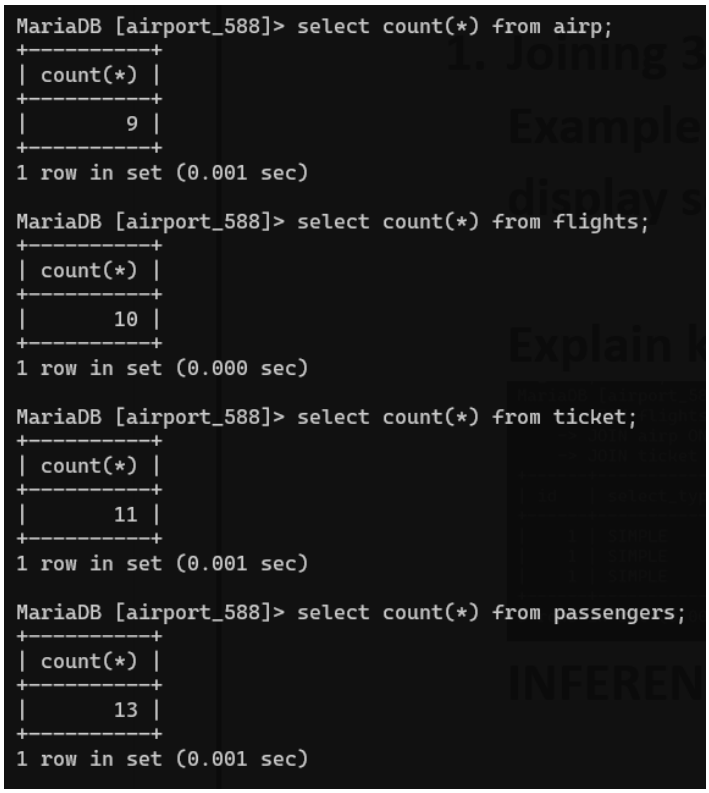
DBT-ASSIGNMENT 2

NAME: SAHANA RAO	SRN: PES1UG20CS588	SECTION: J

Displaying rows in all 4 tables:



```
MariaDB [airport_588]> select count(*) from airp;
+----------+
| count(*) |
+----------+
|        9 |
+----------+
1 row in set (0.001 sec)

MariaDB [airport_588]> select count(*) from flights;
+----------+
| count(*) |
+----------+
|       10 |
+----------+
1 row in set (0.000 sec)

MariaDB [airport_588]> select count(*) from ticket;
+----------+
| count(*) |
+----------+
|       11 |
+----------+
1 row in set (0.001 sec)

MariaDB [airport_588]> select count(*) from passengers;
+----------+
| count(*) |
+----------+
|       13 |
+----------+
1 row in set (0.001 sec)
```

1)Execute plan a select query that has a multi (>2) table join of min. of 3 tables. Repeat this with 2 or more different sets of tables.

Example 1:

Display airport id,source airport name,destination,passenger ssn,ticket id,seat no and class of all passengers who source airport name starts with m.

Procedure: Join tables airport(airp), flights and ticket and display selected attributes

Order 1: Flight->airport->ticket

Explain keyword:

```
MariaDB [airport_588]> SELECT A.airport_id,A.airport_name,F.source,F.destination,T.ssn,T.tid,T.seat_no,T.class
    -> FROM flights F JOIN airp A JOIN ticket T ON T.flight_id = F.flight_id and A.city=F.source
    -> where city like 'M%';
+------------+---------------------------------+-----------+-------------+-----+-----+---------+---------+
| airport_id | airport_name                    | source    | destination | ssn | tid | seat_no | class   |
+------------+---------------------------------+-----------+-------------+-----+-----+---------+---------+
|          1 | Mangaluru Int Port              | Mangaluru | Chennai     | 123 |  95 |      15 | Economy |
|          1 | Mangaluru Int Port              | Mangaluru | Chennai     | 357 | 104 |      41 | Economy |
|          5 | Chatrapati Shivaji Int Airport  | Mumbai    | Delhi       | 741 | 105 |      12 | Economy |
+------------+---------------------------------+-----------+-------------+-----+-----+---------+---------+
3 rows in set (0.001 sec)

MariaDB [airport_588]>
```

```
MariaDB [airport_588]> explain SELECT A.airport_id,A.airport_name,F.source,F.destination,T.ssn,T.tid,T.seat_no,T.class
    -> FROM flights F JOIN airp A JOIN ticket T ON T.flight_id = F.flight_id and A.city=F.source
    -> where city like 'M%';
+------+-------------+-------+--------+---------------+---------+---------+-------------------------+------+-----------------------------------+
| id   | select_type | table | type   | possible_keys | key     | key_len | ref                     | rows | Extra                             |
+------+-------------+-------+--------+---------------+---------+---------+-------------------------+------+-----------------------------------+
|    1 | SIMPLE      | A     | ALL    | NULL          | NULL    | NULL    | NULL                    | 9    | Using where                       |
|    1 | SIMPLE      | T     | ALL    | flight_id     | NULL    | NULL    | NULL                    | 10   | Using join buffer (flat, BNL join) |
|    1 | SIMPLE      | F     | eq_ref | PRIMARY       | PRIMARY | 4       | airport_588.T.flight_id | 1    | Using where                       |
+------+-------------+-------+--------+---------------+---------+---------+-------------------------+------+-----------------------------------+
3 rows in set (0.001 sec)
```

INFERENCE:

Queries all rows of the tables to join

Optimisation:

Create index on city and source from airp and flights table respectively.

```
MariaDB [airport_588]> CREATE INDEX city_index ON airp (city);
Query OK, 0 rows affected (0.009 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [airport_588]>
MariaDB [airport_588]> create index source_index on flights(source)
Query OK, 0 rows affected (0.016 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Explain keyword:

```
MariaDB [airport_588]> explain SELECT A.airport_id,A.airport_name,F.source,F.destination,T.ssn,T.tid,T.seat_no,T.class
    -> FROM flights F JOIN airp A JOIN ticket T ON T.flight_id = F.flight_id and A.city=F.source
    -> where city like 'M%';
+------+-------------+-------+-------+----------------------+-------------+---------+-------------------------+------+-----------------------+
| id   | select_type | table | type  | possible_keys        | key         | key_len | ref                     | rows | Extra                 |
+------+-------------+-------+-------+----------------------+-------------+---------+-------------------------+------+-----------------------+
|    1 | SIMPLE      | A     | range | city_index           | city_index  | 1022    | NULL                    | 2    | Using index condition |
|    1 | SIMPLE      | F     | ref   | PRIMARY,source_index | source_index| 202     | airport_588.A.city      | 1    | Using index condition |
|    1 | SIMPLE      | T     | ref   | flight_id            | flight_id   | 4       | airport_588.F.flight_id | 1    |                       |
+------+-------------+-------+-------+----------------------+-------------+---------+-------------------------+------+-----------------------+
3 rows in set (0.001 sec)
```

The number of rows queried has reduced from 9,10 to 2,1 in airp and flights respectively.

<u>Order 2:</u> Airport->flight->ticket

```
MariaDB [airport_588]> EXPLAIN SELECT A.airport_id,A.airport_name,F.source,F.destination,T.ssn,T.tid,T.seat_no,T.class
    -> FROM airp A JOIN flights F JOIN ticket T ON A.city=F.source AND T.flight_id = F.flight_id
    -> where city like 'M%';
+------+-------------+-------+--------+---------------+---------+---------+-------------------------+------+-----------------------------------+
| id   | select_type | table | type   | possible_keys | key     | key_len | ref                     | rows | Extra                             |
+------+-------------+-------+--------+---------------+---------+---------+-------------------------+------+-----------------------------------+
|    1 | SIMPLE      | A     | ALL    | NULL          | NULL    | NULL    | NULL                    | 9    | Using where                       |
|    1 | SIMPLE      | T     | ALL    | flight_id     | NULL    | NULL    | NULL                    | 10   | Using join buffer (flat, BNL join) |
|    1 | SIMPLE      | F     | eq_ref | PRIMARY       | PRIMARY | 4       | airport_588.T.flight_id | 1    | Using where                       |
+------+-------------+-------+--------+---------------+---------+---------+-------------------------+------+-----------------------------------+
3 rows in set (0.001 sec)
```

Optimisation:

Create index on city and source from airp and flights table respectively.

```
MariaDB [airport_588]> CREATE INDEX city_index ON airp (city);
Query OK, 0 rows affected (0.009 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [airport_588]>
MariaDB [airport_588]> create index source_index on flights(source);
Query OK, 0 rows affected (0.016 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Explain keyword:

```
MariaDB [airport_588]> EXPLAIN SELECT A.airport_id,A.airport_name,F.source,F.destination,T.ssn,T.tid,T.seat_no,T.class
    -> FROM airp A JOIN flights F JOIN ticket T ON A.city=F.source AND T.flight_id = F.flight_id
    -> where city like 'M%';
+------+-------------+-------+-------+---------------------+--------------+---------+-------------------------+------+-----------------------+
| id   | select_type | table | type  | possible_keys       | key          | key_len | ref                     | rows | Extra                 |
+------+-------------+-------+-------+---------------------+--------------+---------+-------------------------+------+-----------------------+
|    1 | SIMPLE      | A     | range | city_index          | city_index   | 1022    | NULL                    | 2    | Using index condition |
|    1 | SIMPLE      | F     | ref   | PRIMARY,source_index | source_index | 202     | airport_588.A.city      | 1    | Using index condition |
|    1 | SIMPLE      | T     | ref   | flight_id           | flight_id    | 4       | airport_588.F.flight_id | 1    |                       |
+------+-------------+-------+-------+---------------------+--------------+---------+-------------------------+------+-----------------------+
3 rows in set (0.001 sec)
```

The number of rows queried has reduced from 9,10 to 2,1 in airp and flights respectively.

<u>ORDER 3:</u> Airport->ticket->flights

```
MariaDB [airport_588]> explain SELECT A.airport_id,A.airport_name,F.source,F.destination,T.ssn,T.tid,T.seat_no,T.class
    -> FROM airp A JOIN ticket T JOIN flights F  ON A.city=F.source AND T.flight_id = F.flight_id
    -> where city like 'M%';
+------+-------------+-------+------+---------------+---------+---------+-------------------------+------+------------------------------------+
| id   | select_type | table | type | possible_keys | key     | key_len | ref                     | rows | Extra                              |
+------+-------------+-------+------+---------------+---------+---------+-------------------------+------+------------------------------------+
|    1 | SIMPLE      | A     | ALL  | NULL          | NULL    | NULL    | NULL                    | 9    | Using where                        |
|    1 | SIMPLE      | T     | ALL  | flight_id     | NULL    | NULL    | NULL                    | 10   | Using join buffer (flat, BNL join) |
|    1 | SIMPLE      | F     | eq_ref | PRIMARY     | PRIMARY | 4       | airport_588.T.flight_id | 1    | Using where                        |
+------+-------------+-------+------+---------------+---------+---------+-------------------------+------+------------------------------------+
3 rows in set (0.001 sec)
```

Optimisation:

Create index on city and source from airp and flights table respectively.

```
MariaDB [airport_588]> CREATE INDEX city_index ON airp (city);
Query OK, 0 rows affected (0.019 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [airport_588]> create index source_index on flights(source);
Query OK, 0 rows affected (0.018 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Explain keyword:

```
MariaDB [airport_588]> explain SELECT A.airport_id,A.airport_name,F.source,F.destination,T.ssn,T.tid,T.seat_no,T.class
    -> FROM airp A JOIN ticket T JOIN flights F  ON A.city=F.source AND T.flight_id = F.flight_id
    -> where city like 'M%';
+------+-------------+-------+-------+---------------------+--------------+---------+-------------------------+------+-----------------------+
| id   | select_type | table | type  | possible_keys       | key          | key_len | ref                     | rows | Extra                 |
+------+-------------+-------+-------+---------------------+--------------+---------+-------------------------+------+-----------------------+
|    1 | SIMPLE      | A     | range | city_index          | city_index   | 1022    | NULL                    | 2    | Using index condition |
|    1 | SIMPLE      | F     | ref   | PRIMARY,source_index | source_index | 202     | airport_588.A.city      | 1    | Using index condition |
|    1 | SIMPLE      | T     | ref   | flight_id           | flight_id    | 4       | airport_588.F.flight_id | 1    |                       |
+------+-------------+-------+-------+---------------------+--------------+---------+-------------------------+------+-----------------------+
3 rows in set (0.001 sec)
```

The number of rows queried has reduced from 9,10 to 2,1 in airp and flights respectively.

EXAMPLE 2:

Display all passengers along with their ssn, first_name, TimeOfFly ,ticket_id,seat no irrespective of they have ticket or not.

Order1: passengers->ticket->flights

```
MariaDB [airport_588]> SELECT P.first_name,P.SSN,F.flight_id,F.TimeOfFly,T.tid,T.seat_no
    -> FROM passengers P left outer JOIN ticket T ON P.ssn=T.ssn
    -> left outer JOIN flights F ON T.flight_id = F.flight_id
    -> where P.age>20 and P.first_name like 'R%';
+------------+-----+-----------+-----------+------+---------+
| first_name | SSN | flight_id | TimeOfFly | tid  | seat_no |
+------------+-----+-----------+-----------+------+---------+
| Radhika    | 470 |      NULL |      NULL | NULL |    NULL |
| Reshma     | 741 |        45 |         2 |  105 |      12 |
| Rahul      | 842 |        70 |         2 |  102 |      11 |
+------------+-----+-----------+-----------+------+---------+
3 rows in set (0.001 sec)
```

Explain keyword:

```
MariaDB [airport_588]> explain SELECT P.first_name,P.SSN,F.flight_id,F.TimeOfFly,T.tid,T.seat_no
    -> FROM passengers P left outer JOIN ticket T ON P.ssn=T.ssn
    -> left outer JOIN flights F ON T.flight_id = F.flight_id
    -> where P.age>20 and P.first_name like 'R%';
+------+-------------+-------+--------+---------------+---------+---------+-------------------------+------+-------------+
| id   | select_type | table | type   | possible_keys | key     | key_len | ref                     | rows | Extra       |
+------+-------------+-------+--------+---------------+---------+---------+-------------------------+------+-------------+
|    1 | SIMPLE      | P     | ALL    | NULL          | NULL    | NULL    | NULL                    | 13   | Using where |
|    1 | SIMPLE      | T     | ref    | SSN           | SSN     | 4       | airport_588.P.SSN       | 1    |             |
|    1 | SIMPLE      | F     | eq_ref | PRIMARY       | PRIMARY | 4       | airport_588.T.flight_id | 1    | Using where |
+------+-------------+-------+--------+---------------+---------+---------+-------------------------+------+-------------+
3 rows in set (0.001 sec)
```

Optimisation: Creating index on fname and age on passengers table

```
MariaDB [airport_588]> create index fname_index on passengers(first_name);
Query OK, 0 rows affected (0.015 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [airport_588]> create index age_index on passengers(age);
Query OK, 0 rows affected (0.016 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Explain keyword:

```
MariaDB [airport_588]> explain SELECT P.first_name,P.SSN,F.flight_id,F.TimeOfFly,T.tid,T.seat_no
    -> FROM passengers P left outer JOIN ticket T ON P.ssn=T.ssn
    -> left outer JOIN flights F ON T.flight_id = F.flight_id
    -> where P.age>20 and P.first_name like 'R%';
+------+-------------+-------+--------+-------------------+-------------+---------+-------------------------+------+-----------------------------------+
| id   | select_type | table | type   | possible_keys     | key         | key_len | ref                     | rows | Extra                             |
+------+-------------+-------+--------+-------------------+-------------+---------+-------------------------+------+-----------------------------------+
|    1 | SIMPLE      | P     | range  | fname_index,age_index | fname_index | 1022    | NULL                    | 3    | Using index condition; Using where |
|    1 | SIMPLE      | T     | ref    | SSN               | SSN         | 4       | airport_588.P.SSN       | 1    |                                   |
|    1 | SIMPLE      | F     | eq_ref | PRIMARY           | PRIMARY     | 4       | airport_588.T.flight_id | 1    | Using where                       |
+------+-------------+-------+--------+-------------------+-------------+---------+-------------------------+------+-----------------------------------+
3 rows in set (0.001 sec)
```

The number of rows queried reduced from 13 to 3 in passengers table.

Order 2: Ticket->FLIGHTS->PASSENGERS

```
MariaDB [airport_588]> SELECT P.first_name,P.SSN,F.flight_id,F.TimeOfFly,T.tid,T.seat_no
    -> FROM ticket T JOIN flights F ON T.flight_id = F.flight_id
    -> right outer join passengers P ON P.ssn=T.ssn
    -> where P.age>20 and P.first_name like 'R%';
+------------+------+-----------+-----------+------+---------+
| first_name | SSN  | flight_id | TimeOfFly | tid  | seat_no |
+------------+------+-----------+-----------+------+---------+
| Radhika    | 470  |      NULL |      NULL | NULL |    NULL |
| Reshma     | 741  |        45 |         2 |  105 |      12 |
| Rahul      | 842  |        70 |         2 |  102 |      11 |
+------------+------+-----------+-----------+------+---------+
3 rows in set (0.007 sec)

MariaDB [airport_588]>
```

Optimisation: Creating index on fname and age on passengers table

```
MariaDB [airport_588]> create index fname_index on passengers(first_name);
Query OK, 0 rows affected (0.015 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [airport_588]> create index age_index on passengers(age);
Query OK, 0 rows affected (0.018 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

The number of rows queried reduced from 13 to 3 in passengers table.

```
MariaDB [airport_588]> explain SELECT P.first_name,P.SSN,F.flight_id,F.TimeOfFly,T.tid,T.seat_no
    -> FROM ticket T JOIN flights F ON T.flight_id = F.flight_id
    -> right outer join passengers P ON P.ssn=T.ssn
    -> where P.age>20 and P.first_name like 'R%';
+------+-------------+-------+--------+-----------------------+-------------+---------+-----------------------+------+--------------------------------------+
| id   | select_type | table | type   | possible_keys         | key         | key_len | ref                   | rows | Extra                                |
+------+-------------+-------+--------+-----------------------+-------------+---------+-----------------------+------+--------------------------------------+
|    1 | SIMPLE      | P     | range  | fname_index,age_index | fname_index | 1022    | NULL                  | 3    | Using index condition; Using where   |
|    1 | SIMPLE      | T     | ref    | SSN,flight_id         | SSN         | 4       | airport_588.P.SSN     | 1    | Using where                          |
|    1 | SIMPLE      | F     | eq_ref | PRIMARY               | PRIMARY     | 4       | airport_588.T.flight_id | 1  |                                      |
+------+-------------+-------+--------+-----------------------+-------------+---------+-----------------------+------+--------------------------------------+
3 rows in set (0.001 sec)
```

2. Create a query that uses a subquery and a correlated subquery, and analyze its execution plan.

Query: To find the flight with least time of fly for every unique source-destination

```
MariaDB [airport_588]> SELECT a.airport_name as Source_Airport,f.flight_id,f.source,f.destination,f.departure_time,f.arrival_time
    -> FROM airp A INNER JOIN flights F ON f.source = A.city
    -> WHERE
    ->    f.TimeOfFly = (
    ->        SELECT MIN(TimeOfFly)
    ->        FROM flights F1
    ->        WHERE F1.source=f.source and f1.destination=f.destination
    ->    );
+-----------------------------------+-----------+----------+-------------+---------------------+---------------------+
| Source_Airport                    | flight_id | source   | destination | departure_time      | arrival_time        |
+-----------------------------------+-----------+----------+-------------+---------------------+---------------------+
| Mangaluru Int Port                |        10 | Mangaluru | Chennai    | 2023-03-14 09:05:02 | 2023-03-14 10:05:02 |
| Meenambakkam Int Airport          |        15 | Chennai  | Bengaluru   | 2023-03-15 12:05:02 | 2023-03-15 14:05:02 |
| Meenambakkam Int Airport          |        30 | Chennai  | Mangaluru   | 2023-03-30 21:09:26 | 2023-03-30 22:09:26 |
| Lokpriya Gopinath Bordoloi Int Airport |   40 | Guwahati | Delhi       | 2023-03-18 12:13:23 | 2023-03-18 13:13:23 |
| Chatrapati Shivaji Int Airport    |        45 | Mumbai   | Delhi       | 2023-03-22 05:13:23 | 2023-03-22 07:13:23 |
| Meenambakkam Int Airport          |        60 | Chennai  | Mumbai      | 2023-03-26 10:30:00 | 2023-03-26 11:30:00 |
| Indira Gandhi Int Airport         |        70 | Delhi    | Bengaluru   | 2023-03-29 13:13:23 | 2023-03-29 15:13:23 |
+-----------------------------------+-----------+----------+-------------+---------------------+---------------------+
7 rows in set (0.001 sec)

MariaDB [airport_588]>
```

Explain keyword:

```
MariaDB [airport_588]> explain SELECT a.airport_name as Source_Airport,f.flight_id,f.source,f.destination,f.departure_time,f.arrival_time
    -> FROM airp A INNER JOIN flights F ON f.source = A.city
    -> WHERE
    ->    f.TimeOfFly = (
    ->         SELECT MIN(TimeOfFly)
    ->         FROM flights F1
    ->         WHERE F1.source=f.source and f1.destination=f.destination
    ->     );
+------+--------------------+-------+------+---------------+------+---------+------+------+----------------------------------------------------+
| id   | select_type        | table | type | possible_keys | key  | key_len | ref  | rows | Extra                                              |
+------+--------------------+-------+------+---------------+------+---------+------+------+----------------------------------------------------+
|    1 | PRIMARY            | A     | ALL  | NULL          | NULL | NULL    | NULL | 9    |                                                    |
|    1 | PRIMARY            | F     | ALL  | NULL          | NULL | NULL    | NULL | 10   | Using where; Using join buffer (flat, BNL join)    |
|    2 | DEPENDENT SUBQUERY | F1    | ALL  | NULL          | NULL | NULL    | NULL | 10   | Using where                                        |
+------+--------------------+-------+------+---------------+------+---------+------+------+----------------------------------------------------+
3 rows in set (0.001 sec)
```

After creating index on (source,destination) and airport nam in Flights and airp table:

```
MariaDB [airport_588]> create index airportname_index on airp(airport_name);
Query OK, 0 rows affected (0.046 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [airport_588]> create index source_index on flights(source);
Query OK, 0 rows affected (0.010 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [airport_588]> create index destination_index on flights(destination);
Query OK, 0 rows affected (0.040 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Explain keyword:

```
MariaDB [airport_588]> explain SELECT a.airport_name as Source_Airport,f.flight_id,f.source,f.destination,f.departure_time,f.arrival_time
    -> FROM airp A INNER JOIN flights F ON f.source = A.city
    -> WHERE
    ->    f.TimeOfFly = (
    ->         SELECT MIN(TimeOfFly)
    ->         FROM flights F1
    ->         WHERE F1.source=f.source and f1.destination=f.destination
    ->     );
+------+--------------------+-------+------+-----------------------------+--------------+---------+--------------------+------+----------------------------------------------------+
| id   | select_type        | table | type | possible_keys               | key          | key_len | ref                | rows | Extra                                              |
+------+--------------------+-------+------+-----------------------------+--------------+---------+--------------------+------+----------------------------------------------------+
|    1 | PRIMARY            | A     | ALL  | NULL                        | NULL         | NULL    | NULL               | 9    |                                                    |
|    1 | PRIMARY            | F     | ALL  | source_index                | NULL         | NULL    | NULL               | 10   | Using where; Using join buffer (flat, BNL join)    |
|    2 | DEPENDENT SUBQUERY | F1    | ref  | source_index,destination_index | source_index | 202     | airport_588.F.source | 1    | Using where                                        |
+------+--------------------+-------+------+-----------------------------+--------------+---------+--------------------+------+----------------------------------------------------+
3 rows in set (0.001 sec)
```

The number of rows queried reduced from 10 to 1 in flights table.

3. Create a query that uses a materialized view and analyze its execution plan.

View consisting of all unique attributes of passenger,ticket and flights table:

```
MariaDB [airport_588]> CREATE VIEW f_p_t AS
    -> SELECT P.ssn,P.first_name,P.last_name,P.age,P.dob,P.gender,P.phone_number,T.tid,T.flight_id,T.class,T.seat_no,F.company,F.source,F.destination,F.arrival_time,F.departure_time,F.capacity,F.TimeOfFly
    -> FROM passengers P inner JOIN ticket T ON P.ssn=T.ssn
    -> inner JOIN flights F ON T.flight_id = F.flight_id;
Query OK, 0 rows affected (0.033 sec)
```

Query: Finding all passengers whose flight company starts with letter ‘A’

```
MariaDB [airport_588]> SELECT first_name,last_name,phone_number,source,destination
    -> FROM f_p_t
    -> WHERE company like 'A%';
+------------+-----------+--------------+---------+-------------+
| first_name | last_name | phone_number | source  | destination |
+------------+-----------+--------------+---------+-------------+
| Adi        | Sharma    | 9765231832   | Chennai | Bengaluru   |
| Yash       | J         | 7631963189   | Chennai | Bengaluru   |
| Priya      | V         | 93777763444  | Chennai | Bengaluru   |
| Rahul      | Deshpande | 7332211445   | Delhi   | Bengaluru   |
| Des        | Shetty    | 8648946586   | Delhi   | Bengaluru   |
| Emilia     | Johnes    | 94447777785  | Delhi   | Bengaluru   |
+------------+-----------+--------------+---------+-------------+
6 rows in set (0.002 sec)
```

Explain keyword:

```
MariaDB [airport_588]> explain SELECT first_name,last_name,phone_number,source,destination
    -> FROM f_p_t
    -> WHERE company like 'A%';
+------+-------------+-------+--------+-----------------+---------+---------+-------------------------+------+-------------+
| id   | select_type | table | type   | possible_keys   | key     | key_len | ref                     | rows | Extra       |
+------+-------------+-------+--------+-----------------+---------+---------+-------------------------+------+-------------+
|    1 | SIMPLE      | t     | ALL    | SSN,flight_index | NULL    | NULL    | NULL                    | 10   |             |
|    1 | SIMPLE      | f     | eq_ref | PRIMARY         | PRIMARY | 4       | airport_588.t.flight_id | 1    | Using where |
|    1 | SIMPLE      | p     | eq_ref | PRIMARY         | PRIMARY | 4       | airport_588.t.SSN       | 1    |             |
+------+-------------+-------+--------+-----------------+---------+---------+-------------------------+------+-------------+
3 rows in set (0.001 sec)

MariaDB [airport_588]>
```

Creating index on company in flights table:

```
MariaDB [airport_588]> create index company_index on flights(company);
Query OK, 0 rows affected (0.048 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Explain keyword:

```
MariaDB [airport_588]> explain SELECT first_name,last_name,phone_number,source,destination
    -> FROM f_p_t
    -> WHERE company like 'A%';
+------+-------------+-------+--------+---------------------+-------------+---------+-------------------------+------+-------------+
| id   | select_type | table | type   | possible_keys       | key         | key_len | ref                     | rows | Extra       |
+------+-------------+-------+--------+---------------------+-------------+---------+-------------------------+------+-------------+
|    1 | SIMPLE      | f     | ALL    | PRIMARY,company_index | NULL        | NULL    | NULL                    | 10   | Using where |
|    1 | SIMPLE      | t     | ref    | SSN,flight_index    | flight_index | 4       | airport_588.f.flight_id | 1    |             |
|    1 | SIMPLE      | p     | eq_ref | PRIMARY             | PRIMARY     | 4       | airport_588.t.SSN       | 1    |             |
+------+-------------+-------+--------+---------------------+-------------+---------+-------------------------+------+-------------+
3 rows in set (0.001 sec)

MariaDB [airport_588]>
```

Number of rows queried reduced from 10 to 1 in passengers table.

4. Create a query that uses a function in the WHERE clause and analyze its execution plan.

Query: To passengers whose first name starts with R(case insensitive)-using lower function

```
MariaDB [airport_588]> drop INDEX idx_f_name on passengers;
Query OK, 0 rows affected (0.043 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [airport_588]> explain SELECT *
    -> FROM passengers
    -> WHERE lower(first_name) LIKE 'r%';
+------+-------------+------------+------+---------------+------+---------+------+------+-------------+
| id   | select_type | table      | type | possible_keys | key  | key_len | ref  | rows | Extra       |
+------+-------------+------------+------+---------------+------+---------+------+------+-------------+
|    1 | SIMPLE      | passengers | ALL  | NULL          | NULL | NULL    | NULL | 13   | Using where |
+------+-------------+------------+------+---------------+------+---------+------+------+-------------+
1 row in set (0.001 sec)

MariaDB [airport_588]>
```

After creating index on first_name in passengers table

```
MariaDB [airport_588]> CREATE INDEX idx_f_name ON passengers(first_name);
Query OK, 0 rows affected (0.042 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [airport_588]> explain SELECT *
    -> FROM passengers
    -> WHERE LOWER(first_name) LIKE 'r%';
+------+-------------+------------+------+---------------+------+---------+------+------+-------------+
| id   | select_type | table      | type | possible_keys | key  | key_len | ref  | rows | Extra       |
+------+-------------+------------+------+---------------+------+---------+------+------+-------------+
|    1 | SIMPLE      | passengers | ALL  | NULL          | NULL | NULL    | NULL | 13   | Using where |
+------+-------------+------------+------+---------------+------+---------+------+------+-------------+
1 row in set (0.001 sec)
```

The index is not being used because the query uses a function on the indexed column: If the query uses a function on the indexed column (such as LOWER() or UPPER()), the database engine may not be able to use the index because it cannot apply the function to the index.

Rather, we can directly use ‘where’ condition

```
MariaDB [airport_588]> explain SELECT *
    -> FROM passengers
    -> WHERE first_name LIKE 'r%';
+------+-------------+------------+-------+---------------+------------+---------+------+------+-----------------------+
| id   | select_type | table      | type  | possible_keys | key        | key_len | ref  | rows | Extra                 |
+------+-------------+------------+-------+---------------+------------+---------+------+------+-----------------------+
|    1 | SIMPLE      | passengers | range | idx_f_name    | idx_f_name | 1022    | NULL | 3    | Using index condition |
+------+-------------+------------+-------+---------------+------------+---------+------+------+-----------------------+
1 row in set (0.001 sec)
```

5. Create a query that uses a dynamic SQL statement and analyze its execution plan.

Query: Display all passengers whose age is 23.

```
MariaDB [airport_588]> PREPARE statement FROM 'explain SELECT * FROM passengers WHERE age =?';
Query OK, 0 rows affected (0.001 sec)
Statement prepared

MariaDB [airport_588]> SET @pas_age = 23;
Query OK, 0 rows affected (0.000 sec)

MariaDB [airport_588]> EXECUTE statement USING @pas_age;
+------+-------------+------------+------+---------------+------+---------+------+------+-------------+
| id   | select_type | table      | type | possible_keys | key  | key_len | ref  | rows | Extra       |
+------+-------------+------------+------+---------------+------+---------+------+------+-------------+
|    1 | SIMPLE      | passengers | ALL  | NULL          | NULL | NULL    | NULL | 13   | Using where |
+------+-------------+------------+------+---------------+------+---------+------+------+-------------+
1 row in set (0.001 sec)
```

After creating index on age:

```
MariaDB [airport_588]> create index age_index on passengers(age);
Query OK, 0 rows affected (0.049 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [airport_588]> PREPARE statement FROM 'explain SELECT * FROM passengers WHERE age =?';
Query OK, 0 rows affected (0.001 sec)
Statement prepared

MariaDB [airport_588]> SET @pas_age = 23;
Query OK, 0 rows affected (0.001 sec)

MariaDB [airport_588]> EXECUTE statement USING @pas_age;
+------+-------------+------------+------+---------------+-----------+---------+-------+------+-------+
| id   | select_type | table      | type | possible_keys | key       | key_len | ref   | rows | Extra |
+------+-------------+------------+------+---------------+-----------+---------+-------+------+-------+
|    1 | SIMPLE      | passengers | ref  | age_index     | age_index | 5       | const | 2    |       |
+------+-------------+------------+------+---------------+-----------+---------+-------+------+-------+
1 row in set (0.001 sec)

MariaDB [airport_588]>
```

After creating index, the number of rows queried reduced from 13 to 2.

6. Create a query that involves complex data types (e.g., arrays) and analyze its execution plan.

->Create a new table with one column of array/json type.

```
MariaDB [airport_588]> CREATE TABLE new (
    ->   id INT PRIMARY KEY,
    ->   flight_number VARCHAR(50),
    ->   departure_time DATETIME,
    ->   arrival_time DATETIME,
    ->   passenger_counts JSON
    -> );
Query OK, 0 rows affected (0.012 sec)

MariaDB [airport_588]> INSERT INTO new (id, flight_number, departure_time, arrival_time, passenger_counts)
    -> VALUES
    ->   (1, 'ABC123', '2023-03-20 10:00:00', '2023-03-20 12:00:00', '{"economy": 100, "business": 20, "first_class": 10}'),
    ->   (2, 'XYZ456', '2023-03-20 15:00:00', '2023-03-20 18:00:00', '{"economy": 150, "business": 30, "first_class": 15}');
Query OK, 2 rows affected (0.008 sec)
Records: 2  Duplicates: 0  Warnings: 0

MariaDB [airport_588]> SELECT id, flight_number, JSON_EXTRACT(passenger_counts, '$.economy') as economy_count
    -> FROM new
    -> WHERE JSON_EXTRACT(passenger_counts, '$.economy') > 100;
+----+---------------+---------------+
| id | flight_number | economy_count |
+----+---------------+---------------+
|  2 | XYZ456        | 150           |
+----+---------------+---------------+
1 row in set (0.001 sec)
```

->Explain keyword

```
MariaDB [airport_588]> EXPLAIN SELECT id, flight_number, JSON_EXTRACT(passenger_counts, '$.economy') as economy_count
    -> FROM new
    -> WHERE JSON_EXTRACT(passenger_counts, '$.economy') > 100;
+------+-------------+-------+------+---------------+------+---------+------+------+-------------+
| id   | select_type | table | type | possible_keys | key  | key_len | ref  | rows | Extra       |
+------+-------------+-------+------+---------------+------+---------+------+------+-------------+
|    1 | SIMPLE      | new   | ALL  | NULL          | NULL | NULL    | NULL | 2    | Using where |
+------+-------------+-------+------+---------------+------+---------+------+------+-------------+
1 row in set (0.000 sec)
```

->After creating index on passenger_counts

```
MariaDB [airport_588]> CREATE INDEX idx_passenger_counts ON new(passenger_counts);
Query OK, 0 rows affected, 1 warning (0.017 sec)
Records: 0  Duplicates: 0  Warnings: 1

MariaDB [airport_588]> EXPLAIN SELECT id, flight_number, JSON_EXTRACT(passenger_counts, '$.economy') as economy_count
    -> FROM new
    -> WHERE JSON_EXTRACT(passenger_counts, '$.economy') > 100;
+------+-------------+-------+------+---------------+------+---------+------+------+-------------+
| id   | select_type | table | type | possible_keys | key  | key_len | ref  | rows | Extra       |
+------+-------------+-------+------+---------------+------+---------+------+------+-------------+
|    1 | SIMPLE      | new   | ALL  | NULL          | NULL | NULL    | NULL | 2    | Using where |
+------+-------------+-------+------+---------------+------+---------+------+------+-------------+
1 row in set (0.001 sec)
```

7. Compare the performance of different indexing strategies (e.g., B-tree, hash) on a large table and record the results.

->Creating a large table

```
MariaDB [airport_588]> CREATE TABLE my_table (
    ->   id INT NOT NULL PRIMARY KEY,
    ->   data VARCHAR(255)
    -> );
Query OK, 0 rows affected (0.009 sec)
```

->Inserting million values

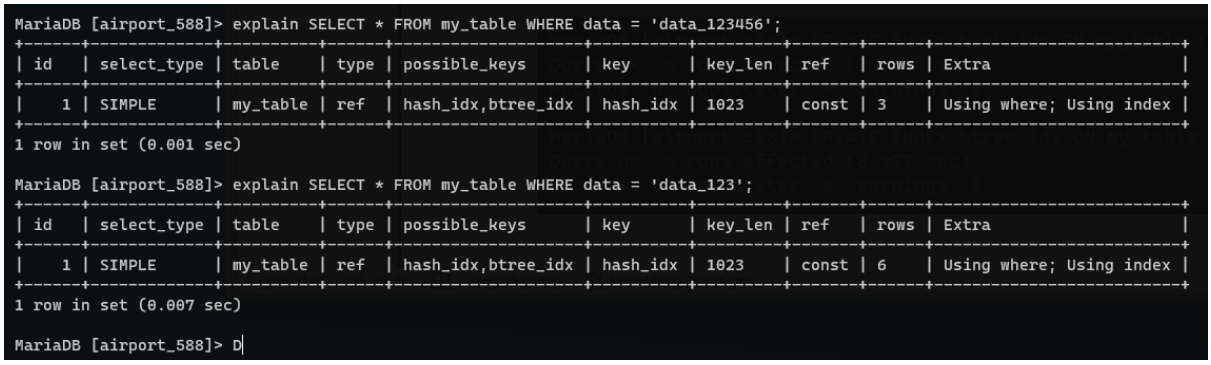
```
MariaDB [airport_588]> INSERT INTO my_table (id, data)
    -> SELECT
    ->   ROW_NUMBER() OVER () AS id,
    ->   CONCAT('data_', FLOOR(RAND() * 1000000)) AS data
    -> FROM
    ->   information_schema.columns c1,
    ->   information_schema.columns c2;
Query OK, 3892729 rows affected (39.942 sec)
Records: 3892729  Duplicates: 0  Warnings: 0
```

->Creating hash and btree index

```
MariaDB [airport_588]> CREATE INDEX hash_idx ON my_table (data) USING HASH;
Query OK, 0 rows affected (15.994 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [airport_588]> CREATE INDEX btree_idx ON my_table (data);
Query OK, 0 rows affected (8.977 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

->Selecting data with value data_123456



```
MariaDB [airport_588]> explain SELECT * FROM my_table WHERE data = 'data_123456';
+------+-------------+----------+------+--------------------+----------+---------+-------+------+--------------------------+
| id   | select_type | table    | type | possible_keys      | key      | key_len | ref   | rows | Extra                    |
+------+-------------+----------+------+--------------------+----------+---------+-------+------+--------------------------+
|    1 | SIMPLE      | my_table | ref  | hash_idx,btree_idx | hash_idx | 1023    | const | 3    | Using where; Using index |
+------+-------------+----------+------+--------------------+----------+---------+-------+------+--------------------------+
1 row in set (0.001 sec)

MariaDB [airport_588]> explain SELECT * FROM my_table WHERE data = 'data_123';
+------+-------------+----------+------+--------------------+----------+---------+-------+------+--------------------------+
| id   | select_type | table    | type | possible_keys      | key      | key_len | ref   | rows | Extra                    |
+------+-------------+----------+------+--------------------+----------+---------+-------+------+--------------------------+
|    1 | SIMPLE      | my_table | ref  | hash_idx,btree_idx | hash_idx | 1023    | const | 6    | Using where; Using index |
+------+-------------+----------+------+--------------------+----------+---------+-------+------+--------------------------+
1 row in set (0.007 sec)

MariaDB [airport_588]> D
```

Since hash index is used, hash index is more efficient.

8.Explore different database architectures (e.g., sharding, replication, distributed databases) and analyze their pros and cons in terms of query performance. Only explanation need to be provided.

There are several different database architectures that can be used to improve query performance, including sharding, replication, and distributed databases. Each architecture has its own advantages and disadvantages, which must be carefully evaluated based on the specific needs and requirements of your application.

1. **Horizontal/Sharding:** Sharding is the process of partitioning a large database into smaller, more manageable pieces called shards. Each shard can be stored on a separate server or cluster of servers, which can improve the performance of read and write operations by reducing the amount of data that needs to be processed.

Pros:

- Scalability: Sharding allows you to scale your database horizontally by adding more shards to your cluster.
- High availability: By replicating the shards across multiple servers, you can achieve high availability and prevent data loss in the event of a server failure.
- Performance: Sharding can improve the performance of read and write operations by reducing the amount of data that needs to be processed.

Cons:

- Complexity: Sharding adds complexity to your database architecture, making it more difficult to manage and maintain.

- Data consistency: Sharding can make it more difficult to maintain data consistency across different shards, especially during updates and deletions.
- Cost: Sharding requires multiple servers and hardware resources, which can be costly to maintain.

2. **Replication:** Replication is the process of creating multiple copies of your database on different servers. Each copy, or replica, can be used to serve read requests, while write requests are sent to a primary server.

Pros:

- High availability: By replicating your database across multiple servers, you can achieve high availability and prevent data loss in the event of a server failure.
- Scalability: Replication can improve the scalability of your database by allowing you to distribute read requests across multiple replicas.
- Performance: Replication can improve the performance of read requests by allowing you to distribute them across multiple replicas.

Cons:

- Complexity: Replication adds complexity to your database architecture, making it more difficult to manage and maintain.
- Data consistency: Replication can make it more difficult to maintain data consistency across different replicas, especially during updates and deletions.
- Write performance: Replication can decrease the performance of write requests, since they need to be sent to the primary server and then replicated to the other replicas.

3. **Distributed databases:** Distributed databases are databases that are spread across multiple servers or nodes. Each node can store a portion of the data, and requests can be routed to the appropriate node based on the location of the data.

Pros:

- Scalability: Distributed databases can be scaled horizontally by adding more nodes to the cluster.
- Performance: Distributed databases can improve the performance of read and write requests by distributing the load across multiple nodes.

- High availability: Distributed databases can achieve high availability by replicating the data across multiple nodes.

Cons:

- Complexity: Distributed databases are complex to manage and maintain, and require specialized skills and expertise.
- Data consistency: Distributed databases can make it more difficult to maintain data consistency across different nodes, especially during updates and deletions.
- Cost: Distributed databases require multiple servers and hardware resources, which can be costly to maintain.

In conclusion, there are several different database architectures that can be used to improve query performance, each with its own advantages and disadvantages. Sharding can improve scalability and performance, but adds complexity and can make it more difficult to maintain data consistency. Replication can improve high availability and scalability, but also adds complexity and can decrease the performance of write requests. Distributed databases can improve scalability and performance, but also require specialized expertise and can be costly to maintain. The choice of architecture ultimately depends on the specific needs and requirements of your application.