RAJALAKSHMI
ENGINEERING COLLEGE

RAJALAKSHMI NAGAR, THANDALAM – 602 105

CS23332 - DATABASE MANAGEMENT SYSTEM

Laboratory Record Notebook

CS23332 DATABASE MANAGEMENT SYSTEMS

NAME	K.H.D. Mohamed Ikram
ROLL NO	230701188
DEPT	CSE
SEC	C

230701188

Ex.No.: 1	**CREATION OF BASE TABLE AND**

Date:	31.07.2024	DML OPERATIONS

- Create MY_EMPLOYEE table with the following structure

NAME	NULL?	TYPE
ID	Not null	Number(4)
Last_name		Varchar(25)
First_name		Varchar(25)
Userid		Varchar(25)
Salary		Number(9,2)

CREATE TABLE MY_EMPLOYEE (ID NUMBER(4) NOT NULL, Last_name VARCHAR2(25), First_name VARCHAR2(25), Userid VARCHAR2(25),Salary NUMBER(9, 2));

- Add the first and second rows data to MY_EMPLOYEE table from the following sample data.

ID	Last_name	First_name	Userid	salary
1	Patel	Ralph	rpatel	895
2	Dancs	Betty	bdancs	860
3	Biri	Ben	bbiri	1100
4	Newman	Chad	Cnewman	750
5	Ropebur	Audrey	aropebur	1550

Begin

```
INSERT INTO MY_EMPLOYEE VALUES (1, 'Patel', 'Ralph', 'rpatel',
895); INSERT INTO MY_EMPLOYEE VALUES (2, 'Dancs', 'Betty',
'bdancs', 860);
End;
```

- Display the table with values.

```
Select * from My_Employee;
```

- Populate the next two rows of data from the sample data. Concatenate the first letter of the first_name with the first seven characters of the last_name to produce Userid.

```
Begin
INSERT INTO MY_EMPLOYEE (ID, Last_name, First_name, Userid,
Salary) VALUES (3, 'Biri', 'Ben', SUBSTR('Biri', 1, 1) ||
SUBSTR('Biri', 1, 7), 1100);
INSERT INTO MY_EMPLOYEE (ID, Last_name, First_name, Userid, Salary)
VALUES (4, 'Newman', 'Chad', SUBSTR('Newman', 1, 1) || SUBSTR('Newman', 1, 7),
750);
End;
```

- Delete Betty dancs from MY _EMPLOYEE table.

```
DELETE FROM MY_EMPLOYEE WHERE Last_name = 'Dancs';
```

- Empty the fourth row of the emp table.

```
DELETE FROM MY_EMPLOYEE WHERE ID = 4;
```

- Make the data additions permanent.

```
COMMIT;
```

- Change the last name of employee 3 to Drexler.

UPDATE MY_EMPLOYEE SET Last_name = 'Drexler' WHERE ID = 3;

- Change the salary to 1000 for all the employees with a salary less than 900.

UPDATE MY_EMPLOYEE SET Salary = 1000 WHERE Salary < 900;

230701188

Ex.No.: 2		DATA MANIPULATIONS
Date:	05.08.2024	

Create the following tables with the given

structure. EMPLOYEES TABLE

NAME	NULL?	TYPE
Employee_id	Not null	Number(6)
First_Name		Varchar(20)
Last_Name	Not null	Varchar(25)
Email	Not null	Varchar(25)
Phone_Number		Varchar(20)
Hire_date	Not null	Date
Job_id	Not null	Varchar(10)
Salary		Number(8,2)
Commission_pct		Number(2,2)
Manager_id		Number(6)
Department_id		Number(4)

- Find out the employee id, names, salaries of all the employees

```
SELECT Employee_id, First_name, Last_name, Salary FROM EMPLOYEES;
```

- List out the employees who works under manager 100

```
SELECT Employee_id, First_name, Last_name FROM EMPLOYEES
WHERE Manager_id = 100;
```

- Find the names of the employees who have a salary greater than or equal

to 4800 SELECT First_name, Last_name FROM EMPLOYEES WHERE

Salary >= 4800;

- List out the employees whose last name is _AUSTIN‘

```
SELECT Employee_id, First_name, Last_name FROM EMPLOYEES WHERE
Last_name = 'AUSTIN';
```

- Find the names of the employees who works in departments 60,70 and 80

```
SELECT First_name, Last_name FROM EMPLOYEES WHERE Department_id IN
(60, 70, 80);
```

- Display the unique Manager_Id.

```
SELECT DISTINCT Manager_id FROM EMPLOYEES;
```

Create an Emp table with the following fields: (EmpNo, EmpName, Job,Basic, DA, HRA,PF, GrossPay, NetPay) (Calculate DA as 30% of Basic and HRA as 40% of Basic)

- Insert Five Records and calculate GrossPay and NetPay.

```
INSERT INTO EMP (EmpNo, EmpName, Job, Basic, DA, HRA, PF, GrossPay, NetPay)
VALUES (1, 'John Doe', 'Manager', 50000, 0.30 * 50000, -- DA as 30% of Basic
 0.40 * 50000, -- HRA as 40% of Basic,0.12 * 50000, -- PF as 12% of Basic
50000 + (0.30 * 50000) + (0.40 * 50000), -- GrossPay (50000 + (0.30 * 50000) +
(0.40 *
50000)) - (0.12 * 50000) -- NetPay
    );

INSERT INTO EMP (EmpNo, EmpName, Job, Basic, DA, HRA, PF, GrossPay, NetPay)
VALUES (2, 'Jane Smith', 'Clerk', 30000,  0.30 * 30000, 0.40 * 30000,
    0.12 * 30000,
    30000 + (0.30 * 30000) + (0.40 * 30000),
    (30000 + (0.30 * 30000) + (0.40 * 30000)) - (0.12 * 30000)
    );

INSERT INTO EMP (EmpNo, EmpName, Job, Basic, DA, HRA, PF, GrossPay, NetPay)
VALUES (3, 'Mike Johnson', 'Salesman', 40000,
    0.30 * 40000,
    0.40 * 40000,
    0.12 * 40000,
    40000 + (0.30 * 40000) + (0.40 * 40000),
    (40000 + (0.30 * 40000) + (0.40 * 40000)) - (0.12 * 40000)
    );

INSERT INTO EMP (EmpNo, EmpName, Job, Basic, DA, HRA, PF, GrossPay, NetPay)
VALUES (4, 'Emily Davis', 'Accountant', 35000,
    0.30 * 35000,

    0.40 * 35000,
    0.12 * 35000,
    35000 + (0.30 * 35000) + (0.40 * 35000),
    (35000 + (0.30 * 35000) + (0.40 * 35000)) - (0.12 * 35000)
    );
```

```
INSERT INTO EMP (EmpNo, EmpName, Job, Basic, DA, HRA, PF, GrossPay, NetPay)
VALUES (5, 'Robert Miller', 'Clerk', 25000,
      0.30 * 25000,
      0.40 * 25000,
      0.12 * 25000,
      25000 + (0.30 * 25000) + (0.40 * 25000),
      (25000 + (0.30 * 25000) + (0.40 * 25000)) - (0.12 * 25000)
     );
```

- Display the employees whose Basic is lowest in each department.

```
SELECT EmpNo, EmpName, Job, Basic FROM EMP E1 WHERE
   Basic = ( SELECT MIN(Basic) FROM EMP E2 WHERE E2.Job =
   E1.Job);
```

- If Net Pay is less than 50000, display employee number,name and net pay

```
SELECT EmpNo, EmpName, NetPay FROM EMP WHERE NetPay < 50000;
```

DEPARTMENT TABLE

NAME	**NULL?**	**TYPE**
Dept_id	Not null	Number(6)
Dept_name	Not null	Varchar(20)
Manager_id		Number(6)

Location_id		Number(4)

JOB_GRADE TABLE

NAME	NULL?	TYPE
Grade_level		Varchar(2)
Lowest_sal		Number
Highest_sal		Number

LOCATION TABLE

NAME	NULL?	TYPE
Location_id	Not null	Number(4)
St_addr		Varchar(40)
Postal_code		Varchar(12)
City	Not null	Varchar(30)
State_province		Varchar(25)
Country_id		Char(2)

- Create the DEPT table based on the DEPARTMENT following the table instance chart below. Confirm that the table is created.

Column name	ID	NAME
Key Type		
Nulls/Unique		
FK table		
FK column		
Data Type	Number	Varchar2
Length	7	25

CREATE TABLE DEPT (Dept_id NUMBER(6) NOT NULL, Dept_name VARCHAR2(20) NOT
NULL,Manager_id NUMBER(6), Location_id NUMBER(4), CONSTRAINT my_dept_id_pk PRIMARY KEY (Dept_id));

• Create the EMP table based on the following instance chart. Confirm that the table is created.

Column name	ID	LAST_NAME	FIRST_NAME	DEPT_ID
Key Type				
Nulls/Unique				
FK table				
FK column				
Data Type	Number	Varchar2	Varchar2	Number
Length	7	25	25	7

CREATE TABLE EMP (EmpNo NUMBER(7) PRIMARY KEY,Last_name VARCHAR2(25)
NOT NULL,First_name VARCHAR2(25),Dept_id NUMBER(7),
CONSTRAINT my_emp_dept_id_fk FOREIGN KEY (Dept_id)
REFERENCES DEPT(Dept_id));

• Modify the EMP table to allow for longer employee last names. Confirm the modification.(Hint: Increase the size to 50)

ALTER TABLE EMP MODIFY (Last_name VARCHAR2(50));

• Create the EMPLOYEES2 table based on the structure of EMPLOYEES table. Include Only the Employee_id, First_name, Last_name, Salary and Dept_id coloumns. Name the columns Id, First_name, Last_name, salary and Dept_id respectively.

CREATE TABLE EMPLOYEES2 (Id NUMBER(6) PRIMARY KEY,First_name VARCHAR2(20),Last_name VARCHAR2(25), Salary NUMBER(8,2),Dept_id NUMBER(4));

• Drop the

EMP Table

DROP TABLE

EMP;

```
Table dropped.
```

• Rename the EMPLOYEES2 table as EMP.

ALTER TABLE EMPLOYEES2 RENAME TO EMP;

```
Table altered.
```

- Add a comment on DEPT and EMP tables. Confirm the modification by describing the table.

```
COMMENT ON TABLE DEPT IS 'This table contains department information.';
COMMENT ON TABLE EMP IS 'This table contains employee information.';
```

- Drop the First_name column from the EMP table and confirm it.

```
ALTER TABLE EMP DROP COLUMN First_name;
```

230701188

Ex.No.: 3		WRITING BASIC SQL SELECT STATEMENTS
Date:	03.06.2024	

Find the Solution for the following:

True OR False

- The following statement executes successfully.

Identify the Errors

```
SELECT employee_id,
 last_name sal*12
 ANNUAL SALARY
FROM employees;
```

False ->Corrected Query and Output

```
Select employee_id,last_name,salary*12 AS "Annual Salary" from Employees;
```

- Show the structure of departments the table. Select all the data from it.

```
DESC department;
```

Select * from Department;

- Create a query to display the last name, job code, hire date, and employee number for each employee, with employee number appearing first.

SELECT employee_id, last_name, job_id,
hire_date FROM employees;

- Provide an alias STARTDATE for the hire date.

SELECT employee_id, last_name, job_id, hire_date AS
STARTDATE FROM employees;

- Create a query to display unique job codes from the employee table.

SELECT DISTINCT job_id FROM employees;

• 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 AS "EMPLOYEE and TITLE" FROM employees;

• Create a query to display all the data from the employees table. Separate each column by a comma. Name the column THE_OUTPUT.

SELECT employee_id || ', ' || last_name || ', ' || job_id || ', ' || hire_date AS THE_OUTPUT FROM employees;

230701188

Ex.No.: 4		**WORKING WITH CONSTRAINTS**
Date:	**07.08.2024**	

• Add a table-level PRIMARY KEY constraint to the EMP table on the ID column.The constraint should be named at creation. Name the constraint my_emp_id_pk.

CREATE TABLE EMP (EmpNo NUMBER(7) PRIMARY KEY,
Last_name VARCHAR2(25) NOT NULL,First_name VARCHAR2(25));

• Create a PRIMARY KEY constraint to the DEPT table using the ID column. The constraint should be named at creation. Name the constraint my_dept_id_pk.

CREATE TABLE DEPT (Dept_id NUMBER(6) NOT NULL,Dept_name VARCHAR2(20) NOT NULL,Manager_id NUMBER(6),
Location_id NUMBER(4),CONSTRAINT my_dept_id_pk PRIMARY KEY (Dept_id));

• Add a column DEPT_ID to the EMP table. Add a foreign key reference on the EMP table that ensures that the employee is not assigned to a nonexistent department. Name the constraint my_emp_dept_id_fk.

ALTER TABLE EMP ADD Dept_id NUMBER(6);
ALTER TABLE EMP ADD CONSTRAINT my_emp_dept_id_fk FOREIGN KEY (Dept_id) REFERENCES DEPT (Dept_id);

• Modify the EMP table. Add a COMMISSION column of NUMBER data type, precision 2, scale 2. Add a constraint to the commission column that ensures that a commission value is greater than zero.

ALTER TABLE EMP
ADD Commission NUMBER(8,2) CONSTRAINT commission_check CHECK (Commission > 0);

230701188

Ex.No.: 5		CREATING VIEWS
Date:	13.08.2024	

• Create a view called EMPLOYEE_VU based on the employee numbers, employee names

and department numbers from the EMPLOYEES table. Change the heading for the employee name to EMPLOYEE.

CREATE VIEW EMPLOYEE_VU AS SELECT employee_id, last_name AS EMPLOYEE, department_id FROM EMPLOYEES;

- Display the contents of the EMPLOYEES_VU view.

SELECT * FROM EMPLOYEE_VU;

- Select the view name and text from the USER_VIEWS data dictionary views.

SELECT view_name, text FROM USER_VIEWS WHERE view_name = 'EMPLOYEE_VU';

- Using your EMPLOYEES_VU view, enter a query to display all employees names and departments.

SELECT EMPLOYEE, department_id FROM EMPLOYEE_VU;

- Create a view named DEPT50 that contains the employee number, employee last names and department numbers for all employees in department 50.Label the view columns EMPNO, EMPLOYEE and DEPTNO. Do not allow an employee to be reassigned to another department through the view.

```
CREATE OR REPLACE VIEW DEPT50 (EMPNO, EMPLOYEE, DEPTNO) AS
SELECT employee_id, last_name,
department_id FROM EMPLOYEES
WHERE department_id =
50 WITH CHECK
OPTION;
```

• Display the structure and contents of the

DEPT50 view. SELECT * FROM DEPT50;

• Attempt to reassign Matos to department 80.

UPDATE DEPT50 SET DEPTNO = 80 WHERE EMPLOYEE = 'Matos';

• Create a view called SALARY_VU based on the employee last names, department names, salaries, and salary grades for all employees. Use the Employees, DEPARTMENTS and JOB_GRADE tables. Label the column Employee,Department, salary, and Grade respectively.

```
CREATE VIEW SALARY_VU AS
SELECT e.last_name AS
    Employee,
    d.department_name AS
    Department, e.salary AS
    Salary,
    j.grade_level AS
Grade FROM
EMPLOYEES e
JOIN DEPARTMENTS d ON e.department_id = d.department_id
JOIN JOB_GRADE j ON e.salary BETWEEN j.lowest_sal AND j.highest_sal;
```

230701188

Ex.No.: 6		RESTRICTING AND SORTING DATA
Date:	14.08.2024	

- Create a query to display the last name and salary of employees earning more than 12000.

SELECT last_name, salary FROM employees WHERE salary > 12000;

- Create a query to display the employee last name and department number for employee number 176.

SELECT last_name, department_id FROM employees WHERE employee_id = 176;

- Create a query to display the last name and salary of employees whose salary is not in the range of 5000 and 12000. (hints: not between).

SELECT last_name, salary FROM employees WHERE salary NOT BETWEEN 5000 AND 12000;

- Display the employee last name, job ID, and start date of employees hired between February 20,1998 and May 1,1998.order the query in ascending order by start date.(hints: between)

SELECT last_name, job_id, hire_date FROM employees WHERE hire_date BETWEEN '02- 20-1998' AND '05-01-1998' ORDER BY hire_date ASC;

• Display the last name and department number of all employees in departments 20 and 50 in alphabetical order by name.(hints: in, orderby)

SELECT last_name, department_id FROM employees WHERE department_id IN (20, 50) ORDER BY last_name ASC;

• Display the last name and salary of all employees who earn between 5000 and 12000 and are in departments 20 and 50 in alphabetical order by name. Label the columns EMPLOYEE, MONTHLY SALARY respectively.(hints: between, in)

SELECT last_name AS "EMPLOYEE", salary AS "MONTHLY SALARY" FROM employees WHERE salary BETWEEN 5000 AND 12000 AND department_id IN (20, 50)
ORDER BY last_name ASC;

• Display the last name and hire date of every employee who was hired in

1994.(hints: like) SELECT last_name, hire_date FROM employees WHERE

hire_date LIKE '%1994%';

• Display the last name and job title of all employees who do not have a manager.(hints: is null)

SELECT last_name, job_id FROM employees WHERE manager_id IS NULL;

• Display the last name, salary, and commission for all employees who earn commissions. Sort data in descending order of salary and commissions.(hints: is not nul,orderby)

SELECT last_name, salary, commission_pct FROM employees WHERE commission_pct IS NOT NULL ORDER BY salary DESC, commission_pct DESC;

• Display the last name of all employees where the third letter of the name is

a.(hints:like) SELECT last_name FROM employees WHERE last_name LIKE '_

_a%';

- Display the last name of all employees who have an a and an ***e*** in their last name.(hints: like)

SELECT last_name FROM employees WHERE last_name LIKE '%a%' AND last_name LIKE '%e%';

- Display the last name and job and salary for all employees whose job is sales representative or stock clerk and whose salary is not equal to 2500 ,3500 or 7000.(hints:in,not in)

SELECT last_name, job_id, salary FROM employees WHERE job_id IN ('SA_REP', 'ST_CLERK') AND salary NOT IN (2500, 3500, 7000);

230701188

Ex.No.: 7		**USING SET OPERATORS**
Date:	**28.08.2024**	

• The HR department needs a list of department IDs 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';
```

• The HR department needs a list of countries that have no departments located in them. Display the country ID and the name of the countries. Use set operators to create this report.

```
SELECT country_id, country_name FROM countries MINUS SELECT country_id, country_name FROM departments;
```

• Produce a list of jobs for departments 10, 50, and 20, in that order. Display job ID and department ID using set operators.

```
SELECT job_id, department_id FROM employees WHERE department_id = 10
UNION
SELECT job_id, department_id FROM employees WHERE department_id = 50
UNION
SELECT job_id, department_id FROM employees WHERE department_id = 20;
```

• Create a report that lists the employee IDs and job IDs 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 (that is, 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;
```

• The HR department needs a report with the following specifications:

• Last name and department ID of all the employees from the EMPLOYEES table, regardless of whether or not they belong to a department.

• Department ID and department name of all the departments from the DEPARTMENTS table, regardless of whether or not they have employees working in them. Write a compound query to accomplish this.

```
SELECT last_name, department_id FROM
employees UNION

SELECT department_name, department_id FROM departments;
```

230701188

Ex.No.: 8		WORKING WITH MULTIPLE TABLES
Date:	04.09.2024	

• Write a query to display the last name, department number, and department name for all employees.

SELECT e.last_name, e.department_id, d.department_name FROM employees e JOIN

departments d ON e.department_id = d.department_id;

• Create a unique listing of all jobs that are in department 80. Include the location of the department in the output.

SELECT DISTINCT e.job_id, d.location_id FROM employees e JOIN departments d ON e.department_id = d.department_id WHERE e.department_id = 80;

• Write a query to display the employee last name, department name, location ID, and city of all employees who earn a commission

SELECT e.last_name, d.department_name, d.location_id, l.city FROM employees e JOIN departments d ON e.department_id = d.department_id JOIN locations l ON d.location_id = l.location_id WHERE e.commission_pct IS NOT NULL;

- Display the employee last name and department name for all employees who have an a(lowercase) in their last names. P

SELECT e.last_name, d.department_name FROM employees e JOIN departments d ON e.department_id = d.department_id WHERE e.last_name LIKE '%a%';

- Write a query to display the last name, job, department number, and department name 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 l.city = 'Toronto';

- Display the employee last name and employee number along with their manager‘s last name and manager number. Label the columns Employee, Emp#, Manager, and Mgr#, Respectively

SELECT e.last_name AS Employee, e.employee_id AS Emp#, m.last_name AS Manager, m.employee_id AS Mgr# FROM employees e LEFT JOIN employees m ON e.manager_id = m.employee_id;

- Modify lab4_6.sql to display all employees including King, who has no manager. Order the results by the employee number.

SELECT e.last_name, e.employee_id, m.last_name AS Manager FROM employees e LEFT JOIN employees m ON e.manager_id = m.employee_id ORDER BY e.employee_id;

• Create a query that displays employee last names, department numbers, and all the employees who work in the same department as a given employee. Give each column an appropriate label

```
SELECT e1.last_name AS Employee, e2.last_name AS Colleague FROM employees e1 JOIN employees e2 ON e1.department_id = e2.department_id WHERE e1.employee_id =
:employee_id;
```

• Show the structure of the JOB_GRADES table. Create a query that displays the name, job, department name, salary, and grade for all employees

```
DESC job_grades;
```

```
SELECT e.last_name, e.job_id, d.department_name, e.salary, j.grade_level FROM employees e JOIN departments d ON e.department_id = d.department_id JOIN job_grades j ON e.salary BETWEEN j.lowest_sal AND j.highest_sal;
```

• Create a query to display the name and hire date of any employee hired after employee Davies.

```
SELECT last_name, hire_date FROM employees WHERE hire_date > (SELECT hire_date FROM employees WHERE last_name = 'Davies');
```

• Display the names and hire dates for all employees who were hired before their managers, along with their manager‘s names and hire dates. Label the columns Employee, Emp Hired, Manager, and Mgr Hired, respectively.

SELECT e.last_name AS Employee, e.hire_date AS Emp_Hired, m.last_name AS Manager, m.hire_date AS Mgr_Hired FROM employees e JOIN employees m ON e.manager_id = m.employee_id WHERE e.hire_date < m.hire_date;

230701188

Ex.No.: 9		**SUB QUERIES**
Date:	**10.09.2024**	

• The HR department needs a query that prompts the user for an employee last name. The

query then displays the last name and hire date of any employee in the same department as the employee whose name they supply (excluding that employee). For example, if the user enters Zlotkey, find all employees who work with Zlotkey (excluding Zlotkey).

SELECT e.last_name, e.hire_date FROM employees e WHERE e.department_id = (SELECT department_id FROM employees WHERE UPPER(last_name) = UPPER('&last_name')) AND UPPER(e.last_name) != UPPER('&last_name');

• 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 order of ascending salary.

SELECT employee_id, last_name, salary FROM employees WHERE salary > (SELECT AVG(salary) FROM employees) ORDER BY salary ASC;

• 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 LOWER(last_name) LIKE
'%u%');
```

- The HR department needs a report that displays the last name, department number, and 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);
```

- Create a report for HR that displays the last name and salary of every employee who reports to King.

```
SELECT last_name, salary FROM employees WHERE manager_id = (SELECT
employee_id FROM employees WHERE UPPER(last_name) = 'KING');
```

- Create a report for HR that displays the department number, last name, and job ID for every employee in the Executive department.

```
SELECT department_id, last_name, job_id FROM employees WHERE
department_id = (SELECT department_id FROM departments WHERE
UPPER(department_name) = 'EXECUTIVE');
```

- Modify the query 3 to display the employee number, last name, and salary of all employees who earn more than the average salary and who work in a department with any employee whose last name contains a *u*.

```
SELECT employee_id, last_name, salary FROM employees WHERE
department_id IN (SELECT department_id FROM employees WHERE
LOWER(last_name) LIKE '%u%') AND salary > (SELECT AVG(salary) FROM
employees);
```

230701188

Ex.No.: 10		AGGREGATING DATA USING GROUP FUNCTIONS
Date:	24.09.2024	

Determine the validity of the following three statements. Circle either True or False.

• Group functions work across many rows to produce one result per

group.

True/False

TRUE

• Group functions include nulls in
calculations. True/False

FALSE

• The WHERE clause restricts rows prior to inclusion in a
group calculation. True/False

TRUE

The HR department needs the following reports:

• 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)) AS "Maximum",ROUND(MIN(salary)) AS "Minimum", ROUND(SUM(salary)) AS "Sum", ROUND(AVG(salary)) AS "Average"FROM employees;

• Modify the above query to display the minimum, maximum, sum, and average salary for each job type.

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

• Write a query to display the number of people with the same job. Generalize the query so that the user in the HR department is prompted for a job title.

SELECT COUNT(*) AS "Number of People" FROM employees WHERE job_id = '&job_title';

• Determine the number of managers without listing them. Label the column Number of Managers. *Hint: Use the MANAGER_ID column to determine the number of managers.*

SELECT COUNT(DISTINCT manager_id) AS "Number of Managers"FROM employees WHERE manager_id IS NOT NULL;

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

SELECT (MAX(salary) - MIN(salary)) AS "DIFFERENCE" FROM employees;

• Create a report to display the manager number and the salary of the lowest-paid employee for that 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) AS "Lowest Salary" FROM employees
WHERE manager_id IS NOT NULL GROUP BY manager_id HAVING MIN(salary) > 6000 ORDER BY MIN(salary) DESC;

• Create a query to display the total number of employees and, of that total, the number of employees hired in 1995, 1996, 1997, and 1998. Create appropriate column headings.

```
SELECT
   COUNT(*) AS "Total Employees",
   SUM(CASE WHEN TO_CHAR(hire_date, 'YYYY') = '1995' THEN 1 ELSE 0 END)
   AS
"Hired in 1995",
   SUM(CASE WHEN TO_CHAR(hire_date, 'YYYY') = '1996' THEN 1 ELSE 0 END)
   AS
"Hired in 1996",
   SUM(CASE WHEN TO_CHAR(hire_date, 'YYYY') = '1997' THEN 1 ELSE 0 END)
   AS
"Hired in 1997",
   SUM(CASE WHEN TO_CHAR(hire_date, 'YYYY') = '1998' THEN 1 ELSE 0 END)
   AS
"Hired in 1998" FROM employees;
```

• Create a matrix query to display the job, the salary for that job based on department number, and the total salary for that job, for departments 20, 50, 80, and 90, giving each column an appropriate heading.

```
SELECT job_id,
   SUM(CASE WHEN department_id = 20 THEN salary ELSE 0 END) AS
   "Dept 20", SUM(CASE WHEN department_id = 50 THEN salary ELSE 0
   END) AS "Dept 50", SUM(CASE WHEN department_id = 80 THEN salary
   ELSE 0 END) AS "Dept 80", SUM(CASE WHEN department_id = 90
   THEN salary ELSE 0 END) AS "Dept 90", SUM(salary) AS "Total Salary"
FROM employees WHERE department_id IN (20, 50, 80, 90) GROUP BY job_id;
```

• Write a query to display each department‘s name, location, number of employees, and the average salary for all the employees in that department. Label the column name-Location, Number of people, and salary respectively. Round the average salary to two decimal places.

```
SELECT d.department_name AS "Department Name", l.city AS
"Location", COUNT(e.employee_id) AS "Number of People",
ROUND(AVG(e.salary), 2) AS "Average Salary" FROM employees e JOIN
departments d ON e.department_id = d.department_id JOIN locations l ON
d.location_id = l.location_id GROUP BY d.department_name, l.city;
```

230701188

Ex.No.: 11		**PL SQL PROGRAMS**
Date:	**09.10.2024**	

PROGRAM 1

Write a PL/SQL block to calculate the incentive of an employee whose ID is 110.

PROGRAM 2

Write a PL/SQL block to show an invalid case-insensitive reference to a quoted and without quoted user-defined identifier.

PROGRAM 3

Write a PL/SQL block to adjust the salary of the employee whose ID 122. Sample table: employees

PROGRAM 4

Write a PL/SQL block to create a procedure using the "IS [NOT] NULL Operator" and show AND operator returns TRUE if and only if both operands are TRUE.

PROGRAM 5

Write a PL/SQL block to describe the usage of LIKE operator including wildcard characters and escape character.

```
Name starts with "D"
Name contains "Dan" followed by one character
Name contains "Daniel_Andrea"

Statement processed.
```

PROGRAM 6

Write a PL/SQL program to arrange the number of two variable in such a way that the small number will store in num_small variable and large number will store in num_large variable.

```
Value in a : 10
Value in b : 5
Smaller number is 5
Larger number is 10

Statement processed.
```

0.00 seconds

PROGRAM 7

Write a PL/SQL procedure to calculate the incentive on a target achieved and display the message either the record updated or not.

```
Record(s) updated

Statement processed.
```

PROGRAM 8

Write a PL/SQL procedure to calculate incentive achieved according to the specific sale limit.

```
incentive
number;
rowcount
number;
Begin
```

```
if sales <
1000 then
incentive:= 0;
elsif sales > 1000 and sales <
2000 then incentive := sales *
0.2;
else
incentive := sales *
0.5; end if;
Update employees set salary = salary + incentive where employee_id =
emp_id; rowcount:= SQL%ROWCOUNT;
if rowcount>0 then
dbms_output.put_line('Record(s)
updated'); else
dbms_output.put_line('No Record(s)
updated'); end if;
end;

Declare
id
numb
er;
sales
numb
er; sal
numb
er;
Begin
id :=
:emp_
id;
sales
:=
:sale;
select salary into sal from employees where employee_id = id;
dbms_output.put_line('Before incentive calculation: '||sal);
calc_incen(id,sales);
select salary into sal from employees where employee_id = id;
dbms_output.put_line('After incentive calculation: '||sal);
end;
```

```
Before incentive calculation: 21000
Record(s) updated
After incentive calculation: 23500

Statement processed.
```

PROGRAM 9

Write a PL/SQL program to count number of employees in department 50 and check whether this department have any vacancies or not. There are 45 vacancies in this department.

```
Total seats : 20
Number of employees in Department 50 : 3
Available vacancies : 17

Statement processed.
```

PROGRAM 10

Write a PL/SQL program to count number of employees in a specific department and check whether this department have any vacancies or not. If any vacancies, how many vacancies are in that department.

```
Total seats : 10
Number of employees in Department : 2
Available vacancies : 8

Statement processed.
```

PROGRAM 11

Write a PL/SQL program to display the employee IDs, names, job titles, hire dates, and salaries of all employees.

```
employee id: 101
name: John
job title: IT_PROG
hire date: 01-jan-1994
salary: 6020
--------------------------------
employee id: 176
name: Jane
job title: HR_REP
hire date: 20-feb-2019
salary: 12500
--------------------------------
employee id: 103
name: Mike
job title: SA_MAN
hire date: 01-mar-1998
salary: 7200
--------------------------------
employee id: 104
name: Emily
job title: AC_ACCOUNT
hire date: 01-jan-1998
salary: 15000
--------------------------------
employee id: 105
name: Robert
job title: ST_CLERK
hire date: 25-jul-2018
salary: 6200
--------------------------------
```

PROGRAM 12

Write a PL/SQL program to display the employee IDs, names, and department names of all employees.

```
employee id: 101
name: John
department name: IT_PROG
--------------------------------
employee id: 176
name: Jane
department name: HR_REP
--------------------------------
employee id: 103
name: Mike
department name: SA_MAN
--------------------------------
employee id: 104
name: Emily
department name: AC_ACCOUNT
--------------------------------
employee id: 105
name: Robert
department name: ST_CLERK
--------------------------------
```

PROGRAM 13

Write a PL/SQL program to display the job IDs, titles, and minimum salaries of all jobs.

```
job id: 101
job title: Software Engineer
minimum salary: 60000
--------------------------------
job id: 102
job title: Data Analyst
minimum salary: 50000
--------------------------------
job id: 103
job title: Project Manager
minimum salary: 70000
--------------------------------
job id: 104
job title: HR Manager
minimum salary: 55000
--------------------------------
job id: 105
job title: Marketing Specialist
minimum salary: 45000
--------------------------------
```

PROGRAM 14

Write a PL/SQL program to display the employee IDs, names, and job history start dates of all employees.

```
employee id: 201
name: James
start date: 01-jan-2010
--------------------------------
employee id: 202
name: King
start date: 01-jan-2012
--------------------------------
employee id: 203
name: Smith
start date: 01-jan-2013
--------------------------------
employee id: 204
name: Steve
start date: 01-jan-2014
--------------------------------
employee id: 205
name: Robert
start date: 01-jan-2015
--------------------------------
```

PROGRAM 15

Write a PL/SQL program to display the employee IDs, names, and job history end dates of all employees.

230701188

Date:	23.10.2024	FUNCTIONS

Program 1

FACTORIAL OF A NUMBER USING FUNCTION

Input : 5

Program 2

Write a PL/SQL program using Procedures IN,INOUT,OUT parameters to retrieve the corresponding book information in library

Input: 5

Input 7

```
The year of publication of the book is 1951
```

Input 3

Ex.No.: 13		WORKING WITH TRIGGER **TRIGGER**
Date:	29.10.2024	

230701188

Program 1

Write a code in PL/SQL to develop a trigger that enforces referential integrity by preventing the deletion of a parent record if child records exist.

Program 2

Write a code in PL/SQL to create a trigger that checks for duplicate values in a specific column and raises an exception if found.

Output:

ORA-20002: Duplicate value detected in specific column.

Program 3

Write a code in PL/SQL to create a trigger that restricts the insertion of new rows if the total of a column's values exceeds a certain threshold.

Output:

ORA-20003: Cannot insert, total column value exceeds threshold.

Program 4

Write a code in PL/SQL to design a trigger that captures changes made to specific columns and logs them in an audit table.

Output:

User_ID	Change_Time	Old_Value	New_Value
SYSTEM	2024-09-19 10:05:00	OldValue1, OldValue2	

NewValue, AnotherNewValue

Program 5

Write a code in PL/SQL to implement a trigger that records user activity (inserts, updates, deletes) in an audit log for a given set of tables.

Output:			
User_ID	**Operation**	**Record_ID**	**Change_Time**
SYSTEM	INSERT	1	2024-09-19 10:10:00
SYSTEM	UPDATE	1	2024-09-19 10:15:00
SYSTEM	DELETE	1	2024-09-19 10:20:00

Program 6

Write a code in PL/SQL to implement a trigger that automatically calculates and updates a running total column for a table whenever new rows are inserted.

Output:

Total_I D

Running_Tota l

1 1500

Program 7

Write a code in PL/SQL to create a trigger that validates the availability of items before allowing an order to be placed, considering stock levels and pending orders.

Output:

ORA-20004: Insufficient stock available for this order.

230701188

Ex.No.: 14		MONGODB
Date:	30.10.2024	

Structure of 'restaurants' collection:

{
"address": {
"building":
"1007",
"coord": [-73.856077, 40.848447],
"street": "Morris Park
Ave", "zipcode": "10462"
},
"borough": "Bronx",
"cuisine":
"Bakery",
"grades": [
{ "date": { "$date": 1393804800000 }, "grade": "A", "score": 2 },
{ "date": { "$date": 1378857600000 }, "grade": "A", "score": 6 },
{ "date": { "$date": 1358985600000 }, "grade": "A", "score": 10 },
{ "date": { "$date": 1322006400000 }, "grade": "A", "score": 9 },
{ "date": { "$date": 1299715200000 }, "grade": "B", "score": 14 }
],
"name": "Morris Park Bake
Shop", "restaurant_id":
"30075445"
}

- Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which prepare dishes except 'American' and 'Chinese' or restaurant's name begin with the letter 'Wil'.

```
db.restaurants.find({ $or: [{ cuisine: { $nin: ["American", "Chinees"] } },{ name: { $regex:
/^Wil/i } }]},{restaurant_id: 1,name: 1,borough: 1,cuisine: 1,_id: 0 });
```

• Write a MongoDB query to find the restaurant Id, name, and grades for those restaurants which achieved a grade of "A" and scored 11 on an ISODate "2014-08-11T00:00:00Z" among many survey dates..

db.restaurants.find({ grades: {$elemMatch: {grade: "A", score: 11}}},{restaurant_id: 1,name: 1, grades: 1, _id: 0 });

• Write a MongoDB query to find the restaurant Id, name and grades for those restaurants where the 2nd element of grades array contains a grade of "A" and score 9 on an ISODate "2014-08- 11T00:00:00Z".

db.restaurants.find({ "grades.1": {$elemMatch: {grade: "A",score: 9}}},{restaurant_id: 1, name: 1, grades: 1, _id: 0 });

• Write a MongoDB query to find the restaurant Id, name, address and geographical location for those restaurants where 2nd element of coord array contains a value which is more than 42 and upto 52..

db.restaurants.find({ "address.coord.1": { $gt: 42, $lte: 52 }},{restaurant_id: 1,name: 1, address: 1, _id: 0 });

• Write a MongoDB query to arrange the name of the restaurants in ascending order along with all the columns.

db.restaurants.find().sort({ name:

1 }); SAMPLE OUTPUT:-

{ _id: ObjectId('671b5e6d56ec9972ca8f5dc4'), address: { building: 5566, coord: [- 73.867377,40.854047] , street: '28th Avenue', zipcode: 10490 } ,
borough: 'Bronx', cuisine: 'BBQ', grades: [{ date: 2014-03-03T00:00:00.028Z, grade: 'A', score: 10 },
{ date: 2013-09-11T00:00:00.028Z, grade: 'A', score: 7},
{ date: 2013-01-24T00:00:00.028Z, grade: 'A', score: 11},
{ date: 2011-11-23T00:00:00.028Z, grade: 'A', score: 9},
{ date: 2011-03-10T00:00:00.028Z, grade: 'B', score: 15}],
name: 'BBQ Haven', restaurant_id: 30075473 }
{ _id: ObjectId('671b5dab56ec9972ca8f5db0'), address: { building: 5566, coord: [-73.859377, 40.850047

],
street: '8th
Avenue',
zipcode:
10470
},
borough:
'Manhattan',
cuisine:
'French',
grades: [
{
date: 2014-03-03T00:00:00.008Z,
grade:
'A',
score:
7
},
{
date: 2013-09-11T00:00:00.008Z,
grade:
'A',
score:
9
},
{
date: 2013-01-24T00:00:00.008Z,
grade:
'A',
score:
10
},{
date: 2011-03-10T00:00:00.008Z,
grade:
'A',
score:
6
}], name: 'Bistro Belle',
restaurant_id: 30075453

}

- Write a MongoDB query to arrange the name of the restaurants in descending along with all the columns.

db.restaurants.find().sort({ name:

-1 }); SAMPLE OUTPUT:-

{
_id:
ObjectId('671b5e9456ec9972ca8f5dc8'),
address: {
buildin
g:
9900,
coord:
[
-73.868977,
40.854847
],
street: '32nd
Avenue',
zipcode:
10494
},
borough:
'Manhattan',
cuisine:
'Russian',
grades: [
{
date: 2014-03-03T00:00:00.032Z,
grade:
'A',
score:
10
},
{
date: 2013-09-11T00:00:00.032Z,
grade:
'B',
score:
5
},
{
date: 2013-01-24T00:00:00.032Z,
grade:
'A',

score:
9
},
{
date: 2011-11-23T00:00:00.032Z,
grade:
'A',
score:
8
},
{
date: 2011-03-10T00:00:00.032Z,
grade:
'A',
score:
11
}

], name: "Tsar's Table",
restaurant_id: 30075477
}
{
_id:
ObjectId('671b5e6d56ec9972ca8f5dbe'),
address: {
buildin
g:
9900,
coord:
[
-73.864977,
40.852847
],
street: '22nd
Avenue',
zipcode:
10484
},
borough:
'Bronx',
cuisine:
'Italian',
grades: [
{

date: 2014-03-03T00:00:00.022Z,
grade:
'A',
score:
8
},
{
date: 2013-09-11T00:00:00.022Z,
grade:
'B',
score:
5
},
{
date: 2013-01-24T00:00:00.022Z,
grade:
'A',
score:
12
},
{
date: 2011-11-23T00:00:00.022Z,
grade:
'A',
score:
9
},
{
date: 2011-03-10T00:00:00.022Z,
grade:
'A',
score:
14
}
], name: 'Trattoria Bella',
restaurant_id: 30075467
}

• Write a MongoDB query to arrange the name of the cuisine in ascending order and for that the same cuisine borough should be in descending order.
db.restaurants.find().sort({ cuisine: 1, borough: -

1 }); SAMPLE OUTPUT:-

{
_id:
ObjectId('671b5d549d3d63480e0a64e
9'), address: {
buildin
g:
2233,
coord:
[
-73.858177,
40.849447
],
street: '5th
Avenue',
zipcode:
10467
},
borough:
'Bronx',
cuisine:
'American',
grades: [
{
date: 2014-03-03T00:00:00.005Z,
grade:
'A',
score:
10
},
{
date: 2013-09-11T00:00:00.005Z,
grade:
'A',
score:
6
},
{
date: 2013-01-24T00:00:00.005Z,
grade:
'B',
score:
12
},
{

date: 2011-11-23T00:00:00.005Z,
grade:
'A',
score:
9
},
{
date: 2011-03-10T00:00:00.005Z,
grade:
'A',
score:
14

}
], name: 'Burger Bistro',
restaurant_id: 30075450
}
{
_id:
ObjectId('671b5e6d56ec9972ca8f5dc4'),
address: {
buildin
g:
5566,
coord:
[
-73.867377,
40.854047
],
street: '28th
Avenue',
zipcode:
10490
},
borough:
'Bronx',
cuisine:
'BBQ',
grades: [
{
date: 2014-03-03T00:00:00.028Z,
grade:
'A',

score:
10
},
{
date: 2013-09-11T00:00:00.028Z,
grade:
'A',
score:
7
},
{
date: 2013-01-24T00:00:00.028Z,
grade:
'A',
score:
11
},
{
date: 2011-11-23T00:00:00.028Z,
grade:
'A',
score:
9
},
{
date: 2011-03-10T00:00:00.028Z,
grade:
'B',
score:
15
}
], name: 'BBQ Haven',
restaurant_id: 30075473
}

- Write a MongoDB query to know whether all the addresses contain the street

or not. db.restaurants.find({"address.street": { $exists: false }});

- Write a MongoDB query which will select all documents in the restaurants collection where the coord field value is Double.

db.restaurants.find({ "address.coord": { $type: "double" }});

SAMPLE OUTPUT:-

{
_id:
ObjectId('671b92d339ec8a9bc8b6588b'),
address: {
building
: '1007',
coord: [
-73.856077,
40.848447
],
street: 'Morris Park
Ave', zipcode:
'10462'
},
borough:
'Bronx',
cuisine:
'Bakery',
grades: [
{
date: 2014-03-03T00:00:00.000Z,
grade:
'A',
score:
2
},
{

date: 2013-09-11T00:00:00.000Z,
grade:
'A',
score:
6
},
{
date: 2013-01-24T00:00:00.000Z,
grade:
'A',
score:
10

},
{
date: 2011-11-23T00:00:00.000Z,
grade:
'A',
score:
9
},
{
date: 2011-03-10T00:00:00.000Z,
grade:
'B',
score:
14
}
], name: 'Morris Park Bake
Shop', restaurant_id:
'30075445'
}
{
_id:
ObjectId('671b5d549d3d63480e0a64e
5'), address: {
buildin
g:
1234,
coord:
[
-73.856577,
40.848647
],
street: '1st
Avenue',
zipcode:
10463
},
borough:
'Bronx',
cuisine:
'Italian',
grades: [
{
date: 2014-03-03T00:00:00.001Z,

grade:
'A',
score:
5
},
{
date: 2013-09-11T00:00:00.001Z,
grade:
'A',
score:
8
},

{
date: 2013-01-24T00:00:00.001Z,
grade:
'B',
score:
12
},
{
date: 2011-11-23T00:00:00.001Z,
grade:
'A',
score:
7
},
{
date: 2011-03-10T00:00:00.001Z,
grade:
'A',
score:
15
}
], name: 'Pasta Palace',
restaurant_id: 30075446
}

- Write a MongoDB query which will select the restaurant Id, name and grades for those restaurants which return 0 as a remainder after dividing the score by 7.

db.restaurants.find({"grades.score": { $mod: [7, 0] }},{restaurant_id: 1,name: 1,grades: 1, _id: 0});

SAMPLE OUTPUT:-

{
grades: [
{
date: 2014-03-03T00:00:00.000Z,
grade:
'A',
score:
2
},
{
date: 2013-09-11T00:00:00.000Z,
grade:
'A',
score:
6
},
{
date: 2013-01-24T00:00:00.000Z,
grade:
'A',
score:
10
},
{
date: 2011-11-23T00:00:00.000Z,

grade:
'A',
score:
9
},
{
date: 2011-03-10T00:00:00.000Z,
grade:
'B',
score:
14
}
], name: 'Morris Park Bake
Shop', restaurant_id:
'30075445'
}
{

grades: [
{
date: 2014-03-03T00:00:00.001Z,
grade:
'A',
score:
5
},
{
date: 2013-09-11T00:00:00.001Z,
grade:
'A',
score:
8
},
{
date: 2013-01-24T00:00:00.001Z,
grade:
'B',
score:
12
},
{
date: 2011-11-23T00:00:00.001Z,
grade:
'A',
score:
7
},
{
date: 2011-03-10T00:00:00.001Z,
grade:
'A',
score:
15
}
], name: 'Pasta Palace',
restaurant_id: 30075446
}

- Write a MongoDB query to find the restaurant name, borough, longitude and attitude and cuisine for those restaurants which contain 'mon' as three letters somewhere in its name.

db.restaurants.find({name: { $regex: /mon/i }},{name: 1, borough: 1,"address.coord.0": 1, "address.coord.1": 1, cuisine: 1, _id: 0});

- Write a MongoDB query to find the restaurant name, borough, longitude and latitude and cuisine for those restaurants which contain 'Mad' as the first three letters of its name.

db.restaurants.find({name: { $regex: /^Mad/i }},{name: 1,borough: 1,"address.coord.0": 1, "address.coord.1": 1, cuisine: 1,_id: 0});

- Write a MongoDB query to find the restaurants that have at least one grade with a score of less than 5.

db.restaurants.find({"grades.score": { $lt: 5 }});

SAMPLE OUTPUT:-

{
_id:
ObjectId('671b92d339ec8a9bc8b6588b'),
address: {
building
: '1007',
coord: [
-73.856077,
40.848447
],
street: 'Morris Park
Ave', zipcode:
'10462'
},
borough:
'Bronx',
cuisine:
'Bakery',
grades: [
{
date: 2014-03-03T00:00:00.000Z,
grade:
'A',
score:
2
},

{
date: 2013-09-11T00:00:00.000Z,
grade:
'A',
score:
6
},
{
date: 2013-01-24T00:00:00.000Z,

grade:
'A',
score:
10
},
{
date: 2011-11-23T00:00:00.000Z,
grade:
'A',
score:
9
},
{
date: 2011-03-10T00:00:00.000Z,
grade:
'B',
score:
14
}
], name: 'Morris Park Bake
Shop', restaurant_id:
'30075445'
}
{
_id:
ObjectId('671b5d549d3d63480e0a64e
6'), address: {
buildin
g:
5678,
coord:
[
-73.856977,
40.848847

],
street: '2nd
Avenue',
zipcode:
10464
},
borough:
'Manhattan',
cuisine:
'Chinese',
grades: [
{
date: 2014-03-03T00:00:00.002Z,
grade:
'B',
score:
4
},
{
date: 2013-09-11T00:00:00.002Z,
grade:
'A',
score:
9
},
{
date: 2013-01-24T00:00:00.002Z,
grade:
'A',
score:
10
},
{

date: 2011-11-23T00:00:00.002Z,
grade:
'A',
score:
8
},
{
date: 2011-03-10T00:00:00.002Z,
grade:
'B',

score:
16
}
], name: 'Dragon Wok',
restaurant_id: 30075447
}

- Write a MongoDB query to find the restaurants that have at least one grade with a score of less than 5 and that are located in the borough of Manhattan.

```
db.restaurants.find({"grades.score": { $lt: 5 },borough: "Manhattan"});
```

- Write a MongoDB query to find the restaurants that have at least one grade with a score of less than 5 and that are located in the borough of Manhattan or Brooklyn.

```
db.restaurants.find({"grades.score": { $lt: 5 },borough: { $in: ["Manhattan", "Brooklyn"] }});
```

- Write a MongoDB query to find the restaurants that have at least one grade with a score of less than 5 and that are located in the borough of Manhattan or Brooklyn, and their cuisine is not American.

```
db.restaurants.find({"grades.score": { $lt: 5 },borough: { $in: ["Manhattan", "Brooklyn"]}, cuisine: { $ne: "American" }});
```

- Write a MongoDB query to find the restaurants that have at least one grade with a score of less than 5 and that are located in the borough of Manhattan or Brooklyn, and their cuisine is not American or Chinese.

```
db.restaurants.find({"grades.score": { $lt: 5 },borough: { $in: ["Manhattan", "Brooklyn"]
},cuisine: { $nin: ["American", "Chinese"] }});
```

- Write a MongoDB query to find the restaurants that have a grade with a score of 2 and a

grade with a score of 6.

```
db.restaurants.find({grades: {$all: [{ $elemMatch: { score: 2 } },{ $elemMatch: { score:
6 }
}]}});
```

SAMPLE OUTPUT:-


```
{
_id:
ObjectId('671b92d339ec8a9bc8b6588b'),
address: {
building
: '1007',
coord: [
-73.856077,
40.848447
],
street: 'Morris Park
Ave', zipcode:
'10462'
},
borough:
'Bronx',
cuisine:
'Bakery',
grades: [
{
date: 2014-03-03T00:00:00.000Z,
grade:
'A',
score:
2
},
{
date: 2013-09-11T00:00:00.000Z,
grade:
'A',
```

score:
6
},
{
date: 2013-01-24T00:00:00.000Z,
grade:
'A',
score:
10
},
{
date: 2011-11-23T00:00:00.000Z,
grade:
'A',
score:
9
},
{
date: 2011-03-10T00:00:00.000Z,
grade:
'B',
score:
14
}

], name: 'Morris Park Bake
Shop', restaurant_id:
'30075445'
}
{
_id:
ObjectId('671b5c5f9d3d63480e0a64e
4'), address: {
buildin
g:
1007,
coord:
[
-73.856077,
40.848447
],
street: 'Morris Park
Ave', zipcode:
10462

},
borough:
'Bronx',
cuisine:
'Bakery',
grades: [
{
date: 2014-03-03T00:00:00.000Z,
grade:
'A',
score:
2
},
{
date: 2013-09-11T00:00:00.000Z,
grade:
'A',
score:
6
},
{
date: 2013-01-24T00:00:00.000Z,
grade:
'A',
score:
10
},
{
date: 2011-11-23T00:00:00.000Z,
grade:
'A',
score:
9
},
{
date: 2011-03-10T00:00:00.000Z,
grade:
'B',
score:
14
}
], name: 'Morris Park Bake
Shop', restaurant_id:
30075445

}

- Write a MongoDB query to find the restaurants that have a grade with a score of 2 and a grade with a score of 6 and are located in the borough of Manhattan.

```
db.restaurants.find({borough: "Manhattan",grades: {$all: [{ $elemMatch: { score: 2 } },{
$elemMatch: { score: 6 }}]}});
```

- Write a MongoDB query to find the restaurants that have a grade with a score of 2 and a grade with a score of 6 and are located in the borough of Manhattan or Brooklyn.

```
db.restaurants.find({borough: { $in: ["Manhattan", "Brooklyn"] },grades: {$all: [
{ $elemMatch: { score: 2 } },{ $elemMatch: { score: 6 }}]}});
```

- Write a MongoDB query to find the restaurants that have a grade with a score of 2 and a grade with a score of 6 and are located in the borough of Manhattan or Brooklyn, and their cuisine is not American.

```
db.restaurants.find({borough: { $in: ["Manhattan", "Brooklyn"] },grades: {$all: [{
$elemMatch:
{ score: 2 }},{ $elemMatch: { score: 6 }}]},cuisine: { $ne: "American" }});
```

- Write a MongoDB query to find the restaurants that have a grade with a score of 2 and a grade with a score of 6 and are located in the borough of Manhattan or Brooklyn, and their cuisine is not American or Chinese.

```
db.restaurants.find({borough: { $in: ["Manhattan", "Brooklyn"] },grades: {$all: [
{ $elemMatch: { score: 2 } },{ $elemMatch: { score: 6 }}]},cuisine: { $nin:
["American", "Chinese"] }});
```

- Write a MongoDB query to find the restaurants that have a grade with a score of 2 or a grade

with a score of 6.
db.restaurants.find({$or: [{ "grades.score": 2 },{ "grades.score": 6 }]}); SAMPLE

{
_id:
ObjectId('671b5d549d3d63480e0a64e
9'), address: {
buildin
g:
2233,
coord:
[
-73.858177,
40.849447
],
street: '5th
Avenue',
zipcode:
10467
},
borough:
'Bronx',
cuisine:
'American',
grades: [
{
date: 2014-03-03T00:00:00.005Z,
grade:
'A',
score:
10
},
{
date: 2013-09-11T00:00:00.005Z,
grade:
'A',

score:
6
},
{
date: 2013-01-24T00:00:00.005Z,
grade:
'B',
score:
12
},
{
date: 2011-11-23T00:00:00.005Z,
grade:
'A',
score:
9
},
{
date: 2011-03-10T00:00:00.005Z,
grade:
'A',
score:
14
}

], name: 'Burger Bistro',
restaurant_id: 30075450
}
{
_id:
ObjectId('671b5dab56ec9972ca8f5d
af'), address: {
buildin
g:
4455,
coord:
[
-73.858977,
40.849847
],
street: '7th
Avenue',
zipcode:
10469

},
borough:
'Bronx',
cuisine:
'Thai',
grades: [
{
date: 2014-03-03T00:00:00.007Z,
grade:
'A',
score:
9
},
{
date: 2013-09-11T00:00:00.007Z,
grade:
'B',
score:
6
},
{
date: 2013-01-24T00:00:00.007Z,
grade:
'A',
score:
12
},
{
date: 2011-11-23T00:00:00.007Z,
grade:
'A',
score:
8
},
{
date: 2011-03-10T00:00:00.007Z,
grade:
'B',
score:
14
}
], name: 'Thai Delight',
restaurant_id: 30075452
}

Sample document of 'movies' collection

{
_id: ObjectId("573a1390f29313caabcd42e8"),
plot: 'A group of bandits stage a brazen train hold-up, only to find a determined posse hot on their heels.',
genres: ['Short', 'Western'
], runtime: 11,
cast: [
'A.C. Abadie',
"Gilbert M. 'Broncho Billy'
Anderson", 'George Barnes',
'Justus D. Barnes'
],
poster: 'https://m.media-amazon.com/images/M/MV5BMTU3NjE5NzYtYTYyNS00MDVmLWIwYjgtMmYwYWIxZ DYyNzU2XkEyXkFqcGdeQXVyNzQzNzQxNzI@._V1_SY1000_SX677_AL_.jpg',
title: 'The Great Train Robbery',
full plot: "Among the earliest existing films in American cinema - notable as the first film that presented a narrative story to tell - it depicts a group of cowboy outlaws who hold up a train and rob the passengers. They are then pursued by a Sheriff's posse. Several scenes have color included - all hand tinted.",
languages: ['English'],
released: ISODate("1903-12-01T00:00:00.000Z"),
directors: ['Edwin S.
Porter'], rated: 'TV-G',
awards: { wins: 1, nominations: 0, text: '1 win.'
}, lastupdated: '2015-08-13
00:27:59.177000000',
year: 1903,
imdb: { rating: 7.4, votes: 9847, id:
439 }, countries: ['USA'],
type:
'movi
e',
tomat
oes: {
viewer: { rating: 3.7, numReviews: 2559, meter: 75 },
fresh: 6,
critic: { rating: 7.6, numReviews: 6, meter: 100 },
rotten: 0,
lastUpdated: ISODate("2015-08-08T19:16:10.000Z")
}

• Find all movies with full information from the 'movies' collection that released in the year 1893.

```
db.movies.find({ year: 1893 });
```

• Find all movies with full information from the 'movies' collection that have a runtime greater than 120 minutes.

```
db.movies.find({ runtime: { $gt: 120 } });
```

SAMPLE OUTPUT:-


```
{
_id: ObjectId('573a1390f29313caabcd42ec'),
plot: 'An astronaut stranded on Mars must survive
alone.', genres: [
'
S
c
i
-
F
i
'
,
'
D
r
a
m
a
'
],
runti
me:
135,
```

cast:
[
'Matt
Damon',
'Jessica
Chastain'
], poster: 'https://m.media-
amazon.com/images/poster4.jpg', title: 'Mars Alone',
fullplot: 'An astronaut, left alone on Mars, struggles to survive
with limited resources while awaiting rescue.',
langua
ges: [
'Englis
h'
],
released: 2015-10-02T00:00:00.000Z,
directo
rs: [
'Ridley
Scott'
],
rated:
'PG-
13',
awards
: {
wins:
8,
nominations: 6,
text: '8 wins & 6 nominations.'
},
lastupdated: '2021-08-09 17:22:30.000000000',
year:
2015,
imdb:
{
rating
: 8,
votes: 25650,
id: 443
},

countri
es: [
'USA'

],
type:
'movi
e',
tomat
oes: {
view
er: {
rating
: 4.5,
numReviews: 2201,
meter: 93
},
fresh:
18,
critic:
{
rating
: 8.5,
numReviews: 25,
meter: 96
},
rotten: 1,
lastUpdated: 2021-07-19T21:20:55.000Z
}
}

- Find all movies with full information from the 'movies' collection that have the "Short" genre.

```
db.movies.find({ genres: "Short" });
```

SAMPLE OUTPUT:-

```
{
_id: ObjectId('573a1390f29313caabcd42e8'),
plot: 'A group of bandits stage a brazen train hold-up, only to find
a determined posse hot on their heels.',
g
e
n
r
e
s:
[
'
S
h
o
rt
',
'
W
e
st
e
r
n'
],
runti
me:
11,
cast:
[
'A.C. Abadie',
''Gilbert M. 'Broncho Billy'
Anderson'', 'George Barnes',
'Justus D. Barnes'
```

], poster:
'https://m.media-amazon.com/images/M/MV5BMTU3NjE5NzYtYTYyNS
00MDVmLWIwYjgtMmYwYWIxZDYyNzU2XkEyXkFqcGdeQXVyNzQzNz
QxNzI@._V1_SY1000_SX677_AL_.jpg',
title: 'The Great Train Robbery',
fullplot: "Among the earliest existing films in American
cinema - notable as the first film that presented a narrative
story to tell - it depicts a group of cowboy outlaws who hold
up a train and rob the passengers. They are then pursued by a
Sheriff's posse. Several scenes have color included - all hand
tinted.",
langua
ges: [
'Englis
h'
],
released: 1903-12-01T00:00:00.000Z,
directors
: [
'Edwin
S. Porter'
],
rated:
'TV-
G',
award
s: {
wins:
1,
nominations: 0,
text: '1 win.'
},
lastupdated: '2015-08-13 00:27:59.177000000',
year:
1903,
imdb:
{
rating
: 7.4,
votes: 9847,

id: 439
},

countri
es: [
'USA'
],
type:
'movi
e',
tomat
oes: {
view
er: {
rating
: 3.7,
numReviews: 2559,
meter: 75
},
fresh:
6,
critic:
{
rating
: 7.6,
numReviews: 6,
meter: 100
},
rotten: 0,
lastUpdated: 2015-08-08T19:16:10.000Z
}
}

- Retrieve all movies from the 'movies' collection that were directed by “William K.L. Dickson” and include complete information for each movie.

```
db.movies.find({ directors: "William K.L. Dickson" });
```

- Retrieve all movies from the 'movies' collection that were released in the USA and include complete information for each movie.

```
db.movies.find({ countries: "USA" });
```

• Retrieve all movies from the 'movies' collection that have complete information and are rated as "UNRATED".

```
db.movies.find({ rated: "UNRATED" });
```

• Retrieve all movies from the 'movies' collection that have complete information and have received more than 1000 votes on IMDb.

```
db.movies.find({ "imdb.votes": { $gt: 1000 } });
```

• Retrieve all movies from the 'movies' collection that have complete information and have an IMDb rating higher than 7.

```
db.movies.find({ "imdb.rating": { $gt: 7 } });
```

• Retrieve all movies from the 'movies' collection that have complete information and have a viewer rating higher than 4 on Tomatoes.

```
db.movies.find({ "tomatoes.viewer.rating": { $gt: 4 } });
```

- Retrieve all movies from the 'movies' collection that have received an

award. db.movies.find({ "awards.wins": { $gt: 0 } });

- Find all movies with title, languages, released, directors, writers, awards, year, genres, runtime, cast, countries from the 'movies' collection in MongoDB that have at least one nomination.

db.movies.find({ "awards.nominations": { $gt: 0 }},{title: 1,languages: 1,released: 1,

directors: 1, writers: 1,awards: 1,year: 1,genres: 1,runtime: 1,cast: 1,countries: 1});

- Find all movies with title, languages, released, directors, writers, awards, year, genres, runtime, cast, countries from the 'movies' collection in MongoDB with cast including "Charles Kayser".

```
db.movies.find({ cast: "Charles Kayser" },{title: 1,languages: 1,released: 1,directors: 1,writers: 1,awards: 1,year: 1,genres: 1,runtime: 1,cast: 1,countries: 1});
```

- Retrieve all movies with title, languages, released, directors, writers, countries from the 'movies' collection in MongoDB that was released on May 9, 1893.

```
db.movies.find({ released: ISODate("1893-05-09T00:00:00Z") },{title: 1,languages: 1,released: 1,directors: 1,writers: 1,countries: 1});
```

- Retrieve all movies with title, languages, released, directors, writers, countries from the 'movies' collection in MongoDB that have the word "scene" in the title.

```
db.movies.find(
{ title: { $regex: /scene/i } },{title: 1,languages: 1,released: 1,directors: 1,writers: 1,
c
ountries: 1});
```

230701188

Ex.No.: 15		OTHER DATABASE OBJECTS
Date:	04.11.2024	

- Create a sequence to be used with the primary key column of the DEPT table. The

sequence should start at 200 and have a maximum value of 1000. Have your sequence increment by ten numbers. Name the sequence DEPT_ID_SEQ.

```
Sequence created.
```

Create Sequence dept_id_sequence start with 200 increment by 10 maxvalue 1000;

- Write a query in a script to display the following information about your sequences: sequence name, maximum value, increment size, and last number

SELECT sequence_name, max_value,increment_by AS increment_size,last_number FROM user_sequences WHERE sequence_name = 'DEPT_ID_SEQUENCE';

• Write a script to insert two rows into the DEPT table. Name your script lab12_3.sql. Be sure to use the sequence that you created for the ID column. Add two departments named Education and Administration. Confirm your additions. Run the commands in your script.

Insert into departments values(dept_id_sequence.nextval,'HR',111,1010,'US','United States'); Insert into departments
values(dept_id_seq.nextval,'Admin',112,1011,'IN','India');

• Create a nonunique index on the foreign key column (DEPT_ID) in the EMP table.

Create index emp_dept_index on Employees(department_id);

• Display the indexes and uniqueness that exist in the data dictionary for the EMP table.
SELECT index_name, uniqueness FROM user_indexes WHERE table_name =

'Employees'; Output :

Index_name :
EMPLOYEE_INDEX
Uniqueness : NONUNIQUE

Ex.No.: 16		CONTROLLING USER ACCESS
Date:	06.11.2024	

• What privilege should a user be given to log on to the Oracle Server? Is this a system or an

object

privil

ege?

• The privilege is CREATE SESSION. This privilege allows a user to log on to the Oracle Server. It is a system privilege, not an object privilege.

• What privilege should a user be given to create tables?

• The privilege is CREATE TABLE . This is a system privilege that allows a user to create tables in their own schema.

• If you create a table, who can pass along privileges to other users on your table?

• The owner of the table (the user who created it) can pass along privileges to other users. This is done using the GRANT command. For example:

GRANT SELECT ON my_table TO other_user;

• You are the DBA. You are creating many users who require the same system privileges. What should you use to make your job easier?

• Use a role to bundle common system privileges.
Assign this role to users instead of granting privileges individually.

• What command do you use to change your password?

• ALTER USER username IDENTIFIED BY new_password;

• Grant another user access to your DEPARTMENTS table. Have the user grant you query Access to his or her DEPARTMENTS table.

• GRANT SELECT ON DEPARTMENTS TO other_user;
• GRANT SELECT ON DEPARTMENTS TO your_username;

• Query all the rows in your DEPARTMENTS table.

• SELECT * FROM DEPARTMENTS;

• Add a new row to your DEPARTMENTS table. Team 1 should add Education as department

number 500. Team 2 should add Human Resources department number 510. Query the other team‘s table.
• Team 1 should

execute: INSERT INTO DEPARTMENTS

(DEPARTMENT_ID, DEPARTMENT_NAME) VALUES

(500,
'Education
');
COMMIT;

• Team 2 should execute:

INSERT INTO DEPARTMENTS (DEPARTMENT_ID, DEPARTMENT_NAME)
VALUES
(510, 'Human Resources');
COMMIT;

• To query the other team’s table:

SELECT * FROM other_user.DEPARTMENTS;

- Query the USER_TABLES data dictionary to see information about the tables that you own.

- SELECT * FROM USER_TABLES;

- Revoke the SELECT privilege on your table from the other team.

- To revoke the SELECT privilege on your table from the

other team: REVOKE SELECT ON DEPARTMENTS

FROM other_user;

- Remove the row you inserted into the DEPARTMENTS table in step 8 and save the changes.

- To remove the row you inserted into the DEPARTMENTS table and save the changes: For Team 1 (removing the Education department with ID 500):
DELETE FROM DEPARTMENTS WHERE DEPARTMENT_ID = 500; COMMIT;
For Team 2 (removing the Human Resources department with ID 510):

DELETE FROM DEPARTMENTS WHERE DEPARTMENT_ID = 510; COMMIT;