CONSTRAINTS AND CLAUSES

SQL_QUERY:

```
create database d1
use d1;
create table people(id int,name varchar(20),country varchar(80)  DEFAULT
'india',age int check(age>= 18));
insert into  people values (1,'a','india',18);
insert into  people values (3,'c','india',28);
insert into  people values (2,'w','pakistan',98);
insert into  people values (10,'k','afghanisthan',87);
insert into  people values (30,'b','canada',35);
insert into  people values (7,'m','canada',43);

select * from people;
create index i1 on people(id);
select * from people order by country desc,name desc;
select top 3*from people;
select  count(name),country from people group by country;
select count(name),country from people  where age>='30' group by country having
count(country)>1 order by country desc;
select count(country),name from people group by name having count(name)>0;
select *into newtable from people;
select * from newtable;
--select country,name into neww  from people where country ='canada'or
country='india';
select * from neww;
select * from people where not country ='india';
select count(name) from people;
select * from people where country in ('india','canada');
select country from people where exists (select name from people where
country='canada' and age>=50);
select country,name from people where country like '%an';
select country,name from people where country like '_a%';
select min(age) from people;
select max(age) from people;
select avg(age) from people;
select sum(age)from people;
truncate table person;
```

CREATE CONSTRAINTS:

Constraints can be specified when the table is created with the CREATE TABLE statement or after the table is created with the ALTER TABLE statement.

```
CREATE TABLE tablename (
 column1 datatype constraint,
 column2 datatype constraint,
 column3 datatype constraint,
);
```

NOT NULL CONSTRAINT:

Ensures that a column cannot have a NULL value.

```
CREATE TABLE tablename (
column1 datatype NOT NULL,
column2 datatype NOT NULL,
column3 datatype NOT NULL…
);
```

UNIQUE CONSTRAINT:

UNIQUE Constraint enforces a column or set of columns to have unique values. If a column has a unique constraint, it means that particular column cannot have duplicate values in a table.

```
CREATE TABLE tablename (
column1 datatype UNIQUE,
column2 datatype UNIQUE,
column3 datatype UNIQUE,..
);
```

DEFAULT CONSTRAINT:

The DEFAULT constraint provides a default value to a column when there is no value provided while inserting a record into a table.

```
CREATE TABLE tablename (
column1 datatype DEFAULT,
column2 datatype DEFAULT,
column3 datatype DEFAULT,..
);
```

CHECK CONSTRAINT:

This constraint is used for specifying range of values for a particular column of a table. When this constraint is being set on a column, it ensures that the specified column must have the value falling in the specified range.

```
CREATE TABLE tablename (
column1 datatype ,
column2 datatype ,
column3 datatype CHECK(column3 condition);
);
```

PRIMARY KEY CONSTRAINT:

A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table.

```
CREATE TABLE tablename (
column1 datatype ,
column2 datatype ,
column3 datatype PRIMARY KEY
);
```

INDEX CONSTRAINT:

The CREATE INDEX statement is used to create indexes in tables.Indexes are used to retrieve data from the database more quickly than otherwise. The users cannot see the indexes, they are just used to speed up searches/queries.

```
CREATE INDEX index_name
ON table_name (column1, column2, ...);
```

SELECT TOP:

The SELECT TOP clause is used to specify the number of records to return.

```
SELECT TOP number|percent column_name(s)
FROM table_name
WHERE condition;
```

MIN AND MAX FUNCTIONS

The MIN() function returns the smallest value of the selected column.
The MAX() function returns the largest value of the selected column.

```
SELECT MIN(column_name)
FROM table_name
WHERE condition;
SELECT MAX(column_name)
FROM table_name
WHERE condition;
```

COUNT,SUM AND AVERAGE FUNCTIONS

The COUNT() function returns the number of rows that matches a specified criterion.
The AVG() function returns the average value of a numeric column.
The SUM() function returns the total sum of a numeric column.

```
SELECT COUNT(column_name)
FROM table_name
WHERE condition;
SELECT AVG(column_name)
FROM table_name
WHERE condition;
SELECT SUM(column_name)
FROM table_name
WHERE condition;
```

LIKE OPERATOR:

The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.There are two wildcards often used in conjunction with the LIKE operator:
% - The percent sign represents zero, one, or multiple characters
_ - The underscore represents a single character

```
SELECT column1, column2, ...
FROM table_name
WHERE columnN LIKE pattern;
```

GROUP BY:

The GROUP BY statement groups rows that have the same values into summary rows, like "find the number of customers in each country".The GROUP BY statement is often used with aggregate functions (COUNT, MAX, MIN, SUM, AVG) to group the result-set by one or more columns.

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

HAVING:

The HAVING clause was added to SQL because the WHERE keyword could not be used with aggregate functions.

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);
```

EXISTS:

EXISTS operator is used to test for the existence of any record in a subquery.The EXISTS operator returns true if the subquery returns one or more records.

```
SELECT column_name(s)
name
WHERE EXISTS
(SELECT column_name FROM table_name WHERE condition);
```