

**DATABASE MANAGEMENT SYSTEMS**

**LAB ASSSIGNMENT-1**

**SUBMITTED BY- RIMJHIM MITTAL**

**ROLL NUMBER- 102103430**

**CLASS- 2CO16**

---

1. Create table Student (Rno, Name, DOB, Gender, Class, College, City, Marks)

```
CREATE TABLE Student(  
    Rno NUMBER(9) NOT NULL,  
    Stu_Name VARCHAR2(50) NOT NULL,  
    DOB DATE NOT NULL,  
    Gender VARCHAR2(10),  
    Stu_Class VARCHAR2(10),  
    College VARCHAR2(100),  
    City VARCHAR2(15),  
    Marks NUMBER(3));
```

+ Code + Text

Table created.

2. Insert 5 records in student table

```
INSERT  
INTO Student  
VALUES( 102103430,'Rimjhim Mittal','27-JUN-03','FEMALE','2CO16','TIET','Patiala',80);  
INSERT  
INTO Student  
VALUES( 102103432,'Harsh Jain','13-JUN-03','MALE','2CO16','TIET','Patiala',80);  
INSERT  
INTO Student  
VALUES( 102103447,'Shreeya Chatterji','15-JUN-03','FEMALE','2CO16','TIET','Patiala',80);  
INSERT  
INTO Student  
VALUES( 102103424,'Lakshaya Aggarwal','06-NOV-03','MALE','2CO16','TIET','Patiala',80);  
INSERT  
INTO Student  
VALUES( 102103443,'Hrsh Dhingra','14-MAR-03','MALE','2CO16','TIET','Patiala',80);
```

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

3. Display the information of all the students

```
SELECT * FROM Student;
```

RNO	STU_NAME	DOB	GENDER	STU_CLASS	COLLEGE	CITY	MARKS
102103430	Rimjhim Mittal	27-JUN-03	FEMALE	2C016	TIET	Patiala	80
102103432	Harsh Jain	13-JUN-03	MALE	2C016	TIET	Patiala	80
102103447	Shreeya Chatterji	15-JUN-03	FEMALE	2C016	TIET	Patiala	80
102103424	Lakshaya Aggarwal	06-NOV-03	MALE	2C016	TIET	Patiala	80
102103443	Hrsh Dhingra	14-MAR-03	MALE	2C016	TIET	Patiala	80

[Download CSV](#)

5 rows selected.

#### 4. Display the detail structure of student table

DESCRIBE Student;

TABLE STUDENT		
Column	Null?	Type
RNO	NOT NULL	NUMBER(9,0)
STU_NAME	NOT NULL	VARCHAR2(50)
DOB	NOT NULL	DATE
GENDER	-	VARCHAR2(10)
STU_CLASS	-	VARCHAR2(10)
COLLEGE	-	VARCHAR2(100)
CITY	-	VARCHAR2(15)
MARKS	-	NUMBER(3,0)

[Download CSV](#)

#### 5. Display Rno, Name and Class information of 'Patiala' students.

```
SELECT Rno, stu_name, stu_class
FROM STUDENT
WHERE CITY= 'Patiala';
```

RNO	STU_NAME	STU_CLASS
102103430	Rimjhim Mittal	2C016
102103432	Harsh Jain	2C016
102103447	Shreeya Chatterji	2C016
102103424	Lakshaya Aggarwal	2C016
102103443	Hrsh Dhingra	2C016

[Download CSV](#)

5 rows selected.

#### 6. Display information on ascending order of RNo

```
SELECT *
FROM Student
ORDER BY Rno;
```

RNO	STU_NAME	DOB	GENDER	STU_CLASS	COLLEGE	CITY	MARKS
102103424	Lakshaya Aggarwal	06-NOV-03	MALE	2C016	TIET	Patiala	80
102103430	Rimjhim Mittal	27-JUN-03	FEMALE	2C016	TIET	Patiala	80
102103432	Harsh Jain	13-JUN-03	MALE	2C016	TIET	Patiala	80
102103443	Hrsh Dhingra	14-MAR-03	MALE	2C016	TIET	Patiala	80
102103447	Shreeya Chatterji	15-JUN-03	FEMALE	2C016	TIET	Patiala	80

[Download CSV](#)

5 rows selected.

7. Change the marks of Rno 102103432 to 89.

```
UPDATE Student
SET Marks = 89
WHERE RNo= 102103432;
SELECT * FROM Student
WHERE RNo= 102103432;
```

RNO	STU_NAME	DOB	GENDER	STU_CLASS	COLLEGE	CITY	MARKS
102103432	Harsh Jain	13-JUN-03	MALE	2C016	TIET	Patiala	89

[Download CSV](#)

8. Change the name and city of Rno 102103424

```
UPDATE Student
SET City= 'Delhi',
Stu_Name= 'Laccha Aggarwal'
WHERE RNo= 102103424;
```

```
SELECT * FROM Student
WHERE RNo= 102103424;
```

1 row(s) updated.

RNO	STU_NAME	DOB	GENDER	STU_CLASS	COLLEGE	CITY	MARKS
102103424	Laccha Aggarwal	06-NOV-03	MALE	2C016	TIET	Delhi	80

[Download CSV](#)

9. Delete the information of 'Delhi' city records

```
DELETE FROM Student
WHERE City= 'Delhi';
SELECT * FROM Student;
```

RNO	STU_NAME	DOB	GENDER	STU_CLASS	COLLEGE	CITY	MARKS
102103430	Rimjhim Mittal	27-JUN-03	FEMALE	2C016	TIET	Patiala	80
102103432	Harsh Jain	13-JUN-03	MALE	2C016	TIET	Patiala	89
102103447	Shreeya Chatterji	15-JUN-03	FEMALE	2C016	TIET	Patiala	80
102103443	Hrsh Dhingra	14-MAR-03	MALE	2C016	TIET	Patiala	80

[Download CSV](#)

4 rows selected.

10. Delete the records of student where marks<30.

```
DELETE FROM Student
WHERE Marks>85;
```

```
SELECT * FROM Student;
```

1 row(s) deleted.

RNO	STU_NAME	DOB	GENDER	STU_CLASS	COLLEGE	CITY	MARKS
102103430	Rimjhim Mittal	27-JUN-03	FEMALE	2C016	TIET	Patiala	80
102103447	Shreeya Chatterji	15-JUN-03	FEMALE	2C016	TIET	Patiala	80
102103443	Hrsh Dhingra	14-MAR-03	MALE	2C016	TIET	Patiala	80

[Download CSV](#)

3 rows selected.

**DATABASE MANAGEMENT SYSTEMS****LAB ASSSIGNMENT-2****SUBMITTED BY- RIMJHIM MITTAL****ROLL NUMBER- 102103430****CLASS- 2C016**

Based on Emp table

Columns are EmpNo, Ename, Job, Salary, Commission, DeptNo

Insert 5 records by storing Null value in some records for commission column.

```
CREATE TABLE EMP
(EMPNO NUMBER(4) NOT NULL,
ENAME VARCHAR2(10),
EJOB VARCHAR2(9),
SAL NUMBER(7, 2),
COMM NUMBER(7, 2),
DEPTNO NUMBER(2));

INSERT INTO EMP VALUES
(7369, 'SMITH', 'CLERK', 800, NULL, 20);
INSERT INTO EMP VALUES
(7499, 'ALLEN', 'SALESMAN', 1600, 300, 30);
INSERT INTO EMP VALUES
(7521, 'ALARIC', 'SALESMAN', 1250, 500, 10);
INSERT INTO EMP VALUES
(7566, 'JONES', 'MANAGER', 2975, NULL, 20);
INSERT INTO EMP VALUES
(7654, 'MARTIN', 'SALESMAN', 1250, 1400, 30);
INSERT INTO EMP VALUES
(7698, 'RAJ', 'CLERK', 2850, NULL, 30);
INSERT INTO EMP VALUES
(7782, 'CLARK', 'MANAGER', 2450, NULL, 10);
INSERT INTO EMP VALUES
(7788, 'RAMADA', 'ANALYST', 3000, NULL, 20);

SELECT * FROM EMP;
```

EMPNO	ENAME	EJOB	SAL	COMM	DEPTNO
7369	SMITH	CLERK	800	-	20
7499	ALLEN	SALESMAN	1600	300	30
7521	ALARIC	SALESMAN	1250	500	10
7566	JONES	MANAGER	2975	-	20
7654	MARTIN	SALESMAN	1250	1400	30
7698	RAJ	CLERK	2850	-	30
7782	CLARK	MANAGER	2450	-	10
7788	RAMADA	ANALYST	3000	-	20

Q1) Get employee no and employee name who works in dept no 10 ?

```
SELECT EMPNO, ENAME
FROM EMP
WHERE DEPTNO= 10;
```

EMPNO	ENAME
7521	ALARIC
7782	CLARK

Q2) Display the employee names of those clerks whose salary is greater than 2000 ?

```
SELECT ENAME  
FROM EMP  
WHERE EJOB='CLERK'  
AND SAL>2000;
```

ENAME
RAJ

Q3) Display name and job of Salesperson & Clerks ?

```
SELECT ENAME , EJOB  
FROM EMP  
WHERE EJOB='CLERK'  
OR EJOB='SALESMAN';
```

ENAME	EJOB
SMITH	CLERK
ALLEN	SALESMAN
ALARIC	SALESMAN
MARTIN	SALESMAN
RAJ	CLERK

Q4) Display all details of employees whose salary between 2000 and 3000 ?

```
SELECT *  
FROM EMP  
WHERE SAL  
BETWEEN 2000 AND 3000;
```

EMPNO	ENAME	EJOB	SAL	COMM	DEPTNO
7566	JONES	MANAGER	2975	-	20
7698	RAJ	CLERK	2850	-	30
7782	CLARK	MANAGER	2450	-	10
7788	RAMADA	ANALYST	3000	-	20

Q5) Display all details of employees whose dept no is 10, 20, or 30 ?

```
SELECT *  
FROM EMP  
WHERE DEPTNO IN (10,20,30);
```

EMPNO	ENAME	EJOB	SAL	COMM	DEPTNO
7369	SMITH	CLERK	800	-	20
7499	ALLEN	SALESMAN	1600	300	30
7521	ALARIC	SALESMAN	1250	500	10
7566	JONES	MANAGER	2975	-	20
7654	MARTIN	SALESMAN	1250	1400	30
7698	RAJ	CLERK	2850	-	30
7782	CLARK	MANAGER	2450	-	10
7788	RAMADA	ANALYST	3000	-	20

Q6) Display name of those employees whose commission is NULL ?

```
SELECT ENAME
FROM EMP
WHERE COMM IS NULL;
```

ENAME
SMITH
JONES
RAJ
CLARK
RAMADA

Q7) Display dept no & salary in ascending order of dept no and with in each dept no salary should be in descending order ?

```
SELECT DEPTNO, SAL
FROM EMP
ORDER BY DEPTNO, SAL DESC;
```

DEPTNO	SAL
10	2450
10	1250
20	3000
20	2975
20	800
30	2850
30	1600
30	1250

Q8) Display name of employees having two 'a' or 'A' chars in the name ?

```
SELECT ENAME
FROM EMP
WHERE ENAME LIKE '%A%A%'
OR ENAME LIKE '%a%a%' ;
```

ENAME
ALARIC
RAMADA

Q9) Display the name of the employees whose second char is 'I' or 'L' ?

```
SELECT ENAME
FROM EMP
WHERE ENAME LIKE '_L%'
OR ENAME LIKE '_l%' ;
```

ENAME
ALLEN
ALARIC
CLARK

Q10) Display the name of the employees whose first or last char is 'a' or 'A' ?

```
SELECT ENAME
FROM EMP
WHERE ENAME LIKE 'a%'
OR ENAME LIKE 'A%'
OR ENAME LIKE '%a'
OR ENAME LIKE '%a%' ;
```

ENAME
ALLEN
ALARIC
RAMADA

Q11) Display maximum, minimum, average salary of deptno 10 employees.

```
SELECT AVG(SAL), MAX(SAL), MIN(SAL)
FROM emp
WHERE DEPTNO = 10;
```

Avg(Sal)	Max(Sal)	Min(Sal)
1850	2450	1250

Q12) Display total number of employees working in deptno 20

```
SELECT COUNT(*)
FROM EMP
WHERE DEPTNO= 20;
```

```
COUNT(*)
```

Q13) Display total salary paid to clerks

```
SELECT SUM(SAL)
FROM EMP
WHERE EJOB= 'CLERK';
```

SUM(SAL)
3650

Q14) Display system date

```
SELECT TO_CHAR(SYSDATE, 'MM-DD-YYYY HH24:MI:SS') "NOW"
FROM DUAL;
```

NOW
01-24-2023 18:02:26

Q15) Display the result of  $(12*12)/13$

```
SELECT (12*12)/13
FROM DUAL;
```

(12*12)/13
11.07692307692307692307692307692308

Q16) Display info of 'raj' irrespective to the case in which the data is stored.

```
SELECT *
FROM EMP
WHERE UPPER(ENAME) ='RAJ';
```

EMPNO	ENAME	EJOB	SAL	COMM	DEPTNO
7698	RAJ	CLERK	2850	-	30

# **DATABASE MANAGEMENT SYSTEMS**

## **LAB ASSIGNMENT-3**

**SUBMITTED BY- RIMJHIM MITTAL**

**ROLL NUMBER- 102103430**

**CLASS- 2C016**

---

**Q1) Use the following functions**

**1. chr (n):**

```
SELECT CHR(82) || CHR(73) || CHR(77) || CHR(74) || CHR(72) || CHR(73) || CHR(77) FROM
```

CHR(82)    CHR(73)    CHR(77)    CHR(74)    CHR(72)    CHR(73)    CHR(77)
RIMJHIM

**2. concat(char1,char2):**

```
SELECT CONCAT(CONCAT('Hello', ' '), 'World') FROM DUAL;
```

CONCAT(CONCAT('HELLO', ''), 'WORLD')
Hello World

**3. instr(string,char):**

```
SELECT INSTR('RIMJHIM', 'I', 2) FROM DUAL;
```

INSTR('RIMJHIM','I',2)
2

**4. length(n):**

```
SELECT LENGTH('PL/SQL') FROM DUAL;
```

LENGTH('PL/SQL')
6

#### 5. **lpad(char1 ,n [,char2]):**

```
SELECT LPAD('MITTAL', 14, 'RIMJHIM ') from dual;
```

LPAD('MITTAL',14,'RIMJHIM')
RIMJHIM MITTAL

#### 6. **ltrim(string [,char(s)]):**

```
SELECT LTRIM('HELLO', 'HE') from dual;
```

LTRIM('HELLO','HE')
LL0

#### 7. **rpad(char1 ,n [,char2]):**

```
SELECT RPAD('MITTAL', 14, 'RIMJHIM ') from dual;
```

RPAD('MITTAL',14,'RIMJHIM')
MITTALRIMJHIM

#### 8. **rtrim(string [,char(s)]):**

```
SELECT RTRIM('HELLO', 'OL') from dual;
```

```
RTRIM('HELLO','OL')
```

```
HE
```

#### 9. **replace(char ,search\_string ,replacement\_string):**

```
SELECT replace('I AM A GOOD GIRL' , 'GOOD' , 'BAD') from dual;
```

```
REPLACE('IAMAGOODGIRL','GOOD','BAD')
```

```
I AM A BAD GIRL
```

#### 10. **substr(string ,position ,substring length):**

```
SELECT SUBSTR('ABCDEFG',-5,4) "Substring" FROM DUAL;
```

```
Substring
```

```
CDEF
```

#### 11. **initcap(char):**

```
SELECT INITCAP('the HELLO soap') "Capitals"  
FROM DUAL;
```

```
Capitals
```

```
The Hello Soap
```

#### 12. **lower(string):**

```
SELECT LOWER('LOWERME')  
FROM DUAL;
```

LOWER( 'LOWERME' )
lowerme

### 13. **upper(string):**

```
SELECT UPPER( 'uPPerCase' ) "Name"  
FROM DUAL;
```

Name
UPPERCASE

### 14. **translate(char ,from string ,to string):**

```
SELECT TRANSLATE( 'FATHER' , 'FA' , 'MO' ) FROM DUAL;
```

TRANSLATE( 'FATHER' , 'FA' , 'MO' )
MOTHER

### 15. **abs(n):**

```
SELECT ABS(-28) "Absolute" FROM DUAL;
```

Absolute
28

### 16. **ceil(n):**

```
SELECT CEIL(3.545) FROM DUAL;
```

CEIL(3.545)
4

### 17. **cos(n):**

```
SELECT FLOOR(COS(3.14159265359)) "Cosine of 180 degrees" FROM DUAL;
```

Cosine of 180 degrees
-1

### 18. **exp(n):**

```
SELECT ROUND(EXP(3),3) "e to the 3RD power" FROM DUAL;
```

e to the 3RD power
20.086

### 19. **floor(n):**

```
SELECT FLOOR(5.34) "FLOOR" FROM DUAL;
```

FLOOR
5

### 20. **mod(m ,n):**

```
SELECT MOD(38,5) "MOD" FROM DUAL;
```

MOD
3

## 21. **power(x ,y):**

```
SELECT power(5 ,3) "POWER" FROM DUAL;
```

POWER
125

## 22. **round(x [y]):**

```
SELECT round(5.67483648, 2) "ROUND" FROM DUAL;
```

ROUND
5.67

## 23. **sign(n):**

```
SELECT SIGN(-15) FROM DUAL;  
SELECT SIGN(8) FROM DUAL;  
SELECT SIGN(-0) FROM DUAL;
```

SIGN(-15)
-1

[Download CSV](#)

SIGN(8)
1

[Download CSV](#)

SIGN(-0)
0

#### 24. **sqrt(n);**

```
SELECT Sqrt(256) FROM DUAL;
```

SQRT(256)
16

#### 25. **trunc(x ,n):**

```
SELECT trunc(25465.57895, 2), trunc(25465.57895, -1), trunc(25465.57895, 0) FROM D
```

TRUNC(25465.57895,2)	TRUNC(225465.57895,-1)	TRUNC(25465.57895,0)
25465.57	225460	25465

#### 26. **sysdate:**

```
SELECT TO_CHAR(SYSDATE, 'MM-DD-YYYY HH24:MI:SS') "NOW" FROM DUAL;
```

NOW
01-25-2023 15:56:11

## 27. **add\_months(d ,n):**

```
SELECT ADD_MONTHS('27-JAN-03', 5)  
FROM DUAL;
```

ADD_MONTHS('27-JAN-03',5)
27-JUN-03

## 28. **last\_day():**

```
SELECT last_day('20-FEB-2013')  
FROM DUAL;
```

LAST_DAY('20-FEB-2013')
28-FEB-13

## 29. **months\_between(date1 ,date2):**

```
SELECT months_between('15-AUG-1947', '30-JAN-1949')  
FROM DUAL;
```

MONTHS_BETWEEN('15-AUG-1947','30-JAN-1949')
-17.48387096774193548387096774193548387097

## 30. **next\_day(date ,char):**

```

SELECT next_day('28-FEB-12', 2), NEXT_DAY('24-JUN-22', 'MONDAY')
FROM DUAL;
--This assumes 2 to be monday

```

NEXT_DAY('28-FEB-12',2)	NEXT_DAY('24-JUN-22','MONDAY')
05-MAR-12	27-JUN-22

### 31. greatest(expr):

```

SELECT GREATEST('HARRY', 'HARRIOT', 'HAROLD') "Greatest",
GREATEST(1, '3.925', '2.4') "Greatest"
FROM DUAL;

```

-If the first expr is numeric, then Oracle determines the argument with the highest numeric precedence, implicitly converts the remaining arguments to that data type before the comparison, and returns that data type. If the first expr is not numeric, then each expr after the first is implicitly converted to the data type of the first expr before the comparison. Character comparison is based on the numerical codes of the characters in the database character set and is performed on whole strings treated as one sequence of bytes, rather than character by character.

Greatest	Greatest
HARRY	3.925

### 32.least(expr):

```

SELECT LEAST('HARRY', 'HARRIOT', 'HAROLD') "Least",
LEAST(1, '3.925', '2.4') "LEAST"
FROM DUAL;

```

Least	LEAST
HAROLD	1

Q2) Display current time in hour : min : sec format

```
select to_char(sysdate, 'HH24:MI:SS') as "Current Time"
from dual;
```

Current Time
06:31:55

Q3) Display salary + commission of emp table

```
SELECT ENAME, COMM, SAL, (SAL+NVL(COMM, 0)) FROM EMP;
```

ENAME	COMM	SAL	(SAL+NVL(COMM,0))
ALARIC	500	1250	1750
JONES	-	2975	2975
MARTIN	1400	1250	2650
RAJ	-	2850	2850
CLARK	-	2450	2450
RAMADA	-	3000	3000
SMITH	-	800	800
ALLEN	300	1600	1900

Q4) Store any date value in hiredate column of table ?

```
ALTER TABLE EMP
ADD hire_date DATE;
UPDATE EMP
SET hire_date='21-feb-16'
WHERE ENAME = 'ALARIC';
UPDATE EMP
SET hire_date='01-mar-85'
WHERE ENAME = 'JONES';
UPDATE EMP
SET hire_date='05-DEC-20'
WHERE ENAME = 'MARTIN';
UPDATE EMP
SET hire_date='30-JAN-00'
```

```

WHERE ENAME = 'RAJ';
UPDATE EMP
SET hire_date='05-DEC-85'
WHERE ENAME = 'CLARK';
UPDATE EMP
SET hire_date='05-JAN-23'
WHERE ENAME = 'SMITH';

SELECT ENAME, hire_date from EMP;

```

ENAME	HIRE_DATE
ALARIC	21-FEB-16
JONES	01-MAR-85
MARTIN	05-DEC-20
RAJ	30-JAN-00
CLARK	05-DEC-85
RAMADA	-
SMITH	05-JAN-23
ALLEN	-

Q5) Display name of employee(s) who join the company in 1985 ?

```
SELECT ENAME, HIRE_DATE FROM EMP WHERE EXTRACT(YEAR FROM HIRE_DATE)=1985;
```

ENAME	HIRE_DATE
JONES	01-MAR-85
CLARK	05-DEC-85

Q6) Display name of the employee(s) who join the company this year ?

```
SELECT ENAME, HIRE_DATE FROM EMP WHERE EXTRACT(YEAR FROM HIRE_DATE)=EXTRACT(YEAR FR
```

ENAME	HIRE_DATE
SMITH	05-JAN-23





# Lab Assignment-4

1. Create table emp which has the following attributes (employee table)  
(@empno, ename, job, sal, deptno)

Where empno is primary key, ename is unique, job in (Prof, AP, and Lect), sal is not NULL, and deptno default is 10.

The screenshot shows the Oracle Live SQL interface. The SQL Worksheet pane contains the following code:

```
1 create table Emp(
2     EmpNo number PRIMARY KEY,
3     Ename varchar2(20) UNIQUE ,
4     job varchar2(20) CHECK(job in ('Prof', 'AP', 'Lect')),
5     salary number NOT NULL,
6     depNo number DEFAULT 10
7 );
8
9
10
11
12
```

Below the code, the message "Table created." is displayed. The interface includes standard navigation and toolbar buttons.

## Insert appropriate records

The screenshot shows the Oracle Live SQL interface. The SQL Worksheet pane contains the following code:

```
4     Ename varchar2(20) UNIQUE ,
5     job varchar2(20) CHECK(job in ('Prof', 'AP', 'Lect')),
6     salary number NOT NULL,
7     depNo number DEFAULT 10
8 );
9 insert into emp values(1,'Maya','Prof',25000,default);
10 insert into emp values(14,'Rahul','AP',36000,default);
11 insert into emp values(38,'Aman','Lect',20000,30);
12 insert into emp values(24,'Anika','AP',35000,default);
13 insert into emp values(56,'Zoya','Prof',17000,20);
14 SELECT *from emp;
15
```

Below the code, a data grid displays the inserted records:

EMPNO	ENAME	JOB	SALARY	DEPNO
1	Maya	Prof	25000	10
14	Rahul	AP	36000	10
38	Aman	Lect	20000	30
24	Anika	AP	35000	10
56	Zoya	Prof	17000	20

The interface includes standard navigation and toolbar buttons.

Check error messages in case of violation

The screenshot shows a SQL Worksheet in the Oracle Live SQL interface. The code attempts to insert multiple rows into the 'emp' table, but fails due to unique constraint violations. The errors are:

```
ORA-00001: unique constraint (SQL_QETGZBUFIEBXSXHZHSLYISHDF.SYS_C00112954996) violated ORA-06512: at "SYS.DBMS_SQL", line 1721  
ORA-00001: unique constraint (SQL_QETGZBUFIEBXSXHZHSLYISHDF.SYS_C00112954996) violated ORA-06512: at "SYS.DBMS_SQL", line 1721  
ORA-00001: unique constraint (SQL_QETGZBUFIEBXSXHZHSLYISHDF.SYS_C00112954997) violated ORA-06512: at "SYS.DBMS_SQL", line 1721  
ORA-01400: cannot insert NULL into ("SQL_QETGZBUFIEBXSXHZHSLYISHDF"."EMP"."SALARY") ORA-06512: at "SYS.DBMS_SQL", line 1721
```

Below the worksheet, a footer bar displays: "2023 Oracle - Live SQL 23.1.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 - Database Documentation - Ask Tom - Dev Gym" and "Built with ❤ using Oracle APEX - Privacy - Terms of Use".

and list all the constraint names for given table.

The screenshot shows a SQL Worksheet in the Oracle Live SQL interface. The code lists all constraints for the 'EMP' table using the 'user\_constraints' view:

```
16  
17  insert into emp values(1,'Arun','Lect',34000,default);  
18  insert into emp values(1,'Arun','Lect',34000,default);  
19  insert into emp values(21,'Aman','AP',35100,20);  
20  insert into emp values(22,'Rachit','AP',NULL,default);  
21  
22  
23  select constraint_name,constraint_type from user_constraints where table_name='EMP';  
24  
25  
26
```

Below the worksheet, a table displays the results of the query:

CONSTRAINT_NAME	CONSTRAINT_TYPE
SYS_C00112954994	C
SYS_C00112954995	C
SYS_C00112954996	P
SYS_C00112954997	U

At the bottom of the page, there is a "Download CSV" button and a footer bar with the same information as the previous screenshot.

2. Create table book:

Rno number-PK

DOI-date

GOR-date

GOR>DOI

Insert appropriate records, check error messages in case of violation and list all the constraint names for given table.

Create table book:

The screenshot shows the Oracle Live SQL interface. The title bar says "Live SQL". The main area is a "SQL Worksheet" with the following code:

```
1 CREATE TABLE BOOK (
2   RNO NUMBER PRIMARY KEY,
3   DOI DATE,
4   DOR DATE ,CHECK(DOR>DOI)
5 );
```

Below the code, a message "Table created." is displayed. The interface includes standard buttons for Feedback, Help, Save, and Run.

Table created.

Insert appropriate records:

The screenshot shows the Oracle Live SQL interface. At the top, there's a toolbar with 'Live SQL' and other navigation options. Below it is a 'SQL Worksheet' area containing the following SQL code:

```
3   DOI DATE,  
4   DOR DATE ,CHECK(DOR>DOI)  
5 );  
6  
7 INSERT INTO BOOK VALUES(1,'01-JAN-2022','15-JAN-2022');  
8 INSERT INTO BOOK VALUES(2,'01-FEB-2022','15-FEB-2022');  
9 INSERT INTO BOOK VALUES(3,'01-MAR-2022','15-MAR-2022');  
10 INSERT INTO BOOK VALUES(4,'01-JUN-2022','15-DEC-2022');  
11 SELECT *FROM BOOK;  
12  
13
```

Below the code, a table titled 'BOOK' is displayed with the following data:

RNO	DOI	DOR
1	01-JAN-22	15-JAN-22
2	01-FEB-22	15-FEB-22
3	01-MAR-22	15-MAR-22
4	01-JUN-22	15-DEC-22

At the bottom of the worksheet, there's a 'Download CSV' button and a footer with copyright information.

Check error messages in case of violation :

The screenshot shows the Oracle Live SQL interface. At the top, there's a toolbar with 'Live SQL' and other navigation options. Below it is a 'SQL Worksheet' area containing the following SQL code:

```
6  
7 INSERT INTO BOOK VALUES(1,'01-JAN-2022','15-JAN-2022');  
8 INSERT INTO BOOK VALUES(2,'01-FEB-2022','15-FEB-2022');  
9 INSERT INTO BOOK VALUES(3,'01-MAR-2022','15-MAR-2022');  
10 INSERT INTO BOOK VALUES(4,'01-JUN-2022','15-DEC-2022');  
11 SELECT *FROM BOOK;  
12  
13 INSERT INTO BOOK VALUES(5,'01-MAY-2022','15-MAR-2022');  
14 INSERT INTO BOOK VALUES(1,'01-MAY-2022','15-SEPT-2022');  
15 INSERT INTO BOOK VALUES(1,'01-MAY-2022','15-SEPT-2022');  
16  
17
```

Below the code, error messages are displayed:

```
ORA-02290: check constraint (SQL_QETGZBUFIEBXSXHZHSLYISHDF.SYS_C00112955284) violated ORA-06512: at "SYS.DBMS_SQL", line 1721  
ORA-01861: literal does not match format string ORA-06512: at "SYS.DBMS_SQL", line 1721  
ORA-01861: literal does not match format string ORA-06512: at "SYS.DBMS_SQL", line 1721
```

At the bottom of the worksheet, there's a footer with copyright information.

List all the constraint names for given table:

The screenshot shows the Oracle Live SQL interface. At the top, there's a header with 'Live SQL' and user information. Below it is a 'SQL Worksheet' tab. The code entered is:

```
12
13  INSERT INTO BOOK VALUES(5,'01-MAY-2022','15-MAR-2022');
14  INSERT INTO BOOK VALUES(1,'01-MAY-2022','15-SEPT-2022');
15  INSERT INTO BOOK VALUES(1,'01-MAY-2022','15-SEPT-2022');
16
17
18
19  SELECT CONSTRAINT_NAME,CONSTRAINT_TYPE FROM USER_CONSTRAINTS WHERE TABLE_NAME='BOOK';
20
```

Below the code, the results are displayed in a table:

CONSTRAINT_NAME	CONSTRAINT_TYPE
SYS_C00112955284	C
SYS_C00112955285	P

A 'Download CSV' button is available below the table. A message at the bottom indicates '2 rows selected.'

At the bottom of the interface, there's a footer with copyright information.

3. Create table st Rno-Number ,Class-  
Char Marks-Number  
P r i m a r y  
key(rno,class) Marks>0  
Insert appropriate records, check error messages in  
case of violation and list all the constraint names for  
given table.

Create table:

The screenshot shows the Oracle Live SQL interface. In the top navigation bar, there are icons for Feedback, Help, and a user account (ntank\_be21@thapar.edu). Below the navigation bar is a toolbar with buttons for Clear, Find, Actions, Save, and Run. The main area is titled "SQL Worksheet". The code entered is:

```
1 CREATE TABLE ST(
2     RNO NUMBER,
3     CLASS VARCHAR2(20),
4     MARKS NUMBER CHECK(MARKS>0),
5     PRIMARY KEY(RNO,CLASS)
6 );
7
8
9
10
11
12
```

Below the code, the message "Table created." is displayed. At the bottom of the worksheet, there is a footer bar with the text "2023 Oracle - Live SQL 23.1.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 - Database Documentation - Ask Tom - Dev Gym" and "Built with ❤ using Oracle APEX - Privacy - Terms of Use".

Insert appropriate records:

The screenshot shows the Oracle Live SQL interface. The SQL Worksheet contains the following code:

```
4     MARKS NUMBER CHECK(MARKS>0),
5     PRIMARY KEY(RNO,CLASS)
6 );
7
8
9 INSERT INTO ST VALUES (1, 'Class 1', 80);
10 INSERT INTO ST VALUES(2, 'Class 1', 90);
11 INSERT INTO ST VALUES(3, 'Class 2', 85);
12 SELECT *FROM ST;
13
14
```

After executing the code, the message "1 row(s) inserted." is shown twice. Below the code, a table is displayed with the data:

RNO	CLASS	MARKS
1	Class 1	80
2	Class 1	90
3	Class 2	85

At the bottom of the worksheet, there is a footer bar with the text "2023 Oracle - Live SQL 23.1.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 - Database Documentation - Ask Tom - Dev Gym" and "Built with ❤ using Oracle APEX - Privacy - Terms of Use".

Check error messages in case of violation:

Live SQL

SQL Worksheet

```

11  INSERT INTO ST VALUES(3, 'Class 2', 85);
12  SELECT *FROM ST;
13
14
15
16  INSERT INTO ST VALUES (1, 'Class 1', 80);
17  INSERT INTO ST VALUES(4,'Class 2',-1);
18
19
20
21
22
23

```

ORA-00001: unique constraint (SQL\_VNTBSONBYZPONITXQWEJKMGZM.SYS\_C00112965917) violated ORA-06512: at "SYS.DBMS\_SQL", line 1721

ORA-02290: check constraint (SQL\_VNTBSONBYZPONITXQWEJKMGZM.SYS\_C00112965916) violated ORA-06512: at "SYS.DBMS\_SQL", line 1721

2023 Oracle - Live SQL 23.1.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 - Database Documentation · Ask Tom · Dev Gym  
Built with ❤ using Oracle APEX · Privacy · Terms of Use

List all the constraint names for given table:

Live SQL

SQL Worksheet

```

11  INSERT INTO ST VALUES(3, 'Class 2', 85);
12  SELECT *FROM ST;
13
14
15
16  INSERT INTO ST VALUES (1, 'Class 1', 80);
17  INSERT INTO ST VALUES(4,'Class 2',-1);
18
19  SELECT CONSTRAINT_NAME,CONSTRAINT_TYPE FROM USER_CONSTRAINTS WHERE TABLE_NAME='ST';
20
21
22
23

```

CONSTRAINT_NAME	CONSTRAINT_TYPE
SYS_C00112965916	C
SYS_C00112965917	P

[Download CSV](#)

2023 Oracle - Live SQL 23.1.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 - Database Documentation · Ask Tom · Dev Gym  
Built with ❤ using Oracle APEX · Privacy · Terms of Use

4. Create table S which has the following attributes (Salesperson table)

*(sno, sname, city)*

Where sno is primary

key Create table:

The screenshot shows the Oracle Live SQL interface. At the top, there are tabs for 'Live SQL' and 'Feedback'. Below the tabs, there are buttons for 'Clear', 'Find', 'Actions', 'Save', and 'Run'. The main area is titled 'SQL Worksheet'. The code entered is:

```
1 CREATE TABLE S
2 (
3     SNO NUMBER PRIMARY KEY,
4     SNAME VARCHAR2(20),
5     CITY VARCHAR2(20)
6 );
7
8
9
10
11
12
13
```

Below the code, the message 'Table created.' is displayed. At the bottom of the interface, there is a footer bar with the text '2023 Oracle - Live SQL 23.1.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 - Database Documentation - Ask Tom - Dev Gym' and 'Built with ❤ using Oracle APEX - Privacy - Terms of Use'.

Insert appropriate records:

The screenshot shows the Oracle Live SQL interface. At the top, there are tabs for 'Live SQL' and 'Feedback'. Below the tabs, there are buttons for 'Clear', 'Find', 'Actions', 'Save', and 'Run'. The main area is titled 'SQL Worksheet'. The code entered is:

```
4     SNAME VARCHAR2(20),
5     CITY VARCHAR2(20)
6 );
7
8 INSERT INTO S VALUES(1, 'RAHUL', 'PATIALA');
9 INSERT INTO S VALUES(2, 'RASHMI', 'JAMMU');
10 INSERT INTO S VALUES(3, 'NAMAN', 'JAIPUR');
11 INSERT INTO S VALUES(4, 'KAVYA', 'MUMBAI');
12 SELECT *FROM S;
13
14
15
```

Below the code, a table is displayed with the following data:

SNO	SNAME	CITY
1	RAHUL	PATIALA
2	RASHMI	JAMMU
3	NAMAN	JAIPUR
4	KAVYA	MUMBAI

At the bottom of the interface, there is a footer bar with the text '2023 Oracle - Live SQL 23.1.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 - Database Documentation - Ask Tom - Dev Gym' and 'Built with ❤ using Oracle APEX - Privacy - Terms of Use'.

Check error messages in case of violation:

Live SQL

Feedback Help ntank\_be21@thapar.edu

SQL Worksheet

```

6 );
7
8 INSERT INTO S VALUES(1, 'RAHUL', 'PATIALA');
9 INSERT INTO S VALUES(2, 'RASHMI', 'JAMMU');
10 INSERT INTO S VALUES(3, 'NAMAN', 'JAIPUR');
11 INSERT INTO S VALUES(4, 'KAVYA', 'MUMBAI');
12 SELECT *FROM S;
13
14 INSERT INTO S VALUES(1, 'PRABH', 'PATNA');
15 INSERT INTO S VALUES(3, 'HARDIK', 'BANGALORE');
16
17
18

```

ORA-00001: unique constraint (SQL\_VNTBSONBYZPDWITXQWEJKMGZM.SYS\_C00112966924) violated ORA-06512: at "SYS.DBMS\_SQL", line 1721

ORA-00001: unique constraint (SQL\_VNTBSONBYZPDWITXQWEJKMGZM.SYS\_C00112966924) violated ORA-06512: at "SYS.DBMS\_SQL", line 1721

2023 Oracle - Live SQL 23.1.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 · Database Documentation · Ask Tom · Dev Gym  
Built with ❤ using Oracle APEX · Privacy · Terms of Use

List all the constraint names for given table:

Live SQL

Feedback Help ntank\_be21@thapar.edu

SQL Worksheet

```

12 SELECT *FROM S;
13
14 INSERT INTO S VALUES(1, 'PRABH', 'PATNA');
15 INSERT INTO S VALUES(3, 'HARDIK', 'BANGALORE');
16
17
18 SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE FROM USER_CONSTRAINTS WHERE TABLE_NAME='S';
19
20
21
22
23

```

CONSTRAINT_NAME	CONSTRAINT_TYPE
SYS_C00112966924	P

[Download CSV](#)

2023 Oracle - Live SQL 23.1.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 · Database Documentation · Ask Tom · Dev Gym  
Built with ❤ using Oracle APEX · Privacy · Terms of Use

5. Create table P which has the following attributes  
(Part table)

(pno, pname, color) Where pno is primary

key Create table:

The screenshot shows the Oracle Live SQL interface. In the SQL Worksheet, the following SQL code is written:

```
1 CREATE TABLE S
2 (
3     SNO NUMBER PRIMARY KEY,
4     SNAME VARCHAR2(20),
5     CITY VARCHAR2(20)
6 );
7
8
9
10
11
12
13
```

After running the code, the message "Table created." is displayed below the worksheet.

At the bottom of the interface, there is a footer bar with the text: "2023 Oracle - Live SQL 23.1.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 - Database Documentation - Ask Tom - Dev Gym" and "Built with ❤ using Oracle APEX - Privacy - Terms of Use".

Insert appropriate records:

The screenshot shows the Oracle Live SQL interface. In the SQL Worksheet, the following SQL code is written:

```
2 PNO NUMBER PRIMARY KEY,
3 PNAME VARCHAR2(20),
4 COLOR VARCHAR2(10)
5 );
6
7 INSERT INTO P VALUES(1,'BEN','RED');
8 INSERT INTO P VALUES(2,'HELEN','BLACK');
9 INSERT INTO P VALUES(3,'SAM','BLUE');
10 INSERT INTO P VALUES(4,'ANGEL','GREEN');
11 SELECT *FROM P;
12
```

Below the worksheet, a table displays the inserted data:

PNO	PNAME	COLOR
1	BEN	RED
2	HELEN	BLACK
3	SAM	BLUE
4	ANGEL	GREEN

At the bottom of the interface, there is a "Download CSV" button and a footer bar with the text: "2023 Oracle - Live SQL 23.1.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 - Database Documentation - Ask Tom - Dev Gym" and "Built with ❤ using Oracle APEX - Privacy - Terms of Use".

Check error messages in case of violation:

Live SQL

SQL Worksheet

```

5  /;
6
7  INSERT INTO P VALUES(1,'BEN','RED');
8  INSERT INTO P VALUES(2,'HELEN','BLACK');
9  INSERT INTO P VALUES(3,'SAM','BLUE');
10 INSERT INTO P VALUES(4,'ANGEL','GREEN');
11 SELECT *FROM P;
12
13 INSERT INTO P VALUES(2,'HARRY','OFFWHITE');
14 INSERT INTO P VALUES(4,'ZYAN','SILVER');
15
16

```

ORA-00001: unique constraint (SQL\_WORRGBMFYRFZUPPKSGCXGBBHJ.SYS\_C00112969283) violated ORA-06512: at "SYS.DBMS\_SQL", line 1721

ORA-00001: unique constraint (SQL\_WORRGBMFYRFZUPPKSGCXGBBHJ.SYS\_C00112969283) violated ORA-06512: at "SYS.DBMS\_SQL", line 1721

2023 Oracle - Live SQL 23.1.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 · Database Documentation · Ask Tom · Dev Gym  
Built with ❤ using Oracle APEX · Privacy · Terms of Use

List all the constraint names for given table:

Live SQL

SQL Worksheet

```

8  INSERT INTO P VALUES(2, HELEN , BLACK );
9  INSERT INTO P VALUES(3, 'SAM', 'BLUE');
10 INSERT INTO P VALUES(4, 'ANGEL', 'GREEN');
11 SELECT *FROM P;
12
13 INSERT INTO P VALUES(2,'HARRY','OFFWHITE');
14 INSERT INTO P VALUES(4,'ZYAN','SILVER');
15
16
17 SELECT CONSTRAINT_NAME,CONSTRAINT_TYPE FROM USER_CONSTRAINTS WHERE TABLE_NAME='P';

```

CONSTRAINT_NAME	CONSTRAINT_TYPE
SYS_C00112969283	P

[Download CSV](#)

2023 Oracle - Live SQL 23.1.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 · Database Documentation · Ask Tom · Dev Gym  
Built with ❤ using Oracle APEX · Privacy · Terms of Use

Main Navigation

6. Create table SP which has the following attributes (sno, pno qty)

Where combination of (sno, pno) is primary key, also sno and pno are foreign keys

Create table:



The screenshot shows the Oracle Live SQL interface. In the top right corner, there are buttons for Feedback, Help, and a user account (ntank\_be21@thapar.edu). Below the header, there are buttons for Clear, Find, Actions, Save, and Run. The main area is a SQL Worksheet titled "SQL Worksheet". The code entered is:

```
25
26
27 CREATE TABLE SP(
28     SNO INT,PNO INT,QTY INT,
29     PRIMARY KEY(SNO,PNO),
30     FOREIGN KEY(SNO) REFERENCES S(SNO),
31     FOREIGN KEY(PNO) REFERENCES P(PNO)
32 );
33
34
```

After running the code, the message "Table created." is displayed below the worksheet.

Table created.

Insert appropriate records:

List all the constraint names for given table:

Live SQL

SQL Worksheet

```
28   SNO INT,PNO INT,QTY INT,
29   PRIMARY KEY(SNO,PNO),
30   FOREIGN KEY(SNO) REFERENCES S(SNO),
31   FOREIGN KEY(PNO) REFERENCES P(PNO)
32 );
33
34 INSERT INTO SP VALUES(1,4,50);
35 INSERT INTO SP VALUES(2,3,42);
36 SELECT *FROM SP;
37
```

1 row(s) inserted.

1 row(s) inserted.

SNO	PNO	QTY
1	4	50
2	3	42

2023 Oracle - Live SQL 23.1.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 · Database Documentation · Ask Tom · Dev Gym  
Built with ❤ using Oracle APEX · Privacy · Terms of Use

Live SQL

SQL Worksheet

```
31   FOREIGN KEY(PNO) REFERENCES P(PNO)
32 );
33
34 INSERT INTO SP VALUES(1,4,50);
35 INSERT INTO SP VALUES(2,3,42);
36 SELECT *FROM SP;
37
38 SELECT CONSTRAINT_NAME,CONSTRAINT_TYPE,TABLE_NAME FROM USER_CONSTRAINTS WHERE TABLE_NAME='SP';
39
40
```

CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME
SYS_C00113414068	P	SP
SYS_C00113414069	R	SP
SYS_C00113414070	R	SP

Download CSV

2023 Oracle - Live SQL 23.1.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 · Database Documentation · Ask Tom · Dev Gym  
Built with ❤ using Oracle APEX · Privacy · Terms of Use

7.Create table dept which has the following attributes (department table)

*(deptno, dname)*

Where deptno is primary key, dname in (Acc, comp,

elect) Create table:

Live SQL

SQL Worksheet

```
1 CREATE TABLE DEPT(
2   DEPTNO INT,
3   DNAME VARCHAR2(20),
4   PRIMARY KEY(DEPTNO),
5   CHECK(DNAME IN('ACC','COMP','ELECT'))
6 );
7
8
9
10
```

Table created.

2023 Oracle - Live SQL 23.1.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 - Database Documentation - Ask Tom - Dev Gym  
Built with ❤ using Oracle APEX - Privacy - Terms of Use

Insert appropriate records:

Live SQL

SQL Worksheet

```
2   DEPTNO INT,
3   DNAME VARCHAR2(20),
4   PRIMARY KEY(DEPTNO),
5   CHECK(DNAME IN('ACC','COMP','ELECT'))
6 );
7
8   INSERT INTO DEPT VALUES(1,'ACC');
9   INSERT INTO DEPT VALUES(2,'COMP');
10  INSERT INTO DEPT VALUES(3,'ELECT');
11  SELECT *FROM DEPT;
```

1 row(s) inserted.

1 row(s) inserted.

DEPTNO	DNAME
1	ACC
2	COMP
3	ELECT

2023 Oracle - Live SQL 23.1.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 - Database Documentation - Ask Tom - Dev Gym  
Built with ❤ using Oracle APEX - Privacy - Terms of Use

8. Create table emp which has the following attributes (employee table)  
(@empno, ename, job, sal, deptno)

Where empno is primary key, ename is unique, job in (Prof, AP, and Lect), sal is not NULL, and deptno is foreign key

Live SQL

SQL Worksheet

```
13 CREATE TABLE EMP(
14   EMPNO INT, ENAME VARCHAR(20), JOB VARCHAR2(20), SALARY INT NOT NULL, DEPTNO INT,
15   PRIMARY KEY(EMPNO),  
16   UNIQUE (ENAME),  
17   CHECK (JOB IN('PROF,AP,LECT')),  
18   FOREIGN KEY (DEPTNO) REFERENCES DEPT(DEPTNO)
19 );
20
```

Table created.

2023 Oracle - Live SQL 23.1.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 · Database Documentation · Ask Tom · Dev Gym  
Built with ❤ using Oracle APEX · Privacy · Terms of Use

# LAB ASSIGNMENT-5

1. Create the following tables and insert some tuples in these tables shown below.

Where sid is the primary key for the Sailors table, bid is the primary key for the Boats table and sid and bid are the foreign keys for the Reserves table referencing to the Sailors and Boats table, respectively.

Sailors(sid: integer, sname: string, rating: integer, age: real) Boats(bid: integer, bname: string, color: string) Reserves(sid: integer, bid: integer, day: date)

After inserting the records in these tables, the instances should look like as follows:

**Sailors**

<i>sid</i>	<i>sname</i>	<i>rating</i>	<i>age</i>
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

**Reserves**

<i>sid</i>	<i>bid</i>	<i>day</i>
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

**Boats**

<i>bid</i>	<i>bname</i>	<i>color</i>
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

>> Sailors(sid: integer, sname: string, rating: integer, age: real)

>> Boats(bid: integer, bname: string, color: string)

Live SQL

SQL Worksheet

```
1 CREATE TABLE SAILORS(
2     SID INT,
3     NAME VARCHAR(20),
4     RATING INT,
5     AGE REAL,
6     PRIMARY KEY(SID)
7 );
8 CREATE TABLE BOATS(
9     BID INT,
10    BNAME VARCHAR2(20),
11    COLOR VARCHAR2(20),
12    PRIMARY KEY(BID)
13 );
14
15 CREATE TABLE RESERVES(
```

Table created.

Table created.

2023 Oracle - Live SQL 23.1.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 - Database Documentation - Ask Tom - Dev Gym  
Built with ❤ using Oracle APEX - Privacy - Terms of Use

>> Reserves(sid: integer, bid: integer, day: date)

Live SQL

SQL Worksheet

```
6     PRIMARY KEY(SID)
7 );
8 CREATE TABLE BOATS(
9     BID INT,
10    BNAME VARCHAR2(20),
11    COLOR VARCHAR2(20),
12    PRIMARY KEY(BID)
13 );
14
15 CREATE TABLE RESERVES(
16     SID INT REFERENCES SAILORS(SID),
17     BID INT REFERENCES BOATS(BID),
18     DAY DATE
19 );
20
```

Table created.

2023 Oracle - Live SQL 23.1.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 - Database Documentation - Ask Tom - Dev Gym  
Built with ❤ using Oracle APEX - Privacy - Terms of Use

>> Inserting Record in Sailors table:

Live SQL

SQL Worksheet

```

18    DAY DATE
19  );
20
21  INSERT INTO SAILORS VALUES(22,'DUSTIN',7,45.0);
22  INSERT INTO SAILORS VALUES(29,'BRUTUS',1,33.0);
23  INSERT INTO SAILORS VALUES(31,'LUBBER',8,55.5);
24  INSERT INTO SAILORS VALUES(32,'ANDY',8,25.5);
25  INSERT INTO SAILORS VALUES(58,'RUSTY',10,35.0);
26  INSERT INTO SAILORS VALUES(64,'HORATIO',7,35.0);
27  INSERT INTO SAILORS VALUES(71,'ZORBA',10,16.0);
28  INSERT INTO SAILORS VALUES(74,'HORATIO',9,35.0);
29  INSERT INTO SAILORS VALUES(85,'ART',3,25.5);
30  INSERT INTO SAILORS VALUES(95,'BOB',3,63.5);
31  SELECT *FROM SAILORS;
32

```

1 row(s) inserted.

2023 Oracle - Live SQL 23.1.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0 · Database Documentation · Ask Tom · Dev Gym  
Built with ❤ using Oracle APEX · Privacy · Terms of Use

SID	NAME	RATING	AGE
22	DUSTIN	7	45
29	BRUTUS	1	33
31	LUBBER	8	55.5
32	ANDY	8	25.5
58	RUSTY	10	35
64	HORATIO	7	35
71	ZORBA	10	16
74	HORATIO	9	35
85	ART	3	25.5
95	BOB	3	63.5

## >> Inserting Record in Reserves table:

Live SQL

SQL Worksheet

```

24  INSERT INTO SAILORS VALUES(32,'ANDY',8,25.5);
25  INSERT INTO SAILORS VALUES(58,'RUSTY',10,35.0);
26  INSERT INTO SAILORS VALUES(64,'HORATIO',7,35.0);
27  INSERT INTO SAILORS VALUES(71,'ZORBA',10,16.0);
28  INSERT INTO SAILORS VALUES(74,'HORATIO',9,35.0);
29  INSERT INTO SAILORS VALUES(85,'ART',3,25.5);
30  INSERT INTO SAILORS VALUES(95,'BOB',3,63.5);
31  SELECT *FROM SAILORS;
32
33  INSERT INTO Boats VALUES(101,'Interlake', 'BLUE');
34  INSERT INTO Boats VALUES(102,'Interlake', 'RED');
35  INSERT INTO Boats VALUES(103,'Clipper', 'GREEN');
36  INSERT INTO Boats VALUES(104,'Marine', 'RED');
37  SELECT * FROM Boats;
38

```

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

2023 Oracle - Live SQL 23.1.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0 · Database Documentation · Ask Tom · Dev Gym  
Built with ❤ using Oracle APEX · Privacy · Terms of Use

BID	BANME	COLOR
101	Interlake	BLUE
102	Interlake	RED
103	Clipper	GREEN
104	Marine	RED

>>Inserting Record in Boats table:

The screenshot shows the Oracle Live SQL interface. On the left, the SQL Worksheet pane displays the following SQL code:

```
36 INSERT INTO Boats VALUES(104,'Marine','RED');
37 SELECT * FROM Boats;
38
39 INSERT INTO Reserves VALUES(22,101,'10-OCT-98');
40 INSERT INTO Reserves VALUES(22,102,'10-OCT-98');
41 INSERT INTO Reserves VALUES(22,103,'10-AUG-98');
42 INSERT INTO Reserves VALUES(22,104,'10-JUL-98');
43 INSERT INTO Reserves VALUES(31,102,'11-OCT-98');
44 INSERT INTO Reserves VALUES(31,103,'11-JUN-98');
45 INSERT INTO Reserves VALUES(31,104,'11-DEC-98');
46 INSERT INTO Reserves VALUES(64,101,'09-MAY-98');
47 INSERT INTO Reserves VALUES(64,102,'09-AUG-98');
48 INSERT INTO Reserves VALUES(74,103,'09-AUG-98');
49 SELECT *FROM RESERVES;
50
```

On the right, the Results pane shows a table with three columns: SID, BID, and DAY. The data is as follows:

SID	BID	DAY
22	101	10-OCT-98
22	102	10-OCT-98
22	103	10-AUG-98
22	104	10-JUL-98
31	102	11-OCT-98
31	103	11-JUN-98
31	104	11-DEC-98
64	101	09-MAY-98
64	102	09-AUG-98
74	103	09-AUG-98

Below the table, a message indicates 1 row(s) inserted for each of the five rows of data.

2. Write SQL command for the following:

1. i) Show the names and ages of all sailors.
2. ii) Show the details of the boats which are red and blue in color.
3. iii) Find the oldest and youngest sailors' age.
4. iv) Find the ages of sailors whose name begins and ends with B and has at least three characters.
5. v) Show the average rating of the sailors.
6. vi) Find all sailors with a rating above 7.
7. vii) Find the number of boats reserved by the sailor named Horatio.
8. viii) Find the colors of boats reserved by Lubber.
9. ix) Show the details of the sailors who have reserved the boat with bid 102.
10. x) Find the sid of sailors who have reserved green boats.
11. xi) Find the names of sailors who have reserved boat number 103.
12. xii) Find the sids and names of sailors who have reserved a red boat.
13. xiii) Find the names of the sailors who have reserved a green or a blue boat.
14. xiv) Find the names of sailors who have reserved both a red and a green boat.
15. xv) Find the names of sailors who have reserved at least one boat.

## Solution:

- i) Show the names and ages of all sailors:

Live SQL

SQL Worksheet

```

41 INSERT INTO Reserves VALUES(22,103, '10-AUG-98');
42 INSERT INTO Reserves VALUES(22,104, '10-JUL-98');
43 INSERT INTO Reserves VALUES(31,102, '11-OCT-98');
44 INSERT INTO Reserves VALUES(31,103, '11-JUN-98');
45 INSERT INTO Reserves VALUES(31,104, '11-DEC-98');
46 INSERT INTO Reserves VALUES(64,101, '9-MAY-98');
47 INSERT INTO Reserves VALUES(64,102, '9-AUG-98');
48 INSERT INTO Reserves VALUES(74,103, '9-AUG-98');
49 SELECT *FROM RESERVES;
50
51 SELECT NAME,AGE FROM SAILORS;
52

```

NAME	AGE
DUSTIN	45
BRUTUS	33
LUBBER	55.5
ANDY	25.5
RUSTY	35
HORATIO	35
ZORBA	16
HORATIO	35
ART	25.5
BOB	63.5

2023 Oracle - Live SQL 23.1.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 - Database Documentation - Ask Tom - Dev Gym  
Built with ❤ using Oracle APEX - Privacy - Terms of Use

ii) Show the details of the boats which are red and blue in color.

Live SQL

SQL Worksheet

```

43 INSERT INTO Reserves VALUES(31,102, '11-OCT-98');
44 INSERT INTO Reserves VALUES(31,103, '11-JUN-98');
45 INSERT INTO Reserves VALUES(31,104, '11-DEC-98');
46 INSERT INTO Reserves VALUES(64,101, '9-MAY-98');
47 INSERT INTO Reserves VALUES(64,102, '9-AUG-98');
48 INSERT INTO Reserves VALUES(74,103, '9-AUG-98');
49 SELECT *FROM RESERVES;
50
51 SELECT NAME,AGE FROM SAILORS;
52
53 SELECT *FROM BOATS WHERE COLOR='RED' OR COLOR='BLUE';
54

```

BID	BANME	COLOR
101	Interlake	BLUE
102	Interlake	RED
104	Marine	RED

[Download CSV](#)

3 rows selected.

2023 Oracle - Live SQL 23.1.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 - Database Documentation - Ask Tom - Dev Gym  
Built with ❤ using Oracle APEX - Privacy - Terms of Use

iii) Find the oldest and youngest sailors' age.

The screenshot shows the Oracle Live SQL interface. At the top, there are tabs for 'Live SQL' and 'SQL Worksheet'. The 'Actions' dropdown shows 'ntank\_be21@thapar.edu'. Below the tabs are buttons for 'Feedback', 'Help', 'Clear', 'Find', 'Actions', 'Save', and 'Run'. The SQL Worksheet contains the following code:

```
48 INSERT INTO Reserves VALUES(74,103,'9-AUG-98');
49 SELECT *FROM RESERVES;
50
51 SELECT NAME,AGE FROM SAILORS;
52
53 SELECT *FROM BOATS WHERE COLOR='RED' OR COLOR='BLUE';
54
55 SELECT MIN(AGE) FROM SAILORS;
56
57 SELECT MAX(AGE) FROM SAILORS;
58
```

Below the code, the results for the minimum age are displayed in a table:

MIN(AGE)
16

And the results for the maximum age are displayed in another table:

MAX(AGE)
63.5

At the bottom of the interface, there is a footer bar with the text: "2023 Oracle - Live SQL 23.1.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 - Database Documentation - Ask Tom - Dev Gym" and "Built with ❤️ using Oracle APEX - Privacy - Terms of Use".

iv) Find the ages of sailors whose name begins and ends with B and has at least three characters.

The screenshot shows the Oracle Live SQL interface. At the top, there are tabs for 'Live SQL' and 'SQL Worksheet'. The 'Actions' dropdown shows 'ntank\_be21@thapar.edu'. Below the tabs are buttons for 'Feedback', 'Help', 'Clear', 'Find', 'Actions', 'Save', and 'Run'. The SQL Worksheet contains the following code:

```
49 SELECT *FROM RESERVES;
50
51 SELECT NAME,AGE FROM SAILORS;
52
53 SELECT *FROM BOATS WHERE COLOR='RED' OR COLOR='BLUE';
54
55 SELECT MIN(AGE) FROM SAILORS;
56
57 SELECT MAX(AGE) FROM SAILORS;
58
59 SELECT AGE FROM SAILORS WHERE NAME LIKE 'B%_%B';
60
```

Below the code, the results for the average age are displayed in a table:

AGE
63.5

At the bottom of the interface, there is a footer bar with the text: "2023 Oracle - Live SQL 23.1.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 - Database Documentation - Ask Tom - Dev Gym" and "Built with ❤️ using Oracle APEX - Privacy - Terms of Use".

v) Show the average rating of the sailors.

Live SQL

SQL Worksheet

```

51 SELECT NAME,AGE FROM SAILORS;
52
53 SELECT *FROM BOATS WHERE COLOR='RED' OR COLOR='BLUE';
54
55 SELECT MIN(AGE) FROM SAILORS;
56
57 SELECT MAX(AGE) FROM SAILORS;
58
59 SELECT AGE FROM SAILORS WHERE NAME LIKE 'B%_B';
60
61 SELECT AVG(RATING) FROM SAILORS;
62

```

Avg(Rating)

6.6

[Download CSV](#)

2023 Oracle - Live SQL 23.1.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 - Database Documentation · Ask Tom · Dev Gym  
Built with ❤ using Oracle APEX · Privacy · Terms of Use.

vi) Find all sailors with a rating above 7.

Live SQL

SQL Worksheet

```

53 SELECT *FROM BOATS WHERE COLOR= RED OR COLOR= BLUE ;
54
55 SELECT MIN(AGE) FROM SAILORS;
56
57 SELECT MAX(AGE) FROM SAILORS;
58
59 SELECT AGE FROM SAILORS WHERE NAME LIKE 'B%_B';
60
61 SELECT AVG(RATING) FROM SAILORS;
62
63 SELECT *FROM SAILORS WHERE RATING>7;
64

```

SID	NAME	RATING	AGE
31	LUBBER	8	55.5
32	ANDY	8	25.5
58	RUSTY	10	35
71	ZORBA	10	16
74	HORATIO	9	35

2023 Oracle - Live SQL 23.1.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 - Database Documentation · Ask Tom · Dev Gym  
Built with ❤ using Oracle APEX · Privacy · Terms of Use.

vii) Find the number of boats reserved by the sailor namedHoratio.

Live SQL

SQL Worksheet

```

55 SELECT MIN(AGE) FROM SAILORS;
56
57 SELECT MAX(AGE) FROM SAILORS;
58
59 SELECT AGE FROM SAILORS WHERE NAME LIKE 'B%_%B';
60
61 SELECT AVG(RATING) FROM SAILORS;
62
63 SELECT *FROM SAILORS WHERE RATING>7;
64
65 SELECT COUNT(DISTINCT NAME) FROM SAILORS WHERE NAME='HORATIO';
66

```

COUNT(DISTINCTNAME)

1

[Download CSV](#)

2023 Oracle - Live SQL 23.1.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 - Database Documentation - Ask Tom - Dev Gym  
Built with ❤ using Oracle APEX - Privacy - Terms of Use

viii) Find the colors of boats reserved by Lubber.

Live SQL

SQL Worksheet

```

57 SELECT MAX(AGE) FROM SAILORS;
58
59 SELECT AGE FROM SAILORS WHERE NAME LIKE 'B%_%B';
60
61 SELECT AVG(RATING) FROM SAILORS;
62
63 SELECT *FROM SAILORS WHERE RATING>7;
64
65 SELECT COUNT(DISTINCT NAME) FROM SAILORS WHERE NAME='HORATIO';
66
67 SELECT DISTINCT COLOR FROM SAILORS,RESERVES,BOATS WHERE SAILORS.SID=RESERVES.SID AND RESERVES.BID=BOATS.BID AND SAILORS.NAME='LUBBER';
68

```

COLOR
RED
GREEN

[Download CSV](#)

2 rows selected.

2023 Oracle - Live SQL 23.1.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 - Database Documentation - Ask Tom - Dev Gym  
Built with ❤ using Oracle APEX - Privacy - Terms of Use

ix) Show the details of the sailors who have reserved the boat with bid 102.

Live SQL

SQL Worksheet

```

59 SELECT AVG(RATING) FROM SAILORS WHERE NAME LIKE 'D%_SD';
60
61 SELECT AVG(RATING) FROM SAILORS;
62
63 SELECT *FROM SAILORS WHERE RATING>7;
64
65 SELECT COUNT(DISTINCT NAME) FROM SAILORS WHERE NAME='HORATIO';
66
67 SELECT DISTINCT COLOR FROM SAILORS,RESERVES,BOATS WHERE SAILORS.SID=RESERVES.SID AND RESERVES.BID=BOATS.BID AND SAILORS.NAME='LUBBER';
68
69 SELECT *FROM SAILORS,RESERVES WHERE SAILORS.SID=RESERVES.SID AND RESERVES.BID=102;
70

```

SID	NAME	RATING	AGE	SID	BID	DAY
22	DUSTIN	7	45	22	102	10-OCT-98
31	LUBBER	8	55.5	31	102	11-OCT-98
64	HORATIO	7	35	64	102	09-AUG-98

[Download CSV](#)

3 rows selected.

2023 Oracle - Live SQL 23.1.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 - Database Documentation - Ask Tom - Dev Gym  
Built with ❤ using Oracle APEX - Privacy - Terms of Use

x) Find the sid of sailors who have reserved green boats.

Live SQL

SQL Worksheet

```

61 SELECT AVG(RATING) FROM SAILORS;
62
63 SELECT *FROM SAILORS WHERE RATING>7;
64
65 SELECT COUNT(DISTINCT NAME) FROM SAILORS WHERE NAME='HORATIO';
66
67 SELECT DISTINCT COLOR FROM SAILORS,RESERVES,BOATS WHERE SAILORS.SID=RESERVES.SID AND RESERVES.BID=BOATS.BID AND SAILORS.NAME='LUBBER';
68
69 SELECT *FROM SAILORS,RESERVES WHERE SAILORS.SID=RESERVES.SID AND RESERVES.BID=102;
70
71 SELECT SID FROM RESERVES,BOATS WHERE RESERVES.BID=BOATS.BID AND COLOR='GREEN';
72

```

SID
22
31
74

[Download CSV](#)

3 rows selected.

2023 Oracle - Live SQL 23.1.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 - Database Documentation - Ask Tom - Dev Gym  
Built with ❤ using Oracle APEX - Privacy - Terms of Use

xi) Find the names of sailors who have reserved boat number 103.

Live SQL

SQL Worksheet

```

63 SELECT *FROM SAILORS WHERE RATING>7;
64
65 SELECT COUNT(DISTINCT NAME) FROM SAILORS WHERE NAME='HORATIO';
66
67 SELECT DISTINCT COLOR FROM SAILORS,RESERVES,BOATS WHERE SAILORS.SID=RESERVES.SID AND RESERVES.BID=BOATS.BID AND SAILORS.NAME='LUBBER';
68
69 SELECT *FROM SAILORS,RESERVES WHERE SAILORS.SID=RESERVES.SID AND RESERVES.BID=102;
70
71 SELECT SID FROM RESERVES,BOATS WHERE RESERVES.BID=BOATS.BID AND COLOR='GREEN';
72
73 SELECT NAME FROM SAILORS,RESERVES WHERE SAILORS.SID=RESERVES.SID AND RESERVES.BID=103;
74

```

NAME
DUSTIN
LUBBER
HORATIO

[Download CSV](#)

3 rows selected.

2023 Oracle - Live SQL 23.1.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 - Database Documentation - Ask Tom - Dev Gym  
Built with ❤ using Oracle APEX - Privacy - Terms of Use

xii) Find the sids and names of sailors who have reserved a red boat.

Live SQL

SQL Worksheet

```

66
67 SELECT DISTINCT COLOR FROM SAILORS,RESERVES,BOATS WHERE SAILORS.SID=RESERVES.SID AND RESERVES.BID=BOATS.BID AND SAILORS.NAME='LUBBER';
68
69 SELECT *FROM SAILORS,RESERVES WHERE SAILORS.SID=RESERVES.SID AND RESERVES.BID=102;
70
71 SELECT SID FROM RESERVES,BOATS WHERE RESERVES.BID=BOATS.BID AND COLOR='GREEN';
72
73 SELECT NAME FROM SAILORS,RESERVES WHERE SAILORS.SID=RESERVES.SID AND RESERVES.BID=103;
74
75 SELECT DISTINCT SAILORS.NAME,SAILORS.SID FROM SAILORS,RESERVES,BOATS WHERE SAILORS.SID=RESERVES.SID
76 AND RESERVES.BID=BOATS.BID AND COLOR='RED';
77

```

NAME	SID
DUSTIN	22
HORATIO	64
LUBBER	31

[Download CSV](#)

3 rows selected.

2023 Oracle - Live SQL 23.1.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 - Database Documentation - Ask Tom - Dev Gym  
Built with ❤ using Oracle APEX - Privacy - Terms of Use

xiii) Find the names of the sailors who have reserved a green or a blue boat.

Live SQL

SQL Worksheet

```

59 SELECT * FROM SAILORS,RESERVES WHERE SAILORS.SID=RESERVES.SID AND RESERVES.BID=102;
60
61 SELECT SID FROM RESERVES,BOATS WHERE RESERVES.BID=BOATS.BID AND COLOR='GREEN';
62
63 SELECT NAME FROM SAILORS,RESERVES WHERE SAILORS.SID=RESERVES.SID AND RESERVES.BID=103;
64
65 SELECT DISTINCT SAILORS.NAME,SAILORS.SID FROM SAILORS,RESERVES,BOATS WHERE SAILORS.SID=RESERVES.SID
66           AND RESERVES.BID=BOATS.BID AND COLOR='RED';
67
68 SELECT DISTINCT SAILORS.NAME FROM SAILORS,RESERVES,BOATS WHERE SAILORS.SID=RESERVES.SID
69           AND RESERVES.BID=BOATS.BID AND (BOATS.COLOR='BLUE' OR BOATS.COLOR='GREEN');
70
71
72
73
74
75
76
77
78
79
80

```

NAME
HORATIO
LUBBER
DUSTIN

[Download CSV](#)

3 rows selected.

2023 Oracle - Live SQL 23.1.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 - Database Documentation - Ask Tom - Dev Gym  
Built with ❤ using Oracle APEX - Privacy - Terms of Use

xiv) Find the names of sailors who have reserved both a red and a green boat.

Live SQL

SQL Worksheet

```

74
75 SELECT DISTINCT SAILORS.NAME,SAILORS.SID FROM SAILORS,RESERVES,BOATS WHERE SAILORS.SID=RESERVES.SID
76           AND RESERVES.BID=BOATS.BID AND COLOR='RED';
77
78 SELECT DISTINCT SAILORS.NAME FROM SAILORS,RESERVES,BOATS WHERE SAILORS.SID=RESERVES.SID
79           AND RESERVES.BID=BOATS.BID AND (BOATS.COLOR='BLUE' OR BOATS.COLOR='GREEN');
80
81 SELECT DISTINCT SAILORS.NAME FROM SAILORS,RESERVES,BOATS WHERE SAILORS.SID=RESERVES.SID AND RESERVES.BID=BOATS.BID AND COLOR='RED'
82 INTERSECT
83 SELECT DISTINCT SAILORS.NAME FROM SAILORS,RESERVES,BOATS WHERE SAILORS.SID=RESERVES.SID AND RESERVES.BID=BOATS.BID AND COLOR='GREEN';
84

```

NAME
DUSTIN
HORATIO
LUBBER

[Download CSV](#)

3 rows selected.

2023 Oracle - Live SQL 23.1.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 - Database Documentation - Ask Tom - Dev Gym  
Built with ❤ using Oracle APEX - Privacy - Terms of Use

xv) Find the names of sailors who have reserved at least one boat.

Live SQL

SQL Worksheet

76 | | | AND RESERVES.BID=BOATS.BID AND COLOR='RED';  
77  
78 SELECT DISTINCT SAILORS.NAME FROM SAILORS,RESERVES,BOATS WHERE SAILORS.SID=RESERVES.SID  
79 | | | AND RESERVES.BID=BOATS.BID AND (BOATS.COLOR='BLUE' OR BOATS.COLOR='GREEN');  
80  
81 SELECT DISTINCT SAILORS.NAME FROM SAILORS,RESERVES,BOATS WHERE SAILORS.SID=RESERVES.SID AND RESERVES.BID=BOATS.BID AND COLOR='RED'  
82 INTERSECT  
83 SELECT DISTINCT SAILORS.NAME FROM SAILORS,RESERVES,BOATS WHERE SAILORS.SID=RESERVES.SID AND RESERVES.BID=BOATS.BID AND COLOR='GREEN';  
84  
85 SELECT DISTINCT SAILORS.NAME FROM SAILORS,RESERVES WHERE SAILORS.SID=RESERVES.SID;  
86

NAME
HORATIO
LUBBER
DUSTIN

[Download CSV](#)

3 rows selected.

2023 Oracle - Live SQL 23.1.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 - Database Documentation - Ask Tom - Dev Gym  
Built with ❤ using Oracle APEX - Privacy - Terms of Use

# ASSIGNMENT\_6

## 1. Display the system date

```
mysql> SELECT CURRENT_DATE();
+-----+
| CURRENT_DATE() |
+-----+
| 2023-03-26 |
+-----+
1 row in set (0.00 sec)
```

## 2. Display current day

```
mysql> SELECT DAYNAME(CURRENT_DATE());
+-----+
| DAYNAME(CURRENT_DATE()) |
+-----+
| Sunday |
+-----+
1 row in set (0.00 sec)
```

## 3. Display current month and spell out year

```
mysql> SELECT CONCAT(MONTHNAME(CURRENT_DATE()), ' ', YEAR(CURRENT_DATE()));
+-----+
| CONCAT(MONTHNAME(CURRENT_DATE()), ' ', YEAR(CURRENT_DATE())) |
+-----+
| March 2023 |
+-----+
1 row in set (0.00 sec)
```

## 4. Display spell out current date

```
mysql> SELECT DATE_FORMAT(CURRENT_DATE(), '%W, %M %e, %Y');
+-----+
| DATE_FORMAT(CURRENT_DATE(), '%W, %M %e, %Y') |
+-----+
| Sunday, March 26, 2023 |
+-----+
1 row in set (0.00 sec)
```

## 5. Check whether it is AM or PM right now

```
mysql> SELECT IF(TIME_FORMAT(CURRENT_TIME(), '%p') = 'AM', 'It is currently AM', 'It is currently PM')
+-----+
| IF(TIME_FORMAT(CURRENT_TIME(), '%p') = 'AM', 'It is currently AM', 'It is currently PM') |
+-----+
| It is currently PM |
+-----+
1 row in set (0.00 sec)
```

6. Display the date of next Friday

```
mysql> SELECT DATE_ADD(CURRENT_DATE(), INTERVAL (5 - DAYOFWEEK(CURRENT_DATE())) % 7 + 1 DAY);
+-----+
| DATE_ADD(CURRENT_DATE(), INTERVAL (5 - DAYOFWEEK(CURRENT_DATE())) % 7 + 1 DAY) |
+-----+
| 2023-03-31
|
+-----+
1 row in set (0.00 sec)
```

7. Round the system date on month

```
mysql> SELECT DATE_FORMAT(CURRENT_DATE() + INTERVAL 15 DAY, '%Y-%m-01');
+-----+
| DATE_FORMAT(CURRENT_DATE() + INTERVAL 15 DAY, '%Y-%m-01') |
+-----+
| 2023-04-01
|
+-----+
1 row in set (0.00 sec)
```

8. Truncate the system date on month

```
mysql> SELECT LAST_DAY(CURRENT_DATE()) + INTERVAL 1 DAY - INTERVAL DAY(LAST_DAY(CURRENT_DATE())) DAY;
+-----+
| LAST_DAY(CURRENT_DATE()) + INTERVAL 1 DAY - INTERVAL DAY(LAST_DAY(CURRENT_DATE())) DAY |
+-----+
| 2023-03-01
|
+-----+
1 row in set (0.00 sec)
```

9. Round the system date on year

```
mysql> SELECT DATE_FORMAT(CURRENT_DATE() + INTERVAL 6 MONTH, '%Y-01-01');
+-----+
| DATE_FORMAT(CURRENT_DATE() + INTERVAL 6 MONTH, '%Y-01-01') |
+-----+
| 2023-01-01
|
+-----+
1 row in set (0.00 sec)
```

10. Truncate the system date on year

```
mysql> SELECT DATE_FORMAT(CURRENT_DATE(), '%Y-01-01');
+-----+
| DATE_FORMAT(CURRENT_DATE(), '%Y-01-01') |
+-----+
| 2023-01-01
|
+-----+
1 row in set (0.00 sec)
```

11. Find the day after three days

```
mysql> SELECT DATE_ADD(CURRENT_DATE(), INTERVAL 3 DAY);
+-----+
| DATE_ADD(CURRENT_DATE(), INTERVAL 3 DAY) |
+-----+
| 2023-03-29
|
+-----+
1 row in set (0.00 sec)
```

Create a table train having three four columns

12. Train Number, date of Departure, time of departure, time of arrival

```
mysql> use lab6;
Database changed
mysql> CREATE TABLE train (
    ->     train_number INT,
    ->     departure_date DATE,
    ->     departure_time TIME,
    ->     arrival_time TIME
    -> );
Query OK, 0 rows affected (0.02 sec)
```

13. Insert five columns in train table

```
mysql> INSERT INTO train (train_number, departure_date, departure_time, arrival_time)
-> VALUES
-> (1001, '2023-03-27', '08:00:00', '12:30:00'),
-> (1002, '2023-03-28', '10:15:00', '14:45:00'),
-> (1003, '2023-03-29', '14:30:00', '19:00:00'),
-> (1004, '2023-03-30', '18:45:00', '23:15:00'),
-> (1005, '2023-03-31', '22:00:00', '02:30:00);
Query OK, 5 rows affected (0.01 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

14. Display all the records

```
mysql> SELECT * FROM train;
+-----+-----+-----+-----+
| train_number | departure_date | departure_time | arrival_time |
+-----+-----+-----+-----+
| 1001 | 2023-03-27 | 08:00:00 | 12:30:00 |
| 1002 | 2023-03-28 | 10:15:00 | 14:45:00 |
| 1003 | 2023-03-29 | 14:30:00 | 19:00:00 |
| 1004 | 2023-03-30 | 18:45:00 | 23:15:00 |
| 1005 | 2023-03-31 | 22:00:00 | 02:30:00 |
+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

15. Display the time values inserted in the columns

```
mysql> SELECT departure_time, arrival_time FROM train;
+-----+-----+
| departure_time | arrival_time |
+-----+-----+
| 08:00:00 | 12:30:00 |
| 10:15:00 | 14:45:00 |
| 14:30:00 | 19:00:00 |
| 18:45:00 | 23:15:00 |
| 22:00:00 | 02:30:00 |
+-----+-----+
5 rows in set (0.00 sec)
```

16. Display those trains which arrived on PM

```
mysql> SELECT * FROM train WHERE arrival_time >= '12:00:00' AND arrival_time < '24:00:00';
+-----+-----+-----+-----+
| train_number | departure_date | departure_time | arrival_time |
+-----+-----+-----+-----+
|      1001 | 2023-03-27    | 08:00:00       | 12:30:00      |
|      1002 | 2023-03-28    | 10:15:00       | 14:45:00      |
|      1003 | 2023-03-29    | 14:30:00       | 19:00:00      |
|      1004 | 2023-03-30    | 18:45:00       | 23:15:00      |
+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

17. Display train number who are going to depart in next on hour.

```
mysql> SELECT train_number FROM train
      -> WHERE departure_time >= NOW() AND departure_time < DATE_ADD(NOW(), INTERVAL 1 HOUR);
Empty set (0.00 sec)
```

# ASSIGNMENT\_7

1. Check the structure of tables.

```
mysql> show tables;
+----------------+
| Tables_in_lab4 |
+----------------+
| book           |
| dept           |
| emp            |
| emp2           |
| p              |
| s              |
| sp             |
| st             |
+----------------+
8 rows in set (0.00 sec)

mysql> desc p;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| pno   | int  | NO   | PRI | NULL    |       |
| pname | varchar(50) | YES  |     | NULL    |       |
| color  | varchar(50) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> desc s;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| sno   | int  | NO   | PRI | NULL    |       |
| sname | varchar(50) | YES  |     | NULL    |       |
| city   | varchar(50) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> desc sp;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| sno   | int  | NO   | PRI | NULL    |       |
| pno   | int  | NO   | PRI | NULL    |       |
| qty   | int  | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> desc st;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| Rno   | int  | NO   | PRI | NULL    |       |
| Class | char(1) | NO   | PRI | NULL    |       |
| Marks | int  | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> desc book;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| Rno   | int  | NO   | PRI | NULL    |       |
| DOI   | date | YES  |     | NULL    |       |
| DOR   | date | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

```

mysql> desc dept;
+-----+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| deptno | int    | NO   | PRI | NULL    |       |
| dname   | varchar(50) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> desc emp;
+-----+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| empno | int    | NO   | PRI | NULL    |       |
| ename  | varchar(50) | YES  | UNI | NULL    |       |
| job    | varchar(10) | YES  |     | NULL    |       |
| sal    | decimal(10,2) | NO   |     | NULL    |       |
| deptno | int    | YES  |     | 10      |       |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> desc emp2;
+-----+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| empno | int    | NO   | PRI | NULL    |       |
| ename  | varchar(50) | YES  | UNI | NULL    |       |
| job    | varchar(50) | YES  |     | NULL    |       |
| sal    | decimal(10,2) | NO   |     | NULL    |       |
| deptno | int    | YES  | MUL | NULL    |       |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

```

- Check the constraint name for applied constraints?

```

mysql> SHOW CREATE TABLE emp;
+-----+-----+
| Table | Create Table |
+-----+-----+
| emp   | CREATE TABLE `emp` (
  `empno` int NOT NULL,
  `ename` varchar(50) DEFAULT NULL,
  `job` varchar(10) DEFAULT NULL,
  `sal` decimal(10,2) NOT NULL,
  `deptno` int DEFAULT '10',
  PRIMARY KEY (`empno`),
  UNIQUE KEY `ename` (`ename`),
  CONSTRAINT `emp_chk_1` CHECK ((`job` in ('Prof','AP','Lect'))),
  ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci |
+-----+-----+
1 row in set (0.00 sec)

```

- Drop the unique constraint on ENAME

```

mysql> ALTER TABLE emp DROP INDEX ENAME ;
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> |

```

- Drop the Foreign Key constraint on DEPTNO

```
alter table emp drop constraint SYS_C00117255289;
```

- Add Foreign Key constraint on DEPTNO

```
alter table emp add foreign key (deptno) references dept(deptno);
```

- Change Data type of ENAME

```

mysql> ALTER TABLE emp MODIFY column ename VArCHAR(50);
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0

```

7. Change width of DNAME

```
mysql> ALTER TABLE dept MODIFY column dname VARCHAR(100)
Query OK, 3 rows affected (0.04 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

8. Add COMM column in EMP table

```
mysql> ALTER TABLE emp ADD COLUMN COMM DECIMAL(10,2);
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

9. Drop CITY column from J

```
table alter table J drop
column city;
```

10. Create duplicate copy of EMP table

```
mysql> CREATE TABLE emp_copy AS SELECT * FROM emp;
Query OK, 4 rows affected (0.02 sec)
Records: 4  Duplicates: 0  Warnings: 0
```

11. Copy structure of DEPT table in another table with different column names

```
mysql> CREATE TABLE NEW_TABLE (
    ->     NEW_DEPTNO INT NOT NULL,
    ->     NEW_DNAME VARCHAR(14),
    ->     PRIMARY KEY (NEW_DEPTNO)
    -> ) AS
    -> SELECT DEPTNO AS NEW_DEPTNO, DNAME AS NEW_DNAME FROM DEPT WHERE 1=0;
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

12. Change the name and job of the employee whose EMPNO =100

```
mysql> UPDATE EMP
    -> SET ENAME = 'PULKIT', JOB = 'MANAG
    -> WHERE EMPNO = 100;
Query OK, 0 rows affected (0.00 sec)
Rows matched: 0  Changed: 0  Warnings: 0
```

13. Delete the record of employee who belongs to computer department

```
mysql> DELETE FROM EMP
    -> WHERE Deptno = (SELECT deptno FROM dept WHERE dname= 'comp ');
Query OK, 0 rows affected (0.00 sec)
```

14. Drop DEPT Table

```
mysql> DROP TABLE DEPT;
ERROR 3730 (HY000): Cannot drop table 'dept' referenced by a foreign key constraint 'emp2_ibfk_1' on table 'emp2'.
```

15. Drop duplicate table of EMP table

```
mysql> DROP TABLE emp_copy;
Query OK, 0 rows affected (0.01 sec)
```

# ASSIGNMENT\_8

1. Create the following schema and insert some tuples in these tables shown below.
- Author (ID, Name, Birth\_Year, Death\_Year (NULL in case of Author is alive)) Book (ID, Author\_ID, Title, Publish\_Year, Publishing\_House, Rating) Adaptation (Book\_ID, Type, Title, Release\_Year, Rating)

```
mysql> SELECT * FROM adaptation
-> ;
+-----+-----+-----+-----+-----+
| Book_ID | Type | Title | Release_Year | Rating |
+-----+-----+-----+-----+-----+
| 3 | Movie | Harry Potter and the Philosopher's Stone | 2001 | 4.0 |
| 4 | Movie | Harry Potter and the Chamber of Secrets | 2002 | 4.2 |
| 5 | Movie | Twilight | 2008 | 3.1 |
| 6 | Movie | New Moon | 2009 | 3.3 |
| 7 | Movie | The Hunger Games | 2012 | 4.1 |
| 8 | Movie | Catching Fire | 2013 | 4.3 |
| 9 | TV Series | Game of Thrones | 2011 | 4.8 |
| 10 | TV Series | Game of Thrones | 2012 | 4.7 |
+-----+-----+-----+-----+-----+
8 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM BOOK;
+-----+-----+-----+-----+-----+
| ID | Author_ID | Title | Publish_Year | Publishing_House | Rating |
+-----+-----+-----+-----+-----+
| 1 | 1 | Five Point Someone | 2004 | Rupa Publications | 3.8 |
| 2 | 1 | 2 States | 2009 | Rupa Publications | 4.1 |
| 3 | 2 | Harry Potter and the Philosopher's Stone | 1997 | Bloomsbury Publishing | 4.5 |
| 4 | 2 | Harry Potter and the Chamber of Secrets | 1998 | Bloomsbury Publishing | 4.3 |
| 5 | 3 | Twilight | 2005 | Little, Brown and Company | 3.2 |
| 6 | 3 | New Moon | 2006 | Little, Brown and Company | 3.5 |
| 7 | 4 | The Hunger Games | 2008 | Scholastic Corporation | 4.0 |
| 8 | 4 | Catching Fire | 2009 | Scholastic Corporation | 4.2 |
| 9 | 5 | A Game of Thrones | 1996 | Bantam Spectra | 4.7 |
| 10 | 5 | A Clash of Kings | 1998 | Bantam Spectra | 4.6 |
+-----+-----+-----+-----+-----+
10 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM AUTHOR;
+-----+-----+-----+-----+
| ID | Name | Birth_Year | Death_Year |
+-----+-----+-----+-----+
| 1 | Chetan Bhagat | 1974 | NULL |
| 2 | J.K. Rowling | 1965 | NULL |
| 3 | Stephenie Meyer | 1973 | NULL |
| 4 | Suzanne Collins | 1962 | NULL |
| 5 | George R.R. Martin | 1948 | NULL |
+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

2. Write SQL command for the following using Join:

- a. Display the title of each book and the name of its author.

```

mysql> SELECT Book.Title, Author.Name
-> FROM Book
-> JOIN Author ON Book.Author_ID = Author.ID;
+-----+-----+
| Title          | Name       |
+-----+-----+
| Five Point Someone | Chetan Bhagat |
| 2 States        | Chetan Bhagat |
| Harry Potter and the Philosopher's Stone | J.K. Rowling |
| Harry Potter and the Chamber of Secrets | J.K. Rowling |
| Twilight         | Stephenie Meyer |
| New Moon         | Stephenie Meyer |
| The Hunger Games | Suzanne Collins |
| Catching Fire    | Suzanne Collins |
| A Game of Thrones | George R.R. Martin |
| A Clash of Kings | George R.R. Martin |
+-----+-----+
10 rows in set (0.00 sec)

```

- b. Display the name of each author together with the title of the book they wrote and the year in which that book was published (Show only books published after 2005).

```

mysql> SELECT Author.Name, Book.Title, Book.Publish_Year
-> FROM Author
-> JOIN Book ON Author.ID = Book.Author_ID
-> WHERE Book.Publish_Year > 2005;
+-----+-----+-----+
| Name      | Title          | Publish_Year |
+-----+-----+-----+
| Chetan Bhagat | 2 States        | 2009 |
| Stephenie Meyer | New Moon        | 2006 |
| Suzanne Collins | The Hunger Games | 2008 |
| Suzanne Collins | Catching Fire    | 2009 |
+-----+-----+-----+
4 rows in set (0.00 sec)

```

- c. For each book, show its title, adaptation title, adaptation year, and publication year. Consider only books with a rating lower than the rating of their corresponding adaptation.

```

mysql> SELECT Book.Title, Adaptation.Title, Adaptation.Release_Year, Book.Publish_Year
-> FROM Book
-> JOIN Adaptation ON Book.ID = Adaptation.Book_ID
-> WHERE Book.Rating < Adaptation.Rating;
+-----+-----+-----+-----+
| Title      | Title          | Release_Year | Publish_Year |
+-----+-----+-----+-----+
| The Hunger Games | The Hunger Games | 2012 | 2008 |
| Catching Fire | Catching Fire    | 2013 | 2009 |
| A Game of Thrones | Game of Thrones | 2011 | 1996 |
| A Clash of Kings | Game of Thrones | 2012 | 1998 |
+-----+-----+-----+-----+
4 rows in set (0.00 sec)

```

- d. Display the title of each book together with its rating. Consider only those books that were published by authors who are still alive. (Use Inner Join).

```

mysql> SELECT Book.Title, Book.Rating
-> FROM Book
-> JOIN Author ON Book.Author_ID = Author.ID
-> WHERE Author.Death_Year IS NULL;
+-----+-----+
| Title          | Rating |
+-----+-----+
| Five Point Someone | 3.8 |
| 2 States        | 4.1 |
| Harry Potter and the Philosopher's Stone | 4.5 |
| Harry Potter and the Chamber of Secrets | 4.3 |
| Twilight         | 3.2 |
| New Moon         | 3.5 |
| The Hunger Games | 4.0 |
| Catching Fire    | 4.2 |
| A Game of Thrones | 4.7 |
| A Clash of Kings | 4.6 |
+-----+-----+
10 rows in set (0.00 sec)

```

- e. Display the title of each book along with the name of its author. Show all books, even those without an author. Show all authors, even those who haven't published a book yet.(Use Full JOIN).

```
SELECT Book.Title,
Author.Name FROM Book
FULL JOIN Author ON Book.Author_ID = Author.ID;
```

- f. Generate all possible pairs of book titles and author names. Consider only books whose author's name is 'Chetan Bhagat' (Use Cross JOIN).

```
mysql> SELECT Book.Title, Author.Name
-> FROM Book
-> CROSS JOIN Author
-> WHERE Author.Name = 'Chetan Bhagat';
+-----+-----+
| Title          | Name   |
+-----+-----+
| Five Point Someone | Chetan Bhagat |
| 2 States        | Chetan Bhagat |
| Harry Potter and the Philosopher's Stone | Chetan Bhagat |
| Harry Potter and the Chamber of Secrets | Chetan Bhagat |
| Twilight         | Chetan Bhagat |
| New Moon         | Chetan Bhagat |
| The Hunger Games | Chetan Bhagat |
| Catching Fire    | Chetan Bhagat |
| A Game of Thrones | Chetan Bhagat |
| A Clash of Kings  | Chetan Bhagat |
+-----+-----+
10 rows in set (0.00 sec)
```

- g. Select each book's title, the name of its publishing house and the title of its adaptation on the type of the adaptation ('Movie'). (Use Right JOIN).

```
mysql> SELECT Book.Title, Book.Publishing_House, Adaptation.Title
-> FROM Adaptation
-> RIGHT JOIN Book ON Adaptation.Book_ID = Book.ID
-> WHERE Adaptation.Type = 'Movie';
+-----+-----+-----+
| Title          | Publishing_House | Title      |
+-----+-----+-----+
| Harry Potter and the Philosopher's Stone | Bloomsbury Publishing | Harry Potter and the Philosopher's Stone |
| Harry Potter and the Chamber of Secrets | Bloomsbury Publishing | Harry Potter and the Chamber of Secrets |
| Twilight         | Little, Brown and Company | Twilight   |
| New Moon         | Little, Brown and Company | New Moon   |
| The Hunger Games | Scholastic Corporation | The Hunger Games |
| Catching Fire    | Scholastic Corporation | Catching Fire|
+-----+-----+-----+
6 rows in set (0.00 sec)
```

- h. Show the title of each book and the name of its author — but only if the author was born in the 20th century. Otherwise, the author's name field should be NULL (Use Left JOIN).

```
mysql> SELECT Book.Title, CASE WHEN Author.Birth_Year BETWEEN 1900 AND 1999 THEN Author.Name ELSE NULL END AS Author_Name
-> FROM Book
-> LEFT JOIN Author ON Book.Author_ID = Author.ID;
+-----+-----+
| Title          | Author_Name |
+-----+-----+
| Five Point Someone | Chetan Bhagat |
| 2 States        | Chetan Bhagat |
| Harry Potter and the Philosopher's Stone | J.K. Rowling |
| Harry Potter and the Chamber of Secrets | J.K. Rowling |
| Twilight         | Stephenie Meyer |
| New Moon         | Stephenie Meyer |
| The Hunger Games | Suzanne Collins |
| Catching Fire    | Suzanne Collins |
| A Game of Thrones | George R.R. Martin |
| A Clash of Kings  | George R.R. Martin |
+-----+-----+
10 rows in set (0.00 sec)
```

3. Consider the following relation and execute the given queries(Aggregate/Group By/Having):

**i) Show the total number of prizes awarded.**

```
SELECT COUNT(winner) FROM nobel
```

**ii) List each subject - just once**

```
SELECT DISTINCT subject FROM nobel
```

**iii) Show the total number of prizes awarded for Physics**

```
SELECT COUNT(*) FROM nobel WHERE subject='Physics'
```

**iv) For each subject show the subject and the number of prizes.**

```
SELECT subject,
```

```
COUNT(*) FROM nobel
```

```
GROUP BY subject
```

**v) For each subject show the first year that the prize was awarded.**

```
SELECT subject,MIN(yr) FROM
```

```
nobel GROUP BY subject
```

**vi) For each subject show the number of prizes awarded in the year 2000.**

```
SELECT subject,
```

```
COUNT(winner) FROM nobel
```

```
WHERE yr=2000
```

```
GROUP BY subject
```

**vii) Show the number of different winners for each**

**subject.** SELECT DISTINCT subject,

```
COUNT(DISTINCT winner) FROM nobel
```

```
GROUP BY subject
```

**viii) For each subject show how many years have had prizes awarded.**

```
SELECT subject, COUNT(DISTINCT yr)
```

```
FROM nobel
```

```
GROUP BY subject
```

**ix) Show the years in which three prizes were given for**

**Physics.** SELECT yr FROM nobel

```
WHERE
```

```
subject='Physics' GROUP
```

```
BY yr
```

```
HAVING COUNT(yr)=3
```

**x) Show winners who have won more than once.**

SELECT winner FROM nobel

GROUP BY winner

HAVING COUNT(winner)>1

**xi) Show winners who have won more than one subject.**

SELECT winner FROM nobel

GROUP BY winner

HAVING COUNT(DISTINCT subject) > 1

**xii) Show the year and subject where 3 prizes were given. Show only years 2000 onwards.**

SELECT yr, subject FROM nobel

WHERE yr >= 2000

GROUP BY yr, subject

HAVING COUNT(DISTINCT winner)=3

1) WAP to find the greatest of three numbers.

```
DECLARE
    num1 NUMBER := 2;
    num2 NUMBER := 5;
    num3 NUMBER := 7;
    greatest NUMBER;
BEGIN
    IF(num1 > num2 AND num1 > num3) THEN
        greatest := num1;
    ELSIF(num2 > num1 AND num2 > num3) THEN
        greatest := num2;
    ELSE
        greatest := num3;
    END IF;
    DBMS_OUTPUT.PUT_LINE('The greatest number is: ' || greatest);
END;
```

Statement processed.  
The greatest is 5

WAP to check whether number is odd or even.

```
DECLARE
    num NUMBER := 5;
BEGIN
    IF(num MOD 2 = 0) THEN
        DBMS_OUTPUT.PUT_LINE('The number is even');
    ELSE
        DBMS_OUTPUT.PUT_LINE('The number is odd');
    END IF;
END;
```

Statement processed.  
The number is odd

WAP to find the grade.

```
DECLARE
    marks NUMBER := 75;
    grade CHAR(1);
BEGIN
    IF(marks > 80) THEN
        grade := 'A';
    ELSIF(marks > 70) THEN
        grade := 'B';
    ELSIF(marks > 50) THEN
        grade := 'C';
    ELSIF(marks > 40) THEN
        grade := 'D';
    ELSE
        grade := 'E';
    END IF;
    DBMS_OUTPUT.PUT_LINE('The grade is: ' || grade);
END;
```

```
Statement processed.  
The grade is: B
```

WAP to print the table of a given number.(use for loop)

```
DECLARE  
    num NUMBER := 6;  
BEGIN  
    FOR i IN 1..10 LOOP  
        DBMS_OUTPUT.PUT_LINE(num || ' x ' || i || ' = ' || num*i);  
    END LOOP;  
END;
```

```
Statement processed.  
6 x 1 = 6  
6 x 2 = 12  
6 x 3 = 18  
6 x 4 = 24  
6 x 5 = 30  
6 x 6 = 36  
6 x 7 = 42  
6 x 8 = 48  
6 x 9 = 54  
6 x 10 = 60
```

WAP to find out the factorial of a given number.(use while loop)

```
DECLARE  
    num NUMBER := &num;  
    factorial NUMBER := 1;  
BEGIN  
    WHILE(num > 0) LOOP  
        factorial := factorial*num;  
        num := num-1;  
    END LOOP;  
    DBMS_OUTPUT.PUT_LINE('The factorial is: ' || factorial);  
END;
```

WAP to find out the Fibonacci series.

```
DECLARE  
    num NUMBER := 5;  
    fib1 NUMBER := 0;  
    fib2 NUMBER := 1;  
    fib3 NUMBER;  
BEGIN  
    DBMS_OUTPUT.PUT_LINE(fib1);  
    DBMS_OUTPUT.PUT_LINE(fib2);  
    FOR i IN 3..num LOOP  
        fib3 := fib1 + fib2;  
        DBMS_OUTPUT.PUT_LINE(fib3);  
        fib1 := fib2;  
        fib2 := fib3;  
    END LOOP;
```

```
END;
```

```
Statement processed.
```

```
0  
1  
1  
2  
3
```

WAP to find the reverse of a number

```
DECLARE  
    num NUMBER := 12;  
    reverse NUMBER := 0;  
    rem NUMBER;  
BEGIN  
    WHILE(num > 0) LOOP  
        rem := num MOD 10;  
        reverse := reverse*10 + rem;  
        num := FLOOR(num/10); -- Use integer division using the FLOOR function  
    END LOOP;  
    DBMS_OUTPUT.PUT_LINE('The reverse number is: ' || reverse);  
END;
```

```
Statement processed.
```

```
The reverse number is: 21
```

Write PL/SQL block that performs addition (+), subtraction (-), multiplication (\*) and division (/) of two numbers as choice by the user.

```
DECLARE  
    -- Declare variables for two numbers and the operation choice  
    num1 NUMBER;  
    num2 NUMBER;  
    choice CHAR(1);  
    result NUMBER;  
  
BEGIN  
    -- Prompt the user to enter two numbers  
    num1 := 3;  
    num2 := 7;  
  
    -- Prompt the user to choose an operation  
    choice := '+';  
  
    -- Perform the operation based on user choice  
    CASE choice  
        WHEN '+' THEN  
            result := num1 + num2;  
            dbms_output.put_line(num1 || ' + ' || num2 || ' = ' || result);  
        WHEN '-' THEN  
            result := num1 - num2;  
            dbms_output.put_line(num1 || ' - ' || num2 || ' = ' || result);  
        WHEN '*' THEN  
            result := num1 * num2;  
            dbms_output.put_line(num1 || ' * ' || num2 || ' = ' || result);  
        WHEN '/' THEN  
            IF num2 = 0 THEN  
                dbms_output.put_line('Cannot divide by zero');  
            ELSE
```

```

    result := num1 / num2;
    dbms_output.put_line(num1 || ' ' || num2 || ' = ' || result);
END IF;
ELSE
    dbms_output.put_line('Invalid choice');
END CASE;
END;

```

Statement processed.  
3 + 7 = 10

Write PL/SQL block to print 5, 10, 15,20 by using For Loop.

```

BEGIN
FOR i IN 1..4 LOOP
    DBMS_OUTPUT.PUT_LINE(i * 5);
END LOOP;
END;

```

Statement processed.  
5  
10  
15  
20

Write PI/SQL block to display welcome message like good morning, good afternoon, good

```

DECLARE
    current_hour NUMBER := TO_NUMBER(TO_CHAR(SYSDATE, 'HH24'));
BEGIN
IF current_hour >= 0 AND current_hour < 12 THEN
    DBMS_OUTPUT.PUT_LINE('Good morning!');
ELSIF current_hour >= 12 AND current_hour < 18 THEN
    DBMS_OUTPUT.PUT_LINE('Good afternoon!');
ELSE
    DBMS_OUTPUT.PUT_LINE('Good night!');
END IF;
END;

```

Statement processed.  
Good morning!



Double-click (or enter) to edit

```
DECLARE
    emp_salary NUMBER;
    emp_count NUMBER;
BEGIN
    -- Too Many Rows
    SELECT COUNT(*) INTO emp_count FROM hr.employees;
    IF(emp_count > 1) THEN
        RAISE_APPLICATION_ERROR(-20100, 'Too Many Rows');
    END IF;

    -- No Data Found
    SELECT salary INTO emp_salary FROM hr.employees WHERE employee_id = 9999;
    IF(emp_salary IS NULL) THEN
        RAISE_APPLICATION_ERROR(-20101, 'No Data Found');
    END IF;
END;
```

```
ORA-20100: Too Many Rows ORA-06512: at line 8
ORA-06512: at "SYS.DBMS_SQL", line 1721
```

Write a PL/SQL code to display a message to check whether the record is deleted or not.

```
DECLARE
    emp_id employees.employee_id%TYPE:= 234;
BEGIN
    DELETE FROM employees WHERE employee_id = emp_id;
    IF(SQL%ROWCOUNT = 1) THEN
        DBMS_OUTPUT.PUT_LINE('Record Deleted Successfully');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Record Not Found');
    END IF;
END;
```

Double-click (or enter) to edit

Write a PL/SQL code to display a message to provide the information about the number of records deleted by the delete statement issued in a PL/SQL block. Cursor EMP (empno, ename, job, sal, deptno)

```
DECLARE
    emp_count NUMBER;
BEGIN
    DELETE FROM employees WHERE department_id = 10;
    emp_count := SQL%ROWCOUNT;
    DBMS_OUTPUT.PUT_LINE('Total ' || emp_count || ' records deleted');
END;
```

Double-click (or enter) to edit

Write a PL/SQL code to demonstrate %TYPE and %ROWTYPE to display details of employees in EMP table.

```
DECLARE
    emp employees%ROWTYPE;
BEGIN
    SELECT * INTO emp FROM employees WHERE employee_id = 100;
    DBMS_OUTPUT.PUT_LINE('Employee Details:');
    DBMS_OUTPUT.PUT_LINE('Employee ID: ' || emp.employee_id);
    DBMS_OUTPUT.PUT_LINE('Employee Name: ' || emp.first_name || ' ' || emp.last_name);
    DBMS_OUTPUT.PUT_LINE('Job: ' || emp.job_id);
    DBMS_OUTPUT.PUT_LINE('Salary: ' || emp.salary);
    DBMS_OUTPUT.PUT_LINE('Department ID: ' || emp.department_id);
END;
```

Write a PL/SQL code to display the empno, ename, job of employees of department number 10 for EMP table (using Cursor).

```
DECLARE
  CURSOR emp_cursor IS
    SELECT employee_id, first_name, last_name, job_id
    FROM employees
    WHERE department_id = 10;
BEGIN
  DBMS_OUTPUT.PUT_LINE('Employee Details:');
  FOR emp IN emp_cursor LOOP
    DBMS_OUTPUT.PUT_LINE('Employee ID: ' || emp.employee_id);
    DBMS_OUTPUT.PUT_LINE('Employee Name: ' || emp.first_name || ' ' || emp.last_name);
    DBMS_OUTPUT.PUT_LINE('Job: ' || emp.job_id);
    DBMS_OUTPUT.PUT_LINE('-----');
  END LOOP;
END;
```

Write a PL/SQL code to display the employee number and name of top 5 highest paid Employees (using Cursor).

```
DECLARE
  CURSOR emp_cursor IS
    SELECT employee_id, first_name || ' ' || last_name AS emp_name, salary
    FROM employees
    ORDER BY salary DESC
    FETCH FIRST 5 ROWS ONLY;
BEGIN
  DBMS_OUTPUT.PUT_LINE('Top 5 highest paid Employees:');
  FOR emp IN emp_cursor LOOP
    DBMS_OUTPUT.PUT_LINE('Employee ID: ' || emp.employee_id);
    DBMS_OUTPUT.PUT_LINE
```

Write a PL/SQL code to calculate the total salary of first n records of emp table. The value of n is passed to cursor as parameter

```
DECLARE
  n NUMBER := 5;
  total_salary NUMBER := 0;
  CURSOR emp_cursor IS
    SELECT salary FROM emp WHERE ROWNUM <= n;
BEGIN
  FOR emp_record IN emp_cursor LOOP
    total_salary := total_salary + emp_record.salary;
  END LOOP;
  DBMS_OUTPUT.PUT_LINE('Total salary of first ' || n || ' employees is: ' || total_salary);
END;
```

PL/SQL code for local procedure to return multiple values through parameters:

```
CREATE OR REPLACE PROCEDURE arithmetic_operations(
    num1 IN NUMBER,
    num2 IN NUMBER,
    sum OUT NUMBER,
    diff OUT NUMBER,
    prod OUT NUMBER,
    quotient OUT NUMBER) IS
BEGIN
    sum := num1 + num2;
    diff := num1 - num2;
    prod := num1 * num2;
    quotient := num1 / num2;
END;
```

PL/SQL code for local procedure to raise salary and update it to the database:

```
CREATE OR REPLACE PROCEDURE raise_salary(
    empid IN NUMBER,
    bonus IN NUMBER) IS
    current_salary NUMBER;
BEGIN
    SELECT salary INTO current_salary FROM emp WHERE eno = empid;
    UPDATE emp SET salary = current_salary + bonus WHERE eno = empid;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Salary of employee ' || empid || ' has been raised by ' || bonus);
END;
```

PL/SQL code for stored procedure to delete employee by employee number:

```
CREATE OR REPLACE PROCEDURE fire_employee(
    empid IN NUMBER) IS
BEGIN
    DELETE FROM emp WHERE eno = empid;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Employee ' || empid || ' has been fired');
END;
```

PL/SQL code for local function to return total number of employees in EMP table:

```
CREATE OR REPLACE FUNCTION get_employee_count RETURN NUMBER IS
    employee_count NUMBER;
BEGIN
    SELECT COUNT(*) INTO employee_count FROM emp;
    RETURN employee_count;
END;
```

PL/SQL code for stored function to add two numbers:

```
CREATE OR REPLACE FUNCTION add_numbers(
    num1 IN NUMBER,
    num2 IN NUMBER) RETURN NUMBER IS
    result NUMBER;
BEGIN
    result := num1 + num2;
    RETURN result;
END;
```

PL/SQL code for stored function to return the number of records updated by UPDATE statement:

```
CREATE OR REPLACE FUNCTION get_update_count RETURN NUMBER IS
    update_count NUMBER;
BEGIN
    UPDATE emp SET salary = salary + 100 WHERE eno = 100;
    update_count := SQL%ROWCOUNT;
    RETURN update_count;
END;
```

```
END;
```

PL/SQL code for trigger to convert ENAME column values to uppercase:

```
CREATE OR REPLACE TRIGGER convert_ename_to_uppercase
BEFORE INSERT OR UPDATE ON emp
FOR EACH ROW
BEGIN
    :NEW.ename := UPPER(:NEW.ename);
END;
```

PL/SQL code for trigger to show old and new values of ENAME after every update:

```
CREATE OR REPLACE TRIGGER show_old_new_ename
AFTER UPDATE ON emp
FOR EACH ROW
BEGIN
    DBMS_OUTPUT.PUT_LINE('Old value of ENAME: ' || :OLD.ename || ', New value of ENAME: ' || :NEW.ename);
END;
```

PL/SQL code for trigger to prevent operations on EMP table on Sunday:

```
CREATE OR REPLACE TRIGGER prevent_sunday_operations
BEFORE INSERT OR UPDATE OR DELETE ON emp
BEGIN
    IF TO_CHAR(SYSDATE, 'DAY') = 'SUNDAY' THEN
        RAISE_APPLICATION_ERROR(-20001, 'No operations allowed on Sundays');
    END IF;
END;
/
```

PL/SQL code for trigger to enforce commission to be less than or equal to salary:

```
CREATE OR REPLACE TRIGGER commission_check
BEFORE INSERT OR UPDATE ON EMP
FOR EACH ROW
BEGIN
    IF :new.COMMISSION > :new.SALARY THEN
        RAISE_APPLICATION_ERROR(-20001, 'Commission cannot be greater than salary.');
    END IF;
END;
/
```

Trigger to implement the primary key constraint on column ENO of table EMP:

```
CREATE OR REPLACE TRIGGER eno_pk
BEFORE INSERT ON EMP
FOR EACH ROW
DECLARE
    l_cnt NUMBER;
BEGIN
    SELECT COUNT(*)
    INTO l_cnt
    FROM EMP
    WHERE ENO = :NEW.ENO;
    IF l_cnt > 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Duplicate ENO not allowed');
    END IF;
END;
/
```

Trigger to implement foreign key constraint on DEPTNO column of EMP table:

```
CREATE OR REPLACE TRIGGER deptno_fk
BEFORE INSERT OR UPDATE ON EMP
FOR EACH ROW
DECLARE
    l_cnt NUMBER;
```

```
BEGIN
  SELECT COUNT(*)
  INTO l_cnt
  FROM DEPT
  WHERE DEPTNO = :NEW.DEPTNO;
  IF l_cnt = 0 THEN
    RAISE_APPLICATION_ERROR(-20001, 'Invalid DEPTNO');
  END IF;
END;
/
```

[Colab paid products](#) - [Cancel contracts here](#)

