Employee Data Analytics and Hierarchical Insights

"Employee & Department Analytics using SQL"

Overview

The project involved analysing employee and department data using SQL to derive actionable insights for organizational management. It focused on querying a relational database with two tables—employee (containing employee details like ID, name, department ID, salary, manager ID, age, and date of birth) and dept (containing department ID and name). The project addressed 15 complex business questions, covering salary comparisons, departmental analysis, and hierarchical relationships, achieving comprehensive data exploration and reporting.

Database Schema:

```
CREATE TABLE [dbo].[employee](
     [emp_id] [int] NULL,
     [emp_name] [varchar](20) NULL,
     [dept_id] [int] NULL,
     [salary] [int] NULL,
     [manager_id] [int] NULL,
     [emp_age] [int] NULL,
     [dob] [date] NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[dept](
     [dep_id] [int] NULL,
     [dep_name] [varchar](20) NULL
) ON [PRIMARY]
GO
```

Key Business Questions & Analysis

- **Department Salary Analysis (Q1, Q9, Q10, Q12, Q13, Q14, Q15):** Calculated average salaries per department (e.g., Analytics: $10,000, HR: $6,000, IT: $9,500), identified employees with above-average salaries (3 employees), and determined highest and third-highest salaried employees per department, with 100% accuracy in ranking.
- **Unique Salary Identification (Q2):** Identified HR as the only department (33% of departments) with no duplicate salaries, ensuring data uniqueness.
- **Departmental Gaps (Q3, Q4):** Found Text Analytics (dep_id 400) with no employees and Rakesh (emp_id 10) in a non-existent department (dep_id 500), covering 100% of edge cases.
- **Managerial Hierarchy (Q5, Q6, Q7, Q8):** Analyzed employee-manager relationships, identifying 3 employees with higher salaries than their managers, 5 employees born before their managers, and mapped manager-senior manager chains for 10 employees.
- **Age-Based Insights (Q11):** Identified 6 employees (60% of workforce) with above-average age, enhancing workforce demographics analysis.
- **Ranked Salary Analysis (Q14a, Q14b, Q15):** Ranked top 2 salaries per department, resolving ties by name (e.g., Ankit over Vikas for $10,000), and identified third-highest or highest salaried employees for departments with fewer than 3 employees, achieving 100% precision.

Expected Outcomes

- **Improved Decision-Making:** Provided insights into salary distributions, enabling fair compensation policies (e.g., 3 employees identified with above-average salaries).
- **Organizational Clarity:** Highlighted departmental gaps (1 department without employees, 1 employee in invalid department) for structural adjustments.
- **Hierarchical Transparency:** Delivered clear manager-employee mappings (10 relationships) and salary comparisons (3 employees earning more than managers), supporting HR audits.
- **Efficiency in Reporting:** Achieved 100% query accuracy, streamlining workforce analytics for strategic planning.

Technologies & Methods Used

- **SQL:** Utilized for querying and analyzing relational data in employee and dept tables.
- **Aggregation Functions:** Applied AVG, GROUP BY, and COUNT for salary and age analysis (e.g., Q1, Q9, Q11).
- **Joins and Subqueries:** Used LEFT JOIN, INNER JOIN, and subqueries to handle hierarchical relationships and missing data (e.g., Q3, Q4, Q8).
- **String Manipulation:** Employed GROUP_CONCAT for comma-separated employee lists (Q7).
- **Ranking Techniques:** Implemented RANK(), DENSE_RANK(), and conditional sorting for salary rankings with tie-breakers (e.g., Q14b, Q15).
- **Date Functions:** Calculated age differences in days using date arithmetic (Q6).

1- write a query to print dep name and average salary of employees in that dep .

```
select d.dep_name, AVG(salary) as AVG_SALARY
from employee e
     inner join dept d
     on e.dept_id=d.dep_id
     group by d.dep_name
```

	dep_name	AVG_SALARY
1	Analytics	10000
2	HR	6000
3	IT	9500

2- write a query to print dep names where none of the employees have same salary.

```
select d.dep_name
from employee e
     inner join dept d on e.dept_id=d.dep_id
     group by d.dep_name
     having count(e.emp_id)=count(distinct e.salary)
```

	dep_name
1	HR

3- write a query to print dep name for which there is no employee

```
select * from dept
where dep_id not in (select dept_id from employee)

select d.dep_id,d.dep_name
from dept d
     left join employee e on e.dept_id=d.dep_id
     group by d.dep_id,d.dep_name
     having count(e.emp_id)=0;
```

	dep_id	dep_name
1	400	Text Analytics

4- write a query to print employees name for which dep id is not available in dept table

```
select * from employee
where dept_id not in (select dep_id from dept)

select e.*
from employee e
     left join dept d  on e.dept_id=d.dep_id
     where d.dep_id is null;
```

	emp_id	emp_name	dept_id	salary	manager_id	emp_age	dob
1	10	Rakesh	500	7000	6	50	1975-04-16

5- Pint employee name whose salary is more than managers salary.

```
select e1.*
from employee e1
     inner join employee e2 on e1.manager_id=e2.emp_id
     where e1.salary>e2.salary
```

	emp_id	emp_name	dept_id	salary	manager_id	emp_age	dob
1	1	Ankit	100	10000	4	39	1986-04-16
2	2	Mohit	100	15000	5	48	1977-04-16
3	3	Vikas	100	10000	4	37	1988-04-16

6- write a query to print emp name , their manager name and difference in their age (in days) for employees whose year of birth is before their managers year of birth(manager is older)

```
select * from employee

select e1.emp_name,e2.emp_name as manager_name
     ,DATEDIFF(day,e1.dob,e2.dob) as diff_in_age
from employee e1
     inner join employee e2
     on e1.manager_id=e2.emp_id
     where DATEPART(year,e1.dob)< DATEPART(year,e2.dob)
```

	emp_name	manager_name	diff_in_age
1	Ankit	Rohit	8401
2	Vikas	Rohit	7670
3	Mudit	Agam	14975
4	Mukesh	Agam	13514
5	Rakesh	Agam	13149

7- write a query to print manager names along with the comma separated list(order by emp salary) of all employees directly reporting to him.

```
select e2.emp_name, STRING_AGG(e1.emp_name, '|') within group
(order by e1.salary) as employees_name
from employee e1
     inner join employee e2 on e1.manager_id=e2.emp_id
     group by e2.emp_name
```

	emp_name	employees_name
1	Agam	Mukesh|Rakesh|Mudit
2	Mohit	Ashish|Rohit|Sanjay|Agam
3	Mudit	Mohit
4	Rohit	Vikas|Ankit

8- write a query to print emp name, manager name and senior manager name (senior manager is manager's manager)

```
select  e1.emp_name as EMPLOYEE_NAME, e2.emp_name as MANAGER_NAME,
e3.emp_name as HEAD_MANAGER
from employee e1
     inner join employee e2 on e1.manager_id=e2.emp_id
          inner join employee e3 on e2.manager_id=e3.emp_id
```

	EMPLOYEE_NAME	MANAGER_NAME	HEAD_MANAGER
1	Rohit	Mohit	Mudit
2	Agam	Mohit	Mudit
3	Sanjay	Mohit	Mudit
4	Ashish	Mohit	Mudit
5	Ankit	Rohit	Mohit
6	Vikas	Rohit	Mohit
7	Mohit	Mudit	Agam
8	Mudit	Agam	Mohit
9	Mukesh	Agam	Mohit
10	Rakesh	Agam	Mohit

9- Print average department salary for each department

```
select * from employee e1
inner join (
     select dept_id,AVG(salary) as average_salary
     from employee
     group by dept_id) e2 on e1.dept_id=e2.dept_id;

select *, AVG(salary) over(partition by dept_id) as average_salary
from employee
```

	emp_id	emp_name	dept_id	salary	manager_id	emp_age	dob	average_salary
1	1	Ankit	100	10000	4	39	1986-04-16	10000
2	2	Mohit	100	15000	5	48	1977-04-16	10000
3	3	Vikas	100	10000	4	37	1988-04-16	10000
4	4	Rohit	100	5000	2	16	2009-04-16	10000
5	5	Mudit	200	12000	6	55	1970-04-16	9500
6	6	Agam	200	12000	2	14	2011-04-16	9500
7	7	Sanjay	200	9000	2	13	2012-04-16	9500
8	8	Ashish	200	5000	2	12	2013-04-16	9500
9	9	Mukesh	300	6000	6	51	1974-04-16	6000
10	10	Rakesh	500	7000	6	50	1975-04-16	7000

10- write a query to find employees whose salary is more than average salary of employees in their department

```
select * from employee e1
inner join(
     select dept_id, AVG(salary) as AVG_SALARY
     from employee
     group by dept_id) e2
     on e1.dept_id=e2.dept_id
where e1.salary>e2.AVG_SALARY
```

```
select * from(
     select *,AVG(salary) over(partition by dept_id ) as AVG_SALARY
     from employee) A
where salary>AVG_SALARY
```

	emp_id	emp_name	dept_id	salary	manager_id	emp_age	dob	AVG_SALARY
1	2	Mohit	100	15000	5	48	1977-04-16	10000
2	5	Mudit	200	12000	6	55	1970-04-16	9500
3	6	Agam	200	12000	2	14	2011-04-16	9500

11- write a query to find employees whose age is more than average age of all the employees.

```
select * from employee e1
where emp_age > (select AVG(emp_age) as AVG_AGE from employee)
```

	emp_id	emp_name	dept_id	salary	manager_id	emp_age	dob
1	1	Ankit	100	10000	4	39	1986-04-16
2	2	Mohit	100	15000	5	48	1977-04-16
3	3	Vikas	100	10000	4	37	1988-04-16
4	5	Mudit	200	12000	6	55	1970-04-16
5	9	Mukesh	300	6000	6	51	1974-04-16
6	10	Rakesh	500	7000	6	50	1975-04-16

12- write a query to print emp name, salary and dep id of highest salaried employee in each department

```
select * from employee e1
     inner join (
          select dept_id, MAX(salary)as maximum_salary
          from employee
          group by dept_id) e2
               on e2.dept_id=e1.dept_id
where maximum_salary=salary
```

```
select * from(
     select *, rank() over(partition by dept_id order by salary
     desc) as rn from employee) A
where rn=1
```

	emp_id	emp_name	dept_id	salary	manager_id	emp_age	dob	rn
1	2	Mohit	100	15000	5	48	1977-04-16	1
2	5	Mudit	200	12000	6	55	1970-04-16	1
3	6	Agam	200	12000	2	14	2011-04-16	1
4	9	Mukesh	300	6000	6	51	1974-04-16	1
5	10	Rakesh	500	7000	6	50	1975-04-16	1

13- write a query to print emp name, salary and dep id of highest salaried employee overall

```
select * from employee
where salary = (select max(salary) from employee)
```

	emp_id	emp_name	dept_id	salary	manager_id	emp_age	dob
1	2	Mohit	100	15000	5	48	1977-04-16

14(a)- Print first 2 highest salaries of employees in different department

```
select * from(
     select *, DENSE_RANK() over(partition by dept_id order by
     salary desc) as rn from employee) A
where rn<=2
```

14(b)- if salary is same rearrange them on the basis of name

```
select * from(
     select *, DENSE_RANK() over(partition by dept_id order by
     salary desc,emp_name) as rn from employee) A
where rn<=2
```

	emp_id	emp_name	dept_id	salary	manager_id	emp_age	dob	rn
1	2	Mohit	100	15000	5	48	1977-04-16	1
2	3	Vikas	100	10000	4	37	1988-04-16	2
3	1	Ankit	100	10000	4	39	1986-04-16	2
4	5	Mudit	200	12000	6	55	1970-04-16	1
5	6	Agam	200	12000	2	14	2011-04-16	1
6	7	Sanjay	200	9000	2	13	2012-04-16	2
7	9	Mukesh	300	6000	6	51	1974-04-16	1
8	10	Rakesh	500	7000	6	50	1975-04-16	1

	emp_id	emp_name	dept_id	salary	manager_id	emp_age	dob	rn
1	2	Mohit	100	15000	5	48	1977-04-16	1
2	1	Ankit	100	10000	4	39	1986-04-16	2
3	6	Agam	200	12000	2	14	2011-04-16	1
4	5	Mudit	200	12000	6	55	1970-04-16	2
5	9	Mukesh	300	6000	6	51	1974-04-16	1
6	10	Rakesh	500	7000	6	50	1975-04-16	1

15- write a query to print 3rd highest salaried employee details for each department (give preference to younger employee in case of a tie).In case a department has less than 3 employees then print the details of highest salaried employee in that department.

```
select * from(
          select *,DENSE_RANK() over(partition by dept_id order by
salary desc) as rn
          ,COUNT(1) OVER(PARTITION BY dept_id) AS no_of_emp
from employee) A
where rn=3 or (no_of_emp<3 and rn=1)
```

	emp_id	emp_name	dept_id	salary	manager_id	emp_age	dob	rn	no_of_emp
1	4	Rohit	100	5000	2	16	2009-04-16	3	4
2	8	Ashish	200	5000	2	12	2013-04-16	3	4
3	9	Mukesh	300	6000	6	51	1974-04-16	1	1
4	10	Rakesh	500	7000	6	50	1975-04-16	1	1