Chapter 4 : SQL (Structured Query Language) 15-20
19705 Queries.

SQL is a standard language for accessing and manipulating data bases.

Features of SQL

1. High performance.
2. Scalability and flexibility.
3. Robust transactional support.
4. High security.
5. Management eare.
6. Open source.

SQL vs MySQL

SQL	MySQL
1) SQL is a language for accessing and manipulating data bases.	1) MySQL is a software or an application.
2) To enable the creation of data management system	2) To enable data handling, storing, modifying, deleting etc.
3) SQL codes & commands are used in various DBMS and RDBMS	3) My SQL has SQL at its Core and is dependent on SQL for future updates.

Various Syntax in SQL

1. Create Database

Syntax : CREATE DATABASE Database Name;

Example: CREATE Database test DB;

2. Drop Database :

DROP DATABASE Database Name;

Ex:- DROP Database testDB;

3) Select Database:
USE DatabaseName;

4 Show Database:
SHOW DATABASES;

5. Create Table:
CREATE TABLE table_name (column1 datatype, column2 datatype, ..., columnN datatype, Primary key (1 or more column));

6) Drop Table:
DROP TABLE table_name;

7) Insert Data into Tables:
INSERT INTO TABLE_Name (column1, ... columnN) VALUES (value1, value2, value3, ... valueN);

8) Select statement:
Select column1, column2 ... column from table_name.

9) Alter Table:
i) TO add a New column in an existing table:
ALTER TABLE table_name ADD column_name datatype;
ii) TO DROP COLUMN in an existing table:
ALTER TABLE tbs table_name DROP column column_name;
iii) TO change the datatype of a column:
ALTER TABLE table_name MODIFY COLUMN column_name datatype;
iv) TO add a NOT NULL constraint to a column in a table:
ALTER TABLE table_name MODIFY column_name datatype NOT NULL;

10) SQL Distinct clause.
Select Distinct col1, col2, ... coln from table_name;

11) SQL where clause
Select Distinct col1, col2, ... coln from table_name where condition;

12) SQL AND/OR clause.
select col1, col2, ... coln from table_name where condition 1 and/or condition2;

13) SQL IN clause
select col1, col2, ..., coln from table_name where column IN (value1, value2, ...);

14) SQL BETWEEN CLAUSE
Select col1, col2, ... coln from table_name where column_name BETWEEN val-1 AND val-2;

15) SQL ORDER BY clause.
Select col1, col2, ... coln from table_name where column_name ORDER BY col_name ASC/DESC;

Several Parts of SQL.

1) DDL : Data definition language.
2) DML : Data manipulation language.
3) DCL : Data control language.

1) DDL : CREATE / DROP / ALTER.
2) DML : SELECT / UPDATE / INSERT / DELETE.
3) DCL : GRANT / REVOKE.

Pros / cons of SQL (Advantages / Disadvantages)

* Advantages.
1. High performance and speed.
2. Well defined standard exist.
3. No coding is required to certain extent.

* Disadvantages
1. Difficulty in interfacing.
2. More features implemented in monetary way.

Database : Customer Table_name : Customer_info.

Customer_id	Customer_name	Contact_name	Address	City	Postalcode	State
1	C07					

Database: customer Table_name: customer_info.

Customer_id	Customer_name	contact_name	Address	City	Postal_code	State.
1	Co7	Cristiano Ronaldo	Italy	Milan	015	One.
2	MS10	Messi	Spain	Barcelona	017	T400
3	bruno10	Bruno	England	Manchester	011	Three
4	ibra9	Ibra	Italy	Milan	015	One.
5	Sancho7	Jordan Sancho	France	Paris	018	Three.

Q) Write SQL queries for the following cases.

1. Create a database named Customer.

=> Create database Customer;

2. Create a table named customer_info

OR

Create a table named customer_info with customer_id as integer, customer_name as varchar, contact_name as char, address as char, po city as char, postal code as int, and state as char.na

=> Create table customer_info(Customer_id int, Customer_name varchar(25), contact_name char(20), address char(20) char(20), city char(25), postal_code int not null, state char(20), primary key(customer_id));

3. Insert the above data into the table.

-> Insert into customer-info values (1, 'cr7', 'Cristiano Ronaldo', 'Italy', 'Milan', 015, 'One');

-> Insert into Customer_info values (2, 'MS10', 'Messi', 'Spain', 'Barcelona', 017, 'Two');

-> Insert into customer_info values (3, 'bruno10', 'Bruno', 'England', 'Manchester', 011, 'Three');

-> Insert into customer_info values (4, 'ibra9', 'Ibra', 'Italy', 'Milan', 015, 'one');

-> Insert into customer_info values (5, 'sancho7', 'Jordan Sancho', 'France', 'Paris', 018, 'Three');

4. Display all the data from the table.

=> select * from Customer_info;

5. Display customer_name and city from the table.

=> Select customer_name, city from Customer_info;

Select statement

Select * from Customer_info;

-> The Select DISTINCT statement is used to return only distinct (different) values. It removes duplicate values.

Syntax:

SQL Distinct Clause.

Select Distinct col1, col2, ..., colN from table_name;

Eg:-

Select state from Customer_info; Displays 5 values

Select DISTINCT state from Customer_info;

Displays One Two Three

WHERE Clause.

SQL where clause.

Select ~~distinct~~ col1, col2, ..., colN from table-name where condition;

Eg:- Select * from Customer_info where state = 'one'; # Displays 1 & 4 datas.

Select * from Customer_info where customer_id = "xxxx" & 2;

Some other operators used in WHERE clause

= Equal

!= or <> Not equal

> Greater than

< Less than

>= Greater than or equal to

<= Less than or equal to.

BETWEEN Between an inclusive range.

LIKE search for a pattern

IN To specify multiple possible values for a column.

AND/OR Clause.

Select col1, col2, ..., colN from table_name where condition1 and/or condition2;

Eg:- Select * from Customer_info WHERE state = 'One' AND state = 'three' City = 'Milan';

Select * from Customer_info where state = 'one' OR state = 'three';

NOT

Eg:- Select * from Customer_info where NOT state = 'one';

ORDER BY

Select col1, col2, ..., colN from table-name where column_name ORDER BY col_name ASC / DESC;

- The orderBY keyword is used to sort the result in ascending or descending order.

The order by keyword sort the records in ascending order by default. To sort the records in descending order we use the DESC keyword.

Example: Select * from customer_info ORDER BY state;

Displays states alphabetically

one
one
three
two

Select * from customer_info ORDER BY state DESC;

SQL IN clause.

Select col1, col2, ..., colN from table_name where column_name IN (value1, value2...)

Ex: Select * from Customer_info where CITY IN ('Milan', 'Paris');

SQL BETWEEN clause.

Select col1, col2, ..., colN from table_name where column_name BETWEEN val-1 AND val-2;

9)# ALTER Table

- Alter table statement is used to add comma, delete or modify columns in an existing table.
- It is also used to add or drop various constraints on an existing table.

Example: Add Phonenumber in Customer_Info.

i) To add a new column in an existing table.

Syntax:-

ALTER TABLE tablename ADD column column_name;

Eg: ALTER Table Customer_Info ADD Phonenumber int;

C_id	C_name	con_name	add	city	P.C.	state	Phone number

ii) To drop column in an existing table.

Eg:- Alter table customer_info DROP column Phonenumber;

iii) To change the DATA TYPE of a column.

Alter TABLE table_name MODIFY column column_name datatype;

Imp

Update Statement.

- It is used to modify or manipulate the existing record or data in a table or a relation.

Syntax:

```
Update table_name
Set column1 = value1, column2 = value2, ....
, columnn = valueN where condition;
```

Example:

The following SQL statement update the first customer (Customer_id = 1) with a new contact_name and a new city.

```
Update Customer_info
Set contact_name = 'XYZ', city = 'abc'
where Customer_Id = 1;
```

Customer_id	Customer_name	Contact_name	Address	City	Postal code	State
1	C07	XYZ	Italy	abc	014	One.

Delete Statement.

- It is used to delete existing records in a table.

Example: Delete * from Customer_info;

Example: Delete from Customer_info where customer_username = 'ms10';

SQL Like operator

% : The percent sign represents zero, one or multiple characters.

_ → The underscore represents a single character.
a% : Finds any values that starts with a.
%a : Finds any values that ends with a.
%or% : Finds any value that have 'or' in any position
_r% : That have 'r' in the second position
a%o : Starts with 'a' and ends with 'o'.
a__%__% : That start with 'a' and are at least 3 characters in length.

Example: Select * from Customer_Info
where ~~Customer~~ Contact_name = '%a';

Aggregate function.

Database : Products.
Table_name: Products_info.

ProductID	Name	SupplierID	categoryID	unit(in packs)	Price
1	Beans	1	1	10	200
2	cauliflower	1	2	20	400
3	cabbage	1	2	10	350
4	Carrot	2	3	30	150
5	Ginger	2	3	40	350.

Some of the aggregate functions are:-

i) Maximum : Max()
ii) Minimum : Min()
iii) Count()
iv) Average() : Avg()
v) Sum()

1) Maximum : Max()
Syntax: Select MAX(column_name) from table_name
where condition;

E.g:- Select Max(price) from Products_info;
output = 400

ii) Eg Minimum: Min()
Syntax: Select MIN(column_name) from table_name where condition;
Eg:- Select Min(Price) from Products_info;
O/p = 150.

iii) Count(): Returns the no. of rows that matches a specified criteria.
Syntax: Select Count(column_name) from tablename where condition;
Eg:-
Select Count(Price) from Products_info;
Select count(ProductID) from Products_info;

iv) Retu Average(): Returns the average value of the numeric column.
Eg:-
Select ~~average~~ avg(price) from Products_info;

v) Sum(): Return the total sum of the numeric column.
Eg:-
Select sum(price) from Products_info;

BETWEEN.
Eg: Select * from Products_info where price between 250 and <150;

//table

Q2 Make a relation below using constraints
company (company_name, city, salary)
constraint to be used

a. company_name should be primary key
b. company_name should not be null.
c. salary should be greater than 5000.
(CHECK constraint).

company
CREATE TABLE company (company_name varchar(25),
city char(20), salary decimal(7,2)
primary key (company_name));
check (salary > 5000), primary key
(company_name));

OR
CREATE TABLE company (company_name
char(30) not null primarykey, city varchar(30),
salary decimal(7,2) check (salary > 5000));

Q) Make a table below using constraint
employee (emp_id, empname, street)
constraints to be used.

a. emp_id should be primary key.
b. Street should not be null.
c. Employee name should begin from 'C'.

Create table employee (emp_id int primary key,
empname char(30) check (empname like 'C%'),
street varchar(30) not null);

Q, Consider the relational database
Employee (Empname, street, city)
Works (Empname, Cmpname, salary)
Company (Cmpname, city)
Manages (Empname, Cmpname)

Write SQL for.

(i) Modify the database so that Amrit now lives in Banepa.

(ii) Delete all the tuples in the works relation for employees of XYZ corporation.

(iii) Increase salary of all employees of ABC company by 10%.

i) Ans

```
Update Employee
set Empname = 'Amrit' city = 'Banepa'
where
    Empname = 'Amrit';
```

Example: Modify the database so that Amrit now lives in Pokhara and in street = 2.

```
Update employee set street = 2,
set city = 'Pokhara', street
where
    Empname = 'Amrit'
```

ii) Delete from works where Cmpname = 'XYZ corporation'

iii) Update works
set salary = salary + 10/100 of salary or 1.1* salary
where Cmpname = 'ABC company';

Group By: It is used to group the result of a select query based on one or more columns. It is also used with SQL functions to group the result from one or more tables.

Syntax:

```
select col1, col2, .... colN from table_name
where column_name GROUP BY col_name
ASC/DESC;
```

Table or Relation name: Employee.

empid	name	age	salary
501	Abc	22	9000
502	Efg	29	8000
503	Hij	34	6000
504	KLM	44	9000
505	NOP	35	8000

Here we want to find name and age of employees grouped by their salaries. End result we will get a dataset with unique salary listed with the 1st employees name and age to have that salary

E.g:- select name, age from employee Group by salary;

output:

name	age
Hij	34
Efg	29
Abc	22

Select name, salary from Employee where age > 25
group by salary;

output:

name	salary
Hij	6000
Efg	8000
KLm	9000

HAVING clause:

- It is used to provide more precise condition for a statement.
- It is used to mention condition in group by best SQL queries just like where clause in SQL query.

Syntax:- Select column_name /function (column_name)
from table_name
where condition
Group by column_name
Having function (column_name) condition;

Table or Relation name: sales.

orderID	order-name	previous-balance	customer
11	ord1	2000	abc
12	ord2	1000	cde
13	ord3	2000	fgh
14	ord4	1000	cde
15	ord5	2000	abc

displays 1st row of abc

E.g:- Select *
from Sales Group by customer
HAVING SUM (previous_balance) >3000;

orderid	order_name	previous_balance	customer
11	ord1	2000	abc

Q. How does Group by clause work? What is the difference betn where and Having clause? Explain each with example.

SQL query:- It is a request to access data from the database to manipulate it or retrieve it.

* SQL sub query: A sub query or inner query or a nested query is a query within another SQL query and embedded within the where clause.
Sub queries can be used with the SELECT, INSERT, UPDATE AND DELETE along with operators like =, <, >, <=, >=, IN, BETWEEN etc.

e.g.

```
Select * from customers
where id in(select id from customers
where salary >4500);
```

↓ execute

Embedded SQL

Account (Account_no, branch_name, Amount)

if (n==1)

Embedded SQL

Account (Account-no, branch-name, Amount)

```
if (n==1)
    select Account-no from Account;
else if (n==2)
    select branch-name from Accound;
else if (n==3)
    select Amount from Account;
else
    select * from Account;
```

- Embedded SQL is a method of combining the computing power of programming language and database manipulation capabilities of SQL. Access to a database from a general purpose programming language is required because:-

(1) SQL is not powerful as other programming language. There are queries that cannot be expressed in SQL but can be programmed in programming language like C, C++, Java, ~~dotnet etc~~, .Net &.

(2) Non-declarative action such as printing, interacting with users or sending result to GUI can't be done in SQL.

SQL standard defines embedding of SQL as embedded SQL. The language in which SQL query is embedded is known as host language.

Imp

Stored procedures.

- A stored procedure is a prepared SQL code that you will save so that the code can be reused over and over again
- It is a subroutine like a subprogram in a regular computing language stored in database
- In case you have a SQL query that you will write over and over again, save it as a stored procedure and then just call to execute it.
- Parameters can also be passed in a stored procedure so that the stored procedure can act based on the parameter that is passed

Example:

Stored Procedure Syntax:

```
CREATE PROCEDURE procedure_name
AS
sql_statement
GO;
```

Execute a stored procedure

```
EXEC procedure_name;
```

Example:

```
CREATE PROCEDURE SelectAllCustomers
AS
Select * FROM CUSTOMERS
Go;
EXEC SelectAllCustomers;
```

* Advantages:

1. SQL stored procedures are more reliable than equivalent external procedure.

2) Supports passing a parameter. Thus are dynamic in nature.
3) Easy to implement because they use a simple high level language.
4) Reside in the database and are automatically backup and stored.
5) Support a simple but powerful condition and error handling model.

* Disadvantages:
1) Stored procedure languages are vendor-specific. Therefore if we switch to another vendor database it requires rewriting the existing stored procedure.
2) Languages from different vendors have different level of sophistication.
for eg:- Oracle SQL has more features than microsoft SQL.
3) Tools support for writing and debugging stored procedure is often not so good.

Imp Q) Define stored procedure. List the advantages & disadvantages of stored procedure. Explain how stored procedure are created with example.

Imp ## QBE (Query By Example)
- It is another language for doing query in relational databases.
- It provides user friendly graphical interface for manipulating databases.
- Without QBE an user needs to write input command which may result in error if the syntax is not correct.

- A QBE features provides a simple interface for a user to enter queries.
- Instead of writing an entire SQL command the user can fill in blanks or select items to define the query that they want to perform.

for eg:- A user may want to select an entry from a table called 'Table1' with an idea of ID of 123.

Using SQL,

Select * from Table1 where ID=123;

Using QBE,

- The QBE interface may allow the user to just click on 'Table1' type in 123 in the ID field and click "search";

Table	ID	
Table1	123	Search

↑ click on it.

Fig: QBE interface

Purpose of QBE is to make easier to run database queries and to avoid frustration of SQL syntax error.

Example QBE:

loan (loan_number, branch_name, amount)

→ To display all loan(~~no~~ loan_number having ~~bra~~ balance greater than 10,000.

NOTE: P. denotes or used to select entire content of relation.

P. ALL denotes or select distinct content in the attribute.

Operators {=, >, <, ¬ (is not equals to)

Solution

loan	loan_number	branch_name	amount
	P.		>100 00

Q.(2) To display all branch name distinctly having balance greater than 10000

Soln

loan	loan_number	branch_name	amount
		P.A/C	>10000

Q.(3) Find the loan_number of all branches that are not located in Satdobato.

Soln

loan	loannumber	branch_name	amount
	P.	~Satdobato	

7) Write short notes on QBE.

8) Write down the o/p of following query with reference to Emp relation.

Ename	Manager	Empno
Robit Ejan	M1	
Rohit	M3	
Rohan		
Rohan		

Q2 Write down the output of the following query with reference to emp relation.

Relation: emp

Ename	Manager	Empno
Ejan	M_1	e56
Rohit	M_3	e4
Roban	M01	e05
Roban	M01	e06

i) Select * from emp where ename like ('_j%') or ename like ('R%');

ii) Select * from emp where ename like ('_j%') and ename like ('R%');

iii) Select * from emp where ename like ('_j%') and ename like ('E%');

i) Ans

Ename	Manager	Empno
Ejan	M_1	e56
Rohit	M_3	e4
Roban	M01	e05
Roban	M01	e06

ii) => No rows selected.

iii) =>

Ename	Manager	Empno
Ejan	M_1	e56.

EID	Name	Department	start_time	end_time	gender
101	John	Sale	10:30	14:30	Male
201	von	Publication	08:30	16:30	Male
302	Neman	Sale	15:00	18:30	Female
405	Charles	Accocant	09:30	16:00	Male
305	Babbage	store	08:30	18:30	Female
int	char(40)	char(40)	time	time	char(10)

Write SQL queries for the following.

1. Create the relation name Employee.
2. List all records of employee who are working in sale department and working more than 6 hocers.
3. Append a record in employee relation where data record is 309, samir, Sale, 08:00, 15:30, null
4. Change the department of all females into accocant. /change/modify/update
5. Remove all records from database who are working in sale department

Note:

1, # datatype

time hh:mm:ss (nnnnnnn)

date YYYY_MM_DD

datetime YYYY_MM_DD hh:mm:ss(nnn]

2, DATEDIFF.

Function that handles larger differences between start date and enddate values // (start and end value)

Syntax: DATEDIFF
(datepart, startdate, enddate)

year -> yy, yyyy
month -> mm, m
day -> dd, d
week -> wk
hour -> hh
minute -> mi
second -> ss, s.

1) CREATE TABLE Employee (EID int, Name char(40), Department char(40), start_time time, end_time time, gender char(10));

2) Select * from Employee where
Department = 'sale'
AND
DATEDIFF (hh, start_time, end_time) > 6;

3) Insert into employee values (309, 'Samir', 'sal
'08:00', '15:30', 'Male')
(Time/Date should be in single quotation).

4) Update Employee
Set Department = 'Account'
where gender = 'Female'

5) Delete from employee where Department = 'Sale';

Table name : Order_list.

Order_no	Client_no	order_date
0191C	C101	2013-08-14
0191b	C102	Today's date
0131C	C 501	2013-03-01

NOTE:	SYSDATETIME()
	GETDATE()
	CURRENT_TIMESTAMP()

Write SQL queries :

(i) Create table as above.

(ii) Append data to the table.

i)=> Create table Order_list (Order_no varchar(20), Client_no varchar(20), Order_date date)

ii)=> Insert into Order_list values ('0191b', 'C102', GETDATE());

SET operations in SQL.

1. Union (A ∪ B)
2. Intersect (A ∩ B)
3. Minus (A - B)

1. Union :- Combines result of two or more select statement.
- Eliminate duplicate rows from its result set.
- No. of columns and datatype must be same

Table 1

ID	Name
1	cr7
2	MS10

Table 2

ID	name
2	MS10
3	MM10

SQL Query: Select * from Table 1
UNION
Select * from Table2;

Output:

ID	Name
1	cr7
2	MS10
3	MM10

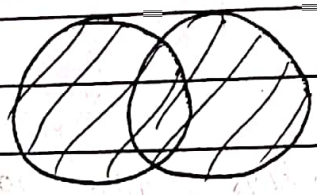


Table1 ∪ Table2.

2. Intersect
- Only returns the record that are common from both select statement.
- No. of ~~row~~ columns & datatype must be same.

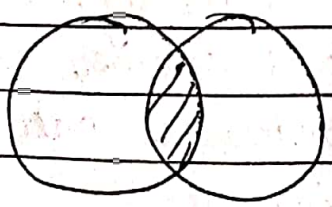


Table1 ∩ Table2.

Output:

ID	Name
2	MS10

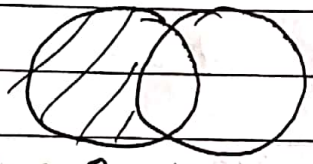
SQL Query: Select * from Table1
INTERSECT
Select * from Table2;

3) Minus
- Returns only those result which belongs to the first set of the result

SQL Query: Select * from Table1
MINUS
Select * from Table2;

Output:

ID	Name
1	Ram



A - B Table1 - Table2

SQL join com

VVImp
Joins in SQL.
- A SQL join combine rows from two or more tables based on a related column between them.
- It creates a set of rows in a temporary table.

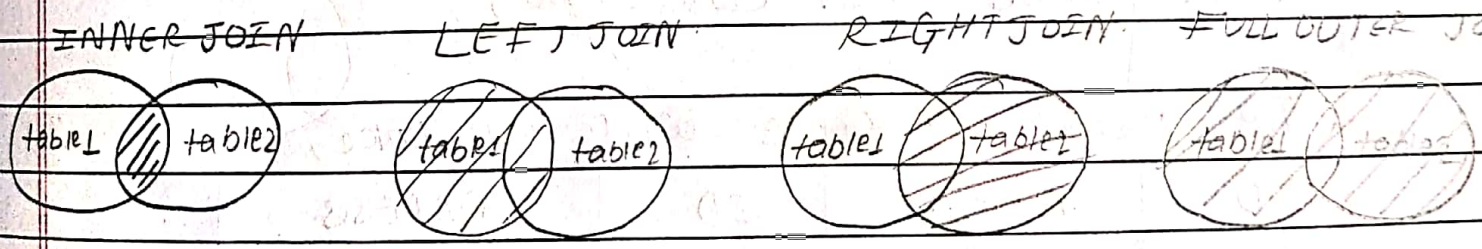
* Join Vs Subquery.

	Join	Subquery.
1	Joins are faster.	1. It is slower in most cases.
2.	Joins can do precalculation of what data is to be loaded and how much time it takes to process.	2. No preprocess calculation is needed or carried out

3. At least 2 tables are needed to process.	3. No needed of 2 tables.

* Types of Joins:

1. Cross Join: cartesian product.
2. (INNER) JOIN or EQUIJOIN: Returns records that have matching values in both tables.
   i) Natural join.
3. Outer Join
   i) LEFT (OUTER) JOIN: Returns all records from the left table, & the matched records from the right table.
   ii) RIGHT (OUTER) JOIN: Returns all records from the right table, and the matched records from th left table.
   iii) FULL (OUTER) JOIN: Returns all records when there is a match in either left or right table.



± Cross JOIN or Cartesian product.

- This type of JOIN returns the cartesian product of rows from the tables in Join. It will return a table which consists of records which combines each row from the first table with each row of the second table.

Syntax:

```
SELECT column_name_list
FROM
table_name1 CROSS JOIN table_name2;
```

Following is the class table;

ID	Name
1	abhi
2	adam
4	alex

& the class_info table

ID	ADDRESS
1	ktm
2	Lalitpur
3	Bhaktapur

Cross JOIN query will be,

```
SELECT * FROM
class CROSS JOIN class_info;
```

The result set table will look like;

ID	NAME	ID	ADDRESS
1	Abhi	1	ktm
2	Adam	1	ktm
4	Alex	1	ktm
1	Abhi	2	Lalitpur
2	Adam	2	Lalitpur
4	Alex	2	Lalitpur
1	Abhi	3	Bhaktapur
2	Adam	3	Bhaktapur
4	Alex	3	Bhaktapur

2. INNER Join or EQUI Join:

- This is a simple JOIN in which the result is based on matched data as per the equality condition specified in the SQL query.

Syntax:

```
SELECT column-name-list FROM
table-name INNER JOIN table_name2
ON table-name1.column_name = table-name2.
column_name;
```

Following is the 'class' table;

ID	NAME
1	abi
2	adam
3	alex
4	anu

and the class-info table

ID	ADDRESS
1	Ktm
2	Lalitpur
3	Bhaktapur

Inner Join query will be

```
SELECT * from class INNER JOIN class_info on class.id=
class_info.id;
```

The result set table will look like;

ID	NAME	ID	Address
1	Abhi	1	Ktm
2	Adam	2	Lalitpur
3	Alex	3	Bhaktapur

a) Natural Join (Not applicable in SQL server)
- Natural JOin is a type of Inner join which is based on column having same name and same datatype present in both tables to be joined.

Syntax :-

```
SELECT * FROM
table-name1 NATURAL JOIN table-name2;
```

Natural join query will be,

```
SELECT * from class NATURAL JOIN class-info;
```

The result will be;

ID	NAME	Address
1	Abhi	Ktm
2	Adam	Lalitpur
3	Alex	Bhaktapur

3, OUTER JOIN

- Outer join is based on both matched and unmatched data. Outer joins subdivide further into :

Table : class

id	name
1	abhi
2	adam
3	alex
4	ana
5	ashish.

Table : class_info

id	address
1	ktm
2	Lalitpur
3	Bhaktapur
7	BRT
8	BRJ.

i) Left Outer Join

- This join returns all the rows from left table combine with the matching rows of right table. If ya get no matching in right it returns NULL values. To specify a condition, we use the ON keyword with Outer join.

Syntax:

```
SELECT column_name_list FROM
table_name1 LEFT OUTER JOIN table_name2
ON table_name1.column_name = table_name2.column_
name;
```

Example:

```
SELECT * FROM class LEFT OUTER JOIN classinfo
ON class.id = class_info.id;
```

id	name	id	address
1	abhi	1	ktm
2	adam	2	Lalitpur
3	alex	3	Bhaktapur
4	anu	NULL	NULL
5	ashish	NULL	NULL

ii) Right Outer Join

- This join returns all the rows from right table are combined with the matching rows of left table. If you get no column matching in the left table, it returns null value.

Syntax:

```
SELECT * FROM
SELECT column_name_list FROM
table_name1 RIGHT OUTER JOIN table_name2
ON table_name1.column_name = table_name2.
columnname;
```

Example:

```
SELECT * FROM Class RIGHT OUTER JOIN
class_info ON (class.id = class_info.id);
```

Output:

id	name	id	address
1	abhi	1	ktm
2	adam	2	Lalitpur
3	alex	3	Bhaktapur
NULL	NULL	7	BRT
NULL	NULL	8	BRJ

iii) Full Outer Join:

The SQL Full join is the result of combination of both left & right outer join tables have all the records from both tables. It puts NULL on the place of matches not found.

Syntax:

```
SELECT column_name_list FROM table-name1
FULL OUTER JOIN table-name2 ON table1.
.column-name = table-name2.column-name;
```

Example:

```
SELECT * FROM class FULL OUTER JOIN
class-info ON (class.id = class-info.id);
```

Output:-

id	name	id	address
1	abhi	1	ktm
2	adam	2	Lalitpur
3	alex	3	Bhaktapur
4	anu	NULL	NULL
5	ashish	NULL	NULL
NULL	NULL	7	BRT
NULL	NULL	8	BRJ

Imp

Q.3. Consider the following three relations (7-8 marks)
Doctor (Name, age, address)
Works (Name, Depart_no, salary)
Department (Depart_no, departname, floor, room).

Write SQL statements for the following:
i) Display the name of doctor who do not work in any department.
ii) Modify the database so that Dr. Hari now lives in Pokhara.
iii) Delete (Display) all record of Doctor working in OPD department.
iv) Display the name of doctors who work in atleast two departments.

Answers:-

i)=> Select Doctor.Name from Doctor, Works where
Doctor.Name = Works.Name
AND
Works.Depart_no is NULL;

ii)=> Update Doctor
set address = 'Pokhara'
where Doctor.Name = 'Dr.Hari';

i)=> Select. Delete Doctor.Name, Doctor.age, Doctor.address
from Doctor, Works, Department
where Doctor.Name = Works.Name AND
Works.Depart_no = Department.Depart_no
AND Department.depname = 'OPD department';

iv) Select Works.Name from Works
Group by Works.Name
Having Count(Depart_no)>=2;