TITLE : Online Food Odering System

TEAM N0 - 10 :

Deliwala Aneri - 201912037

Nipun Modi - 201912038

Shubham Agarwal - 201912039

Tarun - 201912040

FUNCTIONAL REQUIREMENTS

- ✓ List the shop who serves all the types of beverages.
- ✓ List the details of the shops and the customers who have served the customers older than 25 years.
- ✓ List of all the customers who made payment through COD
- ✓ Retrieve the details of the employee who have the salary greater than the average salary of all the employees of that department.
- ✓ Select all the customers and the total payments done by them greater than 500.
- ✓ Retrieve the names of the top 3 shops on the basis of number of orders received by them.
- ✓ Select the food item that are in demand in a [particular week of particular month]
- ✓ Retrieve the details of the employee who have their issue status as pending.
- ✓ Count the total number of employees in each department
- ✓ List the food items and their count that how many times it has been ordered
- ✓ List the employees who got the rating greater than 4
- ✓ List the customers who have not made any order till now .
- ✓ List of shops who served all the food items.
- ✓ Retrieve the details of the employee who have the salary greater than the average salary of all the employees.
- ✓ Retrieve the details of the customer who ordered in a [particular month].
- ✓ Retrieve the details of the shops who have served the customer till now.
- ✓ Count the number of complaints made for each problem.
- ✓ Count the number of orders placed by each customer.
- ✓ Retrieve customer name, food name and the shop from which the customer had placed the order.

ENTITY RELATIONSHIP MODEL :

RELATIONAL MODEL :

SQL DDL STATEMENTS

Triggers Used :

1) When a new employee is added to the delivery department it will automatically be inserted in delivery staff table.

```
CREATE TRIGGER deilvery_staff_in
        AFTER INSERT ON public.employee
        FOR EACH ROW
EXECUTE PROCEDURE public.deilvery_staff_ins();

CREATE FUNCTION public.deilvery_staff_ins() RETURNS trigger
   LANGUAGE plpgsql
   AS $$begin
        if new.dep_id='DEP001' then
                insert into delivery_staff(emp_id)values(new.emp_id);
        end if;
        RETURN NULL;
   end;
$$;
```

2) When a customer places an order, its entry will appear in the order pending table until the payment of the order is confirmed and when the payment is confirmed the order status will changed to delivered.

```
CREATE TRIGGER order_pen_in
         AFTER INSERT ON public.orders
         FOR EACH ROW
 EXECUTE PROCEDURE public.order_pen_ins();

CREATE FUNCTION public.order_pen_ins() RETURNS trigger
   LANGUAGE plpgsql
   AS $$begin
        insert into order_pen values(new.order_id);
   RETURN NULL;
   end;
$$;
```

```
CREATE TRIGGER pay_pen_de
        BEFORE UPDATE ON public.payment
        FOR EACH ROW
EXECUTE PROCEDURE public.pay_pen_del();

CREATE FUNCTION public.pay_pen_del() RETURNS trigger
   LANGUAGE plpgsql
   AS $$begin
        if new.status='paid' then
                delete from pay_pen where pay_id=new.pay_id;
                update orders set status='delivered' where order_id=new.order_id;
        end if;
    RETURN new;
    end;
$$;
```

DDL STATEMENTS WITH SOME CONSTRAINTS AND CHECKS :

1) Contact must be only 10 digit number and any number greater than 5999999999 and less than or equal to 9999999999.

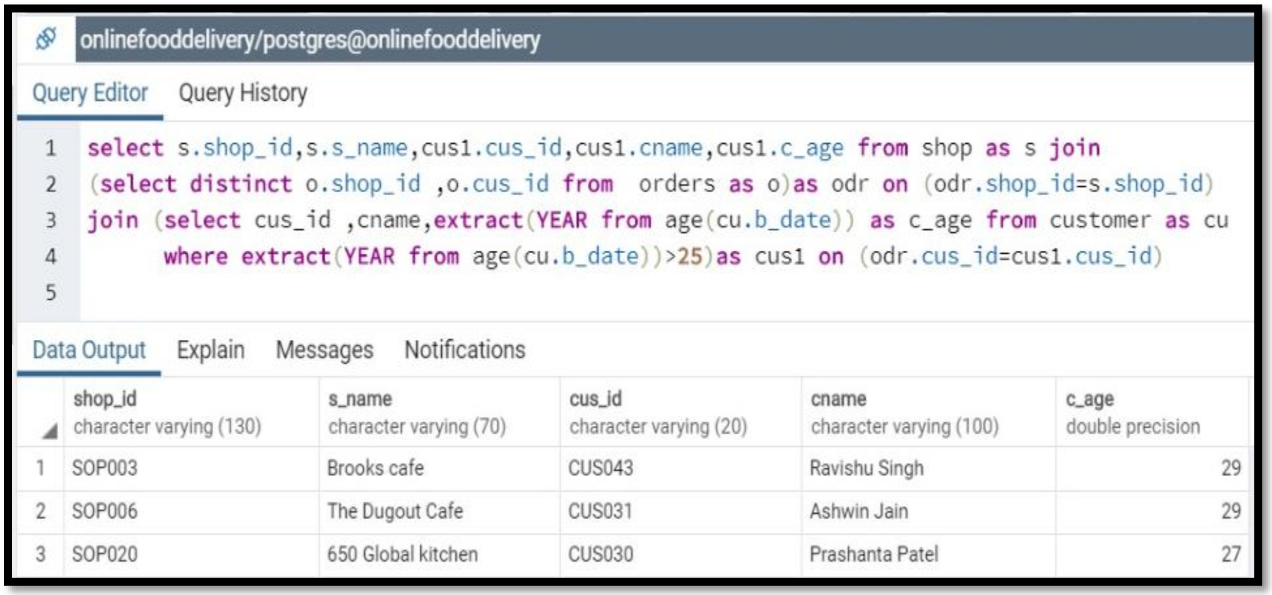
```
ALTER TABLE customer
        ADD CONSTRAINT contact_check
CHECK ( contact  > 5999999999 AND contact <= 9999999999) ;
```

2) Issue status must be either SOLVED or UNDER PROCESS and the problem should be from one of the following : FOOD, DELIVERY, PAYMENT

```
ALTER TABLE customer
        ADD CONSTRAINT issue_check CHECK (
                status in( 'SOLVED', 'UNDER PROCESS' ),
                problem in( 'FOOD',  'DELIVERY',  'PAYMENT' )
        )
```

TOP 5 FUNCTIONAL REQUIREMENTS, QUERIES SUPPORTING THESE REQUIREMENTS, AND OUTPUT CAPTURED FOR EACH OF THEM

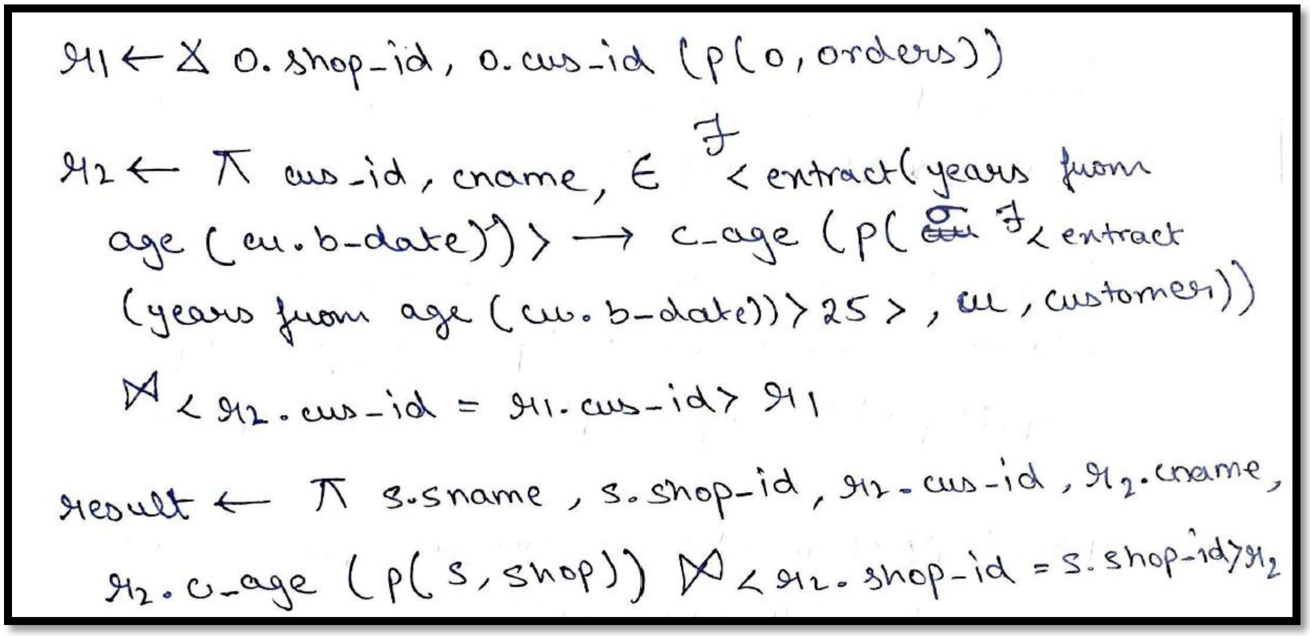
- List the details of the shops and the customers who have served the customers older than 25 years



```sql
select s.shop_id,s.s_name,cus1.cus_id,cus1.cname,cus1.c_age from shop as s join
(select distinct o.shop_id ,o.cus_id from  orders as o)as odr on (odr.shop_id=s.shop_id)
join (select cus_id ,cname,extract(YEAR from age(cu.b_date)) as c_age from customer as cu
     where extract(YEAR from age(cu.b_date))>25)as cus1 on (odr.cus_id=cus1.cus_id)

```

	shop_id character varying (130)	s_name character varying (70)	cus_id character varying (20)	cname character varying (100)	c_age double precision
1	SOP003	Brooks cafe	CUS043	Ravishu Singh	29
2	SOP006	The Dugout Cafe	CUS031	Ashwin Jain	29
3	SOP020	650 Global kitchen	CUS030	Prashanta Patel	27



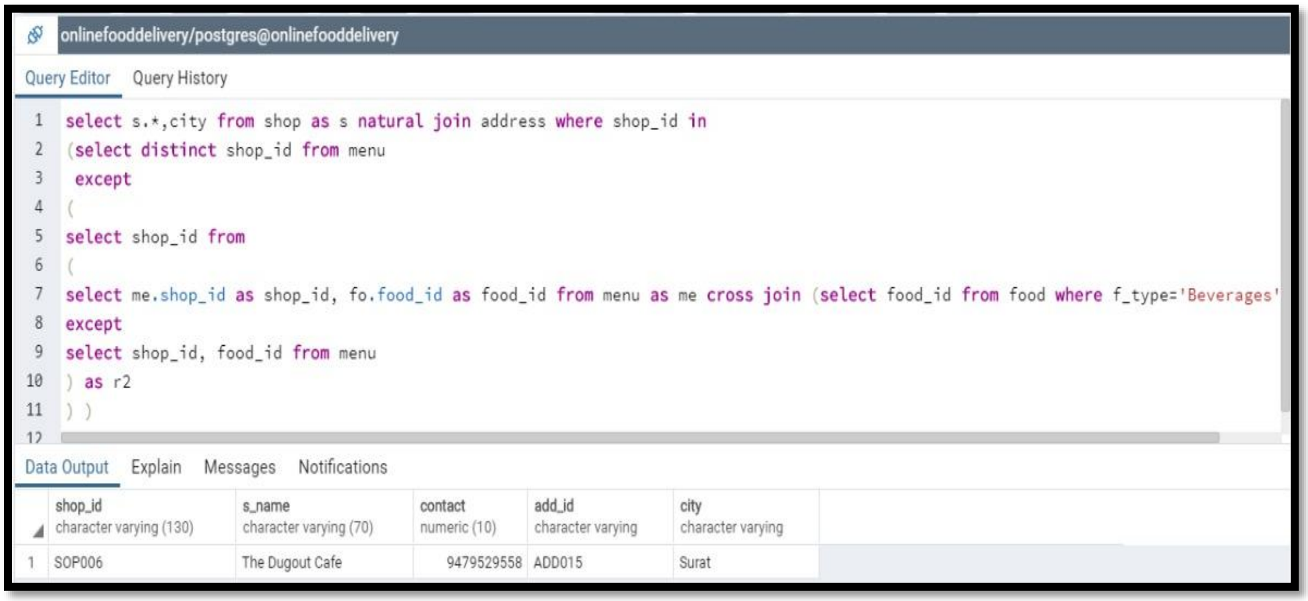
$r1 \leftarrow \delta$ o.shop-id, o.cus-id (ρ(o, orders))

$r2 \leftarrow \pi$ cus-id, cname, $E\ \mathcal{F}$ < entract(years from age (cu.b-date)) > $\longrightarrow$ c-age (ρ(σ ~~cu~~ $\mathcal{F}$ < entract (years from age (cu.b-date)) > 25 > , cu , customer))

$\bowtie$ < r2.cus-id = r1.cus-id > r1

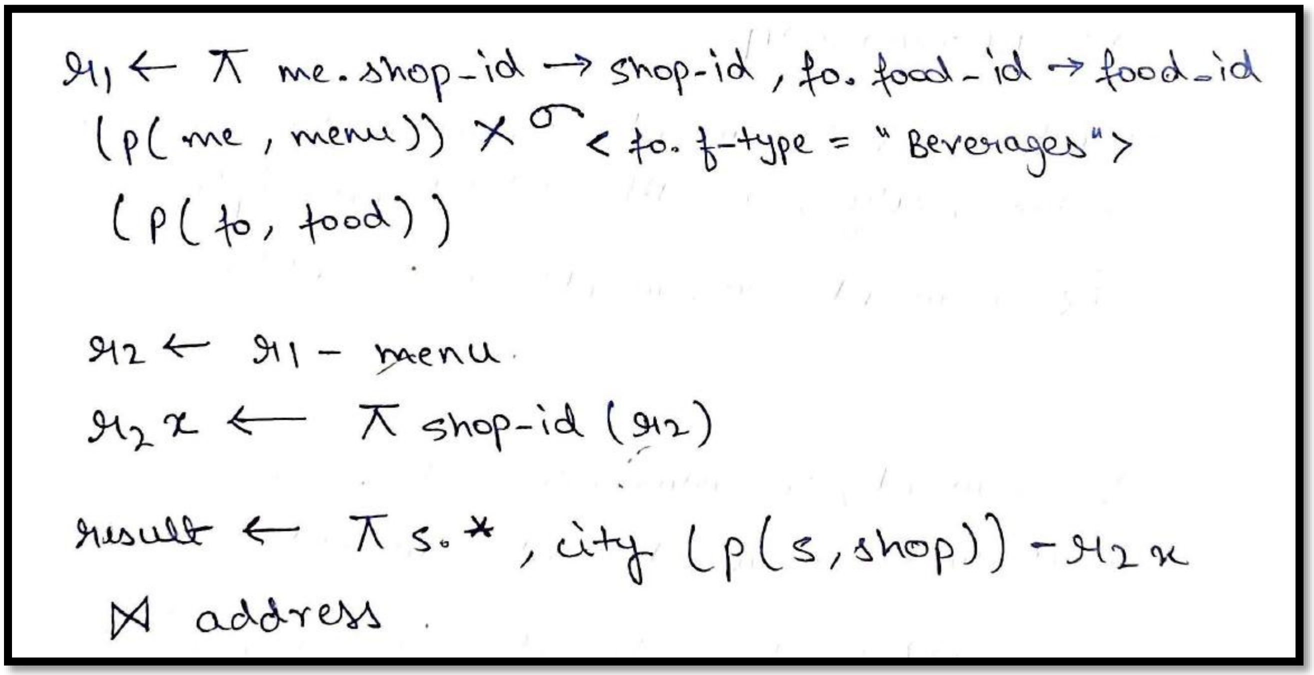
result $\leftarrow \pi$ s.sname , s.shop-id , r2.cus-id , r_2.cname, r_2.c-age (ρ(s, shop)) $\bowtie$ < r2.shop-id = s.shop-id > r_2

- List the shop who serves all the types of beverages.



```sql
select s.*,city from shop as s natural join address where shop_id in
(select distinct shop_id from menu
 except
(
select shop_id from
(
select me.shop_id as shop_id, fo.food_id as food_id from menu as me cross join (select food_id from food where f_type='Beverages'
except
select shop_id, food_id from menu
) as r2
) )
```

	shop_id character varying (130)	s_name character varying (70)	contact numeric (10)	add_id character varying	city character varying
1	SOP006	The Dugout Cafe	9479529558	ADD015	Surat



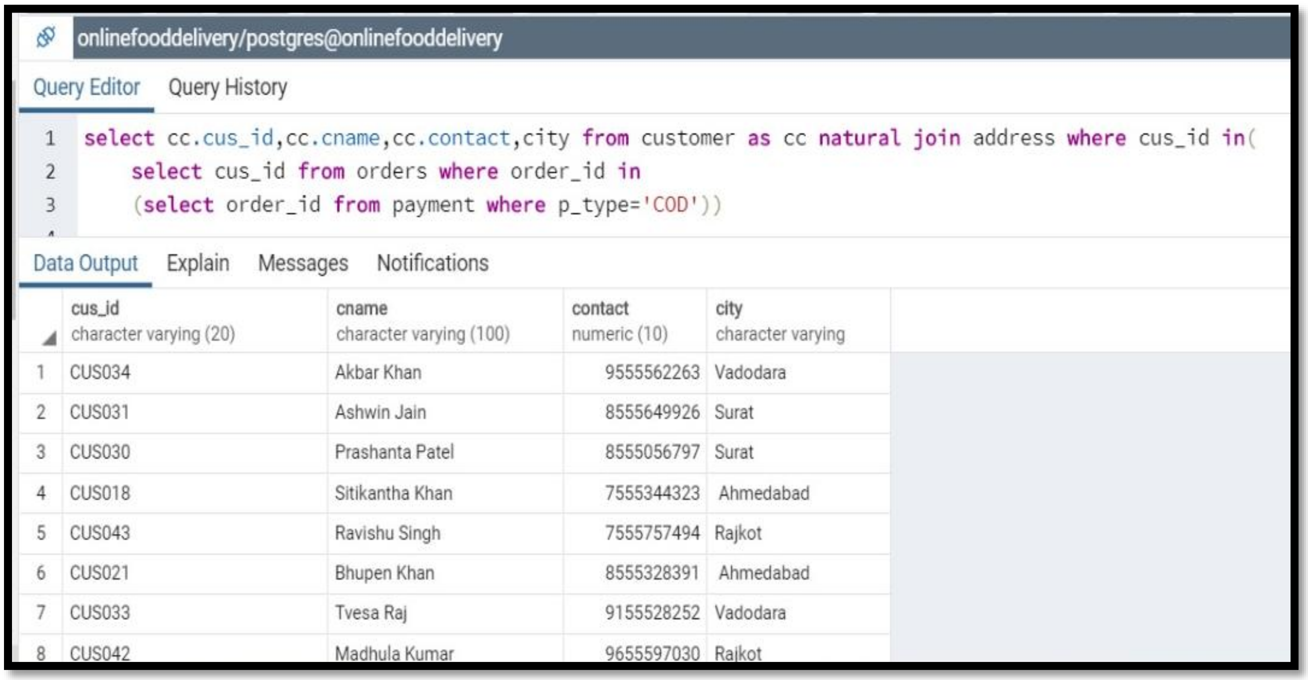
$r_1 \leftarrow \pi_{me.shop\text{-}id \rightarrow shop\text{-}id,\ fo.food\text{-}id \rightarrow food\text{-}id} (\rho(me, menu)) \times \sigma_{< fo.f\text{-}type = \text{"Beverages"} >} (\rho(fo, food))$.

$r_2 \leftarrow r_1 - menu$.

$r_2x \leftarrow \pi_{shop\text{-}id} (r_2)$

$result \leftarrow \pi_{s.*,\ city} (\rho(s, shop)) - r_2x \bowtie address$.

- List of all the customers who made payment through COD



onlinefooddelivery/postgres@onlinefooddelivery

Query Editor | Query History

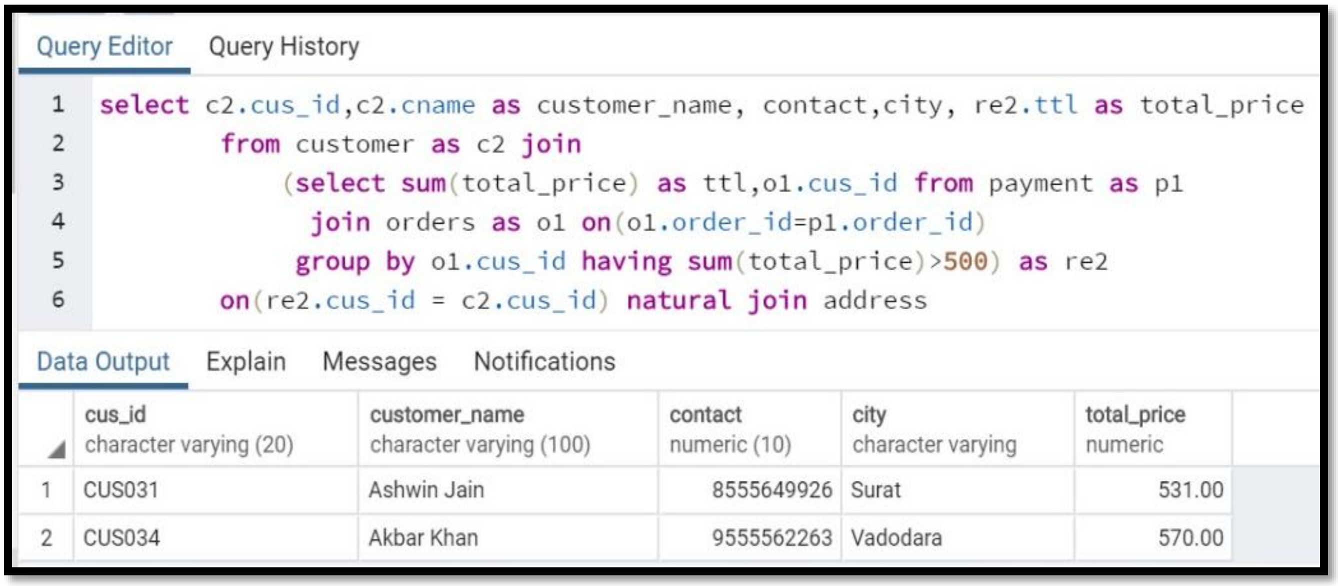
```
select cc.cus_id,cc.cname,cc.contact,city from customer as cc natural join address where cus_id in(
    select cus_id from orders where order_id in
    (select order_id from payment where p_type='COD'))
```

Data Output | Explain | Messages | Notifications

	cus_id character varying (20)	cname character varying (100)	contact numeric (10)	city character varying
1	CUS034	Akbar Khan	9555562263	Vadodara
2	CUS031	Ashwin Jain	8555649926	Surat
3	CUS030	Prashanta Patel	8555056797	Surat
4	CUS018	Sitikantha Khan	7555344323	Ahmedabad
5	CUS043	Ravishu Singh	7555757494	Rajkot
6	CUS021	Bhupen Khan	8555328391	Ahmedabad
7	CUS033	Tvesa Raj	9155528252	Vadodara
8	CUS042	Madhula Kumar	9655597030	Rajkot

π cc.cus_id , cc.cname , cc.contact , city (ρ(cc, customer))
SEMIJOIN< cc.cus_id = order.cus_id > orders
SEMIJOIN < order.order_id = payment.~~payment~~ order_id >
(σ < ptype = "COD" > , payment) $\bowtie$ address

- Select all the customers and the total payments done by them more than 500.



Query Editor Query History

```sql
select c2.cus_id,c2.cname as customer_name, contact,city, re2.ttl as total_price
        from customer as c2 join
            (select sum(total_price) as ttl,o1.cus_id from payment as p1
              join orders as o1 on(o1.order_id=p1.order_id)
             group by o1.cus_id having sum(total_price)>500) as re2
        on(re2.cus_id = c2.cus_id) natural join address
```

Data Output Explain Messages Notifications

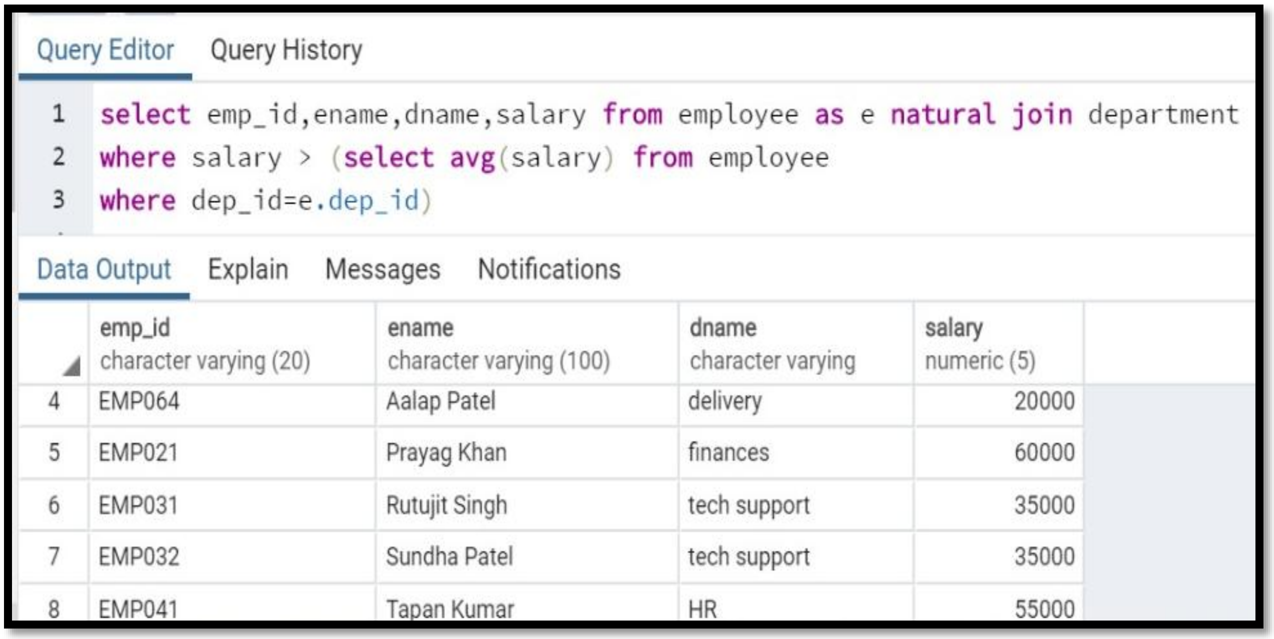
	cus_id character varying (20)	customer_name character varying (100)	contact numeric (10)	city character varying	total_price numeric
1	CUS031	Ashwin Jain	8555649926	Surat	531.00
2	CUS034	Akbar Khan	9555562263	Vadodara	570.00

$r1 \leftarrow \pi \; \rho_{<o1.cus\text{-}id>} \mathcal{F}_{<sum(total\_price)> \rightarrow ttl,}$
$o1.cus\text{-}id \; (\rho(p1, payment))$

$r2 \leftarrow (\rho(o1, orders)) \bowtie_{<o1.cus\text{-}id = r1.cus\text{-}id>}$
$\sigma_{<ttl > 500>} \; r1$

$result \leftarrow \pi \; c2.cus\text{-}id, \; c2.cname \rightarrow customer\_name,$
$contact, \; city, \; r2.ttl \rightarrow total\text{-}price \; (\rho(c2, customer))$
$\bowtie_{<c2.cus\text{-}id = r2.cus\text{-}id>} r2 \bowtie address$

- Retrieve the details of the employee who have the salary greater than the average salary of all the employees of that department



```sql
select emp_id,ename,dname,salary from employee as e natural join department
where salary > (select avg(salary) from employee
where dep_id=e.dep_id)
```

	emp_id character varying (20)	ename character varying (100)	dname character varying	salary numeric (5)
4	EMP064	Aalap Patel	delivery	20000
5	EMP021	Prayag Khan	finances	60000
6	EMP031	Rutujit Singh	tech support	35000
7	EMP032	Sundha Patel	tech support	35000
8	EMP041	Tapan Kumar	HR	55000

π emp-id , ename , dname , salary (employee) SEMI JOIN <salary > AVG (salary)> (ρ , σ <employee. dep-id = emp. (dep-id> (emp , employee)) $\bowtie$ (department)

CONCLUDING REMARKS

This software will help the customers to order food online from the shops that are located within their region (pin code) and report any issue is the customer is not satisfied with the order.

Our basic purpose task is to keep track of the information about all the food items provided by the shops and the customers. The main objective of the project is to use the data retrieved from the queries for the analysis purpose.

Current Assumption

- Only 4 cities of Gujarat are selected where the service is being provided namely (Surat, Rajkot, Ahmedabad, Vadodara) and within these cities some specific pin codes are selected.
- Department of the employees can be any one of them only marketing, Delivery, Finance, HR and Tech Support with different salaries accordingly.
- Each Delivery staff is allocated only one particular area for service.
- There are around 4 - 5 shops in a given area which offer different types of food.
- Food belong to any one of these category (main course, beverage, snacks, desert)
- Payment can be done either by COD or online mode.

Scope of Improvement

- Range of service area can be widen with more number of shops depending on the demand.
- Salary of the employee can be given depending on various factors like experience, department, qualification.
- Separate account of delivery staff can be made which will reflect their past and upcoming orders to be completed, their ratings, and complaints registered on their name if any.
- Food category can be increased and can be more specific.
- More payment options can be added like paytm, google pay, etc.
- Food delivery status can be added like, received , confirmed, prepared, on the way, etc.
- Can be made more user friendly through UI.