CSE 4308
Database Management Systems Lab

Lab-10

Group-B

Mohammod Mahbub Ur Rahman
220042148
BSc. in Software Engineering

Task 1

a.

```
DELIMITER //
CREATE FUNCTION name_and_id()
RETURNS VARCHAR(100)
DETERMINISTIC -- always returns the same results if given the same input values
NO SQL
BEGIN
    RETURN 'Mahbub, 220042148';
END //
DELIMITER ;

SELECT name_and_id() AS name_and_id;
```

Here DELIMITER defines the start and the end of a function or procedure in mySQL. **CREATE FUNCTION** defines the name of the function. after **RETURNS** we define which type of value the function should return.

DETERMINISTIC always returns the same results if given the same input values. NO SQL means the function will not do any sql queries.

In the return I am giving my name and ID and the function is returning it.

To execute a function we use the following syntax `SELECT name_and_id() AS name_and_id;`

b.

```
DELIMITER //
CREATE FUNCTION c_product(a INT, b INT)
RETURNS INT
DETERMINISTIC
NO SQL
BEGIN
    RETURN a * b;
END //
DELIMITER ;

SELECT c_product(2, 5) AS product;
```

Here I am taking 2 int and returning their product.

c.

```
DELIMITER //
CREATE FUNCTION c_number_type(num
DECIMAL(10,2))
RETURNS VARCHAR(20)
DETERMINISTIC
NO SQL
BEGIN
    IF MOD(num, 1) = 0 THEN
        RETURN 'Whole Number';
    ELSE
        RETURN 'Fraction';
    END IF;
END //
DELIMITER ;
-- Whole Number
SELECT c_number_type(5.0) AS number_type;
-- Fraction
SELECT c_number_type(5.5) AS number_type;
```

Here I am calculating the mod of the num and 1. If the mod is 0 then the number is a whole number. if it has fraction part then the mod won't be 0 so it returns “fraction”

Task 2

```
DELIMITER //

CREATE FUNCTION stats_count(pokemon_id INT)
RETURNS INT
DETERMINISTIC
READS SQL DATA
BEGIN
    DECLARE total_stats INT;

    SELECT hp + attack + defense + speed
    INTO total_stats
    FROM pokemon
    WHERE pokemon.pokemon_id = pokemon_id;

    RETURN total_stats;
END //

DELIMITER ;

SELECT stats_count(1) AS total_stats;
```

Here I am declaring a variable and storing the sum of the stats of the pokemon whose ID is given as a parameter using sql queries.

Task 3

```
DELIMITER //

CREATE FUNCTION get_pokemon_type(pokemon_id INT)
RETURNS VARCHAR(100)
DETERMINISTIC
READS SQL DATA
BEGIN
    DECLARE type_string VARCHAR(100);

    SELECT CONCAT(t1.type_name, IFNULL(CONCAT(', ', t2.type_name), ''))
    INTO type_string
    FROM pokemon p
    LEFT JOIN types t1 ON p.primary_type_id = t1.type_id
    LEFT JOIN types t2 ON p.secondary_type_id = t2.type_id
    WHERE p.pokemon_id = pokemon_id;

    RETURN type_string;
END //

DELIMITER ;

SELECT get_pokemon_type(2) AS pokemon_type;
```

Here I am declaring a variable to store the type of the pokemon. I am using concat to add together 2 type of a pokemon (if it has any)

Task 4

```
DELIMITER //

CREATE PROCEDURE get_pokemon(IN trainer_id
INT)
BEGIN
    SELECT p.name AS pokemon_name,
get_pokemon_type(p.pokemon_id) AS pokemon_type
    FROM trainer_pokemon tp
    JOIN pokemon p ON tp.pokemon_id =
p.pokemon_id
    WHERE tp.trainer_id = trainer_id;
END //

DELIMITER ;

CALL get_pokemon(1);
```

Procedure doesn't return anything. I am using sql queries to find the trainer’s pokemons and taking their names and types.

Task 5

```
DELIMITER //

CREATE PROCEDURE level_up(IN pokemon_id INT, IN x INT)
BEGIN
    -- Before stats
    SELECT hp, attack, defense, speed
    FROM pokemon
    WHERE pokemon.pokemon_id = pokemon_id;

    -- Update stats
    UPDATE pokemon
    SET hp = hp + x,
        attack = attack + x,
        defense = defense + x,
        speed = speed + x
    WHERE pokemon.pokemon_id = pokemon_id;

    -- Display after stats
    SELECT hp, attack, defense, speed
    FROM pokemon
    WHERE pokemon.pokemon_id = pokemon_id;
END //

DELIMITER ;

-- Call the procedure
CALL level_up(1, 10);
```

Here I am updating the stats By X of a given pokemon id.