(8 hours)

2 Q / 1 SQ
(15-20 marks)

Imp / Chapter-4.

Relational Database Query languages

* **SQL (Structured Query language) :-**
It is the standard language for accessing and manipulating databases.

Features of SQL :-
1) High performance
2) Scalability and Flexibility
3) High Security
4) Management ease
5) Open source
6) Easier to integrate for application development

SQL Vs MySQL :-

SQL	MySQL
i) SQL is a language for accessing and manipulating database.	i) It is a software or an application.
ii) It enables the creation of data management system.	ii) It enables data handling, storing, modifying and deleting.
iii) SQL ~~force~~ codes and commands are used in different DBMS system like MySQL.	iii) MySQL has SQL at its core and is dependent on SQL for future updates.

Advantages of SQL:-

1) It provides high speed and well-defined standard.
2) It helps to provide security and provide high scalability.
3) Since, it is an open source, it is easier to learn and implement in the real world.

Disadvantages of SQL:-

1) With introduction of many versions, it is difficult in interfacing one another.
2) Many software are company dependent so one may have a feature that the other don't.

Various Syntaxes in SQL:-

SQL Select Statement

```
SELECT column 1, column 2 ..... column N
FROM table name;
```

SQL Distinct Clause

```
SELECT ~~Distinct~~ DISTINCT column 1, column 2, ... column N
FROM table-name;
```

SQL WHERE Clause

```
SELECT column 1, column 2, .... column N
FROM table-name;
WHERE condition;
```

SQL AND/OR Clause

```
SELECT column1, column2 ..... columnN
FROM table_name
Where condition-1 {AND/OR} condition-2;
```

SQL IN CLAUSE

```
SELECT col1, col2, ..... colN
FROM table_name,
WHERE column_name IN (value1, value2, ....);
```

SQL BETWEEN CLAUSE

```
SELECT col1, col2 , ....., colN
FROM table_name
WHERE column_name BETWEEN val-1 AND val-2;
```

SQL ORDER BY CLAUSE

```
SELECT col1, col2, ....., colN
FROM table_name
WHERE CONDITION
ORDER BY column_name {ASC/DESC};
```

(ascending/descending)

SQL GROUP BY CLAUSE

```
SELECT SUM (Column_name)
FROM table_name
WHERE condition.
GROUP BY column_name,
```

SQL LIKE CLAUSE

```
SELECT col1, col2, .... colN
FROM table-name
WHERE column_name LIKE {PATTERN};
```

SQL COUNT CLAUSE

```
SELECT count COUNT (Column-name)
FROM table_name
WHERE condition;
```

SQL HAVING CLAUSE

```
SELECT SUM (column-name)
FROM table-name
WHERE CONDITION
GROUP BY column-name
HAVING (arithmetic function condition);
```

CREATE TABLE STATEMENT

```
CREATE TABLE table-name (column1, datatype, column2
datatype, .... column N datatype, PRIMARY KEY
(column Name));
```

DROP Table statement.

```
Drop table table-name;
```

ALTER Table Statement.

ALTER TABLE table-name ADD/DROP

ALTER TABLE STATEMENT (RENAME).

Alter TABLE table-name, RENAME to new table-name;

INSERT INTO STATEMENT

INSERT INTO table-name (column 1, column 2, column N)
VALUES (value 1, value 2 .... value N);

V.imp

UPDATE STATEMENT

UPDATE table-name
SET column1, = value 1, column2 = val 2 .... column N = val
[WHERE CONDITION];

Example :-

Table Student

roll no.	name	faculty	SGPA
1	Abc	BCE	NULL

Q7. Add a value for SGPA of 3.0 to roll no. 1.

Ans>

Update Statement
Set SGPA = 3.0
where rollno = 1;

] - In single line also.

o/p:-

roll-no	name	faculty	SGPA
1	Abc	BCE	3.0

Q> Update the name of roll no. 1 to cdf.

Ans:⇒

```
Update Student
set name = 'cdj';
where roll no = 1;
```

(drop delete others)

DELETE STATEMENT

```
DELETE FROM table-name
Where {CONDITION} ;
```

E.g.

Q> Delete from Student where roll-no 1.

(Delete a particular row)

```
Delete Student
where roll no = 1;
```

CREATE DATABASE

```
CREATE DATABASE database-name;
```

DROP DATABASE

```
DROP DATABASE database-name;
```

USE DATABASE;

```
USE database-name;
```

Database: Customer Table-name: Customer_Info

Customer ID	Customer_username	Contact_Name	Address	City	Postal_code	State.
int, primary key	varchar [30]	char [40]	char [30]	char [40]	int	char [340]
1.	ronal doer_7	Samir Wagle	Neworad	ktm	014	Pradesh three
2.	mesi 10	Ramesh_dekhak	kainchaur	ktm	014	Pradesh three
3.	hazard_11	Sinchan_karki	Campus chowk	BRT	041	Pradesh one
4.	pogta 06	Anzeet_Bhandari	Birla chowk	BRT	051	Pradesh two
5.	moller_09	Abhishek_Sharma	Mandhir Chowk	NPL	071	Pradesh five

ALTER Table

- ALTER table statement is used to add, delete, or modify columns in the existing table.
- It is also used to drop various constraints on an existing table. For example: If we want to add phone number in above table Customer_Info, the query will be;

DELETE

It is used to delete the existing record in a table.
For e.g. delete from customer_Info with Customer_Id = 1

Alter Table Customer_Info ADD phone number (54329);

- In order to drop a column, we can write the following SQL Query.

Alter Table Customer_Info drop column Phonenumber.

Imp.

UPDATE

This SQL statement updates the value of the existing record. For e.g. if we want to add a new contact name and a new city for the customer_id 1, we can write the following SQL query.

```
Update Customer_Info
set Contact_name = 'Abc', city = 'hij'
    ~~city = 'kathmandu';~~
where customerID = 1;
```

Order By

The order by keyword is used to short the results set in ascending or descending order. The order By keyword shorts the record in ascending order by default. In order to short the record in descending order, we use the DESC keyword.

Eg. Select * from Customer_Info order By postal_code;

Select * from Customer_Info Order By Postal-code DESC;

Select Column

It is used to display the selected output from the table

Eg. Select * from Customer_Info;

Q7. To display only the customer name and postal code from the table Customer_Info.

⇒ Select Customer_username, postal_code from Customer_Inf

The select distinct statement is used to return only distinct (or different) values. It also remove duplicate values.
E.g. Select distinct state from Customer_Info;

Where Clause

The where clause is used to provide certain condition for displaying the output from the table.
E.g. Select * from Customer_Info where state='pradesh three

Some other operator used in where clause are,

= Equal
<> or != Not Equal
> Greater than
< Less than
>= Greater than or equal
<= less than or equal
Between: Between an inclusive range.
Like: Search for a pattern
IN: To specify multiple possible values for a column.

IN: E.g. Select * from Customer_Info where City IN ('ktm', 'NPL');

SQL Like Operator :-

% : The percent sign represent 0, 1 or multiple character.

_ : The underscore represent a single character.

Examples:

a% → It finds any value that starts with a.

% a → It finds any value that ends with a.

% or % → It finds any value that have or in any position.

_r% → It finds value that have r in the second position.

a%o → It finds value that starts with a and ends with o.

a__%__%__ → It starts with a and have at least 3 characters.

E.g. ← % and % % or %

Eg.

Select * from Customer_Info where city like 'k%';

Questions: (Write down SQL for the following questions).

1) Create a database named Customer.
2) Create a table name Customer_Info as specified abo
3) Insert all the data as specified above.
4) Add a new column phone number.
5) Modify the value of Messi 10 to Banepa as his new address. (update)
6) Remove the data with postal_code 071.
7) Display the data from the table whose city nam starts with B.
8) Display the distinct value of city from the table

1) Create database Customer;

2) Customer_Username var
Create table Customer_Info (CustomerID int not null, ^
char(40), Contact_name ~~varchar~~ char (30), Address char (30), City
char (40), Postal_code int, State char (30),
Primary key (CustomerID));

3) Insert into Customer_Info values (1, 'ronaldocr_7'
'Samir_Wagle', 'Newroad', 'ktm', 014, 'Pradesh three');
Insert into Customer_Info values (2, 'messi10', 'Ramesh_
Lekhak', 'Lainchaur', 'ktm', 014, 'Pradesh three');
Insert into Customer_Info values (3, 'hazard_11', 'Sinchan
karki', 'Campus Chowk', 'BRT', 041, 'Pradesh One');
Insert into Customer_Info values (4, 'pogba06', 'Anzeet_
Bhandari', 'Birla Chowk', 'BRT', 051, 'Pradesh two');
Insert into Customer_Info values (5, 'muller_09', 'Abhishe
sharma', 'Mandhir Chowk', 'NPL', 071, 'Pradesh five');

4) Alter table Customer_Info ADD phone_number bigint;

5) Update Customer_Info
set ~~City~~ Address = 'Banepa'
where Customer_username = 'messi10';

6) ~~de~~Delete from Customer_Info with postal_code = 071;

7) Select * from Customer_Info where city LIKE 'B%';

8) Select distinct city from Customer_Info;

tuples → rows
table → relation.

Q7. 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 Nuwakot
ii) Delete all tuples in the works relation for employees of XYz corporation.
iii) Increase salary of all employees of ABC company by 10%

i) Update Employee
set city = 'Nuwakot' // Set Employee.city = 'Nuwakot'.
where Empname = 'Amit'

i) Update Employee
set Employee. city = 'Nuwakot'
where Employee. Empname = 'Amit';

ii) Delete from works where
works. Empname = 'XYz Corporation';

iii) Update works
set works. salary = 1.1 * salary
where works. cmpname = 'ABC Company';

salary = salary + 10% of salary
= 1.1 * salary

Imp

NOTES:-

1. Delete: The delete statement is used to delete the values only.

2. Drop: The drop statement is used to delete both the values and structure of the database, or the table.

3. Update: The update statement is used to modify or change the data in the table.

4. Alter: The alter statement is used to change the schema or design of the table.

* Aggregate functions:-

Database : Products

Table name = Products_info

Product ID	Name	Supplier Id	Category ID	Unit	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	10	350

The aggregate function perform certain operation on a set of values and return a single value as a result. Some of the aggregate functions are;

1) Average: avg()
It is used to return the average value of a numeric column.

ii) Maximum: max()
It is used to return the maximum value from the numeric column.

iii) Minimum: min()
It is used to return the minimum value from the numeric column.

iv) Count: count()
It is used to return the number of rows that matches a specified criteria.

v) Sum: sum()
It returns the total sum of a numeric column.

E.g.
i. Select avg (Price) from Products_Info;
ii. Select max (Price) from Products_Info;
iii. Select min (Price) " " " ;
iv. Select count (Product ID) from " " ;
v. Select sum (Price) from Products_Info;

GROUP By:-
It is used to group the result of a select query based on one or more column. It is also used with SQL functions to group the result from one or more tables.
Syntax:
Select column_name, function (column_name) from table_name where

website → Template download
→ C++
→ PHP / Laravel / ASP.net (frameworks)
(basics)

condition group by column_name;

Table: 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

→ We want to find name and age of employees grouped by their salaries.

```
Select name, age
from Employee group by salary;
```

Output:-

name	age
hij	34
efg	29
abc	22

(based on unique salary and order)

In result, we will get a data set with unique salary listed along with the first employee name and age to have that salary.

E.g.

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

Output:-

group by - asending order

Name	Salary
hi	6000
egg	8000
kjm	9000

Q>. Make a table below using constraints,
Company (company_name, city, salary)
Constraints:
a) Company_name should be primary key
b) Company_name should not be null.
c) Salary should be greater than 5000.00
Write SQL for the table.

⇒ Create table Company (company_name varchar (30) not null, city char (30), salary decimal (10, 2), ~~Primary key (company_name)~~, check (salary >5000), Primary key (company_name));

Q. employee (emp-id, empname, street)
Constraints:
a) emp-id ~~salary~~ should be primary key.
b) street should be not null.
c) employee name should begin from 'a'.
write SQL for the table.

⇒ CREATE TABLE employee (emp-id ~~int varchar(20) not null~~ int, empname ~~char~~ char(30) check (empname like 'a%'), street char(30) not null, Primary key (emp-id));

HAVING CLAUSE:-

Q. Table name: Sales

OId	Order-name	previous balance	Customer
11	Ord 1	2000	abc
12	Ord 2	1000	cde
13	Ord 3	2000	fgh
14	Ord 4	1000	cde
15	Ord 5	2000	abc

→ It is used to provide more precise condition for a statemen
→ It is used to mention condition in Group By based on SQL queries just like where clause. ~~It is used~~
→ It is used instead of where clause when we cannot use aggregate function with where clause.

Syntax:-
SELECT column-name, function from table-name where

Output:

	Previous-balance
abc	2000 + 2000 = 4000
cde	1000 + 1000 = 2000
fgh	2000

HAVING

~~having~~

Column-name Condition Group By Column-name function (Column-name) condition.

Examples-

Select * from sales Group By Customer HAVING sum (previous_balance > 3000);

OUTPUTS-

Oid	Order-name	previous-balance	Customer
11	ord1	2000	abc.

abc sum is and not entry to display in output.

Q7. How does Group By clause work? What is the difference between where and HAVING clause? Explain each with example.

(L.O. No. 5) 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 like a subroutine or a sub-program in a regular computing language stored in database.
- In case, you have an SQL query that you write over and over again you can save it as a stored procedure and just call it to execute it.
- Parameters can also be passed in a stored procedure so that the stored ~~produce~~ procedure can act based on the parameter values that is passed.

SQL Server Basics:

Stored Procedure Syntax:

```
Create Procedure Procedure_name
As
    sql_statement
Go;
```

Execute a stored Procedure:

```
Exec procedure_name;
```

Examples-

```
Create procedure SelectAllCustomers
As
Select * from Customers
Go;

Exec SelectAllCustomers;
```

Advantages of Stored Procedure :-

1) It is easier to implement and understand.
2) It helps programmer to avoid mistakes.
3) It provides automatic back-up as the query is stored in database.
4) It supports passing of parameters, thus making it dynamic in nature.

Disadvantages of Stored Procedure :-

1) Stored procedure are vendor specific. Therefore, if you switch from one vendor to another vendor it requires

rewriting the code for the stored procedure.

2) When working with stored procedure, debugging can be hard to detect.

QBE (Query By Example) :-

QBE is a graphical language, where queries look like tables. QBE is the name of both a data-manipulation language and an early database system that included this language. Here, we consider only the data-manipulation language. It has distinct features :-

1) Unlike most query languages and programming languages, QBE has a two-dimensional syntax. Queries look like tables. A query in a one-dimensional language (for e.g. SQL) can be written in one (possibly long) line. A two dimensional language requires two dimensions for its expression.

2) QBE queries are expressed "by example". Instead of giving a procedure for obtaining the desired answer, the user gives an example of what is desired.

We express queries in QBE by skeleton tables. These tables show the relation schema. Rather than clutter the display with all skeletons, the user selects those skeletons ~~under~~ needed for a given query and fills in the skeletons with example rows. An example row consists of constants and example elements, which are domain variables. To avoid confusion between the two, QBE uses an underscore character (_) before domain variables, as in _x, and lets constants appear without any qualification.

branch	branch-name	branch-city	assets

customer	customer-name	customer-street	customer-city

loan	loan-number	branch-name	amount

borrower	Customer-name	loan-number

account	account-number	branch-name	balance

depositer	customer-name	account-number

Fig: QBE skeleton tables for the bank example

- QBE (unlike SQL) performs duplicate elimination automatically. To suppress duplicate elimination, we insert the command ALL.
- QBE allows queries that involve arithmetic comparisons. (E.g. >, <, =, ≥, ≤ -).
- QBE allows queries that span several different relations.

Q. 1) Write the output of the following query with reference to Emp relation.

ename	manager	emp no
Ejan	M1	e56
Rohit	M8	e4
Rohan	M01	e05
Rohan	M01	e06

i) Select * from Emp where ename like with '_j%' or ename like 'r%';

ii) Select * from Emp where ename like with '_j%' and ename like 'r%'; starting letter — second letter

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

iv) Select * from Emp where ename like '%i' or ename like '___a%';

⇒ i)

ename	manager	emp no
Ejan	M1	e56
Rohit	M3	e4
Rohan	M01	e05
Rohan	M01	e06.

ii) No rows selected.

iii)

ename	manager	emp no
Ejan	M1	e56

iv)

ename	manager	emp no.
Rohan	M01	e05
Rohan	M01	e06.

v) DELETE from Employee where department = 'sale';

Q.2) 2)

EID	Name	Department	start_time	end_time	Gender
101	John	Sale	10:30	14:30	male
201	Von	Publication	8:30	16:30	male
301	Neumann	Sale	15:00	18:30	Female
401	Charles	Account	09:30	16:00	male
501	Babbage	Sale	08:70	18:80	Female

Write SQL code for the following:-

i) Create Employee relation.

ii) List all the records of employee who are working in sale department and working more than 6hrs.

iii) Append a record in Employee where data record is 309, Samir, sale, 08:30, 15:30, male.

iv) Change the department of all females in account.

v) Remove all records from database who are working in sale department.

⇒ i) CREATE TABLE Employee (EID int not null, Name ~~varchar~~ char (30), Department ~~varchar~~ char (30), start_time time, end_time time, Gende ~~varchar~~ char (20), PRIMARY KEY (EID));

EID, Name, Department, start_time, end_time, gender.

ii) SELECT * from Employee where Department = Sale and (DATEDIFF (HH, start_time, end_time) > 6);

iii) Insert into Employee values (309, 'Samir', 'sale', '08:30', '15:30', 'male');

iv) Update Employee
set Department = 'Account'
Where Gender = 'Female';

Embedded SQL:-

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

Embedded SQL is a method of combining computing power of programming language and database manipulation capabilities of SQL. We need to access the database from a general purpose programming language because;

i) SQL is not powerful as general purpose programming language.

ii) There are different queries which cannot be expressed in SQL but can be programmed in other programming languages like C, C++, Python etc.

iii) Non-declarative action such as printing, interacting with the user or sending result to the GUI cannot be done with SQL.

SQL standard defines embedding of SQL with programming language as embedded SQL. The language in which

SQL query is embedded is known as host language.

Set operations in SQL:-

There are basically three types of set operation in SQL.

1) Union :- It combines the result of two or more select statement. It eliminates the duplicate row from the final result. The number of columns and datatype must be same for the different tables.

Table 1.

Id	Name
1	Cr7
2	ms10

Table 2

Id	Name
2	ms10
3	nm10

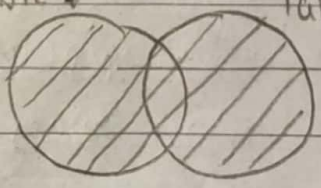
```
Select * from Table1
Union
Select * from Table2;
```

Output:-

Id	Name
1	cr7
2	ms10
3	nm10

2) Intersection :- It only return the records that are common from both select statement.

Table 1 Table 2.



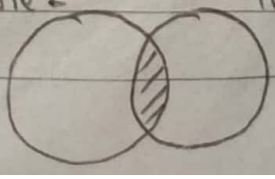
- The no. of columns and datatypes must be same.

```
Select * from Table1
INTERSECT
Select * from Table2;
```

Outputs-

Id	Name
2	ms10

Table 1 Table 2.

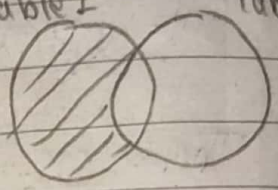


3) Minus :- It returns only those results which belong to the first set of the result.

Select * Table 1
Minus
Select * Table 2;

output:-

Id	Name
1	Cr 7



SQL Query:-
A SQL query is a request to access data from the database in order to manipulate it, or retrieve it.

SQL Sub-Query:-
- A sub-query (inner query or 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 statement along with the operators like =, <, >, <=, >=, BETWEEN etc.

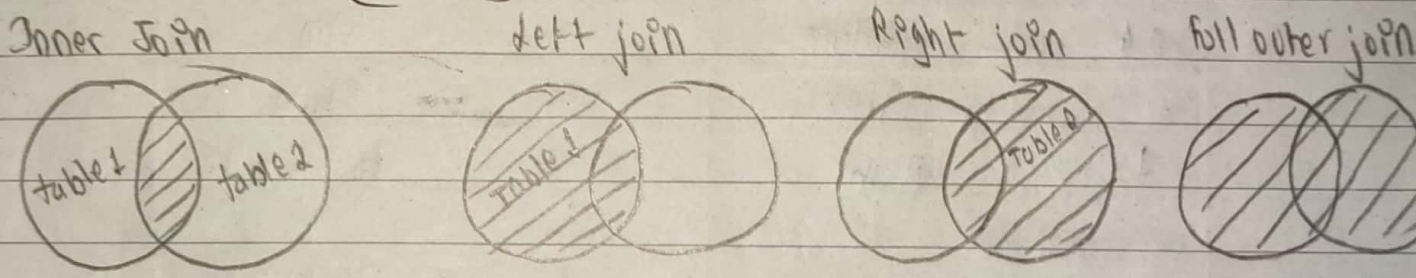
Eg.
Select * from customers
where ID IN (select Id from customers where salary > 4500) ;

V.V.I.
Joins in SQL:-
- A SQL join clause combines rows from two or more tables based on a related column between them.
- It creates a set of rows in a temporary table.

Q7. Difference between join and sub-query.

Types of joins-

1. Cross join : Cartesian product
2. (INNER) join JOIN or EQUI JOIN
   → NATURAL JOIN
3. OUTER JOIN
   → LEFT (OUTER) JOIN
   → RIGHT (OUTER) JOIN
   → FULL (OUTER) JOIN.



1) CROSS JOIN : Cartesian Product-

It returns the cartesian product of rows from the table in join. It will return a table which consist of record which combines each row from first table with each row from second table.

Syntax-

```
SELECT column-name_list
FROM
table-name 1 CROSS JOIN table-name 2;
```

Table name: Class

ID	Name
1	abhi
2	adam
4	alex

Table name: Class-Info

ID	Address
1	ktm
2	Lalitpur
3	Bhaktapur

Cross Join query will be

```
SELECT * FROM
class CROSS JOIN class-info;
```

→ Table 1 with each data of Table 2

Output:-

ID	Name	ID	Address
1	Abhi	1	ktm
2	Adam	1	ktm
4	Alex	1	ktm
1	Abhi	2	Lalitpur
⋮	⋮	⋮	⋮
1	Abhi	3	Bhaktapur
⋮	⋮	⋮	⋮

2) INNER JOIN:-

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

Syntax:-

```
SELECT column_name_list FROM
table_name1 INNER JOIN table_name2
ON table_name1.column_name = table_name2.column_nam
```

INNER JOIN query will be:-

SELECT * FROM
class INNER JOIN class_info
ON class.id = class_info.id ;

Output:-

Eg. Table name: class

ID	name
1	Abhi
2	adam
3	alex
4	abc

Table: Class_Info

ID	Address
1	ktm
2	Lalitpur
3	Bhaktapur

Output:-

ID	Name	ID	Address
1	Abhi	1	ktm
2	Adam	2	Lalitpur
3	Alex	3	Bhaktapur

Id-4 doesn't have any corresponding data in table 2.

⇒ NATURAL JOIN:-

It is a type of inner join which is based on column having same name and same datatype present in both the tables to be joined. The syntax for natural join is:-

SELECT * FROM table_name1
NATURAL JOIN table_name2 ;

Eg:- SELECT * from class
class NATURAL JOIN class_info;

So, the output will be:-

ID	Name	Address
1	Abhi	ktm
2	Adam	Lalitpur
3	Alex	Bhaktapur

3) OUTER JOINS-

Table : class

Pd	name
1	abhi
2	adam
3	alex
4	abc
5	ashish

Table: class_info

id	address
1	ktm
2	lalitpur
3	Bhaktapur
7	BRT
8	BRJ

Outer join is based on both matched and unmatched data. Outer join can be further divided into.

1. left outer join :-

This join returns all the rows from the left table combined with the matching row of the right table. If we get no matching in the right table, it returns the null values. To specify a condition, we use the 'ON' keyword with outer join.

Syntax:-

SELECT column_name FROM
table_name1 LEFT OUTER JOIN table_name2
ON table_name1.column_name = table_name2.column_name;

Example:-

Select * from class left Outer Join
class_info ON (class.id = class_info.id);

OUTPUT:-

Pd	name	id	address
1	abhi	1	ktm
2	adam	2	lalitpur
3	alex	3	Bhaktapur
4	abc	NULL	NULL
5.	ashish	NULL	NULL

2. Right outer join:-

This join returns all the rows from the right table which are combined with the matching row of the left table. If we get no column matching on the left table it will return null value.

Syntax:-

SELECT column_name FROM
table_name 1 RIGHT OUTER JOIN table_name 2
ON table_name 1 .column_name = table_name 2 .column_name;

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.

3. Full Outer Join:-

The SQL full join is the result of combination of both left and right outer join and join tables where have all the records from both tables. It puts null values on the places of matches not found.

Syntax:-

SELECT column_name FROM
table_name 1 FULL OUTER JOIN table_name 2
ON table_name 1 . column_name = table_name 2 . 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	~~ashishabc~~	NULL	NULL
5	ashish	NULL	NULL
NULL	NULL	7	BRT
NULL	NULL	8	BRJ

Questions:-

1) Point out the pros of joins. Explain different type of joins in SQL.
2) Write short notes on.
   i) Joins ii) Outer Joins iii) Importance of Joins.

Imp ~~QBE~~ (Query By Example):-

- It is another language for manipulating and retrieving data from the database.
- It provides a user friendly graphical interface of manipulating databases.
- Without the QBE, a user needs to write the input code for the specific purpose. If the code is not correct the overall operation may not run.

- QBE provides a simple interface for the database developer or the user to enter their queries.

Examples-

Instead of writing the whole code of the SQL command, the user can fill in the blanks or select certain pre-defined codes they want to perform. for ~~exa~~

A user may want to retrieve the data from the table student with roll no 123.

Using SQL:-

```
Select * from Student
where roll_no = 123;
```

Using QBE:-

The user may be allow to just click on student table type 123 in the roll_no field and click search.

The purpose of QBE is to make the database query easier to use and to avoid mistakes while writing SQL quer

2017 Fall.

2. b) Write the SQL statements for the following queries by reference of liquors - Pizza relation &

Serial No.	liquors	Start year	Bottles	Ready year
1	Gorkha	1997	10	1998
2	Divine wine	1998	5	2000
3	Old Durbar	1997	12	20001
4	Khukuri Rum	1991	10	1992
5	Xing	1994	5	1995

i) Create the liquors Pizza relation.

(ii) Insert the records in liquors_Info as above.
(iii) List all the records which were ready by 2000.
(iv) Remove all records from data base that required more than 2 years to get ready.

i) ⇒ CREATE TABLE liquors_Info (SerialNo int not null, varchar(30), StartYear int, Bottles int, ReadyYear int, PRIMARY KEY (serialNo));

ii) ⇒ INSERT into liquors_Info values (1, 'Gorkha', 1997, 10, 1998)
INSERT into liquors_Info values (2, 'Divine Wine', 1998, 5, 2000)
INSERT into liquors_Info values (3, 'Old Durbar', 1997, 12, 2001)
INSERT into liquors_Info values (4, 'khukuri Rum', 1991, 10, 1992)
INSERT into liquors_Info values (5, 'Xpng', 1994, 5, 1995);

iii) ⇒ SELECT * from liquors_Info where ReadyYear <= 2000;

iv) ⇒ DELETE from liquors_Info where (ReadyYear − StartYear) > 2;
(minus)

V.V.I.

Q> Consider the following three relations:-
i) Doctor (Name, Age, Address)
ii) Works (Name, Depart_no, Salary)
iii) Department (Depart_no, Depart_name, Floor, Room)
Write down SQL for the following scenario.
i) Display the name of doctor who do not work in any department.
ii) Modify the database so that doctor Dr. Hari now lives in Pokhara.
iii) Delete all record of doctor working in OPD department.
iv) Display the name of doctor who work in atleast two department.

i) SELECT ~~Name from Doctor~~ ~~FROM Works~~ ~~with~~ ~~where Department_no = NULL~~;

Select Doctor.Name from Doctor, Works where Doctor.Name = Works.N AND Depart_no Is null;

combination of tables.

= value.
(= null)

ii) Update Doctor
set ~~Doctor~~ Address = 'Pokhara'
where Doctor.Name = ~~'Hari'~~ 'Dr. Hari';

iii) DELETE from Department
where Department.Depart_name = 'OPD';

iv) SELECT ~~Name from Doctor~~ from Works.
where (Depart_no >= 2);

v) Delete works.Name from Works, Department
where works.Depart_no = Department.Depart_no
AND depname = 'OPD';

(where & function name can't be used at a time)

iv) Select Name from homes
Group by Name
Having count (Depart_no) >= 2;

2016-Fall:

2.b) Q>. Consider a relational schema Teacher (Teacher ID, Teacher Name, Office). Write SQL for the following task.

i) To create a table as mentioned above.

ii) CREATE TABLE Teacher (Teacher_ID Int not null, Teacher Name varchar (30), office varchar (30), PRIMARY KEY (Teacher_ID));

iii) To eliminate duplicate roles.
Select distinct (Teacher ID) from Teacher;

iv) To add a new column 'Ggender' in the table.
ALTER TABLE Teacher ADD Gender varchar (10)

v) To sort data in a table.
Select * from Teacher
Order By Teacher ID;

vi) To delete rows from the table.
Delete from Teacher
OR,
Delete * from Teacher.

vii) Count No. of ~~offices~~ rows based on office.
Select count (office) from Teacher;