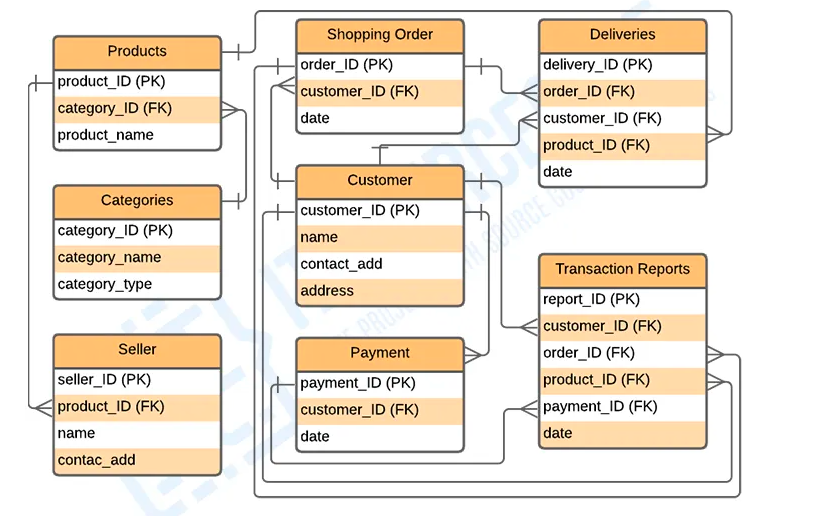
**SQL Assignment 2**

1. For an online purchasing database, create entity relationship diagrams. Create a database object from your entity diagram.

Answer :

### Online Shopping System ER Diagram Tables



### Table Name: Customer

| Field | Description | Type | Length |
| --- | --- | --- | --- |
| customer\_ID(PK) | Customer ID | int | 15 |
| name | name of customer | varchar | 50 |
| contact\_add | customer contact | int | 11 |
| address | address of customer | varchar | 250 |

### Table Name: Categories

| Field | Description | Type | Length |
| --- | --- | --- | --- |
| category\_ID(PK) | category Id | int | 11 |
| category\_name | category name | varchar | 255 |
| category\_type | category type | varchar | 255 |

### Table Name: Shopping Order

| Field | Description | Type | Length |
| --- | --- | --- | --- |
| order\_ID(PK) | order Id | int | 11 |
| customer\_ID(FK) | customer name | int | 11 |
| date | date of order | Date |  |

### Table Name: Deliveries

| Field | Description | Type | Length |
| --- | --- | --- | --- |
| acc\_ID(PK) | Account Id | int | 11 |
| customer\_ID(FK) | customer id | int | 11 |
| date | date of delivery | Date |  |

### Table Name: Products

| Field | Description | Type | Length |
| --- | --- | --- | --- |
| product\_ID(PK) | Product Id | int | 11 |
| category\_ID(FK) | category id | int | 11 |
| product\_name | product name | varchar | 255 |

### Table Name: Seller

| Field | Description | Type | Length |
| --- | --- | --- | --- |
| seller\_ID(PK) | seller Id | int | 11 |
| product\_ID(FK) | product ID | int | 11 |
| seller\_name | seller name | varchar | 255 |

### Table Name: Payment

| Field | Description | Type | Length |
| --- | --- | --- | --- |
| payment\_ID(PK) | payment Id | int | 11 |
| customer\_ID(FK) | customer ID | int | 11 |
| date | Date of Payment | Date |  |

### Table Name: Transaction Report

| Field | Description | Type | Length |
| --- | --- | --- | --- |
| report\_ID(PK) | report Id | int | 11 |
| customer\_ID (FK) | customer id | int | 11 |
| order\_ID(FK) | order id | int | 11 |
| product\_ID(FK) | product id | int | 11 |
| payment\_ID(FK) | payment id | int | 11 |

1. List the SQL aggregate function and demonstrate how to utilize it.

Answer :

# SQL Aggregate Functions

* SQL aggregation function is used to perform the calculations on multiple rows of a single column of a table. It returns a single value.
* It is also used to summarize the data.

## Types of SQL Aggregation Function

### 1. COUNT FUNCTION

* COUNT function is used to Count the number of rows in a database table. It can work on both numeric and non-numeric data types.

COUNT()

### 2. SUM Function

Sum function is used to calculate the sum of all selected columns. It works on numeric fields only.

SUM()

### 3. AVG function

The AVG function is used to calculate the average value of the numeric type. AVG function returns the average of all non-Null values.

1. AVG()

### 4. MAX Function

MAX function is used to find the maximum value of a certain column. This function determines the largest value of all selected values of a column.

MAX()

### 5. MIN Function

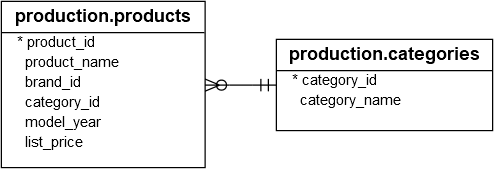
MIN function is used to find the minimum value of a certain column. This function determines the smallest value of all selected values of a column.

MIN()

1. In SQL, create a pivot query.

Answer :

Consider following table for Pivot table .



SQL Server PIVOT operator rotates a table-valued expression. It turns the unique values in one column into multiple columns in the output and performs aggregations on any remaining column values.

You follow these steps to make a query a pivot table:

* First, select a base dataset for pivoting.
* Second, create a temporary result by using a derived table or [common table expression](https://www.sqlservertutorial.net/sql-server-basics/sql-server-cte/) (CTE)
* Third, apply the PIVOT operator.

Let’s apply these steps in the following example.

First, select category name and product id from the production.products and production.categories tables as the base data for pivoting:

SELECT

category\_name,

product\_id

FROM

production.products p

INNER JOIN production.categories c

ON c.category\_id = p.category\_id

Second, create a temporary result set using a derived table:

SELECT \* FROM (

SELECT

category\_name,

product\_id

FROM

production.products p

INNER JOIN production.categories c

ON c.category\_id = p.category\_id

)

Third, apply the PIVOT operator:

SELECT \* FROM

(

SELECT

category\_name,

product\_id

FROM

production.products p

INNER JOIN production.categories c

ON c.category\_id = p.category\_id

) t

PIVOT(

COUNT(product\_id)

FOR category\_name IN (

[Children Bicycles],

[Comfort Bicycles],

[Cruisers Bicycles],

[Cyclocross Bicycles],

[Electric Bikes],

[Mountain Bikes],

[Road Bikes])

) AS pivot\_table;

This query generates the following output:



1. With an example, describe how to join in SQL.

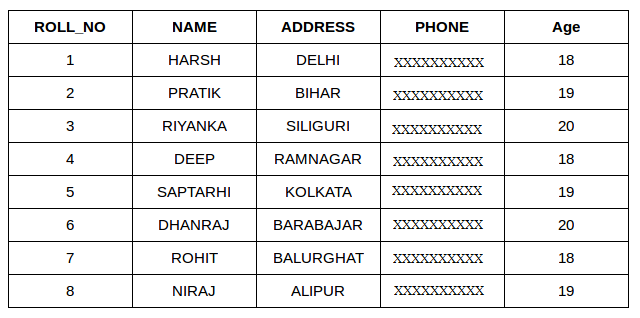
Answer :

A SQL Join statement is used to combine data or rows from two or more tables based on a common field between them. Different types of Joins are:

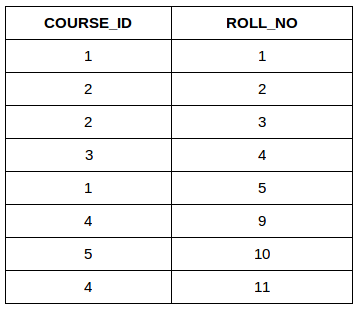
* INNER JOIN
* LEFT JOIN
* RIGHT JOIN
* FULL JOIN

Consider the two tables below:

**Student**

****

**StudentCourse**

****

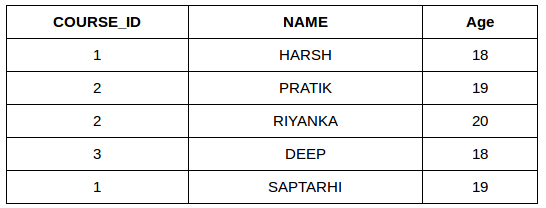
The simplest Join is INNER JOIN.

**INNER JOIN:** The INNER JOIN keyword selects all rows from both the tables as long as the condition satisfies. This keyword will create the result-set by combining all rows from both the tables where the condition satisfies i.e value of the common field will be same.  
  
SELECT StudentCourse.COURSE\_ID, Student.NAME, Student.AGE FROM Student

INNER JOIN StudentCourse

ON Student.ROLL\_NO = StudentCourse.ROLL\_NO;

**Output**:



**LEFT JOIN**: This join returns all the rows of the table on the left side of the join and matching rows for the table on the right side of join. The rows for which there is no matching row on right side, the result-set will contain *null*. LEFT JOIN is also known as LEFT OUTER JOIN.

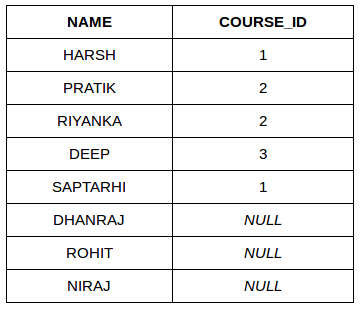
SELECT Student.NAME,StudentCourse.COURSE\_ID

FROM Student

LEFT JOIN StudentCourse

ON StudentCourse.ROLL\_NO = Student.ROLL\_NO;

**Output**:



**3. RIGHT JOIN**: RIGHT JOIN is similar to LEFT JOIN. This join returns all the rows of the table on the right side of the join and matching rows for the table on the left side of join. The rows for which there is no matching row on left side, the result-set will contain *null*. RIGHT JOIN is also known as RIGHT OUTER JOIN

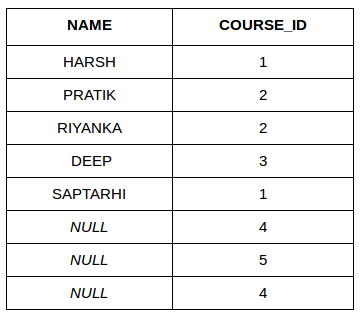
SELECT Student.NAME,StudentCourse.COURSE\_ID

FROM Student

RIGHT JOIN StudentCourse

ON StudentCourse.ROLL\_NO = Student.ROLL\_NO;

**Output:**

****

**4. FULL JOIN: FULL JOIN creates the result-set by combining result of both LEFT JOIN and RIGHT JOIN. The result-set will contain all the rows from both the tables. The rows for which there is no matching, the result-set will contain *NULL* values.**

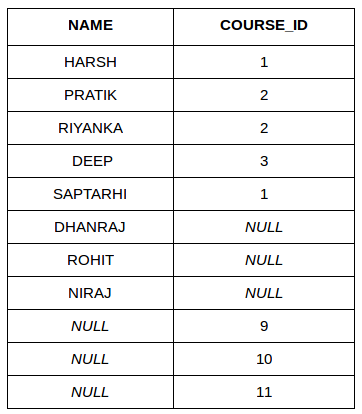
**SELECT Student.NAME,StudentCourse.COURSE\_ID**

**FROM Student**

**FULL JOIN StudentCourse**

**ON StudentCourse.ROLL\_NO = Student.ROLL\_NO;**

**Output:**

****

1. How to locate the 4th highest value in a column in a row. Create your table.

Answer :

| Flat\_no | ElectricityBill |
| --- | --- |
| 101 | 1000 |
| 102 | 1010 |
| 103 | 980 |
| 104 | 1200 |
| 105 | 1500 |
| 106 | 1229 |
| 107 | 2010 |

SELECT DISTINCT ElectricityBill AS NthHighestElectricityBill

FROM Bills

ORDER BY ElectricityBill DESC

LIMIT 1

OFFSET n-1;

4th highest :

1200