## Database Definition

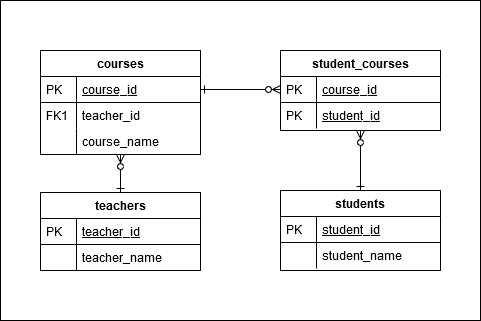
CREATE TABLE teachers ( teacher\_id BIGINT PRIMARY KEY , teacher\_name VARCHAR(64));

CREATE TABLE courses ( course\_id VARCHAR(16) PRIMARY KEY , course\_name VARCHAR(128) NOT NULL  
 , teacher\_id BIGINT , FOREIGN KEY (teacher\_id) REFERENCES teachers(teacher\_id));

CREATE TABLE students ( student\_id BIGINT PRIMARY KEY , student\_name VARCHAR(64));

CREATE TABLE student\_courses ( course\_id VARCHAR(16) , student\_id BIGINT , FOREIGN KEY(course\_id) REFERENCES courses(course\_id) , FOREIGN KEY (student\_id) REFERENCES students(student\_id) , PRIMARY KEY (course\_id, student\_id));

**1/ Draw an entity relationship diagram (or state in words the entity relationships) that accurately describes this database.**



1.student is enrolled in 0 or more courses  
2.teacher teaches 0 or more courses  
3.course has 0 or more students enrolled there is a many-to-many relationship between students and courses  
4.there is a one-to-many relationship between teachers and courses

**2/ Implement a query to get a list of all students and how many courses each student is enrolled in.**

SELECT s.student\_id, s.student\_name, COUNT(sc.course\_id) AS course\_count  
FROM students s  
LEFT JOIN student\_courses sc  
ON s.student\_id = sc.student\_id  
GROUP BY  
s.student\_id , s.student\_name;

**3/ Implement a query that shows the number of full-time and part-time students. A full-time student is enrolled in at least 4 courses. A part-time student is enrolled in at least 1 course, but no more than 3.**

WITH enrolled\_student\_course\_counts AS (  
SELECT s.student\_id, s.student\_name, COUNT(sc.course\_id) AS course\_count  
FROM students s  
LEFT JOIN student\_courses sc  
ON s.student\_id = sc.student\_id  
GROUP BY  
s.student\_id, s.student\_name  
HAVING COUNT(sc.course\_id) > 0)  
, student\_enrollment\_statuses AS (  
SELECT student\_id, student\_name  
, CASE WHEN course\_count >= 4 THEN 'full-time'  
WHEN course\_count BETWEEN 1 AND 3 THEN 'part-time'  
END AS student\_enrollment\_status  
FROM enrolled\_student\_course\_counts)  
SELECT  
UPPER(student\_enrollment\_status) AS student\_enrollment\_status  
, COUNT(student\_enrollment\_status) AS student\_enrollment\_status\_count  
FROM student\_enrollment\_statuses  
GROUP BY student\_enrollment\_status

**4/ Write a query that shows which teacher(s) are teaching the most number of courses.**

WITH teacher\_course\_rankings AS (  
SELECT t.teacher\_id , t.teacher\_name , COUNT(c.course\_id) AS teacher\_course\_count  
 , RANK() OVER(ORDER BY COUNT(c.course\_id) DESC) AS teacher\_course\_rank FROM teachers t  
 LEFT JOIN courses c  
 ON t.teacher\_id = c.teacher\_id  
GROUP BY t.teacher\_id, t.teacher\_name)  
SELECT teacher\_id, teacher\_name  
FROM teacher\_course\_rankings  
WHERE teacher\_course\_rank = 1;

**5/ Write a query that shows which teacher(s) are teaching the least number of courses.**

WITH teacher\_course\_rankings AS (  
 SELECT t.teacher\_id , t.teacher\_name , COUNT(c.course\_id) AS teacher\_course\_count  
 , RANK() OVER (ORDER BY COUNT(c.course\_id)) AS teacher\_course\_rank  
 FROM teachers t  
 LEFT JOIN courses c  
 ON t.teacher\_id = c.teacher\_id  
GROUP BY t.teacher\_id , t.teacher\_name)  
SELECT teacher\_id, teacher\_name  
FROM teacher\_course\_rankings  
WHERE teacher\_course\_rank = 1;

**6/ Write a query that shows which teacher(s) are teaching the most number of students.**

WITH teacher\_student\_rankings AS (SELECT t.teacher\_id , t.teacher\_name , COUNT(DISTINCT sc.student\_id) AS teacher\_student\_count , RANK() OVER (ORDER BY COUNT(DISTINCT sc.student\_id) DESC) AS teacher\_student\_rank FROM teachers t  
 LEFT JOIN courses c  
 ON t.teacher\_id = c.teacher\_id  
 LEFT JOIN student\_courses sc  
 ON c.course\_id = sc.course\_id  
 GROUP BY t.teacher\_id , t.teacher\_name)  
SELECT teacher\_id , teacher\_name  
FROM teacher\_student\_rankings  
WHERE teacher\_student\_rank = 1;

**7/ Write a query that shows which teacher(s) are teaching the least number of students.**

WITH teacher\_student\_rankings AS (SELECT t.teacher\_id, t.teacher\_name , COUNT(DISTINCT sc.student\_id) AS teacher\_student\_count , RANK() OVER (ORDER BY COUNT(DISTINCT sc.student\_id)) AS teacher\_student\_rank FROM teachers t LEFT JOIN courses c ON t.teacher\_id = c.teacher\_id  
LEFT JOIN student\_courses sc  
 ON c.course\_id = sc.course\_id  
GROUP BY t.teacher\_id , t.teacher\_name)  
SELECT teacher\_id, teacher\_name  
FROM teacher\_student\_rankings  
WHERE teacher\_student\_rank = 1;

**8/ Write a query that shows what the average number of courses taught by a teacher**.

WITH teacher\_course\_counts AS (SELECT t.teacher\_id , t.teacher\_name , COUNT(c.course\_id) AS teacher\_course\_count FROM teachers t LEFT JOIN courses c  
 ON t.teacher\_id = c.teacher\_id  
GROUP BY t.teacher\_id, t.teacher\_name)  
SELECT AVG(teacher\_course\_count) avg\_courses\_taught  
FROM teacher\_course\_counts;

**9/ Write a query that tells us how many students are not enrolled. Who are these unenrolled students?**

WITH student\_course\_counts AS (SELECT s.student\_id, s.student\_name, COUNT(sc.course\_id) AS course\_count FROM students s LEFT JOIN student\_courses sc ON s.student\_id = sc.student\_id  
GROUP BY s.student\_id)  
, student\_enrollment\_statuses AS (  
 SELECT student\_id, student\_name, CASE WHEN course\_count = 0 THEN 'unenrolled'  
 ELSE 'enrolled'  
 END AS student\_enrollment\_status  
 FROM student\_course\_counts )  
SELECT UPPER(student\_enrollment\_status) AS student\_enrollment\_status  
 , COUNT(student\_enrollment\_status) AS student\_enrollment\_status\_count  
FROM student\_enrollment\_statuses WHERE student\_enrollment\_status = 'unenrolled'  
GROUP BY student\_enrollment\_status;

**10/ Write a query that lists the courses in order of most popular to least popular.**

SELECT c.course\_id , c.course\_name , COUNT(sc.student\_id) AS student\_count  
FROM courses c LEFT JOIN student\_courses sc ON c.course\_id = sc.course\_id  
GROUP BY c.course\_id, c.course\_name ORDER BY 3 DESC\*;

**Ques.1. Write an SQL query to fetch the EmpId and FullName of all the employees working under Manager with id – ‘986’.**

SELECT EmpId, FullName FROM EmployeeDetails WHERE ManagerId = 986;

**Ques.2. Write an SQL query to fetch the different projects available from the EmployeeSalary table.**

SELECT DISTINCT(Project) FROM EmployeeSalary;

**Ques.3. Write an SQL query to fetch the count of employees working in project ‘P1’.**

SELECT COUNT(\*) FROM EmployeeSalary WHERE Project = 'P1';

**Ques.4. Write an SQL query to find the maximum, minimum, and average salary of the employees.**

SELECT Max(Salary), Min(Salary), AVG(Salary) FROM EmployeeSalary;

**Ques.5. Write an SQL query to find the employee id whose salary lies in the range of 9000 and 15000.**

SELECT EmpId, Salary FROM EmployeeSalary WHERE Salary BETWEEN 9000 AND 15000;

**Ques.6. Write an SQL query to fetch those employees who live in Toronto and work under manager with ManagerId – 321.**

SELECT EmpId, City, ManagerId FROM EmployeeDetails WHERE City='Toronto' AND ManagerId='321';

**Ques.7. Write an SQL query to** f**etch all the employees who either live in California or work under a manager with ManagerId – 321.**

SELECT EmpId, City, ManagerId FROM EmployeeDetails WHERE City='California' OR ManagerId='321';

**Ques.8. Write an SQL query to fetch all those employees who work on Project other than P1.**

SELECT EmpId FROM EmployeeSalary WHERE NOT Project='P1'; or

SELECT EmpId FROM EmployeeSalary WHERE Project <> 'P1';

**Ques.9. Write an SQL query to display the total salary of each employee adding the Salary with Variable value.**

SELECT EmpId, Salary+Variable as TotalSalary FROM EmployeeSalary;

**Ques.10. Write an SQL query to fetch the employees whose name begins with any two characters, followed by a text “hn” and ending with any sequence of characters.**

SELECT FullName FROM EmployeeDetails WHERE FullName LIKE ‘\_\_hn%’;

**Ques.11. Write an SQL query to fetch all the EmpIds which are present in either of the tables – ‘EmployeeDetails’ and ‘EmployeeSalary’.**

SELECT EmpId FROM EmployeeDetails UNION SELECT EmpId FROM EmployeeSalary;

**Ques.12. Write an SQL query to fetch common records between two tables.**

SELECT \* FROM EmployeeSalary INTERSECT SELECT \* FROM ManagerSalary;

SELECT \* FROM EmployeeSalary WHERE EmpId IN (SELECT EmpId from ManagerSalary);

**Ques.13. Write an SQL query to fetch records that are present in one table but not in another table.**

SELECT \* FROM EmployeeSalary MINUS SELECT \* FROM ManagerSalary;

MySQL – Since MySQL doesn’t have MINUS operator so we can use LEFT join-

SELECT EmployeeSalary.\* FROM EmployeeSalary LEFT JOIN

ManagerSalary USING (EmpId) WHERE ManagerSalary.EmpId IS NULL;

**Ques.14. Write an SQL query to fetch the EmpIds that are present in both the tables –   ‘EmployeeDetails’ and ‘EmployeeSalary.**

SELECT EmpId FROM EmployeeDetails where EmpId IN (SELECT EmpId FROM EmployeeSalary);

**Ques.15. Write an SQL query to fetch the EmpIds that are present in EmployeeDetails but not in EmployeeSalary.**

SELECT EmpId FROM EmployeeDetails where EmpId Not IN (SELECT EmpId FROM EmployeeSalary);

**Ques.16. Write an SQL query to fetch the employee full names and replace the space with ‘-’.**

SELECT REPLACE(FullName, ' ', '-') FROM EmployeeDetails;

**Ques.17. Write an SQL query to fetch the position of a given character(s) in a field.**

SELECT INSTR(FullName, 'Snow') FROM EmployeeDetails;

**Ques.18. Write an SQL query to display both the EmpId and ManagerId together.**

SELECT CONCAT(EmpId, ManagerId) as NewId FROM EmployeeDetails;

**Ques.19. Write a query to fetch only the first name(string before space) from the FullName column of the EmployeeDetails table.**

SELECT SUBSTRING(FullName, 1, CHARINDEX(' ',FullName)) FROM EmployeeDetails;

**Ques.20. Write an SQL query to upper case the name of the employee and lower case the city values.**

SELECT UPPER(FullName), LOWER(City) FROM EmployeeDetails;

**Ques.21. Write an SQL query to find the count of the total occurrences of a particular character – ‘n’ in the FullName field.**

SELECT FullName, LENGTH(FullName) - LENGTH(REPLACE(FullName, 'n', '')) FROM EmployeeDetails;

**Ques.22. Write an SQL query to update the employee names by removing leading and trailing spaces.**

UPDATE EmployeeDetails SET FullName = LTRIM(RTRIM(FullName));

**Ques.23. Fetch all the employees who are not working on any project.**

SELECT EmpId FROM EmployeeSalary WHERE Project IS NULL;

**Ques.24. Write an SQL query to fetch employee names having a salary greater than or equal to 5000 and less than or equal to 10000.**

SELECT FullName FROM EmployeeDetails WHERE EmpId IN (SELECT EmpId FROM EmployeeSalary WHERE Salary BETWEEN 5000 AND 10000);

**Ques.25. Write an SQL query to find the current date-time.**

SELECT getdate();

**Ques.26. Write an SQL query to fetch all the Employees details from EmployeeDetails table who joined in the Year 2020.**

SELECT \* FROM EmployeeDetails WHERE DateOfJoining BETWEEN '2020/01/01' AND '2020/12/31';

**Ques.27. Write an SQL query to fetch all employee records from EmployeeDetails table who have a salary record in EmployeeSalary table.**

SELECT \* FROM EmployeeDetails E WHERE EXISTS(SELECT \* FROM EmployeeSalary S WHERE E.EmpId = S.EmpId);

**Ques.28. Write an SQL query to fetch project-wise count of employees sorted by project’s count in descending order.**

SELECT Project, count(EmpId) EmpProjectCount FROM EmployeeSalary GROUP BY Project ORDER BY EmpProjectCount DESC;

**Ques.29. Write a query to fetch employee names and salary records. Display the employee details even if the salary record is not present for the employee.**

SELECT E.FullName, S.Salary

FROM EmployeeDetails E

LEFT JOIN

EmployeeSalary S

ON E.EmpId = S.EmpId;

**Ques.30. Write an SQL query to join 3 tables.**  
Ans. Considering 3 tables TableA, TableB, and TableC, we can use 2 joins clauses like below-

SELECT column1, column2

FROM TableA

JOIN TableB ON TableA.Column3 = TableB.Column3

JOIN TableC ON TableA.Column4 = TableC.Column4;

**Ques. 31. Write an SQL query to fetch all the Employees who are also managers from the EmployeeDetails table.**

SELECT DISTINCT E.FullName

FROM EmployeeDetails E

INNER JOIN EmployeeDetails M

ON E.EmpID = M.ManagerID;

**Ques.32. Write an SQL query to fetch duplicate records from EmployeeDetails (without considering the primary key – EmpId).**

SELECT FullName, ManagerId, DateOfJoining, City, COUNT(\*)

FROM EmployeeDetails

GROUP BY FullName, ManagerId, DateOfJoining, City

HAVING COUNT(\*) > 1;

**Ques.33. Write an SQL query to remove duplicates from a table without using a temporary table.**

DELETE E1 FROM EmployeeDetails E1

INNER JOIN EmployeeDetails E2

WHERE E1.EmpId > E2.EmpId

AND E1.FullName = E2.FullName

AND E1.ManagerId = E2.ManagerId

AND E1.DateOfJoining = E2.DateOfJoining

AND E1.City = E2.City;

**Ques.34. Write an SQL query to fetch only odd rows from the table.**

SELECT \* FROM EmployeeDetails

WHERE MOD (EmpId, 2) <> 0;

Using Row\_number in SQL server and checking that the remainder when divided by 2 is 1-

SELECT E.EmpId, E.Project, E.Salary

FROM (

SELECT \*, Row\_Number() OVER(ORDER BY EmpId) AS RowNumber

FROM EmployeeSalary

) E

WHERE E.RowNumber % 2 = 1;

**Ques.35. Write an SQL query to fetch only even rows from the table.**

SELECT \* FROM EmployeeDetails

WHERE MOD (EmpId, 2) = 0;

**Ques.36. Write an SQL query to create a new table with data and structure copied from another table.**

CREATE TABLE NewTable

SELECT \* FROM EmployeeSalary;

**Ques.37. Write an SQL query to create an empty table with the same structure as some other table.**

CREATE TABLE NewTable

SELECT \* FROM EmployeeSalary where 1=0;

**Ques.38. Write an SQL query to fetch top n records?**

SELECT TOP N \* FROM EmployeeSalary ORDER BY Salary DESC;

**Ques.39. Write an SQL query to find the nth highest salary from table.**

SELECT TOP 1 Salary

FROM (

SELECT DISTINCT TOP N Salary

FROM Employee

ORDER BY Salary DESC

)

ORDER BY Salary ASC;

**Ques.40. Write SQL query to find the 3rd highest salary from a table without using the TOP/limit keyword.**

SELECT Salary

FROM EmployeeSalary Emp1

WHERE 2 = (

SELECT COUNT( DISTINCT ( Emp2.Salary ) )

FROM EmployeeSalary Emp2

WHERE Emp2.Salary > Emp1.Salary )

For nth highest salary-

SELECT Salary

FROM EmployeeSalary Emp1

WHERE N-1 = (

SELECT COUNT( DISTINCT ( Emp2.Salary ) )

FROM EmployeeSalary Emp2

WHERE Emp2.Salary > Emp1.Salary )

**EMPLOYEE**

empid  -empname -managerid -  deptid  -salary  -DOB

1        emp 1       0        1       6000    1982-08-06 00:00:00.000

2        emp 2       0        5       6000    1982-07-11 00:00:00.000

3        emp 3       1        1       2000    1983-11-21 00:00:00.000

13       emp 13      2        5       2000    1984-03-09 00:00:00.000

11       emp 11      2        1       2000    1989-07-23 00:00:00.000

9        emp 9       1        5       3000    1990-09-11 00:00:00.000

8        emp 8       3        1       3500    1990-05-15 00:00:00.000

7        emp 7       2        5       NULL    NULL

3        emp 3       1        1       2000    1983-11-21 00:00:00.000

**--DEPARTMENT TABLE**

deptid   deptname

1        IT

2        Admin

**1. Employee and Manager ID are in the same table; can you get manager names for employees?**

with empCTE as

(

select e.empid, e.empname, e.managerid,

      CAST('' as varchar(50)) as Mname from employee e

where managerid = 0

union all

select e1.empid, e1.empname, e1.managerid,

       CAST(c.empname as varchar(50)) as Mname from employee e1

inner join empCTE as C on e1.managerid=c.empid

where e1.managerid>0

) select \* from empCTE

**2. Can you get employee details whose department id is not valid or department id not present in department table?**

SELECT E.EMPID,E.EMPNAME, E.DEPTID FROM EMPLOYEE E

left outer join DEPARTMENT d

on E.DEPTID = D.DEPTID

WHERE D.DEPTID IS NULL

**3. Can you get the list of employees with same salary?**

Select distinct e.empid,e.empname,e.salary

from employee e, employee e1

where e.salary =e1.salary

and e.empid != e1.empid

**4. How can you find duplicate records in Employee table?**

SELECT EMPID,EMPNAME, SALARY, COUNT(\*) AS CNT

FROM EMPLOYEE

GROUP BY EMPID,EMPNAME, SALARY

HAVING COUNT(\*)>1

**5. How can you  DELETE DUPLICATE RECORDS?**

There are multiple options to perform this operation.

1. Using row count to restrict delete only 1 record
2. **set** rowcount 1
3. **DELETE** **FROM** EMPLOYEE **WHERE** EMPID IN (
4. **SELECT** EMPID
5. **FROM** EMPLOYEE
6. **GROUP** **BY** EMPID,EMPNAME, SALARY
7. **HAVING** COUNT(\*)>1
8. )
9. **set** rowcount 0

2. Use auto increment primary key "add" if not available in the table, as in given example.

1. **alter** **table** employee
2. **add** empidpk **int** identity (1,1)

3. Now, perform query on min of auto pk id, group by duplicate check columns - this will give you latest duplicate records .

1. **select** \* **from** employee **where**
2. empidpk not in ( **select** **min**(empidpk) **from** employee
3. **group** **by** EMPID,EMPNAME, SALARY )

**5.** Now, delete.

1. **Delete** **from** employee **where**
2. empidpk not in ( **select** **min**(empidpk) **from** employee
3. **group** **by** EMPID,EMPNAME, SALARY )

**6. Find the second highest salary.**

1. **Select** **max**(Salary) **from** employee
2. **where** Salary not in (**Select** **max**(Salary) **from** employee)

**8. Can you write a query to find employees with age greater than 30?**

1. **select** \* **from** employee
2. **where** datediff(year,dob, getdate()) >30

**9. Write an SQL Query to print the name of the distinct employees whose DOB is between 01/01/1960 to 31/12/1987**

1. **SELECT** **DISTINCT** EmpName **FROM** Employee
2. **WHERE** DOB BETWEEN '01/01/1960' AND '12/31/1987'

**10.  Please write a query to get the maximum salary from each department.**

1. **select** DeptId, **max**(salary) **as** Salary **from** employee **group** **by** deptid

**11. What is wrong with the following query?**

1. **SELECT** empName **FROM** employee **WHERE** salary <> 6000

**Answer**

 The following query will not fetch record with the salary of 6000 but also will skip the record with NULL.  
As per SQL Server logic, it works on 3 values in matching conditions. TRUE or FALSE and UNKNOWN. Here,  NULL implies UNKNOWN.

 to fix this:

1. **SELECT** empName  **FROM**
2. employee **WHERE** salary **is** NULL or salary <> 6000

**12. Can you show one row twice in results from a table?**

**select** deptname **from** department d **where** d.deptname='it'

1. **union** all
2. **select** deptname **from** department d1 **where** d1.deptname='it'

 -- also cross join alias same table

1. **select** d.deptname **from** department d, department d1
2. **where** d.deptname='it'

**13. Could you tell the output or result of the following SQL statements?**

1. **select** '7'
2. -- output = 7
3. **select** 7
4. -- output = 7
5. **select** count (7)
6. -- output = 1
7. **SELECT** COUNT('7')
8. -- output = 1
9. **SELECT** COUNT(\*)
10. -- output = 1

**14. What is an alternative for TOP clause in SQL?**

-- Alternative - ROWCOUNT function

1. **Set** rowcount 3
2. **Select** \* **from** employee **order** **by** empid **desc**
3. **Set** rowcount 0

-- Alternative and  WITH and ROWNUMBER function

-- between 1 and 2

1. **With** EMPC **AS**
2. ( **SELECT** empid, empname,salary,
3. ROW\_NUMBER() OVER (**order** **by** empid **desc**) **as** RowNumber
4. **FROM** employee )
5. **select** \*
6. **from** EMPC
7. **Where** RowNumber Between 1 and 7

**15.  Will the following statements  run or give error?**NO error.

1. **SELECT** COUNT(\*) + COUNT(\*)
2. **Output**  = 2
3. **SELECT** (**SELECT** 'c#')
4. **Output** = c#

**16. Can you write a query to get employee names starting with a vowel?**

Using like operator and expression,

1. **Select** empid, empname **from** employee **where** empname like '[aeiou]%'

**17. Can you write a query to get employee names ending with a vowel?**

1. **Select** empid, empname **from** employee **where** empname like '%[aeiou]'

**18. Can you write a query to get employee names starting and ending with a vowel?**

Here you will get only one record of "empone".

1. **select** empid, empname **from** employee **where** empname like '[aeiou]%[aeiou]'

**19.  Write a query to get employees whos ID is even.**

1. **select** \* **from** employee
2. **where** empid %2 =0

**20. Write a query to get employees whos ID is an odd number.**

1. **select** \* **from** employee
2. **where** empid %2 !=0

**21. How can you get random employee record from the table?**

1. **select** **top** 1 \* **from** employee **order** **by** newid()

**22.(Tricky) Below is the table data which has 1 columns and 7 rows**

1. **Table** -TESTONE
2. DATACOL
3. 10/12
4. 1a/09
5. 20/14
6. 20/1c
7. 3112
8. 11/16
9. mm/pp

**Give data in a table is of format 'NN/NN', verify that the first and last two characters are numbers and that the middle character is '/'.**

**Answer**

Print the expression 'NUMBER' if valid, 'NOT NUM' if not valid.  
  
This can be done using like operator and expression. Checking numbers and not characters.

1. **SELECT** DataCol, 'CHECK' =
2. CASE
3. **WHEN** datacol like '%[0-9]%[^A-Z]%/%[^A-Z]%[0-9]%' **then** 'NUMBER'
4. **else** 'NOT NUM'
5. **end**
6. **from** TestOne

**23. Consider following 3 tables with one column**

1. Tbl1
2. col1
3. 1
4. 1
5. 1
6. Tbl2
7. col1
8. 2
9. 2
10. 2
11. Tbl3
12. col1
13. 3
14. 3
15. 3

**How many rows will following query return? (0, 3 or 9)**

1. **Select** \* **from** Tbl1 **inner** join tbl2 **on** tbl1.col1=tbl2.col1
2. Left outer join Tbl3 **on** Tbl3.Col1=Tbl2.Col1

**Answer- 0 .**

**24. If all values from tbl2 are deleted. What will be the output of the following query?**

select Tbl1.\* from tbl1,tbl2

Ans - 0 Rows.

**25. Can you write a query to print prime numbers from 1 to 100?**  
For this, we have to use a loop as in other programming languages.

1. **DECLARE**
3. @i **INT**,
4. @a **INT**,
5. @count **INT**,
6. @result **varchar**(**Max**)
8. **SET** @i = 1
9. **set** @result=''
11. WHILE (@i <= 100)
12. **BEGIN**
13. **SET** @count = 0
14. **SET** @a = 1
15. -- logic to check prime number
16. WHILE (@a <= @i)
17. **BEGIN**
18. IF (@i % @a = 0)
19. **SET** @count = @count + 1
21. **SET** @a = @a + 1
22. **END**
23. IF (@count = 2)
24. **set** @result = @result+cast(@i **as** **varchar**(10))+' , '
26. **SET** @i = @i + 1
27. **END**
29. **set** @result = (**select** substring(@result, 1, (len(@result) - 1)))
30. print(@result)

**26. Write query to print numbers from 1 to 100 without using loops**

This can be done using Common Table Expression without using a loop.

1. ;**with** numcte
2. **AS**
3. (
4. **SELECT** 1 [**SEQUENCE**]
6. **UNION** ALL
8. **SELECT** [**SEQUENCE**] + 1 **FROM** numcte **WHERE** [**SEQUENCE**] <100
9. )
11. **SELECT** \* **FROM** numcte

**27. What will be the output of following SQL?(tricky)**

1. **Select** $
3. Options  -
4. a. 0.00,
5. b. $,
6. c. 0,
7. d. Syntax Error
9. Answer  = 0.00

**28. What will be the output of following SQL queries?**

1. **Select** **select** 'TD'
2. Options -
3. 1. TD,
4. 2. Syntax Error,
5. **3. select** TD
7. Answer - Syntax Error. (Incorrect syntax near the keyword 'select'. )
9. **select** \* **from** 'Employee'
11. Answer -  Incorrect syntax near 'Employee' .

**29. What will be the outputs in the following SQL queries with aggregate functions?**

1. **SELECT** SUM (1+4\*5)
3. Options - a.21,      b.25,         c.Error        d.10
5. Answer -: 21

8. **SELECT** **MAX**  (1,3,8)
10. Options - a.8,        b. 12,        c.Error        d.1
12. Answer -: Error. **Max** **function** takes **only** 1 argument.

15. **SELECT** **Max** ('TD')
17. Options -  a.TD         b. Error      c. 1       d.0
19. Answer-: TD

22. **SELECT** **Max** ('TD'+'AD')
24. Options -  a.TDAD         b. Error      c. T2D       d.0
25. Answer-: TDAD

## 30. What will be the output of following queries? [Tricky involving 0]

1. **SELECT** 0/0
3. A. Divide **by** 0 error,   B. 0
4. C. NULL,                   D. Incorrect syntax error
6. Answer -:  Divide **by** 0 error
8. **SELECT**  0/6
10. A. Divide **by** 0 error,   B. 0
11. C. 6,                         D. Incorrect syntax error
13. Answer -:  0

**31 What will be the output of given statement?**

SELECT SUM (NULL)  
  
**Answer** = Error. Cannot pass null type in SUM function.  
  
Operand data type NULL is invalid for avg operator. 

**32. What will be the output of given statement?**

SELECT MAX (NULL)  
  
**Answer** = Error. Operand data type NULL is invalid for MAX operator.  
  
**33. Will following statement give error or 0 as output?**

SELECT AVG (NULL)  
  
Answer = Error. Operand data type NULL is invalid for Avg operator.   
  
**Note**MIN, MAX,SUM,AVG none of these function takes NULL parameter/argument. Also, these functions accept only one argument.

## 34. Will the following statements execute? if yes what will be output?

SELECT NULL+'1'  
  
**Answer** - Yes, no error. The output will be NULL. Perform any operation on NULL will get the NULL result.  
  
**35. Will following statement give Divide by Zero error?**

SELECT NULL/0  
  
**Answer**- No. It will execute and result will be NULL.