Reading One: Definition of terms

Rows/Records

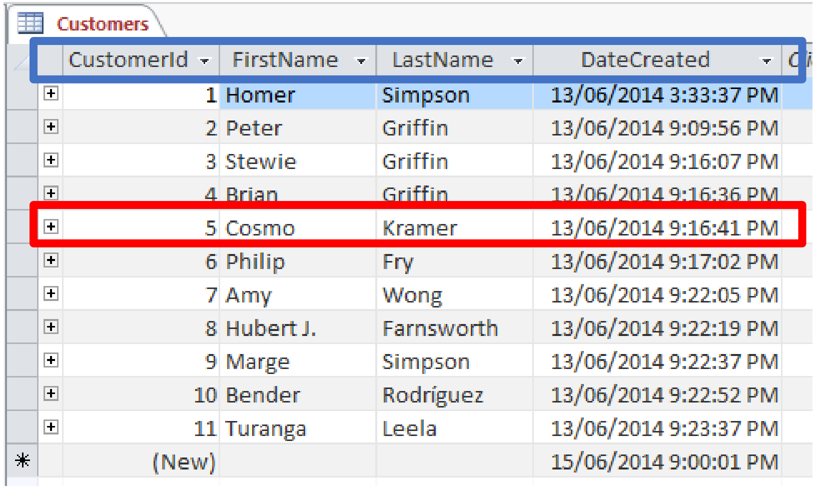
Each value entered into a table is called a **data value**. **Data values** make up a record in a table.

A **row** is also called a **record** (that is each entry of a table)

Columns/Field

A **column** in a table contains specific information about every record in a table. A column is also called a **field** (the two can be used interchangeably).

Consider the customers table below:



Customers

CustomerId	FirstName	LastName	DateCreated
1	Homer	Simpson	13/06/2014 3:33:37 PM
2	Peter	Griffin	13/06/2014 9:09:56 PM
3	Stewie	Griffin	13/06/2014 9:16:07 PM
4	Brian	Griffin	13/06/2014 9:16:36 PM
5	Cosmo	Kramer	13/06/2014 9:16:41 PM
6	Philip	Fry	13/06/2014 9:17:02 PM
7	Amy	Wong	13/06/2014 9:22:05 PM
8	Hubert J.	Farnsworth	13/06/2014 9:22:19 PM
9	Marge	Simpson	13/06/2014 9:22:37 PM
10	Bender	Rodríguez	13/06/2014 9:22:52 PM
11	Turanga	Leela	13/06/2014 9:23:37 PM
(New)			15/06/2014 9:00:01 PM

The blue rectangle shows the **columns or fields** in this table, namely CustomerId, FirstName, LastName, DateCreated, etc.

The red rectangle shows the rows or records. This shows that there are 11 **records or rows** in this customers table.

Database Objects

A **database object** is any defined object in a database that is used to store or reference data. Anything which we make from **CREATE command** is known as Database Object. It can be used to hold and manipulate the data.

Examples includes: tables, views, stored procedures, functions, index, sequence, etc.

Note: Tables are the most commonly used database object.

Reading two: Standard SQL Syntax/Commands

SQL syntax comprises several types of statements that allow you to perform various commands and operations. There are four (4) of these statements in SQL.

- ✓ Data Definition Language (DDL): creation of data
- ✓ Data Manipulation Language (DML): manipulation of data
- ✓ Data Control Language (DCL): assignment and removal of permissions to use this data
- ✓ Transaction Control Language (TCL): saving and restoring changes to a database

DDL: Data Definition Language	DML: Data Manipulation Language	DCL: Data Control Language	TCL: Transaction Control Language
Create: Creates a new table, a view of a table, or other object in database	**Select:** Retrieves certain records from one or more tables	**Grant:** Gives a privilege or certain permissions to users	**Commit:** saves the transaction in the database
Alter: Modifies an existing database object, such as a table.	**Insert:** Used to insert data into tables	**Revoke:** Takes back privileges granted from database user	**Rollback:** reverts to the last non-committed state
Drop: Deletes an entire table, a view of a table or other object in the database.	**Update:** Updates records. It allows you to renew existing data of your tables		
Rename: allows you to rename an object	**Delete:** Deletes records		
Truncate: instead of deleting an entire table through DROP, we can also remove its data and continue to have the table as an object in the database			

Reading three: Datatypes in SQL

We must always specify the type of data that will be inserted in each column of the table**.** Different data types represent different types of information that can be contained in a specific column.

Data types and constraints are applied to fields and they define the type of data that the field will accept.

Learn more about datatypes in PostgreSQL here.

Reading four: SQL Constraints

Constraint are specific rules, or limits, that we define in our tables. The role of constraints is to outline the existing relationships between different tables in our database. e.g. NOT NULL

SQL Constraints are:

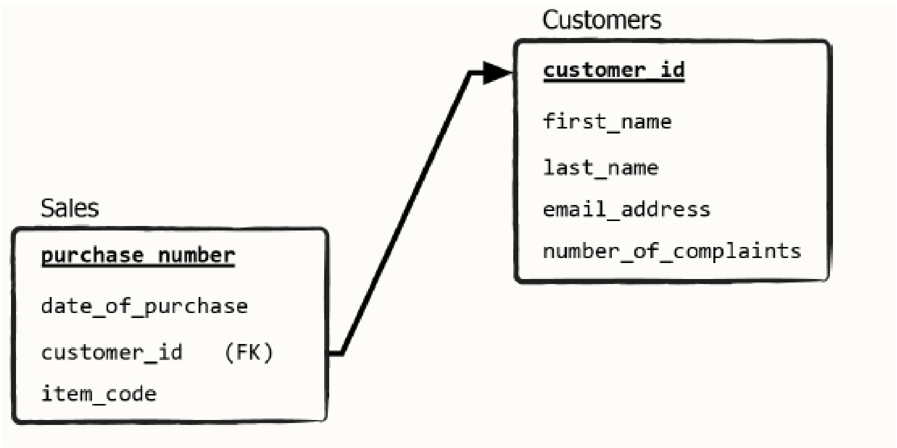
- **Entity Integrity Constraint:**

  This constraint emphasis that the **Primary Key (PK)** cannot have an unknown value. A column whose value exists and is unique for every record in a table is called a **primary key.**

  Each table can have one and only one primary key**.** Primary keys are the **unique identifiers of a table** and cannot contain null (missing) values. Also, not all tables you work with will have a primary key

- **Referential Integrity Constraint:**

  This ensures validity of the data using a combination of primary keys and foreign keys. A foreign key (FK) points to a column of another table and, thus, links the two tables. The foreign key maintains the *referential integrity* within the database



  In the diagram above, customer_id is a FK in the sales table. However, in the customers table, customer_id is the PK. Clearly, there is a link between these tables. This explains the intuition behind relational databases.

- **Semantic Integrity Constraint:**

  This constraint deals with correctness of the data. This means each field must take the appropriate data type.

- **Domain Constraint:**

  This specifies the permissible values for a given attribute. Under the domain constraint, UNIQUE KEY is used whenever you would like to specify that you don't want to see duplicate data in a given field. If you attempt to insert an already existing, duplicate value in the **unique column**, SQL will display an error.

- **Null Constraint:**

  This specifies that attribute values cannot be null. To enforce the null constraint in SQL, we use NOT NULL. The "not null" restriction is applied using NOT NULL. This means that when you insert values in the table, you cannot leave the respective field empty (null).

- **Check Constraint:**

  This constraint enforces domain integrity. It does this by limiting the values that are accepted by an attribute.

Reading five: Notes on SQL Best Practices

1. **Clean Code**

Code should be *focused and understandable,* which means it must be readable, logical, and changeable. Unlike other programming languages, **SQL is not case sensitive.**

Therefore, as best practice, it is advisable to write keywords (reserved words) in UPPER case to improve readability of your code. Keywords includes, ADD, CREATE, ALTER etc.

Also, at the end of a piece od code, add semicolon **(;)** which is called a statement terminator.

2. **Assigning Names**

When assigning names to variables or SQL objects always chose shorter, meaningful names, conveying specific information. In addition, variables, objects or databases cannot have names that coincide with SQL keywords (this helps to remove confusion and ambiguity) in codes.

Also, never put a white space when giving names to tables or fields. For example:

- First Name **(wrong)**
- First_name **(correct)**
- FirstName **(correct)**

3. **Comments**

These lines of text that will not run as code; they convey a message to someone who reads our code. For one line comment, you use -- (double hyphens). For large comments, you use /* comments will be in here */

For example:

One line comment:

-- Creating the sales table

Large comment:

/*

The sales table has four columns namely

purchase_number, date_of_purchase, customer_id.

We will also add the necessary data type and constraints to these columns.

*/