

Ex.No.: 10	AGGREGATING DATA USING GROUP FUNCTIONS	
Date:		

Name: Arun Bharathi M B
Roll no:231901007

****Determine the validity of the statements:****

1. ****Group functions work across many rows to produce one result per group.****

- ****Answer:**** True

- ****Explanation:**** Group functions, such as `SUM`, `AVG`, `MAX`, `MIN`, and `COUNT`, aggregate multiple rows within a group to produce a single result for that group.

2. ****Group functions include nulls in calculations.****

- ****Answer:**** False

- ****Explanation:**** Most group functions ignore `NULL` values in their calculations, with the exception of `COUNT(*)`, which includes all rows regardless of `NULL` values.

3. ****The WHERE clause restricts rows prior to inclusion in a group calculation.****

- ****Answer:**** True

- ****Explanation:**** The `WHERE` clause filters rows before any grouping or aggregation occurs, while the `HAVING` clause is used to filter groups after the aggregation.

****SQL Queries for the HR department reports:****

4. ****Find the highest, lowest, sum, and average salary of all employees.****

```
```sql
```

```
SELECT
```

```
 ROUND(MAX(salary)) AS Maximum,
```

```
 ROUND(MIN(salary)) AS Minimum,
```

```
 ROUND(SUM(salary)) AS Sum,
```

```
 ROUND(AVG(salary)) AS Average
```

```
FROM employees;
```

```
```
```

```
```sql
```

```
SELECT
```

5. \*\*Modify the above query to display the minimum, maximum, sum, and average salary for each job type.\*\*

```
```sql  
SELECT
```

```

    job_id,
    ROUND(MAX(salary)) AS Maximum,
    ROUND(MIN(salary)) AS Minimum,
    ROUND(SUM(salary)) AS Sum,
    ROUND(AVG(salary)) AS Average
FROM employees
GROUP BY job_id;
'''

```

6. **Write a query to display the number of people with the same job, prompting for a job title.**

```

'''sql
SELECT COUNT(*) AS "Number of People"
FROM employees
WHERE job_id = :job_title; -- Replace `:job_title` with user input for interactive queries
'''

```

7. **Determine the number of managers without listing them.**

```

'''sql
SELECT COUNT(DISTINCT manager_id) AS "Number of Managers"
FROM employees
WHERE manager_id IS NOT NULL;
'''

```

8. **Find the difference between the highest and lowest salaries.**

```

'''sql
SELECT (MAX(salary) - MIN(salary)) AS DIFFERENCE
FROM employees;
'''

```

9. **Display the manager number and the salary of the lowest-paid employee for that manager, excluding any groups where the minimum salary is \$6,000 or less.**

```

'''sql
SELECT manager_id, MIN(salary) AS Salary
FROM employees
WHERE manager_id IS NOT NULL
GROUP BY manager_id
HAVING MIN(salary) > 6000
ORDER BY Salary DESC;
'''

```

10. **Display the total number of employees and the number of employees hired in 1995, 1996, 1997, and 1998.**

```

'''sql
SELECT

```

```

COUNT(*) AS "Total Employees",
SUM(CASE WHEN EXTRACT(YEAR FROM hire_date) = 1995 THEN 1 ELSE 0 END) AS
"Hired in 1995",
SUM(CASE WHEN EXTRACT(YEAR FROM hire_date) = 1996 THEN 1 ELSE 0 END) AS
"Hired in 1996",
SUM(CASE WHEN EXTRACT(YEAR FROM hire_date) = 1997 THEN 1 ELSE 0 END) AS
"Hired in 1997",
SUM(CASE WHEN EXTRACT(YEAR FROM hire_date) = 1998 THEN 1 ELSE 0 END) AS
"Hired in 1998"
FROM employees;

```

11. **Create a matrix query to display the job, the salary for that job based on department number, and the total salary for that job, for departments 20, 50, 80, and 90.**

```

```sql
SELECT
 job_id AS Job,
 SUM(CASE WHEN department_id = 20 THEN salary ELSE 0 END) AS "Dept 20 Salary",
 SUM(CASE WHEN department_id = 50 THEN salary ELSE 0 END) AS "Dept 50 Salary",
 SUM(CASE WHEN department_id = 80 THEN salary ELSE 0 END) AS "Dept 80 Salary",
 SUM(CASE WHEN department_id = 90 THEN salary ELSE 0 END) AS "Dept 90 Salary",
 SUM(salary) AS "Total Salary"
FROM employees
WHERE department_id IN (20, 50, 80, 90)
GROUP BY job_id;

```

12. \*\*Display each department's name, location, number of employees, and the average salary for all employees in that department.\*\*

```

```sql
SELECT
  d.department_name AS "Department",
  l.city AS "Location",
  COUNT(e.employee_id) AS "Number of People",
  ROUND(AVG(e.salary), 2) AS "Average Salary"
FROM employees e
JOIN departments d ON e.department_id = d.department_id
JOIN locations l ON d.location_id = l.location_id
GROUP BY d.department_name, l.city;

```