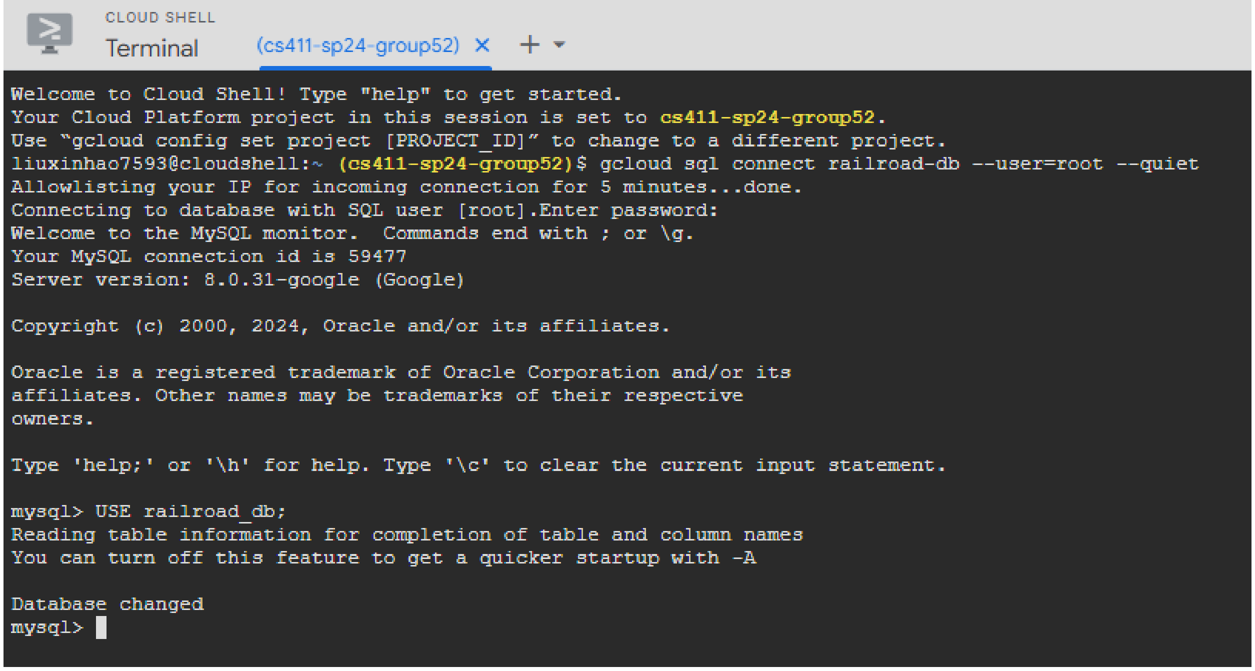
CS 411 Class Project Team 52 Stage 3
US Railway Safety and Operational Data Analysis Platform

Screenshot of connection:

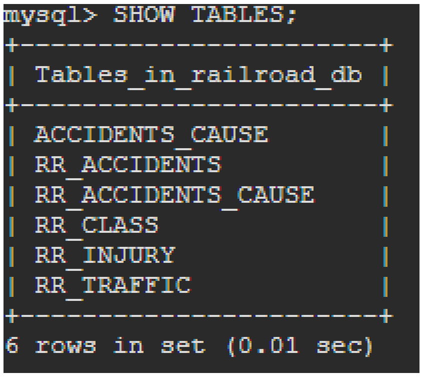


```
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to cs411-sp24-group52.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
liuxinhao7593@cloudshell:~ (cs411-sp24-group52)$ gcloud sql connect railroad-db --user=root --quiet
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 59477
Server version: 8.0.31-google (Google)

Copyright (c) 2000, 2024, 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.

mysql> USE railroad_db;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> 
```

Table DDL commands:

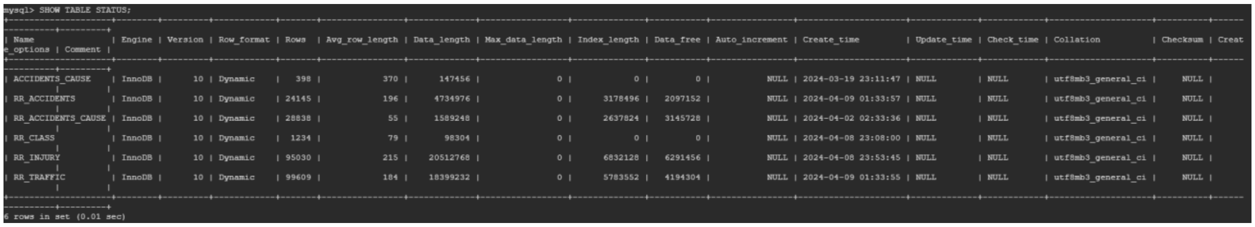
```
CREATE TABLE ACCIDENTS_CAUSE (
    FRA_CAUSE_CODE CHAR(4) PRIMARY KEY,
    ACCIDENT_CAUSE_CODE_DESCRIPTION VARCHAR(255),
    ADL_CAUSE_SUBGROUP VARCHAR(255),
    CAUSE_GROUP CHAR(3),
    FRA_ACCIDENT_CAUSE_GROUP VARCHAR(255),
    CATEGORY CHAR(1),
    TM_CM CHAR(2)
);

CREATE TABLE RR_INJURY (
    REPORT_CODE VARCHAR(255) PRIMARY KEY,
    DATE DATE,
    RAILROAD VARCHAR(255),
    TYPPERS CHAR(1),
    AGE INT,
    STATE INT,
    CASFATAL CHAR(1),
    COUNTY VARCHAR(255),
    HZMEXPOS CHAR(1),
    LATITUDE REAL,
    LONGITUD REAL,
    FOREIGN KEY (RAILROAD) REFERENCES RR_CLASS(RAILROAD)
    ON UPDATE CASCADE
    ON DELETE SET NULL
);

CREATE TABLE RR_TRAFFIC (
    TRAFFIC_CODE VARCHAR(255) PRIMARY KEY,
    RAILROAD VARCHAR(255),
    IYR INT,
    IMO INT,
    STATE INT,
    COUNTY VARCHAR(255),
    YSMI INT,
    FRTRNMI INT,
    PASTRNMI INT,
    OTHERMI INT,
    FOREIGN KEY (RAILROAD) REFERENCES RR_CLASS(RAILROAD)
    ON UPDATE CASCADE
    ON DELETE SET NULL
);

CREATE TABLE RR_ACCIDENTS (
    ACCIDNO VARCHAR(255) PRIMARY KEY,
    RAILROAD VARCHAR(255),
    DATE DATE,
    TRAIN_TYPE CHAR(1),
    ACC_TYPE INT,
    CARS INT,
    CARSDMG INT,
    CARSHZD INT,
    TOTAL_NUMBER_LOCO INT,
    TOTAL_LOCO_DERAIL INT,
    TOTAL_NUMBER_CAR INT,
    TOTAL_CONSIST INT,
    TOTAL_CAR_DERAIL INT,
    TOTAL_DERAIL INT,
    EVACUATE INT,
    STATION VARCHAR(255),
    STATE_CODE INT,
    TEMP INT,
    VISIBLTY INT,
    WEATHER INT,
    PRICAUSE VARCHAR(255),
    TRACK_TYPE CHAR(2),
    TRACK_CLASS CHAR(1),
    HIGHSPD INT,
    ACCDMG INT,
    LATITUDE REAL,
    LONGITUD REAL,
    STATE VARCHAR(2),
    TRAIN_WEIGHT INT,
    FOREIGN KEY (RAILROAD) REFERENCES RR_CLASS(RAILROAD)
    ON UPDATE CASCADE
    ON DELETE SET NULL,
    FOREIGN KEY (PRICAUSE) REFERENCES
ACCIDENTS_CAUSE(FRA_CAUSE_CODE)
    ON UPDATE CASCADE
    ON DELETE SET NULL
);

CREATE TABLE RR_CLASS (
    RAILROAD VARCHAR(255) PRIMARY KEY,
    RAILROAD_NAME VARCHAR(255),
    RAILROAD_SUCCESSOR VARCHAR(255),
    RRCLASSIFICATION INT
);


CREATE TABLE RR_ACCIDENTS_CAUSE (
    ACCIDNO VARCHAR(255),
    CAUSE VARCHAR(255),
    PRIMARY KEY (ACCIDNO, CAUSE),
    FOREIGN KEY (ACCIDNO) REFERENCES RR_ACCIDENTS(ACCIDNO)
    ON UPDATE CASCADE
    ON DELETE CASCADE,
    FOREIGN KEY (CAUSE) REFERENCES
ACCIDENTS_CAUSE(FRA_CAUSE_CODE)
    ON UPDATE CASCADE
    ON DELETE CASCADE);

CREATE TABLE RR_ACCIDENTS_CAUSE (
    ACCIDNO VARCHAR(255),
    CAUSE VARCHAR(255),
    PRIMARY KEY (ACCIDNO, CAUSE));
```

Table information:



```
mysql> SHOW TABLES;
+-----------------------+
| Tables_in_railroad_db |
+-----------------------+
| ACCIDENTS_CAUSE       |
| RR_ACCIDENTS          |
| RR_ACCIDENTS_CAUSE    |
| RR_CLASS              |
| RR_INJURY             |
| RR_TRAFFIC            |
+-----------------------+
6 rows in set (0.01 sec)
```

Table status:



```
mysql> SHOW TABLE STATUS;
+--------------------+--------+---------+------------+-------+----------------+-------------+-----------------+--------------+-----------+----------------+---------------------+-------------+------------+--------------------+----------+------
----------+---------+
| Name               | Engine | Version | Row_format | Rows  | Avg_row_length | Data_length | Max_data_length | Index_length | Data_free | Auto_increment | Create_time         | Update_time | Check_time | Collation          | Checksum | Creat
e_options | Comment |
+--------------------+--------+---------+------------+-------+----------------+-------------+-----------------+--------------+-----------+----------------+---------------------+-------------+------------+--------------------+----------+------
----------+---------+
| ACCIDENTS_CAUSE    | InnoDB |      10 | Dynamic    |   398 |            370 |      147456 |               0 |            0 |         0 |           NULL | 2024-03-19 23:11:47 | NULL        | NULL       | utf8mb3_general_ci |     NULL |
          |         |
| RR_ACCIDENTS       | InnoDB |      10 | Dynamic    | 24145 |            196 |     4734976 |               0 |      3178496 |   2097152 |           NULL | 2024-04-09 01:33:57 | NULL        | NULL       | utf8mb3_general_ci |     NULL |
          |         |
| RR_ACCIDENTS_CAUSE | InnoDB |      10 | Dynamic    | 28838 |             55 |     1589248 |               0 |      2637824 |   3145728 |           NULL | 2024-04-02 02:33:36 | NULL        | NULL       | utf8mb3_general_ci |     NULL |
          |         |
| RR_CLASS           | InnoDB |      10 | Dynamic    |  1234 |             79 |       98304 |               0 |            0 |         0 |           NULL | 2024-04-08 23:08:00 | NULL        | NULL       | utf8mb3_general_ci |     NULL |
          |         |
| RR_INJURY          | InnoDB |      10 | Dynamic    | 95030 |            215 |    20512768 |               0 |      6832128 |   6291456 |           NULL | 2024-04-08 23:53:45 | NULL        | NULL       | utf8mb3_general_ci |     NULL |
          |         |
| RR_TRAFFIC         | InnoDB |      10 | Dynamic    | 99609 |            184 |    18399232 |               0 |      5783552 |   4194304 |           NULL | 2024-04-09 01:33:55 | NULL        | NULL       | utf8mb3_general_ci |     NULL |
          |         |
+--------------------+--------+---------+------------+-------+----------------+-------------+-----------------+--------------+-----------+----------------+---------------------+-------------+------------+--------------------+----------+------
----------+---------+
6 rows in set (0.01 sec)
```

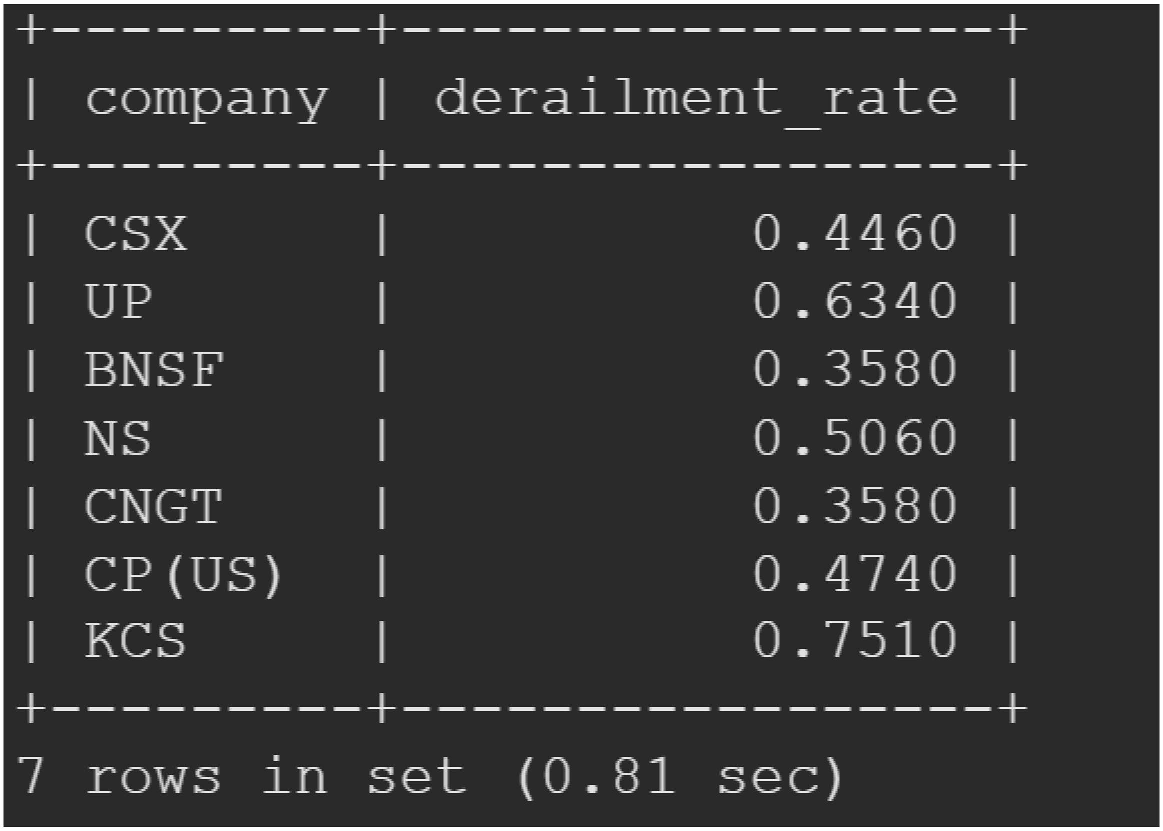
Number of rows in each table:
ACCIDENTS_CAUSE: 398
RR_ACCIDENTS: 24145
RR_ACCIDENTS_CAUSE: 28838
RR_CLASS: 1234
RR_INJURY: 95030
RR_TRAFFIC: 99609

Five Advanced Query:

1: Given a time period (2013-2022), for freight train derailment accidents, happened on mainline and siding only, for class 1 railroad accidents only, I want to see the mainline and siding derailment rate by the railroad company

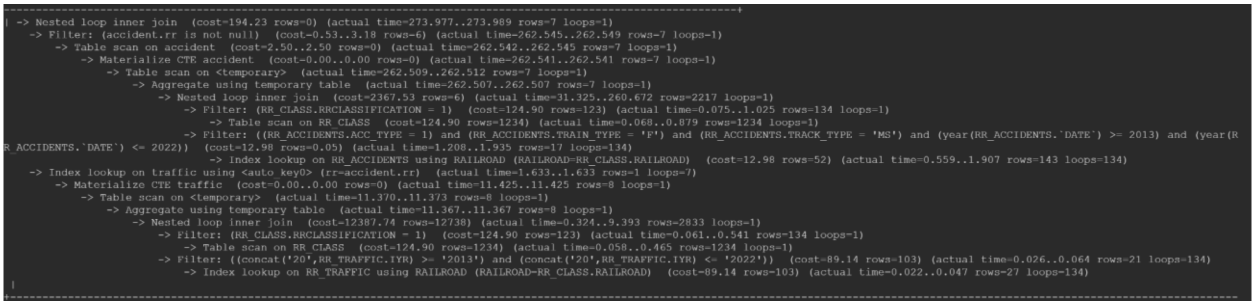
Code:

```
WITH
accident AS (
    SELECT RR_CLASS.RAILROAD_SUCCESSOR AS rr, COUNT(*) AS
derailment
    FROM RR_ACCIDENTS
    JOIN RR_CLASS ON RR_ACCIDENTS.RAILROAD = RR_CLASS.RAILROAD
    WHERE RR_CLASS.RRCLASSIFICATION = 1
    AND RR_ACCIDENTS.ACC_TYPE = 1
    AND RR_ACCIDENTS.TRAIN_TYPE = "F"
    AND RR_ACCIDENTS.TRACK_TYPE = "MS"
    AND YEAR(RR_ACCIDENTS.`DATE`) >= 2013
    AND YEAR(RR_ACCIDENTS.`DATE`) <= 2022
    GROUP BY RR_CLASS.RAILROAD_SUCCESSOR
),
traffic AS (
    SELECT RR_CLASS.RAILROAD_SUCCESSOR AS rr,
SUM(RR_TRAFFIC.FRTRNMI + RR_TRAFFIC.OTHERMI) AS mile
    FROM RR_TRAFFIC
    JOIN RR_CLASS ON RR_TRAFFIC.RAILROAD = RR_CLASS.RAILROAD
    WHERE RR_CLASS.RRCLASSIFICATION = 1
    AND CONCAT('20', RR_TRAFFIC.IYR) >= '2013'
    AND CONCAT('20', RR_TRAFFIC.IYR) <= '2022'
    GROUP BY RR_CLASS.RAILROAD_SUCCESSOR
)
SELECT accident.rr AS company, (accident.derailment /
traffic.mile)*1000000 AS derailment_rate
FROM accident
JOIN traffic ON accident.rr = traffic.rr;
```



```
+---------+------------------+
| company | derailment_rate  |
+---------+------------------+
| CSX     |           0.4460 |
| UP      |           0.6340 |
| BNSF    |           0.3580 |
| NS      |           0.5060 |
| CNGT    |           0.3580 |
| CP(US)  |           0.4740 |
| KCS     |           0.7510 |
+---------+------------------+
7 rows in set (0.81 sec)
```

Explain analyze without indexing:

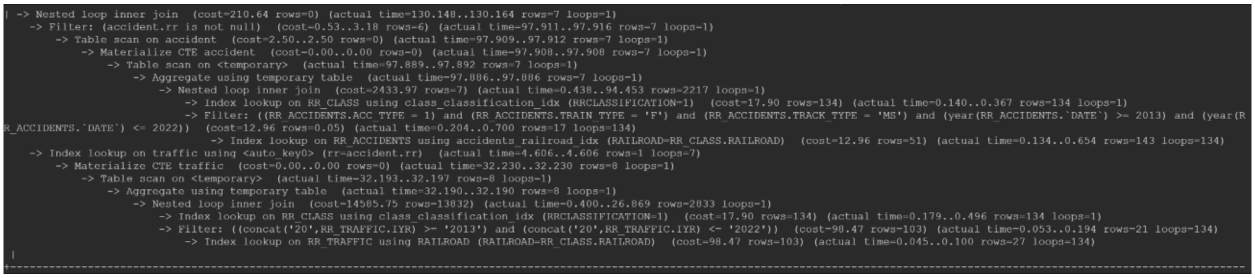


```
-------------------------------------------------------------------------------------------------------------------------------+
| -> Nested loop inner join  (cost=194.23 rows=0) (actual time=273.977..273.989 rows=7 loops=1)
    -> Filter: (accident.rr is not null)  (cost=0.53..3.18 rows=6) (actual time=262.545..262.549 rows=7 loops=1)
        -> Table scan on accident  (cost=2.50..2.50 rows=0) (actual time=262.542..262.545 rows=7 loops=1)
            -> Materialize CTE accident  (cost=0.00..0.00 rows=0) (actual time=262.541..262.541 rows=7 loops=1)
                -> Table scan on <temporary>  (actual time=262.509..262.512 rows=7 loops=1)
                    -> Aggregate using temporary table  (actual time=262.507..262.507 rows=7 loops=1)
                        -> Nested loop inner join  (cost=2367.53 rows=6) (actual time=31.325..260.672 rows=2217 loops=1)
                            -> Filter: (RR_CLASS.RRCLASSIFICATION = 1)  (cost=124.90 rows=123) (actual time=0.075..1.025 rows=134 loops=1)
                                -> Table scan on RR_CLASS  (cost=124.90 rows=1234) (actual time=0.068..0.879 rows=1234 loops=1)
                            -> Filter: ((RR_ACCIDENTS.ACC_TYPE = 1) and (RR_ACCIDENTS.TRAIN_TYPE = 'F') and (RR_ACCIDENTS.TRACK_TYPE = 'MS') and (year(RR_ACCIDENTS.`DATE`) >= 2013) and (year(R
R_ACCIDENTS.`DATE`) <= 2022))  (cost=12.98 rows=0.05) (actual time=1.208..1.935 rows=17 loops=134)
                                -> Index lookup on RR_ACCIDENTS using RAILROAD (RAILROAD=RR_CLASS.RAILROAD)  (cost=12.98 rows=52) (actual time=0.559..1.907 rows=143 loops=134)
    -> Index lookup on traffic using <auto_key0> (rr=accident.rr)  (actual time=1.633..1.633 rows=1 loops=7)
        -> Materialize CTE traffic  (cost=0.00..0.00 rows=0) (actual time=11.425..11.425 rows=8 loops=1)
            -> Table scan on <temporary>  (actual time=11.370..11.373 rows=8 loops=1)
                -> Aggregate using temporary table  (actual time=11.367..11.367 rows=8 loops=1)
                    -> Nested loop inner join  (cost=12387.74 rows=12738) (actual time=0.324..9.393 rows=2833 loops=1)
                        -> Filter: (RR_CLASS.RRCLASSIFICATION = 1)  (cost=124.90 rows=123) (actual time=0.061..0.541 rows=134 loops=1)
                            -> Table scan on RR_CLASS  (cost=124.90 rows=1234) (actual time=0.058..0.465 rows=1234 loops=1)
                        -> Filter: ((concat('20',RR_TRAFFIC.IYR) >= '2013') and (concat('20',RR_TRAFFIC.IYR) <= '2022'))  (cost=89.14 rows=103) (actual time=0.026..0.064 rows=21 loops=134)
                            -> Index lookup on RR_TRAFFIC using RAILROAD (RAILROAD=RR_CLASS.RAILROAD)  (cost=89.14 rows=103) (actual time=0.022..0.047 rows=27 loops=134)
 |
+-------------------------------------------------------------------------------------------------------------------------------
```

INDEX 1:

INDEX on RR_CLASS.RRCLASSIFICATION, RR_CLASS.RAILROAD_SUCCESSOR (place index on some non-primary key columns from join, where, and group by)

```
CREATE INDEX class_classification_idx ON
RR_CLASS(RRCLASSIFICATION);
CREATE INDEX class_successor_idx ON
RR_CLASS(RAILROAD_SUCCESSOR);
```



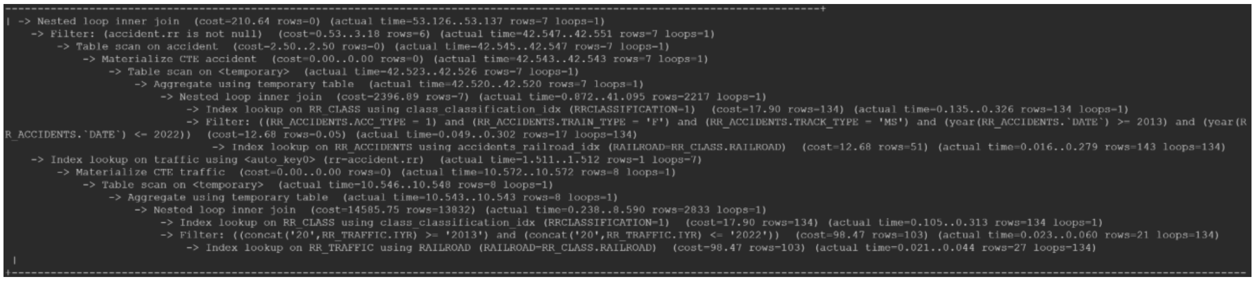
```
| -> Nested loop inner join  (cost=210.64 rows=0) (actual time=130.148..130.164 rows=7 loops=1)
    -> Filter: (accident.rr is not null)  (cost=0.53..3.18 rows=6) (actual time=97.911..97.916 rows=7 loops=1)
        -> Table scan on accident  (cost=2.50..2.50 rows=0) (actual time=97.909..97.912 rows=7 loops=1)
            -> Materialize CTE accident  (cost=0.00..0.00 rows=0) (actual time=97.908..97.908 rows=7 loops=1)
                -> Table scan on <temporary>  (actual time=97.889..97.892 rows=7 loops=1)
                    -> Aggregate using temporary table  (actual time=97.886..97.886 rows=7 loops=1)
                        -> Nested loop inner join  (cost=2433.97 rows=7) (actual time=0.438..94.453 rows=2217 loops=1)
                            -> Index lookup on RR_CLASS using class_classification_idx (RRCLASSIFICATION=1)  (cost=17.90 rows=134) (actual time=0.140..0.367 rows=134 loops=1)
                            -> Filter: ((RR_ACCIDENTS.ACC_TYPE = 1) and (RR_ACCIDENTS.TRAIN_TYPE = 'F') and (RR_ACCIDENTS.TRACK_TYPE = 'MS') and (year(RR_ACCIDENTS.`DATE`) >= 2013) and (year(R
R_ACCIDENTS.`DATE`) <= 2022))  (cost=12.96 rows=0.05) (actual time=0.204..0.700 rows=17 loops=134)
                                -> Index lookup on RR_ACCIDENTS using accidents_railroad_idx (RAILROAD=RR_CLASS.RAILROAD)  (cost=12.96 rows=51) (actual time=0.134..0.654 rows=143 loops=134)
    -> Index lookup on traffic using <auto_key0> (rr=accident.rr)  (actual time=4.606..4.606 rows=1 loops=7)
        -> Materialize CTE traffic  (cost=0.00..0.00 rows=0) (actual time=32.230..32.230 rows=8 loops=1)
            -> Table scan on <temporary>  (actual time=32.193..32.197 rows=8 loops=1)
                -> Aggregate using temporary table  (actual time=32.190..32.190 rows=8 loops=1)
                    -> Nested loop inner join  (cost=14585.75 rows=13832) (actual time=0.400..26.869 rows=2833 loops=1)
                        -> Index lookup on RR_CLASS using class_classification_idx (RRCLASSIFICATION=1)  (cost=17.90 rows=134) (actual time=0.179..0.496 rows=134 loops=1)
                        -> Filter: ((concat('20',RR_TRAFFIC.IYR) >= '2013') and (concat('20',RR_TRAFFIC.IYR) <= '2022'))  (cost=98.47 rows=103) (actual time=0.053..0.194 rows=21 loops=134)
                            -> Index lookup on RR_TRAFFIC using RAILROAD (RAILROAD=RR_CLASS.RAILROAD)  (cost=98.47 rows=103) (actual time=0.045..0.100 rows=27 loops=134)
 |
+----------------------------------------------------------------------------------------------------------------------------------------------------------------
```

As we can see, the overall cost is increased from 194 to 210. However, the specific cost related to RRCLASSIFICATION is reduced from 124 to 17.9. The cost related to RAILROAD_SUCCESSOR is not showing up in the analysis.

INDEX 2:

INDEX on RR_CLASS.RRCLASSIFICATION and RR_ACCIDENTS.DATE (place index on non-primary key columns from join and where)

CREATE INDEX class_classification_idx ON RR_CLASS(RRCLASSIFICATION);
CREATE INDEX accidents_date_idx ON RR_ACCIDENTS(DATE);



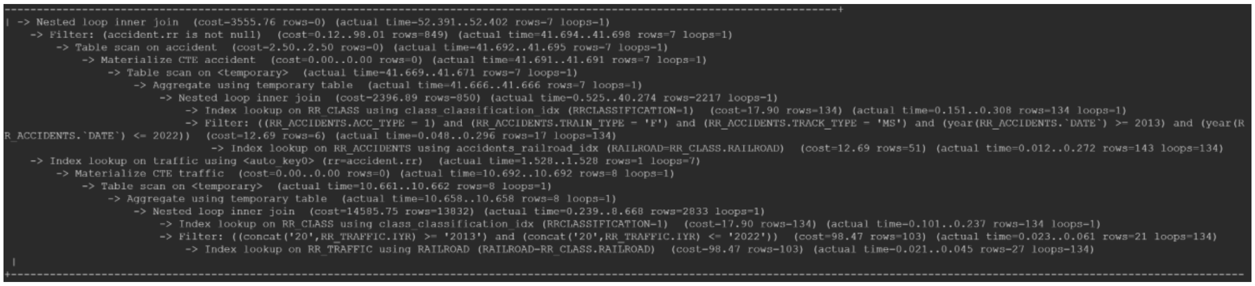
```
---------------------------------------------------------------------------------------------------------------+
| -> Nested loop inner join  (cost=210.64 rows=0) (actual time=53.126..53.137 rows=7 loops=1)
    -> Filter: (accident.rr is not null)  (cost=0.53..3.18 rows=6) (actual time=42.547..42.551 rows=7 loops=1)
        -> Table scan on accident  (cost=2.50..2.50 rows=0) (actual time=42.545..42.547 rows=7 loops=1)
            -> Materialize CTE accident  (cost=0.00..0.00 rows=0) (actual time=42.543..42.543 rows=7 loops=1)
                -> Table scan on <temporary>  (actual time=42.523..42.526 rows=7 loops=1)
                    -> Aggregate using temporary table  (actual time=42.520..42.520 rows=7 loops=1)
                        -> Nested loop inner join  (cost=2396.89 rows=7) (actual time=0.872..41.095 rows=2217 loops=1)
                            -> Index lookup on RR_CLASS using class_classification_idx (RRCLASSIFICATION=1)  (cost=17.90 rows=134) (actual time=0.135..0.326 rows=134 loops=1)
                            -> Filter: ((RR_ACCIDENTS.ACC_TYPE = 1) and (RR_ACCIDENTS.TRAIN_TYPE = 'F') and (RR_ACCIDENTS.TRACK_TYPE = 'MS') and (year(RR_ACCIDENTS.`DATE`) >= 2013) and (year(R
R_ACCIDENTS.`DATE`) <= 2022))  (cost=12.68 rows=0.05) (actual time=0.049..0.302 rows=17 loops=134)
                                -> Index lookup on RR_ACCIDENTS using accidents_railroad_idx (RAILROAD=RR_CLASS.RAILROAD)  (cost=12.68 rows=51) (actual time=0.016..0.279 rows=143 loops=134)
    -> Index lookup on traffic using <auto_key0> (rr=accident.rr)  (actual time=1.511..1.512 rows=1 loops=7)
        -> Materialize CTE traffic  (cost=0.00..0.00 rows=0) (actual time=10.572..10.572 rows=8 loops=1)
            -> Table scan on <temporary>  (actual time=10.546..10.548 rows=8 loops=1)
                -> Aggregate using temporary table  (actual time=10.543..10.543 rows=8 loops=1)
                    -> Nested loop inner join  (cost=14585.75 rows=13832) (actual time=0.238..8.590 rows=2833 loops=1)
                        -> Index lookup on RR_CLASS using class_classification_idx (RRCLASSIFICATION=1)  (cost=17.90 rows=134) (actual time=0.105..0.313 rows=134 loops=1)
                        -> Filter: ((concat('20',RR_TRAFFIC.IYR) >= '2013') and (concat('20',RR_TRAFFIC.IYR) <= '2022'))  (cost=98.47 rows=103) (actual time=0.023..0.060 rows=21 loops=134)
                            -> Index lookup on RR_TRAFFIC using RAILROAD (RAILROAD=RR_CLASS.RAILROAD)  (cost=98.47 rows=103) (actual time=0.021..0.044 rows=27 loops=134)
 |
+----------------------------------------------------------------------------------------------------------------------------------------------------------------
```

The cost is the same as INDEX 1.
INDEX 3:

INDEX on RR_CLASS.RRCLASSIFICATION, RR_ACCIDENTS.DATE, RR_ACCIDENTS.ACC_TYPE, RR_ACCIDENTS.TRAIN_TYPE, and RR_ACCIDENTS.DATE (place index on all non-primary key columns from where and join)

CREATE INDEX class_classification_idx ON RR_CLASS(RRCLASSIFICATION);
CREATE INDEX accidents_acctype_idx ON RR_ACCIDENTS(ACC_TYPE);

```
CREATE INDEX accidents_train_type_idx ON
RR_ACCIDENTS(TRAIN_TYPE);
CREATE INDEX accidents_track_type_idx ON
RR_ACCIDENTS(TRACK_TYPE);
CREATE INDEX accidents_date_idx ON RR_ACCIDENTS(DATE);

ALTER TABLE RR_ACCIDENTS DROP INDEX accidents_railroad_idx;
ALTER TABLE RR_CLASS DROP INDEX class_classification_idx;
ALTER TABLE RR_ACCIDENTS DROP INDEX accidents_acctype_idx;
ALTER TABLE RR_ACCIDENTS DROP INDEX accidents_train_type_idx;
ALTER TABLE RR_ACCIDENTS DROP INDEX accidents_track_type_idx;
ALTER TABLE RR_ACCIDENTS DROP INDEX accidents_date_idx;
ALTER TABLE RR_CLASS DROP INDEX class_successor_idx;
```



```
---------------------------------------------------------------------------------------------------------------------+
| -> Nested loop inner join  (cost=3555.76 rows=0) (actual time=52.391..52.402 rows=7 loops=1)
    -> Filter: (accident.rr is not null)  (cost=0.12..98.01 rows=849) (actual time=41.694..41.698 rows=7 loops=1)
        -> Table scan on accident  (cost=2.50..2.50 rows=0) (actual time=41.692..41.695 rows=7 loops=1)
            -> Materialize CTE accident  (cost=0.00..0.00 rows=0) (actual time=41.691..41.691 rows=7 loops=1)
                -> Table scan on <temporary>  (actual time=41.669..41.671 rows=7 loops=1)
                    -> Aggregate using temporary table  (actual time=41.666..41.666 rows=7 loops=1)
                        -> Nested loop inner join  (cost=2396.89 rows=850) (actual time=0.525..40.274 rows=2217 loops=1)
                            -> Index lookup on RR_CLASS using class_classification_idx (RRCLASSIFICATION=1)  (cost=17.90 rows=134) (actual time=0.151..0.308 rows=134 loops=1)
                            -> Filter: ((RR_ACCIDENTS.ACC_TYPE = 1) and (RR_ACCIDENTS.TRAIN_TYPE = 'F') and (RR_ACCIDENTS.TRACK_TYPE = 'MS') and (year(RR_ACCIDENTS.`DATE`) >= 2013) and (year(R
R_ACCIDENTS.`DATE`) <= 2022))  (cost=12.69 rows=6) (actual time=0.048..0.296 rows=17 loops=134)
                                -> Index lookup on RR_ACCIDENTS using accidents_railroad_idx (RAILROAD=RR_CLASS.RAILROAD)  (cost=12.69 rows=51) (actual time=0.012..0.272 rows=143 loops=134)
    -> Index lookup on traffic using <auto_key0> (rr=accident.rr)  (actual time=1.528..1.528 rows=1 loops=7)
        -> Materialize CTE traffic  (cost=0.00..0.00 rows=0) (actual time=10.692..10.692 rows=8 loops=1)
            -> Table scan on <temporary>  (actual time=10.661..10.662 rows=8 loops=1)
                -> Aggregate using temporary table  (actual time=10.658..10.658 rows=8 loops=1)
                    -> Nested loop inner join  (cost=14585.75 rows=13832) (actual time=0.239..8.668 rows=2833 loops=1)
                        -> Index lookup on RR_CLASS using class_classification_idx (RRCLASSIFICATION=1)  (cost=17.90 rows=134) (actual time=0.101..0.237 rows=134 loops=1)
                        -> Filter: ((concat('20',RR_TRAFFIC.IYR) >= '2013') and (concat('20',RR_TRAFFIC.IYR) <= '2022'))  (cost=98.47 rows=103) (actual time=0.023..0.061 rows=21 loops=134)
                            -> Index lookup on RR_TRAFFIC using RAILROAD (RAILROAD=RR_CLASS.RAILROAD)  (cost=98.47 rows=103) (actual time=0.021..0.045 rows=27 loops=134)
 |
+--------------------------------------------------------------------------------------------------------------------------------------------------
```

The overall cost increases drastically from 194 to 3555. And the cost of operations on the WHERE clause does not change much after adding the index on all of them.

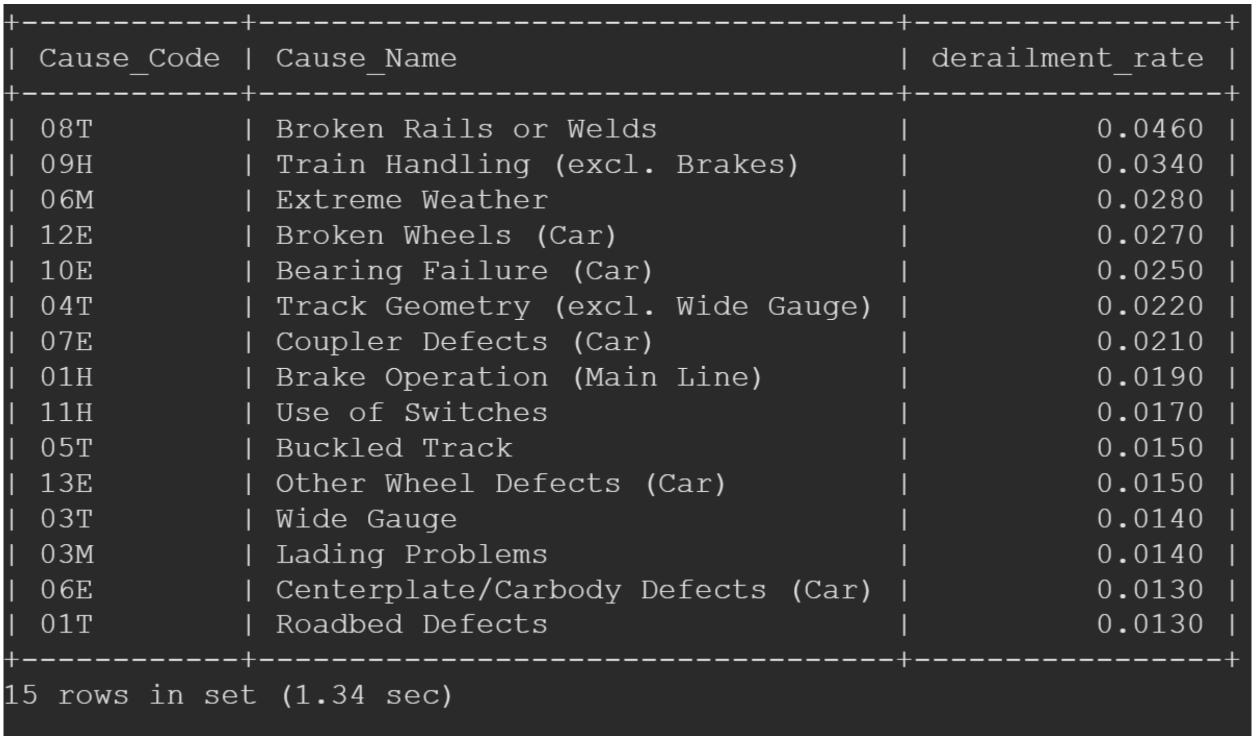
Thus, for query 1, the best indexing approach is not using any additional index. The reason is probably the columns that applied to index do not have too many unique values.

2:

Given a time period (2013-2022), for freight train derailment accidents, happened on mainline and siding only, for class 1 railroad accidents only, I want to see the mainline and siding derailment rate by accident primary causes

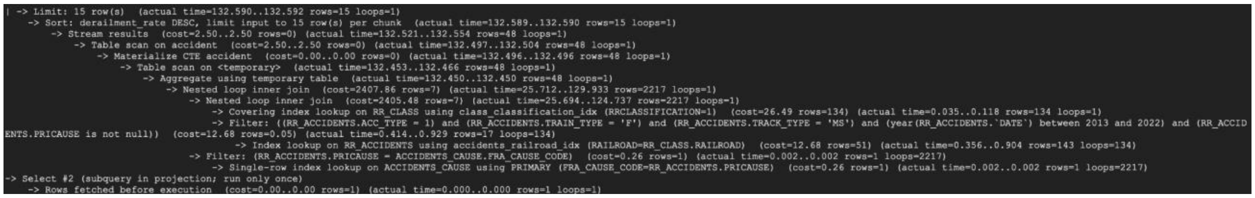
Code:

```
WITH
accident AS (
    SELECT ACCIDENTS_CAUSE.CAUSE_GROUP AS Cause_Code,
ACCIDENTS_CAUSE.ADL_CAUSE_SUBGROUP AS Cause_Name, COUNT(*) AS
derailment
    FROM RR_ACCIDENTS
    JOIN RR_CLASS ON RR_ACCIDENTS.RAILROAD = RR_CLASS.RAILROAD
    JOIN ACCIDENTS_CAUSE ON RR_ACCIDENTS.PRICAUSE =
ACCIDENTS_CAUSE.FRA_CAUSE_CODE
    WHERE RR_CLASS.RRCLASSIFICATION = 1
    AND RR_ACCIDENTS.ACC_TYPE = 1
    AND RR_ACCIDENTS.TRAIN_TYPE = "F"
    AND RR_ACCIDENTS.TRACK_TYPE = "MS"
    AND YEAR(RR_ACCIDENTS.`DATE`) BETWEEN 2013 AND 2022
    GROUP BY ACCIDENTS_CAUSE.CAUSE_GROUP,
ACCIDENTS_CAUSE.ADL_CAUSE_SUBGROUP
),
traffic AS (
    SELECT SUM(RR_TRAFFIC.FRTRNMI + RR_TRAFFIC.OTHERMI) AS mile
    FROM RR_TRAFFIC
    JOIN RR_CLASS ON RR_TRAFFIC.RAILROAD = RR_CLASS.RAILROAD
    WHERE RR_CLASS.RRCLASSIFICATION = 1
    AND CONCAT('20', RR_TRAFFIC.IYR) >= '2013'
    AND CONCAT('20', RR_TRAFFIC.IYR) <= '2022'
)
SELECT accident.Cause_Code AS Cause_Code, accident.Cause_Name AS
Cause_Name, (accident.derailment / (SELECT mile FROM
traffic))*1000000 AS derailment_rate
FROM accident
ORDER BY derailment_rate DESC LIMIT 15;
```



```
+------------+------------------------------------+------------------+
| Cause_Code | Cause_Name                         | derailment_rate  |
+------------+------------------------------------+------------------+
| 08T        | Broken Rails or Welds              |           0.0460 |
| 09H        | Train Handling (excl. Brakes)      |           0.0340 |
| 06M        | Extreme Weather                    |           0.0280 |
| 12E        | Broken Wheels (Car)                |           0.0270 |
| 10E        | Bearing Failure (Car)              |           0.0250 |
| 04T        | Track Geometry (excl. Wide Gauge)  |           0.0220 |
| 07E        | Coupler Defects (Car)              |           0.0210 |
| 01H        | Brake Operation (Main Line)        |           0.0190 |
| 11H        | Use of Switches                    |           0.0170 |
| 05T        | Buckled Track                      |           0.0150 |
| 13E        | Other Wheel Defects (Car)          |           0.0150 |
| 03T        | Wide Gauge                         |           0.0140 |
| 03M        | Lading Problems                    |           0.0140 |
| 06E        | Centerplate/Carbody Defects (Car)  |           0.0130 |
| 01T        | Roadbed Defects                    |           0.0130 |
+------------+------------------------------------+------------------+
15 rows in set (1.34 sec)
```

Explain analyze without indexing:



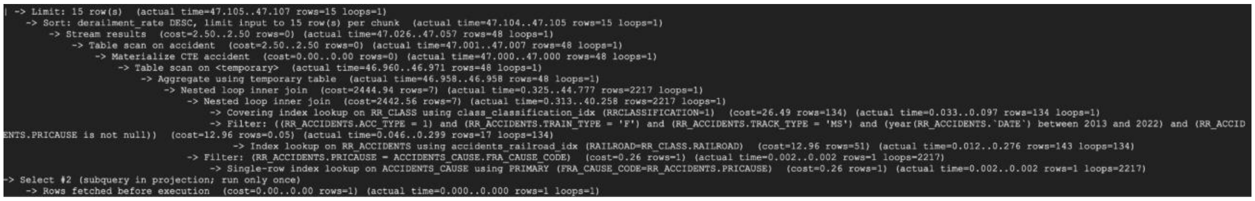
```
| -> Limit: 15 row(s)  (actual time=132.590..132.592 rows=15 loops=1)
    -> Sort: derailment_rate DESC, limit input to 15 row(s) per chunk  (actual time=132.589..132.590 rows=15 loops=1)
        -> Stream results  (cost=2.50..2.50 rows=0) (actual time=132.521..132.554 rows=48 loops=1)
            -> Table scan on accident  (cost=2.50..2.50 rows=0) (actual time=132.497..132.504 rows=48 loops=1)
                -> Materialize CTE accident  (cost=0.00..0.00 rows=0) (actual time=132.496..132.496 rows=48 loops=1)
                    -> Table scan on <temporary>  (actual time=132.453..132.466 rows=48 loops=1)
                        -> Aggregate using temporary table  (actual time=132.450..132.450 rows=48 loops=1)
                            -> Nested loop inner join  (cost=2407.86 rows=7) (actual time=25.712..129.933 rows=2217 loops=1)
                                -> Nested loop inner join  (cost=2405.48 rows=7) (actual time=25.694..124.737 rows=2217 loops=1)
                                    -> Covering index lookup on RR_CLASS using class_classification_idx (RRCLASSIFICATION=1)  (cost=26.49 rows=134) (actual time=0.035..0.118 rows=134 loops=1)
                                    -> Filter: ((RR_ACCIDENTS.ACC_TYPE = 1) and (RR_ACCIDENTS.TRAIN_TYPE = 'F') and (RR_ACCIDENTS.TRACK_TYPE = 'MS') and (year(RR_ACCIDENTS.`DATE`) between 2013 and 2022) and (RR_ACCID
ENTS.PRICAUSE is not null))  (cost=12.68 rows=0.05) (actual time=0.414..0.929 rows=17 loops=134)
                                        -> Index lookup on RR_ACCIDENTS using accidents_railroad_idx (RAILROAD=RR_CLASS.RAILROAD)  (cost=12.68 rows=51) (actual time=0.356..0.904 rows=143 loops=134)
                                -> Filter: (RR_ACCIDENTS.PRICAUSE = ACCIDENTS_CAUSE.FRA_CAUSE_CODE)  (cost=0.26 rows=1) (actual time=0.002..0.002 rows=1 loops=2217)
                                    -> Single-row index lookup on ACCIDENTS_CAUSE using PRIMARY (FRA_CAUSE_CODE=RR_ACCIDENTS.PRICAUSE)  (cost=0.26 rows=1) (actual time=0.002..0.002 rows=1 loops=2217)
-> Select #2 (subquery in projection; run only once)
    -> Rows fetched before execution  (cost=0.00..0.00 rows=1) (actual time=0.000..0.000 rows=1 loops=1)
```

The costs of two Nested loop joins are 2407.86 and 2405.48.

INDEX 1:

INDEX on RR_CLASS.RRCLASSIFICATION

```
CREATE INDEX class_classification_idx ON
RR_CLASS(RRCLASSIFICATION);

ALTER TABLE RR_CLASS DROP INDEX class_classification_idx;
```

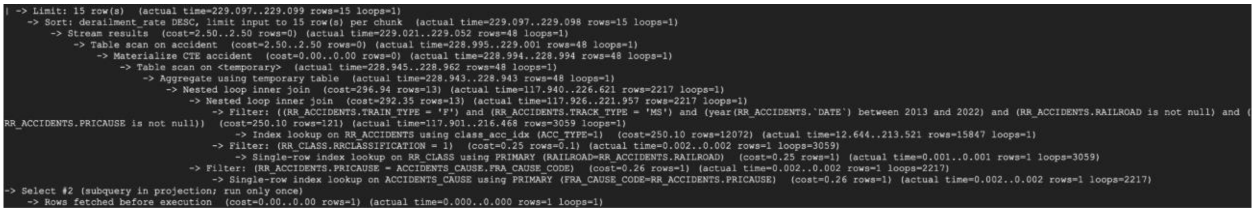


```
| -> Limit: 15 row(s)  (actual time=47.105..47.107 rows=15 loops=1)
    -> Sort: derailment_rate DESC, limit input to 15 row(s) per chunk  (actual time=47.104..47.105 rows=15 loops=1)
        -> Stream results  (cost=2.50..2.50 rows=0) (actual time=47.026..47.057 rows=48 loops=1)
            -> Table scan on accident  (cost=2.50..2.50 rows=0) (actual time=47.001..47.007 rows=48 loops=1)
                -> Materialize CTE accident  (cost=0.00..0.00 rows=0) (actual time=47.000..47.000 rows=48 loops=1)
                    -> Table scan on <temporary>  (actual time=46.960..46.971 rows=48 loops=1)
                        -> Aggregate using temporary table  (actual time=46.958..46.958 rows=48 loops=1)
                            -> Nested loop inner join  (cost=2444.94 rows=7) (actual time=0.325..44.777 rows=2217 loops=1)
                                -> Nested loop inner join  (cost=2442.56 rows=7) (actual time=0.313..40.258 rows=2217 loops=1)
                                    -> Covering index lookup on RR_CLASS using class_classification_idx (RRCLASSIFICATION=1)  (cost=26.49 rows=134) (actual time=0.033..0.097 rows=134 loops=1)
                                    -> Filter: ((RR_ACCIDENTS.ACC_TYPE = 1) and (RR_ACCIDENTS.TRAIN_TYPE = 'F') and (RR_ACCIDENTS.TRACK_TYPE = 'MS') and (year(RR_ACCIDENTS.`DATE`) between 2013 and 2022) and (RR_ACCID
ENTS.PRICAUSE is not null))  (cost=12.96 rows=0.05) (actual time=0.046..0.299 rows=17 loops=134)
                                        -> Index lookup on RR_ACCIDENTS using accidents_railroad_idx (RAILROAD=RR_CLASS.RAILROAD)  (cost=12.96 rows=51) (actual time=0.012..0.276 rows=143 loops=134)
                                -> Filter: (RR_ACCIDENTS.PRICAUSE = ACCIDENTS_CAUSE.FRA_CAUSE_CODE)  (cost=0.26 rows=1) (actual time=0.002..0.002 rows=1 loops=2217)
                                    -> Single-row index lookup on ACCIDENTS_CAUSE using PRIMARY (FRA_CAUSE_CODE=RR_ACCIDENTS.PRICAUSE)  (cost=0.26 rows=1) (actual time=0.002..0.002 rows=1 loops=2217)
-> Select #2 (subquery in projection; run only once)
    -> Rows fetched before execution  (cost=0.00..0.00 rows=1) (actual time=0.000..0.000 rows=1 loops=1)
```

The costs of two Nested loop joins are 2444.94 and 2442.56.

INDEX 2:
INDEX on RR_CLASS.RRCLASSIFICATION, RR_ACCIDENTS.ACC_TYPE, RR_TRAFFIC.IYR

```
CREATE INDEX class_classification_idx ON
RR_CLASS(RRCLASSIFICATION);
CREATE INDEX class_acc_idx ON RR_ACCIDENTS(ACC_TYPE);
CREATE INDEX class_year_idx ON RR_TRAFFIC(IYR);

ALTER TABLE RR_CLASS DROP INDEX class_classification_idx;
ALTER TABLE RR_ACCIDENTS DROP INDEX class_acc_idx;
ALTER TABLE RR_TRAFFIC DROP INDEX class_year_idx;
```

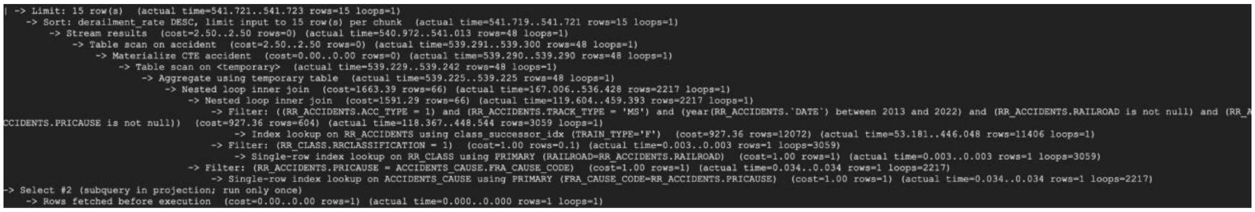


```
| -> Limit: 15 row(s)  (actual time=229.097..229.099 rows=15 loops=1)
    -> Sort: derailment_rate DESC, limit input to 15 row(s) per chunk  (actual time=229.097..229.098 rows=15 loops=1)
        -> Stream results  (cost=2.50..2.50 rows=0) (actual time=229.021..229.052 rows=48 loops=1)
            -> Table scan on accident  (cost=2.50..2.50 rows=0) (actual time=228.995..229.001 rows=48 loops=1)
                -> Materialize CTE accident  (cost=0.00..0.00 rows=0) (actual time=228.994..228.994 rows=48 loops=1)
                    -> Table scan on <temporary>  (actual time=228.945..228.962 rows=48 loops=1)
                        -> Aggregate using temporary table  (actual time=228.943..228.943 rows=48 loops=1)
                            -> Nested loop inner join  (cost=296.94 rows=13) (actual time=117.940..226.621 rows=2217 loops=1)
                                -> Nested loop inner join  (cost=292.35 rows=13) (actual time=117.926..221.957 rows=2217 loops=1)
                                    -> Filter: ((RR_ACCIDENTS.TRAIN_TYPE = 'F') and (RR_ACCIDENTS.TRACK_TYPE = 'MS') and (year(RR_ACCIDENTS.`DATE`) between 2013 and 2022) and (RR_ACCIDENTS.RAILROAD is not null) and (
RR_ACCIDENTS.PRICAUSE is not null))  (cost=250.10 rows=121) (actual time=117.901..216.468 rows=3059 loops=1)
                                        -> Index lookup on RR_ACCIDENTS using class_acc_idx (ACC_TYPE=1)  (cost=250.10 rows=12072) (actual time=12.644..213.521 rows=15847 loops=1)
                                    -> Filter: (RR_CLASS.RRCLASSIFICATION = 1)  (cost=0.25 rows=0.1) (actual time=0.002..0.002 rows=1 loops=3059)
                                        -> Single-row index lookup on RR_CLASS using PRIMARY (RAILROAD=RR_ACCIDENTS.RAILROAD)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=3059)
                                -> Filter: (RR_ACCIDENTS.PRICAUSE = ACCIDENTS_CAUSE.FRA_CAUSE_CODE)  (cost=0.26 rows=1) (actual time=0.002..0.002 rows=1 loops=2217)
                                    -> Single-row index lookup on ACCIDENTS_CAUSE using PRIMARY (FRA_CAUSE_CODE=RR_ACCIDENTS.PRICAUSE)  (cost=0.26 rows=1) (actual time=0.002..0.002 rows=1 loops=2217)
-> Select #2 (subquery in projection; run only once)
    -> Rows fetched before execution  (cost=0.00..0.00 rows=1) (actual time=0.000..0.000 rows=1 loops=1)
```

The costs of two Nested loop joins are 296.94 and 292.35.

INDEX 3:
INDEX on RR_CLASS.RRCLASSIFICATION, RR_ACCIDENTS.ACC_TYPE, RR_ACCIDENTS.TRAIN_TYPE, RR_ACCIDENTS.`DATE`, RR_TRAFFIC.IYR

```
CREATE INDEX class_classification_idx ON
RR_CLASS(RRCLASSIFICATION);
CREATE INDEX class_acc_idx ON RR_ACCIDENTS(ACC_TYPE);
CREATE INDEX class_train_idx ON RR_ACCIDENTS(TRAIN_TYPE);
CREATE INDEX class_date_idx ON RR_ACCIDENTS(`DATE`);
CREATE INDEX class_year_idx ON RR_TRAFFIC(IYR);

ALTER TABLE RR_CLASS DROP INDEX class_classification_idx;
ALTER TABLE RR_ACCIDENTS DROP INDEX class_acc_idx;
ALTER TABLE RR_ACCIDENTS DROP INDEX class_train_idx;
ALTER TABLE RR_ACCIDENTS DROP INDEX class_date_idx;
ALTER TABLE RR_TRAFFIC DROP INDEX class_year_idx;
```



```
| -> Limit: 15 row(s)  (actual time=541.721..541.723 rows=15 loops=1)
    -> Sort: derailment_rate DESC, limit input to 15 row(s) per chunk  (actual time=541.719..541.721 rows=15 loops=1)
        -> Stream results  (cost=2.50..2.50 rows=0) (actual time=540.972..541.013 rows=48 loops=1)
            -> Table scan on accident  (cost=2.50..2.50 rows=0) (actual time=539.291..539.300 rows=48 loops=1)
                -> Materialize CTE accident  (cost=0.00..0.00 rows=0) (actual time=539.290..539.290 rows=48 loops=1)
                    -> Table scan on <temporary>  (actual time=539.229..539.242 rows=48 loops=1)
                        -> Aggregate using temporary table  (actual time=539.225..539.225 rows=48 loops=1)
                            -> Nested loop inner join  (cost=1663.39 rows=66) (actual time=167.006..536.428 rows=2217 loops=1)
                                -> Nested loop inner join  (cost=1591.29 rows=66) (actual time=119.604..459.393 rows=2217 loops=1)
                                    -> Filter: ((RR_ACCIDENTS.ACC_TYPE = 1) and (RR_ACCIDENTS.TRACK_TYPE = 'MS') and (year(RR_ACCIDENTS.`DATE`) between 2013 and 2022) and (RR_ACCIDENTS.RAILROAD is not null) and (RR_A
CCIDENTS.PRICAUSE is not null))  (cost=927.36 rows=604) (actual time=118.367..448.544 rows=3059 loops=1)
                                        -> Index lookup on RR_ACCIDENTS using class_successor_idx (TRAIN_TYPE='F')  (cost=927.36 rows=12072) (actual time=53.181..446.048 rows=11406 loops=1)
                                    -> Filter: (RR_CLASS.RRCLASSIFICATION = 1)  (cost=1.00 rows=0.1) (actual time=0.003..0.003 rows=1 loops=3059)
                                        -> Single-row index lookup on RR_CLASS using PRIMARY (RAILROAD=RR_ACCIDENTS.RAILROAD)  (cost=1.00 rows=1) (actual time=0.003..0.003 rows=1 loops=3059)
                                -> Filter: (RR_ACCIDENTS.PRICAUSE = ACCIDENTS_CAUSE.FRA_CAUSE_CODE)  (cost=1.00 rows=1) (actual time=0.034..0.034 rows=1 loops=2217)
                                    -> Single-row index lookup on ACCIDENTS_CAUSE using PRIMARY (FRA_CAUSE_CODE=RR_ACCIDENTS.PRICAUSE)  (cost=1.00 rows=1) (actual time=0.034..0.034 rows=1 loops=2217)
-> Select #2 (subquery in projection; run only once)
    -> Rows fetched before execution  (cost=0.00..0.00 rows=1) (actual time=0.000..0.000 rows=1 loops=1)
```

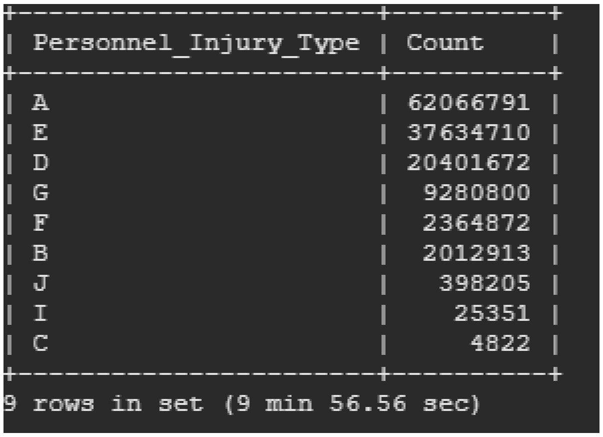
The costs of two Nested loop joins are 1663.39 and 1591.29.

Report:
The final index design: RR_CLASS.RRCLASSIFICATION, RR_ACCIDENTS.ACC_TYPE, RR_TRAFFIC.IYR
In the index 1, we only added 1 index (RR_CLASS.RRCLASSIFICATION). The cost of it (2444.94 and 2442.56) is almost the same as the cost of the query without using any index (2407.86 and 2405.48).
In the index 3, we added indexes on every attribute used in where (RR_CLASS.RRCLASSIFICATION, RR_ACCIDENTS.ACC_TYPE, RR_ACCIDENTS.TRAIN_TYPE, RR_ACCIDENTS.`DATE`, RR_TRAFFIC.IYR), which has a slightly better performance (1663.39 and 1591.29).
In the index 2, we chose three attributes to add indexes (RR_CLASS.RRCLASSIFICATION, RR_ACCIDENTS.ACC_TYPE, RR_TRAFFIC.IYR), which performs much better (296.94 and 292.35) than the other three designs.
Therefore, the final index design of this query is to add indexes on RR_CLASS.RRCLASSIFICATION, RR_ACCIDENTS.ACC_TYPE, RR_TRAFFIC.IYR.

3: Given a time period (2013-2022), for class 1 railroad accidents only, I want to see the rank of the type of personnel injured on railroad property

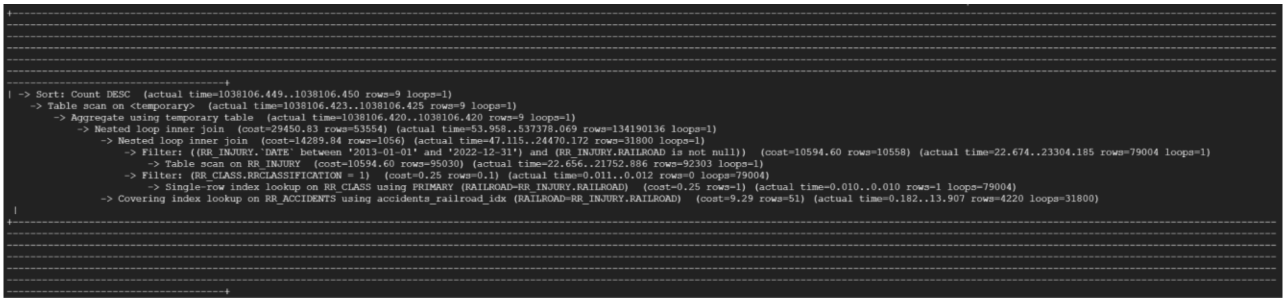
Code:

```
SELECT
    inj.TYPPERS AS Personnel_Injury_Type,
    COUNT(*) AS Count
FROM
    (SELECT RAILROAD, TYPPERS
     FROM RR_INJURY
     WHERE DATE BETWEEN '2013-01-01' AND '2022-12-31'
    ) AS inj
JOIN
    (SELECT RAILROAD
     FROM RR_CLASS
     WHERE RRCLASSIFICATION = 1
    ) AS class
ON
    inj.RAILROAD = class.RAILROAD
JOIN
    (SELECT RAILROAD
     FROM RR_ACCIDENTS
    ) AS acc
ON
    inj.RAILROAD = acc.RAILROAD
GROUP BY
    inj.TYPPERS
ORDER BY
    Count DESC;
```



```
+----------------------+----------+
| Personnel_Injury_Type | Count    |
+----------------------+----------+
| A                    | 62066791 |
| E                    | 37634710 |
| D                    | 20401672 |
| G                    |  9280800 |
| F                    |  2364872 |
| B                    |  2012913 |
| J                    |   398205 |
| I                    |    25351 |
| C                    |     4822 |
+----------------------+----------+
9 rows in set (9 min 56.56 sec)
```

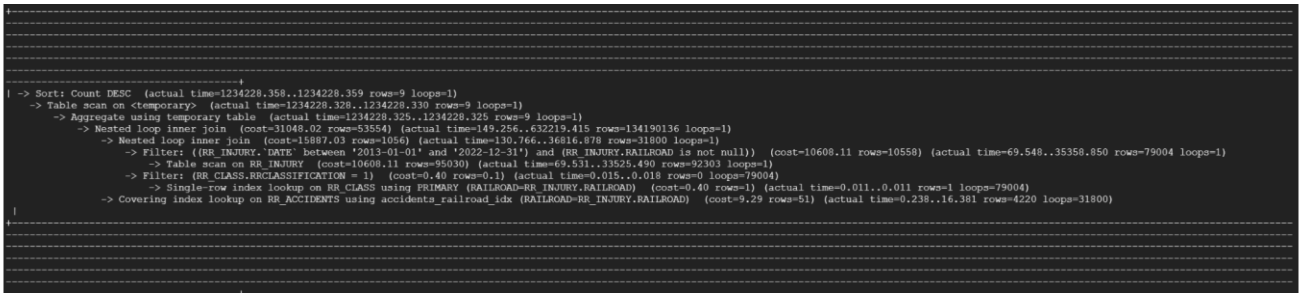
Cost with no index:



```
| -> Sort: Count DESC  (actual time=1038106.449..1038106.450 rows=9 loops=1)
    -> Table scan on <temporary>  (actual time=1038106.423..1038106.425 rows=9 loops=1)
        -> Aggregate using temporary table  (actual time=1038106.420..1038106.420 rows=9 loops=1)
            -> Nested loop inner join  (cost=29450.83 rows=53554) (actual time=53.958..537378.069 rows=134190136 loops=1)
                -> Nested loop inner join  (cost=14289.84 rows=1056) (actual time=47.115..24470.172 rows=31800 loops=1)
                    -> Filter: ((RR_INJURY.`DATE` between '2013-01-01' and '2022-12-31') and (RR_INJURY.RAILROAD is not null))  (cost=10594.60 rows=10558) (actual time=22.674..23304.185 rows=79004 loops=1)
                        -> Table scan on RR_INJURY  (cost=10594.60 rows=95030) (actual time=22.656..21752.886 rows=92303 loops=1)
                    -> Filter: (RR_CLASS.RRCLASSIFICATION = 1)  (cost=0.25 rows=0.1) (actual time=0.011..0.012 rows=0 loops=79004)
                        -> Single-row index lookup on RR_CLASS using PRIMARY (RAILROAD=RR_INJURY.RAILROAD)  (cost=0.25 rows=1) (actual time=0.010..0.010 rows=1 loops=79004)
                -> Covering index lookup on RR_ACCIDENTS using accidents_railroad_idx (RAILROAD=RR_INJURY.RAILROAD)  (cost=9.29 rows=51) (actual time=0.182..13.907 rows=4220 loops=31800)
 |
```

INDEX 1: INDEX ON RR_INJURY.RAILROAD, RR_ACCIDENTS.RAILROAD

```
CREATE INDEX inj_rr_idx ON RR_INJURY(RAILROAD);
CREATE INDEX acc_rr_idx ON RR_ACCIDENTS(RAILROAD);

ALTER TABLE RR_INJURY DROP INDEX inj_rr_idx;
ALTER TABLE RR_ACCIDENTS DROP INDEX acc_rr_idx;
```

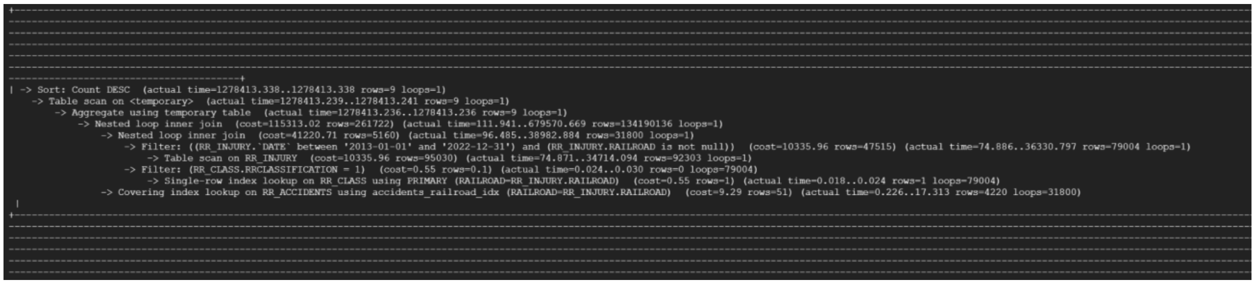


```
| -> Sort: Count DESC  (actual time=1234228.358..1234228.359 rows=9 loops=1)
    -> Table scan on <temporary>  (actual time=1234228.328..1234228.330 rows=9 loops=1)
        -> Aggregate using temporary table  (actual time=1234228.325..1234228.325 rows=9 loops=1)
            -> Nested loop inner join  (cost=31048.02 rows=53554) (actual time=149.256..632219.415 rows=134190136 loops=1)
                -> Nested loop inner join  (cost=15887.03 rows=1056) (actual time=130.766..36816.878 rows=31800 loops=1)
                    -> Filter: ((RR_INJURY.`DATE` between '2013-01-01' and '2022-12-31') and (RR_INJURY.RAILROAD is not null))  (cost=10608.11 rows=10558) (actual time=69.548..35358.850 rows=79004 loops=1)
                        -> Table scan on RR_INJURY  (cost=10608.11 rows=95030) (actual time=69.531..33525.490 rows=92303 loops=1)
                    -> Filter: (RR_CLASS.RRCLASSIFICATION = 1)  (cost=0.40 rows=0.1) (actual time=0.015..0.018 rows=0 loops=79004)
                        -> Single-row index lookup on RR_CLASS using PRIMARY (RAILROAD=RR_INJURY.RAILROAD)  (cost=0.40 rows=1) (actual time=0.011..0.011 rows=1 loops=79004)
                -> Covering index lookup on RR_ACCIDENTS using accidents_railroad_idx (RAILROAD=RR_INJURY.RAILROAD)  (cost=9.29 rows=51) (actual time=0.238..16.381 rows=4220 loops=31800)
 |
```

INDEX 2: INDEX ON RR_INJURY.DATE, RR_CLASS.RRCLASSIFICATION

```
CREATE INDEX inj_date_idx ON RR_INJURY(DATE);
CREATE INDEX class_cls_idx ON RR_CLASS(RRCLASSIFICATION);
CREATE INDEX inj_type_idx ON RR_INJURY(TYPPERS);
```

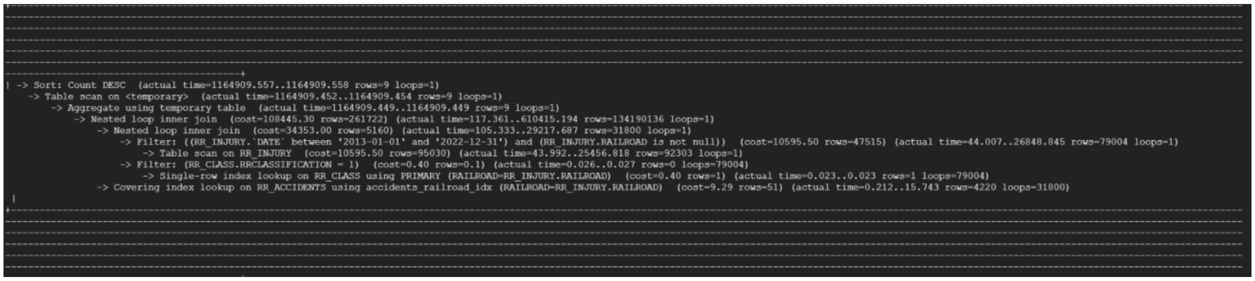
```
ALTER TABLE RR_INJURY DROP INDEX inj_date_idx;
ALTER TABLE RR_ACCIDENTS DROP INDEX class_cls_idx;
ALTER TABLE RR_INJURY DROP INDEX inj_type_idx ;
```



```
| -> Sort: Count DESC  (actual time=1278413.338..1278413.338 rows=9 loops=1)
    -> Table scan on <temporary>  (actual time=1278413.239..1278413.241 rows=9 loops=1)
        -> Aggregate using temporary table  (actual time=1278413.236..1278413.236 rows=9 loops=1)
            -> Nested loop inner join  (cost=115313.02 rows=261722) (actual time=111.941..679570.669 rows=134190136 loops=1)
                -> Nested loop inner join  (cost=41220.71 rows=5160) (actual time=96.485..38982.884 rows=31800 loops=1)
                    -> Filter: ((RR_INJURY.`DATE` between '2013-01-01' and '2022-12-31') and (RR_INJURY.RAILROAD is not null))  (cost=10335.96 rows=47515) (actual time=74.886..36330.797 rows=79004 loops=1)
                        -> Table scan on RR_INJURY  (cost=10335.96 rows=95030) (actual time=74.871..34714.094 rows=92303 loops=1)
                    -> Filter: (RR_CLASS.RRCLASSIFICATION = 1)  (cost=0.55 rows=0.1) (actual time=0.024..0.030 rows=0 loops=79004)
                        -> Single-row index lookup on RR_CLASS using PRIMARY (RAILROAD=RR_INJURY.RAILROAD)  (cost=0.55 rows=1) (actual time=0.018..0.024 rows=1 loops=79004)
                -> Covering index lookup on RR_ACCIDENTS using accidents_railroad_idx (RAILROAD=RR_INJURY.RAILROAD)  (cost=9.29 rows=51) (actual time=0.226..17.313 rows=4220 loops=31800)
 |
```

INDEX 3: INDEX ON RR_INJURY.RAILROAD, RR_INJURY.DATE, RR_ACCIDENTS.RAILROAD, RR_CLASS.RRCLASSIFICATION

```
CREATE INDEX inj_rr_idx ON RR_INJURY(RAILROAD);
CREATE INDEX acc_rr_idx ON RR_ACCIDENTS(RAILROAD);
CREATE INDEX inj_date_idx ON RR_INJURY(DATE);
CREATE INDEX class_cls_idx ON RR_CLASS(RRCLASSIFICATION);
CREATE INDEX inj_type_idx ON RR_INJURY(TYPPERS);

ALTER TABLE RR_INJURY DROP INDEX inj_rr_idx;
ALTER TABLE RR_ACCIDENTS DROP INDEX acc_rr_idx;
ALTER TABLE RR_INJURY DROP INDEX inj_date_idx;
ALTER TABLE RR_ACCIDENTS DROP INDEX class_cls_idx;
ALTER TABLE RR_INJURY DROP INDEX inj_type_idx ;
```



```
| -> Sort: Count DESC  (actual time=1164909.557..1164909.558 rows=9 loops=1)
    -> Table scan on <temporary>  (actual time=1164909.452..1164909.454 rows=9 loops=1)
        -> Aggregate using temporary table  (actual time=1164909.449..1164909.449 rows=9 loops=1)
            -> Nested loop inner join  (cost=108445.30 rows=261722) (actual time=117.361..610415.194 rows=134190136 loops=1)
                -> Nested loop inner join  (cost=34353.00 rows=5160) (actual time=105.333..29217.687 rows=31800 loops=1)
                    -> Filter: ((RR_INJURY.`DATE` between '2013-01-01' and '2022-12-31') and (RR_INJURY.RAILROAD is not null))  (cost=10595.50 rows=47515) (actual time=44.007..26848.845 rows=79004 loops=1)
                        -> Table scan on RR_INJURY  (cost=10595.50 rows=95030) (actual time=43.992..25456.818 rows=92303 loops=1)
                    -> Filter: (RR_CLASS.RRCLASSIFICATION = 1)  (cost=0.40 rows=0.1) (actual time=0.026..0.027 rows=0 loops=79004)
                        -> Single-row index lookup on RR_CLASS using PRIMARY (RAILROAD=RR_INJURY.RAILROAD)  (cost=0.40 rows=1) (actual time=0.023..0.023 rows=1 loops=79004)
                -> Covering index lookup on RR_ACCIDENTS using accidents_railroad_idx (RAILROAD=RR_INJURY.RAILROAD)  (cost=9.29 rows=51) (actual time=0.212..15.743 rows=4220 loops=31800)
 |
```

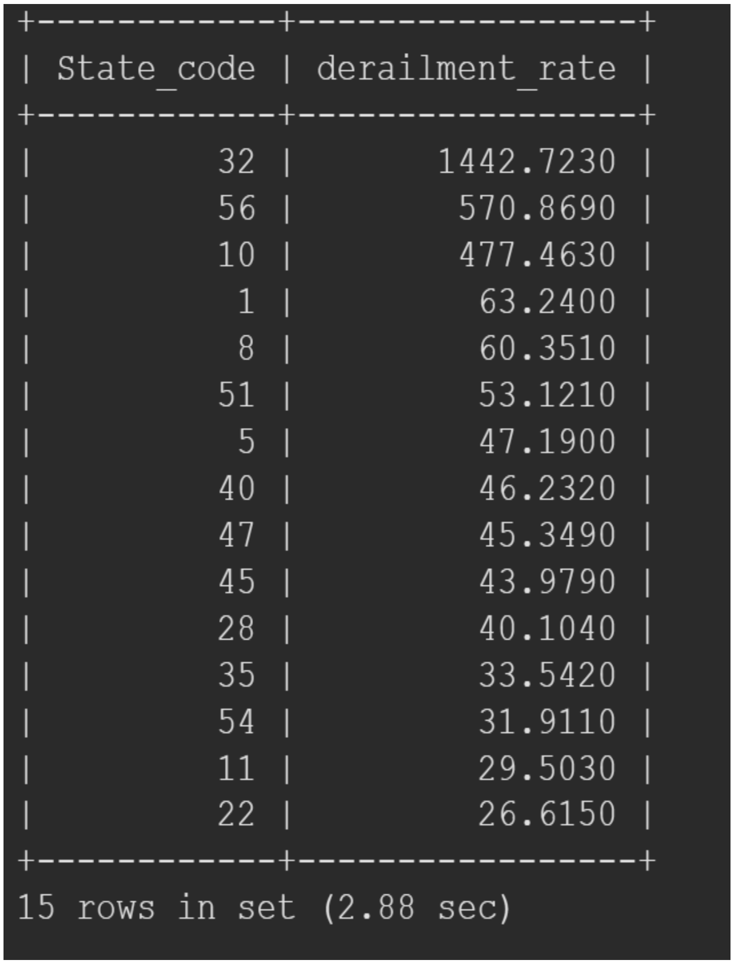
Report:
From the EXPLAIN ANALYZE results of three different indexing above, there is no better effect compared to that without index. We would not use indexing for this query because the cost for nested loop inner join(29450.83) is the lowest in the query

without index. INDEX 1 has slightly greater cost, while the other two indexes have much greater cost. This might be because there are not enough unique values in the attribute we added index.

4: Given a time period (2013-2022), for freight train derailment accidents, happened on mainline and siding only, for class 1 railroad accidents only, I want to see the mainline and siding derailment rate by state

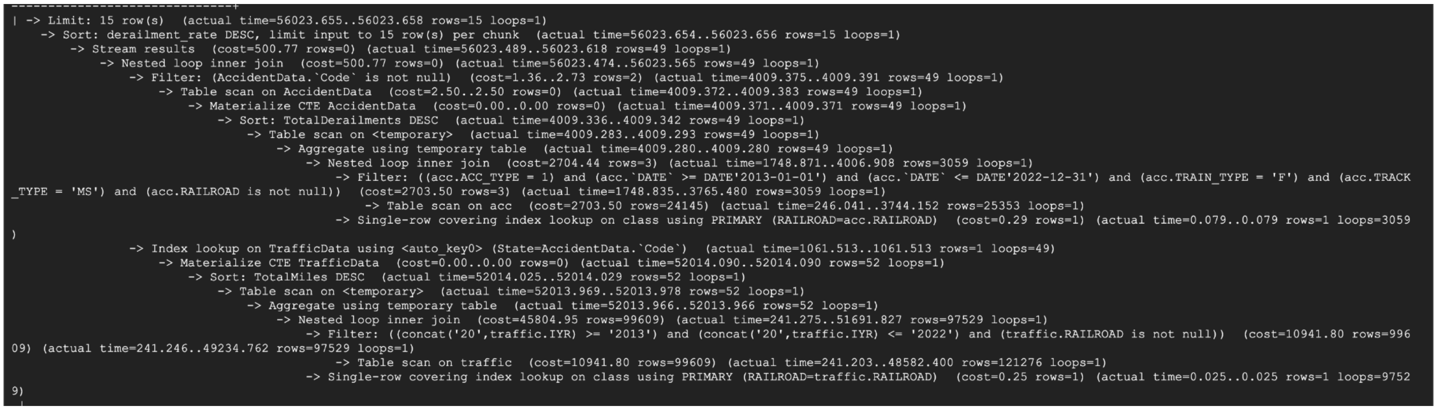
Code:

```
WITH
AccidentData AS (
    SELECT
        acc.STATE_CODE AS Code,
        COUNT(*) AS TotalDerailments
    FROM
        RR_ACCIDENTS acc
    JOIN
        RR_CLASS class ON acc.RAILROAD = class.RAILROAD
    WHERE
        acc.DATE >= '2013-01-01' AND
        acc.DATE <= '2022-12-31' AND
        acc.ACC_TYPE = 1 AND
        acc.TRAIN_TYPE = 'F' AND
        acc.TRACK_TYPE = 'MS'
    GROUP BY
        acc.STATE_CODE
    ORDER BY TotalDerailments DESC
),
TrafficData AS (
    SELECT
        traffic.STATE AS State,
        SUM(traffic.FRTRNMI + traffic.OTHERMI) AS TotalMiles
    FROM
        RR_TRAFFIC traffic
    JOIN
        RR_CLASS class ON traffic.RAILROAD = class.RAILROAD
    WHERE
        CONCAT('20', traffic.IYR) >= '2013'
        AND CONCAT('20', traffic.IYR) <= '2022'
    GROUP BY
        traffic.STATE
    ORDER BY TotalMiles DESC
)
SELECT
    AccidentData.Code AS State_code,
    IFNULL(((AccidentData.TotalDerailments /
TrafficData.TotalMiles)*1000000),0) AS derailment_rate
FROM
    AccidentData
JOIN
    TrafficData ON AccidentData.Code = TrafficData.State
ORDER BY derailment_rate DESC
LIMIT 15;
```



```
+------------+-----------------+
| State_code | derailment_rate |
+------------+-----------------+
|         32 |       1442.7230 |
|         56 |        570.8690 |
|         10 |        477.4630 |
|          1 |         63.2400 |
|          8 |         60.3510 |
|         51 |         53.1210 |
|          5 |         47.1900 |
|         40 |         46.2320 |
|         47 |         45.3490 |
|         45 |         43.9790 |
|         28 |         40.1040 |
|         35 |         33.5420 |
|         54 |         31.9110 |
|         11 |         29.5030 |
|         22 |         26.6150 |
+------------+-----------------+
15 rows in set (2.88 sec)
```

Explain analyze without indexing:

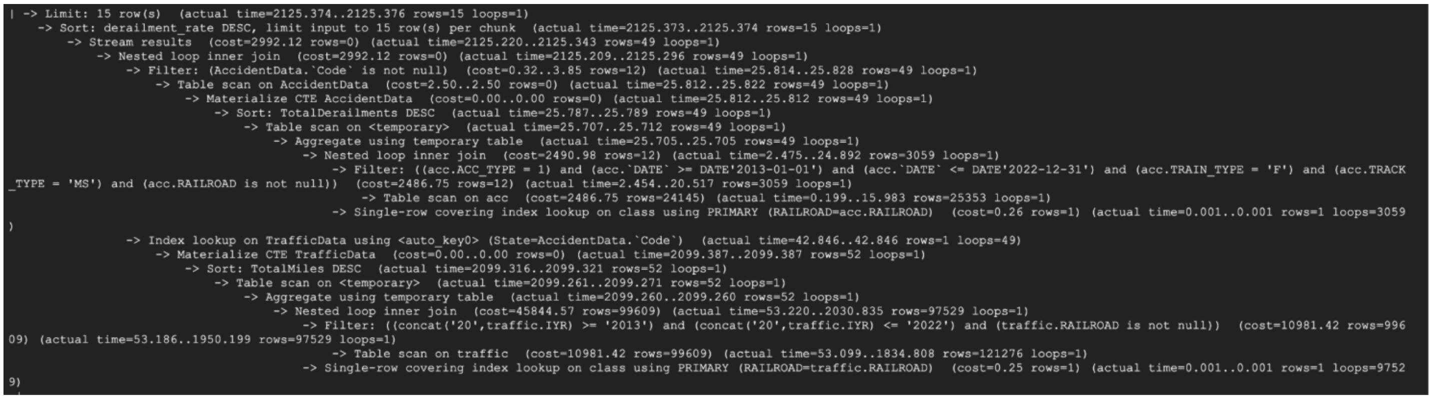


```
-----------------------------+
| -> Limit: 15 row(s)  (actual time=56023.655..56023.658 rows=15 loops=1)
    -> Sort: derailment_rate DESC, limit input to 15 row(s) per chunk  (actual time=56023.654..56023.656 rows=15 loops=1)
        -> Stream results  (cost=500.77 rows=0) (actual time=56023.489..56023.618 rows=49 loops=1)
            -> Nested loop inner join  (cost=500.77 rows=0) (actual time=56023.474..56023.565 rows=49 loops=1)
                -> Filter: (AccidentData.`Code` is not null)  (cost=1.36..2.73 rows=2) (actual time=4009.375..4009.391 rows=49 loops=1)
                    -> Table scan on AccidentData  (cost=2.50..2.50 rows=0) (actual time=4009.372..4009.383 rows=49 loops=1)
                        -> Materialize CTE AccidentData  (cost=0.00..0.00 rows=0) (actual time=4009.371..4009.371 rows=49 loops=1)
                            -> Sort: TotalDerailments DESC  (actual time=4009.336..4009.342 rows=49 loops=1)
                                -> Table scan on <temporary>  (actual time=4009.283..4009.293 rows=49 loops=1)
                                    -> Aggregate using temporary table  (actual time=4009.280..4009.280 rows=49 loops=1)
                                        -> Nested loop inner join  (cost=2704.44 rows=3) (actual time=1748.871..4006.908 rows=3059 loops=1)
                                            -> Filter: ((acc.ACC_TYPE = 1) and (acc.`DATE` >= DATE'2013-01-01') and (acc.`DATE` <= DATE'2022-12-31') and (acc.TRAIN_TYPE = 'F') and (acc.TRACK
_TYPE = 'MS') and (acc.RAILROAD is not null))  (cost=2703.50 rows=3) (actual time=1748.835..3765.480 rows=3059 loops=1)
                                                -> Table scan on acc  (cost=2703.50 rows=24145) (actual time=246.041..3744.152 rows=25353 loops=1)
                                            -> Single-row covering index lookup on class using PRIMARY (RAILROAD=acc.RAILROAD)  (cost=0.29 rows=1) (actual time=0.079..0.079 rows=1 loops=3059
)
                -> Index lookup on TrafficData using <auto_key0> (State=AccidentData.`Code`)  (actual time=1061.513..1061.513 rows=1 loops=49)
                    -> Materialize CTE TrafficData  (cost=0.00..0.00 rows=0) (actual time=52014.090..52014.090 rows=52 loops=1)
                        -> Sort: TotalMiles DESC  (actual time=52014.025..52014.029 rows=52 loops=1)
                            -> Table scan on <temporary>  (actual time=52013.969..52013.978 rows=52 loops=1)
                                -> Aggregate using temporary table  (actual time=52013.966..52013.966 rows=52 loops=1)
                                    -> Nested loop inner join  (cost=45804.95 rows=99609) (actual time=241.275..51691.827 rows=97529 loops=1)
                                        -> Filter: ((concat('20',traffic.IYR) >= '2013') and (concat('20',traffic.IYR) <= '2022') and (traffic.RAILROAD is not null))  (cost=10941.80 rows=996
09) (actual time=241.246..49234.762 rows=97529 loops=1)
                                            -> Table scan on traffic  (cost=10941.80 rows=99609) (actual time=241.203..48582.400 rows=121276 loops=1)
                                        -> Single-row covering index lookup on class using PRIMARY (RAILROAD=traffic.RAILROAD)  (cost=0.25 rows=1) (actual time=0.025..0.025 rows=1 loops=9752
9)
```

INDEX 1:

INDEX on RR_ACCIDENTS.STATE_CODE, RR_ACCIDENTS.DATE (place index on some non-primary key columns from where, and group by)

```
CREATE INDEX rr_accidents_idx ON RR_ACCIDENTS(STATE_CODE);
CREATE INDEX accidents_date_idx ON RR_ACCIDENTS(DATE);
ALTER TABLE RR_ACCIDENTS DROP INDEX rr_accidents_idx;
ALTER TABLE RR_ACCIDENTS DROP INDEX accidents_date_idx;
```



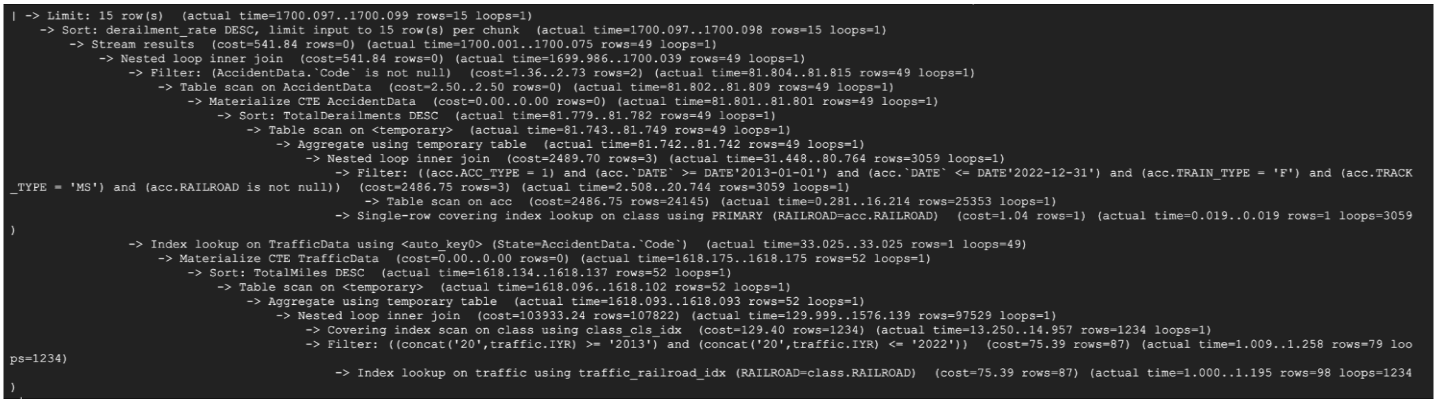
```
| -> Limit: 15 row(s)  (actual time=2125.374..2125.376 rows=15 loops=1)
    -> Sort: derailment_rate DESC, limit input to 15 row(s) per chunk  (actual time=2125.373..2125.374 rows=15 loops=1)
        -> Stream results  (cost=2992.12 rows=0) (actual time=2125.220..2125.343 rows=49 loops=1)
            -> Nested loop inner join  (cost=2992.12 rows=0) (actual time=2125.209..2125.296 rows=49 loops=1)
                -> Filter: (AccidentData.`Code` is not null)  (cost=0.32..3.85 rows=12) (actual time=25.814..25.828 rows=49 loops=1)
                    -> Table scan on AccidentData  (cost=2.50..2.50 rows=0) (actual time=25.812..25.822 rows=49 loops=1)
                        -> Materialize CTE AccidentData  (cost=0.00..0.00 rows=0) (actual time=25.812..25.812 rows=49 loops=1)
                            -> Sort: TotalDerailments DESC  (actual time=25.787..25.789 rows=49 loops=1)
                                -> Table scan on <temporary>  (actual time=25.707..25.712 rows=49 loops=1)
                                    -> Aggregate using temporary table  (actual time=25.705..25.705 rows=49 loops=1)
                                        -> Nested loop inner join  (cost=2490.98 rows=12) (actual time=2.475..24.892 rows=3059 loops=1)
                                            -> Filter: ((acc.ACC_TYPE = 1) and (acc.`DATE` >= DATE'2013-01-01') and (acc.`DATE` <= DATE'2022-12-31') and (acc.TRAIN_TYPE = 'F') and (acc.TRACK
_TYPE = 'MS') and (acc.RAILROAD is not null))  (cost=2486.75 rows=12) (actual time=2.454..20.517 rows=3059 loops=1)
                                                -> Table scan on acc  (cost=2486.75 rows=24145) (actual time=0.199..15.983 rows=25353 loops=1)
                                            -> Single-row covering index lookup on class using PRIMARY (RAILROAD=acc.RAILROAD)  (cost=0.26 rows=1) (actual time=0.001..0.001 rows=1 loops=3059
)
                -> Index lookup on TrafficData using <auto_key0> (State=AccidentData.`Code`)  (actual time=42.846..42.846 rows=1 loops=49)
                    -> Materialize CTE TrafficData  (cost=0.00..0.00 rows=0) (actual time=2099.387..2099.387 rows=52 loops=1)
                        -> Sort: TotalMiles DESC  (actual time=2099.316..2099.321 rows=52 loops=1)
                            -> Table scan on <temporary>  (actual time=2099.261..2099.271 rows=52 loops=1)
                                -> Aggregate using temporary table  (actual time=2099.260..2099.260 rows=52 loops=1)
                                    -> Nested loop inner join  (cost=45844.57 rows=99609) (actual time=53.220..2030.835 rows=97529 loops=1)
                                        -> Filter: ((concat('20',traffic.IYR) >= '2013') and (concat('20',traffic.IYR) <= '2022') and (traffic.RAILROAD is not null))  (cost=10981.42 rows=996
09) (actual time=53.186..1950.199 rows=97529 loops=1)
                                            -> Table scan on traffic  (cost=10981.42 rows=99609) (actual time=53.099..1834.808 rows=121276 loops=1)
                                        -> Single-row covering index lookup on class using PRIMARY (RAILROAD=traffic.RAILROAD)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=9752
9)
```

INDEX 2:

INDEX on RR_TRAFFIC.RAILROAD, RR_ACCIDENTS.STATE_CODE (place index on some non-primary key columns from join,where, and group by)

```
CREATE INDEX rr_traffic_railroad_idx ON RR_TRAFFIC(RAILROAD);
CREATE INDEX rr_accidents_idx ON RR_ACCIDENTS(STATE_CODE);
ALTER TABLE RR_TRAFFIC DROP INDEX rr_traffic_railroad_idx;
```

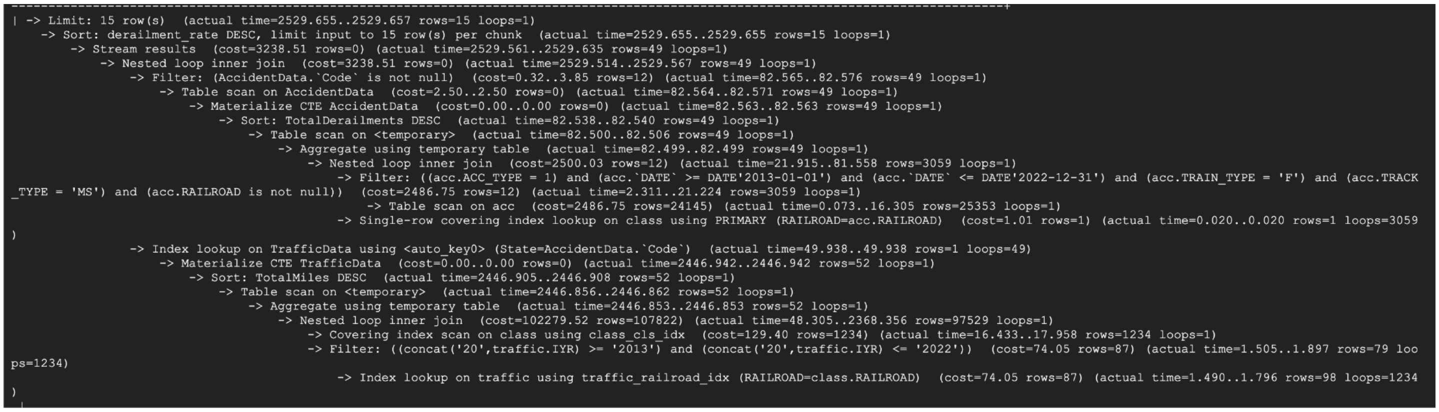
ALTER TABLE RR_ACCIDENTS DROP INDEX rr_accidents_idx;



```
| -> Limit: 15 row(s)  (actual time=1700.097..1700.099 rows=15 loops=1)
    -> Sort: derailment_rate DESC, limit input to 15 row(s) per chunk  (actual time=1700.097..1700.098 rows=15 loops=1)
        -> Stream results  (cost=541.84 rows=0) (actual time=1700.001..1700.075 rows=49 loops=1)
            -> Nested loop inner join  (cost=541.84 rows=0) (actual time=1699.986..1700.039 rows=49 loops=1)
                -> Filter: (AccidentData.`Code` is not null)  (cost=1.36..2.73 rows=2) (actual time=81.804..81.815 rows=49 loops=1)
                    -> Table scan on AccidentData  (cost=2.50..2.50 rows=0) (actual time=81.802..81.809 rows=49 loops=1)
                        -> Materialize CTE AccidentData  (cost=0.00..0.00 rows=0) (actual time=81.801..81.801 rows=49 loops=1)
                            -> Sort: TotalDerailments DESC  (actual time=81.779..81.782 rows=49 loops=1)
                                -> Table scan on <temporary>  (actual time=81.743..81.749 rows=49 loops=1)
                                    -> Aggregate using temporary table  (actual time=81.742..81.742 rows=49 loops=1)
                                        -> Nested loop inner join  (cost=2489.70 rows=3) (actual time=31.448..80.764 rows=3059 loops=1)
                                            -> Filter: ((acc.ACC_TYPE = 1) and (acc.`DATE` >= DATE'2013-01-01') and (acc.`DATE` <= DATE'2022-12-31') and (acc.TRAIN_TYPE = 'F') and (acc.TRACK
_TYPE = 'MS') and (acc.RAILROAD is not null))  (cost=2486.75 rows=3) (actual time=2.508..20.744 rows=3059 loops=1)
                                                -> Table scan on acc  (cost=2486.75 rows=24145) (actual time=0.281..16.214 rows=25353 loops=1)
                                            -> Single-row covering index lookup on class using PRIMARY (RAILROAD=acc.RAILROAD)  (cost=1.04 rows=1) (actual time=0.019..0.019 rows=1 loops=3059
)
                -> Index lookup on TrafficData using <auto_key0> (State=AccidentData.`Code`)  (actual time=33.025..33.025 rows=1 loops=49)
                    -> Materialize CTE TrafficData  (cost=0.00..0.00 rows=0) (actual time=1618.175..1618.175 rows=52 loops=1)
                        -> Sort: TotalMiles DESC  (actual time=1618.134..1618.137 rows=52 loops=1)
                            -> Table scan on <temporary>  (actual time=1618.096..1618.102 rows=52 loops=1)
                                -> Aggregate using temporary table  (actual time=1618.093..1618.093 rows=52 loops=1)
                                    -> Nested loop inner join  (cost=103933.24 rows=107822) (actual time=129.999..1576.139 rows=97529 loops=1)
                                        -> Covering index scan on class using class_cls_idx  (cost=129.40 rows=1234) (actual time=13.250..14.957 rows=1234 loops=1)
                                        -> Filter: ((concat('20',traffic.IYR) >= '2013') and (concat('20',traffic.IYR) <= '2022'))  (cost=75.39 rows=87) (actual time=1.009..1.258 rows=79 loo
ps=1234)
                                            -> Index lookup on traffic using traffic_railroad_idx (RAILROAD=class.RAILROAD)  (cost=75.39 rows=87) (actual time=1.000..1.195 rows=98 loops=1234
)
```

INDEX 3:
INDEX on RR_TRAFFIC.RAILROAD, RR_ACCIDENTS.STATE_CODE (place index on some non-primary key columns from join, where)

CREATE INDEX rr_traffic_railroad_idx ON RR_TRAFFIC(RAILROAD);
CREATE INDEX accidents_date_idx ON RR_ACCIDENTS(DATE);
ALTER TABLE RR_TRAFFIC DROP INDEX rr_traffic_railroad_idx;
ALTER TABLE RR_ACCIDENTS DROP INDEX accidents_date_idx;



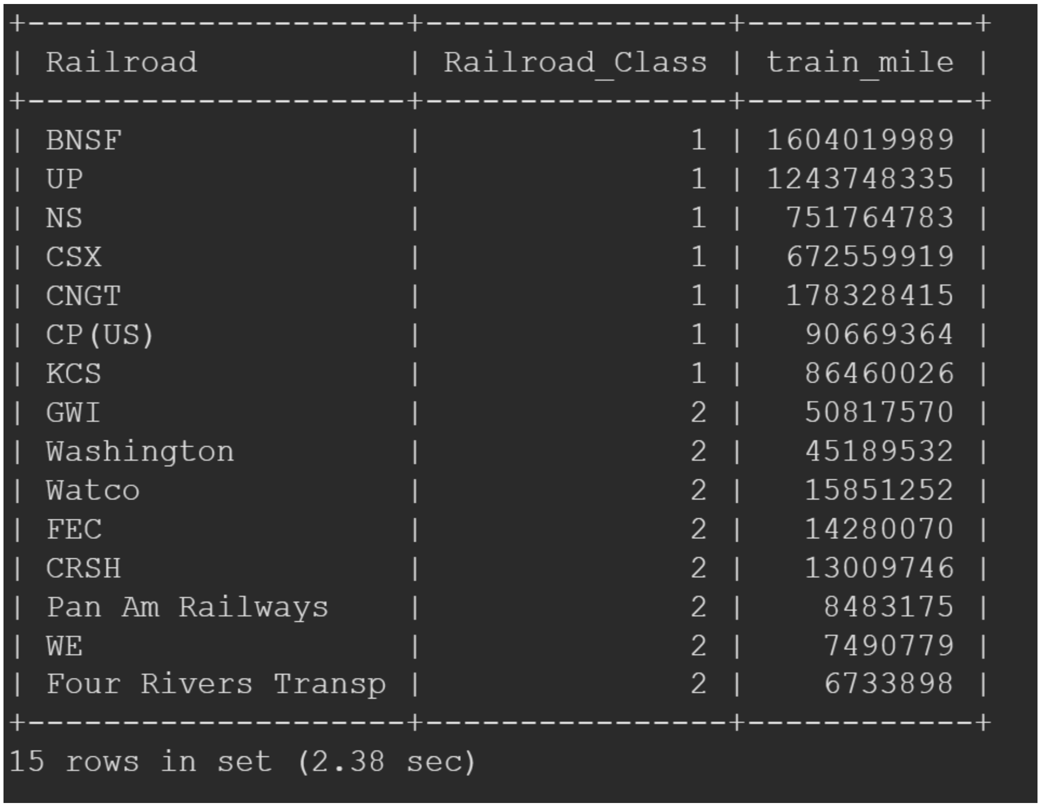
```
--------------------------------------------------------------------------------------------------------------------------------------+
| -> Limit: 15 row(s)  (actual time=2529.655..2529.657 rows=15 loops=1)
    -> Sort: derailment_rate DESC, limit input to 15 row(s) per chunk  (actual time=2529.655..2529.655 rows=15 loops=1)
        -> Stream results  (cost=3238.51 rows=0) (actual time=2529.561..2529.635 rows=49 loops=1)
            -> Nested loop inner join  (cost=3238.51 rows=0) (actual time=2529.514..2529.567 rows=49 loops=1)
                -> Filter: (AccidentData.`Code` is not null)  (cost=0.32..3.85 rows=12) (actual time=82.565..82.576 rows=49 loops=1)
                    -> Table scan on AccidentData  (cost=2.50..2.50 rows=0) (actual time=82.564..82.571 rows=49 loops=1)
                        -> Materialize CTE AccidentData  (cost=0.00..0.00 rows=0) (actual time=82.563..82.563 rows=49 loops=1)
                            -> Sort: TotalDerailments DESC  (actual time=82.538..82.540 rows=49 loops=1)
                                -> Table scan on <temporary>  (actual time=82.500..82.506 rows=49 loops=1)
                                    -> Aggregate using temporary table  (actual time=82.499..82.499 rows=49 loops=1)
                                        -> Nested loop inner join  (cost=2500.03 rows=12) (actual time=21.915..81.558 rows=3059 loops=1)
                                            -> Filter: ((acc.ACC_TYPE = 1) and (acc.`DATE` >= DATE'2013-01-01') and (acc.`DATE` <= DATE'2022-12-31') and (acc.TRAIN_TYPE = 'F') and (acc.TRACK
_TYPE = 'MS') and (acc.RAILROAD is not null))  (cost=2486.75 rows=12) (actual time=2.311..21.224 rows=3059 loops=1)
                                                -> Table scan on acc  (cost=2486.75 rows=24145) (actual time=0.073..16.305 rows=25353 loops=1)
                                            -> Single-row covering index lookup on class using PRIMARY (RAILROAD=acc.RAILROAD)  (cost=1.01 rows=1) (actual time=0.020..0.020 rows=1 loops=3059
)
                -> Index lookup on TrafficData using <auto_key0> (State=AccidentData.`Code`)  (actual time=49.938..49.938 rows=1 loops=49)
                    -> Materialize CTE TrafficData  (cost=0.00..0.00 rows=0) (actual time=2446.942..2446.942 rows=52 loops=1)
                        -> Sort: TotalMiles DESC  (actual time=2446.905..2446.908 rows=52 loops=1)
                            -> Table scan on <temporary>  (actual time=2446.856..2446.862 rows=52 loops=1)
                                -> Aggregate using temporary table  (actual time=2446.853..2446.853 rows=52 loops=1)
                                    -> Nested loop inner join  (cost=102279.52 rows=107822) (actual time=48.305..2368.356 rows=97529 loops=1)
                                        -> Covering index scan on class using class_cls_idx  (cost=129.40 rows=1234) (actual time=16.433..17.958 rows=1234 loops=1)
                                        -> Filter: ((concat('20',traffic.IYR) >= '2013') and (concat('20',traffic.IYR) <= '2022'))  (cost=74.05 rows=87) (actual time=1.505..1.897 rows=79 loo
ps=1234)
                                            -> Index lookup on traffic using traffic_railroad_idx (RAILROAD=class.RAILROAD)  (cost=74.05 rows=87) (actual time=1.490..1.796 rows=98 loops=1234
)
```

Report:
From the EXPLAIN ANALYZE results of the three different indexings above, there is no much better effect than without an index. We would choose Index 2 for this query because it performs slightly better than the others. Thus, the best indexing approach is not using any additional index. The reason is that the columns applied to the index do not have too many unique values.

5: Given a time period (2013-2022), happened on mainline and siding only, for class 1 railroad accidents only, I want to see the freight traffic volume by RR class

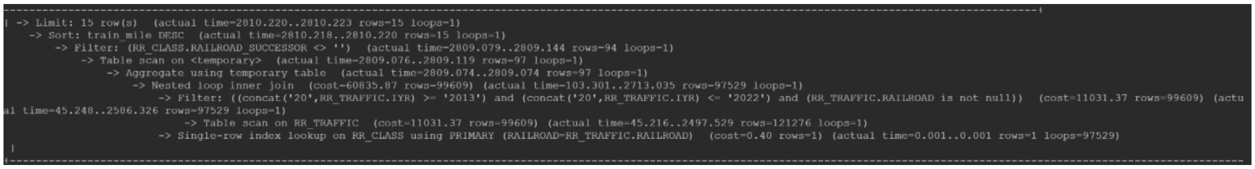
Code:

```
SELECT RR_CLASS.RAILROAD_SUCCESSOR AS Railroad,
RR_CLASS.RRCLASSIFICATION AS Railroad_Class,
SUM(RR_TRAFFIC.FRTRNMI + RR_TRAFFIC.OTHERMI) AS train_mile
    FROM RR_TRAFFIC
    JOIN RR_CLASS ON RR_TRAFFIC.RAILROAD = RR_CLASS.RAILROAD
    WHERE CONCAT('20', RR_TRAFFIC.IYR) >= '2013'
    AND CONCAT('20', RR_TRAFFIC.IYR) <= '2022'
    GROUP BY RR_CLASS.RAILROAD_SUCCESSOR,
RR_CLASS.RRCLASSIFICATION
    HAVING RR_CLASS.RAILROAD_SUCCESSOR != ''
    ORDER BY train_mile DESC
    LIMIT 15;
```



```
+--------------------+----------------+------------+
| Railroad           | Railroad_Class | train_mile |
+--------------------+----------------+------------+
| BNSF               |              1 | 1604019989 |
| UP                 |              1 | 1243748335 |
| NS                 |              1 |  751764783 |
| CSX                |              1 |  672559919 |
| CNGT               |              1 |  178328415 |
| CP(US)             |              1 |   90669364 |
| KCS                |              1 |   86460026 |
| GWI                |              2 |   50817570 |
| Washington         |              2 |   45189532 |
| Watco              |              2 |   15851252 |
| FEC                |              2 |   14280070 |
| CRSH               |              2 |   13009746 |
| Pan Am Railways    |              2 |    8483175 |
| WE                 |              2 |    7490779 |
| Four Rivers Transp |              2 |    6733898 |
+--------------------+----------------+------------+
15 rows in set (2.38 sec)
```

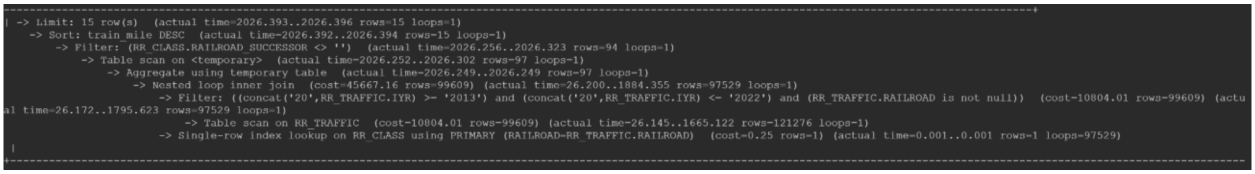
EXPLAIN ANALYZE without INDEX:



```
| -> Limit: 15 row(s)  (actual time=2810.220..2810.223 rows=15 loops=1)
    -> Sort: train_mile DESC  (actual time=2810.218..2810.220 rows=15 loops=1)
        -> Filter: (RR_CLASS.RAILROAD_SUCCESSOR <> '')  (actual time=2809.079..2809.144 rows=94 loops=1)
            -> Table scan on <temporary>  (actual time=2809.076..2809.119 rows=97 loops=1)
                -> Aggregate using temporary table  (actual time=2809.074..2809.074 rows=97 loops=1)
                    -> Nested loop inner join  (cost=60835.87 rows=99609) (actual time=103.301..2713.035 rows=97529 loops=1)
                        -> Filter: ((concat('20',RR_TRAFFIC.IYR) >= '2013') and (concat('20',RR_TRAFFIC.IYR) <= '2022') and (RR_TRAFFIC.RAILROAD is not null))  (cost=11031.37 rows=99609) (actu
al time=45.248..2586.326 rows=97529 loops=1)
                            -> Table scan on RR_TRAFFIC  (cost=11031.37 rows=99609) (actual time=45.216..2497.529 rows=121276 loops=1)
                        -> Single-row index lookup on RR_CLASS using PRIMARY (RAILROAD=RR_TRAFFIC.RAILROAD)  (cost=0.40 rows=1) (actual time=0.001..0.001 rows=1 loops=97529)
 |
```

INDEX 1: INDEX ON RR_TRAFFIC.RAILROAD, RR_CLASS.RAILROAD_SUCCESSOR, RR_CLASS.RRCLASSIFICATION (place index on some non-primary key columns from join, where, group by, and having)

```
CREATE INDEX traffic_railroad_idx ON RR_TRAFFIC(RAILROAD);
CREATE INDEX class_classification_idx ON
RR_CLASS(RRCLASSIFICATION);
CREATE INDEX class_successor_idx ON
RR_CLASS(RAILROAD_SUCCESSOR);
```

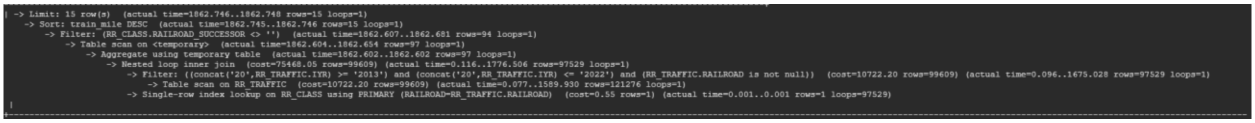


```
| -> Limit: 15 row(s)  (actual time=2026.393..2026.396 rows=15 loops=1)
    -> Sort: train_mile DESC  (actual time=2026.392..2026.394 rows=15 loops=1)
        -> Filter: (RR_CLASS.RAILROAD_SUCCESSOR <> '')  (actual time=2026.256..2026.323 rows=94 loops=1)
            -> Table scan on <temporary>  (actual time=2026.252..2026.302 rows=97 loops=1)
                -> Aggregate using temporary table  (actual time=2026.249..2026.249 rows=97 loops=1)
                    -> Nested loop inner join  (cost=45667.16 rows=99609) (actual time=26.200..1884.355 rows=97529 loops=1)
                        -> Filter: ((concat('20',RR_TRAFFIC.IYR) >= '2013') and (concat('20',RR_TRAFFIC.IYR) <= '2022') and (RR_TRAFFIC.RAILROAD is not null))  (cost=10804.01 rows=99609) (actu
al time=26.172..1795.623 rows=97529 loops=1)
                            -> Table scan on RR_TRAFFIC  (cost=10804.01 rows=99609) (actual time=26.145..1665.122 rows=121276 loops=1)
                        -> Single-row index lookup on RR_CLASS using PRIMARY (RAILROAD=RR_TRAFFIC.RAILROAD)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=97529)
 |
```

The inner join cost is reduced from 60835 to 45667, and table scan cost is reduced from 11031 to 10804.

INDEX 2: INDEX ON RR_TRAFFIC.RAILROAD (place index on non-primary key columns from join)

```
CREATE INDEX traffic_railroad_idx ON RR_TRAFFIC(RAILROAD);
```



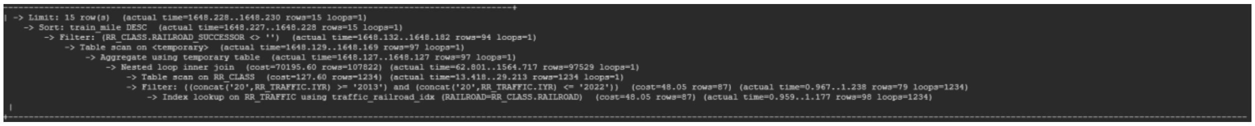
```
| -> Limit: 15 row(s)  (actual time=1862.746..1862.748 rows=15 loops=1)
    -> Sort: train_mile DESC  (actual time=1862.745..1862.746 rows=15 loops=1)
        -> Filter: (RR_CLASS.RAILROAD_SUCCESSOR <> '')  (actual time=1862.607..1862.681 rows=94 loops=1)
            -> Table scan on <temporary>  (actual time=1862.604..1862.654 rows=97 loops=1)
                -> Aggregate using temporary table  (actual time=1862.602..1862.602 rows=97 loops=1)
                    -> Nested loop inner join  (cost=75468.05 rows=99609) (actual time=0.116..1776.506 rows=97529 loops=1)
                        -> Filter: ((concat('20',RR_TRAFFIC.IYR) >= '2013') and (concat('20',RR_TRAFFIC.IYR) <= '2022') and (RR_TRAFFIC.RAILROAD is not null))  (cost=10722.20 rows=99609) (actual time=0.096..1675.028 rows=97529 loops=1)
                            -> Table scan on RR_TRAFFIC  (cost=10722.20 rows=99609) (actual time=0.077..1589.930 rows=121276 loops=1)
                        -> Single-row index lookup on RR_CLASS using PRIMARY (RAILROAD=RR_TRAFFIC.RAILROAD)  (cost=0.55 rows=1) (actual time=0.001..0.001 rows=1 loops=97529)
 |
```

The cost increases from 60835 to 75468.

INDEX 3: RR_TRAFFIC.RAILROAD, RR_CLASS.RRCLASSIFICATION, and RR_TRAFFIC (place index on some non-primary key columns from where, group by, and having)

```
CREATE INDEX class_classification_idx ON
RR_CLASS(RRCLASSIFICATION);
CREATE INDEX traffic_railroad_idx ON RR_TRAFFIC(RAILROAD);
```

CREATE INDEX class_year_idx ON RR_TRAFFIC(IYR);



```
| -> Limit: 15 row(s)  (actual time=1648.228..1648.230 rows=15 loops=1)
    -> Sort: train_mile DESC  (actual time=1648.227..1648.228 rows=15 loops=1)
        -> Filter: (RR_CLASS.RAILROAD_SUCCESSOR <> '')  (actual time=1648.132..1648.182 rows=94 loops=1)
            -> Table scan on <temporary>  (actual time=1648.129..1648.169 rows=97 loops=1)
                -> Aggregate using temporary table  (actual time=1648.127..1648.127 rows=97 loops=1)
                    -> Nested loop inner join  (cost=70195.60 rows=107822) (actual time=62.801..1564.717 rows=97529 loops=1)
                        -> Table scan on RR_CLASS  (cost=127.60 rows=1234) (actual time=13.418..29.213 rows=1234 loops=1)
                        -> Filter: ((concat('20',RR_TRAFFIC.IYR) >= '2013') and (concat('20',RR_TRAFFIC.IYR) <= '2022'))  (cost=48.05 rows=87) (actual time=0.967..1.238 rows=79 loops=1234)
                            -> Index lookup on RR_TRAFFIC using traffic_railroad_idx (RAILROAD=RR_CLASS.RAILROAD)  (cost=48.05 rows=87) (actual time=0.959..1.177 rows=98 loops=1234)
 |
```

The cost increases from 60835 to 70195.

Thus, based on the above indexing approaches, for query 5, we can go with index approach 1, which is create an index on RR_TRAFFIC.RAILROAD, RR_CLASS.RAILROAD_SUCCESSOR, RR_CLASS.RRCLASSIFICATION.