1. I have 3.5 years of work experience in ETL and DWH Testi

2. The tools which I have worked on are Informatica as ETL DWH purpose used oracle, SQL, UNIX is for file-level validation purpose, JIRA is used for ticketing tool.

3. We were usually receiving different source files into the S3 server in txt, cvs formate, those source files will be loaded into SFTP for basic file-level validation.

4. In this file-level validation, based on the FRS document we used to perform filename validation, file extensions valid file separator validation, date formate validation. we also need to check whether the file is having a count or n

5. Once we are good at file-level validation we would need to load the data into staging level, staging level is nothing but loading data into oracle by using SQL.

6. At the staging level, we do mandatory checks, that check would be data validation, this we used to do count validat null or not null, duplicate validation, Sample validation, Metad validation- in metadata we also need to check column leng column size, data type, key column based on BRS document

7. Once we are good at this staging level will load the data into MDM level by using Informatica. From here we do check the first name, last name by using Match & Merge rules like exact match rule, fuzzy match rule, auto-match rule.

8. Based on this will generate one golden record in attribute level. On top of that will apply Surveyship rules like Source priority, recency, Frequency, aggregation.

9. Based on this again we generate a golden record at the Source level, this golden record will be exported into DWH.

10. Based on DLD document will do check count, duplicate, metadata, and date formate Validation.

Apart from this we also need to do day-to-day activities like email checks, priority checks, daily and weekly report handling.

File level Validations:

Compare with FRS document & Perform validations

1. Naming validations
2. Extension validations
3. File date format validations
4. File Separator validations
5. File Size validation
6. File Header validation
7. File Column Count validation
8. File Sequence validation
9. File enclosement

Staging level validations:

Job status [Fail / pass / Aborted]

Count of Records

Processed files are archieved or not

Duplicate validations

Metadata validations

- Datatype validations
- Length validations

Ex: Data type: Int / Number (100), Char (200), Varchar (50) — Length

Mandatory column validations

- Null
- Not Null

Sample data validation

Cleanse rules validation

- Unwanted empty spaces → Remove (Trim)
- Some character functions based on requirements: upper, low
- Reg exp (Regular expression) functions usage

MDM level Validations :

Verify the data load Job status [Pass/ fail / Aborted]

Count Validations

Processed files are archieved or not

Sample data Validations

Match & Merge rules based on attributes [first name, last name, city, state, zip]

Generate ov Records / Golden records

Survey ship Rules

Source Priority

Recency

frequency

Aggregation

DWH level validations:

Job status

log (count)

out folder files check

Duplicate checks [as per DID]

Null & Not Null

MINUS

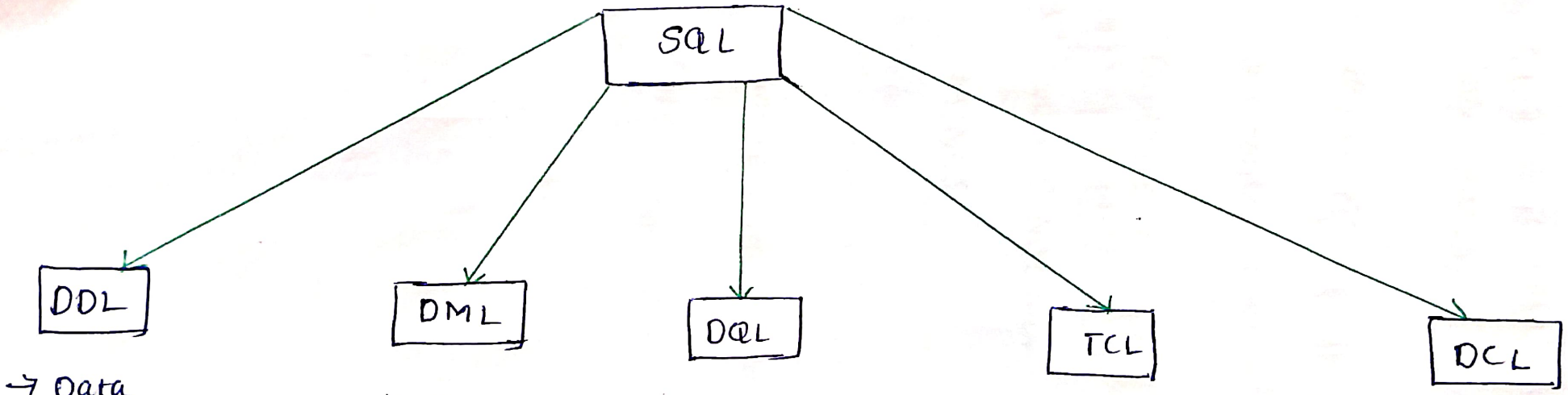
Metadata validation

Sample data validation

Date format validations

SQL - Structure Query language

SQL server Majorly devided into 5 Sub language (or) Server Commands



→ Data Definition Language

Commands

1. Create
2. Alter
3. Drop
4. Truncate
5. Rename

→ Data Manipulation Language

Commands

1. Insert
2. update
3. Delete

→ Data Query language

Commands

1. Select
2. print

→ Transactional Control Language

Commands

1. Commit
2. Roll back
3. save point

→ Data Control language

Commands

1. Grant
2. Revoke

DDL Commands: DDL Commands we can apply on Structure or Metadata

Thes commands are divided into 5 types

1. Create
2. Alter
3. Truncate
4. Rename
5. Drop.

1. Create : By using create command we can create the table.

Syntax:

```
Create table tablename
        ( column1 datatype (size),
          column2 datatype (size),
          :
          columnn datatype (size));
```

Example:

```
Create table emp
        (eno number(2)
         ename varchar2(20));
```

O/p: EMP

Eno	Ename

2. Alter: It is used for modifying the existing table structure.

* Alter with Add: This statement is used for adding extra columns to existing columns

Syntax: Alter table tablename add

(column1 datatype (size),

column2 datatype (size),

:

columnn datatype (size)) ;

Example: Alter table emp add

(sal numbe (10,2),

doj date);

o/p: EMP

Eno	Ename	Sal	doj

Alter with Modify: we can change the data type.

Syntax: Alter table tablename modify columnname datatype(size);

Example 1:

Alter table emp modify eno number(4);

Increasing the size 2 to 4

Example 2:

Alter table emp modify ename varchar2(10);

Decreasing the size 20 to 10

Alter with Drop: we can delete the column (with data).

Syntax: Alter table tablename drop column <Column name>;

Example:

Alter table emp drop doj;

O/p: EMP

Eno	Ename	Sal

Alter with Rename: To rename a column

Syntax: Alter table tablename rename column <old column name> to <new column name>;

Example:

Alter table emp rename column sal to Salary;

O/p: EMP

Eno	Ename	Salary

3. Truncate: We will use this command to delete the table data permanently

Syntax: truncate table tablename;

Example: truncate table emp;

Table truncated.

If we perform roll back also we can not retrieve the data after truncating.

4. DROP: It is used to delete the entire structure of the table.

Syntax: drop table tablename ;

Example: drop table Emp;

5. Rename: It is used for renaming the existing table name to new table name

Syntax: rename <old name> to <newname>;

Example: rename emp to employee;

DML Commands: To modify the table data we will use DML commands. These commands we can apply on data.

1. Insert: By using insert command we can add values into a table.

1. By using value method
2. By using Address method

Value Method:

Method-1 Oka table lo evevi columns unnayo, vaati corresponding datatypes kuda gurthu untae, 1st method use chesi

Syntax: Insert into tablename values (val1, val2, ... valn);

Example Insert into emp values (1, 'Jag', 35000);

Intially we can create the table

```
Create table emp (eno numbe (3)
                  ename varchar(20)
                  Sal  number(10,2));
```

O/p: EMP

ENo	EName	Salary
1	Jag	35000

Method 2: We can take whatever columns required

Syntax: Insert into tablename (col1, col2, ... coln) values (val1, val2...n);

Example: ① Insert into emp (ename, eno, salary) values ('Raj', 2, 39000);

O/p: EMP

ENO	EName	Salary
1	Jag	35000
2	Raj	30000

Example ②

we dont the salary

Insert into emp (eno, ename) values (3, 'Siva');

EMP

O/p:

ENO	EName	Salary
1	Jag	35000
2	Raj	30000
3	Siva	-

for suppose we don't the values, we can mention NULL

Insert into emp (eno, ename, salary) values (4, null, null);

O/p: EMP

ENO	EName	Salary
1	Jag	35000
2	Raj	3000
3	Siva	-
4	-	-

Address Method (&) : (N-1) Inserting values into all columns

Syntax: Insert into tablename values (&eno, &ename, & salary)

then press enter

o/p: After pressing the enter it processes and asking atta

Enter value for eno: 5 → press Enter

Enter value for ename: Vamsi → press Enter

Enter value for salary: 20000 → press enter

1 row created

EMP

ENO	EName	Salary
1	Jag	35000
2	Raj	30000
3	Siva	—
4	—	—
5	Vamsi	20000

If we enter another employee details no need to write commands. Just write reputation command (/)

Ex: SQL> / — press Enter

After pressing enter it will run the last command we u

it prompts the user to enter details

M-2 Inserting values into specific columns

Syntax: Insert into tablename (col1, col2, col3. .) values (&val1, val2, ..);

Example: Insert into emp (eno, ename) values (&eno, '&ename');

Press enter

Enter value for Eno : 2

Enter value for Ename : Radha

1 row created

O/P: EMP

ENO	EName	Salary
1	Jag	35000
2	Raj	30000
3	Siva	–
4	–	–
5	Vamsi	20000
2	Radha	–

Null value

Insert into emp (eno, ename, Salary) Value (&eno, '&name', Null)

Enter value for Eno :- 7

Enter value for Ename : Murali

1 row created

ENO	EName	Salary
1	Jag	35000
2	Raj	30000
3	Siva	–
4	–	–
5	Vamsi	20000
2	Radha	–
7	Murali	–

2. Delete:

To remove any records from existing table.

Delete command is used for removing/deleting the all the records or either specified records from a table.

* It is a temperary deletion

By using rollback we get back deleted records.

To delete all Records

Syntax: delete tablename;

Example: delete emp; ↵

Rollback; ↵ we can get deleted records

To delete Specific Records

Syntax: delete tablename where <condition>;

Example: delete emp where eno = 5;

delete emp where eno = 2;

To delete permanently

Syntax: delete from tablename;

3. Update: It is used for modifying the existing data in a table.

Syntax: Update tablename set col1 = value1,
col2 = val2, ...
where <condition>;

Example update emp set salary = 5000,
where eno = 3;

O/p:

eno	ename	salary
3	Siva	50000

update emp set ename = 'Mani', salary = 70000,
where eno = 4;

update emp set salary = 20000, where eno = 2;

update emp set salary = 30,000, where ename = 'Radha';

O/p: EMP

ENO.	EName	Salary
1	Jag	35000
2	Raj	20,000
3	Siva	50000
4	Mani	70000
5	Vamsi	20,000
2	Radha	30000
7	Murali	—

DQL Commands: Using this commands we can retrieving the data.

1. **Select:** To retrieving the data (or) to take o/p we will use this Select command.

Syntax: Select * from tablename;

Ex: Select * from emp;

o/p: It will display the all columns in the table

EMP

ENo	EName	Salary
1	Jag	35,000
2	Raj	20000
3	Siva	50000
4	Mani	70000
5	Vamsi	20000
2	Radha	30000
7	Murali	—

2. **Print:** To print the values.

Syntax: print Message

Ex: Print (my name is Jag);

o/p: My name is Jag.

TCL Commands: Transaction Control language is a language that manages transactions within the database

It is used to execute the changes made by the DML statements

1. **Commit:** It is used to permanently save any transaction into the database

   Syntax: commit ;

2. **Rollback:** It is used to restore the database to that state which was last commited

   Syntax: Rollback;

3. **Savepoint:** The changes done till savepoint will be unchanged and all the transactions after savepoint will be rolled back.

   Syntax: savepoint <name>;

   savepoints helps to save

DCL Commands: To control the data means to give permission to particular person.

1. Grant: Giving permission

Syntax: grant permission/DML on tablename to User List (column name, domain name)

Example grant select on emp to 'Jay';

2. Revoke: What ever we give permission that we want to take return

Syntax: revoke permission/DML on tablename from User List (column name).

Example revoke select on emp from 'Jay';

Aggregate functions:

1. Count
2. Sum
3. Min
4. Max
5. Avg

1. COUNT: We can identify the no.of records/ rows in a table

Syn: Select Count(*) from tablename emp;

* NULL's won't consider

2. SUM: We can find the total values in the specified column

Syn: Select SUM (columnname) from tablename;

3. MIN: We can ~~find~~ identify the lowest values in the specified column.

Syn: Select MIN (columnname) from tablename;

4. MAX: We can identify the highest value in the Specified Column.

Syn: Select MAX (columnname) from tablename;

5. AVG: We can find the average of values in the Specified Column

Syn: Select AVG (columnname) from tablename;

SQL Clauses

1. Group by
2. Having
3. Order by
4. Distinct

1. **Group by:** This statement is used to arrange identical data into groups. We can used with the Select Statement.

* The Group by statement is used with Aggregate Function

Syn: Select columnname, Aggregate function from tablename group by columnname;

Ex: ① Select dept, count(1) from emp group by dept;

O/p: (null) = 7000 2

Admin = 7000 1

② Select dept, Max(sal) from emp group by dept;

O/p:

(null) = 7000

Admin = 7000

* WHERE Clause - We can use this before group by Clause.

Ex: → Select dept, Max(sal) from emp where name='sai' group by dept;

→ Select dept, count(1) from emp where dept='IT' group by dept;

2. Having Clause:

* We can use having clause after group by clause.

Syn: Select columnname, aggregate function from tablename group by columnname having aggreate function condition;

Ex: Select gender, count(1) from emp group by gender having count(1) = 1;

3. Order by clause: Order by statement comes at the end of the sql statement.

Ex: → Select gender, count(1) from emp where id is not null group by gender having count(1) = 1 order b

→ Select * From emp order by desc;

→ Top 2 records:

Select * from emp order by rownum ≤ 2;

→ Last 2 records:

Select * from emp where rownum ≤ 2 order by rownum desc;

4. Distinct Clause: The Statement is used to return only distinct (different) values

Syn: Select distinct column1, column2 ... from tablename;

Query Creation Order

Select

|

columnname

|

From

|

Tablename

|

where clause / with condition

|

Group by

|

Having clause with condition

|

Order by (desc/ASC)

|

•

;

Query Execution Order

From

|

where clause with condition

|

Group by

|

Having clause with condition

|

Select

|

distinct

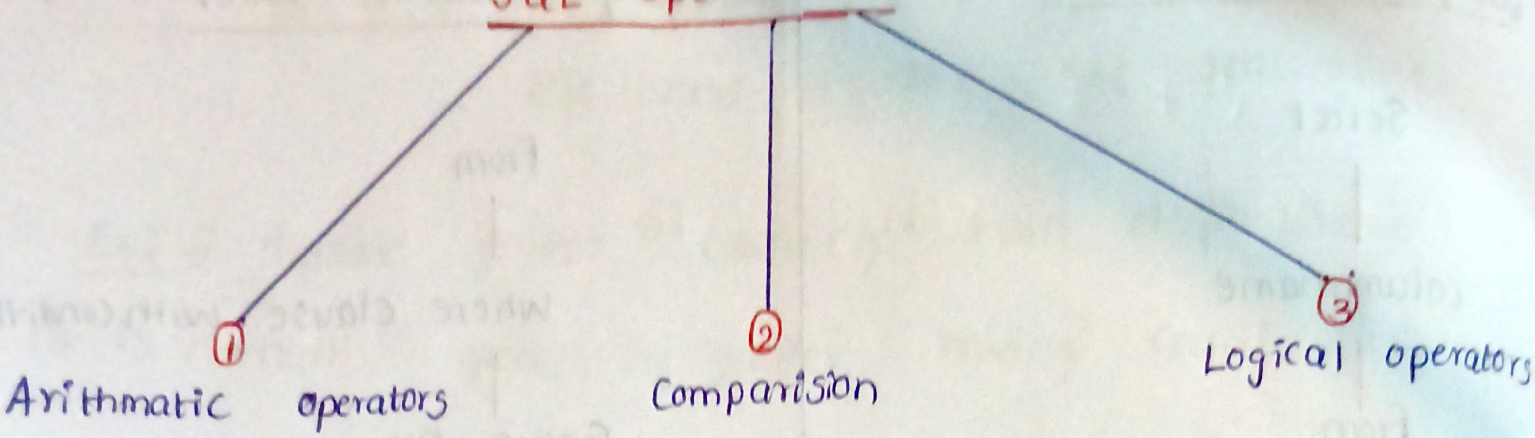
|

Order by

|

limit / offset

SQL Operators



Arithmatic operators	Comparision	Logical operators
+ → a+b	= → a=b	ALL
- → a-b	!= → a!=b	AND
* → a*b	<> → a<>b	ANY
/ → a/b	> → a>b	BETWEEN
% → a%b	< → a<b	IN
	>= → a>=b	NOT
	<= → a<=b	OR
	!< → a!<b	EXISTS
	!> → a!>b	LIKE

1. Arithmatic Operators:

Select columnname, ↑ Columname $\pm$ / $*$ from emp

→ Select Salary, Salary + 500 as total from emp;

→ Select Salary, Salary - 500 as total from emp;

→ Select Salary, Salary * 500 as total from emp;

→ Select Salary, Salary / 500 as total from emp;

2. Comparision Operators:

=
!= not equal
<> — not equal to

→ Select * from emp where Salary > 1000;

<
>=
<=

→ Select * from emp where Salary !< 1000;

!>

3. Logical Operators:

→ Select * from emp where Salary all (select Salary from emp where dept = 'IT');

→ Select * from emp where Salary < Any (select Salary from emp where dept = 'IT');

→ Select * from emp where name = 'sai' AND age <= 40;

→ Select * from emp where name = 'sai' OR age <= 40;

→ Select * from emp where age BETWEEN 25 AND 40;

→ Select * from emp where gender IN ('M');

→ Select * from emp where gender NOT IN ('M');

→ 2nd Highest Salary

Select MAX (Salary) from emp where Salary NOT IN (Select max(salary) from emp);

→ Ex Select * from emp where EXISTS (Select Salary from emp where dept = 'IT');

→ Select * from emp where NOT EXISTS (select 1 from emp)

→ Select * from emp where Salary > SOME (Select salary from emp where Salary > 20);

Like:

- Select * from emp where name like 'A%'; — Ajay
- Select * from emp where name like '%y'; — Ajay
- Select * from emp where name like '__A%'; — AjAy
- How Find last two characters in column name?

  Select * From emp where name like '%A__';

→ How to add another column?

Select 'Columnname', a.* from emp a;

Set Operators

Thumb rules:

1. Both tables column datatype should be match in Source & Target
2. Source table column count = Target table column count
3. Source table column Sequence = Target table column Sequence
4. To overcome the datatype conversion issue, we can use to-char function

* Select to-char (Id), Name from tableA

UNION / UNION ALL / MINUS / Intersect

Select Id, Name from table B;

1. UNION - Avoiding the duplicates and Sorting the data
2. UNION ALL - It display the both tables data
3. INTERSECT - It returns only common rows. ~~returned by the~~
4. MINUS - It is used to return all rows in the first table that are not in second table.

* Select columnname from TableA

UNION

Select columnname from TableB;

Source Table A-S

ID
1
2
3
4
5
1
2

Target Table A-T

ID
1
2
1
2
3
6
7

UNION

1
2
3
4
5
6
7

UNIONALL

1
2
3
4
5
1
2
1
2
1
2
3
6
7

Intersect

1
2
3

MINUS

4
5

SQL JOINS

1. Equi / Inner / Natural Join
2. Left outer Join
3. Right outer Join
4. Full outer Join
5. Self Join
6. Cross Join

1. Equi / Inner / Natural Join: It will pull both tables Matching data

Syn: * Select * from tabl A

Inner Join

table B on table A.id = table B.id;

* Select * From table A, table B where table A.id = table B.id;

2. Left outer Join: Return all records from the left Side table along with Right Side table Matching records.

Syn: * Select * from table A

leftouter Join

table B on table A.id = table B.id;

* Select * from table A, table B where table A.id (+) = table B.id

3. Right outer Join: Return all records from the right side table along with left side table matching records.

Syn: * Select * from table A

Right outer Join

table B on table A.id = table B.id;

* Select * from table A, table B where table A.id = table B.id(+);

4. Full outer Join: Both tables complete information it will display.

Syn: Select * from table A

Full outer Join

table B on table A.id = table B.id;

5. Self Join: It is a regular join, but the table is joined with itself.

Syn: Select * from Table A a1 Table A b1

where b1.Mgr Id = a1.emp id;

6. Cross Join: * The Cross Join keyword returns all records from both tables.

* It will work like a cartisian product. that means, Multiplication of the rows.

Syn: Select * from tableA
Cross Join tableB;

Table A

1
1
2
4
5
NULL

Table B

1
5
2
3
6
-

Equi / Inner / Natural Join

Id	Id-1
1	1
1	1
2	2
5	5

left outer Join

Id	Id-1
1	1
1	1
2	2
4	NULL
5	5
NULL	NULL

Right outer

Id	Id-1
1	1
1	1
5	5
2	2
NULL	3
NULL	6

Full outer

Id	Id-1
1	1
1	1
2	2
NULL	3
4	NULL
5	NULL
NULL	NULL
NULL	6

Cross Join

5 x 6

1	1
1	5
1	2
1	3
1	6
1	1
1	5
1	3
1	6
2	1
2	5
2	2
2	3
2	6
⋮	

Self Join

Character functions

LOWER - Select LOWER ('ETL TESTING') from dual;

O/P: etl testing

UPPER - Select UPPER ('etl testing') from dual;

O/P: ETL TESTING

INITCAP - Select INITCAP ('etl testing') from dual;

O/P: Etl Testing

LENGTH - Select LENGTH ('ETL Testing') from dual;

O/P: 11

* This will count spaces

SUBSTR * Select SUBSTR (columnname, string starting position, length of the string) from tablename;

Sub string

* Select SUBSTR ('ETL Testing', 1, 5) from dual;

O/P: ETL_T

INSTR :* Select INSTR ('ETL Testing', 'E') from dual;

o/p: 1

Test wheather a given character occurs in the given string or not. If the character occurs in the string then the returns the first position of its occurance otherwise it returns zero.

Character

* Select INSTR ('ETL Testing', 'T') from dual;

o/p: 2

* Select INSTR ('ETL Testing', 'T', 2) from dual;

2nd occurance character

o/p: 5

REPLACE - Replace a given set of characters in a string with another set of characters

* Select REPLACE (columnname, Replaceble string, Replaced string) from dual;

* Select REPLACE ('ETL TESTING', 'ETL', 'DWH') from dual;

o/p: DWH TESTING

TRANSLATE: We can use this function to replace characters in a given string with your coded characters.

```
Select TRANSLATE (columnname, 'character', 'replaceble charac
                                        from tablename;
```

EMP

Name
Mani
Ramu
Usha
Sandhya

```
Select TRANSLATE (Name, 'M', 'A') from emp;
```

o/p:

Name
Aani
Raau
Usha
Sandhya

```
Select TRANSLATE (Name, 'M a', 'A b') from emp;
```

o/p:

Name
Abni
Rbau
Ushb
Sbndhyb

M - A
a - b

SOUNDEX:

```
Select SOUNEX ('smith') from dual;
```

o/p: Smith
Smyth
Smythe

LPAD: We can append a character into the left side of the string.

```
Select LPAD ('ETL_TESTING', '*', 14) from dual;
```

o/p: *** ETL TESTING

RPAD: We can append a character into the rightside of the string.

```
Select RPAD ('ETL TESTING', '*', 14) from dual;
```

o/p: ETL TESTING ***

LTRIM: We can remove a spaces in left hand side

```
Select LTRIM('==ETL TESTING') from dual;
```

o/p: ETL TESTING

RTRIM : We can remove a spaces in right hand Side.

```
Select RTRIM ('ETL TESTING--') from dual;
```

o/p: ETL TESTING

TRIM :

```
Select trim ('ETL TESTING') from dual;
```

o/p: ETLTESTING

CONCAT : combines a given string with another string

```
Select CONCAT (CONCAT(columnname, string), string) from tablename;
```

```
Select CONCAT ('ETL TESTING', 'TEST Engineer') from dual;
```

o/p: ETL TESTINGTEST Engineer

```
Select CONCAT ('Jaya', 'Kumar') from dual;
```

o/p: Jayakumar

```
Select 'ETL TESTING' || 'TEST ENGINEER' From dual;
```

o/p: ETL TESTINGTEST ENGINEER

DATE FUNCTIONS

Current-date:

```
Select current-date from dual ;
```

o/p: 24-12-2021

Sysdate:

```
Select sysdate from dual;
```

Add_Months:

```
Select add-months ( date , 2 ) from dual;
```
How many months we need to add

Timestamp

Return a datetime value based on the arguments

```
Select TIMESTAMP ('2017-07-23', '13:10:11') from dual;
```
o/p:- 2017-07-23 13:10:4

```
Select Current TIMESTAMP From dual;
```

We can get the current date along with the timestamp as well

Extract: We can extract the particular date or year or Month in the specified date column.

```
Select EXTRACT ( day   from sysdate) from dual; → 24
                 year                             2021
                 Month                            12
```

From_TZ:

Select From_TZ (timestamp '2021-12-23 18:42:24', '12:00')
from dual;

o/p: 23-DEC-21 18:42:24 PM +12:00

LAST-DAY: We can find out that last day of the Specified Month

Select LAST_DAY (sysdate) from dual;

o/p: 31-Dec-2021

Select Last_day (date '2017-05-03') from dual;

o/p: 31-May-2017

LOCALTIMESTAMP:

Select localtimestamp From dual;

o/p: 24-Dec-2021 09:12:24:15 AM
HH MIN SEC MS

MONTHS_BETWEEN

Select MONTHS_BETWEEN (date '2017-07-01',
date '2021-01-01') from dual

o/p: 53 Months

NEXT-DAY

Select NEXT-DAY ('2021-12-24', 'Sunday') from dual;

o/p: 26-Dec-2021

Select NEXT-DAY (Sysdate, 'Friday') from dual;

o/p: 31-Dec-2021

TO-CHAR

Select TO-CHAR (Sysdate, 'YY-MM-dd') from dual;

o/p: 21-12-24

Select To-char (sysdate, 'YYYY-MM-dd') from dual;

o/p: 2021-12-24

Select To-char (sysdate, 'YY-Mon-day') from dual;

o/p: 21-DEC-Friday

Select To-char (Sysdate, 'YY-mm-dd') from dual; → 21-12-24

Select To-char (Sysdate, 'year-Month-day') from dual

o/p: 2021-DECEMBER-FRIDAY

To-date:

Select To-date (sysdate, 'dd-mm-yy') from dual;

o/p: 24-12-21

Select To-date ('01-January-2017', 'dd-MM-yy') from dual

o/p: 01-Jan-17

Number Functions

Addition:

```
Select value1 + value2 From tablename;
```

Substraction:

```
Select value1 - value2 From tablename;
```

Multiplication:

```
Select value1 * value2 From tablename;
```

Division:

```
Select value1/value2 From tablename;
```

ABS:

We can convert negative values into positive values

```
Select ABS(-10) From dual;
```

o/p: 10

CEIL:

Returns the smallest integer value

```
Select CEIL (120.45) from tablename:
```

o/p: 121

Floor:

Select Floor (120.45) from tablename;

o/p: 120

MOD: We can findout the even and odd numbers returns the remainder of n÷m

Select MOD (10,2) from dual;

o/p: 0

```
2)10(5
  10
  --
   0
```

ROUND:

Select ROUND (8.2) from dual;

o/p: 8

Select ROUND (8.6) from dual;

o/p: 9

TRUNC:

Select TRUNC (3.141596, 3) from dual;

o/p: 3.141

POWER :

```
Select POWER(2,3) from dual;
```

O/p: 8

GREATEST :

```
Select GREATEST (columnname) from tablename;
```

A

ID
1
7
9
13
20

```
Select GREATEST (ID) from A;
```

O/p: 20

LEAST :

```
select LEAST (columnname) from tablename;
```

```
Select LEAST (ID) from A;
```

O/p : 1

SQL General Functions

NVL(): In SQL, NVL converts a null value to actual value.

A

ID
5
7
0
Null
20
Null
3

Syntax: Select NVL (exp1, exp2) from tablenar

exp1: is the source value or expression that may contain a Null.

exp2: is the target value for converting the Null.

Example:

Select NVL (Id, 100) from A;

A

O/P:

ID
5
7
0
100
20
100
3

NVL2():

* The NVL2 function examines the first expression.
* If the first expression is not null, then the NVL2 function returns the second expression.
* If the first expression is null, then the NVL2 function returns the third expression.

i.e exp1 is not null NVL2 returns exp2. If exp1 is null NVL2 returns exp3.

Syntax:

Select NVL2 (exp1, exp2, exp3) from tablename;

Exp1: is the source value, that may contain null

Exp2: is the value returned, if exp1 is not null

Exp3: is the value returned if exp1 is null.

Table A

ID
500
100
NULL
300
NULL
1000

Select NVL2 (Id, Id+100, Id-100) from table A;

o/p:

ID
600
200
-100
400
-100
1100

Select NVL2 (Id, 2000, 3000) from table A

o/p:

Id
2000
2000
3000
2000
3000
2000

DECODE( ):

It will work like case statement (or) if-then-else statement.

If the default value omitted, a null value is returned where a search value doesnot match any of the result values.

Syntax:

Select DECODE (Columnname, Search1, result1, Search2, result2, ... Searchn, resultn, default) from tablename;

Table A

Id	Name	designation	city	Salary
1	Jag	Test Engineer	HYD	20000
2	Siva	Admin	CHN	50000
3	Mani	Support	NULL	60000
4	Raju	Developer	BNG	70000

```
Select DECODE ( city, 'HYD', 'Hyderabad', 'CHN', 'Chennai',
                'Null', 'Mumbai', 'BNG', 'Bangalore')
                from Table A;
```

```
Select NVL ( DECODE (city, 'HYD', 'Hyderabad', 'CHN', 'Chennai'
'BNG', 'Bangalore') 'Mumbai') from Table A;
```

o/p:

HYD	Hyderabad
CHN	Chennai
Null	Mumbai
	Bangalore

COALESCE( ):

The Function examines the first expression, IF the first expression is not null, it returns that expression otherwise it does a COALESCE of the remaining expressions.

Syntax:

Select COALESCE (exp1, exp2, ... expn) from tablename;

Table A:

ID	Salary
02	8400
015	7000
010	6200
NULL	3200
NULL	3100
NULL	2500

Select COALESCE (Id, Salary, 10) from TableA;

o/p:

ID
02
015
010
3200
3100
2500

NULLIF( ):

The NULLIF function compares two expressions, IF they are equal the function returns null. IF they are not equal the function returns the first expressio

Syntax:

Select NULLIF (exp1, exp2) from tablename;

Table A:

First-Name	Last-Name
Mani	Subbu
Jagadeesh	Raju
Mohan	Ashok
Kumar	Reddy
Ganesh	Venkey
Subbu	Venkata Mohan

Select length (first-name) length (second-name), NULLIF (length (first-name), length (second-name)) from table A;

O/P:

First-name	Last-name	Result
4	5	4
9	4	9
5	5	NULL
5	5	NULL
6	6	NULL
5	13	5

LNNVL( ):

The function can be used only in the where clause of a query. It takes as an argument a condition and returns TRUE, if the condition is False/unknown and FALSE, if the condition is TRUE.

NANVL( ):

→ The function is useful only for a floating point numbers of type BINARY_FLOAT or BINARY_DOUBLE.

→ It instructs the database to return an alternative value n2. If the input value n1 is NAN (not a number)

→ If n1 is not NAN, then database return n1, this function is usefull for mapping non values to NULL.

Syntax:

```
Select NANVL ( n1,n2) from tablename;
```

→ NVL (vs) NVL2 (vs) COALESCE

→ DECODE (vs) NVL2

→ NVL (vs) COALESCE

Constraints

Syntax:

Create table tablename (columnname datatype(size) Constraint name)

1. NOT NULL
2. UNIQUE
3. Primary Key
4. Foriegn Key
5. Check
6. Default

1. NOT NULL:

This constraint cannot store a null value in a column

```
Create table emp1 ( Id int NOT NULL,
                    Name varchar(200) NOT NULL,
                    Salary int NOT NULL,
                    Age int
                    Gender varchar(100)
                  );
```

UNIQUE :

This constraint tell that all values in the column must be unique. It cannot allow the duplicate values, It can allow null value.

UNIQUE constraint it can allow multiple null values and should not load the duplicate values.

→ Create table emp 2 (Id int ~~NOT NULL~~ UNIQUE,
Name varchar(200) Unique,
Salary int unique,
Age int,
~~Salary~~ varchar (100)
gender
);

→ Create table emp3 (Id int NOTNULL UNIQUE,
Name varchar(100) UNIQUE,
Salary varchar(100) NOTNULL
Age int
);

Primary Key:

* Uniquely identify a each record in a table.
* It cannot be allow a duplicate and null values.
* Combination of NOT NULL and UNIQUE, that means it won't allow NULL values and avoid duplicates.

```
Create table emp4 (Id int PRIMARY KEY,
                   Name varchar(100) NOT NULL,
                   Age int NOT NULL,
                   Salary int
                   );
```

Foriegn key: which can uniquely identify each row in a another table and this Constraint is used to specify a field as foriegn key.

Parent table

```
Create table persons (PersonID int PRIMARY KEY,
                      Name varchar(100),
                      Age int);
```

Child table

```
Create table order (orderID int NOT NULL,
                    OrderNumber int NOT NULL,
                    personId int,
                    primary key (orderID),
                    foriegn key (personID) References person (personID));
```

CHECK:

It evaluate validate the values of a column to meet a particular condition

```
Create table emp5 ( Id int,
                    Name varchar(100),
                    Age int,
                    Check (Age > 20)
                  );
```

DEFAULT:

This Constraint specifies a default value for the column when no value is specified by the user

```
Create table emp6 ( Id int,
                    Name varchar(100)
                    Age int,
                    City varchar(100) Default 'HYD'
                  );
```

NULL → 'HYD'

whenever NULL is present table, it will replaces with HYD.