DAY-4

**Q.1 Write a query which uses join and let us know the internal processing of the query?**

The INNER JOIN query retrieves records from only those rows of both the tables in the JOIN query, where there is a match found between the values for the columns on which the INNER JOIN is being applied.

select OrderID,OrderDate,shipperName

from orders

inner join shipper

on orders.OrderID=shipper.shipperID;

**Q.2 Write a query to calculate age based on the date of birth**

CREATE TABLE `priyanka\_sql`.`student` (

`Stu\_id` INT NOT NULL,

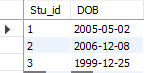
`DOB` DATE NULL,

PRIMARY KEY (`Stu\_id`));

INSERT INTO `priyanka\_sql`.`student` (`Stu\_id`, `DOB`) VALUES ('1', '2005/05/02');

INSERT INTO `priyanka\_sql`.`student` (`Stu\_id`, `DOB`) VALUES ('2', '2006/12/08');

INSERT INTO `priyanka\_sql`.`student` (`Stu\_id`, `DOB`) VALUES ('3', '1999/12/25');



SELECT DATE\_FORMAT(FROM\_days(DATEDIFF(now(),dob)), '%Y')+0 AS Age from student;



**Q.3 Write a query which demonstrate date related functions.**

i. CURDATE();

Returns the current date as a value in 'YYYY-MM-DD' or YYYYMMDD format, depending on whether the function is used in a string or in a numeric context

select curdate();



ii. CURTIME();

Returns the current time as a value in 'HH:MM:SS' or HHMMSS format, depending on whether the function is used in a string or in a numeric context. The value is expressed in the current time zone.select curtime();



iii. DATEDIFF()

DATEDIFF() returns expr1 . expr2 expressed as a value in days from one date to the other. Both expr1 and expr2 are date or date-and-time expressions. Only the date parts of the values are used in the calculation.

SELECT DATEDIFF('1997-12-31 23:59:59','1997-12-28');



iv. DATE\_FORMAT

**%Y**

Year, numeric, four digits

**%W**

Weekday name (Sunday..Saturday)

**%M**

Month name (January..December)

SELECT DATE\_FORMAT('2001-12-08 23:05:00', '%W %M %Y');



v. DAYNAME(date)

Returns the name of the weekday for date.

SELECT DAYNAME('2017-08-28');



vi. DAYOFWEEK(date)

Returns the weekday index for date (1 = Sunday, 2 = Monday, ., 7 = Saturday). These index values correspond to the ODBC standard.

SELECT DAYOFWEEK('1998-02-03');



vii. DAYOFYEAR(date)

Returns the day of the year for date, in the range 1 to 366.

SELECT DAYOFYEAR('2001-12-08');



viii. EXTRACT(unit FROM date)

The EXTRACT() function uses the same kinds of unit specifiers as DATE\_ADD() or DATE\_SUB(), but extracts parts from the date rather than performing date arithmetic

SELECT EXTRACT(YEAR FROM '2022-07-02');



ix. LAST\_DAY(date)

Takes a date or datetime value and returns the corresponding value for the last day of the month. Returns NULL if the argument is invalid.

SELECT LAST\_DAY('2022-06-23');



x. NOW()

Returns the current date and time as a value in 'YYYY-MM-DD HH:MM:SS' or YYYYMMDDHHMMSS format, depending on whether the function is used in a string or numeric context. This value is expressed in the current time zone.

select now();



xi. STR\_TO\_DATE(str,format)

This is the inverse of the DATE\_FORMAT() function. It takes a string str and a format string format. The STR\_TO\_DATE() function returns a DATETIME value if the format string contains both date and time parts. Else, it returns a DATE or TIME value if the string contains only date or time parts.

SELECT STR\_TO\_DATE('04/15/2002', '%m/%d/%Y');



xii. SYSDATE()

Returns the current date and time as a value in 'YYYY-MM-DD HH:MM:SS' or YYYYMMDDHHMMSS format, depending on whether the function is used in a string or in a numeric context.

SELECT SYSDATE();



xiii. WEEK(date[,mode])

This function returns the week number for date. The two-argument form of WEEK() allows you to specify whether the week starts on a Sunday or a Monday and whether the return value should be in the range from 0 to 53 or from 1 to 53.

SELECT WEEKDAY('2005-06-24 22:23:00');



xiv. WEEKOFYEAR(date)

Returns the calendar week of the date as a number in the range from 1 to 53. WEEKOFYEAR() is a compatibility function that is equivalent to WEEK(date,3).

SELECT WEEKOFYEAR('2018-08-20');



xv. YEAR(date)

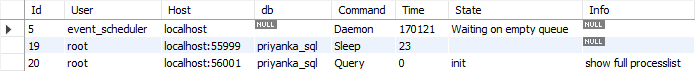
Returns the year for date, in the range 1000 to 9999, or 0 for the .zero. date.

SELECT YEAR('23-02-03');

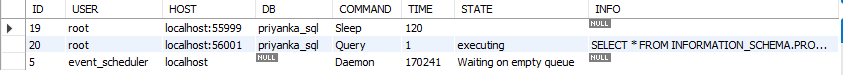


**Q.4 Write a query to check the current running processes**

show full processlist;



SELECT \* FROM INFORMATION\_SCHEMA.PROCESSLIST;



**Q.5 Write a query to insert values in a specific table of your choice in which one of the column has unique constrains and if one tries to insert data which is duplicate.**

INSERT INTO `priyanka\_sql`.`student` (`Stu\_id`, `DOB`, `stu\_name`) VALUES ('6', '2006-09-18', 'PRIYANKA');

ALTER TABLE `priyanka\_sql`.`student`

ADD UNIQUE INDEX `stu\_name\_UNIQUE` (`stu\_name` ASC) VISIBLE;

INSERT INTO student (Stu\_id,DOB,stu\_name) VALUES (6,'2006-09-18','PRIYANKA')

ON DUPLICATE KEY UPDATE stu\_name='PRIYANKA';

UPDATE student SET stu\_name='PRIYANKA' WHERE Stu\_id=6;

|  |  |  |
| --- | --- | --- |
|  |  |  |

**Q.6 Write queries which include inner joins along with subqueries**

select \*

from

(select OrderID,CustomerID

from orders) orders

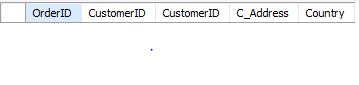
inner join

(select CustomerID,C\_Address,Country

from customers

where CustomerID>2) customers

on orders.OrderID=customers.CustomerID;



**Q.7 Go through the concept of Deadlocks and how can we avoid it**

A deadlock is a situation in which two computer programs sharing the same resource are effectively preventing each other from accessing the resource, resulting in both programs ceasing to function. A deadlock in the operating system is a situation of indefinite blocking of one or more processes that compete for resources.

Deadlock involves resources needed by two or more processes at the same time that cannot be shared. We can understand this from the above example, two cars require the road at the same time but it cannot be shared as it is one way.

There are four necessary conditions for deadlock. Deadlock happens only when all four conditions occur simultaneously for unshareable single instance resources.

The conditions for deadlock are:

1. Mutual exclusion
2. Hold and wait
3. No preemption
4. Circular wait.

**There are three ways to handle deadlock:**

1. **Deadlock prevention:** The possibility of deadlock is excluded before making requests, by eliminating one of the necessary conditions for deadlock.

**Example:** Only allowing traffic from one direction, will exclude the possibility of blocking the road.

1. **Deadlock avoidance:** Operating system runs an algorithm on requests to check for a safe state. Any request that may result in a deadlock is not granted.

**Example:** Checking each car and not allowing any car that can block the road. If there is already traffic on road, then a car coming from the opposite direction can cause blockage.

1. **Deadlock detection & recovery:** OS detects deadlock by regularly checking the system state, and recovers to a safe state using recovery techniques. **Example:** Unblocking the road by backing cars from one side. Deadlock prevention and deadlock avoidance are carried out before deadlock occurs.

**Q.8 Check out for the commands which can help us to trigger dead lock**

CREATE TABLE `abc` ( `id` INT NOT NULL AUTO\_INCREMENT, `name` VARCHAR(255) NOT NULL, `marks` INT NOT NULL, PRIMARY KEY (`id`) ) ;

INSERT INTO abc (id, name, marks) VALUES (1, "abc", 5);

INSERT INTO abc (id, name, marks) VALUES (2, "xyz", 1);

BEGIN;

UPDATE abc SET marks=marks-1 WHERE id=1;

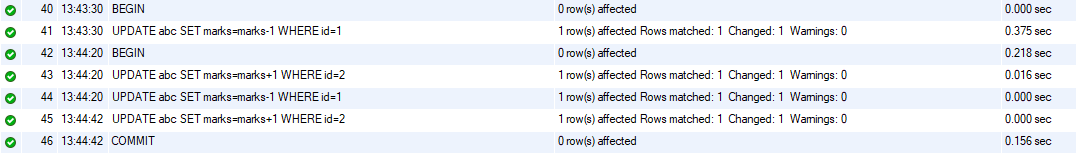
BEGIN;

UPDATE abc SET marks=marks+1 WHERE id=2;

UPDATE abc SET marks=marks-1 WHERE id=1;

UPDATE abc SET marks=marks+1 WHERE id=2;

COMMIT;



select \* from priyanka\_sql.abc;

