Q.1) Create following table using SQL statements.

Client table

Field name Data type Attributes

Clientno char(6) Primary key

Name Varchar(20) Not Null

City Varchar(50) Not Null

Pin Number

Mobile Char(10)

**Answer**:-

create table clienttable (Field_Name varchar(100), Clientno char(20), name varchar(100) not null, city varchar(100) not null, pinNumber numeric(30), mobile char(100), primary key(Clientno));

Q.2) write sql queries for the following:

1.create table grocery(item_no int, item_name varchar(20), mfd varchar (10), expdate varchar(12), address varchar(20), city varchar(10));

**Ans:- create table grocery(item_no numeric(30), item_name varchar(20), mfd varchar (10), expdate varchar(12), address varchar(20), city varchar(10));**

2.To drop the above created table

**Ans:-**

drop table grocery;

Q.3) write sql query for the following:

1. create database - emp

ANS:- create database emp;

**Note:- For Using This database execute below query**

use emp;

2.create table - employee with 4 columns

Ans:- create table employee(id numeric(20), name varchar(100), age numeric(20), address(100));

3.To see table structure

ANS:- desc employee;

Syntax:- desc tableName;

**Desc:- describe.**

**Q.4)**

| eid | ename | esalary | email | dob |
|---|---|---|---|---|
| 1 | XYZ | 3000.89 | xyz@gmail.com | 1990-08-30 |
| 2 | ABC | 2300.90 | abc@gmail.com | 1980-03-21 |

1] Write Query to insert 2 Records in above table using single query.

**Ans:- create table employee( eid numeric(20),ename varchar(100), esalary decimal(10,2),email varchar(100),dob varchar(100));**

**insert into employee values(1,'XYZ',3000.89,'xyz@gmail.com','1990-08-30'),(2,'ABC',2300.90,'abc@gmail.com','1980-08-21');**

2]Write Query to update esalary to 4000.68 and dob to 1993-08-30 of employee whose eid is 1 in a single query.

**update employee set esalary=4000.68 , dob='1993-08-30' where eid=1;**

3] Write Query to delete all records from the table in single query.

**Ans:- truncate table employee;**

4) create a table called CUSTOMERS and adds four columns, three of which, ID, NAME and AGE specify not to accept NULL.fourth column address accepts null value.

Ans:-

```
CREATE TABLE CUSTOMERS (
   ID numeric NOT NULL,
   NAME VARCHAR(255) NOT NULL,
   AGE numeric NOT NULL,
   ADDRESS VARCHAR(255)
);
```

**5) create a database called CARSHOWROOM having the schema as defined below It has the following four relations:**

1) INVENTORY: Stores name, price, model, year of manufacturing, and fuel type for each car in inventory of the showroom,
2) CUSTOMER: Stores customer id, name, address, phone number and email for each customer,
3) SALE: Stores the invoice number, car id, customer id, sale date, mode of payment, sales person's employee id and selling price of the car sold,
4) EMPLOYEE: Stores employee id, name, date of birth, date of joining, designation and salary of each employee in the showroom.

Ans:-

```
CREATE TABLE INVENTORY (
   car_id numeric PRIMARY KEY,
   name VARCHAR(255),
   price DECIMAL(10, 2),
   model VARCHAR(255),
   year_of_manufacturing numeric,
   fuel_type VARCHAR(255)
);

CREATE TABLE CUSTOMER (
   customer_id numeric PRIMARY KEY,
   name VARCHAR(255),
   address VARCHAR(255),
```

```
    phone_number VARCHAR(20),
    email VARCHAR(255)
);
CREATE TABLE EMPLOYEE (
    employee_id numeric PRIMARY KEY,
    name VARCHAR(255),
    date_of_birth DATE,
    date_of_joining DATE,
    designation VARCHAR(255),
    salary DECIMAL(10, 2)
);

CREATE TABLE SALE (
    invoice_number numeric PRIMARY KEY,
    car_id numeric,
    customer_id numeric,
    sale_date DATE,
    mode_of_payment VARCHAR(255),
    employee_id numeric,
    selling_price DECIMAL(10, 2),
    FOREIGN KEY (car_id) REFERENCES INVENTORY(car_id),
    FOREIGN KEY (customer_id) REFERENCES CUSTOMER(customer_id),
    FOREIGN KEY (employee_id) REFERENCES EMPLOYEE(employee_id)
);
```

7. FILTER IT FOR MUMBAI LOCATION

| ID | NAME | DEPARTMENT | SALARY | LOCATION |
|---|---|---|---|---|
| 1 | Jay | HR | 5000 | Mumbai |
| 2 | Yash | Management | 7000 | Pune |
| 3 | Annand | Training | 6000 | Chennai |
| 4 | Ruhi | Sales | 4000 | Delhi |
| 5 | Pratik | Admin | 3000 | Pune |
| 6 | Raj | Training | 5000 | Mumbai |

Ans:- Crate table and Insert Value:- create table employee(id numeric(20), name varchar(100), department varchar(100), salary numeric(30),location varchar(100));

```
insert into employee values
    (1,'Jay','HR',5000,'Mumbai'),
    (2,'Yash','Management',7000,'Pune'),
    (3,'Annand','Training',6000,'Channai'),
    (4,'Ruhi','Sales',4000,'Delhi'),
    (5,'Pratik','Admin',3000,'Pune'),
```

(6,'Raj','Training',5000,'Mumbai');

2. SHOW EMPLOYEES

Ans :- select * from employee;

3. UPDATE EMPLOYEE SALARY

Ans:-update employee set salary=89000 where id=1;

4. UPDATE EMPLOYEE DEPARTMENT

Ans:- update employee set department='Development' where id=2;

5. DELETE EMPLOYEE

Ans:- delete from  employee where id=1;

6. SORT BY SALARY.

Ans:- select * from employee order by salary;

7. FILTER IT FOR MUMBAI LOCATION

Ans:- select location from employee where department='IT'

**Q.8)**

| id | name | price |
|---|---|---|
| 1 | TV | 45000 |
| 2 | Bike | 85000 |
| 3 | Washing Machine | 40000 |
| 4 | Fan | 18000 |
| 5 | Car | 95000 |

ANS: ANS:-

```
CREATE TABLE products (
 id numeric(10),
 name VARCHAR(100),
 price numeric (32),
 quantity numeric(10)
);
```

```
INSERT INTO products (id, name, price, quantity)
VALUES
 (1, 'TV', 45000, 3),
 (2, 'Bike', 85000, 2),
 (3, 'Washing Machine', 40000, 4),
 (4, 'Fan', 18000, 10),
 (5, 'Car', 95000, 2);
```

1. Display all records with name and price fileds.

```
SELECT name, price FROM products;
```

2. display records of product with 20% discount on price

Ans:- SELECT name, price * 20/100 AS discounted_price

FROM products;

3. find the records of product whose price is less than 60000/-

Ans:- select * from products where price<60000;

4. find the records of product whose quantity is between 3 to 10

Ans:- select * from products where quantity between 3 and 10;

---

Consider a students table having different columns and values mentioned below:

| Stu_id | Stu_name | Stu_address | Stu_phno | Stu_percentage |
|---|---|---|---|---|
| 1 | Rahul | Agra | 9557806625 | 85 |
| 2 | Ankit | Delhi | 8855664471 | 75 |
| 3 | Shailendra | Noida | 7213457896 | 92 |

Write SQL queries for the following:

```
CREATE TABLE students (
    Stu_id INT,
    Stu_name VARCHAR(255),
    Stu_address VARCHAR(255),
    Stu_phno VARCHAR(255),
    Stu_percentage numeric
);
INSERT INTO students (Stu_id, Stu_name, Stu_address, Stu_phno, Stu_percentage)
VALUES (1, 'Rahul', 'Agra', '9557806625', 85);

INSERT INTO students (Stu_id, Stu_name, Stu_address, Stu_phno, Stu_percentage)
VALUES (2, 'Ankit', 'Delhi', '8855664471', 75);

INSERT INTO students (Stu_id, Stu_name, Stu_address, Stu_phno, Stu_percentage)
VALUES (3, 'Shailendra', 'Noida', '7213457896', 92);
```

1. Retrieve student id, student name, address, and percentage of all those students who scored more than 80 percent.

```
SELECT Stu_id, Stu_name, Stu_address, Stu_percentage
FROM students
WHERE Stu_percentage > 80;
```

2. Update the percentage of Rahul by 2 percent.

```
update students set stu_percentage=2 where stu_name='Rahul';
```

3. Student Ankit has left the school, so delete the whole record of him from the table.

```
DELETE FROM students WHERE Stu_name = 'Ankit';
```

---

| ID | NAME | DEPARTMENT | SALARY | LOCATION |
|---|---|---|---|---|
| 1 | Jay | HR | 5000 | Chennai |
| 2 | Yash | Management | 7000 | Pune |
| 3 | Annand | Training | 6000 | Chennai |
| 4 | Ruhi | Sales | 4000 | Delhi |

| | | | | |
|---|---|---|---|---|
| 5 | Pratik | Admin | 3000 | NULL |

Based on above table write SQL queries to:

```
-- Create table
CREATE TABLE employees (
    ID numeric
    NAME VARCHAR(50),
    DEPARTMENT VARCHAR(50),
    SALARY numeric
    LOCATION VARCHAR(50)
);

-- Insert data
INSERT INTO employees (ID, NAME, DEPARTMENT, SALARY, LOCATION)
VALUES
    (1, 'Jay', 'HR', 5000, 'Chennai'),
    (2, 'Yash', 'Management', 7000, 'Pune'),
    (3, 'Annand', 'Training', 6000, 'Chennai'),
    (4, 'Ruhi', 'Sales', 4000, 'Delhi'),
    (5, 'Pratik', 'Admin', 3000, NULL);
```

1. Display details employees whose name start with P
   Ans:- SELECT * FROM employees WHERE NAME LIKE 'P%';
2. Display names of employees getting paid in range 3000-5000
   Ans:- SELECT NAME FROM employees WHERE SALARY BETWEEN 3000 AND 5000;
3. Display all records in the decreasing order of salary
   Ans:- SELECT * FROM employees ORDER BY SALARY DESC;
4. Display name as ENAME and Department As Dept for the first 2 records
   Ans:- SELECT NAME AS ENAME, DEPARTMENT AS Dept FROM employees LIMIT 2;
5. Display details employees whose name has 'a' as second last letter
   Ans:- SELECT * FROM employees WHERE NAME LIKE '%a_';

---

| ID | NAME | DEPARTMENT | SALARY | LOCATION |
|---|---|---|---|---|
| 1 | Jay | HR | 5000 | Chennai |
| 2 | Yash | Management | 7000 | Pune |
| 3 | Annand | Training | 6000 | Chennai |
| 4 | Ruhi | Sales | 4000 | Delhi |
| 5 | Pratik | Admin | 3000 | NULL |

Based on above table write SQL queries to:

```
-- Create table
CREATE TABLE employees (
    ID numeric,
    NAME VARCHAR(50),
    DEPARTMENT VARCHAR(50),
    SALARY numeric,
    LOCATION VARCHAR(50)
);

-- Insert data
INSERT INTO employees (ID, NAME, DEPARTMENT, SALARY, LOCATION)
VALUES
    (1, 'Jay', 'HR', 5000, 'Chennai'),
    (2, 'Yash', 'Management', 7000, 'Pune'),
    (3, 'Annand', 'Training', 6000, 'Chennai'),
```

(4, 'Ruhi', 'Sales', 4000, 'Delhi'),

(5, 'Pratik', 'Admin', 3000, NULL);

1. Display Id and Name employees working in chennai or mumbai or pune.
   Ans:- SELECT ID, NAME FROM employees
   WHERE LOCATION IN ('Chennai', 'Mumbai', 'Pune');
2. Display name of employee whose location is not entered.
   Ans:- SELECT NAME
   FROM employees
   WHERE LOCATION IS NULL;

3.Display the list of employee in ascending order of their salary.

Ans:- SELECT * FROM employees ORDER BY SALARY ASC;

---

Consider the following MOVIE table and write the SQL

queries based on it.

| MovieID | MovieName | Category | ReleaseDate | ProductionCost | BusinessCost |
|---|---|---|---|---|---|
| 001 | Hindi_Movie | Musical | 2018-04-23 | 124500 | 130000 |
| 002 | Tamil_Movie | Action | 2016-05-17 | 112000 | 118000 |
| 003 | English_Movie | Horror | 2017-08-06 | 245000 | 360000 |
| 004 | Bengali_Movie | Adventure | 2017-01-04 | 72000 | 100000 |
| 005 | Telugu_Movie | Action | - | 100000 | - |
| 006 | Punjabi_Movie | Comedy | - | 30500 | - |

```
CREATE TABLE MOVIE (
    MovieID numeric,
    MovieName VARCHAR(100),
    Category VARCHAR(100),
    ReleaseDate DATE,
    ProductionCost numeric(30),
    BusinessCost numeric(30)
);

INSERT INTO MOVIE (MovieID, MovieName, Category, ReleaseDate, ProductionCost, BusinessCost)
VALUES
    ('001', 'Hindi_Movie', 'Musical', '2018-04-23', 124500, 130000),
    ('002', 'Tamil_Movie', 'Action', '2016-05-17', 112000, 118000),
    ('003', 'English_Movie', 'Horror', '2017-08-06', 245000, 360000),
    ('004', 'Bengali_Movie', 'Adventure', '2017-01-04', 72000, 100000),
    ('005', 'Telugu_Movie', 'Action', NULL, 100000, NULL),
    ('006', 'Punjabi_Movie', 'Comedy', NULL, 30500, NULL);
```

---

a) Display all the information from the Movie table.
   SELECT * FROM Movie;

b) List business done by the movies showing only
MovieID, MovieName and Total_Earning. Total_
Earning to be calculated as the sum of ProductionCost
and BusinessCost.

Ans:- SELECT MovieID, MovieName, (ProductionCost + BusinessCost) AS Total_Earning FROM Movie;

b) List the different categories of movies.

Ans: SELECT DISTINCT Category FROM Movie;

d) Find the net profit of each movie showing its
MovieID, MovieName and NetProfit. Net Profit is to be
calculated as the difference between Business Cost
and Production Cost.

Ans:- SELECT MovieID, MovieName, (BusinessCost - ProductionCost) AS NetProfit FROM Movie;

---

Write sql queries and explain the functions to perform the
following operations:

i) To display the day like “Monday”, “Tuesday”,
from the date when India got independence.

Ans:- SELECT DATE_FORMAT('1947-08-15', '%W') AS independence_day;

ii) To display the specified number of characters
from a particular position of the given string.

Ans:- SELECT SUBSTRING('Hello World', 7, 5) AS substring_result;

iii) To display the name of the month in which
you were born.

Ans:- SELECT DATE_FORMAT('1990-05-15', '%M') AS birth_month;

iv) To display your name in capital letters.

Ans:- SELECT UPPER('your_name') AS capitalized_name;

---

1. Write a query to concat strings "MySQL","is","Fun"

   Ans:- SELECT CONCAT('MySQL', ' ', 'is', ' ', 'Fun') AS concatenated_string;

2. Write query to replace "Coding" with "MySQL" in the sentence:
   "I Love Coding"

   Ans:- SELECT REPLACE('I Love Coding', 'Coding', 'MySQL') AS replaced_sentence;

3.Write a query to display the number of characters from "I Love India"

Ans:- SELECT LENGTH('I Love India') AS character_count;

---

Write commands in SQL for (i) to (iii) and output for (iv) and (v).

Table : Store

| StoreId | Name | Location | City | NoOfEmp | DateOpen | SalesAmt |
|---|---|---|---|---|---|---|
| S101 | Planet Fashion | Bandra | Mumbai | 7 | 2015-10-16 | 40000 |
| S102 | Vogue | Karol Bagh | Delhi | 8 | 2015-07-14 | 120000 |
| S103 | Trends | Powai | Mumbai | 10 | 2015-06-24 | 30000 |
| S104 | Super Fashion | Thane | Mumbai | 11 | 2015-02-06 | 45000 |
| S105 | Annabelle | South Extn. | Delhi | 8 | 2015-04-09 | 60000 |
| S106 | Rage | Defence Colony | Delhi | 5 | 2015-03-01 | 20000 |

---->>

```
CREATE TABLE Store (
    StoreId VARCHAR(10),
    Name VARCHAR(50),
    Location VARCHAR(50),
    City VARCHAR(50),
    NoOfEmp numeric,
    DateOpen DATE,
    SalesAmt numeric
);
```

```
INSERT INTO Store (StoreId, Name, Location, City, NoOfEmp, DateOpen, SalesAmt)
VALUES
    ('S101', 'Planet Fashion', 'Bandra', 'Mumbai', 7, '2015-10-16', 40000),
    ('S102', 'Vogue', 'Karol Bagh', 'Delhi', 8, '2015-07-14', 120000),
    ('S103', 'Trends', 'Powai', 'Mumbai', 10, '2015-06-24', 30000),
    ('S104', 'Super Fashion', 'Thane', 'Mumbai', 11, '2015-02-06', 45000),
    ('S105', 'Annabelle', 'South Extn. Delhi', 'Delhi', 8, '2015-04-09', 60000),
    ('S106', 'Rage', 'Defence Colony', 'Delhi', 5, '2015-03-01', 20000);
```

(i) To display names of stores along with Sales Amount of those stores that are located in Mumbai.

Ans:- SELECT Name, SalesAmt FROM Store WHERE City = 'Mumbai';

(ii) To display the details of store in alphabetical order of name.

Ans:- SELECT * FROM Store ORDER BY Name;

(iii) To display the City and the number of stores located in that City, only if number of stores is more than 2.

Ans:- SELECT City, COUNT(*) AS NumOfStores FROM Store GROUP BY City HAVING COUNT(*) > 2;

(iv) SELECT MIN(DATEOPEN) FROM STORE;

Ans:- SELECT MIN(DateOpen) FROM Store;

(v) SELECT COUNT(STOREID), NOOFEMP FROM STORE GROUP BY NOOFEMP HAVING MAX(SALESAMT)<60000;

Ans:- SELECT COUNT(StoreId), NoOfEmp

FROM Store

GROUP BY NoOfEmp

HAVING MAX(SalesAmt) < 60000;

---

ITEMS_ORDERED:

| customerid | order_date | item | quantity | price |
|---|---|---|---|---|
| 10330 | 30-Jun-1999 | Pogo stick | 1 | 28.00 |
| 10101 | 30-Jun-1999 | Raft | 1 | 58.00 |
| 10298 | 01-Jul-1999 | Skateboard | 1 | 33.00 |
| 10101 | 01-Jul-1999 | Life Vest | 4 | 125.00 |
| 10299 | 06-Jul-1999 | Parachute | 1 | 1250.00 |
| 10339 | 27-Jul-1999 | Umbrella | 1 | 4.50 |
| 10449 | 13-Aug-1999 | Unicycle | 1 | 180.79 |

CUSTOMERS:

| customerid | firstname | lastname | city | state |
|---|---|---|---|---|
| 10101 | John | Gray | Lynden | Washington |
| 10298 | Leroy | Brown | Pinetop | Arizona |
| 10299 | Elroy | Keller | Snoqualmie | Washington |
| 10315 | Lisa | Jones | Oshkosh | Wisconsin |
| 10325 | Ginger | Schultz | Pocatello | Idaho |
| 10329 | Kelly | Mendoza | Kailua | Hawaii |
| 10330 | Shawn | Dalton | Cannon Beach | Oregon |
| 10338 | Michael | Howell | Tillamook | Oregon |
| 10339 | Anthony | Sanchez | Winslow | Arizona |
| 10408 | Elroy | Cleaver | Globe | Arizona |

----→

```
CREATE TABLE ITEMS_ORDERED (
    customerid INT,
```

```
    order_date DATE,
    item VARCHAR(50),
    quantity INT,
    price DECIMAL(10, 2)
);

INSERT INTO ITEMS_ORDERED (customerid, order_date, item, quantity, price)
VALUES
    (10330, '1999-06-30', 'Pogo stick', 1, 28.00),
    (10101, '1999-06-30', 'Raft', 1, 58.00),
    (10298, '1999-07-01', 'Skateboard', 1, 33.00),
    (10101, '1999-07-01', 'Life Vest', 4, 125.00),
    (10299, '1999-07-06', 'Parachute', 1, 1250.00),
    (10339, '1999-07-27', 'Umbrella', 1, 4.50),
    (10449, '1999-08-13', 'Unicycle', 1, 180.79);
----------->
CREATE TABLE CUSTOMERS (
    customerid INT,
    firstname VARCHAR(50),
    lastname VARCHAR(50),
    city VARCHAR(50),
    state VARCHAR(50)
);

INSERT INTO CUSTOMERS (customerid, firstname, lastname, city, state)
VALUES
    (10101, 'John', 'Gray', 'Lynden', 'Washington'),
    (10298, 'Leroy', 'Brown', 'Pinetop', 'Arizona'),
    (10299, 'Elroy', 'Keller', 'Snoqualmie', 'Washington'),
    (10315, 'Lisa', 'Jones', 'Oshkosh', 'Wisconsin'),
    (10325, 'Ginger', 'Schultz', 'Pocatello', 'Idaho'),
    (10329, 'Kelly', 'Mendoza', 'Kailua', 'Hawaii'),
    (10330, 'Shawn', 'Dalton', 'Cannon Beach', 'Oregon'),
    (10338, 'Michael', 'Howell', 'Tillamook', 'Oregon'),
    (10339, 'Anthony', 'Sanchez', 'Winslow', 'Arizona'),
    (10408, 'Elroy', 'Cleaver', 'Globe', 'Arizona');
```

1.How many people are in each unique state in the customers table? Select the state and display the number of people in each. Hint: count is used to count rows in a column, sum works on numeric data only.

Ans:-
```
SELECT state, COUNT(*) AS number_of_people
FROM CUSTOMERS
GROUP BY state;
```

2.From the items_ordered table, select the item, maximum price, and minimum price for each specific item in the table. Hint: The items will need to be broken up into separate groups.

```
SELECT item, MAX(price) AS maximum_price, MIN(price) AS minimum_price

FROM ITEMS_ORDERED

GROUP BY item;
```

3.How many orders did each customer make? Use the items_ordered table. Select the customerid, number of orders they made, and the sum of their orders

```
SELECT customerid, COUNT(*) AS number_of_orders, SUM(quantity * price) AS total_order_amount

FROM ITEMS_ORDERED

GROUP BY customerid;
```

---

create two tables UNIFORM (UCode, UName, UColor) and COST (UCode, Size, Price) in the SchoolUniform database. UCode is Primary Key in table UNIFORM.

UCode and Size is the Composite Key in table COST.Therefore, Ucode is a common attribute between the two tables which can be used to fetch the common data from both the tables.

Uniform table:

| Ucode | Uname | Ucolor |
|---|---|---|
| 1 | Shirt | White |
| 2 | Pant | Grey |
| 3 | Tie | Blue |

Cost table:

| Ucode | Size | Price |
|---|---|---|
| 1 | L | 580 |
| 1 | M | 500 |
| 2 | L | 890 |
| 2 | M | 810 |

Write the following sql queries to List the UCode, UName, UColor, Size and Price of related tuples of tables UNIFORM and COST.

```
-- Create the UNIFORM table

CREATE TABLE UNIFORM (

    UCode numeric PRIMARY KEY,

    UName VARCHAR(50),

    UColor VARCHAR(50)

);


-- Insert records into the UNIFORM table

INSERT INTO UNIFORM (UCode, UName, UColor)

VALUES (1, 'Shirt', 'White'),

       (2, 'Pant', 'Grey'),

       (3, 'Tie', 'Blue');


-- Create the COST table
```

```
CREATE TABLE COST (
    UCode numeric
    Size VARCHAR(10),
    Price DECIMAL(10, 2),
    PRIMARY KEY (UCode, Size),
    FOREIGN KEY (UCode) REFERENCES UNIFORM(UCode)
);

-- Insert records into the COST table
INSERT INTO COST (UCode, Size, Price)
VALUES (1, 'L', 580),
       (1, 'M', 500),
       (2, 'L', 890),
       (2, 'M', 810);
```

a) Using condition in where clause

```
Ans:- SELECT UNIFORM.UCode, UNIFORM.UName, UNIFORM.UColor, COST.Size, COST.Price
FROM UNIFORM, COST
WHERE UNIFORM.UCode = COST.UCode;
```

b) Explicit use of JOIN clause

```
Ans:- SELECT UNIFORM.UCode, UNIFORM.UName, UNIFORM.UColor, COST.Size, COST.Price
FROM UNIFORM
JOIN COST ON UNIFORM.UCode = COST.UCode;
```

c) Explicit use of NATURAL JOIN clause

```
SELECT UNIFORM.UCode, UNIFORM.UName, UNIFORM.UColor, COST.Size, COST.Price
FROM UNIFORM
NATURAL JOIN COST;
```

---

ITEMS_ORDERED:

| customerid | order_date | item | quantity | price |
|---|---|---|---|---|
| 10330 | 30-Jun-1999 | Pogo stick | 1 | 28.00 |
| 10101 | 30-Jun-1999 | Raft | 1 | 58.00 |
| 10298 | 01-Jul-1999 | Skateboard | 1 | 33.00 |
| 10101 | 01-Jul-1999 | Life Vest | 4 | 125.00 |
| 10299 | 06-Jul-1999 | Parachute | 1 | 1250.00 |
| 10339 | 27-Jul-1999 | Umbrella | 1 | 4.50 |
| 10449 | 13-Aug-1999 | Unicycle | 1 | 180.79 |
| 10439 | 14-Aug-1999 | Ski Poles | 2 | 25.50 |
| 10101 | 18-Aug-1999 | Rain Coat | 1 | 18.30 |

CUSTOMERS:

| customerid | firstname | lastname | city | state |
|---|---|---|---|---|
| 10101 | John | Gray | Lynden | Washington |
| 10298 | Leroy | Brown | Pinetop | Arizona |
| 10299 | Elroy | Keller | Snoqualmie | Washington |
| 10315 | Lisa | Jones | Oshkosh | Wisconsin |
| 10325 | Ginger | Schultz | Pocatello | Idaho |
| 10329 | Kelly | Mendoza | Kailua | Hawaii |
| 10330 | Shawn | Dalton | Cannon Beach | Oregon |
| 10338 | Michael | Howell | Tillamook | Oregon |

| | | | | |
|---|---|---|---|---|
| 10339 | Anthony | Sanchez | Winslow | Arizona |
| 10408 | Elroy | Cleaver | Globe | Arizona |

1.Write a query using a join to determine which items were ordered by each of the customers in the customers table. Select the customerid, firstname, lastname, order_date, item, and price for everything each customer purchased in the items_ordered table.

Ans:- SELECT c.customerid, c.firstname, c.lastname, io.order_date, io.item, io.price

FROM customers c

JOIN items_ordered io ON c.customerid = io.customerid;

2.Repeat exercise #1, however display the results sorted by state in descending order.

Ans:- SELECT c.customerid, c.firstname, c.lastname, io.order_date, io.item, io.price

FROM customers c

JOIN items_ordered io ON c.customerid = io.customerid

ORDER BY c.state DESC;

---

SUBQUERIES:

CUSTOMER TABLE:

| CUST_ID | NAME | OCCUPATION | AGE |
|---|---|---|---|
| 101 | PETER | ENGINEER | 32 |
| 102 | JOSEPH | DEVELOPER | 30 |
| 103 | JOHN | LEADER | 28 |
| 104 | STEPHEN | SCIENTIST | 45 |
| 105 | SUZI | CARPENTER | 26 |
| 106 | BOB | ACTOR | 25 |
| 107 | NULL | NULL | NULL |

ORDERS TABLE:

| ORDER_ID | CUST_ID | PROD_NAME | ORDER_DATE |
|---|---|---|---|
| 1 | 101 | LAPTOP | 2022-01-10 |
| 2 | 103 | DESKTOP | 2022-02-11 |
| 3 | 106 | IPHONE | 2022-03-13 |
| 4 | 104 | MOBILE | 2022-03-05 |
| 5 | 102 | TV | 2022-03-20 |

1. Find the details of the customers whose details is not in the customer table.
   Ans:- SELECT *
   FROM CUSTOMER
   WHERE CUST_ID NOT IN (SELECT CUST_ID FROM ORDERS);
2. The customer details who have not placed an order.
   Ans:- SELECT *
   FROM CUSTOMER
   WHERE CUST_ID NOT IN (SELECT CUST_ID FROM ORDERS);

3. Find the name of the customers who has purchased laptop.
   SELECT NAME
   FROM CUSTOMER
   WHERE CUST_ID IN (SELECT CUST_ID FROM ORDERS WHERE PROD_NAME = 'LAPTOP');
4. Find the details of customers who purchased iphone.
   SELECT *
   FROM CUSTOMER
   WHERE CUST_ID IN (SELECT CUST_ID FROM ORDERS WHERE PROD_NAME = 'IPHONE');

5. Find the details of the customers whose details is not in the orders table.
   Ans:- SELECT *
   FROM CUSTOMER
   WHERE CUST_ID NOT IN (SELECT CUST_ID FROM ORDERS);
6. How many customers from customers table has made an order.
   Ans:- SELECT COUNT(DISTINCT CUST_ID) AS num_customers
   FROM ORDERS;

---

Create following table named as Student_info:  sid is primary key auto_incremented

| sid | name | dept | percentage |
|---|---|---|---|
| 1 | Harry | IT | 95.26 |
| 2 | Mac | CS | 98.45 |
| 3 | Rox | Mechanical | 75.69 |
| 4 | Shree | IT | 85.65 |
| 5 | Ramesh | CS | 89.45 |

Assume there are 1000 records of the student in the student_info table and solve the following questions.

Find the records  of the students scoring the highest percentage.

Ans:- SELECT * FROM Student_info WHERE percentage = (SELECT MAX(percentage) FROM Student_info);

Find the information of the students who are having the second highest percentage in the table.

Ans:- SELECT * FROM Student_info WHERE percentage = (SELECT MAX(percentage) FROM Student_info WHERE percentage < (SELECT MAX(percentage) FROM Student_info));