Name: Vishal Shashikant Salvi.

UID: 2019230069

Class: SE Comps

Batch: C

Experiment No 6

Aim: Working on View Command

Theory:

Create View Statement

Views can be considered as virtual tables. Generally speaking, a table has a set of definition, and it physically stores the data. A view also has a set of definitions, which is build on top of table(s) or other view(s), and it does not physically store the data.

The syntax for creating a view is as follows:

```
CREATE VIEW "VIEW_NAME" AS "SQL Statement";
```

"SQL Statement" can be any of the SQL statements we have discussed in this tutorial.

Let's use a simple example to illustrate. Say we have the following table:

Table **Customer**

Column Name	Data Type
First_Name	char(50)
Last_Name	char(50)
Address	char(50)
City	char(50)
Country	char(25)
Birth_Date	datetime

and we want to create a view called **V_Customer** that contains only the First_Name, Last_Name, and Country columns from this table, we would type in,

```
CREATE VIEW V_Customer
AS SELECT First_Name, Last_Name, Country
FROM Customer;
```

Now we have a view called **V_Customer** with the following structure:

View **V_Customer**

Column Name	Data Type
First_Name	char(50)
Last_Name	char(50)
Country	char(25)

We can also use a view to apply joins to two tables. In this case, users only see one view rather than two tables, and the SQL statement users need to issue becomes much simpler. Let's say we have the following two tables:

Table **Store_Information**

Store_Name	Sales	Txn_Date
Los Angeles	1500	Jan-05-1999
San Diego	250	Jan-07-1999
Los Angeles	300	Jan-08-1999
Boston	700	Jan-08-1999

Table **Geography**

Region_Name	Store_Name
East	Boston
East	New York
West	Los Angeles
West	San Diego

and we want to build a view that has sales by region information. We would issue the following SQL statement:

```
CREATE VIEW V_REGION_SALES
AS SELECT A1.Region_Name REGION, SUM(A2.Sales) SALES
FROM Geography A1, Store_Information A2
WHERE A1.Store_Name = A2.Store_Name
GROUP BY A1.Region_Name;
```

This gives us a view, **V_REGION_SALES**, that has been defined to store sales by region records. If we want to find out the content of this view, we type in,

```
SELECT * FROM V_REGION_SALES;
```

Result:

REGION	**SALES**
East	**700**
West	**2050**

CREATE VIEW Syntax

```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

SQL Updating a View

A view can be updated with the CREATE OR REPLACE VIEW command.

SQL CREATE OR REPLACE VIEW Syntax

```
CREATE OR REPLACE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

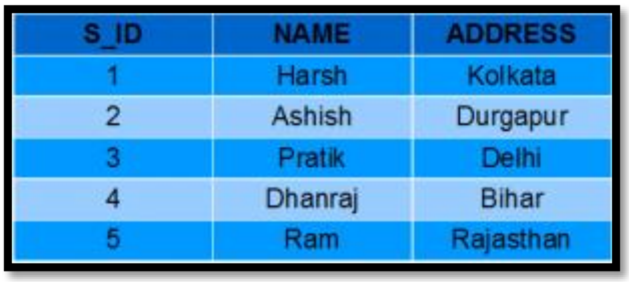
SQL Dropping a View

A view is deleted with the DROP VIEW command.

SQL DROP VIEW Syntax

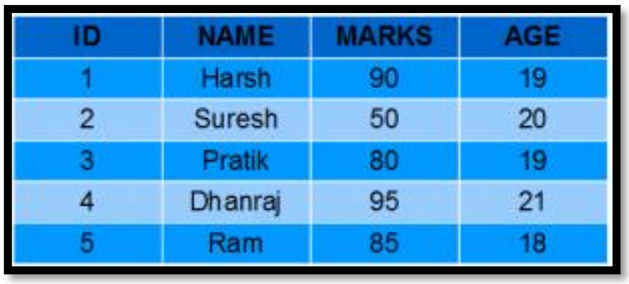
```
DROP VIEW view_name;
```

Sample Tables:



S_ID	NAME	ADDRESS
1	Harsh	Kolkata
2	Ashish	Durgapur
3	Pratik	Delhi
4	Dhanraj	Bihar
5	Ram	Rajasthan

StudentMarks



ID	NAME	MARKS	AGE
1	Harsh	90	19
2	Suresh	50	20
3	Pratik	80	19
4	Dhanraj	95	21
5	Ram	85	18

CREATING VIEWS

We can create View using **CREATE VIEW** statement. A View can be created from a single table or multiple tables.

Syntax:

```
CREATE VIEW view_name AS

SELECT column1, column2.....

FROM table_name

WHERE condition;
```

view_name: Name for the View
table_name: Name of the table
condition: Condition to select rows

Examples:

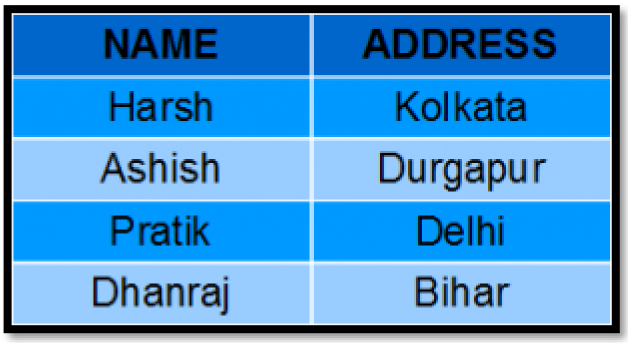
- **Creating View from a single table:**
  - In this example we will create a View named DetailsView from the table StudentDetails.
    Query:

```
CREATE VIEW DetailsView AS

SELECT NAME, ADDRESS

FROM StudentDetails

WHERE S_ID < 5;
```

To see the data in the View, we can query the view in the same manner as we query a table.

```
SELECT * FROM DetailsView;
```

Output:



NAME	ADDRESS
Harsh	Kolkata
Ashish	Durgapur
Pratik	Delhi
Dhanraj	Bihar

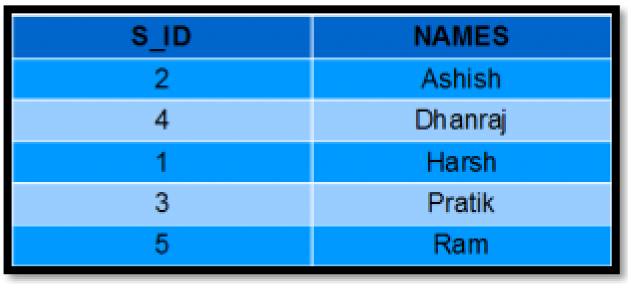
- In this example, we will create a view named StudentNames from the table StudentDetails.
  Query:

```
CREATE VIEW StudentNames AS

SELECT S_ID, NAME

FROM StudentDetails

ORDER BY NAME;
```

If we now query the view as,

```
SELECT * FROM StudentNames;
```

Output:



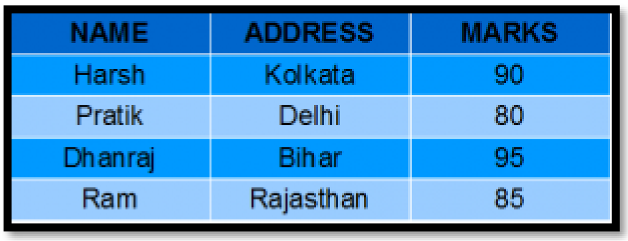
S_ID	NAMES
2	Ashish
4	Dhanraj
1	Harsh
3	Pratik
5	Ram

- **Creating View from multiple tables**: In this example we will create a View named MarksView from two tables StudentDetails and StudentMarks. To create a View from multiple tables we can simply include multiple tables in the SELECT statement. Query:
- CREATE VIEW MarksView AS
- SELECT StudentDetails.NAME, StudentDetails.ADDRESS, StudentMarks.MARKS
- FROM StudentDetails, StudentMarks
- WHERE StudentDetails.NAME = StudentMarks.NAME;

To display data of View MarksView:

```
SELECT * FROM MarksView;
```

Output:



NAME	ADDRESS	MARKS
Harsh	Kolkata	90
Pratik	Delhi	80
Dhanraj	Bihar	95
Ram	Rajasthan	85

DELETING VIEWS

We have learned about creating a View, but what if a created View is not needed any more? Obviously we will want to delete it. SQL allows us to delete an existing View. We can delete or drop a View using the DROP statement.

Syntax:

```
DROP VIEW view_name;
```

view_name: Name of the View which we want to delete.

For example, if we want to delete the View **MarksView**, we can do this as:

```
DROP VIEW MarksView;
```

UPDATING VIEWS

There are certain conditions needed to be satisfied to update a view. If any one of these conditions is **not** met, then we will not be allowed to update the view.

1. The SELECT statement which is used to create the view should not include GROUP BY clause or ORDER BY clause.
2. The SELECT statement should not have the DISTINCT keyword.
3. The View should have all NOT NULL values.
4. The view should not be created using nested queries or complex queries.
5. The view should be created from a single table. If the view is created using multiple tables then we will not be allowed to update the view.

- We can use the **CREATE OR REPLACE VIEW** statement to add or remove fields from a view.

- **Syntax**:

```
CREATE OR REPLACE VIEW view_name AS
SELECT column1,coulmn2,..
FROM table_name
WHERE condition;
```

For example, if we want to update the view **MarksView** and add the field AGE to this View from **StudentMarks** Table, we can do this as:

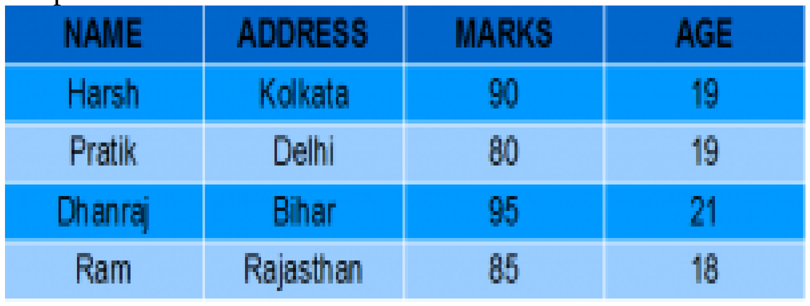
```
CREATE OR REPLACE VIEW MarksView AS
```

```
SELECT StudentDetails.NAME, StudentDetails.ADDRESS, StudentMarks.MARKS,
StudentMarks.AGE

FROM StudentDetails, StudentMarks

WHERE StudentDetails.NAME = StudentMarks.NAME;
```

If we fetch all the data from MarksView now as:

```
SELECT * FROM MarksView;
```

Output:



NAME	ADDRESS	MARKS	AGE
Harsh	Kolkata	90	19
Pratik	Delhi	80	19
Dhanraj	Bihar	95	21
Ram	Rajasthan	85	18

- **Inserting a row in a view**:
  We can insert a row in a View in a same way as we do in a table. We can use the INSERT INTO statement of SQL to insert a row in a View.**Syntax**:

```
INSERT INTO view_name(column1, column2 , column3,..)

VALUES(value1, value2, value3..);
```

- **view_name**: Name of the View

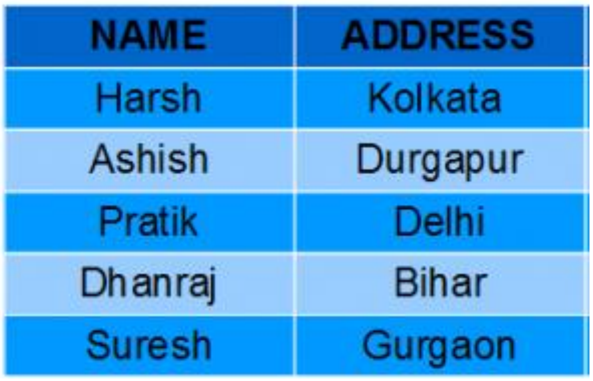
Example:
In the below example we will insert a new row in the View DetailsView which we have created above in the example of “creating views from a single table”.

```
INSERT INTO DetailsView(NAME, ADDRESS)

VALUES("Suresh","Gurgaon");
```

If we fetch all the data from DetailsView now as,

```
SELECT * FROM DetailsView;
```

Output:



NAME	ADDRESS
Harsh	Kolkata
Ashish	Durgapur
Pratik	Delhi
Dhanraj	Bihar
Suresh	Gurgaon

Deleting a row from a View:
Deleting rows from a view is also as simple as deleting rows from a table. We can use the DELETE statement of SQL to delete rows from a view. Also deleting a row from a view first delete the row from the actual table and the change is then reflected in the view.

Syntax:

- DELETE FROM view_name
- WHERE condition;
-
- **view_name**:Name of view from where we want to delete rows
- **condition**: Condition to select rows

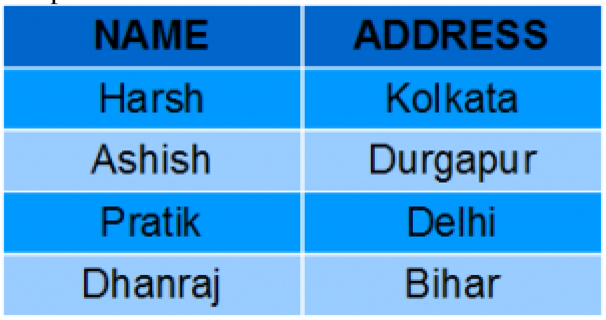
Example:
In this example we will delete the last row from the view DetailsView which we just added in the above example of inserting rows.

```
DELETE FROM DetailsView

WHERE NAME="Suresh";
```

If we fetch all the data from DetailsView now as,

```
SELECT * FROM DetailsView;
```

Output:



NAME	ADDRESS
Harsh	Kolkata
Ashish	Durgapur
Pratik	Delhi
Dhanraj	Bihar

WITH CHECK OPTION

The WITH CHECK OPTION clause in SQL is a very useful clause for views. It is applicable to a updatable view. If the view is not updatable, then there is no meaning of including this clause in the CREATE VIEW statement.

- The WITH CHECK OPTION clause is used to prevent the insertion of rows in the view where the condition in the WHERE clause in CREATE VIEW statement is not satisfied.
- If we have used the WITH CHECK OPTION clause in the CREATE VIEW statement, and if the UPDATE or INSERT clause does not satisfy the conditions then they will return an error.

Example:
In the below example we are creating a View SampleView from StudentDetails Table with WITH CHECK OPTION clause.

```
CREATE VIEW SampleView AS

SELECT S_ID, NAME

FROM  StudentDetails

WHERE NAME IS NOT NULL

WITH CHECK OPTION;
```

In this View if we now try to insert a new row with null value in the NAME column then it will give an error because the view is created with the condition for NAME column as NOT NULL.
For example,though the View is updatable but then also the below query for this View is not valid:

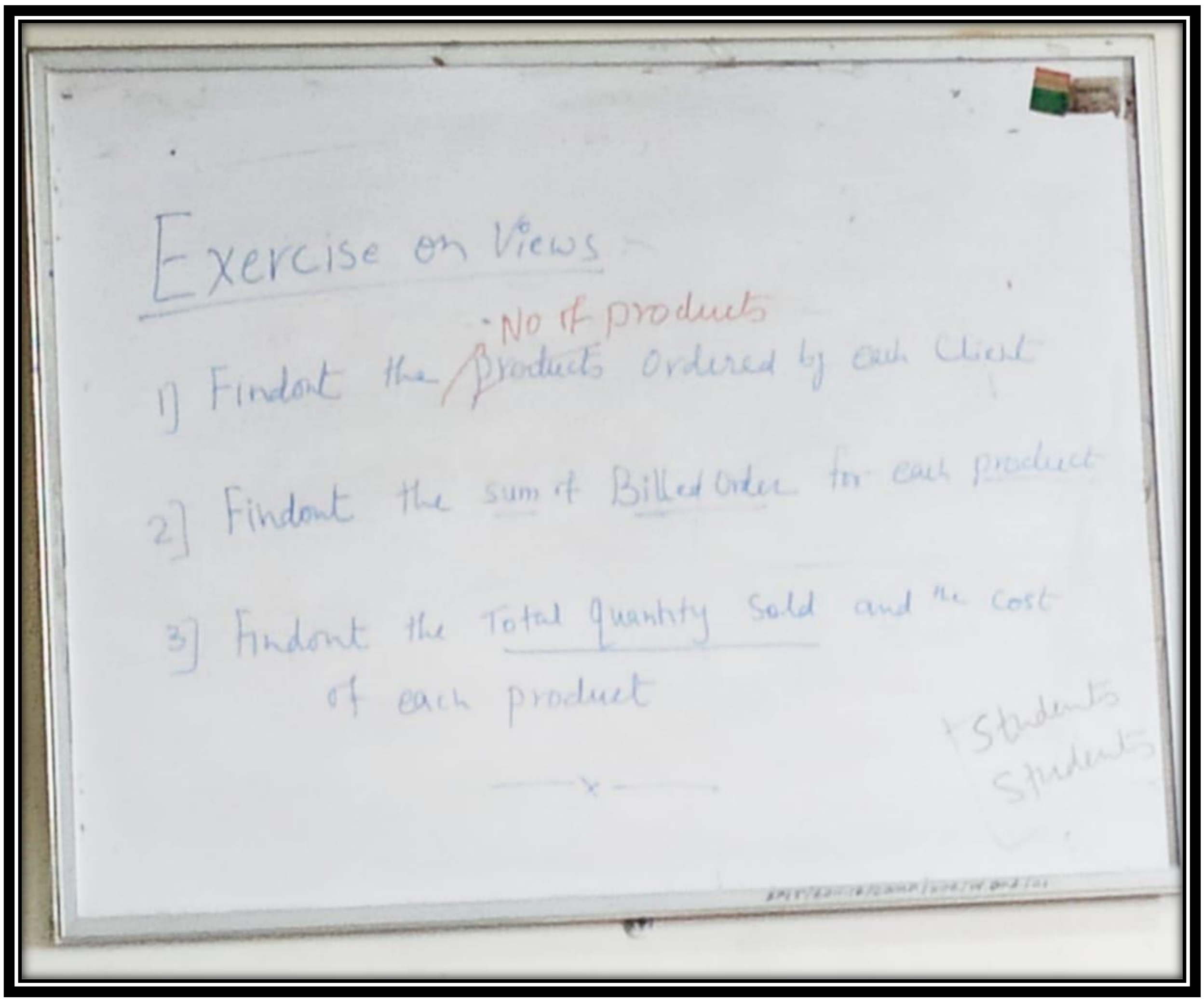
```
INSERT INTO SampleView(S_ID)

VALUES(6);
```

A view is nothing more than a SQL statement that is stored in the database with an associated name. A view is actually a composition of a table in the form of a predefined SQL query.

A view can contain all rows of a table or select rows from a table. A view can be created from one or many tables which depends on the written SQL query to create a view.

Views, which are a type of virtual tables allow users to do the following −

- Structure data in a way that users or classes of users find natural or intuitive.
- Restrict access to the data in such a way that a user can see and (sometimes) modify exactly what they need and no more.
- Summarize data from various tables which can be used to generate reports.



Exercise on Views

·No of products

1] Findout the /products Ordered by each Client

2] Findout the sum of Billed Order for each product

3] Findout the Total Quantity Sold and the cost of each product

—x—

Students
Students

```
1)mysql> create view first as select so.client_no , count(p.description) from product_master p
inner join sales_order_details sod on sod.product_no = p.product_no inner join sales_order so
on so.order_no = sod.order_no group by so.client_no;


Query OK, 0 rows affected (0.03 sec)

mysql> select * from first

+-----------      +----------------------    +

| client_no | count(p.description) |

+-----------      +----------------------    +

| C00001         |          5                  |

| C00002         |          1                  |

| C00003         |          4                  |

| C00004         |          2                  |

| C00005         |          2                  |

+-----------      +----------------------    +
5 rows in set (0.00 sec)

2)mysql> create view second as

-> select product_no , sum(qty_disp*product_rate) from sales_order_details
-> where order_no in (select order_no from sales_order where billed_yn = 'Y')
-> group by product_no;

mysql> select * from second;
+------------     +---------------------------        +
| product_no | sum(qty_disp*product_rate) |
+------------     +---------------------------        +
| P00001         |          5250.00          |
| P03453         |          6300.00          |
| P06734         |          12000.00        |
| P07868         |          9450.00          |
| P07885         |          5250.00          |
+------------     +---------------------------        +
5 rows in set (0.01 sec)

3)mysql> create view third as

-> select product_no , sum(qty_ordered) as Quantity_sold , sum(qty_ordered*product_rate) as
cost
-> from sales_order_details
-> group by product_no;

mysql> select * from third;
```

```
+------------      +          ---------------+----------+
| product_no |               Quantity_sold | cost    |
+------------      +          ---------------+----------+
| P00001          |           34        | 17850.00 |
| P03453          |           6          |6300.00 |
| P06734          |           1          | 12000.00 |
| P07868          |           3          |9450.00 |
| P07885          |           5          | 26250.00 |
| P07965          |           3          | 25200.00 |
| P07975          |           6          |6300.00 |
 +------------+---------------+----------+

7 rows in set (0.00 sec)
```

Conclusion:

Thus, Views in SQL are kind of virtual tables. A view also has rows and columns as they are in a real table in the database. We can create a view by selecting fields from one or more tables present in the database. A View can either have all the rows of a table or specific rows based on certain condition.