

## Question 1:

1. Each grade in the "takes" relation are mapped to a grade point as follows:

Grade	Point
A+	4.2
A	4.0
A-	3.7
B+	3.5
B	3.0
B-	2.7
C+	2.3
C	2.0
C-	1.5
D	1.0

Create a table, "grade\_point" to store these mappings. (Declare primary key constraint appropriately. No need to alter the "takes" relation to refer this new relation.)

### SQL

```
create table grade_point (  
    Grade varchar(3) primary key ,  
    Point decimal(2,1) not null  
);
```

### Output log

```
university> create table grade_point (  
    Grade varchar(3) primary key ,  
    Point decimal(2,1) not null  
    )  
[2024-09-17 13:09:39] completed in 21 ms
```

## Question 2:

2. Write a single query to insert the above (Q1) mappings to the table.

### SQL

```
insert into grade_point (Grade, Point)  
values('A+',4.2),('A',4.0),('A-',3.7),('B+',3.5),('B',3.0),('B-',2.7),('C+',2.3),  
('C',2.0),('C-',1.5),('D',1.0);
```

	Grade	Point
1	A+	4.2
2	A	4.0
3	A-	3.7
4	B+	3.5
5	B	3.0
6	B-	2.7
7	C+	2.3
8	C	2.0
9	C-	1.5
10	D	1.0

#### Output log

```
university> insert into grade_point (Grade, Point)
values ('A+',4.2),('A',4.0),('A-',3.7),('B+',3.5),('B',3.0),('B-',2.7),('C+',2.3),('C',2.0),('C-',1.5),('D',1.0)
[2024-09-17 13:15:29] 10 rows affected in 8 ms
```

## Question 3:

3. Find the sum of GP of each student, using the above table. Make sure students who have got a null grade in every course are displayed with a GP of null. Display the student name and sum of GP sorted by sum of GP in descending order.

#### SQL

```
select student.name,sum(grade_point.Point) as GP
from student left outer join takes on student.ID = takes.ID
left outer join grade_point on takes.grade=grade_point.Grade
group by student.name
order by GP desc;
```

	name	GP
1	Shankar	14.0
2	Brown	8.0
3	Zhang	7.7
4	Williams	7.2
5	Levy	6.5
6	Bourikas	4.5
7	Tanaka	4.0
8	Sanchez	3.7
9	Brandt	3.0
10	Peltier	2.7
11	Chavez	2.3
12	Aoi	2.0
13	Snow	<null>

### Output log

```
university> select student.name,sum(grade_point.Point) as GP
            from student left outer join takes on student.ID = takes.ID
            left outer join grade_point on takes.grade=grade_point.Grade
            group by student.name
            order by GP desc
[2024-09-17 14:24:03] 13 rows retrieved starting from 1 in 29 ms (execution: 5 ms, fetching: 24 ms)
```

## Question 4:

4. Create a SQL function to count the number of students who have taken a course given the course id.

### SQL

```
create function num_of_enrollment(course_id varchar(8)) returns int
deterministic
begin
    declare num int;

    select count(takes.course_id) into num
        from takes
        where takes.course_id =course_id;

    return num;
end;
```

```
1 create
2     definer = H3llJoY@`%` function num_of_enrollment(course_id varchar(8)) returns int deterministic
3 begin
4     declare num int;
5
6     select count(takes.course_id) into num
7     from takes
8     where takes.course_id =course_id;
9
10    return num;
11 end;
```

### Output log

```
university> create function num_of_enrollment(course_id varchar(8)) returns int
deterministic
begin
    declare num int;

    select count(takes.course_id) into num
    from takes
    where takes.course_id =course_id;

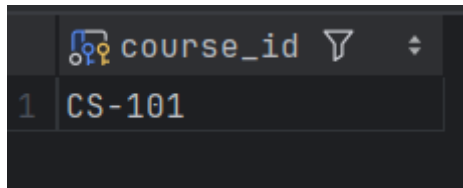
    return num;
end
[2024-09-17 14:39:31] completed in 15 ms
```

## Question 5:

5. Using the above function, find the courses that have more than 5 students enrolled. Output the course\_ids in ascending order

### SQL

```
select distinct takes.course_id
from takes
where num_of_enrollment(takes.course_id)>5
order by course_id ;
```



	course_id
1	CS-101

### Output log

```
university> select distinct takes.course_id
from takes
where num_of_enrollment(takes.course_id)>5
order by course_id
[2024-09-17 14:46:59] 1 row retrieved starting from 1 in 65 ms (execution: 4 ms, fetching: 61 ms)
```

## Question 6

6. Create a trigger to check if the grade is one of (A+, A-, A, B+, B, B-, C+, C, C-, D) before inserting values to takes relation. If the grade is not one of these values, make the grade "NULL".

### SQL

```
DELIMITER $$
CREATE TRIGGER check_grades
before insert on takes
for each row
begin
    IF NOT EXISTS(
        select 1
        from grade_point
```

```

        where grade_point.Grade = new.grade
    )then
        SET new.grade = NULL;
    end if;
end $$;
delimiter ;

```

### Output log

```

university> CREATE TRIGGER check_grades
before insert on takes
for each row
begin
    IF NOT EXISTS(
        select 1
        from grade_point
        where grade_point.Grade = new.grade
    )then
        SET new.grade = NULL;
    end if;
end
[2024-09-17 14:55:18] completed in 16 ms

```

## Question 7

7. Create a view "faculty" showing only the ID, name, and department of instructors.

### SQL

```

create view faculty as
select ID,name,dept_name
from instructor

```

	ID	name	dept_name
1	10101	Srinivasan	Comp. Sci.
2	12121	Wu	Finance
3	15151	Mozart	Music
4	22222	Einstein	Physics
5	32343	El Said	History
6	33456	Gold	Physics
7	45565	Katz	Comp. Sci.
8	58583	Califieri	History
9	76543	Singh	Finance
10	76766	Crick	Biology
11	83821	Brandt	Comp. Sci.
12	98345	Kim	Elec. Eng.

### Output log

```

university> create view faculty as
select ID,name,dept_name
from instructor
[2024-09-17 15:12:48] completed in 12 ms

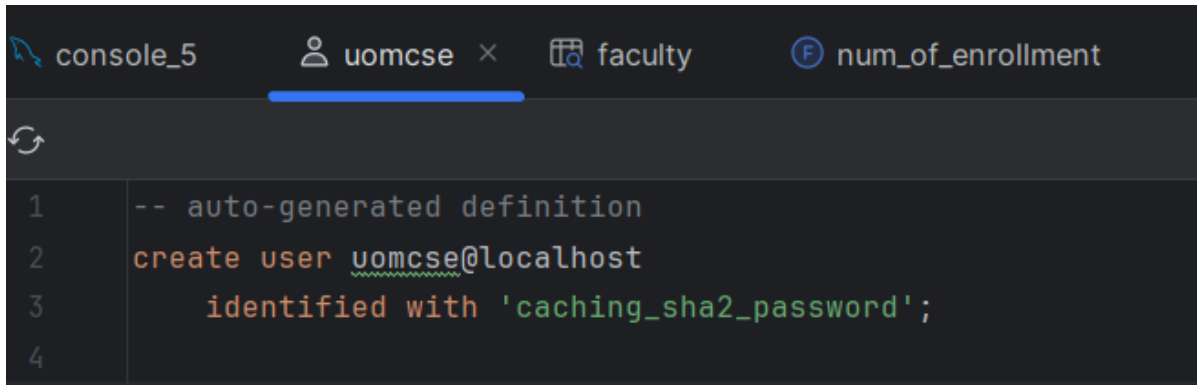
```

## Question 8

8. Create a user "uomcse" with the password "uomcse123".

SQL

```
create user 'uomcse'@'localhost' identified by 'uomcse123';
```

A screenshot of a SQL IDE interface. At the top, there are tabs for 'console\_5', 'uomcse', 'faculty', and 'num\_of\_enrollment'. The 'uomcse' tab is active. Below the tabs, there is a code editor with a dark background. The code in the editor is: 

```
1 -- auto-generated definition
2 create user uomcse@localhost
3     identified with 'caching_sha2_password';
4
```

Output log

```
university> create user 'uomcse'@'localhost' identified by 'uomcse123'
[2024-09-17 15:31:14] completed in 10 ms
```

## Question 9

9. Grant "uomcse" user, select privileges to your "faculty" view.

SQL

```
GRANT SELECT ON faculty TO 'uomcse'@'localhost';
```

Output Log

```
university> GRANT SELECT ON faculty TO 'uomcse'@'localhost'
[2024-09-17 15:31:24] completed in 7 ms
```

## Question 10

10. Grant "uomcse" user, all privileges to the "takes" relation.

SQL

```
GRANT all privileges ON takes TO 'uomcse'@'localhost';
```

Output Log

```
university> GRANT all privileges ON takes TO 'uomcse'@'localhost'
[2024-09-17 15:35:55] completed in 5 ms
```

68

69 ✓ `SHOW GRANTS FOR 'uomcse'@'localhost';`

Output Result 62 x

Grants for uomcse@localhost

1	<code>GRANT USAGE ON *.* TO 'uomcse'@'localhost'</code>
2	<code>GRANT SELECT ON `university`.`faculty` TO 'uomcse'@'localhost'</code>
3	<code>GRANT ALL PRIVILEGES ON `university`.`takes` TO 'uomcse'@'localhost'</code>