Laboratory 07

Jeffrey Daniel Leiva Cascante
Tamara Nicole Rodríguez Luna
Computer engineering department, Tecnológico de Costa Rica

IC4301 – Databases I

Teacher: Msc. Adriana Álvarez Figueroa.

October 07th 2022.

In this document are the evidence of laboratory 07 where you can see the objects created, data inserted, modified, among others.
Evidence:
1. 1. Investigate the parameters of the dbms_scheduler.create_job procedure, list them and indicate what each one means.

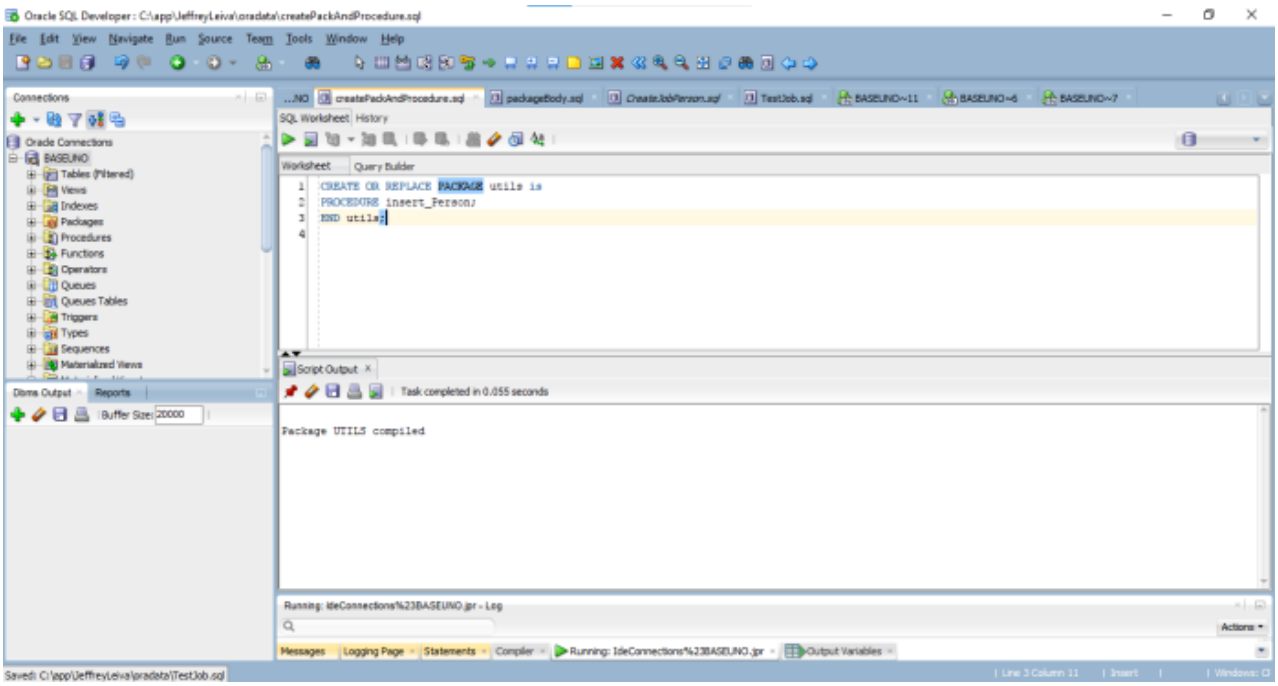
Parameter	Description
Job_name	This attribute specifies the name of the job and uniquely identifies the job. The name has to be unique in the SQL namespace. For example, a job cannot have the same name as a table in a schema.
Job_type	This attribute specifies the type of job that you are creating. If it is not specified, an error is generated. The three supported values are: plsql_block This specifies that the job is an anonymous PL/SQL block. Job or program arguments are not supported when the job or program type is plsql_block. In this case, the number of arguments must be 0.
Job_action	This attribute specifies the action of the job. The following actions are possible: For a PL/SQL block, the action is to execute PL/SQL code. These blocks must end with a semi-colon. For example, my_proc(); or BEGIN my_proc(); END; or DECLARE arg pls_integer := 10; BEGIN my_proc2(arg); END;. Note that the Scheduler wraps job_action in its own block and passes the following to PL/SQL for execution: DECLARE ... BEGIN job_action END; This is done to declare some internal Scheduler variables.
number_of_arguments	This attribute specifies the number of arguments that the job expects. The range is 0-255, with the default being 0.
program_name	The name of the program associated with this job.
start_date	This attribute specifies the first date on which this job is scheduled to start. If start_date and repeat_interval are left null, then the job is scheduled to run as soon as

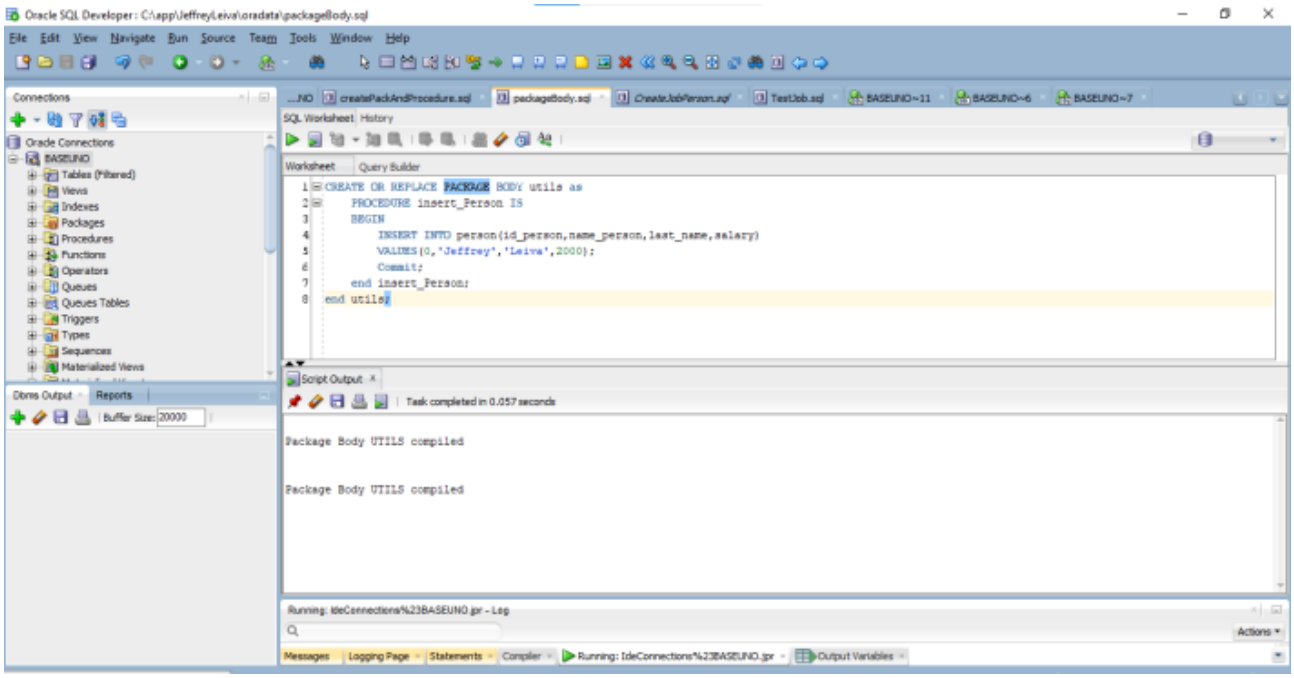
	the job is enabled.
repeat_interval	This attribute specifies how often the job should repeat. You can specify the repeat interval by using calendaring or PL/SQL expressions. The expression specified is evaluated to determine the next time the job should run. If repeat_interval is not specified, the job will run only once at the specified start date.
schedule_name	The name of the schedule, window, or window group associated with this job.
end_date	This attribute specifies the date after which the job will expire and will no longer be executed. When end_date is reached, the job is disabled. The STATE of the job will be set to COMPLETED, and the enabled flag will be set to FALSE.
job_class	This attribute specifies the job class that the job belongs to. If no job class is specified, then the job is assigned to the default class. Note that the owner of a job must have EXECUTE privileges on a job class in order to run a job using the resources of that class. If an invalid value for job_class is specified, an error is generated.
comments	This attribute specifies a comment about the job. By default, this attribute is NULL.
enabled	This attribute specifies whether the job is created enabled or not. The possible settings are TRUE or FALSE. By default, this attribute is set to FALSE and, therefore, the job is created as disabled. A disabled job means that the metadata about the job has been captured and the job exists as a database object but the Scheduler will ignore it and the job coordinator will not pick the job for processing. In order for the job coordinator to process the job, the job has to be enabled. You can enable a job by setting this argument to TRUE or by using the ENABLE procedure.
auto_drop	This flag specifies whether the job will be automatically dropped once it has been executed for non-repeating jobs or when its status is changed to COMPLETED for repeating jobs. The metadata is removed from the database if this flag is set to TRUE.

	If this flag is set to FALSE, the jobs are not dropped and their metadata is kept until the job is explicitly dropped and can be queried using the *_SCHEDULER_JOBS views. This metadata can be removed from the table using the DROP_JOB procedure. By default, jobs are created with auto_drop set to TRUE.

(Corporation, s. f.)

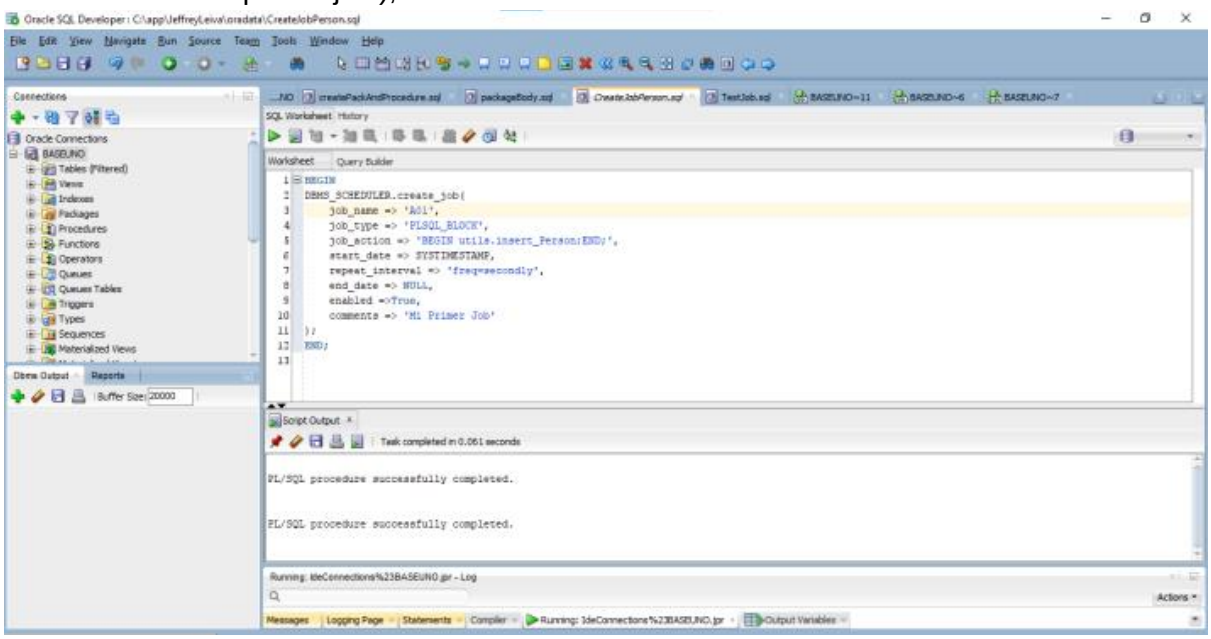
2. Create a package that contains a procedure that registers employees in the table employee.

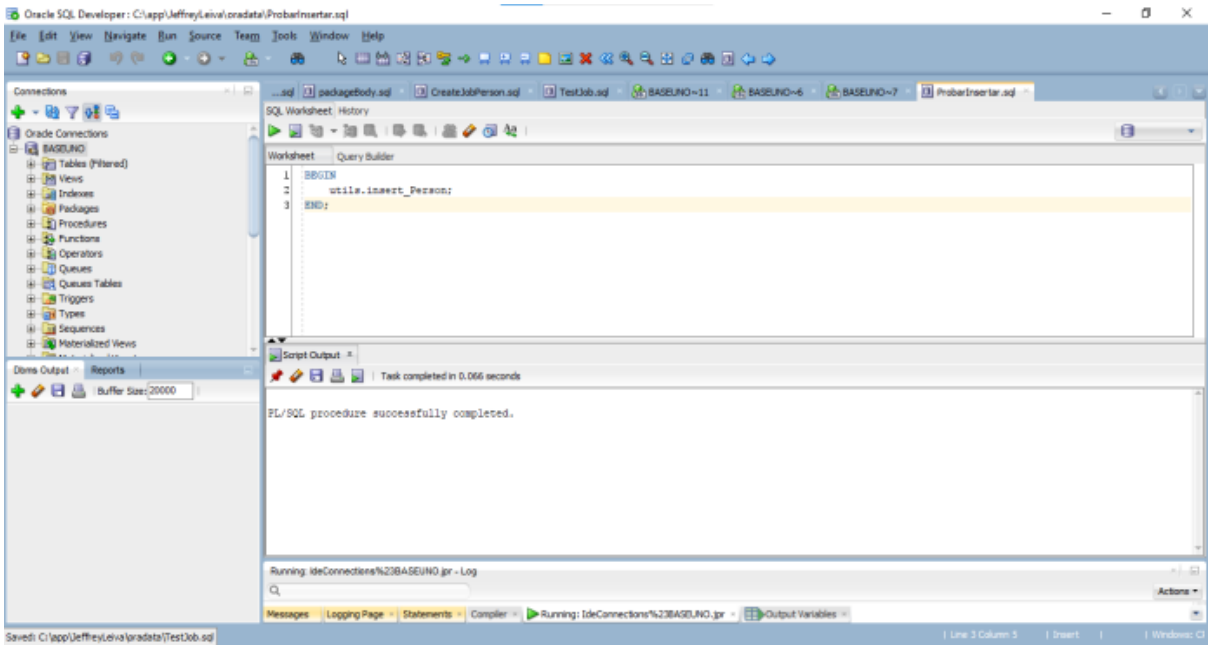




3. Create a test to call the procedure as a job and run it. Base on the following code.

```
DBMS_SCHEDULER.create_job (
job_name => 'A01',
job_type => 'PLSQL_BLOCK',
job_action => 'BEGIN utils.newEmployee;
END;',
start_date => SYSTIMESTAMP,
repeat_interval => 'freq=secondly',
end_date => NULL,
 enabled => TRUE,
 comments => 'Mi primer job');
```

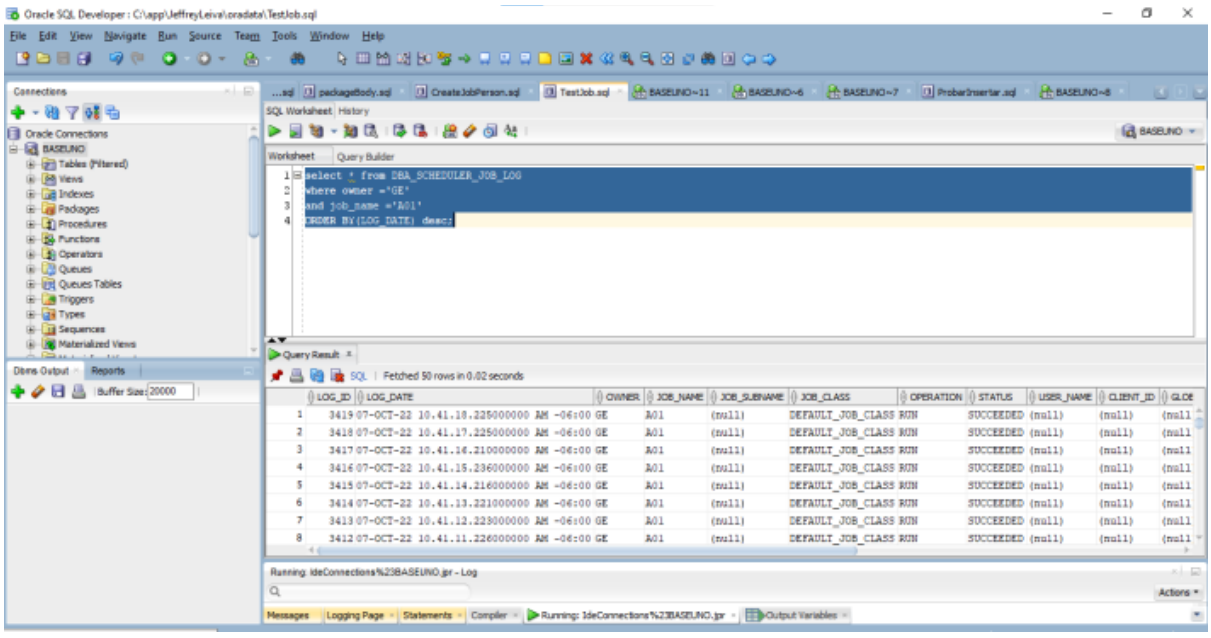




4.Make the following query and validate that your job is running. Set the select to values that you have (owner, job_name). Evidence with an image. 10pts

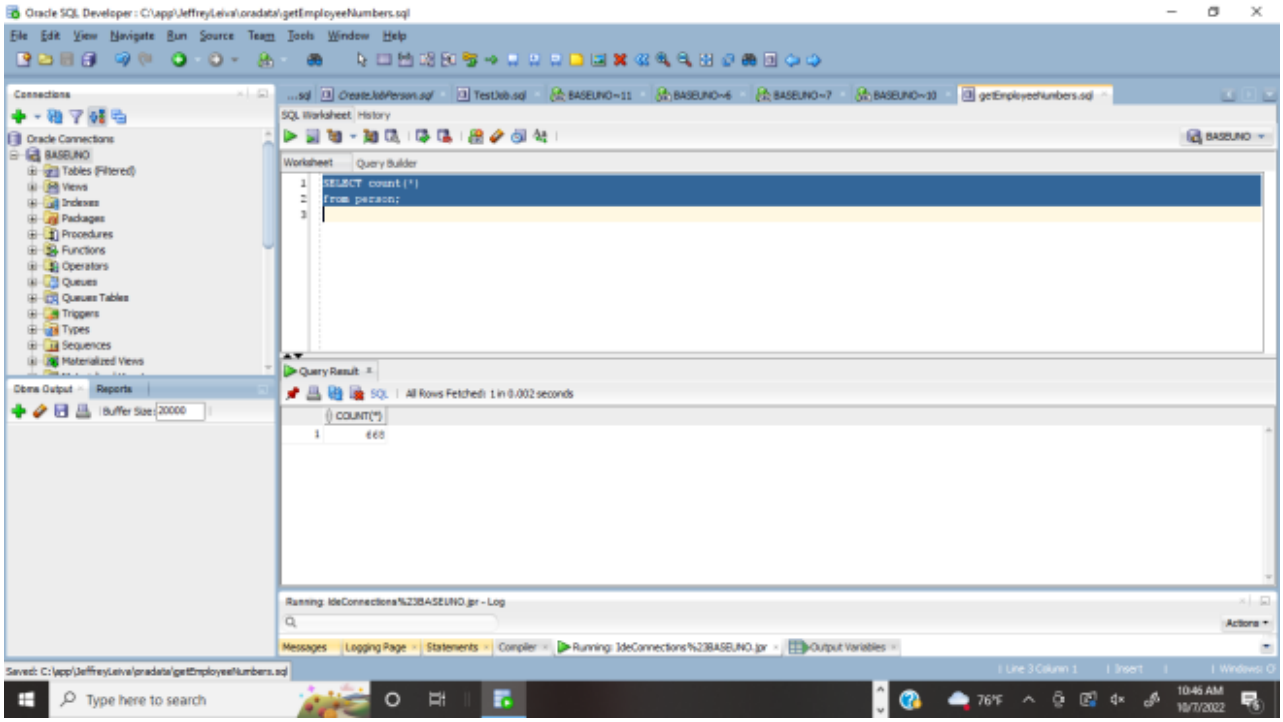
select * from DBA_SCHEDULER_JOB_LOG where owner ='GE' and job_name = 'A01'

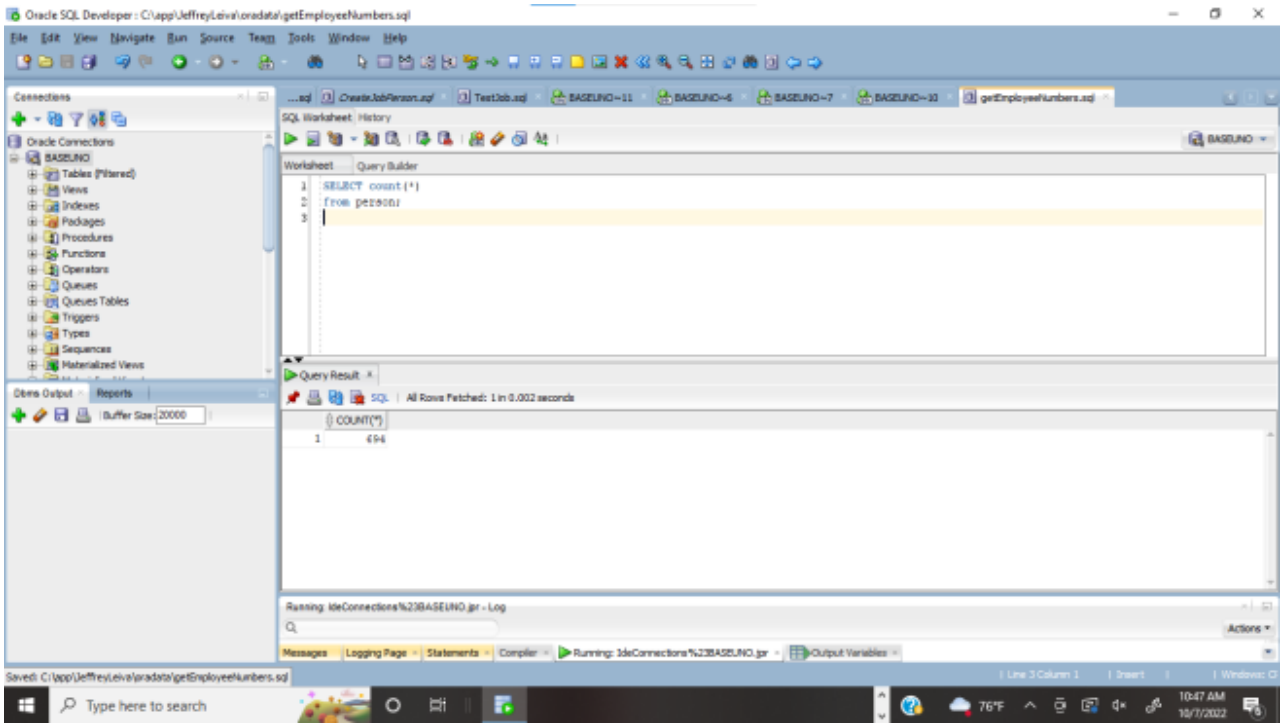
a. Capture the screen showing that your job is running correctly.

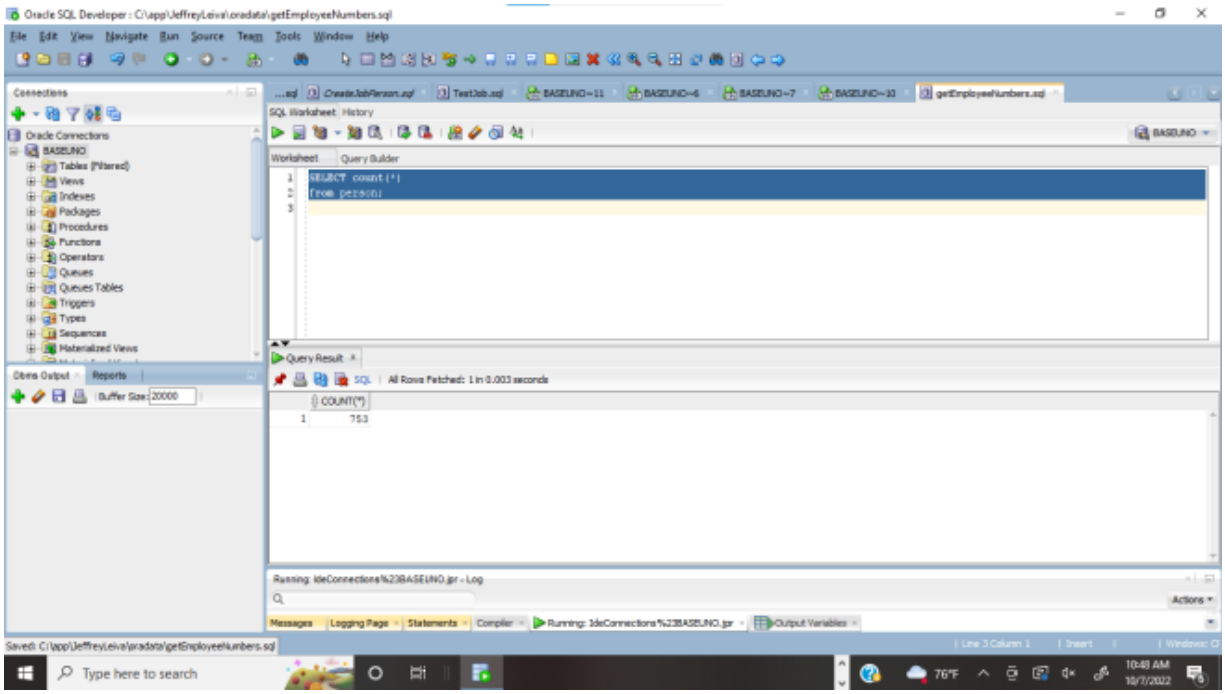


5. Write a sql to show that the number of employees is increasing in the table and demonstrate it with a screenshot 5pts

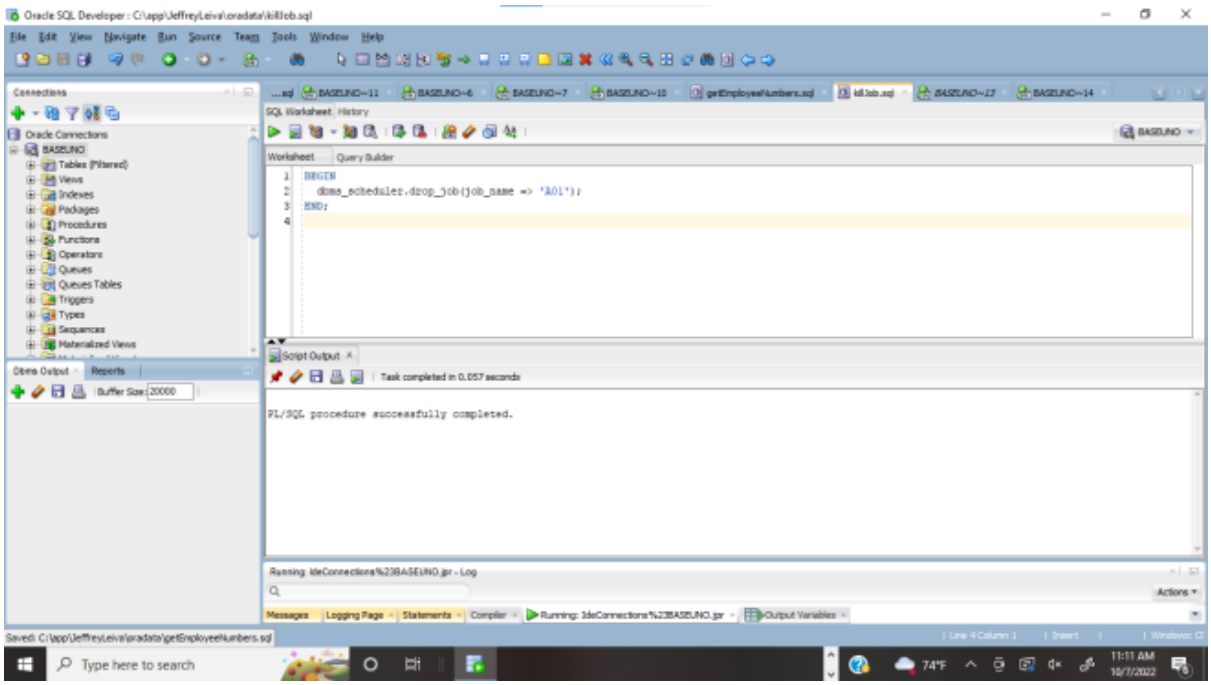
Next there are three screenshots executing the getEmployeeNumbers







6. Research how to kill a job. Execute the statement and add an image with the evidence.



Bibliographic reference:
Corporation, O. (s. f.). DBMS_SCHEDULER. Oracle Corporation. Recuperado 7 de octubre de 2022, de https://docs.oracle.com/cd/B14117_01/appdev.101/b10802/d_sched.htm