

WEEK – 1

Implement the following using DUAL table:

- a) Character functions
- b) Numeric functions
- c) Date functions
- d) Conversion functions.

The DUAL table is a special, single-row, single-column table used in Oracle (and similar) databases primarily to satisfy the SQL syntax requirement of a FROM clause when executing functions or calculations that don't need data from a user table.

Implementation Using DUAL Table

a) Character Functions

Purpose: Manipulate or return information about character strings.

- **DUAL Use:** Allows developers to test string operations directly.

b) Numeric Functions

- **Purpose:** Perform mathematical calculations.
- **DUAL Use:** Enables SQL to be used as a calculator for one-off operations.

c) Date Functions

- **Purpose:** Operate on and retrieve date and time information.
- **DUAL Use:** Standard method to get the current system date/time or perform date arithmetic.

d) Conversion Functions

- **Purpose:** Change data from one datatype (e.g., Date, Number) to another (e.g., Character).
- **DUAL Use:** Testing formatting and conversion logic before applying it to table data.

SQL*Plus: Release 11.2.0.1.0 Production on Mon Jul 21 13:57:34 2025

Copyright (c) 1982, 2010, Oracle. All rights reserved. Enter user-name:

24071A6627@Orcl11g/24071A6627

Connected to:

Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production

With the Partitioning, OLAP, Data Mining and Real Application Testing options

Character functions

1. select ascii('a') from dual;

ASCII('A')

97

2. select chr('100') from dual;

C

-

D

3. select concat('hello','world') from dual;

CONCAT('HE

lloWorld

4. select initcap('database management systems') from dual;

INITCAP('DATABASEMANAGEMENT

Database Management Systems

5. select instr('character','h') from

dual; INSTR('CHARACTER','H')

2

6. select length('dual table') from dual;

LENGTH('DUALTABLE')

10

7. select lower('Schema and INSTANCE') from dual;

LOWER('SCHEMAANDINS

schema and instance

8. select upper('datatype') from dual;
UPPER('D
DATATYPE
9. select rpad('primary',8,'A') from dual;
RPAD('PR
primaryA
10. select lpad('primary',8,'A') from
dual; LPAD('PR
Aprimary
11. select replace('jack and jond','j','bl') from
dual; REPLACE('JACKAN
black and blond
12. select translate('jack and jond','j','bl') from dual;
TRANSLATE('JA
back and bond
13. select rtrim('vnrvjiet','iet') from dual;
RTRIM
vnrvj
14. select ltrim('vnrvjiet','vnr') from
dual; LTRI
jiet
15. select substr('Oracle','1','3') from
dual; SUBSTR
Ora

Numeric functions

1. select mod(14,5) from dual;
MOD(14,5)

4

2. select power(2,2) from dual;
POWER(2,2)

4

3. select round(5.778,2) from dual;
ROUND(5.778,2)

~~5.78~~-----

4. select trunc(5.778,2) from dual;
TRUNC(5.778,2)

~~5.77~~-----

5. select sqrt(25) from dual;
SQRT(25)

~~5~~-----

6. select floor(4.99) from dual;
FLOOR(4.99)

~~4~~-----

7. select ceil(4.01) from dual;
CEIL(4.01)

5

8. select (14*8) from dual;
(14*8)

112

Date functions

1. select add_months(date '2025-09-15',1) from dual;
ADD_MONTH
15-OCT-25
2. select current_date from dual;
CURRENT_D
21-JUL-25
3. select sysdate from dual;
SYSDATE
21-JUL-25
4. select current_timestamp from dual;
CURRENT_TIMESTAMP
21-JUL-25 15:41:16.427000+05:30-----
5. select extract(day from date '2025-12-02') from dual;
EXTRACT(DAYFROMDATE'2025-12-02')
2-----
6. select last_day(date '2016-02-01') from dual;
LAST_DAY(
29-FEB-16
7. select months_between(date '2031-03-12',date '2024-08-28') from dual;
MONTHS_BETWEEN(DATE'2024-08-28',DATE'2031-03-12')
78:483871-----
8. select next_day(date '2012-02-12','thursday') from dual;
NEXT_DAY(
16-FEB-12
9. select round(date '2023-12-01','YYYY') from dual;
ROUND(DAT
01-JAN-24

10. select trunc(date '2023-12-01','YYYY') from dual;
TRUNC(DAT
01-JAN-23

Conversion functions

1. select to_char (date '2017-01-01') from dual;
TO_CHAR(D
01-JAN-17
2. select to_date('01 jan 2017') from dual;
TO_DATE('01-JAN-17

WEEK – 2

Practice DDL and DML commands on a basic table without integrity constraints.

DDL Commands (Data Definition Language)

DDL commands are used to define or modify the structure of database objects. They handle the schema itself.

- **CREATE:** Defines and builds a new table, specifying column names and data types.
- **ALTER:** Changes the existing structure of a table, such as adding, modifying, or dropping a column.
- **TRUNCATE:** Removes all rows from a table structure rapidly, but keeps the empty structure intact. It's classified as DDL because it resets the table's storage allocation.
- **DROP:** Completely removes a table (structure and all data) from the database.

DML Commands (Data Manipulation Language)

DML commands are used to manage and manipulate the data within the structured tables.

- **INSERT:** Adds new rows (records) of data into a table.
- **SELECT:** Retrieves data from the database. (Often called DQL - Data Query Language).
- **UPDATE:** Modifies existing data in one or more rows of a table.
- **DELETE:** Removes specific rows from a table based on a condition (or all rows if no condition is specified).

1 .Creation of a table using command 'CREATE'

```
SQL> create table student(name varchar2(30), rollno number(15), branch varchar2(10),  
joining_date date);
```

Table created.

2 . Adding column to a table using command 'ALTER'

```
SQL> alter table student add email varchar2(15);
```

Table altered.

3 . Modify a column of a table using command 'ALTER'

```
SQL> alter table student modify email varchar(10);
```

Table altered.

4 . Inserting elements in to table using command 'INSERT'

```
insert into stud values('banana',673,'aiml','20-AUG-2028','abc@gil.com');
```

1 row created.

```
insert into stud values('apple',100,'cse','20-SEP-2022','a@gil.com');
```

1 row created.

```
insert into stud values('laptop',345,'ece','12-OCT-2025','a@gil.com');
```

1 row created.

```
insert into stud values('mobile',765,'aiml','22-NOV-2029','a@gil.com');
```

1 row created.

5 . Update a column of a table using command 'UPDATE'

```
UPDATE student SET email = 'xyz@gmail.com' WHERE rollno = 100;
```

6 . Display only few columns of a table using command 'SELECT'

```
SELECT name, rollno, branch FROM student;
```

| NAME | ROLLNO | BRANCH |
|------|--------|--------|
|------|--------|--------|

| | | |
|--------|-----|------|
| banana | 673 | aiml |
|--------|-----|------|

| | | |
|-------|-----|-----|
| apple | 100 | cse |
|-------|-----|-----|

| | | |
|--------|-----|-----|
| laptop | 345 | ece |
|--------|-----|-----|

| | | |
|--------|-----|------|
| mobile | 765 | aiml |
|--------|-----|------|

7. . Display the entire table

SELECT * FROM student;

NAME ROLLNO BRANCH JOINING _DATE EMAIL

banana 673 aiml 20-AUG-2028 abc@gil.com
apple 100 cse 20-SEP-2022 a@gil.com
laptop 345 ece 12-OCT-2025 a@gil.com
mobile 765 aiml 22-NOV-2029 a@gil.com

8. Delete a row from table by using where condition

SQL> DELETE FROM student WHERE name = 'laptop';

1 row deleted.

9. Use TRUNCATE command

SQL> TRUNCATE TABLE student;

Table truncated.

10. Use DROP command to drop table

SQL> DROP TABLE student;

Table dropped.

WEEK – 3

Practice DDL and DML commands on a Relational Database, specifying the Integrity constraints. (Primary Key, Foreign Key, CHECK, NOT NULL)

Integrity Constraints

Integrity constraints are rules defined at the column or table level to enforce data validity and consistency within a relational database.

- **Primary Key (PK)**
 - Definition A column or set of columns that uniquely identifies each record in a table. It ensures that no two rows have the same primary key value.
 - Rules Enforced Uniqueness (no duplicate values) and Not Null (must always have a value).
- **Foreign Key (FK)**
 - Definition A column or set of columns in one table that refers to the Primary Key of another table (or the same table).
 - Rules Enforced: Referential Integrity, which means data in the referencing column(s) must match an existing value in the referenced Primary Key column(s), thereby linking the two tables.
- **CHECK**
 - Definition A constraint that defines a condition that every value in a column must satisfy.
 - Rules Enforced: Domain Integrity, ensuring that data values fall within a specific, defined range or meet a logical requirement (e.g., salary must be greater than zero).
- **NOT NULL**
 - Definition A simple constraint that ensures a column cannot contain a null value.
 - Rules Enforced: Requires that every record has an explicit value supplied for that column.

SQL*Plus: Release 11.2.0.1.0 Production on Mon Aug 4 14:56:23 2025 Copyright
(c) 1982, 2010, Oracle. All rights reserved. Enter user-name:
24071A6627@Orcl11g Enter password: Connected to: Oracle Database 11g
Enterprise Edition Release 11.2.0.1.0 - 64bit Production With the Partitioning,
OLAP, Data Mining and Real Application Testing options

1. Use the NOT NULL Constraint

```
SQL> CREATE TABLE data( fname VARCHAR2(20) NOT NULL, lname VARCHAR2(20),  
id NUMBER(10));
```

Table created.

```
SQL> insert into data values('shin','chan',11);
```

1 row created.

```
SQL> select* from person;
```

| fname | lname | id |
|-------|-------|----|
| shin | chan | 11 |

```
SQL> insert into person values('motu','patlu',22);
```

1 row created.

```
SQL> select* from person;
```

| fname | lname | id |
|-------|-------|----|
| shin | chan | 11 |
| motu | patlu | 22 |

```
SQL> insert into person values(NULL,'doreamon',18); insert into person  
values(NULL,'doreamon',18); ERROR at line 1: ORA-01400: cannot  
insert NULL into ("24071A6627"."DATA"."fname")
```

2. Use the NOT NULL Constraint on ALTER table

```
SQL> SQL> ALTER TABLE data MODIFY id NUMBER(10) NOT NULL;  
Table altered. SQL> insert into person values('ninja','hattori',NULL); insert  
into person values('ninja','hattori',NULL); ERROR at line 1: ORA-01400:  
cannot insert NULL into ("24071A6627"."DATA"."id") *
```

3. Use UNIQUE Constraint

```
SQL> create table univ(sno number(10),book_name varchar2(20) unique,DOT Date);  
Table created.  
SQL> insert into univ values(111,'abc','21-may-2021');  
1 row created.  
SQL> insert into univ values(111,'abc','21-may-2021');  
insert into univ values(111,'abc','21-may-2021');  
* ERROR at line 1: ORA-00001: unique constraint  
(24071A6627.SYS_C00103522) violated
```

4. Use UNIQUE Constraint on ALTER table

```
SQL> alter table person add unique(ID);  
Table altered.
```

5. Use CHECK Constraint

```
SQL> CREATE TABLE orderss(  
    id NUMBER(20),  
    name VARCHAR2(20),  
    CONSTRAINT id_c CHECK(id >= 1)  
);  
Table created.  
SQL> INSERT INTO orderss VALUES(0, 'dfs');
```

* ERROR at line 1: ORA-02290: check constraint
(24071A6627.ID_C) violated SQL> INSERT INTO orderss
VALUES(5, 'abc'); 1 row created. SQL> SELECT * FROM
orderss;

| ID | NAME |
|----|------|
|----|------|

| | |
|----|-----|
| -- | --- |
|----|-----|

| | |
|---|-----|
| 5 | abc |
|---|-----|

SQL> ALTER TABLE orderss DROP CONSTRAINT id_c;
Table altered.

6. Use DEFAULT Constraint

SQL> CREATE TABLE city(name VARCHAR2(15), pincode NUMBER(6) DEFAULT
'101010');

Table created.

SQL> INSERT INTO city(name) VALUES ('ffwrf');

1 row created.

SQL> SELECT * FROM city;

| NAME | PINCODE |
|------|---------|
|------|---------|

| | |
|-------|--|
| ----- | |
|-------|--|

| | |
|-------|--------|
| ffwrf | 101010 |
|-------|--------|

7. Use DEFAULT Constraint on ALTER TABLE

SQL> CREATE TABLE person(id NUMBER(10), last_name VARCHAR2(20),
fname, VARCHAR2(20), age NUMBER(10));

Table created.

SQL> ALTER TABLE person ADD city VARCHAR2(17);

Table altered.

SQL> ALTER TABLE person MODIFY city DEFAULT 'hyderabad';

Table altered.

SQL> INSERT INTO person(id, last_name, fname, age) VALUES (16, 'iuy', 'kij', 76);

1 row created.

```
SQL> SELECT * FROM person;
ID   LAST_NAME FNAME AGE   CITY
--  -
16   iuy      kij   76   hyderabad
```

8. Use PRIMARY KEY Constraint

```
SQL> CREATE TABLE univ(sno NUMBER(10), book_name VARCHAR2(20) PRIMARY
KEY, DOT DATE);
```

Table created.

```
SQL> INSERT INTO univ VALUES(12, 'FGD', '16-MAY-2020');
```

1 row created.

```
SQL> SELECT * FROM univ;
```

```
SNO BOOK_NAME DOT
```

```
--  -
12   FGD      16-MAY-20
```

9. Use PRIMARY KEY AS CONSTRAINT

```
SQL> create table persons9 (id int Not null,lastname varchar(20) not null,firstname
varchar(20),age int,constraint pk_person2 primary key(id));
```

Table created.

```
SQL> insert into persons9 values(14,'abc','def',15);
```

1 row created.

```
SQL> select* from persons9;
```

```
      ID LASTNAME      FIRSTNAME      AGE
-----
      14 viskre      15
```

WEEK – 4

Apply the concepts of Joins, SET operations and SQL functions on any two relational schema.

• Sailors:-

| <i>sid</i> | <i>sname</i> | <i>rating</i> | <i>age</i> |
|------------|--------------|---------------|------------|
| 22 | Dustin | 7 | 45.0 |
| 29 | Brutus | 1 | 33.0 |
| 31 | Lubber | 8 | 55.5 |
| 32 | Andy | 8 | 25.5 |
| 58 | Rusty | 10 | 35.0 |
| 64 | Horatio | 7 | 35.0 |
| 71 | Zorba | 10 | 16.0 |
| 74 | Horatio | 9 | 35.0 |
| 85 | Art | 3 | 25.5 |
| 95 | Bob | 3 | 63.5 |

Reserves:-

| <i>sid</i> | <i>bid</i> | <i>day</i> |
|------------|------------|------------|
| 22 | 101 | 10/10/98 |
| 22 | 102 | 10/10/98 |
| 22 | 103 | 10/8/98 |
| 22 | 104 | 10/7/98 |
| 31 | 102 | 11/10/98 |
| 31 | 103 | 11/6/98 |
| 31 | 104 | 11/12/98 |
| 64 | 101 | 9/5/98 |
| 64 | 102 | 9/8/98 |
| 74 | 103 | 9/8/98 |

Boats:-

| <i>bid</i> | <i>bname</i> | <i>color</i> |
|------------|--------------|--------------|
| 101 | Interlake | blue |
| 102 | Interlake | red |
| 103 | Clipper | green |
| 104 | Marine | red |

Joins (Combining Tables)

Joins unite records from two or more relational tables based on a relationship between specified columns.

- **INNER JOIN:** Returns only rows where there's a match in both tables.
- **OUTER JOIN:** Returns all rows from one table and matching rows from the other, using NULL for unmatched data.
 - **LEFT JOIN:** Prioritizes all rows from the left table.
 - **RIGHT JOIN:** Prioritizes all rows from the right table.
 - **FULL JOIN:** Returns all rows from both tables, whether they match or not.
- **CROSS JOIN:** Combines every row from the first table with every row from the second (Cartesian product).

SET Operations (Combining Result Sets)

SET operations merge the results of multiple SELECT queries. The queries must have the same number of columns with compatible data types.

- **UNION:** Combines results and removes duplicates.
- **UNION ALL:** Combines results and keeps all duplicates.
- **INTERSECT:** Returns only the rows that are common to both result sets.
- **MINUS (or EXCEPT):** Returns rows from the first query that are not present in the second.

SQL Functions (Data Processing)

SQL functions process input values to return a result.

Aggregate Functions (Group)

These functions operate on a set of rows and return a single summary value.

- **COUNT():** Gets the total number of rows.
- **SUM() / AVG():** Calculates the total sum or average of numeric values.
- **MAX() / MIN():** Finds the highest or lowest value in a set.


```
create table sailorss(sid int,sname varchar(10),rating int,age float);
```

Table created.

```
SQL>descsailorss;
```

| Name | Null? Type |
|--------|--------------|
| ----- | |
| SID | NUMBER(38) |
| SNAME | VARCHAR2(10) |
| RATING | NUMBER(38) |
| AGE | FLOAT(126) |

Table created.

```
SQL> insert into sailors(sid,sname,rating,age)values(22,'dustin',7,45.0);
```

1 row created.

```
SQL> insert into sailors(sid,sname,rating,age)values(29,'brutus',1,33.0);
```

1 row created.

```
SQL> insert into sailors(sid,sname,rating,age)values(31,'lubber',8,55.5);
```

1 row created.

```
SQL> insert into sailors(sid,sname,rating,age)values(32,'andy',8,25.5);
```

1 row created.

```
SQL> insert into sailors(sid,sname,rating,age)values(58,'rusty',10,35.0);
```

1 row created.

```
SQL> insert into sailors(sid,sname,rating,age)values(64,'Horatio',7,35.0);
```

1 row created.

```
SQL> insert into sailors(sid,sname,rating,age)values(71,'zorba',10,16.0);
```

1 row created.

```
SQL> insert into sailors(sid,sname,rating,age)values(74,'horatio',9,35.0);
```

1 row created.

```
SQL> insert into sailors(sid,sname,rating,age)values(85,'art',3,25.5);
```

1 row created.

```
SQL> insert into sailors(sid,sname,rating,age)values(95,'bob',3,63.5);
```

1 row created.

```
SQL> create table reserves(
```

2 sidint,
3 bidint,
4 day date);

Table

created.

```
SQL> insert into reserves(sid,bid,day)values(22,104,to_date('10-10-98','dd-mm-yy'));
```

1 row created.

```
SQL> insert into reserves(sid,bid,day)values(22,102,to_date('10-10-98','dd-mm-yy'));
```

1 row created.

```
SQL> insert into reserves(sid,bid,day)values(22,103,to_date('10-08-98','dd-mm-yy'));
```

1 row created.

```
SQL> insert into reserves(sid,bid,day)values(22,104,to_date('10-07-98','dd-mm-yy'));
```

1 row created.

```
SQL> insert into reserves(sid,bid,day)values(31,102,to_date('11-10-98','dd-mm-yy'));
```

1 row created.

```
SQL> insert into reserves(sid,bid,day)values(31,103,to_date('11-06-98','dd-mm-yy'));
```

1 row created.

```
SQL> insert into reserves(sid,bid,day)values(31,104,to_date('11-12-98','dd-mm-yy'));
```

1 row created.

```
SQL> insert into reserves(sid,bid,day)values(64,101,to_date('09-05-98','dd-mm-yy'));
```

1 row created.

```
SQL> insert into reserves(sid,bid,day)values(64,102,to_date('09-08-98','dd-mm-yy'));
```

1 row created.

```
SQL> insert into reserves(sid,bid,day)values(74,103,to_date('09-08-98','dd-mm-yy'));
```

1 row created.

```
SQL> create table boats(  
2 bidint,  
3 bname varchar(15),  
4 color varchar(10));
```

Table created.

```
SQL> insert into boats(bid,bname,color)values(101,'interlake','blue');
```

1 row created.

```
SQL> insert into boats(bid,bname,color)values(102,'interlake','red');
```

1 row created.

```
SQL> insert into boats(bid,bname,color)values(103,'clipper','green');
```

1 row created.

```
SQL> insert into boats(bid,bname,color)values(104,'marine','red');
```

1 row created.

1.Find the names and ages of all sailors

```
SQL> select sname,age from sailors;
```

| SNAME | AGE |
|-------|-----|
|-------|-----|

| | |
|---------|------|
| dustin | 45 |
| brutus | 33 |
| lubber | 55.5 |
| andy | 25.5 |
| rusty | 35 |
| Horatio | 35 |
| zorba | 16 |
| horatio | 35 |
| art | 25.5 |
| bob | 63.5 |

10 rows selected.

2.Find all sailors with rating > 7

```
SQL> select sname from sailors where rating>7;
```

| SNAME |
|-------|
|-------|

lubber

andy

rusty

zorba

horatio

3. Find the names of sailors who have reserved boat number 103

```
SQL> select s.sname from sailors s join reserves r on s.sid=r.sid where r.bid=103;
```

SNAME

dustin

lubber

horatio

4. Find the sids of sailors who have reserved red boat

```
SQL> select distinct s.sname from sailors join reserves r on s.sid=r.sid join boats b on r.bid=b.bid  
where color='red';
```

SNAME

Lubber

Dustin

Horatio

5. Find the names of sailors who have reserved red boat

```
select distinct r.sid from sailors join reserves r on s.sid=r.sid join boats b on r.bid=b.bid where  
color='red';
```

SID

22

31

64

6.Find ages of Sailors whose name begin and end with b and has atleast 3 characters

SQL> select s.age from sailors s where s.sname like 'b%b' and length(s.sname)>=3;

AGE

63.5

7.Find names of Sailors who have reserved red or green boat

SQL> select distinct s.sname from sailors s,reservesr,boats b where s.sid=r.sid and r.bid=b.bid and b.color='red' union select distinct s.sname from sailors s,reservesr,boats b where s.sid=r.sid and r.bid=b.bid and b.color='green';

SNAME

Horatio

dustin

horatio

lubber

8.Find names of sailors who have reserved red boat but not green boat

SQL> select distinct s.sname from sailors s,reservesr,boats b where s.sid=r.sid and r.bid=b.bid and b.color='red' minus select distinct s.sname from sailors s,reservesr,boats b where s.sid=r.sid and r.bid=b.bid and b.color='green';

SNAME

Horatio

9. Find names of Sailors who have reserved red and green boat

```
SQL>select distinct s.sname from sailors s,reservesr,boats b where s.sid=r.sid and r.bid=b.bid
and b.color='red' intersect select distinct s.sname from sailors s,reservesr,boats b where
s.sid=r.sid and r.bid=b.bid and b.color='green';
```

SNAME

dustin

lubber

10. Find names of Sailors who have reserved atleast one boat

```
SQL> select s.sname from sailors s join reserves r on s.sid=r.sid group by s.sid,s.sname having
count(r.bid)>=1;
```

SNAME

lubber

dustin

Horatio

11. Findsids of Sailors who have rating of 10 or reserved boat 104

```
SQL> select distinct s.sid from sailors s,reservesr,boats b where s.sid=r.sid and r.bid=b.bid and
s.rating=10 union select distinct s.sid from sailors s,reservesr,boats b where s.sid=r.sid and
r.bid=b.bid and r.bid=104;
```

SID

22

31

12. Find the names of Sailors who have not reserved a red boat

```
SQL> select s.sname from sailors s where s.sid not in(select r.sid from reserves r join boats b on
r.bid=b.bid where b.color='red');
```

SNAME

brutus

andy

rusty

zorba

horatio

art

Bob

WEEK – 5

Apply the concepts of Joins, SET operations and SQL functions on the following schema:

a) Employee:

| Name | Datatype | width | Constraint | Description |
|----------|----------|-------|-------------|-------------------|
| Empno | Integer | 4 | Primary Key | Employee Number |
| Ename | Varchar | 20 | | Employee Name |
| Job | Char | 12 | | Designation |
| Mgr | Integer | 4 | | Manager Number |
| Hiredate | Date | | | |
| Sal | Number | (8,2) | | Salary |
| Comm | Number | (6,2) | | Commission |
| Deptno | Integer | 2 | Foreign Key | Department Number |

b) Dept:

| Name | Datatype | width | Constraint | Description |
|--------|----------|-------|-------------|-------------------|
| Deptno | Integer | 2 | Primary Key | Department Number |
| Dname | Varchar | 12 | | Department Name |
| Loc | Char | 10 | | Location |

c) Salgrade:

| Name | Datatype | width | Constraint | Description |
|-------|----------|-------|------------|-------------|
| Grade | Integer | 1 | | Grade |
| Hisal | Integer | 4 | | Upper |
| Losal | Integer | 5 | | Lower |

Activate Windows
Go to Settings to activate Windows.

Joins (Combining Tables)

Joins unite records from two or more relational tables based on a relationship between specified columns.

- **INNER JOIN:** Returns only rows where there's a match in both tables.
- **OUTER JOIN:** Returns all rows from one table and matching rows from the other, using NULL for unmatched data.
 - **LEFT JOIN:** Prioritizes all rows from the left table.
 - **RIGHT JOIN:** Prioritizes all rows from the right table.
 - **FULL JOIN:** Returns all rows from both tables, whether they match or not.
- **CROSS JOIN:** Combines every row from the first table with every row from the second (Cartesian product).

SET Operations (Combining Result Sets)

SET operations merge the results of multiple SELECT queries. The queries must have the same number of columns with compatible data types.

- **UNION:** Combines results and removes duplicates.
- **UNION ALL:** Combines results and keeps all duplicates.
- **INTERSECT:** Returns only the rows that are common to both result sets.
- **MINUS (or EXCEPT):** Returns rows from the first query that are not present in the second.

SQL Functions (Data Processing)

SQL functions process input values to return a result.

Aggregate Functions (Group)

These functions operate on a set of rows and return a single summary value.

- **COUNT():** Gets the total number of rows.
- **SUM() / AVG():** Calculates the total sum or average of numeric values.
- **MAX() / MIN():** Finds the highest or lowest value in a set.

EMP TABLE STRUCTURE AND SAMPLE DATA

```
CREATE TABLE EMP (  
EMPNO INT PRIMARY KEY,  
ENAME VARCHAR(10),  
JOB VARCHAR(9),  
MGR INT,  
HIREDATE DATE,  
SAL DECIMAL(7,2),  
COMM DECIMAL(7,2),  
DEPTNO INT,  
FOREIGN KEY (DEPTNO) REFERENCES DEPT(DEPTNO)  
);
```

```
INSERT INTO EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)  
VALUES (7566, 'JONES', 'MANAGER', 7839, '1981-04-02', 2975, NULL, 20);
```

```
INSERT INTO EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)  
VALUES (7698, 'BLAKE', 'MANAGER', 7839, '1981-05-01', 2850, NULL, 30);
```

```
INSERT INTO EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)  
VALUES (7782, 'CLARK', 'MANAGER', 7839, '1981-06-09', 2450, NULL, 10);
```

```
INSERT INTO EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)  
VALUES (7788, 'SCOTT', 'ANALYST', 7566, '1987-04-19', 3000, NULL, 20);
```

```
INSERT INTO EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)  
VALUES (7902, 'FORD', 'ANALYST', 7566, '1981-12-03', 3000, NULL, 20);
```

```
INSERT INTO EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)  
VALUES (7844, 'TURNER', 'SALESMAN', 7698, '1981-09-08', 1500, 0, 30);
```

```
INSERT INTO EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)  
VALUES (7900, 'JAMES', 'CLERK', 7698, '1981-12-03', 950, NULL, 30);
```

```
INSERT INTO EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)  
VALUES (7654, 'MARTIN', 'SALESMAN', 7698, '1981-09-28', 1250, 1400, 30);
```

```
INSERT INTO EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)  
VALUES (7499, 'ALLEN', 'SALESMAN', 7698, '1981-02-20', 1600, 300, 30);
```

```
INSERT INTO EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)  
VALUES (7521, 'WARD', 'SALESMAN', 7698, '1981-02-22', 1250, 500, 30);
```

```
INSERT INTO EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
VALUES (7934, 'MILLER', 'CLERK', 7782, '1982-01-23', 1300, NULL, 10);
```

```
INSERT INTO EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
VALUES (7876, 'ADAMS', 'CLERK', 7788, '1987-05-23', 1100, NULL, 20);
```

```
INSERT INTO EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
VALUES (7658, 'SMITH', 'CLERK', 7902, '1980-12-17', 800, NULL, 20);
```

EMP TABLE QUERIES AND CONCISE OUTPUTS

1. Display all different job types.

```
SQL> SELECT DISTINCT JOB FROM EMP;
JOB
```

```
----
```

```
CLERK
```

```
SALESMAN
```

```
MANAGER
```

```
ANALYST
```

```
PRESIDENT
```

5 rows selected.

2. List the details of all employees in deptno 10 and 20 in alphabetical order.

```
SQL> SELECT * FROM EMP WHERE DEPTNO IN (10,20) ORDER BY
ENAME; EMPNO ENAME JOB SAL DEPTNO
```

```
-----
```

```
7782 CLARK MANAGER 2450 10
```

```
7839 KING PRESIDENT 5000 10
```

```
7934 MILLER CLERK 1300 10
```

```
7902 FORD ANALYST 3000 20
```

```
7566 JONES MANAGER 2975 20
```

```
7788 SCOTT ANALYST 3000 20
```

```
7369 SMITH CLERK 800 20
```

7 rows selected

3. List the names of employees who have “th” or “ll” in their names

```
SQL> SELECT ENAME FROM EMP WHERE ENAME LIKE '%TH%' OR ENAME LIKE
'%LL%';
```

```
ENAME
```

ALLEN

SMITH

MILLER

3 rows selected.

4. List the names, jobs and salaries of all employees who have a manager.

SQL> SELECT ENAME, JOB, SAL FROM EMP WHERE MGR IS NOT NULL;

| ENAME | JOB | SAL |
|-------|-----|-----|
|-------|-----|-----|

| | | |
|-------|-------|-----|
| SMITH | CLERK | 800 |
|-------|-------|-----|

| | | |
|-------|----------|------|
| ALLEN | SALESMAN | 1600 |
|-------|----------|------|

| | | |
|------|----------|------|
| WARD | SALESMAN | 1250 |
|------|----------|------|

| | | |
|-------|---------|------|
| JONES | MANAGER | 2975 |
|-------|---------|------|

| | | |
|--------|----------|------|
| MARTIN | SALESMAN | 1250 |
|--------|----------|------|

| | | |
|-------|---------|------|
| BLAKE | MANAGER | 2850 |
|-------|---------|------|

| | | |
|-------|---------|------|
| CLARK | MANAGER | 2450 |
|-------|---------|------|

| | | |
|-------|---------|------|
| SCOTT | ANALYST | 3000 |
|-------|---------|------|

| | | |
|--------|----------|------|
| TURNER | SALESMAN | 1500 |
|--------|----------|------|

| | | |
|-------|-------|-----|
| JAMES | CLERK | 950 |
|-------|-------|-----|

| | | |
|------|---------|------|
| FORD | ANALYST | 3000 |
|------|---------|------|

| | | |
|--------|-------|------|
| MILLER | CLERK | 1300 |
|--------|-------|------|

12 rows selected.

5. Give name remuneration of all employees.

SQL> SELECT ENAME, SAL + NVL(COMM,0) AS REMUNERATION FROM EMP;
ENAME REMUNERATION

| | |
|-------|-----|
| SMITH | 800 |
|-------|-----|

| | |
|--------|------|
| ALLEN | 1900 |
| WARD | 1750 |
| JONES | 2975 |
| MARTIN | 2650 |
| BLAKE | 2850 |
| CLARK | 2450 |
| SCOTT | 3000 |
| KING | 5000 |
| TURNER | 1500 |
| JAMES | 950 |
| FORD | 3000 |
| MILLER | 1300 |

13 rows selected.

6. List name and salary increased by 15% of all employees.

```
SELECT ENAME, SAL + 15 AS NEW_SALARY FROM EMP;
ENAME NEW_SALARY
```

| | |
|--------|---------|
| ALLEN | 1615.00 |
| WARD | 1265.00 |
| JONES | 2990.00 |
| MARTIN | 1265.00 |
| SMITH | 815.00 |
| BLAKE | 2865.00 |
| CLARK | 2465.00 |
| SCOTT | 3015.00 |
| KING | 5015.00 |
| TURNER | 1515.00 |
| ADAMS | 1115.00 |
| JAMES | 965.00 |
| FORD | 3015.00 |

MILLER 1315.00

7. Find all the employees who were hired during 1982.

```
SQL> SELECT ENAME, HIREDATE FROM EMP WHERE EXTRACT(YEAR FROM
HIREDATE)=1982;
ENAME HIREDATE
```

```
-----
MILLER 23-JAN-82
1 row selected.
```

8. Display name, annual salary, commission of all salesmen whose monthly salary is greater than commission.

```
SQL> SELECT ENAME, SAL*12 AS ANNUAL_SAL, COMM FROM EMP WHERE
JOB='SALESMAN' AND SAL > NVL(COMM,0);
ENAME ANNUAL_SAL COMM
```

```
ALLEN 19200 300
WARD 15000 500
TURNER18000 0
3 rows selected.
```

9. Produce the output as “smith has held the position of clerk in dept. 20 since 17-dec-80”.

```
SQL> SELECT LOWER(ENAME) || ' has held the position of ' || LOWER(JOB) || ' in dept. ' ||
DEPTNO || ' since ' || TO_CHAR(HIREDATE,'DD-MON-YY')
FROM EMP WHERE ENAME='SMITH';
```

```
LOWER(ENAME)||' HAS...'
-----
smith has held the position of clerk in dept. 20 since 17-DEC-80
1 row selected.
```

10. Find average salary and average total remuneration of all employees other than salesman.

```
SQL> SELECT ROUND(AVG(SAL),2) AS AVG_SAL, ROUND(AVG(SAL +
NVL(COMM,0)),2) AS AVG_TOTAL_REM FROM EMP WHERE JOB!='SALESMAN';
```

```
AVG_SAL AVG_TOTAL_REM
-----
```

2853.00 2853.00

1 row selected.

11. Find maximum, minimum and average salaries in each department.

```
SQL> SELECT DEPTNO, MAX(SAL) AS MAX_SAL, MIN(SAL) AS MIN_SAL,
ROUND(AVG(SAL),2) AS AVG_SAL FROM EMP GROUP BY DEPTNO;
DEPTNO MAX_SAL MIN_SAL AVG_SAL
```

10 5000 1300 2916.67

20 3000 800 2261.67

30 1600 950 1275.00

3 rows selected.

12. Find the maximum, minimum and average salaries in each job.

```
SQL> SELECT JOB, MAX(SAL), MIN(SAL), ROUND(AVG(SAL),2) FROM EMP GROUP
BY JOB;
```

```
JOB      MAX MIN AVG
```

MANAGER 2975 2450 2758.33

ANALYST 3000 3000 3000.00

CLERK 1300 800 1016.67

SALESMAN 1600 1250 1400.00

PRESIDENT 5000 5000 5000.00

5 rows selected.

13. Find the departments which have more than three employees.

```
SQL> SELECT DEPTNO, COUNT(*) AS CNT FROM EMP GROUP BY DEPTNO HAVING
COUNT(*)>3;
```

```
DEPTNO CNT
```

30 5

1 row selected.

14. Display employee names and their respective department numbers.

```
SQL> SELECT ENAME, DEPTNO FROM EMP;
```

```
ENAME DEPTNO
```

---- ----

SMITH 20

ALLEN 30

13 rows selected.

15. Give the salary grades for all the employees.

SQL> -- Assuming salary_grade table or using CASE for illustration

```
SQL> SELECT ENAME, SAL,  
CASE WHEN SAL<=1000 THEN 1 WHEN SAL<=2000 THEN 2 WHEN SAL<=3000 THEN  
3 ELSE 4 END AS SAL_GRADE FROM EMP;
```

```
ENAME SAL SAL_GRADE  
SMITH 800 1  
ALLEN1600 2
```

13 rows selected.

16. Display the employee names who earn highest salary in each job.

```
SQL> SELECT ENAME, JOB, SAL FROM EMP WHERE (JOB, SAL) IN (SELECT JOB,  
MAX(SAL) FROM EMP GROUP BY JOB);  
ENAME JOB SAL
```

```
-----  
KING PRESIDENT 5000  
SCOTT ANALYST 3000  
MILLER CLERK 1300  
ALLEN SALESMAN 1600
```

4 rows selected.

17. Find the employee details whose salary is greater than blake's salary.

```
SQL> SELECT * FROM EMP WHERE SAL > (SELECT SAL FROM EMP WHERE  
ENAME='BLAKE');
```

```
EMPNO ENAME JOB    SAL  
-----  
7839 KING PRESIDENT 5000  
7788 SCOTT ANALYST 3000  
7902 FORD ANALYST 3000
```

3 rows selected.

18. Find employee details of employees who have the same job and salary as that of Scott.

```
SQL> SELECT * FROM EMP WHERE JOB = (SELECT JOB FROM EMP WHERE  
ENAME='SCOTT') AND SAL = (SELECT SAL FROM EMP WHERE ENAME='SCOTT');
```

-- Only SCOTT matches in this dataset

```
EMPNO ENAME JOB SAL  
7788 SCOTT ANALYST 3000
```

1 row selected.

19. Display the maximum salaries in accounting and research department.

```
SQL> -- Assuming dept 10=ACCOUNTING, dept 20=RESEARCH
```

```
SQL> SELECT DEPTNO, MAX(SAL) FROM EMP WHERE DEPTNO IN (10,20) GROUP  
BY DEPTNO;
```

```
DEPTNO MAX(SAL)
```

```
10    5000
```

```
20    3000
```

2 rows selected.

20. Display salary grades of all employees except of those employees whose salary grade is 3 and 4.

```
SQL> SELECT ENAME, SAL,  
CASE WHEN SAL<=1000 THEN 1 WHEN SAL<=2000 THEN 2 WHEN SAL<=3000 THEN  
3 ELSE 4 END AS SAL_GRADE  
FROM EMP WHERE CASE WHEN SAL<=1000 THEN 1 WHEN SAL<=2000 THEN 2  
WHEN SAL<=3000 THEN 3 ELSE 4 END NOT IN (3,4);
```

```
ENAME SAL SAL_GRADE
```

```
SMITH 800 1
```

```
ALLEN 1600 2
```

```
WARD 1250 2
```

6 rows selected.

21. Give the names and salaries of the employees whose salary is maximum in their respective departments.

```
SQL> SELECT ENAME, SAL FROM EMP e WHERE SAL = (SELECT MAX(SAL) FROM  
EMP WHERE DEPTNO = e.DEPTNO);
```

```
ENAME SAL
```

```
KING 5000
```

FORD 3000
ALLEN1600

3 rows selected.

22. List the employees whose salary is greater than the salaries of all employees who are working as salesman.

SQL> SELECT ENAME, SAL FROM EMP WHERE SAL > ALL (SELECT SAL FROM EMP WHERE JOB='SALESMAN');

ENAME SAL
KING 5000
SCOTT 3000
FORD 3000

3 rows selected.

23. Write a query which will return the day of the week entered in the format of sysdate.

SQL> SELECT TO_CHAR(TO_DATE('21-07-2025','DD-MM-YYYY'),'DAY') FROM DUAL;

TO_CHAR(...)

MONDAY

1 row selected.

24. Find the difference between highest and lowest salaries

SQL> SELECT MAX(SAL)-MIN(SAL) AS SAL_DIFF FROM EMP;

SAL_DIFF

4200

1 row selected.

25. Generate the output as smith – clerk.

SQL> SELECT LOWER(ENAME) || ' - ' || LOWER(JOB) FROM EMP WHERE ENAME='SMITH';

```
LOWER(ENAME)||' - '||LOWER(JOB)
```

```
-----
```

```
smith – clerk
```

```
1 row selected.
```