Aggregate Functions, Clauses & Views

Aggregate Functions:

Aggregate functions work on a specific column data in the table. These functions classified into

1. COUNT( ): This function is used to count number of records in the table. We use in 3 ways.

a) COUNT(*): It returns the number of records are available in the table including duplicate & null values.

Example: SELECT COUNT(*) FROM TICKET;

Output: 10

b) COUNT(COLUMN_NAME): It returns the number of records including duplicate values in the table.

Example: SELECT COUNT(PRICE) FROM TICKET;

Output: 10

c) COUNT(DISTINCT COLUMN_NAME): It returns number of records without duplicate & null values.

Example: SELECT COUNT(DISTINCT PRICE) FROM TICKET;

Output: 6

2. SUM( ): It returns sum value from a specific column data.

Example: SELECT SUM(PRICE) FROM TICKET;

Output: 22400

3. AVG( ): It returns average value from a specific column data.

Example: SELECT AVG(PRICE) FROM TICKET;

Output: 2240.00

4. MAX( ): It returns maximum value from a specific column data.

Example: SELECT MAX(PRICE) FROM TICKET;

Output: 3000

5. MIN( ): It returns minimum value from a specific column data.

Example: SELECT MIN(PRICE) FROM TICKET;

Output: 1500

GROUP BY: Group by clause is used to divide similar data as a group. If we implement group by clause in the query then we should use aggregate function in the query.

When we execute group by clause, first the data within the table will divide a separate group and later the aggregate function will be executed on each group data to get result.

Example 1: SELECT BNO, COUNT(*) TOTAL_TICKETS FROM TICKET GROUP BY BNO;

BNO	TOTAL_TICKETS
10	2
20	2
30	1
40	2
50	2
60	1

Example 2: SELECT BNO, SUM(PRICE) TOTAL_BILL FROM TICKET GROUP BY BNO;

BNO	TOTAL_BILL
10	3000
20	4000
30	3000
40	4800
50	5000
60	2600

HAVING: Having clause is also used for filtering the records after grouping the data just like where clause.

Example 1: SELECT BNO, COUNT(*) TOTAL_TICKETS FROM TICKET GROUP BY BNO HAVING COUNT(*) > 1;

BNO	TOTAL_TICKETS
10	2
20	2
40	2
50	2

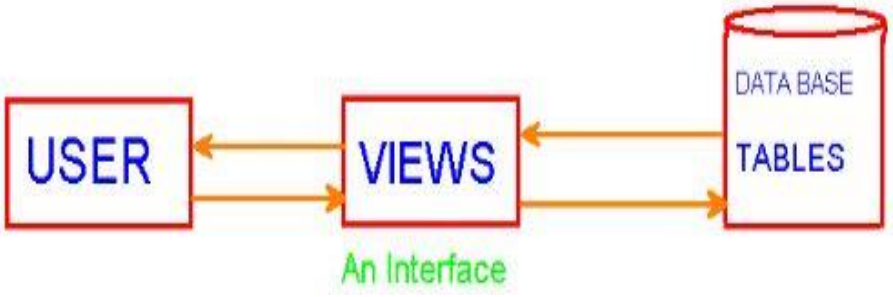
Example 2: SELECT BNO, SUM(PRICE) TOTAL_BILL FROM TICKET GROUP BY BNO HAVING SUM(PRICE)>3000;

BNO	TOTAL_BILL
20	4000
40	4800
50	5000

VIEWS:

View is database object which is like table but logical. It is dependent whereas table an independent because view is extracted from the table.

View will act as an interface between the data provider (table) and the user.



We can create a view in 2 ways

1. Simple View/Updatable View
2. Complex View/Non-Updatable View

Syntax: CREATE VIEW <VIEW_NAME> AS SELECT */<COLUMN_NAME> FROM <TABLE_NAME>;

Simple View/Updatable View:

When we create a view on a single table is called Simple view. We can perform DML operations on simple view so that a simple view is called as Updatable view.

Example 1: CREATE VIEW SV1 SELECT * FROM BUS;

SELECT * FROM SV1;

BNO	BNAME	SOURCE	DESTINATION	ARRTIME	DEPTIME
10	ORANGE	HYDERABAD	BANGALORE	19:30:00	05:00:00
20	KAVERI	HYDERABAD	CHENNAI	20:30:00	06:00:00
30	KESINENI	HYDERABAD	MUMBAI	21:30:00	10:00:00
40	KOMITLA	BANGALORE	DELHI	11:30:00	20:00:00
50	DIWAKAR	CHENNAI	MYSORE	10:30:00	19:00:00
60	KALESWARI	MUMBAI	TIRUPATI	08:30:00	23:00:00

Example 2: DROP VIEW SV1;

Example 3: CREATE VIEW SV2 SELECT BNO, BNAME FROM BUS;

SELECT * FROM SV2;

BNO	BNAME
10	ORANGE
20	KAVERI
30	KESINENI
40	KOMITLA
50	DIWAKAR
60	KALESWARI

Example 4: DROP VIEW SV2;

Complex View/Non-Updatable View:

When we create a view more than one table is called Complex view. We can't perform DML operations on complex view so that complex view is called as Non-updatable view.

Example: CREATE VIEW CV1 AS SELECT BUS.BNO, TICKET.TID, TICKET.PRICE FROM BUS JOIN TICKET ON BUS.BNO=TICKET.BNO;

SELECT * FROM CV1;

BNO	TID	PRICE
10	103	1500
10	107	1500
20	101	2000
20	104	2000
30	102	3000
40	109	2400
40	110	2400
50	105	2500
50	108	2500
60	106	2600