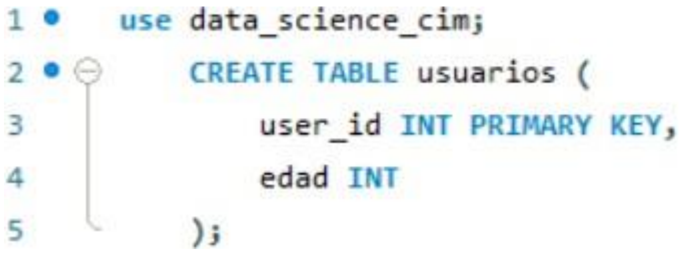
MySQL Workbench

1. A database named **data_science_cim** is created and then the two tables are created.

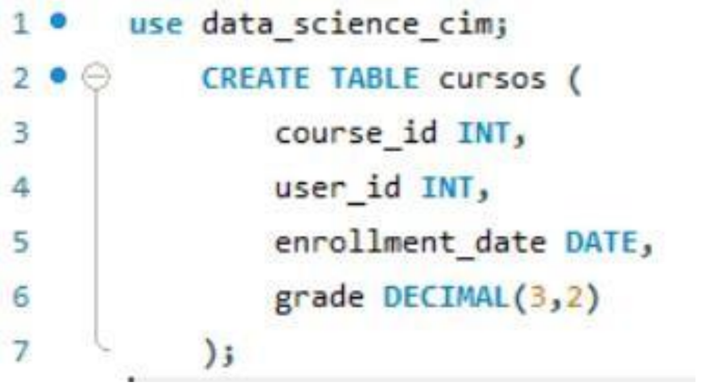
- **Table1:** *users*



```sql
use data_science_cim;
    CREATE TABLE usuarios (
        user_id INT PRIMARY KEY,
        edad INT
    );
```

Fig.1 *User table creation*

- **Table 2:** *courses*

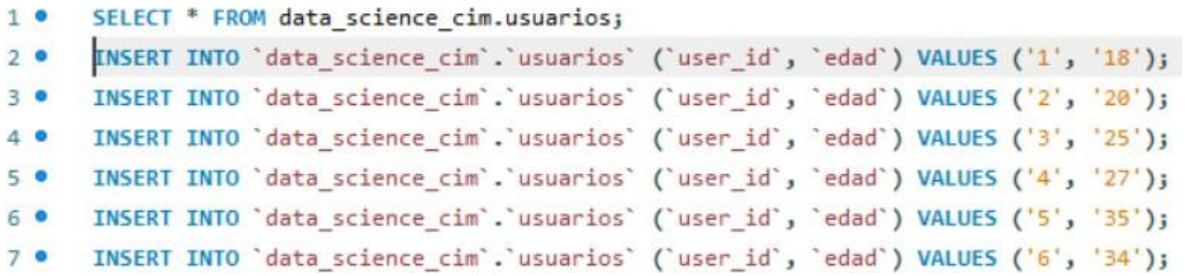


```sql
use data_science_cim;
    CREATE TABLE cursos (
        course_id INT,
        user_id INT,
        enrollment_date DATE,
        grade DECIMAL(3,2)
    );
```

Fig.2 *Course table creation*

2. The data are randomly inserted into the two tables:

- **Table1:** *users*

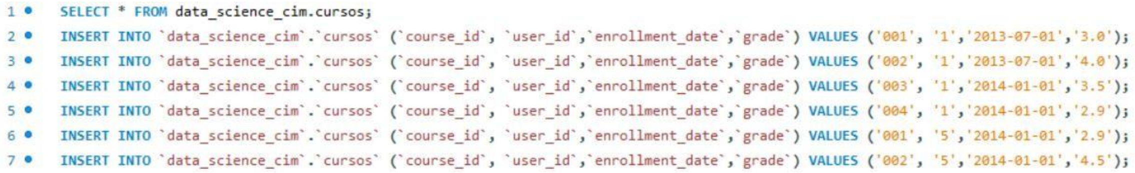


```sql
SELECT * FROM data_science_cim.usuarios;
INSERT INTO `data_science_cim`.`usuarios` (`user_id`, `edad`) VALUES ('1', '18');
INSERT INTO `data_science_cim`.`usuarios` (`user_id`, `edad`) VALUES ('2', '20');
INSERT INTO `data_science_cim`.`usuarios` (`user_id`, `edad`) VALUES ('3', '25');
INSERT INTO `data_science_cim`.`usuarios` (`user_id`, `edad`) VALUES ('4', '27');
INSERT INTO `data_science_cim`.`usuarios` (`user_id`, `edad`) VALUES ('5', '35');
INSERT INTO `data_science_cim`.`usuarios` (`user_id`, `edad`) VALUES ('6', '34');
```

Fig.3 *Assignment of data to user table (50 users)*

50 rows are generated with random information as shown in the previous image, with the ISERT INTO command in the users table, column names and column values.

- **Table 2:** *courses*



```
SELECT * FROM data_science_cim.cursos;
INSERT INTO `data_science_cim`.`cursos` (`course_id`, `user_id`,`enrollment_date`,`grade`) VALUES ('001', '1','2013-07-01','3.0');
INSERT INTO `data_science_cim`.`cursos` (`course_id`, `user_id`,`enrollment_date`,`grade`) VALUES ('002', '1','2013-07-01','4.0');
INSERT INTO `data_science_cim`.`cursos` (`course_id`, `user_id`,`enrollment_date`,`grade`) VALUES ('003', '1','2014-01-01','3.5');
INSERT INTO `data_science_cim`.`cursos` (`course_id`, `user_id`,`enrollment_date`,`grade`) VALUES ('004', '1','2014-01-01','2.9');
INSERT INTO `data_science_cim`.`cursos` (`course_id`, `user_id`,`enrollment_date`,`grade`) VALUES ('001', '5','2014-01-01','2.9');
INSERT INTO `data_science_cim`.`cursos` (`course_id`, `user_id`,`enrollment_date`,`grade`) VALUES ('002', '5','2014-01-01','4.5');
```

Fig.4 *Assignment of data to course table (50 courses)*

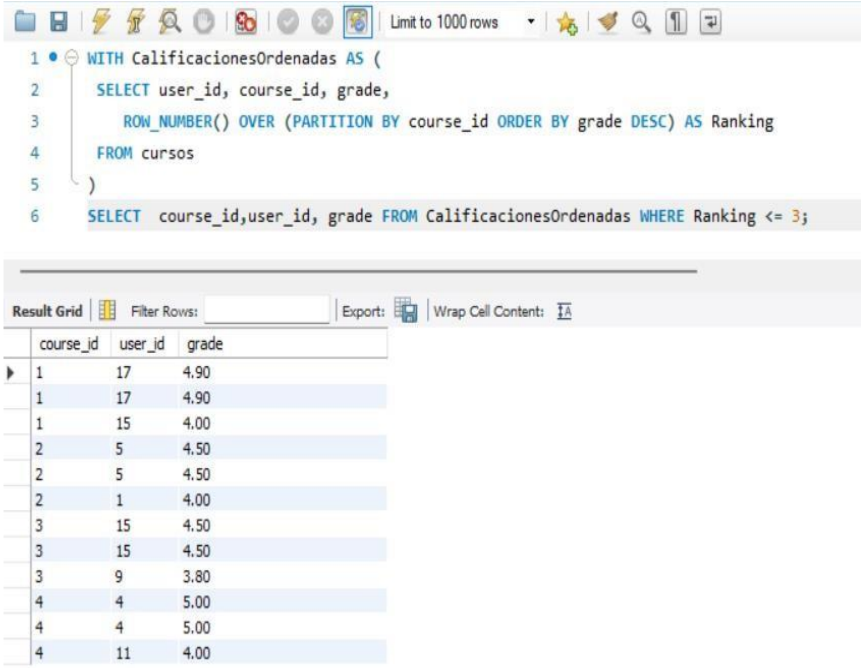
50 rows are generated with random information as shown in the image above, with the ISERT INTO command in the table courses, columns name and column values. We take into account 4 courses and a user registers to the 4 courses, 1 user maximum 2 times to the same course.

EXERCISE:

a. Create a query that shows the top 3 users per course based on grade.

We use the function "ROW_NUMBER" which assigns a unique number to each row within each course, which sorts the rating in descending order (first row is the highest ranking) and finally we select only those rows where the ranking is less than 3 users.

Result: see attached file "**resultado_punto_a**" and "**ejercicio_3(a)_sql.sql**".



```
WITH CalificacionesOrdenadas AS (
  SELECT user_id, course_id, grade,
     ROW_NUMBER() OVER (PARTITION BY course_id ORDER BY grade DESC) AS Ranking
  FROM cursos
)
SELECT  course_id,user_id, grade FROM CalificacionesOrdenadas WHERE Ranking <= 3;
```

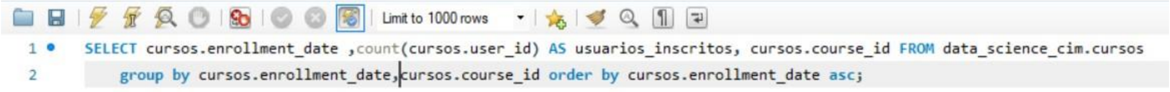
course_id	user_id	grade
1	17	4.90
1	17	4.90
1	15	4.00
2	5	4.50
2	5	4.50
2	1	4.00
3	15	4.50
3	15	4.50
3	9	3.80
4	4	5.00
4	4	5.00
4	11	4.00

Fig.5 *Result of the exercise (a) in MSQL workbench*

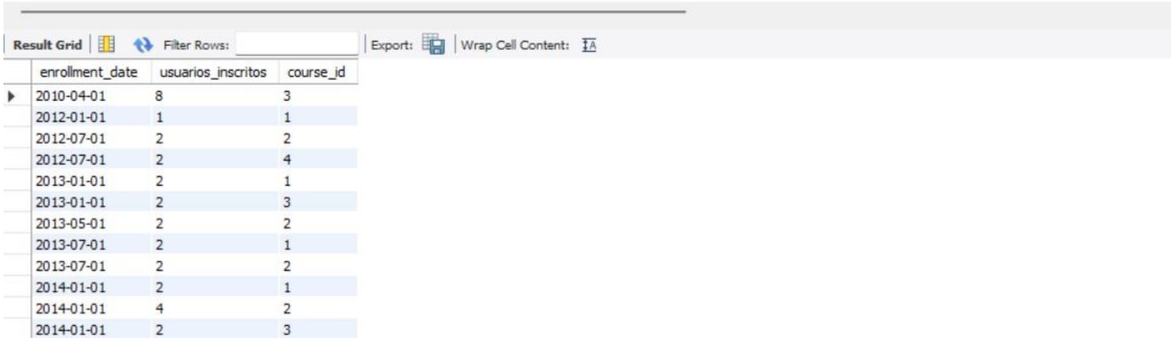
b. Create a query that shows the number of users enrolled date by date in each course.

the command was executed where the "*enrollment_date"* column is selected from the *courses* table, the "*course_id"* column is selected, and a new column called "*enrolled_users*" is created, which will have the user_id count values grouped according to the "enrollment_date" and "course_id" columns, sorting the data set in ascending order from the "enrollment" column.

Result: see attached file "**resultado_punto_b**" and “**ejercicio_3(b)_sql.sql**”.



```
SELECT cursos.enrollment_date ,count(cursos.user_id) AS usuarios_inscritos, cursos.course_id FROM data_science_cim.cursos
    group by cursos.enrollment_date,cursos.course_id order by cursos.enrollment_date asc;
```



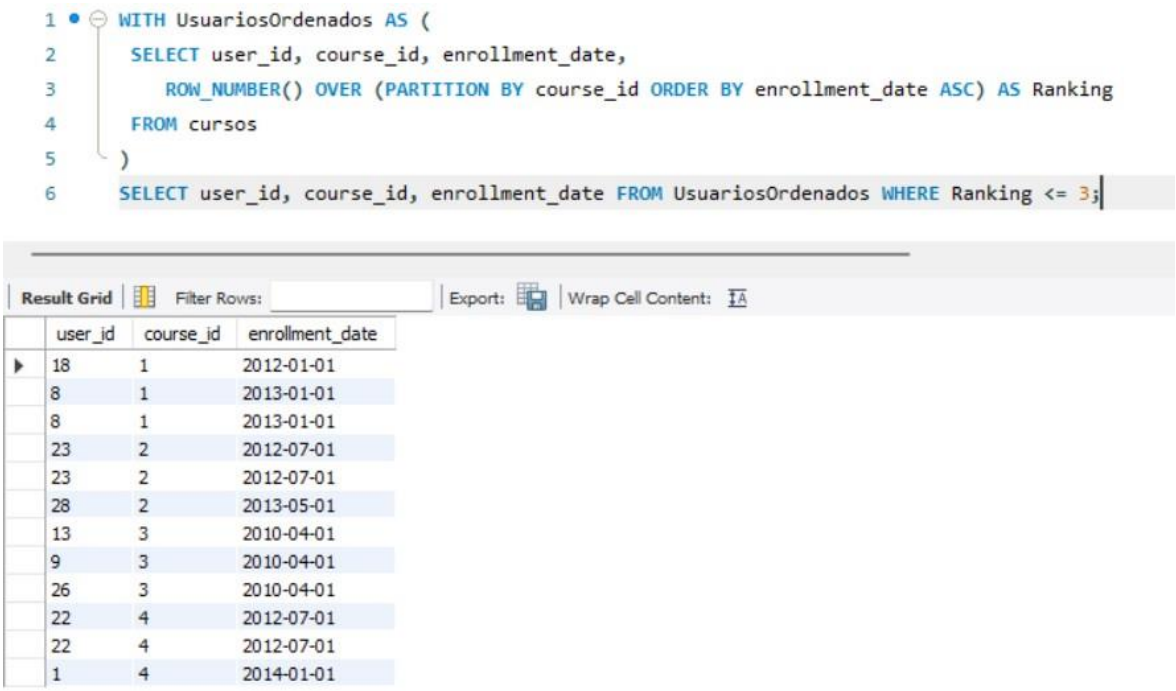
enrollment_date	usuarios_inscritos	course_id
2010-04-01	8	3
2012-01-01	1	1
2012-07-01	2	2
2012-07-01	2	4
2013-01-01	2	1
2013-01-01	2	3
2013-05-01	2	2
2013-07-01	2	1
2013-07-01	2	2
2014-01-01	2	1
2014-01-01	4	2
2014-01-01	2	3

Fig.6 *Exercise result (b) in MSQL workbench*

c. Create a query that shows the 3 oldest users in each course.

In this query, the "ROW_NUMBER ()" clause, the same as used in (a) above, is used to assign a unique number to each row within each course, sorted by enrollment date in ascending order. This means that the user who enrolled first in each course will have a ranking of 1. Then, we select only those rows where the ranking is less than or equal to 3, which will give us the three oldest users in each course.

Result: see attached file "**resultado_punto_c**" and “**ejercicio_3(c)_sql.sql**”.



```sql
WITH UsuariosOrdenados AS (
 SELECT user_id, course_id, enrollment_date,
    ROW_NUMBER() OVER (PARTITION BY course_id ORDER BY enrollment_date ASC) AS Ranking
 FROM cursos
)
SELECT user_id, course_id, enrollment_date FROM UsuariosOrdenados WHERE Ranking <= 3;
```

user_id	course_id	enrollment_date
18	1	2012-01-01
8	1	2013-01-01
8	1	2013-01-01
23	2	2012-07-01
23	2	2012-07-01
28	2	2013-05-01
13	3	2010-04-01
9	3	2010-04-01
26	3	2010-04-01
22	4	2012-07-01
22	4	2012-07-01
1	4	2014-01-01

Fig.7 *Result of exercise (c) in MSQL workbench*