

# LAB MANUAL

CS2231: Database Management System

LAB 08

**Instructor**

**Safiyah Batool**

LAB 08

**Schema**

Doctors (docid, docname, docdept, docfee)

Patients (patid, patname, patage, pathistory)

Visits (docid, patid, no\_of\_visits)

**Step1:** Create above tables with appropriate constraints.

**Step2:** Insert 10 records each in the doctors and patients tables and 20 records in the visits table.

**Step3:** Display all records inserted in the three tables.

SELECT \* FROM doctors;

SELECT \* FROM patients;

SELECT \* FROM visits;

**MODIFY THE TABLE**

**Step4:** Add a column ‘experience’ to doctors table.

ALTER TABLE doctors ADD (experience INT NOT NULL);

// It will automatically assign 0 to all columns.

// Use DEFAULT as follows to assign a default value of your choice:

ALTER TABLE doctors ADD (experience INT DEFAULT ‘5’ NOT NULL);

**Step5:** Drop Column experience from table doctors.

ALTER TABLE doctors DROP COLUMN experience;

**Step6:** [Adding a Constraint] Modify the patients table so that the patname is constrained to be UNIQUE.

ALTER TABLE patients ADD (CONSTRAINT patients\_name\_un UNIQUE (patname);

**Step7:** Drop constraint applied in step 6.

ALTER TABLE doctors DROP CONSTARINT patients\_name\_un;

**Step8:** Change Data type of docfee from number to varchar.

ALTER TABLE doctors MODIFY COLUMN docfee varchar(10);

**NOTE: Change the datatype of docfee again to number**

**MODIFY THE ROWS/DATA**

**Step9:** Change the docfee of a doctor to some other value.

UPDATE doctors SET docfee = 5000 WHERE docid = ‘1’;

**Step10:** Change values for multiple columns.

UPDATE doctors SET docfee = 5000, docname = ‘Amjad’ WHERE docid = ‘1’;

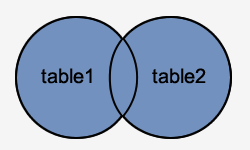
**SQL JOIN**

SQL JOINS are used to retrieve data from multiple tables. Most join queries contain WHERE clause conditions that compare two columns, each from a different table. Such a condition is called a **join condition.**

**Cartesian Join/ Cross Join**

It returns the Cartesian product of the tables involved in the join. All rows under the specified columns are returned. Simply, in Cartesian join there is no join condition.

Visualization



Syntax

SELECT patients.patid, doctors.docid

FROM patients, doctors;

**// It can be written by using aliases as follows**

SELECT p.patid, d.docid

FROM patients p, doctors d;

**// It can be written without using the table associations**

SELECT patid, docid

FROM patients, doctors;

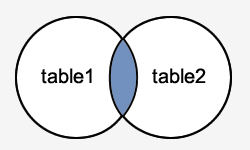
**//All rows of all columns can be retrieved as follows**

SELECT \* FROM patients, doctors;

**Equijoin**

An equijoin is a join with a join condition containing an equality operator ( = ). An equijoin combines rows that have equivalent values for the specified columns.

Visualization



Syntax

SELECT p.patid , d.docid, p.patname, d.docname

FROM patients p , doctors d

WHERE p.patname = d.docname;

**Non Equijoin**

Non Equijoins is used to return result from two or more tables where exact join is not possible.

Syntax

SELECT p.patid, p.patname, p.patage, v.docid, v.no\_of\_visits

FROM patients p, visits v

WHERE v.no\_of\_visits BETWEEM 5 AND 10;

**LAB ACTIVITY:**

Write Queries to make the required changes to tables/columns. If a change cannot be made due to any constraint, change the constraint to make the required change possible.

1. Add town and city attributes to the patient’s column. [Insert 5 rows]
2. Update the previously inserted rows with the values of town and city.
3. Delete no\_of\_visits column from the visits table.
4. Add date\_of\_visit column to the visits table with data type as varchar2. [Insert 5 rows]
5. Update the previously inserted rows with the value of date\_of\_visit.
6. Add experience attribute to the doctors table. [Insert 5 rows]
7. Update the previously inserted rows with the value of experience.
8. Change the data type of the date\_of\_visit column to date.
9. Update the date\_of\_visit column for previously inserted rows.