SQL Correlated Subqueries

Correlated subqueries are used to for row-by-row processing. each subquery is executed once for every row of the outer query.

— Basic SQL.

① Introduction.

② substitution variable.

③ SQL functions.

① single row function

ⓘ character function.

ⓘⓘ general function.

ⓘⓘⓘ Number function.

ⓘⓥ Conversion function.

ⓥ Date.

② multiple row function | group Function

aggregate function | analytical function

statastical function.

① Numeric function.

④ JOIN

① equijoin | Innerjoin | simple join.

i) Natural join.

ii) using clause.

iii) ON clause.

iv) self join.

② nonequijoin

③ outer join

i) Left-outer join

ii) Right outer join

iii) Full outer join

④ cross join | cartesian product.

⑤ subquery

① single row subquery.

② multiple row subquery.

⑥ set operators

① union ② union All ③ Intersect ④ MINUS

Scanned with CamScanner

** Basic SQL **

SQL := stands for - Structured Query Language.

Data := Collection of Row Information.

Database := logical & meaningfull information.

DBMS := Database Managment system.

software (Oracle, mysql, MS-SQL, Server, DB2, SAS, Ingress).

SDLC := Software developement life cycle.
is called as a software developement process.

Analysis Requirement gathering
↓
DFD-(front end) ← Design → ERD(Entity Relationship Diagram) (back end)
prototype.
↓
Build & documentation (Agreement)
↓
Implementation / coding (SQL coding)
↓
Testing
↓
product

life cycle of SDLC.

Scanned with CamScanner

RDBMS :- (Relational DataBase Managment system).

*** SQL command :-

To used to define data structure & schema objects

① DDL :- (Data Defination language).
create, Altee, Drop, Truncate, Rename, comment.

② DML :- (Data Manipulation language).
Insert, Update, Delete, merge, select.

③ TCL :- (Transaction control language).
commit, Rollback, save point.
commit - To saves the changes.
RollBack - To Rollback the changes (Transaction in database)

④ DCL :- (Data control language).
Grant, Revoke.

= Select :- Is used to retrive data from the database.
↓
Identifies the coloum to be displayed.

= from :- Identifies the table containing those column.

Scanned with CamScanner

* Capabilities of SQL select statement :=

There are three capabilities of SQL select statement.

① projection :=

Specifies column data will be retrive.

② selection :=

Specifies Row data will be retrive.

③ Joining :=

Join two or more Table.

The SQL Statement are not case-sensitive.

Q. what is Null value ?

A null is a value that is unavailable, unassigned, unknown.

The null is not same as a zero or blank space.

- zero is a value.

Q. Use of column Alias :=

- Rename a column heading.
- Is useful with calculation.

double quatation mark

select first_name "first name" from employees.

use As keyword or also use space for name & double quatation mark. (If spaces or special character)

It is a case-sensitive.

Scanned with CamScanner

Q. Use of concatenation operators :- (||)

- links column or character strings to other column.

- It represented by two vertical bar. (||).

e.g. select first_name || salary as fs from employees.

o/p. fs
Ganesh 20000
Amol 40000

* Literal character string :-

The date or character literal value must be enclosed by single quotation marks.

e.g. select first_name ||' join date is ' || Hire_date from employees;

o/p
Ganesh join date is 25/08/1998.

* Alternative quote (q) mark :-

use character
[ ], { }, ( ), or < >

use of q quotation mark is use Apostrophe mark in sentences.

Scanned with CamScanner

e.g.

Select last_name || q'[It's a minimum salary]' salary from employees;

q use of Distinct keyword :-

Remove duplication from the result set by using Distinct keyword.

q. Describe command :-

Describe command to display the structure of table.

Describe employees; &
Desc employees;

q. where clause -

filters data on row level (before grouping).

- where clause can be used ~~other~~ after than Select statement also.
- where clause applied to each & single row.
- where clause used before ~~by~~ group by clause.

e.g. select employee_id, first_name, last_name, department_id from employees
where department_id = 20;

Date & character string value are enclosed in single quotation mark.

- character value are case-sensitive.
- Date value format sensitive DD-mon-RR.

Scanned with CamScanner

e.g.

```
Select employee_id, first_name, last_name
from employees
where first_name = "Karen";
```

* **BETWEEN condition :=**

In Range operation, minimun value should be first then maximum value.

e.g. select * from employees
where salary Between 4000 and 120000;
min max

* **In condition :=**

To Test for values in specified set of values.

— It is a membership condition.

e.g. select * from employees
where employee_id IN(101, 102, 103, 105);

Note:- without using IN operator

```
Select * from employees
where employee_id = 100 or
      employee_id = 101 or
      employee_id = 105;
```

* In condition can be used with any datatype.

Scanned with CamScanner

* **LIKE condition :=**

To perform searches of valid search string values.

the use -

·%· - denotes zero or many characters

_ - denotes one character.

e.g. select * from employees
where first_name like '_e%';

* **NULL condition :=**

IS null condition test for nulls.

- The NULL condition includes the IS NULL condition and IS not NULL condition.

e.g.

Select * from employees
where commision_pct is not Null;

&

select * from employees
where commission_pct is Null;

* **AND operator :=**

And requires both condition to be True.

Comparision operator :- =, <, <=, >, >=, != / <>

BETWEEN ---- AND ----, IN (set)
Like, ~~IN~~ IS NULL.

Scanned with CamScanner

* Logical condition :-

operator	meaning.
AND	Returns True if both component condition is true.
OR	Returns true if either component condition is true.
NOT	Returns True if the following condition is False.

Select & column1, & column2, salary
from ~~employees~~ & table_name
where & expression
order by & column_name

- where condition
- order by clause
- column expression
- table names
- Entire select statement.

} - use substitution variable to supplement the following.

Scanned with CamScanner

~~when~~ use substitution variable to supplement the following -

- where conditions
- order by clause
- Column expressions
- Table names
- Entire select statement.

(9)

* Substitution Variables :- (Script).

There are two Types -

① & substitution variable :- Temporarily store values with single ampersand (&) and double ampersand (&&)

② && substitution variable :-

① & substitution variable :- use a variable prefixed with an ampersand (&) to prompt the user for a value.

② && substitution variable :- If you want to reuse variable value without prompting to user each time.

Define command :- create and assign a value to a variable.

undefine command :- use undefine command to remove a variable.

e.g.

```
Define emp_id = 105;

select * from employees
        where employee_id = &emp_id;

undefine emp_id;
```

Substitution variable

```
Select first_name, Salary from employees
        where department_id = &n;
```

Scanned with CamScanner

* VERIFY command :-

TO confirm the changes in the SQL statement.

* ORDER BY clause :-

Is used to sort the fetched data in either ascending order or Descending order to one or more column.

- Boot retrived rows with order by clause.

· Ascending (ASC) :- Ascending order.
· Descending (DESC) :- Descending order.

e.g. select * from employees ~~where~~ order by salary desc;

* dual :- Dual is System Define table.

** Execution of sql query **

① FROM -
② WHERE -
③ Group By -
③ SELECT -
④ HAVING -
⑤ order By -

↓

FROM
WHERE
GROUP BY
HAVING
SELECT
ORDER BY.

Scanned with CamScanner

Q. Difference Between DML or DDL command ?

DDL	DML
1. DDL stands for Data Defination language.	1. DML stands for Data manipulation language.
2. These are used to define data structure.	2. It is used to manipulate the existing database.
3. It is used to database structure or schema.	3. It is used to managing data within schema object.
4. It works on whole table	4. It works on one or more rows.
5. commands are - create, Alter, drop, truncate, Rename.	5. SELECT, Insert, update, delete, merge.
6. DDL is autocommit command.	6. DML is auto Rollback command.
7. ex. drop table table_name	7. select * from table_name.

Q. DBMS & RDBMS difference ?

DBMS	RDBMS
1. Normal DBMS	1. Relational DBMS.
2. can't show relation betn two or more tables.	2. can show relation betn two or more tables.

Scanned with CamScanner

3. less secure	3. More secure.
4. complicated	4. Remove the complexity.
5. can't used normalizatn rules	5. can used Normalizatn Rules.
6. can't used lock system	6. can used lock system.
7. Take more time for execution.	7. take less for execution.
8. No security of data.	8. there are multilevel of security.
9. Not support distributed database.	9. support distributed database.
10. DBMS applⁿ stored data as file.	10. RDBMS applⁿ store data in tabular form.

Scanned with CamScanner

* SQL Functions :-

- perform calculation on data.
- modify individual data item.
- manipulate output for group of row.
- convert column data types.

There are two types of SQL function :-

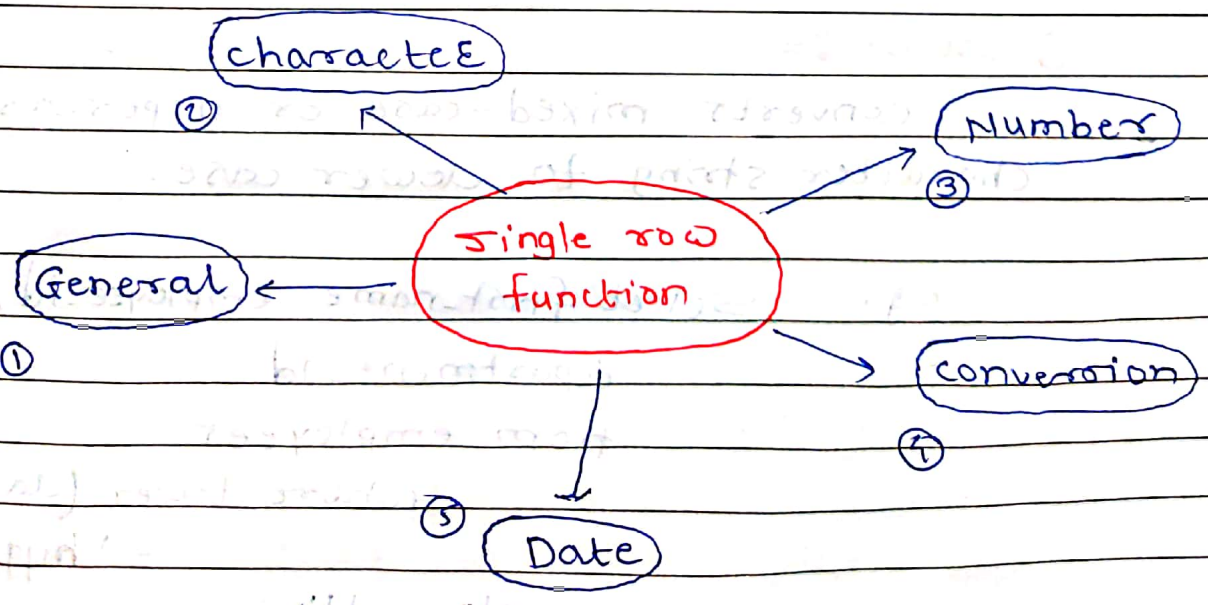
Function

- Single Row functions
  - Return one result per row.
- Multiple Row function
  - Return one result per set of rows.

#1) Single Row Function :-

Single row function operates only one row and return one result per row.

There are five types of single row function :-



Scanned with CamScanner

= ① Character function :=

The single row character function accept character dates as input & can return both character and numeric values.

There are Two Types –

Character function

(I) case-manipulation function	(II) Character manipulation function
- Lower	- concat
- Upper	- substr
- Initcap	- length
	- Instr
	- Lpad / Rpad
	- TRIM
	- REPLACE

I] case-manipulation Function :-

① Lower :=

converts mixed-case or uppercase ~~to~~ character string to lower case.

e.g.
```
Select first_name, employee_id, last_name,
       department_id
from employees
       where Lower (last_name)
             = 'higgins';
```

o/p – Higgins

Scanned with CamScanner

② UPPER :=

converts mixed-case or lowercase character string to uppercase.

e.g. UPPER('SQL Course')

o/p :- SQL COURCE.

③ Initcap :=

It converts first letter of each word to uppercase & remaining letters to lowercase.

e.g. Initcap('SQL course');

o/p :- Sql Course.

II] Character Manipulation function :-

① Concat :=

Joins values together (you are limit to using two parameters with CONCAT)

e.g. concat('Hello', 'world');

o/p :- Helloworld.

concat('Hi', 'Ganesh');

o/p :- Higanesh.

Scanned with CamScanner

② SUBSTR :=

Extracts a string of determined length.

e.g. substr ('Ganesh', 1, 3);

o/p:- Gan

③ LENGTH :=

Shows the length of string as a Numeric value.

e.g. length ('Ganesh Khedkar');

o/p:- 14

④ INSTR :=

Find the numeric position of a named character string.

eg. INSTR ('Helloworld', 'w');

o/p:- 6

⑤ LPAD :=

pads the character value right justified

eg. LPAD (salary, 10, '*');

o/p:- ****12000

Scanned with CamScanner

⑥ RPAD :=

pads the character value left justified.

eg. RPAD('salary', 10, '*');
O/P:- $4000 * * * * *

⑦ TRIM :=

Trims heading or trailing character from a character string.

remove the first & last letter of from a character string.

eg. TRIM ('G' from 'Ganesh')
O/P:- anesh.

⑧ REPLACE :=

It's a Trim character or Trim source.

replace first & last letter from a character string.

eg. Replace ('Jack and jUE', 'J', 'BL')

O/P:- Black and blue.

example :=

```
Select concat (first_name, last_name) as name,
subste (first_name, -2), instr(last_name, 'a'),
length(first_name), trim ('A' from first_name),
replace (first_name, 'Al', 'BL'), rpad(salary, 10, '*'),
lpad( subste (salary, -3), 10, '*') from
employees where
lower (first_name) = 'alexander';
```

Scanned with CamScanner

= ② Number Function :=

Number function accept numeric input & return numeric values

There are three types of Number function.

I] Round :=

check the nearest value.

The round function rounds the column expressions, or value to n decimal places.

The value is rounded to zero decimal.

e.g.

select round(99.999, 1) from dual;

o/p :- 100.000

II] TRUNC :=

Zero the value after the point.

The Trunc function truncates the column, expressions or value to n decimal places.

e.g. select trunc(99.999, -1),
trunc(55.55, -1),
trunc(6782, -2)
from dual;

III] MOD :=

mod value always return the Remainder.

- Returns remainder of division.
- The MOD function find the remainder of the first ~~expression~~ argument devide by the second argument.

Scanned with CamScanner

eg.

```
Select first_name, last_name, salary
        mod (salary, 5000) as mod
        from employees where first_name
                = 'Alexander';
```

= ③ Date Function :=

The Date function -

- The Database stores Dates in an internal numeric format -

century, currency, year, month, date, hours, minutes, and seconds.

- The defaut date display format :-

DD - MON - RR

e.g.

```
select first_name, last, Hire_date
        from employees where
        hire_date < '01-FEB-88';
```

- SysDate function : -

It is a system Define function Returns system Dates, current date OR database server Date & times.

e.g.

```
select sysdate from
        Dual;
```

* Function :- Date function

There are six types of a Date function...

Scanned with CamScanner

① Months-BETWEEN :-

- Number of month between two dates.

e.g. MONTH_BETWEEN ('20-01-2019', '21-02-2020');

② ADD-MONTHS :-

- ADD calender months to date

e.g. ADD_MONTHS ('20-01-2020',2);

③ NEXT_DAY :-

- Next day of the date specified.

e.g. NEXT_DAY ('01-Aug-2019', monday);

④ LAST_DAY :-

- Last Day of the month.

e.g. Last_Day ('01-Aug-2019');

⑤ ROUND Date :-

Round date.

जर month round date केले तर ~~month~~ Days devide houn जर 15 पेक्षा कमी असेल तर month ची starting ची date येईल.

जर year round केले तर year ~~det~~ months devide hoti 6 पेक्षा कमी ~~असेल~~ तर 1-1 नसेल तर 1-12-20

months च्या starting पासून
years च्या starting पासून

(6) TRUNC :-

Truncate Date

[year का starting पाएगा
month का starting पाएगा]

eg.

trunc('sysDate, 'month') o/p - 01-02-2020

trunc (sysdate, 'year')

o/p - 01-01-2020

DD-MON-RR -

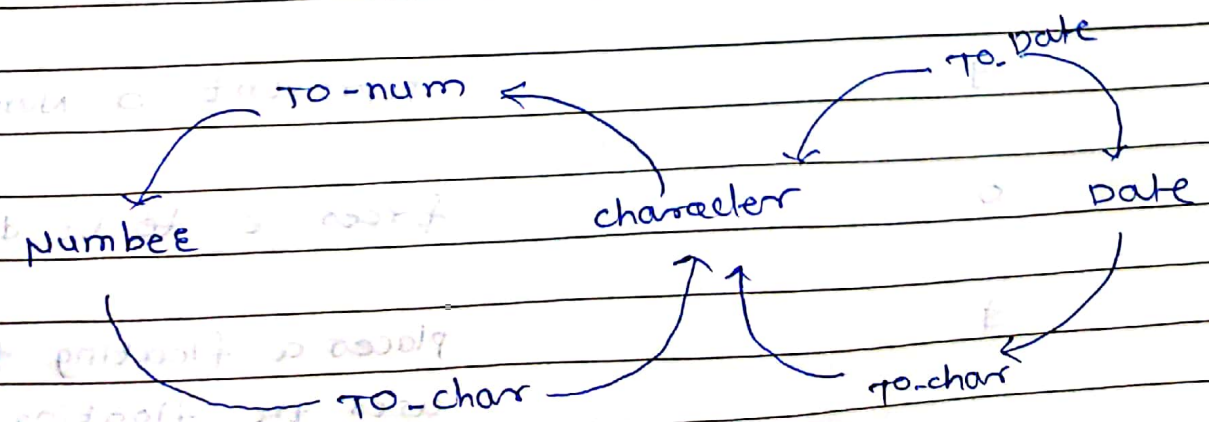
[past का feature की century के लिए use करे।

*(5) Implicity Data type Conversion :-

The oracle server can automatically convert the following.

FROM	TO
varchar2 or char	Number
varchar2 or char	Date
Number	varchar2
Date	varchar2

Implicit Data type conversion -



Scanned with CamScanner

① TO-char function with Dates :-

Syntax -

TO_char (date, 'format model')

- must be enclosed by single qoutation marks.
- is case sensitive.

e.g. select last_name,
TO_char (hire_date, 'fmDD month year') AS HIREDATE
from employees ;

fm element to remove padded blanks or suppress leading zeros.

e.g. (fmDD MONTH YYYY)
07 September 2015

o/p - 7 September 2015.

② TO-char function with Number :-

③ varchar2

Element	Result.
9	Represent a Number.
0	forces 0 to be displayed
$	places a floating $ sign
	uses the floating local current symbol.

Scanned with CamScanner

- print a decimal point.
, prints a comma as thousand indicator.

e.g. select TO_char (salary, '$99,999.00') as salary from employees where last_name = 'Ernst';

o/p :- 6000.00

By using L

₹ 6000.00.

e.g.

Examples :-

select to_number ('$7', '$9') from dual;

o/p → 7

dd s p t h

numeric spellout suffix

select hire_date from employes.
where TO_char (hire_date, 'month') = 'Aug';

Scanned with CamScanner

(4) General function :-

I works with Null values.

1) NVL
11) NVL2
111) NULLIF
1V) COALESCE

~~भाग पाउणे~~ function work with any data type & pertain to the use of Null values.

1) NVL function :-

Converts a Null value to an actual value.

Syntax -

NVL (expr1, expr2)

Datatypes that can be used are date, character and number.

e.g. NVL [commission_pct, 0)
NVL (Hire_date, '01-Aug-98')
NVL (job_id, 'No job yet');

11) NVL2 function :-

- The function examines first expr1"
- If the first expression is not-Null then the NVL2 function return the second expression.
- If the first expression is Null then NVL2 function return third expression.

Scanned with CamScanner

Syntax -

NVL2 (expr 1, expr 2, expr 3).

eg. select last_name, salary, commission_pct,
NVL2 (comm_pct, 'salary + comm_pct', salary) income
from employees.
where department_Id IN (50, 80);

III) NULLIF :-

Function compares two expression.
They are equal two expressions then return Null.
- They are not equal returns first expression.

Syntax -

NULLIF (expr1, expr 2)

eg. select first_name, length (first_name) "Expr1",
last_name, length (last_name) "Expr2",
NULLIF (Length (first_name), Length (last_name))
Result from employees;

IV) COALESCE :-

The function returns the first non-null expression in the list.
- The coalesce function can take multiple alternate values.
- The first expression is not-null then the coalesce function return the expression otherwise.

Syntax -

coalesce (expr1, expr2, ----, expr n)

Scanned with CamScanner

eg. select last_name,
coalesce (manager_id, commission_pct.
comm from employees
ORDER BY commission_pct;

- All values in the list are null, the coalesce function will return NULL.

* Conditional Expression :-
provide the use of IF-THEN ELSE logic within a SQL statement.

Two methods :-

1) case expression.
2) Decode expression.

① case expression :-

Syntax -
CASE expr when comparison_expr
Then return_expr,
when comparison_expr
Then
return_expr,
Else elseexpr.

e.g. select first_name, last_name, salary,
case Last_name when 'king' then
salary + 1000
when 'kochher' then salary + 2000
Else salary End " Revised salary "
from employees;

Scanned with CamScanner

② DECODE expression :-

Syntax

```
DECODE (col1 expr , search1 , result1 ,
                [ search1, result1 - salary]
                revised salary [ default ] )
                from employees ;
```

e.g.

```
select last_name , salary,
    DECODE( last_name, 'king', salary + 1000 ,
                'kochher', salary + 2000 -
                salary )
    Revised salary
    from employees;
```

* Nested function :-

```
select last_name ,
UPPER (concat (substr (last_name ,1,8) ,
                '_US')) "Nested fun"
from employees
where department_id = 60 ;
```

o/p

Last_name	Nested ~~of~~ function
Hunold	Hunold_US
Ernest	Ernest_US
Lorentz	Lorentz_US.

Scanned with CamScanner

Q. Difference Between case expression & Decode expression.

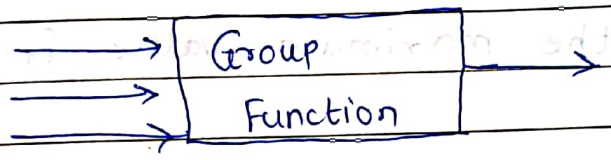
⟹

CASE	DECODE
- case is a expression.	- Decode is a function
- case executes faster than Decode	- Decode executes slower than case.
- case can work with PLSQL as well as SQL statement.	- Decode is used in SQL statement.
- Case can work with logical operator & searches subqueries.	- Decode can work with only scaler values.
- include data consistancy.	- not data consistancy.

Scanned with CamScanner

#2> Multiple Row Function | Group Function | Aggregate function Statastical Function / analytical Function :=

group functon operates on set of rows to return one result & one result per group



There are seven types of group function.

1. AVG
2. SUM
3. MIN
4. MAX
5. COUNT
6. STDDEV
7. VARINCE

1. AVG :=

Find the Avarage of given row or selected row.

e.g. select AVG(salary) from employees;

2. SUM :=

calculate sum of selected rows.

e.g. select sum(salary) from employees;

Scanned with CamScanner

③ MIN :=

Find the mininum value from selected row.

eg.

Select MIN (salary) from employees;

④ MAX :=

Find the maximum value from selected row.

eg.

select MAX (salary) from employees;

⑤ COUNT Function :=

i) COUNT (*) :-

- Return the number of rows in table. This includes Null values & duplication.

eg. select count (*) from employees;

ii) COUNT (expr) :-

- Return the number of rows with non-null values for expr.

eg. select COUNT (commission_pct)
from employees
where department_id = 80;

O/P commission_pct
3.

Scanned with CamScanner

(iii) ~~DISTINCT~~

COUNT (DISTINCT expr) :=

- Return the number of ~~rows~~ unique, non-values, that are in the column identified by expr.

```
Select count ( DISTINCT department_id )
        from employees;
```

o/p - 7.

NOTE :-

All group function ignore Null value in the column.

- The NVL function forces group function to include Null values.

e.g. Select Avg (commission_pct) from employees;
- 0.02325.

e.g. using NVL function :-

```
Select AVG (NVL (commission_pct, 0)) from
        employees;
```

Common example :-

```
Select AVG (Salary), max (salary), min (salary),
        sum (salary), count (*) from employees;
        where job_id like '%REP%';
```

Scanned with CamScanner

** Group By Clause :=

Group by clause to divide the row in a table into group's.

- Use the group function to return Summary Information for each group.

Syntax -

group by expression.

eg.

Select department_id, AVG (salary)
from employees
where group by department_id;

NOTE:-

- you can not use WHERE clause to Restrict groups.
- you use Having clause to restrict groups.
- you can not ~~yus~~ use group functions the where clause.

** HAVING clause :=

Restricting the group on basic of aggregate function.

- Having clause use to Restrict groups.

** ~~Number~~ Numeric Function :=

There are Three types of Numeric function -

1. ABS
2. power
3. SQRT.

Scanned with CamScanner

① ABS :-

Return absolute value of 'n'.

eg. select ABS(-15) from ~~emplo~~ dual;

o/p :- 15

② POWER :-

Returns 'm' raised to 'nth' power.

e.g. select power(5, 2) from dual;

o/p :- 25.

③ SQRT :-

Returns the square root of 'n'.

e.g. select SQRT(25) from dual;

o/p :- 5.

Q. Find out only repeated value.

```
select department_id
    from employees
        group by department_id
    having count(*) > 1;
```

Q. Delete Duplicate record from table.

```
Delet from employees e
    where employee_id != (select max(employee_id)
        from employees e1
    where (e.employee_id = e1.employee_id);
```

Scanned with CamScanner

Q. Difference between Having clause & where clause.

Having clause	Where clause.
1. Having clause filters data after group by has been performed so it's filters on "groups".	1. where clause filters data on row level (before grouping).
2. Having clause used only with select statement.	2. It can be used after than select statement also.
3. Having clause applied to summarized row.	3. Where clause applied to each & single row.
4. Having clause is used after group by clause.	4. where clause is used before group by clause.

Scanned with CamScanner

* JOIN *

To avoid a cartition product retrive data from one or more tables, join two or more table

JOIN include

- (A) equi join / Inner join / simple join
  - ① Natural join
  - ② ON clause
  - ③ using clause
  - ④ self join
- (B) Non-equi join
- (C) outer join
  - ① LEFT
  - ② RIGHT
  - ③ FULL.

(A) Inner join :-

At least having 1 common column name in both tables with same data types.

- It gives match data.

① Natural join :-

At least having 1 common column name in both tables with(of) same data types.

- keyword 'Natural join'.

e.g. select department_id, department_name, location_id, city
from departments
natural join location

Natural join clause is based on all columns in the two tables that have same name.

Scanned with CamScanner

② Using clause :-

The several column having same names but diff. data types, don't match Natural join applied using ~~to~~ the 'USING' Clause.

- The using clause to match only one column when more than one column matches.

- Natural join & USING clauses are manually same.

e.g. select employees.employee_id,
employees.last_name,
departments.location_id,
department_id
from employees join
departments
USING (department_id);

③ ON Clause :-

same name & same data-types

- The join condition is separated from other search condition.
  - ON clause make easy to understand code.

different column name but same data types

e.g. Select e.employee_id, e.last_name, e.department_id,
d.department_id, d.location_id
from employees e join
departments d

ON (e.employee_id = d.department_id);

Scanned with CamScanner

② self join :-

- Sometimes you need to join a table to itself. so you can use self join.

e.g.
```
select e.last_name emp,
       m.last_name mgr
       from employees e join
            manger m
       ON (e.manager_id = m.employee_id);
```

* multiple table join :- (3 Tables).

e.g.
```
select employee_id, city, department_id
       from employees e
       join departments d
       ON ~~(employee_id~~ (e.department_id, d.department_id)
       join location l
       ON (d.location_id = l.location_id);
```

③ Nonequi join :-

nonequijoin is a join table on condition base other than an equality operator.

- Never use "=" operator.

Retrive unmatch data from table.

e.g.
```
select e.last_name, e.salary, j.~~job~~ gradeslevel
from employees e join job_grades j
ON (e.salary Between j.lowest_sal AND
                     j.highest_sal.
```

Scanned with CamScanner

ⓒ OUTER JOIN :-

Returning Records with match and unmatch data.

① Left outer join :-

Retrive all rows in left table and retrive only match row in left right table.

e.g.
```
Select e.last_name, e.department_id,
d.department_name,
from employees Left outer join
departments d
ON (e.department_id = d.department_id);
```

② Right outer join :-

Retrive all rows in right table and retrive only match row in left Right table.

e.g.
```
Select e.last_name, e.department_id,
d.department_name,
from employeese Right outer join
departments d
ON (e.department_id = d.department_id);
```

③ Full outer join :-

retrive all match & unmatch data from both tables.

e.g.
```
select e.last_name, e.department_id, d.department_name
from employees e full outer join
departments d
ON (e.department_id = d.department_id)
```

Scanned with CamScanner

* Cartesian product :- / Cross join -

When a join condition is invalid or omitted.

- All rows in first table are joined to all rows in second table.
- To avoid a cartesian product,
- Always include a valid join condition.

e.g.

```
select last_name, department_name,
from employees cross join
departments;
```

Scanned with CamScanner

** Subquery **

* Subquery :-

A query within query is called subquery. Subquery use to query based on unknown values. subquery use to solve this problem

- subquery requires less time & if improve the performance.

Types of subquery :-

i) single row subquery.
ii) multiple row subquery.

① single row subquery :-

single row subquery that returns only one row from the inner SELECT statement.

* use of comparision operator in subquery -

=, <, >, <=, >=, <>/ !=

e.g.

select emp whose job-id is same as that of emp 141.

```
Select * from employees where emp-id = 141;
```

* Single row subquery -

```
Select last-name, salary from employees
where job-id = ( select job-id from employees
where employee-id = 141 );
```

Scanned with CamScanner

② multiple Row subquery :=

Return more than one row.

multiple-row-operator:-

i) IN :- Equal to any no. in list.

ii) ANY :- Any condition satisfy then given result.

iii) ALL:- must all condition should be satisfy.

eg.

IN

Select last_name, salary, d-id from employees where salary IN (select min(sal) from employees group by d-id);

ANY

e.g. select e-id, last_name, job-id, salary from employees where salary < ANY (select salary from employees where job-id = 'IT_prog') — AND job_id <> 'IT_prog';

ALL

e.g. Select emp_id, last_name, salary from employees where salary < ALL (select salary from employees where job_id = 'IT-prog') — AND job-id <> 'IT prog');

Scanned with CamScanner

Q Find out top five (5) max salary ?

```
select * from ( select salary from
        employees where
            order by salary desc )
        where rownum <=5 ;
```

Q. Find the third highest salary record from table.

```
Select distinct(salary) from employees e1
    where 3 =
        ( select count (salary) from
            employees e2
        where e1.salary <= e2.salary ) ;
```

Q. find top 2 salary from the record ?

```
Select salary from employees
    where rownum <=2 order by
        salary desc ;
```

Q find second highest salary from record ?

```
Select max( salary) from employeer
    where salary not in ( select max(sal)
        from employees ) ;
```

Scanned with CamScanner

** Set operators **

* Set operators :-

Set operators combine the result of two or more queries into on result.

① UNION operators :-

The union operator returns result from both queries after eliminating duplication. (avoid duplication).

e.g.

t_1	t_2	Result
1	2	1
2	4	2
4	6	4
5	8	5
8		6
		8

e.g.
```
select emp_id, job_id
    from employees
        union
select emp_id, job_id
    from job_History;
```

- Number of column & data ~~dat~~ types of the column should be same.

② UNION ALL :-

UNION ALL operator returns result from both queries, including duplication.

(including all with duplication).

Scanned with CamScanner

eg.

t_1	t_2		result
1	2		1
2	4		2
4	6	→ union All	2
5	8		4
8			4
			5
			6
			8
			8

e.g. select emp-id, job-id
from employees
UNION ALL
select emp-id, job-id
from Job-History;

③ Intersect operators :-

The Intersect operator returns rows that are common to both.

eg.

t_1	t_2	result
1	2	2
2	4	4
4	6	8
5	8	
8		

④ Minus operators :-

The MINUS operator returns rows that are present in first query & not in second query.

eg.

		result
1	2	1
2	4	5
4	6	
5	8	
8		

Scanned with CamScanner

Q. Difference between UNION & UNION ALL?

→

UNION	UNION ALL
i) UNION removes all duplication records.	i) UNION ALL allows duplication records.
ii) UNION is faster than UNION ALL.	ii) UNION ALL can be slower than UNION.
iii) UNION does not support BLOB or CLOB	iii) UNION ALL support BLOB.
iv) o/p is sorted in ascending order	iv) o/p is not sorted just gets appends.

BLOB :- Binary large object
It is datatype.

Q. Difference between REPLACE & Translate?

REPLACE	TRANSLATE
i) Replace entire string at a time.	i) Replace character one-to-one basis.
ii) return a string if no match found.	ii) return NULL if no match found.

Scanned with CamScanner

Q. Difference Between primary key & unique key ?

→

primary key	Unique key.
- It is a key that uniquely identify each record in a table.	- It is defined set of attributes within table that uniquely identify each record in a table.
- It can not allow Null value	- It can allow Null value.
- Each table can have at most one primary key.	- Each table can have multiple unique key.
- duplication not allowed	- duplication not allowed.

Q. Difference between set operator & JOIN ?

Set operator	JOIN
- combine multiple queries into single query.	- Join two or more table.
- merge the row.	- Join the column.
- Number of colum & datatypes of column, should be same.	- combine the Number of column based on condition.
- UNION, UNION ALL, Intersect, MINUS.	- Inner, outer, self, nonequijoin.

Scanned with CamScanner

Q. Difference between Rowid & Rownum ?

Rowid	Rownum.
– Rowid is permanant	– Rownum is not permanant, It is pseudo column.
– Rowid is used in indexing	– Rownum is used is select statement. where want a temparary.
eg) Find the distinct record	eg) find the last 3 salary.
) Delete Duplicate rows	) find the first 3 salary.

Scanned with CamScanner

** Data Manipulation language **

* Data Manipulation language :=

① Insert statement :=

Insert a new row containing values for each column, list value in the default order of the column in the table.

list the column in the insert value.

* Inserting row –

```
Insert into dept( dept_id, d_name, manger_id,
                  loc_id)
values (70, 'sales', 100, 7000);
```

* Inserting rows with Null values –

```
Insert into emp values (1, 'finance', Null, Null);
```

* Inserting special values :-

```
Insert into emp( emp_id, e_name, email,
                 hire_date)
values( 1, John, 'john12@gmail.com,
        sysDate);
```

* Creating Script :–

use & substitution in sql statement to promt for value.

' & ' is a placeholder for the variable value.

eg. Insert into emp (emp_id, emp_name, loc_id)
values (&emp_id, &emp_name,
&loc_id);

Scanned with CamScanner

* ② Truncate statement :=

Remove all rows from table, leaving the table empty & the table structure intact i.e. it present.

Important

* Database Transaction :=

Begin when the 1st DML statement is executed.

① COMMIT :- To save data permanant in (db). database.

② Rollback :- To undo the data changes i.e. 'ctrl + z'.

Note := DDL & TCL are autocommit

Advantages - (commit & Rollback).

- Ensure data consistancy.
- preview data changes before making permanant.
- Group logically realated operation.
- once you done commit then ~~do~~ cant do rollback.

Important

NOTE

* Schema :-

It is a collection of object, schema object are the logical structure that directly refer to the data in database.

Scanned with CamScanner

Q. Difference between Delete & Truncate ?

TRUNCATE	DELETE
- unconditionally remove all rows in table.	- selectively remove data from table.
- DDL command.	- DML command.
- This operation can't be rollback.	- This operation can be roll back.
- where clause can't use t	- where clause can be used with DELETE.

Q. Difference Between DROP, delete & Truncate.

DROP	Delete	Truncate.
- Delete structure + data	- Delete only data	- delete only data.
- where clause can not use	where clause can be use	where clause can not use.
- DDL command	DML command	DDL command.
- can'not be rollback	- can be rollback	- can not be rollback.
	- slower than truncate	- faster.

Scanned with CamScanner

Constraints is Aplying limitation on a column.

* * Constraints * *

* Constraints :=

constraints to prevent invalid data entry into tables.

constraints enforce rules on the data in a table whenever a row is inserted, updated or deleted from that table, The constraints must be satisfied for the operation to succed.

The constraints prevent the deletion of a table, if there are dependencies.

All constraints are stored in the data dictionary.

constraints used to specify rules on a data, in a table

constraints limit the type of data that can into table to column.

constraints improve the Accuracy & reliability of the data.

① column level constraints :-

e.g :- create table emp (eno number(10), ename varchar2(10), dno number(10) constraints dd_fk foreign key references dept (dno));

syntax column [constraints constraints_name] constraint_type,

② Table level constraints :-

column, - - - - -
[constraint constraint_name] constraint_type (column, - - -)

There are six types of constraints -

① NOT. NULL constraints.
② unique constraints.
③ primary key constraints.
④ ~~NOT NULL~~ Foreign key constraints.
⑤ check constraints.
⑥ Default constraints.

Scanned with CamScanner

① Not Null constraints :=

The not Null constraints ensure that the column contains no Null values.

— The column without Not Null contraints can contains null value by default.

— This is defined at the column level.

— Null value not allowed Here. (but duplication allowed)

② UNIQUE constraints :=

unique key requires that every value in column must be unique.

— i.e data in column not repeated. (duplication not allowed)

— unique key allow Null values. (Null value allowed)

③ PRIMARY key := constraints

uniquely indentify each record in a table is called as primary key.

— each table can have at most one primary key.

— can't allow Null value.

e.g. create table emp (id number(10) constraint id_pk primary key, sal number(10), name varchar2

— This is column level constraints.

e.g. create table emp (id number(10), sal number(10), name varchar2(20) not null, constraint pk_id primary key (id));

— this is table level constraints.

Scanned with CamScanner

④ FOREIGN KEY constraints :=

The foreign is defined in the child table and the table containing the referenced column in the parent table.

Foreign key - is used to define the colum in the child table at the table constraint level.

— References indentifies the table & column in the table.

= ON Delete cascade : =

It delete data from child & parent Table.

= ON DELETE SET NULL : =

It only delete data from child table & make it Null in child.

← converz foreign key value to NULL when the parent table is removed.

Note : = The without ON DELETE cascade & on Delete set null option, the row in the parent table can not be deleted if it is referenced in the child table.

e.g.

```
create table emp (id number (10), sal number (10),
name varchar2 (20) not Null, department_id
number (10) constraint fk_depart_id foreign key
references dept (deptno));
```

Scanned with CamScanner

5 CHECK constraints :=

check constraint define a condition that each row must satisfy.

Null value allowed, duplication allowed. by criteria or condition must be True.

e.g.

constraints pqR check (eid > 3);

&

constraint emp_salary_min check (salary > 0);

- The single column can have multiple check constraints.

6 Default constraint :=

provide a default value to column to insert.

e.g. CREATE TABLE EMPLOYEE
(id number(10), name varcher2(20),
Not null, city varcher2(200)
Default 'sandnes');

Scanned with CamScanner

(5) CHECK constraints :=

check constraint define a condition that each row must satisfy.

Null value allowed, duplication allowed. by criteria or condition must be True.

e.g.

constraints pgR check (eid > 3);

&

constraint emp_salary_min check (salary > 0);

- The single column can have multiple check constraints.

(6) Default constraint :=

provide a default value to column to insert.

e.g. CREATE TABLE EMPLOYEE
(id number(10), name varchar2(20),
Not null, city varchar2(200)
Default 'sandnes');

Scanned with CamScanner

* Date_time Data types :-

① Timpstamp :- Date with fractional second.

② Interval year to month :- stored as an interval of year & month.

③ Interval Day to second :- stored as an interval of days, ~~minutes~~, hours, minutes, & second.

e.g. create table emp(id number (10), sal number(10), Hiredate timestamp(7));

o/p.

id	sal	Hiredate.
1	10000	12 june 1998 .
2	150000	26 NOV 2005,

But when we retrive data i.e.

select hiredate from emp where id = 1;

o/p. 12-june-1998 12:00:00,000000 AM

* Timestamp with local time zone data Type.

Create table emp (order Date timpstamp with local time zone);

insert into emp values (15, jan, 2019 09:34:28 AM)

o/p 15-jan-2019 09:34:28 AM.

Scanned with CamScanner

** DATABASE OBJECT **

① Table :=
Basic unit of storage.
composed of rows.

② View := Logically represents subsets data from one or more table.

③ Sequence := Generate Numeric value.

④ Index := improve the performance of some queries.

⑤ Synonym := Gives alternative names to objects.

① View :=

A view is something of virtual table.
- It is logically represents subsets data from one or more table.
- A view derives it's data from the table on which it is based.
- These tables are called base table.

Types :=
① Simple view
② complex view.

Scanned with CamScanner

① Simple view :=

- Derives data from only one table.
- contains no function or group of data.
- can perform DML operation through the view.

② Complex view :=

- Derives data from many table.
- contains function or group of data.
- not always allow DML operation through the view.

Syntax / eg. Simple view —

```
create ~~to~~ view employees1
        as select employees_id, last_name, salary
           from employees.
           where department_id = 20;
```

Syntax / eg. complex view —

```
create or Replace view ~~as~~ ( did, cname, minsal,
        max sal, AVG sal )
        As select d.department_name,
           MIN (e.salary), max (e.salary),
           AVG (e.salary)
        from employees e JOIN departments
        ON (e.department_id = d.department_id)
           Group by d.department_name;
```

Scanned with CamScanner

* With check option :=

It is possible to perform referentiary integrity check through view.

— The with check option clause relates to insert or updates on the view only.

— define a condition that data being inserted or updated must satisfy or not.

e.g. create or replace view empval
As select * from employees
where department id = 20
with check option contraint empval_ck;

* WITH READ ONLY :—

To prevent DML operation on any row in the view.

eg. create or Replace view empval
(emp_no, emp_name, job_title)
As select emp_id, last_name, job_id
from employees
where department_id = 20
with read only;

* Removing view :—

Syntax — Drop view view_nam;

eg. drop view empval;

— remove the view defination from the database.

Scanned with CamScanner

② Sequence :=

- A sequence is a database object that creates integer values.
- Generate numeric values
- can automatically generate unique numbers.

advantages —

- generate unique numbers.
- sharable object
- replace application code.
- can be used to create primary key values.

Syntax :-

```
create sequence sequence_name
[increment By n]
[start with n]
[{ maxvalue n | non maxvalue }]
[{ minvalue n | nonminvalue }]
[cycle | nocycle]
[cache n | nocache]
```

e.g.

```
create sequence asequence
increment by 10
start with 2
max value 9999
no cache;
```

* Using Sequence :-

```
Insert into departments (department_id,
department_name, location_id)
values (sequence.Nextval, 'support', 2500);
```

Scanned with CamScanner

= Nextval :-

Return the next Available sequence.

- It returns a unique values every time it is refreshed, even for different users.

= Currval :-

Obtain the current sequence values.

Note - * current value (currval) is NULL

e.g. I have create sequence below

create sequence xyz;

- Create sequence automatically +1 and max value is unlimited;

* Disadvantages

- A rollback occurs
- The system crashes.
- The sequence is used in another table.

* modifying sequence :-

Alter sequence a sequence
Increment by 20
maxvalue 99999
nocache
nocycle;

* Drop sequence -

Syntax: DROP sequence sequence_name;

Scanned with CamScanner

index is used to Retrive data from database
index is a schema object

3 Indexes :=

— Improve the performance of some queries.

— can be used by oracle server to speed up retrival rows by using a pointer.

— Is used & maintained automatically by oracle server.

Syntax :-

```
create Index Index_name
on table (column [, column] ... );
```

eg. create Index emp_last_name_idx
on employees (last_name);

Types of INDEX —

There are two types of Index —

1. B-Tree index — (clustered index / unique index)
2. Bitmap index — (non-clustered index / non-uniq index)
   ↳ not primary key / not unique key

1 clustered Index | B_Tree Index :-

It is a Binary Tree information is then sorted seperately in desc.

— clustered index sort & store data row in table or view based on their key value.

— unique data.

2 Non-clustered Index | Bitmap Index | nonunique Index :-

— are created outside on the database table & contain a sorted list of references to the table itself.

- Repeation of data.
- This index that a user can create.

eg.

create unique index emp_id_idx
on employees (employee_id);

Q. can create an index more than one column in Same table.

. create index first_last_idx
on employees (first_name, last_name);

* **Draw backs of Index :-**

① Increase performance of query
. but decrease performance of data manipulation.

② can not apply index on table or column where value changes consistancy.

* **Drop index :-**

Drop index index_name;

Scanned with CamScanner

4 Synonyms :-

To give an alternative name to a table or database object.

Syntax :-

Create [public] synonyms synonyms_name for object.

public :- create a synonyms that is accessible to all users.

e.g.

create synonyms d-sum for dept_sum_vw;

↓

view name

↓

create synonym depts for departments;

drop synonyms :-

Drop synonym d-sum;

Synonym by default types — private.

Scanned with CamScanner

* Clustering :-

It's method of sorting tables that often used together (in joining) in one area on disk. As there realated table ~~sorted~~ stored in one area of disk,

performance of joining will improve.

- In order to place tables in cluster table have to a common column.

Def^n

- A cluster is created with name & key. A key is the column, which must be present in each table that is placed in cluster the key is known as "cluster key".

- create cluster with the help of two tables "courses" & "Batches"

e.g. create cluster course_Batch
(ccode varchar2(10));

* Creating cluster index - create index clust_index on cluster course_Batch

* placing table into a cluster - when the placing table into cluster table must have column that matches with cluster key.

→ create table courses (ccode varchar2(10), name)
cluster course_Batch (ccode);

→ create table batch (bcode varchar2(20), ccode)
cluster course batch (ccode);

Scanned with CamScanner

Page:
Date: / /

⑦ Data Manipulation language.

i) Insert statement.

ii) Truncate statement.

iii) Database transaction.

i) commit.

ii) rollback.

⑧ constraints

① ~~Row~~ column level constraints

② Table level constraints.

i) NOT NULL constraints.

ii) UNIQUE constraints.

iii) PRIMARY key constraints.

iv) FOREIGN key constraints.

v) Check constraints.

vi) Default constraints.

⑨ Time date data types.

① Timestamp.

② Interval to year & month

③ Interval to days to second

— Timestamp with time zone data types.

⑩ Database object -

1) Table

2) view — simple view & complex view

3) sequence

4) synonym

5) Index, — clustered index | non-clustered index.

⑩-II) Clustering - creating cluster index & placing table into a cluster.

Scanned with CamScanner

create ~~col~~ table & alter table.

note

N O T E S

Q. use of constraints ?
Q. use of join ?
Q. use of set operator ?
Q. use of SQL function ?
Q. use of join ?
Q. use of schema object (database object) ?
Q. what is timestamp ?
Q. Drawbacks of Natural joins.

- use of SQL ~~statement~~ function -
  - To perform calculation on data
  - modify individual data items.
  - manipulate output for group of rows
  - To convert column data types.

- use of set operators -
  To combine the result of two or more queries into an result.

1) Reading news
2)

Ganesh Khedkar