SELECT \*

FROM

PortfolioProject..CovidDeaths$

ORDER BY 3,4

-----------------------------------

--TOTAL CASES vs TOTAL LIVES LOST

SELECT

location, date, total\_cases, total\_deaths, (total\_cases/total\_deaths)\*100 AS percent\_lost\_lives

FROM

PortfolioProject..CovidDeaths$

WHERE

location like '%states%'

ORDER BY

1,2;

--------------------------------

SELECT

location, date, total\_cases, total\_deaths, (total\_cases/total\_deaths)\*100 AS percent\_lost\_lives

FROM

PortfolioProject..CovidDeaths$

WHERE

location = 'Canada'

ORDER BY

1,2;

--------------------------

---TOTAL COVID CASES vs TOTAL POPULATION

SELECT

location, date, population, total\_cases, (total\_cases/population)\*100 AS percent\_population\_infected

FROM

PortfolioProject..CovidDeaths$

WHERE

location = 'Canada'

ORDER BY

1,2;

-----COUNTRIES WITH HIGHEST COVID RATES vs TOTAL POPULATION

SELECT

location, population, MAX(total\_cases) AS infection\_count,

MAX(total\_cases/population)\*100 AS percent\_population\_infected

FROM

PortfolioProject..CovidDeaths$

GROUP BY

location, population

ORDER BY

percent\_population\_infected DESC;

-----COUNTRIES WITH HIGHEST LIVES LOSSES PER TOTAL POPULATION

SELECT

location, MAX(CAST(total\_deaths AS int)) AS total\_lives\_lost

FROM

PortfolioProject..CovidDeaths$

GROUP BY

location

ORDER BY

total\_lives\_lost DESC;

---LIVES LOST BY CONTINENTS

SELECT

location, MAX(CAST(total\_deaths AS int)) AS total\_lives\_lost

FROM

PortfolioProject..CovidDeaths$

WHERE

continent IS NULL

GROUP BY

location

ORDER BY

total\_lives\_lost DESC;

-----------------------------

--GLOBAL COVID NUMBERS

SELECT

date, SUM(new\_cases)AS total\_cases, SUM(CAST(new\_deaths AS int)) AS total\_lives\_lost,

SUM(CAST(new\_deaths AS int))/ SUM(new\_cases)\*100 AS percent\_total\_deaths

FROM

PortfolioProject..CovidDeaths$

WHERE

continent IS NOT NULL

GROUP BY

date

ORDER BY

1,2;

----------------------------

SELECT

SUM(new\_cases)AS total\_cases, SUM(CAST(new\_deaths AS int)) AS total\_lives\_lost,

SUM(CAST(new\_deaths AS int))/SUM(new\_cases)\*100 AS percent\_total\_deaths

FROM

PortfolioProject..CovidDeaths$

WHERE

continent IS NOT NULL

ORDER BY

1,2;

--------------TOTAL POPULATION vs TOTAL VACCINATION

SELECT

dea.continent,

dea.location,

dea.date,

dea.population,

vac.new\_vaccinations,

SUM(CONVERT(bigint, vac.new\_vaccinations)) OVER (PARTITION BY dea.location ORDER BY dea.location, dea.date) AS rolling\_num\_vac\_people

FROM

PortfolioProject..CovidDeaths$ dea

JOIN

PortfolioProject..CovidVaccinations$ vac

ON

dea.location = vac.location

AND

dea.date = vac.date

WHERE

dea.continent IS NOT NULL

ORDER BY

2,3;

----------------------------------

--USING COMMON TABLE EXPRESSION(CTE)

With PopvsVac (continent, location, date, population, new\_vaccinations, rolling\_num\_vac\_people)

AS

(

SELECT

dea.continent,

dea.location,

dea.date,

dea.population,

vac.new\_vaccinations,

SUM(CONVERT(bigint, vac.new\_vaccinations)) OVER (PARTITION BY dea.location ORDER BY dea.location, dea.date) AS rolling\_num\_vac\_people

FROM

PortfolioProject..CovidDeaths$ dea

JOIN PortfolioProject..CovidVaccinations$ vac

ON dea.location = vac.location

AND dea.date = vac.date

WHERE

dea.continent IS NOT NULL

)

---------------

SELECT \*, (rolling\_num\_vac\_people/population)\*100

FROM PopvsVac;

-----TEMPORARY TABLE---

CREATE TABLE #percentofvacpeople

(continent nvarchar (255),

location nvarchar (255),

date datetime,

population numeric,

new\_vaccinations numeric,

rolling\_num\_vac\_people numeric

)

INSERT INTO #percentofvacpeople

SELECT

dea.continent,

dea.location,

dea.date,

dea.population,

vac.new\_vaccinations,

SUM(CONVERT(bigint, vac.new\_vaccinations)) OVER (PARTITION BY dea.location ORDER BY dea.location, dea.date) AS rolling\_num\_vac\_people

FROM

PortfolioProject..CovidDeaths$ dea

JOIN PortfolioProject..CovidVaccinations$ vac

ON dea.location = vac.location

AND dea.date = vac.date

SELECT \*, (rolling\_num\_vac\_people/population)\*100

FROM #percentofvacpeople;

--------------------------

DROP TABLE IF EXISTS #percentofvacpeople

CREATE TABLE #percentofvacpeople

(continent nvarchar (255),

location nvarchar (255),

date datetime,

population numeric,

new\_vaccinations numeric,

rolling\_num\_vac\_people numeric

)

INSERT INTO #percentofvacpeople

SELECT

dea.continent,

dea.location,

dea.date,

dea.population,

vac.new\_vaccinations,

SUM(CONVERT(bigint, vac.new\_vaccinations)) OVER (PARTITION BY dea.location ORDER BY dea.location, dea.date) AS rolling\_num\_vac\_people

FROM

PortfolioProject..CovidDeaths$ dea

JOIN PortfolioProject..CovidVaccinations$ vac

ON dea.location = vac.location

AND dea.date = vac.date

SELECT \*, (rolling\_num\_vac\_people/population)\*100

FROM #percentofvacpeople;

----CREATING VIEW

CREATE VIEW percentofvacpeople AS

SELECT

dea.continent,

dea.location,

dea.date,

dea.population,

vac.new\_vaccinations,

SUM(CONVERT(bigint, vac.new\_vaccinations)) OVER (PARTITION BY dea.location ORDER BY dea.location, dea.date) AS rolling\_num\_vac\_people

FROM

PortfolioProject..CovidDeaths$ dea

JOIN PortfolioProject..CovidVaccinations$ vac

ON dea.location = vac.location

AND dea.date = vac.date

WHERE

dea.continent IS NOT NULL

--------------------

SELECT \*

FROM

percentofvacpeople;

JOIN TABLES

SELECT

    employees.name AS employee\_name,

    employees.role AS employee\_role,

    departments.name AS department\_name

FROM

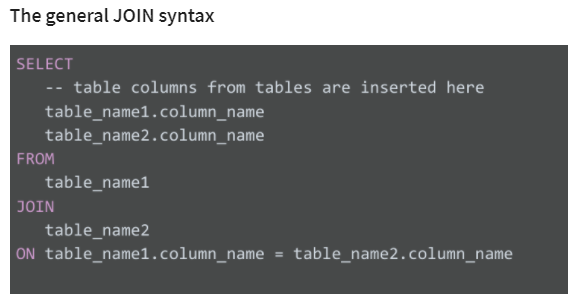
    employee\_data.employees

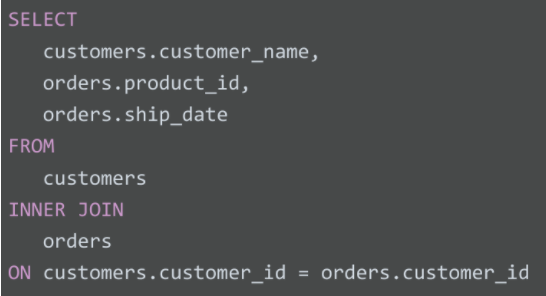
FULL OUTER JOIN

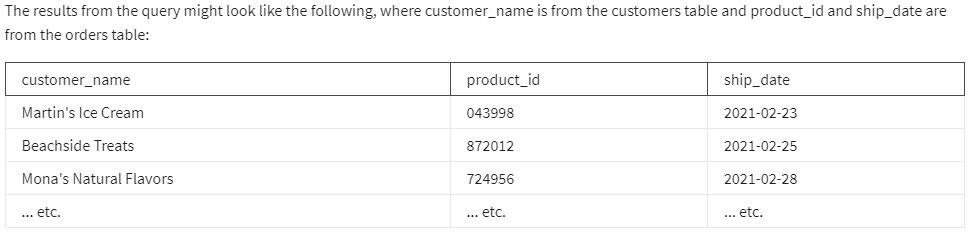
    employee\_data.departments

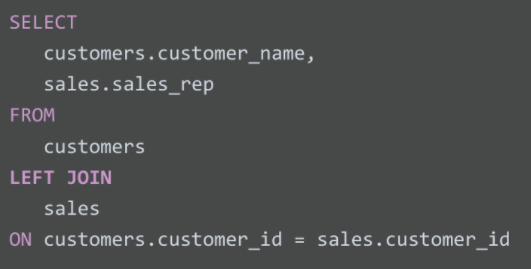
ON

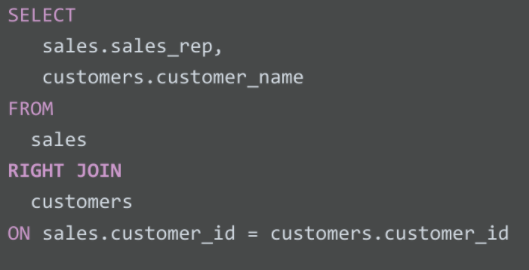
    employees.department\_id = departments.department\_id

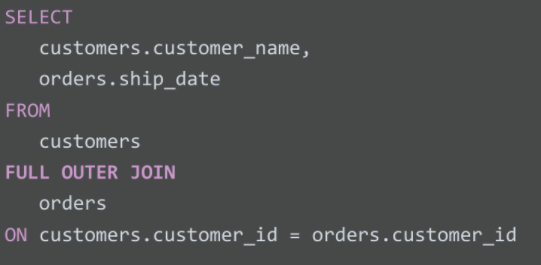












-- Let's say table\_1 has 100 rows and table\_2 has 10 rows.

-- They share 10 keys in common.

-- Using INNER JOIN --> We get 10 rows in our results.

SELECT

    COUNT(\*)

FROM

    table\_1

INNER JOIN

    table\_2

ON table\_1.key = table\_2.key;

-- Using LEFT JOIN --> We get 100 rows in our results.

SELECT

    COUNT(\*)

FROM

    table\_1

LEFT JOIN

    table\_2

ON table\_1.key = table\_2.key;

SQL QUERIES AND SUBQUERIES

NUMBER OF BIKES AVAILABLE AT A STATION TO THE AVERAGE NUMBER OF BIKES AVAILABLE

SELECT

    station\_id,

    num\_bikes\_available,

    (SELECT

     AVG(num\_bikes\_available)

    FROM bigquery-public-data.new\_york\_citibike.citibike\_stations) AS avg\_num\_bikes\_available

FROM bigquery-public-data.new\_york\_citibike.citibike\_stations

NUMBER OF RIDES THAT HAVE STARTED AT EACH STATION OVER TIME

SELECT

    station\_id,

    name,

    number\_of\_rides AS num\_of\_rides\_starting\_at\_station

FROM

    (

        SELECT

            start\_station\_id,

            COUNT(\*) number\_of\_rides

        FROM

            bigquery-public-data.new\_york\_citibike.citibike\_trips

        GROUP BY

            start\_station\_id

    )

    AS station\_num\_trips

    INNER JOIN

    bigquery-public-data.new\_york\_citibike.citibike\_stations ON station\_id = start\_station\_id

    ORDER BY

        number\_of\_rides DESC

LIST OF STATIONS THAT BIKESHARE SUBSCRIBERS USE BIKES

SELECT

    station\_id,

    name

FROM

    bigquery-public-data.new\_york\_citibike.citibike\_stations

WHERE

    station\_id IN

    (

        SELECT

            start\_station\_id

        FROM

            bigquery-public-data.new\_york\_citibike.citibike\_trips

        WHERE

            usertype =

        'Subscriber'

    )