Experiment no. 06

Aim: To aggregate data using Group function.

```
Create table employee
    emp_id      int(3)
    emp_name    varchar(10)
    dept_id     int(5)
    salary      int(10)
    exp         int(2)
    job_title   varchar(20)
);
```

table created.

```
insert into employee values ( 1, 'Alice', 101, 50000, 3,'
insert into employee values( 2, 'Ram', 102, 60000, 5,'M
insert into employee values( 3,' Raj ', 103, Null, 6, Nu
```

(1) sum

```
select sum (salary) from employee;
sum(salary)
110000
```

(2) Average = (Avg())

```
Select Avg(salary) from employee ;
Avg (salary)
55000.00
```

(3) Min()

```
select MAX(salary) from employee;
MAX(salary)
60000
```

(4) max()

Select MAX(salary) from employee;

MAX(salary)

60000

(5) count() - Single count

select (dept_id) from employee

Count (dept_id) from employee

Count (dept_id)

3

(6) Count(*) → counting all rows

select count(*) from employee;

count(*)

3

(7) Count() → using distinct keyword

select count (distinct dept_id) from employee;

count (distinct dept_id);

3

(8) NVL() → number

select emp_name, NVL(salary, 60000) as final salary

from employee

emp_name	final_salary
Alice	50000
Ram	60000
Raj	70000

(9) NVL → string

```
select emp_name, NVL(job_title, 'Not assigned') role
from employee;
```

emp_name	role
Alice	Sales
Ram	HR
Raj	Not assigned

(10) Group By

```
select dept_id, sum(salary) from employee
Group by dept_id;
```

dept_id	sum(salary)
101	50000
102	60000
103	NULL

(11) Illegal Usage

```
Select emp_name, sum(salary) from employee;
```

No result

(12) order by

```
select emp_name, salary from employee order by salary;
```

emp_name	Salary
Raj	NULL
Alice	50000
Ram	60000

13) Order by ⇒ ASC

Select emp_name, salary from employee order by salary ASC;

emp_name	Salary
Raj	NULL
Alice	50 000
Ram	60 000

14) Order by ⇒ DESC

select emp_name, salary from employee order by salary DESC;

emp_name	Salary
Ram	60000
Alice	50000
Raj	NULL

15) Having.

Select dept_id, Avg(salary) from employee
Group by dept_id
Having AVG(salary) > 50000;

dept_id	AVG salary
102	60000.000

Experiment no: 07

Aim: To study of different types of subqueries.

```
SQL> create table emp (eid number(2), name varchar(
     did number(3), jid number(3), salary numb
     Job varchar(5));

Table created.

SQL> insert into emp (eid, name, did, jid, salary, Job)
     values (1, 'a', 100, 501, 10000, 'designer'),
     (2, 'b', 101, 502, 10000, 'programmer'),
     (3, 'c' null, 502, 5000, 'programmer'),
     (null, 'd', null, 502, null, 'programmer'),
     (5, 'e', 101, 503, 20000, 'tester'),
     (6, 'e' 102, 504, 3000, 'tester');

     6 rows created.

SQL> select * from emp;
```

EID	NAME	DID	JID	SALARY	JOB
1	A	100	501	10000	Designer
2	B	101	502	1000	Programmer
3	C		502	500	programmer
	D		501		programmer
5	E	101	503	20000	Tester
6	E	102	504	3000	Tester

6 rows selected.

(A) Single line queries:-

(1) Employee having salary greater than (eid = 2) employee.

SQL> select * from emp
where salary > (select salary from emp where eid=2);

EID	NAME	DID	JID	SALARY	JOB
1	a	100	501	10000	Designer
5	e	101	503	20000	Tester
6	e	102	504	3000	Tester

② Employee whose Jid is same as employee (eid=2) & salary greater than employee (eid=3)

SQL> select * from emp
where jid = (select jid from emp where eid = 2)
AND
SALARY > (select salary from emp where eid = 3);

EID	NAME	DID	JID	SALARY	JOB
2	B	101	502	1000	Programmer

③ Detail of Employee having minimum salary

SQL> select * from emp
where salary = (select min(salary) from emp);

EID	NAME	DID	JID	SALARY	JOB
3	C		502	500	Programmer

④ Minimum salary according to the department minimum salary greater than department (did = 101)

```
SQL> select did, min(salary) as min_salary
     from emp
     Group By did
     Having min(Salary) > (
                Select Min(salary)
                from emp
                where did = 101);
```

DID	Min_Salary
100	10000
102	3000

⑤ Job with lowest average salary.

```
SQL> select jid
     from emp
        Group By jid
          Having AVG(Salary) = (
                select min(avg.sal)
                from (
                     Select AVG(salary)
                       as avg.sal
                     from emp
                      Group By jid)
          );
```

JID
502

(2) multiline queries:-

(1) Employee earning same salary as minimum salary for each department.

```
SQL> select * from emp
     where salary = ( select min(salary)
                      from emp
                      where did = e.did );
```

EID	Name	DID	JID	SALARY	JOB
1	A	100	501	10000	Designer
2	B	101	502	1000	programmer
6	E	102	504	3000	Tester.

2) Employee other than programmer having salary less than any programmer.

```
SQL> select * from emp
     where job='programmer'
     AND salary < Any (
                      select salary
                      from emp
                      where job = 'programmer'
                      );
```

EID	NAME	DID	JID	SALARY	JOB
3	C		502	500	programmer

③ Employee other than programmer having salary greater than all programmers.

```
SQL> select * from emp
        where job = "programmer"
        And salary > All (
            select salary
            from tmp
            where job =
                'programmer' and
                    salary is not null);
```

EID	NAME	DID	JID	SALARY	JOB
6	e	102	504	3000	tester
1	a	100	501	10000	designer
5	e	101	503	20000	tester.

Conclusion:

In this practical we learn about a subquery is a query within another query that helps in breaking down complex problem into smaller steps.

Experiment No.08

Aim: To study different constraints in SQL.

```
SQL> Create table marks,
     (Roll_No, number (2) primary key,
      marks number(2) NOT NULL  check (marks<=30)

     Table created,

SQL> Create table student
     (ID number(2) primary key,
      Name varchar (2)  NOT NULL,
     SIS_ID number(6) UNIQUE,
     Roll_No number(2),
     foreign key (Roll_No) References marks (Roll_No)
       on DELETE CASCADE);

     Table created.

SQL> Insert into marks values ( 1,25);
     1 row created.

SQL> insert into student values (1, 'A', 1001, 1);
     1 row created,

SQL> Desc marks;
```

Name	Null?	Type
Roll_No	NOT NULL	NUMBER(2)
Marks		NUMBER (2)

SQL> Desc Student;

Name	Null?	Type
ID	NOT NULL	NUMBER(2)
Name	NOT NULL	VARCHAR2(20)
SIS_ID		NUMBER(8)
Roll_No		NUMBER(2)

① NOT NULL

SQL> Insert into student values
(2, Null, 1003, 1);

ERROR at line 1:
ORA-01400: Cannot insert null into
("SYSTEM"."STUDENT"."NAME")

2) UNIQUE key

SQL> Insert into student values
(3, 'C', 1001, 1);
ERROR at line 1:
ORA-00001: Unique constraint (SYSTEM.SYS_C00[illegible])
violated,

③ Primary key

```
SQL> Insert into student values
     (1, 'B', 1002, 1);

ERROR at line 1:
ORA-00001: unique constraint (SYSTEM.SYS_
C008325) violated.
```

④ Check

```
SQL> Insert into marks values (2,35);
ERROR at line 1:
ORA-02290: CHECK constraint (SYSTEM.
SYS_C008322) violated.
```

⑤ foreign Key :-

```
SQL> Insert into student values
     (4, 'D', 1004, 5);
ERROR at line 1:
ORA-00291: integrity constraint. (SYSTEM.
SYS_C008327) violated.
```

⑥ View:-

```
SQL> select constraint_name, constraint_type,
     table_name
     from user_constraint
     where table_name = 'student';
```

Constraint_name	constraint_type	Table_name
SYS_C008324	R (foreign key)	student
SYS_C008327	R (foreign key)	student
SYS_C008325	P (primary key)	student
SYS_C008326	U (unique)	student

⑦ Disable

```
SQL> ALTER table student
     Disable constraint SYS_C008327;
Table altered.

SQL> insert into student values (5,'f', 1005, 99);
1 row created.
```

⑧ Enable

```
SQL> Alter table student
     enable constraint SYS_C008327;
Table altered.
```

SQL> Insert into student values (5, 'E', 1005, 9);

Error at line 1:

ORA-02291: integrity constraint (SYSTEM.SYS_C008327) violated.

(9) Drop

SQL> Alter table student
Drop primary key;

Table altered.

SQL> Desc student;

Name	Null?	Type
ID		NUMBER(2)
Name	Not Null	VARCHAR(2)
SIS_ID		NUMBER(6)
Roll_No		NUMBER(2)

(10) ADD

SQL> Alter table student
Add constraint PK_student primary key (ID)
Table altered.

SQL> Desc student

Name	Null?	Type
ID	NOT NULL	NUMBER(2)
Name	NOT NULL	VARCHAR(2)
SIS_ID		NUMBER(6)
Roll_No		NUMBER(2)

(11) Cascade

SQL> select * from student;

ID	Name	SIS-ID	Roll_No
1	A	1001	1
5	E	1005	99

SQL> delete from marks
where Roll_no = 1;
1 row deleted.

SQL> select * from student;

ID	Name	SIS-ID	Roll_No
5	E	1005	99

Conclusion: In this practical, we successfully learned how to apply various constraints in a database to ensure data integrity and accuracy.