CSE 5261/CSE 5162 Computing Lab II /Information Systems Lab II [ADBS LAB Manual]

II SEM M.Tech [CSE/CSIS] 2019-2020

Course Objectives

- To understand and use different constructs in SQL.
- Learn to create views and authorization.
- To develop data models for different database application.

Course Outcomes

At the end of this course, students will have the

- Ability to write complex SQL queries.
- Ability to implement various real world database applications.

Evaluation plan

- Internal assessment: 60 marks
  - ✓ The assessment will depend on punctuality, program execution, maintaining the observation note and answering the questions in viva voce.
  - ✓ Total marks for the 6 experiments is 30 Marks. (Total 3 Biweekly Evaluations. Each evaluation has 10 Marks (5 Marks for execution + 2 Marks for Observation + 3 Marks for Viva))
  - ✓ Mini project is for 30 Marks (10 marks for Report, 15 Marks for Implementation and 5 marks for viva)

- End semester assessment: 40 marks (Lab no. 1 to Lab No. 6)
- Total (Internal assessment (60 Marks) + End semester assessment (40 Marks): 100 marks

INSTRUCTIONS TO THE STUDENTS

Pre-Lab Session Instructions

- Students should carry the Lab Manual Book and the required stationery to every lab session.
- Be in time and follow the institution dress code.
- Must Sign in the log register provided.
- Make sure to occupy the allotted seat and answer the attendance.
- Adhere to the rules and maintain the decorum.

In-Lab Session Instructions

- Follow the instructions on the allotted exercises.
- Show the program and results to the Lab Instructor on completion of experiments.
- You must have your lab notebook signed by your Lab Instructor before you leave lab each day. Any pages not signed on the day the experiment was performed will adversely affect your lab notebook grade.
- Prescribed textbooks and class notes can be kept ready for reference if required.

General Instructions for the exercises in Lab

- Implement the given exercise individually and not in a group.
- Plagiarism (copying from others) is strictly prohibited and would invite severe penalty in evaluation.
- In case a student misses a lab session, he/she may ensure that the experiment is completed during the extra lab session (of other batch) with the permission of the HOD.
- Questions for lab tests and examination are not necessarily limited to the questions in the manual, but may involve some variations and/or combinations of the questions.

THE STUDENTS SHOULD NOT

- Bring mobile phones or any other electronic gadgets to the lab.
- Go out of the lab without permission.

Lab 1: REVIEW OF SQL

STRUTURED QUERY LANGUAGE (SQL):

The SQL is a computer language for storing, manipulating and retrieving data stored in relational database.

SQL Statements can be categorized as -

DDL (Data definition language): Are used to define database structure or schema.
DML (Data manipulation language): Are used to change or alter data with the database or schema.
DCL (Data Control Language): Are used to control access or privileges.
TCL (Transaction Control Language): Are used to manage transactions in the database.

DATA TYPES IN SQL:

Numeric: NUMBER, NUMBER(s, p), INTEGER, INT, FLOAT, DECIMAL
Character: CHAR(n), VARCHAR(n), VARCHAR2(n), CHAR VARYING(n)
Bit String: BLOB, CLOB
Boolean: true, false, and null
Date and Time: DATE (YYYY-MM-DD) TIME(HH:MM:SS)
Timestamp: DATE + TIME

DDL COMMANDS IN SQL:

The commands used are:

● CREATE - It is used to create a new relation/table by giving it a name and specifying its attributes and initial constraints. The attributes are specified first, and each attribute is given a name, a data type to specify its domain of values, and possibly attribute constraints, such as NOT NULL. The key, entity integrity, referential integrity constraints can be specified within the CREATE TABLE statement after attributes are declared, or they can be added later using the ALTER TABLE command.

● ALTER – The structure of a table can be modified by using the ALTER TABLE command. This command is used to add a new column, modify the existing column definition and to include or drop integrity constraint.

● DROP - It will delete the table structure provided the table should be empty.

● TRUNCATE - If there is no further use of records stored in a table and the structure has to be retained, and then the records alone can be deleted.

● DESC - This is used to view the structure of the table.

CREATION OF TABLE:

SYNTAX: CREATE TABLE <table_name>(column1 datatype, column2 datatype...);
EXAMPLE:

SQL>C**REATE TABLE** EMPLOYEE
(Fname VARCHAR(15),
Minit CHAR,
Lname VARCHAR(15),
Ssn CHAR(9),
Bdate DATE,
Address VARCHAR(30),
Sex CHAR,
Salary DECIMAL(10,2),
Super_ssn CHAR(9),
Dno INT);

ALTER TABLE

(a) To Add column to existing Table
Syntax: **ALTER TABLE** table-name ADD(**column-name** datatype);
EXAMPLE:
ALTER TABLE Employee ADD (phone_no char (20));

(b)To Add Multiple columns to existing Table
Syntax: ALTER TABLE table-name ADD(**column-name1** datatype1, **column-name2** datatype2, **column-name3** datatype3);
EXAMPLE:
ALTER TABLE Employee ADD(salary number(7), age(5));

(c) Dropping a Column from a Table
Syntax: ALTER TABLE <Table Name>DROP COLUMN <CoumnName>;
EXAMPLE:
ALTER TABLE Employee DROP COLUMN phone_no ;
(d) Modifying Existing Columns
Syntax: ALTER TABLE <Table Name>MODIFY (<CoumnName><Newdata type>(<size>));
EXAMPLE:
ALTER TABLE Employee MODIFY (EName VarChar(25))

(e) **To Rename a column**
Using alter command we can rename an existing column
Syntax: ALTER TABLE *table_name* **rename** old-column-name to column-name;

EXAMPLE:
ALTER TABLE Employee RENAME address to Location;

RENAMING TABLES
Syntax: RENAME <oldtable> to <new table>;
EXAMPLE:
RENAME Employee to Employee 1;

TRUNCATE TABLE

The SQL **TRUNCATE TABLE** command is used to delete complete data from an existing table
Syntax: TRUNCATE TABLE <TABLE NAME>;
Example:
TRUNCATE TABLE Employee;

DESTROYING TABLES

We can also use DROP TABLE command to delete complete table but it would remove complete table structure form the database and you would need to re-create this table once again if you wish you store some data.

Syntax: DROP TABLE <TABLE NAME>;
Example:
DROP TABLE Employee;

DESCRIBE TABLES
Syntax: DESC <TABLE NAME>;
Example: DESC employee

DML: Data Manipulation Language (DML) statements are used for managing data within schema objects and to manipulate data of a database objects.
DML Commands: Select ,Insert, Update, Delete

SELECT - retrieve data from the a database
INSERT - insert data into a table
UPDATE - updates existing data within a table
DELETE - deletes all records from a table, the space for the records remain

Example 1.1:

Selecting the information from table(s)
Syntax: Select col1, col2, col3,……, coln from <table_name> where < condition >

Ex:

a) List all the students
   Select * from student;
b) List age of all students with column aliased as 'student_age' rather stu_age
   Select stu_age student_age from student;
c) Find the sum of all three subject marks and name it as tot_marks.

Select subject1_marks + subject2_marks + subject3_marks tot_marks from student.

Example 1.2 Inserting Data into Tables:

Syntax: Insert into <tablename> (<col1>,<col2>) values (<exp>,<exp>);

Ex: insert into STUDENT(reg_no, stu_name) values (102, 'KRISH');

Example 1.3 Updating the contents of a table

a) Updating all rows

Syntax: Update <tablename> set <col>=<exp>, <col>=<exp>;

Ex: Update STUDENT set stu_name='MANAV';

b) Updating selected records

Syntax: Update <tablename> set <col>=<exp>,<col>=<exp>where <condition>;

Ex: Update STUDENT set stu_name='YADAV' where reg_no=101;

1.4 Delete operations

a) Removal of specified row/s

Syntax: Delete from <tablename> where <condition>;

Ex: Delete from STUDENT where reg_no=102;

b) Remove all rows

Syntax: Delete from <tablename>;

Ex: Delete from STUDENT;

LAB EXERCISES:

1. Consider the following COMPANY relational database schema. Create EMPLOYEE, DEPARTMENT, DEPT_LOCATIONS, PROJECT, WORKS_ON, and DEPENDENT tables with the following structure.

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

2. Display names of all employees.
3. Display all the employees from ‘HOUSTON’.
4. Add a column named Phone_no to employee table.

5. Assign the Phone_no for all employees.
6. View the structure of the table employee using describe.
7. Delete all the employees from ‘HUMBLE’
8. Rename employee as employee1.
9. Drop the table employee1.

LAB NO. 2: INTEGRITY CONSTRAINTS IN SQL

SQL supports a number of different integrity constraints.

- **primary key** (Aj1, Aj2, . . . , Ajm): The **primary-key** specification says that attributes Aj1, Aj2, . . . , Ajm form the primary key for the relation. The **primary key** attributes are required to be **nonnull** and **unique**; that is, no tuple can have a null value for a primary-key attribute, and no two tuples in the relation can be equal on all the primary-key attributes.

- **foreign key** (Ak1, Ak2, . . . , Akn) references s: The **foreign key** specification says that the values of attributes (Ak1, Ak2, . . . , Akn) for any tuple in the relation must correspond to values of the primary key attributes of some tuple in relation s.

- **not null**: The **not null** constraint on an attribute specifies that the null value is not allowed for that attribute.

The referential integrity constraint is specified between two relations and is used to maintain the consistency among tuples in the two relations.

Constraints can be specified when a table is created with the CREATE TABLE statement or you can use ALTER TABLE statement to create constraints even after the table is created.

Example 2.1:

```
CREATE TABLE DEPARTMENT
(
      dname VARCHAR2(20)  NOT NULL ,
      dnumber INTEGER  NOT NULL ,
      mgrssn NUMBER(9)  NULL
);

ALTER TABLE DEPARTMENT
      ADD PRIMARY KEY (dnumber);

CREATE TABLE DEPARTMENT_LOCATIONS
(
      dnumber  INTEGER  NOT NULL ,
      dlocation  VARCHAR2(20)  NOT NULL
);


ALTER TABLE DEPARTMENT_LOCATIONS
      ADD  PRIMARY KEY (dnumber,dlocation);


ALTER TABLE DEPARTMENT_LOCATIONS
      ADD ( FOREIGN KEY (dnumber) REFERENCES DEPARTMENT(dnumber));
```

Example 2.2:

The following data definition statements with integrity constraints in SQL for COMPANY relational database.

```
CREATE TABLE EMPLOYEE
        ( Fname                 VARCHAR(15)             NOT NULL,
          Minit                 CHAR,
          Lname                 VARCHAR(15)             NOT NULL,
          Ssn                   CHAR(9)                 NOT NULL,
          Bdate                 DATE,
          Address               VARCHAR(30),
          Sex                   CHAR,
          Salary                DECIMAL(10,2),
          Super_ssn             CHAR(9),
          Dno                   INT                     NOT NULL,
        PRIMARY KEY (Ssn),
CREATE TABLE DEPARTMENT
        ( Dname                 VARCHAR(15)             NOT NULL,
          Dnumber               INT                     NOT NULL,
          Mgr_ssn               CHAR(9)                 NOT NULL,
          Mgr_start_date        DATE,
        PRIMARY KEY (Dnumber),
        UNIQUE (Dname),
        FOREIGN KEY (Mgr_ssn) REFERENCES EMPLOYEE(Ssn) );
CREATE TABLE DEPT_LOCATIONS
        ( Dnumber               INT                     NOT NULL,
          Dlocation             VARCHAR(15)             NOT NULL,
        PRIMARY KEY (Dnumber, Dlocation),
        FOREIGN KEY (Dnumber) REFERENCES DEPARTMENT(Dnumber) );
CREATE TABLE PROJECT
        ( Pname                 VARCHAR(15)             NOT NULL,
          Pnumber               INT                     NOT NULL,
          Plocation             VARCHAR(15),
          Dnum                  INT                     NOT NULL,
        PRIMARY KEY (Pnumber),
        UNIQUE (Pname),
        FOREIGN KEY (Dnum) REFERENCES DEPARTMENT(Dnumber) );
CREATE TABLE WORKS_ON
        ( Essn                  CHAR(9)                 NOT NULL,
          Pno                   INT                     NOT NULL,
          Hours                 DECIMAL(3,1)            NOT NULL,
        PRIMARY KEY (Essn, Pno),
        FOREIGN KEY (Essn) REFERENCES EMPLOYEE(Ssn),
        FOREIGN KEY (Pno) REFERENCES PROJECT(Pnumber) );
CREATE TABLE DEPENDENT
        ( Essn                  CHAR(9)                 NOT NULL,
          Dependent_name        VARCHAR(15)             NOT NULL,
          Sex                   CHAR,
          Bdate                 DATE,
          Relationship          VARCHAR(8),
        PRIMARY KEY (Essn, Dependent_name),
        FOREIGN KEY (Essn) REFERENCES EMPLOYEE(Ssn) );
```

Lab Exercises

1. Consider the following UNIVERSITY database schema. Create tables DEPARTMENT, INSTRUCTOR, COURSE, SECTION, and TEACHES with the following structure.

DEPARTMENT

dept_name	*building*	*budget*
Biology	Watson	90000
Comp. Sci.	Taylor	100000
Elec. Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000

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-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

INSTRUCTOR

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
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

SECTION

course_id	*sec_id*	*semester*	*year*	*building*	*room_number*	*time_slot_id*
BIO-101	1	Summer	2009	Painter	514	B
BIO-301	1	Summer	2010	Painter	514	A
CS-101	1	Fall	2009	Packard	101	H
CS-101	1	Spring	2010	Packard	101	F
CS-190	1	Spring	2009	Taylor	3128	E
CS-190	2	Spring	2009	Taylor	3128	A
CS-315	1	Spring	2010	Watson	120	D
CS-319	1	Spring	2010	Watson	100	B
CS-319	2	Spring	2010	Taylor	3128	C
CS-347	1	Fall	2009	Taylor	3128	A
EE-181	1	Spring	2009	Taylor	3128	C
FIN-201	1	Spring	2010	Packard	101	B
HIS-351	1	Spring	2010	Painter	514	C
MU-199	1	Spring	2010	Packard	101	D
PHY-101	1	Fall	2009	Watson	100	A

TEACHES

ID	*course_id*	*sec_id*	*semester*	*year*
10101	CS-101	1	Fall	2009
10101	CS-315	1	Spring	2010
10101	CS-347	1	Fall	2009
12121	FIN-201	1	Spring	2010
15151	MU-199	1	Spring	2010
22222	PHY-101	1	Fall	2009
32343	HIS-351	1	Spring	2010
45565	CS-101	1	Spring	2010
45565	CS-319	1	Spring	2010
76766	BIO-101	1	Summer	2009
76766	BIO-301	1	Summer	2010
83821	CS-190	1	Spring	2009
83821	CS-190	2	Spring	2009
83821	CS-319	2	Spring	2010
98345	EE-181	1	Spring	2009

2. In DEPARTMENT table make dept_name as primary key.

3. In INSTRUCTOR table make ID as primary key.

4. In COURSE table make course_id as primary key.

5. In SECTION table make course_id, sec_id, semester, year as primary key.

6. In TEACHES table make ID, course_id, sec_id, semester, year as primary key.

7. Do not allow primary key to have null values in all the tables. .

8. Do not allow name in INSTRUCTOR table to have null values.

9. Make dept_name of COURSE as foreign key which refers to dept_name of Department.

10. Make dept_name of INSTRUCTOR as foreign key which refers to dept_name of Department.

11. Make course_id of SECTION as foreign key which refers to course_id of COURSE.

12. Make ID of TEACHES as foreign key which refers to ID of INSTRUCTOR.

13. Make course_id, sec_id, semester, year of TEACHES as foreign key which refers to course_id, sec_id, semester, year of SECTION.

14. Insert few tuples into INSTRUCTOR and DEPARTMENT which satisfies the above constraints.

15.Try to insert few tuples into INSTRUCTOR and DEPARTMENT which violates some of the above constraints.

LAB NO. 3: BASIC QUERIES IN SQL

We will start with simple queries, and then progress to more complex ones in a step-by-step manner. The basic form of the SELECT statement, sometimes called a **mapping** or a **select-from-where block**, is formed of the three clauses SELECT, FROM, and WHERE and has the following form:

SELECT <attribute list>
FROM <table list>
WHERE <condition>;

where
- <attribute list> is a list of attribute names whose values are to be retrieved by the query.
- <table list> is a list of the relation names required to process the query.
- <condition> is a conditional (Boolean) expression that identifies the tuples to be retrieved by the query.

In SQL, the basic logical comparison operators for comparing attribute values with one another and with literal constants are =, <, <=, >, >=, and <>. These correspond to the relational algebra operators =, <, ≤, >, ≥, and ≠, respectively.

Examples

Example 3.1: Retrieve the birth date and address of the employee(s) whose name is 'John B. Smith'.

```
SELECT Bdate, Address
FROM EMPLOYEE
WHERE Fname = 'John' AND Minit = 'B' AND Lname = 'Smith';
```

Example 3.2: Retrieve the name and address of all employees who work for the 'Research' department.

```
SELECT Fname, Lname, Address
FROM EMPLOYEE, DEPARTMENT
WHERE Dname = 'Research' AND Dnumber = Dno;
```

We use the keyword DISTINCT in the SELECT clause, meaning that only distinct tuples should remain in the result.

Example 3.3: Retrieve all distinct salary values of every employee.

```
SELECT DISTINCT Salary
FROM EMPLOYEE
```

Substring Pattern Matching: LIKE Operator

The LIKE is used to compare a value to similar values using wildcard operators.

SQL supports two Wildcard operator

(a) The parentage sign (%): % replaces an arbitrary number of zero or more characters.

(b) The underscore (_): _ replaces a single character.

Example 3.4: Retrieve all employees whose address is in Houston, Texas.

```
SELECT Fname, Lname
FROM EMPLOYEE
WHERE Address LIKE '%Houston,TX%';
```

To retrieve all employees who were born during the 1970s, we can use Query 5. Here, '7' must be the third character of the string (according to our format for date), so we use the value '_ _ 7 _ _ _ _ _ _ _ _', with each underscore serving as a placeholder for an arbitrary character.

Example 3.5: Find all employees who were born during the 1950s.

```
SELECT Fname, Lname
FROM EMPLOYEE
WHERE Bdate LIKE '_ _ 7 _ _ _ _ _ _ _ _';
```

Set Operations

The SQL operations **union**, **intersect**, and **except (minus)** operate on relations and correspond to the mathematical set-theory operations ∪, ∩, and -.

Example 3.6: (Use University database schema)

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

```
(select course_id
from section
where semester = 'Fall' and year= 2009)
union
(select course_id
from section
where semester = 'Spring' and year= 2010);
```

The union operation automatically eliminates duplicates, unlike the select clause. If we want to retain all duplicates, we must write **union all** in place of **union**.

Lab Exercises (Use company database schema for the following exercise problems**)**

1. Select all EMPLOYEE Ssns.

2. All combinations of EMPLOYEE Ssn and DEPARTMENT Dname in the database.

3. Retrieve all the attribute values of any EMPLOYEE who works in DEPARTMENT number 5.

4. Retrieve all the attributes of an EMPLOYEE and the attributes of the DEPARTMENT in which he or she works for every employee of the 'Research' department.

5. For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birth date.

6. For each employee, retrieve the employee's first and last name and the first and last name of his or her immediate supervisor.

7. Retrieve the names of all employees in department 5 who work more than 10 hours per week on the ProductX project.
8. List the names of all employees who have a dependent with the same first name as themselves.
9. Find the names of all employees who are directly supervised by 'Franklin Wong'.

10. Find a list of all project numbers for projects that involve an employee whose last name is 'Smith', either as a worker or as a manager of the department that controls the project.

11. Find the names of all instructors whose department name includes the substring 'co'.

Use UNIVERSITY database schema for the following exercise.

1. Find the titles of courses in the Comp. Sci. department that have 3 credits.

2. Find the highest salary of any instructor.

3. Find the set of all courses taught either in Fall 2009 as well as in Spring 2010.

4. Find the set of all courses taught either in Fall 2009 but not in Spring 2010.

LAB No. 4: COMPLEX QUERIES IN SQL

Aggregate Functions

Aggregate functions are functions that take a collection (a set or multiset) of values as input and return a single value. SQL offers five built-in aggregate functions:

- Average: **avg**
- Minimum: **min**
- Maximum: **max**
- Total: **sum**
- Count: **count**

The input to **sum** and **avg** must be a collection of numbers, but the other operators can operate on collections of nonnumeric data types, such as strings, as well.

Example 4.1: (Use university database)

Find the average salary of instructors in the Computer Science department." We write this query as follows:

select avg (*salary*)
from *instructor*
where *dept name*= 'Comp. Sci.'

Example 4.2 Find the total number of instructors who teach a course in the Spring 2010 semester.

select count (**distinct** ID)
from teaches
where semester = 'Spring' **and** year = 2010

Aggregation with Grouping

Queries using aggregate functions (COUNT,AVG,MIN,MAX,SUM),Group by, Order by, Having

ORDER BY

The ORDER BY clause is used to sort the tuples in a query result based on the values of some attribute(s).

Example 4.3: Retrieve a list of employees and the projects each works in, ordered by the employee's department, and within each department ordered alphabetically by employee last name. (Use company database)

```
SELECT DNAME, LNAME, FNAME, PNAME
FROM DEPARTMENT, EMPLOYEE, WORKS_ON, PROJECT
WHERE DNUMBER=DNO
AND SSN=ESSN
AND PNO=PNUMBER
ORDER BY DNAME, LNAME
```

Note: count (*) returns number of rows in the result of the query. (*) refers to rows (tuples).

The default order is in ascending order of values. We can specify the keyword DESC if we want a descending order; the keyword ASC can be used to explicitly specify ascending order, even though it is the default

Ex: ORDER BY DNAME **DESC**, LNAME **ASC**, FNAME **ASC**

GROUPING

- In many cases, we want to apply the aggregate functions to subgroups of tuples in a relation
- Each subgroup of tuples consists of the set of tuples that have the same value for the grouping attribute(s).

- The function is applied to each subgroup independently

- SQL has a GROUP BY-clause for specifying the grouping attributes, which must also appear in the SELECT-clause.

Example 5.2: For each department, retrieve the department number, the number of employees in the department, and their average salary. (Use COMPANY database)

```
SELECT DNO, COUNT (*), AVG (SALARY)
FROM EMPLOYEE GROUP BY DNO
```

THE HAVING-CLAUSE

Sometimes we want to retrieve the values of these functions for only those groups that satisfy certain conditions. The HAVING-clause is used for specifying a selection condition on groups (rather than on individual tuples)

Example 4.4: For each project on which more than two employees work, retrieve the project number, project name, and the number of employees who work on that project. (Use COMPANY database)

```
SELECT PNUMBER, PNAME, COUNT (*)
FROM PROJECT, WORKS_ON
WHERE PNUMBER=PNO
GROUP BY PNUMBER, PNAME
HAVING COUNT (*) > 2
```

Lab Exercises (Use company database following exercise)

1. Find the sum of the salaries of all employees, maximum salary, the minimum salary, and the average salary.

2. Find the sum of the salaries of all employees of the 'Research' department as well as the maximum salary, the minimum salary, and the average salary in this department.

3. Retrieve the total number of employees in the company.

4. Retrieve the number of employees in the 'Research' department.

5. Count the number of distinct salary values in the database.

6. For each department whose average employee salary is more than $30,000, retrieve the department name and the number of employees working for that department.

7. For each department, retrieve the department number, the number of employees in the department, and their average salary.

8. For each project, retrieve the project number, the project name, and the number of employees who work on that project.

9. For each project, retrieve the project number, the project name, and the number of employees from department 5 who work on the project.

Additional Exercises (Use University database)

1. Find the number of tuples in the course relation.

2. Find the average salary of instructors in the Computer Science department.

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

4. Find the average salary in each department.

5. Find the average salary of all instructor.

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

7. Find the average salary of instructors in those departments where the average salary is more than $42,000.

8. 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.

9. Find the enrollment of each section that was offered in Autumn 2009.

LAB NO. 5: VIEWS IN SQL

Views (Virtual Tables) in SQL

A **view** in SQL terminology is a single table that is derived from other tables. These other tables can be *base tables* or previously defined views. A view does not necessarily exist in physical form; it is considered to be a **virtual table**, in contrast to **base tables**, whose tuples are always physically stored in the database. This limits the possible update operations that can be applied to views, but it does not provide any limitations on querying a view.

A viewed table (also called a View/External view) provides a window onto one or more tables. A view is simply a predefined derived table.

Once a view is defined, you can use it as if it were a base table by naming it in SQL statements. We cannot add, delete, or update records in it. Because a view is a derived table.

SQL CREATE VIEW Syntax

CREATE VIEW view_name AS
SELECT column_name(s)
FROM table_name
WHERE condition;

Example 5.1: Create a view faculty who needs to access all data in the INSTRUCTOR relation except salary. (use University database)

create view *faculty* **as**
select *ID*, *name*, *dept name*
from *instructor*

Lab Exercises (Use university database)

1. Create a view that lists all course sections offered by Physics department in the Fall 2009 semester with the building and room number of each section.
2. Select all the courses from *Physics_fall_2009* view.
3.. Create a view *department_total_salary* consisting of department name and total salary of that department.
4. Create a view *instructor_info* that lists the ID, name, and building-name of each instructor in the University.
5. Create view *history_instructors* that lists all information of instructors in history department.

Use Company database for the following exercise

1. Create a view that has the department name, manager name, and manager salary for every department.
2. Create a view that has the employee name, supervisor name, and employee salary for each employee who works in the ‘Research’ department.
3. Create a view that has the project name, controlling department name, number of employees, and total hours worked per week on the project for each project.
4. Create a view that has the project name, controlling department name, number of employees, and total hours worked per week on the project for each project with more than one employee working on it.

LAB NO. 6: JOINS TYPES

Consider two relations student and takes.

1. CARTESIAN PRODUCT: Which concatenates each tuple of the first relation with every tuple of the second

```
SELECT *
FROM student, takes;
```

2. NATURAL JOIN: Considers only those pairs of tuples with the same value on those attributes that appear in the schemas of both relation

```
SELECT *
FROM student natural join takes;
```

3. INNER JOIN: Cartesian product followed by selection.(same as natural join)

```
SELECT *
FROM student AS S inner join takes AS T ON S.ID = T.ID;
```

4. LEFT OUTER JOIN: Cartesian product + selection but includes rows from left table which are unmatched.

```
SELECT *
FROM student AS S left outer join takes AS T ON S.ID = T.ID;
```

5. RIGHT OUTER JOIN: Cartesian product + selection but includes rows from right table which are unmatched.

```
SELECT *
FROM student AS S right outer join takes AS T ON S.ID = T.ID;
```

6. LEFT OUTER JOIN: It is the combination of both left outer and right outer join

```
SELECT *
FROM student AS S full outer join takes AS T ON S.ID = T.ID;
```

Lab Exercises

1. Create student and takes tables with the following structure.

Takes

ID	course_id	sec_id	semester	year	grade
00128	CS-101	1	Fall	2009	A
00128	CS-347	1	Fall	2009	A-
12345	CS-101	1	Fall	2009	C
12345	CS-190	2	Spring	2009	A
12345	CS-315	1	Spring	2010	A
12345	CS-347	1	Fall	2009	A
19991	HIS-351	1	Spring	2010	B
23121	FIN-201	1	Spring	2010	C+
44553	PHY-101	1	Fall	2009	B-
45678	CS-101	1	Fall	2009	F
45678	CS-101	1	Spring	2010	B+
45678	CS-319	1	Spring	2010	B
54321	CS-101	1	Fall	2009	A-
54321	CS-190	2	Spring	2009	B+
55739	MU-199	1	Spring	2010	A-
76543	CS-101	1	Fall	2009	A
76543	CS-319	2	Spring	2010	A
76653	EE-181	1	Spring	2009	C
98765	CS-101	1	Fall	2009	C-
98765	CS-315	1	Spring	2010	B
98988	BIO-101	1	Summer	2009	A
98988	BIO-301	1	Summer	2010	*null*

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
98765	Bourikas	Elec. Eng.	98
98988	Tanaka	Biology	120

2. Compute the Cartesian product of student and takes relation.
3. Compute the inner join of student and takes relation.
4. Compute the natural join of student and takes relation.
5. Compute the left outer join of student and takes relation.
6. Compute the right outer join of student and takes relation.
7. Compute the full outer join of student and takes relation.

LAB No. 7 to 11: MINI PROJECT

Develop menu driven project for management of database application such as Library Information System, Inventory Control system, Student Information System, Time Table Development System etc.

All the students are instructed to form a team of two members and submit title of the project on 7th week.. Students need to submit report and show the implementation on 12th week.

Report Format:

- Title, Team Members
- Problem Statement
- Relational database schema with at least 5 relations.
- Integrity constraints

For the selected problem statement:

1. Design and implement the database.
2. Implement basic and complex queries. (at least 5 basic and 5 complex queries)

LAB NO.12 Mini Project Evaluation

Report 10 Marks, Implementation 15 Marks and 5 Marks viva. Total 30 marks

LAB NO. 13: Lab Test

For lab test experiments will be covered from Lab No. 1 to 6. Lab test duration is 2 hours duration. Write up 15 Marks and Implementation 25 Marks. Total Marks is 40

References:

1. A. Silberschatz, H. F. Korth, and S. Sudarshan, "*Database System Concepts*", (6e), McGraw Hill, 2013.

2. R. Elmasri and S. B. Navathe, "Fundamentals of Database Systems", (7e), Pearson, 2017.

3. M. Tamer Ozsu, Patrick Valduriez, *"Principles of Distributed Database Systems",* (3e), Springer, 2011.