
DREAM HOME CASE STUDY

Consider the relational schema for part of the DreamHome case study is:
Branch (branchNo, street, city, postcode)
Staff (staffNo , fName, lName , position, sex, DOB, salary, branch No)
PrivateOwner (ownerNo, fName, 'Name, address, telNo, eMail, password)
PropertyForRent(propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)
Client (clientNo, fName, lName, telNo, prefType, maxRent, eMail)
Viewing (clientNo, propertyNo, viewDate, comment)
Registration (clientNo, branchNo, staffNo, dateJoined)
NOTE : PKs and FKs underlined with continuous & dotted lines

1. **Create a database with name "DreamHome" and now create all the tables listed above with constraints.**

SQL>create database DreamHome;
SQL>create table **branch**(branchNo number(10) primary key,street varchar2(20),city varchar2(20),postcode varchar2(20));
SQL>create table **staff**(staffNo number(10) primary key,fname varchar2(10),lname varchar2(10),position varchar2(10),sex char(1), dob date,salary number(10),branchNo number(10));
SQL>create table **propertyforrent**(propertyNo number(10) primary key,street varchar2(10),city varchar2(10),postcode number(10),type varchar2(10),rooms number(5),rent number(10),ownerNo number(10),staffNo number(10) references staff(staffNo),branchNo number(10) references branch(branchNo));
SQL>create table **client**(clientNo number(10) primary key, fname varchar2(20),lname varchar2(20),telNo number(10),prefType varchar2(10),maxRent varchar2(10),email varchar2(20));
SQL>create table **PrivateOwner** (ownerNo number(10) primary key, fName varchar2(20), Name varchar2(20), address varchar2(20), telNo number(10), eMail varchar2(20), password varchar2(15));
SQL> create table **Viewing**(clientNo number(10) references client(client no), propertyNo number(10) references propertyforrent(propertyNo), viewDate date, comments varchar2(10));
SQL>create table **Registration**(clientNo number(10) references client(clientNo), branchNo number(10) references branch(branchNo), staffNo number(10) references staff(staffNo), dateJoined date);

2. **Insert a new row into the table supplying data for all columns.**

SQL>insert into branch values(&branchNo, '&street', '&city', &postcode);
SQL>insert into Staff values(&staffNo, '&fName','&lName','&position','&sex','&DOB', &salary,&branchNo);

SQL>insert into PrivateOwner values(&ownerNo,'&fName','&lName','& address', &telNo,'&eMail','&password');

SQL>insert into PropertyForRent values(&propertyNo,'&street', '&city',&postcode, '&type',&rooms,&rent,&ownerNo, &staffNo,&branchNo);

SQL>insert into Client values(&clientNo,'&fName','&lName',&telNo,'&prefType', &maxRent,'&eMail');

SQL>insert into Viewing values(&clientNo,&propertyNo,'&viewDate','&comments');

SQL>insert into Registration values(&clientNo,&branchNo,&staffNo,'&dateJoined');

Branch

branchNo	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

PropertyForRent

propertyNo	street	city	postcode	type	rooms	rent	ownerNo	staffNo	branchNo
PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46	SA9	B007
PL94	6 Argyll St	London	NW2	Flat	4	400	CO87	SL41	B005
PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40		B003
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93	SG37	B003
PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87	SG37	B003
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93	SG14	B003

Client

clientNo	fName	lName	telNo	prefType	maxRent
CR76	John	Kay	0207-774-5632	Flat	425
CR56	Aline	Stewart	0141-848-1825	Flat	350
CR74	Mike	Ritchie	01475-392178	House	750
CR62	Mary	Tregear	01224-196720	Flat	600

PrivateOwner

ownerNo	fName	lName	address	telNo
CO46	Joe	Keogh	2 Fergus Dr, Aberdeen AB2 7SX	01224-861212
CO87	Carol	Farrel	6 Achray St, Glasgow G32 9DX	0141-357-7419
CO40	Tina	Murphy	63 Well St, Glasgow G42	0141-943-1728
CO93	Tony	Shaw	12 Park Pl, Glasgow G4 0QR	0141-225-7025

Viewing

clientNo	propertyNo	viewDate	comment
CR56	PA14	24-May-04	too small
CR76	PG4	20-Apr-04	too remote
CR56	PG4	26-May-04	
CR62	PA14	14-May-04	no dining room
CR56	PG36	28-Apr-04	

Registration

clientNo	branchNo	staffNo	dateJoined
CR76	B005	SL41	2-Jan-04
CR56	B003	SG37	11-Apr-03
CR74	B003	SG37	16-Nov-02
CR62	B007	SA9	7-Mar-03

3. **Modify data in the database using UPDATE**
   SQL>update <table name> set <column name>=value;
4. **Delete data from the database using DELETE**
   SQL>delete from <Table name>; (to delete all records)
   SQL>delete from <Table name> where <condition>; (to delete specific records)
5. **Changing a table definition using ALTER**
   SQL>Alter table <table name> add <column name> data type;
6. **Removing a table using DROP**
   SQL>drop table <table name>;
7. **Removing rows in table using TRUNCATE**
   SQL>truncate table <table name>;
8. **Create an index and removing an index**
   SQL>create index <index name> on table name(column name);
9. **Practice other standard SQL commands for creating, modifying, displaying data of tables.**
   SQL>select * from <table name>;
10. **List full details of all staff.**
    sql> Select * from staff;
11. **List all staff with a salary greater than £10000.**
    sql> select * from staff where salary>10000;
12. **List the property numbers of all properties that have been viewed.**
    sql> SELECT propertyNo FROM Viewing;
13. **Produce a list of salaries for all staff, showing only the staffNo, (Name, IName, and salary details.**
    sql> SELECT staffNo, fName, lName, salary/12 FROM Staff;
14. **List all cities where there is either a branch office or a property for rent. .**
    sql> (SELECT city FROM BranchWHERE city IS NOT NULL) UNION
    (SELECT cityFROM PropertyForRent WHERE city IS NOT NULL);
15. **List all cities where there is a branch office but no properties for rent.**
    sql> (SELECT city FROM Branch) EXCEPT (SELECT city FROM PropertyForRent);
16. **List all cities where there is both a branch office and at least one property for rent.**
    SQL>(select city from branch) intersect (select city from propertyforrent);
17. **List the names and comments of all clients who have viewed a property for rent.**
    SQL>SELECT c.clientNo, fName, lName, propertyNo, comment FROM Client c,
    Viewing v WHERE c.clientNo = v.clientNo;
18. **Produce a status report on property viewings.**
    SQL>select * from propertyforrent natural join viewing;
19. **List complete details of all staff who work at the branch in Glasgow.**
    SQL>SELECT staffNo, fName, lName, position FROM Staff
    WHERE branchNo =(SELECT branchNo FROM Branch WHERE street = 'glasglow');
20. **List the addresses of all branch offices in London or Glasgow**

SQL>SELECT *FROM Branch WHERE city = 'London' OR city = 'Glasgow';

21. **List all staff with a salary between £20,000 and £30,000.**
SQL>SELECT staffNo, fName, lName, position, salary FROM Staff WHERE salary BETWEEN 20000 AND 30000;

22. **Identify all clients who have viewed all properties with three rooms.**
SQL>select clientNo from viewing where propertyNo=(select propertyNo from propertyforrent where rooms=3);

23. **How many properties cost more than £350 per month to rent?**
SQL> SELECT COUNT(*) AS count FROM PropertyForRent WHERE rent > 350;

24. **How many different properties were viewed in May 2013?**
SQL>SELECT COUNT(DISTINCT propertyNo) AS count FROM Viewing
WHERE viewDate BETWEEN '1-May-17' AND '31-May-17';

25. **Find the total number of Managers and the sum of their salaries.**
SQL>SELECT COUNT(staffNo) AS count, SUM(salary) AS sum FROM Staff WHERE position = 'Manager';

26. **Find the minimum, maximum, and average staff salary.**
SQL>SELECT MIN(salary) AS min, MAX(salary) AS max, AVG(salary) AS avg
FROM Staff;

27. **Find the number of staff working in each branch and the sum of their salaries.**
SQL>SELECT branchNo, COUNT(staffNo) AS count, SUM(salary) AS sum
FROM Staff GROUP BY branchNo ORDER BY branchNo;

28. **List all managers and supervisors.**
SQL>SELECT staffNo, fName, lName, position FROM Staff WHERE position IN ('Manager', 'Supervisor');

29. **Find all owners with the string 'Glasgow' in their address.**
SQL> SELECT clientNo, fName, lName, address, telNo FROM PrivateOwner
WHERE address LIKE '%Glasgow%';

30. **List the details of all viewings on property PG4 where a comment has not been supplied.**
SQL> SELECT clientNo, viewDate FROM Viewing WHERE propertyNo = 'PG4' AND comment IS NULL;

31. **Produce a list of salaries for all staff, arranged in descending order of salary.**
SQL> SELECT staffNo, fName, lName, salary FROM Staff ORDER BY salary DESC;

32. **Produce an abbreviated list of properties arranged in order of property type.**
SQL> SELECT propertyNo, type, rooms, rent FROM PropertyForRent ORDER BY type;

33. **Find the number of staff working in each branch and the sum of their salaries.**
SQL> SELECT branchNo, COUNT(staffNo) AS count, SUM(salary) AS sum
FROM Staff GROUP BY branchNo ORDER BY branchNo;

34. **For each branch office with more than one member of staff, find the number of staff working in each branch and the sum of their salaries.**
SQL>SELECT branchNo, COUNT(staffNo) AS count,SUM(salary) AS sum FROM Staff

GROUP BY branchNo HAVING COUNT(staffNo) > 1 ORDER BY branchNo;

35. **List the staff who work in the branch at '163 Main St'.**
SQL> SELECT staffNo, fName, lName, position FROM Staff WHERE branchNo =(SELECT branchNo FROM Branch WHERE street = '163 Main St');

36. **List all staff whose salary is greater than the average salary, and show by how much their salary is greater than the average.**
SQL> SELECT staffNo, fName, lName, position, salary – (SELECT AVG(salary) FROM Staff) As SalDiff FROM Staff WHERE salary > (SELECT AVG(salary) FROM Staff);

37. **List the properties that are handled by staff who work in the branch at '163 Main St'.**
SQL> SELECT propertyNo, street, city, postcode, type, rooms, rent FROM PropertyForRent WHERE staffNo IN(SELECT staffNo FROM Staff WHERE branchNo =(SELECT branchNo FROM Branch WHERE street = '163 Main St'));

38. **Find all staff whose salary is larger than the salary of at least one member of staff at branch BOO3.**
SQL>SELECT staffNo, fName, lName, position, salary FROM Staff WHERE salary > SOME (SELECT salary FROM Staff WHERE branchNo = 'B003');

39. **Find all staff whose salary is larger than the salary of every member of staff at branch B003**
SQL>SELECT staffNo, fName, lName, position, salary FROM Staff WHERE salary > ALL(SELECT salary FROM Staff WHERE branchNo = 'B003');

40. **List the names of all clients who have viewed a property, along with any comments supplied.**
SQL>SELECT c.clientNo, fName, lName, propertyNo, comments FROM Client c, Viewing v WHERE c.clientNo = v.clientNo;

41. **For each branch office, list the staff numbers and names of staff who manage properties and the properties that they manage.**
sql> SELECT s.branchNo, s.staffNo, fName, lName, propertyNo FROM Staff s, PropertyForRent p WHERE s.staffNo = p.staffNo ORDER BY s.branchNo, s.staffNo, propertyNo;

42. **For each branch, list the staff numbers and names of staff who manage properties, including the city in which the branch is located and the properties that the staff manage.**
sql> SELECT b.branchNo, b.city, s.staffNo, fName, lName, propertyNo FROM Branch b, Staff s, PropertyForRent p WHERE b.branchNo = s.branchNo AND s.staffNo = p.staffNo ORDER BY b.branchNo, s.staffNo, propertyNo;

43. **Find the number of properties handled by each staff member, along with the branch number of the member of staff.**
SQL>SELECT s.branchNo, s.staffNo, COUNT(*) AS count FROM Staff s, PropertyForRent p WHERE s.staffNo = p.staffNo GROUP BY s.branchNo, s.staffNo ORDER BY s.branchNo, s.staffNo;

44. **List all branch offices and any properties that are in the same city.**

sql>SELECT b.*, p.* FROM Branch1 b, PropertyForRent1 p WHERE b.bCity = p.pCity;

45. **List all properties and any branch offices that are in the same city.**
sql>SELECT p.*,b.* FROM PropertyForRent1 p, Branch1 b WHERE p.pCity= b.bCity;

46. **List the branch offices and properties that are in the same city along with any • unmatched branches or properties.**
sql> SELECT b.*, p.* FROM Branch1 b LEFT JOIN PropertyForRent1 p ON b.bCity = p.pCity;

47. **Find all staff who work in a London branch office.**
sql>SELECT staffNo, fName, lName, position FROM Staff s WHERE EXISTS(SELECT * FROM Branch b WHERE s.branchNo = b.branchNo AND city = 'London');

48. **Construct a list of all cities where there is either a branch office or a property.**
sql> (SELECT city FROM Branch) UNION (SELECT cityFROM PropertyForRent);

49. **Construct a list of all cities where there is both a branch office and a property.**
sql> (SELECT city FROM Branch)INTERSECT (SELECT city FROM PropertyForRent);

50. **Create a view so that the manager at branch B003 can see the details only for staff who work in his or her branch office.**
SQL>create view manager3staff as select * from staff where branchno='b003';

51. **Create a view of the staff details at branch B003 that excludes salary information, so that only managers can access the salary details for staff who work at their branch.**
SQL>create view staff3 as select staffno,fname,iname,position,sex from staff where branchno='b003';

52. **Create a view of staff who manage properties for rent, which includes the branch number they work at, their staff number, and the number of properties they manage.**
SQL>create view staffpro(branchno,staffno,cnt) as select s.branchno,s.staffno, count(*) from staff s,propertyforrent p where s.staffno=p.staffno groupby s.branchno,s.staffno;

53. **Removing a view using DROP VIEW**
SQL>drop view <view name>;

54. **Give the user with authorization identifier Man ager all privileges on the Staff table.**
SQL>grant all privileges on staff to manager with grant option;

55. **Give users Personnel and Director the privileges SELECT and UPDATE on column salary of the Staff table.**
SQL>grant select,update(salary) on staff to personnel ,director;

56. **Revoke the privilege SELECT on the Branch table from all users.**
SQL>revoke select on branch from public;

57. **Revoke all privileges you have given to Director on the Staff table.**
SQL>revoke all privileges on staff from director;

HOTEL CASE STUDY

Consider the relational schema for part of the Hotel case study is:

Hotel (hotelNo, hotelName, city)

Room(roomNo,hotelNo, type, price)

Booking(hotelNo, guestNo, dateFrom, dateTo, roomNo)

Guest(guestNo, guestName, guestAddress)

NOTE : PKs and FKs underlined with continuous & dotted lines

1. Create a database with name "Hotel" and now create all the tables listed above with constraints.

SQL>create tableHotel(hotelNo number(10) primary key, hotelName varchar2(20), city varchar2(20));

SQL>create table Room(roomNo number(10) primary key,hotelNo number(10) references tableHotel(hotelNo), type varchar2(20), price number(10));

SQL>create table Booking(hotelNo number(10) references tableHotel(hotelNo), guestNo number(10) references Guest(guestNo), dateFrom date, dateTo date, roomNo number(10) references Room(roomNo));

SQL>create table Guest(guestNo number(10) primary key, guestName varchar2(20), guestAddress varchar2(20));

2. Insert a new row into the table supplying data for all columns

SQL>insert into Hotel values (&hotelNo, '&hotelName', '& city');

SQL>insert into Room values (&roomNo, &hotelNo, '&type', &price);

SQL>insert into Booking values (&hotelNo, &guestNo, '&dateFrom', '&dateTo', &roomNo);

SQL>insert into Guest values (&guestNo, '&guestName', '&guestAddress');

Hotel

hotelno	hotelname	city
ch01	Omni Shoreham	London
ch02	Phoenix Park	London
dc01	Latham	Berlin
fb01	Grosvenor	London
fb02	Watergate	Paris

Guest

guestno	guestname	guestaddress
10001	John Kay	56 High St, London
10002	Mike Ritchie	18 Tain St, London
10003	Mary Tregear	5 Tarbot Rd, Aberdeen
10004	Joe Keogh	2 Fergus Dr, Aberdeen
10005	Carol Farrel	6 Achray St, Glasgow
10006	Tina Murphy	63 Well St, Glasgow
10007	Tony Shaw	12 Park Pl, Glasgow

Room

roomno	hotelno	type	price
501	fb01	single	19.00
601	fb01	double	29.00
701	ch02	single	10.00
701	fb01	family	39.00
801	ch02	double	15.00
901	dc01	single	18.00
1001	ch01	single	29.99
1001	dc01	double	30.00
1001	fb02	single	58.00
1101	ch01	family	59.99
1101	dc01	family	35.00
1101	fb02	double	86.00

Booking

hotelno	guestno	datefrom	dateto	roomno
ch01	10006	Apr 21, 2004	NULL	1101
ch02	10002	Apr 25, 2004	May 6, 2004	801
dc01	10003	May 20, 2004	NULL	1001
dc01	10007	May 13, 2004	May 15, 2004	1001
fb01	10001	Apr 1, 2004	Apr 8, 2004	501
fb01	10001	May 1, 2004	NULL	701
fb01	10002	May 4, 2016	May 29, 2004	601
fb01	10004	Apr 15, 2004	May 15, 2004	601
fb01	10005	May 2, 2004	May 7, 2004	501
fb02	10003	Apr 5, 2004	Apr 4, 2010	1001
fb02	10005	May 12, 2004	May 4, 2030	1101

3. **Modify data in the database using UPDATE**
   SQL>update <table name> set <column name>=value;
4. **Delete data from the database using DELETE**
   SQL>delete from <Table name>; (to delete all records)
   SQL>delete from <Table name> where <condition>; (to delete specific records)
5. **Changing a table definition using ALTER**
   SQL>Alter table <table name> add <column name> data type;
6. **Removing a table using DROP**
   SQL>drop table <table name>;
7. **Removing rows in table using TRUNCATE**
   SQL>truncate table <table name>;
8. **Practice other standard SQL commands for creating, modifying, displaying data of tables.**
   SQL>select * from <table name>;
9. **List full details of all hotels.**
   sql>SELECT * FROM hotel;

10. List full details of all hotels in London.

sql> SELECT * FROM hotel WHERE city = 'London';

11. List the names and addresses of all guests living in London, alphabetically ordered by name.

SQL> SELECT guestname, guestaddress FROM guest WHERE guestaddress like 'London' ORDER BY guestname;

12. List all double or family rooms with a price below £40.00 per night, in ascending order of price.

SQL> SELECT * FROM room WHERE price < 40 AND type IN ('Double', 'Family') ORDER BY price;

13. List the bookings for which no dateTo has been specified.

SQL> SELECT * FROM booking WHERE dateto IS NULL;

14. How many hotels are there?

SQL> SELECT COUNT(*) FROM hotel;

15. What is the average price of a room?

SQL> SELECT AVG(price) FROM room;

16. What is the total revenue per night from all double rooms?

SQL>SELECT SUM(price) FROM room WHERE type = 'Double' ;

17. How many different guests have made bookings for August?

SQL>SELECT COUNT(DISTINCT guestno) FROM booking WHERE (datefrom <= '8/31/17' AND dateto >= '8/1/17');

18. List the price and type of all rooms at the Grosvenor Hotel.

SQL>SELECT price, type FROM room WHERE hotelno = (SELECT hotelno FROM hotel WHERE hotelname = 'Grosvenor');

19. List all guests currently staying at the Grosvenor Hotel.

SQL> SELECT (guestno, guestname, guestaddress) FROM guest, booking, hotel WHERE guest.guestno =booking.guestno AND hotel.hotelno = booking.hotelno AND(datefrom <= 'SYSTEM DATE' AND dateto >= 'SYSTEM DATE') AND hotelname = 'Grosvenor';

20. What is the total income from bookings for the Grosvenor Hotel today?

SQL> SELECT SUM(price) FROM booking b, room r, hotel hWHERE (b.datefrom <= 'SYSTEM DATE' AND b.dateto >= 'SYSTEM DATE') AND r.hotelno = h.hotelno AND r.hotelno = b.hotelno AND r.roomno = b.roomno AND h.hotelname = 'Grosvenor';

21. List the rooms that are currently unoccupied at the Grosvenor Hotel.

SQL> SELECT (r.hotelno, r.roomno, r.type, r.price) FROM room r, hotel h WHERE r.hotelno = h.hotelno AND h.hotelname = 'Grosvenor' AND roomno NOT IN (SELECT roomno FROM booking b, hotel h WHERE (datefrom <= 'SYSTEM DATE' AND dateto >= 'SYSTEM DATE') AND b.hotelno=h.hotelno AND hotelname = 'Grosvenor');

22. **What is the lost income from unoccupied rooms at the Grosvenor Hotel?**
sql> SELECT SUM(price) FROM room r, hotel h WHERE r.hotelno = h.hotelno AND h.hotelname = 'Grosvenor' AND roomno NOT IN (SELECT roomno FROM booking b, hotel h WHERE (datefrom <= 'SYSTEM DATE' AND dateto >= 'SYSTEM DATE') AND b.hotelno = h.hotelno AND r.hotelno=b.hotelno AND r.roomno=b.roomno AND h.hotelname = 'Grosvenor');

23. **List the number of rooms in each hotel.**
sql> SELECT hotelno,COUNT(roomno) FROM room GROUP BY hotelno;

24. **List the number of rooms in each hotel in London.**
sql>SELECT r.hotelno, COUNT(roomno) FROM room r, hotel h WHERE r.hotelno=h.hotelno AND city = 'London' GROUP BY r.hotelno;

25. **What is the average number of bookings for each hotel in August?**
sql> SELECT hotelno, y/31 FROM (SELECT hotelno, COUNT(hotelno) AS y FROM booking WHERE (datefrom <= '8/31/17' AND dateto >= '8/1/17' GROUP BY hotelno);

26. **What is the most commonly booked room type for each hotel in London?**
sql> SELECT type, MAX(y)FROM (SELECT type, COUNT(type) AS y FROM booking b, hotel h, room r WHERE r.roomno = b.roomno AND r.hotelno = b.hotelno AND b.hotelno = h.hotelno AND city = 'London' GROUP BY type) GROUP BY type;

27. **What is the lost income from unoccupied rooms at each hotel today?**
sql> SELECT r.hotelno, SUM(price) FROM room r WHERE NOT EXIST (SELECT * FROM booking b WHERE r.roomno = b.roomno AND r.hotelno = b.hotelno AND (datefrom <= 'SYSTEM DATE' AND dateto >= 'SYSTEM DATE')) GROUP BY hotelno;

28. **Insert rows into each of these tables.**
INSERT INTO hotel VALUES ('h11', 'hilton', 'sacramento');
INSERT INTO room VALUES ('hr1111', 'h11', 'single', 120);
INSERT INTO BOOKING('H11','G11','02-MAY-2018',07-MAY-2018','HR1111');
INSERT INTO GUEST('G11','WASEEM','JAGTIAL');

29. **Update the price of all rooms by 5%.**
sql> UPDATE room SET price = price*1.05;

30. **Demonstrate that queries written using the UNION operator and same can be rewritten using the OR .**
SQL>select * from table1 union select * from table2;
SQL>select * from table1 union select * from table2;

31. **Apply the syntax for inserting data into a table.**
SQL>insert into <table name> values (value1,value2,.....,value n);

SAILOR & BOAT CASE STUDY

Given relation schemas are
Sailors (sid, sname, rating, age)
Boats (bid, bname, color)
Reserves (sid , bid, day)
NOTE : PKs and FKs underlined with continuous & dotted lines

Creating Tables:

SQL>create table Sailors(sid number(10) primary key, sname varchar2(20), rating number(2), age number(3,2));

SQL>create table Boats(bid number(10) primary key, bname varchar2(20), color varchar2(10));

SQL>create table Reserves(sid number(10) references Sailors(sid), bid number(10) references Boats(bid), day date);

Inserting Values into Tables:

SQL>insert into Sailors values(&sid,'&sname',&rating,&age);

SQL>insert into Boats values(&bid,'&bname','&color');

SQL>insert into Reserves values(&sid ,&bid,'&day');

Boats

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Reserves

sid	bid	day
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

Sailors

sid	sname	rating	age
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

1. **Find the names and ages of all sailors.**

   SQL> SELECT sname, age FROM sailors;

2. **Find all sailors with a rating above 7.**

   SQL> SELECT S.sid, S.sname, S.rating, S.age FROM Sailors S WHERE S.rating > 7;

3. **Find the names of sailors who have reserved boat 103.**

   SQL> SELECT S.sname FROM Sailors S, Reserves R WHERE S.sid = R.Sid AND R.bid = 103;

4. **Find the sids of sailors who have reserved a red boat.**

   SQL>SELECT S.sid FROM Sailors S, Reserves R, Boats B WHERE S.sid = R.sid AND R.bid = B.bid AND B.color = 'red';

5. **Find the names of sailors who have reserved a red boat.**

   SQL>SELECT S.sid FROM Sailors S, Reserves R, Boats B WHERE S.sid = R.sid AND R.bid = B.bid AND B.color = 'red';

6. **Find the colors of boats reserved by Lubber.**

   SQL>SELECT B.color FROM Sailors S, Reserves R, Boats B WHERE S.sid = R.sid AND R.bid = B.bid AND S.sname ='Lubber';

7. **Find the names of sailors who have reserved at least one boat.**

SQL> SELECT S.sname FROM Sailors S, Reserves R WHERE S.sid = R.sid ;

8. **Find the names of sailors who have reserved at least two boats.**

SQL>select distinct s.sname from sailors s,reserves r1,reserves r2 where s.sid=r1.sid and r1.sid=r2.sid and r1.bid!=r2.bid;

9. **Find the names of sailors who have reserved a red or a green boat.**

SQL> SELECT S.sname FROM Sailors S, Reserves R, Boats B WHERE S.sid = R.sid AND R.bid=B.bid AND (B.color='red' OR B.color='green');

10. **Find the names of sailors who have reserved a red and a green boat.**

SQL> SELECT S.sname FROM Sailors S, Reserves R, Boats B WHERE S.sid = R.sid AND R.bid = B.bid AND B.color='red' UNION SELECT S2.sname FROM Sailors S2, Boats B2, Reserves R2 WHERE S2.sid = R2.sid AND R2.bid = B2.bid AND B2.color = 'green';

11. **Find the sids of all sailors who have reserved re d boats but not green boats.**

SQL> SELECT S.sname FROM Sailors S, Reserves R, Boats B WHERE S.sid = R.sid AND R.bid = B.bid AND B.color = 'red' MINUS SELECT S2.sname FROM Sailors S2, Boats B2, Reserves R2 WHERE S2.sid = R2.sid AND R2.bid = B2.bid AND B2.color= 'green';

12. **Find all sids of sailors who have a rating of 10 or have reserved boat 104.**

SQL>SELECT S.sid FROM Sailors SWHERE S.rating = 10 UNION SELECT R.sid FROM Reserves R WHERE R.bid = 104;

13. **Find the names of sailors who have not reserved a red boat.**

SQL>SELECT S.sid FROM Sailors S, Reserves R, Boats B WHERE S.sid = R.sid AND R.bid = B.bid AND B.color! = 'red';

14. **Find sailors whose rating is better than some sailor called Horatio.**

SQL> SELECT S.sid FROM Sailors S WHERE S.rating > ANY(SELECT S2.rating FROM Sailors S2 WHERE S2.sname = 'Horatio');

15. **Find sailors whose rating is better than every sailor called Horatio.**

SQL>SELECT S.sid FROM Sailors S WHERE S.rating > ALL(SELECT S2.rating FROM Sailors S2 WHERE S2.sname = 'Horatio');

16. **Find the names of sailors who have reserved all boats.**

SQL>SELECT S.sname FROM Sailors S WHERE NOT EXISTS ((SELECT B.bid FROM Boats B) MINUS (SELECT R.bid FROM Reserves R WHERE R.sid = S.sid));

17. **Find the names of sailors who have reserved at least two boats.**

SQL> SELECT DISTINCT S.sname FROM Sailors S, Reserves R1, Reserves R2 WHERE S.sid = R1.sid AND R1.sid = R2.sid AND R1.bid != R2.bid;

18. **Find the names of sailors who have reserved all boats called Interlake.**

SQL> SELECT S.sname FROM Sailors S WHERE NOT EXISTS (SELECT B.bid FROM Boats B WHERE B.bname ='Interlake' AND NOT EXISTS(SELECT R.bid FROM Reserves R WHERE R.bid = B.bid AND R.sid = S.sid));

19. **Find sailors w ho have reserved all red boats.**
SQL>select * from sailors s where NOT EXISTS(select B.bid from Boats B where B.bid NOT IN (select R.bid from reserves R where R.sid=S.sid AND B.color='green'));

20. **Find the sailor name, boat id, and reservation date for each reservation.**
SQL>select sname,bid,day from sailors S,boats B,Reserves R where S.sid=R.sid and B.bid=R.bid;

21. **Find the sids of sailors with age over 20 who have not reserved a red boat.**
SQL>select R.sid from reserves R,boats B where B.bid=R.sid and B.color='red';

22. **Find the average age of all sailors.**
SQl>SELECT AVG (S.age) FROM Sailors S;

23. **Find the average age of sailors with a rating of 10.**
SQL>SELECT AVG(S.age) FROM Sailors S WHERE S.rating = 10;

24. **Find the name and age of the oldest sailor.**
SQl>SELECT S.sname, MAX (S.age) FROM Sailors S;

25. **Count the number of different sailor names.**
SQL> SELECT COUNT (DISTINCT S.sname) FROM Sailors S;

26. **Find the names of sailors who are older than the oldest sailor with a rating of 10.**
SQL> SELECT S.sname FROM Sailors S WHERE S.age > (SELECT MAX(S2.age) FROM Sailors S2 WHERE S2.rating = 10);

27. **Find the sailors with the highest rating.**
SQL>SELECT S.sid FROM Sailors S WHERE S.rating >= ALL(SELECT S2.rating FROM Sailors S2);

28. **Find the age of the youngest sailor for each rating level.**
SQl> SELECT S.rating, MIN(S.age) FROM Sailors S GROUP BY S.rating ;

29. **Find age of the youngest sailor who is eligible to vote for each rating level with at least 2 such sailors.**
SQl> SELECT S.rating, MIN(S.age) AS minage FROM Sailors S WHERE S.age >=18 GROUP BY S.rating HAVING COUNT(*) > 1;

30. **Find the average age of sailors for each rating level that has at least two sailors.**
SQL> SELECT S.rating, AVG(S.age) AS average FROM Sailors S GROUP BY S.rating HAVING COUNT(*) > 1;

31. **For each red boat, find the n umber of reservations for this boat.**
SQL>SELECT B.bid, COUNT(*) AS sailorcount FROM Boats B, Reserves R WHERE R.bid = B.bid AND B.color = 'red' GROUP BY B.bid;

32. **Find the average age of sailors who are of voting age (i.e., at least 18 years old) for each rating level that has at least two sailors.**
SQl>SELECT S.rating, AVG(S.age) AS average FROM Sailors S WHERE S.age >=18

GROUP BY S.rating HAVING 1 <(SELECT COUNT(*) FROM Sailors S2 WHERE S.rating = S2.rating);

33. **Delete the records of sailors who have rating 8 (deleting some rows in a table).**
SQL>delete from sailors where rating=8;

34. **Loading data which is present in the text into the table.**
SQL>insert into <table name> values (value1,value2,…..,value n);