Project – Technical Report (Step 1 and 2)

BUAN 6320

Group 8:

Cathy Dominguez,

Jacob Elwell,

Sagar Hagawane,

Albina Jahir,

Virinchi Kopparam,

Ritesh Kulkarni,

Mihir Santosh Nevpurkar,

Sonal Seth

JSOM – UTD

Introduction

This document describes the steps that were taken to deliver the project. The project consisted of designing and implementing a database that stores information for a retail store, Gap. This document was developed by eight students with the intent to create a database for Gap. The database reflect the paths the goods take to reach the customer from a vendor.

Overview

Gap is a big retail store that sales various items, not only clothing. This database is only a part of their vendor to customer system, however the store has much more data that we won't be discussing in this document.

Assumptions and Special Considerations

It is assumed that an order can only have one product placed in it for easy accounting purposes and rules by this GAP store. For example, a T-shirt, an electronic item, a pair of pants, etc. all must be a part of their own separate order. However, a T-shirt, electronics, etc. can be part of multiple orders. Giving it that 1:M relationship between orders and products. Also, at this Gap distribution channel, a warehouse only has one designated vendor they work with to provide merchandise. However, a vendor could service multiple warehouses at the same time, giving it the 1:M relationship.

Requirements and Definition Document

The five entities that will be discussed are: Customer, Orders, Product, Warehouse and Vendor. Those entities and attributes are described below:

Customer Entity

The first entity is Customer where we will house Gap's customer data. The data that will be housed in this entity contains the 6 attributes: Customer_ID, First_Name, Last_Name , Address, Phone_no, and Email. The primary key in this entity is Customer_ID. There is no foreign key.

Orders Entity

The second entity is Orders where we will house Gap's orders for each transaction. The data that will be housed in this entity contains the 7 attributes: Order_ID, Product_ID, Customer_ID, Total_Price, Order_Date, Delivery_Address, and Payment_Method. The primary key in this entity is Order_ID. The foreign key is Customer_ID and Product_ID.

Payment_method could be cash, card, or check as a way of paying for your order.

Product Entity

The third entity is the Product table where we will house Gap's products sold. The data that will be housed in this entity contains the 5 attributes: Product_ID, Product_Type, Product_Description,Unit_Price, and Warehouse_ID. The primary key in this entity is Product_ID. The foreign key in this entity comes from the Warehouse Table and Warehouse_ID.

Product_type could be clothing, shoes, groceries, electronics, or other. Product_description is the description of the type of product, for example other could be chocolate or clothing could be white t-shirt.

Warehouse Entity

The fourth entity is Warehouse where we will house Gap's warehouse information for the store. The data that will be housed in this entity contains the 5 attributes: Warehouse_ID, Vendor_Code, Warehouse_Address, Warehouse_Region, and Email. The primary key in this entity is Warehouse_ID. The foreign key in this entity comes from the Vendor Table and it is Vendor_Code.

Warehouse_Region is where it is located to service the Stores in that area (DFW, Plano, Etc).

Vendor Entity

The fifth and last entity is Vendor where we will house Gap's vendor's information. Its primary key is Vendor_code which provides a unique code for each vendor. The data that will be housed in this entity contains five attributes: Vendor_code, Vendor_name, Address, Email, and Phone_no. The primary key in this entity is Vendor_code. There is no foreign key.

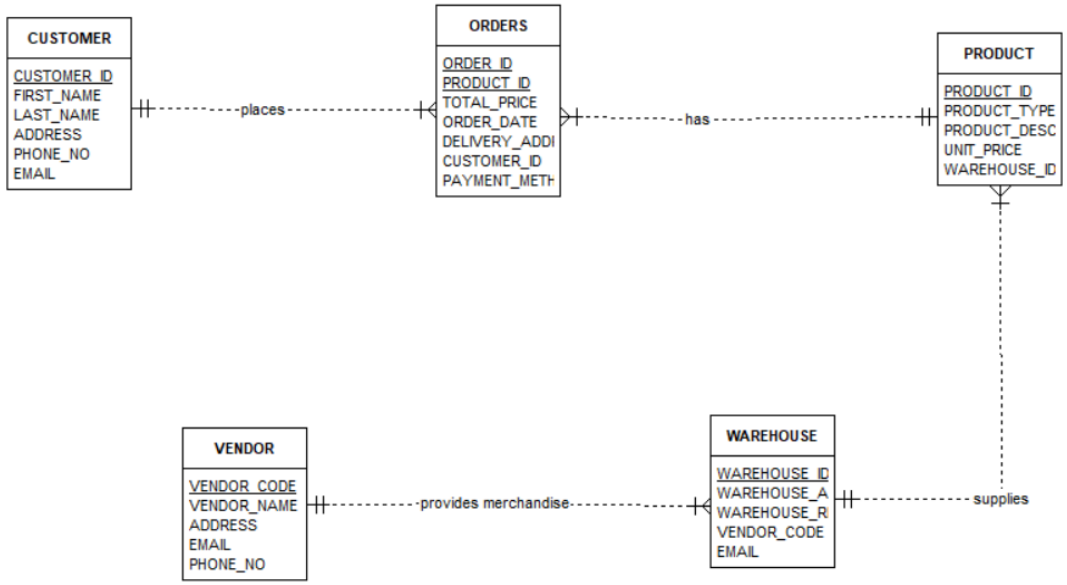
Relationship and Cardinality Description

Entity	Cardinality	Relationship	Business Rules	Entity

Customer	1:M	Customer **Places** Orders	• A customer can place one or many orders. • Each order is placed by a customer. • An order is placed by one customer • A customer must place at least one order to be considered a customer	Orders
Orders	1:M	Orders **has** Products	• An order has one and only one product (stated in assumptions) • A product can be in one, or many orders. • A product must be apart of an order to be considered as an order.	Product
Warehouse	1:M	Warehouse **supplies** products	• A Warehouse can supply one or many products. • A product is supplied by a warehouse.	Product

			• A product must be supplied from a warehouse to be sold. • A product can come from one warehouse.	
Vendor	1:M	Vendor **provides merchandise** to Warehouse	• A Vendor provides merchandise to one or many warehouses. • A Warehouse only has one designated vendor . • A Vendor must have one warehouse designated to them to be considered a vendor.	Warehouse

Entity Relationship Diagram



DDL Source Code

/* 1. DROP statements to clean up objects from previous run */

-- Triggers - Trigger

DROP TRIGGER TRG_PRODUCT;

DROP TRIGGER TRG_ORDERS;

DROP TRIGGER TRG_CUSTOMER;

DROP TRIGGER TRG_WAREHOUSE;

DROP TRIGGER TRG_VENDOR;

-- Sequences

DROP SEQUENCE SEQ_PRODUCT_ID;

DROP SEQUENCE SEQ_ORDER_ID;

DROP SEQUENCE SEQ_CUSTOMER_ID;

DROP SEQUENCE SEQ_WAREHOUSE_ID;

DROP SEQUENCE SEQ_VENDOR_CODE;

-- Views

DROP VIEW CUSTOMERINFO;

```
-- Tables
DROP TABLE PRODUCT;
DROP TABLE ORDERS;
DROP TABLE CUSTOMER;
DROP TABLE WAREHOUSE;
DROP TABLE VENDOR;

/* 2.(DDL) CREATE statements to create new objects */
-- 2.1 Create Tables

-- Customer
CREATE TABLE CUSTOMER
( CUSTOMER_ID INTEGER NOT NULL,
FIRST_NAME VARCHAR2 (30) NOT NULL,
LAST_NAME VARCHAR2 (30) NOT NULL,
ADDRESS VARCHAR (30),
PHONE_NO NUMBER (30),
EMAIL VARCHAR (30),
CONSTRAINT PK_CUSTOMER PRIMARY KEY (CUSTOMER_ID)
);

-- Order
CREATE TABLE ORDERS
( ORDER_ID INTEGER NOT NULL,
PRODUCT_ID INTEGER NOT NULL,
CUSTOMER_ID INTEGER NOT NULL,
TOTAL_PRICE VARCHAR2 (30) NOT NULL,--alter the type to integer
ORDER_DATE DATE NOT NULL,
DELIVERY_ADDRESS VARCHAR (30),
PAYMENT_METHOD VARCHAR2 (30),
```

```
CONSTRAINT PK_ORDER PRIMARY KEY (ORDER_ID),
CONSTRAINT FK_CUSTOMER_ID FOREIGN KEY (CUSTOMER_ID) REFERENCES
CUSTOMER
);

--VENDOR
CREATE TABLE VENDOR
( VENDOR_CODE INTEGER NOT NULL,
VENDOR_NAME VARCHAR2 (30) NOT NULL,
ADDRESS VARCHAR2 (30) NOT NULL,
EMAIL VARCHAR2 (30) NOT NULL,
PHONE_NO NUMBER (30),
CONSTRAINT PK_VENDOR PRIMARY KEY (VENDOR_CODE)
);

--WAREHOUSE
CREATE TABLE WAREHOUSE
( WAREHOUSE_ID INTEGER NOT NULL,
VENDOR_CODE INTEGER,
WAREHOUSE_ADDRESS VARCHAR2 (30) NOT NULL,
WAREHOUSE_REGION VARCHAR2 (30) NOT NULL,
EMAIL VARCHAR2 (30),
CONSTRAINT PK_WAREHOUSE PRIMARY KEY (WAREHOUSE_ID),
CONSTRAINT FK_VENDOR_CODE FOREIGN KEY (VENDOR_CODE)
REFERENCES VENDOR
);

--Product
CREATE TABLE PRODUCT
( PRODUCT_ID INTEGER NOT NULL,
WAREHOUSE_ID INTEGER NOT NULL,
```

```
PRODUCT_TYPE VARCHAR2 (30) NOT NULL,
PRODUCT_DESCRIPTION VARCHAR2 (30) NOT NULL,
UNIT_PRICE INTEGER NOT NULL,
CONSTRAINT PK_PRODUCT PRIMARY KEY (PRODUCT_ID),
CONSTRAINT FK_WAREHOUSE_ID FOREIGN KEY (WAREHOUSE_ID)
REFERENCES WAREHOUSE
);

-- 2.2 Alter Table - add constaraint

ALTER TABLE ORDERS MODIFY TOTAL_PRICE INTEGER;

-- 2.3 Create Sequences
CREATE SEQUENCE SEQ_CUSTOMER_ID
INCREMENT BY 1
START WITH 10000000000
NOMAXVALUE
MINVALUE 0
NOCACHE;


CREATE SEQUENCE SEQ_ORDER_ID
INCREMENT BY 1
START WITH 20000000000
NOMAXVALUE
MINVALUE 0
NOCACHE;

CREATE SEQUENCE SEQ_PRODUCT_ID
INCREMENT BY 1
START WITH 30000000000
```

```
NOMAXVALUE
MINVALUE 0
NOCACHE;

CREATE SEQUENCE SEQ_WAREHOUSE_ID
INCREMENT BY 1
START WITH 40000000000
NOMAXVALUE
MINVALUE 0
NOCACHE;

CREATE SEQUENCE SEQ_VENDOR_CODE
INCREMENT BY 1
START WITH 50000000000
NOMAXVALUE
MINVALUE 0
NOCACHE;

-- 2.4 Create Triggers
CREATE OR REPLACE TRIGGER TRG_CUSTOMER
BEFORE INSERT OR UPDATE ON CUSTOMER
FOR EACH ROW
BEGIN
   IF INSERTING THEN
      IF :NEW.CUSTOMER_ID IS NULL THEN
         :NEW.CUSTOMER_ID := SEQ_CUSTOMER_ID.NEXTVAL;
      END IF;
   END IF;
END;
/
```

```
CREATE OR REPLACE TRIGGER TRG_ORDERS
BEFORE INSERT OR UPDATE ON ORDERS
FOR EACH ROW
BEGIN
   IF INSERTING THEN
      IF :NEW.ORDER_ID IS NULL THEN
         :NEW.ORDER_ID := SEQ_ORDER_ID.NEXTVAL;
      END IF;
   END IF;
END;

/
CREATE OR REPLACE TRIGGER TRG_VENDOR
BEFORE INSERT OR UPDATE ON VENDOR
FOR EACH ROW
BEGIN
   IF INSERTING THEN
      IF :NEW.VENDOR_CODE IS NULL THEN
         :NEW.VENDOR_CODE := SEQ_VENDOR_CODE.NEXTVAL;
      END IF;
   END IF;
END;
/
CREATE OR REPLACE TRIGGER TRG_WAREHOUSE
BEFORE INSERT OR UPDATE ON WAREHOUSE
FOR EACH ROW
BEGIN
   IF INSERTING THEN
      IF :NEW.WAREHOUSE_ID IS NULL THEN
         :NEW.WAREHOUSE_ID := SEQ_WAREHOUSE_ID.NEXTVAL;
      END IF;
```

```
   END IF;
END;
/
CREATE OR REPLACE TRIGGER TRG_PRODUCT
BEFORE INSERT OR UPDATE ON PRODUCT
FOR EACH ROW
BEGIN
   IF INSERTING THEN
      IF :NEW.PRODUCT_ID IS NULL THEN
         :NEW.PRODUCT_ID := SEQ_PRODUCT_ID.NEXTVAL;
      END IF;
   END IF;
END;
/

-- 2.5 Create views
CREATE OR REPLACE VIEW CUSTOMERINFO AS
SELECT CUSTOMER_ID, FIRST_NAME, LAST_NAME, ADDRESS FROM
CUSTOMER;
```

DML and Query Source Code

```
/* 3.(DML) Insert data into tables*/

-- 3.1 Insert data into CUSTOMER Table
-- Customer
INSERT INTO CUSTOMER ( FIRST_NAME, LAST_NAME, ADDRESS, PHONE_NO,
EMAIL) VALUES ( 'MIHIR', 'NEVPURKAR', '7575 Frankford ROAD', '222333444',
'mihir@gmail.com');
INSERT INTO CUSTOMER ( FIRST_NAME, LAST_NAME, ADDRESS, PHONE_NO,
EMAIL) VALUES ('SAGAR', 'HAGAWANE', '7421 FRANKFORD ROAD', '222555666',
'sagar@gmail.com');
```

INSERT INTO CUSTOMER (FIRST_NAME, LAST_NAME, ADDRESS, PHONE_NO, EMAIL) VALUES ('RITESH', 'KULKARNI', '7421 Frankford ROAD', '555333444', 'ritesh@gmail.com');
INSERT INTO CUSTOMER (FIRST_NAME, LAST_NAME, ADDRESS, PHONE_NO, EMAIL) VALUES ('JOHN', 'WATSON', '7575 Frankford ROAD', '242333444', 'john@gmail.com');
INSERT INTO CUSTOMER (FIRST_NAME, LAST_NAME, ADDRESS, PHONE_NO, EMAIL) VALUES ('SONAL', 'SETH', '7576 Frankford ROAD', '242333441', 'sonal@gmail.com');
INSERT INTO CUSTOMER (FIRST_NAME, LAST_NAME, ADDRESS, PHONE_NO, EMAIL) VALUES ('JAY', 'SHETTY', '7577 Frankford ROAD', '242333442', 'jay@gmail.com');
INSERT INTO CUSTOMER (FIRST_NAME, LAST_NAME, ADDRESS, PHONE_NO, EMAIL) VALUES ('RAHUL', 'KUMAR', '7578 Frankford ROAD', '242333443', 'rahul@gmail.com');
INSERT INTO CUSTOMER (FIRST_NAME, LAST_NAME, ADDRESS, PHONE_NO, EMAIL) VALUES ('ROCKY', 'BHAI', '7579 Frankford ROAD', '242333445', 'rocky@gmail.com');
INSERT INTO CUSTOMER (FIRST_NAME, LAST_NAME, ADDRESS, PHONE_NO, EMAIL) VALUES ('SANJAY', 'DUTT', '7580 Frankford ROAD', '242333446', 'sanjay@gmail.com');
INSERT INTO CUSTOMER (FIRST_NAME, LAST_NAME, ADDRESS, PHONE_NO, EMAIL) VALUES ('RANBIR', 'KAPOOR', '7581 Frankford ROAD', '242333447', 'ranbir@gmail.com');
INSERT INTO CUSTOMER (FIRST_NAME, LAST_NAME, ADDRESS, PHONE_NO, EMAIL) VALUES ('SUNIL', 'SHETTY', '7582 Frankford ROAD', '242333448', 'sunil@gmail.com');

-- 3.2 Insert data into VENDOR Table
-- Vendor

```
INSERT INTO VENDOR (  VENDOR_NAME, ADDRESS, EMAIL, PHONE_NO)
VALUES ( 'RM SUPPLIERS', 'DALLAS PKWY', 'rm@gmail.com', '777666555');
INSERT INTO VENDOR (  VENDOR_NAME, ADDRESS, EMAIL, PHONE_NO)
VALUES ( 'RK SUPPLIERS', 'DALLAS PKWY', 'rm@gmail.com', '777666555');
INSERT INTO VENDOR (  VENDOR_NAME, ADDRESS, EMAIL, PHONE_NO)
VALUES ( 'MN SUPPLIERS', 'PLANO PKWY', 'mn@gmail.com', '777444555');
INSERT INTO VENDOR (  VENDOR_NAME, ADDRESS, EMAIL, PHONE_NO)
VALUES ( 'SH SUPPLIERS', 'PLANO PKWY', 'sh@gmail.com', '777344555');
INSERT INTO VENDOR (  VENDOR_NAME, ADDRESS, EMAIL, PHONE_NO)
VALUES ( 'AH SUPPLIERS', 'DFW PKWY', 'ah@gmail.com', '776344555');
INSERT INTO VENDOR (  VENDOR_NAME, ADDRESS, EMAIL, PHONE_NO)
VALUES ( 'SS SUPPLIERS', 'PLANO PKWY', 'ss@gmail.com', '777544555');
INSERT INTO VENDOR (  VENDOR_NAME, ADDRESS, EMAIL, PHONE_NO)
VALUES ( 'SSA SUPPLIERS', 'PLANO PKWY', 'ssa@gmail.com', '777544556');
INSERT INTO VENDOR (  VENDOR_NAME, ADDRESS, EMAIL, PHONE_NO)
VALUES ( 'SSB SUPPLIERS', 'DFW PKWY', 'ssb@gmail.com', '777544557');
INSERT INTO VENDOR (  VENDOR_NAME, ADDRESS, EMAIL, PHONE_NO)
VALUES ( 'SSC SUPPLIERS', 'DFW PKWY', 'ssc@gmail.com', '777544558');
INSERT INTO VENDOR (  VENDOR_NAME, ADDRESS, EMAIL, PHONE_NO)
VALUES ( 'SSD SUPPLIERS', 'PLANO PKWY', 'ssd@gmail.com', '777544559');
INSERT INTO VENDOR (  VENDOR_NAME, ADDRESS, EMAIL, PHONE_NO)
VALUES ( 'SSE SUPPLIERS', 'DALLAS PKWY', 'sse@gmail.com', '777544552');

-- 3.3 Insert data into WAREHOUSE Table
-- Warehouse
INSERT INTO WAREHOUSE (  VENDOR_CODE, WAREHOUSE_ADDRESS,
WAREHOUSE_REGION, EMAIL ) VALUES ( 50000000000, 'DALLAS PKWY',
'DALLAS', 'wh4000@gmail.com');
INSERT INTO WAREHOUSE (  VENDOR_CODE, WAREHOUSE_ADDRESS,
WAREHOUSE_REGION, EMAIL ) VALUES ( 50000000001, 'DALLAS PKWY',
'DALLAS', 'wh4001@gmail.com');
```

```
INSERT INTO WAREHOUSE (  VENDOR_CODE, WAREHOUSE_ADDRESS,
WAREHOUSE_REGION, EMAIL ) VALUES ( 50000000002, 'PLANO PKWY',
'PLANO', 'wh4002@gmail.com');
INSERT INTO WAREHOUSE (  VENDOR_CODE, WAREHOUSE_ADDRESS,
WAREHOUSE_REGION, EMAIL ) VALUES ( 50000000003, 'PLANO PKWY',
'PLANO', 'wh4003@gmail.com');
INSERT INTO WAREHOUSE (  VENDOR_CODE, WAREHOUSE_ADDRESS,
WAREHOUSE_REGION, EMAIL ) VALUES ( 50000000004, 'DFW PKWY', 'DFW',
'wh4004@gmail.com');
INSERT INTO WAREHOUSE (  VENDOR_CODE, WAREHOUSE_ADDRESS,
WAREHOUSE_REGION, EMAIL ) VALUES ( 50000000005, 'PLANO PKWY',
'PLANO', 'wh4005@gmail.com');
INSERT INTO WAREHOUSE (  VENDOR_CODE, WAREHOUSE_ADDRESS,
WAREHOUSE_REGION, EMAIL ) VALUES ( 50000000005, 'DFW PKWY', 'DFW',
'wh4006@gmail.com');
INSERT INTO WAREHOUSE (  VENDOR_CODE, WAREHOUSE_ADDRESS,
WAREHOUSE_REGION, EMAIL ) VALUES ( 50000000005, 'PLANO PKWY',
'PLANO', 'wh4007@gmail.com');
INSERT INTO WAREHOUSE (  VENDOR_CODE, WAREHOUSE_ADDRESS,
WAREHOUSE_REGION, EMAIL ) VALUES ( 50000000005, 'DALLAS PKWY',
'DALLAS', 'wh4008@gmail.com');
INSERT INTO WAREHOUSE (  VENDOR_CODE, WAREHOUSE_ADDRESS,
WAREHOUSE_REGION, EMAIL ) VALUES ( 50000000005, 'DFW PKWY', 'DFW',
'wh4009@gmail.com');


-- 3.4 Insert data into PRODUCT Table
-- Product
INSERT INTO PRODUCT (  WAREHOUSE_ID, PRODUCT_TYPE,
PRODUCT_DESCRIPTION, UNIT_PRICE) VALUES (  40000000000, 'CLOTHING',
'SHIRT', 10);
```

```
INSERT INTO PRODUCT (  WAREHOUSE_ID, PRODUCT_TYPE,
PRODUCT_DESCRIPTION, UNIT_PRICE) VALUES (  40000000000, 'CLOTHING',
'PANT', 15);
INSERT INTO PRODUCT (  WAREHOUSE_ID, PRODUCT_TYPE,
PRODUCT_DESCRIPTION, UNIT_PRICE) VALUES (  40000000001, 'ELECTRONICS',
'MOUSE', 50);
INSERT INTO PRODUCT (  WAREHOUSE_ID, PRODUCT_TYPE,
PRODUCT_DESCRIPTION, UNIT_PRICE) VALUES (  40000000002, 'GROCERY',
'VEGETABLES', 5);
INSERT INTO PRODUCT (  WAREHOUSE_ID, PRODUCT_TYPE,
PRODUCT_DESCRIPTION, UNIT_PRICE) VALUES (  40000000003, 'ELECTRONICS',
'AIRPODS', 90);
INSERT INTO PRODUCT (  WAREHOUSE_ID, PRODUCT_TYPE,
PRODUCT_DESCRIPTION, UNIT_PRICE) VALUES (  40000000003, 'ELECTRONICS',
'LAPTOP', 90);
INSERT INTO PRODUCT (  WAREHOUSE_ID, PRODUCT_TYPE,
PRODUCT_DESCRIPTION, UNIT_PRICE) VALUES (  40000000002, 'GROCERY',
'SOAP', 10);
INSERT INTO PRODUCT (  WAREHOUSE_ID, PRODUCT_TYPE,
PRODUCT_DESCRIPTION, UNIT_PRICE) VALUES (  40000000001, 'ELECTRONICS',
'KEYBOARD', 30);
INSERT INTO PRODUCT (  WAREHOUSE_ID, PRODUCT_TYPE,
PRODUCT_DESCRIPTION, UNIT_PRICE) VALUES (  40000000002, 'CLOTHING',
'JEANS', 25);
INSERT INTO PRODUCT (  WAREHOUSE_ID, PRODUCT_TYPE,
PRODUCT_DESCRIPTION, UNIT_PRICE) VALUES (  40000000000, 'GROCERY',
'JUICE', 15);

-- 3.5 Insert data into ORDERS Table
-- Orders
-- For Apex Users
```

INSERT INTO ORDERS (PRODUCT_ID, CUSTOMER_ID, TOTAL_PRICE, ORDER_DATE, DELIVERY_ADDRESS, PAYMENT_METHOD) VALUES (30000000000, 10000000000, 40, 'OCT-25-2007', '7575 Frankford Road', 'CREDIT CARD');
INSERT INTO ORDERS (PRODUCT_ID, CUSTOMER_ID, TOTAL_PRICE, ORDER_DATE, DELIVERY_ADDRESS, PAYMENT_METHOD) VALUES (30000000001, 10000000000, 90, 'OCT-25-2007', '7575 Frankford Road', 'CREDIT CARD');
INSERT INTO ORDERS (PRODUCT_ID, CUSTOMER_ID, TOTAL_PRICE, ORDER_DATE, DELIVERY_ADDRESS, PAYMENT_METHOD) VALUES (30000000002, 10000000000, 100,'OCT-25-2007', '7575 Frankford Road', 'CREDIT CARD');
INSERT INTO ORDERS (PRODUCT_ID, CUSTOMER_ID, TOTAL_PRICE, ORDER_DATE, DELIVERY_ADDRESS, PAYMENT_METHOD) VALUES (30000000001, 10000000001, 45, 'OCT-31-2007', '7421 Frankford Road', 'CREDIT CARD');
INSERT INTO ORDERS (PRODUCT_ID, CUSTOMER_ID, TOTAL_PRICE, ORDER_DATE, DELIVERY_ADDRESS, PAYMENT_METHOD) VALUES (30000000002, 10000000002, 50, 'DEC-31-2007', '7421 Frankford Road', 'CASH');
INSERT INTO ORDERS (PRODUCT_ID, CUSTOMER_ID, TOTAL_PRICE, ORDER_DATE, DELIVERY_ADDRESS, PAYMENT_METHOD) VALUES (30000000003, 10000000003, 90, 'JAN-31-2008', '7575 Frankford Road', 'DEBIT CARD');
INSERT INTO ORDERS (PRODUCT_ID, CUSTOMER_ID, TOTAL_PRICE, ORDER_DATE, DELIVERY_ADDRESS, PAYMENT_METHOD) VALUES (30000000004, 10000000003, 180,'JAN-31-2008', '7575 Frankford Road', 'DEBIT CARD');
INSERT INTO ORDERS (PRODUCT_ID, CUSTOMER_ID, TOTAL_PRICE, ORDER_DATE, DELIVERY_ADDRESS, PAYMENT_METHOD) VALUES (30000000004, 10000000004, 90, 'OCT-31-2008', '7576 Frankford Road', 'CASH');
INSERT INTO ORDERS (PRODUCT_ID, CUSTOMER_ID, TOTAL_PRICE, ORDER_DATE, DELIVERY_ADDRESS, PAYMENT_METHOD) VALUES (30000000005, 10000000004, 90, 'OCT-31-2008', '7576 Frankford Road', 'CASH');

```
INSERT INTO ORDERS (PRODUCT_ID, CUSTOMER_ID, TOTAL_PRICE,
ORDER_DATE, DELIVERY_ADDRESS, PAYMENT_METHOD) VALUES
(30000000001, 10000000004, 75, 'OCT-31-2008', '7576 Frankford Road', 'CASH');
INSERT INTO ORDERS (PRODUCT_ID, CUSTOMER_ID, TOTAL_PRICE,
ORDER_DATE, DELIVERY_ADDRESS, PAYMENT_METHOD) VALUES
(30000000001, 10000000005, 60, 'AUG-31-2008', '7577 Frankford Road', 'DEBIT CARD');
INSERT INTO ORDERS (PRODUCT_ID, CUSTOMER_ID, TOTAL_PRICE,
ORDER_DATE, DELIVERY_ADDRESS, PAYMENT_METHOD) VALUES
(30000000002, 10000000005, 150,'AUG-31-2008', '7577 Frankford Road', 'DEBIT CARD');
INSERT INTO ORDERS (PRODUCT_ID, CUSTOMER_ID, TOTAL_PRICE,
ORDER_DATE, DELIVERY_ADDRESS, PAYMENT_METHOD) VALUES
(30000000001, 10000000005, 150,'AUG-31-2008', '7577 Frankford Road', 'DEBIT CARD');

-- For SQL Users
--INSERT INTO ORDERS (PRODUCT_ID, CUSTOMER_ID, TOTAL_PRICE,
ORDER_DATE, DELIVERY_ADDRESS, PAYMENT_METHOD) VALUES
(30000000000, 10000000000, 40, '25-OCT-2007', '7575 Frankford Road', 'CREDIT
CARD');
--INSERT INTO ORDERS (PRODUCT_ID, CUSTOMER_ID, TOTAL_PRICE,
ORDER_DATE, DELIVERY_ADDRESS, PAYMENT_METHOD) VALUES
(30000000001, 10000000000, 90, '25-OCT-2007', '7575 Frankford Road', 'CREDIT
CARD');
--INSERT INTO ORDERS (PRODUCT_ID, CUSTOMER_ID, TOTAL_PRICE,
ORDER_DATE, DELIVERY_ADDRESS, PAYMENT_METHOD) VALUES
(30000000002, 10000000000, 100,'25-OCT-2007', '7575 Frankford Road', 'CREDIT
CARD');
--INSERT INTO ORDERS (PRODUCT_ID, CUSTOMER_ID, TOTAL_PRICE,
ORDER_DATE, DELIVERY_ADDRESS, PAYMENT_METHOD) VALUES
(30000000001, 10000000001, 45, '31-OCT-2007', '7421 Frankford Road', 'CREDIT
CARD');
```

```
--INSERT INTO ORDERS (PRODUCT_ID, CUSTOMER_ID, TOTAL_PRICE,
ORDER_DATE, DELIVERY_ADDRESS, PAYMENT_METHOD) VALUES
(30000000002, 10000000002, 50, '31-DEC-2007', '7421 Frankford Road', 'CASH');
--INSERT INTO ORDERS (PRODUCT_ID, CUSTOMER_ID, TOTAL_PRICE,
ORDER_DATE, DELIVERY_ADDRESS, PAYMENT_METHOD) VALUES
(30000000003, 10000000003, 90, '31-JAN-2008', '7575 Frankford Road', 'DEBIT CARD');
--INSERT INTO ORDERS (PRODUCT_ID, CUSTOMER_ID, TOTAL_PRICE,
ORDER_DATE, DELIVERY_ADDRESS, PAYMENT_METHOD) VALUES
(30000000004, 10000000003, 180,'31-JAN-2008', '7575 Frankford Road', 'DEBIT CARD');
--INSERT INTO ORDERS (PRODUCT_ID, CUSTOMER_ID, TOTAL_PRICE,
ORDER_DATE, DELIVERY_ADDRESS, PAYMENT_METHOD) VALUES
(30000000004, 10000000004, 90, '31-OCT-2008', '7576 Frankford Road', 'CASH');
--INSERT INTO ORDERS (PRODUCT_ID, CUSTOMER_ID, TOTAL_PRICE,
ORDER_DATE, DELIVERY_ADDRESS, PAYMENT_METHOD) VALUES
(30000000005, 10000000004, 90, '31-OCT-2008', '7576 Frankford Road', 'CASH');
--INSERT INTO ORDERS (PRODUCT_ID, CUSTOMER_ID, TOTAL_PRICE,
ORDER_DATE, DELIVERY_ADDRESS, PAYMENT_METHOD) VALUES
(30000000001, 10000000004, 75, '31-OCT-2008', '7576 Frankford Road', 'CASH');
--INSERT INTO ORDERS (PRODUCT_ID, CUSTOMER_ID, TOTAL_PRICE,
ORDER_DATE, DELIVERY_ADDRESS, PAYMENT_METHOD) VALUES
(30000000001, 10000000005, 60, '31-AUG-2008', '7577 Frankford Road', 'DEBIT CARD');
--INSERT INTO ORDERS (PRODUCT_ID, CUSTOMER_ID, TOTAL_PRICE,
ORDER_DATE, DELIVERY_ADDRESS, PAYMENT_METHOD) VALUES
(30000000002, 10000000005, 150,'31-AUG-2008', '7577 Frankford Road', 'DEBIT CARD');
--INSERT INTO ORDERS (PRODUCT_ID, CUSTOMER_ID, TOTAL_PRICE,
ORDER_DATE, DELIVERY_ADDRESS, PAYMENT_METHOD) VALUES
(30000000001, 10000000005, 150,'31-AUG-2008', '7577 Frankford Road', 'DEBIT CARD');


--COMMIT;
```

Select * from customer;

Select * from orders;

Select * from vendor;

Select * from warehouse;

Select * from product;

/*4. Write 20 different queries on the Tables created*/

/*

Query 1: Select all columns and all rows from one table (5 points)

Query 2: Select five columns and all rows from one table (5 points)

Query 3: Select all columns from all rows from one view (5 points)

Query 4: Using a join on 2 tables, select all columns and all rows from the tables without the use of a Cartesian product (5 points)

Query 5: Select and order data retrieved from one table (5 points)

Query 6: Using a join on 3 tables, select 5 columns from the 3 tables.

Use syntax that would limit the output to 10 rows (5 points)

Query 7: Select distinct rows using joins on 3 tables (5 points)

Query 8: Use GROUP BY and HAVING in a select statement using one or more tables (5 points)

Query 9: Use IN clause to select data from one or more tables (5 points)

Query 10: Select length of one column from one table (use LENGTH function) (5 points)

Query 11: Delete one record from one table. Use select statements to demonstrate the table contents before and after the DELETE statement. Make sure you use ROLLBACK afterwards so that the data will not be physically removed (5 points)

Query 12: Update one record from one table. Use select statements to demonstrate the table contents before and after the UPDATE statement. Make sure you use ROLLBACK afterwards so that the data will not be physically removed (5 points)

Perform 8 Additional Advanced Queries (40 points)

*/

```
-- Query 1

SELECT * FROM CUSTOMER;

-- QUERY 2

SELECT CUSTOMER_ID, FIRST_NAME, LAST_NAME, ADDRESS, PHONE_NO
FROM CUSTOMER;

-- QUERY 3

SELECT * FROM CUSTOMERINFO;

--QUERY 4

SELECT * FROM CUSTOMER C
INNER JOIN ORDERS O ON C.CUSTOMER_ID = O.CUSTOMER_ID;

-- QUERY 5

SELECT * FROM ORDERS
ORDER BY TOTAL_PRICE DESC;

-- QUERY 6

SELECT C.FIRST_NAME,
C.LAST_NAME,
P.PRODUCT_DESCRIPTION,
O.TOTAL_PRICE,
O.ORDER_DATE
```

```
FROM CUSTOMER C
LEFT JOIN ORDERS O ON C.CUSTOMER_ID = O.CUSTOMER_ID
LEFT JOIN PRODUCT P ON O.PRODUCT_ID = P.PRODUCT_ID;

-- QUERY 7

SELECT DISTINCT C.FIRST_NAME,
C.LAST_NAME,
P.PRODUCT_DESCRIPTION,
O.TOTAL_PRICE,
O.ORDER_DATE
FROM CUSTOMER C
LEFT JOIN ORDERS O ON C.CUSTOMER_ID = O.CUSTOMER_ID
LEFT JOIN PRODUCT P ON O.PRODUCT_ID = P.PRODUCT_ID;

-- QUERY 8

-- FILTER TOTAL SALE OF DIFFERENT PRODUCT TYPES GIVEN TOTAL SALES
IS MORE THAN 50

SELECT P.PRODUCT_TYPE, SUM(O.TOTAL_PRICE) FROM ORDERS O
LEFT JOIN PRODUCT P ON O.PRODUCT_ID = P.PRODUCT_ID
GROUP BY P.PRODUCT_TYPE
HAVING SUM(O.TOTAL_PRICE) > 50;

-- QUERY 9

SELECT *
FROM CUSTOMER C
LEFT JOIN ORDERS O ON C.CUSTOMER_ID = O.CUSTOMER_ID
LEFT JOIN PRODUCT P ON O.PRODUCT_ID = P.PRODUCT_ID
```

```
WHERE P.PRODUCT_TYPE IN ('CLOTHING', 'GROCERY');

-- QUERY 10

SELECT EMAIL, LENGTH(EMAIL) AS EMAIL_LENGHT FROM CUSTOMER;

-- QUERY 11
SELECT * FROM PRODUCT;

DELETE FROM PRODUCT
WHERE WAREHOUSE_ID = 40000000000;

SELECT * FROM PRODUCT;

ROLLBACK;

SELECT * FROM PRODUCT;

-- QUERY 12

SELECT * FROM ORDERS;

UPDATE ORDERS
SET TOTAL_PRICE = 100
WHERE ORDER_ID = 20000000000;

SELECT * FROM ORDERS;

ROLLBACK;

SELECT * FROM ORDERS;
```

-- QUERY 13

-- FIND SALES FROM THE WAREHOUSE WHOSE WARE HOUSE IS IN PLANO REGION AND VENDORS ARE ALSO FROM PLANO

```
SELECT W.WAREHOUSE_ID, SUM(O.TOTAL_PRICE) AS SALE FROM ORDERS O
LEFT JOIN PRODUCT P ON O.PRODUCT_ID = P.PRODUCT_ID
LEFT JOIN WAREHOUSE W ON W.WAREHOUSE_ID = P.WAREHOUSE_ID
LEFT JOIN VENDOR V ON W.VENDOR_CODE = V.VENDOR_CODE
WHERE V.ADDRESS LIKE '%PLANO%' AND W.WAREHOUSE_ADDRESS LIKE
'%PLANO%'
GROUP BY W.WAREHOUSE_ID;
```

-- QUERY 14

-- FIND THE TOTAL SALES DONE THROUGH EACH PAYMENT METHOD AND BEFORE 1ST OCT 2008

```
-- FOR APEX
SELECT PAYMENT_METHOD, SUM(TOTAL_PRICE) AS TOTAL_SALES FROM
ORDERS
WHERE ORDER_DATE < 'OCT-01-2008'
GROUP BY PAYMENT_METHOD;

--FOR SQL
--SELECT PAYMENT_METHOD, SUM(TOTAL_PRICE) AS TOTAL_SALES FROM
ORDERS
--WHERE ORDER_DATE < '01-OCT-2008'
--GROUP BY PAYMENT_METHOD;
```

-- QUERY 15

-- FIND THE TOTAL COUNT OF PRODUCT_TYPE SOLD WHOSE WAREHOUSE IS IN DALLAS

```
SELECT P.PRODUCT_TYPE, COUNT(O.PRODUCT_ID) AS QUANTITIES
FROM ORDERS O
LEFT JOIN PRODUCT P ON O.PRODUCT_ID = P.PRODUCT_ID
LEFT JOIN WAREHOUSE W ON W.WAREHOUSE_ID = P.WAREHOUSE_ID
WHERE W.WAREHOUSE_REGION = 'DALLAS'
GROUP BY P.PRODUCT_TYPE;
```

-- QUERY 16

-- TOTAL SALES OF THE DIFFERENT VENDORS WHEN WAREHOUSE AND VENDOR ADDRESS ARE SAME AND ITS DALLAS

```
SELECT V.VENDOR_NAME, SUM(TOTAL_PRICE) FROM ORDERS O
LEFT JOIN PRODUCT P ON O.PRODUCT_ID = P.PRODUCT_ID
LEFT JOIN WAREHOUSE W ON W.WAREHOUSE_ID = P.WAREHOUSE_ID
LEFT JOIN VENDOR V ON W.VENDOR_CODE = V.VENDOR_CODE
WHERE W.WAREHOUSE_REGION = 'DALLAS' AND V.ADDRESS LIKE
'%DALLAS%'
GROUP BY V.VENDOR_NAME;
```

-- QUERY 17

-- HOW MANY WAREHOUSE FROM EACH REGION MADE SALE AND HOW MANY HAVE NOT MADE ANY SALES

```
SELECT A.WAREHOUSE_REGION,
COUNT(CASE WHEN SALE = 1 THEN SALE END) AS SALES,
COUNT(CASE WHEN SALE = 0 THEN SALE END) AS NO_SALES
FROM
(
SELECT W.WAREHOUSE_REGION, W.WAREHOUSE_ID,
COUNT(DISTINCT CASE WHEN O.ORDER_ID IS NOT NULL THEN
W.WAREHOUSE_ID END) AS SALE,
COUNT(DISTINCT CASE WHEN O.ORDER_ID IS NULL THEN W.WAREHOUSE_ID
END) AS NO_SALE
FROM WAREHOUSE W
LEFT JOIN PRODUCT P ON P.WAREHOUSE_ID = W.WAREHOUSE_ID
LEFT JOIN ORDERS O ON O.PRODUCT_ID  = P.PRODUCT_ID
GROUP BY W.WAREHOUSE_ID, W.WAREHOUSE_REGION
) A
GROUP BY A.WAREHOUSE_REGION;

-- QUERY 18

-- FIND OUT COUNT OF WAREHOUSE FROM EACH REGION WHICH HAVE SOLD
PRODUCT WORTH LESS THAN 500

SELECT W.WAREHOUSE_REGION, SUM(O.TOTAL_PRICE) AS TOTAL_SALES
FROM ORDERS O
LEFT JOIN PRODUCT P
ON O.PRODUCT_ID = P.PRODUCT_ID
LEFT JOIN WAREHOUSE W ON
W.WAREHOUSE_ID = P.WAREHOUSE_ID
GROUP BY W.WAREHOUSE_REGION
HAVING SUM(O.TOTAL_PRICE) < 500 ;
```

```
-- QUERY 19

-- LIST ALL THE REGION WHERE ATLEAST 3 WAREHOUSES ARE THERE

SELECT WAREHOUSE_REGION, COUNT(WAREHOUSE_ID) FROM WAREHOUSE
GROUP BY WAREHOUSE_REGION
HAVING COUNT(WAREHOUSE_ID) >=3;

-- QUERY 20

-- LIST ALL THE CUSTOMERS WHO HAVE BOUGHT BOTH ELECTORNICS AND
GROCERY PRODUCT TYPE

SELECT DISTINCT C.FIRST_NAME, C.LAST_NAME, P1.PRODUCT_TYPE as
P1_Type ,P2.PRODUCT_TYPE as P2_Type
FROM CUSTOMER C
INNER JOIN ORDERS O1 ON O1.CUSTOMER_ID = C.CUSTOMER_ID
INNER JOIN ORDERS O2 ON O1.CUSTOMER_ID = O2.CUSTOMER_ID
LEFT JOIN PRODUCT P1 ON O1.PRODUCT_ID = P1.PRODUCT_ID
LEFT JOIN PRODUCT P2 ON O2.PRODUCT_ID = P2.PRODUCT_ID
WHERE O1.PRODUCT_ID > O2.PRODUCT_ID
AND ((P1.PRODUCT_TYPE = 'GROCERY' AND P2.PRODUCT_TYPE =
'ELECTRONICS') OR (P2.PRODUCT_TYPE = 'GROCERY' AND P1.PRODUCT_TYPE
= 'ELECTRONICS')) ;

/* ------------------------------------------------- VALIDATION ----------------------------------------
------------------ */
select o.*,  (o.total_price/P.unit_price) AS quantity from orders o
left join product p  on o.product_id = p.product_id;
```

```
SELECT C.ADDRESS, O.DELIVERY_ADDRESS FROM CUSTOMER C
LEFT JOIN ORDERS O ON C.CUSTOMER_ID = O.CUSTOMER_ID;

SELECT * FROM ORDERS O
LEFT JOIN CUSTOMER C ON O.CUSTOMER_ID = C.CUSTOMER_ID;
```

DDL Output

```
Trigger dropped.
Trigger dropped.
Trigger dropped.
Trigger dropped.
Trigger dropped.
Sequence dropped.
Sequence dropped.
Sequence dropped.
Sequence dropped.
Sequence dropped.
View dropped.
Table dropped.
Table dropped.
Table dropped.
Table dropped.
Table dropped.
Table created.
Table created.
Table created.
Table created.
Table created.
Table altered.
Sequence created.
Sequence created.
```

Sequence created.

Sequence created.

Sequence created.

Trigger created.

Trigger created.

Trigger created.

Trigger created.

Trigger created.

View created.

DML Output

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

```
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
```

1 row(s) inserted.

1 row(s) inserted.

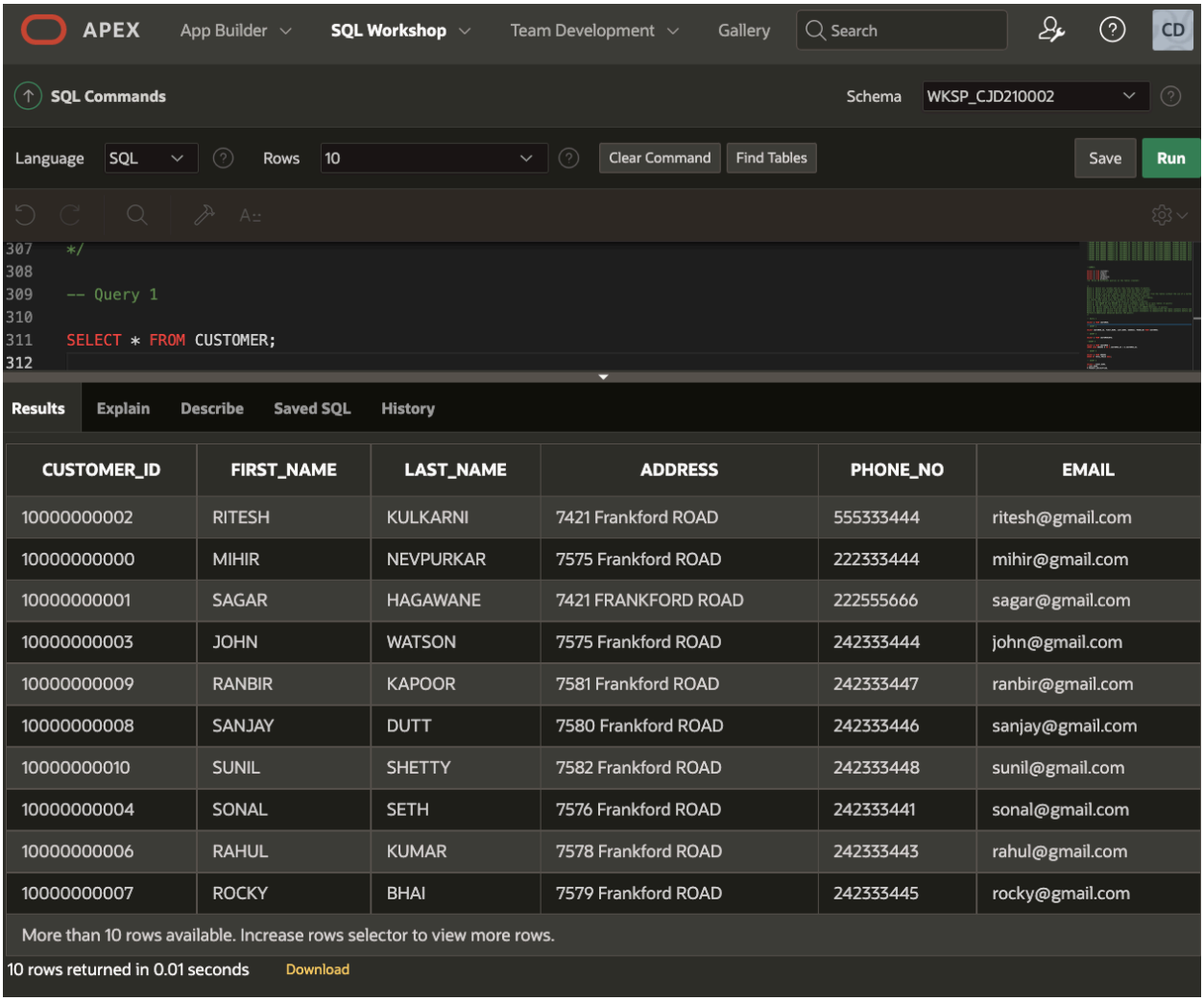
1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

Query Output

QUERY 1:



CUSTOMER_ID	FIRST_NAME	LAST_NAME	ADDRESS	PHONE_NO	EMAIL
10000000002	RITESH	KULKARNI	7421 Frankford ROAD	555333444	ritesh@gmail.com
10000000000	MIHIR	NEVPURKAR	7575 Frankford ROAD	222333444	mihir@gmail.com
10000000001	SAGAR	HAGAWANE	7421 FRANKFORD ROAD	222555666	sagar@gmail.com
10000000003	JOHN	WATSON	7575 Frankford ROAD	242333444	john@gmail.com
10000000009	RANBIR	KAPOOR	7581 Frankford ROAD	242333447	ranbir@gmail.com
10000000008	SANJAY	DUTT	7580 Frankford ROAD	242333446	sanjay@gmail.com
10000000010	SUNIL	SHETTY	7582 Frankford ROAD	242333448	sunil@gmail.com
10000000004	SONAL	SETH	7576 Frankford ROAD	242333441	sonal@gmail.com
10000000006	RAHUL	KUMAR	7578 Frankford ROAD	242333443	rahul@gmail.com
10000000007	ROCKY	BHAI	7579 Frankford ROAD	242333445	rocky@gmail.com

QUERY 2:

APEX App Builder SQL Workshop Team Development Gallery Search

SQL Commands Schema WKSP_CJD210002

Language SQL Rows 10 Clear Command Find Tables Save Run

```
-- QUERY 2

SELECT CUSTOMER_ID, FIRST_NAME, LAST_NAME, ADDRESS, PHONE_NO FROM CUSTOMER;

-- QUERY 3
```

Results Explain Describe Saved SQL History

CUSTOMER_ID	FIRST_NAME	LAST_NAME	ADDRESS	PHONE_NO
10000000002	RITESH	KULKARNI	7421 Frankford ROAD	555333444
10000000000	MIHIR	NEVPURKAR	7575 Frankford ROAD	222333444
10000000001	SAGAR	HAGAWANE	7421 FRANKFORD ROAD	222555666
10000000003	JOHN	WATSON	7575 Frankford ROAD	242333444
10000000009	RANBIR	KAPOOR	7581 Frankford ROAD	242333447
10000000008	SANJAY	DUTT	7580 Frankford ROAD	242333446
10000000010	SUNIL	SHETTY	7582 Frankford ROAD	242333448
10000000004	SONAL	SETH	7576 Frankford ROAD	242333441
10000000006	RAHUL	KUMAR	7578 Frankford ROAD	242333443
10000000007	ROCKY	BHAI	7579 Frankford ROAD	242333445

More than 10 rows available. Increase rows selector to view more rows.

10 rows returned in 0.01 seconds Download

QUERY 3:

```
-- QUERY 3

SELECT * FROM CUSTOMERINFO;

--QUERY 4
```

Results | Explain | Describe | Saved SQL | History

CUSTOMER_ID	FIRST_NAME	LAST_NAME	ADDRESS
10000000002	RITESH	KULKARNI	7421 Frankford ROAD
10000000000	MIHIR	NEVPURKAR	7575 Frankford ROAD
10000000001	SAGAR	HAGAWANE	7421 FRANKFORD ROAD
10000000003	JOHN	WATSON	7575 Frankford ROAD
10000000009	RANBIR	KAPOOR	7581 Frankford ROAD
10000000008	SANJAY	DUTT	7580 Frankford ROAD
10000000010	SUNIL	SHETTY	7582 Frankford ROAD
10000000004	SONAL	SETH	7576 Frankford ROAD
10000000006	RAHUL	KUMAR	7578 Frankford ROAD
10000000007	ROCKY	BHAI	7579 Frankford ROAD

More than 10 rows available. Increase rows selector to view more rows.

10 rows returned in 0.03 seconds Download

QUERY 4:

Language SQL Rows 10 Clear Command Find Tables Save Run

```
--QUERY 4

SELECT * FROM CUSTOMER C
INNER JOIN ORDERS O ON C.CUSTOMER_ID = O.CUSTOMER_ID;

-- QUERY 5
```

Results Explain Describe Saved SQL History

CUSTOMER_ID	FIRST_NAME	LAST_NAME	ADDRESS	PHONE_NO	EMAIL	ORDER_ID	PRODUCT_ID	CUST
10000000003	JOHN	WATSON	7575 Frankford ROAD	242333444	john@gmail.com	20000000006	30000000004	1000
10000000001	SAGAR	HAGAWANE	7421 FRANKFORD ROAD	222555666	sagar@gmail.com	20000000003	30000000001	1000
10000000000	MIHIR	NEVPURKAR	7575 Frankford ROAD	222333444	mihir@gmail.com	20000000000	30000000000	1000
10000000000	MIHIR	NEVPURKAR	7575 Frankford ROAD	222333444	mihir@gmail.com	20000000001	30000000001	1000
10000000002	RITESH	KULKARNI	7421 Frankford ROAD	555333444	ritesh@gmail.com	20000000004	30000000002	1000
10000000004	SONAL	SETH	7576 Frankford ROAD	242333441	sonal@gmail.com	20000000009	30000000001	1000
10000000000	MIHIR	NEVPURKAR	7575 Frankford ROAD	222333444	mihir@gmail.com	20000000002	30000000002	1000
			7576					

cathy.j.dominguez@gmail.com cjd210002 Oracle APEX 22.1.0-17

QUERY 5:

```
SELECT * FROM ORDERS
ORDER BY TOTAL_PRICE DESC;

-- QUERY 6
```

Results | Explain | Describe | Saved SQL | History

ORDER_ID	PRODUCT_ID	CUSTOMER_ID	TOTAL_PRICE	ORDER_DATE	DELIVERY_ADDRESS	PAYMENT_METHOD
20000000006	30000000004	10000000003	180	01/31/2008	7575 Frankford Road	DEBIT CARD
20000000012	30000000001	10000000005	150	08/31/2008	7577 Frankford Road	DEBIT CARD
20000000011	30000000002	10000000005	150	08/31/2008	7577 Frankford Road	DEBIT CARD
20000000002	30000000002	10000000000	100	10/25/2007	7575 Frankford Road	CREDIT CARD
20000000008	30000000005	10000000004	90	10/31/2008	7576 Frankford Road	CASH
20000000001	30000000001	10000000000	90	10/25/2007	7575 Frankford Road	CREDIT CARD
20000000007	30000000004	10000000004	90	10/31/2008	7576 Frankford Road	CASH
20000000005	30000000003	10000000003	90	01/31/2008	7575 Frankford Road	DEBIT CARD
20000000009	30000000001	10000000004	75	10/31/2008	7576 Frankford Road	CASH
20000000010	30000000001	10000000005	60	08/31/2008	7577 Frankford Road	DEBIT CARD

More than 10 rows available. Increase rows selector to view more rows.

10 rows returned in 0.01 seconds Download

QUERY 6:

```sql
SELECT C.FIRST_NAME,
C.LAST_NAME,
P.PRODUCT_DESCRIPTION,
O.TOTAL_PRICE,
O.ORDER_DATE
FROM CUSTOMER C
LEFT JOIN ORDERS O ON C.CUSTOMER_ID = O.CUSTOMER_ID
LEFT JOIN PRODUCT P ON O.PRODUCT_ID = P.PRODUCT_ID;

-- QUERY 7
```

Results | Explain | Describe | Saved SQL | History

FIRST_NAME	LAST_NAME	PRODUCT_DESCRIPTION	TOTAL_PRICE	ORDER_DATE
JOHN	WATSON	VEGETABLES	90	01/31/2008
MIHIR	NEVPURKAR	SHIRT	40	10/25/2007
SONAL	SETH	LAPTOP	90	10/31/2008
JOHN	WATSON	AIRPODS	180	01/31/2008
SONAL	SETH	AIRPODS	90	10/31/2008
SAGAR	HAGAWANE	PANT	45	10/31/2007
MIHIR	NEVPURKAR	PANT	90	10/25/2007
SONAL	SETH	PANT	75	10/31/2008
JAY	SHETTY	PANT	150	08/31/2008
JAY	SHETTY	PANT	60	08/31/2008

More than 10 rows available. Increase rows selector to view more rows.

QUERY 7:

Language SQL Rows 10 Clear Command Find Tables Save Run

```sql
-- QUERY 7

SELECT DISTINCT C.FIRST_NAME,
C.LAST_NAME,
P.PRODUCT_DESCRIPTION,
O.TOTAL_PRICE,
O.ORDER_DATE
FROM CUSTOMER C
LEFT JOIN ORDERS O ON C.CUSTOMER_ID = O.CUSTOMER_ID
LEFT JOIN PRODUCT P ON O.PRODUCT_ID = P.PRODUCT_ID;

-- QUERY 8
```

Results Explain Describe Saved SQL History

FIRST_NAME	LAST_NAME	PRODUCT_DESCRIPTION	TOTAL_PRICE	ORDER_DATE
RITESH	KULKARNI	MOUSE	50	12/31/2007
SUNIL	SHETTY	-	-	-
MIHIR	NEVPURKAR	SHIRT	40	10/25/2007
JOHN	WATSON	AIRPODS	180	01/31/2008
SONAL	SETH	AIRPODS	90	10/31/2008
SAGAR	HAGAWANE	PANT	45	10/31/2007
MIHIR	NEVPURKAR	MOUSE	100	10/25/2007
JOHN	WATSON	VEGETABLES	90	01/31/2008
SANJAY	DUTT	-	-	-
SONAL	SETH	PANT	75	10/31/2008

More than 10 rows available. Increase rows selector to view more rows.

10 rows returned in 0.02 seconds Download

QUERY 8:

Language SQL Rows 10 Clear Command Find Tables Save Run

```sql
-- FILTER TOTAL SALE OF DIFFERENT PRODUCT TYPES GIVEN TOTAL SALES IS MORE THAN 50

SELECT P.PRODUCT_TYPE, SUM(O.TOTAL_PRICE) FROM ORDERS O
LEFT JOIN PRODUCT P ON O.PRODUCT_ID = P.PRODUCT_ID
GROUP BY P.PRODUCT_TYPE
HAVING SUM(O.TOTAL_PRICE) > 50;

-- QUERY 9

SELECT *
FROM CUSTOMER C
```

Results Explain Describe Saved SQL History

PRODUCT_TYPE	SUM(O.TOTAL_PRICE)
ELECTRONICS	660
CLOTHING	460
GROCERY	90

3 rows returned in 0.01 seconds Download

QUERY 9:

```sql
-- QUERY 9

SELECT *
FROM CUSTOMER C
LEFT JOIN ORDERS O ON C.CUSTOMER_ID = O.CUSTOMER_ID
LEFT JOIN PRODUCT P ON O.PRODUCT_ID = P.PRODUCT_ID
WHERE P.PRODUCT_TYPE IN ('CLOTHING', 'GROCERY');

-- QUERY 10
```

Results | Explain | Describe | Saved SQL | History

CUSTOMER_ID	FIRST_NAME	LAST_NAME	ADDRESS	PHONE_NO	EMAIL	ORDER_ID	PRODUCT_ID	CUST
10000000001	SAGAR	HAGAWANE	7421 FRANKFORD ROAD	222555666	sagar@gmail.com	20000000003	30000000001	1000
10000000000	MIHIR	NEVPURKAR	7575 Frankford ROAD	222333444	mihir@gmail.com	20000000000	30000000000	1000
10000000000	MIHIR	NEVPURKAR	7575 Frankford ROAD	222333444	mihir@gmail.com	20000000001	30000000001	1000
10000000004	SONAL	SETH	7576 Frankford ROAD	242333441	sonal@gmail.com	20000000009	30000000001	1000
10000000005	JAY	SHETTY	7577 Frankford ROAD	242333442	jay@gmail.com	20000000012	30000000001	1000
10000000003	JOHN	WATSON	7575 Frankford ROAD	242333444	john@gmail.com	20000000005	30000000003	1000

cathy.j.dominguez@gmail.com cjd210002 en

Oracle APEX 22.1.0-17

QUERY 10:

```
367   LEFT JOIN PRODUCT P ON O.PRODUCT_ID = P.PRODUCT_ID
368   WHERE P.PRODUCT_TYPE IN ('CLOTHING', 'GROCERY');
369
370   -- QUERY 10
371
372   SELECT EMAIL, LENGTH(EMAIL) AS EMAIL_LENGHT FROM CUSTOMER;
373
374   -- QUERY 11
375   SELECT * FROM PRODUCT;
376
377   DELETE FROM PRODUCT
```

Results | Explain | Describe | Saved SQL | History

EMAIL	EMAIL_LENGHT
ritesh@gmail.com	16
mihir@gmail.com	15
sagar@gmail.com	15
john@gmail.com	14
ranbir@gmail.com	16
sanjay@gmail.com	16
sunil@gmail.com	15
sonal@gmail.com	15
rahul@gmail.com	15
rocky@gmail.com	15

More than 10 rows available. Increase rows selector to view more rows.

10 rows returned in 0.01 seconds Download

QUERY 11:

Language SQL Rows 20 Clear Command Find Tables Save

```
-- QUERY 11
SELECT * FROM PRODUCT;

DELETE FROM PRODUCT
WHERE WAREHOUSE_ID = 40000000000;

SELECT * FROM PRODUCT;

ROLLBACK;
```

Results Explain Describe Saved SQL History

PRODUCT_ID	WAREHOUSE_ID	PRODUCT_TYPE	PRODUCT_DESCRIPTION	UNIT_PRICE
30000000012	40000000001	ELECTRONICS	MOUSE	50
30000000005	40000000003	ELECTRONICS	LAPTOP	90
30000000013	40000000002	GROCERY	VEGETABLES	5
30000000014	40000000003	ELECTRONICS	AIRPODS	90
30000000004	40000000003	ELECTRONICS	AIRPODS	90
30000000002	40000000001	ELECTRONICS	MOUSE	50
30000000007	40000000001	ELECTRONICS	KEYBOARD	30
30000000018	40000000002	CLOTHING	JEANS	25
30000000017	40000000001	ELECTRONICS	KEYBOARD	30
30000000015	40000000003	ELECTRONICS	LAPTOP	90
30000000016	40000000002	GROCERY	SOAP	10

11 rows returned in 0.01 seconds Download

QUERY 12:

Language SQL Rows 20 Clear Command Find Tables Save Run

```sql
-- QUERY 12

SELECT * FROM ORDERS;

UPDATE ORDERS
SET TOTAL_PRICE = 100
WHERE ORDER_ID = 20000000000;

SELECT * FROM ORDERS;

ROLLBACK;
```

Results Explain Describe Saved SQL History

ORDER_ID	PRODUCT_ID	CUSTOMER_ID	TOTAL_PRICE	ORDER_DATE	DELIVERY_ADDRESS	PAYMENT_METHOD
20000000006	30000000004	10000000003	180	01/31/2008	7575 Frankford Road	DEBIT CARD
20000000003	30000000001	10000000001	45	10/31/2007	7421 Frankford Road	CREDIT CARD
20000000000	30000000000	10000000000	100	10/25/2007	7575 Frankford Road	CREDIT CARD
20000000001	30000000001	10000000000	90	10/25/2007	7575 Frankford Road	CREDIT CARD
20000000004	30000000002	10000000002	50	12/31/2007	7421 Frankford Road	CASH
20000000009	30000000001	10000000004	75	10/31/2008	7576 Frankford Road	CASH
20000000002	30000000002	10000000000	100	10/25/2007	7575 Frankford Road	CREDIT CARD
20000000008	30000000005	10000000004	90	10/31/2008	7576 Frankford Road	CASH
20000000012	30000000001	10000000005	150	08/31/2008	7577 Frankford Road	DEBIT CARD
20000000007	30000000004	10000000004	90	10/31/2008	7576 Frankford Road	CASH
20000000005	30000000003	10000000003	90	01/31/2008	7575 Frankford Road	DEBIT CARD
20000000010	30000000001	10000000005	60	08/31/2008	7577 Frankford Road	DEBIT CARD

cathy.j.dominguez@gmail.com cjd210002 Copyright © 1999, 2022, Oracle and/or its affiliates. Oracle APEX 22.1.0-17

QUERY 13:

```
-- QUERY 13

--  FIND SALES FROM THE WAREHOUSE WHOSE WARE HOUSE IS IN PLANO REGION AND VENDORS ARE ALSO FROM PLANO

SELECT W.WAREHOUSE_ID, SUM(O.TOTAL_PRICE) AS SALE FROM ORDERS O
LEFT JOIN PRODUCT P ON O.PRODUCT_ID = P.PRODUCT_ID
LEFT JOIN WAREHOUSE W ON W.WAREHOUSE_ID = P.WAREHOUSE_ID
LEFT JOIN VENDOR V ON W.VENDOR_CODE = V.VENDOR_CODE
WHERE V.ADDRESS LIKE '%PLANO%' AND W.WAREHOUSE_ADDRESS LIKE '%PLANO%'
GROUP BY W.WAREHOUSE_ID;

-- QUERY 14
```

Results | Explain | Describe | Saved SQL | History

WAREHOUSE_ID	SALE
40000000003	360

1 rows returned in 0.06 seconds Download

QUERY 14:

SQL Commands Schema WKSP_CJD210002

Language SQL Rows 20 Clear Command Find Tables Save Run

```
-- QUERY 14

-- FIND THE TOTAL SALES DONE THROUGH EACH PAYMENT METHOD AND BEFORE 1ST OCT 2008

-- FOR APEX
SELECT PAYMENT_METHOD, SUM(TOTAL_PRICE) AS TOTAL_SALES FROM ORDERS
WHERE ORDER_DATE < 'OCT-01-2008'
GROUP BY PAYMENT_METHOD;

--FOR SQL
--SELECT PAYMENT_METHOD, SUM(TOTAL_PRICE) AS TOTAL_SALES FROM ORDERS
```

Results | Explain | Describe | Saved SQL | History

PAYMENT_METHOD	TOTAL_SALES
CREDIT CARD	335
CASH	50
DEBIT CARD	630

3 rows returned in 0.01 seconds Download

QUERY 15:

```sql
-- FIND THE TOTAL COUNT OF  PRODUCT_TYPE SOLD WHOSE WAREHOUSE IS IN DALLAS

SELECT P.PRODUCT_TYPE, COUNT(O.PRODUCT_ID) AS QUANTITIES
FROM ORDERS O
LEFT JOIN PRODUCT P ON O.PRODUCT_ID = P.PRODUCT_ID
LEFT JOIN WAREHOUSE W ON W.WAREHOUSE_ID = P.WAREHOUSE_ID
WHERE W.WAREHOUSE_REGION = 'DALLAS'
GROUP BY P.PRODUCT_TYPE;

-- QUERY 16
```

Results | Explain | Describe | Saved SQL | History

PRODUCT_TYPE	QUANTITIES
ELECTRONICS	3

1 rows returned in 0.05 seconds Download

QUERY 16:

Language SQL Rows 20 Clear Command Find Tables Save

```sql
-- QUERY 16

-- TOTAL SALES OF THE DIFFERENT VENDORS WHEN WAREHOUSE AND VENDOR ADDRESS ARE SAME AND ITS DALLAS

SELECT V.VENDOR_NAME, SUM(TOTAL_PRICE) FROM ORDERS O
LEFT JOIN PRODUCT P ON O.PRODUCT_ID = P.PRODUCT_ID
LEFT JOIN WAREHOUSE W ON W.WAREHOUSE_ID = P.WAREHOUSE_ID
LEFT JOIN VENDOR V ON W.VENDOR_CODE = V.VENDOR_CODE
WHERE W.WAREHOUSE_REGION = 'DALLAS' AND V.ADDRESS LIKE '%DALLAS%'
GROUP BY V.VENDOR_NAME;
```

Results | Explain | Describe | Saved SQL | History

VENDOR_NAME	SUM(TOTAL_PRICE)
RK SUPPLIERS	300

1 rows returned in 0.04 seconds Download

QUERY 17:

SQL Commands | Schema: WKSP_CJD210002 | Language: SQL | Rows: 20 | Clear Command | Find Tables | Save | Run

```sql
-- HOW MANY WAREHOUSE FROM EACH REGION MADE SALE AND HOW MANY HAVE NOT MADE ANY SALES

SELECT A.WAREHOUSE_REGION,
COUNT(CASE WHEN SALE = 1 THEN SALE END) AS SALES,
COUNT(CASE WHEN SALE = 0 THEN SALE END) AS NO_SALES
FROM
(
SELECT W.WAREHOUSE_REGION, W.WAREHOUSE_ID,
COUNT(DISTINCT CASE WHEN O.ORDER_ID IS NOT NULL THEN W.WAREHOUSE_ID END) AS SALE,
COUNT(DISTINCT CASE WHEN O.ORDER_ID IS NULL THEN W.WAREHOUSE_ID END) AS NO_SALE
FROM WAREHOUSE W
LEFT JOIN PRODUCT P ON P.WAREHOUSE_ID = W.WAREHOUSE_ID
LEFT JOIN ORDERS O ON O.PRODUCT_ID  = P.PRODUCT_ID
GROUP BY W.WAREHOUSE_ID, W.WAREHOUSE_REGION
) A
GROUP BY A.WAREHOUSE_REGION;
```

Results | Explain | Describe | Saved SQL | History

WAREHOUSE_REGION	SALES	NO_SALES
DALLAS	1	2
DFW	0	3
PLANO	1	3

3 rows returned in 0.03 seconds Download

QUERY 18:

```sql
GROUP BY A.WAREHOUSE_REGION;

-- QUERY 18

-- FIND OUT COUNT OF WAREHOUSE FROM EACH REGION WHICH HAVE SOLD PRODUCT WORTH LESS THAN 500

SELECT W.WAREHOUSE_REGION, SUM(O.TOTAL_PRICE) AS TOTAL_SALES
FROM ORDERS O
LEFT JOIN PRODUCT P
ON O.PRODUCT_ID = P.PRODUCT_ID
LEFT JOIN WAREHOUSE W ON
W.WAREHOUSE_ID = P.WAREHOUSE_ID
GROUP BY W.WAREHOUSE_REGION
HAVING SUM(O.TOTAL_PRICE) < 500 ;

-- QUERY 19

-- LIST ALL THE REGION WHERE ATLEAST 3 WAREHOUSES ARE THERE
```

Results | Explain | Describe | Saved SQL | History

WAREHOUSE_REGION	TOTAL_SALES
PLANO	360
DALLAS	300

2 rows returned in 0.02 seconds Download

QUERY 19:

```
-- QUERY 19

-- LIST ALL THE REGION WHERE ATLEAST 3 WAREHOUSES ARE THERE

SELECT WAREHOUSE_REGION, COUNT(WAREHOUSE_ID) FROM WAREHOUSE
GROUP BY WAREHOUSE_REGION
HAVING COUNT(WAREHOUSE_ID) >=3;

-- QUERY 20

-- LIST ALL THE CUSTOMERS WHO HAVE BOUGHT BOTH ELECTORNICS AND GROCERY PRODUCT TYPE

SELECT DISTINCT C.FIRST_NAME, C.LAST_NAME, P1.PRODUCT_TYPE as P1_Type ,P2.PRODUCT_TYPE as P2_Type
FROM CUSTOMER C
INNER JOIN ORDERS O1 ON O1.CUSTOMER_ID = C.CUSTOMER_ID
```

Results | Explain | Describe | Saved SQL | History

WAREHOUSE_REGION	COUNT(WAREHOUSE_ID)
PLANO	4
DALLAS	3
DFW	3

3 rows returned in 0.01 seconds Download

QUERY 20:

```
-- QUERY 20

-- LIST ALL THE CUSTOMERS WHO HAVE BOUGHT BOTH ELECTORNICS AND GROCERY PRODUCT TYPE

SELECT DISTINCT C.FIRST_NAME, C.LAST_NAME, P1.PRODUCT_ID as P1_ID ,P2.PRODUCT_ID as P2_ID, P1.PRODUCT_TYPE as P1_Type ,P2.PRODUCT_TYPE as P2_Type
FROM CUSTOMER C
INNER JOIN ORDERS O1 ON O1.CUSTOMER_ID = C.CUSTOMER_ID
INNER JOIN ORDERS O2 ON O1.CUSTOMER_ID = O2.CUSTOMER_ID
LEFT JOIN PRODUCT P1 ON O1.PRODUCT_ID = P1.PRODUCT_ID
LEFT JOIN PRODUCT P2 ON O2.PRODUCT_ID = P2.PRODUCT_ID
WHERE O1.PRODUCT_ID > O2.PRODUCT_ID
AND ((P1.PRODUCT_TYPE = 'GROCERY' AND P2.PRODUCT_TYPE = 'ELECTRONICS') OR (P2.PRODUCT_TYPE = 'GROCERY' AND P1.PRODUCT_TYPE = 'ELECTRONICS')) ;


/* ------------------------------------------------------ VALIDATION ------------------------------------------------------ */
select o.*,  (o.total_price/P.unit_price) AS quantity from orders o
```

Results | Explain | Describe | Saved SQL | History

FIRST_NAME	LAST_NAME	P1_ID	P2_ID	P1_TYPE	P2_TYPE
SONAL	SETH	30000000004	30000000003	ELECTRONICS	GROCERY

1 rows returned in 0.03 seconds Download

11 rows selected.

13 rows selected.

11 rows selected.

10 rows selected.

10 rows selected.

11 rows selected.

11 rows selected.

11 rows selected.

13 rows selected.

13 rows selected.

18 rows selected.

18 rows selected.

3 rows selected.

7 rows selected.

11 rows selected.

10 rows selected.

3 row(s) deleted.

7 rows selected.

Statement processed.

Statement processed.

Statement processed.

0 row(s) updated.

Statement processed.

Statement processed.

Statement processed.

Statement processed.

Statement processed.

Statement processed.

Statement processed.

Statement processed.

Statement processed.

Statement processed.

Statement processed.

Statement processed.

Statement processed.