RELATIONAL DATABASES

OBJECTIVES

- INSERT STATEMENT USING THE VALUES CLAUSE
- INSERT SPECIAL VALUES, NULL VALUES AND DATE VALUES
- INSERT TO COPY ROWS FROM ONE TABLE TO ANOTHER

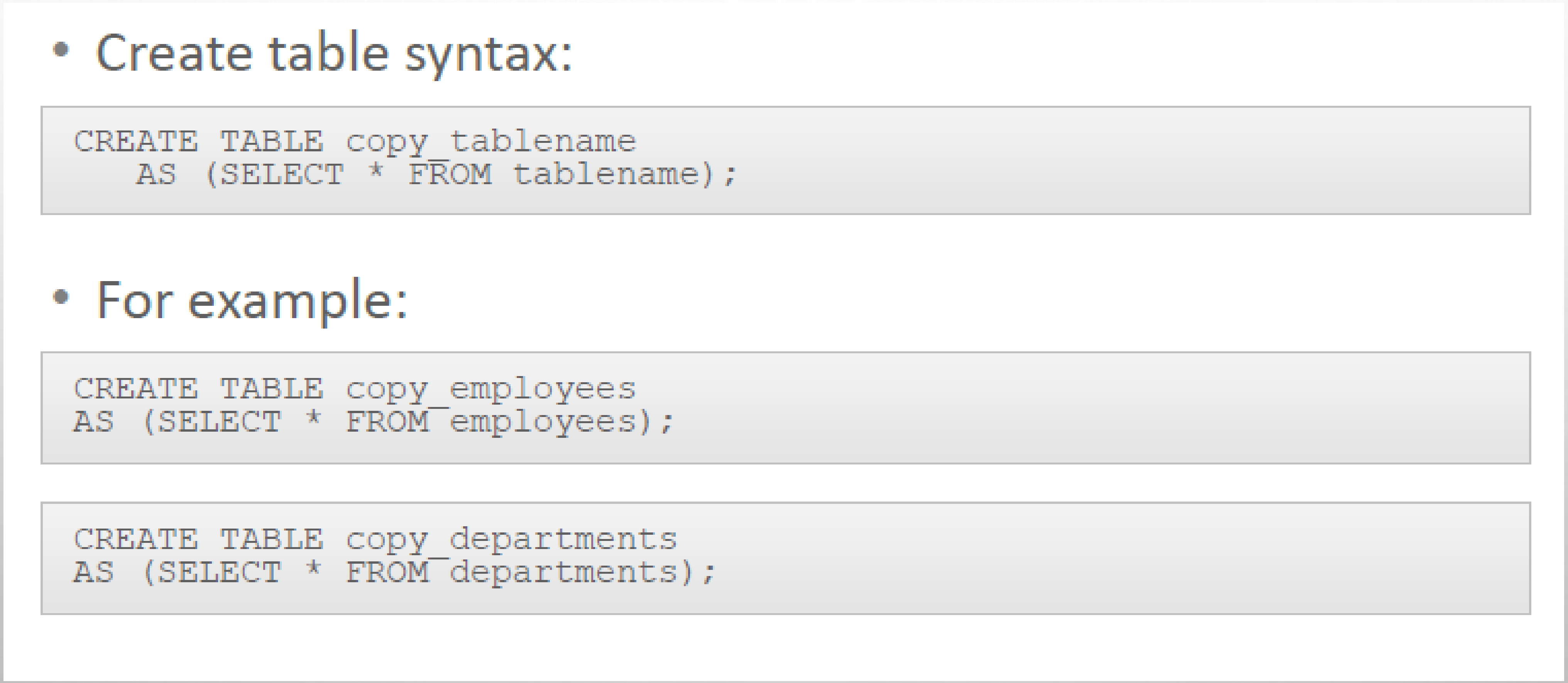
PURPOSE

- MAKING CHANGES TO THE DATABASE
- DATABASES ARE DYNAMIC
- CONSTANTLY INSERTING UPDATING AND DELETING DATA.

COPY TABLES BEFORE INSERTING

- TO KEEP YOUR SCHEMA TABLES IN THEIR ORIGINAL STATE, YOU WILL MAKE A COPY OF EACH TABLE BEFORE COMPLETING THE PRACTICE ACTIVITIES.
- YOU WILL USE THE COPY OF THE TABLE THAT YOU CREATE NOT THE ORIGINAL TO INSERT DATA.
- IF YOU MAKE CHANGES IN ERROR THEN YOU CAN USE THE ORIGINAL TABLE TO RESTORE THE COPY.
- YOU SHOULD NAME EACH COPIED TABLE: COPY_TABLENAME.
- THE COPIED TABLE DOES NOT INHERIT THE PRIMARY KEY TO FOREIGN KEY INTEGRITY RULES OF THE ORIGINAL TABLE.
- THE COLUMN DATA TYPES ARE INHERITED IN THE COPIED TABLE.

SYNTAX TO CREATE A COPY OF A TABLE



- Create table syntax:

```
CREATE TABLE copy_tablename
   AS (SELECT * FROM tablename);
```

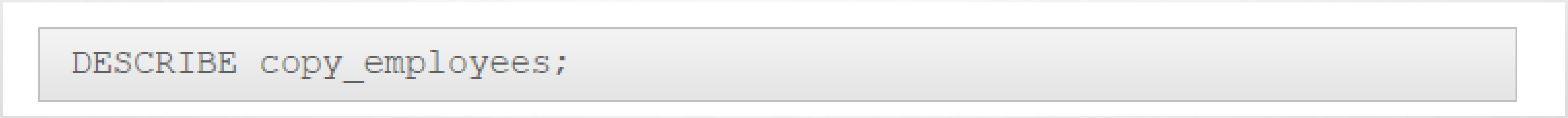
- For example:

```
CREATE TABLE copy_employees
AS (SELECT * FROM employees);
```

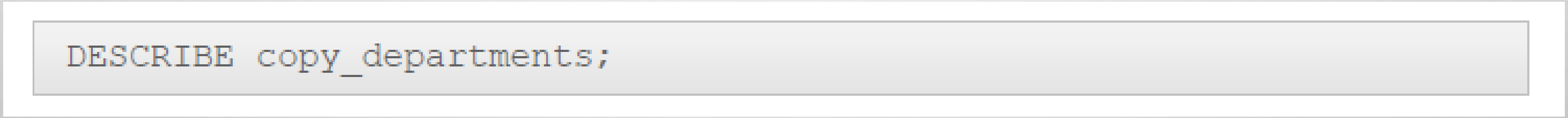
```
CREATE TABLE copy_departments
AS (SELECT * FROM departments);
```

SYNTAX TO CREATE A COPY OF A TABLE

- TO VERIFY AND VIEW THE COPY OF THE TABLE YOU DESCRIBE IT.



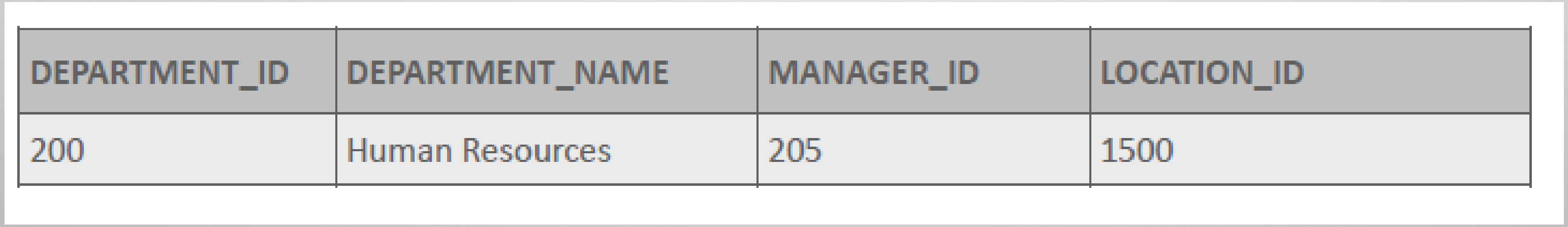
```
DESCRIBE copy_employees;
```



```
DESCRIBE copy_departments;
```

INSERT

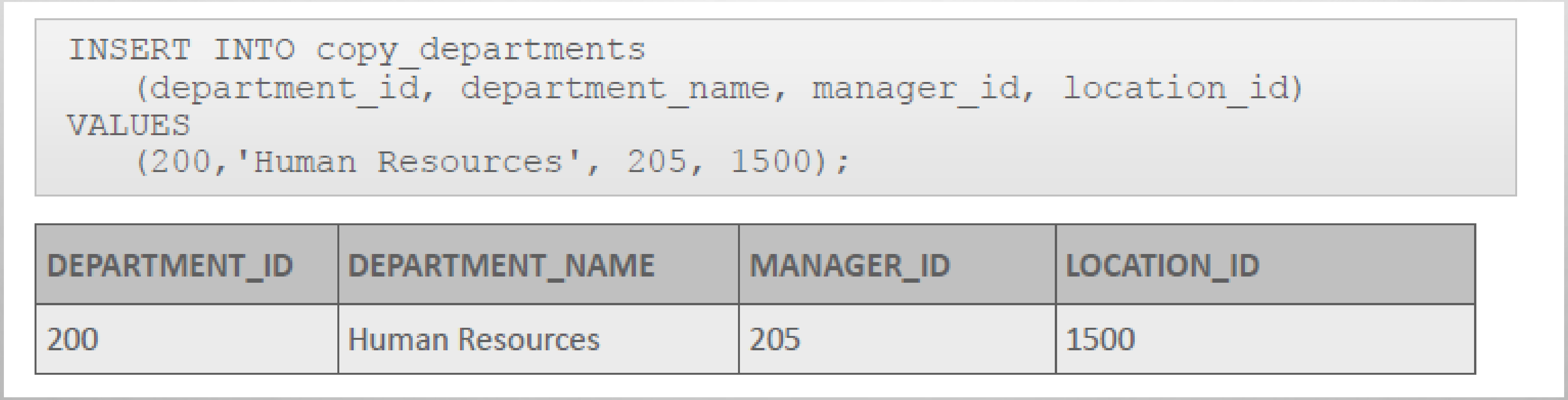
- THE INSERT STATEMENT IS USED TO ADD A NEW ROW TO A TABLE.
- THE STATEMENT REQUIRES THREE VALUES:
  - THE NAME OF THE TABLE
  - THE NAMES OF THE COLUMNS IN THE TABLE TO POPULATE
  - CORRESPONDING VALUES FOR EACH COLUMN
- WE WANT TO INSERT THE DATA BELOW IN A COPY OF THE DEPARTMENT TABLE:



DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
200	Human Resources	205	1500

INSERT

- THE SYNTAX USES INSERT TO ADD A NEW DEPARTMENT TO THE COPY DEPARTMENT TABLE.
- THIS STATEMENT EXPLICITLY LISTS EACH COLUMN AS IT APPEARS IN THE TABLE.
- THE VALUES FOR EACH COLUMN ARE LISTED IN THE SAME ORDER.
- NOTE THAT THE NUMBER VALUES ARE NOT ENCLOSED IN QUOTES.

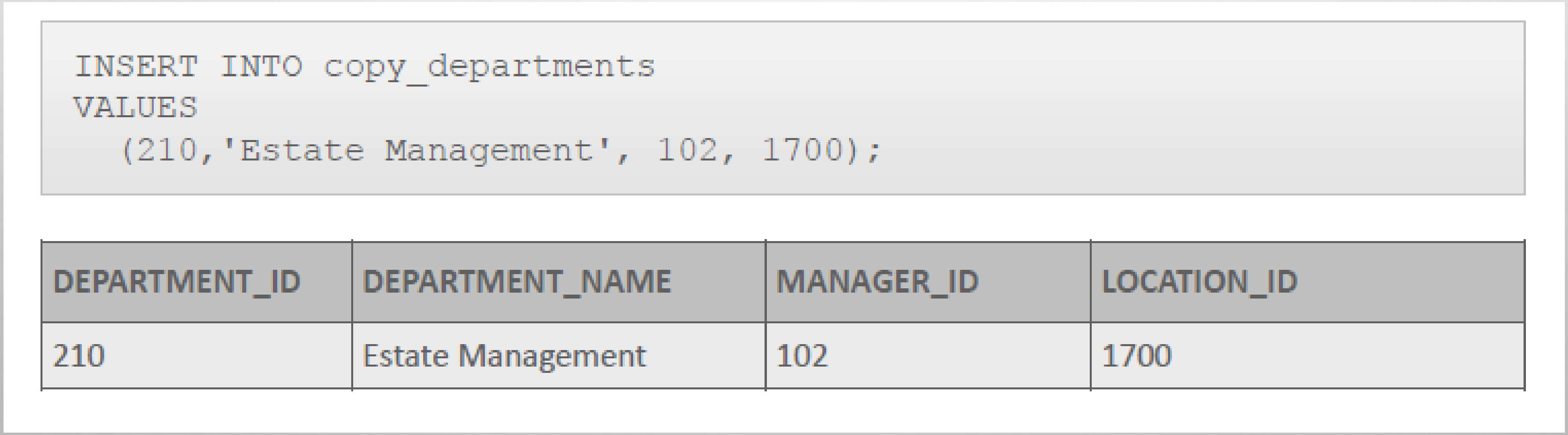


```
INSERT INTO copy_departments
   (department_id, department_name, manager_id, location_id)
VALUES
   (200,'Human Resources', 205, 1500);
```

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
200	Human Resources	205	1500

INSERT

- ANOTHER WAY TO INSERT VALUES IS TO IMPLICITLY ADD THEM BY OMITTING THE COLUMN NAMES.
- ONE PRECAUTION: THE VALUES FOR EACH COLUMN MUST MATCH EXACTLY THE DEFAULT ORDER IN WHICH THEY APPEAR IN THE TABLE AND A VALUE MUST BE PROVIDED FOR EACH COLUMN

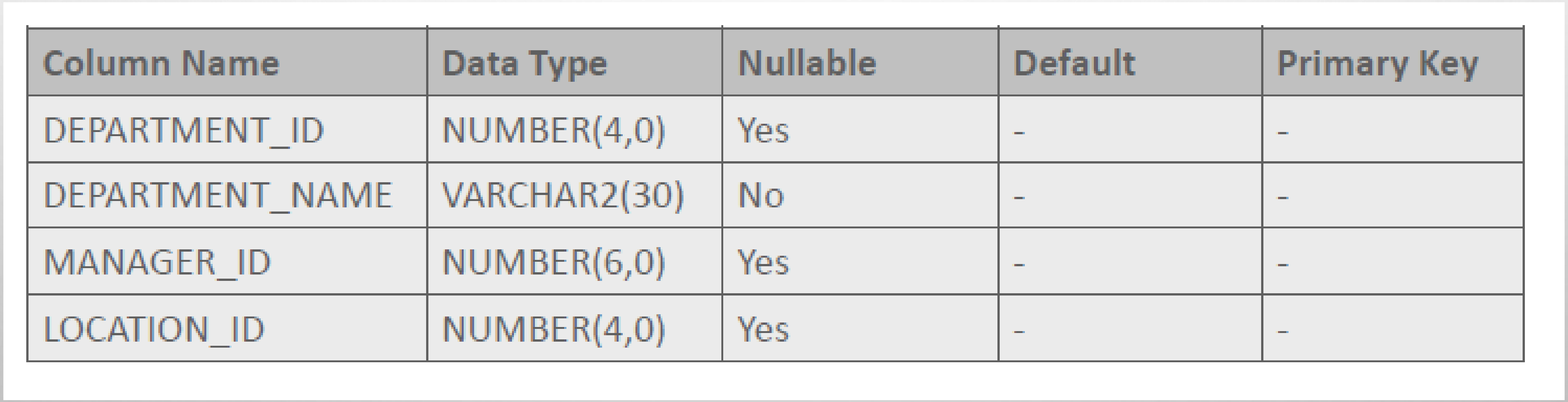


```
INSERT INTO copy_departments
VALUES
  (210,'Estate Management', 102, 1700);
```

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
210	Estate Management	102	1700

CHECK THE TABLE FIRST

- BEFORE INSERTING DATA INTO A TABLE, YOU MUST CHECK SEVERAL TABLE DETAILS.
- THE DESCRIBE TABLENAME STATEMENT WILL RETURN A DESCRIPTION OF THE TABLE STRUCTURE.

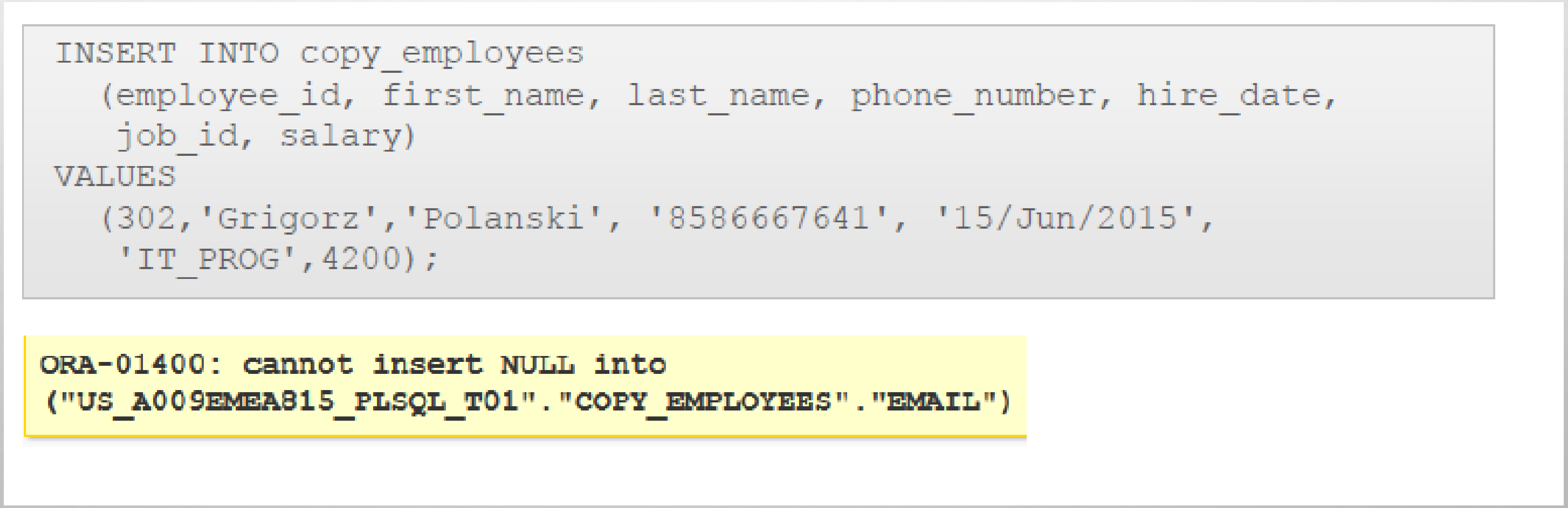


Column Name	Data Type	Nullable	Default	Primary Key
DEPARTMENT_ID	NUMBER(4,0)	Yes	-	-
DEPARTMENT_NAME	VARCHAR2(30)	No	-	-
MANAGER_ID	NUMBER(6,0)	Yes	-	-
LOCATION_ID	NUMBER(4,0)	Yes	-	-

- NUMBER SPECIFIES THE PRECISION AND SCALE. PRECISION IS THE TOTAL NUMBER OF DIGITS, SCALE IS THE NUMBER OF DIGITS TO THE RIGHT OF THE DECIMAL PLACE.

INSERTING ROWS WITH NULL VALUES

- THE INSERT STATEMENT NEED NOT SPECIFY EVERY COLUMN.
- IF EVERY COLUMN THAT REQUIRES A VALUE IS ASSIGNED A VALUE THEN THE INSERT WILL WORK.
- AN IMPLICIT INSERT WILL AUTOMATICALLY INSERT A NULL VALUE IN COLUMNS THAT ALLOW NULLS.
- TO EXPLICITLY ADD A NULL VALUE TO A COLUMN THAT ALLOWS NULLS, USE THE NULL KEYWORD IN THE VALUES LIST.

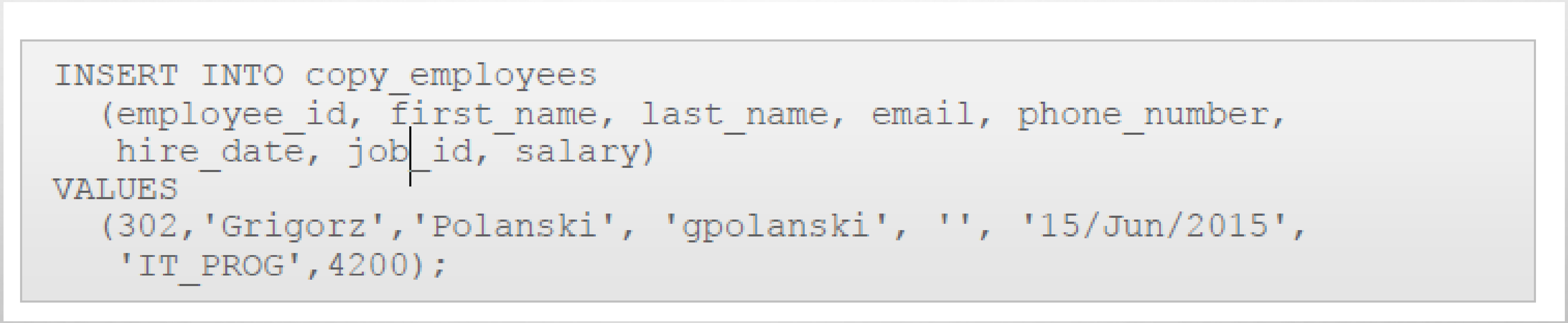


```
INSERT INTO copy_employees
  (employee_id, first_name, last_name, phone_number, hire_date,
   job_id, salary)
VALUES
  (302,'Grigorz','Polanski', '8586667641', '15/Jun/2015',
   'IT_PROG',4200);
```

```
ORA-01400: cannot insert NULL into
("US_A009EMEA815_PLSQL_T01"."COPY_EMPLOYEES"."EMAIL")
```

INSERTING EMPTY STRINGS

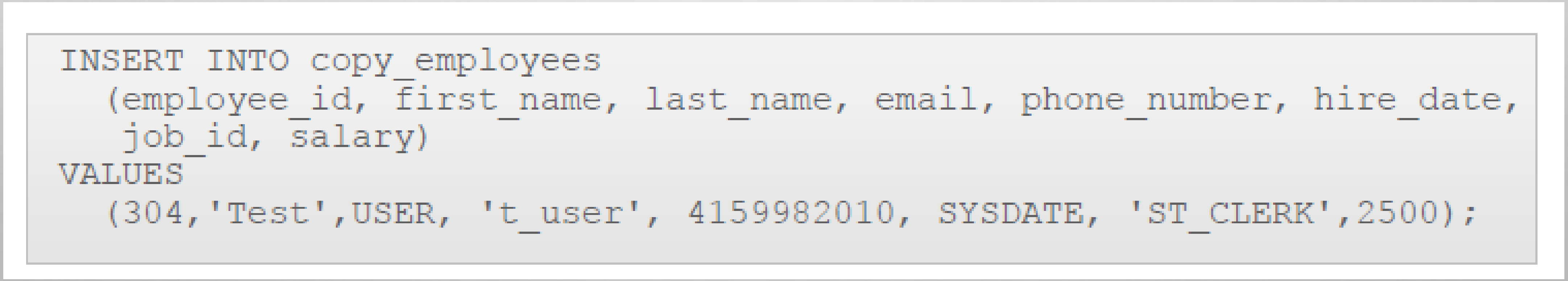
- TO SPECIFY EMPTY STRINGS USE EMPTY SINGLE QUOTATION MARKS FOR MISSING DATA.



```
INSERT INTO copy_employees
  (employee_id, first_name, last_name, email, phone_number,
   hire_date, job_id, salary)
VALUES
  (302,'Grigorz','Polanski', 'gpolanski', '', '15/Jun/2015',
   'IT_PROG',4200);
```

INSERTING SPECIAL VALUES

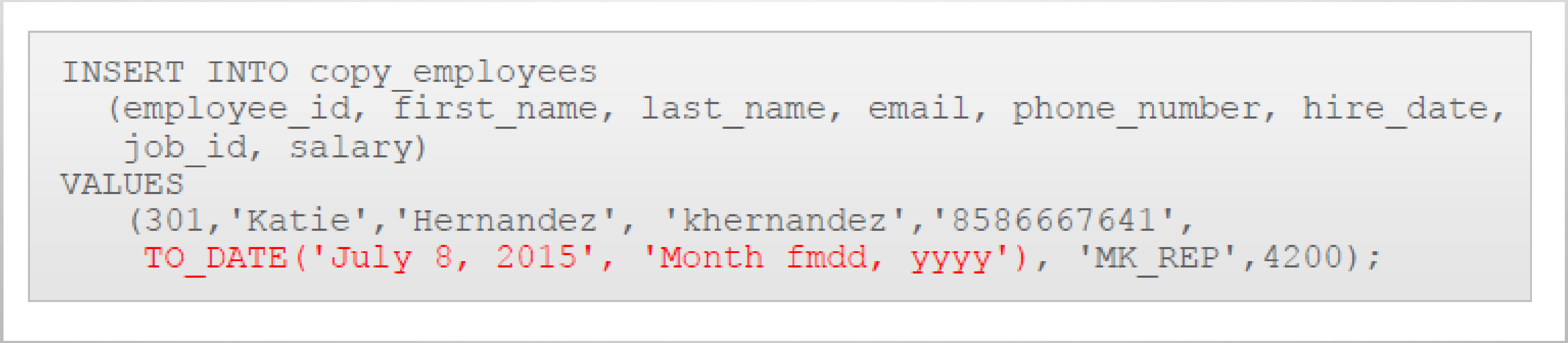
- SPECIAL VALUES SUCH AS SYSDATE AND USER CAN BE ENTERED IN THE VALUES LIST OF AN INSERT STATEMENT.
- SYSDATE WILL PUT THE CURRENT DATE AND TIME IN A COLUMN
- USER WILL INSERT THE CURRENT SESSION'S USERNAME, WHICH IN ORACLE APPLICATION EXPRESS IS OAE_PUBLIC_USER



```
INSERT INTO copy_employees
  (employee_id, first_name, last_name, email, phone_number, hire_date,
   job_id, salary)
VALUES
  (304,'Test',USER, 't_user', 4159982010, SYSDATE, 'ST_CLERK',2500);
```

INSERTING DATE VALUES

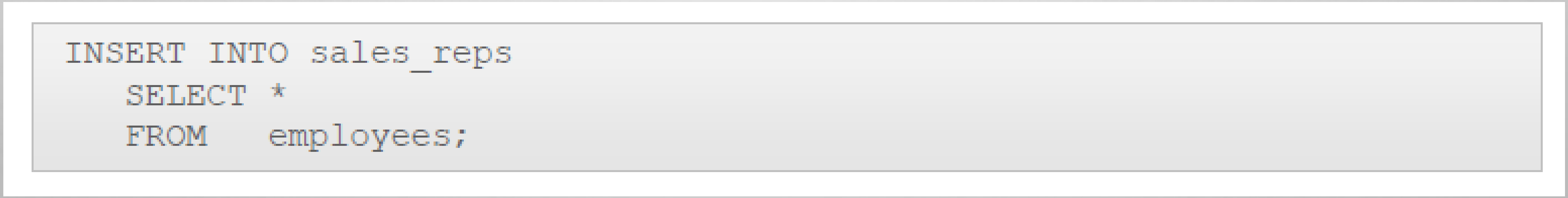
- THE DEFAULT FORMAT MODEL FOR DATE DATA TYPES IS DD-MON-YYYY.
- IF WE WANT TO INSERT A ROW WITH A NON-DEFAULT FORMAT FOR A DATE COLUMN WE MUST USE THE TO_DATE FUNCTION TO CONVERT THE DATE VALUE (A CHARACTER STRING) TO A DATE.



```
INSERT INTO copy_employees
   (employee_id, first_name, last_name, email, phone_number, hire_date,
    job_id, salary)
VALUES
    (301,'Katie','Hernandez', 'khernandez','8586667641',
     TO_DATE('July 8, 2015', 'Month fmdd, yyyy'), 'MK_REP',4200);
```

USING A SUBQUERY TO COPY ROWS

- EACH INSERT STATEMENT WE HAVE SEEN SO FAR ADDS ONLY ONE ROW TO THE TABLE.
- BUT SUPPOSE WE WANT TO COPY 100 ROWS FROM ONE TABLE TO ANOTHER.
- SQL ALLOWS A SUBQUERY WITHIN AN INSERT STATEMENT TO DO THIS.
- ALL THE ROWS ARE INSERTED IN THE TABLE.
- WE DON'T NEED A VALUES CLAUSE.



```
INSERT INTO sales_reps
   SELECT *
   FROM   employees;
```

USING A SUBQUERY TO COPY ROWS

- THE NUMBER OF COLUMNS AND THEIR DATA TYPES IN THE COLUMN LIST OF THE INSERT CLAUSE MUST MATCH THE NUMBER OF COLUMNS AND THEIR DATA TYPES IN THE SUBQUERY.
- NO PARENTHESIS AROUND THE SUBQUERY.