**Practical 1**

**Aim:** Implementation of DDL commands of SQL with suitable examples

* 1. Create table b) Alter table c) Truncate table d) Drop table

Implementation of DML commands of SQL with suitable examples

1. Select b) Insert c) Update d) Delete

**Solution:**

CREATE, ALTER, DROP, TRUNCATE

(i) Create following tables according to the following definition.

**DEPOSIT**: Actno, Cname, Bname, Amount, Adate

**BRANCH**: Bname,City

**CUSTOMERS**: Cname, City

**BORROW**: Loanno, Cname, Bname, Amount

|  |  |
| --- | --- |
| **BORROW** | |
| **Columnname** | **Size** |
| Loanno | 5 |
| CNAME | 20 |
| BNAME | 20 |
| AMOUNT | 8,2 |

|  |  |
| --- | --- |
| **DEPOSIT** | |
| **Columnname** | **Size** |
| Actno | 5 |
| Cname | 20 |
| Bname | 20 |
| Amount | 8,2 |
| Adate | |

|  |  |
| --- | --- |
| **BRANCH** | |
| **Columnname** | **Size** |
| Bname | 20 |
| City | 20 |

|  |  |
| --- | --- |
| **CUSTOMERS** | |
| **Columnname** | **Size** |
| Cname | 20 |
| City | 20 |

**Query:** For DEPOSIT table - CREATE TABLE DEPOSIT (

Actno INT,

Cname VARCHAR(20),

Bname VARCHAR(20),

Amount DECIMAL(8,2),

Adate DATE

);

For BRANCH table - CREATE TABLE BRANCH (

Bname VARCHAR(20),

City VARCHAR(20)

);

For CUSTOMERS table - CREATE TABLE CUSTOMERS (

Cname VARCHAR(20),

City VARCHAR(20)

);

For BORROW table - CREATE TABLE BORROW (

Loanno INT,

Cname VARCHAR(20),

Bname VARCHAR(20),

Amount DECIMAL(8,2)

);

1. Add new column DOB into the CUSTOMER table.

**Query:** ALTER TABLE customers ADD dob DATE;

1. Add LOCATION column in BRANCH table.

**Query:** ALTER TABLE branch ADD location VARCHAR(20);

1. Modify customer name space to 30 characters.

**Query:** ALTER TABLE customers MODIFY Cname VARCHAR (30);

ALTER TABLE deposit MODIFY Cname VARCHAR (30);

ALTER TABLE borrow MODIFY Cname VARCHAR (30);

1. Change datatype of Loan number column to numeric type.

**Query**: ALTER TABLE borrow MODIFY loanno NUMERIC;

1. Delete LOCATION column.

**Query**: ALTER TABLE branch DROP COLUMN location;

1. Create DEPOSIT\_TMP table that contains all information of DEPOSIT table.

**Query**: CREATE TABLE deposit\_tmp AS SELECT \* FROM deposit;

1. Create BRANCH\_TMP table that contains only structure same as BRANCH table.

**Query**: CREATE TABLE branch\_tmp AS SELECT \* FROM branch WHERE false;

1. Remove all rows from DEPOSIT\_TMP table.

**Query**: TRUNCATE TABLE deposit\_tmp;

1. Delete DEPOSIT\_TMP.

**Query**: DROP TABLE deposit\_tmp;

(ii) Insert the data as shown below.

**DEPOSIT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ACTNO** | **CNAME** | **BNAME** | **AMOUNT** | **ADATE** |
| 1 | Abhinav | STATE BANK OF INDIA | 1000.00 | 1-MAR-95 |
| 2 | Aditya | KOTAK MAHINDRA BANK | 5000.00 | 4-JAN-96 |
| 3 | Minal | ANDHRA BANK | 3500.00 | 17-NOV-95 |
| 4 | Shrushti | YES BANK | 1200.00 | 17-DEC-95 |
| 5 | Radha | DENA BANK | 3000.00 | 27-MAR-96 |
| 6 | Akash | UCO | 2000.00 | 31-MAR-96 |
| 7 | Aniket | CITY | 1000.00 | 5-SEP-95 |
| 8 | Chandan | UNION | 5000.00 | 2-JUL-95 |
| 9 | Dushant | NAGPUR | 7000.00 | 10-AUG-95 |

**Query:** INSERT INTO deposit (Actno, Cname, Bname, Amount, Adate)

VALUES (1, 'Abhinav', 'STATE BANK OF INDIA', 1000.00, '03-01-1995')

**Query:** INSERT ALL

INTO deposit VALUES (2, 'Aditya', 'KOTAK MAHINDRA BANK', 5000.00, '01-04-1996')

INTO deposit VALUES (3, 'Minal', 'ANDHRA BANK', 3500.00, '11-17-1995')

INTO deposit VALUES (4, 'Shrushti', 'YES BANK', 1200.00, '12-17-1995')

INTO deposit VALUES (5, 'Radha', 'DENA BANK', 3000.00, '03-27-1996')

INTO deposit VALUES (6, 'Akash', 'UCO', 2000.00, '03-31-1996')

INTO deposit VALUES (7, 'Aniket', 'CITY', 1000.00, '09-05-1995')

INTO deposit VALUES (8, 'Chandan', 'UNION', 5000.00, '07-02-1995')

INTO deposit VALUES (9, 'Dushant', 'NAGPUR', 7000.00, '08-10-1995')

SELECT \* FROM DUAL;

**BRANCH**

|  |  |
| --- | --- |
| **BNAME** | **CITY** |
| STATE BANK OF INDIA | NAGPUR |
| KOTAK MAHINDRA BANK | NAGPUR |
| ANDHRA BANK | DELHI |
| YES BANK | DELHI |
| DENA BANK | NAGPUR |
| UCO | BANGLORE |
| CITY | BOMBAY |
| UNION | BOMBAY |
| STATE BANK OF INDIA | DELHI |
| KOTAK MAHINDRA BANK | BOMBAY |

**Query:** INSERT ALL

INTO branch VALUES ('STATE BANK OF INDIA','NAGPUR')

INTO branch VALUES ('KOTAK MAHINDRA BANK', 'NAGPUR')

INTO branch VALUES ('ANDHRA BANK', 'DELHI')

INTO branch VALUES ('YES BANK', 'DELHI')

INTO branch VALUES ('DENA BANK', 'NAGPUR')

INTO branch VALUES ('UCO', 'BANGLORE')

INTO branch VALUES ('CITY', 'BOMBAY')

INTO branch VALUES ('UNION', 'BOMBAY')

INTO branch VALUES ('STATE BANK OF INDIA', 'DELHI')

INTO branch VALUES ('KOTAK MAHINDRA BANK', 'BOMBAY')

SELECT \* FROM DUAL;

**CUSTOMERS**

|  |  |
| --- | --- |
| **CNAME** | **CITY** |
| Abhinav | CALCUTTA |
| Aditya | DELHI |
| Mrinal |  |
| Shrushti | PATNA |
| Radha | NAGPUR |
| Akash | NAGPUR |
| Aniket | SURAT |
| Chandan | BOMBAY |
| Dushant | BOMBAY |

**Query:** INSERT INTO customers (cname, city) VALUES ('Abhinav', 'CALCUTTA');

INSERT INTO customers (cname, city) VALUES ('Aditya', 'DELHI');

**Query:** INSERT INTO customers VALUES ('Mrinal', null);

**Query:** INSERT ALL

INTO customers VALUES ('Shrushti', 'PATNA')

INTO customers VALUES ('Radha', 'NAGPUR')

INTO customers VALUES ('Akash', 'NAGPUR')

INTO customers VALUES ('Aniket', 'SURAT')

INTO customers VALUES ('Chandan', 'BOMBAY')

INTO customers VALUES ('Dushant', 'BOMBAY')

SELECT \* FROM DUAL;

**BORROW**

|  |  |  |  |
| --- | --- | --- | --- |
| **LOANNO** | **CNAME** | **BNAME** | **AMOUNT** |
| 10 | Abhinav | STATE BANK OF INDIA | 1000.00 |
| 20 | Aditya | KOTAK MAHINDRA BANK | 5000.00 |
| 30 | Minal | ANDHRA BANK | 3000.00 |
| 40 | Shrushti | YES BANK | 2000.00 |
| 50 | Radha | DENA BANK | 8000.00 |
| 60 | Akash | UCO | 3000.00 |
| 70 | Aniket | CITY | 4000.00 |

**Query:** INSERT ALL

INTO borrow VALUES (10,'Abhinav','STATE BANK OF INDIA', 1000.00)

INTO borrow VALUES (20,'Aditya','KOTAK MAHINDRA BANK', 5000.00)

INTO borrow VALUES (30,'Minal','ANDHRA BANK', 3000.00)

INTO borrow VALUES (40,'Shrushti','YES BANK', 2000.00)

INTO borrow VALUES (50,'Radha','DENA BANK', 8000.00)

INTO borrow VALUES (60,'Akash','UCO', 3000.00)

INTO borrow VALUES (70,'Aniket','CITY', 4000.00)

SELECT \* FROM DUAL;

(iii) From the above given tables perform the following queries:

1. Describe deposit, branch.

**Query:** DESC deposit;

DESC branch;

1. Describe borrow, customers.

**Query:** DESC borrow;

DESC customers;

1. List all data from table DEPOSIT.

**Query:** SELECT \* FROM deposit;

1. Give account no and amount of depositors.

**Query:** SELECT actno AS AccountNumber, amount FROM deposit;

1. Give name of depositors having amount greater than 4000.

**Query:** SELECT cname FROM deposit WHERE amount > 4000;

1. List all data from table BORROW.

**Query:** SELECT \* FROM borrow;

1. Give customer name and loan amount of borrowers.

**Query:** SELECT cname AS customername, amount FROM borrow;

1. Give name of borrows from YES BANK branch and loan amount greater than 3000.

**Query:** SELECT cname WHERE bname='YES BANK' AND amount >3000;

1. List all data from table CUSTOMERS.

**Query:** SELECT \* FROM customers;

1. List name of customers from BOMBAY.

**Query:** SELECT cname FROM customers WHERE city='BOMBAY';

1. List all data from table BRANCH.

**Query:** SELECT \* FROM branch;

1. Give name of customers who opened account after date '1-12-96'.

**Query:** SELECT cname FROM deposit WHERE adate > '12-01-96';

(iv) Change the name of borrow to loan (rename borrow to loan).

**Query:** ALTER TABLE borrow RENAME TO loan;

(v) Give the SQL solutions for following queries

1. Select branch names from the loan relation

**Query:** SELECT bname FROM loan;

1. List all the tuples & attributes from the loan relation

**Query:** SELECT \* FROM loan;

1. Select branch\_name & loan no. From the loan relation.

**Query:** SELECT bname, loanno FROM loan;

1. Select all the loan numbers & amount from the loan relation.

**Query:** SELECT loanno, amount FORM loan;

1. Find all loan no. & amount of the DENA BANK branch where the amount is Greater than 1300

**Query:** SELECT loanno, amount FROM loan WHERE bname='DENA BANK' AND

amount > 1300;

1. Give 10% interest to all depositors.

**Query:** UPDATE deposit SET amount = amount \* 1.10;

1. Give 10% interest to all depositors having branch UCO

**Query:** UPDATE deposit SET amount = amount \* 1.10 WHERE bname='UCO';

1. Delete deposit of VIJAY

**Query:** DELETE FROM deposit WHERE cname='VIJAY';

1. Delete customers from BOMBAY city

**Query:** DELETE FROM customers WHERE city='BOMBAY';

1. Delete depositors if the branch is ANDHRA BANK and depositor name is Aditya Karansingh Kachhawah.

**Query:** DELETE FROM deposit WHERE bname='ANDHRA BANK' AND cname='Aditya

Karansingh Kachhawah';

1. Delete depositors having deposit less than 500

**Query:** DELETE FROM deposit WHERE amount <500;

1. Delete borrower having loan more than 10000

**Query:** DELETE FROM loan WHERE amount >10000;

1. Delete borrower having loan more than 1000 and branch KOTAK MAHINDRA BANK

**Query:** DELETE FROM loan WHERE amount >1000 AND bname='KOTAK

MAHINDRA BANK';

1. Delete borrower having branch name UCO

**Query:** DELETE FROM loan WHERE bname='UCO'

**Practical 2**

**Aim:** Study and implementation of different types of constraints.

Create the tables described below

**PRODUCT\_MASTER**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Datatype** | **Size** | **Constraints** |
| Product\_No | Char | 6 | Primary Key |
| Description | Varchar2 | 15 | Not Null |
| ProfitPercent | Number | 4,2 | Not Null |
| UnitMeasure | Varchar2 | 10 | Not Null |
| SellPrice | Number | 8,2 | Not Null |
| CostPrice | Number | 8,2 | Not Null |

|  |  |
| --- | --- |
| SELLPRICE | Between 10000 to 80000 |
| COSTPRICE | >1000 |

**SALESMAN\_MASTER**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Datatype** | **Size** | **Constraints** |
| **Salesman\_No** | Char | 6 | Primary Key |
| **SalesmanName** | Varchar2 | 20 | Not Null |
| **Address** | Varchar2 | 30 | Not Null |
| **City** | Varchar2 | 20 |  |
| **PinCode** | Number | 6 |  |
| **State** | Varchar2 | 20 |  |
| **SalAmt** | Number | 8,2 | Not Null |

|  |  |
| --- | --- |
| CITY | CAN’T BE NAGPUR |
| PINCODE | UNIQUE |

**Client\_Master**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Datatype** | **Size** | **Constraints** |
| Client\_No | Char | 6 | Primary Key |
| Name | Varchar2 | 20 | Not Null |
| Address | Varchar2 | 100 |  |
| City | Varchar2 | 15 |  |
| Pincode | Number | 6 |  |
| State | Varchar2 | 20 |  |
| Bal\_Due | Number | 8,2 |  |

**Sales\_Order**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column Name** | **Datatype** | **Size** | **Default** | **Constraints** |
| Order\_No | Char | 6 |  | Primary Key |
| Client\_No | Char | 6 |  | Foreign Key References Client\_No of Client\_Master Table |
| OrderDate | Date |  |  | Not Null |
| Salesman\_No | Char | 6 |  | Foreign Key References Salesman\_No of Salesman\_Master Table |

**Solution Query:**

1. **For PRODUCT\_MASTER Table:**

CREATE TABLE **PRODUCT\_MASTER** (

Product\_No CHAR(6) PRIMARY KEY,

Description VARCHAR2(15) NOT NULL,

ProfitPercent NUMBER(4,2) NOT NULL,

UnitMeasure VARCHAR2(10) NOT NULL,

SellPrice NUMBER(8,2) NOT NULL CHECK (SellPrice BETWEEN 10000 AND 80000),

CostPrice NUMBER(8,2) NOT NULL CHECK (CostPrice > 1000)

);

1. **For SALESMAN\_MASTER Table:**

CREATE TABLE **SALESMAN\_MASTER** (

Salesman\_No CHAR(6) PRIMARY KEY,

SalesmanName VARCHAR2(20) NOT NULL,

Address VARCHAR2(30) NOT NULL,

City VARCHAR2(20),

PinCode NUMBER(6),

State VARCHAR2(20),

SalAmt NUMBER(8,2) NOT NULL,

CONSTRAINT pincode\_unique UNIQUE (PinCode),

CONSTRAINT city\_not\_nagpur CHECK (City <> 'NAGPUR')

);

1. **For CLIENT\_MASTER Table:**

CREATE TABLE **CLIENT\_MASTER** (

Client\_No CHAR(6) PRIMARY KEY,

Name VARCHAR2(20) NOT NULL,

Address VARCHAR2(100),

City VARCHAR2(15),

Pincode NUMBER(6),

State VARCHAR2(20),

Bal\_Due NUMBER(8,2)

);

1. **For SALES\_ORDER Table:**

CREATE TABLE **SALES\_ORDER** (

Order\_No CHAR(6) PRIMARY KEY,

Client\_No CHAR(6),

OrderDate DATE NOT NULL,

Salesman\_No CHAR(6),

FOREIGN KEY (Client\_No) REFERENCES CLIENT\_MASTER(Client\_No),

FOREIGN KEY (Salesman\_No) REFERENCES SALESMAN\_MASTER(Salesman\_No)

);

**Practical 3**

**Aim:** Study and implementation of different types of constraints.

Create the tables described below

**PRODUCT\_MASTER**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Datatype** | **Size** | **Constraints** |
| Product\_No | Char | 6 | Primary Key |
| Description | Varchar2 | 15 | Not Null |
| ProfitPercent | Number | 4,2 | Not Null |
| UnitMeasure | Varchar2 | 10 | Not Null |
| SellPrice | Number | 8,2 | Not Null |
| CostPrice | Number | 8,2 | Not Null |

|  |  |
| --- | --- |
| SELLPRICE | Between 10000 to 80000 |
| COSTPRICE | >1000 |

**SALESMAN\_MASTER**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Datatype** | **Size** | **Constraints** |
| **Salesman\_No** | Char | 6 | Primary Key |
| **SalesmanName** | Varchar2 | 20 | Not Null |
| **Address** | Varchar2 | 30 | Not Null |
| **City** | Varchar2 | 20 |  |
| **PinCode** | Number | 6 |  |
| **State** | Varchar2 | 20 |  |
| **SalAmt** | Number | 8,2 | Not Null |

|  |  |
| --- | --- |
| CITY | CAN’T BE NAGPUR |
| PINCODE | UNIQUE |

**Client\_Master**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Datatype** | **Size** | **Constraints** |
| Client\_No | Char | 6 | Primary Key |
| Name | Varchar2 | 20 | Not Null |
| Address | Varchar2 | 100 |  |
| City | Varchar2 | 15 |  |
| Pincode | Number | 6 |  |
| State | Varchar2 | 20 |  |
| Bal\_Due | Number | 8,2 |  |

**Sales\_Order**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column Name** | **Datatype** | **Size** | **Default** | **Constraints** |
| Order\_No | Char | 6 |  | Primary Key |
| Client\_No | Char | 6 |  | Foreign Key References Client\_No of Client\_Master Table |
| OrderDate | Date |  |  | Not Null |
| Salesman\_No | Char | 6 |  | Foreign Key References Salesman\_No of Salesman\_Master Table |

**Solution Query:**

1. **For PRODUCT\_MASTER Table:**

CREATE TABLE **PRODUCT\_MASTER** (

Product\_No CHAR(6) PRIMARY KEY,

Description VARCHAR2(15) NOT NULL,

ProfitPercent NUMBER(4,2) NOT NULL,

UnitMeasure VARCHAR2(10) NOT NULL,

SellPrice NUMBER(8,2) NOT NULL CHECK (SellPrice BETWEEN 10000 AND 80000),

CostPrice NUMBER(8,2) NOT NULL CHECK (CostPrice > 1000)

);

1. **For SALESMAN\_MASTER Table:**

CREATE TABLE **SALESMAN\_MASTER** (

Salesman\_No CHAR(6) PRIMARY KEY,

SalesmanName VARCHAR2(20) NOT NULL,

Address VARCHAR2(30) NOT NULL,

City VARCHAR2(20),

PinCode NUMBER(6),

State VARCHAR2(20),

SalAmt NUMBER(8,2) NOT NULL,

CONSTRAINT pincode\_unique UNIQUE (PinCode),

CONSTRAINT city\_not\_nagpur CHECK (City <> 'NAGPUR')

);

1. **For CLIENT\_MASTER Table:**

CREATE TABLE **CLIENT\_MASTER** (

Client\_No CHAR(6) PRIMARY KEY,

Name VARCHAR2(20) NOT NULL,

Address VARCHAR2(100),

City VARCHAR2(15),

Pincode NUMBER(6),

State VARCHAR2(20),

Bal\_Due NUMBER(8,2)

);

1. **For SALES\_ORDER Table:**

CREATE TABLE **SALES\_ORDER** (

Order\_No CHAR(6) PRIMARY KEY,

Client\_No CHAR(6),

OrderDate DATE NOT NULL,

Salesman\_No CHAR(6),

FOREIGN KEY (Client\_No) REFERENCES CLIENT\_MASTER(Client\_No),

FOREIGN KEY (Salesman\_No) REFERENCES SALESMAN\_MASTER(Salesman\_No)

);

**Practical 4**

**Aim:** Study and Implementation of

1. Aggregate functions
2. Group By & Having clause
3. Order by clause

**Table: Furniture**

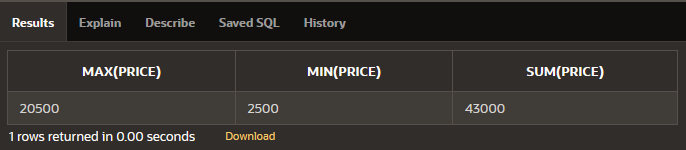
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **FCODE** | **NAME** | **MATERIAL** | **PRICE** | **MANUFDATE** | **WCODE** |
| 1002 | Coffee Chair | Wood | 3000 | 19-NOV-2017 | W01 |
| 1003 | Dining table | Wood | 20500 | 12-JAN-2019 | W02 |
| 1004 | Coffee Table | Glass | 5000 | 06-JAN-2019 | W02 |
| 1005 | Chair | Wood | 2500 | 07-MAY-2017 | W01 |
| 1006 | Recliner | Fibre | 12000 | 31-MAR-2018 | W03 |

**Solution:**

1. Find the maximum, minimum, and total price.

**Query:** SELECT MAX(price), MIN(price), SUM(price) FROM Furniture;

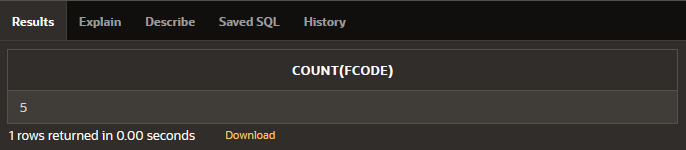
**Output:**

****

1. Find the total number of furniture.

**Query:** SELECT COUNT(fcode) FROM Furniture;

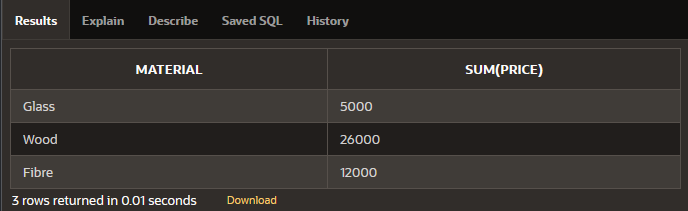
**Output:**

****

1. List the total price for each material.

**Query:** SELECT material, SUM(price) FROM Furniture GROUP BY (material);

**Output:**

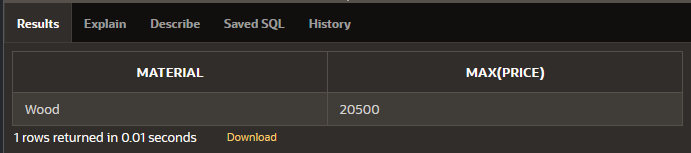


1. Give maximum price for wood.

**Query:** SELECT material, MAX(price) FROM Furniture GROUP BY (material)

HAVING material='Wood';

**Output:**

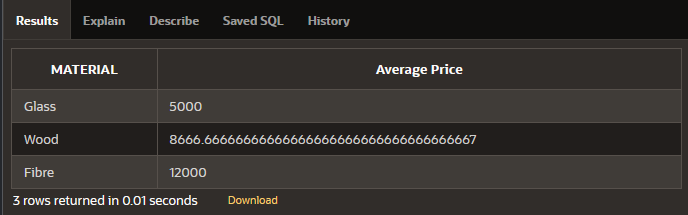


1. Display the average price of each material.

**Query:** SELECT material, AVG(price) AS "Average Price" FROM Furniture

GROUP BY (material);

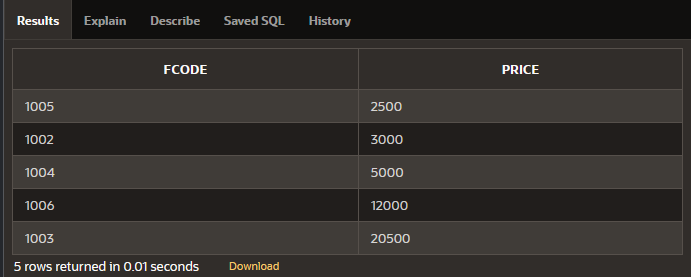
**Output:**



1. List fcode and the amount of all furniture in ascending order of price.

**Query:** SELECT fcode, price FROM Furniture ORDER BY price ASC;

**Output:**

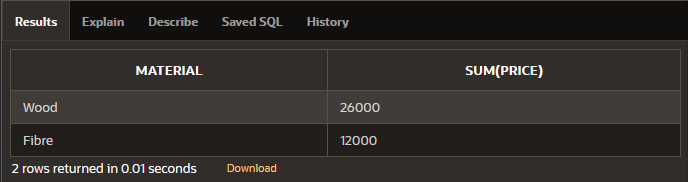


1. List the materials having a sum of price more than 10000.

**Query:** SELECT material, SUM(price) FROM Furniture GROUP BY (material)

HAVING SUM(price)>10000;

**Output:**



1. Give the name of furniture whose number of wcode is less than 2.

**Query:** SELECT name FROM Furniture

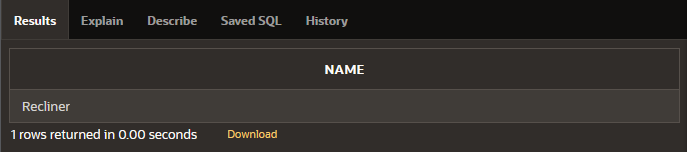
WHERE wcode IN (

SELECT wcode FROM Furniture GROUP BY wcode

HAVING COUNT(wcode) < 2

);

**Output:**



1. Give the name of materials whose average price is greater than 15000.

**Query:** SELECT material, AVG(price) FROM Furniture GROUP BY (material)

HAVING AVG(price)>15000;

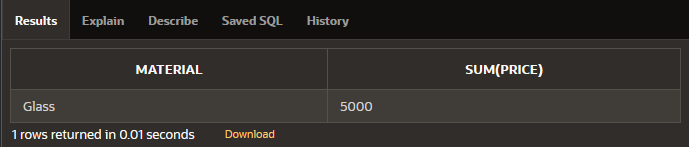
**Output:** No data found.

1. Give the name of materials whose total price is less than 6000.

**Query:** SELECT material, SUM(price) FROM Furniture GROUP BY (material)

HAVING SUM(price)<6000;

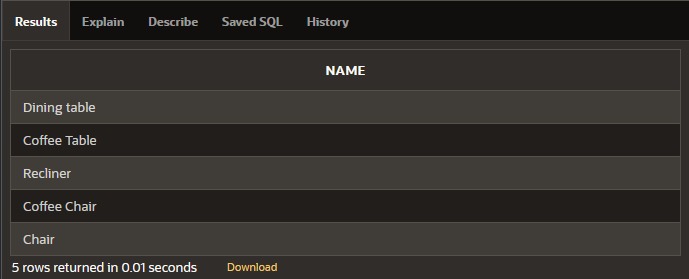
**Output:**



1. Display the name of the furnitures in descending order of manufacturing date.

**Query:** SELECT name FROM Furniture ORDER BY manufdate DESC;

**Output:**



**Practical 5**

**Aim:** Study and Implementation of different types of joins like cross join, natural join, inner join, and outer joins.

**Solution:**

1. Find the details of customer Aditya.

**Query:** SELECT \* FROM customers NATURAL JOIN deposit WHERE Cname='Aditya'

1. Find the names of customers who are borrowers and depositors and having living city Nagpur.

**Query:** SELECT \* FROM customers,loan,deposit

WHERE customers.cname=loan.cname AND customers.cname=deposit.cname

AND customers.city='NAGPUR'

1. Find the name, department number, and department name of all employees.

**Query:** SELECT ename,emp.deptno,dname FROM emp,dept

WHERE dept.deptno=emp.deptno

1. Display all jobs that are in department 30. Include the location of the department in the output.

**Query:** SELECT job,loc FROM emp,dept

WHERE dept.deptno=emp.deptno AND emp.deptno=30

1. Find the employee name, department number, and department name for all employees who work in NEW YORK.

**Query:** SELECT ename,emp.deptno,dname FROM emp,dept

WHERE emp.deptno = dept.deptno AND dept.loc='NEW YORK'

1. Find the name, account no, amount and city of customer ANIL.

**Query:** SELECT customers.cname,actno,amount,city FROM customers,deposit

WHERE customers.cname=deposit.cname AND customers.cname='ANIL'

1. Find the names of customers who are borrowers and depositors.

**Query:** SELECT deposit.cname FROM deposit,loan WHERE deposit.cname=loan.cname

1. Find the names of customers who are borrowers and depositors and having living city Nagpur and branch city mumbai.

**Query:** SELECT \* FROM customers,loan,deposit,branch

WHERE customers.cname=loan.cname AND customers.cname=deposit.cname

AND deposit.bname=branch.bname AND customers.city='NAGPUR'

AND branch.city='MUMBAI'

1. Find the name, department number, and department name of all employees having salary 5000

**Query:** SELECT ename,emp.deptno,dname FROM emp,dept

WHERE emp.deptno = dept.deptno AND emp.sal >5 000

1. Retrieve all employees and their department names, including those without a department.

**Query:** SELECT \* FROM emp LEFT OUTER JOIN dept ON emp.deptno = dept.deptno

1. Retrieve all departments and their employees, including departments without employees.

**Query:** SELECT \* FROM emp RIGHT OUTER JOIN dept ON emp.deptno = dept.deptno

1. Retrieve all employees and departments.

**Query:** SELECT \* FROM emp FULL OUTER JOIN dept ON emp.deptno = dept.deptno

1. Retrieve employees and their department names, sorting the result by department name in ascending order.

**Query:** SELECT \* FROM emp, dept WHERE emp.deptno = dept.deptno

ORDER BY dept.dname ASC

1. Retrieve all customers and their deposit amounts, including those without deposits.

**Query:** SELECT customers.cname,deposit.amount

FROM customers LEFT OUTER JOIN deposit

ON customers.cname = deposit.cname

1. Retrieve all deposits and the corresponding customer information, including deposits without associated customers.

**Query:** SELECT \* FROM customers RIGHT OUTER JOIN deposit

ON customers.cname = deposit.cname

1. Retrieve all customers and deposits, including those without matches on either side.

**Query:** SELECT \* FROM customers FULL OUTER JOIN deposit

ON customers.cname = deposit.cname

**Practical 6**

**Aim:** Study and Implementation of Sub queries

**Solution:**

Employee (emp\_no, emp\_name, emp\_sal, emp\_comm, dept\_no, Job\_id, location)

Job (job\_id, job\_title, min\_sal, max\_sal)

department (dept\_no, dept\_name, location\_id).

employee\_manager (emp\_no, emp\_hiredate, mng\_no,mng\_name).

**Query:** CREATE TABLE Employee (

emp\_no INT PRIMARY KEY,

emp\_name VARCHAR(100),

emp\_sal DECIMAL(10, 2),

emp\_comm DECIMAL(10, 2),

dept\_no INT, Job\_id INT,

location VARCHAR(100)

);

CREATE TABLE Job (

job\_id INT PRIMARY KEY,

job\_title VARCHAR(100),

min\_sal DECIMAL(10, 2),

max\_sal DECIMAL(10, 2)

);

CREATE TABLE Department (

dept\_no INT PRIMARY KEY,

dept\_name VARCHAR(100),

location\_id INT

);

CREATE TABLE Employee\_Manager (

emp\_no INT PRIMARY KEY,

emp\_hiredate DATE, mng\_no INT,

mng\_name VARCHAR(100)

);

1. Create a report that displays the employee number, name, and salary of all employees who earn more than the average salary. Sort the results in order of ascending salary

**Query:** SELECT emp\_no, emp\_name, emp\_sal FROM Employee

WHERE emp\_sal > (SELECT AVG(emp\_sal) FROM Employee)

ORDER BY emp\_sal ASC;

1. The HR department needs a report that displays the name, department number, and job ID of all employees whose department location ID is 1700.

**Query:** SELECT e.emp\_name, d.dept\_no, e.job\_id FROM Employee e

JOIN department d ON e.dept\_no = d.dept\_no

WHERE d.location\_id = 1700;

1. Create a report for HR that displays the department number, name, and job ID for every employee in the Executive department

**Query:** SELECT d.dept\_no, d.dept\_name, e.job\_id

FROM Employee e JOIN department d ON e.dept\_no = d.dept\_no

WHERE d.dept\_name = 'Executive';

1. Create a report that displays a list of all employees whose salary is more than the salary of any employee from department 60..

**Query:** SELECT emp\_no, emp\_name, emp\_sal FROM Employee

WHERE emp\_sal > ANY (SELECT emp\_sal FROM Employee

WHERE dept\_no = 60

);

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| customer\_id | first\_name | last\_name | age | country |
| 1 | John | Doe | 31 | USA |
| 2 | Robert | Luna | 22 | USA |
| 3 | David | Robinson | 22 | UK |
| 4 | John | Reinhardt | 25 | UK |
| 5 | Betty | Doe | 28 | UAE |

Table: Customers

|  |  |  |
| --- | --- | --- |
| order\_id | amount | customer\_id |
| 1 | 200 | 4 |
| 2 | 500 | 1 |
| 3 | 300 | 3 |
| 4 | 800 | 1 |
| 5 | 150 | 2 |

Table: Orders

CREATE TABLE Customer ( Customer\_id INT PRIMARY KEY, First\_name

VARCHAR(50), Last\_name VARCHAR(50), Age INT, Country VARCHAR(50)

);

INSERT INTO Customer (Customer\_id, First\_name, Last\_name, Age, Country)

VALUES(1, 'John', 'Doe', 31,'USA');

INSERT INTO Customer (Customer\_id, First\_name, Last\_name, Age, Country)

VALUES(2, 'Robert', 'Luna', 22, 'USA');

INSERT INTO Customer (Customer\_id, First\_name, Last\_name, Age, Country)

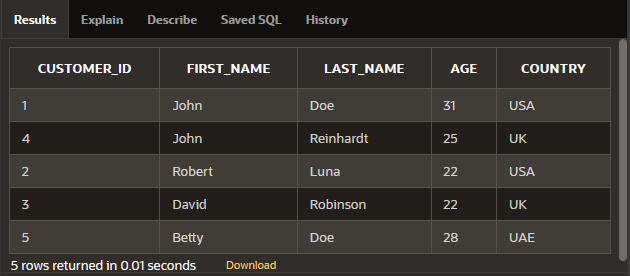
VALUES(3, 'David', 'Robinson', 22, 'UK');

INSERT INTO Customer (Customer\_id, First\_name, Last\_name, Age, Country)

VALUES(4, 'John', 'Reinhardt', 25, 'UK');

INSERT INTO Customer (Customer\_id, First\_name, Last\_name, Age, Country)

VALUES(5, 'Betty', 'Doe', 28, 'UAE');



CREATE TABLE Orders (

Order\_id INT PRIMARY KEY,

Amount INT,

Customer\_id INT,

FOREIGN KEY (Customer\_id) REFERENCES Customer(Customer\_id)

);

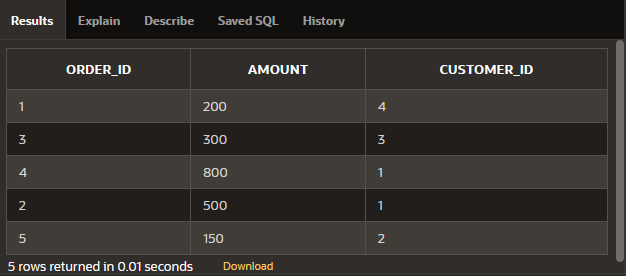
INSERT INTO Orders (Order\_id, Amount, Customer\_id) VALUES(1, 200, 4);

INSERT INTO Orders (Order\_id, Amount, Customer\_id) VALUES(2, 500, 1);

INSERT INTO Orders (Order\_id, Amount, Customer\_id) VALUES(3, 300, 3);

INSERT INTO Orders (Order\_id, Amount, Customer\_id) VALUES(4, 800, 1);

INSERT INTO Orders (Order\_id, Amount, Customer\_id) VALUES(5, 150, 2);



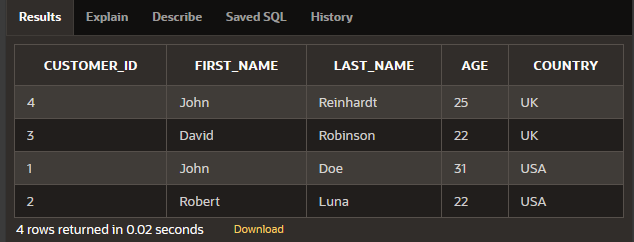
1. Find the details of customers who have placed an order.

**Query:** SELECT c.\* FROM Customer c

WHERE EXISTS (SELECT 1 FROM Orders o

WHERE o.Customer\_id = c.Customer\_id

);



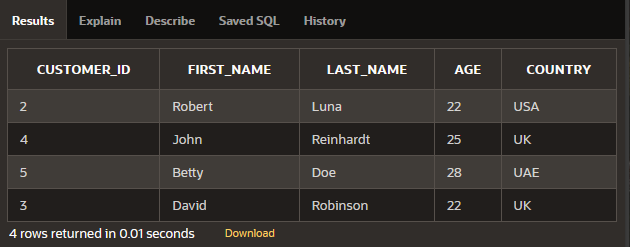
1. Find all customers who do not have order more than $500.

**Query:** SELECT c.\* FROM Customer c

WHERE NOT EXISTS (SELECT 1 FROM Orders o

WHERE o.Customer\_id = c.Customer\_id

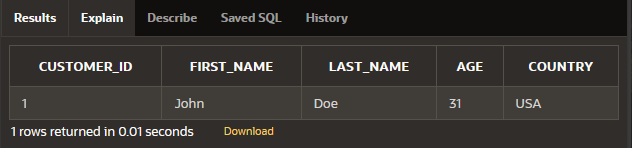
AND o.Amount > 500 );



1. Find the customers who have placed highest order.

**Query:** SELECT c.\* FROM Customer c JOIN Orders o ON c.Customer\_id = o.Customer\_id

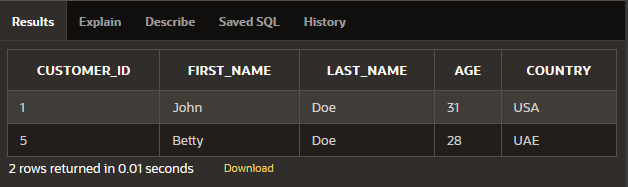
WHERE o.Amount = (SELECT MAX(Amount) FROM Orders);



1. Find all customers whose age is greater than the average age of all customers.

**Query:** SELECT \* FROM Customer

WHERE Age > (SELECT AVG(Age) FROM Customer)



1. Find the name of customers whose age is equal to age of John.

**Query:** SELECT First\_name, Last\_name FROM Customer

WHERE Age = (SELECT MAX(Age) FROM Customer

WHERE First\_name = 'John');

