SQL Datatypes

They define the type of values that can be stored in a column

DATATYPE	DESCRIPTION	USAGE
CHAR	string(0-255), can store characters of fixed length	CHAR(50)
VARCHAR	string(0-255), can store characters up to given length	VARCHAR(50)
BLOB	string(0-65535), can store binary large object	BLOB(1000)
INT	integer(-2,147,483,648 to 2,147,483,647)	INT
TINYINT	integer(-128 to 127)	TINYINT
BIGINT	integer(-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807)	BIGINT
BIT	can store x-bit values. x can range from 1 to 64	BIT(2)
FLOAT	Decimal number - with precision to 23 digits	FLOAT
DOUBLE	Decimal number - with 24 to 53 digits	DOUBLE
BOOLEAN	Boolean values 0 or 1	BOOLEAN
DATE	date in format of YYYY-MM-DD ranging from 1000-01-01 to 9999-12-31	DATE
YEAR	year in 4 digits format ranging from 1901 to 2155	YEAR

SQL Datatypes

Signed & Unsigned

numeric

- INT
- FLOAT
- DOUBLE

TINYINT UNSIGNED (0 to 255)

TINYINT (-128 to 127)

Types of SQL Commands

DDL (Data Definition Language) : create, alter, rename, truncate & drop

DQL (Data Query Language) : select

DML (Data Manipulation Language) : , insert, update & delete

DCL (Data Control Language) : grant & revoke permission to users

TCL (Transaction Control Language) : start transaction, commit, rollback e

Database related Queries

CREATE DATABASE *db_name;*

CREATE DATABASE IF NOT EXISTS *db_name;*

```
CREATE DATABASE IF NOT EXISTS college;
```

DROP DATABASE *db_name;*

DROP DATABASE IF EXISTS *db_name;*

SHOW DATABASES;

SHOW TABLES;

Table related Queries

Create

```
CREATE TABLE table_name (
    column_name1 datatype constraint,
    column_name2 datatype constraint,
);
```

```
CREATE TABLE student (
    rollno INT PRIMARY KEY,
    name VARCHAR(50)
);
```

Table related Queries

Select & View ALL columns

SELECT * FROM *table_name;*

```
SELECT * FROM student;
```

Table related Queries

Insert

INSERT INTO *table_name*
(colname1, colname2)
VALUES
(col1_v1, col2_v1),
(col1_v2, col2_v2);

```
INSERT INTO student
(rollno, name)
VALUES
(101, "karan"),
(102, "arjun");
```

Constraints

SQL constraints are used to specify rules for data in a table.

NOT NULL columns cannot have a null value

```
col1 int NOT NULL
```

UNIQUE all values in column are different

```
col2 int UNIQUE
```

PRIMARY KEY makes a column unique & not null but used only for one

```
id int PRIMARY KEY
```

```
CREATE TABLE temp (
  id int not null,
  PRIMARY KEY (id)
);
```

Constraints

FOREIGN KEY prevent actions that would destroy links between tables

```
CREATE TABLE temp (
  cust_id int,
  FOREIGN KEY (cust_id) references customer(id)
);
```

DEFAULT sets the default value of a column

```
salary INT DEFAULT 25000
```

Constraints

CHECK it can limit the values allowed in a column

```sql
CREATE TABLE city (
  id INT PRIMARY KEY,
  city VARCHAR(50),
  age INT,
  CONSTRAINT age_check CHECK (age >= 18 AND city="Delhi")
);
```

```sql
CREATE TABLE newTab (
  age INT CHECK (age >= 18)
);
```

Select in Detail

used to select any data from the database

Basic Syntax

```
SELECT col1, col2 FROM table_name;
```

To Select ALL

```
SELECT * FROM table_name;
```

Where Clause

To define some conditions

```
SELECT col1, col2 FROM table_name
WHERE conditions;
```

```sql
SELECT * FROM student WHERE marks > 80;
SELECT * FROM student WHERE city = "Mumbai";
```

Where Clause

Using Operators in WHERE

Arithmetic Operators : +(addition) , -(subtraction), *(multiplication), /(division), %(modulus)

Comparison Operators : = (equal to), != (not equal to), > , >=, <, <=

Logical Operators : AND, OR , NOT, IN, BETWEEN, ALL, LIKE, ANY

Bitwise Operators : & (Bitwise AND), | (Bitwise OR)

Operators

Between (selects for a given range)

```
SELECT * FROM student WHERE marks BETWEEN 80 AND 90;
```

In (matches any value in the list)

```
SELECT * FROM student WHERE city IN ("Delhi", "Mumbai");
```

NOT (to negate the given condition)

```
SELECT * FROM student WHERE city NOT IN ("Delhi", "Mumba
```

Limit Clause

Sets an upper limit on number of (tuples)rows to be returned

```
SELECT * FROM student LIMIT 3;
```

```
SELECT col1, col2 FROM table_name
LIMIT number;
```

Order By Clause

To sort in ascending (ASC) or descending order (DESC)

```
SELECT * FROM student
ORDER BY city ASC;
```

```
SELECT col1, col2 FROM table_name
ORDER BY col_name(s) ASC;
```

Aggregate Functions

Aggregare functions perform a calculation on a set of values, and return a single value.

- COUNT( )
- MAX( )
- MIN( )
- SUM( )
- AVG( )

Get Maximum Marks

```
SELECT max(marks)
FROM student;
```

Get Average marks

```
SELECT avg(marks)
FROM student;
```

Group By Clause

Groups rows that have the same values into summary rows.
It collects data from multiple records and groups the result by one or more column.

*Generally we use group by with some *aggregation function*.

Count number of students in each city

```sql
SELECT city, count(name)
FROM student
GROUP BY city;
```

Having Clause

Similar to Where i.e. applies some condition on rows.

Used when we want to apply any condition after grouping.

Count number of students in each city where max marks cross 90.

```
SELECT count(name), city
FROM student
GROUP BY city
HAVING max(marks) > 90;
```

General Order

```
SELECT column(s)
FROM table_name
WHERE condition
GROUP BY column(s)
HAVING condition
ORDER BY column(s) ASC;
```

Where

↓

rows

having

↓

groups

Table related Queries

Update (to update existing rows)

```
UPDATE table_name
SET col1 = val1, col2 = val2
WHERE condition;
```

```
UPDATE student
SET grade = "O"
WHERE grade = "A";
```

Table related Queries

Delete (to delete existing rows)

DELETE FROM *table_name*
WHERE *condition;*

```
DELETE FROM student
WHERE marks < 33;
```

```
CREATE TABLE student (
  id INT PRIMARY KEY,
  courseID INT,
  FOREIGN KEY(courseID) REFERENCES course(id)
  ON DELETE CASCADE
  ON UPDATE CASCADE
);
```

Table related Queries

Alter (to change the schema)

ADD Column

```
ALTER TABLE table_name
ADD COLUMN column_name datatype constraint;
```

DROP Column

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

RENAME Table

```
ALTER TABLE table_name
RENAME TO new_table_name;
```

Table related Queries

CHANGE Column (rename)

```
ALTER TABLE table_name
CHANGE COLUMN old_name new_name new_datatype new_constraint;
```

MODIFY Column (modify datatype/ constraint)

```
ALTER TABLE table_name
MODIFY col_name new_datatype new_constraint;
```

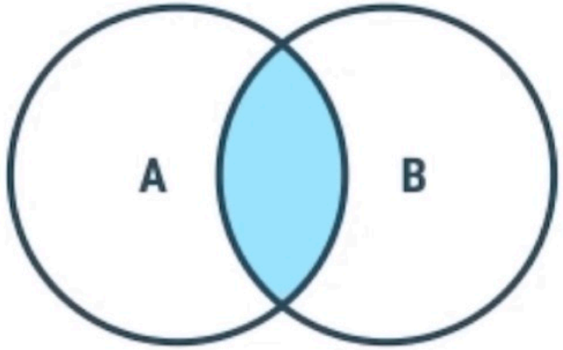
Table related Queries

Truncate (to delete table's data)

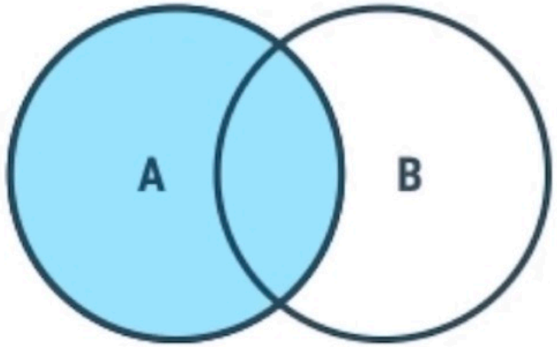
```
TRUNCATE TABLE table_name ;
```

```
UPDATE student
SET grade = "O"
WHERE grade = "A";
```

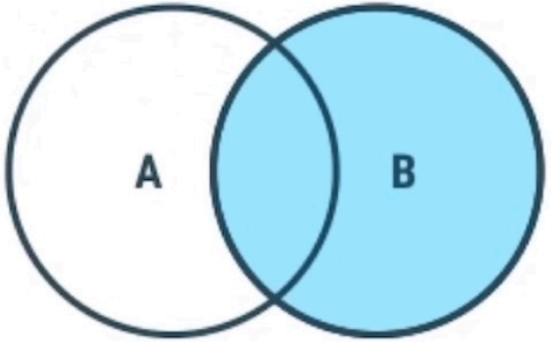
Types of Joins



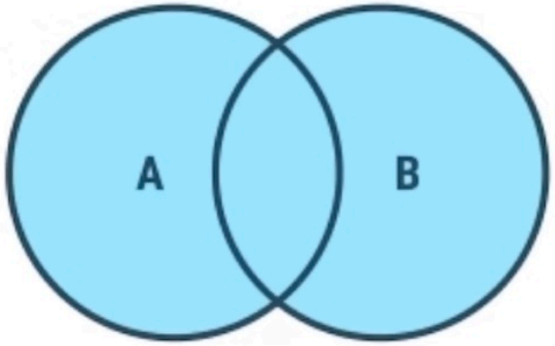
Inner Join



Left Join



Right Join

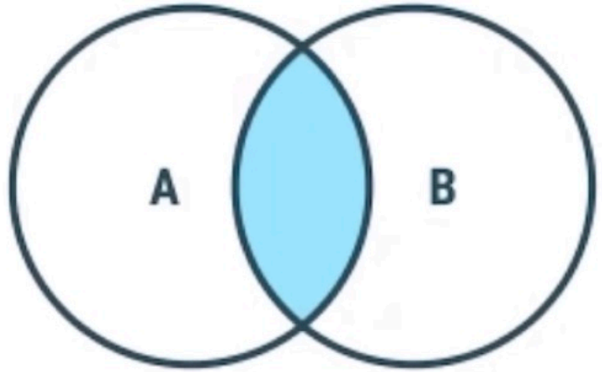


Full Join

Outer Joins

Inner Join

Returns records that have matching values in both tables

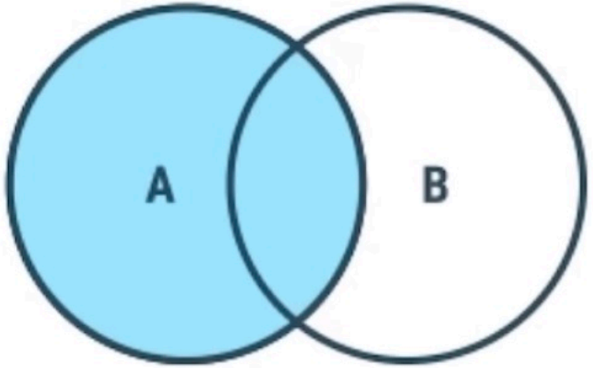


Syntax

```
SELECT column(s)
FROM tableA
INNER JOIN tableB
ON tableA.col_name = tableB.col_name;
```

Left Join

Returns all records from the left table, and the matched records from the right table

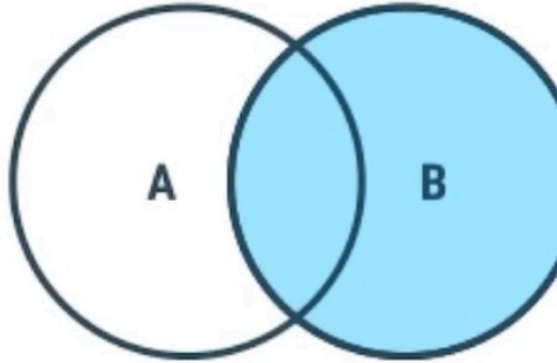


Syntax

```
SELECT column(s)
FROM tableA
LEFT JOIN tableB
ON tableA.col_name = tableB.col_name;
```

Right Join

Returns all records from the right table, and the matched records from the left table

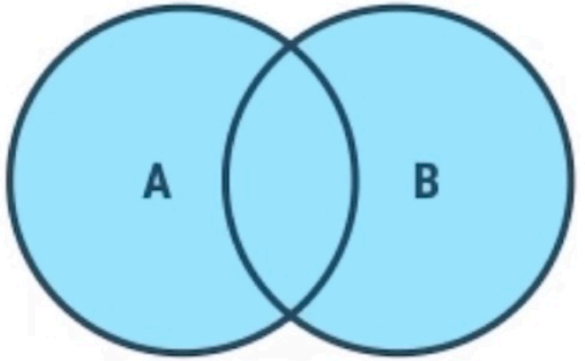


Syntax

```
SELECT column(s)
FROM tableA
RIGHT JOIN tableB
ON tableA.col_name = tableB.col_name;
```

Full Join

Returns all records when there is a match in either left or right table



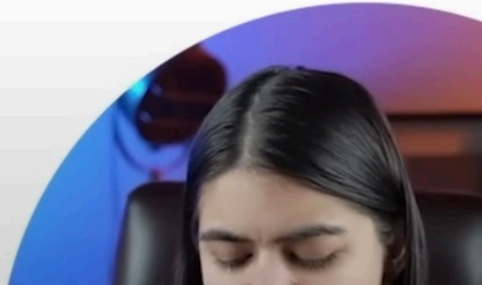
Syntax in MySQL

```
SELECT * FROM student as a
LEFT JOIN course as b
ON a.id = b.id
UNION
SELECT * FROM student as a
RIGHT JOIN course as b
ON a.id = b.id;
```

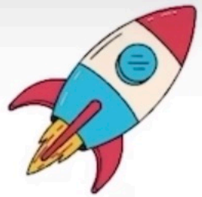
LEFT JOIN

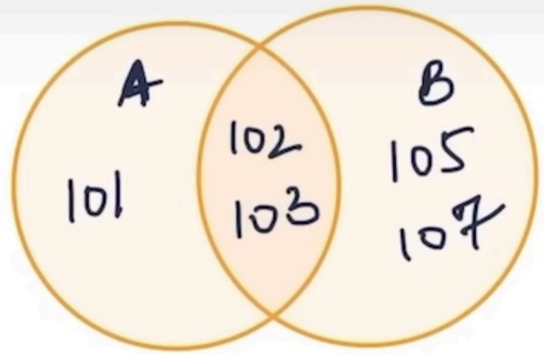
UNION

RIGHT JOIN



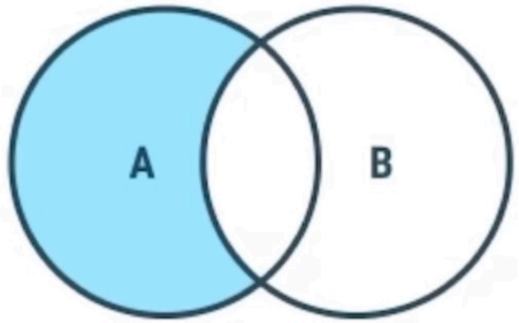
Think & Ans



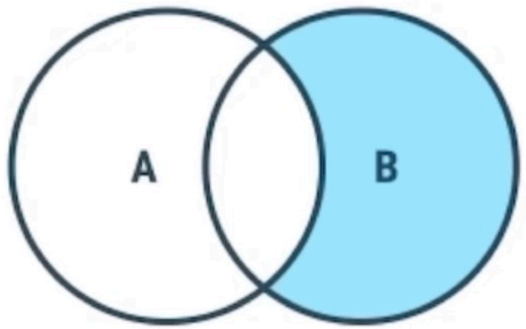


a.id is NULL

Qs: Write SQL commands to display the right exclusive join :



Left Exclusive Join



Right Exclusive Join

Right

```sql
SELECT *
FROM student as a
LEFT JOIN course as b
ON a.id = b.id
WHERE b.id IS NULL;
```

Self Join

It is a regular join but the table is joined with itself.

Syntax

```
SELECT column(s)
FROM table as a
JOIN table as b
ON a.col_name = b.col_name;
```

Union

It is used to combine the result-set of two or more SELECT statements.

Gives UNIQUE records.

To use it :

- every SELECT should have same no. of columns
- columns must have similar data types
- columns in every SELECT should be in same order

Syntax

```
SELECT column(s) FROM tableA
UNION
SELECT column(s) FROM tableB
```

SQL Sub Queries

A Subquery or Inner query or a Nested query is a query within another SQL query.

It involves 2 select statements.

Syntax

```
SELECT column(s)
FROM table_name
WHERE col_name operator
( subquery );
```

1) Select
2) From
3) Where