

DATA MANAGEMENT AND DATABASE DESIGN

INFO 6210 : FINAL PROJECT

Airline Management Database Design

Presented By :

Sudeep Surisetty

001887695

Instructor :

Yusuf Ozbek

1. PROJECT DESCRIPTION

With the increase in competition among the airline companies, the flight fares have come within the affordability of a common man. Due to which travel by air is being preferred by more number of people and there has been an exponential rise in the number of passengers and the number of flight trips. To tackle such a huge traffic every airline needs to maintain consistent and durable database design for smooth flow of its operations.

The system is based on airline management. Airline Management System primarily deals with data management of passengers, flight routes, and its employees. The system provides an overview of the underlying operational factors.

ASSUMPTIONS :

- I have taken into account for only one airliner operations management which includes only passenger flights and not the cargo flights.
- There are connecting flights involved, every flight schedule mentioned is only up to the destination specified.
- The details of flights are taken as a snapshot of a single day.
- There are different types of job roles in an airline company but for the simplicity of the system, only few job roles are considered.

2. ENTITIES

Person: Employees can also be passengers so creating Person entity.

Address: Every person has one or more address.

Employee: Airline has employees.

Passenger: Each passenger is the one who is a customer of the airline.

Flight passenger: This contains information about which flight a passenger is on, their itinerary number and seat number.

Flights: List of all scheduled flights. And each flight has a route and with departure, arrival time.

Flight Status: Includes predefined types of status for a flight.

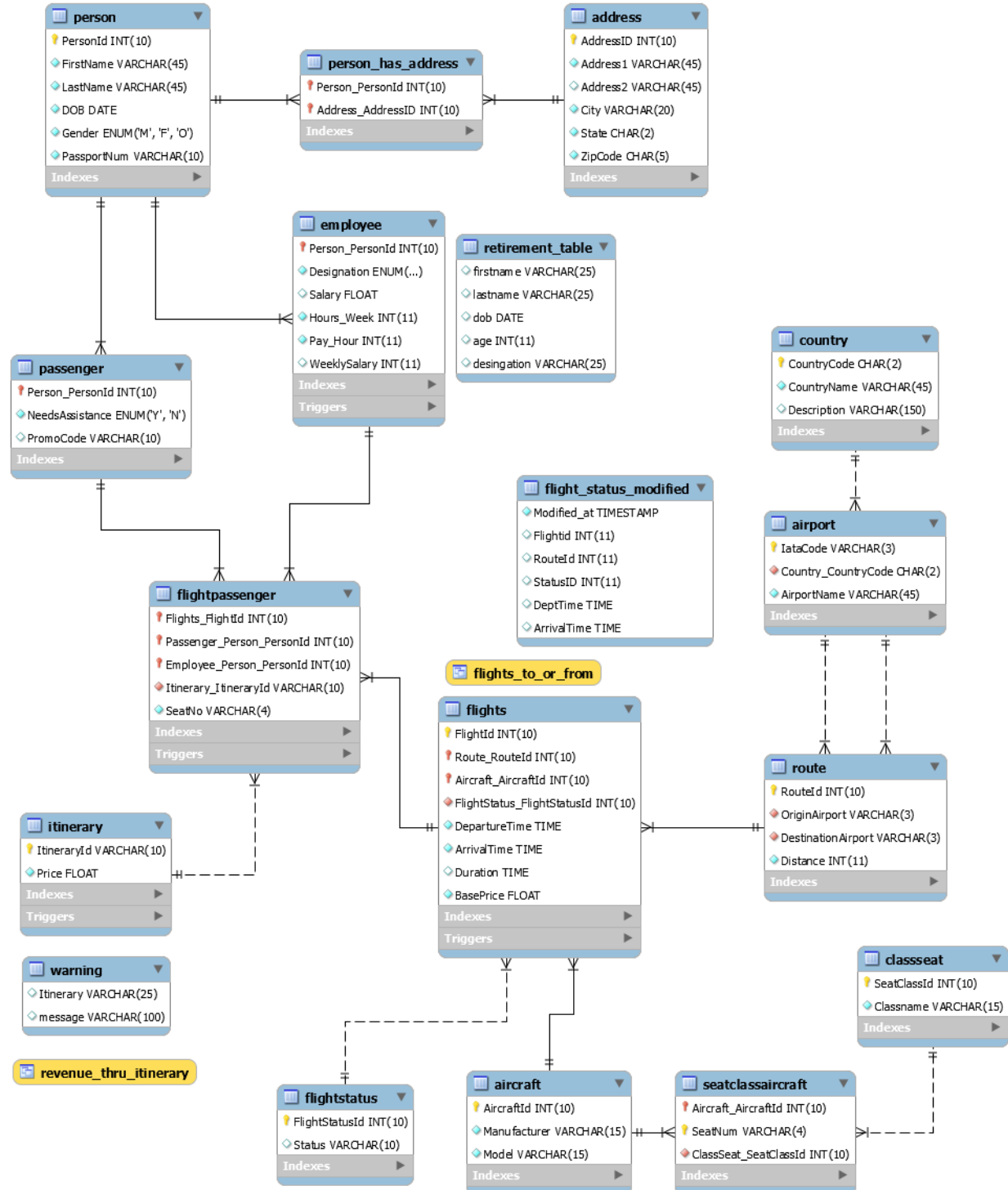
Route: This has the information about origin and destinations.

Airport: Every airport belongs to one country and has IATA code.

Itinerary: The table contains a list of all passenger itineraries. Many passengers can have the same itinerary. Many passengers can have many itineraries. It also has the cost of per person for the itinerary(not the flight).

Aircraft: Each aircraft has the same set of seat classes which in turn has seat numbers.

3. ER/EER DIAGRAM



ER Diagram contains following relationships. (Associative entity has been included for m:n relations)

ENTITY 1	ENTITY 2	CARDINALITY
Person	Address	m : n
Passenger	Person	1 : 1
Employee	Person	1 : 1
Passenger	FlightPassenger	1 : m
Itinerary	FlightPassenger	1 : m
FlightPassenger	Employee	m : 1
FlightPassenger	Flights	m : 1
ClassSeat	SeatClassAircraft	1 : m
SeatClassAircraft	Aircraft	m : 1
Aircraft	Flights	1 : m
FlightStatus	Flights	1 : m
Country	Airport	1 : m
Airport	Route	1 : m
Flights	Route	m : 1

4. MySQL

TRIGGERS, STORED PROCEDURES, FUNCTIONS, VIEWS, QUERIES

TRIGGERS

1) DONOT ALLOCATE SEAT IF ITS ALREADY RESERVED

Creating a trigger before insert and update which check the seat availability. Inside the trigger we are obtaining the Route and Aircraft type.

And also "is_seat_available () " function is called which returns boolean to mention if any passenger is present for the seat.

TRIGGER:

```
CREATE DEFINER=`root`@`localhost` TRIGGER  
`airlinedb`.`flightpassenger_BEFORE_INSERT` BEFORE INSERT ON `flightpassenger` FOR  
EACH ROW
```

```
BEGIN
```

```
DECLARE IdRoute int;
```

```
DECLARE IdAircraft int;
```

```
DECLARE SeatNo varchar(4);
```

```
DECLARE avail boolean;
```

```
SET avail = 1;
```

```
SET SeatNo = NEW.SeatNo;
```

```
SET IdRoute =(SELECT flights.route_routeld FROM flightpassenger
```

```
INNER JOIN flights
```

```
ON flightpassenger.Flights_FlightId = flights.flightid
```

```
WHERE Flights_FlightId = NEW.Flights_FlightId
```

```
GROUP BY flights.route_routeld);
```

```
SET IdAircraft =(SELECT flights.aircraft_aircraftId FROM flightpassenger  
INNER JOIN flights  
ON flightpassenger.Flights_FlightId = flights.flightid  
WHERE Flights_FlightId = NEW.Flights_FlightId  
GROUP BY flights.route_routeld);
```

```
SET avail = is_seat_available(NEW.Flights_FlightId,IdRoute,IdAircraft,SeatNo);
```

```
if(avail = 0)  
THEN  
set NEW.SeatNo=SeatNo;  
ELSE  
set NEW.SeatNo='NULL';  
END IF;
```

```
END
```

FUNCTION :

```
CREATE DEFINER=`root`@`localhost` FUNCTION `is_seat_available`(IdFlight int, IdRoute  
int, IdAircraft int, seat varchar(4)) RETURNS tinyint(1)
```

```
BEGIN
```

```
DECLARE var varchar(10);
```

```
DECLARE valid boolean;
```

```
SET var = "";
```

```
SET var = (SELECT flightpassenger.Passenger_Person_PersonId FROM flights
```

```
INNER JOIN FlightPassenger
```

```
ON FlightId = flightpassenger.Flights_FlightId
```

```
WHERE (FlightId=IdFlight AND flights.Route_RouteId=IdRoute AND  
flights.Aircraft_AircraftId = IdAircraft AND flightpassenger.SeatNo = seat)  
);
```

```
IF (var != "")
```

```
THEN
```

```
SET valid = 1;
```

```
ELSE
```

```
SET valid = 0;
```

```
END IF;
```

```
return valid;
```

```
END
```

* Trying to book seat which is already taken by a passenger in flightID = 307

The screenshot shows the MySQL Workbench interface. The top menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. The left sidebar contains a Navigator pane with sections for MANAGEMENT (Server Status, Client Connections, Users and Privileges, Status and System Variables, Data Export, Data Import/Restore), INSTANCE (Startup / Shutdown, Server Logs, Options File), PERFORMANCE (Dashboard, Performance Reports, Performance Schema Setup), and SCHEMAS (Filter objects, employee, flightpassenger, flights, flightstatus). The main editor area shows a SQL query: `SELECT * FROM airlinedb.flightpassenger;`. Below the query editor is a toolbar with icons for Result Grid, Filter Rows, Edit, Export/Import, and Wrap Cell Content. The results are displayed in a table with columns: Flights_FlightId, Passenger_Person_PersonId, Employee_Person_PersonId, Itinerary_ItineraryId, and SeatNo. The table contains 20 rows of data. The bottom pane shows the Output window with a table of Action Output, including columns for #, Time, Action, Message, and Duration / Fetch. The output shows two actions: a drop table operation and a select operation. The status bar at the bottom indicates the time is 2:04 AM on 12/13/2017.

MySQL Workbench

Local instance MySQL Router

File Edit View Query Database Server Tools Scripting Help

Navigator

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

PERFORMANCE

- Dashboard
- Performance Reports
- Performance Schema Setup

SCHEMAS

Filter objects

- employee
- flightpassenger
- flights
- flightstatus

SQL File 1* itinerary - Table SQL File 3* flightpassenger

Limit to 1000 rows

1 • `SELECT * FROM airlinedb.flightpassenger;`

Result Grid

Flights_FlightId	Passenger_Person_PersonId	Employee_Person_PersonId	Itinerary_ItineraryId	SeatNo
300	10	4	BK24580	3E
307	16	3	CF78099	5G
307	19	5	CV28514	5G
308	2	3	DR45927	5F
309	4	4	DR45927	1B
314	3	5	CF78099	2C
...

Information

Table: **itinerary**

Columns:

- ItineraryId** varchar(10) PK
- Price float UN

Object Info Session

Query Completed

Output

Action Output

#	Time	Action	Message	Duration / Fetch
4	02:00:47	drop table flight_status_modified	0 row(s) affected	0.172 sec
5	02:04:30	SELECT * FROM airlinedb.flightpassenger LIMIT 0, 10...	20 row(s) returned	0.000 sec / 0.000 sec

2:04 AM
12/13/2017

MySQL Workbench

Local instance MySQL Router x

File Edit View Query Database Server Tools Scripting Help

Navigator

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

PERFORMANCE

- Dashboard
- Performance Reports
- Performance Schema Setup

SCHEMAS

Filter objects

- employee
- flightpassenger
- flights
- flightstatus

Information

Table: **itinerary**

Columns:

- ItineraryId** varchar(10) PK
- Price float UN

Object Info Session

Query Completed

SQL File 1* itinerary - Table SQL File 3* flightpassenger x

Limit to 1000 rows

```
1 SELECT * FROM airlinedb.flightpassenger;
```

Result Grid

Flights_FlightId	Passenger_Person_PersonId	Employee_Person_PersonId	Itinerary_ItineraryId	SeatNo
300	10	4	BK24580	3E
307	16	3	CF78099	5G
307	19	5	CV28514	NULL
308	2	3	DR45927	5F
309	4	4	DR45927	1B
314	3	5	CF78099	2C

flightpassenger 1 x

Apply Revert

Output

Action Output

#	Time	Action	Message	Duration / Fetch
4	02:00:47	drop table flight_status_modified	0 row(s) affected	0.172 sec
5	02:04:30	SELECT * FROM airlinedb.flightpassenger LIMIT 0, 10...	20 row(s) returned	0.000 sec / 0.000 sec

2:05 AM
12/13/2017

*trying to book a new seat which is not reserved before

The screenshot displays the MySQL Workbench interface. The top menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. The left sidebar contains a Navigator pane with sections for MANAGEMENT (Server Status, Client Connections, Users and Privileges, Status and System Variables, Data Export, Data Import/Restore), INSTANCE (Startup / Shutdown, Server Logs, Options File), PERFORMANCE (Dashboard, Performance Reports, Performance Schema Setup), and SCHEMAS (Filter objects, employee, flightpassenger, flights, flightstatus). The main editor window shows a SQL query: `SELECT * FROM airlinedb.flightpassenger;`. Below the editor, the Result Grid displays the query results in a table format. The table has five columns: Flights_FlightId, Passenger_Person_PersonId, Employee_Person_PersonId, Itinerary_ItineraryId, and SeatNo. The results show 20 rows of data. The bottom pane shows the Output window with a table of Action Output, including columns for #, Time, Action, Message, and Duration / Fetch. The output shows two actions: a drop table operation and a select operation. The status bar at the bottom indicates the query was completed at 2:05 AM on 12/13/2017.

MySQL Workbench

Local instance MySQL Router

File Edit View Query Database Server Tools Scripting Help

Navigator

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

PERFORMANCE

- Dashboard
- Performance Reports
- Performance Schema Setup

SCHEMAS

Filter objects

- employee
- flightpassenger
- flights
- flightstatus

SQL File 1* itinerary - Table SQL File 3* flightpassenger

Limit to 1000 rows

1 • `SELECT * FROM airlinedb.flightpassenger;`

Result Grid

Flights_FlightId	Passenger_Person_PersonId	Employee_Person_PersonId	Itinerary_ItineraryId	SeatNo
300	10	4	BK24580	3E
307	16	3	CF78099	5G
307	19	5	CV28514	5F
308	2	3	DR45927	5F
309	4	4	DR45927	1B
314	3	5	CF78099	2C
...

Information

Table: **itinerary**

Columns:

- ItineraryId** varchar(10) PK
- Price float UN

Object Info Session

Query Completed

Output

Action Output

#	Time	Action	Message	Duration / Fetch
4	02:00:47	drop table flight_status_modified	0 row(s) affected	0.172 sec
5	02:04:30	SELECT * FROM airlinedb.flightpassenger LIMIT 0, 10...	20 row(s) returned	0.000 sec / 0.000 sec

2:05 AM
12/13/2017

MySQL Workbench

Local instance MySQL Router

File Edit View Query Database Server Tools Scripting Help

Navigator

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

PERFORMANCE

- Dashboard
- Performance Reports
- Performance Schema Setup

SCHEMAS

Filter objects

- employee
- flightpassenger
- flights
- flightstatus

Information

Table: **itinerary**

Columns:

- ItineraryId** varchar(10) PK
- Price float UN

Object Info Session

Query Completed

SQL File 1* itinerary - Table SQL File 3* flightpassenger

Limit to 1000 rows

```
1 SELECT * FROM airlinedb.flightpassenger;
```

Result Grid

Flights_FlightId	Passenger_Person_PersonId	Employee_Person_PersonId	Itinerary_ItineraryId	SeatNo
300	10	4	BK24580	3E
307	16	3	CF78099	5G
307	19	5	CV28514	5F
308	2	3	DR45927	5F
309	4	4	DR45927	1B
314	3	5	CF78099	2C
314	9	3	YE23002	4E

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	01:51:13	select person.FirstName as FirstName, Passenger_Per...	2 row(s) returned	0.062 sec / 0.000 sec

2:06 AM
12/13/2017

2) Flight time check

Here it will check if arrival and departure time are same or not in 1st loop.

Then it will check if the updated values are properly done within the restrictions and then calculates the duration accordingly.

```
CREATE DEFINER=`root`@`localhost` TRIGGER `airlinedb`.`flights_BEFORE_UPDATE`  
BEFORE UPDATE ON `flights` FOR EACH ROW
```

```
BEGIN
```

```
IF(NEW.ArrivalTime = NEW.DepartureTime) || (NEW.ArrivalTime = OLD.DepartureTime)  
|| (NEW.DepartureTime = OLD.ArrivalTime)
```

```
THEN
```

```
SET NEW.DepartureTime = OLD.DepartureTime;
```

```
SET NEW.ArrivalTime = OLD.ArrivalTime;
```

```
END IF;
```

```
IF((NEW.ArrivalTime <> OLD.ArrivalTime ) || (NEW.DepartureTime <> OLD.ArrivalTime))
```

```
THEN
```

```
IF (NEW.ArrivalTime < NEW.DepartureTime)
```

```
THEN
```

```
SET NEW.DepartureTime = OLD.DepartureTime;
```

```
SET NEW.ArrivalTime = OLD.ArrivalTime;
```

```
END IF;
```

```
SET NEW.Duration = timediff(NEW.arrivaltime,NEW.departuretime);
```

```
END IF;END
```

*Trying to change the 307 Flight ID departure time more than arrival time and gets rejected.

The screenshot shows the MySQL Workbench interface. The left sidebar contains the 'Navigator' pane with sections for 'MANAGEMENT' (Server Status, Client Connections, Users and Privileges, Status and System Variables, Data Export, Data Import/Restore), 'INSTANCE' (Startup / Shutdown, Server Logs, Options File), 'PERFORMANCE' (Dashboard, Performance Reports, Performance Schema Setup), and 'SCHEMAS' (Filter objects, flightpassenger, flights, flightstatus, itinerary). The 'flights' table is selected in the 'SCHEMAS' pane. The main query editor shows the query: `SELECT * FROM airlinesb.flights;`. The 'Result Grid' displays the following data:

FlightId	Route_RouteId	Aircraft_AircraftId	FlightStatus_FlightStatusId	DepartureTime	ArrivalTime	Duration	BasePrice
300	6	4	1	05:02:00	07:01:00	01:59:00	551
307	7	4	1	15:26:10	14:06:00	04:39:50	486
308	1	4	1	04:35:00	08:45:00	04:10:00	134
309	3	2	3	03:54:00	07:37:00	03:43:00	567
314	4	4	1	05:19:00	09:46:00	04:27:00	412
316	9	2	3	07:45:00	09:10:00	01:25:00	70
327	2	3	2	01:00:00	03:40:00	02:40:00	167
330	7	4	3	10:35:00	13:33:00	02:58:00	100

The 'Output' pane at the bottom shows the execution log:

#	Time	Action	Message	Duration / Fetch
3	16:02:14	select person.FirstName as FirstName, Passenger_Per...	2 row(s) returned	0.000 sec / 0.000 sec

The system clock at the bottom right indicates 4:04 PM on 12/13/2017.

MySQL Workbench

Local instance MySQL Router

File Edit View Query Database Server Tools Scripting Help

Navigator

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

PERFORMANCE

- Dashboard
- Performance Reports
- Performance Schema Setup

SCHEMAS

Filter objects

- address
- aircraft
- airport
- classseat

Information

No object selected

Object Info Session

query flightpassenger - Table SQL File 2* flights - Table flights

Limit to 1000 rows

```
1 • SELECT * FROM airlinesdb.flights;
```

Result Grid

FlightId	Route_RouteId	Aircraft_AircraftId	FlightStatus_FlightStatusId	DepartureTime	ArrivalTime	Duration	BasePrice
300	6	4	1	05:02:00	07:01:00	01:59:00	551
307	7	4	1	09:26:10	14:06:00	04:39:50	486
308	1	4	1	04:35:00	08:45:00	04:10:00	134
309	3	2	3	03:54:00	07:37:00	03:43:00	567
314	4	4	1	05:19:00	09:46:00	04:27:00	412
316	9	2	3	07:45:00	09:10:00	01:25:00	70
327	2	3	2	01:00:00	03:40:00	02:40:00	167
330	7	4	3	10:35:00	13:33:00	02:58:00	100

flights 1 x

Apply Revert

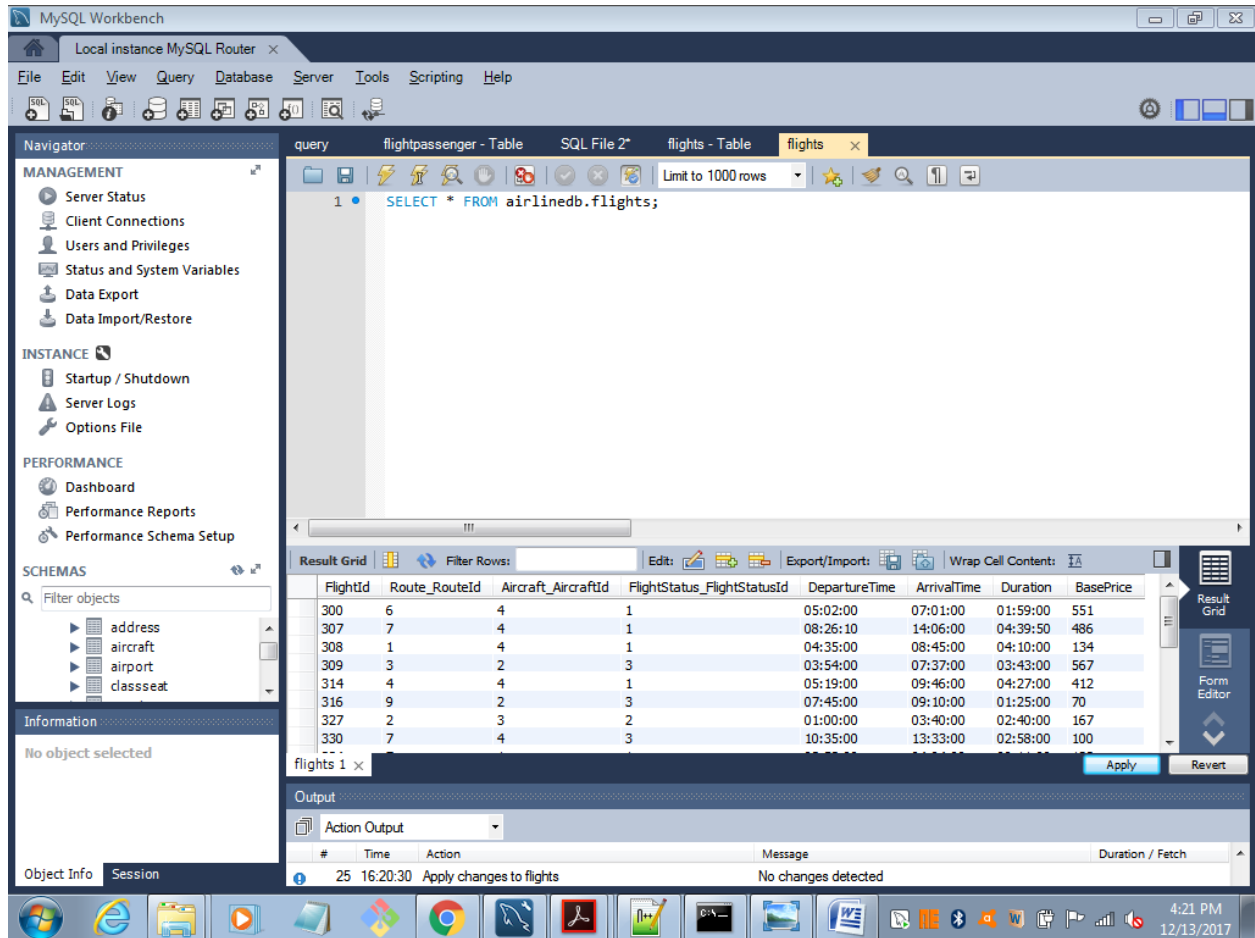
Output

Action Output

#	Time	Action	Message	Duration / Fetch
25	16:20:30	Apply changes to flights	No changes detected	

4:21 PM 12/13/2017

* The time gets updated when both departure and arrival are given properly



The screenshot shows the MySQL Workbench interface. The left sidebar contains the Navigator pane with sections for MANAGEMENT, INSTANCE, PERFORMANCE, and SCHEMAS. The SCHEMAS section is expanded, showing a list of objects including address, aircraft, airport, and classseat. The main query editor displays the SQL query: `SELECT * FROM airlinesdb.flights;`. Below the query editor, the Result Grid shows the data for the 'flights' table. The table has columns: FlightId, Route_RouteId, Aircraft_AircraftId, FlightStatus_FlightStatusId, DepartureTime, ArrivalTime, Duration, and BasePrice. The data is displayed in a grid format with 10 rows. The bottom status bar shows the message: "Apply changes to flights" and "No changes detected".

FlightId	Route_RouteId	Aircraft_AircraftId	FlightStatus_FlightStatusId	DepartureTime	ArrivalTime	Duration	BasePrice
300	6	4	1	05:02:00	07:01:00	01:59:00	551
307	7	4	1	08:26:10	14:06:00	04:39:50	486
308	1	4	1	04:35:00	08:45:00	04:10:00	134
309	3	2	3	03:54:00	07:37:00	03:43:00	567
314	4	4	1	05:19:00	09:46:00	04:27:00	412
316	9	2	3	07:45:00	09:10:00	01:25:00	70
327	2	3	2	01:00:00	03:40:00	02:40:00	167
330	7	4	3	10:35:00	13:33:00	02:58:00	100

MySQL Workbench

Local instance MySQL Router

File Edit View Query Database Server Tools Scripting Help

Navigator

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

PERFORMANCE

- Dashboard
- Performance Reports
- Performance Schema Setup

SCHEMAS

Filter objects

- address
- aircraft
- airport
- classseat

Information

No object selected

Object Info Session

query

flightpassenger - Table SQL File 2* flights - Table flights

Limit to 1000 rows

1 • SELECT * FROM airlinedb.flights;

Result Grid

FlightId	Route_RouteId	Aircraft_AircraftId	FlightStatus_FlightStatusId	DepartureTime	ArrivalTime	Duration	BasePrice
300	6	4	1	05:02:00	07:01:00	01:59:00	551
307	7	4	1	08:26:10	14:06:00	05:39:50	486
308	1	4	1	04:35:00	08:45:00	04:10:00	134
309	3	2	3	03:54:00	07:37:00	03:43:00	567
314	4	4	1	05:19:00	09:46:00	04:27:00	412
316	9	2	3	07:45:00	09:10:00	01:25:00	70
327	2	3	2	01:00:00	03:40:00	02:40:00	167
330	7	4	3	10:35:00	13:33:00	02:58:00	100

flights 1 x

Apply Revert

Output

Action Output

#	Time	Action	Message	Duration / Fetch
25	16:20:30	Apply changes to flights	No changes detected	

4:22 PM 12/13/2017

3) Flight Status modified

```
CREATE DEFINER=`root`@`localhost` TRIGGER `airlinedb`.`flights_STATUS_AFTER_UPDATE`  
AFTER UPDATE ON `flights` FOR EACH ROW  
  
BEGIN  
  
IF OLD.FlightStatus_FlightStatusId <> NEW.FlightStatus_FlightStatusId  
  
THEN  
  
INSERT INTO flight_status_modified VALUES(  
        NOW(),  
NEW.FlightId,NEW.Route_RouteId,NEW.FlightStatus_FlightStatusId,NEW.DepartureTime,NE  
W.ArrivalTime  
        );  
  
END IF;  
  
END
```

MySQL Workbench

Local instance MySQL Router

File Edit View Query Database Server Tools Scripting Help

Navigator

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

PERFORMANCE

- Dashboard
- Performance Reports
- Performance Schema Setup

SCHEMAS

Filter objects

- address
- aircraft
- airport
- classseat

Information

No object selected

Object Info Session

query flightpassenger - Table SQL File 2* flights - Table flights

Limit to 1000 rows

```
1 • SELECT * FROM airlinesdb.flights;
```

Result Grid

FlightId	Route_RouteId	Aircraft_AircraftId	FlightStatus_FlightStatusId	DepartureTime	ArrivalTime	Duration	BasePrice
350	6	3	3	08:55:00	13:04:00	04:09:00	285
355	8	4	4	11:25:00	14:20:00	02:55:00	205
358	5	3	2	06:06:00	10:59:00	04:53:00	672
360	9	2	1	09:45:00	10:17:00	00:32:00	471
361	4	3	1	02:04:00	13:19:00	11:15:00	222
364	2	1	3	15:13:00	18:18:00	03:05:00	686
367	7	4	3	09:42:00	12:34:00	02:52:00	66
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

flights 1 x

Apply Revert

Output

Action Output

#	Time	Action	Message	Duration / Fetch
25	16:20:30	Apply changes to flights	No changes detected	

4:26 PM 12/13/2017

MySQL Workbench

Local instance MySQL Router

File Edit View Query Database Server Tools Scripting Help

Navigator

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

PERFORMANCE

- Dashboard
- Performance Reports
- Performance Schema Setup

SCHEMAS

Filter objects

- address
- aircraft
- airport
- classseat

Information

No object selected

Object Info Session

query flightpassenger - Table SQL File 2* flights - Table flights

1 • SELECT * FROM airlinesdb.flights;

Limit to 1000 rows

Result Grid

FlightId	Route_RouteId	Aircraft_AircraftId	FlightStatus_FlightStatusId	DepartureTime	ArrivalTime	Duration	BasePrice
350	6	3	3	08:55:00	13:04:00	04:09:00	285
355	8	4	4	11:25:00	14:20:00	02:55:00	205
358	5	3	2	06:06:00	10:59:00	04:53:00	672
360	9	2	1	09:45:00	10:17:00	00:32:00	471
361	4	3	1	02:04:00	13:19:00	11:15:00	222
364	2	1	3	15:13:00	18:18:00	03:05:00	686
367	7	4	4	09:42:00	12:34:00	02:52:00	66
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

flights 1 x

Apply Revert

Output

Action Output

#	Time	Action	Message	Duration / Fetch
25	16:20:30	Apply changes to flights	No changes detected	

4:27 PM 12/13/2017

MySQL Workbench

Local instance MySQL Router

File Edit View Query Database Server Tools Scripting Help

Navigator

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

PERFORMANCE

- Dashboard
- Performance Reports
- Performance Schema Setup

SCHEMAS

Filter objects

- address
- aircraft
- airport
- classseat

Information

No object selected

Object Info Session

query

flightpassenger - Table SQL File 2* flights - Table flights

```
1 CREATE TABLE IF NOT EXISTS flight_status_modified (Modified_at TIMESTAMP, Flightid int, RouteId int,
2 ;
3
4 SELECT * FROM flight_status_modified;
5
6 DROP TABLE flight_status_modified ;
```

Limit to 1000 rows

Result Grid

Modified_at	Flightid	RouteId	StatusID	DeptTime	ArrivalTime
2017-12-13 16:27:15	367	7	4	09:42:00	12:34:00

flight_status_modified 3

Output

Action Output

#	Time	Action	Message	Duration / Fetch
26	16:20:32	Apply changes to flights	No changes detected	

4:27 PM 12/13/2017

4) Weekly Salary gets calculated based on hours/week and pay/hour.

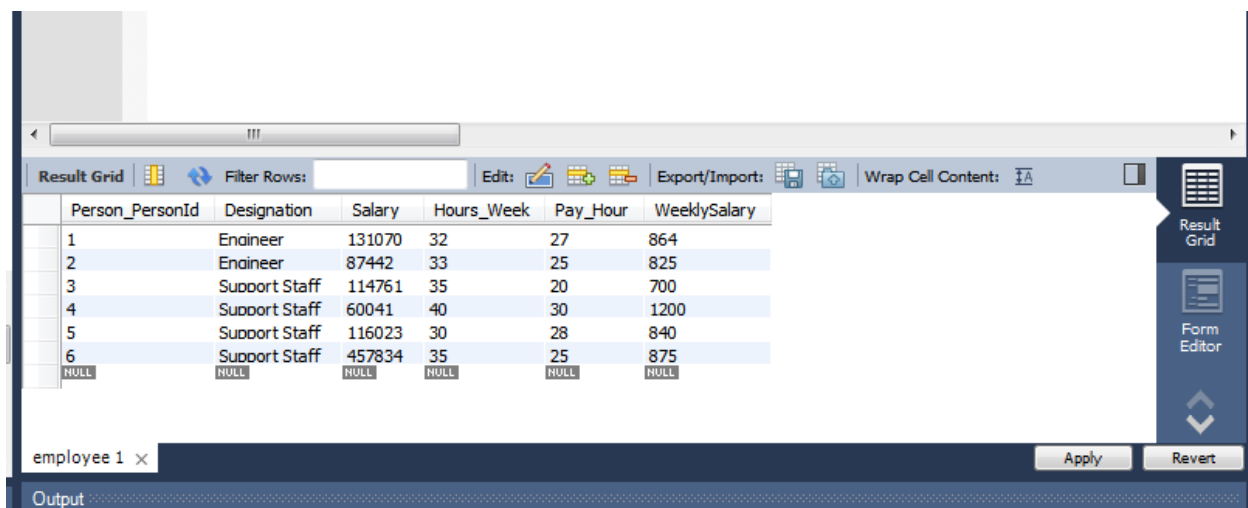
```
CREATE DEFINER='root'@'localhost' TRIGGER `airlinedb`.`employee_BEFORE_INSERT`  
BEFORE INSERT ON `employee` FOR EACH ROW
```

```
BEGIN
```

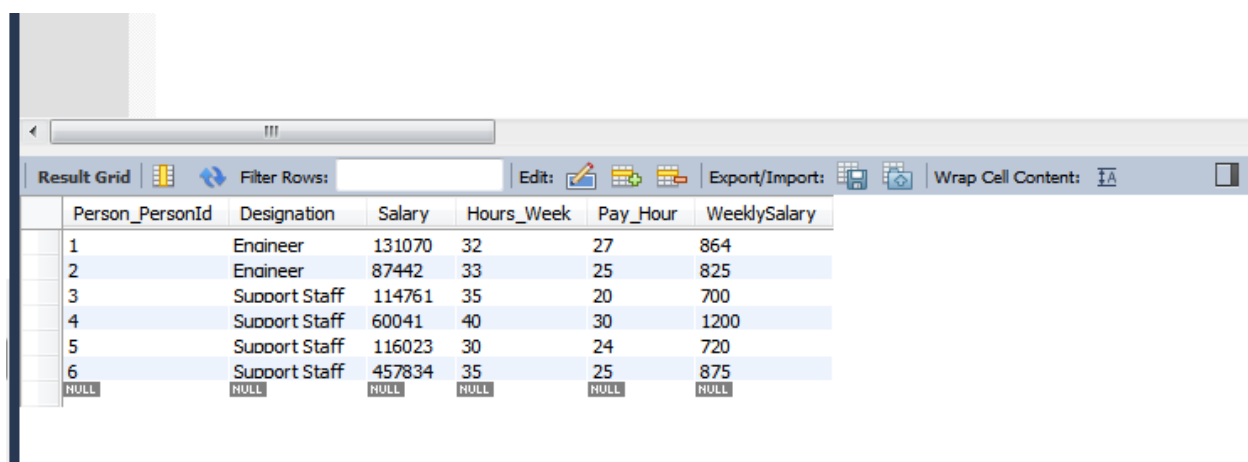
```
set new.weeklysalary = new.hours_week * new.pay_hour;
```

```
END
```

*updating person 5 - hourly pay



Person_PersonId	Designation	Salary	Hours_Week	Pay_Hour	WeeklySalary
1	Engineer	131070	32	27	864
2	Engineer	87442	33	25	825
3	Support Staff	114761	35	20	700
4	Support Staff	60041	40	30	1200
5	Support Staff	116023	30	24	720
6	Support Staff	457834	35	25	875
NULL	NULL	NULL	NULL	NULL	NULL



Person_PersonId	Designation	Salary	Hours_Week	Pay_Hour	WeeklySalary
1	Engineer	131070	32	27	864
2	Engineer	87442	33	25	825
3	Support Staff	114761	35	20	700
4	Support Staff	60041	40	30	1200
5	Support Staff	116023	30	24	720
6	Support Staff	457834	35	25	875
NULL	NULL	NULL	NULL	NULL	NULL

5) Setting Base Price to default or old value

Creating a table called Warning and setting prices based on insert or updates.

Making sure valid base price is entered before update as well, else it will revert back to old price value. And logs are stored in warning table.

*Before insert

```
CREATE DEFINER=`root`@`localhost` TRIGGER `airlinedb`.`itinerary_BEFORE_INSERT`  
BEFORE INSERT ON `itinerary` FOR EACH ROW
```

```
BEGIN
```

```
DECLARE msg VARCHAR(50);
```

```
IF (NEW.Price < 50)
```

```
THEN
```

```
SET NEW.Price = 50;
```

```
SET msg = 'Setting price to 50$ as it cant be less than that';
```

```
INSERT INTO Warning VALUES(NEW.ItineraryId,msg);
```

```
END IF;
```

```
END
```

* Before update

```
CREATE DEFINER=`root`@`localhost` TRIGGER `airlinedb`.`itinerary_BEFORE_UPDATE`  
BEFORE UPDATE ON `itinerary` FOR EACH ROW
```

```
BEGIN
```

```
DECLARE msg VARCHAR(50);
```

IF (NEW.Price < 50)

THEN

SET NEW.Price = OLD.Price;

INSERT INTO Warning VALUES(NEW.ItineraryId,'Setting price to old price as it cant be less than that');

END IF;

END

The screenshot shows a software interface with a list of warnings on the left and a 'Form Editor' button on the right. The list contains three entries:

ID	Description
CF78099	Setting price to 50\$ as it cant be less than that
BK24580	Setting price to 50\$ as it cant be less than that
BK24580	Setting price to old price as it cant be less than ...

Below the list is a 'warning 1' tab with a close button. To the right of the list is a 'Form Editor' button with a downward arrow icon. At the bottom right, there is a 'Read Only' status indicator.

FUNCTIONS:

1) Check the flight duration of a flight through the flight ID

Providing a facility to directly know the flight duration just by giving FlightID as input.

FUCNTION :

```
CREATE DEFINER=`root`@`localhost` FUNCTION `flight_duration`(idFlight int) RETURNS  
time
```

```
BEGIN
```

```
    DECLARE duration time;
```

```
    Set duration = (select timediff(arrivaltime,departuretime) as Duration from flights
```

```
    where flights.FlightId = idFlight );
```

```
return duration;
```

```
END
```


MySQL Workbench

Local instance MySQL Router

File Edit View Query Database Server Tools Scripting Help

Navigator

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variable
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

PERFORMANCE

- Dashboard
- Performance Reports
- Performance Schema Setup

SCHEMAS

Filter objects

- dest_facts
- Functions
 - f() flight_duration
 - f() is_seat_available

Information

No object selected

Object Info Session

SQL File 1* SQL File 4 x

Limit to 1000 rows

1

Call stored function `airlinedb.flight_duration`

Enter values for parameters of your function and click <Execute> to create an SQL editor and run the call:

idFlight 307 int

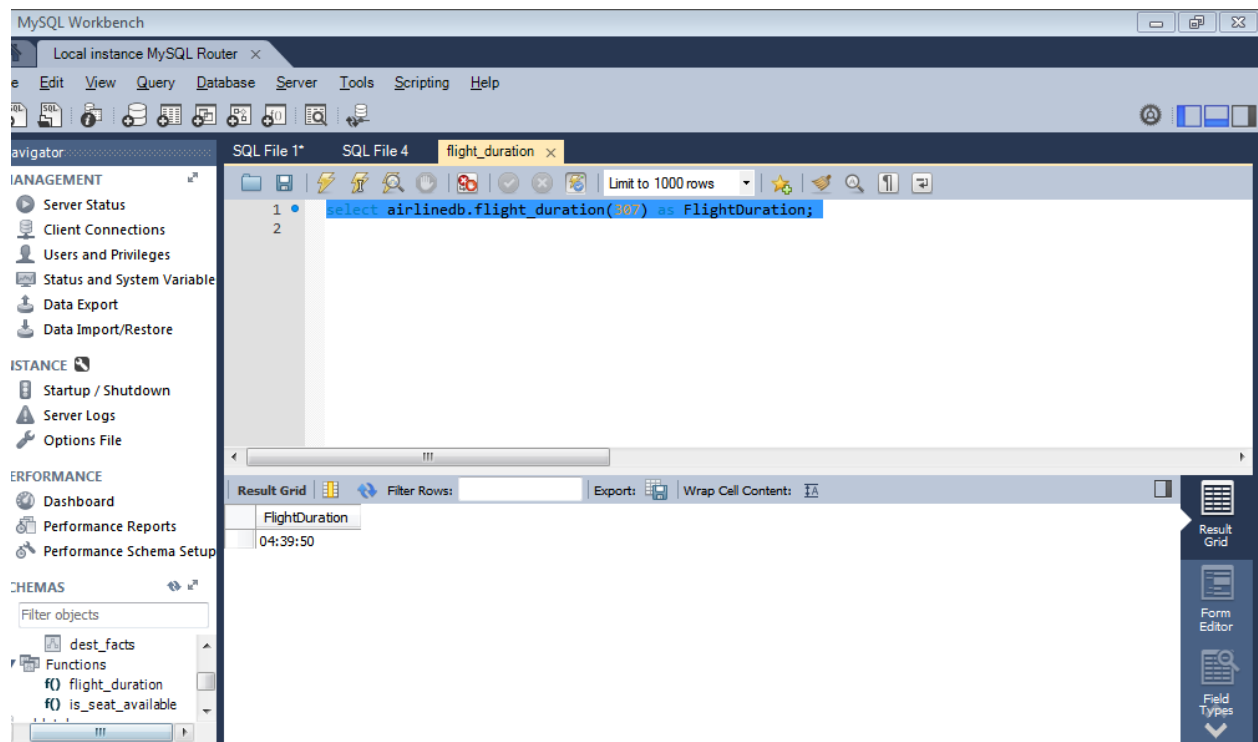
Execute Cancel

Output

Action Output

#	Time	Action	Message	Duration / Fetch
2	14:36:32	select <code>airlinedb.flight_duration(307)</code> LIMIT 0, 1000	1 row(s) returned	0.016 sec / 0.000 sec
3	14:37:14	select <code>airlinedb.flight_duration(307)</code> as FlightDuration LIMIT...	1 row(s) returned	0.000 sec / 0.000 sec

2:40 PM
12/12/2017



PROCEDURES :

1) To give a quick introduction / description to the passengers about the destination place they are heading to based on the Flight ID.

DELIMITER //

CREATE PROCEDURE dest_facts(IN idFlight int)

BEGIN

SELECT

FlightId,route.DestinationAirport,airport.AirportName,country.CountryName,country.Description from Flights

INNER JOIN route

ON RouteId=flights.Route_RouteId

INNER JOIN airport

ON route.DestinationAirport = airport.IataCode

LEFT JOIN country

on airport.Country_CountryCode = country.CountryCode

where flights.FlightId = idFlight;

END//

DELIMITER ;

CALL dest_facts(340);

MySQL Workbench

Local instance MySQL Router ...

File Edit View Query Database Server Tools Scripting Help

Navigator

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

PERFORMANCE

- Dashboard
- Performance Reports
- Performance Schema Setup

SCHEMAS

Filter objects

- country
- employee
- flightpassenger
- flights

Information

Table: **itinerary**

Columns:

- itineraryId** varchar(10) PK
- Price float UN

Object Info Session

query2

```
29 DELIMITER //
30 CREATE PROCEDURE dest_facts(idFlight int)
31 BEGIN
32
33 SELECT FlightId,route.DestinationAirport,airport.AirportName,country.CountryName,country.Descriptio
34 INNER JOIN route
35 ON RouteId=flights.Route_RouteId
36 INNER JOIN airport
37 ON route.DestinationAirport = airport.IataCode
38 LEFT JOIN country
39 on airport.Country_CountryCode = country.CountryCode
40 where flights.FlightId = idFlight;
41
42
43 END//
44 DELIMITER ;
45
46 CALL dest_facts(340);
```

Result Grid

FlightId	DestinationAirport	AirportName	CountryName	Description
340	IAH	George Bush Intercontinental	United States	Americans eat about 100 acres of pizza each da...

Result 3

Output

Action Output

#	Time	Action	Message	Duration / Fetch
18	02:42:35	SELECT FlightId,FirstName,LastName,SeatNo,Classna...	15 row(s) returned	0.015 sec / 0.000 sec

Query Completed

2:48 AM 12/13/2017

2) To show the analytics where revenue generated by each itinerary

A procedure "catg_revenue()" is called for this and also using Rollup to show the complete revenue after summation of each itinerary revenue.

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `catg_revenue`()  
BEGIN  
Select ItineraryId,Count,Revenue from revenue_thru_itinerary  
order by Revenue ASC;  
END
```

VIEW :

This procedure uses a view know as "revenue_thru_itinerary"

```
CREATE  
    ALGORITHM = UNDEFINED  
    DEFINER = `root`@`localhost`  
    SQL SECURITY DEFINER  
VIEW `revenue_thru_itinerary` AS  
    SELECT  
        `i`.`ItineraryId` AS `ItineraryId`,  
        COUNT(`i`.`ItineraryId`) AS `Count`,  
        SUM(`i`.`Price`) AS `Revenue`  
    FROM  
        (`flightpassenger` `fp`  
        JOIN `itinerary` `i` ON ((`fp`.`Itinerary_Id` = `i`.`ItineraryId`)))  
    GROUP BY `i`.`ItineraryId` WITH ROLLUP
```

MySQL Workbench

Local instance MySQL Router ...

File Edit View Query Database Server Tools Scripting Help

Navigator: flight_duration - Routine flights - Table flights query* catg_revenue - Routine revenue_thru_itinerary - View

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

PERFORMANCE

- Dashboard
- Performance Reports
- Performance Schema Setup

SCHEMAS

Filter objects

- revenue_thru_itinerary
- Stored Procedures
 - catg_revenue
 - dest_facts

Information

Procedure: catg_revenue

Object Info Session

```
62 END//
63 DELIMITER ;
64
65 CALL dest_facts(340);
66
67
68
69 #Procedure 2 to calculate revenue through itinerary
70 DELIMITER //
71 CREATE DEFINER='root'@'localhost' PROCEDURE `catg_revenue`()
72 BEGIN
73   Select ItineraryId,Count,Revenue from revenue_thru_itinerary
74   order by Revenue ASC;
75 END//
76 DELIMITER ;
77
78 CALL catg_revenue();
79
```

Result Grid

ItineraryId	Count	Revenue
BK24580	2	844
ZY87787	4	844
DR45927	3	1092
YE23002	2	1914
CF78099	3	2337
NULL	20	8567

Result 9

Output

Action Output

#	Time	Action	Message	Duration / Fetch
23	02:49:33	SELECT FlightID from Flights INNER JOIN route ON R...	2 row(s) returned	0.031 sec / 0.000 sec

Query Completed

2:56 AM 12/13/2017

VIEWS :

1) To calculate revenue through itinerary

```
CREATE VIEW `revenue_thru_itinerary` AS
```

```
SELECT
```

```
  `i`.`ItineraryId` AS `ItineraryId`,
```

```
  COUNT(`i`.`ItineraryId`) AS `Count`,
```

```
  SUM(`i`.`Price`) AS `Revenue`
```

```
FROM
```

```
  (`flightpassenger` `fp`
```

```
  JOIN `itinerary` `i` ON ((`fp`.`Itinerary_Id` = `i`.`ItineraryId`)))
```

```
GROUP BY `i`.`ItineraryId` WITH ROLLUP;
```

```
SELECT * from revenue_thru_itinerary;
```

2) To display all the flights to or from a place and also ordering the results through arrival/departure time.

```
CREATE VIEW flights_to_or_from AS
```

```
SELECT flights.FlightId,route.Routeld, route.OriginAirport, route.DestinationAirport  
,flights.ArrivalTime, flights.DepartureTime, flightstatus.Status,ar.Manufacturer,ar.Model
```

```
FROM Flights
```

```
INNER JOIN route
```

```
ON flights.Route_Routeld = route.Routeld
```

```
INNER JOIN flightstatus
```

```
ON flights.FlightStatus_FlightStatusId= flightstatus.FlightStatusId
```

```
INNER JOIN aircraft ar
```

```
ON flights.Aircraft_AircraftId = ar.AircraftId
```

```
INNER JOIN airport a
```

```
ON route.OriginAirport = a.IataCode
```

```
AND route.DestinationAirport = a.IataCode
```

```
IS NOT NULL;
```

```
SELECT * from flights_to_or_from
```

```
where OriginAirport = 'BOS'
```

```
ORDER BY ArrivalTime DESC;
```

```
SELECT * from flights_to_or_from
```

```
where DestinationAirport = 'ORD'
```

```
ORDER BY ArrivalTime ASC;
```


MySQL Workbench

Local instance MySQL Router ...

File Edit View Query Database Server Tools Scripting Help

Navigator

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

PERFORMANCE

- Dashboard
- Performance Reports
- Performance Schema Setup

SCHEMAS

Filter objects

- warning
- Views
 - flights_to_or_from
 - revenue_thru_itinerary

Information

View: flights_to_or_from

Columns:

- FlightId int(10) UN
- RouteId int(10) UN
- OriginAirport varchar(3)

Object Info Session

SQL File 3* flightpassenger flight_duration - Routine flights - Table flights query x

Limit to 1000 rows

```
106 ON flights.FlightStatus_FlightStatusId= flightstatus.FlightStatusId
107 INNER JOIN aircraft ar
108 ON flights.Aircraft_AircraftId = ar.AircraftId
109 INNER JOIN airport a
110 ON route.OriginAirport = a.IataCode
111 AND route.DestinationAirport = a.IataCode
112 IS NOT NULL;
113
114
115 • SELECT * from flights_to_or_from
116 where OriginAirport = 'BOS'
117 ORDER BY ArrivalTime DESC;
118
119 • SELECT * from flights_to_or_from
120 where DestinationAirport = 'ORD'
121 ORDER BY ArrivalTime ASC;
122
```

Result Grid

FlightId	RouteId	OriginAirport	DestinationAirport	ArrivalTime	DepartureTime	Status	Manufacturer	Model
327	2	ATL	ORD	03:40:00	01:00:00	Cancelled	Cessna	Titer
358	5	RSW	ORD	10:59:00	06:06:00	Cancelled	Cessna	Titer
364	2	ATL	ORD	18:18:00	15:13:00	Delayed	Airbus	A380

flights_to_or_from11 x

Output

Action Output

#	Time	Action	Message	Duration / Fetch
25	03:12:02	SELECT * from flights_to_or_from where OriginAirport = ...	1 row(s) returned	0.062 sec / 0.000 sec

Query Completed

3:12 AM 12/13/2017

QUERIES :

1) To find passengers with a specific itinerary

The screenshot shows the MySQL Workbench interface. The SQL Editor contains the following query:

```
1 use airlinedb;
2
3 select person.FirstName as FirstName, Passenger_Person_PersonId as ID, Itinerary_ItineraryId as Itiner
4 from flightpassenger
5 inner join person
6 on person.PersonId= Passenger_Person_PersonId
7 inner join itinerary
8 on flightpassenger.Itinerary_ItineraryId = itinerary.ItineraryId
9 where itinerary.ItineraryId='YH65424'
10 ;
11
12
```

The Result Grid shows the following data:

FirstName	ID	Itinerary	Price	SeatNo
Claudina	15	YH65424	410	1A
Nicolai	12	YH65424	410	3E

The Information tab shows the structure of the **itinerary** table:

Column	Type	PK
ItineraryId	varchar(10)	PK
Price	float	UN

The Output tab shows the execution log:

#	Time	Action	Message	Duration / Fetch
2	01:54:39	use airlinedb	0 row(s) affected	0.000 sec
3	01:54:43	select person.FirstName as FirstName, Passenger_Per...	2 row(s) returned	0.000 sec / 0.000 sec

The status bar at the bottom indicates "Query Completed" and the system time is 1:58 AM on 12/13/2017.

2) To display all passengers and their seat class with seat number

The screenshot shows the MySQL Workbench interface. The main window displays a SQL query in the 'SQL File 3*' tab. The query is as follows:

```
13 GROUP BY FirstName;
14
15 • select * from classeat;
16
17 #this is to show all persons and their seat class
18 • SELECT FlightId,FirstName,LastName,SeatNo,Classname,seaclassaircraft.ClassSeat_SeatClassId from f
19 left join flightpassenger
20 on FlightId=flightpassenger.Flights_FlightId
21 inner join person
22 on Passenger_Person_PersonId = PersonId
23 inner join aircraft
24 on Aircraft_AircraftId=aircraft.AircraftId
25 right join seaclassaircraft
26 on seaclassaircraft.SeatNum = SeatNo
27 inner join classeat
28 on seaclassaircraft.ClassSeat_SeatClassId = SeatClassId
29 group by firstname
```

Below the query editor, the 'Result Grid' shows the results of the query. The table has columns: FlightId, FirstName, LastName, SeatNo, Classname, and ClassSeat_SeatClassId. The results are as follows:

FlightId	FirstName	LastName	SeatNo	Classname	ClassSeat_SeatClassId
350	Maisev	McOuorkel	2B	Economv Class	4
308	Mile	Vondra	5F	First Class	2
364	Nicolai	Cruikshanks	3E	Travel Class	1
300	Pollv	McGinnis	3E	Travel Class	1
340	Rancell	Vala	3D	Travel Class	1
337	Wilburt	Soillane	2C	Economv Class	4

The 'Information' pane on the left shows the table 'itinerary' with columns 'itineraryId' (varchar(10) PK) and 'Price' (float UN). The 'Output' pane at the bottom shows the query execution details: 'SELECT FlightId,FirstName,LastName,SeatNo,Classna...' and '80 row(s) returned'.

3) To display the number of bookings made by a specific employee

The screenshot shows the MySQL Workbench interface. The left sidebar contains the Navigator, Information, and Schemas panels. The main editor displays a SQL query in the 'query3' tab. The query is as follows:

```
8 SELECT flights.aircraft_aircraftId FROM flightpassenger
9 INNER JOIN flights
10 ON flightpassenger.flights_flightid = flights.flightid
11 WHERE flights_flightid = 307
12 GROUP BY flights.route_routeId;

15 SELECT Employee_Person_PersonId from flightpassenger;

17 SELECT person.FirstName as 'Employee First Name', person.LastName 'Employee Last Name', count(Emplo
18 INNER JOIN Employee
19 ON Employee_Person_PersonId = Person_PersonId
20 INNER JOIN person
21 ON PersonId = Employee_Person_PersonId
22 GROUP BY (Employee_Person_PersonId);
23
24
```

The 'Result Grid' shows the following data:

Employee First Name	Employee Last Name	Bookings Made
Mac	Bartocci	7
Bvrann	Brouham	7
Wilburt	Sollane	6

The 'Information' panel shows the view 'flights_to_or_from' with columns: FlightId (int(10) UN), RouteId (int(10) UN), and OriginAirport (varchar(3)). The 'Output' panel shows the message: 'SELECT * from flights_to_or_from where DestinationAir... 3 row(s) returned' with a duration of 0.000 sec / 0.000 sec.

4) To display the list of employees who are about to get retired

The screenshot shows the MySQL Workbench interface. The left sidebar contains the Navigator, Information, and Schemas panels. The main editor displays a SQL query in a file named 'query3'. The query is as follows:

```
GROUP BY(Employee_Person_PersonId);

#4 Show list of employees about to retire
drop table retirement_table;

create table if not exists retirement_table (firstname varchar(25), lastname varchar(25), dob date);

INSERT INTO retirement_table
SELECT firstname,lastname,dob,YEAR(NOW())-YEAR(DOB),employee.Designation from person
INNER JOIN employee
where PersonId= employee.Person_PersonId
AND (YEAR(NOW())-YEAR(DOB))>=64;

select * from retirement_table;
```

The 'Result Grid' shows the output of the query, displaying a single row of data:

firstname	lastname	dob	age	desingation
Mac	Bartocci	1953-01-12	64	Support Staff

The 'Output' panel at the bottom shows the execution details of the query, indicating that 3 rows were returned and the execution time was 0.000 sec.

5) To know which aircraft is being preferred by more number of passengers

The screenshot shows the MySQL Workbench interface. The left sidebar contains the 'Navigator' pane with sections for 'MANAGEMENT', 'INSTANCE', 'PERFORMANCE', and 'SCHEMAS'. The 'SCHEMAS' section is expanded, showing a list of objects including 'Views', 'Stored Procedures', 'catg_revenue', and 'dest_facts'. The main editor window displays a SQL query in the 'SQL File 3*' tab. The query is as follows:

```
10 INNER JOIN route
11 ON RouteId=flights.Route_RouteId
12 INNER JOIN airport
13 ON route.DestinationAirport = airport.IataCode
14 where flights.FlightId = '300';
15
16
17
18 • Select count(flightpassenger.Passenger_Person_PersonId) as 'No of Passenger',aircraft.Manufacturer
19 inner join flights
20 on flightpassenger.Flights_FlightId = FlightId
21 inner join aircraft
22 on flights.Aircraft_AircraftId = AircraftId
23 GROUP BY (aircraft.Manufacturer)
24 ORDER BY (Passenger_Person_PersonId) DESC;
25
26
27
```

Below the query editor, the 'Result Grid' shows the results of the query. The grid has two columns: 'No of Passenger' and 'Manufacturer'. The results are as follows:

No of Passenger	Manufacturer
11	Embraer
4	Cessna
4	Airbus
1	Boeing

The 'Output' pane at the bottom shows the execution details for the query. It indicates that the query was executed at 16:59:13, returned 4 rows, and took 0.000 seconds to execute.