AIM: To create and modify table attributes using DDL commands.

* CREATE**:** 
  + Description: To create database or objects in the database
  + Syntax:

CREATE database\_name;

CREATE TABLE table\_name(

Col\_name1 datatype(),

Col\_name2 datatype(),…

Col\_namen datatype());

* DESC**:**
  + Description: To describe the tables
  + Syntax: DESC table\_name;
* ALTER**:**
  + Description: Alters the structure of the database
  + Syntax: ALTER TABLE table\_name

ADD col\_name datatype();

* DROP**:**
  + Description: Delete objects from the database
  + Syntax: DROP table\_name;
* RENAME**:**
  + Description: Renames an object
  + Syntax: RENAME COLUMN table\_name to new\_table\_name
* CHECK**:**
  + Description: It is used to limit the value range that can be placed in a column
  + Syntax: CHECK (column\_name condition);
* MODIFY**:**
  + Description: It is used to modify a existing column in a table.
  + Syntax: MODIFY column\_name new\_data\_type;
* TRUNCATE**:**
  + Description: It is used to deletes the data inside a table, but not the table itself.
  + Syntax: TRUNCATE table table\_name;

1. To create a table.

QUERIES:

CREATE database 2020it0097;

use 2020it0097;

OUTPUT:

[Start] Executing MySQL query...

[Done] Finished MySQL query.



2. Create tables with the necessary fields

QUERIES:

use 2020it0097;

create table airport(

name varchar(20),

city varchar (20),

state varchar(20),

elevation int,

heliport BOOL,

code varchar(3)

);

CREATE table Route(

    rnumber INT,

    rdesc VARCHAR(20),

    ffrom VARCHAR(20),

    tto VARCHAR(20),

    distance INT

);

create table Flights(

    flight\_no VARCHAR(20),

    aircraft\_id INT,

    departs time,

    arrives time,

    fare\_type INT,

    legno INT,

    traveldate DATE,

    route INT,

    status VARCHAR(20)

);

CREATE table Aircraft(

    aid INT,

    aname VARCHAR(20),

    cruising\_range INT,

    airlines VARCHAR(20),

    capacity INT

);

CREATE TABLE Airline(

    name VARCHAR(20),

    type VARCHAR(20),

    commencement INT,

    hub INT

);

CREATE table certified(

    serial\_no INT PRIMARY KEY AUTO\_INCREMENT,

    eid INT,

    aid INT,

    validity DATE,

    country VARCHAR(20)

);

CREATE table pilots(

    eid INT,

    ename VARCHAR(20),

    salary INT,

    age INT,

    rank VARCHAR(20)

);

CREATE table Passenger(

    Pid VARCHAR(20),

    name VARCHAR(20),

    address VARCHAR(20),

    telno VARCHAR(20),

    age INT,

    email\_id varchar(20)

);

CREATE table ticket(

    ticket\_no INT,

    ticket\_date VARCHAR(20),

    passenger\_id INT,

    booking\_date DATE,

    fare\_type VARCHAR(20),

    total\_fare decimal,

    payment\_mode VARCHAR(20),

    status VARCHAR(20)

);

CREATE table itinerary(

    Ticket\_no INT,

    flight\_no INT,

    leg\_no INT,

    flight\_date DATE,

    fare\_type VARCHAR(20)

);

CREATE table leg(

    flight\_no INT,

    leg1 VARCHAR(20),

    leg2 VARCHAR(20)

);

CREATE TABLE fares(

    faretype INT,

    fdescription VARCHAR(20),

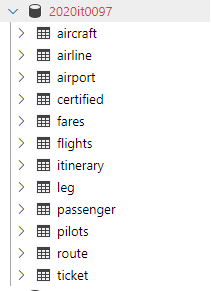
    base\_fare decimal,

    charges decimal,

    cost decimal

);

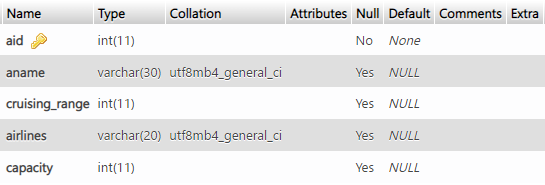
OUTPUT:

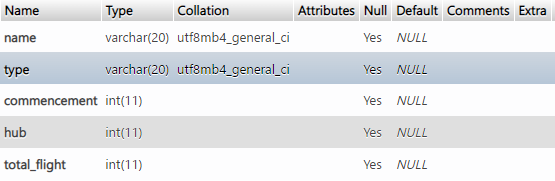


3. Describe the tables

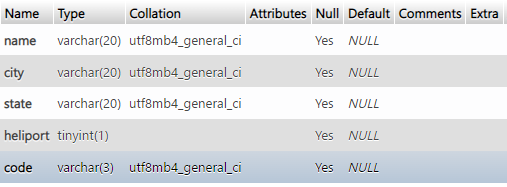
QUERIES: DESC aircraft;

OUTPUT:

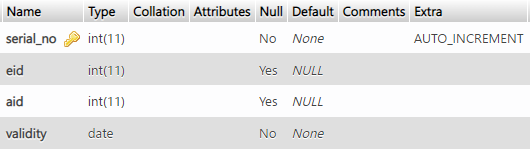


DESC airline;

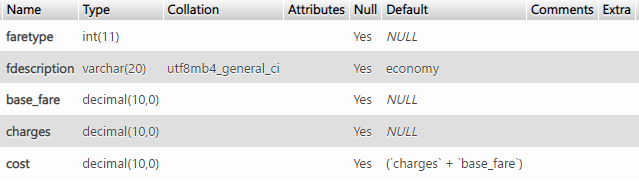
DESC airport;



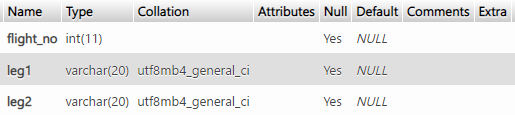
DESC certified;



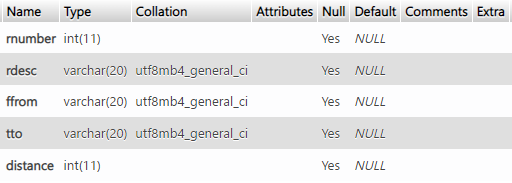
DESC fares;



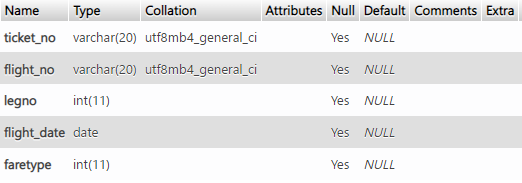
DESC leg;



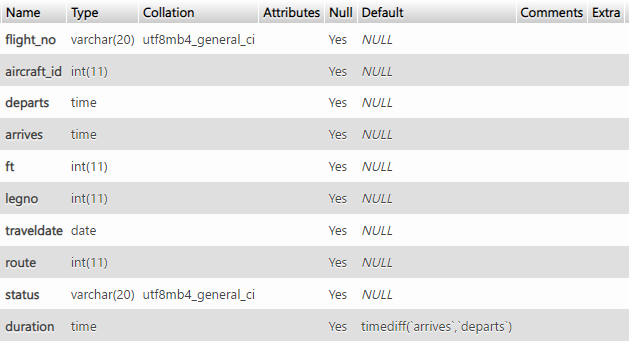
DESC route;



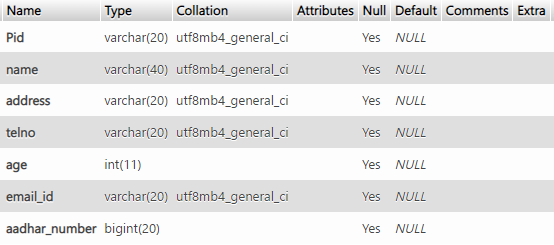
DESC itinerary;



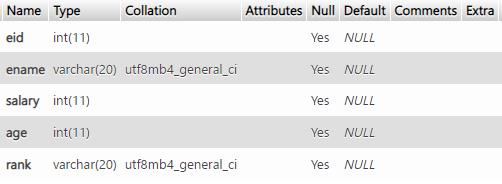
DESC flights;



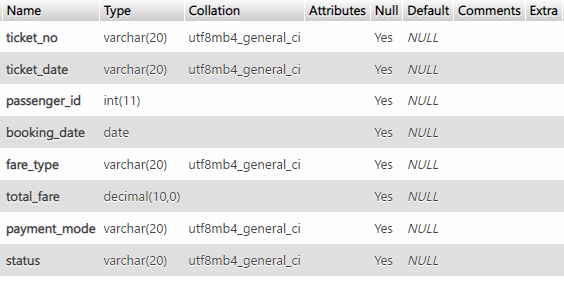
DESC passenger;



DESC pilots;



DESC ticket;



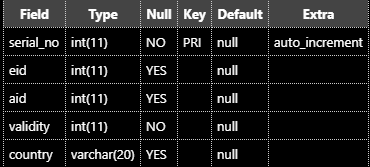
4. Alter the table certified by adding not null constraint to validity

QUERIES:

ALTER table certified

MODIFY validity INT NOT NULL;

OUTPUT:



5. Add column “total flights” to airline table

QUERIES: ALTER table airline ADD total\_flight INT;

OUTPUT: The following attribute is being added to airline table.



6. Alter the table aircraft to add a primary key constraint on the column aid.

QUERIES:

alter table aircraft

modify aid INT primary key;

OUTPUT: aid is now set to primary key



7. Alter the table airport adding unique constraints to the elevation column

QUERIES:

alter table airport

modify elevation INT UNIQUE;

OUTPUT: KEY is set to unique



8. Alter the table airport drop the column elevation

QUERIES:

alter table airport

drop elevation;

OUTPUT: Before and after execution

|  |  |
| --- | --- |
|  |  |

9. Alter the table flight, rename the column fair type to ft.

QUERIES:

ALTER TABLE `flights` CHANGE `fare\_type` `ft` INT;

OUTPUT:





10. Alter table passengers add column aadhar numbers

QUERIES:

ALTER TABLE passenger

ADD aadhar\_number BIGINT(20);

OUTPUT:



11. Modify the table fares to check whether the fare type is between 1 and 4

QUERIES:

ALTER TABLE fares

ADD CHECK (faretype BETWEEN 1 AND 4);

OUTPUT:

[Start] Executing MySQL query...

[Done] Finished MySQL query.

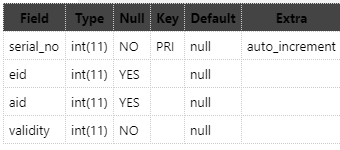
12. Alter table certified Delete the country attribute

QUERIES:

ALTER TABLE certified

DROP country;

OUTPUT:



13. Change the size of the attribute aname to 30 in the aircraft table

QUERIES:

ALTER TABLE aircraft

MODIFY aname VARCHAR(30);

OUTPUT:



14. Alter table fares add default constraints on FDescription as “economy”

QUERIES:

ALTER TABLE fares

ALTER fdescription SET DEFAULT "economy";

OUTPUT:



15. Alter the table employee to add check constraints “agecons” on age, do not accept values <17.

QUERIES:

ALTER TABLE pilots

ADD CONSTRAINT agecons check (age > 17);

16. Alter table flights add check constraint “legcons” on legno , leg no<=2;

QUERIES:

ALTER table flights

ADD CONSTRAINT legcons CHECK(legno>1);

17. Drop the constraints legcons

QUERIES:

ALTER table flights

DROP CONSTRAINT legcons ;

OUTPUT:

[Start] Executing MySQL query...

[Done] Finished MySQL query.

18. Modify the ticket table Ticketno to start with ‘T’

QUERIES:

ALTER table ticket

ADD CHECK (ticket\_no LIKE "T%") ;

19. Create a table booking from Ticket with ticket date and booking date attribute.

QUERIES:

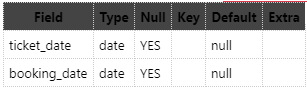
CREATE TABLE booking\_from\_ticket (

    ticket\_date DATE,

    booking\_date DATE

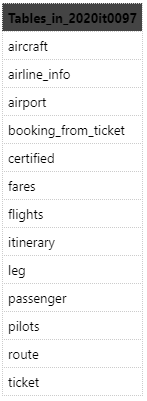
);

OUTPUT:



20. Rename table airline as airline\_info

OUTPUT:



21. Create a ticket info table from the ticket table with any three attributes.

QUERIES:

CREATE TABLE ticket\_info(

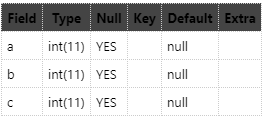
    a INT,

    b INT,

    c INT

);

OUTPUT:



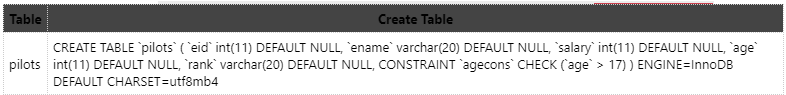
22. Truncate the ticket info table.

QUERIES: TRUNCATE TABLE ticket\_info;

23. Display all the constraints of a table.

QUERIES: show CREATE TABLE pilots;

OUTPUT:



RESULT: Execution of DDL commands with mySQL is successful and the necessary database is created successful.

AIM: To execute Simple Queries and retrieve necessary data.

DEFINITION FOR THE TABLES:

INSERT INTO flights

VALUES

    (1, 8, '1:46:36', '7:22:22', 4, 1, '21-05-08', 3, 'airing'),

    (2, 9, '4:46:36', '14:33:1', 3, 1, '21-04-01', 4, 'airing'),

    (3, 10, '6:44:36', '21:03', 2, 2, '21-12-01', 4, 'airing'),

    (4, 11, '13:44:52', '19:44:36', 2, 1, '21-04-07', 1, 'delayed'),

    (5, 12, '7:35:1', '14:33:1', 1, 1, '21-06-06', 2, 'delayed');

INSERT INTO aircraft

VALUES

    (8, 'boeing 747', 37000, 'Diva', 370),

    (9, 'boeing 717', 30000, 'Taizai', 700),

    (10, 'airbus a380', 50000, 'Squiler', 800),

    (11, 'douglas dc', 15000, 'Kuuhaku', 214),

    (12, 'boeing 70', 70000, 'Chigau', 236);

INSERT INTO airline\_info

VALUES

    ('Diva', 'schedule', 3, 4, 3),

    ('Taizai', 'charter', 2, 7, 4),

    ('Squiler', 'schedule', 1, 1, 7),

    ('Kuuhaku', 'regional', 2, 1, 2),

    ('Chigau', 'intl', 3, 3, 10);

INSERT INTO pilots

VALUES

    (1, 'Hartmann', 800000, 37, 'captain'),

    (2, 'Doolittle', 700000, 40, 'senior first officer'),

    (3, 'Lindberg', 370000, 56, 'second officer'),

    (4, 'Amelia Earheart', 500000, 23, 'first officer'),

    (5, 'Shankar', 220000, 22, 'trainee');

INSERT INTO route

VALUES

    (10, 'great circle', 'japan', 'italy', 9723),

    (4, 'jetstream', 'new york', 'madagascar', 14028),

    (3, 'ocean track', 'chennai', 'carribean', 15527),

    (2, 'domestic', 'mumbai', 'chennai', 1336),

    (1, 'regional', 'chennai', 'mumbai', 1340),

    (7, 'intercontinent', 'singapore', 'philippines', 2355),

    (6, 'intracontinent', 'nigeria', 'london', 4745);

INSERT INTO ticket

VALUES

    ('T4be', '21-06-30', '72c', '21-06-27', 1, 3400, 'cash', 'confirmed'),

    ('T37a', '21-08-17', '41db', '21-08-10', 2, 20000, 'neft', 'confirmed'),

    ('T05u', '21-03-14', '35pcm', '21-03-10', 3, 27000, 'upi', 'confirmed'),

    ('T24k', '21-02-10', '192kekz', '21-02-03', 4, 7000, 'cash', 'confirmed'),

    ('Tdb19', '21-05-07', '96khz', '21-05-04', 4, 15000, 'credit', 'pending'),

    ('T4270', '21-04-24', '32bit', '21-04-23', 3, 34000, 'debit', 'pending');

INSERT INTO passenger

VALUES

    ('72c', 'Satoru', 'chennai, tamil nadu', 73201, 15, 'satoru@mail', 54789541254),

    ('41db', 'Watanabe saki', 'delhi, delhi', 43127, 16, 'watanabe@mail', 47878421314),

    ('35pcm', 'William James Moriarty', 'bombay, maharastra', 57410, 26, null , 57841246956),

    ('192kekz', 'Nagisa Furukawa', 'imanity, elkia', 32180, 21, 'furukawa@mail', 12478569548),

    ('96khz', 'Okazaki Tomoya', 'akihabara, tokyo', 61357, 20, 'tomoya@mail', 78452159684),

    ('32bit', 'Batra', 'britania, camolet', 43109, 76, null , 57848756245);

alter table fares

alter cost set DEFAULT (charges + base\_fare);

INSERT  INTO fares (faretype, fdescription, base\_fare, charges)

VALUES

    (1,'economy', 1000, 5000),

    (2,'economy plus', 1700, 6000),

    (3,'business', 3000, 25000),

    (4,'first class', 5000, 28000);

INSERT INTO certified (eid, aid, validity)

VALUES

    (1, 8, '25-01-01'),

    (2, 9, '33-01-01'),

    (3, 10, '31-01-01'),

    (4, 11, '29-01-01'),

    (5, 12, '41-01-01');

INSERT INTO airport VALUES

    ('curtana', 'tokyo', 'japan', 1, 700),

    ('mido', 'new york', 'new york', 1, 024),

    ('davinci', 'chennai', 'tamilnadu', 1, 023),

    ('alioth', 'mumbai', 'maharastra', 0, 100),

    ('violet', 'britania', 'camolet', 1, 747);

INSERT INTO itinerary VALUES

    ('T4be', 1, 1, '21-05-08', 1),

    ('T37a', 2, 1, '21-04-01', 2),

    ('T05u', 3, 2, '21-12-01', 3),

    ('T24k', 4, 1, '21-04-07', 4),

    ('Tdb19', 5, 1, '21-06-06', 4),

    ('T4e70', 1, 1, '21-05-08', 3);

1. Retrieve the names of all trainee pilots

QUERIES:

SELECT ename

FROM pilots

where rank = "trainee";

OUTPUT:



2. Select the flights that have only one stop.

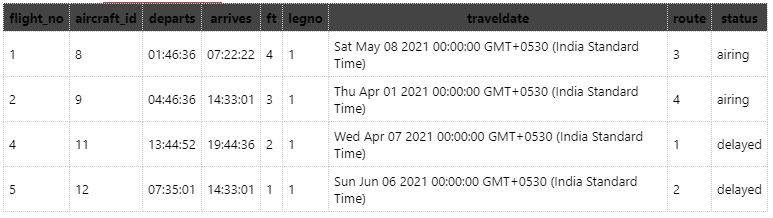
QUERIES:

SELECT \*

FROM flights

WHERE legno=1;

OUTPUT:



3. Find the flights with same arrival time

QUERIES:

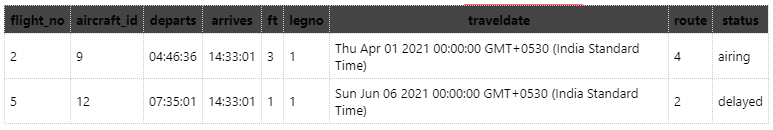
select \* from flights where arrives in (

    select arrives from flights

    group by arrives having count(\*) > 1

);

OUTPUT:



4. Find the duration of all flights

QUERIES: ALTER TABLE flights ADD `duration` TIME DEFAULT TIMEDIFF(arrives, departs);

OUTPUT:



5. Find the airlines that offer flight service

QUERIES:

SELECT DISTINT airlines from aircraft, flights WHERE aid = aircraft\_id AND status NOT LIKE 'cancelled';

OUTPUT:

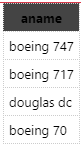


6. Select the flights that are shorter than 12 hours.

QUERIES:

select aname from flights, aircraft where aircraft\_id = aid AND duration < '12:00:00';

OUTPUT:



7. Update flights table change the status to `cancelled` for flights in route 3 and status to delayed for flights in route 4.

QUERIES:

UPDATE flights SET status = 'cancelled' WHERE route = 3;

UPDATE flights SET status = 'delayed' WHERE route = 4;

OUTPUT:



8. Extend the validity of ‘shankar’ by 1 year;

QUERIES:

UPDATE

    certified

SET

    validity = DATE\_ADD(validity, INTERVAL 1 YEAR)

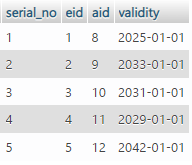
WHERE validity IN (

    SELECT validity

    FROM certified c, pilots p

    WHERE c.eid=p.eid AND p.ename = 'Shankar');

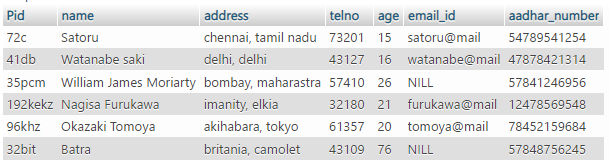
OUTPUT:

****

9. Update the value NILL for passengers with no mail id.

QUERIES: UPDATE passenger SET email\_id = 'NILL' WHERE email\_id IS NULL;

OUTPUT:



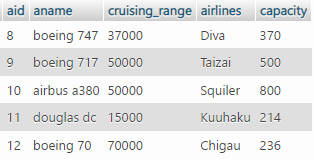
10. Change the cruising range to 50000 and capacity to 500 for boeing 717

QUERIES:

UPDATE aircraft SET cruising\_range = 50000, capacity = 500

WHERE aname = 'boeing 717';

OUTPUT:

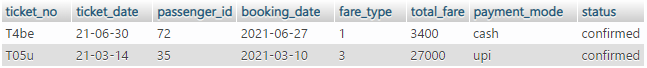


DESCRIPTION: 11. Print the ticket information of all confirmed tickets between March and June 2021

QUERIES:

SELECT \* FROM `ticket` WHERE booking\_date BETWEEN '2021-03-01' AND '2021-06-30' AND status = 'confirmed';

OUTPUT:



DESCRIPTION: 12. Retrieve the name of routes starting with m or c

QUERIES:

SELECT rdesc FROM route WHERE ffrom LIKE 'm%' OR ffrom LIKE 'c%';

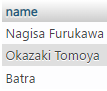
OUTPUT:



DESCRIPTION: 13. Retrieve the names of Passengers whose name not start with m, s, or w

QUERIES: SELECT name FROM passenger WHERE name NOT LIKE 'S%' AND name NOT LIKE 'W%' AND name NOT LIKE 'M%';

OUTPUT:



DESCRIPTION: 14. Print the names of all international airports

QUERIES: SELECT name FROM `airport`;

OUTPUT:



15. Print the names of Air force stations

QUERIES: SELECT name FROM `airport` WHERE heliport;

OUTPUT:



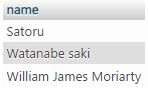
16. Find the name of all passenger who stay in the city Chennai, Mumbai, or Delhi.

QUERIES:

SELECT name FROM `passenger`

WHERE address LIKE 'chennai%' OR address LIKE 'delhi%' OR address LIKE 'bombay%';

OUTPUT:



17. Name of all city ending with t or e

QUERIES: SELECT city FROM `airport` WHERE city LIKE '%a' OR city LIKE '%k';

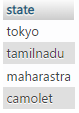
OUTPUT:



18. Names of state that ends with t in it

QUERIES: SELECT state FROM `airport` WHERE state LIKE '%t%';

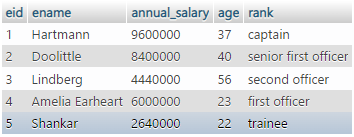
OUTPUT:



19. Display the annual salary of all Pilots

QUERIES: SELECT eid, ename, salary\*12 AS annual\_salary, age, rank FROM `pilots`;

OUTPUT:



20. Increase fares of first class by 10%

QUERIES: UPDATE fares SET base\_fare = base\_fare + base\_fare\*.1 WHERE fdescription = 'first class';

OUTPUT:



21. Find the passengers who belong to Chennai or aged less than 17

QUERIES: SELECT name FROM `passenger` WHERE address LIKE 'chennai%' OR age < 17;

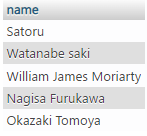
OUTPUT:



22. Find those passengers who either belong to Chennai or not aged > 75

QUERIES: SELECT name FROM `passenger` WHERE age < 75 OR address LIKE 'chennai%';

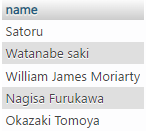
OUTPUT:



23. Passenger neither belong to camolet nor have an mail id

QUERIES: SELECT name FROM passenger WHERE NOT(email\_id is NULL OR address LIKE '%camolet');

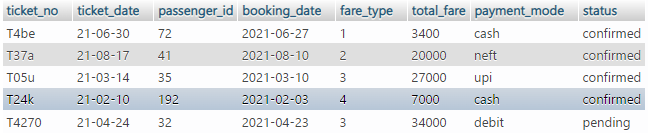
OUTPUT:



24. Find information about all tickets excluding paid by credit and status pending

QUERIES: SELECT \* FROM `ticket` WHERE NOT (status='pending' AND payment\_mode='credit');

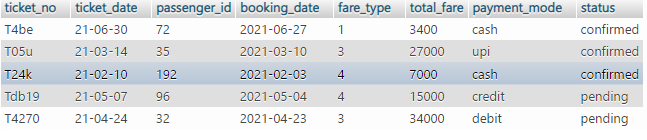
OUTPUT:



25. Find details of all the tickets where total amount < 20000 or excluding the combination of ticket dates greater or equal to 2021-08-17.

QUERIES: SELECT \* FROM `ticket` WHERE total\_fare < 20000 OR NOT (ticket\_date >='21-08-17');

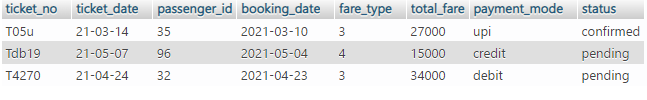
OUTPUT:



26. Exclude combination of ticket date equal to '2021-08-17' or total amount less than 10000.

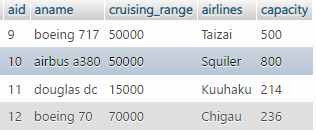
QUERIES: SELECT \* FROM `ticket` WHERE NOT (total\_fare < 10000 OR ticket\_date ='21-08-17');

OUTPUT:

27. Find all aircraft subject to following conditions. Exclude combination of aircrafts with cruising range equal to 37000 or number of seats higher than 300 and a Boeing aircraft

QUERIES: SELECT \* FROM `aircraft` WHERE NOT cruising\_range = 37000 OR NOT (capacity > 300 AND aname LIKE 'boeing%');

OUTPUT:



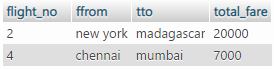
28. Display flight id, from location, to location and price of flights whose departure is in the month of April

QUERIES:

SELECT `itinerary`.`flight\_no`, `route`.`ffrom`, `route`.`tto`, `ticket`.`total\_fare` FROM itinerary JOIN ticket ON `itinerary`.`ticket\_no` = `ticket`.`ticket\_no` JOIN flights ON `itinerary`.`flight\_no`= `flights`.`flight\_no` JOIN route ON `flights`.`route`=`route`.`rnumber`

WHERE `itinerary`.`flight\_date` BETWEEN '2021-04-01' AND '2021-04-30';

OUTPUT:



29. Delete route 10

COMMAND NAME: DELETE FROM route WHERE rnumber = 10;

OUTPUT:



30. Copy the table route to rtemp and delete top 5 rows in rtemp

QUERIES:

CREATE TABLE rtemp AS SELECT \* FROM route;

DELETE FROM rtemp LIMIT 5;

OUTPUT:



RESULT: Using DML commands, the necessary data has been extracted by executing SQL queries.

AIM: To make use of nested and sub-queries using aggregate function

1. Select on the flights of type schedule.

QUERIES:

SELECT SUM(total\_flight) as no\_of\_flights

FROM `airline\_info`

WHERE type = "schedule";

OUTPUT:



2. How many flights are operated on each route.

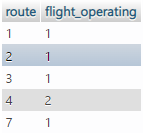
QUERIES:

SELECT route, COUNT(\*) as flight\_operating

FROM flights

GROUP BY route;

OUTPUT:



3. Find all the tickets booked by Tomoya

QUERIES:

SELECT COUNT(\*) AS no\_of\_tickets\_booked

FROM `ticket` t, `passenger` p

WHERE t.passenger\_id = p.Pid AND

p.name = 'Okazaki Tomoya';

OUTPUT:



4. Display id of passenger who booked maximum and minimum number of tickets.

QUERIES:

(

    SELECT t.passenger\_id

    FROM `ticket` t

    GROUP BY t.passenger\_id

    ORDER BY COUNT(t.passenger\_id) DESC LIMIT 1

)

UNION(

    SELECT t.passenger\_id

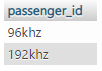
    FROM `ticket` t

    GROUP BY t.passenger\_id

    ORDER BY COUNT(t.passenger\_id) LIMIT 1

    );

OUTPUT:



5. How many flights went to chennai?

QUERIES:

SELECT COUNT(flight\_no)

FROM flights f

WHERE f.route IN

(SELECT rnumber FROM route WHERE tto = "chennai");

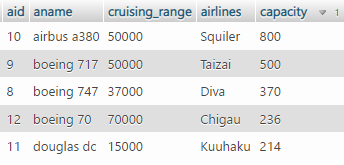
OUTPUT:



6. Order by aircraft table by number of seats, in descending order.

QUERIES: SELECT \* FROM aircraft ORDER BY capacity DESC;

OUTPUT:



7. Were there more flights to Mumbai in April 2013 or July 2013?

QUERIES:

SELECT traveldate as in\_month, COUNT(f.traveldate) as no\_of\_flights

FROM flights f, route r

WHERE f.route = r.rnumber AND r.tto = 'mumbai' AND

((f.traveldate BETWEEN '2021-04-01' AND '2021-04-30') OR (f.traveldate BETWEEN '2021-06-01' AND '2021-06-31'));

OUTPUT:



8. Show all of the Alaska Airlines flights between June 1st, 2013 and June 3rd, 2013

QUERIES:

SELECT flight\_no

FROM flights

WHERE traveldate BETWEEN '2021-04-01' AND

'2021-04-08' AND

aircraft\_id = (SELECT aid FROM aircraft WHERE airlines = "Taizai");

OUTPUT:



9. Display number of flights between 6.00 PM and 11.59 PM on all days.

QUERIES: SELECT COUNT(flight\_no) FROM flights WHERE arrives BETWEEN '18:00:00' AND '23:59:00';

OUTPUT:



10. Write a query to find the number of passengers travelled on each flight.

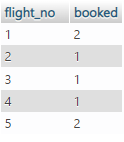
QUERIES:

SELECT flight\_no, COUNT(flight\_no) as booked

  FROM itinerary

  GROUP BY flight\_no;

OUTPUT:



11. Write a query to find the number of tickets as “No\_of\_Tickets” booked by chennai customers.

QUERIES:

SELECT COUNT(ticket\_no) as Number\_of\_tickets

FROM ticket

WHERE passenger\_id = (SELECT Pid FROM passenger WHERE address LIKE "chennai%");

OUTPUT:



12. Write a query to find the number of flights flying from each location.

QUERIES:

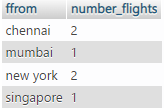
SELECT ffrom, COUNT(flight\_no) AS number\_flights

FROM flights , route

WHERE flights.route = route.rnumber

GROUP BY ffrom;

OUTPUT:



RESULT: Thus nested and subqueries using aggregate functions has been executed.

AIM: To study about nested and subqueries using ALL, ANY, IN, exists not exists, some

* ANY: returns TRUE if ANY of the subquery values meet the condition.
* ALL: returns TRUE if ALL of the subquery values meet the condition
* EXISTS: returns TRUE if the subquery returns one or more records.:
* SOME: evaluates the condition between the outer and inner tables and evaluates to true if the final result returns any one row

1. Find the Names and Addresses of passengers flying from chennai to mumbai and from chennai to mumbai or both.

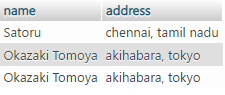
QUERIES:

SELECT p.name, p.address

FROM passenger p, route r, flights f, itinerary i, ticket t

WHERE f.route = r.rnumber AND f.flight\_no = i.flight\_no AND i.ticket\_no = t.ticket\_no AND t.passenger\_id = p.Pid AND ((ffrom = 'mumbai' OR ffrom = 'chennai')OR (ffrom = 'chennai' OR ffrom = 'mumbai'));

OUTPUT:



2. Find the flights which had at least 30% booking of its capacity.

QUERIES:

SELECT a.aname, result.booked

FROM

( SELECT flight\_no, COUNT(flight\_no)\*125 as booked

  FROM itinerary

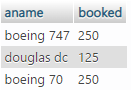
  GROUP BY flight\_no

  HAVING booked > 1

) AS result, flights f, aircraft a

WHERE result.flight\_no = f.flight\_no AND f.aircraft\_id = a.aid AND result.booked > a.capacity\*.3;

OUTPUT:



3. Find the names of all airlines that ever flew more than 2 flights in one day.

QUERIES:

SELECT al.name

FROM flights f, aircraft ac, airline\_info al

WHERE f.aircraft\_id = ac.aid AND ac.airlines = al.name

GROUP BY f.aircraft\_id HAVING COUNT(f.aircraft\_id) > 1 ;

OUTPUT:



4. Display number of flights between 6.00 AM and 6.00 PM from Chennai.

QUERIES:

SELECT COUNT(f.flight\_no) as no\_of\_flights

FROM flights f, route r

WHERE f.route = r.rnumber AND r.ffrom='chennai' AND (f.departs BETWEEN '06:00:00' AND '18:00:00');

OUTPUT:



5. Find the names of aircraft such that all pilots certified to operate them have salaries more than Rs 220000

QUERIES:

SELECT result.aname

FROM pilots p JOIN (

    SELECT ac.aname, c.eid

    FROM aircraft ac INNER JOIN certified c ON ac.aid = c.aid

) as result ON result.eid = p.eid

WHERE p.salary > 220000;

OUTPUT:



6. Find the number of flights operated from singapore, nigeria and mumbai.

QUERIES:

SELECT COUNT(f.flight\_no)

FROM flights f

WHERE f.route IN (

    SELECT `route`.`rnumber`

    FROM route

    WHERE ffrom = 'mumbai' OR ffrom = 'singapore' or ffrom = 'nigeria'

);

OUTPUT:



7. Find the pilots aged above 30 with maximum certification.

QUERIES: SELECT ename FROM pilots WHERE age>30 AND rank="captain";

OUTPUT:



8. How many tickets are booked by passenger Tomoya.

QUERIES:

SELECT COUNT(ticket\_no)

FROM ticket

WHERE ticket\_no IN (

    SELECT t.ticket\_no

    FROM ticket t, passenger p

    WHERE t.passenger\_id = p.Pid AND p.name = 'Okazaki Tomoya'

);

OUTPUT:



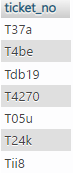
9. Date wise print the number of tickets.

QUERIES:

SELECT ticket\_no FROM `ticket`

ORDER BY ticket\_date DESC;

OUTPUT:



10. Find the number of unoccupied seats in each flight.

QUERIES:

SELECT (ac.capacity - result.booked) as unoccupied\_seats

FROM

( SELECT flight\_no, COUNT(flight\_no) as booked

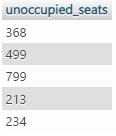
  FROM itinerary

  GROUP BY flight\_no

) AS result, aircraft ac, flights f

WHERE result. flight\_no = f.flight\_no AND f.aircraft\_id = ac.aid;

OUTPUT:



AIM: To study about SET operations and JOINS

* SET OPERATIONS: SET operators are special type of operators which are used to combine the result of two queries.
* JOIN: A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

1. Display the number of passengers in each flight.

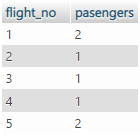
QUERIES:

SELECT flight\_no, COUNT(flight\_no) as booked

  FROM itinerary

  GROUP BY flight\_no

OUTPUT:



2. Find the average salary for pilots certified for each aircraft.

QUERIES: select AVG(pilots.salary) from pilots inner join certified where pilots.eid = certified.eid;

OUTPUT:



3. Find the Names and Addresses of passengers flying from chennai to mumbai and from chennai to mumbai or both.

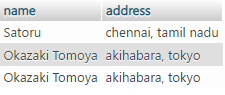
QUERIES:

SELECT p.name, p.address

FROM passenger p, route r, flights f, itinerary i, ticket t

WHERE f.route = r.rnumber AND f.flight\_no = i.flight\_no AND i.ticket\_no = t.ticket\_no AND t.passenger\_id = p.Pid AND ((ffrom = 'mumbai' OR ffrom = 'chennai')OR (ffrom = 'chennai' OR ffrom = 'mumbai'));

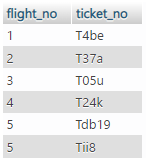
OUTPUT:



4. List all Flight No's with tickets issued against them.

QUERIES: select itinerary.flight\_no, ticket.ticket\_no from itinerary inner join ticket on itinerary.ticket\_no = ticket.ticket\_no;

OUTPUT:



5. List the Flight No's on the itinerary for Ticket T24k

QUERIES:

select itinerary.flight\_no, ticket.ticket\_no from itinerary inner join ticket on itinerary.ticket\_no = ticket.ticket\_no

WHERE ticket.ticket\_no = 'T24k';

OUTPUT:



6. List the description and seating capacity of the aircraft allocated to the flights on TicketNo T24k.

QUERIES:

SELECT aircraft.capacity, route.rdesc

FROM aircraft, flights, route

WHERE flights.flight\_no IN

(select itinerary.flight\_no from itinerary inner join ticket on itinerary.ticket\_no = ticket.ticket\_no

WHERE ticket.ticket\_no = 'T24k') AND flights.route = route.rnumber AND aircraft.aid = flights.aircraft\_id;

OUTPUT:



7. List all cities that can be reached from chennai though a direct flight nor with one stop

QUERIES:

(

    SELECT r.tto

    FROM flights f, route r

    WHERE f.route = r.rnumber AND f.legno = 1 AND ffrom = 'chennai'

) UNION

(

    SELECT r.tto

    FROM leg l INNER JOIN flights f ON f.flight\_no = l.flight\_no INNER JOIN route r ON f.route = r.rnumber

    WHERE ffrom = 'chennai'

);

OUTPUT:



8. How many flights did flight 8 make in a day

QUERIES:

SELECT result.count, result.dates

FROM

(SELECT COUNT(aircraft\_id) as count, traveldate as dates, aircraft\_id

FROM flights

GROUP BY traveldate

HAVING COUNT(traveldate) > 1)as result

WHERE `result`.`aircraft\_id` = '8';

OUTPUT:



9. Find the distinct flight numbers of all flights from Chennai to Delhi by Alaska Airlines Inc.

QUERIES:

SELECT DISTINCT(f.flight\_no)

FROM flights f, route r

WHERE f.aircraft\_id IN

(SELECT aid

FROM aircraft

WHERE airlines='Squiler')

AND f.route = r.rnumber

AND r.ffrom = 'new york' AND tto = 'madagascar';

OUTPUT:



10. Find the number of passengers travelled on 1/11/2021 by flight 8 print the total cost of the flight.

QUERIES:

SELECT COUNT(t.ticket\_no) as no\_of\_passengers, SUM(t.total\_fare)

FROM ticket t

WHERE t.ticket\_no IN (

    SELECT i.ticket\_no

    FROM itinerary i

    WHERE i.flight\_no IN (

        SELECT f.flight\_no

        FROM flights f

        WHERE f.traveldate = '2021-05-08'

    )

);

OUTPUT:



RESULT: The given problems were successfully solved by executing suitable set operation and various joins.

AIM: To incorporate usage of VIEWS in SQL queries

DESCRIPTION:

Views in SQL are kind of virtual tables. A view also has rows and columns as they are in a real table in the database. We can create a view by selecting fields from one or more tables present in the database. A View can either have all the rows of a table or specific rows based on certain condition.

SYNTAX:

CREATE VIEW view\_name AS

SELECT column1, column2.....

FROM table\_name

WHERE condition;

QUERIES:

1. Create a view for those subscribers who belong to the city 'New York'.

create view subscribers\_of\_NY as select id,email\_id from subscriber where city='NY';

select \* from subscribers\_of\_NY;

2. Create a view to count the number of unique subscriber’s orders by subscription date.

create view count as select subscription\_date,count(distinct(id)) as count from subscriber group by subscription\_date order by subscription\_date;

select \* from count;

3. Create a view to count the number of videos of each channel.

create view video\_count as select channel\_id,count(distinct(video\_id)) from video group by channel\_id;

select \* from video\_count;

4. Create a view to find the most popular videos.

create view popularv as select video\_id,no\_of\_views from video order by no\_of\_views desc;

select \* from popularv;

5. Find the average number of subscribers for each channel.

create view count as select subscription\_date,count(distinct(id)) as count from subscriber group by subscription\_date order by subscription\_date;

select \* from count;

6. Create a view to fetch the top 10 videos having more number of views, along with the channel details.

create view popular as select video\_id,no\_of\_views from video order by no\_of\_views desc limit 5;

select \* from popular;

7. Create a view to display video details sorted by name and duration.

create view ordered as Select \* from video order by name,duration asc;

select \* from ordered;

8. Create a view to display date wise the number of videos published.

create view datewise as select published\_date,count(video\_id) from video group by published\_date;

select \* from datewise;

9. Create a view to find customer wise the number of channels subscribed.

create view subscription as select channel\_id,count(id) as subscribers from subscriber group by channel\_id;

select \* from subscription;

10. Create a view to display video details ordered by duration.

create view ordered\_d as Select \* from video order by duration asc;

select \* from ordered\_d;

RESULT: The given problems were successfully solved using the concept of views.

AIM: To execute triggers operations

QUERIES:

1. Create a trigger to raise application error “invalid flight id” if the flight id is not in the flights table.

create trigger invalid\_flight\_id

 -> after insert on reservation

 -> for each row

 -> begin

 -> if new.flight\_id is null then

 -> update reservation set flight\_id=10 where flight\_id=NUll;

 -> end if;

 -> end$

Query OK, 0 rows affected (0.12 sec)

1. Create a trigger to display the number of available seats for each flight.

create trigger canc

 -> after insert on reservation

 -> for each row

 -> begin

 -> insert into disp(flight\_id,capacity,travel\_date,available\_seats)

 -> values (new.flight\_id,new.capacity,new.travel\_date,new.available\_seats);

 -> end$

Query OK, 0 rows affected (0.03 sec)

1. Write a trigger or triggers that raises an exception when an attempt to insert a ticket occurs if the flight is already sold out

create trigger sold

 -> after insert on reservation

 -> for each row

 -> begin

 -> if available\_seats=0 then

 -> SIGNAL SQLSTATE '02000' SET MESSAGE\_TEXT = 'Warning!!!';

 -> end if;

 -> end$

Query OK, 0 rows affected (0.02 sec)

1. Write a trigger that decrements the available seat field when a ticket is deleted.

create trigger sold

 -> after delete

 -> on reservation

 -> for each row

 -> begin

 -> update disp set available\_seats = available\_seats -old.available\_seats;

 -> end$

Query OK, 0 rows affected (0.02 sec)

1. Create a trigger to cancel the tickets when the flight is cancelled

create trigger canc

-> after insert on reservation

 -> for each row

 -> begin

 -> if status='cancelled'

 -> then

 -> update disp set available\_seats=available\_seats+1 where status='cancelled';

 -> end if;

 -> end$

Query OK, 0 rows affected (0.02 sec)

1. update the salary of all first officer captains create a trigger to ensure the salary of first officer is less than the captain

create trigger upd

 -> after insert on pilots

-> for each row

 -> if rank\_ofc = 'Captain'

 -> then

 -> update pl set salary=salary+1 where rank\_ofc='First';

 -> end if;

 -> end$

Query OK, 0 rows affected (0.03 sec)

RESULT: The queries to execute trigger operations has been executed successfully.

AIM: To study and understand the concept of procedure

DESCRIPTION:

A stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again. So if you have an SQL query that you write over and over again, save it as a stored procedure, and then just call it to execute it.

SYNTAX:

CREATE PROCEDURE procedure\_name

 AS

 sql\_statement

 GO;

 EXEC procedure\_name;

1. Create a procedure to insert values into above tables.

delimiter $

create procedure ins\_customer(in cno int, in fname varchar(30), in lname varchar(30),in

city varchar(30),in country varchar(30),in credit\_limit float)

 -> begin

 -> insert into customer values(cno,fname,lname,city,country,credit\_limit);

 -> end$

create procedure ins\_product(in p\_code int,in p\_name varchar(20),in p\_vendor

varchar(20),in quantity int,in buy\_price float,in MRP float)

 -> begin

 -> insert into product values(p\_code,p\_name,p\_vendor,quantity,buy\_price,MRP);

 -> end$

create procedure ins\_orders(in o\_no int,in o\_date date,in delivery date,in shipped

date,in status varchar(30),in customer\_number varchar(12))

 -> begin

 -> insert into orders values(o\_no,o\_date,delivery,shipped,status,customer\_number);

 -> end$

create procedure ins\_orderdetails(in o\_no int,in p\_code int,in quantity\_ordered int,in

price\_each float,in order\_line\_number int)

 -> begin

 -> insert into order\_details

    values(o\_no,p\_code,quantity\_ordered,price\_each,order\_line\_number);

 -> end$

3) Create procedures to list all

* 1. Product details
  2. Customer details.
  3. Order details.

(a)

delimiter $

create procedure disp\_prod()

 -> begin

 -> select \* from product;

 -> end $

(b)

create procedure disp\_customer()

 -> begin

 -> select \* from customer;

 -> end $

(c)

create procedure disp\_order()

 -> begin

 -> select \* from order\_details;

 -> end $

Query OK, 0 rows affected (0.03 sec)

1. Create a procedure to find the product with maximum order and minimum order. (using OUT)

create procedure maxy(out maxo int)

 -> begin

 -> select max(quantity\_ordered) into maxo from order\_details;

 -> end$

create procedure maxi(out maxin int)

 -> begin

 -> select min(quantity\_ordered) into maxin from order\_details;

 -> end$

Query OK, 0 rows affected (0.03 sec)

1. Create a procedure to list the customers who have made purchase greater than 5000. (using IN)

create procedure disp5000(in val int)

 -> begin

 -> select \* from customers where credit\_limit >val;

 -> end$

1. Create a procedure to list orders date wise, get the date from the user.

create procedure orderdate()

 -> begin

 -> select \* from orders order by shipped;

 -> end$

1. Create a procedure to list the customers from Chennai who have made the maximum purchase

create procedure dispche()

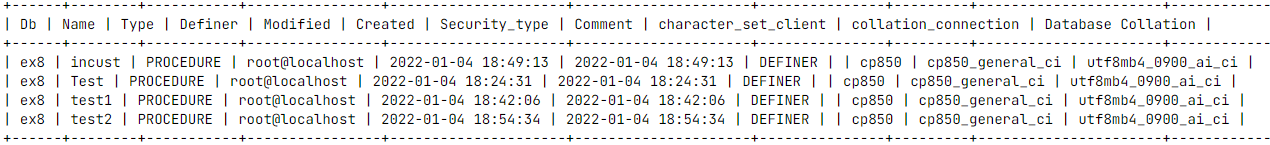
 -> begin

 -> select \* from customer where city='Chennai' having credit\_limit=max(credit\_limit);

 -> end$

1. Show or list all the stored procedures.

show procedure status where db='ex8';



1. List the product vendors who supplies more than one product. (use INOUT)

create procedure prod\_count(inout val int,in vendo varchar(20))

 -> begin

 -> select count(p\_code) into val from product where p\_vendor=vendo;

 -> end$

1. Drop any one procedure

drop procedure Test;

Query OK, 0 rows affected (0.03 sec)

RESULT: Thus the procedures are studied and executed successfully.

AIM: To execute Transactions and its operations

QUERIES:

1. Create Table accounts with fields Sno , Name , Account Number , Balance and Pin

create table accounts

   (Sno integer, name varchar(30), acc\_number integer, Balance

  integer, pin integer);

1. Apply Auto increment option to Sno Column

ALTER TABLE accounts

MODIFY Sno INT auto\_increment PRIMARY KEY;

1. Insert Values into Account Table

insert into accounts (name,acc\_number,Balance,pin)

values(" Raju ",123456789,25056,2731);

insert into accounts (name,acc\_number,Balance,pin)

values(" Ramya ",5678921,58942,2708);

insert into accounts (name,acc\_number,Balance,pin)

values(" Riya ",32112345,11256,3821);

insert into accounts (name,acc\_number,Balance,pin)

values(" Vino ",78945896,58746,1234);

1. Disable the auto commit mode –

start transaction;

set autocommit =0;

1. Create a Transaction to
2. Transfer 1000/- from Raju to Ramya

update accounts set Balance=Balance-1000 where name like "%raju%";

    Query OK, 0 rows affected (0.03sec)

update accounts set Balance=Balance+1000 where name like"%riya%";

    Query OK, 0 rows affected (0.03sec)

select \* from accounts;

Sno     name    acc\_number  Balance     pin

**---------------------------------------------**

1       Raju    123456789   24056       2731

2       Ramya   5678921     58942       2708

3       Riya    32112345    12256       3821

4       Vino    78945896    58746       1234

1. Credit 1000/- to the account number 78945896

update accounts set Balance=Balance+10000

where acc\_number=78945896;

select \* from accounts;

Sno     name    acc\_number  Balance     pin

**---------------------------------------------**

1       Raju    123456789   24056       2731

2       Ramya   56789212    58942       2708

3       Riya    32112345    12256       3821

4       Vino    78945896    68746       1234

1. Debit 5000 from Riya’s account

update accounts set Balance=Balance-5000

where name like "%riya%";

select \* from accounts;

Sno     name    acc\_number      Balance     pin

**---------------------------------------------------**

1       Raju    123456789       24056       2731

2       Ramya   5678921         58942       2708

3       Riya    32112345        7256        3821

4       Vino    78945896        68746       1234

1. Commit the transaction

commit

1. View the current auto commits mode settings.

SHOW VARIABLES WHERE Variable\_name='autocommit';

Variable\_name   Value

**----------------------**

autocommit      OFF

1. Create a transaction and define save points as follows
2. Transfer 2000/- from Riya to Vinu (s1)

start transaction;

update accounts set Balance=Balance-2000

WHERE name like "%riya";

update accounts set Balance=Balance+2000

WHERE name like "%vino";

savepoint s1;

1. Calculate 5% interest for all customers and update balance(s2)

update accounts set Balance=Balance+Balance\*(5/100);

savepoint s2;

1. Delete Ramya’s account (s3)

delete from accounts WHERE name like "%Ramya";

savepoint s3;

1. Insert (ajay , 45612332 , 25000)

insert into accounts(name,acc\_number,Balance,pin)

values("ajay",45612332,25000,2323);

update accounts set pin = 2369 WHERE name like "%Ramya";

savepoint s4;

1. Change the pin number of ramya to 2369 (s4)

update accounts set pin = 2369 WHERE name like "%Ramya";

savepoint s4;

1. Roll back to save point S4

mysql> select \* from accounts;

mysql> rollback to s4;

Query OK , 0 rows affected (0.06sec)

Sno name acc\_number Balance pin

---------------------------------------------------------------------------------

1 Raju 123456789 25259 2731

2 Ramya 5678921 61889 2708

3 Riya 32112345 7619 3821

4 Vino 78945896 72183 1234

5 ajay 45612332 25000 2323

1. Roll back to point S2

mysql> select \* from accounts;

mysql> rollback to s2;

Query OK , 0 rows affected (0.06sec)

Sno name acc\_number Balance pin

---------------------------------------

1 Raju 123456789 25259 2731

2 Ramya 5678921 61889 2708

3 Riya 32112345 7619 3821

4 Vino 78945896 72183 1234

1. Create a read only transaction to
2. View all customer balance

mysql> start transaction read only;

mysql> select \* from accounts;

Sno name acc\_number Balance pin

----------------------------------------------------------------------------

1 Raju 123456789 25259 2731

2 Ramya 5678921 61889 2708

3 Riya 32112345 7619 3821

4 Vino 78945896 72183 1234

1. To view account number and balance of all customers

mysql> select acc\_number , Balance from accounts;

acc\_number Balance

-------------------------------------

123456789 25259

5678921 61889

32112345 7619

78945896 72183

1. Update the balance of riya to 50,000/-

mysql> update accounts set Balance=50000 WHERE name="riya";

ERROR 1792 (25006) at line 67: Cannot execute statement in a READ ONLY transaction.

14. Lock the account table in read mode

mysql> lock table accounts read ;

Query OK , 0 rows affected (0.06sec)

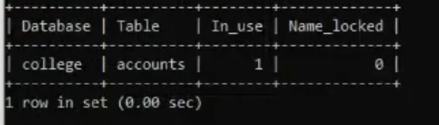
15. Update pin number of ramya to 1020

mysql> update accounts set pin=1020 WHERE name="ramya";

ERROR 1099 (HY000) at line 71: Table 'accounts' was locked with a READ lock and can't be updated

16. View all the lock details

mysql> SHOW open tables WHERE in\_use>1;



17. Unlock account table

mysql> unlock tables;

Query OK , 0 rows affected (0.06sec)

18. Lock the amount table for write access

mysql> lock table accounts write ;

Query OK , 0 rows affected (0.06sec)

19. Increase the balance of all account holders by 10%

mysql> update accounts set Balance=Balance+ Balance\*(10/100);

Query OK , 0 rows affected (0.06sec)

20. Lock the account table for read access in different session

mysql> start transaction;

mysql> lock table accounts read ;

mysql> commit;

22. Set the transaction isolation level to read uncommitted

mysql> set transaction isolation level read uncommitted;

Query OK , 0 rows affected (0.06sec)

23. Set the transaction isolation level to serializable

mysql> set transaction isolation level serializable;

Query OK , 0 rows affected (0.06sec)

24. Create an index i1 for account table on account number attribute

mysql> create index i1 on accounts(acc\_number);

Query OK , 0 rows affected (0.06sec)

25. Create an index i2 for account table on account no and pin

mysql> create index i2 on accounts(acc\_number,pin);

Query OK , 0 rows affected (0.06sec)

26. View all the indexes

mysql> SHOW index from accounts;

Table Non\_unique Key\_name Seq\_in\_index Column\_name Collation Cardinality Sub\_part

Packed Null Index\_type Comment Index\_comment Visible Expression

accounts 0 PRIMARY 1 Sno A 2 NULL NULL BTREE YES NULL

accounts 1 i1 1 acc\_number A 4 NULL NULL YES BTREE YES NULL

accounts 1 i2 1 acc\_number A 4 NULL NULL YES BTREE YES NULL

accounts 1 i2 2 pin A 4 NULL NULL YES BTREE YES NULL

27. View the performance of select ac from acc where balance>1000

mysql> select acc\_number from accounts WHERE Balance>1000;

acc\_number

-----------

123456789

5678921

32112345

78945896

Query OK (0.3sec)

28.Add index using balance table

mysql> create index i3 on accounts(Balance);

Query OK , 0 rows affected (0.06sec)

29. View the performance of query in question 27

mysql> select acc\_number from accounts WHERE Balance>1000;

acc\_number

-----------

123456789

5678921

32112345

78945896

Query OK (0.03sec)

30. Drop all the indexes

mysql> alter table accounts DROP index i1;

mysql> alter table accounts DROP index i2;

mysql> alter table accounts DROP index i3;

RESULT: The queries are executed successfully.