# **Part 1.1:**

1. First of all I have created the mentioned tables in Initialize\_Tables.sql file.
2. Columns names starts with C\_ for readability.
3. Second I have created range partition for the Attendance table as it will expand a lot in the future and by doing that we will be able to archive old data since the data is stored on partitions and these can be archived easily.
4. Moreover creating partition for table should be done when we create the table as it should be empty. However if we decided to make it after that we should create another table, make the partitions then transfer the data and finally rename it to original one.
5. Add values check for C\_STATUS in attendance table to make sure we don’t receive any other data in case we didn’t implement validation on the stored procedure.

# **Part 1.2:**

1. I have created the mark attendance procedure with some extra fields (O\_ERROR\_CODE, O\_ERROR\_DESC) as they are useful to catch the result of the execution.
2. Add checks for null values as an additional layer of security to insure we get correct values beside the constraints on the table itself.
3. Add validation for C\_STATUS column.
4. I have used the merge for inserting or updating values as it has better performance than if we use select query to check if the record is exists then updating it or creating it if its not exists.
5. At the end if we have any error we will raise an exception and return what we caught and in case there is unknown error it will be caught in OTHERS exception.

# **Part 2.1:**

Well in this task I couldn’t find anything could be changed in the query. However it has bottlenecks that we can solve by:

1. Since the C\_STATUS column is having only two values we can create a bitmap index on it as it’s very useful for this kind of data, for example we can apply it too for gender (male, female).
2. As I already created range partition on Attendance table using (C\_ATTENDANCEDATE) column we can only access data that exists in this range avoiding full-table-scan.
3. Also I have created an index on the C\_EMPLOYEEID in Attendance table.

# **Part 3.1:**

For this part there is no need to complicate the code so I didn’t add any new datatypes. However in real situations the employee will have many data as (phone-number, birthdate, address, etc...) so In this case we could implement a new datatype to handle all of these information in one object of this datatype.

# **Part 3.2:**

1. I have added C\_BIRTHDATE for employee table in order to make a constraint for (first, last and birthdate) of the employee.
2. Add not null constraints for many attributes.

# **Part 4.1:**

Well in my previous job I used to develop and test directly on the database using PL/SQL developer Tool.

1. I used to implement any new feature on the TEST DB so it’s used for (development & staging).
2. Get it tested by testing team.
3. Make a copy of any (package, stored procedure, function) if it’s live before updating it.
4. Prepare script files.
5. Execute the script files on production DB during the downtime of the operation.
6. Sometimes we need to insert new data and I used to make a dblink between the TEST DB and the Production DB to make it easier for transfer.
7. In case I had to make a rollback I can re-execute the copy of the object by changing its name to the live one.
8. Same thing is applied for configuration tables as I get a copy of them in order to return the same values in case something went wrong.

# **Part 4.2:**

1. Git init.
2. Git branch NewBranch / git checkout –b NewBranch.
3. Git commit –m “message about the new commit”.
4. Git push.