Date:		

1. 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 JOIN employees e2 ON e.department_id = e2.department_id WHERE e2.last_name = :emp_name
AND e.employee_id != e2.employee_id;

LAST_NAME	HIRE_DATE
Johnson	03/01/1998
Austin	08/22/2021

2. 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;

EMPLOYEE_ID	LAST_NAME	SALARY
176	Smith	12500
106	Wilson	13500
104	Davis	15000
107	Andrea	16000

3. 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 DISTINCT e1.employee_id, e1.last_name FROM employees e1 JOIN employees e2 ON e1.department_id = e2.department_id WHERE e2.last_name LIKE '%u%';

EMPLOYEE_ID	LAST_NAME
101	Matos
103	Johnson
109	Austin

4. 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 e.last_name, e.department_id, e.job_id FROM employees e
JOIN departments d ON e.department_id = d.department_id WHERE d.location_id = 1700;

LAST_NAME	DEPARTMENT_ID	JOB_ID
Miller	10	ST_CLERK
Andrea	10	IT_PROG

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

SELECT e.last_name, e.salary FROM employees e JOIN employees m ON e.manager_id = m.employee_id WHERE m.last_name = 'King';

LAST_NAME	SALARY
Smith	12500
Davis	15000
Andrea	16000

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

SELECT e.department_id, e.last_name, e.job_id FROM employees e JOIN departments dON e.department_id = d.department_id WHERE d.department_name = 'Executive';

DEPARTMENT_ID	LAST_NAME	JOB_ID
50	Matos	IT_PROG
50	Johnson	SA_MAN
50	Austin	AC_MGR

7. 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 anyemployee whose last name contains a *u*.

SELECT e1.employee_id, e1.last_name, e1.salary FROM employees e1 JOIN employees e2 ON e1.department_id = e2.department_id WHERE e2.last_name LIKE'%u%' AND e1.salary > (SELECT AVG(salary) FROM employees);

E **LS**

EMPLOYEE_ID	LAST_NAME	SALARY
106	Wilson	13500
104	Davis	15000

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

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

1. Group functions work across many rows to produce one result per group. True/False

TRUE

2. Group functions include nulls in calculations.
True/False

FALSE

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

TRUE

The HR department needs the following reports:

4. 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;
```

Maximum	Minimum	Sum	Average
16000	4600	158500	7925

5. 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;
```

JOB_ID	Maximum	Minimum	Sum	Average
IT_PROG	16000	6000	51600	8600
AC_ACCOUNT	15000	15000	15000	15000
AC_MGR	7100	7100	7100	7100
SA_MAN	7200	7200	7200	7200
SA_REP	13500	5500	30800	7700
HR_REP	12500	4600	35300	7060
ST_CLERK	6200	5300	11500	5750

6.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';
```

Number of People
6

7. 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;
```

Number of Managers
5

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

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

DIFFERENCE
11400

9. 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;

MANAGER_ID	Lowest Salary
103	13500
101	12500

10. 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;

Total Employees	Hired in 1995	Hired in 1996	Hired in 1997	Hired in 1998
20	1	1	2	3

11. 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;
```

JOB_ID	Dept 20	Dept 50	Dept 80	Dept 90	Total Salary
IT_PROG	0	6000	0	0	6000
AC_ACCOUNT	15000	0	0	0	15000
AC_MGR	0	7100	0	0	7100
SA_MAN	0	7200	0	0	7200
SA_REP	0	0	13500	0	13500
HR_REP	4600	0	0	0	4600

12.Write a query to display each department‘s name, location, number of employees, andthe 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 twodecimal 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 "AverageSalary" 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;
```

Department Name	Location	Number of People	Average Salary
IT	London	3	6766.67
ST_CLERK	Dubai	1	13500
ST_CLERK	Sydney	1	5300
Customer Service	Mumbai	1	12500
Admin	New York	2	11100
ST_CLERK	San Francisco	2	9800

Ex.No.: 11		PL SQL PROGRAMS
Date:		

PROGRAM 1

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

```
declare
a employees.employee_id%type;
b employees.salary%type;
begin
Select salary into a from employees where employee_id = 110;
b:=0.05*a;
dbms_output.put_line('Salary after incentive : '||(a+b));
end;
```

```
Salary after incentive : 6300

Statement processed.

0.01 seconds
```

PROGRAM 2

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

```
declare
non_quoted_variable varchar2(10) := 'Hi';
"quoted_variable" varchar2(10) := 'Hello';
begin
dbms_output.put_line(NON_QUOTED_VARIABLE);
dbms_output.put_line("quoted_variable");
dbms_output.put_line("QUOTED_VARIABLE");
end;
```

```
Hi
Hello

Statement processed.
```

```
ORA-06550: line 7, column 23:
PLS-00201: identifier 'QUOTED_VARIABLE' must be declared
ORA-06550: line 7, column 1:
PL/SQL: Statement ignored
```

PROGRAM 3

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

```
declare
old_salary employees.salary%type;
new_salary employees.salary%type;
begin
new_salary:= :sal;
Select salary into old_salary from employees where employee_id = 122;
dbms_output.put_line('Before updation: '||old_salary);
Update employees set salary = salary + new_salary where employee_id = 122;
Select salary into new_salary from employees where employee_id = 122;
dbms_output.put_line('After updation: '||new_salary);
end;
```

```
Before updation: 8000
After updation: 9000

Statement processed.

0.00 seconds
```

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.

```
Create or replace procedure proc1( a boolean, b boolean) IS
BEGIN
if(a is not null) and (b is not null) then
if(a = TRUE and b = TRUE) then
dbms_output.put_line('TRUE');
else
dbms_output.put_line('FALSE');
end if;
else
dbms_output.put_line('NULL VALUES in arguments');
end if;
end proc1;


BEGIN
proc1(TRUE,TRUE);
proc1(TRUE,FALSE);
proc1(NULL,NULL);
end;
```

```
TRUE
FALSE
NULL VALUES in arguments

Statement processed.

0.00 seconds
```

PROGRAM 5

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

```
Declare
name varchar2(20);
num number(3);
Begin
num := :n;
Select first_name into name from employees where employee_id=num;
if name like 'D%' then
dbms_output.put_line('Name starts with "D"');
end if;
if name like 'Dan_el%' then
dbms_output.put_line('Name contains "Dan" followed by one character');
end if;
name := 'Daniel_Andrea';
if name like 'Daniel\_Andrea' escape '\' then
dbms_output.put_line('Name contains "Daniel_Andrea"');
end if;
end;
```

```
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.

```
declare
a number(2);
b number(2);
num_small number(2);
num_large  number(2);
begin
a := :s;
b := :l;
dbms_output.put_line('Value in a : '||a);
dbms_output.put_line('Value in b : '||b);
if a>b then
num_small := b;
num_large := a;
else
num_small :=a;
num_large :=b;
end if;
dbms_output.put_line('Smaller number is '||num_small);
dbms_output.put_line('Larger number is '||num_large);
end;
```

```
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.

```
Create or replace procedure calc_incen(emp_id number,achievement number,target number)
AS
incentive number;
rowcount number;
Begin
if achievement > target then
incentive:= achievement*0.2;
else
incentive:=0;
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 number;
achievement number;
target number;
Begin
id := :emp_id;
achievement := :achieve;
target := :target_;
calc_incen(id,achievement,target);
end;
```

```
Record(s) updated

Statement processed.
```

PROGRAM 8

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

```
Create or replace procedure calc_incen(emp_id number,sales number) AS
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 number;
sales number;
sal number;
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.

```
declare
emp_count number;
vacancy number := 20;
begin
Select count(*) into emp_count from employees where department_id = 10;
dbms_output.put_line('Total seats : '||vacancy);
dbms_output.put_line('Number of employees in Department 50 : '||emp_count);
if emp_count>vacancy then
dbms_output.put_line('No vacancies available');
else
dbms_output.put_line('Available vacancies : '||(vacancy-emp_count));
end if;
end;
```

```
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.

```
declare
dept_id number;
emp_count number;
vacancy number := 10;
begin
dept_id := :id;
Select count(*) into emp_count from employees where department_id = dept_id;
dbms_output.put_line('Total seats : '||vacancy);
dbms_output.put_line('Number of employees in Department : '||emp_count);
if emp_count>vacancy then
dbms_output.put_line('No vacancies available');
else
dbms_output.put_line('Available vacancies : '||(vacancy-emp_count));
end if;
end;
```

```
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.

```
begin
for i in (select employee_id, first_name, job_id, hire_date, salary from employees)
loop
dbms_output.put_line('employee id: ' || i.employee_id);
dbms_output.put_line('name: ' || i.first_name);
dbms_output.put_line('job title: ' || i.job_id);
dbms_output.put_line('hire date: ' || to_char(i.hire_date, 'dd-mon-yyyy'));
dbms_output.put_line('salary: ' || i.salary);
dbms_output.put_line(' ------------------------------ ');
end loop;
end;
```

```
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.

```
begin
for i in (select e.employee_id, e.first_name, e.job_id from employees e)
loop
dbms_output.put_line('employee id: ' || i.employee_id);
dbms_output.put_line('name: ' || i.first_name);
dbms_output.put_line('department name: ' || i.job_id);
dbms_output.put_line(' ------------------------------ ');
end loop;
end;
```

```
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.

```
Begin
for i in (select job_id,job_title,min_salary from jobs)
loop
dbms_output.put_line('job id: ' || i.job_id);
dbms_output.put_line('job title: ' || i.job_title);
dbms_output.put_line('minimum salary: ' || i.min_salary);
dbms_output.put_line(' ------------------------------ ');
end loop;
end;
```

```
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.

```
Begin
for i in (select employee_id,employee_name,start_date from job_history)
loop
dbms_output.put_line('employee id: ' || i.employee_id);
dbms_output.put_line('name: ' || i.employee_name);
dbms_output.put_line('start date: ' ||to_char(i.start_date, 'dd-mon-yyyy'));
dbms_output.put_line(' ------------------------------ ');
end loop;
end;
```

```
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 ofall employees.

```
Begin
for i in (select employee_id,employee_name,end_date from job_history)
loop
dbms_output.put_line('employee id: ' || i.employee_id);
dbms_output.put_line('name: ' || i.employee_name);
dbms_output.put_line('end date: ' ||to_char(i.end_date, 'dd-mon-yyyy'));
dbms_output.put_line(' ------------------------------ ');
end loop;
end;
```

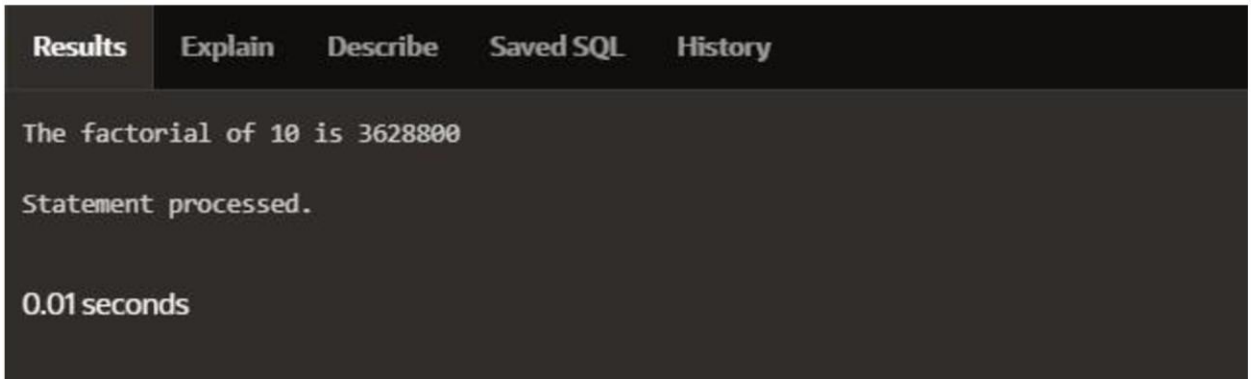
```
employee id: 201
name: James
end date: 10-oct-2015
--------------------------------
employee id: 202
name: King
end date: 15-sep-2016
--------------------------------
employee id: 203
name: Smith
end date: 20-mar-2017
--------------------------------
employee id: 204
name: Steve
end date: 05-apr-2018
--------------------------------
employee id: 205
name: Robert
end date: 12-may-2019
--------------------------------
```

Ex.No.: 12		PL SQL PROGRAMS
Date:	19/09/2024	

Program 1

FACTORIAL OF A NUMBER USING FUNCTION

```
DECLARE
   n NUMBER := 10;
   result NUMBER;

   FUNCTION itfact(num NUMBER) RETURN NUMBER IS
      fact NUMBER := 1;
   BEGIN
      FOR i IN 1..num LOOP
         fact := fact * i;
      END LOOP;
      RETURN fact;
   END;

BEGIN
   result := itfact(n);
   DBMS_OUTPUT.PUT_LINE('The factorial of ' || n || ' is ' || result);
END;
```



Program 2

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

```
CREATE OR REPLACE PROCEDURE book_info(
   p_book_id IN NUMBER,
   p_author OUT VARCHAR2,
   p_title OUT VARCHAR2,
   p_published_date OUT DATE
) AS
BEGIN
   SELECT author, title, published_date
   INTO p_author, p_title, p_published_date
   FROM books
   WHERE book_id = p_book_id;

EXCEPTION
   WHEN NO_DATA_FOUND THEN
      p_author := NULL;
      p_title := NULL;
      p_published_date := NULL;
   WHEN OTHERS THEN
      RAISE;
END book_info;

DECLARE
   v_author VARCHAR2(100);
   v_title VARCHAR2(100);
   v_published_date DATE;
   v_book_id NUMBER := 1;
BEGIN
   book_info(v_book_id, v_author, v_title, v_published_date);

   IF v_author IS NOT NULL THEN
      DBMS_OUTPUT.PUT_LINE('Book ID: ' || v_book_id);
      DBMS_OUTPUT.PUT_LINE('Author: ' || v_author);
      DBMS_OUTPUT.PUT_LINE('Title: ' || v_title);
      DBMS_OUTPUT.PUT_LINE('Published Date: ' || TO_CHAR(v_published_date, 'YYYY-
MM-DD'));
    ELSE
      DBMS_OUTPUT.PUT_LINE('No book found with ID: ' || v_book_id);
   END IF;
END;
```

```
Book ID: 1
Author: William Shaespeare
Title: Hamlet
Published Date: 1590-12-12

Statement processed.
```

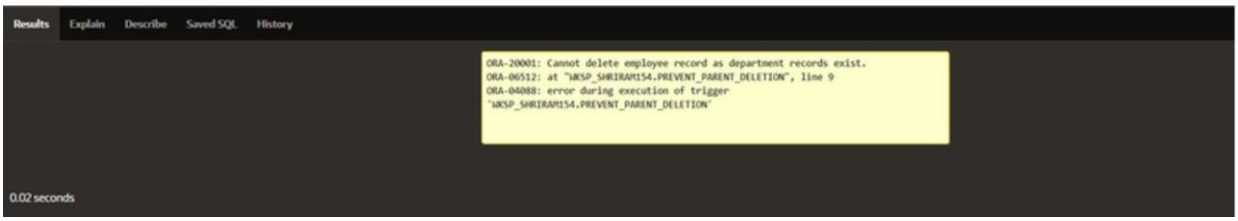
0.02 seconds

Ex.No.: 13		WORKING WITH TRIGGERS
Date:	20/09/2024	

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.

```
CREATE OR REPLACE TRIGGER prevent_parent_deletion
BEFORE DELETE ON employees
FOR EACH ROW
DECLARE
   pl_dept_count NUMBER;
BEGIN
   SELECT COUNT(*)
   INTO pl_dept_count
   FROM department
   WHERE dept_id = :OLD.employee_id;
   IF pl_dept_count > 0 THEN
      RAISE_APPLICATION_ERROR(-20001, 'Cannot delete employee record as
department records exist.');
   END IF;
END;
```

```
DELETE   FROM   employees
WHERE employee_id = 70;
```

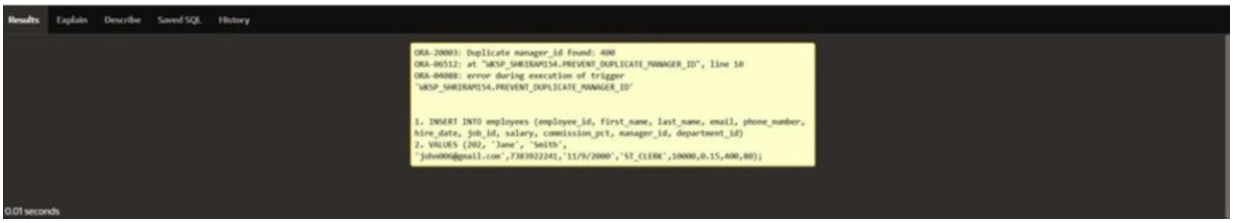


Program 2

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.

```
CREATE OR REPLACE TRIGGER prevent_duplicate_manager_id
BEFORE INSERT OR UPDATE ON employees
FOR EACH ROW
DECLARE
   pl_count NUMBER;
BEGIN
   SELECT COUNT(*)
   INTO pl_count
   FROM employees
   WHERE manager_id = :NEW.manager_id
   AND employee_id != :NEW.employee_id;
   IF pl_count > 0 THEN
      RAISE_APPLICATION_ERROR(-20003, 'Duplicate manager_id found: ' ||
:NEW.manager_id);
   END IF;
END;
```

```
INSERT INTO employees (employee_id, first_name, last_name, email, phone_number,
hire_date, job_id, salary, commission_pct, manager_id, department_id)
VALUES (202, 'Jane', 'Smith',
'john006@gmail.com',7383922241,'11/9/2000','ST_CLERK',10000,0.15,400,80);
```

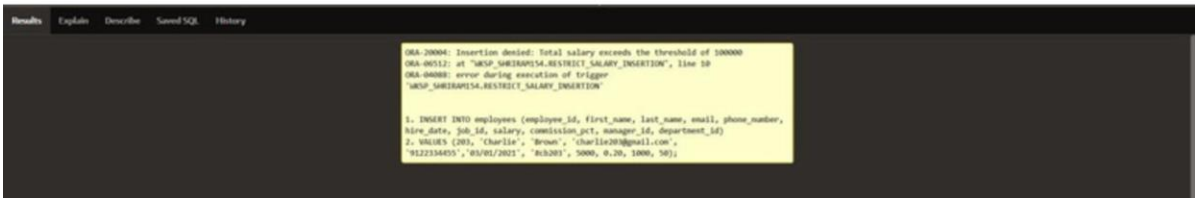


Program 3

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.

```
CREATE OR REPLACE TRIGGER restrict_salary_insertion
BEFORE INSERT ON employees
FOR EACH ROW
DECLARE
   total_salary NUMBER;
   threshold NUMBER := 100000;
BEGIN

   SELECT SUM(salary)
   INTO total_salary
   FROM employees;
   IF (total_salary + :NEW.salary) > threshold THEN
      RAISE_APPLICATION_ERROR(-20004, 'Insertion denied: Total salary exceeds the
threshold of ' || threshold);
   END IF;
END;
```

```
INSERT INTO employees (employee_id, first_name, last_name, email, phone_number,
hire_date, job_id, salary, commission_pct, manager_id, department_id)
VALUES (203, 'Charlie', 'Brown', 'charlie203@gmail.com', '9122334455','03/01/2021',
'#cb203', 5000, 0.20, 1000, 50);
```



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.

```
CREATE OR REPLACE TRIGGER audit_changes
AFTER UPDATE OF salary, job_id ON employees
FOR EACH ROW
BEGIN
   IF :OLD.salary != :NEW.salary OR :OLD.job_id != :NEW.job_id THEN
      INSERT INTO employee_audit (
         employee_id,
         old_salary,
         new_salary,
         old_job_title,
         new_job_title,
         change_timestamp,
         changed_by
      ) VALUES (
         :OLD.employee_id,
         :OLD.salary,
         :NEW.salary,
         :OLD.job_id,
         :NEW.job_id,
         SYSTIMESTAMP,
         USER
      );
   END IF;
END;


UPDATE employees
SET salary = 55000, job_id = 'ST_CLERK'
WHERE employee_id = 176;


SELECT * FROM employee_audit;
```

AUDIT_ID	EMPLOYEE_ID	OLD_SALARY	NEW_SALARY	OLD_JOB_ID	NEW_JOB_ID	CHANGE_TIMESTAMP	CHANGED_BY
1	20	50000	55000	manager	manager	15-OCT-24 10.00.00.000000 AM	admin
2	122	60000	65000	Manager	Manager	15-OCT-24 10.15.00.000000 AM	admin
5	27	45000	47000	Analyst	Senior Analyst	15-OCT-24 10.30.00.000000 AM	user1
22	176	7500	55000	#ce005	ST_CLERK	16-OCT-24 04.25.06.252580 PM	APEX_PUBLIC_USER
3	9	70000	75000	Senior Developer	Lead Developer	15-OCT-24 10.45.00.000000 AM	user2
4	4	80000	85000	Team Lead	Project Manager	15-OCT-24 11.00.00.000000 AM	admin

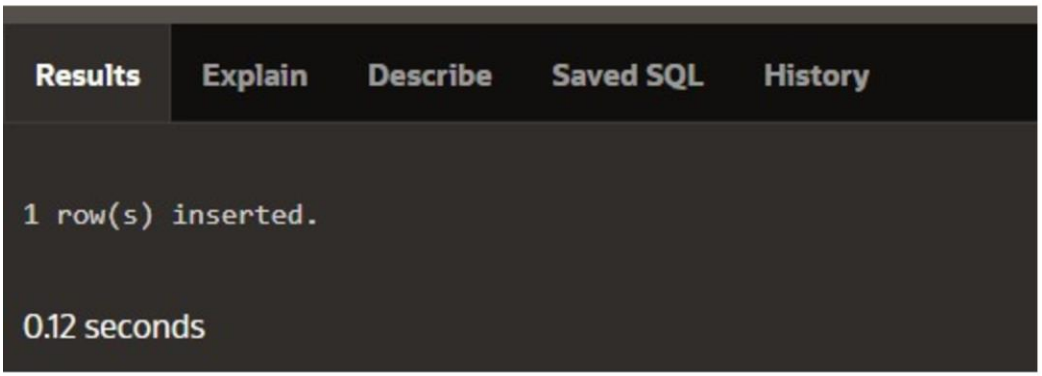
6 rows returned in 0.00 seconds Download

PROGRAM 5

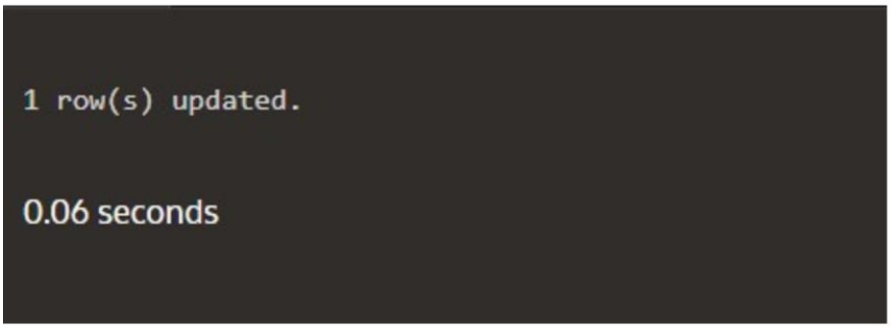
Write a code in PL/SQL to implement a trigger that records user activity (inserts, updates, deletes) in an audit log for a given set of tables.

```
CREATE OR REPLACE TRIGGER trg_audit_employees
AFTER INSERT OR UPDATE OR DELETE ON employees
FOR EACH ROW
DECLARE
   v_old_values CLOB;
   v_new_values CLOB;
BEGIN
   IF INSERTING THEN
      v_old_values := NULL;
      v_new_values := 'employee_id: ' || :NEW.employee_id || ', ' ||
                      'first_name: ' || :NEW.first_name || ', ' ||
                      'salary: ' || :NEW.salary;

      INSERT INTO audit_log (action, table_name, record_id, changed_by, new_values)
      VALUES ('INSERT', 'employees', :NEW.employee_id, USER, v_new_values);

   ELSIF UPDATING THEN
      v_old_values := 'employee_id: ' || :OLD.employee_id || ', ' ||
                      'first_name: ' || :OLD.first_name || ', ' ||
                      'salary: ' || :OLD.salary;
      v_new_values := 'employee_id: ' || :NEW.employee_id || ', ' ||
                      'first_name: ' || :NEW.first_name || ', ' ||
                      'salary: ' || :NEW.salary;

      INSERT INTO audit_log (action, table_name, record_id, changed_by, old_values,
new_values)
      VALUES ('UPDATE', 'employees', :NEW.employee_id, USER, v_old_values,
v_new_values);

   ELSIF DELETING THEN
      v_old_values := 'employee_id: ' || :OLD.employee_id || ', ' ||
                      'first_name: ' || :OLD.first_name || ', ' ||
                      'salary: ' || :OLD.salary;
      v_new_values := NULL;

      INSERT INTO audit_log (action, table_name, record_id, changed_by, old_values)
      VALUES ('DELETE', 'employees', :OLD.employee_id, USER, v_old_values);
   END IF;
END trg_audit_employees;
```

```
INSERT INTO employees (employee_id, first_name, salary)
VALUES (3, 'Ball', 50000);
```




```
UPDATE employees
SET salary = 55000
WHERE employee_id = 3;
```




```
DELETE FROM employees
WHERE employee_id = 3;
```

```
SELECT * FROM audit_log;
```

AUDIT_ID	ACTION	TABLE_NAME	RECORD_ID	CHANGED_BY	CHANGE_TIMESTAMP	OLD_VALUES	NEW_VALUES
1	INSERT	employees	3	APEX_PUBLIC_USER	16-OCT-24 04.39.17.957508 PM	-	employee_id: 3, first_name: Ball, salary: 50000
3	DELETE	employees	3	APEX_PUBLIC_USER	16-OCT-24 04.41.49.077471 PM	employee_id: 3, first_name: Ball, salary: 55000	-
2	UPDATE	employees	3	APEX_PUBLIC_USER	16-OCT-24 04.40.03.395035 PM	employee_id: 3, first_name: Ball, salary: 50000	employee_id: 3, first_name: Ball, salary: 55000

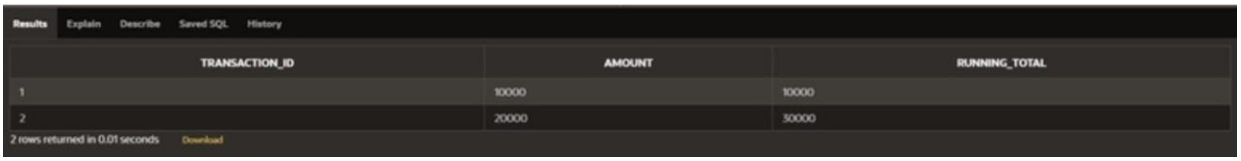
3 rows returned in 0.00 seconds Download

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.

```
CREATE TABLE transactions (
   transaction_id NUMBER PRIMARY KEY,
   amount NUMBER,
   running_total NUMBER
);

CREATE OR REPLACE TRIGGER update_running_total
FOR INSERT ON transactions
COMPOUND TRIGGER

   TYPE amount_array IS TABLE OF NUMBER INDEX BY PLS_INTEGER;
   new_amounts amount_array;

   BEFORE EACH ROW IS
   BEGIN
      new_amounts(:NEW.transaction_id) := :NEW.amount;
   END BEFORE EACH ROW;

   AFTER STATEMENT IS
   BEGIN
      DECLARE
         v_total NUMBER;
      BEGIN
         SELECT NVL(MAX(running_total), 0)
         INTO v_total
         FROM transactions;

         FOR i IN new_amounts.FIRST .. new_amounts.LAST LOOP
            v_total := v_total + new_amounts(i);
            UPDATE transactions
            SET running_total = v_total
            WHERE transaction_id = i;
         END LOOP;
      END;
   END AFTER STATEMENT;

END update_running_total;


INSERT INTO transactions (transaction_id, amount)
```

```
VALUES (1, 10000);


INSERT INTO transactions (transaction_id, amount)
VALUES (2, 20000);
```



Results | Explain | Describe | Saved SQL | History

TRANSACTION_ID	AMOUNT	RUNNING_TOTAL
1	10000	10000
2	20000	30000

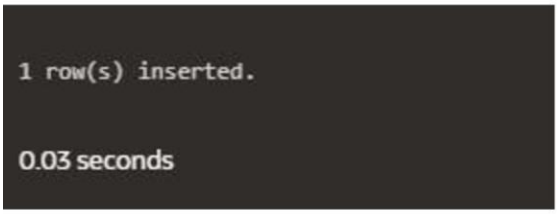
2 rows returned in 0.01 seconds Download

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.

```
CREATE TABLE inventory (
   item_id NUMBER PRIMARY KEY,
   item_name VARCHAR2(100),
   stock_level NUMBER
);


CREATE TABLE orders (
   order_id NUMBER PRIMARY KEY,
   item_id NUMBER,
   quantity NUMBER,
   order_status VARCHAR2(20),
   CONSTRAINT fk_item FOREIGN KEY (item_id) REFERENCES inventory(item_id)
);




CREATE OR REPLACE TRIGGER validate_stock_before_order
BEFORE INSERT ON orders
FOR EACH ROW
DECLARE
   v_stock_level NUMBER;
   v_pending_orders NUMBER;
BEGIN
   SELECT stock_level
   INTO v_stock_level
   FROM inventory
   WHERE item_id = :NEW.item_id;
   SELECT NVL(SUM(quantity), 0)
   INTO v_pending_orders
   FROM orders
   WHERE item_id = :NEW.item_id
     AND order_status = 'Pending';
   IF (:NEW.quantity + v_pending_orders) > v_stock_level THEN
      RAISE_APPLICATION_ERROR(-20001, 'Insufficient stock for item: ' || :NEW.item_id);
   END IF;
END;
```

```
INSERT INTO orders (order_id, item_id, quantity, order_status)
VALUES (1, 101, 5, 'Pending');
```




```
INSERT INTO orders (order_id, item_id, quantity, order_status)
VALUES (2, 103, 20, 'Pending');
```

ITEM_ID	ITEM_NAME	STOCK_LEVEL
101	hp_laptop	10
102	keyboard	20
103	mouse	15

3 rows returned in 0.01 seconds Download

ORDER_ID	ITEM_ID	QUANTITY	ORDER_STATUS
1	101	5	Pending

1 rows returned in 0.01 seconds Download

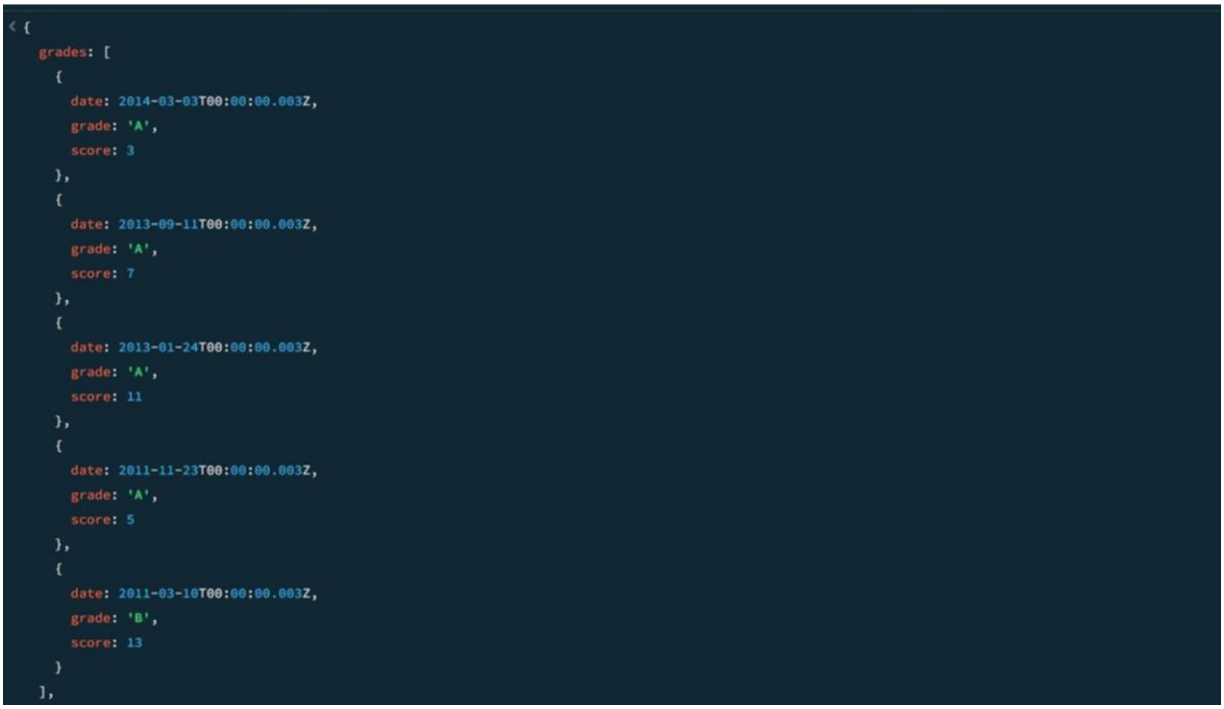
Ex.No.: 14		MONGO DB
Date:	26/09/2024	

1. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which prepared dish except 'American' and 'Chinees' or restaurant's name begins with 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
 }
);
```

2. 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 of survey dates..

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



3. 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
 }
);
```

4. 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
 }
);
```

5. 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: 2011-11-23T00:00:00.008Z,
      grade: 'B',
      score: 15
    },
    {
      date: 2011-03-10T00:00:00.008Z,
```

```
    grade: 'A',
    score: 6
   }
  ],
  name: 'Bistro Belle',
  restaurant_id: 30075453
}
```

6. 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: {
    building: 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: {
    building: 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
}
```

7. Write a MongoDB query to arrange the name of the cuisine in ascending order and for that same cuisine borough should be in descending order.

```
db.restaurants.find().sort({ cuisine: 1, borough: -1 });
```

SAMPLE OUTPUT:-

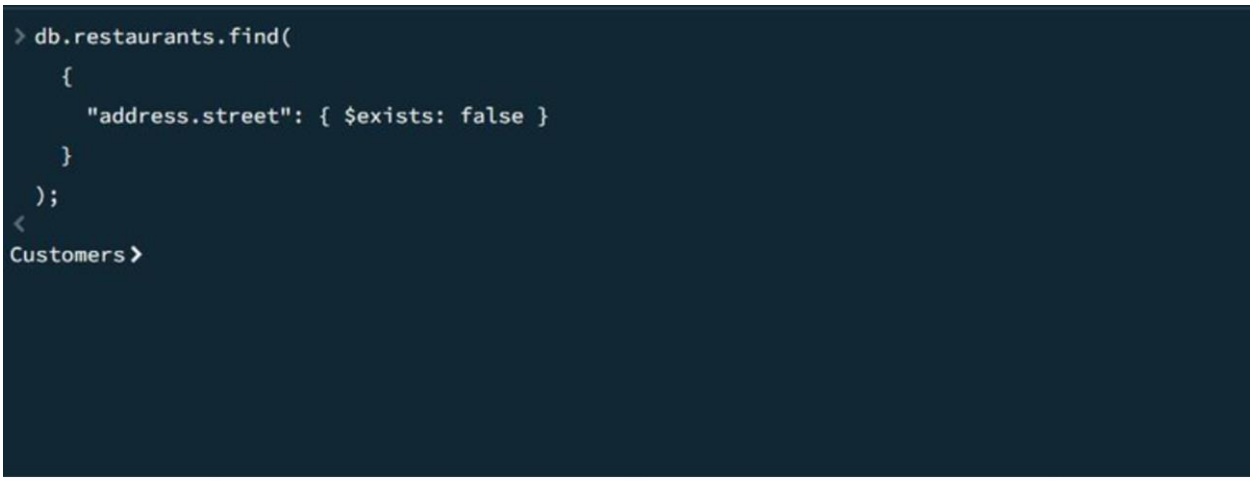
```
{
  _id: ObjectId('671b5d549d3d63480e0a64e9'),
  address: {
    building: 2233,
    coord: [
      -73.858177,
      40.849447
    ],
    street: '5th Avenue',
    zipcode: 10467
  },
  borough: 'Bronx',
  cuisine: 'American',
```

```
  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: {
    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
}
```

8. Write a MongoDB query to know whether all the addresses contains the street or not.

```
db.restaurants.find(
  {
    "address.street": { $exists: false }
  }
);
```



9. 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('671b5d549d3d63480e0a64e5'),
 address: {
  building: 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
}
```

10. Write a MongoDB query which will select the restaurant Id, name and grades for those
restaurants which returns 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
}
```

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

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

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

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

13. 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('671b5d549d3d63480e0a64e6'),
  address: {
```

```
    building: 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
}
```

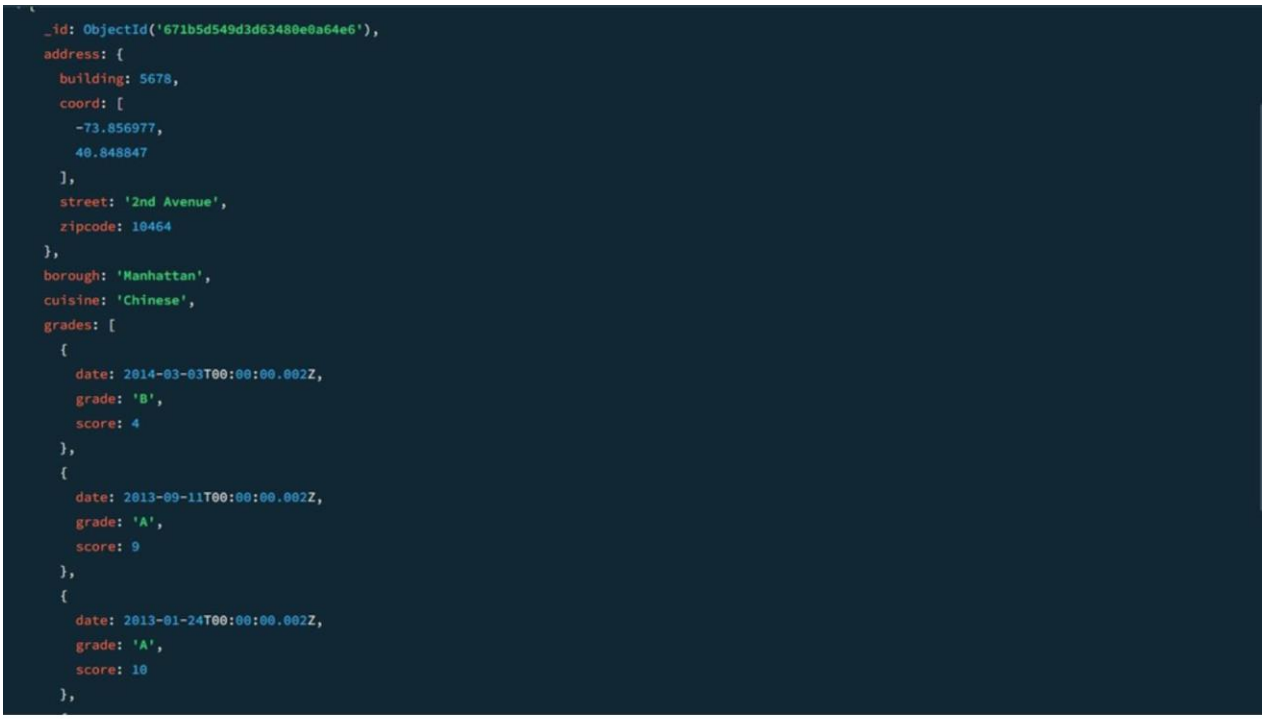
14. Write a MongoDB query to find the restaurants that have at least one grade with a score of less than 5 and that are located in the borough of Manhattan.

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


```
_id: ObjectId('671b5d549d3d63480e0a64e6'),
address: {
  building: 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
  },
  {
```

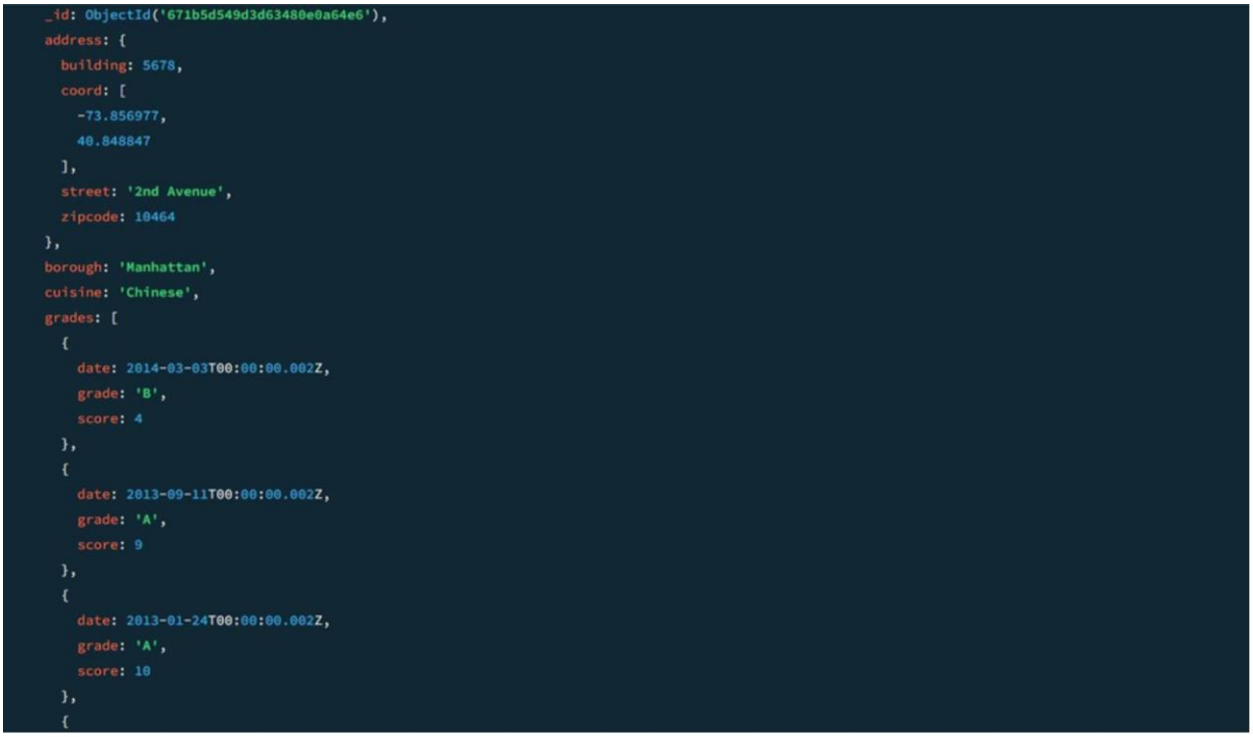

15. Write a MongoDB query to find the restaurants that have at least one grade with a score of less than 5 and that are located in the borough of Manhattan or Brooklyn.

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



16. 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" }
  }
);
```



17. 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"] }
 }
);
```

18. 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('671b5c5f9d3d63480e0a64e4'),
  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
}
```

19. 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 } }
   ]
  }
 }
);
```

20. 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 } }
    ]
   }
  }
);
```

21. 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" }
 }
);
```

22. 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"] }
 }
);
```

23. 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 OUTPUT:-

```
{
 _id: ObjectId('671b5d549d3d63480e0a64e9'),
 address: {
  building: 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('671b5dab56ec9972ca8f5daf'),
 address: {
  building: 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
}
```

MOVIES COLLECTION

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

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

2. 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: [
    'Sci-Fi',
    'Drama'
  ],
  runtime: 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.',
  languages: [
```

```
  'English'
],
released: 2015-10-02T00:00:00.000Z,
directors: [
  '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
},
countries: [
  'USA'
],
type: 'movie',
tomatoes: {
viewer: {
    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
  }
}
```

3. Find all movies with full information from the 'movies' collection that have "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.',
  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/MV5BMTU3NjE5NzYtYTYyNS00MDVmLWIwYjg
tMmYwYWIxZDYyNzU2XkEyXkFqcGdeQXVyNzQzNzQxNzI@._V1_SY1
000_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.",
  languages: [
    'English'
  ],
  released: 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: 'movie',
  tomatoes: {
  viewer: {
      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
 }
}
```

4. 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" });
```

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

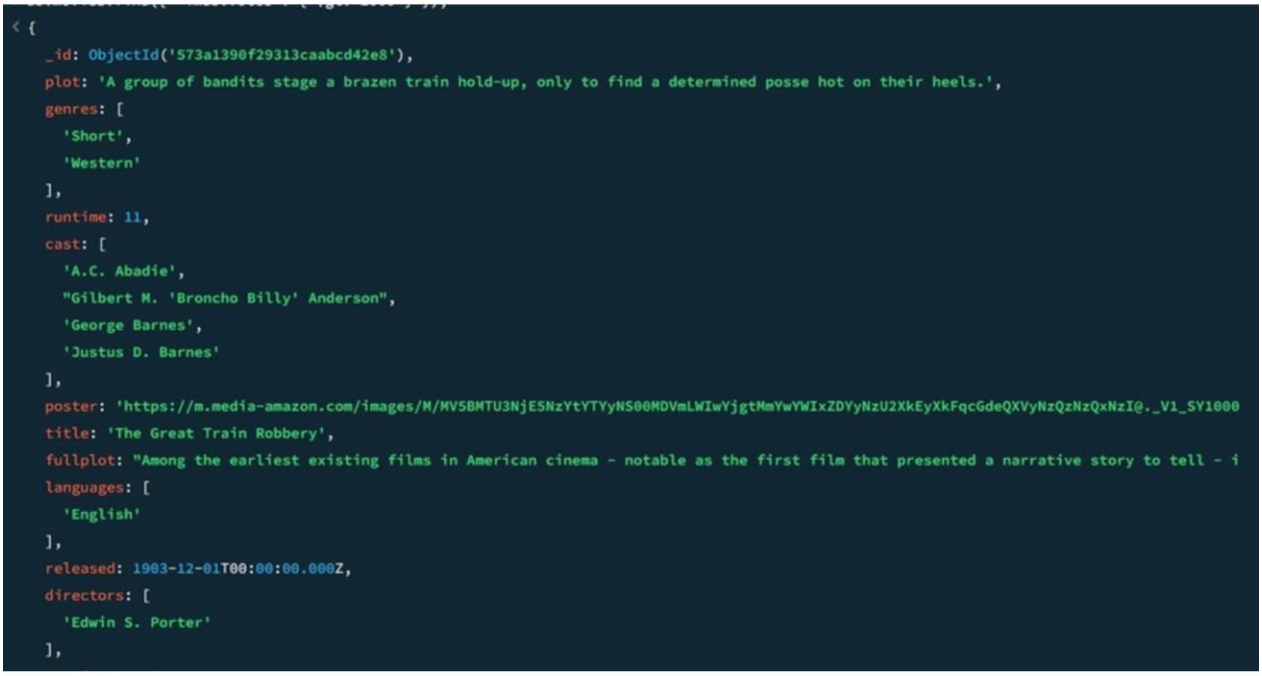
```
db.movies.find({ countries: "USA" });
```

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

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

8. 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 } });
```

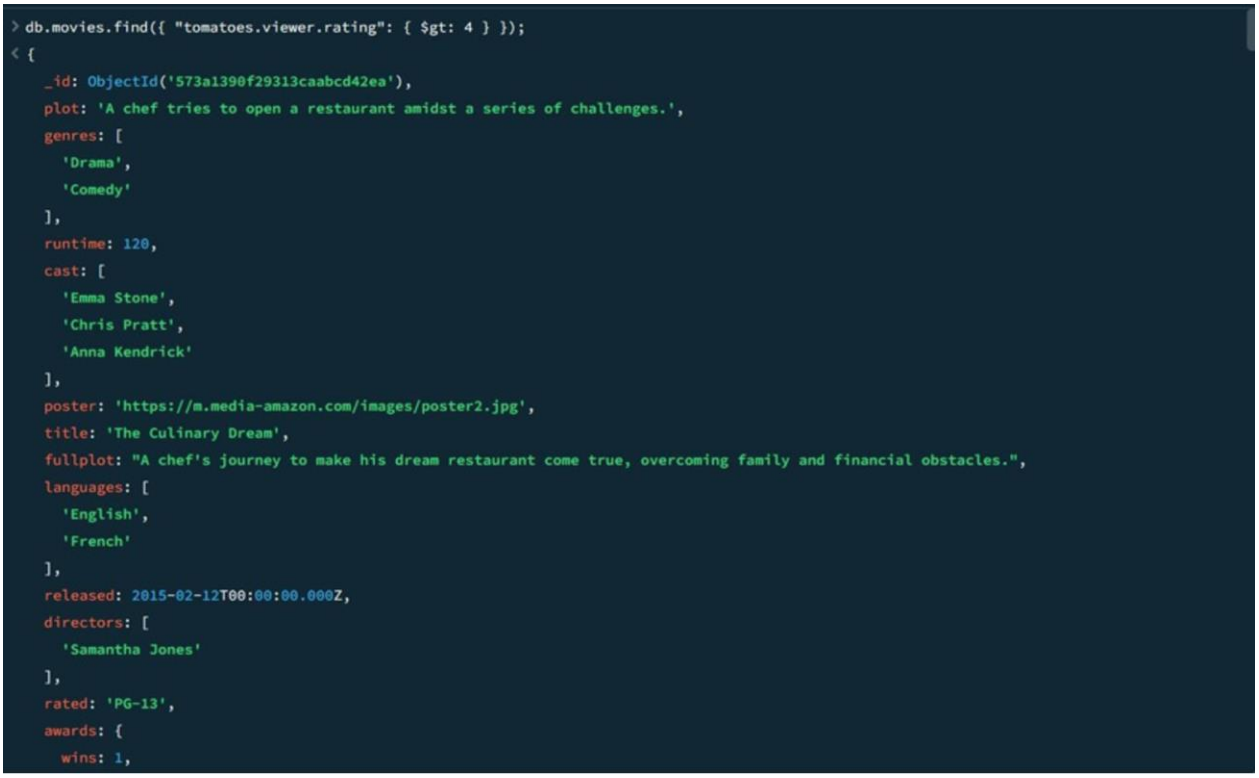


9. Retrieve all movies from the 'movies' collection that have complete information and have an IMDb rating higher than 7.

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

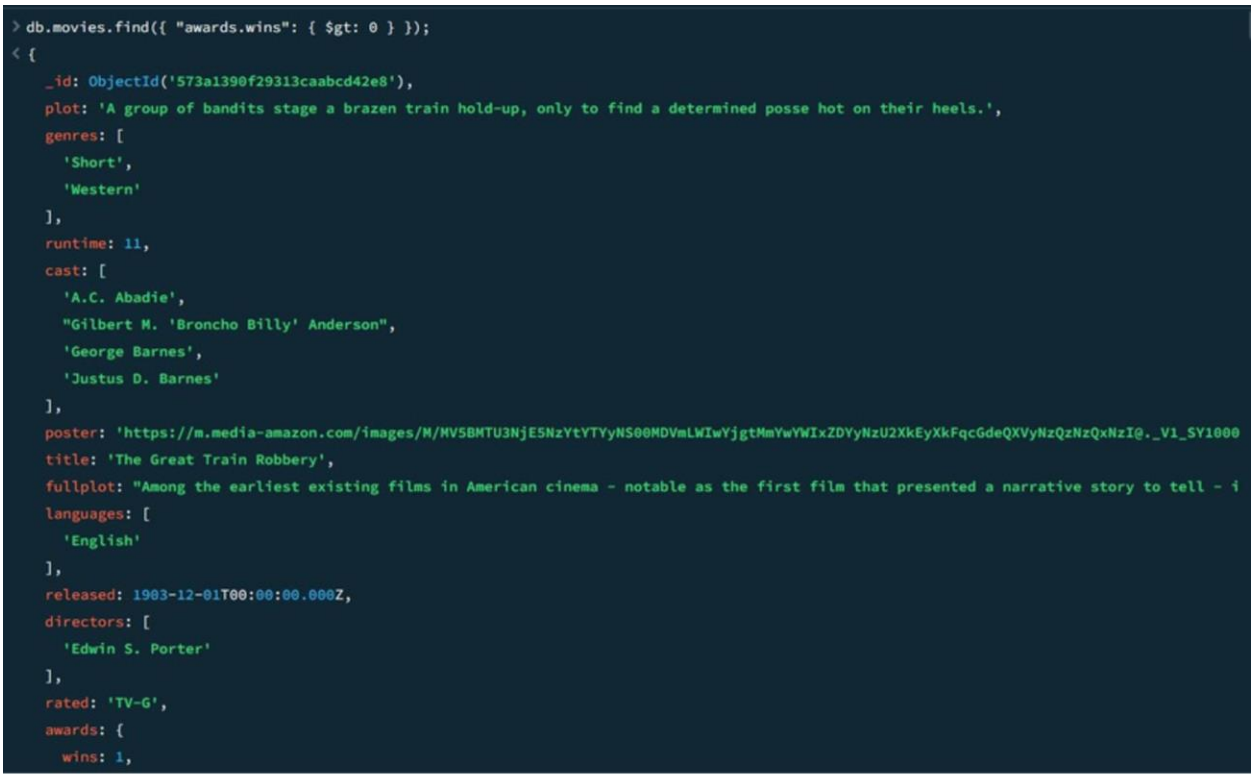
10. Retrieve all movies from the 'movies' collection that have complete information and have a viewer rating higher than 4 on Tomatoes.

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



11. Retrieve all movies from the 'movies' collection that have received an award.

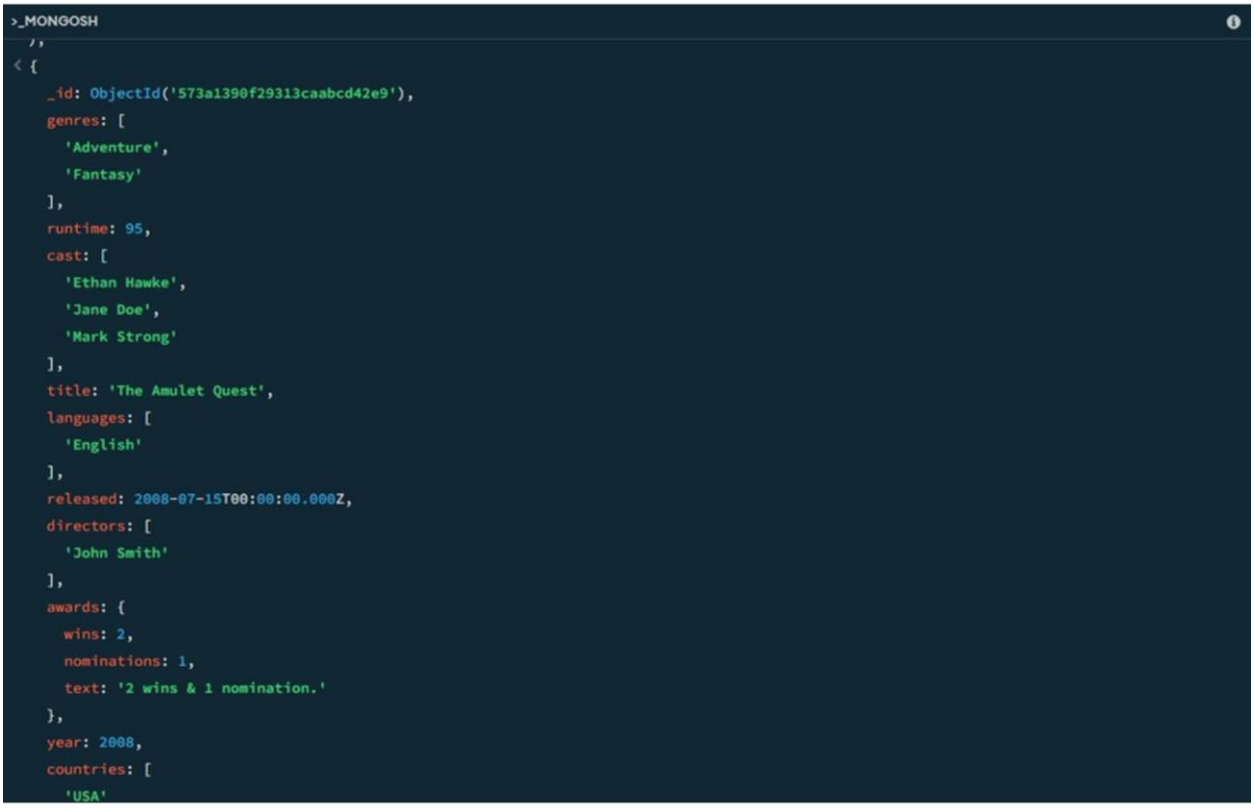
db.movies.find({ "awards.wins": { $gt: 0 } });



12. 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
```

```
    }
);
```



13. 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
    }
);
```

14. Retrieve all movies with title, languages, released, directors, writers, countries from the 'movies' collection in MongoDB that 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
    }
);
```

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

```
db.movies.find(
    { title: { $regex: /scene/i } },
    {
        title: 1,
        languages: 1,
```

```
        released: 1,
        directors: 1,
        writers: 1,
        countries: 1
    }
);
```

Ex.No.: 15		OTHER DATABASE OBJECTS
Date:	27/09/2024	

1) 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.

```
CREATE SEQUENCE DEPT_ID_SEQ
START WITH 200
INCREMENT BY 10
MAXVALUE 1000
NOCACHE
NOCYCLE;
```

2. 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,
      LAST_NUMBER
FROM USER_SEQUENCES;
```

Results | Explain | Describe | Saved SQL | History

SEQUENCE_NAME	MAX_VALUE	INCREMENT_BY	LAST_NUMBER
DEPT_ID_SEQ	1000	10	200
ISEQ$$_323104505	9999999999999999999999999999	1	41
ISEQ$$_323114704	9999999999999999999999999999	1	21

3 rows returned in 0.03 seconds Download

3 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 DEPT (DEPT_ID, DEPT_NAME)
VALUES (DEPT_ID_SEQ.NEXTVAL, 'Education');
```

```
INSERT INTO DEPT (DEPT_ID, DEPT_NAME)
```

```
VALUES (DEPT_ID_SEQ.NEXTVAL, 'Administration');

SELECT * FROM DEPT
WHERE DEPT_NAME IN ('Education', 'Administration');
```

DEPT_ID	DEPT_NAME
210	Administration
200	Education

2 rows returned in 0.04 seconds Download

4. Create a non unique index on the foreign key column (DEPARTMENT_ID) in the EMPLOYEES table.

```
CREATE INDEX employees_department_id_idx
ON EMPLOYEES (DEPARTMENT_ID);
```

5. 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';
```

INDEX_NAME	UNIQUENESS
EMPLOYEES_DEPARTMENT_ID_IDX	NONUNIQUE
SYS_C0016680725	UNIQUE

2 rows returned in 0.05 seconds Download

Ex.No.: 16		CONTROLLING USER ACCESS
Date:	03/10/2024	

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

The privilege a user should be given to log on to the Oracle Server is the CREATE SESSION privilege.

Type of Privilege: This is a system privilege.

GRANT CREATE SESSION TO username;

2. What privilege should a user be given to create tables?

the user needs the CREATE TABLE privilege.
The CREATE TABLE privilege allows the user to create new tables in their own schema.

GRANT CREATE TABLE TO username;

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

When you create a table, only you as the table owner (or a user with the ADMIN OPTION or GRANT ANY PRIVILEGE system privilege) can grant privileges on your table to other users.

GRANT SELECT ON your_table TO other_user;

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

As a DBA, to simplify the process of granting the same system privileges to multiple users, you should use roles.
CREATE ROLE my_role;

GRANT CREATE SESSION TO my_role;
GRANT CREATE TABLE TO my_role;

GRANT my_role TO user1;
GRANT my_role TO user2;

5. What command do you use to change your password?

ALTER USER username IDENTIFIED BY new_password;

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

Grant Access to Your **DEPARTMENTS** Table

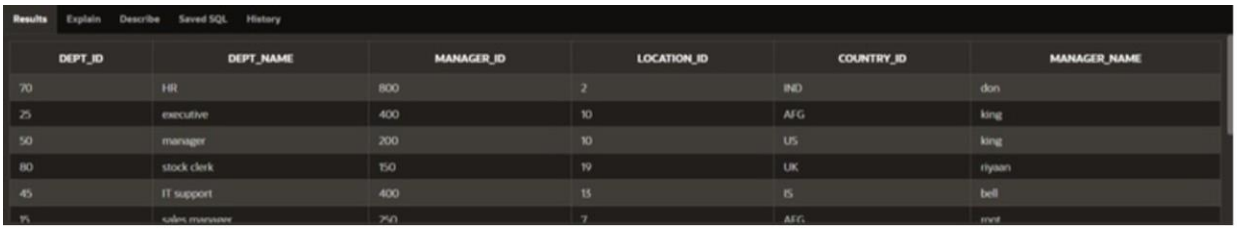
GRANT SELECT ON your_username.DEPARTMENTS TO other_user;

Grant Query Access to Other User's **DEPARTMENTS** Table

GRANT SELECT ON other_user.DEPARTMENTS TO your_username;

7. Query all the rows in your DEPARTMENTS table.

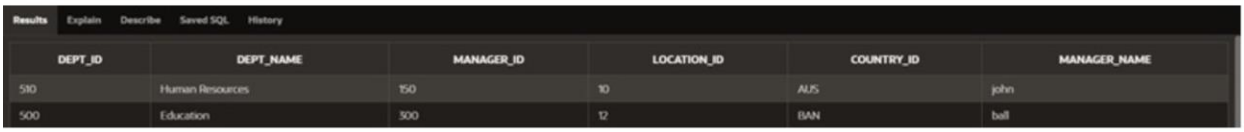
SELECT * FROM DEPARTMENT;



DEPT_ID	DEPT_NAME	MANAGER_ID	LOCATION_ID	COUNTRY_ID	MANAGER_NAME
70	HR	800	2	IND	don
25	executive	400	10	AFG	king
50	manager	200	10	US	king
80	stock clerk	150	19	UK	riyan
45	IT support	400	13	IS	bell
15	sales manager	250	7	AFG	rayt

8. 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.

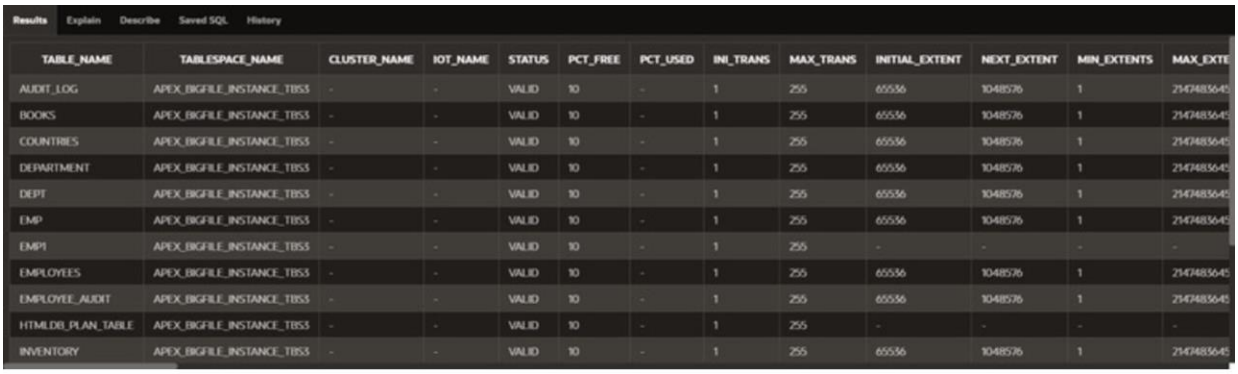
```
INSERT INTO DEPARTMENT(dept_id,
DEPT_NAME,manager_id,location_id,country_id,manager_name)
VALUES (500, 'Education',300,12,'BAN','ball');

INSERT INTO DEPARTMENT(dept_id,
DEPT_NAME,manager_id,location_id,country_id,manager_name)
VALUES (510, 'Human Resources',150,10,'AUS','john');

SELECT * FROM DEPARTMENT;
```



DEPT_ID	DEPT_NAME	MANAGER_ID	LOCATION_ID	COUNTRY_ID	MANAGER_NAME
510	Human Resources	150	10	AUS	john
500	Education	300	12	BAN	ball

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

```
SELECT * FROM USER_TABLES;
```



Results | Explain | Describe | Saved SQL | History

TABLE_NAME	TABLESPACE_NAME	CLUSTER_NAME	IOT_NAME	STATUS	PCT_FREE	PCT_USED	INI_TRANS	MAX_TRANS	INITIAL_EXTENT	NEXT_EXTENT	MIN_EXTENTS	MAX_EXTE
AUDIT_LOG	APEX_BIGFILE_INSTANCE_TBS3	-	-	VALID	10	-	1	255	65536	1048576	1	2147483645
BOOKS	APEX_BIGFILE_INSTANCE_TBS3	-	-	VALID	10	-	1	255	65536	1048576	1	2147483645
COUNTRIES	APEX_BIGFILE_INSTANCE_TBS3	-	-	VALID	10	-	1	255	65536	1048576	1	2147483645
DEPARTMENT	APEX_BIGFILE_INSTANCE_TBS3	-	-	VALID	10	-	1	255	65536	1048576	1	2147483645
DEPT	APEX_BIGFILE_INSTANCE_TBS3	-	-	VALID	10	-	1	255	65536	1048576	1	2147483645
EMP	APEX_BIGFILE_INSTANCE_TBS3	-	-	VALID	10	-	1	255	65536	1048576	1	2147483645
EMP1	APEX_BIGFILE_INSTANCE_TBS3	-	-	VALID	10	-	1	255	-	-	-	-
EMPLOYEES	APEX_BIGFILE_INSTANCE_TBS3	-	-	VALID	10	-	1	255	65536	1048576	1	2147483645
EMPLOYEE_AUDIT	APEX_BIGFILE_INSTANCE_TBS3	-	-	VALID	10	-	1	255	65536	1048576	1	2147483645
HTMLDB_PLAN_TABLE	APEX_BIGFILE_INSTANCE_TBS3	-	-	VALID	10	-	1	255	-	-	-	-
INVENTORY	APEX_BIGFILE_INSTANCE_TBS3	-	-	VALID	10	-	1	255	65536	1048576	1	2147483645

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

```
REVOKE SELECT ON team1_user.DEPARTMENTS FROM other_user;
```

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

```
DELETE FROM DEPARTMENT
WHERE DEPT_ID IN (500, 510);
```