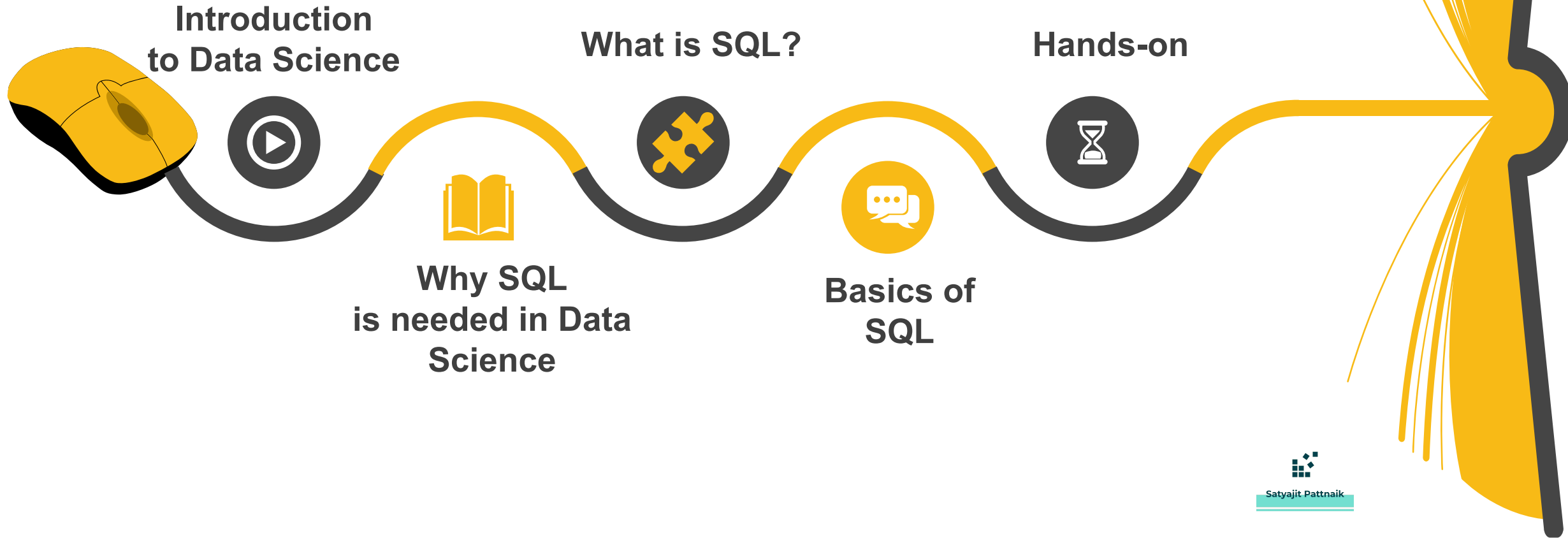




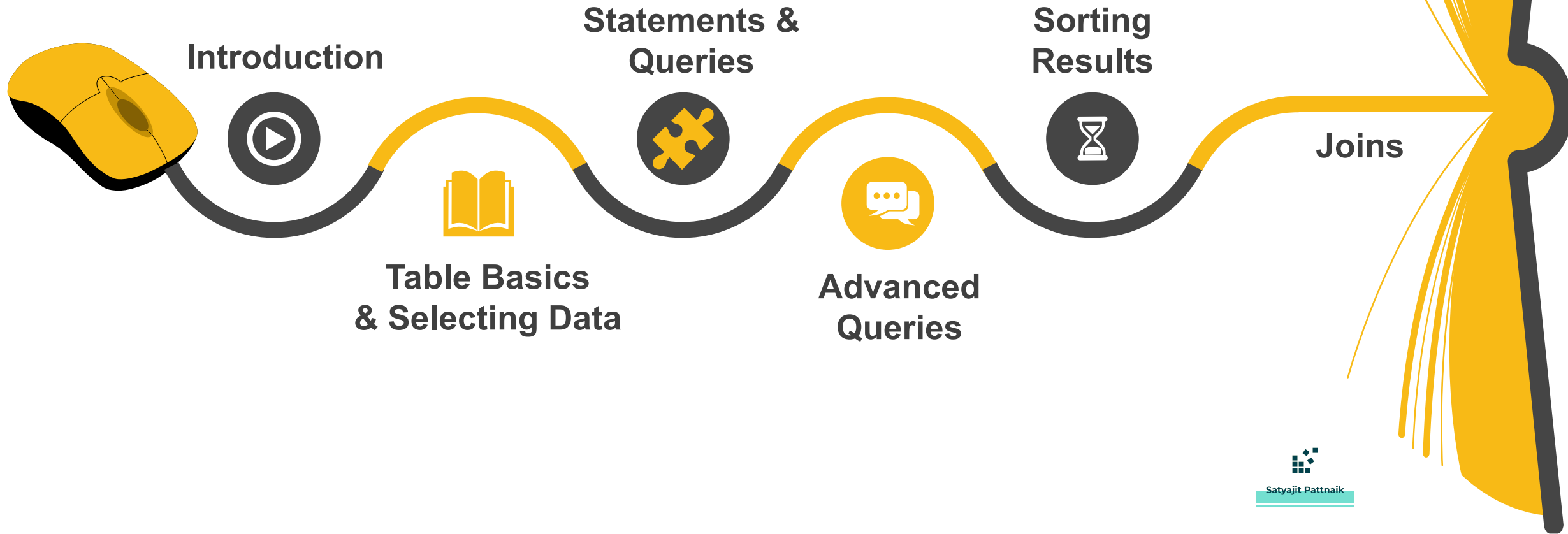
SQL FOR

DATA SCIENCE

Overall Timeline



SQL Timeline



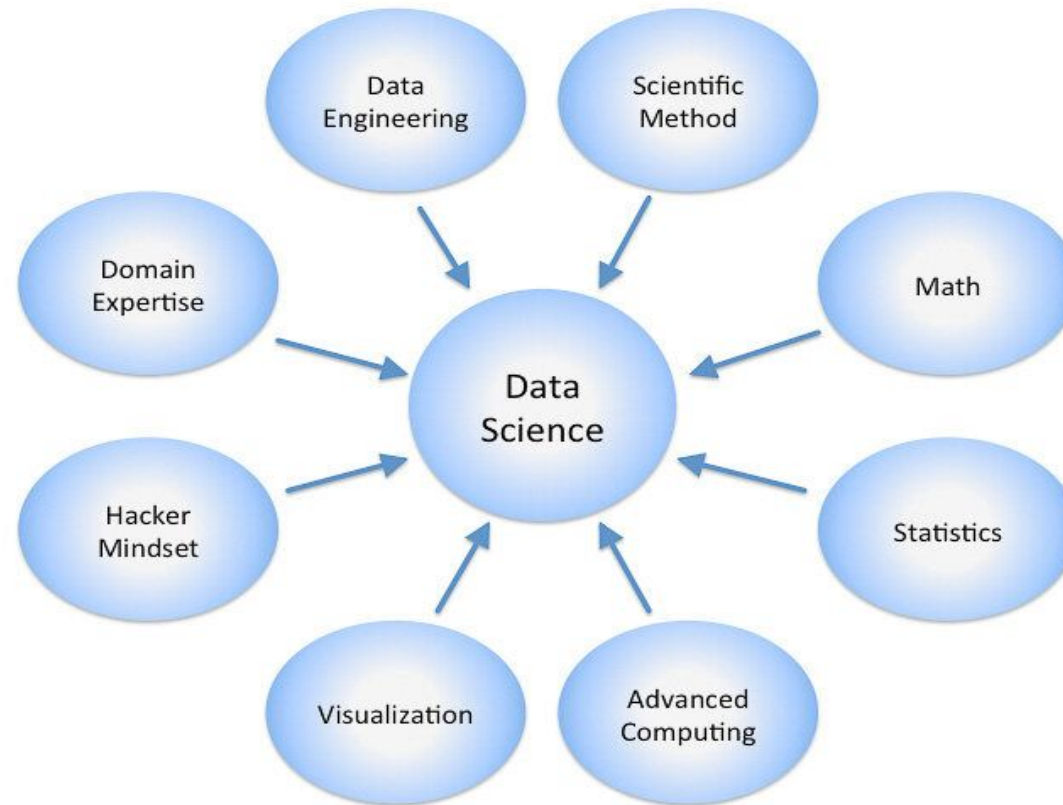


Introduction to Data Science



Satyajit Pattnaik

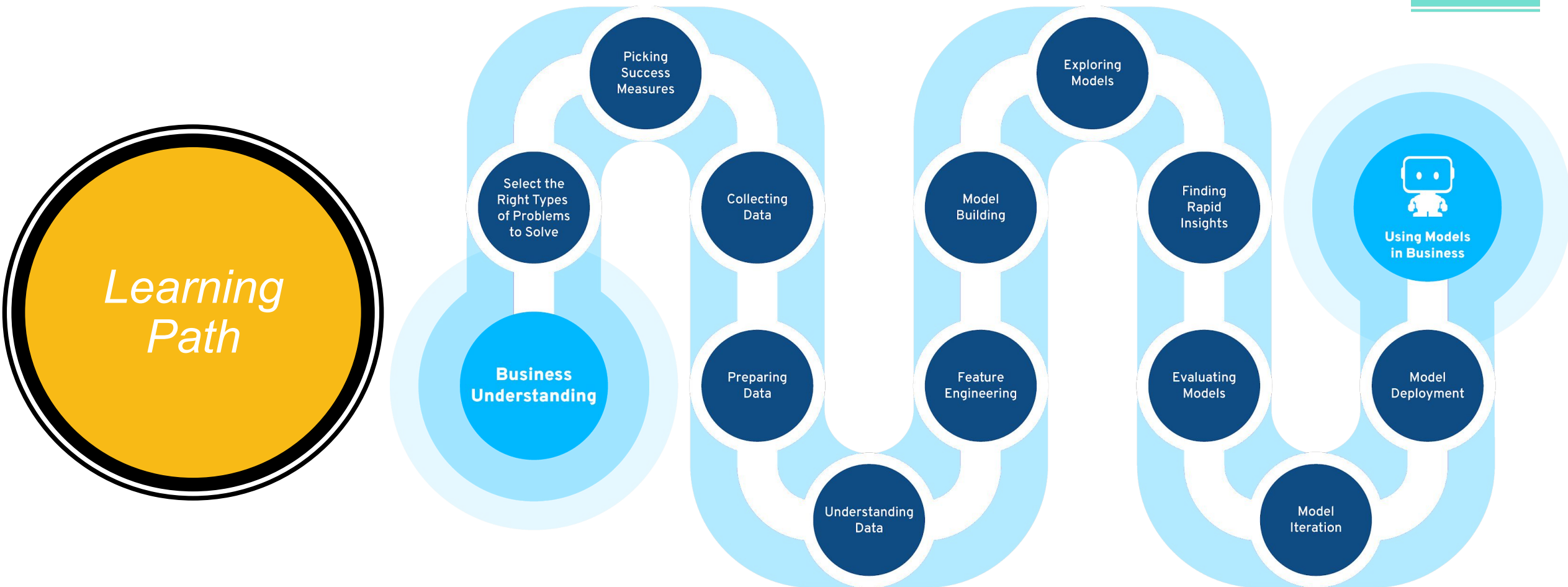
What is Data Science?



Learning Path for a Data Scientist



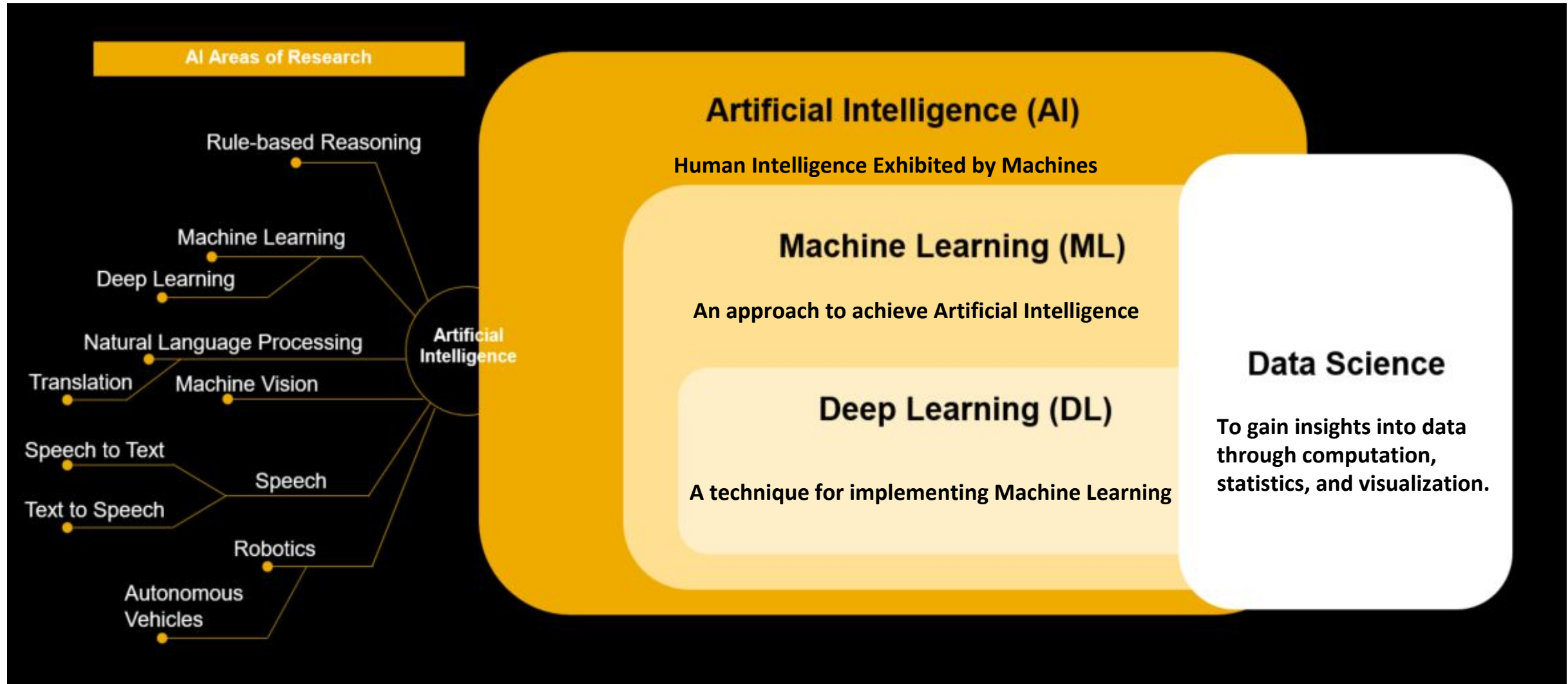
Satyajit Pattnaik



Biggest Confusion: AI vs ML vs DL vs DS



Satyajit Pattnaik



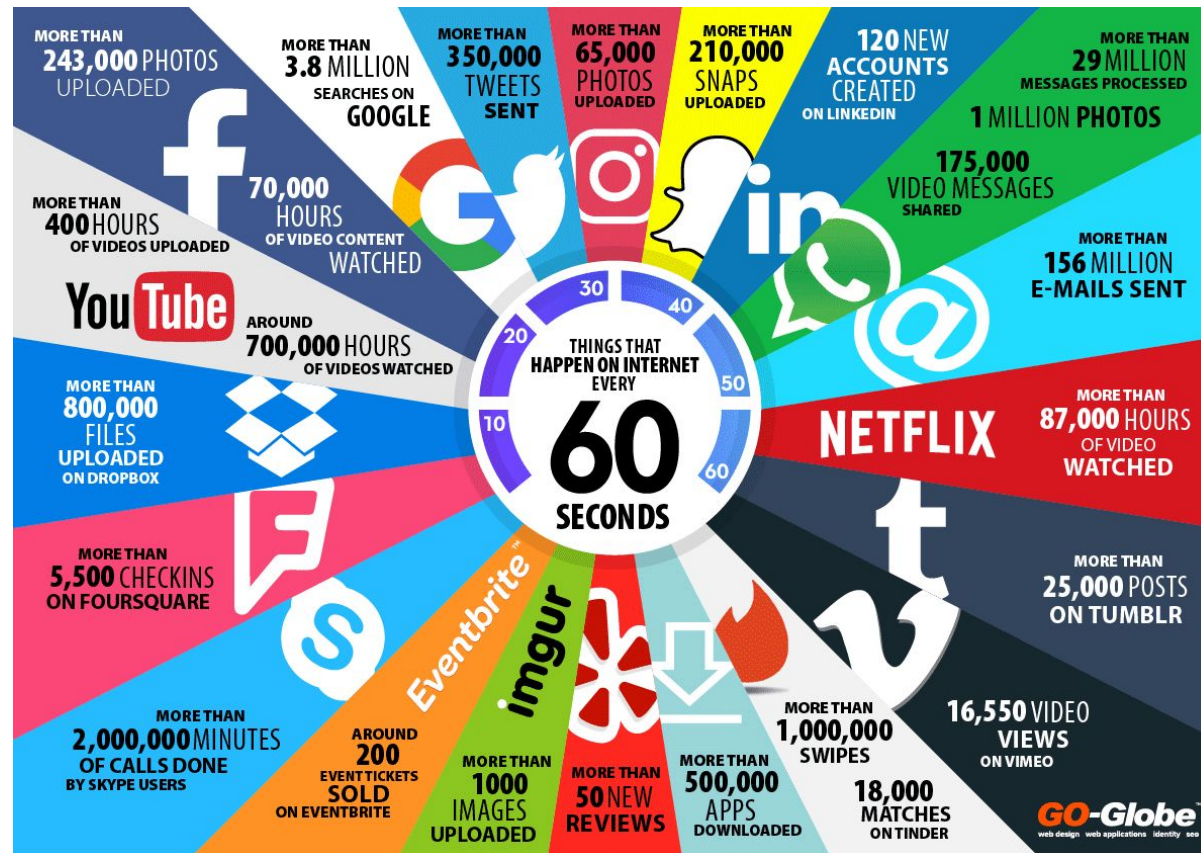
Overview: What is Machine Learning?



Satyajit Pattnaik

Machine learning is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed.

Machine learning focuses on the development of computer programs that can teach themselves to grow and change when exposed to new data.



Why do we need Machine Learning?



Satyajit Pattnaik

- Volume of data collected growing day by day.
- Data production is 44 times greater in 2020 than in 2009.
- Every day, 2.5 quintillion bytes of data are created
- Data is nearly doubling in size every two years.
- Knowledge Discovery is needed to make sense and use of data.
- Machine Learning is a technique in which computers learn from data to obtain insight and help in knowledge discovery

APPLICATION OF MACHINE LEARNING



Satyajit Pattnaik



Recommendation
Engine



Face
Recognition



Maps



Translate



Siri

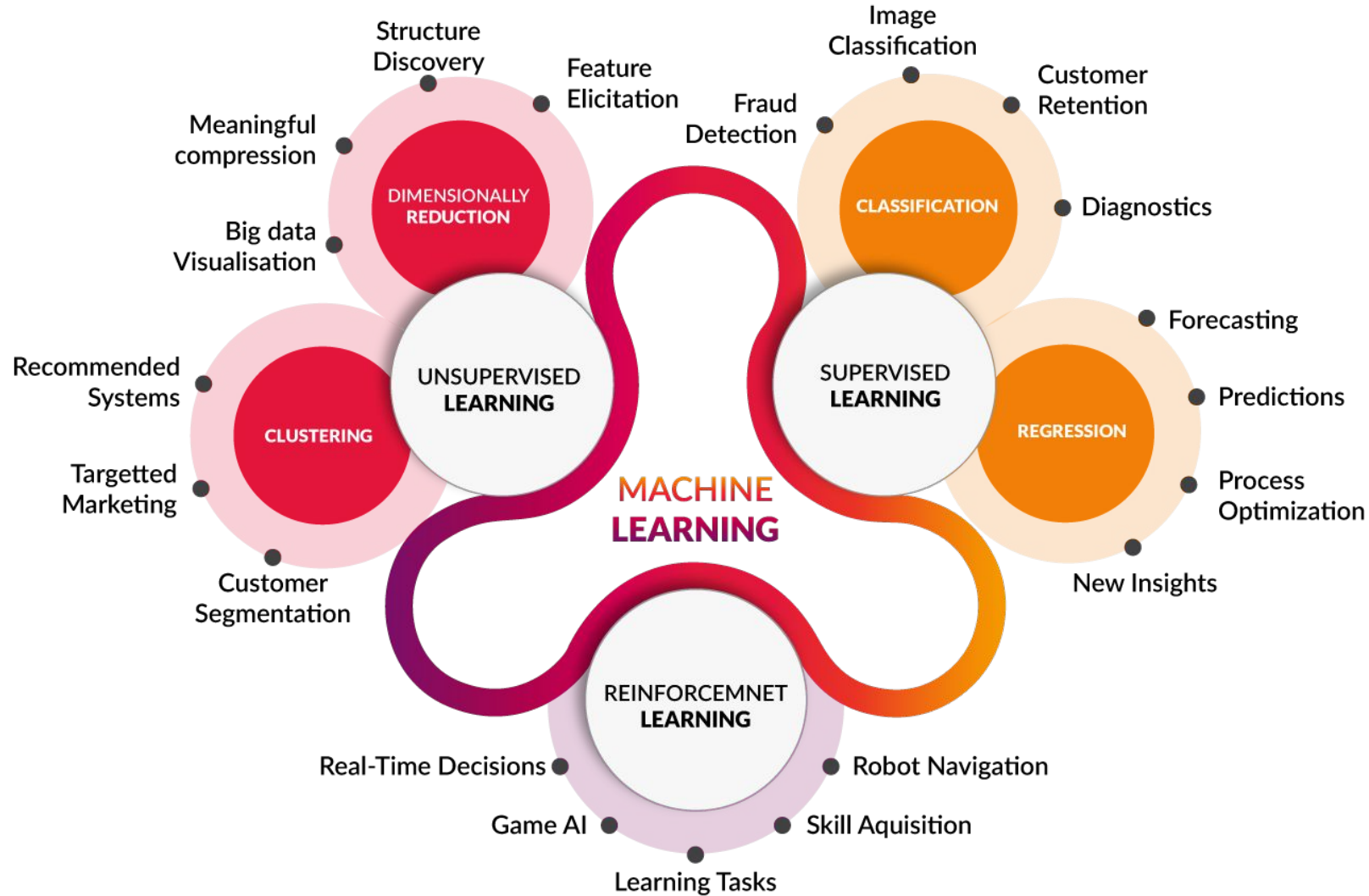
NETFLIX

Recommendation
Engine

Machine Learning - Types & Applications



Satyajit Pattnaik



TOP 12 INTERESTING CAREERS TO EXPLORE IN DATA



DATA SCIENTIST

These people use their analytical and technical capabilities to extract meaningful insights from data.



DATA ENGINEER

They ensure uninterrupted flow of data between servers and applications and are also responsible for data architecture.



BIG DATA ENGINEER

Big Data Engineers build the designs created by solutions architects. They develop, maintain, test and evaluate big data solutions within organisations.



MACHINE LEARNING SCIENTIST

They work in the research and development of algorithms that are used in adaptive systems. They build methods for predicting product suggestions and demand forecasting, and explore Big Data to automatically extract patterns.



BUSINESS ANALYTICS SPECIALIST

A business analytics specialist supports various development initiatives, assists in testing activities and in the development of test scripts, performing research in order to understand business issues, and developing practical cost-effective solutions to problems.



DATA VISUALIZATION DEVELOPER

They design, develop and provide production support of interactive data visualizations used across the enterprise. They possess an artistic mind that conceptualizes, design, and develop reusable graphic/data visualizations and uses strong technical knowledge for implementing these visualizations using the latest technologies.



BUSINESS INTELLIGENCE (BI) ENGINEER

They have data analysis expertise and the experience of setting up reporting tools, querying and maintaining data warehouses. They are hands-on with big data and take a data driven approach to solving complex problems.



BI SOLUTION ARCHITECT

They come up with solutions quickly to help businesses in making time sensitive decisions, have strong communication & analytical skills, passion for data visualization, and a drive for excellence and self motivation.



BI SPECIALIST

They are responsible for supporting an enterprise wide business intelligence framework. This position requires critical thinking, attention to detail, and effective communication skills



ANALYTICS MANAGER

An analytics manager is responsible for configuration, design, implementation, and support of data analysis solution or BI tool. They are specifically required to analyze huge quantities of information gathered through transactional activity.



MACHINE LEARNING ENGINEER

Machine Learning engineer's final "output" is the working software, and their "audience" for this output consists of other software components that run autonomously with minimal human supervision. The decisions are made by machines and they affect how a product or service behaves.



STATISTICIAN

They gather numerical data and then display it, and help companies to make sense of quantitative data and to spot trends and make predictions.



Satyajit Pattnaik



Introduction



Satyajit Pattanaik

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.

TABLE BASICS



Satyajit Pattnaik

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

SELECTING THE DATA



Satyajit Pattnaik

Select statement is used to query the database and retrieve selected data that matches the criteria.

Here's a format of a select query:

```
select column1, column2 from tablename where [condition];  
-- where condition is optional
```



The **where** clause (optional) specifies which data values or rows will be returned or displayed, based on the criteria described after the keywords **where**:

- = equal
- > greater than
- < lesser than
- >= greater than or equal
- <= lesser than or equal
- <> not equal
- LIKE like operator



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';
```





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';
```



LEARNING

CREATING TABLES



Satyajit Pattnaik

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



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 ";"**.

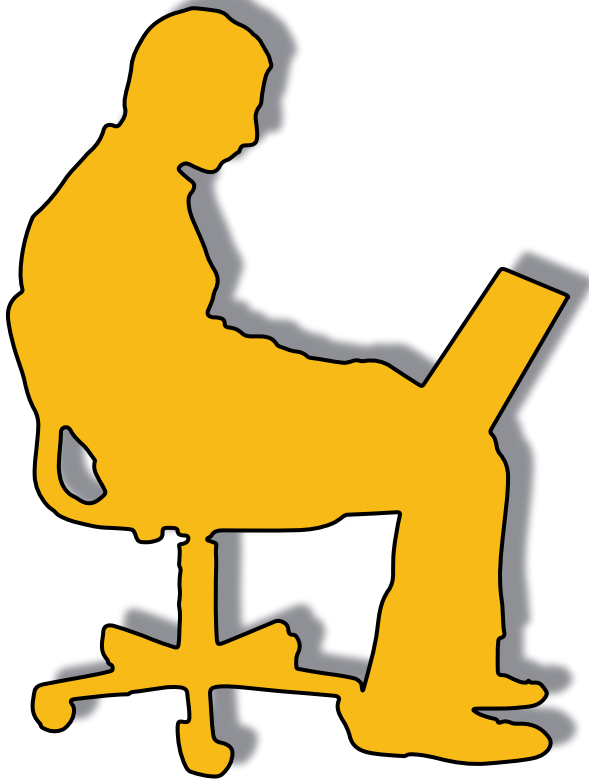


The table and column names;

- 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:



Satyajit Pattnaik

<code>char(size)</code>	Fixed-length character string. Size is specified in parenthesis. Max 255 bytes.
<code>varchar(size)</code>	Variable-length character string. Max size is specified in parenthesis.
<code>number(size)</code>	Number value with a max number of column digits specified in parenthesis.
<code>date</code>	Date value
<code>number(size,d)</code>	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



Satyajit Pattnaik

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.

In the example here, the column name **first** will match up with the value **'Luke'**, and the column name **state** will match up with the value **'Georgia'**.

```
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 exercises

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.

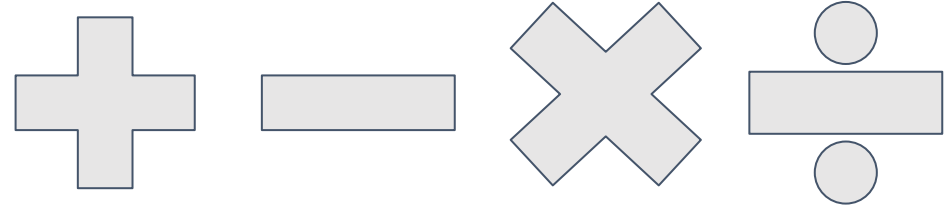
Select all columns for everyone whose last name contains "ebe".

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

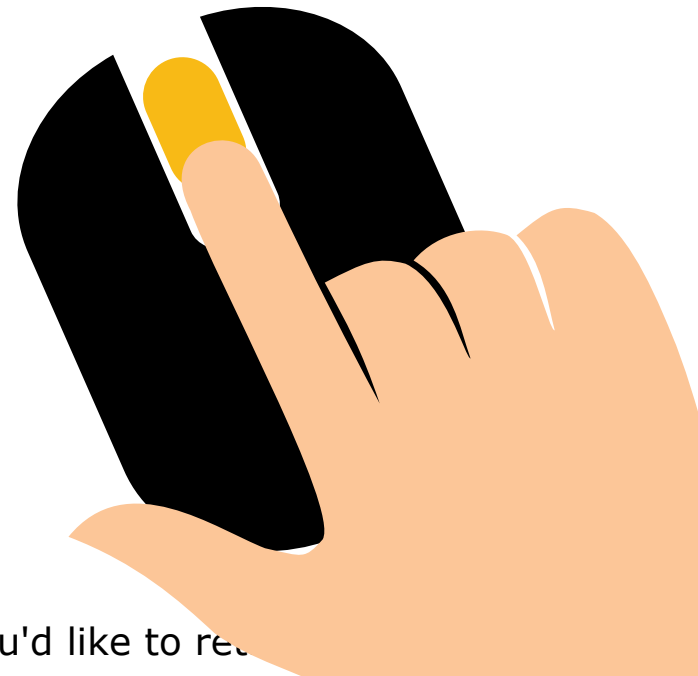
Select all columns for everyone over 80 years old.

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 re-



Satyajit Pattnaik



UPDATE



Satyajit Pattnaik

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';
```

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.

```
delete from "tablename"
```

```
where "columnname"  
OPERATOR "value"  
[and|or "column"  
OPERATOR "value"];
```

```
[ ] = optional
```


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:

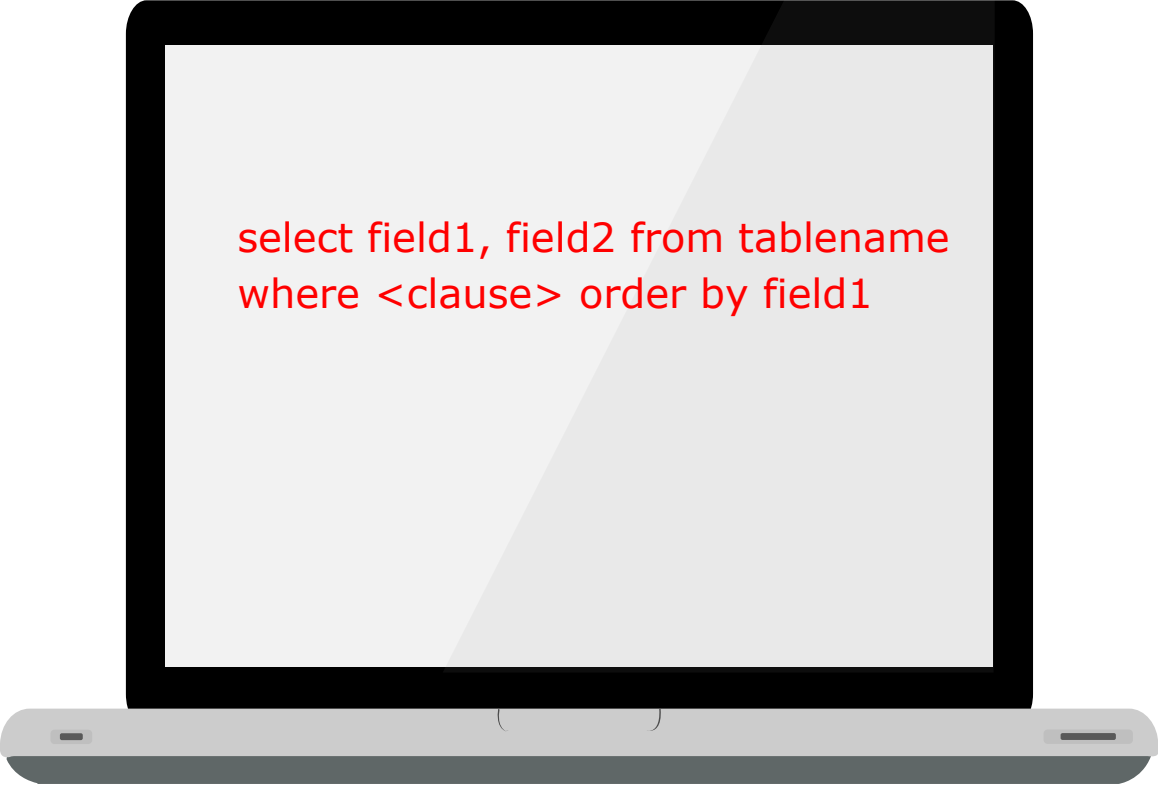
```
drop table  
myemployees_ts0211;
```

SORTING



Satyajit Pattnaik

- 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)



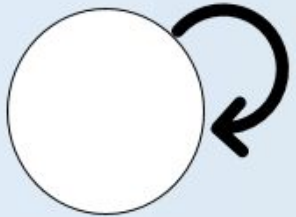
```
select field1, field2 from tablename  
where <clause> order by field1
```

JOINS

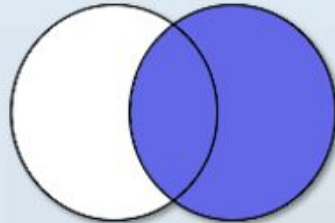
- In most of the real world problems, we might need data from multiple tables, that's where **Joins** comes into picture.

Joins in MySQL

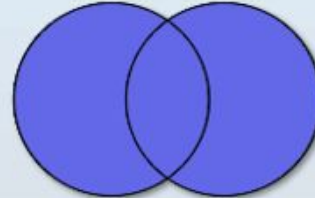
Self Join



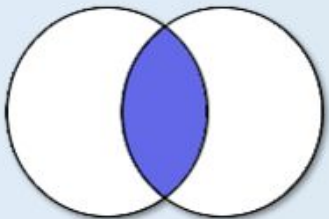
Right Join



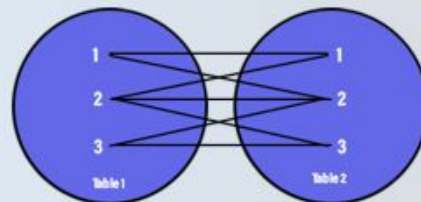
Full Join



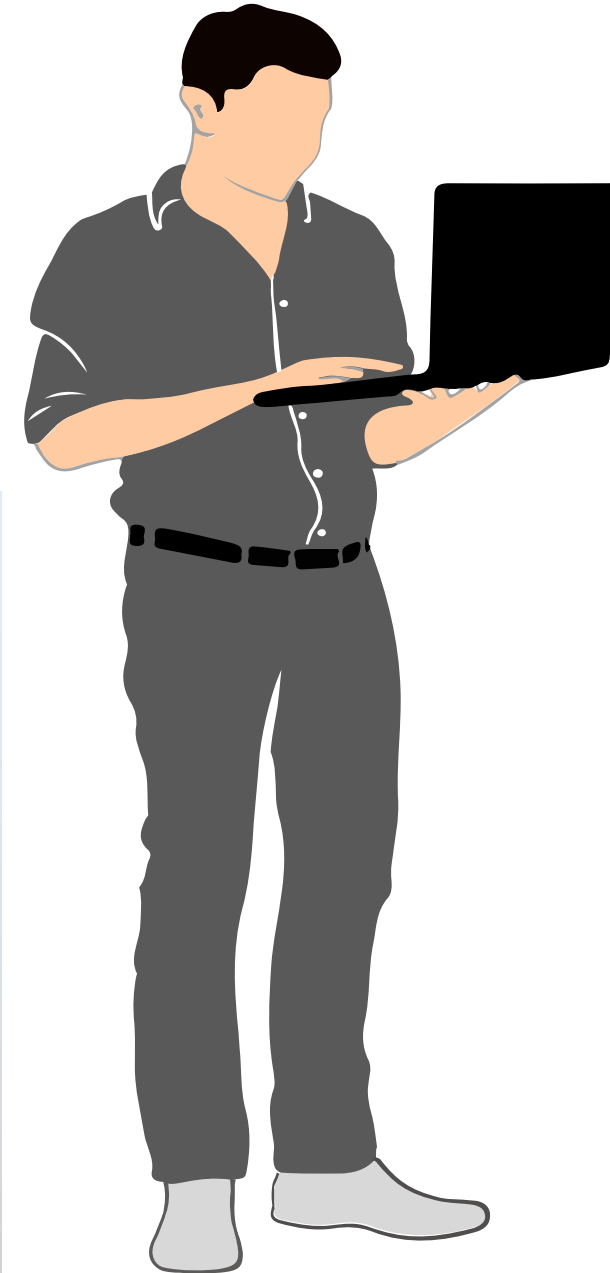
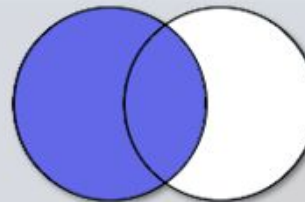
Inner Join



Cross Join



Left Join

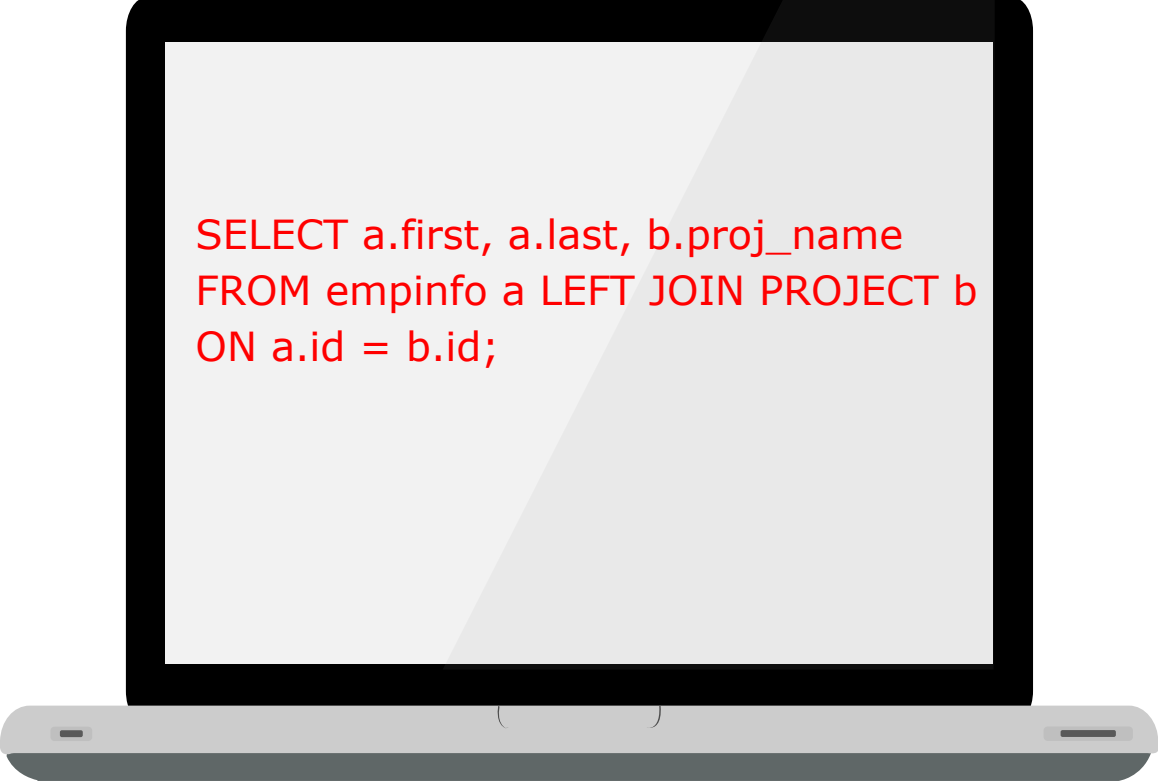


JOINS - Left Join



Satyajit Pattnaik

- 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.



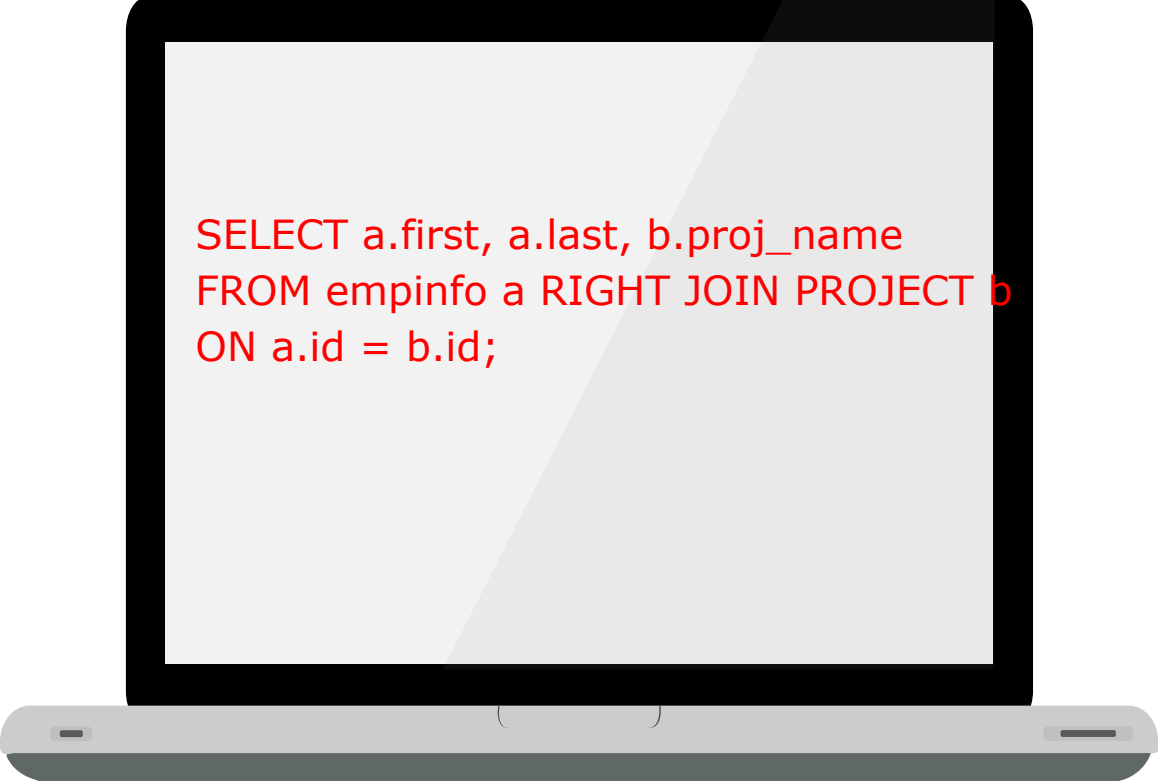
```
SELECT a.first, a.last, b.proj_name  
FROM empinfo a LEFT JOIN PROJECT b  
ON a.id = b.id;
```

JOINS - Right Join



Satyajit Pattnaik

- 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;
```

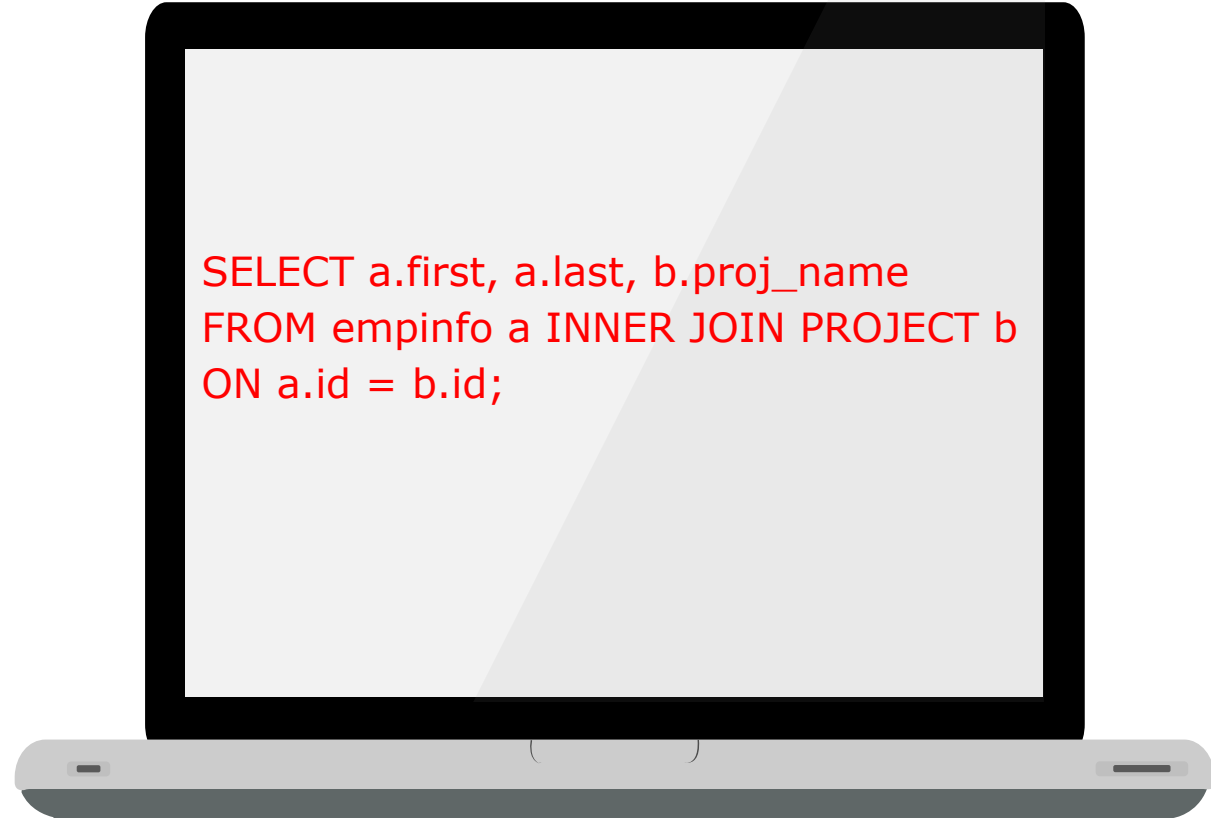
JOINS - Inner Join



Satyajit Pattnaik

- 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;
```





THANKS
FOR WATCHING!!