The name SQL is derived from Structured Query Language. Originally, SQL was called SEQUEL (for Structured English Query Language) and was designed and implemented at IBM Research for a database system called System R. A joint effort between ANSI and ISO has led to standard version of SQL, called SQL-86 or SQL1. This version was expanded to a new standard SQL2.

SQL is a comprehensive database language; it has statement for data definition, query and update. Hence, it is both a DDL and DML. In addition, it has facilities for defining views on the database for specifying security and authorization, for defining integrity constraints, and for specifying transaction control. It also has rules for embedding SQL statements into general-purpose programming languages.

Data Definition, Constraints and Schema Changes in SQL:

SQL uses the terms table, row and column for relation, tuple, and attribute, respectively. The SQL commands for data definition are CREATE, ALTER and DROP.

Schema and Catalog Concepts in SQL:

The concept of an SQL schema was introduced in order to group together tables and other constructs that belong to the same database application. An SQL schema is identified by a schema name, and includes an authorization identifier to indicate the user or account who owns the schema, as well as descriptors for each element in the schema. Schema elements include the tables, constraints, views, domains and others.

A schema is created via the CREATE SCHEMA statement, which can include all the schema elements definitions. Alternatively, the schema can be defined by giving it a name and an authorization and its elements can be defined later on.

**CREATE SCHEMA COMPANY AUTHORIZATION ‘Jsmith’**
This statement creates a schema called company, owned by the user with authorization identifier Jsmith.

SQL also uses the concept of catalog- a named collection of schemas in an SQL environment. A catalog always contains a special schema called INFORMATION_SCHEMA, which provides information on all the element descriptors of all the schemas in the catalog to authorized users. Integrity constraints such as referential integrity can be defined between relations only if they exist in the same catalog. Schemas within the same catalog can also share elements, such as domain definitions.

The CREATE TABLE Command in SQL:

The CREATE TABLE command is used to specify a new relation by giving it a name and specifying attributes and initial constraints.

The general format:

```
CREATE TABLE newtablename(
    Three-part attribute description,
    Three-part attribute description,
    Optional table constraints
    ...
);
```

The attributes are specified first and each attribute is given a name, a data type to specify its domain of values and any attribute constraints such as NOT NULL. The
key, entity integrity and referential integrity constraints can be specified after the
attributes are declared. Or they can be added later using ALTER TABLE command.

The SQL schema in which the relations are declared is implicitly specified in the
environment in which the CREATE TABLE statements are executed. It can explicitly
be specified by attaching the schema name to the relation names using the dot (.)
operator.

**CREATE TABLE** COMPANY.EMPLOYEE rather than **CREATE TABLE** EMPLOYEE

The relations declared through CREATE TABLE statement are called base tables (or
base relations). These are actually created and stored as a file by the DBMS. Base
relations are distinguished from virtual relations created through the CREATE
VIEW statement, which may or may not correspond to an actual physical file.

In SQL, the attributes in a base table are considered to be ordered in the
sequence in which they are specified, in the CREATE TABLE statement.

Attributes Data Types and Domains in SQL:

The basic data types available:

Numeric data types include:

- integer numbers: INTEGER, INT or SMALLINT.
- Floating point: real numbers: FLOAT or REAL, DOUBLE PRECISION.
- Formatted numbers can be declared using DECIMAL(i,j) or DEC(i,j) or
  NUMERIC(i, j) – where i is the total number of decimal digits and j, the
  number of digits after the decimal point.

Character-string: Data types are either fixed length –or varying length strings.

- CHAR(n) or CHARACTER(n) where n is the number of characters.
- VARCHAR(n), CHAR VARYING (n) or CHARACTER VARYING(n), where n is the
  maximum number of characters. When specifying a literal string value, it
  should be placed between single quotes. Strings are case sensitive. In case
  the data item is shorter than the allocated size, it is padded with blank
spaces. For comparison purposes, strings are considered ordered in lexicographic order.

- Concatenation operator, denoted `||` that can concatenate two strings in SQL.
- CHARACTER LARGE OBJECT or CLOB is used to specify the data type of columns that have large text values, such as documents. The CLOB maximum length can be specified in Kilobytes (K), Megabytes (M) or Gigabytes (G), example CLOB(20M)

Bit-string data types are bit arrays that compactly stores bits or BOOLEAN values.

- Fixed length n declared as BIT(n) or
- varying length -BIT VARYING(n) where n is the maximum number of bits.
  The default for n is 1. Literal bit strings are placed between single quotes and preceded by B, example B’0110’.
- BINARY LARGE OBJECT or BLOB is available to specify columns that have large binary values, such as images. The BLOB maximum length can be specified in Kilobytes (K), Megabytes (M) or Gigabytes (G), example BLOB(20M)

Boolean data type: Have the traditional TRUE, FALSE values. In SQL because of the presence of NULL values, a three-valued logic is used (TRUE, FALSE and UNDETERMINED)

The DATE data type has 10 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. SQL checks for valid dates and times and there is a check between the month number and its allowed number of days as well. Time values are represented like literal strings between single quotes preceded by the keywords DATE or TIME. Example: DATE’2012-02-09’ or TIME’11:12:12’
The timestamp data type (TIMESTAMP) includes the DATE and TIME fields and other items. Example: TIMESTAMP ‘2012-02-09 09:23:12.67455’.

User defined Data types:
SQL allows users to specify the data type directly using the CREATE DOMAIN statement. This makes it easier to change the schema and also improves the schema readability.

```
CREATE DOMAIN SSN_TYPE AS CHAR(9);
```

We can use SSN_TYPE in place of CHAR(9) for the relevant attributes in our database relations.

**Specifying constrains and default values in SQL:**

Constraints on Attribute Values:

Since SQL allows NULL for attribute values, a constraint NOT NULL may be specified if NULL is not allowed for a particular attribute. This constraint should always be specified for the primary key attributes of each relation. For example:

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

An attribute can be assigned a default value by appending the clause `DEFAULT<value>` to an attribute definition. The default value is included in any new tuple if an explicit value is not provided for the attribute. If no default clause is specified, the default `default value(!)` is NULL.

```
Dno INT NOT NULL DEFAULT 1,
```

The CHECK clause can also be used in conjunction with CREATE DOMAIN statement. For example:
CREATE DOMAIN D_NUM AS INTEGER CHECK (D_NUM >0 AND D_NUM< 21)
We can then use the created domain D_NUM as the attribute data type for all attributes that refer to department numbers.

Dno    D_NUM    DEFAULT 1,

Specifying Key and Referential Integrity Constraints:
Because keys and referential integrity constraints are very important, there are special clauses within the CREATE TABLE statement to specify them. These are table constraints
The PRIMARY KEY clause specifies that one or more attributes make up the primary key of the relation.
If a primary key has a single attribute, the clause can follow the attribute directly. For example:

DNumber  INT PRIMARY KEY;

The UNIQUE clause specifies alternate (secondary) keys.

Dname   VARCHAR(15) UNIQUE;

Table constraints
A table constraint, identified by the word CONSTRAINT
A constraint is defined as:

CONSTRAINT constrainName Constraint clause,
CONSTRAINT is a reserved word,
constraintName is an identifier that uniquely identify the constraint.

Constraint Clauses:
Additional table constraints can be specified in a table, including keys and referential integrity.

Keys constraints are specified in a PRIMARY KEY constraint and/or UNIQUE constraint.
The UNIQUE clause specifies alternate (or secondary) keys.
The PRIMARY KEY clause specified one or more attributes that make up the primary key of a relation.
Referential constraints: Referential integrity is specified via the FOREIGN KEY clause.

CREATE TABLE EMPLOYEE
( . . . ,
  Dno INT NOT NULL DEFAULT 1,
  CONSTRAINT EMPPK PRIMARY KEY (SSN),
  CONSTRAINT EMPSUPERFK
    FOREIGN KEY (SUPERSSN) REFERENCES EMPLOYEE(SSN)
    ON DELETE SET NULL ON UPDATE CASCADE,
  CONSTRAINT EMPDETFK
    FOREIGN KEY (Dno) REFERENCES DEPARTMENT(DNUMBER)
    ON DELETE SET DEFAULT ON UPDATE CASCADE
);

CREATE TABLE DEPART_LOCATIONS
( . . . ,
  PRIMARY KEY (Dnumber, Dlocation),
  FOREIGN KEY (SUPERSSN) REFERENCES EMPLOYEE(SSN)
    ON DELETE SET NULL ON UPDATE CASCADE,
  CONSTRAINT EMPDETFK
    FOREIGN KEY (Dnumber) REFERENCES DEPARTMENT(Dnumber)
    ON DELETE CASCADE ON UPDATE CASCADE
);
**Referential Trigger Actions:**

A referential integrity constraint can be violated when tuples are inserted or deleted or when a foreign key attribute value is modified. It is then necessary to specify the actions to be taken if such a constraint is violated upon deletion of a referenced tuple or upon modification of a referenced primary key value by attaching a referential trigger action clause to any foreign key value. The options for triggered action clauses are SET NULL, CASCADE and SET DEFAULT.

The action taken by the DBMS for SET NULL or SET DEFAULT is the same for both ON DELETE and ON UPDATE, the value of the affected referencing attributes is changed to NULL for SET NULL or to the specified default value for SET DEFAULT.

The action for CASCADE ON DELETE is to delete all the referencing tuples, whereas the action for CASCADE ON UPDATE is to change the value of the foreign key to the new primary key value for all referencing tuples.

```sql
CREATE TABLE DEPARTMENT
  ( . . ,
    MGRSSN CHAR(9) NOT NULL DEFAULT '888665555',
    ...,
    CONSTRAINT DEPTPK
      PRIMARY KEY (DNUMBER),
    CONSTRAINT DEPTSK
      UNIQUE (DNAME),
    CONSTRAINT DEPTMGRFK
      FOREIGN KEY (MGRSSN) REFERENCES EMPLOYEE(SSN)
    ON DELETE SET DEFAULT  ON UPDATE CASCADE
  )
```

The DROP SCHEMA and DROP TABLE Commands:

If a whole database schema is no longer needed, the DROP SCHEMA command can be used to delete the database. There are two DROP options: CASCADE and RESTRICT

**DROP SCHEMA COMPANY CASCADE**
Causes the database schema to be removed and all its tables.

**DROP SCHEMA COMPANY RESTRICT**
Causes the schema to be dropped only if there are no elements in it; otherwise the DROP command will not be executed.

In order to delete a base relation within a schema and its definition, we use the DROP TABLE clause. To remove all dependent relation by entering the command:

**DROP TABLE DEPENDENT CASCADE.**
If the RESTRICT option is used, a relation(table) is deleted only if it is not referenced in any constraints. With the CASCADE option, all such constraints and views that reference the table are dropped from the schema along with the table itself.

The ALTER TABLE Command:

The definition of a base table can be changed by using the ALTER TABLE command, which is a schema evolution command. Possible modifications include adding or dropping an attribute, changing an attribute definition, and adding or dropping table constraints.

**ALTER TABLE COMPANY.EMPLOYEE ADD JOB VARCHAR(12);**
Adds an attribute JOB for each individual EMPLOYEE tuple.

**ALTER TABLE COMPANY.EMPLOYEE DROP ADDRESS CASCADE;** ( to drop an attribute)
ALTER TABLE COMPANY.DEPARTMENT ALTER MGRSSN DROP DEFAULT;
(drop default value for the MGRSSN attribute)

ALTER TABLE COMPANY.DEPARTMENT ALTER MGRSSN SET DEFAULT ‘333445555’;
(changes default value to MGRSSN attribute).

Specifying Key and Constraints can also be dropped or changed on a table. A constraint must have been given a name when it is specified. A constraint is dropped by using the statement:

DROP CONSTRAINT ConstraintName CASCADE;

ADD CONSTRAINT ConstrainName ...

Basic Queries in SQL:

The SELECT statement: SELECT is the basic statement to retrieve information from a database. SQL allows a table (relation) to have two or more tuples identical in all their attribute values, consequently an SQL table is not a set of tuples, it is rather a bag of tuples. Some SQL relations are constrained to be sets because of key constraint has been declared.

The SELECT-FROM-WHERE Structure of SQL Queries:

The basic form of the SELECT statement, sometimes called a mapping is formed of the three clauses SELECT, FROM, and WHERE:
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 relation names
- <condition> is a conditional expression that identifies the tuples to be retrieved by the query.

A more general version is:

SELECT <attribute list>
FROM <table list>
[ WHERE <condition>]  
[ORDER BY <attribute list>]

with the clauses between [] being optional.

Q0: Retrieve the bithdate 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’;

QUERY 1:
Retrieve the name and address of all employees who work for the ‘Research’ department.
Q1:  
SELECT  
    FNAME, LNAME, ADDRESS  
FROM  
    EMPLOYEE, DEPARTMENT  
WHERE  
    DNAME= ‘Research’ AND DNUMBER=DNO

QUERY 2:  
For every project located in “Stafford”, list the project number, the  
controlling department number, and the department manager’s last name,  
address, and birthdate.

Q2:  
SELECT  
    PNUMBER, DNUM, LNAME, ADDRESS, BDATE  
FROM  
    PROJECT, DEPARTMENT, EMPLOYEE  
WHERE  
    PLOCATION= ‘Stafford’ AND DNUM=DNUMBER AND  
    MGRSSN=SSN;

Dealing with Ambiguous Attribute Names and Renaming:  
In SQL, the same name can be used for two (or more) attributes as long as  
the attributes are in different relations.  
If a query refers to two or more attributes with the same name, we must  
qualify the attribute name with the relation name and separate them with  
a dot(.). This is done by prefixing the relation name to the attribute name.  
Ambiguity also arises in the case of queries that refer to the same relation  
twice as in the following  
QUERY:  
For each employee, retrieve the employee’s first and last name and the  
first and last name of his or her immediate supervisor.

Q8:  
SELECT  
    E.FNAME, E.LNAME, S.FNAME, S.LNAME  
FROM  
    EMPLOYEE AS E, EMPLOYEE AS S  
WHERE  
    E.SUPERSSN=S.SSN

In this case, we declare tuple variable or aliases, for the EMPLOYEE relation.
An alias can follow the keyword **AS** or it can directly follow the relation name as in : **FROM** EMPLOYEE E, EMPLOYEE S

It is also possible to rename the relation attributes within the query in SQL by giving them aliases; for example:

EMPLOYEE AS E (FN,MI, LN, SSN,BD,ADDR,SEX,SAL,SSSN,DNO)

**Unspecified WHERE-Clause and Use of **

-A missing WHERE clause indicates that no condition on tuple selection. Hence this type of SELECT will retrieve all tuple of the relations specified in the FROM qualify and are selected for the query result.

QUERIES 9 and 10:

Select all employees ssns

```
SELECT SSN
FROM EMPLOYEE;
```

Select all combinations of EMPLOYEE SSN and DEPARTMENT DNAME in the database.

```
SELECT SSN, DNAME
FROM EMPLOYEE, DEPARTMENT;
```

It is very important to specify every selection in the WHERE clause. If such conditions are overlooked, incorrect or very large relations maybe result.

To retrieve all attributes values of selected tuples , we do not need to list each attribute name explicitly in SQL; we can just enter (*) which stands for all attribute.
QUERY: Retrieve all attributes of an EMPLOYEE and the attributes of the
DEPARTMENT he/she works in for every employee of the “Research” department.

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

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

Tables as Sets in SQL:

SQL does not automatically eliminate duplicate tuples in the results of queries:

- They are expensive.
- The user may want to see duplicate tuples

An SQL table with a key is however a set with no duplicate tuples.

If we want to eliminate duplicate tuples from the result of an SQL query, we use the keyword DISTINCT in the SELECT clause, so as to eliminate all duplicate tuples.

```
SELECT ALL Salary
```
FROM EMPLOYEE;

SELECT DISTINCT Salary
FROM EMPLOYEE;

SET Operations:

SQL has incorporated some Set building operations as defined in mathematics, such as set union (UNION) and set intersection (INTERSECT) and set difference (EXCEPT). The relations resulting from these set operations are sets of tuples, in which duplicates are eliminated from the results.

(SELECT DISTINCT Pnumber
FROM PROJECT, DEPARTMENT, EMPLOYEE
WHERE Dnum= Dnumber AND Mgr_ssn= Ssn AND Lname=‘Smith’);

UNION

(SELECT DISTINCT Pnumber
FROM PROJECT, WORKS_ON, EMPLOYEE
WHERE Pnumber= Pno AND Essn= Ssn AND Lname= ‘Smith’);

The first SELECT query retrieves the projects that involve ‘Smith’ as a manager of the department that controls the project.

The second SELECT query retrieves the projects that involve ‘Smith’ as worker on the project.
SQL also has corresponding multiset operations which use the set operator followed by the keyword ALL (UNION ALL, INTERSECT ALL, EXCEPT ALL). Their results are bags in that duplicates are not eliminated.

Substring Pattern Matching and arithmetic Operators:

String pattern matching:

The LIKE operator is used for comparing strings values in tuples and for string pattern matching.

Partial strings are specified using two reserved characters % replaces an arbitrary number (0 or more) of characters, whereas _ replaces a single character.

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

Retrieves all employees whose address is in Houston, Texas

```
SELECT Fname, Lname
FROM EMPLOYEE
WHERE BDate LIKE '__5________'
```

Retrieves all employees born in the 1950’s

Note:

- if the underscore or % is part of the string, it should be preceded by the keyword ESCAPE. For example ‘AB\_CD\%F’ ESCAPE ‘\’ represents the string “AB_CD%F” and ‘\’ is specified as the escape character. Any character not used in the string, can be chosen as the escape character.

-If the single quote symbol ‘ or apostrophe is part of the string, then they are entered as “ so it will not be interpreted as the string ending symbol.
Substring comparison implies that the attribute values are strings.

Arithmetic operators in queries:

The standard arithmetic operators +, -, and / can be applied to numeric values or attributes with numeric domains. For example: suppose we want to see the effect of giving a 10% raise to all employees working on ‘ProductX’, we can issue the query:

```
SELECT E.Fname, E.Lname, 1.1*E.Salary AS Increased_sal
FROM EMPLOYEE AS E, WORKS_ON AS W, PROJECT AS P
WHERE E.Ssn=W.Essn AND W.Po= P.Pnumber AND P.Pname= ‘ProductX’;
```

For string data types, the concatenate operator || can be used to append two string values.

For date, time and interval data types, operators include incrementing (+) decrementing (-) a date, a time or timestamp by an interval.

BETWEEN: is a comparison operator that is used to check or retrieve tuples whose attribute value is between two values:

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

(Salary BETWEEN 30000 AND 40000) is equivalent to (Salary <= 40000) AND (Salary >= 30000)

Ordering of Query Results:
ORDER BY clause allows the user to order the result of the query by the values of one or more attributes.

```
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.Po= P.Pnumber
```

ORDER BY D.Dname, E.Lname, E.Fname;

The default order is the ascending order of values. We can specify the order using either DESC for descending or ASC for ascending.

```
ORDER BY D.Dname DESC, E.Lname ASC, E.Fname ASC;
```

INSERT, DELETE and UPDATE statements in SQL:

Three commands are used to modify the database: INSERT, DELETE, and UPDATE.

The INSERT command:

```
INSERT INTO EMPLOYEE
```

A second form of the INSERT statement allows the user to specify explicit attributes names that correspond to the values provided in the INSERT command.
This is useful if a relation has many attributes, but only a few of those are assigned values in the new tuple. However, the values must include ALL the attributes with NOT NULL specification and no default value.

Attributes with NULL allowed or DEFAULT values are the ones that can be left out.

```
INSERT INTO EMPLOYEE (Fname, Lname, Dno, Ssn)
VALUES (‘Richard’, ‘Marini’, 4, ‘654321234’);
```

Attributes not specified above are set to their DEFAULT or NULL values. Also, the tuple values should be listed in the same order as described in the INSERT line.

Inserting multiple tuples:

It is possible to insert multiple tuples in a single INSERT command by separating each tuple with a comma. The attribute values representing a tuple are put between parentheses.

A DBMS that fully implements SQL should support and enforce all the integrity constraints that are specified in the DDL.

QUERY-INSERT:

A variation of the INSERT command inserts multiple tuple into a relation in conjunction with creating a relation and loading it with the results of a query.

For example, we want to create a temporary table that has the employee last name, project name and hours per week for each employee working on a project:
CREATE TABLE WORKS_ON_INFO
(Emp_Name VARCHAR(15),
Proj_Name VARCHAR(15),
Hours_per_week DECIMAL (3, 1) );

INSERT INTO WORKS_ON_INFO (Emp_Name, Proj_Name, Hours_per_week)
SELECT E.Lname, P.Pname, W.Hours
FROM PROJECT P, EMPLOYEE E, WORKS_ON W
WHERE P.Pnumber= W.Pno AND W.Essn= E.Ssn;

A table WORKS_ON_INFO is created and is loaded with the joined information retrieved from the database by the query above.

We can now query WORKS_ON_INFO as we would for any other table. When it is no longer needed, we can remove it using the DROP TABLE command.

The DELETE command:

The DELETE command removes tuples from a relation. It includes a WHERE clause to select the tuples to be deleted. Tuples are explicitly deleted from only one table at a time. However, the deletion may propagate to other tuples in other relations if referential triggered actions are specified in the referential integrity constraints of the DDL.

Depending on the number of tuples selected by the WHERE clause, 0, 1 or more tuples can be deleted by a single DELETE command.

A missing WHERE clause will cause all tuples in the table to be deleted. However, the empty table will remain in the database. The DROP TABLE command must be used to delete the table.
DELETE FROM EMPLOYEE
WHERE Lname= ‘Brown’;

DELETE FROM EMPLOYEE
WHERE Ssn= ‘123456789’;

DELETE FROM EMPLOYEE
WHERE Dno= 5;

DELETE FROM EMPLOYEE;

The UPDATE command:

The **UPDATE** command is used to modify attribute values of one or more selected tuples. The WHERE clause in the update command selects the tuples to be modified from a single relation. An additional SET clause in the UPDATE command specifies the attributes to be modified and their new values.

**UPDATE**  PROJECT

SET Plocation= ‘Bellaire’, Dnum=5

WHERE Pnumber= 10;

Several tuples can be modified with a single UPDATE command.

**UPDATE**  EMPLOYEE

SET Salary= Salary*1.1

WHERE Dno= 5;

SQL in a nutshell:

Comments: Comments start with -- and are terminated with the end of line or if there are multiple lines, we use the pairs /* .. */.
Names: Database items such as databases, tables, and columns are names using identifiers. An identifier is a sequence of letters and digits that have a maximum length of 128 characters and must begin with a letter. Names are case sensitive.

Types: A datatype must be associated with every value stored in the databases. Datatypes are associated with attributes and all values in a column must be of the same datatype.

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOLEAN</td>
<td>True or false or unknown</td>
</tr>
<tr>
<td>CHAR</td>
<td>Single character</td>
</tr>
<tr>
<td>VARCHAR(n)</td>
<td>Variable length string with maximum size = n</td>
</tr>
<tr>
<td>BLOB</td>
<td>Large binary objects, such as images</td>
</tr>
<tr>
<td>TEXT</td>
<td>Same as VARCHAR strings with a very large max size</td>
</tr>
<tr>
<td>INTEGER (INT)</td>
<td>Normal-size integer (2 bytes)</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>Small Integer (1 byte)</td>
</tr>
<tr>
<td>BIGINT</td>
<td>Large integer (4 bytes)</td>
</tr>
<tr>
<td>DECIMAL(precision, scale)</td>
<td>Decimal number: precision is the display width and scale is the number of fractional digits (2 for currency)</td>
</tr>
<tr>
<td>FLOAT</td>
<td>Floating point number</td>
</tr>
<tr>
<td>DOUBLE (REAL)</td>
<td>Double precision floating-point number</td>
</tr>
<tr>
<td>DATE</td>
<td>Date in YYYY-MM-DD format</td>
</tr>
<tr>
<td>TIME</td>
<td>Time in HH:MM:SS format</td>
</tr>
<tr>
<td>DATETIME</td>
<td>Time in YYYY-MM-DD HH:MM:SS format</td>
</tr>
</tbody>
</table>

**STRINGS:**

Strings are character sequences enclosed between pairs of single quotes. Special characters that can be used within strings:

<table>
<thead>
<tr>
<th>Special Character</th>
<th>Meaning when used in a string</th>
</tr>
</thead>
<tbody>
<tr>
<td>\</td>
<td>Single quote character</td>
</tr>
<tr>
<td>&quot;</td>
<td>Double quote character</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td><code>\b</code></td>
<td>Backspace</td>
</tr>
<tr>
<td><code>\n</code></td>
<td>New line</td>
</tr>
<tr>
<td><code>\r</code></td>
<td>Carriage return</td>
</tr>
<tr>
<td><code>\t</code></td>
<td>Tab</td>
</tr>
<tr>
<td><code>\\</code></td>
<td>Backslash</td>
</tr>
<tr>
<td><code>%</code></td>
<td>Without a preceding backslash, the percent is interpreted by the string operator LIKE as a wildcard matching 0 or more characters</td>
</tr>
<tr>
<td><code>_</code></td>
<td>Without a preceding backslash, the underscore character is interpreted by the string operator LIKE as a wildcard matching a single character.</td>
</tr>
</tbody>
</table>

Operators: Arithmetic: The usual arithmetic operators such as +, -, *, and / are available.

LOGICAL: The logical operators return the value TRUE or FALSE, they are:

- NOT (or !)
- AND (or &&)
- OR (or | |)
- XOR

COMPARISON
- = (equal)
- >= (greater than or equal)
- > (greater than)
- <= (less than or equal)
- < (less than)
- <> (not equal or !=)
- IS (NULL test)
- BETWEEN (value is in a specified range)

The IS operator can be used to test whether a value is NULL or NOT NULL as in:

```
extension IS NULL (IS NOT NULL)
```
The BETWEEN operator is used to determine whether or not a value is within a specified range (including the end points)

BETWEEN low AND high

STRING:
The string operators LIKE and NOT LIKE are used for string pattern matching:
String LIKE pattern_string
String NOT LIKE pattern_string

PRECEDENCE:
The precedence of the commonly used operators is:

<table>
<thead>
<tr>
<th>Operators</th>
</tr>
</thead>
<tbody>
<tr>
<td>OR, XOR</td>
</tr>
<tr>
<td>AND</td>
</tr>
<tr>
<td>BETWEEN</td>
</tr>
<tr>
<td>=, &gt;=, &gt;, &lt;=, &lt;, &lt;&gt;, !=, IS, LIKE, NOT LIKE</td>
</tr>
<tr>
<td>-</td>
</tr>
<tr>
<td>+</td>
</tr>
<tr>
<td>*</td>
</tr>
<tr>
<td>/</td>
</tr>
<tr>
<td>-(unary minus)</td>
</tr>
</tbody>
</table>

FUNCTIONS:
- CEIL (rounds up to the nearest integer), and
- SIN (sine function, argument in radians)