Managing compensation is a complex endeavor that covers multiple facets including salaries, wages, bonuses, benefits, and additional incentives given to employees for their work. It involves navigating a delicate balance between budget constraints and providing fair rewards to employees.

In the practical scenario of our dummy data project, we address similar elements. This entails a fictional organization- Cityla, handling compensation management across various outlined factors. I've utilized Pandas to analyse data.

Project Venice (Exceeded Expectations)

- **Employees (3 OR ABOVE years in the company):**
  - Bonus: Regular bonus (10% for Grade A/B, 15% for C/D) + **50k additional bonus** (paid in full).
  - Hike:
    - Below 50th percentile: **18% hike**.
    - Above 50th percentile: **12% hike**.
- **Employees (2-3 years in the company):**
  - Bonus: Regular bonus (prorated for those joined after April 1, 2023)+ **25K**
- **Employees (Below Rating 7):**
  - Regular Bonus: **50% bonus** (paid in full).
  - Hike: **5%**

Project Tokyo (Met Expectations)

- **Employees (3 OR ABOVE years in the company):**
  - Bonus: Regular bonus (10% for Grade A/B, 15% for C/D) + **20k additional bonus** (paid in full).
  - Hike:
    - Below 50th percentile: **12% hike**.
    - Above 50th percentile: **10% hike**.
- **Employees (2-3 years in the company):**
  - Bonus: Regular bonus (prorated for those joined after April 1, 2023)+**10K**
- **Employees (Below Rating 7):**
  - Regular Bonus: **50% bonus** (paid in full).
  - Hike: **5%**
  - (if joined after April 1, 2023): Prorated hike and bonus.

Project Gemcom (Below Expectations)- **5% Hike**

- **Employees (All):**
  - Bonus: 50% of Regular bonus (10% for Grade A/B, 15% for C/D) (paid in full).
  - Hike (if joined after April 1, 2023): Prorated hike and bonus.

Note:

- Regular bonus percentages are based on employee grade.
- Joining after April 1, 2023 impacts bonus and hike calculations (prorated).

Table creation & Data Exploration

Query –

```
CREATE TABLE Salaries(
Start_Year VARCHAR(50),
Start_Date DATE,
Employee_Name VARCHAR(50),
unique_ID VARCHAR(50),
Level_ID  VARCHAR(50),
Grade VARCHAR(50),
Project VARCHAR(50),
Current_Base_Pay numeric,
Bonus_entitlement numeric,
Rating numeric

)
```

```
select * FROM Salaries
WHERE unique_id = '557'
```

Output

Data Output | Messages | Notifications

	start_year character varying (50)	start_date date	employee_name character varying (50)	unique_id character varying (50)	level_id character varying (50)	grade character varying (50)	project character varying (50)	current_base_pay numeric	bonus_entitlement numeric	rating numeric
1	2023	2023-08-21	K. Shreya	557	3	Senior Software Engineer	Gemcon	2697410	0.15	7

No. of records-

Query –

```
SELECT count(distinct unique_id) FROM Salaries;
```

Output –

Data Output | Messages | Notifications

	count bigint
1	561

```
----------------------------------- employee count as per grade----------------------

select grade,
count(distinct unique_id) as employees
FROM Salaries
GROUP BY grade;
```

Output-

Data Output | Messages | Notifications

	grade character varying (50)	employees bigint
1	Associate	122
2	Lead Engineer	85
3	Senior Software Engineer	164
4	Software Engineer	190

```
-------------------------------------employee project allocation & grade(intermediate)-----------------------------

select grade,
COUNT( CASE WHEN project = 'Tokyo' THEN unique_id END) AS Tokyo,
COUNT( CASE WHEN project = 'Venice' THEN unique_id END) AS Venice,
COUNT( CASE WHEN project = 'Gemcon' THEN unique_id END) AS Gemcon
FROM
Salaries
GROUP BY
grade
ORDER BY
grade;
```

Output-

Data Output | Messages | Notifications

	grade character varying (50)	tokyo bigint	venice bigint	gemcon bigint
1	Associate	30	43	49
2	Lead Engineer	25	27	33
3	Senior Software Engineer	61	60	43
4	Software Engineer	56	63	71

```
----------------------------------average ratings across projects-----------------------------------------------

select grade,
ROUND(AVG( CASE WHEN project = 'Tokyo' THEN rating END),2) AS Tokyo,
ROUND(AVG( CASE WHEN project = 'Venice' THEN rating END),2) AS Venice,
ROUND(AVG( CASE WHEN project = 'Gemcon' THEN rating END),2) AS Gemcon
FROM
Salaries
GROUP BY
grade
ORDER BY
grade;
```

Output-

Data Output | Messages | Notifications

	grade character varying (50)	tokyo numeric	venice numeric	gemcon numeric
1	Associate	8.57	7.88	7.96
2	Lead Engineer	8.00	7.89	8.27
3	Senior Software Engineer	8.03	7.93	7.74
4	Software Engineer	7.77	7.98	7.96

```sql
--------------------------Average Base Pay across projects and grade----------------------------------
select grade,
ROUND(AVG( CASE WHEN project = 'Tokyo' THEN current_Base_Pay END),0) AS Tokyo,
ROUND(AVG( CASE WHEN project = 'Venice' THEN current_Base_Pay  END),0) AS Venice,
ROUND(AVG( CASE WHEN project = 'Gemcon' THEN current_Base_Pay  END),0) AS Gemcon
FROM
Salaries
GROUP BY
grade
ORDER BY
grade;
```

Output-

Data Output | Messages | Notifications

	grade character varying (50)	tokyo numeric	venice numeric	gemcon numeric
1	Associate	1489989	1523179	1488618
2	Lead Engineer	2535402	2858086	2766288
3	Senior Software Engineer	2421193	2417559	2378832
4	Software Engineer	2052550	2076845	2127771

```sql
--------------------------------Percentile calculation-------------------------------------------------
ALTER TABLE Salaries
ADD COLUMN percentile_category VARCHAR(50);

WITH Percentiles AS (
    SELECT
        *,
        PERCENT_RANK() OVER (PARTITION BY grade ORDER BY current_base_pay) AS percentile_rank
    FROM
        Salaries
)
UPDATE Salaries AS s
SET percentile_category =
    CASE
        WHEN p.percentile_rank <= 0.25 THEN 'Below 25th'
        WHEN p.percentile_rank <= 0.50 THEN 'Between 25th to 50th'
        WHEN p.percentile_rank <= 0.75 THEN 'Between 50th to 75th'
        ELSE 'Above 75th'
    END
FROM
    Percentiles AS p
WHERE
    s.unique_id = p.unique_id;
```

Output-

ric	percentile_category character varying (50)
7	Below 25th
6	Below 25th
8	Below 25th
10	Below 25th
9	Below 25th
8	Below 25th
10	Below 25th
8	Between 25th to 50th
8	Between 25th to 50th
10	Between 25th to 50th
10	Between 25th to 50th
8	Between 25th to 50th
8	Between 25th to 50th
7	Between 25th to 50th

In this above query:

• We use a CTE named Percentiles to calculate the percentile rank for each employee within their grade.Then, we update the Salaries table using the UPDATE ... FROM ... WHERE syntax, joining the original table with the CTE based on the employee_id.Finally, we set the percentile_category column based on the percentile rank calculated in the CTE.

```
----now we wish to see no of employees in percentile rank vs project----------------------

select percentile_category,
COUNT( CASE WHEN project = 'Tokyo' THEN unique_id END) AS Tokyo,
COUNT( CASE WHEN project = 'Venice' THEN unique_id END) AS Venice,
COUNT( CASE WHEN project = 'Gemcon' THEN unique_id END) AS Gemcon
FROM
Salaries
GROUP BY
percentile_category
ORDER BY
percentile_category;
```

Output -

Data Output | Messages | Notifications

	percentile_category character varying (50)	tokyo bigint	venice bigint	gemcon bigint
1	Above 75th	36	59	46
2	Below 25th	48	47	47
3	Between 25th to 50th	40	45	54
4	Between 50th to 75th	48	42	49

```
----------------------------------------years in the company-------------------------------


-- Add a new column for time_in_company
ALTER TABLE Salaries
ADD COLUMN time_in_company NUMERIC;

-- Update the newly added column with the time difference in years and months

 UPDATE Salaries
SET time_in_company =
    ROUND(
        EXTRACT(YEAR FROM AGE('2024-04-01', start_date))::NUMERIC +
        EXTRACT(MONTH FROM AGE('2024-04-01', start_date))/12, 2
    );
```

Output-

time_in_company numeric
2.33
2.58
2.33
2.58
3.08
3.00
3.08
2.58

```
---this time in the company will come in handy when we dole out bonus basis time spent----------

------below calculations are to calculate prorated salaries mainly affecting those with <1 tenure

-- Add a new column for prorated
ALTER TABLE Salaries
ADD COLUMN prorated NUMERIC;

-- Update the newly added column based on the condition
UPDATE Salaries
SET prorated =
    CASE
        WHEN time_in_company >= 1 THEN 1
        ELSE time_in_company
    END;

----the above query will ensure that our final output will continue to adhere to prorated details.
```

	prorated numeric
5	0.75
5	0.75
3	0.83
2	0.92
2	0.92
2	0.92
2	0.92
0	1

```
----calculate regular bonus-------------------------------------

-- Add a new column for regular_bonus
ALTER TABLE Salaries
ADD COLUMN regular_bonus NUMERIC;

-- Update the newly added column based on the conditions
UPDATE Salaries
SET regular_bonus =
    Round(
        CASE
        WHEN project IN ('Venice', 'Tokyo') AND rating >= 7 THEN current_base_pay * bonus_entitlement * prorated
        ELSE current_base_pay * bonus_entitlement * prorated * 0.5
    END,2
);
```

Output-

	regular_bonus numeric
83	41500.00
92	46000.00
92	46000.00
92	92000.00
92	92000.00
1	100000.00
1	50000.00
1	50000.00

```
-----how much are we distributing as a regular bonus

select sum(regular_bonus) FROM  Salaries;

select ROUND(AVG(regular_bonus),2) FROM  Salaries;
```

The company is distributing 198768 bonus per person.

Data Output | Messages | Notifications

	round numeric
1	198768.99

```
-- Update the newly added column based on the conditions


UPDATE Salaries
SET Additional_Bonus =
    CASE
        WHEN project = 'Venice' AND time_in_company >= 3 AND rating >= 7 THEN 50000
        WHEN project = 'Venice' AND time_in_company >= 2 AND time_in_company < 3 AND rating >= 7 THEN 25000
        WHEN project = 'Tokyo' AND time_in_company >= 3 AND rating >= 7 THEN 20000
        WHEN project = 'Tokyo' AND time_in_company >= 2 AND time_in_company < 3 AND rating >= 7 THEN 10000
        ELSE 0
    END;
```

```sql
-----------------TOTAL_BONUS PAYOUT-----------------------------------------------------------------

ALTER TABLE Salaries
ADD COLUMN Total_Bonus numeric;

UPDATE Salaries
SET Total_bonus = regular_bonus + additional_bonus;
```

Output

regular_bonus numeric	additional_bonus numeric	total_bonus numeric
50000.00	0	50000.00
120493.70	25000	145493.70
61198.00	0	61198.00
62694.95	0	62694.95
126689.10	10000	136689.10
127390.30	10000	137390.30
64460.35	0	64460.35
130263.00	50000	180263.00

```sql
----------------------Now we set to display salary revision data-----------------------------------

ALTER TABLE Salaries
ADD COLUMN Hike_percent numeric;



UPDATE Salaries
SET Hike_percent=
    CASE
        WHEN project = 'Venice' AND percentile_category='Below 25th' AND rating >= 7 THEN 0.18
        WHEN project = 'Venice' AND percentile_category='Between 25th to 50th' AND rating >= 7 THEN 0.18
        WHEN project = 'Venice' AND percentile_category='Between 50th to 75th' AND rating >= 7 THEN 0.12
        WHEN project = 'Venice' AND percentile_category='Above 75th' AND rating >= 7 THEN 0.12
        WHEN project = 'Tokyo' AND  percentile_category='Below 25th' AND rating >= 7 THEN 0.12
        WHEN project = 'Tokyo' AND  percentile_category='Between 25th to 50th' AND rating >= 7 THEN 0.12
        WHEN project = 'Tokyo' AND  percentile_category='Between 50th to 75th' AND rating >= 7 THEN 0.10
        WHEN project = 'Tokyo' AND  percentile_category='Above 75th' AND rating >= 7 THEN 0.10
        ELSE 0.05
    END;
```

```
----------------------New Base Pay-----------------------------------------------

ALTER TABLE Salaries
ADD COLUMN New_Base_Pay numeric;


UPDATE Salaries
SET New_Base_Pay  = ROUND(Current_base_pay +(current_base_pay*prorated*hike_percent),0)


SELECT grade,Project, rating, prorated, current_base_pay, new_base_pay,total_bonus FROM Salaries
WHERE grade = 'Associate' AND project = 'Gemcon'
```

Output-

additional_bonus numeric	total_bonus numeric	hike_percent numeric	new_base_pay numeric
0	62694.95	0.05	1316594
10000	136689.10	0.12	1418918
10000	137390.30	0.12	1426771
0	64460.35	0.05	1353667
50000	180263.00	0.18	1537103
0	66002.85	0.05	1386060
50000	182369.30	0.18	1561958
10000	142407.10	0.12	1482960