SQL Cheat Sheet

* **Primary key** – a unique identifier for each row in a table. It can be composed of several fields. Aka independent or parent entity (can stand alone and needs no context to understand what it means such as the department ID within the department table because that table shows the name associate with the ID within that table)
* **Foreign key** – a reference to a primary key in another table that helps build a relationship between two tables. AKA dependent or child entity (almost meaningless without the context of the parent table such as the department ID in the employee table. We don’t know what the department ID means unless we look at the parent table containing the name associated with that ID.
* **Entity** – object defined in system model about which data is stored in the database (usually what header or name of what is in the table, such as “Course” table contains course number, title and objective, “Course” would be the entity. **Usually refers to columns**
* **Attribute** – properties or characteristics that provide supporting data for each entity (in the example above, course title and objective would be attribute as it describes and adds more detail to the entity) **usually refers to rows**
* **Table** – collection of rows
* **Database** – collection of tables

**Order of Operations**

* SELECT
  + [columns] one or all using \*
* FROM
  + [one of many] 1-single, many-JOIN
* ON
  + [JOIN] n-1 (minimum)
* WHERE
  + [filter data]
* GROUP BY
  + [aggregate data]
* HAVING
  + [filter results]
* ORDER BY
  + [sort data]
* LIMIT
  + [quantitative limitation]

**Syntax for Creating Table**

1. # Remove any databases with the same name as the one we're about to create

DROP DATABASE IF EXISTS zboiskin\_edx;

1. # Creating a Database named edx

CREATE DATABASE zboiskin\_edx;

CREATE TABLE zboiskin\_edx.RECORDS (

`course\_id` VARCHAR(300) NOT NULL,

`Course\_Short\_Title` VARCHAR(300) DEFAULT NULL,

`Course\_Long\_Title` VARCHAR(300) DEFAULT NULL,

`userid\_DI` VARCHAR(300) NOT NULL,

`gender` VARCHAR(300) DEFAULT NULL,

`grade` VARCHAR(300) DEFAULT NULL,

`incomplete\_flag` VARCHAR(300) DEFAULT NULL

);

**Syntax of Joins**

EX1.

select SalesReps.Name, Offices.City

FROM

SalesReps

LEFT JOIN

Offices

ON SalesReps.RepOffice = Offices.OfficeNbr;

EX2.

select SalesReps.Name, SUM(Customers.CreditLimit)

FROM

SalesReps

Inner JOIN

Customers

ON SalesReps.RepNbr = Customers.CustRep

GROUP BY

SalesReps.RepNbr;

EX3.

SELECT SalesReps.Name, SUM(Customers.CreditLimit)

FROM

SalesReps

LEFT JOIN

Customers

ON SalesReps.RepNbr = Customers.CustRep

WHERE Customers.CreditLimit > 2000

GROUP BY

SalesReps.RepNbr

HAVING SUM(Customers.CreditLimit) > 95000;

Other Examples

select \* from SalesReps,Offices

where SalesReps.RepOffice = Offices.OfficeNbr;

INSERT INTO SalesReps

VALUES('999', 'Zac Boiskin', NULL, 2000,50000);

Syntax Join Examples

Inner Join:

**INNER JOIN:** The INNER JOIN keyword selects all rows from both the tables as long as the condition satisfies. This keyword will create the result-set by combining all rows from both the tables where the condition satisfies i.e value of the common field will be same.

SELECT table1.column1,table1.column2,table2.column1,....

FROM table1

INNER JOIN table2

ON table1.matching\_column = table2.matching\_column;

**table1**: First table.

**table2**: Second table

**matching\_column**: Column common to both the tables.

**LEFT JOIN**: This join returns all the rows of the table on the left side of the join and matching rows for the table on the right side of join. The rows for which there is no matching row on right side, the result-set will contain *null*. LEFT JOIN is also known as LEFT OUTER JOIN.**Syntax:**

SELECT table1.column1,table1.column2,table2.column1,....

FROM table1

LEFT JOIN table2

ON table1.matching\_column = table2.matching\_column;

table1: First table.

table2: Second table

matching\_column: Column common to both the tables.

**Note**: We can also use LEFT OUTER JOIN instead of LEFT JOIN, both are same.

**FULL JOIN:** FULL JOIN creates the result-set by combining result of both LEFT JOIN and RIGHT JOIN. The result-set will contain all the rows from both the tables. The rows for which there is no matching, the result-set will contain *NULL* values.**Syntax:**

SELECT table1.column1,table1.column2,table2.column1,....

FROM table1

FULL JOIN table2

ON table1.matching\_column = table2.matching\_column;

table1: First table.

table2: Second table

matching\_column: Column common to both the tables.