SQL for Analytics

SATYAJIT PATTNAIK

SQL Timeline

Introduction

Table Basics & Selecting Data

Statements & Queries

Advanced Queries

Sorting Results

Joins

SQL Timeline Contd..

Agg Functions

Ordering

Having clause

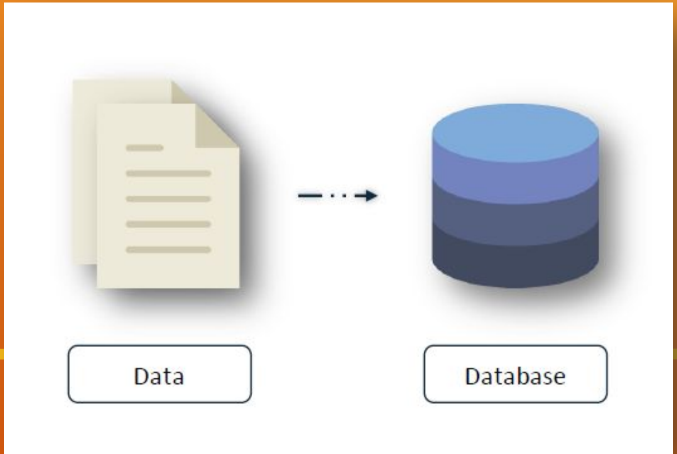
Regex

Nested Queries & Views

Normalization

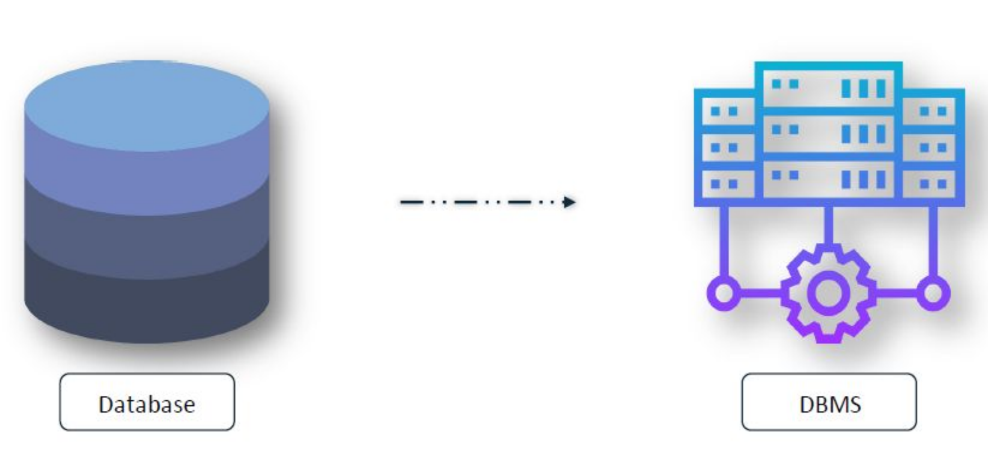
Database

Organised collection of Data stored in an electronic format

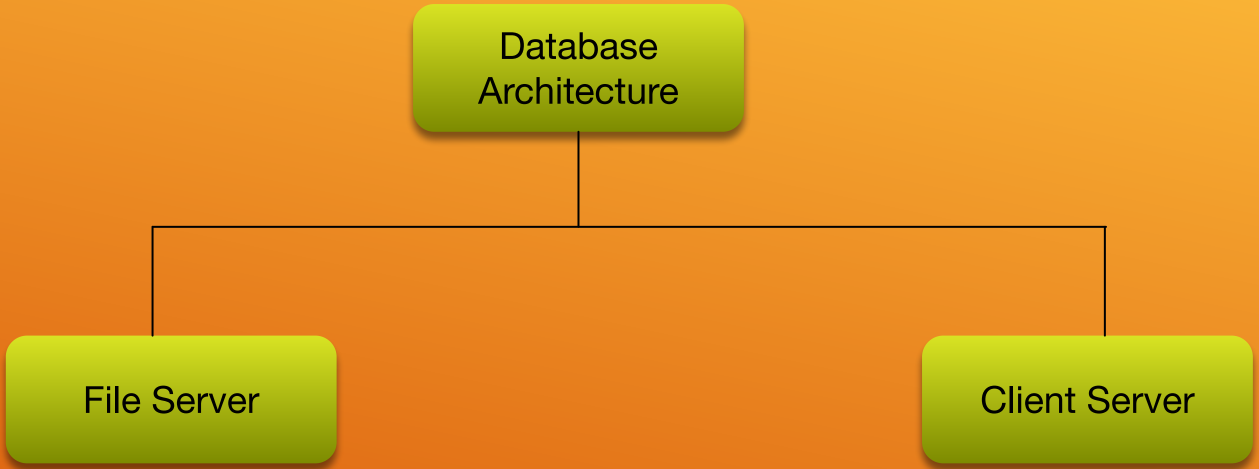


Database Management System

A System Software for creating and managing database



Database Architecture



Database Architecture

File Server

Client Server

Database Architecture

File Server

- ✔ Client request a file from the disk or a file-server
- ✔ Software opens the files, stores it in memory, makes changes in memory, and then saves it back to the disk drive (or file server)
- ✔ What happens if someone else opens the same file after you do, then you save your changes, and then they save their changes?
  - ○ Potential data loss
  - ○ Solution is file-locking
  - ○ File locking is very inconvenient however as it means that nobody else can edit the file if someone else has it open first. (Logically good for small companies)

Client Server

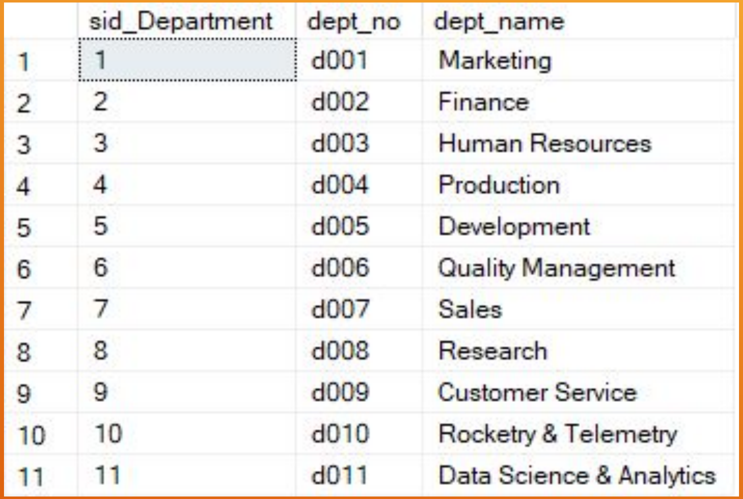
- ✔ Desktop software requests data from the database
- ✔ Data is held in memory but when a change is made, a request (SQL statement) is generated.
- ✔ When you save the data these requests are passed to the database in the order they were generated.
- ✔ So, multiple people can work in the same table at the same time with no worries
  - ○ Risk of data loss is low
  - ○ Still possible for someone else to change data after you change it.
  - ○ Versioning is an option to record the changes (DB Admins are owner of these things in production)

SQL Command Categories

Data Query Language	Data Definition Language	Data Manipulation Language	Data Control Language
Select	Create Table	Insert	Grant
	Alter Table	Update	Revoke
	Drop Table	Delete	

Data Types in SQL

Defines the type of data the column holds



	sid_Department	dept_no	dept_name
1	1	d001	Marketing
2	2	d002	Finance
3	3	d003	Human Resources
4	4	d004	Production
5	5	d005	Development
6	6	d006	Quality Management
7	7	d007	Sales
8	8	d008	Research
9	9	d009	Customer Service
10	10	d010	Rocketry & Telemetry
11	11	d011	Data Science & Analytics

Integer

Character

Different Data Types

1 Numeric

2 Character

3 Date & Time

Introduction

What is SQL?

- ✔ SQL (pronounced "ess-que-el") stands for Structured Query Language
- ✔ SQL is used to communicate with a database.
- ✔ It is the standard language for relational database management systems
- ✔ SQL statements are used to perform tasks such as update data on a database, or retrieve data from a database. Some common relational database management systems that use SQL are: Oracle, Sybase, Microsoft SQL Server, Access, Ingres, etc.
- ✔ The standard SQL commands such as "Select", "Insert", "Update", "Delete", "Create", and "Drop" can be used to accomplish almost everything that one needs to do with a database.
- ✔ SQL programming can be used to perform multiple actions on data such as :
  - ○ Querying
  - ○ Inserting
  - ○ Updating
  - ○ Deleting
  - ○ Extracting etc.

Primary & Foreign Key

A Primary key is used to ensure data in the specific column is Unique and Not Null

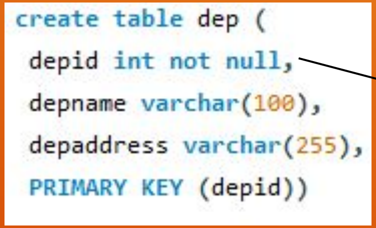
Primary key = Unique + Not Null

A foreign key is a column or group of columns in a relational database table that provides a link between data in two tables. It is a column (or columns) that references a column (most often the primary key) of another table

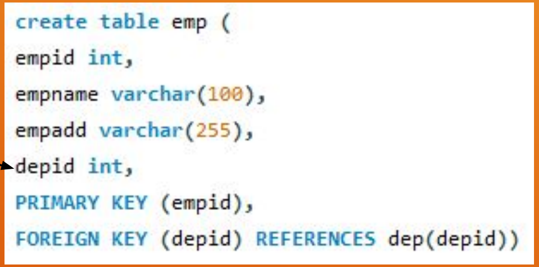
Primary & Foreign Key - Contd..

- Primary key:
  - no null values
  - unique identification

- Foreign key:
  - correspond to the values of the primary key in another table



```sql
create table dep (
 depid int not null,
 depname varchar(100),
 depaddress varchar(255),
 PRIMARY KEY (depid))
```



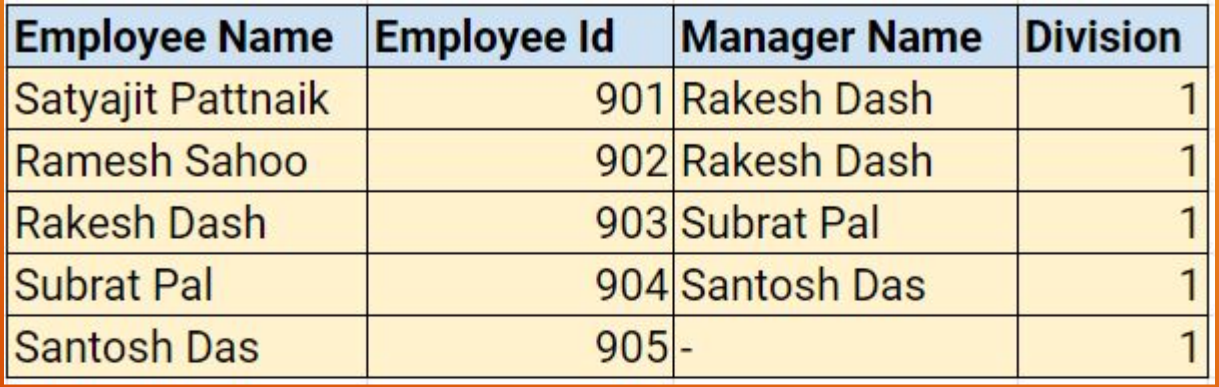
```sql
create table emp (
empid int,
empname varchar(100),
empadd varchar(255),
depid int,
PRIMARY KEY (empid),
FOREIGN KEY (depid) REFERENCES dep(depid))
```

SQL – First Step

- Create a Database
- Create a Table
- Insert Table Data
- Update Table Data
- Alter Database
- Alter Table
- Alter Column
- Duplicate a Database
- Drop a Database
- Database Backup
- Restore a Database
- Rename Database

TABLE BASICS

A relational database system contains one or more objects called tables. The data or information for the database are stored in these tables. Tables are uniquely identified by their names and are comprised of columns and rows. Columns contain the column name, data type, and any other attributes for the column. Rows contain the records or data for the columns. Here is a sample table called "employee".



Employee Name	Employee Id	Manager Name	Division
Satyajit Pattnaik	901	Rakesh Dash	1
Ramesh Sahoo	902	Rakesh Dash	1
Rakesh Dash	903	Subrat Pal	1
Subrat Pal	904	Santosh Das	1
Santosh Das	905	-	1

The LIKE pattern matching operator can also be used in the conditional selection of the where clause. Like is a very powerful operator that allows you to select only rows that are "like" what you specify. The percent sign "%" can be used as a wildcard to match any possible character that might appear before or after the characters specified.

This SQL statement will match any first names that start with 'Er'. Strings must be in single quotes.
Or you can specify:

For example:

```
select first, last, city
   from empinfo
   where first LIKE 'Er%';
```

This statement will match any last names that end in a 's'.

```
select first, last
   from empinfo
   where last LIKE '%s';
```

This will only select rows where the first name equals 'Eric' exactly

```
select * from empinfo
   where first = 'Eric';
```

SELECTING THE DATA

Sample Table: empinfo

first	last	id	age	city	state
John	Jones	99980	45	Payson	Arizona
Mary	Jones	99982	25	Payson	Arizona
Eric	Edwards	88232	32	San Diego	California
Mary Ann	Edwards	88233	32	Phoenix	Arizona
Ginger	Howell	98002	42	Cottonwood	Arizona
Sebastian	Smith	92001	23	Gila Bend	Arizona
Gus	Gray	22322	35	Bagdad	Arizona
Mary Ann	May	32326	52	Tucson	Arizona
Erica	Williams	32327	60	Show Low	Arizona
Leroy	Brown	32380	22	Pinetop	Arizona
Elroy	Cleaver	32382	22	Globe	Arizona

Enter the following sample select statements in the SQL Interpreter Form at the bottom of this page. Before you press "submit", write down your expected results. Press "submit", and compare the results.

```
select first, last, city from empinfo;

select last, city, age from empinfo
      where age > 30;

select first, last, city, state from empinfo
      where first LIKE 'J%';

select * from empinfo;

select first, last, from empinfo
      where last LIKE '%s';

select first, last, age from empinfo
      where last LIKE '%illia%';

select * from empinfo where first = 'Eric';
```

CREATING TABLES

The create table statement is used to create a new table. Here is the format of a simple create table statement:	create table "tablename" ("column1" "data type", "column2" "data type", "column3" "data type");
Format of create table if you were to use optional constraints:	create table "tablename" ("column1" "data type" [constraint], "column2" "data type" [constraint], "column3" "data type" [constraint]); [ ] = optional
Note: You may have as many columns as you'd like, and the constraints are optional.	Example: create table employee (first varchar(15), last varchar(20), age number(3), address varchar(30), city varchar(20), state varchar(20));

To create a new table,

- ❑ enter the keywords create table followed by the table name,
- ❑ followed by an open parenthesis,
- ❑ followed by the first column name,
- ❑ followed by the data type for that column,
- ❑ followed by any optional constraints, and followed by a closing parenthesis.

It is important to make sure you use an open parenthesis before the beginning table, and a closing parenthesis after the end of the last column definition.

Make sure you separate each column definition with a comma. **All SQL statements should end with a ";".**

The table and column names;

- must start with a letter
- and can be followed by letters, numbers, or underscores
- not to exceed a total of 30 characters in length.

Do not use any SQL reserved keywords as names for tables or column names (such as "select", "create", "insert", etc).

Data types specify what the type of data can be for that particular column. If a column called "Last_Name", is to be used to hold names, then that particular column should have a "**varchar**" (variable-length character) data type.

Here are the most common Data types:

char(size)	Fixed-length character string. Size is specified in parenthesis. Max 255 bytes.
varchar(size)	Variable-length character string. Max size is specified in parenthesis.
number(size)	Number value with a max number of column digits specified in parenthesis.
date	Date value
number(size,d)	Number value with a maximum number of digits of "size" total, with a maximum number of "d" digits to the right of the decimal.

What are constraints?

When tables are created, it is common for one or more columns to have constraints associated with them. A constraint is basically a rule associated with a column that the data entered into that column must follow.

- For example, a "unique" constraint specifies that no two records can have the same value in a particular column. They must all be unique.
- The other two most popular constraints are "not null" which specifies that a column can't be left blank, and "primary key". A "primary key" constraint defines a unique identification of each record (or row) in a table.

All of these and more will be covered in the future Advanced release of this Tutorial. Constraints can be entered in this SQL interpreter, however, they are not supported in this Intro to SQL tutorial & interpreter. They will be covered and supported in the future release of the Advanced SQL tutorial - that is, if "response" is good.

It's now time for you to design and create your own table. You will use this table throughout the rest of the tutorial. If you decide to change or redesign the table, you can either drop it and recreate it or you can create a completely different one. The SQL statement drop will be covered later.

Create Table Exercise

You have just started a new company. It is time to hire some employees. You will need to create a table that will contain the following information about your new employees:

firstname, last name, title, age, and salary.

After you create the table, you should receive a small form on the screen with the appropriate column names. If you are missing any columns, you need to double check your SQL statement and recreate the table. Once it's created successfully, go to the "Insert" lesson.

IMPORTANT: When selecting a table name, it is important to select a unique name that no one else will use or guess. Your table names should have an underscore followed by your initials and the digits of your birth day and month.

For example, Tom Smith, who was born on November 2nd, would name his table myemployees_ts0211 Use this convention for all of the tables you create. Your tables will remain on a shared database until you drop them, or they will be cleaned up if they aren't accessed in 4-5 days

INSERT

The insert statement is used to insert or add a row of data into the table.

To insert records into a table, enter the key words insert into followed by the table name, followed by an open parenthesis, followed by a list of column names separated by commas, followed by a closing parenthesis, followed by the keyword values, followed by the list of values enclosed in parenthesis. The values that you enter will be held in the rows and they will match up with the column names that you specify.

Strings should be enclosed in single quotes, and numbers should not.

```
insert into "tablename"
(first_column,...last_column)
 values (first_value,...last_value);
```

Example:

```
insert into empinfo
 (first, last, id, age, city, state)
 values ('Luke', 'Duke', 45454,
'22', 'Hazard Co', 'Georgia');
```

Note: All strings should be enclosed between single quotes: 'string'

Insert statement exercise

It is time to insert data into your new employee table.

Your first three employees are the following:

Jonie Weber, Secretary, 28, 19500.00

Potsy Weber, Programmer, 32, 45300.00

Dirk Smith, Programmer II, 45, 75020.00

Enter these employees into your table first, and then insert at least 5 more of your own list of employees in the table.

After they're inserted into the table, enter select statements to:

Select all columns for everyone in your employee table.

Select all columns for everyone with a salary over 30000.

Select first and last names for everyone that's under 30 years old.

Select first name, last name, and salary for anyone with "Programmer" in their title.

Select all columns for everyone whose last name contains "ebe".

Select the first name for everyone whose first name equals "Potsy".

Select all columns for everyone over 80 years old.

Select all columns for everyone whose last name ends in "ith".

Create at least 5 of your own select statements based on specific information that you'd like to retrieve.

UPDATE

The update statement is used to update or change records that match a specified criteria. This is accomplished by carefully constructing a where clause.

```
update "tablename"
set "columnname" =
   "newvalue"
[,"nextcolumn" =
  "newvalue2"...]
where "columnname"
 OPERATOR "value"
[and|or "column"
 OPERATOR "value"];
[] = optional
```

Examples:

```
update phone_book
set area_code = 623
where prefix = 979;
```

```
update phone_book
set last_name = 'Smith',
prefix=555, suffix=9292
where last_name = 'Jones';
```

```
update employee
set age = age+1
where first_name='Mary' and
last_name='Williams';
```

Update statement exercises

After each update, issue a select statement to verify your changes.

1. Jonie Weber just got married to Bob Williams. She has requested that her last name be updated to Weber-Williams.
2. Dirk Smith's birthday is today, add 1 to his age.
3. All secretaries are now called "Administrative Assistant". Update all titles accordingly.
4. Everyone that's making under 30000 are to receive a 3500 a year raise.
5. Everyone that's making over 33500 are to receive a 4500 a year raise.
6. All "Programmer II" titles are now promoted to "Programmer III".
7. All "Programmer" titles are now promoted to "Programmer II".

Create at least 5 of your own update statements and submit them.

DELETING RECORDS

The delete statement is used to delete records or rows from the table.

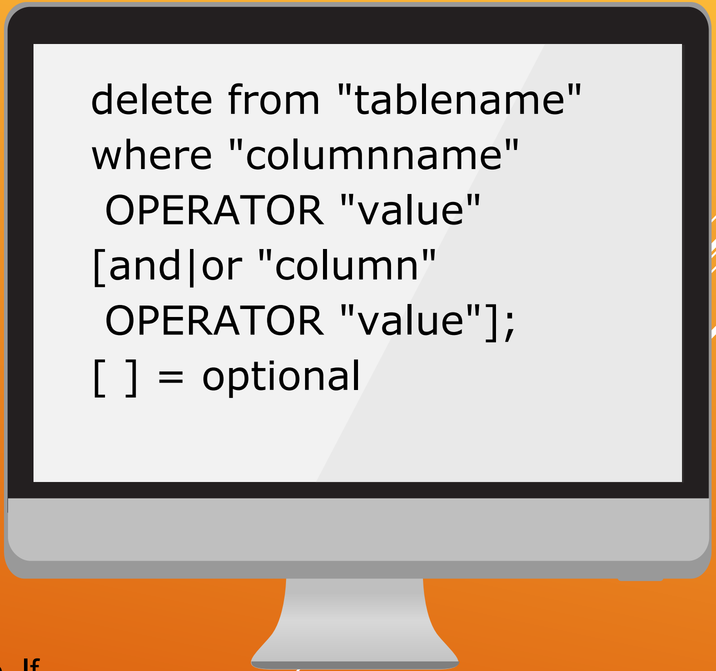
Examples:

```
delete from employee;
```

Note: if you leave off the where clause, **all records will be deleted!**

```
delete from employee
 where lastname = 'May';

delete from employee
 where firstname = 'Mike' or firstname = 'Eric';
```

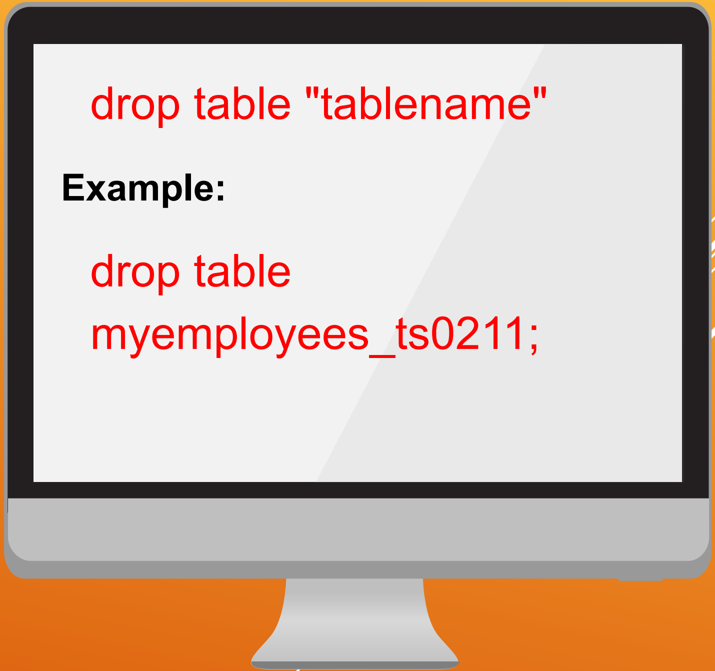


```
delete from "tablename"
where "columnname"
 OPERATOR "value"
[and|or "column"
 OPERATOR "value"];
[ ] = optional
```

To delete an entire record/row from a table, enter "delete from" followed by the table name, followed by the where clause which contains the conditions to delete. If you leave off the where clause, all records will be deleted.

DROP A TABLE

- The **drop table** command is used to delete a table and all rows in the table.
- To delete an entire table including all of its rows, issue the **drop table** command followed by the tablename.
- **drop table** is different from deleting all of the records in the table.
- Deleting all of the records in the table leaves the table including column and constraint information.
- Dropping the table removes the table definition as well as all of its rows.



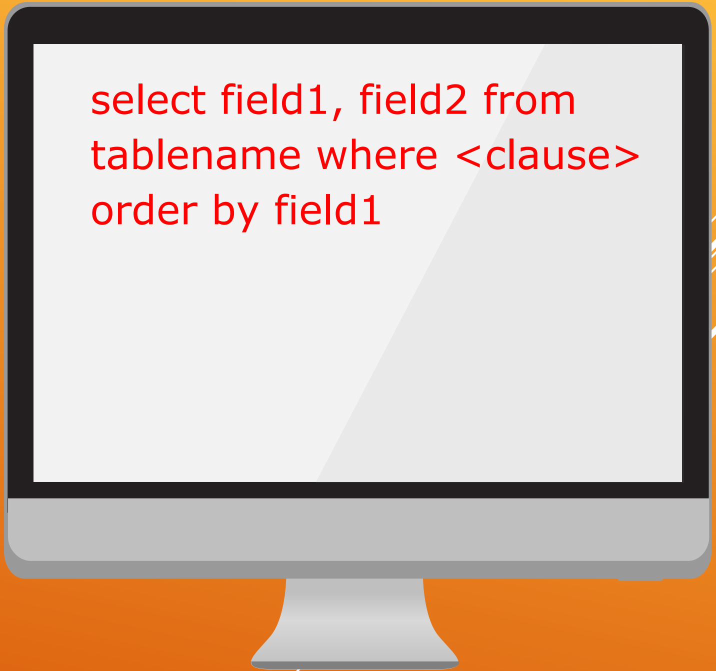
```
drop table "tablename"
```

Example:

```
drop table
myemployees_ts0211;
```

JOINS

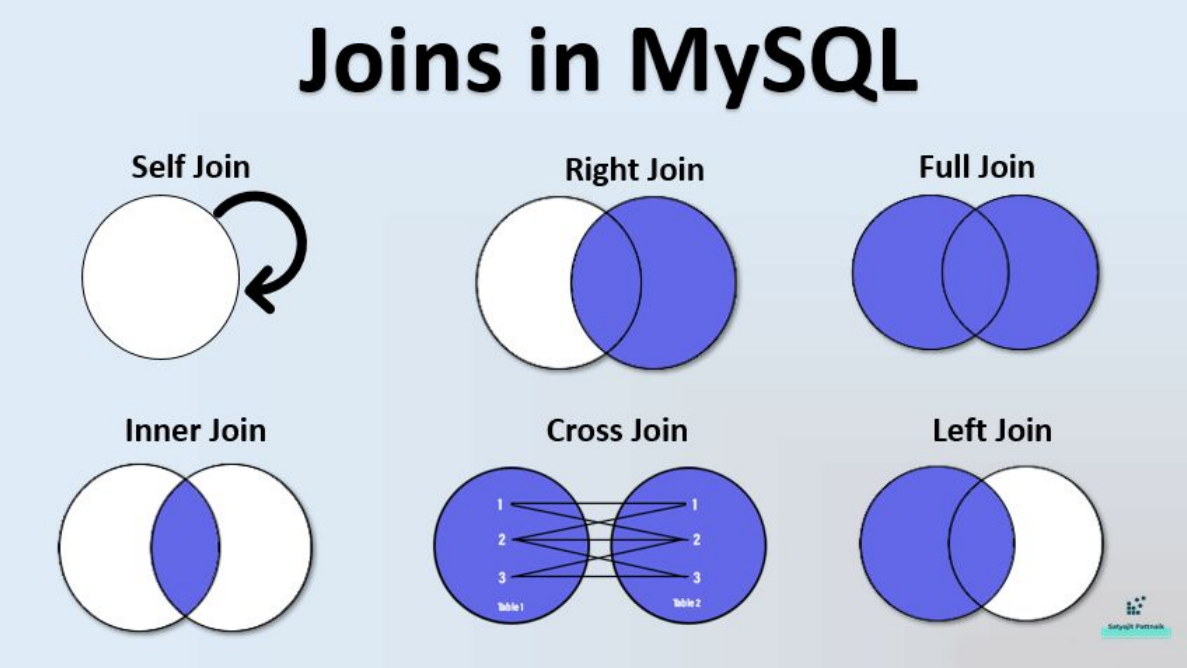
- In order to sort a table, we need to use the **ORDER BY** Clause
- In order to get the results sorted based on certain columns, we need to use this. Usage of **asc** or **desc** has to be defined to have the results in ascending or descending orders. Default value being **asc** (ascending)



```
select field1, field2 from
tablename where <clause>
order by field1
```

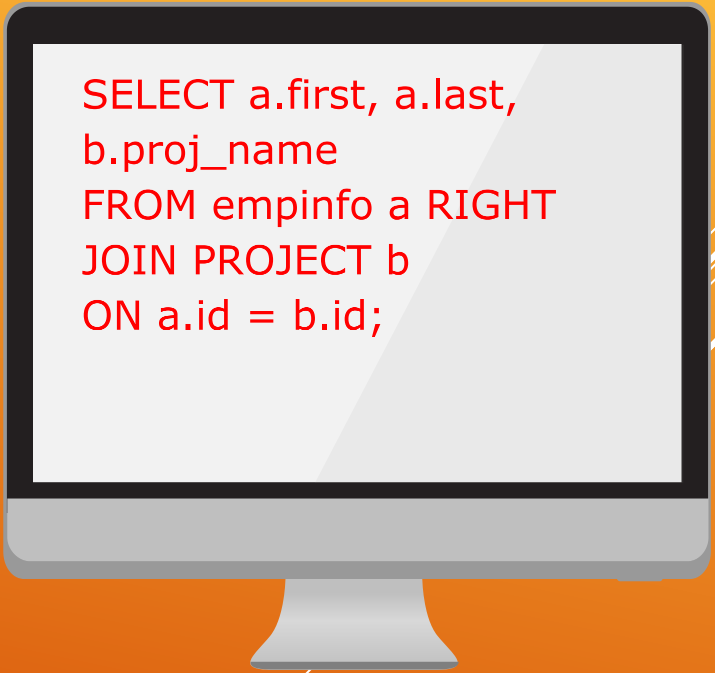
What are constraints?

- In most of the real world problems, we might need data from multiple tables, that’s where **Joins** comes into picture.



JOINS - Right Join

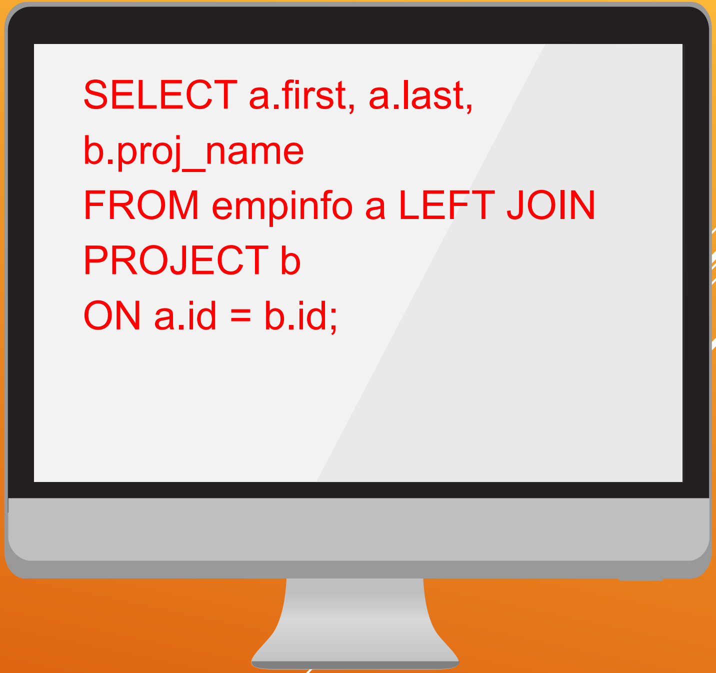
- The RIGHT JOIN keyword returns all records from the right table (table2), and the matched records from the left table (table1).



```
SELECT a.first, a.last,
b.proj_name
FROM empinfo a RIGHT
JOIN PROJECT b
ON a.id = b.id;
```

JOINS - Left Join

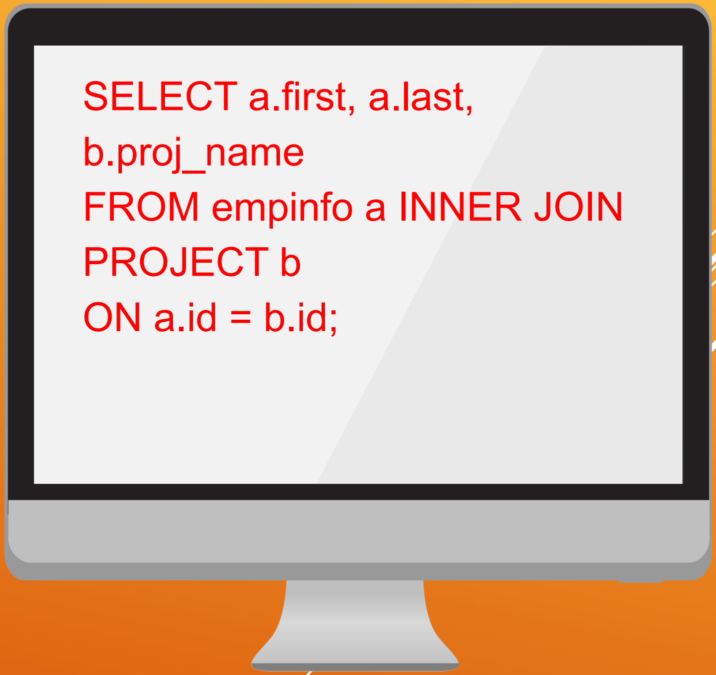
- The LEFT JOIN keyword returns all records from the left table (table1), and the matched records from the right table (table2). The result is NULL from the right side, if there is no match.



```
SELECT a.first, a.last,
b.proj_name
FROM empinfo a LEFT JOIN
PROJECT b
ON a.id = b.id;
```

JOINS - Inner Join

- The INNER Join keyword selects records that have matching values in both tables

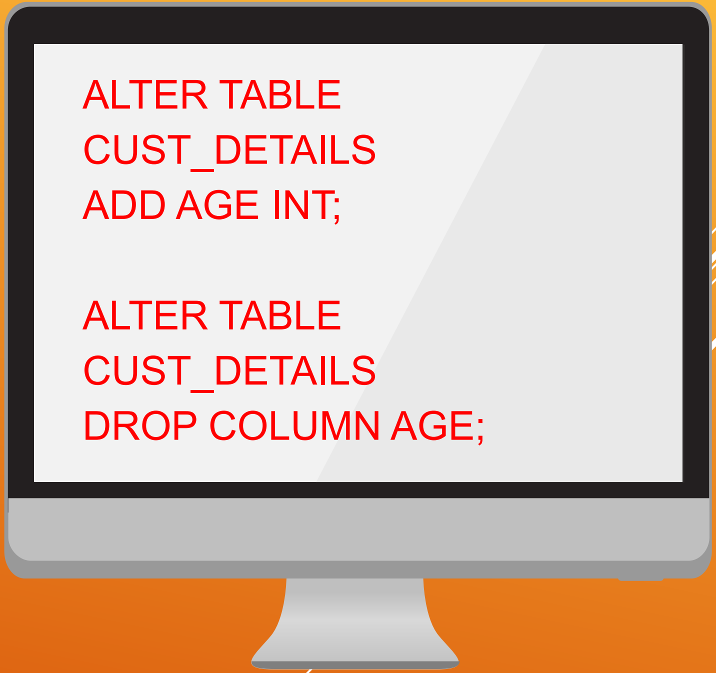


```
SELECT a.first, a.last,
b.proj_name
FROM empinfo a INNER JOIN
PROJECT b
ON a.id = b.id;
```

Alter Table

```
ALTER TABLE
tablename
ADD columnname
datatype;

ALTER TABLE
tablename
DROP COLUMN
columnname ;
```

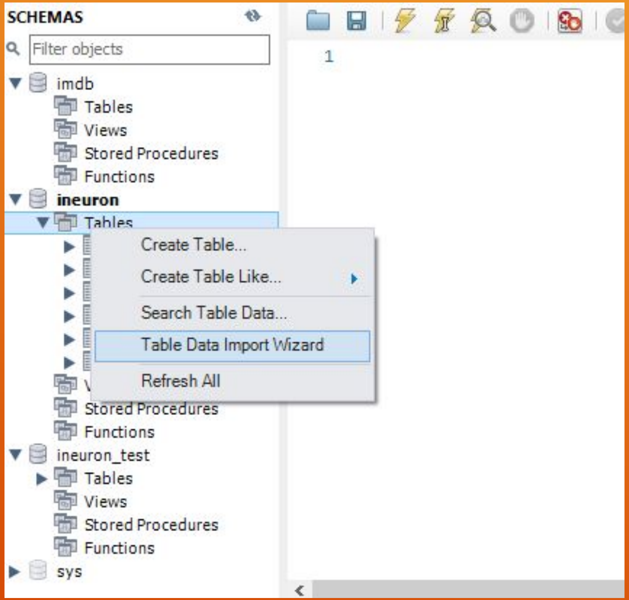


```
ALTER TABLE
CUST_DETAILS
ADD AGE INT;

ALTER TABLE
CUST_DETAILS
DROP COLUMN AGE;
```

Data Importing

1. Manual Importing



Data Importing

1. Importing through command line

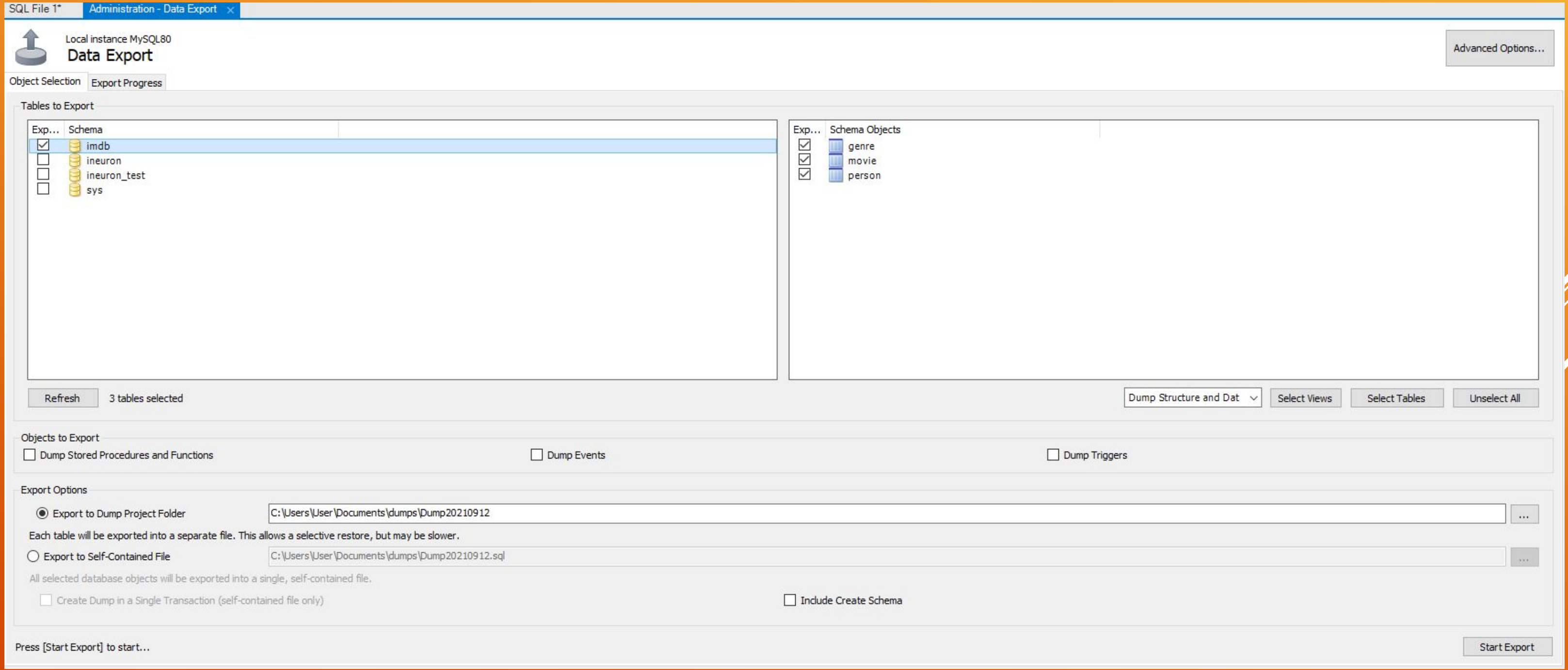
- Open MySQL Workbench, Create a new database to store the tables you'll import (eg- FacilitySerivces) → Then create the table using CREATE query
- Copy the MySQL bin directory path: C:\Program Files\MySQL\MySQL Server 8.0\bin
- Go to the folder in command line by using: cd path
- Connect to MySQL database: mysql -u root -p (root is basically your username)
- If you are logged in successfully, then set the global variables by using below command so that the data can be imported from local computer folder.
    - mysql> SET GLOBAL local_infile = 1;
    - Query OK, 0 rows affected (0.00 sec)
    - (you've just instructed MySQL server to allow local file upload from your computer)
- Quit current server connection (mysql> quit)
- Load the file from CSV file to the MySQL database. In order to do this, please follow the commands: (We'll connect with the MySQL server again with the local-infile system variable. This basically means you want to upload data into a file from a local machine)
    - mysql --local-infile=1 -u root -p (give password)
    - Show Databases; (It'll show all the databases in MySQL server.)
    - mysql> USE dbase; (makes the database that you had created in step 1 as default schema to use for the next sql scripts)

    ```
    mysql> LOAD DATA LOCAL INFILE 'fullpath\\file.csv'
    INTO TABLE tablename
    FIELDS TERMINATED BY ','
    ENCLOSED BY '"'
    LINES TERMINATED BY '\r\n' IGNORE 1 ROWS;
    ```

- **Note: VERY IMP** - Please replace single backward (\) slash in the path with double back slashes (\\) instead of single slash

Data Exporting

Server → Data Export



Local instance MySQL80

Data Export

Tables to Export

Exp...	Schema
☑	imdb
□	ineuron
□	ineuron_test
□	sys

Exp...	Schema Objects
☑	genre
☑	movie
☑	person

3 tables selected

Dump Structure and Dat

Objects to Export

□ Dump Stored Procedures and Functions □ Dump Events □ Dump Triggers

Export Options

● Export to Dump Project Folder C:\Users\User\Documents\dumps\Dump20210912

Each table will be exported into a separate file. This allows a selective restore, but may be slower.

○ Export to Self-Contained File C:\Users\User\Documents\dumps\Dump20210912.sql

All selected database objects will be exported into a single, self-contained file.

□ Create Dump in a Single Transaction (self-contained file only) □ Include Create Schema

Press [Start Export] to start...

Aggregate Functions

✔ Sometimes we examine & analyse data of varying magnitudes, hence we realise the need of grouping similar types of values together & look them at as one bunch.

For example: Consider a table containing data consisting of the marks scored by students in their 12th board exams. While you would want to know how the students performed in all the subjects put together, it is equally important to see how they performed in each subject. You can gain even further insights if you group these students by state. Hence, it is imperative that you learn the usage of aggregate functions in your queries.

✔ groupby() → To aggregate values of a column C1 ‘grouped by’ a certain column C2.

✔ count() → Count the number of rows.

✔ min(), max()→ Finding the minimum & maximum values for a particular column

For example: select min(score) from employee;

✔ avg() → Find the average

ORDERING

Quite often, you would want to display the retrieved records in a particular order, for example, in increasing order of income, joining date, alphabetical order, etc. This is commonly useful when you are making a report or presenting the data to someone else.

```
select firstName from
employees order by
firstName asc limit 3;
```

HAVING CLAUSE

You have already learnt how to filter individual values based on a given condition.
But how do you do this on grouped values?

Suppose your manager asks you to count all the employees whose salaries are more than the average salary in that particular department.

Now, intuitively, you know that two aggregate functions would be used here, namely, count() and avg(). You decide to apply the 'where' condition on the average salary of the department, but to your surprise, the query fails. This is exactly what the having clause is for.

The 'having' clause is typically used when you have to apply a filter condition on an 'aggregated value'. This is because the 'where' clause is applied before aggregation takes place, and thus, it is not useful when you want to apply a filter on an aggregated value.

The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions.

Exercise

Keywords in SQL

List the keywords below in order of their occurrence in a query:

1. group by
2. order by
3. select
4. where
5. from
6. limit
7. having

Answer is:

STRING FUNCTIONS

Used to manipulate the string data and make it more understandable for analysis.

For example: amitabhbachchan, or Amitabh Bachchan, which one of them is more readable, obviously the later one right.

concat → Concatenates two strings
substr/substring → Extracts a substring from a string
upper → Converts a string to uppercase
lower → Converts a string to lowercase
character_length → Calculates the length of the string variable.
mid → find the middle elements → MID(str,pos,len)

https://www.w3schools.com/sql/sql_ref_mysql.asp

DATE & TIME FUNCTIONS

Used to manipulate the date & time columns

For example: If you want to change the date format, or just want to see the exact day, or so on.

Let’s create a transaction_details table and play around.

datediff → Return the number of days between the two date values
SELECT DATEDIFF(sysdate(), order_date) from transaction_details
date_format → Format a date variable
SELECT DATE_FORMAT("2017-06-15", "%Y");
day → Return the day of the month for a date.
SELECT DAY("2017-06-15");
quarter → Return the quarter of the year for a date.
SELECT QUARTER("2017-06-15");
addday → Add days to the date variable
SELECT ADDDATE("2017-06-15", INTERVAL 10 DAY);

and so on….
https://www.w3schools.com/sql/func_mysql_adddate.asp

REGEX

So far you have already known about the wildcards like “like” operator, but in cases wildcards may fall short for some advanced use cases, regular expressions comes into picture.

Regex, or Regular expressions, is a sequence of characters, used to search and locate specific sequences of characters that match a pattern.

Example 1: Match beginning of the name

```
select * from customer WHERE email REGEXP '^w'
```

REGEX

Example 2: Find the customers, which email address having a ‘z’, ‘v’ or ‘p’

```
SELECT * FROM customer
WHERE email REGEXP "[zvp]";
```

Example 3: Find the customers, which email address containing characters from ‘x’ to ‘z’

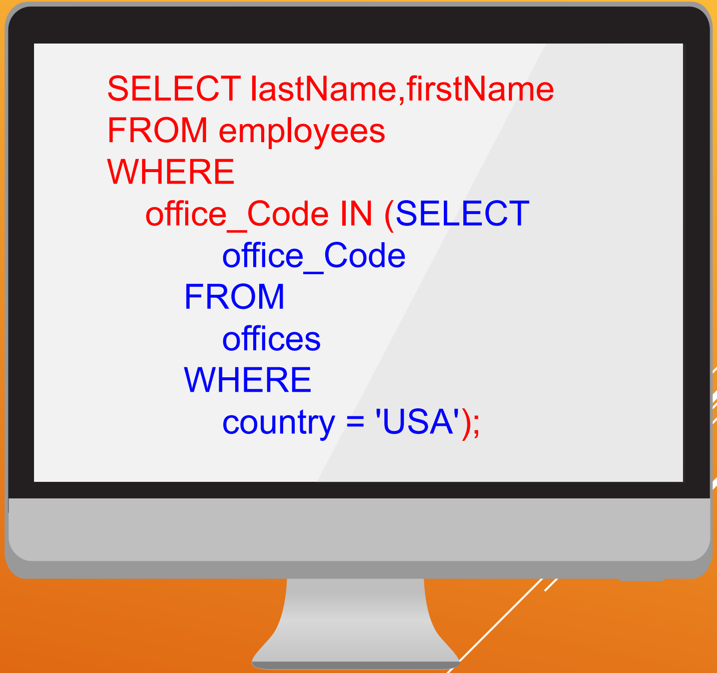
```
SELECT * FROM customer
WHERE email REGEXP "[x-z]";
```

NESTED QUERIES

By now, you know that a database is a collection of multiple related tables. Now, while generating insights from data, you may need to refer to these multiple tables in a query. There are two ways to deal with such types of queries. These include the following:

1. Joins
2. Nested queries/Subqueries

A subquery is called an inner query while the query that contains the subquery is called an outer query. A subquery can be used anywhere that expression is used and must be closed in parentheses.



```
SELECT lastName,firstName
FROM employees
WHERE
    office_Code IN (SELECT
            office_Code
        FROM
            offices
        WHERE
            country = 'USA');
```

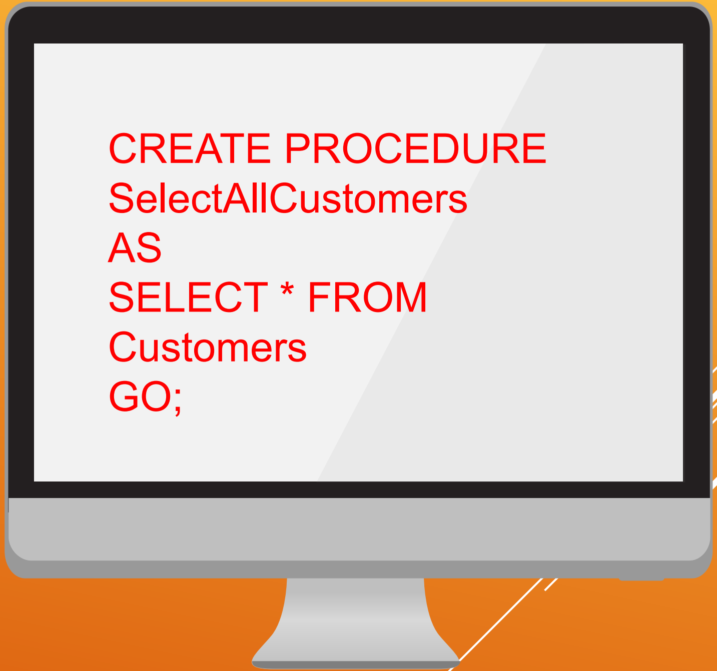
STORED PROCEDURES

A stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again.

So if you have an SQL query that you write over and over again, save it as a stored procedure, and then just call it to execute it.

You can also pass parameters to a stored procedure, so that the stored procedure can act based on the parameter value(s) that is passed.

.

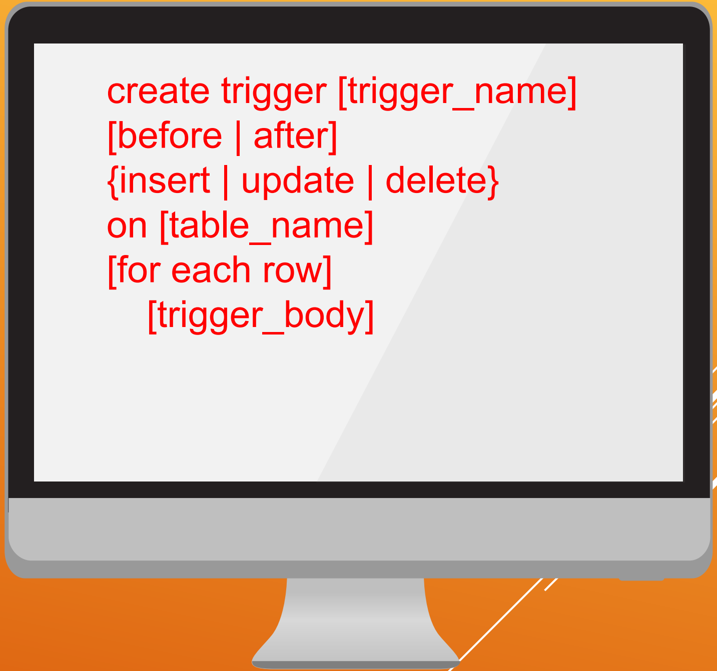


```
CREATE PROCEDURE
SelectAllCustomers
AS
SELECT * FROM
Customers
GO;
```

TRIGGERS

A trigger is a stored procedure in database which automatically invokes whenever a special event in the database occurs. For example, a trigger can be invoked when a row is inserted into a specified table or when certain table columns are being updated.

- DDL Trigger
- DML Trigger
- Logon Trigger



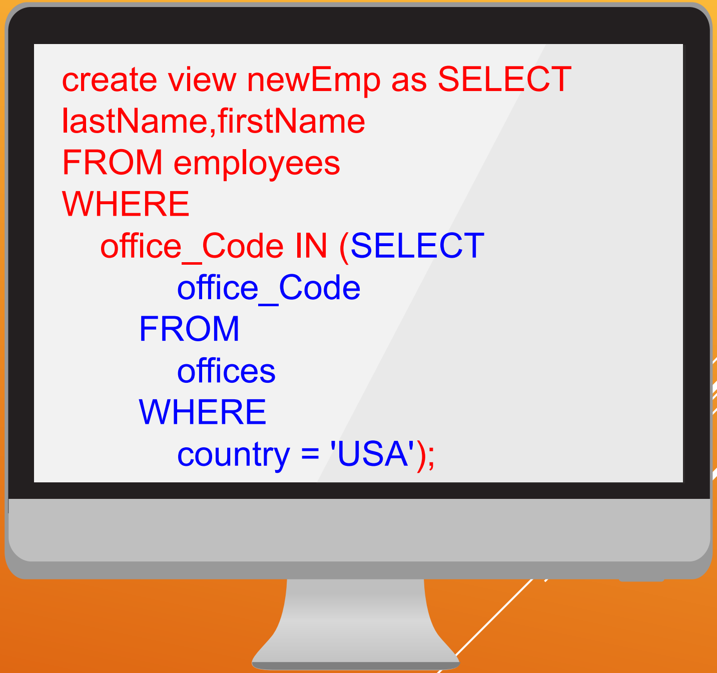
```
create trigger [trigger_name]
[before | after]
{insert | update | delete}
on [table_name]
[for each row]
    [trigger_body]
```

VIEWS

Views are virtual tables that do not store any data of their own but display data stored in other tables.

Advantages:

- Hide the complexity of data.
- Act as aggregated tables.
- If you are doing an user level access control, you can give an user access to a view without giving access to the tables behind it
- It can allow for massive performance improvements.

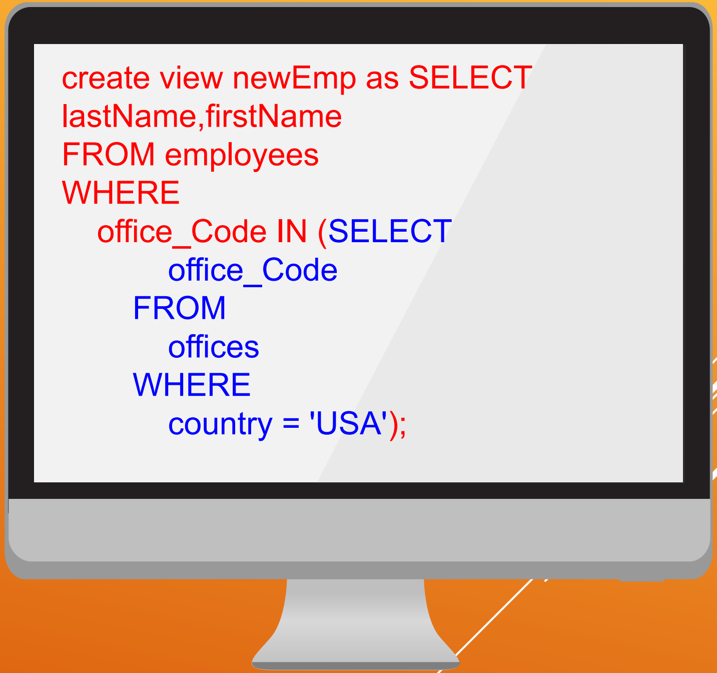


```
create view newEmp as SELECT
lastName,firstName
FROM employees
WHERE
   office_Code IN (SELECT
         office_Code
      FROM
         offices
      WHERE
         country = 'USA');
```

VIEWS

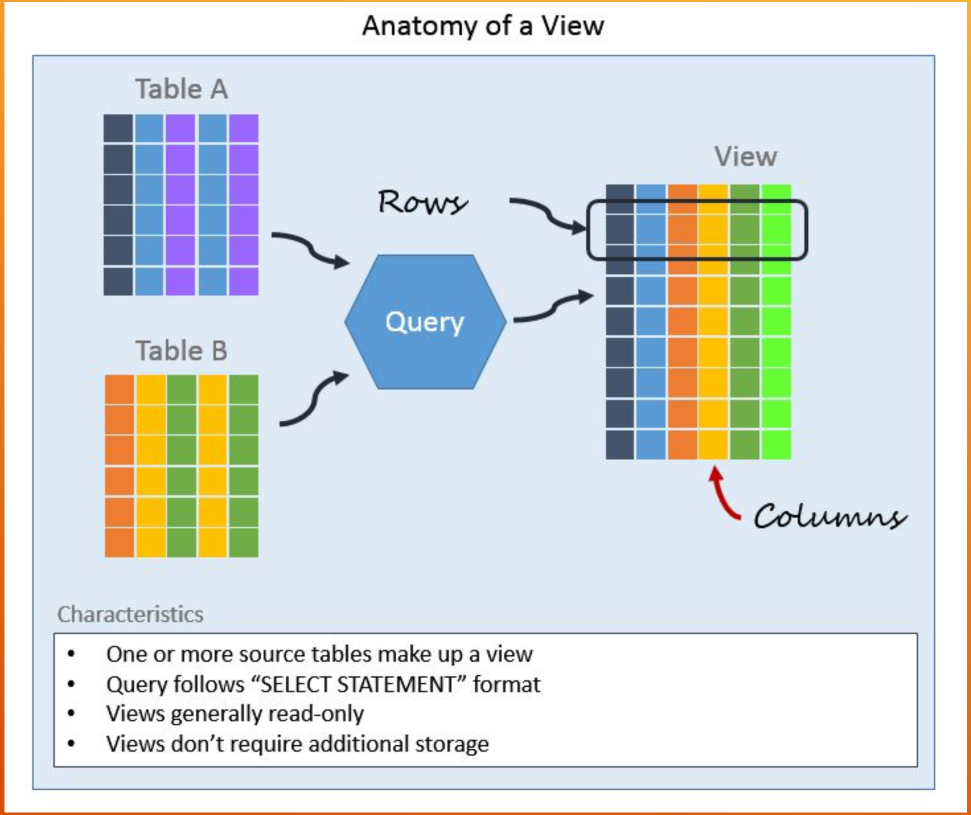
Disadvantages:

- If done wrong, it can result in performance issues.
- You may not be able to update the view, forcing you back to the original tables.



```
create view newEmp as SELECT
lastName,firstName
FROM employees
WHERE
   office_Code IN (SELECT
         office_Code
      FROM
         offices
      WHERE
         country = 'USA');
```

VIEWS



Anatomy of a View

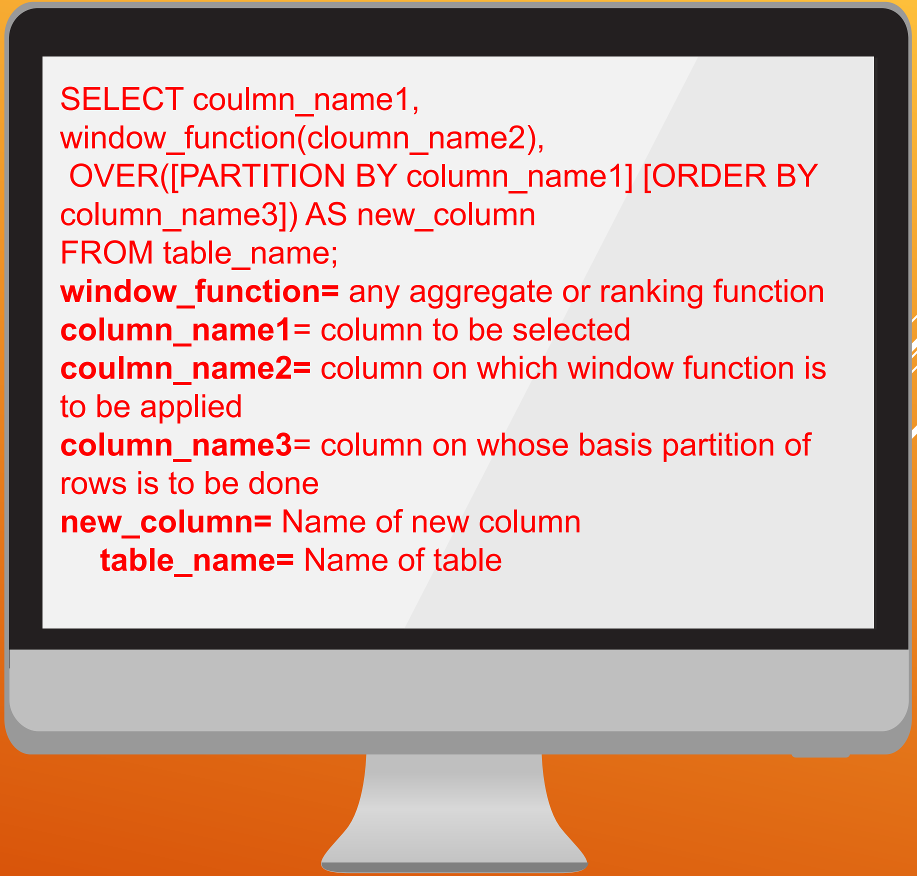
Characteristics

- One or more source tables make up a view
- Query follows “SELECT STATEMENT” format
- Views generally read-only
- Views don’t require additional storage

WINDOW FUNCTION

Window functions applies aggregate and ranking functions over a particular window (set of rows). OVER clause is used with window functions to define that window. OVER clause does two things :

- Partitions rows into form set of rows. (PARTITION BY clause is used)
- Orders rows within those partitions into a particular order. (ORDER BY clause is used)



```
SELECT coulmn_name1,
window_function(cloumn_name2),
 OVER([PARTITION BY column_name1] [ORDER BY
column_name3]) AS new_column
FROM table_name;
```

window_function= any aggregate or ranking function
column_name1= column to be selected
coulmn_name2= column on which window function is to be applied
column_name3= column on whose basis partition of rows is to be done
new_column= Name of new column
table_name= Name of table

WINDOW FUNCTION

- Row_number() → Gives a sequential integer to every row within its partition
- Rank() functions → Ranking records
- First_value() functions → Returns the value of the specific expression with respect to first row in the window frame

SQL-Python Connectivity

- pymysql → This package contains a pure-Python MySQL client library, based on PEP 249

  → pip install pymysql

```
dbcon = pymysql.connect(host="localhost",user= "root",password= "root",database= "breast_cancer")
```

SUMMARY

You learnt about the crux of SQL queries. These are the basic to intermediate level concepts of SQL that will be tested rigorously in your interviews. Hence, practice these points again and again to gain a deep understanding of the several SQL statements.

Leave your top three takeaways from this session in the comments section below.

Exercise

Write a query to retrieve the names of all employees who have an age greater than what Gus Gray has.

Answer is:

Exercise

Write a query to retrieve the names of all employees who have an age greater than what Gus Gray has, and store the output in a view.

Answer is:

Exercise

Write a query to retrieve the names of all employees who have an age greater than what Gus Gray has, and store the output in a view.

Answer is: