**10\*34**

**MSSQL – DATABASE – 26/10/2023**

**----------------------------------------------------------------------**

**SOFTWARE : DATA TRAVEL**

**WEBSITE / WEB APPLICATION**

**1. DESIGNING – USER INTERFACE – FRONTEND**

**---ENTER YOUR DOB**

**2. RAW INPUT- PROCESS/CALCULATION -> PROGRAMMING LANGUAGE**

**NAME -> SHORT FORM**

**NAME ->**

**3. DATABASE -> FUTURE USE ->**

**DBMS -> Database management system**

1. **RDBMS -> In form of tables**

**MySql, MSSQL, ORACLE, MSACCESS etc**

1. **NO-RDMS -> key-value , hierarchical(Parent-child), file, xml**

**Firebase, MongoDB, SqLite , XML**

**----------------------------------------------------------------------------------------------**

**RDBMS : RDBMS stands for Relational Database management system. RDBMS provides a way to store and manage the data for future use. RDBMS provides security , easy management and multiple ways to access desired data in multiple ways.**

**RDBMS is one of the most popular database management system where data is stored in form of tables.**

**In RDBMS , table is the simplest way to store the data.**

**SQL (Structured Query Language)**

**Turbo C – Application , C Programming**

**VS CODE – APPLICATION , Html,C,Java**

**RDBMS – Application , SQL – language**

**SQL : SQL is a language that is used to communicate with RDBMS. SQL provides some pre-defined queries that is used to store, manipulate(change), and fetch/retrieve/access data from the RDBMS application.**

**PL/SQL – Advanced SQL – Programmable SQL**

**Website**

**----------------------------------------------------------------------------------------------**

**Table : Table is the collection of Rows and columns. In Database language rows are knows as Tuple and columns are known as field. So table is the collection of tuple and field.**

**In RDBMS ,table should be consistent and easy to use. A table should have only logical related data.**

**Table is divided into 2 parts:**

1. **Schema/Design : structure of table that defines maximum number of values that can be stored in table and what should be the types of each values. A well defined schema can result a well mannered record.**
2. **Records/Data : Rows of table that contains data.**

**Firstly we need to create the schema and then we can store records.**

**DBMS -APP, RDBMS-APP , SQL- LANGUAGE, TABLE-EASIEST MANNER TO HAVE DATA**

**----------------------------------------------------------------------------------------------**

**Types of SQL : According to the working of queries, it is divided into 4 categories :**

1. **DDL(Data Definition Language)**
2. **DML(Data Manipulation Language)**
3. **TCL (Transaction Control Language)**
4. **DCL(Data Control Language)**

**DDL(Data Definition Language) : DDL have some queries that is used to perform different operations on schema of the table.**

1. **CREATE : use to create a new database object.**
2. **ALTER : used to manipulate schema.**
3. **TRUNCATE : used to save schema/used to reset schema. It permanently deletes all records but saves the schema .**
4. **DROP : used to remove the schema with all records.**

**--------------------------------------------------------------------------------**

**DML (Data Manipulation Language) : DML have some queries that is used to create or manipulate the records of the tables.**

1. **INSERT : create new records**
2. **UPDATE : Used to change any pre-existing record**
3. **DELETE : delete is used to remove a pre-existing row from the table**
4. **SELECT : used to fetch any desired record from complex table**

**-----------------------------------------------------------------------------------**

**TCL (Transaction Control Language ) :each Transaction may contains multiple changes in database. Every time we need to control the transaction to maintain the consistency of database.**

**TCL contains some queries that is used to maintain the transactions of database.**

1. **COMMIT : save the transaction permanently in database.**
2. **ROLLBACK : is used to undo the recent un-saved transaction in database**
3. **SAVE TRANSACTION / SAVEPOINT : save transaction is used to set mark to the transaction.**

**--------------------------------------------------------------------------------**

**DCL : (Data Control Language) : DCL is used to secure the data with un-authorized access.**

1. **GRANT : Use to provide permissions to the different users of a database.**
2. **REVOKE : is used to take back given permission from the users.**
3. **RENAME : Rename is used to change name of table and database.**

**-----------------------------------------------------------------------------------------------**

**Database : Database is the collection of one entity related table.**

**----------------------------------------------------------------------------------------------**

**Create : Create command is used to create a new database or table. Create command is used to create multiple database objects like stored procedure, indexes, functions, views etc.**

**Syntax to create database :**

**Create** database <database-name>

Syntax to Create table :

Server -> unique database -> unique table -> unique column

Create table <table\_name>

(

Column\_name data\_type(size) ,

Column\_name data\_type(size) ,

-

-

);

**Data Types :**

**Whole Numbers :**

Smallint : 2 byte

Int : 4 byte

Bigint : 8 byte

**Decimal Number :**

Float

Decimal

**String Values :**

Char(0-255) : fixed length memory location

Varchar(0-255) : variable length of memory location

Text

Multi-language string value : nchar(), nvarchar(), ntext

**Date :**

Date : yyyy-mm-dd

Datetime : yyyy-mm-dd hh:MM:ss

**Bit : literal- 0/1**

**---------------------------------------------------------------------------------------------**

**Constraints :** Constraints are some rules and regulations that should be satisfied before adding any value to the column.

Constraints are used to apply extra validation to the column definition. Constraints can be add by using create command and alter command to the column.

1. NOT NULL :
2. CHECK :
3. DEFAULT
4. PRIMARY KEY
5. UNIQUE KEY
6. FOREIGN KEY

create database app23

**Student -> name, mobile no, email, course, branch, total fee**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ram** | **8778766765** | [**ram@g.com**](mailto:ram@g.com) **-Pk** | **btech** | **cs** | **100000** |
| **shyam** | **787667665** | **s@g.com** | **BCA** | **null** | **800000** |

**Submitted\_fee**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr – Identity,PK** | **Userid -Fk** | **amount** | **date** | **mode** |
| **1** | **ra@g.com** | **20000** | **23-10-2023** | **Cash** |
| **2** | **r@g.com** | **50000** | **27-10-2023** | **gpay** |

**Product:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sr -PK** | **title** | **description** | **mrp** | **salerate** | **color** |
| **1** | **T-shirt** | **-----** | **5000** | **2000** | **Black** |
| **2** | **Shirt** | **------** | **3000** | **1000** | **blue** |

**USER**

|  |  |  |
| --- | --- | --- |
| **name** | **Email -PK** | **Mobileno** |
| **Suraj** | **s@gmail.com** | **988987676** |
| **Ram** | **r@gmail.com** | **356543523** |

**CART**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sr-pk** | **Username – FK(user)** | **Product-FK(product)** | **quantity** | **Total\_price** | **date** |
| **1** | **s@gmail.com** | **1** | **4** | **6000** | **23-10-2023** |
| **2** | **s@gmail.com** | **2** | **2** | **2000** | **24-10-2023** |

**Submitted\_fee-> amount , date, mode**

**30000, 23-10-2023 , cash**

**20000, 28-10-2023, netbanking**

**50000, 30-10-2023, gpay**

**-------------------------------------------------------------------------------------------**

**Ex:**

--name-not null, mobno, email---pk,total\_fee=0,>0 ,

create table tbl\_student

(

name varchar(50) not null,

mobileno bigint ,

email varchar(100) primary key,

total\_fee int check (total\_fee>0) default 0 ,

course varchar(50)

)

create table tbl\_fee

(

sr int primary key identity(101,1),

userid varchar(100) foreign key references tbl\_student(email) on delete cascade on update cascade,

amount int default 0,

feedate date,

mode varchar(30)

)

**-------------------------------------------------------------------**

**CREATE-, ALTER-, DROP-, TRUNCATE**

**Insert : Insert command is used to add new records to the table. By using insert a new row is added to the table. It can not be used to add any particular column value to a pre-existing row.**

**Syntax : Insert values to all present columns :**

Insert into table\_name values(values\_1,value\_2,value\_3,……..,value\_n)

Where n is total number of columns in your table.

Identitifier , values/literal/constant value

-----------------------------

Insert into student values(‘’, 0 , 9878767766 , ‘’ , ‘yyyy-MM-dd’)

Insert into student values(null , 0 , 987887787, null , ‘yyyy-MM-dd’)

---------------------------------

**Syntax-2 : Insert values to particular columns of table(let have default values to other columns) :**

Insert into table\_name (column\_name\_1, column\_name\_2, ……, column\_name\_n) values (value\_1, value\_2, ……. , value\_n)

---------------------------------------

Name, sr, mobno, email, regdate - student

Insert into student values()

Insert into student(name, email) values()

Insert into student(email, name, sr) values(‘’, ‘’, 0)

--------------------------------------------------------------

**Syntax to add multiple rows by single insert command :**

Insert into table\_name values(),(),();

Insert into table\_name(sr, name, email) values(value\_1, value\_2, value\_3),( value\_1, value\_2, value\_3),( value\_1, value\_2, value\_3),( value\_1, value\_2, value\_3);

----------------------------------------------------------------------------------------

**Select : Select is a DML command,**  that is used to fetch any part of records from the complex table.

Select is the mostly used query of SQL.

**Syntax :**

Select column\_name\_1, column\_name\_2 from table\_name;

**Syntax to select all columns :**

Select \* from table\_name;

**Select with condition: Conditions are used to filter selected records.**

**Without condition select, returns all saved rows/records of table.**

**------------------------------------------------------------------------------------**

Conditions :

**Where Clause :** Where clause is used to apply conditions that is used to filter out records on which query will execute.

Conditional Operator :

= : returns true where both operands are exactly same

<> : returns true if both operands are different

> : returns true if first operand is greater than second

< : returns true if second operands is greater than first

>= : returns true either first operand is greater or both are equal

<= : returns true either second operand is greater or both are equal

In : returns true if first operand is present in the given list

Not in : returns true if first operand is not present in given list

AND : is used to combine 2 conditions and returns true only if both conditions are true.

OR : is used to combine 2 conditions and returns true only if any one out of 2 conditions are true.

Is : returns true if first operand is null

Is not : returns true if first operand is not null

Between : returns true if first operand value lies between specified range

Like : like is used to compare operand with a given pattern.

select \* from ststudent where regid=1635

select \* from ststudent where email='techpilelko@gmail.com'

select \* from ststudent where tfee=regfee

select \* from ststudent where course<>'other'

> , >=, < , <= : numeric data types

select \* from ststudent where regid>1000

select \* from ststudent where regid>=1000

select \* from ststudent where name>'Ram'

select \* from ststudent where regid<=500

select \* from ststudent where course='Diploma(C.s)'

and

or

condition and condition

(operand operator operand) or condition

and ->

true and true = true

true and false = false

false and true = false

false and false = false

false or false = false

true or false = true

true or true = true

false or true = true

select \* from ststudent where course='Diploma(c.s)' and year='second'

select \* from ststudent where course='Diploma(c.s)' or course='diploma(i.t)'

select \* from ststudent where (course='Diploma(c.s)' or course='Diploma(i.t)') and year='second'

select \* from ststudent where (course='Diploma(c.s)' or course='Diploma(i.t)') and (year='first' or year='second')

select \* from ststudent where (course='diploma(c.s)' and year='first') or (course='diploma(i.t)' and year='second')

select \* from ststudent where picture is null

select \* from ststudent where fathername<>'' is null

select \* from ststudent where fathername is not null

select \* from tbl\_student where 2=2

select \* from ststudent where fathername is not null and fathername<>''

create table tbl\_fee

(

id varchar(1) primary key

)

select \* from ststudent where regid>200 and regid<300

select \* from ststudent where regid between

201 and 299 start\_value and end\_value

select \* from ststudent where course='Diploma(c.s)' or course='Diploma(i.t)' or course='BCA'

select \* from ststudent where course in ('Diploma(c.s)','Diploma(i.t)','bca','aaa')

select \* from ststudent where course<>'Diploma(c.s)' and course<>'Diploma(i.t)'

select \* from ststudent where course not in ('Diploma(c.s)','Diploma(i.t)','BCA')

where column\_name like 'pattern'

special symbols : % , \_

% - any number of characters may present

\_ - any one character should be present

select \* from ststudent where course like 'b.tech%'

select \* from ststudent where college like '%lucknow%'

select \* from ststudent where name like '[a-d][^aeiou]%'

select \* from ststudent where mobno like ''

select \* from ststudent where mobno like '[6-9]\_\_\_\_\_\_\_\_\_'

select \* from ststudent where name like '[a-e]\_\_e\_\_%'

=,>, <,>=,<=,<>,is , is not , between , in, not in , like ,

and , or

select \* from ststudent where regid>200 and regid<400

select \* from ststudent where name like '[aeiou]%' ANd course='b.tech(c.s)'

where course=b.techfirst year second year third year

select \* from ststudent where course='b.tech(c.s)' and year in ('first','second','third')

select \* from ststudent

select \* from ststudent where course like 'B.tech%' and mode in ('offline','online')

0-1000 , b.tech(c.s)-first, b.tech(i.t)-second

select \* from ststudent where regid between 0 and 999 and ((course='B.tech(c.s)' and year='First') or (course='B.tech(I.T)' and year='Second'))

--------------------------------------------------------------------------

Order by Clause : Order by clause is used with select command. Order by is used to arrange the selected records in ascending or descending order on basis of any column.

Syntax :

Select column\_name(s) from table\_name where <condition> order by column\_name asc/desc;

----------------------------------------------------

Aggregate Function : Aggregate functions are some pre-defined functions that is used to perform operation on given parameter and returns a single output.

Mostly aggregate functions are used with group by clause.

-------------------------------------------------

Sum() : used only on numeric columns and returns addition of all values

Avg() : used only on numeric columns and returns average value.

Count() : can be used on any type column or row. It is used to count number of values selected.

Min() : can be used on any type of column and returns minimum present value in that column

Max() : can be used on any type of column, and returns maximum present value in that column.

---------------

select name as 'firstname',[obtained marks]+10 as 'marks' from test

scalar valued function- one value returned

table-valued function - table return

aggregate function -

select sum([obtained marks]),min(name),max([obtained marks]) from test

where course='Diploma/CS'

select sum([obtained marks]) from test where [sr no] between 1 and 50

select \*,sum([obtained marks]) from test

select sum([obtained marks]) from test where [sr no] not in (50,51,52,53,54,55,56,57,58,59,60)

select \* from test where [obtained marks]=

(select max([obtained marks]) from test) or [obtained marks]=(select min([obtained marks]) from test)

--select all record execpt 50-60 srno

select \* from test

select \* from test where [sr no] not in (select [sr no] from test where [sr no] between 50 and 80)

select [sr no] from test where [sr no] between 50 and 80

where [sr no]=[obtained marks]

select sum([sr no]),sum([obtained marks]) from test

sum(param), min(param) , max(param), avg(param) , count(column\_name)

select \* from test where course='Diploma/CS'

select \* from

select max([obtained marks]) from test where course='Diploma/cs'

select \* from test where [obtained marks]=(select max([obtained marks]) from test ) and course='Diploma/CS'

select max([obtained marks]) from test where course='Diploma/CS'

select \* from test where [obtained marks]=(select max([obtained marks]) from test where course='Diploma/CS') and course='Diploma/CS'

select avg([obtained marks]) as 'Avg of Diploma/CS' from test where course='Diploma/cs'

count(column\_name) - total no. of present values in column except null values

count(\*) - total number of selected records

--select minimum marks of csjm college

select \* from test where college='csjm' and [obtained marks]=(select min([obtained marks]) from test where college='csjm')

select count(\*) from test where college='csjm'

select count(\*) from test where course='Diploma/cs' and [Obtained Marks]>20

-------------------------------------------------------------------------

Group By clause : Group by clause is used to divide all records of table in different groups based on a value and you can perform any operation on each groups of table.

Group by clause is mainly used to perform any aggregate operation on different groups of table.

Syntax :

Select column\_name(s) from table\_name where <condition> group by column\_name ;

Select column\_name(s) from table\_name group by column\_name order by column\_name asc/desc;

select \* from ststudent

select mode ,count(\*) from ststudent group by mode

select course,year,count(\*) from ststudent group by course,year

select \* from ststudent

select course,sum(tfee) from ststudent group by course

select \* from ststudent

select count(\*) from ststudent where mode='offline'

select count(\*) from ststudent where mode='offline' and fathername is not null

select count(\*),count(fathername) from ststudent where mode='offline'

What is difference in where and having clause : where is used to apply conditions on table. Where clause is used to filter the records of table.

Having clause is also used to apply condition. But having can be used only after group by clause. Having is used to filter selected records by group by clause.

Where can be used with select , delete, update anywhere but having can be used only with group by.

|  |  |  |  |
| --- | --- | --- | --- |
| rollno | name | college | City |
| 1 | Ram | IET | Lucknow |
| 2 | Shyam | SR | Lucknow |
| 3 | Suraj | SR | Allahabad |
| 4 | Raj | IET | Lucknow |
| 5 | Shivam | AIT | Varanasi |

|  |  |
| --- | --- |
| college | total |
| IET | 1 |
| SR | 2 |

Select from test group by college,city

Select college ,count(\*) as total from student where rollno<=3 group by college having count(\*)>1

----------------------------------------------------

**Common Table Expression(CTE) :CTE is a concept of SQL, when result set of select command react like a table. Actually it is a virtual table and does not have any physical space.**

**Lifetime of table generated by CTE is until the query is executing.**

**So by using CTE we can create a virtual table and further we can use it like a table and perform any operation on that**

**Syntax :**

With cte as (select\_command\_to\_create\_virtual\_table)

Query\_where\_table\_is\_being\_used\_as\_a\_normal\_table

;

with mytable as

(

select college,count([sr no]) as 'total' from test where [sr no] between 1 and 60 and course='diploma/cs' group by college

)select sum(total) as 'total' from mytable

---------------------------------------------------

Joining : Joining is used to select columns from more than table together.

Joining provides mapping of the records with each other between more than one tables.

To join multiple tables together, tables should have relationship to describe that which record will map together.

To tables should have foreign key relation with each other.

4 Types of Joining :

1. Left join
2. Right join
3. Inner join
4. Full outer join

----------------------------------------------------

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| name | mobileno | email | Total\_fee | course |
| Suraj Singh | 9878766778 | suraj@gmail.com | 60000 | Diploma/CS |
| Ram | 9645334455 | ram@gmail.com | 80000 | B.Tech/CS |
| Priya | 7876676556 | priya@gmail.com | 70000 | Diploma/CS |
| Siya | 9667667778 | siya@gmail.com | 80000 | B.Tech/IT |
| Riya | 6776454433 | riya@gmail.com | 30000 | Diploma/CS |
| Ravi Vishwakarma | 8778676556 | ravi@gmail.com | 70000 | B.Tech/IT |
| Kuldeep Singh | 6776688987 | kuldeep@gmail.com | 90000 | B.Tech/CS |

|  |  |  |  |
| --- | --- | --- | --- |
| Userid-FK | amount | feedate | mode |
| suraj@gmail.com | 40000 | 2023-10-23 | Cash |
| ram@gmail.com | 30000 | 2023-10-23 | Cash |
| suraj@gmail.com | 20000 | 2023-10-25 | UPI |
| priya@gmail.com | 40000 | 2023-10-25 | UPI |
| priya@gmail.com | 10000 | 2023-10-26 | CASH |
| priya@gmail.com | 20000 | 2023-10-27 | CASH |
| ravi@gmail.com | 30000 | 2023-10-23 | UPI |
| riya@gmail.com | 10000 | 2023-10-25 | UPI |
| ravi@gmail.com | 20000 | 2023-10-26 | NEFT |

1. select all employees list

Query : select \* from employee\_data

2. Select EEID,name and Job\_title of all employee

Query: select EEID,[full name],job\_title from employee\_data

3. Select all employee list who is 30 to 50 years old

Query: select \* from employee\_data where age between 30 and 50

select \* from employee where age>=30 and age<=50

4. Select EEID,name,Department of all employees who is from United state

Query: select EEID, [full name],department from employee\_data where country='United States'

5. select all employee list in ascending order of name

Query: select \* from employee\_data order by [full name] asc

6. select all employess list in ascending order of department name

Query: select \* from employee\_data order by department

7. select all employees list who is older than 30 year

Query: select \* from employee\_data where age>30

8. select all employee list of IT and Finance department

Query: select \* from employee\_data where department in ('IT','Finance')

select \* from employee\_data where department='IT' or department='Finance'

9. Select all employee list of IT, Finance and Sales department who is older than 40 year

Query: select \* from employee\_data where department in ('IT','Sales','Finance') and age>40

select \* from employee\_data where (department='IT' or department='Sales' or department='Finance') and age>40

10. Select all employee list who is from IT department and lives in United state or China country

Query: select \* from employee\_data where department='IT' and country in ('United States', 'China')

11. Select name and EEID of all employees who is Male and lives in United State

Query: select [Full name],EEID from employee\_data where gender='Male' and country='United States'

12. Select all employee list who is between 30 to 50 age and works in IT or Finance Department.

Query : select \* from employee\_data where age between 30 and 50 and department in ('IT','Finance')

13. Select list of all employees who is older than 50 years and earn less than 100000 annualy.

Query: select\* from employee\_data where age>50 and annual\_salary<100000

14. Select max and min annual salary of employees

Query: select max(annual\_Salary) as 'Max salary' , min(annual\_salary) as 'Min Salary' from employee\_data

15. Select average annual salary of United state employee

Query: select avg(annual\_salary) from employee\_data where country='United States'

16. Select min age of working employee who is from IT department

Query: select min(age) as 'Min age' from employee\_data where department='IT'

17. Select list of all employees who earns maximum annual salary

Query: select \* from employee\_data where annual\_salary=(select max(annual\_salary) from employee\_data)

18. Select list of oldest employee who earns maximum annual salary

Query:select \* from employee\_data where age=(select max(age) from employee\_data) and annual\_salary=(select max(annual\_salary) from employee\_data where age=(select max(age) from employee\_data))

select \* from employee\_data where annual\_salary=(select max(annual\_salary) from employee\_data)

select all records of it student who obtained maximum marks

19. Select list of all countries and total number of employees in each countries

Query:select country,count(\*) from Employee\_Data group by country

select count(\*) from Employee\_Data where country='China'

20. Select list of all Department and total number of employees in every department

Query: select count(\*) from Employee\_Data where Department='IT'

select Department,count(\*) from Employee\_Data group by Department

select \* from Employee\_Data order by Department

21. Select list of all department with maximum and minimum annual\_salary of each department

Query: select department, max(annual\_salary) as 'Max annual salary', min(annual\_salary) as 'Min annual salary' from employee\_data group by Department

22. Select maximum annual\_salary given in every job\_tile

Query: select Job\_Title,max(annual\_salary) from Employee\_Data group by job\_title

23. Select list of every job\_tile where maximum given annual salary is greater than 100000

Query: select Job\_Title,max(annual\_salary) from Employee\_Data group by job\_title having max(annual\_salary)>100000

24. Select list of every country and total number of cities registered in each country in order of country name

Query: select \* from Employee\_Data order by country

select count(distinct city) as 'no of city', country from Employee\_Data group by country

25. Select list of all employees except IT and Sales department

Query: select \* from Employee\_Data where Department not in ('It','sales')

26. Select list of every employee hired in 2019

Query:

select \* from Employee\_Data where Hire\_Date like '%2019%'

**Inner Join : Inner Join is used to select common records from parent and child. Inner join returns only intersection records between more than one table.**

**To apply joining, tables must should have a common column.**

Syntax :

Select table\_name.column\_name(s) from first\_table\_name inner join second\_table\_name on first\_table\_name.common\_column=second\_table\_name.common\_column;

Select user.\*,address.\* from user inner join address on user.emailid=address.emailid

Select user.name, address.address, address.city, address.pincode from user inner join address on user.emailid=address.emailid

Joining result :

|  |  |  |  |
| --- | --- | --- | --- |
| name | Address | city | pincode |
| Shivam | Plot no 43 | Lucknow | 226022 |
| Shivam | 9/345 Vikas nagar | Lucknow | 226022 |
| Ram | Plot 55 | Lucknow | 226022 |

-----------------------------------------------------------

USER : Registration

|  |  |  |  |
| --- | --- | --- | --- |
| name | Emailed -pk | password | mobileno |
| Shivam | s@gmail.com | 123 | 9878776677 |
| Shubham | sh@gmail.com | 8777 | 7876677877 |
| Ram | ram@gmail.com | 6677 | 7887676567 |

Name, mobno, address, picode, city

Inner join : common records

Address ~ user

Select user.\*,address.\* from user inner join address on user.emailid=address.emailid

user.primary\_key\_column=address.foreign\_key\_column

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Emailed - FK | Id – identity-PK | Address | landmark | city | pincode |
| [s@gmail.com](mailto:s@gmail.com) | 1 | Plot no 43 | Jagrani | Lucknow | 226022 |
| s@gmail.com | 2 | 9/345 Vikas nagar | Techpile | Lucknow | 226022 |
| ram@gmail.com | 3 | Plot 55 | Techpile | Lucknow | 226022 |

Create table tbl\_address

(

Emailed varchar(100) foreign key references tbl\_user(emailed) on delete cascade on update set null,

Id int identity(1,1) primary key,

Address varchar(50),

Landmark varchar(100),

City varchar(50),

);

Tbl\_product

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Pid-PK identity | title | description | color | brand | price | quantity |
| 1 | SHIRT | Summer Session wearable | Black | XYZ | 500 | 4 |
| 2 | T-shirt | Summer Session wearable | Black | XYZ | 500 | 4 |

Tbl\_cart

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Cart\_id - identity, PK | quantity | Pid -FK(product) | Userid -FK (user) | date | total |
| 1 | 2 | 1 | ram@gmail.com | 2023-10-20 | 1000 |
| 2 | 2 | 1 | s@gmail.com | 2023-10-22 | 1000 |
|  |  |  |  |  |  |

Update + Alter :

Update :manipulate record

Alter : manipulate schema(add a column, delete a pre-existing column, change definition of column)

----------------------

Update+condition, delete+condition, select+condition

Delete : to remove any row of table with condition.

Delete is a DML command that is used to remove any pre-existing record from the table. Delete command does not remove records permanently, records removed by delete command can be rollback(recover the deleted records).

Delete command can be used with condition.

Note : delete can not be used to remove any particular column of a row. It will remove whole row atleast.

Syntax :

Delete from table\_name; //delete all records of table

Delete from table\_name where <condition> ; // removes only those records which meets the condition

Select column\_name(s) from table\_name where <condition>

delete from table\_name

----------------------------------------------------------------------------------

Difference in Update & Alter

Difference in Delete & Truncate

**Update : Update command** is used to manipulate any cell value of the table.

Update is used to add, remove or change any pre-existing value in any cell of the table.

Syntax :

**Update table\_name set column\_name=value ;**

**Update table\_name set column\_name=value where <condition> ;**

Note : Without condition, update will change column of every row of table.

Condition : Update, delete, select

Sp\_rename

Alter -> add a column, delete a column, change definition of column(data type & size)

Sp\_rename ‘’,’’

Sp\_rename ‘’,’’,’column’

Alter table table\_name add/drop/alter

---------------------------------------------------------------------

Alter : Alter is used to manipulate schema of table.

**Add a column by using alter : by using add keyword**

Alter is used with add keyword when we want to add a new column or constraint to the schema.

**alter table table add column\_name data\_type(size)**

alter table tbl\_student add college varchar(50) default 'IET Lucknow'

**Delete a column by using alter :by using drop keyword :**

Alter is used with drop keyword when we want to remove a column or a constraint from the table.

**Alter table table\_name drop column column\_name**

**Alter table table\_name drop constraint constraint\_name;**

alter table tbl\_student drop column college

alter table tbl\_student drop constraint DF\_\_tbl\_stude\_\_colle\_\_607251E5

select \* from tbl\_student

delete -> we can not use delete to remove a column, itll remove a whole row

truncate -> delete all record , you can not use conditions with truncate

alter -> only apply on schema

select \* from tbl\_student where email=''

delete from tbl\_student

truncate table tbl\_student

select \* from tbl\_fee

delete from tbl\_fee

truncate table tbl\_fee

update tbl\_student set mobileno=null where email='riya@gmail.com'

select \* from tbl\_student

update tbl\_student set mobileno=987876676 where email='ravi@gmail.com'

alter table tbl\_student add college varchar(50)

default 'IET Lucknow'

alter table tbl\_student drop column college

alter table tbl\_student drop constraint DF\_\_tbl\_stude\_\_colle\_\_607251E5

alter table tbl\_student add constraint constraint\_name check (condition)

alter table tbl\_name add constraint constraint\_name default 'techpile' for column\_name

alter table tbl\_name add constraint constraint\_name primary key(column\_name)

select \* from tbl\_student

ALTER – ADD

Syntax : alter table table\_name add column\_name data\_type(size) properties;

Alter table table\_name add constraint constraint\_name primary key(column\_name);

Alter table table\_name add constraint constraint\_name default <value> for column\_name;

Alter table table\_name add constraint constraint\_name check (condition);

Alter table table\_name add constraint constraint\_name foreign key(column\_name) references parent\_table\_name(primary\_column);

--------------------------------------------------------

Alter – drop

Alter table table\_name drop column column\_name

Alter table table\_name drop constraint constraint\_id

---------------------------------------------------------

Alter – alter : used to change definition of column

Alter table table\_name alter column\_name new\_data\_type(size) not null;

execute sp\_rename 'tbl\_student','table\_student'

sp\_rename 'table\_student.mobileno','mobno','column'

alter table tbl\_student alter column name varchar(100)

alter table tbl\_student drop column column\_name

alter table tbl\_student add column\_name data\_type(size)

column - alter-drop, alter-alter

SP\_RENAME

alter -modify : defintiion change

alter - change : NAME

-------------------------------------------------------

**Case : Case is used to handle conditions in sql queries. Case is used to return different value based on the different condition in SQL. Case is same as if-else statements of programming. Case can be used with select , update to access different values based on the different given conditions in same statement.**

**Case with single condition :** Case can be used with single pair of when – then statement just as if statement. It returns given expression with then if condition meets otherwise it returns NULL value.

**Syntax :**

Case when <condition> then statement end //single condition

**Ex:**

Select \*,case when course like ‘btech%’ then ‘graduate’ end as ‘degree’ from tbl\_student

-----------------------------------------------

**Case with if-else statement :** case can be used with one pair of when-then statement following with a else statement. It returns statement of then when condition is true otherwise returns else statement as value.

Syntax:

**Case when <condition> then statement else statement end**

**Ex:**

**Select \* ,**case when course like ‘b%’ then ‘graduate’ else ‘diploma’ end as ‘degree’ from tbl\_student

----------------------------------------------------

**Case to return different value based on different conditions :** case can be used with multiple pairs of when-then statements to returns different values based on different given conditions. It is same as else-if ladder statement of programming.

**syntax :**

Case when <condition> then statement

when <condition> then statement ……

else statement

end

select \*,case when course like 'btech%' then 'graduate' else 'diploma' end as 'degree' from table\_student

select \*,case when mode='cash' then 'cash' else 'online' end as 'Payment method' from tbl\_fee

select \* from table\_student

ravi-8998776655

riya-7678877668

update table\_student set mobno=(case when email='ravi@gmail.com' then 8998776655 when email='riya@gmail.com' then 7678877668 else mobno end)

--------------------------------------------------------------------------------------

Commit ; //ctrl+s

Begin transaction

//statement /queries

Delete from table\_student where email=’kuldeep@gmail.com’

Commit;

-----------------------------------------------------------------------------------------

begin transaction

delete from table\_student where email='kuldeep@gmail.com'

commit;

rollback;

----------------------------------------------------------------------------------------

**9918998665**

**9838643662, 9415241704**

**---------------------------------------------------------------------------------------**

Create , alter, truncate , drop

Insert , update , delete , select

INNER JOIN :

Table\_1(student)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Rollno -PK | NAME | COLLEGE | COURSE | BRANCH |
| 1 |  |  |  |  |
| 2 |  |  |  |  |
| 3 |  |  |  |  |
| 4 |  |  |  |  |
| 5 |  |  |  |  |

Table\_2(test)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Testid -PK | Rollno | | date | | obtainedmarks | | testtile |
| 1 | 1 | |  | | 30 | | C Prog. |
| 2 | 2 | |  | | 25 | | C Prog. |
| 3 | 7 | |  | | 40 | | C Prog. |
| 4 | 2 | |  | | 20 | | HTML |
| INNER JOIN | | LEFT JOIN | | RIGHT JOIN | | FULL OUTER JOIN | |
| 3 | | 6 | | 4 | | 7 | |

Select table\_1.\* , table\_2.\* from table\_1 full outer join table\_2 on table\_1.rollno = table\_2.rollno where table\_2.obtainedmarks>20

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Roll\_no | name | college | course | branch | Testid -PK | Rollno | date | obtainedmarks | testtile |
| 1 |  |  |  |  | 1 | 1 |  | 30 | C Prog. |
| 2 |  |  |  |  | 2 | 2 |  | 25 | C Prog. |
| 2 |  |  |  |  | 4 | 2 |  | 20 | HTML |
| 3 |  |  |  |  | null | null | null | null | null |
| 4 |  |  |  |  | null | null | null | null | null |
| 5 |  |  |  |  | null | null | null | null | null |
| null | null | null | null | null | 3 | 7 |  | 40 | C Prog. |

Left join= inner join+all unmatched record of first table, mapped with null value

Right join = inner join + unmatched records of second table , mapped with null value

Full outer join = inner join + unmatched records of first table mapped with null value + unmatched record of second table mapped with null value

Inner join + left exclusive join + right exclusive join

---------------------------------------------------------------------------------------

Char = 1 byte = 8 bit

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

2^8-1

|  |  |
| --- | --- |
| 0 | 0 |
| 0 | 1 |
| 1 | 0 |
| 0 | 1 |
|  |  |

=0

2^2-1=4-1=3

4 byte = 32 bits

Integer =4294967296

Text = 4294967296-1 characters

Varchar = 1 byte = 8 bit

2^8-1 = 256-1=255

--------------------------------------------------------------------------------

STORED PROCEDURE

INDEX

VIEW , FUNCTION

FIREBASE, MONGODB – PRE-DEFINED FUNCTION

----------------------------------------------------------------------------

CUSTOMERS -> Registration , Login , product search , order

VENDORS -> Registration, Listing (Products add)

1. Customers
2. Categories
3. Employees
4. Shippers
5. Supplier
6. Product
7. Orders
8. Orderdetails

Orders -> Parent -> Customers , Employees, Shippers

Orderdetails -> Parent -> Orders , Product

Product -> Parent -> Suppliers, Category

Create table orders

(

Orderid int primary key identity(10000,1),

Customerid int foreign key references customers(customerid) on delete cascade on update cascade,

Employeeid int foreign key references employees(employeeid),

Orderdate date,

Shipperid int foreign key references shippers(shipperid)

)

Order place -> orders, orderdetails

1. Select all customer records in order of name

Select \* from customers order by customername

1. Select all customer records in ascending order of country name

Select \* from customers order by country

1. Select all customer records in descending order of customer id

Select \* from cutomers order by customerid desc

1. Select customer id, customer name and contact name of all customers

Select customerid, customername, contactname from customers

1. Select all customer records from Germany and Mexico

Select \* from customers where country in (‘Germany’,’Mexico’)

1. Select all customer records between 1 to 60 customer id

Select \* from customers where customerid<=60

Select \* from customers where customerid between 1 and 60

1. Select all employee whose employee id is 4

Select \* from employees where employeeid=4

1. Select all Category id, category name in ascending order of category name

Select categoryid, categoryname from categories order by categoryname

----------------------------------------

1. Select how many total unique categories are registered in table

SELECT count(distinct categoryname) FROM Categories;

1. Select how many total employees are registered

Select count(\*) as ‘Total employees’ from employees

1. Select every city name and total number of supplier registered in each cities

Select city, count(suppliername) as ‘total supplier’ from supplier group by city

1. Select country name and total number of supplier registered in every country

**SELECT country, count(\*) as 'Suppliers' FROM Suppliers group by country;**

-------------------------------------------

JOINING :

1. Select customer id, customer name and orderdate of all orders

Which join - left join (orders is left table), common column – orders-customerid(FK) , Customers(Customerid)PK

Select c.customername , o.customerid, o.orderdate from orders o left join customers c on c.customerid=o.customerid

14. Select all Shipper id, shipper name , order date of every orders

15. Select every order id and total quantity of products in every order from orderdetail table

16. Select every order id with total number of products and total amount of every orders

17. Select Product name , price of every product

18. Select product id, product name, price , Supplier id , supplier name , contact name, address, city of each products

19. Select product id, product name, price, and category name of each products

20. Select category name and number of products added in every category

21. Select every supplier name and total number of added products of every suppliers

----------------------------------------------------

22. Select product name which have maximum price

23. Select product detail which have minimum price

24. Select category name which have less than 20 products registered

Select category.categoryname, count(\*) as ‘Total products’ from category left join products on products.cid=category.cid group by category.categoryname

25. Select all product list which price is between 10 to 20rs.

1. Customers , 2. Categories 3. Employees 4. Shipper

5. Supplier 6. Products 7. Orders 8.orderdetail

Raj#1234

1/5 ratio :

5 - 1