1. Displaying something on the screen -
2. E.g. SELECT ‘TEXT To be displayed’
   AS result;
3. SQL helps us interact with the data stored in relational databases.
4. SQL stands for Structured Query Language.
5. Table : Collection of rows and columns. Relational database is a collection of tablesA row is called a record and a column is called a field.
6. Query is request for data from a database
7. We can select from a table using the SELECT statement.
   `SELECT column_name1, column_name2`
   `FROM Table_name ;`
8. You can use star to select multiple columns.
   `SELECT * FROM Table_name ;`
9. You can use limit keyword at the end of the query to get a limited number of records.
   `SELECT * FROM Table_name limit 10;`
10. To select distinct values from a field or column we use the DISTINCT keyword. Distinct does not list null values.
    `SELECT DISTINCT column_name FROM Table_name ;`
11. Count helps you in counting the number of records. COUNT(*) tells you how many rows are there in a table.
    ```
    SELECT COUNT(*)
    FROM table_name;
    ```
11. We can combine count with distinct like
    ```
    Select count( distinct column_a)
    From table_name
    ```
    This will select distinct values of column_a from Table_name.
12. We can count non missing values in a column using Count
    ```
    Select count(column_a)
    From table_name
    ```
13. Filtering results in sql
    Using where keyword we can filter rows on the basis of text and numeric values.
    Various comparison operators :
    =, <>, <, >, <=, >=
    SELECT col_a, col_b
    FROM table_name
    Where col_c = ‘ABCD’
    14. Note : In postgres SQL you must use single quotes with where.

15. We can use and or or with where. You can use multiple where condition using

    And , remember with every and you will have to mention column name
    SELECT col_1, col_2
    FROM table_name
    WHERE col_1 = 'A' AND col_3 >200 ;

16. We use OR when only some, not all conditions are to be met.
17. Remember you do need to specify column when using OR clause.
18. When combining AND and OR enclose individual clauses in bracket.
    SELECT col_1 FROM table_name WHERE (OR CLAUSE) AND (OR CLAUSE)
19. We can use BETWEEN for ranges. Remember! BETWEEN is inclusive
    SELECT col_1, col_2
    FROM table_name
    WHERE col_4 BETWEEN val_1 and val_2
20. IN keyword allows you to specify multiple values in a WHERE clause.
    SELECT col_1
    FROM table_name
    WHERE col_2 IN (2, 3,4,5,6)
21. In SQL NULL represents missing or unknown value.
22. We can check for NULL values using the ISNULL expression.
23. To know about the records that are not missing we use the IS NOT NULL Operator.
24. ISNULL example :
    SELECT count(*) FROM table_name WHERE col_1 ISNULL;
25. IS NOT NULL example :
    SELECT col_3 from Table_name WHERE col_2 IS NOT NULL;
26. LIKE and NOT LIKE :
    LIKE operator is used to search for a string pattern. We use wildcard to search for a pattern. % wild card will match 0, 1 or many characters.
_ wildcard will match a single character.

```
SELECT name
FROM companies
WHERE name LIKE 'Data%';
```

27. LIKE and NOT LIKE :
    LIKE operator is used to search for a string pattern. We use wildcard to search for a pattern. % wild card will match 0, 1 or many characters.
_ wildcard will match a single character.
SELECT name FROM Table_name WHERE name LIKE 'AN%' ;
SELECT name FROM Table_name WHERE name NOT LIKE '_IYA' ;

28. There are various aggregate functions - AVG, MAX, SUM, MIN.
    Example :
    SELECT MAX(num_col_1)
    FROM table_name'
29. We can perform basic arithmetic using operators like +,- * /
    SELECT (4*3)
    Division however will give integer result.
30. We use the keyword AS for aliasing.
    SELECT MAX(budget) AS max_budget, MAX(gross) as max_gross
    FROM table_name
31. SQL example :
    select (max(release_year) - min(release_year))/10 as number_of_decades
    from films
32. ORDER BY is used to sort the result in ascending or descending order.
    According to the values of one or more columns.
    To sort results in descending order use the DESC keyword.

```
SELECT col_name1, col_name2
 FROM table_name
 ORDER BY col_name1 DESC;
```

33. ORDER BY sorts alphabetically.
34. GROUPBY is used to group records it always goes with an aggregate function. GROUP BY always comes after the from Clause.
35. We can use GROUP BY along with ORDER BY
    For example : SELECT col_name1,count(col_name2)
    FROM table_name
    GROUP BY col_3
    ORDER BY count DESC
36. Aggregate functions cannot be used with WHERE clause this is when HAVING keyword comes into picture
Invalid query :

```
SELECT release_year
FROM films
GROUP BY release_year
WHERE COUNT(title) > 10;
```

37. We get all the records where the key col value in one record is equal to key column value in the other record
SYNTAX for inner join :

```
SELECT *
```

```
FROM left_table
INNER JOIN right_table
ON left_table.id = right_table.id;
```

37. Aliasing in inner join :(cities and countries are the table)

```
SELECT cities.name as city, countries.name as country, region
FROM cities
  INNER JOIN countries
    ON cities.country_code = countries.code;
```

38. WITH clause helps us in giving name to a subquery

39. We can use output of a query with the help of WITH clause

```
WITH query_name1 AS (
     SELECT ...
     )
   , query_name2 AS (
     SELECT ...
       FROM query_name1
         ...
     )
```

40. We can use MOD function in sql

Select MOD(4, 3) as result This can be used with WHERE clause.

You can also use modulo operator in sql.

Select 17 % 5;

41. You can use count values for different columns

42. **(INNER) JOIN: Returns records that have matching values in both tables**

- **LEFT (OUTER) JOIN: Returns all records from the left table, and the matched records from the right table**
- **RIGHT (OUTER) JOIN: Returns all records from the right table, and the matched records from the left table**
- **FULL (OUTER) JOIN: Returns all records when there is a match in either left or right table**

43. You can combine multiple joins in a single query.

Example :-- 6. Select fields

```
SELECT c.code, name, region, e.year, fertility_rate,
unemployment_rate
 FROM countries AS c
 INNER JOIN populations AS p
  ON c.code = p.country_code
 INNER JOIN economies AS e
  ON c.code = e.code;
```

We can use using when the keyword is same.

SYNTAX for USING keyword :

```
Select col_name1, col_name2
From table1 inner join table2
using(id)
```

Note : Column id is common in both the tables.

43. You can join a table with itself - This is called selfish join

In selfish join you join a table with itself.

SYNTAX :

```
Select p1.col1, p2.col1, p1.col3
FROM table1 as p1
INNER JOIN table1 as p2
ON p1.key_col = p2.key_col
```

44. You can self join a column on more than one key column use and along with on :

```
Select p1.col1, p2.col1, p1.col3

FROM table1 as p1

INNER JOIN table2 as p2

ON p1.keycol = p2.keycol

AND p1.keycol1 = p2.keycol2
```

45. CASE keyword is like a if/else statement if this then this, if this then this, else this .

Example of case :

```
Select name, continent, size,

CASE WHEN surface_area > 200 THEN 'large'

WHEN surface_area >100 THEN 'medium'

ELSE 'small' END as geosize

FROM COUNTRIES
```

46. INTO command for creating a new table

Remember INTO comes before FROM.

```
 SELECT name, continent, size,

CASE WHEN surface_area > 200 THEN 'large'

WHEN surface_area >100 THEN 'medium'

ELSE 'small' END

INTO countries_plus
```

FROM countries

47. LEFT join : It is one of the most common join after the inner join

You have a left table and a right table. From left table you take all entries, from the right table you take only those entries that are common if suppose for an entry in the left table you do not have a matching entry in the right table then null values are stored in those columns

SYNTAX FOR LEFT JOIN :

Select t1.col1, t2.col2, col3, col4

FROM table1 as t1

LEFT JOIN table2 as t2

ON t1.col1 = t2.col2

Here, we are left joining table1 with table2

48. RIGHT JOIN : RIGHT JOINS ARE SAME AS LEFT JOIN but here you have at least one record for every record in the right table if the keys don't match then you will have null values at the starting.

SYNTAX FOR RIGHT JOIN :

Select c.name, l.name, percent

FROM languages as l

RIGHT JOIN cities as c

ON l.code = cities.country_code

49. A FULL JOIN combines both the left and the right joins. Every record from each table is present. If there is no intersection then the gaps are filled with NULL values.

50. For any date we refer to date in this format

```
 select *

 from rupay_dcb_transactions


where txn_date_time = '2020-01-25'

limit 100;
```

51.SELECT c1.name AS country, region, l.name AS language,

```
      basic_unit, frac_unit

FROM countries AS c1

  FULL JOIN languages AS l

    USING (code)

  FULL JOIN currencies AS c2

    USING (code)

WHERE region LIKE 'M%esia';
```

Note : When you use alias for a table then you use that alias name to pointing out the columns from that table like l.name instead of languages.name in the above example.

52. Cross joins create all possible combinations for two tables.

Cross joins combine all the records in one table to all the records on the other table.

Suppose there are 3 records in table A and there are 4 records in table B. Then when we cross join table A with table B we get 12 rows.

53. Remember CROSS JOIN does not use ON or USING

54. Calculating remainder using % operator can be used with where.

55. We can use LENGTH(string) to calculate the length of the string.

56. Select top rows from sql syntax: select top 4 * from table_name.

57. We can do union of two tables if we want to join two tables row - wise

58. Code for checking the first and last letter for vowel

```
select distinct CITY
FROM STATION
WHERE substring(CITY,1,1) IN ('A','a','e','E','i','I','o','O','u','U')
and substring(CITY,length(CITY),length(CITY))
IN('A','a','e','E','i','I','o','O','u','U')
order by CITY
```

(here STATION is the table and CITY is the column)

Usage of substring: We are using substring(Colname, character_start, character_End)

unlike array indexing starts from 1.

59. Getting columns from a table

```
SELECT COLUMN_NAME
 FROM INFORMATION_SCHEMA.COLUMNS
 WHERE TABLE_NAME ='table_name'
```

60. How to do secondary sort ?

second argument in order by is for secondry sort e.g.

order by col1, col2.

61. How to order by last three strings in a name (suppose name is the name of the column) ?

```
order by substring(name, length(name)-2,  length(name))
```

62. Order by orders in ascending order(increasing order) by default.

63. Working with case statements.

```
 SELECT col_name,
CASE
   WHEN A+B<C or B+C<A or A+C<B or A+B=C or B+C=A or A+C=B THEN 'Not A Triangle'
   WHEN A=B and B=C THEN 'Equilateral'
   WHEN A=B or B=C or A=C THEN 'Isosceles'
   WHEN A!=B and B!=C THEN 'Scalene'
END
FROM TRIANGLES
```

64.

select Concat(Name,'(',substring(Occupation,1,1),')')

from OCCUPATIONS

order by Name;

SELECT 'There are a total of' ,COUNT(Occupation) as c, concat(lower(Occupation),'s.')

FROM OCCUPATIONS

group by occupation

order by c,occupation;

65. "Set" in sql is used to update column values, mostly it is used with update:

e.g.

Update tablename

Set col_name = value

where col_name2 = a

65. Set @r1=2;

This is used to set the value of a local variable

66. FIELD() FUNCTION: Returns index position of a value in a set of values.

Example of FIELD FUNCTION:

```
SELECT FIELD("q", "s", "q", "l");
```

output: 2

67.set @r1=0, @r2=0, @r3=0, @r4=0;

```
select min(Doctor), min(Professor), min(Singer), min(Actor)
from(
  select case when Occupation='Doctor' then (@r1:=@r1+1)
              when Occupation='Professor' then (@r2:=@r2+1)
              when Occupation='Singer' then (@r3:=@r3+1)
              when Occupation='Actor' then (@r4:=@r4+1) end as
RowNumber,
    case when Occupation='Doctor' then Name end as Doctor,
    case when Occupation='Professor' then Name end as Professor,
    case when Occupation='Singer' then Name end as Singer,
    case when Occupation='Actor' then Name end as Actor
  from OCCUPATIONS
  order by Name
) Temp
group by RowNumber
```