ASSIGNMENT-4

Data-Base Concepts:

1. **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;**

1. **Left Outer Join:** A LEFT OUTER JOIN is one of the join operations 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)

}

1. **Right Outer Join: A** RIGHT OUTER JOIN is one of the join operations 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*

}

1. **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;

1. **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;

1. **Average:** The AVG() function returns the average value of a numeric column.

**Syntax:** SELECT AVG(column\_name) FROM table\_name

1. **Finding Top Rows:** The SELECT TOP clause is used to specify the number of records to return. The SELECT TOP clause can be very useful on large tables with thousands of records. Returning a large number of records can impact on performance.

**Syntax:**

SELECT TOP number|percent column\_name(s)  
FROM table\_name;

1. **Primary 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. Most tables should have a primary key, and each table can have only ONE primary key.

**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)  
)

1. **Foreign Key:** A FOREIGN KEY in one table points to a PRIMARY KEY in another table. Let's illustrate the foreign key with an example.

**Example:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **P\_Id** | **LastName** | | **FirstName** | | **Address** | | **City** |
| 1 | Kutala | | karthik | | 2925 Keystone | | texas |
| 2 | sama | | vidya | | 22250 Keystone | | texas |
| 3 | sam | | Kalyan | | 2300 Presscott dr | | texas |
|  | |  | |  | |

**Use-Cases:**

1. **Tables for employment management system:**

**Library management system:**

**Database Name: bookdetails**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Size** | **Relation** |
| Accno | Varchar | 50 | Primary key |
| Author | Varchar | 50 | Not null |
| Title | Varchar | 50 | Not null |
| Publication | Varchar | 50 | Not null |
| Edition | Varchar | 50 | Not null |
| No\_of\_copies | int |  | Not null |
| Volumn | Varchar | 50 | Not null |
| Date\_pur | Varchar | 50 | Not null |
| Price | Decimal | (18,2) | Not null |
| Status | Varchar(50) | 50 | Not null |

**Database Name: student**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** |  | **Data Type** | **Size** | **Relation** |  |
| Name |  | Varchar | 50 | Not null |  |
| Regno |  | Varchar | 50 | Primary key |  |
| Date\_of\_issue |  | Datetime |  | Not null |  |
| Addresss |  | Varchar | 50 | Not null |  |
| Date\_of\_return |  | Datetime |  | Not null |  |
| Course |  | Varchar | 50 | Not null |  |
| Accno |  | Varchar | 50 | Foreign key |  |
| Gender |  | Varchar | 50 |  | Not null |

**Database Name: issue**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Size** | **Relation** |
| Regno | Varchar | 50 | Foreign key |
| Date\_of\_issue\_books | Datetime |  | Not null |
| Date\_of\_return\_books | Datetime |  | Not null |
| Accno | Varchar | 50 | Not Null |
| Name | Varchar | 50 | Not null |
| Course | Varchar | 50 | Not null |
| Author | Varchar | 50 | Not null |
| Volumn | Varchar | 50 | Not null |
| Edition | Varchar | 50 | Not null |

**Database Name: return**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Size** | **Relation** |
| Regno | Varchar | 50 | Foreign key |
| Accno | Varchar | 50 | Not Null |
| Date\_of\_return\_books | Datetime |  | Not null |
| Date\_of\_issue\_books | Datetime |  | Not null |
| Name | Varchar | 50 | Not null |
| Course | Varchar | 50 | Not null |
| Author | Varchar | 50 | Not null |
| Volumn | Varchar | 50 | Not null |
| Edition | Varchar | 50 | Not null |