SE IT	Roll number :
Experiment no. : 6	Date of Implementation :
Aim : To implement constraints	
Tool Used : PostgreSQL	
Related Course outcome : At the end of the course, Student should be able to: 3. Create and populate a RDBMS, using SQL. 4. Write queries in SQL to retrieve any type of information from a data base.	

Rubrics for assessment of Experiment:

Indicator	Poor	Average	Good
Timeliness Maintains Experiment deadline (3)	Experiment not done (0)	One or More than One week late (1-2)	Maintains deadline (3)
Completeness and neatness Complete all parts of Experiment(3)	N/A	< 80% complete (1-2)	100% complete (3)
Originality Extent of plagiarism(2)	Copied it from someone else(0)	At least try to implement but could not succeed (1)	Implemented (2)
Knowledge In depth knowledge of the Experiment(2)	Unable to answer any questions(0)	Unable to answer few questions (1)	Able to answer all questions (2)

Assessment Marks :

Timeliness	
Completeness and neatness	
Originality	
Knowledge	
Total	

Total : (Out of 10)

Teacher's Sign :

EXPERIMENT 6	Constraints
Aim	To implement Integrity Constraints
Tools	PostgreSQL
Theory	Constraints are the rules enforced on data columns on table. These are used to prevent invalid data from being entered into the database. This ensures the accuracy and reliability of the data in the database. Constraints could be column level or table level. Column level constraints are applied only to one column where as table level constraints are applied to the whole table. Defining a data type for a column is a constraint in itself. For example, a column of type DATE constrains the column to valid dates. Following are commonly used constraints available in PostgreSQL. • **NOT NULL Constraint**: Ensures that a column cannot have NULL value. • **UNIQUE Constraint**: Ensures that all values in a column are different. • **PRIMARY Key**: Uniquely identifies each row/record in a database table. • **FOREIGN Key**: Constrains data based on columns in other tables. • **CHECK Constraint**: The CHECK constraint ensures that all values in a column satisfy certain conditions. **NOT NULL Constraint** By default, a column can hold NULL values. If you do not want a column to have a NULL value, then you need to define such constraint on this column specifying that NULL is now not allowed for that column. A NOT NULL constraint is always written as a column constraint. A NULL is not the same as no data, rather, it represents unknown data. **UNIQUE Constraint** The UNIQUE Constraint prevents two records from having identical values in a particular column. In the COMPANY table, for example, you might want to prevent two or more people from having identical age. **PRIMARY KEY Constraint** The PRIMARY KEY constraint uniquely identifies each record in a database table. There can be more UNIQUE columns, but only one primary key in a table. Primary keys are important when designing the database tables. Primary keys are unique ids. We use them to refer to table rows. Primary keys become foreign keys in other tables, when creating relations among tables. Due to a 'longstanding coding oversight', primary keys can be NULL in SQLite. This is not the case with other databases A primary key is a field in a table which uniquely identifies each row/record in a database table. Primary keys must contain unique values. A primary key column cannot have NULL values. A table can have only one primary key, which may consist of single or multiple fields. When multiple fields are used as a primary key, they are called a **composite key**. If a table has a primary key defined on any field(s), then you can not have two records having the same value of that field(s).

	FOREIGN KEY Constraint A foreign key constraint specifies that the values in a column (or a group of columns) must match the values appearing in some row of another table. We say this maintains the referential integrity between two related tables. They are called foreign keys because the constraints are foreign; that is, outside the table. Foreign keys are sometimes called a referencing key. **CHECK Constraint** The CHECK Constraint enables a condition to check the value being entered into a record. If the condition evaluates to false, the record violates the constraint and isn't entered into the table.
Procedure	1. Create following table company1(id int, name text not null, age int not null, address varchar(50), salary real); 2. Insert rows with name or age as NULL values 3. Create following table company3(id not null, name text not null, age int unique address varchar(50), salary real); 4. Insert 2 rows with same age values in two rows 5. Create following table company4(id int primary key, name text, age int, address varchar(50), salary real); 6. Insert 2 rows with same id values in two rows 7. Create following tables employee(id int primary key, name text, age int, address varchar(50), salary real); department(id int primary key, name varchar(50) not null, emp_id int references employee(id)); 8. Insert 2 rows in employee table with id 1 and 2 9. Insert row in department table with different empid 10. Create following table company5(id int, name text, age int, address varchar(50), salary real check(salary > 0)); 11. Insert row in with salary value less than 0
Post Lab Questions:	1. Explain Assertions 2. Explain Security and authorization in SQL