

INSERT INTO RESIDENTS Values(1,'Albie',"O'Ryan",1);
INSERT INTO RESIDENTS Values(2,'Tariq',"Khan",1);
INSERT INTO RESIDENTS Values(3,'Ismail',"Salib",1);
INSERT INTO RESIDENTS Values(4,'Callen',"Beek",2);
INSERT INTO RESIDENTS Values(5,'Milosz',"Polansky",2);
INSERT INTO RESIDENTS Values(6,'Ashanti',"Lucas",2);
INSERT INTO RESIDENTS Values(7,'Randy',"Woodrue",2);
INSERT INTO RESIDENTS Values(8,'Aislinn',"Lawrence",3);
INSERT INTO RESIDENTS Values(9,'Monique',"French",3);
INSERT INTO RESIDENTS Values(10,'Amara',"Dejsuwan",4);
INSERT INTO RESIDENTS Values(11,'Rosalie',"Blackmore",4);
INSERT INTO RESIDENTS Values(12,'Carina',"Britton",4);
INSERT INTO RESIDENTS Values(13,'Valentino',"Ortega",5);
INSERT INTO RESIDENTS Values(14,'Kaylem',"Kent",5);
INSERT INTO RESIDENTS Values(15,'Alessia',"Wagner",6);
INSERT INTO RESIDENTS Values(16,'Tyrone',"Galvan",6);
INSERT INTO RESIDENTS Values(17,'Constance',"Fleming",6);
INSERT INTO RESIDENTS Values(18,'Eamonn',"Bain",6);
INSERT INTO RESIDENTS Values(19,'Misbah',"Yacob",7);
INSERT INTO RESIDENTS Values(20,'Gianluca',"Esposito",7);
INSERT INTO RESIDENTS Values(21,'Elinor',"Lake",7);
INSERT INTO RESIDENTS Values(22,'Ray',"Rosas",8);
INSERT INTO RESIDENTS Values(23,'Damon',"Caldwell",8);
```

INSERT INTO RESIDENTS Values(24,'Dawood',"Busby",8);

INSERT INTO RESIDENTS Values(25,'Dora',"Harries",8);

INSERT INTO RESIDENTS Values(26,'Leroy',"Stokes",8);

INSERT INTO RESIDENTS Values(27,'Tamia',"Hess",9);

INSERT INTO RESIDENTS Values(28,'Amelia',"Sanders",9);

INSERT INTO RESIDENTS Values(29,'Zarah',"Byers",10);

INSERT INTO RESIDENTS Values(30,'Sara',"Farrow",10);

INSERT INTO RESIDENTS Values(31,'Delilah',"Roy",10);

INSERT INTO RESIDENTS Values(32,'Dougie',"McDaniel",11);

INSERT INTO RESIDENTS Values(33,'Tahir',"Halabi",11);

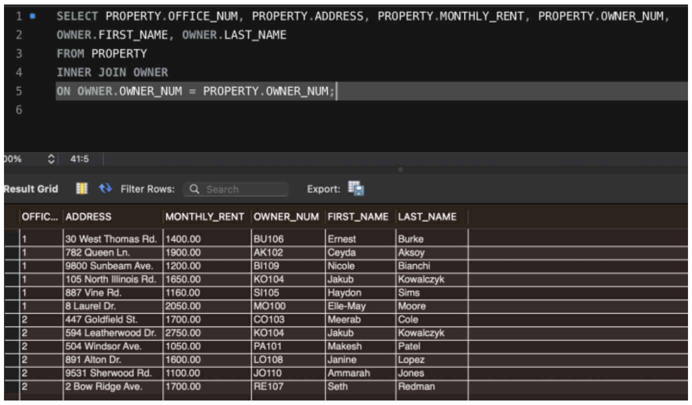
INSERT INTO RESIDENTS Values(34,'Mila',"Zhikin",12);

INSERT INTO RESIDENTS Values(35,'Glenn',"Donovan",12);

INSERT INTO RESIDENTS Values(36,'Zayn',"Fowler",12);

Chapter 5

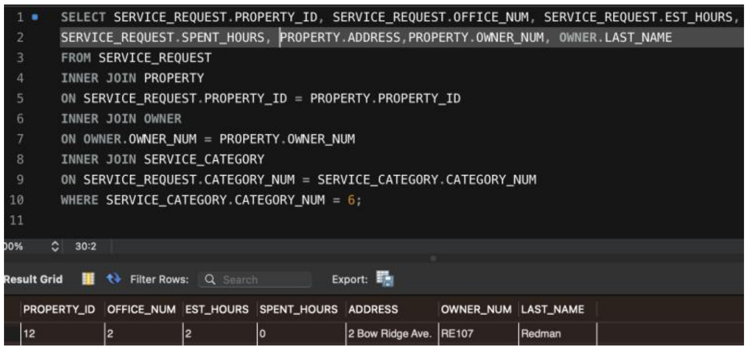
1. **For every property, list the management office number, address, monthly rent, owner number, owner's first name, and owner's last name.**



```sql
SELECT PROPERTY.OFFICE_NUM, PROPERTY.ADDRESS, PROPERTY.MONTHLY_RENT, PROPERTY.OWNER_NUM,
OWNER.FIRST_NAME, OWNER.LAST_NAME
FROM PROPERTY
INNER JOIN OWNER
ON OWNER.OWNER_NUM = PROPERTY.OWNER_NUM;
```

OFFIC...	ADDRESS	MONTHLY_RENT	OWNER_NUM	FIRST_NAME	LAST_NAME
1	30 West Thomas Rd.	1400.00	BU106	Ernest	Burke
1	782 Queen Ln.	1900.00	AK102	Ceyda	Aksoy
1	9800 Sunbeam Ave.	1200.00	BI109	Nicole	Bianchi
1	105 North Illinois Rd.	1650.00	KO104	Jakub	Kowalczyk
1	887 Vine Rd.	1160.00	SI105	Haydon	Sims
1	8 Laurel Dr.	2050.00	MO100	Elle-May	Moore
2	447 Goldfield St.	1700.00	CO103	Meerab	Cole
2	594 Leatherwood Dr.	2750.00	KO104	Jakub	Kowalczyk
2	504 Windsor Ave.	1050.00	PA101	Makesh	Patel
2	891 Alton Dr.	1600.00	LO108	Janine	Lopez
2	9531 Sherwood Rd.	1100.00	JO110	Ammarah	Jones
2	2 Bow Ridge Ave.	1700.00	RE107	Seth	Redman

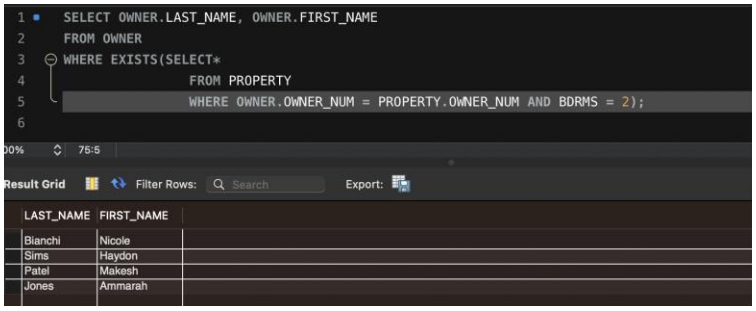
2. **For every service request for furniture replacement, list the property ID, management office number, address, estimated hours, spent hours, owner number, and owner's last name.**



```sql
SELECT SERVICE_REQUEST.PROPERTY_ID, SERVICE_REQUEST.OFFICE_NUM, SERVICE_REQUEST.EST_HOURS,
SERVICE_REQUEST.SPENT_HOURS, PROPERTY.ADDRESS,PROPERTY.OWNER_NUM, OWNER.LAST_NAME
FROM SERVICE_REQUEST
INNER JOIN PROPERTY
ON SERVICE_REQUEST.PROPERTY_ID = PROPERTY.PROPERTY_ID
INNER JOIN OWNER
ON OWNER.OWNER_NUM = PROPERTY.OWNER_NUM
INNER JOIN SERVICE_CATEGORY
ON SERVICE_REQUEST.CATEGORY_NUM = SERVICE_CATEGORY.CATEGORY_NUM
WHERE SERVICE_CATEGORY.CATEGORY_NUM = 6;
```

```
PROPERTY_ID  OFFICE_NUM  EST_HOURS  SPENT_HOURS  ADDRESS             OWNER_NUM  LAST_NAME
12           2           2          0            2 Bow Ridge Ave.    RE107      Redman
```

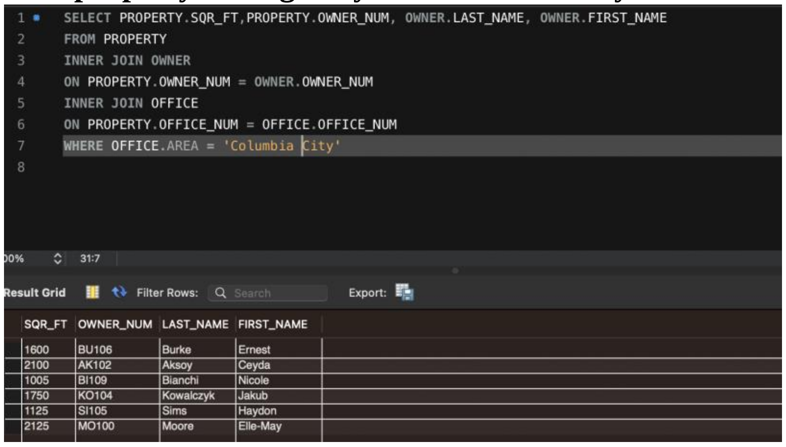
3. **Repeat Exercise 4, but this time use the EXISTS operator in your query.**



```sql
SELECT OWNER.LAST_NAME, OWNER.FIRST_NAME
FROM OWNER
WHERE EXISTS(SELECT*
                FROM PROPERTY
                WHERE OWNER.OWNER_NUM = PROPERTY.OWNER_NUM AND BDRMS = 2);
```

```
LAST_NAME  FIRST_NAME
Bianchi    Nicole
Sims       Haydon
Patel      Makesh
Jones      Ammarah
```

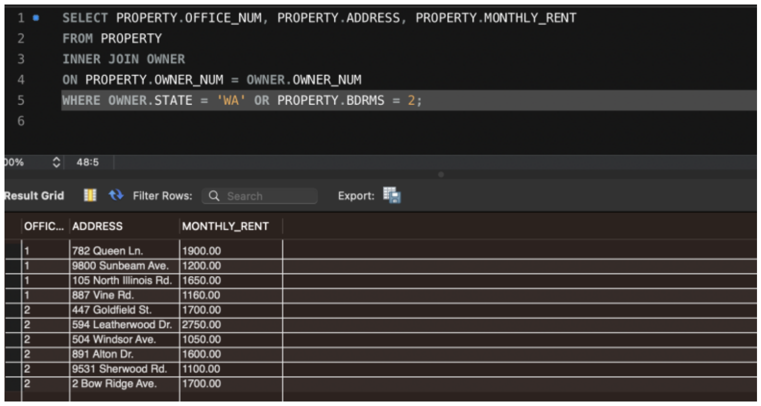
4. **List the square footage, owner number, owner last name, and owner first name for each property managed by the Columbia City office.**



```sql
SELECT PROPERTY.SQR_FT,PROPERTY.OWNER_NUM, OWNER.LAST_NAME, OWNER.FIRST_NAME
FROM PROPERTY
INNER JOIN OWNER
ON PROPERTY.OWNER_NUM = OWNER.OWNER_NUM
INNER JOIN OFFICE
ON PROPERTY.OFFICE_NUM = OFFICE.OFFICE_NUM
WHERE OFFICE.AREA = 'Columbia City'
```

```
SQR_FT  OWNER_NUM  LAST_NAME  FIRST_NAME
1600    BU106      Burke      Ernest
2100    AK102      Aksoy      Ceyda
1005    BI109      Bianchi    Nicole
1750    KO104      Kowalczyk  Jakub
1125    SI105      Sims       Haydon
2125    MO100      Moore      Elle-May
```

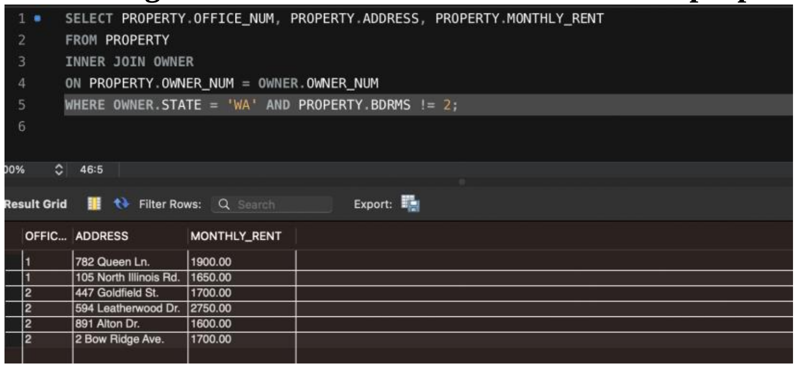
5. **List the office number, address, and monthly rent for properties whose owners live in Washington state or own two-bedroom properties.**



```sql
SELECT PROPERTY.OFFICE_NUM, PROPERTY.ADDRESS, PROPERTY.MONTHLY_RENT
FROM PROPERTY
INNER JOIN OWNER
ON PROPERTY.OWNER_NUM = OWNER.OWNER_NUM
WHERE OWNER.STATE = 'WA' OR PROPERTY.BDRMS = 2;
```

OFFIC...	ADDRESS	MONTHLY_RENT
1	782 Queen Ln.	1900.00
1	9800 Sunbeam Ave.	1200.00
1	105 North Illinois Rd.	1650.00
1	887 Vine Rd.	1160.00
2	447 Goldfield St.	1700.00
2	594 Leatherwood Dr.	2750.00
2	504 Windsor Ave.	1050.00
2	891 Alton Dr.	1600.00
2	9531 Sherwood Rd.	1100.00
2	2 Bow Ridge Ave.	1700.00

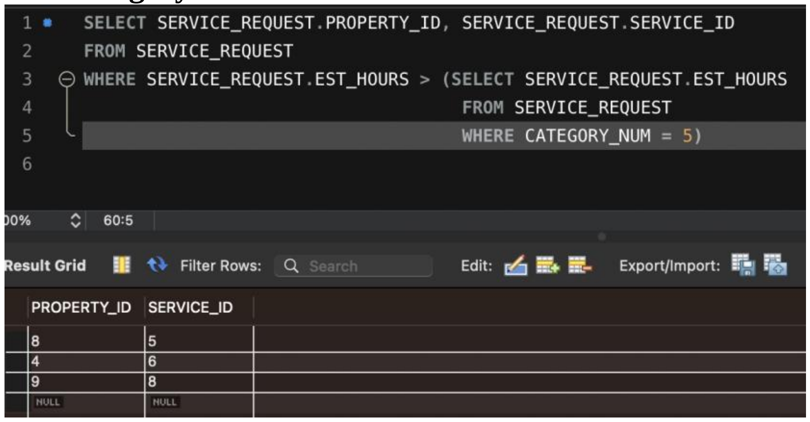
6. **List the office number, address, and monthly rent for properties whose owners live in Washington state but do not own two-bedroom properties.**



```sql
SELECT PROPERTY.OFFICE_NUM, PROPERTY.ADDRESS, PROPERTY.MONTHLY_RENT
FROM PROPERTY
INNER JOIN OWNER
ON PROPERTY.OWNER_NUM = OWNER.OWNER_NUM
WHERE OWNER.STATE = 'WA' AND PROPERTY.BDRMS != 2;
```

OFFIC...	ADDRESS	MONTHLY_RENT
1	782 Queen Ln.	1900.00
1	105 North Illinois Rd.	1650.00
2	447 Goldfield St.	1700.00
2	594 Leatherwood Dr.	2750.00
2	891 Alton Dr.	1600.00
2	2 Bow Ridge Ave.	1700.00

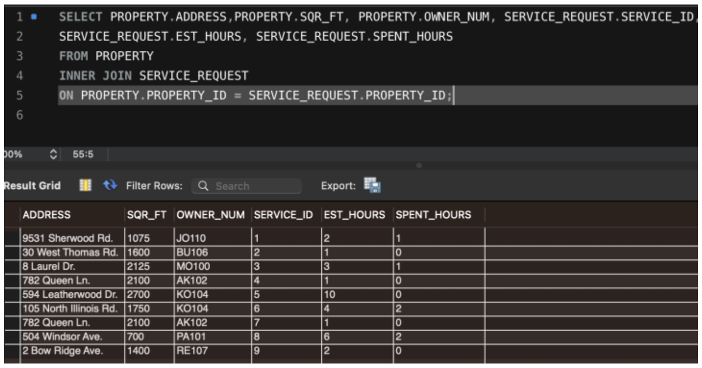
7. **Find the service ID and property ID for each service request whose estimated hours are greater than the number of estimated hours on every service request on which the cat- egory number is 5.**



```sql
SELECT SERVICE_REQUEST.PROPERTY_ID, SERVICE_REQUEST.SERVICE_ID
FROM SERVICE_REQUEST
WHERE SERVICE_REQUEST.EST_HOURS > (SELECT SERVICE_REQUEST.EST_HOURS
                                   FROM SERVICE_REQUEST
                                   WHERE CATEGORY_NUM = 5)
```

PROPERTY_ID	SERVICE_ID
8	5
4	6
9	8
NULL	NULL

8. **Repeat Exercise 14, but this time be sure each property is included regardless of whether the property currently has any service requests for category 4.**

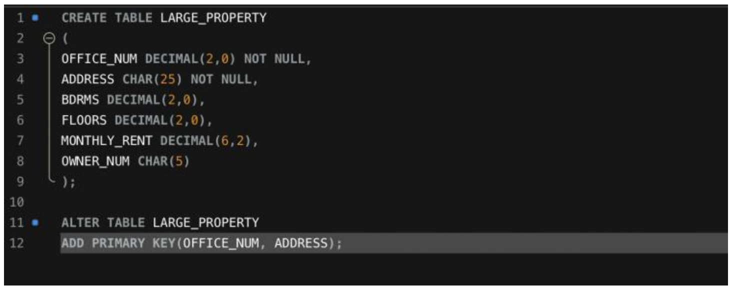


```sql
SELECT PROPERTY.ADDRESS,PROPERTY.SQR_FT, PROPERTY.OWNER_NUM, SERVICE_REQUEST.SERVICE_ID,
SERVICE_REQUEST.EST_HOURS, SERVICE_REQUEST.SPENT_HOURS
FROM PROPERTY
INNER JOIN SERVICE_REQUEST
ON PROPERTY.PROPERTY_ID = SERVICE_REQUEST.PROPERTY_ID;
```

ADDRESS	SQR_FT	OWNER_NUM	SERVICE_ID	EST_HOURS	SPENT_HOURS
9531 Sherwood Rd.	1075	JO110	1	2	1
30 West Thomas Rd.	1600	BU106	2	1	0
8 Laurel Dr.	2125	MO100	3	3	1
782 Queen Ln.	2100	AK102	4	1	0
594 Leatherwood Dr.	2700	KO104	5	10	0
105 North Illinois Rd.	1750	KO104	6	4	2
782 Queen Ln.	2100	AK102	7	1	0
504 Windsor Ave.	700	PA101	8	6	2
2 Bow Ridge Ave.	1400	RE107	9	2	0

Chapter 6

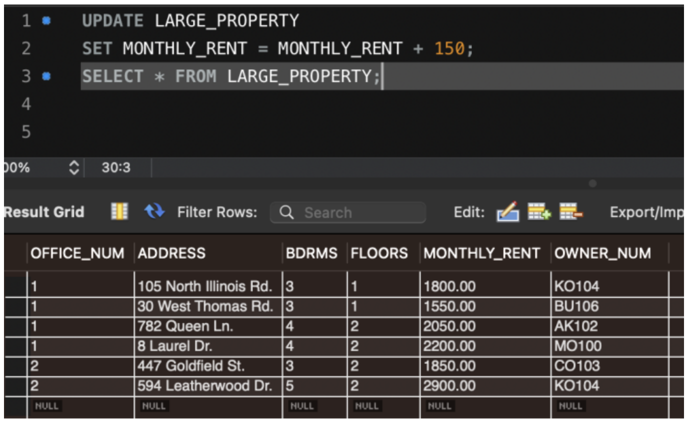
1. **Create a LARGE_PROPERTY table with the structure shown in Figure 6-29. (Hint: If you have trouble creating the primary key, see Figure 3-36 in Module 3.)**



```sql
CREATE TABLE LARGE_PROPERTY
(
OFFICE_NUM DECIMAL(2,0) NOT NULL,
ADDRESS CHAR(25) NOT NULL,
BDRMS DECIMAL(2,0),
FLOORS DECIMAL(2,0),
MONTHLY_RENT DECIMAL(6,2),
OWNER_NUM CHAR(5)
);

ALTER TABLE LARGE_PROPERTY
ADD PRIMARY KEY(OFFICE_NUM, ADDRESS);
```

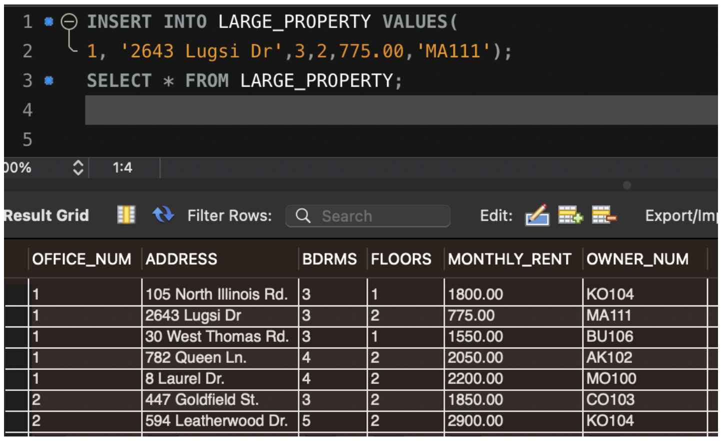
2. **StayWell has increased the monthly rent of each large property by $150. Update the monthly rents in the LARGE_PROPERTY table accordingly.**



```sql
UPDATE LARGE_PROPERTY
SET MONTHLY_RENT = MONTHLY_RENT + 150;
SELECT * FROM LARGE_PROPERTY;
```

OFFICE_NUM	ADDRESS	BDRMS	FLOORS	MONTHLY_RENT	OWNER_NUM
1	105 North Illinois Rd.	3	1	1800.00	KO104
1	30 West Thomas Rd.	3	1	1550.00	BU106
1	782 Queen Ln.	4	2	2050.00	AK102
1	8 Laurel Dr.	4	2	2200.00	MO100
2	447 Goldfield St.	3	2	1850.00	CO103
2	594 Leatherwood Dr.	5	2	2900.00	KO104
NULL	NULL	NULL	NULL	NULL	NULL

3. **Insert a row into the LARGE_PROPERTY table for a new property. The office number is 1, the address is 2643 Lugsi Dr, the number of bedrooms is 3, the number of floors is 2, the monthly rent is $775, and the owner number is MA111.**



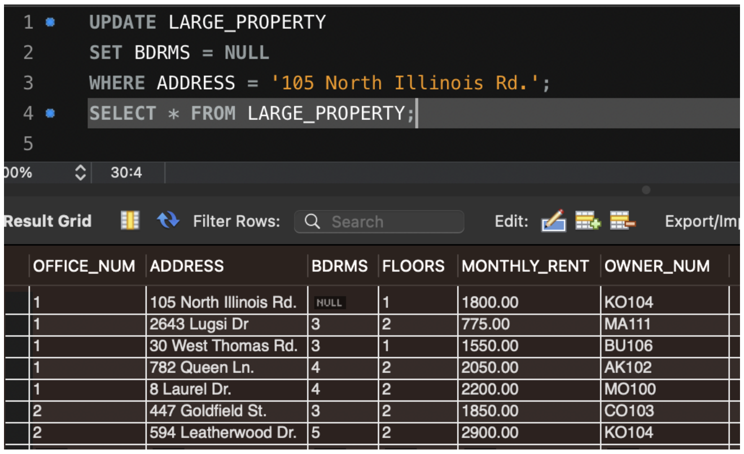
```
INSERT INTO LARGE_PROPERTY VALUES(
1, '2643 Lugsi Dr',3,2,775.00,'MA111');
SELECT * FROM LARGE_PROPERTY;
```

```
00%    1:4
```

Result Grid   Filter Rows: Search   Edit: Export/Im

OFFICE_NUM	ADDRESS	BDRMS	FLOORS	MONTHLY_RENT	OWNER_NUM
1	105 North Illinois Rd.	3	1	1800.00	KO104
1	2643 Lugsi Dr	3	2	775.00	MA111
1	30 West Thomas Rd.	3	1	1550.00	BU106
1	782 Queen Ln.	4	2	2050.00	AK102
1	8 Laurel Dr.	4	2	2200.00	MO100
2	447 Goldfield St.	3	2	1850.00	CO103
2	594 Leatherwood Dr.	5	2	2900.00	KO104

4. **The property in managed by Columbia City with the address 105 North Illinois Rd is in the process of being remodeled and the number of bedrooms is unknown. Change the bedroom's value in the LARGE_PROPERTY table to null.**



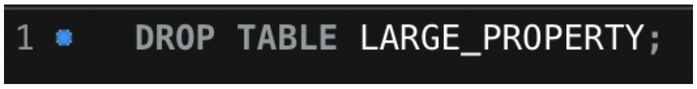
```
UPDATE LARGE_PROPERTY
SET BDRMS = NULL
WHERE ADDRESS = '105 North Illinois Rd.';
SELECT * FROM LARGE_PROPERTY;
```

```
00%    30:4
```

Result Grid   Filter Rows: Search   Edit: Export/Im

OFFICE_NUM	ADDRESS	BDRMS	FLOORS	MONTHLY_RENT	OWNER_NUM
1	105 North Illinois Rd.	NULL	1	1800.00	KO104
1	2643 Lugsi Dr	3	2	775.00	MA111
1	30 West Thomas Rd.	3	1	1550.00	BU106
1	782 Queen Ln.	4	2	2050.00	AK102
1	8 Laurel Dr.	4	2	2200.00	MO100
2	447 Goldfield St.	3	2	1850.00	CO103
2	594 Leatherwood Dr.	5	2	2900.00	KO104

5. **Delete the LARGE_PROPERTY table from the database.**

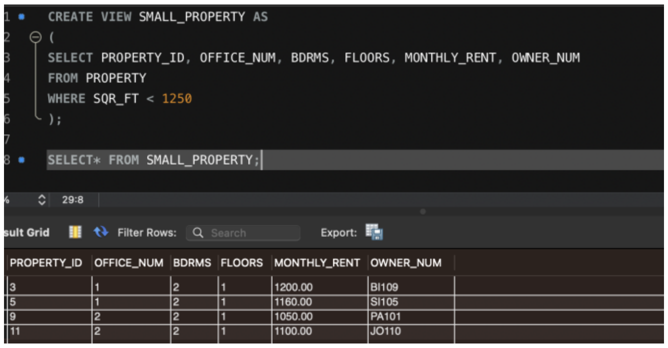


```
DROP TABLE LARGE_PROPERTY;
```

Chapter 7

1. **Create a view named SMALL_PROPERTY. It consists of the property ID, office number, bedrooms, floor, monthly rent, and owner number for every property whose square footage is less than 1,250 square feet.**

a. **Write and execute the CREATE VIEW command to create the SMALL_PROPERTY view.**

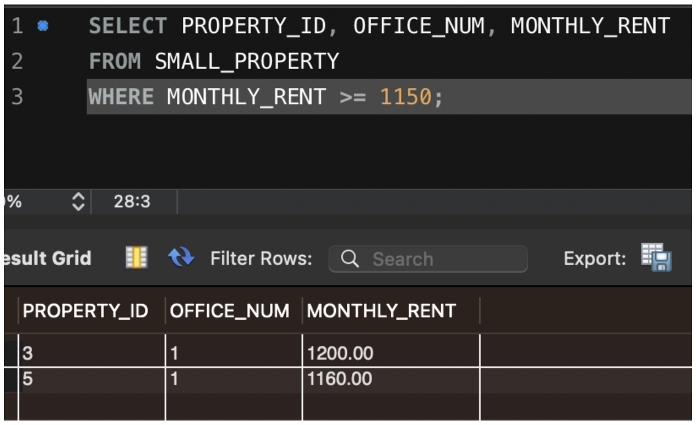


```sql
CREATE VIEW SMALL_PROPERTY AS
(
SELECT PROPERTY_ID, OFFICE_NUM, BDRMS, FLOORS, MONTHLY_RENT, OWNER_NUM
FROM PROPERTY
WHERE SQR_FT < 1250
);

SELECT* FROM SMALL_PROPERTY;
```

PROPERTY_ID	OFFICE_NUM	BDRMS	FLOORS	MONTHLY_RENT	OWNER_NUM
3	1	2	1	1200.00	BI109
5	1	2	1	1160.00	SI105
9	2	2	1	1050.00	PA101
11	2	2	1	1100.00	JO110

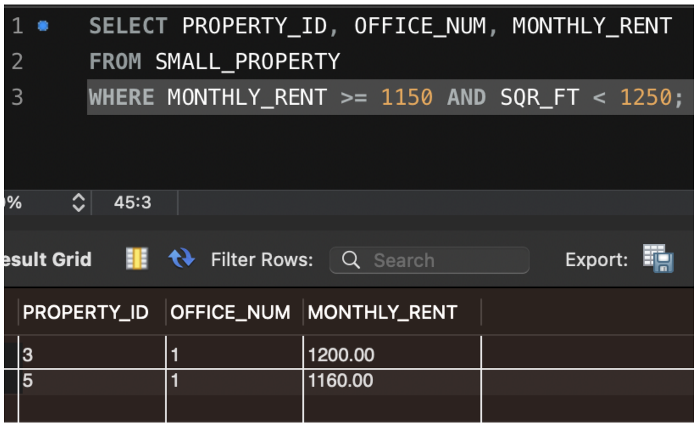
b. **Write and execute the command to retrieve the office number, property ID, and monthly rent for every property in the SMALL_PROPERTY view with a monthly rent of $1150 or more.**



```sql
SELECT PROPERTY_ID, OFFICE_NUM, MONTHLY_RENT
FROM SMALL_PROPERTY
WHERE MONTHLY_RENT >= 1150;
```

PROPERTY_ID	OFFICE_NUM	MONTHLY_RENT
3	1	1200.00
5	1	1160.00

c. **Write and execute the query that the DBMS actually executes.**

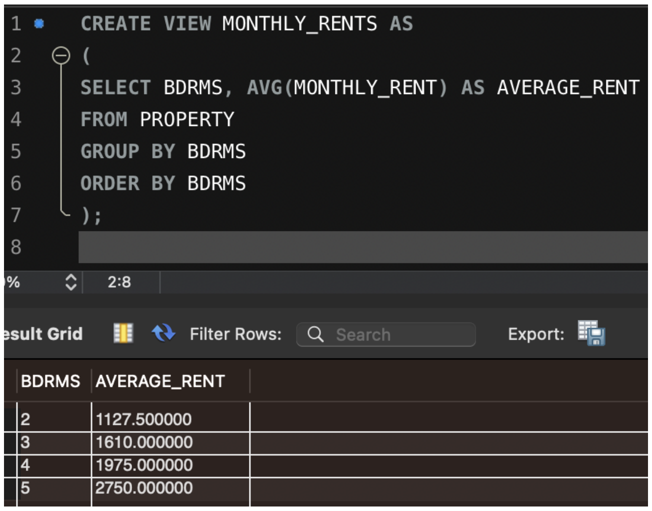


```
SELECT PROPERTY_ID, OFFICE_NUM, MONTHLY_RENT
FROM SMALL_PROPERTY
WHERE MONTHLY_RENT >= 1150 AND SQR_FT < 1250;
```

PROPERTY_ID	OFFICE_NUM	MONTHLY_RENT
3	1	1200.00
5	1	1160.00

d. **Does updating the database through this view create any problems? If so, what are they? If not, why not?**
   i. No because views technically do not exist, therefore they cannot alter data in tables.

2. **Create a view named MONTHLY_RENTS. It consists of two columns: The first is the number of bedrooms, and the second is the average monthly rent for all properties in the PROPERTY table that have that number of bedrooms. Use AVERAGE_RENT as the column name for the average monthly rent. Group and order the rows by number of bedrooms.**
   a. **Write and execute the CREATE VIEW command to create the MONTHLY_RENTS view.**

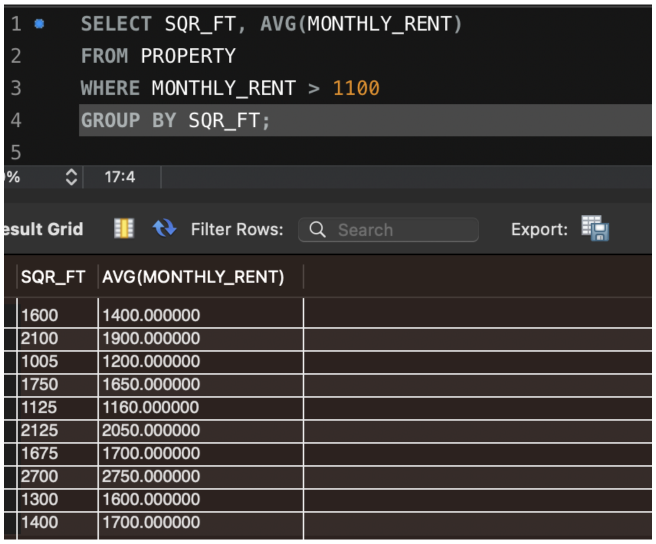


```
CREATE VIEW MONTHLY_RENTS AS
(
SELECT BDRMS, AVG(MONTHLY_RENT) AS AVERAGE_RENT
FROM PROPERTY
GROUP BY BDRMS
ORDER BY BDRMS
);
```

```
2:8
```

BDRMS	AVERAGE_RENT
2	1127.500000
3	1610.000000
4	1975.000000
5	2750.000000

b. **Write and execute the command to retrieve the square footage and average fee for each square footage for which the average fee is greater than $1,100.**



```
SELECT SQR_FT, AVG(MONTHLY_RENT)
FROM PROPERTY
WHERE MONTHLY_RENT > 1100
GROUP BY SQR_FT;
```

```
17:4
```

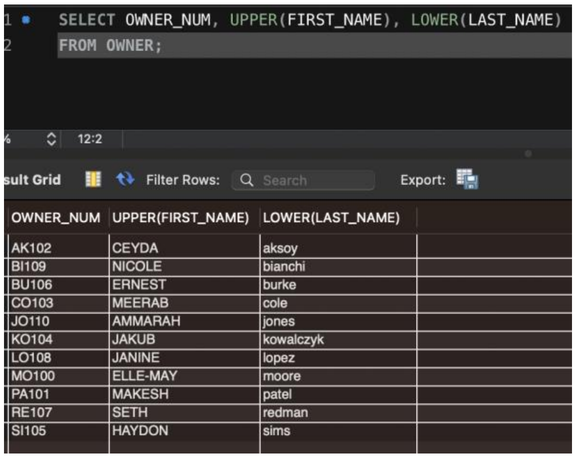
SQR_FT	AVG(MONTHLY_RENT)
1600	1400.000000
2100	1900.000000
1005	1200.000000
1750	1650.000000
1125	1160.000000
2125	2050.000000
1675	1700.000000
2700	2750.000000
1300	1600.000000
1400	1700.000000

c. **Does updating the database through this view create any problems? If so, what are they? If not, why not?**

   i. No because views technically do not exist, therefore they cannot alter data in tables.

Chapter 8

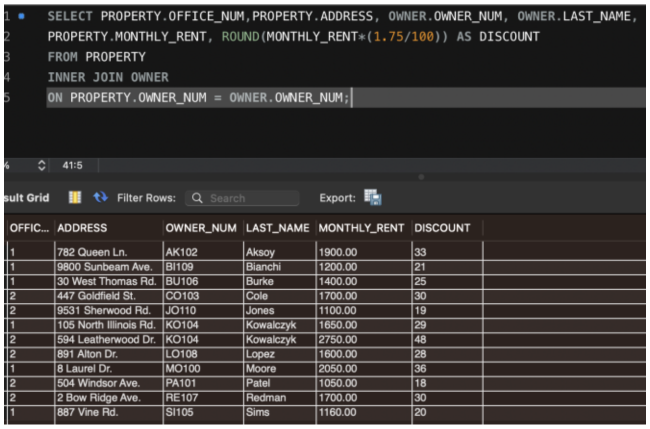
1. **List the owner number, first name, and last name for all owners. The first name should appear in uppercase letters and the last name should appear in lowercase letters.**



```sql
SELECT OWNER_NUM, UPPER(FIRST_NAME), LOWER(LAST_NAME)
FROM OWNER;
```

OWNER_NUM	UPPER(FIRST_NAME)	LOWER(LAST_NAME)
AK102	CEYDA	aksoy
BI109	NICOLE	bianchi
BU106	ERNEST	burke
CO103	MEERAB	cole
JO110	AMMARAH	jones
KO104	JAKUB	kowalczyk
LO108	JANINE	lopez
MO100	ELLE-MAY	moore
PA101	MAKESH	patel
RE107	SETH	redman
SI105	HAYDON	sims

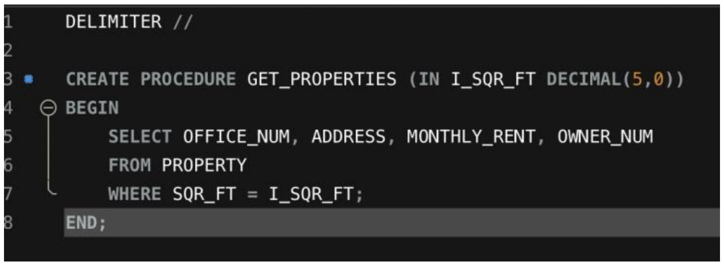
2. **StayWell is offering a monthly discount for residents who pay their rent on a quarterly basis. The discount is 1.75 percent of the monthly fee. For each property, list the office number, address, owner number, owner's last name, monthly rent, and discount. The discount should be rounded to the nearest dollar.**



```sql
SELECT PROPERTY.OFFICE_NUM,PROPERTY.ADDRESS, OWNER.OWNER_NUM, OWNER.LAST_NAME,
PROPERTY.MONTHLY_RENT, ROUND(MONTHLY_RENT*(1.75/100)) AS DISCOUNT
FROM PROPERTY
INNER JOIN OWNER
ON PROPERTY.OWNER_NUM = OWNER.OWNER_NUM;
```

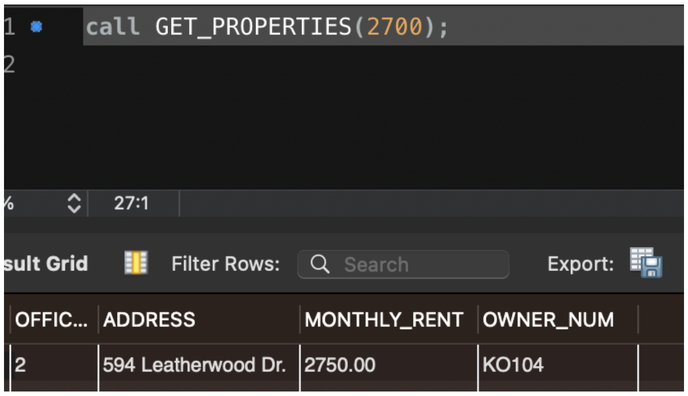
OFFIC...	ADDRESS	OWNER_NUM	LAST_NAME	MONTHLY_RENT	DISCOUNT
1	782 Queen Ln.	AK102	Aksoy	1900.00	33
1	9800 Sunbeam Ave.	BI109	Bianchi	1200.00	21
1	30 West Thomas Rd.	BU106	Burke	1400.00	25
2	447 Goldfield St.	CO103	Cole	1700.00	30
2	9531 Sherwood Rd.	JO110	Jones	1100.00	19
1	105 North Illinois Rd.	KO104	Kowalczyk	1650.00	29
2	594 Leatherwood Dr.	KO104	Kowalczyk	2750.00	48
2	891 Alton Dr.	LO108	Lopez	1600.00	28
1	8 Laurel Dr.	MO100	Moore	2050.00	36
2	504 Windsor Ave.	PA101	Patel	1050.00	18
2	2 Bow Ridge Ave.	RE107	Redman	1700.00	30
1	887 Vine Rd.	SI105	Sims	1160.00	20

3. **Write PL/SQL or T-SQL procedures to retrieve and output the office number, address, monthly rent, and owner number for every property whose square footage is equal to the square footage stored in I_SQR_FT.**



```sql
DELIMITER //

CREATE PROCEDURE GET_PROPERTIES (IN I_SQR_FT DECIMAL(5,0))
BEGIN
    SELECT OFFICE_NUM, ADDRESS, MONTHLY_RENT, OWNER_NUM
    FROM PROPERTY
    WHERE SQR_FT = I_SQR_FT;
END;
```



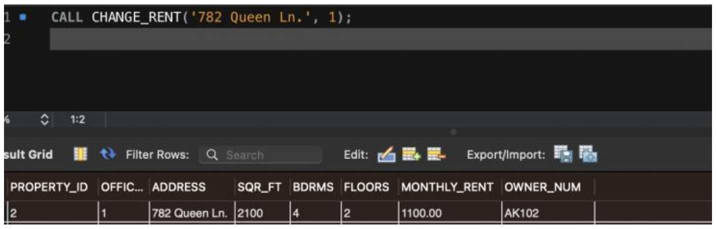
```
call GET_PROPERTIES(2700);
```

% 27:1

sult Grid Filter Rows: Search Export:

OFFIC...	ADDRESS	MONTHLY_RENT	OWNER_NUM
2	594 Leatherwood Dr.	2750.00	KO104

4. **Write a stored procedure in PL/SQL or T-SQL that changes the monthly rent of a property with a given address and office number. How would you use this stored procedure to change the monthly rent for the property with the address "782 Queen Ln." and office number 1 to $1,100?**

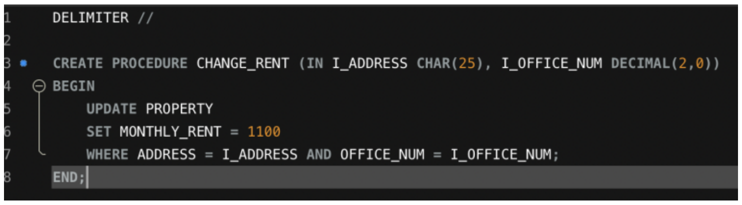


```
CALL CHANGE_RENT('782 Queen Ln.', 1);
```

% 1:2

sult Grid Filter Rows: Search Edit: Export/Import:

PROPERTY_ID	OFFIC...	ADDRESS	SQR_FT	BDRMS	FLOORS	MONTHLY_RENT	OWNER_NUM
2	1	782 Queen Ln.	2100	4	2	1100.00	AK102



```
DELIMITER //

CREATE PROCEDURE CHANGE_RENT (IN I_ADDRESS CHAR(25), I_OFFICE_NUM DECIMAL(2,0))
BEGIN
    UPDATE PROPERTY
    SET MONTHLY_RENT = 1100
    WHERE ADDRESS = I_ADDRESS AND OFFICE_NUM = I_OFFICE_NUM;
END;
```

<u>REFERENCES</u>

“MySQL Tutorial for Beginners [Full Course].” *YouTube*, YouTube, 19 Mar. 2019, www.youtube.com/watch?v=7S_tz1z_5bA.

Shellman, M., Afyouni, H. A., Pratt, P. J., & Last, M. Z. (2021). *A guide to SQL*. Cengage.

Bagui, S., & Earp, R. (2003). *Database design using entity-relationship diagrams*. Boca Raton: Auerbach.