INTRODUCTION

Different types of commands in SQL:

A). **DDL commands: -** To create a database objects.

1. The Create Table Command.
2. Modifying the structure of tables.
3. Dropping a column from a table.
4. Modifying existing columns.
5. Renaming :
   - i) renaming the table.
   - ii) renaming the column.
6. Truncating the tables.
7. Destroying tables.

B). **DML commands: -** To manipulate data of a database objects.

1. Inserting Data into Tables.
2. Delete operations.
3. Updating the contents of a table.
4. Types of data constrains.
   - i) not null constraint at column level.
   - ii) unique constraint.
   - iii) unique constraint at table level.
   - iv) primary key constraint at column level
   - v) primary key constraint at table level.
   - vi) foreign key constraint at column level.
   - vii) foreign key constraint at table level
   - viii) check constraint
   - ix) check constraint constraint at table level.

C). **DQL command: -** To retrieve the data from a database.

1. Viewing data in the tables.
2. Filtering table data
   - i) Selected columns and all rows.
   - ii) Selected rows and all columns.
   - iii) Selected columns and selected rows.
3. Sorting data in a table.

D). **DCL/TCL commands: -** To control the data of a database.

1. Grant privileges using the GRANT statement
2. Revoke permissions using the REVOKE statement

Assignment-2

COMMANDS:

1. create table
2. insert into
3. Alter table --- add, rename, modify columns
4. Update ---using & not using where
5. Select--- all columns and some columns
6. Delete ---all rows and some rows
7. Rename ---table
8. drop table
9. FLASHBACK TABLE
10. drop table customer2 purge

11. copy the structure and data from another table
12. copy only table without data
13. insert data from other table
14. delete, drop and truncate

```
CREATE TABLE customer (
   cust_id varchar2(3),
   cust_fname VARCHAR2(15),
   cust_lname VARCHAR2(15),
   territory VARCHAR2(15),
   cred_lmt NUMBER(5),
   mngr_id NUMBER,
   marital_status VARCHAR2(10),
   sex CHAR(1),
   income NUMBER(5)
);
```

customer (cust_id ,cust_fname ,cust_lname ,territory ,cred_lmt, mngr_id , marital_status ,sex ,income)

1. Create and insert given data in table customer.
2. Alter table and Add column stay_from_year
```
alter table customer add stay_from_year number(4);
```
3. Set value of stay_from_year as 2001 for Italy/ America and 2003 otherwise
```
update customer set stay_from_year = 2001 where territory = 'Italy';
update customer set stay_from_year = 2003 where territory = 'America';
```
4. Display credit limit attribute for America
```
select cred_lmt from customer where territory = 'America';
```
5. Delete the record corresponding to Meg Sen
```
delete from customer where cust_fname = 'Meg' and cust_lname = 'Sen';
```
6. Show all attributes for Italy// Show all data in the territory Italy
```
select * from customer where territory = 'Italy';
```
7. If territory is India and status is Single set value of credit to 7000
```
update customer set cred_lmt = 7000 where territory = 'India' and marital_status = 'Single';
```
8. Rename cust_fname to first_name
```
alter table customer rename column cust_fname to first_name;
```
9. Rename cust_lname to last_name
```
alter table customer rename column cust_lname to last_name;
```
10. Create table cust1 from the old table customer(copy structure as well as data using CTAS statement).
```
create table cust1 as select * from customer;
```
11. Create tables cust2 without values of cust1 using CTAS statement.
```
create table cust2 as select * from customer where 1=0;
```
12. Create tables cust3 with attributes cust_id,cust_fname,cust_lname,income from old customer table only 5 rows.(using CTAS statement).
```
create table cust3 as select cust_id, first_name as cust_fname, last_name as cust_lname, income from customer where rownum <= 5;
```
13. Create tables cust4 with attributes name customer_id,firstname,lastname, income from old **customer** table(using CTAS statement).
```
CREATE TABLE cust4 AS SELECT cust_id AS customer_id, first_name AS firstname, last_name AS lastname, income FROM customer;
```
14. Drop column income from cust1.
```
alter table cust1 drop column income;
```
15. Rename table cust1 to cust_one
```
rename cust1 to cust_one;
```
16. Insert values into cust2 table from customer table
```
insert into cust2 select * from customer;
```
17. Insert values into cust3 table with attributescust_ id, f_name, l_name,Income from customer table where income > 50000
```
insert into cust3 select cust_id, first_name, last_name, income from customer where income > 50000;
```
18. alter the table cust4 change cust id to varchar(6) and income to number(5)
```
alter table cust4 modify (customer_id varchar(6), income number(5));
```
19. Add new attribute mngr_name to cust4 and insert 5 records
```
alter table cust4 add mngr_name varchar2(15);
```
20. Add attribute territory to cust4
```
alter table cust4 add territory varchar2(15);
```
21. Drop table cust3 and then bring it back.
```
drop table cust3;
flashback table cust3 to before drop;
```
22. Increase the size of the column custid by 5.
```
alter table cust3 modify cust_id varchar2(5);
```
23. Suppose the customer with id no C63 has changed her last name & now it is just same as the customer with id no C68.
```
update cust4 set lastname = (select lastname from cust4 where customer_id = 'C68') where customer_id = 'C63';
```
24. ~~Update customer set lname=(select lname from customer where cid=C63) where cid=C68.~~
25. Display the records where territory=America & crd_lmt=25000.
```
select * from customer where territory = 'America' and cred_lmt = '25000';
```
26. Display the records of all Indian customers whose income>20000.
```
select * from customer where territory = 'India' and cred_lmt >= '25000';
```
27. Display the name of the customer having crd_lmt between 2000 and 7000.
```
select first_name || ' ' || last_name as name from customer where cred_lmt between 2000 and 7000;
```
28. Display the records of the customers having income 20000,24000,300,4500 using only one query.
```
select * from customer where income in (20000,24000,300,4500);
```
29. Display the records in ascending order of first name
```
select * from customer order by first_name;
```
30. Display the records in descending order of income.
```
select * from customer order by income desc;
```
31. Insert a duplicate record and display all the records.
```
insert into customer select * from customer where cust_id = 'C71';
```
32. Suppose your friend wants to select a name from the names of the customers. Show the different names of the student.
```
select distinct first_name || ' ' || last_name as full_name from customer;
```

```
create table cust_100 (
emp_id number(3) primary key,
first_name varchar2(10) check (first_name = initcap(first_name)),
last_name varchar2(10) not null check (last_name = initcap(last_name)),
e_mail varchar2(20) check (e_mail = UPPER(e_mail)),
ph_no varchar2(15),
--hire_date Date check (hire_date > '01-Jan-1980'),
job_id varchar2(20) check (job_id LIKE 'FI%' OR job_id LIKE 'AD%' OR job_id LIKE 'IT%'),
salary number(8,2) check (salary >= 4000 and salary <= 25000),
mgr_id number(3),
dept_id number(3),
constraint fk_dept foreign key(dept_id) references dept(dept_id));
```

Assignment-3

Create table **dept** with the following attributes: `create table dept (dept_id number(3) primary key, dept_name varchar2(15));`

Column name	Data type(size)	Constraints
dept_id	number(3)	primary key
dept_name	varchar2(15)	------

Insert 4 depts with names and id's 90, 69, 100 and 110.

Create table **cust_100** with the following attributes:

Column name	Data type(size)	Constraints
emp_id	number(3)	Primary key
first_name	varchar2(10)	Initial letter capital
last_name	varchar2(10)	Initial letter capital and not null
e_mail	varchar2(20)	All upper case
ph_no	varchar2(15)	------
hire_date	Date	Should be > than 01-jan-1980
job_id	varchar2(10)	Must begin with FI or AD or IT
salary	number(8,2)	≥ 4000 & ≤ 25000
mgr_id	number(3)	------
dept_id	number(3)	Foreign key, refer table dept

1. Add 10 records to cust_100
2. Drop column mrg_id `alter table cust_100 drop column mgr_id;`
3. Add column mgr_id and make it self referenced such that first 4 id's correspond to first emp_id, next 4 correspond to fifth emp_id and the last 2 correspond to the ninth emp_id.

```
alter table cust_100 add mgr_id number(3);
alter table cust_100 add constraint fk_mgr_empid foreign key (mgr_id) references cust_100(emp_id);
update cust_100 set mgr_id = 101 where emp_id >= 101 and emp_id <= 104
update cust_100 set mgr_id = 105 where emp_id >= 105 and emp_id <= 108;
update cust_100 set mgr_id = 109 where emp_id >= 109 and emp_id <= 110;
```

Assignment-4

1. Oracle table DUAL
2. Types of functions
3. Numeric Functions
(ABS,POWER,ROUND,SQRT,EXP,GREATEST,LEAST,MOD,TRUNC,FLOOR,CEIL)
4. String Functions
(LOWER,UPPER,SUBSTR,INSTR,LPAD,RPAD,TRIM,LTRIM,RTRIM,LENGTH,INITCAP, SOUNDEX)
5. Conversion Functions
(TO_CHAR, TO_NUMBER,TO_DATE)
6. Date Functions
(SYSDATE, SYSTIMESTAMP, ADD_MONTHS, LAST_DAY, MONTHS_BETWEEN, NEXT_DAY, ROUND)

Create the following tables with the data types and constraints

sailor:

Attribute	Datatype	Constraints
SID	**varchar2(4)**	**primary key and start with small s**
SNAME	**varchar2(15)**	**initial letter capital**
MNAME	**varchar2(15)**	

```
create table sailor (
sid varchar2(4) primary key check (sid like 's%'),
sname varchar2(15) check (sname = initcap(sname)),
mname varchar2(15)
);
```

SURNAME	**varchar2(15)**	**not null**
RATING	**number(2)**	**default zero**
AGE	**number(3,1)**	**not null**

Attribute	Datatype	Constraints
BID	**number(3)**	**primary key & start with small b**
BNAME	**varchar2(10)**	**all upper case**
COLOR	**varchar2(6)**	**red,green,blue**

```
create table boat (
bid number(3) primary key,
bname varchar2(10) check (bname = upper(bname)),
color varchar2(6) check (color in ('red','green','blue'))
);
```

```
create table reserved (
sid varchar2(4),
bid number(3),
day date check (day < '1-Jan-2000'),
constraint pk_reserved primary key (SID, BID),
constraint fk_reserved_sailor foreign key (sid) references sailor(sid),
constraint fk_reserved_boat foreign key (bid) references boat(bid)
);
```

Attribute	Datatype	Constraints
SID	**varchar2(4)**	**Foreign key referencing sailor**
BID	**number(3)**	**Foreign key referencing boat**
DAY	**date**	**<1-JAN-2000**
SID,BID		**composite primary key**

SID	SNAME	MNAME	SURNAME	RATING	AGE
s22	Fredrico		Roberts	7	45
s31	Lubber		Sheen	8	55.5
s32	Charlotte		Gordin	8	25.5
s58	Mary	Beth	Lyon	10	35
s64	Horatio		Powell	7	35.4
s71	Zorba		Alex	10	16
s29	Brutus		Slater	1	33.8
s95	Deep	Graceb	Davis	3	63.5
s74	Horatio		Forrest	9	35
s85	Sara	Art	Powell	3	25.5
s80	Deep	Kumar	Kumar	6	17
s87	Deep	Kumar	Jha	8	51

SID	BID	R_DAY
s22	101	10-OCT-98
s22	103	10-AUG-98
s22	102	10-OCT-98
s22	104	10-JUL-98
s31	102	11-OCT-98
s31	102	11-JUN-98
s31	104	11-DEC-98
s64	101	09-MAY-98
s64	102	09-AUG-98
s74	103	09-AUG-98
s80	102	07-JUL-98
s87	101	08-JUL-98
s87	102	12-DEC-98

Q24. SELECT SNAME || ' ' || MNAME || ' ' || SURNAME AS NAMES_OF_RESERVED_SAILORS FROM SAILOR WHERE SID IN (SELECT SID FROM RESERVED WHERE BID IN (SELECT BID FROM BOAT WHERE COLOR='red')) MINUS SELECT SNAME || ' ' || MNAME || ' ' || SURNAME AS NAMES_OF_RESERVED_SAILORS FROM SAILOR WHERE SID IN (SELECT SID FROM RESERVED WHERE BID IN (SELECT BID FROM BOAT WHERE COLOR='green'));

BID	BNAME	COLOR
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Q28. SELECT SNAME || ' ' || MNAME || ' ' || SURNAME AS NAMES_OF_SAILORS FROM SAILOR WHERE SID IN (SELECT SID FROM RESERVE WHERE BID IN (SELECT BID FROM BOAT)) MINUS SELECT SNAME || ' ' || MNAME || ' ' || SURNAME AS NAMES_OF_SAILORS FROM SAILOR WHERE SID IN (SELECT SID FROM RESERVE WHERE BID IN (SELECT BID FROM BOAT WHERE COLOR='red'));

Assignment 4 PART A

For the above schema, perform the following query –

1) Find the names and ages of all sailors. select sname || ' ' || mname || ' ' || surname as fullname, age from sailor;
2) Show names under the heading of names_of_sailors and add 2 to age. select sname || ' ' || mname || ' ' || surname as names_of_sailors, age+2 as age from sailor;
3) Select all records from sailors in ascending order by name; select * from sailor order by sname, mname, surname;
4) Show all sailors name. select sname from sailor;
5) Select all distinct sailors name. select distinct sname from sailor;
6) Show all distinct sailors names, ratings who have rating between 5 and 10. select distinct sname, rating from sailor where rating between 5 and 10;
7) Select all records from sailors in ascending order by rating and descending order by age. select * from sailor order by rating, age desc;
8) Select all records from sailors whose rating>7. select * from sailor order by rating, age desc;
9) Find records for sailor name Horatio and age=35.4. select * from sailor where sname = 'Horatio' and age = '35.4';
10) Find records for sailor name Horatio or age=35.4. select * from sailor where sname = 'Horatio' or age = '35.4';
11) Select names of sailors who have reserved boat 104. select sname from sailor s join reserved r on r.sid = s.sid and r.bid = 104;
12) Find sid of sailors who have reserved red boat select sid from reserved r join boat b on b.bid = r.bid and b.color = 'red';
13) Select records for name beginning with 'B'. select * from sailor where sname like 'B%';
14) Select records for name containing 'B'/'b'. select * from sailor where sname like '%B%' or sname like '%b%';
15) Select names for rating present. select sname from sailor where rating is not null;
16) Select names for rating absent. select sname from sailor where rating is null;
17) Find color of boats reserved by Lubber. select distinct color from boat b, reserved r, sailor s where s.sid = r.sid and r.bid = b.bid and s.sname = 'Lubber';
18) Find a sailor name that have reserved at least one boat. select sname from sailor s join reserved r on r.sid = s.sid group by sname having count(*) > 0;
19) Compute the increments of rating of persons who have sailed on diff boats on the same day. select sname from sailor s join reserved r on r.sid = s.sid group by s.sname, r.day having count(distinct r.bid) > 1 and count(distinct r.day) = 1;
20) Find name of sailors whose name begins and ends with B and has at least 3 characters. select sname from sailor where sname like 'B%b' or sname like 'b%B' or sname like 'B%B' or sname like 'b%b';
21) Find names of sailors whose name begins and ends with 'B' and has exactly 3 chars. select sname from sailor where sname like 'B_b' or sname like 'b_B' or sname like 'B_B' or sname like 'b_b';
22) Find names of sailors who have reserved a red boat or a green boat. select distinct sname || ' ' || surname as name from sailor s, boat b, reserved r where s.sid = r.sid and b.bid = r.bid and (b.color = 'red' or b.color = 'green');
23) Find names of sailors who have reserved a red boat but not a green boat.
24) Find names of sailors who have reserved boat 103. select sname || ' ' || mname || ' ' || surname as fullname from sailor where sid in (select sid from reserved where bid = 103);
25) Find names of sailors who have reserved red boat. select sname || ' ' || mname || ' ' || surname as fullname from sailor where sid in (select sid from reserved r join boat b on b.bid = r.bid and b.color = 'red');
26) Find names of sailors who have not reserved red boat. Q.28 SELECT SNAME || ' ' || MNAME || ' ' || SURNAME AS NAMES_OF_SAILORS FROM SAILOR WHERE SID IN (SELECT SID FROM RESERVE WHERE BID IN (SELECT BID FROM BOAT)) MINUS SELECT SNAME || ' ' || MNAME || ' ' || SURNAME AS NAMES_OF_SAILORS FROM SAILOR WHERE SID IN (SELECT SID FROM RESERVE WHERE BID IN (SELECT BID FROM BOAT WHERE COLOR='red'));
28) Find names of sailors who have not reserved red boat.
29) Count distinct sailor name from sailors. select count(distinct sname) from sailor;
30)a) Find all records for the rating>some sailor name where sailor name like 'Horatio'. SELECT * FROM SAILOR WHERE RATING>SOME(SELECT RATING FROM SAILOR WHERE SNAME='Horatio');
b) Find all records for the rating>all sailor name where sailor name like 'Horatio'. SELECT * FROM SAILOR WHERE RATING>ALL(SELECT RATING FROM SAILOR WHERE SNAME='Horatio');
32(a)Find all records for the rating<some sailor name where sailor name like 'Horatio'
(b)Find all records for the rating<all sailor name where sailor name like 'Horatio'. Similiar like previous one
32) Select all records for s_name neither Lubber nor Horatio. select * from sailor where sname not in ('Lubber', 'Horatio');
33) Find names of sailors whose rating is>10/20/30 using multirow subquery operator. SELECT SNAME || ' ' || MNAME || ' ' || SURNAME AS NAMES_OF_SAILORS FROM SAILOR WHERE RATING>ANY(7,8,9);
34) Find names of sailors whose rating is>10 & 20 & 30 using multirow subquery operator. SELECT SNAME || ' ' || MNAME || ' ' || SURNAME AS NAMES_OF_SAILORS FROM SAILOR WHERE RATING>ALL(7,8,9);
35) Find average age of sailors with rating 10. select avg(age) from sailor where rating = 10;
36) Find the name of sailor who are older than oldest sailor of rating=10. select * from sailor where age > (select max(age) as age from sailor where rating = 10);
37) Find the age of youngest sailor for each rating level. select min(age) from sailor group by rating;
38) Find the name of each sailor who is eligible to vote for each rating level. SELECT SNAME || ' ' || MNAME || ' ' || SURNAME AS NAMES_OF_SAILORS FROM SAILOR WHERE AGE>18 GROUP BY RATING;
39) Find the age of youngest sailor who is eligible to vote for each rating level with at least two such sailors. select min_age from (select rating, min(age) as min_age, count(*) as total_count from sailor group by rating having min(age) >= 18) where total_count > 1 and rating is not null;

40) Find the average age of sailor for each rating level with at least two such sailor. select rating, avg(age) from sailor group by rating having count(*) > 1;
41) For each red boat count the no of reservations for this boat. select b.bid, count(*) from reserved r join boat b on b.bid = r.bid and b.color = 'red' group by b.bid;
42) Find sailor with highest rating. select sname || ' ' || mname || ' ' || surname as name_of_sailor from sailor where rating = (select max(rating) from sailor);
44) Find those rating for which the average age of sailors is minimum over all rating.
45) Find sailors who have reserved all boats. SELECT rating FROM sailor GROUP BY rating HAVING AVG(age) = (SELECT MIN(average_age) FROM (SELECT AVG(age) AS average_age FROM sailor GROUP BY rating));
select * from sailor where sid in (select s.sid from sailor s, reserved r, boat b where s.sid = r.sid and b.bid = r.bid group by s.sid having (count(distinct b.bid) >= (select count(*) from boat)));

ASSIGNMENT 4 PART B

46) Display s_name with left side padding by at least 3 *. SELECT LPAD(SNAME, LENGTH(SNAME)+3, '*') AS padded_s_name FROM sailor;
47) Display length of each name. select length(sname) from sailor;
48) Display all sailors names in uppercase. select upper(sname) from sailor;
49) Display all sailors' names in lower case. select lower(sname) from sailor;
50) Display all sailors names in sentence case. select initcap(sname) from sailor;
51) Display 4th to 7th letter of sailors name. select substr(sname, 4, 4) as subname from sailor;
52) Display 4th and 7th letter of sailors name. select substr(sname, 4, 1) as forth_letter, substr(sname, 7, 1) as seventh_letter from sailor;
53) Concat s_id, s_name. select sid || sname from sailor;
54) Display square root of rating. select sqrt(rating) from sailor;
55) Display floor values of all ages. select floor(age) from sailor;
56) Display ceiling values of all ages. select ceil(age) from sailor;
57) Select all s_name with 1st 2 letters off. select substr(sname, 3) as converted from sailor;
58) List months between today and reservation date. select months_between(sysdate, day) as total_monnths from reserved;
59) Select day between today and reservation date. SELECT ABS(ROUND(SYSDATE-DAY)) AS DAYS_FROM_CURR_TO_RESV FROM RESERVED;
60) Shift all reservation day by 2 months. SELECT DAY,ADD_MONTHS(DAY,2) AS SECOND_NEXT_MONTH FROM RESERVED;
61) Shift all reservation day earlier by 3 months. SELECT DAY,ADD_MONTHS(DAY,-3) AS THIRD_PREV_MONTH FROM RESERVED;
62) Suppose after sailing they enjoy their next Monday as holiday. Find that day. SELECT NEXT_DAY(DAY, 'MONDAY') AS NEXT_HOLIDAY FROM reserved;
63) Display 3 * before and after each s_name. select lpad(rpad(sname, length(sname)+3, '*'), length(sname)+6, '*') as sname from sailor;
64) Find the date when sailing ends. select max(day) as sailing_end_day from reserved;
65) Display all reservation day. select distinct day from reserved;
66)Find the position of 'Kumar' in the sailors name. select instr(sname||mname||surname, 'Kumar') as kumar_pos from sailor;
67)display all saiors name order by its length. select sname || ' ' || mname || ' ' || surname as fullname from sailor order by length(fullname);
68)display sid,sname and availability of middle name which pint as 'available' or 'not available'.
69)display all reservation day like '22nd March twenty ten' and 12/09/1998.
70)find the day of weekdays of reservation date. select to_char (day, 'DAY') as day_of_week from reserved;
71)find the number of day of weekdays of reservation date. select to_char(day, 'D') from reserved;
72) Find the number of days passed upto reservation date of that year. select to_char(day, 'DDD') as day_passed from reserved;
73) Display the number of weeks of the year for reservation day. select to_char(day, 'WW') as week_no from reserved;

68. SELECT SID,SNAME,NVL2(MNAME,'AVAILABLE','NOT AVAILABLE') as availability FROM SAILOR;
69. i) SELECT TO_CHAR(DAY,'DDTH Month year') as day FROM RESERVED;
ii) SELECT TO_CHAR(DAY,'DD/MM/YYYY') FROM RESERVED;

Adding Foreign Key Constraints

```
alter table employee add constraint fk_emp_self foreign key(super_ssn) references employee;
alter table employee add constraint fk_emp foreign key(dno) references department;
alter table dept_locations add constraint fk_dept foreign key(dnumber) references department;
alter table project add constraint fk_proj foreign key(dnum) references department;
alter table works_on add constraint fk_work_ssn foreign key(essn) references employee;
alter table works_on add constraint fk_work_pno foreign key(pno) references project;
alter table dependent add constraint fk_depend foreign key(essn) references employee;
```


Assignment-5

```
create table employee (fname varchar2(20), minit varchar2(20), lname varchar2(20), ssn number primary key, bdate date, address
varchar2(50), sex char(1), salary number, superssn number, dno number);
```

EMPLOYEE

FNAME	MINIT	LNAME	**SSN**	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO

```
create table department (dname varchar2(20), dnumber number primary key, mgrssn number, mgrstartdate date);
```

DEPARTMENT

DNAME	**DNUMBER**	MGRSSN	MGRSTARTDATE

```
create table dept_locations (dnumber number, dlocation varchar2(50));
alter table dept_locations add constraint pk primary key(dnumber,dlocation);
```

DEPT_LOCATIONS

DNUMBER	**DLOCATION**

```
create table project (pname varchar2(50), pnumber number primary key, plocation varchar2(50), dnum number);
```

PROJECT

PNAME	**PNUMBER**	PLOCATION	DNUM

```
create table works_on (essn number, pno number, hours number);
alter  table works_on add constraint pk_works primary key(essn,pno);
```

WORKS_ON

ESSN	**PNO**	HOURS

```
create table dependent (essn number, dependent_name varchar2(50), sex char(1), bdate date, relationship varchar2(20));
alter  table dependent add constraint pk_dependent primary key(essn,dependent_name);
```

DEPENDENT

ESSN	**DEPENDENT_NAME**	SEX	BDATE	RELATIONSHIP

For value insertion
https://github.com/tolgahanakgun/Elmasri-Database/blob/master/Employee_Database_Script_OracleLive.sql

For the above schema, perform the following query –

1. For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birth date.
2. Retrieve the name of each employee who works on all *the* projects controlled by department number 5.
3. Make a list of all project numbers for projects that involve an employee whose last name is 'Smith', either as a worker or as a manager of the department that controls the project.
4. Retrieve the names of employees who have no dependents.
5. List the names of managers who have at least one dependent.
6. For each employee, retrieve the employee's first and last name and the first and last name of his or her immediate supervisor.
7. Show the resulting salaries if every employee working on the 'ProductX' project is given a 10 percent raise..
8. Retrieve a list of employees and the projects they are working on, ordered by department and, within each department, ordered alphabetically by last name, first name.
9. Retrieve the names of all employees who do not have supervisors.
10. Retrieve the name of each employee who has a dependent with the same last name as the employee.
11. Retrieve the social security numbers of all employees who work on project numbers 1,2.
12. Returns the names of employees whose salary is greater than the salary of all the employees in department 5:
13. Find the sum of the salaries of all employees, the maximum salary, the minimum salary, and the average salary.
14. Find the sum of the salaries of all employees of the 'Research' department, as well as the maximum salary, the minimum salary, and the average salary in this department.
15. Retrieve the names of all employees who have two or more dependents.
16. Count the *total* number of employees whose salaries exceed $40,000 in each department, but only for departments where more than five employees work.
17. For each project, retrieve the project number, the project name, and the number of employees who work on that project.
18. For each project on *which more than two employees work*, retrieve the project number, the project name, and the number of employees who work on the project.
19. For each project, retrieve the project number, the project name, and the number of employees from department 5 who work on the project.
20. For each department that has more than five employees, retrieve the department number and the number of its employees who are making more than $40,000.

Assignment-6

A) Create the following Table: (ID number(3),Name varchar2(20),location varchar2(10)

Value of ID should be Auto generated(using Sequence) create sequence tourist_seq start with 105 increment by 1;

create table tourist (id number(3), name varchar2(20), location varchar2(15));

ID	NAME	LOCATION
108	Ram	Kolkata
107	Kaushik	Naihati
106	Subhendu	Narayanpara
105	Arti	Bhatpara

CREATE TABLE tour (
tour_id VARCHAR2(3) CHECK (tour_id LIKE 'T%'),
tour_spot VARCHAR2(10),
fare NUMBER(5),
type VARCHAR2(10) CHECK (type IN ('Delux', 'General'))
);

B)Create TOUR table

TOUR_ID	TOUR_SPOT	FARE	TYPE
TI	Gangtok	6000	Delux
T2	Puri	2000	General
T3	Nainital	9000	Delux
T4	Andaman	12000	General
T5	Madurai	7000	Deux

Express the following SQL: create view tourview as select tour_spot, fare from tour where type = 'Delux';

1) Create a view TOURVIEW for deluxe type tour containing two fields, tour-spotnames and fares.
2) Find all the Tour spots for fare greater than 8000 and Delux type tour from a) TOUR table, b) TOURVIEW view select tour_spot, fare from tour where type = 'Delux' and fare > 8000;
select tour_spot, fare from tourview where type = 'Delux' and fare > 8000;
3) insert three rows to TOURVIEW.the location will be Bangladesh, Delhi, Hyderabad.
4)Display these Records.Are they seen in TOURVIEW? Are they seen in TOUR Table?
5)Make them Seen in TOURVIEW. inserting in tourview view automatically insert those values to tour table but as the type of the tour hasn't been set durign insertion that's why it will be not shown in the view but in tour table it will be shown.
6)Create a view Tour_type_num containing two fields i.e. tour_type and its total number of count.
7) try to insert records via this view. Errors!!,Explain why? Error: ORA-01733: virtual column not allowed here
to make them seen in the tourview we need to update the tour table accordingly and set their type to 'Delux';

create view tour_type_num as select type, count(*) as total_count from tour group by type;

Assignment-7 (for practice)

Customer1(**Cust id : integer,** cust_name: string) I am creating as customer1 as in my database customer table is already present.
Item(**item_id: integer**, item_name: string, price: integer)
Sale(**bill_no: integer**, bill_data: date, **cust_id: integer, item_id: integer**, qty_sold: integer)

For the above schema, perform the following—

a) Create the tables with the appropriate integrity constraints
b) Insert around 10 records in each of the tables
c) List all the bills for the current date with the customer names and item numbers select bill_date, cust_name, item_id from sale s join customer1 c on c.cust_id = s.cust_id;
d) List the total Bill details with the quantity sold, price of the item and the final amount select qty_sold, price, qty_sold*price as final_amount from sale s join item i on i.item_id = s.item_id;
e) List the details of the customer who have bought a product which has a price>200 select c.cust_id, c.cust_name from customer1 c, sale s, item i where c.cust_id = s.cust_id and s.item_id = i.item_id and price > 200;
f) Give a count of how many products have been bought by each customer select cust_id, count(item_id) from sale group by cust_id;
g) Give a list of products bought by a customer having cust_id as 5 Select item_name from item i join sale s on s.item_id = i.item_id where cust_id = 5;
h) List the item details which are sold as of today select i.item_id, i.item_name, i.price, s.qty_sold from sale s join item i on i.item_id = s.item_id where s.bill_date = to_char(sysdate);
i) Create a view which lists out the bill_no, bill_date, cust_id, item_id, price, qty_sold, amount create view item_details as select s.bill_no, s.bill_date, s.cust_id, s.item_id, i.price, s.qty_sold, i.price*s.qty_sold as amount from sale s join item i on i.item_id = s.item_id;
j) Create a view which lists the daily sales date wise for the last one week create view last_week_sale as select * from item_details where bill_date >= trunc(sysdate) - 7 order by bill_date;

create table customer1 (cust_id number primary key, cust_name varchar2(20));
create table item (item_id number primary key, item_name varchar2(20), price number);
create table sale (bill_no number primary key, bill_date date, cust_id number, item_id number, qty_sold number);
alter table sale add constraint fk_cust_id foreign key (cust_id) references customer1(cust_id);
alter table sale add constraint fk_item_id foreign key (item_id) references item(item_id);

Assignment-8

1) Write a PL/SQL program that takes marks as input and displays grade using if-else ladder
2) Write a PL/SQL program to display all even numbers up to a number using simple loop
3) Write a PL/SQL program to find the factorial of a number using For loop
4) Write a PL/SQL program to display the Fibonacci series upto 20000 using While loop
5) Write a PL/SQL program to insert random numbers in a table random_num (rand_no number(20))
6) Write a PL/SQL program to fill up the table sphere(rad number(2),area number(10,2), volume number(15,2)) with radius values from 1 – 20
7) Write a PL/SQL program to display name, age and rating from table sailor from user input s_id

Practice assignment

8) Write a PL/SQL program to display all multiples of 3 up to a number using while loop
9) Write a PL/SQL program to display multiples of 5 using for loop
10) Write a PL/SQL program to calculate the salary from basic pay
11) Write a PL/SQL program to STRING REVERSE.
12) Write a PL/SQL program to find LEAP YEAR.

Assignment-9

1) Take s_id as input from keyboard and display the corresponding record. If s_id is not present in the sailor table, then raise the proper predefined exception.

2) Take b_name as input from keyboard & show the corresponding boat table. If more than one record satisfy for the particular b_name, raise the proper predefined exception.

3) Take basic as input from keyboard and da=60% of basic, hra=15% of basic, interim=35% of (basic + da). Display da, hra, interims & total. If interim is between 3000 and 5000, then total=2*basic. If interim is greater than 5000, then total=3*basic. (User defined exception).

Assignment-10

1) Create a PL/SQL procedure to calculate factorial of a number
2) Create a PL/SQL procedure that takes a user-input date and then prints if the year is a leap year.
3) Create a PL/SQL procedure to find prime number.
4) Create a PL/SQL procedure to STRING REVERSE.
5) Create a PL/SQL function that takes a user-input date and then prints if the year is a leap year.

Practice assignment

6) Create a PL/SQL procedure to implement Fibonacci series.
7) Write a PL/SQL procedure that takes marks as user input and prints the corresponding grade.
8) Create a PL/SQL function to calculate factorial of a number
9) Create a PL/SQL function to implement Fibonacci series.

Assignment-11

1) Create a trigger
a. This trigger is fired when an update or delete operation is performed on the table CUST_MSTR. The trigger first checks for the operation being performed on the table. Then depending on the operation being performed

, variable is assigned in the value update or delete . Previous values of the modified record of the table CUST_MSTR are inserted into the AUDIT_CUST table.

The Schema of Cust_Mstr table which stores details about customer:

CUST_MSTR(Cust_No,FName,Lname,Mname,DOB_Inc)

b. Write a database trigger on the TRANS_MSTR that checks the following:
The account number for which the transaction is being performed is a valid account number. The Transaction Amount is not zero and is positive and in case of a withdrawal the amount does not exceed the current balance for the account number.

The Schema of Trans_Mstr table which stores details about transaction:

TRANS_MSTR(Trans_no,Acct_no,Dt,type,Particular,Dr_Cr,Amt,Balance)

Assignment-12

1) From table reserve, write a PL\SQL program using cursor to show all boat id nos. reserved and their corresponding day when a sailor id is entered as user input.

2) Write a PL\SQL program to display table sailor using cursor.

3) Write a PL\SQL program where you use the table customer and for a particular user input territory value, all other attributes are displayed corresponding to that territory.

4) WRITE A PROGRAM TO FIND ALL S_ID,S_NAME,THEIR RESERVED BOAT FOR THE SAILORS WHOSE 20<AGE<55.

Practice Assignment for Home

Student(Stud_no : integer, Stud_name: string)
Membership(Mem_no: integer, Stud_no: integer)
Book(book_no: integer, book_name:string, author: string)
Iss_rec(iss_no:integer, iss_date: date, Mem_no: integer, book_no: integer)

```
create table student (stud_no int primary key, name varchar(30));
create table membership(mem_no int primary key, stud_no int);
create table book(book_no int primary key, book_name varchar2(30), author varchar2(30));
create table iss_rec(iss_no int primary key, iss_date date, mem_no int, book_no int);
alter table membership add constraint fk_stud_no foreign key (stud_no) references student (stud_no);
alter table iss_rec add constraint fk_mem_no foreign key (mem_no) references membership(mem_no);
alter table iss_rec add constraint fk_book_no foreign key (book_no) references book(book_no);
```

For the above schema, perform the following –

a) Create the tables with the appropriate integrity constraints
b) Insert around 10 records in each of the tables
c) List all the student names with their membership numbers

```
select mem_no, stud_name from student s join membership m on m.stud_no = s.stud_no;
```

d) List all the issues for the current date with student and Book names

```
select stud_name, book_name from student s, book b, membership m, iss_rec i where i.mem_no = m.mem_no and m.stud_no = s.stud_no and i.book_no = b.book_no and i.iss_date = to_char(sysdate);
```

e) List the details of students who borrowed book whose author is CJDATE

```
select stud_name, book_name from student s, book b, membership m, iss_rec i where i.mem_no = m.mem_no and m.stud_no = s.stud_no and i.book_no = b.book_no and b.author='CJDATE';
```

f) Give a count of how many books have been bought by each student

```
select stud_name, count(book_name) as total_books from student s, book b, membership m, iss_rec i where i.mem_no=m.mem_no and m.stud_no=s.stud_no and i.book_no=b.book_no group by s.stud_no, stud_name;
```

g) Give a list of books taken by student with stud_no as 5

```
select book_name from book b, membership m, iss_rec i where i.book_no = b.book_no and i.mem_no = m.mem_no and m.stud_no = 5;
```

h) List the book details which are issued as of today

```
select book_name, author from book b, membership m, iss_rec i where i.book_no = b.book_no and i.mem_no = m.mem_no and i.iss_date = to_char(sysdate);
```

i) Create a view which lists out the iss_no, iss _date, stud_name, book name

```
create view issue_details as select iss_no, iss_date,
stud_name, book_name from student s, book b,
membership m, is
s_rec i where i.mem_no = m.mem_no and m.stud_no =
s.stud_no and i.book_no = b.book_no;
```

j) Create a view which lists the daily issues-date wise for the last one week

```
create view last_week_issue as select iss_date,
stud_name, book_name from student s, book b,
membership m, iss_rec i where i.mem_no = m.mem_no
and m.stud_no = s.stud_no and i.book_no = b.book_no
and iss_date >= to_char(sysdate - 7) group by iss_date,
stud_name, book_name;
```