SRN PES2UG22CS385

DBMS: LAB 10

TRIGGERS AND DCL

University Fest Management System

INSTRUCTIONS

- In Lab 10 you are expected to solve 2 tasks that are to be completed and submitted.
- For this Lab 10, you would have to continue with the University Fest DB given to you last week.
- As a part of LAB 10, 2 tasks are to be completed as described below:
    - **TASK 1:** This task has 2+1 questions. You are expected to go through the questions and write appropriate SQL queries.
    - **Task 2:** This task 2 questions based on DCL commands. You are expected to go through the questions and write appropriate DCL statements. Follow the instructions stated in the questions for the outputs to be submitted.
- Ensure that your mysql **command client prompt is modified as per your SRN** using the command:

    prompt YOUR_SRN>
- Note that this step is **mandatory**.
- As a part of the submission process, the following are to be submitted:
    - A **PDF** document, containing all the Screenshots for all 2 tasks as suggested
        - Name of the file: `<your SRN>_University_Fest_DB_Lab10.pdf`

TASK 1:

Understand the various scenarios described in the given questions and write the corresponding SQL queries for the same. Support your answers with the help of the screenshot.

Question 1:

Oftentimes, it is necessary to conduct events on multiple days due to a large number of participants. A committee in the university decided that any event in which the number of participants exceeds 10 must be conducted for 2 days. Write a trigger that is activated whenever a participant registers for an event. This trigger should automatically allocate a day to the event when the number of participants exceeds 10. Demonstrate the working of your trigger using the following cases:

INSERT INTO REGISTRATION VALUES ('E11', 'P1009', 'R61');

INSERT INTO REGISTRATION VALUES ('E12', 'P1001', 'R61');

INSERT INTO REGISTRATION VALUES ('E17', 'P1001', 'R60');

NOTE:

- When the number of participants exceeds 10, allocate the day next to the existing date in the event_conduction table.
- **Before and After execution of the insert statements, display the "event_conduction" table**

```
mysql> prompt PES2UG22CS385>;
PROMPT set to 'PES2UG22CS385>'
PES2UG22CS385>DELIMITER //
PES2UG22CS385>
PES2UG22CS385>CREATE TRIGGER allocate_event_day
    -> AFTER INSERT ON registration
    -> FOR EACH ROW
    -> BEGIN
    ->     DECLARE participant_count INT;
    ->     DECLARE event_date DATE;
    ->     DECLARE next_day DATE;
    ->
    ->     -- Count the number of participants for the event
    ->     SELECT COUNT(*) INTO participant_count
    ->     FROM registration
    ->     WHERE event_id = NEW.event_id;
    ->
    ->     -- If participant count exceeds 10, allocate an additional day
    ->     IF participant_count > 10 THEN
    ->         -- Get the current date of conduction for the event
    ->         SELECT MAX(date_of_conduction) INTO event_date
    ->         FROM event_conduction
    ->         WHERE event_id = NEW.event_id;
    ->
    ->         -- Allocate the next day
    ->         SET next_day = DATE_ADD(event_date, INTERVAL 1 DAY);
    ->
    ->         -- Insert the next day into event_conduction
    ->         INSERT INTO event_conduction (event_id, date_of_conduction)
    ->         VALUES (NEW.event_id, next_day);
    ->     END IF;
    -> END //
Query OK, 0 rows affected (0.27 sec)

PES2UG22CS385>
PES2UG22CS385>DELIMITER ;
PES2UG22CS385>
```

BEFORE:

```
PES2UG22CS385>select * from event_conduction;
+----------+--------------------+
| event_id | date_of_conduction |
+----------+--------------------+
| E1       | 2023-04-15         |
| E10      | 2022-04-17         |
| E11      | 2022-04-18         |
| E12      | 2022-04-19         |
| E13      | 2022-04-15         |
| E14      | 2022-04-16         |
| E15      | 2022-04-17         |
| E16      | 2022-04-18         |
| E17      | 2022-04-19         |
| E17      | 2022-04-20         |
| E18      | 2022-04-15         |
| E19      | 2022-04-16         |
| E2       | 2023-04-16         |
| E20      | 2022-04-17         |
| E21      | 2022-04-18         |
| E22      | 2022-04-19         |
| E23      | 2022-04-15         |
| E24      | 2022-04-16         |
| E25      | 2021-04-15         |
| E26      | 2021-04-16         |
| E27      | 2021-04-17         |
| E29      | 2021-04-19         |
| E3       | 2023-04-17         |
| E30      | 2021-04-15         |
| E31      | 2021-04-16         |
| E32      | 2021-04-17         |
| E32      | 2021-04-18         |
| E4       | 2023-04-18         |
| E5       | 2023-04-19         |
| E6       | 2023-04-15         |
| E7       | 2023-04-16         |
| E8       | 2023-04-17         |
| E9       | 2022-04-15         |
| E9       | 2022-04-16         |
+----------+--------------------+
34 rows in set (0.03 sec)
```

```
PES2UG22CS385>INSERT INTO registration VALUES ('E11', 'P1009', 'R61');
Query OK, 1 row affected (0.16 sec)

PES2UG22CS385>INSERT INTO registration VALUES ('E12', 'P1001', 'R61');
Query OK, 1 row affected (0.01 sec)

PES2UG22CS385>INSERT INTO registration VALUES ('E17', 'P1001', 'R60');
Query OK, 1 row affected (0.01 sec)

PES2UG22CS385>
```

AFTER:

```
PES2UG22CS385>select * from event_conduction;
+----------+--------------------+
| event_id | date_of_conduction |
+----------+--------------------+
| E1       | 2023-04-15         |
| E10      | 2022-04-17         |
| E11      | 2022-04-18         |
| E11      | 2022-04-19         |
| E12      | 2022-04-19         |
| E13      | 2022-04-15         |
| E14      | 2022-04-16         |
| E15      | 2022-04-17         |
| E16      | 2022-04-18         |
| E17      | 2022-04-19         |
| E17      | 2022-04-20         |
| E17      | 2022-04-21         |
| E18      | 2022-04-15         |
| E19      | 2022-04-16         |
| E2       | 2023-04-16         |
| E20      | 2022-04-17         |
| E21      | 2022-04-18         |
| E22      | 2022-04-19         |
| E23      | 2022-04-15         |
| E24      | 2022-04-16         |
| E25      | 2021-04-15         |
| E26      | 2021-04-16         |
| E27      | 2021-04-17         |
| E29      | 2021-04-19         |
| E3       | 2023-04-17         |
| E30      | 2021-04-15         |
| E31      | 2021-04-16         |
| E32      | 2021-04-17         |
| E32      | 2021-04-18         |
| E4       | 2023-04-18         |
| E5       | 2023-04-19         |
| E6       | 2023-04-15         |
| E7       | 2023-04-16         |
| E8       | 2023-04-17         |
| E9       | 2022-04-15         |
| E9       | 2022-04-16         |
+----------+--------------------+
36 rows in set (0.00 sec)

PES2UG22CS385>
```

Question 2

Due to the size of stalls allowed in the fest, stall owners were able to prepare the various items in limited quantities. Hence, there is a possibility that a food item requested by a customer is no longer available as it has been sold out. Therefore, every time a transaction is made, stall owners would like to check if the desired number of the requested item is available. If yes, then the transaction is valid. Otherwise, the transaction is deemed invalid. Write a trigger that automates the checking of transaction validity. If the transaction is valid, it must deduct the requested quantity from the available quantity of the item. If the transaction is invalid, it must display an appropriate error message. Demonstrate the working of your trigger using two cases:

- **Transaction with the requested number of items lesser than the available number of items**
- **Transaction with the requested number of items greater than the available number of items**

Note: Demonstrate both valid and invalid transaction

Example for valid transaction:

```
mysql> INSERT INTO PURCHASED VALUES('P1001','S1','Caprese Salad','2022-04-19 10:00:15',9);
Query OK, 1 row affected (0.01 sec)
```

Example for invalid transaction:

```
mysql> INSERT INTO PURCHASED VALUES('P1001','S1','Classic Caesar Salad','2022-04-19 10:00:15',49);
ERROR 1644 (45000): Not enough number of requested items present in stall
```

```
PES2UG22CS385>DELIMITER //
PES2UG22CS385>
PES2UG22CS385>CREATE TRIGGER check_item_availability
    ->     BEFORE INSERT ON purchased
    ->     FOR EACH ROW
    -> BEGIN
    ->     DECLARE available_quantity INT;
    ->
    ->     -- Fetch the available quantity of the item from the stall_items table
    ->     SELECT si.total_quantity INTO available_quantity
    ->     FROM stall_items si
    ->     WHERE si.item_name = NEW.item_name AND si.stall_id = NEW.stall_id;
    ->
    ->     -- Check if enough items are available
    ->     IF available_quantity < NEW.quantity THEN
    ->         -- Raise an error if not enough items are available
    ->         SIGNAL SQLSTATE '45000'
    ->         SET MESSAGE_TEXT = 'Not enough number of requested items present in stall';
    ->     ELSE
    ->         -- Deduct the purchased quantity from the available quantity
    ->         UPDATE stall_items
    ->         SET total_quantity = total_quantity - NEW.quantity
    ->         WHERE stall_id = NEW.stall_id AND item_name = NEW.item_name;
    ->     END IF;
    -> END //
Query OK, 0 rows affected (0.07 sec)

PES2UG22CS385>
PES2UG22CS385>DELIMITER ;
PES2UG22CS385>
```

```
PES2UG22CS385>
PES2UG22CS385>DELIMITER ;
PES2UG22CS385>INSERT INTO purchased VALUES ('P1001', 'S1', 'Caprese Salad', '2022-04-19 10:00:15', 9);
Query OK, 1 row affected (0.10 sec)

PES2UG22CS385>INSERT INTO purchased VALUES ('P1001', 'S1', 'Classic Caesar Salad', '2022-04-19 10:00:15', 49);
ERROR 1644 (45000): Not enough number of requested items present in stall
PES2UG22CS385>
```

Question 3:

There are various teams involved in making a fest a grand success. Organizing teams, in particular, bring in a lot of revenue by conducting several fun-filled and exciting events. The university management is interested in finding the best-performing organizing team in a fest. The best-performing organizing team can be defined as the team that brings in the highest revenue from event registration sales. Write a procedure that returns the best-performing team id and their revenue, given the fest id (for students to solve)

NOTE:

The procedure returns a single tuple with two attributes. It must be in the given format:

teamid	revenue

Sample output:

```
mysql> call GetTopRevenueTeams('F101');
+--------+----------+
| teamid | revenue  |
+--------+----------+
| T4     | 13850.00 |
+--------+----------+
1 row in set (0.07 sec)
```

//Add procedure code and output screenshot

```
PES2UG22CS385>DELIMITER //
PES2UG22CS385>
PES2UG22CS385>CREATE PROCEDURE GetBestPerformingTeam(IN festID VARCHAR(5))
    -> BEGIN
    ->     DECLARE best_team_id VARCHAR(5);
    ->     DECLARE highest_revenue DECIMAL(10, 2);
    ->
    ->     SELECT
    ->         t.team_id,
    ->         COALESCE(SUM(e.price), 0) AS total_revenue
    ->     INTO
    ->         best_team_id, highest_revenue
    ->     FROM
    ->         team t
    ->     JOIN
    ->         event e ON t.team_id = e.team_id
    ->     LEFT JOIN
    ->         registration r ON e.event_id = r.event_id
    ->     WHERE
    ->         t.fest_id = festID
    ->     GROUP BY
    ->         t.team_id
    ->     ORDER BY
    ->         total_revenue DESC
    ->     LIMIT 1;
    ->
    ->     SELECT best_team_id AS teamid, highest_revenue AS revenue;
    -> END //
Query OK, 0 rows affected (0.10 sec)

PES2UG22CS385>
PES2UG22CS385>DELIMITER ;

PES2UG22CS385>CALL GetBestPerformingTeam('F101');
+--------+----------+
| teamid | revenue  |
+--------+----------+
| T4     | 13850.00 |
+--------+----------+
1 row in set (0.09 sec)

Query OK, 0 rows affected (0.10 sec)

PES2UG22CS385>
```

TASK 2:

This task is focused on the different users who would be using the university fest database. According to the case study, many users would use the database for different purposes. Each would have its own set of permissions for its functions without any violations.

In this task, there would be 2 questions or situations that you would have to solve. In the context of the university fest database there would be many types of users like:

- **Participants**: who only have permission to view the information about the FEST, TEAM, EVENTS, and the STALLS present.
- **Team coordinator**: who is responsible for taking care of all the management of teams and events conducted by various teams. Any updation of an existing event/team, deletion of a canceled event/team, or insertion of a new event/team is handled.
- **Stall coordinator**: who is responsible for managing the entire stalls, items, items sold in stalls, and the purchases made. This coordinator would have all the permissions to view, insert, update, and delete information related to any of the tables related to stalls and their management and purchases.
- **Fest Coordinator**: who is granted all the permissions on the database and can do any of the modifications, alterations, etc. to the database

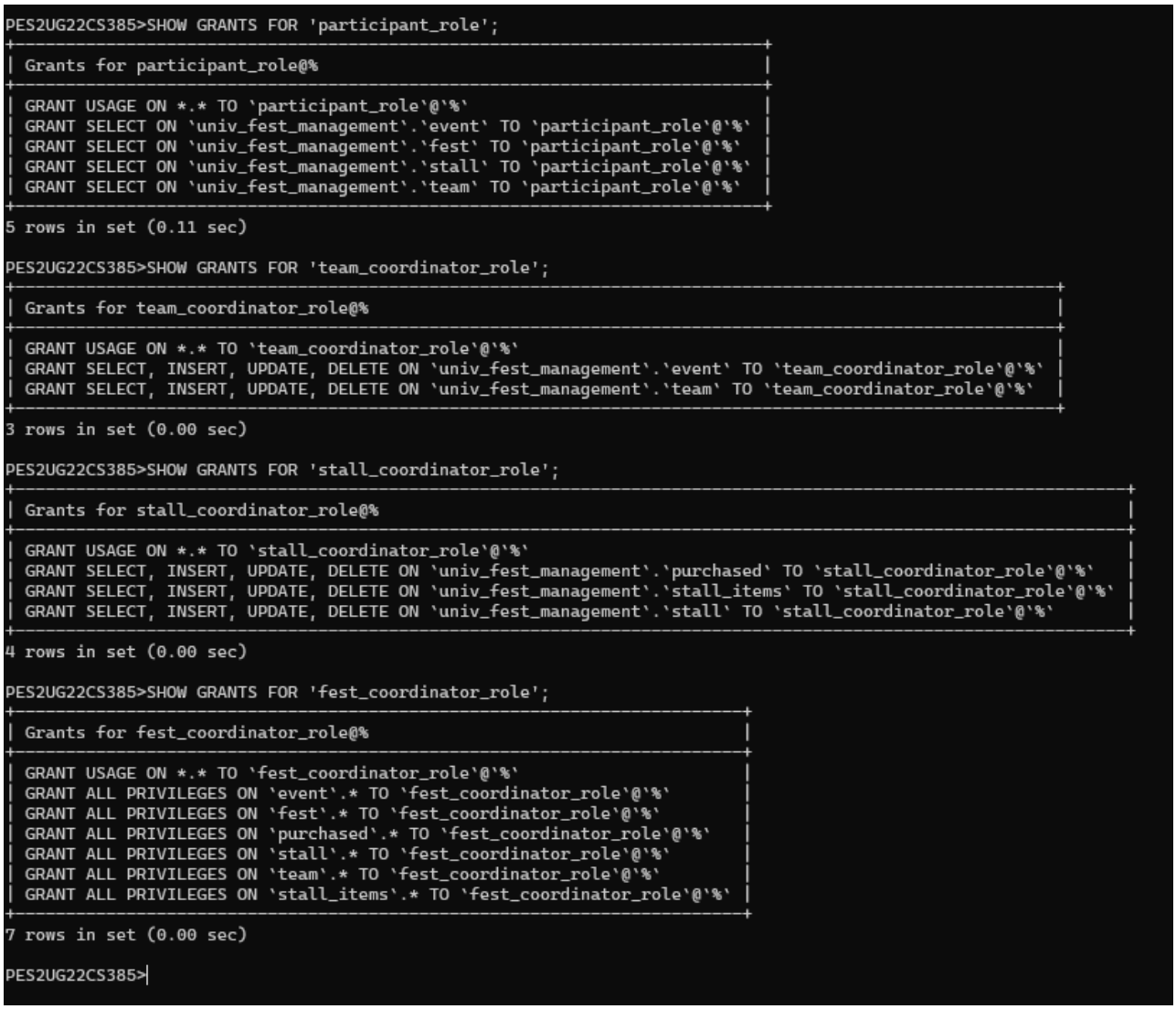
Question 1:

Create the required roles who would probably use the university fest database as specified above. And also grant the required permissions for the roles.

NOTE:

- Display all the roles after their creation
- Display the permissions you have given to each of the roles

```
PES2UG22CS385>CREATE ROLE 'participant_role';
Query OK, 0 rows affected (0.01 sec)

PES2UG22CS385>CREATE ROLE 'team_coordinator_role';
Query OK, 0 rows affected (0.01 sec)

PES2UG22CS385>CREATE ROLE 'stall_coordinator_role';
Query OK, 0 rows affected (0.01 sec)

PES2UG22CS385>CREATE ROLE 'fest_coordinator_role';
Query OK, 0 rows affected (0.01 sec)

PES2UG22CS385>GRANT SELECT ON fest TO 'participant_role';
Query OK, 0 rows affected (0.04 sec)

PES2UG22CS385>GRANT SELECT ON team TO 'participant_role';
Query OK, 0 rows affected (0.01 sec)

PES2UG22CS385>GRANT SELECT ON event TO 'participant_role';
Query OK, 0 rows affected (0.01 sec)

PES2UG22CS385>GRANT SELECT ON stall TO 'participant_role';
Query OK, 0 rows affected (0.01 sec)

PES2UG22CS385>GRANT SELECT, INSERT, UPDATE, DELETE ON team TO 'team_coordinator_role';
Query OK, 0 rows affected (0.01 sec)

PES2UG22CS385>GRANT SELECT, INSERT, UPDATE, DELETE ON event TO 'team_coordinator_role';
Query OK, 0 rows affected (0.01 sec)

PES2UG22CS385>GRANT SELECT, INSERT, UPDATE, DELETE ON stall TO 'stall_coordinator_role';
Query OK, 0 rows affected (0.01 sec)

PES2UG22CS385>GRANT SELECT, INSERT, UPDATE, DELETE ON stall_items TO 'stall_coordinator_role';
Query OK, 0 rows affected (0.01 sec)

PES2UG22CS385>GRANT SELECT, INSERT, UPDATE, DELETE ON purchased TO 'stall_coordinator_role';
Query OK, 0 rows affected (0.01 sec)

PES2UG22CS385>GRANT ALL PRIVILEGES ON fest.* TO 'fest_coordinator_role';
Query OK, 0 rows affected (0.02 sec)

PES2UG22CS385>GRANT ALL PRIVILEGES ON team.* TO 'fest_coordinator_role';
Query OK, 0 rows affected (0.01 sec)

PES2UG22CS385>GRANT ALL PRIVILEGES ON event.* TO 'fest_coordinator_role';
Query OK, 0 rows affected (0.01 sec)

PES2UG22CS385>GRANT ALL PRIVILEGES ON stall.* TO 'fest_coordinator_role';
Query OK, 0 rows affected (0.01 sec)

PES2UG22CS385>GRANT ALL PRIVILEGES ON stall_items.* TO 'fest_coordinator_role';
Query OK, 0 rows affected (0.01 sec)

PES2UG22CS385>GRANT ALL PRIVILEGES ON purchased.* TO 'fest_coordinator_role';
Query OK, 0 rows affected (0.01 sec)

PES2UG22CS385>
```



//Add commands and appropriate screenshots

Question 2:

The Fest database designed would be used by multiple organizations. As a Database Administrator, you are given the responsibility to create the users and assign the roles to each user of the database for a certain university XYZ.

Description of users for university XYZ:

There would be two people who would be using the fest database as a participant:

“Alex” and “Emily”.

“Diana” is the user who would be acting as the fest coordinator.

“Andres” has been selected as the team coordinator.

“Bella” has been recruited as the stall coordinator.

“Derik”, who is just a user, has not been any responsibility yet.

NOTE:

- For the above given, create the users, and assign them their roles
- The name of the user is the same as the name of the person
- DO NOT separately grant permission to each user, all of this should be done by assigning a particular role to each person
- Display the privileges of each user

//Add commands and appropriate screenshots

```
PES2UG22CS385>CREATE USER 'Alex'@'localhost' IDENTIFIED BY 'password1';
Query OK, 0 rows affected (0.63 sec)

PES2UG22CS385>CREATE USER 'Emily'@'localhost' IDENTIFIED BY 'password2';
Query OK, 0 rows affected (0.02 sec)

PES2UG22CS385>CREATE USER 'Diana'@'localhost' IDENTIFIED BY 'password3';
Query OK, 0 rows affected (0.01 sec)

PES2UG22CS385>CREATE USER 'Andres'@'localhost' IDENTIFIED BY 'password4';
Query OK, 0 rows affected (0.02 sec)

PES2UG22CS385>CREATE USER 'Bella'@'localhost' IDENTIFIED BY 'password5';
Query OK, 0 rows affected (0.02 sec)

PES2UG22CS385>CREATE USER 'Derik'@'localhost' IDENTIFIED BY 'password6';
Query OK, 0 rows affected (0.01 sec)

PES2UG22CS385>GRANT participant_role TO 'Alex'@'localhost';
Query OK, 0 rows affected (0.01 sec)

PES2UG22CS385>GRANT participant_role TO 'Emily'@'localhost';
Query OK, 0 rows affected (0.01 sec)

PES2UG22CS385>GRANT fest_coordinator_role TO 'Diana'@'localhost';
Query OK, 0 rows affected (0.01 sec)

PES2UG22CS385>GRANT team_coordinator_role TO 'Andres'@'localhost';
Query OK, 0 rows affected (0.01 sec)

PES2UG22CS385>GRANT stall_coordinator_role TO 'Bella'@'localhost';
Query OK, 0 rows affected (0.01 sec)

PES2UG22CS385>
```

No command for Derik because he is not granted any role.

```
PES2UG22CS385>SHOW GRANTS FOR 'Alex'@'localhost';
+-----------------------------------------------+
| Grants for Alex@localhost                     |
+-----------------------------------------------+
| GRANT USAGE ON *.* TO `Alex`@`localhost`      |
| GRANT `participant_role`@`%` TO `Alex`@`localhost` |
+-----------------------------------------------+
2 rows in set (0.00 sec)

PES2UG22CS385>SHOW GRANTS FOR 'Emily'@'localhost';
+-----------------------------------------------+
| Grants for Emily@localhost                    |
+-----------------------------------------------+
| GRANT USAGE ON *.* TO `Emily`@`localhost`     |
| GRANT `participant_role`@`%` TO `Emily`@`localhost` |
+-----------------------------------------------+
2 rows in set (0.00 sec)

PES2UG22CS385>SHOW GRANTS FOR 'Diana'@'localhost';
+-------------------------------------------------------+
| Grants for Diana@localhost                            |
+-------------------------------------------------------+
| GRANT USAGE ON *.* TO `Diana`@`localhost`             |
| GRANT `fest_coordinator_role`@`%` TO `Diana`@`localhost` |
+-------------------------------------------------------+
2 rows in set (0.00 sec)

PES2UG22CS385>SHOW GRANTS FOR 'Andres'@'localhost';
+-------------------------------------------------------+
| Grants for Andres@localhost                           |
+-------------------------------------------------------+
| GRANT USAGE ON *.* TO `Andres`@`localhost`            |
| GRANT `team_coordinator_role`@`%` TO `Andres`@`localhost` |
+-------------------------------------------------------+
2 rows in set (0.00 sec)

PES2UG22CS385>SHOW GRANTS FOR 'Bella'@'localhost';
+-------------------------------------------------------+
| Grants for Bella@localhost                            |
+-------------------------------------------------------+
| GRANT USAGE ON *.* TO `Bella`@`localhost`             |
| GRANT `stall_coordinator_role`@`%` TO `Bella`@`localhost` |
+-------------------------------------------------------+
2 rows in set (0.00 sec)

PES2UG22CS385>SHOW GRANTS FOR 'Derik'@'localhost';
+-----------------------------------------+
| Grants for Derik@localhost              |
+-----------------------------------------+
| GRANT USAGE ON *.* TO `Derik`@`localhost` |
+-----------------------------------------+
1 row in set (0.00 sec)

PES2UG22CS385>
```