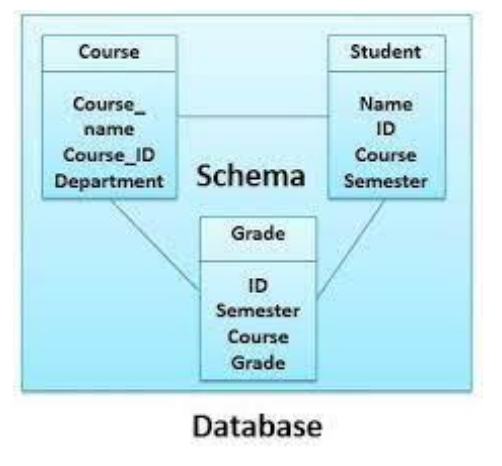
**DATABASE MANAGEMENT SYSTEM LAB (CS-262)**



**WEEK - 1:**

1. **Create the tables in the database.**

SQL> create table Course(Course\_name char(40),Course\_ID varchar(40),Department char(40));

Table created.

SQL> create table Student(Name char(40),ID varchar(40),Course varchar(40),Semester char(40));

Table created.

SQL> create table Grade(ID varchar(40),Semesterchar(40),Course varchar(40),Gradesnumber);

Table created.

SQL> select \* from tab;

TNAME TABTYPE CLUSTERID

------------------------------ ------- ----------

COURSE TABLE

GRADE TABLE

STUDENT TABLE

1. **Alter the course table by adding a new attribute instructor.**

SQL> ALTER TABLE COURSE ADD INSTRUCTOR VARCHAR(40);

Table altered.

1. **Alter the student table by modifying the datatype of Student\_ID**

SQL> ALTER TABLE STUDENT MODIFY ID NUMBER;

Table altered.

1. **Alter the course table by dropping instructor attribute**

SQL> ALTER TABLE COURSE DROP COLUMN INSTRUCTOR;

Table altered.

1. **Show the structure of the grade table.**

SQL> DESC GRADE;

Name Null? Type

---------------------------------------------------------------------------------------------------

ID VARCHAR2(40)

SEMESTER VARCHAR2(40)

COURSE CHAR(40)

GRADE VARCHAR2(40)

1. **Load the database with data using insert statements.**

SQL> INSERT INTO COURSE VALUES('DBMS','CS226','CSE');

1 row created.

SQL> INSERT INTO STUDENT VALUES('SUFIYAN','Y21CS162','CS226','4th SEM’);

1 row created.

SQL>INSERT INTO GRADE VALUES('Y21CS162','4th SEM','CSE',10);

1. row created.
2. **Retrieve the contents of a table using select statements.**

SQL> SELECT \* FROM COURSE;

COURSE\_NAME COURSE\_ID DEPARTMENT

---------------------------------------- ---------------------------------------- ------------------

DBMS CS226 CSE

WT CS225 CSE

CS CS221CSE

SQL>SELECT COURSE\_ID FROM COURSE;

COURSE\_ID

----------------

CS226

CS225

CS221

SQL>SELECT \* FROM COURSE WHERE COURSE\_ID=’CS226’;

COURSE\_NAME COURSE\_ID DEPARTMENT

---------------------------------------- ---------------------------------------- ------------------

DBMS CS226 CSE

1. **Modify the contents of tables using update statements.**

SQL> UPDATE COURSE SET COURSE\_ID=’CS224' WHERE COURSE\_ID='CS221';

1 row updated.

SQL> SELECT \* FROM COURSE;

COURSE\_NAME COURSE\_ID DEPARTMENT

---------------------------------------- ---------------------------------------- ------------------

DBMS CS226 CSE

WT CS225 CSE

CS CS221 CSE

1. **Delete selected records from the tables.**

SQL> DELETE FROM GRADE WHERE SEMESTER='THIRD SEMESTER';

1 row deleted.

SQL> DELETE FROM STUDENT WHERE ID=161;

1 row deleted.

SQL> DELETE FROM COURSE WHERE COURSE\_ID='Y21CS161';

1 row deleted.

1. **Truncate a table.**

SQL> TRUNCATE TABLE GRADE;

Table truncated.

1. **Drop a table.**

SQL> DROP TABLE COURSE;

Table dropped.

1. **Commit the changes made to the database.**

SQL> COMMIT;

Commit complete.

**WEEK - 2 :**

**1.. Create course and student tables by adding following constraints on the attributes:**

a. Primary keys on course\_id and student\_id attributes.

b. Foreign key constraint on course\_id attribute of student table.

c. Unique on course name

d. Not null on student\_name

e. Check constraint on grade attribute of grade table. Grade should be between 5 to 10.

f. Set default value for department attribute of course table.

* CREATE TABLE course(course\_name varchar(20) UNIQUE,

                    course\_idvarchar(10) PRIMARY KEY,

                    Department varchar(10) DEFAULT 'CSE');

       Table created.

* CREATE TABLE student(name varchar(20) NOT NULL,

                     ID   varchar(10) PRIMARY KEY,

                     course varchar(10),

                     semester varchar(10),

                     FOREIGN KEY(course) REFERENCES course(course\_id));

       Table created.

* CREATE TABLE grade(ID varchar(10),

                   Semester varchar(10),

                   course   varchar(10),

                   grade int CHECK(grade >= 5 AND grade <= 10));

Table created.

**2. Write insert statements that violate the constraints specified in problem 1**

* SQL > INSERT INTO course VALUES('programming with python','CS2154','CSE');

    INSERT INTO course VALUES('programming with python',

                           \*

    ERROR at line 1:

    ORA-00001: unique constraint (SUFIYAN.SYS\_C008264) violated

* SQL > INSERT INTO course VALUES   ('DBMS','CS262','CSE');

    INSERT INTO course VALUES   ('DBMS',

                              \*

    ERROR at line 14:

    ORA-00001: unique constraint (ROSHAN.SYS\_C008263) violated

* SQL > INSERT INTO student VALUES (NULL,'Y21CS123','CS125','4th sem');

    INSERT INTO student VALUES (NULL,

                            \*

    ERROR at line 28:

    ORA-01400: cannot insert NULL into ("SUFIYAN"."STUDENT"."NAME")

* SQL > INSERT INTO student VALUES ('Shreyas','Y21CS160','CS222','Sem4');

    INSERT INTO student VALUES ('Shreyas',

                             \*

    ERROR at line 40:

    ORA-00001: unique constraint (SUFIYAN.SYS\_C008266) violated

* SQL > INSERT INTO student VALUES ('Shreyas','Y21CS164','CS232','Sem5');

    INSERT INTO student VALUES ('Shreyas',

      \*

    ERROR at line 52:

    ORA-02291: integrity constraint (SUFIYAN.SYS\_C008267) violated - parent key not

    found

* SQL > INSERT INTO grade VALUES ('Y21CS164','Sem5','CS222',12);

    INSERT INTO grade VALUES ('Y21CS164',

    \*

    ERROR at line 65:

    ORA-02290: check constraint (ROSHAN.SYS\_C008268) violated

**3. Write update statements that violate the constraints specified in problem 1.**

* SQL >UPDATE course SET course\_name='programming with Python' WHERE course\_id='CS215'

\*

ERROR at line 3:

ORA-00001: unique constraint (SUFIYAN.SYS\_C008264) violated

* SQL > UPDATE course SET course\_id='CS215' WHERE course\_name='DBMS';

UPDATE course SET course\_id='CS215' WHERE course\_name='DBMS'

\*

ERROR at line 9:

ORA-00001: unique constraint (SUFIYAN.SYS\_C008263) violated

* SQL > UPDATE student SET name=NULL WHERE id='Y21CS164';

UPDATE student SET name=NULL WHERE id='Y21CS164'

\*

ERROR at line 15:

ORA-01407: cannot update ("SUFIYAN"."STUDENT"."NAME") to NULL

* SQL > UPDATE student SET id='Y21CS164' WHERE name='Sufiyan';

UPDATE student SET id='Y21CS164' WHERE name='Roshan Ali'

\*

ERROR at line 22:

ORA-00001: unique constraint (SUFIYAN.SYS\_C008266) violated

* SQL > UPDATE student SET course='CS222' WHERE id='Y21CS164';

UPDATE student SET course='CS222' WHERE id='Y21CS164'

\*

ERROR at line 37:

ORA-02291: integrity constraint (SUFIYAN.SYS\_C008267) violated - parent key not found

* SQL > UPDATE grade SET grade=2 WHERE id='Y21CS161';

UPDATE grade SET grade=2 WHERE id='Y21CS161'

\*

ERROR at line 43:

ORA-02290: check constraint (SUFIYAN.SYS\_C008268) violated

**4. Write delete statements that violate the constraints specified in problem 1.**

* SQL > DELETE FROM course WHERE course\_id='CS125';

DELETE FROM course WHERE course\_id='CS125'

\*

ERROR at line 5:

ORA-02292: integrity constraint (SUFIYAN.SYS\_C008267) violated - child record

Found

**5. Drop the constraints specified in problem 1 using alter table statements.**

* ALTER TABLE student DROP CONSTRAINT SYS\_C008267;

TABLE altered.

* ALTER TABLE course DROP CONSTRAINT SYS\_C008263;

TABLE altered.

* ALTER TABLE student MODIFY name NULL;

TABLE altered.

* ALTER TABLE student DROP CONSTRAINT SYS\_C008266;

TABLE altered.

* ALTER TABLE grade DROP constraint SYS\_C008268;

Table altered.

**6. Add the constraints specified in problem 1 using alter table statements**

* SQL > ALTER TABLE student ADD CONSTRAINTpk\_studentPRIMARY KEY(id);

Table altered.

* SQL > ALTER TABLE course ADD CONSTRAINTpk\_coursePRIMARY KEY(course\_id);

Table altered.

* SQL > ALTER TABLE student ADD CONSTRAINTfK\_courseFOREIGN KEY(course) REFERENCES course(course\_id);

Table altered.

* SQL > ALTER TABLE course ADD CONSTRAINTu\_courseUNIQUE(course\_name);

Table altered.

* SQL > ALTER TABLE student MODIFY name NOT NULL;

Table altered.

* SQL > ALTER TABLE grade ADD CONSTRAINTck\_gradeCHECK(grade >=5AND grade <=10);

Table altered.

**WEEK -3:**

1. **Create University database by copying it from other user**

* create table prereq

  (course\_id  varchar(8),

   prereq\_id  varchar(8),

 constraint pk\_prereq primary key (course\_id, prereq\_id),

 constraint fk1\_prereq foreign key (course\_id) references course

   on delete cascade,

 constraint fk2\_prereq foreign key (prereq\_id) references course

 );

* create table time\_slot

 (time\_slot\_id  varchar(4),

  day   varchar(1),

  start\_hr  numeric(2) check (start\_hr>= 0 and start\_hr< 24),

  start\_min  numeric(2) check (start\_min>= 0 and start\_min< 60),

  end\_hr   numeric(2) check (end\_hr>= 0 and end\_hr< 24),

  end\_min  numeric(2) check (end\_min>= 0 and end\_min< 60),

constraint pk\_time primary key (time\_slot\_id, day, start\_hr, start\_min)

 );

* create table advisor

 (s\_ID   varchar(5),

  i\_ID   varchar(5),

constraint pk\_advisor primary key (s\_ID),

constraint fk1\_advisor foreign key (i\_ID) references instructor (ID)

  on delete set null,

constraint fk2\_advisor foreign key (s\_ID) references student (ID)

  on delete cascade

 );

* create table takes

  (ID   varchar(5),

   course\_id  varchar(8),

   sec\_id   varchar(8),

   semester  varchar(6),

   year   numeric(4,0),

   grade          varchar(2),

 constraint pk\_takes primary key (ID, course\_id, sec\_id, semester, year),

 constraint fk1\_takes foreign key (course\_id,sec\_id, semester, year) references section

   on delete cascade,

 constraint fk2\_takes foreign key (ID) references student

   on delete cascade

  );

* create table student

 (ID   varchar(5),

  name   varchar(20) not null,

  dept\_name  varchar(20),

  tot\_cred  numeric(3,0) check (tot\_cred>= 0),

constraint pk\_student primary key (ID),

constraint fk\_student foreign key (dept\_name) references department

  on delete set null

 );

* create table teaches

  (ID   varchar(5),

   course\_id  varchar(8),

   sec\_id   varchar(8),

   semester  varchar(6),

   year   numeric(4,0),

 constraint pk\_teaches primary key (ID, course\_id, sec\_id, semester, year),

 constraint fk1\_teaches foreign key (course\_id,sec\_id, semester, year) references section

on delete cascade,

  constraint fk2\_teaches foreign key (ID) references instructor

   on delete cascade

 );

create table section

 (course\_id  varchar(8),

         sec\_id   varchar(8),

  semester  varchar(6)  constraint check\_section check (semester in ('Fall', 'Winter','Spring',

'Summer')),

  year   numeric(4,0) check (year > 1701 and year < 2100),

  building  varchar(15),

  room\_number  varchar(7),

  time\_slot\_id  varchar(4),

constraint pk\_section primary key (course\_id, sec\_id, semester, year),

constraint fk1\_section foreign key (course\_id) references course

  on delete cascade,

constraint fk2\_section foreign key (building, room\_number) references classroom

  on delete set null

 );

 create table instructor

  (ID   varchar(5),

   name   varchar(20) not null,

   dept\_name  varchar(20),

   salary   numeric(8,2) constraint check\_instructor check (salary > 29000),

constraint pk\_instructor primary key(ID),

constraint fk\_instructor foreign key (dept\_name) references department

  on delete set null

 );

create table course

(course\_id  varchar(8),

 title   varchar(50),

 dept\_name  varchar(20),

 credits  numeric(2,0)  constraint check\_course check (credits > 0),

 constraint pk\_course primary key (course\_id),

 constraint fk\_course foreign key (dept\_name) references department

 on delete set null

);

create table department

 (dept\_name  varchar(20),

  building  varchar(15),

  budget          numeric(12,2) constraint check\_dept check (budget > 0),

constraint pk\_dept primary key (dept\_name)

);

create table classroom

 (building  varchar(15),

  room\_number  varchar(7),

  capacity  numeric(4,0),

constraint pk\_classroom primary key (building, room\_number)

 );

1. **Write SQL Select Statements for the following simple queries that retrieve data from a single table:**

* *Find the details of all students*.

SQL >select \* from student;

ID NAME DEPT\_NAME TOT\_CRED

----- -------------------- -------------------- ----------

00128 Zhang Comp. Sci. 102

12345 Shankar Comp. Sci. 32

19991 Brandt History 80

23121 Chavez Finance 110

44553 Peltier Physics 56

45678 Levy Physics 46

54321 Williams Comp. Sci. 54

55739 Sanchez Music 38

70557 Snow Physics 0

76543 Brown Comp. Sci. 58

76653 Aoi Elec. Eng. 60

* *Find the department names of instructors*

SQL>select name,dept\_name from instructor;

NAME DEPT\_NAME

-------------------- --------------------

Srinivasan Comp. Sci.

Wu Finance

Mozart Music

Einstein Physics

El Said History

Gold Physics

Katz Comp. Sci.

Califieri History

Singh Finance

Crick Biology

Brandt Comp. Sci.

Kim Elec. Eng.

* *Find the names of all the instructors from Biology department*

SQL > select name from instructor where dept\_name='Biology';

NAME

--------------------

Crick

* *Find the names of all instructors in the Computer Science department who have salaries greater than $70,000*.

SQL >select name from instructor where dept\_name='Comp. Sci.' and salary>90000;

NAME

--------------------

Brandt

* *Find the names of courses in Computer science department which have 3 credits.*

SQL > select title from course where dept\_name='Comp. Sci.' and credits=3;

TITLE

--------------------------------------------------

Robotics

Image Processing

Database System Concepts

* *Find the names of the instructors, their present salaries and the resulting salaries if they were given a 10% raise.*

SQL >select name,salary as present\_salary,salary+(salary\*0.1) as result\_salary from instructor;

NAME PRESENT\_SALARY RESULT\_SALARY

-------------------- -------------- -------------

Srinivasan 65000 71500

Wu 90000 99000

Mozart 40000 44000

Einstein 95000 104500

El Said 60000 66000

Gold 87000 95700

Katz 75000 82500

Califieri 62000 68200

Singh 80000 88000

Crick 72000 79200

Brandt 92000 101200

Kim 80000 88000

12 rows selected.

* *Find the names of instructors with salary amounts between $90,000 and $100,000.*

SQL >select name from instructor where dept\_name='Comp. Sci.' and salary between 90000 and 100000;

NAME

--------------------

Brandt

* *Find all instructors whose salary is unknown.*

SQL > select name from instructor where salary is null;

no rows selected

* *Find the names of all departments whose building name includes the substring ‘Watson’*.

SQL >select dept\_name from department where building like '%Watson%';

DEPT\_NAME

--------------------

Biology

Physics

* *Find departments whose names contain the string “sci” as a substring, regardless of the case.*

SQL >selectdept\_name from department where dept\_name like '%Sci%';

DEPT\_NAME

--------------------

Comp. Sci.

* *List the names of all instructors in the Physics department in alphabetic order.*

SQL > select name from instructor where dept\_name='Physics' order by name ASC;

NAME

--------------------

Einstein

Gold

* *List the entire instructor relation in descending order of salary. If several instructors have the same salary, order them in ascending order by name.*

SQL >select \* from instructor order by salary desc,nameasc;

ID NAME DEPT\_NAME SALARY

----- -------------------- -------------------- ----------

22222 Einstein Physics 95000

83821 Brandt Comp. Sci. 92000

12121 Wu Finance 90000

33456 Gold Physics 87000

98345 Kim Elec. Eng. 80000

76543 Singh Finance 80000

45565 Katz Comp. Sci. 75000

76766 Crick Biology 72000

10101 Srinivasan Comp. Sci. 65000

58583 Califieri History 62000

32343 El Said History 60000

15151Mozart Music 40000

1. **Write SQL queries for retrieving data from multiple tables using Joins:**

* *Find all possible combinations of instructors and the courses they teach*.

SQL >SELECTi.id,i.name,t.course\_id,c.title FROM instructor i,coursec,teaches t where t.id=i.id and

c.course\_id=t.course\_id;

ID NAME COURSE\_I TITLE

----- -------------------- -------- --------------------------------------------------

10101 Srinivasan CS-101 Intro. to Computer Science

10101 Srinivasan CS-315 Robotics

10101 Srinivasan CS-347 Database System Concepts

12121 Wu FIN-201 Investment Banking

15151 Mozart MU-199 Music Video Production

22222 Einstein PHY-101 Physical Principles

32343 El Said HIS-351 World History

45565 Katz CS-101 Intro. to Computer Science

45565 Katz CS-319 Image Processing

76766 Crick BIO-101 Intro. to Biology

76766 Crick BIO-301 Genetics

83821 Brandt CS-190 Game Design

83821 Brandt CS-190 Game Design

83821 Brandt CS-319 Image Processing

98345 Kim EE-181 Intro. to Digital Systems

15 rows selected.

* *Retrieve the names of all instructors, along with their department names and department building name.*

SQL >SELECT i.name,i.dept\_name,d.building FROM instructor i,department d where i.dept\_name=d.dept\_name;

NAME DEPT\_NAME BUILDING

-------------------- -------------------- ---------------

Srinivasan Comp. Sci. Taylor

Wu Finance Painter

Mozart Music Packard

Einstein Physics Watson

El Said History Painter

Gold Physics Watson

Katz Comp. Sci. Taylor

Califieri History Painter

Singh Finance Painter

Crick Biology Watson

Brandt Comp. Sci. Taylor

Kim Elec. Eng. Taylor

12 rows selected.

* *Find the names of instructors who have taught at least one course.*

SQL >SELECT distinct(i.name) FROM instructor i,teaches t WHERE i.id=t.id;

NAME

--------------------

Srinivasan

Wu

Mozart

Einstein

El Said

Katz

Crick

Brandt

Kim

9 rows selected.

* *For the student with ID 12345 (or any other value), show all course\_id and title of all courses registered for by the student.*

SQL >SELECTc.course\_id,c.title FROM course c,takes t WHERE t.id='12345' and t.course\_id=c.course\_id;

COURSE\_I TITLE

-------- --------------------------------------------------

CS-101 Intro. to Computer Science

CS-190 Game Design

CS-315 Robotics

CS-347 Database System Concepts

* *Find instructor names and course identifiers for instructors in the Computer Science department.*

SQL >SELECT i.name,c.course\_id FROM instructor i,coursec,teaches t WHERE i.dept\_name='Comp. Sci.' and t.id=i.id and t.course\_id=c.course\_id;

NAME COURSE\_I

-------------------- --------

Srinivasan CS-101

Srinivasan CS-315

Srinivasan CS-347

Katz CS-101

Katz CS-319

Brandt CS-190

Brandt CS-190

Brandt CS-319

8 rows selected.

* *For all instructors in the university who have taught some course, find their names and the course ID of all courses they taught*.

SQL >SELECT i.name,t.course\_id FROM instructor i,teaches t WHERE i.ID=t.ID;

NAME COURSE\_I

-------------------- --------

Srinivasan CS-101

Srinivasan CS-315

Srinivasan CS-347

Wu FIN-201

Mozart MU-199

Einstein PHY-101

El Said HIS-351

Katz CS-101

Katz CS-319

Crick BIO-101

Crick BIO-301

Brandt CS-190

Brandt CS-190

Brandt CS-319

Kim EE-181

15 rows selected.

* Find the names of all instructors whose salary is greater than at least one instructor in the Biology department. Or Find the names of all instructors who earn more than the lowest paid instructor in the Biology department

SQL > SELECT name FROM instructor WHERE salary>(SELECT MIN(salary) FROM instructor WHERE dept\_name='Biology');

NAME

--------------------

Wu

Einstein

Gold

Katz

Singh

Brandt

Kim

7 rows selected.

* Find full details of instructors who teach at least one course.

SQL >SELECT distinct(i.ID),i.name,i.dept\_name,i.salary FROM instructor i,teaches t WHERE i.ID=t.ID;

ID NAME DEPT\_NAME SALARY

----- ---------------------------------------------- -- ----------

10101 Srinivasan Comp. Sci. 65000

12121 Wu Finance 90000

15151 Mozart Music 40000

22222 Einstein Physics 95000

32343 El Said History 60000

45565 Katz Comp. Sci. 75000

76766 Crick Biology 72000

83821 Brandt Comp. Sci. 92000

98345 Kim Elec. Eng. 80000

9 rows selected.

* Find the instructor names and the courses they taught for all instructors in the Biology department who have taught some course.

SQL > SELECT i.name,t.course\_id,c.title FROM instructor i,teachest,course c WHERE i.dept\_name='Biology’ and t.ID=i.ID and t.course\_id=c.course\_id;

NAME COURSE\_I TITLE

-------------------- -------- --------------------------------------------------

Crick BIO-101 Intro. to Biology

Crick BIO-301 Genetics

* Find the set of all courses taught either in Fall 2009 or in Spring 2010, or both.

SQL >( SELECTc.course\_id,c.title FROM course c,teaches t WHERE t.semester='Fall' and t.year=2009 and c.course\_id=t.course\_id) UNION ( SELECT c.course\_id,c.title FROM course c,teaches t WHERE t.semester='Spring' and t.year=2010 and c.course\_id=t.course\_id);

COURSE\_I TITLE

-------- --------------------------------------------------

CS-101 Intro. to Computer Science

CS-315 Robotics

CS-319 Image Processing

CS-347 Database System Concepts

FIN-201 Investment Banking

HIS-351 World History

MU-199 Music Video Production

PHY-101 Physical Principles

8 rows selected.

* Find all courses taught in the Fall 2009 semester but not in the Spring 2010 semester.

SQL > ( SELECT c.course\_id,c.title FROM course c,teaches t WHERE t.semester='Fall' and t.year=2009 and c.course\_id=t.course\_id) MINUS ( SELECT c.course\_id,c.title FROM course c,teaches t WHERE t.semester='Spring' and t.year=2010 and c.course\_id=t.course\_id);

COURSE\_I TITLE

-------- --------------------------------------------------

CS-347 Database System Concepts

PHY-101 Physical Principles

* Find the names of all students who have taken any Comp. Sci. course ever. (there should be no duplicate names)

SQL >SELECT distinct(s.name) FROM student s,takes t WHERE s.ID=t.ID and t.course\_id like '%CS%';

NAME

--------------------

Bourikas

Brown

Levy

Shankar

Williams

Zhang

6 rows selected.

* Display the IDs of all instructors who have never taught a course. (Don’t write nested query)

SQL >SELECT ID FROM instructor MINUS SELECT distinct(ID) FROM teaches;

ID

-----

33456

58583

76543

**WEEK 4:**

1. Write SQL Select Statements using Aggregate Functions, Group By and Having clauses for the following queries that retrieve data from university database:

* Find the maximum and average capacity of buildings in the university.

SQL> SELECT building,sum(capacity) MAX\_CAPACITY,avg(capacity) AVG\_CAPACITY FROM classroom GROUP BY building**;**

BUILDING MAX\_CAPACITY AVG\_CAPACITY

--------------- ------------ ------------

Packard 500 500

Painter 10 10

Taylor 70 70

Watson 80 40

* Display the least budget of the departments.

SQL> SELECT min(budget) LEAST\_BUDGET FROM department;

LEAST\_BUDGET

------------

50000

* Find the total number of courses and credits offered by Biology department.

SQL> SELECT COUNT(title) COURSES,SUM(credits) CREDITS FROM course GROUP BY

dept\_name HAVING dept\_name='Biology';

COURSES CREDITS

------------- ----------

3 11

* Find the average salary of instructors in the Computer Science department

SQL> SELECT AVG(salary) FROM instructor GROUP BY dept\_name HAVING dept\_name='Comp. Sci.';

AVG(SALARY)

-----------

77333.3333

* Find the total number of instructors who teach a course in the Spring 2010 semester.

SQL> SELECT count(distinct(ID)) INSTRUCTORS FROM teaches GROUP BY semester,year HAVING semester='Spring' and year=2010;

INSTRUCTORS

------------------

6

* Find the average salary in each department.

SQL> SELECT avg(salary),dept\_name FROM instructor GROUP BY dept\_name;

AVG(SALARY) DEPT\_NAME

------------------ --------------------

72000 Biology

77333.3333 Comp. Sci.

80000 Elec. Eng.

85000 Finance

61000 History

40000 Music

91000 Physics

7 rows selected.

* Find the number of instructors in each department who teach a course in the Spring 2010 semester.

SQL > SELECTi.dept\_name,COUNT(i.ID) instructor\_count FROM teaches t,instructori WHERE i.ID=t.ID GROUP BY t.semester,t.year,i.dept\_name HAVING semester='Spring' and year=2010;

DEPT\_NAME INSTRUCTOR\_COUNT

----------------------- ------------------------------

Comp. Sci. 4

Finance 1

History 1

Music 1

* Find the department name and average salary of the department for only those departments where the average salary of the instructors is more than $42,000.

SQL> SELECT i.dept\_name,AVG(i.salary) FROM instructor i GROUP BY i.dept\_name HAVING AVG(i.salary)>42000;

DEPT\_NAME AVG(I.SALARY)

-------------------- -------------

Biology 72000

Comp. Sci. 77333.3333

Elec. Eng. 80000

Finance 85000

History 61000

Physics 91000

6 rows selected.

* For each course section offered in 2009, find the average total credits (tot\_cred) of all students enrolled in the section, if the section had at least 2 students.

SQL> SELECT s.course\_id,s.sec\_id,AVG(st.tot\_cred) stu\_tot\_cred FROM student st,section s,takes t WHERE s.course\_id=t.course\_id AND s.sec\_id=t.sec\_id AND s.year=t.year AND t.id=st.id AND s.year=2009 GROUP BY s.course\_id,s.sec\_id,s.year HAVING COUNT(t.id) >=2;

COURSE\_I SEC\_ID STU\_TOT\_CRED

-------- -------- ------------

CS-101 1 65

CS-190 2 43

CS-347 1 67

* For each department, find the maximum salary of instructors in that department. You may assume that every department has at least one instructor.

SQL> SELECT dept\_name,MAX(salary) FROM instructor GROUP BY dept\_name;

DEPT\_NAME MAX(SALARY)

-------------------- -----------

Biology 72000

Comp. Sci. 92000

Elec. Eng. 80000

Finance 90000

History 62000

Music 40000

Physics 95000

7 rows selected.

* For the student with ID 12345 (or any other value), show the total number of credits scored for all courses (taken by that student). Don't display the tot\_creds value from the student table, you should use SQL aggregation on courses taken by the student

SQL> SELECT t.ID,SUM(c.credits) FROM takes t,course c WHERE t.course\_id=c.course\_id and t.ID=12345 GROUP BY t.ID;

ID SUM(C.CREDITS)

----- --------------

12345 14

* Display the total credits for each of the students, along with the ID of the student; don't bother about the name of the student. (Don't display the tot\_creds value from the student table, you should use SQL aggregation on courses taken by the student. For students who have not registered for any course, tot\_creds should be 0)

SQL> SELECT t.ID,SUM(c.credits) FROM takes t,course c WHERE t.course\_id=c.course\_id GROUP BY t.ID;

ID SUM(C.CREDITS)

----- --------------

00128 7

12345 14

19991 3

23121 3

44553 4

45678 11

54321 8

55739 3

76543 7

76653 3

98765 7

98988 8

12 rows selected.

1. Write nested queries for answering the following queries that retrieve data from university database:

* Find the total number of (distinct) students who have taken course sections taught by the instructor with ID 110011.

SQL> SELECT COUNT(distinct(s.ID)) STUDENTS\_COUNT FROM teaches t,takes s WHERE s.course\_id=t.course\_id and t.ID=10101;

STUDENTS\_COUNT

--------------

6

* Find the names of all instructors whose salary is greater than at least one instructor in the Biology department

SQL> SELECT i.name FROM instructor i WHERE i.salary>(SELECT MIN(salary) FROM instructor WHERE dept\_name='Biology');

NAME

--------------------

Wu

Einstein

Gold

Katz

Singh

Brandt

Kim

7 rows selected.

* Find the departments that have the highest average salary.

SQL> SELECT dept\_name FROM instructor GROUP BY dept\_name HAVING AVG(salary)=(SELECT MAX(AVG(salary))FROM instructor GROUP BY dept\_name);

DEPT\_NAME

--------------------

Physics

* Find all the courses taught in the both the Fall 2009 and Spring 2010 semesters

SQL> SELECT c.title FROM course c,teaches t WHERE t.course\_id=c.course\_id and semester='Fall' and c.course\_id in (SELECT c.course\_id FROM course c,teaches t WHERE t.course\_id=c.course\_id and (semester='Spring'));

TITLE

--------------------------------------------------

Intro. to Computer Science

* Find all the courses taught in the Fall 2009 semester but not in the Spring 2010 semester.

SQL> SELECT c.title FROM course c,teaches t WHERE t.course\_id=c.course\_id and t.semester='Fall' and t.year=2009 and c.course\_id not in (SELECT c.course\_id FROM course c,teaches t WHERE t.course\_id=c.course\_id and t.semester ='Spring' and t.year=2010);

TITLE

--------------------------------------------------

Database System Concepts

Physical Principles

* Find all courses taught in both the Fall 2009 semester and in the Spring 2010 semester. (Write correlated nested Query)

SQL> SELECT c.title FROM course c,teaches t WHERE t.course\_id=c.course\_id and t.semester='Fall' and t.year=2009 and EXISTS(SELECT \* FROM teaches s WHERE t.course\_id=s.course\_id and s.semester ='Spring' and s.year=2010);

TITLE

--------------------------------------------------

Intro. to Computer Science

* Find all students who have taken all courses offered in the Biology department. (Write Correlated nested Query)

SQL>SELECT s.ID FROM STUDENT s WHERE NOT EXISTS((SELECT c.course\_id FROM course c WHERE c.dept\_name='Biology') MINUS ( SELECT t.course\_id FROM takes t WHERE t.ID=s.ID));

No rows selected

* Find all courses that were offered at most once in 2009.

SQL >SELECT DISTINCT(course\_id) FROM teaches WHERE course\_id in

(SELECT course\_id FROM teaches WHERE year=2009 GROUP BY course\_id HAVING COUNT(course\_id)<=1);

COURSE\_ID

------------------

CS-101

CS-347

PHY-101

BIO-101

EE-181

* Find all courses that were offered at least twice in 2009”

SQL >SELECT DISTINCT(course\_id) FROM teaches WHERE course\_id in

(SELECT course\_id FROM teaches WHERE year=2009 GROUP BY course\_id HAVING COUNT(course\_id)>1);

COURSE\_ID

------------------

CS-190

* Find the average instructors’ salaries of those departments where the average salary is greater than $42,000.

SQL> SELECT i.dept\_name,AVG(i.salary) FROM instructor i GROUP BY i.dept\_name HAVING AVG(i.salary)>42000;

DEPT\_NAME AVG(I.SALARY)

-------------------- -------------

Biology 72000

Comp. Sci. 77333.3333

Elec. Eng. 80000

Finance 85000

History 61000

Physics 91000

6 rows selected.

* Find the departments with the maximum budget.

SQL >SELECT dept\_name FROM department WHERE budget=(SELECT MAX(budget) FROM department);

DEPT\_NAME

------------------

Finance

* Find the names of instructors who have not taught any course.

SQL >SELECT i.ID FROM instructor i WHERE NOT EXISTS(SELECT \* FROM teaches t WHERE t.ID=i.ID );

ID

-----

33456

58583

76543

* Find the IDs and names of all students who have not taken any course offering before Spring 2009.

SQL> (SELECT ID, name FROM student)MINUS(SELECT s.id,s.name FROM student s,teaches t WHERE s.id=t.id and t.year<2009);

ID NAME

----- --------------------

00128 Zhang

12345 Shankar

19991 Brandt

23121 Chavez

44553 Peltier

45678 Levy

54321 Williams

55739 Sanchez

70557 Snow

76543 Brown

76653 Aoi

98765 Bourikas

98988 Tanaka

13 rows selected.

* Find the lowest, across all departments, of the per-department maximum salary computed.

SQL >SELECT MAX(salary) LEAST\_MAX\_SALARY FROM instructor WHERE salary=(SELECT MIN(AVG(salary)) FROM instructor GROUP BY dept\_name);

LEAST\_MAX\_SALARY

----------------------------

40000

* Display the IDs and names of the instructors who have taught all Comp. Sci. courses

SQL> SELECT i.id , i.name FROM instructor i WHERE NOT EXISTS((SELECT c.course\_id FROM course c WHERE c.dept\_name='Comp. Sci.')MINUS(SELECT t.course\_id FROM teaches t WHERE i.id=t.id));

no rows selected