**Creating tables**

* CREATE TABLE customers (id INTEGER PRIMARY KEY, name TEXT, age INTEGER, weight REAL);

[Many data types](https://www.khanacademy.org/computer-programming/sql-create-table-with-multiple-data-types/5945065256124416)

* CREATE TABLE customers (id INTEGER PRIMARY KEY, age INTEGER);

[Using primary keys](https://www.khanacademy.org/computer-programming/sql-create-table-with-a-primary-key/5189331400654848)

See also: [specifying defaults](https://www.khanacademy.org/computer-programming/sql-specifying-default-values-in-create-table/6155515281408000), [using foreign keys](https://www.khanacademy.org/computer-programming/sql-create-table-with-foreign-key-reference/4700108318965760). For more details, see the following: [SQLite reference for CREATE](https://www.sqlite.org/lang_createtable.html).

**Inserting data**

* INSERT INTO customers VALUES (73, "Brian", 33);

[Inserting data](https://www.khanacademy.org/computer-programming/sql-inserting-values-in-tables/5382515271532544)

* INSERT INTO customers (name, age) VALUES ("Brian", 33);

[Inserting data for named columns](https://www.khanacademy.org/computer-programming/sql-inserting-values-in-tables/5382515271532544)

See also: [The SQLite reference for INSERT](https://www.sqlite.org/lang_insert.html).

**Querying data**

* SELECT \* FROM customers;

[Select everything](https://www.khanacademy.org/computer-programming/sql-selecting-rows/5163767537205248)

* SELECT \* FROM customers WHERE age > 21;

[Filter with condition](https://www.khanacademy.org/computer-programming/sql-selecting-with-where-conditions/6216095996444672)

* SELECT \* FROM customers WHERE age < 21 AND state = "NY";

[Filter with multiple conditions](https://www.khanacademy.org/computer-programming/sql-selecting-with-where-conditions/6216095996444672)

* SELECT \* FROM customers WHERE plan IN ("free", "basic");

[Filter with IN](https://www.khanacademy.org/computer-programming/sql-filter-with-in/6529475600842752)

* SELECT name, age FROM customers;

[Select specific columns](https://www.khanacademy.org/computer-programming/sql-selecting-rows/5163767537205248)

* SELECT \* FROM customers WHERE age > 21 ORDER BY age DESC;

[Order results](https://www.khanacademy.org/computer-programming/sql-select-with-order-by/6218182226477056)

* SELECT name, CASE WHEN age > 18 THEN "adult" ELSE "minor" END "type" FROM customers;

[Transform with CASE](https://www.khanacademy.org/computer-programming/sql-transform-select-results-with-case/5100246984163328)

See also: [filtering with LIKE](https://www.khanacademy.org/computer-programming/sql-filtering-with-like/5913941557510144), [restricting with LIMIT](https://www.khanacademy.org/computer-programming/sql-using-select-with-limit/6005320342175744), [using ROUND and other core functions](https://www.khanacademy.org/computer-programming/sql-round-and-other-core-functions/6203890437586944). For more details, see: [the SQLite reference for SELECT](https://www.sqlite.org/lang_select.html).

**Aggregating data**

* SELECT MAX(age) FROM customers;

[Aggregate functions](https://www.khanacademy.org/computer-programming/sql-select-with-aggregate-functions/4797964233080832)

* SELECT gender, COUNT(\*) FROM students GROUP BY gender;

[Grouping data](https://www.khanacademy.org/computer-programming/sql-grouping-select-results-with-group-by/5520132919132160)

See also: [restricting results with HAVING](https://www.khanacademy.org/computer-programming/sql-using-having-with-grouped-by/6199455011438592).

**Joining related tables**

* SELECT customers.name, orders.item FROM customers JOIN orders ON customers.id = orders.customer\_id;

[Inner join](https://www.khanacademy.org/computer-programming/sql-join-on-tables/5409956539006976)

* SELECT customers.name, orders.item FROM customers LEFT OUTER JOIN orders ON customers.id = orders.customer\_id;

[Outer join](https://www.khanacademy.org/computer-programming/sql-join-on-tables/5409956539006976)

**Updating and deleting data**

* UPDATE customers SET age = 33 WHERE id = 73;

[Updating data](https://www.khanacademy.org/computer-programming/sql-update-and-delete/5559819222253568)

* DELETE FROM customers WHERE id = 73;

[Deleting data](https://www.khanacademy.org/computer-programming/sql-update-and-delete/5559819222253568)

Also see: [ALTER TABLE](https://www.khanacademy.org/computer-programming/sql-alter-table/6147136605519872).

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

CASE SYNTAX

SELECT type, heart\_rate,

CASE

WHEN heart\_rate > 220-30 THEN "above max"

WHEN heart\_rate > ROUND(0.90 \* (220-30)) THEN "above target"

WHEN heart\_rate > ROUND(0.50 \* (220-30)) THEN "within target"

ELSE "below target"

END as "hr\_zone"

FROM exercise\_logs;

SELECT COUNT(\*),

CASE

WHEN heart\_rate > 220-30 THEN "above max"

WHEN heart\_rate > ROUND(0.90 \* (220-30)) THEN "above target"

WHEN heart\_rate > ROUND(0.50 \* (220-30)) THEN "within target"

ELSE "below target"

END as "hr\_zone"

FROM exercise\_logs

GROUP BY hr\_zone;

JOIN SYNTAX

SELECT students.first\_name, students.last\_name, students.email, student\_grades.test, student\_grades.grade

FROM students

JOIN student\_grades

ON students.id = student\_grades.student\_id

WHERE grade > 90;

//left outer join

SELECT students.first\_name, students.last\_name, student\_projects.title

FROM students // table form which you want all values

LEFT OUTER JOIN student\_projects // table which has null values

ON students.id = student\_projects.student\_id;

/\* self join \*/

SELECT students.first\_name, students.last\_name, buddies.email as buddy\_email

FROM students

JOIN students buddies

ON students.buddy\_id = buddies.id;

Graphical user interface, text, application

Description automatically generated

Graphical user interface, table

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application, Word

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

2. Create a PL/SQL block that selects the maximum department\_id in the departments table and stores it in the v\_max\_deptno variable. Display the maximum department\_id. Declare v\_max\_deptno to be the same datatype as the department\_id column. Include a SELECT statement to retrieve the highest department\_id from the departments table. Display the variable v\_max\_deptno.

**DECLARE**

**v\_max\_deptno departments.department\_id%TYPE;**

**BEGIN**

**SELECT MAX(department\_id) INTO v\_max\_deptno FROM departments;**

**DBMS\_OUTPUT.PUT\_LINE('Max id: ' || v\_max\_deptno);**

**END;**

v\_emp\_lname employees.last\_name%TYPE

v\_balance NUMBER(7,2);

v\_min\_balance v\_balance%TYPE :=1000;

SET SERVEROUTPUT ON

DECLARE

today DATE :=SYSDATE;

tomorrow today%TYPE;

BEGIN

tomorrow :=SYSDATE +1;

DBMS\_OUTPUT.PUT\_LINE('Hello World');

DBMS\_OUTPUT.PUT\_LINE('Today is' || to\_char(today,'DD.MM.YY'));

DBMS\_OUTPUT.PUT\_LINE('Tomorrow is' ||to\_char(tomorrow,'DD.MM.YY'));

END;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

DECLARE

customer VARCHAR2(50) := 'Womansport';

credit\_rating VARCHAR2(50) := 'EXCELLENT';

BEGIN

DECLARE

customer NUMBER(7) := 201;

c\_name VARCHAR2(25) := 'Unisports';

credit\_rating := ‘GOOD’;

END;

DECLARE

max\_deptno NUMBER(8,2);

BEGIN

SELECT max(department\_id)

INTO max\_deptno

FROM departments;

DBMS\_OUTPUT.PUT\_LINE('The maximum department\_id is:' || ‘ ‘ || v\_max\_deptno);

END;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

DECLARE  
v\_rows\_deleted VARCHAR2(30);  
v\_empno employees.employee\_id%TYPE := 176;  
BEGIN  
DELETE FROM employees  
WHERE employee\_id = v\_empno;  
v\_rows\_deleted := (SQL%ROWCOUNT ||  
' row deleted.');  
DBMS\_OUTPUT.PUT\_LINE (v\_rows\_deleted);  
END;

DECLARE

v\_dept\_id number(8);

max\_deptno NUMBER(8,2);

v\_dept\_name departments.department\_name%TYPE := ‘Education’;

v\_rows\_changed varchar2(20);

BEGIN

SELECT max(department\_id)

INTO max\_deptno

FROM departments;

DBMS\_OUTPUT.PUT\_LINE('The maximum department\_id is:' || ‘ ‘ || v\_max\_deptno);

v\_dept\_id number(8):=v\_max\_deptno + 10;

INSERT INTO employees

(department\_name, department\_id, location\_id)

VALUES(v\_dept\_name, v\_dept\_id, NULL);

v\_rows\_changed := (SQL%ROWCOUNT ||

' row changed.');

DBMS\_OUTPUT.PUT\_LINE (v\_rows\_deleted);

END;

write a basic loop to insert records into test1 until id becomes 5 (inclusive) , name is always Test

1 Test

2 Test

3 Test

4 Test

5 Test

CREATE TABLE test1

(id number,

name char(50)

)

DECLARE

v\_iter number :=1;

BEGIN

LOOP

INSERT INTO test1 VALUES (v\_iter, 'Test');

v\_iter := v\_iter +1;

EXIT WHEN v\_iter > 5;

END LOOP;

END;

\*\*\*USING WHILE LOOP\*\*\*\*

DECLARE

v\_iter number :=1;

BEGIN

WHILE v\_iter <=5 LOOP

INSERT INTO test1 VALUES (v\_iter, 'Test');

v\_iter := v\_iter +1;

END LOOP;

END;

Text

Description automatically generated with medium confidence

DECLARE

v\_dept\_name departments.department\_name%TYPE := ‘Education’;

BEGIN

DECLARE

max\_deptno NUMBER(8,2);

BEGIN

SELECT max(department\_id)

INTO max\_deptno

FROM departments;

DBMS\_OUTPUT.PUT\_LINE('The maximum department\_id is:' || ‘ ‘ || v\_max\_deptno);

END;

END;

CREATE TABLE messages

(RESULTS NUMBER)

\*\*\*\*\*IF LOOP\*\*\*\*\*\*\*\*\*\*\*

DECLARE

v\_grade CHAR(1) := UPPER('&grade');

v\_appraisal VARCHAR2(20);

BEGIN

IF v\_grade = 'A' THEN v\_appraisal :='Excellent';

ELSIF v\_grade = 'B' THEN v\_appraisal :='Very Good';

ELSIF v\_grade='C' THEN v\_appraisal :='Good';

ELSE v\_appraisal :='No Such grade';

END IF;

LOOP

execution

calculation

EXIT a> 100;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE ('Grade: '|| v\_grade || ' Appraisal ' || v\_appraisal);

END;

\*\*\*\*COMPOSITE VARIABLE\*\*\*\*

Composite variable  
v\_emprecord employees%ROWTYPE;  
SELECT \* INTO v\_emprecord FROM employees WHERE emp\_id=3  
v\_emprecord.salary := 6000;  
DBMS\_OUTPUT.PUT\_LINE (v\_emprecord.fname);

Composite Variable  
record ( employees%ROWTYPE , department\_name char(30) )  
v\_myrecord record

CREATE TABLE retired\_emps  
(empno number(7),  
ename varchar2(50),  
job varchar2(10),  
mgr number(6),  
hiredate date,  
leavedate date,  
sal number (9,2),  
comm number (2,2),  
deptno number(5))

DECLARE

v\_employee\_number number:= 124;

v\_emp\_rec employees%ROWTYPE;

BEGIN

SELECT \* INTO v\_emp\_rec FROM employees

WHERE employee\_id = v\_employee\_number;

INSERT INTO retired\_emps(empno, ename, job, mgr,

hiredate, leavedate, sal, comm, deptno)

VALUES (v\_emp\_rec.employee\_id, v\_emp\_rec.last\_name,

v\_emp\_rec.job\_id, v\_emp\_rec.manager\_id,

v\_emp\_rec.hire\_date, SYSDATE,

v\_emp\_rec.salary, v\_emp\_rec.commission\_pct,

v\_emp\_rec.department\_id);

END;

\*\*\*\*countries\*\*\*\*\*\*\*\*\* Basic Loop: Example

DECLARE  
v\_countryid locations.country\_id%TYPE := 'CA';  
v\_loc\_id locations.location\_id%TYPE;  
v\_counter NUMBER(2) := 1;  
v\_new\_city locations.city%TYPE := 'Montreal';  
BEGIN  
SELECT MAX(location\_id) INTO v\_loc\_id FROM locations  
WHERE country\_id = v\_countryid;  
LOOP  
INSERT INTO locations(location\_id, city, country\_id)  
VALUES((v\_loc\_id + v\_counter), v\_new\_city, v\_countryid);  
v\_counter := v\_counter + 1;  
EXIT WHEN v\_counter > 3;  
END LOOP;  
END;  
/

Using INDEX BY Table Methods

DECLARE  
TYPE elname\_table\_type IS TABLE OF  
employees.last\_name%TYPE  
INDEX BY PLS\_INTEGER;  
cva\_elname elname\_table\_type;  
BEGIN  
FOR i IN 100..105 LOOP  
SELECT last\_name INTO cva\_elname (i) FROM employees WHERE employee\_id = i ;  
END LOOP;  
FOR i IN 100..105 LOOP  
DBMS\_OUTPUT.PUT\_LINE (cva\_elname(i));  
END LOOP;  
END;  
/