Suppose you are an ETL Developer of a company. Your company acquired a new client. The new client is a hospital who wants to utilize their HR/Timesheet data to gain insights into the risk of burnout of nurses working there. You as an ETL Developer are assigned to take lead on the project. Please gather necessary requirements in order to make informed decisions.

**Logical Modelling**

**Dimension tables identified:**

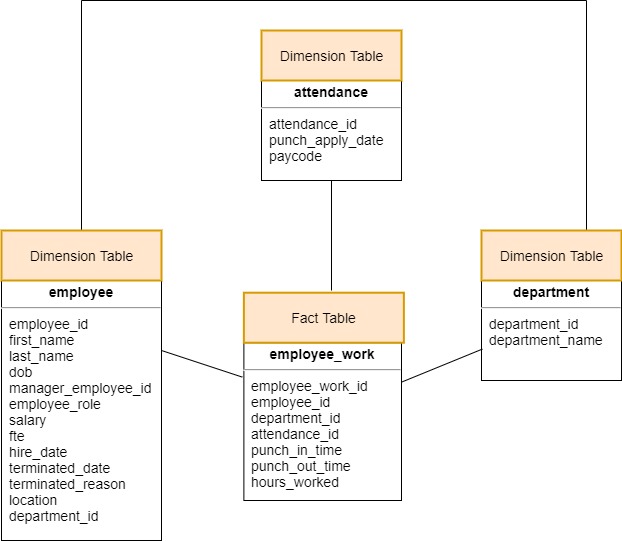
* employee
* attendance
* department

**Fact table:**

* employee\_work
* The fact table has measures punch\_in\_time, punch\_out\_time and hours\_worked as they are all values that need to be aggregated and according to the business requirement, getting insight into burnout of nurses working there, getting to know the amount of work they have done is important.

**Relationships between the tables:**

* An employee has a department assigned to them
* An employee can work for a different department
* An employee has various attendance throughout the day
* An employee on leave does not have punch\_in\_time and punch\_out\_time but has hours\_worked



*Schema for the healthcare scenario*

**Physical Modelling**

Identifying the domains, datatypes, and constraints

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Domain** | **Datatype and Constraints** |
| **department** | | |
| department\_id | Unique identifier for department table | INT, PRIMARY KEY |
| department\_name | Name of the department | TEXT, NOT NULL |
| **Employee** | | |
| employee\_id | Unique identifier of employee table | INT, PRIMARY KEY |
| first\_name | First name of the employee | TEXT, NOT NULL |
| last\_name | Last name of the employee | TEXT, NOT NULL |
| dob | Date of birth of the employee | DATE, NOT NULL |
| manager\_employee\_id | Manager of the employee | INT, NOT NULL, FOREIGN KEY |
| employee\_role | The role of the employee | TEXT, NOT NULL |
| salary | Salary of the employee | INT, NOT NULL |
| fte | Full time equivalent where 1 represents full time | FLOAT, NOT NULL |
| hire\_date | Date when the employee got hired | DATE, NOT NULL |
| terminated\_date | Date when the employee’s work span got terminated | DATE, NOT NULL |
| terminated\_reason | The reason for the employee to leave the firm | TEXT, NOT NULL |
| location | The location of the employee | TEXT, NOT NULL |
| department\_id | The department for which the employee has been allocated | INT, NOT NULL, FOREIGN KEY |
| **attendance** | | |
| attendance\_id | Unique identifier of attendance table | INT, PRIMARY KEY, AUTO INCREMENT |
| punch\_apply\_date | The date for which the attendance is valid | DATE, NOT NULL |
| paycode | The status of the employee for the day where the employee can be working, on call, on leave or in charge | TEXT, NOT NULL |
| **employee\_work** | | |
| employee\_work\_id | Unique identifier of employee\_work table | INT, PRIMARY KEY, AUTO INCREMENT |
| employee\_id | The employee who is relevant to the work | INT, NOT NULL, FOREIGN KEY |
| department\_id | The department in which the employee is working for the day | INT, NOT NULL, FOREIGN KEY |
| attendance\_id | The attendance of the employee | INT, NOT NULL, FOREIGN KEY |
| punch\_in\_time | The start time for the work | TIMESTAMP |
| punch\_out\_time | The end time for the work | TIMESTAMP |
| hours\_worked | Total hours worked by the employee in this instance | FLOAT, NOT NULL |

**Requirements**

1. **Clients should be able to know if an employee was working on a particular day or not.**

* To check if an employee was working on a particular day, the day must be passed as a condition to be met and the paycode must be checked. If the paycode for the employee is anything other than ABSENT, the employee has been working for the day. The join must be performed between employee table and attendance table.

If they worked,

1. **What time did they start and left?**

* With the continuation of the previous query, we JOIN the fact table called employee\_work ON attendance\_id with attendance table and return the minimum value of punch\_in\_time using MIN function and maximum value of punch\_out\_time using MAX function and must be grouped by the particular day supplied.

1. **How many hours?**

* Now, the total hours can be counted by using the COUNT function on hours\_worked column, and must be grouped by the particular day supplied.

1. **Were they charge on the day?**

* The paycode can be checked if it is CHARGE or not for all the appointments held on that day.

If they didn’t,

1. **Were they on call?**

* The paycode can be checked if it is ON\_CALL or not for all the appointments held on that day.

1. **Clients should be able to know if the employee had a Morning (Starting between 5: 00 AM - 11:00 AM) or Evening (Starting after 12:00 PM) shift.**

* Join operation between employee and employee\_work
* Minimum value of punch\_in\_time for the entire day for an employee
* Maximum value of punch\_out \_time for the entire day for an employee
* Check if the time values of the date fall in the range 5-11 for A.M., 12+ for P.M.

1. **Clients should be able to know if the employees are working regularly on a weekend (SUN, SAT)**

* Check if the paycode is not ABSENT
* Check if the day is either 0 or 1 using DAY function for punch\_apply\_date

1. **Clients want to analyze if any employee has to cover for other team members regularly.**

* Find the total sum of hours\_worked grouping by each day
* Check if the hours\_worked is greater than 8 (assuming overtime working means covering for other employee)
* JOIN employee and employee\_work table

1. **Clients want to analyze the data on a biweekly basis starting from 2021-01-01**

* Condition the punch\_apply\_date to be equal or greater to 2021-01-01
* Group by the interval of 14days
* JOIN all the tables with employee\_work table
* SELECT necessary attributes to return

1. **Clients want to analyze the data based on the employee role.**

* JOIN all tables to employee\_work
* Supply the employee role to be a particular employee role
* SELECT necessary attributes to return

1. **Clients want to analyze the salary distribution by department.**

* JOIN employee and department on department\_id
* Average the salary of the employees
* GROUP BY department\_id
* SELECT department and average salary