Guillaume Minette de Saint Martin

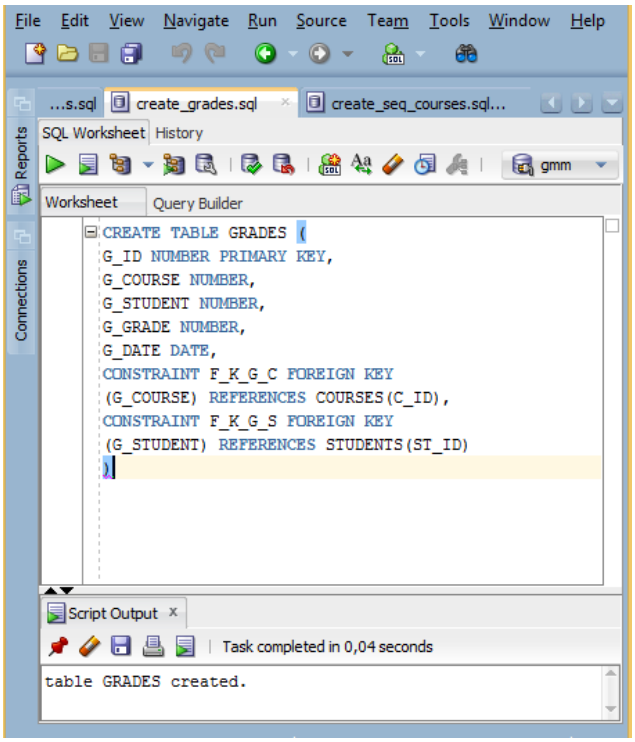
Advanced data Technologies

Lab 1

1. Create tables to store data about the learning process of the university.

The following files are used to create tables:

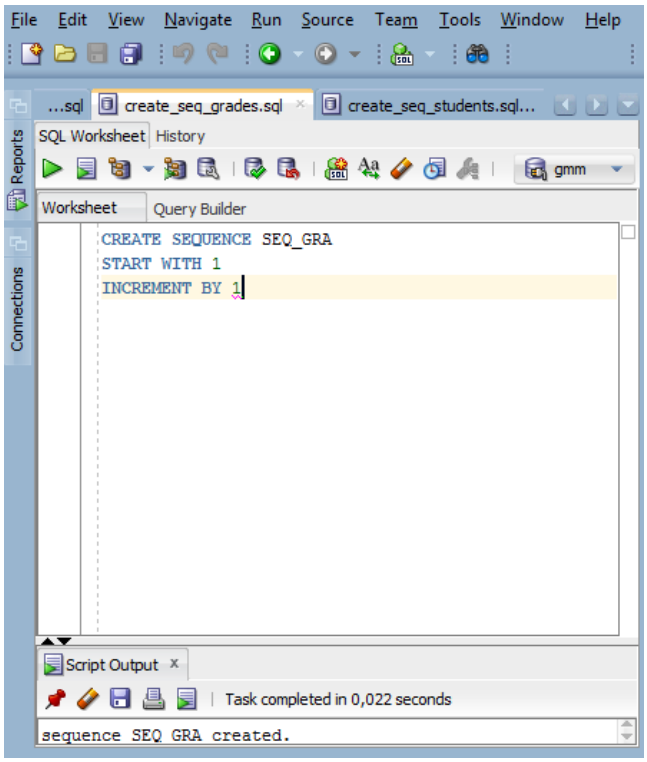
- create_courses.sql
- create_grades.sql
- create_students.sql
- create_teachers.sql



2. Create sequences needed to fill in the primary key fields of the tables.

The following files are used to create sequences:

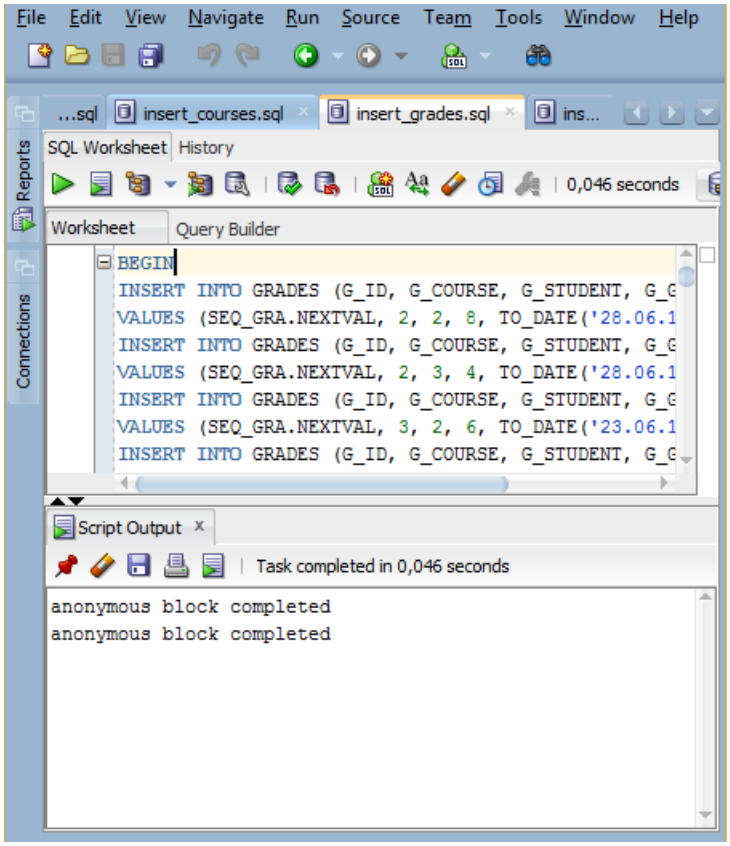
- create_seq_courses.sql
- create_seq_grades.sql
- create_seq_students.sql
- create_seq_teachers.sql



3. Insert data into tables you previously created, using the sequences defined above.

The following files are used to insert data into tables:

- insert_courses.sql
- insert _grades.sql
- insert _students.sql
- insert _teachers.sql



4. Create SELECT statements that complete the following tasks:

For the following results, output screenshots are made with the French version of the software interface, I hope it will not be a problem.

Query file containing the following queries is ***querry.sql***.

1. (Easy) Retrieve all courses that are smaller than 4 CP.

```
select *
from courses
where C_C_POINTS<4;
```

Just filter rows according to their CP value.

SQL | Toutes les lignes extraites : 2 en 0,374 secondes

	C_ID	C_TITLE	C_TEACHER	C_C_POINTS
1	5	Grid Networking	5	2
2	7	Enterprise Architecture	6	2

2. (Easy) Retrieve names of the courses together with the names of their teachers.

```
select C_TITLE, T_NAME

from courses, teachers

where C_TEACHER=T_ID;
```

Join the two tables

SQL | Toutes les lignes extraites : 6 en 0,042 secondes

	C_TITLE	T_NAME
1	Advanced Data Technologies	Lavendelis
2	AI in Business	Lavendelis
3	Systems Theory	Grundspenkis
4	Knowledge Engineering	Kirikova
5	Grid Networking	Zagurskis
6	Enterprise Architecture	Grabis

3. (Easy) Retrieve information needed in student's individual plan (teacher, name of the course and grade for each course that the student has passed). Do it for any student ID number.

```
select ST_ID, T_NAME, C_TITLE, G_GRADE

from courses, teachers, grades, students

where C_TEACHER=T_ID and ST_ID=G_STUDENT and G_COURSE=C_ID;
```

Join tables and display needed columns

SQL | 50 lignes extraites en 0,048 secondes

	ST_ID	T_NAME	C_TITLE	G_GRADE
1	2	Lavendelis	Advanced Data Technologies	8
2	3	Lavendelis	Advanced Data Technologies	4
3	2	Grundspenkis	Systems Theory	6
4	3	Grundspenkis	Systems Theory	7
5	4	Grundspenkis	Systems Theory	9
6	5	Grundspenkis	Systems Theory	3
7	6	Grundspenkis	Systems Theory	10

4. (Medium) Retrieve all students and sort them according to the average grade, so that the best student is shown first.

```
select ST_ID, AVG(G_GRADE)
```

```
from students, grades
where ST_ID=G_STUDENT group by ST_ID order by AVG(G_GRADE) desc;
```

Display with a descendant sorting

SQL | Toutes les lignes extraites : 7 en 0,041 secondes

	ST_ID	AVG(G_GRADE)
1	8	9
2	4	7,8
3	7	7,4
4	2	7
5	3	7
6	6	6,6666666666666666666666666666666666667
7	5	6,2

5. (Medium) Find all students that have passed all (6) exams and received at least 4.

```
select ST_ID from students, grades
where ST_ID=G_STUDENT and G_GRADE>=4
group by ST_ID
having count(G_ID)=6;
```

Group by student and count the number of passed exams for each of them

SQL | Toutes les lignes extraites : 1 en 0,034 secondes

	ST_ID
1	5

6. (Hard) Calculate the average marks of each teacher and each course. Do it in the same query!

```
select T_ID, C_ID, avg(G_GRADE)
from teachers, courses, grades
where G_COURSE=C_ID and C_TEACHER=t_id
group by cube(T_ID,C_ID)
having(( grouping(T_ID)=1 or grouping(C_ID)=1) and (grouping(T_ID)=0 or
grouping(C_ID)=0));
```

First get all the averages and then eliminate these that are not needed by only displaying the needed ones.

SQL | Toutes les lignes extraites : 11 en 0,108 secondes

	T_ID	C_ID	AVG(G_GRADE)
1	(null)	6	8
2	(null)	2	6,8
3	(null)	4	7
4	(null)	5	7,1428571428571428571428571428571428571 4
5	(null)	3	6,7142857142857142857142857142857142857 1
6	(null)	7	8,1666666666666666666666666666666666666 7
7	6	(null)	8,1666666666666666666666666666666666666 7
8	2	(null)	7,5
9	4	(null)	7
10	5	(null)	7,1428571428571428571428571428571428571 4
11	3	(null)	6,7142857142857142857142857142857142857 1

7. (Hard) Retrieve all students whose average mark is higher than the average mark of the student with ID "*061RDB121*".

```
select s.ST_ID, avg(g.G_GRADE)
from students s, grades g
where s.ST_ID=g.G_STUDENT
group by s.ST_ID
having avg(g.G_GRADE)>(
        select avg(r.G_GRADE)
        from grades r, students t
        where t.ST_ID=r.G_STUDENT and t.ST_ID_NUM='061RDB121'
);
```

Firs join tables, group by student and filter to only display these that have an average mark higher than one in particular

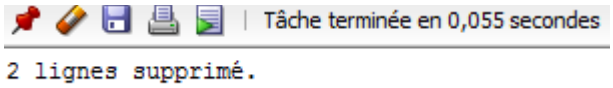
SQL | Toutes les lignes extraites : 1 en 0,067 secondes

	ST_ID	AVG(G.G_GRADE)
1	8	9

8. (Easy) Delete any grade from the table GRADES, by specifying student and course.

```
delete from grades
where G_STUDENT='2' and G_COURSE='4';
```

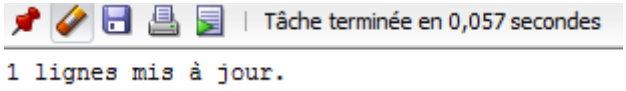
Simple delete querry



9. (Easy) Change students ID number of any student.

```
update students
set ST_ID_NUM='xxxxxxxx' where ST_ID_NUM='051RDB131';
```

Simple update and set querry



10. (Easy) Add one column to any table.

```
ALTER TABLE STUDENTS
ADD ST_FRANCAIS INTEGER;
```

Add column ST_FRANCAIS of type integer to the student table