

Title: Write SQL code for creating of View. Perform Insert ,Modify, Delete records through view, Delete the View.
Working with Nested -Query.

Theory:

1. Simple View

→ Simple View in SQL is the view created by involving only single table. In other words we can say that there is only one base table in case of Simple View in SQL

Syntax : **CREATE VIEW** view-name **AS**
SELECT column-name
FROM table-name
WHERE condition

Example : **CREATE VIEW** Brazil_Customers **AS**
SELECT CustomerName, ContactName
FROM Customers
WHERE Country = 'Brazil';

2. Complex view

→ On other hand, Complex View is created by involving more than one table i.e., multiple tables get projected in Complex view.

Syntax : **CREATE VIEW** view-name **AS**
SELECT column-name
FROM table-name
JOIN aggregate-function
GROUP BY column-name

Example : **CREATE VIEW** dept_income **AS**
SELECT d.Name as DepartmentName, sum(e.salary) as
TotalSalary
FROM Employees e
JOIN Departments d on e.DepartmentId = d.id
GROUP BY d.Name

3. INSERT query syntax and example.

→ *INSERT* command is used to insert data into the row of a table.

Syntax : **INSERT INTO** TABLE_NAME (col1, col2, col3, ... col N)
VALUES (value1, value2, value3, valueN);

Example : **INSERT INTO** fyfs (name, roll) VALUES ("Rupesh", "FS43");

4. UPDATE query syntax and example.

→ *UPDATE* command is used to update or modify the value of a column in the table.

Syntax : **UPDATE** table_name SET [column_name1=value1] [WHERE
CONDITION]

Example : **UPDATE** students SET User_Name = "Rupesh" WHERE
Student_Id = '43'

5. DELETE query syntax and example.

→ *DELETE* is used to remove one or more row from a table.

Syntax : **DELETE FROM** table_name [WHERE condition];

Example : **DELETE FROM** javatpoint WHERE Author="Rupesh";

6. DROP VIEW

→ *DROP VIEW* command is used to delete the view

Syntax : **DROP VIEW** view_name

Example : **DROP VIEW** [Brazil Customers]

7. NESTED QUERY

→ Query written inside a query is called as SQL Nested Query

Syntax : **SELECT** Column1,Column2... **From** Table_Name
WHERE Column_Name Operator(**SELECT** Column1,Column
From Table_Name_2)
Operator (**SELECT** Column1,Column2.....**From**
Table_Name_3)

Example : **SELECT** Model **FROM** Product
WHERE ManufacturerID **IN** (**SELECT** ManufacturerID
FROM
Manufacturer
WHERE Manufacturer = 'Dell')

Outputs:

Initial tables setup:

Students table:

	name	marks	favSubject
▶	Sam	70	Python
	Johnson	60	DBMS
	Markus	70	Java
	Stoinis	60	DBMS
	Smith	80	Python
	Williamson	30	Python
	Emily	20	Java
	Deviliers	90	JavaScript

academicSubjects table:

	sub_name	studentRating	teachingSemester
▶	DBMS	6.5	3
	Java	7	3
	Javascript	7.9	4
	Python	8.5	3
✱	NULL	NULL	NULL

Creating Simple view:

The screenshot shows the SQL Enterprise Manager interface. On the left, the 'SCHEMAS' pane displays a tree view of the database structure, including 'company', 'friends', 'sakila', 'students', 'academicsubjects', 'students', 'Views', 'passedstudents', 'Stored Procedures', 'Functions', 'sys', and 'world'. The 'students' schema is expanded, showing 'Tables' and 'Views'. The 'Views' folder is selected, and the 'passedstudents' view is highlighted. The main pane shows the SQL script for creating the 'passedstudents' view. The script is as follows:

```
13
14 -- Simple View
15 CREATE VIEW passedStudents AS SELECT * FROM students WHERE marks>=35;
16 SELECT * FROM passedStudents;
17 CREATE VIEW failedStudents AS SELECT * FROM students WHERE marks<35;
18 SELECT * FROM failedStudents;
19
20 -- Complex View
```

Below the script, the 'Result Grid' shows the data for the 'passedstudents' view. The table has three columns: 'name', 'marks', and 'favSubject'. The data is as follows:

name	marks	favSubject
Sam	70	Python
Johnson	60	DBMS
Markus	70	Java
Stoinis	60	DBMS
Smith	80	Python
Devillers	90	JavaScript

The screenshot shows the SQL Enterprise Manager interface. On the left, the 'SCHEMAS' pane displays a tree view of the database structure, including 'company', 'friends', 'sakila', 'students', 'academicsubjects', 'students', 'Views', 'passedstudents', 'Stored Procedures', 'Functions', 'sys', and 'world'. The 'students' schema is expanded, showing 'Tables' and 'Views'. The 'Views' folder is selected, and the 'passedstudents' view is highlighted. The main pane shows the SQL script for creating the 'passedstudents' view. The script is as follows:

```
13
14 -- Simple View
15 CREATE VIEW passedStudents AS SELECT * FROM students WHERE marks>=35;
16 SELECT * FROM passedStudents;
17 CREATE VIEW failedStudents AS SELECT * FROM students WHERE marks<35;
18 SELECT * FROM failedStudents;
19
20 -- Complex View
```

Below the script, the 'Result Grid' shows the data for the 'passedstudents' view. The table has three columns: 'name', 'marks', and 'favSubject'. The data is as follows:

name	marks	favSubject
Williamson	30	Python
Emily	20	Java

Creating complex view:

The screenshot shows the SQL Studio interface. On the left, the 'SCHEMAS' pane displays a tree view with 'students' expanded, showing 'Tables' and 'Views'. The main editor contains SQL code for creating a complex view. The 'Result Grid' shows the output of the view.

```
19
20 -- Complex View
21 CREATE VIEW subjectToMarks AS SELECT favSubject, AVG(marks) FROM students GROUP BY favSubject;
22 SELECT * FROM subjectToMarks;
23 CREATE VIEW studentsToFavSubject AS SELECT favSubject, COUNT(*) AS No_of_students FROM students GROUP BY favSubject;
24 SELECT * FROM studentsToFavSubject;
25
26
```

favSubject	AVG(marks)
Python	60.0000
DBMS	60.0000
Java	45.0000
JavaScript	90.0000

The screenshot shows the SQL Studio interface. On the left, the 'SCHEMAS' pane displays a tree view with 'students' expanded, showing 'Tables' and 'Views'. The main editor contains SQL code for creating a complex view. The 'Result Grid' shows the output of the view.

```
19
20 -- Complex View
21 CREATE VIEW subjectToMarks AS SELECT favSubject, AVG(marks) FROM students GROUP BY favSubject;
22 SELECT * FROM subjectToMarks;
23 CREATE VIEW studentsToFavSubject AS SELECT favSubject, COUNT(*) AS No_of_students FROM students GROUP BY favSubject;
24 SELECT * FROM studentsToFavSubject;
25
26
```

favSubject	No_of_students
Python	3
DBMS	2
Java	2
JavaScript	1

studentsToFavSubject 57 x

Output:

Insert delete and update on view:

The screenshot shows the SQL Studio interface. On the left, the 'SCHEMAS' pane displays a tree view with 'students' expanded, showing 'Tables' and 'Views'. The main editor contains SQL code for inserting, deleting, and updating data in a view. The 'Result Grid' shows the output of the view.

```
26
27 -- Insert, delete, update query on view
28 INSERT INTO passedStudents VALUES("Jonty", 36, "Javascript");
29 UPDATE passedStudents SET marks = 75 WHERE name = "Stoinis";
30 DELETE FROM passedStudents WHERE name = "Sam";
31 SELECT * FROM passedStudents; -- Should reflect changes
32
33 DROP VIEW passedStudents;
```

name	marks	favSubject
Johnson	60	DBMS
Markus	70	Java
Stoinis	75	DBMS
Smith	80	Python
Devillers	90	JavaScript
Jonty	36	JavaScript

Drop view:

SCHEMAS

Filter objects

company

friends

sakila

students

Tables

academicSubjects

students

Views

failedStudents

studentstofavSubject

subjecttomarks

Stored Procedures

Functions

sys

world

Administration Schemas

Information

Limit to 1000 rows

31

SELECT * FROM passedStudents; -- Should reflect changes

32

33

DROP VIEW passedStudents;

34

SELECT * FROM passedStudents; -- Should give error

35

36

-- Preparing 2nd table

37

CREATE TABLE academicSubjects(

38

sub_name VARCHAR(20) PRIMARY KEY,

39

studentRating FLOAT,

Output

Action Output

#	Time	Action	Message
1	18:07:34	DROP VIEW passedStudents	0 row(s) affected
2	18:07:34	SELECT * FROM passedStudents LIMIT 0, 1000	Error Code: 1146. Table 'students.passedStudents' doesn't exist

Nested query on single table:

SCHEMAS

Filter objects

company

friends

sakila

students

Tables

academicSubjects

students

Views

failedStudents

studentstofavSubject

subjecttomarks

Stored Procedures

Functions

sys

world

Administration Schemas

Information

Limit to 1000 rows

50

-- Nested query on single table

51

-- Gives marks of students with marks more than max marks of students having favSubjects as DBMS or Java

52

53

SELECT marks, name FROM students WHERE marks > (SELECT MAX(marks) FROM students WHERE favSubject IN("DBMS", "Java"));

54

Result Grid

marks	name
80	Smith
90	Devillers

students 63 x

Output

Action Output

#	Time	Action	Message
1	18:11:06	SELECT marks, name FROM students WHERE marks > (SELECT MAX(marks) FROM stud...	2 row(s) returned

Nested query using two tables:

SCHEMAS

Filter objects

company

friends

sakila

students

Tables

academicSubjects

students

Views

failedStudents

studentstofavSubject

subjecttomarks

Stored Procedures

Functions

sys

world

Administration Schemas

Information

Limit to 1000 rows

55

-- Nested query using two tables

56

-- Gives students having favSubject having rating over 7 (Ratings are in other table)

57

58

SELECT * FROM students WHERE favSubject IN(SELECT sub_name FROM academicSubjects WHERE studentRating > 7);

59

Result Grid

name	marks	favSubject
Smith	80	Python
Williamson	30	Python
Devillers	90	JavaScript
Jonty	36	JavaScript

students 64 x

Output

Action Output

#	Time	Action	Message
1	18:11:06	SELECT marks, name FROM students WHERE marks > (SELECT MAX(marks) FROM stud...	2 row(s) returned
2	18:14:41	SELECT * FROM students WHERE favSubject IN(SELECT sub_name FROM academicSu...	4 row(s) returned

Conclusion: Thus, we created, modified(Inserted updated deleted) and deleted views