SQL. add, modify, rename, drop.

1) DDL – Create, alter, rename, drop
2) DML – Insert, update, delete, Merge
3) DQL – Select Stmt.
4) DCL – Grant, revoke.
5) TCL – Commit, rollback, Savepoint, Truncate.

DDL:

1) create table tab_name (parameter1, parameter2 ...);
2) Alter – add, modify, rename, drop.

(i) alter table tab_name add (column1, column2, ...);
(ii) alter table tab_name modify (column1, column2 ...);
Note: Modify only possible when the column is empty.
(iii) alter table tab_name rename column old_column_name to New_column_name;
(iv) alter table tab_name drop (col1, col2 ...); or drop column column_name;
↳ drop more than 1 column ↓ dropping only one column.

3) rename old_table_name to new_tab_name;
4) drop table tab_name;

DML: 1) insert into tab_name values (1, 2, ...);
high level Insert: insert into tab_name (select col1, col2, from tab_name);
2) update table_name set column1 = value1, column2 = value2, ...;
high level update: update tab_name set column1 = value1; (with out where clause).
3) delete table_name; (high level delete), with where clause deletes desired rows.

DQL: Select Stmt (*) → projection operator.

4) DCL:

1) Grant
2) Revoke
} Permissions (insert, update, delet, Select).

*Grant all on tab-name to Public; (will assign all permissions of table to all users in DB).

*revoke update, insert on tab-name from Public;

Sharing Selected Columns to other users: (only insert, update supported).

Select, delete → Not supported.

*~~Grant insert on~~ tab-name(~~column1~~, column2, ..) ~~to~~ user1;

> Grant insert(empno, ename), update(empno, ename, job) on emp to Public;

Scott ⟶ User1
Privilages

Note: changes made in the User1 reflect in Scott only when Commit in User1;

*
grant all on emp to public;
↓
revoke all on emp from public;
doesn't work ← user1

grant → Public ✓ grant → User1 ✓
revoke → public ✗ revoke → User1 ✗

(only Public will cancell the privilages where as (pelifing User1 doesn't work).

TCL: ↑ implicit (DDL)
→ Explicit (DML)

Commit: saving DML operations permanently.

rollback: get back the data of non Commited DML operations.

savepoint: go back to the un Commited Save point.

rollback to S1

Note: Placing Commit, (or) rollback will clear save points

DDL;
DDL;
S1;
DDL;
DDL;
rollback to S1
Commit;
↓
saves first DDL statements

Truncate: Delete + Commit

Structure of table is alive but data is lost permanently.

Operators in SQL:

Arithematic	Comparision operators / Relational	Logical	Special operators
+	=	AND	IN, NOT IN
-	!=, <>	OR	BETWEEN, NOT BETWEEN (AND)
*	<=	NOT	LIKE, NOT LIKE
/	>=	Order of exec: NOT AND OR.	IS NULL, IS NOT NULL
	>		
	<		
	Some / Any.		

Truth Tables:

AND:

AND	T	F	N
T	T	F	N
F	F	F	F
N	N	F	N

T → 1
F → 0

NOT:

NOT	T	F	N
NOT	F	T	N

OR:

OR	T	F	N
T	T	T	T
F	T	F	N
N	T	N	N

Priority:
All comparision operators
NOT
AND
OR.

q_1 AND q_2 OR q_3 or q_4 AND q_5 AND q_6 AND q_7 → AND will take the priority.

(R_1) (R_2)

R_1 OR q_3 OR R_2 (either of the values)

Special operators: NOT IN with NULL value returns no rows.

IN, NOT IN (Searches particular value from list of values).

Select * from emp where hiredate NOT IN ('18-JAN-1987', '20-FEB-2010');

↓

Relogrix only oracle formate

dd-Mon-yy (or)

dd-Month-yyyy.

Between, Not Between: is always used with 'AND' operator.

Select * from emp where Sal Not Between 2000 AND 5000;

↓

Special operator

My NOT SAL between 2000 AND 5000; result same as above.

↓

Logical operator.

* between lo_value and hi_value (between should operate b/w lo-val & hi-val but not hi-val & lo-val).

Like, Not Like: Like operator used to search for particular pattern of character.

Wild card character: The char's which are used to search particular pattern of characters is called wild card char's.

2 char's: 1) %. (represent any seq of zero or more char's)

2) _ (under score) {Represent any single char, only At that position}.

1) Placing %. before or after doesn't bother what ever or any no. of char's.

> Select * from ~~ename~~ emp ~~like~~ where ename like 'K%.' OR ename like '%T%.'

%ANA%., %K%., %K , K%.,

2) '_' represents single char

> Select * from emp where ename Not Like '_ _ _';

'_ _ Ki_ _', '_K%.';

~~IN, NOT IN~~:

IS NULL, IS NOT NULL:

> Select * from emp where Comm IS NOT NULL;

* Any Arithematic operation with NULL is Null. ~~Zero~~

Alias: Colum names in the select List can be aliased with as.

Note: Alias names not recognised in Where, group by, having and

" " will be recognised in order by clause (works while displaying)

Clauses: select distinct Column1, Column2 from tab where= group by having order by.

1) distinct
2) from
3) where
4) group by
5) having
6) order by.

operators of the same priority are evaluated from left to Right.

7) Join clause
8) ON clause

1) Distinct: Will eliminate all duplicate rows mentioned in the distinct clause.
> Select distinct ename, Job from emp; [Supress the duplicate distinct values in both columns and display that 2 columns].

2) From: placing one or more tables.
Writing a select stmt in from clause called In line view.

3) Where clause: Filter the data from the table. Supports with update, delete, Select but not with INSERT. * Doesn't recognise Alias

Select * from tab_name where Arithmetic expression, Function, column name, constant, list of values.

4) group by clause: Used to calculate grouped values by grouping the rows.
> Select Column1, Column2 from tab_name group by Column1, Column2;
* Column Aliases cannot be used in group by clause.
* By default rows are sorted by Ascending order of the columns included in the group by list.

5) Having clause: Making a condition on grouped results. And group functions only exist in the having clause
Having clause doesn't recognise alias names of grouped results. only grouped f^ns or grouped expressions.

grouped expression result

First rows are grouped
↓
Second Group f^n is applied to identified groups
↓
Third groups that match the criteria in the having clause are displayed.

* Existence of group by clause does not guarantee the existance of having clause. But existence of having clause demands the group by clause.

6) Order by clause: Used to sort the list of specified columns, or grouped columns in order by clause either in ascending or descending. can sort by using alias names aswell.

> Select * from emp order by deptno, sal; (first 'deptno' is sorted in Asc then 'sal' will be sorted with in the dept)

* order by deptno, sal desc;

* order by 2; → second column will be sorted (in the select list)

select empno, ename, deptno from emp order by 3 (select list deptno will be sorted)

" " order by 5 x (5 colums not there in select list)

7) Join:

8) ON:

Precedence:

distinct <Column List> ③

① ⑤ Select <Column List> / from <Tab_Name> ② where <Condition>

⑤ distinct

distinct clause { ④ ③ group by <Column List>; ⑤ ④ having <Condition on grouped result>; ⑦ ⑥ order by <Column List> }

Precedence of operators: ① All comparition operators ② NOT operator ③ AND ④ OR

FUNCTIONS IN SQL.

1) Single row f^n : Returns single result for every row of a queried Table.
2) Multiple row f^n: perform calculation on group of rows and return one result per group of Row.
3) General f^n: NVL, DECODE

* Single Row f^n: can appear in Select List, Where clause, Order by clause, Start with clause, Connect by clause.
  - 1. char. f^n
  - 2. Num f^n
  - 3 Date f^n
  - 4 Conversion f^n.

1) char f^n: (→ No of Arguments)
1) Lower(1); 4) length(1); 7) chr(Num); 10) Replace('s', 'ss', 'vv')
2) Upper(1); 5) Concat(s1, s2); 8) SubStr('s', M, N); 11) Translate('s', 'ss', 'RS')
3) InItCap(1); 6) Ascii('chr'); 9) Instr('s', 'ss', M); 12) Lpad
13) Rpad 14) RTRIM 15) LTRIM 16) TRIM.

SubStr('s', M, N): returns String
M = 0 => Starts from 1st
M = -ve → Starts from reverse to end.
N = Omitted → displays all chars untill end
N = -ve, 0 → Null. returns.

INSTR('s', 'ss', M, N): Searches the position Sub string 'ss' in Main String 's' how many times it has been occured. Return Position. returns integer
M = +ve Starts from beginning
M = -ve Starts from ending to beginning.
'N' N of times occured in Main string 's'
Default M = N = 1 If M = N = 0 Result is 0 → error.

Replace('KIRAN', 'KI', '123'): 123RAN (Value by value) Replace
s v1 v2

TRANSLATE('KIRAN', 'ABCKDI', '1234567') = 46RAN (Char by char Replace)
(Specifying 'space' will be replaced space with every char.)

Lpad: Lpad('Kiran', 15, '*$'); pads '*$' from left in total 15 chars.
Rpad:

TRIM: Trims the specified chars from the supplied string

TRIM(' Kiran '); empty spaces get trimmed.

~~TRIM ('XX')~~

> Trim (leading 'Ki' from 'KiKiKiran'); Trim leading 'Ki'

> Trim (Trailing 'Ki' from 'KiKiranKiKi'); ending 'Ki'.

> LTRIM ('ABABABCDABB', 'AB'), 'AB' is trimmed from left side.

> RTRIM ('ABABCDEFABAB', 'AB'); " " Right Side

Number fn': 1) Round 3) Floor 5) Power(m,n) 7) Sqrt(n) 9) Sign(-n)
2) Truncate 4) Ceil 6) mod(m,n) 8) Abs(n) 10) Sin(30°)

Round(m,n) - m truncated to 'n' decimal points. (+ve after decimal point, -ve before decimal point).

Round (20.49) - 20
Round (20.50) - 21

Round (20234.5693, 2) - 20234.56
Rounding 2 digit.
Round (2054.24, -2) - 2000

Truncate: Truncate(m, n) - 'm' truncated to 'n' decimal points.

Truncate
Trunc (23.99) - 23
Trunc (23.01) - 23

Trunc (25.345, 2) - 25.34
Trunc (2564.342, -2) - 2500

ceil (25.99) - 26
ceil (25.001) - 26

floor (25.99) - 25
floor (25.001) - 25

Date functions: DD-MON-YY → Standard formate

Date + Number = Adding no of days to the date
Date - Number = Substract no days from date
Date - date = Returns no of days between two dates.
Date + N/24 = Adding 'N' hours to that particular date

Days → week days/7
Days → months days/30
months → years months/12

1) Add_months(D, ± N) - Add or Sub 'N' no of months to that particular date

2) Months_between(d1, d2) - Returns no of months between d1, d2.

3)* Next_day(date, 'char'/NUM') - Returns date of coming day.

4)* Last_day(date) - Returns last date of that particular month.

Rounding, Truncating on Dates:

Round(date, 'dd') - check for Nearest time of that day (Before PM (or) After PM)

Round(date, 'day') - takes to the Nearest Sunday

Round(date, 'month') - Check for day: <15 (or) >15 → <15 takes to the 1st of same months; >15 takes to the 1st of Next Month

Round(date, 'year') - checks for months <6 (or) >6: <6 takes to the 1st day, 1st month of same year; >6 " 1st day, 1st month of Next year.

Trunc(date, 'day'): takes back to the last Sunday.

Trunc(date, 'month'): takes back to the 1st of same Month.

Trunc(date, 'year'): " " " 1st day, 1st Month of same year.

Conversion functions: 1) to_date() : used to convert character to date formate

to_date(char, 'formate') formate is optional. However it will display as per the supplied formate.

2) to_char('date', 'format'); Converts date to character formate

to_char('13-may', 'dd-mon'), X

↳ First Convert using to_date then to_char ✓.

3) to_number('char'); Converts 'Number char's to Number.

to_number('123') ✓

to_number('as12') X

DD-MON-YY

DD	MON	YY	HH:MI:SS AM	CC	Q	WW	SP
dy	MONTH	YYYY	hh24	↓ Century		↓ Week of year	
day	MM	year	hh12 (default)		↑ Quarter		↑ Spell out

* display the employes who joined 2 years ago.

Spell Numeric:

Select to_char(to_date('12345', 'JSP'), 'JSP') from dual;

Julian spell out. MAX = 1 Million (10L).

2) Multiple Row Fn's (or) Group Fn's: this fn's can Appear in Select list, having clause and order by clause only *.

group_function_name(distinct / ALL columns)

* distinct makes the fn to consider Non duplicate values.
* ALL " " " All values.
* All group fn except count(*) will ignore Null values* except count(*).
* When group fn is declared in select list remaining columns can also be declared but they should be placed in group by clause.

1) Sum (distinct/All)
2) MAX (distinct/All)
3) MIN (distinct/All)
4) Avg (distinct/All)
5) Count (*/distinct/All) ↳ Includes Null
6) Stddev (distinct/All)
7) Variance (distinct/All)

Pg: 77 (Work out)

3) General fn's:
1) NVL
2) Decode
3) greatest(m1,m2...)
4) least(m1,m2...) Vid.
5) Vsize(column)
6) User

> select user, vid from dual;

1) NVL (column, data-type matching value);

2) ~~decode (column)~~, dcode: used to check for multiple condition (like If else).

select deptno, decode (deptno, 10, sal*1.5, 20, sal*2.5, sal-1000) from emp;
(If, then, If, then, else (last one).)

Specifying more than one value in else results Null value.
* Data type should match in then, else.

Case statmt: case stmt supports comparision operators. (decode doesn't)

Select sal, job, case when sal >=800 AND JOB='CLERK' THEN 'Kiran'
when sal = 5000 Then 'First'
when deptno = 10 Then 'Ten'
else 'Last'
end case from emp;

Then Part can be either varchar (or) Numeric but all then's should be in synch.

Constraints: (conditions).

Data Integrity: It is a state in which all the data values stored in the database are correct

Entity (Row)

Data Integrity

- Entity Integrity → Row as a unique entity (PK, FK relation)
- Domain Integrity → Entries for a given column (Not Null, check constraint)
- Referential Integ → Foreign key relation
- User defined Integrity → Data base Triggers.

Constraints in Oracle:

2 ways of declaring constraints
1. Column level (In line Style)
2. Table level (Out of line Style)

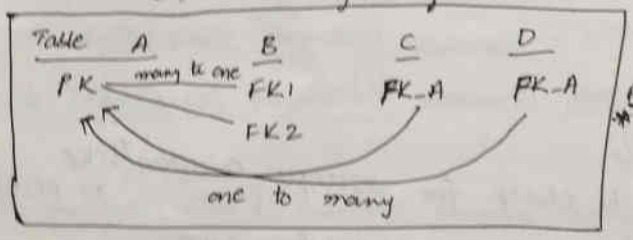
1. Primary key (Unique + Not Null + Index)
2. Foreign key (Referential Key)
3. Unique
4. Not Null
5. Check
6. default

Unique = Unique + Null + Index

Note: When a table is created based on other table all constraints will be copied to new table except Not Null property.
create tab B As Select * from A; B have only Not Null remaining constraints dont be copied. (View will have all constr)

Primary key: The values of P.K Columns should be Unique, Not Null and the key Columns will be Indexed.

Foreign Key: forming a relation with P.K of other Table. one Table can have more than one foreign keys. And this F.K can referenced to one P.K or diff P.K of diff Tables.



Foreign key can have: Null values
duplicate values

*on delete cascade: When trying to remove records from 'Parent table' automatically records of the 'dependent table' will be deleted.

> create table b (no number(10) references a(no) on delete cascade);
PK of 'A'

Removing rows of 'A' will Automatically delete 'B' Rows.

delete (or) drop will throw error depending child records exists.

Cascade Constraints: Allows to remove Parent table even child records exists

drop table A cascade constraints;

table is lost in depending table B, C. It will destroy relationship b/w child tables. But data is not removed from depending Tables.

Unique: Columns specifies as unique, the values should be unique.

> create table A (no number(10) unique); ⇒ Nulls Allowed

Not Null: Null values not allowed for specified columns with Not Null. Default will be Null property.

> create table A (no number(10) Not Null);

Default: when ever the value not specified or default passed as value then default value will be taken.

> create table A (No number(10) default 100);
A (Name varchar2(10) default 'abc');

*Null value passed as value then overides the default value.

More than one Constraint for Single Column: At a time declaring

> create table A (No Number(10) references A on delete cascade unique NotNull);

Check: checks the condition before taking the values of check column.

> create table A (No Number(10) check (No > 10 and No < 100));

Composite Primary Keys, Composite foreign keys: (MAX 32 Columns)

Defining more than one column in Table Level is called CPK (or) CFK.

> create table A (no number(10), no1 number(10), no2 number(10) unique Not Null,
Constraint A-PK Primary Key (no, no1));

> creat table B (no number(20), no1 Number(20), Loc varchar2(20) default 'HYD',
Constraint B-FK foreign Key (No, No1) references A on
delete cascade);

Note: A Composite F.K can be declared only with the Composite Primary Key. Ex: In above ex Table A C.P.K has 2 columns so C.FK should also declare with 2 columns referenced to that table.

Self reference Key: A table contains foreign key referenced to the same table Primary Key.

> create table A (no number(10), no1 Number(20),
Constraint A-PK Primary Key (NO), (make sure commas)*
Constraint A-FK foreign Key (NO1) references A on delete cascade);

Table Level declaration

Adding Constraints: With 'Alter' (PK, FK, Unique)

> Alter table tab_name add Constraint const_name Primary key (list);
> " " " Unique (list)
> " " foreign key (list) references tab_name on delete cascade;

Ex: Alter table A add constraint A_FK foreign key (NO) references B on delete cascade;

Adding properties (Not NULL, default): * (for removig do reverse).

For existing Table: ① desc tab_name
② Take the size of the column to be modified.

⑧ Alter table tab_name modify colu_name datatype default 'value' Not Null;

Ex: desc A
NO Number (7, 2)

>Alter table A modify No (7,2) default 20 Not Null;*

Enabling, disabling, droping Constraints:

> Alter table A disable/enable/drop Constraint const_name;
(a_pk, a_fk)

System Tables (Constraints Existing):

1) User_Constraints. (Constraint_name, Constraint_type are columns of that Table)
2) User_Cons_Columns.

User_Constraints: holds the complete details of Constrains defined on Tab names.

> Select * from User_Constraints where table_name = 'TAB-Name';

> Select constraint_name, constraint_type from User_Constraints where Table_Name = 'EMP';

User_Cons_Columns: only brief information about the constraints on table column.

> Select * from User_Cons_Columns where Table_name = 'EMP'; (Capitals).

JOINS: Joining the rows of two (or) more tables, horizontally. Based on the condition. Multiple table should be placed in 'from' clause (views, materialized views)

1) Inner Join (Natural Join, Equi Join).
2) Left outer Join
3) Right outer Join
4) FULL outer Join
5) Self Join
6) Cartesian Join.
7) Non Equi Join

Note: Join b/w Number, Varchar will work. Join b/w Null values will be ignored.

* If there are 'N' no of Tables 'N-1' Joines are required.

T1 (Join) T2 ① — 3 Tables so, 2 Joins.
↓
Result (Join) T3 ②
↓
Final Join Result ③.

T1	T2
10	10
20	20
30	40

Inner Join: $(T1 \cap T2) = \{10, 20\}$

Left outer Join: $T1 \cap (T1 \cup T2) = \begin{Bmatrix}10\\20\\30\end{Bmatrix} \cap \left[\begin{Bmatrix}10\\20\\30\end{Bmatrix} \cup \begin{Bmatrix}10\\20\\40\end{Bmatrix}\right] = \{10, 20, 30\}$
$\Leftrightarrow \{10, 20, 30, 40\}$

Right outer Join: $T2 \cap (T1 \cup T2)$: $\begin{Bmatrix}10\\20\\40\end{Bmatrix} \cap \left[\begin{Bmatrix}10\\20\\30\end{Bmatrix} \cup \begin{Bmatrix}10\\20\\40\end{Bmatrix}\right] = \{10, 20, 40\}$.
$\Leftrightarrow \{10, 20, 30, 40\}$

Full outer Join: $(T1 \cup T2)$: $\begin{Bmatrix}10\\20\\30\end{Bmatrix} \cup \begin{Bmatrix}10\\20\\40\end{Bmatrix} = \{10, 20, 30, 40\}$.

Joining 3 Tables:

① Select * from A Inner Join B ON (A.Col = B.Col) Inner Join C ON (A.Col = C.Col) ✓

② Select * from emp ~~Natural~~ Join Dept Using (deptno);
Select * from emp Natural Join dept. Join will find the common column name.

Using clause: Using clause will take common column b/w the 2 tables And the name should be same else throws error.

③ Select * from A, B, C where (A.Col = B.Col) AND (A.Col = C.Col);

Cartesian Join: Rows of one table will be multiplied with all the rows of other Table.

T1	T2	T1 × T2
4	5	20 rows.

Self Join: Same Table Appears TWICE in the from clause with corresponding Alice Names

Non Equi Join: It is a Join condition when no column in one Table corresponds directly to a column in the other Table.

> Select E.ename, E.Sal, S.grade from Emp E, Salgrade S where E.Sal between S.losal and S.hisal;

Self reference:*

>Select m.ename Manager, e.ename from emp e, emp m where m.mgr = e.empno;

emp e — ename

mgr m — ename

SET OPERATORS: 1) Used to combine the columns of Select Stmt's based up on the operator. Here the columns are Joined Vertically.

2) The Select Stmts that are involved in Joining should have same no. of columns and corresponding data types must be matched.

3) Max 32 queries can be Joined using Set operators.

Compound Queries: Sql Stmts containing Set operators are called Compound Queries.

Component Query: Each Sql Stmt involved in Compound Query.

4) The two Select Stmt cannot have the order by clause the final result of the entire Set operation can be ordered.

$A = \{1, 2, 2, 3, 4, 5\}_5$ $B = \{1, 2, 3, 3, 4\}_5$

① Union All: Combines elements of both sets including duplicates (5+5 = 10).

$A \underset{All}{\cup} B = \{1, 1, 2, 2, 2, 3, 3, 3, 4, 4\}_{10}$

② Union: Combines elements of both records and removes duplicate values.

$A \cup B = \{1, 2, 3, 4, 5\}$

③ Intersect: Combines elements of both records which are common to both sets and duplicate common values will be removed. $A \cap B \{1, 2, 3, 4\}$

* No duplicates are allowed in the o/p set except Union All.

④ Minus: Returns unique values of 'A' that are removed from B. Duplicate values of set 'A' not Allowed.

A
1 x
1 x
2 x
2 x
3
4 x
5 x
3

B
1
2
2
5
4
6

A − B = {3} (unique value w.r.t B')

q_1 union q_2 Intersection q_3 Minus q_4 Union q_5

R_1 Intersection q_3 → R_2 Minus q_4 → R_3 Union q_5

Executing from Left to Right one by one.

NOTE: When taking multiple columns in the queries. The queries out put will be with distinct values of the set operator.

q_1: A	X
1	aa
1	bb
2	cc
3	dd

q_2: B	Y
1	aa
2	bb
2	bb
3	ee

$q_1 \cup q_2$: AB U XY — 2 Columns.

A	X
1	aa
2	bb
2	cc
3	dd → Extra value taking corresponding value
3	ee

Here: Actually 'A' has to display 1, 2, 3 But

A ∪ B
1
2
3

A ∪ Y
aa
bb
cc
dd
ee

Makes the union operation on both columns with possible combinations and removes duplicates. (When taking more than one column in select list).

A	
N	C
1	aa
1	bb
2	cc
3	dd

B	
M	d
1	aa
2	bb
3	ee

> Select N, C from A
> Union
> Select M, d from B;

O/P: A ∪ B

First make union on N, M → N

Then take possible combinations of C, d → C

N	C
1	aa (check aa with 1 or any thing else)
2	bb (check bb with 2 and found bb with 1 aswell)
3	dd Similarly rest
1	bb
2	cc
3	ee

VIEWS: A view is a data base object which is holding a select stmt in it.

- View is stored in the Logical Memory. (Stored in 'User_Views')

> create view view-name AS select * from Table_name; (Same as copying a Table).

* DML Performed on view will be reflected on Table and vice versa.

* It is used for sharing selected rows & columns with other users. & it will improve performance while oranipulating or retreiving data thro views.

Sharing View: grant all on V1 to User1

Permissions: insert, update, delete | select
DML , DQL

Droping a view: drop view V1; (only view is destroyed no reflection on Table).

To see Existing views: select * from user_views; | select * from Tab; (Tables + Views)

'with check option' clause: Used to check for condition in the where clause while inserting the values into the Table.

Ex: ① create or Replace view V1 AS select * from emp where deptno = 10;

> insert into V1 values (1100, ..., 20); (With out check option, it will be inserted into emp Table). ↳ Not allowed

error - view with check option - where clause violation ⊛.

② create or Replace view V1 AS select * from emp where sal > 1500 with check option;

> insert into V1 values (1122, ... 1000, 1000, 10);
sal
↳ error (While clause violation).

View Based on view:

> create view V2 AS select * from V1;

→ DML performed on V2 will be reflected in V1 and V1 holding Table.
DML " " Table " " " V1 and V2 views.

Read only option:

> create view V1 AS select * from emp with Read only; (No DMLs Allowed on view).

NOTE: Altering the table will lead to its corresponding views destroyed i.e view doesn't open saying error (view with errors). DDL on Tables → views Invalid.*
*Alter add columns doesn't reflect but existing columns droped throws error.

* Alter Not allowed on views(*).

Read only views: views based on Arithematic Expressions.

> create or replace view V1 AS select sal S, sal*10 A, sal*20 B from emp;
* View will be created with Alias Names.
* with out the alias names, a view cannot be created with expressions.
* once a view created with Alias Names, we cannot perform 'insert', update, delete operation on Alias column in to view as the view contains virtual columns (select stmt expression with Alias Name)
Insert, update, delete cannot be performed on virtual columns (i.e Alias Names because it contain expressions).
* we can make update, delete on Non expression column of view w.r.t Table. (*)

No Insert, Update, delete performed on A (Sal*10), B (Sal*20) virtual columns with expressions. But update, delete performed on S (Sal) it doesn't have expression.

View based on Aggregate functions:

> create view V1 AS select max(sal), min(sal) from emp group by sal;

Adv: Used for reporting purpose improves performance while retreiving data than way views.

View will Support Constraints Automatically:

> create or replace view V1 AS select ename, Job from emp;
- error as empno in emp when, trying to insert values in view V1 (*).
- A view has been created based on a table 'A' when altering the structure of table 'A' will reflect on views. throws an error.

Force:* It allows to create a view with out a table that is not in the Data Base.

> create or replace force view AS select * from XXYY;
↳ Table that doesn't exist in the DB.
- view created with compilation errors.
> select * from V1; - error (view scott.V1 has errors).

Advantage: Used to register the name in the Data Base.

JOIN VIEWS:

Key Preserved Table: The table whose Key Column is not duplicated in "view result" is known As K.P.T.

> create view V1 AS select e.empno, e.name, d.dept from EMP E, DEPT D;

> Select * from V1;

K.P.T (unique values) empno	ename	(duplicates) d.deptno
7900		10
⋮		10
		20
		20
12 rows		30
		30

Here in this example empno has unique values in the o/p so it will be Key Preserved Table.
But deptno has duplicated repeated.

T1 (parent) — Column1 (PK) ⋮

T2 (child) — PK → K.P Column ⋮ FK (T1)

The primary key, (or) unique key of child table will be Key Preserved Table. And the o/p of the view will have unique values of that column.

In Join view emp.deptno, dept.deptno Not Allowed only either of one column allowed
Because Join View doesn't allow duplicate columns.

1) *In Join view only one Table can be manipulated i.e Key Preserved Table.

2) *Both the Tables of View are allowed only through view using "Instead of Triggers" in PL/SQL.

update, delete operations performed on Key preserved Table Columns of view will reflect in the K.P.T.
But operations performed on Non K.P.T Columns doesn't take an effect on its corresponding Table.
Insert Supported with K.P.T
Not supported with Non K.P.T.

Materialized views: It is a static view (i.e holding data).
- It will not support DML on it.
- DML on Table not reflected on its view.
* To create it "create materialized view" permission is required from Admin.
- Used to maintain historic data &, data analysis & Reporting purpose

System: Grant create materialized view to Slott;

Even the Table deleted the view entries with data*

>create materialized view V1 AS select * from emp;
> drop table emp;
> Select * from V1; data exists as of emp (Ily to copy table*).

Synonym: Used to hide the original name and owner (user a/c) of the Table. When
It resides in "User_Synonyms".

* Provides Security by hiding Identity of the Component.
* create or Replace Synonym S ~~As Select * from emp;~~ for X;(

DML on Synony reflected in Table
" " Table reflected in Synonym
Ily view.
Copied table doesn't reflect the DML on main Table.

Droping A Table no longer exists of Synonym
Droping A Synonym doesn't reflect on Table
*Ily view.

>create Synonym S for emp; (we can make selected columns as view).

*diff b/w view & Synonym: A view can ~~Share~~ created selected rows & columns with other users.
Synonym makes the entire copy of the table & supports to share entire object with other user.

Sequence: Used to generate sequence of numbers automatically.
It is a data base object stored in 'user_sequences'.
It uses 2 Pseudo columns
(1) nextval
(2) currval

Syntax: create sequence seq_name
increment by ±N (-1)
Start with N (5)
MAXvalue M (5)
MINvalue P (-5)
cache integer/No cache (2)
cycle / No cycle (Yes)
order / No order (Yes)

default values
Increment by will be +ve
Start with 1
sequence is No cycle
cache will be 20
sequence is order sequence.

O/P: 5, 4, 3, 2, 1, 0, -1, -2, -3, -4, -5 w/ cycle
5, 4 .... -5.

Alter Seq NO:

> Alter sequence seq_name
increment by N
Maxvalue/NOMAXvalue (No space)
Minvalue/NoMINvalue (No space)
cycle/Nocycle
cache N/Nocache

Note: No possibility to change Start with. only possible is drop.

INdex: (User_Indexes): It is a pointer locates the physical Address of data Memory Location.
- It will improve the performance of oracle while retriving or Manipulating data from Table.
- It is automatically activated when Index column is used in "where clause". When Table droped corresponding Indexes also be droped.

Types of Indexes:
1) Normal Index (or) Simple Index.
2) Composite Index
3) Unique Index
4) Function Based Index.
5) Bit map Index.

1) Normal Index (or) Simple Index: created upon a Table by considering only one Column.

> create 'Index' Idx1 on emp (deptno); } When ever 'deptno' used in where clause retrieval will be very speed.

2) Composite Index: Index created on multiple columns. MAX 32 Columns Allowed.

> create 'Index' Idx1 on emp (Job, deptno, empno);

3) Unique Index: Indicates the entry of the values to the Index Column unique should be unique. i.e No duplicates allowed.

> create 'unique Index' Idx1 on emp (empno);

If the column is having unique property, it cannot be Indexed. B'Coz ~~it is already~~ unique itself in Index.

Note: Both unique & Bit map cannot be Indexed at a time

4) Function Based Index: Fn based Indexes are designed to improve query performance when the function is used in 'Where clause'.

> create Index Id1 on emp (upper(Job));

Select * from emp where upper(Job) = 'MANAGER';

5) Bit map Index: It Specifies an Index has to be created with a Bit map for each 'distinct Key' in the Table.

- Bit map Index stores the 'RowIds' Associated with distinct key values of the column as a Bitmap ('1' and '0's).

> Sttet create bit map Index Id1 on emp (deptno);

Deptno	Start Row Id	End RowId	Bit pattern
10	----AAA	----AAN	1010 01
20	----AAA	----AAN	10 01
30	----AAA	----AAN	1001 10

> create bitmap Index Id1 on emp (mgr);

↓ no space

Clusters (user_clusters): It holds the common column shared by 2 or more tables.

- It will improve the performance while retreiving or manipulating data from parent, child Tables.

Note: It cannot be applied to the existing Tables. It has to be created before creating Tables.

1. > create cluster c1 (deptno number(10));
2. > create table dept (deptno number(10) primary key, dname varchar2(20), loc varchar2(20)) cluster c1 (deptno);
3. > create table emp (empno number(10) primary key, deptno number(10) references emp on del cascade) cluster c1 (deptno);
4. > create index id1 on cluster c1;

droping cluster: drop cluster c1; (drops only cluster)

drop cluster c1 including Tables; (cluster, parent, child will be droped).

Roles: Used to share the group of 'Privileges' of multiple objects with other users easily. Defined by "DBA" only.

- It holds collection of permissions to be shared and stored in "User_roles".

> create role hr; create role hr1; create role hr2;

> grant all on emp to hr;

> grant all on dept to hr; > grant select, insert on Salgrade to hr;

> grant hr to user1; > grant hr, hr1 to hr2;

> revoke select on Salgrade from hr;

droping role: drop role hr;

Locks:

Joins:

TABLE A	TABLE B
1 } (3)	1 (2)
1	1
1	2 (2)
2 } (2)	3 (2)
2	2
3 → (1)	3
4 → (1)	10 (1)
5 → (1)	11 (1)
	12 (1)

Inner Join: (Matching values of both Tables)

1, 2, 3,

A	B	O/P values
1 (3)	1 (2)	3 × 2 = 6
2 (2)	2 (2)	2 × 2 = 4
3 (1)	3 (2)	1 × 2 = 2
		Total - 12 (Rows)

Left outer Join: A B (Matching & Non matching of Left Table & Matching of Right Table)

A	B
1 (3)	1 (2)
2 (2)	2 (2)
3 (1)	3 (2)
Non matching 4 (1)	
5 (1)	

Total Rows =
(1) 3 × 2 = 6
(2) 2 × 2 = 4
(3) 1 × 2 = 2
(4) 1 = 1
(5) 1 = 1
(14) Rows.

Right Join:

1 (3)	1 (2)	= 6
2 (2)	2 (2)	= 4
3 (1)	3 (2)	= 2
	10 (1)	= 1
	11 (1)	= 1
	12 (1)	= 1

15 Rows.

Full Join:

A	B	
1 (3)	1 (3)	= 6
2 (2)	2 (2)	= 4
3 (1)	3 (2)	= 2
4 (1)	10 (1)	= 1 → (4)
5 (1)	11 (1)	= 1 → (5)
	12 (1)	= 2 → (10)
	10 (1)	= 1 → (11)
		= 1 → (12

Common

Non matching

18 Rows.

Sum of Common Rows + Non matching elements of both Tables.

3 Tables:

A	B	C
1 (3)	1(2)	1(3)
2 (2)	2(1)	2(2)
3 (1)	3(2)	
4 (1)	10(2)	
	11(1)	

12(1)

Inner Join: Matching of A, B, C (1, 2)

1's → 3 × 2 × 3 = 18

2's → 2 × 1 × 2 = 4

Total rows = 22

Left Join: Left Join of A, B Result + C.

1 → 3 × 2 = 6

2 → 2 × 1 = 2

3 → 1 × 2 = 2

4 → 1 × 1 = 1

Left Result A/B = 11

(1) 6 × 3 = 18

(2) 2 × 2 = 4

(3) 2(3) = 2

(4) 1(4) = 1

25

Note: While joining 3 tables: A, B, C, ...

A Join B ON (A.C1 = B.C1) Join C ON (A.C1 = C.C1)

While joining first table column of table A and B will become one row, and that row would be checked against 'C'.

System Tables:

DICT: Data dictionary table. Holds the list of System Tables in Data Base (which holds the meta data of all tables). DICT acts as Repository.
- Indicates all the tables of the data base. Like

dba_tables, all_tables, user_tables
dba_tab_columns, all_tab_columns, user_tab_columns.

> Select * from dict where table_name Like '%.EMP%'; Like operator belongs list of names on EMP in repository.

1) TAB: It holds system default tables, + User created tables in the user login (Schema) (Tables, views, Synonyms) → select * from tab;

2) CAT: is Synonym for "user_catalog" which will list that login (Tables, views, Synonyms and Sequences) → select * from cat;

TAB: It holds the list of Tables available in User login.

User_Tables: holds the detailed information about the tables.

User_tab_columns: holds the brief information about columns defined in Tables.

1) Select * from user_tables where table_name = 'EMP';
2) Select * from user_tab_columns where table_name = 'EMP';

All_tab_privs_made: holds the list of permissions given to other users.

All_tab_privs_recd: holds the list of permissions received from other users.

Above two: Select * from All_tab_privs_made;

All_users: System table which holds the list of users in the server.
Select * from All_users;

TABLE: (user_tables, user_tab_columns, TAB)

Views: (TAB, CAT, user_views)

Synonyms: (user_synonyms, TAB, CAT)

Sequence: (user_sequences, CAT)

Index: (user_indexes)

Clusters: (user_clusters)

Roles: (user_roles)

Admin

> create user user_name identified by password;

> grant connect, resource to user1;

> grant connect, resource to user1 identified by user1;

> Keep user user1 console ; console tabs later with other users.

Locking/unlocking A/c:

Alter user user_name account lock/unlock;

Alter ~~user~~ Alter user scott account lock;

Constraints: (System Tables)

User_Constraints: Holds the complete details of constraints defined on the table columns.

User_Cons_Columns: holds brief info about the constraints applied on the columns.

> desc user_constraints

> Select * from user_constraints where table_name = 'EMP';

P(PK), C(CHK/NN), U(Unique), R(Referential)

Db objects: Table, View, Synonym, Sequence, Index, Cluster, Role.

Copying Table:

```
create table B AS select * from A;
create view v1 AS select * from emp where deptno = 10;
create Synonym s1 for emp;
create Index id1 on emp ( ename, mgr);
create unique Index id1 on emp ( empno);
create bitmap Index id1 on emp (Job);
create cluster c1 (deptno number(10);
create table emp ( , , , ) cluster c1 (deptno);
create table dept ( , , , ) cluster c1 (deptno);
create role hr;
```