Relational Commercial Languages - SQL

Unit 4: SQL (10 Hours) Data Definition and Data Types *(Attribute Data Types and Domains Create Database, Create Table, Drop Table, Drop Constraint, Drop Database);* Specifying Constraints *(Attribute Constraints, Attribute Defaults, Key and Referential Integrity Constraints, Naming Constraints, Constraint using CHECK);*	**2 Hours**
Basic Retrieval Queries *(SELECT-FROM-WHERE Structure, Ambiguous Attribute Names, Aliasing, Renaming, Tuple Variables; Unspecified WHERE clause, Use of * in Select, Substring Pattern Matching using LIKE, Arithmetic Operators, Order by Clause);*	**3 Hours**
Complex Retrieval Queries*; (Where Clause using IS NULL, Logical Connectives, Nested Query, Correlated Nested Query, Using BETWEEN, IN and EXISTS Clauses in Where, Renaming Attributes, JOIN, Natural JOIN, OUTER JOIN (Left/Right), Aggregate Functions, GROUP BY and HAVING Clause);*	**3 Hours**
INSERT, DELETE, and UPDATE Statements;	**1 Hour**
Views *(CREATE, DROP);*	**1 Hour**

Relational Commercial Languages - SQL

- **Structure Query Language(SQL) is a programming language used for storing and managing data in RDBMS.**
- **SQL was the first commercial language introduced for E.F Codd's Relational model.**
- **Today almost all RDBMS(MySql, Oracle, Infomix, Sybase, MS Access) uses SQL as the standard database language.**
- **SQL is used to perform all type of data operations in RDBMS.**

Data Definition, Constraints, and Schema Changes

Specifies a new base relation by giving it a name, and specifying each of its attributes and their data types (INTEGER, FLOAT, DECIMAL(i,j), CHAR(n), VARCHAR(n))
A constraint NOT NULL may be specified on an attribute.

```
CREATE TABLE DEPARTMENT
(
        DNAME                   VARCHAR(10)     NOT NULL,
        DNUMBER                 INTEGER         NOT NULL,
        MGRSSN                  CHAR(9),
        MGRSTARTDATE            CHAR(9)
);
```

Data Definition, Constraints, and Schema Changes

CREATE TABLE command for specifying the primary key attributes and referential integrity constraints (foreign keys).

```
CREATE TABLE DEPARTMENT (
        DNAME                   VARCHAR(10)  NOT NULL,
        DNUMBER                 INTEGER      NOT NULL,
        MGRSSN                  CHAR(9),
        MGRSTARTDATE            CHAR(9),
        PRIMARY KEY             (DNUMBER),
        FOREIGN KEY (MGRSSN) REFERENCES EMPLOYEE(Ssn);
```

Attribute Data Types and Domains in SQL

The basic data types available for attributes include numeric, character string, bit string, Boolean, date, and time.

- Numeric data types - INTEGER or INT, and SMALLINT) and floating-point (real) numbers of various precision (FLOAT or REAL, and DOUBLE PRECISION). Formatted numbers can be declared by using DECIMAL(i, j)—or DEC(i, j) or NUMERIC(i, j)—where i, the precision, is the total number of decimal digits and j, the scale, is the number of digits after the decimal point. The default for scale is zero.
- Character-string data types -CHAR(n) or CHARACTER(n), where n is the number of characters—or varying length—VARCHAR(n) or CHAR VARYING(n) or CHARACTER VARYING(n), where n is the maximum number of characters. Another variable-length string data type called CHARACTER LARGE OBJECT or CLOB is also available to specify columns that have large text values, such as documents. The CLOB maximum length can be specified in kilobytes (K), megabytes (M), or gigabytes (G). For example, CLOB(20M) specifies a maximum length of 20 megabytes.

Attribute Data Types and Domains in SQL

The basic data types available for attributes include numeric, character string, bit string, Boolean, date, and time.

- Bit-string data types - f fixed length n—BIT(n)—or varying length—BIT VARYING(n), where n is the maximum number of bits. The default for n, the length of a character string or bit string, is 1. Literal bit strings are placed between single quotes but preceded by a B to distinguish them from character strings; for example, B‘10101’. Another variable-length bitstring data type called BINARY LARGE OBJECT or BLOB is also available to specify columns that have large binary values, such as images. The maximum length of a BLOB can be specified in kilobits (K), megabits (M), or gigabits (G). For example, BLOB(30G) specifies a maximum length of 30 gigabits.
- A Boolean data type - TRUE or FALSE
- The DATE data type - has ten positions, and its components are YEAR, MONTH, and DAY in the form YYYY-MM-DD. The TIME data type has at least eight positions, with the components HOUR, MINUTE, and SECOND in the form HH:MM:SS. For example, DATE ‘2014-09-27’ or TIME ‘09:12:47’

Attribute Data Types and Domains in SQL

- The DATE data type - has ten positions, and its components are YEAR, MONTH, and DAY in the form YYYY-MM-DD. The TIME data type has at least eight positions, with the components HOUR, MINUTE, and SECOND in the form HH:MM:SS. For example, DATE '2014-09-27' or TIME '09:12:47'

  DATE - format YYYY-MM-DD
  DATETIME - format: YYYY-MM-DD HH:MI:SS
  TIMESTAMP - format: YYYY-MM-DD HH:MI:SS
  YEAR - format YYYY or YY

Data Definition, Constraints, and Schema Changes

Giving Names to Constraints

A constraint may be given a constraint name, following the keyword CONSTRAINT. The names of all constraints within a particular schema must be unique. A constraint name is used to identify a particular constraint in case the constraint must be dropped later and replaced with another constraint

```
CREATE TABLE DEPARTMENT (
        DNAME                   VARCHAR(10)  NOT NULL,
        DNUMBER                 INTEGER      CONSTRAINT dep_dno_pk PRIMARY KEY,
        MGRSSN                  CHAR(9),
        MGRSTARTDATE            CHAR(9),
        PRIMARY KEY             (DNUMBER),
        FOREIGN KEY (MGRSSN) REFERENCES EMPLOYEE(Ssn);
```

Data Definition, Constraints, and Schema Changes

Different ways to define Constraints:

```
CREATE TABLE employee(
id number(5) PRIMARY KEY,
name char(20),
dept char(10),
age number(2),
salary number(10),
location char(10)
);
```

```
CREATE TABLE employee(
id number(5) CONSTRAINT emp_id_pk PRIMARY KEY,
name char(20),
dept char(10),
age number(2),
salary number(10),
location char(10)
);
```

Data Definition, Constraints, and Schema Changes

Different ways to define Constraints:

```
CREATE TABLE employee(
id number(5),
name char(20),
dept char(10),
age number(2),
salary number(10),
location char(10),
CONSTRAINT emp_id_pk PRIMARY KEY (id)
);
```

```
CREATE TABLE employee
( id number(5), NOT NULL,
name char(20),
dept char(10),
age number(2),
salary number(10),
location char(10),
ALTER TABLE employee ADD CONSTRAINT
PK_EMPLOYEE_ID PRIMARY KEY (id)
);
```

Data Definition, Constraints, and Schema Changes

Define Constraints using UNIQUE:

```
CREATE TABLE employee(
id number(5) UNIQUE,
name char(20),
dept char(10),
age number(2),
salary number(10),
location char(10),
CONSTRAINT emp_id_pk PRIMARY KEY (id)
);
```

Data Definition, Constraints, and Schema Changes

CHECK Constraint:

CHECK restrict attribute or domain values using the clause following an attribute or domain definition. For example, suppose that department numbers are restricted to integer numbers between 1 and 20, use CHECK command as

Dnumber INT NOT NULL CHECK (Dnumber > 0 AND Dnumber < 21);

The CHECK clause can also be used in conjunction with the CREATE DOMAIN statement. For example, we can write the following statement:

CREATE DOMAIN D_NUM AS INTEGER CHECK (D_NUM > 0 AND D_NUM < 21);

Data Definition, Constraints, and Schema Changes

DROP TABLE <Table name>;
Used to remove a relation (base table) and its definition
The relation can no longer be used in queries, updates, or any other commands since its description no longer exists

Example:
DROP TABLE DEPENDENT;

Data Definition, Constraints, and Schema Changes

ALTER TABLE : Used to add an attribute to one of the base relations.

- The new attribute will have NULLs in all the tuples of the relation right after the command is executed; hence, the NOT NULL constraint is not allowed for such an attribute

Examples:
ALTER TABLE EMPLOYEE ADD JOB VARCHAR(12);

ALTER TABLE DEPARTMENT ADD PRIMARY KEY (DNUMBER);
ALTER TABLE table_name DROP column_name;

ALTER TABLE table_name MODIFY column_name datatype;

RENAME old_table_name To new_table_name;

Data Definition, Constraints, and Schema Changes

ALTER TABLE : Used to add an attribute to one of the base relations.

- The new attribute will have NULLs in all the tuples of the relation right after the command is executed; hence, the NOT NULL constraint is not allowed for such an attribute

Examples:

```
ALTER TABLE EMPLOYEE ADD JOB VARCHAR(12);
ALTER TABLE DEPARTMENT ADD PRIMARY KEY (DNUMBER);
ALTER TABLE employee ADD CONSTRAINT PK_EMPLOYEE_ID PRIMARY KEY (id)
ALTER TABLE table_name DROP column_name;
ALTER TABLE table_name MODIFY column_name datatype;
RENAME old_table_name To new_table_name;
```

REFERENTIAL INTEGRITY OPTIONS

We can specify RESTRICT, CASCADE, SET NULL or SET DEFAULT on referential integrity constraints (foreign keys)

```
CREATE TABLE DEPARTMENT (
DNAME          VARCHAR(10)  NOT NULL,
DNUMBER        INTEGER      NOT NULL,
MGRSSN         CHAR(9),
MGRSTARTDATE   CHAR(9),
PRIMARY KEY (DNUMBER),
FOREIGN KEY (MGRSSN) REFERENCES EMPLOYEE (Ssn)
ON DELETE SET DEFAULT ON UPDATE CASCADE);
```

REFERENTIAL INTEGRITY OPTIONS

```
CREATE TABLE EMPLOYEE (
ENAME        VARCHAR(30)  NOT NULL,
ESSN         CHAR(9),
BDATE        DATE,
DNO          INTEGER  DEFAULT 1,
SUPERSSN     CHAR(9),
PRIMARY KEY (ESSN),
FOREIGN KEY (DNO) REFERENCES DEPARTMENT (Dnumber)
ON DELETE SET DEFAULT ON UPDATE  CASCADE,
FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE (Ssn)
ON DELETE SET NULL ON UPDATE CASCADE);
```

Specifying Updates in SQL

There are three SQL commands to modify the database: INSERT, DELETE, and UPDATE

INSERT:
In its simplest form, it is used to add one or more tuples to a relation. Attribute values should be listed in the same order as the attributes were specified in the CREATE TABLE command.

Specifying Updates in SQL

Example:

- INSERT INTO EMPLOYEE
  VALUES ('Richard','K','Marini', '653298653', '30-DEC-52',
  '98 Oak Forest,Katy,TX', 'M', 37000,'987654321', 4);
- An alternate form of INSERT specifies explicitly the attribute names that correspond to the values in the new tuple.
- Attributes with NULL values can be left out

Example: Insert a tuple for a new EMPLOYEE for whom we only know the FNAME, LNAME, and SSN attributes.

```
INSERT INTO    EMPLOYEE (Fname, Lname, Ssn)
VALUES ('Richard', 'Marini', '653298653');
```

DELETE

Removes tuples from a relation

- Includes a WHERE-clause to select the tuples to be deleted
- Referential integrity should be enforced
- Tuples are deleted from only one table at a time (unless CASCADE is specified on a referential integrity constraint)
- A missing WHERE-clause specifies that all tuples in the relation are to be deleted; the table then becomes an empty table
- The number of tuples deleted depends on the number of tuples in the relation that satisfy the WHERE-clause

DELETE

Examples:

```
DELETE FROM EMPLOYEE
WHERE Lname='Brown';

DELETE FROM EMPLOYEE
WHERE Ssn='123456789';

DELETE FROM EMPLOYEE
WHERE Dno  IN
                (SELECT Dnumber
                FROM  DEPARTMENT
                WHERE Dname='Research');

DELETE FROM EMPLOYEE;
```

UPDATE

- Used to modify attribute values of one or more selected tuples
- A WHERE-clause selects the tuples to be modified
- An additional SET-clause specifies the attributes to be modified and their new values
- Each command modifies tuples in the same relation
- Referential integrity should be enforced

UPDATE

Example: Change the location and controlling department number of project number 10 to 'Bellaire' and 5, respectively.

```
UPDATE   PROJECT
SET   Plocation = 'Bellaire', Dnum = 5
WHERE    Pnumber=10;
```

Summary of SQL Queries

A query in SQL can consist of up to six clauses.
But only the first two SELECT and FROM are mandatory. Other clauses are optional and are specified in the following order:

```
SELECT      <attribute list>
FROM        <table list>
[WHERE      <condition>]
[GROUP BY   <grouping attribute(s)>]
[HAVING     <group condition>]
[ORDER BY   <attribute list>];
```

Summary of SQL Queries

SELECT - lists the attributes or functions to be retrieved

FROM - specifies all relations (or aliases) needed in the query but not those needed in nested queries

WHERE - specifies the conditions for selection and join of tuples from the relations specified in the FROM-clause

GROUP BY - specifies grouping attributes

HAVING - specifies a condition for selection of groups

ORDER BY - specifies an order for displaying the result of a query

A query is evaluated by first applying the WHERE-clause, then GROUP BY and HAVING, and finally the SELECT-clause

Example: COMPANY Database

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

Summary of SQL Queries

Example: 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.

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

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

Summary of SQL Queries

Example: 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.

```
SELECT      Pnumber, Pname, COUNT(*)
FROM        PROJECT, WORKS_ON
WHERE       Pnumber=Pno
GROUP BY    Pno
HAVING      COUNT (*) > 2 ;
```

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

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

Basic form of the SQL

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.

Basic form of the SQL

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, and to the C/C++ programming language operators =, <, <=, >, >=, and !=.

The main syntactic difference is the not equal operator. SQL has additional comparison operators.

SQL Queries

Example: Retrieve the SSN values for all employees.

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

SQL Queries (Unspecified WHERE Clause and Use of the Asterisk)

Example: Retrieve the SSN values for all employees.

SELECT SSN
FROM EMPLOYEE;

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

Example:

SELECT SSN
FROM EMPLOYEE, DEPARTMENT;

If more than one relation is specified in the FROM-clause and there is no join condition, then the CARTESIAN PRODUCT of tuples is selected

Example:

SELECT SSN, DNAME
FROM EMPLOYEE, DEPARTMENT

It is extremely important not to overlook specifying any selection and join conditions in the WHERE-clause; otherwise, incorrect and very large relations may result

SQL Queries

Example: To eliminate duplicate tuples in a query result, the keyword DISTINCT is used

```
SELECT  SALARY
FROM   EMPLOYEE

SELECT  DISTINCT SALARY
FROM   EMPLOYEE;
```

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

Basic SQL Queries

Example: To retrieve all the attribute values of the selected tuples, a * is used, which stands for all the attributes.

```
SELECT  *
FROM  EMPLOYEE
WHERE Dno=5;

SELECT  *
FROM  EMPLOYEE, DEPARTMENT
WHERE Dname='Research' AND Dno=Dnumber ;
```

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

Basic SQL Queries

Example: Retrieve the birthdate and address of the employee whose name is 'John B. Smith'.

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

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

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

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

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

Basic SQL Queries

Example: For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birthdate.

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

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

SQL Queries

Example: For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birthdate.

```
SELECT   Pnumber, Dnum, Lname, Bdate, Address
FROM     PROJECT, DEPARTMENT, EMPLOYEE
WHERE    Dnum=Dnumber AND Mgrssn=Ssn AND
         Plocation='Stafford‘;
```

There are two join conditions. The join condition Dnum=Dnumber relates a project to its controlling department. The join condition MGRSSN=SSN relates the controlling department to the employee who manages that department

GROUP BY

Use
of
GROUP BY

GROUP BY

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

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

GROUP BY

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

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

- The EMPLOYEE tuples are divided into groups- Each group having the same value for the grouping attribute Dno
- The COUNT and AVG functions are applied to each such group of tuples separately
- The SELECT-clause includes only the grouping attribute and the functions to be applied on each group of tuples .
- A join condition can be used in conjunction with grouping

GROUP BY

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

```
SELECT      Pnumber, Pname, COUNT (*)
FROM        PROJECT, WORKS_ON
WHERE       Pnumber=Pno
GROUP BY    Pnumber;
```

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

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

In this case, the grouping and functions are applied after the joining of the two relations

THE HAVING-CLAUSE

Use
of
HAVING

THE HAVING-CLAUSE

Example: 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.

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

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

THE HAVING-CLAUSE

Example: 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.

```
SELECT      Pnumber, Pname, COUNT(*)
FROM        PROJECT, WORKS_ON
WHERE       Pnumber=Pno
GROUP BY    Pnumber
HAVING      COUNT (*) > 2;
```

ORDER BY

Use
of
ORDER BY

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: 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.

```
SELECT    Dname, Lname, Fname, Pname
FROM      DEPARTMENT, EMPLOYEE, WORKS_ON, PROJECT
WHERE     Dnumber=Dno AND Ssn=Essn AND Pno=Pnumber
ORDER BY  Dname, Lname;
```

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

ORDER BY

Example:
Retrieve a list of employees and the projects they are working on, ordered by department and, within each department, ordered alphabetically by last name, then first name.

```
SELECT D.Dname, E.Lname, E.Fname, P.Pname
FROM DEPARTMENT AS D, EMPLOYEE AS E, WORKS_ON AS W, PROJECT AS P
WHERE D.Dnumber = E.Dno AND E.Ssn = W.Essn AND W.Pno = P.Pnumber
ORDER BY D.Dname, E.Lname, E.Fname;
```

if we want descending alphabetical order on Dname and ascending order on Lname, Fname, the ORDER BY clause can be written as ORDER BY D.Dname DESC, E.Lname ASC, E.Fname ASC

NESTING OF QUERIES

A complete SELECT query, called a nested query, can be specified within the WHERE-clause of another query, called the outer query. Many of the previous queries can be specified in an alternative form using nesting

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

```
SELECT    FNAME, LNAME, ADDRESS
FROM      EMPLOYEE
WHERE     DNO IN   (SELECT  DNUMBER
                    FROM DEPARTMENT
                    WHERE    DNAME='Research' );
```

NESTING OF QUERIES

SELECT column_name1, column_name2
FROM table1 , table2
WHERE column_name OPERATOR
(SELECT column_name1, column_name2
FROM table1, table2
WHERE condition);

Example:

Operator can be IN, =, < , > , <> etc
WHERE column_name IN (Nested Queries)
Or
WHERE column_name = (Nested Queries)

NESTING OF QUERIES

Operators in The WHERE Clause

Operator	Description
=	Equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
<>	Not equal. Note: In some versions of SQL this operator may be written as !=
BETWEEN	Between a certain range
LIKE	Search for a pattern
IN	To specify multiple possible values for a column

AGGREGATE FUNCTIONS

COUNT, SUM, MAX, MIN, and AVG

Example: Find the maximum salary, the minimum salary, and the average salary among all employees.

```
SELECT   MAX(Salary), MIN(Salary), AVG(Salary)
FROM     EMPLOYEE;
```

The above aggregate function can be done using GROUP BY clause and applies to group attributes.

AGGREGATE FUNCTIONS

Example: Find the maximum salary, the minimum salary, and the average salary among employees who work for the 'Research' department.

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

AGGREGATE FUNCTIONS

Example: Find the maximum salary, the minimum salary, and the average salary among employees who work for the 'Research' department.

```
SELECT   MAX(Salary), MIN(Salary), AVG(Salary)
FROM     EMPLOYEE, DEPARTMENT
WHERE    Dno=Dnumber AND Dname='Research';
```

AGGREGATE FUNCTIONS - COUNT

Example: Retrieve the total number of employees in the company

```
SELECT   COUNT (*)
FROM     EMPLOYEE
```

Example: Retrieve the number of employees in the 'Research' department

```
SELECT   COUNT (*)
FROM     EMPLOYEE, DEPARTMENT
WHERE    DNO=DNUMBER AND DNAME='Research';
```

Ambiguous Attribute Names

SELECT EMPLOYEE.Fname, EMPLOYEE.LName, EMPLOYEE.Address
FROM EMPLOYEE, DEPARTMENT
WHERE DEPARTMENT.DName = 'Research' AND
DEPARTMENT.Dnumber = EMPLOYEE.Dno;

Aliasing

SELECT E.Fname, E.Lname, S.Fname, S.Lname
FROM EMPLOYEE AS E, EMPLOYEE AS S
WHERE E.Super_ssn = S.Ssn;

It is also possible to rename the relation attributes within the query in SQL by giving them aliases. For example, if we write

EMPLOYEE AS E(Fn, Mi, Ln, Ssn, Bd, Addr, Sex, Sal, Sssn, Dno) in the FROM clause, Fn becomes an alias for Fname, Mi for Minit, Ln for Lname, and so on.

SELECT Fn, Mi, Ln
FROM EMPLOYEE AS E(Fn, Mi, Ln, Ssn, Bd, Addr, Sex, Sal, Sssn, Dno);

Ambiguous Attribute Names, Aliasing, Renaming, and Tuple Variables

Aliasing or Renaming

```
SELECT E.Fname, E.LName, E.Address
FROM EMPLOYEE AS E, DEPARTMENT AS D
WHERE D.DName = 'Research' AND D.Dnumber = E.Dno;
```

Substring Pattern Matching and Arithmetic Operators

Substring Pattern Matching

This feature allows comparison conditions on only parts of a character string, using the LIKE comparison operator. This can be used for string pattern matching. Partial strings are specified using two reserved characters: % replaces an arbitrary number of zero or more characters, and the underscore (_) replaces a single character.
Example:
Query. 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. 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.

Substring Pattern Matching

To retrieve all employees who were born during the 1970s. 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.

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

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


Arithmetic Operators:

Another feature allows the use of arithmetic in queries. The standard arithmetic operators for addition (+), subtraction (−), multiplication (*), and division (/) can be applied to numeric values or attributes with numeric domains. For example, suppose that we want to see the effect of giving all employees who work on the 'ProductX' project a 10% raise;

Example:
Show the resulting salaries if every employee working on the 'ProductX' project is given a 10% raise.

```
SELECT E.Fname, E.Lname, 0.1 * E.Salary AS Increased_sal
FROM EMPLOYEE AS E, WORKS_ON AS W, PROJECT AS P
WHERE E.Ssn = W.Essn AND W.Pno = P.Pnumber AND
P.Pname = 'ProductX';
```

Substring Pattern Matching and Arithmetic Operators

Arithmetic Operators:

Retrieve all employees in department 5 whose salary is between Rs30,000 and Rs40,000.

```
SELECT *
FROM EMPLOYEE
WHERE (Salary BETWEEN 30000 AND 40000) AND Dno = 5;
```

The condition (Salary BETWEEN 30000 AND 40000) in is equivalent to the condition ((Salary >= 30000) AND (Salary <= 40000)).

VIEW

Example: SQL CREATE VIEW Examples.

In SQL, a view is a virtual table based on the result-set of an SQL statement. A view contains rows and columns, just like a real table. You can add SQL statements and functions to a view and present the data as if the data were coming from one single table.

CREATE VIEW Syntax

```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

VIEW

Example: SQL CREATE VIEW Examples.

CREATE VIEW Syntax

```
CREATE VIEW Brazil_Customers AS
SELECT CustomerName, ContactName
FROM Customers
WHERE Country = 'Brazil';
```

We can query the view above using SQL query as:

```
SELECT * FROM Brazil_Customers;
```

SET Operations in SQL

Four different types of SET operations, along with example:

1. **UNION**
2. **UNION ALL**
3. **INTERSECT**
4. **MINUS**

https://www.w3schools.com/sql/sql_in.asp

SET Operations in SQL

UNION Operation

UNION is used to combine the results of two or more SELECT statements. However it will eliminate duplicate rows from its result set. In case of union, number of columns and datatype must be same in both the tables, on which UNION operation is being applied.

Union SQL query will be,

```
SELECT *
FROM First
UNION
SELECT *
FROM Second;
```

SET Operations in SQL

UNION Operation

Example of UNION

First	
ID	Name
1	abhi
2	adam

Second	
ID	Name
2	adam
3	Chester

SELECT * FROM First
UNION
SELECT * FROM Second;

The result set table will look like:

ID	NAME
1	abhi
2	adam
3	Chester

SET Operations in SQL

UNION ALL

This operation is similar to Union. But it also shows the duplicate rows.

First

ID	Name
1	abhi
2	adam

Second

ID	Name
2	adam
3	Chester

SELECT * FROM First
UNION ALL
SELECT * FROM Second;

The result set table will look like:

ID	NAME
1	abhi
2	adam
2	adam
3	Chester

SET Operations in SQL

Intersect query will be,

First

ID	Name
1	abhi
2	adam

Second

ID	Name
2	adam
3	Chester

**SELECT *
FROM First**

INTERSECT

**SELECT *
FROM Second;**

ID	NAME
2	adam

SET Operations in SQL

```
SELECT *
FROM A

MINUS

SELECT *
FROM B;
```

SET Operations in SQL

In SQL Server, we can write a Cartesian product using CROSS JOIN command as follows.

```
SELECT *
FROM A
CROSS JOIN
B;
```

Alternatively, we can simply use a comma to replace CROSS JOIN notation:

```
SELECT *
FROM A, B;
```

SET Operations in SQL

SQL JOIN

- **Cross JOIN or Cartesian Product**
- **INNER Join or EQUI Join**
- **Natural JOIN**
- **OUTER JOIN**
- **LEFT Outer Join**
- **RIGHT Outer Join**
- **Full Outer Join**

SET Operations in SQL

Example of Cross JOIN (Cartesian Product)

```
SELECT *
FROM class
CROSS JOIN
class_info;
```

Table1: Class

ID	NAME
1	abhi
2	adam
4	alex

Table2: class_info

ID	Address
1	DELHI
2	MUMBAI
3	CHENNAI

ID	NAME	ID	Address
1	abhi	1	DELHI
2	adam	1	DELHI
4	alex	1	DELHI
1	abhi	2	MUMBAI
2	adam	2	MUMBAI
4	alex	2	MUMBAI
1	abhi	3	CHENNAI
2	adam	3	CHENNAI
4	alex	3	CHENNAI

SET Operations in SQL

INNER Join or EQUI Join

This is a simple JOIN in which the result is based on matched data as per the equality condition specified in the SQL query.

Syntax is,

```
SELECT *
FROM class
INNER JOIN
class_info
WHERE class.id = class_info.id;
```

SET Operations in SQL

Example of INNER JOIN

Syntax is,

```
SELECT *
FROM class INNER JOIN class_info
WHERE class.id = class_info.id;
```

Table: class	
ID	NAME
1	abhi
2	adam
3	alex
4	anu

Table: class_info	
ID	Address
1	DELHI
2	MUMBAI
3	CHENNAI

Resultant Table			
ID	NAME	ID	Address
1	abhi	1	DELHI
2	adam	2	MUMBAI
3	alex	3	CHENNAI

SET Operations in SQL

Natural JOIN

Natural Join is a type of Inner join which is based on column having same name and same datatype present in both the tables to be joined.

Syntax is,

```
SELECT *
FROM class NATURAL JOIN class_info;
```

SET Operations in SQL

Example of NATURAL JOIN

```
SELECT *
FROM class NATURAL JOIN class_info;
```

Table: class	
ID	NAME
1	abhi
2	adam
3	alex
4	anu

Table: class_info	
ID	Address
1	DELHI
2	MUMBAI
3	CHENNAI

Resultant Table		
ID	NAME	Address
1	abhi	DELHI
2	adam	MUMBAI
3	alex	CHENNAI

SET Operations in SQL

OUTER JOIN

Outer Join is based on both matched and unmatched data.

Outer Joins subdivide further into,

1. **Left Outer Join**
2. **Right Outer Join**
3. **Full Outer Join**

SET Operations in SQL

LEFT Outer Join

The left outer join returns a resultset table with the matched data from the two tables and then the remaining rows of the left table and null from the right table's columns.

Syntax for Left Outer Join is,

```
SELECT column-name1, column-name2
FROM table1
LEFT OUTER JOIN table2
ON table1.column-name = table2.column-name;
```

SET Operations in SQL

RIGHT Outer Join

The right outer join returns a resultset table with the matched data from the two tables being joined, then the remaining rows of the right table and null for the remaining left table's columns.

Syntax for Right Outer Join is,

```
SELECT column-name1, column-name2
FROM table1
RIGHT OUTER JOIN table2
ON table1.column-name = table2.column-name;
```

SET Operations in SQL

Full Outer Join

The full outer join returns a resultset table with the matched data of two table then remaining rows of both left table and then the right table.

Syntax for Full Outer Join is,

```
SELECT column-name1, column-name2
FROM table1 FULL OUTER JOIN table2
ON table1.column-name = table2.column-name;
```

The EXISTS and NOT EXISTS Functions in SQL

EXISTS and NOT EXISTS

EXISTS and NOT EXISTS are Boolean functions that return TRUE or FALSE; hence, they can be used in a WHERE clause condition. EXISTS and NOT EXISTS are typically used in conjunction with a correlated nested query.

The EXISTS function in SQL is used to check whether the result of a nested query is empty (contains no tuples) or not. The result of EXISTS is a Boolean value TRUE if the nested query result contains at least one tuple, or FALSE if the nested query result contains no tuples.

```
SELECT Fname, Lname
FROM EMPLOYEE
WHERE EXISTS ( SELECT *
                FROM DEPENDENT
                WHERE Ssn = Essn );
```

The EXISTS and NOT EXISTS Functions in SQL

EXISTS and NOT EXISTS

Example: Retrieve the names of employees who have atleast one dependents.

```
SELECT Fname, Lname
FROM EMPLOYEE
WHERE EXISTS ( SELECT *
                FROM DEPENDENT
                WHERE Ssn = Essn );
```

The correlated nested query retrieves all DEPENDENT tuples related to a particular EMPLOYEE tuple. If exist, the EMPLOYEE tuple is selected because the WHERE-clause condition will evaluate to TRUE in this case.

Example: Retrieve the names of employees who have no dependents.

```
SELECT Fname, Lname
FROM EMPLOYEE
WHERE NOT EXISTS ( SELECT *
                FROM DEPENDENT
                WHERE Ssn = Essn );
```

VIEWS

VIEWS (Create and Drop)

Views in SQL are kind of virtual tables with rows and columns as they are in a real table in the database. We can create a view by selecting fields from one or more tables present in the database. A View can either have all the rows of a table or specific rows based on certain condition.
We learn about creating , deleting (dropping) Views.

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

VIEWS

Creating Views:

We can create View using CREATE VIEW statement. A View can be created from a single table or multiple tables.

Syntax:

```
CREATE VIEW view_name AS
SELECT column1, column2.....
FROM table_name
WHERE condition;
```

view_name: Name for the View
table_name: Name of the table
condition: Condition to select rows

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

VIEWS

Creating Views:

Example1: Create views with views name <emp_salary> with fields Fname, Lname, Dno and Salary of department no 5.

```
CREATE VIEW emp_salary AS
SELECT Fname, Lname, Dno, Salary
FROM EMPLOYEE
WHERE Dno=5;
```

To see the data in the View, we can query the view in the same manner as we query a table.

```
SELECT  *
FROM emp_salary;
```

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

Example2: Create views with views name <emp_dep> with fields Fname, Lname, Dname and Salary of employees working in department no 5.

VIEWS

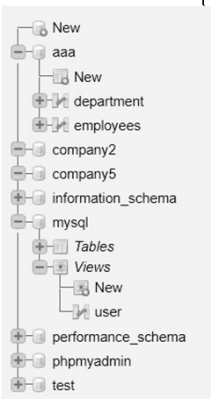
DELETING VIEWS:

SQL allows us to delete an existing View if we don't need the views Tables. We can delete or drop a View using the DROP statement.

Syntax:

```
DROP VIEW view_name;
```

```
DROP VIEW emp_salary;
```



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