Category	Exception Name
Miscellaneous	ActionNotAllowedException
Apex Code Errors	APEXException
Apex Code Errors	ApexPageMessageException
Apex Code Errors,Business Logic & Workflow Errors	ApexTriggerException
Business Logic & Workflow Errors	ApprovalException
Apex Code Errors	AssertException
Apex Code Errors	AsyncException
Authentication & Security	AuthProviderException
Apex Code Errors	BatchableContextException
Apex Code Errors	BatchableException
Integration Errors	CalloutException
Integration Errors	CalloutTimeoutException
Limit Errors	CpuLimitException
Apex Code Errors	CustomValidationException
Apex Code Errors	Database.SaveResultException
Apex Code Errors	Database.StatefulException
Apex Code Errors	DatabaseException
Deployment & Data Management	DataLoaderException
Deployment & Data Management	DataTranslationException
Deployment & Data Management	DeploymentException
Apex Code Errors	DmlException
Apex Code Errors	DmlExternalObjectException
Deployment & Data Management	DuplicateManagementException
Deployment & Data Management Errors	DuplicateRecordException
Deployment & Data Management	EmailException
Deployment & Data Management	EmailTemplateException
Miscellaneous	EncryptionException
Business Logic & Workflow Errors	EventBusException
Integration Errors	ExternalObjectException
Apex Code Errors	FieldCustomValidationException
Deployment & Data Management	FileDownloadException
Business Logic & Workflow Errors	FunctionException
Apex Code Errors	GovernorLimitException
Apex Code Errors	HeapSizeLimitException
Integration Errors	JSONException
Integration Errors	JsonGeneratorException
Authentication & Security	LicenseException
Apex Code Errors	LimitExceededException
Apex Code Errors	LimitException
Apex Code Errors	ListException
Authentication & Security	LoginIpRangeException
Session & Platform Errors	MalformedRequestException
Session & Platform Errors	MalformedSearchException
Apex Code Errors	MathException
Apex Code Errors	MixedDmlException
Miscellaneous	MobilePushException
Authentication & Security	NoAccessException
Apex Code Errors	NullPointerException

Authentication & Security	OAuthException
Authentication & Security	PermissionSetException
Session & Platform Errors	PlatformException
Miscellaneous	ProcessException
Apex Code Errors	QueryException
Apex Code Errors	QueryTooComplicatedException
Apex Code Errors	RegexException
Integration Errors	RemoteException
Authentication & Security	RequiredFeatureMissing
Deployment & Data Management	RetryableException
Miscellaneous	SearchException
Miscellaneous	SearchNotFoundException
Apex Code Errors	SerializationException
Session & Platform Errors	SessionExpiredException
Integration Errors	SiteExternalEventException
Apex Code Errors	SObjectException
Miscellaneous	SocialException
Integration Errors	SocketTimeoutException
Apex Code Errors	StringException
Integration Errors	System.CalloutException
Apex Code Errors	TimeZoneException
Apex Code Errors	TransactionException
Apex Code Errors	TypeException
Miscellaneous	UnsupportedOperationException
Visualforce	VisualforceException
Business Logic & Workflow Errors	WorkflowException
Integration Errors	X509CertificateException
Miscellaneous	XmlException

Exception Message
Action is not allowed for this operation
An unknown error occurred in the Apex code
An error occurred with a page message
An error occurred in the trigger
An error occurred during the approval process
Assertion failed in Apex code
An error occurred with the asynchronous operation
An error occurred with the authentication provider
An error occurred while processing a batch context
An error occurred while executing a batch operation
An error occurred during an HTTP callout
The callout timed out while waiting for a response
Apex CPU time limit exceeded
Custom validation failed
Save operation failed for one or more records
An error occurred in a stateful batch process
An error occurred while performing a database operation
An error occurred with the data loader
An error occurred while translating data
An error occurred during deployment
An error occurred with a DML operation
Error occurred with external object DML operations
Duplicate record found during record processing
Duplicate record detected
Error occurred while sending an email
An error occurred with the email template
An error occurred during encryption or decryption
An error occurred with the event bus
An error occurred with an external object
Custom field validation failed
An error occurred while downloading the file
An error occurred with the function
A governor limit was exceeded
Heap size limit exceeded
An error occurred while parsing JSON
An error occurred while generating JSON
License limits were exceeded or unavailable
A limit was exceeded during processing
A limit was exceeded for a specific operation
An error occurred with a list operation
The login IP is outside the allowed range
The request format was invalid
The search query is malformed
An error occurred during a mathematical operation
Mixed DML operation involving setup and non-setup objects
An error occurred with mobile push notifications
Insufficient permissions to perform the operation
Attempted to access a null reference

An error occurred with the OAuth authentication process
An error occurred with the permission set
An error occurred with the platform service
An error occurred in the process flow
An error occurred while executing the SOQL query
The SOQL query is too complicated
An error occurred with the regular expression
An error occurred with the remote operation
The required feature is not available
The operation can be retried
An error occurred while performing a search operation
No results found for the search query
An error occurred during object serialization
The session has expired
An error occurred with the external event
An error occurred with the SObject operation
An error occurred with social features
The socket connection timed out
An error occurred with string manipulation
An error occurred during an HTTP callout
An error occurred with time zone handling
An error occurred during the transaction
An error occurred with type conversion
The operation is unsupported
An error occurred while rendering the Visualforce page
An error occurred during the workflow process
An error occurred with the X509 certificate
An error occurred with XML processing

Description
Thrown when an operation is not permitted due to org settings or restrictions.
A generic exception for uncategorized Apex errors.
Thrown when a Visualforce page attempts to display a message and encounters an issue.
Thrown when exceptions occur within a trigger, such as logic failures or unhandled exceptions.
Thrown when issues occur with approval processes, such as missing records or invalid configurations.
Thrown when an assert() statement evaluates to false in test classes or logic validation.
Thrown when there are issues with asynchronous operations like future methods or batch Apex.
Thrown when issues occur with authentication providers (e.g., SAML, OAuth).
Thrown when the batch context is invalid or corrupted.
Thrown when errors occur in batch Apex, such as issues in start, execute, or finish methods.
Thrown when issues occur during external web service callouts, such as timeouts or invalid endpoints.
Thrown when a web service callout exceeds the maximum timeout duration.
Thrown when the Apex code execution exceeds the allowed CPU time limit (10,000 ms).
Thrown when a validation rule or trigger prevents DML operations.
Thrown when a DML operation returns partial success and partial failure.
Thrown for errors in batch processes that use the Database.Stateful interface.
Thrown for database-related errors not covered by other exceptions.
Thrown when errors occur during data import/export operations via the Salesforce Data Loader.
Thrown when data translation fails in a multilingual org.
Thrown during issues with metadata deployment (e.g., CI/CD pipelines).
Thrown when issues occur during DML operations like insert, update, delete, or undelete.
Thrown when performing DML operations on external objects that are not allowed.
Thrown when duplicate rules or duplicate jobs encounter conflicts.
Thrown when attempting to insert or update records that violate duplicate rules.
Thrown when sending an email fails, such as exceeding the daily email limit or invalid email addresses.
Thrown when errors occur with email templates (e.g., invalid merge fields).
Thrown when encryption or decryption errors occur.
Thrown when publishing platform events fails.
Thrown when issues occur with external objects (Salesforce Connect).
Thrown when custom validation logic in triggers or classes fails.
Thrown when an error occurs during file download operations.
Thrown when issues occur in Salesforce Functions.
Thrown when a governor limit (e.g., DML, SOQL) is exceeded during runtime.
Thrown when the total memory allocated for a transaction exceeds the heap size limit.
Thrown during JSON serialization or deserialization errors, such as parsing malformed JSON strings.
Thrown when errors occur during JSON generation.
Thrown when accessing functionality or features not covered by the user's license.
Thrown when an action exceeds configurable org limits, like API limits or file storage.
Thrown when Salesforce governor limits are exceeded, such as too many SOQL queries or CPU time.
Thrown when performing invalid operations on lists, such as accessing an out-of-bounds index.
Thrown when a user tries to log in from an IP address outside the allowed range.
Thrown when a web service request is not correctly formed.
Thrown when an SOSL search string is not properly formatted.
Thrown for invalid mathematical operations, such as division by zero or exceeding number limits.
Thrown when attempting to mix setup and non-setup object DML in the same transaction.
Thrown for issues related to mobile push notifications.
Thrown when a user attempts to access a resource they do not have permission to view or modify.
Thrown when attempting to access a method or property of a null object.

Thrown during issues with OAuth authentication in connected apps.
Thrown for issues with permission set assignments or configurations.
Thrown for general platform-related issues not covered by other exceptions.
Thrown when issues occur in process builders or flows.
Thrown when SOQL or SOSL queries fail, such as invalid queries or non-existent records.
Thrown when a query is too complex for Salesforce to execute.
Thrown for errors in regular expressions.
Thrown when a remote system throws an error in response to a Salesforce integration.
Thrown when a feature required for execution is disabled or unavailable in the organization.
Thrown for transient errors in integrations, signaling the process can be retried.
Thrown when errors occur during SOSL search operations.
Thrown when no results are found for a search query.
Thrown when an object cannot be serialized or deserialized properly.
Thrown when a session expires and the user attempts to perform an action.
Thrown for issues with external events in Salesforce sites or Experience Cloud.
Thrown when there is an issue with accessing or manipulating SObject fields.
Thrown when issues occur with Salesforce Social Studio or social account integrations.
Thrown when a socket connection times out.
Thrown when performing invalid operations on strings, such as invalid substring or indexOf.
Thrown when an error occurs in HTTP callouts or other external system integrations.
Thrown when an invalid or unsupported time zone is used.
Thrown when attempting to perform an invalid operation within a transaction.
Thrown when attempting an invalid typecast between incompatible data types.
Thrown when an unsupported operation is attempted, such as calling a method not applicable to a certain d
Thrown when errors occur in Visualforce pages, such as invalid components or syntax errors.
Thrown when a workflow rule fails to execute correctly.
Thrown when certificate-related issues occur in integrations or authentication.
Thrown during errors in XML parsing or serialization.

Why It Occurs/When
Occurs when trying to perform restricted actions based on user permissions or org configuration.
Occurs during runtime issues not specifically covered by other exception types.
Occurs due to invalid message configurations in Visualforce.
Occurs when a trigger contains unhandled exceptions or fails during DML operations.
Occurs during errors in approval process actions or configurations.
Occurs when a test method fails to meet expected results.
Occurs due to invalid configuration or exceeding limits on asynchronous operations.
Occurs due to misconfigured authentication settings or expired tokens.
Occurs when accessing batch-related information outside its scope.
Occurs when logic in batch methods fails or when handling large datasets incorrectly.
Occurs due to invalid endpoints, payload issues, or timeout errors in HTTP callouts.
Occurs due to slow external services or network issues.
Occurs when inefficient code or complex logic takes too long to execute.
Occurs when a record does not meet validation criteria.
Occurs during Database.insert or Database.update operations with allOrNone set to false.
Occurs when stateful batch logic encounters serialization or state management issues.
Occurs when there are issues during database operations like locking or invalid queries.
Occurs due to invalid file formats, incorrect mappings, or restricted access.
Occurs due to invalid translations or unsupported language codes.
Occurs when deployment validations fail or metadata conflicts arise.
Occurs due to validation rule failures, record locking, missing required fields, or permissions issues
Occurs when unsupported operations (e.g., delete) are attempted on external objects.
Occurs when a duplicate rule prevents the creation or update of records.
Occurs due to duplicate management rules configured in Salesforce.
Occurs when email addresses are invalid or the organization exceeds the daily email limit.
Occurs when email templates contain invalid or missing merge fields or other formatting issues.
Occurs due to issues with encrypted fields, key rotations, or unsupported encryption algorithms.
Occurs due to exceeding platform event limits or invalid event data.
Occurs due to invalid external data source configuration or access issues.
Occurs when custom validation logic doesn’t pass.
Occurs due to invalid file URLs or permission issues.
Occurs due to misconfigured or failing Salesforce Functions (e.g., runtime issues).
Occurs due to inefficient code that violates Salesforce governor limits.
Occurs when large collections, objects, or queries consume excessive memory.
Occurs when trying to parse or deserialize invalid JSON data or missing required fields.
Occurs when attempting to serialize unsupported data types or circular references.
Occurs when a user’s license doesn’t support the requested feature or API.
Occurs when configured thresholds (like API calls, file size, etc.) are breached.
Occurs when exceeding governor limits (e.g., 100 SOQL queries or 10 seconds of CPU time).
Occurs when trying to access an index that doesn’t exist or performing operations on empty lists.
Occurs when IP restrictions are configured in the org or profile.
Occurs due to invalid SOAP or REST requests.
Occurs when the SOSL search syntax is invalid or unsupported.
Occurs during division by zero or when performing calculations on invalid values.
Occurs when attempting DML operations on both user records and custom objects in the same tran
Occurs when mobile notifications fail due to invalid configurations or expired device tokens.
Occurs when a user lacks permissions or sharing rules to access a specific resource.
Occurs when a variable or object has not been initialized and is accessed.

Occurs when OAuth tokens are invalid, expired, or misconfigured.
Occurs when permission sets conflict or exceed license limits.
Occurs during issues with platform services or APIs.
Occurs due to invalid flow configurations or runtime errors in process builders.
Occurs due to invalid queries, missing required filters, or attempting to access null query results.
Occurs due to overly complex queries with too many joins or nested conditions.
Occurs due to invalid regex patterns or inputs that fail regex matching.
Occurs when external systems fail or send error responses.
Occurs when attempting to use a feature that is not enabled in the Salesforce org.
Occurs when external services fail intermittently.
Occurs due to invalid SOSL syntax, missing filters, or exceeding search limits.
Occurs when a search query doesn't match any records.
Occurs when objects contain invalid fields or unsupported data types during serialization.
Occurs when the user session times out or becomes invalid.
Occurs during errors in handling external events or invalid configurations.
Occurs when attempting to access fields that don't exist or are unavailable due to field-level securit
Occurs due to invalid social media credentials or integration settings.
Occurs during long-running external connections.
Occurs due to invalid string manipulation, like accessing invalid ranges or missing substrings.
Occurs due to endpoint issues, connection timeouts, or malformed HTTP requests.
Occurs due to incorrect or invalid time zone values in Apex code or integrations.
Occurs during invalid commits, rollbacks, or when transactions conflict.
Occurs when trying to cast one data type to another incompatible type.
Occurs when using a method not allowed for a specific data type or unsupported functionality.
Occurs due to incorrect syntax, invalid components, or missing controller methods in Visualforce pa
Occurs due to invalid workflow rule configuration or conditions.
Occurs when certificates are expired, invalid, or mismatched.
Occurs when XML data is malformed or contains invalid characters.

Resolving Methods
- Review org settings and user permissions.- Ensure compliance with organization restrictions.
- Enable detailed debug logs to investigate the root cause.
- Debug Visualforce message logic and ensure proper syntax.
- Use error handling in triggers. - Bulkify trigger logic and add proper exception messages.
- Validate approval process criteria and ensure all referenced records are available.
- Verify preconditions in test methods. - Ensure logic is correct before assertions.
- Ensure correct usage of future and batch methods.- Monitor limits and debug logs for asynchronous jobs
- Verify authentication provider setup.- Refresh tokens as needed.
- Ensure batch operations are correctly scoped and structured.
- Add error handling in batch methods. - Log errors for debugging and retry failed operations.
- Validate endpoints and request structure. - Implement retries for transient errors.
- Increase HttpRequest.setTimeout.- Optimize callout payload and retry failed requests.
- Optimize loops and reduce nested logic. - Use Queueable or Batch Apex for heavy processing.
- Review validation rules and triggers.- Test inputs against validation logic.
- Loop through SaveResult to handle errors.- Log and report failures.
- Avoid storing large or unsupported objects in batch classes.
- Use try-catch for transactions.- Avoid locking records unnecessarily.
- Validate data files and mappings.- Check field-level security and permissions.
- Ensure translation workbench settings are correct.- Validate language codes.
- Validate metadata before deployment.- Use pre-deployment checks and test classes.
- Use Database.saveResult to capture partial successes. - Validate field values before DML operations.
- Ensure external objects allow the operation.- Check external data source capabilities.
- Adjust duplicate rules or bypass them using Database.DMLOptions.
- Use Database.DMLOptions to bypass duplicate rules or adjust rules to allow certain exceptions.
- Validate email addresses. - Ensure daily email limits are not exceeded.
- Validate email template syntax and merge fields.
- Ensure correct encryption setup and keys.- Avoid unsupported operations on encrypted fields.
- Validate platform event schema.- Avoid exceeding daily event limits.
- Validate external object schema and data source configuration.
- Ensure input values satisfy validation rules.- Add meaningful error messages.
- Validate file URL.- Check user permissions for accessing files.
- Validate function deployment and runtime configuration.- Use proper logging mechanisms.
- Bulkify operations.- Optimize SOQL/DML calls and avoid queries in loops.
- Query fewer records using LIMIT. - Process records in chunks or use asynchronous Apex.
- Validate JSON structure before processing. - Use JSON.deserializeStrict to catch malformed data.
- Ensure proper object structure.- Avoid unsupported or recursive data.
- Upgrade the license or adjust features to align with current licenses.
- Monitor limits and optimize usage through setup and best practices.
- Optimize SOQL/DML usage by avoiding queries in loops. - Bulkify operations and refactor code logic.
- Use list.size() to validate index ranges before accessing. - Avoid hardcoding indexes.
- Update IP ranges in profile settings.- Use login IP ranges sparingly to avoid user access issues.
- Validate request structure.- Use tools like Postman for testing payloads.
- Verify SOSL syntax and ensure search strings are valid.
- Validate divisor values to ensure they are not zero. - Use conditional checks before performing calculati
- Use asynchronous methods like @future to separate setup and non-setup DML operations.
- Validate mobile push configurations.- Ensure device tokens are valid and up-to-date.
- Check user permissions and profiles. - Ensure appropriate sharing rules and field-level security.
- Check for null values using if (object != null). - Initialize objects before usage.

- Verify OAuth configuration and tokens.- Regenerate tokens if necessary.
- Verify permission set assignments.- Ensure license limits are not exceeded.
- Investigate error logs and contact Salesforce Support if needed.
- Debug the process builder or flow.- Ensure input data matches required schema.
- Validate query inputs. - Use try-catch blocks to handle empty results or invalid queries.
- Simplify queries by reducing joins and conditions.- Use indexed fields for better performance.
- Test regex patterns before using them.- Handle invalid inputs gracefully.
- Validate external system availability and request payloads.- Handle integration errors gracefully.
- Check org features and enable the required feature via Salesforce Setup.
- Implement retry logic with backoff strategies for external integrations.
- Validate SOSL search strings. - Limit search results to avoid exceeding limits.
- Handle empty search results gracefully.- Display a "no results found" message to users.
- Validate object structure before serialization. - Use custom serializers if needed.
- Ensure proper session timeout settings.- Refresh the session programmatically if possible.
- Debug site event handlers and ensure external configurations are valid.
- Verify field access and ensure field names are spelled correctly. - Check field-level security.
- Validate social media credentials.- Check integration configurations.
- Increase timeout settings.- Ensure external services respond in time.
- Check string length before performing operations. - Validate input values for string operations.
- Validate request payloads and endpoints. - Use HttpRequest.setTimeout to manage timeouts effectively
- Validate time zone inputs and use TimeZone class for conversions.
- Ensure proper use of Savepoint and rollback.
- Use instanceof to verify compatibility before typecasting.
- Check API documentation for supported methods. - Avoid using deprecated methods.
- Validate Visualforce page syntax. - Test pages thoroughly in all scenarios.
- Validate workflow rules and criteria.- Debug workflows using the Debug Log.
- Renew or update expired certificates.- Ensure proper certificate configuration.
- Validate XML structure. - Use proper XML libraries for parsing and encoding.

s.

ions.

Error Name
DmlException: Insert Failed
DmlException: Update Failed
DmlException: Delete Failed
DmlException: Undelete Failed
DmlException: Record Access Lock Row
DmlException: INVALID_CROSS_REFERENCE_KEY
DmlException: FIELD_INTEGRITY_EXCEPTION
DmlException: REQUIRED_FIELD_MISSING
DmlException: DUPLICATE_VALUE
DmlException: CANNOT_INSERT_UPDATE_ACTIVATE_ENTITY
DmlException: QUERY_TIMEOUT
DmlException: INVALID_OPERATION
DmlException: INVALID_TYPE
DmlException: CANNOT_DELETE
DmlException: INVALID_ID_FIELD
DmlException: UNABLE_TO_LOCK_ROW
DmlException: SYSTEM_LIMIT_EXCEPTION
DmlException: FIELD_CUSTOM_VALIDATION_EXCEPTION
DmlException: MALFORMED_QUERY
DmlException: SOBJECT_TYPE_ERROR

Description
The insert operation on a record failed. Causes: Missing required fields, validation rule failure, unique constraint vi
The update operation on a record failed. Causes: Invalid data, field-level security restrictions, updates to read-only
The delete operation on a record failed. Causes: Parent-child relationship constraints, record in use, or foreign key
The undelete operation on a record failed. Causes: Record permanently deleted, permission issues, or data integrit
The record is locked and cannot be accessed due to concurrent processes. Causes: Record locked by another use
A reference to a record that does not exist or is invalid. Causes: Invalid record ID or missing parent record.
The field value violates a field integrity constraint. Causes: Invalid data type, improper value, or field out of range.
A required field was not provided. Causes: Missing required field in insert or update operation.
The DML operation tried to insert a duplicate value. Causes: Violation of unique constraints, duplicate record rules.
A DML operation on a record failed due to a trigger or entity activation issue. Causes: Trigger logic error, activatior
A query exceeded the time limit during DML operation. Causes: Complex query, too many records processed.
The operation is invalid for the record or object. Causes: Trying to perform an operation not supported on the recor
The operation referenced an incorrect object or field type. Causes: Incorrect field data type or non-existent field.
The delete operation could not be performed. Causes: Record protected by system rules, active workflows, or appr
The operation tried to use an invalid field as an ID. Causes: Incorrectly referencing an object field as an ID.
The record could not be locked because it was already locked by another process. Causes: Conflicting DML operat
The operation exceeded a system limit. Causes: Exceeded governor limits like heap size, query rows, or DML state
A custom validation rule failed. Causes: Custom validation rules on the object being modified.
The query used in the DML operation is malformed. Causes: Incorrect SOQL or query syntax.
An invalid object type was referenced. Causes: Incorrectly specifying an object type that doesn't exist or is inacces

Resolve
Ensure all required fields are populated. Check validation rules and duplicate record rules.
Ensure data is valid and user has the appropriate field-level security. Check if fields are editable.
Check for relationships preventing deletion, or if the record is in use by another process.
Ensure the record has not been permanently deleted. Check permissions and data integrity.
Retry the operation after a delay, or investigate other processes locking the record.
Verify the existence of the referenced record or correct the ID.
Ensure the field values are correct and adhere to field requirements (data type, picklist values, etc.).
Provide values for all required fields. Check the object schema for mandatory fields.
Ensure uniqueness of the data or handle the duplicate records via logic (e.g., upsert).
Debug and fix the trigger or activation logic. Ensure that the triggers are not conflicting with the DML op
Optimize the query to reduce the number of records processed, and avoid complex joins.
Ensure the operation is supported by the record or object. Validate the allowed operations for the object
Validate the field type in the operation and ensure correct references to fields and objects.
Disable approval processes or workflows, or check system restrictions before deletion.
Correct the field references to ensure valid IDs are used for the operation.
Implement retry logic to wait and attempt the operation again.
Refactor code to reduce resource usage, like query size or DML operations.
Review and correct any custom validation rules. Ensure that data meets the validation conditions.
Correct the query syntax. Ensure the SOQL query is valid.
Ensure the object type is valid and accessible. Check object names and API names.

When and Why It Occurs
Happens when inserting a record that violates constraints or missing data.
Occurs when updating fields that are read-only or not accessible.
Happens when attempting to delete a record that is linked to others or in use.
Occurs when trying to restore a record that has been permanently deleted.
Happens in environments where multiple processes try to modify the same record simultaneously.
Occurs when a record references an invalid or non-existent parent or related record.
Happens when inserting or updating data that does not meet the field constraints.
Occurs when a required field is not included in the operation.
Happens when inserting or updating records that violate unique constraints.
Occurs when there is an issue in trigger logic or related process causing the failure.
Happens when a query takes too long to execute due to the number of records or complexity.
Occurs when performing an operation not supported by the object type.
Happens when referencing an incorrect or non-existent field or object.
Occurs when deleting records that are part of active workflows or approval processes.
Happens when a non-ID field is mistakenly used as an ID in the DML operation.
Happens when two processes try to modify the same record at the same time.
Occurs when a system limit is exceeded during a transaction.
Happens when custom validation rules prevent the DML operation from completing.
Occurs when the SOQL query used in the DML operation contains syntax errors.
Happens when an incorrect or inaccessible object type is referenced in the DML operation.

Error Name
AsyncException: Future Method Timeout
AsyncException: Too Many Future Calls
AsyncException: Callout Exception in Future
AsyncException: Queueable Job Timeout
AsyncException: Too Many Queueable Jobs
AsyncException: Chained Job Limit Exceede
AsyncException: Batch Job Timeout
AsyncException: Too Many Batch Jobs
AsyncException: Scheduled Job Limit Excee
AsyncException: Batch Apex Query Limit
AsyncException: System Shutdown
AsyncException: Mixed DML Operation
AsyncException: CronTrigger Limit Exceeded
AsyncException: Too Many Async Calls

Description
A @future method execution exceeded the time limit.
More than the allowed number of @future method calls (200 per transaction) were invoked.
A callout was attempted in a @future method without marking it as callout=true.
A Queueable job execution exceeded the time limit.
More than 50 Queueable jobs were added to the queue in a single transaction.
A chained Queueable job exceeded the maximum limit of 50 chainable jobs.
A batch job execution exceeded the allowed processing time for a single batch.
The maximum number of batch jobs (5 active or 100 queued) was exceeded.
The maximum number of scheduled jobs (100) was exceeded.
A batch class's query in the start() method exceeds the limit of 50 million records.
An async job was interrupted due to a Salesforce system shutdown or maintenance.
A DML operation on setup and non-setup objects was attempted in an asynchronous context.
A scheduled Apex job cannot be scheduled because the CronTrigger limit (100 jobs) was exceeded.
The maximum number of asynchronous calls (e.g., future, batch, Queueable) was exceeded for a tra

Resolve
Optimize the logic inside the @future method to reduce execution time. Avoid long-running operations like queries
Reduce the number of @future calls by batching operations or refactoring logic to avoid multiple calls.
Mark the future method with @future(callout=true) to enable HTTP callouts.
Optimize the logic in the Queueable job to avoid long-running operations. Consider breaking the job into smaller ch
Reduce the number of jobs added or refactor logic to avoid enqueuing multiple Queueable jobs.
Review and limit the chaining of Queueable jobs to stay within the limit.
Optimize the batch logic to process smaller records per batch. Reduce time-consuming operations like complex qu
Limit the number of active or queued batch jobs. Monitor and manage the batch queue effectively.
Reduce the number of scheduled jobs by combining or canceling unnecessary schedules.
Refactor the query to limit the number of returned records or use filters to reduce query size.
Retry the job after the system maintenance is complete.
Separate DML operations on setup and non-setup objects into different transactions.
Cancel unnecessary scheduled jobs to stay within the CronTrigger limit.
Limit the number of async calls by combining logic or reducing operations in the transaction.

When and Why It Occurs
Happens when a @future method runs for too long, exceeding Salesforce's time limits.
Occurs when a transaction tries to invoke too many future methods.
Happens when making an HTTP callout in a future method without proper annotation.
Occurs when the execution of a Queueable job exceeds the processing time limit.
Happens when a transaction exceeds the governor limit of 50 Queueable jobs.
Occurs when chaining too many Queueable jobs in a single transaction.
Happens when a batch class's execute() method takes too long to process records.
Happens when the system exceeds the maximum number of active or queued batch jobs.
Occurs when too many scheduled jobs are running or queued.
Happens when a batch job retrieves too many records in the start() method.
Occurs when Salesforce performs system maintenance or upgrades.
Happens when attempting a mixed DML operation, even in asynchronous code.
Happens when trying to schedule more than the allowed number of jobs.
Occurs when a transaction exceeds the governor limit for async calls.

Error Name
LimitException: Too Many SOQL Queries
LimitException: Too Many DML Statement
LimitException: Too Many SOSL Queries
LimitException: Too Many CPU Time
LimitException: Too Many Query Rows
LimitException: Too Many Callouts
LimitException: Too Many Future Calls
LimitException: Too Many Email Invocatio
LimitException: Too Many Apex Jobs
LimitException: Too Many Child Relationsh
LimitException: Heap Size Too Large
LimitException: Too Many Static Variables
LimitException: Too Many Email Recipient
LimitException: Too Many Describes
LimitException: Too Many Recursive Trigg
LimitException: Too Many Scripts
LimitException: Too Many Relationships
LimitException: Too Many Batch Jobs
LimitException: Too Many Rows Processe
LimitException: Too Many Fields

Description
The transaction exceeded the maximum limit of 100 SOQL queries.
The transaction exceeded the limit of 150 DML statements.
The transaction exceeded the maximum limit of 20 SOSL queries.
The transaction exceeded the CPU time limit (e.g., 10 seconds for synchronous Apex).
The SOQL query returned more than the allowed 50,000 rows.
The transaction exceeded the maximum limit of 100 HTTP callouts.
The transaction exceeded the limit of 200 future method calls.
More than 10 email invocations (via Messaging.sendEmail) were attempted.
The transaction tried to enqueue more than 50 Queueable jobs.
A query attempted to retrieve more than 100 child relationships in one parent-to-child query
The transaction exceeded the allowed heap size (6 MB for synchronous, 12 MB for asynchr
The transaction attempted to use more than 100 static variables.
More than 5,000 email recipients were specified in one email.
The transaction exceeded the maximum limit of 100 describe calls.
A recursive trigger was invoked more than 16 times.
The script statements exceeded the allowed limit (200,000 for synchronous, 1,000,000 for *a*
A query attempted to include more than 55 parent-to-child relationships.
More than 5 active batch jobs or 100 queued batch jobs were created.
A batch job attempted to process more than 50 million rows in total.
More than 500 fields were queried in a single SOQL query.

Resolve
Refactor code to use fewer queries, combine queries, or use relationships (e.g., parent-child qu
Combine DML operations into fewer statements (e.g., bulk insert/update).
Optimize code to reduce the number of SOSL queries or combine searches where possible.
Optimize loops, queries, and methods to reduce processing time. Avoid recursive logic.
Use filters to limit the number of rows returned or use batch Apex for large datasets.
Consolidate callouts, avoid unnecessary requests, or consider asynchronous methods like batc
Reduce the number of future calls by batching operations or refactoring logic.
Consolidate email logic or use email templates to reduce calls.
Limit the number of jobs added or refactor logic to reduce enqueuing multiple jobs.
Reduce the number of child relationships queried or use smaller chunks of related records.
Optimize data structures, avoid storing large objects in memory, and use transient fields in Visu
Reduce the number of static variables by combining or restructuring logic.
Reduce the number of recipients per email or send multiple smaller emails.
Cache describe calls or use Schema.describeSObjects() for bulk describe information.
Implement logic to prevent infinite recursion (e.g., static variables to track recursion depth).
Optimize logic to reduce script statements by refactoring loops and queries.
Simplify the query to reduce the number of relationships included.
Monitor and reduce batch job creation, or cancel unnecessary jobs.
Split the dataset into smaller batches or optimize batch job processing logic.
Reduce the number of fields retrieved in the query to below the limit.

When and Why It Occurs
Occurs when too many SOQL queries are executed within a single transaction.
Happens when executing more DML operations than the limit allows.
Occurs when more than 20 SOSL queries are executed in one transaction.
Happens when the transaction consumes too much CPU time.
Occurs when a query retrieves too many rows from the database.
Happens when performing too many HTTP callouts in a single transaction.
Happens when attempting to invoke too many future methods in a transaction.
Occurs when exceeding the email invocation limit per transaction.
Happens when adding too many Queueable jobs in one transaction.
Happens when a query includes too many child relationship queries.
Occurs when the transaction uses too much memory for variables or objects.
Happens when excessive static variables are declared in Apex code.
Occurs when exceeding the recipient limit for a single email.
Happens when too many schema describe calls are made in a transaction.
Happens when a trigger invokes itself repeatedly without proper termination.
Happens when the number of script statements executed exceeds the allowed limit.
Happens when a query is overly complex with too many relationships.
Happens when the batch job limit is exceeded.
Occurs when processing a very large dataset in a batch Apex job.
Happens when attempting to retrieve an excessive number of fields in a query.

Error Name
ListException: List Index Out of Bounds
ListException: Null List Access
ListException: Empty List Access
ListException: List Assignment Problem
ListException: Duplicate Records in Set/l
ListException: Invalid Operation on List
ListException: List Size Too Large
ListException: List Element Type Mismat
ListException: Unsupported Type in List
ListException: Invalid Query Result Acce
ListException: Adding Null Value

Description
Attempted to access an element in a list at an invalid index.
Tried to perform an operation on a null list.
Tried to access an element in an empty list.
Assigning incompatible types to a list variable (e.g., mixing types).
Tried to insert duplicate records into a set or list that requires unique values.
Performed an unsupported operation on a list, such as modifying a list in a for-each loop.
Attempted to create or manipulate a list that exceeds the size limit of 1,000,000 records.
Attempted to add or retrieve elements from a list with incompatible types.
Tried to store an unsupported data type in a list (e.g., custom objects with unsupported serial
Tried to access a list result from a SOQL query when no records were returned.
Attempted to add a null value to a list where nulls are not allowed.

Resolve
Validate the index before accessing a list. Ensure that the index is within the valid range of the list size.
Initialize the list before performing any operation. Use if (list != null) checks to avoid null pointer errors.
Check if the list is empty (list.isEmpty()) before attempting to access its elements.
Ensure the data types of the list and the assigned values match. Cast values explicitly if necessary.
Use a Set to remove duplicates or validate the records before adding them.
Avoid modifying a list inside a for-each loop. Instead, iterate using index-based loops or use temporary lists.
Reduce the size of the list by batching operations or using smaller datasets.
Ensure that the list type matches the type of the elements being added or retrieved.
Ensure the type being added is supported and serializable. For custom types, implement proper methods like t
Check if the query returned any results (if (!list.isEmpty())) before accessing elements.
Check for null values before adding elements to a list.

When and Why It Occurs
Occurs when trying to access an index that doesn’t exist (negative or exceeds list size).
Happens when attempting to manipulate a list variable that has not been initialized or is explicitly null
Occurs when trying to retrieve an element from a list that contains no items.
Happens when assigning a value of one type to a list expecting a different type.
Occurs when adding duplicate records to a List or Set that must maintain uniqueness.
Happens when performing an operation that modifies the list during iteration or unsupported list modif
Happens when working with extremely large datasets that exceed Salesforce’s limits for list size.
Happens when performing type-unsafe operations with lists, such as adding wrong data types.
Occurs when adding unsupported or unserializable data types to a list.
Happens when attempting to access records from a query that returned zero rows.
Occurs when trying to add a null value to a list that expects only non-null elements.

.

fications.

Error Name
NullPointerException: Accessing Null Value
NullPointerException: Null List Access
NullPointerException: Null Map Access
NullPointerException: Null Set Access
NullPointerException: SOQL Query Return N
NullPointerException: Null Variable in Loop
NullPointerException: Null Field Reference
NullPointerException: Null Function Parame
NullPointerException: Null Apex Pages Valu
NullPointerException: Null Static Resource

Description
Attempted to access a method or property on a null object reference.
Tried to perform an operation on a null list.
Tried to access a key or value in a map that is null.
Tried to perform an operation on a null set.
Accessed results from a SOQL query when no records were returned.
A null object is used in a loop iteration or logic.
Attempted to access a field of an object that is null.
Passed a null value as a parameter to a method or function.
Tried to access a value from ApexPages.currentPage().getParameters() that
Tried to reference a static resource that is not initialized or missing.

Resolve
Check if the object is null before accessing its methods or properties using if (object != null).
Initialize the list before performing any operations. Use if (list != null) checks to avoid errors.
Ensure the map is initialized and check for key existence using map.containsKey(key) before
Initialize the set properly before performing any operations.
Use if (!list.isEmpty()) or if (list != null) to check the query result before accessing it.
Validate the objects in a loop or ensure initialization of variables before entering the loop.
Check if the parent object is null before accessing its fields using if (parentObj != null).
Validate all parameters before passing them to methods or provide default values.
Check if the parameter exists before accessing it using if (params.containsKey(key)).
Validate the existence of the static resource before referencing it.

When and Why It Occurs
Occurs when an object reference is not initialized and is accessed.
Happens when a list variable is not initialized or explicitly set to null.
Occurs when accessing a map that hasn’t been initialized or when trying to retrieve a non-existent k
Happens when attempting to access or modify a null set.
Occurs when a SOQL query doesn’t return any records, and you try to access the results without va
Happens when null values are included in collections or are directly referenced during loop operatio
Occurs when an object that contains fields has not been initialized.
Happens when a null argument is passed to a method expecting a non-null value.
Happens when accessing a page parameter that doesn’t exist in the request context.
Occurs when a static resource is deleted, missing, or not deployed properly in the environment.

ɛey.

ılidation.

ıns.

Error Name
QueryException: List has no rows for assignm
QueryException: Too many rows for assignme
QueryException: Non-Selective Query
QueryException: Aggregate Query Error
QueryException: Field not Selectable
QueryException: Too Many SOQL Queries
QueryException: Invalid Field
QueryException: Missing WHERE Clause
QueryException: Invalid Aggregate Function
QueryException: Invalid Relationship

Description
Attempted to assign a query result to a single sObject, but the query returned no rows.
Tried to assign a query result to a single sObject, but the query returned more than one ro
A query on a large dataset without using indexed fields caused a governor limit error.
Malformed or improperly written aggregate function in the SOQL query.
Tried to query a field that is not accessible or does not exist in the object.
Exceeded the limit of 100 SOQL queries in a single transaction.
Tried to query a field that does not exist or is incorrectly spelled.
Querying a large object table without a WHERE clause caused a governor limit.
Used an invalid or unsupported field in an aggregate query.
Used an incorrect relationship name in a query, such as a bad child-parent relationship ref

Resolve
Use a try-catch block or check if (!list.isEmpty()) before assignment. Alternatively, use List<sObject> to store
Use LIMIT 1 in the SOQL query to ensure only one record is returned, or use a List<sObject> for multiple resu
Add indexed fields to the query filter, such as Id, Name, or custom fields with External ID.
Ensure the aggregate query syntax is correct. Validate functions like SUM(), COUNT(), etc.
Check the field permissions and spelling in the schema. Use DescribeSObjectResult to verify fields.
Optimize the code by reducing SOQL queries inside loops and using bulkified queries.
Verify the field name in the object schema or use Schema.DescribeSObjectResult.
Add a filter condition in the WHERE clause to reduce the number of records queried.
Check if the field is suitable for the function and ensure the query syntax is correct.
Verify the relationship name in the object schema or use Schema.DescribeSObjectResult.

When and Why It Occurs
Happens when a SOQL query expects exactly one record but returns none.
Occurs when a query unexpectedly returns multiple rows, but you’re assigning it to a single variable.
Happens when querying a large object table (over 100,000 records) without indexed filters in WHERE cla
Occurs when using aggregate functions like COUNT(), MAX(), or SUM() with invalid syntax or grouping.
Happens when querying a field that is hidden, not API-accessible, or misspelled.
Happens when running too many SOQL queries in one execution context, typically inside loops or recurs
Happens when the field is misspelled, deleted, or renamed in the object schema.
Happens when querying large datasets without filters, violating non-selective query rules.
Happens when using an aggregate function like COUNT() or SUM() on unsupported field types.
Happens when referencing relationships incorrectly in SOQL, like child-to-parent or parent-to-child.

ause.

ive calls.

Error Name
FinalException: Final Method Cannot Be Overridden
FinalException: Final Class Cannot Be Inherited
FinalException: Final Variable Cannot Be Modified
FinalException: Attempted to Assign Final Field
FinalException: Final Local Variable Cannot Be Assig
FinalException: Final Parameter Cannot Be Reassigr
FinalException: Final Inner Class Cannot Be Instantia
FinalException: Final Class Inheritance Limitation
FinalException: Record is read-only

Description
Attempted to override a method that is marked as final.
Attempted to inherit from a class marked as final.
Attempted to modify a final variable after initialization.
Attempted to assign a value to a field marked as final.
Attempted to reassign a value to a local variable that is final.
Attempted to reassign a final parameter in a method.
Tried to instantiate an inner class that is marked as final in a way that is not allowed.
Attempted to inherit from a final class in a context that doesn’t allow it.
Attempted to modify a record that is read-only, usually because it's in a "final" state (e.g., locked or in an immutab

Resolve
Remove the final modifier from the method or avoid overriding the method. Avoid extending a final class or modify the class to not be final if inheritance is necessary. Ensure that the value of the final variable is assigned only once and not modified thereafter. Do not assign new values to final fields after they are initialized. Reassign only non-final local variables, as final variables can only be assigned once. Avoid reassigning final parameters inside a method, as they cannot be changed after they are initialized. Ensure the final inner class is instantiated properly, or consider removing the final modifier. Reconsider the design and avoid extending a final class or modify the class definition. Ensure that the record is not in a read-only state before attempting to modify it. Use Database.update() for

When and Why It Occurs
Occurs when you try to override a method that is declared as final, which cannot be overridden in subclasses.
Happens when trying to subclass a final class, which is not allowed.
Occurs when trying to change the value of a final variable after it has been assigned.
Happens when trying to assign a new value to a final field in a class or method.
Occurs when trying to assign a new value to a final local variable within a method.
Happens when trying to change the value of a method parameter marked as final.
Occurs when trying to instantiate a final inner class or subclass it incorrectly.
Happens when trying to extend a final class in a non-allowed context, such as when the class restricts inheritance.
Happens when trying to update or delete a record that is in a state where changes are not allowed (e.g., locked, clo

sed, or finalized).

Error Name
MathException: Division by Zero
MathException: Square Root of Negati
MathException: Logarithm of Negative
MathException: Exponent Overflow
MathException: Exponent Underflow
MathException: Invalid Input
MathException: Rounding Error
MathException: Trigonometric Error
MathException: Overflow in Factorial

Description
Attempted to divide a number by zero.
Attempted to calculate the square root of a negative number.
Attempted to calculate the logarithm of a negative number or zero.
Result of an exponent operation is too large for the data type to handle.
Result of an exponent operation is too small for the data type to handle.
Input to a mathematical function is not valid (e.g., NaN or unexpected data).
A rounding operation results in unexpected behavior due to floating-point precision
Attempted to perform a trigonometric operation on an invalid value.
Factorial calculation exceeds the maximum limit for the data type.

Resolve
Check the denominator before performing division (if (denominator != 0)).
Validate the input before calling Math.sqrt(). Ensure the number is greater than or equal to
Validate the input before calling Math.log(). Ensure the number is positive.
Use smaller exponent values or refactor the formula to avoid exceeding the limit.
Avoid using extremely small numbers or adjust the precision of calculations.
Validate all inputs before performing mathematical operations. Use isNaN() checks if neces
Use appropriate precision methods like Math.round(), Math.floor(), or Math.ceil() where app
Ensure that the input values to trigonometric functions like Math.sin(), Math.cos(), etc., are
Use an approximation for factorial calculations for large values or optimize the formula.

When and Why It Occurs
Occurs when dividing a number by zero, which is undefined mathematically.
Happens when using the Math.sqrt() method with a negative number as input.
Occurs when using the Math.log() or Math.log10() methods with non-positive values.
Happens when the result of a power operation (e.g., Math.pow()) exceeds the maximum limit for data
Occurs when using Math.pow() with very small exponent values, resulting in an underflow error.
Happens when the input to a math function is not a valid number or is corrupted.
Occurs due to limitations in how floating-point numbers are represented and rounded in computer sys
Happens when input values are out of the acceptable range for trigonometric operations.
Happens when calculating factorials for large numbers using iterative or recursive methods.

storage.

tems.