Advanced Big Data:
Distributed Databases

Dan Zaratsian

AI/ML Architect, Gaming Solutions @ Google

d.zaratsian@gmail.com

https://github.com/zaratsian

TOPICS

- Session 1: Course Intro, Trends, and Approach to AI/ML
- Session 2: SQL and NoSQL
- Session 3: Cloud Machine Learning Services
- Session 4: Distributed ML with Spark and Tensorflow
- Session 5: Cloud Generative AI Services and Architectures
- Session 6: Serverless ML, Architectures, and Deploying ML

Which Database should I use?

#GCPSketchnotes @PVERGADIA THECLOUDGIRL.DEV

03.22.2023

RELATIONAL

Cloud SQL

Managed MySQL, PostgreSQL, SQL Server
SLA: 99.95%

Good For:
- Fastest lift & shift migrations - OLTP workloads
- Managed experience
- Common API & control plane across database engines

Use Case:
- Web Frameworks
- ERP
- CRM
- Ecommerce & Web

AlloyDB

Managed PostgreSQL-compatible, with 100x faster analytics & 4x faster transaction queries
SLA: 99.99%

Good For:
- PostgreSQL workloads that need high performance & high availability or scale
- HTAP (hybrid transaction/ analytical processing)
- Migrations off commercial databases

Use Case:
- Operational Analytics
- ERP, CRM
- Financial Services
- Ecommerce & Web
- SaaS

RELATIONAL & NON-RELATIONAL

Spanner

Multi-dialect (PostgreSQL, GoogleSQL) database, unlimited scaling
SLA: 99.999%

Good For:
Highest scale and availability requirements without compromising on SQL capabilities, consolidating multiple databases for cost savings

Use Case:
- Order & Inventory Management
- Online Banking, payments & ledger
- Gaming: player profiles & gameplay data
- Electronic Medical Records
- Catalog Metadata Management

NON-RELATIONAL (NO SQL)

Firestore

Serverless, document database with built-in cross-client sync and offline caching
SLA: 99.999%

Good For:
- Rapid and cost-efficient application development (no need for middle tier)
- Easy aggregation from multiple data sources

Use Case:
- Web & Mobile Apps
- News Feeds, Social Chat, Influencer Engagements
- Game Saves, Player Profiles
- Retail Catalogs, Point of Sales

Bigtable

Key-value store for large scale, low latency workloads, Supports HBase APIs
SLA: 99.999%

Good For:
- Cost-sensitive applications
- Need high throughput & consistent single-digit millisecond latencies irrespective of scale

Use Case:
- Real-Time Analytics (Personalization, Fraud detection)
- IoT/Clickstream/ Time Series
- Feature Stores, Operational Data Hubs & Data Fabrics
- Batch Unstructured Data Processing
- Financial Markets, Crypto Ledgers

IN MEMORY

Memorystore

Managed Redis & Memcached In memory cache & key-value store
SLA: 99.9%

Good For:
- Sub-millisecond latencies for reads & writes
- Cache to improve app performance & throughput and reduce costs

Use Case:
- Database caching
- Session Store
- Jobs and Queues
- Leaderboards
- Fast Data Ingestion

BARE METAL

Oracle

Lift & shift oracle workloads to Google Cloud, run on certified, pre-configured and tuned infrastructure high performance and availability

Good For:
Oracle applications and databases migrating to cloud

Use Case:
- Legacy Applications
- Datacenter Exits

Fully managed experience for low-touch administration with improved availability, security & governance, support

Source: https://cloud.google.com/blog/topics/developers-practitioners/your-google-cloud-database-options-explained

Google Cloud

Google BigQuery

Google Cloud Platform's **enterprise data warehouse** for analytics

Familiar **SQL 2011** query language and functions

Exabyte-scale, serverless data warehousing

Encrypted, durable, **secure**, And highly available

Built-in support for Generative AI

Unique

Real-time insights from streaming data

Unique

Built-in **ML and Geospatial** for predictive insights

Unique

High-speed, in-memory **BI Engine** for faster reporting and analysis

Unique

Nested and repeated fields, user-defined functions in JavaScript

Unique

Your database choice depends on your needs

Availability

Multi-Region

Skill Set

Operations

Compatibility

Consistency

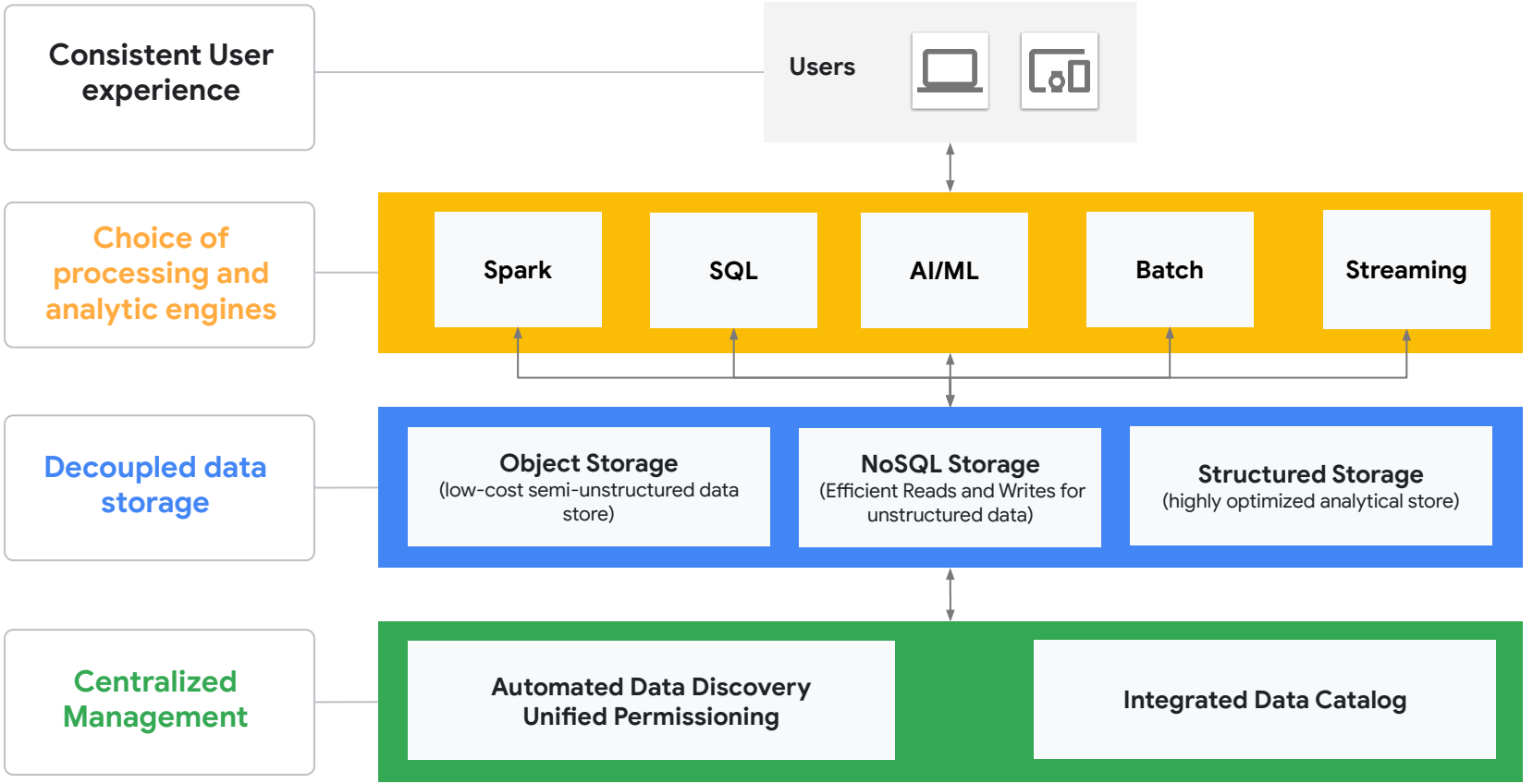
Cost

Future proofing

Data model

Open standard

Data Warehouse building blocks



SQL vs NoSQL

SQL

[MySQL, Postgres, BigQuery, Oracle, etc.]

- Relational databases
- Data stored in tables (tabular)
- Requires schema
- Mostly vertically scalable
- Generally better for multi-row transactions
- Best for heavy transactional workloads
- Uses SQL standard languages for queries

NoSQL

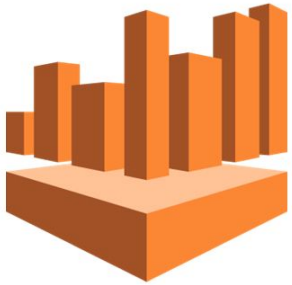
[HBase, Cassandra, MongoDB, Firestore, etc.]

- Unstructured format
- Non-relational database
- Does **not** require a schema
- Easily scalable and distributed
- Horizontally scalable
- Low-latency scans and lookups
- Best for cases where query path is well defined
- Query language not as standardized and syntax is specific to DB.

Note: New DBs are blurring the line. SQL databases now support NoSQL and nested JSON, while NoSQL databases allow users to write SQL queries.

SQL

Google BigQuery



Amazon Athena

Amazon **Redshift**

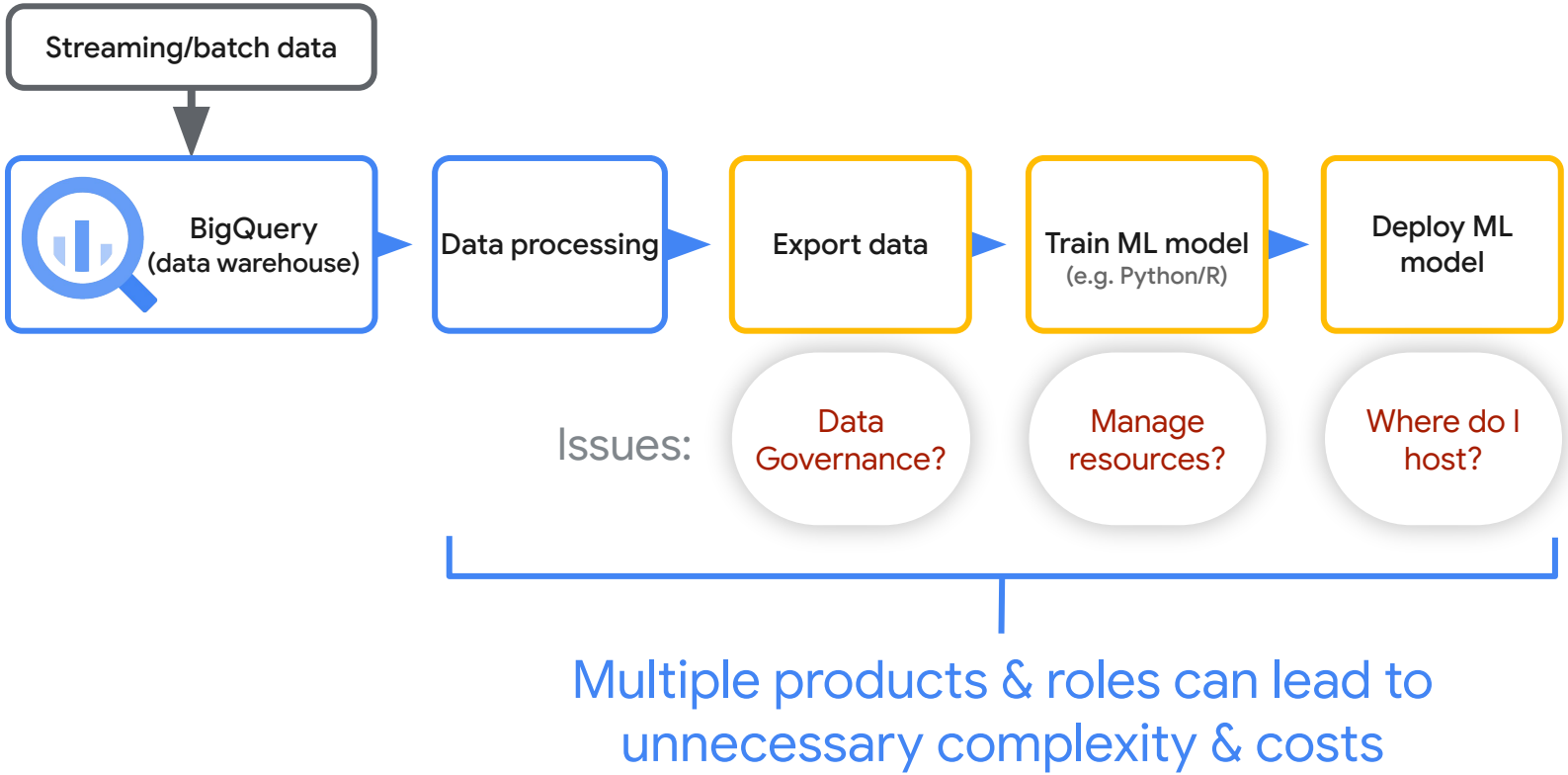
Azure Synapse Analytics

snowflake

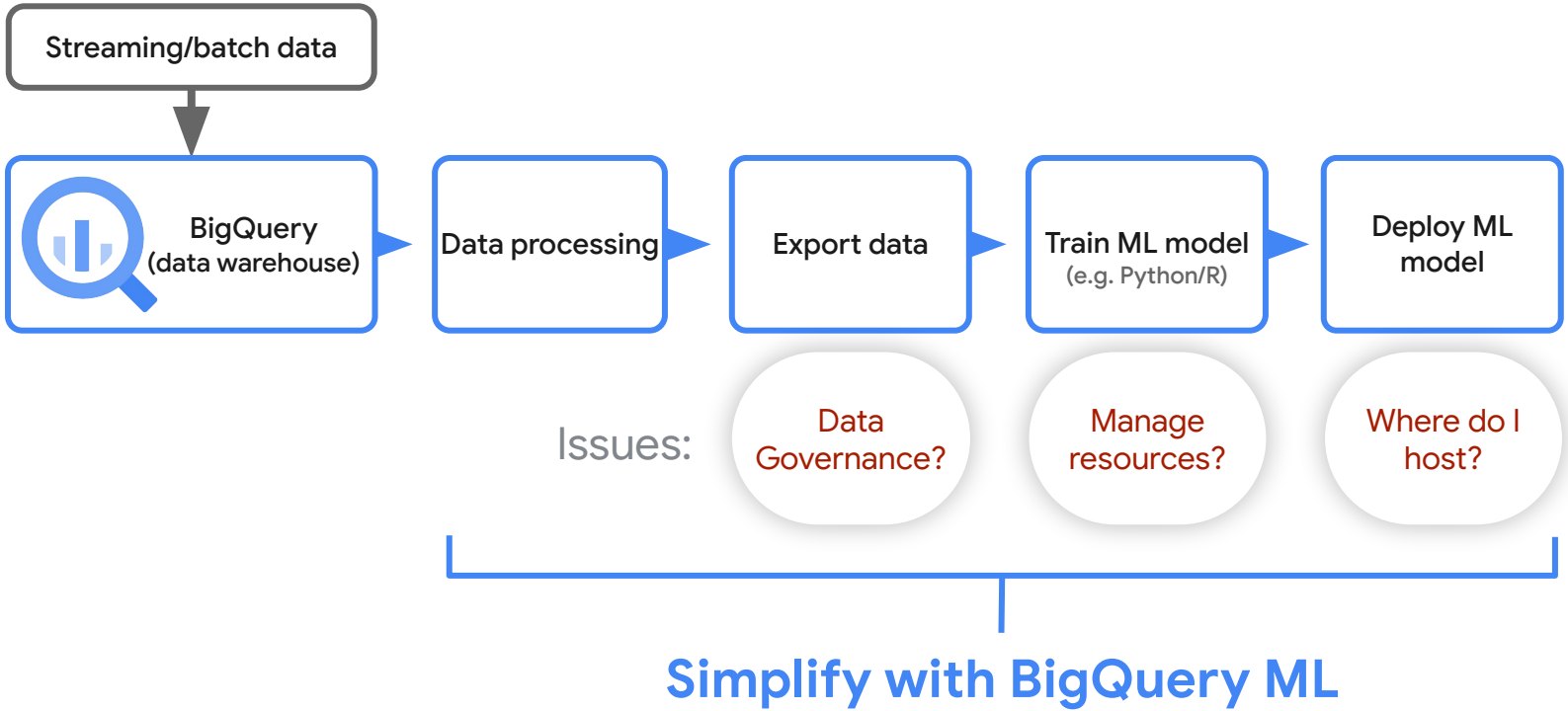
BigQuery

Serverless and cost-effective enterprise data warehouse that works across clouds and scales with your data

Typical ML workflow poses many challenges



BigQuery ML greatly simplifies the ML workflow



BigQuery ML brings BigQuery and Vertex AI together

BigQuery
Cloud Data Warehouse

BigQuery ML
SQL interface to Vertex

Vertex AI
GCP's AI Platform

BigQuery ML integrates with Vertex AI in many areas

Model Training

- Vertex Training for training Tensorflow and XGBoost models
- Vertex Tables for training AutoML tabular models AutoML and Forecasting models [On roadmap]
- Vertex Vizer for hyperparameter tuning

Pre-trained Models

- Inference with LLM foundation models
- Inference for images with Vision API
- Inference for text with NLP API
- Inference for text with Translate API
- Inference for docs with Document AI [On roadmap]

MLOps

- Register models with Vertex AI Model Registry
- Export models and deploy them to Vertex AI Prediction and Vertex Explainable AI
- Model monitoring with Vertex AI
- Vertex AI Pipeline components for use in Notebooks

ML with just a few SQL statements

1 Train

```
CREATE OR REPLACE MODEL `bqml.penguins_model`
OPTIONS (model_type='linear_reg',
         input_label_cols=['body_mass_g']) AS
SELECT * FROM `public-data.ml_datasets.penguins`
WHERE body_mass_g IS NOT NULL
```

2 Predict

```
SELECT * FROM
ML.PREDICT (
    MODEL`bqml_tutorial.penguins_model`,
    (SELECT * FROM`public-data.ml_datasets.penguins`
    WHERE body_mass_g IS NOT NULL AND
          island = "Biscoe"))
```

Train and deploy ML models in SQL

Execute ML workflows without moving data from BigQuery

Built-in infrastructure management, security & compliance

BQML features & capabilities

Rich Built-in Models

- **Supervised Learning**: Linear / Logistic regression, Boosted trees, Random Forest, DNN, Wide & Deep models, AutoML Tables
- **Unsupervised Learning**: K-Means, PCA, Matrix Factorization, Autoencoder
- **Time Series:** ARIMA_PLUS (univariate) and ARIMA_PLUS_XREG (multivariate)

Feature Engineering & others

- Rich feature engineering support
- Model evaluation
- Model management

ML Ops & Cycles

- Hyperparameter Tuning
- Explainable AI
- Integration with Vertex Model Registry
- Components for Vertex Pipelines
- Model Monitoring (soon)

AI/ML Applications

- Generative AI
- NLP & Translation
- Computer Vision
- Embeddings
- Recommendation
- Forecasting
- Anomaly Detection
- Speech & Document AI

Inference Engine

- **Remote Model Inference:** with Vertex Endpoint
- **Imported Model Inference:** with TF, TFLite, XGBoost & ONNX (enabling sklearn, Pytorch, SparkML)
- **Export Modes:** for online inference in Vertex

Google Cloud

Predict against Vertex LLMs in BigQuery

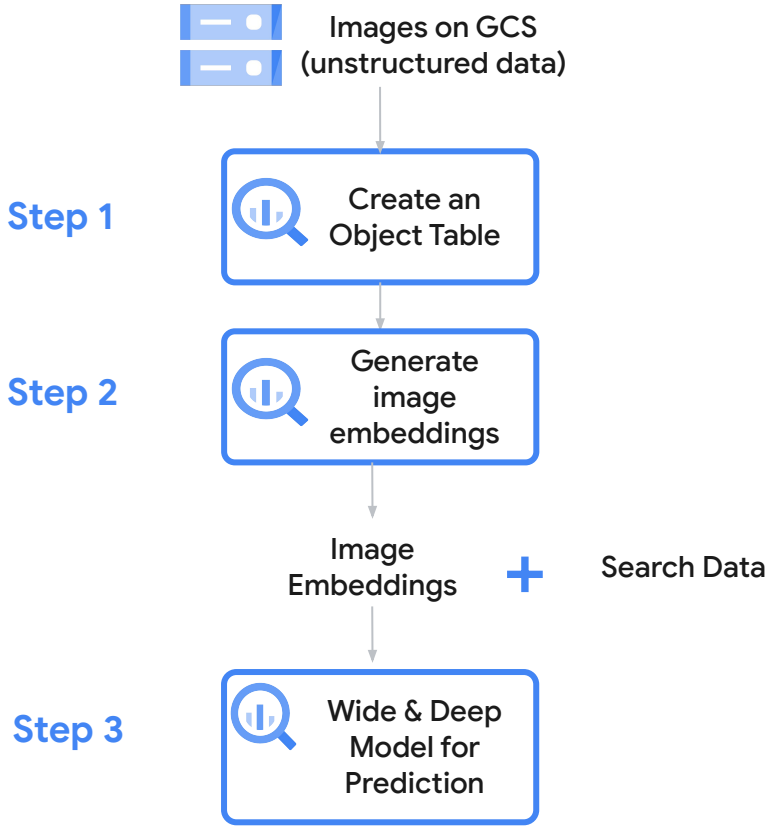
```
SELECT * FROM
ML.GENERATE_TEXT (
  MODEL`my_company.llm_model`,
  (SELECT
   CONCAT("Give the country name for city: ", city) AS prompt
   FROM example_table),
   STRUCT (0.2 AS temperature,
           1024 AS max_output_tokens,
           0.8 AS top_p,
    40 AS top_k))
```

Use LLM models in SQL

Execute Generative AI workflows inside BigQuery

BigQuery Example (SQL + NoSQL) Data processing

Predict Click-through rate of rentals (based on image and structured data)



Now you can combine structured and unstructured data in BQML

Access to image data by creating an Object Table

```
CREATE OR REPLACE EXTERNAL TABLE
`images.property-images`
WITH CONNECTION `us.property-images`
OPTIONS(
object_metadata="DIRECTORY",
      uris=["gs://demo/images/*"]
);
```

uri	content_type	size
gs://as-bqml-launch-demo/images/ER010-aerialrearext.jpg	image/jpeg	1940969
gs://as-bqml-launch-demo/images/KD1111-rearext.jpg	image/jpeg	1508587
gs://as-bqml-launch-demo/images/ER011-aerialrearext.jpg	image/jpeg	1949278
gs://as-bqml-launch-demo/images/B800-rearext.jpg	image/jpeg	338409
gs://as-bqml-launch-demo/images/b372-rearext.jpg	image/jpeg	1841978
gs://as-bqml-launch-demo/images/J10975-aerial-2.jpg	image/jpeg	1606464

Step 1: Create an object table → Step 2: Generate image embeddings → Step 3: Wide & Deep model for prediction

Create image embeddings by importing a Tensorflow image model

```
CREATE OR REPLACE MODEL
`pipeline.resnet_imagenet_embeddings_model`
OPTIONS(
 model_type="TENSORFLOW",
 color_space="RGB",
 model_path="gs://demo/resnet-embeddings/*")
```

```
SELECT *
FROM ML.PREDICT (
    MODEL `pipeline.resnet_imagenet_embeddings_model`,
    (SELECT * FROM `images.property-images`)
```

img_name	pc1	pc2	pc3
EC1-Bar.jpg	1.8493...	1.3984...	1.9969...
EC1-ext.jpg	-18.16...	-8.0527...	1.3630...
EC1-Bar2.jpg	-1.382...	-0.5212...	-3.135...
EC1-Bar3.jpg	3.8091...	-5.0284...	2.1694...
ec1-pool.jpg	-4.891...	7.1742...	-5.182...

Step 1: Create an object table → **Step 2: Generate image embeddings** → Step 3: Wide & Deep model for prediction

Train a Wide and Deep model for prediction

image_embeddings

img_name	pc1	pc2	pc3
EC1-Bar.jpg	1.8493...	1.3984...	1.9969...
EC1-ext.jpg	-18.16...	-8.0527...	1.3630...
EC1-Bar2.jpg	-1.382...	-0.5212...	-3.135...
EC1-Bar3.jpg	3.8091...	-5.0284...	2.1694...
ec1-pool.jpg	-4.891...	7.1742...	-5.182...

search_data

img_name	distanceToB...	location	state
EC1-Bar.jpg	Oceanfront	Corolla	NC
EC1-ext.jpg	Oceanfront	Corolla	NC
EC1-Bar2.jpg	Oceanfront	Corolla	NC
EC1-Bar3.jpg	Oceanfront	Corolla	NC
ec1-pool.jpg	Oceanfront	Corolla	NC

img_name	distanceTo...	location	state	pc1	pc2	pc3
EC1-Bar.jpg	Oceanfront	Corolla	NC	1.84936261...	1.39845214...	1.99695636...
EC1-ext.jpg	Oceanfront	Corolla	NC	-18.166918...	-8.0527233...	1.36307251...
EC1-Bar2.jpg	Oceanfront	Corolla	NC	-1.3826520...	-0.5212412...	-3.1352281...
EC1-Bar3.jpg	Oceanfront	Corolla	NC	3.80916701...	-5.0284545...	2.16948761...
ec1-pool.jpg	Oceanfront	Corolla	NC	-4.8918667...	7.17424064...	-5.1822880...

image__embeddings_with_search

```
CREATE OR REPLACE MODEL pipeline.wide_and_deep
OPTIONS(MODEL_TYPE='DNN_LINEAR_COMBINED_CLASSIFIER
',
          AUTO_CLASS_WEIGHTS =TRUE,
          ACTIVATION_FN = 'RELU',
          BATCH_SIZE = 64,
          DROPOUT = 0.1,
          EARLY_STOP = FALSE,
          HIDDEN_UNITS = [128, 64,32,64,128],
          INPUT_LABEL_COLS = ['clicked'],
          LEARN_RATE=0.00001,
          MAX_ITERATIONS = 25,
          OPTIMIZER = 'RMSPROP',
          DATA_SPLIT_METHOD="CUSTOM",
          DATA_SPLIT_COL = "split_col_bool",
          ENABLE_GLOBAL_EXPLAIN = TRUE
          )
Select * from image_embeddings_with_search
```

Step 1: Create an object table → Step 2: Generate image embeddings → **Step 3: Wide & Deep model for prediction**

Use ML.predict to predict the click-through rate of rental listings

```
SELECT
  *
FROM
  ML.PREDICT(MODEL
`bqml-demo.pipeline.wide_and_deep`,
(SELECT * FROM base));
```

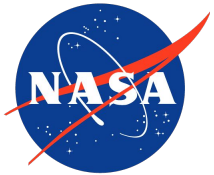
name	img_name	avg_prob
Aqua Dream	J10975-aerial-2.jpg	0.77494834...
Atlantis	KD1111-rearext.jpg	0.78830668...
Creme de la Creme	J20944-aerialrearext.jpg	0.69067030...
Heavens to Betsy	ER011-aerialrearext.jpg	0.82573245...
Mariner's Compass	ER008-aerialrearext.jpg	0.74198232...
SEAesta	KD1301-rearext.jpg	0.69142866...
Salty Paws	B800-rearext.jpg	0.70336384...
Sound to Sea Beach Club	ec3-aerialrearext.jpg	0.78168062...

Step 1: Create an object table → Step 2: Generate image embeddings → **Step 3: Wide & Deep model for prediction**

BigQuery Best Practices

- Don’t SELECT * unless you need every field
- Filter early and often using WHERE clauses
- Do the biggest joins first
- Avoid self-join if you can, since it squares the number of rows processed
- Built-in functions are faster than User Defined Functions (UDFs)

BigQuery Public Datasets



Cloud SQL

Managed Database Platform for MySQL, PostgreSQL and SQL Server Databases

Cloud SQL

Fully managed relational database service

Supports PostgreSQL, MySQL and SQL Server

Full compatibility with source database engines

Fully Managed & Enterprise Ready

Easy to set up, operate, and scale

Trusted

Enterprise-grade data protection, security and governance

Developer Friendly

Application centric observability and API-first administration

More than

90%

of Google Cloud's top 100 customers use Cloud SQL

When to choose Cloud SQL

- Full compatibility with MySQL, PostgreSQL and SQL Server databases
- Built-in high availability along with option for cross-region read replication
- Automate database provisioning, storage capacity management, and other time-consuming tasks
- Industry leading observability for PostgreSQL and MySQL with Query and System Insights
- Proactive database wellness with Active Assist feature
- Lift and shift migration of databases from on-prem or other clouds
- Common control plane and API across MySQL, PostgreSQL and SQL Server
- Integration with analytics and data visualization with BigQuery and Looker

Considerations

- Zero downtime for maintenance
- Global database for read/write
- Super-Admin Access to tune features

With Cloud SQL you can

Innovate faster | Reduce risk | Gain observability

Spanner

Enterprise-grade, globally-distributed, and strongly-consistent managed SQL database service built for the cloud

What is Spanner?

Relational

ACID transactions,
SQL, Schemas

Horizontally scalable

Distributed RDBMS,
Near unlimited scale

Fully managed ++

Simplified administration,
Enterprise grade

- 99.999% uptime SLA
- Automatic sharding
- Superior price-performance
- No maintenance downtime
- Automatic failure recovery
  RPO =0, RTO = 0
- Zero-touch global replication
- Online, unlimited scaling
- Strong external consistency
- Security and compliance

Spanner processes over 2 billion requests per second at peak
Spanner has more than 6 exabytes of data under management

When to choose Spanner

- Fully managed relational database with unlimited horizontal scale
- Need high availability with zero downtime and online schema changes
- High performance ACID transactions with strong consistency over regions
- PostgreSQL interface to empower developers
- Unified analytics and AI on transactional data with BigQuery and Vertex AI
- In-house real-time CDC and replication with Spanner change streams

Considerations

- You need low-latency local writes across regions
- Your schema is unknown or highly sparse
- You need a drop-in replacement for your legacy database

With Spanner you get

Zero Downtime Unlimited scale 99.999% uptime

NoSQL

Types of NoSQL

Key-Value Stores

Cloud Firestore

Amazon DynamoDB

Column Oriented

Cloud
Bigtable

Document DBs

Graph DBs

Amazon Neptune

Titan

NOTE: NoSQL Database can span across multiple categories. This slide is just intended for example purpose.

Bigtable

Globally distributed, fully managed NoSQL database with high performance at any scale

Bigtable

Real-time data serving and operational analytics at any scale

Flexibility at scale

Flexible schema, eventual consistency*

High throughput

Millions of RPS, Predictable single-digit ms latency

Compatible

HBase API, Apache Spark, Integrates with Apache Beam ecosystem

Example use cases

- Personalization
- Fraud detection
- Data Fabric/Operational Data Store
- Time Series
- Product/Content metadata
- Customer 360

Bigtable has over **10 Exabytes of data** under management

Bigtable processes more than **5 Billion** requests per second at peak

Battle tested by Google

* Strong consistency within a single cluster

When to choose Bigtable

- Build responsive applications with consistent single-digit ms latency
- NoSQL database to seamlessly scale to match storage and throughput needs
- Greater flexibility to evolve your applications, or need for sparse schemas
- Ensure high availability with multi-primary replication in up to 8 regions
- Modernize Apache HBase database with no downtime
- Unified analytics and AI on transactional data with BigQuery integration
- Use cases such as personalization, fraud detection, real-time analytics, IoT and more

Considerations

- Full SQL support
- Need secondary indexes
- Need drop in replacement for Cassandra or DynamoDB

With Bigtable you get

Autoscaling | Low Latency | 99.999% uptime

Firestore

Serverless NoSQL document database built for automatic scaling, high performance in the cloud

Firestore

Unlock application innovation with simplicity, speed and confidence

Serverless, document database

JSON-compatible data model, serializable ACID transactions, elastic scalability, up to 99.999% availability SLA, pay only for what you consume

Secure, backend as a service

Connect directly and securely to the database, making middle tiers optional

Real-time sync & offline access

Built-in data syncing, and fallback to on-device caching when a client loses network connectivity

Well integrated

Deliver results faster with native integrations with Google Cloud, Firebase and 3rd party developer services via Extensions

Firestore by the numbers

Over

4 million

databases have been created in Firestore

Firestore apps power more than

1 billion

monthly active end-users using Firebase Auth

When to choose Firestore

- Document database that effortlessly scales to meet any demand, with no maintenance, and with ACID support
- Accelerate development of mobile, web and IoT apps with direct connectivity to the database
- Built-in live sync and offline mode to make it easy to develop real-time applications
- Libraries for popular languages like Node.js, Java, Go, Ruby and PHP
- Seamless integration with Firebase

Considerations

- You need capacity-based pricing
- Your data is relational in nature
- You need customer-managed encryption keys

With Firestore you get

Serverless | Unlimited scale | 99.999% uptime

Memorystore

Fully Managed **Redis** and **Memcached** database for sub-millisecond data access

For workloads that require microseconds latency, Memorystore is a scalable, secure, highly available in-memory service

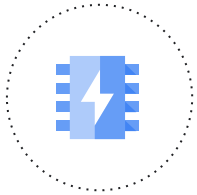
Example use cases

- Caching
- Leaderboard
- Real-time analytics
- Jobs and queues
- Session store
- Fast data ingestion

Fully compatible with Redis and Memcached, offering easy migration for Redis and Memcached workloads.

Over **90%** of the top 100 Google Cloud customers use Memorystore.

When to choose Memorystore



- Need fully managed Redis or Memcached database service
- Build application caches that provide sub-millisecond data access
- Scalability when application grows, and high availability (99.9%)
- Multiple read replicas with up to 6 nodes to serve read traffic
- Web, Mobile Application, Gaming Leaderboard
- Social, chat, news feed applications

Considerations

- Higher data persistence needed
- Lack of SQL Support
- Transaction support is not full ACID compliant

With Memorystore, you get

Scale | High Availability | Choice of engine

Assignment

SQL & NoSQL Assignments

SQL Assignment - Please complete this as an individual assignment and upload your code as a notebook file, script, google doc, etc.

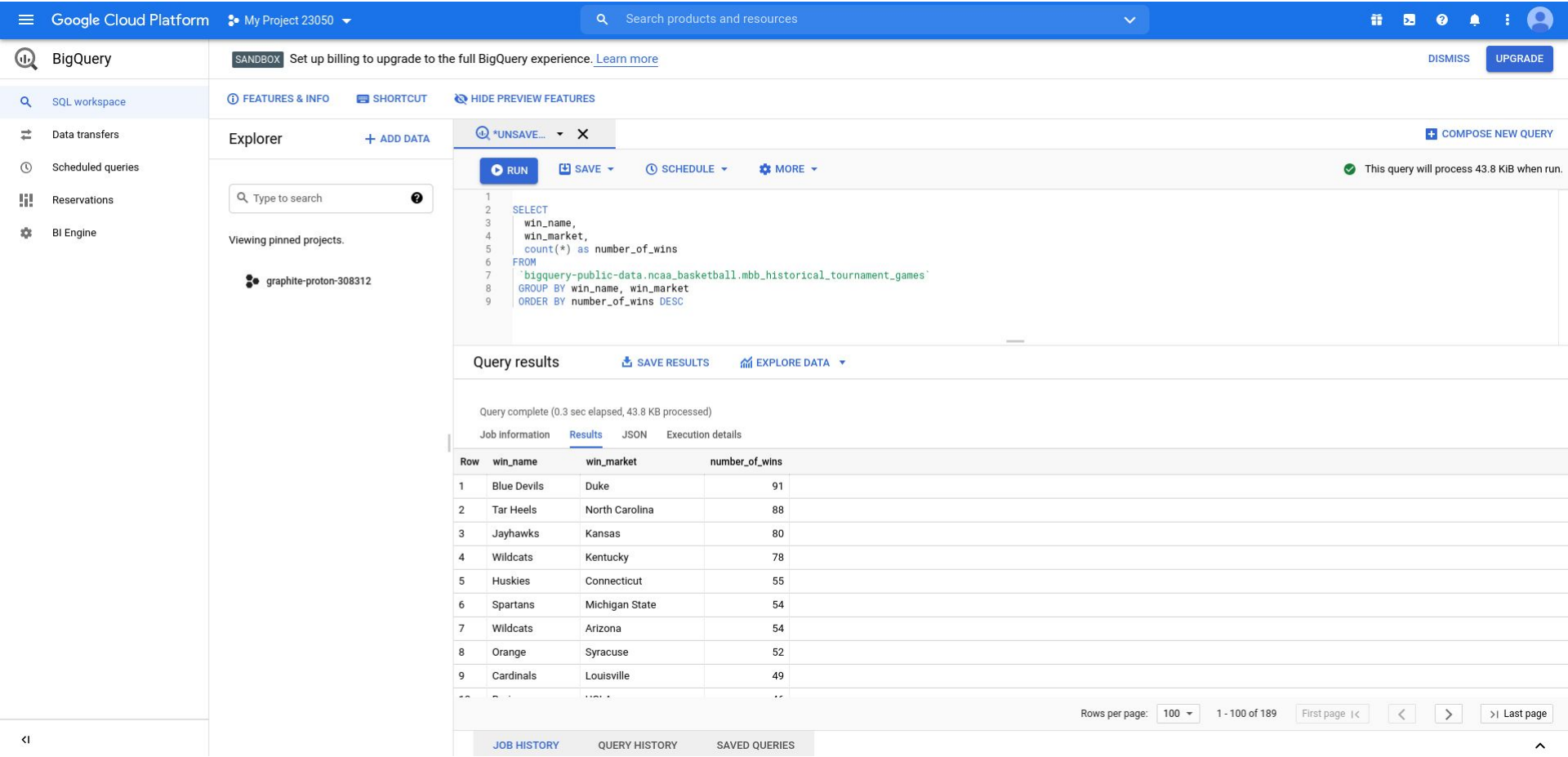
NoSQL Assignment - Please complete this as an individual assignment and upload your code as a notebook file, script, google doc, etc.

Due Date: Thursday, March 14 at 11:59pm EST

Email with any questions: d.zaratsian@gmail.com

Google BigQuery Sandbox Environment

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


Google Cloud Platform
My Project 23050
Search products and resources
BigQuery
SANDBOX Set up billing to upgrade to the full BigQuery experience. Learn more
DISMISS
UPGRADE
SQL workspace
Data transfers
Scheduled queries
Reservations
BI Engine
FEATURES & INFO
SHORTCUT
HIDE PREVIEW FEATURES
Explorer
ADD DATA
Type to search
Viewing pinned projects.
graphite-proton-308312
*UNSAVE...
COMPOSE NEW QUERY
RUN
SAVE
SCHEDULE
MORE
This query will process 43.8 KiB when run.
1 SELECT
2 win_name,
3 win_market,
4 count(*) as number_of_wins
5 FROM
6 `bigquery-public-data.ncaa_basketball.mbb_historical_tournament_games`
7 GROUP BY win_name, win_market
8 ORDER BY number_of_wins DESC
9
Query results
SAVE RESULTS
EXPLORE DATA
Query complete (0.3 sec elapsed, 43.8 KB processed)
Job information
Results
JSON
Execution details
Row win_name win_market number_of_wins
1 Blue Devils Duke 91
2 Tar Heels North Carolina 88
3 Jayhawks Kansas 80
4 Wildcats Kentucky 78
5 Huskies Connecticut 55
6 Spartans Michigan State 54
7 Wildcats Arizona 54
8 Orange Syracuse 52
9 Cardinals Louisville 49
Rows per page: 100
1 - 100 of 189
First page
Last page
JOB HISTORY
QUERY HISTORY
SAVED QUERIES

Colaboratory, or "Colab" for short, allows you to write and execute Python in your browser, with

- Zero configuration required
- Free access to GPUs
- Easy sharing

https://colab.sandbox.google.com/

[13]
```
from pyspark.ml.feature import VectorIndexer, VectorAssembler, StringIndexer
from pyspark.ml.evaluation import RegressionEvaluator
from pyspark.ml import Pipeline
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
```

[9]
```
spark = SparkSession.builder.appName("Bikesharing SparkSQL").master("local[*]").getOrCreate()
```

[14]
```
bikeshare_stations = spark.read.load('bikeshare_stations.csv', format="csv", header=True, inferSchema=True)
```

[16]
```
bikeshare_stations.show(5)
```

```
+----------+--------------------+------+--------+---------+--------------------+
|station_id|                name|status|latitude|longitude|            location|
+----------+--------------------+------+--------+---------+--------------------+
|      3793|   Rio Grande & 28th|active|30.29333|-97.74412|(30.29333, -97.74...|
|      3291|  11th & San Jacinto|active|30.27193|-97.73854|(30.27193, -97.73...|
|      4058|Hollow Creek & Ba...|active|30.26139|-97.77234|(30.26139, -97.77...|
|      3797|   21st & University|active|30.28354|-97.73953|(30.28354, -97.73...|
|      3838|       Nueces & 26th|active|30.29068|-97.74292|(30.29068, -97.74...|
+----------+--------------------+------+--------+---------+--------------------+
only showing top 5 rows
```

```
bikeshare_stations.createOrReplaceTempView("bikeshare_stations")
spark.sql("SELECT name, count(*) as count FROM bikeshare_stations group by name order by count desc").show(truncate=False)
```

```
+-------------------------------------+-----+
|name                                 |count|
+-------------------------------------+-----+
|Lavaca & 6th                         |2    |
|Lake Austin & Enfield                |1    |
|ACC - West & 12th Street             |1    |
|Convention Center / 3rd & Trinity    |1    |
```

Thanks!

Advanced Big Data

Dan Zaratsian
AI/ML Architect, Gaming Solutions @ Google
d.zaratsian@gmail.com
https://github.com/zaratsian

Next up...

Session 2 SQL & NoSQL	Session 3 Distributed ML	Session 4 GenAI	Session 5 Serverless	Session 6 Cloud
			 Cloud Functions	
			 Cloud Run	
				
			 Vertex AI	 Vertex AI
		**GCP PubSub**		