```
SELECT AVG(SALARY) FROM EMPLOYEE;
```

2. Query to Find the Second Highest Salary from an Employee Table:

```
-- METHOD 1: (USING DENSE_RANK)
SELECT * FROM (SELECT FIRST_NAME,LAST_NAME,SALARY,DENSE_RANK() OVER (ORDER BY S

-- METHOD 2: (USING LIMIT CLAUSE).
SELECT DISTINCT(SALARY) FROM Employee ORDER BY SALARY DESC LIMIT 1,1;

-- METHOD 3: (USING SELF JOIN)              HERE, [(N-1) = 2-1 = 1]
SELECT FIRST_NAME,SALARY FROM Employee AS E1 WHERE 1 = (SELECT COUNT(DISTINCT S
```

3. Query to Fetch the List of Employees with Same Salary from an Employee Table:

```
SELECT SALARY FROM Employee GROUP BY SALARY HAVING COUNT(*) > 1;
```

4. Query to Fetch Top 5 and Bottom 5 Records from an Employee Table:

```
-- Top 5 Records
SELECT * FROM EMPLOYEE LIMIT 5;
-- Bottom 5 Records
SELECT * FROM (SELECT * FROM EMPLOYEE ORDER BY EMPLOYEE_ID DESC LIMIT 5) AS K ORI
```

5. Query to Find Duplicate Rows in a Table:

```
SELECT COLUMN1, COLUMN2, COUNT(*)
FROM TABLE_NAME
GROUP BY COLUMN1, COLUMN2
HAVING COUNT(*) > 1;
```

6. Query to Fetch Random 20 Records From Table:

```
SELECT * FROM TABLE_NAME ORDER BY RAND() LIMIT 20;
```

7. Query to Fetch First 50% Records From Employee Table:

```
SELECT * FROM EMPLOYEE WHERE EMPLOYEE_ID <= (SELECT COUNT(*)/2 FROM EMPLOYEE;
```

8. Query to Fetch Random 20% of Total Records:

First We will find the output of 20% from total records i.e.

```
SELECT ROUND(COUNT(*) * 0.2) FROM TABLE_NAME;    # Round will give round-off va
```

Assume the output we get is given by ‘K’, then to fetch 20% random records,

```
SELECT * FROM TABLE_NAME ORDER BY RAND() LIMIT K;
```

9. Query to Fetch Intersecting Records From Table1 and Table2:

```
SELECT * FROM TABLE1 WHERE COLUMN1 IN (SELECT COLUMN2 FROM TABLE2);
```

Let's assume, We have Employee Table with Employee_ID (1,2,3,4,5,6,7,8) & Bonus Table with Employee_Reference_ID (1,2,3,1,2). Then:

```
SELECT * FROM EMPLOYEE WHERE EMPLOYEE_ID IN (SELECT EMPLOYEE_REFERENCE_ID FROM
```

10. Query to Create a Primary Key on a Suitable Column in a Table:

```
SELECT COLUMN,COUNT(*) FROM TABLE_NAME GROUP BY ID HAVING COUNT(*) > 1;
```

Assume ID is a column in the Table. We check on column 'ID' to see whether there is any duplicate records or not. The output we get is like :

Here, We can see that there is no duplicated records in ID column and it does not have any null values so we can make ID as primary key.

11. Query to Fetch only Odd Rows And Only Even Rows From Employee Table:

```
# To Fetch Only Odd Rows

SELECT * FROM EMPLOYEE WHERE EMPLOYEE_ID %2 != 0;

# To Show Only Even Rows
```

```
SELECT * FROM EMPLOYEE WHERE EMPLOYEE_ID %2 = 0;
```

12. Query to Count number of Employees from Employee Table:

```
SELECT COUNT(*) AS TOTAL_EMPLOYEE FROM EMPLOYEE;
```

13. Query to Show One Row Twice in Results of a Table:

```
SELECT * FROM TABLE_NAME UNION ALL SELECT * FROM TABLE_NAME;
```

14. Query to Use Subqueries to Filter Data:

```
SELECT * FROM TABLE_NAME WHERE COLUMN_NAME IN (SELECT COLUMN_NAME FROM ANOTHER_
```

15. Query to Find the Employee with the Highest Salary in Each Department From Employee Table:

```
SELECT DEPARTMENT, EMPLOYEE_ID, SALARY
FROM (SELECT DEPARTMENT, EMPLOYEE_ID, SALARY, ROW_NUMBER()
      OVER (PARTITION BY DEPARTMENT ORDER BY SALARY DESC) AS RowNum
      FROM EMPLOYEE) AS Ranked
WHERE RowNum = 1;
```

Conclusion

In this article, I have explored 15 common SQL coding interview questions that are commonly asked in interviews for data-related roles. Knowing these questions demonstrates your *SQL proficiency* as well as showcases your *problem-solving skills*. Keep learning, practicing and exploring new SQL challenges to continuously