Database Concepts

Ninth Edition

Chapter 3

Structured Query Language

Learning Objectives

- Learn basic SQL statements for creating database structures
- Learn basic SQL statements for adding data to a database
- Learn basic SQL SELECT statements and options for processing a single table
- Learn basic SQL SELECT statements for processing multiple tables with subqueries
- Learn basic SQL SELECT statements for processing multiple tables with joins
- Learn basic SQL statements for modifying and deleting data from a database
- Learn basic SQL statements for modifying and deleting database tables and constraints

Structured Query Language

Learn basic SQL statements for creating database structures

- **Structured Query Language (SQL)** is not a complete programming language; rather, it is a data sublanguage
  - Developed by IBM in the late 1970's
  - Endorsed and adopted by ANSI in 1992
  - Endorsed as a standard by the International Organization for Standardization (ISO)
  - It is text oriented
  - Can be used by MS Access using Query by Example (QBE)

SQL Statement Categories

Learn basic SQL statements for creating database structures

- SQL statement categories are as follows:
  - Data definition language (DDL)
    - Statements used to create database structures
  - Data manipulation language (DML)
    - Statements used to query, insert, modify and delete data
  - SQL/Persistent stored modules (SQL/PSM)
    - Statements to extend SQL by adding procedural programming capabilities
  - Transaction control language (TCL)
    - Statements used to mark transaction boundaries
  - Data control language (DCL)
    - Statements used to grant and revoke database permissions

Wedgewood Pacific Example

Learn basic SQL statements for creating database structures

- This chapter uses the Wedgewood Pacific (WP) company as an example.
  - Founded in 1957 in Seattle, WA
  - Manufacture and sell consumer drone aircraft
  - In January, 2018, the FAA said that over one million drones had been registered
  - WP's database has four tables (Department, Employee, Project, and Assignment)

Wedgewood Pacific Relations

Learn basic SQL statements for creating database structures

- The following relations are used for the Wedgewood Pacific (WP) example used in this chapter:

DEPARTMENT (DepartmentName, BudgetCode, OfficeNumber, DepartmentPhone)

EMPLOYEE (EmployeeNumber, FirstName, LastName, *Department*, Position, *Supervisor*, OfficePhone, EmailAddress)

PROJECT (ProjectID, ProjectName, *Department*, MaxHours, StartDate, EndDate)

ASSIGNMENT (*ProjectID*, *EmployeeNumber*, HoursWorked)

WP's Referential Integrity Constraints

Learn basic SQL statements for creating database structures

- A referential integrity constraint is used to link (or reference) relations. This means that a foreign key in a relation must also exist in the relation in which it serves as the primary key. WP's referential integrity constraints:

Department in EMPLOYEE must exist in DepartmentName in DEPARTMENT

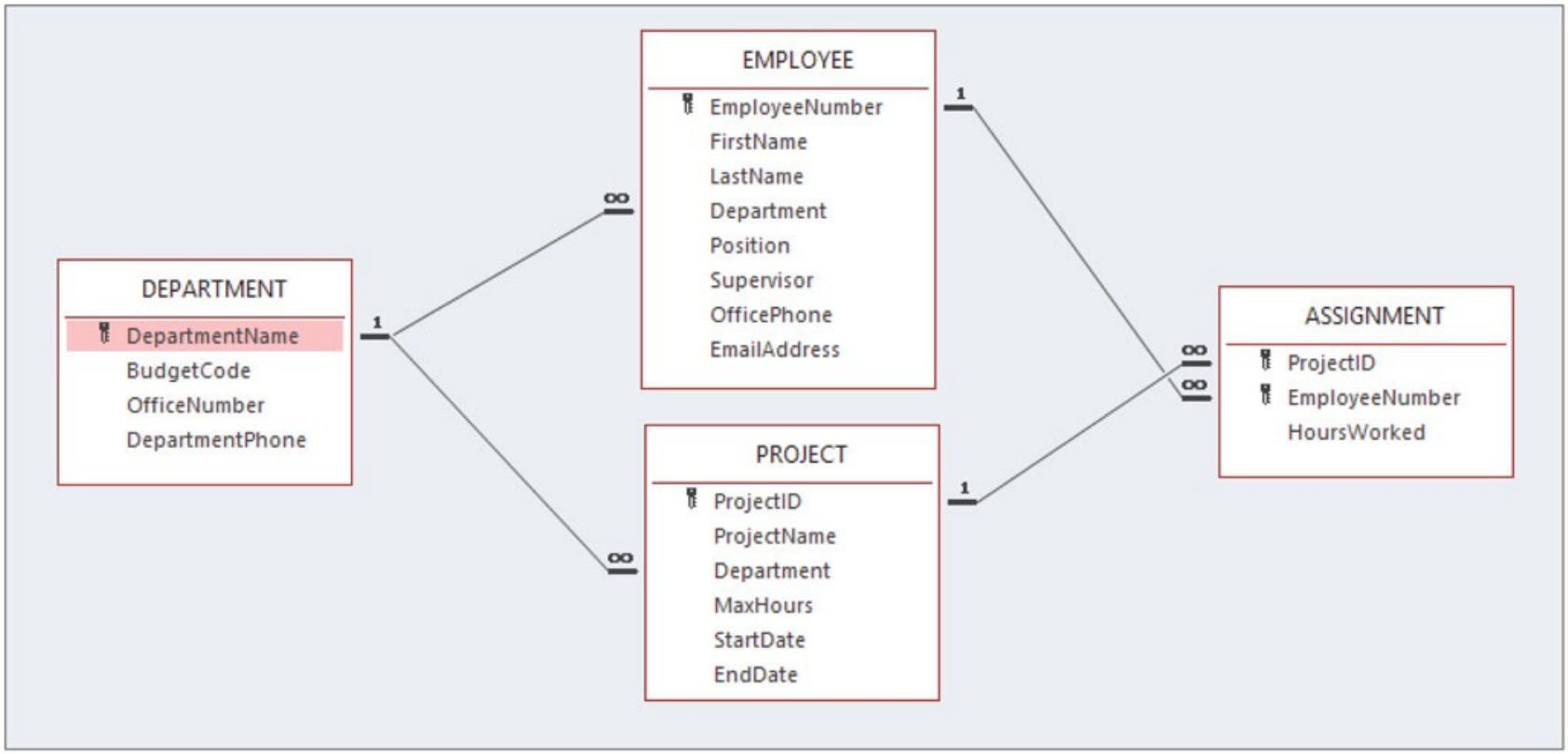
Supervisor in EMPLOYEE must exist in EmployeeNumber in EMPLOYEE

Department in PROJECT must exist in DepartmentName in DEPARTMENT

ProjectID in ASSIGNMENT must exist in ProjectID in PROJECT

EmployeeNumber in ASSIGNMENT must exist in EmployeeNumber in EMPLOYEE

Figure 3.1(a) Database Column Characteristics for the WP Database



Access 2019, Windows 10, Microsoft Corporation

Figure 3.1(b) DEPARTMENT Table

Column Name	Type	Key	Required	Remarks
DepartmentName	Short Text (35)	Primary Key	Yes	
BudgetCode	Short Text (30)	No	Yes	
OfficeNumber	Short Text (15)	No	Yes	
DepartmentPhone	Short Text (12)	No	Yes	

Figure 3.1(c) EMPLOYEE Table

Column Name	Type	Key	Required	Remarks
EmployeeNumber	AutoNumber	Primary Key	Yes	Surrogate Key
FirstName	Short Text (25)	No	Yes	
LastName	Short Text (25)	No	Yes	
Department	Short Text (35)	Foreign Key	Yes	Links to DepartmentName in DEPARTMENT
Position	Short Text (35)	No	No	
Supervisor	Number	Foreign Key	No	Long Integer. Links to EmployeeNumber in EMPLOYEE
OfficePhone	Short Text (12)	No	No	
EmailAddress	Short Text (100)	No	Yes	

Figure 3.1(d) PROJECT Table

ColumnName	Type	Key	Required	Remarks
ProjectID	Number	Primary Key	Yes	Long Integer
ProjectName	Short Text (50)	No	Yes	
Department	Short Text (35)	Foreign Key	Yes	Links to DepartmentName in DEPARTMENT
MaxHours	Number	No	Yes	Double
StartDate	Date	No	No	
EndDate	Date	No	No	

Figure 3.1(e) ASSIGNMENT Table

Column Name	Type	Key	Required	Remarks
ProjectID	Number	Primary Key, Foreign Key	Yes	Long Integer Links to ProjectID in PROJECT
EmployeeNumber	Number	Primary Key, Foreign Key	Yes	Long Integer Links to EmployeeNumber in EMPLOYEE
HoursWorked	Number	No	No	Double

Figure 3.2(a) DEPARTMENT Table

Sample Data for the WP Database

DepartmentName	BudgetCode	OfficeNumber	DepartmentPhone
Administration	BC-100-10	BLDG01-210	360-285-8100
Legal	BC-200-10	BLDG01-220	360-285-8200
Human Resources	BC-300-10	BLDG01-230	360-285-8300
Finance	BC-400-10	BLDG01-110	360-285-8400
Accounting	BC-500-10	BLDG01-120	360-285-8405
Sales and Marketing	BC-600-10	BLDG01-250	360-285-8500
InfoSystems	BC-700-10	BLDG02-210	360-285-8600
Research and Development	BC-800-10	BLDG02-250	360-285-8700
Production	BC-900-10	BLDG02-110	360-285-8800

Figure 3.2(b) Sample Data for the WP Database: EMPLOYEE Table (1 of 2)

Employee Number	FirstName	LastName	Department	Position	Supervisor	OfficePhone	EmailAddress
1	Mary	Jacobs	Administration	CEO		360-285-8110	Mary.Jacobs@WP.com
2	Rosalie	Jackson	Administration	Admin Assistant	1	360-285-8120	Rosalie.Jackson@WP.com
3	Richard	Bandalone	Legal	Attorney	1	360-285-8210	Richard.Bandalone@WP.com
4	George	Smith	Human Resources	HR3	1	360-285-8310	George.Smith@WP.com
5	Alan	Adams	Human Resources	HR1	4	360-285-8320	Alan.Adams@WP.com
6	Ken	Evans	Finance	CFO	1	360-285-8410	Ken.Evans@WP.com
7	Mary	Abernathy	Finance	FA3	6	360-285-8420	Mary.Abernathy@WP.com
8	Tom	Caruthers	Accounting	FA2	6	360-285-8430	Tom.Caruthers@WP.com
9	Heather	Jones	Accounting	FA2	6	360-285-8440	Heather.Jones@WP.com
10	Ken	Numoto	Sales and Marketing	SM3	1	360-285-8510	Ken.Numoto@WP.com

Figure 3.2(b) Sample Data for the WP Database: EMPLOYEE Table (2 of 2)

Employee Number	FirstName	LastName	Department	Position	Supervisor	OfficePhone	EmailAddress
11	Linda	Granger	Sales and Marketing	SM2	10	360-285-8520	Linda.Granger@WP.com
12	James	Nestor	InfoSystems	CIO	1	360-285-8610	James.Nestor@WP.com
13	Rick	Brown	InfoSystems	IS2	12		Rick.Brown@WP.com
14	Mike	Nguyen	Research and Development	CTO	1	360-285-8710	Mike.Nguyen@WP.com
15	Jason	Sleeman	Research and Development	RD3	14	360-285-8720	Jason.Sleeman@WP.com
16	Mary	Smith	Production	OPS3	1	360-285-8810	Mary.Smith@WP.com
17	Tom	Jackson	Production	OPS2	16	360-285-8820	Tom.Jackson@WP.com
18	George	Jones	Production	OPS2	17	360-285-8830	George.Jones@WP.com
19	Julia	Hayakawa	Production	OPS1	17		Julia.Hayakawa@WP.com
20	Sam	Stewart	Production	OPS1	17		Sam.Stewart@WP.com

Figure 3.2(c) Sample Data for the WP Database: PROJECT Table

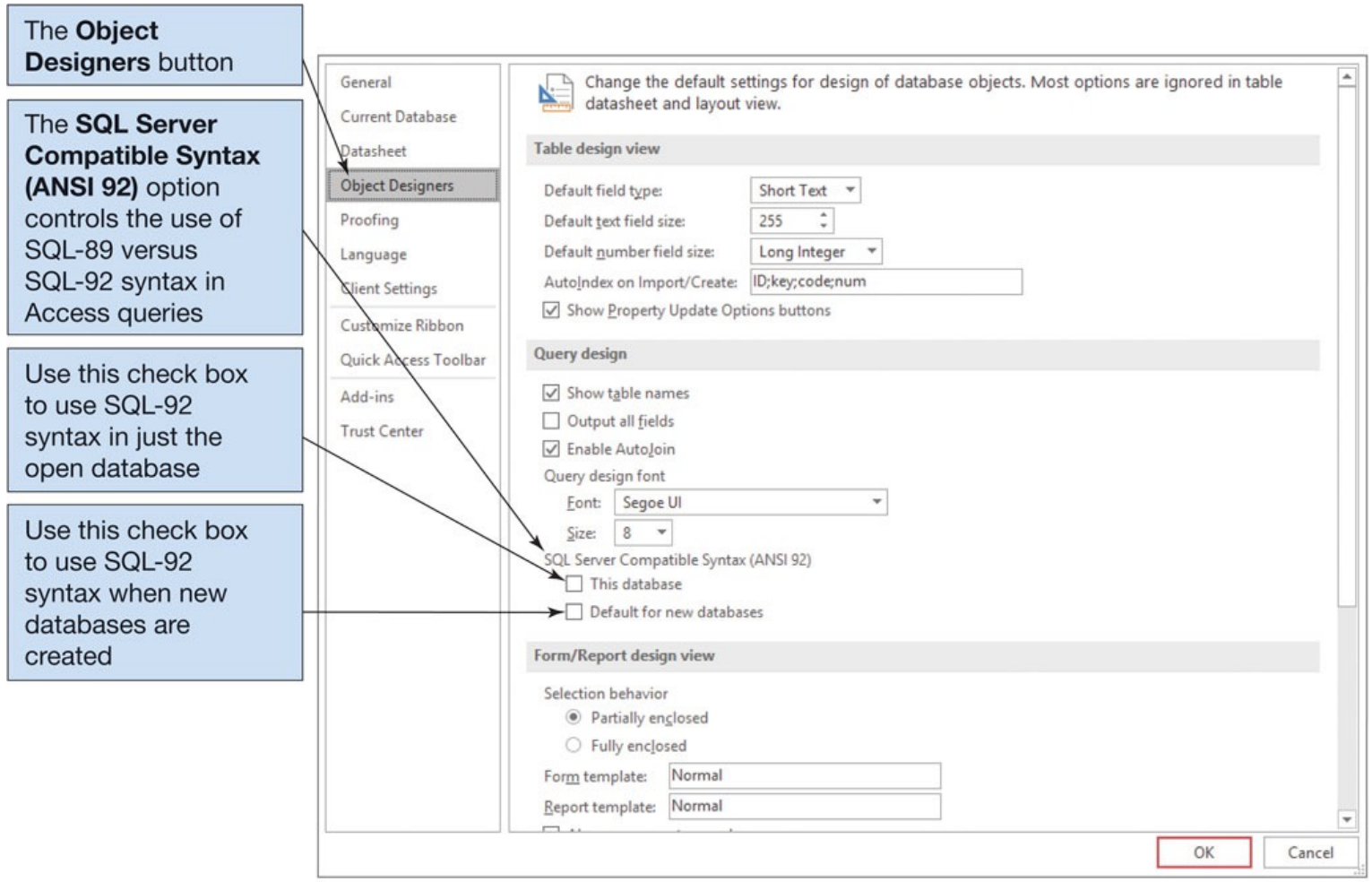
ProjectID	ProjectName	Department	MaxHours	StartDate	EndDate
1000	2019 Q3 Production Plan	Production	100.00	05/10/19	06/15/19
1100	2019 Q3 Marketing Plan	Sales and Marketing	135.00	05/10/19	06/15/19
1200	2019 Q3 Portfolio Analysis	Finance	120.00	07/05/19	07/25/19
1300	2019 Q3 Tax Preparation	Accounting	145.00	08/10/19	10/15/19
1400	2019 Q4 Production Plan	Production	100.00	08/10/19	09/15/19
1500	2019 Q4 Marketing Plan	Sales and Marketing	135.00	08/10/19	09/15/19
1600	2019 Q4 Portfolio Analysis	Finance	140.00	10/05/19	

Figure 3.2(d) Sample Data for the WP Database: ASSIGNMENT Table

ProjectID	EmployeeNumber	HoursWorked
1000	1	30.00
1000	6	50.00
1000	10	50.00
1000	16	75.00
1000	17	75.00
1100	1	30.00
1100	6	75.00

Partial List of Data

Figure 3.3 The Microsoft Access 2019 Options Object Designers Page



Access 2019, Windows 10, Microsoft Corporation

SQL for Data Definition Language (DDL)

Learn basic SQL statements for creating database structures

- The SQL data definition statements include:
  - **CREATE**
    - to create database objects
  - **ALTER**
    - to modify the structure and/or characteristics of database objects
  - **DROP**
    - to delete database objects
  - **TRUNCATE**
    - to delete table data while keeping the structure

SQL CREATE TABLE Statement

Learn basic SQL statements for creating database structures

- The SQL CREATE TABLE statement format is as follows:

```
CREATE TABLE NewTableName (
        ColumnName      DataType        OptionalColumnConstraints,
        ColumnName      DataType        OptionalColumnConstraints,
        ColumnName      DataType        OptionalColumnConstraints,
        optional table constraints
        ...
        );
```

Figure 3.4 SQL CREATE TABLE Statements

```
CREATE  TABLE DEPARTMENT(
     DepartmentName        Char(35)          PRIMARY KEY,
     BudgetCode            Char(30)          NOT NULL,
     OfficeNumber          Char(15)          NOT NULL,
     DepartmentPhone       Char(12)          NOT NULL
     );

CREATE  TABLE EMPLOYEE(
     EmployeeNumber        Int               PRIMARY KEY,
     FirstName             Char(25)          NOT NULL,
     LastName              Char(25)          NOT NULL,
     Department            Char(35)          NOT NULL DEFAULT 'Human Resources',
     Position              Char(35)          NULL,
     Supervisor            Int               NULL,
     OfficePhone           Char(12)          NULL,
     EmailAddress          VarChar(100)      NOT NULL UNIQUE
     );

CREATE  TABLE PROJECT (
     ProjectID             Int               PRIMARY KEY,
     ProjectName           Char(50)          NOT NULL,
     Department            Char(35)          NOT NULL,
     MaxHours              Numeric(8,2)      NOT NULL DEFAULT 100,
     StartDate             Date              NULL,
     EndDate               Date              NULL
     );
```

Create a new database:

- Create the following table using SQL in the query window of MS Access:

```
CREATE  TABLE DEPARTMENT(
     DepartmentName        Char(35)        PRIMARY KEY,
     BudgetCode            Char(30)        NOT NULL,
     OfficeNumber          Char(15)        NOT NULL,
     DepartmentPhone       Char(12)        NOT NULL
     );
```

Create the second table using the following method.

- DO NOT ADD THE AUTO INCREMENT

```
);

CREATE  TABLE EMPLOYEE(
      EmployeeNumber      Int              NOT NULL AUTO_INCREMENT,
      FirstName           Char(25)         NOT NULL,
      LastName            Char(25)         NOT NULL,
      Department          Char(35)         NOT NULL DEFAULT 'Human Resources',
      Position            Char(35)         NULL,
      Supervisor          Int              NULL,
      OfficePhone         Char(12)         NULL,
      EmailAddress        VarChar(100)     NOT NULL UNIQUE,
      CONSTRAINT          EMPLOYEE_PK      PRIMARY KEY(EmployeeNumber),
      CONSTRAINT          EMP_DEPART_FK    FOREIGN KEY(Department)
                             REFERENCES DEPARTMENT(DepartmentName)
                                ON UPDATE CASCADE,
      CONSTRAINT          EMP_SUPER_FK     FOREIGN KEY(Supervisor)
                             REFERENCES EMPLOYEE(EmployeeNumber)

      );
```

Create the remaining tables

```
CREATE  TABLE PROJECT (
      ProjectID           Int              NOT NULL,
      ProjectName         Char(50)         NOT NULL,
      Department          Char(35)         NOT NULL,
      MaxHours            Numeric(8,2)     NOT NULL DEFAULT 100,
      StartDate           Date             NULL,
      EndDate             Date             NULL,
      CONSTRAINT          PROJECT_PK       PRIMARY KEY (ProjectID),
      CONSTRAINT          PROJ_DEPART_FK   FOREIGN KEY(Department)
                             REFERENCES DEPARTMENT(DepartmentName)
                                ON UPDATE CASCADE
      );

CREATE  TABLE ASSIGNMENT (
      ProjectID           Int          NOT NULL,
      EmployeeNumber      Int          NOT NULL,
      HoursWorked         Numeric(6,2) NULL,
      CONSTRAINT          ASSIGNMENT_PK    PRIMARY KEY (ProjectID, EmployeeNumber),
      CONSTRAINT          ASSIGN_PROJ_FK   FOREIGN KEY (ProjectID)
                             REFERENCES PROJECT (ProjectID)
                                ON UPDATE NO ACTION
                                ON DELETE CASCADE,
      CONSTRAINT          ASSIGN_EMP_FK    FOREIGN KEY (EmployeeNumber)
                             REFERENCES EMPLOYEE (EmployeeNumber)
                                ON UPDATE NO ACTION
                                ON DELETE NO ACTION
```

Data Types

- There are many data types available in all DBMS products. To see these you can refer to page 146 in your text.
- Char for character
- Numeric for numeric data
- Dates for date

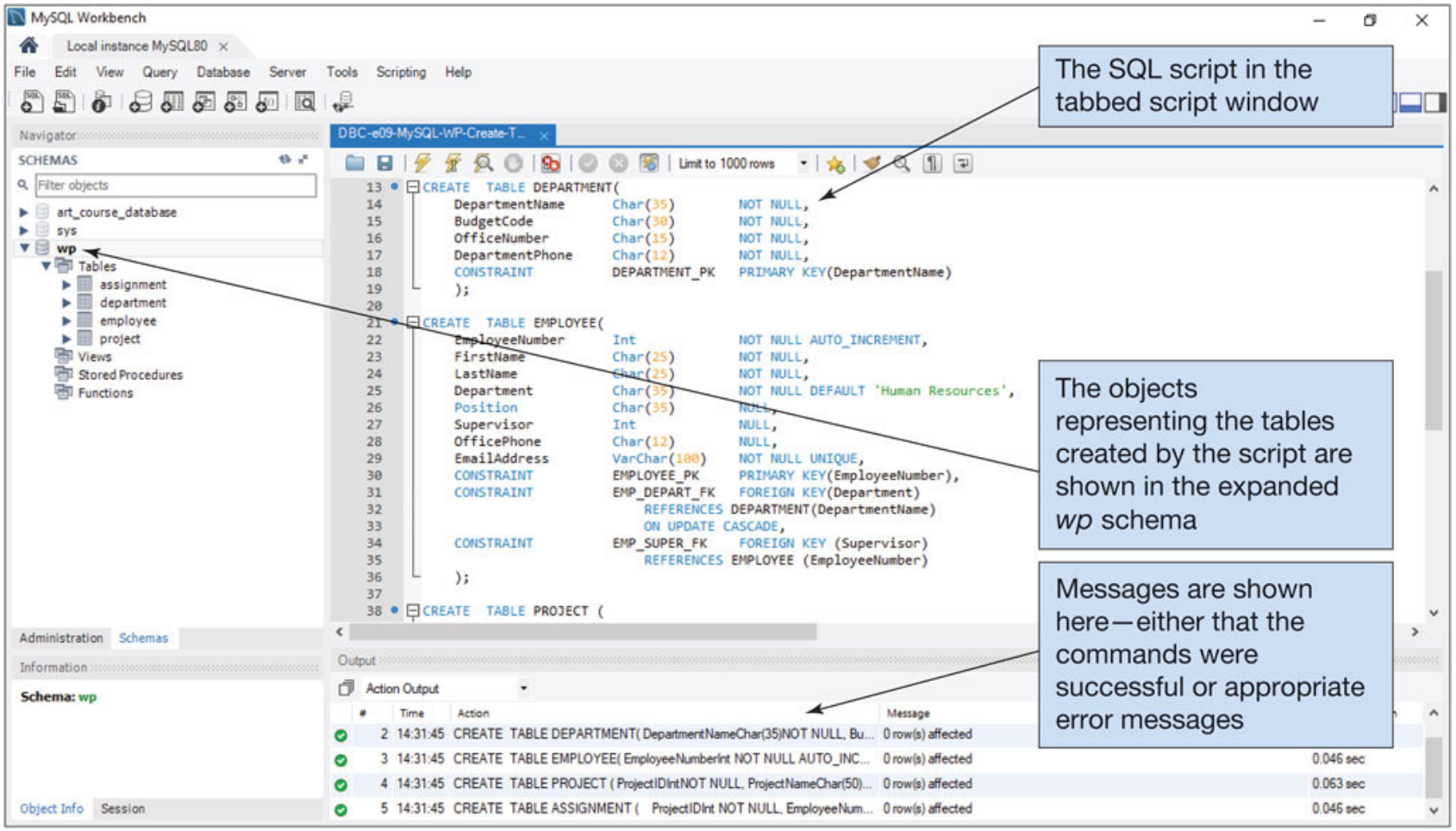
Figure 3.6 Creating Primary Keys with SQL Table Constraints

```
CREATE  TABLE DEPARTMENT(
     DepartmentName        Char(35)          NOT NULL,
     BudgetCode            Char(30)          NOT NULL,
     OfficeNumber          Char(15)          NOT NULL,
     DepartmentPhone       Char(12)          NOT NULL,
     CONSTRAINT            DEPARTMENT_PK     PRIMARY KEY(DepartmentName)
     );

CREATE  TABLE EMPLOYEE(
     EmployeeNumber        Int               NOT NULL AUTO_INCREMENT,
     FirstName             Char(25)          NOT NULL,
     LastName              Char(25)          NOT NULL,
     Department            Char(35)          NOT NULL DEFAULT 'Human Resources',
     Position              Char(35)          NULL,
     Supervisor            Int               NULL,
     OfficePhone           Char(12)          NULL,
     EmailAddress          VarChar(100)      NOT NULL UNIQUE,
     CONSTRAINT            EMPLOYEE_PK       PRIMARY KEY(EmployeeNumber)
     );

CREATE  TABLE PROJECT (
     ProjectID             Int               NOT NULL,
     ProjectName           Char(50)          NOT NULL,
     Department            Char(35)          NOT NULL,
     MaxHours              Numeric(8,2)      NOT NULL DEFAULT 100,
     StartDate             Date              NULL,
     EndDate               Date              NULL,
     CONSTRAINT            PROJECT_PK        PRIMARY KEY (ProjectID)
     );

CREATE  TABLE ASSIGNMENT (
     ProjectID             Int               NOT NULL,
     EmployeeNumber        Int               NOT NULL,
     HoursWorked           Numeric(6,2)      NULL,
     CONSTRAINT            ASSIGNMENT_PK     PRIMARY KEY (ProjectID, EmployeeNumber)
     );
```

Figure 3.7 Creating Foreign Keys with SQL Table Constraints

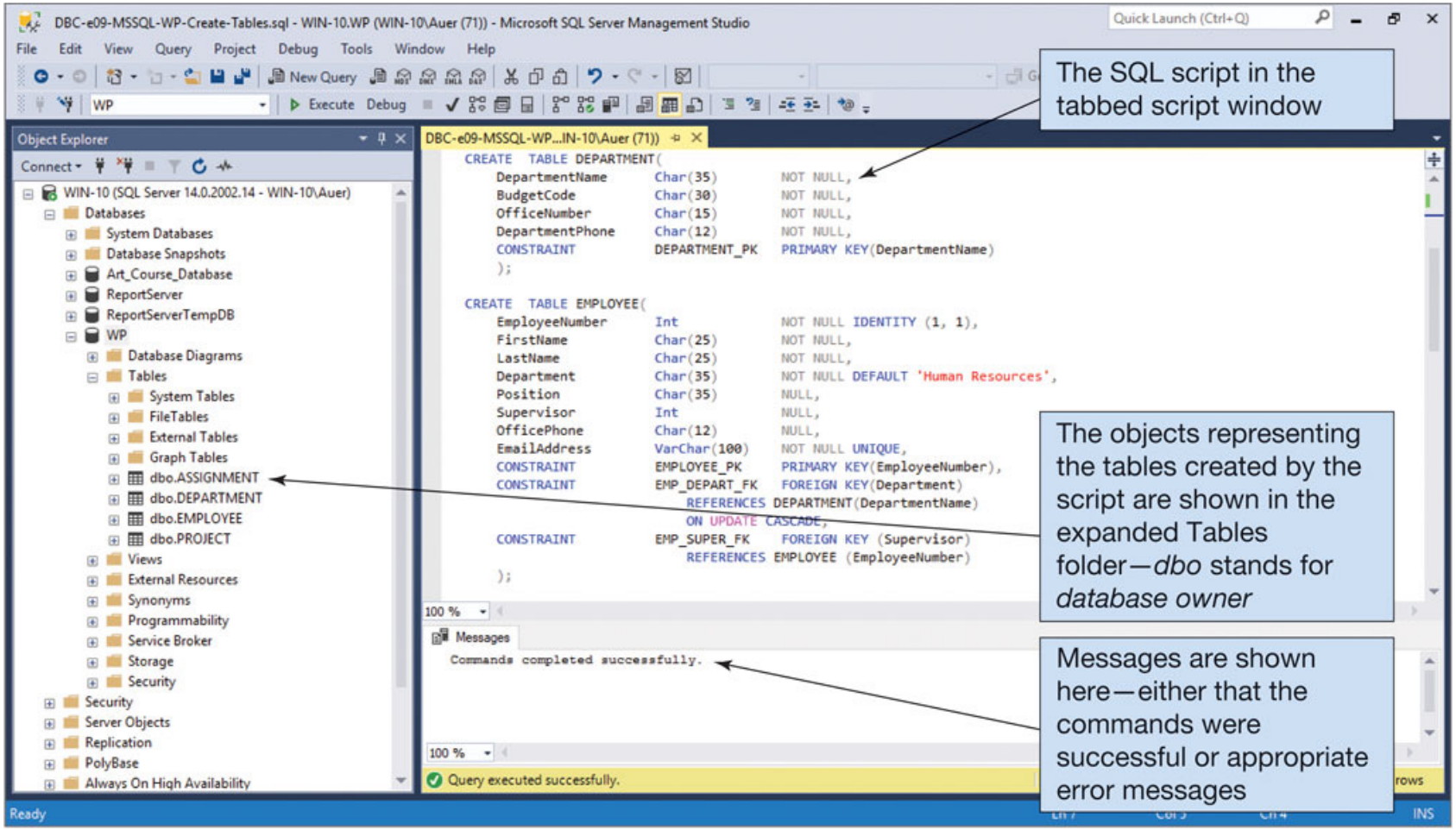
```
CREATE  TABLE DEPARTMENT(
        DepartmentName      Char(35)          NOT NULL,
        BudgetCode          Char(30)          NOT NULL,
        OfficeNumber        Char(15)          NOT NULL,
        DepartmentPhone     Char(12)          NOT NULL,
        CONSTRAINT          DEPARTMENT_PK     PRIMARY KEY(DepartmentName)
        );

CREATE  TABLE EMPLOYEE(
        EmployeeNumber      Int               NOT NULL AUTO_INCREMENT,
        FirstName           Char(25)          NOT NULL,
        LastName            Char(25)          NOT NULL,
        Department          Char(35)          NOT NULL DEFAULT 'Human Resources',
        Position            Char(35)          NULL,
        Supervisor          Int               NULL,
        OfficePhone         Char(12)          NULL,
        EmailAddress        VarChar(100)      NOT NULL UNIQUE,
        CONSTRAINT          EMPLOYEE_PK       PRIMARY KEY(EmployeeNumber),
        CONSTRAINT          EMP_DEPART_FK     FOREIGN KEY(Department)
                               REFERENCES DEPARTMENT(DepartmentName)
                                  ON UPDATE CASCADE,
        CONSTRAINT          EMP_SUPER_FK      FOREIGN KEY(Supervisor)
                               REFERENCES EMPLOYEE(EmployeeNumber)
        );

CREATE  TABLE PROJECT (
        ProjectID           Int               NOT NULL,
        ProjectName         Char(50)          NOT NULL,
        Department          Char(35)          NOT NULL,
        MaxHours            Numeric(8,2)      NOT NULL DEFAULT 100,
        StartDate           Date              NULL,
        EndDate             Date              NULL,
        CONSTRAINT          PROJECT_PK        PRIMARY KEY (ProjectID),
        CONSTRAINT          PROJ_DEPART_FK    FOREIGN KEY(Department)
                               REFERENCES DEPARTMENT(DepartmentName)
                                  ON UPDATE CASCADE
        );

CREATE  TABLE ASSIGNMENT (
        ProjectID           Int            NOT NULL,
        EmployeeNumber      Int            NOT NULL,
        HoursWorked         Numeric(6,2)   NULL,
        CONSTRAINT          ASSIGNMENT_PK     PRIMARY KEY (ProjectID, EmployeeNumber),
        CONSTRAINT          ASSIGN_PROJ_FK    FOREIGN KEY (ProjectID)
                               REFERENCES PROJECT (ProjectID)
                                  ON UPDATE NO ACTION
                                  ON DELETE CASCADE,
        CONSTRAINT          ASSIGN_EMP_FK     FOREIGN KEY (EmployeeNumber)
                               REFERENCES EMPLOYEE (EmployeeNumber)
                                  ON UPDATE NO ACTION
                                  ON DELETE NO ACTION
```

Figure 3.8 Processing the SQL CREATE TABLE Statements Using MySQL 8.0



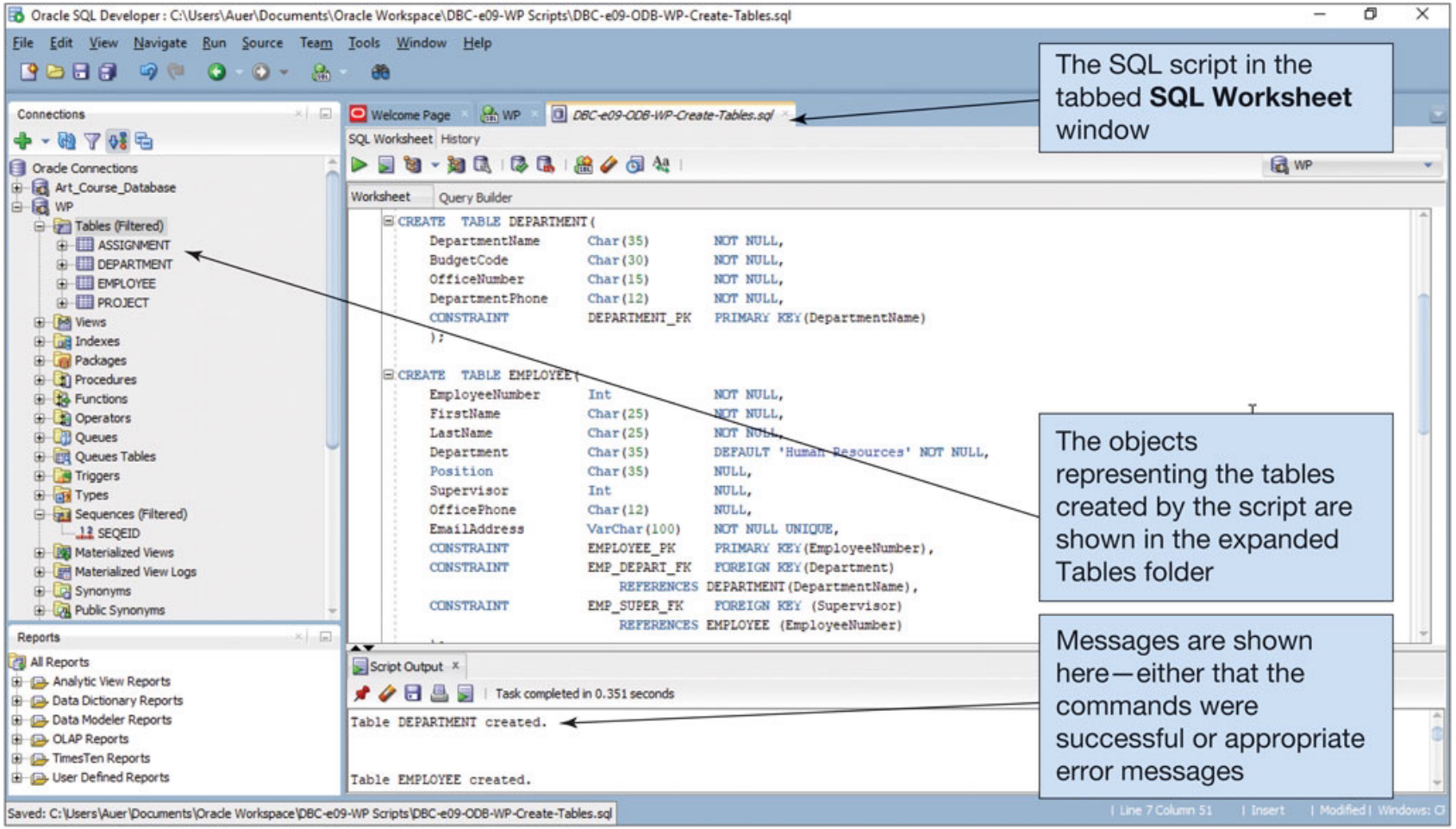
MySQL Community Server 8.0, MySQL Workbench, Oracle Corporation

Figure 3.9 Processing the SQL CREATE TABLE Statements Using MS SQL Server 2017



SQL Server 2017, SQL Server Management Studio, Microsoft Corporation

Figure 3.10 Processing the SQL CREATE TABLE Statements Using Oracle Database XE



Oracle Database XE, SQL Developer 18.4, Oracle Corporation

SQL for Data Manipulation (DML) – Inserting Data

- SQL DML is used to query databases and to modify data in the tables.
- There are three possible data modification operations:
  - INSERT (adding data to a relation)
  - UPDATE (modifying data in a relation)
  - DELETE (deleting data in a relation)

Inserting Data

Learn basic SQL SELECT for adding data to a database

```
/* *** SQL-INSERT-CH03-01 *** */
INSERT INTO DEPARTMENT VALUES('Administration',
'BC-100-10', 'BLDG01-210, '360-285-8100');
```

The order of the columns on the INSERT command does not matter as long as the values match the order of the columns as seen below.

```
/* *** SQL-INSERT-CH03-05 *** */
INSERT INTO EMPLOYEE(EmailAddress, FirstName,          LastName,
        Department, Position, OfficePhone)
        VALUES(
        'Mary.Jacobs@WP.com', 'Mary', 'Jacobs',
        'Administration', 'CEO', '360-285-8110');
```

The SQL Query Framework

Learn basic SQL SELECT statements and options for processing a single table

- The basic framework for the SQL Query statement has three statements as follows:
  - the **SQL SELECT** clause
    - specifies which *columns* are to be listed in the query results
  - the **SQL FROM** clause
    - specifies which *tables* are to be used in the query
  - the **SQL WHERE** clause
    - specifies which *rows* are to be listed in the query results

Reading Specified Columns from a Single Table

Learn basic SQL SELECT statements and options for processing a single table

```
/* *** SQL-Query-CH03-01 *** */
SELECT  ProjectID, ProjectName, Department, MaxHours,StartDate,  EndDate
FROM  PROJECT;
```

	ProjectID	ProjectName	Department	MaxHours	StartDate	EndDate
▶	1000	2019 Q3 Production Plan	Production	100.00	2019-05-10	2019-06-15
	1100	2019 Q3 Marketing Plan	Sales and Marketing	135.00	2019-05-10	2019-06-15
	1200	2019 Q3 Portfolio Analysis	Finance	120.00	2019-07-05	2019-07-25
	1300	2019 Q3 Tax Preparation	Accounting	145.00	2019-08-10	2019-10-15
	1400	2019 Q4 Production Plan	Production	100.00	2019-08-10	2019-09-15
	1500	2019 Q4 Marketing Plan	Sales and Marketing	135.00	2019-08-10	2019-09-15
	1600	2019 Q4 Portfolio Analysis	Finance	140.00	2019-10-05	NULL
*	NULL	NULL	NULL	NULL	NULL	NULL

MySQL Community Server 8.0, MySQL Workbench, Oracle Corporation

Single Table Queries Displaying All Columns

Learn basic SQL SELECT statements and options for processing a single table

```
/* *** SQL-Query-CH03-02 *** */
SELECT          *
FROM            PROJECT;
```

ProjectID	ProjectName	Department	MaxHours	StartDate	EndDate
1000	2019 Q3 Production Plan	Production	100.00	2019-05-10	2019-06-15
1100	2019 Q3 Marketing Plan	Sales and Marketing	135.00	2019-05-10	2019-06-15
1200	2019 Q3 Portfolio Analysis	Finance	120.00	2019-07-05	2019-07-25
1300	2019 Q3 Tax Preparation	Accounting	145.00	2019-08-10	2019-10-15
1400	2019 Q4 Production Plan	Production	100.00	2019-08-10	2019-09-15
1500	2019 Q4 Marketing Plan	Sales and Marketing	135.00	2019-08-10	2019-09-15
1600	2019 Q4 Portfolio Analysis	Finance	140.00	2019-10-05	NULL

MySQL Community Server 8.0, MySQL Workbench, Oracle Corporation

Single Table Queries Specifying Column Order

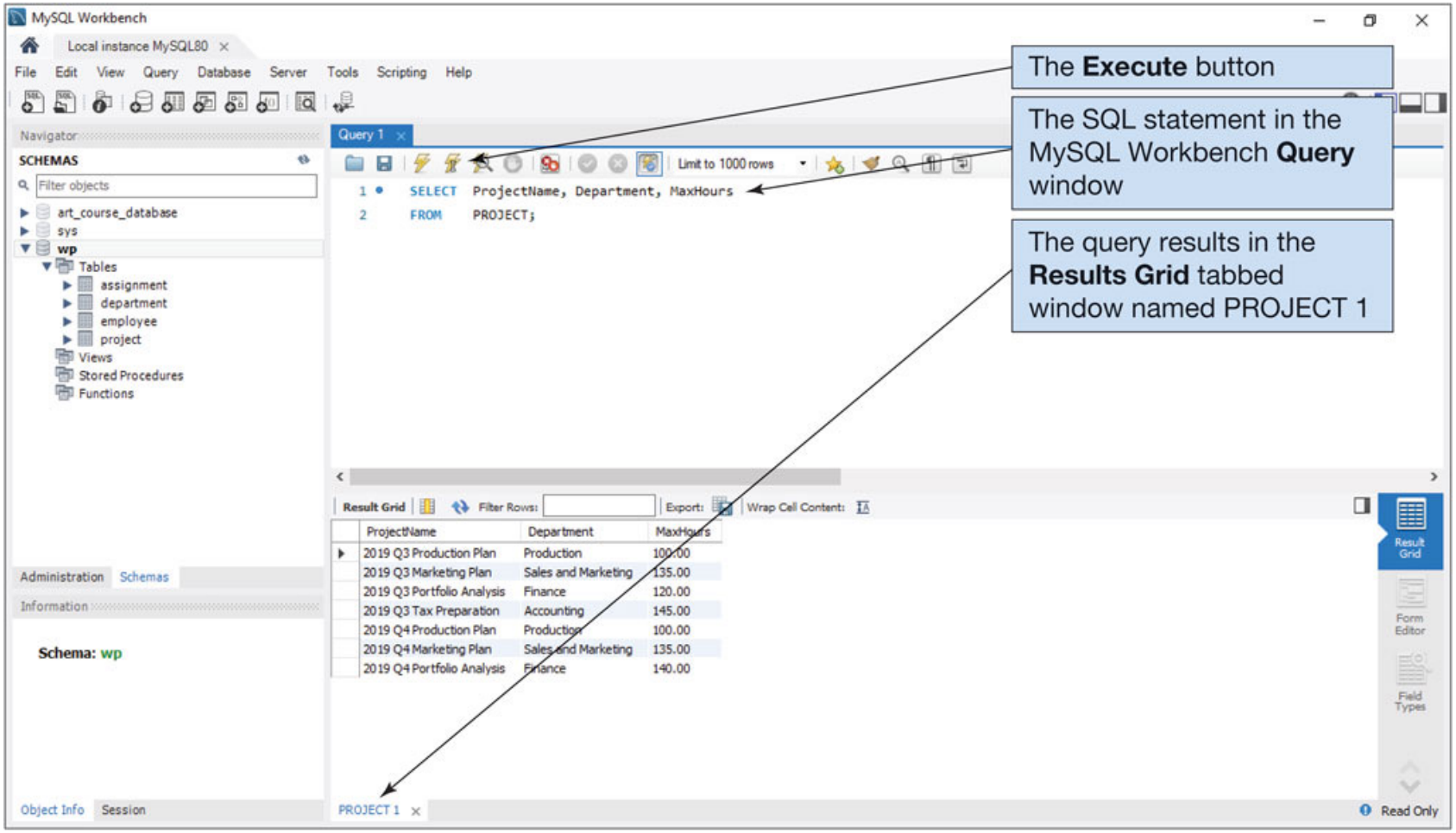
Learn basic SQL SELECT statements and options for processing a single table

```
/* *** SQL-Query-CH03-02 *** */
SELECT          ProjectName, Department, MaxHours
FROM            PROJECT;
```

	ProjectName	Department	MaxHours
▶	2019 Q3 Production Plan	Production	100.00
	2019 Q3 Marketing Plan	Sales and Marketing	135.00
	2019 Q3 Portfolio Analysis	Finance	120.00
	2019 Q3 Tax Preparation	Accounting	145.00
	2019 Q4 Production Plan	Production	100.00
	2019 Q4 Marketing Plan	Sales and Marketing	135.00
	2019 Q4 Portfolio Analysis	Finance	140.00

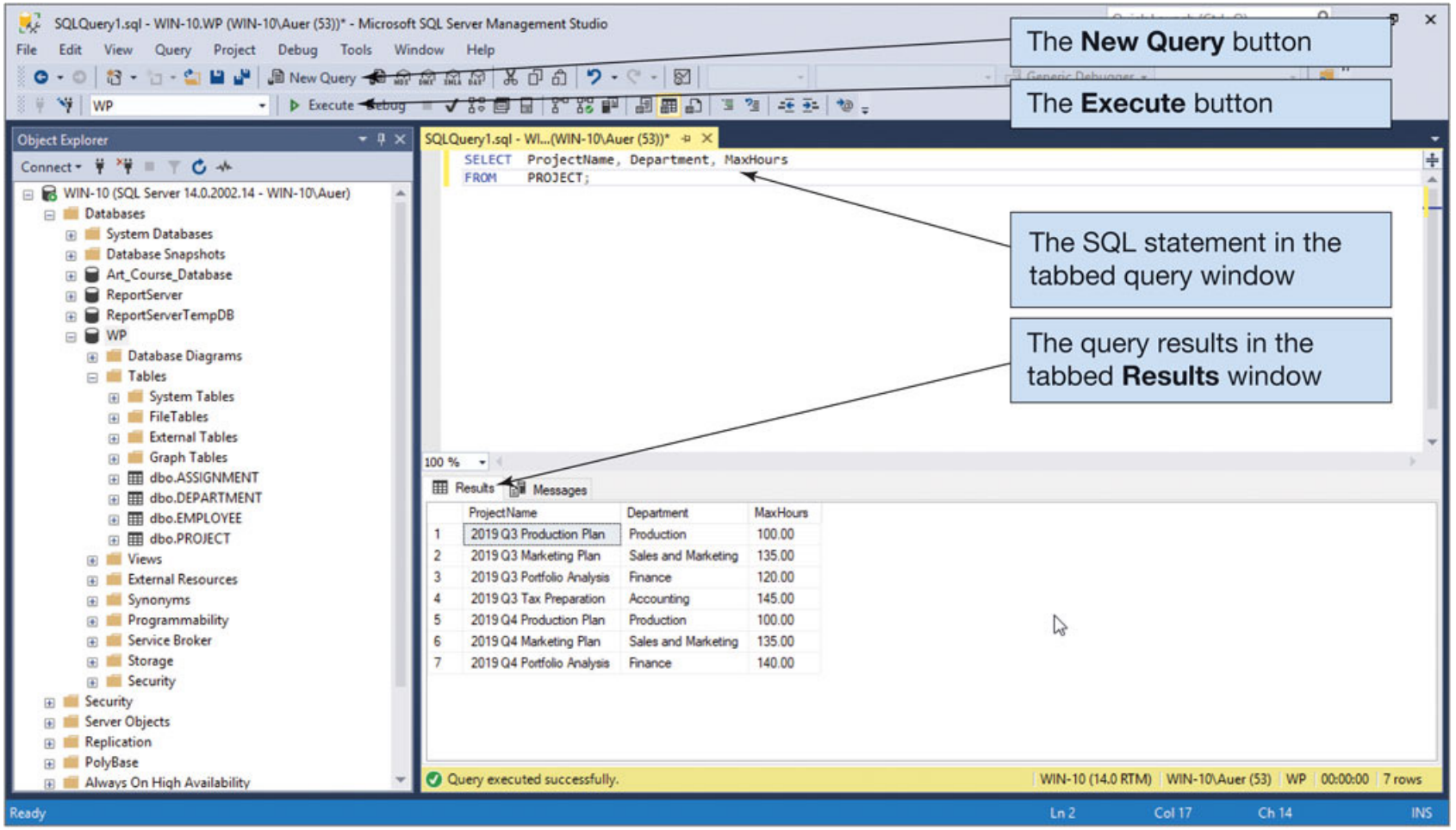
MySQL Community Server 8.0, MySQL Workbench, Oracle Corporation

Figure 3.12 SQL Query Results in the MySQL Workbench



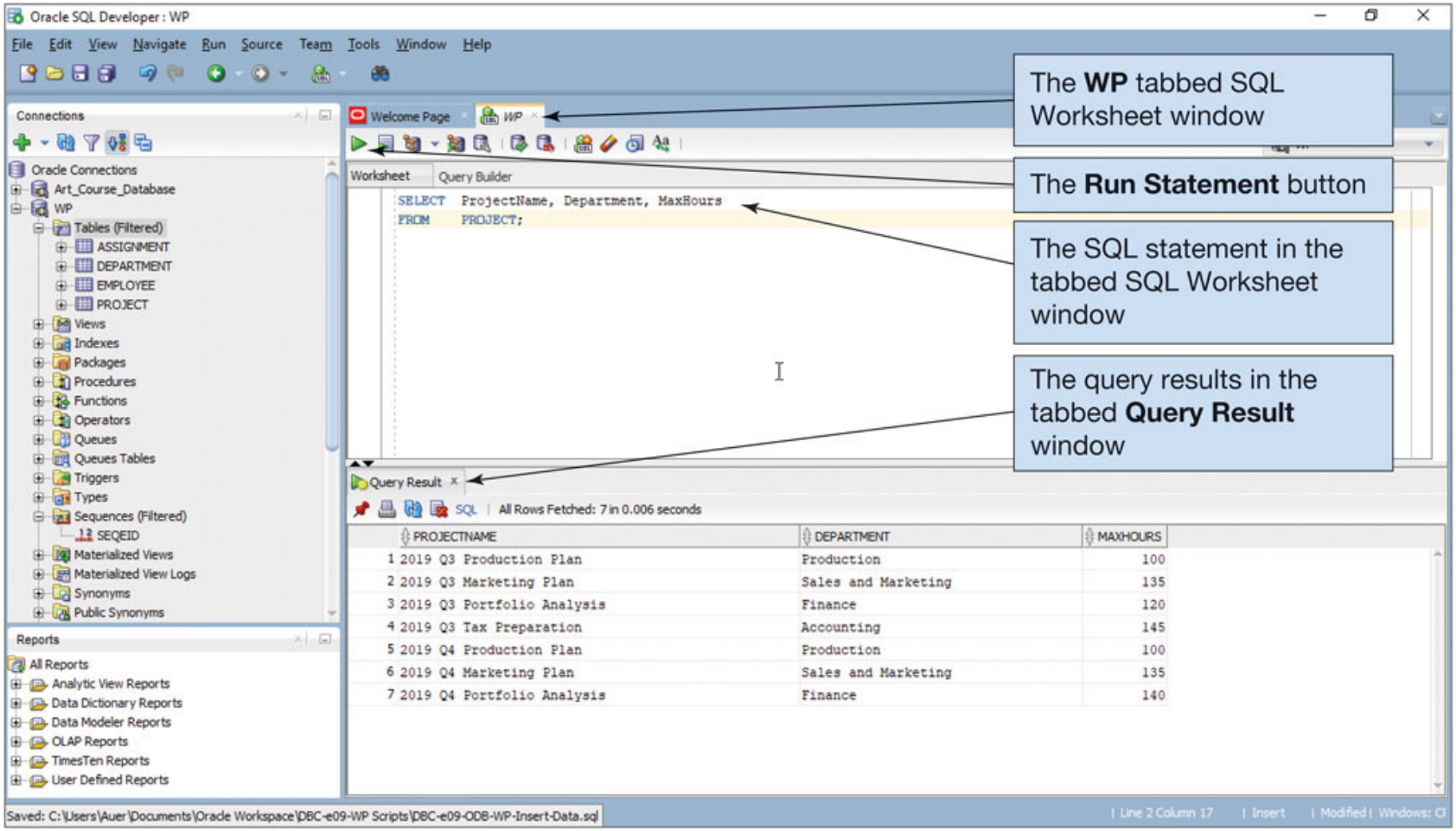
MySQL Community Server 8.0, MySQL Workbench, Oracle Corporation

Figure 3.13 SQL Query Results in the Microsoft SQL Server Management Studio



SQL Server 2017, SQL Server Management Studio, Microsoft Corporation

Figure 3.14 SQL Query Results in the Oracle SQL Developer



Oracle Database XE, SQL Developer 18.4, Oracle Corporation

Reading Specified Rows from a Single Table

Learn basic SQL SELECT statements and options for processing a single table

```
/* *** SQL-Query-CH03-05 *** */
SELECT Department
FROM   PROJECT;
```

	Department
▶	Accounting
	Finance
	Finance
	Production
	Production
	Sales and Marketing
	Sales and Marketing

MySQL Community Server 8.0, MySQL Workbench, Oracle Corporation

Single Table Queries The DISTINCT Keyword

Learn basic SQL SELECT statements and options for processing a single table

```
/* *** SQL-Query-CH03-05 *** */
SELECT Department
FROM   PROJECT;
```

Department
Accounting
Finance
Production
Sales and Marketing

MySQL Community Server 8.0, MySQL Workbench, Oracle Corporation

Single Table Queries The WHERE Clause

Learn basic SQL SELECT statements and options for processing a single table

```
/* *** SQL-Query-CH03-07 *** */
SELECT    *
FROM      PROJECT
WHERE     Department = 'Finance';
```

	ProjectID	ProjectName	Department	MaxHours	StartDate	EndDate
▶	1200	2019 Q3 Portfolio Analysis	Finance	120.00	2019-07-05	2019-07-25
	1600	2019 Q4 Portfolio Analysis	Finance	140.00	2019-10-05	NULL

MySQL Community Server 8.0, MySQL Workbench, Oracle Corporation

Figure 3.15 SQL Comparison Operators

SQL Comparison Operators

Operator	Meaning
=	Is equal to
<>	Is NOT Equal to
<	Is less than
>	Is greater than
<=	Is less than OR equal to
>=	Is greater than OR equal to
IN	Is equal to one of a set of values
NOT IN	Is NOT equal to any of a set of values
BETWEEN	Is within a range of numbers (includes the end points)
NOT BETWEEN	Is NOT within a range of numbers (includes the end points)
LIKE	Matches a set of characters
NOT LIKE	Does NOT match a set of characters
IS NULL	Is equal to NULL
IS NOT NULL	Is NOT equal to NULL

Single Table Queries Using Comparison Operators

Learn basic SQL SELECT statements and options for processing a single table

```
/* *** SQL-Query-CH03-09 *** */
SELECT*
FROM   PROJECT
WHERE MaxHours > 135;
```

	ProjectID	ProjectName	Department	MaxHours	StartDate	EndDate
▶	1300	2019 Q3 Tax Preparation	Accounting	145.00	2019-08-10	2019-10-15
	1600	2019 Q4 Portfolio Analysis	Finance	140.00	2019-10-05	NULL

MySQL Community Server 8.0, MySQL Workbench, Oracle Corporation

Single Table Queries Reading Specified Columns and Rows

Learn basic SQL SELECT statements and options for processing a single table

```
/* *** SQL-Query-CH03-10 *** */
SELECT FirstName, LastName, Department, OfficePhone
FROM   EMPLOYEE
WHERE Department = 'Accounting';
```

	FirstName	LastName	Department	OfficePhone
▶	Tom	Caruthers	Accounting	360-285-8430
	Heather	Jones	Accounting	360-285-8440

MySQL Community Server 8.0, MySQL Workbench, Oracle Corporation

Single Table Queries Sorting the Results of a Query

Learn basic SQL SELECT statements and options for processing a single table

```
/* *** SQL-Query-CH03-11 *** */
SELECT   FirstName, LastName,
Department, OfficePhone
FROM     EMPLOYEE
ORDER BY Department;
```

By default, SQL Server sorts in ascending order. If you need the sort in descending order it would be:

ORDER BY Department DESC;

FirstName	LastName	Department	OfficePhone
Tom	Caruthers	Accounting	360-285-8430
Heather	Jones	Accounting	360-285-8440
Mary	Jacobs	Administration	360-285-8110
Rosalie	Jackson	Administration	360-285-8120
Ken	Evans	Finance	360-285-8410
Mary	Abernathy	Finance	360-285-8420
George	Smith	Human Resources	360-285-8310
Alan	Adams	Human Resources	360-285-8320
James	Nestor	InfoSystems	360-285-8610
Rick	Brown	InfoSystems	NULL
Richard	Bandalone	Legal	360-285-8210
Mary	Smith	Production	360-285-8810
Tom	Jackson	Production	360-285-8820
George	Jones	Production	360-285-8830
Julia	Hayakawa	Production	NULL
Sam	Stewart	Production	NULL
Mike	Nguyen	Research and Development	360-285-8710
Jason	Sleeman	Research and Development	360-285-8720
Ken	Numoto	Sales and Marketing	360-285-8510
Linda	Granger	Sales and Marketing	360-285-8520

Single Table Queries Sorting by Multiple Columns

Learn basic SQL SELECT statements and options for processing a single table

```
/* *** SQL-Query-CH03-13 *** */
SELECT    FirstName, LastName, Department, OfficePhone
FROM      EMPLOYEE
ORDER BY  Department DESC, LastName ASC;
```

The result is:

	FirstName	LastName	Department	OfficePhone
▶	Linda	Granger	Sales and Marketing	360-285-8520
	Ken	Numoto	Sales and Marketing	360-285-8510
	Mike	Nguyen	Research and Development	360-285-8710
	Jason	Sleeman	Research and Development	360-285-8720
	Julia	Hayakawa	Production	NULL
	Tom	Jackson	Production	360-285-8820
	George	Jones	Production	360-285-8830
	Mary	Smith	Production	360-285-8810
	Sam	Stewart	Production	NULL
	Richard	Bandalone	Legal	360-285-8210
	Rick	Brown	InfoSystems	NULL
	James	Nestor	InfoSystems	360-285-8610
	Alan	Adams	Human Resources	360-285-8320
	George	Smith	Human Resources	360-285-8310
	Mary	Abernathy	Finance	360-285-8420
	Ken	Evans	Finance	360-285-8410
	Rosalie	Jackson	Administration	360-285-8120
	Mary	Jacobs	Administration	360-285-8110
	Tom	Caruthers	Accounting	360-285-8430
	Heather	Jones	Accounting	360-285-8440

MySQL Community Server 8.0, MySQL Workbench, Oracle Corporation.

Single Table Queries SQL WHERE Clause Options

Learn basic SQL SELECT statements and options for processing a single table

- Three options for the SQL WHERE clauses are:
  - compound clauses
  - ranges
  - wildcards
- An example of the compound clause is below:

```
/* *** SQL-Query-CH03-14 *** */
SELECT     FirstName, LastName, Department, OfficePhone
FROM       EMPLOYEE
WHERE      Department = 'Accounting'
    AND    OfficePhone = '360-285-8430';
```

Figure 3.16 SQL Logical Operators

SQL Logical Operators

Operator	Meaning
AND	Both arguments are TRUE
OR	One or the other or both of the arguments are TRUE
NOT	Negates the associated operator

Single Table Queries: SQL WHERE Clauses Using Ranges of Values

Learn basic SQL SELECT statements and options for processing a single table

```
/* *** SQL-Query-CH03-19 *** */
SELECT   FirstName, LastName, Department, OfficePhone
FROM     EMPLOYEE
WHERE    EmployeeNumber >= 2
   AND   EmployeeNumber <= 5;
```

	FirstName	LastName	Department	OfficePhone
▶	Rosalie	Jackson	Administration	360-285-8120
	Richard	Bandalone	Legal	360-285-8210
	George	Smith	Human Resources	360-285-8310
	Alan	Adams	Human Resources	360-285-8320

MySQL Community Server 8.0, MySQL Workbench, Oracle Corporation.

Single Table Queries: SQL WHERE Clauses Using Wildcards

Learn basic SQL SELECT statements and options for processing a single table

When using an underscore it means that any character can occur in the spot occupied by the underscore.

```
/* *** SQL-Query-CH03-21 *** */
SELECT    *
FROM      PROJECT
WHERE     ProjectName LIKE '2019 Q_ Portfolio Analysis';
```

Notice below that an underscore is used for each space needed.

```
/* *** SQL-Query-CH03-22 *** */
SELECT    *
FROM      EMPLOYEE
WHERE     OfficePhone LIKE '360-285-88__';
```

Single Table Queries: SQL WHERE Clauses That Use NULL Values

Learn basic SQL SELECT statements and options for processing a single table

To include or exclude rows that contain NULL values, we use the IS NULL or IS NOT NULL comparison values.

```
/* *** SQL-Query-CH03-26 *** */
SELECT   FirstName, LastName, Department, OfficePhone
FROM     EMPLOYEE
WHERE    OfficePhone IS NULL;

/* *** SQL-Query-CH03-27 *** */
SELECT   FirstName, LastName, Department, OfficePhone
FROM     MPLOYEE
WHERE    OfficePhone IS NOT NULL;
```

Single Table Queries: SQL Queries That Perform Calculations

Learn basic SQL SELECT statements and options for processing a single table

- It is possible to perform certain types of calculations in SQL query statements.
  - One group of calculations involves the use of SQL built-in functions.
  - Another group involves simple arithmetic operations on the columns in the SELECT statement.

Figure 3.17 SQL Built-in Aggregate Functions

SQL Built-in Aggregate Functions

Function	Meaning
COUNT(*)	Count the number of rows in the table
COUNT ({Name})	Count the number of rows in the table where column {Name} IS NOT NULL
SUM	Calculate the sum of all values (numeric columns only)
AVG	Calculate the average of all values (numeric columns only)
MIN	Calculate the minimum value of all values
MAX	Calculate the maximum value of all values

Single Table Queries: Using SQL Built-in Aggregate Functions

Learn basic SQL SELECT statements and options for processing a single table

```
/* *** SQL-Query-CH03-28 *** */
SELECT    COUNT(*)
FROM      PROJECT;

/* *** SQL-Query-CH03-29 *** */
SELECT    COUNT(*) AS NumberOfProjects
FROM      PROJECT;
```

	NumberOfProjects
▶	7

Single Table Queries: Using Multiple Aggregate Functions in an SQL Statement

Learn basic SQL SELECT statements and options for processing a single table

```
/* *** SQL-Query-CH03-32 *** */
SELECT   SUM(MaxHours) AS TotalMaxHours,
         AVG(MaxHours) AS AverageMaxHours,
         MIN(MaxHours) AS MinimumMaxHours,
         MAX(MaxHours) AS MaximumMaxHours
FROM     PROJECT
WHERE    ProjectID <= 1200;
```

	TotalMaxHours	AverageMaxHours	MinmumMaxHours	MaximumMaxHours
▶	355.00	118.333333	100.00	135.00

Single Table Queries: Using Expressions in SQL SELECT Statements

Learn basic SQL SELECT statements and options for processing a single table

```
/* *** SQL-Query-CH03-35 *** */
SELECT    ProjectID, ProjectName, MaxHours,
          (24.50 * MaxHours) AS MaxProjectCost
FROM      PROJECT;
```

	ProjectID	ProjectName	MaxHours	MaxProjectCost
▶	1000	2019 Q3 Production Plan	100.00	2450.0000
	1100	2019 Q3 Marketing Plan	135.00	3307.5000
	1200	2019 Q3 Portfolio Analysis	120.00	2940.0000
	1300	2019 Q3 Tax Preparation	145.00	3552.5000
	1400	2019 Q4 Production Plan	100.00	2450.0000
	1500	2019 Q4 Marketing Plan	135.00	3307.5000
	1600	2019 Q4 Portfolio Analysis	140.00	3430.0000

Single Table Queries: Grouping Rows Using SQL SELECT Statements

Learn basic SQL SELECT statements and options for processing a single table

- In SQL, you can use the SQL GROUP BY clause to group rows by common values.
  - This is a powerful feature, but it can be difficult to understand.
  - What do we mean by a group? Consider the EMPLOYEE table in Figure 3.18 on the next slide, where we show the employees grouped by which department they work in.
  - Because there are 9 departments, we have the employees divided into 9 groups.

Figure 3.18 Department Groups in the EMPLOYEE Table

This group of rows is employees in the **Administration** department

This group of rows is employees in the **Production** department

EmployeeNumber	FirstName	LastName	Department	Position	Supervisor	OfficePhone	EmailAddress
1	Mary	Jacobs	Administration	CEO	NULL	360-285-8110	Mary.Jacobs@WP.com
2	Rosalie	Jackson	Administration	Admin Assistant	1	360-285-8120	Rosalie.Jackson@WP.com
3	Richard	Bandalone	Legal	Attorney	1	360-285-8210	Richard.Bandalone@WP.com
4	George	Smith	Human Resources	HR3	1	360-285-8310	George.Smith@WP.com
5	Alan	Adams	Human Resources	HR1	4	360-285-8320	Alan.Adams@WP.com
6	Ken	Evans	Finance	CFO	1	360-285-8410	Ken.Evans@WP.com
7	Mary	Abernathy	Finance	FA3	6	360-285-8420	Mary.Abernathy@WP.com
8	Tom	Caruthers	Accounting	FA2	6	360-285-8430	Tom.Caruthers@WP.com
9	Heather	Jones	Accounting	FA2	6	360-285-8440	Heather.Jones@WP.com
10	Ken	Numoto	Sales and Marketing	SM3	1	360-285-8510	Ken.Numoto@WP.com
11	Linda	Granger	Sales and Marketing	SM2	10	360-285-8520	Linda.Granger@WP.com
12	James	Nestor	InfoSystems	CIO	1	360-285-8610	James.Nestor@WP.com
13	Rick	Brown	InfoSystems	IS2	12	NULL	Rick.Brown@WP.com
14	Mike	Nguyen	Research and Development	CTO	1	360-285-8710	Mike.Nguyen@WP.com
15	Jason	Sleeman	Research and Development	RD3	14	360-285-8720	Jason.Sleeman@WP.com
16	Mary	Smith	Production	OPS3	1	360-285-8810	Mary.Smith@WP.com
17	Tom	Jackson	Production	OPS2	16	360-285-8820	Tom.Jackson@WP.com
18	George	Jones	Production	OPS2	17	360-285-8830	George.Jones@WP.com
19	Julia	Hayakawa	Production	OPS1	17	NULL	Julia.Hayakawa@WP.com
20	Sam	Stewart	Production	OPS1	17	NULL	Sam.Stewart@WP.com

MySQL Community Server 8.0, MySQL Workbench, Oracle Corporation.

Single Table Queries: Using the GROUP BY Clause

Learn basic SQL SELECT statements and options for processing a single table

```
/* *** SQL-Query-CH03-36 *** */
SELECT      Department, Count(*) AS NumberOfEmployees
FROM        EMPLOYEE
GROUP BY  Department;
```

	Department	NumberOfEmployees
▶	Accounting	2
	Administration	2
	Finance	2
	Human Resources	2
	InfoSystems	2
	Legal	1
	Production	5
	Research and Development	2
	Sales and Marketing	2

MySQL Community Server 8.0, MySQL Workbench, Oracle Corporation.

Single Table Queries: Using the GROUP BY Clause with the HAVING Clause

Learn basic SQL SELECT statements and options for processing a single table

```
/* *** SQL-Query-CH03-36 *** */
SELECT      Department, Count(*) AS NumberOfEmployees
FROM        EMPLOYEE
GROUP BY    Department
HAVING      COUNT(*) > 1;
```

Department	NumberOfEmployees
Accounting	2
Administration	2
Finance	2
Human Resources	2
InfoSystems	2
Production	5
Research and Development	2
Sales and Marketing	2

MySQL Community Server 8.0, MySQL Workbench, Oracle Corporation.

Multiple Table Queries: SQL For Data Manipulation (DML)

Learn basic SQL SELECT Statements for processing multiple tables with subqueries

- The queries considered so far have involved data from a single table. However, at times, more than one table must be processed to obtain the desired information. SQL provides two different techniques for querying data from multiple tables:
  - the SQL Subquery
  - the SQL Join

Multiple Table Queries: Querying with Subqueries

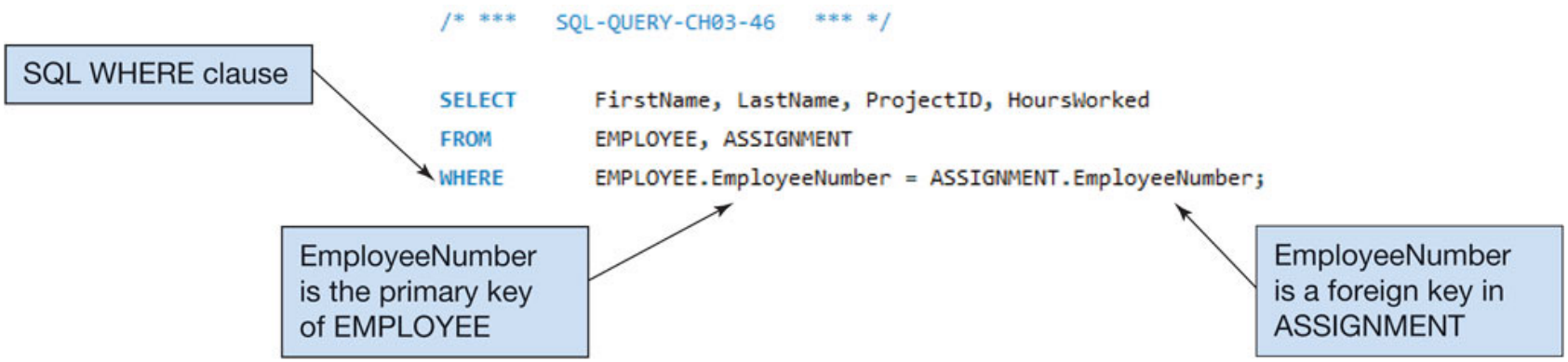
Learn basic SQL SELECT Statements for processing multiple tables with subqueries

```
/* *** SQL-Query-CH03-41  *** */
SELECT          FirstName, LastName
FROM            EMPLOYEE
WHERE           EmployeeNumber  IN
                (SELECT DISTINCT  EmployeeNumber
                 FROM ASSIGNMENT
                 WHERE HoursWorked  > 50);
```

	FirstName	LastName
▶	Ken	Evans
	Ken	Numoto
	Linda	Granger
	Mary	Smith
	Tom	Jackson

MySQL Community Server 8.0, MySQL Workbench, Oracle Corporation.

Figure 3.19 Using Primary Key and Foreign Key Values in the SQL WHERE Clause in an Implicit SQL JOIN



MySQL Community Server 8.0, MySQL Workbench, Oracle Corporation.

Multiple Table Queries: Querying with Joins (1 of 2)

Learn basic SQL SELECT Statements for processing multiple tables with Joins

```
/* *** SQL-Query-CH03-46  *** */
SELECT    FirstName, LastName, ProjectID, HoursWorked
FROM      EMPLOYEE, ASSIGNMENT
WHERE     EMPLOYEE.EmployeeNumber  = ASSIGNMENT.EmployeeNumber;
```

	FirstName	LastName	ProjectID	HoursWorked
▶	Mary	Jacobs	1000	30.00
	Mary	Jacobs	1100	30.00

MySQL Community Server 8.0, MySQL Workbench, Oracle Corporation.

Multiple Table Queries: Querying with Joins (2 of 2)

Learn basic SQL SELECT Statements for processing multiple tables with Joins

```
/* *** SQL-Query-CH03-46  *** */
SELECT    FirstName, LastName, ProjectID, HoursWorked
FROM      EMPLOYEE, ASSIGNMENT
WHERE     EMPLOYEE.EmployeeNumber  = ASSIGNMENT.EmployeeNumber;
```

	FirstName	LastName	ProjectID	HoursWorked
▶	Mary	Jacobs	1000	30.00
	Mary	Jacobs	1100	30.00
	Mary	Jacobs	1400	30.00
	Mary	Jacobs	1500	30.00

MySQL Community Server 8.0, MySQL Workbench, Oracle Corporation.

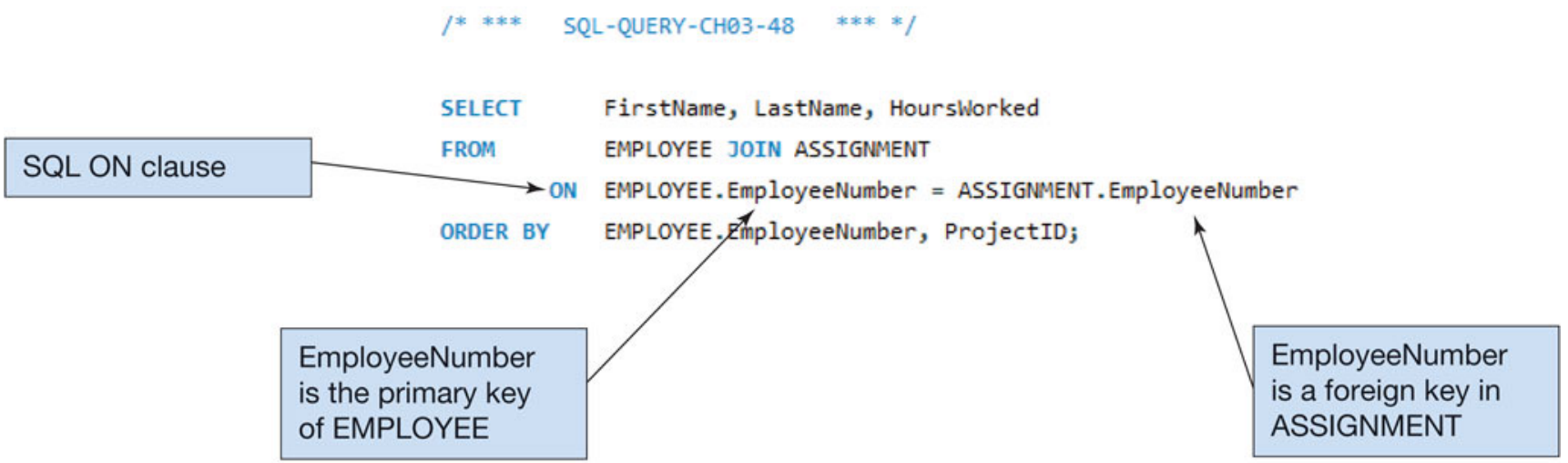
Multiple Table Queries: Using the JOIN ON with the GROUP BY

Learn basic SQL SELECT Statements for processing multiple tables with Joins

The JOIN ON syntax still requires a statement of primary key to foreign key as shown below:

```
/* *** SQL-Query-CH03-49 *** */
SELECT    FirstName, LastName, SUM(HoursWorked) AS TotalHoursWorked
FROM      EMPLOYEE AS E JOIN ASSIGNMENT AS A
    ON    E.EmployeeNumber = A.EmployeeNumber
GROUP BY LastName, FirstName
ORDER BY LastName, FirstName;
```

Figure 3.22 Using Primary Key and Foreign Key Values in the SQL ON Clause in an Explicit SQL Join



MySQL Community Server 8.0, MySQL Workbench, Oracle Corporation.

Multiple Table Queries: Comparing Subqueries and Joins

Learn basic SQL SELECT Statements for processing multiple tables with Joins

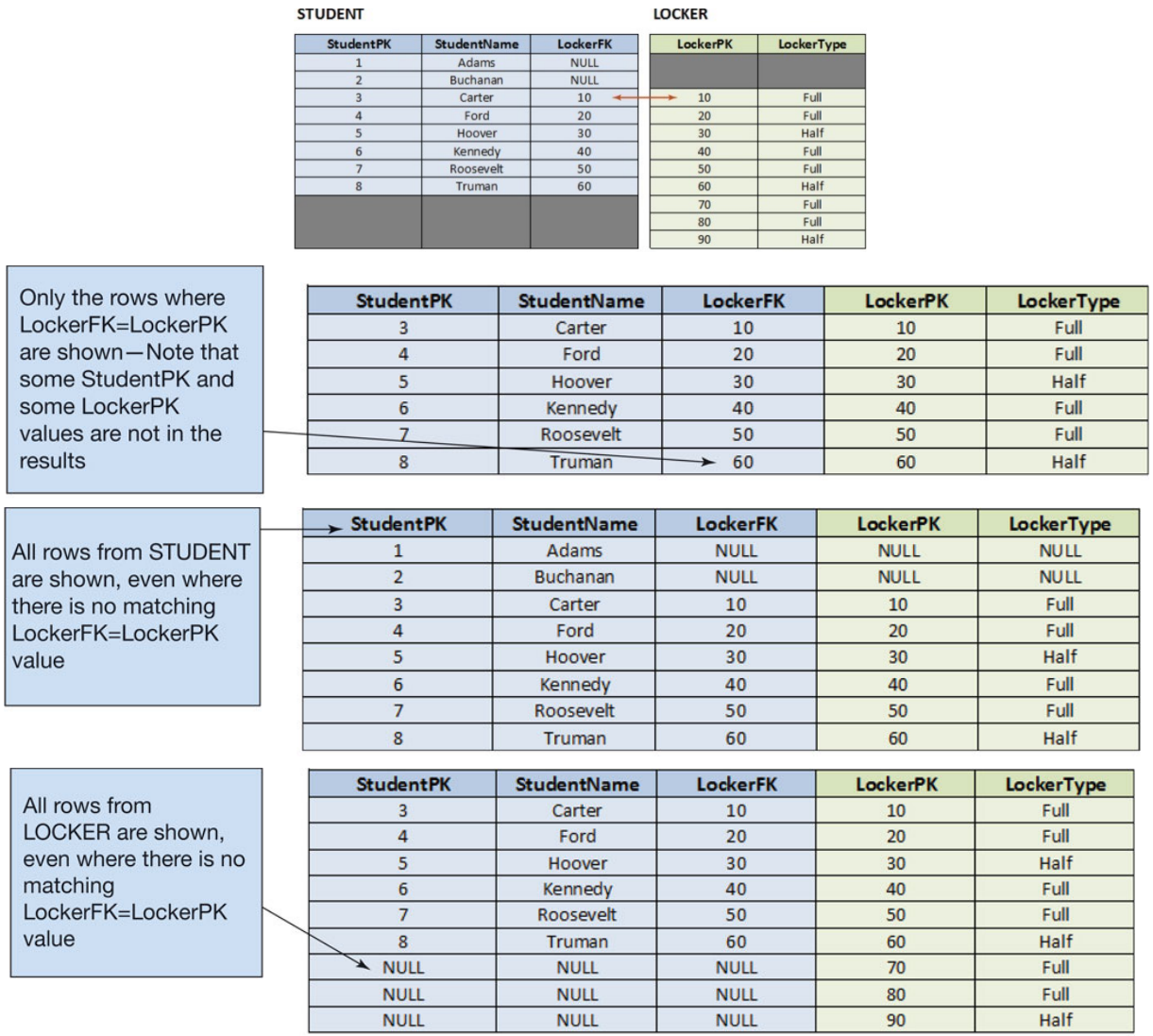
- Subqueries and joins both process multiple tables, but they differ slightly.
  - A subquery can only be used to retrieve data from the top table
  - A join can be used to obtain data from any number of tables.

Multiple Table Queries: SQL Inner Joins

Learn basic SQL SELECT Statements for processing multiple tables with Joins

- SQL Inner Join is also referred to as an SQL equijoin.
- An Inner Join only displays data from the rows that match based on join conditions:
  - if a row has a value that does not match the WHERE clause condition, that row will not be included in the join result

Figure 3.25 Types of SQL JOINS



SQL for Data Manipulation (DML)– Data Modification and Deletion

Learn basic SQL for modifying and deleting data from a database

- The SQL DML contains commands for the three possible data modifications operations:
  - Insert
  - Modify
  - Delete

Modifying Data Example 1

Learn basic SQL for modifying and deleting data from a database

```
/* *** SQL-UPDATE-CH03-01 *** */
UPDATE          EMPLOYEE
SET             OfficePhone = '360-285-8620'
WHERE           EmployeeNumber = 13;
```

To see the results, we use the following command:

```
/* *** SQL-Query-CH03-58 *** */
SELECT          *
FROM            EMPLOYEE
WHERE           EmployeeNumber = 13;
```

EmployeeNumber	FirstName	LastName	Department	Position	Supervisor	OfficePhone	EmailAddress
13	Rick	Brown	InfoSystems	IS2	12	360-285-8620	Rick.Brown@WP.com

MySQL Community Server 8.0, MySQL Workbench, Oracle Corporation.

Modifying Data Example 2

Learn basic SQL for modifying and deleting data from a database

```
/* *** EXAMPLE CODE - DO NOT RUN *** */
/* *** SQL-UPDATE-CH03-03 *** */
UPDATE          EMPLOYEE
SET             Department = 'Finance', OfficePhone = '360-285-8420'
WHERE           EmployeeNumber = 9;
```

Deleting Data

Learn basic SQL for modifying and deleting data from a database

```
/* *** EXAMPLE CODE - DO NOT RUN *** */
/* *** SQL-DELETE-CH03-01 *** */
DELETE
FROM            PROJECT
WHERE           Department = 'Sales and Marketing';
```

The example above is a valid statement, but note that if you fail to include the WHERE clause then you will have just deleted all records in the table.

SQL For Data Definition (DDL) – Table and Constraint Modification and Deletion

Learn basic SQL statements for modifying and deleting database tables and constraints

- Two of the most useful data definition SQL statements are:
  - the SQL DROP TABLE statement
  - the SQL ALTER TABLE statement

The SQL DROP TABLE Statement

Learn basic SQL statements for modifying and deleting database tables and constraints

```
/* *** SQL-DROP-TABLE-CH03-01 *** */
DROP TABLE ASSIGNMENT;
```

The SQL ALTER TABLE Statement

Learn basic SQL statements for modifying and deleting database tables and constraints

```
/* *** SQL-ALTER-TABLE-CH03-01  *** */
ALTER TABLE ASSIGNMENT
        DROP CONSTRAINT ASSIGN_EMP_FK;
```

Copyright