**Inner join**

The INNER JOIN keyword selects all rows from both tables as long as there is a match between the columns in both tables.

**Syntax:**

SELECT column\_name(s)  
FROM table1  
INNER JOIN table2  
ON table1.column\_name*=*table2.column\_name;

**Example:**

SELECT Customers.CustomerName, Orders.OrderID  
FROM Customers  
INNER JOIN Orders  
ON Customers.CustomerID=Orders.CustomerID  
ORDER BY Customers.CustomerName;

**Left outer join**

A Left outer join is one of the [JOIN operations](http://docs.oracle.com/javadb/10.8.3.0/ref/rrefsqlj29840.html#rrefsqlj29840) that allow you to specify a join clause. It preserves the unmatched rows from the first (left) table, joining them with a null row in the shape of the second (right) table.

**Syntax:**

[TableExpression](http://docs.oracle.com/javadb/10.8.3.0/ref/rreftableexpression.html#rreftableexpression) LEFT [ OUTER ] JOIN [TableExpression](http://docs.oracle.com/javadb/10.8.3.0/ref/rreftableexpression.html#rreftableexpression)

{

ON booleanExpression |

[USING clause](http://docs.oracle.com/javadb/10.8.3.0/ref/rrefsqljusing.html#rrefsqljusing)

}

**Example:**

SELECT DEPTNO, DEPTNAME, EMPNO, LASTNAME

FROM DEPARTMENT LEFT OUTER JOIN EMPLOYEE

ON MGRNO = EMPNO

**Right outer join**

A RIGHT OUTER JOIN is one of the [JOIN operations](https://db.apache.org/derby/docs/10.2/ref/rrefsqlj29840.html#rrefsqlj29840) that allow you to specify a JOIN clause. It preserves the unmatched rows from the second (right) table, joining them with a NULL in the shape of the first (left) table. A LEFT OUTER JOIN B is equivalent to B RIGHT OUTER JOIN A, with the columns in a different order.

**Syntax:**

[TableExpression](https://db.apache.org/derby/docs/10.2/ref/rreftableexpression.html#rreftableexpression) **RIGHT [ OUTER ] JOIN** [TableExpression](https://db.apache.org/derby/docs/10.2/ref/rreftableexpression.html#rreftableexpression)

**{**

**ON** booleanExpression

**}**

**Example:**

SELECT E.EMPNO, E.LASTNAME, M.EMPNO, M.LASTNAME

FROM EMPLOYEE E RIGHT OUTER JOIN

DEPARTMENT RIGHT OUTER JOIN EMPLOYEE M

ON MGRNO = M.EMPNO

ON E.WORKDEPT = DEPTNO

**Group by**

The GROUP BY statement is used in conjunction with the aggregate functions to group the result-set by one or more columns.

**Syntax:**

SELECT column\_name, aggregate\_function(column\_name)  
FROM table\_name  
WHERE column\_name operator value  
GROUP BY column\_name;

**Example:**

SELECT Shippers.ShipperName,COUNT(Orders.OrderID) AS NumberOfOrders FROM Orders  
LEFT JOIN Shippers  
ON Orders.ShipperID=Shippers.ShipperID  
GROUP BY ShipperName;

**Having**

The HAVING clause was added to SQL because the WHERE keyword could not be used with aggregate functions.

**Syntax:**

SELECT column\_name, aggregate\_function(column\_name)  
FROM table name  
WHERE column\_name operator value  
GROUP BY column\_name  
HAVING aggregate\_function(column\_name) operator value;

**Example:**

SELECT Employees.LastName, COUNT(Orders.OrderID) AS NumberOfOrders FROM (Orders  
INNER JOIN Employees  
ON Orders.EmployeeID=Employees.EmployeeID)  
GROUP BY LastName  
HAVING COUNT(Orders.OrderID) > 10;

**Avg**

The AVG() function returns the average value of a numeric column.

**Syntax:**

SELECT AVG(column\_name) FROM table\_name

**Example:**

SELECT ProductName, Price FROM Products  
WHERE Price>(SELECT AVG(Price) FROM Products);

**Finding top rows**

**Syntax:**

Select Top (top\_value) [ percent ] [ with ties]

Expressions

From Tables

[WHERE conditions]

[ ORDER by expression [ ASC | DESC ]];

**Example:**

employee\_id, last\_name, first\_name

FROM employees

WHERE last\_name = 'name'

ORDER BY employee\_id;

**Primay key:**

The PRIMARY KEY constraint uniquely identifies each record in a database table. Primary keys must contain UNIQUE values. A primary key column cannot contain NULL values.

**Example:**

CREATE TABLE Persons  
(  
P\_Id int NOT NULL,  
LastName varchar(255) NOT NULL,  
FirstName varchar(255),  
Address varchar(255),  
City varchar(255),  
PRIMARY KEY (P\_Id)  
)

**Forgein key**

A FOREIGN KEY in one table points to a PRIMARY KEY in another table.

**Example:**

CREATE TABLE Orders  
(  
O\_Id int NOT NULL,  
OrderNo int NOT NULL,  
P\_Id int,  
PRIMARY KEY (O\_Id),  
FOREIGN KEY (P\_Id) REFERENCES Persons(P\_Id)  
)

**Use cases:**

**Tables for employee management system**

|  |  |
| --- | --- |
| Use Case Selection | Description |
| Use case Name | Employment Details |
| Level | Sub-Functional Level |
| Primary Actor | Employment Details |
| Stakeholders and Interest | Employment:  Hire new Employee.  Providing ID Details  Payroll  Various departments |
| Pre-Condition | Employee has submitted registration forms. |
| Post-Condition | Records for employee has been added. |
| Main Success Scenario | 1.They are assigned for a position.  2.Tasks has been issued |
| Alternative Flows | 1.Company benefits |
| Special Requirements | Given task has been submitted in time to the superiors. |
| Technology and Data Variation | Different works like onsite or off site |
| Frequency of Occurrence | Could be nearly continuous. |
| Open Issues | If the performance is not accurate. |

**Tables for Library management system**

|  |  |
| --- | --- |
| Use Case Selection | Description |
| Use case Name | Add Records |
| Level | Sub-Functional Level |
| Primary Actor | Student, Faculty Member |
| Stakeholders and Interest | Student:  Wants to register in the library  Faculty Member:  Wants to register in the library  Administrator:  Responsible for the management of online activities and modifications in the database.  Librarian:  Responsible to keep the specific records for  Particular students and faculty member. |
| Pre-Condition | Student and faculty members have submitted their registration forms. |
| Post-Condition | Record for a student/faculty member has been added. |
| Main Success Scenario | 1.student/faculty member visits the online or offline library services.  2. student/faculty member sign-up to get registered online.  3. He/she provides correct information and a secret password.  4. she/he got registered. |
| Alternative Flows | 1.Student / Faculty member visits the website to access the online library services.  2. he/she tries to sign-up.  3. she/he fails and receives an error.  4. she/he will visit the librarian to report a complaint.  5. Librarian will forward this complaint to the administrator. |
| Special Requirements | 1.The maximum response time for online registration is 1 minute  2.The system can support various languages. |
| Technology and Data Variation | Student or faculty member may not be able to navigate on site. |
| Frequency of Occurrence | Could be nearly continuous. |
| Open Issues | If student/Faculty member is not registered online they have to wait for one day process to complain to the admin. |