GROCERY MANAGEMENT SYSTEM

Problem Statement: Build a Model of Grocery Management chain where enormous amounts of data are collected and that need to be stored efficiently and suitable frontend for the ease of use.

Task 1:
Writing SQL Queries

- Simple
- Complex
- Nested

Code: (Queries)

```
\c grocery

/*display name and email from customer*/
select name || ' ' || email from customer;

/*display the items whose price >400 and taxable*/
select * from items where price > 400 AND taxable = 't' AND
qty > 5;

/*display the total expenses,income and tax in finance*/
```

```
select sum(finance.expense) as expenses,sum(finance.income) as
income,sum(finance.tax) as tax from finance;

/*displaying items with rating less then 5*/
select distinct item_name
from item as i,feedback as f
where i.item_id=f.item_id and rating<5;

/*delete feedback whose feedback_id is 7*/
delete * from feedback where (feedback_id='7' and cust_id in
(select cust_id from feedback where item_id ='7'));
select * from feedback where feedback_id ='7';

/*delete a row and return the deleted row */
delete * from feedback where item_id = '5' returning *;


/*update */
update items set description ='gava,apple,mango' where item_id
= '1';
select * from items where item_id ='1';

/*like. count the number of stores in bangalore*/
select count(store_id) from store where address like
'%Bangalore';

/* display the store_id and address which in are in bangalore
using like*/
select store_id,address from store where address like
'%Bangalore';

/*select the details of suppliers whose name starts with S */
```

```sql
select supplier_id,name,phone from supplier where name
like='S%'

/*Order by*/
select item_name,price from items order by price;
select item_name,price,weight from items order by weight desc;


/*Nested  details of the supplier whose store id=8*/
select name,phone from supplier  where supplier_id=(select
supplier_id from supplies where store_id=8);

/* details of the supplier who supplies goods to store in
Chikballapur*/
select name,phone from supplier where supplier_id=
(select supplier_id from supplies where store_id=
(select store_id from store where address like
'%Chikkaballapur'));

/*find the feedback and rating for the items that are
cancelled  with minimum price*/
select rating,cust_feedback from feedback where item_id=
(select item_id from cancellation where price=
(select min(price) from cancellation));

/*intersect*/
/*items with rating less than 5 and cust_feedback negative*/
select item_id from feedback where rating<5
intersect
select item_id from feedback where cust_feedback like '%Bad'
or cust_feedback like '%horrible%' or cust_feedback like '%not
worth';
```

```sql
/*minus*/
/*item id of the product with positive reviews = all items -
items with negative customer feedback*/
select item_id from feedback
minus
select item_id from feedback where cust_feedback like '%Bad%'
or  cust_feedback like '%not worth%' or cust_feedback like
'%horrible%';



/*union*/
/*items with no tax and price les than 600*/
select item_name from items where taxable='f'
union
select item_name from items where price<600;
```

Outputs:

```
 Tarun tbacup0@imdb.com
 Narine nstpaul1@goodreads.com
 Shubhman gvaz2@skype.com
 Anatola aagutter3@etsy.com
 Philis pfranzetti4@blog.com
 Gabriel gradbourn5@addthis.com
 Madan mdeppe6@parallels.com
 Barbi btoffolo7@yellowbook.com
 Kamlesh kritmeyer8@tumblr.com
 Suraj cfilinkov9@issuu.com
(10 rows)
```

```
 item_id | qty | item_name |   description   | taxable | price | weight | store_id | checkout_id
---------+-----+-----------+-----------------+---------+-------+--------+----------+-------------
       3 |  10 | Egg       | big and healthy | t       |   945 |      8 |        3 |           3
       8 |   6 | Oil       | Healthy         | t       |   873 |     21 |        8 |           8
       9 |   8 | Coconut   | Fresh,cheap     | t       |   966 |     76 |        9 |           9
(3 rows)
```

```
 expenses | income | tax
----------+--------+------
   316230 | 368000 | 1156
(1 row)
```

```
             ^
 feedback_id | rating | cust_feedback | cust_id | item_id
-------------+--------+---------------+---------+---------
           7 |    1.4 | Bad           |       7 |       7
(1 row)
```

```
UPDATE 1
 item_id | qty | item_name |   description    | taxable | price | weight | store_id | checkout_id
---------+-----+-----------+------------------+---------+-------+--------+----------+-------------
       1 |   6 | Fruits    | gava,apple,mango | f       |   599 |     75 |        1 |           1
(1 row)
```

```
 count
-------
     3
(1 row)
```

```
 store_id |                  address
----------+-----------------------------------------
        1 | 100ft Ring Road, BSK 3rd stage, Bangalore
        5 | 6486 Oakridge Center,Bangalore
        8 | Suttagalli, BSK 3rd stage, Bangalore
(3 rows)
```

```
  item_name   | price | weight
--------------+-------+--------
 Vegetables   |   679 |     88
 Noodles      |   252 |     85
 Cereals      |   970 |     78
 Coconut      |   966 |     76
 Fruits       |   599 |     75
 Bottles      |   128 |     51
 Oil          |   873 |     21
 Nandini Milk |   314 |     10
 Egg          |   945 |      8
 Bread        |   412 |      5
(10 rows)
```

```
 name |   phone
------+------------
 Anil | 1185520362
(1 row)

   name   |   phone
----------+------------
 hanamant | 9933844827
(1 row)

 rating | cust_feedback
--------+---------------
    3.0 | Normal
(1 row)

 item_id
---------
       7
       3
       5
(3 rows)
```

```
  item_name
--------------
 Fruits
 Nandini Milk
 Bottles
 Bread
 Noodles
(5 rows)

grocery=#
```

Task 2

Multiple users with different access privilege levels for different parts of the database should be created.

users.sql | our queries.sql

Users > ishu > Desktop > users.sql

```
1  create user manager with password 'manager@123';
2  grant select,insert,update,delete on all tables in schema public to manager;
3  revoke update on finance from manager;
4  -- create check1 with password "check1";
5
6  create user checkout1 with password 'checkout1';
7  grant select,insert,update,delete  on table checkout,cancellation to checkout1;
8
9
10 create user checkout2 with password 'checkout2';
11 grant select,insert,update,delete on table checkout,cancellation to checkout2;
12
13 create user checkout3 with password 'checkout3';
14 grant select,insert,update,delete  on table checkout,cancellation to checkout3;
15
16
17 create user user1 with password 'user1';
18 grant update,select on table feedback to user1;
```

List of roles

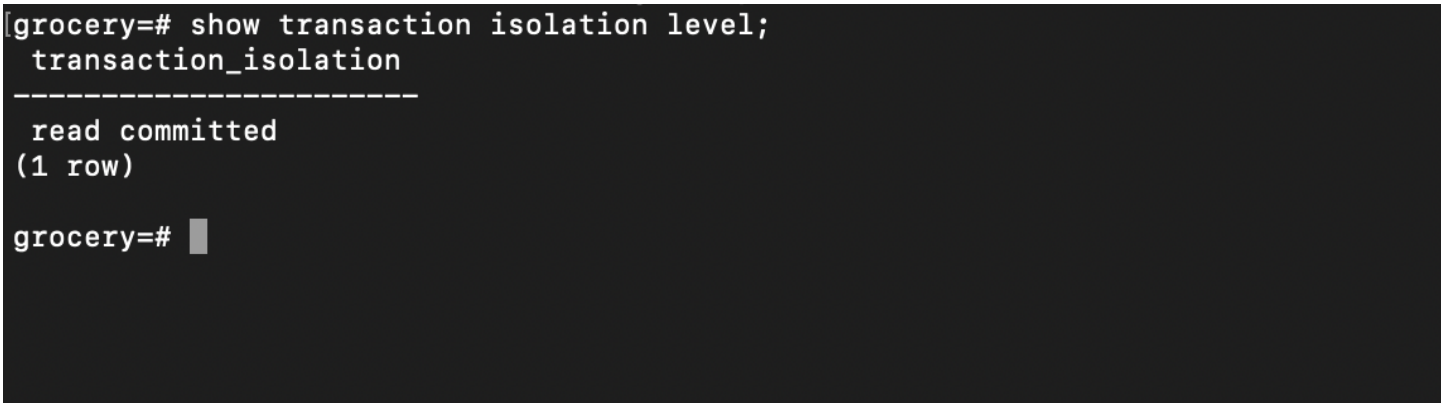
Role name	Attributes	Member of
checkout1		{}
checkout2		{}
checkout3		{}
ishu	Superuser, Create role, Create DB	{}
manager		{}
postgres	Superuser, Create role, Create DB, Replication, Bypass RLS	{}
user1		{}

```
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
grocery=# \i /Users/ishu/Desktop/users.sql;
psql:/Users/ishu/Desktop/users.sql:1: ERROR:  role "manager" already exists
GRANT
REVOKE
psql:/Users/ishu/Desktop/users.sql:6: ERROR:  role "checkout1" already exists
GRANT
psql:/Users/ishu/Desktop/users.sql:10: ERROR:  role "checkout2" already exists
GRANT
psql:/Users/ishu/Desktop/users.sql:13: ERROR:  role "checkout3" already exists
GRANT
psql:/Users/ishu/Desktop/users.sql:17: ERROR:  role "user1" already exists
GRANT
grocery=# \du
                             List of roles
 Role name |                         Attributes                         | Member of
-----------+------------------------------------------------------------+-----------
 checkout1 |                                                            | {}
 checkout2 |                                                            | {}
 checkout3 |                                                            | {}
 ishu      | Superuser, Create role, Create DB                          | {}
 manager   |                                                            | {}
 postgres  | Superuser, Create role, Create DB, Replication, Bypass RLS | {}
 user1     |                                                            | {}

grocery=# \du+
                                   List of roles
 Role name |                         Attributes                         | Member of | Description
-----------+------------------------------------------------------------+-----------+-------------
 checkout1 |                                                            | {}        |
 checkout2 |                                                            | {}        |
 checkout3 |                                                            | {}        |
 ishu      | Superuser, Create role, Create DB                          | {}        |
 manager   |                                                            | {}        |
 postgres  | Superuser, Create role, Create DB, Replication, Bypass RLS | {}        |
 user1     |                                                            | {}        |

grocery=# select * from pg_catalog.pg_user;
  usename  | usesysid | usecreatedb | usesuper | userepl | usebypassrls |  passwd  | valuntil | useconfig
-----------+----------+-------------+----------+---------+--------------+----------+----------+-----------
 postgres  |       10 | t           | t        | t       | t            | ******** |          |
 ishu      |    16384 | t           | t        | f       | f            | ******** |          |
 manager   |    19008 | f           | f        | f       | f            | ******** |          |
 checkout1 |    19009 | f           | f        | f       | f            | ******** |          |
 checkout2 |    19010 | f           | f        | f       | f            | ******** |          |
 checkout3 |    19011 | f           | f        | f       | f            | ******** |          |
 user1     |    19012 | f           | f        | f       | f            | ******** |          |
(7 rows)

grocery=#
```

Initiate concurrent Transactions and demonstrate the concurrency control for the conflicting actions.

In the Grocery management project 'READ COMMITTED' Isolation level is applied as queries results should be the values which are properly updated or committed.This isolation level guarantees that any data read is committed at the moment it is read. Thus it does not allow dirty read. The transaction holds a read or write lock on the current row, and thus prevents other transactions from reading, updating or deleting it.

By default Postgres imposed READ COMMITTED Isolation level ,Hence changes are not required.



Contributions

PES1UG19CS174	Guruprasad B N	SQL Queries + Task2
PES1UG19CS176	H V Shashikant Reddy	SQL Queries + Task2
PES1UG19CS191	Ishwar Sitarama Joshi	SQL Queries + Task2