NIDA WISDOM for Change คณะสถิติประยุกต์

Analytics on Cloud using Google BigQuery and Colaboratory

Thanachart Ritbumroong, Ph.D.
Lecturer
Business Analytics and Data Science

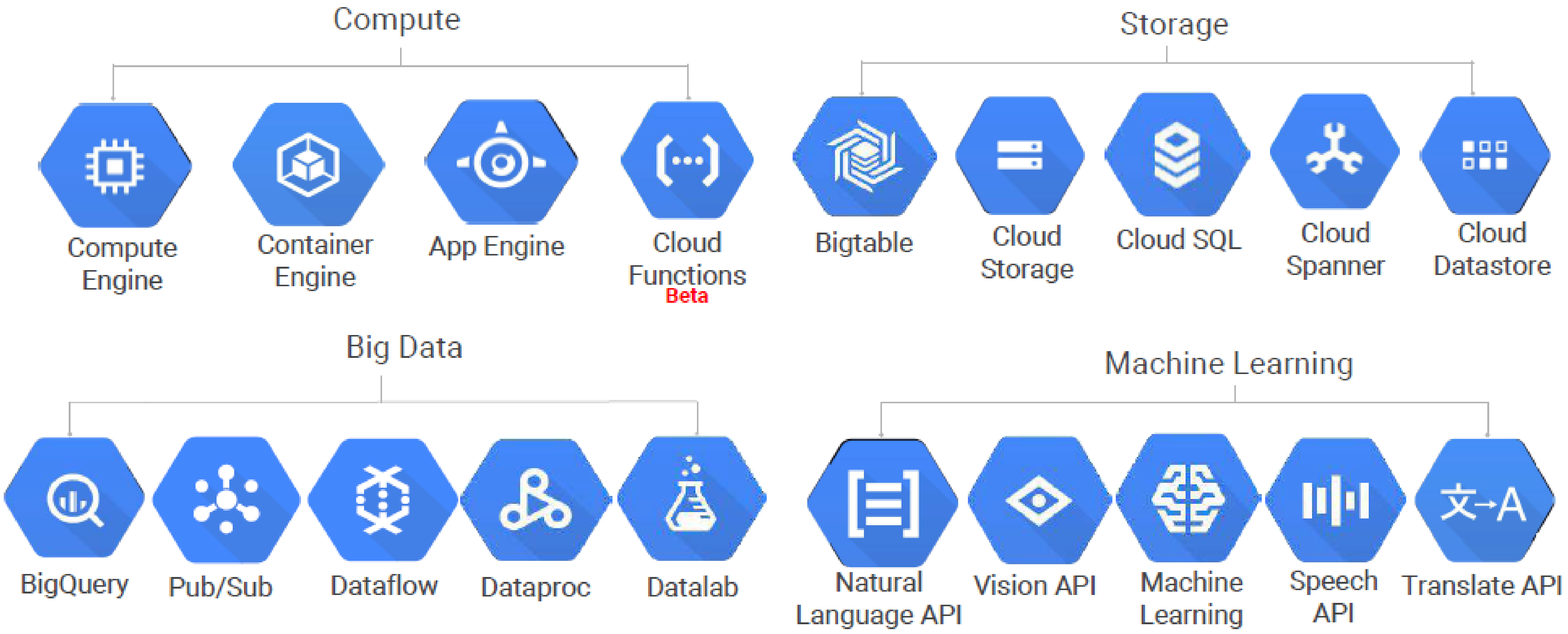
Workshop Objectives

- Understand Google Cloud Platform Environment for Analytics Tasks

- Practice using Google BigQuery for data processing

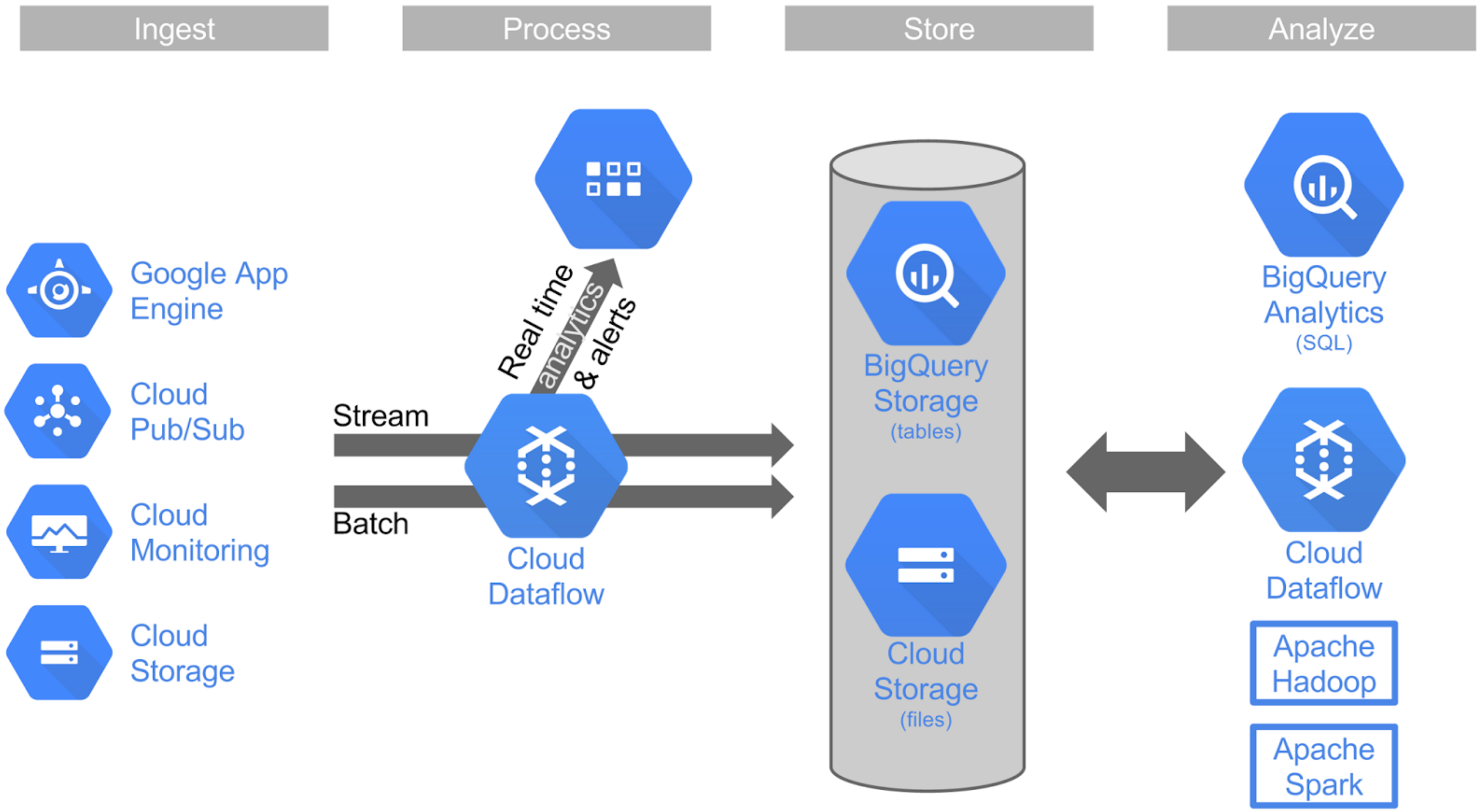
- Practice using Google Colaboratory for data analytics

Google Cloud Platform



https://www.e-goo.it/wp-content/uploads/2017/10/GoogleCloudPlatformServices.png

Data Analytics on Google Cloud Platform



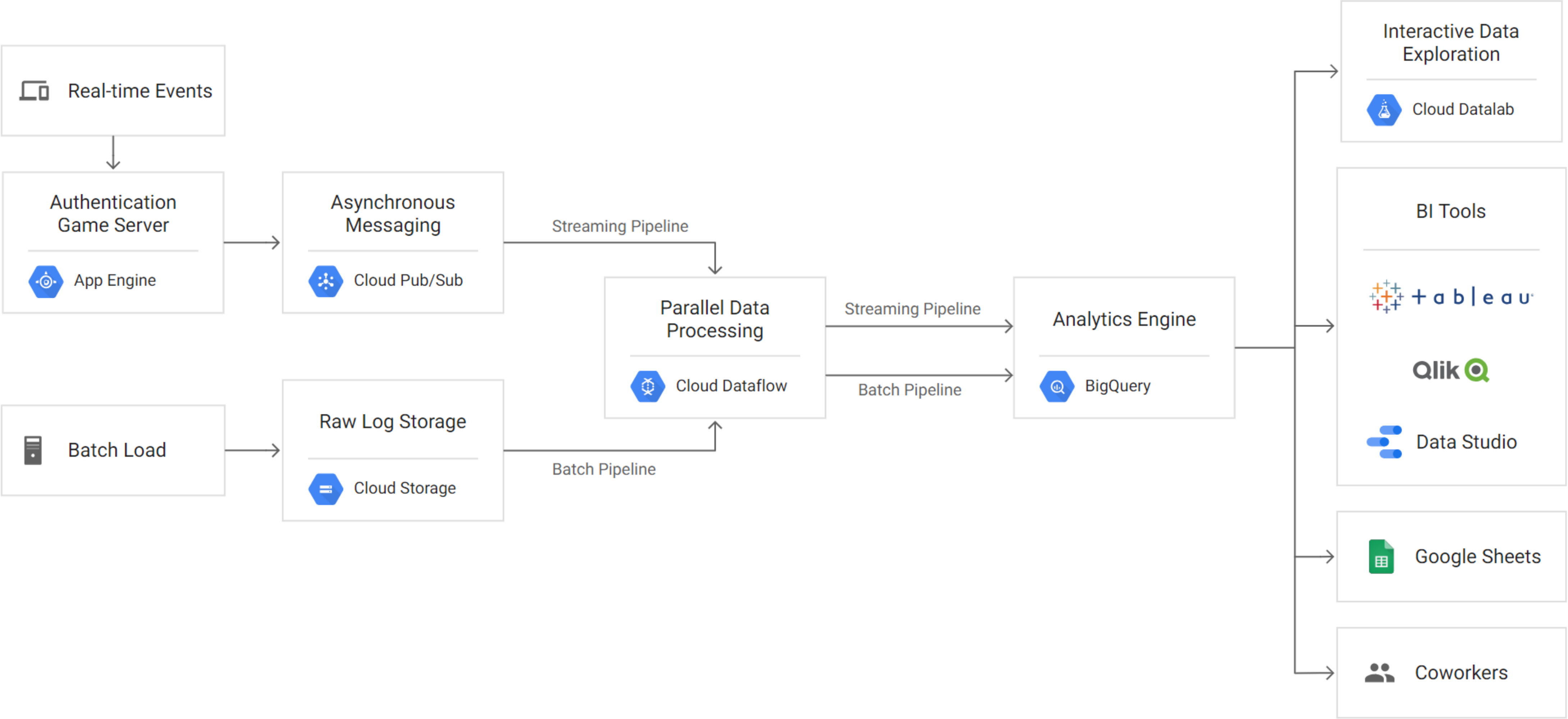
https://www.storagereview.com/google_tackles_big_data_through_updates_to_bigquery_dataflow

Google BigQuery

BigQuery is Google's **serverless, highly scalable, enterprise data warehouse** designed to make all your data analysts productive at an unmatched price-performance. Because there is no infrastructure to manage, you can focus on analyzing data to find meaningful insights using familiar SQL without the need for a database administrator.

https://cloud.google.com/bigquery/

DATA WAREHOUSING SOLUTION ARCHITECTURE



When to use Google BigQuery

- BigQuery is not a replacement to existing technologies (RDBBMS, OLAP Services, etc.) but it complements them very well.
- Analytics can be made faster and at a larger scale on BigQuery.
- BigQuery pricing is pay-per-usage. Large processing will result in high cost.

Skills and Knowledge required for using BigQuery

- SQL Programming
- Data Modeling and Data Processing

Accessing Google BigQuery

1. Go to https://bigquery.cloud.google.com/
2. Choose to your project:
   nida-workshop
3. Click data set **supermarket** to show list of tables

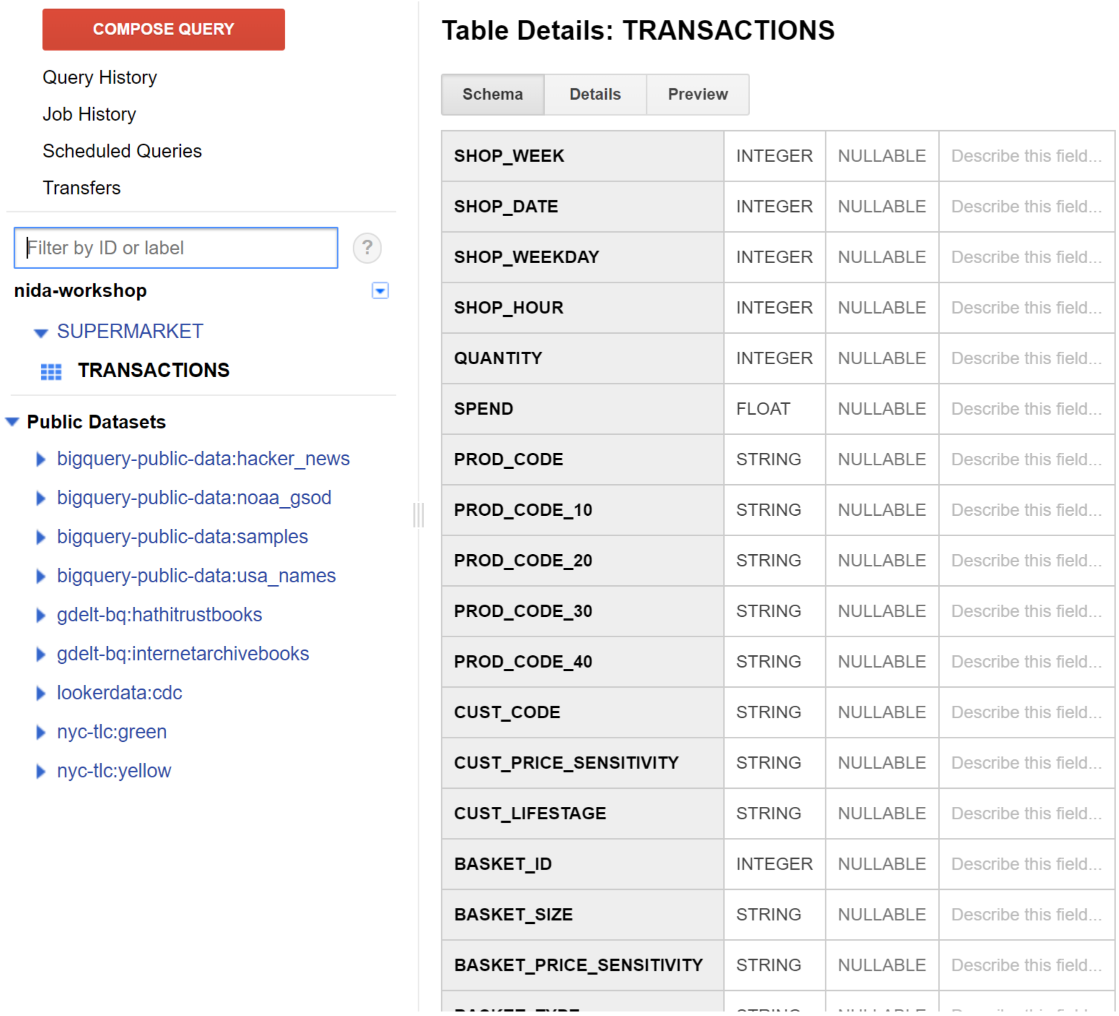


Table Details: TRANSACTIONS

SHOP_WEEK	INTEGER	NULLABLE	Describe this field...
SHOP_DATE	INTEGER	NULLABLE	Describe this field...
SHOP_WEEKDAY	INTEGER	NULLABLE	Describe this field...
SHOP_HOUR	INTEGER	NULLABLE	Describe this field...
QUANTITY	INTEGER	NULLABLE	Describe this field...
SPEND	FLOAT	NULLABLE	Describe this field...
PROD_CODE	STRING	NULLABLE	Describe this field...
PROD_CODE_10	STRING	NULLABLE	Describe this field...
PROD_CODE_20	STRING	NULLABLE	Describe this field...
PROD_CODE_30	STRING	NULLABLE	Describe this field...
PROD_CODE_40	STRING	NULLABLE	Describe this field...
CUST_CODE	STRING	NULLABLE	Describe this field...
CUST_PRICE_SENSITIVITY	STRING	NULLABLE	Describe this field...
CUST_LIFESTAGE	STRING	NULLABLE	Describe this field...
BASKET_ID	INTEGER	NULLABLE	Describe this field...
BASKET_SIZE	STRING	NULLABLE	Describe this field...
BASKET_PRICE_SENSITIVITY	STRING	NULLABLE	Describe this field...

DATA SET DETAILS

Column Name	Description	Type	Sample Values
shop_week	Identifies the week of the basket	Char	Format is YYYYWW where the first 4 characters identify the fiscal year and the other two characters identify the specific week within the year (e.g. 200735). Being the fiscal year, the first week doesn’t start in January. (See time.csv file for start/end dates of each week)
shop_date	Date when shopping has been made. Date is specified in the yyyymmdd format	Char	20060413, 20060412
shop_weekday	Identifies the day of the week	Num	1=Sunday, 2=Monday, …, 7=Saturday
shop_hour	Hour slot of the shopping	Num	0=00:00 …23=23:00 -00:59, 1=01:00 -23:59 -01:59,

DATA SET DETAILS

Column Name	Description	Type	Sample Values
Quantity	Number of items of the same product bought in this basket	Num	Integer number
spend	Spend associated to the items bought	Num	Number with two decimal digits
prod_code	Product Code	Char	PRD0900001, PRD0900003
prod_code_10	Product Hierarchy Level 10 Code	Char	CL00072, CL00144
prod_code_20	Product Hierarchy Level 20 Code	Char	DEP00021, DEP00051
prod_code_30	Product Hierarchy Level 30 Code	Char	G00007, G00015
prod_code_40	Product Hierarchy Level 40 Code	Char	D00002, D00003

DATA SET DETAILS

Column Name	Description	Type	Sample Values
cust_code	Customer Code	Char	CUST0000001624, CUST0000001912
cust_price_sensitivity	Customer's Price Sensitivity	Char	LA=Less Affluent, MM=Mid Market, UM=Up Market, XX=unclas

cust_lifestage	Customer's Lifestage	Char	YA=Young Adults, OA=Older Adults,
			YF=Young Families,
			OF=Older Families,
			PE=Pensioners, OT=Other,
			XX=unclassified

basket_id	Basket ID. All items in a basket share the same basket_id value.	Num	994100100000020, 994100100000344
basket_size	Basket size	Char	L=Large, M=Medium, S=Small
basket_price_sensitivity	Basket price sensitivity	Char	LA=Less Affluent, MM=Mid Market, UM=Up Market, XX=unclassified

DATA SET DETAILS

Column Name	Description	Type	Sample Values
basket_type	Basket type	Char	Small Shop, Top Up, Full Shop, XX
basket_dominant_mission	Shopping dominant mission	Char	Fresh, Grocery, Mixed, Non Food, XX
store_code	Store Code	Char	STORE00001, STORE00002
store_format	Format of the Store	Char	LS, MS, SS, XLS
store_region	Region the store belongs to	Char	E02, W01, E01, N03

Getting started with SQL

SQL stands for Structured Query Language.
SQL lets you access and manipulate databases.

Column

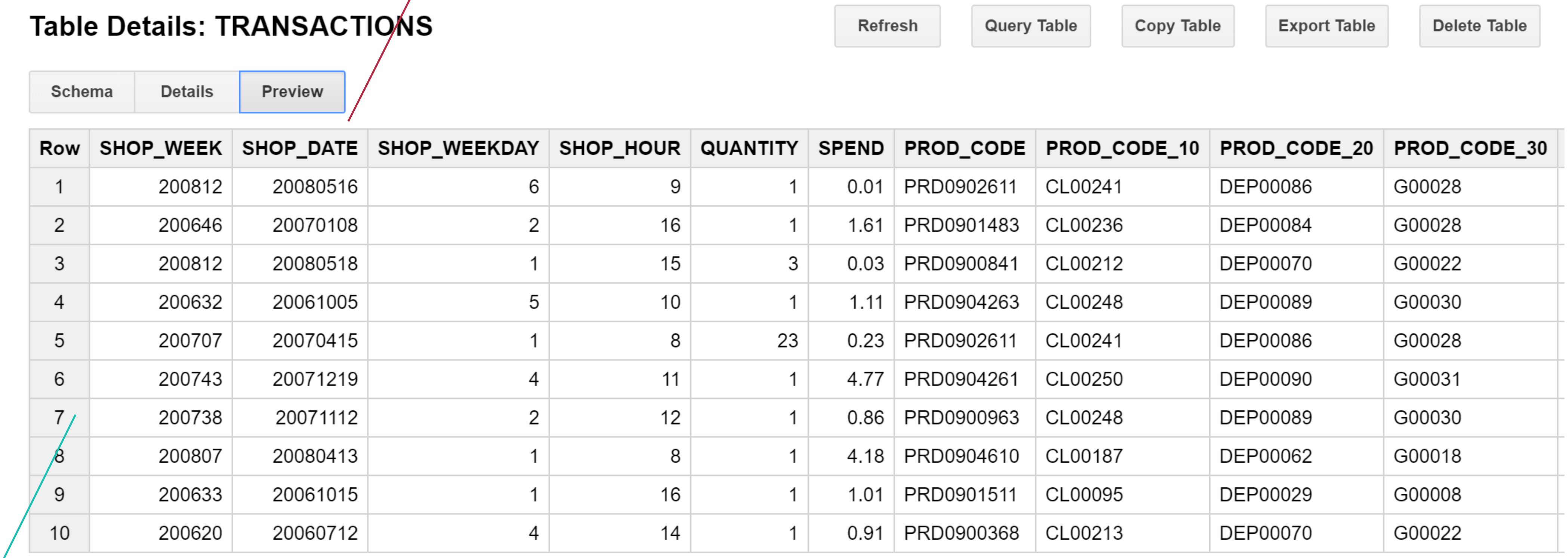
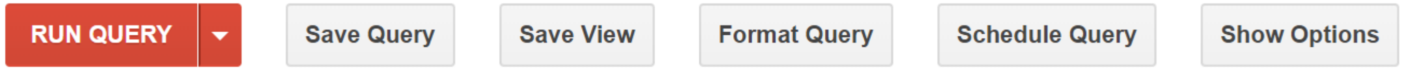


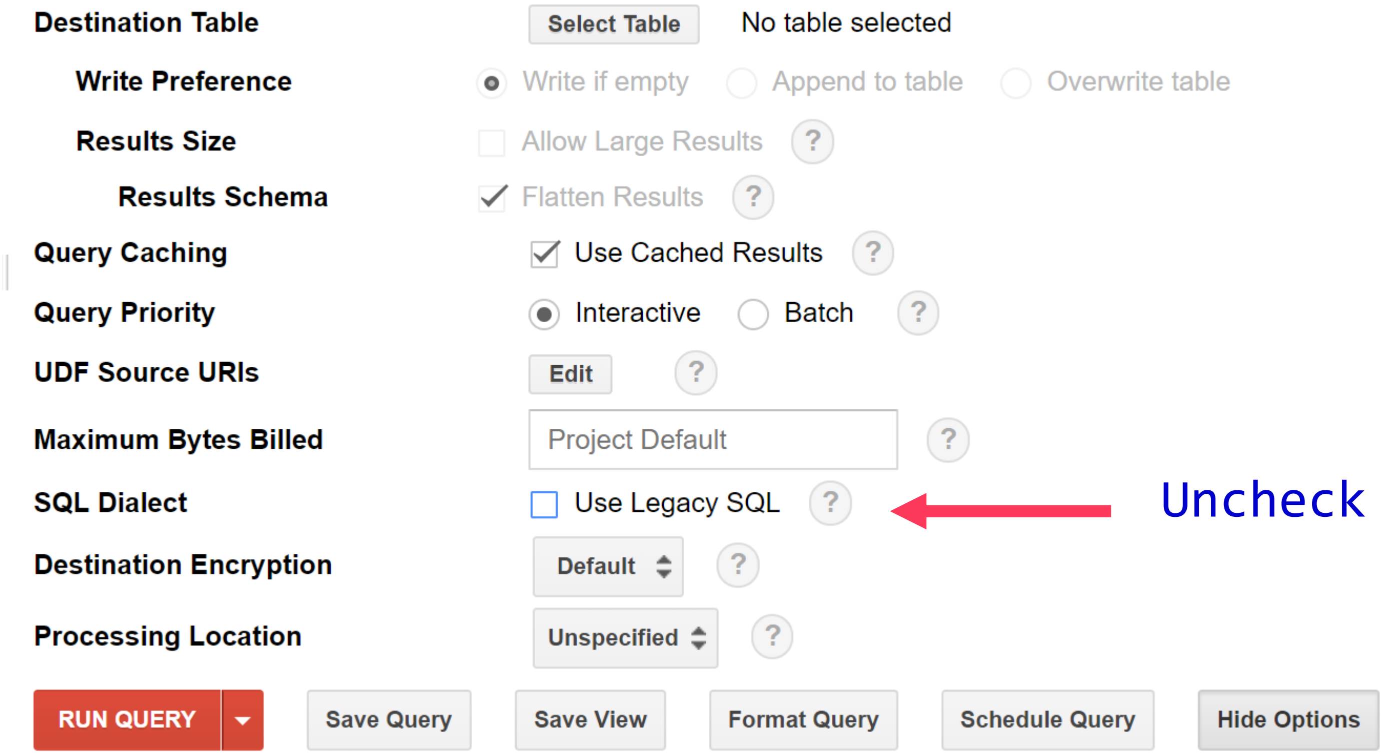
Table Details: TRANSACTIONS

Row	SHOP_WEEK	SHOP_DATE	SHOP_WEEKDAY	SHOP_HOUR	QUANTITY	SPEND	PROD_CODE	PROD_CODE_10	PROD_CODE_20	PROD_CODE_30
1	200812	20080516	6	9	1	0.01	PRD0902611	CL00241	DEP00086	G00028
2	200646	20070108	2	16	1	1.61	PRD0901483	CL00236	DEP00084	G00028
3	200812	20080518	1	15	3	0.03	PRD0900841	CL00212	DEP00070	G00022
4	200632	20061005	5	10	1	1.11	PRD0904263	CL00248	DEP00089	G00030
5	200707	20070415	1	8	23	0.23	PRD0902611	CL00241	DEP00086	G00028
6	200743	20071219	4	11	1	4.77	PRD0904261	CL00250	DEP00090	G00031
7	200738	20071112	2	12	1	0.86	PRD0900963	CL00248	DEP00089	G00030
8	200807	20080413	1	8	1	4.18	PRD0904610	CL00187	DEP00062	G00018
9	200633	20061015	1	16	1	1.01	PRD0901511	CL00095	DEP00029	G00008
10	200620	20060712	4	14	1	0.91	PRD0900368	CL00213	DEP00070	G00022

Row

Using Standard SQL on BigQuery




Destination Table — Select Table — No table selected

Write Preference — ● Write if empty ○ Append to table ○ Overwrite table

Results Size — □ Allow Large Results ?

Results Schema — ☑ Flatten Results ?

Query Caching — ☑ Use Cached Results ?

Query Priority — ● Interactive ○ Batch ?

UDF Source URIs — Edit ?

Maximum Bytes Billed — Project Default ?

SQL Dialect — □ Use Legacy SQL ?

Destination Encryption — Default ?

Processing Location — Unspecified ?

Uncheck the box

SELECT statement

Standard syntax

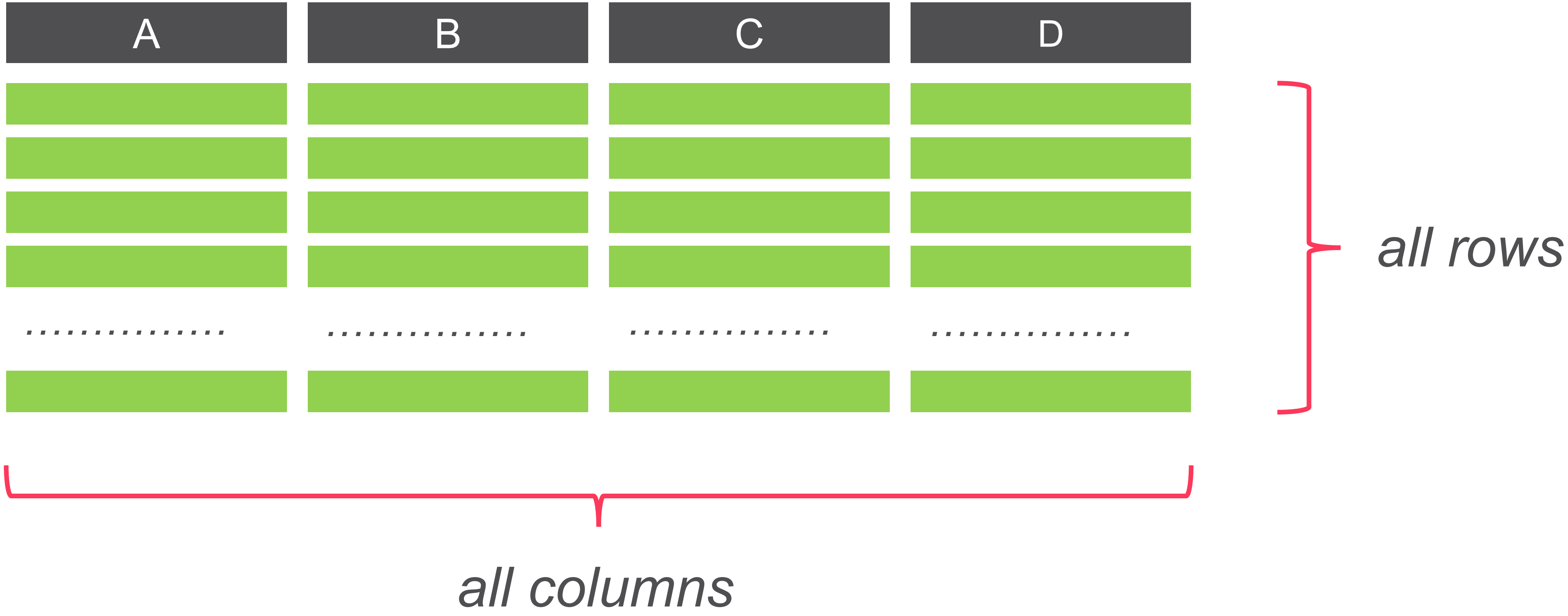
```
SELECT column1, column2, ...
FROM table_name
```

Easy query

```
SELECT *
FROM table_name
LIMIT 100
```

Lazy Query

```sql
SELECT *
FROM table_name
```

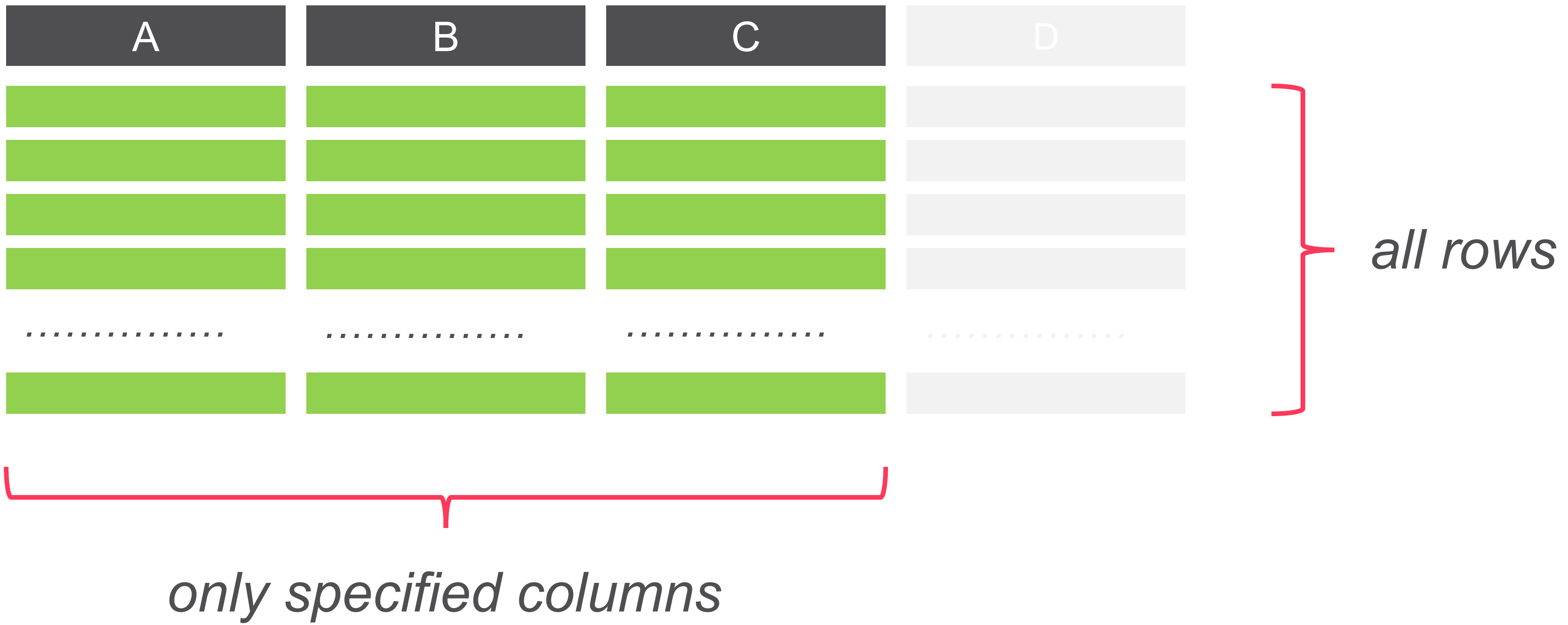


all rows

all columns

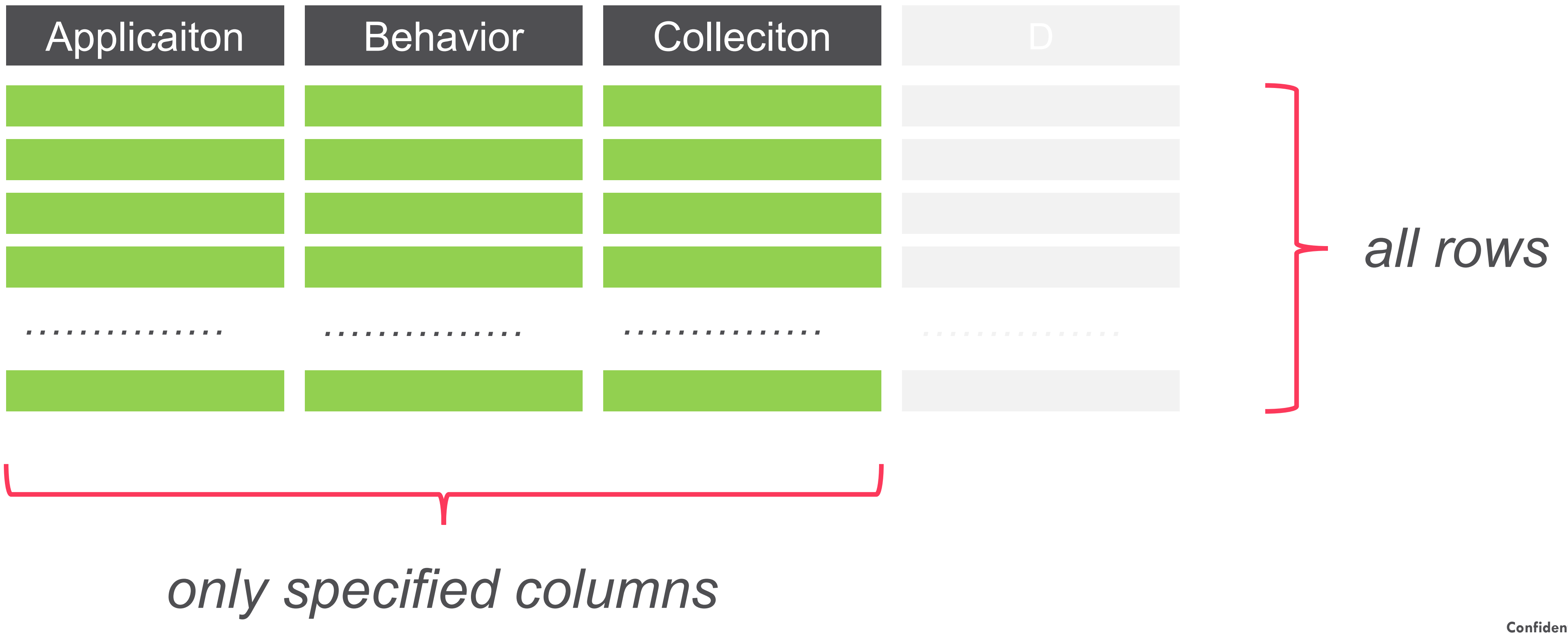
Query with Specific Columns

```
SELECT A, B, C
FROM table_name
```



Column Alias

```sql
SELECT A as Applicaiton, B as Behavior, C as Collection
FROM table_name
```



Applicaiton

Behavior

Colleciton

D

all rows

only specified columns

Exercise

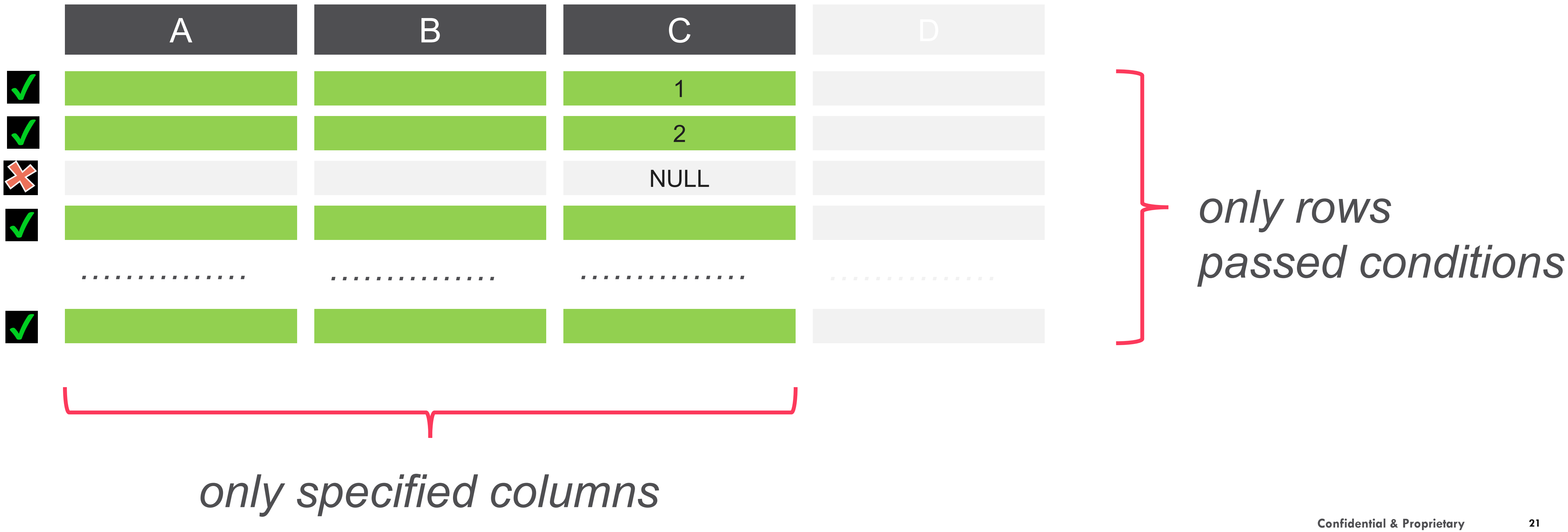
- List all records from TRANSACTIONS table

- List only **CUST_CODE, PROD_CODE, QUANTITY,** and **SPEND** for all records from TRANSACTIONS table

- List only **CUST_CODE, PROD_CODE, QUANTITY,** and **SPEND** for all records from TRANSACTIONS table rename all columns

NULL

```sql
SELECT A, B, C
FROM table_name
WHERE C IS NOT NULL
```



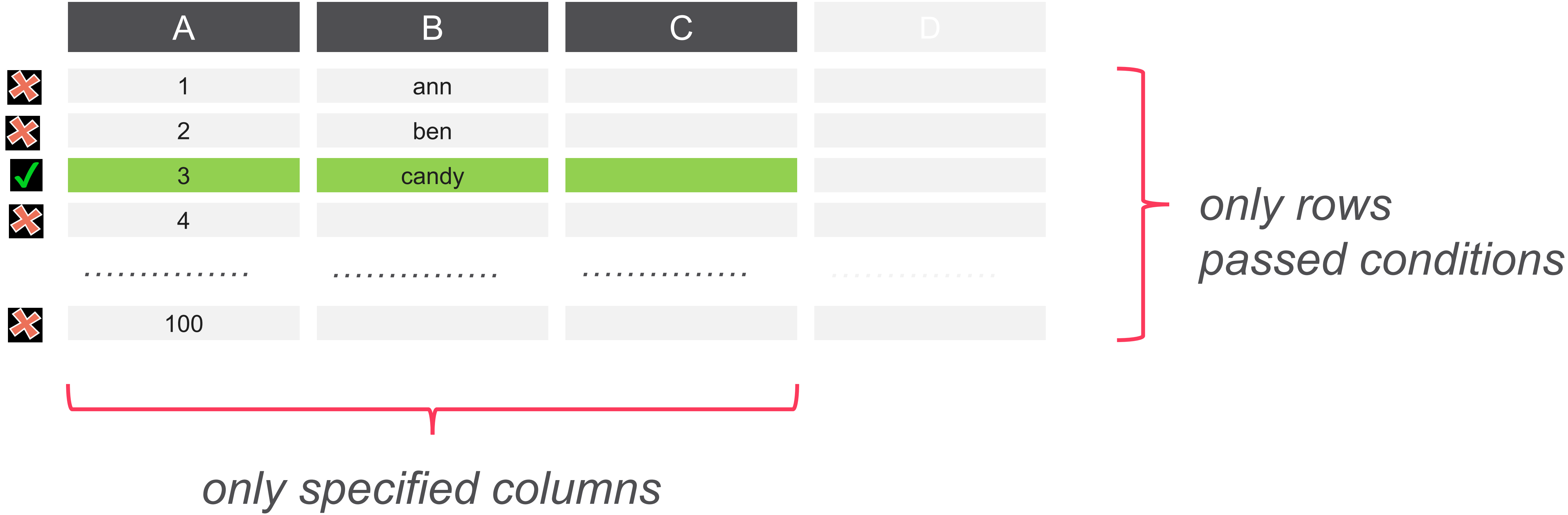
	A	B	C	D
✓			1	
✓			2	
✗			NULL	
✓				
	....	....	....	....
✓				

only rows passed conditions

only specified columns

WHERE multiple conditions

```
SELECT A, B, C
FROM table_name
WHERE A >= 3
AND B LIKE '%an%'
```



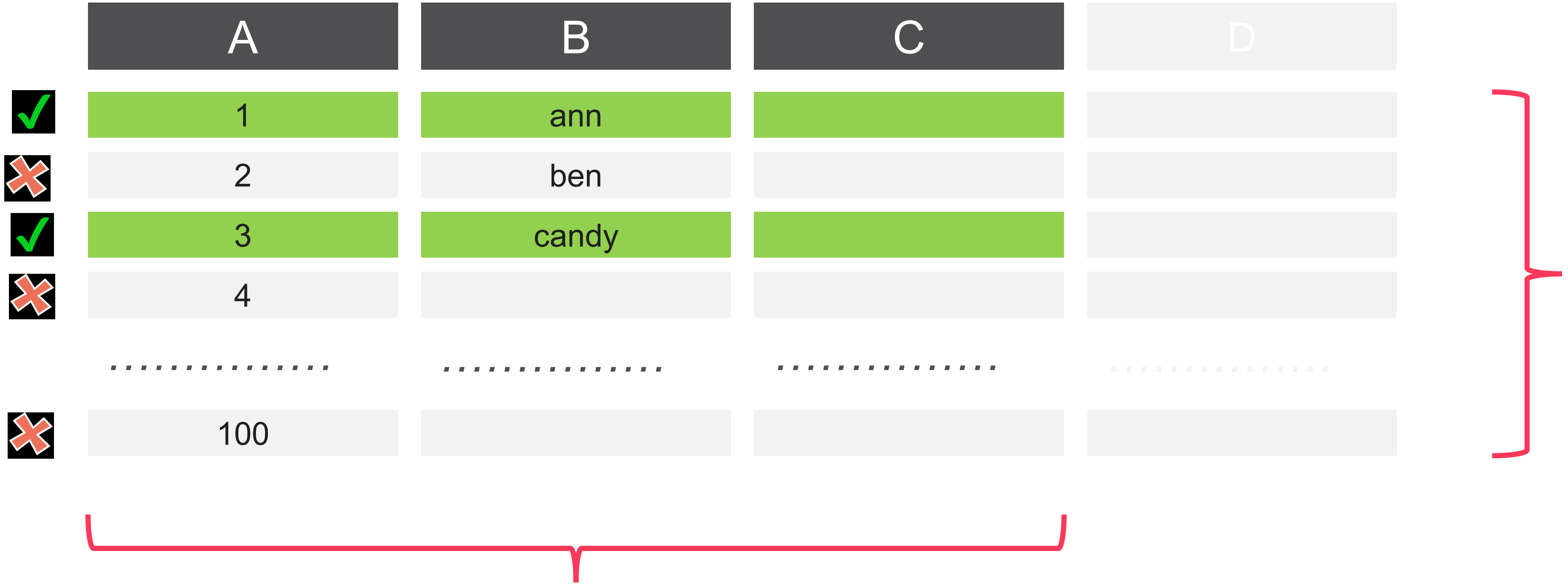
	A	B	C	D
✗	1	ann		
✗	2	ben		
✓	3	candy		
✗	4			
	....	....	....	....
✗	100			

only rows passed conditions

only specified columns

IN

```
SELECT A, B, C
FROM table_name
WHERE B IN ('ann', 'candy')
```



	A	B	C	D
✓	1	ann		
✗	2	ben		
✓	3	candy		
✗	4			
	....	....	....	....
✗	100			

only rows passed conditions

only specified columns

Exercise

- List **BASKET_ID** from **TRANSACTIONS** table where **BASKET_ID** is NULL

- List all records from **TRANSACTIONS** table where **SHOP_WEEKDAY** is 7 and **SHOP_HOUR** is greater than 18

- List all records from **TRANSACTIONS** table where **BASKET_DOMINANT_MISSION** is Fresh, Grocery, or Nonfood

8:30 AM - Marketing System meeting
PM - Review PDA user manual
John again to arrange meeting
Sydney
CASHBOOK
Pay Invoice
Card Type
Name on card
Card number
Expiry date
Zip / Poctal code

Basic Data Processing

SQL Arithmetic Operators

Operator	Description
+	Add
-	Subtract
*	Multiply
/	Divide
%	Modulo

IF

```
SELECT A, B, IF(A>=B, 'greater' , 'lesser') AS RESULTS
FROM table_name
```

A	B	RESULTS
9	2	greater
7	4	greater
5	6	lesser
3	8	lesser
1	10	lesser

CASE WHEN

```
SELECT A, B,
CASE
     WHEN A>B THEN 'greater'
     WHEN A=B THEN 'equal'
     WHEN A<B THEN 'lesser'
     ELSE 'N/A'
END AS RESULTS
FROM table_name
```

A	B	RESULTS
9	2	greater
7	4	greater
5	6	lesser
3	8	lesser
1	10	lesser

Math/Numeric Functions

Function	Description
ABS	Returns the absolute value of a number
ACOS	Returns the arc cosine of a number
ASIN	Returns the arc sine of a number
ATAN	Returns the arc tangent of a number
ATN2	Returns the arc tangent of n and m
CEILING	Returns the smallest integer value that is >= a number
COS	Returns the cosine of a number
COT	Returns the cotangent of a number
DEGREES	Converts a value in radians to degrees
EXP	Returns e raised to the power of a specified number
FLOOR	Returns the largest integer value that is <= to a number
LOG	Returns the natural logarithm of a number, or the logarithm of a number to a specified base
LOG10	Returns the natural logarithm of a number to base 10
PI	Returns the value of PI
POWER	Returns the value of a number raised to the power of another number
RADIANS	Converts a degree value into radians
RAND	Returns a random number
ROUND	Rounds a number to a specified number of decimal places
SIGN	Returns the sign of a number
SIN	Returns the sine of a number
SQRT	Returns the square of a number
SQUARE	Returns the square of a number
TAN	Returns the tangent of a number

Date Functions

Function	Description
CURRENT_TIMESTAMP	Returns the current date and time
DATEADD	Adds a time/date interval to a date and then returns the date
DATEDIFF	Returns the difference between two dates
DATEFROMPARTS	Returns a date from the specified parts (year, month, and day values)
DATENAME	Returns a specified part of a date (as string)
DATEPART	Returns a specified part of a date (as integer)
DAY	Returns the day of the month for a given date
GETDATE	Returns the current database system date and time
GETUTCDATE	Returns the current database system UTC date and time
ISDATE	Returns 1 if the expression is a valid date, otherwise 0
MONTH	Returns the month part for a given date (a number from 1 to 12)
SYSDATETIME	Returns the date and time of the SQL Server
YEAR	Returns the year part for a given date

Conversion Functions

Function	Description
CAST	Converts a value (of any type) into a specified datatype
CONVERT	Converts a value (of any type) into a specified datatype

Date Functions

Function	Description
COALESCE	Returns the first non-null value in a list
CURRENT_USER	Returns the name of the current user in the SQL Server database
ISNULL	Return a specified value if the expression is NULL, otherwise return the expression
ISNUMERIC	Tests whether an expression is numeric
NULLIF	Returns NULL if two expressions are equal
SESSION_USER	Returns the name of the current user in the SQL Server database
SESSIONPROPERTY	Returns the session settings for a specified option
SYSTEM_USER	Returns the login name for the current user
USER_NAME	Returns the database user name based on the specified id

Exercise

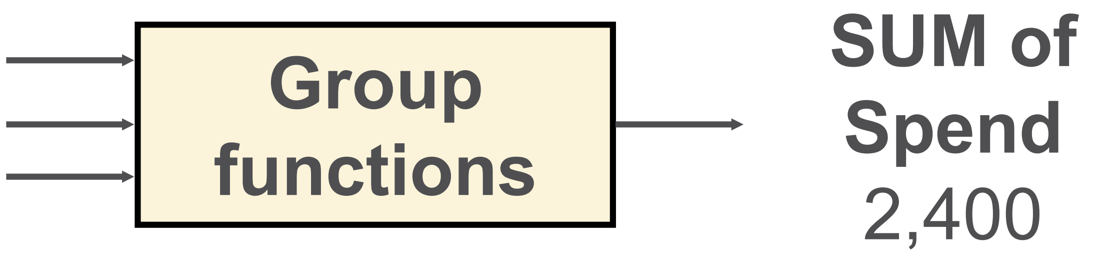
Use TRANSACTIONS table

- Classify **SHOP_HOUR** to be Morning, Afternoon, and Evening
- Classify **SHOP_WEEKDAY** into day, evening, and night

Group Functions

- Group functions operate on sets of rows to give one result per group.

Charge	Area	Spend
Pre-paid	BKK	150
Pre-paid	BKK	250
Pre-paid	Non-BKK	200
Post-paid	BKK	400
Post-paid	BKK	800
Post-paid	Non-BKK	600



Group functions

SUM of Spend 2,400

Types of Group Functions

Function	Description	Example

COUNT( )	Number of rows (*,including duplicate rows with null)	COUNT(*)
		COUNT(CUST_CODE)

SUM( )	Sum values of exp , ignoring null values	SUM(SPEND)
MAX( )	Maximum value of exp , ignoring null values	MAX(SPEND)
MIN( )	Minimum value of exp , ignoring null values	MIN(SPEND)
AVG( )	Average value of exp , ignoring null values	AVG(SPEND)

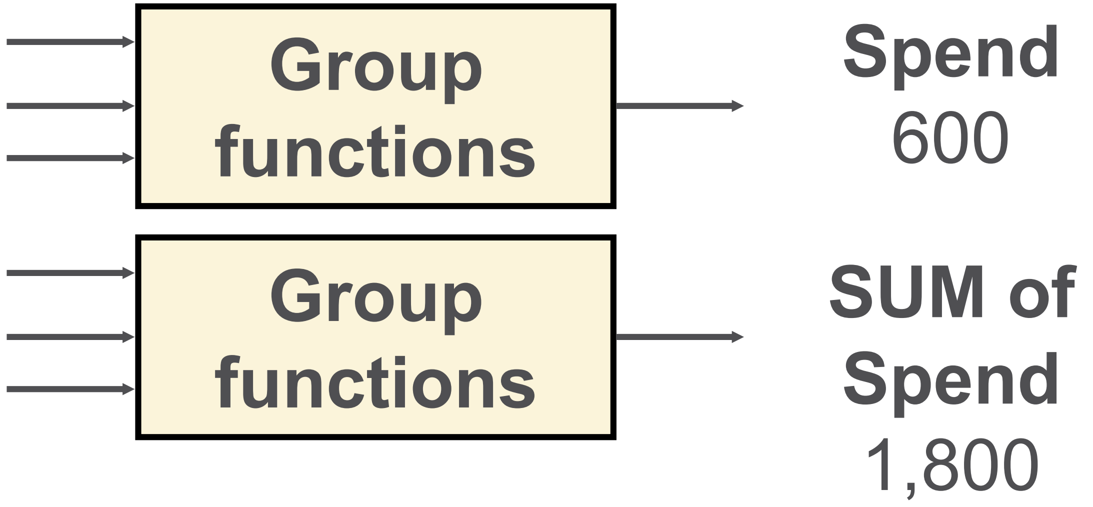
Group Functions: Syntax

```
SELECT column1, column2, group_function(column), …
FROM table_name
WHERE conditions
GROUP BY column_name
ORDER BY column_name
```

Creating Groups of Data

```
SELECT Charge, SUM(Spend)
FROM table_name
GROUP BY Charge
```

Charge	Area	Spend
Pre-paid	BKK	150
Pre-paid	BKK	250
Pre-paid	Non-BKK	200
Post-paid	BKK	400
Post-paid	BKK	800
Post-paid	Non-BKK	600



Group functions

SUM of Spend 600

Group functions

SUM of Spend 1,800

Restricting Group Results with the HAVING Clause

- When you use the **HAVING** clause, the SQL command restricts groups as follows:
  1. Rows are grouped.
  2. The group function is applied.
  3. Groups matching the **HAVING** clause are displayed.

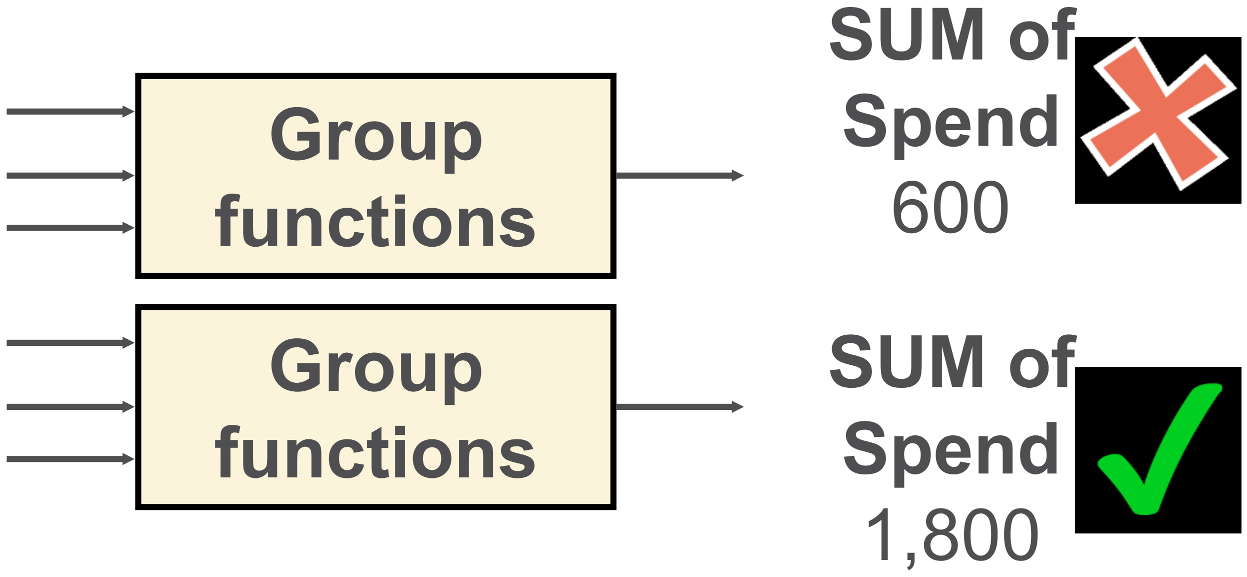
```
SELECT column1, column2, group_function(column), …
FROM table_name
WHERE conditions
GROUP BY column_name
HAVING   group_condition
ORDER BY column_name
```

- Having : deal with results of Group function
- Where: deal with real value in table

Using the **HAVING** Clause

```
SELECT Charge, SUM(Spend)
FROM table_name
GROUP BY Charge
HAVING SUM(Spend) > 1,000
```

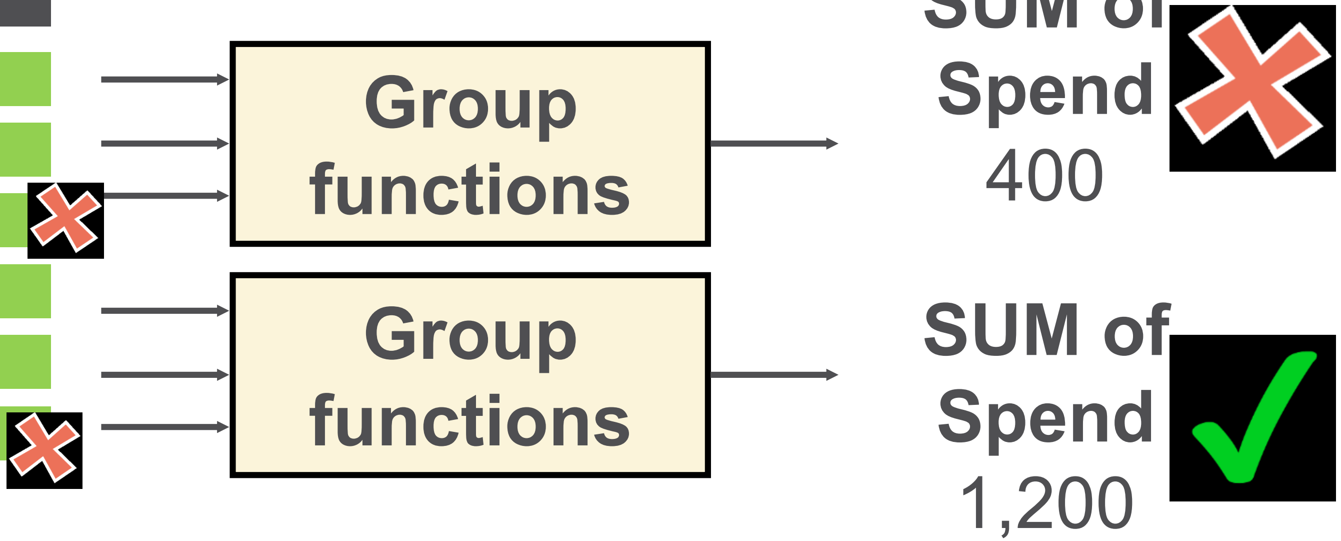
Charge	Area	Spend
Pre-paid	BKK	150
Pre-paid	BKK	250
Pre-paid	Non-BKK	200
Post-paid	BKK	400
Post-paid	BKK	800
Post-paid	Non-BKK	600



Using the **HAVING** and **WHERE** Clause

```
SELECT Charge, SUM(Spend)
FROM table_name
WHERE AREA = 'BKK'
GROUP BY Charge
HAVING SUM(Spend) > 1,000
```

Charge	Area	Spend
Pre-paid	BKK	150
Pre-paid	BKK	250
Pre-paid	Non-BKK	200
Post-paid	BKK	400
Post-paid	BKK	800
Post-paid	Non-BKK	600



Customer Single View

- Definition:
- an accessible and consistent set of information about how a customer has interacted with your company, including what they have bought, their personal data, opinions and feedback

- Benefits:
- understand how purchases, interactions, and behaviors of customers will drive future actions

https://blogs.oracle.com/marketingcloud/why-a-single-view-of-your-customer-is-vital-for-success

Key Elements of Customer Single View

Transactional Data

- Past/Current/Pending transactions
- Communication history
- Interaction history

Demographic Profile

- Age, Gender
- Geography
- Socioeconomic status
- Role
- Life event attributes

Behavioral Patterns

- Product preferences
- Lifestyle

https://www.forbes.com/sites/forbesinsights/2017/11/10/a-single-view-of-the-customer-is-the-foundation-for-an-effective-retail-digital-customer-experience/#564640774398

How to build Customer Single View

Journey

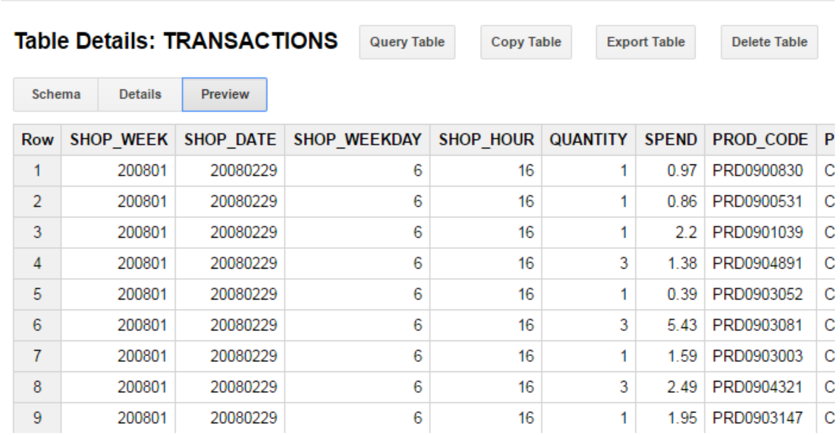
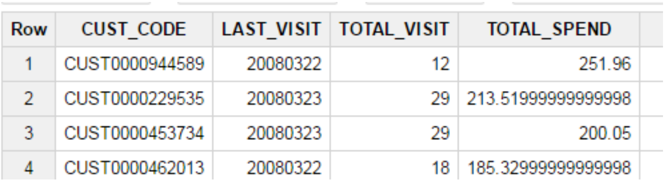


Table Details: TRANSACTIONS

Row	SHOP_WEEK	SHOP_DATE	SHOP_WEEKDAY	SHOP_HOUR	QUANTITY	SPEND	PROD_CODE	P
1	200801	20080229	6	16	1	0.97	PRD0900830	C
2	200801	20080229	6	16	1	0.86	PRD0900531	C
3	200801	20080229	6	16	1	2.2	PRD0901039	C
4	200801	20080229	6	16	3	1.38	PRD0904891	C
5	200801	20080229	6	16	1	0.39	PRD0903052	C
6	200801	20080229	6	16	3	5.43	PRD0903081	C
7	200801	20080229	6	16	1	1.59	PRD0903003	C
8	200801	20080229	6	16	3	2.49	PRD0904321	C
9	200801	20080229	6	16	1	1.95	PRD0903147	C

Transactions



Row	CUST_CODE	LAST_VISIT	TOTAL_VISIT	TOTAL_SPEND
1	CUST0000944589	20080322	12	251.96
2	CUST0000229535	20080323	29	213.51999999999998
3	CUST0000453734	20080323	29	200.05
4	CUST0000462013	20080322	18	185.32999999999998

Single Record

Exercise

Use TRANSACTIONS table

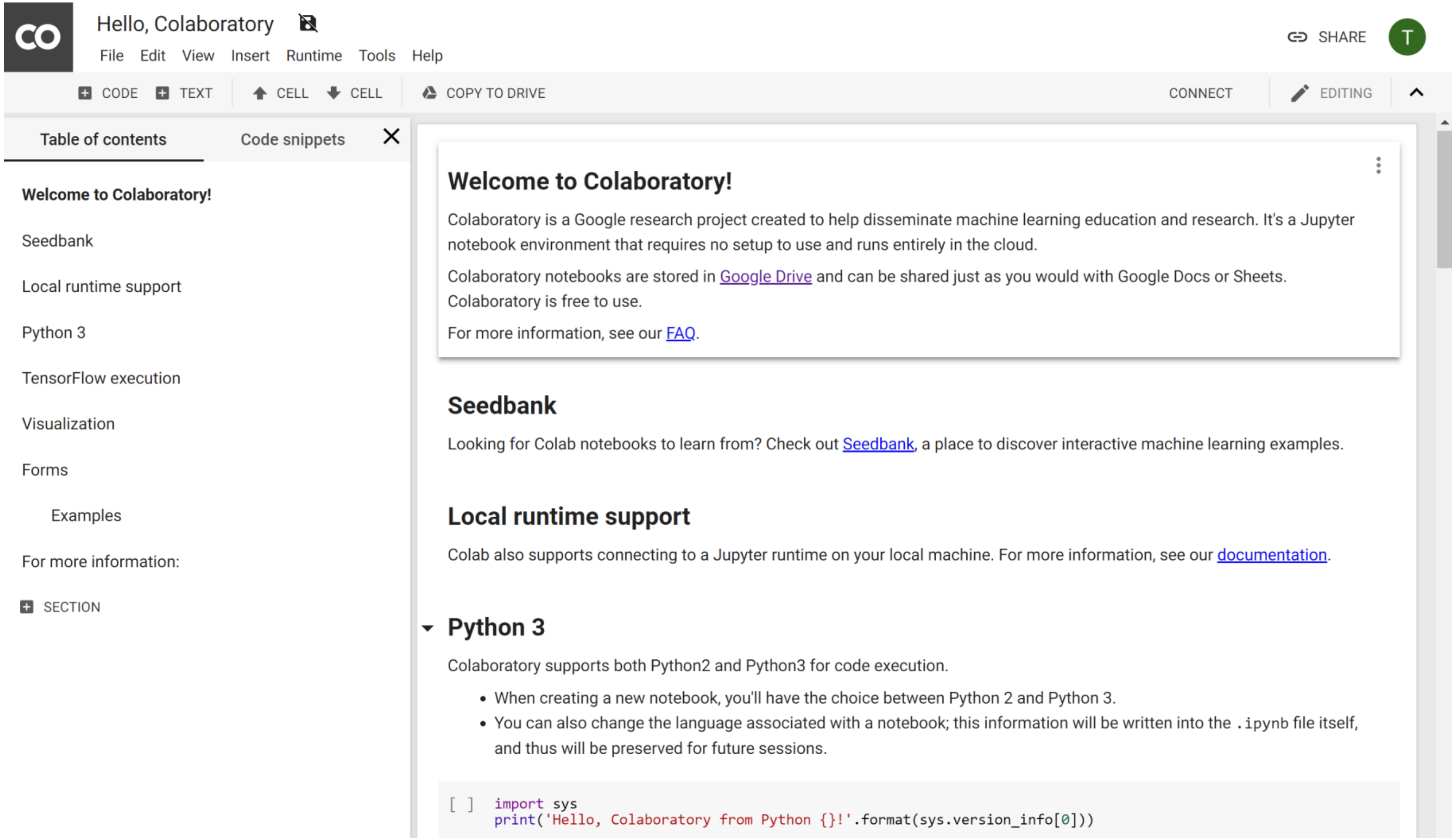
- Create CUSTOMER SINGLE VIEW aggregating behaviors of an individual customer including
  - Total Number of Baskets (Count Distinct of BASKET_ID)
  - Total Number of Products (Count Distinct of PROD_CODE)
  - Total Sales (Sum of Spend)

Exploratory Data Analysis

Understanding Your Data

Introduction to Python on Colab

https://colab.research.google.com



Welcome to Colaboratory!

Seedbank

Local runtime support

Python 3

TensorFlow execution

Visualization

Forms

Examples

For more information:

SECTION

Welcome to Colaboratory!

Colaboratory is a Google research project created to help disseminate machine learning education and research. It's a Jupyter notebook environment that requires no setup to use and runs entirely in the cloud.

Colaboratory notebooks are stored in Google Drive and can be shared just as you would with Google Docs or Sheets. Colaboratory is free to use.

For more information, see our FAQ.

Seedbank

Looking for Colab notebooks to learn from? Check out Seedbank, a place to discover interactive machine learning examples.

Local runtime support

Colab also supports connecting to a Jupyter runtime on your local machine. For more information, see our documentation.

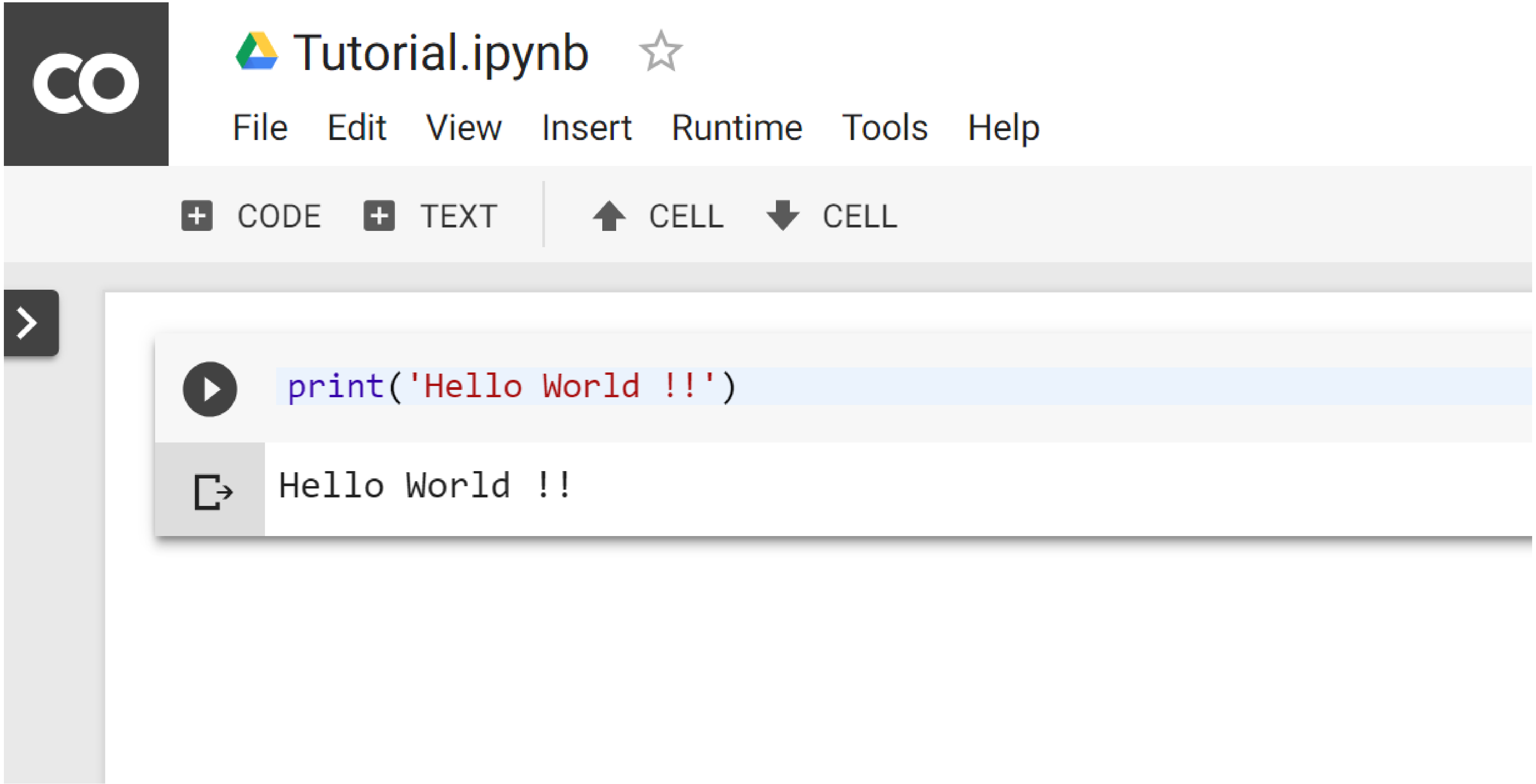
Python 3

Colaboratory supports both Python2 and Python3 for code execution.

- When creating a new notebook, you'll have the choice between Python 2 and Python 3.
- You can also change the language associated with a notebook; this information will be written into the `.ipynb` file itself, and thus will be preserved for future sessions.

```
[ ]  import sys
     print('Hello, Colaboratory from Python {}!'.format(sys.version_info[0]))
```

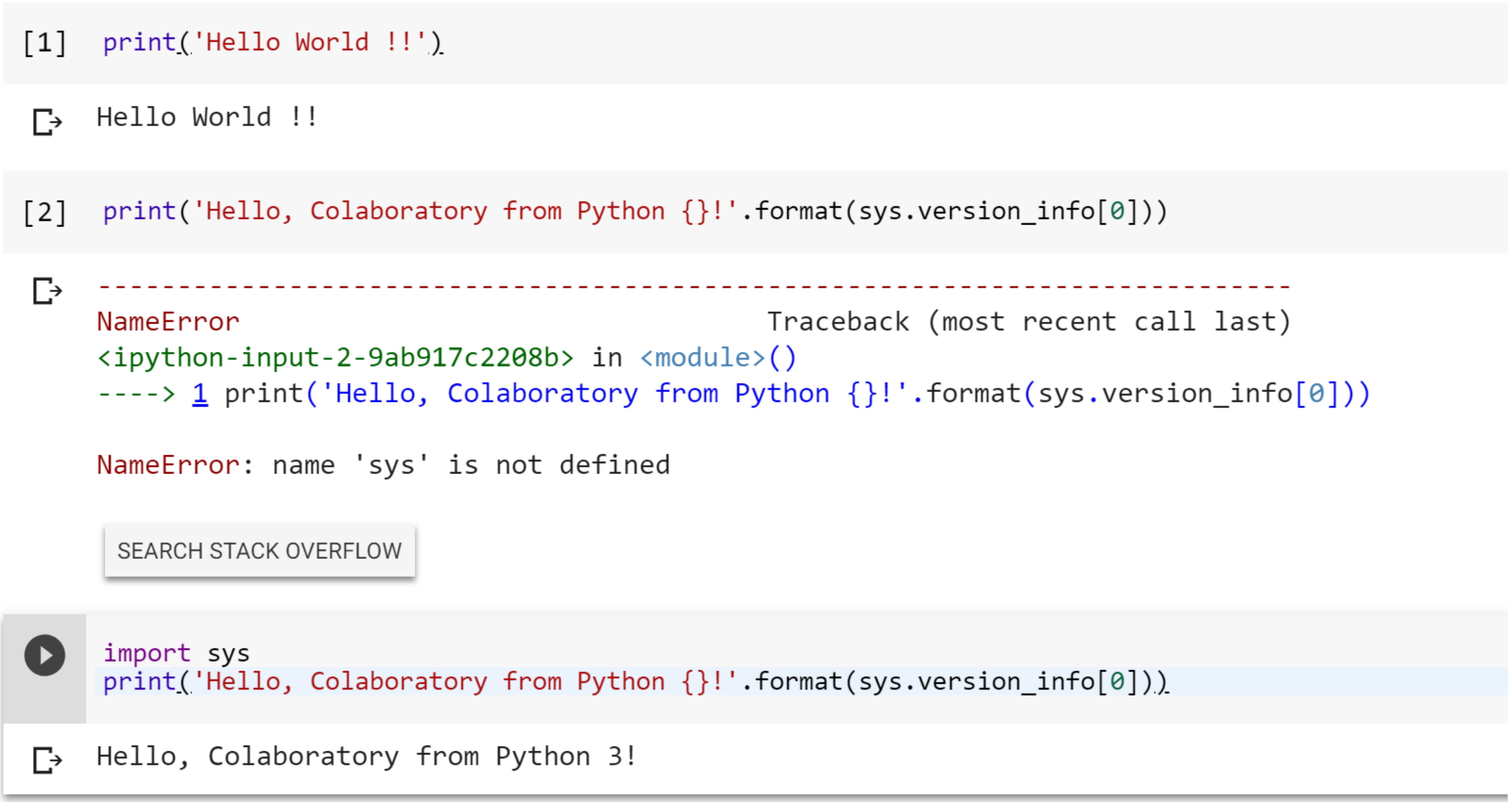
Hello World !!



```
print('Hello World !!')
```

```
Hello World !!
```

Extending your python capabilities with import



```
[1]  print('Hello World !!')

     Hello World !!
```

```
[2]  print('Hello, Colaboratory from Python {}!'.format(sys.version_info[0]))

     ---------------------------------------------------------------------------
     NameError                                 Traceback (most recent call last)
     <ipython-input-2-9ab917c2208b> in <module>()
     ----> 1 print('Hello, Colaboratory from Python {}!'.format(sys.version_info[0]))

     NameError: name 'sys' is not defined
```

```
import sys
print('Hello, Colaboratory from Python {}!'.format(sys.version_info[0]))

Hello, Colaboratory from Python 3!
```

Popular Packages for Data Science

Core Libraries & Statistics

▪ NumPy	http://www.numpy.org/
▪ SciPy	https://scipy.org/scipylib/
▪ Pandas	https://pandas.pydata.org/
▪ StatsModels	http://www.statsmodels.org/devel/

Visualization

▪ Matplotlib	https://matplotlib.org/index.html
▪ Seaborn	https://seaborn.pydata.org/
▪ Ploty	https://plot.ly/python/
▪ Bokeh	https://bokeh.pydata.org/en/latest/
▪ Pydot	https://pypi.org/project/pydot/

https://medium.com/activewizards-machine-learning-company/top-20-python-libraries-for-data-science-in-2018-2ae7d1db8049

Popular Packages for Data Science

Machine Learning

Scikit-learn	http://scikit-learn.org/stable/

Deep Learning

TensorFlow	https://www.tensorflow.org/
PyTorch	https://pytorch.org/
Keras	https://keras.io/

Data Scraping

Scrapy	https://scrapy.org/

https://medium.com/activewizards-machine-learning-company/top-20-python-libraries-for-data-science-in-2018-2ae7d1db8049

Authentication to Google BigQuery

from <module_name> import <name(s)>

An alternate form of the import statement allows individual objects from the module to be imported *directly into the caller's symbol table*:

```
from google.colab import auth
auth.authenticate_user()
```

https://realpython.com/python-modules-packages/

IMPORT other Packages

```
import pandas as pd
import seaborn as sns
```

PANDAS

Pandas is a Python library that provides high-level data structures and a vast variety of tools for analysis. The great feature of this package is the ability to translate rather complex operations with data into one or two commands.

SEABORN

Seaborn is essentially a higher-level API based on the matplotlib library. It contains more suitable default settings for processing charts. Also, there is a rich gallery of visualizations including some complex types like time series, jointplots, and violin diagrams.

https://medium.com/activewizards-machine-learning-company/top-20-python-libraries-for-data-science-in-2018-2ae7d1db8049

Read Data from BigQuery

```
project_id = 'nida-workshop'

sql = '''
SELECT CUST_CODE, SUM(SPEND) AS TOTAL_SALES,
COUNT(DISTINCT BASKET_ID) AS TOTAL_VISIT,
COUNT(DISTINCT PROD_CODE) AS TOTAL_PRODUCT
FROM `nida-workshop.SUPERMARKET.TRANSACTIONS`
WHERE CUST_CODE IS NOT NULL GROUP BY CUST_CODE
'''


df = pd.io.gbq.read_gbq(sql, project_id=project_id, verbose=False,
dialect='standard')
```

Showing the results from the Query

- **DataFrame** is a 2-dimensional labeled data structure with columns of potentially different types.
- **df.head**(n) shows the first n rows of **df**

```
df.head(5)
```

https://pandas.pydata.org/pandas-docs/version/0.22/dsintro.html

Aggregating data

- .mean() calculate mean

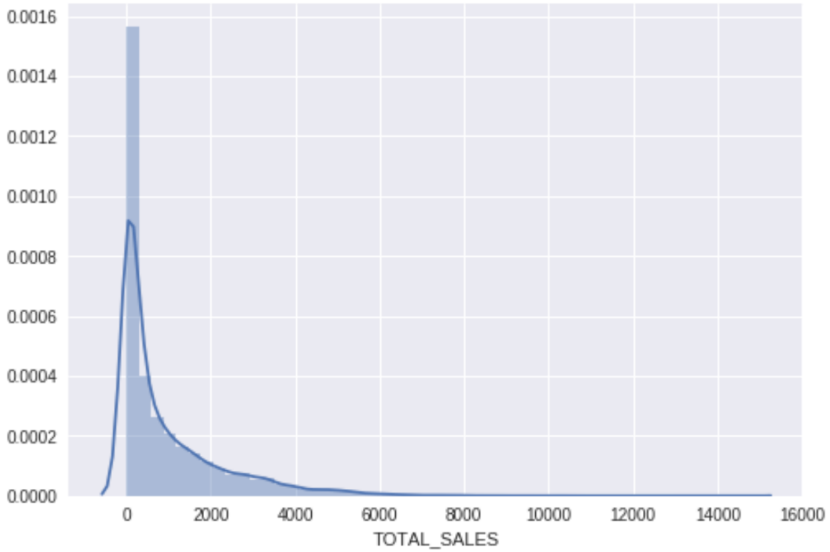
```
df.mean()
```

https://pandas.pydata.org/pandas-docs/version/0.22/dsintro.html

Distribution Plot

- distplot Plot the distribution
- df["TOTAL_SALES"] Select Column named TOTAL_SALES

```
sns.distplot(df["TOTAL_SALES"])
```



Distribution Plot

Loop through columns to plot distribution of each column

```
for i, col in enumerate(df.columns[1:]):
    sns.plt.figure(i)
    sns.distplot(df[col])
```

Other Plots

```
sns.lmplot(x='TOTAL_VISIT', y='TOTAL_SALES', data=df, fit_reg=False)

sns.pairplot(df)

sns.heatmap(df.corr(),annot=True)
```

CLUSTERING ALGORITHM

Segment Your Customers

Clustering Algorithms

- Clustering is

  the task of dividing the population or data points into a number of groups

  such that data points in the same groups are more similar to other data points in the same group than those in other groups.

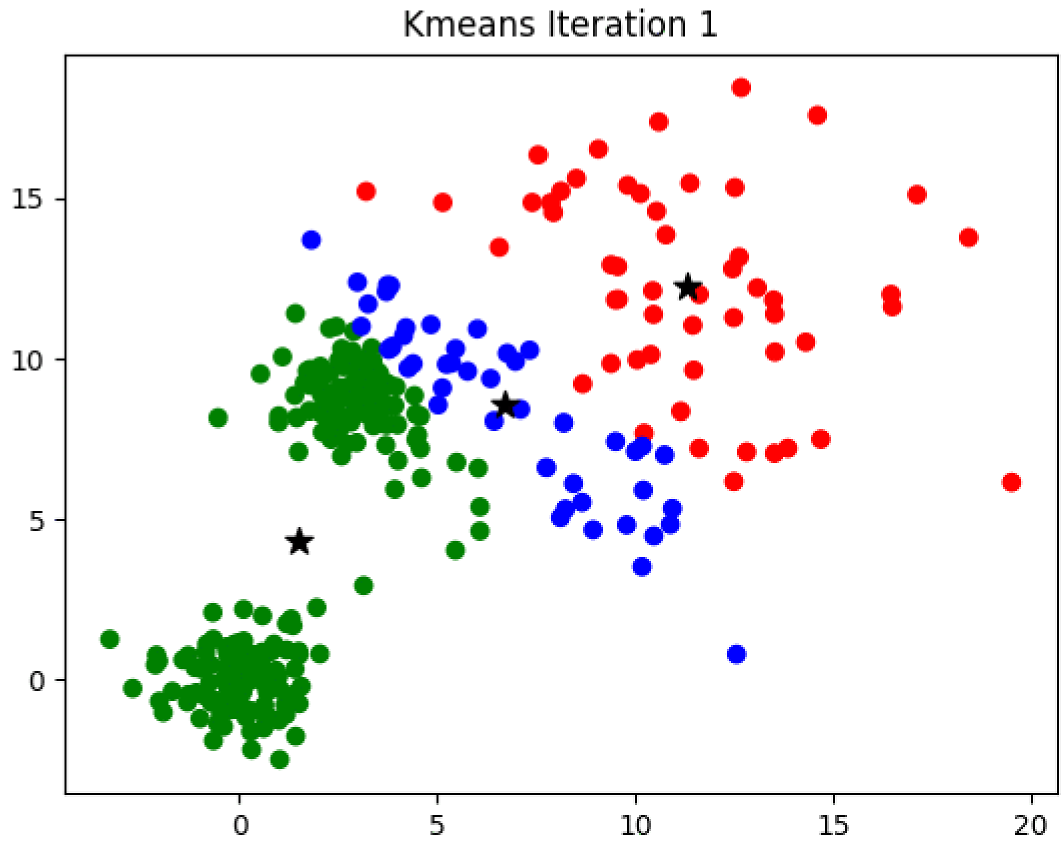
https://www.analyticsvidhya.com/blog/2016/11/an-introduction-to-clustering-and-different-methods-of-clustering/

Types of clustering algorithms

- **Connectivity models** – hierarchical clustering
- **Centroid models** – K-means clustering
- **Distribution models** – Expectation-maximization
- **Density Models** – DBSCAN

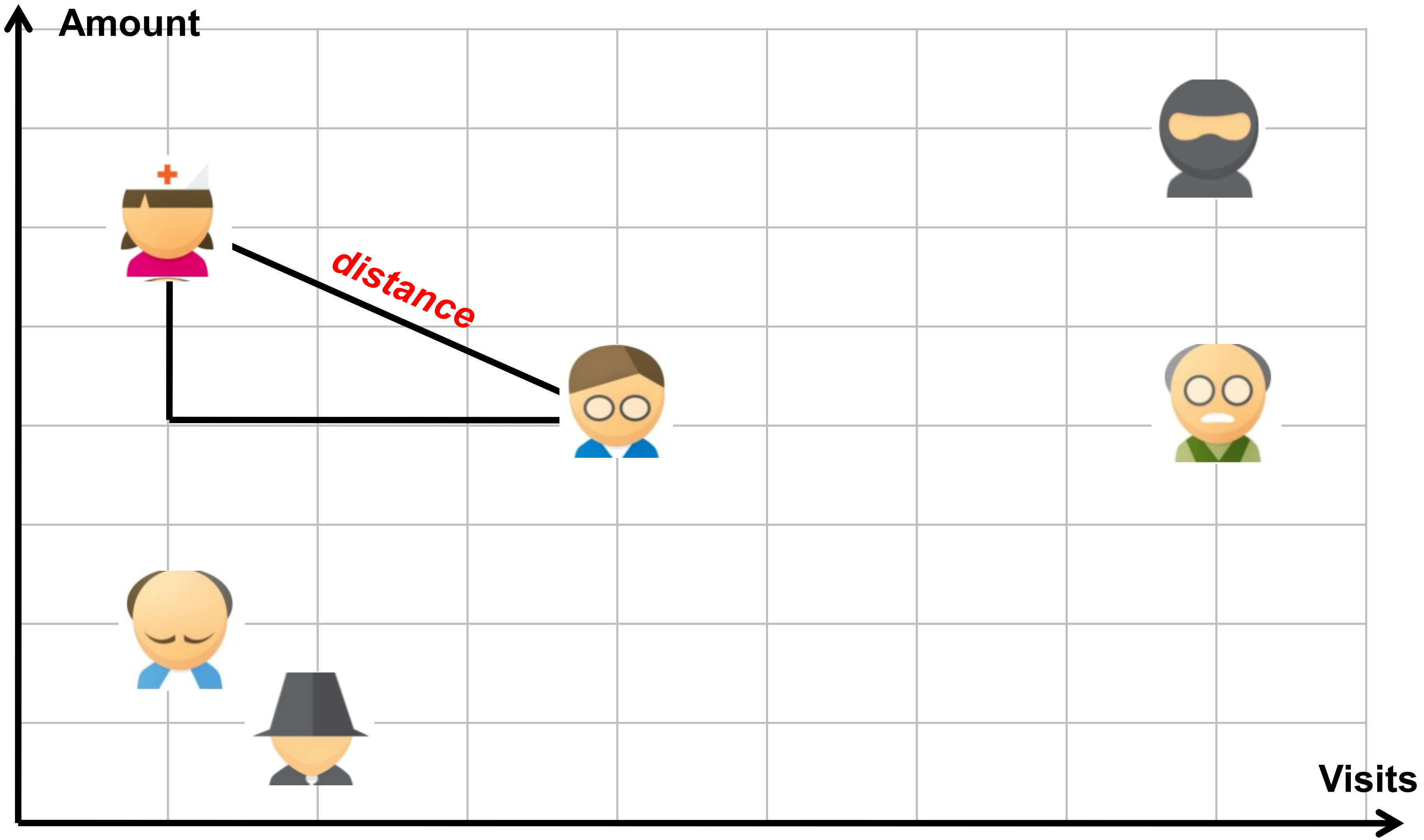
https://www.analyticsvidhya.com/blog/2016/11/an-introduction-to-clustering-and-different-methods-of-clustering/

K-Means Clustering



https://sandipanweb.wordpress.com/2017/03/19/hard-soft-clustering-with-k-means-weighted-k-means-and-gmm-em/

Calculating Distance



Standardize data

```
from sklearn.preprocessing import StandardScaler
df_std = pd.DataFrame(StandardScaler().fit_transform(df[df.columns[1:]]))
df_std.head(5)
```

Clustering

```
from sklearn.cluster import KMeans
cluster = KMeans(n_clusters=7)
```

Merge Result to Original Data

```
df['cluster'] = cluster.fit_predict(df_std)
df.head(5)
```

Determine size of clusters

```
df.cluster.value_counts()
```

Aggregate data by clusters

```
dfCluster = df.groupby('cluster', as_index=False).mean()
dfCluster['NO_CUST_CODE'] = df[['cluster','CUST_CODE']].groupby('cluster').count()

dfCluster
```

Plot clusters

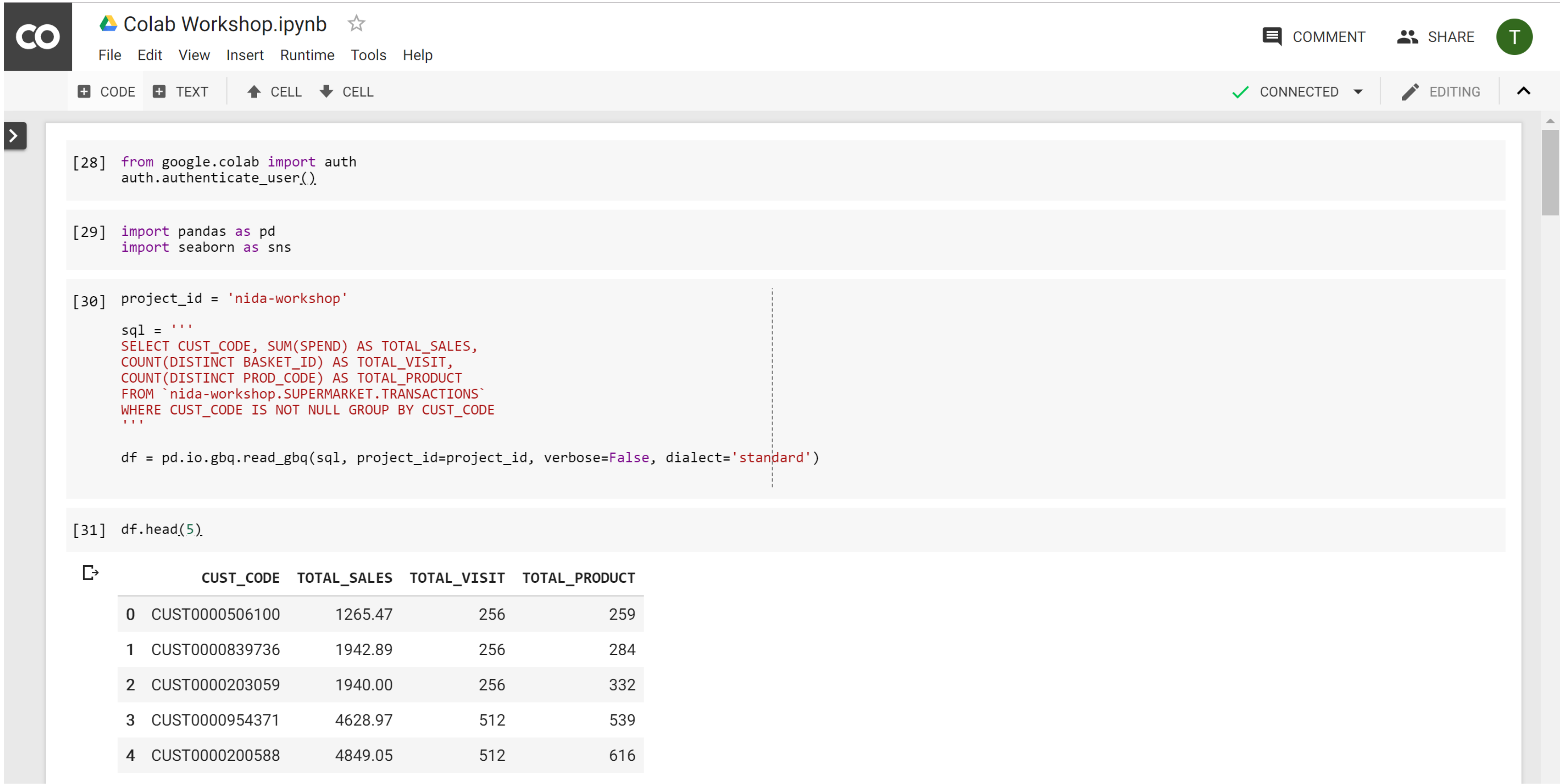
```
 for i, col in enumerate(dfCluster.columns[2:4]):
  sns.plt.figure(i)
  fig, ax = sns.plt.subplots()
  ax.scatter(dfCluster['TOTAL_SALES'], dfCluster[col], s=dfCluster['NO_CUST_CODE'], alpha = 0.5)
  ax.set_xlabel("TOTAL_SALES")
  ax.set_ylabel(col)

  for j, txt in enumerate(dfCluster['cluster']):
      ax.annotate(txt, (dfCluster['TOTAL_SALES'][j], dfCluster[col][j]), horizontalalignment='middle',
verticalalignment='middle')
```

Understand results of clustering

- What are differences between cluster 1 and 5?
- What are differences between cluster 6 and 4?

Colab Workshop Notebook

https://goo.gl/nJRLAU



```
[28] from google.colab import auth
     auth.authenticate_user()
```

```
[29] import pandas as pd
     import seaborn as sns
```

```
[30] project_id = 'nida-workshop'

     sql = '''
     SELECT CUST_CODE, SUM(SPEND) AS TOTAL_SALES,
     COUNT(DISTINCT BASKET_ID) AS TOTAL_VISIT,
     COUNT(DISTINCT PROD_CODE) AS TOTAL_PRODUCT
     FROM `nida-workshop.SUPERMARKET.TRANSACTIONS`
     WHERE CUST_CODE IS NOT NULL GROUP BY CUST_CODE
     '''

     df = pd.io.gbq.read_gbq(sql, project_id=project_id, verbose=False, dialect='standard')
```

```
[31] df.head(5)
```

	CUST_CODE	TOTAL_SALES	TOTAL_VISIT	TOTAL_PRODUCT
0	CUST0000506100	1265.47	256	259
1	CUST0000839736	1942.89	256	284
2	CUST0000203059	1940.00	256	332
3	CUST0000954371	4628.97	512	539
4	CUST0000200588	4849.05	512	616

Questions?