**SQL Lab**

Julie Fisher

In collaboration with David Keough, Karelys Osuna, Brandon Harden, Nick Enko, Gilbert Diaz, Sinney Chan, and Austin White

1. Defining the Database
   1. Primary keys i – vi were chosen as they are unique identifiers for each of the specific table. Primary key g was created from an auto increment, as there is not one unique key.
      1. Student: S\_ID
      2. Faculty: F\_ID
      3. Course: Course\_ID
      4. Course\_Section: C\_Sec\_ID
      5. Location: LOC\_ID
      6. Term: Term\_ID
      7. Enrollment: enr\_ID

Foreign keys:

1. Enrollment
   1. S\_ID – cascade updates, no action on delete
   2. C\_Sec\_ID - cascade updates, restrict delete
2. Course\_Section
   1. Course\_ID – cascade updates, no action on delete
   2. Term\_ID – cascade updates, no action on delete
   3. F\_ID – cascade updates, no action on delete
   4. LOC\_ID - cascade updates, no action on delete
3. Student
   1. F\_ID - cascade updates, restrict delete
4. Faculty
   1. LOD\_ID - cascade updates, restrict delete
   2. All CREATE TABLE commands:

CREATE SCHEMA `LIPSCOMB\_Student\_Registration`;

USE LIPSCOMB\_Student\_Registration;

CREATE TABLE `COURSE` (

`COURSE\_ID` INTEGER PRIMARY KEY,

`COURSE\_NO` VARCHAR(10) NOT NULL,

`COURSE\_NAME` VARCHAR(50) NOT NULL,

`CREDITS` TINYINT NOT NULL);

CREATE TABLE `LOCATION` (

`LOC\_ID` INTEGER PRIMARY KEY,

`BLDG\_CODE` VARCHAR(10) NOT NULL,

`ROOM` VARCHAR(5) NOT NULL,

`CAPACITY` INTEGER NOT NULL

);

CREATE TABLE `FACULTY` (

`F\_ID` INTEGER PRIMARY KEY,

`F\_LAST` VARCHAR(50) NOT NULL,

`F\_FIRST` VARCHAR(50) NOT NULL,

`F\_MI` VARCHAR(1),

`F\_PHONE` VARCHAR(15),

`F\_RANK` VARCHAR(30) NOT NULL,

`F\_SUPER` INTEGER,

`F\_PIN` VARCHAR(10) NOT NULL,

`LOC\_ID` INTEGER,

FOREIGN KEY (`LOC\_ID`)

REFERENCES location (`LOC\_ID`)

ON UPDATE CASCADE

ON DELETE RESTRICT);

CREATE TABLE `STUDENT` (

`S\_ID` INTEGER PRIMARY KEY,

`S\_LAST` VARCHAR(50) NOT NULL,

`S\_FIRST` VARCHAR(50) NOT NULL,

`S\_MI` VARCHAR(1) NOT NULL,

`S\_ADDRESS` VARCHAR(75) NOT NULL,

`S\_CITY` VARCHAR(50) NOT NULL,

`S\_STATE` VARCHAR(2) NOT NULL,

`S\_ZIP` VARCHAR(5) NOT NULL,

`S\_PHONE` VARCHAR(15) NOT NULL,

`S\_CLASS` VARCHAR(5) NOT NULL,

S\_DOB DATE NOT NULL,

`S\_PIN` VARCHAR(4) NOT NULL,

`DATE\_ENROLLED` DATE,

`F\_ID` INTEGER,

FOREIGN KEY (`F\_ID`)

REFERENCES FACULTY (`F\_ID`)

ON UPDATE CASCADE

ON DELETE RESTRICT);

CREATE TABLE `TERM` (

`TERM\_ID` INTEGER PRIMARY KEY,

`TERM\_SEMESTER` VARCHAR(20) NOT NULL,

`TERM\_YEAR` YEAR NOT NULL,

`STATUS` VARCHAR(6) NOT NULL,

`START\_DATE` DATE NOT NULL

);

CREATE TABLE `COURSE\_SECTION` (

`C\_SEC\_ID` INTEGER PRIMARY KEY,

`COURSE\_ID` INTEGER NOT NULL,

`TERM\_ID` INTEGER NOT NULL,

`SEC\_NUM` INTEGER NOT NULL,

`F\_ID` INTEGER NOT NULL,

`MTG\_DAYS` VARCHAR(7) NOT NULL,

`START\_TIME` TIME NOT NULL,

`END\_TIME` TIME NOT NULL,

`LOC\_ID` INTEGER,

`MAX\_ENRL` INTEGER,

FOREIGN KEY (`COURSE\_ID`)

REFERENCES course (`COURSE\_ID`)

ON UPDATE CASCADE

ON DELETE NO ACTION,

FOREIGN KEY (`TERM\_ID`)

REFERENCES term (`TERM\_ID`)

ON UPDATE CASCADE

ON DELETE NO ACTION,

FOREIGN KEY (`F\_ID`)

REFERENCES faculty (`F\_ID`)

ON UPDATE CASCADE

ON DELETE NO ACTION,

FOREIGN KEY (`LOC\_ID`)

REFERENCES location (`LOC\_ID`)

ON UPDATE CASCADE

ON DELETE NO ACTION

);

CREATE TABLE `ENROLLMENT` (

`ENR\_ID` INTEGER PRIMARY KEY AUTO\_INCREMENT,

`S\_ID` INTEGER NOT NULL,

`C\_SEC\_ID` INTEGER NOT NULL,

`GRADE` VARCHAR(5) DEFAULT NULL,

FOREIGN KEY (`S\_ID`)

REFERENCES STUDENT (`S\_ID`)

ON UPDATE CASCADE

ON DELETE NO ACTION,

FOREIGN KEY (`C\_SEC\_ID`)

REFERENCES COURSE\_SECTION (`C\_SEC\_ID`)

ON UPDATE CASCADE

ON DELETE RESTRICT

);

alter table faculty

add foreign key (F\_SUPER)

references faculty (F\_ID);

* 1. There are no changes when the record attached to a foreign key is deleted. The original value will remain. In addition, deletion was restricted for C\_Sec\_ID in the Enrollment table, F\_ID in the Student table, and LOC\_ID in the Faculty table. These restrictions were created to ensure reassignment of course section, faculty, and location for associated records before the foreign key is removed from the system. For example, a new faculty advisor would need to be assigned to a student before the exiting faculty could be deleted. Upon change/update, that change will cascade to all tables.
  2. Choice of attribute constraints
     1. String length was limited to discourage SQL injection attacks
     2. Information such as first and last names, address info were required to ensure table integrity
     3. Info like phone numbers was allowed to be null to allow for when the information is unavailable
     4. Check F\_SUPER to ensure the faculty exists under F\_ID
     5. Ensure room doesn’t already have faculty assigned to it

1. Populating your Database
   1. Prepared data in Excel
      1. Converted date fields into YYYY-MM-DD format by column
      2. Split TERM\_DESC into two columns – semester and year
      3. Added /N to empty fields to represent null values
2. Checking your Database
   1. Course\_section tuple insert failures
      1. ERROR CODE:1062 DUPLICATE ENTRY '12' FOR KEY 'PRIMARY'
      2. ERROR CODE:1062 DUPLICATE ENTRY '12' FOR KEY 'PRIMARY'
      3. ERROR CODE:1062 DUPLICATE ENTRY '2' FOR KEY 'PRIMARY'
   2. Faculty tuple insert failures
      1. ERROR CODE:1062 DUPLICATE ENTRY '4' FOR KEY 'PRIMARY'
      2. Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`lipscomb\_student\_registration`.`faculty`, CONSTRAINT `faculty\_ibfk\_1` FOREIGN KEY (`LOC\_ID`) REFERENCES `location` (`LOC\_ID`) ON UPDATE CASCADE)
      3. Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`lipscomb\_student\_registration`.`faculty`, CONSTRAINT `faculty\_ibfk\_2` FOREIGN KEY (`F\_SUPER`) REFERENCES `faculty` (`F\_ID`)) – Added foreign key to check referential integrity of F\_SUPER against F\_ID
      4. Trigger not created, but should error due to the LOC\_ID room already having been assigned to a different faculty member
   3. Error Code: 1062. Duplicate entry '4' for key 'PRIMARY'
   4. Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails (`lipscomb\_student\_registration`.`faculty`, CONSTRAINT `faculty\_ibfk\_1` FOREIGN KEY (`LOC\_ID`) REFERENCES `location` (`LOC\_ID`) ON UPDATE CASCADE)
   5. Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails (`lipscomb\_student\_registration`.`course\_section`, CONSTRAINT `course\_section\_ibfk\_2` FOREIGN KEY (`TERM\_ID`) REFERENCES `term` (`TERM\_ID`) ON DELETE NO ACTION ON UPDATE CASCADE)
3. Simple Database Queries
   1. Students with As and Bs
      1. Select distinct S.S\_ID,S\_LAST,S\_FIRST

From Enrollment E

Join student S

On E.S\_ID = S.S\_ID

Where E.grade in ("A","B")

* + 1. Tammy Jones, Jorge Perez, Lisa Johnson
  1. Terms for 2007 academic year
     1. SELECT \*

FROM TERM T

WHERE TERM\_YEAR LIKE 2007

* + 1. 2 Spring 2007 CLOSED 2008-01-09

3 Summer 2007 CLOSED 2006-05-15

4 Fall 2007 CLOSED 2007-08-28

* 1. Building code info in ascending order
     1. Select BLDG\_CODE, ROOM, CAPACITY

from Location

ORDER BY BLDG\_CODE, ROOM

* + 1. BUS 105 42

BUS 211 55

BUS 402 1

BUS 404 35

BUS 421 35

BUS 424 1

BUS 433 1

CR 101 150

CR 103 35

CR 105 35

CR 202 40

LIB 217 2

LIB 222 1

* 1. Tuition charges
     1. SELECT COURSE\_NO, COURSE\_NAME,

concat('$', format(CREDITS \* 730, 2)) AS TUITION\_CHARGE

FROM COURSE

* + 1. IT 101 Intro. to Info. Tech. $2,190.00

IS 301 Systems Analysis $2,190.00

IT 240 Intro. to Database Systems $2,190.00

CS 120 Intro. To Programming in C++ $2,190.00

IT 451 Web-Based Systems $2,190.00

* 1. Max enrollment for course sections with max, min, average for Spring 2008
     1. SELECT A.C\_SEC\_ID, A.SUM\_MAX\_ENROLLMENTS, B.AVERAGE\_CURRENT\_ENROLLMENT,

B.MAX\_CURRENT\_ENROLLMENT, B.MIN\_CURRENT\_ENROLLMENT

FROM (

SELECT E.C\_SEC\_ID, SUM(CS.MAX\_ENRL) AS SUM\_MAX\_ENROLLMENTS

FROM ENROLLMENT E

LEFT JOIN COURSE\_SECTION CS

ON E.C\_SEC\_ID = CS.C\_SEC\_ID

GROUP BY C\_SEC\_ID) A,

(

SELECT avg(CURRENT\_ENROLLMENT) AVERAGE\_CURRENT\_ENROLLMENT,

MAX(CURRENT\_ENROLLMENT) MAX\_CURRENT\_ENROLLMENT, MIN(CURRENT\_ENROLLMENT) MIN\_CURRENT\_ENROLLMENT

FROM (

SELECT E.C\_SEC\_ID, COUNT(S\_ID) AS CURRENT\_ENROLLMENT

FROM ENROLLMENT E

LEFT JOIN COURSE\_SECTION CS

ON E.C\_SEC\_ID = CS.C\_SEC\_ID

LEFT JOIN COURSE C

ON CS.COURSE\_ID = C.COURSE\_ID

LEFT JOIN TERM T

ON CS.TERM\_ID = T.TERM\_ID

WHERE TERM\_SEMESTER = 'Summer' and TERM\_YEAR = 2008

GROUP BY E.C\_SEC\_ID

) A

) B

* + 1. 1 560 2.6667 3 2

4 35 2.6667 3 2

5 70 2.6667 3 2

6 60 2.6667 3 2

9 105 2.6667 3 2

11 150 2.6667 3 2

12 105 2.6667 3 2

13 70 2.6667 3 2

* 1. Total number of courses Lisa has a grade for
     1. SELECT COUNT(DISTINCT C.COURSE\_ID) COURSE\_COUNT

FROM ENROLLMENT E

JOIN STUDENT S

ON E.S\_ID = S.S\_ID

JOIN COURSE\_SECTION CS

ON E.C\_SEC\_ID = CS.C\_SEC\_ID

JOIN COURSE C

ON CS.COURSE\_ID = C.COURSE\_ID

WHERE S\_FIRST = 'LISA'

AND S\_LAST = 'JOHNSON'

AND GRADE IS NOT NULL

* + 1. 3
  1. Building code with capacity over 100
     1. SELECT BLDG\_CODE, SUM(CAPACITY)

FROM LOCATION L

GROUP BY BLDG\_CODE

HAVING SUM(CAPACITY) > 100

* + 1. BUS 170

CR 260

* 1. Student and faculty table join
     1. select S.S\_ID, S.S\_LAST, S.S\_FIRST, S.F\_ID, F.F\_LAST

FROM STUDENT S

JOIN FACULTY F

ON S.F\_ID = F.F\_ID

* + 1. 1 Jones Tammy 1 Marx

2 Perez Jorge 1 Marx

3 Marsh John 1 Marx

4 Smith Mike 2 Zhulin

5 Johnson Lisa 4 Brown

6 Nguyen Ni 3 Langley

* 1. Faculty teaching in Summer 2008
     1. SELECT F.F\_LAST

FROM COURSE\_SECTION CS

JOIN FACULTY F

ON CS.F\_ID = F.F\_ID

JOIN TERM T

ON CS.TERM\_ID = T.TERM\_ID

WHERE TERM\_SEMESTER = 'Summer' AND TERM\_YEAR = 2008

* + 1. Marx

Zhulin

Langley

* 1. Tammy’s courses and grades
     1. SELECT S.S\_ID, COURSE\_NAME, GRADE

FROM ENROLLMENT E

JOIN STUDENT S

ON E.S\_ID = S.S\_ID

JOIN COURSE\_SECTION CS

ON E.C\_SEC\_ID = CS.C\_SEC\_ID

JOIN COURSE C

ON CS.COURSE\_ID = C.COURSE\_ID

WHERE S\_FIRST = 'Tammy'

AND S\_LAST = 'Jones'

* + 1. 1 Intro. to Info. Tech. A

1 Systems Analysis A

1 Intro. to Database Systems B

1 Web-Based Systems B

* 1. Student and faculty table join #2
     1. SELECT S\_LAST, S\_FIRST, S\_PHONE

FROM STUDENT S

UNION

SELECT F\_LAST, F\_FIRST, F\_PHONE

FROM FACULTY F

* + 1. Jones Tammy 3250987654

Perez Jorge 3258765432

Marsh John 3257654321

Smith Mike 3256543210

Johnson Lisa 3255432109

Nguyen Ni 3254321098

Marx Teresa 3251234567

Zhulin Mark 3252345678

Langley Colin 3253456789

Brown Jonnel 3254567890

Sealy James 3255678901

1. Slightly Complex Database Queries
   1. Students with the same S\_CLASS value as Jorge Perez
      1. SELECT S\_FIRST, S\_LAST

FROM STUDENT S

WHERE S\_CLASS = (

SELECT S\_CLASS

FROM STUDENT S

WHERE S\_FIRST = 'Jorge' AND S\_LAST = 'Perez')

* + 1. Tammy Jones

Jorge Perez

* 1. Students enrolled in the same sections as Jorge Perez
     1. SELECT DISTINCT S\_LAST, S\_FIRST

FROM STUDENT S

JOIN ENROLLMENT E

ON S.S\_ID = E.S\_ID

WHERE E.C\_SEC\_ID IN (

SELECT E.C\_SEC\_ID

FROM STUDENT S

JOIN ENROLLMENT E

ON S.S\_ID = E.S\_ID

WHERE S\_FIRST = 'Jorge' AND S\_LAST = 'Perez')

* + 1. Jones Tammy

Perez Jorge

Marsh John

Johnson Lisa

* 1. Students with the same S\_CLASS value as Jorge Perez and in the same sections
     1. SELECT DISTINCT S\_LAST, S\_FIRST

FROM STUDENT S

JOIN ENROLLMENT E

ON S.S\_ID = E.S\_ID

WHERE E.C\_SEC\_ID IN (

SELECT E.C\_SEC\_ID

FROM STUDENT S

JOIN ENROLLMENT E

ON S.S\_ID = E.S\_ID

WHERE S\_FIRST = 'Jorge'

AND S\_LAST = 'Perez')

AND S\_CLASS = (

SELECT S\_CLASS

FROM STUDENT S

WHERE S\_FIRST = 'Jorge'

AND S\_LAST = 'Perez')

* + 1. Jones Tammy

Perez Jorge

* 1. Students who have taken courses with Jorge Perez in the CR building
     1. SELECT S.S\_FIRST, S.S\_LAST

FROM STUDENT S

JOIN ENROLLMENT E

ON S.S\_ID = E.S\_ID

JOIN COURSE\_SECTION CS

ON E.C\_SEC\_ID = CS.C\_SEC\_ID

WHERE CS.C\_SEC\_ID IN (

SELECT CS.C\_SEC\_ID

FROM STUDENT S

JOIN ENROLLMENT E

ON S.S\_ID = E.S\_ID

JOIN COURSE\_SECTION CS

ON E.C\_SEC\_ID = CS.C\_SEC\_ID

JOIN LOCATION L

ON CS.LOC\_ID = L.LOC\_ID

WHERE S.S\_ID IN (

SELECT S.S\_ID

FROM STUDENT S

WHERE S.S\_FIRST = 'Jorge'

and s.S\_LAST = 'Perez')

AND BLDG\_CODE = 'CR')

* + 1. Tammy Jones

Jorge Perez

John Marsh

Lisa Johnson

* 1. Courses taken by students who aren’t seniors and courses in term 6
     1. SELECT COURSE\_NAME

FROM COURSE C

JOIN COURSE\_SECTION CS

ON C.COURSE\_ID = CS.COURSE\_ID

JOIN ENROLLMENT E

ON CS.C\_SEC\_ID = E.C\_SEC\_ID

JOIN STUDENT S

ON E.S\_ID = S.S\_ID

WHERE S\_CLASS <> 'SR'

UNION

SELECT COURSE\_NAME

FROM COURSE C

JOIN COURSE\_SECTION CS

ON C.COURSE\_ID = CS.COURSE\_ID

JOIN ENROLLMENT E

ON CS.C\_SEC\_ID = E.C\_SEC\_ID

JOIN STUDENT S

ON E.S\_ID = S.S\_ID

WHERE TERM\_ID = 6

* + 1. Intro. to Info. Tech.

Systems Analysis

Intro. to Database Systems

Web-Based Systems

* 1. Courses in term 6 taken by students who aren’t seniors
     1. SELECT DISTINCT COURSE\_NAME

FROM COURSE C

JOIN COURSE\_SECTION CS

ON C.COURSE\_ID = CS.COURSE\_ID

JOIN ENROLLMENT E

ON CS.C\_SEC\_ID = E.C\_SEC\_ID

JOIN STUDENT S

ON E.S\_ID = S.S\_ID

WHERE S\_CLASS <> 'SR'

AND COURSE\_NAME IN (

SELECT COURSE\_NAME

FROM COURSE C

JOIN COURSE\_SECTION CS

ON C.COURSE\_ID = CS.COURSE\_ID

JOIN ENROLLMENT E

ON CS.C\_SEC\_ID = E.C\_SEC\_ID

JOIN STUDENT S

ON E.S\_ID = S.S\_ID

WHERE TERM\_ID = 6)

* + 1. Intro. to Info. Tech.

Systems Analysis

Intro. to Database Systems

* 1. Courses taken by freshman, sophomores, and juniors, but weren’t in term 6
     1. SELECT DISTINCT COURSE\_NAME

FROM COURSE C

JOIN COURSE\_SECTION CS

ON C.COURSE\_ID = CS.COURSE\_ID

JOIN ENROLLMENT E

ON CS.C\_SEC\_ID = E.C\_SEC\_ID

JOIN STUDENT S

ON E.S\_ID = S.S\_ID

WHERE S\_CLASS in ('FR','SO','JR')

AND COURSE\_NAME not IN (

SELECT COURSE\_NAME

FROM COURSE C

JOIN COURSE\_SECTION CS

ON C.COURSE\_ID = CS.COURSE\_ID

JOIN ENROLLMENT E

ON CS.C\_SEC\_ID = E.C\_SEC\_ID

JOIN STUDENT S

ON E.S\_ID = S.S\_ID

WHERE TERM\_ID = 6)

* + 1. Web-Based Systems
  1. All junior faculty members and their supervisors
     1. SELECT F.F\_FIRST, F.F\_LAST, F1.F\_FIRST, F1.F\_LAST

FROM FACULTY F

JOIN FACULTY F1

ON F.F\_SUPER = F1.F\_ID

* + 1. Teresa Marx Jonnel Brown

Colin Langley Jonnel Brown

James Sealy Mark Zhulin

1. Experimenting with Views
   1. Faculty view without F\_PIN
      1. CREATE VIEW faculty\_view as

SELECT F\_ID, F\_LAST, F\_FIRST, F\_MI, LOC\_ID, F\_PHONE, F\_RANK, F\_SUPER

FROM FACULTY F

* + 1. 1 Marx Teresa J 9 3251234567 Associate 4

2 Zhulin Mark M 10 3252345678 Full

3 Langley Colin A 12 3253456789 Assistant 4

4 Brown Jonnel D 11 3254567890 Full

5 Sealy James L 13 3255678901 Associate 2

* 1. Faculty view with added tuple
     1. insert into faculty\_view

(F\_ID,F\_LAST,F\_FIRST,F\_MI,LOC\_ID,F\_PHONE,F\_RANK,F\_SUPER)

values (6, "May", "Lisa", "I", 11, "3256789012", "Assistant", null)

* 1. Faculty view to confirm addition
     1. 1 Marx Teresa J 3251234567 Associate 4 6339 9

2 Zhulin Mark M 3252345678 Full 1121 10

3 Langley Colin A 3253456789 Assistant 4 9871 12

4 Brown Jonnel D 3254567890 Full 8297 11

5 Sealy James L 3255678901 Associate 2 6089 13

6 May Lisa I 3256789012 Assistant null 11

* 1. Effect of b to database: inserting a tuple into the view does change the underlying information. Had to update faculty table to include a default value for F\_PIN otherwise insert doesn’t work.
  2. Faculty view with location
     1. SELECT F\_FIRST, F\_LAST, BLDG\_CODE, ROOM

FROM faculty\_view FV

JOIN LOCATION L

ON FV.LOC\_ID = L.LOC\_ID

* + 1. Teresa Marx BUS 424

Mark Zhulin BUS 402

Colin Langley LIB 217

Jonnel Brown BUS 433

James Sealy LIB 222

Lisa May BUS 433

* 1. Remove faculty view: DROP VIEW faculty\_view
  2. Effect of f to database: Removing the view doesn’t alter the underlying table, including changes made to the view: Lisa May is still listed in the faculty table.

1. Updating the Database
   1. Change the room to BUS 211 for all classes taught for Brown
      1. CREATE TEMPORARY TABLE tmpUPDATE (

C\_SEC\_ID INT);

INSERT INTO tmpUPDATE

SELECT C\_SEC\_ID FROM COURSE\_SECTION CS

JOIN FACULTY F

ON CS.F\_ID = F.F\_ID

WHERE F.F\_LAST = 'BROWN';

UPDATE COURSE\_SECTION

SET LOC\_ID = 8

WHERE C\_SEC\_ID IN (

SELECT C\_SEC\_ID FROM tmpUPDATE);

* + 1. 1 1 4 1 2 MWF 10:00:00 10:50:00 1 140

2 1 4 2 3 TR 09:30:00 10:45:00 7 35

3 1 4 3 3 MWF 08:00:00 08:50:00 2 35

4 2 4 1 4 TR 11:00:00 12:15:00 8 35

5 2 5 2 4 TR 02:00:00 03:15:00 8 35

6 3 5 1 1 MWF 09:00:00 09:50:00 5 30

7 3 5 2 1 MWF 10:00:00 10:50:00 5 30

8 4 5 1 5 TR 08:00:00 09:15:00 3 35

9 5 5 1 2 MWF 02:00:00 02:50:00 5 35

10 5 5 2 2 MWF 03:00:00 03:50:00 5 35

11 1 6 1 1 MTWRF 08:00:00 09:30:00 1 50

12 2 6 1 2 MTWRF 08:00:00 09:30:00 6 35

13 3 6 1 3 MTWRF 08:00:00 09:30:00 5 35

* 1. Create and fill a new table, enrollent\_numbers, showing course number and number of students enrolled per section for Spring 2008
     1. CREATE TABLE enrollment\_numbers (

COURSE\_ID INT,

C\_SEC\_ID INT,

ENROLLMENTS INT);

INSERT INTO enrollment\_numbers

SELECT COURSE\_ID, CS.C\_SEC\_ID, COUNT(S\_ID) AS ENROLLMENTS

FROM COURSE\_SECTION CS

JOIN ENROLLMENT E

ON CS.C\_SEC\_ID = E.C\_SEC\_ID

JOIN TERM T

ON CS.TERM\_ID = T.TERM\_ID

WHERE T.TERM\_SEMESTER = 'SPRING' and TERM\_YEAR = 2008

GROUP BY COURSE\_ID, CS.C\_SEC\_ID

* + 1. select \* from enrollment\_numbers

2 5 2

3 6 2

5 9 3