**Extensions**

Manually convert extension codes AWS\_ORACLE\_EXT and ORAFCE from Amazon Relational Database Service (Amazon RDS) for Oracle to Amazon RDS for PostgreSQL or Amazon Aurora PostgreSQL-Compatible Edition (psql) by using native built-in codes shown in the following tables.

**AWS\_ORACLE\_EXT – AWS Oracle Extension**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **From (Extension code)** | **To (psql built-in)** |
| 1 | AWS\_ORACLE\_EXT.SYSDATE() | NOW() |
| 2 | AWS\_ORACLE\_EXT.TO\_CHAR() | TO\_CHAR() |
| 3 | AWS\_ORACLE\_EXT.TO\_NUMBER() | TO\_NUMBER() |
| 4 | AWS\_ORACLE\_EXT.SYSTIMESTAMP() | CURRENT\_TIMESTAMP |
| 5 | AWS\_ORACLE\_EXT.CURRENT\_DATE() | LOCALTIMESTAMP(0) |
| 6 | AWS\_ORACLE\_EXT.ADD\_MONTHS() | MAKE\_INTERVAL() |
| 7 | AWS\_ORACLE\_EXT.LAST\_DAY() | DATE\_TRUNC() |
| 8 | AWS\_ORACLE\_EXT.MONTHS\_BETWEEN() | AGE(), EXTRACT() |
| 9 | AWS\_ORACLE\_EXT.SESSIONTIMEZONE() | COALESCE(CURRENT\_SETTING('TIMEZONE', TRUE), 'UTC') |
| 10 | AWS\_ORACLE\_EXT.TO\_DATE() | TIMESTAMP(0) |
| 11 | AWS\_ORACLE\_EXT.SUBSTR | SUBSTR() |
| 12 | AWS\_ORACLE\_EXT.INSTR | POSITION() |
| 13 | AWS\_ORACLE\_EXT.DBMS\_RANDOM | RANDOM() |
| 14 | AWS\_ORACLE\_EXT.TRUNC | DATE\_TRUNC() |
| 15 | AWS\_ORACLE\_EXT.REGEXP\_SUBSTR | UNNEST(REGEXP\_MATCHES()) |
| 16 | AWS\_ORACLE\_EXT.REGEXP\_REPLACE | REGEXP\_REPLACE |
| 17 | AWS\_ORACLE\_EXT.SET\_PACKAGE\_VARIABLE | SET\_CONFIG() |
| 18 | AWS\_ORACLE\_EXT.GET\_PACKAGE\_VARIABLE | CURRENT\_SETTING() |
| 19 | AWS\_ORACLE\_EXT.SET\_PACKAGE\_VARIABLE\_RECORD\_FIELD | SET\_CONFIG() |
| 20 | AWS\_ORACLE\_EXT.GET\_PACKAGE\_VARIABLE\_RECORD\_FIELD | CURRENT\_SETTING() |
| 21 | AWS\_ORACLE\_EXT.SYS\_CONTEXT | Assign |
| 22 | AWS\_ORACLE\_EXT.DBMS\_SQL | EXECUTE |
| 23 | AWS\_ORACLE\_EXT.ARRAY | Array |
| 24 | AWS\_ORACLE\_EXT.DBMS\_JOB | Table, AWS Batch, AWS CloudWatch |

The following examples show the AWS Schema Conversion Tool (AWS SCT) aws\_oracle\_ext extensions for Oracle with their equivalent PostgreSQL default or native built-in code along with the result.

1. **AWS\_ORACLE\_EXT.SYSDATE**

|  |  |
| --- | --- |
| Extension code | select AWS\_ORACLE\_EXT.SYSDATE(); |
| Built-in code | select NOW(); |
| Result | "2020-05-27 11:43:23" |

1. **AWS\_ORACLE\_EXT.TO\_CHAR**

|  |  |
| --- | --- |
| Extension code | select AWS\_ORACLE\_EXT.TO\_CHAR(now(),'YYYYMMDD'); |
| Built-in code | select TO\_CHAR(now(),'YYYYMMDD'); |
| Result | "20200527" |

1. **AWS\_ORACLE\_EXT.TO\_NUMBER**

|  |  |
| --- | --- |
| Extension code | select AWS\_ORACLE\_EXT.TO\_NUMBER('123'); |
| Built-in code | select TO\_NUMBER('123','999'); |
| Result | 123 |

1. **AWS\_ORACLE\_EXT.SYSTIMESTAMP**

|  |  |
| --- | --- |
| Extension code | select AWS\_ORACLE\_EXT.SYSTIMESTAMP(); |
| Built-in code | select CURRENT\_TIMESTAMP; |
| Result | "2020-05-29 11:52:34.154406+00" |

1. **AWS\_ORACLE\_EXT.CURRENT\_DATE**

|  |  |
| --- | --- |
| Extension code | select AWS\_ORACLE\_EXT.CURRENT\_DATE (); |
| Built-in code | select LOCALTIMESTAMP; |
| Result | "2020-05-29 11:52:34" |

1. **AWS\_ORACLE\_EXT.ADD\_MONTHS**

|  |  |
| --- | --- |
| Extension code | select AWS\_ORACLE\_EXT.ADD\_MONTHS (LOCALTIMESTAMP(0),1); |
| Built-in code | select (LOCALTIMESTAMP(0) + MAKE\_INTERVAL(months=>1)); |
| Result | "2020-07-01 09:39:04" |

1. **AWS\_ORACLE\_EXT.LAST\_DAY**

|  |  |
| --- | --- |
| Extension code | select AWS\_ORACLE\_EXT.LAST\_DAY (CURRENT\_TIMESTAMP::date); |
| Built-in code | SELECT (DATE\_TRUNC('month', CURRENT\_TIMESTAMP::date) + '1 month'::INTERVAL - '1 day'::INTERVAL)::date; |
| Result | "2020-06-30" |

1. **AWS\_ORACLE\_EXT.MONTHS\_BETWEEN**

|  |  |
| --- | --- |
| Extension code | select AWS\_ORACLE\_EXT.MONTHS\_BETWEEN ('31-DEC-2021', '01-JAN-2020'); |
| Built-in code | SELECT EXTRACT(YEAR FROM AGE) \* 12 + EXTRACT(MONTH FROM AGE) + (EXTRACT(DAYS FROM AGE)/31) FROM AGE ('31-DEC-2021', '01-JAN-2020'); |
| Result | 23.9677419354839 |

1. **AWS\_ORACLE\_EXT.SESSIONTIMEZONE**

|  |  |
| --- | --- |
| Extension code | select AWS\_ORACLE\_EXT.SESSIONTIMEZONE(); |
| Built-in code | select COALESCE(CURRENT\_SETTING('TIMEZONE', TRUE), 'UTC'); |
| Result | "UTC" |

1. **AWS\_ORACLE\_EXT.TO\_DATE**

|  |  |
| --- | --- |
| Extension code | select AWS\_ORACLE\_EXT.TO\_DATE('01-JUN-2020'); |
| Built-in code | select '01-JUN-2020'::TIMESTAMP(0); |
| Result | "2020-06-01 00:00:00" |

1. **AWS\_ORACLE\_EXT.SUBSTR**

|  |  |
| --- | --- |
| Extension code | select AWS\_ORACLE\_EXT.SUBSTR('abc def',1,5); |
| Built-in code | select SUBSTR('abc def',1,5); |
| Result | "abc d" |

1. **AWS\_ORACLE\_EXT.INSTR**

|  |  |
| --- | --- |
| Extension code | select AWS\_ORACLE\_EXT.INSTR('abc def',' '); |
| Built-in code | select POSITION(' ' in 'abc def'); |
| Result | 4 |

1. **AWS\_ORACLE\_EXT.DBMS\_RANDOM**

|  |  |
| --- | --- |
| Extension code | select AWS\_ORACLE\_EXT.DBMS\_RANDOM$VALUE(2, 5); |
| Built-in code | select RANDOM() \* (5-2) + 2; |
| Result | 4.63098958646879 & 4.29649342782795 |

1. **AWS\_ORACLE\_EXT.TRUNC**

|  |  |
| --- | --- |
| Extension code | select AWS\_ORACLE\_EXT.TRUNC('2020-05-27'::timestamp,'MM'); |
| Built-in code | select DATE\_TRUNC('Month','2020-05-27'::timestamp); |
| Result | "2020-05-01 00:00:00" |

1. **AWS\_ORACLE\_EXT.REGEXP\_SUBSTR**

|  |  |
| --- | --- |
| Extension code | select AWS\_ORACLE\_EXT.REGEXP\_SUBSTR('hello.world', '[^,.]+'); |
| Built-in code | select UNNEST(REGEXP\_MATCHES('hello.world', '[^,.]+')); |
| Result | "hello" |

1. **AWS\_ORACLE\_EXT.REGEXP\_REPLACE**

|  |  |
| --- | --- |
| Extension code | select AWS\_ORACLE\_EXT.REGEXP\_REPLACE('This is Test DB', 'Test', 'Prod'); |
| Built-in code | select REGEXP\_REPLACE('This is Test DB', 'Test', 'Prod'); |
| Result | "This is Prod DB" |

1. **AWS\_ORACLE\_EXT.SET\_PACKAGE\_VARIABLE**

|  |  |
| --- | --- |
| Extension code | perform AWS\_ORACLE\_EXT.SET\_PACKAGE\_VARIABLE('schema', 'pkg', 'variable', 'value'); |
| Built-in code | perform SET\_CONFIG('schema.pkg.variable', 'value'::TEXT, false); |
| Result | "value" |

1. **AWS\_ORACLE\_EXT.GET\_PACKAGE\_VARIABLE**

|  |  |
| --- | --- |
| Extension code | select AWS\_ORACLE\_EXT.GET\_PACKAGE\_VARIABLE(‘schema’, ‘pkg’, 'variable’)::INTEGER |
| Built-in code | select CURRENT\_SETTING('schema.pkg.variable')::INTEGER |
| Result | "value" |

1. **AWS\_ORACLE\_EXT.SET\_PACKAGE\_VARIABLE\_RECORD\_FIELD**

|  |  |
| --- | --- |
| Extension code | perform AWS\_ORACLE\_EXT.SET\_PACKAGE\_VARIABLE\_RECORD\_FIELD ('schema', 'pkg', 'variable', 'record', ‘value’); |
| Built-in code | perform SET\_CONFIG ('schema.pkg.variable.record', ‘value’::TEXT, false) |
| Result | "value" |

1. **AWS\_ORACLE\_EXT.GET\_PACKAGE\_VARIABLE\_RECORD\_FIELD**

|  |  |
| --- | --- |
| Extension code | select AWS\_ORACLE\_EXT.GET\_PACKAGE\_VARIABLE\_RECORD\_FIELD (‘schema’, ‘pkg’, 'variable’, ‘record’)::INTEGER |
| Built-in code | select CURRENT\_SETTING('schema.pkg.variable.record')::INTEGER |
| Result | "value" |

1. **AWS\_ORACLE\_EXT.SYS\_CONTEXT**

The following examples include the relevant Oracle code, which is converted using the extension and its built-in code.

|  |  |
| --- | --- |
| Oracle code | * SYS\_CONTEXT('USERENV', 'OS\_USER') * SYS\_CONTEXT('USERENV', 'SESSION\_USER') * SYS\_CONTEXT('USERENV', 'PROXY\_USER') * SYS\_CONTEXT('USERENV','CLIENT\_IDENTIFIER') |
| Extension code | * AWS\_ORACLE\_EXT.SYS\_CONTEXT('USERENV', 'OS\_USER') * AWS\_ORACLE\_EXT.SYS\_CONTEXT('USERENV', 'SESSION\_USER') * AWS\_ORACLE\_EXT.SYS\_CONTEXT('USERENV', 'PROXY\_USER') * AWS\_ORACLE\_EXT.SYS\_CONTEXT('USERENV', 'CLIENT\_IDENTIFIER') |
| Built-in code | * return NULL; * return SESSION\_USER::character varying; * return NULL; * return CURRENT\_SETTING ('CLIENT.IDENTIFIER', true); |

1. **AWS\_ORACLE\_EXT.DBMS\_SQL**

|  |  |
| --- | --- |
| Oracle code | **create or replace procedure dept\_ins** (p\_dept\_no integer default 123)  as  v\_sql varchar2(500);  c integer;  v\_procedure varchar2(30);  v\_count integer;  begin  v\_sql := 'insert into dept(deptno) values (:dept\_no)';  c := dbms\_sql.open\_cursor;  dbms\_sql.parse(c, v\_sql, dbms\_sql.NATIVE);  dbms\_sql.bind\_variable(c, ':dept\_no', p\_dept\_no);  v\_count := dbms\_sql.execute(c);  dbms\_sql.close\_cursor(c);  end; |
| Extension code | **CREATE or REPLACE PROCEDURE orcl.dept\_ins** (IN p\_dept\_no integer default 123)  AS $BODY$  DECLARE  v\_sql CHARACTER VARYING(500);  c NUMERIC(38);  v\_count NUMERIC(38);  BEGIN  v\_sql := CONCAT\_WS('', 'insert into orcl.dept(deptno) values (:dept\_no)');  c := aws\_oracle\_ext.dbms\_sql$open\_cursor();  PERFORM aws\_oracle\_ext.dbms\_sql$parse(c::integer, v\_sql::text);  PERFORM aws\_oracle\_ext.dbms\_sql$bind\_variable((c)::INTEGER, (':dept\_no')::VARCHAR, p\_dept\_no);  v\_count := aws\_oracle\_ext.dbms\_sql$execute((c)::INTEGER);  c := aws\_oracle\_ext.dbms\_sql$close\_cursor((c)::NUMERIC);  END;  $BODY$  LANGUAGE plpgsql; |
| Built-in code | **CREATE OR REPLACE PROCEDURE orcl.dept\_ins** (p\_dept\_no integer DEFAULT 123)  AS $BODY$  DECLARE  v\_sql CHARACTER VARYING(500);  BEGIN  v\_sql := CONCAT\_WS('', 'insert into orcl.dept(deptno) values (', p\_dept\_no, ')');  execute v\_sql;  END;  $BODY$  LANGUAGE 'plpgsql'; |

1. **AWS\_ORACLE\_EXT.ARRAY**

|  |  |
| --- | --- |
| Oracle code | SUBTYPE dept\_row\_typ IS dept%ROWTYPE;  TYPE t\_dept\_row\_tab IS TABLE OF dept\_row\_typ INDEX BY BINARY\_INTEGER;  TYPE my\_ref\_cur\_typ IS REF CURSOR;  **CREATE OR REPLACE FUNCTION dept\_rec** ()  RETURN t\_dept\_row\_tab  IS  v\_ref\_cur my\_ref\_cur\_typ;  v\_sql VARCHAR2 (1000);  i PLS\_INTEGER := 0;  v\_rec dept\_row\_typ;  v\_tab t\_dept\_row\_tab;  BEGIN  v\_sql := 'SELECT \* FROM DEPT ';  OPEN v\_ref\_cur  FOR v\_sql;  LOOP  FETCH v\_ref\_cur INTO v\_rec;  EXIT WHEN v\_ref\_cur%NOTFOUND;  i := i + 1;  v\_tab (i) := v\_rec;  END LOOP;  CLOSE v\_ref\_cur;  RETURN v\_tab;  END dept\_rec; |
| Extension code | CREATE TYPE orcl.my\_pkg$dept\_row\_typ AS (...);  **CREATE OR REPLACE FUNCTION orcl.my\_pkg$dept\_rec**(  IN p\_array$name VARCHAR,  IN p\_array$function\_name VARCHAR)  RETURNS VARCHAR  AS  $BODY$  DECLARE  aws\_oracle\_ext$array\_id$temporary BIGINT;  v\_ref\_cur REFCURSOR;  v\_sql CHARACTER VARYING(1000);  i INTEGER := 0;  v\_rec orcl.my\_pkg$dept\_row\_typ;  v\_tab VARCHAR(100) := 'v\_tab';  BEGIN  aws\_oracle\_ext$array\_id$temporary := aws\_oracle\_ext.array$create\_array(p\_array\_name => 'v\_tab', p\_procedure\_name => 'orcl.my\_pkg$dept\_rec');  PERFORM aws\_oracle\_ext.array$add\_fields\_to\_array(p\_array\_id => aws\_oracle\_ext$array\_id$temporary, p\_fields => '[{"":"orcl.dept"}]');  PERFORM aws\_oracle\_ext.array$create\_storage\_table(p\_array\_name => 'v\_tab', p\_procedure\_name => 'orcl.my\_pkg$dept\_rec', p\_cast\_type\_name => 'orcl.my\_pkg$dept\_row\_typ', pWithData => FALSE);  v\_sql := 'SELECT \* FROM orcl.dept ';  OPEN v\_ref\_cur FOR  EXECUTE v\_sql;  LOOP  FETCH v\_ref\_cur INTO v\_rec;  EXIT WHEN (NOT FOUND);  i := i + 1;  PERFORM aws\_oracle\_ext.array$set\_value(p\_array\_path => 'v\_tab[' || i || ']', p\_procedure\_name => 'orcl.my\_pkg$dept\_rec', p\_value => v\_rec);  END LOOP;  CLOSE v\_ref\_cur;  PERFORM aws\_oracle\_ext.array$assign('v\_tab', 'orcl.my\_pkg$dept\_rec', p\_array$name, p\_array$function\_name);  PERFORM aws\_oracle\_ext.array$clear\_procedure('orcl.my\_pkg$dept\_rec');  RETURN p\_array$name;  END;  $BODY$  LANGUAGE plpgsql; |
| Built-in code | CREATE TYPE orcl.my\_pkg$dept\_row\_typ AS (...);  **CREATE OR REPLACE FUNCTION orcl.my\_pkg$dept\_rec**()  RETURNS orcl.my\_pkg$dept\_row\_typ[]  AS  $BODY$  DECLARE  v\_ref\_cur REFCURSOR;  v\_sql CHARACTER VARYING(1000);  i INTEGER := 0;  v\_rec orcl.my\_pkg$dept\_row\_typ;  v\_tab orcl.my\_pkg$dept\_row\_typ[];  BEGIN  v\_sql := 'SELECT \* FROM orcl.dept ';  OPEN v\_ref\_cur FOR  EXECUTE v\_sql;  LOOP  FETCH v\_ref\_cur INTO v\_rec;  EXIT WHEN (NOT FOUND);  i := i + 1;  v\_tab[i] = v\_rec;  END LOOP;  CLOSE v\_ref\_cur;  RETURN v\_tab;  END;  $BODY$  LANGUAGE plpgsql; |

1. **AWS\_ORACLE\_EXT.DBMS\_JOB**

|  |  |
| --- | --- |
| Oracle code | DBMS\_JOB.SUBMIT(job=>v\_job, what=> 'BEGIN DL\_Load.refresh\_group('''||p\_group\_name||''','||p\_refresh\_pid||'); END;'); |
| Extension code | AWS\_ORACLE\_EXT.DBMS\_JOB$SUBMIT(what := CONCAT\_WS('', 'BEGIN DL\_Load.refresh\_group(''', p\_group\_name, ''',', p\_refresh\_pid, '); END;')) |
| Built-in code | * CREATE TABLE orcl.jobs (   job bigint NOT NULL,  name character varying(256),  description character varying(2000),  log\_user character varying(128) DEFAULT SESSION\_USER,  priv\_user character varying(128) DEFAULT CURRENT\_USER,  schema\_user character varying(128) DEFAULT CURRENT\_SCHEMA,  search\_path character varying DEFAULT array\_to\_string(current\_schemas(true), ','::text),  what character varying NOT NULL,  last\_date timestamp with time zone,  this\_date timestamp with time zone,  next\_date timestamp with time zone NOT NULL,  "interval" character varying NOT NULL,  total\_time numeric DEFAULT 0,  broken character varying(1) DEFAULT 'N'::character varying,  failures integer DEFAULT 0,  last\_failure\_message text,  status text,  CONSTRAINT c\_jobs\_broken CHECK (((broken)::text = ANY ((ARRAY['Y'::character varying, 'N'::character varying])::text[])))  );   * INSERT INTO orcl.jobs(what, next\_date, "interval") VALUES (CONCAT\_WS('', 'BEGIN CALL orcl.dl\_load$refresh\_group(''', p\_group\_name, ''',', p\_refresh\_pid, '); END;'), NOW(), 'interval'); * Configure AWS Batch and CloudWatch to trigger this job |

**ORAFCE - Oracle Functionality (en)**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **From (Extension code)** | **To (psql built-in)** |
| 1 | ORACLE.SYSDATE | NOW() |
| 2 | ORACLE.TO\_CHAR() | TO\_CHAR() |
| 3 | ORACLE.LENGTH() | LENGTH() |
| 4 | ORACLE.SUBSTR() | SUBSTR() |
| 5 | ORACLE.TO\_DATE() | TO\_DATE() |
| 6 | ORACLE.BTRIM() / RTRIM() / LTRIM() | TRIM() / BTRIM() / RTRIM() / LTRIM() |
| 7 | ORACLE.LPAD() / RPAD() | LPAD() / RPAD() |
| 8 | NVL() | COALESCE() |
| 9 | DECODE() | CASE |
| 10 | DBMS\_OUTPUT.PUT\_LINE | RAISE NOTICE |

The following examples show the orafce codes for Oracle with the equivalent Amazon RDS for Oracle and Amazon RDS for PostgreSQL or Aurora PostgreSQL-Compatible native built-in codes, along with their results.

1. **ORACLE.SYSDATE**

|  |  |
| --- | --- |
| Oracle code | SYSDATE |
| orafce code | ORACLE.SYSDATE() |
| Built-in code | NOW() |
| Result | "2020-06-01 18:39:44.634759+00" |

1. **ORACLE.TO\_CHAR**

|  |  |
| --- | --- |
| Oracle code | TO\_CHAR(SYSDATE,'MM/DD/YYYY') |
| orafce code | ORACLE.TO\_CHAR(NOW(), 'MM/DD/YYYY') |
| Built-in code | TO\_CHAR(NOW(),'MM/DD/YYYY') |
| Result | "06/01/2020" |

1. **ORACLE.LENGTH**

|  |  |
| --- | --- |
| Oracle code | LENGTH('abc def') |
| orafce code | ORACLE.LENGTH('abc def') |
| Built-in code | LENGTH('abc def') |
| Result | 7 |

1. **ORACLE.SUBSTR()**

|  |  |
| --- | --- |
| Oracle code | SUBSTR('abc def',1,5) |
| orafce code | ORACLE.SUBSTR('abc def',1,5) |
| Built-in code | SUBSTR('abc def',1,5) |
| Result | "abc d" |

1. **ORACLE.TO\_DATE()**

|  |  |
| --- | --- |
| Oracle code | TO\_DATE('01-JAN-2020','DD-MON-YYYY') |
| orafce code | ORACLE.TO\_DATE('01-JAN-2020','DD-MON-YYYY') |
| Built-in code | TO\_DATE('01-JAN-2020','DD-MON-YYYY') |
| Result | "2020-01-01" |

1. **ORACLE.BTRIM() / LTRIM() / RTRIM()**

|  |  |
| --- | --- |
| Oracle code | * LTRIM(' abc def ') * RTRIM(' abc def ') * TRIM(' abc def ') |
| orafce code | * ORACLE.LTRIM(' abc def ') * ORACLE.RTRIM(' abc def ') * ORACLE.BTRIM(' abc def ') |
| Built-in code | * LTRIM(' abc def ') * RTRIM(' abc def ') * TRIM(' abc def ') / BTRIM(' abc def ') |
| Result | * "abc def " * " abc def" * "abc def" |

1. **ORACLE.LPAD() / RPAD()**

|  |  |
| --- | --- |
| Oracle code | * RPAD('abc def',10,'\*') * LPAD('abc def',10,'\*') |
| orafce code | * ORACLE.RPAD('abc def',10,'\*') * ORACLE.LPAD('abc def',10,'\*') |
| Built-in code | * RPAD('abc def',10,'\*') * LPAD('abc def',10,'\*') |
| Result | * "abc def\*\*\*" * "\*\*\*abc def" |

1. **NVL()**

|  |  |
| --- | --- |
| Oracle code | NVL (null,'null value') |
| orafce code | NVL (null,'null value') |
| Built-in code | COALESCE (null,'null value') |
| Result | "null value" |

1. **DECODE()**

|  |  |
| --- | --- |
| Oracle code | DECODE('abc','abcd','found','not-found') |
| orafce code | DECODE('abc','abcd','found','not-found') |
| Built-in code | CASE WHEN 'abc'='abcd' THEN 'found' ELSE 'not-found' END |
| Result | "not-found" |

1. **DBMS\_OUTPUT.PUT\_LINE**

|  |  |
| --- | --- |
| Oracle code | DBMS\_OUTPUT.PUT\_LINE('DBMS Output'); |
| orafce code | PERFORM DBMS\_OUTPUT.PUT\_LINE('DBMS Output'); |
| Built-in code | RAISE NOTICE 'DBMS Output'; |
| Result | "DBMS Output" |

There are a few more extensions. However, the extensions listed previously are for the most commonly used Oracle codes that get converted using extensions and need to be migrated to PostgreSQL default or native built-in codes.

**PERFORMANCE COMPARISON**

**Performance-comparison check**

The major reason to avoid using extensions with AWS SCT is the slow performance in executing and fetching the results. The cost can be understood from the following [PostgreSQL Explain Plan](https://www.postgresql.org/docs/current/sql-explain.html) (execution plan of a statement) on Oracle’s SYSDATE function migration to PostgreSQL’s NOW() function between all three codes (aws\_oracle\_ext, orafce, and psql default).

**aws\_oracle\_ext (AWS SCT extension)**

|  |
| --- |
| psql=> Explain Analyze Select AWS\_ORACLE\_EXT.SYSDATE(); |
| "Result (cost=0.00..0.26 rows=1 width=8) (actual time=0.169..0.170 rows=1 loops=1)" |
| "Planning Time: 0.111 ms" |
| "Execution Time: 0.180 ms" |

**orafce**

|  |
| --- |
| psql=> Explain Analyze Select ORACLE.SYSDATE(); |
| "Result (cost=0.00..0.01 rows=1 width=8) (actual time=0.005..0.005 rows=1 loops=1)" |
| "Planning Time: 0.016 ms" |
| "Execution Time: 0.019 ms" |

**PostgreSQL psql (pg\_catalog) default built-in**

|  |
| --- |
| psql=> Explain Analyze Select NOW()::TIMESTAMP(0); |
| "Result (cost=0.00..0.02 rows=1 width=8) (actual time=0.005..0.005 rows=1 loops=1)" |
| "Planning Time: 0.021 ms" |
| "Execution Time: 0.018 ms" |

**Performance-comparison result**

The previous execution-cost results conclude that you should avoid using the aws\_oracle\_ext and orafce extensions for Oracle with AWS SCT. However, for a very few uses or package calls, this approach might be needed.

The cost-comparison results between orafce and native psql are very minor and usually almost the same. Hence, if orafce is already configured and the Amazon RDS for Oracle schema is already created in your Amazon RDS for PostgreSQL or Aurora PostgreSQL-Compatible database, you can use orafce if code is already migrated using orafce and your requirement is already satisfied through this migration.