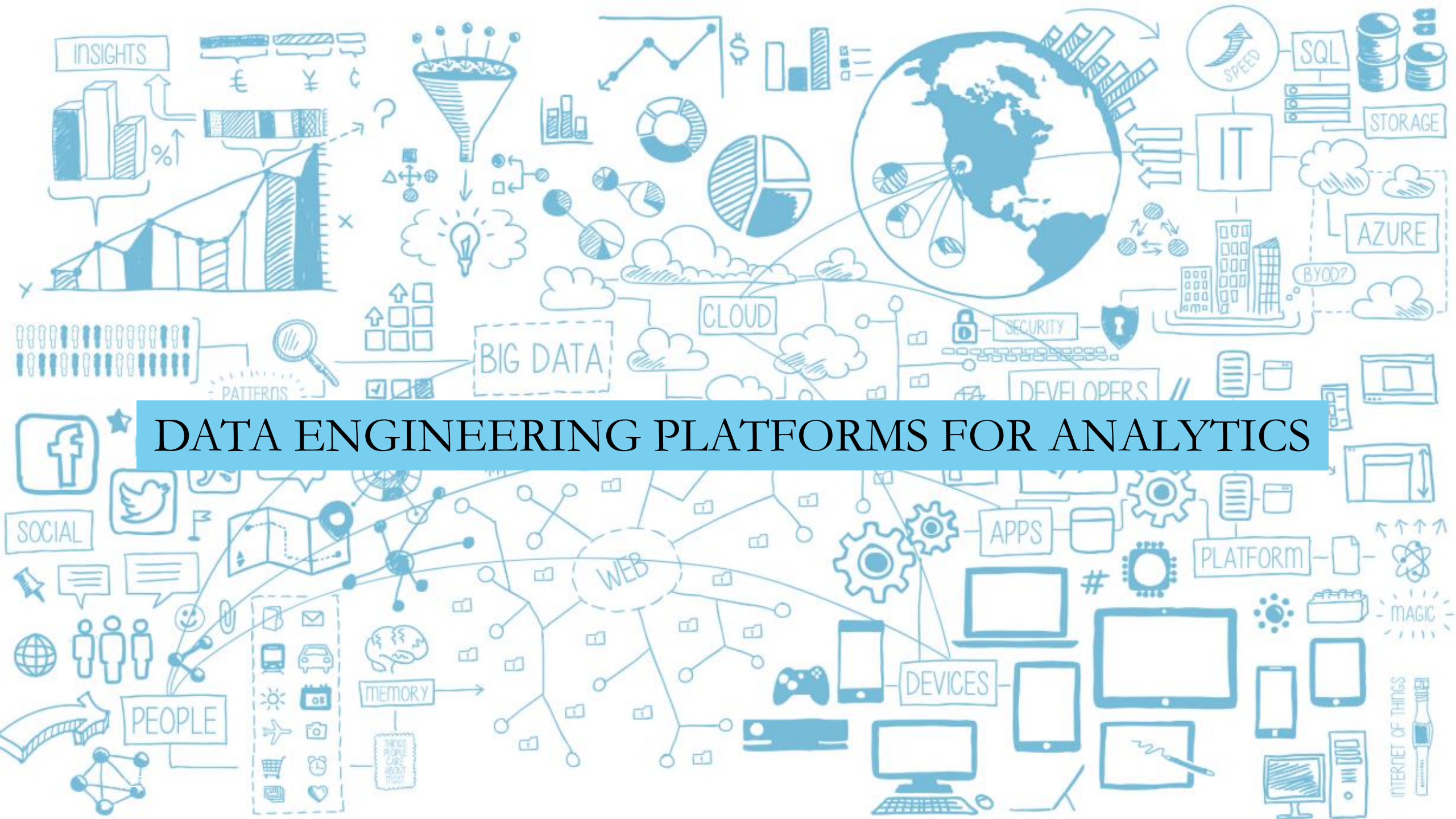


DATA ENGINEERING PLATFORMS FOR ANALYTICS

Syllabus

- Session 1 – Foundations of data systems
- Session 2 – Relational Databases
- Session 3 – Structured Query Language(SQL)
- Session 4 – Advanced SQL
- Session 5 – Analytical data platforms

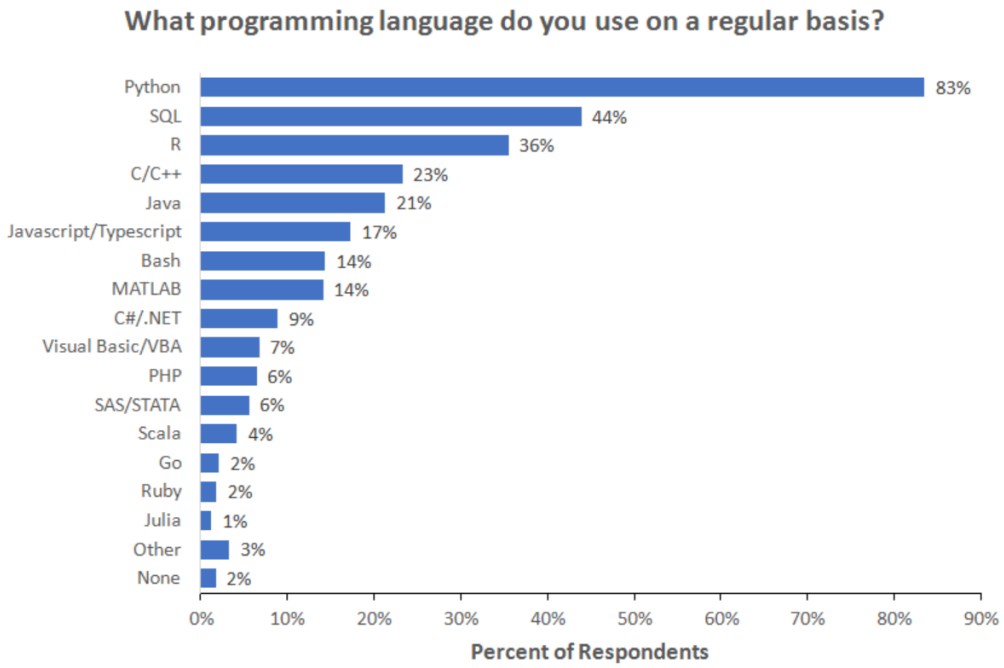
Syllabus

- Session 6 – Business Intelligence
- Session 7 – Data Pipelines (GCP)
- Session 8 – Document & Graph databases
- Session 9 – Columnar, Key Value DB & Blockchain
- Session 10 – Final Exam & Project Presentations

SQL Introduction

Structured Query Language (SQL)

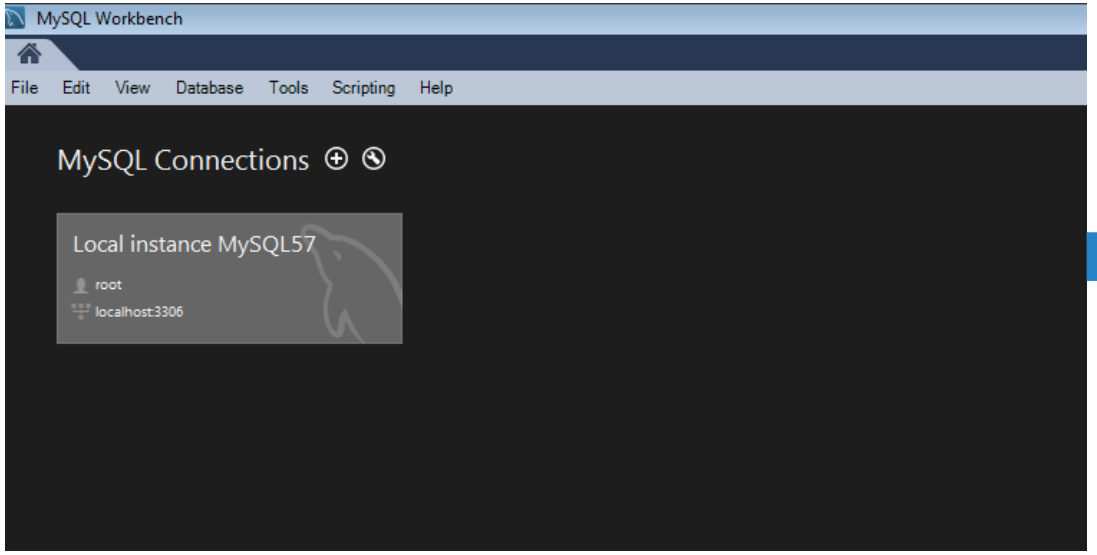
- The ANSI standard language for the definition and manipulation of relational database.
- First Version was developed at IBM by Donald D. Chamberlin and Raymond F. Boyce. [SQL]
- Developed using Dr. E.F. Codd's paper, "A Relational Model of Data for Large Shared Data Banks."



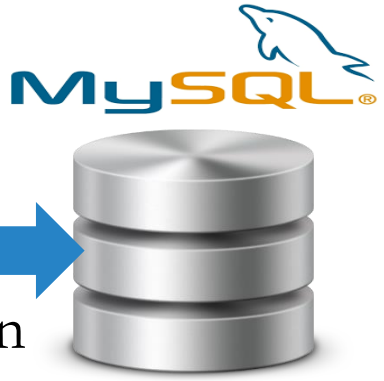
Note: Data are from the 2018 Kaggle Machine Learning and Data Science Survey. You can learn more about the study here: http://www.kaggle.com/kaggle/kaggle-survey-2018. A total of 18827 respondents answered the question.

BUSINESS BROADWAY
DATA SCIENCE | CUSTOMER ANALYTICS | MACHINE LEARNING

SQL Session



SQL Connection



MySQL Workbench

MySQL Database

String Data Types

String Types	Description
CHAR	A fixed-length non-binary (character) string
VARCHAR	A variable-length non-binary string
BINARY	A fixed-length binary string
VARBINARY	A variable-length binary string
TINYBLOB	A very small BLOB (binary large object)
BLOB	A small BLOB
MEDIUMBLOB	A medium-sized BLOB
LONGBLOB	A large BLOB
TINYTEXT	A very small non-binary string

String Data Types - Examples

CHAR(2)	“IL”, “CA”, “MA”,..
VARCHAR(50)	“Illinois”, “California”, “Massachusetts”,..
BLOB	11010010101010101001010101010

Numeric Data - Integer Types

Numeric Types	Description	Storage (Bytes)	Min – Max values signed, unsigned
TINYINT	A very small integer	1	[-128 to 127], [0 to 255]
SMALLINT	A small integer	2	[-32768 to -32767], [0 to 65535]
MEDIUMINT	A medium-sized integer	3	[-8388608 to 8388607], [0 to 16777215]
INT	A standard integer	4	[-2147483648 to 2147483647], [0 to 4294967295]
BIGINT	A large integer	8	[-9223372036854775808 to 9223372036854775807], [0 to 18446744073709551615]

Numeric Data – Fixed and Floating Point

Numeric Types	Description	Type
DECIMAL (M,D) or NUMERIC (M,D)	A fixed-point number	Exact
FLOAT (M,D)	A single-precision floating point number	Approximate
DOUBLE (M,D)	A double-precision floating point number	Approximate

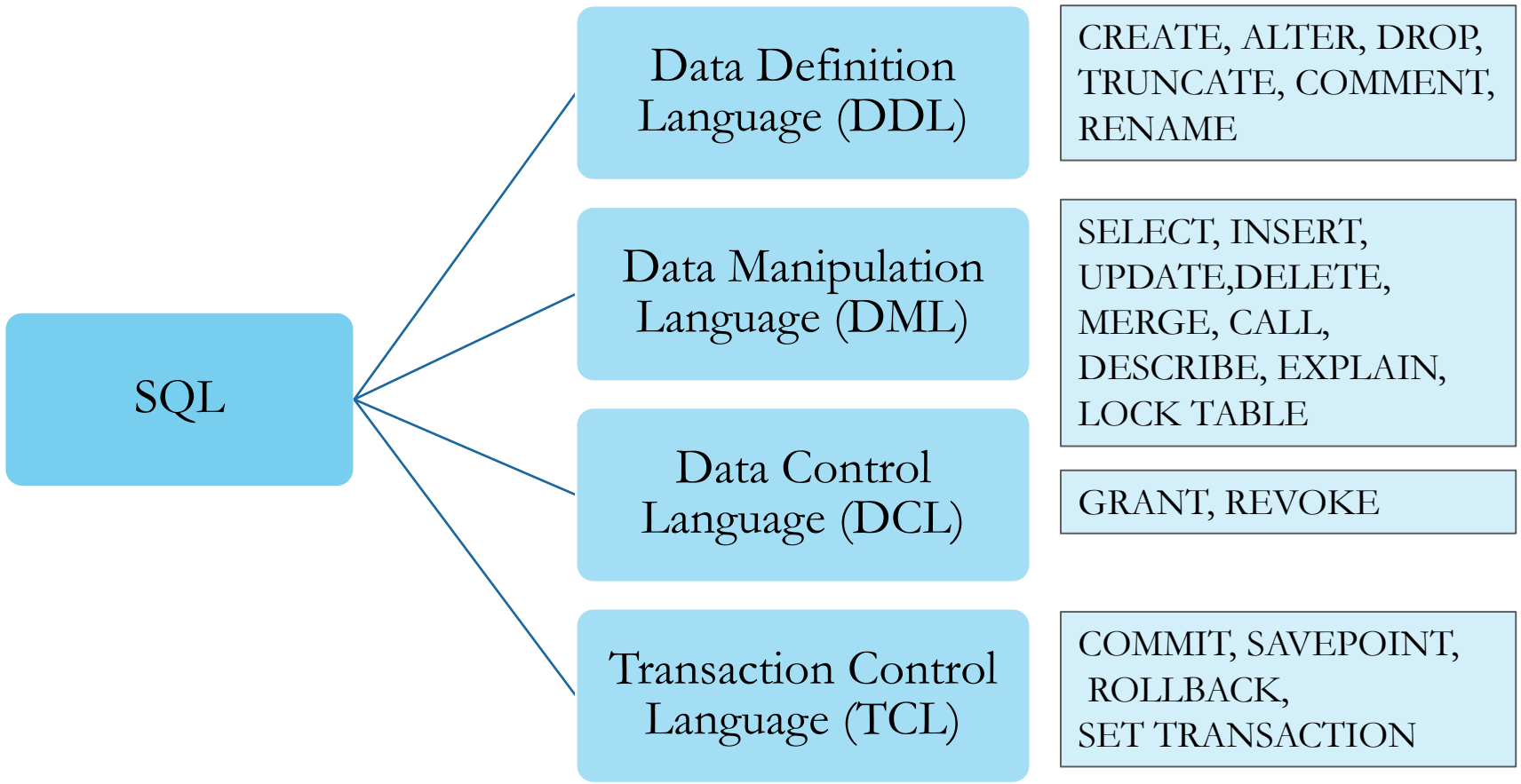
(M,D) means than values can be stored with up to M digits in total, of which D digits may be after the decimal point.

e.g. DECIMAL(5,2) -999.99 to 999.99.

Date and Time Data Types

Date Types	Description
DATE	A date value in 'CCYY-MM-DD' format
TIME	A time value in 'hh:mm:ss' format
DATETIME	A date and time value in 'CCYY-MM-DD hh:mm:ss' format
TIMESTAMP	A timestamp value in 'CCYY-MM-DD hh:mm:ss' format
YEAR	A year value in CCYY or YY format

SQL Command Types



DDL - Data Definition Language

- CREATE - to create objects in the database
- ALTER - alters the structure of the database
- DROP - delete objects from the database
- TRUNCATE - remove all records from a table (including spaces)
- COMMENT - add comments to the data dictionary
- RENAME - rename an object

DML - Data Manipulation Language

- SELECT - retrieve data from the a database
- INSERT - insert data into a table
- UPDATE - updates existing data within a table
- DELETE - deletes all records from a table (spaces remain)
- DESCRIBE – obtain information about table structure
- EXPLAIN - explain access path to data (query execution plans)

CRUD Operations

In SQL database map directly to DML statements

- Create (INSERT)
- Read (SELECT)
- Update (UPDATE)
- Delete (DELETE)

DCL – Data Control Statements

- GRANT - gives user's access privileges to database
- REVOKE - withdraw access privileges given with GRANT

```
GRANT SELECT, INSERT, UPDATE, DELETE ON table_name TO user_name;

REVOKE INSERT ON table_name TO user_name;
```

TCL - Transaction Control

Allows statements to be grouped together into logical transactions

- COMMIT - save work done
- SAVEPOINT - identify a point in a transaction to which you can later roll back
- ROLLBACK - restore database to original since the last commit
- SET TRANSACTION - change transaction options like isolation level and what rollback segment to use

Manipulating Data

Manipulating Data

1. Create Database
2. Create tables
3. Insert data
4. Update data
5. Delete the database

Exercise 1

SQL Introduction (DDL/DML)

Categorizing Data

Categorizing Data

1. SELECT and WHERE clauses
2. Arithmetic, Logical and Comparison Operators
3. Operators used singularly and in combinations
4. Conditional Statements
5. Basic use of DISTINCT and LIMIT

Where Clause

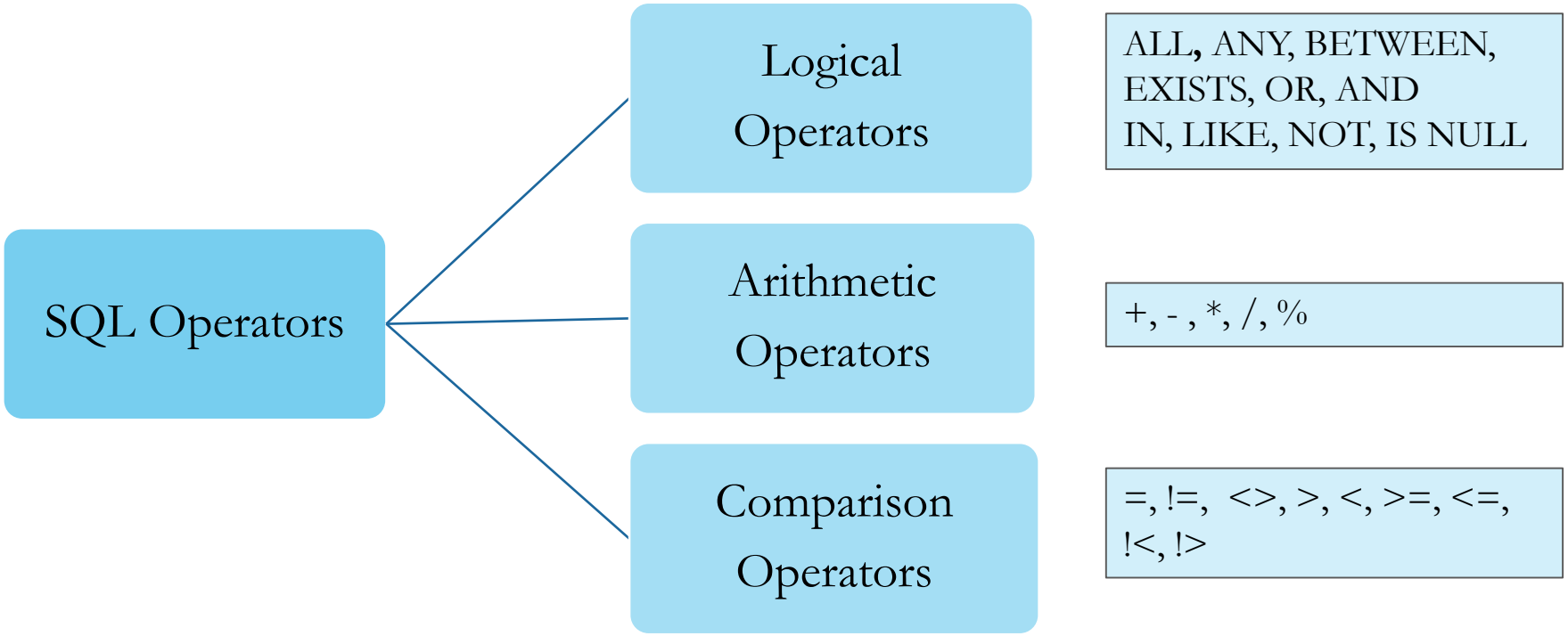
- The WHERE clause is used to filter records
- All rows for which the predicate in the WHERE clause is True are returned by the SQL DML statement or query criterion

```
SELECT column_name,column_name
FROM table_name
WHERE column_name operator value;
```

SQL Operator

- An operator performs on separate data items and returns a result. The data items are called operands or arguments
- Reserved word or character used primarily in WHERE clause to perform comparisons and arithmetic operations

Operator Types



Logical Operators

LOGICAL OPERATORS	DESCRIPTION
ALL	Compare a value to all values in another value set.
ANY	Compare a value to any applicable value in the list according to the condition.
BETWEEN	Search for values that are within a set of values, given the minimum value and the maximum value.
EXISTS	Search for the presence of a row in a specified table that meets certain criteria.
OR	At least one of the conditions must be true
AND	All the specified conditions must be true
IN	Compare a value to a list of literal values that have been specified.
LIKE	Compare a value to similar values using wildcard operators.
NOT	Reverses the meaning of the logical operator. This is a negate operator.
IS NULL	The NULL operator is used to compare a value with a NULL value.

Arithmetic Operators

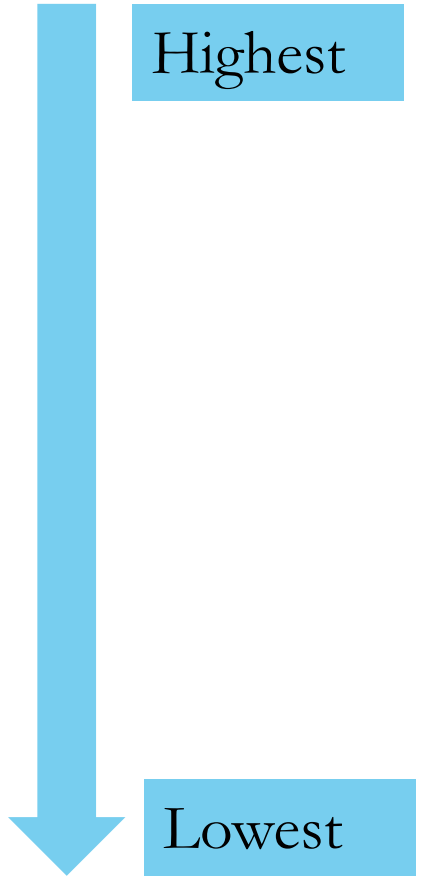
ARITHMETIC OPERATORS	DESCRIPTION
+ (ADD)	Addition
- (SUBTRACT)	Subtraction
* (Multiply)	Multiplication
/ (Divide)	Division
% (Modulo)	Integer remainder of a division

Comparison Operators

COMPARISON OPERATORS	DESCRIPTION
=	Checks if the values of two operands are equal or not, if yes then condition becomes true.
!=	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.
<>	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.
!<	Checks if the value of left operand is not less than the value of right operand, if yes then condition becomes true.
!>	Checks if the value of left operand is not greater than the value of right operand, if yes then condition becomes true.

Operator Precedence

```
INTERVAL
BINARY, COLLATE
!
- (unary minus), ~ (unary bit inversion)
^
*, /, DIV, %, MOD
-, +
<<, >>
&
|
= (comparison), <=>, >=, >, <=, <, <>, !=, IS, LIKE, REGEXP, IN
BETWEEN, CASE, WHEN, THEN, ELSE
NOT
AND, &&
XOR
OR, ||
= (assignment), :=
```



Select Case

- SELECT CASE is used for equality tests of one expression against multiple values

```
CASE expression
   WHEN test THEN result
   …
   ELSE otherResult
END;
```

Select Distinct

- SELECT DISTINCT statement is used to return only distinct (different) values

```
SELECT DISTINCT column_name,column_name
FROM table_name;
```

Limit

- LIMIT clause is used to specify the number of records to return
- Useful on large tables with thousands of records; avoids performance issues

```
SELECT column_name(s)
FROM table_name
LIMIT number;
```

Exercise 2

Categorizing Data

Summarizing Data

Summarizing Data

1. COUNT(*) and COUNT(column) functions
2. COUNT with DISTINCT and WHERE clause
3. Aggregate functions - SUM, MIN, MAX, AVG
4. Aggregate functions - Statistical
5. Aggregate functions with conditional statements

Aggregate Function

- Performs a calculation on a set of numerical values and returns a single value
- Ignores NULL values when it performs calculation
- Required by GROUP BY clause

```
SELECT AGGREGATE FUNCTION (column_name)
FROM table_name;
```

Count (expr) Function

- Returns a count of the number of non-NULL values of expression in the rows retrieved by a SELECT statement
- The result is a BIGINT value

```
SELECT COUNT(*)
FROM table_name;
```

Count (*) Function

- Returns a count of the number of rows retrieved, whether or not they contain NULL values
- Sometimes optimized when SELECT retrieves from one table, and there is no WHERE clause

```
SELECT COUNT (*)
FROM table_name;
```

SQL Aggregate functions

- An aggregate functions are keywords in SQL used to provide summarization information for an SQL statement, such as counts, totals, and averages normally used in conjunction with a column name or expression that processes the incoming data to produce a result.

Aggregate Functions

AGGREGATE FUNCTIONS	DESCRIPTION
AVG()	Return the average value of the argument
COUNT()	Return a count of the number of rows returned
COUNT(DISTINCT)	Return the count of a number of different values
MAX()	Return the maximum value
MIN()	Return the minimum value
STD()	Return the population standard deviation
STDDEV()	Return the population standard deviation
STDDEV_POP()	Return the population standard deviation
STDDEV_SAMP()	Return the sample standard deviation
SUM()	Return the sum
VARIANCE()	Return the population standard variance

Exercise 3

Summarizing Data

Sorting & Grouping Data

Order By Clause

- Sort data in ascending or descending order based on one or more columns
- Default sort is ascending

```
SELECT column_name(s)
FROM table_name
WHERE column_name operator value
ORDER BY column_name, column_name;
```

Group By Clause

- Used in collaboration with the SELECT statement to arrange identical data into groups
- Selected columns must appear in the GROUP BY clause
- Follows the WHERE clause and precedes the ORDER BY clause

```
SELECT column_name(s)
FROM table_name
WHERE column_name operator value
GROUP BY column_name, column_name
ORDER BY column_name, column_name;
```

Order By Versus Group By

- All non-aggregate columns selected must be listed in the GROUP BY clause
- GROUP BY clause generally not necessary unless using aggregate functions

```
SELECT column_name(s)
FROM table_name
WHERE column_name operator value
GROUP BY column_name, column_name
HAVING conditions
ORDER BY column_name, column_name;
```

Having Clause

- When used in conjunction with the GROUP BY clause in a SELECT statement, tells which groups to include in the output
- Must follow the GROUP BY clause and precede ORDER BY

```
SELECT column_name(s)
FROM table_name
WHERE column_name operator value
GROUP BY column_name, column_name
HAVING conditions
ORDER BY column_name, column_name;
```

Sorting & Grouping Data

- Sorting data using ORDER BY clause
- Aggregating and grouping data using GROUP BY clause
- Sorting and Grouping with conditional statements
- Listing top results using LIMIT

Exercise 4

Sorting & Grouping Data

Appendix

References

- http://www.mysqltutorial.org/
- http://www.techonthenet.com/mysql/
- http://stackoverflow.com/
- http://www.tutorialspoint.com/

Exercise 1 : Solutions

Exercise 2 : Solutions

Exercise 3 : Solutions

Exercise 4 : Solutions