PUNJAB UNIVERSITY COLLEGE OF INFORMATION AND TECHNOLOGY

DATABASE PROJECT

AIRLINE MANAGEMENT SYSTEM

Group Members:

- **BSEF21M057 Muhammad Safee**
- **BSEF21M039 Zohaib Ali**
- **BSEF21M058 Muhammad Naseem**

Instructor Name: Sir Asif Sohail

Table of Contents

Introduction:

The Airline Management System is designed to make airline operations smoother and more passenger-friendly. The Airline Management System project represents a considerable effort in creating an efficient and user-friendly solution for the management of airline operations. This system covers essential functions like creating flight schedules, handling passenger bookings, ticketing, and generating reports. It aims to significantly enhance the efficiency of airline operations, improve the overall passenger experience, and provide airline staff with a robust platform to manage flights effectively.

We will try to craft a well-structured database that stores critical data like flights, passenger details, booking, airports, payments, flights, route, etc.

We will ensure data is neatly organized in tables, linked together for easy access, and secured with unique keys to ensure no duplicate entries exist.

These keys, such as passenger IDs and flight numbers, uniquely identify records and prevent data redundancy. This design will support efficient airline operations and a seamless passenger experience.

Entity Relation Diagram of System

The following Entity-Relationship Diagram (ERD) represents the relationships of different entities in this database system with respect to their attributes and primary as well as foreign keys. It also describes this system's 1:1, 1:M, and M:N relationships.

ERD Model:

The entities for the Airline Management System (AMS) are as follows:

AMS (Passenger, Flight, Aircraft, Booking, Payment, Airport, Route, Baggage)

Attributes of the Entities:

Passenger (PassengerID, PassengerName, ContactInfo)

Flight (FlightID, DepartureAirportID, ArrivalAirportID, DepartureTime, ArrivalTime, AircraftID)

Aircraft (AircraftID, Model, SeatingCapacity)

Booking (BookingID, PassengerID, FlightID, SeatNo, BookingStatus, Fare, ClassType)

Payment (PaymentID, BookingID, PaymentDate, PaymentMethod, Amount)

Airport (AirportID, AirportName, Location)

Route (RouteID, OriginAirportID, DestinationAirportID, Distance)

Baggage (BaggageID, PassengerID, FlightID, Weight, BaggageType)

Explanation of Attributes:

Passenger:

Passenger (PassengerID, PassengerName, ContactInfo)

PassengerID: Unique identifier for each passenger. (Primary Key)

PassengerName: Name of the passenger.

ContactInfo: Contact information of the passenger (email).

Flight:

Flight (FlightID, DepartureAirportID (Airport), ArrivalAirportID (Airport), DepartureTime, ArrivalTime, Duration, AircraftID (Aircraft))

FlightID: Unique identifier for each flight. (Primary Key)

DepartureAirportID: ID for the departure airport. (Foreign Key referencing Airport)

ArrivalAirportID: ID for the arrival airport. (Foreign Key referencing Airport)

DepartureTime: Time of departure for the flight.

ArrivalTime: Time of arrival for the flight.

AircraftID: Unique identifier for the aircraft assigned to the flight. (Foreign Key referencing Aircraft)

Aircraft:

Aircraft (AircraftID, Model, SeatingCapacity, Manufacturer)

AircraftID: Unique identifier for each aircraft. (Primary Key)

Model: Model of the aircraft.

SeatingCapacity: Total seating capacity of the aircraft.

Booking:

Booking (BookingID, PassengerID (passenger), FlightNo (flight), SeatNo, BookingStatus, Fare, ClassType)

BookingID: Unique identifier for each booking. (Primary Key)

PassengerID: Reference to the passenger associated with the booking. (Foreign Key referencing Passenger)

FlightID: Reference to the flight booked. (Foreign Key referencing Flight)

SeatNo: Assigned seat number for the booking.

BookingStatus: Status of the booking (confirmed, pending, canceled, etc.).

Fare: Fare amount for the booking.

ClassType: Class type of the booking (economy, business, first class, etc.).

Payment:

Payment (PaymentID, BookingID (booking), PaymentDate, PaymentMethod, Amount)

PaymentID: Unique identifier for each payment transaction. (Primary Key)

BookingID: Reference to the booking for which the payment was made. (Foreign Key referencing Booking)

PaymentDate: Date of the payment transaction.

PaymentMethod: Method used for payment (credit card, cash, etc.).

Amount: Amount paid for the booking.

Airport:

Airport (AirportID, AirportName, Location)

AirportID: Unique code or identifier for each airport. (Primary Key)

AirportName: Name of the airport.

Location: Geographic location of the airport.

Route:

Route (RouteID, OriginAirportID (airport), DestinationAirportID (airport), Distance, AvgDuration)

RouteID: Unique identifier for each route. (Primary Key)

OriginAirportID: Reference to the starting point airport. (Foreign Key referencing Airport)

DestinationAirportID: Reference to the destination airport. (Foreign Key referencing Airport)

Distance: Distance between the origin and destination.

Baggage:

Baggage (BaggageID, PassengerID (passenger), FlightID (flight), Weight, BaggageType)

BaggageID: Unique identifier for each baggage. (Primary Key)

PassengerID: Reference to the passenger associated with the baggage. (Foreign Key referencing Passenger)

FlightID: Reference to the flight associated with the baggage. (Foreign Key referencing Flight)

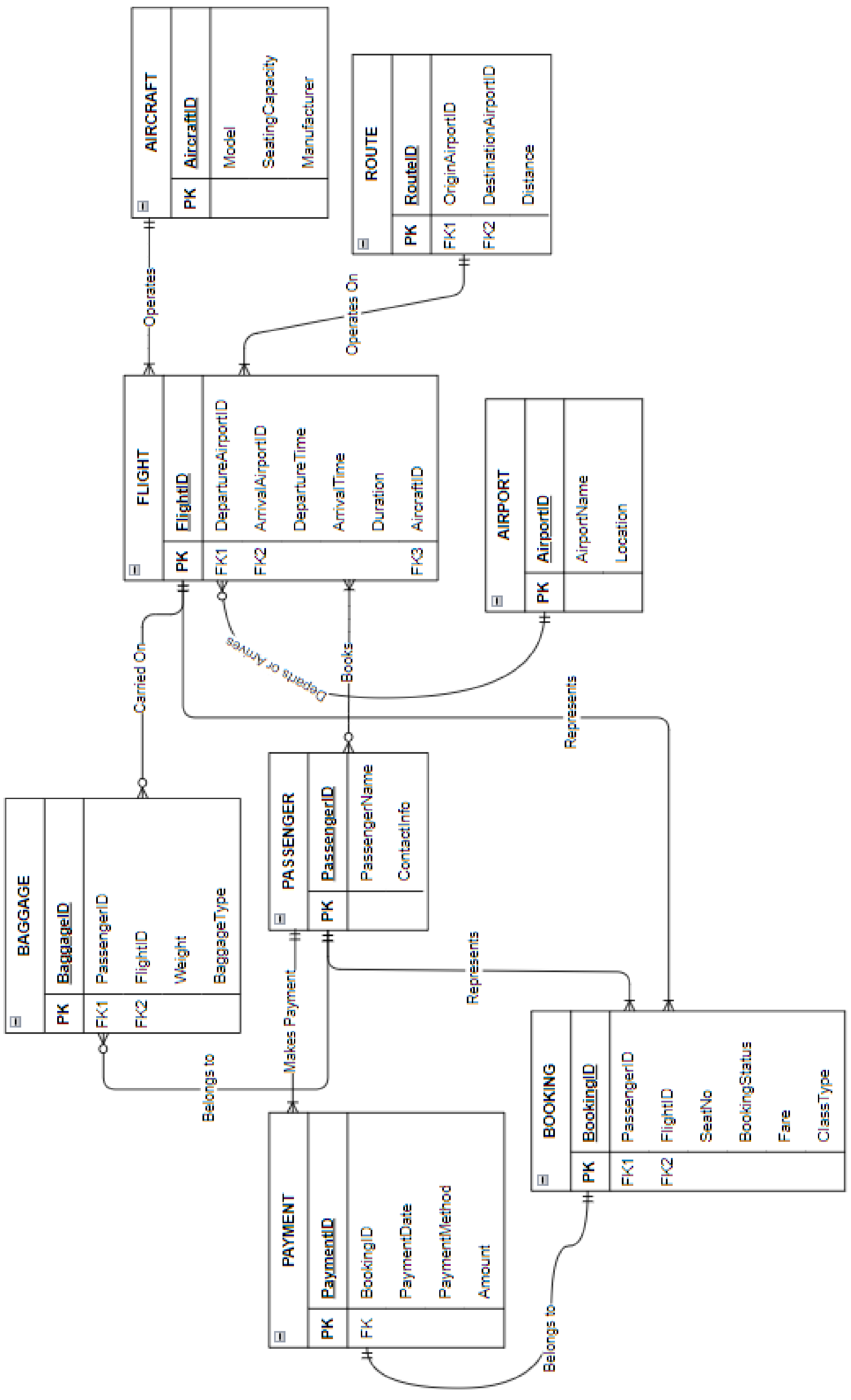
Weight: Weight of the baggage.

BaggageType: Type of baggage (checked, carry-on, special items, etc.).

Connectivity Table:

ENTITY	RELATIONSHIP	CONNECTIVITY	ENTITY
Passenger	Books	M:N	Flight
Passenger	MakesPayments	1:M	Payment
Flight	Operates	M:1	Aircraft
Flight	OperatesOnRoute	M:1	Route
Booking	Represents	M:1	Passenger
Booking	Represents	M:1	Flight
Payment	PaymentForBooking	1:1	Booking
Airport	Departure/Arrival	1:M	Flight
Route	FliesThrough	1:M	Flight
Baggage	BelongsTo	M:1	Passenger
Baggage	CarriedOn	M:1	Flight

ERD DIAGRAM:


AIRCRAFT
PK AircraftID
Model
SeatingCapacity
Manufacturer
ROUTE
PK RouteID
FK1 OriginAirportID
FK2 DestinationAirportID
Distance
FLIGHT
PK FlightID
FK1 DepartureAirportID
FK2 ArrivalAirportID
DepartureTime
ArrivalTime
Duration
FK3 AircraftID
AIRPORT
PK AirportID
AirportName
Location
PASSENGER
PK PassengerID
PassengerName
ContactInfo
BAGGAGE
PK BaggageID
FK1 PassengerID
FK2 FlightID
Weight
BaggageType
BOOKING
PK BookingID
FK1 PassengerID
FK2 FlightID
SeatNo
BookingStatus
Fare
ClassType
PAYMENT
PK PaymentID
FK BookingID
PaymentDate
PaymentMethod
Amount
Operates
Operates On
Departs or Arrives
Books
Carried On
Represents
Represents
Makes Payment
Belongs to
Belongs to

Relational Schema (with Normalization):

AMS (PassengerID, PassengerName, ContactInfo, FlightID, DepartureTime, ArrivalTime, AircraftID, Model, SeatingCapacity, Manufacturer, BookingID, SeatNo, BookingStatus, Fare, ClassType, PaymentID, PaymentDate, PaymentMethod, Amount, AirportID, AirportName, Location, RouteID, Distance, BaggageID, Weight, BaggageType)

Normalization:

FIRST NORMAL FORM(1NF):

To convert it into 1NF, we should remove repeating groups and ensure that each attribute contains only atomic (indivisible) values.

R11(PassengerID, PassengerName, ContactInfo, BookingID, SeatNo, BookingStatus, Fare, ClassType, PaymentID, PaymentDate, PaymentMethod, Amount, BaggageID, Weight, BaggageType)

R12(FlightID, DepartureTime, ArrivalTime, AircraftID, Model, SeatingCapacity, Manufacturer, AirportID, AirportName, Location, RouteID, Distance)

SECOND NORMAL FORM(2NF):

A table is in 2nd Normal Form (2NF) if it must be in 1st Normal Form (1NF), meaning that each attribute must contain only atomic values and there must not be any partial dependencies.

R11(PassengerID, PassengerName, ContactInfo)

R21(BookingID, SeatNo, BookingStatus, Fare, ClassType, PaymentID, PaymentDate, PaymentMethod, Amount, BaggageID, Weight, BaggageType)

R12(FlightID, DepartureTime, ArrivalTime, AirportID, AirportName, Location, RouteID, Distance)

R22(AircraftID, Model, SeatingCapacity, Manufacturer)

THIRD NORMAL FORM(3NF):

For a table to be in third normal form, there must not exist any transitive dependency in the relation which states that non-prime attributes should not depend on other non-prime attributes within the same table.

R11(PassengerID, PassengerName, ContactInfo)

R21(BookingID, PassengerID, FlightID, SeatNo, BookingStatus, Fare, ClassType)

R31(PaymentID, BookingID, PaymentDate, PaymentMethod, Amount)

R32(BaggageID, PassengerID, FlightID, Weight, BaggageType)

R12(FlightID, AirportID, DepartureTime, ArrivalTime, AircraftID)

R22(AircraftID, Model, SeatingCapacity, Manufacturer)

R32(AirportID, AirportName, Location)

R42(RouteID, AirportID, Distance)

Now naming the above tables:

Passenger (PassengerID, PassengerName, ContactInfo)

Flight (FlightID, DepartureAirportID, ArrivalAirportID, DepartureTime, ArrivalTime, AircraftID)

Aircraft (AircraftID, Model, SeatingCapacity)

Booking (BookingID, PassengerID, FlightID, SeatNo, BookingStatus, Fare, ClassType)

Payment (PaymentID, BookingID, PaymentDate, PaymentMethod, Amount)

Airport (AirportID, AirportName, Location)

Route (RouteID, OriginAirportID, DestinationAirportID, Distance)

Baggage (BaggageID, PassengerID, FlightID, Weight, BaggageType)

All relational schemas within the system witness that there doesn't exist any such relation within this system, so all the tables are already in 3NF.

Relations Description:

Table Name: Passenger

Attribute	Data Type	Size	Constraints
PassengerID	CHAR	4	Primary Key
PassengerName	VARCHAR2	50	Not Null
ContactInfo	VARCHAR2	50	Not Null

Table Name: Flight

Attribute	Data Type	Size	Constraints
FlightID	CHAR	4	Primary Key
DepartureAirportID	CHAR	4	Foreign Key(Airport)
ArrivalAirportID	CHAR	4	Foreign Key(Airport)
DepartureTime	DATE		
ArrivalTime	DATE		
AircraftID	CHAR	4	Foreign Key(Aircraft)

Table Name: Aircraft

Attribute	Data Type	Size	Constraints
AircraftID	CHAR	4	Primary Key
Model	VARCHAR2	20	Not Null
SeatingCapacity	NUMBER	3	Check (<=100)

Table Name: Booking

Attribute	Data Type	Size	Constraints
BookingID	CHAR	4	Primary Key
PassengerID	CHAR	4	Foreign Key(Passenger)
FlightID	CHAR	4	Foreign Key(Flight)
SeatNo	NUMBER	5	Not Null
BookingStatus	VARCHAR2	20	Can be(Confirmed, Pending, Cancelled)
Fare	NUMBER	(8,2)	Not Null
ClassType	VARCHAR2	20	Can be(Business, Economy, First class)

Table Name: Payment

Attribute	Data Type	Size	Constraints
PaymentID	CHAR	4	Primary Key
BookingID	CHAR	4	Foreign Key(Booking)
PaymentDate	DATE		Not Null
PaymentMethod	VARCHAR2	20	Can be(Credit, Debit, Cash)
Amount	NUMBER	(8,2)	Not Null

Table Name: Airport

Attribute	Data Type	Size	Constraints
AirportID	CHAR	4	Primary Key
AirportName	VARCHAR2	50	Not Null
Location	VARCHAR2	50	Not Null

Table Name: Route

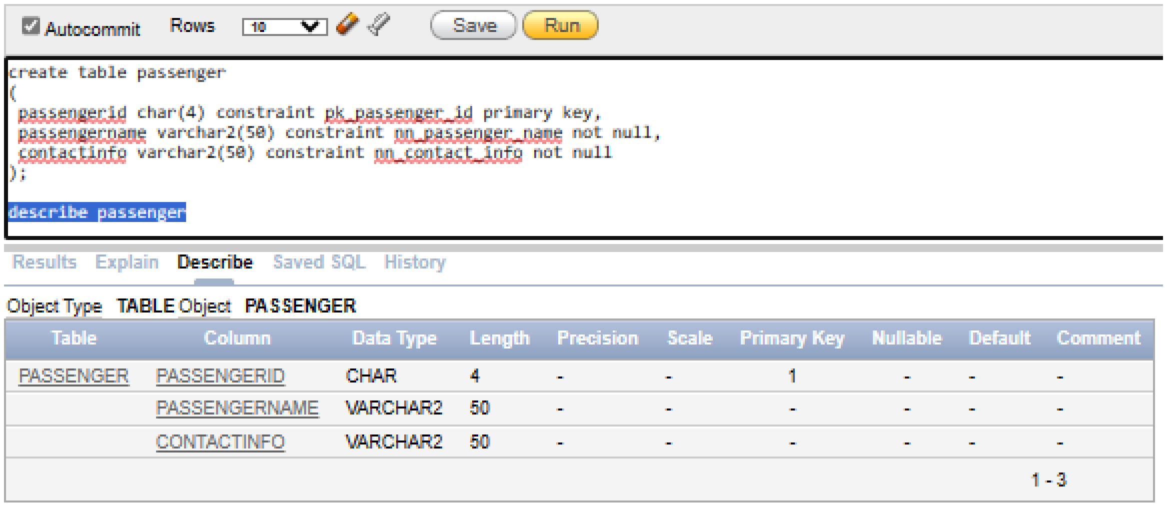
Attribute	Data Type	Size	Constraints
RouteID	CHAR	4	Primary Key
OriginAirportID	CHAR	4	Foreign Key(Airport)
DestinationAirportID	CHAR	4	Foreign Key(Airport)
Distance	NUMBER	(8,2)	Not Null

Table Name: Baggage

Attribute	Data Type	Size	Constraints
BaggageID	CHAR	4	Primary Key
PassengerID	CHAR	4	Foreign Key(Passenger)
FlightID	CHAR	4	Foreign Key(Flight)
Weight	NUMBER	(4,1)	Not greater than 20kg
BaggageType	VARCHAR2	30	Can be (Checked, Carry-on, special items)

SQL Statements for Table Creation:

Passenger:



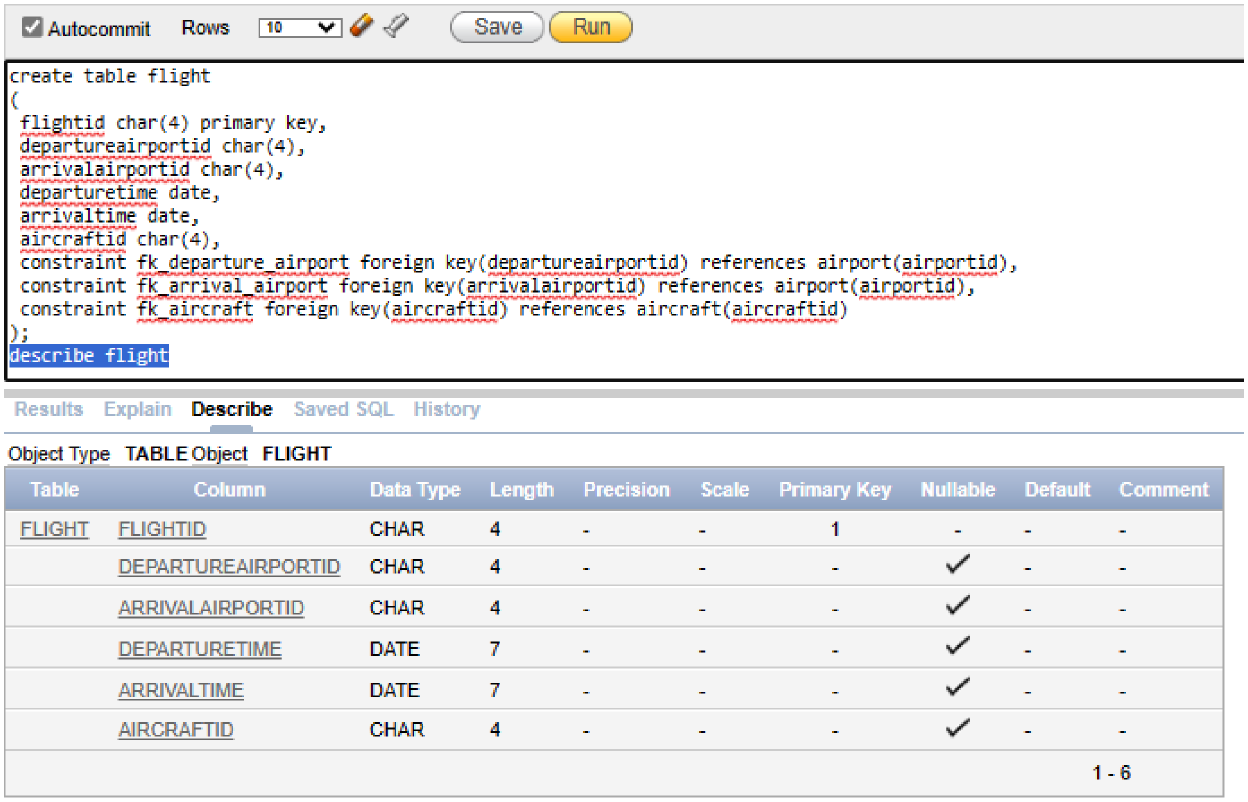
```
create table passenger
(
 passengerid char(4) constraint pk_passenger_id primary key,
 passengername varchar2(50) constraint nn_passenger_name not null,
 contactinfo varchar2(50) constraint nn_contact_info not null
);

describe passenger
```

Object Type TABLE Object PASSENGER

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
PASSENGER	PASSENGERID	CHAR	4	-	-	1	-	-	-
	PASSENGERNAME	VARCHAR2	50	-	-	-	-	-	-
	CONTACTINFO	VARCHAR2	50	-	-	-	-	-	-

1 - 3

Flight:



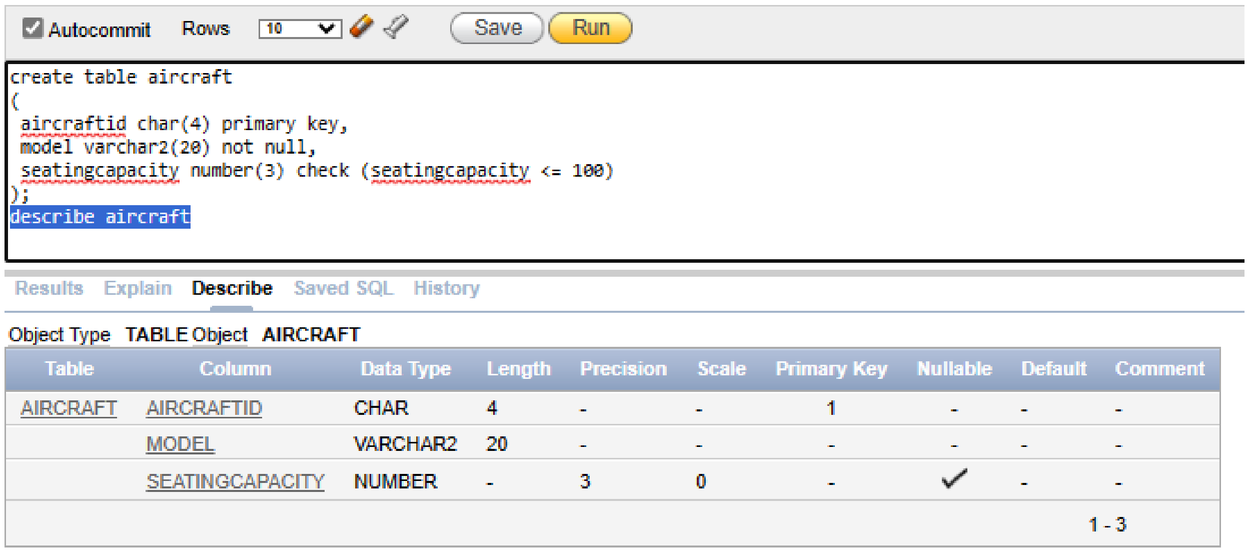
```
create table flight
(
 flightid char(4) primary key,
 departureairportid char(4),
 arrivalairportid char(4),
 departuretime date,
 arrivaltime date,
 aircraftid char(4),
 constraint fk_departure_airport foreign key(departureairportid) references airport(airportid),
 constraint fk_arrival_airport foreign key(arrivalairportid) references airport(airportid),
 constraint fk_aircraft foreign key(aircraftid) references aircraft(aircraftid)
);
describe flight
```

Object Type TABLE Object FLIGHT

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
FLIGHT	FLIGHTID	CHAR	4	-	-	1	-	-	-
	DEPARTUREAIRPORTID	CHAR	4	-	-	-	✓	-	-
	ARRIVALAIRPORTID	CHAR	4	-	-	-	✓	-	-
	DEPARTURETIME	DATE	7	-	-	-	✓	-	-
	ARRIVALTIME	DATE	7	-	-	-	✓	-	-
	AIRCRAFTID	CHAR	4	-	-	-	✓	-	-

1 - 6

Aircraft:



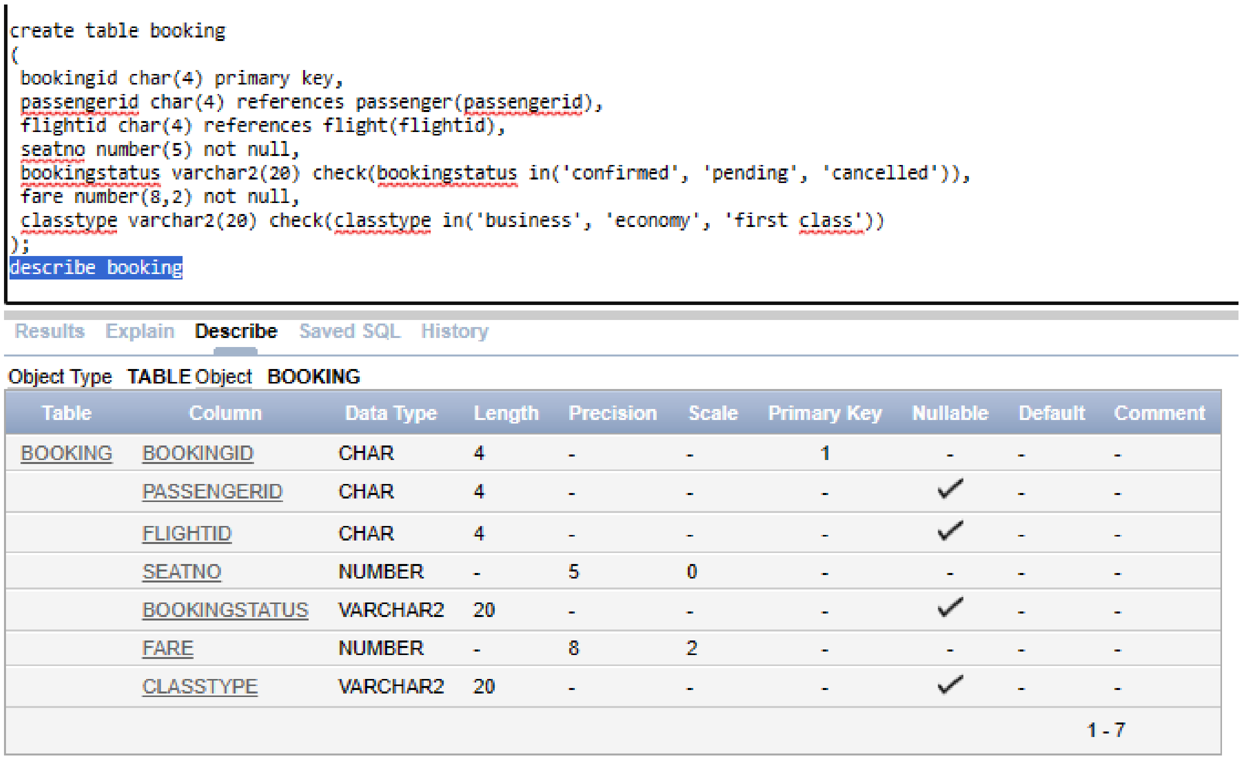
```
create table aircraft
(
 aircraftid char(4) primary key,
 model varchar2(20) not null,
 seatingcapacity number(3) check (seatingcapacity <= 100)
);
describe aircraft
```

Object Type TABLE Object AIRCRAFT

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
AIRCRAFT	AIRCRAFTID	CHAR	4	-	-	1	-	-	-
	MODEL	VARCHAR2	20	-	-	-	-	-	-
	SEATINGCAPACITY	NUMBER	-	3	0	-	✓	-	-

1 - 3

Booking:



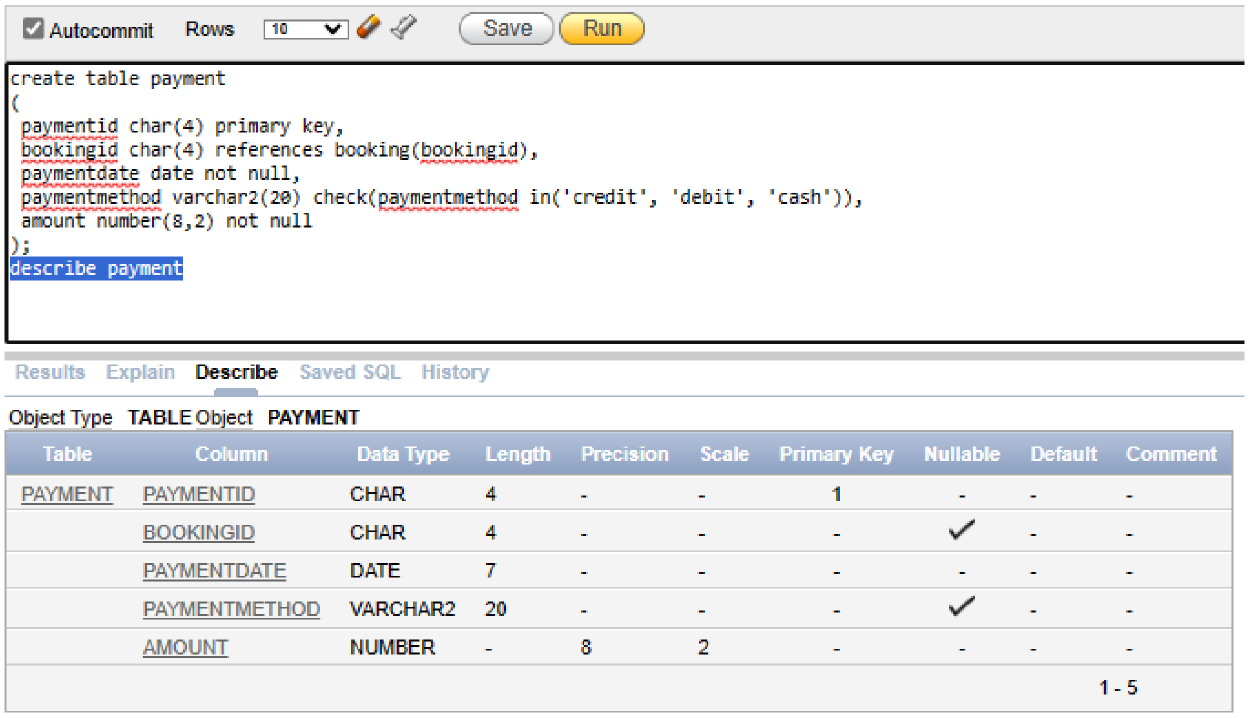
```
create table booking
(
 bookingid char(4) primary key,
 passengerid char(4) references passenger(passengerid),
 flightid char(4) references flight(flightid),
 seatno number(5) not null,
 bookingstatus varchar2(20) check(bookingstatus in('confirmed', 'pending', 'cancelled')),
 fare number(8,2) not null,
 classtype varchar2(20) check(classtype in('business', 'economy', 'first class'))
);
describe booking
```

Object Type TABLE Object BOOKING

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
BOOKING	BOOKINGID	CHAR	4	-	-	1	-	-	-
	PASSENGERID	CHAR	4	-	-	-	✓	-	-
	FLIGHTID	CHAR	4	-	-	-	✓	-	-
	SEATNO	NUMBER	-	5	0	-	-	-	-
	BOOKINGSTATUS	VARCHAR2	20	-	-	-	✓	-	-
	FARE	NUMBER	-	8	2	-	-	-	-
	CLASSTYPE	VARCHAR2	20	-	-	-	✓	-	-

1 - 7

Payment:



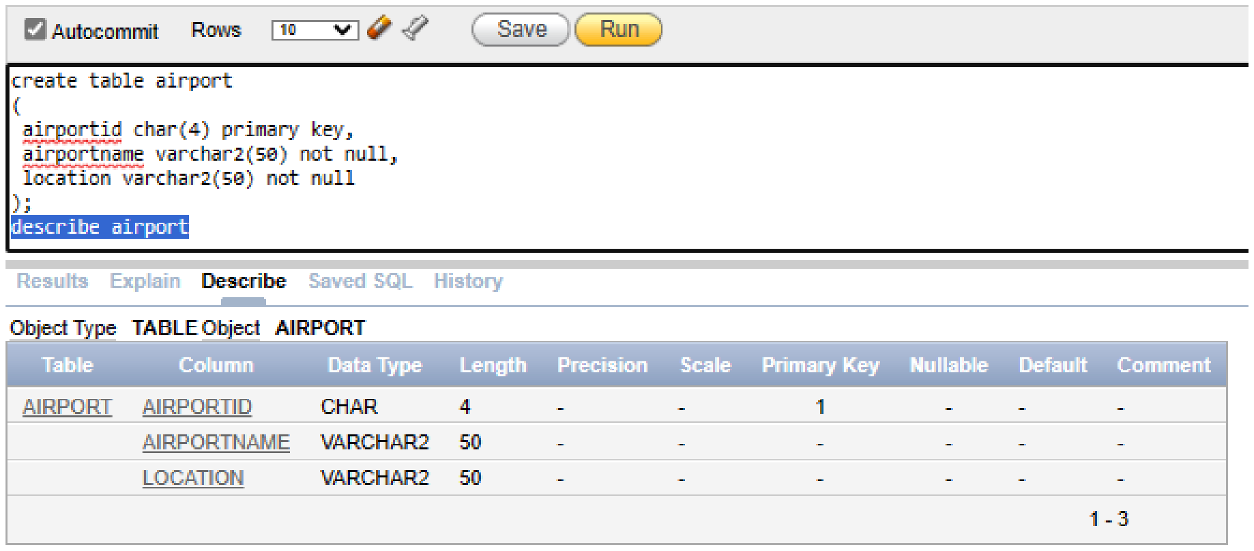
```
create table payment
(
 paymentid char(4) primary key,
 bookingid char(4) references booking(bookingid),
 paymentdate date not null,
 paymentmethod varchar2(20) check(paymentmethod in('credit', 'debit', 'cash')),
 amount number(8,2) not null
);
describe payment
```

Object Type TABLE Object PAYMENT

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
PAYMENT	PAYMENTID	CHAR	4	-	-	1	-	-	-
	BOOKINGID	CHAR	4	-	-	-	✓	-	-
	PAYMENTDATE	DATE	7	-	-	-	-	-	-
	PAYMENTMETHOD	VARCHAR2	20	-	-	-	✓	-	-
	AMOUNT	NUMBER	-	8	2	-	-	-	-

1 - 5

Airport:



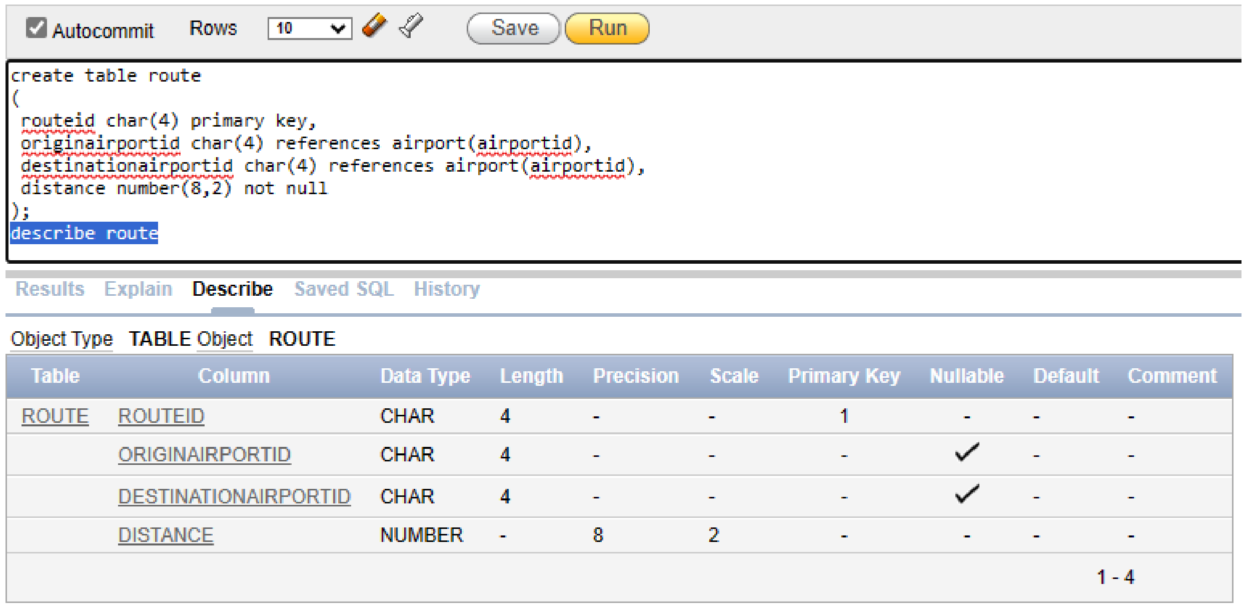
```
create table airport
(
 airportid char(4) primary key,
 airportname varchar2(50) not null,
 location varchar2(50) not null
);
describe airport
```

Object Type TABLE Object AIRPORT

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
AIRPORT	AIRPORTID	CHAR	4	-	-	1	-	-	-
	AIRPORTNAME	VARCHAR2	50	-	-	-	-	-	-
	LOCATION	VARCHAR2	50	-	-	-	-	-	-

1 - 3

Route:



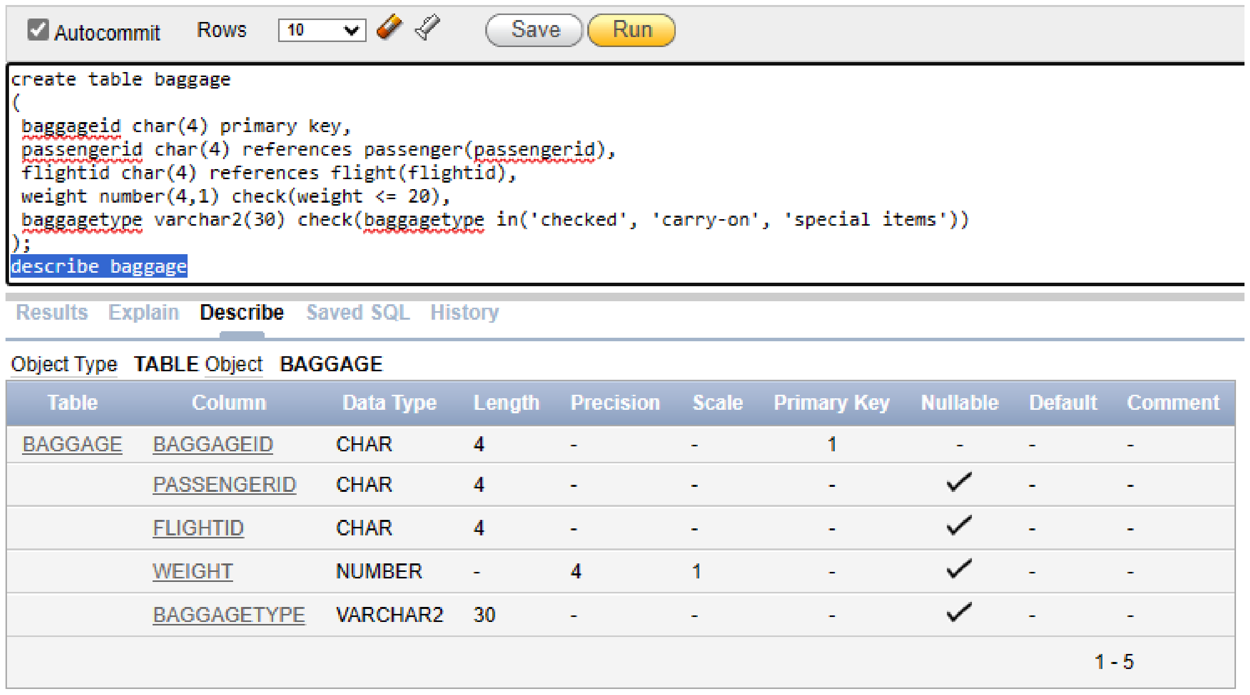
```
create table route
(
 routeid char(4) primary key,
 originairportid char(4) references airport(airportid),
 destinationairportid char(4) references airport(airportid),
 distance number(8,2) not null
);
describe route
```

Object Type TABLE Object ROUTE

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
ROUTE	ROUTEID	CHAR	4	-	-	1	-	-	-
	ORIGINAIRPORTID	CHAR	4	-	-	-	✓	-	-
	DESTINATIONAIRPORTID	CHAR	4	-	-	-	✓	-	-
	DISTANCE	NUMBER	-	8	2	-	-	-	-

1 - 4

Baggage:



```
create table baggage
(
 baggageid char(4) primary key,
 passengerid char(4) references passenger(passengerid),
 flightid char(4) references flight(flightid),
 weight number(4,1) check(weight <= 20),
 baggagetype varchar2(30) check(baggagetype in('checked', 'carry-on', 'special items'))
);
describe baggage
```

Object Type TABLE Object BAGGAGE

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
BAGGAGE	BAGGAGEID	CHAR	4	-	-	1	-	-	-
	PASSENGERID	CHAR	4	-	-	-	✓	-	-
	FLIGHTID	CHAR	4	-	-	-	✓	-	-
	WEIGHT	NUMBER	-	4	1	-	✓	-	-
	BAGGAGETYPE	VARCHAR2	30	-	-	-	✓	-	-

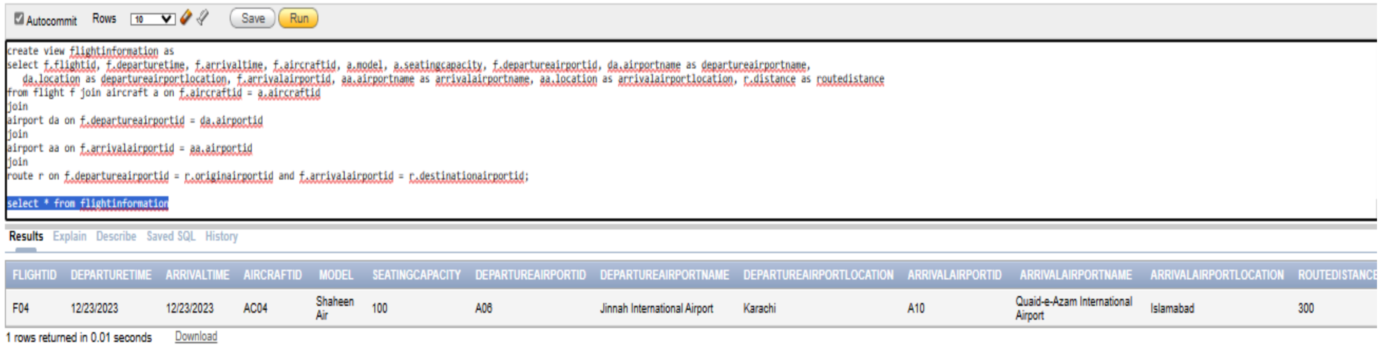
1 - 5

Designing Views:

Flight Information View

This view combines details related to flights, including departure and arrival information, aircraft details, airports, and route distances.

```
create view flightinformation as
select f.flightid, f.departuretime, f.arrivaltime, f.aircraftid, a.model,
a.seatingcapacity, f.departureairportid, da.airportname as departureairportname,
da.location as departureairportlocation, f.arrivalairportid, aa.airportname as
arrivalairportname, aa.location as arrivalairportlocation, r.distance as
routedistance
from flight f join aircraft a on f.aircraftid = a.aircraftid
join airport da
on f.departureairportid = da.airportid
join airport aa
on f.arrivalairportid = aa.airportid
join route r
on f.departureairportid = r.originairportid and f.arrivalairportid =
r.destinationairportid;
```



```
create view flightinformation as
select f.flightid, f.departuretime, f.arrivaltime, f.aircraftid, a.model, a.seatingcapacity, f.departureairportid, da.airportname as departureairportname,
   da.location as departureairportlocation, f.arrivalairportid, aa.airportname as arrivalairportname, aa.location as arrivalairportlocation, r.distance as routedistance
from flight f join aircraft a on f.aircraftid = a.aircraftid
join
airport da on f.departureairportid = da.airportid
join
airport aa on f.arrivalairportid = aa.airportid
join
route r on f.departureairportid = r.originairportid and f.arrivalairportid = r.destinationairportid;

select * from flightinformation
```

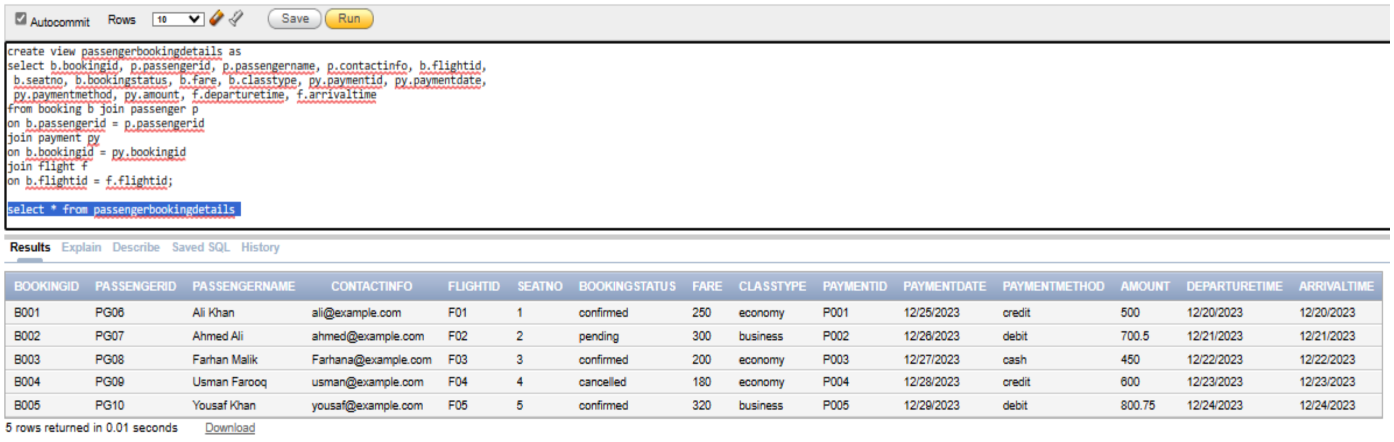
FLIGHTID	DEPARTURETIME	ARRIVALTIME	AIRCRAFTID	MODEL	SEATINGCAPACITY	DEPARTUREAIRPORTID	DEPARTUREAIRPORTNAME	DEPARTUREAIRPORTLOCATION	ARRIVALAIRPORTID	ARRIVALAIRPORTNAME	ARRIVALAIRPORTLOCATION	ROUTEDISTANCE
F04	12/23/2023	12/23/2023	AC04	Shaheen Air	100	A06	Jinnah International Airport	Karachi	A10	Quaid-e-Azam International Airport	Islamabad	300

1 rows returned in 0.01 seconds   Download

Passenger Booking Details View

This view provides information about passenger bookings, including their personal details, booking status, fare, payment details, and flight information.

```
create view passengerbookingdetails as
select b.bookingid, p.passengerid, p.passengername, p.contactinfo, b.flightid,
b.seatno, b.bookingstatus, b.fare, b.classtype, py.paymentid, py.paymentdate,
py.paymentmethod, py.amount, f.departuretime, f.arrivaltime
from booking b join passenger p
on b.passengerid = p.passengerid
join payment py
on b.bookingid = py.bookingid
join flight f
on b.flightid = f.flightid;
```

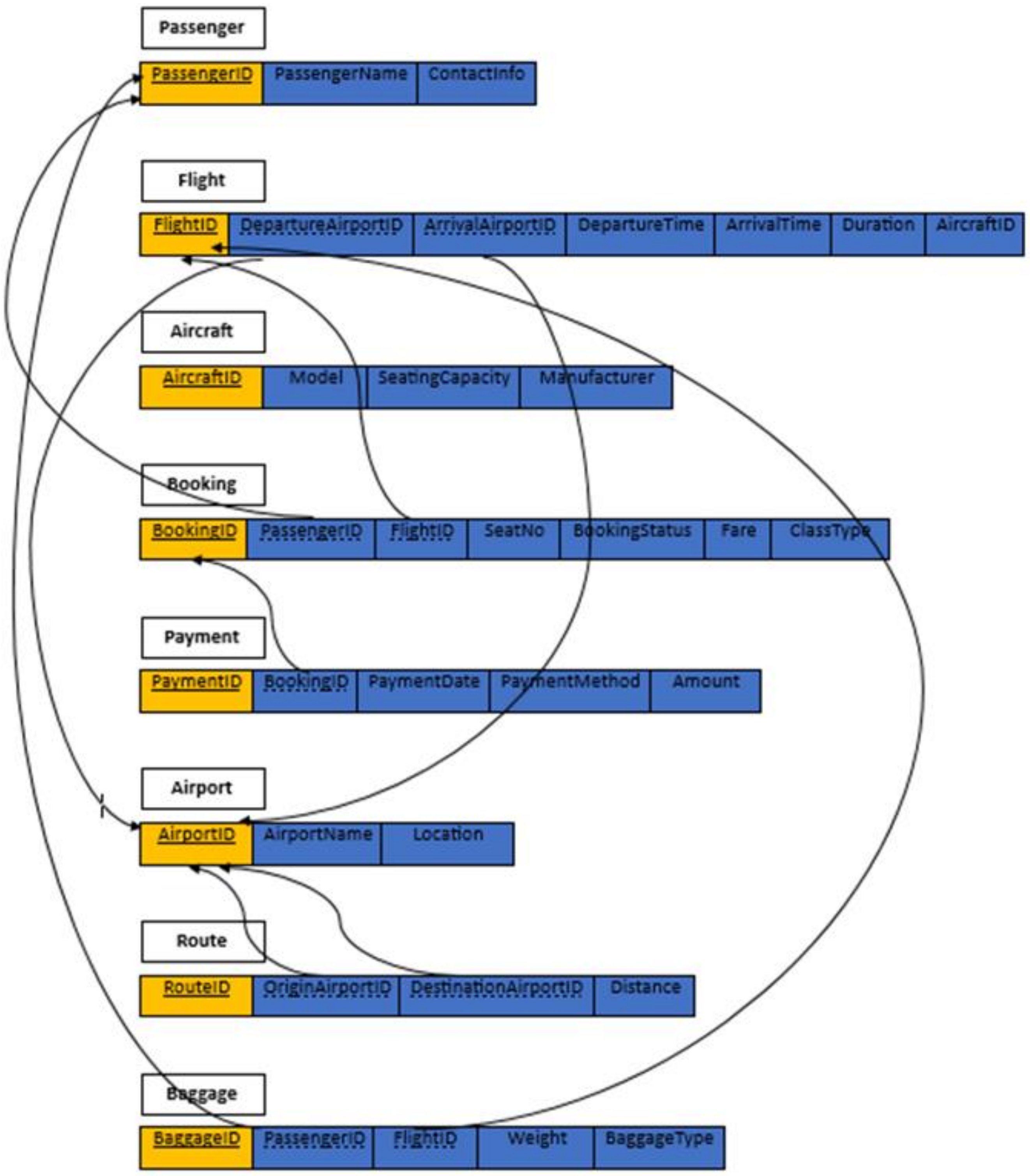


```
create view passengerbookingdetails as
select b.bookingid, p.passengerid, p.passengername, p.contactinfo, b.flightid,
 b.seatno, b.bookingstatus, b.fare, b.classtype, py.paymentid, py.paymentdate,
 py.paymentmethod, py.amount, f.departuretime, f.arrivaltime
from booking b join passenger p
on b.passengerid = p.passengerid
join payment py
on b.bookingid = py.bookingid
join flight f
on b.flightid = f.flightid;

select * from passengerbookingdetails
```

BOOKINGID	PASSENGERID	PASSENGERNAME	CONTACTINFO	FLIGHTID	SEATNO	BOOKINGSTATUS	FARE	CLASSTYPE	PAYMENTID	PAYMENTDATE	PAYMENTMETHOD	AMOUNT	DEPARTURETIME	ARRIVALTIME
B001	PG06	Ali Khan	ali@example.com	F01	1	confirmed	250	economy	P001	12/25/2023	credit	500	12/20/2023	12/20/2023
B002	PG07	Ahmed Ali	ahmed@example.com	F02	2	pending	300	business	P002	12/26/2023	debit	700.5	12/21/2023	12/21/2023
B003	PG08	Farhan Malik	Farhana@example.com	F03	3	confirmed	200	economy	P003	12/27/2023	cash	450	12/22/2023	12/22/2023
B004	PG09	Usman Farooq	usman@example.com	F04	4	cancelled	180	economy	P004	12/28/2023	credit	600	12/23/2023	12/23/2023
B005	PG10	Yousaf Khan	yousaf@example.com	F05	5	confirmed	320	business	P005	12/29/2023	debit	800.75	12/24/2023	12/24/2023

5 rows returned in 0.01 seconds

Relational Data Model (Dependency Diagram)

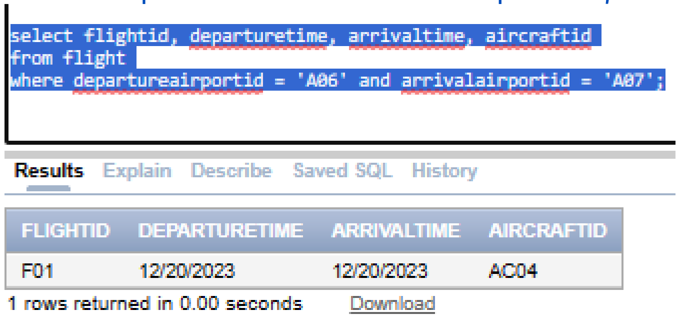


Select Statements for Common Reports:

Explain how will you use database to display things in the application make minimum of 5 of such use case statements and explain how will user get benefits from it or at what occasion they will be used.
Note:
The statements should no be like **select * from table**

Flight Schedule for a Specific Route:

```
select flightid, departuretime, arrivaltime, aircraftid from flight
where departureairportid = 'originairportid' and
arrivalairportid = 'destinationairportid';
```



```
select flightid, departuretime, arrivaltime, aircraftid
from flight
where departureairportid = 'A06' and arrivalairportid = 'A07';
```

Results Explain Describe Saved SQL History

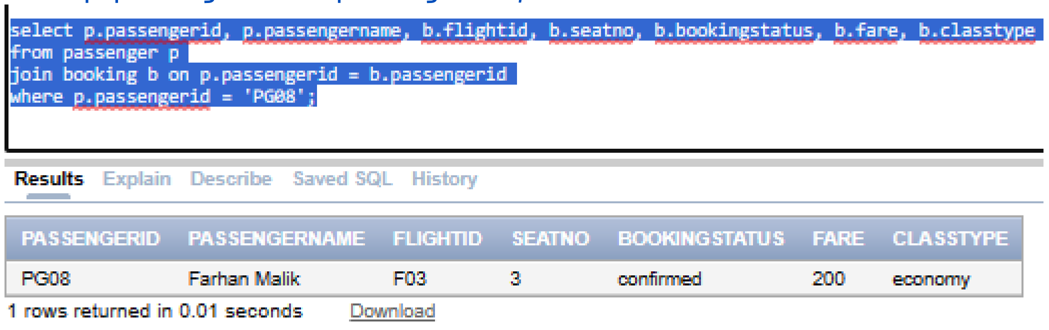
FLIGHTID	DEPARTURETIME	ARRIVALTIME	AIRCRAFTID
F01	12/20/2023	12/20/2023	AC04

1 rows returned in 0.00 seconds Download

Use Case: Passengers looking to plan their trip can check the schedule for flights between specific airports, helping them decide on suitable travel times.

Passenger Booking Details:

```
select p.passengerid, p.passengername, b.flightid, b.seatno, b.bookingstatus,
b.fare, b.classtype
from passenger p
join booking b on p.passengerid = b.passengerid
where p.passengerid = 'passengerid';
```



```
select p.passengerid, p.passengername, b.flightid, b.seatno, b.bookingstatus, b.fare, b.classtype
from passenger p
join booking b on p.passengerid = b.passengerid
where p.passengerid = 'PG08';
```

Results Explain Describe Saved SQL History

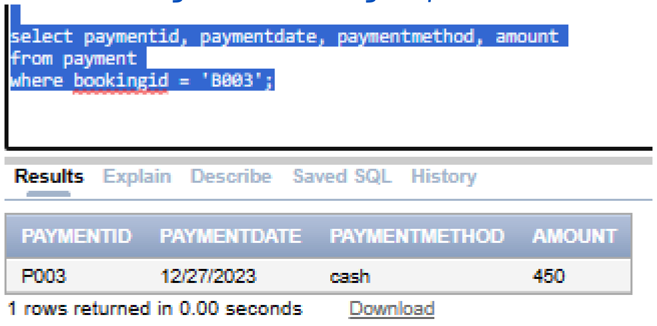
PASSENGERID	PASSENGERNAME	FLIGHTID	SEATNO	BOOKINGSTATUS	FARE	CLASSTYPE
PG08	Farhan Malik	F03	3	confirmed	200	economy

1 rows returned in 0.01 seconds Download

Use Case: Passengers can view their booking details after logging in, allowing them to check their flight details, seat number, fare, and booking status.

Payment History for a Booking:

```
select paymentid, paymentdate, paymentmethod, amount
from payment
where bookingid = 'bookingid';
```



```
select paymentid, paymentdate, paymentmethod, amount
from payment
where bookingid = 'B003';
```

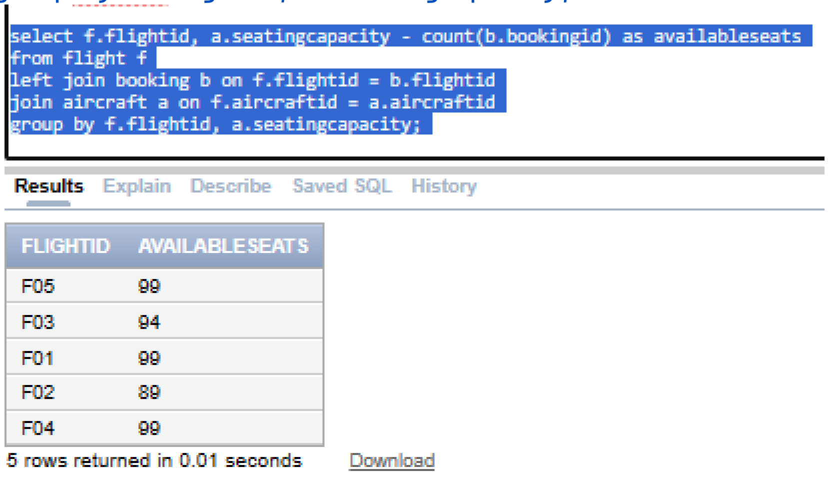
PAYMENTID	PAYMENTDATE	PAYMENTMETHOD	AMOUNT
P003	12/27/2023	cash	450

1 rows returned in 0.00 seconds Download

Use Case: Passengers can view their payment history for a specific booking, ensuring transparency and enabling them to track their payment transactions.

Available Seating Capacity for a Flight:

```
select f.flightid, a.seatingcapacity - count(b.bookingid) as availableseats
from flight f
left join booking b on f.flightid = b.flightid
join aircraft a on f.aircraftid = a.aircraftid
group by f.flightid, a.seatingcapacity;
```



```
select f.flightid, a.seatingcapacity - count(b.bookingid) as availableseats
from flight f
left join booking b on f.flightid = b.flightid
join aircraft a on f.aircraftid = a.aircraftid
group by f.flightid, a.seatingcapacity;
```

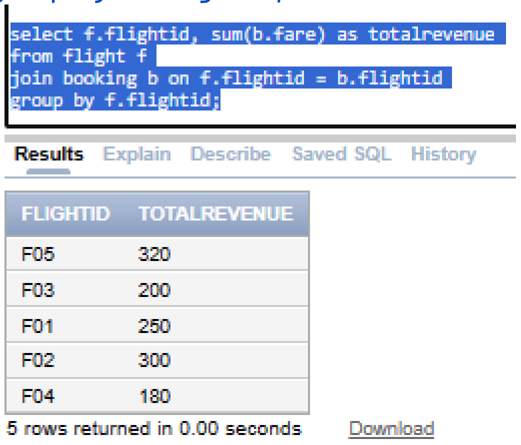
FLIGHTID	AVAILABLESEATS
F05	99
F03	94
F01	99
F02	89
F04	99

5 rows returned in 0.01 seconds Download

Use Case: Airlines can monitor the available seats for each flight in real-time, helping them manage bookings.

Total Revenue Generated by Flights:

```
select f.flightid, sum(b.fare) as totalrevenue
from flight f
join booking b on f.flightid = b.flightid
group by f.flightid;
```



```
select f.flightid, sum(b.fare) as totalrevenue
from flight f
join booking b on f.flightid = b.flightid
group by f.flightid;
```

Results Explain Describe Saved SQL History

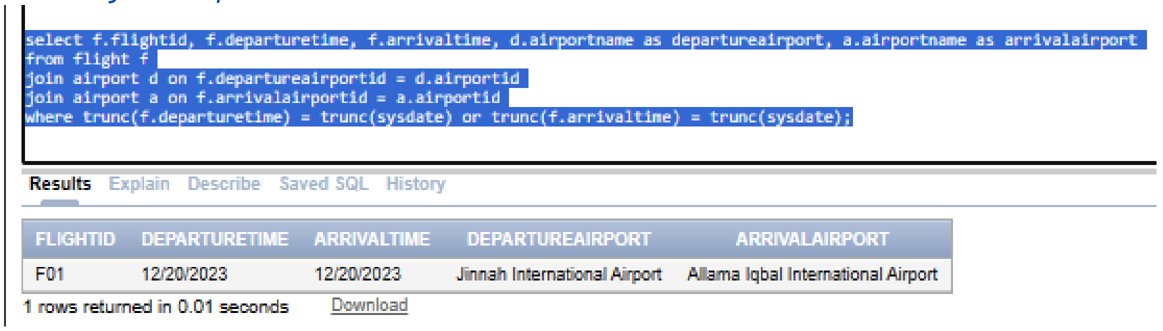
FLIGHTID	TOTALREVENUE
F05	320
F03	200
F01	250
F02	300
F04	180

5 rows returned in 0.00 seconds Download

Use Case: This report helps airline managers analyze the revenue generated by each flight, aiding in identifying profitable routes and optimizing pricing strategies.

Flights Departing and Arriving Today:

```
select f.flightid, f.departuretime, f.arrivaltime, d.airportname as
departureairport, a.airportname as arrivalairport
from flight f
join airport d on f.departureairportid = d.airportid
join airport a on f.arrivalairportid = a.airportid
where trunc(f.departuretime) = trunc(sysdate) or trunc(f.arrivaltime) =
trunc(sysdate);
```



```
select f.flightid, f.departuretime, f.arrivaltime, d.airportname as departureairport, a.airportname as arrivalairport
from flight f
join airport d on f.departureairportid = d.airportid
join airport a on f.arrivalairportid = a.airportid
where trunc(f.departuretime) = trunc(sysdate) or trunc(f.arrivaltime) = trunc(sysdate);
```

Results Explain Describe Saved SQL History

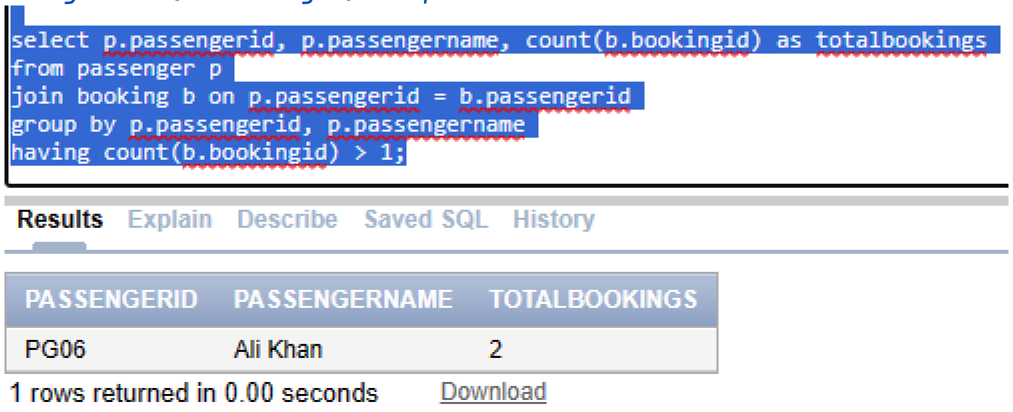
FLIGHTID	DEPARTURETIME	ARRIVALTIME	DEPARTUREAIRPORT	ARRIVALAIRPORT
F01	12/20/2023	12/20/2023	Jinnah International Airport	Allama Iqbal International Airport

1 rows returned in 0.01 seconds Download

Use Case: This report provides real-time flight information for passengers or staff about departures and arrivals on the current day.

Passengers with Multiple Bookings:

```
select p.passengerid, p.passengername, count(b.bookingid) as totalbookings
from passenger p
join booking b on p.passengerid = b.passengerid
group by p.passengerid, p.passengername
having count(b.bookingid) > 1;
```



```
select p.passengerid, p.passengername, count(b.bookingid) as totalbookings
from passenger p
join booking b on p.passengerid = b.passengerid
group by p.passengerid, p.passengername
having count(b.bookingid) > 1;
```

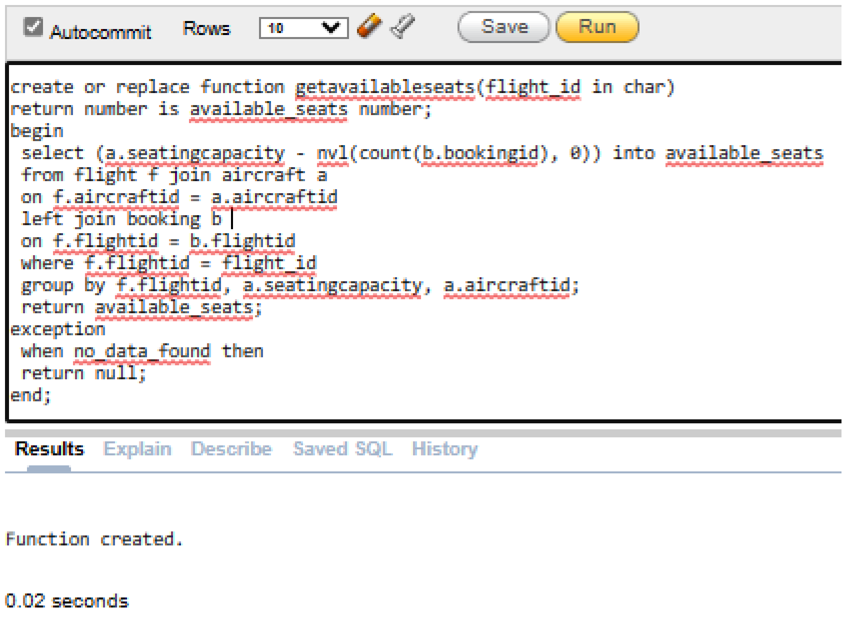
PASSENGERID	PASSENGERNAME	TOTALBOOKINGS
PG06	Ali Khan	2

1 rows returned in 0.00 seconds

Use Case: This report identifies passengers who frequently book flights, assisting in special discounts for regular customers.

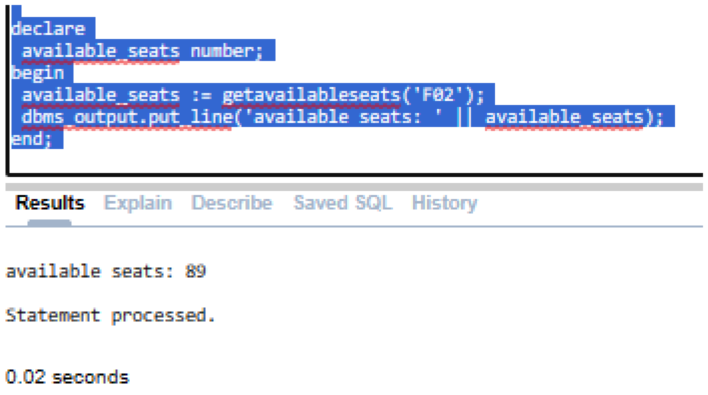
Demonstrating Functions:

Function to Get Available Seats for a Flight:



```
create or replace function getavailableseats(flight_id in char)
return number is available_seats number;
begin
 select (a.seatingcapacity - nvl(count(b.bookingid), 0)) into available_seats
 from flight f join aircraft a
 on f.aircraftid = a.aircraftid
 left join booking b
 on f.flightid = b.flightid
 where f.flightid = flight_id
 group by f.flightid, a.seatingcapacity, a.aircraftid;
 return available_seats;
exception
 when no_data_found then
 return null;
end;
```

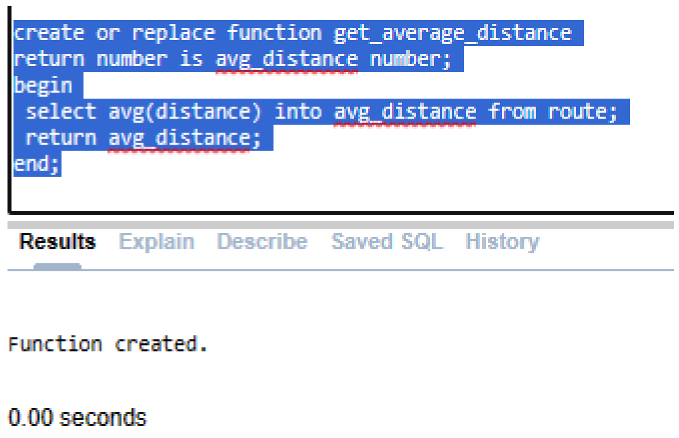
```
Function created.

0.02 seconds
```



```
declare
 available_seats number;
begin
 available_seats := getavailableseats('F02');
 dbms_output.put_line('available seats: ' || available_seats);
end;
```

```
available seats: 89

Statement processed.

0.02 seconds
```

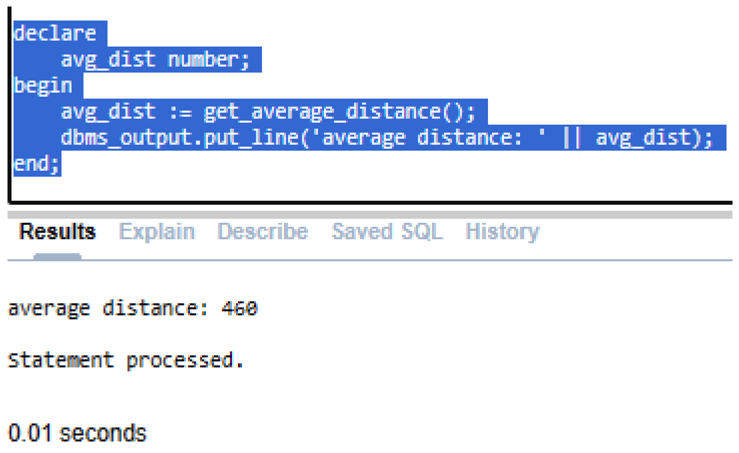
Function to Get Average Distance of Routes:



```
create or replace function get_average_distance
return number is avg_distance number;
begin
 select avg(distance) into avg_distance from route;
 return avg_distance;
end;
```

```
Function created.
```

0.00 seconds



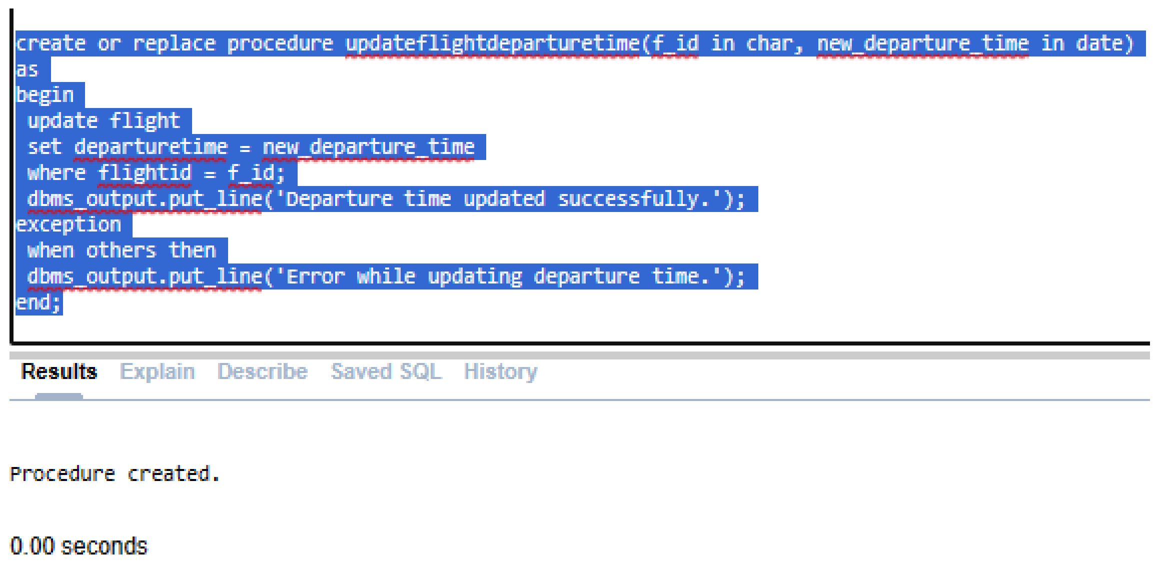
```
declare
    avg_dist number;
begin
    avg_dist := get_average_distance();
    dbms_output.put_line('average distance: ' || avg_dist);
end;
```

```
average distance: 460

Statement processed.
```

0.01 seconds

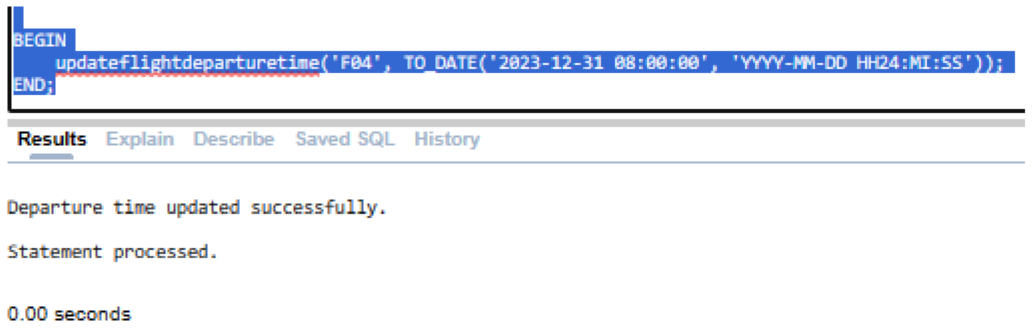
Demonstrating Procedures:

Procedure to Update Flight Departure Time



```
create or replace procedure updateflightdeparturetime(f_id in char, new_departure_time in date)
as
begin
 update flight
 set departuretime = new_departure_time
 where flightid = f_id;
 dbms_output.put_line('Departure time updated successfully.');
exception
 when others then
 dbms_output.put_line('Error while updating departure time.');
end;
```

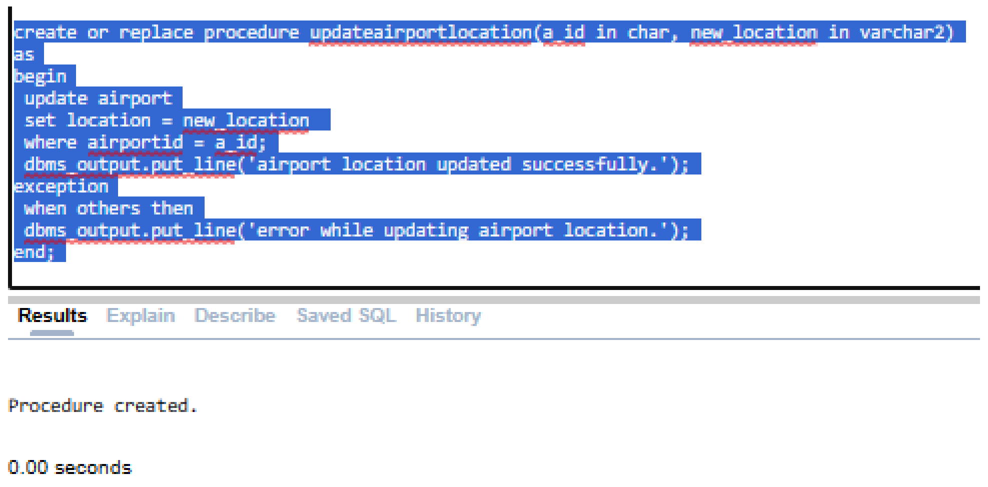
```
Procedure created.

0.00 seconds
```



```
BEGIN
    updateflightdeparturetime('F04', TO_DATE('2023-12-31 08:00:00', 'YYYY-MM-DD HH24:MI:SS'));
END;
```

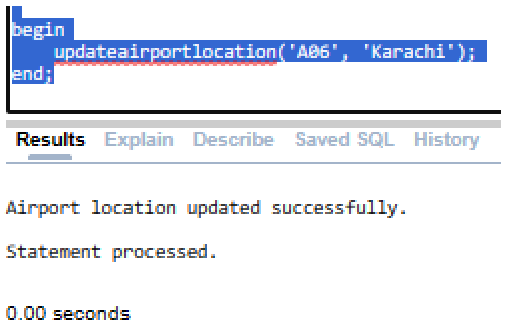
```
Departure time updated successfully.

Statement processed.

0.00 seconds
```

Procedure to update airport location:



```
create or replace procedure updateairportlocation(a_id in char, new_location in varchar2)
as
begin
 update airport
 set location = new_location
 where airportid = a_id;
 dbms_output.put_line('airport location updated successfully.');
exception
 when others then
 dbms_output.put_line('error while updating airport location.');
end;
```

```
Procedure created.

0.00 seconds
```


```
begin
    updateairportlocation('A06', 'Karachi');
end;
```

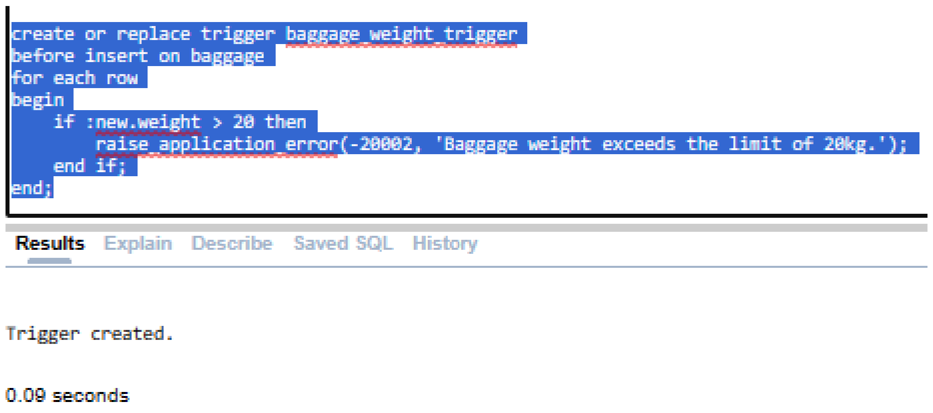
Results Explain Describe Saved SQL History

```
Airport location updated successfully.

Statement processed.
```

0.00 seconds

Demonstrating Triggers:

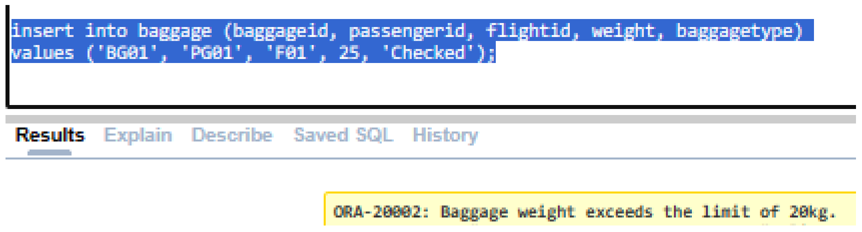
Baggage Weight Trigger:


```
create or replace trigger baggage_weight_trigger
before insert on baggage
for each row
begin
    if :new.weight > 20 then
        raise_application_error(-20002, 'Baggage weight exceeds the limit of 20kg.');
    end if;
end;
```

Results Explain Describe Saved SQL History

```
Trigger created.
```

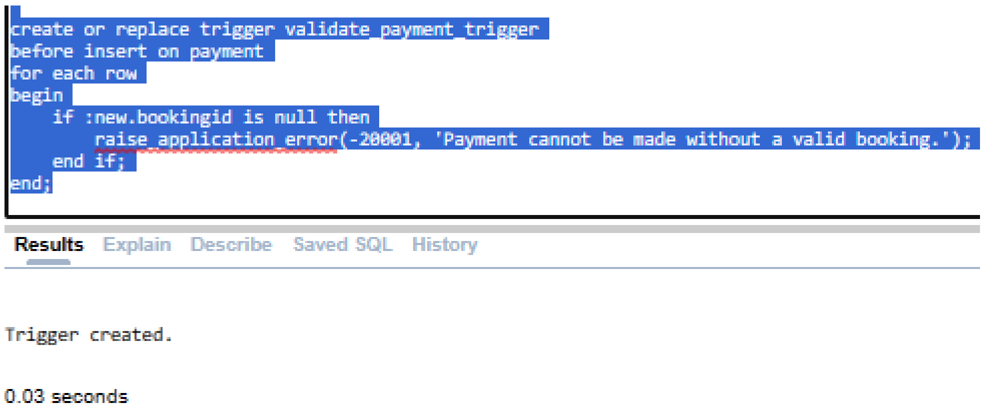
0.09 seconds


```
insert into baggage (baggageid, passengerid, flightid, weight, baggagetype)
values ('BG01', 'PG01', 'F01', 25, 'Checked');
```

Results Explain Describe Saved SQL History

```
ORA-20002: Baggage weight exceeds the limit of 20kg.
```

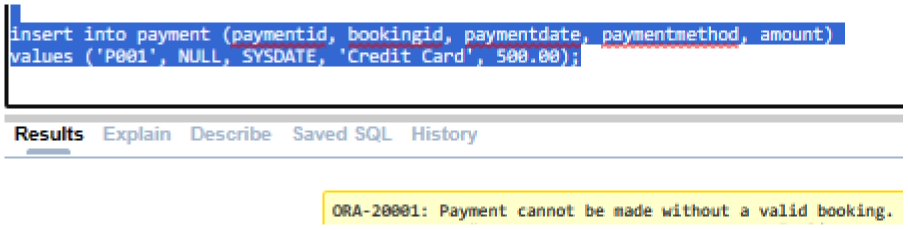
Payment Validation Trigger:



```
create or replace trigger validate_payment_trigger
before insert on payment
for each row
begin
    if :new.bookingid is null then
        raise_application_error(-20001, 'Payment cannot be made without a valid booking.');
    end if;
end;
```

```
Trigger created.
```

0.03 seconds



```
insert into payment (paymentid, bookingid, paymentdate, paymentmethod, amount)
values ('P001', NULL, SYSDATE, 'Credit Card', 500.00);
```

```
ORA-20001: Payment cannot be made without a valid booking.
```