DBMS LAB

Name – RAVI SHEKHAR

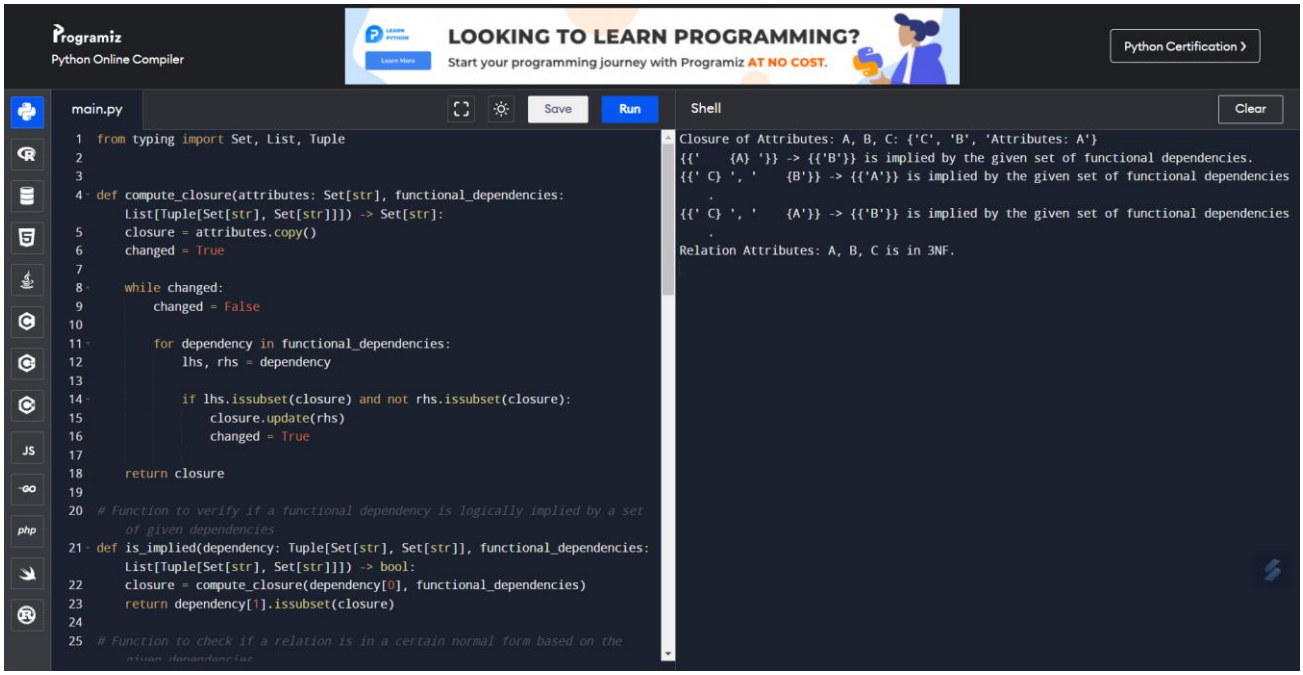
Roll No. – 22CS3075

1.

```
from typing import Set, List, Tuple

def compute_closure(attributes: Set[str], functional_dependencies: List[Tuple[Set[str], Set[str]]]) -> Set[str]:
    closure = attributes.copy()
    changed = True

    while changed:
        changed = False

        for dependency in functional_dependencies:
            lhs, rhs = dependency

            if lhs.issubset(closure) and not rhs.issubset(closure):
                closure.update(rhs)
                changed = True

    return closure

# Function to verify if a functional dependency is logically implied by a set of given dependencies
def is_implied(dependency: Tuple[Set[str], Set[str]], functional_dependencies: List[Tuple[Set[str], Set[str]]]) -> bool:
```

```
    closure = compute_closure(dependency[0], functional_dependencies)
    return dependency[1].issubset(closure)

# Function to check if a relation is in a certain normal form based on the given dependencies
def is_in_normal_form(functional_dependencies: List[Tuple[Set[str], Set[str]]], relation: str, normal_form: str) -> bool:
    for dependency in functional_dependencies:
        lhs, rhs = dependency

        if normal_form == "1NF":
            if len(lhs) > 1 or len(rhs) > 1:
                return False
        elif normal_form == "2NF":
            if not lhs.issubset(rhs) and not is_implied(dependency, functional_dependencies):
                return False
        elif normal_form == "3NF":
            if not is_implied(dependency, functional_dependencies):
                return False

    return True

# Function to parse input data and extract relations and functional dependencies
def parse_input(input_data: str) -> Tuple[str, List[Tuple[Set[str], Set[str]]]]:
    lines = input_data.strip().split('\n')
    relation = lines[0].split('(')[1].strip(')')

    functional_dependencies = []
    for line in lines[1:]:
        lhs_str, rhs_str = line.split('->')
        lhs = set(lhs_str.strip('{}').split(','))
        rhs = set(rhs_str.strip().strip('{}').split(','))
```

```
        functional_dependencies.append((lhs, rhs))

    return relation, functional_dependencies

# Function to display the computed results
def display_output(closure: Set[str], functional_dependencies: List[Tuple[Set[str], Set[str]]], relation: str, normal_form: str) -> None:
    print(f"Closure of {relation}: {closure}")

    for dependency in functional_dependencies:
        lhs, rhs = dependency
        implied = "implied" if is_implied(dependency, functional_dependencies) else "not implied"
        print(f"{{{lhs}}} -> {{{rhs}}} is {implied} by the given set of functional dependencies.")

    print(f"Relation {relation} is {'in' if is_in_normal_form(functional_dependencies, relation, normal_form) else 'not in'} {normal_form}.")

def main():
    input_data = """R (Attributes: A, B, C)
    {A} -> {B}
    {B, C} -> {A}
    {A, C} -> {B}"""

    relation, functional_dependencies = parse_input(input_data)
    attributes = set(relation.split(', '))
    closure = compute_closure(attributes, functional_dependencies)
    normal_form = "3NF"  # Change the normal form here

    display_output(closure, functional_dependencies, relation, normal_form)

if __name__ == "__main__":
    main()
```


Programiz
Python Online Compiler
LOOKING TO LEARN PROGRAMMING?
Start your programming journey with Programiz AT NO COST.
Python Certification
main.py
Save
Run
Shell
Clear
from typing import Set, List, Tuple
def compute_closure(attributes: Set[str], functional_dependencies: List[Tuple[Set[str], Set[str]]]) -> Set[str]:
closure = attributes.copy()
changed = True
while changed:
changed = False
for dependency in functional_dependencies:
lhs, rhs = dependency
if lhs.issubset(closure) and not rhs.issubset(closure):
closure.update(rhs)
changed = True
return closure
# Function to verify if a functional dependency is logically implied by a set of given dependencies
def is_implied(dependency: Tuple[Set[str], Set[str]], functional_dependencies: List[Tuple[Set[str], Set[str]]]) -> bool:
closure = compute_closure(dependency[0], functional_dependencies)
return dependency[1].issubset(closure)
Function to check if a relation is in a certain normal form based on the
Closure of Attributes: A, B, C: {'C', 'B', 'Attributes: A'}
{{' {A} '}} -> {{'B'}} is implied by the given set of functional dependencies.
{{' C} ', ' {B'}} -> {{'A'}} is implied by the given set of functional dependencies
.
{{' C} ', ' {A'}} -> {{'B'}} is implied by the given set of functional dependencies
.
Relation Attributes: A, B, C is in 3NF.