1	Hardik	2000.00
2	Hardik	1500.00
3	Hardik	2000.00
4	Hardik	6500.00
6	Hardik	4500.00
1	Komal	2000.00
2	Komal	1500.00
3	Komal	2000.00
1	Muffy	2000.00
2	Muffy	1500.00
3	Muffy	2000.00
4	Muffy	6500.00
5	Muffy	8500.00
6	Muffy	4500.00

VIEWS:

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.

2 Types of Views Updatable and Read-only -views

Unlike base tables, VIEWs are either updatable or read-only, but not both. INSERT, UPDATE, and DELETE operations are allowed on updatable VIEWs and base tables, subject to any other constraints. INSERT, UPDATE, and DELETE are not allowed on read-only VIEWs, but you can change their base tables, as you would expect. An updatable VIEW is one that can have each of its rows associated with exactly one row in an underlying base table.

When the VIEW is changed, the changes pass unambiguously through the VIEW to that underlying base table. Updatable VIEWs in Standard SQL are defined only for queries that meet these criteria:

1.Built on only one table
2.No GROUP BY clause
3.No HAVING clause
4.No aggregate functions
5.No calculated columns
6.No UNION, INTERSECT, or EXCEPT

7.No SELECT DISTINCT clause
8.Any columns excluded from the VIEW must be NULL-able or have a DEFAULT in the base table, so that a whole row can be constructed for insertion By implication, the VIEW must also contain a key of the table.

In short, we are absolutely sure that each row in the VIEW maps back to one and only one row in the base table. Some updating is handled by the CASCADE option in the referential integrity constraints on the base tables, not by the VIEW declaration.

The definition of updatability in Standard SQL is actually fairly limited, but very safe. The database system could look at information it has in the referential integrity constraints to widen the set of allowed updatable VIEWs. You will find that some implementations are now doing just that, but it is not common yet.

The SQL Standard definition of an updatable VIEW is actually a subset of the possible updatable VIEWs, and a very small subset at that. The major advantage of this definition is that it is based on syntax and not semantics.

Examples of Updatable and Non-updatable View.

CREATE VIEW view_1 AS SELECT * FROM Table1 WHERE x IN (1,2);
-- updatable, has a key!

CREATE VIEW view_2 AS SELECT * FROM Table1 WHERE x = 1 UNION ALL SELECT * FROM Table1 WHERE x = 2;
-- not updatable!

More about Views:

A view takes up **no storage space** other than for the definition of the view in the data dictionary.

A view contains **no data**. All the data it shows comes from the base tables.

A view can provide an additional level of **table security** by restricting access to a set of rows or columns of a table.

A view **hides implementation complexity**. The user can select from the view with a simple SQL, unaware that the view is based internally on a join between multiple tables.

A view lets you **change the data** you can access, applying operators, aggregation functions, filters etc. on the base table.

A view **isolates applications from changes** in definitions of base tables. Suppose a view uses two columns of a base table, it makes no difference to the view if other columns are added, modified or removed from the base table.

To know about the views in your own schema, look up **user_views**.

The underlying SQL definition of the view can be read via **select text from user_views** for the view.

Oracle does not enforce **constraints on views**. Instead, views are subject to the constraints of their base tables.

Creating Views

Database views are created using the **CREATE VIEW** statement. Views can be created from a single table, multiple tables or another view. To create a view, a user must have the appropriate system privilege according to the specific implementation.

Syntax:

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

we can include multiple tables in your SELECT statement in a similar way as we use them in a normal SQL SELECT query.

Example:
Consider the CUSTOMERS table having the following records −

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmadabad	2000.00
2	Khilan	25	Delhi	1500.00
3	Kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

Following is an example to create a view from the CUSTOMERS table. This view would be used to have customer name and age from the CUSTOMERS table.

SQL > CREATE VIEW CUSTOMERS_VIEW AS SELECT name, age FROM CUSTOMERS;

Now, you can query CUSTOMERS_VIEW in a similar way as you query an actual table. Following is an example for the same.

*SQL > SELECT * FROM CUSTOMERS_VIEW;*

This would produce the following result.

NAME	AGE
Ramesh	32
Khilan	25
Kaushik	23
Chaitali	25
Hardik	27
Komal	22
Muffy	24

The With Check Option:

The WITH CHECK OPTION is a CREATE VIEW statement option. The purpose of the WITH CHECK OPTION is to ensure that all UPDATE and INSERTs satisfy the condition(s) in the view definition. If they do not satisfy the condition(s), the UPDATE or INSERT returns an error.

Example:

```
SQL>  CREATE VIEW CUSTOMERS_VIEW AS
      SELECT name, age FROM  CUSTOMERS
      WHERE age IS NOT NULL WITH CHECK OPTION;
```

The WITH CHECK OPTION in this case should deny the entry of any NULL values in the view's AGE column, because the view is defined by data that does not have a NULL value in the AGE column.

Updating a View

A view can be updated under certain conditions which are given below −

- The SELECT clause may not contain the keyword DISTINCT.
- The SELECT clause may not contain summary functions.
- The SELECT clause may not contain set functions.
- The SELECT clause may not contain set operators.
- The SELECT clause may not contain an ORDER BY clause.
- The FROM clause may not contain multiple tables.
- The WHERE clause may not contain subqueries.
- The query may not contain GROUP BY or HAVING.
- Calculated columns may not be updated.
- All NOT NULL columns from the base table must be included in the view in order for the INSERT query to function.

So, if a view satisfies all the above-mentioned rules then you can update that view. The following code block has an example to update the age of Ramesh.

```
SQL > UPDATE CUSTOMERS_VIEW SET AGE = 35 WHERE name = 'Ramesh';
```

This would ultimately update the base table CUSTOMERS and the same would reflect in the view itself. Now, try to query the base table and the SELECT statement would produce the following result.

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	35	Ahmadabad	2000.00
2	Khilan	25	Delhi	1500.00
3	Kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

Inserting Rows into a View

Rows of data can be inserted into a view. The same rules that apply to the UPDATE command also apply to the INSERT command. Here, we cannot insert rows in the CUSTOMERS_VIEW because we have not included all the NOT NULL columns in this view, otherwise you can insert rows in a view in a similar way as you insert them in a table.

Deleting Rows from a View

Rows of data can be deleted from a view. The same rules that apply to the UPDATE and INSERT commands apply to the DELETE command.

Following is an example to delete a record having AGE = 22.

SQL > DELETE FROM CUSTOMERS_VIEW WHERE age = 22;

This would ultimately delete a row from the base table CUSTOMERS and the same would reflect in the view itself. Now, try to query the base table and the SELECT statement would produce the following result.

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	35	Ahmadabad	2000.00
2	Khilan	25	Delhi	1500.00
3	Kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
7	Muffy	24	Indore	10000.00

Dropping Views

Obviously, where you have a view, you need a way to drop the view if it is no longer needed.

Syntax:

```
DROP VIEW view_name;
```

Example:

```
SQL> DROP VIEW CUSTOMERS_VIEW;
```

SET OPERATIONS

- These operators are used to combine information of similar datatype from one or more than one table.
- Datatype of the corresponding columns in all the select statement should be same.
- Different types of set commands are
  - UNION
  - UNION ALL
  - INTERSECT
  - MINUS
- Set operators are combine 2 or more queries into one result .
- The result of each SELECT statement can be treated as a set and SQL set operators can be applied on those sets to arrive at a final result.
- SQL statements containing set operators are referred to as compound queries, and each SELECT statements in a command query in referred to as a compound query.
- Set operations are often called vertical joins, as a result combines data from 2 or more SELECT based on columns instead of rows.

Syntax:

```
<compound query>
        { UNION | UNION ALL | MINUS | INTERSECT }
<compound query>
```