SQL INTERVIEW PREPARATION PART 1

1. **Write a query to fetch all employees with a salary above the department average.**

Solution:

```
WITH AverageSalaryCTE AS (
   SELECT department_name, AVG(salary) AS average_salary
   FROM employees
   GROUP BY department_name
)
SELECT
   e.emp_name,
   e.department_name,
   e.salary
FROM
   employees AS e
JOIN
   AverageSalaryCTE AS cte
ON
   e.department_name = cte.department_name
WHERE
   e.salary > cte.average_salary;
```

2. **Explain different types of joins and find customers who have made purchases but were not assigned a salesperson.**

Solution:

1. INNER JOIN

- Description: Returns only the rows that have matching values in both tables.

- Usage: Use when you want to retrieve records that have a match in both tables.

- Syntax:

```
SELECT columns
```

FROM table1

INNER JOIN table2 ON table1.common_column = table2.common_column;

2. LEFT JOIN (or LEFT OUTER JOIN)

- Description: Returns all rows from the left table and the matched rows from the right table. If there is no match, NULL values will be returned for columns from the right table.

- Usage: Use when you want all records from the left table regardless of whether there is a match in the right table.

- Syntax:

SELECT columns

FROM table1

LEFT JOIN table2 ON table1.common_column = table2.common_column;

3. RIGHT JOIN (or RIGHT OUTER JOIN)

- Description: Returns all rows from the right table and the matched rows from the left table. If there is no match, NULL values will be returned for columns from the left table.

- Usage: Use when you want all records from the right table regardless of whether there is a match in the left table.

- Syntax:

SELECT columns

FROM table1

RIGHT JOIN table2 ON table1.common_column = table2.common_column;

4. FULL JOIN (or FULL OUTER JOIN)

- Description: Returns all rows when there is a match in either the left or right table. If there is no match, NULL values will be returned for non-matching columns from either table.

- Usage: Use when you want all records from both tables, regardless of matches.

- Syntax:

SELECT columns

FROM table1

FULL JOIN table2 ON table1.common_column = table2.common_column;

5. CROSS JOIN

- Description: Returns the Cartesian product of the two tables. Each row from the first table is combined with every row from the second table.

- Usage: Use when you need to combine all rows from two tables without any specific relationship.

- Syntax:

SELECT columns

FROM table1

CROSS JOIN table2;

6. SELF JOIN

- Description: A self join is a join in which a table is joined with itself. It is useful for comparing rows within the same table.

- Usage: Use when you want to compare rows within the same table.

- Syntax:

SELECT a.columns, b.columns

FROM table AS a

JOIN table AS b ON a.common_column = b.common_column;

To find customers who have made purchases but were not assigned a salesperson.

```
SELECT c.customer_id, c.customer_name

FROM customers AS c

LEFT JOIN sales AS s ON c.customer_id = s.customer_id

WHERE s.sales IS NOT NULL AND s.salesperson IS NULL;
```

3. **Calculate total sales by region, filtering regions with total sales above $1,000,000.**
   Solution:

```
SELECT region, SUM(sales) AS Total_Sales

FROM Sales_Table

GROUP BY region
```

```
HAVING SUM(sales) > 1000000;
```

4. **List all departments with an average employee salary above $60,000.**
   Solution:

```
SELECT dept_name, AVG(salary) AS avg_salary
FROM employees
GROUP BY dept_name
HAVING AVG(salary) > 60000;
```

5. **Retrieve the top 3 employees with the highest salaries in each department.**
   Solution:

```
WITH RankedSalaries AS (
   SELECT emp_name, dept_name, salary,
        ROW_NUMBER() OVER (PARTITION BY dept_name ORDER BY salary
DESC) AS salary_rank
   FROM employee
)
SELECT emp_name, dept_name, salary
FROM RankedSalaries
WHERE salary_rank <= 3;
```