--- CREATE TABLE

CREATE TABLE chethan

(

empid NUMBER(10) PRIMARY KEY,

empname VARCHAR2(35),

empdob DATE,

empsalary NUMBER(12),

empgender CHAR(1),

emplocation VARCHAR2(35)

);

--- INSERT DATA INTO TABLE.

INSERT INTO chethan VALUES(1,'CHET','25-03-2000',6000,'M','BNG');

--OR

INSERT INTO chethan VALUES(2,'CHARY',NULL,8000,'M',NULL);--NULL SULD NOT BE IN COLAN.

--OR

INSERT INTO chethan(empid,empname,empsalary,empgender) VALUES(3,'CHARU',9000,'F');--HERE ONLY REQUIRED COLUMNS ARE ADDED.

--- INSERTG MULTIPLE DATA INTO TABLE.

SELECT \* FROM chethan;

DESC CHETHAN;

INSERT ALL

INTO chethan VALUES(1,'CHET','25-03-2000',6000,'M','BNG',600)

INTO chethan VALUES(2,'CHARY',NULL,8000,'M',NULL,800)

INTO chethan VALUES(3,'RAMU',NULL,9000,'M','MUM',900)

SELECT \* FROM DUAL;

---RENAME AND MODIFY A COLUMN USING ALTER.

ALTER TABLE chethan

RENAME COLUMN emplocation TO EMPPLACE;

ALTER TABLE chethan

MODIFY EMPPLACE VARCHAR2(32);

--- ADD,DROP AND UPDATE FROM ALTER.

ALTER TABLE chethan2

RENAME TO chethan3;

ALTER TABLE chethan

ADD EMPDEPARTMENT VARCHAR2(30);

ALTER TABLE chethan

DROP COLUMN EMPDEPARTMENT;

ALTER TABLE chethan

ADD EMPPF NUMBER(10);

UPDATE chethan

SET emppf=empsalary\*2/100;-- IT WILL UPDATE TO FULL TABLE.

UPDATE chethan

SET emppf=empsalary\*3/100

WHERE EMPID=2;-- BY USING WHERE CLAUSE IT WILL UPDATE TO ONLY EMPID=2.

UPDATE chethan

SET empdob='15-05-2005',empplace='MUM'

WHERE EMPID=2;-- BY USING WHERE CLAUSE IT WILL UPDATE MULTIPLE COLUMN OF SINGLE ROW(EMPID=2) (IT IS NOT POSSIBLE TO UPDATE MULTIPLE ROWS).

---DUPLICATE A TABLE WITH RECORDS AND WITHOUT RECORDS.

CREATE TABLE chethan1 AS SELECT \* FROM chethan;--IT WILL COPY BOTH TABLE STRUCTURE AND RECORDS.

CREATE TABLE chethan2 AS SELECT \* FROM chethan

WHERE 1=2;--IT WILL COPY ONLY TABLE STRUCTURE NOT RECORDS.

---DELETE,ROLLBACK AND TRUNCATE TABLE DATA.

DELETE FROM chethan;--IT WILL DELETE ALL THE DATA ONLY NOT TABLE STRUCTURE.

DELETE FROM chethan

WHERE EMPID=3;--IT WILL DELETE ONLY EMPID=3 RECORDS.

ROLLBACK;--IT WILL UNDO DELETED RECORDS.

TRUNCATE TABLE chethan;--ONCE WE TRUNCATE TABLE WE CANNOT ROLL BACK AGAIN.

---ADD COMMENT ON A TABLE AND COLUMN ALSO REMOVE COMMENT.

COMMENT ON TABLE chethan

IS 'Employee Information';--IT IS USED TO COMMENT ON TABLE.

SELECT \* FROM USER\_TAB\_COMMENTS; -- IT WILL SHOWS COMMENT OF ALL THE TABLE.

SELECT \* FROM USER\_TAB\_COMMENTS

WHERE TABLE\_NAME = 'CHETHAN'; -- IT WILL SHOWS COMMENT OF SELECTED TABLE(TABLE NAME SHOULD BE IN CAPS OR SMALL LETTER AS IN TABLE).

COMMENT ON TABLE chethan

IS '';--IT IS USED TO REMOVE COMMENT FROM TABLE.

COMMENT ON COLUMN chethan.empid

IS 'Employee Identitiy Number'; --IT IS USED TO COMMENT ON COLUMNS OF A TABLE.

SELECT \* FROM USER\_COL\_COMMENTS

WHERE TABLE\_NAME='CHETHAN';--IT IS USED TO VIEW COMMENTS ON COLUMNS OF PARTICULAR TABLE.

COMMENT ON COLUMN chethan.empdob

IS 'Employee Date Of Birth ';

COMMENT ON COLUMN chethan.empdob

IS ' '; -- IT IS USED TO REMOVE COMMENTS ON COLUMNS OF PARTICULAR TABLE.

--COMPARISION(>,<,=,!=,<>) OPERATORS.

SELECT \* FROM HR.employees

WHERE salary=8000;

SELECT \* FROM HR.employees

WHERE salary<>8000; --'<>' MEANS NOT EQUAL TO.

SELECT \* FROM HR.employees

WHERE salary!=8000; --'!=' MEANS NOT EQUAL TO.

SELECT \* FROM HR.employees

WHERE salary < 8000;

SELECT \* FROM HR.employees

WHERE salary <= 8000;

SELECT \* FROM HR.employees

WHERE salary > 8000;

SELECT \* FROM HR.employees

WHERE salary >= 8000;

---'BETWEEN....AND' OPERATOR.

--THE 'BETWEEN....AND' OPERATOR SELECTS VALUES WITIN A GIVEN RANGE. THE VALUES CAN BE NUMBERS OR DATE.

--THE 'BETWEEN.....AND' OPERATOR IS INCLUSIVE:BEGIN ANS END VALUES ARE INCLUDED.

SELECT \* FROM HR.employees

WHERE salary BETWEEN 10000 AND 15000;

--OR

SELECT \* FROM HR.employees

WHERE salary >= 10000 AND salary <= 15000;

SELECT \* FROM HR.employees

WHERE salary NOT BETWEEN 10000 AND 15000;

--OR

SELECT \* FROM HR.employees

WHERE salary < 10000 OR salary > 15000

ORDER BY salary DESC;

SELECT \* FROM HR.employees

WHERE hire\_date BETWEEN '17-06-03' AND '24-03-05'

ORDER BY hire\_date DESC;

---'LIKE' OPERATOR.

--THE LIKE OPERATOR IS USED IN A 'WHERE' CLAUSE TO SEARCH FOR A SPECIFIC PATTERN IN A COLUMN.

--THERE ARE TWO WILDCARD USED IN CONJUNCTION WITH THE 'LIKE' OPERATOR.

--THE PERCENTAGE SIGN(%) REPRESENTS ZERO, ONE OR MULTIPLE CHARACTERS.

--THE UNDERSCORE (\_) REPRESENTS A SINGLE CHARACTER.

SELECT \* FROM HR.employees -- IT GIVES ALL THE WORDS STARTS WITH 'V'.

WHERE email LIKE 'V%'; -- WE SHOULD GIVE CAPS 'V' BECAUSE ALL EMAILS IN TABLE ARE CAPITAL LETTER ONLY.

SELECT \* FROM HR.employees

WHERE email LIKE '%VA%'; -- IT GIVES ALL THE WORDS CONTAINS 'VA' LETTERS.

SELECT \* FROM HR.employees

WHERE email LIKE '%H'; --IT GIVES ALL THE WORDS END WITH 'H'.

SELECT \* FROM HR.employees

WHERE email LIKE '\_\_\_\_\_'; --IT GIVES ALL THE EMAIL HAVING '5' LETTERS.

SELECT \* FROM HR.employees

WHERE email LIKE '\_AU%'; --IT GIVES EMAILS HAVING FIRST LETTER ANYTHING SECOND AND THIRD LETTER 'AC' AFTER ANY LETTERS.

SELECT \* FROM HR.employees

WHERE email LIKE 'N%E'; -- IT GIVES ALL EMAILS LEETER STARTS WITH 'N' AND ENDS WITH 'E'.

--'IS NULL' AND 'IS NOT NULL'.

--A NULL IN A TABLE IS A VALUE IN A FIELD THAT APPEARS TO BE BLANK.

--IF A FIELD IN A TABLE IS OPERATIONAL, IT IS POSSIBLE TO UPDATE A RECORD OR INSERT A NEW RECOD WITHOUT ADDING A VALUE TO THIS FIELD. THEN THE FIELD WILL BE SAVED WITH A NULL VALUE.

--NULL VALUE IS DIFFERENT FROM A ZERO OR A FIELD THAT CONTAINS SPACE.

--IT IS NOT POSSIBLE TO TEST FOR NULL VALUES WITH COMPARISON OPERATOR, SUCH AS =,<,>.

--WE HAVE TO USE THE 'IS NULL' AND 'IS NOT NULL' OPERATOR INSTED.

SELECT \* FROM chethan

WHERE empdob IS NULL; -- IT WILL GIVES ALL THE RESULTS HAVING 'empdob' IS NULL.

SELECT \* FROM chethan

WHERE empdob IS NOT NULL; -- IT WILL GIVES ALL THE RESULTS HAVING 'empdob' IS NOT NULL.

SELECT \* FROM chethan

WHERE empdob = NULL; -- IT WILL NOT GIVE ANY RESULT BECAUSE 'NULL' WONT WORK WITH COMPARISION OPERATOR.

--- 'IN' OPERATOR.

--THE 'IN' OPERATOR ALLOWS YOU TO SPECIFY MULTIPLE VALUES IN A 'WHERE' CLAUSE.

-- THE 'IN' OPERATOR IS A SHORTHAND FOR MULTIPLE 'OR' CONDITION.

SELECT \* FROM HR.employees

WHERE employee\_id IN(101,107,110);

--OR

SELECT \* FROM HR.employees

where employee\_id = 101 OR employee\_id = 107 OR employee\_id = 110;

SELECT \* FROM HR.employees

WHERE employee\_id NOT IN(101,107,110);

--OR

SELECT \* FROM HR.employees

where employee\_id <> 101 AND employee\_id <> 107 AND employee\_id <> 110;

UPDATE HR.employees

SET salary =15000

WHERE employee\_id IN (101,107,110);

--- LOGICAL OPERATORS 'AND','OR','NOT'.

--'AND' OPERATOR RETURNS TRUE WHEN BOTH CONDITION ARE TRUE.

-- 'AND' OPERATOR RETURNS FALSE WHEN EITHER OF THE CONDITION IS FALSE.

-- 'OR' OPERATOR RETURNS TRUE WHEN EITHER OF THE CONDITION IS TRUE.

--'OR' OPERATOR RETURNS FALSE WHEN BOTH ARE FALSE.

--'NOT' OPERATOR RETURNS TRUE WHEN THE CONDITION IS FALSE.

--'NOT' OPERATOR RETURNS FALSE WHEN THE CONDITION IS TRUE.

SELECT \* FROM HR.employees

WHERE salary = 9000 AND job\_id = 'IT\_PROG';

SELECT \* FROM HR.employees

WHERE salary >= 9000 AND salary <= 15000;

SELECT \* FROM HR.employees

WHERE salary = 9000 OR job\_id = 'IT\_PROG';

SELECT \* FROM HR.employees

WHERE salary < 9000 OR salary > 15000;

SELECT \* FROM HR.employees

WHERE NOT job\_id = 'IT\_PROG';

SELECT \* FROM HR.employees

WHERE email NOT LIKE 'J%';

---ARITHMETIC OPERATORS.

SELECT +3 FROM DUAL; -- 'DUAL' MEANS DUMMY COLUMN WHICH IS USED FOR THOSE DATA DONT HAVE TABLE.

SELECT -5 FROM DUAL;

SELECT SYSDATE FROM DUAL;

SELECT salary/100 FROM HR.employees;

SELECT FIRST\_NAME,MONTHS\_BETWEEN(TRUNC(SYSDATE),hire\_date) /12

AS AGE FROM hr.employees;

SELECT FIRST\_NAME, salary \* 12

AS Annual\_Salary FROM hr.employees;

SELECT FIRST\_NAME, (salary +500)

FROM hr.employees;

SELECT FIRST\_NAME,hire\_date, (hire\_date +50)

FROM hr.employees;

--- CONCATENATION (CONCAT) OPERATION.

SELECT 'Employee First Name is ' || first\_name

FROM hr.employees;

SELECT first\_name || 'is from ' || job\_id

FROM hr.employees;

SELECT CONCAT(first\_name,' salary is ') || salary

FROM hr.employees;

SELECT CONCAT(CONCAT(first\_name,' salary is '),salary)

FROM hr.employees;

--- 'ALIAS' OPERATOR.

SELECT first\_name,(salary \* 12) AS "Annual\_Salary"

FROM hr.employees;

SELECT first\_name AS "Employee\_Name", salary

FROM hr.employees;

SELECT B.COMPANY, c.emppf

FROM BSEIT B,chethan C; -- IT JOINS THE TABLES.

SELECT B.COMPANY, B1.SECTOR, B1.PRICE

FROM BSEIT B,BSEIT1 B1; -- IT JOINS THE TABLES.

---'ORDER BY' CLAUSE.

SELECT \* FROM hr.employees

ORDER BY department\_id ASC;

SELECT \* FROM hr.employees

ORDER BY first\_name DESC;

SELECT \* FROM hr.employees

WHERE salary > 8000

ORDER BY first\_name ASC;

SELECT \* FROM hr.employees

ORDER BY first\_name DESC , last\_name ASC;

SELECT first\_name,(salary \* 12) AS "Annual\_Salary"

FROM hr.employees

ORDER BY "Annual\_Salary" DESC; -- WE USED ALIAS "Annual\_Salary" in 'ORDER BY' CLAUSE.

----"SET OPERATORS".

--SET OPERATOR COMBINE THE RESULTS OF TWO OR MORE QUERIES INTO SINGLE RESULT.

--QUERIES CONTAINING 'SET' OPERATORS ARE CALLED COMPOUND QUERIES.

--'UNION' OPERATORS.

--THE UNION OPERATOR RETURNS ALL ROWS SELECTED BY EITHER QUERY.

--USE THE 'UNION' OPERATORS TO RETURNS ALL ROWS FROM MULTIPLE TABLES AND ELIMINATE ANY DUPLICATE ROWS.

--THE NUMBER OF COLUMNS AND THE TYPES OF THE COLUMNS BEING SELECTED SHOULD BE "IDENTICAL" IN ALL THE 'SELECT' STATEMENTS USED IN THE QUERY.

--THE NAMES OF THE COLUMNS NEED NOT BE "IDENTICAL".

CREATE TABLE FOOTBALL

(

TNO NUMBER(10) PRIMARY KEY,

NAME VARCHAR2(10),

SECTION VARCHAR2(20)

);

INSERT INTO FOOTBALL VALUES('1','KING','SECTIONA')

INSERT INTO FOOTBALL VALUES('2','PETER','SECTIONB')

INSERT INTO FOOTBALL VALUES('3','JACOB','SECTIONC')

SELECT \* FROM FOOTBALL;

CREATE TABLE HOCKEY

(

TNO NUMBER(10) PRIMARY KEY,

NAME VARCHAR2(10),

SECTION VARCHAR2(20)

);

INSERT INTO HOCKEY VALUES('1','KING','SECTIONA')

INSERT INTO HOCKEY VALUES('3','JACOB','SECTIONC')

INSERT INTO HOCKEY VALUES('4','ANNISON','SECTIONA')

SELECT \* FROM HOCKEY;

SELECT \* FROM FOOTBALL

UNION

SELECT \* FROM HOCKEY; -- IT REMOVE DUPLICATE ROWS AND JOINS UNIC ROWS.

---'UNION ALL' OPERATOR.

--THE 'UNION ALL' OPERATOR RETURNS ALL ROWS FORM MULTIPLE TABLES AND IT WILL NOT ELIMINATE ANY DUPLICATE ROWS..

SELECT \* FROM FOOTBALL

UNION ALL

SELECT \* FROM HOCKEY; -- IT WILL NOT REMOVE DUPLICATE ROWS AND JOINS ALL ROWS.

SELECT \* FROM FOOTBALL

UNION

SELECT \* FROM chethan; --IT WONT GIVE RESULT BESAUSE THE NUMBER OF COLUMNS AND THE TYPES OF THE COLUMNS(NUMBER,VARCHAR2 ETC) ARE NOT SAME.

SELECT tno,name,section FROM FOOTBALL --HERE TNO AND EMPID ARE IN 'NUMBER', NAME AND EMPNAME ARE IN VARCHAR2, SECTION AND EMPPLACE ARE IN VARCHAR2.

UNION

SELECT EMPID,EMPNAME,EMPPLACE FROM chethan; --IT WILL GIVE RESULT BESAUSE THE NUMBER OF COLUMNS AND THE TYPES OF THE COLUMNS ARE SAME.

---'INTERSECT' OPERATORS.

-- THE 'INTERSECT' OPERATOR RETURNS ALL ROWS COMMON TO MULTIPLE QUERIES.

-- THE NUMBER OF COLUMN AND THE DATATYPE OF THE COLUMNS BEING SELECTED BY THE 'SELECT' STATEMENT IN THE QUERIES MUST BE IDENTICAL IN ALL THE 'SELECT' STATEMENTS USED IN THE QUERY.

--THE NAMES OF THE COLUMNS NEED NOT BE IDENTICAL.

SELECT \* FROM FOOTBALL

INTERSECT

SELECT \* FROM HOCKEY; --IT WILL JOIN ALL ROWS OF COMMON DATA ONLY.

---'MINUS' OPERATORS.

--THE 'MINUS' OPERATOR RETURNS ROW RETURNED BY THE FIRST QUERY THAT ARE NOT PRESENT IN THE SECOND QUERY (THE FIRST 'SELECT' STATEMENT 'MINUS' THE SECOND 'SELECT' STATEMENT.

--THE NUMBER OF COLUMNS AND THE DATATYPE OF THE COLUMNS BEING SELECTED BY THE 'SELECT' STATEMENT IN THE QUERIES MUST BE IDENTICAL IN ALL THE 'SELECT' STATEMENT USED IN THE QUERY.

-- THE NAMES OF THE COLUMNS NO NEED TO BE IDENTICAL.

SELECT \* FROM FOOTBALL

MINUS

SELECT \* FROM HOCKEY;--IT WILL ONLY GIVE RESULTS THAT ARE PRESENT IN FIRST QUERY AND NOT PRESENT IN SECOND QUERY.

---CASE MANIPULATION FUNCTIONS(UPPER,LOWER AND INITCAP).

SELECT UPPER('chethan') FROM DUAL;

SELECT UPPER(first\_name)

FROM hr.employees;

SELECT LOWER(email)

FROM hr.employees;

SELECT INITCAP(email)

FROM hr.employees; --'INITCAP' LOOKS NOT RECOGNISED KEY WORD(COLOUR IS NOT BLUE) BUT WORKS FINE.

SELECT UPPER(first\_name),salary

FROM hr.employees

WHERE first\_name = 'David';

---CHARACTER MANIPULATION FUMNCTION

--'INSTR()' FUNCTION.

--'INSTR' FINDS NUMERIC POSITION OF THE FIRST OCCURANCR OF A NAME CHARACTER.

--'INSTR' PERFORMS A CASE-SENSITIVE SEARCH.

SELECT INSTR('Oracle Tutorial','a') FROM DUAL;--THE LETTER 'a' IS IN THIRD PLACE IN THE WORD 'Oracle' COUNT START POSITION IS 1st LETTER.

--OR

SELECT INSTR('Oracle Tutorial','a',1,1) FROM DUAL;--COUNT STATRS FROM LETTER 1('O') AND FIRST LETTER 'a' is in 3rd place.

SELECT INSTR('Oracle Tutorial','A') FROM DUAL; -- NO RESULT BECAUSE THERE IS NO UPPER CASE LETTER 'A'.

SELECT INSTR('Oracle Tutorial','a',6) FROM DUAL;

SELECT INSTR('Oracle Tutorial','a',4) FROM DUAL;--COUNT STARTS FROM LETTER 4th('c') AND POSITION OF 'a' IS IN 14th(include space also) PLACE.

SELECT INSTR('Oracle Tutorial','a',1,2) FROM DUAL;--COUNT STARTS FROM LETTER 1st('o') AND 2nd 'a'IS IN 14th PLACE.

SELECT INSTR('Oracle Tutorial','a',4,2) FROM DUAL;--NO RESULT BECAUSE COUNT STARTS FROM LETTER 4th AND ONLY ONE 'a' AFTER LETTER 'c'.

SELECT INSTR('Oracle Tutorial','a',-4) FROM DUAL;--WE ARE ASIGNED TO SEARCH '-4' SO MINUS SIGN INDICATES COUNT STARTS FROM END.

SELECT INSTR('Oracle Tutorial','a',-1) FROM DUAL;

SELECT INSTR('Oracle Tutorial','a',-1,2) FROM DUAL;

SELECT first\_name, INSTR(first\_name,'a')

FROM hr.employees;

SELECT INSTR('Oracle Tutorial For Beginners','e',7,3) FROM DUAL;--NO RESULT BECAUSE NO 'e' AFTER 7th PLACE.

---'SUBSTR' FUNCTION.

--SYNTAX = SUBSTR(string,start\_position(lenght)).

SELECT SUBSTR('Oracle Tutorial',1) FROM DUAL;

SELECT SUBSTR('Oracle Tutorial',2) FROM DUAL;

SELECT SUBSTR('Oracle Tutorial',0) FROM DUAL;--IT WILL CONSIDER START POSITION AS 1.

SELECT SUBSTR('Oracle Tutorial',-1) FROM DUAL;--IT SHOWS LAST LETTR 'L' ONLY BECAUSE FOR MINUS INDICATE COUNT STARTS FROM END.

SELECT SUBSTR('Oracle Tutorial',-8) FROM DUAL;--START POSITION IS 8th LETTER FROM END BECAUSE '-8'.

SELECT SUBSTR('Oracle Tutorial',1,6) FROM DUAL;--STARTS POSITION FIRST LETTER AND STRING LENGTH IS 6 LETTER..

SELECT SUBSTR('Oracle Tutorial',1,-1) FROM DUAL;-- NO RESULT BECAUSE '-1' MEANS NO LENGTH.

SELECT SUBSTR('Oracle Tutorial',-9,5) FROM DUAL;

SELECT first\_name,SUBSTR(first\_name,1,3)FROM hr.employees;

SELECT first\_name,SUBSTR(first\_name,3,3)FROM hr.employees; --START POSITION IS 3rd LETTER AND LENGHT 3 LETTER

SELECT SUBSTR('Oracle Tutorial For Beginners',21,5) FROM DUAL;

---'LPAD' AND 'RPAD' FUNCTION.

--'LPAD' FUNCTION PADS THE LEFT SIDE OF A STRING WITH A SPECIFIC SET OF CHARACTERS.

--'RPAD' FUNCTION PADS THE RIGHT SIDE OF A STRING WITH A SPECIFIC SET OF CHARACTERS.

--LPAD(string,padded\_length(pad\_string))

SELECT LPAD('Oracle',3,'@') FROM DUAL;

SELECT RPAD('Oracle',3,'@') FROM DUAL;

SELECT LPAD('Oracle',10,'@') FROM DUAL;

SELECT RPAD('Oracle',10,'@') FROM DUAL;

SELECT RPAD('Oracle',-1,'@') FROM DUAL;

SELECT RPAD('Oracle',10,'$') FROM DUAL;

SELECT RPAD('Oracle',10,'CY') FROM DUAL;

SELECT first\_name,LPAD(salary,7,'$') FROM hr.employees;

---'TRIM','LTRIM','RTRIM' FUNCTIONS.

--'TRIM' SYNTAX = TRIM(((LEADING,TRAILING,BOTH)trim\_character FROM)string).

SELECT TRIM(LEADING '0' FROM '0005489') FROM DUAL;

SELECT TRIM(LEADING '0' FROM '0005489') FROM DUAL;

SELECT TRIM(TRAILING '0' FROM '5478000') FROM DUAL;

SELECT TRIM(BOTH '0' FROM '0005478000') FROM DUAL;

SELECT TRIM('0' FROM '0000345000') FROM DUAL;

--'LTRIM' SYNTAX = LTRIM(string(trim\_string)).

SELECT LTRIM('oracle tutorial') FROM DUAL;

SELECT LTRIM('000354','0') FROM DUAL;

SELECT LTRIM('000354','03') FROM DUAL;

SELECT LTRIM('wohoracle','who') FROM DUAL;

---'RTRIM'(string(trim\_string)).

SELECT RTRIM('oracle tutorial') FROM DUAL;

SELECT RTRIM('354000','0') FROM DUAL;

SELECT RTRIM('354300','03') FROM DUAL;

SELECT RTRIM('oraclewoh','who') FROM DUAL;

SELECT first\_name, LTRIM(first\_name, 'Su')

FROM hr.employees;

---'REPLACE' FUNCTION.

--SYNTAX = REPLACE(string,string\_to\_replace(replacement\_string)).

SELECT REPLACE('SQL Tutorial', 'SQL')FROM DUAL;--REMOVES ONLY 'SQL' STRING BECAUSE WE DONT SPECIFY PROPERLY.

SELECT REPLACE('456Tutorial', '456')FROM DUAL;

SELECT REPLACE('456Tutorial456', '456')FROM DUAL;

SELECT REPLACE('SQL Tutorial', 'SQL','Oracle')FROM DUAL;-- HERE IT REPLACES 'SQL' FROM 'Oracle'.

SELECT REPLACE('456Tutorial', '456','Oracle')FROM DUAL;-- HERE IT REPLACES '456' FROM 'Oracle'.

SELECT REPLACE('456456Tutorial', '456','Oracle')FROM DUAL;-- HERE IT REPLACES '456' FROM 'Oracle' TWO TIMES.

SELECT employee\_id,first\_name,REPLACE(first\_name,'Valli','Dalli') FROM hr.employees

---'DUAL' TABLE.

--DUAL TABLE IS A SPECIAL TABLE WITH ONE ROW AND ONE COLUMN.

--'DUAL' TABLE IS PRESENT IN ALL ORACLE DATABASES BY DEFAULT.

--'DUAL' TABLE HAS ONE COLUMN NAMED 'DUMMY' WITH DATA TYPE VARCHAR2(1).

--THE COLUMN VALUE IS 'X'.

DESC DUAL;

SELECT \* FROM DUAL;

SELECT COUNT(\*) FROM DUAL;

SELECT 10+5 FROM DUAL;

SELECT 105 FROM DUAL;

SELECT 1050/15 FROM DUAL;

SELECT 1050\*15 FROM DUAL;

SELECT 'Oracle Tutoial' FROM DUAL;

SELECT SYSDATE FROM DUAL;

---'ROUND' FUNCTIN.

--ROUND RETURNS A NUMBER ROUNDED TO THE NEAREST INTEGER.

--SYNTAX = ROUND(number (decimal\_place)).

SELECT ROUND(123.56)FROM DUAL;--WE ARE NOT SPECIFIED SCALE SO IT ROUND ALL.

SELECT ROUND(123.564,1)FROM DUAL;--HERE IT ROUND FIRST DIGIT (56 TO 60)

SELECT ROUND(123.564,2)FROM DUAL;--HERE IT ROUND SECOND DIGIT (64 TO 60)

SELECT ROUND(123.567,2)FROM DUAL;--HERE IT ROUND SECOND DIGIT (67 TO 70)

SELECT ROUND(123.567,-1)FROM DUAL;--HERE IT ROUND ONES PLACE(ONCE PLACE DEPENDS ON TENS PLACE ROUND TO TENS PLACE) BEFORE DECIMAL (23 TO 20)

SELECT ROUND(126.567,-1)FROM DUAL;--HERE IT ROUND FIRST DIGIT BEFORE DECIMAL POINT (26 TO 30)

SELECT ROUND(123.567,-2)FROM DUAL;--HERE IT ROUND TENS PLACE(TENS PLACE DEPENDS ON HUNDRED PLACE ROUND TO HUNDRED PLACE) BEFORE DECIMAL (123 TO 100)

SELECT ROUND(188.567,-2)FROM DUAL;--HERE IT ROUND TENS PLACE(TENS PLACE DEPENDS ON HUNDRED PLACE ROUND TO HUNDRED PLACE) BEFORE DECIMAL (188 TO 200)

SELECT ROUND(188.567,-3)FROM DUAL;

SELECT ROUND(888.567,-3)FROM DUAL;

---'TRUNC' AND 'MOD' FUNCTION.

--'TRUNC' SYNTAX = TRUNC(number(decimal\_place)).

SELECT TRUNC(345.1234) FROM DUAL;--IT WILL HIDE ALL DECIMAL NUMBERS.

SELECT TRUNC(345.1234,1) FROM DUAL;--IT WILL HIDE ALL DECIALS EXCEPT FIRST DECIMAL.

SELECT TRUNC(345.1234,2) FROM DUAL;--IT WILL HIDE ALL DECIALS EXCEPT FIRST AND SECOND DECIMAL.

SELECT TRUNC(345.1234,-1) FROM DUAL;--IT WILL HIDE FIRST DEGIT BEFOR DECIMAL (HERE 5 AS 0)

SELECT TRUNC(345.1234,-2) FROM DUAL;--IT WILL HIDE FIRST AND SECOND DEGIT BEFOR DECIMAL (HERE 5 AND 4 AS 0)

SELECT TRUNC(345.1234,-3) FROM DUAL;--IT WILL HIDE ALL DEGIT BEFOR DECIMAL (HERE 3,5 AND 4 AS 0)

--'MOD' FUNCTION.

--'MOD' SYNTAX = MOD(m,n)

SELECT MOD(20,3) FROM DUAL;--IT GIVES 'REMINDER' (20/3 = REMINDER 2)

SELECT MOD(20,0) FROM DUAL;

SELECT MOD(11.5,2) FROM DUAL;

SELECT MOD(-11.5,2) FROM DUAL;

SELECT salary, MOD(salary,1000) FROM hr.employees

---DATE FUNCTIONS 'MONTHS\_BETWEEN' AND 'ADD\_MONTHS' FUNCTION.

--'MONTHS\_BETWEEN' SYNTAX = MONTHS\_BETWEEN(date1,date2)

SELECT MONTHS\_BETWEEN('15/07/2018','15/05/2018') FROM DUAL;

SELECT MONTHS\_BETWEEN(TO\_DATE('15/JULY/2018','DD/MM/YYYY'), TO\_DATE('15/MAY/2018','DD/MM/YYYY')) FROM DUAL;

SELECT MONTHS\_BETWEEN('16/7/2018','15/5/2018')FROM DUAL;

SELECT MONTHS\_BETWEEN('30/7/2018','15/5/2018')FROM DUAL;

SELECT MONTHS\_BETWEEN('15/5/2018','15/5/2018')FROM DUAL;

---'ADD\_MONTHS' FUNCTION.

--'ADD\_MONTHS' SYNTAX = ADD\_MONTHS(date,Number\_of\_Months).

SELECT ADD\_MONTHS('15/1/2018',1) FROM DUAL;

SELECT ADD\_MONTHS('15/1/2018',2) FROM DUAL;

SELECT ADD\_MONTHS('15/1/2018',12) FROM DUAL;

SELECT ADD\_MONTHS('15/1/2018',-1) FROM DUAL;

SELECT ADD\_MONTHS('15/1/2018',-11) FROM DUAL;

--- 'NEXT\_DAY' FUNCTION.

--'NEXT\_DAY' SYNTAX = NEXT\_DAY(Date,Weekday).

SELECT SYSDATE FROM DUAL; -- IT GIVES DATE IN THE FORM OF NUMBER.

SELECT TO\_CHAR(SYSDATE, 'DAY/MON/YYYY') DAY FROM DUAL; -- IT GIVES DATE IN THE FORM OF WORDS.

SELECT NEXT\_DAY(SYSDATE,'sunday') FROM DUAL;--IT GIVES NEXT SUNDAY DATE.

SELECT NEXT\_DAY(SYSDATE,'monday') FROM DUAL;--IT GIVES NEXT MONDAY DATE.

SELECT NEXT\_DAY('11/08/2008','monday') FROM DUAL;

--- 'ROUND' IN DATE FUNCTION.

--'ROUND' SYNTAX = ROUND(date,formate).

SELECT ROUND(SYSDATE) FROM DUAL; -- IT WILL ROUND TO NEXT DAY ONLY WHEN DAY'S TIME IS MORE THAN HALF DAY(TIME MORE THAN 12.00PM)

SELECT ROUND(SYSDATE,'DD') FROM DUAL; -- IT WILL ROUND TO NEXT DAY ONLY WHEN DAY'S TIME IS MORE THAN HALF DAY(TIME MORE THAN 12.00PM)

-- AFTER 12.00PM ROUND TO NEXT DYA AND LESS THAN 12.00PM ROUND TO SAME DAY.

SELECT ROUND(SYSDATE,'MONTH') FROM DUAL;--IT WILL ROUND TO NEXT MONTH ONLY WHEN SYSDSTE IS CROSSED MORE THAN HALF MONTH OR MORE THAN 15th.

SELECT ROUND(TO\_DATE('20/11/2020'),'MONTH') FROM DUAL; --IT ROUND TO NEXT MONTH "12th" OR DECEMBER BECAUSE DATE IS 20th(MORE THAN 15).

SELECT ROUND(TO\_DATE('08/11/2020'),'MONTH') FROM DUAL;--IT ROUND TO SAME MONTH "11th" OR NOVEMBER BECAUSE DATE IS 8th (LESS THAN OR EQUAL 15).

SELECT ROUND(TO\_DATE('16/11/2020'),'MONTH') FROM DUAL;--IT ROUND TO SAME MONTH "11th" OR NOVEMBER BECAUSE DATE IS 15th (EQUAL TO 15)

--HERE DATE "0" OT "15th" ARE ROUND TO PREVIOUS MONTH AND ONLY "16th" AND ABOVE ARE ROUND TO NEXT MONTH.

SELECT ROUND(TO\_DATE('16/11/2020'),'YEAR') FROM DUAL;--IT WILL ROUND TO NEXT YEAR.

SELECT ROUND(TO\_DATE('16/5/2020'),'YEAR') FROM DUAL;--IT WILL ROUND TO SAME YEAR.

SELECT ROUND(TO\_DATE('16/6/2020'),'YEAR') FROM DUAL;--IT WILL ROUND TO SAME YEAR.

--HERE MONTHS '0' TO '6' ARE ROUND TO SAME YEAR AND MONTHS 7th TO 12th ARE ROUND TO NEXT YEAR.

SELECT ROUND(TO\_DATE('24/5/2022'),'DAY') FROM DUAL;

SELECT ROUND(TO\_DATE('21/5/2020'),'DAY') FROM DUAL;

SELECT ROUND(TO\_DATE('29/5/2020'),'DAY') FROM DUAL;

--- 'TRUNC' IN DATE FUNCTION.

--TRUNC(date,format).

SELECT TRUNC(SYSDATE) FROM DUAL;

SELECT TRUNC(SYSDATE,'DD') FROM DUAL;

SELECT TRUNC(SYSDATE,'MONTH') FROM DUAL;

SELECT TRUNC(TO\_DATE('20/11/2020'),'MONTH') FROM DUAL;

SELECT TRUNC(TO\_DATE('20/11/2020'),'YEAR') FROM DUAL;

SELECT TRUNC(TO\_DATE('20/11/2020'),'DAY') FROM DUAL;

---CONVERSION FUNCTION (TO\_DATE, TO\_NUMBER, TO\_CHAR).

--TO\_DATE (string,format,NLS).

SELECT TO\_DATE('2018/11/23','YYYY/MM/DD') FROM DUAL;

SELECT TO\_DATE('20181123','YYYY/MM/DD') FROM DUAL;

SELECT TO\_DATE('20102022','DD/MM/YYYY') FROM DUAL;

SELECT TO\_DATE('20/NOV/2022','DD/MM/YYYY') FROM DUAL;

SELECT TO\_DATE('201122','DD/MM/YY') FROM DUAL;

--'TO\_NUMBER' FUNCTION.

--TO\_NUMBER(string,format,NLS).

SELECT TO\_NUMBER('4321','9999') FROM DUAL;

SELECT TO\_NUMBER('4321.50','9999.99') FROM DUAL;

---'TO\_CHAR' FUNCTION.

--TO\_CHAR(string,format,NLS).

SELECT TO\_CHAR('567890.50','9,99,9999.00') FROM DUAL;

SELECT TO\_CHAR('567890.50','$9,99,9999.00') FROM DUAL;

SELECT TO\_CHAR(SYSDATE) FROM DUAL;

SELECT TO\_CHAR(SYSDATE,'MM/DD/YYYY') FROM DUAL;

SELECT TO\_CHAR(SYSDATE,'MM-DD-YYYY') FROM DUAL;

SELECT TO\_CHAR(SYSDATE,'month/DAY/YYYY') FROM DUAL; -- IT CONVERT MONTH TO SMALL LETTER AND DAY TO CAPITAL LETTER.

SELECT TO\_CHAR(SYSDATE,'MM-DDth-YYYY') FROM DUAL;

SELECT TO\_CHAR(SYSDATE,'MM-ddth-YYYY') FROM DUAL;

SELECT TO\_CHAR(SYSDATE,'DDSPTH- Month-year') FROM DUAL;

SELECT TO\_CHAR(SYSDATE,'DdSPTH- MONTH-Year') FROM DUAL;

SELECT TO\_CHAR(SYSDATE,'Ddth "of" Month') FROM DUAL;

SELECT TO\_CHAR(HIRE\_DATE,'DD/MONTH/YYYY'), TO\_CHAR(salary,'$99,999') FROM hr.employees;

---AGGREGATE FUNCTIONS (SUM,AVG,MAX,MIN AND COUNT).

SELECT SUM(salary) FROM hr.employees

WHERE job\_id='FI\_ACCOUNT';

SELECT AVG(salary) FROM hr.employees

WHERE job\_id='FI\_ACCOUNT';

SELECT MAX(salary) FROM hr.employees

WHERE job\_id='FI\_ACCOUNT';

SELECT MAX(HIRE\_DATE) FROM hr.employees

WHERE job\_id='FI\_ACCOUNT';

SELECT MIN(HIRE\_DATE) FROM hr.employees

WHERE job\_id='FI\_ACCOUNT';

SELECT MIN(salary) FROM hr.employees

WHERE job\_id='FI\_ACCOUNT';

SELECT COUNT(\*) FROM hr.employees

WHERE job\_id='FI\_ACCOUNT';

SELECT COUNT(first\_name) FROM hr.employees;

SELECT COUNT(\*) FROM hr.employees;

---'GROUP BY' CLAUSE.

-- 'GROUP BY' SYNTAX = SELECT(COLUMN\_NAME, aggregate\_function,aggregate\_expression) FROM table\_name WHERE CONDITION GROUP BY column\_name1,column\_name2..., HAVING CONDITION;.

SELECT department\_id,SUM(salary),MIN(salary),MAX(salary),ROUND(AVG(salary)),COUNT(department\_id) FROM hr.employees

GROUP BY department\_id

ORDER BY department\_id ASC;

---'RANGING PARTITION' FUNCTION.

--IT IS POWERFULL FUNCTIONALITY TO LOGICALLY PARTITION OBJECT INTO SMALLER PIECES.

--PARTITIONING ENABLES DATA MANAGEMENT OPERATIONS SUCH DATA LOADS, INDEX CREATION AND REBULDING, AND BACKUP / RECOVERY AT THE PARTITION LEVEL,RATHER THAN ON THE ENTIRE TABLE. THIS RESULTS IN SIGNIFICANTLY REDUCED TIMES FOR THESE OPERATIONS.

---WHY PARTITION.

--PERFORMANCE: PARTITIONING IMPROVES QUERY PERFORMANCE. IN MANY CASES, THE RESULTS OF A QUERY CAN BE ACHIEVED BY ACCESSING A SUBSET OF PERTITIONS,RATHER THAN THE ENTIRE TABLE.

--AVAILABILITY: PARTITIONING INCREASES THE AVAILABILITY OF MISSION CRITICAL DATABASES IF CRITICAL TABLES AND INDEXES ARE DIVIDED INTO PARTITIONS TO REDUCE THE MAINTENANCE WINDOW, RECOVERY TIMES, AND IMPACT OF FAILURES.

--TYPES ARE RANGE PARTITIONING, LIST PARTITIONING, HASH PARTITIONING, COMPOSITE PARTITIONING.

--'RANGE PARTITIONING'

--USEFUL WHEN DATA HAS LOGICAL RANGES INTO WHICH IT CAN BE DISTRIBUTED.EXAMPLE A RANGE OF DATES, OR NUMBERS.

--DATA IS MAPPED TO PARTITIONS BASED ON RANGES OF PARTITION KEY VALUES ESTABLISHED FOR EACH PARTITION.

--

COMMIT

DESC chethan

SELECT \* FROM hr.employees;

SELECT \* FROM chethan1

SELECT \* FROM BSEIT1

SELECT \* FROM BSEIT