Types of commands

DDL – Data Definition Language(drop, truncate, alter, create, rename, comment)
DQl – Data Query Language(select)
DML – Data Manipulation Language(insert, update, delete, call, lock table, merge, explain plan)
DCL – Data Control Language(grant revoke)
TCL - transaction Control Language (commit, rollback)

1> Create table

```
create table faculty (
     fid integer ,
     name varchar ( 50 ) not null ,
     dept integer ,
     designation varchar ( 3 ) ,
     primary key fid ,
     foreign key ( dept ) references department ,  #(department is some other table)
     check ( fid >= 0 )
) ;
```

2> Data types
  char(n): fixed-length character string
  varchar(n): variable-length character string, up to n
  integer or int: integer
  smallint: short integer
  numeric(n,d): floating-point number with a total of n digits of which d is after the decimal point
  real: single-precision floating-point number
  double precision: double-precision floating-point number
  float(n): floating-point number with at least n digits
  date: yyyy-mm-dd format
  time: hh:mm:ss format
  time(i): hh:mm:ss:i. . . i format with additional i digits for fraction of a second
  timestamp: both date and time
  interval: relative value in either year-month or day-time format

3> DDLs

a> drop table faculty ;

```
b> alter table faculty add room varchar ( 10 ) ;     #add coloumn
   alter table course drop webpage ;                 #delete coloumn
```

4> Project pie

Applying ΠA,C on

A	B	C
1	1	5
1	2	5
2	3	5
2	4	8

returns

A	C
1	5
2	5
2	8

5> Find roll number of students in B.Tech. program

```
select student.roll
from student , program
where student.roll=program.roll and program.ptype =‘‘B.Tech.’’;
```

6> May use and, or and not to connect predicates

```
select coursecode
from offering
where yr = 2018 and instructor = 10;
```

7> SQL allows between operator (includes both)

```
select coursecode
from offering
where yr between 2016 and 2018;
```

8> renaming attribute or table, Keyword as is used

```
select student.roll as rollnumber
from student , program
where student.roll= program . roll and program.ptype =‘‘B.Tech.’’ ;
```

Renaming is necessary when the same relation needs to be used twice
Example: Find names of students whose cpi is greater than that of “ABC”

```
select T.name
```

```
from student as T , student as S
where T.cpi > S.cpi and S.name = ‘‘ABC’’ ;
```

9> Pattern matching

Uses like to match patterns specified using special characters
_ matches any character(use when youre given number of charcter to find, 0ne _ means one character)
% matches any substring(use when you want a string as part of given string anywhere in it)

Eg 1> Find all departments having “Engineering” in its name

```
select * from department
where name like ‘‘%Engineering%’’ ;
```

Eg 2> Find departments with the name “?E”

```
select * from department
where name like ‘‘_E’’ ;
```

10> Tuples in the final relation can be ordered using order by

```
select *
from department
order by name ;
```

Use desc to obtain tuples in descending order

```
select *
from department
order by name desc ;
```

11> Operators union, intersect and except correspond to ∪, ∩, –

Find names of all faculty members and students

```
(select name from faculty)
    union
(select name from student) ;
```

Find names of faculty members not found in students

```
(select name from faculty)
 except
(select name from student) ;
```

12> Aggregate functions

Five operations that work on multisets: avg, min, max, sum, count

Find the average cpi of students

```
select avg(cpi)
from student;
```

Find the total number of students

```
select count (distinct roll)
from student;
```

13> To apply aggregate operations on separate groups, use group by

Find the number of students in each department

```
select count (distinct roll)
from student;
group by dept ;
```

14> In order to select certain groups, use having clause Only those groups satisfying having clause appear in t

Find average grade in each course where number of students is at least 5

```
select coursecode , avg ( grade )
from registration
group by coursecode
having count (roll) >= 5;
```

15> The predicate in having is applied after forming groups whereas the predicate in where is applied before doing so

Find average grade in each course of type 4 where number of students is at least 5

```
select coursecode , avg(grade)
from registration , course
where registration.coursecode = course.code and ctype = 4
group by coursecode
having count (roll) >= 5;
```

16> The predicates is null and is not null can be used to check for null values

find courses that do not have a webpage

```
select code
```

```
       from course
       where webpage is null ;
```

17> Aggregate functions ignore null
count(*) does not ignore nulls

18> Keyword in is used for set membership tests

Find faculty members who have not offered any course

```
       select * from faculty
       where fid not in (select instructor from offering);

    SELECT * FROM Customers
    WHERE Country IN ('Germany', 'France', 'UK');
```

19> Insert rows

```
      insert into student
      values (1897 , ‘‘ABC’’ , 7 , 0. 0 ) ;
```

If value is not known, specify null

```
      insert into student
      values (1897 , ‘‘ABC’’ , 7 , null ) ;
```

To avoid null, specify schema

```
      insert into student(roll,name,dept )
      values (1897 , ‘‘ABC’’ , 7 ) ;
```

Create a course of code 9 for every department with thesame type

```
      insert into course ( code , title , webpage , ctype )
      select 9 , ‘‘New’’ , null , type
      from course
      where type in (select deptid from department ) ;
```

20> Delete

Delete student with roll number 1946

```
     delete from student
     where roll = 1946;
```

If where is empty, all tuples are deleted

Delete all students whose CPI is less than the average CPI

```
delete from student
where cpi < (select avg(cpi) from student) ;
```

21> Update

Update value of grade ‘E’ to 2

```
update grade
set value = 2.0
where gradecode = ‘E’;
```

22> case statement handles conditional updates

Increase CPI of all students by 10% where CPI is less than 6.0, by 5% when less than 8.0, and 2% otherwise

```
update student
set cpi = case (cpi)
when cpi < 6.0 then cpi * 1.10
when cpi < 8.0 then cpi * 1.05
else cpi * 1.02
end ;
```

23> Joins

Equi join is a special type of join in which we use only an equality operator. Hence, when you make a query for join using equality operator then that join query comes under Equi join.

A natural join is a type of equi join which occurs implicitly by comparing all the same names columns in both tables. The join result has only one column for each pair of equally named columns.

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

```
SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate
FROM Orders
INNER JOIN Customers ON Orders.CustomerID=Customers.CustomerID;
```

LEFT (OUTER) JOIN: Returns all records from the left table, and the matched records from the right table

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
```

```
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID
ORDER BY Customers.CustomerName;
```

RIGHT (OUTER) JOIN: Returns all records from the right table, and the matched records from the left table

```
SELECT Orders.OrderID, Employees.LastName, Employees.FirstName
FROM Orders
RIGHT JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID
ORDER BY Orders.OrderID;
```

FULL (OUTER) JOIN: Returns all records when there is a match in either left or right table

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
FULL OUTER JOIN Orders ON Customers.CustomerID=Orders.CustomerID
ORDER BY Customers.CustomerName;
```

24> EXISTS

The 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 SupplierName
FROM Suppliers
WHERE EXISTS (SELECT ProductName FROM Products WHERE Products.SupplierID = Suppliers.supplierID AND Price < 20);
```

25> The SELECT INTO statement copies data from one table into a new table.

```
SELECT * INTO CustomersBackup2017
FROM Customers;
```

26> The INSERT INTO SELECT statement copies data from one table and inserts it into another table.

```
INSERT INTO Customers (CustomerName, City, Country)
SELECT SupplierName, City, Country FROM Suppliers;
```

27> In SQL, a view is a virtual table based on the result-set of an SQL statement. A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database. You can add SQL statements and functions to a view and present the data as if the data were coming from one single table. A view is created with the CREATE VIEW statement.

28> SQL injection is a code injection technique that might destroy your database.

SQL injection is the placement of malicious code in SQL statements, via web page input.

29> TRUNCATE TABLE is similar to DELETE, but this operation is a DDL (Data Definition Language) command. It also deletes records from a table without removing table structure, but it doesn't use the WHERE clause.

```
TRUNCATE TABLE product;
```

30> ACID Properties => The ACID properties, in totality, provide a mechanism to ensure correctness and consistency of a database in a way such that each transaction is a group of operations that acts a single unit, produces consistent results, acts in isolation from other operations and updates that it makes are durably stored

Atomicity
By this, we mean that either the entire transaction takes place at once or doesn't happen at all. There is no midway i.e. transactions do not occur partially.

Consistency
This means that integrity constraints must be maintained so that the database is consistent before and after the transaction.

Isolation
This property ensures that multiple transactions can occur concurrently without leading to the inconsistency of database state.

Durability
This property ensures that once the transaction has completed execution, the updates and modifications to the database are stored in and written to disk and they persist even if a system failure occurs.

31> Normalization is the process of minimizing redundancy from a relation or set of relations. Redundancy in relation may cause insertion, deletion and updation anomalies.

32> ER Model is used to model the logical view of the system from data perspective which consists of these components:

Entity, Entity Type, Entity Set –

33> Extension
The extension of a given relation is the set of tuples appearing in that relation at any given instance. The extension thus varies with time. It changes as tuples are created, destroyed, and updated.

34> Intension
The intension of a given relation is independent of time. It is the permanent

part of the relation. It corresponds to what is specified in the relational schema. The intension thus defines all permissible extensions. The intension is a combination of two things : a structure and a set of integrity constraints.

For example,
Employee(EmpNo Number(4) Not NULL, EName Char(20), Age Number(2), Dept Char(4)
)
This is the intension of Employee relation.