Query Statement #1

Write a query to compute for the FIRST_VALUE() given the above dataset and return the value along with the entire row.

SQL Command #1

```
SELECT *, FIRST_VALUE(row_num) OVER(ORDER BY Salary) AS First_Value_Answer
FROM Employee limit 1;
```

Query Result/Output #1

row_num	first_name	last_name	salary	First_Value_Answer
1	Karen	Colmenares	2500	1

Query Statement #2

Write a query to compute for the LAST_VALUE() and return the value along with the entire row

SQL Command #2

```
SELECT *, LAST_VALUE(row_num) OVER(ORDER BY Salary desc) AS
Last_Value_Answer FROM Employee limit 1;
```

Query Result/Output #2

row_num	first_name	last_name	salary	Last_Value_Answer
15	Charles	Johnson	6200	15

Query Statement #3

Write a query to compute for LEAD(2) for Guy and return the value along with the Guy's row.

SQL Command #3

```
SELECT * FROM(SELECT *, LEAD(SALARY, 2) OVER(ORDER BY SALARY ASC) AS
Lead_Value FROM Employee) AS fixed_table WHERE first_name = "Guy";
```

Query Result/Output #3

row_num	first_name	last_name	salary	Lead_Value
2	Guy	Himuro	2600	2800

Query Statement #4

Write a query to compute for LAG(4) for Pat and return value along with Pat's row.

SQL Command #4

```
SELECT * FROM(SELECT *, LAG(SALARY, 4) OVER(ORDER BY SALARY ASC) AS
Lag_Value FROM Employee) AS fixed_table WHERE first_name = "Pat";
```

Query Result/Output #4

row_num	first_name	last_name	salary	Lag_Value
14	Pat	Fay	6000	4400

Query Statement #5

Write a query to compute the RANK() and DENSE_RANK() and return the entire dataset, including the rank and dense rank for each employee.

SQL Command #5

```
SELECT *, RANK() OVER(ORDER BY SALARY ASC) AS rank_value, DENSE_RANK()
OVER(ORDER BY SALARY ASC) AS dense_rank_value FROM Employee;
```

Query Result/Output #5

row_num	first_name	last_name	salary	rank_value	dense_rank_value
1	Karen	Colmenares	2500	1	1
2	Guy	Himuro	2600	2	2
3	Irene	Mikkilineni	2700	3	3
4	Sigal	Tobias	2800	4	4
5	Sheli	Baida	2900	5	5
6	Alexander	Khoo	3100	6	6
7	Britney	Everett	3900	7	7
8	Sarah	Bell	4000	8	8
9	Diana	Lorentz	4200	9	9
10	Jennifer	Whalen	4400	10	10
11	David	Austin	4800	11	11
12	Valli	Pataballa	4800	11	11
13	Bruce	Ernst	6000	13	12
14	Pat	Fay	6000	13	12
15	Charles	Johnson	6200	15	13

Query Statement #6

Write a query to compute the RANK() and DENSE_RANK() but only return Valli's and Bruce's rank and dense rank.

SQL Command #6

```
SELECT * FROM(SELECT *, RANK() OVER(ORDER BY SALARY ASC) AS rank_value,
DENSE_RANK() OVER(ORDER BY SALARY ASC) AS dense_rank_value
FROM Employee) AS fixed_table WHERE first_name IN ("Valli", "Bruce");
```

Query Result/Output #6

row_num	first_name	last_name	salary	rank_value	dense_rank_value
12	Valli	Pataballa	4800	11	11
13	Bruce	Ernst	6000	13	12

Query Statement #7

Write a query to compute the ROW_NUMBER() for Irene and Sarah and only return the rows corresponding to them.

SQL Command #7

```
SELECT * FROM(SELECT *, ROW_NUMBER() OVER (ORDER BY salary ASC) AS
row_value
FROM Employee) AS fixed_table WHERE first_name in ("Irene", "Sarah");
```

Query Result/Output #7

row_num	first_name	last_name	salary	row_value
3	Irene	Mikkilineni	2700	3
8	Sarah	Bell	4000	8

Query Statement #8

Write a query to compute the PERCENT_RANK() and return the entire dataset, including the percent rank for each employee. Format your PERCENT_RANK() values to 100%.

SQL Command #8

```
SELECT * FROM(SELECT *, (ROUND(PERCENT_RANK() OVER (ORDER BY SALARY)
* 100, 2)) AS "Percent_Rank(%)"
FROM EMPLOYEE) AS fixed_table;
```

Query Result/Output #8

row_num	first_name	last_name	salary	Percent_Rank(%)
1	Karen	Colmenares	2500	0
2	Guy	Himuro	2600	7.14
3	Irene	Mikkilineni	2700	14.29
4	Sigal	Tobias	2800	21.43
5	Sheli	Baida	2900	28.57
6	Alexander	Khoo	3100	35.71
7	Britney	Everett	3900	42.86
8	Sarah	Bell	4000	50
9	Diana	Lorentz	4200	57.14
10	Jennifer	Whalen	4400	64.29
11	David	Austin	4800	71.43
12	Valli	Pataballa	4800	71.43
13	Bruce	Ernst	6000	85.71
14	Pat	Fay	6000	85.71
15	Charles	Johnson	6200	100

Query Statement #9

Write a query to compute the CUME_DIST() and return the entire dataset, including the percentage rank for each employee. Format your CUME_DIST() values to 2 decimal places.

SQL Command #9

```
SELECT * FROM(SELECT *,
ROUND(PERCENT_RANK() OVER (ORDER BY salary ASC)  * 100, 2) AS
"Percent_Rank(%)",
ROUND(CUME_DIST() OVER (ORDER BY salary ASC) , 2) AS cume_dist_value
FROM Employee) AS fixed_table;
```

Query Result/Output #9

row_num	first_name	last_name	salary	Percent_Rank(%)	cume_dist_value
1	Karen	Colmenares	2500	0	0.07
2	Guy	Himuro	2600	7.14	0.13
3	Irene	Mikkilineni	2700	14.29	0.2
4	Sigal	Tobias	2800	21.43	0.27
5	Sheli	Baida	2900	28.57	0.33
6	Alexander	Khoo	3100	35.71	0.4
7	Britney	Everett	3900	42.86	0.47
8	Sarah	Bell	4000	50	0.53
9	Diana	Lorentz	4200	57.14	0.6
10	Jennifer	Whalen	4400	64.29	0.67
11	David	Austin	4800	71.43	0.8
12	Valli	Pataballa	4800	71.43	0.8
13	Bruce	Ernst	6000	85.71	0.93
14	Pat	Fay	6000	85.71	0.93
15	Charles	Johnson	6200	100	1

Query Statement #10

Write a query to compute the NTILE(4) and return the entire dataset showing approximately equal groups/buckets.

SQL Command #10

```
SELECT *, NTILE(4) OVER (ORDER BY SALARY) AS bucket_no FROM Employee;
```

Query Result/Output #10

row_num	first_name	last_name	salary	bucket_no
1	Karen	Colmenares	2500	1
2	Guy	Himuro	2600	1
3	Irene	Mikkilineni	2700	1
4	Sigal	Tobias	2800	1
5	Sheli	Baida	2900	2
6	Alexander	Khoo	3100	2
7	Britney	Everett	3900	2
8	Sarah	Bell	4000	2
9	Diana	Lorentz	4200	3
10	Jennifer	Whalen	4400	3
11	David	Austin	4800	3
12	Valli	Pataballa	4800	3
13	Bruce	Ernst	6000	4
14	Pat	Fay	6000	4
15	Charles	Johnson	6200	4