

queries.sql

```
1  /* =====
2  DDL STATEMENTS - SINGLE PROGRAM
3  ===== */
4
5  /* CREATE TABLE */
6  CREATE TABLE student (
7      sid NUMBER,
8      sname VARCHAR2(20)
9  );
10
11 /* ALTER TABLE - Add a new column */
12 ALTER TABLE student
13 ADD age NUMBER;
14
15 /* INSERT RECORDS */
16 INSERT INTO student VALUES (1, 'Ravi', 20);
17 INSERT INTO student VALUES (2, 'Anu', 21);
18
19 /* DISPLAY RECORDS */
20 SELECT * FROM student;
21
22 /* TRUNCATE TABLE - Remove all records */
23 TRUNCATE TABLE student;
24
25 /* DROP TABLE - Delete table completely */
26 DROP TABLE student;
```

Output:

SID	SNAME	AGE
1	Ravi	20
2	Anu	21

DROP TABLE student  
\*

ERROR at line 1:  
ORA-00942: table or view "sandbox\_user"."STUDENT" does not exist

OneCompiler

Q ⚙ 🔔 Pricing Learn

queries.sql44c5d4q6x

```
1 - /* =====
2 | PARENT TABLE
3 | ===== */
4
5 - CREATE TABLE department (
6 |   dept_id NUMBER PRIMARY KEY,      -- Primary Key
7 |   dept_name VARCHAR2(20) UNIQUE    -- Unique constraint
8 | );
9
10 - /* =====
11 | CHILD TABLE
12 | ===== */
13
14 - CREATE TABLE employee (
15 |   emp_id NUMBER PRIMARY KEY,      -- Primary Key
16 |   emp_name VARCHAR2(20) NOT NULL,  -- NOT NULL constraint
17 |   age NUMBER CHECK (age >= 18),    -- CHECK constraint
18 |   salary NUMBER CHECK (salary > 5000), -- CHECK constraint
19 |   dept_id NUMBER,
20 |   CONSTRAINT emp_dept_fk FOREIGN KEY (dept_id)
21 |   REFERENCES department(dept_id)  -- Foreign Key
22 | );
23
24 - /* =====
25 | INSERT VALUES
26 | ===== */
27
28 INSERT INTO department VALUES (1, 'IT');
29 INSERT INTO department VALUES (2, 'CSE');
30
31 INSERT INTO employee VALUES (101, 'Ravi', 22, 10000, 1);
32 INSERT INTO employee VALUES (102, 'Anu', 25, 12000, 2);
33
34 - /* =====
35 | IS (NULL) CONDITION
36 | ===== */
37
38 SELECT * FROM employee
39 WHERE dept_id IS NOT NULL;
```

Output:

138 ms

EMP_ID	EMP_NAME	AGE	SALARY	DEPT_ID
101	Ravi	22	10000	1
102	Anu	25	12000	2

EMP_ID	EMP_NAME	AGE	SALARY	DEPT_ID
101	Ravi	22	10000	1
102	Anu	25	12000	2

queries.sql

44c5d4q6x [🔗](#)

```
1  /* Create table */
2  CREATE TABLE employee (
3      emp_id NUMBER,
4      emp_name VARCHAR2(20),
5      salary NUMBER,
6      doj DATE
7  );
8
9  /* Insert values */
10 INSERT INTO employee VALUES (1, 'ravi', 12000, SYSDATE);
11 INSERT INTO employee VALUES (2, 'anu', 15000, SYSDATE);
12 INSERT INTO employee VALUES (3, 'kumar', 10000, SYSDATE);
13
14 /* STRING FUNCTIONS */
15 SELECT UPPER(emp_name) FROM employee;
16 SELECT LENGTH(emp_name) FROM employee;
17
18 /* NUMBER FUNCTIONS */
19 SELECT MAX(salary), MIN(salary) FROM employee;
20 SELECT ROUND(AVG(salary), 2) FROM employee;
21
22 /* DATE FUNCTIONS */
23 SELECT SYSDATE FROM dual;
24 SELECT ADD_MONTHS(doj, 6) FROM employee;
25
26 /* AGGREGATE FUNCTION */
27 SELECT COUNT(emp_id) FROM employee;
```

Output:

UPPER(EMP\_NAME)

-----

RAVI

ANU

KUMAR

LENGTH(EMP\_NAME)

-----

4

3

5

MAX(SALARY) MIN(SALARY)

-----

15000

10000

ROUND(AVG(SALARY), 2)

-----

12333.33

SYSDATE

-----

30-JAN-26

ADD\_MONTHS(DOJ, 6)

-----

30-JUL-26

30-JUL-26

30-JUL-26

COUNT(EMP\_ID)

-----

3

COUNT(EMP\_ID)

-----

3

```

1  /* =====
2  CREATE TABLES
3  ===== */
4
5  CREATE TABLE department (
6      dept_id NUMBER,
7      dept_name VARCHAR2(20)
8  );
9
10 CREATE TABLE employee (
11     emp_id NUMBER,
12     emp_name VARCHAR2(20),
13     dept_id NUMBER
14 );
15
16 /* =====
17 INSERT VALUES
18 ===== */
19
20 INSERT INTO department VALUES (1, 'IT');
21 INSERT INTO department VALUES (2, 'CSE');
22 INSERT INTO department VALUES (3, 'ECE');
23
24 INSERT INTO employee VALUES (101, 'Ravi', 1);
25 INSERT INTO employee VALUES (102, 'Anu', 2);
26 INSERT INTO employee VALUES (103, 'Kumar', NULL);
27
28 /* =====
29 INNER JOIN
30 ===== */
31
32 SELECT e.emp_name, d.dept_name
33 FROM employee e
34 INNER JOIN department d
35 ON e.dept_id = d.dept_id;
36
37 /* =====
38 LEFT OUTER JOIN
39 ===== */
40
41 SELECT e.emp_name, d.dept_name
42 FROM employee e
43 LEFT OUTER JOIN department d
44 ON e.dept_id = d.dept_id;
45
46 /* =====
47 RIGHT OUTER JOIN
48 ===== */
49
50 SELECT e.emp_name, d.dept_name
51 FROM employee e
52 RIGHT OUTER JOIN department d
53 ON e.dept_id = d.dept_id;
54
55 /* =====
56 FULL OUTER JOIN
57 ===== */
58
59 SELECT e.emp_name, d.dept_name
60 FROM employee e
61 FULL OUTER JOIN department d
62 ON e.dept_id = d.dept_id;
63
64 /* =====
65 CROSS JOIN
66 ===== */
67
68 SELECT e.emp_name, d.dept_name
69 FROM employee e
70 CROSS JOIN department d;
71

```

Output:

150 ms

EMP_NAME	DEPT_NAME
Ravi	IT
Anu	CSE

EMP_NAME	DEPT_NAME
Ravi	IT
Anu	CSE
Kumar	

EMP_NAME	DEPT_NAME
Ravi	IT
Anu	CSE
	ECE

EMP_NAME	DEPT_NAME
Ravi	IT
Anu	CSE
	ECE
Kumar	

EMP_NAME	DEPT_NAME
Ravi	IT
Ravi	CSE
Ravi	ECE
Anu	IT
Anu	CSE
Anu	ECE
Kumar	IT
Kumar	CSE
Kumar	ECE

EMP_NAME	DEPT_NAME
Ravi	IT
Ravi	CSE
Ravi	ECE
Anu	IT
Anu	CSE
Anu	ECE
Kumar	IT
Kumar	CSE
Kumar	ECE

```

1  /* =====
2  CREATE TABLE
3  ===== */
4
5  CREATE TABLE employee (
6      emp_id NUMBER,
7      emp_name VARCHAR2(20),
8      salary NUMBER,
9      dept_id NUMBER
10 );
11
12 /* =====
13 INSERT VALUES
14 ===== */
15
16 INSERT INTO employee VALUES (1, 'Ravi', 12000, 1);
17 INSERT INTO employee VALUES (2, 'Anu', 15000, 2);
18 INSERT INTO employee VALUES (3, 'Kumar', 10000, 1);
19 INSERT INTO employee VALUES (4, 'Meena', 18000, 3);
20
21 /* =====
22 SUBQUERY EXAMPLES
23 ===== */
24
25 /* 1. Employee with salary greater than average salary */
26 SELECT emp_name
27 FROM employee
28 WHERE salary > (
29     SELECT AVG(salary) FROM employee
30 );
31
32 /* 2. Employees working in the same department as Ravi */
33 SELECT emp_name
34 FROM employee
35 WHERE dept_id = (
36     SELECT dept_id
37     FROM employee
38     WHERE emp_name = 'Ravi'
39 );
40
41 /* 3. Employee with maximum salary */
42 SELECT emp_name
43 FROM employee
44 WHERE salary = (
45     SELECT MAX(salary) FROM employee
46 );

```

Output:

EMP\_NAME

-----

Anu

Meena

EMP\_NAME

-----

Ravi

Kumar

EMP\_NAME

-----

Meena

EMP\_NAME

-----

Meena

2:53



0.65 KB/S VoLTE 4G 48

PL/SQL Online Editor...  
onecompiler.com

OneCompiler

queries.sql 44bzyy3kx

```
1 DECLARE
2     num1 NUMBER := 10;
3     num2 NUMBER := 20;
4
5 BEGIN
6     IF num1 < num2 THEN
7         DBMS_OUTPUT.PUT_LINE('num1 is smaller than num2');
8     ELSIF num1 = num2 THEN
9         DBMS_OUTPUT.PUT_LINE('num1 and num2 are equal');
10    ELSE
11        DBMS_OUTPUT.PUT_LINE('num2 is smaller than num1');
12    END IF;
13    DBMS_OUTPUT.PUT_LINE('after else if ladder');
14 END;
15 /
```

STDIN  
Input for the program ( Optional )

Output: 102 ms  
num1 is smaller than num2  
after else if ladder

```
1 DECLARE
2     a NUMBER := 1;
3 BEGIN
4     DBMS_OUTPUT.PUT_LINE('Program started');
5
6     LOOP
7         DBMS_OUTPUT.PUT_LINE(a);
8         a := a + 1;
9
10        EXIT WHEN a > 5;
11    END LOOP;
12
13    DBMS_OUTPUT.PUT_LINE('Program completed');
14 END;
15 /
```

STDIN

Input for the program ( Optional )

Output:

113 ms

Program started

1

2

3

4

5

Program completed