# **SQL Project**

**Table description**

In the corresponding xlsm file the initial data could be found. The population of the tables was made automatically by use of random functions implemented in Microsoft Excel. In some cases, additional functions were implemented to ensure the random generated data is unique. Some of the tables, however, were filled in manually to create some universal storages such as table of missions and table of classes. Such tables aggregates information about some universal characteristics that could be used later in larger tables of records. The fields included in the tables are following:

**Mission\_Tracker -** this table is the main one and represents the many-to-many relations between members and missions.

1. Rec\_id - the unique code of each record in the table. Simply incremented from 1 and so on..
2. Memb\_id - the unique code of each member of this organization. It is pulled as a foreign key from the table Members.
3. Mission\_id - the unique code of each mission held by this organization. It is also pulled as a foreign key from the table Missions.
4. Duration - the number of days each particular member works on the particular mission. This data can help for evaluation of needs for additional people on the missions as well as to gain the data about availability of the members.

**Members -** this table accumulates information about members, their prices, personal details and qualifications.

1. Memb\_id - the unique code of each member of this organization. Goes from 100 to 200 with prefix “mem\_”.
2. Memb\_name - the name of the member including their title.
3. Memb\_fee - the salary of each member per day. Also depends on the member’s qualification (plus 20% and 10% for High and Medium qualifications correspondingly) and the extreme conditions of the work (plus 30%, 20% and 10% for Conflict Zones, Epidemic Breaches and Natural Disasters correspondingly). These multipliers are mentioned in additional tables according to their relations.
4. Memb\_age - the age of the member, generated randomly in range from 18 to 70.
5. Class\_id - special codeword describing the abilities of the member, including his role and qualification. Description of each class is given in an additional table.
6. Manager\_id - the id of a member's direct manager. Were created manually to inform consistent hierarchy. First ten members are the highest managers with no supervisors above.

**Class** - this table has an in-depth description of the class, including the role and qualification of the member as well as multipliers that would be used in future calculations.

1. Class\_id - special codeword describing the abilities of the member, including his role and qualification.
2. Memb\_role - the role of the member.
3. Memb\_qual - the qualification of the member.
4. Qual\_index - multiplier that affects the salary of each member per day (plus 20% and 10% for High and Medium qualifications correspondingly).
5. Role\_index - additional salary for different specializations (plus 50, 40, 30 or 20 dollars for physicians, rescuers, technicians, nurses correspondingly).

**Missions** - this table has 15 missions with additional metrics that describe each one of it.

1. Mission\_id - the unique code of each mission held by this organization.
2. Loc\_id - the unique code of each location. This value is used as the reference to another table with description of locations.
3. Mission\_dist - the distance from the hub to the location of the mission, measured in meters.
4. Start\_date - the exact day when the mission started.
5. End\_date - the exact day when the mission ended or will end.
6. Mission\_type - the type of the mission.
7. Mission\_status - the comments regarding the status of the mission (normal, challenges, caution).

**Locations** - this table accumulates information about the possible locations of the mission.

1. Loc\_id - the unique code of each location.
2. Loc\_name - the name of the location.
3. Fee\_index - the multiplier to the member’s salary depending on the location.
4. Food\_lodge\_cost - the amount of expenses for food and living for each of the members based on the mission. The amount mentioned is per day.
5. Travel\_index - the multiplier to the travel cost that varies on the extreme conditions of the destination (plus 50% and 20% for Conflict Zones and Natural Disasters correspondingly).

**Type** - this table gives a brief description of each mission type.

1. Type - the type of the mission (humanitarian, medical, rescue).
2. Supply\_cost - the expenses for a supply for each member required for the mission.
3. Ppl\_req - special metric that approximates recommended number of members for a mission depending on its type.

All of the data was put into the script including primary and foreign keys. Due to the existence of many-to-many relationships, the huge attention was paid to the correct population functions to avoid any repetitions between unique elements and its pairs.

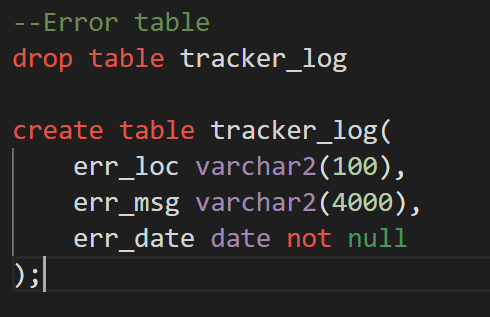
All of this information can be valuable and important for the managers and coordinators of the project to monitor their expenses and missions process. Through the work, new fields were added to the view that combines or divides some of the fields from multiple tables, such as:

1. Memb\_name\_full - full name of the member excluding their title.
2. Memb\_title - the title of the member.
3. Memb\_role\_f - the role of each member. It was added to the view for future convenience.
4. Memb\_qual\_f - the qualification of each member. It was added to the view for future convenience.
5. Mission\_type\_f - the type of the mission.
6. Mission\_duration - the duration of the mission. Calculated as the difference between end date and start date.
7. Travel\_cost - the expenses of the transportation of a member. Each kilometer costs 4.78 dollars. The final price is a product of distance and price per kilometer. Also, it varies on the extreme conditions of the destination (plus 50% and 20% for Conflict Zones and Natural Disasters correspondingly).
8. Total\_cost - the total expenses for each member, the sum of all kinds of expenses.

**Queries**

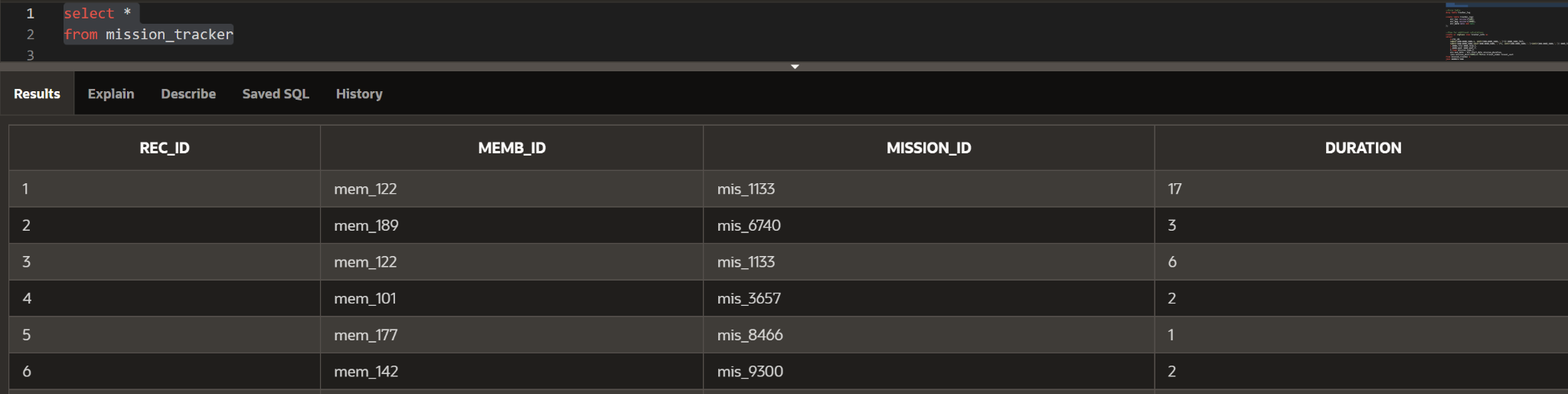
Before proceeding with queries, I decided to create an additional table to record all the occurring errors. Due to the implementation of a variety of functions and procedures, I

decided to track all exceptions and errors in an exact place called tracker\_log which keeps records of error messages, the name of the function/procedure it happened in and the sysdate.



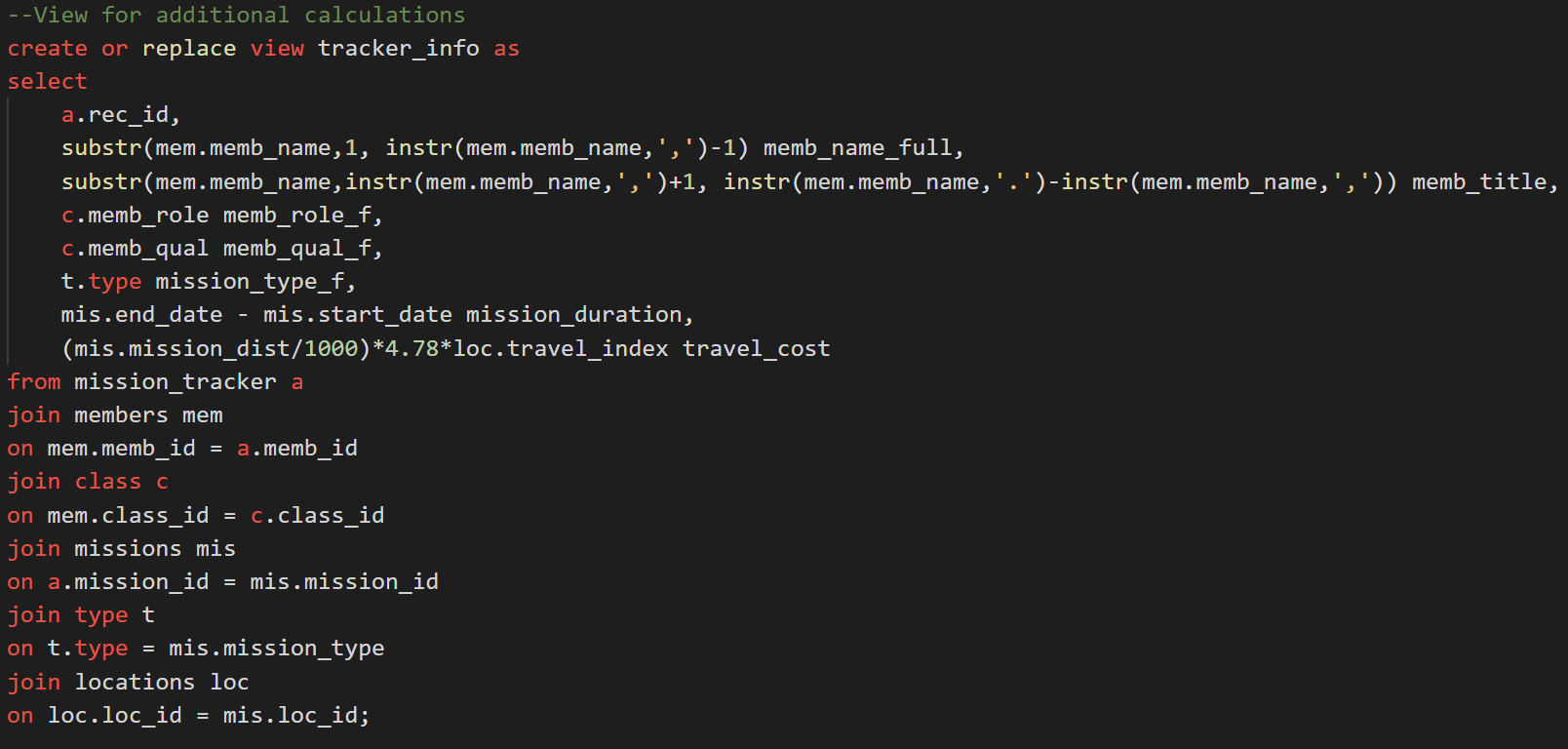
**Pic. 1- Create table tracker\_log**

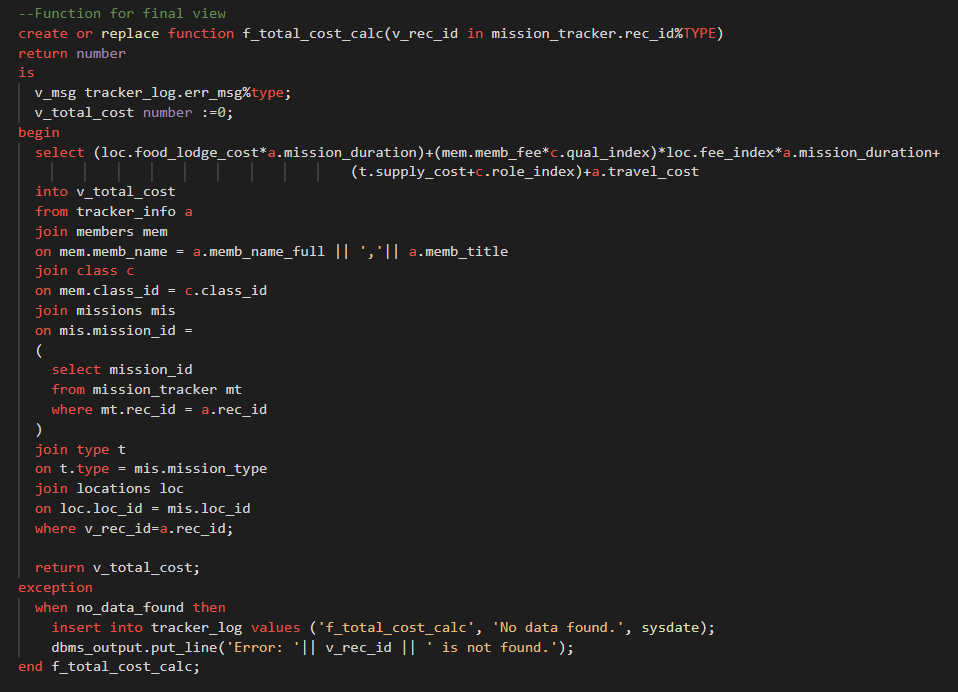
First, it is better to display the information we have by simply selecting everything from the main table.

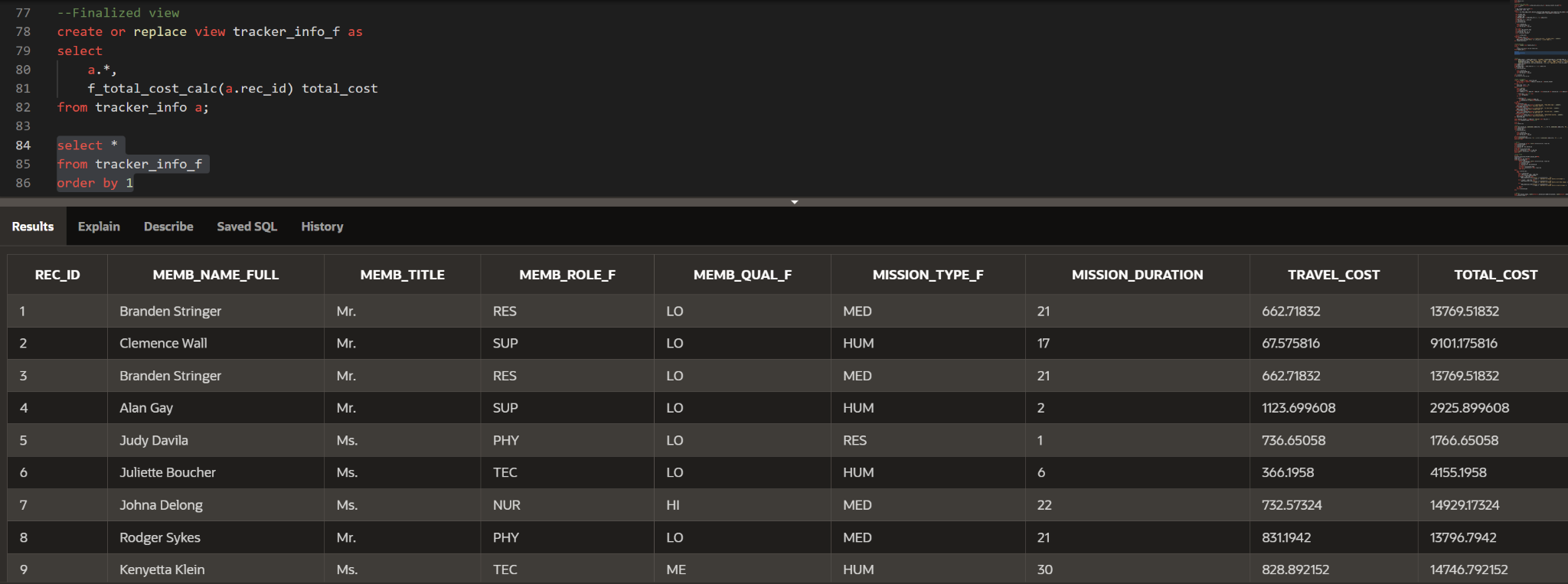


**Pic. 2- Select \***

Secondly, I created the view described before in two steps to add more important and valuable fields to the table. In order to calculate the last value of total cost, I decided to put this operation in a separate function that would return the value of total cost for a particular record id, including exception handling to prevent errors that might occur.



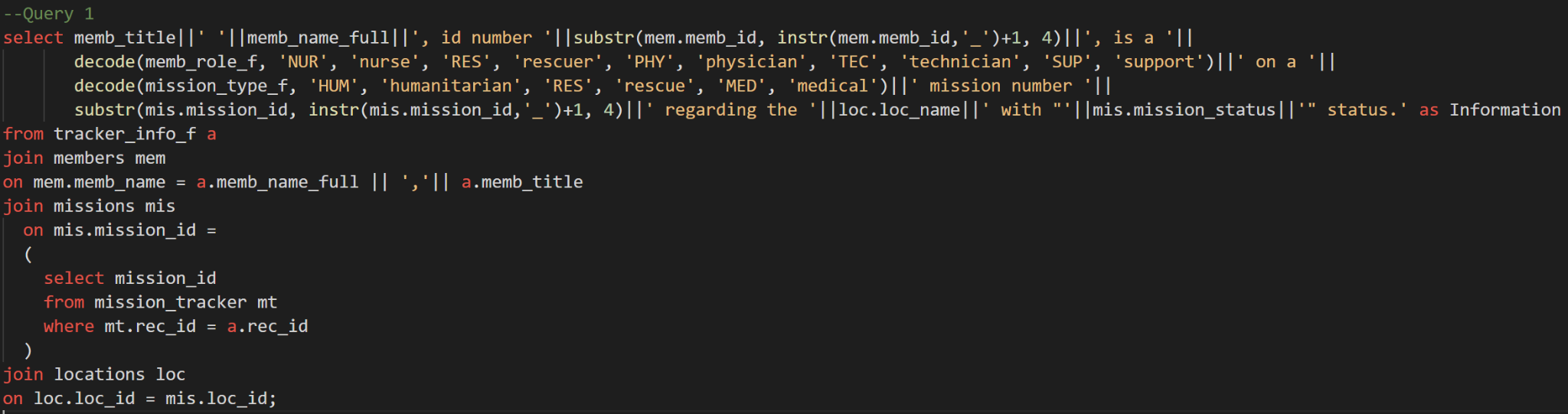


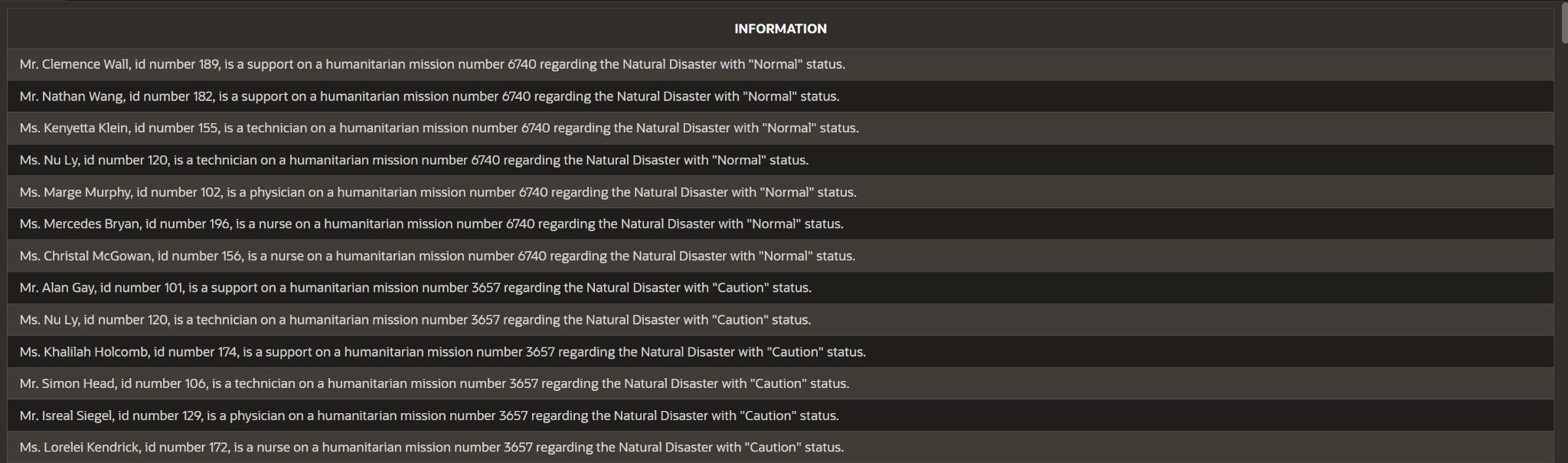


**Pic. 3.a, 3.b, 3.c - Queries for the views**

After these preparations, I created the queries for analysis.

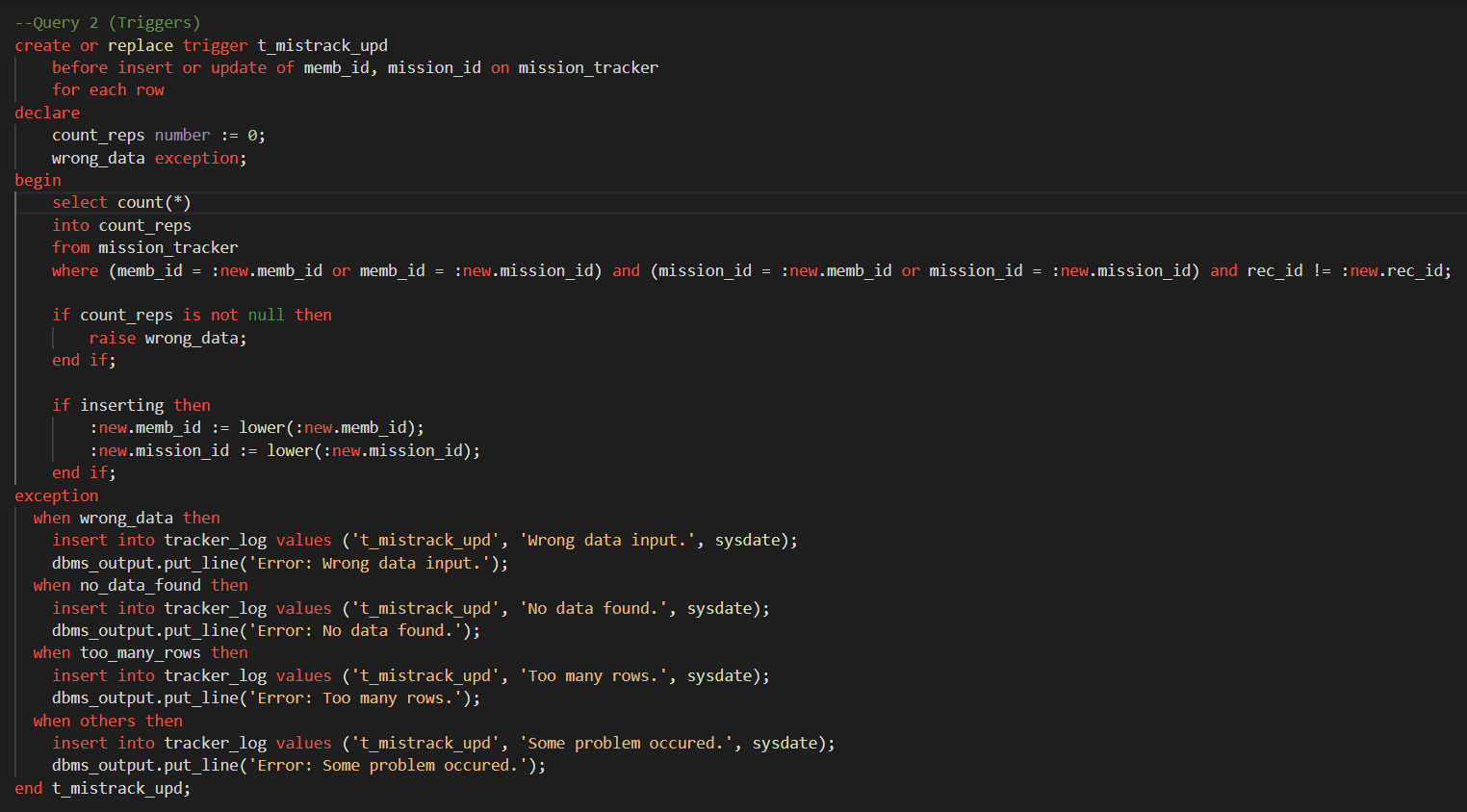
Firstly, I made the query to display the phrase explaining which member was assigned to which mission, including some additional information. This outcome may be used in official reports of the company.

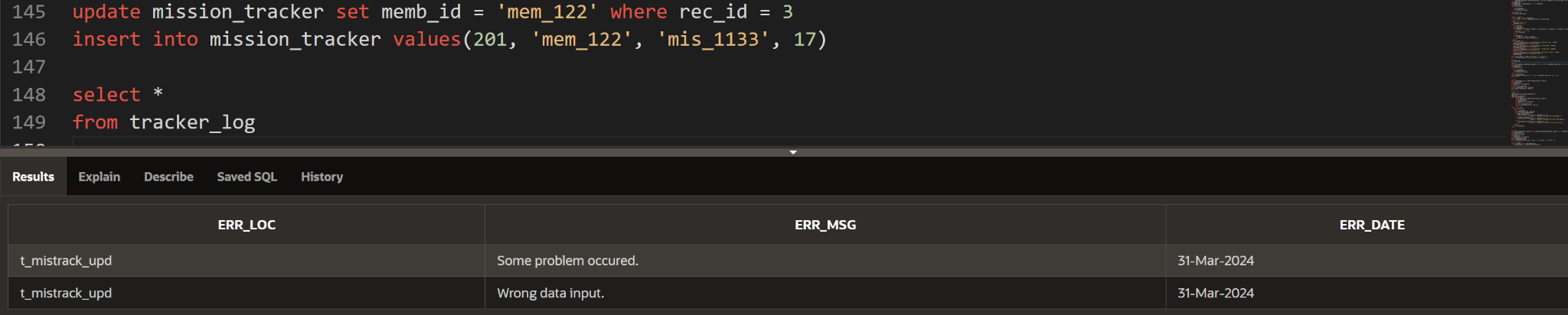




**Pic. 4,5- The information query and result**

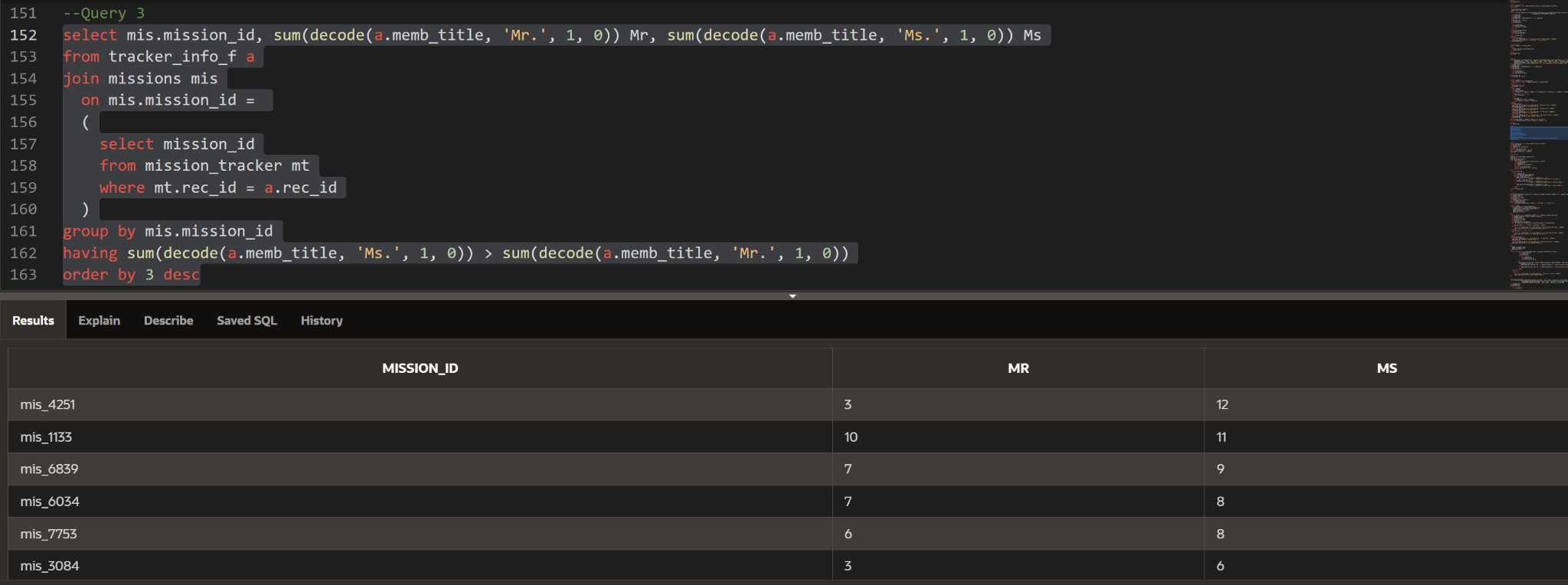
Next, I implemented the trigger to ensure the consistency of data in the mission\_tracker table. In order to do so, the trigger is executed before insert or update of the table. It checks if new data (the unique record of memb\_id and mission\_id) have already existed in the table, so it will not be written twice. If so, the corresponding exceptions is rised. There are also other exceptions that can happen. In case of successful insert, trigger makes sure to make the entries lowercase.





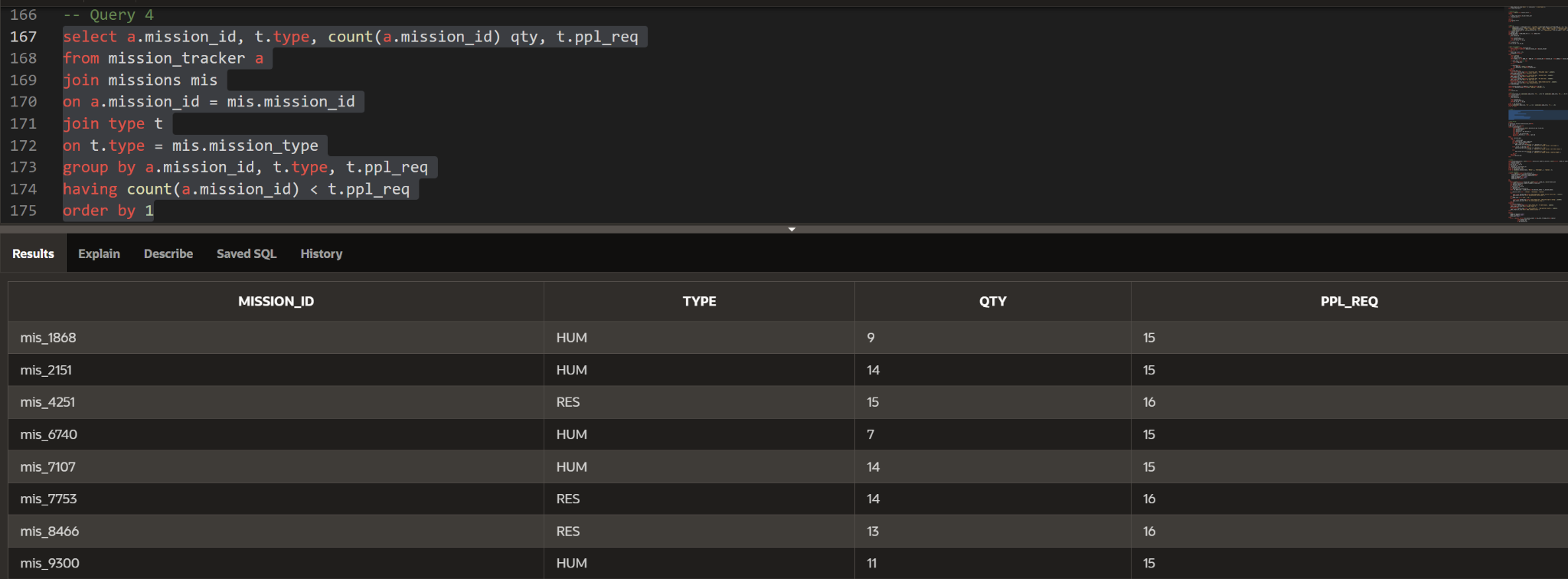
**Pic. 6, 7 - The insert/update trigger and result**

The third query is used to compare the information about how many men and women participate in each mission. In this scenario, only missions where there are more women than men are displayed.

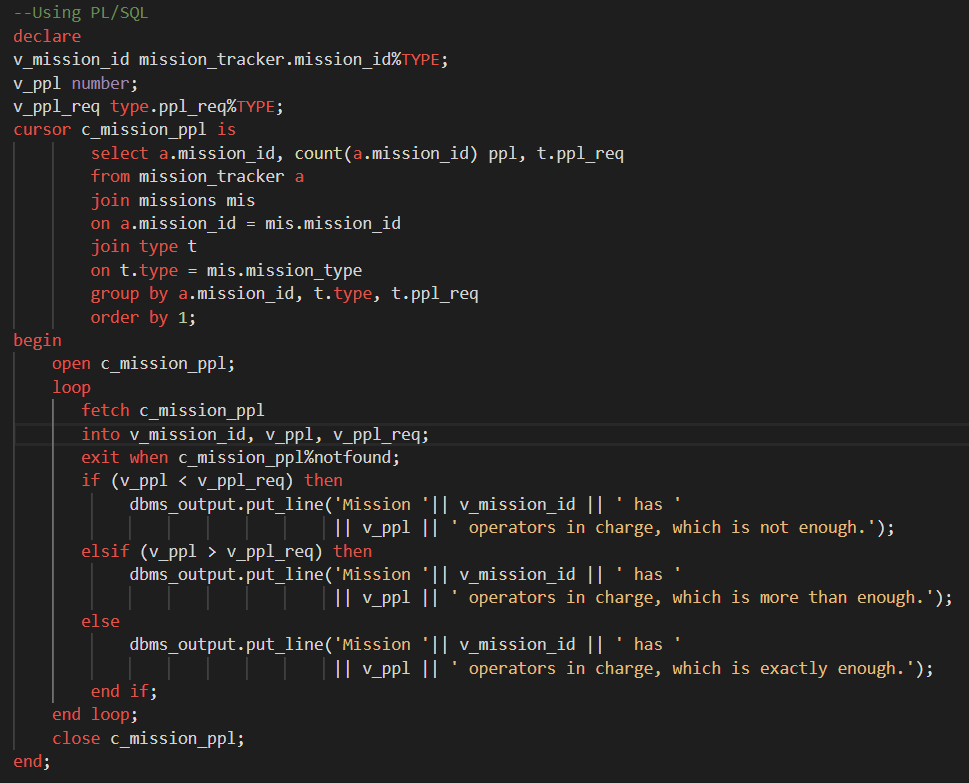


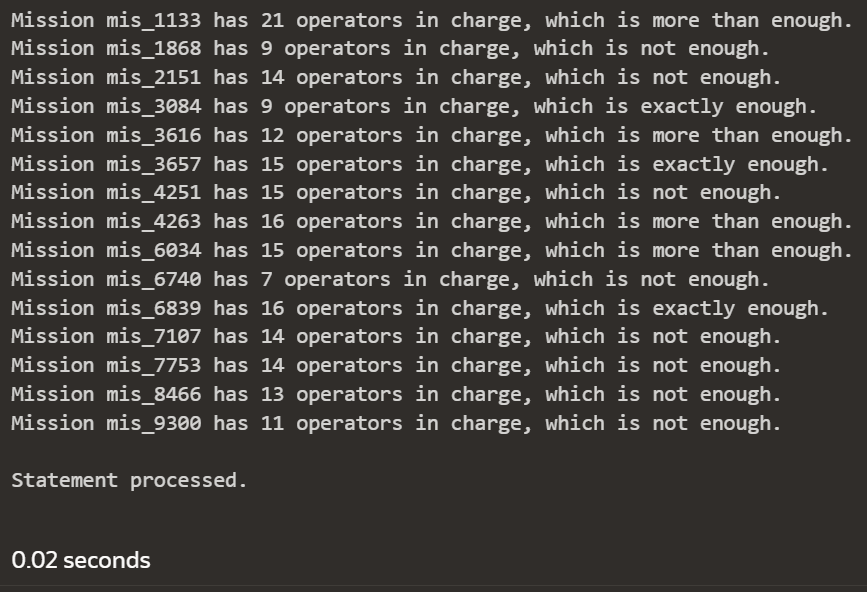
**Pic. 8 - The aggregation query and result**

The following query helps analytics to evaluate how successful each mission is going. For example, in this case we can aggregate all the missions that have less people working on it than it is recommended according to the type of the mission.

  
**Pic. 9 - The count query and result**

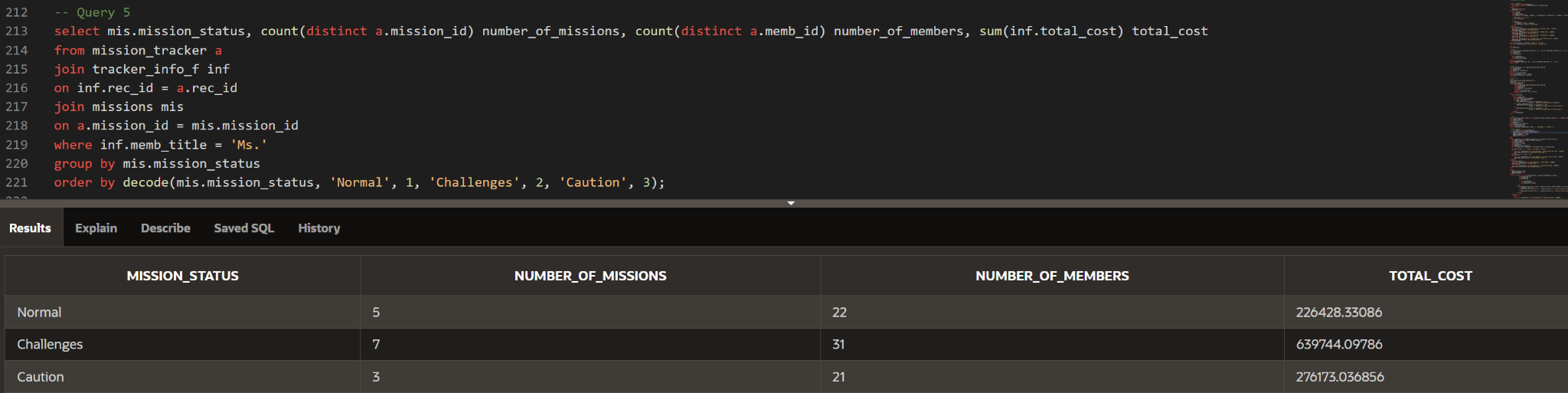
Additionally to this, I wrote an anonymous block of code with an explicit cursor that makes a written report regarding every mission, providing the number of members involved in each one with a written message of whether the crew size is large enough according to the recommendations.





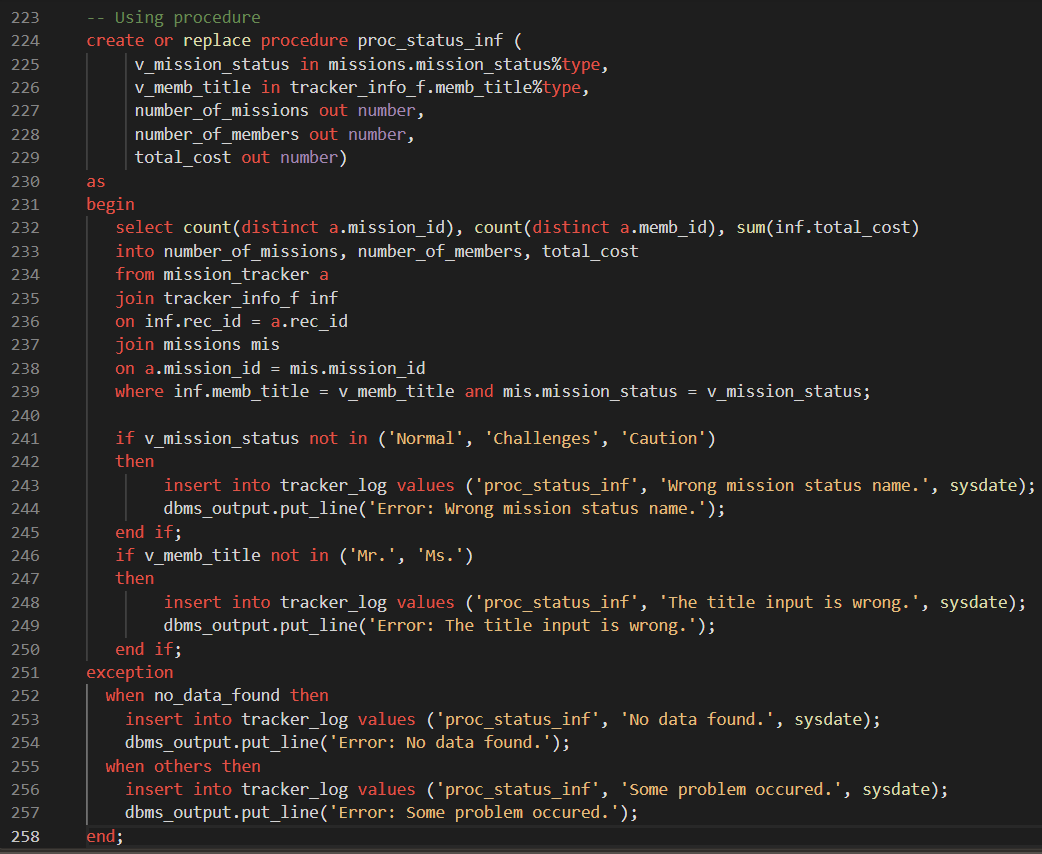
**Pic. 10, 11 - PL/SQL anonymous block with explicit cursor with results**

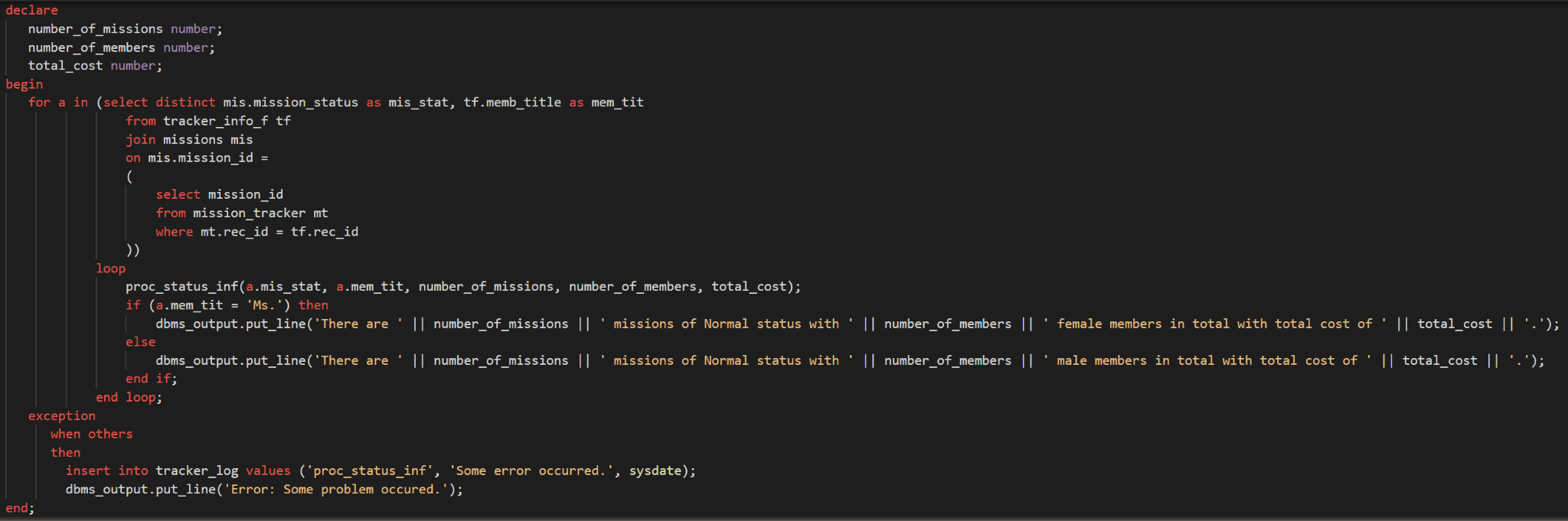
The next query aggregates all information that could be helpful grouped by the mission status. For each of the groups of status, the number of corresponding missions, members involved and total expenses are demonstrated. The additional restriction displays only those members who are women.

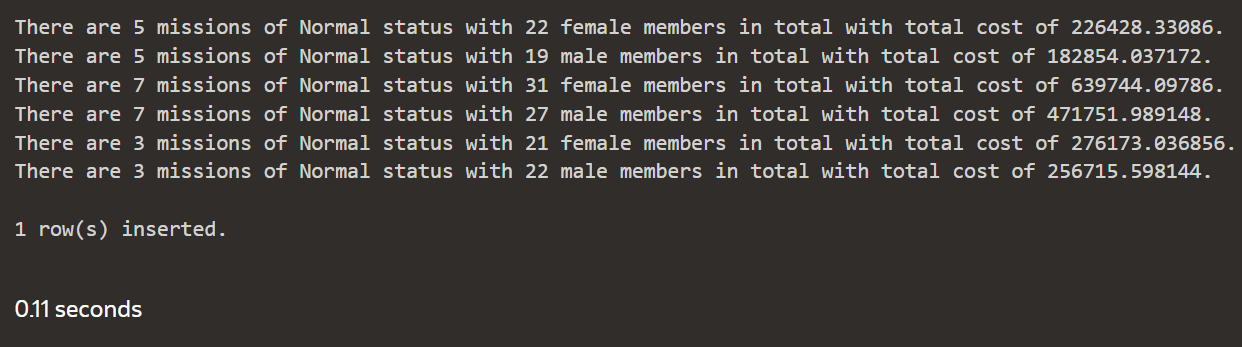


**Pic. 12- The multiple aggregations and calculations query and result**

Also, the procedure was created to count the number of missions, number of members and the total cost for each mission type based on the gender of participants. PLus, it has all required exception handling and error catching. Then, the statement was written to use the created procedure with every combination of mission status and gender using implicit cursor to create a written report about every mission type.

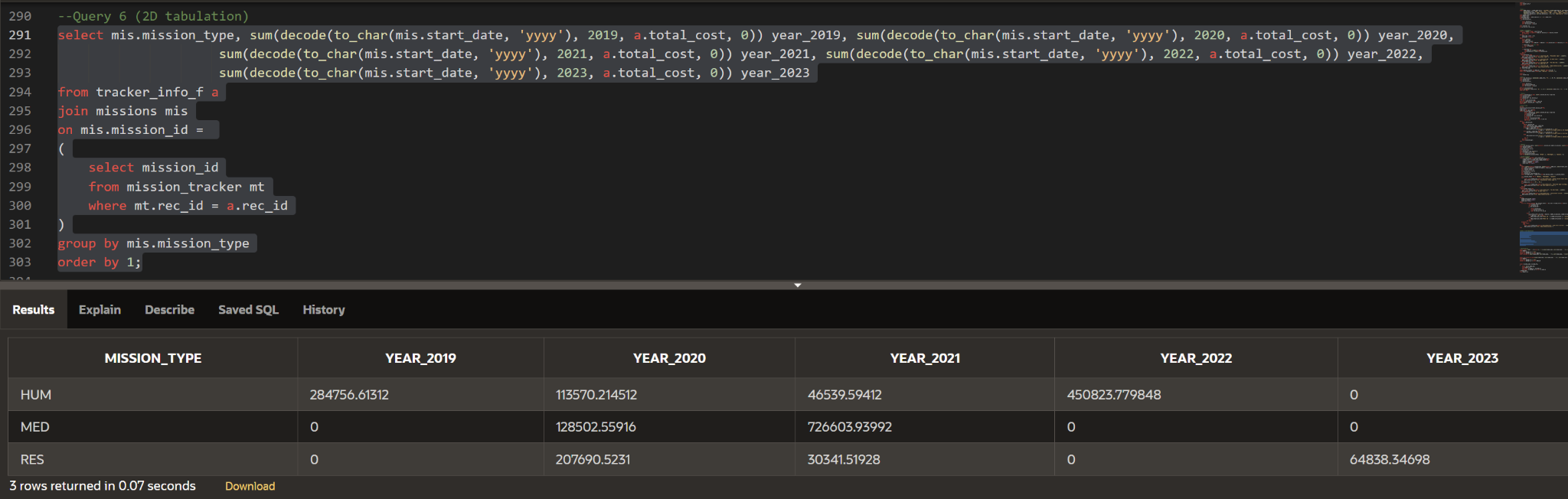






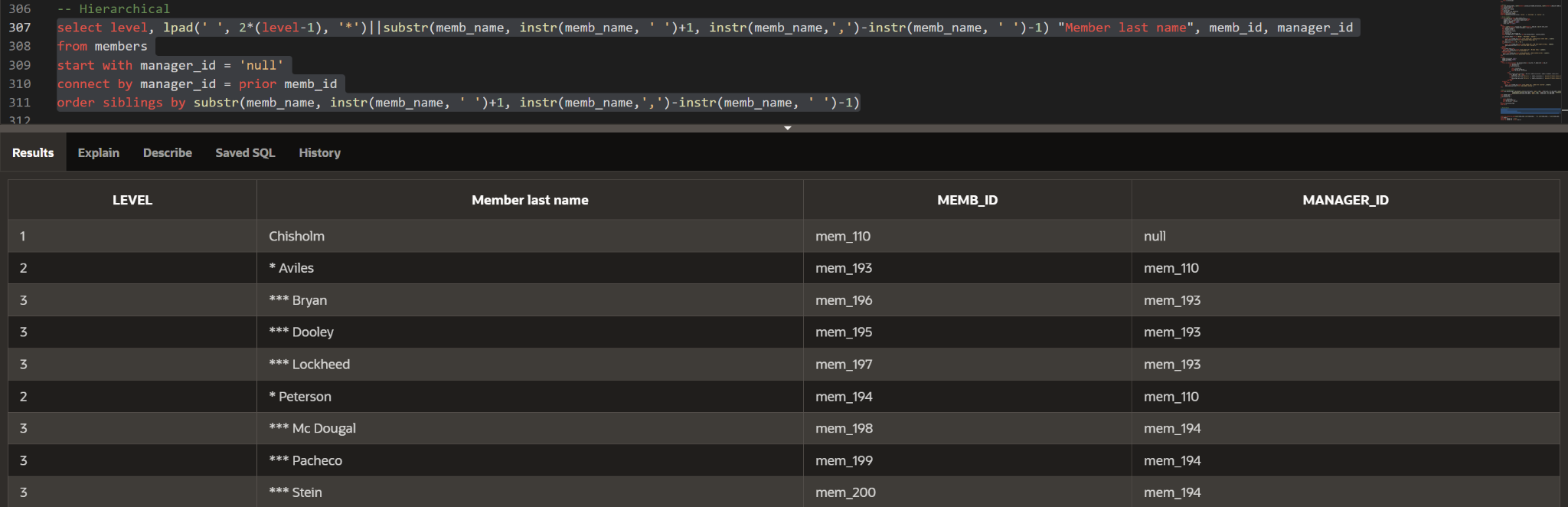
**Pic. 13, 14, 15- Procedure with its implementation and results**

Moreover, I created a query that displays the total costs for each type of missions grouped by a year these missions started. This information can help investors and the government to make a decision about further donations to a particular mission, as well as it helps the coordinators to plan their human resources more wisely.



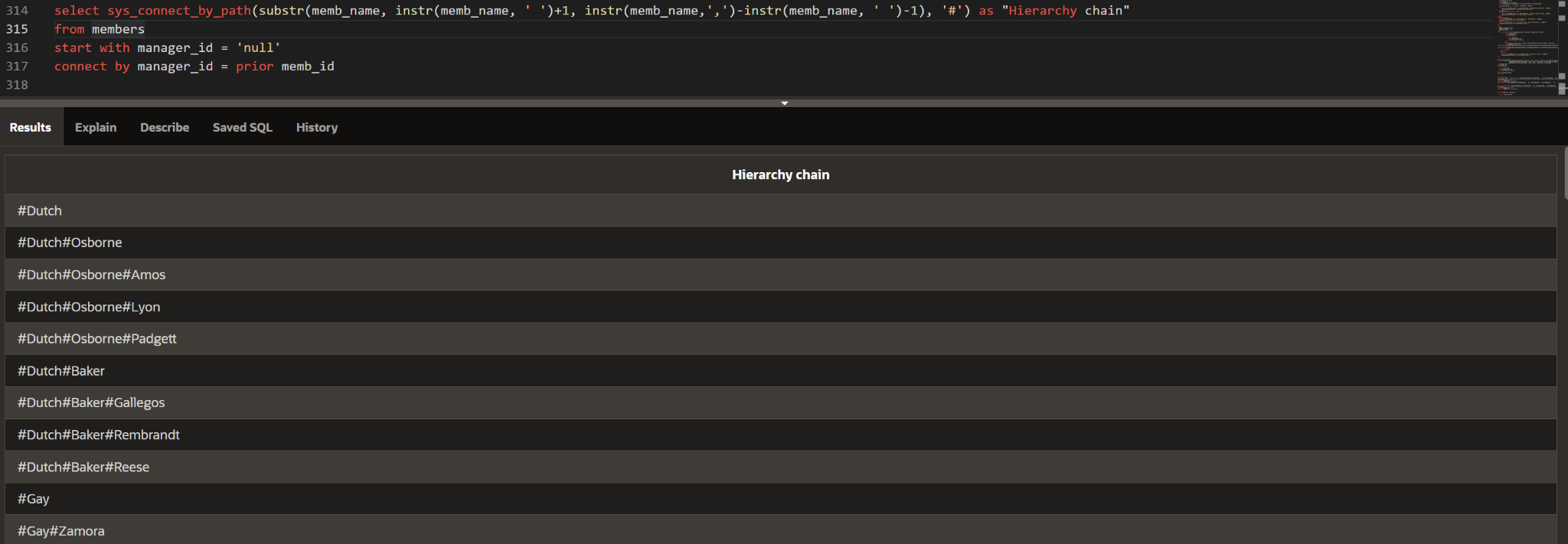
**Pic. 16 - The total costs grouped by the year query and result**

Last, but not least, additional hierarchical queries were created. It is used for better understanding of the inner structure of working crews and can help in better organization of those on the working sites. Firstly, let’s see the correlation between members in a table view. It will state both the level of the member, as well the number of their direct supervisor.



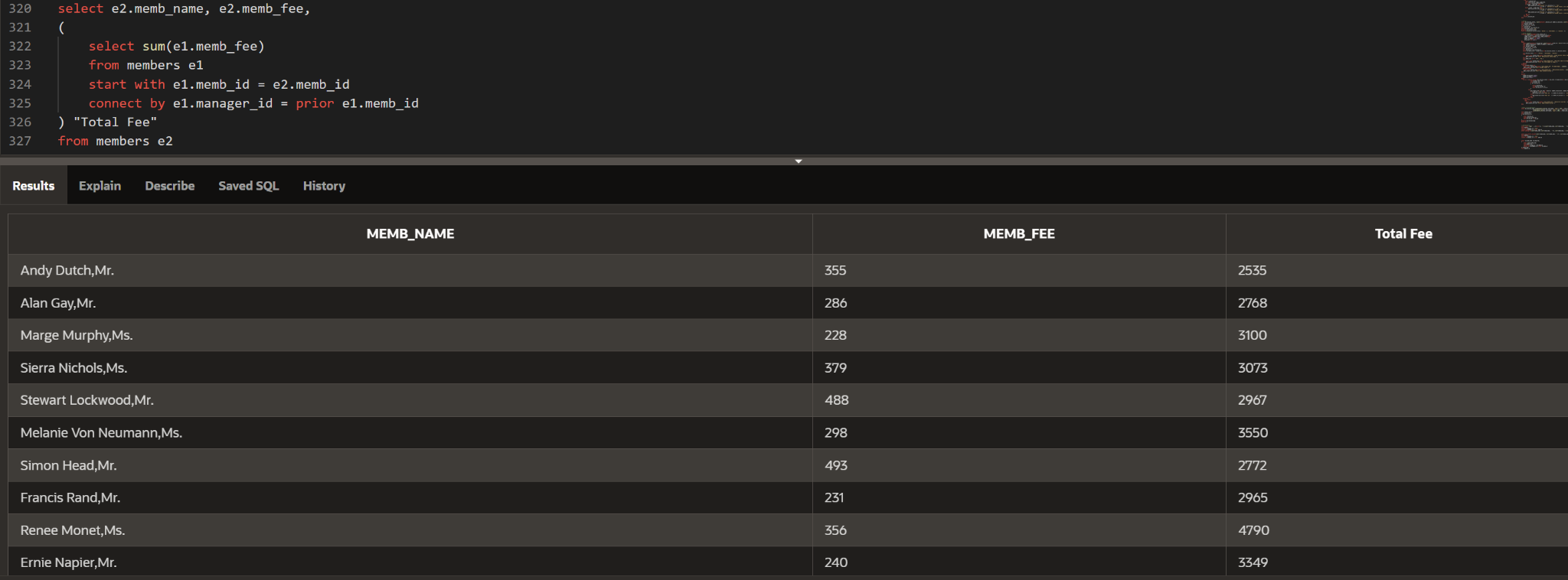
**Pic. 17 - Hierarchical table and results**

Another implication will be with the use of sys\_connect\_by\_path command to perform the hierarchical chains for each member.



**Pic. 18 - Hierarchical chains and result**

Finally, I will use a hierarchical query as a subquery to calculate the total fee for each member and every member under his/her supervision.



**Pic. 19 - Hierarchical subquery and result**

**Suggestions**

These queries and information acquired can help the managers and coordinators to plan their logistics. However, there are a lot of things to be developed and considered further. Thus, the calculation of expenses was done roughly and not very precised to the real market amounts. Additionally, for a better view and understanding, multiple views for different tasks could be created.

Additional attention should be paid to the data inserted. More triggers can be created to ensure the consistency of data present, especially when updated/inserted. Various functions for members/records deletions can be implemented to make it easier to operate the existing data. Moreover, some triggers for updates can be written to automate the process of additional calculations on air.

The tables were created in order to divide different pieces of information to make it more convenient and flexible to work with. However, the methods of dividing the initial data into multiple ones could vary according to the needs of the company. Moreover, in most cases we do not need to see all the data from the created view, so we can display the initial table only, but aggregate and sort our results based on conditions related to the expanded view, so the data will be correctly analyzed and aggregated.

Such techniques as case, subselect and multiple united cursors can be added to make application more efficient and resourceful.

Lastly, the random data created in Excel files should be taken carefully as not everything can be precise.