SQL----> STRUCTURED QUERY LANGUAGE

STATEMENTS /LANGUAGES /SUBSET OF SQL

DDL---> DATA DEFINITION LANGUAGE

TO CREATE OR MODIFY THE STRUCTRE OF OBJECT

--> CREATE

TO CREATE THE TABLE

--> RENAME

TO RENAME A TABLE\_NAME

--> ALTER

TO MODIFY THE STRUCTURE OF TABLE

--> TRUNCATE

TO REMOVE ALL THE RECORDS FROM THE TABLE

--> DROP

TO REMOVE THE TABLE FROM DATABASE

DML---> DATA MANIPULATION

TO MANIPULATE TABLE

--> INSERT

TO INSERT THE RECORDS

--> UPDATE

TO MODIFY THE EXISTING VALUE

--> DELETE

TO REMOVE ALL THE RECORDS OR SET OF RECORDS OR ONE RECORD

TCL---> TRANSACTION CONTROL

TO CONTROL THE TRANSACTIONS

--> COMMIT

TO SAVE THE TRANSACTIONS

--> ROLLBACK

TO TAKES US THE POINT WHERE WE COMMITTED

--> SAVEPOINT

TO CREATE CHECK POINTS OR RESTORATION POINTS

DCL---> DATA CONTROL

TO CONTROL THE FLOW OF DATA

--> GRANT

TO GIVE PERMISSION

--> REVOKE

TO TAKEBACK THE PERMISSION

DQL---> DATA QUERY

TO RETRIEVE OR FETCH THE DATA

--> SELECT

TO DISPLAY THE FINAL OUTPUT

--> PROJECTION

TO RETRIEVE THE DATA BY SELECTING ONLY COLUMNS

--> SELECTION

TO RETRIEVE THE DATA BY SELECTING ROWS AND COLUMNS

--> JOINS

TO RETRIEVE THE DATA FROM MULTIPLE TABLES SIMULTANEOUSLY

CREATE

CREATE TABLE TABLE\_NAME

(

COLUMN\_NAME1 DATATYPE [CONSTRAINTS],

COLUMN\_NAME2 DATATYPE [CONSTRAINST],

,

COLUMN\_NAMEn DATATYPE [CONSTRAINTS]

);

RENAME

RENAME EXISTING\_TABLE\_NAME TO NEW\_TABLE\_NAME;

ALTER

TO ADD A COLUMN

ALTER TABLE TABLE\_NAME

ADD COLUMN\_NAME DATATYPE [CONSTRAINTS];

TO RENAME A COLUMN

ALTER TABLE TABLE\_NAME

RENAME COLUMN EXISTING\_COLUMN\_NAME TO NEW\_COLUMN\_NAME;

TO DROP A COLUMN

ALTER TABLE TABLE\_NAME

DROP COLUMN COLUMN\_NAME;

TO MODIFY THE DATATYPE

ALTER TABLE TABLE\_NAME

MODIFY COLUMN\_NAME NEW\_DATATYPE;

TO MODIFY NULL/NOT NULL

ALTER TABLE TABLE\_NAME

MODIFY COLUMN\_NAME DATATYPE NULL/NOT NULL;

TO ADD A UNIQUE

ALTER TABLE TABLE\_NAME

ADD CONSTRAINTS CONSTRAINT\_REF\_NAME UNIQUE(COLUMN\_NAME);

TO ADD A CHECK

ALTER TABLE TABLE\_NAME

ADD CONSTRAINTS CONSTRAINT\_REF\_NAME CHECK(CONDITION);

TO ADD A PRIMARY KEY

ALTER TABLE TABLE\_NAME

ADD CONSTRAINTS CONSTRAINT\_REF\_NAME PRIMARY KEY(COLUMN\_NAME);

TO ADD A FOREIGN KEY

ALTER TABLE TABLE\_NAME

ADD CONSTRAINTS CONSTRAINT\_REF\_NAME FOREIGN KEY(COLUMN\_NAME)

REFERENCES PARENT\_TABLE\_NAME(COLUMN\_NAME);

TRUNCATE

TRUNCATE TABLE TABLE\_NAME;

DROP

DROP TABLE TABLE\_NAME;

FLASHBACK

FLASHBACK TABLE TABLE\_NAME

TO BEFORE DROP;

PURGE

PURGE TABLE TABLE\_NAME;

INSERT

INSERT INTO TABLE\_NAME VALUES(V1,V2,,,Vn);

INSERT INTO TABLE\_NAME VALUES(&COLUMN1,&COLUMN2,,,&COLUMNn);

INSERT INTO TABLE\_NAME(COLUMN1,COLUMN2,,,COLUMNn)VALUES(V1,V2,,,,Vn);

UPDATE

UPDATE TABLE\_NAME

SET COLUMN\_NAME=NEW\_VALUE

[WHERE <FILTER\_CONDITION>];

UPDATE TABLE\_NAME

SET COLUMN\_NAME=NEW\_VALUE , COLUMN\_NAME=NEW\_VALUE

[WHERE <FILTER\_CONDITION>];

DELETE:

DELETE FROM TABLE\_NAME

[WHERE <FILTER\_CONDITION>];

COMMIT

COMMIT;

ROLLBACK

ROLLBACK;

SAVEPOINT

SAVEPOINT SAVEPOINT\_NAME;

ROLLBACK TO SAVEPOINT\_NAME;

GRANT

GRANT SQL\_STATEMENT

ON TABLE\_NAME

TO USER\_NAME;

REVOKE

REVOKE SQL\_STATEMENT

ON TABLE\_NAME

FROM USER\_NAME;

CARTESIAN JOIN/CROSS JOIN

ANSI

SELECT COLUMN-NAME/EXPRSSION

FROM TABLE\_NAME1 CROSS JOIN TABLE\_NAME2;

ORACLE

SELECT COLUMN\_NAME/EXPRESSION

FROM TABLE\_NAME1 , TABLE\_NAME2;

INNER JOIN/EQUI JOIN

ANSI

SELECT COLUMN\_NAME/EXPRESSION

FROM TABLE\_NAME1 INNER JOIN TABLE\_NAME2

ON <JOIN\_CONDITION>;

ORACLE

SELECT COLUMN\_NAME/EXPRESSION

FROM TABLE\_NAME1 , TABLE\_NAME2

WHERE <JOIN\_CONDITION>;

NATURAL JOIN

ANSI

SELECT COLUMN-NAME/EXPRESSION

FROM TABLE\_NAME1 NATURAL JOIN TABLE\_NAME2;

SELF JOIN

ANSI

SELECT COLUMN\_NAME/EXPRESSION

FROM TABLE\_NAEM T1 JOIN TABLE\_NAME T2

ON T1.COLUMN\_NAME=T2.COLUMN\_NAME;

ORACLE

SELECT COLUMN\_NAME/EXPRESSION

FROM TABLE\_NAME T1, TABLE\_NAME T2

WHERE T1.COLUMN\_NAME=T2.COLUMN\_NAME;

LEFT OUTER JOIN

ANSI

SELECT COLUMN\_NAME/EXPRESSION

FROM TABLE\_NAEM1 LEFT [OUTER] JOIN TABLE\_NAME2

ON TABLE\_NAME1.COLUMN\_NAME=TABLE\_NAME2.COLUMN\_NAME;

ORACLE

SELECT COLUMN\_NAME/EXPRESSION

FROM TABLE\_NAEM1 , TABLE\_NAME2

WHERE TABLE\_NAME1.COLUMN\_NAME=TABLE\_NAME2.COLUMN\_NAME(+);

RIGHT OUTER JOIN

ANSI

SELECT COLUMN\_NAME/EXPRESSION

FROM TABLE\_NAEM1 RIGHT [OUTER] JOIN TABLE\_NAME2

ON TABLE\_NAME1.COLUMN\_NAME=TABLE\_NAME2.COLUMN\_NAME;

ORACLE

SELECT COLUMN\_NAME/EXPRESSION

FROM TABLE\_NAEM1 , TABLE\_NAME2

WHERE TABLE\_NAME1.COLUMN\_NAME(+)=TABLE\_NAME2.COLUMN\_NAME;

FULL OUTER JOIN

ANSI

SELECT COLUMN\_NAME/EXPRESSION

FROM TABLE\_NAEM1 FULL [OUTER] JOIN TABLE\_NAME2

ON TABLE\_NAME1.COLUMN\_NAME=TABLE\_NAME2.COLUMN\_NAME;

DIFFERENCE BETWEEN VARCHAR AND VARCHAR2

VARCHAR-----> 2000 CHARACTERS

VARCHAR2----> 4000 CHARACTERS

SET OPERATORS

TO COMBINE TWO OR MORE SQL QUERIES

--> UNION

IT WILL COMBINE SQL QUERIES BY REMOVING THE DUPLICATED VALUES

--> UNION ALL

IT WILL COMBINE SQL QUERIES BY RETRIEVING ALL THE RECORDS

--> INTERSECT

IT WILL COMBINE SQL QUERIES BY RETRIEVING COMMOM RECORD

--> MINUS

IT WILL COMBINE SQL QUERIES BY RETIEVING TABLE1 RECORDS IF THOSE ARE

NOT PRESENT TABLE2

CONSTRAINTS

IT IS AN EXTRA VALIDATION WHICH IS ASSIGNED FOR AN COLUMN

TYPES OF CONSTRAINTS

UNIQUE

NOT NULL

CHECK

PRIMARY KEY

FOREIGN KEY

JOINS

THE PROCESS OF RETRIEVING THE DATA FROM MULTIPLE TABLES SIMULTANEOUSLY

WHEN OR WHY DO WE USE JOINS

WHENEVER THE DATA TO BE DISPLAYED PRESENT IN MULTIPLE TABLES WE USE

JOINS

TYPES OF JOINS

WE HAVE 5 TYPES OF JOINS

--> CARTESIAN JOIN/CROSS JOIN

--> INNER JOIN/EQUI JOIN (MATCHED RECORDS)

--> NATURAL JOIN

--> SELF JOIN

--> OUTER JOIN (UNMATCHED RECORDS)

--> LEFT OUTER JOIN

--> RIGHT OUTER JOIN

--> FULL OUTER JOIN

WHEN OR WHY DO WE USE SELF JOIN

WHENEVE THE DATA TO BE DISPLAYED PRESENT IN SAME TABLE BUT

IN DIFFERENT RECORDS WE USE SELF JOIN

TO RETRIEVE THE DATA FROM MULTIPLE TABLES(DIFFERENT) ----> INNER JOIN

IN INNER JOIN, JOIN\_CONDITION IS MANDATORY

THE RELATION BETWEEN THE TABLES (PRIMARY KEY AND FOREIGN KEY)

TO RETRIEVE THE DATA FROM MULTIPLE TABLES(SAME) -------> SELF JOIN

EMPLOYEE AND MANAGER RELATION (MGR=EMPNO) (EMPNO=MGR)

LIKE OPERATOR ----> TO RETRIEVE PATTERN MATCHING

DIFFERNECE BETWEEN TRUNCATE AND DROP AND DELETE

TRUNCATE

--> TRUNCATE IS USED TO REMOVE ALL THE RECORDS FROM THE TABLE

--> THERE IS NO BACKUP OPTION FOR TRUNCATE

--> TRUNCATE IS A PERMANENT

DROP

--> DROP IS USED TO REMOVE THE TABLE FROM DATABASE

--> THERE IS A BACKUP OPTION FOR DROP

--> DROP IS A TEMPORARY

DELETE

--> DELETE IS USED TO REMOVE ALL THE RECORDS OR SET OF RECORDS OR ONE RECORD

--> DELETE IS A TEMPORARY

SYNTAX FOR SUB QUERY

SELECT COLUMN\_NAME/EXPRESSION

FROM TABLE\_NAME

WHERE COLUMN\_NAME/EXPRSSION OPERATOR (SELECT COLUMN\_NAME/EXPRESSION

FROM TABLE\_NAME

[WHERE <FILTER\_CONDITION>]);

PRIMARY KEY

--> PRIMARY KEY IS USED TO IDENTIFY A RECORD UNIQUELY FROM THE TABLE

--> PRIMARY KEY WILL NOT ACCEPT DUPLICATED OR REPEATED VALUES

--> PRIMARY KEY WILL NOT ACCEPT NULL

--> PRIMARY KEY IS ALWAYS THE COMBINATION OF UNIQUE AND NOT NULL

--> WE SHOULD HAVE ONLY ONE PRIMARY KEY IN A TABLE

--> PRIMARY KEY IS ALSO CALLED AS UNIQUE KEY

FOREIGN KEY

--> FOREIGN KEY IS USED TO ESTABLISH THE CONNECTION BETWEEN TABLES

--> FOREIGN KEY WILL ACCEPT THE DUPLICATED VALUES

--> FOREIGN KEY WILL ACCEPT NULL VALUES

--> FOEIGN KEY IS NOT A COMBINATION UNIQUE AND NOT NULL

--> WE CAN HAVE MULITPLE FOREIGN KEYS IN A TABLE

--> FOREIGN KEY IS ALSO CALLED AS REFERENTIAL INTEGRITY CONSTRAINT

REFERENTIAL INTEGRITY CONSTRAINT

THE PARENT TABLE CANNOT BE DESTORED UNTIL UNLESS HAVING CONNECTION WITH

CHILD TABLE

SUB QUERY

DEF

WORKING PROCEDURE

WHEN OR WHY DO WE USE SUB QUERY

TYPES OF SUB QUERY

SUB QUERY OPERATORS

NESTED SUB QUERY

TYPES OF ATTRIBUTES

KEY ATTRIBUTES/CANDIDATE KEY

NON-KEY ATTRIBUES

PRIME KEY ATTRIBUTE

NON-PRIME KEY ATTRIBUTE

COMPOSITE KEY ATTRIBUE

SUPER KEY ATTRIBUTE

FOREIGN KEY ATTRIBUTE

TYPES OF FUNCTIONAL DEPENDENCY

TOTAL FUNCTIONAL DEPENDENCY

PARTILA FUNCTIONAL DEPENDENCY

TRANSTITIVE FUNCTIONAL DEPENDENCY

NORMALISATION

FIRST NORMAL FORM (1NF)

SECOND NORMAL FORM (2NF)

THIRD NORMAL FORM (3NF)

BOYCE-CODD NORMAL FORM (BCNF/3.5NF)

DIFFERNECE BETWEEN HAVING CLAUSE AND WHERE CLAUSE

GROUP BY CLAUSE

HAVING CLAUSE

MULTI ROW FUNCTION/GROUP FUNCTION/AGGREGATE FUNCTION