Database Management System
(UE10CS301)
Assignment - 3 Report

Team: 10

Team Members:

Name:	SRN:
Aliyah Kabeer	PES2UG19CS031
Anwesha Kelkar	PES2UG19CS058
Arunav Dey	PES2UG19CS066

Query statements with output screenshots:

1. Simple Queries:

1. SELECT * FROM employees;

```
restaurants=# SELECT * FROM employees;
 designation | staff_id | salary |  name   | managerid | restaurantid
-------------+----------+--------+---------+-----------+--------------
 Cook        |        1 |  20000 | Hari    |         2 |            1
 Cook        |        2 |  10000 | Harita  |         1 |            2
 Cook        |        3 |  50000 | Haria   |         2 |            3
 Cook        |        4 |  70000 | Arita   |         1 |            4
 Cook        |        5 |  90000 | Hart    |         2 |            5
 Cook        |        6 |  15000 | Rita    |         1 |            6
(6 rows)
```

2. SELECT registrationnumber, address,rating FROM restaurants;

```
restaurants=# SELECT registrationnumber, address, rating FROM restaurants;
  registrationnumber |           address            | rating
---------------------+------------------------------+--------
                   1 | Indiranagar, Bengaluru       |      4
                   2 | Koramangala, Bengaluru       |      5
                   3 | Indiranagar, Bengaluru       |      3
                   4 | Koramangala, Bengaluru       |      2
                   5 | Hennur, Bengaluru            |      4
                   6 | Electronic City, Bengaluru   |      1
(6 rows)
```

3. SELECT contactno,website FROM restaurants WHERE ranting=5;

```
restaurants=# SELECT contactno,website FROM restaurants WHERE rating=5;
 contactno  | website
------------+---------
 9976543210 | kfc.com
(1 row)
```

4. UPDATE restaurants SET rating=4 WHERE name='KFC';

```
restaurants=# UPDATE restaurants SET rating=4 WHERE name='KFC';
UPDATE 1
restaurants=# SELECT * FROM restaurants
restaurants-# ;
 registrationnumber | contactno  |          address          |    website     |    name     | rating | managerid | menuid
--------------------+------------+---------------------------+----------------+-------------+--------+-----------+-------
                  1 | 9876543210 | Indiranagar, Bengaluru    | mcdonalds.com  | McDonalds   |      4 |         2 |      2
                  3 | 9476543110 | Indiranagar, Bengaluru    | leongrill.com  | Leon Grill  |      3 |         2 |      1
                  4 | 9977543210 | Koramangala, Bengaluru    | burgerking.com | Burger King |      2 |         1 |      3
                  5 | 9976543310 | Hennur, Bengaluru         | pizzahut.com   | Pizza Hut   |      4 |         2 |      1
                  6 | 9976593210 | Electronic City, Bengaluru | dominos.com    | Dominos     |      1 |         1 |      2
                  2 | 9976543210 | Koramangala, Bengaluru    | kfc.com        | KFC         |      4 |         1 |      2
(6 rows)
```

5. DELETE FROM employees WHERE staff_id=1;

```
restaurants=# DELETE FROM employees WHERE staff_id=1;
DELETE 1
restaurants=# SELECT * FROM employees;
 designation | staff_id | salary |  name  | managerid | restaurantid
-------------+----------+--------+--------+-----------+--------------
 Cook        |        2 |  10000 | Harita |         1 |            2
 Cook        |        3 |  50000 | Haria  |         2 |            3
 Cook        |        4 |  70000 | Arita  |         1 |            4
 Cook        |        5 |  90000 | Hart   |         2 |            5
 Cook        |        6 |  15000 | Rita   |         1 |            6
(5 rows)
```

2. Complex Queries:

1. SELECT name, designation FROM employees WHERE salary>(SELECT avg(salary) FROM employees);

```
restaurants=# SELECT name, designation FROM employees WHERE salary >
restaurants-# (SELECT avg(salary) FROM employees);
 name  | designation
-------+-------------
 Haria | Cook
 Arita | Cook
 Hart  | Cook
(3 rows)
```

2. SELECT * FROM restaurants WHERE rating > (SELECT avg(rating) FROM restaurants) GROUP BY registrationnumber;

```
restaurants=# SELECT * FROM restaurants WHERE rating >( SELECT avg(rating) FROM restaurants) GROUP BY registrationnumber;
 registrationnumber | contactno  |        address         |    website    |   name    | rating | managerid | menuid
--------------------+------------+------------------------+---------------+-----------+--------+-----------+--------
                  1 | 9876543210 | Indiranagar, Bengaluru | mcdonalds.com | McDonalds |      4 |         2 |      2
                  5 | 9976543310 | Hennur, Bengaluru      | pizzahut.com  | Pizza Hut |      4 |         2 |      1
                  2 | 9976543210 | Koramangala, Bengaluru | kfc.com       | KFC       |      5 |         1 |      2
(3 rows)
```

3. UPDATE employees SET salary=1.15*salary WHERE managerid=2;

```
restaurants=# UPDATE employees SET salary=1.15*salary WHERE managerid=2;
UPDATE 3
restaurants=# SELECT * FROM employees;
 designation | staff_id | salary |  name  | managerid | restaurantid
-------------+----------+--------+--------+-----------+--------------
 Cook        |        2 |  10000 | Harita |         1 |            2
 Cook        |        4 |  70000 | Arita  |         1 |            4
 Cook        |        6 |  15000 | Rita   |         1 |            6
 Cook        |        1 |  23000 | Hari   |         2 |            1
 Cook        |        3 |  57500 | Haria  |         2 |            3
 Cook        |        5 | 103500 | Hart   |         2 |            5
(6 rows)
```

4. SELECT e.name from employees e, restaurants r WHERE(e.restaurantid=r.registrationnumber and r.rating=(SELECT min(rating) FROM restaurants));

```
restaurants=# SELECT e.name from employees e, restaurants r WHERE(e.restauranti
d=r.registrationnumber and  r.rating =
  (SELECT min(rating) FROM restaurants));
 name
------
 Rita
(1 row)
```

5. SELECT name, avg(rating) FROM restaurants group by name;

```
restaurants=# SELECT name, avg(rating) FROM restaurants group by name;
     name     | avg
--------------+-----
 KFC          |   4
 Pizza Hut    |   4
 McDonalds    |   4
 Dominos      |   1
 Leon Grill   |   3
 Burger King  |   2
(6 rows)
```

3. Create multiple users with different access privilege levels for different parts of the database.

1. CREATE USER Aliyah WITH ENCRYPTED PASSWORD ‘Aliyah123’;
   GRANT ALL PRIVILEGES ON DATABASE restaurants TO Aliyah;

```
restaurants=# CREATE USER Aliyah WITH ENCRYPTED PASSWORD 'Aliyah123';
CREATE ROLE
restaurants=# GRANT ALL PRIVILEGES ON DATABASE restaurants TO Aliyah;
GRANT
```

2. CREATE USER Anwesha WITH ENCRYPTED PASSWORD ‘Anwesha123’;
   GRANT UPDATE ON restaurants TO Anwesha;

```
restaurants=# GRANT UPDATE ON restaurants TO Anwesha;
GRANT
restaurants=#
```

3. CREATE USER Arunav WITH ENCRYPTED PASSWORD ‘Arunav123’;
   GRANT SELECT, INSERT, UPDATE ON employees TO Arunav;

```
restaurants=# GRANT SELECT, INSERT, UPDATE ON employees TO Arunav;
GRANT
restaurants=#
```

4. Granting and dropping more roles and privileges

```
restaurants=# GRANT INSERT ON ALL TABLES IN SCHEMA PUBLIC TO Aliyah;
GRANT
restaurants=# GRANT DELETE ON ALL TABLES IN SCHEMA PUBLIC TO Aliyah;
GRANT
restaurants=# ALTER USER Anwesha LOGIN;
ALTER ROLE
restaurants=# ALTER USER Anwesha CREATEROLE;
ALTER ROLE
restaurants=# ALTER USER Anwesha CREATEDB;
ALTER ROLE
restaurants=# ALTER USER Arunav CREATEDB;
ALTER ROLE
restaurants=# GRANT CONNECT ON DATABASE restaurants to Anwesha;
GRANT
restaurants=# GRANT INSERT ON employees TO Arunav;
GRANT
restaurants=# GRANT ALL PRIVILEGES ON manager to Arunav;
GRANT
```

```
restaurants=# \du
                                   List of roles
 Role name |                         Attributes                         | Member of
-----------+------------------------------------------------------------+-----------
 Ayesha    | Superuser, Create role, Create DB                          | {}
 aliyah    |                                                            | {}
 anwesha   | Create role, Create DB                                     | {}
 arunav    | Create DB                                                  | {}
 disha     | Cannot login                                               | {}
 postgres  | Superuser, Create role, Create DB, Replication, Bypass RLS | {}

restaurants=# DROP ROLE Disha;
DROP ROLE
restaurants=# \du
                                   List of roles
 Role name |                         Attributes                         | Member of
-----------+------------------------------------------------------------+-----------
 Ayesha    | Superuser, Create role, Create DB                          | {}
 aliyah    |                                                            | {}
 anwesha   | Create role, Create DB                                     | {}
 arunav    | Create DB                                                  | {}
 postgres  | Superuser, Create role, Create DB, Replication, Bypass RLS | {}
```

4. Four triggers with proper functions

1. Restaurant name cannot be NULL

```
[restaurants=# CREATE FUNCTION emp_stamp() RETURNS trigger as $emp_stamp$
[restaurants$# BEGIN
[restaurants$# IF NEW.Name IS NULL THEN RAISE EXCEPTION 'Name cannot be null';
[restaurants$# END IF;
[restaurants$# RETURN NEW;
[restaurants$# END;
[restaurants$# $emp_stamp$ LANGUAGE plpgsql;
CREATE FUNCTION
```

```
[restaurants=# CREATE TRIGGER emp_stamp BEFORE INSERT OR UPDATE ON restaurants FOR EACH ROW EXECUTE
 FUNCTION emp_stamp();
;
CREATE TRIGGER
```

```
[restaurants=# INSERT INTO restaurants values(7, '9838383834', 'Indiranagar, Bengauluru', 'kfc.com'
,NULL,5,1,2);
ERROR:  Name cannot be null
CONTEXT:  PL/pgSQL function emp_stamp() line 3 at RAISE
```

2. This example trigger simply raises a NOTICE message each time a supported command is executed.

```
[restaurants=# CREATE OR REPLACE FUNCTION snitch() RETURNS event_trigger AS $$
[restaurants$# BEGIN
[restaurants$# RAISE NOTICE 'snitch: % %',tg_event, tg_tag;
[restaurants$# END;
[restaurants$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
```

```
[restaurants=# CREATE EVENT TRIGGER snitch ON ddl_command_start EXECUTE FUNCTION snitch();
CREATE EVENT TRIGGER
```

```
[restaurants=# CREATE TABLE temp(tempid int);
NOTICE:  snitch: ddl_command_start CREATE TABLE
CREATE TABLE
[restaurants=# DROP TABLE temp;
NOTICE:  snitch: ddl_command_start DROP TABLE
DROP TABLE
```

3. This trigger ensures that any insert, update or delete of a row in the employees table is recorded (i.e., audited) in the emp_audit table. The current time and user name are stamped into the row, together with the type of operation, name and salary of the employee in the row in which it is performed.

```
restaurants=# CREATE OR REPLACE FUNCTION process_emp_audit() RETURNS TRIGGER AS $emp_audit$
BEGIN
IF (TG_OP= 'DELETE') THEN
INSERT INTO emp_audit SELECT 'D', now(), user,OLD.name, OLD.salary;
ELSEIF (TG_OP='UPDATE') THEN
INSERT INTO emp_audit SELECT 'U',now(),user, OLD.name, OLD.salary;
ELSEIF(TG_OP='INSERT')
THEN INSERT INTO emp_audit SELECT 'I',now(),user,NEW.name, NEW.salary;
END IF;
RETURN NULL;
END;
$emp_audit$ LANGUAGE plpgsql;
CREATE FUNCTION
```

```
restaurants=# CREATE TRIGGER emp_audit
restaurants-# AFTER INSERT OR UPDATE OR DELETE ON employees
restaurants-# FOR EACH ROW EXECUTE FUNCTION process_emp_audit();
CREATE TRIGGER
```

```
restaurants=# INSERT INTO employees VALUES ('Waitress', 7, 15000, 'Tia', 1, 6);
INSERT 0 1
```

```
restaurants=# select * from emp_audit;
 operation |           stamp            | userid | empname | salary
-----------+----------------------------+--------+---------+--------
 I         | 2021-11-07 21:30:30.794258 | Ayesha | Tia     |  15000
(1 row)

restaurants=# select * from employees;
 designation | staff_id | salary |  name   | managerid | restaurantid
-------------+----------+--------+---------+-----------+--------------
 Cook        |        2 |  10000 | Harita |         1 |            2
 Cook        |        4 |  70000 | Arita   |         1 |            4
 Cook        |        6 |  15000 | Rita    |         1 |            6
 Cook        |        1 |  23000 | Hari    |         2 |            1
 Cook        |        3 |  57500 | Haria   |         2 |            3
 Cook        |        5 | 103500 | Hart    |         2 |            5
 Waitress    |        7 |  15000 | Tia     |         1 |            6
(7 rows)
```

4. Trigger to ensure employee salary is not inserted as or updated to less than or equal to 0.

```
restaurants=# CREATE FUNCTION salary_stamp() RETURNS trigger as $salary_stamp$
BEGIN
IF NEW.salary<=0 THEN RAISE EXCEPTION 'Please enter a valid salary';
END IF;
RETURN NEW;
END;
$salary_stamp$ LANGUAGE plpgsql;
CREATE FUNCTION
restaurants=# CREATE TRIGGER salary_stamp
AFTER INSERT OR UPDATE ON employees
FOR EACH ROW EXECUTE FUNCTION salary_stamp();
CREATE TRIGGER
```

```
restaurants=# UPDATE employees SET salary=0 WHERE name='Harita';
ERROR:  Please enter a valid salary
CONTEXT:  PL/pgSQL function salary_stamp() line 3 at RAISE
```

5. Two function with usage of cursors

1. Function to get the names of all employees

```
restaurants=# CREATE OR REPLACE FUNCTION get_employees_name()
RETURNS text as $$
DECLARE
names text default '';
new_name record;
emp_deets cursor
FOR SELECT name
FROM employees;
BEGIN
open emp_deets;
loop
fetch emp_deets into new_name;
exit when not found;
names:=names||', '||new_name.name;
end loop;
close emp_deets;
return names;
end;
$$
language 'plpgsql';
NOTICE:  snitch: ddl_command_start CREATE FUNCTION
CREATE FUNCTION


restaurants=# select get_employees_name();
              get_employees_name
-----------------------------------------------
 , Harita, Arita, Rita, Hari, Haria, Hart, Tia
(1 row)
```

2. Function to get the details (name and address) of all the restaurants

```
restaurants=# CREATE OR REPLACE FUNCTION get_restaurant_details()
restaurants-# RETURNS text as $$
restaurants$# DECLARE
restaurants$# details text default '';
restaurants$# new_rest record;
restaurants$# rest_deets cursor;
restaurants$# for select name, address
restaurants$# from restaurants;
restaurants$# begin
restaurants$# open rest_deets;
restaurants$# loop
restaurants$# fetch rest_deets into new_rest;
restaurants$# exit when not found;
restaurants$# details:=details||', '||new_rest.name||':'||new_rest.address;
restaurants$# end loop;
restaurants$# close rest_deets;
restaurants$# return details;
restaurants$# end;
restaurants$# $$
restaurants-# language 'plpgsql';
NOTICE:  snitch: ddl_command_start CREATE FUNCTION


restaurants=# SELECT get_restaurant_details();
```

```
 , McDonalds:Indiranagar, Bengaluru, Leon Grill:Indiranagar, Bengaluru, Burger King:
Koramangala, Bengaluru, Pizza Hut:Hennur, Bengaluru, Dominos:Electronic City, Bengal
uru, KFC:Koramangala, Bengaluru
(1 row)
```

Contributions of team members and overall time taken:

- Each member contributed equally towards every aspect of this project.
- Our team took approximately 2 and a half hours to complete this assignment.