**Assignment 1 : Services for Data Storage**

**Cloud Services for Storing Trillions of Records**

When you’re working with massive-scale systems—whether in big data analytics, IoT, machine learning, or social media—storing trillions of records or petabyte-scale data isn’t unusual. Several cloud platforms are purpose-built for this challenge. Below are three of the most widely used services, with an overview of what they offer and where they shine.

**1. Amazon S3 (Simple Storage Service)**

📌 **Type:** Object Storage  
🏢 **Provider:** Amazon Web Services (AWS)

Amazon S3 is one of the most established and trusted cloud storage services. It’s designed for unlimited scalability, storing anything from backups and archives to massive data lakes. With industry-leading durability (11 nines) and deep integrations across the AWS ecosystem, S3 is a go-to choice for organizations managing trillions of records.

**Key Highlights:**

* Stores data as objects in buckets.
* Versioning to keep multiple file revisions.
* Lifecycle policies to move data automatically to cheaper tiers.
* Tight integration with tools like Athena, EMR, and Redshift.
* Supports serverless pipelines for machine learning and analytics.

**Example in Action:**  
A global streaming company might use S3 to store logs, user activity, and ML training datasets—often hundreds of terabytes every day—then run analytics with Spark or Athena.

**2. Google BigQuery**

📌 **Type:** Serverless Data Warehouse (Columnar Storage)  
🏢 **Provider:** Google Cloud Platform (GCP)

BigQuery is Google’s fully managed data warehouse, built for blazing-fast SQL queries on petabyte-scale datasets. Since it’s serverless, there’s no infrastructure to manage, and it’s tightly integrated with Google’s analytics and ML ecosystem.

**Key Highlights:**

* Columnar storage optimized for analytical queries.
* Pay only for queries and storage (no servers to maintain).
* Native support for ML with Big Query ML.
* Streaming ingestion for real-time analytics.
* Can query external data in GCS, Spanner, or Cloud SQL.

**Example in Action:**  
A financial services firm could load billions of global transactions into Big Query each day and run near real-time fraud detection and trend forecasting.

**3. Apache Cassandra / Amazon Keyspaces**

📌 **Type:** Distributed NoSQL Database  
🏢 **Open Source / AWS Managed**

Cassandra is a distributed, highly available NoSQL database built for scale. It’s ideal for time-series or sensor data, especially when write-heavy workloads are involved. Amazon Keyspaces offers the same Cassandra-compatible API but with the benefits of a fully managed, serverless AWS service.

**Key Highlights:**

* Linear scalability—performance grows as you add nodes.
* Masterless architecture with no single point of failure.
* Highly available with tunable consistency.
* Wide-column model for flexible schemas.

**Example in Action:**  
An IoT provider might use Cassandra to capture and query time-series data from millions of devices, handling millions of writes per second across multiple regions.

**🧠 At a Glance**

| **Service** | **Storage Type** | **Best For** | **Scalability** | **Real-Time Support** | **Cost Model** |
| --- | --- | --- | --- | --- | --- |
| **Amazon S3** | Object Storage | Data lakes, backups, ML datasets | Virtually unlimited | Event-driven (Lambda, etc.) | Pay-per-GB |
| **Google BigQuery** | Columnar DW | Analytics, dashboards, ML | Petabyte-scale | Yes (Streaming API) | Pay-per-query |
| **Cassandra / Keyspaces** | Wide-column NoSQL | Time-series, write-heavy data | Horizontally scalable | Yes | On-demand or provisioned |

**Putting It All Together**

In practice, organizations rarely rely on just one of these. Instead, they use hybrid architectures that combine the strengths of each:

* **Cassandra or Kafka** for real-time ingestion and fast writes.
* **Amazon S3** as the long-term, cost-effective storage layer.
* **Big Query (or Snowflake/Redshift)** for deep analytics and BI reporting.

This approach balances real-time performance with massive-scale analytics and historical storage.

**🧪 Example: Real-Time Health Monitoring Platform**

* IoT health devices send continuous data streams (heart rate, temperature, etc.) into Kafka.
* Kafka writes to **Cassandra** for real-time anomaly detection and to **S3** for historical storage.
* **Spark** processes daily metrics and updates ML models.
* **Big Query** analyses trends across millions of users for dashboards in Looker or Power BI.

The result: doctors and administrators get both **live insights** and **long-term trend analysis**—all from one scalable pipeline.

⚖️ **Why this matters:**

* Each layer is loosely coupled for flexibility.
* Petabyte-scale systems become manageable without re-architecting every few months.
* You get both real-time responsiveness and cost-effective long-term analytics.

**Assignment 2 : Churn Rate in a Retail Industry**

**What drives churn**

Churn is influenced by a mix of customer background, shopping behavior, engagement, and external factors:

1. **Customer background & acquisition**
   * How long they’ve been a customer, where they first purchased, and what offer brought them in.
   * Distance from home to store and affluence of their neighbourhood.
2. **Purchase behaviour**
   * Recency, frequency, and spend (RFM).
   * Basket size, margins, categories shopped, share of private label.
   * Deal hunting, returns, and irregular buying patterns.
3. **Engagement & loyalty**
   * Loyalty tier, points and wallet balance.
   * App/website visits, cart abandonment, response to emails/SMS/push.
   * Feedback, reviews, and service interactions.
4. **Experience & operations**
   * Stockouts, poor substitutions.
   * Delivery/click & collect reliability.
   * Refunds, customer service responsiveness, in-store experience.
5. **Pricing & promotions**
   * Competitiveness of pricing.
   * Depth and relevance of offers.
6. **Assortment & availability**
   * Discontinued favorites or out-of-stock staples.
   * Access to new products.
7. **Channel & store factors**
   * Store closures, staff changes, checkout issues, payment frictions.
8. **External context**
   * Inflation, fuel prices, weather events, competitor activity, shifting holidays.

**Early warning signals**

* Longer time since last purchase compared to a customer’s usual pattern.
* Drop in engagement with marketing or app visits without buying.
* Stockouts or repeated delivery failures.
* Increase in returns or unresolved service issues.
* Buying only on promotion, showing increased price sensitivity.

**How to measure churn drivers**

* **Descriptive:** Look at retention curves by cohort, and check correlations between churn and service issues.
* **Predictive:**
  + Use survival models (Kaplan–Meier, Cox PH) to estimate time-to-churn.
  + Gradient boosting (LightGBM/XGBoost) for next-N-days churn predictions, with SHAP to explain top drivers.
  + Sequence models (RNN/Transformers) for detailed event streams.
* **Causal:**
  + Uplift models to identify who responds to interventions.
  + A/B tests on win-back offers.
  + Difference-in-difference for operational changes (like store remodels or delivery policy tweaks).

**Practical starter features**

* **Purchase:** Recency, frequency, spend, variability in order intervals, basket sizes, category mix.
* **Price/promo:** % spend on promo, discount depth, price competitiveness.
* **Loyalty/engagement:** Loyalty tier, expiring points, email/app activity, cart abandons.
* **Operations:** Out-of-stock exposure, delivery SLA misses, returns/refunds, support interactions.
* **Context:** Distance to store, competitor presence, local weather anomalies.

**Avoid common traps**

* Don’t leak future info (only use data available up to prediction point).
* Use rolling-window validation instead of random splits.
* Handle class imbalance carefully—optimize for business outcomes, not accuracy.
* Account for seasonality in evaluation.

**Action playbook by driver**

* **Stockouts/assortment:** Proactive subs, back-in-stock alerts, guaranteed staples.
* **Price sensitivity:** Personalized bundles, price-match, or loyalty boosters instead of heavy discounts.
* **Delivery issues:** Priority slots, waived fees, apology coupons.
* **Low engagement:** Reminders, replenishment nudges, app reactivation flows.
* **Expiring points:** Countdown reminders, special offers to keep high-value customers.
* **High returns or bad service:** Quick refunds, personal outreach, credits to “make it right.”

**A simple workflow you can implement now**

1. Define churn as “no purchase in next 90 days” at the customer-month level.
2. Build features only with data available at month-end.
3. Train a LightGBM model and use SHAP for explainability.
4. Choose thresholds based on profit curves (savings vs intervention cost).
5. Combine propensity with uplift to find persuadable customers.
6. Track success via holdout groups or geo splits, measuring incremental retained revenue.