**Micro-Project Report**

Database Management (22416)

Topic:-Pharmacy Management System.

1. Rationale:-

Implementing a pharmacy management system within a database system makes running a pharmacy smoother. It helps keep track of medicines, patient details, and transactions all in one place. This means pharmacists can find information quickly, manage inventory better, and ensure patient safety. It also helps with tasks like filling prescriptions and keeping records up-to-date. Plus, it keeps everything secure and makes it easier to follow rules and regulations. Overall, it's like having a reliable assistant that helps the pharmacy run efficiently and safely.

1. Aims and Benefits:-
2. **Aim :**
3. It helps in tracking the stock of medicines.
4. It allows for the storage and easy retrieval of patient information, prescriptions, and medication history, improving patient care.
5. It simplifies the billing process, creating accurate invoices and tracking payments.
6. To provide insights into inventory turnover, sales trends, and prescription statistics.
7. **Benefits :**
8. It helps in structuring and organizing data related to medications, prescriptions, and inventory, making it easy to retrieve and manage information.
9. Using a database ensures data accuracy and consistency, reducing errors and improving the reliability of the system.
10. Databases minimize data redundancy, saving storage space and ensuring consistent information across the system.
11. Course Outcomes:-

1. Create Database using SQL commands.

2. Manage Database using SQL commands.

3. Implement Advanced SQL concepts on Database.

4. Actual Methodology Followed:-

* 1. Identification of entities of database and their attribute along with datatypes.
  2. Developed schema diagram based on entities and attributes.
  3. Schema diagram is converted into a relational database by using DDL commands in oracle10 g platform.
  4. Performed other DDL commands like alter, rename, truncate and drop on a single table.
  5. Performed DML commands like select, insert, update, delete on a single table.
  6. Performed aggregate functions and JOIN on the table.

1. Actual Resources Used:-

|  |  |  |  |
| --- | --- | --- | --- |
| Sr.no. | Name of Resources | Specification | Qty. |
|  | Computer System | i-5,RAM 4GB | 1 |
|  | Software | Oracle19 | 1 |

1. Output of Micro project:-
   1. **Entities and Attributes:**

Entity in database management system is a real world object which has certain properties known as attributes. An attribute is a piece of data that describes an entity. The following are the Entities:**–**

**Entities:-**

1. Meds
2. Bill
3. Sales

Attributes:-

1. meds :

{med\_id, med\_name, batch\_no, med\_man, med\_exp, total\_med, med\_price, exp\_status}

1. bill :

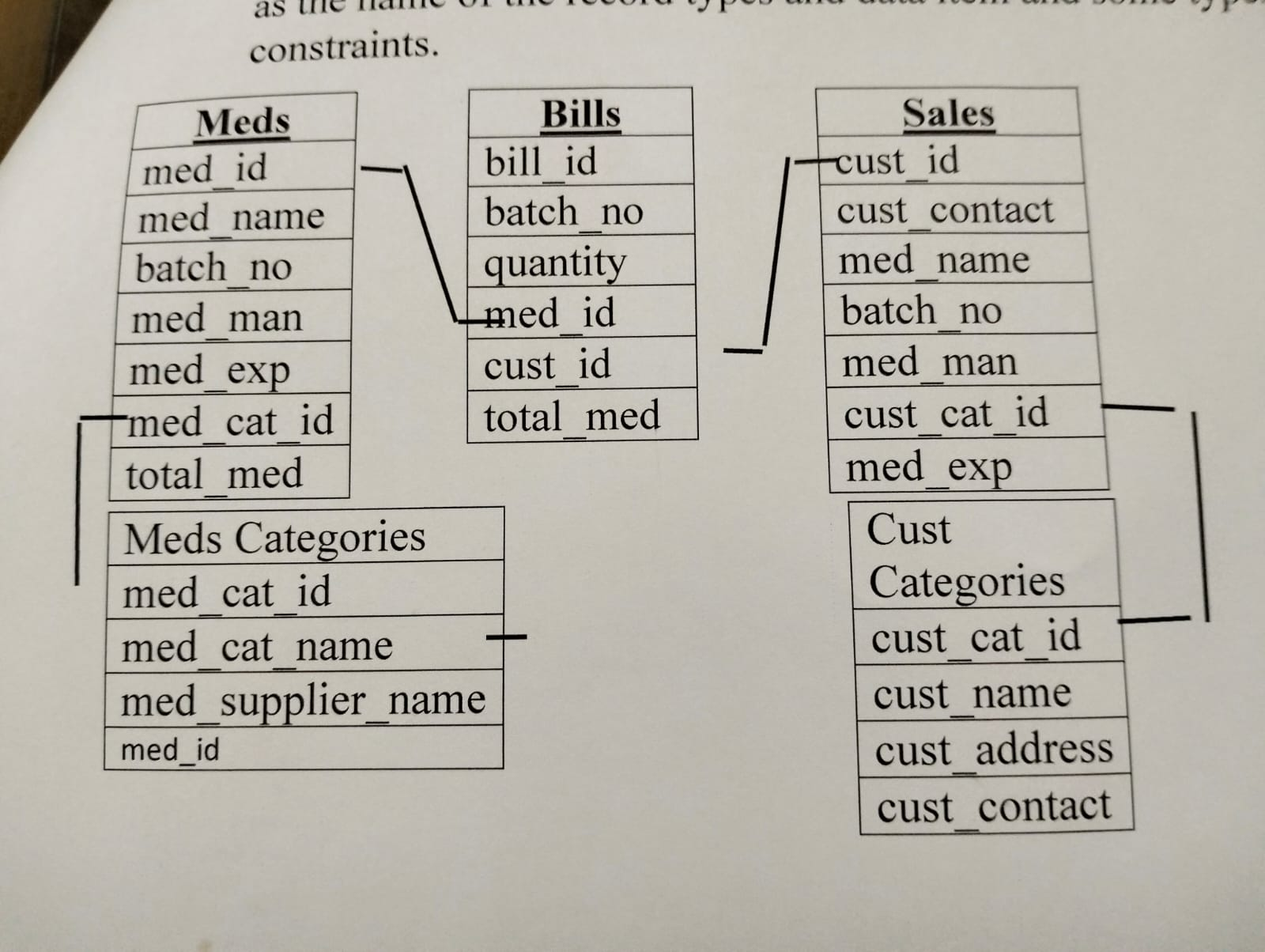
{bill\_id, batch\_no, quantity, med\_id, cust\_id, total}

1. sales :

{cust\_id, cust\_contact, cust\_name, batch\_no, med\_man, med\_exp, med\_price, sysdate, quantity, med\_id}

# Schema diagram (with description):

Description of database is called as schema. A schema diagram which contains entities and the attributes that defines the schema in structural format.



**DDL commands:**

DDL changes the structure of the table by creating a table, by removing a table and by modifying table etc. All the commands of DDL are auto committed that means if permanently save all changes in the database.

SQL commands are in DDL are:

1. Create
   1. **Create**: It is used to create new table in the database. Syntax: Create table < table\_name>

(attribute1 datatype(size));

For this query above table is used:-

1. medicines table:

Create table medicines\_m

(med\_id number(10) Primary key,

med\_name varchar(20),

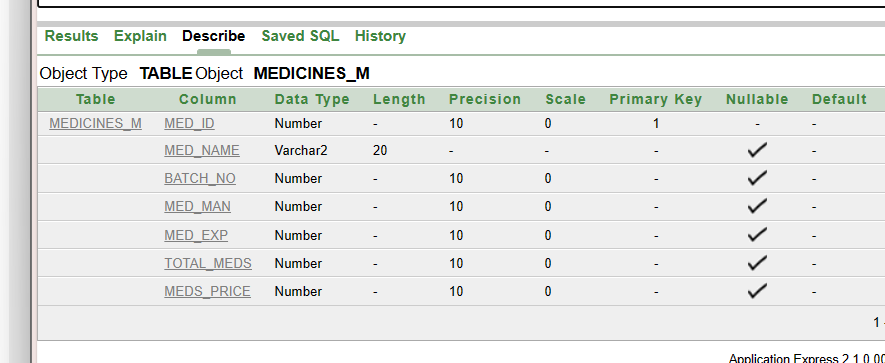
batch\_no numeric(10),

med\_man numeric(10),

med\_exp numeric(10),

total\_meds numeric(10),

meds\_price numeric(10));



2.sales table

Create table sale\_t

(cust\_id varchar2(10) Primary key,

cust\_contact numeric(10),

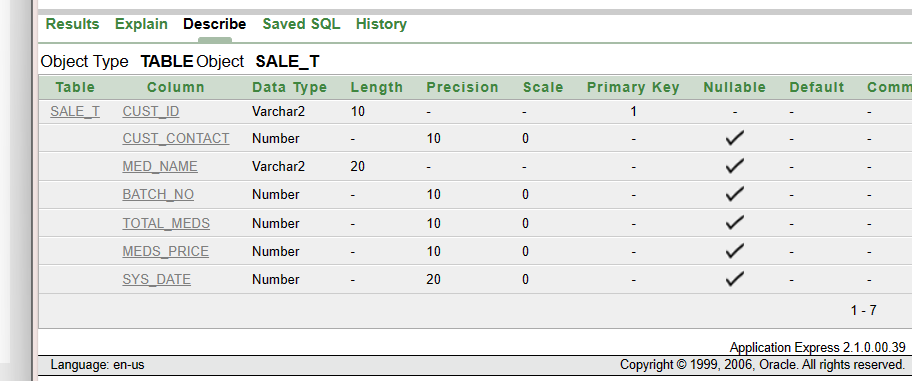
med\_name varchar2(20),

batch\_no numeric(10),

total\_meds numeric(10),

meds\_price numeric(10),

sys\_date numeric(20) );



3.bill table:

Create table Bill\_

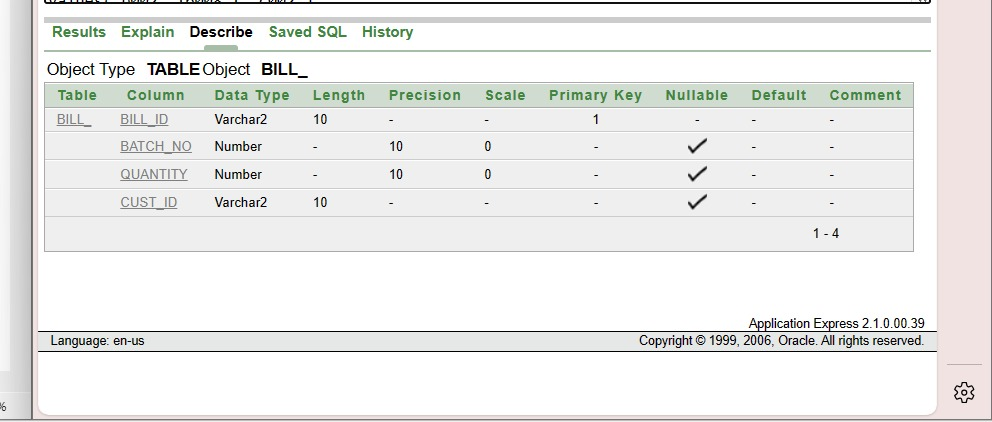
(bill\_id varchar2(10)Primary key,

batch\_no numeric(10),

quantity numeric(10),

cust\_id varchar2(10)constraint fk1100 references sale\_t(cust\_id)on delete cascade);

DESC Bill\_;



4. meds category table:

Create table meds\_cat

(cat\_id varchar2(20),

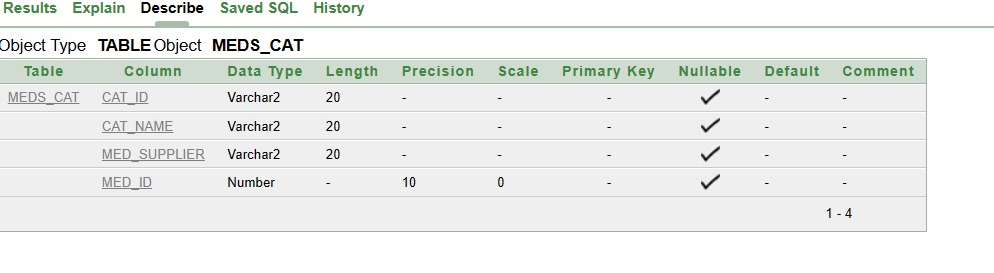
cat\_name varchar2(20),

med\_supplier varchar2(20),

med\_id number(10)constraint fk110 references medicines\_m(med\_id)on

delete cascade);

DESC meds\_cat;



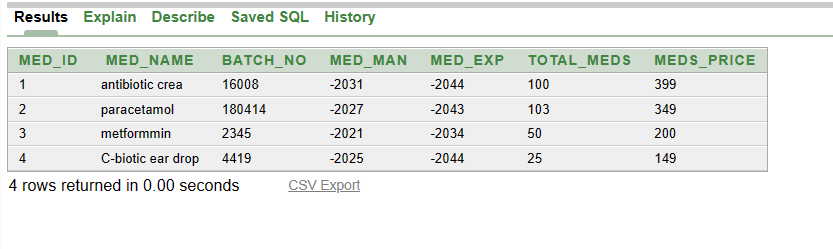
**DML commands:**

This commands is use to perform operation on the rows. Using this command we can do changes in the rows.

1. Select
2. Insert
3. Update

**Select**: It is used for selection of data from the table.

Syntax: Select \* from table\_name;



**Insert**: To insert data into the table.

Syntax: Insert into table\_name

Values(‘attribute1’, ‘attribute2’, attribute3);

1. medicine table:

insert into medicines\_m

values(1, 'antibiotic crea', 16008, 01-10-2022,01-20-2025,100,399);

insert into medicines\_m

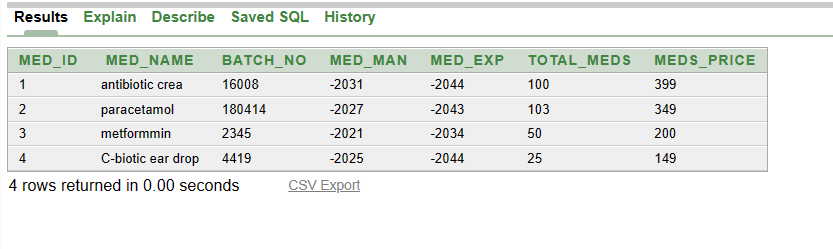
values (2, 'paracetamol', 180414,01-05-2023,01-20-2024,103,349);

insert into medicines\_m

values (3, 'metformmin', 2345, 01-01-2021,01-12-2023,50, 200);

insert into medicines\_m

values (4, 'C-biotic ear drop', 4419, 01-03-2023,01-20-2025,25,149);



2. sales table:

insert into sale\_t

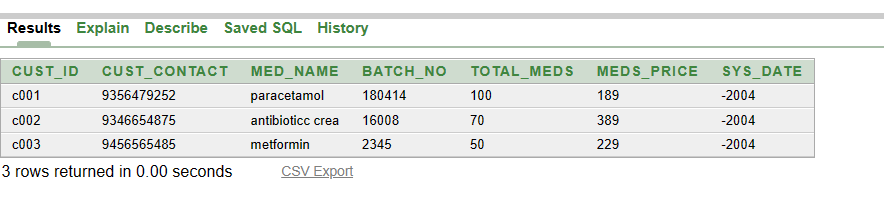
values('c001',9356479252,'paracetamol',180414,100,189,23-04-2023);

insert into sale\_t

values('c002',9346654875,'antibioticc crea',16008,70,389,23-04-2023);

insert into sale\_t

values('c003',9456565485,'metformin',2345,50,229,23-04-2023);



3. bill table:

insert into Bill\_

values('b001',2345,2,'c003');

insert into Bill\_

values('b002',16008,1,'c002');

insert into bill\_

values('b003',180414,2,c001);



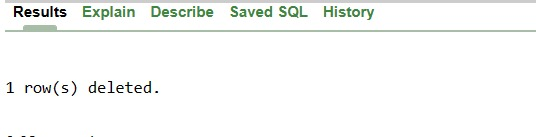
1. **Update**: It is used to update data in the table Syntax: Update table\_name

Set <columnnmae>=<expression> Where<columnname>=<expression>;



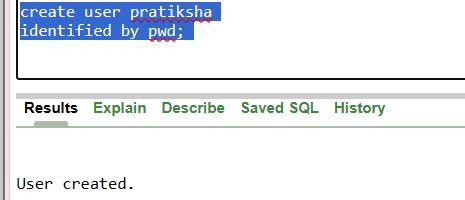
1. **Delete**: It is used to remove data from table. Syntax: Delete from table\_name

Where =condition;



* + **DCL command:-**
    1. First we Can create role by following syntax:- Create user <user\_name>

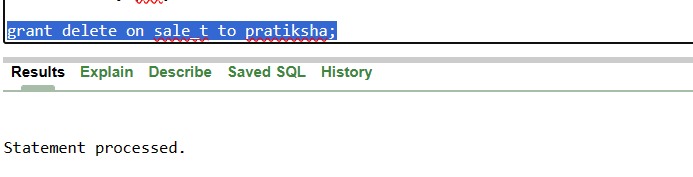
Identified by <password>;



* + 1. **Grant:-**SQL grant command is used to provide access or privileges on the database objects users.

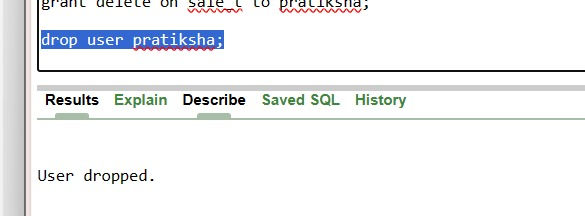
Syntax:-grant <privileges\_name> On object\_name

To [user/public/role] [with grant option];



* + 1. **To remove role:-**

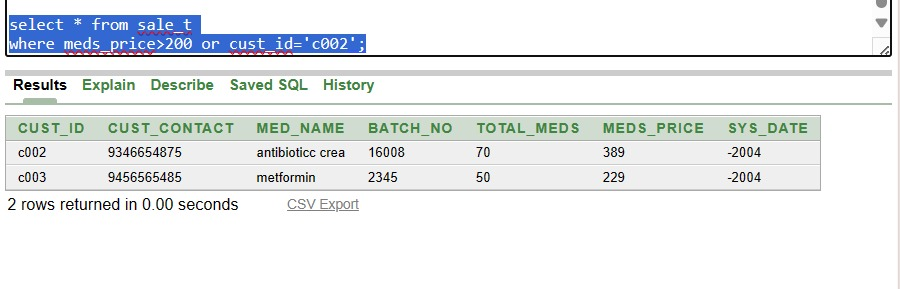
Syntax:- drop role <role\_name>;



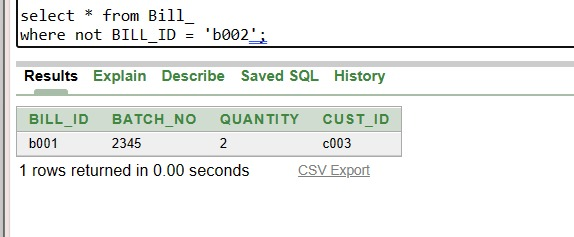
* SQL Command:

**Logical Operator**

* + - 1. **OR**: True if any of the condition separated by OR is true.

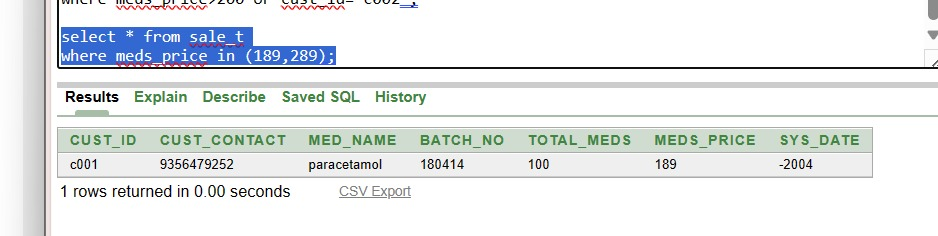


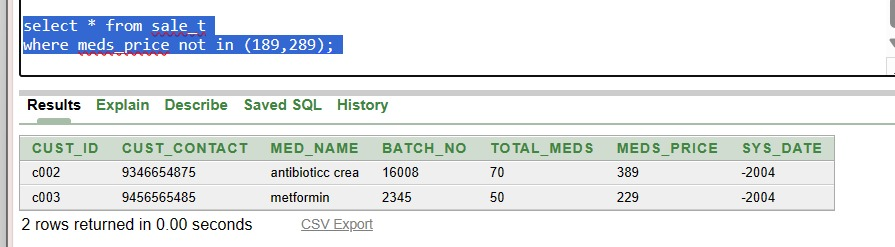
* + - 1. **NOT**: Display a record condition not true.



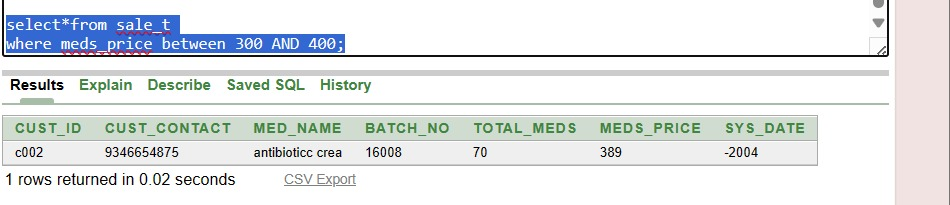
* **Word Comparison Operator:**

**IN & NOT IN**: True if operand is equal to one of the list argument.

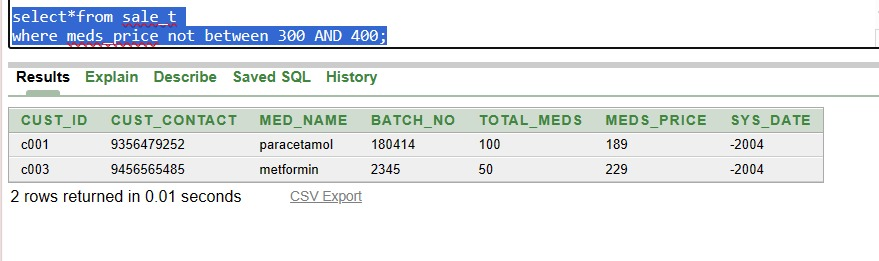




* + - 1. **BETWEEN**: True if operand within the range of comparison.



* + - 1. **Not between:** True if operand within the not range of comparison.

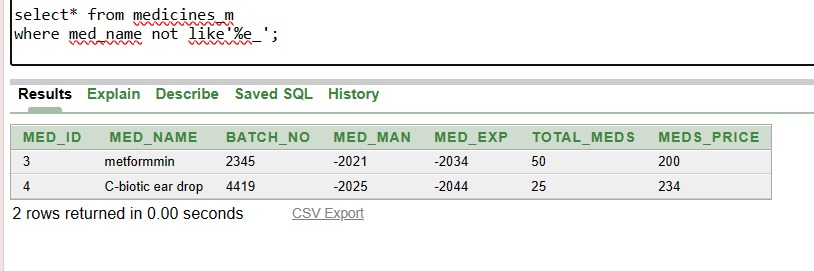


* + - 1. **LIKE:** True if the operand matches a pattern ‘%’ denotes n number of character.

‘\_’ denotes single character

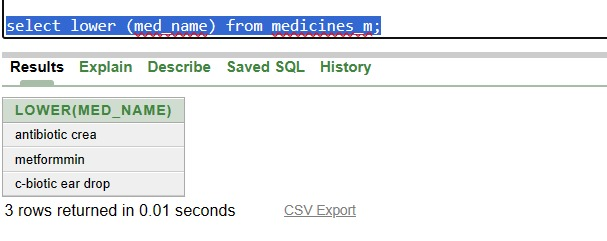


* + - 1. **Not like:-** True if the operand does not matches a pattern.

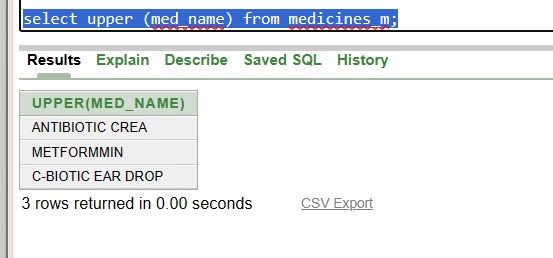


* **String function:**

**Lower():** This function is used to convert the upper case string into lower case.



**Upper():** This function is used to convert the lower case string into upper case.

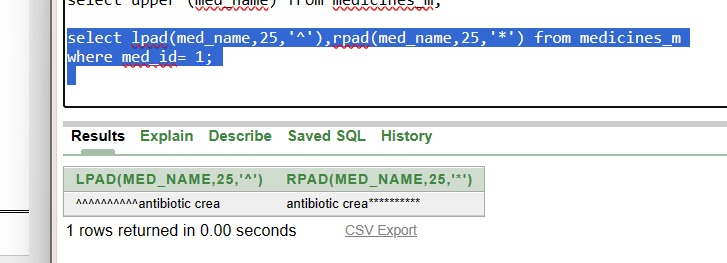


**Initcap():** Return a input string with initial letter capitalize and all other character in lower case.

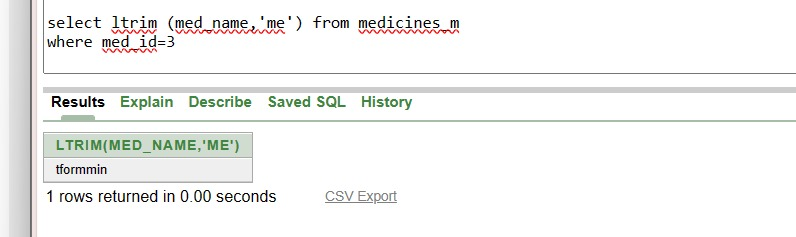


**LPAD() :** It returns character to left padded to given sequence with sequence with sequence of character in char 2.

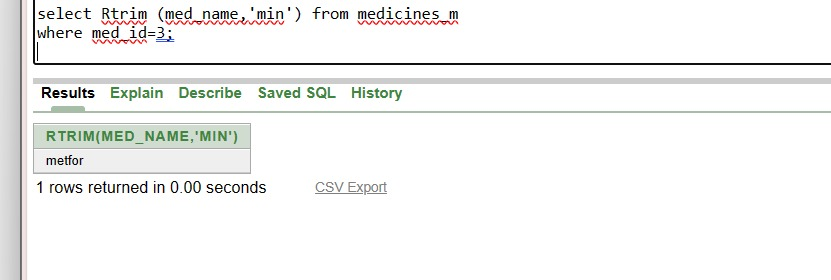
**RPAD()** : It return character to right padded to given sequence with sequence with sequence of character in char 2.



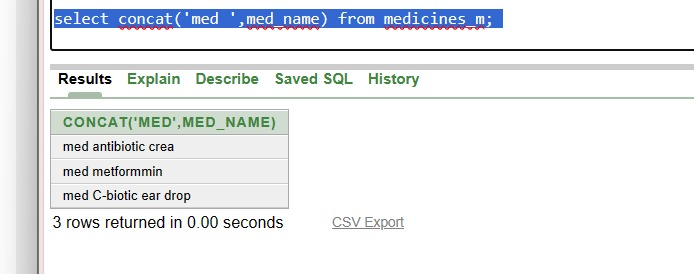
**LTRIM()(char)(set):**It removes on trim from left of character string.



* + - 1. **RTRIM()(char)(set):**It removes on trim from right of character string.

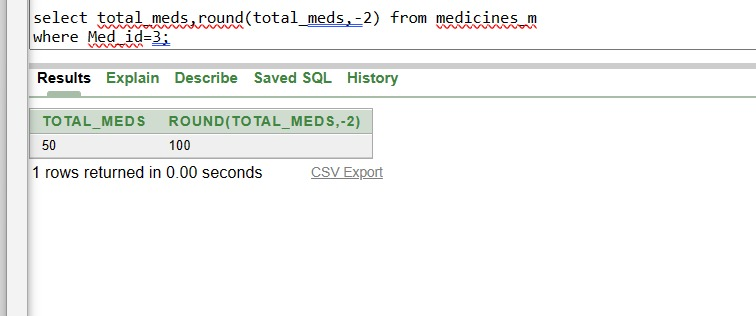


* + - 1. **CONCAT(char1,char2):**char1 is concatenated to char2.It mergs over two or more string values.it is similar to concatenation operator **( || ).**



* **Numeric function**:

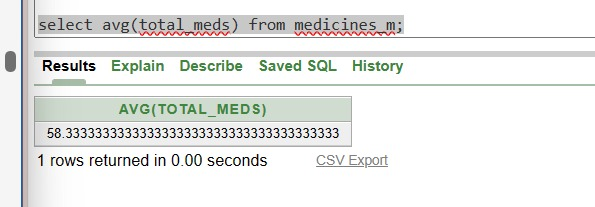
**ROUND()=**it returns numeic expression rounded to an integer can be used to round on expression to a number of decimal points.



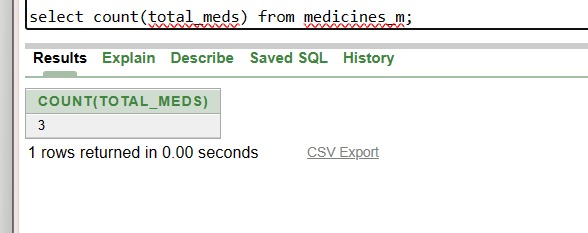
* **Aggregate function:**

Each query in SQL results of groups of values and also field values. This functions can produce a value an entire group or table. They operate on set of rows and returns result based on group of rows.

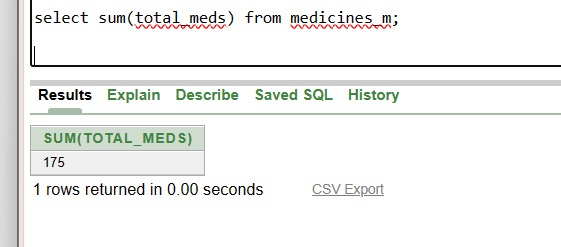
* + - 1. **Avg():** This functions calculates average value of a column of numeric type.



* + - 1. **Count():** The SQL count function returns of rows in a table satisfying criteria the ‘where’ clause.

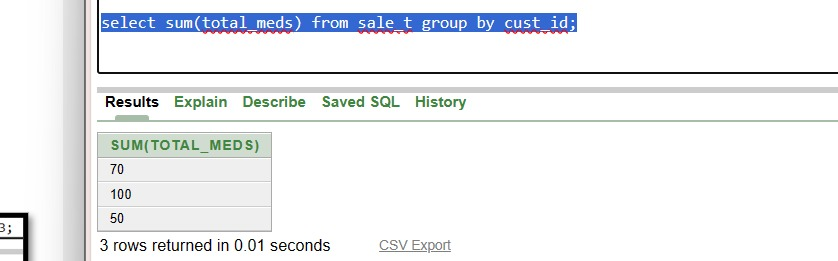


* + - 1. **Sum():** This function is used to return sum of all numeric values in the specified column.

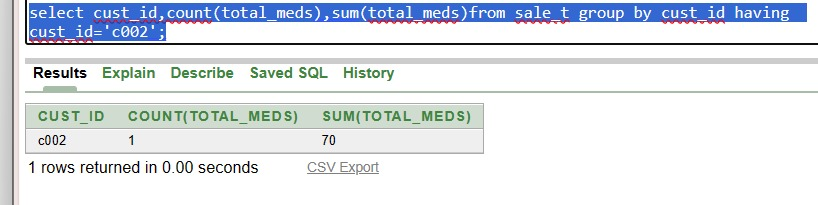


* **Clauses:**

**Group by clause**:- The usage of SQL group by clause is to divide the rows into table into smaller groups. The group by clause is used with SQL select statement. The grouping can happen after retrieval of rows based on certain but can not be applied to group result. In group by clause, aggregate functions can be perform with one command.



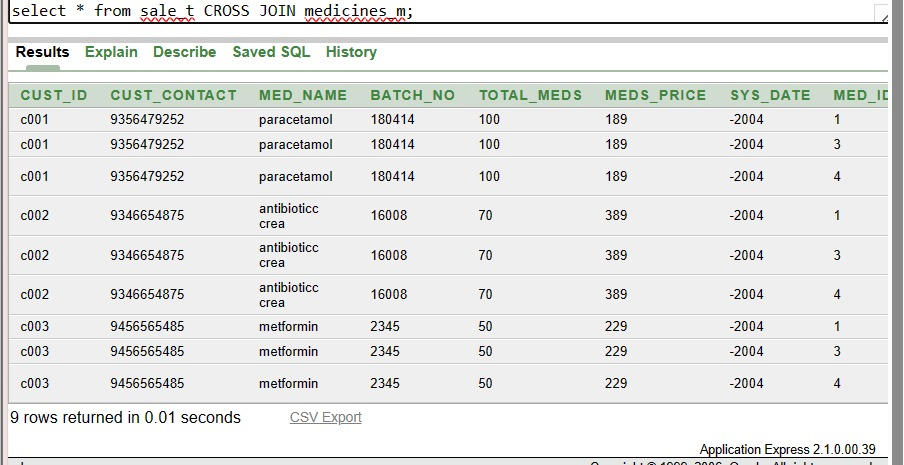
**Having clause**: This can be used in conjunction with the group by clause. Having imposes the condition on the group by clause which further filters the groups created by group by clause.



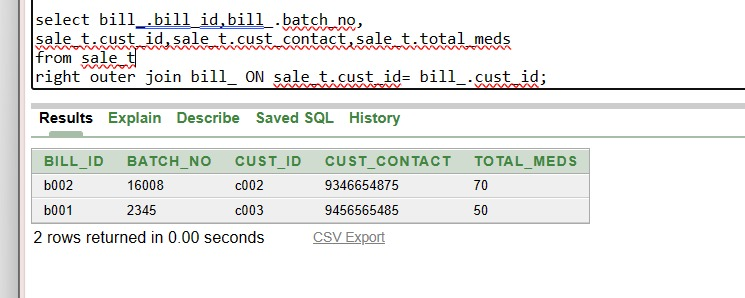
* **Joins:**

Join work on two or more table if they have at least one common field and have relationship between them. Join keep the base stable unchanged. Tables are on columns that have same data type and size in the table.

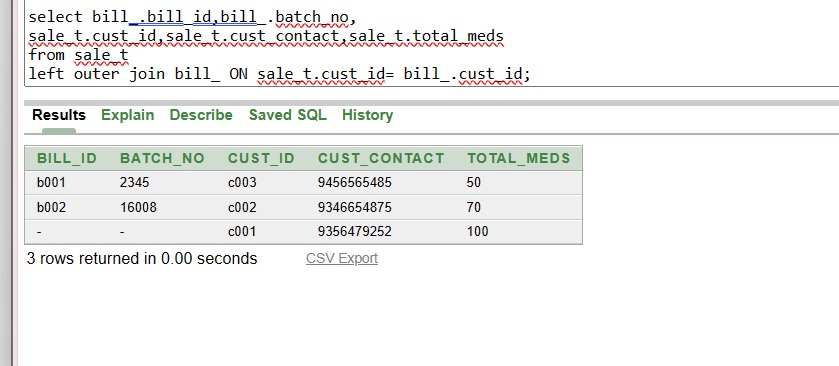
1. **Cross join:** When each row of the first table is combined each of the second table. It is know as cartesion join or cross join. SQL cross join returns the cartesion product of the set of row from the join table.



1. **Right outer Join:** Right join return all the values from the rows of right table and matching value of the left table.

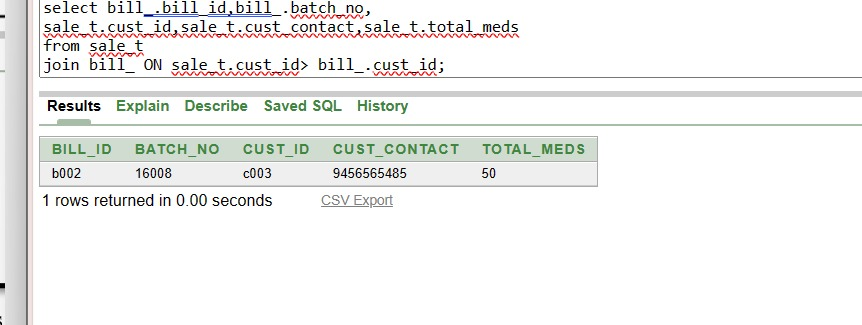


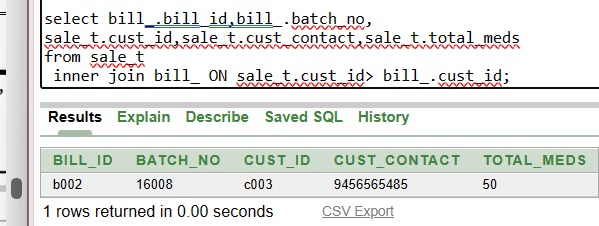
1. **Left outer join:** Left join this join returns all the unmatching rows of the table on the left side of the join and matching rows from the both table. The left join also know as Left Outer Join.



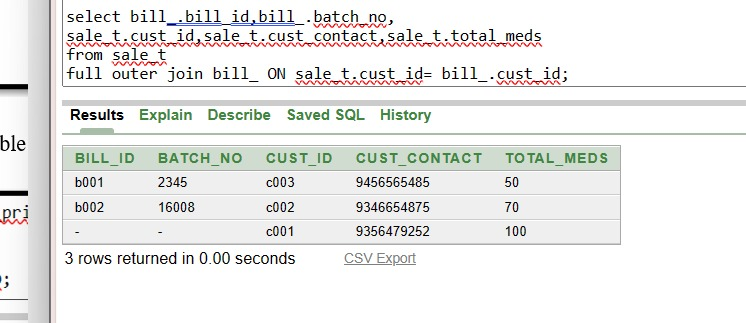
1. **Equijoin**: It is type of join that combined table based on matching values in specified columns. It is possible to perform on more than two tables.



1. **Non-Equijoin**: Non-Equijoin perform the join using comparison operator other than equal to. 
2. **Inner join**: In SQL, inner join select record that are matching values in the both tables as long as condition saticsfies. Return the combination of all rows from both the table where the condition is saticsfies



1. **Full join:** Full join is the result of combination of both left and right outer join. Join table have all record from both tables. It put NULL on the placed of matches not found.



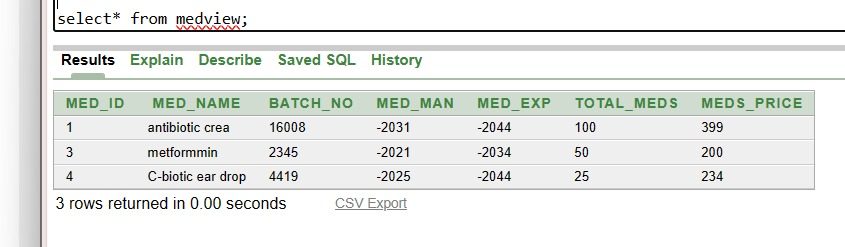
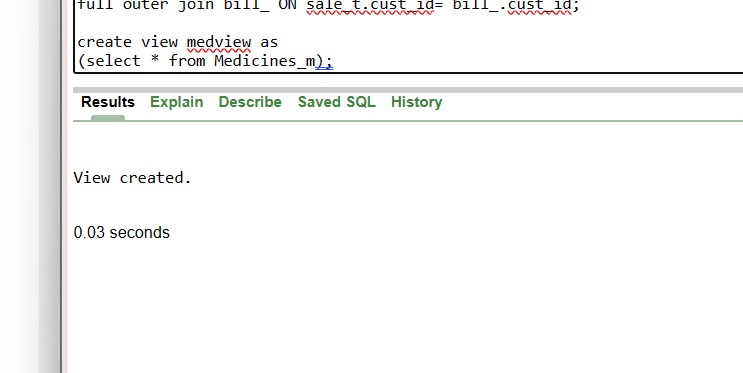
* + **View**:

A view is a logical table based on another table. It represent subset of the data from one or more tables. A view contains no data of it’s own it is like window through which data from table can be viewed. On which view is based called as based tables. The view is store as a select statement in a data dictionary. View doesn’t contain data. Defination of the view stored is user views data dictonary table.

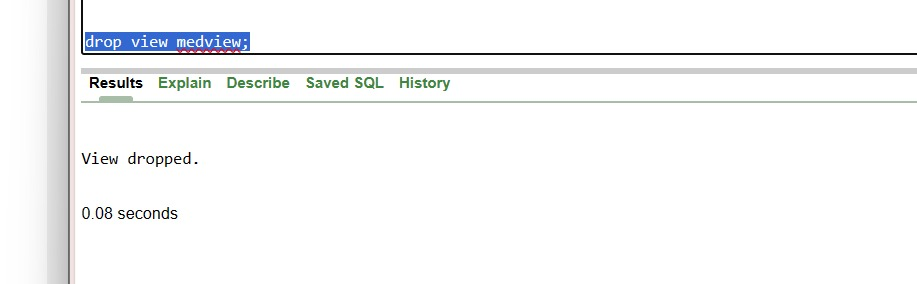
1. **To create view**:

Syntax: create view view\_name as(Select colum1, colum2 From table\_name

Where condition);



1. **To remove view**:

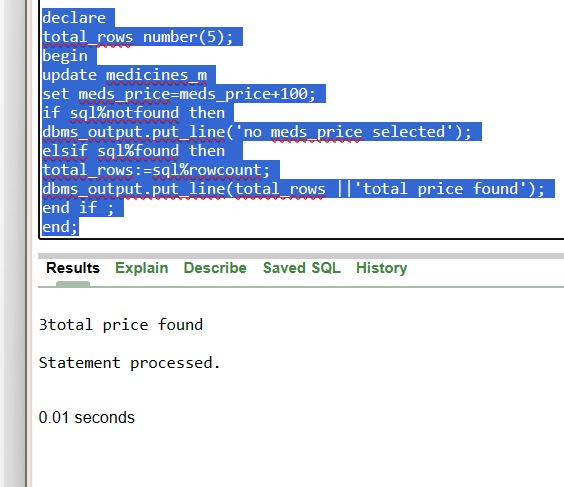
Syntax: Drop view view\_name; 

* + **Cursor:**

PL/SQL cursor oracle creates a memory area known as context area for processing an SQL statement which contain information nested for processing the statement. A cursor contain the context area through a cursor. The cursor holds the rows (1 or more)but can process one row at a time which are returned by SQL statement. The set of rows that the cursor hold is referred as active data set.

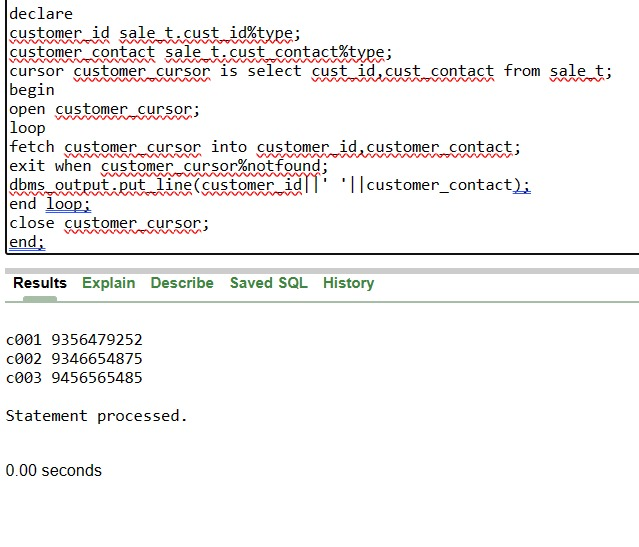
Type of Cursor:

1. Implicit Cursor
2. Explicit Cursor
   * + 1. **Implicit Cursor:** It is automatically created by oracle whenever an SQL statement is executed. The most recent implicit cursor are SQL%Found, SQL%NotFound, SQL%Rowcount, SQL%Isopen. Statement affects one or more rows.



* + - 1. **Explicit Cursor**: Explicit cursor is the programmer defined cursor for gaining more control over the context are. An explicit cursor should be defined in the declaration section of PL/SQL working with explicit cursor involves 4 steps.

1]declare 2]open 3]fetch 4]close



1. Skills developed:-
   1. Design the database schema and create tables for medicines, sales, bills .
   2. Define the relationships between these tables and specify data types for each field.
   3. Enforce the data integrity through the database schema.
2. Application of Micro project:-
3. Store and manage customer details, including their name, contact information, medical history, and prescriptions.
4. Track medicines stock levels, expiration dates, and reorder points.
5. Record sales transactions, generate invoices, and calculate the total cost of medications for patients.