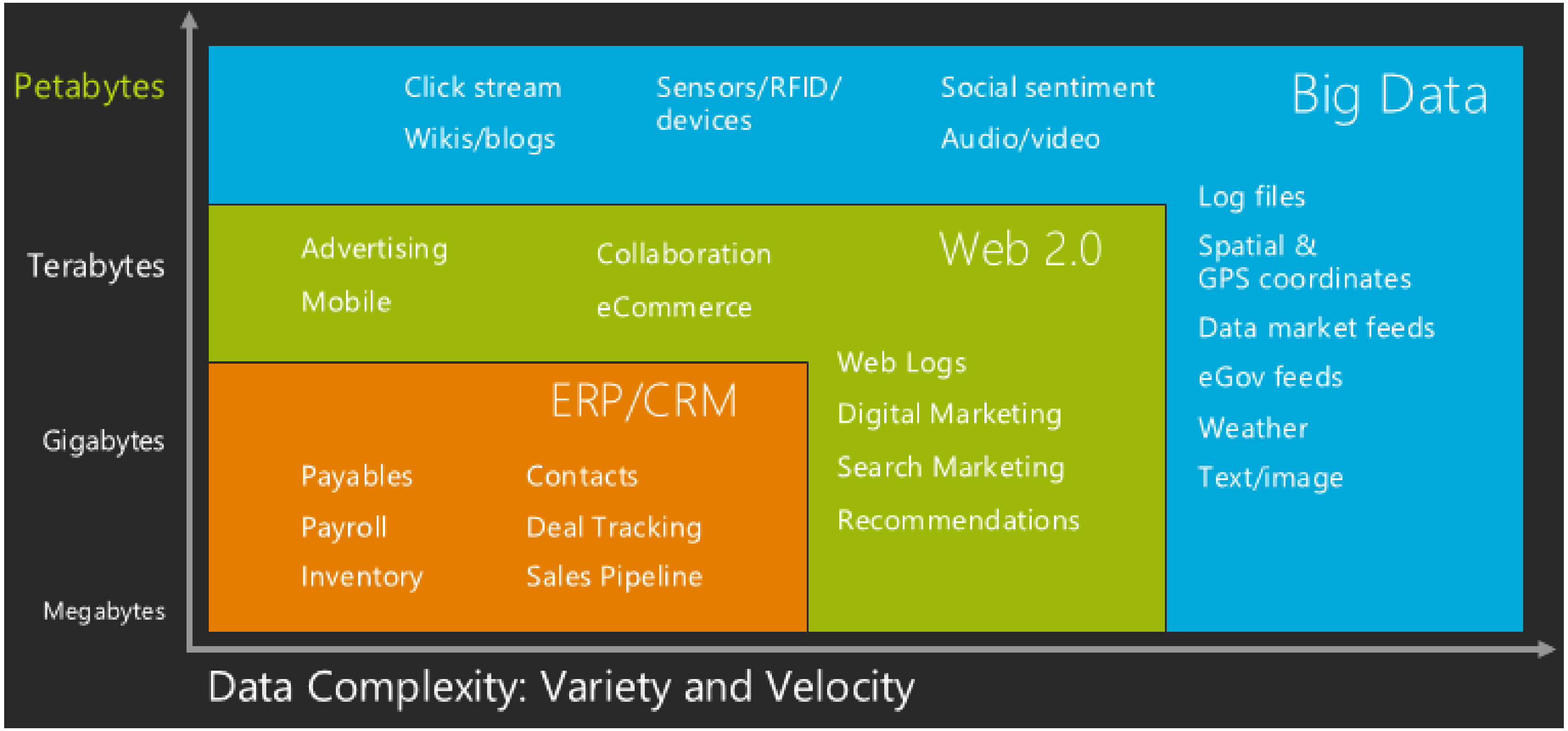
MY472 - Week 11: NoSQL and Working with (Big) Online Databases

Outline

- Database solutions for Big Data
- SQL vs. noSQL
- Cloud solutions
- Examples
  - MongoDB
  - Google BigQuery

Big Data

- Your data can be really big: Gigabytes? Terabytes? Petabytes or more?
- And also very complicated



From: Bigdata Dimension

Database solutions for Big Data

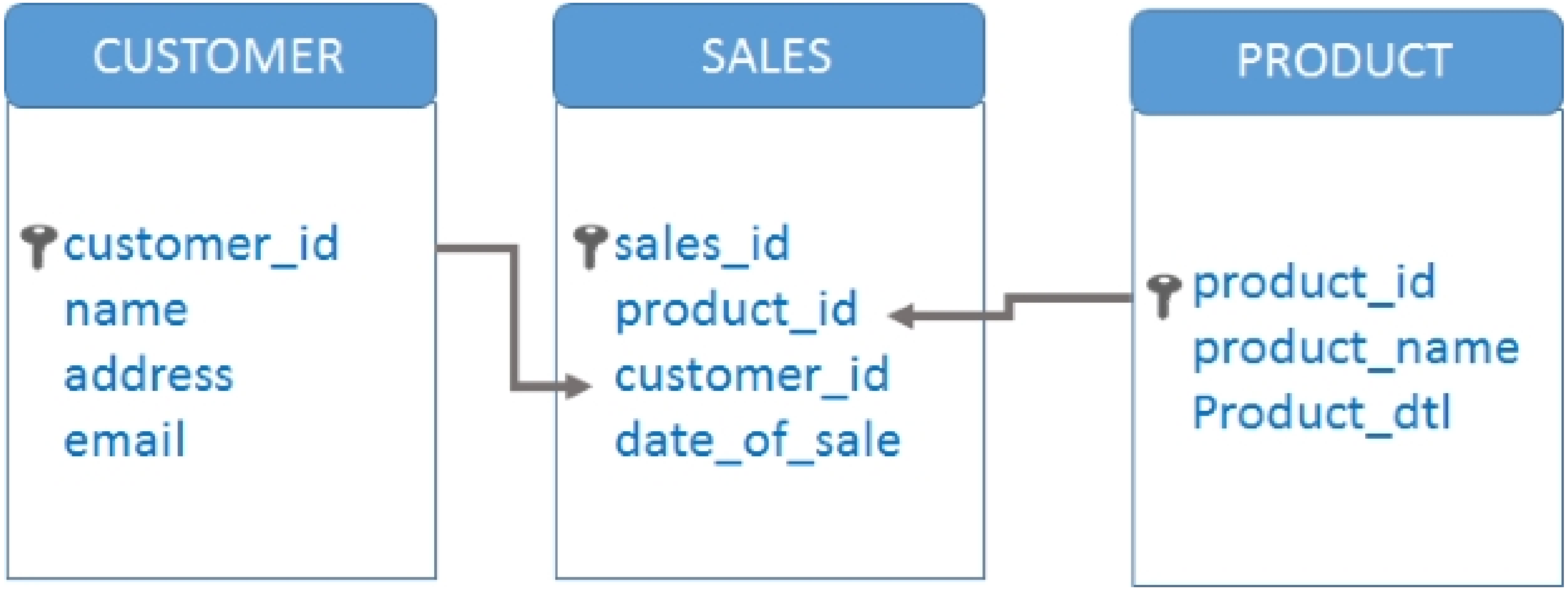
- Different types of databases (SQL vs. NoSQL)
- Cloud solutions using fully managed services

SQL or noSQL?

SQL

- SQL databases have strict structure
- It's all about relations

A simple e-commerce example:



CUSTOMER

customer_id
name
address
email

SALES

sales_id
product_id
customer_id
date_of_sale

PRODUCT

product_id
product_name
Product_dtl

SQL: Review

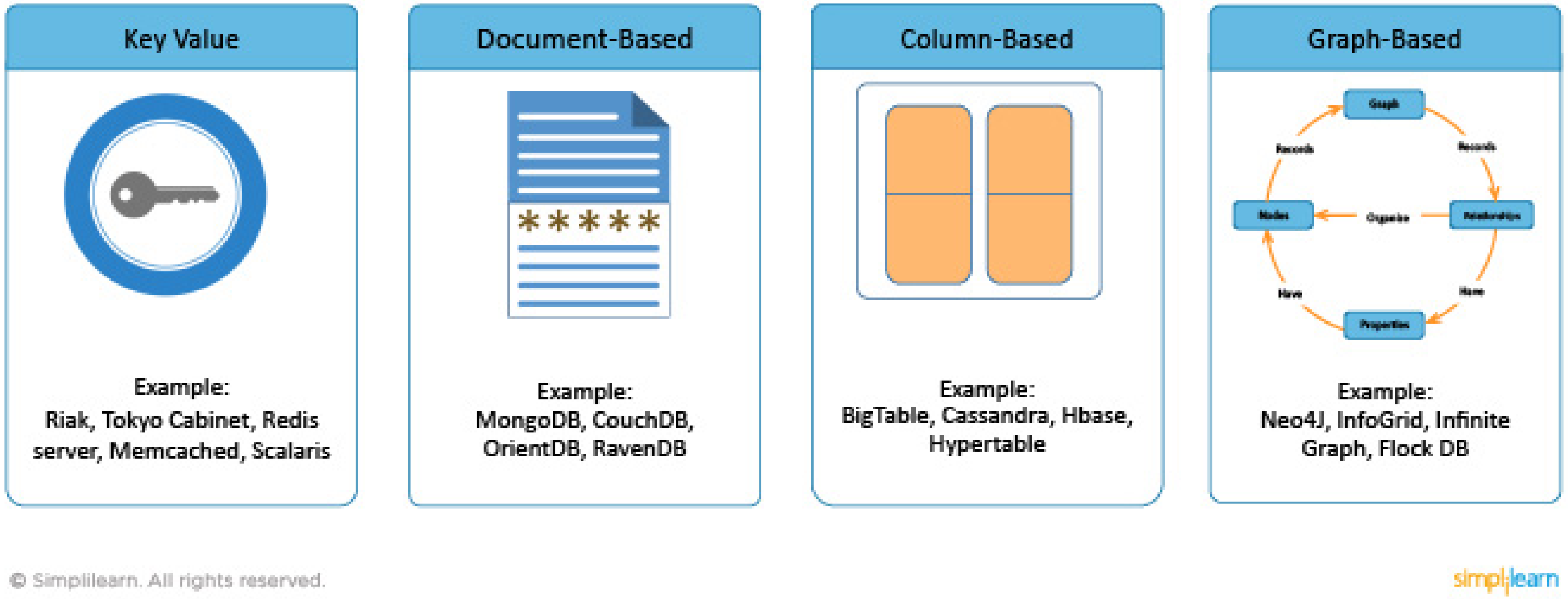
- **SELECT** columns (required)
- **FROM** a table in a database (required)
- **WHERE** rows meet a condition
- **GROUP BY** values of a column
- **ORDER BY** values of a column when displaying results
- **LIMIT** to only X number of rows in resulting table

- **SELECT** can be combined with operators such as **SUM, COUNT**, **AVG**…
- To merge mutliple tables, use **JOIN**
- The result is always a table

noSQL

- Originally referring to "non SQL", "non relational" or "not only SQL"
- Provides a mechanism for storage and retrieval of data which is modeled in means other than the tabular relations used in relational databases
- noSQL databases are good for data with:
    - High **velocity –** lots of data coming in very quickly
    - High **variety** – data can be structured, semi-structured, and unstructured
    - High **volume –** total size of data
    - High **complexity** – stored in many locations

noSQL types



Key Value

Example:
Riak, Tokyo Cabinet, Redis server, Memcached, Scalaris

Document-Based

Example:
MongoDB, CouchDB, OrientDB, RavenDB

Column-Based

Example:
BigTable, Cassandra, Hbase, Hypertable

Graph-Based

Example:
Neo4J, InfoGrid, Infinite Graph, Flock DB

simplilearn

From: Simplelern

noSQL: Pros and Cons

PROS	CONS
Massive scalability	Limited query capabilities
High availability	Not standardized
Schema flexibility	Not matured
Sparse and semistructured data	Developer heavy

MongoDB

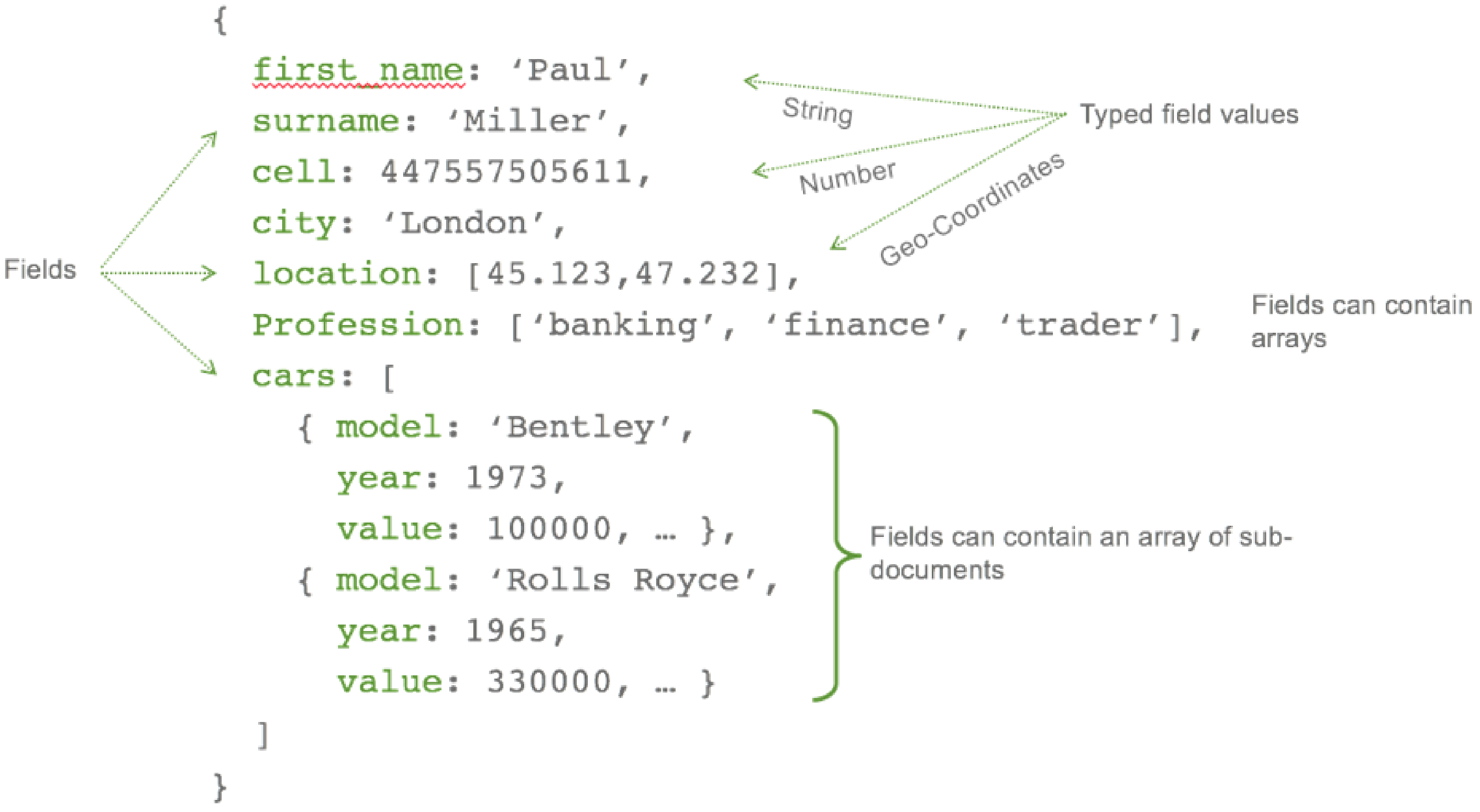
- **Document**-based database
- Concept mapping:

SQL Terms/Concepts	MongoDB Terms/Concepts
database	database
table	collection
row	document or BSON document
column	field

- Each document is constructed as a **BSON** (Binary JSON)

MongoDB documents

A document looks like this:



```
{
  first_name: ‘Paul’,
  surname: ‘Miller’,
  cell: 447557505611,
  city: ‘London’,
  location: [45.123,47.232],
  Profession: [‘banking’, ‘finance’, ‘trader’],
  cars: [
    { model: ‘Bentley’,
      year: 1973,
      value: 100000, … },
    { model: ‘Rolls Royce’,
      year: 1965,
      value: 330000, … }
  ]
}
```

Fields
String
Number
Geo-Coordinates
Typed field values
Fields can contain arrays
Fields can contain an array of sub-documents

From: datawow.io

MongoDB example

See `mongodb-demo.rmd`

- Replication of basic queries from last week using MongoDB
- For a simple selection of documents (i.e. rows in SQL), we will use `find()` method
- For a bit more sophisticated query, we will use `aggregate()` method
- Search query is in **BSON**
- For your reference, we will see the equivalent SQL syntax right above the MongoDB query

MongoDB: JOIN?

- Use **$lookup**:

```
dbMongo$aggregate([
  { "$match": { "party": "Republican" } },
  { "$sort": { "shares_count": -1 } },
  { "$limit": 10 },
  { "$lookup": {
    "localField": "screen_name",
    "from": "congress", "foreignField": "screen_name",
    "as": "congress"
  } }])
```

- This is close to:

```
dbGetQuery(db,  "SELECT posts.*, congress.*
  FROM posts JOIN congress ON congress.screen_name = posts.screen_name
  WHERE party = 'Republican'
  ORDER BY shares_count DESC LIMIT 10")
```

MongoDB: JOIN?

- This will work, but it is not as powerful as SQL's **JOIN**.
- In the end, if you have relational data, use a relational (SQL) database!

Managed services in the cloud

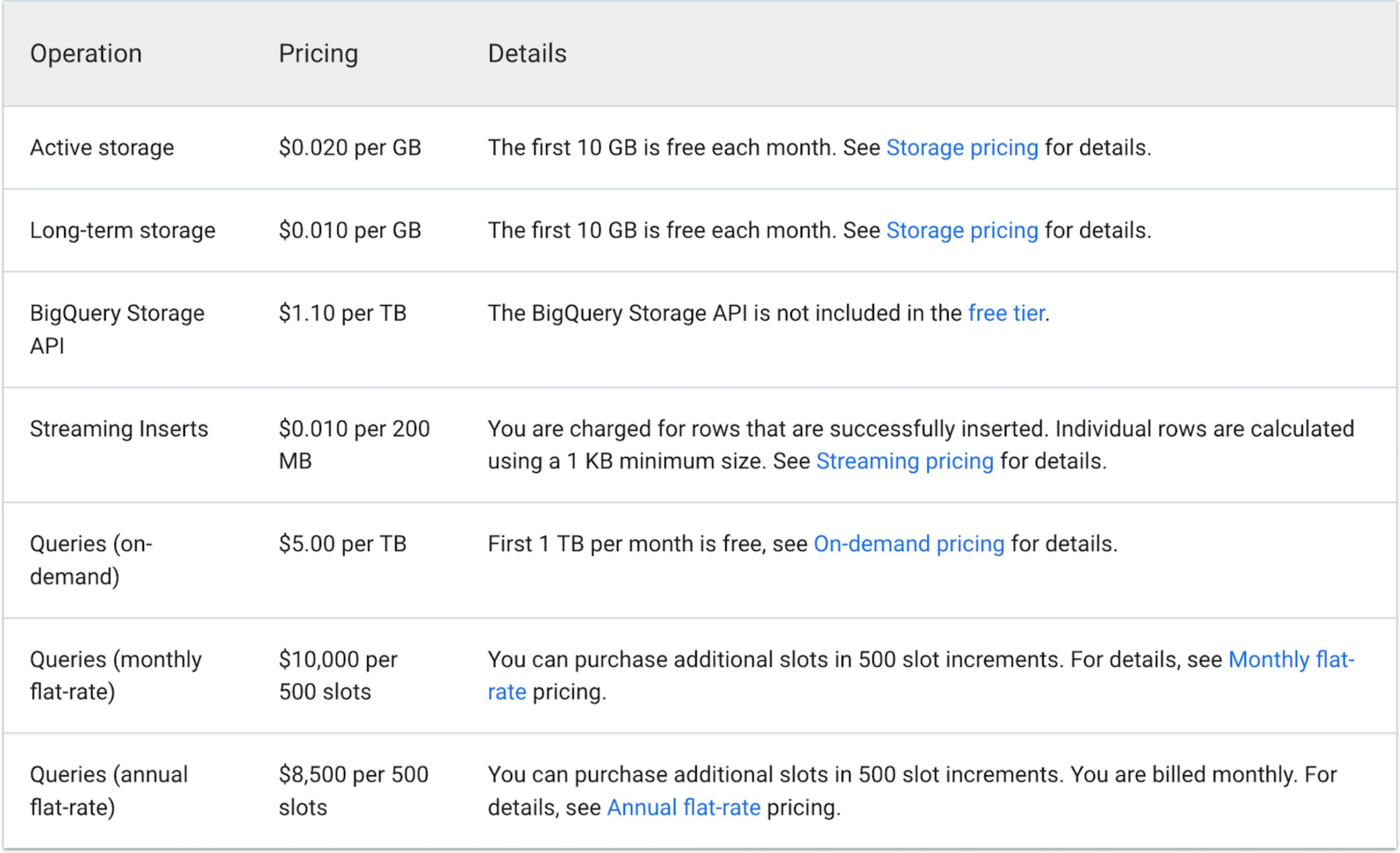
Services

Database Type	AWS	GCP	Azure
Managed RDS	Amazon RDS	Cloud SQL	Azure SQL
Data Warehousing	Redshift	BigQuery	Snowflake
NoSQL (simple key-value)	DynamoDB	BigTable	Azure Tables
NoSQL (document)	MongoDB on EC2	MongoDB on GCE	DocumentDB

Google Cloud Platform: BigQuery

- GCP's data warehousing
- Used by many financial and commercial companies
- Advantages:
    - Integration with other Google data storage solutions (Google Drive, Google Cloud Storage)
    - Scalable: same SQL syntax for datasets of any size
    - Easy to collaborate and export results
    - Affordable pricing and cost control
    - API access allows integration with R or Python
    - Excellent documentation

BigQuery pricing



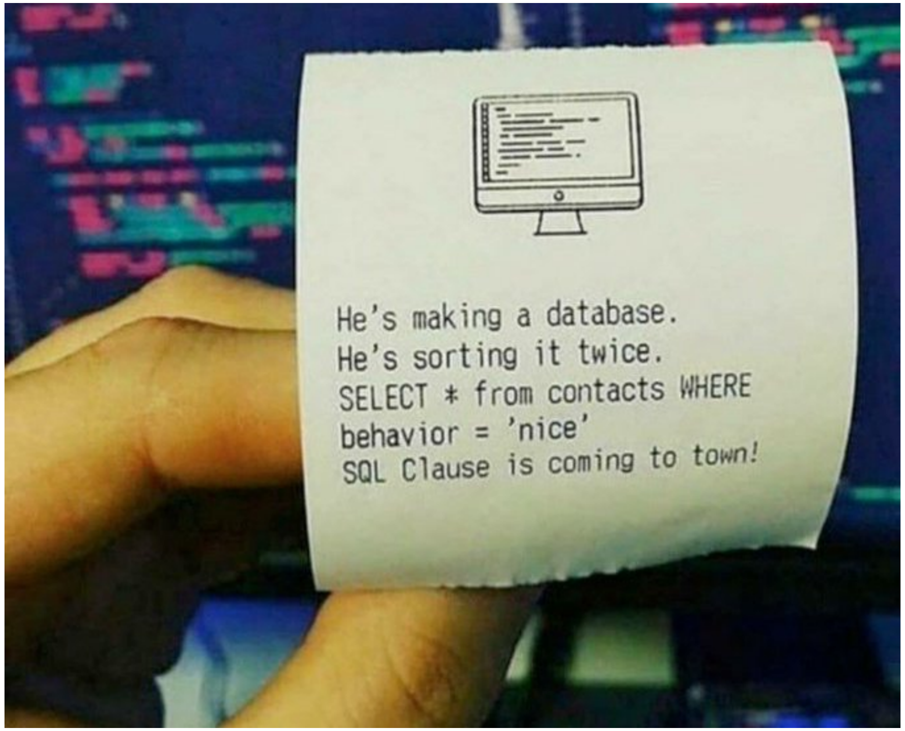
Operation	Pricing	Details
Active storage	$0.020 per GB	The first 10 GB is free each month. See Storage pricing for details.
Long-term storage	$0.010 per GB	The first 10 GB is free each month. See Storage pricing for details.
BigQuery Storage API	$1.10 per TB	The BigQuery Storage API is not included in the free tier.
Streaming Inserts	$0.010 per 200 MB	You are charged for rows that are successfully inserted. Individual rows are calculated using a 1 KB minimum size. See Streaming pricing for details.
Queries (on-demand)	$5.00 per TB	First 1 TB per month is free, see On-demand pricing for details.
Queries (monthly flat-rate)	$10,000 per 500 slots	You can purchase additional slots in 500 slot increments. For details, see Monthly flat-rate pricing.
Queries (annual flat-rate)	$8,500 per 500 slots	You can purchase additional slots in 500 slot increments. You are billed monthly. For details, see Annual flat-rate pricing.

BigQuery example

- `bigquery-demo.rmd`

What's next?

- This week's lab: JOINs and subqueries
- Assessed Assignment #5 due on December 19
- Take-home exam released on December 16 and due on January 17



He’s making a database.
He’s sorting it twice.
SELECT * from contacts WHERE
behavior = ’nice’
SQL Clause is coming to town!

Assessement criteria

- **70–100**: Very Good to Excellent (Distinction)
    - Perceptive, focused use of a good depth of material with a critical edge. Original ideas or structure of argument.
- **60–69**: Good (Merit)
    - Perceptive understanding of the issues plus a coherent well-read and stylish treatment though lacking originality.
- **50–59**: Satisfactory (Pass)
    - A "correct" answer based largely on lecture material. Little detail or originality but presented in adequate framework. Small factual errors allowed.
- **30–49**: Unsatisfactory (Fail)
- **0–29**: Unsatisfactory (Bad fail)
    - Based entirely on lecture material but unstructured and with increasing error component. Concepts are disordered or flawed. Poor presentation. Errors of concept and scope or poor in knowledge, structure and expression.