Learning Progress Review – Data RANGER

Learning never exhauts the mind

Kelompok 5

Rangers :

Indah Ayu Permatasari

Irdam Elba Septian

Irfan Muhammad Ghufron

Kautsar Hilmi

Lellyta Nurani Pangestika

Table of contents

01

Introduction to Advance Analysis

Proses analysis data lebih lanjut dengan menggunakan beberapa tools lanjutan

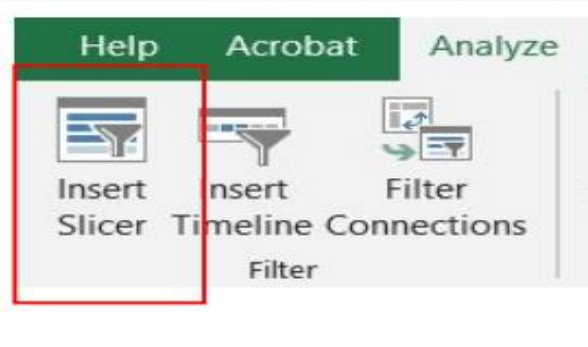
What Is Pivot Table

Pivot table merupakan sebuah fitur excel yang berfungsi untuk meringkas informasi yang ada di tabel database atau baris data.

Pivot Table Features

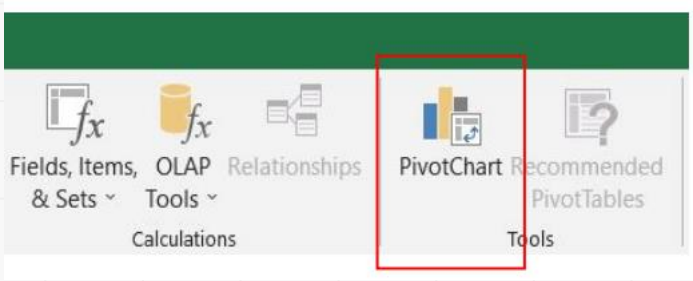
SLICER

Slicer merupakan salah satu fitur excel yang digunakan untuk memfilter data dan pivot table



PIVOT CHART

Pivotchart digunakan untuk memudahkan dalam menampilkan data menjadi lebih menarik dan interaktif.



What Is Forecasting

Forcasting merupakan pembuatan model mempelajari data lalu menggunakan nya untuk memprediksi apa yang akan terjadi dimasa depan terkait bisnis.

Forecasting Methods

CASUAL METHODS

Metode ini didasarkan pada keterikatan antara variable yang diperkirakan dengan variable lain yang mempengaruhinya.

TIME SERIES METHOD

Metode ini merupakan memfokuskan pada menganalisis pola data historis dari waktu ke waktu untuk membuat prediksi tentang nilai di masa depan.

QUALITATIVE METHODS

Metode ini merupakan metode yang dipengaruhi oleh Emosi,Pendidikan,Intuisi,Pengalaman sehingga hasil setiap orang akan berbeda.

What Is Regression Analysis

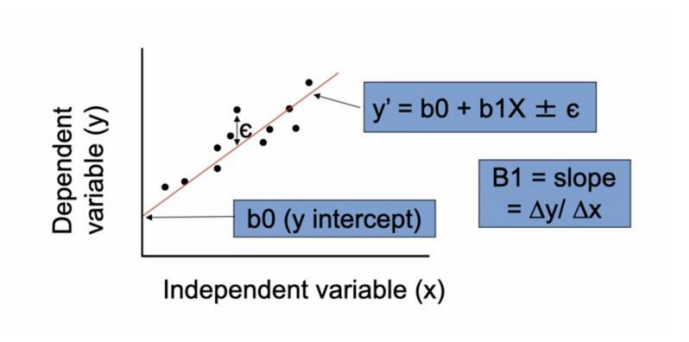
Analisis regresi merupakan serangkaian proses statistik untuk memperkirakan hubungan hubungan antara variable dependen dengan satu atau lebih variable independen.

- Variabel dependen(Y) = Variabel yang digunakan untuk melakukan prediksi.
- Variabel independent(X) = variabel-variabel yang dipakai untuk memperkirakan variable dependen.

Types of Regression Analysis

LINEAR REGRESSION

Output dari sebuah linear regresi adalah fungsi yang memprediksi variabel dependen berdasarkan nilai variabel independen.



MULTIPLE LINEAR REGRESSION

Lebih dari satu variabel independen dapat digunakan untuk menjelaskan variasi dalam variabel dependen, selama tidak berkaitan secara linear.

Y=A+BX1+BX2+BX3+..... +Xn+E.

Y = predict value (dependent variable).

X = feature value (independent variable).

A = intercept parameter

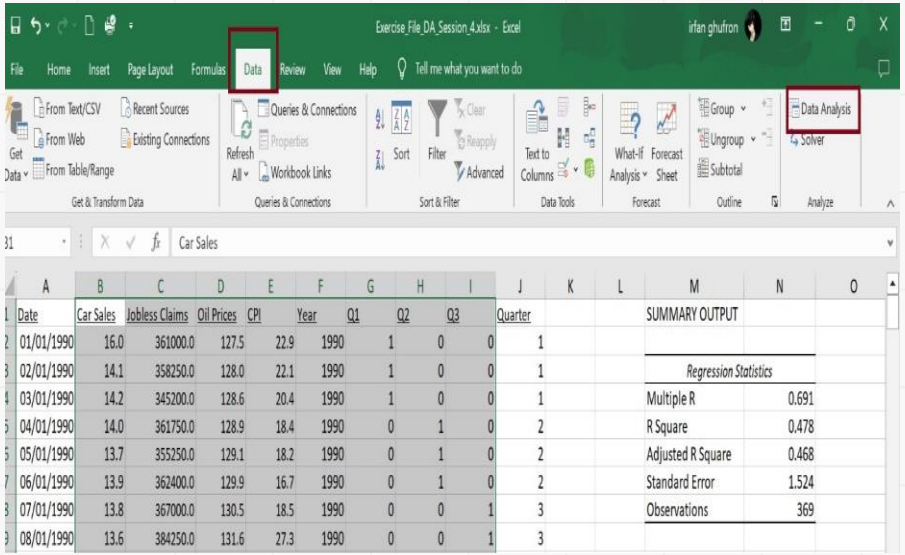
B = slope parameter

E = error (residual)

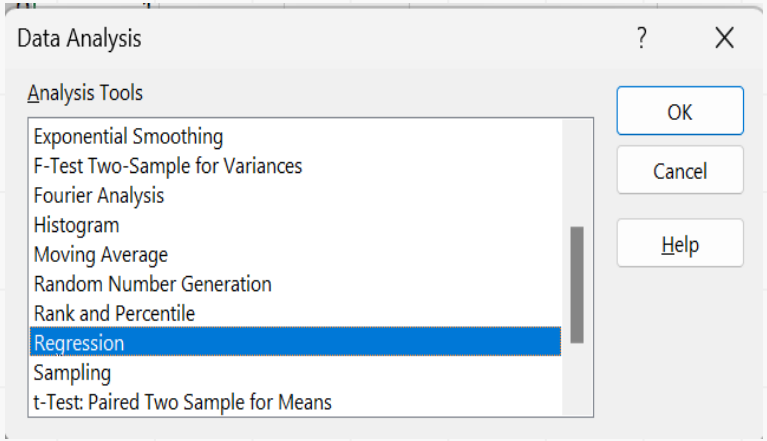
n = the number of variables or parameters.

Regression Analysis In Excel

STEP 1 : PILIH DATA PADA TOOLBAR LALU PILIH DATA ANALYSIS

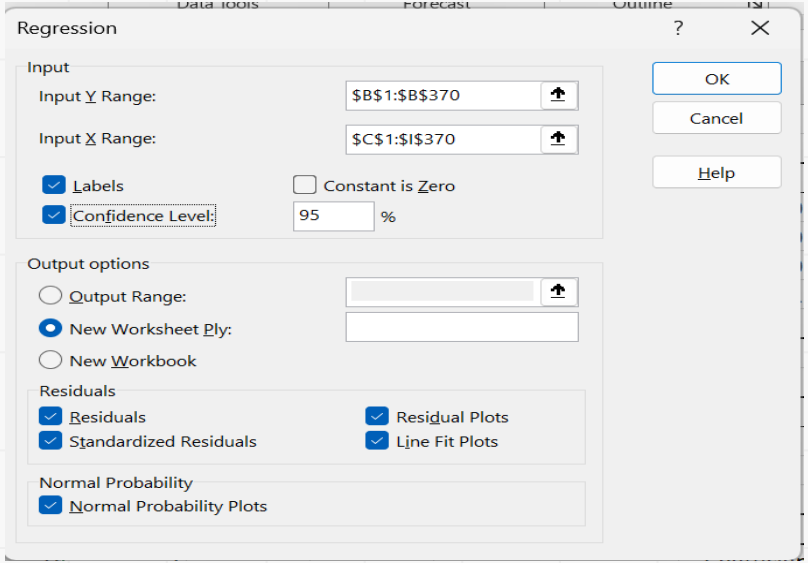


STEP 2 : PILIH REGRESSION



Regression Analysis In Excel

STEP 3 : MASUKAN X RANGE DAN Y RANGE, LALU TEKAN OK



Regression Analysis In Excel

SUMMARY REGRESSION ANALYSIS IN EXCEL

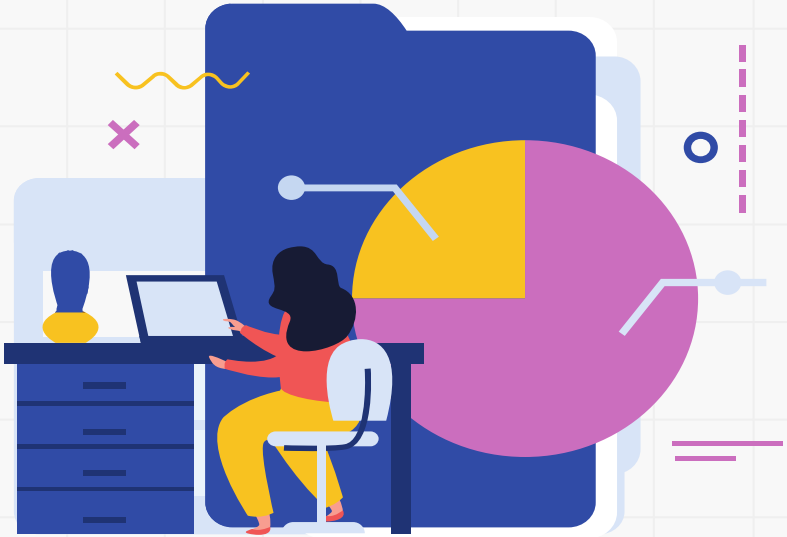
SUMMARY OUTPUT

Regression Statistics	
Multiple R	0.691391998
R Square	0.478022896
Adjusted I	0.467901456
Standard I	1.523777749
Observatic	369

ANOVA

	df	*SS*	*MS*	*F*	*Significance F*
Regressior	7	767.6225	109.6604	47.22875	2.35145E-47
Residual	361	838.2054	2.321899		
Total	368	1605.828			

	Coefficients	*andard Err*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*ower 95.0%*	*pper 95.0%*
Intercept	-4271.058644	523.9175	-8.15216	5.95E-15	-5301.372395	-3240.74	-5301.37	-3240.74
Jobless Cl	-2.81672E-06	2.65E-07	-10.6405	3.51E-23	-3.3373E-06	-2.3E-06	-3.3E-06	-2.3E-06
Oil Prices	-0.493413319	0.06492	-7.60028	2.57E-13	-0.621083037	-0.36574	-0.62108	-0.36574
CPI	0.009262459	0.007578	1.22231	0.222388	-0.005639775	0.024165	-0.00564	0.024165
Year	2.186733262	0.267523	8.173993	5.11E-15	1.660633503	2.712833	1.660634	2.712833
Q1	-1.78166147	0.303228	-5.87566	9.58E-09	-2.377975793	-1.18535	-2.37798	-1.18535
Q2	-1.179312125	0.274135	-4.30194	2.18E-05	-1.718413546	-0.64021	-1.71841	-0.64021
Q3	-0.449680614	0.236325	-1.90281	0.057861	-0.914426628	0.015065	-0.91443	0.015065



02

SQL 1 (Postgre)

Database

A database is an organized collection of structured information, or data, typically stored electronically in a computer system.

SQL

SQL is a relational database type. SQL stores data that follows specific relation or structure such as tabular data.
Application : MySql and PostgreSQL

NoSQL

No SQL is a non relational database type.No SQL stores unstructured data such as image, text, etc.
Application : MongoDB

Why Use Database ?

Access management can not be too complicated.

Databases let us work with large amounts of data efficiently.

Ensuring data security.

Databases store information accurately and reliably.

Database Command

DDL

Data Definition Language

Data definition language is used to define the data. It is used to create, alter, or drop the database/table.

DML

Data Manipulation Language

Data manipulation language is used to manipulate the data. It is used to delete, update, and insert. This function will only be usable after DDL gets executed.

DQL

Data Query Language

Data query language is used to making a query into the database. Regardless the query purpose, this type of command can be used to extract some information. The command can be select, table expression, sorting, limit, or with.

DCL

Data Control Language

Data control language is used to control anything regarding the data. It is various from granting user's access to revoking data access. The commands are such as grant, revoke, commit, and rollback.

Command in Action

DDL

Create Database

```
CREATE DATABASE database_name;

Example:
CREATE DATABASE dataranger_database;
```

Create Table

```
CREATE TABLE table_name (
Nama_Kolom_1 Tipe_Data_1,
Nama_Kolom_2 Tipe_Data_2,
Nama_Kolom_3 Tipe_Data_3
);

Example:
CREATE TABLE Anggota (
no INT,
nama VARCHAR,
tanggal_lahir DATE
);
```

Command in Action

DML

Insert Data

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);

Example:
INSERT INTO Anggota (no, nama, tanggal_lahir) VALUES
(1,”Bunga”, “2024-01-01”);
```

Update Table

```
UPDATE table_name SET column1 = value1 WHERE condition;

Example:
UPDATE anggota SET nama = “Budi” WHERE no = 1;
```

Deleting Data

```
DELETE FROM table_name WHERE condition;

Example:
DELETE FROM Anggota WHERE nama = “Budi”;
Notes: To delete the whole data, just don’t use where
clause or setting where to True.
```

Command in Action

SQL 1 (Postgre)

Querying Data

In order to select or known as querying data, we can basically be able to make use of SELECT command. It is recommended to only select specific column name from a table.
If we by any chance need the whole columns, we can use * to represent the whole columns.

```
SELECT column_name FROM table_name;
Example :
SELECT * FROM Anggota;
```

*Notes: It is not recommended to use * on any query especially for big result.*

DQL

Removing Duplicates

In various use cases, the data might be duplicated and we need to only get the unique values of them. To achieve this goal, we can use DISTINCT keyword.

```
SELECT DISTINCT column_name FROM
table_name;

Example:
SELECT DISTINCT * FROM anggota;
```

Limiting Query Result

```
SELECT column_name FROM table_name
LIMIT n;

Example :

SELECT * FROM Anggota LIMIT 5;
```

Command in Action

DQL

Applying Some Condition

In some cases, we might need to filter some data, we utilize filter to achieve that.

```
SELECT column_name FROM table_name WHERE column_a = condition;

Example:
SELECT * FROM anggota WHERE column_a = “name”;
```

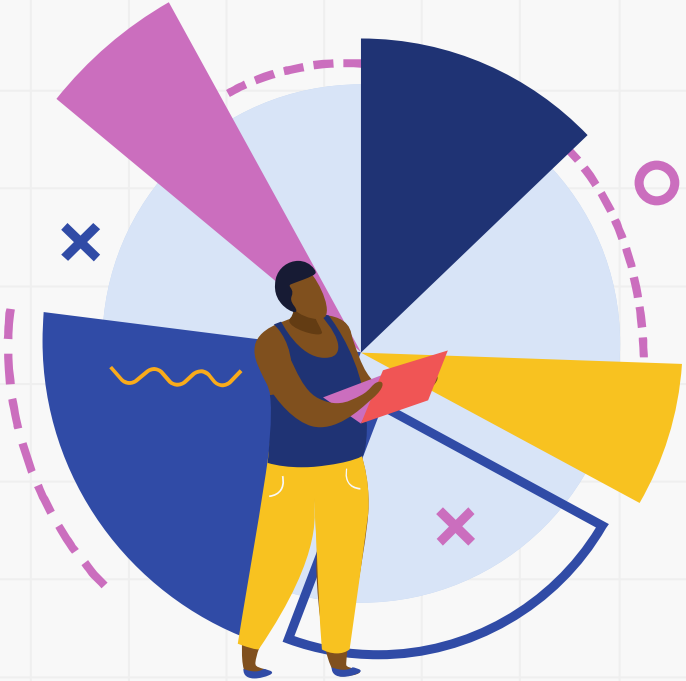
Utilizing IF ELSE in Query

```
SELECT CASE WHEN column_name = condition_a THEN 'a' ELSE 'b'
END AS new_column_name FROM table_a;

Example:
SELECT CASE WHEN Anggota = 'Arief' THEN True ELSE False END AS
is_contain_arief FROM table_a;
```

String Functions

REPLACE	Replaces all occurrences in string of substring from with substring to. Ex : replace('abcdefabcdef', 'ab', 'XX') → XXcdefXXcdef
CONCAT	Concatenates the text representations of all the arguments. NULL arguments are ignored. Ex : concat('abcde', 2, NULL, 22) → abcde222
UPPER	Converts the string to all upper case. Ex: upper('saya') → SAYA
LOWER	Converts the string to all lower case. Ex: lower(SAYA') → saya
REGEX	Pattern matching operation based on the regular expressions and the REGEXP operator. Ex : REGEXP '^sa' → sam, samarth *More Functions: https://www.postgresql.org/docs/current/functions-string.html*



03

SQL 2

Query Aggregation

A process to Aggregate your query result into larger group.

Aggregation Function

1. SUM : Sum up your numeric data.
2. MAX :Find out the maximum values.
3. MIN : Find out the minimum values.
4. AVG : Find out the average values.
5. COUNT : Find out the number of records

```
Syntax :
table_a : your table name
transaction_id : unique id transaction
Transaction_amount : value of transaction

Example :
SELECT
  COUNT(DISTINCT transaction_id) total,
  MAX(transaction_amount) max_val,
  MIN(transaction_amount) min_val,
  AVG(transaction_amount) avg_val
FROM
  table_a
```

Query Aggregation

Grouped

To group your query, use GROUP BY statement. Another use of grouping is to get distinct value without utilizing DISTINCT

```
Syntax :
a : your table name
transaction_date : date of transaction
transaction_id : unique id transaction
transaction_amount : Value of transaction

Example :
SELECT
  transaction_date,
  COUNT(DISTINCT transaction_id) total,
  MAX(transaction_amount) max_val,
  MIN(transaction_amount) min_val,
  AVG(transaction_amount) avg_val
FROM
  a
GROUP BY 1
```

Query Aggregation

Subsetting Aggregate

To subset your query based on aggregated metric, we can't use WHERE clause, instead we use HAVING clause. For example, we want to subset the result where the average of the transaction amount is greater than 5,000.

```
Syntax :
a : your table name
transaction_date : date of transaction
transaction_id : unique id transaction
transaction_amount : amount of transaction

Example :
SELECT
  transaction_date,
  COUNT(DISTINCT transaction_id) total,
  MAX(transaction_amount) max_val,
  MIN(transaction_amount) min_val,
  AVG(transaction_amount) avg_val
FROM
  a
GROUP BY 1
HAVING avg_val > 5000
```

Query Ordering

Ordering

To order your query, use ORDER BY statement. Ordering always takes place at the end of the query before LIMIT if you limit your query.
If you don't specify the order, by default it is ascending. To order your data to descending order, use DESC keyword after the column you used to order the data.

```
Syntax :
a : your table name
transaction_date : date of transaction
transaction_id : unique id transaction
transaction_amount : Value of transaction

Example :
SELECT
  transaction_date,
  COUNT(DISTINCT transaction_id)
total
FROM
  a
GROUP BY 1
HAVING total > 2
ORDER BY 1 DESC, 2
LIMIT 10
```

Utilizing Like Statement

LIKE statement is used together with WHERE statement. It is used to filter column that has STRING type to look certain pattern.

1. "%" sign represents any character regardless the number. Hence, "a%" will equal to started by a and followed by any character. "ayam" will match the filter, but "maya" not satisfy the requirement.

2. "_" sign represents a single character. Hence, "a_am" will equal to started by "a", followed by any single character, but after that should be followed by "am". "alam", "ayam", "azam" will satify the requirement, but "ayah" and "agaam" will not match the filter.

```
Example :
SELECT order_id FROM orders WHERE customer_name LIKE 'ber%'
```

Date Functions

current_date	This statement gets current date. Ex: SELECT CURRENT_DATE
date_trunc()	Truncate to specific precision. Ex: SELECT DATE_TRUNC('month', CURRENT_DATE)
extract()	Get specific date part. Ex: SELECT EXTRACT(MONTH FROM CURRENT_DATE)
date_add()*	Add interval into our date data. Ex: SELECT DATE_ADD(CURRENT_DATE, '1 day'::interval)
date_sub()*	Subtract interval from our date data. Ex: SELECT DATE_SUB(CURRENT_DATE, '1 day'::interval)

** : may not work in all systems*

Subqueries

Subqueries is a query on another query. Subquery can be placed after FROM, after any joining clause, or at filter clause (WHERE or HAVING).

```
Syntax :
SELECT order_id
FROM (SELECT order_id FROM orders) AS a
RIGHT JOIN (SELECT order_id FROM order_details) AS b
         ON a.order_id = b.order_id
WHERE order_id IN (SELECT DISTINCT order_id FROM order_main)
GROUP BY order_id
HAVING SUM(order_amount) >= (SELECT AVG(order_amount) FROM orders)
```

CTE

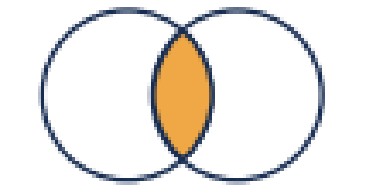
CTE (Common Table Expression) can be treated as temporary table living inside your query. To use CTE, you need to open the queries using WITH statement.

```
Syntax :
WITH cte_1 AS(
SELECT DISTINCT order_id FROM orders
),
cte_2 AS(
SELECT DISTINCT order_id FROM order_main
)
SELECT order_id
FROM cte_1 AS a
RIGHT JOIN cte_2 AS b
     ON a.order_id = b.order_id
WHERE order_id IN (SELECT DISTINCT order_id FROM order_main)
GROUP BY order_id
HAVING SUM(order_amount) >= (SELECT AVG(order_amount) FROM orders)
```

Table Joining

Inner join

Inner join will only takes the record that is intersected to each other based on their join key.



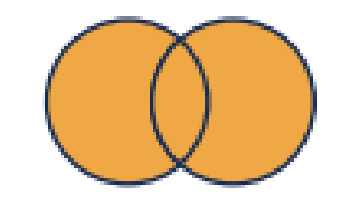
```
Syntax :
INNER JOIN
Inner Join Structure:
SELECT column_name
FROM left
INNER JOIN right ON left.key = right.key

Example:
SELECT left.order_id
FROM orders AS left
INNER JOIN order_details AS right
     ON left.order_id = right.order_id
```

Table Joining

Full join

Full join will take ALL records from both table either their match or not with the key. To avoid NULL values possibility, you can always use COALESCE or IFNULL to get values from both tables.



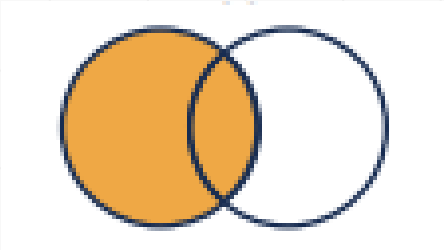
```
Syntax :
SELECT column_name
FROM left
FULL JOIN right ON left.key = right.key

Example:
  SELECT COALESCE(left.order_id,
right.order_id)
order_id
  FROM orders AS left
  FULL JOIN order_details AS right
      ON left.order_id = right.order_id
```

Table Joining

Left Join

Left join will take all records from left table, but only takes intersected records from right table.



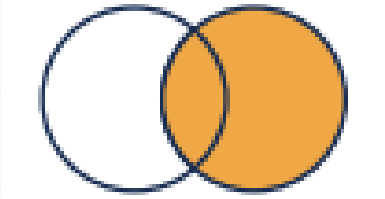
```
Syntax :
SELECT column_name
FROM left
LEFT JOIN right ON left.key = right.key

Example:
SELECT left.order_id order_id
FROM orders AS left
LEFT JOIN order_details AS right
    ON left.order_id = right.order_id
```

Table Joining

Right Join

Right join will take all records from right table, but only takes intersected records from left table.



```
Syntax :
SELECT column_name
FROM left
RIGHT JOIN right ON left.key = right.key

Example:
SELECT right.order_id order_id
FROM orders AS left
RIGHT JOIN order_details AS right
ON left.order_id = right.order_id
```

Key

Primary Key

A set of data that uniquely identifies each record.

Foreign Key

A set of data that refers to primary key of another table

Terima Kasih!

“No data is clean,
But most is useful.”

— Dean Abbott