RAMAIAH
Institute of Technology

Nithya BN
Assistant Professor
Department of Computer Applications
M S Ramaiah Institute of Technology
nithyabn@msrit.edu
Ph:9900087291

MCA23: Database Systems: Unit II

Contents

UNIT II
SQL Basic Retrieval Queries in SQL; INSERT, DELETE and UPDATE statements in SQL; Additional features of SQL, Complex Queries **Pl/SQL**Introduction to PL/SQL, Procedures and Functions, Triggers.

1 Basic Retrieval Queries in SQL

SQL has one basic statement for retrieving information from a database: the SELECT statement.

1.1 The SELECT – FROM – WHERE structure of basic SQL Queries

The basic form of the SELECT statement, sometimes called a mapping or a select – from – where block, is formed of the three clauses SELECT, FROM and WHERE and has the following form.
SELECT < *attributelist* >
FROM < *tablelist* >
WHERE < *condition* >;
Where

- < *attributelist* > is a list of attribute names whose values are to be retrieved by the query
- < *tablelist* > is a list of the relation names required to process the query.
- < *condition* > is a conditional (Boolean) expression that identifies the tuples to be retrieved by the query.
- Aliasing for the table name and attribute names

```
SELECT id as Identity_number FROM persons;

SELECT lastname || firstname as name FROM persons;

SELECT lastname||',' ||firstname as name FROM persons;
```

- Retrieve the birth date and address of the employee(s) whose name is 'John B. Smith'

```
SELECT Bdate, Address
FROM EMPLOYEE
WHERE Fname='John' AND Minit='B'
AND Lname='Smith';
```

- Retrieve the name and address of all employees who work for the 'Research' department

```
SELECT Fname, Lname, Address
FROM EMPLOYEE, DEPARTMENT
WHERE Dname='Research' AND Dnumber=Dno;
```

- 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

```
SELECT Pnumber, Dnum, Lname, Address,
Bdate FROM PROJECT, DEPARTMENT, EMPLOYEE
WHERE Dnum=Dnumber AND Mgr_ssn=Ssn
AND Plocation='Stafford';
```

- In SQL, the same name can be used for two (or more) attributes as long as the attributes are in different tables. In this case, we must specify which attribute you are referring to from what table which addressing same name attributes of different tables as under.

```
SELECT Fname, EMPLOYEE.Name, Address
FROM DEPARTMENT, EMPLOYEE
WHERE DEPARTMENT.Name='Research' AND
DEPARTMENT.Dnumber=EMPLOYEE.Dnumber;
```

- Selecting distinct values in a table

```
select DISTINCT age from persons;
```

- Selecting values from a table based on a particular value

```
select * from persons where age=36;
select * from persons where age between 20 and 30;
```

- Selecting values from two tables

```
select persons.id, persons.age, orders.ordernumber
from persons, orders;
```

- Substring pattern matching and arithmetic operations

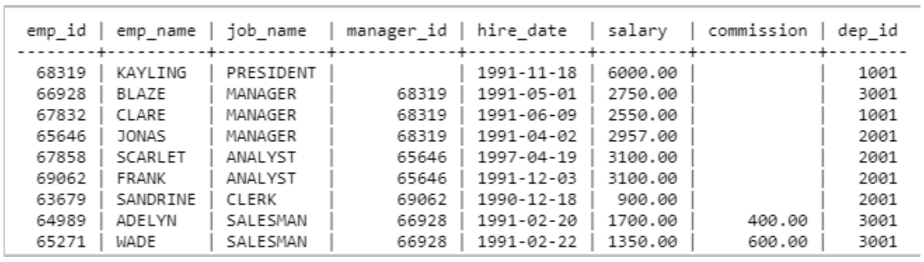
```
select * from persons where lastname like 'S%';
select * from persons where lastname like '%n';
select * from persons where lastname like '%o%';
```

2 Practice Sessions

2.1 Database and Queries

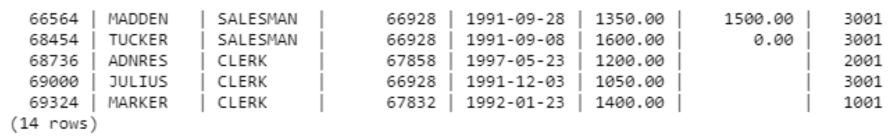
Create the following database schema and write queries.

DEPARTMENT (dept_id (int), dept_name (varchar (20)), dep_location (varchar (15)))
SALARY-GRADE (grade (int), min_sal(int), max_sal(int))
EMPLOYEES (emp_id(int), emp_name (varchar (15)), job_name (varchar (10)), manager_id(int), hire_date(date), salary (float (10,2)), commission (float (7,2)), dept_id(int))



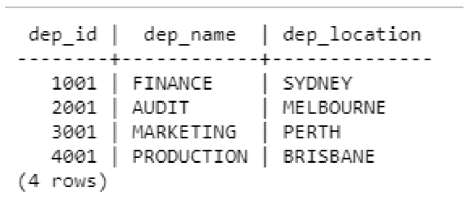
```
 emp_id | emp_name | job_name  | manager_id | hire_date  | salary  | commission | dep_id
--------+----------+-----------+------------+------------+---------+------------+--------
  68319 | KAYLING  | PRESIDENT |            | 1991-11-18 | 6000.00 |            |   1001
  66928 | BLAZE    | MANAGER   |      68319 | 1991-05-01 | 2750.00 |            |   3001
  67832 | CLARE    | MANAGER   |      68319 | 1991-06-09 | 2550.00 |            |   1001
  65646 | JONAS    | MANAGER   |      68319 | 1991-04-02 | 2957.00 |            |   2001
  67858 | SCARLET  | ANALYST   |      65646 | 1997-04-19 | 3100.00 |            |   2001
  69062 | FRANK    | ANALYST   |      65646 | 1991-12-03 | 3100.00 |            |   2001
  63679 | SANDRINE | CLERK     |      69062 | 1990-12-18 |  900.00 |            |   2001
  64989 | ADELYN   | SALESMAN  |      66928 | 1991-02-20 | 1700.00 |     400.00 |   3001
  65271 | WADE     | SALESMAN  |      66928 | 1991-02-22 | 1350.00 |     600.00 |   3001
```

Figure 1: Sample State of Employees table for queries



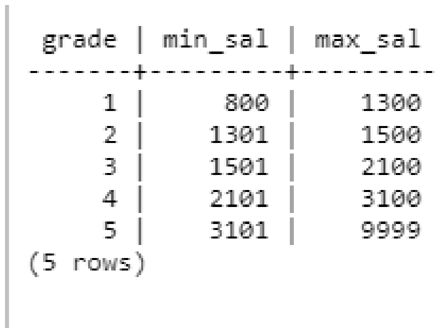
```
  66564 | MADDEN   | SALESMAN  |      66928 | 1991-09-28 | 1350.00 |    1500.00 |   3001
  68454 | TUCKER   | SALESMAN  |      66928 | 1991-09-08 | 1600.00 |       0.00 |   3001
  68736 | ADNRES   | CLERK     |      67858 | 1997-05-23 | 1200.00 |            |   2001
  69000 | JULIUS   | CLERK     |      66928 | 1991-12-03 | 1050.00 |            |   3001
  69324 | MARKER   | CLERK     |      67832 | 1992-01-23 | 1400.00 |            |   1001
(14 rows)
```

Figure 2: Sample State of Employees table for queries



```
 dep_id |  dep_name  | dep_location
--------+------------+--------------
   1001 | FINANCE    | SYDNEY
   2001 | AUDIT      | MELBOURNE
   3001 | MARKETING  | PERTH
   4001 | PRODUCTION | BRISBANE
(4 rows)
```

Figure 3: Sample State of Department table for queries



```
 grade | min_sal | max_sal
-------+---------+---------
     1 |     800 |    1300
     2 |    1301 |    1500
     3 |    1501 |    2100
     4 |    2101 |    3100
     5 |    3101 |    9999
(5 rows)
```

Figure 4: Sample State of Salary_Grade table for queries

1. Write a query in SQL to display all the information of the employees.
2. Write a query in SQL to find the salaries of all employees.
3. Write a query in SQL to display the unique designations for the employees.
4. Write a query in SQL to list the emp_name and salary is increased by 15% and expressed as no.of Dollars.
5. Write a query in SQL to produce the output of employee's name and job name as a format of "Employee & Job".
6. Write a query in SQL to produce the output of employees as follows.Employee JONAS(manager).
7. Write a query in SQL to list the employees with Hire date in the format like February 22, 1991.
8. Write a query in SQL to count the no. of characters without considering the spaces for each name.
9. Write a query in SQL to list the emp_id,salary, and commission of all the employees.
10. Write a query in SQL to display the unique department with jobs.
11. Write a query in SQL to list the employees who does not belong to department 2001.
12. Write a query in SQL to list the employees who joined before 1991.
13. Write a query in SQL to display the average salaries of all the employees who works as ANALYST.
14. Write a query in SQL to display the details of the employee BLAZE.
15. Write a query in SQL to display all the details of the employees whose commission is more than their salary.
16. Write a query in SQL to list the employees whose salary is more than 3000 after giving 25% increment.

17. Write a query in SQL to list the name of the employees, those having six characters to their name.
18. Write a query in SQL to list the employees who joined in the month January.
19. Write a query in SQL to list the name of employees and their manager separated by the string 'works for'.
20. Write a query in SQL to list all the employees whose designation is CLERK.
21. Write a query in SQL to list the employees whose experience is more than 27 years.
22. Write a query in SQL to list the employees whose salaries are less than 3500.
23. Write a query in SQL to list the name, job_name, and salary of any employee whose designation is ANALYST.
24. Write a query in SQL to list the employees who have joined in the year 1991.
25. Write a query in SQL to list the name, id, hire_date, and salary of all the employees joined before 1 apr 91.
26. Write a query in SQL to list the employee name, and job_name who are not working under a manager.
27. Write a query in SQL to list all the employees joined on 1st may 91.
28. Write a query in SQL to list the id, name, salary, and experiences of all the employees working for the manger 68319.
29. Write a query in SQL to list the id, name, salary, and experience of all the employees who earn more than 100 as daily salary.
30. Write a query in SQL to list the employees who are retiring after 31-Dec-99 after completion of 8 years of service period.
31. Write a query in SQL to list those employees whose salary is an odd value.
32. Write a query in SQL to list those employees whose salary contain only 3 digits. 33. Write a query in SQL to list the employees who joined in the month of APRIL.
33. Write a query in SQL to list the employees those who joined in company before 19th of the month.
34. List the employees who are SALESMAN and gathered an experience which month portion is more than 10.
35. Write a query in SQL to list the employees of department id 3001 or 1001 joined in the year 1991.
36. Write a query in SQL to list the employees of department id 3001 or 1001 joined in the year 1991.

37. Write a query in SQL to list all the employees of designation CLERK in department no 2001.
38. Write a query in SQL to list the ID, name, salary, and job_name of the employees for
    - Annual salary is below 34000 but receiving some commission which should not be more than the salary,
    - And designation is SALESMAN and working for department 3001.
39. Write a query in SQL to list the employees who are either CLERK or MANAGER
40. Write a query in SQL to list the employees who joined in any year except the month February.
41. Write a query in SQL to list the employees who joined in the year 91.
42. Write a query in SQL to list the employees who joined in the month of June in 1991.
43. Write a query in SQL to list the employees whose annual salary is within the range 24000 and 50000.
44. Write a query in SQL to list the employees who have joined on the following dates 1st May,20th Feb, and 03rd Dec in the year 1991.
45. Write a query in SQL to list the employees working under the managers 63679,68319,66564,69000.
46. Write a query in SQL to list the employees who joined after the month JUNE in the year 1991.
47. Write a query in SQL to list the employees who joined in 90's.
48. Write a query in SQL to list the managers of department 1001 or 2001.
49. Write a query in SQL to list the employees, joined in the month FEBRUARY with a salary range between 1001 to 2000.
50. Write a query in SQL to list all the employees who joined before or after 1991.
51. Write a query in SQL to list the employees along with department name.
52. Write a query in SQL to list the name, job name, annual salary, department id, department name and grade of the employees who earn 60000 in a year or not working as an ANALYST.
53. Write a query in SQL to list the name, job name, manager id, salary, manager name, manager's salary for those employees whose salary is greater than the salary of their managers.
54. Write a query in SQL to list the employees name, department, salary and commission. For those whose salary is between 2000 and 5000 while location is PERTH.

55. Write a query in SQL to list the grade, employee name for the department id 1001 or 3001 but salary grade is not 4 while they joined the company before 1992-12-31.

56. Write a query in SQL to list the employees whose manager name is JONAS.

57. Write a query in SQL to list the name and salary of FRANK if his salary is equal to max_sal of his grade.

58. Write a query in SQL to list the employees who are working either MANAGER or ANALYST with a salary range between 2000 to 5000 without any commission.

59. Write a query in SQL to list the id, name, salary, and location of the employees working at PERTH,or MELBOURNE with an experience over 10 years.

60. Write a query in SQL to list the employees along with their location who belongs to SYDNEY, MELBOURNE with a salary range between 2000 and 5000 and joined in 1991.

61. Write a query in SQL to list the employees with their location and grade for MARKETING department who comes from MELBOURNE or PERTH within the grade 3 to 5 and experience over 5 years.

62. Write a query in SQL to list the employees who are senior to their own manager.

63. Write a query in SQL to list the employee with their grade for the grade 4.

64. Write a query in SQL to list the employees in department PRODUCTION or AUDIT who joined after 1991 and they are not MARKER or ADELYN to their name.

65. Write a query in SQL to list the employees in the ascending order of their salaries.

66. Write a query in SQL to list the details of the employees in ascending order to the department_id and descending order to the jobs.

67. Write a query in SQL to display all the unique job in descending order.

68. Write a query in SQL to list the id, name, monthly salary, daily salary of all the employees in the ascending order of their annual salary.

69. Write a query in SQL to list the employees in descending order who are either 'CLERK' or 'ANALYST'.

70. Write a query in SQL to display the location of CLARE.

71. Write a query in SQL to list the employees in ascending order of seniority who joined on 1-MAY-91,or 3-DEC-91, or 19-JAN-90.

72. Write a query in SQL to list the employees who are drawing the salary less than 1000 and sort the output in ascending order on salary.

73. Write a query in SQL to list the details of the employees in ascending order on the salary.

74. Write a query in SQL to list the employees in ascending order on job name and descending order on employee id.

75. Write a query in SQL to list the unique jobs of department 2001 and 3001 in descending order.

76. Write a query in SQL to list all the employees except PRESIDENT and MANAGER in ascending order of salaries.

77. Write a query in SQL to list the employees in ascending order of the salary whose annual salary is below 25000.

78. Write a query in SQL to list the employee id, name, annual salary, daily salary of all the employees in the ascending order of annual salary who works as a SALESMAN.

79. Write a query in SQL to list the employee id, name, hire_date, current date and experience of the employees in ascending order on their experiences.

80. Write a query in SQL to list the employees in ascending order of designations of those, joined after the second half of 1991.

81. Write a query in SQL to list the total information of employees table along with department, and location of all the employees working under FINANCE and AUDIT in the ascending department no.

82. Write a query in SQL to display the total information of the employees along with grades in ascending order.

83. Write a query in SQL to list the name, job name, department, salary, and grade of the employees according to the department in ascending order.

84. Write a query in SQL to list the name, job name, salary, grade and department name of employees except CLERK and sort result set on the basis of highest salary.

85. Write a query in SQL to list the employee ID, name, salary, department, grade, experience, and annual salary of employees working for department 1001 or 2001.

86. Write a query in SQL to list the details of the employees along with the details of their departments.

87. Write a query in SQL to list the employees who are senior to their own MANAGERS.

88. Write a query in SQL to list the employee id, name, salary, and department id of the employees in ascending order of salary who works in the department 1001.

89. Write a query in SQL to find the highest salary from all the employees.
90. Write a query in SQL to find the average salary and average total remuneration (salary and commission) for each type of job.
91. Write a query in SQL to find the total annual salary distributed against each job in the year 1991.
92. Write a query in SQL to list the employee id, name, department id, location of all the employees.
93. Write a query in SQL to list the employee id, name, location, department of all the departments 1001 and 2001.
94. Write a query in SQL to list the employee id, name, salary, grade of all the departments 1001 and 2001.
95. Write a query in SQL to list the manager no and the number of employees working for those managers in ascending order on manager id.
96. Write a query in SQL to display the number of employee for each job in each department.
97. Write a query in SQL to list the department where at least two employees are working.
98. Write a query in SQL to display the Grade, Number of employees, and maximum salary of each grade.
99. Write a query in SQL to display the department name, grade, no. of employees where at least two employees are working as a SALESMAN.
100. Write a query in SQL to list the no. of employees in each department where the no. is less than 4.
101. Write a query in SQL to list the name of departments where atleast 2 employees are working in that department.
102. Write a query in SQL to check whether all the employee's numbers are indeed unique.
103. Write a query in SQL to list the no. of employees and average salary within each department for each job name
104. Write a query in SQL to list the names of those employees starting with 'A' and with six characters in length.
105. Write a query in SQL to list the employees whose name is six characters in length and third character must be 'R'.
106. Write a query in SQL to list the name of the employee of six characters long and starting with 'A' and ending with 'N'.
107. Write a query in SQL to list the employees who joined in the month of which second character is 'a'.

108. Write a query in SQL to list the employees whose names containing the character set 'AR' together.
109. Write a query in SQL to list the employees those who joined in 90's.
110. Write a query in SQL to list the employees whose ID not starting with digit 68.
111. Write a query in SQL to list the employees whose names containing the letter 'A'.
112. Write a query in SQL to list the employees whose name is ending with 'S' and six characters long.
113. Write a query in SQL to list the employees who joined in the month having char 'A' at any position.
114. Write a query in SQL to list the employees who joined in the month having second char is 'A'.

2.2 Solutions :Creating the tables and inserting the data

- Creating table emp_bn

```
CREATE TABLE emp\_bn (
emp_id int NOT NULL,
ename varchar(15),
job_name varchar(15),
manager_id int,
hire_date date,
salary decimal(10,2),
commission decimal(7,2),
dept_id int,
PRIMARY KEY (emp_id)
);
```

- Creating table dept_bn

```
CREATE TABLE dept_bn (
dept_id int NOT NULL,
dept_name varchar(20),
dept_location varchar(15),
PRIMARY KEY (dept_id)
);
```

- Creating table salary_grade_bn

```
CREATE TABLE salary_grade_bn (
grade int NOT NULL,
min_sal int,
max_sal int,
primary key(grade)
);
```

- Adding Foreign key reference to emp_bn table

```
ALTER TABLE emp_bn
ADD FOREIGN KEY (dept_id)
REFERENCES dept_bn(dept_id);
```

- Inserting values into dept_bn

```
insert into dept_bn values(1001, 'Finance', 'Sydney');
insert into dept_bn values(2001, 'Audit', 'Melborne');
insert into dept_bn values(3001, 'Marketing', 'Perth');
insert into dept_bn values(4001, 'Production', 'Brisbane');
```

- Inserting values into salary_grade_bn

```
insert into salary_grade_bn values(1,800,1300);
insert into salary_grade_bn values(2,1301,1500);
insert into salary_grade_bn values(3,1501,2100);
insert into salary_grade_bn values(4,2101,3100);
insert into salary_grade_bn values(5,3101,9999);
```

- Inserting values into emp_bn

```
insert into emp_bn values (68319,'Kayling','President',
Null,'18-Nov-1991',6000.00,null,1001);
insert into emp_bn values(66928,'Blaze','Manager'
,68319,'01-May-1991',2750.00,null,3001);
insert into emp_bn values(67832,'Clare','Manager'
,68319,'09-Jun-1991',2550.00,null,1001);
insert into emp_bn values(65646,'Jonas','Manager'
,68319,'02-Apr-1991',2957.00,null,2001);
insert into emp_bn values(67858,'Scarlet','Analyst'
,65646,'19-Apr-1997',3100.00,null,2001);
insert into emp_bn values(69062,'Frank','Analyst'
,65646,'03-Dec-1991',3100.00,null,2001);
insert into emp_bn values(63679,'Sandrine','Clerk'
,69062,'18-Dec-1990',900.00,null,2001);
insert into emp_bn values(64989,'Adelyn','Salesman'
,66928,'20-Feb-1991',1700.00,400.00,3001);
insert into emp_bn values(65271,'Wade','Salesman'
,66928,'22-Feb-1991',1350.00,600.00,3001);
insert into emp_bn values(66564,'Madden','Salesman'
,66928,'28-Sep-1991',1350.00,1500.00,3001);
insert into emp_bn values(68454,'Tucker','Salesman'
,66928,'08-Sep-1991',1600.00,0.00,3001);
insert into emp_bn values(68736,'Adnres','Clerk'
,67858,'23-May-1997',1200.00,null,2001);
insert into emp_bn values(69000,'Julius','Clerk'
,66928,'03-Dec-1991',1050.00,null,3001);
insert into emp_bn values(69324,'Marker','Clerk'
,67832,'23-Jan-1992',1400.00,null,1001);
```

2.3 Solutions for Queries

1. Write a query in SQL to display all the information of the employees.

```
 select * from emp_bn;
```

2. Write a query in SQL to find the salaries of all employees.

```
select salary from emp_bn;
```

3. Write a query in SQL to display the unique designations for the employees.

```
select distinct(job_name) from emp_bn;
```

4. Write a query in SQL to list the emp_name and salary is increased by 15% and expressed as no.of Dollars.

```
SELECT ename, to_char(1.15*salary,'$99,999') AS "Salary" FROM emp_bn;
```

5. Write a query in SQL to produce the output of employees as follows.
   Employee
   JONAS(manager).

```
SELECT ename || '('|| lower(job_name)||')' AS "Employee" FROM emp_bn;
```

6. Write a query in SQL to list the employees with Hire date in the format like February 22, 1991.

```
SELECT emp_id,ename, salary,
to_char(hire_date,'MONTH DD,YYYY') FROM emp_bn;
                         OR
SELECT emp_id,ename,salary,
to_char(hire_date,'MONTH DD,YYYY') AS Hire_Date FROM emp_bn;
```

7. Write a query in SQL to count the no. of characters without considering the spaces for each name.

```
SELECT length(trim(ename))FROM emp_bn;
```

8. Write a query in SQL to list the emp_id,salary, and commission of all the employees.

```
SELECT emp_id,salary,commission FROM emp_bn;
```

9. Write a query in SQL to display the unique department with jobs.

```
SELECT DISTINCT dept_id, job_name FROM emp_bn;
```

10. Write a query in SQL to list the employees who does not belong to department 2001.

```
SELECT * FROM emp_bn WHERE dept_id NOT IN (2001);
```

11. Write a query in SQL to list the employees who joined before 1991.

```
SELECT * FROM emp_bn WHERE hire_date<('01-Jan-1991');
```

12. Write a query in SQL to display the average salaries of all the employees who works as ANALYST.

```
SELECT avg(salary) FROM emp_bn WHERE job_name = 'Analyst';
```

13. Write a query in SQL to display the details of the employee BLAZE.

```
SELECT * FROM emp_bn WHERE ename = 'Blaze';
```

14. Write a query in SQL to display all the details of the employees whose commission is more than their salary.

```
SELECT * FROM emp_bn WHERE commission>salary;
```

15. Write a query in SQL to list the employees whose salary is more than 3000 after giving 25% increment.

```
SELECT * FROM emp_bn WHERE (1.25*salary) > 3000;
```

16. Write a query in SQL to list the name of the employees, those having six characters to their name.

```
SELECT ename FROM emp_bn WHERE length(ename)=6;
```

17. Write a query in SQL to list the employees who joined in the month January.

```
SELECT * FROM emp_bn WHERE to_char(hire_date, 'Mon')='Jan';
```

18. Write a query in SQL to list the name of employees and their manager separated by the string 'works for'.

```
SELECT e.ename || ' works for ' || m.ename
FROM emp_bn e,emp_bn m
WHERE e.manager_id = m.emp_id;
```

19. Write a query in SQL to list all the employees whose designation is CLERK.

```
SELECT * FROM emp_bn WHERE job_name = 'Clerk';
```

20. Write a query in SQL to list the employee name and his experience in days.

```
SELECT ename, floor(sysdate-hire_date)
as experience_in_days from emp_bn;
```

21. Write a query in SQL to extract year, month and date separately from the column hire_date.

```
SELECT EMP_ID, ENAME,
extract(YEAR from hire_date) AS Year_of_Joining
FROM EMP_bn;

                    OR

SELECT EMP_ID, ENAME,
extract(Month from hire_date) AS Month_of_Joining
FROM EMP_bn;

                    OR

SELECT EMP_ID, ENAME, extract(Day from hire_date)
AS Day_of_Joining FROM EMP_bn;
```

22. Write a query in SQL to list the employees whose salaries are less than 3500.

```
SELECT * FROM emp_bn WHERE salary <3500;
```

23. Write a query in SQL to list the name, job_name, and salary of any employee whose designation is ANALYST.

```
SELECT ename,job_name,salary
FROM emp_bn
WHERE job_name = 'Analyst';
```

24. Write a query in SQL to list the employees who have joined in the year 1991.

```
SELECT * FROM emp_bn
WHERE to_char(hire_date,'YYYY') = '1991';
```

25. Write a query in SQL to list the name, id, hire_date, and salary of all the employees joined before 1 apr 91.

```
SELECT e.emp_id,e.ename,e.hire_date,e.salary
FROM emp_bn e WHERE hire_date <'01-Apr-1991';
```

26. Write a query in SQL to list the employee name, and job_name who are not working under a manager.

```
SELECT e.ename,e.job_name FROM emp_bn e
WHERE manager_id IS NULL;
```

27. Write a query in SQL to list all the employees joined on 1st may 91.

```
SELECT * FROM emp_bn WHERE hire_date = '01-May-1991';
```

28. Write a query in SQL to list the id, name, salary, and experience of all the employees who earn more than 100 as daily salary.

```
SELECT emp_id, ename, salary,
(to_date (SYSDATE) - to_date(hire_date))
experience_in_days
FROM emp_bn WHERE (salary/30)>100;
```

29. Write a query in SQL to list those employees whose salary is an odd value.

```
SELECT * FROM emp_bn WHERE mod(salary,2) = 1;
```

30. Write a query in SQL to list those employees whose salary contain only 3 digits.

```
SELECT * FROM emp_bn WHERE length(TRIM(TO_CHAR(salary,'9999'))) = 3;
```

31. Write a query in SQL to list the employees who joined in the month of APRIL.

```
SELECT * FROM emp_bn WHERE to_char(hire_date,'MON') ='APR';
```

32. List the employees who are SALESMAN and gathered an experience which month portion is more than 10.

```
SELECT * FROM emp_bn WHERE job_name = 'Salesman'
AND
floor(months_between(sysdate,hire_date)) > 10;
```

33. Write a query in SQL to list the employees of department id 3001 or 1001 joined in the year 1991.

```
SELECT * FROM emp_bn
WHERE to_char(hire_date,'YYYY') = '1991'
AND
(dept_id =3001 OR dept_id =1001) ;
```

34. Write a query in SQL to list all the employees of designation CLERK in department no 2001.

```
SELECT * FROM emp_bn WHERE job_name ='Clerk'
AND dept_id = 2001;
```

35. Write a query in SQL to list the ID, name, salary, and job_name of the employees for

- Annual salary is below 34000 but receiving some commission which should not be more than the salary,
- And designation is SALESMAN and working for department 3001.

```
SELECT emp_id,ename,salary,job_name
FROM emp_bn WHERE 12*(salary+commission) < 34000
AND commission IS NOT NULL AND commission < salary
AND job_name = 'Salesman'
AND dept_id = 3001;
```

36. Write a query in SQL to list the employees who are either CLERK or MANAGER

```
SELECT * FROM emp_bn WHERE job_name IN ('Clerk','Manager');
```

37. Write a query in SQL to list the employees who joined in any year except the month February.

```
SELECT * FROM emp_bn WHERE to_char(hire_date,'MON')
NOT IN ('FEB');
```

38. Write a query in SQL to list the employees who joined in the year 91.

```
SELECT * FROM emp_bn WHERE hire_date
BETWEEN '01-Jan-1991' AND '31-Dec-1991';
```

39. Write a query in SQL to list the employees who joined in the month of June in 1991.

```
SELECT * FROM emp_bn WHERE hire_date
BETWEEN '01-Jun-1991' AND '30-Jun-1991';
            OR
SELECT * FROM emp_bn WHERE
to_char(hire_date,'Mon-yyyy')='Jun-1991';
```

40. Write a query in SQL to list the employees whose annual salary is within the range 24000 and 50000.

```
SELECT * FROM emp_bn WHERE 12*salary
BETWEEN 24000 AND 50000;
```

41. Write a query in SQL to list the employees who have joined on the following dates 1st May,20th Feb, and 03rd Dec in the year 1991.

```
SELECT * FROM emp_bn WHERE
to_char(hire_date,'DD-MON-YY')
IN ('01-MAY-91','20-FEB-91','03-DEC-91');
```

42. Write a query in SQL to list the employees working under the managers 63679,68319,66564,69000.

```
SELECT * FROM emp_bn WHERE manager_id
IN (63679,68319,66564,69000);
```

43. Write a query in SQL to list the employees who joined after the month JUNE in the year 1991.

```
SELECT * FROM emp_bn WHERE hire_date
BETWEEN '01-JUL-91' AND '31-DEC-92';
```

44. Write a query in SQL to list the employees who joined in 90's.

```
SELECT * FROM emp_bn WHERE to_char(hire_date,'YY')
BETWEEN '90' AND '99';
```

45. Write a query in SQL to list the managers of department 1001 or 2001.

```
SELECT * FROM emp_bn WHERE job_name = 'Manager'
AND
(dept_id = 1001 OR dept_id =2001);
```

46. Write a query in SQL to list the employees, joined in the month FEBRUARY with a salary range between 1001 to 2000.

```
SELECT * FROM emp_bn WHERE
to_char(hire_date,'MON') = 'FEB'
AND
salary BETWEEN 1000 AND 2000;
```

47. Write a query in SQL to list all the employees who joined before or after 1991.

```
SELECT * FROM emp_bn WHERE to_char(hire_date,'YYYY')
NOT IN ('1991');
                 OR
SELECT * FROM emp_bn WHERE to_char (hire_date,'YYYY')
NOT LIKE '1991';
```

48. Write a query in SQL to list the employees along with department name.

```
SELECT e.emp_id, e.ename, e.job_name, e.manager_id,
e.hire_date, e.salary, e.commission,
e.dept_id, d.dept_name
FROM emp_bn e, dept_bn d
WHERE e.dept_id = d.dept_id;
```

49. Write a query in SQL to list the name, job name, annual salary, department id, department name and grade of the employees who earn 60000 in a year or not working as an ANALYST.

```
SELECT e.ename,e.job_name,(12*e.salary)
"Annual Salary",e.dept_id,d.dept_name,s.grade
FROM emp_bn e,dept_bn d, salary_grade_bn s
WHERE e.dept_id = d.dept_id
AND e.salary BETWEEN s.min_sal
AND s.max_sal AND (((12*e.salary)>= 60000)
OR (e.job_name != 'ANALYST'));
```

50. Write a query in SQL to list the name, job name, manager id, salary, manager name, manager's salary for those employees whose salary is greater than the salary of their managers.

```
SELECT w.ename, w.job_name, w.manager_id, w.salary,
m.ename "Manager",m.emp_id,
m.salary "Manager_Salary"
FROM emp_bn w, emp_bn m
WHERE w.manager_id = m.emp_id
AND w.salary > m.salary;
```

51. Write a query in SQL to list the employees name, department, salary and commission. For those whose salary is between 2000 and 5000 while location is PERTH.

```
SELECT e.ename, e.dept_id, e.salary, e.commission
FROM emp_bn e, dept_bn d
WHERE e.dept_id = d.dept_id
AND d.dept_location = 'Perth'
AND e.salary BETWEEN 2000 AND 5000;
```

52. Write a query in SQL to list the grade, employee name for the department id 1001 or 3001 but salary grade is not 4 while they joined the company before 1992-12-31.

```
SELECT s.grade,e.ename FROM emp_bn e, salary_grade_bn s
WHERE e.dept_id IN (1001,3001)
AND hire_date < ('31-Dec-1992')
AND (e.salary BETWEEN s.min_sal
AND s.max_sal AND s.grade NOT IN (4));
```

53. Write a query in SQL to list the employees whose manager name is JONAS.

```
SELECT w.emp_id, w.ename, w.job_name,
w.manager_id, w.hire_date,w.salary,
w.dept_id, m.ename FROM emp_bn w, emp_bn m
WHERE w.manager_id = m.emp_id AND m.ename = 'Jonas';
```

54. Write a query in SQL to list the name and salary of FRANK if his salary is equal to max_sal of his grade.

```
SELECT e.ename,e.salary
FROM emp_bn e, salary_grade_bn s
WHERE e.ename = 'Frank'
AND e.salary BETWEEN s.min_sal
AND s.max_sal AND e.salary = s.max_sal ;
```

55. Write a query in SQL to list the employees who are working either MANAGER or ANALYST with a salary range between 2000 to 5000 without any commission.

```
SELECT * FROM emp_bn WHERE job_name
IN ('Manager','Analyst')AND
salary BETWEEN 2000 AND 5000 AND
commission IS NULL;
```

56. Write a query in SQL to list the id, name, salary, and location of the employees working at PERTH,or MELBOURNE with an experience over 10 years.

```
SELECT e.emp_id, e.ename, e.dept_id, e.salary,
d.dept_location FROM emp_bn e,  dept_bn d
WHERE e.dept_id = d.dept_id AND d.dept_location
IN ('Perth', 'Melborne')
AND (to_date(SYSDATE) - to_date(hire_date))  > 10;
```

57. Write a query in SQL to list the employees along with their location who belongs to SYDNEY, MELBOURNE with a salary range between 2000 and 5000 and joined in 1991.

```
SELECT e.emp_id, e.ename, e.dept_id, e.salary,
d.dept_location FROM emp_bn e, dept_bn d
WHERE e.dept_id = d.dept_id
AND d.dept_location
IN ('Sydney','Melborne')
AND to_char(e.hire_date,'YY') = '91'
AND e.salary BETWEEN 2000 AND 5000;
```

58. Write a query in SQL to list the employees with their location and grade for MARKETING department who comes from MELBOURNE or PERTH within the grade 3 to 5 and experience over 5 years.

```
SELECT e.dept_id, e.emp_id, e.ename,
e.salary,d.dept_name,d.dept_location,s.grade
FROM emp_bn e, salary_grade_bn s, dept_bn d
WHERE e.dept_id = d.dept_id  AND e.salary
BETWEEN s.min_sal AND s.max_sal AND s.grade
IN (3,4,5) AND (to_date(SYSDATE) - to_date(hire_date)) > 5
AND (d.dept_name = 'Marketing' AND D.dept_location
IN ('Melborne','Perth'));
```

59. Write a query in SQL to list the employees who are senior to their own manager.

```
SELECT * FROM emp_bn w,emp_bn m
WHERE w.manager_id = m.emp_id
AND w.hire_date < m.hire_date;
```

60. Write a query in SQL to list the employee with their grade for the grade 4.

```
SELECT * FROM emp_bn e, salary_grade_bn s
WHERE e.salary BETWEEN s.min_sal AND s.max_sal
AND s.grade = 4;
```

61. Write a query in SQL to list the employees in department PRODUCTION or AUDIT who joined after 1991 and they are not MARKER or ADELYN to their name.

```
SELECT e.ename FROM emp_bn e, dept_bn d, salary_grade_bn s
WHERE e.dept_id = d.dept_id AND d.dept_name IN
('Production','Audit') AND e.salary BETWEEN s.min_sal
AND s.max_sal AND e.ename NOT IN ('Marker', 'Adelyn')
AND to_char(hire_date,'YYYY') >'1991';
```

62. Write a query in SQL to list the employees in the ascending order of their salaries.

```
SELECT * FROM emp_bn ORDER BY salary ASC;
```

63. Write a query in SQL to list the details of the employees in ascending order to the department_id and descending order to the jobs.

```
SELECT * FROM emp_bn ORDER BY dept_id ASC, job_name DESC;
```

64. Write a query in SQL to display all the unique job in descending order.

```
SELECT DISTINCT job_name FROM emp_bn ORDER BY job_name DESC;
```

65. Write a query in SQL to list the id, name, monthly salary, daily salary of all the employees in the ascending order of their annual salary.

```
SELECT emp_id,ename,salary Monthly_Salary,salary/30
Daily_Salary,12*salary Annual_Salary FROM emp_bn
ORDER BY Annual_Salary ASC;
```

66. Write a query in SQL to list the employees in descending order who are either 'CLERK' or 'ANALYST'.

```
SELECT * FROM emp_bn WHERE job_name='Clerk'
OR job_name='Analyst'ORDER BY job_name DESC;
```

67. Write a query in SQL to display the location of CLARE.

```
SELECT dept_location FROM dept_bn d,emp_bn e
WHERE e.ename = 'Clare' AND e.dept_id = d.dept_id ;
```

68. Write a query in SQL to list the employees in ascending order of seniority who joined on 1-MAY-91,or 3-DEC-91, or 19-JAN-90.

```
SELECT * FROM emp_bn WHERE hire_date
IN ('01-May-1991','03-Dec-1991','19-Jan-1990')
ORDER BY hire_date ASC;
```

69. Write a query in SQL to list the employees who are drawing the salary less than 1000 and sort the output in ascending order on salary.

```
SELECT * FROM emp_bn WHERE salary < 1000 ORDER BY salary;
```

70. Write a query in SQL to list the details of the employees in ascending order on the salary.

```
SELECT * FROM emp_bn ORDER BY salary ASC;
```

71. Write a query in SQL to list the employees in ascending order on job name and descending order on employee id.

```
SELECT * FROM emp_bn e
ORDER BY e.job_name ASC,
e.emp_id DESC ;
```

72. Write a query in SQL to list the unique jobs of department 2001 and 3001 in descending order.

```
SELECT DISTINCT job_name
FROM emp_bn WHERE dept_id IN (2001,3001)
ORDER BY job_name DESC;
```

73. Write a query in SQL to list all the employees except PRESIDENT and MANAGER in ascending order of salaries.

```
SELECT * FROM emp_bn WHERE job_name
NOT IN ('President','Manager')
ORDER BY salary ASC;
```

74. Write a query in SQL to list the employees in ascending order of the salary whose annual salary is below 25000.

```
SELECT * FROM emp_bn
WHERE (12*salary) < 25000
ORDER BY salary ASC;
```

75. Write a query in SQL to list the employee id, name, annual salary, daily salary of all the employees in the ascending order of annual salary who works as a SALESMAN

```
SELECT e.emp_id,e.ename,12*salary
"Annual Salary",floor((12*salary)/365)
"Daily Salary" FROM emp_bn e
WHERE e.job_name = 'Salesman'
ORDER BY "Annual Salary" ASC;
             OR
SELECT e.emp_id,e.ename,12*salary "Annual Salary",
(12*salary)/365 "Daily Salary"
FROM emp_bn e WHERE e.job_name = 'Salesman'
ORDER BY "Annual Salary" ASC;
```

76. Write a query in SQL to list the employee id, name, hire_date, current date and experience of the employees in ascending order on their experiences.

```
SELECT emp_id, ename, hire_date, CURRENT_DATE,
(to_date(SYSDATE) - to_date(hire_date)) EXP
FROM emp_bn ORDER BY EXP ASC;
```

77. Write a query in SQL to list the employees in ascending order of designations of those, joined after the second half of 1991.

```
SELECT * FROM emp_bn WHERE hire_date>to_char('30-Jun-1991')
AND to_char(hire_date,'YYYY') >'1991'
ORDER BY job_name ASC;
```

78. Write a query in SQL to list the total information of employees table along with department, and location of all the employees working under FINANCE and AUDIT in the ascending department no.

```
SELECT * FROM emp_bn e, dept_bn d
WHERE (dept_name = 'Finance'OR dept_name ='Audit')
AND e.dept_id = d.dept_id ORDER BY e.dept_id ASC;
```

79. Write a query in SQL to display the total information of the employees along with grades in ascending order.

```
   SELECT * FROM emp_bn e, salary_grade_bn s
   WHERE e.salary BETWEEN s.min_sal AND s.max_sal
   ORDER BY grade ASC;
                     OR
 SELECT * FROM emp_bn e, salary_grade_bn s
 WHERE e.salary >= s.min_sal
 AND e.salary <= s.max_sal ORDER BY s.grade ASC;
```

80. Write a query in SQL to list the name, job name, department, salary, and grade of the employees according to the department in ascending order.

```
SELECT e.ename, e.job_name, d.dept_name, e.salary,s.grade
FROM emp_bn e, dept_bn d, salary_grade_bn s
WHERE e.dept_id = d.dept_id  AND e.salary
BETWEEN s.min_sal AND s.max_sal ORDER BY e.dept_id ;
```

81. Write a query in SQL to list the name, job name, salary, grade and department name of employees except CLERK and sort result set on the basis of highest salary.

```
SELECT e.ename, e.job_name, e.salary, s.grade,d.dept_name
FROM emp_bn e, dept_bn d, salary_grade_bn s
WHERE e.dept_id = d.dept_id  AND e.salary
BETWEEN s.min_sal AND s.max_sal
AND e.job_name NOT IN ('Clerk')
ORDER BY e.salary DESC;
```

82. Write a query in SQL to list the employee ID, name, salary, department, grade, experience, and annual salary of employees working for department 1001 or 2001.

```
SELECT e.emp_id, e.ename, e.salary,
s.grade,d.dept_name,(to_date(SYSDATE) - to_date(hire_date))
AS "Experience",12 * e.salary "Annual Salary"
FROM emp_bn e,dept_bn d,salary_grade_bn s
WHERE e.dept_id IN (1001,2001) AND e.dept_id = d.dept_id
AND e.salary BETWEEN s.min_sal AND s.max_sal ;
```

83. Write a query in SQL to list the details of the employees along with the details of their departments.

```
SELECT * FROM emp_bn e, dept_bn d
WHERE e.dept_id= d.dept_id;
```

84. Write a query in SQL to list the employees who are senior to their own MANAGERS.

```
   SELECT * FROM emp_bn w, emp_bn m
   WHERE w.manager_id = m.emp_id
   AND w.hire_date < m.hire_date;
             OR
 SELECT * FROM emp_bn w, emp_bn m WHERE
 w.emp_id= m.manager_id  AND w.hire_date> m.hire_date;
```

85. Write a query in SQL to list the employee id, name, salary, and department id of the employees in ascending order of salary who works in the department 1001.

```
SELECT e.emp_id, e.ename, e.salary, e.dept_id
FROM emp_bn E WHERE e.dept_id = 1001
ORDER BY e.salary ASC;
```

86. Write a query in SQL to find the highest salary from all the employees.

```
SELECT max(salary) FROM emp_bn;
```

87. Write a query in SQL to find the average salary and average total remuneration (salary and commission) for each type of job.

```
SELECT job_name, avg(salary),
avg(salary+commission)
FROM emp_bn GROUP BY job_name;
```

88. Write a query in SQL to find the total annual salary distributed against each job in the year 1991.

```
SELECT job_name, sum(12*salary) FROM emp_bn
WHERE to_char(hire_date,'YYYY') = '1991'
GROUP BY job_name;
```

89. Write a query in SQL to list the employee id, name, department id, location of all the employees.

```
SELECT e.emp_id, e.ename, e.dept_id, d.dept_location
FROM emp_bn e, dept_bn d WHERE e.dept_id = d.dept_id ;
```

90. Write a query in SQL to list the employee id, name, location, department of all the departments 1001 and 2001.

```
SELECT e.emp_id, e.ename, e.dept_id,
d.dept_location,d.dept_name FROM emp_bn e, dept_bn d
WHERE e.dept_id = d.dept_id AND e.dept_id IN (1001,2001);
```

91. List the employee id, name, salary, grade of all the employees

```
SELECT e.emp_id, e.ename, e.salary, s.grade
FROM emp_bn e, salary_grade_bn s
WHERE e.salary BETWEEN s.min_sal AND s.max_sal ;
```

92. Write a query in SQL to list the manager no and the number of employees working for those managers in ascending order on manager id.

```
SELECT w.manager_id,count(*)FROM emp_bn w, emp_bn m
WHERE w.manager_id = m.emp_id GROUP BY w.manager_id
ORDER BY w.manager_id ASC;
```

93. Write a query in SQL to display the number of employee for each job in each department.

```
SELECT dept_id, job_name, count(*) FROM emp_bn
GROUP BY dept_id, job_name;
```

94. Write a query in SQL to list the department where at least two employees are working.

```
SELECT dept_id, count(*) FROM emp_bn GROUP BY dept_id
HAVING count(*) >= 2;
```

95. Write a query in SQL to display the Grade, Number of employees, and maximum salary of each grade.

```
SELECT s.grade, count(*),  max(salary)
FROM emp_bn e, salary_grade_bn s
WHERE e.salary BETWEEN s.min_sal
AND s.max_sal GROUP BY s.grade;
```

96. Write a query in SQL to display the department name, grade, no. of employees where at least two employees are working as a SALESMAN.

```
SELECT d.dept_name, s.grade, count(*) FROM emp_bn e,
dept_bn d, salary_grade_bn s WHERE e.dept_id = d.dept_id
AND e.job_name = 'Salesman' AND e.salary
BETWEEN s.min_sal AND s.max_sal
GROUP BY d.dept_name, s.grade
HAVING count(*) >= 2;
```

97. Write a query in SQL to list the no. of employees in each department where the no. is less than 4.

```
SELECT dept_id, count(*) FROM emp_bn
GROUP BY dept_id HAVING count(*)<4;
```

98. Write a query in SQL to list the name of departments where atleast 2 employees are working in that department.

```
SELECT d.dept_name, count(*) FROM emp_bn e, dept_bn d
WHERE e.dept_id = d.dept_id
GROUP BY d.dept_name HAVING count(*) >= 2;
```

99. Write a query in SQL to check whether all the employee's numbers are indeed unique.

```
SELECT emp_id, count(*) FROM emp_bn GROUP BY emp_id;
```

100. Write a query in SQL to list the no. of employees and average salary within each department for each job name

```
SELECT count(*), avg(salary), dept_id, job_name
FROM emp_bn GROUP BY dept_id, job_name;
```

101. Write a query in SQL to list the names of those employees starting with 'A' and with six characters in length.

```
SELECT ename FROM emp_bn WHERE ename LIKE 'A%'
AND length(ename)=6;
```

102. Write a query in SQL to list the employees whose name is six characters in length and third character must be 'R'.

```
SELECT * FROM emp_bn WHERE length(ename)=6
AND ename LIKE '__r%';
```

103. Write a query in SQL to list the name of the employee of six characters long and starting with 'A' and ending with 'N'.

```
SELECT * FROM emp_bn WHERE length(ename)=6
AND ename LIKE 'A%n';
```

104. Write a query in SQL to list the employees who joined in the month of which second character is 'a'.

```
SELECT * FROM emp_bn WHERE
to_char(hire_date,'mon') LIKE '_a%';
```

105. Write a query in SQL to list the employees whose names containing the character set 'AR' together.

```
SELECT * FROM emp_bn WHERE ename LIKE '%ar%';
```

106. Write a query in SQL to list the employees those who joined in 90's.

```
SELECT * FROM emp_bn WHERE to_char(hire_date,'yy') LIKE '9%';
```

107. Write a query in SQL to list the employees whose ID not starting with digit 68.

```
SELECT emp_id, trim(to_char(emp_id,'99999'))
FROM emp_bn WHERE trim(to_char(emp_id,'99999'))
NOT LIKE '68%';
```

108. Write a query in SQL to list the employees whose names containing the letter 'A'.

```
SELECT * FROM emp_bn WHERE ename LIKE '%a%';
```

109. Write a query in SQL to list the employees whose name is ending with 'S' and six characters long.

```
SELECT * FROM emp_bn WHERE ename LIKE '%s'
AND LENGTH (ename) = 6;
```

110. Write a query in SQL to list the employees who joined in the month having char 'A' at any position.

```
SELECT * FROM emp_bn WHERE to_char (hire_date,'MONTH')
LIKE '%A%';
```

111. Write a query in SQL to list the employees who joined in the month having second char is 'A'.

```
SELECT * FROM emp_bn WHERE to_char(hire_date,'MON')
LIKE '_A%';
```

3 Complex Queries

3.1 Nested Queries

- Some queries require that existing values in the database be fetched and then used in a comparison condition.
- Such queries can be conveniently formulated by using nested queries, which are complete select-from-where blocks within the WHERE clause of another query.
- That other query is called the outer query.
- A complete SELECT query, called a nested query, can be specified within the WHERE-clause of another query, called the outer query

- Many of the previous queries can be specified in an alternative form using nesting

1. Retrieve the name and address of all employees who work for the 'Research' department.

   ```
   SELECT FNAME, LNAME, ADDRESS
   FROM  EMPLOYEE
   WHERE DNO IN  (SELECT  DNUMBER
   FROM DEPARTMENT
   WHERE DNAME='Research' )
   ```

   the comparison operator IN is used

   - The nested query selects the number of the 'Research' department
   - The outer query selects an EMPLOYEE tuple if its DNO value is in the result of either nested query
   - The comparison operator IN compares a value v with a set (or multi-set) of values V, and evaluates to TRUE if v is one of the elements in V
   - In general, we can have several levels of nested queries

2. 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.

   ```
   SELECT DISTINCT Pnumber FROM PROJECT
   WHERE Pnumber IN (SELECT Pnumber FROM PROJECT,
   DEPARTMENT, EMPLOYEE WHERE   Dnum=Dnumber
   AND Mgr_ssn=Ssn AND Lname='Smith')
           OR
   SELECT DISTINCT Pnumber FROM PROJECT
   WHERE Pnumber IN (SELECT  Pno
   FROM WORKS_ON, EMPLOYEE
   WHERE Essn=Ssn AND Lname= 'Smith');
   ```

   SQL allows the use of tuples of values in comparisons by placing them within parentheses.

3. Retrieve the Ssns of all employees who work on the same (project,hours) combination on some project that employee (whose Ssn is equal to '33344555') works on.

   ```
   SELECT DISTINCT Essn FROM WORKS_ON WHERE (Pno,Hours)
   IN (SELECT (Pno,Hours) FROM WORKS_ON
   WHERE Essn='333445555');
   ```

3.2 Other Operators

- =ANY (or =SOME) operator returns TRUE if the value v is equal to some value in the set V and is hence equivalent to IN
- ANY and SOME have the same meaning
- Other operators that can be combined with ANY (or SOME) include $>, >=, <, <=, and <>$
- The keyword ALL also combined with each of these operators
  - Eg., (v>ALL V)
  - Returns TRUE if the value v is greater than all the values in the set (or multiset) V.

1. Retrieve the names of employees whose salary is greater than the salary of all the employees in department 5

```
SELECT Fname,Lname FROM EMPLOYEE
WHERE Salary > ALL
(SELECT Salary FROM EMPLOYEE WHERE Dno=5);
```

3.3 Ambiguity of attribute names

1. Retrieve the name of each employee who has a dependent with the same first name and same sex as the employee.

```
SELECT    E.Fname,E.Lname FROM EMPLOYEE E
WHERE    E.Ssn IN (SELECT  Essn FROM
DEPENDENT WHERE
E.Fname = Dependent_name AND E.Sex=Sex);
```

3.4 Correlated nested queries

- If a condition in the WHERE-clause of a nested query references an attribute of a relation declared in the outer query, the two queries are said to be correlated.
- The result of a correlated nested query is different for each tuple (or combination of tuples) of the relation(s) the outer query.

1. Retrieve the name of each employee who has a dependent with the same first name as the employee.

```
SELECT E.FNAME, E.LNAME
FROM EMPLOYEE AS E WHERE
E.SSN IN (SELECT ESSN FROM DEPENDENT
WHERE ESSN=E.SSN AND E.FNAME=DEPENDENT_NAME);
```

2. A query written with nested SELECT... FROM... WHERE... blocks and using the = or IN comparison operators can always be expressed as a single block query

    ```
    SELECT E.FNAME, E.LNAME
    FROM EMPLOYEE E, DEPENDENT D
    WHERE E.SSN=D.ESSN AND E.FNAME=D.DEPENDENT_NAME;
    ```

    The CONTAINS operator compares two sets (or multi sets) of values, and returns TRUE if one set contains all values in the other set (reminiscent of the division operation of algebra).

3. Retrieve the name of each employee who works on all the projects controlled by department number 5.

    ```
    SELECT FNAME,LNAME FROM EMPLOYEE
    WHERE((SELECT PNO FROM WORKS_ON WHERE SSN=ESSN)
    CONTAINS (SELECT PNUMBER FROM PROJECT WHERE DNUM=5));
    ```

    The Exists Function

    - EXISTS is used to check whether the result of a correlated nested query is empty (contains no tuples) or not
    - The result of EXISTS is a boolean value TRUE or FALSE

4. Retrieve the name of each employee who has a dependent with the same first name as the employee.

    ```
    SELECT  FNAME, LNAME FROM EMPLOYEE
    WHERE EXISTS  (SELECT * FROM DEPENDENT
    WHERE SSN=ESSN AND FNAME=DEPENDENT_NAME);
    ```

5. Retrieve the names of employees who have no dependents.

    ```
    SELECT FNAME, LNAME FROM EMPLOYEE
    WHERE NOT EXISTS   (SELECT * FROM  DEPENDENT
    WHERE SSN=ESSN);
    ```

    - In Q6, the correlated nested query retrieves all DEPENDENT tuples related to an EMPLOYEE tuple. If none exist, the EMPLOYEE tuple is selected.
    - EXISTS is necessary for the expressive power of SQL

6. List the names of managers who have at least one dependent.

    ```
    SELECT Fname, Lname FROM EMPLOYEE
    WHERE EXISTS ( SELECT * FROM DEPENDENT WHERE Ssn=Essn )
    AND EXISTS ( SELECT * FROM DEPARTMENT WHERE Ssn=Mgr_ssn );
    ```

7. Retrieve the name of each employee who works on all the projects controlled by department number 5

   ```
   SELECT Fname, Lname FROM EMPLOYEE WHERE
   NOT EXISTS ((SELECT Pnumber FROM PROJECT WHERE Dnum=5)
   EXCEPT (SELECT Pno FROM WORKS_ON WHERE Ssn=Essn));
   ```

   Explicit sets

   - It is also possible to use an explicit (enumerated) set of values in the WHERE-clause rather than a nested query

8. Retrieve the social security numbers of all employees who work on project number 1, 2, or 3.

   ```
   SELECT DISTINCT ESSN FROM WORKS_ON
   WHERE PNO IN  (1, 2, 3);
   ```

3.5 Aggregate Functions

- Aggregate functions are used to summarize information.
  - from multiple tuples into a single-tuple summary
- Include COUNT, SUM, MAX, MIN, and AVG
- COUNT - returns the number of tuples or values as specified in a query
- SUM, MAX, MIN, and AVG - can be applied to a set or multiset of numeric values and return, respectively, the sum, maximum value, minimum value, and average (mean) of those values.
- The functions MAX and MIN can also be used with attributes that have nonnumeric domains if the domain values have a total ordering among one another.
- Total order - any two values in the domain, it can be determined that one appears before the other in the defined order
- Example
  - DATE, TIME, and TIMESTAMP domains have total orderings on their values
  - alphabetic strings.
- These functions can be used in the SELECT clause or in a HAVING clause
- Find the sum of the salaries of all employees, the maximum salary, the minimum salary, and the average salary among all employees.

  ```
  SELECT  SUM(SALARY), MAX(SALARY), MIN(SALARY),
  AVG(SALARY) FROM EMPLOYEE;
  ```

- Find the maximum salary, the minimum salary, and the average salary among employees who work for the 'Research' department.

```
SELECT MAX(SALARY), MIN(SALARY), AVG(SALARY)
FROM EMPLOYEE, DEPARTMENT WHERE DNO=DNUMBER
AND DNAME='Research'
```

- Retrieve the total number of employees in the company

```
SELECT COUNT (*) FROM EMPLOYEE;
```

- Retrieve the number of employees in the 'Research' department.

```
SELECT  COUNT (*) FROM EMPLOYEE, DEPARTMENT
WHERE DNO=DNUMBER AND DNAME='Research';
```

  - The asterisk (*) refers to the rows (tuples), so COUNT (*) returns the number of rows in the result of the query
  - COUNT function can be used to count values in a column rather than tuples

- Count the number of distinct salary values in the database.

```
SELECT COUNT (DISTINCT Salary)
FROM EMPLOYEE;
```

  - **Note:** duplicates will be eliminated.
  - COUNT function can be used to count values in a column rather than tuples

- Count the number of salary values in the database.

```
SELECT COUNT (Salary) FROM EMPLOYEE;
```

  - Note: duplicate values will be selected (multiset).
  - Any tuples with NULL for SALARY will not be counted.
  - In general, NULL values are discarded when aggregate functions are applied to a particular column (attribute).
  - Produce single tuples or single values.
  - These functions can also be used in selection conditions involving nested queries.
  - A correlated nested query can be specified with an aggregate function, and then use the nested query in the WHERE clause of an outer query.

- Retrieve the names of all employees who have two or more dependents

  ```
  SELECT Lname, Fname FROM EMPLOYEE
  WHERE (SELECT COUNT (*) FROM DEPENDENT
  WHERE Ssn=Essn) >= 2;
  ```

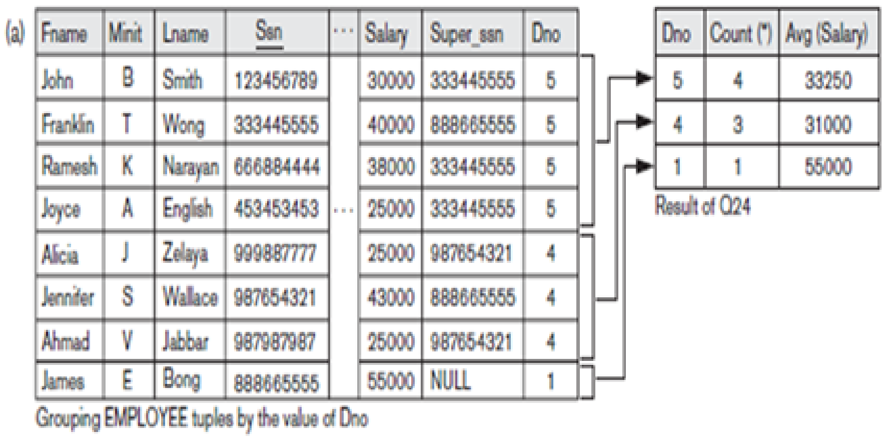
  The correlated nested query counts the number of dependents that each employee has; if this is greater than or equal to two, the employee tuple is selected.

3.6 Grouping

- Grouping is used to create subgroups of tuples before summarization.
- Apply the aggregate functions to subgroups of tuples in a relation.
- Partition the relation into non-overlapping subsets (or groups) of tuples.
- Each group (partition) will consist of the tuples that have the same value of some attribute(s), called the grouping attribute(s).
- Apply the function to each such group independently to produce summary information about each group.
- GROUP BY clause of SQL
- GROUP BY clause specifies the grouping attributes, which should also appear in the SELECT clause
- **Example**For each department, retrieve the department number, the number of employees in the department, and their average salary.

  ```
  SELECT  DNO, COUNT (*), AVG (SALARY)
  FROM EMPLOYEE GROUP BY DNO
  ```

  – the EMPLOYEE tuples are divided into groups–each group having the same value for the grouping attribute DNO.
  – The COUNT and AVG functions are applied to each such group of tuples separately.
  – The SELECT-clause includes only the grouping attribute and the functions to be applied on each group of tuples.
  – A join condition can be used in conjunction with grouping



(a)

Fname	Minit	Lname	Ssn	...	Salary	Super_ssn	Dno
John	B	Smith	123456789		30000	333445555	5
Franklin	T	Wong	333445555		40000	888665555	5
Ramesh	K	Narayan	666884444		38000	333445555	5
Joyce	A	English	453453453	...	25000	333445555	5
Alicia	J	Zelaya	999887777		25000	987654321	4
Jennifer	S	Wallace	987654321		43000	888665555	4
Ahmad	V	Jabbar	987987987		25000	987654321	4
James	E	Bong	888665555		55000	NULL	1

Grouping EMPLOYEE tuples by the value of Dno

Dno	Count (*)	Avg (Salary)
5	4	33250
4	3	31000
1	1	55000

Result of Q24

- If NULLs exist in the grouping attribute, then a separate group is created for all tuples with a NULL value in the grouping attribute.
- For each project, retrieve the project number, project name, and the number of employees who work on that project.

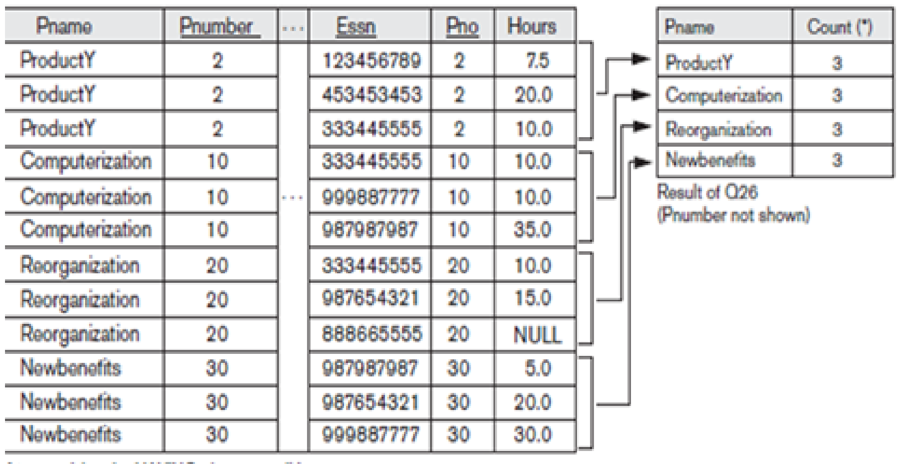
```
SELECT PNUMBER, PNAME, COUNT (*)
FROM PROJECT, WORKS_ON
WHERE PNUMBER=PNO
GROUP BY PNUMBER, PNAME;
```

  - First, joining of the two relations
  - Second, apply grouping
  - Third, apply functions

3.7 The Having-Clause

- to retrieve the values of these functions for only those groups that satisfy certain conditions.
- The HAVING-clause is used for specifying a selection condition on groups (rather than on individual tuples)
- For each project on which more than two employees work , retrieve the project number, project name, and the number of employees who work on that project.

```
SELECT PNUMBER, PNAME, COUNT(*)
FROM PROJECT, WORKS_ON
WHERE PNUMBER=PNO
GROUP BY PNUMBER, PNAME
HAVING COUNT (*) > 2;
```



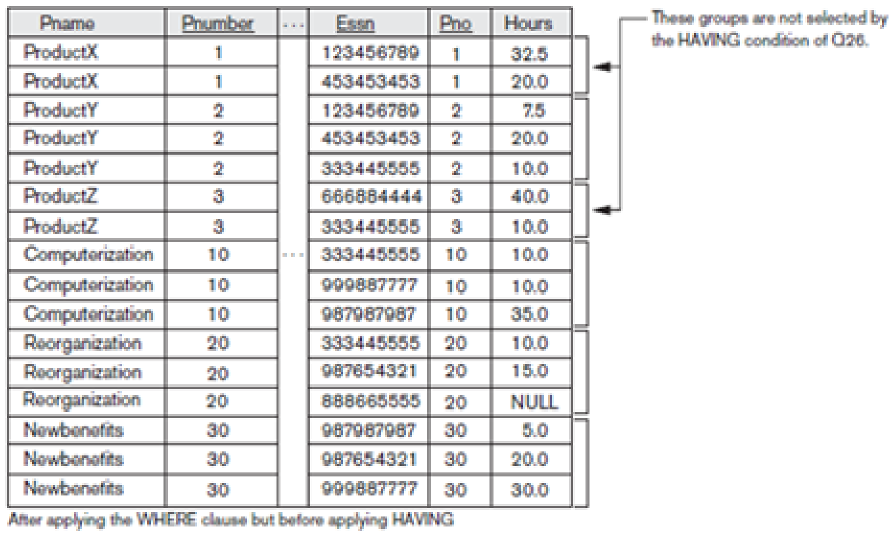
Pname	Pnumber	...	Essn	Pno	Hours
ProductY	2		123456789	2	7.5
ProductY	2		453453453	2	20.0
ProductY	2		333445555	2	10.0
Computerization	10		333445555	10	10.0
Computerization	10	...	999887777	10	10.0
Computerization	10		987987987	10	35.0
Reorganization	20		333445555	20	10.0
Reorganization	20		987654321	20	15.0
Reorganization	20		888665555	20	NULL
Newbenefits	30		987987987	30	5.0
Newbenefits	30		987654321	30	20.0
Newbenefits	30		999887777	30	30.0

After applying the HAVING clause condition

Pname	Count (*)
ProductY	3
Computerization	3
Reorganization	3
Newbenefits	3

Result of Q26
(Pnumber not shown)

(b)



Pname	Pnumber	...	Essn	Pno	Hours
ProductX	1		123456789	1	32.5
ProductX	1		453453453	1	20.0
ProductY	2		123456789	2	7.5
ProductY	2		453453453	2	20.0
ProductY	2		333445555	2	10.0
ProductZ	3		666884444	3	40.0
ProductZ	3		333445555	3	10.0
Computerization	10	...	333445555	10	10.0
Computerization	10		999887777	10	10.0
Computerization	10		987987987	10	35.0
Reorganization	20		333445555	20	10.0
Reorganization	20		987654321	20	15.0
Reorganization	20		888665555	20	NULL
Newbenefits	30		987987987	30	5.0
Newbenefits	30		987654321	30	20.0
Newbenefits	30		999887777	30	30.0

These groups are not selected by the HAVING condition of Q26.

After applying the WHERE clause but before applying HAVING

3.8 Where.....Having

- Selection conditions in the WHERE clause limit the tuples to which functions are applied
- The HAVING clause serves to choose whole groups

- For each project, retrieve the project number, the project name, and the number of employees from department 5 who work on the project.

```
SELECT Pnumber, Pname, COUNT (*)
FROM PROJECT, WORKS_ON, EMPLOYEE
WHERE Pnumber=Pno AND Ssn=Essn AND Dno=5
GROUP BY Pnumber, Pname;
```

- 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.

```
SELECT Dnumber, COUNT (*)
FROM DEPARTMENT, EMPLOYEE
WHERE Dnumber=Dno AND Salary>40000 AND
(SELECT Dno FROM EMPLOYEE
GROUP BY Dno HAVING COUNT (*) > 5);
```

4 PL/SQL (Procedures and Functions)

4.1 Introduction to PL/SQL

- Procedural Language extension for SQL
- PL/SQL is a combination of SQL along with the procedural features of programming languages.
- It was developed by Oracle Corporation in the early 90's to enhance the capabilities of SQL.
- Portable within Oracle data bases
- Callable from any client

4.2 Why PL/SQL

- Acts as host language for stored procedures and triggers.
- Provides the ability to add middle tier business logic to client/server applications.
- Provides Portability of code from one environment to another
- Improves performance of multi-query transactions.
- Provides error handling

4.3 Features of PL/SQL

- PL/SQL is tightly integrated with SQL.
- It offers extensive error checking.
- It offers numerous data types.
- It offers a variety of programming structures.
- It supports structured programming through functions and procedures.
- It supports object-oriented programming.
- It supports developing web applications and server pages.

4.4 Advantages of PL/SQL

- SQL is the standard database language and PL/SQL is strongly integrated with SQL. PL/SQL supports both static and dynamic SQL. Static SQL supports DML operations and transaction control from PL/SQL block. Dynamic SQL is SQL allows embedding DDL statements in PL/SQL blocks.
- PL/SQL allows sending an entire block of statements to the database at one time. This reduces network traffic and provides high performance for the applications.
- PL/SQL gives high productivity to programmers as it can query, transform, and update data in a database.
- PL/SQL saves time on design and debugging by strong features, such as exception handling, encapsulation, data hiding, and object-oriented data types.
- Applications written in PL/SQL are fully portable.
- PL/SQL provides high security level.
- PL/SQL provides access to predefined SQL packages.
- PL/SQL provides support for Object-Oriented Programming.
- PL/SQL provides support for Developing Web Applications and Server Pages.

4.5 Structure of PL/SQL

- PL/SQL is Block Structured
  - A block is the basic unit from which all PL/SQL programs are built. A block can be named (functions and procedures) or anonymous
- Sections of block
  - Header Section

- Header Section
- Header Section
- Executable Section
- Exception Section

4.6 General Syntax

```
HEADER
       Type and Name of block
DECLARE (optional)
- variable declarations; Constants; Cursors;
BEGIN (required)
- SQL statements
   - PL/SQL statements or sub-blocks
EXCEPTION (optional)
- actions to perform when errors occur
END;  (required)
```

Example PL/SQL program

```
DECLARE
a number;
  text1 varchar2(20);
           text2 varchar2(20) := \HI";
BEGIN
         ---------- ---------- ----------
END;
```

- Important Data Types in PL/SQL include NUMBER, INTEGER, CHAR, VARCHAR2, DATE etc

The 'Hello World' Example

```
    DECLARE
    message varchar2(20):= 'Hello, World!'; BEGIN
    dbms_output.put_line(message);
    END;
    /
```

4.7 Data Types for specific columns

```
        Variable_name  Table_name.Column_name%type;
```

- This syntax defines a variable of the type of the referenced column on the referenced table

4.8 General Syntax to declare a variable is

- variable_name datatype [NOT NULL := value];
- variable_name is the name of the variable.

- datatype is a valid PL/SQL datatype.
- NOT NULL is an optional specification on the variable.
- value or DEFAULT values also an optional specification, where you can initialize a variable.
- Each variable declaration is a separate statement and must be terminated by a semicolon.
- to store the current salary of an employee, you can use a variable.

```
DECLARE
salary number(6);
WHERE "salary" is a variable of datatype number and of length 6.
```

- When a variable is specified as NOT NULL, you must initialize the variable when it is declared.

```
DECLARE
salary number(6);
dept varchar2(10) NOT NULL := "HR Dept";
```

- The value of a variable can be changed in the execution or exception section of the PL/SQL Block.
- We can assign values to variables in the two ways given below.
- Directly assign values to variables.
- Assign values to variables directly from the database columns by using a SELECT.. INTO statement.

4.9 Assigning values

- The two ways to assign values to variables
  - Directly assign values to variables.
    - The General Syntax is: variable_name:= value;
  - Assign values to variables directly from the database columns by using a SELECT.. INTO statement.
  - The General Syntax is:

```
SELECT column_name INTO variable_name FROM table_name [WHERE condition];
```

Example 1. The below program will get the salary of an employee with ssn '888665555' and display it on the screen.

```
DECLARE
    var_salary number(10,2);
    var_ssn char(9):= '888665555';
BEGIN
    SELECT salary INTO var_salary
    FROM employee
    WHERE ssn=var_ssn;
    dbms_output.put_line(var_salary);
    dbms_output.put_line(var_ssn);
dbms_output.put_line('The employee '
|| var_ssn || ' has salary ' || var_salary);
END;
/
```

4.10 PL/SQL Constants

- A constant is a value used in a PL/SQL Block that remains unchanged throughout the program. A constant is a user-defined literal value.
- General Syntax to declare a constant is: constant_name CONSTANT datatype := VALUE;
  - constant_name is the name of the constant i.e. similar to a variable name.
  - The word CONSTANT is a reserved word and ensures that the value does not change.
  - **VALUE** - It is a value which must be assigned to a constant when it is declared. You cannot assign a value later.

Valid
DECLARE salary_increase CONSTANT number (3) := 10;

Invalid

```
DECLARE
    salary_increase CONSTANT number(3);
BEGIN
    salary_increase := 100;
    dbms_output.put_line (salary_increase);
END;
/
```

4.11 PL/SQL Scalar Data Types

DataType	**Description**
Numeric	Numeric Values on which arithmetic operations are performed.
Character	Alphanumeric values that represent single characters or strings of characters
Boolean	Logical values on which logical operations are performed
Datetinme	Dates and Times

- PL/SQL provides subtypes of data types. For example, the data type NUMBER has a subtype called INTEGER.

Example 2. Example

```
DECLARE
    num1 INTEGER;
    num2 REAL;
    num3 DOUBLE PRECISION;
BEGIN
    null;
END;
/
```

4.12 PL/SQL Boolean Data Types

- PL/SQL NULL values represent missing or unknown data and they are not an integer, a character, or any other specific data type.
- NULL is not the same as an empty data string or the null character value ”.
- A null can be assigned but it cannot be equated with anything, including itself.

4.13 NULLs in PL/SQL

- The BOOLEAN data type stores logical values that are used in logical operations.
- The logical values are the Boolean values TRUE and FALSE and the value NULL.

4.14 PL/SQL Control Structure

- PL/SQL has a number of control structures which includes:
  - Conditional controls
  - Iterative or loop controls
  - Exception or error controls
- It is these controls, used singly or together, that allow the PL/SQL developer to direct the flow of execution through the program.

4.15 Conditional Controls

- IF....THEN....END IF;
- IF....THEN...ELSE....END IF;
- IF....THEN...ELSIF....THEN....ELSE....END IF;
- The CASE statement selects one sequence of statements to execute.
- However, to select the sequence, the CASE statement uses a selector rather than multiple Boolean expressions.
- A selector is an expression, whose value is used to select one of several alternatives.

The syntax for case statement in PL/SQL is:

```
CASE selector
    WHEN 'value1' THEN S1;
    WHEN 'value2' THEN S2;
    WHEN 'value3' THEN S3;
     ...
    ELSE Sn;  -- default case
    END CASE;
```

Example 3. Example for Case

```
DECLARE
    grade char(1);
BEGIN
        grade := :g;
CASE grade
    when 'A' then dbms_output.put_line('Excellent');
    when 'B' then dbms_output.put_line('Very good');
    when 'C' then dbms_output.put_line('Well done');
    when 'D' then dbms_output.put_line('You passed');
    when 'F' then dbms_output.put_line('Better try  again');
    else dbms_output.put_line('No such grade');
END CASE;
END;
/
```

4.16 PL/SQL Control Structure

Loop

```
LOOP
  ...SQL Statements...
     EXIT;
 END LOOP;
```

WHILE loops

```
WHILE condition LOOP
           ...SQL Statements...
END LOOP;
```

FOR loops

```
FOR <variable(numeric)> IN [REVERSE]  <lowerbound>..<upperbound>
LOOP .... .....
END LOOP;
```

5 Stored Procedures

- A stored procedure or in simple a proc is a named PL/SQL block which performs one or more specific task. This is similar to a procedure in other programming languages.
- A procedure has a header and a body.
  - The header consists of the name of the procedure and the parameters or variables passed to the procedure.
  - The body consists of declaration section, execution section and exception section similar to a general PL/SQL Block.
- A procedure is similar to an anonymous PL/SQL Block but it is named for repeated usage.

5.1 Creating a Procedure

- A procedure is created with the CREATE OR REPLACE PROCEDURE statement.
- The simplified syntax for the CREATE OR REPLACE PROCEDURE statement is as follows:

  -
    ```
    CREATE [OR REPLACE] PROCEDURE procedure_name
    [(parameter_name [IN | OUT | IN OUT] type [, ...])]
    {IS | AS} BEGIN < procedure_body > END procedure_name;
    ```
  - procedure-name specifies the name of the procedure.
  - `[OR REPLACE]`
    option allows modifying an existing procedure.

  - The optional parameter list contains name, mode and types of the parameters. IN represents that value will be passed from outside and OUT represents that this parameter will be used to return a value outside of the procedure.
  - procedure-body contains the executable part.
  - The AS keyword is used instead of the IS keyword for creating a standalone procedure.

Example 1. Example for Creating a Procedure

```
CREATE OR REPLACE PROCEDURE greetings
    AS
    BEGIN
    dbms_output.put_line('Hello World!');
END;
/
```

Output Procedure created.

5.2 Executing a Standalone Procedure

- A standalone procedure can be called in two ways:
  - Using the EXECUTE keyword (sql* plus)
  - Calling the name of the procedure from a PL/SQL block

The above procedure named 'greetings' can be called with the EXECUTE keyword as:
EXECUTE greetings;
The above call would display:
Hello World
PL/SQL procedure successfully completed.
The procedure can also be called from another PL/SQL block:

BEGIN greetings;
END;
/
The above call would display:
Hello World
PL/SQL procedure successfully completed.

5.3 Deleting a Standalone Procedure

- A standalone procedure is deleted with the DROP PROCEDURE statement.
- Syntax for deleting a procedure is:

DROP PROCEDURE procedure-name;

- The greetings procedure can be dropped by using the following statement:

Example 2.

```
BEGIN
DROP PROCEDURE greetings;
END; /
```

6 PL/SQL Function

- A PL/SQL function is same as a procedure except that it returns a value.

6.1 Creating a Function

- A standalone function is created using the CREATE FUNCTION statement. Syntax is:

  ```
  CREATE [OR REPLACE] FUNCTION function_name [(parameter_name [IN
  | OUT | IN OUT] type [, ...])] RETURN return_datatype {IS | AS}
  BEGIN < function_body > END [function_name];
  ```

- function-name specifies the name of the function.
- OR REPLACE option allows modifying an existing function.
- The optional parameter list contains name, mode and types of the parameters. IN represents that value will be passed from outside and OUT represents that this parameter will be used to return a value outside of the procedure.
- The function must contain a return statement.
- RETURN clause specifies that data type you are going to return from the function.
- Function-body contains the executable part.
- The AS keyword is used instead of the IS keyword for creating a standalone function.

Example 1. This function returns the total number of CUSTOMERS in the customers table.
Select * from customers;

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	20000.00
2	Khilan	25	Delhi	1515000.00
3	Kaushik	23	Kota	20000.00
4	Chaitali	25	Mumbai	65000.00
5	Hardik	27	Bhopal	85000.00
6	Komal	22	MP	45000.00

```
CREATE OR REPLACE FUNCTION totalCustomers RETURN number IS
total number(2) := 0;
BEGIN
    SELECT count(*) into total
    FROM customers;
    RETURN total;
END;
/
```

Output Function created.

6.2 Calling a Function

- Creating a function - what the function has to do
- To use a function - call that function to perform the defined task.
  - When a program calls a function, program control is transferred to the called function.
  - A called function performs defined task and when its return statement is executed or when it last end statement is reached, it returns program control back to the main program.
  - To call a function , pass the required parameters along with function name and if function returns a value then you can store returned value.

```
DECLARE c number(2);
    BEGIN c := totalCustomers();
    dbms_output.put_line('Total no. of Customers: ' || c);
END;
/
```

```
When the above code is executed at SQL prompt, it produces the following result:

Total no. of Customers: 6
PL/SQL procedure successfully completed.
```

Example 2. Calling a Function: Example

```
DECLARE
    a number;
    b number;
    c number;
FUNCTION findMax(x IN number, y IN number) RETURN number IS
        z number;
BEGIN
    IF x > y THEN
    z:= x;
    ELSE
    z:= y;
    END IF;
```

```
        RETURN z;
        END;
        BEGIN a:= 23;
        b:= 45;
        c := findMax(a, b);
        dbms_output.put_line(' Maximum of (23,45): ' || c);
        END;
        /
When the above code is executed at SQL prompt, it produces the following result:

Maximum of (23,45): 45 PL/SQL procedure successfully completed.
```

6.3 Specifying Constraints as Assertions

- via declarative assertions, using the
- CREATE ASSERTION statement of the DDL.
- To specify the constraint that the salary of an employee must not be greater than the salary of the manager of the department that the employee works for

Example 3.

```
    CREATE ASSERTION
    SALARY_CONSTRAINT
    CHECK (NOT EXISTS (SELECT *
        FROM EMPLOYEE E, EMPLOYEE M,
        DEPARTMENT D
    WHERE E.Salary>M.Salary
    AND E.Dno=D.Dnumber
    AND D.Mgr_ssn=M.Ssn ) );

This SQL statement creates an assertion to
demand that there's no more than a single president among the employees:

create assertion AT_MOST_ONE_PRESIDENT as CHECK
((select count(*) from EMP e where e.JOB = 'PRESIDENT') <= 1)
```

7 SQL Triggers

- **Objective:** to monitor a database and take action when a condition occurs
- A typical trigger has three components
  - Event
  - Condition

  - action (to be taken when the condition is satisfied)

- A typical trigger has three components
  - event (e.g., an update operation)
  - usually database update operations that are explicitly applied to the database
- Condition
  - determines whether the rule action should be executed
- action (to be taken when the condition is satisfied)
  - The action is usually a sequence of SQL statements
  - also be a database transaction or an external program that will be automatically executed

7.1 Trigger syntax

```
CREATE [OR REPLACE] TRIGGER trigger_name
{BEFORE | AFTER | INSTEAD OF}
{INSERT [OR] | UPDATE [OR] | DELETE}
[OF col_name]
ON table_name
[REFERENCING OLD AS o NEW AS n]
[FOR EACH ROW]
WHEN (condition)
BEGIN
 --- sql statements
END;
```

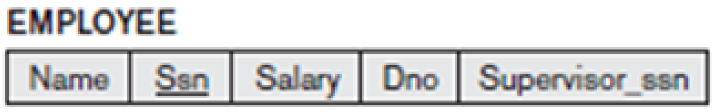
Example 1. SQL Triggers: An Example

```
CREATE OR REPLACE TRIGGER age_violation
BEFORE INSERT OR UPDATE OF
age ON employee
FOR EACH ROW
WHEN (new.age>60)
BEGIN
    inform_supervisor;
END;
```

Example 2. A trigger to compare an employee's age during insert or update operations

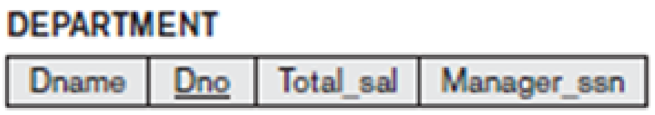
```
CREATE OR REPLACE PROCEDURE INFORM_SUPERVISOR
AS
BEGIN
dbms_output.put_line('AGE VIOLATION!');
END;
```

Example 3.



EMPLOYEE

Name	Ssn	Salary	Dno	Supervisor_ssn



DEPARTMENT

Dname	Dno	Total_sal	Manager_ssn

- the Total_sal attribute is a derived attribute, whose value should be the sum of the salaries of all employees who are assigned to the particular department.
- Maintaining the correct value of such a derived attribute can be done via an active rule.

```
CREATE TRIGGER Total_sal3
AFTER UPDATE OF Dno ON EMPLOYEE
FOR EACH ROW
BEGIN
UPDATE DEPARTMENT
SET Total_sal = Total_sal + NEW.Salary
WHERE Dno = NEW.Dno;
UPDATE DEPARTMENT
SET Total_sal = Total_sal { OLD.Salary
WHERE Dno = OLD.Dno;
END;

CREATE TRIGGER Total_sal4
AFTER DELETE ON EMPLOYEE
FOR EACH ROW
WHEN ( OLD.Dno IS NOT NULL )
UPDATE DEPARTMENT
SET Total_sal = Total_sal { OLD.Salary
WHERE Dno = OLD.Dno;
```