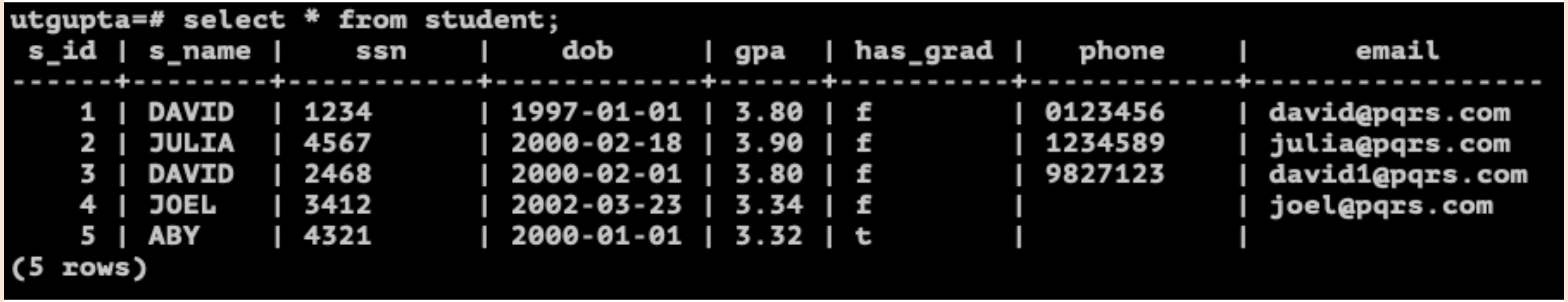
CSE 180: Lab Section

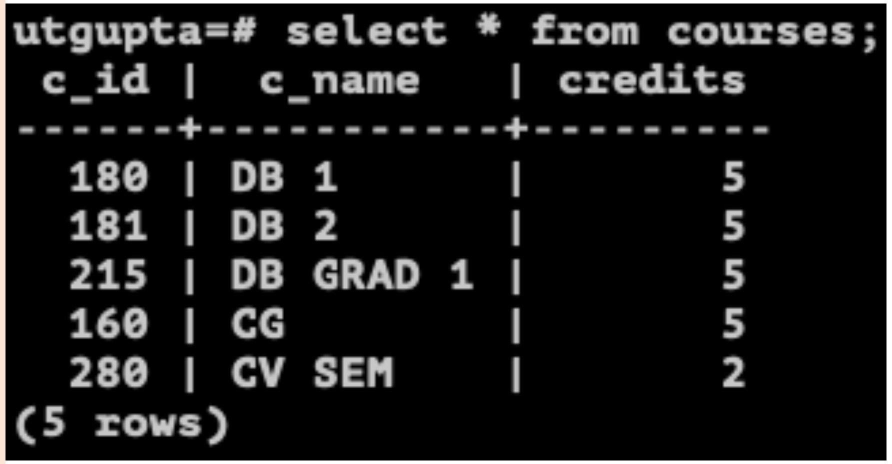
- Recap (Students, Courses, Enrolments)
- Group By
- Having
- More queries from Lecture 5 Slides
- Midterm

Monday, November 6th
Utkarsh Gupta

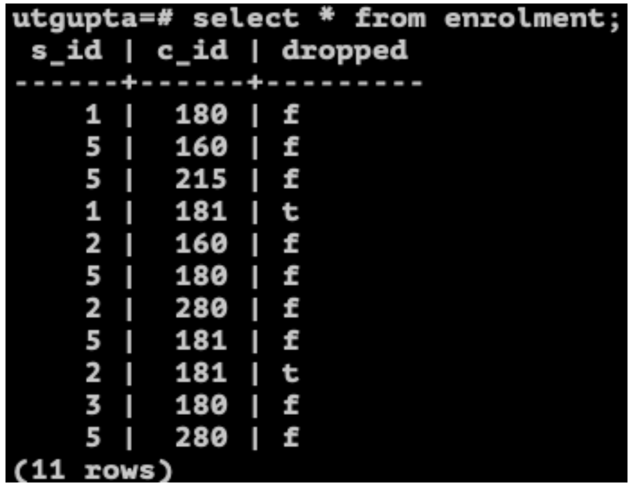
Student, Courses, & Enrolment



```
utgupta=# select * from student;
 s_id | s_name |    ssn     |    dob     | gpa  | has_grad |    phone     |      email
------+--------+------------+------------+------+----------+--------------+-----------------
    1 | DAVID  | 1234       | 1997-01-01 | 3.80 | f        | 0123456      | david@pqrs.com
    2 | JULIA  | 4567       | 2000-02-18 | 3.90 | f        | 1234589      | julia@pqrs.com
    3 | DAVID  | 2468       | 2000-02-01 | 3.80 | f        | 9827123      | david1@pqrs.com
    4 | JOEL   | 3412       | 2002-03-23 | 3.34 | f        |              | joel@pqrs.com
    5 | ABY    | 4321       | 2000-01-01 | 3.32 | t        |              |
(5 rows)
```



```
utgupta=# select * from courses;
 c_id |   c_name    | credits
------+-------------+---------
  180 | DB 1        |       5
  181 | DB 2        |       5
  215 | DB GRAD 1   |       5
  160 | CG          |       5
  280 | CV SEM      |       2
(5 rows)
```

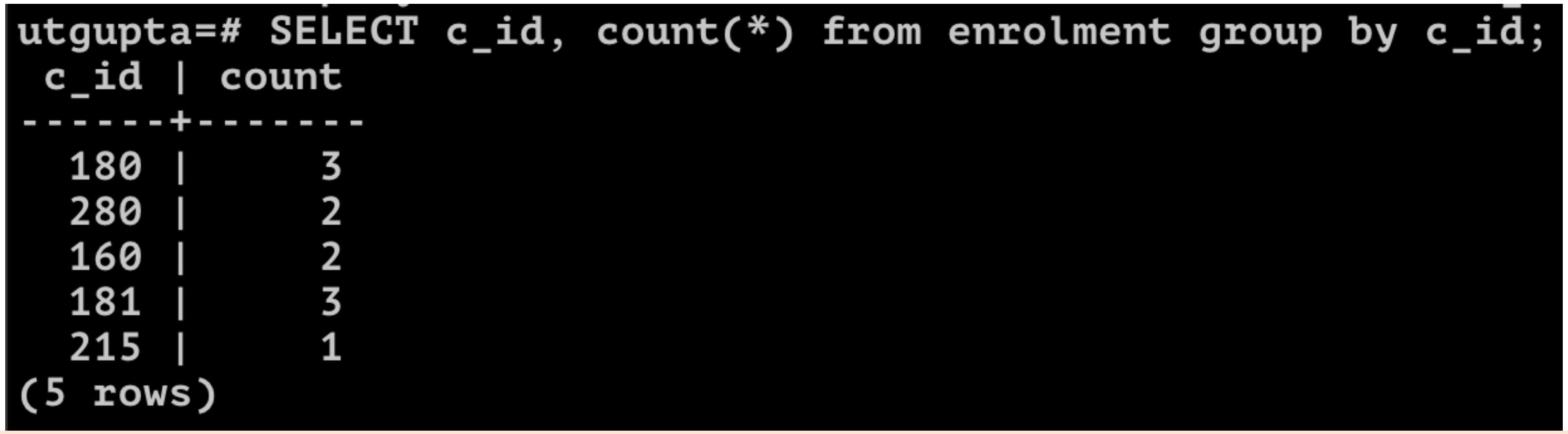


```
utgupta=# select * from enrolment;
 s_id | c_id | dropped
------+------+---------
    1 |  180 | f
    5 |  160 | f
    5 |  215 | f
    1 |  181 | t
    2 |  160 | f
    5 |  180 | f
    2 |  280 | f
    5 |  181 | f
    2 |  181 | t
    3 |  180 | f
    5 |  280 | f
(11 rows)
```

Group By and Having

- When the query requires grouping the rows by one or more columns, one should use GROUP BY clause.
- **Example: Find the total number of students enrolled in the courses.**

```
SELECT c_id, COUNT(*) FROM enrolment e
GROUP BY c_id;
```

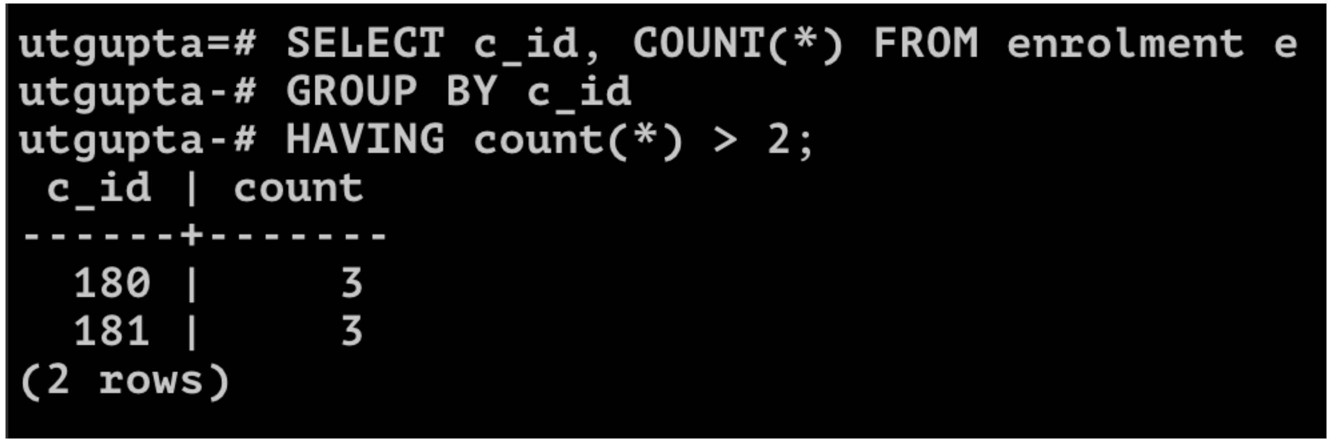


```
utgupta=# SELECT c_id, count(*) from enrolment group by c_id;
 c_id | count
------+-------
  180 |     3
  280 |     2
  160 |     2
  181 |     3
  215 |     1
(5 rows)
```

Group By and Having (cont.)

- **Example: Find the courses where the total number of students are more than 2;**

```
SELECT c_id, COUNT(*) FROM enrolment e
GROUP BY c_id
HAVING count(*) > 2;
```



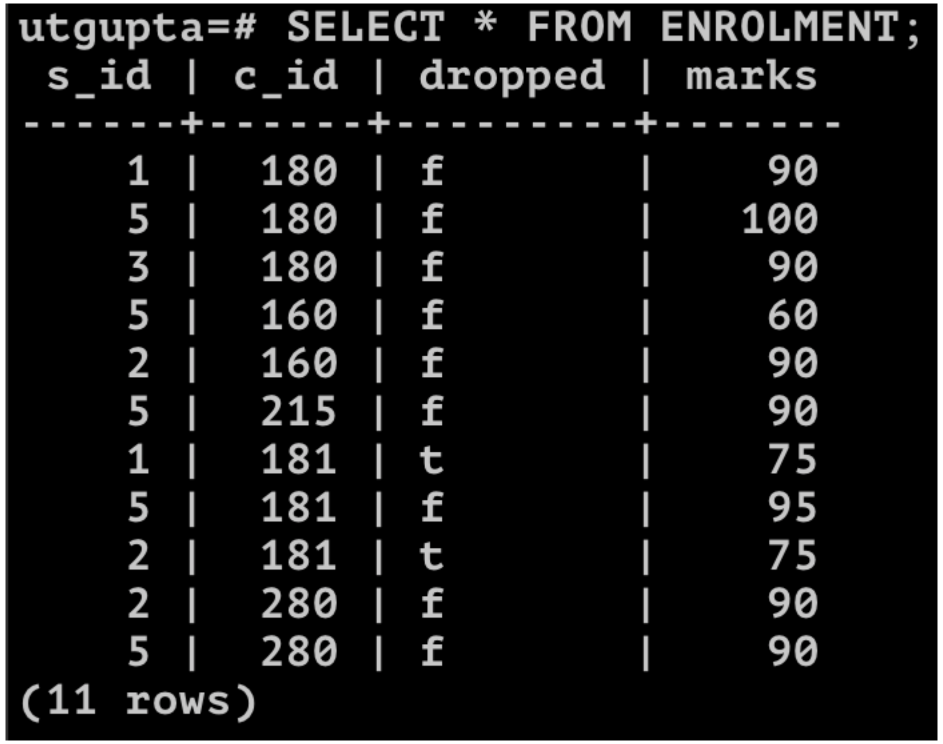
```
utgupta=# SELECT c_id, COUNT(*) FROM enrolment e
utgupta-# GROUP BY c_id
utgupta-# HAVING count(*) > 2;
 c_id | count
------+-------
  180 |     3
  181 |     3
(2 rows)
```

Group By and Having (cont.)

- Add and insert data in the Marks column of the Enrolment table

```
ALTER TABLE Enrolment ADD marks int;
```

```
UPDATE ENROLMENT SET MARKS = 90 WHERE S_ID=1 AND C_ID=180;
UPDATE ENROLMENT SET MARKS = 100 WHERE S_ID=5 AND C_ID=180;
UPDATE ENROLMENT SET MARKS = 90 WHERE S_ID=3 AND C_ID=180;
UPDATE ENROLMENT SET MARKS = 60 WHERE S_ID=5 AND C_ID=160;
UPDATE ENROLMENT SET MARKS = 90 WHERE S_ID=2 AND C_ID=160;
UPDATE ENROLMENT SET MARKS = 90 WHERE S_ID=5 AND C_ID=215;
UPDATE ENROLMENT SET MARKS = 75 WHERE S_ID=1 AND C_ID=181;
UPDATE ENROLMENT SET MARKS = 95 WHERE S_ID=5 AND C_ID=181;
UPDATE ENROLMENT SET MARKS = 75 WHERE S_ID=2 AND C_ID=181;
UPDATE ENROLMENT SET MARKS = 90 WHERE S_ID=2 AND C_ID=280;
UPDATE ENROLMENT SET MARKS = 90 WHERE S_ID=5 AND C_ID=280;
```



```
utgupta=# SELECT * FROM ENROLMENT;
 s_id | c_id | dropped | marks
------+------+---------+-------
    1 |  180 | f       |    90
    5 |  180 | f       |   100
    3 |  180 | f       |    90
    5 |  160 | f       |    60
    2 |  160 | f       |    90
    5 |  215 | f       |    90
    1 |  181 | t       |    75
    5 |  181 | f       |    95
    2 |  181 | t       |    75
    2 |  280 | f       |    90
    5 |  280 | f       |    90
(11 rows)
```

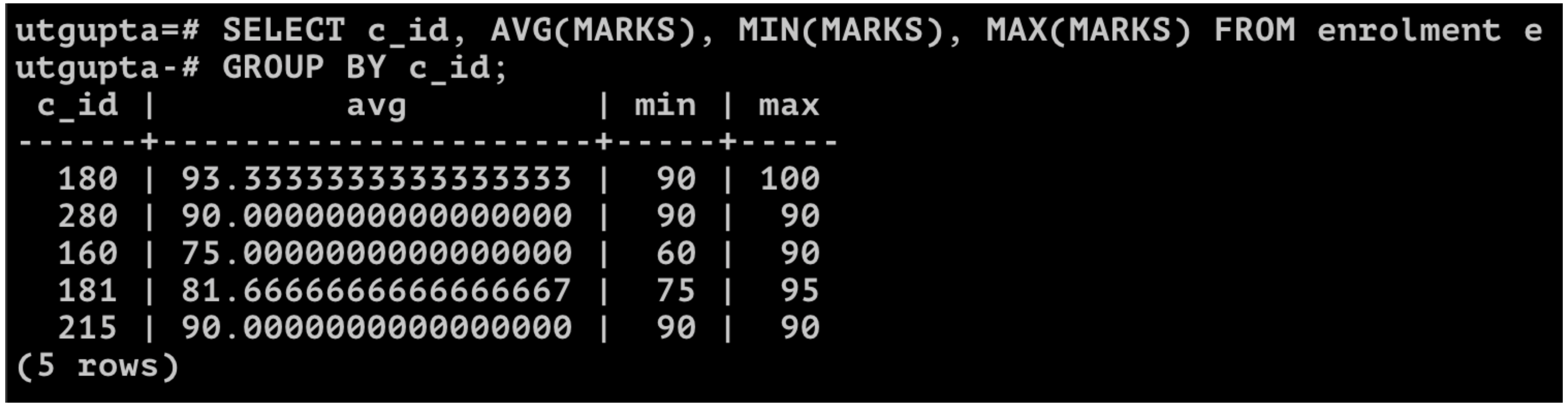
Group By and Having (cont.)

- **Example: Find the AVG, MIN, MAX marks in all the courses.**

SELECT c_id, AVG(MARKS), MIN(MARKS), MAX(MARKS)

FROM Enrolment e

GROUP BY c_id;

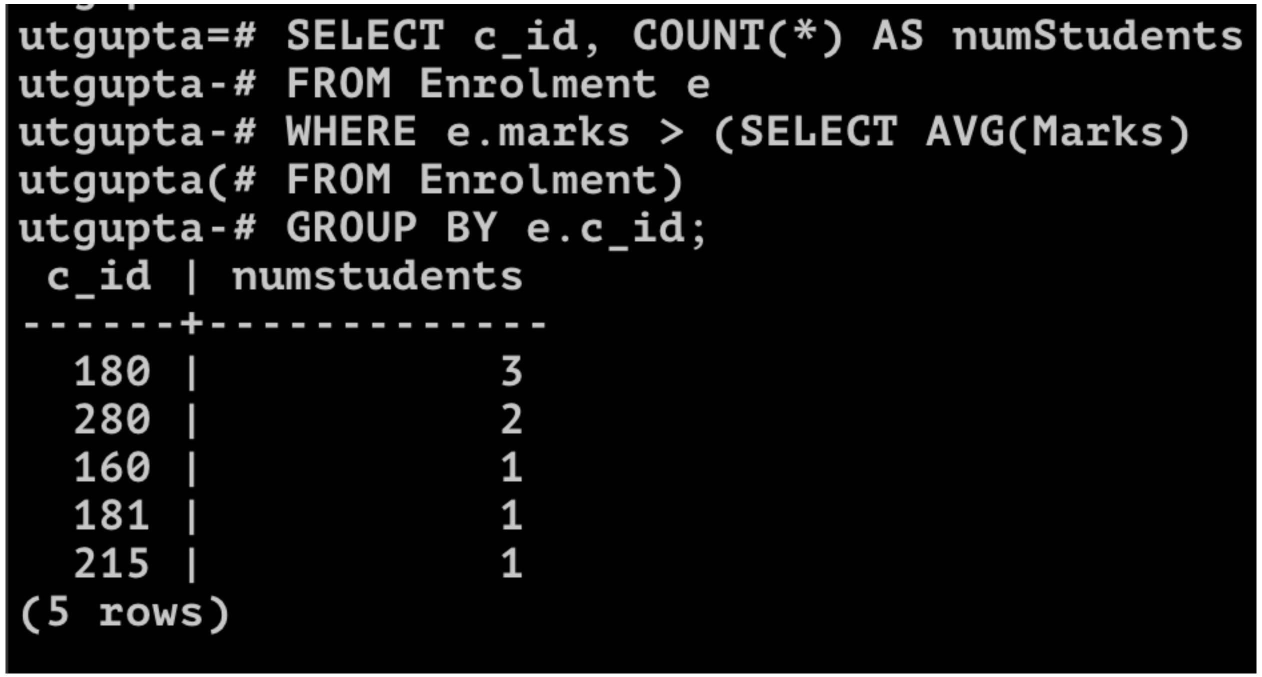


```
utgupta=# SELECT c_id, AVG(MARKS), MIN(MARKS), MAX(MARKS) FROM enrolment e
utgupta-# GROUP BY c_id;
 c_id |         avg          | min | max
------+----------------------+-----+-----
  180 | 93.3333333333333333 |  90 | 100
  280 | 90.0000000000000000 |  90 |  90
  160 | 75.0000000000000000 |  60 |  90
  181 | 81.6666666666666667 |  75 |  95
  215 | 90.0000000000000000 |  90 |  90
(5 rows)
```

Group By and Having (cont.)

- **Example: Find the number of students in the courses whose marks are greater than the average marks of the University.**

```
SELECT c_id, COUNT(*) AS numStudents
FROM Enrolment e
WHERE e.marks > (SELECT AVG(Marks)
                 FROM Enrolment)
GROUP BY e.c_id;
```

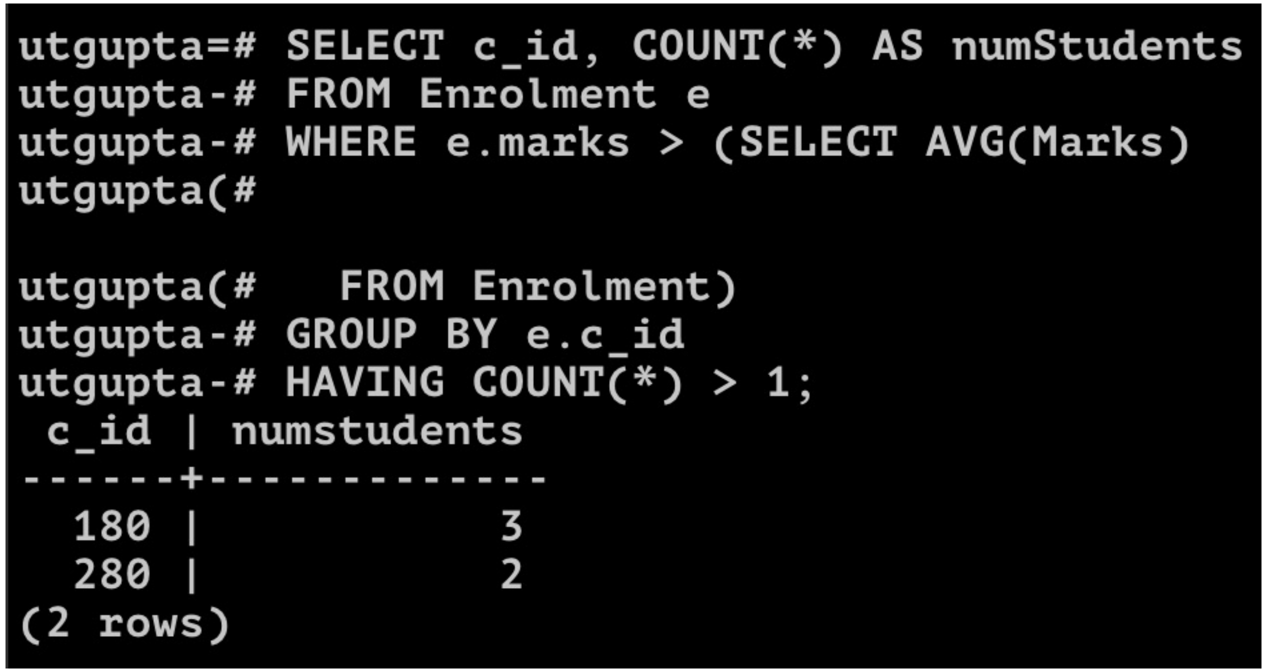


```
utgupta=# SELECT c_id, COUNT(*) AS numStudents
utgupta-# FROM Enrolment e
utgupta-# WHERE e.marks > (SELECT AVG(Marks)
utgupta(# FROM Enrolment)
utgupta-# GROUP BY e.c_id;
 c_id | numstudents
------+-------------
  180 |           3
  280 |           2
  160 |           1
  181 |           1
  215 |           1
(5 rows)
```

Group By and Having (cont.)

- **Example: Find the number of students in the courses whose marks are greater than the average marks of the University and only display the courses where the number of students in a course are greater than 1.**

```
SELECT c_id, COUNT(*) AS numStudents
FROM Enrolment e
WHERE e.marks > (SELECT AVG(Marks)
                 FROM Enrolment)
GROUP BY e.c_id
HAVING COUNT(*) > 1;
```

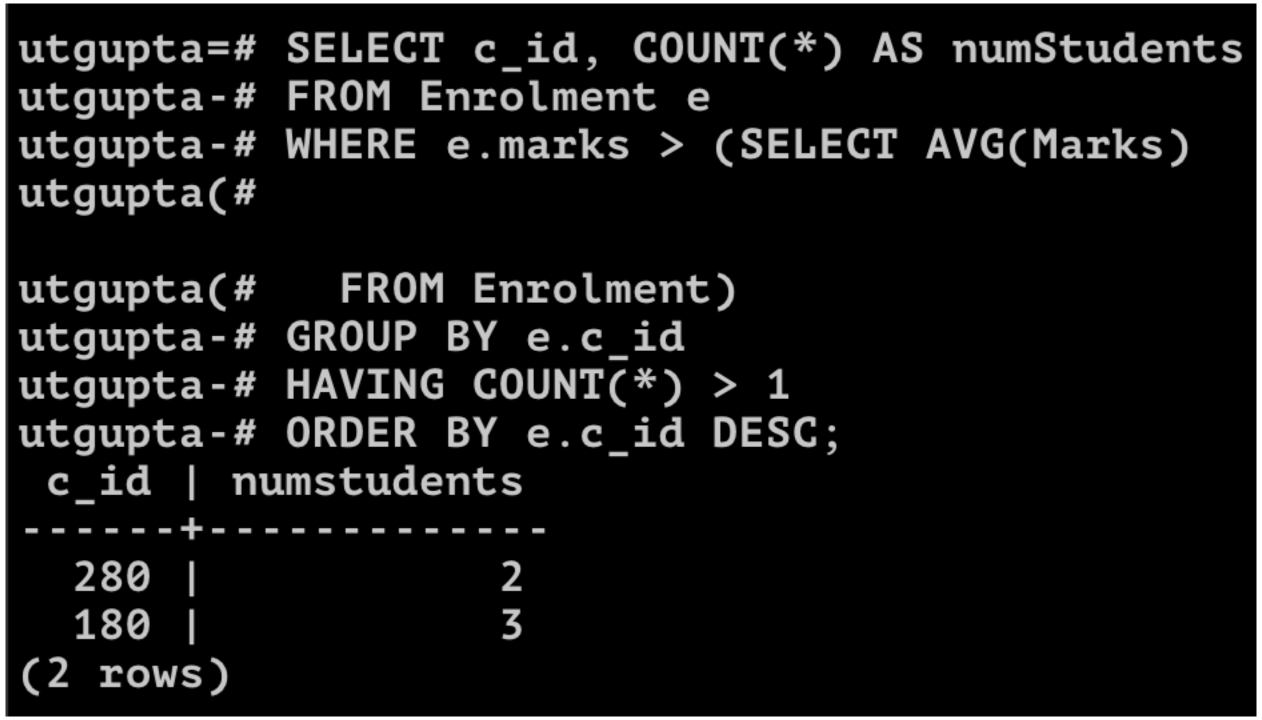


```
utgupta=# SELECT c_id, COUNT(*) AS numStudents
utgupta-# FROM Enrolment e
utgupta-# WHERE e.marks > (SELECT AVG(Marks)
utgupta(#

utgupta(#   FROM Enrolment)
utgupta-# GROUP BY e.c_id
utgupta-# HAVING COUNT(*) > 1;
 c_id | numstudents
------+-------------
  180 |           3
  280 |           2
(2 rows)
```

Group By and Having (cont.)

- **Example: Find the number of students in the courses whose marks are greater than the average marks of the University and only display the courses where the number of students in a course are greater than 1. Display the courses in DESCENDING ORDER of C_ID.**

```
SELECT c_id, COUNT(*) AS numStudents
FROM Enrolment e
WHERE e.marks > (SELECT AVG(Marks)
                 FROM Enrolment)
GROUP BY e.c_id
HAVING COUNT(*) > 1
ORDER BY e.c_id DESC;
```



```
utgupta=# SELECT c_id, COUNT(*) AS numStudents
utgupta-# FROM Enrolment e
utgupta-# WHERE e.marks > (SELECT AVG(Marks)
utgupta(#

utgupta(#   FROM Enrolment)
utgupta-# GROUP BY e.c_id
utgupta-# HAVING COUNT(*) > 1
utgupta-# ORDER BY e.c_id DESC;
 c_id | numstudents
------+-------------
  280 |           2
  180 |           3
(2 rows)
```

Group By and Having (cont.)

- **Example: Find the number of students in the courses whose marks are greater than 70 and at least one student has dropped that particular course. Moreover, display the course only if it has more than 1 student enrolled in it.**

```
SELECT c_id, COUNT(*) AS totalStudents

FROM Enrolment e

WHERE e.marks > 70

GROUP BY e.c_id

HAVING COUNT(*) > 1 AND SOME (e.dropped = TRUE);
```

More queries from Lecture 5

- Reference: Lecture 5 Slides

Sailors Table:

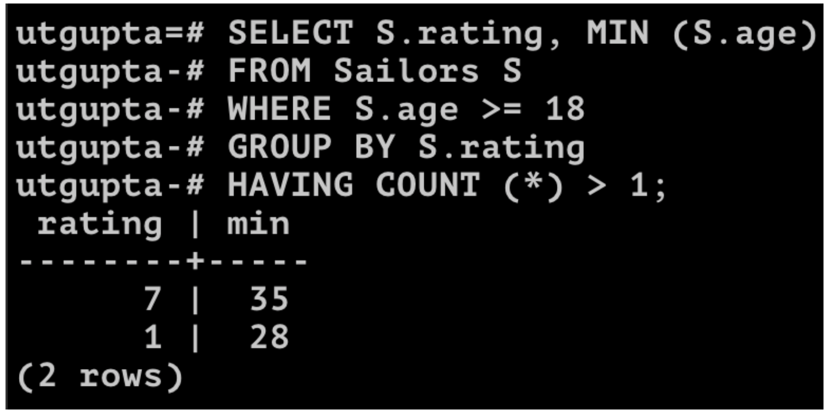
sid	sname	rating	age
22	Dustin	7	45.0
31	Lubber	8	55.5
71	Zorba	10	16.0
64	Horatio	7	35.0
92	Frodo	1	28.0
38	Sam	1	30.0
29	Brutus	1	33.0
58	Rusty	10	35.0

More queries from Lecture 5 (cont.)

- Reference: Lecture 5 Slides

For each rating that has at least 2 sailors whose age is greater than or equal to 18, find the age of the youngest sailor whose age is greater that or equal to 18.

```
SELECT S.rating, MIN (S.age)
FROM Sailors S
WHERE S.age >= 18
GROUP BY S.rating
HAVING COUNT (*) > 1;
```



```
utgupta=# SELECT S.rating, MIN (S.age)
utgupta-# FROM Sailors S
utgupta-# WHERE S.age >= 18
utgupta-# GROUP BY S.rating
utgupta-# HAVING COUNT (*) > 1;
 rating | min
--------+-----
      7 |  35
      1 |  28
(2 rows)
```

sid	sname	rating	age
22	Dustin	7	45.0
31	Lubber	8	55.5
71	Zorba	10	16.0
64	Horatio	7	35.0
92	Frodo	1	28.0
38	Sam	1	30.0
29	Brutus	1	33.0
58	Rusty	10	35.0

More queries from Lecture 5 (cont.)

- Reference: Lecture 5 Slides

For each rating that has at least 2 sailors whose age is greater than or equal to 18, find the age of the youngest sailor in the group where at least one sailor is older than 40 years.

```
SELECT S.rating, MIN (S.age)
FROM Sailors S
WHERE S.age >= 18
GROUP BY S.rating
HAVING COUNT (*) > 1 AND SOME (S.age > 40);
```

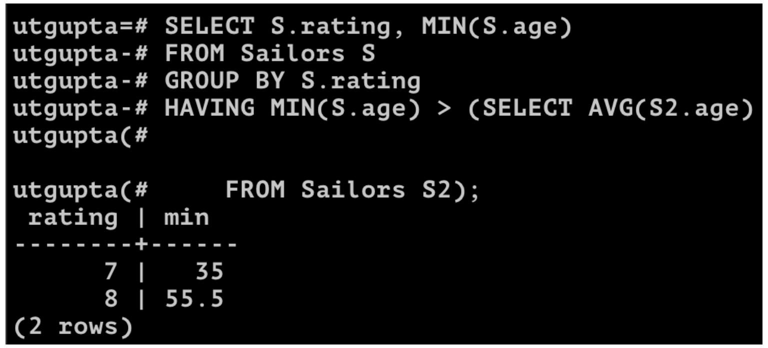
sid	sname	rating	age
22	Dustin	7	45.0
31	Lubber	8	55.5
71	Zorba	10	16.0
64	Horatio	7	35.0
92	Frodo	1	28.0
38	Sam	1	30.0
29	Brutus	1	33.0
58	Rusty	10	35.0

More queries from Lecture 5 (cont.)

- Reference: Lecture 5 Slides

Find the minimum age of sailors in each rating category such that the minimum age of the sailors in that category is greater than the average age of all the sailors

```
SELECT S.rating, MIN(S.age)
FROM Sailors S
GROUP BY S.rating
HAVING MIN(S.age) > (SELECT AVG(S2.age)
                     FROM Sailors S2);
```



```
utgupta=# SELECT S.rating, MIN(S.age)
utgupta-# FROM Sailors S
utgupta-# GROUP BY S.rating
utgupta-# HAVING MIN(S.age) > (SELECT AVG(S2.age)
utgupta(#

utgupta(#     FROM Sailors S2);
 rating | min
--------+------
      7 |   35
      8 | 55.5
(2 rows)
```

sid	sname	rating	age
22	Dustin	7	45.0
31	Lubber	8	55.5
71	Zorba	10	16.0
64	Horatio	7	35.0
92	Frodo	1	28.0
38	Sam	1	30.0
29	Brutus	1	33.0
58	Rusty	10	35.0

More queries from Lecture 5 (cont.)

- Reference: Lecture 5 Slides

Customers

cid	cname	level	type	age
36	Cho	Beginner	snowboard	18
34	Luke	Inter	snowboard	25
87	Ice	Advanced	ski	20
39	Paul	Beginner	ski	33

Activities

cid	slopeid	day
36	s3	01/05/09
36	s1	01/06/09
36	s1	01/07/09
87	s2	01/07/09
87	s1	01/07/09
34	s2	01/05/09

Slopes

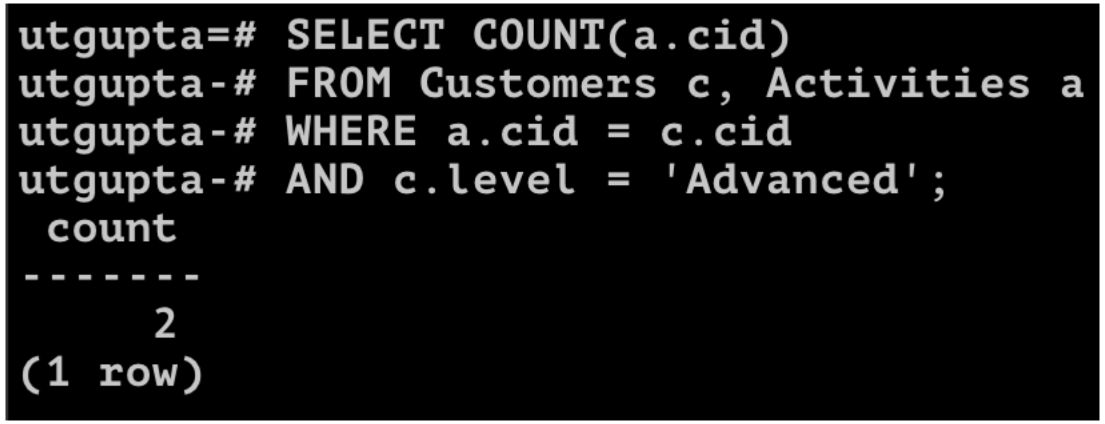
slopeid	name	color
s1	Mountain Run	blue
s2	Olympic Lady	black
s3	Magic Carpet	blue
s4	KT-22	green

More queries from Lecture 5 (cont.)

- Reference: Lecture 5 Slides

Find the number of activities done by advanced customers.

```
SELECT COUNT(a.cid)
FROM Customers c, Activities a
WHERE a.cid = c.cid
AND c.level = 'Advanced';
```



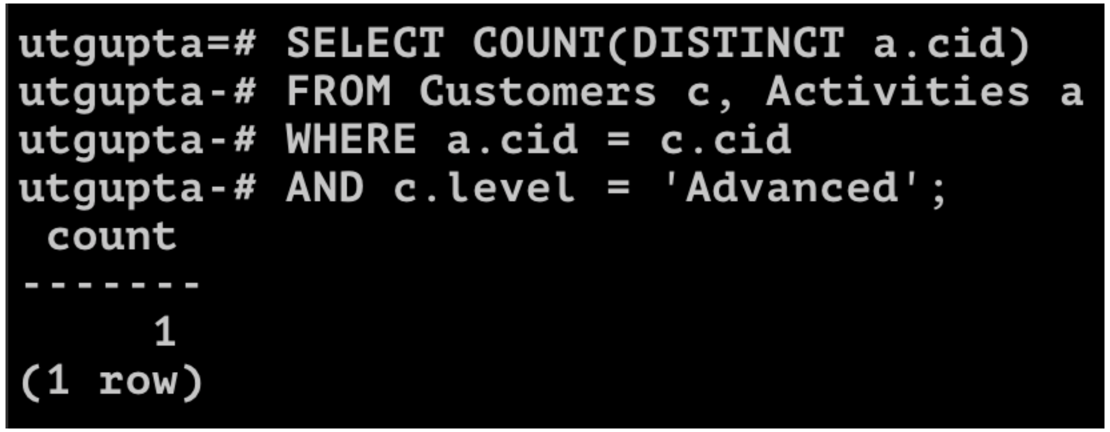
```
utgupta=# SELECT COUNT(a.cid)
utgupta-# FROM Customers c, Activities a
utgupta-# WHERE a.cid = c.cid
utgupta-# AND c.level = 'Advanced';
 count
-------
     2
(1 row)
```

More queries from Lecture 5 (cont.)

- Reference: Lecture 5 Slides

Find the number of activities done by DIFFERENT advanced customers.

```
SELECT COUNT(DISTINCT a.cid)
FROM Customers c, Activities a
WHERE a.cid = c.cid
AND c.level = 'Advanced';
```



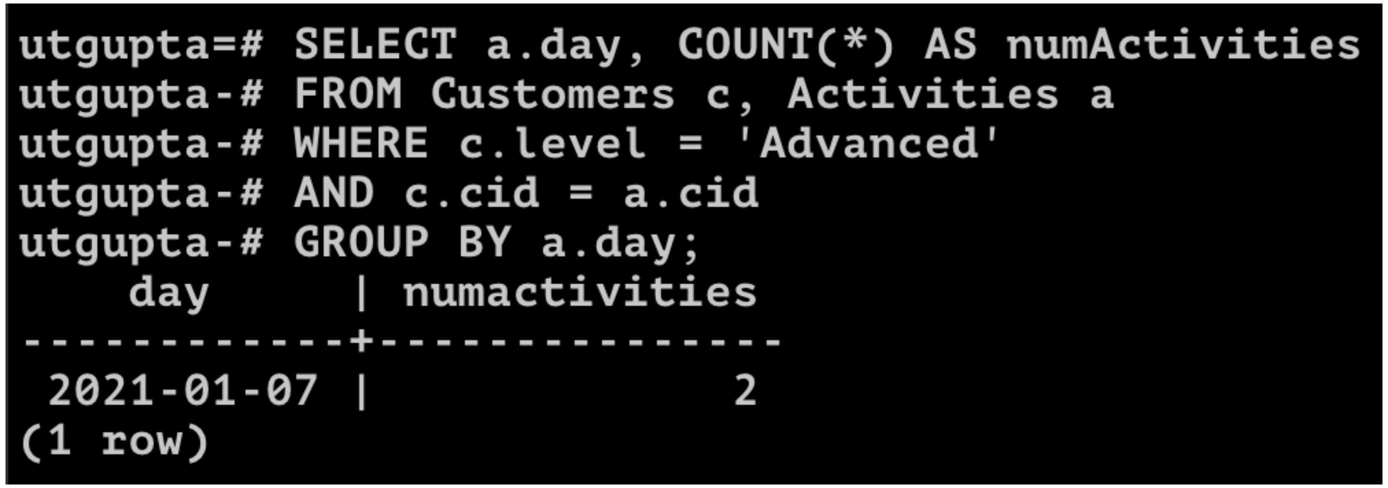
```
utgupta=# SELECT COUNT(DISTINCT a.cid)
utgupta-# FROM Customers c, Activities a
utgupta-# WHERE a.cid = c.cid
utgupta-# AND c.level = 'Advanced';
 count
-------
     1
(1 row)
```

More queries from Lecture 5 (cont.)

- Reference: Lecture 5 Slides

For each day, find the number of activities that are done by advanced Customers.

```
SELECT a.day, COUNT(a.cid) AS numActivities
FROM Customers c, Activities a
WHERE c.level = 'Advanced'
AND c.cid = a.cid
GROUP BY a.day;
```



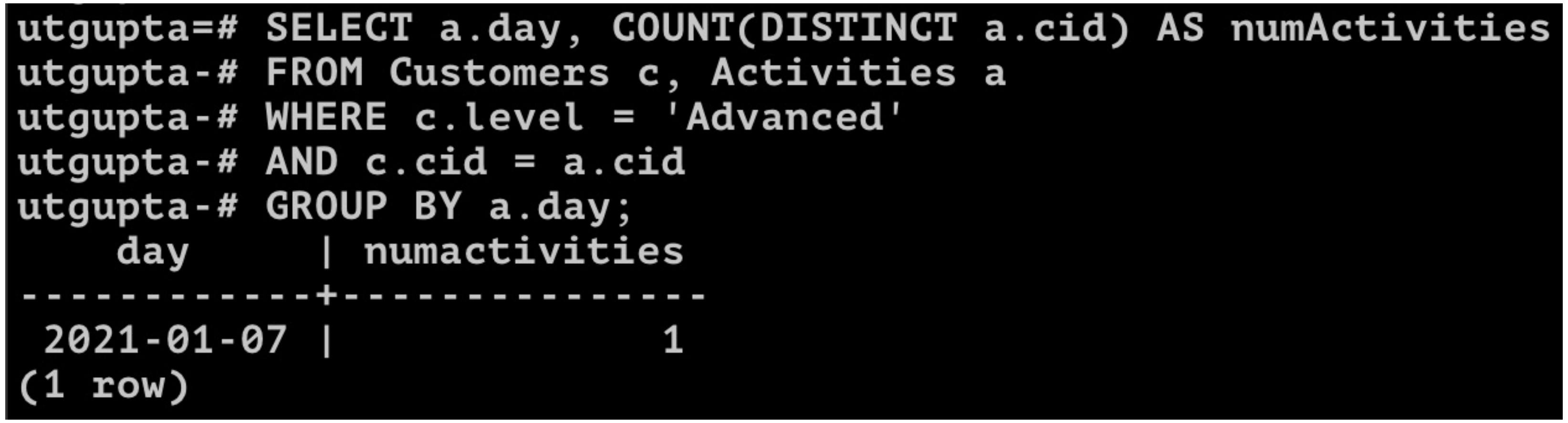
```
utgupta=# SELECT a.day, COUNT(*) AS numActivities
utgupta-# FROM Customers c, Activities a
utgupta-# WHERE c.level = 'Advanced'
utgupta-# AND c.cid = a.cid
utgupta-# GROUP BY a.day;
    day     | numactivities
------------+---------------
 2021-01-07 |             2
(1 row)
```

More queries from Lecture 5 (cont.)

- Reference: Lecture 5 Slides

For each day, find the number of different advanced Customers who did at least one of the activity.

```
SELECT a.day, COUNT(DISTINCT a.cid) AS numActivities
FROM Customers c, Activities a
WHERE c.level = 'Advanced'
AND c.cid = a.cid
GROUP BY a.day;
```



```
utgupta=# SELECT a.day, COUNT(DISTINCT a.cid) AS numActivities
utgupta-# FROM Customers c, Activities a
utgupta-# WHERE c.level = 'Advanced'
utgupta-# AND c.cid = a.cid
utgupta-# GROUP BY a.day;
    day     | numactivities
------------+---------------
 2021-01-07 |             1
(1 row)
```

Midterm

1. The midterm will be held in person during normal class time.
2. Some questions on the midterm may be very similar to questions from your Gradiance homeworks.

-----IMPORTANT POINTS-----

You can bring a cheat sheet to the midterm.
One piece of paper with anything written/typed on the front or back or two pieces of paper with one side used on each.

Remember to bring your student id. It will be checked when you turn in your exam

Midterm (cont.)

- **Topics to review**

This is not an all inclusive list.

- Three value logic
- The ACID database properties
- Writing queries, especially ones with GROUP BY and HAVING
- Writing database modification statements (UPDATE, DELETE and INSERT)
- Equivalence of two queries
- Subqueries and set operators (i.e. EXISTS, NOT EXISTS, IN, ALL, ANY)
- Behavior of aggregate functions (such as how they behave when NULL values are present)
- Given a query and a relation instance, find the expected output
- Legal versus illegal SQL statements

Create commands (Students)

```
CREATE TABLE STUDENT (
        S_ID INT,
        S_NAME VARCHAR(30),
        SSN CHAR(9),
        DOB DATE DEFAULT '2000-01-01' NOT NULL,
        GPA NUMERIC(3,2),
        HAS_GRAD BOOL,
        PHONE CHAR(10),
        EMAIL VARCHAR(50),
        PRIMARY KEY (S_ID),
        UNIQUE (EMAIL),
        UNIQUE (PHONE, S_NAME),
        UNIQUE (SSN)
);
```

```
CREATE TABLE COURSES (
        C_ID INT PRIMARY KEY,
        C_NAME CHAR(9) UNIQUE,
        CREDITS INT DEFAULT 5 NOT NULL
);
```

```
CREATE TABLE ENROLMENT (
        S_ID INT REFERENCES STUDENT,
        C_ID INT REFERENCES COURSES,
        DROPPED BOOL NOT NULL,
        PRIMARY KEY(S_ID, C_ID)
);
```

Insert Commands (Students)

Student

```
INSERT INTO STUDENT VALUES (1, 'DAVID', '1234', '1997-01-01', 3.8, false, '0123456', 'david@pqrs.com');
INSERT INTO STUDENT VALUES (2, 'JULIA', '4567', '2000-02-18', 3.9, false, '1234589', 'julia@pqrs.com');
INSERT INTO STUDENT VALUES (3, 'DAVID', '2468', '2000-02-01', 3.8, false, '9827123', 'david1@pqrs.com');
INSERT INTO STUDENT VALUES (4, 'JOEL', '3412', '2002-03-23', 3.34, false, NULL, 'joel@pqrs.com');
INSERT INTO STUDENT (S_ID, S_NAME, SSN, GPA, HAS_GRAD, PHONE, EMAIL) VALUES (5, 'ABY', '4321', 3.32, true, NULL, NULL);
```

Courses

```
INSERT INTO COURSES VALUES (180, 'DB 1');
INSERT INTO COURSES VALUES (181, 'DB 2');
INSERT INTO COURSES VALUES (215, 'DB GRAD 1');
INSERT INTO COURSES VALUES (160, 'CG');
INSERT INTO COURSES VALUES (280, 'CV SEM', 2);
```

Enrolment

```
INSERT INTO ENROLMENT VALUES (1, 180, false);
INSERT INTO ENROLMENT VALUES (5, 160, false);
INSERT INTO ENROLMENT VALUES (5, 215, false);
INSERT INTO ENROLMENT VALUES (1, 181, true);
INSERT INTO ENROLMENT VALUES (2, 160, false);
INSERT INTO ENROLMENT VALUES (5, 180, false);
INSERT INTO ENROLMENT VALUES (2, 280, false);
INSERT INTO ENROLMENT VALUES (5, 181, false);
INSERT INTO ENROLMENT VALUES (2, 181, true);
INSERT INTO ENROLMENT VALUES (3, 180, false);
INSERT INTO ENROLMENT VALUES (5, 280, false);
```

Create commands (Lecture 4)

```
CREATE TABLE customers (

   cid INT PRIMARY KEY,

   cname VARCHAR(20),

   level CHAR(20),

   type CHAR(20),

   age INT);
```

```
CREATE TABLE slopes (
   slopeid CHAR(3) PRIMARY KEY,
   name VARCHAR(20),
   color VARCHAR(20)
);
```

```
CREATE TABLE activities (
   cid INT,
   slopeid CHAR(3),
   day DATE,
   PRIMARY KEY (cid, slopeid, day),
   FOREIGN KEY (cid) REFERENCES customers,
   FOREIGN KEY (slopeid) REFERENCES slopes
);
```

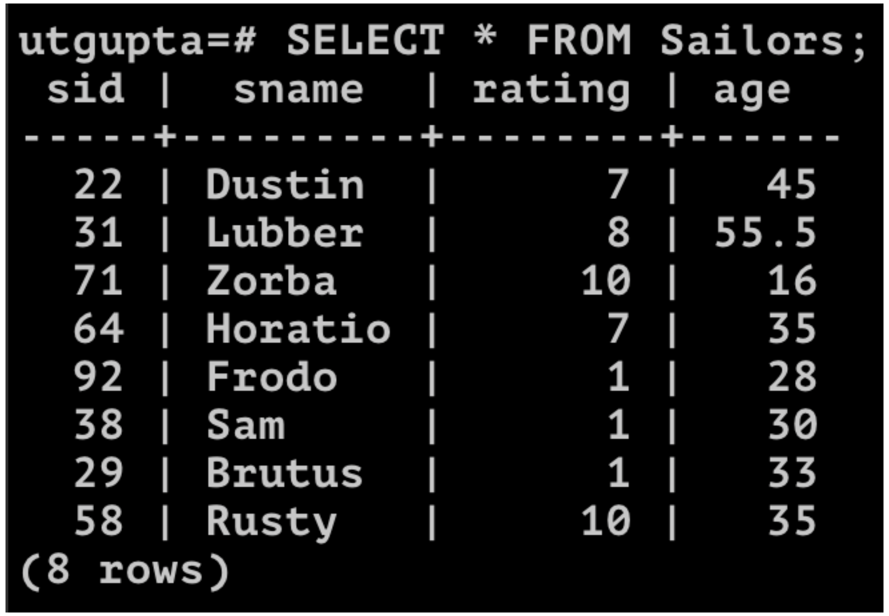
Insert commands (Lecture 4)

```
INSERT INTO customers VALUES (36, 'Cho', 'Beginner', 'snowboard', 18);
INSERT INTO customers VALUES (34, 'Luke', 'Inter', 'snowboard', 25);
INSERT INTO customers VALUES (87, 'Ice', 'Advanced', 'ski', 20);
INSERT INTO customers VALUES (39, 'Paul', 'Beginner', 'ski', 33);

INSERT INTO slopes VALUES ('s1', 'Mountain Run', 'blue');
INSERT INTO slopes VALUES ('s2', 'Olympic Lady', 'black');
INSERT INTO slopes VALUES ('s3', 'Magic Carpet', 'green');
INSERT INTO slopes VALUES ('s4', 'KT-22', 'black');

INSERT INTO activities VALUES (36, 's3', '01/05/21');
INSERT INTO activities VALUES (36, 's1', '01/06/21');
INSERT INTO activities VALUES (36, 's1', '01/07/21');
INSERT INTO activities VALUES (87, 's2', '01/07/21');
INSERT INTO activities VALUES (87, 's1', '01/07/21');
INSERT INTO activities VALUES (34, 's2', '01/05/21');
```

Create and Insert commands (Lecture 5)

```
CREATE TABLE Sailors (
sid INT PRIMARY KEY,
sname VARCHAR(20),
rating INT,
age FLOAT
);
```

```
INSERT INTO Sailors VALUES (22, 'Dustin', 7, 45.0);
INSERT INTO Sailors VALUES (31, 'Lubber', 8, 55.5);
INSERT INTO Sailors VALUES (71, 'Zorba', 10, 16.0);
INSERT INTO Sailors VALUES (64, 'Horatio', 7, 35.0);
INSERT INTO Sailors VALUES (92, 'Frodo', 1, 28.0);
INSERT INTO Sailors VALUES (38, 'Sam', 1, 30.0);
INSERT INTO Sailors VALUES (29, 'Brutus', 1, 33.0);
INSERT INTO Sailors VALUES (58, 'Rusty', 10, 35.0);
```



```
utgupta=# SELECT * FROM Sailors;
 sid |  sname   | rating | age
-----+----------+--------+------
  22 | Dustin   |      7 |   45
  31 | Lubber   |      8 | 55.5
  71 | Zorba    |     10 |   16
  64 | Horatio  |      7 |   35
  92 | Frodo    |      1 |   28
  38 | Sam      |      1 |   30
  29 | Brutus   |      1 |   33
  58 | Rusty    |     10 |   35
(8 rows)
```