المملكة العربية السعودية
جدة
كلية علوم وهندسة الحاسب

جامعة جدة
University of Jeddah

كلية علوم وهندسة الحاسب

College of Computer Science and Engineering

Database Project Proposal

(CEDS 215)

Bakery

Student Name	Student ID
Remas Alsalmi	2410698
Jory Abualjadayel	2410795
Ruba Alghamdi	2410217

Cours Section: IT1

Cours Number: 18532

Instructor: Mead AlQuthmi

Table of Contents:

Contents	Page number
Cover Page	1
Table of Contents:	2
Project Schedule	
Introduction	3
Project description	
Problem description	
Entities	
Conceptual ER model	5
Relational ER model	6
Logical ER model	7
Normalization	8
Creating tables	10
Inserting values	12
Data retrieval queries	15
Stored procedures	17

Project Schedule:

Task	Due date
Group Project Proposal	Mar 7
System analysis and E-R Model	Mar 25
Logical modelling and normalization	
Full project	Mar 29

Tasks:

Member	Task
Remas Alsalmi	Relational Model, Functional dependency, Creating Tables
Jory Abualjadayel	Conceptual ER Model, Inserting Values, Normalization
Ruba Alghamdi	Logical Model, Inserting Values, Normalization
All Members	Queries, Procedures

Phase 1

Introduction:

Project Description:

A bakery is always in need for a database management system due to the necessity of the list below:

- Inventory management.
- Customer management.
- Order management.
- Data security.

Problem Description:

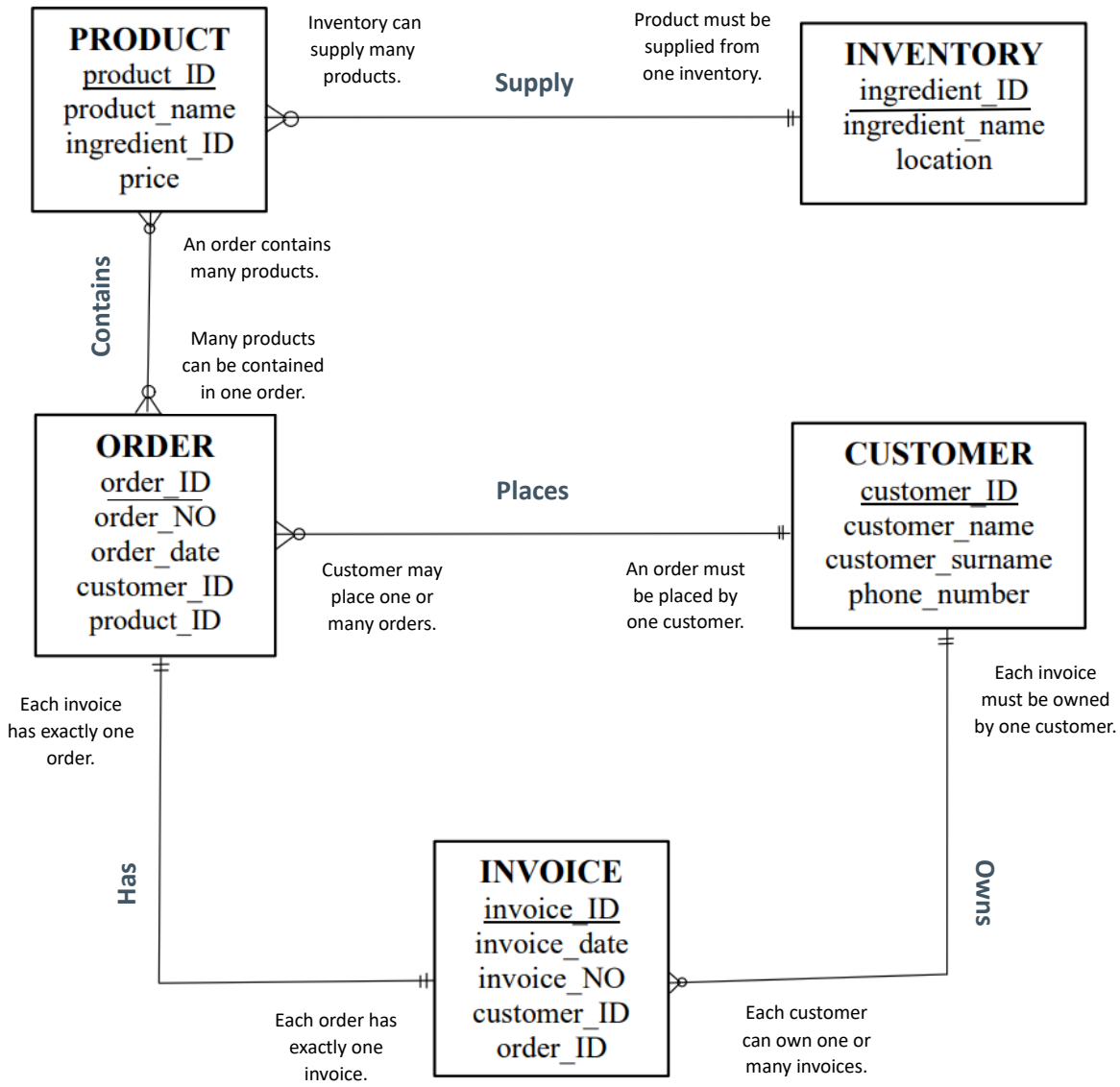
A crowded bakery faces numerous challenges every day in managing its operations, including keeping track of orders, customers, and inventory ingredients. These tasks can become overwhelming without the right tools and systems in place. That is why a well-designed and efficient database management system is crucially required to streamline processes, improve efficiency, and ensure smooth operations.

Entities:

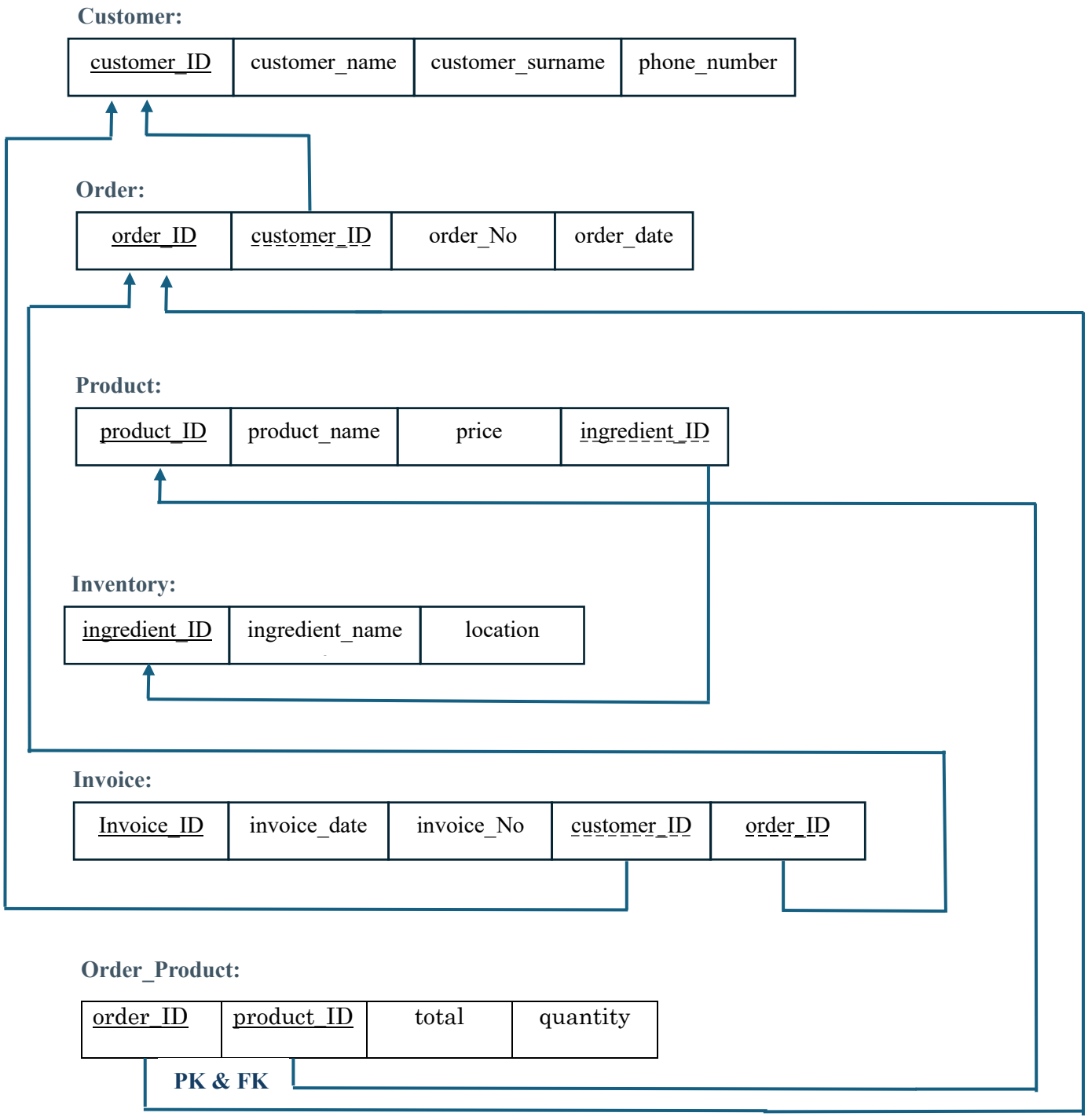
- Customer
- Orders
- Product
- Invoice
- Inventory

Phase 2

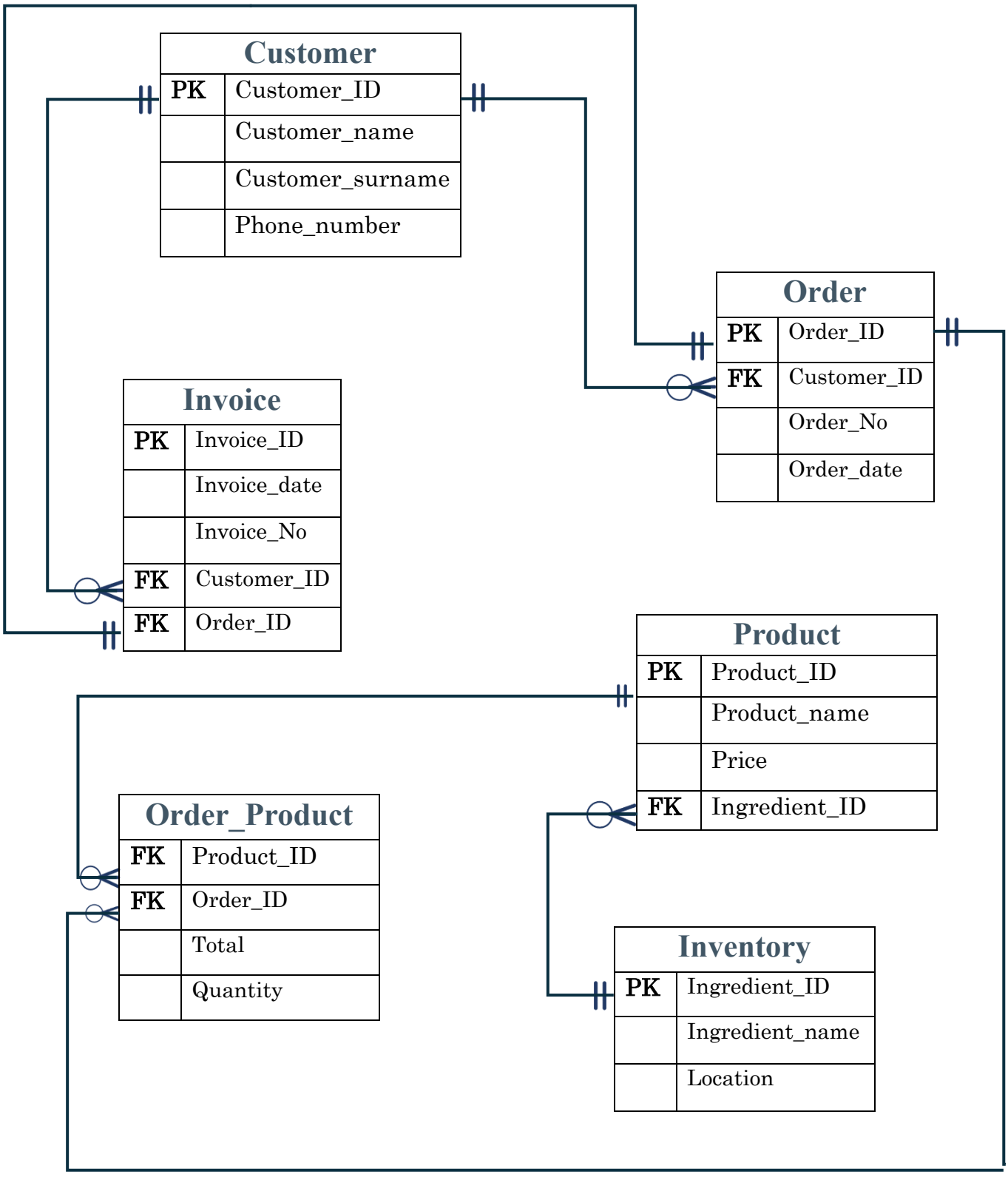
Conceptual ER Model:



Relational Model:



Logical ERD Model:



Phase 3

Unnormalized:

Customer (Customer_ID, customer_name, customer_surname, phone_number,Order(order_ID, order_No, order_date), Product(product_ID, product_name, price), Inventory(Ingredient _ID, ingredient_name, location), Invoice(Invoice_ID, invoice_date, invoice_No))

Functional dependency:

Customer_ID ⟶ customer_name, customer_surname, phone_number

order_ID ⟶ order_No, order_date

product_ID ⟶ product_name, price

Ingredient_ID ⟶ ingredient_name, location

Invoice_ID ⟶ invoice_date, invoice_No

Normalization:

1NF (First Normal Form):

- No multivalued or composite attributes.
- Every attribute value is atomic (has one value).

Customer (customer_ID, customer_name, customer_surname, phone_number, Invoice_ID, invoice_date, invoice_No, order_ID, order_No, order_date)

Order (order_ID, order_No, order_date, customer_ID, customer_name, customer_surname, phone_number, product_ID, product_name, price, Invoice_ID, invoice_date, invoice_No)

Product (product_ID, product_name, price, ingredient _ID, ingredient_name, location, ingredient _ID, ingredient_name, location, order_ID, order_No, order_date)

2NF (Second Normal Form):

- Every non-key attribute is **fully dependent on the primary key**
- No partial functional dependencies.

Customer (customer_ID, customer_name, customer_surname, phone_number)

Order (order_ID, order_No, order_date, customer_ID#)

Product (product_ID, product_name, price, ingredient _ID#)

Order_product (product_ID, order_ID, total, quantity)

Inventory (ingredient_ID, ingredient_name, location)

Invoice (Invoice_ID, invoice_date, invoice_No, order_ID#, customer_ID#)

3NF (Third Normal Form):

- **No transitive dependencies** between the attributes

In our current schema, there are no transitive dependencies.
Each non-key attribute depends only on the primary key.

Customer (customer_ID, customer_name, customer_surname, phone_number)

Order (order_ID, order_No, order_date, customer_ID#)

Product (product_ID, product_name, price, ingredient _ID#)

Order_product (product_ID, order_ID, total, quantity)

Inventory (ingredient_ID, ingredient_name, location)

Invoice (Invoice_ID, invoice_date, invoice_No, order_ID#, customer_ID#)

Phase 4

➢ Creating Tables:

1. Customer table:

```
CREATE TABLE CUSTOMER (
    CUSTOMER_ID NUMBER(7) PRIMARY KEY NOT NULL,
    CUSTOMER_NAME VARCHAR2(20),
    CUSTOMER_SURNAME VARCHAR2(20),
    PHONE_NUMBER VARCHAR2(10)
);
```

2. Orders table:

```
CREATE TABLE ORDERS(
    ORDER_ID NUMBER(7) PRIMARY KEY NOT NULL,
    ORDER_NO NUMBER(10) NOT NULL,
    ORDER_DATE DATE DEFAULT SYSDATE,
    CUSTOMER_ID NUMBER(7),
    CONSTRAINT customer_FK1 FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER (CUSTOMER_ID)
);
```

3. Product table:

```
CREATE TABLE PRODUCT(
    PRODUCT_ID NUMBER(7) PRIMARY KEY NOT NULL,
    PRODUCT_NAME VARCHAR2(30),
    PRICE DECIMAL(8,2),
    INGREDIENT_ID NUMBER(7),
    CONSTRAINT ingredient_FK FOREIGN KEY (INGREDIENT_ID) REFERENCES INVENTORY (INGREDIENT_ID)
);
```

4. Order_product table:

```
CREATE TABLE ORDER_PRODUCT(
    ORDER_ID NUMBER(7) NOT NULL,
    PRODUCT_ID NUMBER(7) NOT NULL,
    QUANTITY NUMBER (8),
    TOTAL DECIMAL (8,2),
    CONSTRAINT order_product_FK PRIMARY KEY (ORDER_ID, PRODUCT_ID),
    CONSTRAINT order_FK1 FOREIGN KEY (ORDER_ID) REFERENCES ORDERS (ORDER_ID),
    CONSTRAINT product_FK FOREIGN KEY (PRODUCT_ID) REFERENCES PRODUCT (PRODUCT_ID)
);
```

5. Inventory table:

```
CREATE TABLE INVENTORY(
    INGREDIENT_ID NUMBER(7) PRIMARY KEY NOT NULL,
    INGREDIENT_NAME VARCHAR2(20),
    LOCATION VARCHAR2(20)
);
```

6. Invoice table:

```
CREATE TABLE INVOICE(
    INVOICE_ID NUMBER(7) PRIMARY KEY NOT NULL,
    INVOICE_DATE DATE DEFAULT SYSDATE,
    INVOICE_NO NUMBER(10),
    CUSTOMER_ID NUMBER(7) NOT NULL,
    ORDER_ID NUMBER(7) NOT NULL,
    CONSTRAINT customer_FK2 FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER (CUSTOMER_ID),
    CONSTRAINT order_FK2 FOREIGN KEY (ORDER_ID) REFERENCES ORDERS (ORDER_ID)
);
```

➢ Inserting values:

1. Customer table:

```
INSERT INTO CUSTOMER VALUES(2498023,'Sarah','Alshahrani','0549862461');
INSERT INTO CUSTOMER VALUES(5389012,'Nora','Quser','0568734592');
INSERT INTO CUSTOMER VALUES(2387128,'Jory','Alguthmi','0549278431');
INSERT INTO CUSTOMER VALUES(2403674,'Jana','Alghamdi','0568465720');
INSERT INTO CUSTOMER VALUES(2987654,'Bayan','Alhabib','0545467328');
```

CUSTOMER_ID	CUSTOMER_NAME	CUSTOMER_SURNAME	PHONE_NUMER
2498023	Sarah	Alshahrani	549862461
5389012	Nora	Quser	568734592
2387128	Jory	Alguthmi	549278431
2403674	Jana	Alghamdi	568465720
2987654	Bayan	Alhabib	545467328

Download CSV

5 rows selected.

2. Inventory table:

```
INSERT INTO INVENTORY VALUES(100, 'FLOUR', 'JEDDAH');
INSERT INTO INVENTORY VALUES(101, 'SUGAR', 'JEDDAH');
INSERT INTO INVENTORY VALUES(103, 'EGGS', 'JEDDAH');
INSERT INTO INVENTORY VALUES(104, 'BUTTER', 'JEDDAH');
INSERT INTO INVENTORY VALUES(105, 'MILK', 'JEDDAH');
INSERT INTO INVENTORY VALUES(106, 'VANILLA', 'JEDDAH');
INSERT INTO INVENTORY VALUES(108, 'CHOCOLATE', 'JEDDAH');
```

INGREDIENT_ID	INGREDIENT_NAME	LOCATION
100	FLOUR	JEDDAH
101	SUGAR	JEDDAH
102	BROWN SUGAR	JEDDAH
103	EGGS	JEDDAH
104	BUTTER	JEDDAH
105	MILK	JEDDAH
106	VANILLA	JEDDAH
108	CHOCOLATE	JEDDAH

Download CSV

8 rows selected.

3. Orders table:

```
INSERT INTO ORDERS VALUES (33684, 23, SYSDATE,2498023);
INSERT INTO ORDERS VALUES (34682, 12, SYSDATE,5389012);
INSERT INTO ORDERS VALUES (74825, 15, SYSDATE,2387128);
INSERT INTO ORDERS VALUES (37739, 50, SYSDATE,2987654);
INSERT INTO ORDERS VALUES (23756, 36, SYSDATE,2403674);
```

ORDER_ID	ORDER_NO	ORDER_DATE	CUSTOMER_ID
33684	23	02-MAY-24	2498023
34682	12	02-MAY-24	5389012
74825	15	02-MAY-24	2387128
37739	50	02-MAY-24	2987654
23756	36	02-MAY-24	2403674

Download CSV

5 rows selected.

4. Invoice table:

```
INSERT INTO INVOICE VALUES (33684, SYSDATE, 1,2498023, 33684);
INSERT INTO INVOICE VALUES (37652, SYSDATE, 2,5389012, 34682);
INSERT INTO INVOICE VALUES (54564, SYSDATE, 3,2387128, 74825);
INSERT INTO INVOICE VALUES (74543, SYSDATE, 4,2987654, 37739);
INSERT INTO INVOICE VALUES (35643, SYSDATE, 5,2403674, 23756);
```

INVOICE_ID	INVOICE_DATE	INVOICE_NO	CUSTOMER_ID	ORDER_ID
33684	02-MAY-24	1	2498023	33684
37652	02-MAY-24	2	5389012	34682
54564	02-MAY-24	3	2387128	74825
74543	02-MAY-24	4	2987654	37739
35643	02-MAY-24	5	2403674	23756

5. product table:

```
68  INSERT INTO PRODUCT VALUES(5678349, 'Banana bread', 35, 100);
69  INSERT INTO PRODUCT VALUES(2894762, 'Tiramisu', 45.5, 101);
70  INSERT INTO PRODUCT VALUES(3849765, 'Raspberry cheesecake', 50, 104);
71  INSERT INTO PRODUCT VALUES(6579345, 'Blueberry Muffin', 30.75, 105);
72  INSERT INTO PRODUCT VALUES(9835410, 'Chocolate fudge cake', 40, 108);
```

PRODUCT_ID	PRODUCT_NAME	PRICE	INGREDIENT_ID
5678349	Banana bread	35	100
2894762	Tiramisu	45.5	101
3849765	Raspberry cheesecake	50	104
6579345	Blueberry Muffin	30.75	105
9835410	Chocolate fudge cake	40	108

Download CSV

5 rows selected.

6. Order_product table:

```
83  INSERT INTO ORDER_PRODUCT VALUES (33684, 5678349, 2, 70.00);
84  INSERT INTO ORDER_PRODUCT VALUES (34682, 3849765, 3, 150.00);
85  INSERT INTO ORDER_PRODUCT VALUES (74825, 2894762, 1, 45.50);
86  INSERT INTO ORDER_PRODUCT VALUES (37739, 9835410, 2, 80.00);
87  INSERT INTO ORDER_PRODUCT VALUES (23756, 6579345, 2, 61.50);
```

ORDER_ID	PRODUCT_ID	QUANTITY	TOTAL
33684	5678349	2	70
34682	3849765	3	150
74825	2894762	1	45.5
37739	9835410	2	80
23756	6579345	2	61.5

Download CSV

5 rows selected.

➢ Data Retrieval Queries:

1. Using where:

```
1  SELECT PRODUCT_NAME FROM PRODUCT WHERE INGREDIENT_ID = 101;
```

PRODUCT_NAME
Tiramisu

2. Using Group by, Order by and Aggregate functions:

```
1  SELECT PRODUCT_NAME, MAX(PRICE) AS MAXIMUM_PRICE FROM PRODUCT
2  GROUP BY PRODUCT_NAME
3  ORDER BY MAXIMUM_PRICE DESC;
4
```

PRODUCT_NAME	MAXIMUM_PRICE
Raspberry cheesecake	50
Tiramisu	45.5
Chocolate fudge cake	40
Banana bread	35
Blueberry Muffin	30.75

Download CSV

5 rows selected.

3. Subquery:

```
SELECT ORDER_NO FROM ORDERS WHERE ORDER_ID = ANY
    (SELECT ORDER_ID FROM ORDER_PRODUCT WHERE QUANTITY = 2)
```

ORDER_NO
36
23
50

Download CSV

3 rows selected.

4. Left join query

```
SELECT PRODUCT_NAME, PRICE, INGREDIENT_NAME
FROM PRODUCT
LEFT JOIN INVENTORY
ON PRODUCT.INGREDIENT_ID = INVENTORY.INGREDIENT_ID
```

PRODUCT_NAME	PRICE	INGREDIENT_NAME
Banana bread	35	FLOUR
Tiramisu	45.5	SUGAR
Raspberry cheesecake	50	BUTTER
Blueberry Muffin	30.75	MILK
Chocolate fudge cake	40	CHOCOLATE

Download CSV

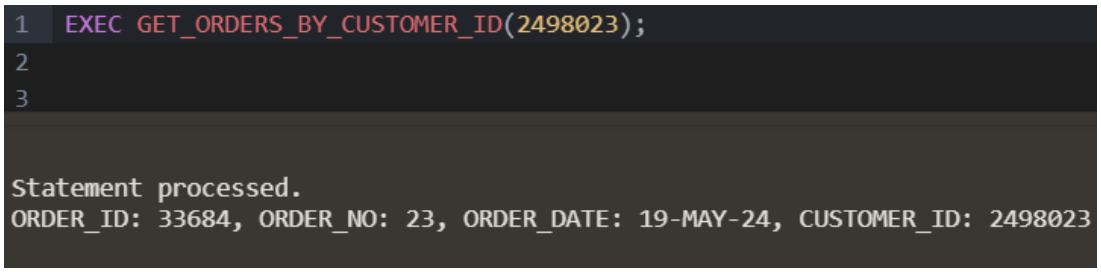
➢ Stored Procedures:

1. Parameter query based select query stored procedure

This procedure prints the order details for a given customer ID

```
CREATE OR REPLACE PROCEDURE GET_ORDERS_BY_CUSTOMER_ID (
    p_CUSTOMER_ID IN NUMBER
)
AS
BEGIN
    FOR rec IN (
        SELECT
            ORDER_ID,
            ORDER_NO,
            ORDER_DATE,
            CUSTOMER_ID
        FROM
            ORDERS
        WHERE
            CUSTOMER_ID = p_CUSTOMER_ID
    ) LOOP
        DBMS_OUTPUT.PUT_LINE('ORDER_ID: ' || rec.ORDER_ID || ', ORDER_NO: ' || rec.ORDER_NO || ', ORDER_DATE: ' || rec.ORDER_DATE || ', CUSTOMER_ID: ' || rec.CUSTOMER_ID);
    END LOOP;
END;
/

Procedure created.
```



```
EXEC GET_ORDERS_BY_CUSTOMER_ID(2498023);


Statement processed.
ORDER_ID: 33684, ORDER_NO: 23, ORDER_DATE: 19-MAY-24, CUSTOMER_ID: 2498023
```

2. Update query based stored procedure:

We created this procedure to update products price based on product name

```
CREATE OR REPLACE PROCEDURE UpdatePrice (
    p_ProductName IN VARCHAR2,
    p_NewPrice IN DECIMAL
)
IS
BEGIN
    UPDATE PRODUCT
    SET PRICE = p_NewPrice
    WHERE PRODUCT_NAME = p_ProductName;

    COMMIT;
END;
```

```
Procedure created.
```

```
EXECUTE UpdatePrice('Banana bread', 38);
EXECUTE UpdatePrice('Blueberry Muffin', 30);
SELECT * FROM PRODUCT;
```

```
Statement processed.

Statement processed.
```

PRODUCT_ID	PRODUCT_NAME	PRICE	INGREDIENT_ID
5678349	Banana bread	38	100
2894762	Tiramisu	45.5	101
3849765	Raspberry cheesecake	50	104
6579345	Blueberry Muffin	30	105
9835410	Chocolate fudge cake	40	108

This procedure displays the total that's greater than the given total

NOTE: This is an addition parameter query procedure that is not required but we added it

```
CREATE OR REPLACE PROCEDURE TotalCheck (MinTotal NUMBER) AS
    CURSOR POINTER IS
        SELECT ORDER_ID, TOTAL
        FROM ORDER_PRODUCT
        WHERE TOTAL > MinTotal;
BEGIN
    FOR var IN POINTER LOOP
        DBMS_OUTPUT.PUT_LINE('Order ID is: ' || var.ORDER_ID || ' ' || 'The Total is: ' || var.TOTAL);
    END LOOP;
END TotalCheck;

Procedure created.
```

```
EXEC TotaLCheck (75);


Statement processed.
Order ID is: 34682 The Total is: 150
Order ID is: 37739 The Total is: 80
```