**EXPERIMENT 10:**

**-Team 16**

**AIM:** Study different types of join operations, Exist, Any, All, and relevant features of SQL.

**INTRODUCTION:**

* The **INNER JOIN** keyword selects records that have matching values in both tables.
* The **LEFT JOIN** keyword returns all records from the left table (table1), and the matched records from the right table (table2). The result is NULL from the right side, if there is no match.
* The **FULL OUTER JOIN** keyword returns all records when there is a match in left (table1) or right (table2) table records. FULL OUTER JOIN can potentially return very large result-sets. **FULL OUTER JOIN and FULL JOIN are the same.**
* A **LEFT JOIN** performs a join starting with the first (left-most) table.Then, any matched records from the second table (right-most) will be included.**LEFT JOIN** and **LEFT OUTER JOIN** are the same.
* A **RIGHT JOIN** performs a join starting with the second (right-most) table and then any matching first (left-most) table records. **RIGHT JOIN** and **RIGHT OUTER JOIN** are the same.
* A **SELF JOIN** is a regular join, but the table is joined with itself.

|  |  |  |  |
| --- | --- | --- | --- |
|  | ISHAN | KHUSH | ALISTER |
| INNER | **SELECT** Item\_id, Item\_cost  **FROM** Shopping\_List  **INNER JOIN** Analysis\_Sdata  **ON** Shopping\_List.Item\_ID = Analysis.Sdata.Item.ID |  |  |
| LEFT | **Info about all the medicines with no sensor currently present.**  **SELECT** Item\_Name, Item\_Category,sensor.ID  **FROM** Medicines **A**  **LEFT JOIN** Analysis\_Mdata **B**  **ON** A.M\_ID = B.M\_ID  **WHERE** B.Sensor\_ID **IS NULL**  **ORDER BY** Item\_wcost |  |  |
| RIGHT | **Info about customers without any home brought vessel.**    **SELECT**  FirstName, LastName, Age **FROM** Band\_Tag **A** **RIGHT JOIN** Customer\_info **B**  **ON** A.Customer\_ID =B.Customer.ID  **WHERE** Vessel\_tagID **IS NULL** |  |  |
| FULL | **Give ids of those items in front that have no sensor and also the sensors that have don’t have an item\_id associated with them**  **SELECT** Front.Item\_ID, Sensors.Sensor\_ID  **FROM** Front **A**  **FULL OUTER JOIN** Sensors **B**  **ON** A.Sensor\_ID = B.Sensor\_ID  **WHERE** Sensor.ID **IS NULL OR** Item.ID **IS NULL** ; |  |  |
| SELF | **List sensorID and location spot of the senors that have the same location**  **SELECT** A.Sensor\_ID **AS** sensorid1, B.Sensor\_ID **AS** sensorid2, A.location\_spot  **FROM** Sensors A, Sensors B  **WHERE** A.Sensor\_ID **<>** B.Sensor\_ID  **AND** A.location\_spot = B.locatoin\_spot  **ORDER BY** A.location\_spot; |  |  |

**EXISTS:**

The EXISTS operator is used to test for the existence of any record in a subquery.

The EXISTS operator returns true if the subquery returns one or more records.

|  |  |  |  |
| --- | --- | --- | --- |
|  | ISHAN | KHUSH | ALISTER |
| EXISTS | **SELECT** Item\_ID  **FROM** Front  **WHERE EXISTS** (SELECT Item\_category **FROM** Readers **WHERE** Front.location\_spot = Readers.location\_spot **AND** item\_quantity < 10); |  |  |
| ANY | **SELECT** Email\_ID,Customer\_ID,Address  **FROM** Customer  **WHERE** Band\_Tag.location = **ANY** (**SELECT** location\_seq **FROM** Band\_Tag **WHERE** location\_seq **LIKE** ‘A[#]%C[#]’); |  |  |
| ALL | **SELECT** Sensor\_ID, location\_spot  **FROM** Sensors  **WHERE** Sensor\_ID = **ALL** (**SELECT** Sensor\_ID **FROM** Sensors **WHERE** sensor\_status = 0); |  |  |