```
1.SELECT * FROM MCSC.TaskList WHERE "EMAIL" = :val1
2.SELECT * FROM MCSC.TaskList WHERE "UID" = :val1
3. SELECT * FROM MCSC.TaskList WHERE "EMAIL" = :val1
4. SELECT * FROM MCSC.TaskList WHERE "UID" = :val1 AND "USER_ID" = :val2
5. SELECT * FROM MCSC.TaskList WHERE "UID" = :val1 AND "EMAIL" = :val2
6. SELECT contestID, contestName FROM MCSC.Contest )
7. SELECT contestID, contestName FROM MCSC.Contest )
8. SELECT contestID, contestName FROM MCSC.Contest WHERE )
9. INSERT INTO MCSC.TaskList ("PLATFORM", "TASKNAME", "TASKURL", "TASKNOTE",
"STATUS", "EMAIL") VALUES (:val1, :val2, :val3, :val4, :val5, :val6)
10. UPDATE MCSC.TaskList
   SET "TASKNAME" = :v1,
      "TASKURL" = :v2,
      "PLATFORM" = :v3,
      "STATUS" = :v4,
      "TASKNOTE" = :v5
   WHERE "EMAIL" = :v7 AND "UID" = :v6
11. DELETE FROM TaskList WHERE "UID" = :val1 AND "EMAIL" =:val2
12. INSERT INTO MCSC.Users (first_name, last_name, email, contactNumber, password,
account_type, image)
      VALUES (:first_name, :last_name, :email, :contactNumber, :password, :account_type,
:image)

13. SELECT "USER_ID" FROM MCSC.USERS WHERE "EMAIL" = :val1
14. `SELECT "USER_ID", "IMAGE", "FIRST_NAME" || "LAST_NAME" AS "FULLNAME" FROM
MCSC.USERS WHERE "EMAIL" = :val1
15. SELECT * FROM MCSC.USERS WHERE "EMAIL" = :val1
16. INSERT INTO MCSC.Category (name, description) VALUES(:v1, :v2)
17. DELETE FROM MCSC.Category WHERE CATEGORY_ID = :v1
18. SELECT * FROM MCSC.CATEGORY WHERE CATEGORY_ID = :v1
19. SELECT * FROM MCSC.CATEGORY
20. SELECT
   "A2"."CATEGORY_ID"         "CATEGORY_ID",
   "A2"."NAME"                "CAT_NAME",
   "A2"."DESCRIPTION"         "CAT_DESCRIPTION",
   "A2"."COURSES"             "COURSES",
   "A1"."COURSE_ID"           "COURSE_ID",
   "A1"."COURSE_NAME"         "COURSE_NAME",
   "A1"."COURSE_DESCRIPTION"  "COURSE_DESCRIPTION",
   "A1"."INSTRUCTOR"          "INSTRUCTOR",
   "A1"."WHAT_YOU_WILL_LEARN" "WHAT_YOU_WILL_LEARN",
   "A1"."PRICE"               "PRICE",
   "A1"."THUMBNAIL"           "THUMBNAIL",
   "A1"."STATUS"              "STATUS",
   "A1"."CREATED_AT"          "CREATED_AT",
   "A1"."SOLD"                "SOLD",
   "A1"."TAG"                 "TAG",
   "A1"."INSTRUCTIONS"        "INSTRUCTIONS",
   "A1"."POINTS"              "POINTS"
FROM
   "MCSC"."CATEGORY" "A2",
   "MCSC"."COURSES"  "A1"
WHERE
   "A2"."CATEGORY_ID" = "A1"."CATEGORY"
21. SELECT
   "A2"."CATEGORY_ID"         "CATEGORY_ID",
   "A2"."NAME"                "CAT_NAME",
   "A2"."DESCRIPTION"         "CAT_DESCRIPTION",
   "A2"."COURSES"             "COURSES",
   "A1"."COURSE_ID"           "COURSE_ID",
   "A1"."COURSE_NAME"         "COURSE_NAME",
   "A1"."COURSE_DESCRIPTION"  "COURSE_DESCRIPTION",
   "A1"."INSTRUCTOR"          "INSTRUCTOR",
   "A1"."WHAT_YOU_WILL_LEARN" "WHAT_YOU_WILL_LEARN",
   "A1"."PRICE"               "PRICE",
   "A1"."THUMBNAIL"           "THUMBNAIL",
   "A1"."STATUS"              "STATUS",
   "A1"."CREATED_AT"          "CREATED_AT",
   "A1"."SOLD"                "SOLD",
   "A1"."TAG"                 "TAG",
   "A1"."INSTRUCTIONS"        "INSTRUCTIONS",
   "A1"."POINTS"              "POINTS"
FROM
   "MCSC"."CATEGORY" "A2",
   "MCSC"."COURSES"  "A1"
WHERE
   "A2"."CATEGORY_ID" = "A1"."CATEGORY"
   AND A2.NAME = :v1
22. SELECT c.course_id AS "COURSE_ID", c.course_name AS "COURSE_NAME",
c.THUMBNAIL, C.CATEGORY AS "CATEGORY_ID",
            c.course_description AS "COURSE_DESCRIPTION", c.PRICE, c.STATUS,
c.CREATED_AT, c.TAG, c.INSTRUCTIONS, c.POINTS, c.WHAT_YOU_WILL_LEARN, c.SOLD,
            i.first_name || ' ' || i.last_name AS "INSTRUCTOR_NAME", i.IMAGE AS
"INSTRUCTOR_IMAGE"
            FROM MCSC.Courses c, MCSC.Category cat, MCSC.USERS I
            WHERE c.category = cat.category_id AND I.USER_ID = C.INSTRUCTOR
            AND cat.category_id = :v1 AND c.status = 'Published'
23. SELECT AVG(A1.RATING) AS AVG_RATING,
        COUNT(A1.RATING) AS TOTAL_RATING
   FROM MCSC.RATINGANDREVIEWS A1
   WHERE A1.CATEGORY_ID = :b1
24. SELECT c.course_id AS "COURSE_ID", c.course_name AS "COURSE_NAME",
c.THUMBNAIL, C.CATEGORY AS "CATEGORY_ID",
            c.course_description AS "COURSE_DESCRIPTION", c.PRICE, c.STATUS,
c.CREATED_AT, c.TAG, c.INSTRUCTIONS, c.POINTS, c.WHAT_YOU_WILL_LEARN, c.SOLD,
            i.first_name || ' ' || i.last_name AS "INSTRUCTOR_NAME", i.IMAGE AS
"INSTRUCTOR_IMAGE"
            FROM MCSC.Courses c, MCSC.Category cat, MCSC.USERS I
            WHERE c.category = cat.category_id AND I.USER_ID = C.INSTRUCTOR
            AND cat.category_id <> :v1 AND c.status = 'Published'
25. SELECT AVG(A1.RATING) AS AVG_RATING,
        COUNT(A1.RATING) AS TOTAL_RATING
   FROM MCSC.RATINGANDREVIEWS A1
   WHERE A1.CATEGORY_ID <> :b1
26. SELECT c.course_id, c.course_name, c.course_description, c.sold, c.THUMBNAIL,
            i.first_name || ' ' || i.last_name AS "INSTRUCTOR_NAME", i.IMAGE AS
"INSTRUCTOR_IMAGE",
            cat.name AS category_name, c.PRICE, c.STATUS, c.CREATED_AT, c.TAG,
c.INSTRUCTIONS, c.POINTS, c.WHAT_YOU_WILL_LEARN,
            c.SOLD, c.category as "CATEGORY_ID", c.INSTRUCTOR as "INSTRUCTOR_ID",
            ROWNUM as rnum
      FROM MCSC.Courses c, MCSC.Users i, MCSC.Category cat
      WHERE i.user_id = c.instructor AND cat.category_id = c.category AND ROWNUM <=
:fetchAtmost
      ORDER BY c.SOLD DESC
27. `SELECT R.rating, R.REVIEW, U.USER_ID, U.FIRST_NAME, U.LAST_NAME,
U.ACCOUNT_TYPE  FROM MCSC.RATINGANDREVIEWS R, MCSC.USERS U WHERE
course_id = :v1 AND U.USER_ID = R.USER_ID
28. UPDATE MCSC.CATEGORY SET COURSES = CASE WHEN COURSES IS NULL THEN
:COURSE_ID ELSE COURSES || ',' || :COURSE_ID END WHERE CATEGORY_ID =
:categoryId

29. INSERT INTO MCSC.COURSES (COURSE_ID, CATEGORY) VALUES (:COURSE_ID,
:categoryId)
30. INSERT INTO MCSC.Instructor(INSTR_ID, COURSES) VALUES (:instruct, :COURSE_ID)
31. INSERT INTO MCSC.Course_StudentsEnrolled(COURSE_ID, STUDENT_ID) VALUES
(:COURSE_ID, :studentId)
32. UPDATE MCSC.Users
     SET courses = CASE WHEN courses IS NULL THEN :COURSE_ID
                    ELSE courses || ',' || :COURSE_ID
                END
     WHERE user_id = :studentId
```

```
33. SELECT
   A2.COURSE_ID         ,
   A2.STUDENT_ID      ,
   A1.USER_ID         ,
   A1.EMAIL           ,
   A1.ACCOUNT_TYPE     ,
   A1.CONTACTNUMBER     ,
   A1.ACTIVE           ,
   A1.APPROVED         ,
   A1.IMAGE            ,
   A1.COURSE_PROGRESS   ,
   A1.LAST_NAME         ,
   A1.FIRST_NAME
FROM
   MCSC.COURSE_STUDENTSENROLLED A2,
   MCSC.USERS                   A1
WHERE
       A2.COURSE_ID = :b1
   AND A2.STUDENT_ID = A1.USER_ID
34. SELECT * FROM MCSC.Course_StudentsEnrolled WHERE COURSE_ID = :COURSE_ID
AND STUDENT_ID = :studentId
25. SELECT SUM(PRICE), COUNT(*)  FROM MCSC.COURSES A, MCSC.INSTRUCTOR B
WHERE A.COURSE_ID = B.COURSES AND A.INSTRUCTOR = :instrId;
36. `SELECT * FROM MCSC.COURSES
37. SELECT * FROM MCSC.COURSES  WHERE COURSE_ID = :id
38. SELECT
CASE WHEN SUM(time_duration) IS NULL THEN 0
    ELSE SUM(time_duration)
END AS "DURATION"
FROM MCSC.SUBSECTION WHERE COURSE_ID = :COURSE_ID
39.  SELECT
   "A1"."COURSE_ID",
   "A1"."COURSE_NAME"         "COURSE_NAME",
   "A1"."COURSE_DESCRIPTION"  "COURSE_DESCRIPTION",
   "A1"."INSTRUCTOR"          "INSTRUCTOR",
   "A1"."WHAT_YOU_WILL_LEARN" "WHAT_YOU_WILL_LEARN",
   "A1"."PRICE"               "PRICE",
   "A1"."THUMBNAIL"           "THUMBNAIL",
   "A1"."STATUS"              "STATUS",
   "A1"."CREATED_AT"          "CREATED_AT",
   "A1"."CATEGORY"            "CATEGORY",
   "A1"."SOLD"                "SOLD",
   "A1"."TAG"                 "TAG",
   "A1"."INSTRUCTIONS"        "INSTRUCTIONS",
   "A1"."POINTS"             "POINTS",
   U.FIRST_NAME,
   U.LAST_NAME,
   U.IMAGE,
   U.EMAIL
FROM
   "MCSC"."COURSES" "A1",
   MCSC.INSTRUCTOR I,
   MCSC.USERS U
WHERE
   "A1"."COURSE_ID" = :COURSE_ID AND
   I.COURSES = A1.COURSE_ID AND
   U.USER_ID = I.INSTR_ID
40. SELECT * FROM MCSC.RATINGANDREVIEWS WHERE COURSE_ID = :COURSE_ID
41. select * from mcsc.section where course_id = :courseId
42. SELECT * FROM MCSC.SUBSECTION WHERE SECTION_ID = :SECTION_ID AND
COURSE_ID = :COURSE_ID
43. SELECT
   "A1"."COURSE_ID",
   "A1"."COURSE_NAME"        "COURSE_NAME",
   "A1"."COURSE_DESCRIPTION"  "COURSE_DESCRIPTION",
   "A1"."INSTRUCTOR"         "INSTRUCTOR",
   "A1"."WHAT_YOU_WILL_LEARN" "WHAT_YOU_WILL_LEARN",
   "A1"."PRICE"              "PRICE",
   "A1"."THUMBNAIL"          "THUMBNAIL",
   "A1"."STATUS"             "STATUS",
   "A1"."CREATED_AT"         "CREATED_AT",
   "A1"."CATEGORY"           "CATEGORY",
   "A1"."SOLD"               "SOLD",
   "A1"."TAG"                "TAG",
   "A1"."INSTRUCTIONS"       "INSTRUCTIONS",
   "A1"."POINTS"             "POINTS",
   R.RATING,
   R.REVIEW,
   U.FIRST_NAME,
   U.LAST_NAME,
   U.IMAGE,
   U.EMAIL
FROM
   "MCSC"."COURSES" "A1",
   MCSC.RATINGANDREVIEWS R,
   MCSC.INSTRUCTOR I,
   MCSC.USERS U
WHERE
```

```
    "A1"."COURSE_ID" = :COURSE_ID AND
    A1.COURSE_ID = R.COURSE_ID (+) AND
    I.COURSES = A1.COURSE_ID AND
    U.USER_ID = I.INSTR_ID
44. select * from mcsc.section where course_id = :courseId  FETCH FIRST :fetch ROWS ONLY
45. SELECT * FROM MCSC.SUBSECTION WHERE SECTION_ID = :SECTION_ID AND
COURSE_ID = :COURSE_ID
    FETCH FIRST :fetch ROWS ONLY
46. SELECT A.course_name AS "COURSE_NAME", A.instructor, U.first_name AS
"FIRST_NAME",
   U.last_name AS "LAST_NAME", U.image, U.EMAIL, A.PRICE, A.THUMBNAIL,
  B.Rating, B.Review, COUNT(C.STUDENT_ID) AS "studentsEnrolled" FROM
MCSC.COURSES A,
  MCSC.Users U, MCSC.RATINGANDREVIEWS B, MCSC.Course_StudentsEnrolled C
  WHERE A.STATUS = 'Published' AND A.COURSE_ID = B.COURSE_ID AND A.instructor =
U.user_id
  AND C.COURSE_ID = A.COURSE_ID
  GROUP BY A.course_name, A.instructor, U.first_name, U.last_name, U.image, U.EMAIL
,A.PRICE,
  A.THUMBNAIL, B.Rating, B.Review
  ORDER BY A.course_name ASC
47. SELECT
    COUNT(P.completed_videos) AS total_completed
FROM
    MCSC.COURSEPROGRESS P
WHERE
    P.COURSE_ID = :COURSE_ID
    AND P.USER_ID = :studentId
48. SELECT
    COUNT(S.video_url) AS total_videos,
    SUM(S.TIME_DURATION) AS DURATION

    FROM
        MCSC.SUBSECTION S


    WHERE
        S.COURSE_ID = :COURSE_ID
49. SELECT COMPLETED_VIDEOS, SECTION_ID, SUBSECTION_ID FROM
MCSC.COURSEPROGRESS
   WHERE COURSE_ID = :COURSE_ID AND USER_ID = :studentId
50. select * from mcsc.section where course_id = :courseId
51. select * from mcsc.subsection where course_id = :courseId
52. select * from mcsc.section where course_id = :courseId
```

53. SELECT * FROM MCSC.SUBSECTION WHERE SECTION_ID = :SECTION_ID AND COURSE_ID = :COURSE_ID
54. SELECT * FROM MCSC.INSTRUCTOR I, MCSC.COURSES c WHERE I.INSTR_ID = :instrId AND C.COURSE_ID = I.COURSES
55. SELECT A.COURSE_ID, A.COURSE_NAME, A.COURSE_DESCRIPTION, A.SOLD, A.PRICE, B.INSTR_ID
    FROM MCSC.COURSES A, MCSC.INSTRUCTOR B, MCSC.COURSE_STUDENTSENROLLED C WHERE A.COURSE_ID = B.COURSES AND A.INSTRUCTOR = :instrId
    AND A.COURSE_ID = C.COURSE_ID
56. INSERT INTO MCSC.SECTION (COURSE_ID, SECTION_NAME) VALUES(:cid, :sn)
57. UPDATE MCSC.SECTION
    SET SECTION_NAME = :sn WHERE SECTION_ID = :sid
58. DELETE FROM MCSC.SECTION WHERE SECTION_ID = :sid
59. UPDATE MCSC.COURSEPROGRESS
    SET SUBSECTION_ID = :subsectionId,
    COMPLETED_VIDEOS = (SELECT VIDEO_URL FROM MCSC.SUBSECTION where SUBSECTION_ID = :subsectionId)
    WHERE COURSE_ID = :COURSE_ID AND
        USER_ID = :USER_ID;
60. INSERT INTO MCSC.COURSEPROGRESS (COURSE_ID, USER_ID) VALUES (:COURSE_ID, :USER_ID);
61. INSERT INTO MCSC.SUBSECTION (SECTION_ID, TITLE, TIME_DURATION, DESCRIPTION, VIDEO_URL) VALUES(:secid,:title, :dur, :des, :uri)
62.`SELECT * FROM MCSC.SUBSECTION WHERE SUBSECTION_ID = :subsId`
63. UPDATE MCSC.SUBSECTION
    SET TITLE = :title, "DESCRIPTION" = :des, VIDEO_URL = :uri, TIME_DURATION = :dur WHERE SUBSECTION_ID = :sid
64. DELETE FROM MCSC.SUBSECTION WHERE SUBSECTION_ID = :SUBSID, SECTION_ID = :SECTIONID
65. SELECT otp FROM MCSC.OTP WHERE "EMAIL" = :val ORDER BY "CREATED_AT" DESC FETCH FIRST 1 ROW ONLY
66. SELECT otp FROM MCSC.OTP WHERE "otp" = :val
67.INSERT INTO MCSC.OTP ("EMAIL", "OTP") VALUES (:v1, :v2)
68.INSERT INTO MCSC.PROFILE (user_id, GENDER, date_Of_Birth, ABOUT, contact_Number) VALUES (:v0, :v1, :v2, :v3, :v4)
69. SELECT
    A.USER_ID, A.FIRST_NAME, A.LAST_NAME, A.EMAIL, A.PASSWORD, A.ACCOUNT_TYPE, A.ACTIVE,
    A.APPROVED, A.TOKEN, A.RESET_PASSWORD_EXPIRES, A.IMAGE, A.COURSES, A.COURSE_PROGRESS, A.CONTACTNUMBER,
    B.PROFILE_ID, B.GENDER, B.DATE_OF_BIRTH, B.ABOUT, B.CONTACT_NUMBER,
    C.ACCOUNT_ID, C.BALANCE
FROM

```
   MCSC.USERS A,
   MCSC.PROFILE B,
   MCSC.ACCOUNT C
WHERE
   A.EMAIL = :val
   AND A.USER_ID = B.USER_ID (+)
   AND A.USER_ID = C.USER_ID (+)
70. SELECT A.CONTESTID AS HOSTEDCONTEST, A.HOST_ID FROM MCSC.HOST A
WHERE A.USER_ID = :val
71. select * from mcsc.ratingandreviews
   where course_id = :COURSE_ID and user_id = :USER_ID

72. select AVG(RATING) AS "averageRating" from mcsc.ratingandreviews where course_id =
:COURSE_ID
73. insert into mcsc.ratingandreviews ( USER_ID, COURSE_ID , RATING, REVIEW)
   VALUES ( :USER_ID, :cid, :rating, :review)
74. select B.FIRST_NAME, B.LAST_NAME, B.EMAIL, B.IMAGE, C.COURSE_ID,
C.COURSE_NAME, A.RATING, A.REVIEW
from mcsc.ratingandreviews A, mcsc.users B, mcsc.courses C
where A.USER_ID = B.USER_ID AND A.COURSE_ID = C.COURSE_ID
order by rating desc
75.  INSERT INTO MCSC.Users (first_name, last_name, EMAIL, CONTACTNUMBER,
password, ACCOUNT_TYPE, image)
      VALUES (:first_name, :last_name, :EMAIL, :CONTACT_NUMBER, :password,
:ACCOUNT_TYPE, :image)
76. `SELECT "USER_ID" FROM MCSC.USERS WHERE "EMAIL" = :val1`
77. SELECT * FROM MCSC.USERS WHERE USER_ID = :USER_ID
78.SELECT * FROM MCSC.USERS WHERE email = :email
79.SELECT USER_ID, FIRST_NAME,LAST_NAME, IMAGE FROM MCSC.USERS WHERE
EMAIL = :em
80.SELECT A3.USER_ID, A3.EMAIL, A3.PASSWORD, A3.ACCOUNT_TYPE,
A3.CONTACTNUMBER, A3.ACTIVE, A3.APPROVED, A3.TOKEN,
A3.RESET_PASSWORD_EXPIRES, A3.IMAGE, A3.COURSES, A3.COURSE_PROGRESS,
A3.LAST_NAME, A3.FIRST_NAME,
A2.PROFILE_ID, A2.GENDER, A2.DATE_OF_BIRTH, A2.ABOUT, A2.CONTACT_NUMBER,
A4.ACCOUNT_ID, A4.BALANCE
FROM MCSC.USERS A3
LEFT JOIN MCSC.PROFILE A2 ON A2.USER_ID = A3.USER_ID
LEFT JOIN MCSC.ACCOUNT A4 ON A4.USER_ID = A3.USER_ID
WHERE A3.USER_ID = :USER_ID
81. SELECT A.CONTESTID AS HOSTEDCONTEST, A.HOST_ID FROM MCSC.HOST A
WHERE A.USER_ID = :val`,  { val: res.rows[0].USER_ID }, `Failed to get host info`, `Host info
fetched
82. UPDATE MCSC.USERS
```

```
   SET IMAGE = :imgurl
   WHERE USER_ID = :USER_ID
83. SELECT COURSE_ID FROM MCSC.Course_StudentsEnrolled WHERE STUDENT_ID =
:studentid
84. UPDATE MCSC.USERS
  SET PASSWORD = :newPass,
     TOKEN = :token,
  WHERE USER_ID = :USER_ID
85. UPDATE MCSC.USERS
   SET FIRST_NAME = :FIRST_NAME,
      LAST_NAME = :LAST_NAME
   WHERE USER_ID = :USER_ID
86. UPDATE MCSC.Profile
   SET GENDER = :gn,
      DATE_OF_BIRTH = TO_DATE(:dob, 'YYYY-MM-DD'),
      ABOUT = :ab,
      CONTACT_NUMBER = :CONTACT_NUMBER
   WHERE USER_ID = :USER_ID
87. DELETE FROM MCSC.USERS WHERE USER_ID = :USER_ID
88.SELECT * FROM MCSC.USERS WHERE USER_ID = :USER_ID
89. SELECT "USER_ID", "FIRST_NAME" || ' '|| "LAST_NAME" AS FULLNAME, "EMAIL" FROM
MCSC.USERS WHERE "ACCOUNT_TYPE" = 'Instructor' AND "USER_ID" = :USER_ID
90. UPDATE MCSC.USERS SET "PASSWORD" = :val1 WHERE "USER_ID" = :val2
91. SELECT * FROM USERS WHERE "EMAIL" = :val1
92. SELECT A.USER_ID, A.FIRST_NAME, A.LAST_NAME, A.EMAIL, A.IMAGE,
B.CONTACT_NUMBER,  B.GENDER, B.DATE_OF_BIRTH, B.ABOUT , D.COURSE_NAME
  FROM MCSC.USERS A, MCSC.PROFILE B, MCSC.COURSE_STUDENTSENROLLED C,
MCSC.COURSES D
  WHERE ACCOUNT_TYPE = 'Student' AND A.USER_ID = B.USER_ID (+)
  AND C.STUDENT_ID (+)= A.USER_ID AND C.COURSE_ID  = D.COURSE_ID(+)
93.  SELECT
         "A2"."USER_ID"               "USER_ID",
         "A3"."EMAIL"                 "EMAIL",
         "A3"."ACCOUNT_TYPE"          "ACCOUNT_TYPE",
         "A3"."CONTACTNUMBER"         "CONTACTNUMBER",
         "A3"."ACTIVE"                "ACTIVE",
         "A3"."APPROVED"              "APPROVED",
         "A3"."IMAGE"                 "IMAGE",
         "A3"."COURSES"               "COURSES",
         "A3"."COURSE_PROGRESS"       "COURSE_PROGRESS",
         "A3"."LAST_NAME"             "LAST_NAME",
         "A3"."FIRST_NAME"            "FIRST_NAME",
         "A2"."PROFILE_ID"            "PROFILE_ID",
         "A2"."GENDER"                "GENDER",
         "A2"."DATE_OF_BIRTH"          "DATE_OF_BIRTH",
         "A2"."ABOUT"                  "ABOUT",
         "A2"."CONTACT_NUMBER"         "CONTACT_NUMBER"
      FROM
         "MCSC"."USERS"   "A3",
         "MCSC"."PROFILE" "A2"
      WHERE
         "A3"."USER_ID" = "A2"."USER_ID" AND
         "A3"."ACCOUNT_TYPE" = 'Instructor'
94. SELECT
   "A1"."COURSE_ID"          "COURSE_ID",
   "A1"."COURSE_NAME"        "COURSE_NAME",
   "A1"."COURSE_DESCRIPTION"  "COURSE_DESCRIPTION",
   "A1"."INSTRUCTOR"         "INSTRUCTOR",
   "A1"."WHAT_YOU_WILL_LEARN" "WHAT_YOU_WILL_LEARN",
   "A1"."PRICE"              "PRICE",
   "A1"."THUMBNAIL"          "THUMBNAIL",
   "A1"."STATUS"             "STATUS",
   "A1"."CREATED_AT"         "CREATED_AT",
   "A1"."CATEGORY"           "CATEGORY",
   "A1"."SOLD"               "SOLD",
   "A1"."TAG"                "TAG",
   "A1"."INSTRUCTIONS"       "INSTRUCTIONS",
   "A1"."POINTS"             "POINTS"
FROM
   "MCSC"."INSTRUCTOR" "A2",
   "MCSC"."COURSES"    "A1"
WHERE
      "A2"."INSTR_ID" = :b1
   AND "A2"."INSTR_ID" = "A1"."INSTRUCTOR"
   AND "A2"."COURSES" = "A1"."COURSE_ID";
95. insert into mcsc.tokens (USER_ID, TOKEN, EXPIRES_AT) VALUES(:USER_ID, :token,
:exp);
96. select * from mcsc.tokens where token = :token;
97. SELECT * FROM TaskList WHERE "USER_ID" = :val1
98. SELECT * FROM TaskList WHERE "uid" = :val1
99. SELECT * FROM TaskList WHERE "USER_ID" = :val1
100. SELECT * FROM TaskList WHERE "uid" = :val1 AND "USER_ID" = :val2
101. INSERT INTO MCSC.ACCOUNT (USER_ID, BALANCE) VALUES (:USER_ID,
:BALANCE);
102. SELECT * FROM MCSC.ACCOUNT WHERE USER_ID = :userid
103. INSERT INTO MCSC.ACCOUNT (USER_ID, BALANCE) VALUES (:userid, 100)
104. SELECT * FROM MCSC.ACCOUNT WHERE USER_ID = :userid
105. UPDATE MCSC.ACCOUNT
```

```
          SET BALANCE = BALANCE - :amnt
          WHERE USER_ID = :userid`
106. INSERT INTO MCSC.TRX_HISTORY (TRXID, USER_ID, ACCOUNT_ID, date_of_trx,
amount_of_trx)
          VALUES (:trxid, :userid, :accountid, TO_TIMESTAMP(:datetrx, 'YYYY-MM-DD
HH24:MI:SS'), :amnt)
107. INSERT INTO MCSC.OTP (EMAIL, otp)
     VALUES (:EMAIL, :otp)
108. SELECT otp_id, EMAIL, otp, created_at, expires_at
     FROM MCSC.OTP
     WHERE EMAIL = :EMAIL
109. SELECT otp FROM MCSC.OTP WHERE "EMAIL" = :val ORDER BY "CREATED_AT"
DESC FETCH FIRST 1 ROW ONLY
110. SELECT otp FROM MCSC.OTP WHERE "otp" = :val
111. INSERT INTO MCSC.OTP ("EMAIL", "OTP") VALUES (:v1, :v2)
*********************************************************************************************************
*********************************************************************************************************
********************************************************************##########################
###########################################################################
########################################################

DECLARE
       dummy INTEGER;
   BEGIN
       EXECUTE IMMEDIATE '
       CREATE OR REPLACE PROCEDURE MCSC.GET_LEADERBOARD (
          p_contestID IN MCSC.SUBMISSION.CONTESTID%TYPE,
          o_cursor OUT SYS_REFCURSOR
       ) IS
       BEGIN
          OPEN o_cursor FOR
             SELECT
                S.USER_ID,
                SUM(S.POINTS) AS TOTAL_POINTS,
                U.FIRST_NAME,
                U.LAST_NAME,
                U.EMAIL,
                U.IMAGE
             FROM
                MCSC.SUBMISSION S
             JOIN
                MCSC.USERS U ON S.USER_ID = U.USER_ID
             WHERE
                S.CONTESTID = p_contestID
```

```
            GROUP BY
               S.USER_ID,
               U.FIRST_NAME,
               U.LAST_NAME,
               U.EMAIL,
               U.IMAGE
            ORDER BY
               TOTAL_POINTS DESC;
      END GET_LEADERBOARD;';
   END;
2.BEGIN
            MCSC.GET_LEADERBOARD(:contestID, :cursor);
         END;
3.  DECLARE
   v1 NUMBER := :id;
   f NUMBER := :flag;


   TYPE t_course_rec IS RECORD (
      COURSE_ID MCSC.Courses.course_id%TYPE,
      COURSE_NAME MCSC.Courses.course_name%TYPE,
      COURSE_DESCRIPTION MCSC.Courses.course_description%TYPE,
      REVIEW MCSC.RatingAndReviews.review%TYPE,
      RATING MCSC.RatingAndReviews.rating%TYPE,
      INSTRUCTOR_NAME VARCHAR2(100),
      INSTRUCTOR_IMAGE VARCHAR2(100)
   );


   TYPE t_course_tab IS TABLE OF t_course_rec INDEX BY PLS_INTEGER;
   courses t_course_tab;


   CURSOR cur_courses IS
      SELECT c.course_id AS COURSE_ID,
           c.course_name AS COURSE_NAME,
           c.course_description AS COURSE_DESCRIPTION,
           r.review AS REVIEW,
           r.rating AS RATING,
           i.first_name || ' ' || i.last_name AS INSTRUCTOR_NAME,
           i.IMAGE AS INSTRUCTOR_IMAGE
        FROM MCSC.Courses c
        JOIN MCSC.Category cat ON c.category = cat.category_id
        JOIN MCSC.USERS i ON i.USER_ID = c.INSTRUCTOR
```

```
     LEFT JOIN MCSC.RatingAndReviews r ON c.course_id = r.course_id
     WHERE cat.category_id = v1 AND c.status = 'Published';


  CURSOR cur_courses_except IS
    SELECT c.course_id AS COURSE_ID,
        c.course_name AS COURSE_NAME,
        c.course_description AS COURSE_DESCRIPTION,
        r.review AS REVIEW,
        r.rating AS RATING,
        i.first_name || ' ' || i.last_name AS INSTRUCTOR_NAME,
        i.IMAGE AS INSTRUCTOR_IMAGE
     FROM MCSC.Courses c
     JOIN MCSC.Category cat ON c.category = cat.category_id
     JOIN MCSC.USERS i ON i.USER_ID = c.INSTRUCTOR
     LEFT JOIN MCSC.RatingAndReviews r ON c.course_id = r.course_id
     WHERE cat.category_id <> v1 AND c.status = 'Published';
BEGIN
  IF f = 1 THEN
    OPEN cur_courses;
    FETCH cur_courses BULK COLLECT INTO courses;
    CLOSE cur_courses;
  ELSE
    OPEN cur_courses_except;
    FETCH cur_courses_except BULK COLLECT INTO courses;
    CLOSE cur_courses_except;
  END IF;


  FOR i IN 1 .. courses.COUNT LOOP
    DBMS_OUTPUT.PUT_LINE('Course ID: ' || courses(i).COURSE_ID);
    DBMS_OUTPUT.PUT_LINE('Course Name: ' || courses(i).COURSE_NAME);
    DBMS_OUTPUT.PUT_LINE('Course Description: ' || courses(i).COURSE_DESCRIPTION);
    DBMS_OUTPUT.PUT_LINE('Review: ' || (courses(i).REVIEW));
    DBMS_OUTPUT.PUT_LINE('Rating: ' || (courses(i).RATING));
    DBMS_OUTPUT.PUT_LINE('Instructor Name: ' || courses(i).INSTRUCTOR_NAME);
    DBMS_OUTPUT.PUT_LINE('Instructor Image: ' || courses(i).INSTRUCTOR_IMAGE);
    DBMS_OUTPUT.PUT_LINE('------------------------------');
  END LOOP;
END;
4.  DECLARE
  -- Cursor to fetch the most selling courses
  CURSOR course_cursor IS
    SELECT c.course_id, c.course_name, c.course_description, c.sold, c.thumbnail,
        i.first_name || ' ' || i.last_name AS instructor_name,
        i.image AS instructor_image,
        cat.name AS category_name, c.price, c.status, c.created_at, c.tag,
        c.instructions, c.points, c.what_you_will_learn,
        c.category AS category_id, c.instructor AS instructor_id
    FROM MCSC.Courses c
    JOIN MCSC.Users i ON c.instructor = i.user_id
    JOIN MCSC.Category cat ON c.category = cat.category_id
    WHERE c.status = 'Published'
    ORDER BY c.sold DESC
    FETCH FIRST :fetchAtmost ROWS ONLY;


  TYPE course_table IS TABLE OF course_cursor%ROWTYPE INDEX BY PLS_INTEGER;
  courses course_table;
  idx PLS_INTEGER := 1;


  -- Cursor to fetch ratings for a specific course
  CURSOR rating_cursor(p_course_id NUMBER) IS
    SELECT r.rating, r.review, u.user_id, u.first_name, u.last_name, u.account_type
    FROM MCSC.RatingAndReviews r
    JOIN MCSC.Users u ON r.user_id = u.user_id
    WHERE r.course_id = p_course_id;


  TYPE rating_table IS TABLE OF rating_cursor%ROWTYPE INDEX BY PLS_INTEGER;
BEGIN
  OPEN course_cursor;


  -- Fetch courses into the associative array
  LOOP
    FETCH course_cursor INTO courses(idx);
    EXIT WHEN course_cursor%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE('Course ID: ' || courses(idx).course_id);
    DBMS_OUTPUT.PUT_LINE('Course Name: ' || courses(idx).course_name);
    DBMS_OUTPUT.PUT_LINE('Course Description: ' || courses(idx).course_description);
    DBMS_OUTPUT.PUT_LINE('Instructor Name: ' || courses(idx).instructor_name);
    DBMS_OUTPUT.PUT_LINE('Instructor Image: ' || courses(idx).instructor_image);
    DBMS_OUTPUT.PUT_LINE('Category Name: ' || courses(idx).category_name);
    DBMS_OUTPUT.PUT_LINE('Price: ' || courses(idx).price);
    DBMS_OUTPUT.PUT_LINE('Status: ' || courses(idx).status);
    DBMS_OUTPUT.PUT_LINE('Created At: ' || courses(idx).created_at);
    DBMS_OUTPUT.PUT_LINE('Tag: ' || courses(idx).tag);
    DBMS_OUTPUT.PUT_LINE('Instructions: ' || courses(idx).instructions);
    DBMS_OUTPUT.PUT_LINE('Points: ' || courses(idx).points);
    DBMS_OUTPUT.PUT_LINE('What You Will Learn: ' || courses(idx).what_you_will_learn);
    DBMS_OUTPUT.PUT_LINE('Sold: ' || courses(idx).sold);
    DBMS_OUTPUT.PUT_LINE('Category ID: ' || courses(idx).category_id);
    DBMS_OUTPUT.PUT_LINE('Instructor ID: ' || courses(idx).instructor_id);


    -- Fetch ratings for the current course
    DECLARE
      ratings rating_table;
      r_idx PLS_INTEGER := 1;
    BEGIN
      OPEN rating_cursor(courses(idx).course_id);


      LOOP
        FETCH rating_cursor INTO ratings(r_idx);
        EXIT WHEN rating_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE('  Rating: ' || ratings(r_idx).rating);
        DBMS_OUTPUT.PUT_LINE('  Review: ' || ratings(r_idx).review);
        DBMS_OUTPUT.PUT_LINE('  User ID: ' || ratings(r_idx).user_id);
        DBMS_OUTPUT.PUT_LINE('  User Name: ' || ratings(r_idx).first_name || ' ' ||
ratings(r_idx).last_name);
        DBMS_OUTPUT.PUT_LINE('  Account Type: ' || ratings(r_idx).account_type);
        r_idx := r_idx + 1;
      END LOOP;


      CLOSE rating_cursor;
    END;


    idx := idx + 1;
  END LOOP;


  CLOSE course_cursor;
END;
5. DECLARE
  v_course_id MCSC.Courses.course_id%TYPE;
  v_instructor_exists NUMBER;
BEGIN
```

```
  -- Check if the instructor exists in the parent table
  SELECT COUNT(*) INTO v_instructor_exists
  FROM MCSC.Users
  WHERE user_id = :instructor;

  -- If instructor does not exist, raise an error
  IF v_instructor_exists = 0 THEN
    RAISE_APPLICATION_ERROR(-20001, 'Instructor ID does not exist in the parent table.');
  END IF;

  INSERT INTO MCSC.Courses (
    course_name,
    course_description,
    instructor,
    what_you_will_learn,
    PRICE,
    TAG,
    category,
    THUMBNAIL,
    status,
    INSTRUCTIONS
  ) VALUES (
    :COURSE_NAME,
    :COURSE_DESCRIPTION,
    :instructor,
    :WHAT_YOU_WILL_LEARN,
    :PRICE,
    :tag,
    :category,
    :THUMBNAIL,
    :status,
    :INSTRUCTIONS
  )
  RETURNING course_id INTO v_course_id;

  :newCourseId := v_course_id;
END;
```

6. DECLARE

```
        v_course_id NUMBER := :course_id;
    BEGIN
        -- Delete from RatingAndReviews
        DELETE FROM MCSC.RatingAndReviews WHERE course_id = v_course_id;

        -- Delete from CourseProgress
        DELETE FROM MCSC.CourseProgress WHERE course_id = v_course_id;

        -- Delete from SubSection
        DELETE FROM MCSC.SubSection WHERE course_id = v_course_id;

        -- Delete from Section
        DELETE FROM MCSC.Section WHERE course_id = v_course_id;

        -- Unenroll students from Course_StudentsEnrolled
        DELETE FROM MCSC.Course_StudentsEnrolled WHERE course_id = v_course_id;

        -- Finally, delete the course from Courses table
        DELETE FROM MCSC.Courses WHERE course_id = v_course_id;


        COMMIT;
      END;
```

7. DECLARE

```
        p_course_id NUMBER := :COURSE_ID;
        -- SECTION
        v_section_id MCSC.SECTION.SECTION_ID%TYPE;
        v_section_name MCSC.SECTION.SECTION_NAME%TYPE;
        v_subsection_id MCSC.SECTION.SUBSECTION_ID%TYPE;
        v_course_id MCSC.SECTION.COURSE_ID%TYPE;


        -- SUBSECTION
        v_title MCSC.SUBSECTION.TITLE%TYPE;
        V_TIME MCSC.SUBSECTION.TIME_DURATION%TYPE;
        V_DESC MCSC.SUBSECTION.DESCRIPTION%TYPE;
        V_VIDEO MCSC.SUBSECTION.VIDEO_URL%TYPE;


        -- Cursor to fetch sections
        CURSOR section_cursor IS
          SELECT SECTION_ID, SECTION_NAME, COURSE_ID, SUBSECTION_ID
          FROM MCSC.SECTION
          WHERE COURSE_ID = p_course_id;


        -- Cursor to fetch subsections for a specific section
        CURSOR subsection_cursor IS
          SELECT SUBSECTION_ID, TITLE, TIME_DURATION, DESCRIPTION, VIDEO_URL
          FROM MCSC.SUBSECTION
```

```
            WHERE SECTION_ID = v_section_id
            AND COURSE_ID = p_course_id;


      BEGIN
         -- Loop through each section
         OPEN section_cursor;
         LOOP
            FETCH section_cursor INTO v_section_id, v_section_name, v_course_id,
v_subsection_id;
            EXIT WHEN section_cursor%NOTFOUND;


            -- Output section details
            DBMS_OUTPUT.PUT_LINE('Section ID: ' || v_section_id || ' | Section Name: ' ||
v_section_name);


            -- Loop through each subsection within the section
            OPEN subsection_cursor;
            LOOP
               FETCH subsection_cursor INTO v_subsection_id, v_title, V_TIME, V_DESC,
V_VIDEO;
               EXIT WHEN subsection_cursor%NOTFOUND;


               -- Output subsection details
               DBMS_OUTPUT.PUT_LINE('  Subsection ID: ' || v_subsection_id || ' | Subsection
Name: ' || v_title || ' | Time Duration: ' || V_TIME || ' | Description: ' || V_DESC || ' | Video URL: ' ||
V_VIDEO);
            END LOOP;
            CLOSE subsection_cursor;


         END LOOP;
         CLOSE section_cursor;


      END;
```
8. DECLARE
```
      v_subsection_id NUMBER := :subsection_id;
      v_course_progress_id NUMBER;
      v_completed_videos VARCHAR2(4000);
   BEGIN
```

```
    -- Check if the subsection is valid
    SELECT subsection_id INTO v_subsection_id
    FROM MCSC.SubSection
    WHERE subsection_id = v_subsection_id;


    IF SQL%NOTFOUND THEN
        RAISE_APPLICATION_ERROR(-20001, 'Invalid subsection');
    END IF;
END;
```

9. CREATE OR REPLACE PROCEDURE GET_PROGRESS

```
(COURSE_ID IN NUMBER, studentId IN VARCHAR2, v_completed_videos OUT NUMBER,
 v_total_videos OUT NUMBER, v_completion_ratio OUT NUMBER)
AS
BEGIN
    SELECT COUNT(P.completed_videos) AS count_completed_videos,
      COUNT(S.video_url) total_videos,
    INTO v_completed_videos, v_total_videos
    FROM MCSC.COURSEPROGRESS P,
        MCSC.SUBSECTION S
    WHERE P.COURSE_ID = v_course_id
    AND P.USER_ID = v_user_id
    AND S.COURSE_ID = P.COURSE_ID
    GROUP BY P.completed_videos;


    -- Calculate the completion ratio
    IF v_total_videos = 0 THEN
        v_completion_ratio := 0;
    ELSE
        v_completion_ratio := (v_completed_videos / v_total_videos) * 100;
    END IF;
END;CREATE OR REPLACE PROCEDURE GET_PROGRESS
(COURSE_ID IN NUMBER, studentId IN VARCHAR2, v_completed_videos OUT NUMBER,
 v_total_videos OUT NUMBER, v_completion_ratio OUT NUMBER)
AS
BEGIN
    SELECT COUNT(P.completed_videos) AS count_completed_videos,
      COUNT(S.video_url) total_videos,
    INTO v_completed_videos, v_total_videos
    FROM MCSC.COURSEPROGRESS P,
        MCSC.SUBSECTION S
    WHERE P.COURSE_ID = v_course_id
    AND P.USER_ID = v_user_id
```

```
        AND S.COURSE_ID = P.COURSE_ID
        GROUP BY P.completed_videos;


        -- Calculate the completion ratio
        IF v_total_videos = 0 THEN
          v_completion_ratio := 0;
        ELSE
          v_completion_ratio := (v_completed_videos / v_total_videos) * 100;
        END IF;
    END;
```

10. DECLARE

```
    v_course_id NUMBER := :COURSE_ID;
    v_user_id VARCHAR2(20) := :studentId;
    v_completed_videos NUMBER;
    v_total_videos NUMBER;
    v_completion_ratio NUMBER;
BEGIN
    -- Fetch the count of completed videos and total videos
    SELECT COUNT(P.completed_videos) AS count_completed_videos,
           COUNT(S.video_url) total_videos,
    INTO v_completed_videos, v_total_videos
    FROM MCSC.COURSEPROGRESS P,
         MCSC.SUBSECTION S
    WHERE P.COURSE_ID = v_course_id
      AND P.USER_ID = v_user_id
      AND S.COURSE_ID = P.COURSE_ID
    GROUP BY P.completed_videos;


    -- Calculate the completion ratio
    IF v_total_videos = 0 THEN
      v_completion_ratio := 0;
    ELSE
      v_completion_ratio := (v_completed_videos / v_total_videos) * 100;
    END IF;


    -- Output the results
    --DBMS_OUTPUT.PUT_LINE('Completed Videos: ' || v_completed_videos);
    --DBMS_OUTPUT.PUT_LINE('Total Videos: ' || v_total_videos);
    --DBMS_OUTPUT.PUT_LINE('Completion Ratio: ' || v_completion_ratio || '%');
END;
```

11. CREATE OR REPLACE PROCEDURE create_payment (

```
    p_userid   IN  MCSC.ACCOUNT.USER_ID%TYPE,
    p_amount   IN  NUMBER,
    p_date     IN  VARCHAR2,
    p_courses  IN  VARCHAR2,
    p_message  OUT VARCHAR2,
    p_trxid    OUT VARCHAR2
)
IS
    l_balance     NUMBER;
    l_account_id  MCSC.ACCOUNT.ACCOUNT_ID%TYPE;
BEGIN
    -- Fetch account data
    BEGIN
        SELECT BALANCE, ACCOUNT_ID
        INTO l_balance, l_account_id
        FROM MCSC.ACCOUNT
        WHERE USER_ID = p_userid;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            -- Create account if not exists
            INSERT INTO MCSC.ACCOUNT (USER_ID, BALANCE)
            VALUES (p_userid, 100);
            COMMIT;
            -- Fetch the newly created account data
            SELECT BALANCE, ACCOUNT_ID
            INTO l_balance, l_account_id
            FROM MCSC.ACCOUNT
            WHERE USER_ID = p_userid;
    END;


    -- Check balance
    IF l_balance < p_amount THEN
        p_message := 'Insufficient Balance, current balance: ' || l_balance;
        RETURN;
    END IF;


    -- Deduct balance
    UPDATE MCSC.ACCOUNT
    SET BALANCE = BALANCE - p_amount
    WHERE USER_ID = p_userid;
    COMMIT;
    -- Generate a smaller TRXID (format: USERID_ACCOUNTID_YYYYMMDD)
    p_trxid := TO_CHAR(p_userid) || '_' || TO_CHAR(l_account_id) || '_' ||
REPLACE(SUBSTR(p_date, 1, 10), '-', '');


    -- Insert into TRX_HISTORY
    INSERT INTO MCSC.TRX_HISTORY (
        TRXID, USER_ID, ACCOUNT_ID, date_of_trx, amount_of_trx
    ) VALUES (
        p_trxid, p_userid, l_account_id,
        TO_TIMESTAMP(p_date, 'YYYY-MM-DD HH24:MI:SS'),
        p_amount
    );
    COMMIT;


    p_message := 'Transaction successful';
EXCEPTION
    WHEN OTHERS THEN
        p_message := 'Transaction failed: ' || SQLERRM;
        ROLLBACK;
END;
12. BEGIN
              create_payment(
                  p_userid   => :USER_ID,
                  p_amount   => :AMOUNT,
                  p_date     => :DATE,
                  p_courses  => :COURSES,
                  p_message  => :MESSAGE,
                  p_trxid    => :TRXID
              );
           END;`
```