

### 1) Oracle Sequences:

#### 1. Customer Table and Sequence

##### a) Create the **customer** Table:

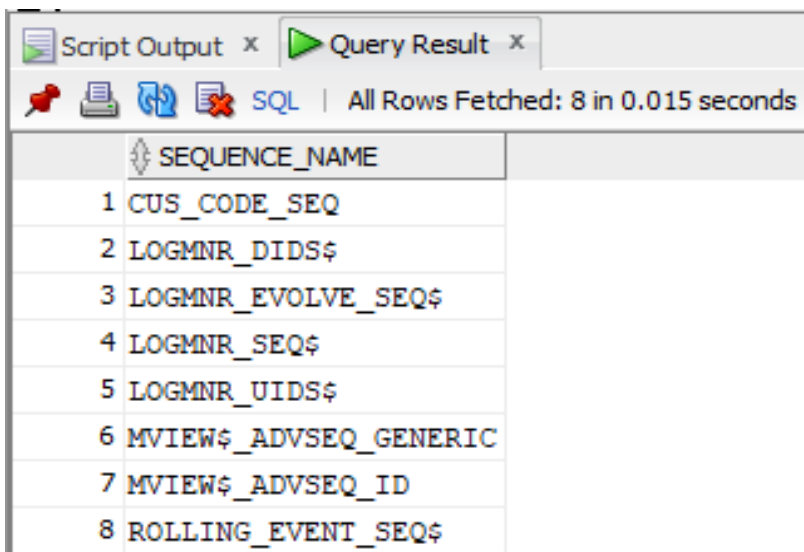
```
CREATE TABLE customers (  
    cus_code INTEGER,  
    cus_lname VARCHAR2(10),  
    cus_fname VARCHAR2(10),  
    cus_initial VARCHAR2(1),  
    cus_areacode INTEGER,  
    cus_phone INTEGER,  
    cus_balance NUMBER(10, 2),  
    PRIMARY KEY (cus_code)  
);
```

##### b) Create Sequence for **cus\_code**:

```
CREATE SEQUENCE cus_code_seq  
START WITH 18887  
INCREMENT BY 1  
NOCACHE;
```

##### c) Display User Sequences:

```
SELECT sequence_name FROM user_sequences;
```



The screenshot shows the Oracle SQL Developer interface. The 'Query Result' tab is active, displaying the results of the query 'SELECT sequence\_name FROM user\_sequences;'. The results are shown in a table with one column, 'SEQUENCE\_NAME', and eight rows. The first row is 'CUS\_CODE\_SEQ'. The other rows are 'LOGMNR\_DIDS\$', 'LOGMNR\_EVOLVE\_SEQ\$', 'LOGMNR\_SEQ\$', 'LOGMNR\_UIDS\$', 'MVIEW\$\_ADVSEQ\_GENERIC', 'MVIEW\$\_ADVSEQ\_ID', and 'ROLLING\_EVENT\_SEQ\$'. The status bar at the bottom indicates 'All Rows Fetched: 8 in 0.015 seconds'.

SEQUENCE_NAME
1 CUS_CODE_SEQ
2 LOGMNR_DIDS\$
3 LOGMNR_EVOLVE_SEQ\$
4 LOGMNR_SEQ\$
5 LOGMNR_UIDS\$
6 MVIEW\$_ADVSEQ_GENERIC
7 MVIEW\$_ADVSEQ_ID
8 ROLLING_EVENT_SEQ\$

##### d) Insert Values into **customer** Using the Created Sequence:

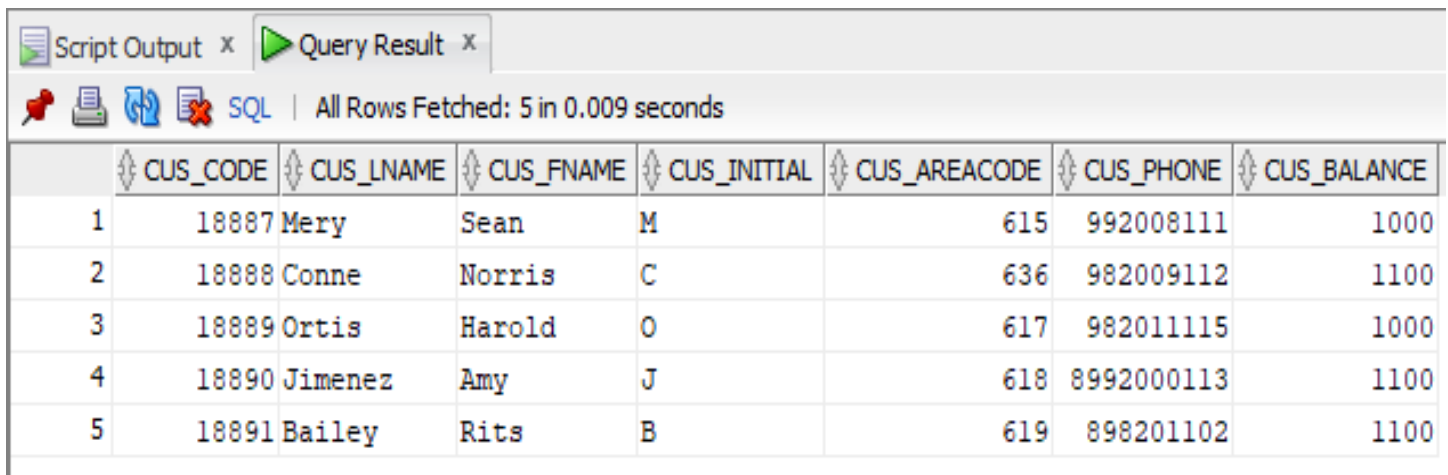
```

INSERT INTO customers VALUES (cus_code_seq.NEXTVAL, 'Mery', 'Sean', 'M', 615, 992008111,
1000.00);
INSERT INTO customers VALUES (cus_code_seq.NEXTVAL, 'Conne', 'Norris', 'C', 636, 982009112,
1100.00);
INSERT INTO customers VALUES (cus_code_seq.NEXTVAL, 'Ortis', 'Harold', 'O', 617, 982011115,
1000.00);
INSERT INTO customers VALUES (cus_code_seq.NEXTVAL, 'Jimenez', 'Amy', 'J', 618, 8992000113,
1100.00);
INSERT INTO customers VALUES (cus_code_seq.NEXTVAL, 'Bailey', 'Rits', 'B', 619, 898201102,
1100.00);

```

### e) Display Customer Records:

```
SELECT * FROM customers;
```



The screenshot shows a database query result window with two tabs: 'Script Output' and 'Query Result'. The 'Query Result' tab is active, displaying the results of the SQL query 'SELECT \* FROM customers;'. The window indicates that all 5 rows were fetched in 0.009 seconds. The results are presented in a table with 8 columns: CUS\_CODE, CUS\_LNAME, CUS\_FNAME, CUS\_INITIAL, CUS\_AREACODE, CUS\_PHONE, and CUS\_BALANCE. The data is as follows:

	CUS_CODE	CUS_LNAME	CUS_FNAME	CUS_INITIAL	CUS_AREACODE	CUS_PHONE	CUS_BALANCE
1	18887	Mery	Sean	M	615	992008111	1000
2	18888	Conne	Norris	C	636	982009112	1100
3	18889	Ortis	Harold	O	617	982011115	1000
4	18890	Jimenez	Amy	J	618	8992000113	1100
5	18891	Bailey	Rits	B	619	898201102	1100

## 2)Trigger:

### 2. Student Report Table and Trigger

#### a) Create the **student\_report** Table:

```

CREATE TABLE student_report (
  tid NUMBER(4) PRIMARY KEY,
  name VARCHAR2(30),
  subj1 NUMBER(2),
  subj2 NUMBER(2),
  subj3 NUMBER(2),
  total NUMBER(3),
  per NUMBER(3),
  CHECK (subj1 <= 20),
  CHECK (subj2 <= 20),
  CHECK (subj3 <= 20)
);

```

**b) Create Trigger to Automatically Calculate **total** and **per** Fields:**

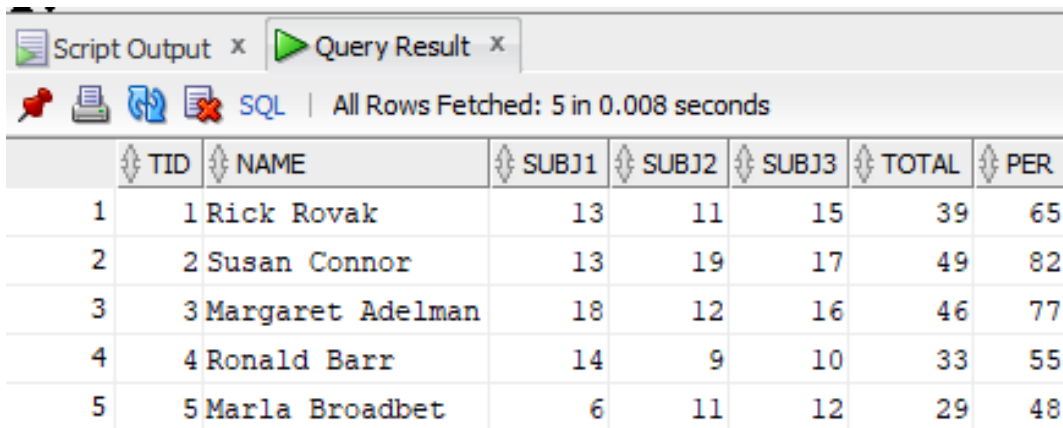
```
CREATE OR REPLACE TRIGGER student_report_trigger
BEFORE INSERT OR UPDATE ON student_report
FOR EACH ROW
BEGIN
    :NEW.total := NVL(:NEW.subj1, 0) + NVL(:NEW.subj2, 0) + NVL(:NEW.subj3, 0);
    :NEW.per := (:NEW.total / 60) * 100;
END;
/
```

**c) Insert Sample Data into **student\_report**:**

```
INSERT INTO student_report VALUES (1, 'Rick Rovak', 13, 11, 15, 0, 0);
INSERT INTO student_report VALUES (2, 'Susan Connor', 13, 19, 17, 0, 0);
INSERT INTO student_report VALUES (3, 'Margaret Adelman', 18, 12, 16, 0, 0);
INSERT INTO student_report VALUES (4, 'Ronald Barr', 14, 9, 10, 0, 0);
INSERT INTO student_report VALUES (5, 'Marla Broadbet', 6, 11, 12, 0, 0);
```

**d) Display Student Report Records:**

```
SELECT * FROM student_report;
```



The screenshot shows a database query result window with a toolbar and a table of results. The toolbar includes icons for saving, printing, and refreshing, along with a status bar indicating 'All Rows Fetched: 5 in 0.008 seconds'. The table has 8 columns: TID, NAME, SUBJ1, SUBJ2, SUBJ3, TOTAL, and PER. It contains 5 rows of data representing student reports.

	TID	NAME	SUBJ1	SUBJ2	SUBJ3	TOTAL	PER
1	1	Rick Rovak	13	11	15	39	65
2	2	Susan Connor	13	19	17	49	82
3	3	Margaret Adelman	18	12	16	46	77
4	4	Ronald Barr	14	9	10	33	55
5	5	Marla Broadbet	6	11	12	29	48

**3)Procedure and Cursor:**

**3. Course Table and Procedures**

**a) Create the **course** Table:**

```
CREATE TABLE course (
    course_num INTEGER PRIMARY KEY,
    course_name VARCHAR2(50),
    dept_name VARCHAR2(15),
    credits INTEGER
);
```

**b) Insert Sample Data into course:**

```
INSERT INTO course VALUES (1001, 'Unix Operating System Lab', 'CSE', 1);
INSERT INTO course VALUES (1002, 'Compiler Construction Lab', 'CSE', 3);
INSERT INTO course VALUES (1003, 'Advanced Database System', 'CSE', 3);
INSERT INTO course VALUES (1004, 'Distributed Systems', 'CSE', 3);
```

**c) Procedure with Cursor to Find Courses Where Name Starts with 'C':**

```
CREATE OR REPLACE PROCEDURE find_courses_starting_with_c IS
    c_name VARCHAR2(50);
    c_credits INTEGER;
    CURSOR cur IS
        SELECT course_name, credits FROM course
        WHERE course_name LIKE 'C%';
BEGIN
    DBMS_OUTPUT.PUT_LINE('Course Name | Credits');
    OPEN cur;
    LOOP
        FETCH cur INTO c_name, c_credits;
        EXIT WHEN cur%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE(c_name || ' | ' || c_credits);
    END LOOP;
    CLOSE cur;
END;
/
```

**d) Procedure with Cursor to Find Courses from 'CSE' Department:**

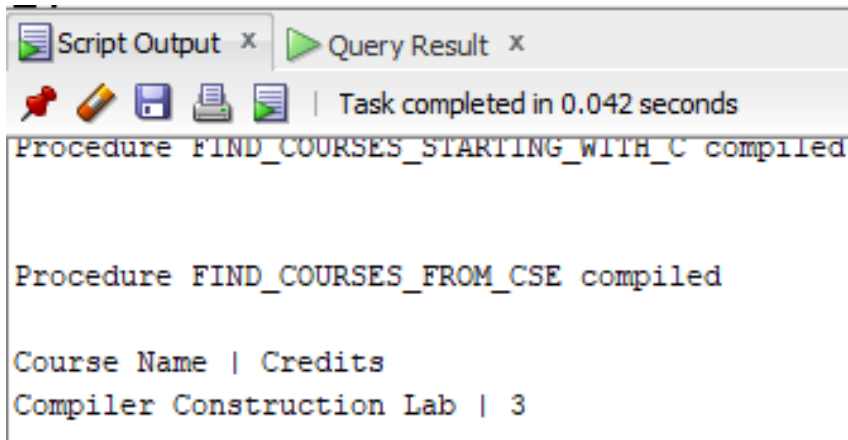
```
CREATE OR REPLACE PROCEDURE find_courses_from_cse IS
    c_name VARCHAR2(50);
    CURSOR cur IS
        SELECT course_name FROM course
        WHERE dept_name = 'CSE';
BEGIN
    DBMS_OUTPUT.PUT_LINE('CSE Course Names');
    OPEN cur;
    LOOP
        FETCH cur INTO c_name;
        EXIT WHEN cur%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE(c_name);
    END LOOP;
    CLOSE cur;
END;
/
```

### e) Execute Procedures:

SET SERVEROUTPUT ON;

-- Execute Procedure for Courses Starting with 'C'

EXEC find\_courses\_starting\_with\_c;



The screenshot shows a SQL Developer window with two tabs: 'Script Output' and 'Query Result'. The 'Script Output' tab is active, displaying the following text: 'Procedure FIND\_COURSES\_STARTING\_WITH\_C compiled', 'Procedure FIND\_COURSES\_FROM\_CSE compiled', and a table with two columns: 'Course Name' and 'Credits'. The table contains one row: 'Compiler Construction Lab' with a value of '3'. The window also shows a status bar indicating 'Task completed in 0.042 seconds'.

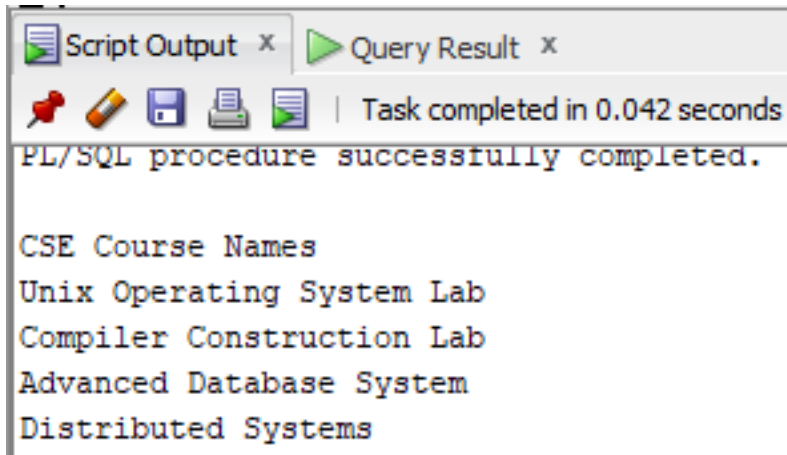
```
Script Output x Query Result x
Task completed in 0.042 seconds
Procedure FIND_COURSES_STARTING_WITH_C compiled

Procedure FIND_COURSES_FROM_CSE compiled

Course Name | Credits
Compiler Construction Lab | 3
```

-- Execute Procedure for Courses from 'CSE'

EXEC find\_courses\_from\_cse;



The screenshot shows a SQL Developer window with two tabs: 'Script Output' and 'Query Result'. The 'Script Output' tab is active, displaying the following text: 'PL/SQL procedure successfully completed.', 'CSE Course Names', 'Unix Operating System Lab', 'Compiler Construction Lab', 'Advanced Database System', and 'Distributed Systems'. The window also shows a status bar indicating 'Task completed in 0.042 seconds'.

```
Script Output x Query Result x
Task completed in 0.042 seconds
PL/SQL procedure successfully completed.

CSE Course Names
Unix Operating System Lab
Compiler Construction Lab
Advanced Database System
Distributed Systems
```