🔥🔥🔥Tech Mahindra🔥🔥🔥

🔥🔥🔥SQL interview questions for a Data Engineer (3-5 years of experience.)🔥🔥🔥

```
--======================
--?? 25/02/2025
--======================
--PROBLEM STATEMENT :--TOP 10 SQL
--1. SQL INTERVIEW QUESTIONS FOR A
DATA ENGINEER (3-5 YEARS OF
EXPERIENCE.)
DROP TABLE ##CITY
CREATE TABLE ##CITY(SALE_ID INT, CITY
VARCHAR (50),SALE_DATE DATE, AMOUNT
INT)
GO
INSERT INTO ##CITY(SALE_ID,
CITY,SALE_DATE, AMOUNT )VALUES
(1,'MUMBAI','2024-01-10','5000'),
(2,'DELHI','2024-01-15','7000'),
(3,'BANGALORE','2024-01-20','10000'),
(4,'CHENNAI','2024-02-05','3000'),
(5,'MUMBAI','2024-02-08','9000'),
(6,'DELHI','2024-01-18','2000'),
(7,'CHENNAI','2024-02-09','3000'),
(8,'NOIDA','2024-02-18','9000')
SELECT * FROM ##CITY

WITH MONTHLYSALES AS (
    SELECT
        FORMAT(SALE_DATE, 'YYYY-MM')
AS SALE_MONTH,
        CITY,
        SUM(AMOUNT) AS TOTAL_SALES
    FROM ##CITY
    GROUP BY FORMAT(SALE_DATE, 'YYYY-
MM'), CITY
),
RANKEDSALES AS (
    SELECT
        SALE_MONTH,
        CITY,
        TOTAL_SALES,
        RANK() OVER (PARTITION BY
SALE_MONTH ORDER BY TOTAL_SALES DESC)
AS RNK
    FROM MONTHLYSALES
)
SELECT SALE_MONTH, CITY, TOTAL_SALES
```

```
FROM RANKEDSALES
WHERE RNK <= 3
ORDER BY SALE_MONTH, RNK;
--2.WRITE AN SQL QUERY TO CALCULATE
THE RUNNING TOTAL OF SALES FOR EACH
CITY.  (SALES_DATA):
DROP TABLE ##CITY
CREATE TABLE ##CITY(SALE_ID INT, CITY
VARCHAR (50),SALE_DATE DATE, AMOUNT
INT)
GO
INSERT INTO ##CITY(SALE_ID,
CITY,SALE_DATE, AMOUNT )VALUES
(1,'MUMBAI','2024-01-10','5000'),
(2,'DELHI ','2024-01-15','7000'),
(3,'MUMBAI','2024-01-20','3000'),
(4,'DELHI ','2024-02-05','6000'),
(5,'MUMBAI','2024-02-08','8000')

SELECT CITY,SUM(AMOUNT) OVER
(PARTITION BY CITY ORDER BY
SALE_DATE)RUNNINGSALE FROM ##CITY
-- 3. FIND THE SECOND HIGHEST SALARY
OF EMPLOYEES.  (EMPLOYEES):
DROP TABLE ##EMPLOYEES
```

```
CREATE TABLE ##EMPLOYEES(EMP_ID
INT,EMP_NAME VARCHAR(50),SALARY INT,
DEPARTMENT VARCHAR(50))
GO
INSERT INTO
##EMPLOYEES(EMP_ID,EMP_NAME,SALARY,
DEPARTMENT) VALUES
(1,'RAVI ','70000','HR'),
(2,'PRIYA','90000','IT'),
(3,'KUNAL','85000','FINANCE'),
(4,'AISHA','60000','IT'),
(5,'RAHUL','95000','HR')

SELECT * FROM (
SELECT *, DENSE_RANK()OVER(ORDER BY
SALARY DESC)RNK FROM ##EMPLOYEES
)AA WHERE RNK=2

-- 4. FIND EMPLOYEES WHO HAVE THE SAME
SALARY AS SOMEONE IN THE SAME
DEPARTMENT.  (EMPLOYEE_SALARY):
DROP TABLE ##EMPLOYEE_SALARY
CREATE TABLE ##EMPLOYEE_SALARY(EMP_ID
INT,EMP_NAME VARCHAR (50),SALARY INT,
DEPARTMENT VARCHAR (50))
```

```
GO
INSERT INTO
##EMPLOYEE_SALARY(EMP_ID,EMP_NAME,SALA
RY,DEPARTMENT) VALUES
(1,'NEHA','50000','HR'),
(2,'RAVI','70000','IT'),
(3,'AMAN','50000','HR'),
(4,'POOJA','90000','IT'),
(5,'KARAN','70000','IT')
SELECT * FROM ##EMPLOYEE_SALARY ORDER
BY DEPARTMENT,SALARY

WITH CTE AS(
SELECT
DEPARTMENT,SALARY,DENSE_RANK()OVER
(PARTITION BY DEPARTMENT ORDER BY
DEPARTMENT,SALARY)RNK
FROM ##EMPLOYEE_SALARY
)SELECT DISTINCT
EMP_ID,EMP_NAME,E.SALARY,E.DEPARTMENT
FROM CTE C JOIN ##EMPLOYEE_SALARY E ON
C.DEPARTMENT=E.DEPARTMENT AND
C.SALARY=E.SALARY
WHERE RNK=1 ORDER BY
E.SALARY,E.DEPARTMENT
```

```sql
--5. WRITE AN SQL QUERY TO FIND
DUPLICATE RECORDS IN A TABLE.
(USERS):
DROP TABLE ##USERS
GO
CREATE TABLE ##USERS(USERID
INT,USERNAME VARCHAR (50),EMAIL
VARCHAR (50))
GO
INSERT INTO
##USERS(USERID,USERNAME,EMAIL)VALUES
(1,'SAMEER','SAMEER@GMAIL.COM'),
(2,'ANJALI','ANJALI@GMAIL.COM'),
(3,'SAMEER','SAMEER@GMAIL.COM'),
(4,'ROHAN','ROHAN@GMAIL.COM'),
(5,'ROHAN','ROHAN@GMAIL.COM')

SELECT * FROM (
SELECT * , DENSE_RANK()OVER(PARTITION
BY USERNAME,EMAIL ORDER BY USERID)RNK
FROM ##USERS
) AA WHERE RNK>1
```

```
-- 6. WRITE AN SQL QUERY TO DELETE
DUPLICATE ROWS WHILE KEEPING ONLY ONE
UNIQUE RECORD. (SAME SAMPLE DATA AS
QUESTION 5)

SELECT * FROM (
SELECT * , DENSE_RANK()OVER(PARTITION
BY USERNAME,EMAIL ORDER BY USERID)RNK
FROM ##USERS
) AA WHERE RNK=1

 --7. WRITE AN SQL QUERY TO PIVOT A
TABLE BY MONTHS. SAMPLE DATA
(SALES_DATA):
 DROP TABLE ##PIVOT
 CREATE TABLE ##PIVOT(SALE_ID INT,CITY
VARCHAR (20),SALE_DATE DATE, AMOUNT
INT)
  GO
INSERT INTO ##PIVOT
(SALE_ID,CITY,SALE_DATE, AMOUNT)
VALUES
(1,'MUMBAI','2024-01-10','5000'),
(2,'DELHI ','2024-02-15','7000'),
(3,'MUMBAI','2024-01-20','3000'),
(4,'DELHI ','2024-03-05','6000'),
(5,'MUMBAI','2024-02-08','8000')

SELECT * FROM ##PIVOT

SELECT CITY, ISNULL(JAN,0)JAN,
ISNULL(FEB,0)FEB, ISNULL(MAR,0)MAR
FROM (
SELECT
CITY,FORMAT(SALE_DATE,'MMM')SALE_DATE,
AMOUNT FROM ##PIVOT
)AA PIVOT (SUM(AMOUNT) FOR SALE_DATE
IN([JAN],[FEB],[MAR])) AS PT


SELECT CITY,
ISNULL([1],0)[1],ISNULL([2],0)[2],ISNU
LL([3],0)[3]
,ISNULL([1],0)+ISNULL([2],0)+ISNULL([3
],0)[GTOTAL] FROM (
SELECT CITY,MONTH(SALE_DATE)SALE_DATE,
AMOUNT FROM ##PIVOT
)AA PIVOT (SUM(AMOUNT) FOR SALE_DATE
IN([1],[2],[3])) AS PT
```

```sql
--8. FIND CUSTOMERS WHO PLACED AT
LEAST 3 ORDERS IN THE LAST 6 MONTHS.
SAMPLE DATA (ORDERS):
DROP TABLE ##ORDERS
CREATE TABLE ##ORDERS(ORDER_ID INT,
CUSTOMER_ID INT, ORDER_DATE DATE,
AMOUNT INT)
GO
INSERT INTO ##ORDERS(ORDER_ID,
CUSTOMER_ID, ORDER_DATE,AMOUNT) VALUES
(1,'101','2024-10-10','1000'),
(2,'102','2024-11-15','2000'),
(3,'101','2024-12-20','1500'),
(4,'103','2025-01-05','2500'),
(5,'101','2025-02-08','3000')

SELECT * FROM (
SELECT * ,COUNT(1)OVER(ORDER BY
CUSTOMER_ID)RNK FROM ##ORDERS WHERE
ORDER_DATE<=DATEADD(MONTH,-
6,GETDATE())
) AA WHERE RNK=3
```

```
--9. NORMALIZATION VS. DENORMALIZATION
– WHAT ARE THEY, AND WHEN SHOULD EACH
BE USED IN A DATA PIPELINE?
FEATURE<--->NORMALIZATION (OLTP)<---
>DENORMALIZATION (OLAP)
GOAL<--->REDUCE REDUNDANCY, ENSURE
INTEGRITY<--->IMPROVE READ/QUERY
PERFORMANCE
JOINS<--->MORE JOINS (COMPLEX
QUERIES)<--->FEWER JOINS (FASTER
QUERIES)
STORAGE<--->LESS STORAGE REQUIRED<---
>MORE STORAGE DUE TO REDUNDANCY
USE CASE<--->TRANSACTIONAL SYSTEMS
(BANKING, E-COMMERCE)<--->ANALYTICAL
SYSTEMS (DATA WAREHOUSES, REPORTING)
UPDATE SPEED<--->FASTER UPDATES (LESS
REDUNDANT DATA)<--->SLOWER UPDATES
(MULTIPLE COPIES OF DATA)
QUERY PERFORMANCE<--->SLOWER (DUE TO
JOINS)<--->FASTER (PRE-AGGREGATED OR
REDUNDANT DATA)
```

--10. INDEXING IN SQL – EXPLAIN CLUSTERED VS. NON-CLUSTERED INDEXES. HOW DO THEY IMPACT QUERY PERFORMANCE?

CLUSTERED INDEX
DETERMINES THE PHYSICAL ORDER OFDATA IN A TABLE. IT CHANGES THE WAY THE DATA IS STOREDON DISK AND CAN BE CREATED ON ONLY ONE COLUMN. ATABLE CAN HAVE ONLY ONE CLUSTERED INDEX.

NON-CLUSTERED INDEX
DOES NOT AFFECT THE PHYSICALORDER OF DATA IN A TABLE. IT IS STORED SEPARATELY ANDCONTAINS A POINTER TO THE ACTUAL DATA. A TABLE CANHAVE MULTIPLE NON-CLUSTERED INDEXES.