Experiment 06
Advanced SQL

Problem Statement

Oracle Sequences:
Consider table customer with primary key (cus_code)

Field Type	Data Type
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)

A. Create sequence on cus_code
B. Display user sequences
C. Insert values into customer using created sequence
D. Display customer records

Trigger:
Consider Student Report table, in which student marks assessment is recorded. In such schema, create a trigger so that the total and percentage of specified marks is automatically inserted whenever a record is inserting. Initial insert 0 for total and per attributes. Maximum marks should be 20 for each subject.

Field	Type	Null	Key
tid	int(4)	No	PRI
name	varchar(30)	Yes	
subj1	int(2)	Yes	
subj2	int(2)	Yes	
subj3	int(2)	Yes	
total	int(3)	Yes	
per	int(3)	Yes	

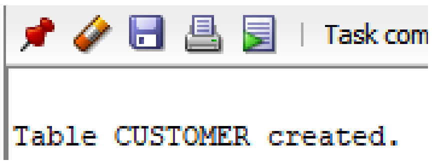
Procedure and Cursor:
Consider Course Table with course_num as primary key.

Field Type	Data Type
course_num	Integer
course_name	varchar2(20)
dept_name	varchar2(15)
credits	Integer

A. Write a procedure which includes cursors: Find course_name and credits where course name starts with ‘C’
B. Write a procedure which includes cursors: Find course names from ‘CSE’ department

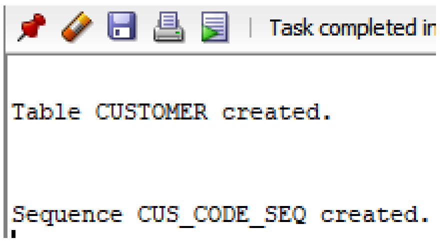
Answers:

1.
```
-- Create Table
CREATE TABLE customer (
    cus_code INTEGER,
    cus_lname VARCHAR2(10),
    cus_fname VARCHAR2(10),
    cus_initial VARCHAR2(1),
    cus_areacode VARCHAR2(3),
    cus_phone VARCHAR2(15),
    cus_balance NUMBER(10, 2),
    PRIMARY KEY (cus_code)
);
```



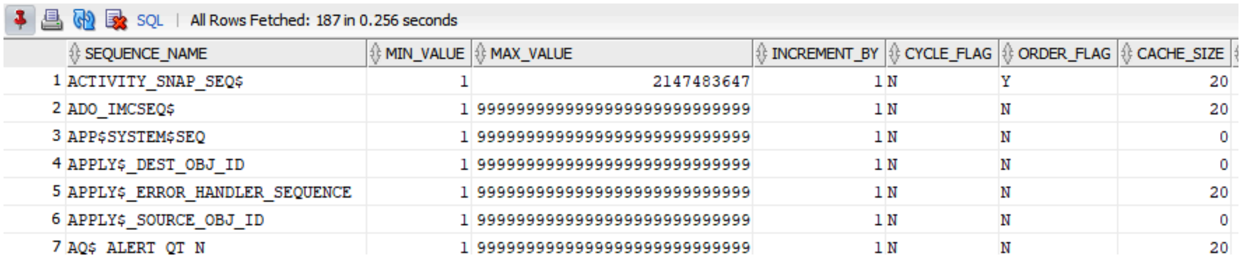
```
Table CUSTOMER created.
```

1. a.
```
-- Create Sequence on cus_code
CREATE SEQUENCE CUS_CODE_SEQ START WITH 1007 NOCACHE;
```



```
Table CUSTOMER created.


Sequence CUS_CODE_SEQ created.
```

1. b.
```
-- Display user sequences
SELECT * FROM USER_SEQUENCES;
```



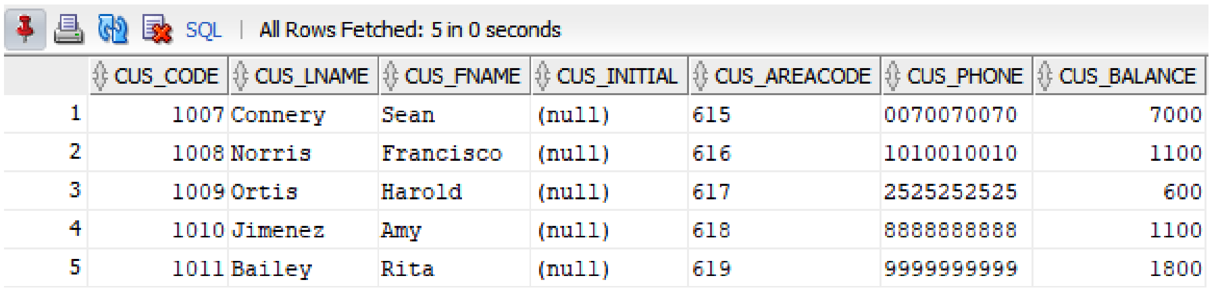
	SEQUENCE_NAME	MIN_VALUE	MAX_VALUE	INCREMENT_BY	CYCLE_FLAG	ORDER_FLAG	CACHE_SIZE
1	ACTIVITY_SNAP_SEQ$	1	2147483647	1	N	Y	20
2	ADO_IMCSEQ$	1	9999999999999999999999999999	1	N	N	20
3	APP$SYSTEM$SEQ	1	9999999999999999999999999999	1	N	N	0
4	APPLY$_DEST_OBJ_ID	1	9999999999999999999999999999	1	N	N	0
5	APPLY$_ERROR_HANDLER_SEQUENCE	1	9999999999999999999999999999	1	N	N	20
6	APPLY$_SOURCE_OBJ_ID	1	9999999999999999999999999999	1	N	N	0
7	AQ$_ALERT_QT_N	1	9999999999999999999999999999	1	N	N	20

1. c.
```
-- Insert Values into customer using created sequence
INSERT INTO customer (cus_code, cus_lname, cus_fname, cus_initial,
cus_areacode, cus_phone, cus_balance)
VALUES (CUS_CODE_SEQ.NEXTVAL, 'Connery', 'Sean', NULL, '615',
'0070070070', 7000.00);

INSERT INTO customer (cus_code, cus_lname, cus_fname, cus_initial,
cus_areacode, cus_phone, cus_balance)
VALUES (CUS_CODE_SEQ.NEXTVAL, 'Norris', 'Francisco', NULL, '616',
'1010010010', 1100.00);

INSERT INTO customer (cus_code, cus_lname, cus_fname, cus_initial,
cus_areacode, cus_phone, cus_balance)
```

```
VALUES (CUS_CODE_SEQ.NEXTVAL, 'Ortis', 'Harold', NULL, '617',
'2525252525', 600.00);

INSERT INTO customer (cus_code, cus_lname, cus_fname, cus_initial,
cus_areacode, cus_phone, cus_balance)
VALUES (CUS_CODE_SEQ.NEXTVAL, 'Jimenez', 'Amy', NULL, '618',
'8888888888', 1100.00);

INSERT INTO customer (cus_code, cus_lname, cus_fname, cus_initial,
cus_areacode, cus_phone, cus_balance)
VALUES (CUS_CODE_SEQ.NEXTVAL, 'Bailey', 'Rita', NULL, '619',
'9999999999', 1800.00);
```

1. d. `-- Display customer records`

```
SELECT * FROM customer;
```

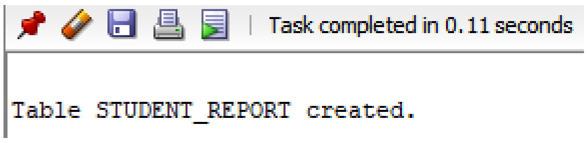


SQL | All Rows Fetched: 5 in 0 seconds

	CUS_CODE	CUS_LNAME	CUS_FNAME	CUS_INITIAL	CUS_AREACODE	CUS_PHONE	CUS_BALANCE
1	1007	Connery	Sean	(null)	615	0070070070	7000
2	1008	Norris	Francisco	(null)	616	1010010010	1100
3	1009	Ortis	Harold	(null)	617	2525252525	600
4	1010	Jimenez	Amy	(null)	618	8888888888	1100
5	1011	Bailey	Rita	(null)	619	9999999999	1800

2.

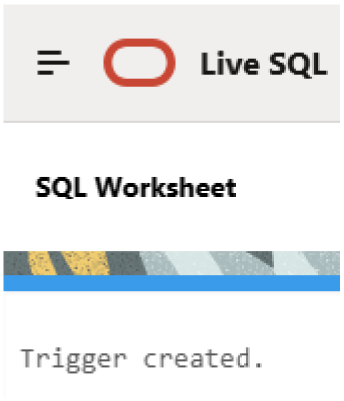
```
-- STUDENT REPORT TABLE
CREATE TABLE student_report (
    tid NUMBER(4),
    name VARCHAR2(30),
    subj1 NUMBER(2),
    subj2 NUMBER(2),
    subj3 NUMBER(2),
    total NUMBER(3),
    per NUMBER(3),
    PRIMARY KEY(tid),
    CHECK(subj1 >= 0 AND subj1 <= 20),
    CHECK(subj2 >= 0 AND subj2 <= 20),
    CHECK(subj3 >= 0 AND subj3 <= 20)
);
```



Task completed in 0.11 seconds

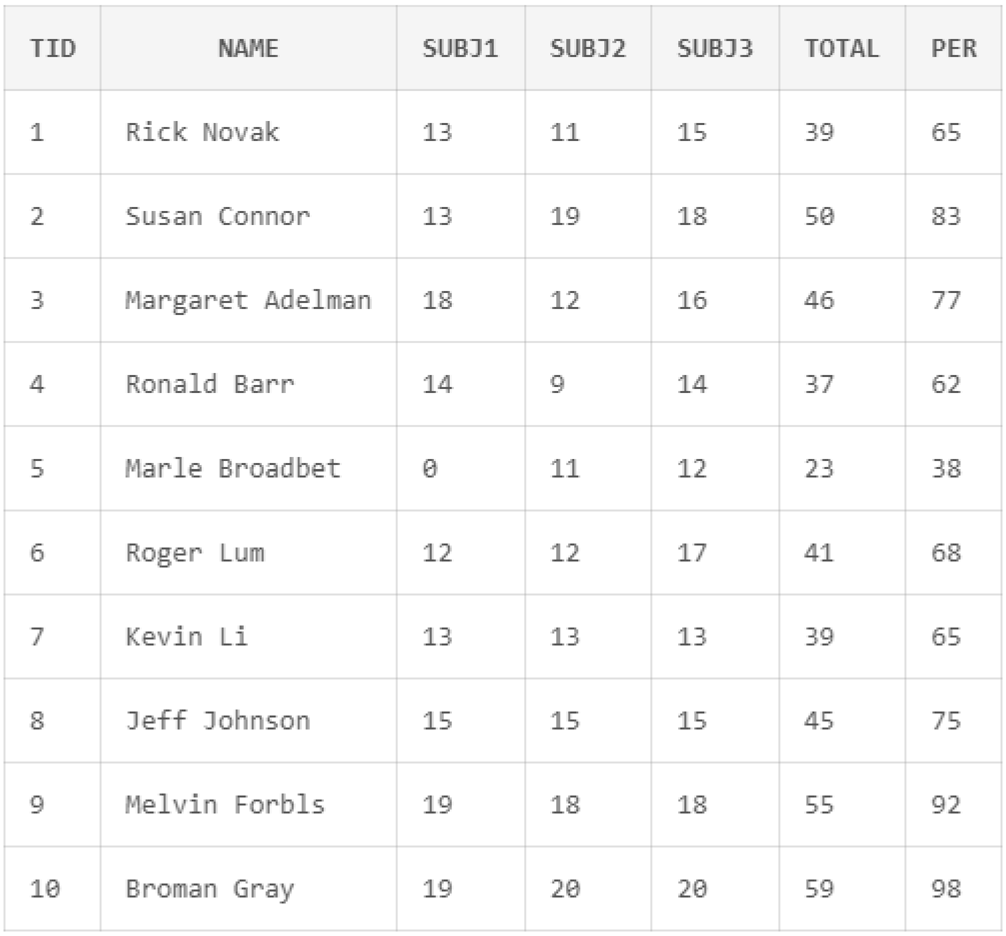
```
Table STUDENT_REPORT created.
```

```
-- CREATE OR REPLACE TRIGGER
CREATE OR REPLACE TRIGGER TRG_CHECK_REPORT
BEFORE INSERT OR UPDATE ON student_report
FOR EACH ROW
BEGIN
    :new.total := :new.subj1 + :new.subj2 + :new.subj3;
    :new.per := ((:new.total) / 60) * 100;
END;
```



```
Trigger created.
```

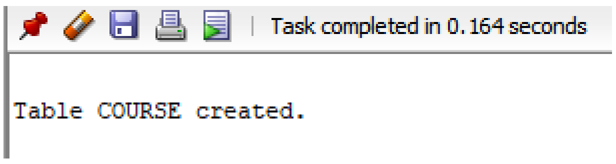
```
-- Inserting data into student_report without the total and per columns
INSERT INTO student_report VALUES (1, 'Rick Novak', 13, 11, 15, NULL,
NULL);
INSERT INTO student_report VALUES (2, 'Susan Connor', 13, 19, 18, NULL,
NULL);
INSERT INTO student_report VALUES (3, 'Margaret Adelman', 18, 12, 16,
NULL, NULL);
INSERT INTO student_report VALUES (4, 'Ronald Barr', 14, 9, 14, NULL,
NULL);
INSERT INTO student_report VALUES (5, 'Marle Broadbet', 0, 11, 12,
NULL, NULL);
INSERT INTO student_report VALUES (6, 'Roger Lum', 12, 12, 17, NULL,
NULL);
INSERT INTO student_report VALUES (7, 'Kevin Li', 13, 13, 13, NULL,
NULL);
INSERT INTO student_report VALUES (8, 'Jeff Johnson', 15, 15, 15, NULL,
NULL);
INSERT INTO student_report VALUES (9, 'Melvin Forbls', 19, 18, 18,
NULL, NULL);
INSERT INTO student_report VALUES (10, 'Broman Gray', 19, 20, 20, NULL,
NULL);

-- Query to view the data
SELECT * FROM student_report;
```



TID	NAME	SUBJ1	SUBJ2	SUBJ3	TOTAL	PER
1	Rick Novak	13	11	15	39	65
2	Susan Connor	13	19	18	50	83
3	Margaret Adelman	18	12	16	46	77
4	Ronald Barr	14	9	14	37	62
5	Marle Broadbet	0	11	12	23	38
6	Roger Lum	12	12	17	41	68
7	Kevin Li	13	13	13	39	65
8	Jeff Johnson	15	15	15	45	75
9	Melvin Forbls	19	18	18	55	92
10	Broman Gray	19	20	20	59	98

3.

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

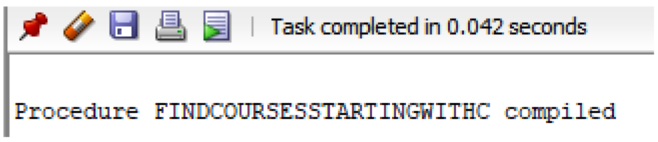


```
Task completed in 0.164 seconds

Table COURSE created.
```

```
-- Inserting data into course Table
INSERT INTO course VALUES(1001, 'Math 1', 'BSH', 3);
INSERT INTO course VALUES(1002, 'Math 2', 'BSH', 3);
INSERT INTO course VALUES(1061, 'Complier Construction Theory', 'CSE',
3);
INSERT INTO course VALUES(1071, 'Advanced Database System Theory',
'CSE', 3);
INSERT INTO course VALUES(1072, 'Distributed System Theory', 'CSE', 3);
INSERT INTO course VALUES(1073, 'Unix Operating System Theory', 'CSE',
3);
INSERT INTO course VALUES(1161, 'Complier Construction Lab', 'CSE', 3);
```

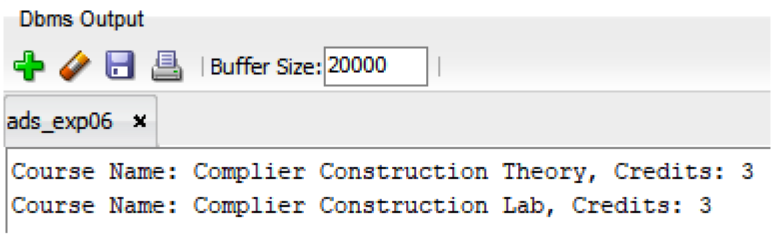
3. a.

```
-- Procedure to find course names and credits where course name starts
with 'C'
CREATE OR REPLACE PROCEDURE FindCoursesStartingWithC IS
    CURSOR c_courses IS
        SELECT course_name, credits
        FROM Course
        WHERE course_name LIKE 'C%';  -- Course names starting with 'C'

    v_course_name Course.course_name%TYPE;   -- Variable to hold course
name
    v_credits Course.credits%TYPE;            -- Variable to hold
credits
BEGIN
    OPEN c_courses;
    LOOP
        FETCH c_courses INTO v_course_name, v_credits;
        EXIT WHEN c_courses%NOTFOUND;  -- Exit loop if no more records
        DBMS_OUTPUT.PUT_LINE('Course Name: ' || v_course_name || ',
Credits: ' || v_credits);
    END LOOP;
    CLOSE c_courses;
END;
/
```



```
Task completed in 0.042 seconds

Procedure FINDCOURSESSTARTINGWITHC compiled
```

```
-- Call the procedure to find courses starting with 'C'
BEGIN
    FindCoursesStartingWithC;
END;
/
```



```
Course Name: Complier Construction Theory, Credits: 3
Course Name: Complier Construction Lab, Credits: 3
```

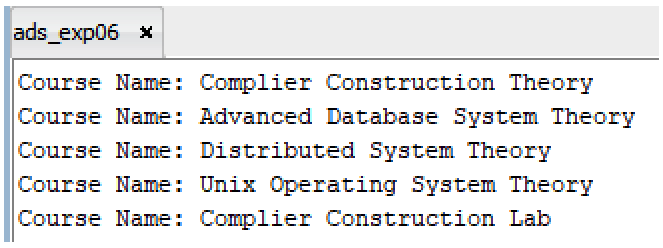
3. b.

```
-- Procedure to find course names from 'CSE' department
CREATE OR REPLACE PROCEDURE FindCoursesInCSE IS
    CURSOR c_cse_courses IS
        SELECT course_name
        FROM Course
        WHERE dept_name = 'CSE';  -- Department name is 'CSE'

    v_course_name Course.course_name%TYPE;  -- Variable to hold course
name
BEGIN
    OPEN c_cse_courses;
    LOOP
        FETCH c_cse_courses INTO v_course_name;
        EXIT WHEN c_cse_courses%NOTFOUND;  -- Exit loop if no more
records
        DBMS_OUTPUT.PUT_LINE('Course Name: ' || v_course_name);
    END LOOP;
    CLOSE c_cse_courses;
END;
/
```

```
Task completed in 0.043 seconds
```

```
Procedure FINDCOURSESINCSE compiled
```

```
-- Call the procedure to find courses in the 'CSE' department
BEGIN
    FindCoursesInCSE;
END;
/
```



```
Course Name: Complier Construction Theory
Course Name: Advanced Database System Theory
Course Name: Distributed System Theory
Course Name: Unix Operating System Theory
Course Name: Complier Construction Lab
```