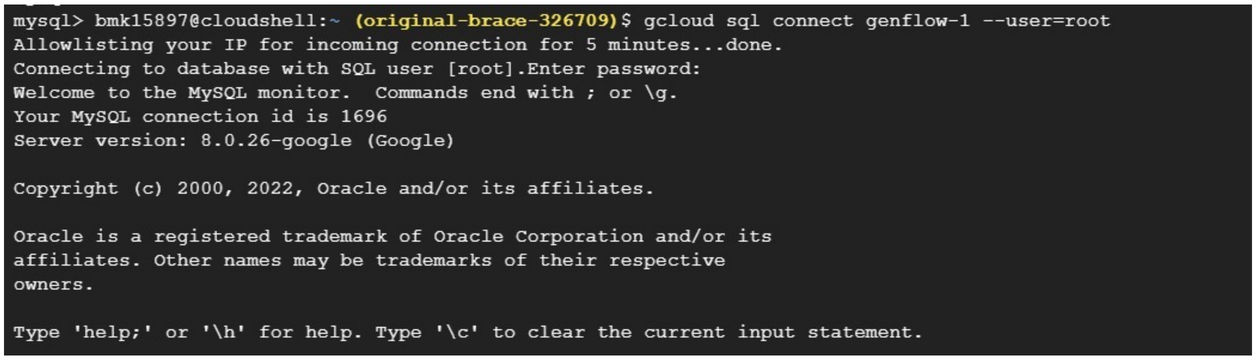
Project Stage 3

Screenshot of Connection to GCP:



```
mysql> bmk15897@cloudshell:~ (original-brace-326709)$ gcloud sql connect genflow-1 --user=root
Allowlisting your IP for incoming connection for 5 minutes...done.
Connecting to database with SQL user [root].Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1696
Server version: 8.0.26-google (Google)

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

Table DDL Commands

Table Name - Business

```
create table Business(
        businessId INTEGER Primary key,
        businessName VARCHAR(255),
        businessLogoUrl VARCHAR(255),
        businessAddress VARCHAR(255),
        businessContact VARCHAR(15)
);
```

Table Name - Users

```
create table Users (
        userId INTEGER primary key,
        businessId INTEGER references Businesses (businessId),
        userType ENUM('process-owner','business-owner'),
        emailAddress VARCHAR(255),
        pwdHash VARCHAR(255)
        isActive BOOLEAN,
        fName VARCHAR(255),
        lName VARCHAR(255),
        phNum VARCHAR(15)
);
```

Table Name - Departments

```
create table Departments (
   deptId INTEGER primary key,
   businessId INTEGER references Businesses (businessId),
   deptName VARCHAR(255),
   deptLoc VARCHAR(255),
   deptContact VARCHAR(255)
);
```

Table Name - UserDepartment

```
create table UserDepartment (
        userId INTEGER references Users (userId),
        deptId INTEGER references Departments (deptId),
        primary key(userId, deptId)
);
```

Table Name - UserPrivileges

```
create table UserPrivileges (
        userId INTEGER references Users (userId),
        privilege ENUM('process-owner','business-owner','read-only'),
        primary key(userId, privilege)
);
```

Table Name - Workflows

```
create table Workflows (
        wfName VARCHAR(50),
        wfId INTEGER primary key,
        wfDescription VARCHAR(255),
        businessId INTEGER references Businesses (businessId)
);
```

Table Name - Processes

```
create table Processes (
        processId INTEGER primary key,
        processName VARCHAR(50),
        seqNumber INTEGER,
        wfId INTEGER references Workflows (wfId)
);
```

Table Name - Parameters

```
create table Parameters(
        paramId INTEGER primary key,
        paramName VARCHAR(50),
        paramType VARCHAR(50),
        isOptional BOOLEAN,
        processId INTEGER references Processes (processId)
);
```

Table Name - WorkflowInstances

```
create table WorkflowInstances(
	wfInstanceId INTEGER primary key,
	createdDT DATETIME,
	updatedDT DATETIME,
	completedDT DATETIME,
	wfId INTEGER references Workflows (wfId)
);
```

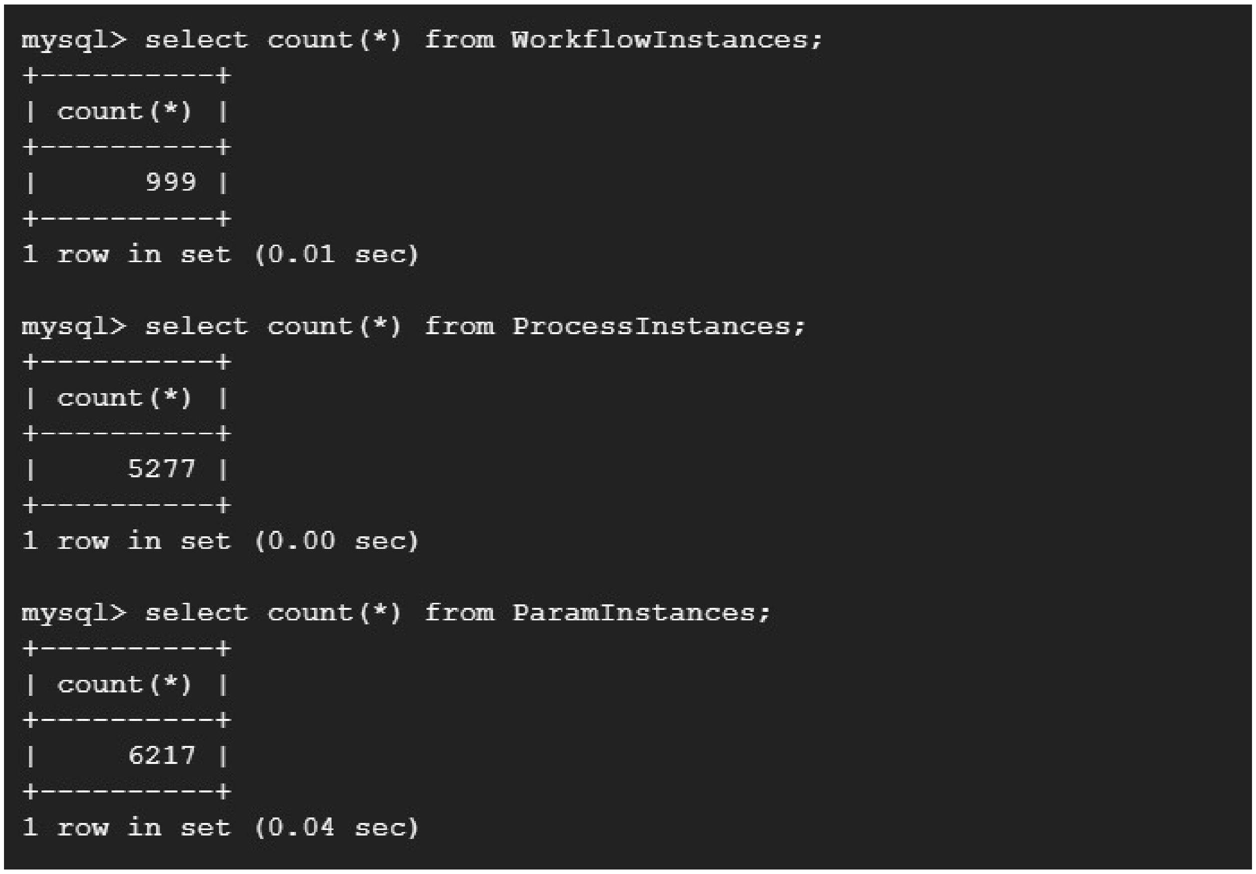
Table Name - ProcessInstances

```
create table ProcessInstances(
	processInstanceId INTEGER primary key,
	createdDT DATETIME,
	completedDT DATETIME,
	processId INTEGER references Processes (processId),
	wfInstanceId INTEGER references WorkflowInstances (wfInstanceId)
);
```

Table Name - ParamInstances

```
create table ParamInstances(
	paramVal VARCHAR(255),
	processInstanceId INTEGER references ProcessInstances (processInstanceId),
	paramId INTEGER references Parameters (paramId),
	primary key(processInstanceId,paramId )
);
```

Counts of Records in Tables



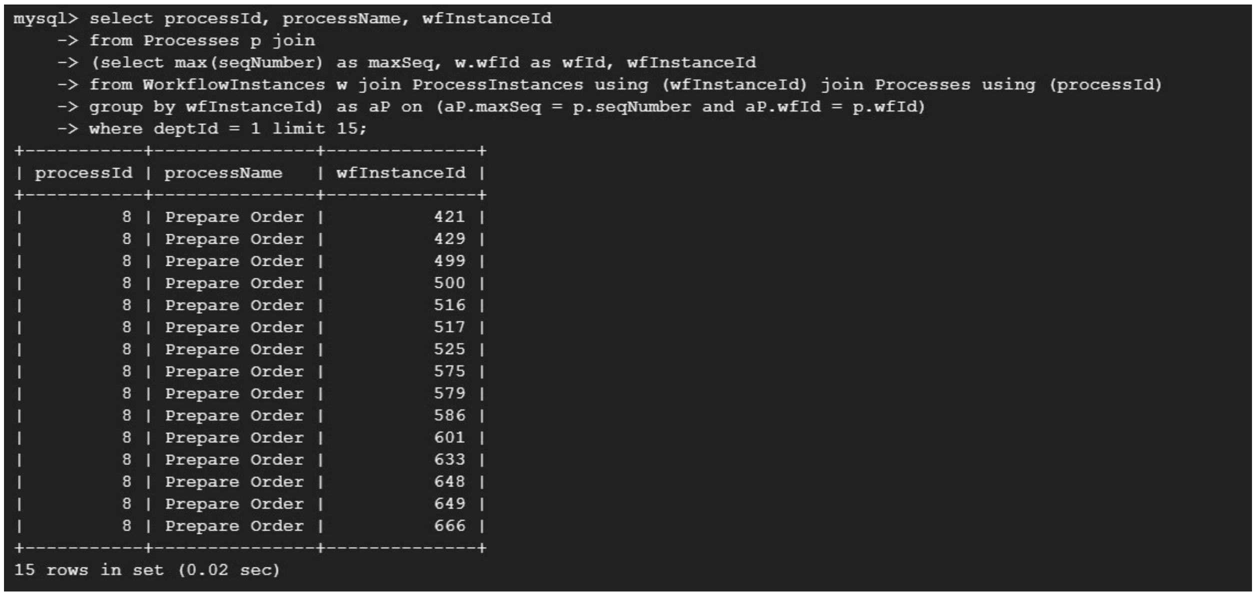
```
mysql> select count(*) from WorkflowInstances;
+----------+
| count(*) |
+----------+
|      999 |
+----------+
1 row in set (0.01 sec)

mysql> select count(*) from ProcessInstances;
+----------+
| count(*) |
+----------+
|     5277 |
+----------+
1 row in set (0.00 sec)

mysql> select count(*) from ParamInstances;
+----------+
| count(*) |
+----------+
|     6217 |
+----------+
1 row in set (0.04 sec)
```

Advanced Queries

Query 1

The current active processes for a department (department 1 considered)
Return type: Process ID, Process Name, Instance ID of the Workflow in which the process is currently active.

select processId, processName, wfInstanceId from Processes p join (select max(seqNumber) as maxSeq, w.wfId as wfId, wfInstanceId from WorkflowInstances w join ProcessInstances using (wfInstanceId) join Processes using (processId) group by wfInstanceId) as aP on (aP.maxSeq = p.seqNumber and aP.wfId = p.wfId) where deptId = 1;



```
mysql> select processId, processName, wfInstanceId
    -> from Processes p join
    -> (select max(seqNumber) as maxSeq, w.wfId as wfId, wfInstanceId
    -> from WorkflowInstances w join ProcessInstances using (wfInstanceId) join Processes using (processId)
    -> group by wfInstanceId) as aP on (aP.maxSeq = p.seqNumber and aP.wfId = p.wfId)
    -> where deptId = 1 limit 15;
+-----------+---------------+--------------+
| processId | processName   | wfInstanceId |
+-----------+---------------+--------------+
|         8 | Prepare Order |          421 |
|         8 | Prepare Order |          429 |
|         8 | Prepare Order |          499 |
|         8 | Prepare Order |          500 |
|         8 | Prepare Order |          516 |
|         8 | Prepare Order |          517 |
|         8 | Prepare Order |          525 |
|         8 | Prepare Order |          575 |
|         8 | Prepare Order |          579 |
|         8 | Prepare Order |          586 |
|         8 | Prepare Order |          601 |
|         8 | Prepare Order |          633 |
|         8 | Prepare Order |          648 |
|         8 | Prepare Order |          649 |
|         8 | Prepare Order |          666 |
+-----------+---------------+--------------+
15 rows in set (0.02 sec)
```

Query 2

Count of the current completed workflow instances.
Return type: Number of Completed Workflow Instances, Workflow Name associated with the Instance.

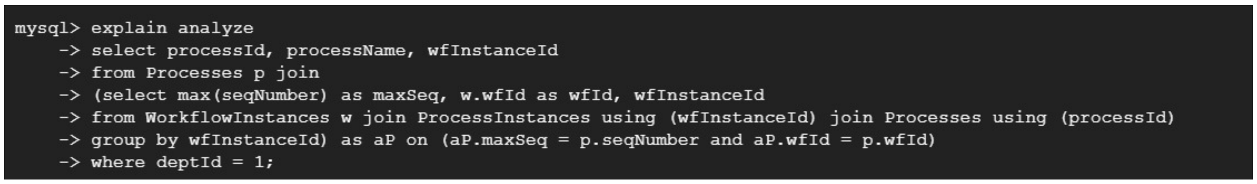
select count(wi.wfInstanceId) as completedWFInstances, wfName from Workflows w join WorkflowInstances wi using (wfId) where wi.completedDT != "0000-00-00 00:00:00" group by wfId;

```
mysql> select count(wi.wfInstanceId) as completedWFInstances, wfName
    -> from Workflows w join WorkflowInstances wi using (wfId)
    -> where wi.completedDT != "0000-00-00 00:00:00"
    -> group by wfId;
+----------------------+-------------------+
| completedWFInstances | wfName            |
+----------------------+-------------------+
|                  334 | Online Food Order |
|                  450 | Dine In Order     |
+----------------------+-------------------+
2 rows in set (0.01 sec)
```

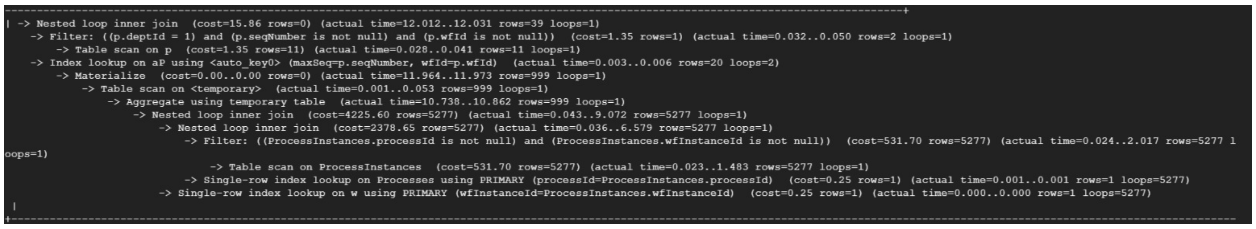
Index Designs

Query 1

- Initial run of ‘EXPLAIN ANALYZE’ without creating an index.



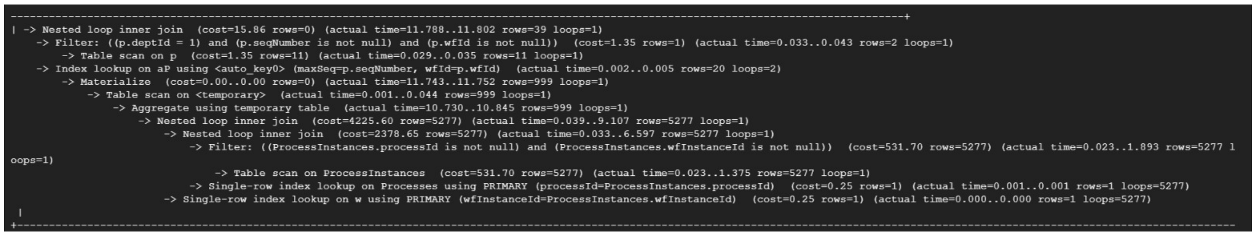
```
mysql> explain analyze
    -> select processId, processName, wfInstanceId
    -> from Processes p join
    -> (select max(seqNumber) as maxSeq, w.wfId as wfId, wfInstanceId
    -> from WorkflowInstances w join ProcessInstances using (wfInstanceId) join Processes using (processId)
    -> group by wfInstanceId) as aP on (aP.maxSeq = p.seqNumber and aP.wfId = p.wfId)
    -> where deptId = 1;
```



```
----------------------------------------------------------------------------------------------------------------------------------+
| -> Nested loop inner join  (cost=15.86 rows=0) (actual time=12.012..12.031 rows=39 loops=1)
    -> Filter: ((p.deptId = 1) and (p.seqNumber is not null) and (p.wfId is not null))  (cost=1.35 rows=1) (actual time=0.032..0.050 rows=2 loops=1)
        -> Table scan on p  (cost=1.35 rows=11) (actual time=0.028..0.041 rows=11 loops=1)
    -> Index lookup on aP using <auto_key0> (maxSeq=p.seqNumber, wfId=p.wfId)  (actual time=0.003..0.006 rows=20 loops=2)
        -> Materialize  (cost=0.00..0.00 rows=0) (actual time=11.964..11.973 rows=999 loops=1)
            -> Table scan on <temporary>  (actual time=0.001..0.053 rows=999 loops=1)
                -> Aggregate using temporary table  (actual time=10.738..10.862 rows=999 loops=1)
                    -> Nested loop inner join  (cost=4225.60 rows=5277) (actual time=0.043..9.072 rows=5277 loops=1)
                        -> Nested loop inner join  (cost=2378.65 rows=5277) (actual time=0.036..6.579 rows=5277 loops=1)
                            -> Filter: ((ProcessInstances.processId is not null) and (ProcessInstances.wfInstanceId is not null))  (cost=531.70 rows=5277) (actual time=0.024..2.017 rows=5277 l
oops=1)
                                -> Table scan on ProcessInstances  (cost=531.70 rows=5277) (actual time=0.023..1.483 rows=5277 loops=1)
                            -> Single-row index lookup on Processes using PRIMARY (processId=ProcessInstances.processId)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=5277)
                        -> Single-row index lookup on w using PRIMARY (wfInstanceId=ProcessInstances.wfInstanceId)  (cost=0.25 rows=1) (actual time=0.000..0.000 rows=1 loops=5277)
 |
+--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
```

Index Design 1

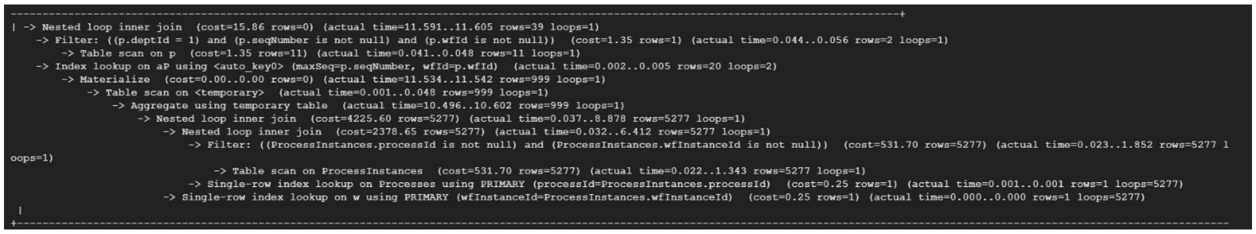
- We added an index on the column ***seqNum*** of the table Processes to check the performance of the query. The index reduced the total time required for the execution by 1 second. But we need more improvement, hence this index on its own isn’t sufficient and so we do not go ahead with this design.



```
----------------------------------------------------------------------------------------------------------------------------------+
| -> Nested loop inner join  (cost=15.86 rows=0) (actual time=11.788..11.802 rows=39 loops=1)
    -> Filter: ((p.deptId = 1) and (p.seqNumber is not null) and (p.wfId is not null))  (cost=1.35 rows=1) (actual time=0.033..0.043 rows=2 loops=1)
        -> Table scan on p  (cost=1.35 rows=11) (actual time=0.029..0.035 rows=11 loops=1)
    -> Index lookup on aP using <auto_key0> (maxSeq=p.seqNumber, wfId=p.wfId)  (actual time=0.002..0.005 rows=20 loops=2)
        -> Materialize  (cost=0.00..0.00 rows=0) (actual time=11.743..11.752 rows=999 loops=1)
            -> Table scan on <temporary>  (actual time=0.001..0.044 rows=999 loops=1)
                -> Aggregate using temporary table  (actual time=10.730..10.845 rows=999 loops=1)
                    -> Nested loop inner join  (cost=4225.60 rows=5277) (actual time=0.039..9.107 rows=5277 loops=1)
                        -> Nested loop inner join  (cost=2378.65 rows=5277) (actual time=0.033..6.597 rows=5277 loops=1)
                            -> Filter: ((ProcessInstances.processId is not null) and (ProcessInstances.wfInstanceId is not null))  (cost=531.70 rows=5277) (actual time=0.023..1.893 rows=5277 l
oops=1)
                                -> Table scan on ProcessInstances  (cost=531.70 rows=5277) (actual time=0.023..1.375 rows=5277 loops=1)
                            -> Single-row index lookup on Processes using PRIMARY (processId=ProcessInstances.processId)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=5277)
                        -> Single-row index lookup on w using PRIMARY (wfInstanceId=ProcessInstances.wfInstanceId)  (cost=0.25 rows=1) (actual time=0.000..0.000 rows=1 loops=5277)
 |
+--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
```

Index Design 2

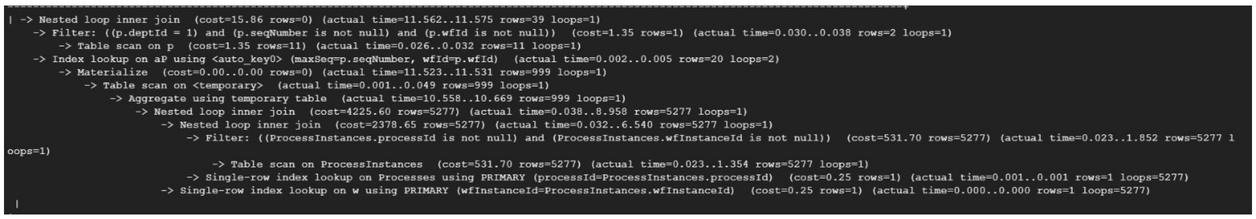
- We added an index on the column ***wfInstanceId*** of the table WorkflowInstances along with an index on the column ***seqNum*** of the table Processes to check the performance of the query. This combination of index didn’t reflect much improvement over the previous index. Hence, we did not go ahead with this design.



```
----------------------------------------------------------------------------------------------------------------------------------+
| -> Nested loop inner join  (cost=15.86 rows=0) (actual time=11.591..11.605 rows=39 loops=1)
    -> Filter: ((p.deptId = 1) and (p.seqNumber is not null) and (p.wfId is not null))  (cost=1.35 rows=1) (actual time=0.044..0.056 rows=2 loops=1)
        -> Table scan on p  (cost=1.35 rows=11) (actual time=0.041..0.048 rows=11 loops=1)
    -> Index lookup on aP using <auto_key0> (maxSeq=p.seqNumber, wfId=p.wfId)  (actual time=0.002..0.005 rows=20 loops=2)
        -> Materialize  (cost=0.00..0.00 rows=0) (actual time=11.534..11.542 rows=999 loops=1)
            -> Table scan on <temporary>  (actual time=0.001..0.048 rows=999 loops=1)
                -> Aggregate using temporary table  (actual time=10.496..10.602 rows=999 loops=1)
                    -> Nested loop inner join  (cost=4225.60 rows=5277) (actual time=0.037..8.878 rows=5277 loops=1)
                        -> Nested loop inner join  (cost=2378.65 rows=5277) (actual time=0.032..6.412 rows=5277 loops=1)
                            -> Filter: ((ProcessInstances.processId is not null) and (ProcessInstances.wfInstanceId is not null))  (cost=531.70 rows=5277) (actual time=0.023..1.852 rows=5277 l
oops=1)
                                -> Table scan on ProcessInstances  (cost=531.70 rows=5277) (actual time=0.022..1.343 rows=5277 loops=1)
                            -> Single-row index lookup on Processes using PRIMARY (processId=ProcessInstances.processId)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=5277)
                        -> Single-row index lookup on w using PRIMARY (wfInstanceId=ProcessInstances.wfInstanceId)  (cost=0.25 rows=1) (actual time=0.000..0.000 rows=1 loops=5277)
 |
+--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
```

Index Design 3

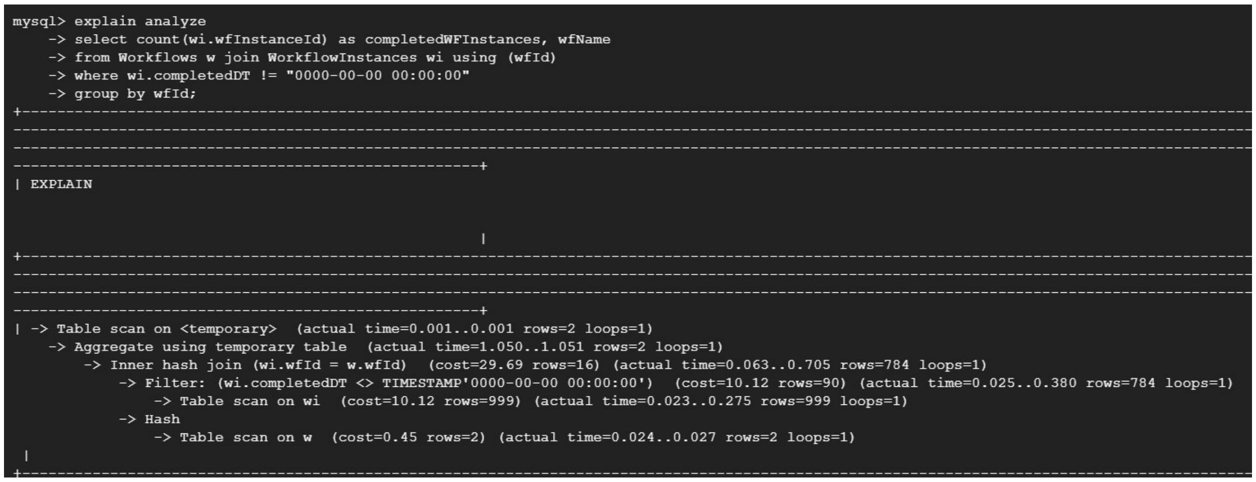
- We added an index on the column ***wfInstanceId*** of the table WorkflowInstances along with an index on the column ***wfId*** of the table Processes to check the performance of the query. This combination of indices had a much better performance over the other indices. **Hence, we moved ahead with this design.**



```
| -> Nested loop inner join  (cost=15.86 rows=0) (actual time=11.562..11.575 rows=39 loops=1)
    -> Filter: ((p.deptId = 1) and (p.seqNumber is not null) and (p.wfId is not null))  (cost=1.35 rows=1) (actual time=0.030..0.038 rows=2 loops=1)
        -> Table scan on p  (cost=1.35 rows=11) (actual time=0.026..0.032 rows=11 loops=1)
    -> Index lookup on aP using <auto_key0> (maxSeq=p.seqNumber, wfId=p.wfId)  (actual time=0.002..0.005 rows=20 loops=2)
        -> Materialize  (cost=0.00..0.00 rows=0) (actual time=11.523..11.531 rows=999 loops=1)
            -> Table scan on <temporary>  (actual time=0.001..0.049 rows=999 loops=1)
                -> Aggregate using temporary table  (actual time=10.558..10.669 rows=999 loops=1)
                    -> Nested loop inner join  (cost=4225.60 rows=5277) (actual time=0.038..8.958 rows=5277 loops=1)
                        -> Nested loop inner join  (cost=2378.65 rows=5277) (actual time=0.032..6.540 rows=5277 loops=1)
                            -> Filter: ((ProcessInstances.processId is not null) and (ProcessInstances.wfInstanceId is not null))  (cost=531.70 rows=5277) (actual time=0.023..1.852 rows=5277 l
oops=1)
                                -> Table scan on ProcessInstances  (cost=531.70 rows=5277) (actual time=0.023..1.354 rows=5277 loops=1)
                            -> Single-row index lookup on Processes using PRIMARY (processId=ProcessInstances.processId)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=5277)
                        -> Single-row index lookup on w using PRIMARY (wfInstanceId=ProcessInstances.wfInstanceId)  (cost=0.25 rows=1) (actual time=0.000..0.000 rows=1 loops=5277)
 |
```

Query 2

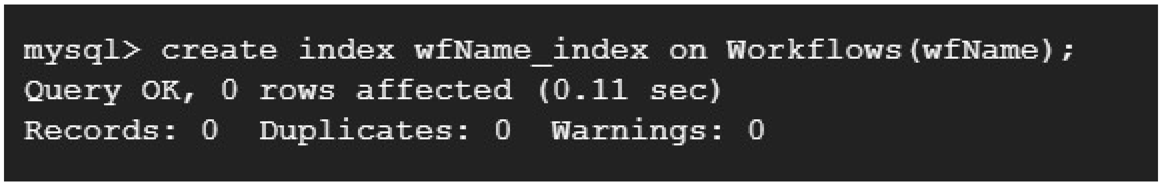
- Initial run of 'EXPLAIN ANALYZE' without creating an index.



```
mysql> explain analyze
    -> select count(wi.wfInstanceId) as completedWFInstances, wfName
    -> from Workflows w join WorkflowInstances wi using (wfId)
    -> where wi.completedDT != "0000-00-00 00:00:00"
    -> group by wfId;
+--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------+
| EXPLAIN

                                                     |
+--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------+
| -> Table scan on <temporary>  (actual time=0.001..0.001 rows=2 loops=1)
    -> Aggregate using temporary table  (actual time=1.050..1.051 rows=2 loops=1)
        -> Inner hash join (wi.wfId = w.wfId)  (cost=29.69 rows=16) (actual time=0.063..0.705 rows=784 loops=1)
            -> Filter: (wi.completedDT <> TIMESTAMP'0000-00-00 00:00:00')  (cost=10.12 rows=90) (actual time=0.025..0.380 rows=784 loops=1)
                -> Table scan on wi  (cost=10.12 rows=999) (actual time=0.023..0.275 rows=999 loops=1)
            -> Hash
                -> Table scan on w  (cost=0.45 rows=2) (actual time=0.024..0.027 rows=2 loops=1)
 |
+--------------------------------------------------------------------------------------------------------------------------------------------
```

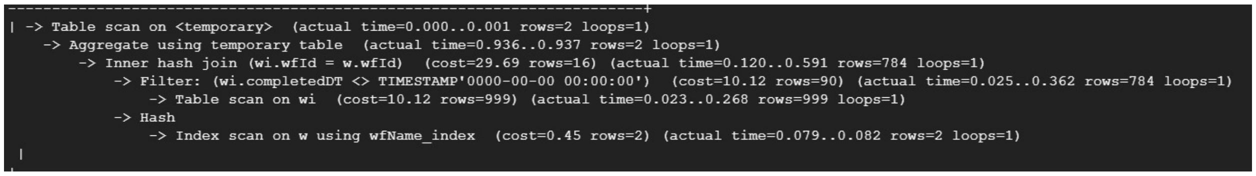
Index Design 1

- We added an index on the column ***wfName*** to check the performance of the query.



```
mysql> create index wfName_index on Workflows(wfName);
Query OK, 0 rows affected (0.11 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

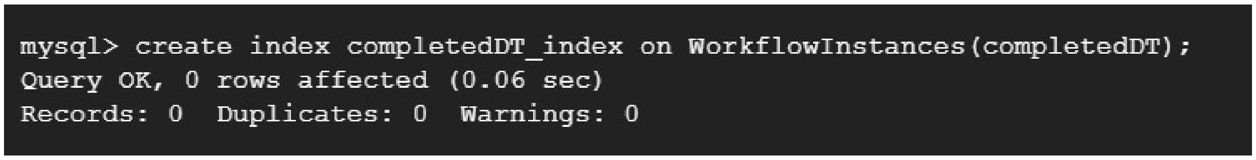
After running the explain analyze command on the same query, we realized that there was negligible improvement in its execution. Hence, we did not go ahead with this index.



```
| -> Table scan on <temporary>  (actual time=0.000..0.001 rows=2 loops=1)
    -> Aggregate using temporary table  (actual time=0.936..0.937 rows=2 loops=1)
        -> Inner hash join (wi.wfId = w.wfId)  (cost=29.69 rows=16) (actual time=0.120..0.591 rows=784 loops=1)
            -> Filter: (wi.completedDT <> TIMESTAMP'0000-00-00 00:00:00')  (cost=10.12 rows=90) (actual time=0.025..0.362 rows=784 loops=1)
                -> Table scan on wi  (cost=10.12 rows=999) (actual time=0.023..0.268 rows=999 loops=1)
            -> Hash
                -> Index scan on w using wfName_index  (cost=0.45 rows=2) (actual time=0.079..0.082 rows=2 loops=1)
 |
```

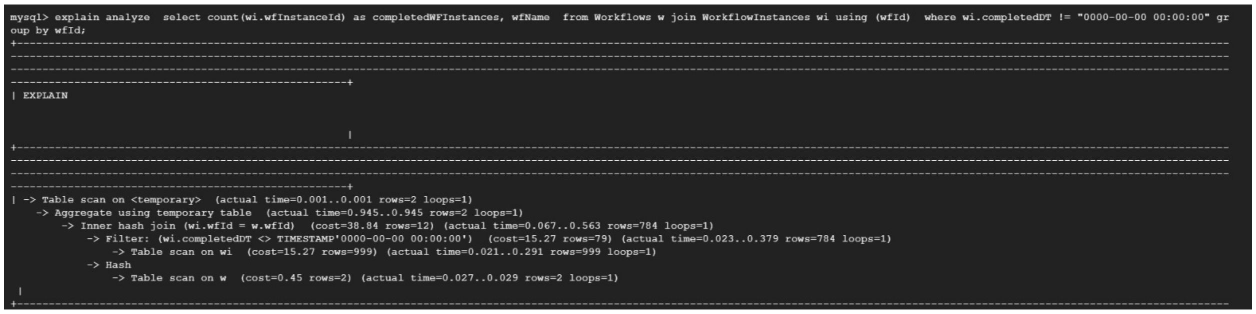
Index Design 2

- We added an index on *completedDT* to check the performance of the query.



```
mysql> create index completedDT_index on WorkflowInstances(completedDT);
Query OK, 0 rows affected (0.06 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

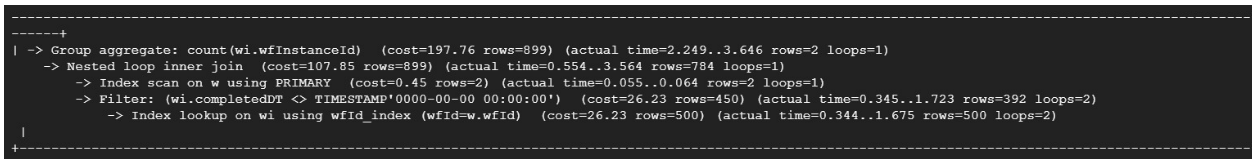
The performance of the query worsened after adding the index as reflected by the cost. Hence we did not go ahead with this index.



```
mysql> explain analyze  select count(wi.wfInstanceId) as completedWFInstances, wfName  from Workflows w join WorkflowInstances wi using (wfId)  where wi.completedDT != "0000-00-00 00:00:00" gr
oup by wfId;
+------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------+
| EXPLAIN


                                                    |
+------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------+
| -> Table scan on <temporary>  (actual time=0.001..0.001 rows=2 loops=1)
    -> Aggregate using temporary table  (actual time=0.945..0.945 rows=2 loops=1)
        -> Inner hash join (wi.wfId = w.wfId)  (cost=38.84 rows=12) (actual time=0.067..0.563 rows=784 loops=1)
            -> Filter: (wi.completedDT <> TIMESTAMP'0000-00-00 00:00:00')  (cost=15.27 rows=79) (actual time=0.023..0.379 rows=784 loops=1)
                -> Table scan on wi  (cost=15.27 rows=999) (actual time=0.021..0.291 rows=999 loops=1)
            -> Hash
                -> Table scan on w  (cost=0.45 rows=2) (actual time=0.027..0.029 rows=2 loops=1)
 |
+------------------------------------------------------------------------------------------------------------------------------------------------------------------------
```

Index Design 3

- We added an index *wfId on the table WorkflowInstances* to check the performance of the query. The query's execution performance decreased significantly after applying the index. Hence we did not go ahead with this design.



```
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------+
| -> Group aggregate: count(wi.wfInstanceId)  (cost=197.76 rows=899) (actual time=2.249..3.646 rows=2 loops=1)
    -> Nested loop inner join  (cost=107.85 rows=899) (actual time=0.554..3.564 rows=784 loops=1)
        -> Index scan on w using PRIMARY  (cost=0.45 rows=2) (actual time=0.055..0.064 rows=2 loops=1)
        -> Filter: (wi.completedDT <> TIMESTAMP'0000-00-00 00:00:00')  (cost=26.23 rows=450) (actual time=0.345..1.723 rows=392 loops=2)
            -> Index lookup on wi using wfId_index (wfId=w.wfId)  (cost=26.23 rows=500) (actual time=0.344..1.675 rows=500 loops=2)
 |
+-------------------------------------------------------------------------------------------------------------------------------------------------------------------------
```

Finally, for this query, we realized that the query couldn't be optimized any further. We applied indices to 3 different columns as recorded above but we couldn't find any significant improvement in the performance. We suspect that the reason behind this is that the columns on which we applied indices have multiple duplicate values and hence, the indices couldn't search faster. Also, the query's performance without any index is good. **Hence, we will proceed with the default index ie. the primary key as index.**