SQL SCRIPTS

SELECT \* FROM bankimported

where location like 's%'; S FIRST LETTER

SELECT \* FROM bankimported

where location like '%s'; S LAST LETTER

SELECT \* FROM bankimported

where location like '%s%'; CONTAINS S ANYWHERE

SELECT AVG(price) AS average\_price FROM sales average of price column

SELECT min(price) AS average\_price FROM sales minimum price

SELECT max(price) AS average\_price FROM sales maximum price

SELECT count(price) AS average\_price FROM sales counts how many prices in column

SELECT sum(price) AS average\_price FROM sales total of price

Inner joins learner1 and 3 using learner id

SELECT \* FROM cedric.learner1

INNER JOIN cedric.learner3

ON learner1.Learner\_ID = learner3.Learner\_ID;

Left joins learner 1 and 3 tables

SELECT \*

FROM cedric.learner1

LEFT JOIN cedric.learner3

ON learner1.Learner\_ID = learner3.Learner\_ID;

Right joins table 1 AND 3

SELECT \*

FROM cedric.learner1

LEFT JOIN cedric.learner3

ON learner1.Learner\_ID = learner3.Learner\_ID;#

Union joins

LEFT JOIN

UNION

RIGHT JOIN

FULL JOINS BOTH TABLES

SELECT   
 b.BOOK\_ID,   
 b.BOOK\_NAME,   
 a.AUTHOR\_NAME,   
 p.PUBLISHER\_NAME   
FROM   
 Books b  
FULL JOIN Authors a ON b.BOOK\_ID = a.AUTHOR\_ID  
FULL JOIN Publishers p ON b.BOOK\_ID = p.PUBLISHER\_ID;

Adding contraint

Alter table banktransac

Add constraint agecontraint

Check(customerage between 18 and 100)

Where / like

SELECT \*  
FROM Customers  
WHERE Country = 'Spain' AND CustomerName LIKE 'G%';

Order descending

SELECT \* FROM Products  
ORDER BY Price DESC;

Select from

SELECT CustomerName, City FROM Customers;

delete

DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste';

Select top 3

SELECT TOP 3 \* FROM Customers;

Starts with L end with on  
WHERE City LIKE 'L\_\_\_on';

any character followed by ondon  
WHERE City LIKE '\_ondon';

contains ‘mer’

WHERE CustomerName LIKE '%mer%';

Ends with ‘es’

WHERE CustomerName LIKE '%es';

Starting with either b, s, or p

WHERE CustomerName LIKE '[bsp]%';

Starting with any letter from a to f

WHERE CustomerName LIKE '[a-f]%';

 "r" in the second position

WHERE CustomerName LIKE '\_r%';

starts with "a"

WHERE CustomerName LIKE 'a\_%';

Selects all products with a price between 10 and 20:

SELECT \* FROM Products  
WHERE Price BETWEEN 10 AND 20;

If salary less 2000, add the alies, ‘its like a string’

Case when salary < 2000 then ‘a. less than 20,000’

If salary over 400k and location is US add 1 to count else 0, end statement creates new column

Select location, salary,

Case when salary > 400000 and location = ‘US then 1 else 0

End as salaries\_over\_400000\_and\_US

From database.salaries

PANDAS:

<https://www.tutorialspoint.com/python_pandas/python_pandas_descriptive_statistics.htm>

opening csv file and perform describing statistics

import pandas as pd

docx = pd.read\_csv(r"C:\Users\user\Downloads\Sales.csv")

print(docx.describe())

predict y axis using linear regression

import pandas as pd

from sklearn.linear\_model import LinearRegression

docx = pd.read\_csv(r"C:\Users\user\Downloads\Sales.csv")

x = docx[['Unit\_Price']]

y = docx[['Sales']]

train\_data = LinearRegression()

train\_data.fit(y, x)

predictions\_of\_sales = train\_data.predict(y)

print(predictions\_of\_sales)

print numbers not multiples of 5

number = 1

while number < 14:

if not number % 5 ==0:

print('The number is:', number)

number+=1

print name alternating between uppercase and lowercase

letter=0

output=''

Name='Cedric'

while letter<len(Name) :

if letter %2==0:

output+=Name[letter].upper()

else :

output+=Name[letter].lower()

letter+=1

print(output)

filtering in pandas

data =pd.read\_csv(“pd.read\_csv(r"C:\Users\user\Downloads\Sales.csv”)

greater\_than = data[(data[‘Salary’]<35000)]

equal\_to = data[(data[‘Salary’]==4000)]

Less\_than = data[(data['Salary']<35000)]  
print(Less\_than)

#Tilde ~ means NOT    
#The & sign means AND   
# The verticle line | means OR

#The & sign means AND  
the\_and\_sign=Sales\_data[(Sales\_data['Unit\_Price']>100)&(Sales\_data['Order\_Quantity']>45)]  
print(the\_and\_sign)

the\_OR\_sign=Sales\_data[(Sales\_data['Unit\_Price']>100)|(Sales\_data['Order\_Quantity']>45)]  
print(the\_OR\_sign)

# NOT ~ examples  
Unit\_price\_more= Sales\_data[~(Sales\_data['Unit\_Price']>150)]  
print(Unit\_price\_more)

#Inverse of AND statment with NOT (~)  
the\_AND\_sign=Sales\_data[~((Sales\_data['Unit\_Price']>100)&(Sales\_data['Order\_Quantity']>45))]  
print(the\_AND\_sign)

#Filtering by Item   
filtering\_by\_item= Sales\_data[Sales\_data['Order\_Priority'].isin(['High','Low'])]  
print(filtering\_by\_item)

#Filtering by letters  
filtering\_by\_letters=Sales\_data[Sales\_data['Region'].str.contains('w',case=False)]  
print(filtering\_by\_letters)

#try and search for Region, all values that don't contain 'W'  
No\_W= Sales\_data[~(Sales\_data['Region'].str.contains('w',case=False))]  
print(No\_W)

#Inverse of Orders that are High and Low  
filtering\_by\_item= Sales\_data[~Sales\_data['Order\_Priority'].isin(['High','Low'])]  
print(filtering\_by\_item)