SELECT

- Examples – List unique values, COUNT, MAX, MIN, SUM, AVG

```
SELECT DISTINCT  V_CODE  FROM PRODUCT;
```

```
SELECT COUNT( DISTINCT V_CODE )
FROM PRODUCT;
```

```
SELECT COUNT( * )  FROM PRODUCT
WHERE P_PRICE >= 3;
```

```
SELECT MAX( P_PRICE )
FROM PRODUCT;
```

SELECT

- Examples – List unique values, COUNT, MAX, MIN, SUM, AVG

```
SELECT P_CODE, P_DESCRIPT, P_PRICE
FROM PRODUCT
WHERE P_PRICE = MAX( P_PRICE ) ;
```

Incorrect!
MAX(columnname) can be used only in the column list of a SELECT statement

```
SELECT P_CODE, P_DESCRIPT, P_PRICE
FROM PRODUCT
WHERE P_PRICE = ( SELECT MAX( P_PRICE ) FROM PRODUCT );
```

SELECT

- Examples – List unique values, COUNT, MAX, MIN, SUM, AVG

```
SELECT SUM(P_QOH) AS TOTQOH
FROM PRODUCT;
```

```
SELECT SUM( P_QOH * P_PRICE ) AS TOTVALUE
FROM PRODUCT;
```

```
SELECT AVG( P_PRICE )
FROM PRODUCT;
```

```
SELECT P_CODE, P_DESCRIPT, P_QOH, P_PRICE, V_CODE
FROM PRODUCT
WHERE P_PRICE > ( SELECT AVG( P_PRICE ) FROM PRODUCT )
ORDER BY P_PRICE DESC;
```

Exercise – A Consulting Company (Cont.)

- Write the SQL code that will list only the distinct EMP_NUM in the table "ASSIGNMENT".
- Write the SQL code to find the average PROJ_VALUE in the table "PROJECT".
- Write the SQL code to count the number of distinct EMP_NUM in the table "ASSIGNMENT".
- Write the SQL code to list all attributes of the project that has the largest amount of PROJ_VALUE.
- Write the SQL code to list all attributes of the project(s) which PROJ_VALUE is higher than the average PROJ_VALUE. Sort the result by PROJ_BALANCE in ascending order.

SELECT

- Grouping data
- Syntax

```
SELECT columnlist
FROM tablelist
[WHERE conditionlist]
[GROUP BY columnlist]
[HAVING conditionlist]
[ORDER BY columnlist[ASC| DESC] ];
```

- WHERE clause is applied to columns
- HAVING clause is applied to output of a GROUP BY operation

SELECT

- Examples – Grouping data

```
SELECT V_CODE, COUNT( DISTINCT P_CODE ) , AVG( P_PRICE)
FROM PRODUCT
GROUP BY V_CODE
HAVING AVG( P_PRICE ) < 200;
```

```
SELECT V_CODE, SUM( P_QOH * P_PRICE ) AS TOTCOST
FROM PRODUCT
GROUP BY V_CODE
HAVING (TOTCOST>100)
ORDER BY TOTCOST DESC;
```

Exercise – A Consulting Company (Cont.)

- Write the SQL code to find the numbers of employees that each project has been assigned to.
- Write the SQL code to list the EMP_NUM and the number of projects s/he has been assigned to for employees who has been assigned to at least 2 projects.

Insert Table Rows with a Select Subquery

- Insert multiple rows from another table
- The values returned by the SELECT subquery should match the attributes and data types of the table in the INSERT statement
- Syntax

```
INSERT INTO  tablename SELECT columnlist FROM  tablename;
```

Insert Table Rows with a Select Subquery

- Example

```
CREATE TABLE VENDOR2(
  V_CODE INT NOT NULL UNIQUE ,
  V_NAME VARCHAR( 35 ) NOT NULL ,
  V_CONTACT VARCHAR( 25 ) NOT NULL,
  PRIMARY KEY ( V_CODE )
) ;

INSERT INTO VENDOR2
SELECT V_CODE, V_NAME, V_CONTACT
FROM VENDOR;
```

OR

```
CREATE TABLE VENDOR2 AS
SELECT V_CODE AS CODE, V_NAME AS NAME,
V_CONTACT AS CONTACT FROM VENDOR;
```

Update

- Modify data in a table
- Syntax

```
UPDATE tablename SET columnname = expression [, columnname = expression]
[WHERE conditionlist];
```

- See http://dev.mysql.com/doc/refman/5.7/en/update.html for details.

Update

- Examples

```
UPDATE PRODUCT SET P_INDATE = '2012-10-15'  WHERE P_CODE = '11AER/31' ;
```

```
UPDATE PRODUCT
SET P_INDATE = '2013-01-10', P_PRICE = 17.99, P_MIN =10
WHERE P_CODE = 'BRT-345';
```

```
UPDATE PRODUCT SET P_PRICE = 28.86 WHERE V_CODE IN (21225, 21266);
```

Update

- Examples

```
UPDATE PRODUCT SET P_PRICE = 150 WHERE P_INDATE < '2013-01-01';
```

```
UPDATE PRODUCT SET P_PRICE = 26  WHERE P_CODE = '11AER/31'  OR
P_CODE = '2232/QTY';
```

Delete

- Delete a table row
- Syntax:

```
DELETE FROM tablename
[WHERE conditionlist];
```

-- WHERE condition is optional. If WHERE condition is not specified, all rows from specified table will be deleted!

- See http://dev.mysql.com/doc/refman/5.7/en/delete.html for details.

Delete

- Examples

```
DELETE FROM PRODUCT WHERE P_CODE = 'BRT-345';
```

```
DELETE FROM PRODUCT;
```

Exercise – A Consulting Company (Cont.)

- Write the SQL code to change the job code to 501 for the person whose employee number (EMP_NUM) is 107.
- Write the SQL code to delete the row for William Smithfield, who was hired on June 22, 2004, and whose job code is 500.
- Write the SQL code to create a copy of EMPLOYEE, naming the copy EMP_1. Then write the SQL code that will add the attributes EMP_PCT and PROJ_NUM to its structure. The EMP_PCT is the bonus percentage to be paid to each employee. The new attribute characteristics are:

  EMP_PCT DECIMAL(4, 2)

  PROJ_NUM CHAR(3)
- Write the SQL code that will change the EMP_YEAR to 14 for employees who were hired before January 1, 1994, and whose job code is at least 501.

JOIN tables

- JOIN is performed when data are retrieved from more than one table
- DBMS creates Cartesian Product of every table when JOIN is performed
- Must select only the rows in which the common attribute values match by using the WHERE clause for a natural join (usually used in equality comparison between foreign key and primary key of related tables)

JOIN tables

- Examples

```
SELECT * FROM PRODUCT, VENDOR;
```

```
SELECT  *
FROM PRODUCT
CROSS JOIN VENDOR;
```

```
SELECT  P_CODE, V_NAME, P_PRICE
FROM PRODUCT
JOIN VENDOR;
```

Cartesian product

JOIN tables

- Examples

```
SELECT *
FROM PRODUCT
NATURAL JOIN
 VENDOR;
```

```
SELECT P_DESCRIPT, P_PRICE, V_NAME, V_CONTACT, V_AREACODE, V_PHONE
FROM PRODUCT, VENDOR
WHERE PRODUCT.V_CODE = VENDOR.V_CODE;
```

```
SELECT P_DESCRIPT, P_PRICE, V_NAME, V_CONTACT, V_AREACODE, V_PHONE
FROM PRODUCT, VENDOR
WHERE PRODUCT.V_CODE = VENDOR.V_CODE
ORDER BY P_PRICE;
```

Natural Join

JOIN tables

- Examples

```
SELECT P_DESCRIPT, P_PRICE, V_NAME, V_CONTACT, V_AREACODE, V_PHONE
FROM PRODUCT, VENDOR
WHERE PRODUCT.V_CODE = VENDOR.V_CODE
AND P_INDATE > '2010-01-01';
```

```
SELECT P_DESCRIPT, P_PRICE, V_NAME, V_CONTACT, V_AREACODE, V_PHONE
FROM PRODUCT P, VENDOR V
WHERE P.V_CODE = V.V_CODE
ORDER BY P_PRICE;
```

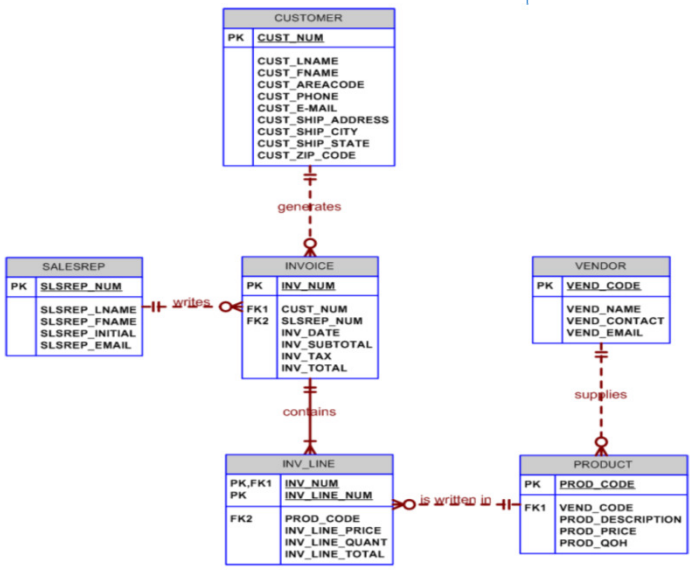
```
SELECT E.manager, M.lastname, E.num, E.lastname
FROM employee E, employee M
WHERE E.manager = M.num
ORDER BY E.manager;
```

num	lastname	manager
101	Waddell	102
102	Orincona	
103	Jones	102
104	Reballoh	102
105	Robertson	102
106	Deltona	102

JOIN tables

- Examples

```
SELECT CUS_LNAME, INVOICE.INV_NUM, INV_DATE, PROD_DESCRIPTION
FROM CUSTOMER, INVOICE, INV_LINE, PRODUCT
WHERE CUSTOMER.CUST_NUM = INVOICE.CUST_NUM
AND INVOICE.INV_NUM = INV_LINE.INV_NUM
AND INV_LINE.PROD_CODE = PRODUCT.PROD_CODE
AND CUSTOMER.CUST_NUM =10014;
```



JOIN tables

Returns only rows with matching values in the column indicated in the USING clause, and that column must exist in both tables.
Syntax:
SELECT column-list **FROM** table 1 **JOIN** table2 **USING (**common-column**)**

- Examples

equivalent

```
SELECT INV_NUM, PROD_CODE, PROD_DESCRIPTION
FROM INVOICE JOIN INV_LINE
USING (INV_NUM)
JOIN PRODUCT USING (PROD_CODE) ;
```

```
SELECT INV_NUM, PROD_CODE, PROD_DESCRIPT
FROM INVOICE NATURAL JOIN INV_LINE
NATURAL JOIN PRODUCT;
```

```
SELECT INVOICE.INV_NUM, INV_LINE.PROD_CODE, PROD_DESCRIPTION
FROM INVOICE, INV_LINE, PRODUCT
WHERE INVOICE.INV_NUM = INV_LINE.INV_NUM
AND INV_LINE.PROD_CODE = PRODUCT.PROD_CODE
```

```
SELECT INVOICE.INV_NUM, PRODUCT.PROD_CODE, PROD_DESCRIPTION
FROM INVOICE
JOIN INV_LINE ON INVOICE.INV_NUM = INV_LINE.INV_NUM
JOIN PRODUCT ON INV_LINE.PROD_CODE = PRODUCT.PROD_CODE;
```

JOIN ON Clause is used EVEN when tables have no common attributes

Syntax:
SELECT column-list **FROM** table 1 **JOIN** table2 **on** join-condition

JOIN tables – Outer Join

- Examples

Syntax:
SELECT column-list FROM table 1 LEFT [OUTER] JOIN table2 ON join-condition

SELECT column-list FROM table 1 RIGHT [OUTER] JOIN table2 ON join-condition

```
SELECT PROD_CODE, VENDOR.VEND_CODE, VEND_NAME
FROM VENDOR
LEFT JOIN PRODUCT
ON VENDOR.VEND_CODE = PRODUCT.VEND_CODE;
```

```
SELECT PROD_CODE, VENDOR.VEND_CODE, VEND_NAME
FROM VENDOR
RIGHT JOIN PRODUCT
ON VENDOR.VEND_CODE = PRODUCT.VEND_CODE;
```

More JOINs

- Examples

```
SELECT DISTINCT CUST_NUM, CUST_LNAME, CUST_FNAME
FROM CUSTOMER JOIN INVOICE USING (CUST_NUM)
              JOIN INV_LINE USING (INV_NUM)
              JOIN PRODUCT USING(PROD_CODE)
WHERE PROD_DESCRIPTION = 'bulb'
```

```
SELECT DISTINCT CUST_NUM, CUST_LNAME, CUST_FNAME
FROM CUSTOMER JOIN INVOICE USING (CUST_NUM)
              JOIN INV_LINE USING (INV_NUM)
              JOIN PRODUCT USING(PROD_CODE)
WHERE PROD_DESCRIPTION LIKE '%painter%' OR PROD_DESCRIPTION LIKE '%key%';
```

More Subqueries

- Example

```
SELECT PROD_CODE, PROD_QOH * PROD_PRICE
FROM PRODUCT
WHERE PROD_QOH * PROD_PRICE >
ALL (SELECT PROD_QOH * PROD_PRICE
      FROM PRODUCT
      WHERE VEND_CODE IN (SELECT VEND_CODE FROM VENDOR
                          WHERE VEND_NAME = ‘Costco'));
```

```
SELECT DISTINCT CUSTOMER.CUST_NUM, CUSTOMER.CUST_LNAME
FROM CUSTOMER,
(SELECT INVOICE.CUST_NUM FROM INVOICE NATURAL JOIN INV_LINE WHERE PROD_CODE = '0923-sat') CP1,
(SELECT INVOICE.CUST_NUM FROM INVOICE NATURAL JOIN INV_LINE WHERE PROD_CODE = 'ast-124') CP2
WHERE CUSTOMER.CUST_NUM = CP1.CUST_NUM AND
CP1.CUST_NUM = CP2.CUST_NUM
```

SQL Functions

- May appear anywhere in a SQL statement
- Operate over single row (Aggregate functions - e.g. COUNT, MAX, MIN, SUM, AVG operate over multiple rows)
- http://dev.mysql.com/doc/refman/5.7/en/date-and-time-functions.html
- https://dev.mysql.com/doc/refman/5.7/en/mathematical-functions.html
- http://dev.mysql.com/doc/refman/5.7/en/string-functions.html
- http://dev.mysql.com/doc/refman/5.7/en/cast-functions.html

Relational Set Operators

```
SELECT CUST_LNAME, CUST_FNAME,  CUST_INITIAL,
CUST_AREACODE
FROM   CUSTOMER
UNION
SELECT CUST_LNAME, CUST_FNAME,  CUST_INITIAL,
CUST_AREACODE
FROM   NEW_CUSTOMER
```

remove duplication

```
SELECT CUST_LNAME, CUST_FNAME,  CUST_INITIAL,
CUST_AREACODE
FROM   CUSTOMER
UNION ALL
SELECT CUST_LNAME, CUST_FNAME,  CUST_INITIAL,
CUST_AREACODE
FROM   NEW_CUSTOMER
```

retain duplicate rows

Exercise – A Consulting Company (Cont.)

- Using the EMPLOYEE, JOB, and PROJECT tables, write the SQL code that will produce the results shown below.

PROJ_NAME	PROJ_VALUE	PROJ_BALANCE	EMP_LNAME	EMP_FNMAE	EMP_INITIAL	JOB_CODE	JOB_DESCRIPTION	JOB_CHG_HOUR
Rolling Tide	805000.00	500345.20	Senior	David	H	501	Systems Analyst	96.75
Evergreen	1454500.00	1002350.00	Arbough	June	E	500	Programmer	35.75
Starflight	2650500.00	2309880.00	Alonzo	Maria	D	501	Systems Analyst	96.75
Amber Wave	3500500.00	2110346.00	Washington	Ralph	B	501	Systems Analyst	96.75

Exercise – A Consulting Company (Cont.)

- Using the data in the ASSIGNMENT table, write the code that will yield the total number of hours worked for each employee and the total charges stemming from those hours worked. The results are shown below. Your results may be a bit different from the following results if you did not insert all records of ASSIGNMENT in previous exercises.

EMP_NUM	EMP_LNAME	SumOfASSIGN_HOURS	SumOfASSIGN_CHARGE
101	News	3.1	387.50
102	Senior	10.1	904.90
103	Arbough	10.5	887.25
104	Ramoras	2.8	270.90
105	Johnson	8.2	931.00
107	Alonzo	4.3	451.50
108	Washington	8.3	840.15