CREATE DATABASE *databasename*;

eg:CREATE DATABASE *test*;

CREATE TABLE *table_name*(
column1 datatype,
column2 datatype,
column3 datatype,
....
);

eg:Persons(PersonID, LastName, FirstName, Address, City)

CREATE TABLE Persons (
PersonID int,
LastName varchar(25),
FirstName varchar(25),
Address varchar(25),
City varchar(25)
);

Table with constraint

CREATE TABLE *table_name*(
column1 datatype constraint,
column2 datatype constraint,
column3 datatype constraint,
.....
);

The following constraints are commonly used in SQL:

- NOT NULL - Ensures that a column cannot have a NULL value
- UNIQUE - Ensures that all values in a column are different
- PRIMARY KEY - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
- FOREIGN KEY - Prevents actions that would destroy links between tables
- CHECK - Ensures that the values in a column satisfies a specific condition
- DEFAULT - Sets a default value for a column if no value is specified
- CREATE INDEX - Used to create and retrieve data from the database very quickly

create a table student(stu_id,name,dept_id,ins_id)
CREATE TABLE student(
stu_id int primarykey,
name varchar(20),
dept_id varchar(15) FOREIGN KEY REFERENCES department(dept_id),
ins_id int FOREIGN KEY REFERENCES instuctor(ins_id)
);

department(dept_id,deptname,location,budget,ins_id,stu_id)
CREATE TABLE department(
dept_id int PRIMARY KEY;
deptname varchar(25) NOT NULL,
location varchar(20),
budget int,
ins_id int,
stu_id int FOREIGN KEY REFERENCES student(stu_id),
CONSTRAINT fk FOREIGN KEY(ins_id) REFERENCES instuctor(ins_id),
);
instuctor(ins_id,insname,dept_id,stu_id,salary)

SQL ALTER TABLE Statement

The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.

The ALTER TABLE statement is also used to add and drop various constraints on an existing table.

ALTER TABLE - ADD Column

To add a column in a table, use the following syntax:

ALTER TABLE table_name

ADD column_name datatype;

ALTER TABLE - DROP Column

ALTER TABLE *table_name*
DROP COLUMN *column_name*;
ALTER TABLE - MODIFY/ALTER Column

ALTER TABLE *table_name*
ALTER/MODIFY COLUMN *column_name datatype*;

ORACLE 10 G

ALTER TABLE *table_name*
MODIFY *column_name datatype*;

QNS

1. ADD A COLUMN DATEOFBIRTH OF TYPE DATE TO STUDENT TABLE

ALTER TABLE student

ADD dateofbirth date;

2.CHANGE THE DATATYPE OF STU_ID TO VARCHAR

ALTER TABLE student

MODIFY STU_ID VARCHAR(3);

3.To create a NOT NULL constraint on the "BUDGET" column when the "DEPARTMENT" table is already created

SQL CHECK Constraint

The CHECK constraint is used to limit the value range that can be placed in a column.

If you define a CHECK constraint on a column it will allow only certain values for this column.

If you define a CHECK constraint on a table it can limit the values in certain columns based on values in other columns in the row.

SQL CHECK on CREATE TABLE

The following SQL creates a CHECK constraint on the "Age" column when the "Persons" table is created. The CHECK constraint ensures that the age of a person must be 18, or older:

MySQL:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    CHECK (Age>=18)
);
```

SQL Server / Oracle / MS Access:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int CHECK (Age>=18)
);
```

Syntax

The basic syntax of an ALTER TABLE command to add a **New Column** in an existing table is as follows.

```
ALTER TABLE table_name ADD column_name datatype;
```

The basic syntax of an ALTER TABLE command to **DROP COLUMN** in an existing table is as follows.

```
ALTER TABLE table_name DROP COLUMN column_name;
```

The basic syntax of an ALTER TABLE command to change the **DATA TYPE** of a column in a table is as follows.

```
ALTER TABLE table_name MODIFY COLUMN column_name datatype;
```

The basic syntax of an ALTER TABLE command to add a **NOT NULL** constraint to a column in a table is as follows.

```
ALTER TABLE table_name MODIFY column_name datatype NOT NULL;
```

The basic syntax of ALTER TABLE to **ADD UNIQUE CONSTRAINT** to a table is as follows.

```
ALTER TABLE table_name
ADD CONSTRAINT MyUniqueConstraint UNIQUE(column1, column2...);
```

The basic syntax of an ALTER TABLE command to **ADD CHECK CONSTRAINT** to a table is as follows.

```
ALTER TABLE table_name
ADD CONSTRAINT MyUniqueConstraint CHECK (CONDITION);
```

The basic syntax of an ALTER TABLE command to **ADD PRIMARY KEY** constraint to a table is as follows.

```
ALTER TABLE table_name
ADD CONSTRAINT MyPrimaryKey PRIMARY KEY (column1, column2...);
```

The basic syntax of an ALTER TABLE command to **DROP CONSTRAINT** from a table is as follows.

```
ALTER TABLE table_name
DROP CONSTRAINT MyUniqueConstraint;
```

The basic syntax of an ALTER TABLE command to **DROP PRIMARY KEY** constraint from a table is as follows.

```
ALTER TABLE table_name
DROP CONSTRAINT MyPrimaryKey;
```

To rename a table, the SQL ALTER TABLE syntax is:

For Oracle, MySQL, MariaDB, PostgreSQL and SQLite:

```
ALTER TABLE table_name

  RENAME TO new_table_name;
```

DROP command

DROP command completely removes a table from the database. This command will also destroy the table structure and the data stored in it. Following is its syntax,

```
DROP TABLE table_name
```

eg:

```
DROP TABLE student;
```

The SQL **TRUNCATE TABLE** command is used to delete complete data from an existing table.

You can also use DROP TABLE command to delete complete table but it would remove complete table structure form the database and you would need to re-create this table once again if you wish you store some data.

Syntax

The basic syntax of a **TRUNCATE TABLE** command is as follows.

```
TRUNCATE TABLE  table_name;
```

INSERT command

Insert command is used to insert data into a table. Following is its general syntax,

```
INSERT INTO table_name VALUES(data1, data2, ..Consider a table student with the following fields.
```

s_id	name	age

```
INSERT INTO student VALUES(101, 'Adam', 15);
```

The above command will insert a new record into **student** table.

s_id	name	age
101	Adam	15

clause

Basic structure of SQL Queries

SELECT $A_1,A_2,.....A_n$ - List desired attributes(A1,A2.....) in the result, include arithmetic expressions with +,-,*,/ etc

FROM $r_1,r_2,.....r_n$ - List of relations(r1,r2.....) to be accessed for query evaluation

WHERE P - predicate involving the attributes of relations in FROM clause.

If WHERE clause omitted, P is true.

Logical connectives AND, OR, NOT and comparison operators <,<=,>,>=,==,<> can be used here

steps

1.First FROM -Generate a cartesian product of the relations listed in FROM clause

2.Then WHERE -Apply the predicates specified here on result of step1

3.Finally SELECT -For each tuple in the result of step 2,output the attributes specified here.

Natural join

SELECT $A_1,A_2,.....A_n$

FROM r_1 NATURAL JOIN r_2 NATURAL JOIN .....r_n

WHERE P

AS clause

-To rename a relations

- to rename atrributes of a result relation

-to compare tuples in the same relation

syntax

oldname AS newname

String functions

are used to perform an operation on input string and return an output string.

Following are the string functions defined in SQL:

1. ASCII(): This function is used to find the ASCII value of a character.

Syntax: SELECT ascii('t');

Output: 116

2. CHAR_LENGTH(): Doesn’t work for SQL Server. Use LEN() for SQL Server. This function is used to find the length of a word.

Syntax: SELECT char_length('Hello!');

Output: 6

3. CHARACTER_LENGTH(): Doesn’t work for SQL Server. Use LEN() for SQL Server. This function is used to find the length of a line.

Syntax: SELECT CHARACTER_LENGTH('geeks for geeks');

Output: 15

4. CONCAT(): This function is used to add two words or strings.

Syntax: SELECT 'Geeks' || ' ' || 'forGeeks' FROM dual;

Output: ‘GeeksforGeeks’

5. CONCAT_WS(): This function is used to add two words or strings with a symbol as concatenating symbol.

Syntax: SELECT CONCAT_WS('_', 'geeks', 'for', 'geeks');

Output: geeks_for_geeks

6. FIND_IN_SET(): This function is used to find a symbol from a set of symbols.

Syntax: SELECT FIND_IN_SET('b', 'a, b, c, d, e, f');

Output: 2

7. FORMAT(): This function is used to display a number in the given format.

Syntax: Format("0.981", "Percent");

Output: ‘98.10%’

8. INSERT(): This function is used to insert the data into a database.

Syntax: INSERT INTO database (geek_id, geek_name) VALUES (5000, 'abc');

Output: successfully updated

9. INSTR(): This function is used to find the occurrence of an alphabet.

Syntax: INSTR('geeks for geeks', 'e');

Output: 2 (the first occurrence of ‘e’)

Syntax: INSTR('geeks for geeks', 'e', 1, 2);

Output: 3 (the second occurrence of ‘e’)

10. LCASE(): This function is used to convert the given string into lower case.

Syntax: LCASE ("GeeksFor Geeks To Learn");

Output: geeksforgeeks to learn

11. LEFT(): This function is used to SELECT a sub string from the left of given size or characters.

Syntax: SELECT LEFT('geeksforgeeks.org', 5);

Output: geeks

12. LENGTH(): This function is used to find the length of a word.

Syntax: LENGTH('GeeksForGeeks');

Output: 13

13. LOCATE(): This function is used to find the nth position of the given word in a string.

Syntax: SELECT LOCATE('for', 'geeksforgeeks', 1);

Output: 6

14. LOWER(): This function is used to convert the upper case string into lower case.

Syntax: SELECT LOWER('GEEKSFORGEEKS.ORG');

Output: geeksforgeeks.org

15. LPAD(): This function is used to make the given string of the given size by adding the given symbol.

Syntax: LPAD('geeks', 8, '0');

Output:

000geeks

16. LTRIM(): This function is used to cut the given sub string from the original string.

Syntax: LTRIM('123123geeks', '123');

Output: geeks

17. MID(): This function is to find a word from the given position and of the given size.

Syntax: Mid ("geeksforgeeks", 6, 2);

Output: for

18. POSITION(): This function is used to find position of the first occurrence of the given alphabet.

Syntax: SELECT POSITION('e' IN 'geeksforgeeks');

Output: 2

19. REPEAT(): This function is used to write the given string again and again till the number of times mentioned.

Syntax: SELECT REPEAT('geeks', 2);

Output: geeksgeeks

20. REPLACE(): This function is used to cut the given string by removing the given sub string.

Syntax: REPLACE('123geeks123', '123');

Output: geeks

21. REVERSE(): This function is used to reverse a string.

Syntax: SELECT REVERSE('geeksforgeeks.org');

Output: ‘gro.skeegrofskeeg’

22. RIGHT(): This function is used to SELECT a sub string from the right end of the given size.

Syntax: SELECT RIGHT('geeksforgeeks.org', 4);

Output: ‘.org’

23. RPAD(): This function is used to make the given string as long as the given size by adding the given symbol on the right.

Syntax: RPAD('geeks', 8, '0');

Output: ‘geeks000’

24. RTRIM(): This function is used to cut the given sub string from the original string.

Syntax: RTRIM('geeksxyxzyyy', 'xyz');

Output: ‘geeks’

25. SPACE(): This function is used to write the given number of spaces.

Syntax: SELECT SPACE(7);

Output: ‘ ‘

26. STRCMP(): This function is used to compare 2 strings.

- If string1 and string2 are the same, the STRCMP function will return 0.
- If string1 is smaller than string2, the STRCMP function will return -1.
- If string1 is larger than string2, the STRCMP function will return 1.

Syntax: SELECT STRCMP('google.com', 'geeksforgeeks.com');

Output: -1

27. SUBSTR(): This function is used to find a sub string from the a string from the given position.

Syntax:SUBSTR('geeksforgeeks', 1, 5);

Output: ‘geeks’

28. SUBSTRING(): This function is used to find an alphabet from the mentioned size and the given string.

Syntax: SELECT SUBSTRING('GeeksForGeeks.org', 9, 1);

Output: ‘G’

29. SUBSTRING_INDEX(): This function is used to find a sub string before the given symbol.

Syntax: SELECT SUBSTRING_INDEX('www.geeksforgeeks.org', '.', 1);

Output: ‘www’

30. TRIM(): This function is used to cut the given symbol from the string.

Syntax: TRIM(LEADING '0' FROM '000123');

Output: 123

31. UCASE(): This function is used to make the string in upper case.

Syntax: UCASE ("GeeksForGeeks");

Output:GEEKSFORGEEKS

set operaions in SQL

Operator	**Returns**
UNION	Combine two or more result sets into a single set, without duplicates.
UNION ALL	Combine two or more result sets into a single set, including all duplicates.
INTERSECT	Takes the data from both result sets which are in common.
EXCEPT	Takes the data from first result set, but not the second (i.e. no matching to each other)

SYNTAX

For set operators, the syntax is simple.

```
SELECT [Column_Name, . . . ] FROM [table1] [set operator]
SELECT [Column_Namse, . . .] FROM [table2] [set operator]
...
...
SELECT [Column_Name, . . . ] FROM [tableN]
```

Aggregate functions

In database management an aggregate function is a function where the values of multiple rows are grouped together as input on certain criteria to form a single value of more significant meaning.

Various Aggregate Functions

1) Count()

2) Sum()

3) Avg()

4) Min()

5) Max()

INNER JOIN

employee(emp_id,emp_name,city,salary,age)

project(pjt_id,emp_id,dept)

```
SELECT employee.emp_id,project.dept
FROM employee INNER JOIN project
ON employee.emp_id=project.emp_id;
```

employee(emp_id,empfname,emplname,dept,pjt,address,DOB,gender)

empdesig(emp_id,emp_position,dateofjoin,salary)

1.Qn: find the employees staying in tvm ?

SELECT empfname,emplname

FROM employee

WHERE address=”tvm”;

2.Qn : find all male employees?

SELECT empfname,emplname

FROM employee

WHERE gender=”male”;

3.Qn: find the date of birth of employees working in production department?

SELECT DOB

FROM employee

WHERE dept=”production”;

employee(emp_id,empfname,emplname,dept,pjt,address,DOB,gender)

empdesig(emp_id,emp_position,dateofjoin,salary)

4.Qn: find the position of employees joined on 2/5/1970?

SELECT emp_position

FROM empdesig

WHERE dateofjoin=”2/5/1970 ”;

5.qn: find the salary of all clerks and managers?

SELECT salary

FROM empdesig

WHERE emp_position=”clerk “ OR emp_position=”manager “;

employee(emp_id,empfname,emplname,dept,pjt,address,DOB,gender)

empdesig(emp_id,emp_position,dateofjoin,salary)

6.Write query to fetch the no of employees working in the “HR” department?

SELECT count(*)

FROM employee

WHERE dept=”HR”;

7.Write a query to fetch the first four characters of emplname from the employee table?

```
SELECT  SUBSTRING(emplname,1,4)
FROM employee ;
```

employee(emp_id,empfname,emplname,dept,pjt,address,DOB,gender)

empdesig(emp_id,emp_position,dateofjoin,salary)

8.Write a query to find all employee details whose salary is between 10000 and 40000

```
SELECT  *
FROM empdesig
WHERE salary >=10000 AND salary <=40000;//BETWEEN 10000 AND 40000
```

9.Write a query to find the no of employees whose DOB is between 2/5/1970 and 31/12/1985 and are grouped according to gender?

```
SELECT  count(*),gender
FROM employee
WHERE DOB BETWEEN “2/5/1970” AND “ 31/12/1985”
                                GROUP BY gender;
```

employee(emp_id,empfname,emplname,dept,pjt,address,DOB,gender)

empdesig(emp_id,emp_position,dateofjoin,salary)

10. write a query to create a new table which consists of data and structure copied from employee table?

```
CREATE TABLE  newemployee
AS
SELECT * FROM employee;
```

11.To fetch all records from employee table ordered by emplname in descending order and department in the ascending order?

```
 SELECT *
FROM employee
     ORDER BY emplname desc,dept asc;
```

12.To find details of employees whose name begin with ’s’?

13.To find details of employees whose emplname ends with ‘ A’ and contains five alphabets?

14.To find details of employees whose emplname ends with ‘ A’ and contains five alphabets?

15.To display all employee details excluding the employees with first names ‘sanjay’ and ‘sonia’ from employee table?

To find name of employees whose name begin with s and 4 characters?

SELECT *

FROM employee

WHERE empfname LIKE “s_ _ _ “;

Qn.emplname ends with ‘ A’ and contains five alphabets

SELECT *

FROM employee

WHERE emplname LIKE “----A”

Qn.To display all employee details excluding the employees with first names ‘sanjay’ and ‘sonia’ from employee table?

SELECT *

FROM employee

WHERE empfname NOT IN (“sanjay”,”sonia”);

16.To display all employee details of the employee with first name ‘sanjay’ in employee table?

SELECT *

FROM employee

WHERE empfname LIKE “sanjay”;

employee(emp_id,empfname,emplname,dept,pjt,address,DOB,gender)

empdesig(emp_id,emp_position,dateofjoin,salary)

17.To display the first record from the employee table?

```
SELECT *
FROM employee
WHERE emp_id=
(SELECT MIN(emp_id)
FROM employee);
```

or

```
SELCT *
FROM employee
WHERE rownum=SELECT  MIN(rownum)
                    FROM employee;
SELECT emp_id
FROM (SELECT rowno,emp_id
          FROM employee )
WHERE MOD(rowno,2)=1;odd
=0 even rows
```

18.To display the even(or odd) records from a table?

```
SELECT *
FROM employee E
WHERE EXISTS
              (SELECT *
                    FROM empdesig P
                    WHERE E.emp_id=P.emp_id);
```

19.To retrieve the list of employees working in the same department?

employee(emp_id,empfname,emplname,pjt,dept,address,DOB,gender)

empdesig(emp_id,emp_position,dateofjoin,salary)

```
SELECT  E.empfname,E1.dept
FROM employee E,employee E1
WHERE E.dept =E2.dept AND E.emp_id!=E1.emp_id;
```

20.To add email validation to your database

1.

Pattern	What the Pattern matches
*	Zero or more instances of string preceding it
+	One or more instances of strings preceding it
.	Any single character
?	Match zero or one instances of the strings preceding it.
^	caret(^) matches Beginning of string
$	End of string
[abc]	Any character listed between the square brackets
[^abc]	Any character not listed between the square brackets
[A-Z]	match any upper case letter.
[a-z]	match any lower case letter
[0-9]	match any digit from 0 through to 9.
[[:<:]]	matches the beginning of words.
[[:>:]]	matches the end of words.
[:class:]	matches a character class i.e. [:alpha:] to match letters, [:space:] to match white space, [:punct:] is match punctuations and [:upper:] for upper class letters.
p1|p2|p3	Alternation; matches any of the patterns p1, p2, or p3
{n}	n instances of preceding element
{m,n}	m through n instances of preceding element

```
SELECT email
FROM employee
WHERE NOT REGEXP_LIKE(email,'[A-Z0-9.-%+-]+@[A-Z0-9.-]+\.[A-Z]{2-4}','$');
```

employee(emp_id,empfname,emplname,pjt,dept,address,DOB,gender)

empdesig(emp_id,emp_position,dateofjoin,salary)

21.To retrieve departments who have less than 2 employees working on it?

```
SELECT dept,COUNT(emp_id)  AS Empno
FROM employee                                    dept Empnoe
GROUP BY dept HAVING COUNT(emp_id<2);
```

22.To fetch 50% records from the employee table?

```
SELECT  *
FROM employee
WHERE emp_id<=
        (SELECT COUNT(emp_id)/2
            FROM employee);
```

23.To find the second highest salary ?

employee(emp_id,empfname,emplname,pjt,dept,address,DOB,gender)

empdesig(emp_id,emp_position,dateofjoin,salary)

```
SELECT *
FROM employee A
WHERE 2= (SELECT COUNT(DISTINCT salary)
            FROM employee B
            WHERE A.salary<=B.salary);
```

employee(emp_id,fname,lname,email,phone,hire_date,job_id,salary,manager_id,dept_id);

department(dept_id,dept_name,location_id)

1.Find the employee who have the highest salary?

2.Find all employees who locate in location with id=100?

3.find all employees whose salaries are greater than the average salary of all employees?

4.To find all department which have atleast one employee with salary greater than 1000? (having no employee also)

5.to find the lowest salary by department wise?

6.To find all employees whose salaries are greater than the lowest salary of the department?