**Q: What is Amazon Aurora?**

Amazon Aurora is a relational database engine that combines the speed and reliability of high-end commercial databases with the simplicity and cost-effectiveness of open source databases. Amazon Aurora MySQL delivers up to five times the performance of MySQL without requiring any changes to most MySQL applications; similarly, Amazon Aurora PostgreSQL delivers up to three times the performance of PostgreSQL. Amazon RDS manages your Amazon Aurora databases, handling time-consuming tasks such as provisioning, patching, backup, recovery, failure detection and repair. You pay a simple monthly charge for each Amazon Aurora database instance you use. There are no upfront costs or long-term commitments required.

**Q: What does "MySQL compatible" mean?**

It means that most of the code, applications, drivers, and tools you already use today with your MySQL databases can be used with Aurora with little or no change. The Amazon Aurora database engine is designed to be wire-compatible with MySQL 5.6 and 5.7 using the InnoDB storage engine. Certain MySQL features like the MyISAM storage engine are not available with Amazon Aurora.

**Q: What does “PostgreSQL compatible” mean?**

It means that most of the code, applications, drivers, and tools you already use today with your PostgreSQL databases can be used with Aurora with little or no change. The Amazon Aurora database engine is designed to be wire-compatible with PostgreSQL 9.6 and higher, and supports the same set of PostgreSQL extensions that are supported with RDS for PostgreSQL 9.6 and higher, making it easy to move applications between the two engines.

**Q: Amazon Aurora replicates each chunk of my database volume six ways across three Availability Zones. Does that mean that my effective storage price will be three or six times what is shown on the pricing page?**

No. Amazon Aurora’s replication is bundled into the price. You are charged based on the storage your database consumes at the database layer, not the storage consumed in Amazon Aurora’s virtualized storage layer.

Q: What are I/Os in Amazon Aurora and how are they calculated?

I/Os are input/output operations performed by the Aurora database engine against its SSD-based virtualized storage layer. Every database page read operation counts as one I/O. The Aurora database engine issues reads against the storage layer in order to fetch database pages not present in memory in the cache. If your query traffic can be totally served from memory or the cache, you will not be charged for retrieving any data pages from memory. If your query traffic cannot be served entirely from memory, you will be charged for any data pages that need to be retrieved from storage. Each database page is 16KB in Aurora MySQL and 8KB in Aurora PostgreSQL.

Aurora was designed to eliminate unnecessary I/O operations in order to reduce costs and to ensure resources are available for serving read/write traffic. Write I/Os are only consumed when persisting redo log records in Aurora MySQL or write ahead log records in Aurora PostgreSQL to the storage layer for the purpose of making writes durable. Write I/Os are counted in 4KB units. For example, a log record that is 1024 bytes will count as one write I/O operation. However, if the log record is larger than 4KB, more than one write I/O operation will be needed to persist it. Concurrent write operations whose log records are less than 4KB may be batched together by the Aurora database engine in order to optimize I/O consumption, if they are persisted on the same storage protection groups. Unlike traditional database engines, Aurora never flushes dirty data pages to storage.

You can see how many I/O requests your Aurora instance is consuming by going to the AWS Console. To find your I/O consumption, go to the RDS section of the console, look at your list of instances, select your Aurora instances, then look for the “Billed read operations” and “Billed write operations” metrics in the monitoring section.

**Q: Do I need to change client drivers to use Amazon Aurora PostgreSQL?**

No, Amazon Aurora will work with standard PostgreSQL database drivers

Amazon Aurora delivers significant increases over MySQL performance by tightly integrating the database engine with an SSD-based virtualized storage layer purpose-built for database workloads, reducing writes to the storage system, minimizing lock contention, and eliminating delays created by database process threads.

**Q: How do I optimize my database workload for Amazon Aurora MySQL?**

Amazon Aurora is designed to be compatible with MySQL, so that existing MySQL applications and tools can run without requiring modification. However, one area where Amazon Aurora improves upon MySQL is with highly concurrent workloads. In order to maximize your workload’s throughput on Amazon Aurora, we recommend building your applications to drive a large number of concurrent queries and transactions.

**Hardware and Scaling**

**Q: What are the minimum and maximum storage limits of an Amazon Aurora database?**

The minimum storage is 10GB. Based on your database usage, your Amazon Aurora storage will automatically grow, up to 128 TB, in 10GB increments with no impact to database performance. There is no need to provision storage in advance.

**Q: How do I scale the compute resources associated with my Amazon Aurora DB Instance?**

You can scale the compute resources allocated to your DB Instance in the AWS Management Console by selecting the desired DB Instance and clicking the Modify button. Memory and CPU resources are modified by changing your DB Instance class.

When you modify your DB Instance class, your requested changes will be applied during your specified maintenance window. Alternatively, you can use the "Apply Immediately" flag to apply your scaling requests immediately. Both of these options will have an availability impact for a few minutes as the scaling operation is performed. Bear in mind that any other pending system changes will also be applied.

**Backup and Restore**

**Q: How do I enable backups for my DB Instance?**

Automated backups are always enabled on Amazon Aurora DB Instances. Backups do not impact database performance.

**Q: Can I take DB Snapshots and keep them around as long as I want?**

Yes, and there is no performance impact when taking snapshots. Note that restoring data from DB Snapshots requires creating a new DB Instance.

**Q: If my database fails, what is my recovery path?**

Amazon Aurora automatically maintains 6 copies of your data across 3 Availability Zones and will automatically attempt to recover your database in a healthy AZ with no data loss. In the unlikely event your data is unavailable within Amazon Aurora storage, you can restore from a DB Snapshot or perform a point-in-time restore operation to a new instance. Note that the latest restorable time for a point-in-time restore operation can be up to 5 minutes in the past.

**Q: What happens to my automated backups and DB Snapshots if I delete my DB Instance?**

You can choose to create a final DB Snapshot when deleting your DB Instance. If you do, you can use this DB Snapshot to restore the deleted DB Instance at a later date. Amazon Aurora retains this final user-created DB Snapshot along with all other manually created DB Snapshots after the DB Instance is deleted. Only DB Snapshots are retained after the DB Instance is deleted (i.e., automated backups created for point-in-time restore are not kept).

**Q: Will I be billed for shared snapshots?**

There is no charge for sharing snapshots between accounts. However, you may be charged for the snapshots themselves, as well as any databases you restore from shared snapshots

**Q: Can I automatically share snapshots?**

We do not support sharing automatic DB snapshots. To share an automatic snapshot, you must manually create a copy of the snapshot, and then share the copy.

**Q: How many accounts can I share snapshots with?**

You may share manual snapshots with up to 20 AWS account IDs. If you want to share the snapshot with more than 20 accounts, you can either share the snapshot as public, or contact support for increasing your quota.

**Q: In which regions can I share my Aurora snapshots?**

You can share your Aurora snapshots in all AWS regions where Aurora is available.

**Q: Can I share my Aurora snapshots across different regions?**

No. Your shared Aurora snapshots will only be accessible by accounts in the same region as the account that shares them.

**Q: Can I share an encrypted Aurora snapshot?**

Yes, you can share encrypted Aurora snapshots.

**Q: How does Amazon Aurora improve my database’s fault tolerance to disk failures?**

Amazon Aurora automatically divides your database volume into 10GB segments spread across many disks. Each 10GB chunk of your database volume is replicated six ways, across three Availability Zones. Amazon Aurora is designed to transparently handle the loss of up to two copies of data without affecting database write availability and up to three copies without affecting read availability. Amazon Aurora storage is also self-healing. Data blocks and disks are continuously scanned for errors and repaired automatically.

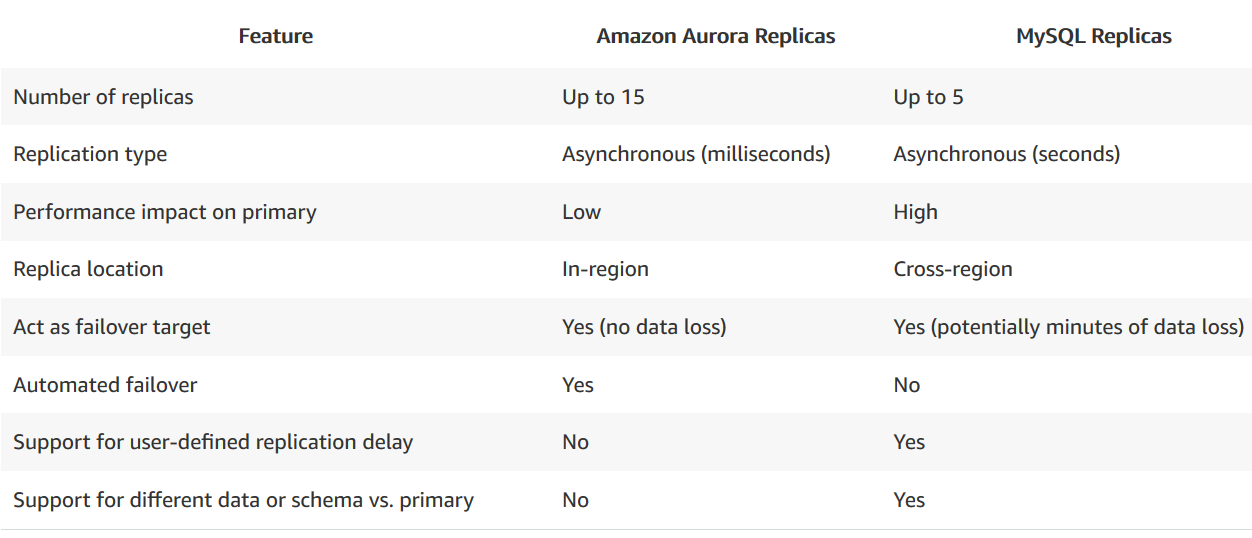
**Q: How does Aurora improve recovery time after a database crash?**

Unlike other databases, after a database crash Amazon Aurora does not need to replay the redo log from the last database checkpoint (typically 5 minutes) and confirm that all changes have been applied, before making the database available for operations. This reduces database restart times to less than 60 seconds in most cases. Amazon Aurora moves the buffer cache out of the database process and makes it available immediately at restart time. This prevents you from having to throttle access until the cache is repopulated to avoid brownouts.

**Q: What kind of replicas does Aurora support?**

Amazon Aurora MySQL and Amazon Aurora PostgreSQL support Amazon Aurora Replicas, which share the same underlying volume as the primary instance in the same AWS region. Updates made by the primary are visible to all Amazon Aurora Replicas. With Amazon Aurora MySQL, you can also create cross-region MySQL Read Replicas based on MySQL’s binlog-based replication engine. In MySQL Read Replicas, data from your primary instance is replayed on your replica as transactions. For most use cases, including read scaling and high availability, we recommend using Amazon Aurora Replicas.

You have the flexibility to mix and match these two replica types based on your application needs:



You have two additional replication options in addition to the ones listed above. You can use Aurora Global Database for much faster physical replication between Aurora clusters in different regions. And for replication between Aurora and non-Aurora MySQL databases (even outside of AWS), you can set up your own, self-managed binlog replication.

**Q: Can I have cross-region replicas with Amazon Aurora?**

Yes, you can set up cross-region Aurora replicas using either physical or logical replication.

Physical replication, called Aurora Global Database, uses dedicated infrastructure that leaves your databases entirely available to serve your application, and can replicate to up to five secondary regions with typical latency of under a second. It's available for both Aurora MySQL and Aurora PostgreSQL. For low-latency global reads and disaster recovery, we recommend using Aurora Global Database.

Aurora supports native logical replication in each database engine (binlog for MySQL and PostgreSQL replication slots for PostgreSQL), so you can replicate to Aurora and non-Aurora databases, even across regions.

Aurora MySQL also offers an easy-to-use logical cross-region read replica feature that supports up to five secondary AWS regions. It is based on single threaded MySQL binlog replication, so the replication lag will be influenced by the change/apply rate and delays in network communication between the specific regions selected.

**Q: Can I create Aurora Replicas on the cross-region replica cluster?**

Yes, you can add up to 15 Aurora Replicas on each cross-region cluster, and they will share the same underlying storage as the cross-region replica. A cross-region replica acts as the primary on the cluster and the Aurora Replicas on the cluster will typically lag behind the primary by 10s of milliseconds.

**Q: Can I fail over my application from my current primary to the cross-region replica?**

Yes, you can promote your cross-region replica to be the new primary from the RDS console. For logical (binlog) replication, the promotion process typically takes a few minutes depending on your workload. The cross-region replication will stop once you initiate the promotion process.

With Aurora Global Database, you can promote a secondary region to take full read/write workloads in under a minute.

**Q: Can I prioritize certain replicas as failover targets over others?**

Yes. You can assign a promotion priority tier to each instance on your cluster. When the primary instance fails, Amazon RDS will promote the replica with the highest priority to primary. If two or more Aurora Replicas share the same priority, then Amazon RDS promotes the replica that is largest in size. If two or more Aurora Replicas share the same priority and size, then Amazon RDS promotes an arbitrary replica in the same promotion tier

**Q: Can I modify priority tiers for instances after they have been created?**

Yes, you can modify the priority tier for an instance at any time. Simply modifying priority tiers will not trigger a failover.

**Q: Can I prevent certain replicas from being promoted to the primary instance?**

You can assign lower priority tiers to replicas that you don’t want promoted to the primary instance. However, if the higher priority replicas on the cluster are unhealthy or unavailable for some reason, then Amazon RDS will promote the lower priority replica.

**Q: How can I improve upon the availability of a single Amazon Aurora database?**

You can add Amazon Aurora Replicas. Aurora Replicas in the same AWS Region share the same underlying storage as the primary instance. Any Aurora Replica can be promoted to become primary without any data loss and therefore can be used for enhancing fault tolerance in the event of a primary DB Instance failure. To increase database availability, simply create 1 to 15 replicas, in any of 3 AZs, and Amazon RDS will automatically include them in failover primary selection in the event of a database outage.

You can use Aurora Global Database if you want your database to span multiple AWS Regions. This will replicate your data with no impact on database performance, and provide disaster recovery from region-wide outages

**Q: What happens during failover and how long does it take?**

Failover is automatically handled by Amazon Aurora so that your applications can resume database operations as quickly as possible without manual administrative intervention.

If you have an Amazon Aurora Replica, in the same or a different Availability Zone, when failing over, Aurora flips the canonical name record (CNAME) for your DB Instance to point at the healthy replica, which is in turn promoted to become the new primary. Start-to-finish, failover typically completes within 30 seconds.

If you are running Aurora Serverless and the DB instance or AZ become unavailable, Aurora will automatically recreate the DB instance in a different AZ.

If you do not have an Amazon Aurora Replica (i.e., single instance) and are not running Aurora Serverless, Aurora will attempt to create a new DB Instance in the same Availability Zone as the original instance. This replacement of the original instance is done on a best-effort basis and may not succeed, for example, if there is an issue that is broadly affecting the Availability Zone.

Your application should retry database connections in the event of connection loss.

Disaster recovery across regions is a manual process, where you promote a secondary region to take read/write workloads.

**Q: If I have a primary database and an Amazon Aurora Replica actively taking read traffic and a failover occurs, what happens?**

Amazon RDS will automatically detect a problem with your primary instance and trigger a failover. If you are using the Cluster Endpoint, your read/write connections will be automatically redirected to an Amazon Aurora Replica that will be promoted to primary.

In addition, the read traffic that your Aurora Replicas were serving will be briefly interrupted. If you are using the Cluster Reader Endpoint to direct your read traffic to the Aurora Replica, the read only connections will be directed to the newly promoted Aurora Replica until the old primary node is recovered as a replica.

**Q: How far behind the primary will my replicas be?**

Since Amazon Aurora Replicas share the same data volume as the primary instance in the same AWS Region, there is virtually no replication lag. We typically observe lag times in the 10s of milliseconds. For MySQL Read Replicas, the replication lag can grow indefinitely based on change/apply rate as well as delays in network communication. However, under typical conditions, under a minute of replication lag is common.

Cross-region replicas using logical replication will be influenced by the change/apply rate and delays in network communication between the specific regions selected. Cross-region replicas using Aurora Global Database will have a typical lag of under a second.

**Q: Can I set up replication between my Aurora MySQL database and an external MySQL database?**

Yes, you can set up binlog replication between an Aurora MySQL instance and an external MySQL database. The other database can run on Amazon RDS, or as a self-managed database on AWS, or completely outside of AWS.

If you're running Aurora MySQL 5.7, consider setting up GTID-based binlog replication. This will provide complete consistency so your replication won’t miss transactions or generate conflicts, even after failover or downtime.

**Q: What is Amazon Aurora Global Database?**

Amazon Aurora Global Database is a feature that allows a single Amazon Aurora database to span multiple AWS regions. It replicates your data with no impact on database performance, enables fast local reads in each region with typical latency of less than a second, and provides disaster recovery from region-wide outages. In the unlikely event of a regional degradation or outage, a secondary region can be promoted to full read/write capabilities in less than 1 minute.

This feature is available for Aurora MySQL and Aurora PostgreSQL.

**Q: How do I create an Aurora Global Database?**

You can create an Aurora Global Database with just a few clicks in the Amazon RDS Management Console. Alternatively, you can use the SDK or CLI. You need to provision at least one instance per region in your Aurora Global Database.

**Q: How many secondary regions can an Aurora Global Database have?**

You can create up to five secondary regions for an Aurora Global Database.

**Q: If I use Aurora Global Database, can I also use logical replication (binlog) on the primary database?**

Yes. If your goal is to analyze database activity, consider using Aurora advanced auditing, general logs, and slow query logs instead, to avoid impacting the performance of your database.

**Q: Will Aurora automatically fail over to a secondary region of an Aurora Global Database?**

No. If your primary region becomes unavailable, you can manually remove a secondary region from an Aurora Global Database and promote it to take full reads and writes. You will also need to point your application to the newly promoted region.

**Q: What is Amazon Aurora Multi-Master?**

Amazon Aurora Multi-Master is a new feature of the Aurora MySQL-compatible edition that adds the ability to scale out write performance across multiple Availability Zones, allowing applications to direct read/write workloads to multiple instances in a database cluster and operate with higher availability.

**Security**

**Q: Can I use Amazon Aurora in Amazon Virtual Private Cloud (Amazon VPC)?**

Yes, all Amazon Aurora DB Instances must be created in a VPC. With Amazon VPC, you can define a virtual network topology that closely resembles a traditional network that you might operate in your own datacenter. This gives you complete control over who can access your Amazon Aurora databases.

**Q: Does Amazon Aurora encrypt my data in transit and at rest?**

Yes. Amazon Aurora uses SSL (AES-256) to secure the connection between the database instance and the application. Amazon Aurora allows you to encrypt your databases using keys you manage through AWS Key Management Service (KMS). On a database instance running with Amazon Aurora encryption, data stored at rest in the underlying storage is encrypted, as are its automated backups, snapshots, and replicas in the same cluster. Encryption and decryption are handled seamlessly.

**Q: Can I encrypt an existing unencrypted database?**

Currently, encrypting an existing unencrypted Aurora instance is not supported. To use Amazon Aurora encryption for an existing unencrypted database, create a new DB Instance with encryption enabled and migrate your data into it.

**Q: How do I access my Amazon Aurora database?**

Access to Amazon Aurora databases must be done through the database port entered on database creation. This is done to provide an additional layer of security for your data

**Serverless**

**Q: What is Amazon Aurora Serverless?**

Amazon Aurora Serverless is an on-demand, autoscaling configuration for the MySQL-compatible and PostgreSQL-compatible editions of Amazon Aurora. An Aurora Serverless DB cluster automatically starts up, shuts down, and scales capacity up or down based on your application's needs. Aurora Serverless provides a relatively simple, cost-effective option for infrequent, intermittent, or unpredictable workloads. Read more in the Amazon Aurora User Guide.

**Q: Which versions of Amazon Aurora are supported for Aurora Serverless?**

Aurora Serverless is currently available for Aurora with MySQL 5.6 compatibility and for Aurora with PostgreSQL 10.7+ compatibility.

**Q: Can I migrate an existing Aurora DB cluster to Aurora Serverless?**

Yes, you can restore a snapshot taken from an existing Aurora provisioned cluster into an Aurora Serverless DB Cluster (and vice versa).

**Q: Can I explicitly set the capacity of an Aurora Serverless cluster?**

While Aurora Serverless automatically scales based on the active database workload, in some cases, capacity might not scale fast enough to meet a sudden workload change, such as a large number of new transactions. In these cases, you can set the capacity explicitly to a specific value with the AWS Management Console, the AWS CLI, or the RDS API.

**Q: Why isn't my Aurora Serverless DB Cluster automatically scaling?**

Once a scaling operation is initiated, Aurora Serverless attempts to find a scaling point, which is a point in time at which the database can safely complete scaling. Aurora Serverless might not be able to find a scaling point if you have long-running queries or transactions in progress, or temporary tables or table locks in use.

**Q: How am I billed for Aurora Serverless?**

In Aurora Serverless, database capacity is measured in Aurora Capacity Units (ACUs). You pay a flat rate per second of ACU usage, with a minimum of 5 minutes of usage each time the database is activated. Storage and I/O prices are the same for provisioned and Serverless configurations. View an Aurora Serverless pricing example.

**Parallel Query**

**Q: What is Amazon Aurora Parallel Query?**

Amazon Aurora Parallel Query refers to the ability to push down and distribute the computational load of a single query across thousands of CPUs in Aurora’s storage layer. Without Parallel Query, a query issued against an Amazon Aurora database would be executed wholly within one instance of the database cluster; this would be similar to how most databases operate.

**Q: What's the target use case?**

Parallel Query is a good fit for analytical workloads requiring fresh data and good query performance, even on large tables. Workloads of this type are often operational in nature.

**Q: What benefits does Parallel Query provide?**

Faster performance: Parallel Query can speed up analytical queries by up to 2 orders of magnitude.

Operational simplicity and data freshness: you can issue a query directly over the current transactional data in your Aurora cluster.

Transactional and analytical workloads on the same database: Parallel Query allows Aurora to maintain high transaction throughput alongside concurrent analytical queries.

**Q: What specific queries improve under Parallel Query?**

Most queries over large data sets that are not already in the buffer pool can expect to benefit. The initial version of Parallel Query can push down and scale out of the processing of more than 200 SQL functions, equijoins, and projections.

**Q: What performance improvement can I expect?**

The improvement to a specific query’s performance depends on how much of the query plan can be pushed down to the Aurora storage layer. Customers have reported more than an order of magnitude improvement to query latency.

**Q: Is there any chance that performance will be slower?**

Yes, but we expect such cases to be rare.

**Q: What changes do I need to make to my query to take advantage of Parallel Query?**

No changes in query syntax are required. The query optimizer will automatically decide whether to use PQ for your specific query. To check if a query is using PQ, you can view the query execution plan by running the EXPLAIN command. If you wish to bypass the heuristics and force Parallel Query for test purposes, use the aurora\_pq\_force session variable.

**Q: How do I turn the feature on or off?**

Parallel Query can be enabled and disabled dynamically at both the global and session level using the aurora\_pq parameter.

**Q: Are there any additional charges associated with using Parallel Query?**

No. You aren’t charged for anything other than what you already pay for instances, IO, and storage.

**Q: Since Parallel Query reduces IO, will turning it on reduce my Aurora IO charges?**

No, IO costs for your query are metered at the storage layer, and will be the same or larger with Parallel Query turned on. Your benefit is the improvement in query performance.

There are two reasons for potentially higher IO costs with Parallel Query. First, even if some of the data in a table is in the buffer pool, PQ requires all data to be scanned at the storage layer, incurring IO. Second, a side effect of avoiding contention in the buffer pool is that running a PQ query does not warm up the buffer pool. As a result, consecutive runs of the same PQ query will incur the full IO cost.

**Q: What versions of Amazon Aurora support Parallel Query?**

Parallel Query is available for the MySQL 5.6-compatible version of Amazon Aurora, starting with v1.18.0. We plan to extend Parallel Query to Aurora with MySQL 5.7 compatibility, and to Aurora with PostgreSQL compatibility.

**Q: Is Parallel Query available with all instance types?**

No. At this time, you can use Parallel Query with instances in the R\* instance family.

**Q: Is Parallel Query compatible with all other Aurora features?**

Not initially. At this time, you can only turn it on for database clusters that aren't running the Serverless or Backtrack features. Further, it doesn’t support functionality specific to Aurora with MySQL 5.7 compatibility.

**Q: If Parallel Query speeds up queries with only rare performance losses, should I simply turn it on for all the time?**

No. While we expect Parallel Query to improve query latency in most cases, you may incur higher IO costs. We recommend that you thoroughly test your workload with the feature enabled and disabled; once you're convinced that Parallel Query is the right choice, you can rely on the query optimizer to automatically decide which queries will use PQ. In the rare case when the optimizer doesn’t make the optimal decision, you can override the setting.

**Q: Can Aurora Parallel Query replace my data warehouse?**

Aurora Parallel Query is not a data warehouse, and doesn’t provide the functionality typically found in such products. It’s designed to speed up query performance on your relational database, and is suitable for use cases such as operational analytics, when you need to perform fast analytical queries on fresh data in your database.