MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
A Constituent Institution of Manipal University

Department of Computer Science and Engineering
Manipal Institute of Technology, Manipal

A Project Report on

Online Grocery-Ordering System

by

Samarth Parashar
Registration Number: 210905206, Roll: 40
And
Aditya Singhvi
Registration Number: 210905216, Roll: 41

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Manipal
12/05/2023

CERTIFICATE

This is to certify that the project titled **MiniProject Online Grocery-Ordering System** is a record of the bonafide work done by **Samarth Parashar (Reg. No. 210905206) and Aditya Singhvi (Reg. No. 210905216)** submitted in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology (B.Tech.) in COMPUTER SCIENCE & ENGINEERING of Manipal Institute of Technology, Manipal, Karnataka, (A Constituent Institute of Manipal Academy of Higher Education), during the academic year 2022-2023.

Name and Signature of Examiners:

1. **Dr. Anup Bhat, Assistant Professor, CSE Dept.**

2. **Mr. Govardhan Hegde, Assistant Professor Selection Grade, CSE Dept.**

DATABASE SYSTEMS LAB
CSE 2262

Introduction

The main objective of the project on online grocery ordering system is to manage the details of grocery, item, item category, shopping stock, customer, order, employees billing of products, managing branches of stores etc. It manages all the information about grocery item, category, shopping cart, customer, order. The project is built to cater to the needs of both the administrator and the user.

The purpose of the project is to build an application program to reduce the manual work for collecting all the details pertaining to the said order and the customer who ordered it. It also enables the customers with a user-friendly access to order groceries at the convenience of their homes.

Problem Statement

The old manual system was suffering from a series of drawbacks. Since whole of the system was to be maintained with hand the process of keeping, maintaining and retrieving the information was very tedious and lengthy. The records were never used to be in a systematic order. There used to be a lot of difficulties in associating any particular transaction with a particular context.

Also, the customers now don't have to go to the stores to order groceries where they had to wait in long never-ending queues. With this online grocery ordering system, they can order things with just with a few clicks, at their doorstep.

Project Objective

The project aims to create an unalterable database to increase efficiency and sale of stores. This project aims to connect the groceries from the stores to the customers in their homes.

Methodology:-

In order to tackle the problem, the work is divided into following phases:

- Visitors/Users can browse all the Categories and grocer Items.
- They also can order easily from the website.
- Admin can Manage Admin, Categories and Grocery Items.
- Admin can also Manage and Track their Order.

DATA REQUIREMENTS:

● We need to store information about various branches. This information includes name, location, branch manager name, manager id and a unique id is assigned to each branch. The branch Id should be unique and should be present for each branch.

● Customer information is stored for each customer who visits a particular branch. This data is stored for retrieval of customer details when required either for customer queries regarding purchase or the branch's need to monitor a customer record. Customer information should include a contact number, email, and name. The customer contact number can be used to uniquely identify customer information and it should be given by each customer who makes a purchase.

● Details of products which are sold in various branches must be stored. These details can be referred to when a product is purchased or sold by a branch. They should include the price, name, id and description of the product. Id of the product can be used to uniquely identify each product and is generally provided at the time of acquiring the product or at the time of manufacture of the product.

● Employees who work in a particular branch must have a unique id. This Id can be used to uniquely identify each employee. Other details like the branch they work in, name, salary, contact and their joining date should be stored.

● Seller information like their id, contact, email, seller name and bank details should be stored. Seller id can be used to uniquely identify each seller and the bank details should be given by each seller so that the transaction can take place smoothly.

● When some product is on sale in a particular branch then details like discount per cent, start date, and end date of the sale must be stored to refer to while billing. These details may or may not be unique for each product in each branch.

● Details of stocks (quantity) available for a particular product in a particular branch must be stored. This information is updated whenever a product is sold or purchased. These details must include the quantity of product in consideration and in which branch these stocks are available.

FUNCTIONAL DEPENDENCIES:-

- `**branch**`:
  - `branch_id -> city, manager_id, manager_name, pin_code`
  - `manager_id -> manager_name`

- `**employee**`:
  - `emp_id, branch_id -> ename, esalary, join_date`
  - `branch_id -> branch_id` (referential integrity constraint)

- `**customer**`:
  - `customer_email -> contact_no, customer_name`

- `**product**`:
  - `product_id -> product_name, product_desc, price`

- `**stock**`:
  - `product_id, branch_id -> stock_quantity`

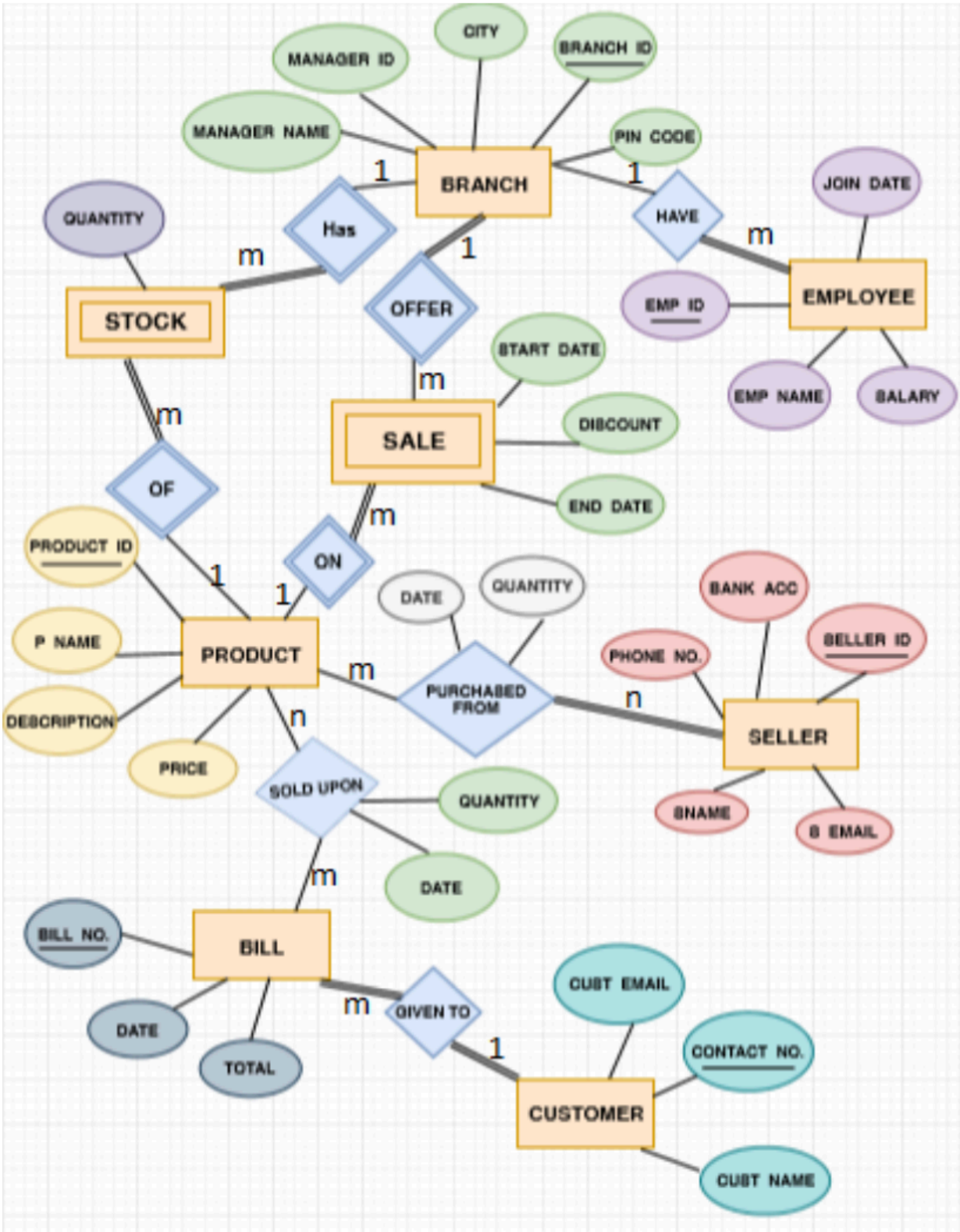
- `**sale**`:
  - `product_id, branch_id -> sdate, edate, discount`

- `**bill**`:
  - `bill_no -> customer_email, bdate, total, branch_id`
  - `customer_email -> customer_name` (transitive dependency)

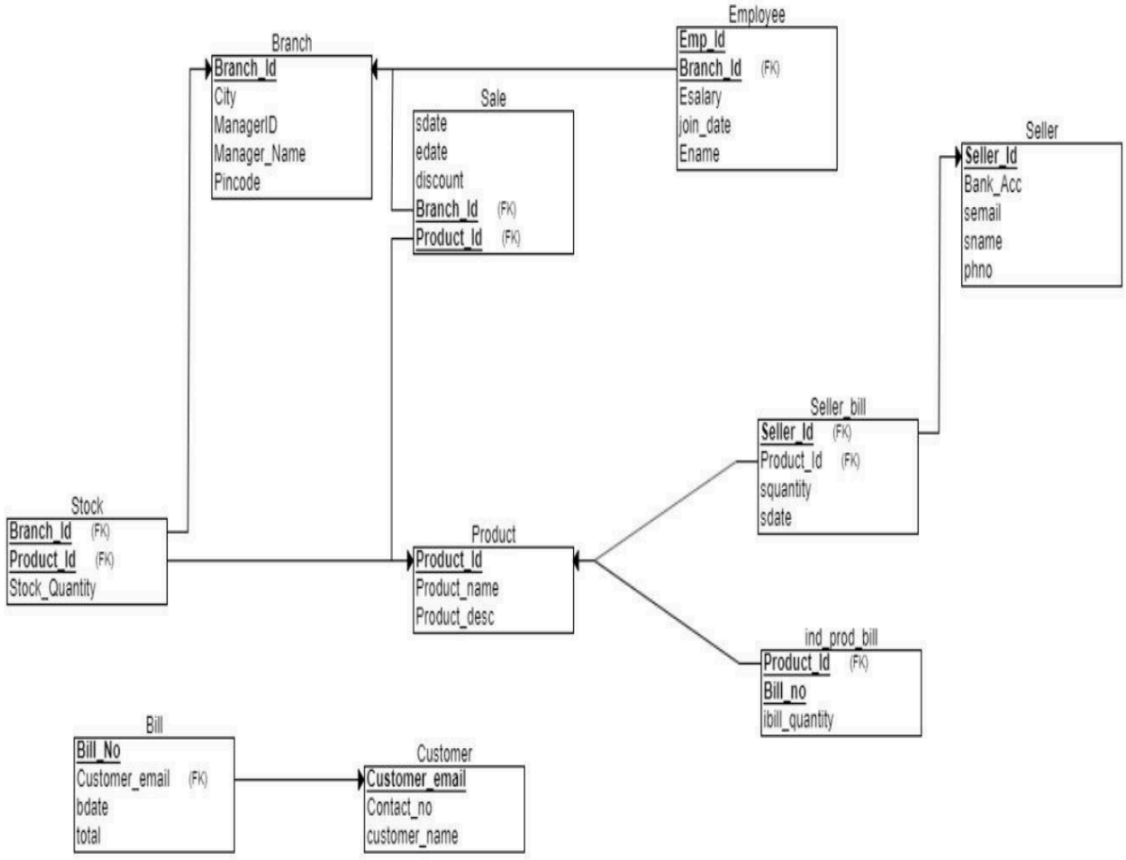
- `**seller**`:
  - `seller_id -> bank_acc, semail, sname, phno`

- `**prod_bill**`:
  - `product_id, bill_no -> ibill_quantity`
- `**seller_bill**`:
  - `seller_id, product_id, sdate -> squantity`

Entity Relationship Diagram



Relational Schema:-



Creation of Tables:-

```
create table branch (
    branch_id       number(5) ,
    city            varchar(10) ,
    manager_id      number(5),
    manager_name    varchar(20),
    pin_code        number(6),
    primary key(branch_id),
    check (pin_code > 100000),
    check(manager_id is NOT NULL)
);

create table employee(
    emp_id  number(5) ,
    ename  VARCHAR(20) ,
    esalary number(9,2) ,
    join_date DATE,
    branch_id references branch,
    primary key(emp_id,branch_id),
    check (length(ename) > 0),
    check (esalary between 10000.00 and 99999999.00)
);


create table customer(
    customer_email varchar(50),
    contact_no number(10),
    customer_name varchar(10),
    primary key(customer_email),
    check (regexp_like(customer_email,'^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$','i')),
    check(regexp_like(to_char(contact_no),'^[6789][0-9]{9}$'))
);
```

```sql
create table product (
    product_id number(5) ,
    product_name varchar(20),
    product_desc varchar(40),
    price number(5,2),
    primary key(product_id),
    check (price > 0),
    check (length(product_name) > 0)
);


CREATE TABLE stock(
    product_id  references product,
    branch_id   references branch,
    stock_quantity    number(3),
    primary key(product_id,branch_id)
);

create table sale(
    product_id  references product,
    branch_id references branch,
    sdate  date,
    edate date,
    discount number(4,2),
    primary key(product_id,branch_id),
    check (sdate < edate),
    check (discount between 01.00 and 99.99)
);
```

```
create table bill(
    bill_no number(4),
    customer_email references customer,
    bdate DATE,
    total number(9,2),
    branch_id references branch,
    primary key(bill_no),
    check (total > 1.00)
);

create table seller(
    seller_id number(5) ,
    bank_acc number(9),
    semail varchar(50),
    sname varchar(10),
    phno number(10),
    primary key(seller_id),
    check (phno >= 1000000000),
    check (bank_acc >= 10000000)
);

create table prod_bill(
    product_id references product,
    bill_no references bill,
    ibill_quantity number(2),
    primary key(product_id,bill_no),
    check (ibill_quantity between 1 and 100)
);


create table seller_bill(
    seller_id references seller,
    product_id references product,
    squantity number(3),
    sdate date,
    primary key(seller_id,product_id,sdate),
    check (squantity between 9 and 9999)
);
```

Table branch:

INSERT INTO branch VALUES(1,'Mumbai','100','Mukesh Ambani',390001);
INSERT INTO branch VALUES(2,'Delhi','200','Ramesh Shah',456123);
INSERT INTO branch VALUES(3,'Kolkata','300','Mukul Roy',456223);

Table employee:

EMPLOYEE TABLE
INSERT INTO employee
VALUES ('101', 'SURESH SINGH', 50000.50, TO_DATE('14-JUN-2020', 'DD-MON-YYYY'), 1);
INSERT INTO employee
VALUES ('102', 'RAMESH SHARMA', 60000.50, TO_DATE('18-APR-2020', 'DD-MON-YYYY'), 1);
INSERT INTO employee
VALUES ('101', 'PRANAV MOHAN', 75000.50, TO_DATE('11-MAR-2020', 'DD-MON-YYYY'), 2);
INSERT INTO employee
VALUES ('102', 'LAKSHAY SAXENA', 90000.80, TO_DATE('24-SEP-2020', 'DD-MON-YYYY'), 2);
INSERT INTO employee
VALUES ('101', 'VAIBHAV SANDHIR', 100000.00, TO_DATE('30-AUG-2020', 'DD-MON-YYYY'), 3);
INSERT INTO employee
VALUES ('102', 'DISHA AGARWAL', 50000.50, TO_DATE('12-JAN-2023', 'DD-MON-YYYY'), 3);

Table customer:-

INSERT INTO customer VALUES ('john@example.com', 9876543210, ‘John');

INSERT INTO customer VALUES ('jane@example.com', 9876543211, ‘Jane');

INSERT INTO customer VALUES ('bob@example.com', 9876543212, ‘Bob');

INSERT INTO customer VALUES ('alice@example.com', 9876543213, ‘Alice’);

INSERT INTO customer VALUES ('david@example.com', 9876543214, ‘David');

INSERT INTO customer VALUES ('sara@example.com', 9876543215, 'Sara');
INSERT INTO customer VALUES ('tom@example.com', 9876543216, 'Tom');
INSERT INTO customer VALUES ('jessica@example.com', 9876543217, 'Jessica');
INSERT INTO customer VALUES ('kevin@example.com', 9876543218, 'Kevin');
INSERT INTO customer VALUES ('emily@example.com', 9876543219, 'Emily');
INSERT INTO customer VALUES ('david@example.com', 9876543214, ‘David’);

Table products:

INSERT INTO product VALUES (1, 'Basmati Rice', 'Aromatic long-grain rice', 149.00);
INSERT INTO product VALUES (2, 'Turmeric Powder', 'Ground turmeric root', 59.90);
INSERT INTO product VALUES (3, 'Red Lentils', 'Split red lentils', 89.50);
INSERT INTO product VALUES (4, 'Chickpeas', 'Dried chickpeas', 79.00);
INSERT INTO product VALUES (5, 'Coriander Powder', 'Ground coriander seeds', 49.90);
INSERT INTO product VALUES (6, 'Cumin Seeds', 'Whole cumin seeds', 69.00);
INSERT INTO product VALUES (7, 'Ghee', 'Clarified butter', 299.00);
INSERT INTO product VALUES (8, 'Mango Pickle', 'Spicy mango pickle', 119.00);
INSERT INTO product VALUES (9, 'Black Tea', 'Loose leaf black tea', 99.50);
INSERT INTO product VALUES (10, 'Garam Masala', 'Spice blend', 54.90);
INSERT INTO product VALUES (11, 'Coconut Oil', 'Cold-pressed coconut oil', 199.00);
INSERT INTO product VALUES (12, 'Saffron', 'Premium quality saffron', 299.00);
INSERT INTO product VALUES (13, 'Jaggery', 'Unrefined cane sugar', 69.00);
INSERT INTO product VALUES (14, 'Mustard Oil', 'Cold-pressed mustard oil', 129.00);
INSERT INTO product VALUES (15, 'Whole Wheat Flour', 'Stone-ground whole wheat flour', 49.90);

Table stock :

```
INSERT INTO stock VALUES (1, 1, 100);
INSERT INTO stock VALUES (2, 1, 200);
INSERT INTO stock VALUES (3, 1, 300);
INSERT INTO stock VALUES (4, 1, 150);
INSERT INTO stock VALUES (5, 1, 250);
INSERT INTO stock VALUES (6, 1, 175);
INSERT INTO stock VALUES (7, 1, 50);
INSERT INTO stock VALUES (8, 1, 100);
INSERT INTO stock VALUES (9, 1, 75);
INSERT INTO stock VALUES (10, 1, 125);
INSERT INTO stock VALUES (11, 1, 25);
INSERT INTO stock VALUES (12, 1, 10);
INSERT INTO stock VALUES (13, 1, 200);
INSERT INTO stock VALUES (14, 1, 75);
INSERT INTO stock VALUES (15, 1, 150);
INSERT INTO stock VALUES (1, 2, 150);
INSERT INTO stock VALUES (2, 2, 100);
INSERT INTO stock VALUES (3, 2, 200);
INSERT INTO stock VALUES (4, 2, 100);
INSERT INTO stock VALUES (5, 2, 150);
INSERT INTO stock VALUES (6, 2, 100);
INSERT INTO stock VALUES (7, 2, 25);
INSERT INTO stock VALUES (8, 2, 50);
INSERT INTO stock VALUES (9, 2, 100);
INSERT INTO stock VALUES (10, 2, 75);
INSERT INTO stock VALUES (11, 2, 15);
INSERT INTO stock VALUES (12, 2, 5);
INSERT INTO stock VALUES (13, 2, 150);
INSERT INTO stock VALUES (14, 2, 50);
INSERT INTO stock VALUES (15, 2, 100);
INSERT INTO stock VALUES (1, 3, 200);
INSERT INTO stock VALUES (2, 3, 150);
INSERT INTO stock VALUES (3, 3, 250);
INSERT INTO stock VALUES (4, 3, 200);
```

```
INSERT INTO stock VALUES (5, 3, 300);
INSERT INTO stock VALUES (6, 3, 175);
INSERT INTO stock VALUES (7, 3, 100);
INSERT INTO stock VALUES (8, 3, 150);
INSERT INTO stock VALUES (9, 3, 125);
INSERT INTO stock VALUES (10, 3, 150);
INSERT INTO stock VALUES (11, 3, 30);
INSERT INTO stock VALUES (12, 3, 15);
INSERT INTO stock VALUES (13, 3, 175);
INSERT INTO stock VALUES (14, 3, 100);
INSERT INTO stock VALUES (15, 3, 200);I
```

Table Sales:-

```
INSERT INTO sale VALUES (1, 1, '12-JAN-2023', '20-JAN-2023', 10.50);
INSERT INTO sale VALUES (2, 1, '15-FEB-2023', '20-FEB-2023', 5.00);
INSERT INTO sale VALUES (3, 1, '10-MAR-2023', '15-MAR-2023', 15.75);
INSERT INTO sale VALUES (4, 1, '01-APR-2023', '10-APR-2023', 20.00);
INSERT INTO sale VALUES (5, 1, '05-MAY-2023', '15-MAY-2023', 12.50);
INSERT INTO sale VALUES (6, 2, '08-JUN-2023', '15-JUN-2023', 8.00);
INSERT INTO sale VALUES (7, 2, '12-JUL-2023', '18-JUL-2023', 5.25);
INSERT INTO sale VALUES (8, 2, '16-AUG-2023', '25-AUG-2023', 15.00);
INSERT INTO sale VALUES (9, 2, '20-SEP-2023', '30-SEP-2023', 25.50);
INSERT INTO sale VALUES (10, 2, '05-OCT-2023', '12-OCT-2023', 10.00);
INSERT INTO sale VALUES (11, 3, '10-NOV-2023', '20-NOV-2023', 18.00);
INSERT INTO sale VALUES (12, 3, '01-DEC-2023', '10-DEC-2023', 10.25);
INSERT INTO sale VALUES (13, 3, '05-JAN-2024', '15-JAN-2024', 12.75);
INSERT INTO sale VALUES (14, 3, '10-FEB-2024', '18-FEB-2024', 22.00);
INSERT INTO sale VALUES (15, 3, '15-MAR-2024', '25-MAR-2024', 16.50);
```

Table Seller:-

```
INSERT INTO seller VALUES (10001, 123456789, 'john@example.com', 'John',
9876543210);
INSERT INTO seller VALUES (10002, 234567890, 'jane@example.com', 'Jane',
8765432109);
INSERT INTO seller VALUES (10003, 345678901, 'peter@example.com', 'Peter',
7654321098);
INSERT INTO seller VALUES (10004, 456789012, 'alice@example.com', 'Alice',
6543210987);
```

INSERT INTO seller VALUES (10005, 567890123, 'bob@example.com', 'Bob', 5432109876);

Table Bill:-

INSERT INTO bill VALUES (1001, 'john@example.com', '10-DEC-2022', 150.00, 1);
INSERT INTO bill VALUES (1002, 'jane@example.com', '12-DEC-2022', 75.00, 2);
INSERT INTO bill VALUES (1003, 'alice@example.com', '15-DEC-2022', 250.00, 3);

Table prod_bill:-

INSERT INTO prod_bill VALUES (1, 1001, 2);
INSERT INTO prod_bill VALUES (2, 1001, 1);
INSERT INTO prod_bill VALUES (3, 1001, 3);
INSERT INTO prod_bill VALUES (1, 1002, 1);
INSERT INTO prod_bill VALUES (2, 1002, 3);
INSERT INTO prod_bill VALUES (4, 1002, 2);
INSERT INTO prod_bill VALUES (5, 1002, 1);
INSERT INTO prod_bill VALUES (2, 1003, 4);
INSERT INTO prod_bill VALUES (3, 1003, 1);

Table Seller_bill:-

INSERT INTO seller_bill VALUES (10001, 1, 50, '10-DEC-2022');
INSERT INTO seller_bill VALUES (10002, 2, 75, '11-DEC-2022');
INSERT INTO seller_bill VALUES (10003, 3, 30, '12-DEC-2022');
INSERT INTO seller_bill VALUES (10001, 5, 100, '12-DEC-2022');
INSERT INTO seller_bill VALUES (10003, 2, 200, '14-DEC-2022');
INSERT INTO seller_bill VALUES (10002, 1, 25, '15-DEC-2022');
INSERT INTO seller_bill VALUES (10003, 1, 80, '15-DEC-2022');

RESULTS AND SNAPSHOTS:-

DATA UPDATE QUERIES:-

```
UPDATE product
SET price= 220.2
WHERE product_id=9;
```

```
1 row(s) updated.
```

```
UPDATE sale
SET sdate=to_date('02-FEB-2000','DD-MON-YYYY')
WHERE product_id=2 AND branch_id=1;
```

```
1 row(s) updated.
```

```
UPDATE branch
SET manager_name='Nitin Patel'
WHERE branch_id=1;
```

```
1 row(s) updated.
```

```
UPDATE seller
SET pHno=9893965437
WHERE seller_id=10005;
```

```
1 row(s) updated.
```

```
UPDATE stock
SET stock_quantity = stock_quantity - (SELECT sum(ibill_quantity) from bill NATURAL JOIN prod_bill
WHERE branch_id = stock.branch_id AND product_id = stock.product_id
GROUP BY product_id);
```

```
1 row(s) updated.
```

DATA DELETE QUERIES:-

```
1  DELETE FROM prod_bill
2  WHERE EXISTS
3  (SELECT 1 from bill
4   WHERE bill_no = prod_bill.bill_no
5   AND
6   bdate < add_months(SYSDATE,-5)
7  );
```

```
3 row(s) deleted.
```

```
1  DELETE FROM bill
2  WHERE bdate < add_months(SYSDATE,-5);
3
```

```
1 row(s) deleted.
```

```
1  DELETE FROM seller S WHERE NOT EXISTS
2  (SELECT max(sdate) FROM seller_bill WHERE
3      seller_id = S.seller_id AND
4      sdate > add_months(SYSDATE,-5)
5  GROUP BY seller_id);
```

```
2 row(s) deleted.
```

DATA RETRIEVAL QUERIES:-

Q) Number of Employees working in various branches?

```
1  SELECT branch_id, count(*) FROM employee
2      GROUP BY branch_id;
3
```

BRANCH_ID	COUNT(*)
1	2
2	2
3	2

Download CSV

3 rows selected.

Q) Get details of Seller who sells all the products?

```
1  SELECT * FROM SELLER S WHERE NOT EXISTS
2  (SELECT product_id FROM product
3      MINUS
4      SELECT DISTINCT(product_id) FROM seller_bill WHERE seller_id = S.seller_id);
5
```

no data found

Q) 5 most costly products in supply chain?

```
1  SELECT * FROM (SELECT * FROM product
2      ORDER by price DESC)
3      WHERE ROWNUM <6;
4
5
6
7
8
```

PRODUCT_ID	PRODUCT_NAME	PRODUCT_DESC	PRICE
7	Ghee	Clarified butter	299
12	Saffron	Premium quality saffron	299
9	Black Tea	Loose leaf black tea	220.2
11	Coconut Oil	Cold-pressed coconut oil	199
1	Basmati Rice	Aromatic long-grain rice	149

Download CSV

5 rows selected.

Q) Employees having salary greater than 80000 rupees?

```
SELECT *
FROM employee
    where esalary > 80000
```

EMP_ID	ENAME	ESALARY	JOIN_DATE	BRANCH_ID
102	LAKSHAY SAXENA	90000.8	24-SEP-20	2
101	VAIBHAV SANDHIR	100000	30-AUG-20	3

Download CSV

2 rows selected.

Q) Get last purchase dates of customers

```
SELECT customer_email,max(bdate)
FROM customer NATURAL JOIN bill
GROUP BY customer_email;
```

CUSTOMER_EMAIL	MAX(BDATE)
alice@example.com	15-DEC-22
jane@example.com	12-DEC-22

Download CSV

2 rows selected.

Functions:-

1.AVERAGE OF ALL SALES AT ALL BRANCHES TILL DATE

```
create or replace function avg_sale
return number as average number;
cursor branches is
    select branch_id from branch;
branch_data branches%rowtype;
branch_count number;
branch_sum number;
begin
    select count(branch_id) into branch_count from branch;
    average := 0;
    open branches;
    loop
        begin
            fetch branches into branch_data;
            exit when branches%NOTFOUND;
            select sum(total) into branch_sum from bill where branch_id = branch_data.branch_id ;
            if branch_sum is null then
                branch_sum := 0;
            end if;
        end;
        average := average + branch_sum;
    end loop;
    close branches;
    average := average / branch_count;
    return(average);
end;
/
DECLARE
average_sale number;
begin
average_sale:=avg_sale();
dbms_output.put_line('average sale of all branches is:'||average_sale);
end;
/
```

```
Function created.

Statement processed.
average sale of all branches is:158.33333333333333333333333333333333333333
```

2.To select the most popular purchase and comparing its sales within 2 branches

```
create or replace function popular_prod(branch1 number, branch2 number, productId number)
return number
as
branch3 number;
qbranch1 number := 0;
qbranch2 number := 0;
temp number;
cursor qb1 is select bill_no from bill where branch_id = branch1;
qb1d qb1%rowtype;
cursor qb2 is select bill_no from bill where branch_id = branch2;
qb2d qb2%rowtype;
begin
    open qb1;
    loop
        temp := 0;
        begin
            fetch qb1 into qb1d;
            exit when qb1%NOTFOUND;
            select ibill_quantity into temp from prod_bill where product_id = productId and bill_no = qb1d.bill_no;
            exception
                when NO_DATA_FOUND then temp := 0;
        end;
        qbranch1 := qbranch1 + temp;
    end loop;
    close qb1;
    open qb2;
    loop
        temp := 0;
        begin
            fetch qb2 into qb2d;
            exit when qb2%NOTFOUND;
            select ibill_quantity into temp from prod_bill where product_id = productId and bill_no = qb2d.bill_no;
            exception
                when NO_DATA_FOUND then temp := 0;
        end;
        qbranch2 := qbranch2 + temp;
    end loop;
    close qb2;
    if (qbranch1 > qbranch2) then branch3 := branch1;
    elsif (qbranch2 > qbranch1) then branch3 := branch2;
    else branch3 := 0;
    end if;
    return branch3;
end;
/

DECLARE
branch_popular number;
begin
branch_popular:=popular_prod(1,2,3);
```

```
Function created.

Statement processed.
Product 3 is most popular in brach_id:1
```

1.A PROCEDURE TO DISPLAY BILL WITH LIST OF PRODUCTS IN IT AND THEIR DETAILS.

```
create or replace procedure DISPLAY_BILL (bill_number number) as
c_name customer.customer_name%type;
c_email customer.customer_email%type;
bill_date bill.bdate%type;
bill_total bill.total%type;
cursor prod_list is select product_id, product_name, ibill_quantity, price from prod_bill natural join product
    where bill_no = bill_number;
products_data prod_list%rowtype;
ptotal bill.total%type;
begin
    select customer_name,customer_email into c_name,c_email from customer
    where customer_email in (select customer_email from bill where bill_no = bill_number);
    select bdate, total into bill_date,bill_total from bill where bill_no = bill_number;
    dbms_output.put_line(chr(10));
    dbms_output.put_line('BILL NUMBER: ' || bill_number);
    dbms_output.put_line('BILL DATE: ' || bill_date);
    dbms_output.put_line('CUSTOMER NAME: ' || c_name);
    dbms_output.put_line('CUSTOMER EMAIL: ' || c_email);
    dbms_output.put_line(rpad('NAME',25,' ') || rpad('ID',10,' ') || rpad('PRICE',10, ' ')
        || rpad('QTY.',7, ' ') || rpad('TOTAL',10, ' '));
    open prod_list;
    loop
        fetch prod_list into products_data;
        exit when prod_list%NOTFOUND;
        ptotal := products_data.price * products_data.ibill_quantity;
        dbms_output.put_line(rpad(products_data.product_name,25,' ') || rpad(products_data.product_id,10,' ') ||
        rpad(products_data.price,10, ' ') || rpad(products_data.ibill_quantity,7, ' ') || rpad(ptotal,10, ' '));
    end loop;
    close prod_list;
    dbms_output.put_line('BILL TOTAL: ' || bill_total);
    dbms_output.put_line(chr(10));
end;
/
SET SERVEROUTPUT ON;
EXEC DISPLAY_BILL(1001);
/

```

```
BILL NUMBER: 1001
BILL DATE: 10-DEC-22
CUSTOMER NAME: John
CUSTOMER EMAIL: john@example.com
NAME                      ID        PRICE     QTY.   TOTAL
Basmati Rice              1         149       2      298
Turmeric Powder           2         59.9      1      59.9
Red Lentils               3         89.5      3      268.5
BILL TOTAL: 150
```

2. A PROCEDURE TO DISPLAY ALL STOCKS IN A PARTICULAR BRANCH.

```
create or replace procedure show_stocks (branchId number)
as
cursor stock_list is
    select product_id, stock_quantity from stock
    where branch_id = branchId;
stock_data stock_list%rowtype;
pname  product.product_name%type;
pprice product.price%type;
pdesc  product.product_desc%type;
psaled  sale.discount%type;
psalesd sale.sdate%type;
psaleed sale.edate%type;
begin
    dbms_output.put_line('Branch Id: ' || branchId);
    dbms_output.put_line(rpad('NAME',17,' ')||rpad('ID',7,' ')||rpad('DESCRIPTION',20,' ')||rpad('PRICE',7,' ')||
    rpad('Qty',5,' ')||rpad('SALE',6,' ')||rpad('Sale_Start',15,' ')||rpad('Sale_End',15,' '));
    open stock_list;
    loop
        begin
            fetch stock_list into stock_data;
            exit when stock_list%NOTFOUND;
            select product_name,product_desc,price into pname,pdesc,pprice from product where product_id = stock_data.product_id;
            select discount,sdate,edate into psaled,psalesd,psaleed from sale where product_id = stock_data.product_id and branch_id = branchId;
            exception
                when NO_DATA_FOUND then
                psaled := 0;
                psalesd := null;
                psaleed := null;
        end;
        dbms_output.put_line(rpad(pname,17,' ')||rpad(stock_data.product_id,7,' ')||rpad(pdesc,20,' ')||
        rpad(stock_data.stock_quantity,5,' ')||rpad(pprice,7,' ')||rpad(psaled,6,' ')||rpad(psalesd,15,' ')||rpad(psaleed,15,' '));
    end loop;
    close stock_list;
end;
/
```

```
Procedure created.
```

TRIGGERS:-

A TRIGGER TO GIVE CUSTOMERS DISCOUNT IF THEY ARE SHOPPING FOR FIRST TIME.

```
create or replace trigger customer_trigger
before insert on bill
for each row
declare
c number;
begin
    select count(*) into c from bill where customer_email = :new.customer_email group by customer_email;
    exception
    when NO_DATA_FOUND then
        :new.total := :new.total * 0.80;
        dbms_output.put_line('You Got 20% Off Coz This Is Your 1st Shopping At Our Stores.');
end;
/
insert into bill values(1004,'alice@example.com','30-MAY-2023',250,1);
```

```
Trigger created.

1 row(s) inserted.
```

A TRIGGER TO CHECK IF SALARY OF EMPLOYEE IS ALWAYS INCREASED AND IF IT IS TRIED TO DECREASE THEN DB WON'T ALLOW SUCH UPDATE.

```
select * from employee;
create or replace trigger salary_update
before update of esalary on employee
for each row
begin
if (:new.esalary < :old.esalary) then
    :new.esalary := :old.esalary;
end if;
end;
/
update employee set esalary=20000 where emp_id=101;
select * from employee;
```

EMP_ID	ENAME	ESALARY	JOIN_DATE	BRANCH_ID
101	SURESH SINGH	50000.5	14-JUN-20	1
102	RAMESH SHARMA	60000.5	18-APR-20	1
101	PRANAV MOHAN	75000.5	11-MAR-20	2
102	LAKSHAY SAXENA	90000.8	24-SEP-20	2
101	VAIBHAV SANDHIR	100000	30-AUG-20	3
102	DISHA AGARWAL	50000.5	12-JAN-23	3

Download CSV

6 rows selected.

Trigger created.

3 row(s) updated.

A TRIGGER WHICH DOESN'T ALLOW UPDATE OF EMPLOYEES SALARY BEFORE SIX MONTHS FROM HIS JOIN DATE.

```
select * from employee;
create or replace trigger employee_salary_update
before update of esalary on employee
for each row
begin
if(SYSDATE - :old.join_date < 180) then
    raise_application_error(-20000,'Invalid Update Option');
end if;
end;
/
update employee set esalary=20000 where emp_id=102 and branch_id=3;
select * from employee;
```

```
Trigger created.

ORA-20000: Invalid Update Option ORA-06512: at "SQL_UMCPJEJMWSWXFIQINKQXFHCQM.EMPLOYEE_SALARY_UPDATE", line 3
ORA-06512: at "SYS.DBMS_SQL", line 1721
```

EMP_ID	ENAME	ESALARY	JOIN_DATE	BRANCH_ID
101	SURESH SINGH	50000.5	14-JUN-20	1
102	RAMESH SHARMA	60000.5	18-APR-20	1
101	PRANAV MOHAN	75000.5	11-MAR-20	2
102	LAKSHAY SAXENA	90000.8	24-SEP-20	2
101	VAIBHAV SANDHIR	100000	30-AUG-20	3
102	DISHA AGARWAL	50000.5	12-JAN-23	3

Download CSV

6 rows selected.

Conclusion:

The database will help keep track of all the transactions that take place. Also, It will store the information about the groceries and its categories. I would like to thank Dr. Anup Bhat for his teachings which have helped us to make this project.

Bibliography/References:

- Oracle PL/SQL Database Programming Learner
- Oracle DP Database Programming with SQL
- W3SCHOOLS
- Silberschatz, Korth, Sudarshan, “Database System Concepts”, McGrawHill, 6th Edition.

THANKYOU