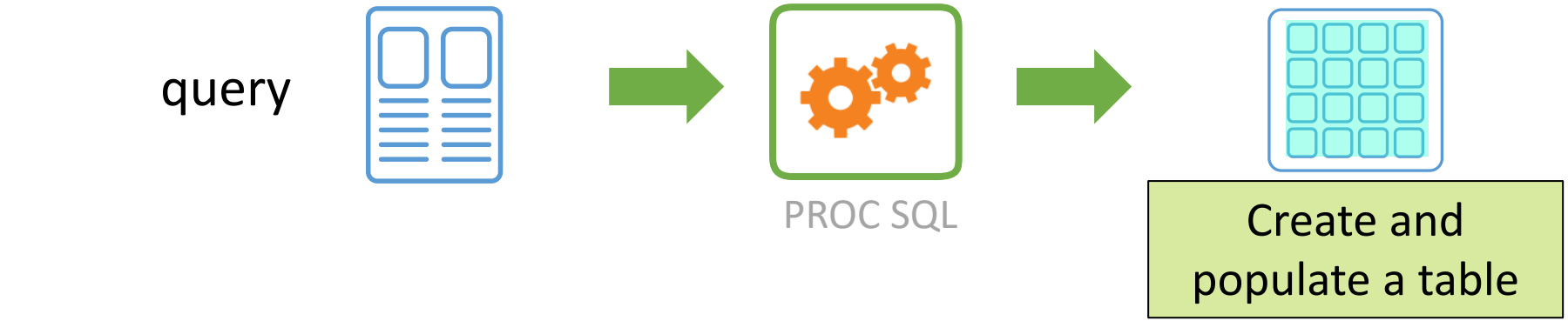
PROC SQL Fundamentals

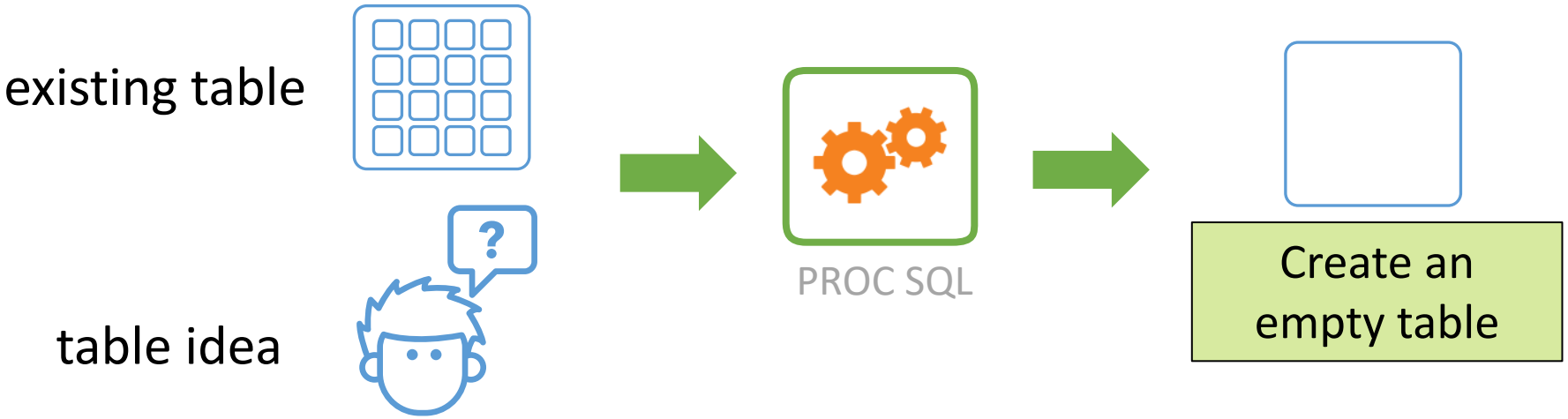
1. Generating Simple Reports

2. Summarizing and Grouping Data

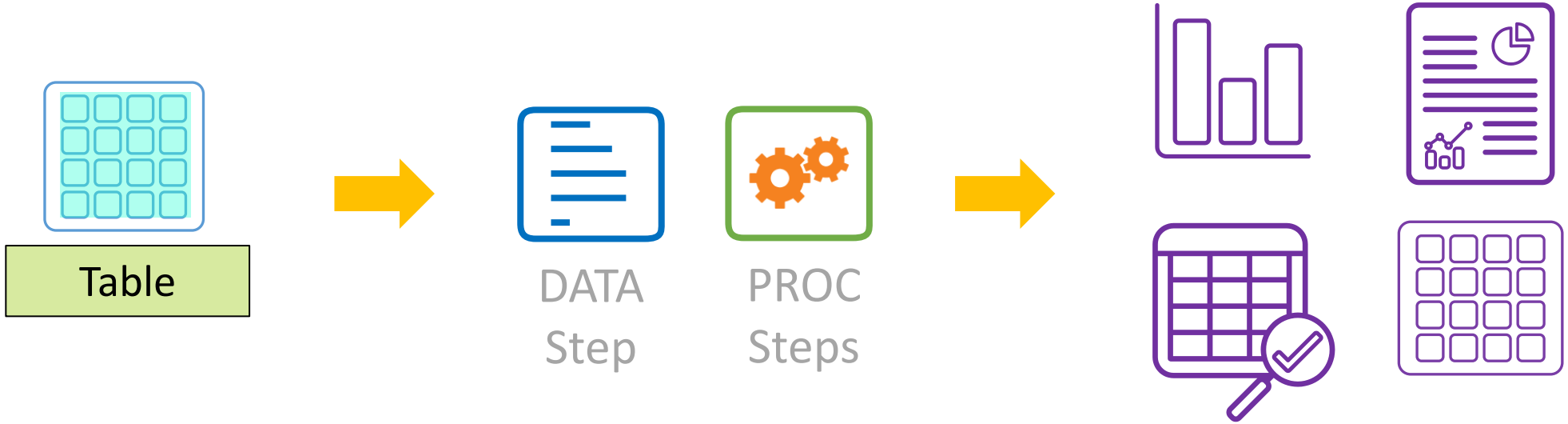
3. Creating and Managing Tables

Creating Tables





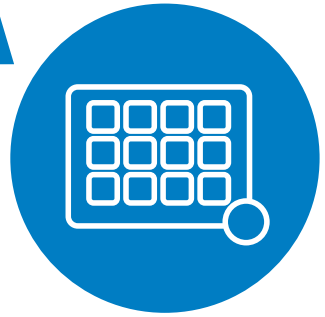
Using Tables Created from a Query



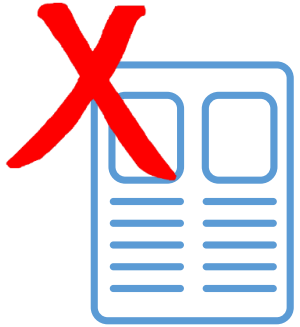
Creating a Table from a Query Result

CREATE TABLE *table-name* **AS** *query*

```
proc sql;
create table work.highcredit as
select FirstName, LastName,
       UserID, CreditScore
    from sq.customer
    where CreditScore > 700;
quit;
```



work.highcredit



NOTE: Table WORK.HIGHCREDIT created, with 26006 rows and 4 columns.

2.10 Activity

- Create top products table, i.e., select products generating the highest profit from the orders table.

1. Examine and run the following query that creates a report (NOT a table). View the results.

```
PROC SQL NUMBER;
TITLE "Top Products";
SELECT Product_ID, SUM(Profit) FORMAT=DOLLAR10.2 AS Total_Profit
       FROM ORION.ORDERS
       GROUP BY Product_ID
       HAVING CALCULATED Total_Profit >= 500;
TITLE;
QUIT;
```

2. Remove the TITLE statements and add the CREATE TABLE statement and create a table named **Top_Products**. Run the query and confirm that the table was created successfully.

2.10 Activity – Correct Answer

- Create a table of top products, i.e., select products generating the highest profit from the orders table.

1. Examine and run the given query (see the previous slide). View the results.

2. Remove the TITLE statements and add the CREATE TABLE statement and create a table named **Top_Products**. Run the query and confirm that the table was created successfully.

1st query: generates a report

Top Products

Row	Product ID	Total_Profit
1	230100200025	$602.60
2	230100500026	$689.75
3	230100700008	$1,578.50
4	230100700009	$1,660.65
5	230100700011	$1,322.00
6	240100400043	$679.20

2nd query generates a table

TOP_PRODUCTS

	Product_ID	Total_Profit
1	230100200025	$602.60
2	230100500026	$689.75
3	230100700008	$1,578.50
4	230100700009	$1,660.65
5	230100700011	$1,322.00
6	240100400043	$679.20

NOTE: Table WORK.TOP_PRODUCTS created, with 19 rows and 2 columns.

Copying the Structure of an Existing Table

CREATE TABLE *table-name*
LIKE *existing-table*;

Creates a copy of the table ***structure***

```
proc sql;
create table work.highcredit
     like sq.customer(keep=FirstName LastName
                           UserID CreditScore);
quit;
```

NOTE: Table WORK.HIGHCREDIT created, with 0 rows and 4 columns.

Creating a Table by Defining Columns

CREATE TABLE *table-name*
(column-name type(length)
<, ...column-name type(length)>);

```
proc sql;
create table work.employee
    (FirstName char(20),
     LastName char(20),
     DOB date format=mmddyy10.,
     EmpID num format=z6.);
quit;
```

Specify column ***names*** and ***attributes***.

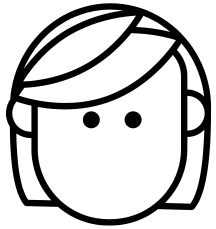
NOTE: Table WORK.EMPLOYEE created, with 0 rows and 4 columns.

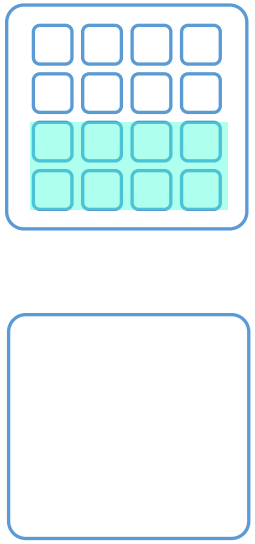
Creating Tables with PROC SQL: Summary

Description	Syntax
Creating a Table from a Query Result	**CREATE TABLE** *table-name* **AS SELECT…;**
Copying the Structure from an Existing Table	**CREATE TABLE** *table-name* **LIKE** *old-table-name***;**
Creating a Table by Defining Columns	**CREATE TABLE** *table-name* (*column-name type(length*) <, ...*column-name type(length*)>)**;**

Inserting Rows into Tables

Use the **INSERT** statement to insert data values into tables.





Inserting Rows with a Query

INSERT INTO *table-name <(column list)>*
SELECT *columns*
FROM *table-name;*

```
proc sql;
insert into work.highcredit
    (FirstName, LastName, UserID, CreditScore)
    select FirstName, LastName,
           UserID, CreditScore
        from sq.customer
        where CreditScore > 700;
quit;
```

Columns from the query ***must*** be in the same position as in the INSERT column list.

Inserting Rows with the VALUES Clause

INSERT INTO *table-name <(column list)>*
VALUES *(value,value,...)*;

```
proc sql;
insert into employee
     (FirstName, LastName, DOB, EmpID)
    values("Diego", "Lopez", "01SEP1980"d, 1280)
    values("Omar", "Fayed", "21MAR1989"d, 1310);
quit;
```

Data values align with column names in the INSERT column list.

Inserting Rows with the SET Clause

INSERT INTO *table-name*
SET *column-name=value,*
column-name=value,...;

```
proc sql;
insert into employee
   set FirstName= "Diego",
       LastName= "Lopez",
       DOB = "01SEP1980"d,
       EmpID = 1280;
quit;
```

Columns within the SET clause ***must*** exist in the table.

Inserting Rows with PROC SQL: Summary

Description	Syntax
A query returning multiple rows based on positional values	**INSERT INTO** *table-name* <(*column list*)> **SELECT** *columns* **FROM** *table-name*;
One clause per row using positional values	**INSERT INTO** *table-name* <(*column list*)> **VALUES** (*value1,value2,...*);
One clause per row using column-value pairs	**INSERT INTO** *table-name* **SET** *column-name=value*, *column-name=value*,...;

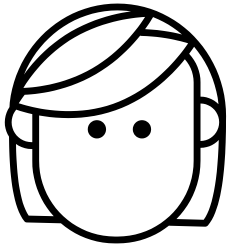
Dropping Tables in SQL

DROP TABLE *table-name*;

```
proc sql;
drop table work.employee;
quit;
```

NOTE: Table WORK.EMPLOYEE has been dropped.

This is useful if you are working with DBMSs that do not allow you to overwrite existing tables.



2.11 Activity

- Create a new table and insert rows into it:

1. Complete the following CREATE TABLE statement using the structure of **EMPLOYEE_MASTER** table and keep only the following columns: **Employee_ID Employee_Name Employee_Hire_Date Department Job_Title**. Run the query and confirm that an empty table was created.

```
proc sql;
create table work.managers
       /*Complete the query*/;
quit;
```

2.11 Activity

- Create a new table and insert rows into it:

2. **Inserting Rows with a Query:** Enter the correct column names and a where clause (Job_Title should contain the word ‘Manager’) to complete the INSERT INTO statement. Run the query. How many rows were inserted into the table **Managers**?

```
proc sql;
insert into work.managers(/*Add columns*/)
select Employee_ID, Employee_Name, Employee_Hire_Date, Department, Job_Title
    from orion.employee_master
    /*Write the where clause*/;
quit;
```

2.11 Activity

- Create a new table and insert rows into it:

3. **Inserting Rows with the SET Clause:** Complete the INSERT INTO statement with the SET clause and insert yourself as a manager into the **Managers** table. Run the query. What does the note in the log say?

```
proc sql;
insert into managers
    set Employee_ID = ,     /*<-----Add a random six-digit ID number starting with 130 */
        Employee_Name = "",        /*<-----Add your full name as "LastName, FirstName" */
        Employee_Hire_Date = ",          /*<-----Add your start date as '01-JAN-1960'd */
        Department = "",                                  /*<-----Add your department */
        Job_Title = "";          /*<-----Add a job title but it must contain 'Manager' */
quit;
```

2.11 Activity

- Create a new table and insert rows into it:

4. Complete the code to drop the **Managers** table.

```
proc sql;
/*Complete the code to drop the managers table*/
quit;
```

continued...

2.11 Activity – Correct Answer

1. **There are four columns and zero rows in the new managers table.**

```
proc sql;
create table work.managers
       like orion.employee_master(keep= Employee_ID Employee_Name
                                        Employee_Hire_Date Department Job_Title);
quit;
```

NOTE: Table WORK.MANAGERS created, with 0 rows and 5 columns.

2. How many rows were inserted into the table **managers**? **38 rows were inserted into work.managers.**

```
insert into work.managers(Employee_ID, Employee_Name,
                          Employee_Hire_Date,
                          Department, Job_Title)
    select Employee_ID, Employee_Name, Employee_Hire_Date,
             Department, Job_Title
    from orion.employee_master
        where UPCASE(Job_Title) contains "MANAGER";
```

2.11 Activity – Correct Answer

3. What does the note in the log say?
NOTE: 1 row was inserted into WORK.MANAGERS.

```
proc sql;
insert into managers
    set Employee_ID= 130549,
        Employee_Name="Koyuncu, Isil",
        Employee_Hire_Date='01-AUG-2020'd,
        Department=    "Operations",
        Job_Title="Supply Chain Manager";
quit;
```

4.

```
proc sql;
drop table managers;
quit;
```

2.11 Activity – Correct Answer

Alternatively, you could write:

```
proc sql;
insert into managers (Employee_ID, Employee_Name,
Employee_Hire_Date, Department, Job_Title)
    values(130549, "Koyuncu, Isil", '01-AUG-2020'd,
"Operations", "Supply Chain Manager")
    values(130550, "Kilci, Firat", '03-OCT-2021'd,
"Analytics", "Data Science Manager");
quit;
```

Additional Statements

Statement	Description
ALTER TABLE	Adds columns to, drops columns from, and changes column attributes in an existing table
UPDATE	Modifies a column's values in existing rows of a table or view
DELETE	Removes one or more rows from a table or view that is specified in the FROM clause

ALTER TABLE

- Rerun steps 1, 2, and 3 of the previous activity, i.e., do not drop the table.

```
PROC SQL;
ALTER TABLE work.managers
ADD Twentieth_Anniversary NUM INFORMAT=date9. FORMAT=date9.,
Upcoming_20th_Anniversary CHAR (3);
QUIT;
```

- Run select * from work.managers to see the altered table.

MODIFY CLAUSE

- If a column is already in the table, then you can change the following column attributes by using the MODIFY clause: length, informat, format, and label.

```
PROC SQL;
ALTER TABLE work.managers
MODIFY Employee_Hire_Date FORMAT=mmddyy10.,
              Twentieth_Anniversary FORMAT=mmddyy10.;
quit;
```

- The values in a table are either truncated or padded with blanks (if character data) as necessary to meet the specified length attribute.
- You cannot change a character column to numeric and vice versa. To change a column's data type, use PUT() or INPUT() functions to recode the column and then DROP the unwanted column.

UPDATE STATEMENT

- Let’s update the values of the newly created columns in the altered managers table.
- If it hasn’t been 20 years since an employee is hired, then we will update the values of Upcoming_20th_Anniversary column to ‘Yes’ and compute their Twentieth_Anniversary.

```
PROC SQL;
UPDATE work.managers
SET Upcoming_20th_Anniversary='Yes’,
       Twentieth_Anniversary=INTNX('YEAR',Employee_Hire_Date,20, "sameday")
WHERE YRDIF(Employee_Hire_Date,TODAY())<20;
QUIT;
```

- Run select * from work.managers to see the updated table.

DELETE STATEMENT

- Removes one or more rows from a table or view that is specified in the FROM clause.

```
PROC SQL;
DELETE FROM work.managers
WHERE UPCASE(Job_Title) CONTAINS "SENIOR";
QUIT;
```

- You cannot use DELETE FROM on a table that is accessed by an engine that does not support UPDATE processing, i.e., if access=readonly.
- CAUTION: If you omit a WHERE clause, then the DELETE statement deletes all the rows.

Syntax Summary

DROP TABLE *table-name;*

Drop Tables

CREATE TABLE *table-name* **AS** *query*
CREATE TABLE *table-name* **LIKE …**
CREATE TABLE *table-name* **(…)**

Create Tables

INSERT INTO *table-name <(column list)> query*
INSERT INTO *table-name <(column list)>* **VALUES …**
INSERT INTO *table-name* **SET …**

Insert Rows

Practice

This exercise reinforces the concepts discussed previously.