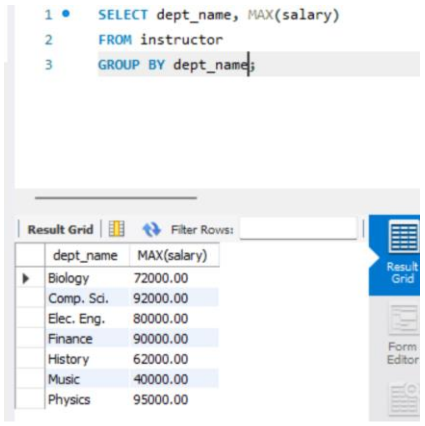
1 SQL Queries

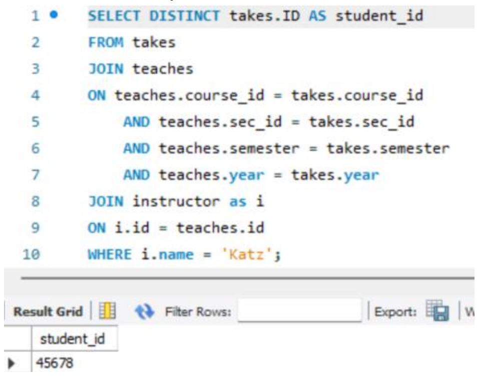
1. For each department, find the maximum salary of instructors in that department.



```sql
SELECT dept_name, MAX(salary)
FROM instructor
GROUP BY dept_name;
```

dept_name	MAX(salary)
Biology	72000.00
Comp. Sci.	92000.00
Elec. Eng.	80000.00
Finance	90000.00
History	62000.00
Music	40000.00
Physics	95000.00

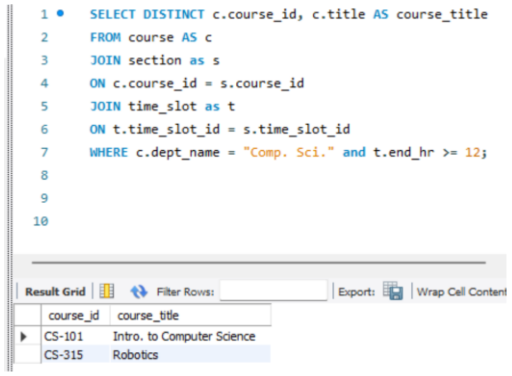
2. Find the IDs of all students who were taught by an instructor named Katz; make sure there are no duplicates in the result.



```sql
SELECT DISTINCT takes.ID AS student_id
FROM takes
JOIN teaches
ON teaches.course_id = takes.course_id
    AND teaches.sec_id = takes.sec_id
    AND teaches.semester = takes.semester
    AND teaches.year = takes.year
JOIN instructor as i
ON i.id = teaches.id
WHERE i.name = 'Katz';
```

student_id
45678

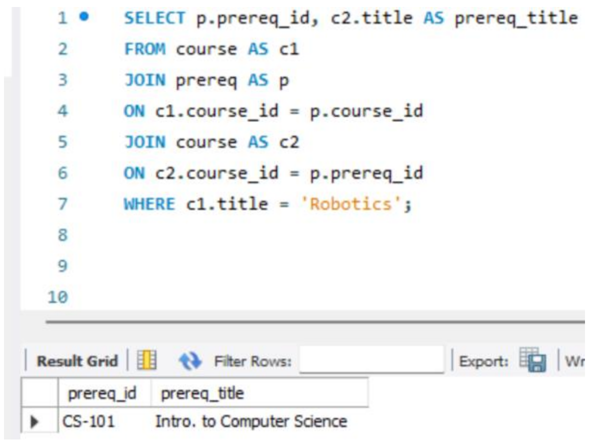
3. Find the ID and title of each course in Comp. Sci. that has had at least one section with afternoon hours (i.e., ends at or after 12:00). (You should eliminate duplicates if any.)



```sql
SELECT DISTINCT c.course_id, c.title AS course_title
FROM course AS c
JOIN section as s
ON c.course_id = s.course_id
JOIN time_slot as t
ON t.time_slot_id = s.time_slot_id
WHERE c.dept_name = "Comp. Sci." and t.end_hr >= 12;
```

course_id	course_title
CS-101	Intro. to Computer Science
CS-315	Robotics

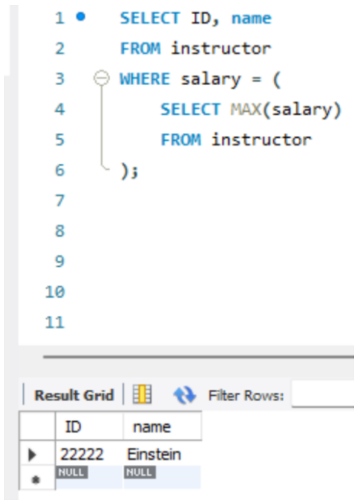
4. Find the IDs and titles of all the courses that are prerequisite to the Robotics course.



```sql
SELECT p.prereq_id, c2.title AS prereq_title
FROM course AS c1
JOIN prereq AS p
ON c1.course_id = p.course_id
JOIN course AS c2
ON c2.course_id = p.prereq_id
WHERE c1.title = 'Robotics';
```

```
prereq_id   prereq_title
CS-101      Intro. to Computer Science
```

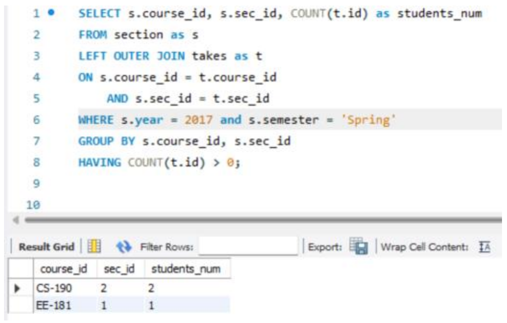
5. Find the IDs and names of all instructors earning the highest salary (there may be more than one with the same salary).



```sql
SELECT ID, name
FROM instructor
WHERE salary = (
    SELECT MAX(salary)
    FROM instructor
);
```

```
ID      name
22222   Einstein
NULL    NULL
```

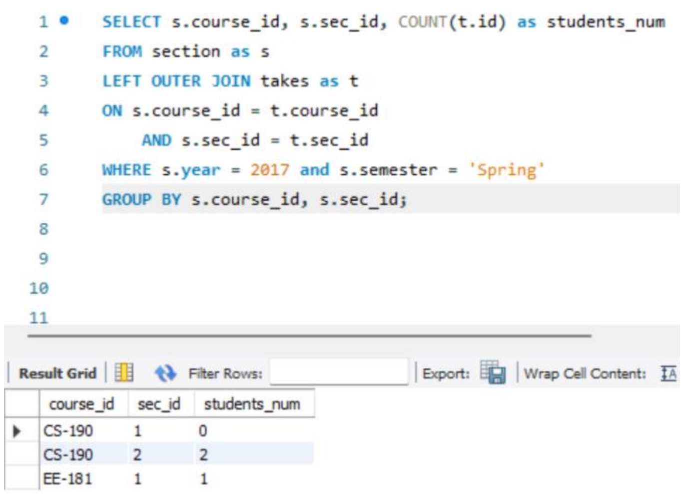
6. Find the enrollment (number of students) in each section that was offered in Spring 2017. The result columns should be course id, section id, students num. You do not need to output sections with 0 students.



```sql
SELECT s.course_id, s.sec_id, COUNT(t.id) as students_num
FROM section as s
LEFT OUTER JOIN takes as t
ON s.course_id = t.course_id
    AND s.sec_id = t.sec_id
WHERE s.year = 2017 and s.semester = 'Spring'
GROUP BY s.course_id, s.sec_id
HAVING COUNT(t.id) > 0;
```

```
course_id   sec_id   students_num
CS-190      2        2
EE-181      1        1
```

7. Rewrite the preceding query, but also output sections with 0 students.

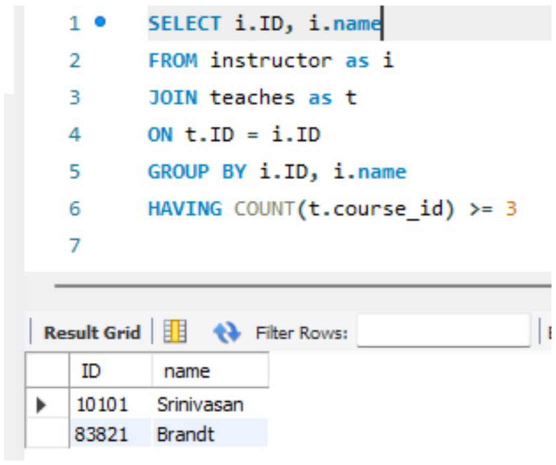


```sql
SELECT s.course_id, s.sec_id, COUNT(t.id) as students_num
FROM section as s
LEFT OUTER JOIN takes as t
ON s.course_id = t.course_id
    AND s.sec_id = t.sec_id
WHERE s.year = 2017 and s.semester = 'Spring'
GROUP BY s.course_id, s.sec_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

```
course_id  sec_id  students_num
CS-190     1       0
CS-190     2       2
EE-181     1       1
```

8. Find the IDs and names of all instructors who have taught at least 3 different courses.

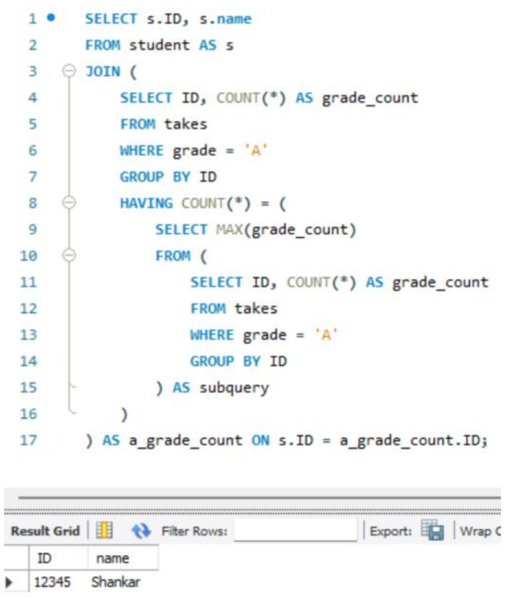


```sql
SELECT i.ID, i.name
FROM instructor as i
JOIN teaches as t
ON t.ID = i.ID
GROUP BY i.ID, i.name
HAVING COUNT(t.course_id) >= 3
```

Result Grid | Filter Rows:

```
ID     name
10101  Srinivasan
83821  Brandt
```

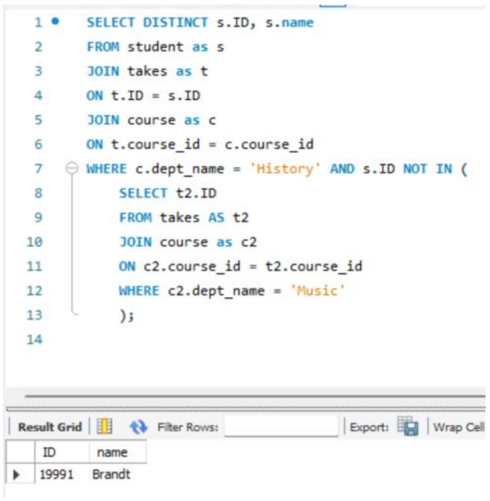
9. Find the ID and name of the student with the highest number of ’A’ grades (there may be more than one such student).



```
SELECT s.ID, s.name
FROM student AS s
JOIN (
    SELECT ID, COUNT(*) AS grade_count
    FROM takes
    WHERE grade = 'A'
    GROUP BY ID
    HAVING COUNT(*) = (
        SELECT MAX(grade_count)
        FROM (
            SELECT ID, COUNT(*) AS grade_count
            FROM takes
            WHERE grade = 'A'
            GROUP BY ID
        ) AS subquery
    )
) AS a_grade_count ON s.ID = a_grade_count.ID;
```

```
ID     name
12345  Shankar
```

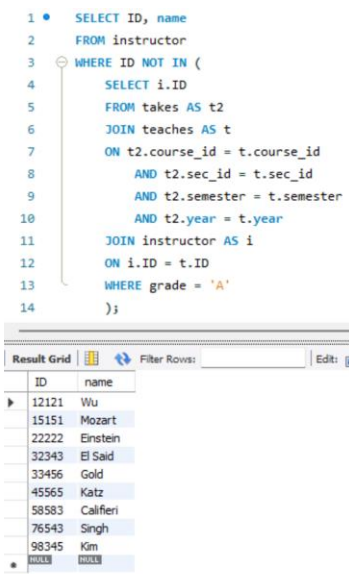
10. Find the ID and name of each History student who has not taken any Music courses.



```
SELECT DISTINCT s.ID, s.name
FROM student as s
JOIN takes as t
ON t.ID = s.ID
JOIN course as c
ON t.course_id = c.course_id
WHERE c.dept_name = 'History' AND s.ID NOT IN (
    SELECT t2.ID
    FROM takes AS t2
    JOIN course as c2
    ON c2.course_id = t2.course_id
    WHERE c2.dept_name = 'Music'
    );
```

```
ID     name
19991  Brandt
```

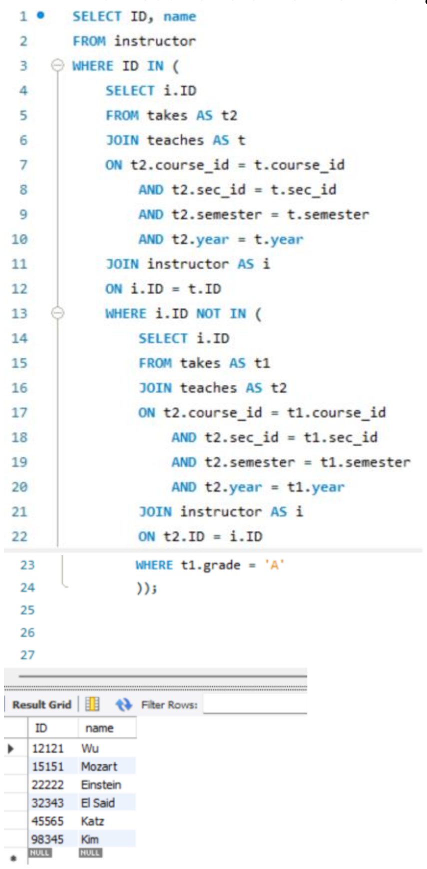
11. Find the ID and name of each instructor who has never given an ’A’ grade in any course she or he has taught. (Instructors who have never taught a course trivially satisfy this condition.)



```sql
SELECT ID, name
FROM instructor
WHERE ID NOT IN (
    SELECT i.ID
    FROM takes AS t2
    JOIN teaches AS t
    ON t2.course_id = t.course_id
        AND t2.sec_id = t.sec_id
        AND t2.semester = t.semester
        AND t2.year = t.year
    JOIN instructor AS i
    ON i.ID = t.ID
    WHERE grade = 'A'
    );
```

```
ID     name
12121  Wu
15151  Mozart
22222  Einstein
32343  El Said
33456  Gold
45565  Katz
58583  Califieri
76543  Singh
98345  Kim
NULL   NULL
```

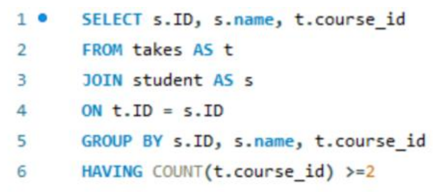
12. Rewrite the preceding query, but also ensure that you include only instructors who have given at least one other non-null grade in some course.



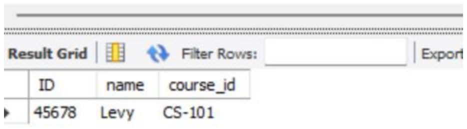
```sql
SELECT ID, name
FROM instructor
WHERE ID IN (
    SELECT i.ID
    FROM takes AS t2
    JOIN teaches AS t
    ON t2.course_id = t.course_id
        AND t2.sec_id = t.sec_id
        AND t2.semester = t.semester
        AND t2.year = t.year
    JOIN instructor AS i
    ON i.ID = t.ID
    WHERE i.ID NOT IN (
        SELECT i.ID
        FROM takes AS t1
        JOIN teaches AS t2
        ON t2.course_id = t1.course_id
            AND t2.sec_id = t1.sec_id
            AND t2.semester = t1.semester
            AND t2.year = t1.year
        JOIN instructor AS i
        ON t2.ID = i.ID
        WHERE t1.grade = 'A'
        ));
```

```
ID     name
12121  Wu
15151  Mozart
22222  Einstein
32343  El Said
45565  Katz
98345  Kim
NULL   NULL
```

13. For each student who have retaken a course at least once (i.e., the student has taken the course at least twice), show the student’s ID, name and the course ID.

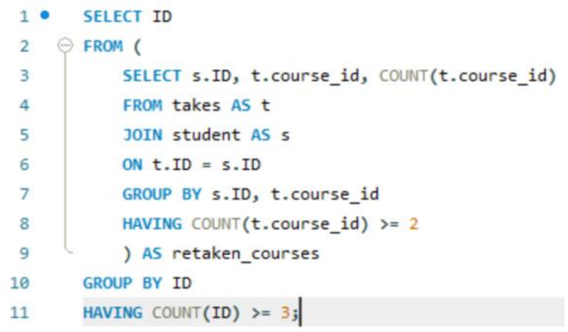


```sql
SELECT s.ID, s.name, t.course_id
FROM takes AS t
JOIN student AS s
ON t.ID = s.ID
GROUP BY s.ID, s.name, t.course_id
HAVING COUNT(t.course_id) >=2
```

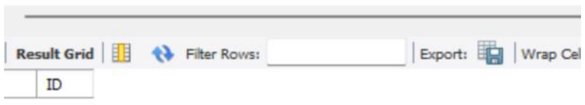


```
ID     name  course_id
45678  Levy  CS-101
```

14. Find the IDs of those students who have retaken at least three distinct courses at least once (i.e., the student has taken the course at least two times).

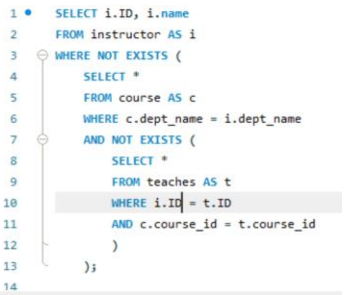


```sql
SELECT ID
FROM (
    SELECT s.ID, t.course_id, COUNT(t.course_id)
    FROM takes AS t
    JOIN student AS s
    ON t.ID = s.ID
    GROUP BY s.ID, t.course_id
    HAVING COUNT(t.course_id) >= 2
    ) AS retaken_courses
GROUP BY ID
HAVING COUNT(ID) >= 3;
```

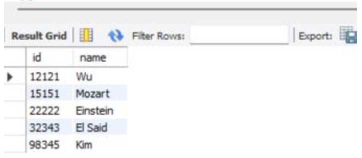


```
ID
```

15. Find the IDs and names of those instructors who have taught every course in their department



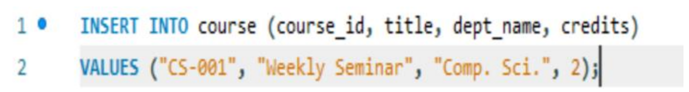
```sql
SELECT i.ID, i.name
FROM instructor AS i
WHERE NOT EXISTS (
    SELECT *
    FROM course AS c
    WHERE c.dept_name = i.dept_name
    AND NOT EXISTS (
        SELECT *
        FROM teaches AS t
        WHERE i.ID = t.ID
        AND c.course_id = t.course_id
        )
    );
```



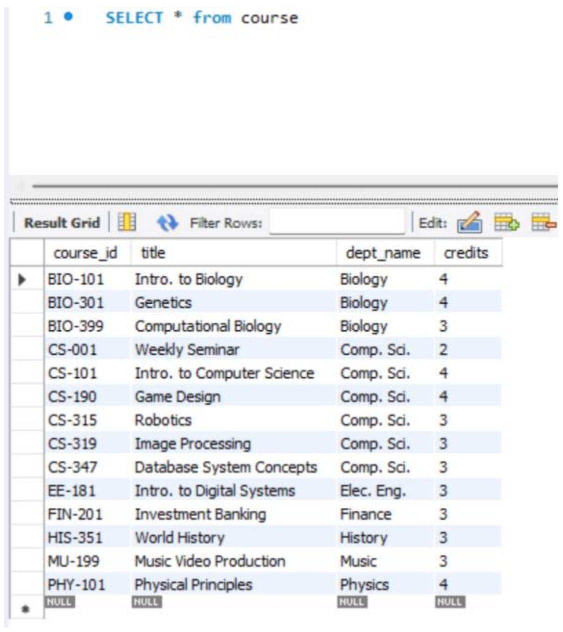
```
id     name
12121  Wu
15151  Mozart
22222  Einstein
32343  El Said
98345  Kim
```

2 SQL DML

1. Create a new course ”CS-001” in the Comp. Sci. department, titled ”Weekly Seminar”, with 2 credits.



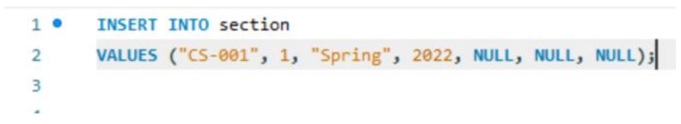
```sql
INSERT INTO course (course_id, title, dept_name, credits)
VALUES ("CS-001", "Weekly Seminar", "Comp. Sci.", 2);
```



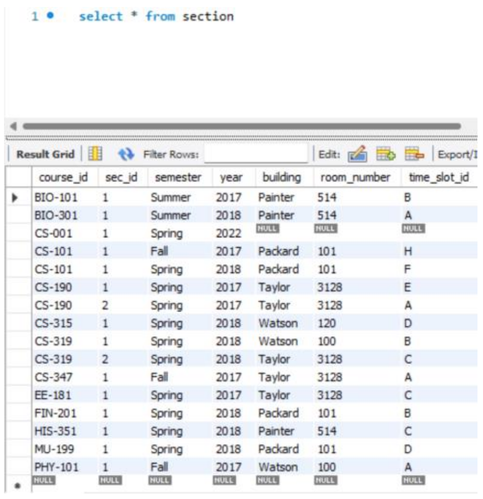
```sql
SELECT * from course
```

course_id	title	dept_name	credits
BIO-101	Intro. to Biology	Biology	4
BIO-301	Genetics	Biology	4
BIO-399	Computational Biology	Biology	3
CS-001	Weekly Seminar	Comp. Sci.	2
CS-101	Intro. to Computer Science	Comp. Sci.	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3
CS-319	Image Processing	Comp. Sci.	3
CS-347	Database System Concepts	Comp. Sci.	3
EE-181	Intro. to Digital Systems	Elec. Eng.	3
FIN-201	Investment Banking	Finance	3
HIS-351	World History	History	3
MU-199	Music Video Production	Music	3
PHY-101	Physical Principles	Physics	4
NULL	NULL	NULL	NULL

2. Create a section of this course in Spring 2022, with sec id of 1, and with the location of this section not yet specified.



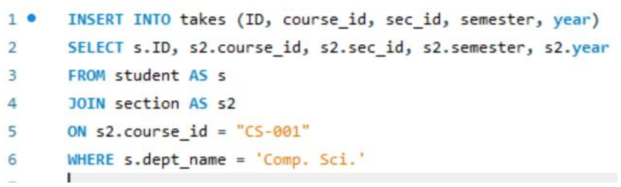
```sql
INSERT INTO section
VALUES ("CS-001", 1, "Spring", 2022, NULL, NULL, NULL);
```



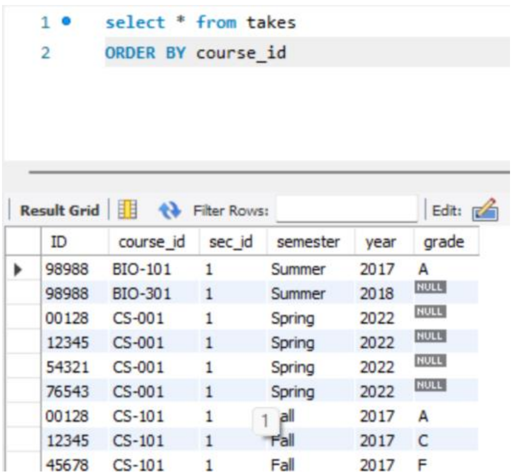
```sql
select * from section
```

course_id	sec_id	semester	year	building	room_number	time_slot_id
BIO-101	1	Summer	2017	Painter	514	B
BIO-301	1	Summer	2018	Painter	514	A
CS-001	1	Spring	2022	NULL	NULL	NULL
CS-101	1	Fall	2017	Packard	101	H
CS-101	1	Spring	2018	Packard	101	F
CS-190	1	Spring	2017	Taylor	3128	E
CS-190	2	Spring	2017	Taylor	3128	A
CS-315	1	Spring	2018	Watson	120	D
CS-319	1	Spring	2018	Watson	100	B
CS-319	2	Spring	2018	Taylor	3128	C
CS-347	1	Fall	2017	Taylor	3128	A
EE-181	1	Spring	2017	Taylor	3128	C
FIN-201	1	Spring	2018	Packard	101	B
HIS-351	1	Spring	2018	Painter	514	C
MU-199	1	Spring	2018	Packard	101	D
PHY-101	1	Fall	2017	Watson	100	A
NULL	NULL	NULL	NULL	NULL	NULL	NULL

3. Enroll every student in the Comp. Sci. department in the above section.



```sql
INSERT INTO takes (ID, course_id, sec_id, semester, year)
SELECT s.ID, s2.course_id, s2.sec_id, s2.semester, s2.year
FROM student AS s
JOIN section AS s2
ON s2.course_id = "CS-001"
WHERE s.dept_name = 'Comp. Sci.'
```

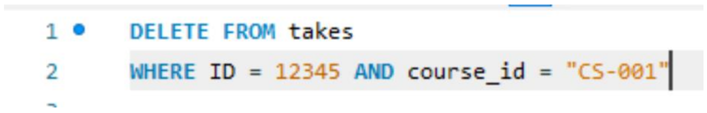


```sql
select * from takes
ORDER BY course_id
```

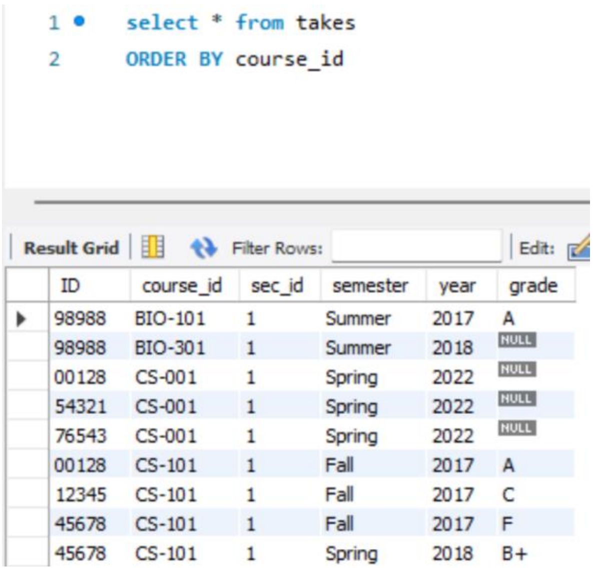
Result Grid | Filter Rows: | Edit:

ID	course_id	sec_id	semester	year	grade
98988	BIO-101	1	Summer	2017	A
98988	BIO-301	1	Summer	2018	NULL
00128	CS-001	1	Spring	2022	NULL
12345	CS-001	1	Spring	2022	NULL
54321	CS-001	1	Spring	2022	NULL
76543	CS-001	1	Spring	2022	NULL
00128	CS-101	1	Fall	2017	A
12345	CS-101	1	Fall	2017	C
45678	CS-101	1	Fall	2017	F

4. Delete enrollments in the above section where the student's ID is 12345.



```sql
DELETE FROM takes
WHERE ID = 12345 AND course_id = "CS-001"
```

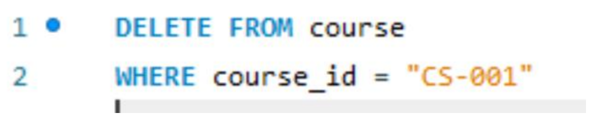


```sql
select * from takes
ORDER BY course_id
```

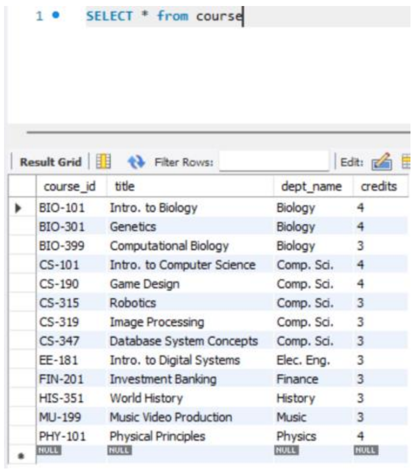
Result Grid | Filter Rows: | Edit:

ID	course_id	sec_id	semester	year	grade
98988	BIO-101	1	Summer	2017	A
98988	BIO-301	1	Summer	2018	NULL
00128	CS-001	1	Spring	2022	NULL
54321	CS-001	1	Spring	2022	NULL
76543	CS-001	1	Spring	2022	NULL
00128	CS-101	1	Fall	2017	A
12345	CS-101	1	Fall	2017	C
45678	CS-101	1	Fall	2017	F
45678	CS-101	1	Spring	2018	B+

5. Delete the course CS-001. What happened to the section and enrollments of this course?



```
DELETE FROM course
WHERE course_id = "CS-001"
```



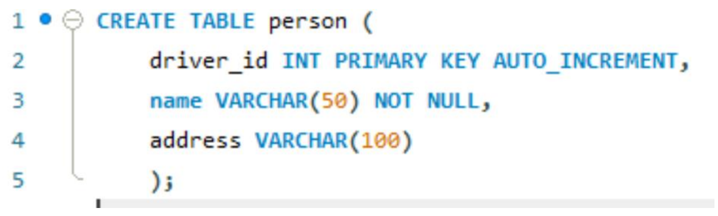
```
SELECT * from course
```

Result Grid | Filter Rows: | Edit:

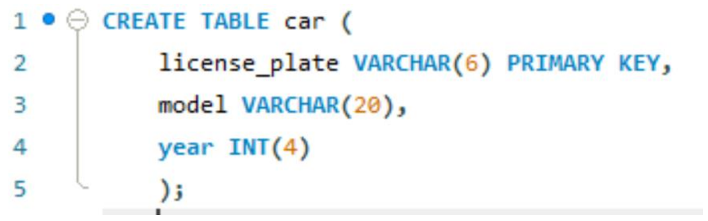
course_id	title	dept_name	credits
BIO-101	Intro. to Biology	Biology	4
BIO-301	Genetics	Biology	4
BIO-399	Computational Biology	Biology	3
CS-101	Intro. to Computer Science	Comp. Sci.	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3
CS-319	Image Processing	Comp. Sci.	3
CS-347	Database System Concepts	Comp. Sci.	3
EE-181	Intro. to Digital Systems	Elec. Eng.	3
FIN-201	Investment Banking	Finance	3
HIS-351	World History	History	3
MU-199	Music Video Production	Music	3
PHY-101	Physical Principles	Physics	4
NULL	NULL	NULL	NULL

-rows with “CS-001” as the course_id were deleted from the section and takes table of the database

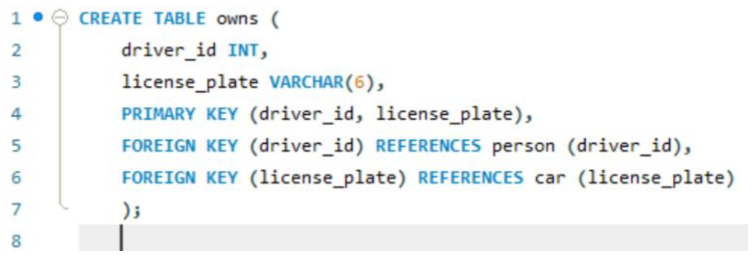
3 SQL DDL



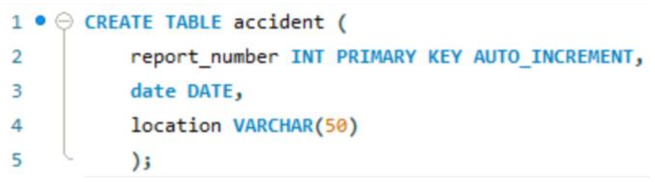
```
CREATE TABLE person (
    driver_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(50) NOT NULL,
    address VARCHAR(100)
    );
```



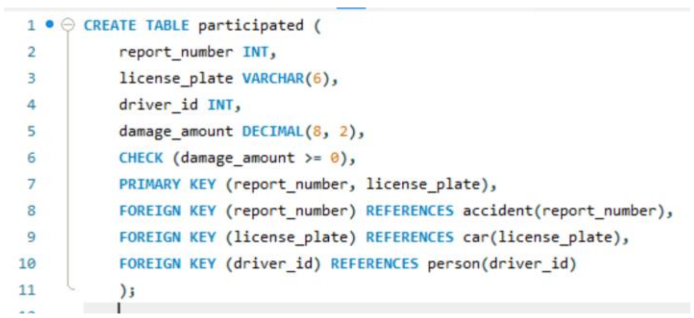
```
CREATE TABLE car (
    license_plate VARCHAR(6) PRIMARY KEY,
    model VARCHAR(20),
    year INT(4)
    );
```



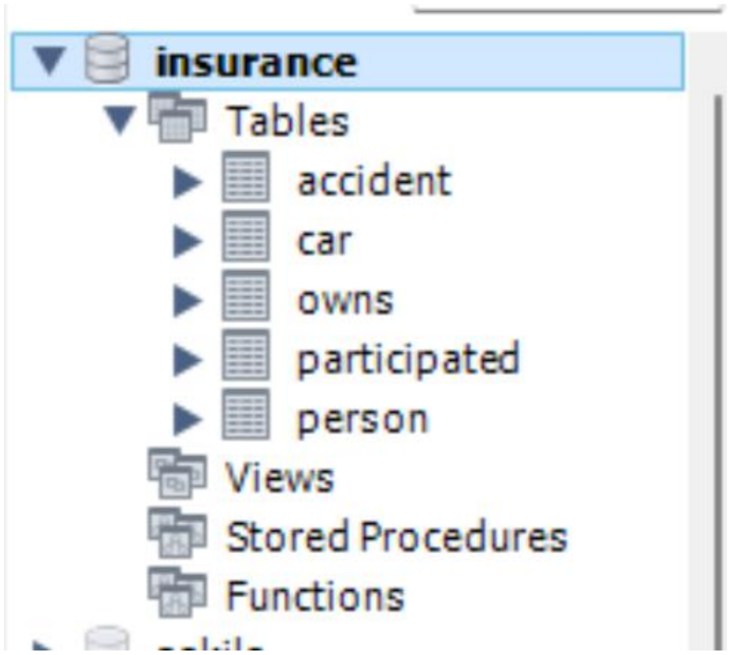
```
CREATE TABLE owns (
    driver_id INT,
    license_plate VARCHAR(6),
    PRIMARY KEY (driver_id, license_plate),
    FOREIGN KEY (driver_id) REFERENCES person (driver_id),
    FOREIGN KEY (license_plate) REFERENCES car (license_plate)
    );
```



```
CREATE TABLE accident (
    report_number INT PRIMARY KEY AUTO_INCREMENT,
    date DATE,
    location VARCHAR(50)
    );
```



```sql
CREATE TABLE participated (
    report_number INT,
    license_plate VARCHAR(6),
    driver_id INT,
    damage_amount DECIMAL(8, 2),
    CHECK (damage_amount >= 0),
    PRIMARY KEY (report_number, license_plate),
    FOREIGN KEY (report_number) REFERENCES accident(report_number),
    FOREIGN KEY (license_plate) REFERENCES car(license_plate),
    FOREIGN KEY (driver_id) REFERENCES person(driver_id)
    );
```


insurance
Tables
accident
car
owns
participated
person
Views
Stored Procedures
Functions