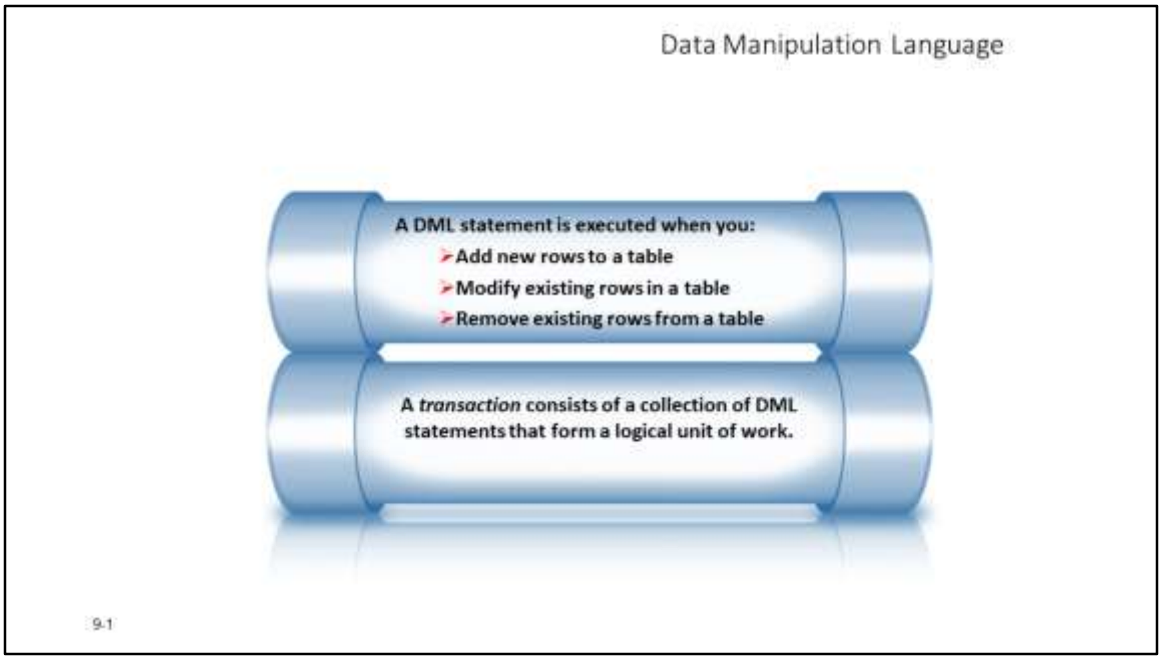


## Data Manipulation Language

A DML statement is executed when you:

- Add new rows to a table
- Modify existing rows in a table
- Remove existing rows from a table

A *transaction* consists of a collection of DML statements that form a logical unit of work.

9-1

Data Manipulation Language

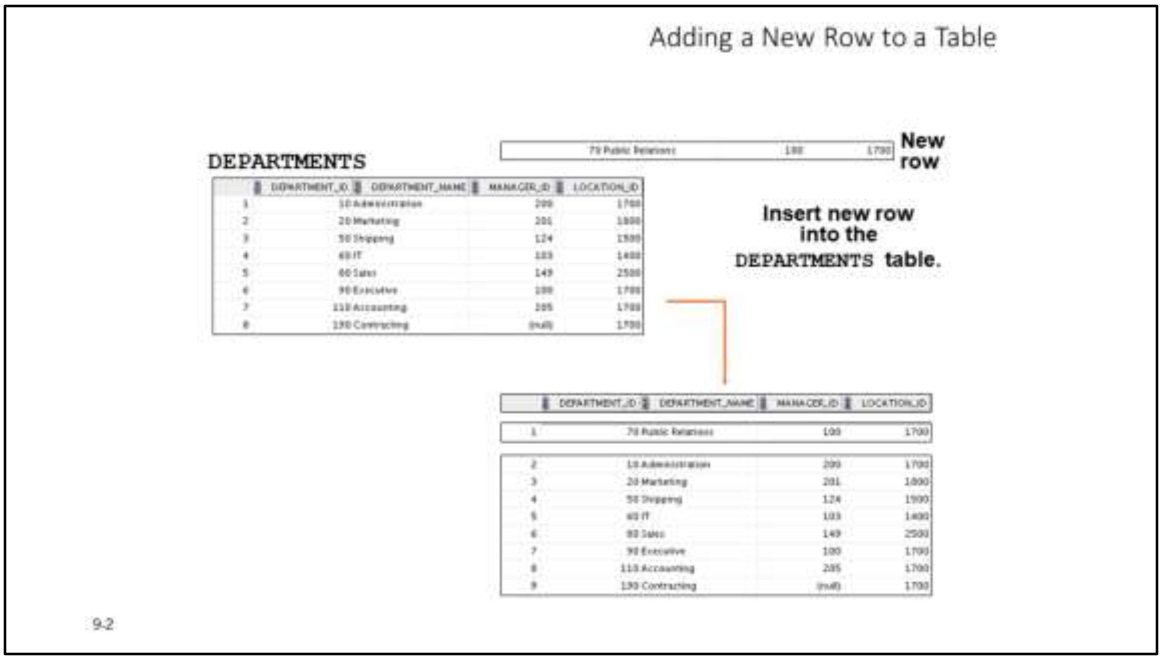
Data manipulation language (DML) is a core part of SQL. When you want to add, update, or delete data in the database, you execute a DML statement. A collection of DML statements that form a logical unit of work is called a *transaction*.

Consider a banking database. When a bank customer transfers money from a savings account to a checking account, the transaction might consist of three separate operations: decreasing the savings account, increasing the checking account, and recording the transaction in the transaction journal. The Oracle server must guarantee that all the three SQL statements are performed to maintain the accounts in proper balance. When something prevents one of the statements in the transaction from executing, the other statements of the transaction must be undone.

**Note**

Most of the DML statements in this lesson assume that no constraints on the table are violated. Constraints are discussed later in this course.

In SQL Developer, click the Run Script icon or press [F5] to run the DML statements. The feedback messages will be shown on the Script Output tabbed page.



## Adding a New Row to a Table

**New row**

|  | DEPARTMENT_ID | DEPARTMENT_NAME | MANAGER_ID | LOCATION_ID |
|---|---|---|---|---|
|  | 70 | Public Relations | 100 | 1700 |

**DEPARTMENTS**

|  | DEPARTMENT_ID | DEPARTMENT_NAME | MANAGER_ID | LOCATION_ID |
|---|---|---|---|---|
| 1 | 10 | Administration | 200 | 1700 |
| 2 | 20 | Marketing | 201 | 1800 |
| 3 | 50 | Shipping | 124 | 1500 |
| 4 | 60 | IT | 103 | 1400 |
| 5 | 80 | Sales | 149 | 2500 |
| 6 | 90 | Executive | 100 | 1700 |
| 7 | 110 | Accounting | 205 | 1700 |
| 8 | 190 | Contracting | (null) | 1700 |

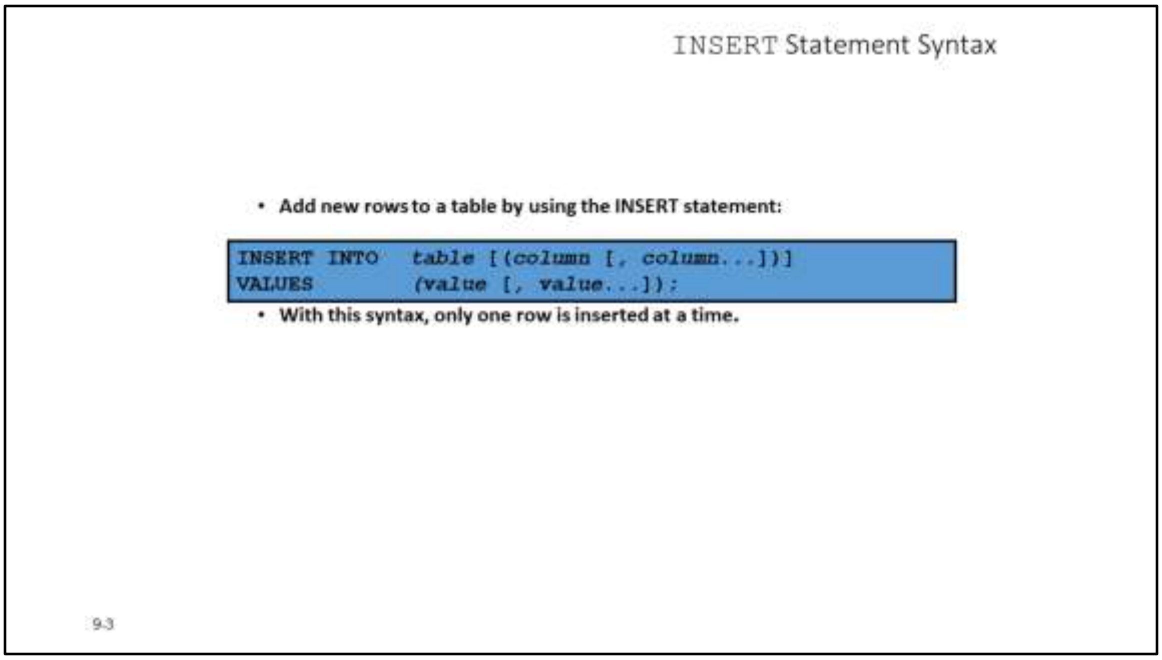
**Insert new row into the `DEPARTMENTS` table.**

|  | DEPARTMENT_ID | DEPARTMENT_NAME | MANAGER_ID | LOCATION_ID |
|---|---|---|---|---|
| 1 | 70 | Public Relations | 100 | 1700 |
| 2 | 10 | Administration | 200 | 1700 |
| 3 | 20 | Marketing | 201 | 1800 |
| 4 | 50 | Shipping | 124 | 1500 |
| 5 | 60 | IT | 103 | 1400 |
| 6 | 80 | Sales | 149 | 2500 |
| 7 | 90 | Executive | 100 | 1700 |
| 8 | 110 | Accounting | 205 | 1700 |
| 9 | 190 | Contracting | (null) | 1700 |

9-2

## Adding a New Row to a Table

The graphic in the slide illustrates the addition of a new department to the `DEPARTMENTS` table.



## INSERT Statement Syntax

- Add new rows to a table by using the INSERT statement:

```
INSERT INTO   table [(column [, column...])]
VALUES        (value [, value...]);
```

- With this syntax, only one row is inserted at a time.

9-3

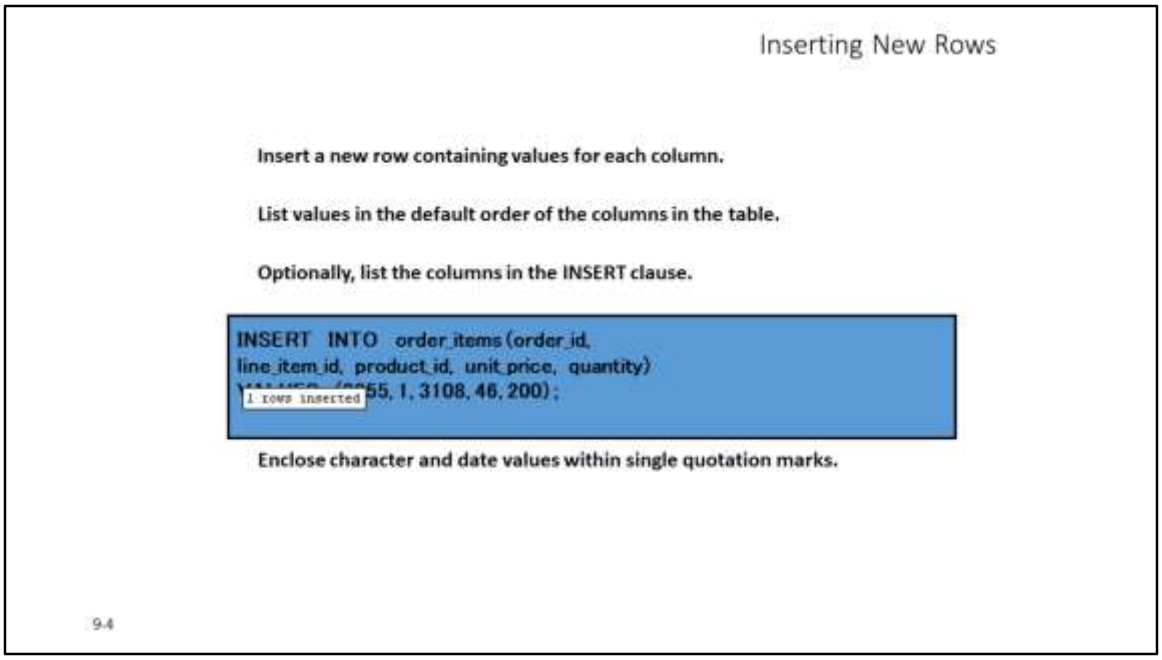
## INSERT Statement Syntax

You can add new rows to a table by issuing the `INSERT` statement.

In the syntax:

| | |
|---|---|
| *table* | Is the name of the table |
| *column* | Is the name of the column in the table to populate |
| *value* | Is the corresponding value for the column |

**Note:** This statement with the `VALUES` clause adds only one row at a time to a table.



## Inserting New Rows

- Insert a new row containing values for each column.
- List values in the default order of the columns in the table.
- Optionally, list the columns in the INSERT clause.

```
INSERT  INTO  order_items (order_id,
line_item_id,  product_id,  unit_price,  quantity)
VALUES  (2255, 1, 3108, 46, 200);
1 rows inserted
```

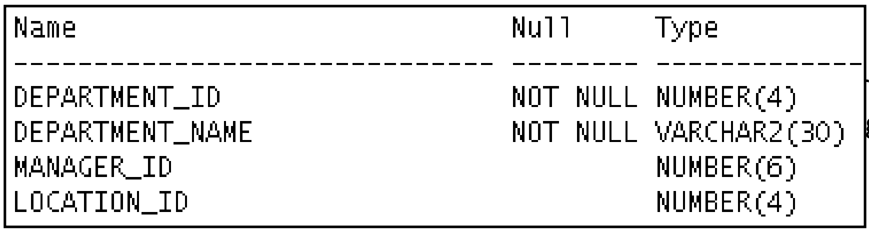
- Enclose character and date values within single quotation marks.

9-4

Inserting New Rows

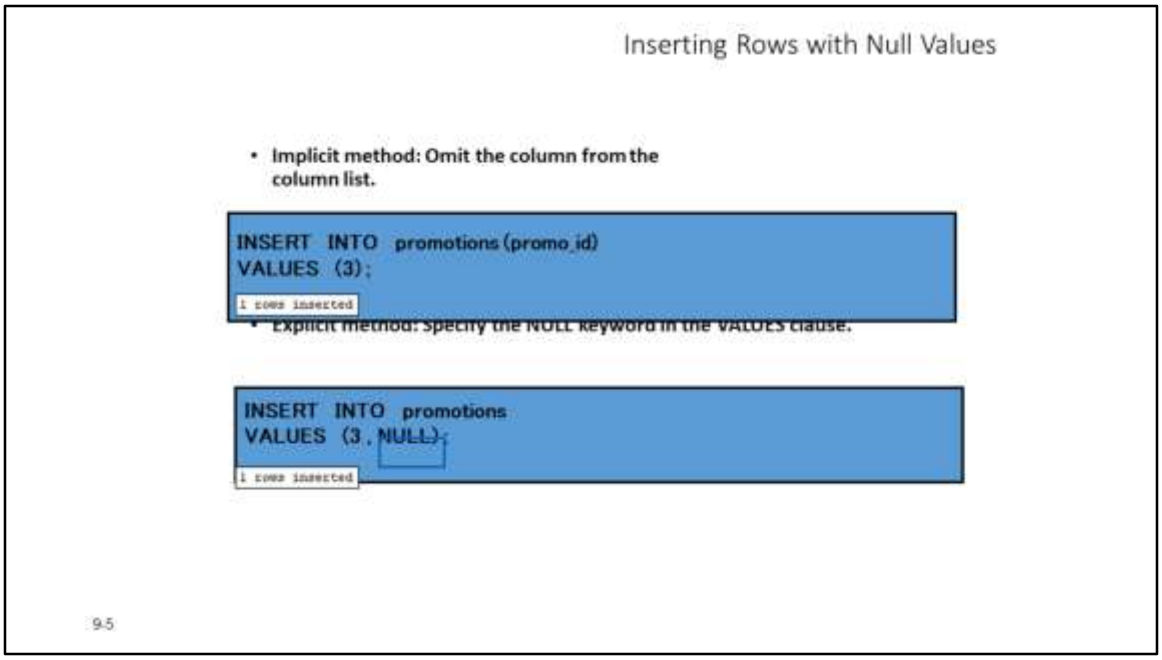
Because you can insert a new row that contains values for each column, the column list is not required in the `INSERT` clause. However, if you do not use the column list, the values must be listed according to the default order of the columns in the table, and a value must be provided for each column.

```
DESCRIBE order_items
```



```
Name                           Null     Type
------------------------------ -------- ------------
DEPARTMENT_ID                  NOT NULL NUMBER(4)
DEPARTMENT_NAME                NOT NULL VARCHAR2(30)
MANAGER_ID                              NUMBER(6)
LOCATION_ID                             NUMBER(4)
```

rks; however, it is
gle quotation



## Inserting Rows with Null Values

- Implicit method: Omit the column from the column list.

```
INSERT INTO promotions (promo_id)
VALUES (3);
1 rows inserted
```

- Explicit method: Specify the NULL keyword in the VALUES clause.

```
INSERT INTO promotions
VALUES (3, NULL);
1 rows inserted
```

9-5

Inserting Rows with Null Values

| Method | Description |
|---|---|
| Implicit | Omit the column from the column list. |
| Explicit | Specify the `NULL` keyword in the `VALUES` list; specify the empty string (`''`) in the `VALUES` list for character strings and dates. |

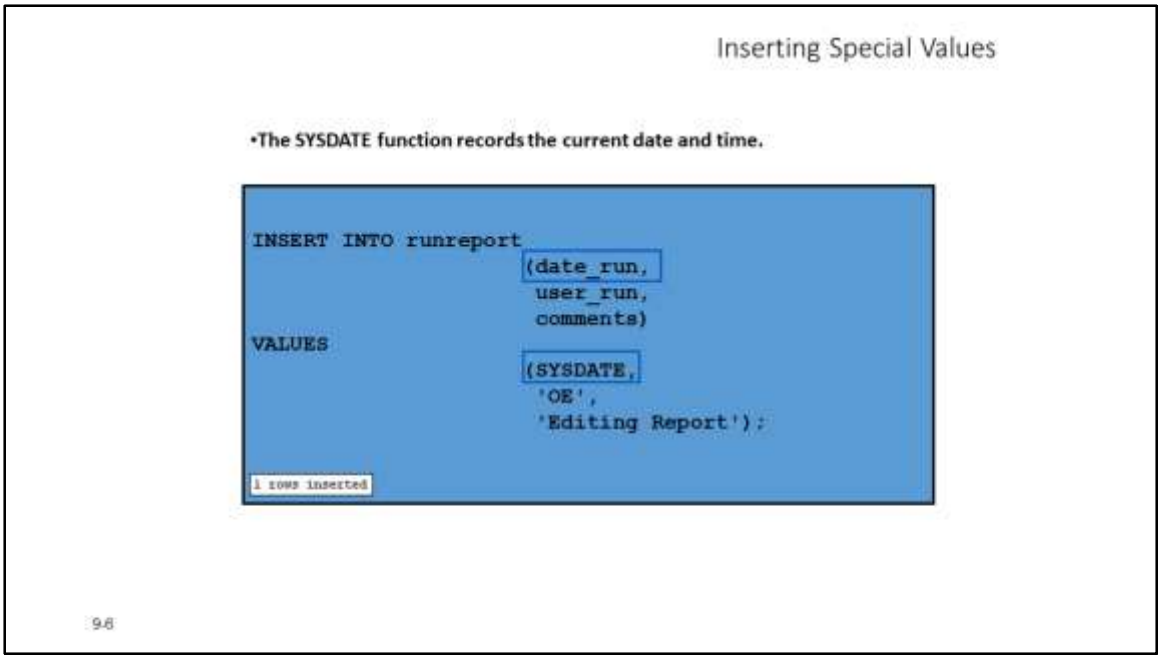
Be sure that you can use null values in the targeted column by verifying the `Null` status with the `DESCRIBE` command.

The Oracle server automatically enforces all data types, data ranges, and data integrity constraints. Any column that is not listed explicitly obtains a null value in the new row.

Common errors that can occur during user input are checked in the following order:

Mandatory value missing for a `NOT NULL` column
Duplicate value violating any unique or primary key constraint
`Any` value violating a `CHECK` constraint
Referential integrity maintained for foreign key constraint
Data type mismatches or values too wide to fit in column

**Note:** Use of the column list is recommended because it makes the `INSERT` statement more readable and reliable, or less prone to mistakes.



## Inserting Special Values

•The SYSDATE function records the current date and time.

```
INSERT INTO runreport
                    (date_run,
                     user_run,
                     comments)
VALUES
                    (SYSDATE,
                     'OE',
                     'Editing Report');
```

```
1 rows inserted
```

9-6

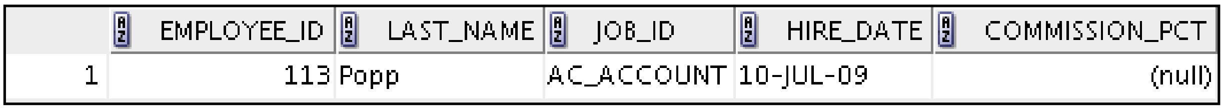
Inserting Special Values

You can use functions to enter special values in your table.

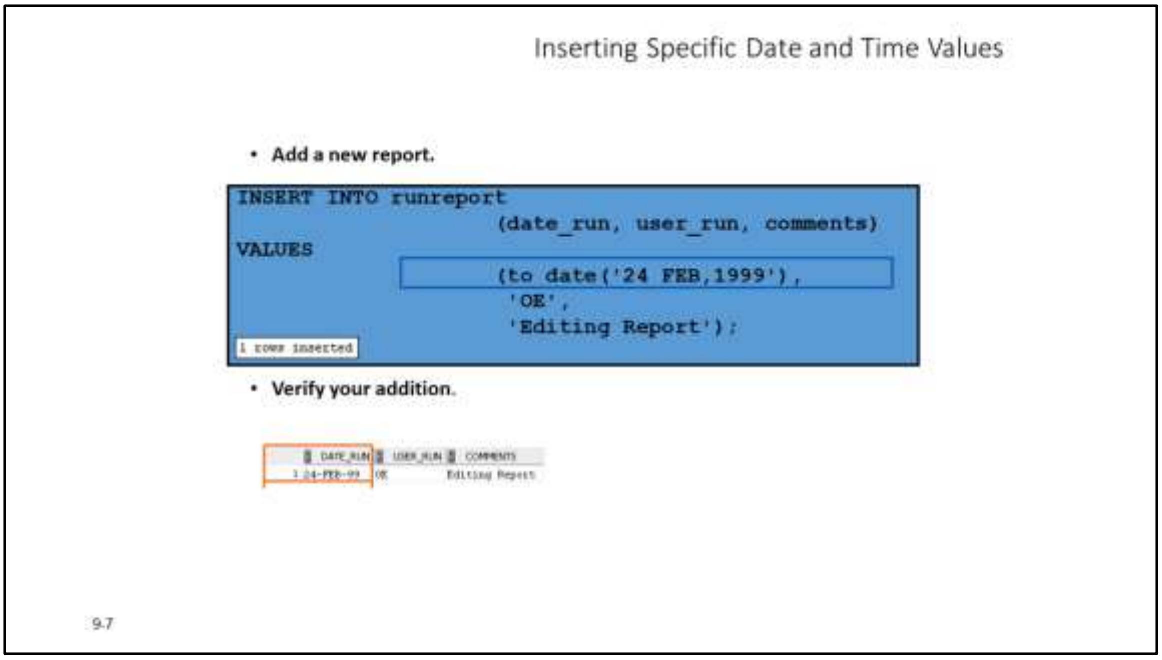
The slide example records information in the `RUNREPORT` table. It supplies the current date and time in the `DATE_RUN` column. It uses the `SYSDATE` function that returns the current date and time of the database server. You may also use the `CURRENT_DATE` function to get the current date in the session time zone. You can also use the `USER` function when inserting rows in a table. The `USER` function records the current username.

**Confirming Additions to the Table**

```
SELECT * FROM runreport;
```



| | EMPLOYEE_ID | LAST_NAME | JOB_ID | HIRE_DATE | COMMISSION_PCT |
|---|---|---|---|---|---|
| 1 | 113 | Popp | AC_ACCOUNT | 10-JUL-09 | (null) |



## Inserting Specific Date and Time Values

- Add a new report.

```
INSERT INTO runreport
                    (date_run, user_run, comments)
VALUES
                    (to_date('24 FEB,1999'),
                     'OE',
                     'Editing Report');
```

```
1 rows inserted
```

- Verify your addition.

|  | DATE_RUN | USER_RUN | COMMENTS |
|---|---|---|---|
| 1 | 24-FEB-99 | OE | Editing Report |

9-7

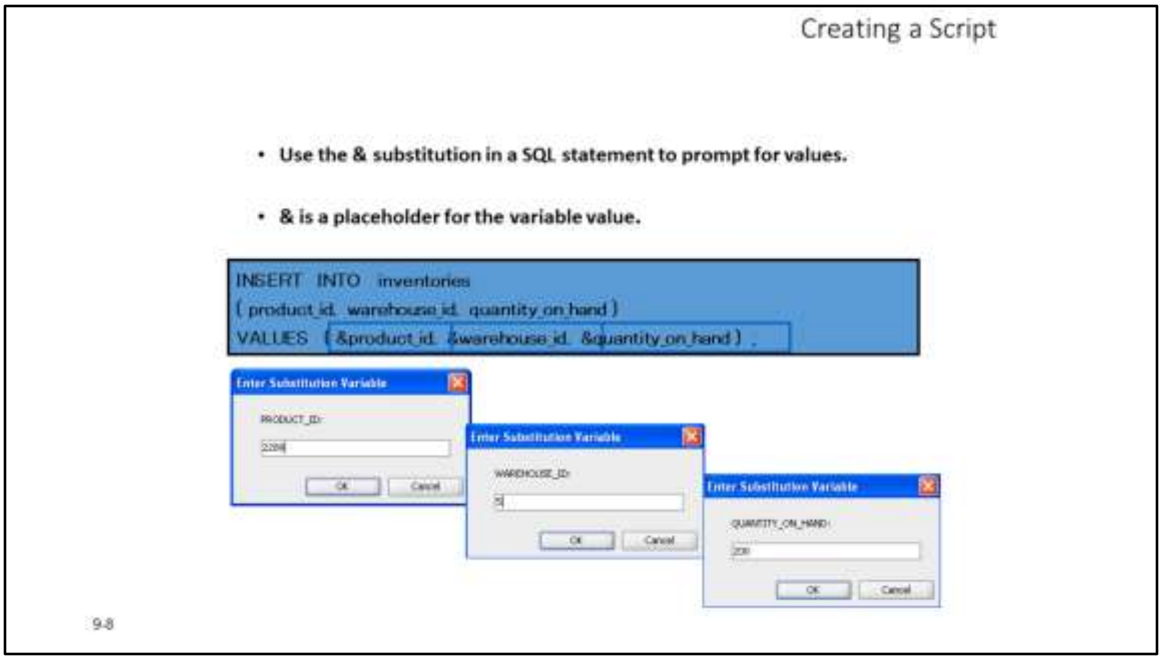
Inserting Specific Date and Time Values

The DD-MON-RR format is generally used to insert a date value. With the RR format, the system provides the correct century automatically.

You may also supply the date value in the DD-MON-YYYY format. This is recommended because it clearly specifies the century and does not depend on the internal RR format logic of specifying the correct century.

If a date must be entered in a format other than the default format (for example, with another century or a specific time), you must use the TO_DATE function.

The example in the slide records a report in the RUNREPORT table. It sets the DATE_RUN column to be 24 February, 1999.



## Creating a Script

- Use the & substitution in a SQL statement to prompt for values.
- & is a placeholder for the variable value.

```
INSERT  INTO  inventories
( product_id, warehouse_id, quantity_on_hand )
VALUES  ( &product_id, &warehouse_id, &quantity_on_hand ) ;
```

Enter Substitution Variable
PRODUCT_ID: 2288

Enter Substitution Variable
WAREHOUSE_ID: 5

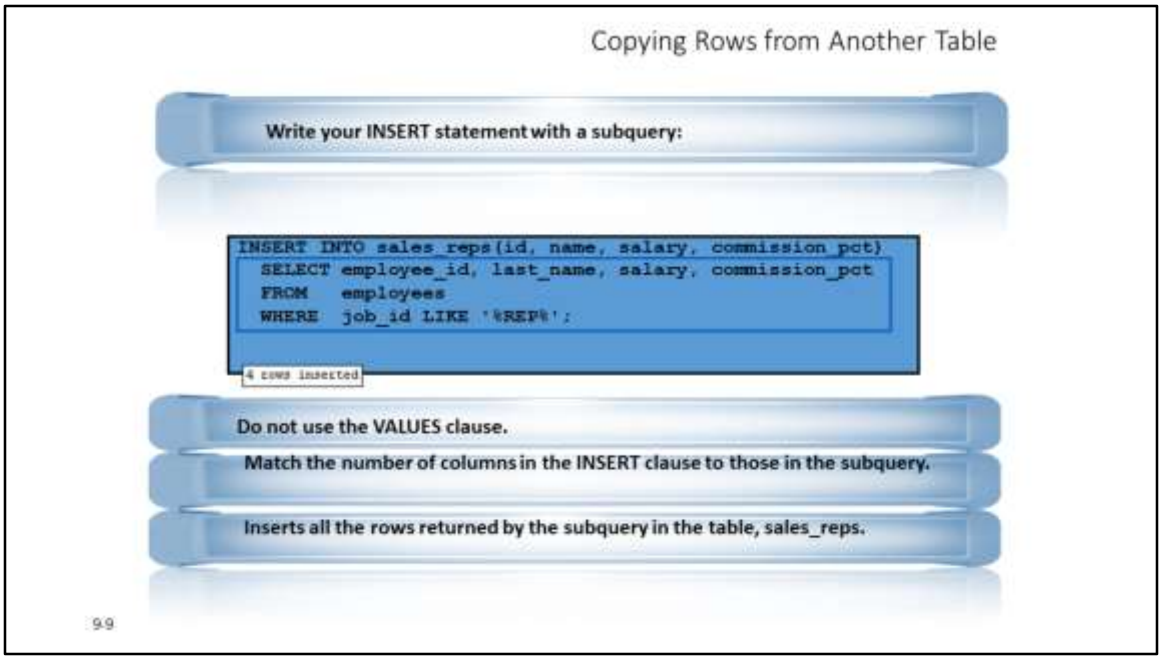
Enter Substitution Variable
QUANTITY_ON_HAND: 200

9-8

Creating a Script

You can save commands with substitution variables to a file and execute the commands in the file. The example in the slide records information for a product in the PRODUCTS table.

Run the script file and you are prompted for input for each of the ampersand (`&`) substitution variables. After entering a value for the substitution variable, click the OK button. The values that you input are then substituted into the statement. This enables you to run the same script file over and over, but supply a different set of values each time you run it.



## Copying Rows from Another Table

Write your INSERT statement with a subquery:

```
INSERT INTO sales_reps(id, name, salary, commission_pct)
  SELECT employee_id, last_name, salary, commission_pct
  FROM   employees
  WHERE  job_id LIKE '%REP%';
```

```
4 rows inserted
```

Do not use the VALUES clause.

Match the number of columns in the INSERT clause to those in the subquery.

Inserts all the rows returned by the subquery in the table, sales_reps.

9-9

## Copying Rows from Another Table

You can use the `INSERT` statement to add rows to a table where the values are derived from existing tables. In the example in the slide, for the `INSERT INTO` statement to work, you must have already created the `sales_reps` table using the `CREATE TABLE` statement. `CREATE TABLE` is discussed in the lesson titled "Using DDL Statements to Create and Manage Tables."

In place of the `VALUES` clause, you use a subquery.

**Syntax**

```
INSERT INTO table [ column (, column) ] subquery;
```

In the syntax:

| | |
|---|---|
| *table* | Is the name of the table |
| *column* | Is the name of the column in the table to populate |
| *subquery* | Is the subquery that returns rows to the table |

The number of columns and their data types in the column list of the `INSERT` clause must match the number of values and their data types in the subquery. Zero or more rows are added depending on the number of rows returned by the subquery. To create a copy of the rows of a table, use `SELECT *` in the subquery:

```
INSERT INTO copy_emp
```

```
SELECT *
FROM   employees;
```