Farmer Consumer Relationship

- This database design was created for a nonprofit organization that aims to close the gap between farmers and consumers by enabling online direct sales of agricultural products.

- The main users of the system will be farmers, who would gain from a robust platform to manage their product listings, marketplaces, and sales without the need for external intermediaries. Easy access will benefit customers.

Some of the key features :

- Product Management
- Order Management
- Payment Tracking
- Customer Support
- Discounts and Reviews
- Farmer and Consumer Analytics
- Return and Exchange policy

Data Dictionary:

ATTRIBUTE	INFORMATION DEFINITION	TYPE
consumer_first_name	This is the first name of the consumer	Characters - maximum of 20 characters
consumer_last_name	This is the last name of the consumer	Characters - maximum of 20 characters
farmer_first_name	This is the first name of thefarmer	Characters - maximum of 20 characters
farmer_last_name	This is the last name of thefarmer	Characters - maximum of 20 characters
consumer_email_id	This is the unique email address of the consumer	Characters - maximum of 30 characters
farmer_email_id	This is the unique email address of the farmer	Characters - maximum of 30 characters
consumer_phone_number	This is the unique phone number of the consumer	Characters - maximum of 20 characters due to varying length of digits for different countries
farmer_phone_number	This is the unique phone number of the farmer	Characters - maximum of 20 characters due to varying length of digits for different countries
farmer_address	This is the location of thefarmer	Characters - maximum of 100
consumer_adress	This is the location of theconsumer	Characters - maximum of 100
product_name	This is the name of the product	Characters - maximum of 20
product_id	This is the unique identifierfor products	Characters - maximum of 10as id must contain 10 digits and can consist of alphabets.
farm_name	This is the name of the Farm	Characters - maximum of 20
product_description	This is the description of/comment on the product	Characters - maximum of 300
product_price	This is the price of the product	Decimal- Positive value only
product_quantity_available	This is the availability stock of product	Integer- Positive value only

product_order_id	This is the unique identifier for orders made for products	Characters - maximum of 10 as id must contain 10 digits and can consist of alphabets with digits too.
product_order_date	This provides the date an order was made	Date Time - this is automatically generated by the system not the user
product_payment_status	This indicates if a consumers transaction was successful or not	Boolean data type/integer between 0 and 1.
product_payment_date	This provides the date of payment	Date Time - this is automatically generated by the system not the user
product_reviews	This provides the review on the product	Characters - maximum of 300
product_exchange_policy	This provide exchange of the product if there's any damage	Characters - maximum of 300
product_discount	This provides the discounts for the products	Integer - must be above 0 and less than 100
product_order_tracking_id	This provides the order status to the consumers	Characters - maximum of 10 as id must contain 10 digits and can consist of alphabets with digits too.
product_cost	This indicates the amount the consumer would pay for a product	Positive decimal
product_quantity_selected	This provides the quantity of the selected product by the consumer	Positive integer

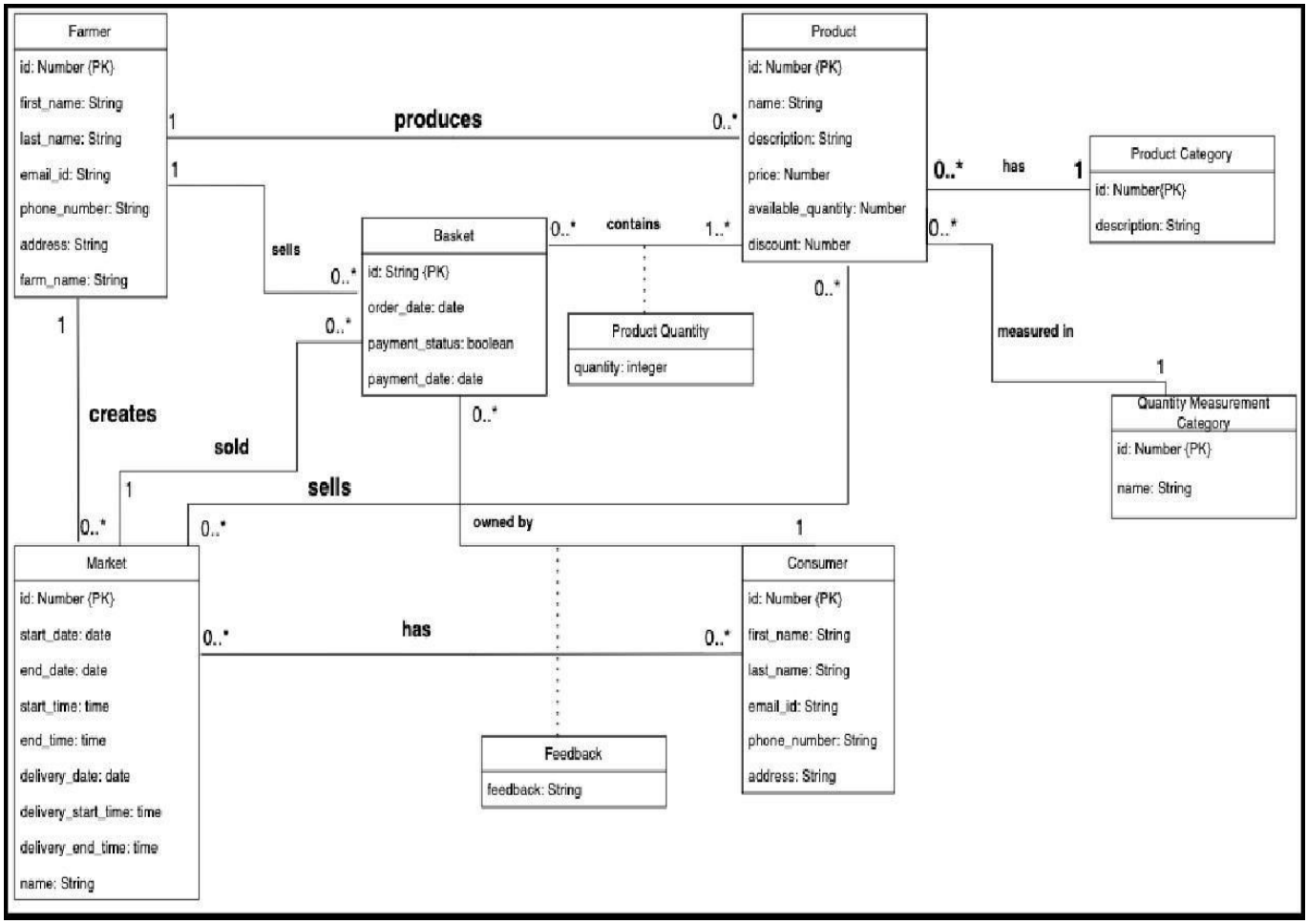

TABLE CREATION AND DATA INSERTION :

FARMER:

```
CREATE TABLE farmer (
 id INT GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY,
 first_name VARCHAR(50) NOT NULL,
 last_name VARCHAR(50),
email_id VARCHAR(100) UNIQUE,
phone_number VARCHAR(15) NOT NULL, address VARCHAR(255) NOT NULL,
farm_name VARCHAR(100) NOT NULL
 );
```

INSERT INTO farmer (first_name, last_name, email_id, phone_number, address, farm_name) VALUES ('Daniel', 'Gbenga', 'daniel.gbenga@example.com', '0243-456-7890', '543 Farm Road, Countryside', 'Daniel G Farms');

Comments: The id is automatically generated if its not provided.

MARKET:

```
CREATE TABLE market (
        id NUMBER GENERATED BYDEFAULTONNULLASIDENTITY PRIMARY KEY,
        start_date DATE NOT NULL,
        end_date DATE NOT NULL,
        start_time TIMESTAMP NOT NULL,
        end_time TIMESTAMP NOT NULL,
        delivery_day VARCHAR2(10),
        delivery_start_time TIMESTAMP,
        delivery_end_time TIMESTAMP,
        name VARCHAR2(100),
        farmer_id NUMBER NOT NULL,
        CONSTRAINT fk_farmer_id
                FOREIGN KEY (farmer_id) REFERENCES Farmer(id)
        );
```

INSERT INTO market (start_date, end_date, start_time, end_time, delivery_day, delivery_start_time, delivery_end_time, name, farmer_id) VALUES (TO_DATE('2023-11-15', 'YYYY-MM-DD'), TO_DATE('2023-11-16', 'YYYY-MM-DD'), TO_TIMESTAMP('2023-11-15 09:00:00', 'YYYY-MM-DD HH24:MI:SS'), TO_TIMESTAMP('2023-11-15 17:00:00', 'YYYY-MM-DD HH24:MI:SS'), 'Wednesday', TO_TIMESTAMP('2023-11-17 10:00:00', 'YYYY-MM-DD HH24:MI:SS'), TO_TIMESTAMP('2023-11-17 16:00:00', 'YYYY-MM-DD HH24:MI:SS'), 'Money Market', 1);
Comment: The TO_DATE function converts String format to DATE data type while TO_TIMESTAMP converts String format to TIMESTAMP. The String must be in a certain format for it to work.

CONSUMER:

```
CREATE TABLE consumer (
id NUMBER GENERATED BYDEFAULTONNULLASIDENTITY PRIMARY KEY,
first_name VARCHAR2(50) NOT NULL,
last_name VARCHAR2(50),
email_id VARCHAR2(100) UNIQUE,
phone_number VARCHAR2(15) NOT NULL,
address VARCHAR2(255) NOT NULL );
```

INSERT INTO consumer (first_name, last_name, email_id, phone_number, address) VALUES ('Chi-uba', 'Smith', 'chi-uba.smith@example.com', '0123-396-7890', '123 Main St, Rouen');

BASKET:

```
CREATE TABLE basket (
id NUMBER GENERATED BYDEFAULTONNULLASIDENTITY PRIMARY KEY,
        order_date DATE NOT NULL,
        payment_status NUMBER(1) DEFAULT 0 NOT NULL,
        payment_date DATE,
        farmer_id NUMBER NOT NULL,
        market_id NUMBER NOT NULL,
        consumer_id NUMBER,
        CONSTRAINT basket_fk_farmer_id
                FOREIGN KEY (farmer_id) REFERENCES Farmer(id),
        CONSTRAINT basket_fk_market_id
                FOREIGN KEY (market_id) REFERENCES Market(id),
         CONSTRAINT basket_fk_consumer_id
                FOREIGN KEY (consumer_id) REFERENCES Consumer(id)
        );
```

INSERT INTO basket (order_date, payment_status, payment_date, farmer_id, market_id, consumer_id) VALUES (TO_DATE('2024-11-10', 'YYYY-MM-DD'), 1, TO_DATE('2024-11-11', 'YYYY-MM-DD'), 1, 1, 1); Comment: I used 0 and 1 to indicate true or false for boolean data type.

PRODUCTCATEGORY:

```
CREATE TABLE product_category (
id NUMBER GENERATED BYDEFAULTONNULLASIDENTITY PRIMARY KEY, description VARCHAR2(100) NOT NULL );
```

INSERT INTO product_category (description) VALUES ('Fruits');

QUANTITY MEASUREMENTCATEGORY:

```
CREATE TABLE quantity_measurement_category (
 id NUMBER GENERATED BYDEFAULTONNULLASIDENTITY PRIMARY KEY, name VARCHAR2(50) NOT NULL
);
```

INSERT INTO quantity_measurement_category (name) VALUES ('Kilogram');

PRODUCT:

```
CREATE TABLE Product (
        id NUMBER GENERATED BYDEFAULTONNULLASIDENTITY PRIMARY KEY,
        name VARCHAR2(100),
        description VARCHAR2(255),
        price NUMBER NOT NULL CHECK (price > 0),
        available_quantity NUMBER DEFAULT 0 CHECK (available_quantity >= 0),
        discount NUMBER DEFAULT 0 CHECK (discount >= 0),
        category NUMBER, measurement NUMBER NOT NULL,
        farmer_id NUMBER NOT NULL,
        CONSTRAINT product_fk_product_category
                FOREIGN KEY (category) REFERENCES product_category(id),
        CONSTRAINT product_fk_measurement_category
                FOREIGN KEY (measurement) REFERENCES
        quantity_measurement_category(id),
                CONSTRAINT product_fk_farmer_id
                        FOREIGN KEY (farmer_id) REFERENCES Farmer(id)
        );
```

```
INSERT INTO product (name, description, price, available_quantity, discount, category, measurement,
farmer_id) VALUES ( 'Apple', 'Fresh apples from the farm', 1.50, 100, 0, 1, 1, 1 );
```

PRODUCTQUANITY:

```
CREATE TABLE product_quantity (
 basket_id NUMBER,
product_id NUMBER,
quantity NUMBER NOT NULL CHECK (quantity >= 0),
CONSTRAINT product_quantity_pk_product_quanity PRIMARY KEY (basket_id, product_id),
CONSTRAINT product_quantity_fk_basket_id FOREIGN KEY (basket_id) REFERENCES Basket(id),
CONSTRAINT product_quantity_fk_product_id FOREIGN KEY (product_id) REFERENCES Product(id)
 );
 INSERT INTO product_quantity (basket_id, product_id, quantity) VALUES (1, 1, 10);
```

FEEDBACK:

CREATE TABLE feedback (

basket_id NUMBER,

consumer_id NUMBER,

feedback VARCHAR2(255),

CONSTRAINT feedback_pk_feedback PRIMARY KEY (basket_id, consumer_id),

CONSTRAINT feedback_fk_feedback_basket FOREIGN KEY (basket_id) REFERENCES basket(id),

CONSTRAINT feedback_fk_feedback_consumer FOREIGN KEY (consumer_id) REFERENCES consumer(id));
INSERT INTO feedback (basket_id, consumer_id, feedback) VALUES (1, 1, 'Quality product');

PRODUCTSSOLDINMARKET:

CREATE TABLE products_sold_in_market (market_id NUMBER, product_id NUMBER,

CONSTRAINT pk_products_sold_in_market PRIMARY KEY (market_id, product_id),

CONSTRAINT fk_market_id FOREIGN KEY (market_id) REFERENCES Market(id),

CONSTRAINT fk_product_id FOREIGN KEY (product_id) REFERENCES Product(id));

INSERT INTO products_sold_in_market (market_id, product_id) VALUES (1, 1);

MARKETATTENDANCE: CREATE TABLE market_attendance (market_id NUMBER, consumer_id NUMBER,
CONSTRAINT pk_market_attendance PRIMARY KEY (market_id, consumer_id),

CONSTRAINT fk_attendance_market FOREIGN KEY (market_id) REFERENCES Market(id),

CONSTRAINT fk_attendance_consumer FOREIGN KEY (consumer_id) REFERENCES Consumer(id));

INSERT INTO market_attendance (market_id, consumer_id) VALUES (1, 1)

SQL QUERIES TO GET INFORMATION FROM THE DATABASE BASED ON APPLICATION DESCRIPTION

1. Displays all farmers' details.

```
SELECT *
FROM farmer;
```

2. Provides a list of consumer's details that attended/ordered from a market

```
SELECT c.id consumer_id,
c.first_name,
c.last_name,
c.email_id,
m.name market_name
FROM market_attendance ma
JOIN consumer c ON ma.consumer_id = c.id
JOIN market m ON ma.market_id = m.id
WHERE m.id = 1; -- Replace 1 with the specific market ID
```

3. Provides a summary of consumers purchases from all markets

```
SELECT c.id AS consumer_id,
c.first_name,
c.last_name,
    COUNT(b.id) AS total_baskets,
    SUM(pq.quantity * p.price) AS total_spent
FROM consumer c
LEFT JOIN basket b ON c.id = b.consumer_id
LEFT JOIN product_quantity pq ON b.id = pq.basket_id
LEFT JOIN product p ON pq.product_id = p.id
GROUP BY c.id, c.first_name, c.last_name;
```

4. Provides feedback of customers on baskets from a specified market

```
SELECT f.basket_id,
f.consumer_id,
c.first_name, c.last_name,
f.feedback
FROM feedback f
JOIN consumer c ON f.consumer_id = c.id
JOIN basket b ON f.basket_id = b.id
WHERE b.market_id = 1;
-- Replace 1 with the specific market ID
```

5. Provides order details of a market for a specific day

```
SELECT
   m.name AS market_name,
   b.order_date,
   p.name AS product_name,
   pq.quantity,
   p.price,
   (pq.quantity * p.price) AS total_price,
   c.first_name AS consumer_first_name,
   c.last_name AS consumer_last_name
FROM basket b
JOIN product_quantity pq ON b.id = pq.basket_id
JOIN product p ON pq.product_id = p.id
JOIN market m ON b.market_id = m.id
```

```
LEFT JOIN consumer c ON b.consumer_id = c.id
WHERE b.order_date = TO_DATE('2024-11-10', 'YYYY-MM-DD')
   AND m.id = 1;
-- The order date and market_id used here are changeable
```

6. Provides number of customers who placed an order in a market

```
SELECT
   m.name AS market_name,
   COUNT(DISTINCT b.consumer_id) AS total_customers
FROM market m
JOIN basket b ON m.id = b.market_id
GROUP BY  m.name;
```

7. Total quantity of each product sold in each markets

```
SELECT
   m.name market_name,
   p.name product_name,
   SUM(pq.quantity) total_quantity_sold
FROM market m
JOIN basket b ON m.id = b.market_id
JOIN product_quantity pq ON b.id = pq.basket_id
JOIN product p ON pq.product_id = p.id
GROUP BY m.name, p.name
ORDER BY m.name, p.name;
```

8. Provides a summary of orders for each market

```
SELECT
   m.name AS market_name,
   COUNT(b.id) AS total_orders,
   SUM(pq.quantity * p.price) AS total_revenue
FROM market m
JOIN basket b ON m.id = b.market_id
JOIN product_quantity pq ON b.id = pq.basket_id
JOIN product p ON pq.product_id = p.id
GROUP BY m.name;
```

9. Provides a summary of orders made to a farmer

```
SELECT
   f.farm_name AS farmer_name,
   m.name AS market_name,
   p.name AS product_name,
   SUM(pq.quantity) AS total_quantity_sold,
   SUM(pq.quantity * p.price) AS total_revenue
FROM farmer f
JOIN product p ON f.id = p.farmer_id
JOIN product_quantity pq ON p.id = pq.product_id
JOIN basket b ON pq.basket_id = b.id
JOIN market m ON b.market_id = m.id
WHERE f.id = 1
GROUP BY f.farm_name, m.name, p.name;
-- Farmer ID- 1 cna be replaced
```

10. Provides a customers order summary

```
SELECT
   c.first_name || ' ' || c.last_name AS consumer_name,
   m.name AS market_name,
   COUNT(b.id) AS total_orders,
```

```
   SUM(pq.quantity * p.price) AS total_spent
FROM consumer c
JOIN basket b ON c.id = b.consumer_id
JOIN product_quantity pq ON b.id = pq.basket_id
JOIN product p ON pq.product_id = p.id
JOIN market m ON b.market_id = m.id
GROUP BY c.first_name, c.last_name, m.name;
```

THANK YOU