Describing the data

- We need to define the data we want to store in tables
- What are the needed tables?
- What data is relevant per table?
- To do that, we need to know what data types that are available

Participant table
id (int, unique)
first name (string)
last name (string)
country (string)
gender (string)
level of English (int)
has Israeli citizenship (boolean)

Payment table
id (int, unique)
participant id (int)
sum (int)
date (string)
payment method (string, limited to cash/paypal)
authorization code (string)

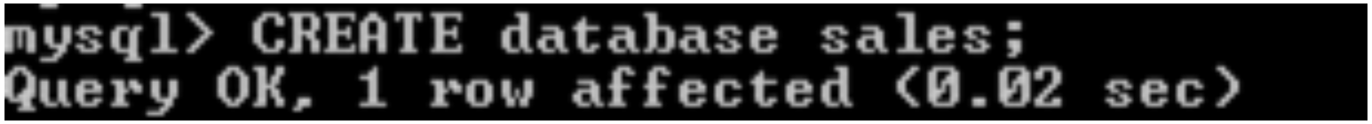
Common MySQL Data Types

- Textual
  - ➢**CHAR**- fixed-length string, up to 255 chars long
  - ➢**VARCHAR** – variable-length string, up to 65,535 chars long (64K)
  - ➢**TEXT** – fixed-length string, up to 65,535 chars
  - ➢**ENUM** – a limited list of options to choose from
- Numerical
  - ➢**INT** – number between -2147483648 to 2147483647
  - ➢**DOUBLE** – a number with a floating decimal point
- Data/Time
  - ➢**DATE**– a date, formatted like: yyyy-mm-dd
  - ➢**DATETIME** – date and time
  - ➢**TIMESTAMP** – the value in seconds since Unix epoch time '1970-01-01 00:00:00' UTC
- and more …

Creating and using DBs

- Creating a Database – this command will create an empty database, without any tables in it

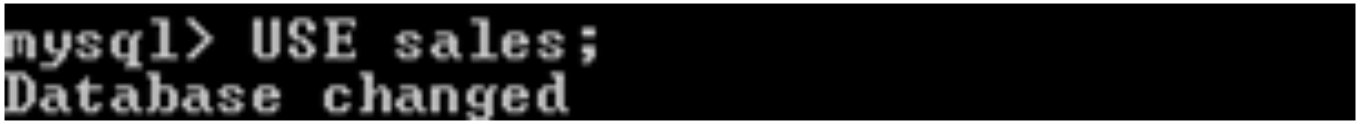
```
CREATE database sales;
```



```
mysql> CREATE database sales;
Query OK, 1 row affected (0.02 sec)
```

- "Using" a Database – this command changes the "active" database, so that we don't have to prefix table names with the database's name

```
USE sales;
```

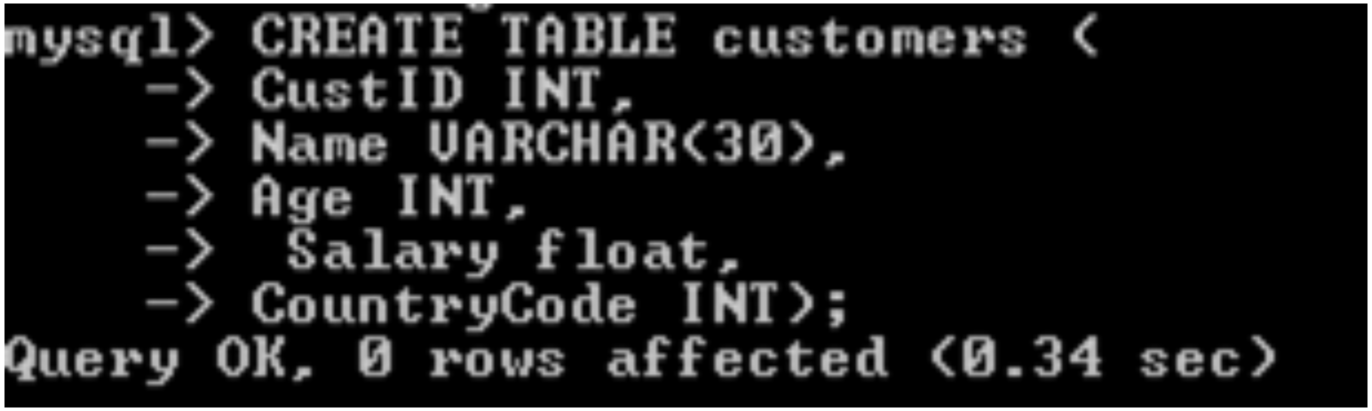


```
mysql> USE sales;
Database changed
```

Creating a Table

- Creating a table in a database

```
CREATE TABLE customers (
CustID INT,
Name VARCHAR(30),
Age INT,
Salary FLOAT,
CountryCode INT);
```

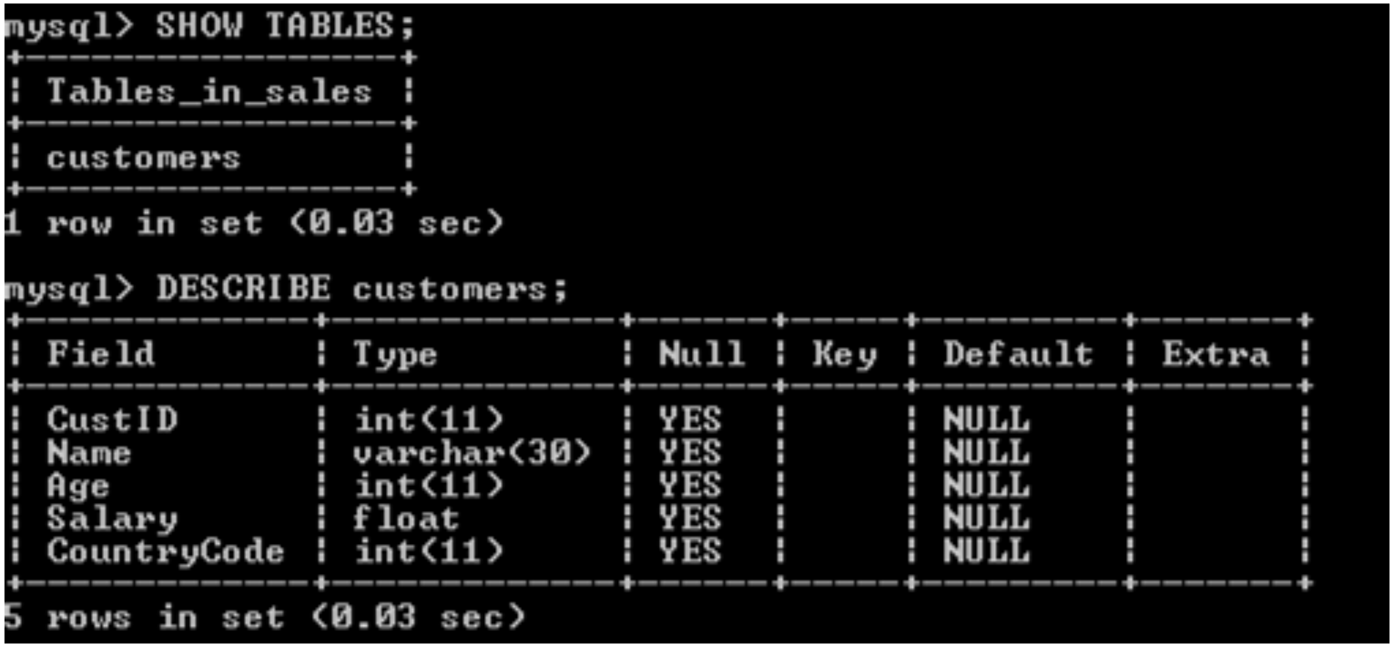


```
mysql> CREATE TABLE customers (
    -> CustID INT,
    -> Name VARCHAR(30),
    -> Age INT,
    ->  Salary float,
    -> CountryCode INT);
Query OK, 0 rows affected (0.34 sec)
```

Showing Metadata

- We can use the following commands to understand the structure of the database and that of a specific table
  - Note: we can't run these before using a specific database.

```
SHOW TABLES;
DESCRIBE customers;
```



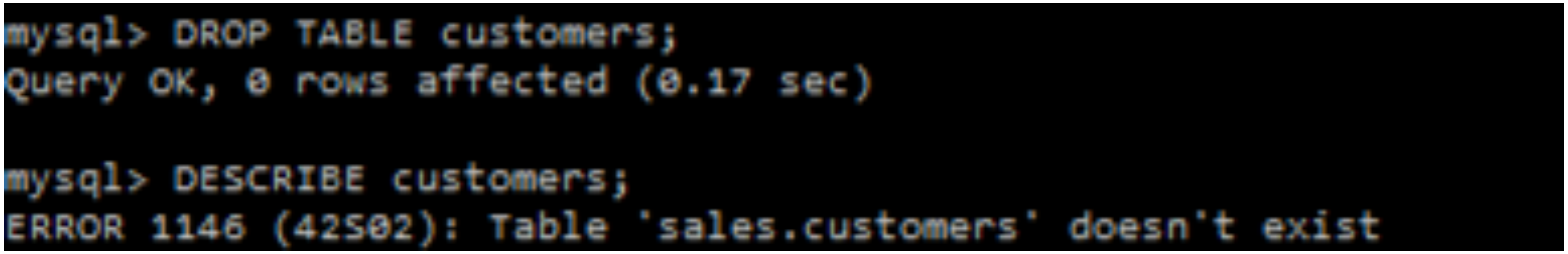
```
mysql> SHOW TABLES;
+-----------------+
| Tables_in_sales |
+-----------------+
| customers       |
+-----------------+
1 row in set (0.03 sec)

mysql> DESCRIBE customers;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| CustID      | int(11)     | YES  |     | NULL    |       |
| Name        | varchar(30) | YES  |     | NULL    |       |
| Age         | int(11)     | YES  |     | NULL    |       |
| Salary      | float       | YES  |     | NULL    |       |
| CountryCode | int(11)     | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
5 rows in set (0.03 sec)
```

The DROP Command

- The DROP SQL command is dropping (deleting) the entire table from your database.

```
DROP TABLE customers;
```



```
mysql> DROP TABLE customers;
Query OK, 0 rows affected (0.17 sec)

mysql> DESCRIBE customers;
ERROR 1146 (42S02): Table 'sales.customers' doesn't exist
```

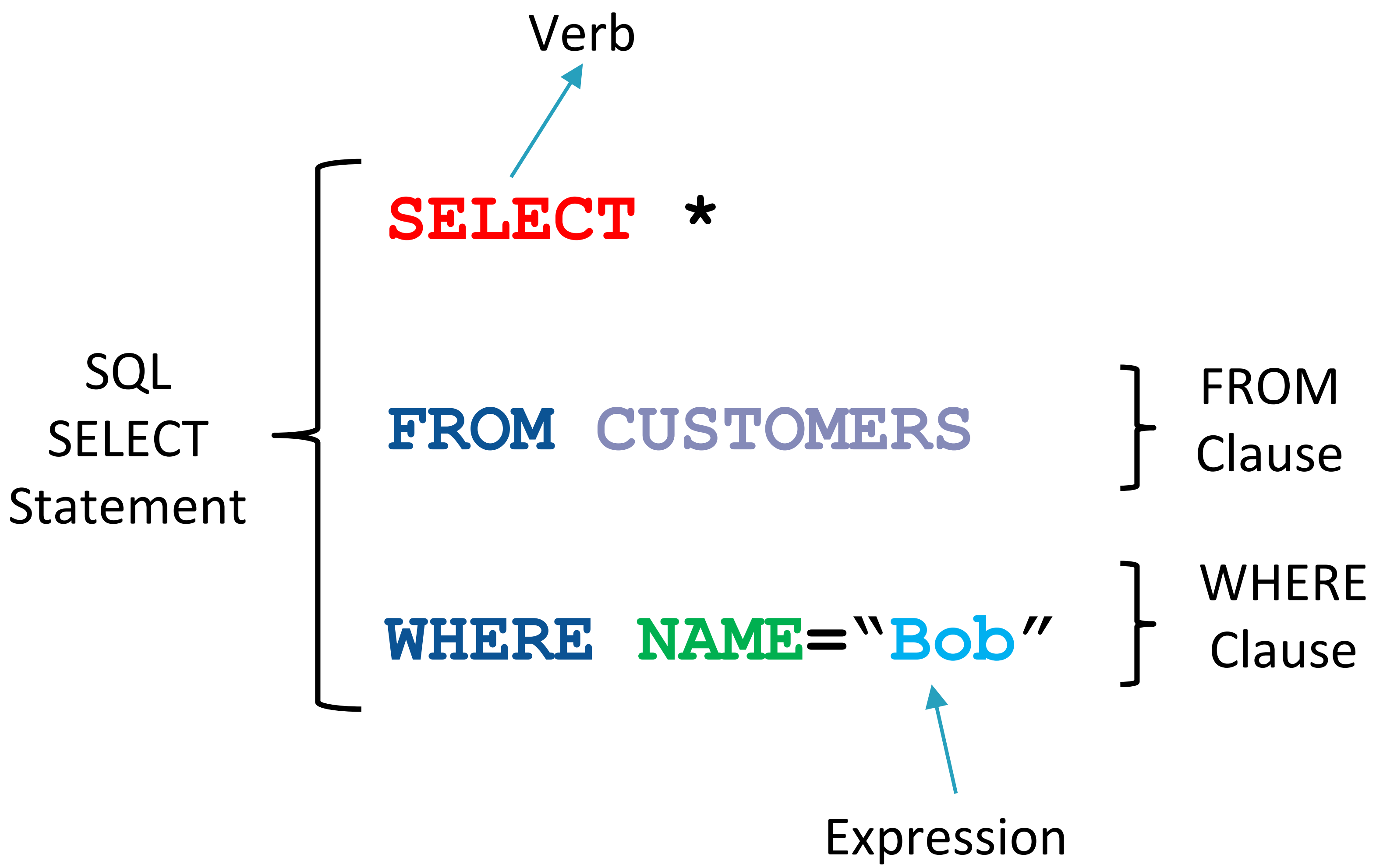
- In case you want to drop an entire database, you can use

```
DROP DBNAME;
```

SQL Statements

- Every statement begins with a **verb**, describing what this statement does
- One or more **clauses** follow
- Some clauses are optional, while others are required

SQL Statement Structure



```
SELECT *

FROM CUSTOMERS

WHERE NAME="Bob"
```

SQL Verbs

- **`INSERT`** – create information
- **`SELECT`** – read information
- **`UPDATE`** – update information
- **`DELETE`** – delete information

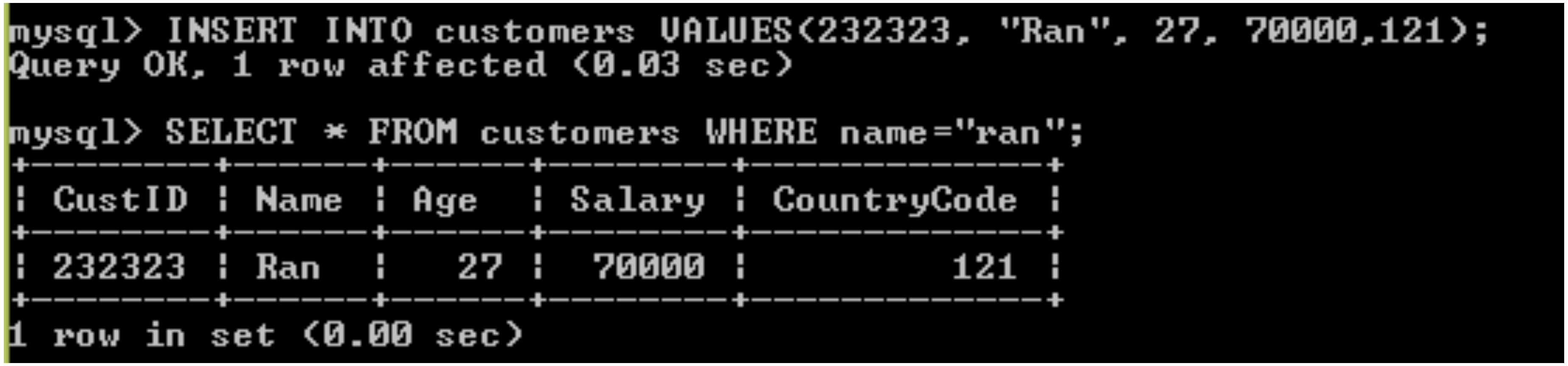
Creating Data - INSERT Statement

- To insert one new row, use the INSERT statement:

```
INSERT INTO customers
VALUES(232323, "Ran", 27, 70000,121);
```

- To see the change reflected in the database:

```
SELECT * FROM customers WHERE name="Ran";
```



```
mysql> INSERT INTO customers VALUES(232323, "Ran", 27, 70000,121);
Query OK, 1 row affected (0.03 sec)

mysql> SELECT * FROM customers WHERE name="ran";
+--------+-------+------+--------+-------------+
| CustID | Name  | Age  | Salary | CountryCode |
+--------+-------+------+--------+-------------+
| 232323 | Ran   |   27 |  70000 |         121 |
+--------+-------+------+--------+-------------+
1 row in set (0.00 sec)
```

Creating Data - INSERT Statement

- If there are missing values, use the **NULL** value as placeholder

```
INSERT INTO customers
VALUES(232323,NULL, 27, 70000,121);
```

- The sequence of values is dependent on the order in which the fields were specified in the CREATE TABLE command

- It is possible to explicitly specify the names of the fields to be set. Now, the order of the values needs to match the order of the fields as specified:

```
INSERT INTO customers (Name, Age, Salary,
CountryCode, CustID)
VALUES(NULL, 27, 70000,121, 232323);
```

Multi-Row Insert

- It is also possible to specify any number of value sets using a single INSERT statement:

```
INSERT INTO customers
VALUES(111111,NULL, 17, 70000,121),
 (222222,NULL, 27, 80000,121),
 (333333,NULL, 37, 90000,121);
```

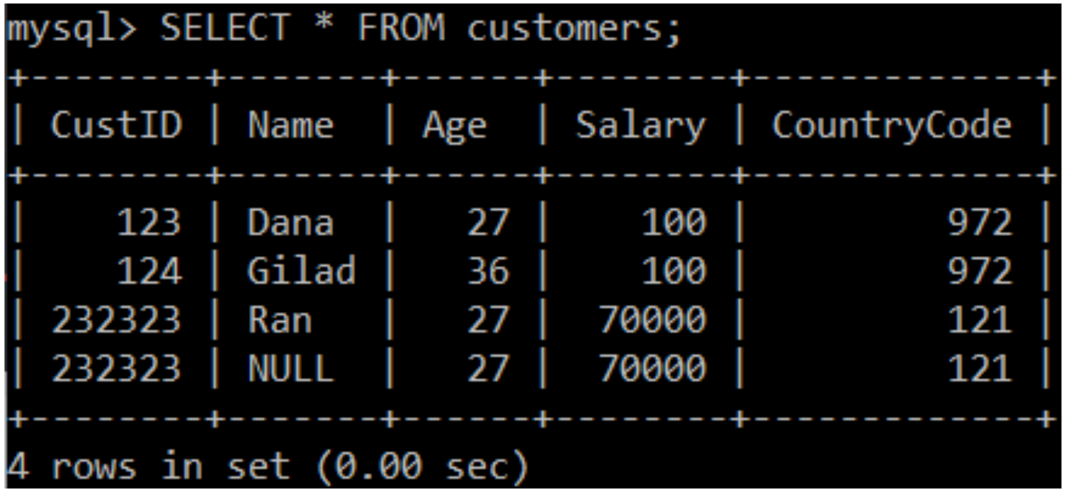
- To insert multiple rows, separate each row with a comma:

```
INSERT INTO customers
VALUES(232323, "Ran", 27, 70000, 121),
VALUES(232345, "Bob", 29, 60000, 121);
```

Reading data - SELECT Statement

- We use a SELECT query to fetch information from a table

```
SELECT * FROM customers;
```

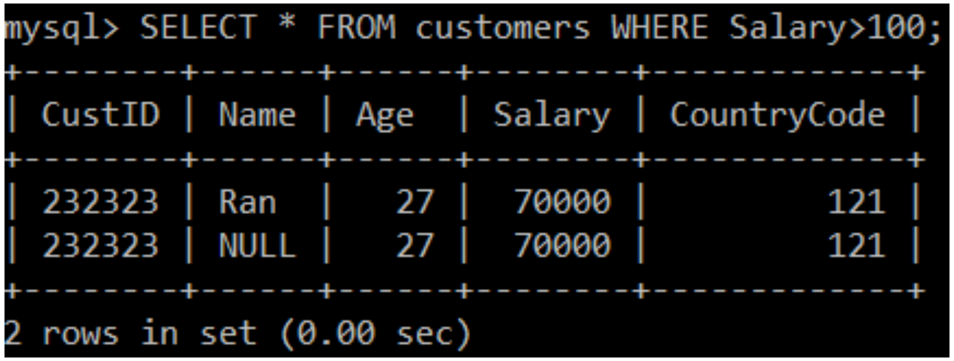


```
mysql> SELECT * FROM customers;
+--------+-------+------+--------+-------------+
| CustID | Name  | Age  | Salary | CountryCode |
+--------+-------+------+--------+-------------+
|    123 | Dana  |   27 |    100 |         972 |
|    124 | Gilad |   36 |    100 |         972 |
| 232323 | Ran   |   27 |  70000 |         121 |
| 232323 | NULL  |   27 |  70000 |         121 |
+--------+-------+------+--------+-------------+
4 rows in set (0.00 sec)
```

WHERE clause

- Unlike the **INSERT** statement (that uses only to create new data), the other three statements enable us to specify a condition indicating which rows to operate on
- The **WHERE** clause is comprised of boolean expressions:
  - != / = to check (in)equality
  - >, >= , <=, <
  - Logical operators - AND, OR
  - List of values lookup operator - IN

```
SELECT * FROM customers WHERE Salary > 100;
```

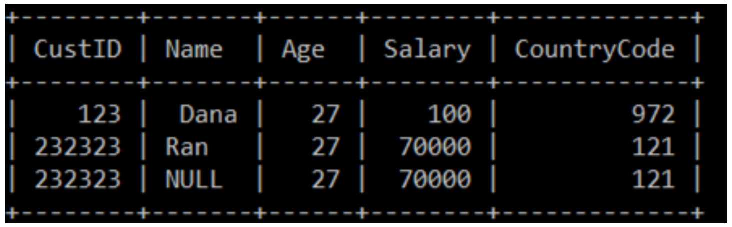


```
mysql> SELECT * FROM customers WHERE Salary>100;
+--------+------+------+--------+-------------+
| CustID | Name | Age  | Salary | CountryCode |
+--------+------+------+--------+-------------+
| 232323 | Ran  |   27 |  70000 |         121 |
| 232323 | NULL |   27 |  70000 |         121 |
+--------+------+------+--------+-------------+
2 rows in set (0.00 sec)
```

WHERE clause

- Using logical operators:

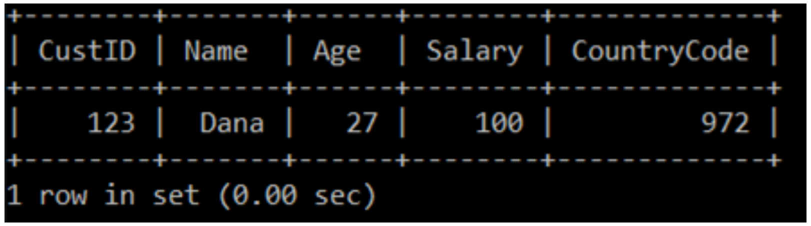
```
SELECT *
FROM customers
WHERE age < 30 OR name=" Dana";
```



```
+---------+--------+------+--------+-------------+
| CustID  | Name   | Age  | Salary | CountryCode |
+---------+--------+------+--------+-------------+
|     123 |   Dana |   27 |    100 |         972 |
|  232323 | Ran    |   27 |  70000 |         121 |
|  232323 | NULL   |   27 |  70000 |         121 |
+---------+--------+------+--------+-------------+
```

- Using the IN operator:

```
SELECT *
FROM customers
WHERE name in ("Bob", "Dana");
```



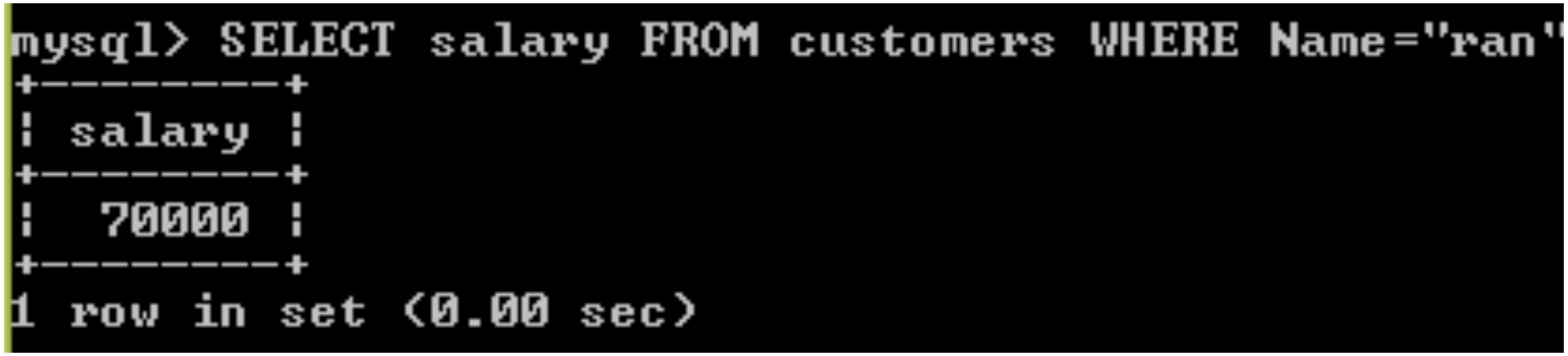
```
+---------+--------+------+--------+-------------+
| CustID  | Name   | Age  | Salary | CountryCode |
+---------+--------+------+--------+-------------+
|     123 |   Dana |   27 |    100 |         972 |
+---------+--------+------+--------+-------------+
1 row in set (0.00 sec)
```

Selecting a Subset of Fields

- It is possible to fetch only those columns which interest us, in any order we like:

```
SELECT salary
FROM customers
WHERE Name="ran";
```

- In this case, only the “salary” field is returned for every row matching the WHERE clause condition

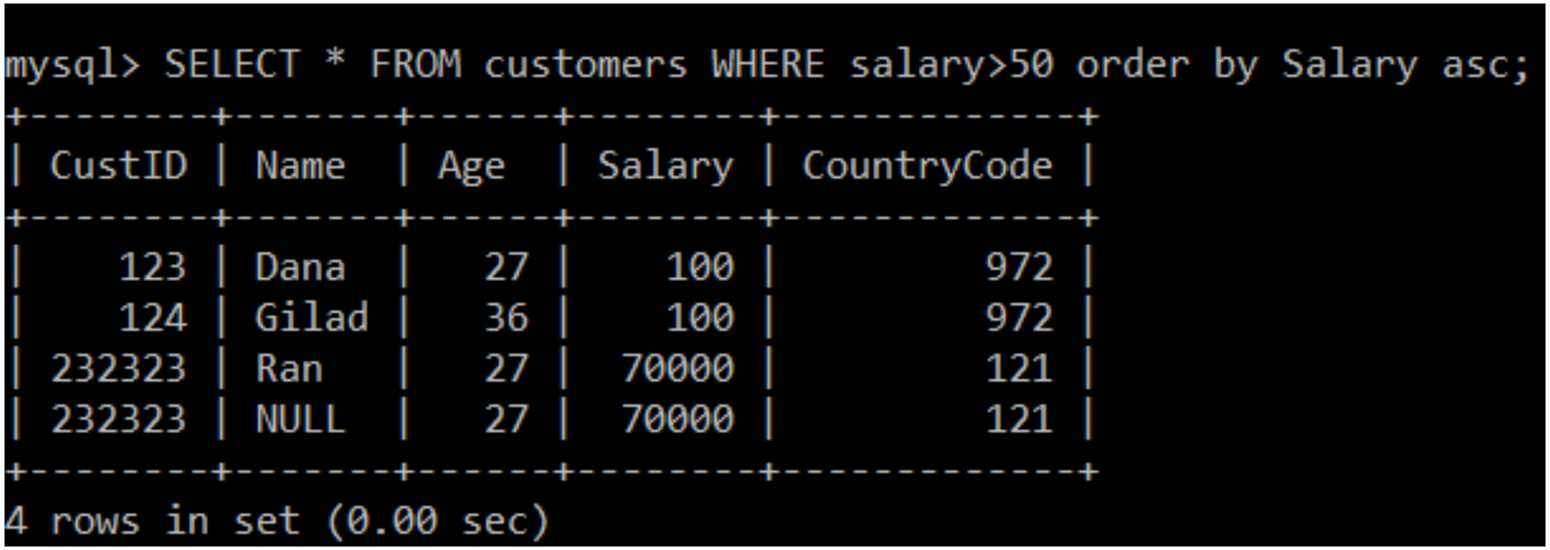


```
mysql> SELECT salary FROM customers WHERE Name="ran"
+--------+
| salary |
+--------+
|  70000 |
+--------+
1 row in set (0.00 sec)
```

ORDER BY clause

- The **ORDER BY** clause allows us to sort the results based on the values of a field, in an ascending or descending order
- More than one field may be specified, allowing for inner-sort

```
SELECT * FROM customers WHERE salary  > 50
order by Salary asc;
```



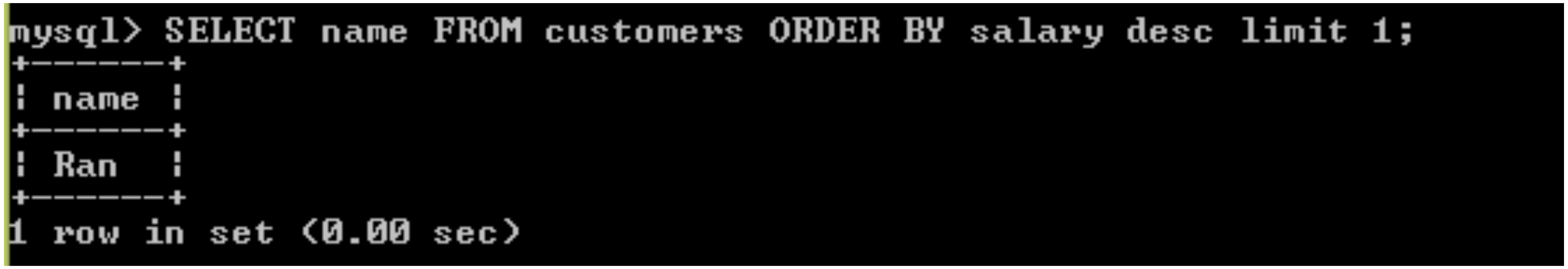
```
mysql> SELECT * FROM customers WHERE salary>50 order by Salary asc;
+--------+-------+------+--------+-------------+
| CustID | Name  | Age  | Salary | CountryCode |
+--------+-------+------+--------+-------------+
|    123 | Dana  |   27 |    100 |         972 |
|    124 | Gilad |   36 |    100 |         972 |
| 232323 | Ran   |   27 |  70000 |         121 |
| 232323 | NULL  |   27 |  70000 |         121 |
+--------+-------+------+--------+-------------+
4 rows in set (0.00 sec)
```

LIMIT clause

- The LIMIT clause restricts the number of results fetched

```
SELECT name
FROM customers
ORDER BY salary desc
limit 1;
```

- What is the logical function performed by the query?



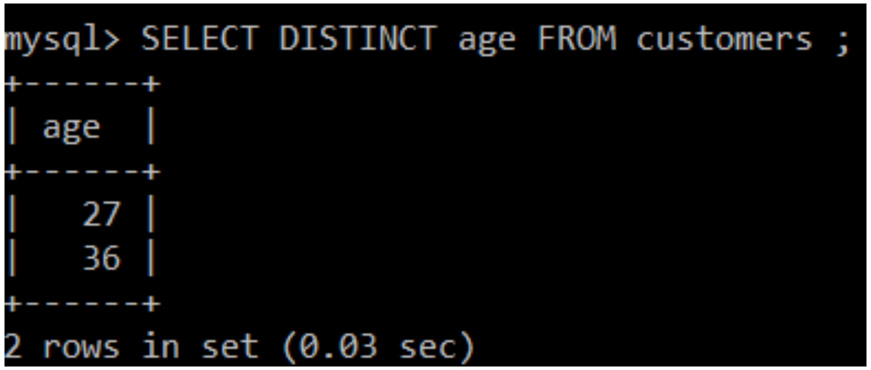
```
mysql> SELECT name FROM customers ORDER BY salary desc limit 1;
+-------+
| name  |
+-------+
| Ran   |
+-------+
1 row in set (0.00 sec)
```

Fetching the person with the highest salary

DISTINCT - SELECT modifier

- DISTINCT fetches only the unique values of a field

```
SELECT DISTINCT age FROM customers ;
```



```
mysql> SELECT DISTINCT age FROM customers ;
+------+
| age  |
+------+
|   27 |
|   36 |
+------+
2 rows in set (0.03 sec)
```

- What would happen if we specified multiple fields following the DISTINCT modifier?

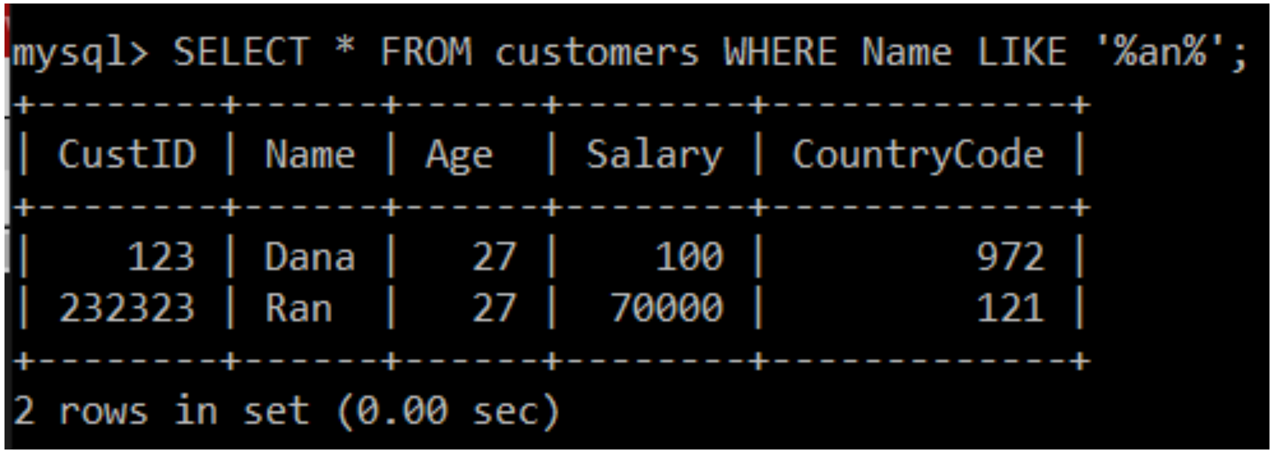
Only rows representing unique combinations of the specified columns would be fetched.

LIKE - WHERE clause operator

- The LIKE operator matches substrings within textual columns
- The % sign denotes “any character”. Hence:

- `LIKE ‘%moo’` - matches strings ending with “moo”
- `LIKE ‘moo%’` - matches strings beginning with “moo”
- `LIKE ‘%moo%’` - matches the substring “moo” anywhere in the string

```
SELECT * FROM customers WHERE Name LIKE '%an%';
```

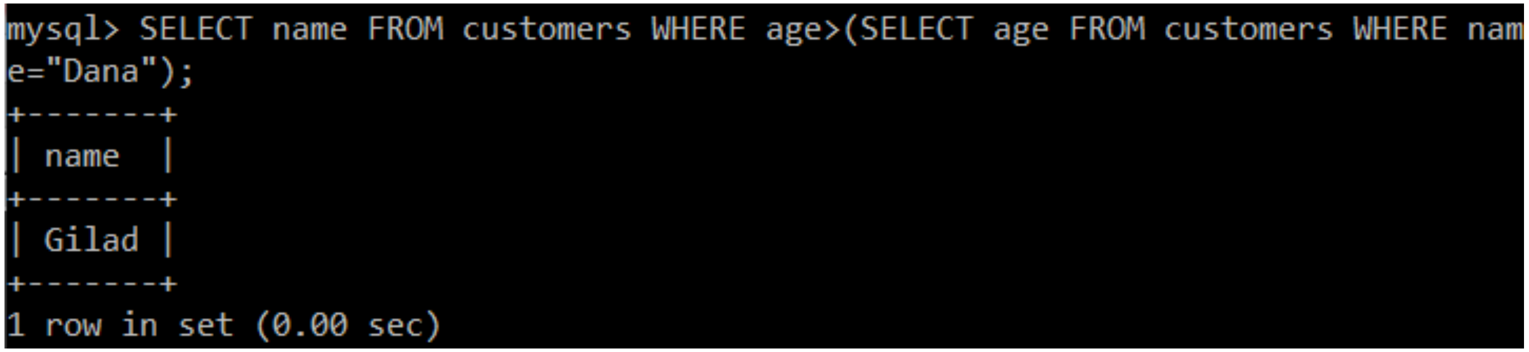


```
mysql> SELECT * FROM customers WHERE Name LIKE '%an%';
+--------+------+------+--------+-------------+
| CustID | Name | Age  | Salary | CountryCode |
+--------+------+------+--------+-------------+
|    123 | Dana |   27 |    100 |         972 |
| 232323 | Ran  |   27 |  70000 |         121 |
+--------+------+------+--------+-------------+
2 rows in set (0.00 sec)
```

Subqueries

- It is possible to embed a query within the WHERE clause
- The embedded query is called a “subquery”
- Subquery values can be used in conditions
- The subquery may fetch data from other tables

```
SELECT name FROM customers
WHERE age > (SELECT age FROM customers WHERE
               name="Dana");
```



```
mysql> SELECT name FROM customers WHERE age>(SELECT age FROM customers WHERE nam
e="Dana");
+-------+
| name  |
+-------+
| Gilad |
+-------+
1 row in set (0.00 sec)
```

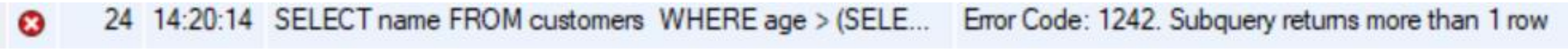
Fetches only those customers who are older than Dana.

Subqueries

```
SELECT name FROM customers
WHERE age > (SELECT age FROM customers WHERE
             name="Dana");
```

- **The return value from the subquery should be only one row.**

This query won't work if we have more than one "Dana" in the database

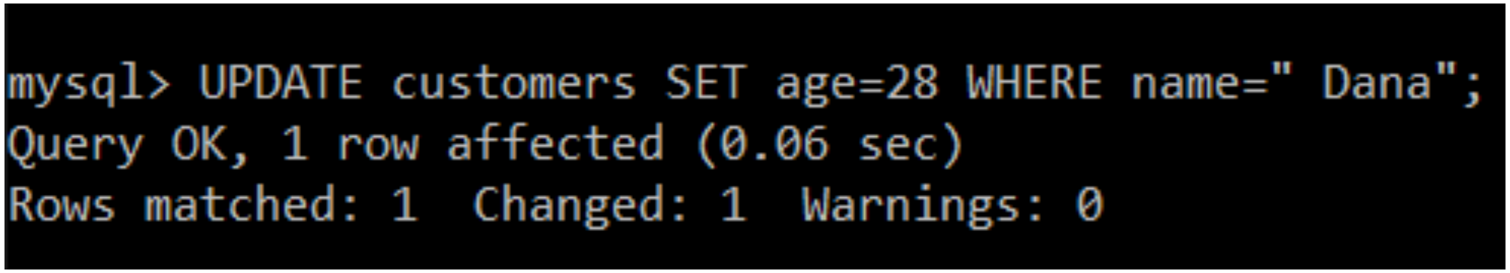


```
24  14:20:14  SELECT name FROM customers  WHERE age > (SELE...  Error Code: 1242. Subquery returns more than 1 row
```

Modifying Data - UPDATE statement

- Updates the values of field(s), per row
- One or more fields may be assigned new values using the **`SET`** clause
- As usual, the WHERE clause determines which rows are to be affected

```
UPDATE customers
SET age=28, salary=75000
WHERE name="Dana";
```

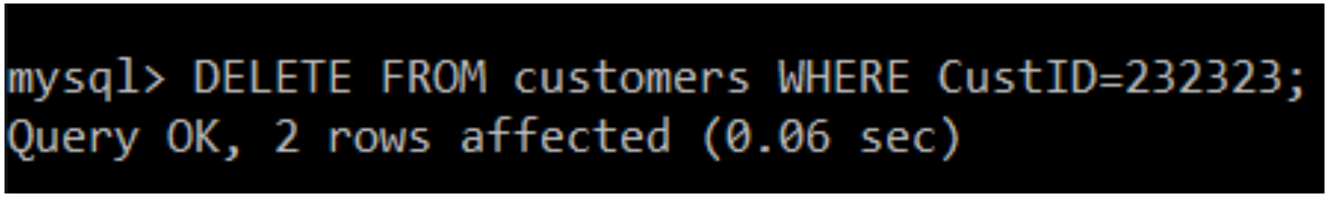


```
mysql> UPDATE customers SET age=28 WHERE name=" Dana";
Query OK, 1 row affected (0.06 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Deleting Data - DELETE statement

- Deletes one or more rows from a table
- The WHERE clause can be used to restrict the number of rows deleted

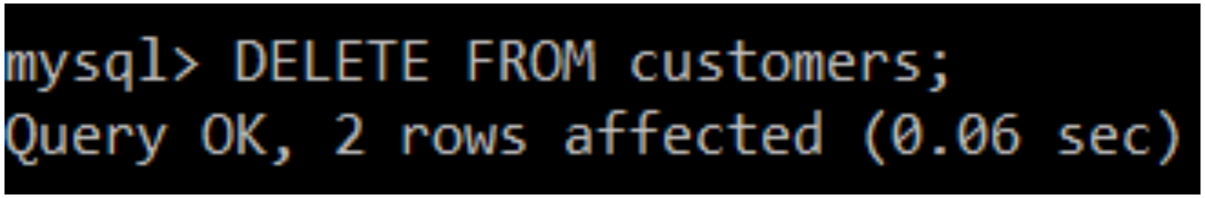
```
DELETE FROM customers
WHERE CustID=232323;
```



```
mysql> DELETE FROM customers WHERE CustID=232323;
Query OK, 2 rows affected (0.06 sec)
```

- To delete all rows in a table (careful!):

```
DELETE FROM customers;
```



```
mysql> DELETE FROM customers;
Query OK, 2 rows affected (0.06 sec)
```