Listing Table Data – The SELECT statement

- SELECT command is used to list table contents
- Syntax:

SELECT *columnlist* FROM *tablename*;

- Example:
  - Listing all table data

    SELECT * FROM VENDOR;
  - Listing selected columns

    SELECT P_DESCRIPT, P_PRICE   FROM PRODUCT;

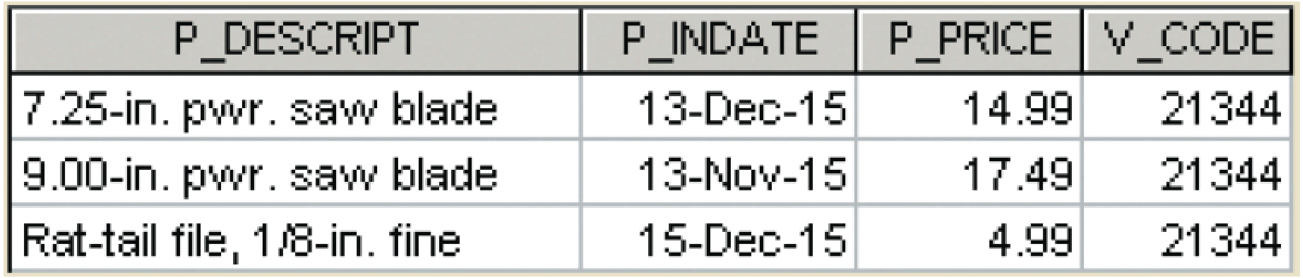
SELECT with Conditional Restrictions

- Syntax:

```
SELECT        columnlist
    FROM      tablelist
    [WHERE    conditionlist];
```

- Example: Display all products supplied by V_CODE 21344

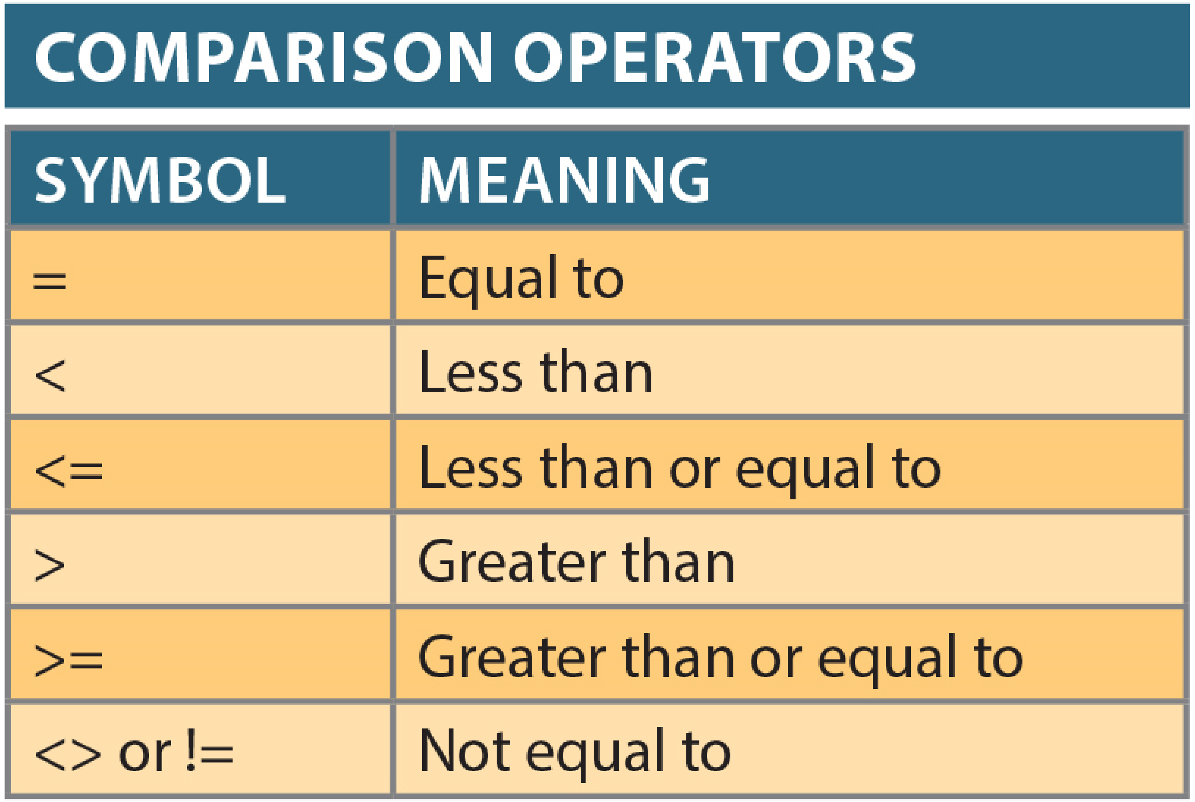
```
SELECT        P_DESCRIPT, P_INDATE, P_PRICE, V_CODE
    FROM      PRODUCT
    WHERE     V_CODE = 21344
```



P_DESCRIPT	P_INDATE	P_PRICE	V_CODE
7.25-in. pwr. saw blade	13-Dec-15	14.99	21344
9.00-in. pwr. saw blade	13-Nov-15	17.49	21344
Rat-tail file, 1/8-in. fine	15-Dec-15	4.99	21344

Comparison Operators

- Adds conditional restrictions on selected character attributes and dates



COMPARISON OPERATORS

SYMBOL	MEANING
=	Equal to
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
<> or !=	Not equal to

SELECT with Conditional Restrictions

- Example 2: Display all products not supplied by V_CODE = 21344

```
SELECT          *
      FROM      PRODUCT
      WHERE     V_CODE <> 21344;
```

- Example 3: Display description and price of products with price less than 10

```
SELECT          P_DESCRIPT, P_PRICE
      FROM      PRODUCT
      WHERE     P_PRICE < 10;
```

Updating Table Data

- UPDATE TABLE command is used to update existing data in a row or table
- Syntax:
  UPDATE *table_name*

  SET *assignment_list*

  [WHERE *where_condition*]
- WHERE clause is used to update data in rows that satisfy the stated conditions

Updating Table Data

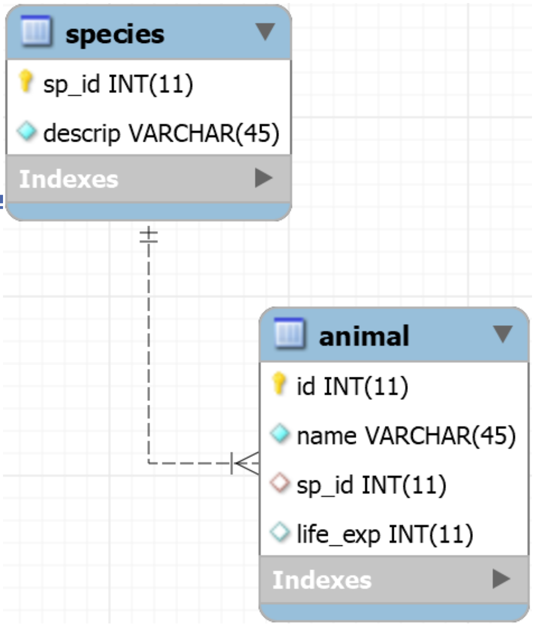
- Example:
  - Updating all rows

```
UPDATE    PRODUCT
   SET       P_PRICE = 0;
```

  - Updating a specific row

```
UPDATE      PRODUCT
   SET          P_PRICE = 1000
   WHERE    V_CODE = 21344
```

Exercises



- Change the maximum length of life expectancy column to INT(3)

ALTER TABLE ANIMAL MODIFY LIFE_EXP INT(3)

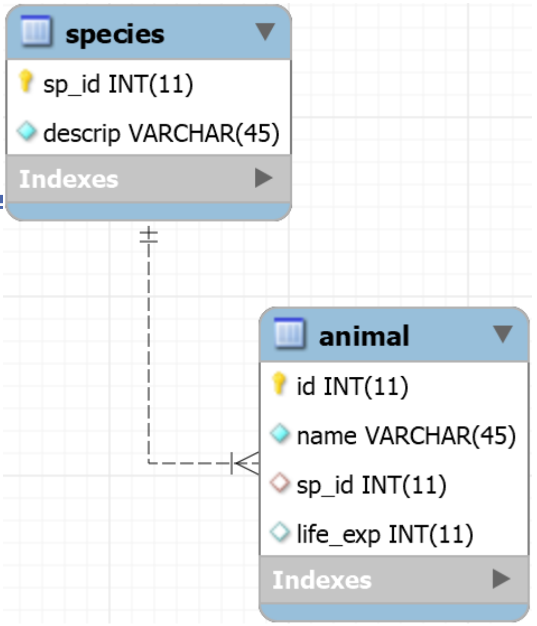
ANIMAL

ID	NAME	SP_ID	LIFE_EXP
1	Cat	4	20
2	Elephant	4	70
3	Trout	3	5
4	Shark	3	25
5	Canary	2	20
6	Albatross	2	40
7	Swift	2	5

SPECIES

SP_ID	DESCRIP
1	INSECT
2	BIRD
3	FISH
4	MAMMAL

Exercises



- Add a new column of a suitable data type named IS_EXTINCT

ALTER TABLE ANIMAL ADD IS_EXTINCT BOOL

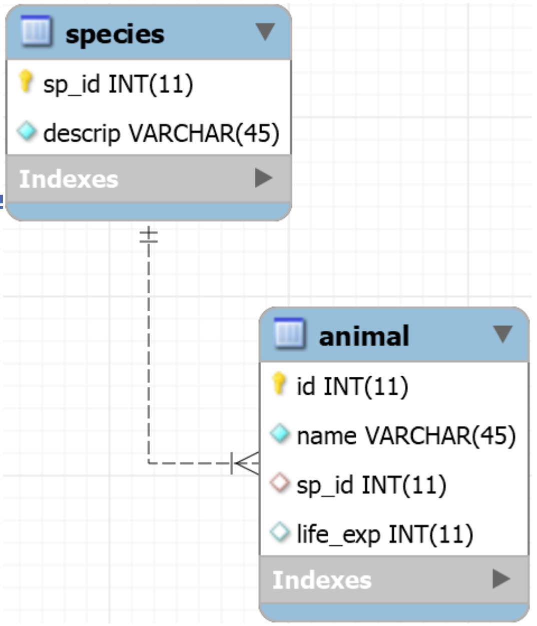
SPECIES

SP_ID	DESCRIP
1	INSECT
2	BIRD
3	FISH
4	MAMMAL

ANIMAL

ID	NAME	SP_ID	LIFE_EXP
1	Cat	4	20
2	Elephant	4	70
3	Trout	3	5
4	Shark	3	25
5	Canary	2	20
6	Albatross	2	40
7	Swift	2	5

Exercises



- Set the value of the new column to TRUE for all animals

```
UPDATE ANIMAL SET IS_EXTINCT = TRUE;
```

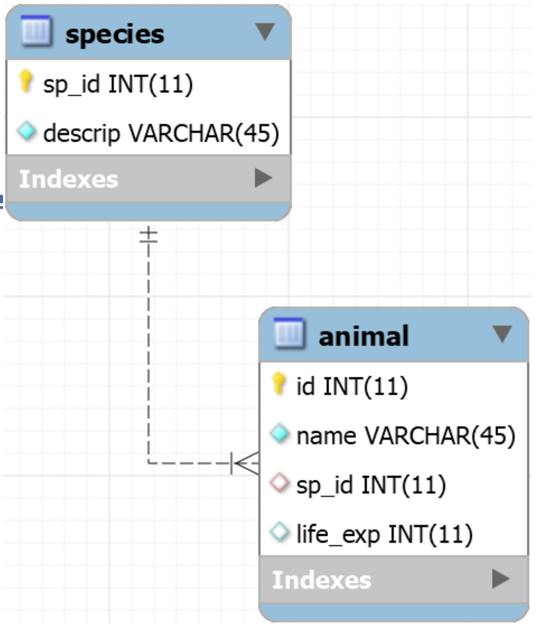
SPECIES

SP_ID	DESCRIP
1	INSECT
2	BIRD
3	FISH
4	MAMMAL

ANIMAL

ID	NAME	SP_ID	LIFE_EXP
1	Cat	4	20
2	Elephant	4	70
3	Trout	3	5
4	Shark	3	25
5	Canary	2	20
6	Albatross	2	40
7	Swift	2	5

Exercises


species
sp_id INT(11)
descrip VARCHAR(45)
Indexes
animal
id INT(11)
name VARCHAR(45)
sp_id INT(11)
life_exp INT(11)
Indexes

- Remove the IS_EXTINCT column from the animal table

```
ALTER TABLE ANIMAL DROP IS_EXTINCT
```

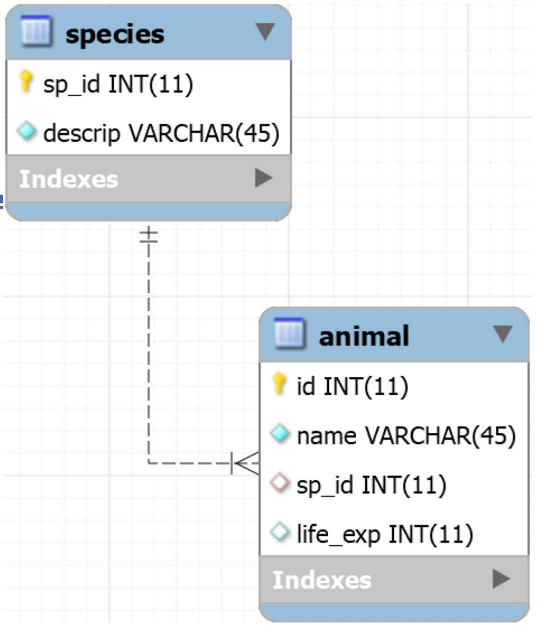
SPECIES

SP_ID	DESCRIP
1	INSECT
2	BIRD
3	FISH
4	MAMMAL

ANIMAL

ID	NAME	SP_ID	LIFE_EXP
1	Cat	4	20
2	Elephant	4	70
3	Trout	3	5
4	Shark	3	25
5	Canary	2	20
6	Albatross	2	40
7	Swift	2	5

Exercises



- Insert a new species with an id of 5 and having description REPTILE

```
INSERT INTO SPECIES VALUES (5, "REPTILE");
```

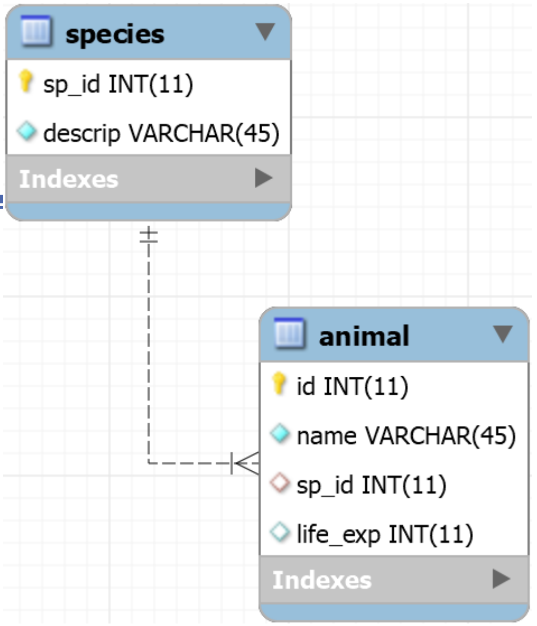
SPECIES

SP_ID	DESCRIP
1	INSECT
2	BIRD
3	FISH
4	MAMMAL

ANIMAL

ID	NAME	SP_ID	LIFE_EXP
1	Cat	4	20
2	Elephant	4	70
3	Trout	3	5
4	Shark	3	25
5	Canary	2	20
6	Albatross	2	40
7	Swift	2	5

Exercises



- Delete all animals with life expectancy less than 10 years

DELETE FROM ANIMAL WHERE LIFE_EXP < 10

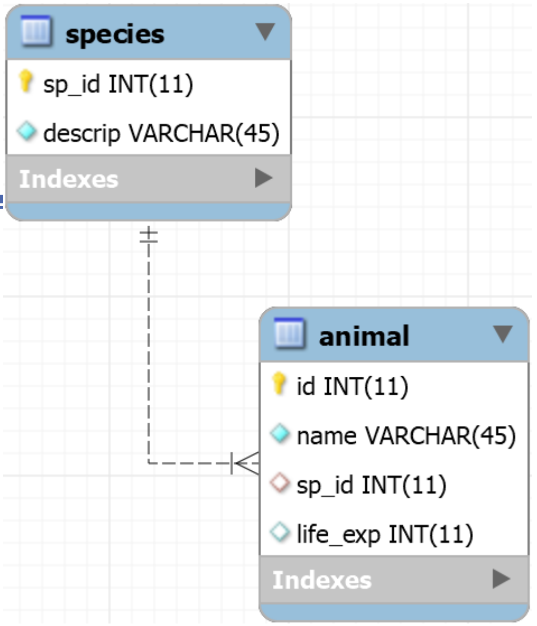
ANIMAL

ID	NAME	SP_ID	LIFE_EXP
1	Cat	4	20
2	Elephant	4	70
3	Trout	3	5
4	Shark	3	25
5	Canary	2	20
6	Albatross	2	40
7	Swift	2	5

SPECIES

SP_ID	DESCRIP
1	INSECT
2	BIRD
3	FISH
4	MAMMAL

Exercises



- Delete species with id of 2 from the species table

DELETE FROM SPECIES WHERE SP_ID = 2

Error: Foreign key constraint fails

This will work

DELETE FROM ANIMAL WHERE SP_ID = 2

DELETE FROM SPECIES WHERE SP_ID = 2

SPECIES

SP_ID	DESCRIP
1	INSECT
2	BIRD
3	FISH
4	MAMMAL

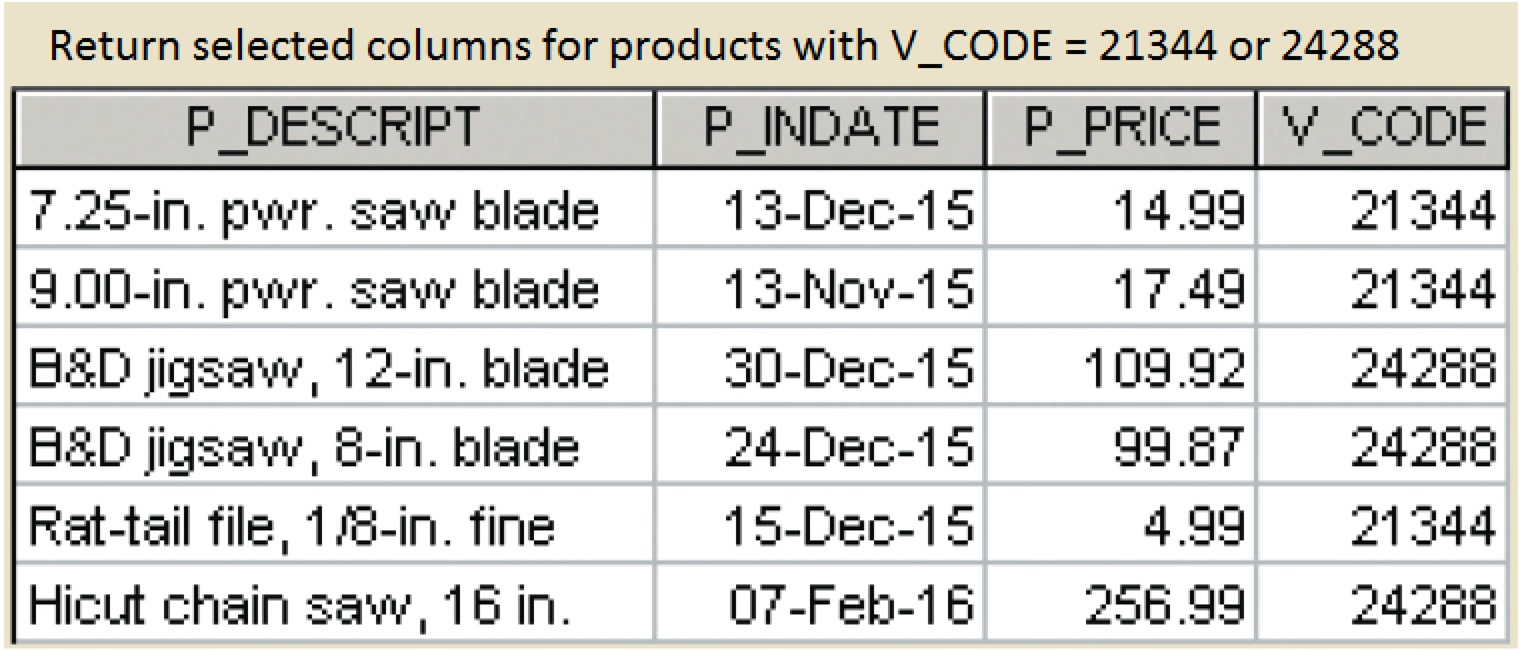
ANIMAL

ID	NAME	SP_ID	LIFE_EXP
1	Cat	4	20
2	Elephant	4	70
3	Trout	3	5
4	Shark	3	25
5	Canary	2	20
6	Albatross	2	40
7	Swift	2	5

Logical Operators: AND, OR and NOT

- **OR** and **AND**: Used to link multiple conditional expressions in a WHERE or HAVING clause
  - **OR** requires only one of the conditional expressions to be true
  - **AND** requires all of the conditional expressions to be true
- **NOT** is used to negate the result of a conditional expression

Selected PRODUCT Table Attributes: The Logical OR

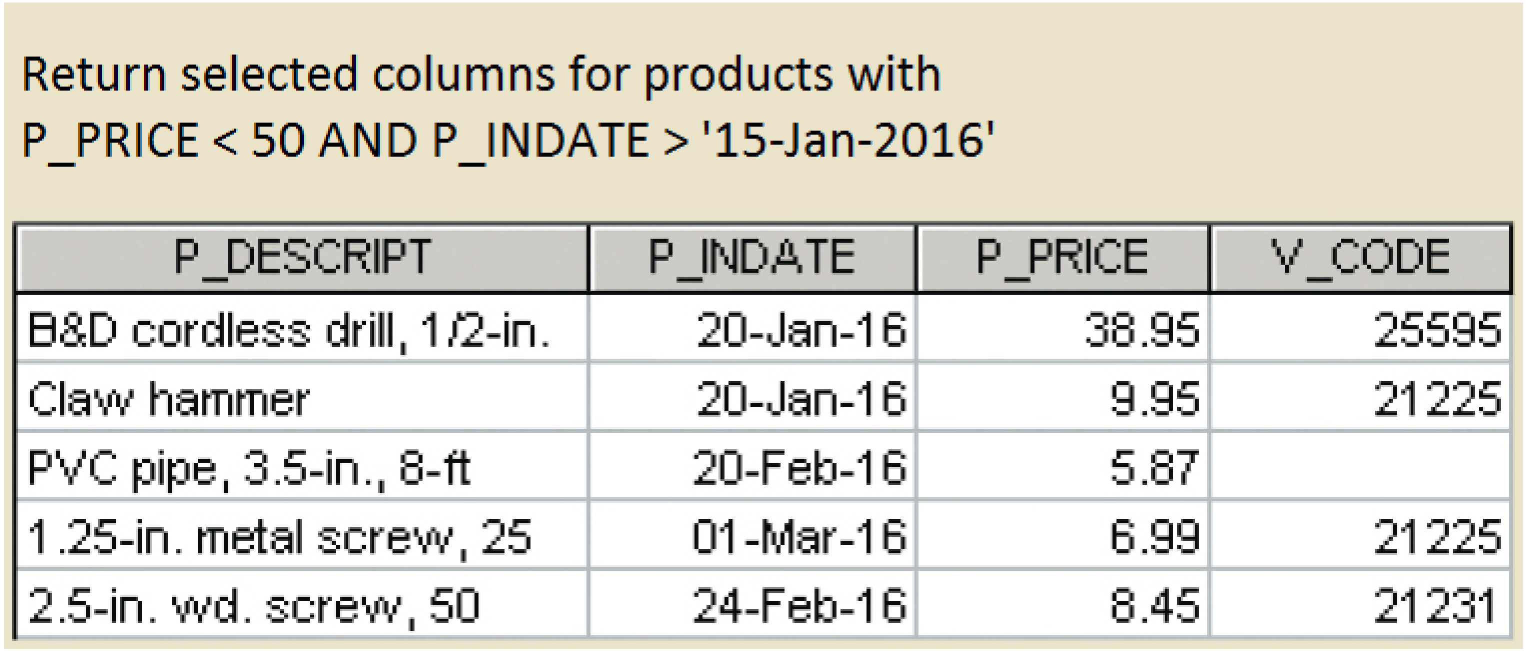


Return selected columns for products with V_CODE = 21344 or 24288

P_DESCRIPT	P_INDATE	P_PRICE	V_CODE
7.25-in. pwr. saw blade	13-Dec-15	14.99	21344
9.00-in. pwr. saw blade	13-Nov-15	17.49	21344
B&D jigsaw, 12-in. blade	30-Dec-15	109.92	24288
B&D jigsaw, 8-in. blade	24-Dec-15	99.87	24288
Rat-tail file, 1/8-in. fine	15-Dec-15	4.99	21344
Hicut chain saw, 16 in.	07-Feb-16	256.99	24288

```
SELECT      P_DESCRIPT, P_INDATE, P_PRICE, V_CODE
   FROM     PRODUCT
   WHERE    V_CODE = 21344 OR V_CODE = 24288;
```

Selected PRODUCT Table Attributes: The Logical AND

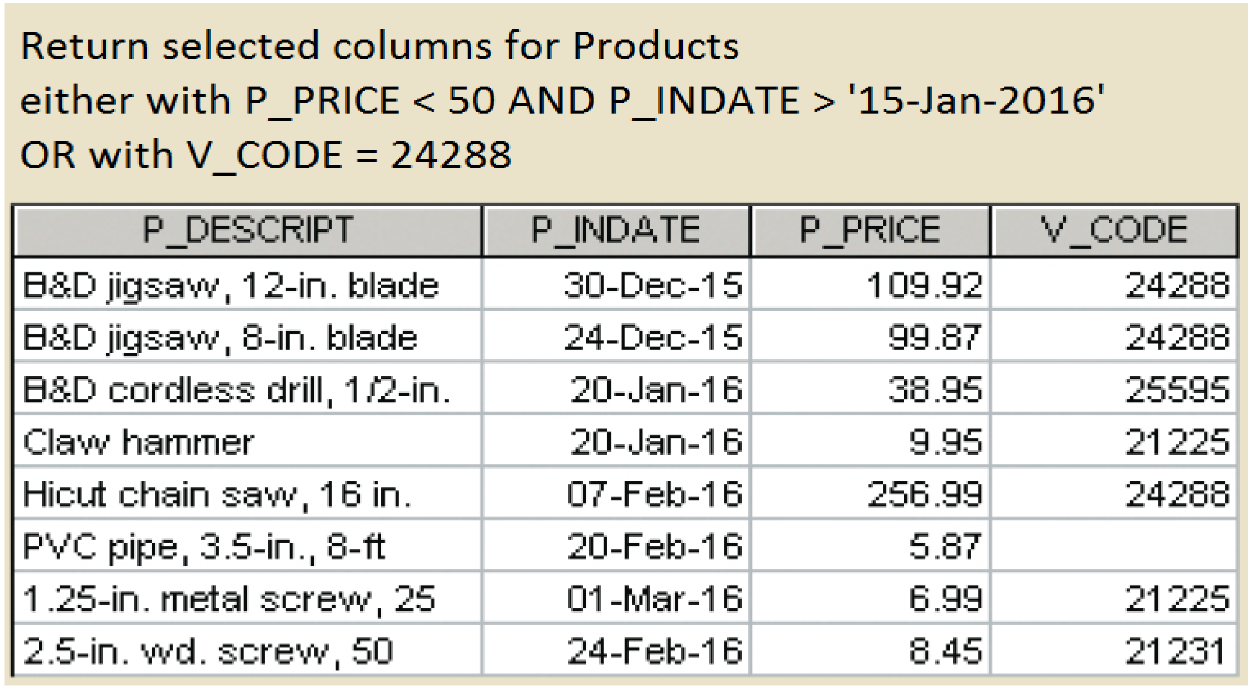


Return selected columns for products with
P_PRICE < 50 AND P_INDATE > '15-Jan-2016'

P_DESCRIPT	P_INDATE	P_PRICE	V_CODE
B&D cordless drill, 1/2-in.	20-Jan-16	38.95	25595
Claw hammer	20-Jan-16	9.95	21225
PVC pipe, 3.5-in., 8-ft	20-Feb-16	5.87	
1.25-in. metal screw, 25	01-Mar-16	6.99	21225
2.5-in. wd. screw, 50	24-Feb-16	8.45	21231

```
SELECT      P_DESCRIPT, P_INDATE, P_PRICE, V_CODE
    FROM    PRODUCT
    WHERE   P_PRICE < 50 AND P_INDATE > ’ 2016-01-15’;
```

Selected PRODUCT Table Attributes: The Logical AND and OR



Return selected columns for Products
either with P_PRICE < 50 AND P_INDATE > '15-Jan-2016'
OR with V_CODE = 24288

P_DESCRIPT	P_INDATE	P_PRICE	V_CODE
B&D jigsaw, 12-in. blade	30-Dec-15	109.92	24288
B&D jigsaw, 8-in. blade	24-Dec-15	99.87	24288
B&D cordless drill, 1/2-in.	20-Jan-16	38.95	25595
Claw hammer	20-Jan-16	9.95	21225
Hicut chain saw, 16 in.	07-Feb-16	256.99	24288
PVC pipe, 3.5-in., 8-ft	20-Feb-16	5.87	
1.25-in. metal screw, 25	01-Mar-16	6.99	21225
2.5-in. wd. screw, 50	24-Feb-16	8.45	21231

```
SELECT      P_DESCRIPT, P_INDATE, P_PRICE, V_CODE
    FROM    PRODUCT
    WHERE   (P_PRICE < 50 AND P_INDATE > '2016-01-15')
            OR V_CODE = 24288;
```

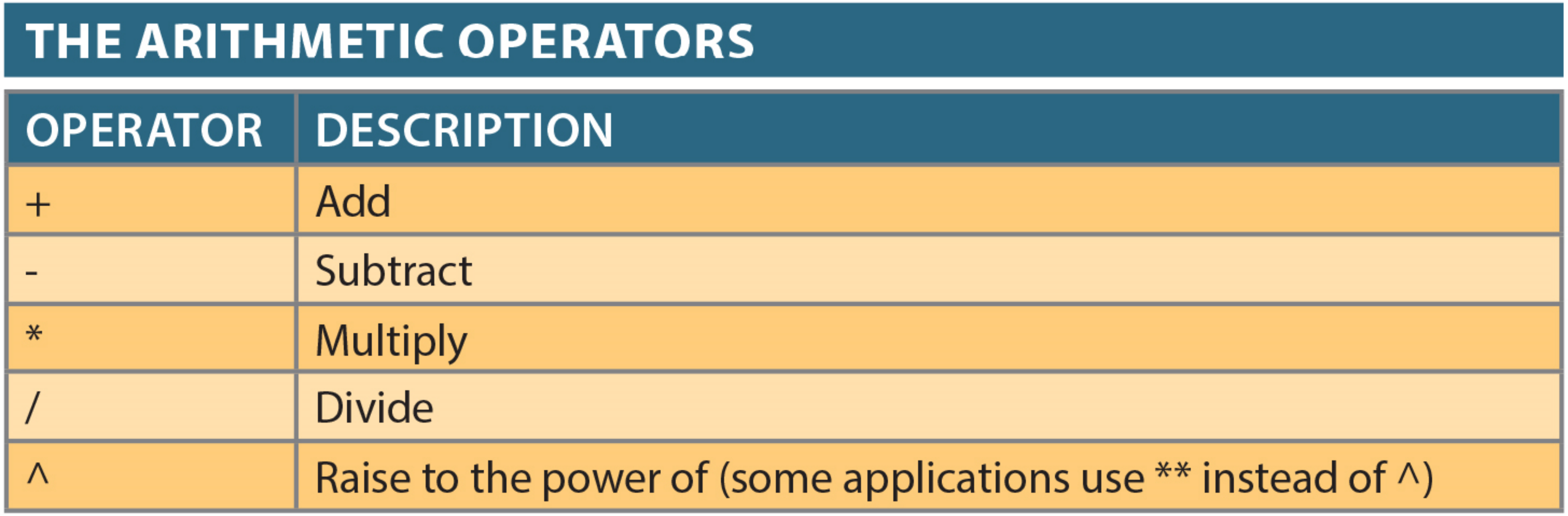
Comparison Operators: Computed Columns and Column Aliases

- SQL accepts any valid expressions/formulas in the computed columns
- **Alias**: Alternate name given to a column or table in any SQL statement to improve the readability
- Computed column, an alias, and date arithmetic can be used in a single query

Arithmetic Operators

- **The Rule of Precedence**: Establish the order in which computations are completed
- Performed in this order:
  - Operations within parentheses
  - Power operations
  - Multiplications and divisions
  - Additions and subtractions
- Remember BODMAS?
  - Bracket, Order, Division, Multiplication, Addition, Subtraction

The Arithmetic Operators



THE ARITHMETIC OPERATORS

OPERATOR	DESCRIPTION
+	Add
-	Subtract
*	Multiply
/	Divide
^	Raise to the power of (some applications use ** instead of ^)

Note: MySQL uses the function Power(a,b) or Pow(a,b) to return a^b.
– e.g. pow(3,2) returns 9

Special Operators

BETWEEN

- Checks whether attribute value is within a range

IS NULL

- Checks whether attribute value is null

LIKE

- Checks whether attribute value matches given string pattern

IN

- Checks whether attribute value matches any value within a value list

EXISTS

- Checks if subquery returns any rows

Examples – BETWEEN and IS NULL

BETWEEN

- SELECT * FROM PRODUCT
  WHERE P_PRICE BETWEEN 50 AND 100;

  is the same as:

- SELECT * FROM PRODUCT
  WHERE P_PRICE >= 50 AND P_PRICE <= 100;

- Note: BETWEEN 100 AND 50 would be the same as
  WHERE P_PRICE <= 50 AND P_PRICE >= 100; this will be incorrect.

IS NULL

- SELECT * FROM PRODUCT WHERE V_CODE IS NULL;

Example - LIKE

- SELECT * FROM VENDOR
        WHERE V_CONTACT LIKE ‘SMITH%’;
  (returns Smith and Smithson from the VENDOR data file shown earlier)

- SELECT * FROM VENDOR
        WHERE V_CONTACT LIKE ‘S%’;
  (returns Singh, Smith and Smithson from the data file shown earlier)

- SELECT * FROM VENDOR
        WHERE V_CONTACT NOT LIKE ‘S%’;
  (returns everyone except Singh, Smith and Smithson from the data file shown earlier)

Wildcards

- % sign = all *following* or *preceding* characters
  - Examples:
    J% => Jim, Jane, Jules, Jones, Johnson, July, …
    Jo% => Jones, Johnson
    %ul% => Jules, July
- _ character = any *one* character
  - Example
    _23-456-6789 => 123-456-6789, 223-456-6789, …
    123-456-6__9 => 123-456-6119, 123-456-6129, …

Example – IN

- SELECT * FROM PRODUCT
  WHERE V_CODE IN (21344, 24288);

- SELECT * FROM VENDOR
  WHERE V_CODE
  IN (SELECT V_CODE FROM PRODUCT)
  - Inner query executed first, returning a list of V_CODEs from the PRODUCT table
  - The IN operator compares the values generated from the sub-query to the V_CODE in the VENDOR table, and select only rows with matching values

Example – EXISTS

- Used to execute a command based on the results of another query

-
  ```
  SELECT * FROM VENDOR
          WHERE EXISTS (SELECT * FROM PRODUCT
                                  WHERE P_QOH <= P_MIN);
  ```
  The main query will execute only if the sub-query returns any rows. It will not be executed otherwise

-
  ```
  SELECT V_CONTACT FROM VENDOR
          WHERE EXISTS (SELECT * FROM PRODUCT
                                  WHERE P_QOH < P_MIN*2);
  ```

Ordering a Listing

- **ORDER BY** clause is useful when listing order is important
- Syntax - SELECT *columnlist*

  FROM *tablelist*

  [WHERE *conditionlist*]

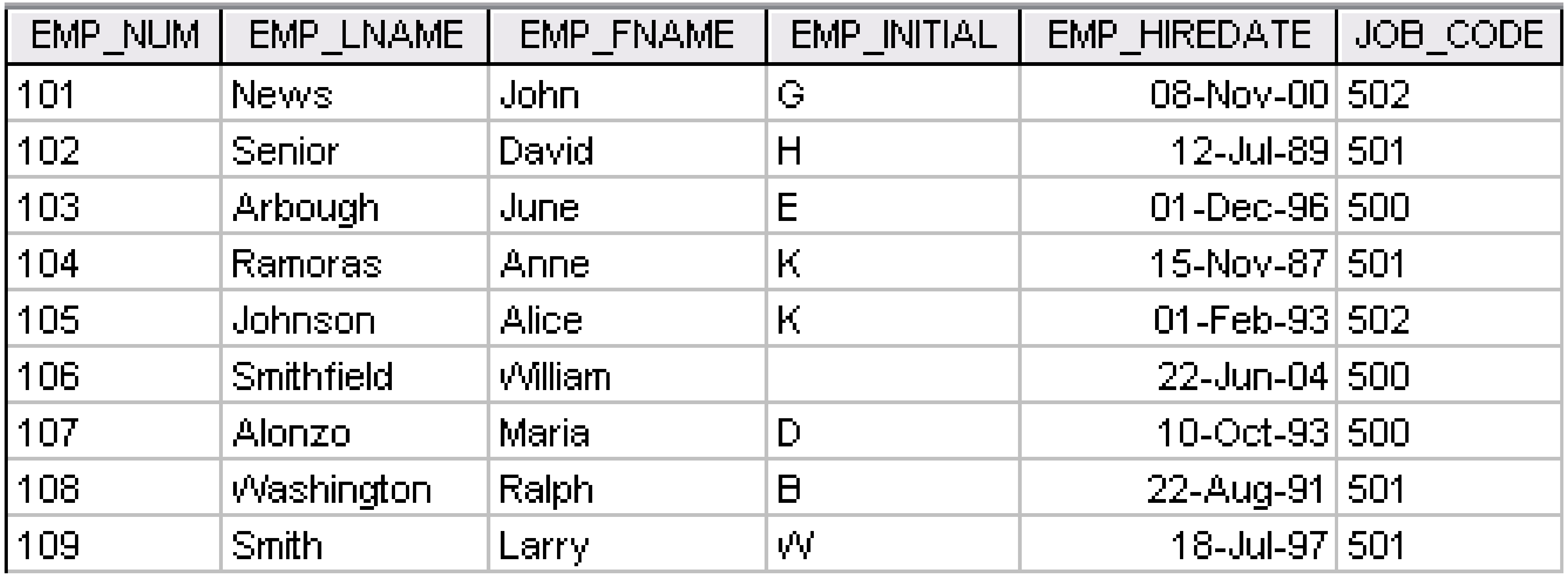
  [ORDER BY *columnlist* [ASC | DESC]];
- **Cascading order sequence**: Multilevel ordered sequence
  - Created by listing several attributes after the ORDER BY clause

Ordering a Listing

- SELECT * FROM PRODUCT
  ORDER BY P_PRICE
  (*lists all products in ascending order of their price*)
- SELECT P_DESCRIPT, P_PRICE FROM PRODUCT
  WHERE P_DESCRIPT LIKE '%hammer%'
  ORDER BY P_PRICE DESC
- SELECT * FROM PRODUCT
  ORDER BY P_DESCRIPT, P_PRICE DESC
  (*lists all products ordered by description in ascending order and then by their price in descending order*)

Exercise – EMPLOYEE table

- list last name of all employees with **ia** in their first names



EMP_NUM	EMP_LNAME	EMP_FNAME	EMP_INITIAL	EMP_HIREDATE	JOB_CODE
101	News	John	G	08-Nov-00	502
102	Senior	David	H	12-Jul-89	501
103	Arbough	June	E	01-Dec-96	500
104	Ramoras	Anne	K	15-Nov-87	501
105	Johnson	Alice	K	01-Feb-93	502
106	Smithfield	William		22-Jun-04	500
107	Alonzo	Maria	D	10-Oct-93	500
108	Washington	Ralph	B	22-Aug-91	501
109	Smith	Larry	W	18-Jul-97	501

Exercise – EMPLOYEE table

- list last name of all employees with **ia** in their first names

SELECT EMP_LNAME FROM EMPLOYEE
        WHERE EMP_FNAME LIKE '%ia%';

EMP_NUM	EMP_LNAME	EMP_FNAME	EMP_INITIAL	EMP_HIREDATE	JOB_CODE
101	News	John	G	08-Nov-00	502
102	Senior	David	H	12-Jul-89	501
103	Arbough	June	E	01-Dec-96	500
104	Ramoras	Anne	K	15-Nov-87	501
105	Johnson	Alice	K	01-Feb-93	502
106	Smithfield	William		22-Jun-04	500
107	Alonzo	Maria	D	10-Oct-93	500
108	Washington	Ralph	B	22-Aug-91	501
109	Smith	Larry	W	18-Jul-97	501