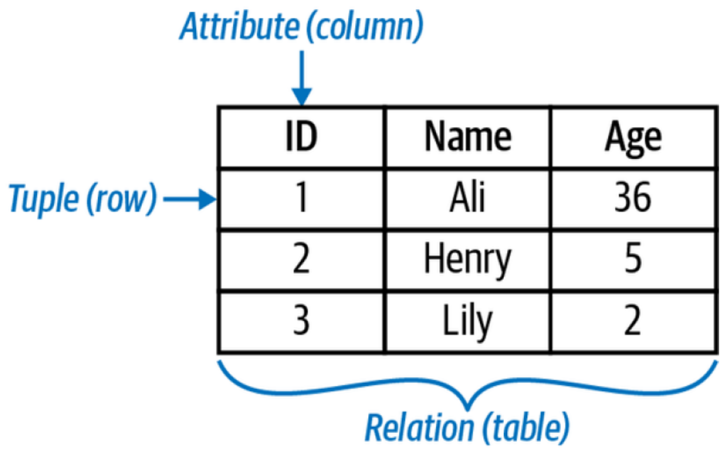
SQL Notes

SQL is a structured query language that involves using relational databases, which requires pre-defined schemas.



ID	Name	Age
1	Ali	36
2	Henry	5
3	Lily	2

CRUD Operations are most used in SQL. (Create, Read, Update, Delete)

SQL statements are divided into 4 main categories:

1. Data Definition Language (DDL)
   - CREATE (Create new database/table)
   - ALTER (Modify database/table structure)
   - DROP (Delete database or table)
   - TRUNCATE (Remove table records)

2. Data Manipulation Language (DML)
   - SELECT (Retrieve data from table)
   - INSERT (Insert new records into table)
   - UPDATE (Update existing records from a table)
   - DELETE (Remove existing records from a table)

3. Data Control Language (DCL)
   - GRANT (Assign privilege to users for accessing data)
   - REVOKE (Remove privilege to users for accessing data)

4. Transaction Control Statement (TCS)
   - COMMIT (Permanent work save into database)
   - ROLLBACK (Restore database to previous form since last commit)
   - SAVEPOINT (Create save point for future rollbacks)
   - SET TRANSACTION (Setting transaction to read-write/read only access)

SQL Query Basic Template

SELECT <column_names>

FROM <table_name>

WHERE <condition on columns from table>

GROUP BY <column_names>

HAVING <condition on grouped-by columns>

ORDER BY <column_names>

SQL works in the following order of statement execution:

1. Gathers all data with FROM clause
2. Filters the data with WHERE clause
3. Groups rows together with GROUP BY clause
4. Filters grouped rows with HAVING clause
5. Specifies columns to display with SELECT clause
6. Sorts the results with ORDER BY clause

Types of RDBMS and its database tools:

1. SQLite: DB Browser for SQLite
2. MySQL: MySQL Workbench
3. Oracle: Oracle SQL Developer (PL/SQL – Procedural Language Extension to SQL)
4. PostgreSQL: pgAdmin
5. SQL Server: SQL Server Management Studio (T-SQL – Transact SQL)

ANSI (American National Standards Institute) standard in SQL refers to SQL code that will run in any RDBMS software.

Identifiers vs Aliases

Identifiers are name of database objects

Aliases rename column or table temporarily, mostly useful in subqueries.

Statements vs Clauses

Statements are blocks of code that starts with a SQL keyword like SELECT and ends with a semicolon.

Clauses are specific sections of the statement that refers to specific SQL keywords like WHERE, FROM etc.

Single vs Multi-Line Comments

Single Line Comment:

-- This is a single line comment

Multi Line Comment:

/* These are

multi line comments */

Single vs Double Quotes

Single quote: Used for string reference

Double quote: Used for identifier reference

Wildcard expressions

%: Represents any n number of characters

_: Represents any single character

Note that these wildcard expressions are used together with LIKE or NOT LIKE keyword in WHERE clause.

SQL Data Types

1. Numeric (INT, DECIMAL, FLOAT)
   - INT: Used for values that do not allow for decimals (i.e. 45)
   - DECIMAL: Used for fixing number of decimals (i.e. 24.524)
   - FLOAT: Store limited number of decimals with power notation (i.e. 2.4524 * 10^5)

   Note: MySQL has the option of setting numeric variables as positive only using UNSIGNED keyword.

2. String (CHAR, VARCHAR, TEXT, NCHAR, NVARCHAR)
   - CHAR: Stores fixed length of characters as ASCII data
   - VARCHAR: Stores maximum length of specified characters as ASCII data
   - TEXT: Used for storing long strings of text like paragraphs
   - NCHAR: Stores fixed length of characters as Unicode data
   - NVARCHAR: Stores maximum length of specified characters as Unicode data

   Note: Unicode data refers to non-ASCII characters (non-English mostly)

3. Date (DATE, TIME, DATETIME, TIMESTAMP, YEAR – For MYSQL)
   - DATE: YYYY-MM-DD
   - DATETIME: YYYY-MM-DD hh:mm:ss
   - TIMESTAMP: YYYY-MM-DD hh:mm:ss UTC
   - TIME: hh:mm:ss
   - YEAR: YYYY

   Note: DATETIME variable type does not store time zone, while TIMESTAMP variable type does store time zone.

4. Boolean (BOOLEAN): FALSE value as 0 and TRUE value as 1
   Note: Boolean variable type is currently not supported in Oracle and SQL server.

5. External files like images, documents etc.
   Approach 1: Store links to files using VARCHAR variable type
   Approach 2: Convert files to binary format and store files using BLOB variable type

Common SQL Operators

Logical Operators

AND: Returns TRUE if both conditions are true or otherwise

OR: Returns TRUE if either condition is true or otherwise

NOT: Returns TRUE if condition is FALSE or otherwise

Comparison Operators (Used in conditional statements)

=	Equality	<=	Less than or equal to
!=, <>	Inequality	>	Greater than
<	Less than	>=	Greater than or equal to

Comparison Keywords (Used in conditional statements)

BETWEEN: Checks if value lies within given range (inclusive)

EXISTS: Checks if row exist in subquery (also known as semi-join)

IN: Checks if value is contained within a list using brackets symbol '( )' (Note that having a single null value within the list will always result in FALSE for conditional statements. Thus, recommend using **NOT EXISTS** keyword instead.)

IS NULL: Checks if a value is null

IS NOT NULL: Checks if a value is not null

LIKE: Checks if value matches a simple pattern (Used with wildcard expressions)

ALL: Returns TRUE if all subquery values meet specified condition (Requires comparison operator)

ANY: Returns TRUE if any subquery values meet specified condition (Requires comparison operator)

Note that "**= ANY**" can be represented by IN keyword and "**!= ANY"** can be represented by NOT IN keyword.

Math Operators

+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulo (Remainder)

Bitwise Operators

&: Bitwise AND

|: Bitwise OR

^: Bitwise XOR

<u>Numeric Functions</u>

Numeric functions are usually applied to attributes with numerical values only.

ABS(x)	Returns absolute value of x	**LOG10(x)**	Returns log of x base 10
SIGN(x)	Returns sign of value of x (-1 for negative, 0 for zero or 1 for positive)	**MOD(x,y)**	Returns remainder of x/y
POWER(x,y)	Returns x to the power of y	**RAND()**	Returns a random number between 0 and 1
SQRT(x)	Returns square root of x	**CEIL(x)**	Returns upper integer boundary of value x
EXP(x)	Returns exponent of x	**FLOOR(x)**	Returns lower integer boundary of value x
LOG(y,x)	Returns log of y base x	**ROUND(x,n)**	Rounds x value to n decimal places
LN(x)	Return natural log of x base e	**TRUNC(x,n)**	Cuts off x value at n decimal places without rounding

<u>String Functions</u>

String functions are usually applied to attributes with string data types.

LENGTH(string): Returns number of characters of a string

UPPER(string): Returns upper case characters of the whole string

LOWER(string): Returns lower case characters of the whole string

TRIM(string): Removes trailing and leading whitespaces from string

TRIM(char FROM string): Removes trailing and leading specified char from string

LTRIM(string): Removes leading whitespaces from string

RTRIM(string): Removes trailing whitespaces from string

CONCAT(string1, string2, …): Concatenate multiple strings into a single string

SUBSTRING(string, start, length): Returns a portion of string starting from start index for n length.

REPLACE(string, old_substring, new_substring): Replace old_substring from a given string with new_substring.

REGEXP 'Regular_Expression': Search for regular expression pattern

Finding index location of string syntax differs based on the software used.

INSTR(string, substring) : MySQL

CHARINDEX(substring, string, position) : SQL Server

POSITION(substring IN string) : PostgreSQL

Note: Regular expression syntax is best tested with the following website, instead of memorizing syntax: https://regex101.com/

Regular Expressions in SQL

*	Zero or more instances of previous element	[…]	Any character within brackets
+	One or more instances of previous element	[^…]	Any character not within brackets
.	Any single character	{n}	N instance of previous element
?	Zero or one instance of previous element	{n,}	At least N instance of previous element
^	Beginning of string	{n,p}	N to p instance of previous element
$	End of string	p1 | p2	Matches any patterns p1, p2, …

These quantifiers are usually used together with REGEXP keyword.

<u>Datetime Format Specifiers</u>

%Y	4-digit year	%d	Day (1-31)
%y	2-digit year	%h	12 hours (1 – 12)
%m	Numeric month (1-12)	%H	24 hours (1 – 24)
%b	Abbreviated month (Jan – Dec)	%i	Minutes (0-59)
%M	Name of month (January – December)	%s	Seconds (0-59)

Note that datetime functions for different SQL engines have different syntax used. Thus, it is recommended to google it rather than memorizing syntax.

<u>Limiting view of number of rows</u>

Different syntax is used for this task depending on SQL software used.

For MySQL, PostgreSQL (Note that LIMIT clause is executed last after ORDER BY clause):

SELECT cl1, cl2

FROM owner

LIMIT n

For SQL Server:

SELECT TOP n cl1, cl2

FROM owner

For Oracle:

SELECT cl1, cl2

FROM owner

WHERE ROWNUM <= n

Functions for summarizing rows into lists

Using together with GROUP BY clause, the following functions can be used for summarizing multiple rows into list of values:

MYSQL	**GROUP_CONCAT (DISTINCT x ORDER BY x SEPARATOR ‘,’)**
SQL Server	**STRING_AGG(x, ‘,’) WITHIN GROUP (ORDER BY x)**
PostgreSQL	**ARRAY_AGG(DISTINCT x ORDER BY x)**

Note the following caveats:

1. SQL server currently does not support “Unique” list
2. PostgreSQL only accepts ‘,’ as default separator
3. SQL server requires input of separator, while input of separator is optional for MYSQL

Complex Group-By Functions

Additional summary information can also be included in queries using the following complex group-by functions:

1. GROUP BY ROLLUP (a,b, ….,e)
   - Group by sets of a – e, a – d, …. a and ()

2. GROUP BY CUBE (a,b, …., e)
   - Group by all combination of sets and ()

3. GROUP BY GROUPING SETS(a,b,…,())
   - Group by individual categories and (), which represents group function on entire table

Note that these functions are mostly supported in Oracle, PostgreSQL and SQL server.

MySQL only supports rollup function using the following syntax:

GROUP BY a, b, …., e WITH ROLLUP

Data model/Schema

Data model/Schema describes how database objects are organized within a database

The following actions can be done with respect to databases:

1. Creating new database
   CREATE DATABASE db_name

   CREATE DATABASE IF NOT EXISTS db_name (MySQL)

2. Delete existing database
   DROP DATABASE db_name

3. Using database (MySQL and SQL Server)
   USE db_name

4. Display names of all existing databases
   SHOW databases (MySQL)

   SELECT name FROM master.sys.databases (SQL Server)

5. Display name of current database
   SELECT database() (MySQL)

   SELECT db_name() (SQL Server)

Creating/Modifying Tables

The following actions can be done with respect to tables:

1. Creating new tables
   ```
   CREATE TABLE IF NOT EXISTS tb_name(
           var1 dtype1,
           var2 dtype2,
           …
   )
   ```

2. Remove whole data inside table
   TRUNCATE TABLE tb_name

3. Remove whole table
   DROP TABLE IF EXISTS tb_name

4. Insert records into tables
   INSERT INTO tb_name (cl1, cl2, …)
   VALUES (cl1, cl2, …), (cl1, cl2, …)

5. Delete records from tables
   DELETE FROM tb_name
   WHERE condition

6. Update records from tables
   UPDATE tb_name
   SET cl1 = value1, …
   WHERE condition

7. Display names of existing tables
   SHOW tables (MySQL)

   SELECT tb_name FROM information_schema.tables (SQL Server)

8. Display columns of existing tables
   DESCRIBE tb_name (MySQL)

   SELECT cl1, cl2, …
   FROM information_schema.columns WHERE table_name = tb_name (SQL Server)

9. Adding new columns
   ALTER TABLE tb_name
   ADD cl_name dtype

10. Removing columns
    ALTER TABLE tb_name
    DROP COLUMN cl_name

11. Change data type of columns
    ALTER TABLE tb_name
    ALTER COLUMN cl_name dtype

SQL Constraints

Constraints are rules that specify what data can be inserted into a table.

Types of SQL Constraints:

1. NOT NULL: Restrict columns to not allow null values
2. DEFAULT: Setting default values that replaces missing values
3. CHECK: Filters inputs based on certain conditions
4. UNIQUE: Restrict columns to only allow unique values
5. PRIMARY KEY: Candidate key that uniquely identifies a row
6. FOREIGN KEY: Candidate key within one table that matches with candidate key in other tables.

Note that primary key must be created first, before creating foreign keys.

Example:

```
CREATE TABLE table1(
        id INTEGER PRIMARY KEY
)
CREATE TABLE table2(
        id INTEGER PRIMARY KEY,
        FOREIGN KEY id1 REFERENCES table1(id)
)
```

For creating table with auto-generated fields, following syntax is used:

cl_name dtype PRIMARY KEY AUTO_INCREMENT (MySQL)

cl_name dtype IDENTITY(1,1) (SQL Server)

The following actions can be done for constraints:

1. Create new constraints when creating tables
   CONSTRAINT ct_name ……

2. Create new constraints when altering tables
   ALTER TABLE tb_name
   ADD CONSTRAINT ct_name …….

3. Modify existing constraints when altering tables
   ALTER TABLE tb_name
   ALTER COLUMN cl_name dtype new_ct

4. Remove existing constraint
   ALTER TABLE tb_name
   DROP CONSTRAINT ct_name

5. Display table constraints
   For MySQL:
   SHOW CREATE TABLE tb_name

   For non-default constraints in SQL Server:
   SELECT table_name, constraint_name, constraint_type
   FROM information_schema.table_constraints
   WHERE table_name = 'tb_name'

   For default constraints in SQL Server:
   SELECT OBJECT_NAME(parent_object_id),
   COL_NAME(parent_object_id, parent_column_id), definition
   FROM sys.default_constraints
   WHERE OBJECT_NAME(parent_object_id) = 'tb_name'

Indexes

Indexes are useful for retrieving data from very large datasets more quickly that speeds up SELECT and WHERE queries.

However, index is not useful under following scenarios:

1. Columns with frequent, large batch updates on insert operations
2. Columns with large number of null values
3. Tables with small number of records
4. Columns that are frequently manipulated

The following actions can be done for indexes:

1. Creating new index
   CREATE INDEX index_name
   ON tb_name (cl1, cl2, …)

2. Remove existing index
   DROP INDEX index_name ON tb_name

Views

Views can be created for the following benefits:

1. Restrict users from accessing sensitive information on the original database
2. Hide and reuse complex queries
3. Provide more meaningful aliases to variables in the original database

There are two types of views in SQL:

1. Inline views: Temporary results from subqueries used in FROM clause
2. Views: Referenced virtual tables from result of query that does not take up additional storage space and can be reused

The following actions can be done for views:

1. Create new views/replace existing views
   CREATE VIEW vw_name AS
   Query

2. Remove existing views
   DROP VIEW vw_name

3. Display existing views
   For MySQL:
   SHOW FULL TABLES WHERE table_type = ‘VIEW’
   For SQL Server:
   SELECT table_name FROM information_schema.views

Note that data cannot be inserted or removed from views once created.

<u>Transactions</u>

Transaction allows for safer updates on database through a sequence of SQL operations as a single unit.

Using the concept of atomicity, either all SQL operations are executed or not under a transaction.

Transaction Keywords in SQL:

1. **START TRANSACTION / BEGIN TRANSACTION**: Used to initiate a new transaction (Start transaction keyword used for MySQL and PostgreSQL, while Begin transaction keyword used for SQL Server)

2. **SAVEPOINT point_name**: Used to save state of transaction at specific points of the query code in which to rollback.

3. **RELEASE SAVEPOINT point_name**: Used to remove state of transaction saved.

4. **ROLLBACK TO point_name**: Rollback changes to before transaction occurs or back to a previous savepoint.

5. **COMMIT**: Make permanent changes to the database

Note that once a transaction is committed, it is no longer reversible.