ORACLE ⇒ 11th generation i.e. 11th g.

Enter password ⇒ system
Confirm password ⇒ system

Queries:-

1) CREATE :- Used to create table in database.

syntax :- CREATE TABLE TABLE NAME (FN (varchar2 (20), LN varchar2(20), MN INT, CITY VARCHAR (20));

output :- Table Tablename created.

2) SELECT :- used to fetch data, column or whole table data from database.

Syntax: SELECT * FROM TABLE NAME; → Run.

op :- All column will shown on console which is created by selecting whole table.

3) INSERT :- insert data/record into database.

syntax :- INSERT INTO VALUES TABLE NAME VALUES ('SAGAR', 'KAPALE', '8237217453', 'GOA'); → RUN

4) INSERT INTO TABLE NAME VALUES ('AJAY', 'LANDE', '9876543210', 'MAPUSA'); → RUN

insert Records in database. (multiple records add) (Run select query after inserting every record).

4) DESC TABLENAME :- Shows Description of Table.

syntax :- DESC TABLE NAME; → Run.

④ SELECT CITY FROM TABLE NAME;
→ particular column will display on console from table.

⑤ SELECT CN1, CN2, CN3 FROM TABLE NAME;
→ select multiple columns from tables on console.

5) WHERE clause:- using '*'
particular record has been shown on console using where clause.

syntax:- SELECT * FROM TABLE NAME WHERE FN='SAGAR';
o/p shows records on console whose first name is 'SAGAR'.
Also SELECT FN, LN FROM TABLE NAME WHERE FN='AJIT'; → Run ('AJIT YEDGE' name showed)

SELECT * FROM TABLE NAME WHERE CITY='CITY Name'
→ city Name of records shown from table.

LOGICAL OPERATOR (AND & OR).

6) AND operator:- select only specified records which is asked.

syntax:- SELECT * FROM TABLENAME WHERE FN='AMAR' AND LN='SALUNKE'; → Run
o/p → Records of AMAR SALUNKE only shown on console

7) OR operator:- shows records all present in whole table.

syntax:- SELECT * FROM TABLE NAME WHERE FN='AMAR' OR LN='JOSHI'; → Run
o/p:- Records of AMAR and JOSHI has shown on console.

⑥ Add Record in small case and fetch complete table.

⑦ SELECT *FROM TABLE NAME WHERE FN='sagar', OR 'KAPALE'; → RUN.

8) DELETE :- Only Records has been deleted from table (column name like FN, LN, CITY, etc as it is).

syntax:- DELETE FROM TABLE NAME ; → Run.

9) ROLLBACK :- Back to some stages which was deleted.

syntax:- ROLLBACK;

Imp 10) DROP :- Table does not exist. (data is deleted.

syntax:- DROP TABLE NAME; → Run

Imp Delete & drop difference:-

- Delete:- ① Records delered structure of table as it is
- Drop:- ① Records & structure both deleted
- ~~Delete:- ② Data retrived~~

Delete	Drop.
1) Records deleted, structure of table is as it is.	1) Records & structure both deleted.
2) Data retrieved after rollback in delete.	2) Data not retrieved after rollback in drop.

⑩ DELETE FROM TABLENAME WHERE FN = 'ARVIND';
(All records of ARVIND has been deleted | complete row).

⑪ DELETE FROM TABLENAME WHERE FN = 'AJIT' AND CITY = 'PUNE'; → Run.
(selected row deleted).

11) ALTER & ADD :- Use for modification of existing structure.

Query :- ALTER TABLE TABLENAME ADD SALARY VARCHAR2(30);
o/p :- Table TABLENAME altered.
Salary column has been added. (~~column~~ currently that column name shown (null)).

12) UPDATE :- changes in existing records or updating existing record.

Query : UPDATE TABLENAME SET SALARY = '60000' WHERE FN = 'ANUP' ; → Run.
o/p :- (1 row updated ; ~~fetch~~ table → salary will set to ANUP.

⑫ ALTER TABLE TABLE NAME DROP COLUMN CITY;
(CITY column has been deleted from table).

⑬ UPDATE TABLENAME SET MN = '9371819530' WHERE FN = 'AJIT'; → Run.

Note :- ALTER → manipulate the ~~data.~~ column
UPDATE → manipulate the data.

13) MODIFY :- Datatype of column name has been modified eg. VARCHAR → int.
(Datatype & Datasize is verified while modifying new datatype).

Query:- ALTER TABLE TABLENAME MODIFY SALARY INT;
o/p:- Datatype of salary changes from varchar to integer.

Note:- First delete information from 1st datatype then change next datatype.

@ DELETE FROM TABLENAME WHERE SALARY = '60000';
→ Records has been deleted whosever having salary=60000

14) RENAME :- Rename the table name, column. etc.

Query:- ALTER TABLE TABLENAME RENAME COLUMN OLD COLUMN NAME TO NEW COLUMN NAME;
eg. ALTER TABLE TABLENAME RENAME COLUMN MN TO MOB; → Run.
o/p:- column name gets renamed from MN to MOB.

AGGREGATE FUNCTIONS

update salary to every record.

15) MAX :- maximum amount of Record or data has been shown on console.
Query:- SELECT MAX (COLUMN NAME i.e. SALARY), TABLE NAME; FROM.
o/p:- maximum salary shown.

16) MIN :-
Query:- SELECT MIN (SALARY) FROM TABLENAME ; → Run
o/p:- minimum salary shown.

17) AVG :- i.e Average

Query:- SELECT AVG(SALARY) FROM * TABLE NAME ; Run.
O/P:- Average of all salaries have shown on console.

18) SUM :- i.e. summation.

Query :- SELECT SUM (SALARY) FROM TABLENAME ; Run
→ sum of all salaries.

19) COUNT :-

Query :- SELECT COUNT(SALARY) FROM TABLENAME ; Run
→ count has been shown.

ARITHMATIC OPERATORS

20) Less THAN i.e. '<' :- Less than values shown quantity/data.

Query:- SELECT * FROM TABLENAME WHERE SALARY<'5000
O/P:- (Records shown on console whose salary less than 5000

21) Greater Than i.e. '>' :-

Query:- SELECT * FROM TABLE NAME WHERE SALARY > '50000'
O/P :- (Records shown on console whose salary greater than 50000)'.

22) Greater Than equal To :- i.e. '>=' :

Query:- SELECT * FROM TABLE NAME WHERE SALARY >= '50000'
→ Salary greater than '50000' has been shown on console

23) Less Than equal To :- i.e. '<='

Q→ SELECT * FROM TABLE NAME WHERE SALARY <='5000'
→ (Salary less than & equal to 50000 has shown)

24) Equal To :- i.e '='

Q→ SELECT * FROM TABLE NAME WHERE SALARY='50000';
→ only 50000 Salary record shown on console.

25) Not equal i.e. '!=' :-

Q→ SELECT * FROM TABLE NAME WHERE SALARY !='50000';
→ All records shown ~~only~~ except salary whose not equal to 50000.

Not equal i.e. '<>' :-

Q:- SELECT * FROM TABLE NAME WHERE SALARY <>'50000'; → Run.
→ All records shown except salary whose not equal to 50000.

26) DISTINCT :- For finding unique record we use distinct double record eliminate & single record shows of two person. ~~Distinct~~ [Distinct → unique records on console].

Q Add / insert duplicate records.
Q → SELECT DISTINCT FN FROM TABLE NAME;
→ Double records eliminate from First name column.

Q→ SELECT DISTINCT * FROM TABLE NAME; → Run.
→ Double / duplicate records eliminates from whole table.

© "---" used for comment out line.

© /* SELECT DISTINCT * FROM TABLE NAME */;
(all query / whole query is comment out.

27) COMMIT:- used to hard save the data. it can not rollback.

Q. COMMIT; → Run
→ Data hard saved.

28) BETWEEN:- Records which lying between two quantities shown on console.
more than 2 values we use between query.

Query:- SELECT * FROM TABLE NAME WHERE SALARY BETWEEN '50000' AND '70000'; → Run.

op → Records / data shown on console whose salary lying between 50000 to 70000.

29) 'ASC' :- Records in Ascending order shown on console.

Query:- SELECT SALARY FROM TABLENAME ORDER BY SALARY ASC; → Run.

o/p:- Salary column in ascending column has been shown.

30) DESC:- Records in Descending order shown on console

Q :- SELECT SALARY FROM TABLENAME ORDER BY SALARY DESC; → Run

o/p:- salary column in descending order shown on console

Note :- same for other columns also like city, mob no.

Note:- without ASC & DESC takes & records shown by default in ascending order.

@ SELECT SALARY FROM TABLE NAME ORDER BY SALARY;
→ By default records shown in asending order on console.

@ ORDER BY :- used to show records according to ascending & descending order.

PATTERN SEARCH

8|) LIKE :- using like command with '%' sign.

Query:- SELECT * FROM TABLE NAME WHERE FN LIKE 'A%';
o/p → Records shows from first name (column) started with letter 'A'.

@ SELECT * FROM TABLE NAME WHERE FN LIKE '%A';
→ Records shows from first name (column) end with letter 'A';

@ SELECT * FROM TABLE NAME WHERE FN LIKE '%A%';
→ Records shows from first name column having letter 'A'.

@ DESC TABLE NAME; → RUN.
(null record with null column shown)
1 blank record insert with blank 1st name & 2nd name.
(blank or null value shown).

Constraints

32) UNIQUE :- Unique constraints ensures all values in column are different.

33) NOT NULL :- Not null constraints forces a column to Not accept Null. values.

new table created.

query:- CREATE TABLE TABLENAME (EID INT UNIQUE, DESIGNATION VARCHAR2(20) NOT NULL, LOCATION VARCHAR2 (20));

→ (Table tablename has been created), with EID column is unique & Designation column as not null.

⊙ DESC table name; →Run.

insert records :- in new table.

① INSERT INTO table name values ('1007', 'SQA', 'ODC7');

② INSERT INTO table name VALUES ('1007', 'SQAF', 'ODC8');

~~③ INSERT INTO table name VALUES~~

→ error i.e. unique constraints shown on console.

① INSERT INTO table name VALUES ('1008', ' ', 'ODC9');

→ error i.e. cannot insert null values.

② UPDATE table name SET LOCATION = 'ODC5' WHERE EID = '1008'; →run

→ location is set to empEID 1008

③ UPDATE tablename EID = '1007' WHERE LOCATION = 'ODC9';

→ error i.e. unique constraints (EID is unique).

34) PRIMARY KEY :- This constraint uniquely identifies each record in table. primary key is combination of unique & not null.

Table 1

Query :- CREATE TABLE TABLENAME1(E'd int, FN varchar2(20), LN varchar2(20), city varchar2(10), Primary key (E'd));

- SELECT * FROM TABLENAME;
- desc TABLENAME;

- Insert into values (101, 'Anup', 'Tak', 'Nanded');
  → if same record inserted error will show because of unique E'd.
- insert without E'd
  → error not enough values
- insert into values (' ', 'Anup', 'Tak', 'Nanded');

Table 2

35) Foreign key :-

36) CREATE table table name2(EmployeeID int, Designation varchar2(10), salary int, foreign key (EmployeeID) REFERENCES Table name (EID));

(Table linked with previous table).

- insert into table name2 values (107, 'QA', 75000);
  (error - integrity constraints) parent key not found.
- insert into table name2 values (101, 'QA', '75000');
  1 row inserted.

DEFAULT & CHECK

36) DEFAULT :- Default constraint is used to provide default value for column.

37) CHECK :- Check constraint used to limit the value range that can be placed in column. If you define check constraint on single column it allows only certain values for this column.

38) Join :-

i) Inner Join :-

Query :- SELECT TN1.CN1, TN1.CN2, TN2.CN1, TN2.CN2 FROM TN1 INNER JOIN TN2 ON TN1.CN(primary key) = TN2.CN(Foreign key)

→ When 1 & 2 tables combined, using inner join common data shown on console from both table.

2) LEFT Join & 3) RIGHT JOIN :-

Q :- SELECT TN1.CN1, TN1.CN2, TN2.CN1, TN2.CN2 FROM TN1 ~~INNER~~ LEFT JOIN / RIGHT JOIN TN2 ON TN1.CN(primary key) = TN2.CN(foreign key)

→ (2) After join returns whole rows from left table and common from right table.

(3) After join returns whole right table & common from left table

iv) FULL JOIN :-

→ Shows both tables joins and shown on console.

* SQL Table Constraints:-

- SQL constraints is used to specify rules for data in table. constraints can be used when table is created with CREATE TABLE statement.

- **NOT NULL** :- Not null constraints forces a column to not accept NULL values. because while creating tables if value not present it take NULL value in CN.

- **UNIQUE**:- The UNIQUE constraints ensures, all values in column are different.

- **PRIMARY KEY** :- Primary key constraint uniquely identifies each record in table. Primary key must contain UNIQUE values, & can not contain NULL values.

- **FOREIGN KEY**:- A foreign key is used to link two tables together. A foreign key is a field (or collection of fields) in one table that refers to the PRIMARY KEY in another table.

- **CHECK**:- The CHECK constraint is used to limit value range that can be placed in a column. If you define a CHECK constraint on single column it allows only certain values in this column.

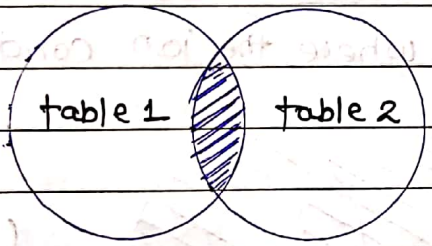
- **DEFAULT** :- The DEFAULT constraint is used to provide defaut value for a column.

Primary Key	Foreign key	Unique key.
Primary key constraint uniquely identifies each record in a table. Primary key must contain UNIQUE values.	Foreign key is a key used to link two tables together. A FOREIGN key is a Field (or collection of Fields) in one table that refers to the PRIMARY key in another table.	The UNIQUE constraint ensures that all values in column are differents.
Primary key can not a NULL value	Foreign key can accept multiple NULL value.	UNIQUE constraints may have NULL values
Each table can have only one primary key.	We can have more than one foreign key in a table.	Each table can have more than one unique constraint.
Primary key is clustered index.	Foreign keys do not automatically create an index, clustered or non-clustered.	Unique key is ~~non~~ unique non-clustered index.

Q SQL Join :- JOINS are used to retrieve data from multiple tables. A SQL server JOIN is performed whenever two or more tables are joined in a SQL statement.

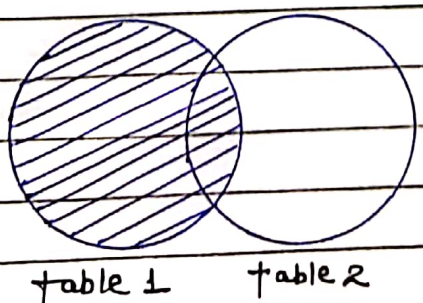
Syntax:- SELECT TN1.CN1, TN1.CN2, TN2.CN1, TN2.CN2
FROM TN1 INNER JOIN TN2 ON
TN1.CN (Primary key) = TN2.CN (Foreign key).

1) INNER JOIN :- it is most common type of join. SQL server INNER JOINS returns all rows from multiple tables where two tables contains common values in the join condition is met.



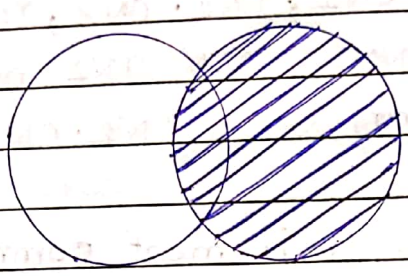
(only common rows/ data).

2) LEFT JOIN :- This join returns all rows from Left-hand side table specified in the ON condition & only those rows from the other table where the joined fields are equal (join condition is met).



(All left hand table data & common in right hand table).

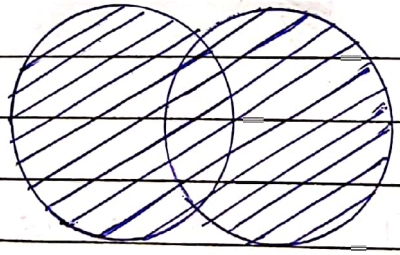
3) RIGHT JOIN :- This type of Join returns all rows from Right-hand table specified in the ON condition & only those rows from other table where the joined fields are equal (join condition is met).



(All right side table data & common from left hand table)

Table 1 Table 2

4) FULL JOIN :- This type of Join returns all rows from the left hand side & Right hand side with nulls in place. where the join condition is met.



(Join both table & shows both data of table ~~on~~ on consale).

Table 1 Table 2