**Possible Causes When *Some* Topics Are Created, and Others Are Not**

**1. No Data in the Source Table**

If a table has **no new data**, the connector may **not create the corresponding topic**, even if it’s listed.



**Why?** Kafka Connect only creates the topic **when it emits a record**.



**Solution:**

* Insert a row in the missing source table
* Observe whether the topic gets created



* Use a tool like kafka-console-consumer to monitor the topic



**2. Primary Key / Offset Column Missing or Misconfigured**



In JDBC source connectors with incrementing or timestamp+incrementing modes, **each table must have the required offset column(s)**.



If a table **lacks an id or timestamp column**, or their values are null, **no data is captured**.



**Solution:**

* Make sure each table listed in table.whitelist has the **required incrementing.column.name** and/or timestamp.column.name
* Check logs for table-specific warnings

**3. Misconfigured Topic Routing (Transforms)**

If you're using RegexRouter or other transforms, it's possible the **transformed topic name is incorrect** or **conflicts with another topic**.



**Solution:**

* Temporarily remove the transforms config
* Let the connector create topics using default behavior
* Observe logs for final resolved topic names

**4. Errors on Specific Tables**

If a particular table causes a **schema issue** (e.g., unsupported data type), the connector **may skip it** silently or log an error.



**Solution:**

* Check connect logs (docker logs connect or confluent local services connect log)
* Look for errors like:



**5. Connector Filters or Include/Exclude Lists**

Check if you're using filters like:

"table.whitelist": "table1,table2",



"table.blacklist": "table3"



Or maybe:

"topics.regex": ".\*"

Filters can conflict or override each other!

**Solution:**

* Use only one of table.whitelist, topics, or topics.regex
* Ensure the table isn't accidentally filtered out