FINAL PROJECT

TOPIC: ONLINE MARKET: CAKE SHOP

AUTHOR: Damir Amankos/210107051

Description:

1. **"The System for Ordering" Database project:** **The prime motive of this database is to make the order transaction speedy and smooth for the customers, to maintain and secure the order records.The system is capable enough to book the cake as per the customer need and event.**

- **A customer can register to purchase an item. The customer will provide the account number and bank name.**
- **After registration, each customer will have a unique customer, user id, and password.**
- **A customer can purchase one or more items in different quantities.**

End users:

- **Casual End Users:**programmers of the company
- **Parametric end users:**Reservation clerks basically check availability for a given request,check whether the requested product is in stock
- **Parametric end users:** Clerks who are working at receiving end for company enter the product identifies via barcodes and descriptive information through buttons to update a central database

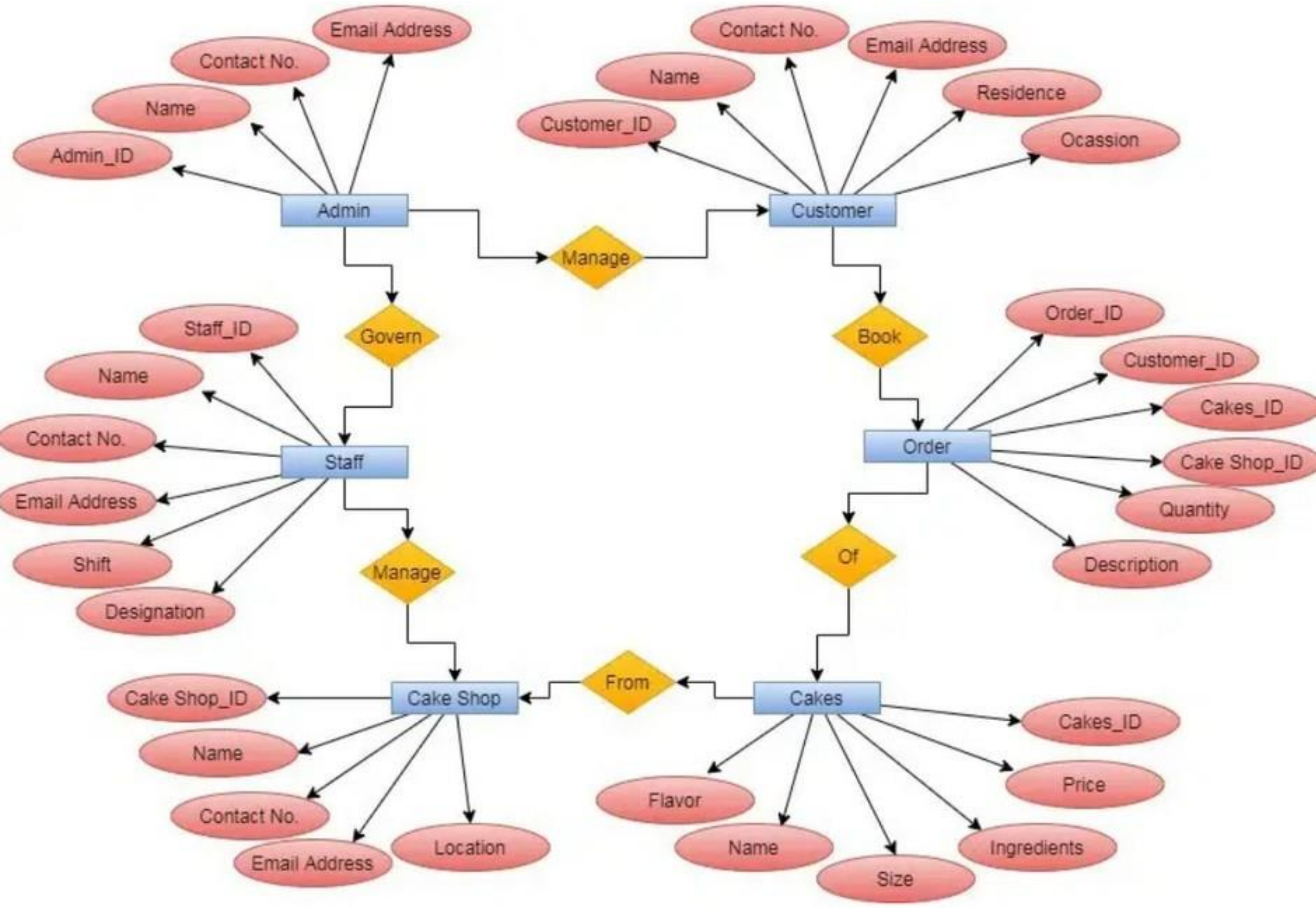
Data obsolescence:

- the best solution is to partition tables by the the **key** which differentiates stale from current data (such as **date, currency_id** or things like that).
- Setting up daily obsolescence for on any rows retrieved by **queries**

Project Idea:

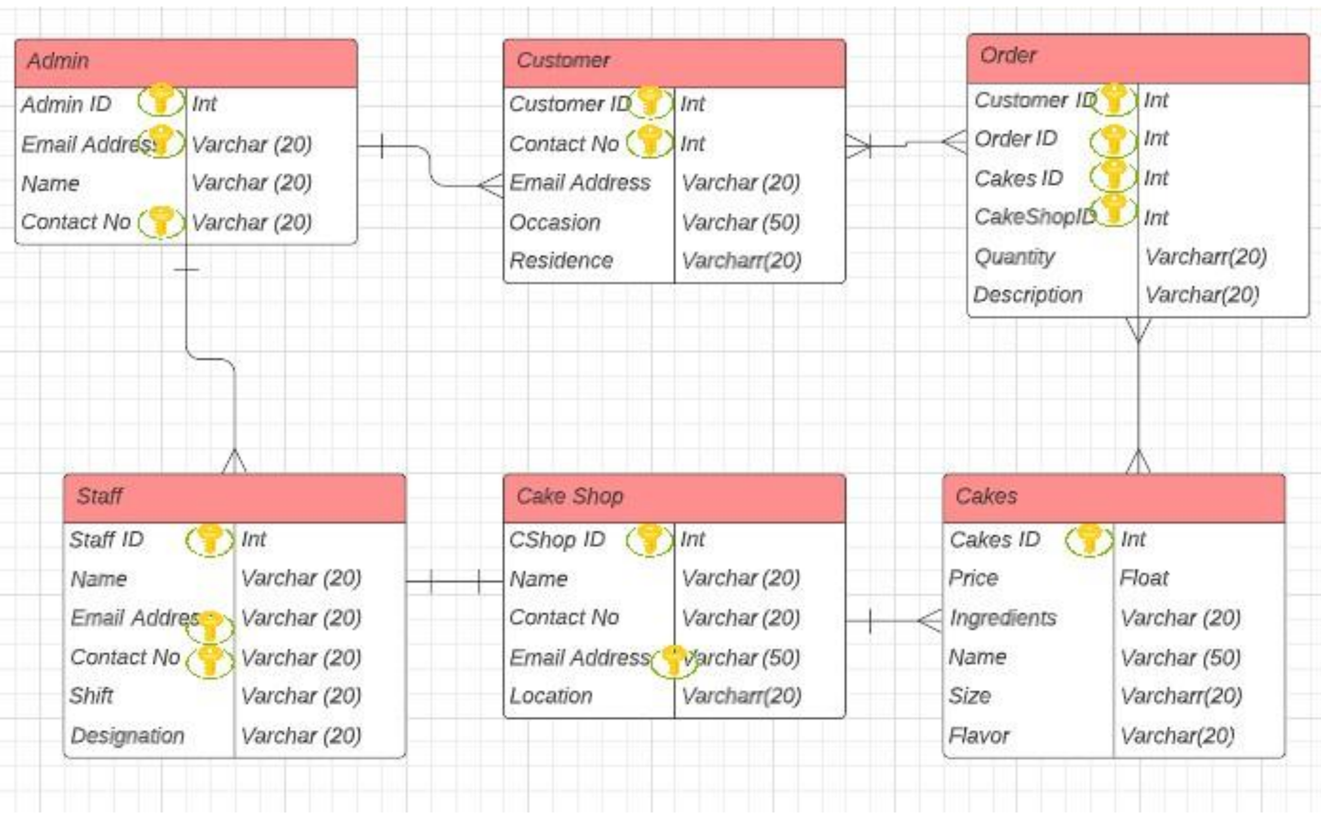
- The Database System increasing Walmart Sales
- Environment

Entity Relationship Diagram



Relationships:

- Admin-Customer: 1 to n
- Customer-Order: n to n
- Order-Cakes: n to n
- Cake shop-Cakes: 1 to n
- Cake shop-staff: 1 to 1
- Admin-staff: 1 to n



Admin	
Admin ID 🔑	Int
Email Address 🔑	Varchar (20)
Name	Varchar (20)
Contact No 🔑	Varchar (20)

Customer	
Customer ID 🔑	Int
Contact No 🔑	Int
Email Address	Varchar (20)
Occasion	Varchar (50)
Residence	Varcharr(20)

Order	
Customer ID 🔑	Int
Order ID 🔑	Int
Cakes ID 🔑	Int
CakeShopID 🔑	Int
Quantity	Varcharr(20)
Description	Varchar(20)

Staff	
Staff ID 🔑	Int
Name	Varchar (20)
Email Address 🔑	Varchar (20)
Contact No 🔑	Varchar (20)
Shift	Varchar (20)
Designation	Varchar (20)

Cake Shop	
CShop ID 🔑	Int
Name	Varchar (20)
Contact No	Varchar (20)
Email Address 🔑	Varchar (50)
Location	Varcharr(20)

Cakes	
Cakes ID 🔑	Int
Price	Float
Ingredients	Varchar (20)
Name	Varchar (50)
Size	Varcharr(20)
Flavor	Varchar(20)

Normalization

Functional dependencies:

1) CUSTOMER ID--->CONTACT NO, NAME, EMAIL ADDRESS, RESIDENCE, OCCASION
SUPERKEYS:(CUSTOMER_ID); (CUSTOMER_ID, CONTACT_NO); (CUSTOMER_ID,NAME); (CUSTOMER_ID,EMAIL)

2) ORDER ID--->CUSTOMER ID, CAKES ID, CAKE SHOP ID, QUANTITY, DESCRIPTION
SUPERKEYS:(ORDER_ID); (ORDER_ID, CUSTOMER_ID))

3) CAKES ID--->PRICE, INGREDIENTS, SIZE, NAME, FLAVOR
SIZE---> INGREDIENTS, PRICE $\longrightarrow$ NON-PRIME--->NON-PRIME, SO IT IS TRANSITIVE DEPENDENCY
SUPERKEYS:CAKES_ID

table 1

C_ID	Name	Flavor

table 2

Size	Price	Ingr

4) CAKE_SHOP_ID--->NAME, CONTACT_NO, LOCATION, EMAIL ADDRESS
NAME---> LOCATION
SUPERKEYS:(CAKE_SHOP ID); (CAKE SHOP ID, NAME))

table 1

Sh_ID	email	c_no

table 2

name	loctn

5) STUFF_ID--->NAME, CONTACT_NO, SHIFT, DESIGNATION, EMAIL ADDRESS
DESIGNATION---> SHIFT
SUPERKEYS:(STUFF ID); (STUFF ID, NAME); (STUFF_ID,CONTACT_NO)

table 1

S_ID	Name	Email

table 2

Dsgntn	Shift

6) ADMIN_ID--->NAME, CONTACT_NO, EMAIL ADDRESS
SUPERKEYS:(ADMIN ID); (ADMIN ID, NAME); (ADMIN_ID,CONTACT_NO)

Table

```sql
create table Customer (
 Customer_ID VARCHAR(50),
 Name VARCHAR(50),
 Contact_No VARCHAR(50),
 Email Address VARCHAR(50),
 Residence VARCHAR(50),
 Occasion VARCHAR(50)
);
insert into Customer (Customer_ID, Name, Contact_No, Email Address, Residence, Occasion) values ('223.144.94.79/25', 'Galvan Sea', '612-931-9772',
'gsea0@unblog.fr', '459 Crest Line Street', 'Action|Horror');
insert into Customer (Customer_ID, Name, Contact_No, Email Address, Residence, Occasion) values ('71.71.42.246/27', 'Danika Lockhart', '737-705-
6839', 'dlockhart1@elpais.com', '66 Almo Trail', 'Documentary');............................
```

```sql
create table Order (
 Order_ID VARCHAR(50),
 Customer_ID VARCHAR(50),
 Cakes_ID VARCHAR(50),
 Cake_Shop_ID VARCHAR(50),
 Quantity INT,
 Description VARCHAR(50)
);
insert into Customer (Order_ID, Customer_ID, Cakes_ID, Cake_Shop_ID, Quantity, Description) values ('7.170.171.212/14', '228.90.207.232/11',
'ac6:eba2:7966:3929:a8a3:937:a068:cd54/57', '2e10:5faa:afaa:f2ab:1416:d356:e56d:f204/47', 7, 'Horror|Thriller');
insert into Customer (Order_ID, Customer_ID, Cakes_ID, Cake_Shop_ID, Quantity, Description) values ('205.207.245.69/25', '200.190.210.1/16',
'f445:582d:6514:6335:8bd9:9b08:6f32:2aab/15', '6de6:60a3:579f:defa:77ec:91fb:e7a:33ce/80', 10, 'Comedy|Crime|Horror|Thriller');
```

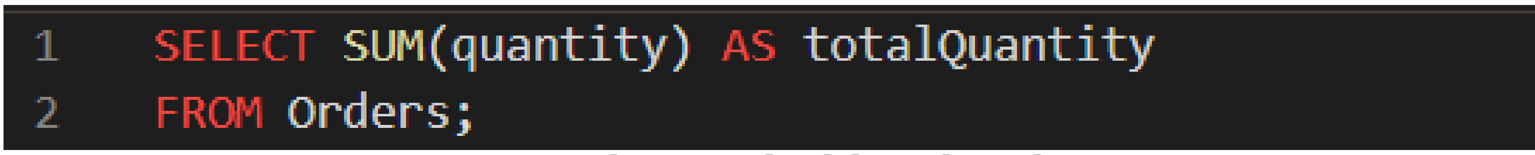
```sql
create table Cakes (
 Cakes_ID VARCHAR(50),
 Price VARCHAR(50),
 Ingredients TEXT,
 Size VARCHAR(50),
 Name VARCHAR(50),
 Flavor TEXT
);
insert into Customer (Cakes_ID, Price, Ingredients, Size, Name, Flavor) values ('77.231.31.99/12', '$15.05', 'augue', '2XL', 'Treeflex', 'augue quam
sollicitudin');
insert into Customer (Cakes_ID, Price, Ingredients, Size, Name, Flavor) values ('115.220.42.170/18', '$21.13', 'vitae consectetuer eget', 'M', 'Fintone', 'quis
tortor');
```

```sql
create table Cake Shop (
 Cake_Shop_ID VARCHAR(50),
 Contact_No VARCHAR(50),
 Email_Address VARCHAR(50),
 Location VARCHAR(50),
 Name VARCHAR(50)
);
insert into Cake Shop (Cake_Shop_ID, Contact_No, Email_Address, Location, Name) values ('120.161.59.175/23', '821-848-9157', 'abrinsford0@dagondesi
'496 Sage Point', null);
```

```sql
create table Staff (
 Staff_ID VARCHAR(50),
 Contact_No VARCHAR(50),
 Email_Address VARCHAR(50),
 Location VARCHAR(50),
 Shift VARCHAR(50),
 Designation VARCHAR(50)
);
insert into Staff (Staff_ID, Contact_No, Email_Address, Location, Shift, Designation) values ('83.16.155.219/14', '317-573-4507',
'sbethune0@eventbrite.com', '45 Ilene Trail', 'Services', 'Weekly');
```

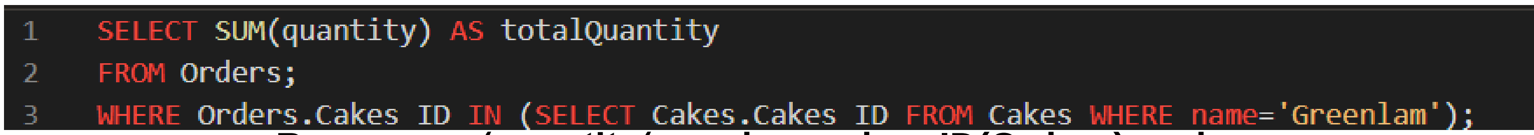
```sql
create table Admin (
 Admin_ID  VARCHAR(50),
 Contact_No VARCHAR(50),
 Email_Address VARCHAR(50),
 Name VARCHAR(50)
);
insert into Admin (Admin_ID, Contact_No, Email_Address, Name) values ('152.54.14.234/18', '983-398-7120', 'tkensitt0@virginia.edu',
```

Query MySQL



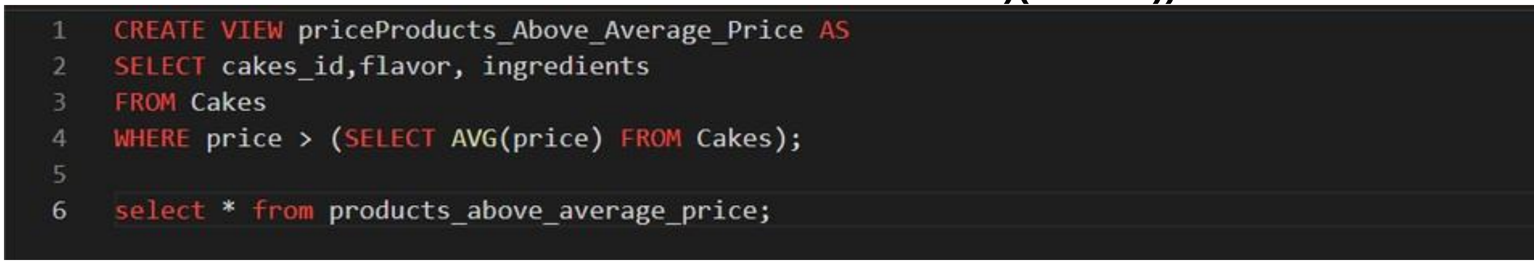
```sql
SELECT SUM(quantity) AS totalQuantity
FROM Orders;
```

R: = γ sum(quantity)(Orders)



```sql
SELECT SUM(quantity) AS totalQuantity
FROM Orders;
WHERE Orders.Cakes_ID IN (SELECT Cakes.Cakes_ID FROM Cakes WHERE name='Greenlam');
```

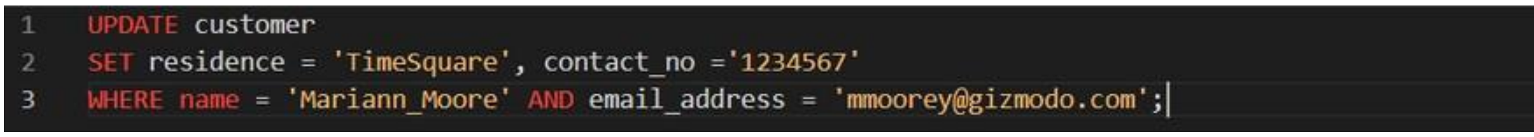
R: = γ sum(quantity(σ orders.cakes ID(Orders) and σ Πorders.cakes Id NAME = 'Greenlam')(Orders))



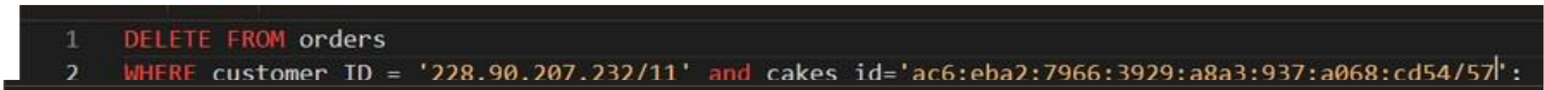
```sql
CREATE VIEW priceProducts_Above_Average_Price AS
SELECT cakes_id,flavor, ingredients
FROM Cakes
WHERE price > (SELECT AVG(price) FROM Cakes);

select * from products_above_average_price;
```

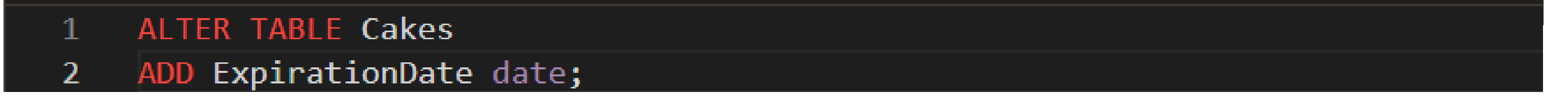
3) R: = σ price > (Y avg(price)(Cakes))(Cakes)



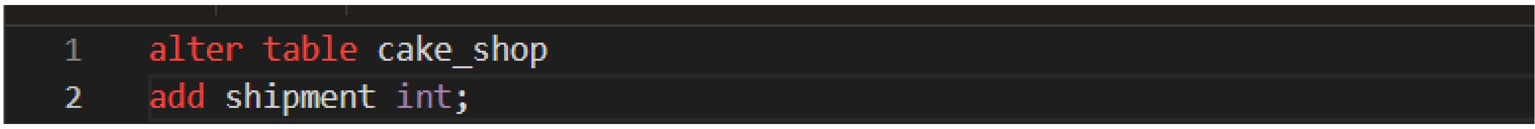
```sql
UPDATE customer
SET residence = 'TimeSquare', contact_no ='1234567'
WHERE name = 'Mariann_Moore' AND email_address = 'mmoorey@gizmodo.com';
```



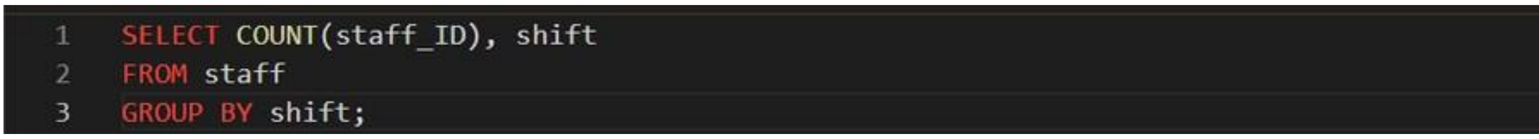
```sql
DELETE FROM orders
WHERE customer_ID = '228.90.207.232/11' and cakes_id='ac6:eba2:7966:3929:a8a3:937:a068:cd54/57';
```



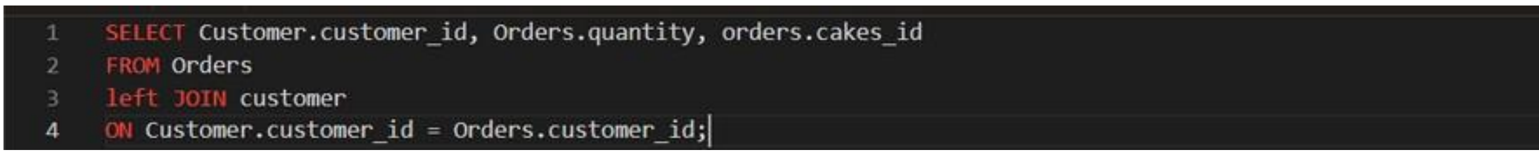
```sql
ALTER TABLE Cakes
ADD ExpirationDate date;
```



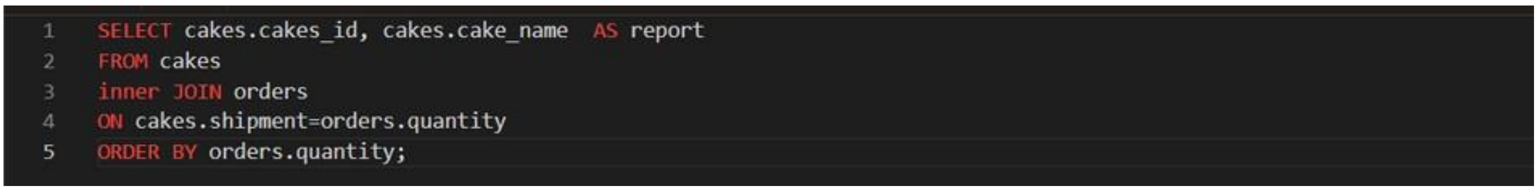
```sql
alter table cake_shop
add shipment int;
```



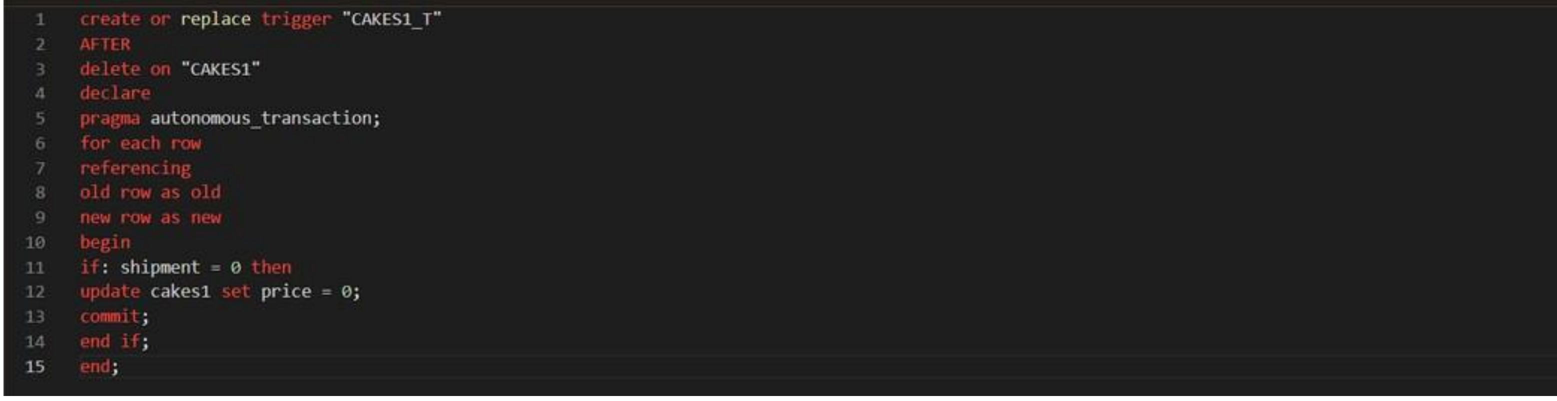
```sql
SELECT COUNT(staff_ID), shift
FROM staff
GROUP BY shift;
```

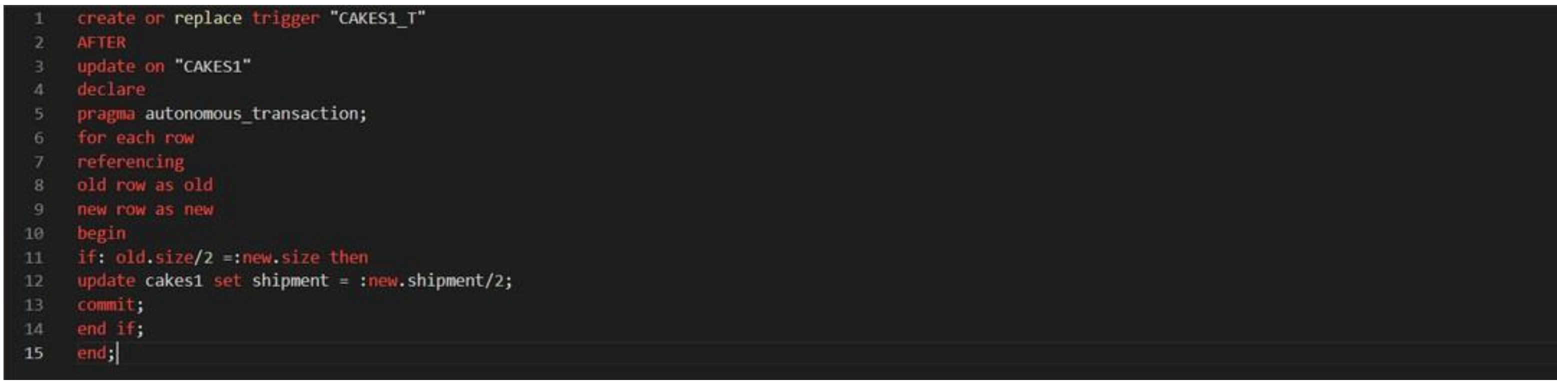


```sql
SELECT Customer.customer_id, Orders.quantity, orders.cakes_id
FROM Orders
left JOIN customer
ON Customer.customer_id = Orders.customer_id;
```



```sql
SELECT cakes.cakes_id, cakes.cake_name  AS report
FROM cakes
inner JOIN orders
ON cakes.shipment=orders.quantity
ORDER BY orders.quantity;
```


```
create or replace trigger "CAKES1_T"
AFTER
delete on "CAKES1"
declare
pragma autonomous_transaction;
for each row
referencing
old row as old
new row as new
begin
if: shipment = 0 then
update cakes1 set price = 0;
commit;
end if;
end;
```


```
create or replace trigger "CAKES1_T"
AFTER
update on "CAKES1"
declare
pragma autonomous_transaction;
for each row
referencing
old row as old
new row as new
begin
if: old.size/2 =:new.size then
update cakes1 set shipment = :new.shipment/2;
commit;
end if;
end;
```