

# CSC 436 Assignment 3 Documentation

## Programming Language Lookup Database

### Section 1: Data Integrity Constraints

The database enforces data accuracy through the following constraints:

#### Primary Key Constraints

Every table has a primary key with AUTO\_INCREMENT to automatically assign unique values. The primary keys of the strong entity tables are their own ID fields, and the child implementation tables' primary and foreign keys are implementation\_id.

#### NOT NULL Constraints

NOT NULL constraints ensure that all essential fields are not null: language\_name, function\_name, structure\_name, operator\_name, and all foreign keys. This ensures that no partial data is inserted.

#### UNIQUE Constraints

The language\_name field has a UNIQUE constraint to ensure that there are no duplicate programming language names.

#### CHECK Constraints

CHECK constraints are not entirely supported by the MySQL environment on cPanel, so validation of data is done during insertion.

#### Foreign Key Constraints

Foreign keys establish relationships between tables. The implementation table has a foreign key to the language table via language\_id. The other three child tables have foreign keys to both the implementation table and their respective entity tables. All foreign keys have ON DELETE CASCADE and ON UPDATE CASCADE actions to ensure data integrity.

#### Validation During Data Insertion

Before inserting data, all mandatory fields were checked to ensure they contained values. Language names were checked for duplication. Parent records were inserted before child records. Foreign key values were checked to ensure they existed in parent tables.

### Constraint Summary

Table	Primary Key	NOT NULL	Foreign Keys
language	language_id	language_name	None
function_table	function_id	function_name	None
data_structure	structure_id	structure_name	None
operator	operator_id	operator_name	None
implementation	implementation_id	language_id	language_id
function_implementati on	implementation_id	function_id	implementation_id, function_id
structure_implementat ion	implementation_id	structure_id	implementation_id, structure_id
operator_implementati on	implementation_id	operator_id	implementation_id, operator_id

## Section 2: Table Screenshots

Insert screenshots from phpMyAdmin Browse tab for: language, function\_table, data\_structure, operator, implementation, functionImplementation, structureImplementation, operatorImplementation

### Data\_structure

	← T →	▼	structure_id	structure_name	category	description
<input type="checkbox"/>	 Edit	 Copy	 Delete	1 Array	Linear	Fixed-size sequential collection
<input type="checkbox"/>	 Edit	 Copy	 Delete	2 List	Linear	Dynamic sequential collection
<input type="checkbox"/>	 Edit	 Copy	 Delete	3 Dictionary	Associative	Key-value pair collection
<input type="checkbox"/>	 Edit	 Copy	 Delete	4 Set	Collection	Unordered unique elements
<input type="checkbox"/>	 Edit	 Copy	 Delete	5 Stack	Linear	LIFO data structure
<input type="checkbox"/>	 Edit	 Copy	 Delete	6 Queue	Linear	FIFO data structure
<input type="checkbox"/>	 Edit	 Copy	 Delete	7 Linked List	Linear	Pointer-based sequential collection
<input type="checkbox"/>	 Edit	 Copy	 Delete	8 Tree	Hierarchical	Hierarchical node structure
<input type="checkbox"/>	 Edit	 Copy	 Delete	9 Graph	Non-Linear	Nodes connected by edges
<input type="checkbox"/>	 Edit	 Copy	 Delete	10 Hash Table	Associative	Hash function indexed structure

### FunctionImplementation

	 Edit	 Copy	 Delete	implementation_id	function_id
<input type="checkbox"/>	 Edit	 Copy	 Delete	1	1
<input type="checkbox"/>	 Edit	 Copy	 Delete	2	1
<input type="checkbox"/>	 Edit	 Copy	 Delete	3	1
<input type="checkbox"/>	 Edit	 Copy	 Delete	4	1
<input type="checkbox"/>	 Edit	 Copy	 Delete	5	2
<input type="checkbox"/>	 Edit	 Copy	 Delete	6	2
<input type="checkbox"/>	 Edit	 Copy	 Delete	7	2
<input type="checkbox"/>	 Edit	 Copy	 Delete	8	3
<input type="checkbox"/>	 Edit	 Copy	 Delete	9	3
<input type="checkbox"/>	 Edit	 Copy	 Delete	10	3
<input type="checkbox"/>	 Edit	 Copy	 Delete	11	4
<input type="checkbox"/>	 Edit	 Copy	 Delete	12	4
<input type="checkbox"/>	 Edit	 Copy	 Delete	13	4
<input type="checkbox"/>	 Edit	 Copy	 Delete	14	5
<input type="checkbox"/>	 Edit	 Copy	 Delete	15	5
<input type="checkbox"/>	 Edit	 Copy	 Delete	16	6
<input type="checkbox"/>	 Edit	 Copy	 Delete	17	6
<input type="checkbox"/>	 Edit	 Copy	 Delete	18	9
<input type="checkbox"/>	 Edit	 Copy	 Delete	19	9
<input type="checkbox"/>	 Edit	 Copy	 Delete	20	10
<input type="checkbox"/>	 Edit	 Copy	 Delete	21	10

## Function\_table

		function_id	function_name	category	description
<input type="checkbox"/>	<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	1	print	I/O	Outputs text or values to standard output
<input type="checkbox"/>	<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	2	input	I/O	Reads input from standard input
<input type="checkbox"/>	<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	3	len	Utility	Returns the length of a collection or string
<input type="checkbox"/>	<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	4	sort	Sorting	Sorts elements in a collection
<input type="checkbox"/>	<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	5	map	Functional	Applies a function to each element
<input type="checkbox"/>	<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	6	filter	Functional	Filters elements based on a condition
<input type="checkbox"/>	<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	7	reduce	Functional	Reduces a collection to a single value
<input type="checkbox"/>	<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	8	open	File I/O	Opens a file for reading or writing
<input type="checkbox"/>	<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	9	split	String	Splits a string into an array/list
<input type="checkbox"/>	<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	10	join	String	Joins elements into a string

## Implementation

		implementation_id	language_id	version	date_added	example	result	notes	syntax
<input type="checkbox"/>	<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	1	1	3.x	2026-02-22	print("Score: ", 95, sep="")	Score: 95	Supports multiple arguments with customizable separator	print(value, sep="", end="\n")
<input type="checkbox"/>	<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	2	2	C++11	2026-02-22	std::cout << "Temperature: " << 72 << "F" << std::endl;	Temperature: 72F	Chain multiple values with << operator	std::cout << value << std::endl;
<input type="checkbox"/>	<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	3	3	Java 8+	2026-02-22	System.out.println("Items in cart: " + 5);	Items in cart: 5	Use + for string concatenation	System.out.println(value);
<input type="checkbox"/>	<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	4	4	ES6+	2026-02-22	console.log(`User \${name} logged in`);	User John logged in	Template literals use backticks and \${} for interpolation	console.log(value);
<input type="checkbox"/>	<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	5	1	3.x	2026-02-22	age = int(input("Enter your age: "))	User enters: 25 -> age = 25	Always returns string; cast to int/float as needed	variable = input(prompt)
<input type="checkbox"/>	<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	6	2	C++11	2026-02-22	int age; std::cout << "Enter age: "; std::cin >> a...	User enters: 25 -> age = 25	Use getline(cin, str) for strings with spaces	std::cin >> variable;
<input type="checkbox"/>	<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	7	3	Java 8+	2026-02-22	Scanner sc = new Scanner(System.in); String name = sc.nextLine();	User enters: Alice -> name = "Alice"	Use nextInt(), nextDouble() for other types	Scanner sc = new Scanner(System.in); sc.nextLine();
<input type="checkbox"/>	<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	8	1	3.x	2026-02-22	words = ["apple", "banana", "cherry"]	3	Works on strings, lists, tuples, dicts, sets	len(sequence)
<input type="checkbox"/>	<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	9	2	C++11	2026-02-22	std::string name = "Alice"; name.size();	5	Returns size_type; use length() for strings too	container.size()
<input type="checkbox"/>	<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	10	3	Java 8+	2026-02-22	int[] nums = {10, 20, 30, 40}; nums.length;	4	Arrays use length (no parens). Strings use length...	array.length or string.length() or list.size()
<input type="checkbox"/>	<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	11	1	3.x	2026-02-22	prices = [9.99, 9.99, 49.99, 19.99]; prices.sort();	[9.99, 19.99, 29.99, 49.99]	.sort() modifies in place; sorted() returns new list...	list.sort() or sorted(iterable)
<input type="checkbox"/>	<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	12	2	C++11	2026-02-22	std::vector<int> v = {5, 2, 8, 1}; std::sort(v.begin(), v.end());	{1, 2, 5, 8}	Requires #include <algorithm>; average O(n log n)	std::sort(begin, end)
<input type="checkbox"/>	<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	13	3	Java 8+	2026-02-22	List<String> names = Arrays.asList("Zoe", "Amy", "...");	[Amy, Max, Zoe]	Use Comparator for custom sorting	Arrays.sort(array) or Collections.sort(list)
<input type="checkbox"/>	<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	14	1	3.x	2026-02-22	nums = [1, 2, 3, 4]; squared = list(map(lambda x: x * x, nums));	[1, 4, 9, 16]	Returns iterator; wrap in list() to see results	map(function, iterable)
<input type="checkbox"/>	<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	15	4	ES6+	2026-02-22	const prices = [10, 20, 30]; const withTax = price * 1.1;	[10.8, 21.6, 32.4]	Arrow functions make this concise	array.map(callback)
<input type="checkbox"/>	<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	16	1	3.x	2026-02-22	ages = [12, 18, 25, 8, 30]; adults = list(filter(lambda age: age >= 18, ages));	[18, 25, 30]	Returns elements where function returns True	filter(function, iterable)
<input type="checkbox"/>	<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	17	4	ES6+	2026-02-22	const scores = [85, 42, 93, 67, 55]; const passing = scores.filter(lambda score: score >= 60);	[85, 93, 67]	Returns new array with elements that pass the test	array.filter(callback)
<input checked="" type="checkbox"/>	<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	18	1	3.x	2026-02-22	csv_line = "John,25,Engineer" fields = csv_line.split(",")	["John", "25", "Engineer"]	Default separator is whitespace	string.split(separator)

## Language

	<input type="button" value="←"/>	<input type="button" value="→"/>	<input type="button" value="▼"/>	language_id	language_name
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	5	C
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	2	C++
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	7	Go
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	3	Java
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	4	JavaScript
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	1	Python
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	6	Ruby
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	8	Rust

## Operator

	<input type="button" value="←"/>	<input type="button" value="→"/>	<input type="button" value="▼"/>	operator_id	operator_name	category	description
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	1	Addition	Arithmetic	Adds two operands
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	2	Subtraction	Arithmetic	Subtracts operands
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	3	Multiplication	Arithmetic	Multiplies operands
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	4	Division	Arithmetic	Divides operands
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	5	Modulus	Arithmetic	Returns remainder
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	6	Equal	Comparison	Checks equality
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	7	Not Equal	Comparison	Checks inequality
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	8	Greater Than	Comparison	Checks if greater
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	9	Less Than	Comparison	Checks if less
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	10	Logical AND	Logical	Both must be true
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	11	Logical OR	Logical	One must be true
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	12	Logical NOT	Logical	Inverts boolean
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	13	Assignment	Assignment	Assigns value
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	14	Bitwise AND	Bitwise	Bitwise AND operation
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	15	Bitwise OR	Bitwise	Bitwise OR operation

Operator\_implementation

	<input type="button" value="←"/>	<input type="button" value="→"/>		implementation_id	operator_id	symbol
<input type="checkbox"/>		<a href="#">Edit</a>			35	1 +
<input type="checkbox"/>		<a href="#">Edit</a>			36	1 +
<input type="checkbox"/>		<a href="#">Edit</a>			37	2 -
<input type="checkbox"/>		<a href="#">Edit</a>			38	3 *
<input type="checkbox"/>		<a href="#">Edit</a>			39	3 *
<input type="checkbox"/>		<a href="#">Edit</a>			40	4 / or //
<input type="checkbox"/>		<a href="#">Edit</a>			41	4 /
<input type="checkbox"/>		<a href="#">Edit</a>			42	5 %
<input type="checkbox"/>		<a href="#">Edit</a>			43	5 %
<input type="checkbox"/>		<a href="#">Edit</a>			44	6 ==
<input type="checkbox"/>		<a href="#">Edit</a>			45	6 === or ==
<input type="checkbox"/>		<a href="#">Edit</a>			46	7 !=
<input type="checkbox"/>		<a href="#">Edit</a>			47	8 >
<input type="checkbox"/>		<a href="#">Edit</a>			48	9 <
<input type="checkbox"/>		<a href="#">Edit</a>			49	10 and
<input type="checkbox"/>		<a href="#">Edit</a>			50	10 &&
<input type="checkbox"/>		<a href="#">Edit</a>			51	11 or
<input type="checkbox"/>		<a href="#">Edit</a>			52	11
<input type="checkbox"/>		<a href="#">Edit</a>			53	12 not
<input type="checkbox"/>		<a href="#">Edit</a>			54	12 !
<input type="checkbox"/>		<a href="#">Edit</a>			55	13 =

## Structure\_implementation

	<input type="button" value="←"/> <input type="button" value="→"/>	<input type="button" value="▼"/>	implementation_id	structure_id	
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	22	2
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	23	2
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	24	2
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	25	2
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	26	3
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	27	3
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	28	3
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	29	4
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	30	4
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	31	5
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	32	5
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	33	6
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	34	6

## Section 3/4: SQL Queries

### Query 1: SELECT with WHERE and Logical Operators

Purpose: Find all functions in Functional or String categories

SELECT function\_id, function\_name, category FROM function\_table WHERE category = 'Functional'  
OR category = 'String';

	<input type="button" value="←"/> <input type="button" value="→"/>	<input type="button" value="▼"/>	function_id	function_name	category
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	5 map	Functional
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	6 filter	Functional
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	7 reduce	Functional
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	9 split	String
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	10 join	String

### Query 2: JOIN Expression

Purpose: Compare sort function across languages using multiple table joins

SELECT l.language\_name, f.function\_name, i.syntax, i.example FROM implementation i JOIN functionImplementation fi ON i.implementation\_id = fi.implementation\_id JOIN language l ON i.language\_id = l.language\_id JOIN function\_table f ON fi.function\_id = f.function\_id WHERE f.function\_name = 'sort';

language_name	function_name	syntax	example
Python	sort	list.sort() or sorted(iterable)	prices = [29.99, 9.99, 49.99, 19.99] prices.sort()
C++	sort	std::sort(begin, end)	std::vector<int> v = {5, 2, 8, 1}; std::sort(v.begin, v.end);
Java	sort	Arrays.sort(array) or Collections.sort(list)	List<String> names = Arrays.asList("Zoe", "Amy", "..."); Collections.sort(names);

### Query 3: GROUP BY with Aggregate Functions

Purpose: Count operators per category using GROUP BY and COUNT

SELECT category, COUNT(\*) AS operator\_count FROM operator GROUP BY category ORDER BY operator\_count DESC;

category	operator_count
Arithmetic	5
Comparison	4
Logical	3
Bitwise	2
Assignment	1

### Query 4: View Query

Purpose: Use the pre built view to find all Python function implementations

SELECT language\_name, function\_name, syntax, example FROM vw\_function\_lookup WHERE language\_name = 'Python';

language_name	function_name	syntax	example
Python	filter	filter(function, iterable)	ages = [12, 18, 25, 8, 30] adults = list(filter(lambda x: x >= 18, ages))
Python	input	variable = input(prompt)	age = int(input("Enter your age: "))
Python	join	separator.join(iterable)	words = ["Hello", "World", "2024"] " ".join(words)
Python	len	len(sequence)	words = ["apple", "banana", "cherry"] len(words)
Python	map	map(function, iterable)	nums = [1, 2, 3, 4] squared = list(map(lambda x: x * x, nums))
Python	print	print(value, sep=" ", end="\n")	print("Score:", 95, sep=" ")
Python	sort	list.sort() or sorted(iterable)	prices = [29.99, 9.99, 49.99, 19.99] prices.sort()
Python	split	string.split(separator)	csv_line = "John,25,Engineer" fields = csv_line.split(",")

### Query 5: COUNT all available functions

**Purpose:** show the number of functions that could be “translated” or that are stored in the database.

```
SELECT COUNT(*) AS total_functions FROM function_table;
```

total_functions
-----------------

10
----