# **Problem**

We have been tasked by HR Analytica to generate reusable data assets to power 2 of their client HR analytics tools.

We’ve also been asked specifically to generate database views that HR Analytica team can use for 2 key dashboards, reporting solutions and ad-hoc analytics requests.

Additionally - we’ve been alerted to the presence of date issues with our datasets where there were data-entry issues related to all DATE related fields - we will need to incorporate the fixes as we compile our solution.

## 2.1. Required Insights

The following insights must be generated for the 2 dashboards requested by HR Analytica:

### **2.1.1. Company Level**

For all following metrics - we need a current snapshot of all the data as well as a split by gender specifically:

* Total number of employees
* Average company tenure in years
* Average latest payrise percentage
* Statistical metrics for salary values including:
  + MIN, MAX, STDDEV, Inter-quartile range and median

### **2.1.2. Department Level**

All of the metrics as per the company level but at a department level, including an additional department level tenure metrics split by gender.

### **2.1.3. Title Level**

Similar to the department level metrics but instead, at a title level of granularity.

We will also need to generate similar insights for both department and title levels where we need employee count and average tenure for each level instead of company wide numbers.

## 2.2. Employee Deep Dive

The following insights must be generated for the Employee Deep Dive tool that can spotlight recent events for a single employee over time:

* See all the various employment history ordered by effective date including salary, department, manager and title changes
* Calculate previous historic payrise percentages and value changes
* Calculate the previous position and department history in months with start and end dates
* Compare an employee’s current salary, total company tenure, department, position and gender to the average benchmarks for their current position

## 2.3. Visual OutputsGraphical user interface Description automatically generated

# **3. Exploration**

In the following section we will be exploring all of the provided data as we start breaking down our approach to solve the case study problem.

## 3.1. Entity Relationship Diagram

Our project uses all available data within the employees schema and contains information about a company HR Analytica is supporting with its bespoke analytical tools.

We have been provided the entity-relationship diagram by HR Analytica for this project as shown below:

Graphical user interface, application

Description automatically generated

## 3.2. Data Exploration

Since we’ve been alerted to the presence of data issues for all date related fields - we will need to inspect each table to see what adjustments we need to make.

Additionally - we will start profiling all of our available tables to see how we can join them for our complete analytical solution.

From our initial inspection of the ERD - it also seems like there are slow changing dimension tables as we can see from\_date and to\_date columns for some tables.

Firstly let’s explore the available indexes from the employees schema before moving onto individual tables

### **3.2.1. Table Indexes**

We can query our tables and their index information by accessing the pg\_indexes table as shown below:

**SELECT** **\***

**FROM** pg\_indexes

**WHERE** schemaname **=** 'employees';

| **schemaname** | **tablename** | **indexname** | **indexdef** |
| --- | --- | --- | --- |
| employees | employee | idx\_16988\_primary | CREATE UNIQUE INDEX idx\_16988\_primary ON employees.employee USING btree (id) |
| employees | department\_employee | idx\_16982\_primary | CREATE UNIQUE INDEX idx\_16982\_primary ON employees.department\_employee USING btree (employee\_id, department\_id) |
| employees | department\_employee | idx\_16982\_dept\_no | CREATE INDEX idx\_16982\_dept\_no ON employees.department\_employee USING btree (department\_id) |
| employees | department | idx\_16979\_primary | CREATE UNIQUE INDEX idx\_16979\_primary ON employees.department USING btree (id) |
| employees | department | idx\_16979\_dept\_name | CREATE UNIQUE INDEX idx\_16979\_dept\_name ON employees.department USING btree (dept\_name) |
| employees | department\_manager | idx\_16985\_primary | CREATE UNIQUE INDEX idx\_16985\_primary ON employees.department\_manager USING btree (employee\_id, department\_id) |
| employees | department\_manager | idx\_16985\_dept\_no | CREATE INDEX idx\_16985\_dept\_no ON employees.department\_manager USING btree (department\_id) |
| employees | salary | idx\_16991\_primary | CREATE UNIQUE INDEX idx\_16991\_primary ON employees.salary USING btree (employee\_id, from\_date) |
| employees | title | idx\_16994\_primary | CREATE UNIQUE INDEX idx\_16994\_primary ON employees.title USING btree (employee\_id, title, from\_date) |

In the above output we can see that there seems to be unique indexes on most available tables.

The following tables seem to have unique indexes on a single column:

* employees.employee
* employees.department

The rest of the tables seem to have multiple records for the employee\_id values based off the indexes:

* employees.department\_employee
* employees.department\_manager
* employees.salary
* employees.title

Let’s start by analysing the single column index columns before we move onto the other tables below.

## 3.3. Individual Table Analysis

Each tab below contains the exploration information for each table. We start with the 2 unique single column index tables before the 4 remaining historical tables.

The historical tables all have from\_date and to\_date records which signal some sort of historical slow changing dimension style table.

These tables show the relationship between employee\_id and specific information for certain periods of time defined the by from\_date and to\_date with the arbitrary end date of 9999-01-01

Click on each tab below to see more information on each table.

* [3.3.1. employee](https://import.cdn.thinkific.com/357412/multimedia_imports/05-final-solution-april-28-2021-11-52/05-final-solution.html#employee)
* [3.3.2. department](https://import.cdn.thinkific.com/357412/multimedia_imports/05-final-solution-april-28-2021-11-52/05-final-solution.html#department)
* [3.3.3. department\_employee](https://import.cdn.thinkific.com/357412/multimedia_imports/05-final-solution-april-28-2021-11-52/05-final-solution.html#department_employee)
* [3.3.4. department\_manager](https://import.cdn.thinkific.com/357412/multimedia_imports/05-final-solution-april-28-2021-11-52/05-final-solution.html#department_manager)
* [3.3.5. salary](https://import.cdn.thinkific.com/357412/multimedia_imports/05-final-solution-april-28-2021-11-52/05-final-solution.html#salary)
* [3.3.6. title](https://import.cdn.thinkific.com/357412/multimedia_imports/05-final-solution-april-28-2021-11-52/05-final-solution.html#title)

Based off the ERD - it seems that the employee table seems like a good place to start our exploration as it contains the following fields of interest:

* birth\_date
* gender
* first and last name
* hire\_date

Let’s first inspect a few rows:

**SELECT** **\***

**FROM** employees.employee

**LIMIT** 5;

| **id** | **birth\_date** | **first\_name** | **last\_name** | **gender** | **hire\_date** |
| --- | --- | --- | --- | --- | --- |
| 10001 | 1953-09-02 | Georgi | Facello | M | 1986-06-26 |
| 10002 | 1964-06-02 | Bezalel | Simmel | F | 1985-11-21 |
| 10003 | 1959-12-03 | Parto | Bamford | M | 1986-08-28 |
| 10004 | 1954-05-01 | Chirstian | Koblick | M | 1986-12-01 |
| 10005 | 1955-01-21 | Kyoichi | Maliniak | M | 1989-09-12 |

Next let’s confirm that there is indeed only a single record per employee record as implied by the index:

**WITH** id\_cte **AS** (

**SELECT**

**id**,

COUNT(**\***) **AS** row\_count

**FROM** employees.employee

**GROUP** **BY** **id**

)

**SELECT**

row\_count,

COUNT(**DISTINCT** **id**) **AS** employee\_count

**FROM** id\_cte

**GROUP** **BY** row\_count

**ORDER** **BY** row\_count;

| **row\_count** | **employee\_count** |
| --- | --- |
| 1 | 300024 |

Our initial hypothesis is that not all of our employees will exist in our other tables as there should be natural employee churn for any company - let’s keep this in mind as we continue with our data exploration.

# **4. Analysis**

For our complete SQL solution we will need to split up components into the following parts:

1. Data Cleaning & Date Adjustments
2. Current Snaphsot Analysis
3. Historical Analysis

Finally we will generate the required data points for each of the sample visual outputs we’ve received from HR Analytica.

The key aspect of our entire SQL analysis will be to generate a completely reusable data asset in the form of multiple analytical views for the HR Analytica team to consume.

All of our analytical outptus will be generated in an entirely new view schema called mv\_employees which will be refered to throughout our SQL code snippets and the final complete SQL script.

Let’s first start with the data cleaning component to adjust the dates and fix the date data issues.

## 4.1. Data Cleaning

Firstly - we will need to adjust all of our relevant date fields due to the data issue identified by HR Analytica.

We will be incrementing all of the date fields except the arbitrary end date of 9999-01-01 - we will also need to cast our results back to a DATE data type as PostgreSQL interval addition forces the data type to a TIMESTAMP which we’d like to avoid to keep our data as similar to the original as possible.

To account for future updates and to maximise the efficiency and productivity for the HR Analytica team - we will be implementing our adjusted datasets as materialized views with exact original indexes as per the original tables in the employees schema.

Click here to show data cleaning materialized view code snippet

## 4.2. Current Snapshot Analysis

For our current company, department and title level dashboard outputs we will first create a current snapshot view which we will use as the base for each of the aggregated layers for the different dashboard outputs.

### **4.2.1. Analysis Plan**

Let’s start by listing out the steps we need to include for our granular current snapshot:

1. Apply LAG window functions on the salary materialized view to obtain the latest previous\_salary value, keeping only current valid records with to\_date = '9999-01-01'
2. Join previous salary and all other required information from the materialized views for the dashboard analysis (omitting the department\_manager view)
3. Apply WHERE filter to keep only current records
4. Make sure to include the gender column from the employee view for all calculations
5. Use the hire\_date column from the employee view to calculate the number of tenure years
6. Include the from\_date columns from the title and department are included to calculate tenure
7. Use the salary table to calculate the current average salary
8. Include department and title information for additional group by aggregations
9. Implement the various statistical measures for the salary amount
10. Combine all of these elements into a single final current snapshot view

Because we will need to compile the entire analysis process as a single view - we will implement each of the above steps inside a single query in the following sections.

After the current snapshot view is created - we will then implement the 3 different aggregation layers for the company, department and title level dashboard data outputs.

### **4.2.2. Current Employee Snapshot**

We will implement the first 10 components directly as a new view called mv\_employees.current\_employee\_snapshot

The complete code required to generate the view is included below:

*-- note the CASCADE option as there may be derived views downstream*

**DROP** **VIEW** **IF** **EXISTS** mv\_employees.current\_employee\_snapshot **CASCADE**;

**CREATE** **VIEW** mv\_employees.current\_employee\_snapshot **AS**

*-- apply LAG to get previous salary amount for all employees*

**WITH** cte\_previous\_salary **AS** (

**SELECT** **\*** **FROM** (

**SELECT**

employee\_id,

to\_date,

LAG(amount) **OVER** (

**PARTITION** **BY** employee\_id

**ORDER** **BY** from\_date

) **AS** amount

**FROM** mv\_employees.salary

) all\_salaries

*-- keep only latest valid previous\_salary records only*

*-- must have this in subquery to account for execution order*

**WHERE** to\_date **=** '9999-01-01'

),

*-- combine all elements into a joined CTE*

cte\_joined\_data **AS** (

**SELECT**

employee.**id** **AS** employee\_id,

employee.gender,

employee.hire\_date,

title.title,

salary.amount **AS** salary,

cte\_previous\_salary.amount **AS** previous\_salary,

department.dept\_name **AS** department,

*-- need to keep the title and department from\_date columns for tenure*

title.from\_date **AS** title\_from\_date,

department\_employee.from\_date **AS** department\_from\_date

**FROM** mv\_employees.employee

**INNER** **JOIN** mv\_employees.title

**ON** employee.**id** **=** title.employee\_id

**INNER** **JOIN** mv\_employees.salary

**ON** employee.**id** **=** salary.employee\_id

*-- join onto the CTE we created in the first step*

**INNER** **JOIN** cte\_previous\_salary

**ON** employee.**id** **=** cte\_previous\_salary.employee\_id

**INNER** **JOIN** mv\_employees.department\_employee

**ON** employee.**id** **=** department\_employee.employee\_id

*-- NOTE: department is joined only to the department\_employee table!*

**INNER** **JOIN** mv\_employees.department

**ON** department\_employee.department\_id **=** department.**id**

*-- apply where filter to keep only relevant records*

**WHERE** salary.to\_date **=** '9999-01-01'

**AND** title.to\_date **=** '9999-01-01'

**AND** department\_employee.to\_date **=** '9999-01-01'

),

*-- finally we can apply all our calculations in this final output*

final\_output **AS** (

**SELECT**

employee\_id,

gender,

title,

salary,

department,

*-- salary change percentage*

ROUND(

100 **\*** (salary **-** previous\_salary) **/** previous\_salary::NUMERIC,

2

) **AS** salary\_percentage\_change,

*-- tenure calculations*

DATE\_PART('year', now()) **-**

DATE\_PART('year', hire\_date) **AS** company\_tenure\_years,

DATE\_PART('year', now()) **-**

DATE\_PART('year', title\_from\_date) **AS** title\_tenure\_years,

DATE\_PART('year', now()) **-**

DATE\_PART('year', department\_from\_date) **AS** department\_tenure\_years

**FROM** cte\_joined\_data

)

**SELECT** **\*** **FROM** final\_output;

Click here to see sample rows from mv\_employees.current\_employee\_snapshot

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

### **4.2.3. Dashboard Aggregation Views**

The next step is to perform different levels of aggregations to generate the required data outputs for each of the company, department and title level dashboards.

#### **4.2.3.1. Company Level**

*-- company level aggregation view*

**DROP** **VIEW** **IF** **EXISTS** mv\_employees.company\_level\_dashboard;

**CREATE** **VIEW** mv\_employees.company\_level\_dashboard **AS**

**SELECT**

gender,

COUNT(**\***) **AS** employee\_count,

ROUND(100 **\*** COUNT(**\***)::NUMERIC **/** SUM(COUNT(**\***)) **OVER** ()) **AS** employee\_percentage,

ROUND(AVG(company\_tenure\_years)) **AS** company\_tenure,

ROUND(AVG(salary)) **AS** avg\_salary,

ROUND(AVG(salary\_percentage\_change)) **AS** avg\_salary\_percentage\_change,

*-- salary statistics*

ROUND(MIN(salary)) **AS** min\_salary,

ROUND(MAX(salary)) **AS** max\_salary,

ROUND(PERCENTILE\_CONT(0.5) WITHIN **GROUP** (**ORDER** **BY** salary)) **AS** median\_salary,

ROUND(

PERCENTILE\_CONT(0.75) WITHIN **GROUP** (**ORDER** **BY** salary) **-**

PERCENTILE\_CONT(0.25) WITHIN **GROUP** (**ORDER** **BY** salary)

) **AS** inter\_quartile\_range,

ROUND(STDDEV(salary)) **AS** stddev\_salary

**FROM** mv\_employees.current\_employee\_snapshot

**GROUP** **BY** gender;

Click here to see sample rows from mv\_employees.company\_level\_dashboard

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |

#### **4.2.3.2. Department Level**

*-- department level aggregation view*

**DROP** **VIEW** **IF** **EXISTS** mv\_employees.department\_level\_dashboard;

**CREATE** **VIEW** mv\_employees.department\_level\_dashboard **AS**

**SELECT**

gender,

department,

COUNT(**\***) **AS** employee\_count,

ROUND(100 **\*** COUNT(**\***)::NUMERIC **/** SUM(COUNT(**\***)) **OVER** (

**PARTITION** **BY** department

)) **AS** employee\_percentage,

ROUND(AVG(department\_tenure\_years)) **AS** department\_tenure,

ROUND(AVG(salary)) **AS** avg\_salary,

ROUND(AVG(salary\_percentage\_change)) **AS** avg\_salary\_percentage\_change,

*-- salary statistics*

ROUND(MIN(salary)) **AS** min\_salary,

ROUND(MAX(salary)) **AS** max\_salary,

ROUND(PERCENTILE\_CONT(0.5) WITHIN **GROUP** (**ORDER** **BY** salary)) **AS** median\_salary,

ROUND(

PERCENTILE\_CONT(0.75) WITHIN **GROUP** (**ORDER** **BY** salary) **-**

PERCENTILE\_CONT(0.25) WITHIN **GROUP** (**ORDER** **BY** salary)

) **AS** inter\_quartile\_range,

ROUND(STDDEV(salary)) **AS** stddev\_salary

**FROM** mv\_employees.current\_employee\_snapshot

**GROUP** **BY**

gender, department;

Click here to see sample rows from mv\_employees.department\_level\_dashboard

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |

#### **4.2.3.3. Title Level**

*-- title level aggregation view*

**DROP** **VIEW** **IF** **EXISTS** mv\_employees.title\_level\_dashboard;

**CREATE** **VIEW** mv\_employees.title\_level\_dashboard **AS**

**SELECT**

gender,

title,

COUNT(**\***) **AS** employee\_count,

ROUND(100 **\*** COUNT(**\***)::NUMERIC **/** SUM(COUNT(**\***)) **OVER** (

**PARTITION** **BY** title

)) **AS** employee\_percentage,

ROUND(AVG(title\_tenure\_years)) **AS** title\_tenure,

ROUND(AVG(salary)) **AS** avg\_salary,

ROUND(AVG(salary\_percentage\_change)) **AS** avg\_salary\_percentage\_change,

*-- salary statistics*

ROUND(MIN(salary)) **AS** min\_salary,

ROUND(MAX(salary)) **AS** max\_salary,

ROUND(PERCENTILE\_CONT(0.5) WITHIN **GROUP** (**ORDER** **BY** salary)) **AS** median\_salary,

ROUND(

PERCENTILE\_CONT(0.75) WITHIN **GROUP** (**ORDER** **BY** salary) **-**

PERCENTILE\_CONT(0.25) WITHIN **GROUP** (**ORDER** **BY** salary)

) **AS** inter\_quartile\_range,

ROUND(STDDEV(salary)) **AS** stddev\_salary

**FROM** mv\_employees.current\_employee\_snapshot

**GROUP** **BY**

gender, title;

Click here to see sample rows from mv\_employees.title\_level\_dashboard

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |

## 4.3. Historic Employee Analysis

For the historic employee deep dive analysis - we will need to split up our interim outputs into 3 parts:

**1. Current Employee Information**

* Full name
* Gender
* Birthday
* Department
* Title/Position tenure
* Company tenure
* Current salary
* Latest salary change percentage
* Manager name

**2. Salary comparison to various benchmarks including:**

* Company tenure
* Title/Position
* Department
* Gender

**3. The last 5 historical employee events categorised into:**

* Salary increase/decrease
* Department transfer
* Manager reporting line change
* Title changes

### **4.3.1. Analysis Plan**

**1. Current Employee Information**

The current employee information is very similar to our previous mv\_employees.current\_employee\_snapshot view we created - but with the addition of the manager information and the full name and birthday details.

For this step we will try to replace our previously created view to incorporate this new information.

By replacing the view using a CREATE OR REPLACE VIEW statement - we can avoid dropping our previously created derived views generated for each of the dashboards - saving us a bit of time!

In the final version of our complete SQL script we will be generating all of the views from scratch - but this is something useful to keep in mind when updating views, the only thing to take note of is that the column orders must be identical for the existing columns in the original view and any new columns must be placed afterwards.

**2. Salary Comparisons**

We will generate separate views based off the new updated mv\_employees.current\_employee\_snapshot to generate baseline bencmarks for us to compare to our spotlighted eemployee.

**3. Employee Events**

We will need to generate a new historical view that incorporates all of the materialized views for our current snapshot - however we will further extend this by implementing additional effective\_date and expiry\_date columns to correctly capture the most accurate validitiy period for each record.

We will need to discard any invalid records which result from the join after applying the GREATEST and LEAST logic on our date fields instead of simply applying a WHERE filter to keep only the current records like we implemented for the current snapshot views.

We will also need to classify our events. To do this - we will use a CASE WHEN statement to accurately identify the types of events by comparing the salary, title, department and manager details with their LAG equivalent to check for changes.

We can also apply a ROW\_NUMBER window function to keep only the latest 5 records by most recent effective\_date

**An important note on the employees who are no longer with the company:**

Since we will have employee records present in our view who are no longer employed by the company - we can only compare their latest salary amount before leaving the company to the current benchmarks.

To simplify our SQL script - we will include a benchmark comparison for all employee salary records to the current benchmarks only for simplicity.

### **4.3.2. Challenge Exercise**

This is quite a challenging task - if you were to refactor the existing SQL scripts to include a benchmark comparison at every single point in time - how would you do this using the existing data?

### **4.3.3. SQL Solution**

The script for this historical employee component is quite long.

Click here to see entire SQL solution

# **5. Report**

Finally let’s summarise the results of our analysis and compile a complete end to end SQL script we can use to recreate our entire workflow.

We will also regenerate the example data points provided for our visual examples for both the current and historic analysis components.

Finally there will be a few questions for the final case study quiz after the following outputs.

## 5.1. Final SQL Script

The script is broken into 3 sections:

1. Create materialized views to fix date data issues
2. Current employee snapshot view
3. Aggregated dashboard views
4. Salary benchmark views
5. Historic employee deep dive view
6. */\*---------------------------------------------------*
7. *1. Create materialized views to fix date data issues*
8. *----------------------------------------------------\*/*
9. **DROP** **SCHEMA** **IF** **EXISTS** mv\_employees **CASCADE**;
10. **CREATE** **SCHEMA** mv\_employees;
11. *-- department*
12. **DROP** **MATERIALIZED** **VIEW** **IF** **EXISTS** mv\_employees.department;
13. **CREATE** **MATERIALIZED** **VIEW** mv\_employees.department **AS**
14. **SELECT** **\*** **FROM** employees.department;
15. *-- department employee*
16. **DROP** **MATERIALIZED** **VIEW** **IF** **EXISTS** mv\_employees.department\_employee;
17. **CREATE** **MATERIALIZED** **VIEW** mv\_employees.department\_employee **AS**
18. **SELECT**
19. employee\_id,
20. department\_id,
21. (from\_date **+** interval '18 years')::DATE **AS** from\_date,
22. **CASE**
23. **WHEN** to\_date **<>** '9999-01-01' **THEN** (to\_date **+** interval '18 years')::DATE
24. **ELSE** to\_date
25. **END** **AS** to\_date
26. **FROM** employees.department\_employee;
27. *-- department manager*
28. **DROP** **MATERIALIZED** **VIEW** **IF** **EXISTS** mv\_employees.department\_manager;
29. **CREATE** **MATERIALIZED** **VIEW** mv\_employees.department\_manager **AS**
30. **SELECT**
31. employee\_id,
32. department\_id,
33. (from\_date **+** interval '18 years')::DATE **AS** from\_date,
34. **CASE**
35. **WHEN** to\_date **<>** '9999-01-01' **THEN** (to\_date **+** interval '18 years')::DATE
36. **ELSE** to\_date
37. **END** **AS** to\_date
38. **FROM** employees.department\_manager;
39. *-- employee*
40. **DROP** **MATERIALIZED** **VIEW** **IF** **EXISTS** mv\_employees.employee;
41. **CREATE** **MATERIALIZED** **VIEW** mv\_employees.employee **AS**
42. **SELECT**
43. **id**,
44. (birth\_date **+** interval '18 years')::DATE **AS** birth\_date,
45. first\_name,
46. last\_name,
47. gender,
48. (hire\_date **+** interval '18 years')::DATE **AS** hire\_date
49. **FROM** employees.employee;
50. *-- salary*
51. **DROP** **MATERIALIZED** **VIEW** **IF** **EXISTS** mv\_employees.salary;
52. **CREATE** **MATERIALIZED** **VIEW** mv\_employees.salary **AS**
53. **SELECT**
54. employee\_id,
55. amount,
56. (from\_date **+** interval '18 years')::DATE **AS** from\_date,
57. **CASE**
58. **WHEN** to\_date **<>** '9999-01-01' **THEN** (to\_date **+** interval '18 years')::DATE
59. **ELSE** to\_date
60. **END** **AS** to\_date
61. **FROM** employees.salary;
62. *-- title*
63. **DROP** **MATERIALIZED** **VIEW** **IF** **EXISTS** mv\_employees.title;
64. **CREATE** **MATERIALIZED** **VIEW** mv\_employees.title **AS**
65. **SELECT**
66. employee\_id,
67. title,
68. (from\_date **+** interval '18 years')::DATE **AS** from\_date,
69. **CASE**
70. **WHEN** to\_date **<>** '9999-01-01' **THEN** (to\_date **+** interval '18 years')::DATE
71. **ELSE** to\_date
72. **END** **AS** to\_date
73. **FROM** employees.title;
74. *-- Index Creation*
75. **CREATE** **UNIQUE** **INDEX** **ON** mv\_employees.employee **USING** btree (**id**);
76. **CREATE** **UNIQUE** **INDEX** **ON** mv\_employees.department\_employee **USING** btree (employee\_id, department\_id);
77. **CREATE** **INDEX** **ON** mv\_employees.department\_employee **USING** btree (department\_id);
78. **CREATE** **UNIQUE** **INDEX** **ON** mv\_employees.department **USING** btree (**id**);
79. **CREATE** **UNIQUE** **INDEX** **ON** mv\_employees.department **USING** btree (dept\_name);
80. **CREATE** **UNIQUE** **INDEX** **ON** mv\_employees.department\_manager **USING** btree (employee\_id, department\_id);
81. **CREATE** **INDEX** **ON** mv\_employees.department\_manager **USING** btree (department\_id);
82. **CREATE** **UNIQUE** **INDEX** **ON** mv\_employees.salary **USING** btree (employee\_id, from\_date);
83. **CREATE** **UNIQUE** **INDEX** **ON** mv\_employees.title **USING** btree (employee\_id, title, from\_date);
84. */\*-----------------------------------*
85. *2. Current employee snapshot view*
86. *-------------------------------------\*/*
87. **DROP** **VIEW** **IF** **EXISTS** mv\_employees.current\_employee\_snapshot;
88. **CREATE** **VIEW** mv\_employees.current\_employee\_snapshot **AS**
89. *-- apply LAG to get previous salary amount for all employees*
90. **WITH** cte\_previous\_salary **AS** (
91. **SELECT** **\*** **FROM** (
92. **SELECT**
93. employee\_id,
94. to\_date,
95. LAG(amount) **OVER** (
96. **PARTITION** **BY** employee\_id
97. **ORDER** **BY** from\_date
98. ) **AS** amount
99. **FROM** mv\_employees.salary
100. ) all\_salaries
101. *-- keep only latest valid previous\_salary records only*
102. *-- must have this in subquery to account for execution order*
103. **WHERE** to\_date **=** '9999-01-01'
104. ),
105. *-- combine all elements into a joined CTE*
106. cte\_joined\_data **AS** (
107. **SELECT**
108. employee.**id** **AS** employee\_id,
109. *-- include employee full name*
110. CONCAT\_WS(' ', employee.first\_name, employee.last\_name) **AS** employee\_name,
111. employee.gender,
112. employee.hire\_date,
113. title.title,
114. salary.amount **AS** salary,
115. cte\_previous\_salary.amount **AS** previous\_salary,
116. department.dept\_name **AS** department,
117. *-- include manager full name*
118. CONCAT\_WS(' ', manager.first\_name, manager.last\_name) **AS** manager,
119. *-- need to keep the title and department from\_date columns for tenure calcs*
120. title.from\_date **AS** title\_from\_date,
121. department\_employee.from\_date **AS** department\_from\_date
122. **FROM** mv\_employees.employee
123. **INNER** **JOIN** mv\_employees.title
124. **ON** employee.**id** **=** title.employee\_id
125. **INNER** **JOIN** mv\_employees.salary
126. **ON** employee.**id** **=** salary.employee\_id
127. *-- join onto the CTE we created in the first step*
128. **INNER** **JOIN** cte\_previous\_salary
129. **ON** employee.**id** **=** cte\_previous\_salary.employee\_id
130. **INNER** **JOIN** mv\_employees.department\_employee
131. **ON** employee.**id** **=** department\_employee.employee\_id
132. *-- NOTE: department is joined only to the department\_employee table!*
133. **INNER** **JOIN** mv\_employees.department
134. **ON** department\_employee.department\_id **=** department.**id**
135. *-- add in the department\_manager information onto the department table*
136. **INNER** **JOIN** mv\_employees.department\_manager
137. **ON** department.**id** **=** department\_manager.department\_id
138. *-- join again on the employee\_id field to another employee for manager's info*
139. **INNER** **JOIN** mv\_employees.employee **AS** manager
140. **ON** department\_manager.employee\_id **=** manager.**id**
141. *-- apply where filter to keep only relevant records*
142. **WHERE** salary.to\_date **=** '9999-01-01'
143. **AND** title.to\_date **=** '9999-01-01'
144. **AND** department\_employee.to\_date **=** '9999-01-01'
145. *-- add in department\_manager to\_date column filter*
146. **AND** department\_manager.to\_date **=** '9999-01-01'
147. )
148. *-- finally we can apply all our calculations in this final output*
149. **SELECT**
150. employee\_id,
151. employee\_name,
152. manager,
153. gender,
154. title,
155. salary,
156. department,
157. *-- salary change percentage*
158. ROUND(
159. 100 **\*** (salary **-** previous\_salary) **/** previous\_salary::NUMERIC,
160. 2
161. ) **AS** salary\_percentage\_change,
162. *-- tenure calculations*
163. DATE\_PART('year', now()) **-**
164. DATE\_PART('year', hire\_date) **AS** company\_tenure\_years,
165. DATE\_PART('year', now()) **-**
166. DATE\_PART('year', title\_from\_date) **AS** title\_tenure\_years,
167. DATE\_PART('year', now()) **-**
168. DATE\_PART('year', department\_from\_date) **AS** department\_tenure\_years
169. **FROM** cte\_joined\_data;
170. */\*---------------------------*
171. *3. Aggregated dashboard views*
172. *-----------------------------\*/*
173. *-- company level aggregation view*
174. **DROP** **VIEW** **IF** **EXISTS** mv\_employees.company\_level\_dashboard;
175. **CREATE** **VIEW** mv\_employees.company\_level\_dashboard **AS**
176. **SELECT**
177. gender,
178. COUNT(**\***) **AS** employee\_count,
179. ROUND(100 **\*** COUNT(**\***)::NUMERIC **/** SUM(COUNT(**\***)) **OVER** ()) **AS** employee\_percentage,
180. ROUND(AVG(company\_tenure\_years)) **AS** company\_tenure,
181. ROUND(AVG(salary)) **AS** avg\_salary,
182. ROUND(AVG(salary\_percentage\_change)) **AS** avg\_salary\_percentage\_change,
183. *-- salary statistics*
184. ROUND(MIN(salary)) **AS** min\_salary,
185. ROUND(MAX(salary)) **AS** max\_salary,
186. ROUND(PERCENTILE\_CONT(0.5) WITHIN **GROUP** (**ORDER** **BY** salary)) **AS** median\_salary,
187. ROUND(
188. PERCENTILE\_CONT(0.75) WITHIN **GROUP** (**ORDER** **BY** salary) **-**
189. PERCENTILE\_CONT(0.25) WITHIN **GROUP** (**ORDER** **BY** salary)
190. ) **AS** inter\_quartile\_range,
191. ROUND(STDDEV(salary)) **AS** stddev\_salary
192. **FROM** mv\_employees.current\_employee\_snapshot
193. **GROUP** **BY** gender;
194. *-- department level aggregation view*
195. **DROP** **VIEW** **IF** **EXISTS** mv\_employees.department\_level\_dashboard;
196. **CREATE** **VIEW** mv\_employees.department\_level\_dashboard **AS**
197. **SELECT**
198. gender,
199. department,
200. COUNT(**\***) **AS** employee\_count,
201. ROUND(100 **\*** COUNT(**\***)::NUMERIC **/** SUM(COUNT(**\***)) **OVER** (
202. **PARTITION** **BY** department
203. )) **AS** employee\_percentage,
204. ROUND(AVG(department\_tenure\_years)) **AS** department\_tenure,
205. ROUND(AVG(salary)) **AS** avg\_salary,
206. ROUND(AVG(salary\_percentage\_change)) **AS** avg\_salary\_percentage\_change,
207. *-- salary statistics*
208. ROUND(MIN(salary)) **AS** min\_salary,
209. ROUND(MAX(salary)) **AS** max\_salary,
210. ROUND(PERCENTILE\_CONT(0.5) WITHIN **GROUP** (**ORDER** **BY** salary)) **AS** median\_salary,
211. ROUND(
212. PERCENTILE\_CONT(0.75) WITHIN **GROUP** (**ORDER** **BY** salary) **-**
213. PERCENTILE\_CONT(0.25) WITHIN **GROUP** (**ORDER** **BY** salary)
214. ) **AS** inter\_quartile\_range,
215. ROUND(STDDEV(salary)) **AS** stddev\_salary
216. **FROM** mv\_employees.current\_employee\_snapshot
217. **GROUP** **BY**
218. gender, department;
219. *-- title level aggregation view*
220. **DROP** **VIEW** **IF** **EXISTS** mv\_employees.title\_level\_dashboard;
221. **CREATE** **VIEW** mv\_employees.title\_level\_dashboard **AS**
222. **SELECT**
223. gender,
224. title,
225. COUNT(**\***) **AS** employee\_count,
226. ROUND(100 **\*** COUNT(**\***)::NUMERIC **/** SUM(COUNT(**\***)) **OVER** (
227. **PARTITION** **BY** title
228. )) **AS** employee\_percentage,
229. ROUND(AVG(title\_tenure\_years)) **AS** title\_tenure,
230. ROUND(AVG(salary)) **AS** avg\_salary,
231. ROUND(AVG(salary\_percentage\_change)) **AS** avg\_salary\_percentage\_change,
232. *-- salary statistics*
233. ROUND(MIN(salary)) **AS** min\_salary,
234. ROUND(MAX(salary)) **AS** max\_salary,
235. ROUND(PERCENTILE\_CONT(0.5) WITHIN **GROUP** (**ORDER** **BY** salary)) **AS** median\_salary,
236. ROUND(
237. PERCENTILE\_CONT(0.75) WITHIN **GROUP** (**ORDER** **BY** salary) **-**
238. PERCENTILE\_CONT(0.25) WITHIN **GROUP** (**ORDER** **BY** salary)
239. ) **AS** inter\_quartile\_range,
240. ROUND(STDDEV(salary)) **AS** stddev\_salary
241. **FROM** mv\_employees.current\_employee\_snapshot
242. **GROUP** **BY**
243. gender, title;
244. */\*-----------------------*
245. *4. Salary Benchmark Views*
246. *-------------------------\*/*
247. *-- Note the slightly verbose column names - this helps us avoid renaming later!*
248. **DROP** **VIEW** **IF** **EXISTS** mv\_employees.tenure\_benchmark;
249. **CREATE** **VIEW** mv\_employees.tenure\_benchmark **AS**
250. **SELECT**
251. company\_tenure\_years,
252. AVG(salary) **AS** tenure\_benchmark\_salary
253. **FROM** mv\_employees.current\_employee\_snapshot
254. **GROUP** **BY** company\_tenure\_years;
255. **DROP** **VIEW** **IF** **EXISTS** mv\_employees.gender\_benchmark;
256. **CREATE** **VIEW** mv\_employees.gender\_benchmark **AS**
257. **SELECT**
258. gender,
259. AVG(salary) **AS** gender\_benchmark\_salary
260. **FROM** mv\_employees.current\_employee\_snapshot
261. **GROUP** **BY** gender;
262. **DROP** **VIEW** **IF** **EXISTS** mv\_employees.department\_benchmark;
263. **CREATE** **VIEW** mv\_employees.department\_benchmark **AS**
264. **SELECT**
265. department,
266. AVG(salary) **AS** department\_benchmark\_salary
267. **FROM** mv\_employees.current\_employee\_snapshot
268. **GROUP** **BY** department;
269. **DROP** **VIEW** **IF** **EXISTS** mv\_employees.title\_benchmark;
270. **CREATE** **VIEW** mv\_employees.title\_benchmark **AS**
271. **SELECT**
272. title,
273. AVG(salary) **AS** title\_benchmark\_salary
274. **FROM** mv\_employees.current\_employee\_snapshot
275. **GROUP** **BY** title;
276. */\*----------------------------------*
277. *5. Historic Employee Deep Dive View*
278. *-----------------------------------\*/*
279. *-- drop cascade required as there is 1 derived view!*
280. **DROP** **VIEW** **IF** **EXISTS** mv\_employees.historic\_employee\_records **CASCADE**;
281. **CREATE** **VIEW** mv\_employees.historic\_employee\_records **AS**
282. *-- we need the previous salary only for the latest record*
283. *-- other salary increase/decrease events will use a different field!*
284. **WITH** cte\_previous\_salary **AS** (
285. **SELECT**
286. employee\_id,
287. amount
288. **FROM** (
289. **SELECT**
290. employee\_id,
291. to\_date,
292. LAG(amount) **OVER** (
293. **PARTITION** **BY** employee\_id
294. **ORDER** **BY** from\_date
295. ) **AS** amount,
296. *-- need to rank by descending to get latest record*
297. ROW\_NUMBER() **OVER** (
298. **PARTITION** **BY** employee\_id
299. **ORDER** **BY** to\_date **DESC**
300. ) **AS** record\_rank
301. **FROM** mv\_employees.salary
302. ) all\_salaries
303. *-- keep only latest previous\_salary records only*
304. *-- must have this in subquery to account for execution order*
305. **WHERE** record\_rank **=** 1
306. ),
307. cte\_join\_data **AS** (
308. **SELECT**
309. employee.**id** **AS** employee\_id,
310. employee.birth\_date,
311. *-- calculated employee\_age field*
312. DATE\_PART('year', now()) **-**
313. DATE\_PART('year', employee.birth\_date) **AS** employee\_age,
314. *-- employee full name*
315. CONCAT\_WS(' ', employee.first\_name, employee.last\_name) **AS** employee\_name,
316. employee.gender,
317. employee.hire\_date,
318. title.title,
319. salary.amount **AS** salary,
320. *-- need to separately define the previous\_latest\_salary*
321. *-- to differentiate between the following lag record!*
322. cte\_previous\_salary.amount **AS** previous\_latest\_salary,
323. department.dept\_name **AS** department,
324. *-- use the `manager` aliased version of employee table for manager*
325. CONCAT\_WS(' ', manager.first\_name, manager.last\_name) **AS** manager,
326. *-- calculated tenure fields*
327. DATE\_PART('year', now()) **-**
328. DATE\_PART('year', employee.hire\_date) **AS** company\_tenure\_years,
329. DATE\_PART('year', now()) **-**
330. DATE\_PART('year', title.from\_date) **AS** title\_tenure\_years,
331. DATE\_PART('year', now()) **-**
332. DATE\_PART('year', department\_employee.from\_date) **AS** department\_tenure\_years,
333. *-- we also need to use AGE & DATE\_PART functions here to generate month diff*
334. DATE\_PART('months', AGE(now(), title.from\_date)) **AS** title\_tenure\_months,
335. GREATEST(
336. title.from\_date,
337. salary.from\_date,
338. department\_employee.from\_date,
339. department\_manager.from\_date
340. ) **AS** effective\_date,
341. LEAST(
342. title.to\_date,
343. salary.to\_date,
344. department\_employee.to\_date,
345. department\_manager.to\_date
346. ) **AS** expiry\_date
347. **FROM** mv\_employees.employee
348. **INNER** **JOIN** mv\_employees.title
349. **ON** employee.**id** **=** title.employee\_id
350. **INNER** **JOIN** mv\_employees.salary
351. **ON** employee.**id** **=** salary.employee\_id
352. **INNER** **JOIN** mv\_employees.department\_employee
353. **ON** employee.**id** **=** department\_employee.employee\_id
354. *-- NOTE: department is joined only to the department\_employee table!*
355. **INNER** **JOIN** mv\_employees.department
356. **ON** department\_employee.department\_id **=** department.**id**
357. *-- add in the department\_manager information onto the department table*
358. **INNER** **JOIN** mv\_employees.department\_manager
359. **ON** department.**id** **=** department\_manager.department\_id
360. *-- join again on the employee\_id field to another employee for manager's info*
361. **INNER** **JOIN** mv\_employees.employee **AS** manager
362. **ON** department\_manager.employee\_id **=** manager.**id**
363. *-- join onto our previous cte\_previous\_salary only for previous\_latest\_salary*
364. **INNER** **JOIN** cte\_previous\_salary
365. **ON** mv\_employees.employee.**id** **=** cte\_previous\_salary.employee\_id
366. ),
367. *-- now we apply the window function to order our transactions*
368. *-- we will filter out the top 5 in the next CTE step*
369. cte\_ordered\_transactions **AS** (
370. **SELECT**
371. employee\_id,
372. birth\_date,
373. employee\_age,
374. employee\_name,
375. gender,
376. hire\_date,
377. title,
378. LAG(title) **OVER** w **AS** previous\_title,
379. salary,
380. *-- previous latest salary is based off the CTE*
381. previous\_latest\_salary,
382. LAG(salary) **OVER** w **AS** previous\_salary,
383. department,
384. LAG(department) **OVER** w **AS** previous\_department,
385. manager,
386. LAG(manager) **OVER** w **AS** previous\_manager,
387. company\_tenure\_years,
388. title\_tenure\_years,
389. title\_tenure\_months,
390. department\_tenure\_years,
391. effective\_date,
392. expiry\_date,
393. *-- we use a reverse ordered effective date window to capture last 5 events*
394. ROW\_NUMBER() **OVER** (
395. **PARTITION** **BY** employee\_id
396. **ORDER** **BY** effective\_date **DESC**
397. ) **AS** event\_order
398. **FROM** cte\_join\_data
399. *-- apply logical filter to remove invalid records resulting from the join*
400. **WHERE** effective\_date **<=** expiry\_date
401. *-- define window frame with chronological ordering by effective date*
402. WINDOW
403. w **AS** (**PARTITION** **BY** employee\_id **ORDER** **BY** effective\_date)
404. ),
405. *-- finally we apply our case when statements to generate the employee events*
406. *-- and generate our benchmark comparisons for the final output*
407. *-- we aliased our FROM table as "base" for compact code!*
408. final\_output **AS** (
409. **SELECT**
410. base.employee\_id,
411. base.gender,
412. base.birth\_date,
413. base.employee\_age,
414. base.hire\_date,
415. base.title,
416. base.employee\_name,
417. base.previous\_title,
418. base.salary,
419. *-- previous latest salary is based off the CTE*
420. previous\_latest\_salary,
421. *-- previous salary is based off the LAG records*
422. base.previous\_salary,
423. base.department,
424. base.previous\_department,
425. base.manager,
426. base.previous\_manager,
427. *-- tenure metrics*
428. base.company\_tenure\_years,
429. base.title\_tenure\_years,
430. base.title\_tenure\_months,
431. base.department\_tenure\_years,
432. base.event\_order,
433. *-- only include the latest salary change for the first event\_order row*
434. **CASE**
435. **WHEN** event\_order **=** 1
436. **THEN** ROUND(
437. 100 **\*** (base.salary **-** base.previous\_latest\_salary) **/**
438. base.previous\_latest\_salary::NUMERIC,
439. 2
440. )
441. **ELSE** **NULL**
442. **END** **AS** latest\_salary\_percentage\_change,
443. *-- also include the amount change*
444. **CASE**
445. **WHEN** event\_order **=** 1
446. **THEN** ROUND(
447. base.salary **-** base.previous\_latest\_salary
448. )
449. **ELSE** **NULL**
450. **END** **AS** latest\_salary\_amount\_change,
451. *-- event type logic by comparing all of the previous lag records*
452. **CASE**
453. **WHEN** base.previous\_salary **<** base.salary
454. **THEN** 'Salary Increase'
455. **WHEN** base.previous\_salary **>** base.salary
456. **THEN** 'Salary Decrease'
457. **WHEN** base.previous\_department **<>** base.department
458. **THEN** 'Dept Transfer'
459. **WHEN** base.previous\_manager **<>** base.manager
460. **THEN** 'Reporting Line Change'
461. **WHEN** base.previous\_title **<>** base.title
462. **THEN** 'Title Change'
463. **ELSE** **NULL**
464. **END** **AS** event\_name,
465. *-- salary change*
466. ROUND(base.salary **-** base.previous\_salary) **AS** salary\_amount\_change,
467. ROUND(
468. 100 **\*** (base.salary **-** base.previous\_salary) **/** base.previous\_salary::NUMERIC,
469. 2
470. ) **AS** salary\_percentage\_change,
471. *-- benchmark comparisons - we've omit the aliases for succinctness!*
472. *-- tenure*
473. ROUND(tenure\_benchmark\_salary) **AS** tenure\_benchmark\_salary,
474. ROUND(
475. 100 **\*** (base.salary **-** tenure\_benchmark\_salary)
476. **/** tenure\_benchmark\_salary::NUMERIC
477. ) **AS** tenure\_comparison,
478. *-- title*
479. ROUND(title\_benchmark\_salary) **AS** title\_benchmark\_salary,
480. ROUND(
481. 100 **\*** (base.salary **-** title\_benchmark\_salary)
482. **/** title\_benchmark\_salary::NUMERIC
483. ) **AS** title\_comparison,
484. *-- department*
485. ROUND(department\_benchmark\_salary) **AS** department\_benchmark\_salary,
486. ROUND(
487. 100 **\*** (salary **-** department\_benchmark\_salary)
488. **/** department\_benchmark\_salary::NUMERIC
489. ) **AS** department\_comparison,
490. *-- gender*
491. ROUND(gender\_benchmark\_salary) **AS** gender\_benchmark\_salary,
492. ROUND(
493. 100 **\*** (base.salary **-** gender\_benchmark\_salary)
494. **/** gender\_benchmark\_salary::NUMERIC
495. ) **AS** gender\_comparison,
496. *-- usually best practice to leave the effective/expiry dates at the end*
497. base.effective\_date,
498. base.expiry\_date
499. **FROM** cte\_ordered\_transactions **AS** base *-- used alias here for the joins below*
500. **INNER** **JOIN** mv\_employees.tenure\_benchmark
501. **ON** base.company\_tenure\_years **=** tenure\_benchmark.company\_tenure\_years
502. **INNER** **JOIN** mv\_employees.title\_benchmark
503. **ON** base.title **=** title\_benchmark.title
504. **INNER** **JOIN** mv\_employees.department\_benchmark
505. **ON** base.department **=** department\_benchmark.department
506. **INNER** **JOIN** mv\_employees.gender\_benchmark
507. **ON** base.gender **=** gender\_benchmark.gender
508. *-- apply filter to only keep the latest 5 events per employee*
509. *-- WHERE event\_order <= 5*
510. )
511. *-- finally we are done with the historic values*
512. **SELECT** **\*** **FROM** final\_output;
513. *-- This final view powers the employee deep dive tool*
514. *-- by keeping only the 5 latest events*
515. **DROP** **VIEW** **IF** **EXISTS** mv\_employees.employee\_deep\_dive;
516. **CREATE** **VIEW** mv\_employees.employee\_deep\_dive **AS**
517. **SELECT** **\***
518. **FROM** mv\_employees.historic\_employee\_records
519. **WHERE** event\_order **<=** 5;

## 5.2. Current Company Snapshot Results

Using the mv\_employees.company\_level\_dashboard view - we can generate the same data outputs used to create the example visual shown below:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |

## 5.3. Department Level Results

**SELECT** **\*** **FROM** mv\_employees.department\_level\_dashboard

**ORDER** **BY** department, gender;

Click here to see query result

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |

## 5.4. Title Level Results

**SELECT** **\*** **FROM** mv\_employees.title\_level\_dashboard

**ORDER** **BY** title, gender;

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |

## 5.5. Salary Benchmark Results

### **5.5.1. Company Tenure Benchmark**

**SELECT** **\*** **FROM** mv\_employees.tenure\_benchmark

**ORDER** **BY** company\_tenure\_years;

Click here to see query result

|  |  |
| --- | --- |
|  |  |

### **5.5.2. Gender Benchmark**

**SELECT** **\*** **FROM** mv\_employees.gender\_benchmark;

Click here to see query result

|  |  |
| --- | --- |
|  |  |
|  |  |

### **5.5.3. Department Benchmark**

**SELECT** **\*** **FROM** mv\_employees.department\_benchmark

**ORDER** **BY** department\_benchmark\_salary **DESC**;

Click here to see query result

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

### **5.5.4. Title Benchmark**

**SELECT** **\*** **FROM** mv\_employees.title\_benchmark

**ORDER** **BY** title\_benchmark\_salary **DESC**;

Click here to see query result

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

## 5.6. Historic Employee Deep Dive Example

Click here to review the Employee Deep Dive visual output

Graphical user interface

Description automatically generated

**SELECT** **\*** **FROM** mv\_employees.employee\_deep\_dive

**WHERE** employee\_name **=** 'Leah Anguita'

**ORDER** **BY** event\_order;

Click here to see query result

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

## 5.7. Further Simplified Employee Events

We can also further reduce the above deep dive output into 2 separate views to simplify the data outputs required for the deep dive employee tool:

1. Current employee and salary benchmark details
2. Latest 5 historic employee events with detailed event info

### **5.7.1. Current Employee Deep Dive Outputs**

We can also generate all of our data outputs in order as per the deep dive visual example:

**SELECT**

employee\_id,

employee\_name,

UPPER(title **||** ' - ' **||** department) **AS** line\_1,

**CASE**

**WHEN** gender **=** 'M'

**THEN** UPPER('MALE ' **||** employee\_age **||** ', BIRTHDAY ' **||** birth\_date)

**ELSE** UPPER('FEMALE ' **||** employee\_age **||** ', BIRTHDAY ' **||** birth\_date)

**END** **AS** line\_2,

title\_tenure\_months,

company\_tenure\_years,

TO\_CHAR(salary, '$FM999,999,999') **AS** salary,

latest\_salary\_percentage\_change,

manager,

*-- salary benchmark values*

TO\_CHAR(tenure\_benchmark\_salary, '$FM999,999,999') **AS** tenure\_benchmark\_salary,

tenure\_comparison,

TO\_CHAR(title\_benchmark\_salary, '$FM999,999,999') **AS** title\_benchmark\_salary,

title\_comparison,

TO\_CHAR(department\_benchmark\_salary, '$FM999,999,999') **AS** department\_benchmark\_salary,

department\_comparison,

TO\_CHAR(gender\_benchmark\_salary, '$FM999,999,999') **AS** gender\_benchmark\_salary,

gender\_comparison

**FROM** mv\_employees.employee\_deep\_dive

**WHERE** employee\_name **=** 'Leah Anguita'

**AND** event\_order **=** 1;

**SELECT**

employee\_id,

event\_order,

event\_name,

**CASE**

**WHEN** event\_name **IN** ('Salary Increase', 'Salary Decrease')

**THEN** 'New salary: ' **||** TO\_CHAR(salary, '$FM999,999,999')

**WHEN** event\_name **=** 'Dept Transfer'

**THEN** 'To: ' **||** department

**WHEN** event\_name **=** 'Reporting Line Change'

**THEN** 'New manager: ' **||** manager

**WHEN** event\_name **=** 'Title Change'

**THEN** 'To: ' **||** title

**END** **AS** line\_1,

**CASE**

**WHEN** event\_name **=** 'Salary Increase'

**THEN** 'Increase: ' **||** TO\_CHAR(salary\_amount\_change, '$FM999,999,999') **||**

' (+' **||** ROUND(salary\_percentage\_change::NUMERIC, 1) **||** ' %)'

**WHEN** event\_name **=** 'Salary Decrease'

**THEN** 'Decrease: ' **||** TO\_CHAR(salary\_amount\_change, '$FM999,999,999') **||**

' (' **||** ROUND(salary\_percentage\_change::NUMERIC, 1) **||** ' %)'

**WHEN** event\_name **=** 'Dept Transfer'

**THEN** 'From: ' **||** previous\_department

**WHEN** event\_name **=** 'Reporting Line Change'

**THEN** 'Previous manager: ' **||** previous\_manager

**WHEN** event\_name **=** 'Title Change'

**THEN** 'To: ' **||** previous\_title

**END** **AS** line\_2,

effective\_date **AS** event\_date

**FROM** mv\_employees.employee\_deep\_dive

**WHERE** employee\_name **=** 'Leah Anguita'

**ORDER** **BY** event\_order;

| **employee\_id** | **event\_order** | **event\_name** | **line\_1** | **line\_2** | **event\_date** |
| --- | --- | --- | --- | --- | --- |
| 11669 | 1 | Title Change | To: Senior Engineer | To: Engineer | 2020-05-12 |
| 11669 | 2 | Salary Increase | New salary: $47,373 | Increase: $327 (+0.7 %) | 2020-05-11 |
| 11669 | 3 | Dept Transfer | To: Customer Service | From: Production | 2019-06-12 |
| 11669 | 4 | Salary Increase | New salary: $47,046 | Increase: $3,365 (+7.7 %) | 2019-05-11 |
| 11669 | 5 | Salary Decrease | New salary: $43,681 | Decrease: $-249 (-0.6 %) | 2018-05-11 |