NEW SUMMIT COLLEGE

Santinagar, Kathmandu

(Affiliated to Tribhuvan University)

LABORATORY ASSIGNMENT

REPORT OF

ADVANCED DATABASE (CSC 461)

Submitted by:

Name: .....

Roll No: .....

Semester: Eighth

Faculty: Science and Technology

Level: Bachelor

Program: Computer Science and Information Technology

Submitted to:

Instructor: Indra Chaudhary

Advanced Database Laboratory Assignment Log Sheet

Task No.	Title	Signature
Using Oracle's HR Schema [Example Tablespace]		
1	Retrieving Data using SQL SELECT statement	
2	Restricting and sorting data	
3	Using Single-Row Functions to Customize Output	
4	Using Conversion Functions and Conditional Expressions	
5	Reporting Aggregated Data using the Group Functions	
6	Displaying Data from Multiple Tables	
7	Using subqueries to solve queries	
8	Using Set Operators	
Using Own Tablespace and Schema		
9	Using DDL statements	
10	Using DCL statements	
11	Using DML statement	
12	Create Other schema objects	
13	ORDBMS (based on Oracle) Tasks	
14	NoSQL (based on MongoDB) Tasks	

ADVANCED DATABASE | LABORATORY ASSIGNMENT

Task 1: Retrieving Data using SQL SELECT statement

1. Determine the structure of DEPARTMENTS table and contents.

```
describe departments;
```

Script Output | Task completed in 1.882 seconds

```
Show tables handled.
Name            Null?    Type
--------------- -------- ------------
DEPARTMENT_ID   NOT NULL NUMBER(4)
DEPARTMENT_NAME NOT NULL VARCHAR2(30)
MANAGER_ID               NUMBER(6)
LOCATION_ID              NUMBER(4)
```

```
select * from departments;
```

Query Result | SQL | All Rows Fetched: 27 in 0.479 seconds

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	30	Purchasing	114	1700
4	40	Human Resources	203	2400
5	50	Shipping	121	1500
6	60	IT	103	1400
7	70	Public Relations	204	2700
8	80	Sales	145	2500
9	90	Executive	100	1700
10	100	Finance	108	1700
11	110	Accounting	205	1700
12	120	Treasury	(null)	1700
13	130	Corporate Tax	(null)	1700
14	140	Control And Credit	(null)	1700
15	150	Shareholder Services	(null)	1700
16	160	Benefits	(null)	1700

2. Determine the structure of Employees table.

```
describe employees;
```

```
Name           Null?    Type
-------------- -------- ------------
EMPLOYEE_ID    NOT NULL NUMBER(6)
FIRST_NAME              VARCHAR2(20)
LAST_NAME      NOT NULL VARCHAR2(25)
EMAIL          NOT NULL VARCHAR2(25)
PHONE_NUMBER            VARCHAR2(20)
HIRE_DATE      NOT NULL DATE
JOB_ID         NOT NULL VARCHAR2(10)
SALARY                  NUMBER(8,2)
COMMISSION_PCT          NUMBER(2,2)
MANAGER_ID              NUMBER(6)
DEPARTMENT_ID           NUMBER(4)
```

3. The HR department wants a query to display the last name, job code, hire date and employee number for each employee, with the employee number appearing first. Provide an alias STARTDATE for the HIRE_DATE column.

```
select employee_id, last_name, job_id, hire_date startdate
from employees;
```

Query Result

SQL | Fetched 50 rows in 0.294 seconds

	EMPLOYEE_ID	LAST_NAME	JOB_ID	STARTDATE
1	100	King	AD_PRES	17-JUN-03
2	101	Kochhar	AD_VP	21-SEP-05
3	102	De Haan	AD_VP	13-JAN-01
4	103	Hunold	IT_PROG	03-JAN-06
5	104	Ernst	IT_PROG	21-MAY-07
6	105	Austin	IT_PROG	25-JUN-05
7	106	Pataballa	IT_PROG	05-FEB-06
8	107	Lorentz	IT_PROG	07-FEB-07
9	108	Greenberg	FI_MGR	17-AUG-02
10	109	Faviet	FI_ACCOUNT	16-AUG-02
11	110	Chen	FI_ACCOUNT	28-SEP-05
12	111	Sciarra	FI_ACCOUNT	30-SEP-05
13	112	Urman	FI_ACCOUNT	07-MAR-06
14	113	Popp	FI_ACCOUNT	07-DEC-07
15	114	Raphaely	PU_MAN	07-DEC-02
16	115	Khoo	PU_CLERK	18-MAY-03

4. The HR department needs a query to display all unique job codes from the EMPLOYEES table.

```
select distinct job_id
from employees;
```

Query Result

SQL | All Rows Fetched: 19

	JOB_ID
1	AC_ACCOUNT
2	AC_MGR
3	AD_ASST
4	AD_PRES
5	AD_VP
6	FI_ACCOUNT
7	FI_MGR
8	HR_REP
9	IT_PROG
10	MK_MAN
11	MK_REP
12	PR_REP
13	PU_CLERK
14	PU_MAN
15	SA_MAN
16	SA_REP

5. The HR department wants more descriptive column heading for its report on employees. Modify the Task_1_3 to Name the column heading EMP #, Employee, Job and Hire Date, respectively.

```
select employee_id "Emp#", last_name "Employee", job_id "Job", hire_date "Hire Date"
from employees;
```

Query Result

SQL | Fetched 50 rows in 0.006 seconds

	Emp#	Employee	Job	Hire Date
1	100	King	AD_PRES	17-JUN-03
2	101	Kochhar	AD_VP	21-SEP-05
3	102	De Haan	AD_VP	13-JAN-01
4	103	Hunold	IT_PROG	03-JAN-06
5	104	Ernst	IT_PROG	21-MAY-07
6	105	Austin	IT_PROG	25-JUN-05
7	106	Pataballa	IT_PROG	05-FEB-06
8	107	Lorentz	IT_PROG	07-FEB-07
9	108	Greenberg	FI_MGR	17-AUG-02
10	109	Faviet	FI_ACCOUNT	16-AUG-02
11	110	Chen	FI_ACCOUNT	28-SEP-05
12	111	Sciarra	FI_ACCOUNT	30-SEP-05
13	112	Urman	FI_ACCOUNT	07-MAR-06
14	113	Popp	FI_ACCOUNT	07-DEC-07
15	114	Raphaely	PU_MAN	07-DEC-02
16	115	Khoo	PU_CLERK	18-MAY-03
17	116	Baida	PU_CLERK	24-DEC-05

6. The HR department has requested a report of all employees and their job IDs. Display the last name concatenated with the job ID (Separated by a comma and space) and name the column *Employee and Title*.

```
select last_name||', '||job_id "Employee and Title" from employees;
```

Query Result

SQL | Fetched 50 rows in 0.048 seconds

	Employee and Title
1	Abel, SA_REP
2	Ande, SA_REP
3	Atkinson, ST_CLERK
4	Austin, IT_PROG
5	Baer, PR_REP
6	Baida, PU_CLERK
7	Banda, SA_REP
8	Bates, SA_REP
9	Bell, SH_CLERK
10	Bernstein, SA_REP
11	Bissot, ST_CLERK
12	Bloom, SA_REP
13	Bull, SH_CLERK
14	Cabrio, SH_CLERK
15	Cambrault, SA_MAN
16	Cambrault, SA_REP
17	Chen, FI_ACCOUNT
18	Chung, SH_CLERK

Task 2: Restricting and sorting data

The HR department needs your assistance with creating some queries

1. Because of budget issues, the HR department needs a report that displays the last name and salary employees earning more than $12,000.

```
select last_name, salary
from employees
where salary > 12000;
```

Query Result

SQL | All Rows Fetched: 8 in 0

	LAST_NAME	SALARY
1	King	24000
2	Kochhar	17000
3	De Haan	17000
4	Greenberg	12008
5	Russell	14000
6	Partners	13500
7	Hartstein	13000
8	Higgins	12008

2. Write a query that displays the last name and department number for employee number 176.

```
select last_name, department_id
from employees
where employee_id = 176;
```

	LAST_NAME	DEPARTMENT_ID
1	Taylor	80

3. Write a query that displays the last name and salary for all employees whose salary is not in the 5,000 - 12,000 range.

```
select last_name, salary
from employees
where salary>12000 or salary<5000;
```

	LAST_NAME	SALARY
1	King	24000
2	Kochhar	17000
3	De Haan	17000
4	Austin	4800
5	Pataballa	4800
6	Lorentz	4200
7	Greenberg	12008
8	Khoo	3100
9	Baida	2900
10	Tobias	2800
11	Himuro	2600
12	Colmenares	2500
13	Nayer	3200
14	Mikkilineni	2700
15	Landry	2400
16	Markle	2200
17	Bissot	3300
18	Atkinson	2800

4. Run a query to display the last name, job ID, and start date for the employees whose last names are Matos and Taylor. Order the query in ascending order by start date.

```
select last_name, job_id, hire_date
from employees
where last_name='Matos' or last_name='Taylor'
order by hire_date;
```

	LAST_NAME	JOB_ID	HIRE_DATE
1	Taylor	SH_CLERK	24-JAN-06
2	Matos	ST_CLERK	15-MAR-06
3	Taylor	SA_REP	24-MAR-06

5. Display the last name and department number of all employees in departments 20 or 50 in ascending alphabetical order by name.

```
select last_name, department_id
from employees
where department_id in (20,50)
order by last_name;
```

Query Result

SQL | All Rows Fetched: 47 in 0.038 seconds

	LAST_NAME	DEPARTMENT_ID
1	Atkinson	50
2	Bell	50
3	Bissot	50
4	Bull	50
5	Cabrio	50
6	Chung	50
7	Davies	50
8	Dellinger	50
9	Dilly	50
10	Everett	50
11	Fay	20
12	Feeney	50
13	Fleaur	50
14	Fripp	50
15	Gates	50
16	Gee	50
17	Geoni	50

6. Modify task_2_3 to list the last name and salary of employees who earn between $5,000 and $12,000 and are in department 20 or 50. Label the columns Employee and Monthly salary, respectively.

```
select last_name "Employee", salary "Monthly Salary"
from employees
where (salary between 5000 and 12000) and department_id
in (20,50);
```

	Employee	Monthly Salary
1	Weiss	8000
2	Fripp	8200
3	Kaufling	7900
4	Vollman	6500
5	Mourgos	5800
6	Fay	6000

7. The HR department needs a report that displays the last name and hire date for all employees who were hired in 1994.

```
select last_name, hire_date
```

```
from employees
where hire_date like '%94';
```

Query Result

SQL | All Rows Fetched: 0 in 0.026 seconds

LAST_N...	HIRE_D...

8. Display the last name and job title of all employees who do not have a manager.

```
select last_name, job_id
from employees
where manager_id is null;
```

	LAST_NAME	JOB_ID
1	King	AD_PRES

9. Display the last name, salary and commission for all employees who earn commissions. sort data in descending order of salary and commissions.

```
select last_name, salary, commission_pct
from employees
where commission_pct is not null
order by salary, commission_pct;
```

	LAST_NAME	SALARY	COMMISSION_PCT
1	Kumar	6100	0.1
2	Banda	6200	0.1
3	Johnson	6200	0.1
4	Ande	6400	0.1
5	Lee	6800	0.1
6	Grant	7000	0.15
7	Tuvault	7000	0.15
8	Sewall	7000	0.25
9	Marvins	7200	0.1
10	Bates	7300	0.15
11	Smith	7400	0.15
12	Cambrault	7500	0.2
13	Doran	7500	0.3
14	Olsen	8000	0.2
15	Smith	8000	0.3
16	Livingston	8400	0.2
17	Taylor	8600	0.2
18	Hutton	8800	0.25

10. Display all employee last names in which the third letter of the name is *a*.

```
select last_name
from employees
where last_name like '__a%';
```

	LAST_NAME
1	Grant
2	Grant
3	Whalen

11. Display the last names of all employees who have both an *a* and *e* in their last names.

```
select last_name
from employees
where last_name like '%a%' and last_name like '%e%';
```

	LAST_NAME
1	Baer
2	Bates
3	Colmenares
4	Davies
5	De Haan
6	Faviet
7	Fleaur
8	Gates
9	Hartstein
10	Markle
11	Nayer
12	Partners
13	Patel
14	Philtanker
15	Raphaely
16	Sewall
17	Whalen

12. Display the last name, job and salary for all employees whose job is either that of a sales representative (SA_REP) or a stock clerk (ST_CLERK), and whose salary is not equal to $2,500, $3,500, or $7,000.

```
select last_name, job_id, salary
from employees
where (job_id='SA_REP' or job_id='ST_CLERK') and salary
not in(2500,3500,7000);
```

Query Result

SQL | All Rows Fetched: 43 in 0.004 seconds

	LAST_NAME	JOB_ID	SALARY
1	Nayer	ST_CLERK	3200
2	Mikkilineni	ST_CLERK	2700
3	Landry	ST_CLERK	2400
4	Markle	ST_CLERK	2200
5	Bissot	ST_CLERK	3300
6	Atkinson	ST_CLERK	2800
7	Olson	ST_CLERK	2100
8	Mallin	ST_CLERK	3300
9	Rogers	ST_CLERK	2900
10	Gee	ST_CLERK	2400
11	Philtanker	ST_CLERK	2200
12	Ladwig	ST_CLERK	3600
13	Stiles	ST_CLERK	3200
14	Seo	ST_CLERK	2700
15	Davies	ST_CLERK	3100
16	Matos	ST_CLERK	2600
17	Tucker	SA_REP	10000
18	Bernstein	SA_REP	9500

13. Display the last name, salary, and commission for all employees whose commission amount is 20%.

```
select last_name "Employee", salary "Monthly Salary",
commission_pct
from employees
where commission_pct=.2;
```

	Employee	Monthly Salary	COMMISSION_PCT
1	Zlotkey	10500	0.2
2	Olsen	8000	0.2
3	Cambrault	7500	0.2
4	Bloom	10000	0.2
5	Fox	9600	0.2
6	Taylor	8600	0.2
7	Livingston	8400	0.2

14. Members of the HR department want to have more flexibility with the queries that you are writing. They would like a report that displays the last name and salary of employees who earn more than an amount that the user specifies after a prompt.

```
select last_name, salary
from employees
where salary> &Salary;
```

For &Salary=12000

	LAST_NAME	SALARY
1	King	24000
2	Kochhar	17000
3	De Haan	17000
4	Greenberg	12008
5	Russell	14000
6	Partners	13500
7	Hartstein	13000
8	Higgins	12008

15. Create a query that prompts the user for third_field and generates employee ID, last name from employees table. Also sort the output by third_field.

(Don't Know)

Task 3: Using Single-Row Functions to Customize Output

1. The HR department needs a report to display the employee number, last name, salary, and salary increased by 15.5% (expressed as a whole number) for each employee. Label the column New Salary.

```
select employee_id, last_name, salary,
salary+(salary*15.5/100) "New Salary"
from employees;
```

	EMPLOYEE_ID	LAST_NAME	SALARY	New Salary
1	100	King	24000	27720
2	101	Kochhar	17000	19635
3	102	De Haan	17000	19635
4	103	Hunold	9000	10395
5	104	Ernst	6000	6930
6	105	Austin	4800	5544
7	106	Pataballa	4800	5544
8	107	Lorentz	4200	4851
9	108	Greenberg	12008	13869.24
10	109	Faviet	9000	10395
11	110	Chen	8200	9471
12	111	Sciarra	7700	8893.5
13	112	Urman	7800	9009
14	113	Popp	6900	7969.5
15	114	Raphaely	11000	12705
16	115	Khoo	3100	3580.5

2. Modify script of task_3_1 to add a column that subtracts the old salary from the new salary. Label the column **Increase**.

```
select employee_id, last_name, salary,
salary+(salary*15.5/100) "New Salary",
(salary+(salary*15.5/100))-salary "Increase"
from employees;
```

	EMPLOYEE_ID	LAST_NAME	SALARY	New Salary	Increase
1	100	King	24000	27720	3720
2	101	Kochhar	17000	19635	2635
3	102	De Haan	17000	19635	2635
4	103	Hunold	9000	10395	1395
5	104	Ernst	6000	6930	930
6	105	Austin	4800	5544	744
7	106	Pataballa	4800	5544	744
8	107	Lorentz	4200	4851	651
9	108	Greenberg	12008	13869.24	1861.24
10	109	Faviet	9000	10395	1395
11	110	Chen	8200	9471	1271
12	111	Sciarra	7700	8893.5	1193.5
13	112	Urman	7800	9009	1209
14	113	Popp	6900	7969.5	1069.5
15	114	Raphaely	11000	12705	1705
16	115	Khoo	3100	3580.5	480.5
17	116	Baida	2900	3349.5	449.5
18	117	Tobias	2800	3234	434

3. Write a query that displays the last name and the length of the last name for all employees whose name starts with letters J, A or M. Give each column an appropriate label. Sort the results by the employees' last names.

```
select last_name "Name", length(last_name) "Length of Name"
from employees
where last_name like 'J%' or last_name like 'A%' or last_name like 'M%'
order by last_name;
```

	Name	Length of Name
1	Abel	4
2	Ande	4
3	Atkinson	8
4	Austin	6
5	Johnson	7
6	Jones	5
7	Mallin	6
8	Markle	6
9	Marlow	6
10	Marvins	7
11	Matos	5
12	Mavris	6
13	Mccain	6
14	Mcewen	6
15	Mikkilineni	11
16	Mourgos	7

4. Write a query that displays the last name (with the first letter uppercase and all other letters lowercase) and the length of the last name for all employees whose name starts with the letter J, A or M. Give each column an appropriate label. Sort the results by the employees' last names.

```
select initcap(last_name) "Name", length(last_name) "Length of Name"
from employees
where last_name like 'J%' or last_name like 'A%' or last_name like 'M%'
order by last_name;
```

	Name	Length of Name
1	Abel	4
2	Ande	4
3	Atkinson	8
4	Austin	6
5	Johnson	7
6	Jones	5
7	Mallin	6
8	Markle	6
9	Marlow	6
10	Marvins	7
11	Matos	5
12	Mavris	6
13	Mccain	6
14	Mcewen	6
15	Mikkilineni	11
16	Mourgos	7

5. Create a query to display the last name and salary for all employees. Format the salary to be 15 characters long, left-padded with the "$" symbol. Label the column **SALARY**.

```
select last_name, lpad(salary,15,'$') Salary
from employees;
```

	LAST_NAME	SALARY
1	King	$$$$$$$$$$24000
2	Kochhar	$$$$$$$$$$17000
3	De Haan	$$$$$$$$$$17000
4	Hunold	$$$$$$$$$$$$9000
5	Ernst	$$$$$$$$$$$$6000
6	Austin	$$$$$$$$$$$$4800
7	Pataballa	$$$$$$$$$$$$4800
8	Lorentz	$$$$$$$$$$$$4200
9	Greenberg	$$$$$$$$$$12008
10	Faviet	$$$$$$$$$$$$9000
11	Chen	$$$$$$$$$$$$8200
12	Sciarra	$$$$$$$$$$$$7700
13	Urman	$$$$$$$$$$$$7800
14	Popp	$$$$$$$$$$$$6900
15	Raphaely	$$$$$$$$$$11000
16	Khoo	$$$$$$$$$$$$3100
17	Baida	$$$$$$$$$$$$2900
18	Tobias	$$$$$$$$$$$$2800

6. Write a query to display the current date of the oracle database server.

```
select sysdate from dual;
```

	SYSDATE
1	19-SEP-23

7. The HR department wants to find the duration of employment for each employee. For each employee, display the last name and calculate the number of months between today and the date on which the employee was hired. Label the column MONTHS_WORKED. Order your results by the number of months employed. Round the number of months up to the closest whole number.

```
Select last_name, round(months_between(sysdata, hire_date), 0) Months_worked from employees;
```

	LAST_NAME	MONTHS_WORKED
1	King	243
2	Kochhar	216
3	De Haan	272
4	Hunold	213
5	Ernst	196
6	Austin	219
7	Pataballa	211
8	Lorentz	199
9	Greenberg	253
10	Faviet	253
11	Chen	216
12	Sciarra	216
13	Urman	210
14	Popp	189
15	Raphaely	249
16	Khoo	244
17	Baida	213
18	Tobias	218

Task 4: Using Conversion Functions and Conditional Expressions

1. Display the last name, hire date, and day of the week on which the employee started. Label the column DAY.

```
select last_name, hire_date, to_char(hire_date,'Day') "Day"
```

```
from employees;
```

	LAST_NAME	HIRE_DATE	Day
1	King	17-JUN-03	Tuesday
2	Kochhar	21-SEP-05	Wednesday
3	De Haan	13-JAN-01	Saturday
4	Hunold	03-JAN-06	Tuesday
5	Ernst	21-MAY-07	Monday
6	Austin	25-JUN-05	Saturday
7	Pataballa	05-FEB-06	Sunday
8	Lorentz	07-FEB-07	Wednesday
9	Greenberg	17-AUG-02	Saturday
10	Faviet	16-AUG-02	Friday
11	Chen	28-SEP-05	Wednesday
12	Sciarra	30-SEP-05	Friday
13	Urman	07-MAR-06	Tuesday
14	Popp	07-DEC-07	Friday
15	Raphaely	07-DEC-02	Saturday
16	Khoo	18-MAY-03	Sunday
17	Baida	24-DEC-05	Saturday

2. Create a query that displays the employees' last names and commission amounts. If an employee does not earn commission, show "No Commission." Label the column COMM.

```
select last_name, NVL2(commission_pct,
To_CHAR(commission_pct), 'no commission') "COMM"
FROM employees
```

	LAST_NAME	COMM
1	King	no commission
2	Kochhar	no commission
3	De Haan	no commission
4	Hunold	no commission
5	Ernst	no commission
6	Austin	no commission
7	Pataballa	no commission
8	Lorentz	no commission
9	Greenberg	no commission
10	Faviet	no commission
11	Chen	no commission
12	Sciarra	no commission
13	Urman	no commission
14	Popp	no commission
15	Raphaely	no commission
16	Khoo	no commission
17	Baida	no commission
18	Tobias	no commission

3. Using CASE function and Decode Function, write a query that displays the grade of all employees based on the value of the column JOB_ID, using the following data:

Job	**Grade**
AD_PRES	A

ST_MAN	B
IT_PROG	C
SA_REP	D
ST_CLERK	E
None of the above	0

```
SELECT job_id, decode (job_id,
'ST_CLERK', 'E',
'SA_REP', 'D',
'IT_PROG', 'C',
'ST_MAN', 'B',
'AD_PRES', 'A',
'0') GRADE
FROM employees;
```

	JOB_ID	GRADE
1	AC_ACCOUNT	0
2	AC_MGR	0
3	AD_ASST	0
4	AD_PRES	A
5	AD_VP	0
6	AD_VP	0
7	FI_ACCOUNT	0
8	FI_ACCOUNT	0
9	FI_ACCOUNT	0
10	FI_ACCOUNT	0
11	FI_ACCOUNT	0
12	FI_MGR	0
13	HR_REP	0
14	IT_PROG	C
15	IT_PROG	C
16	IT_PROG	C
17	IT_PROG	C
18	IT_PROG	C

Task 5: Reporting Aggregated Data using the Group Functions.

1. Find the highest, lowest, sum, and average salary of all employees. Label the columns *Maximum, Minimum, Sum and Average* respectively. Round your results to the nearest whole number.

```
SELECT ROUND(MAX(salary),0) "Maximum",
ROUND(MIN(salary),0) "Minimum",
ROUND(SUM(salary),0) "Sum",
ROUND(AVG(salary),0) "Average"
FROM employees;
```

	Maximum	Minimum	Sum	Average
1	24000	2100	691416	6462

2. Modify the query in task_5_1 to display the minimum, maximum, sum and average salary for each job type.

```
SELECT job_id, ROUND(MAX(salary),0) "Maximum",
ROUND(MIN(salary),0) "Minimum",
ROUND(SUM(salary),0) "Sum",
ROUND(AVG(salary),0) "Average"
FROM employees
GROUP BY job_id;
```

	JOB_ID	Maximum	Minimum	Sum	Average
1	IT_PROG	9000	4200	28800	5760
2	AC_MGR	12008	12008	12008	12008
3	AC_ACCOUNT	8300	8300	8300	8300
4	ST_MAN	8200	5800	36400	7280
5	PU_MAN	11000	11000	11000	11000
6	AD_ASST	4400	4400	4400	4400
7	AD_VP	17000	17000	34000	17000
8	SH_CLERK	4200	2500	64300	3215
9	FI_ACCOUNT	9000	6900	39600	7920
10	FI_MGR	12008	12008	12008	12008
11	PU_CLERK	3100	2500	13900	2780
12	SA_MAN	14000	10500	61000	12200
13	MK_MAN	13000	13000	13000	13000
14	PR_REP	10000	10000	10000	10000
15	AD_PRES	24000	24000	24000	24000
16	SA_REP	11500	6100	250500	8350
17	MK_REP	6000	6000	6000	6000
18	ST_CLERK	3600	2100	55700	2785

3. Write a query to display the number of people with the same job.

```
SELECT job_id, COUNT(*)
FROM employees
GROUP BY job_id;
```

	JOB_ID	COUNT(*)
1	AC_ACCOUNT	1
2	AC_MGR	1
3	AD_ASST	1
4	AD_PRES	1
5	AD_VP	2
6	FI_ACCOUNT	5
7	FI_MGR	1
8	HR_REP	1
9	IT_PROG	5
10	MK_MAN	1
11	MK_REP	1
12	PR_REP	1
13	PU_CLERK	5
14	PU_MAN	1
15	SA_MAN	5
16	SA_REP	30
17	SH_CLERK	20
18	ST_CLERK	20

4. Find the difference between the highest and lowest salaries. Label the column DIFFERENCE.

```
SELECT MAX(salary) - MIN(salary) DIFFERENCE
FROM employees;
```

Script Output × Query Result ×

SQL | All Rows Fetched: 1 in 0.028 seconds

	DIFFERENCE
1	21900

5. Create a query to display the manager number and the salary of the lowest-paid employee for the manager. Exclude anyone whose manager is not known. Exclude any groups where the minimum salary is $6,000 or less. Sort the output in descending order of salary.

```
SELECT manager_id, MIN(salary)
FROM employees
WHERE manager_id IS NOT NULL
GROUP BY manager_id
HAVING MIN(salary) > 6000
ORDER BY MIN(salary) DESC;
```

	MANAGER_ID	MIN(SALARY)
1	102	9000
2	205	8300
3	145	7000
4	146	7000
5	108	6900
6	147	6200
7	149	6200
8	148	6100

Task 6: Displaying Data from Multiple Tables

1. Write a query for the HR department to produce the addresses of all the departments. Use the LOCATIONS and COUNTRIES tables. Show the location Id, street address, city, state or province and country in the output. Using inner join.

```
SELECT location_id, street_address, city,
state_province, country_name
FROM locations l
INNER JOIN countries c
on l.country_id = c.country_id;
```

	LOCATION_ID	STREET_ADDRESS	CITY	STATE_PROVINCE	COUNTRY_NAME
1	1000	1297 Via Cola di Rie	Roma	(null)	Italy
2	1100	93091 Calle della Testa	Venice	(null)	Italy
3	1200	2017 Shinjuku-ku	Tokyo	Tokyo Prefecture	Japan
4	1300	9450 Kamiya-cho	Hiroshima	(null)	Japan
5	1400	2014 Jabberwocky Rd	Southlake	Texas	United States of America
6	1500	2011 Interiors Blvd	South San Francisco	California	United States of America
7	1600	2007 Zagora St	South Brunswick	New Jersey	United States of America
8	1700	2004 Charade Rd	Seattle	Washington	United States of America
9	1800	147 Spadina Ave	Toronto	Ontario	Canada
10	1900	6092 Boxwood St	Whitehorse	Yukon	Canada
11	2000	40-5-12 Laogianggen	Beijing	(null)	China
12	2100	1298 Vileparle (E)	Bombay	Maharashtra	India
13	2200	12-98 Victoria Street	Sydney	New South Wales	Australia
14	2300	198 Clementi North	Singapore	(null)	Singapore
15	2400	8204 Arthur St	London	(null)	United Kingdom
16	2500	Magdalen Centre, The Oxford Science Park	Oxford	Oxford	United Kingdom
17	2600	9702 Chester Road	Stretford	Manchester	United Kingdom
18	2700	Schwanthalerstr. 7031	Munich	Bavaria	Germany

2. The HR department need a report of all employees. Write a query to display the last name, department number, and department name for all employees.

```
select e.last_name,d.department_id,d.department_name
from employees e join departments d
on(e.department_id = d.department_id) order by
e.department_id
```

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Fay	20	Marketing
3	Hartstein	20	Marketing
4	Tobias	30	Purchasing
5	Colmenares	30	Purchasing
6	Baida	30	Purchasing
7	Raphaely	30	Purchasing
8	Khoo	30	Purchasing
9	Himuro	30	Purchasing
10	Mavris	40	Human Resources
11	Feeney	50	Shipping
12	Fleaur	50	Shipping
13	Fripp	50	Shipping
14	Gates	50	Shipping
15	Gee	50	Shipping
16	Geoni	50	Shipping
17	Grant	50	Shipping
18	Jones	50	Shipping

3. The HR department needs a report of employees in Toronto. Display the last name, job, department number, department name and city for all employees who work in Toronto.

```
SELECT e.last_name, e.job_id, e.department_id,
d.department_name
FROM employees e JOIN departments d
ON (e.department_id = d.department_id) JOIN locations l
```

```
ON (d.location_id = l.location_id) WHERE LOWER(l.city) =
'toronto';
```

Query Result

All Rows Fetched: 2 in 0.077 seconds

	LAST_NAME	JOB_ID	DEPARTMENT_ID	DEPARTMENT_NAME
1	Hartstein	MK_MAN	20	Marketing
2	Fay	MK_REP	20	Marketing

Task 7: Using subqueries to solve queries

1. Create a report that displays the employee number, last name, and salary of all employees who earn more than the average salary. Sort the results in ascending order by salary.

```
select employee_id, last_name, salary
from employees
where department_id in (select department_id from
employees )
and salary > (select avg(salary) from employees)
order by salary;
```

	EMPLOYEE_ID	LAST_NAME	SALARY
1	123	Vollman	6500
2	203	Mavris	6500
3	165	Lee	6800
4	113	Popp	6900
5	161	Sewall	7000
6	155	Tuvault	7000
7	164	Marvins	7200
8	172	Bates	7300
9	171	Smith	7400
10	154	Cambrault	7500
11	160	Doran	7500
12	111	Sciarra	7700
13	112	Urman	7800
14	122	Kaufling	7900
15	120	Weiss	8000
16	159	Smith	8000
17	153	Olsen	8000
18	121	Fripp	8200

2. Write a query that displays the employee number and last name of all employees who work in a department with any employee whose last name contains a ***u***.

```
SELECT employee_id, last_name
FROM employees
WHERE department_id IN (SELECT department_id
FROM employees WHERE last_name like '%u%');
```

	EMPLOYEE_ID	LAST_NAME
1	107	Lorentz
2	106	Pataballa
3	105	Austin
4	104	Ernst
5	103	Hunold
6	199	Grant
7	198	OConnell
8	197	Feeney
9	196	Walsh
10	195	Jones
11	194	McCain
12	193	Everett
13	192	Bell
14	191	Perkins
15	190	Gates
16	189	Dilly
17	188	Chung

3. The HR department needs a report that displays the last name, department number, job id of all employees whose department location id is 1700.

```
SELECT last_name, department_id, job_id
FROM employees
WHERE department_id IN (SELECT department_id
FROM departments
WHERE location_id = 1700);
```

	LAST_...	DEPARTMENT_ID	JOB_ID
1	King	90	AD_PRES
2	Kochhar	90	AD_VP
3	De Haan	90	AD_VP
4	Greenberg	100	FI_MGR
5	Faviet	100	FI_ACCOUNT
6	Chen	100	FI_ACCOUNT
7	Sciarra	100	FI_ACCOUNT
8	Urman	100	FI_ACCOUNT
9	Popp	100	FI_ACCOUNT
10	Raphaely	30	PU_MAN
11	Khoo	30	PU_CLERK
12	Baida	30	PU_CLERK
13	Tobias	30	PU_CLERK
14	Himuro	30	PU_CLERK
15	Colmenares	30	PU_CLERK
16	Whalen	10	AD_ASST
17	Higgins	110	AC_MGR
18	Gietz	110	AC_ACCOUNT

Task 8: Using Set Operators

1. The HR department needs a list of departments for departments that do not contain the job ID ST_CLERK. Use set operators to create this report.

```
SELECT department_id
FROM departments
MINUS
SELECT department_id
FROM employees
WHERE job_id = 'ST_CLERK';
```

	DEPARTMENT_ID
1	10
2	20
3	30
4	40
5	60
6	70
7	80
8	90
9	100
10	110
11	120
12	130
13	140
14	150
15	160
16	170
17	180
18	190

2. Create a report that lists the employee ID and job ID of those employees who currently have a job title that is the same as their job title when they were initially hired by the company (they changed jobs but have now gone back to doing their original job).

```
SELECT employee_id,job_id
FROM employees
INTERSECT
SELECT employee_id,job_id
FROM job_history;
```

	EMPLOYEE_ID	JOB_ID
1	176	SA_REP
2	200	AD_ASST

Task 9: Using DDL statements

1. Create a Tablespace "cms" datafile of size 500 MB.

```
SQL> CREATE TABLESPACE cms DATAFILE 'cms01.dtf' SIZE 500M;

Tablespace created.
```

2. After creation of Tablespace "cms" datafile you realized that its previous allocated size supposed 700 MB. So, alter the size of Tablespace "cms" datafile from 500 MB to 700MB.

```
SQL> ALTER DATABASE DATAFILE 'cms01.dtf' RESIZE 700M;

Database altered.
```

3. Write a SQL query to create user name JOHN and JENNY with default Tablespace "cms" with unlimited quota. Also grant the create session and create table to both users.

```
SQL> CREATE USER JOHN IDENTIFIED BY muskan_admin
  2  DEFAULT TABLESPACE cms
  3  QUOTA UNLIMITED ON cms;

User created.

SQL> CREATE USER JENNY IDENTIFIED BY muskan_admin
  2  DEFAULT TABLESPACE cms
  3  QUOTA UNLIMITED ON cms;

User created.

SQL> GRANT CREATE SESSION, CREATE TABLE TO JOHN;

Grant succeeded.

SQL> GRANT CREATE SESSION, CREATE TABLE TO JENNY;

Grant succeeded.

SQL>
```

4. Create the following tables from the user JOHN.

department		
Field	**Data type**	**Constraint type**
dept_id	number (4)	PRIMARY KEY
dept_name	varchar (25)	NOT NULL
dept_block_no	number (4)	

student		
Field	**Data type**	**Constraint type**
stu_id	number (4)	PRIMARY KEY
stu_name	varchar (25)	NOT NULL
stu_address	varchar (25)	
dept_id	number (4)	FOREIGN KEY
age	number (3)	

staff		
Field	**Data type**	**Constraint type**
staff_id	number (4)	PRIMARY KEY

staff_name	varchar (25)	
dept_id	number (4)	FOREIGN KEY

subject		
Field	**Data type**	**Constraint type**
sub_id	number (4)	PRIMARY KEY
sub_name	varchar (25)	
sub_code	varchar (25)	
staff_id	number (4)	FOREIGN KEY

marks		
Field	**Data type**	**Constraint type**
marks_obtain	number (4)	
sub_id	number (4)	PRIMARY KEY, FOREIGN KEY
stu_id	number (4)	PRIMARY KEY, FOREIGN KEY

Connecting as JOHN user

```
SQL> connect JOHN;
Enter password:
Connected.
SQL>
```

```
CREATE TABLE department (
  dept_id NUMBER(4) PRIMARY KEY,
  dept_name VARCHAR2(25) NOT NULL,
  dept_block_no NUMBER(4)
);

-- Create the "student" table
```

```
CREATE TABLE student (
  stu_id NUMBER(4) PRIMARY KEY,
  stu_name VARCHAR2(25) NOT NULL,
  stu_address VARCHAR2(25),
  dept_id NUMBER(4) REFERENCES department(dept_id),
  age NUMBER(3)
);

-- Create the "staff" table
CREATE TABLE staff (
  staff_id NUMBER(4) PRIMARY KEY,
  staff_name VARCHAR2(25),
  dept_id NUMBER(4) REFERENCES department(dept_id)
);

-- Create the "subject" table
CREATE TABLE subject (
  sub_id NUMBER(4) PRIMARY KEY,
  sub_name VARCHAR2(25),
  sub_code VARCHAR2(25),
  staff_id NUMBER(4) REFERENCES staff(staff_id)
);

-- Create the "marks" table
CREATE TABLE marks (
  marks_obtain NUMBER(4),
```

```
  sub_id NUMBER(4),
  stu_id NUMBER(4),
  PRIMARY KEY (sub_id, stu_id),
  FOREIGN KEY (sub_id) REFERENCES subject(sub_id),
  FOREIGN KEY (stu_id) REFERENCES student(stu_id)
);
```

```
Table DEPARTMENT created.

Table STUDENT created.

Table STAFF created.

Table SUBJECT created.

Table MARKS created.
```

5. Write ALTER TABLE statement to modify the range of columns stu_name from varchar (25) to varchar (40) from an existing table student.

```
ALTER TABLE student
MODIFY stu_name VARCHAR2(40);
```

```
Table STUDENT altered.

Name        Null?    Type
----------- -------- ------------
STU_ID      NOT NULL NUMBER(4)
STU_NAME    NOT NULL VARCHAR2(40)
STU_ADDRESS          VARCHAR2(25)
DEPT_ID              NUMBER(4)
AGE                  NUMBER(3)
```

6. Write ALTER TABLE statement to drop the age attribute from the existing table student.

```
ALTER TABLE student
DROP COLUMN age;

DESCRIBE student;
```

```
Table STUDENT altered.

Name        Null?    Type
----------- -------- ------------
STU_ID      NOT NULL NUMBER(4)
STU_NAME    NOT NULL VARCHAR2(40)
STU_ADDRESS          VARCHAR2(25)
DEPT_ID              NUMBER(4)
```

Task 10: Using DCL Statement

1. Create the role named SUPPORT with SELECT and UPDATE permission on student, marks, subject, staff and department table created by JOHN.

```
C:\Users\DELL>sqlplus / as sysdba

SQL*Plus: Release 11.2.0.2.0 Production on Tue Sep 19 17:28:05 2023

Copyright (c) 1982, 2014, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production

SQL> -- Create the "SUPPORT" role
SQL> CREATE ROLE SUPPORT;

Role created.

SQL>
SQL> -- Grant SELECT and UPDATE permissions on the tables to the "SUPPORT" role
SQL> GRANT SELECT, UPDATE ON JOHN.student TO SUPPORT;

Grant succeeded.

SQL> GRANT SELECT, UPDATE ON JOHN.marks TO SUPPORT;

Grant succeeded.

SQL> GRANT SELECT, UPDATE ON JOHN.subject TO SUPPORT;

Grant succeeded.

SQL> GRANT SELECT, UPDATE ON JOHN.staff TO SUPPORT;

Grant succeeded.

SQL> GRANT SELECT, UPDATE ON JOHN.department TO SUPPORT;
```

2. Create the role named MANAGER with INSERT and DELETE permission on student, marks, subject, staff and department table created by JOHN. Also grant SUPPORT role to the MANAGER role.

```
SQL> -- Connect to your Oracle database using SQL*Plus or any other SQL client as a privileged user.
SQL>
SQL> -- Create the "MANAGER" role
SQL> CREATE ROLE MANAGER;

Role created.

SQL>
SQL> -- Grant INSERT and DELETE permissions on the tables to the "MANAGER" role
SQL> GRANT INSERT, DELETE ON JOHN.student TO MANAGER;

Grant succeeded.

SQL> GRANT INSERT, DELETE ON JOHN.marks TO MANAGER;

Grant succeeded.

SQL> GRANT INSERT, DELETE ON JOHN.subject TO MANAGER;

Grant succeeded.

SQL> GRANT INSERT, DELETE ON JOHN.staff TO MANAGER;

Grant succeeded.

SQL> GRANT INSERT, DELETE ON JOHN.department TO MANAGER;

Grant succeeded.

SQL>
SQL> -- Grant the "SUPPORT" role to the "MANAGER" role
SQL> GRANT SUPPORT TO MANAGER;

Grant succeeded.

SQL>
```

3. Assign MANAGER role to the user JOHN and SUPPORT role to the user JENNY.

```
SQL> GRANT SUPPORT TO MANAGER;

Grant succeeded.

SQL> GRANT MANAGER TO JOHN;

Grant succeeded.

SQL>
SQL> -- Assign the "SUPPORT" role to the user "JENNY"
SQL> GRANT SUPPORT TO JENNY;

Grant succeeded.

SQL>
```

Task 11: Using DML Statement

1. Enter the following data in the respective tables.

department		
dept_id	dept_name	dept_block_no
1	Computer	100
2	Mathematics	200
3	Economics	300
4	Account	400
5	Physics	500

staff		
staff_id	staff_name	dept_id
11	Mohan	1
22	Pratima	2
33	Madan	1
44	Kamala	3
55	Sandhya	4
66	Umesh	2
77	Ramesh	1

student			
stu_id	stu_name	stu_address	dept_id
10	Maya	Palpa	1
11	Abin	Ktm	2
12	Aarav	Ktm	1
13	Ashna	Palpa	3
14	Anuj	Pokhara	4
15	Manish	Banepa	2
16	Pinky	Ktm	1

subject			
sub_id	sub_name	sub_code	staff_id
20	DBMS	D-20	11
21	C++	C-21	22
22	NM	N-22	11
23	TOC	T-23	77
24	PHP	P-24	44
25	AI	A-25	33
26	ASP	A-26	55
27	CG	C-27	66
28	C-Prog	C-28	44

marks		
marks_obtain	sub_id	stu_id
60	20	10
55	21	12
58	22	10
49	20	13
61	23	14
67	22	13
60	24	16
55	26	15
33	25	11

```
-- Insert data into the "department" table
INSERT INTO department (dept_id, dept_name, dept_block_no)
VALUES (1, 'Computer', 100);

INSERT INTO department (dept_id, dept_name, dept_block_no)
VALUES (2, 'Mathematics', 200);

INSERT INTO department (dept_id, dept_name, dept_block_no)
VALUES (3, 'Economics', 300);

INSERT INTO department (dept_id, dept_name, dept_block_no)
VALUES (4, 'Account', 400);
```

```
INSERT INTO department (dept_id, dept_name, dept_block_no)
VALUES (5, 'Physics', 500);

-- Insert data into the "staff" table
INSERT INTO staff (staff_id, staff_name, dept_id)
VALUES (11, 'Mohan', 1);

INSERT INTO staff (staff_id, staff_name, dept_id)
VALUES (22, 'Pratima', 2);

INSERT INTO staff (staff_id, staff_name, dept_id)
VALUES (33, 'Madan', 1);

INSERT INTO staff (staff_id, staff_name, dept_id)
VALUES (44, 'Kamala', 3);

INSERT INTO staff (staff_id, staff_name, dept_id)
VALUES (55, 'Sandhya', 4);

INSERT INTO staff (staff_id, staff_name, dept_id)
VALUES (66, 'Umesh', 2);

INSERT INTO staff (staff_id, staff_name, dept_id)
VALUES (77, 'Ramesh', 1);

-- Insert data into the "student" table
INSERT INTO student (stu_id, stu_name, stu_address, dept_id)
VALUES (10, 'Maya', 'Palpa', 1);

INSERT INTO student (stu_id, stu_name, stu_address, dept_id)
VALUES (11, 'Abin', 'Ktm', 2);

INSERT INTO student (stu_id, stu_name, stu_address, dept_id)
```

```
VALUES (12, 'Aarav', 'Ktm', 1);

INSERT INTO student (stu_id, stu_name, stu_address, dept_id)
VALUES (13, 'Ashna', 'Palpa', 3);

INSERT INTO student (stu_id, stu_name, stu_address, dept_id)
VALUES (14, 'Anuj', 'Pokhara', 4);

INSERT INTO student (stu_id, stu_name, stu_address, dept_id)
VALUES (15, 'Manish', 'Banepa', 2);

INSERT INTO student (stu_id, stu_name, stu_address, dept_id)
VALUES (16, 'Pinky', 'Ktm', 1);

-- Insert data into the "subject" table
INSERT INTO subject (sub_id, sub_name, sub_code, staff_id)
VALUES (20, 'DBMS', 'D-20', 11);

INSERT INTO subject (sub_id, sub_name, sub_code, staff_id)
VALUES (21, 'C++', 'C-21', 22);

INSERT INTO subject (sub_id, sub_name, sub_code, staff_id)
VALUES (22, 'NM', 'N-22', 11);

INSERT INTO subject (sub_id, sub_name, sub_code, staff_id)
VALUES (23, 'TOC', 'T-23', 77);

INSERT INTO subject (sub_id, sub_name, sub_code, staff_id)
VALUES (24, 'PHP', 'P-24', 44);

INSERT INTO subject (sub_id, sub_name, sub_code, staff_id)
VALUES (25, 'AI', 'A-25', 33);

INSERT INTO subject (sub_id, sub_name, sub_code, staff_id)
```

```
VALUES (26, 'ASP', 'A-26', 55);

INSERT INTO subject (sub_id, sub_name, sub_code, staff_id)
VALUES (27, 'CG', 'C-27', 66);

INSERT INTO subject (sub_id, sub_name, sub_code, staff_id)
VALUES (28, 'C-Prog', 'C-28', 44);

-- Insert data into the "marks" table
INSERT INTO marks (marks_obtain, sub_id, stu_id)
VALUES (60, 20, 10);

INSERT INTO marks (marks_obtain, sub_id, stu_id)
VALUES (55, 21, 12);

INSERT INTO marks (marks_obtain, sub_id, stu_id)
VALUES (58, 22, 10);

INSERT INTO marks (marks_obtain, sub_id, stu_id)
VALUES (49, 20, 13);

INSERT INTO marks (marks_obtain, sub_id, stu_id)
VALUES (61, 23, 14);

INSERT INTO marks (marks_obtain, sub_id, stu_id)
VALUES (67, 22, 13);

INSERT INTO marks (marks_obtain, sub_id, stu_id)
VALUES (60, 24, 16);

INSERT INTO marks (marks_obtain, sub_id, stu_id)
VALUES (55, 26, 15);

INSERT INTO marks (marks_obtain, sub_id, stu_id)
```

```
VALUES (33, 25, 11);
```

	DEPT_ID	DEPT_NAME	DEPT_BLOCK_NO
1	1	Computer	100
2	2	Mathematics	200
3	3	Economics	300
4	4	Account	400
5	5	Physics	500

	STAFF_ID	STAFF_NAME	DEPT_ID
1	11	Mohan	1
2	22	Pratima	2
3	33	Madan	1
4	44	Kamala	3
5	55	Sandhya	4
6	66	Umesh	2
7	77	Ramesh	1

	STU_ID	STU_NAME	STU_ADDRESS	DEPT_ID
1	10	Maya	Palpa	1
2	11	Abin	Ktm	2
3	12	Aarav	Ktm	1
4	13	Ashna	Palpa	3
5	14	Anuj	Pokhara	4
6	15	Manish	Banepa	2
7	16	Pinky	Ktm	1

	SUB_ID	SUB_NAME	SUB_CODE	STAFF_ID
1	20	DBMS	D-20	11
2	21	C++	C-21	22
3	22	NM	N-22	11
4	23	TOC	T-23	77
5	24	PHP	P-24	44
6	25	AI	A-25	33
7	26	ASP	A-26	55
8	27	CG	C-27	66
9	28	C-Prog	C-28	44

2. Change the dept_name of dept_id 3 to Statistics in department table.

```
UPDATE department

SET dept_name = 'Statistics' WHERE dept_id = 3;
```

	DEPT_ID	DEPT_NAME	DEPT_BLOCK_NO
1	1	Computer	100
2	2	Mathematics	200
3	3	Statistics	300
4	4	Account	400
5	5	Physics	500

Task 12: Creating other Schema Objects.

1. Create a view of name subject_view with attributes subject name and subject code.

```
CREATE VIEW subject_view AS
SELECT sub_name AS subject_name, sub_code AS subject_code
FROM john.subject;
```

2. Confirm that the view works. Display the contents of the subject_view.

```
SQL> SELECT * FROM subject_view;

SUBJECT_NAME                   SUBJECT_CODE
------------------------------ ------------------------------
DBMS                           D-20
C++                            C-21
NM                             N-22
TOC                            T-23
PHP                            P-24
AI                             A-25
ASP                            A-26
CG                             C-27
C-Prog                         C-28

9 rows selected.
```

3. Alter the Subject_view by adding subject id along with attributes subject name and subject code.

```
SQL> CREATE OR REPLACE VIEW subject_view AS
  2  SELECT sub_id AS subject_id, sub_name AS subject_name, sub_code AS subject_code
  3  FROM john.subject;

View created.

SQL> SELECT * FROM subject_view;

SUBJECT_ID SUBJECT_NAME             SUBJECT_CODE
---------- ------------------------ ------------------------
        20 DBMS                     D-20
        21 C++                      C-21
        22 NM                       N-22
        23 TOC                      T-23
        24 PHP                      P-24
        25 AI                       A-25
        26 ASP                      A-26
        27 CG                       C-27
        28 C-Prog                   C-28

9 rows selected.

SQL>
```

4. Create synonyms emp for staff table.

```
CREATE SYNONYM emp FOR JOHN.staff;
```

5. Confirm that the synonyms work. Display the contents of the emp table using recently created synonyms.

```
SQL> SELECT * FROM emp;

  STAFF_ID STAFF_NAME                  DEPT_ID
---------- ------------------------ ----------
        11 Mohan                             1
        22 Pratima                           2
        33 Madan                             1
        44 Kamala                            3
        55 Sandhya                           4
        66 Umesh                             2
        77 Ramesh                            1

7 rows selected.
```

Task 13: ORDBMS (based on Oracle) Tasks.

1. Write a SQL query to create user name <your name>. Also grant all privileges.

```
Command Prompt - sqlplus  / as sysdba
Microsoft Windows [Version 10.0.19045.3448]
(c) Microsoft Corporation. All rights reserved.

C:\Users\DELL>sqlplus / as sysdba

SQL*Plus: Release 11.2.0.2.0 Production on Tue Sep 26 08:52:43 2023

Copyright (c) 1982, 2014, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production

SQL> create user Muskan identified by Muskan123;

User created.

SQL> grant all privileges to Muskan;

Grant succeeded.

SQL> commit;

Commit complete.

SQL>
```

2. Login Oracle database with your newly created user in task 13.1.

```
SQL> connect Muskan;
Enter password:
Connected.
SQL>
```

3. Define a new object type named "EmployeeType" with attributes: emp_id, emp_name, address, salary.

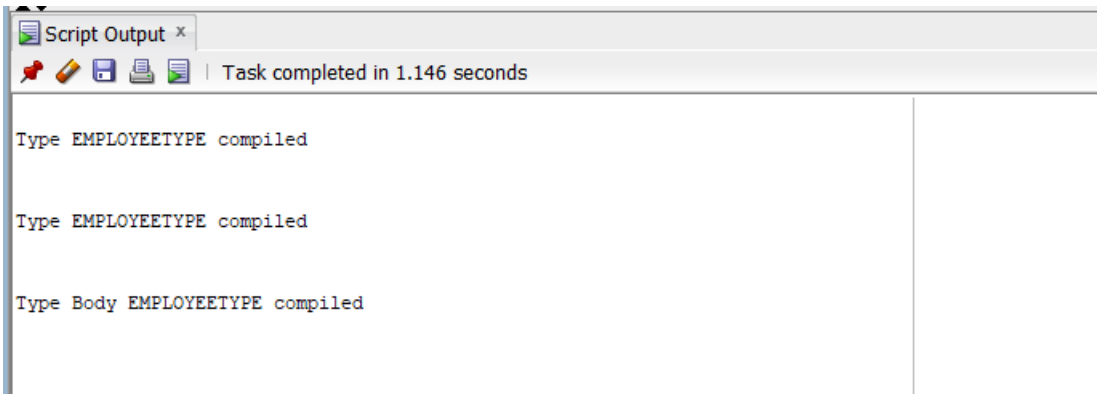
```
CREATE OR REPLACE TYPE EmployeeType AS OBJECT (emp_id NUMBER, emp_name VARCHAR(20), address VARCHAR(20), salary NUMBER) NOT FINAL;
```

```
Type EMPLOYEETYPE compiled
```

4. Create a constructor method to initialize the object with these attributes.

```
CREATE OR REPLACE TYPE EmployeeType AS OBJECT (emp_id
NUMBER, emp_name VARCHAR(20), address VARCHAR(20), salary
NUMBER,
CONSTRUCTOR FUNCTION EmployeeType(emp_id NUMBER, emp_name
VARCHAR, address VARCHAR, salary NUMBER) RETURN SELF AS
RESULT) NOT FINAL;
/
CREATE OR REPLACE TYPE BODY EmployeeType AS
CONSTRUCTOR FUNCTION EmployeeType(emp_id Number, emp_name
VARCHAR, address VARCHAR, salary NUMBER) RETURN SELF AS
RESULT IS
BEGIN
   SELF.emp_id := emp_id;
   SELF.emp_name := emp_name;
   SELF.address := address;
   SELF.salary := salary;

 END;
 END;
/
```



5. Create a method named "getDetails" that displays all the attributes of an employee object.

```
CREATE OR REPLACE TYPE EmployeeType AS OBJECT (emp_id
NUMBER, emp_name VARCHAR(20), address VARCHAR(20), salary
NUMBER, MEMBER FUNCTION getDetails RETURN VARCHAR) NOT
FINAL;
/
CREATE OR REPLACE TYPE BODY EmployeeType AS
MEMBER FUNCTION getDetails RETURN VARCHAR IS
BEGIN
   RETURN 'Employee ID: ' || TO_CHAR(self.emp_id) ||
   ', Employee Name : ' || self.emp_name ||
   ', Address: ' || self.address ||
```

```
    ', Salary : ' || TO_CHAR(self.salary);
END;
END;
/
```

6. Create a table named "EmployeeTable" that stores EmployeeType objects.

```
CREATE TABLE EmployeeTable (emp_data EmployeeType);
```

```
Table EMPLOYEETABLE created.
```

7. Insert at least three records into “EmployeeTable” created in task 13.6

```
-- Insert the first record
INSERT INTO EmployeeTable (emp_data) VALUES
(EmployeeType(101, 'Muskan', 'Radhe Radhe', 60000));

-- Insert the second record
INSERT INTO EmployeeTable (emp_data) VALUES
(EmployeeType(102, 'Ramesh', 'Shantinagar', 70000));

-- Insert the third record
INSERT INTO EmployeeTable (emp_data) VALUES
(EmployeeType(103, 'Ronish', 'Chabahil', 80000));

-- Insert the fourth record
INSERT INTO EmployeeTable (emp_data) VALUES
(EmployeeType(104, 'Nikish', 'Baneshwor', 90000));

-- Insert the fifth record
INSERT INTO EmployeeTable (emp_data) VALUES
(EmployeeType(105, 'Saphal', 'Bhaktapur', 100000));
```

	EMP_DATA
1	[MUSKAN.EMPLOYEETYPE]
2	[MUSKAN.EMPLOYEETYPE]
3	[MUSKAN.EMPLOYEETYPE]
4	[MUSKAN.EMPLOYEETYPE]
5	[MUSKAN.EMPLOYEETYPE]

8. Create a new object type "ManagerType" that inherits from "EmployeeType" and has an additional attribute dept_name.

```
CREATE OR REPLACE TYPE ManagerType UNDER EmployeeType(
   dept_name varchar(20));
```

```
Type MANAGERTYPE compiled
```

9. Implement a method to display the details of a manager, including inherited attributes.

```
CREATE OR REPLACE TYPE ManagerType1 UNDER EmployeeType(
dept_name  varchar(20),  MEMBER  FUNCTION  getALLDetails
Return VARCHAR
);
/
CREATE OR REPLACE TYPE BODY ManagerType1 AS
MEMBER FUNCTION getALLDetails RETURN VARCHAR IS
BEGIN
   RETURN 'Employee ID: ' || TO_CHAR(self.emp_id) ||
   ', Employee Name : ' || self.emp_name ||
   ', Address : ' || self.address ||
   ', Salary : ' || self.salary ||
   ', Department Name: ' || self.dept_name;
END;
END;
/
```

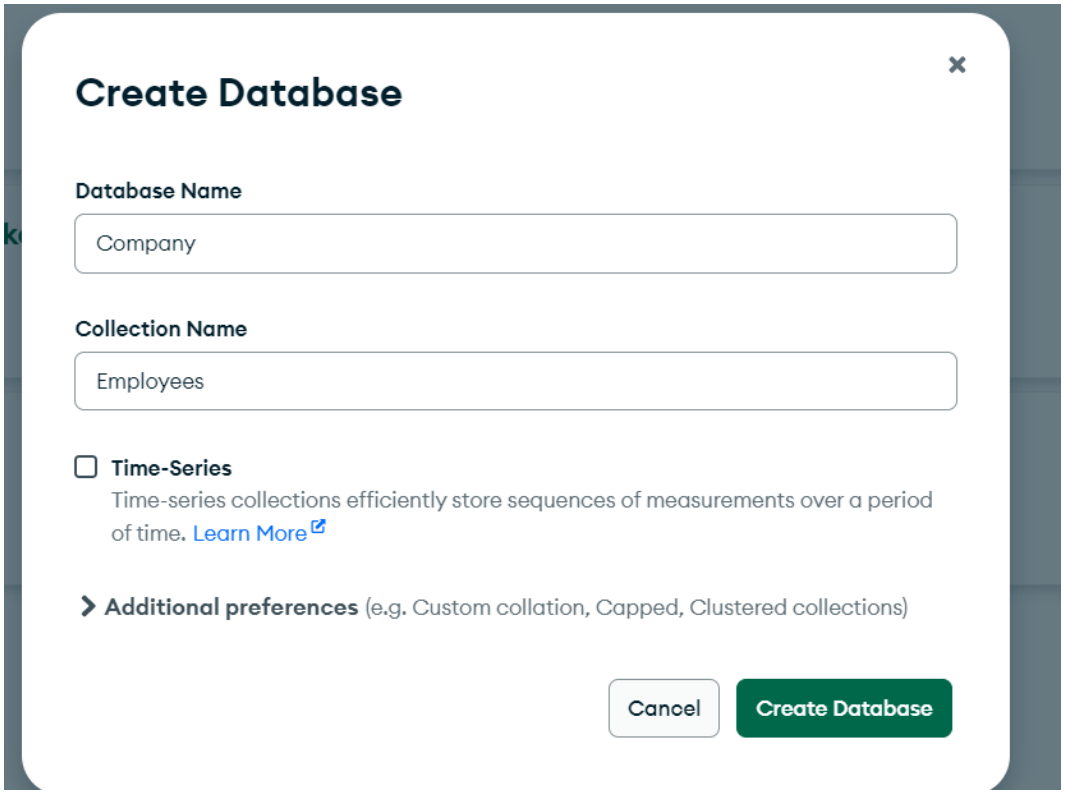
10. Create a new table named "ManagerTable" that stores ManagerType objects.

```
CREATE TABLE ManagerTable (
  manager_data ManagerType
);
```

```
Table MANAGERTABLE created.
```

Task 14: NoSQL (based on MongoDB) Tasks.

1. Create a new database named "Company" and a collection named "Employees".



2. Insert at least five employee documents into the "Employees" collection, including fields like name, age, department, mobile and salary. Some records may not have all the attributes. Also, some employees may have multiple values for the mobile.

```
from dotenv import load_dotenv, find_dotenv
import os
from pymongo import MongoClient

load_dotenv(find_dotenv())
password = os.environ.get("MONGODB_PWD")

connection_string =
f"mongodb+srv://bhandari008:{password}@sms.rl47bz7.mongodb.n
et/?retryWrites=true&w=majority"
client = MongoClient(connection_string)

db = client["Company"]
collection = db["Employees"]
```

```
# Insert employee documents
employees = [
    {
        "name": "Muskan Bhandari",
        "age": 21,
        "department": "HR",
        "mobile": ["123-456-7890", "987-654-3210"],
        "salary": 55000
    },
    {
        "name": "Ramesh Basnet",
        "age": 28,
        "department": "IT",
        "mobile": "555-123-4567",
        "salary": 60000
    },
    {
        "name": "Saphal Subedi",
        "department": "Finance",
        "mobile": ["111-222-3333", "444-555-6666"],
        "salary": 62000
    },
    {
        "name": "Nikish Subedi",
        "age": 35,
        "department": "Sales",
        "salary": 50000
    },
    {
        "name": "Sajag Dhungana",
        "department": "Marketing",
        "mobile": "777-888-9999",
        "salary": 58000
    }
```

```
]

# Insert the employee documents into the collection
collection.insert_many(employees)

# Close the MongoDB client
client.close()
```

```
_id: ObjectId('6512568d8100ce604df12a89')
name: "Muskan Bhandari"
age: 21
department: "HR"
▸ mobile: Array (2)
salary: 55000
```

```
_id: ObjectId('6512568d8100ce604df12a8a')
name: "Ramesh Basnet"
age: 28
department: "IT"
mobile: "555-123-4567"
salary: 60000
```

```
_id: ObjectId('6512568d8100ce604df12a8b')
name: "Saphal Subedi"
department: "Finance"
▸ mobile: Array (2)
salary: 62000
```

3. Write queries to retrieve all employees, employees in a specific department, and employees with a salary greater than a certain value.

```
all_employees = collection.find({})
for employee in all_employees:
    print(employee)

hr_employees = collection.find({"department": "HR"})
for employee in hr_employees:
    print(employee)

high_salary_employees = collection.find({"salary":
{"$gt": 60000}})
for employee in high_salary_employees:
```

```
print(employee)
```

```
{'_id': ObjectId('6512568d8100ce604df12a89'), 'name': 'Muskan Bhandari', 'age': 21, 'department': 'HR', 'mobile': ['123-456-7890', '987-654-3210'], 'salary': 55
000}
{'_id': ObjectId('6512568d8100ce604df12a8a'), 'name': 'Ramesh Basnet', 'age': 28, 'department': 'IT', 'mobile': '555-123-4567', 'salary': 60000}
{'_id': ObjectId('6512568d8100ce604df12a8b'), 'name': 'Saphal Subedi', 'department': 'Finance', 'mobile': ['111-222-3333', '444-555-6666'], 'salary': 62000}
{'_id': ObjectId('6512568d8100ce604df12a8c'), 'name': 'Nikish Subedi', 'age': 35, 'department': 'Sales', 'salary': 50000}
{'_id': ObjectId('6512568d8100ce604df12a8d'), 'name': 'Sajag Dhungana', 'department': 'Marketing', 'mobile': '777-888-9999', 'salary': 58000}
{'_id': ObjectId('6512568d8100ce604df12a89'), 'name': 'Muskan Bhandari', 'age': 21, 'department': 'HR', 'mobile': ['123-456-7890', '987-654-3210'], 'salary': 55
000}
{'_id': ObjectId('6512568d8100ce604df12a8b'), 'name': 'Saphal Subedi', 'department': 'Finance', 'mobile': ['111-222-3333', '444-555-6666'], 'salary': 62000}
(env) PS F:\8thSem\mini_project\sms\src>
```

4. Update the department and salary of an employee with a given name.

```
employee_name_to_update = "Muskan Bhandari"

# Define the new department and salary values
new_department = "Finance"
new_salary = 65000

result = collection.update_one({"name": employee_name_to_update}, {"$set": {"department": new_department, "salary": new_salary}})
```

Before:

```
_id: ObjectId('6512568d8100ce604df12a89')
name: "Muskan Bhandari"
age: 21
department: "HR"
▸ mobile: Array (2)
salary: 55000
```

After:

```
_id: ObjectId('6512568d8100ce604df12a89')
name: "Muskan Bhandari"
age: 21
department: "Finance"
▸ mobile: Array (2)
salary: 65000
```

5. Increase the salary of all employees in a particular department by a specific percentage.

```
target_department = "HR"
percentage_increase = 10  # 10% increase

# Calculate the salary increase amount
salary_increase_amount = (percentage_increase / 100) *
1.0  # Convert to decimal

# Update all employees in the target department
result = collection.update_many({"department":
target_department}, {"$mul": {"salary": 1 +
salary_increase_amount}})
```

Before:

```
_id: ObjectId('6512568d8100ce604df12a89')
name: "Muskan Bhandari"
age: 21
department: "Finance"
▸ mobile: Array (2)
salary: 65000
```

After:

```
▸ _id: ObjectId('6512568d8100ce604df12a89')
name: "Muskan Bhandari"
age: 21
department: "Finance"
▸ mobile: Array (2)
salary: 71500
```