Project - E-commerce Database Report
BUAN 6320

JSOM UTD

Group Members:
Mehak Dhawan
Najah Ahmed
Pragathi Beedu
Hruthik Rudravaram
Sagar Tailor

Introduction

The database design document describes the design and implementation of an e-commerce website. This database is written by the members of the group to demonstrate their understanding of SQL scripts, ERD diagrams and the business rules involved in a database. It includes the data definition language script and the data manipulation language script.

Design Decisions

We have created a database design for an e-commerce website. Five entities define the structure of the database: customer entity with records the data for all customers of the e-commerce business, the order entity maintains the records for current and historical orders, the payment entity records the transactions for each order and the type of transaction, the product entity records the various products in inventory and the vendor entity records the products supplied by that vendor.

Purpose

The purpose of this project is to develop and implement a new e-commerce database system that will improve the overall efficiency, accuracy, and speed of our e-commerce operations. This new system will help us better manage our product inventory, order processing, and customer data, which will in turn enhance the customer experience and increase revenue.

Objective

The objective of this project is to design, develop, test, and implement a new e-commerce database system that will meet the following key requirements:

- Ability to store and manage large amounts of product data and inventory information.
- Capability to process and track online orders in real-time,

Overall, the objective of this project is to improve the e-commerce capabilities of an organization and provide a seamless and enjoyable experience for its customers.

Project Scope

The scope of the project is to demonstrate the skills and knowledge acquired by the students of the group in creating and manipulating the data. They also demonstrate the skills learned in class by representing the data on the ERD.

In-Scope work

- Project requirements documentation
- Entity-relationship model
- DDL scripts
- DML scripts
- SQL scripts
- Comprehensive report

Database Goals, Expectations, and Deliverables

Database Goals

Our goal is to centralize and streamline e-commerce data storage and management, to improve inventory management and accuracy to reduce stockouts and increase revenue. An organized database system with enable real-time order processing and tracking for faster fulfillment and increased customer satisfaction. It will also help enhance customer data management to provide more personalized and targeted marketing and customer service.

Expectations

- The database system will be scalable and able to handle increased e-commerce volumes as our business grows.
- The database system will be user-friendly and easy for authorized personnel to manage and update product and order data.

- The database system can be fully integrated with a CRM system to provide a complete view of customer data and order history.
- The database system will comply with all relevant data privacy and security regulations and best practices.

Deliverables

A fully functional e-commerce database system that meets the defined objectives and requirements given in the project guidelines. Documentation for the new database system and an ERD to visualize the DBMS system.

Database Benefits

This database will benefit the e-commerce website owners to manage and use their data in an organized and effective manner. They will be able to record and maintain a history of their data and have an organized platform to store new data.

Requirements Definition Document

Business Rules

1. Each CUSTOMER has one or multiple ORDERS.
2. An ORDER belongs to one and only one CUSTOMER.
3. For every ORDER there is one and only one PAYMENT.
4. Each ORDER belongs to one and only one ORDER.
5. ORDER may have one or multiple PRODUCT.
6. PRODUCT can be included in one or multiple ORDER.
7. Each PRODUCT can be sold by at least one or multiple VENDORS.
8. Every VENDOR has one or multiple PRODUCT that they sell.

Entity and Attribute Description

Entities

Entity name - Customer

Entity description - The main resource who keeps the business running.

Main attributes of Customer:

Cust_id - Primary key - A unique identifier for each customer

Cust_First_Name - First name of the customer

Cust_Last_Name - Last name of the customer

Cust_PhNum - Contact information of the customer

Cust_zip - Zipcode for each individual customer

Entity name - Order

Entity description - Record of all orders placed.

Main attributes of Order:

Order_ID - (Primary Key) unique id for each order.

Cust_ID - (Foreign Key) to associate each order with the customer who placed it.

Product_ID - (Foreign Key) to associate each order with the product that has been ordered.

Order_Date - Date on which the order has been placed.

Order_Quantity – Quantity of each product.

Entity name - Payment

Entity description - Records all payment details for every order.

Main attributes of Payment:

Payment_ID - (Primary Key) unique id for each payment

Order_ID - (Foreign Key) Connects each payment to the unique order id.

Payment_type – Stores the type of payment method.

Payment_Date - Records the date of the payment.

Payment_Amt - Records the payment amount.

Entity name - Product

Entity description - This keeps a record of the multiple products available on the e-commerce website.

Main attributes of product

Product_ID - (Primary Key) Unique Id for each product type.

Product_Name - Name of the product.

Product_price - Price of each product type.

Product_Weight - Records the weight of each product.

Product_stock - Records the data for product inventory.

Vendor_ID - (Foreign Key) Related the products to the product vendor.

Entity name - Vendor

Entity description

Main attributes of vendor

Vendor_ID - (Primary Key) unique id for each vendor that provides products.

Vendor_phnum- Contact details of the vendor

Vendor_name- Name of each vendor

Vendor_zip- Records the zipcode for each vendor

Vendor_Product- Records the product each product supplies.

Relationship and cardinality description

Relationship - Runs between Customer and Order

Cardinality - 1:M between customer and order

Business rule:

- Each CUSTOMER has one or multiple ORDERS.
- An ORDER belongs to one and only one CUSTOMER.

Relationship- Runs between order and payment

Cardinality- 1:1 between order and payment

Business rule:

- For every ORDER there is one and only one PAYMENT.

- Each ORDER belongs to one and only one ORDER.

Relationship- Runs between Order and Product

Cardinality- M:1 between order and product

Business rule:

- ORDER may have one or multiple PRODUCT.
- PRODUCT can be included in one or multiple ORDER.

Relationship- Runs between Product and Vendor

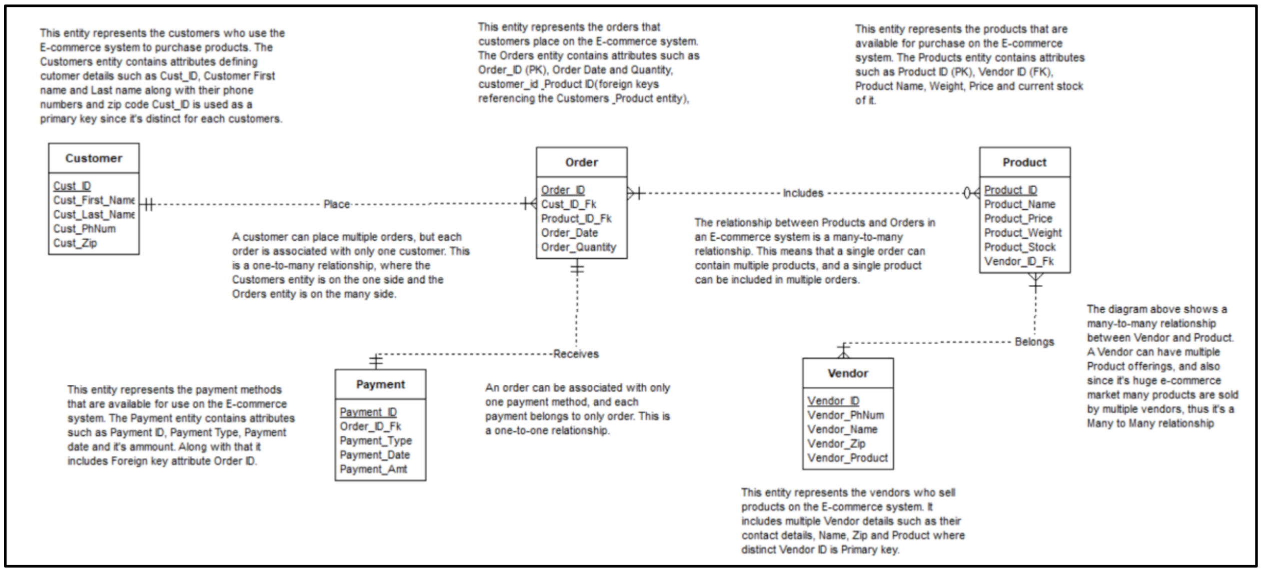
Cardinality- M:1 between product and vendor

Business rule:

- Each PRODUCT can be sold by at least one or multiple VENDORS.
- Every VENDOR has one or multiple PRODUCTS that they sell.

Detailed Database Design

Entity Relationship Diagram:



This entity represents the customers who use the E-commerce system to purchase products. The Customers entity contains attributes defining cutomer details such as Cust_ID, Customer First name and Last name along with their phone numbers and zip code Cust_ID is used as a primary key since it's distinct for each customers.

This entity represents the orders that customers place on the E-commerce system. The Orders entity contains attributes such as Order_ID (PK), Order Date and Quantity, customer_id ,Product ID(foreign keys referencing the Customers ,Product entity),

This entity represents the products that are available for purchase on the E-commerce system. The Products entity contains attributes such as Product ID (PK), Vendor ID (FK), Product Name, Weight, Price and current stock of it.

A customer can place multiple orders, but each order is associated with only one customer. This is a one-to-many relationship, where the Customers entity is on the one side and the Orders entity is on the many side.

The relationship between Products and Orders in an E-commerce system is a many-to-many relationship. This means that a single order can contain multiple products, and a single product can be included in multiple orders.

The diagram above shows a many-to-many relationship between Vendor and Product. A Vendor can have multiple Product offerings, and also since it's huge e-commerce market many products are sold by multiple vendors, thus it's a Many to Many relationship

This entity represents the payment methods that are available for use on the E-commerce system. The Payment entity contains attributes such as Payment ID, Payment Type, Payment date and it's ammount. Along with that it includes Foreign key attribute Order ID.

An order can be associated with only one payment method, and each payment belongs to only order. This is a one-to-one relationship.

This entity represents the vendors who sell products on the E-commerce system. It includes multiple Vendor details such as their contact details, Name, Zip and Product where distinct Vendor ID is Primary key.

Project DDL

```
--Drop statements to clean up objects from previous run

--Drop Triggers
DROP TRIGGER TR_CUST;
DROP TRIGGER TR_VENDOR;
DROP TRIGGER TR_PROD;
DROP TRIGGER TR_ORDER_ID;
DROP TRIGGER TR_ORDER_DATE;
DROP TRIGGER TR_PAYMENT_ID;
DROP TRIGGER TR_PAYMENT_DATE;

--Drop Sequences
DROP SEQUENCE Seq_Cust_ID;
DROP SEQUENCE Seq_Vendor_ID;
DROP SEQUENCE Seq_Prod_ID;
DROP SEQUENCE Seq_Order_ID;
DROP SEQUENCE Seq_Order_Date;
DROP SEQUENCE Seq_Payment_ID;
DROP SEQUENCE Seq_Payment_Date;

--Drop Views
DROP VIEW CUST_INFO;
DROP VIEW PURCHASE_ORDER_INFO;

--Drop Indices
DROP INDEX IDX_CUST_FIRST_NAME;
DROP INDEX IDX_VENDOR_NAME;
DROP INDEX IDX_PRODUCT_NAME;

--Drop Tables
DROP TABLE CUSTOMER;
DROP TABLE VENDOR;
DROP TABLE PRODUCT;
DROP TABLE PURCHASE_ORDER;
DROP TABLE PAYMENT;


--Creating tables based on entities
```

```
--Creating Table Customer
CREATE TABLE CUSTOMER
(
   CUST_ID INTEGER PRIMARY KEY,
   CUST_FIRST_NAME VARCHAR(20),
   CUST_LAST_NAME VARCHAR(20),
   CUST_PHNUM INTEGER,
   CUST_ZIP INTEGER
)

--Creating Table Vendor
CREATE TABLE VENDOR
(
   VENDOR_ID INTEGER PRIMARY KEY,
   VENDOR_PHNUM INTEGER,
   VENDOR_NAME VARCHAR(20),
   VENDOR_ZIP INTEGER,
   VENDOR_PRODUCT VARCHAR(50)
)

--Creating Table Product
CREATE TABLE PRODUCT
(
   PRODUCT_ID INTEGER PRIMARY KEY,
   PRODUCT_NAME VARCHAR(20),
   PRODUCT_PRICE INTEGER,
   PRODUCT_WEIGHT INTEGER,
   PRODUCT_STOCK INTEGER,
   VENDOR_ID INTEGER,
   CONSTRAINT FK_VENDOR_ID FOREIGN KEY(VENDOR_ID) REFERENCES
VENDOR (VENDOR_ID)
)

--Creating Table Purchase Order
CREATE TABLE PURCHASE_ORDER
(
   ORDER_ID INTEGER PRIMARY KEY,
   ORDER_DATE DATE,
   ORDER_QUANTITY INTEGER,
   CUST_ID INTEGER,
   PRODUCT_ID INTEGER,
   CONSTRAINT FK_CUST_ID FOREIGN KEY (CUST_ID) REFERENCES
CUSTOMER(CUST_ID),
   CONSTRAINT FK_PRODUCT_ID FOREIGN KEY(PRODUCT_ID) REFERENCES
PRODUCT(PRODUCT_ID)
)

--Creating Table Payment
CREATE TABLE PAYMENT
(
   PAYMENT_ID INTEGER PRIMARY KEY,
   ORDER_ID INTEGER,
   PAYMENT_TYPE VARCHAR(20),
   PAYMENT_DATE DATE,
   PAYMENT_AMT INTEGER,
   FOREIGN KEY (ORDER_ID) REFERENCES PURCHASE_ORDER(ORDER_ID)
)


--Creating indices for natural keys, foreign keys and frequently used columns
CREATE INDEX IDX_CUST_FIRST_NAME ON CUSTOMER (CUST_FIRST_NAME);
CREATE INDEX IDX_VENDOR_NAME ON VENDOR (VENDOR_NAME);
CREATE INDEX IDX_PRODUCT_NAME ON PRODUCT (PRODUCT_NAME);


--Creating Views
--Creating View to display all Customer Information
CREATE OR REPLACE VIEW CUST_INFO AS
SELECT * FROM CUSTOMER;

--Creating View to display all Purchase Order Information
CREATE OR REPLACE VIEW PURCHASE_ORDER_INFO AS
SELECT * FROM PURCHASE_ORDER;

--Creating Sequences
--Creating Sequence for Customer_ID
CREATE SEQUENCE Seq_Cust_ID
START WITH 1
INCREMENT BY 1
MINVALUE 1
```

```
NOMAXVALUE;

--Creating Sequence for Vendor_ID
CREATE SEQUENCE Seq_Vendor_ID
START WITH 10
INCREMENT BY 10
MINVALUE 10
NOMAXVALUE;

--Creating Sequence for Product_ID
CREATE SEQUENCE Seq_Prod_ID
START WITH 100
INCREMENT BY 100
MINVALUE 100
NOMAXVALUE;

--Creating Sequence for Order_ID
CREATE SEQUENCE Seq_Order_ID
START WITH 100
INCREMENT BY 100
MINVALUE 100
NOMAXVALUE;

--Creating Sequence for Order_Date
CREATE SEQUENCE Seq_Order_Date START WITH 1;

--Creating Sequence for Payment_ID
CREATE SEQUENCE Seq_Payment_ID
START WITH 10
INCREMENT BY 10
MINVALUE 10
NOMAXVALUE;

--Creating Sequence for Payment_Date
CREATE SEQUENCE Seq_Payment_Date START WITH 1;


--Creating Triggers
--Creating Trigger for CustomerID
CREATE OR REPLACE TRIGGER TR_CUST
```

```
BEFORE INSERT OR UPDATE ON CUSTOMER
FOR EACH ROW
BEGIN
:NEW.CUST_ID :=SEQ_Cust_ID.nextval;
END ;

--Creating trigger for Vendor_ID
CREATE OR REPLACE TRIGGER TR_VENDOR
BEFORE INSERT OR UPDATE ON VENDOR
FOR EACH ROW
BEGIN
:NEW.VENDOR_ID :=Seq_Vendor_ID.nextval;
END ;

--Creating trigger for Product_ID
CREATE OR REPLACE TRIGGER TR_PROD
BEFORE INSERT OR UPDATE ON PRODUCT
FOR EACH ROW
BEGIN
:NEW.PRODUCT_ID :=SEQ_Prod_ID.nextval;
END ;

--Creating trigger for Order_ID
CREATE OR REPLACE TRIGGER TR_ORDER_ID
BEFORE INSERT OR UPDATE ON PURCHASE_ORDER
FOR EACH ROW
BEGIN
:NEW.ORDER_ID :=Seq_Order_ID.nextval;
END;

--Creating trigger for Order_Date
CREATE OR REPLACE TRIGGER TR_ORDER_DATE
BEFORE INSERT OR UPDATE ON PURCHASE_ORDER
FOR EACH ROW
BEGIN
  :NEW.ORDER_ID := Seq_Order_Date.NEXTVAL;
  :NEW.ORDER_DATE := SYSDATE;
END;

--Creating trigger for Payment_ID
```

```
CREATE OR REPLACE TRIGGER TR_PAYMENT_ID
BEFORE INSERT OR UPDATE ON PAYMENT
FOR EACH ROW
BEGIN
:NEW.PAYMENT_ID :=Seq_Payment_ID.nextval;
END;

--Creating trigger for Payment_Date
CREATE OR REPLACE TRIGGER TR_PAYMENT_DATE
BEFORE INSERT OR UPDATE ON PAYMENT
FOR EACH ROW
BEGIN
  :NEW.PAYMENT_ID := Seq_Payment_Date.NEXTVAL;
  :NEW.PAYMENT_DATE := SYSDATE;
END;


COMMIT;
```

--Output of DDL:

```
Trigger TR_CUST dropped.

Trigger TR_VENDOR dropped.

Trigger TR_PROD dropped.

Trigger TR_ORDER_ID dropped.

Trigger TR_ORDER_DATE dropped.

Trigger TR_PAYMENT_ID dropped.

Trigger TR_PAYMENT_DATE dropped.

Sequence SEQ_CUST_ID dropped.

Sequence SEQ_VENDOR_ID dropped.
Sequence SEQ_PROD_ID dropped.

Sequence SEQ_ORDER_ID dropped.

Sequence SEQ_ORDER_DATE dropped.

Sequence SEQ_PAYMENT_ID dropped.

Sequence SEQ_PAYMENT_DATE dropped.

View CUST_INFO dropped.

View PURCHASE_ORDER_INFO dropped.

Index IDX_CUST_FIRST_NAME dropped.

Index IDX_VENDOR_NAME dropped.

Index IDX_PRODUCT_NAME dropped.

Table PAYMENT dropped.

Table PURCHASE_ORDER dropped.

Table PRODUCT dropped.

Table VENDOR dropped.

Table CUSTOMER dropped.

Table CUSTOMER created.

Table VENDOR created.

Table PRODUCT created.

Table PURCHASE_ORDER created.

Table PAYMENT created.
```

```
Index IDX_CUST_FIRST_NAME created.

Index IDX_VENDOR_NAME created.

Index IDX_PRODUCT_NAME created.

View CUST_INFO created.

View PURCHASE_ORDER_INFO created.

Sequence SEQ_CUST_ID created.

Sequence SEQ_VENDOR_ID created.

Sequence SEQ_PROD_ID created.

Sequence SEQ_ORDER_ID created.

Sequence SEQ_ORDER_DATE created.

Sequence SEQ_PAYMENT_ID created.

Sequence SEQ_PAYMENT_DATE created.

Trigger TR_CUST compiled.

Trigger TR_VENDOR compiled.

Trigger TR_PROD compiled.

Trigger TR_ORDER_ID compiled.

Trigger TR_ORDER_DATE compiled.

Trigger TR_PAYMENT_ID compiled.

Trigger TR_PAYMENT_DATE compiled.
```

–Project DML with Output:

```
--Populate all the tables

INSERT INTO CUSTOMER (CUST_FIRST_NAME, CUST_LAST_NAME, CUST_PHNUM,
CUST_ZIP) VALUES ('PRANEETHA','BEEDU','9452670572','75252');
INSERT INTO CUSTOMER (CUST_FIRST_NAME, CUST_LAST_NAME, CUST_PHNUM,
CUST_ZIP) VALUES ('JOHN', 'DOE', '2744059684', '90210');
INSERT INTO CUSTOMER (CUST_FIRST_NAME, CUST_LAST_NAME, CUST_PHNUM,
CUST_ZIP) VALUES ('JANE', 'SMITH', '9484657380', '60601');
INSERT INTO CUSTOMER (CUST_FIRST_NAME, CUST_LAST_NAME, CUST_PHNUM,
CUST_ZIP) VALUES ('DAVID', 'LEE', '9462538579', '10001');
INSERT INTO CUSTOMER (CUST_FIRST_NAME, CUST_LAST_NAME, CUST_PHNUM,
CUST_ZIP) VALUES ('SARAH', 'JOHNSON', '2534758690', '90210');
INSERT INTO CUSTOMER (CUST_FIRST_NAME, CUST_LAST_NAME, CUST_PHNUM,
CUST_ZIP) VALUES ('MICHAEL', 'BROWN', '4657876390', '60601');
INSERT INTO CUSTOMER (CUST_FIRST_NAME, CUST_LAST_NAME, CUST_PHNUM,
CUST_ZIP) VALUES ('AMY', 'DAVIS', '8175550141', '76102');
INSERT INTO CUSTOMER (CUST_FIRST_NAME, CUST_LAST_NAME, CUST_PHNUM,
CUST_ZIP) VALUES ('JASON', 'BROWN', '2145550132', '75204');
INSERT INTO CUSTOMER (CUST_FIRST_NAME, CUST_LAST_NAME, CUST_PHNUM,
CUST_ZIP) VALUES ('EMMA', 'WILLIAMS', '8325550165', '77019');
INSERT INTO CUSTOMER (CUST_FIRST_NAME, CUST_LAST_NAME, CUST_PHNUM,
CUST_ZIP) VALUES ('JULIA', 'RODRIGUEZ', '5125550167', '78704');
```

```
1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.
```

```
1 row inserted.


1 row inserted.

INSERT INTO VENDOR (VENDOR_PHNUM,
VENDOR_NAME,VENDOR_ZIP,VENDOR_PRODUCT) VALUES
('9876534561','AMAZON','67536','AMAZON TV');
INSERT INTO VENDOR (VENDOR_PHNUM,
VENDOR_NAME,VENDOR_ZIP,VENDOR_PRODUCT) VALUES
('9876534571','SHOPIFY','67533','RICHBOOK');
INSERT INTO VENDOR (VENDOR_PHNUM,
VENDOR_NAME,VENDOR_ZIP,VENDOR_PRODUCT) VALUES
('9876534581','WALMART','67530','KELLOGGS CORN CEREAL');
INSERT INTO VENDOR (VENDOR_PHNUM,
VENDOR_NAME,VENDOR_ZIP,VENDOR_PRODUCT) VALUES
('9876534591','TEMU','67532','SPEAKERS');
INSERT INTO VENDOR (VENDOR_PHNUM,
VENDOR_NAME,VENDOR_ZIP,VENDOR_PRODUCT) VALUES
('9876534551','SHEIN','67534','NIKE SNEAKERS');
INSERT INTO VENDOR (VENDOR_PHNUM,
VENDOR_NAME,VENDOR_ZIP,VENDOR_PRODUCT) VALUES
('9876534561','BESTBUY','67531','UNDER AROMOUR TSHIRT');
INSERT INTO VENDOR (VENDOR_PHNUM,
VENDOR_NAME,VENDOR_ZIP,VENDOR_PRODUCT) VALUES
('9876534571','EBAY','67535','CLEANING MOP');
INSERT INTO VENDOR (VENDOR_PHNUM,
VENDOR_NAME,VENDOR_ZIP,VENDOR_PRODUCT) VALUES
('9876534581','TARGET','67537','GRANOLA BAR');
INSERT INTO VENDOR (VENDOR_PHNUM,
VENDOR_NAME,VENDOR_ZIP,VENDOR_PRODUCT) VALUES
('9876534591','OVERSTOCK','67539','AIR FRYER');
INSERT INTO VENDOR (VENDOR_PHNUM,
VENDOR_NAME,VENDOR_ZIP,VENDOR_PRODUCT) VALUES
('9876534501','NEWEGG','67538','NINTENDO SWITCH');

1 row inserted.

1 row inserted.
```

```
1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.
```

```
INSERT INTO PRODUCT (PRODUCT_NAME, PRODUCT_PRICE, PRODUCT_WEIGHT,
PRODUCT_STOCK, VENDOR_ID) VALUES ('AMAZON TV','250.99','32','48', '30');
INSERT INTO PRODUCT (PRODUCT_NAME, PRODUCT_PRICE, PRODUCT_WEIGHT,
PRODUCT_STOCK, VENDOR_ID) VALUES ('NINTENDO SWITCH','294.49','2','34','20');
INSERT INTO PRODUCT (PRODUCT_NAME, PRODUCT_PRICE, PRODUCT_WEIGHT,
PRODUCT_STOCK, VENDOR_ID) VALUES ('SPEAKERS','90.99','3','44','60');
INSERT INTO PRODUCT (PRODUCT_NAME, PRODUCT_PRICE, PRODUCT_WEIGHT,
PRODUCT_STOCK, VENDOR_ID) VALUES ('RICHBOOK','7.99','0.3','89', '100');
INSERT INTO PRODUCT (PRODUCT_NAME, PRODUCT_PRICE, PRODUCT_WEIGHT,
PRODUCT_STOCK, VENDOR_ID) VALUES ('NIKE SNEAKERS','159.99','0.9','47', '70');
INSERT INTO PRODUCT (PRODUCT_NAME, PRODUCT_PRICE, PRODUCT_WEIGHT,
PRODUCT_STOCK, VENDOR_ID) VALUES ('UNDER AROMOUR
TSHIRT','25.00','0.2','98', '80');
INSERT INTO PRODUCT (PRODUCT_NAME, PRODUCT_PRICE, PRODUCT_WEIGHT,
PRODUCT_STOCK, VENDOR_ID) VALUES ('GRANOLA BAR','25.90','3.4','67', '40');
INSERT INTO PRODUCT (PRODUCT_NAME, PRODUCT_PRICE, PRODUCT_WEIGHT,
PRODUCT_STOCK, VENDOR_ID) VALUES ('KELLOGGS CORN
CEREAL','7.49','2.5','117', '50');
INSERT INTO PRODUCT (PRODUCT_NAME, PRODUCT_PRICE, PRODUCT_WEIGHT,
PRODUCT_STOCK, VENDOR_ID) VALUES ('CLEANING MOP','15.40','1.5','34', '90');
INSERT INTO PRODUCT (PRODUCT_NAME, PRODUCT_PRICE, PRODUCT_WEIGHT,
PRODUCT_STOCK, VENDOR_ID) VALUES ('AIR FRYER','39.90','6','32', '10');
```

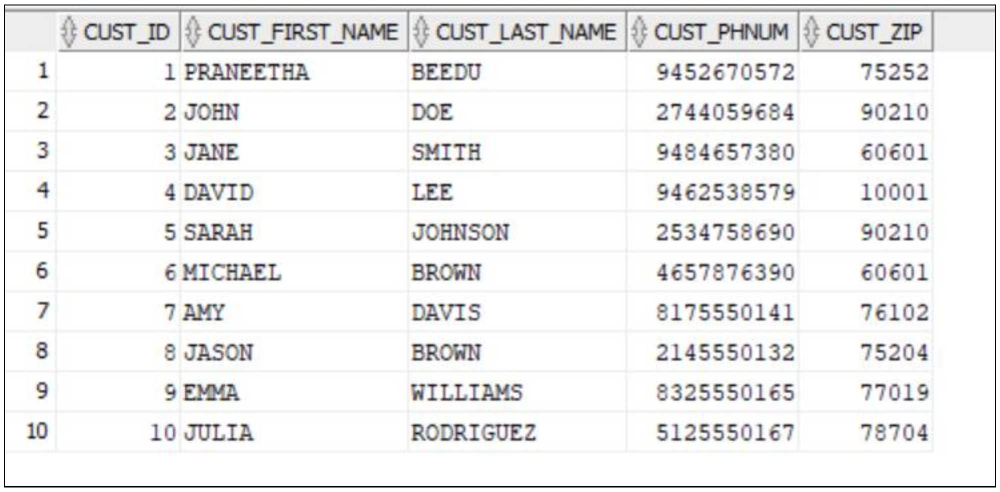
```
1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

INSERT INTO PURCHASE_ORDER (ORDER_QUANTITY, CUST_ID, PRODUCT_ID)
VALUES ('2','1','900');
INSERT INTO PURCHASE_ORDER (ORDER_QUANTITY, CUST_ID, PRODUCT_ID)
VALUES ('5','2','100');
INSERT INTO PURCHASE_ORDER (ORDER_QUANTITY, CUST_ID, PRODUCT_ID)
VALUES ('10','3','200');
INSERT INTO PURCHASE_ORDER (ORDER_QUANTITY, CUST_ID, PRODUCT_ID)
VALUES ('1','4','200');
INSERT INTO PURCHASE_ORDER (ORDER_QUANTITY, CUST_ID, PRODUCT_ID)
VALUES ('3','5','300');
INSERT INTO PURCHASE_ORDER (ORDER_QUANTITY, CUST_ID, PRODUCT_ID)
VALUES ('11','6','400');
INSERT INTO PURCHASE_ORDER (ORDER_QUANTITY, CUST_ID, PRODUCT_ID)
VALUES ('15','7','500');
INSERT INTO PURCHASE_ORDER (ORDER_QUANTITY, CUST_ID, PRODUCT_ID)
VALUES ('2','1','600');
INSERT INTO PURCHASE_ORDER (ORDER_QUANTITY, CUST_ID, PRODUCT_ID)
VALUES ('10','8','800');
INSERT INTO PURCHASE_ORDER (ORDER_QUANTITY, CUST_ID, PRODUCT_ID)
VALUES ('2','10','700');
```

```
INSERT INTO PURCHASE_ORDER (ORDER_QUANTITY, CUST_ID, PRODUCT_ID)
VALUES ('10','9','900' );
INSERT INTO PURCHASE_ORDER (ORDER_QUANTITY, CUST_ID, PRODUCT_ID)
VALUES ('10','9','900' );

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.


INSERT INTO PAYMENT (ORDER_ID,PAYMENT_TYPE,PAYMENT_AMT) VALUES
('100','CASH','1000');
INSERT INTO PAYMENT (ORDER_ID,PAYMENT_TYPE,PAYMENT_AMT) VALUES
('200','CASH','2500');
INSERT INTO PAYMENT (ORDER_ID,PAYMENT_TYPE,PAYMENT_AMT) VALUES
('300','CARD','3000');
INSERT INTO PAYMENT (ORDER_ID,PAYMENT_TYPE,PAYMENT_AMT) VALUES
('400','CARD','4000');
INSERT INTO PAYMENT (ORDER_ID,PAYMENT_TYPE,PAYMENT_AMT) VALUES
('500','CASH','5500');
```

```
INSERT INTO PAYMENT (ORDER_ID,PAYMENT_TYPE,PAYMENT_AMT) VALUES
('600','CARD','6700');
INSERT INTO PAYMENT (ORDER_ID,PAYMENT_TYPE,PAYMENT_AMT) VALUES
('700','CARD','8000');
INSERT INTO PAYMENT (ORDER_ID,PAYMENT_TYPE,PAYMENT_AMT) VALUES
('800','CASH','10000');
INSERT INTO PAYMENT (ORDER_ID,PAYMENT_TYPE,PAYMENT_AMT) VALUES
('900','CASH','8000');
INSERT INTO PAYMENT (ORDER_ID,PAYMENT_TYPE,PAYMENT_AMT) VALUES
('1200','CASH','200');

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.
```

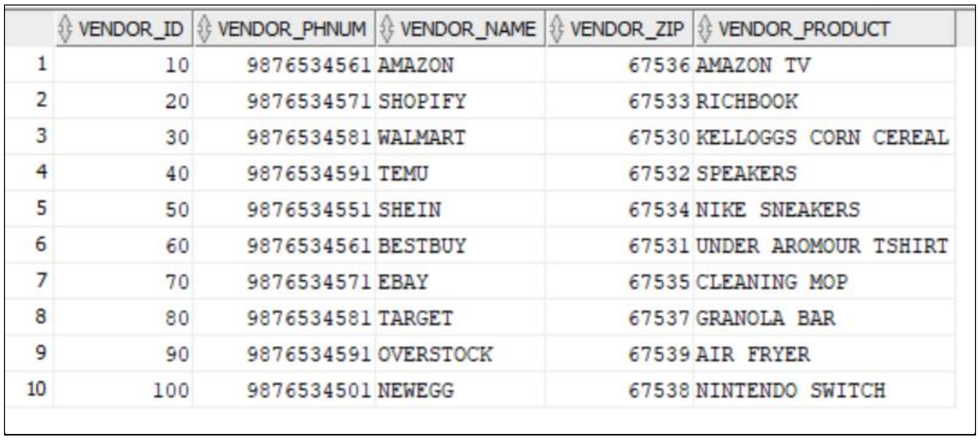
-- Basic Queries

--1) Select all columns and all rows from one table.
--Select all columns and rows from Customer table
SELECT * FROM CUSTOMER;



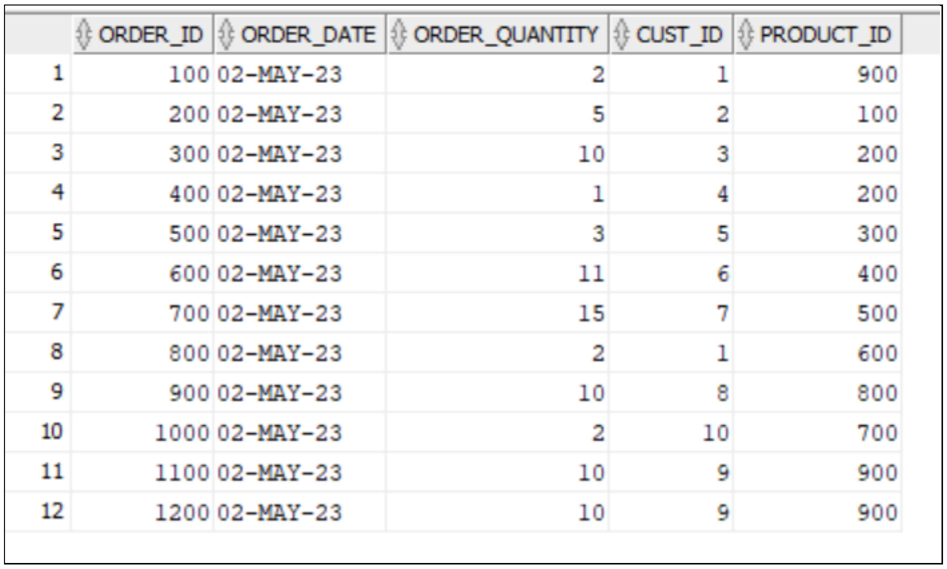
	CUST_ID	CUST_FIRST_NAME	CUST_LAST_NAME	CUST_PHNUM	CUST_ZIP
1	1	PRANEETHA	BEEDU	9452670572	75252
2	2	JOHN	DOE	2744059684	90210
3	3	JANE	SMITH	9484657380	60601
4	4	DAVID	LEE	9462538579	10001
5	5	SARAH	JOHNSON	2534758690	90210
6	6	MICHAEL	BROWN	4657876390	60601
7	7	AMY	DAVIS	8175550141	76102
8	8	JASON	BROWN	2145550132	75204
9	9	EMMA	WILLIAMS	8325550165	77019
10	10	JULIA	RODRIGUEZ	5125550167	78704

--2) Select five columns and all rows from one table
--Select five columns and all rows from Vendor table
SELECT VENDOR_ID, VENDOR_PHNUM, VENDOR_NAME, VENDOR_ZIP, VENDOR_PRODUCT
FROM VENDOR;



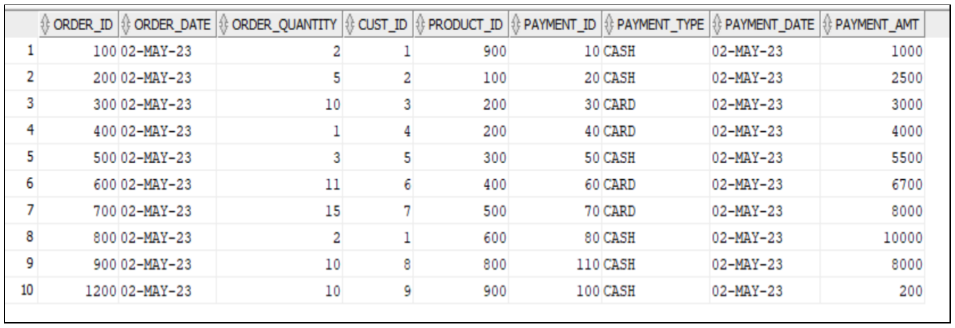
	VENDOR_ID	VENDOR_PHNUM	VENDOR_NAME	VENDOR_ZIP	VENDOR_PRODUCT
1	10	9876534561	AMAZON	67536	AMAZON TV
2	20	9876534571	SHOPIFY	67533	RICHBOOK
3	30	9876534581	WALMART	67530	KELLOGGS CORN CEREAL
4	40	9876534591	TEMU	67532	SPEAKERS
5	50	9876534551	SHEIN	67534	NIKE SNEAKERS
6	60	9876534561	BESTBUY	67531	UNDER AROMOUR TSHIRT
7	70	9876534571	EBAY	67535	CLEANING MOP
8	80	9876534581	TARGET	67537	GRANOLA BAR
9	90	9876534591	OVERSTOCK	67539	AIR FRYER
10	100	9876534501	NEWEGG	67538	NINTENDO SWITCH

--3) Select all columns from all rows from one view
--Select all columns from view - Purchase_Order_Info
SELECT * FROM PURCHASE_ORDER_INFO;



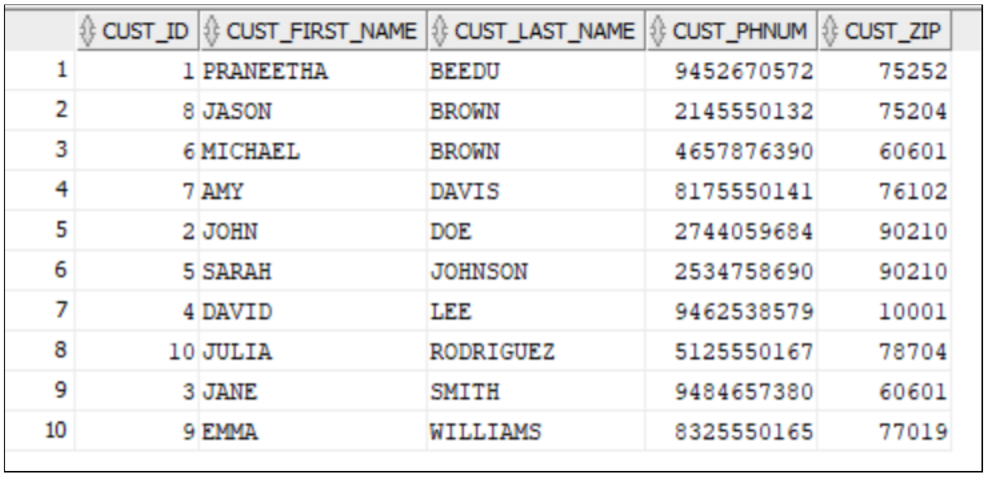
	ORDER_ID	ORDER_DATE	ORDER_QUANTITY	CUST_ID	PRODUCT_ID
1	100	02-MAY-23	2	1	900
2	200	02-MAY-23	5	2	100
3	300	02-MAY-23	10	3	200
4	400	02-MAY-23	1	4	200
5	500	02-MAY-23	3	5	300
6	600	02-MAY-23	11	6	400
7	700	02-MAY-23	15	7	500
8	800	02-MAY-23	2	1	600
9	900	02-MAY-23	10	8	800
10	1000	02-MAY-23	2	10	700
11	1100	02-MAY-23	10	9	900
12	1200	02-MAY-23	10	9	900

--4) Using a join on 2 tables, select all columns and all rows from the tables without the use of a Cartesian product
--Select all columns joining tables Payment and Order
SELECT O.*,
P.PAYMENT_ID,P.PAYMENT_TYPE,P.PAYMENT_DATE,P.PAYMENT_AMT FROM PURCHASE_ORDER O
JOIN PAYMENT P
ON O.Order_ID = P.Order_ID;



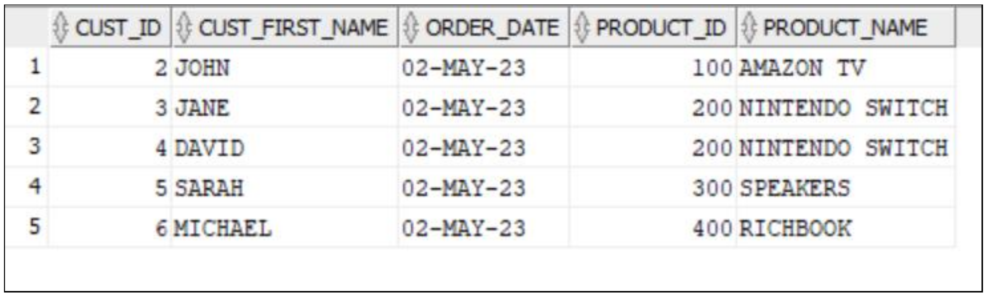
	ORDER_ID	ORDER_DATE	ORDER_QUANTITY	CUST_ID	PRODUCT_ID	PAYMENT_ID	PAYMENT_TYPE	PAYMENT_DATE	PAYMENT_AMT
1	100	02-MAY-23	2	1	900	10	CASH	02-MAY-23	1000
2	200	02-MAY-23	5	2	100	20	CASH	02-MAY-23	2500
3	300	02-MAY-23	10	3	200	30	CARD	02-MAY-23	3000
4	400	02-MAY-23	1	4	200	40	CARD	02-MAY-23	4000
5	500	02-MAY-23	3	5	300	50	CASH	02-MAY-23	5500
6	600	02-MAY-23	11	6	400	60	CARD	02-MAY-23	6700
7	700	02-MAY-23	15	7	500	70	CARD	02-MAY-23	8000
8	800	02-MAY-23	2	1	600	80	CASH	02-MAY-23	10000
9	900	02-MAY-23	10	8	800	110	CASH	02-MAY-23	8000
10	1200	02-MAY-23	10	9	900	100	CASH	02-MAY-23	200

--5) Select and order data retrieved from one table
--Select data from Customer and order by Customer last name in ascending order
SELECT * FROM CUSTOMER
ORDER BY CUST_LAST_NAME ASC;



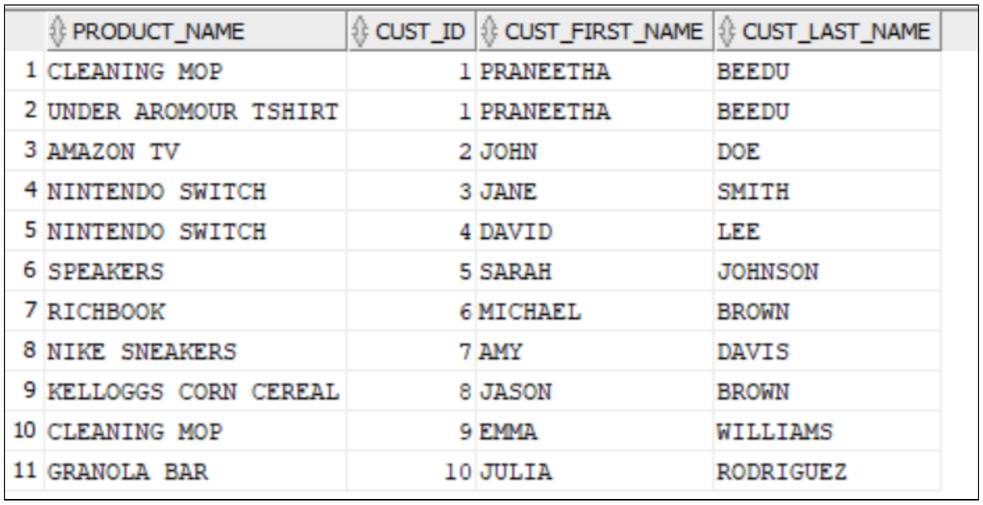
	CUST_ID	CUST_FIRST_NAME	CUST_LAST_NAME	CUST_PHNUM	CUST_ZIP
1	1	PRANEETHA	BEEDU	9452670572	75252
2	8	JASON	BROWN	2145550132	75204
3	6	MICHAEL	BROWN	4657876390	60601
4	7	AMY	DAVIS	8175550141	76102
5	2	JOHN	DOE	2744059684	90210
6	5	SARAH	JOHNSON	2534758690	90210
7	4	DAVID	LEE	9462538579	10001
8	10	JULIA	RODRIGUEZ	5125550167	78704
9	3	JANE	SMITH	9484657380	60601
10	9	EMMA	WILLIAMS	8325550165	77019

--6) Using a join on 3 tables, select 5 columns from the 3 tables. Use syntax that would limit the output to 5 rows
--Select 5 columns and limit output to 5 rows from tables - Customer, Purchase_Order and Product
SELECT C.CUST_ID, C.CUST_FIRST_NAME, O.ORDER_DATE, O.PRODUCT_ID, P.PRODUCT_NAME FROM CUSTOMER C
JOIN PURCHASE_ORDER O ON O.CUST_ID = C.CUST_ID
JOIN PRODUCT P ON P.PRODUCT_ID = O.PRODUCT_ID
FETCH FIRST 5 ROWS ONLY;



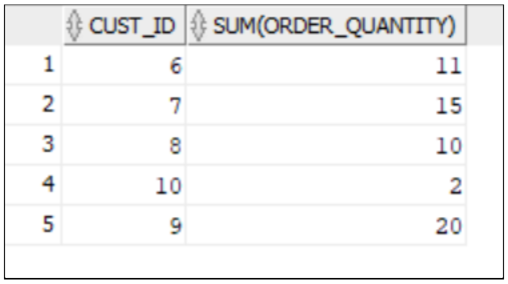
	CUST_ID	CUST_FIRST_NAME	ORDER_DATE	PRODUCT_ID	PRODUCT_NAME
1	2	JOHN	02-MAY-23	100	AMAZON TV
2	3	JANE	02-MAY-23	200	NINTENDO SWITCH
3	4	DAVID	02-MAY-23	200	NINTENDO SWITCH
4	5	SARAH	02-MAY-23	300	SPEAKERS
5	6	MICHAEL	02-MAY-23	400	RICHBOOK

```
--7) Select distinct rows using joins on 3 tables. Display the products ordered by Customers
--Select distinct rows from tables - Product, Purchase_Order and Customer
SELECT DISTINCT P.PRODUCT_NAME, C.CUST_ID, C.CUST_FIRST_NAME,
C.CUST_LAST_NAME FROM PRODUCT P
JOIN PURCHASE_ORDER O ON O.PRODUCT_ID = P.PRODUCT_ID
JOIN CUSTOMER C ON C.CUST_ID = O.CUST_ID;
```



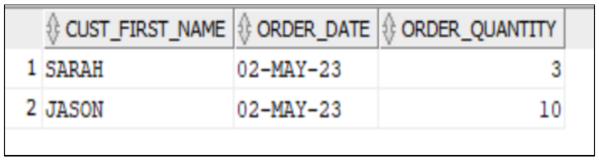
	PRODUCT_NAME	CUST_ID	CUST_FIRST_NAME	CUST_LAST_NAME
1	CLEANING MOP	1	PRANEETHA	BEEDU
2	UNDER AROMOUR TSHIRT	1	PRANEETHA	BEEDU
3	AMAZON TV	2	JOHN	DOE
4	NINTENDO SWITCH	3	JANE	SMITH
5	NINTENDO SWITCH	4	DAVID	LEE
6	SPEAKERS	5	SARAH	JOHNSON
7	RICHBOOK	6	MICHAEL	BROWN
8	NIKE SNEAKERS	7	AMY	DAVIS
9	KELLOGGS CORN CEREAL	8	JASON	BROWN
10	CLEANING MOP	9	EMMA	WILLIAMS
11	GRANOLA BAR	10	JULIA	RODRIGUEZ

```
--8) Use GROUP BY and HAVING in a select statement using one or more tables
--Select Customer ID and the sum of Order quantity ordered by Customers where the Cust_ID>5
SELECT CUST_ID, SUM (ORDER_QUANTITY) FROM PURCHASE_ORDER GROUP BY
CUST_ID HAVING CUST_ID>5;
```



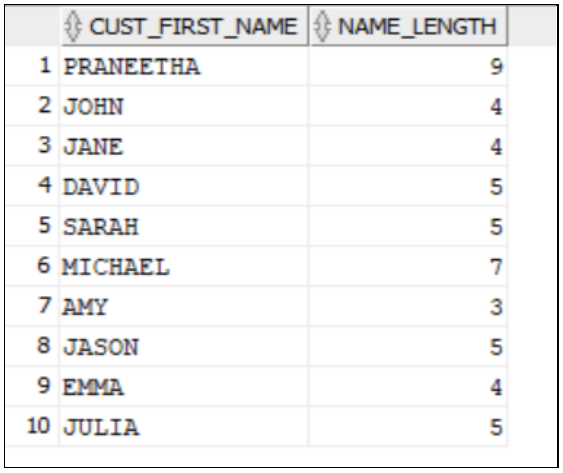
	CUST_ID	SUM(ORDER_QUANTITY)
1	6	11
2	7	15
3	8	10
4	10	2
5	9	20

--9) Use IN clause to select data from one or more tables
--Select Customer first name, order date and order quantity placed by customer whose first name is either Jason or Sarah
SELECT C.CUST_FIRST_NAME, O.ORDER_DATE, O.ORDER_QUANTITY
FROM CUSTOMER C, PURCHASE_ORDER O
WHERE C.CUST_ID = O.CUST_ID
AND C.CUST_FIRST_NAME IN ('JASON','SARAH');



	CUST_FIRST_NAME	ORDER_DATE	ORDER_QUANTITY
1	SARAH	02-MAY-23	3
2	JASON	02-MAY-23	10

--10) Select length of one column from one table
--Display the length of Customer First name
SELECT CUST_FIRST_NAME, LENGTH(CUST_FIRST_NAME) AS NAME_LENGTH
FROM CUSTOMER;



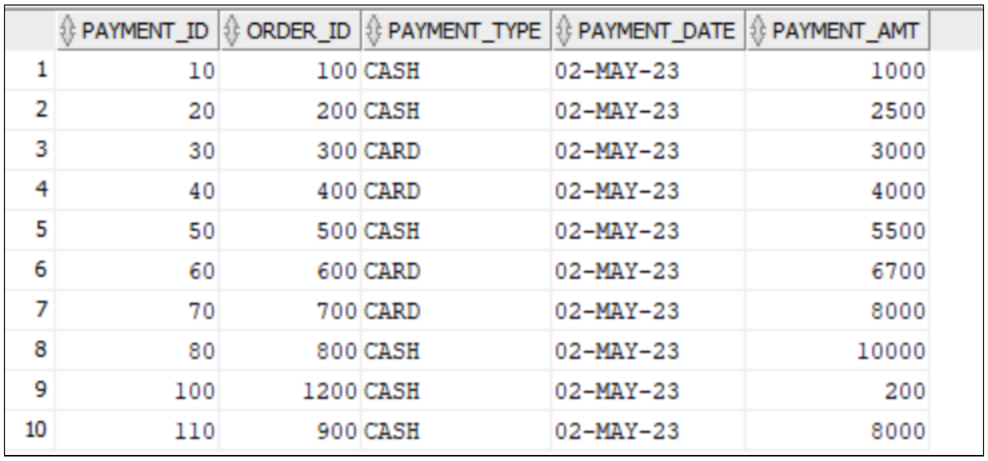
	CUST_FIRST_NAME	NAME_LENGTH
1	PRANEETHA	9
2	JOHN	4
3	JANE	4
4	DAVID	5
5	SARAH	5
6	MICHAEL	7
7	AMY	3
8	JASON	5
9	EMMA	4
10	JULIA	5

--11) Delete one record from one table.
--Use select statements to demonstrate the table contents before and after the DELETE statement
--Delete row from Payment where order id = 100
SELECT * FROM PAYMENT;

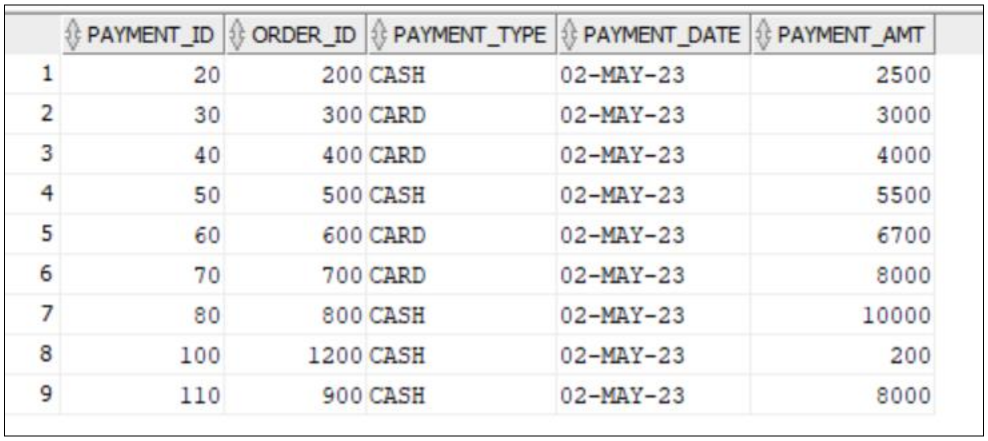
DELETE FROM PAYMENT WHERE ORDER_ID = ‘100’;

SELECT * FROM PAYMENT;

ROLLBACK;
SELECT * FROM PAYMENT;



	PAYMENT_ID	ORDER_ID	PAYMENT_TYPE	PAYMENT_DATE	PAYMENT_AMT
1	10	100	CASH	02-MAY-23	1000
2	20	200	CASH	02-MAY-23	2500
3	30	300	CARD	02-MAY-23	3000
4	40	400	CARD	02-MAY-23	4000
5	50	500	CASH	02-MAY-23	5500
6	60	600	CARD	02-MAY-23	6700
7	70	700	CARD	02-MAY-23	8000
8	80	800	CASH	02-MAY-23	10000
9	100	1200	CASH	02-MAY-23	200
10	110	900	CASH	02-MAY-23	8000



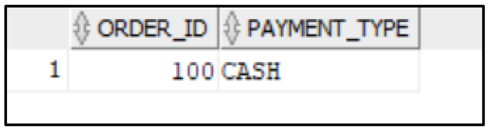
	PAYMENT_ID	ORDER_ID	PAYMENT_TYPE	PAYMENT_DATE	PAYMENT_AMT
1	20	200	CASH	02-MAY-23	2500
2	30	300	CARD	02-MAY-23	3000
3	40	400	CARD	02-MAY-23	4000
4	50	500	CASH	02-MAY-23	5500
5	60	600	CARD	02-MAY-23	6700
6	70	700	CARD	02-MAY-23	8000
7	80	800	CASH	02-MAY-23	10000
8	100	1200	CASH	02-MAY-23	200
9	110	900	CASH	02-MAY-23	8000

--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
--Update Payment type from Cash to Card where order id = 100
SELECT ORDER_ID, PAYMENT_TYPE FROM PAYMENT WHERE ORDER_ID = '100';

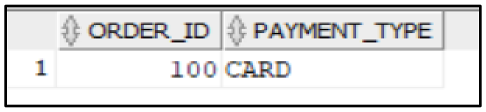
UPDATE PAYMENT
SET PAYMENT_TYPE = 'CARD' WHERE ORDER_ID = '100';

SELECT ORDER_ID, PAYMENT_TYPE FROM PAYMENT WHERE ORDER_ID = '100';

ROLLBACK;
SELECT * FROM PAYMENT;



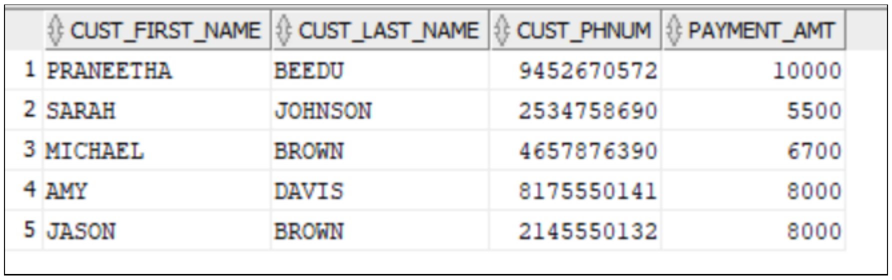
	ORDER_ID	PAYMENT_TYPE
1	100	CASH



	ORDER_ID	PAYMENT_TYPE
1	100	CARD

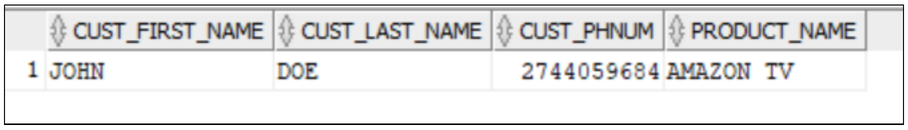
--Advanced Queries

--13) List Customer Information where the Payment Amount for product is greater than 5000
SELECT C.CUST_FIRST_NAME, C.CUST_LAST_NAME, C.CUST_PHNUM, P.PAYMENT_AMT
From CUSTOMER C
JOIN PURCHASE_ORDER O ON C.CUST_ID = O.CUST_ID
JOIN PAYMENT P ON P.ORDER_ID = O.ORDER_ID
WHERE PAYMENT_AMT > 5000;



	CUST_FIRST_NAME	CUST_LAST_NAME	CUST_PHNUM	PAYMENT_AMT
1	PRANEETHA	BEEDU	9452670572	10000
2	SARAH	JOHNSON	2534758690	5500
3	MICHAEL	BROWN	4657876390	6700
4	AMY	DAVIS	8175550141	8000
5	JASON	BROWN	2145550132	8000

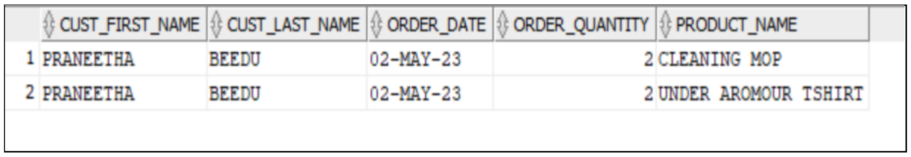
--14) List Customer First Name, Last Name, Phone number who ordered product Amazon TV
SELECT DISTINCT C.CUST_FIRST_NAME, C.CUST_LAST_NAME,C.CUST_PHNUM,P.PRODUCT_NAME FROM CUSTOMER C
JOIN PURCHASE_ORDER O ON O.CUST_ID = C.CUST_ID
JOIN PRODUCT P ON P.PRODUCT_ID = O.PRODUCT_ID
WHERE PRODUCT_NAME = 'AMAZON TV';



	CUST_FIRST_NAME	CUST_LAST_NAME	CUST_PHNUM	PRODUCT_NAME
1	JOHN	DOE	2744059684	AMAZON TV

--15) Display Customer name, order date, order quantity and product ordered where the customer zip = 75252

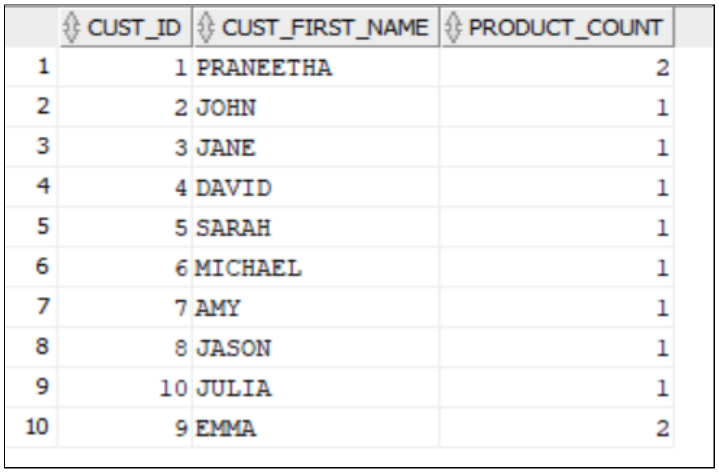
```
SELECT C.CUST_FIRST_NAME, C.CUST_LAST_NAME, O.ORDER_DATE,
O.ORDER_QUANTITY, P.PRODUCT_NAME FROM CUSTOMER C
INNER JOIN PURCHASE_ORDER O ON C.CUST_ID = O.CUST_ID
INNER JOIN PRODUCT P ON O.PRODUCT_ID = P.PRODUCT_ID
WHERE C.CUST_ZIP = '75252';
```



	CUST_FIRST_NAME	CUST_LAST_NAME	ORDER_DATE	ORDER_QUANTITY	PRODUCT_NAME
1	PRANEETHA	BEEDU	02-MAY-23	2	CLEANING MOP
2	PRANEETHA	BEEDU	02-MAY-23	2	UNDER AROMOUR TSHIRT

--16) Display Customer ID, Customer Name and their product counts

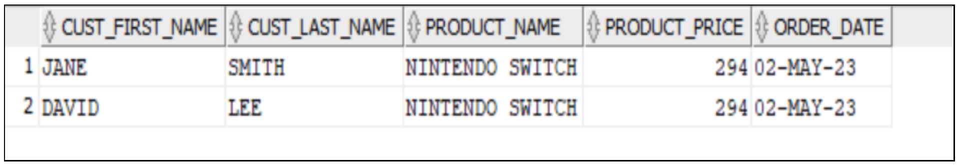
```
SELECT C.CUST_ID, C.CUST_FIRST_NAME, COUNT(P.PRODUCT_ID) AS
PRODUCT_COUNT FROM CUSTOMER C
LEFT JOIN PURCHASE_ORDER O ON C.CUST_ID = O.CUST_ID
LEFT JOIN PRODUCT P ON O.PRODUCT_ID = P.PRODUCT_ID
GROUP BY C.CUST_ID, C.CUST_FIRST_NAME;
```



	CUST_ID	CUST_FIRST_NAME	PRODUCT_COUNT
1	1	PRANEETHA	2
2	2	JOHN	1
3	3	JANE	1
4	4	DAVID	1
5	5	SARAH	1
6	6	MICHAEL	1
7	7	AMY	1
8	8	JASON	1
9	10	JULIA	1
10	9	EMMA	2

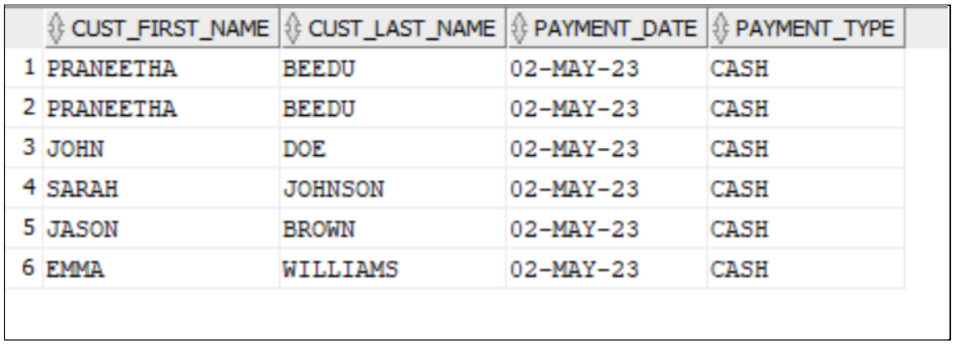
--17) Display the names of Customers and products who ordered product with maximum product price

```
SELECT C.CUST_FIRST_NAME, C.CUST_LAST_NAME, P.PRODUCT_NAME,
P.PRODUCT_PRICE, O.ORDER_DATE FROM CUSTOMER C
JOIN PURCHASE_ORDER O ON C.CUST_ID = O.CUST_ID
JOIN PRODUCT P ON P.PRODUCT_ID = O.PRODUCT_ID
WHERE P.PRODUCT_PRICE = (SELECT MAX (PRODUCT_PRICE)FROM PRODUCT );
```



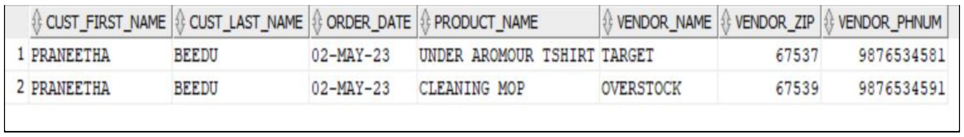
	CUST_FIRST_NAME	CUST_LAST_NAME	PRODUCT_NAME	PRODUCT_PRICE	ORDER_DATE
1	JANE	SMITH	NINTENDO SWITCH	294	02-MAY-23
2	DAVID	LEE	NINTENDO SWITCH	294	02-MAY-23

--18) Display names of customers and payment date who has payment type as CASH
SELECT C.CUST_FIRST_NAME, C.CUST_LAST_NAME, PY.PAYMENT_DATE,
PY.PAYMENT_TYPE FROM CUSTOMER C
JOIN PURCHASE_ORDER O ON C.CUST_ID = O.CUST_ID
JOIN PAYMENT PY ON PY.ORDER_ID = O.ORDER_ID
WHERE PY.PAYMENT_TYPE = 'CASH';



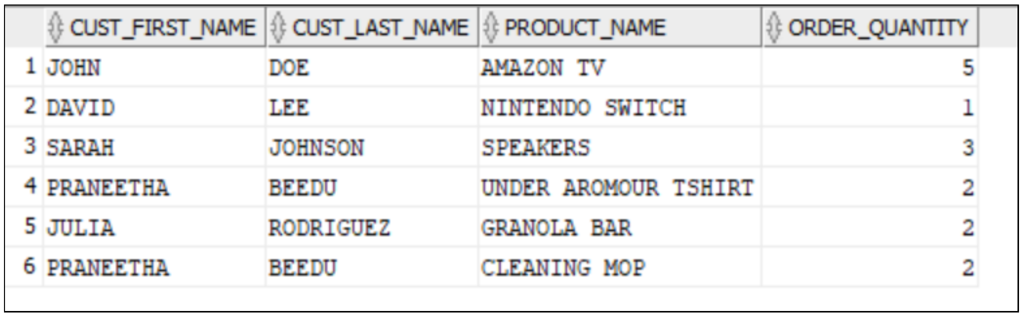
	CUST_FIRST_NAME	CUST_LAST_NAME	PAYMENT_DATE	PAYMENT_TYPE
1	PRANEETHA	BEEDU	02-MAY-23	CASH
2	PRANEETHA	BEEDU	02-MAY-23	CASH
3	JOHN	DOE	02-MAY-23	CASH
4	SARAH	JOHNSON	02-MAY-23	CASH
5	JASON	BROWN	02-MAY-23	CASH
6	EMMA	WILLIAMS	02-MAY-23	CASH

--19) Display vendor and product details where order is oplaced by Customer - Praneetha Beedu
SELECT C.CUST_FIRST_NAME, C.CUST_LAST_NAME, O.ORDER_DATE,
P.PRODUCT_NAME, V.VENDOR_NAME, V.VENDOR_ZIP, V.VENDOR_PHNUM FROM
CUSTOMER C
JOIN PURCHASE_ORDER O ON O.CUST_ID = C.CUST_ID
JOIN PRODUCT P ON P.PRODUCT_ID = O.PRODUCT_ID
JOIN VENDOR V ON V.VENDOR_ID = P.VENDOR_ID
WHERE CUST_FIRST_NAME = 'PRANEETHA' AND CUST_LAST_NAME = 'BEEDU';



	CUST_FIRST_NAME	CUST_LAST_NAME	ORDER_DATE	PRODUCT_NAME	VENDOR_NAME	VENDOR_ZIP	VENDOR_PHNUM
1	PRANEETHA	BEEDU	02-MAY-23	UNDER AROMOUR TSHIRT	TARGET	67537	9876534581
2	PRANEETHA	BEEDU	02-MAY-23	CLEANING MOP	OVERSTOCK	67539	9876534591

--20) Display customer and payment details where the order quantity is between 1 and 5
SELECT C.CUST_FIRST_NAME, C.CUST_LAST_NAME, P.PRODUCT_NAME, O.ORDER_QUANTITY FROM CUSTOMER C
JOIN PURCHASE_ORDER O ON O.CUST_ID = C.CUST_ID
JOIN PRODUCT P ON P.PRODUCT_ID = O.PRODUCT_ID
WHERE O.ORDER_QUANTITY BETWEEN '1' AND '5';



	CUST_FIRST_NAME	CUST_LAST_NAME	PRODUCT_NAME	ORDER_QUANTITY
1	JOHN	DOE	AMAZON TV	5
2	DAVID	LEE	NINTENDO SWITCH	1
3	SARAH	JOHNSON	SPEAKERS	3
4	PRANEETHA	BEEDU	UNDER AROMOUR TSHIRT	2
5	JULIA	RODRIGUEZ	GRANOLA BAR	2
6	PRANEETHA	BEEDU	CLEANING MOP	2