

## Practice project 8.15 Employee Data Analysis.

### Course 4

### SQL Training

#### Task

1. Write a query to **create** an **employee table** with the fields employee id, first name, last name, job id, salary, manager id, and department id.

The screenshot shows the SQL Developer interface. The left pane displays the 'SCHEMAS' tree with 'employee\_data\_analysis' expanded. The central editor shows the SQL script to create the 'Employee\_Table' with the following fields: Emp\_id (int), F\_name (varchar(40)), L\_name (varchar(30)), Job\_id (nchar(10)), Salary (int), manager\_id (int), and Dept\_id (int). The right pane shows a message: 'Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.' The bottom pane shows the 'Output' window with the 'Action Output' tab selected, displaying the execution results of the SQL script.

#	Time	Action	Message	Duration / Fetch
1	16:58:09	SELECT * FROM patient_diagnosis_report.patient...	8 row(s) returned	0.110 sec / 0.000 sec
2	16:59:31	CREATE DATABASE Employee_Data_Analysis	1 row(s) affected	0.156 sec
3	16:59:58	Use Employee_Data_Analysis	0 row(s) affected	0.000 sec
4	17:05:25	CREATE TABLE Employee_Table( Emp_id int, F_...	0 row(s) affected, 1 warning(s): 3720 NATIONAL/...	1.406 sec

The screenshot shows the SQL Developer interface with the 'employee\_table' selected in the 'SCHEMAS' tree. The central editor displays the table structure with columns: Emp\_id, F\_name, L\_name, Job\_id, Salary, manager\_id, and Dept\_id. The right pane shows a message: 'Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.' The bottom pane shows the 'Output' window with the 'Action Output' tab selected, displaying the execution results of the SQL script.

Emp_id	F_name	L_name	Job_id	Salary	manager_id	Dept_id
--------	--------	--------	--------	--------	------------	---------

2. Write a query to **insert** values into the employee table. (Imported the dataset into Employee\_table...)

The screenshot shows the SQL Developer interface. The left pane displays the 'SCHEMAS' tree with 'employee\_data\_analysis' expanded, showing 'employee\_table' under 'Tables'. The main pane shows a SQL query: `SELECT * FROM employee_data_analysis.employee_table;`. Below the query, the 'Result Grid' displays 10 rows of employee data.

Emp_id	F_name	L_name	Job_id	Salary	manager_id	Dept_id
101	ankit	jain	HP 124	200000	2	24
102	sarvesh	patel	HP 123	150000	2	24
103	krishna	gee	HP 125	500000	5	44
104	rana	gee	HP 122	250000	3	54
105	soniya	jain	HP 121	400000	1	22
106	nithin	kumar	HP 120	300000	4	34
107	karan	patel	HP 126	300001	2	34
108	shilpa	jain	HP 127	300001	5	24
109	mukesh	singh	HP 128	300001	4	44

3. Write a query to find the **first name** and **salary** of the employee whose **salary is higher than the employee with the last name Kumar** from the employee table.

The screenshot shows the SQL Developer interface. The left pane displays the 'SCHEMAS' tree with 'employee\_data\_analysis' expanded, showing 'employee\_table' under 'Tables'. The main pane shows a SQL query: `SELECT f_name, l_name, Salary from Employee_table where salary > (select salary from Employee_table where l_name = 'kumar');`. Below the query, the 'Result Grid' displays 5 rows of employee data.

f_name	l_name	Salary
krishna	gee	500000
soniya	jain	400000
karan	patel	300001
shilpa	jain	300001
mukesh	singh	300001

4. Write a query to display the **employee id** and **last name** of the employee whose **salary is greater than the average salary** from the employee table.

The screenshot shows the SQL Developer interface. On the left, the 'SCHEMAS' pane displays a tree view with 'employee\_data\_analy' expanded, showing 'employee\_table'. The main editor window, titled 'SQL File 5\*', contains the following SQL query:

```
1 • SELECT emp_id, l_name from Employee_table
2   where salary > ALL( Select AVG(Salary) from Employee_table)
3
```

Below the query editor, the 'Result Grid' tab is active, displaying the results of the query in a table:

emp_id	l_name
103	gee
105	jain
107	patel
108	jain
109	singh

5. Write a query to display the **employee id**, **first name**, and **salary** of the employees who earn a **salary that is higher than the salary of all the shipping clerks (JOB\_ID = HP122)**. Sort the results of the salary in ascending order.

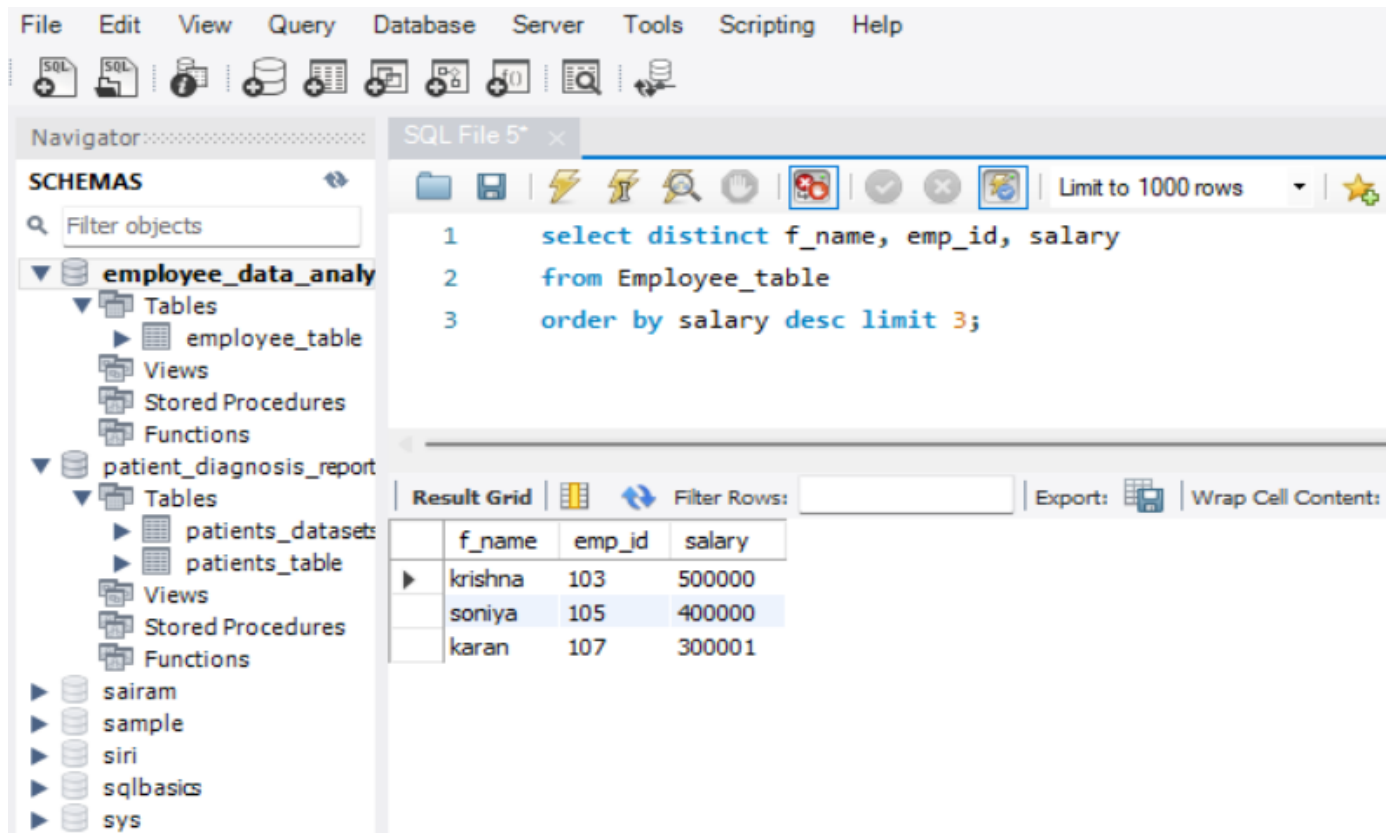
The screenshot shows the SQL Developer interface. On the left, the 'SCHEMAS' pane displays a tree view with 'employee\_data\_analy' expanded, showing 'employee\_table'. The main editor window, titled 'SQL File 5\*', contains the following SQL query:

```
1 • SELECT emp_id, f_name, salary from Employee_table
2   where salary > ALL(SELECT Salary from Employee_table where
3     (job_id = 'HP122'))
4   order by salary asc, f_name, emp_id;
```

Below the query editor, the 'Result Grid' tab is active, displaying the results of the query in a table:

emp_id	f_name	salary
106	nithin	300000
107	karan	300001
109	mukesh	300001
108	shilpa	300001
105	soniya	400000
103	krishna	500000

6. Write a query to display the **first name**, **employee id**, and **salary** of the first three employees with **highest salaries**.



The screenshot shows the SQL Developer interface. The left pane displays the 'SCHEMAS' tree with 'employee\_data\_analy' expanded, showing 'employee\_table'. The central pane contains the following SQL query:

```
1 select distinct f_name, emp_id, salary
2 from Employee_table
3 order by salary desc limit 3;
```

The right pane shows the 'Result Grid' with the following data:

	f_name	emp_id	salary
▶	krishna	103	500000
	soniya	105	400000
	karan	107	300001