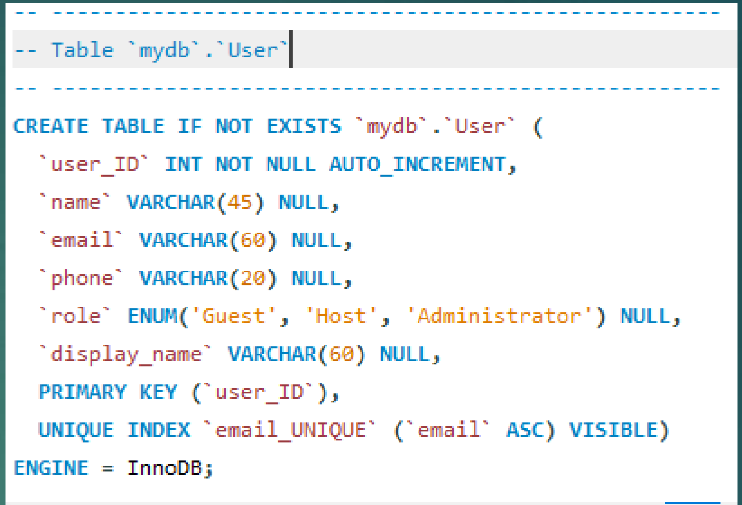
Airbnb-like Database

DESIGN AND IMPLEMENTATION

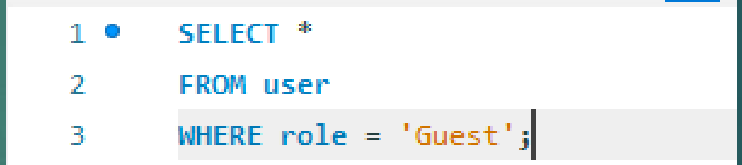
MADE BY JURAJ PAUKER (3220418)

User Entity

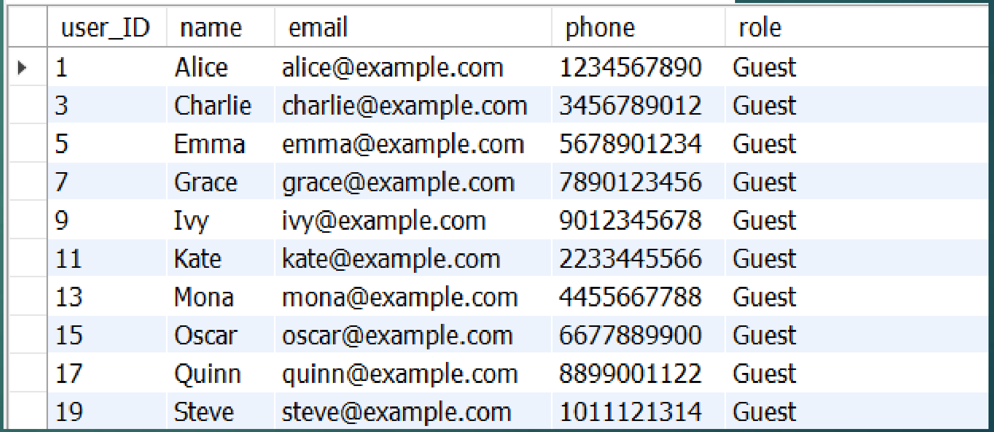
- The first picture shows the statement for creating the User table
- The user_ID attribute is the primary key of the User table
- The test case retrieves all records from the user table where the role column is set to 'Guest'



```
-- -----------------------------------------------------
-- Table `mydb`.`User`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`User` (
  `user_ID` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(45) NULL,
  `email` VARCHAR(60) NULL,
  `phone` VARCHAR(20) NULL,
  `role` ENUM('Guest', 'Host', 'Administrator') NULL,
  `display_name` VARCHAR(60) NULL,
  PRIMARY KEY (`user_ID`),
  UNIQUE INDEX `email_UNIQUE` (`email` ASC) VISIBLE)
ENGINE = InnoDB;
```



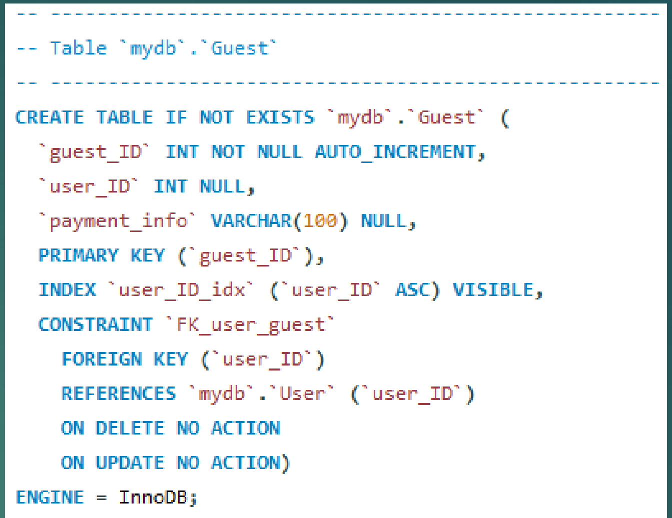
```
SELECT *
FROM user
WHERE role = 'Guest';
```



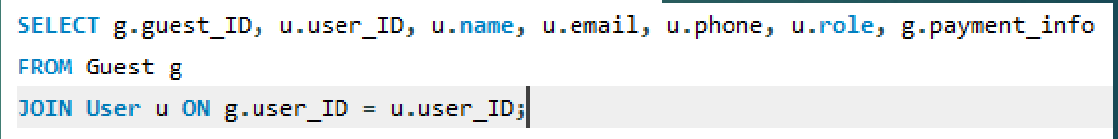
user_ID	name	email	phone	role
1	Alice	alice@example.com	1234567890	Guest
3	Charlie	charlie@example.com	3456789012	Guest
5	Emma	emma@example.com	5678901234	Guest
7	Grace	grace@example.com	7890123456	Guest
9	Ivy	ivy@example.com	9012345678	Guest
11	Kate	kate@example.com	2233445566	Guest
13	Mona	mona@example.com	4455667788	Guest
15	Oscar	oscar@example.com	6677889900	Guest
17	Quinn	quinn@example.com	8899001122	Guest
19	Steve	steve@example.com	1011121314	Guest

Guest Entity

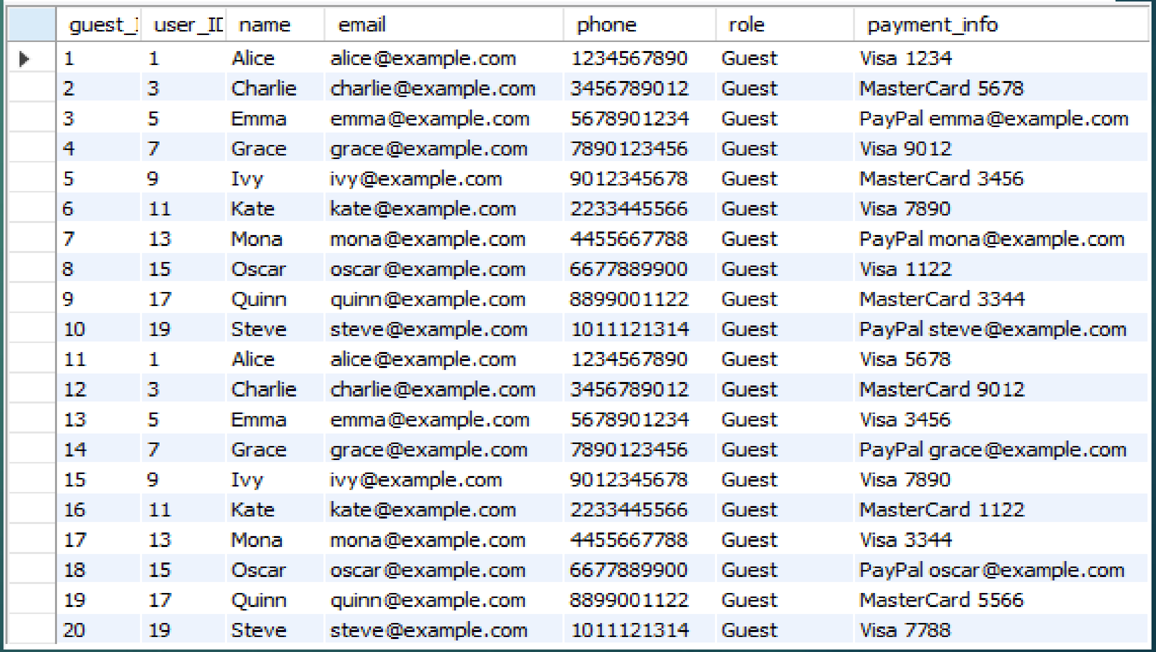
- The first picture shows the statement for creating the Guest table
- The guest_ID attribute is the primary key of the Guest table, while the user_ID attribute is the foreign key which references the User table
- The test case retrieves guest details along with their associated user information by joining the Guest and User tables on the user_ID field



```
-- -----------------------------------------------------
-- Table `mydb`.`Guest`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Guest` (
  `guest_ID` INT NOT NULL AUTO_INCREMENT,
  `user_ID` INT NULL,
  `payment_info` VARCHAR(100) NULL,
  PRIMARY KEY (`guest_ID`),
  INDEX `user_ID_idx` (`user_ID` ASC) VISIBLE,
  CONSTRAINT `FK_user_guest`
    FOREIGN KEY (`user_ID`)
    REFERENCES `mydb`.`User` (`user_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```



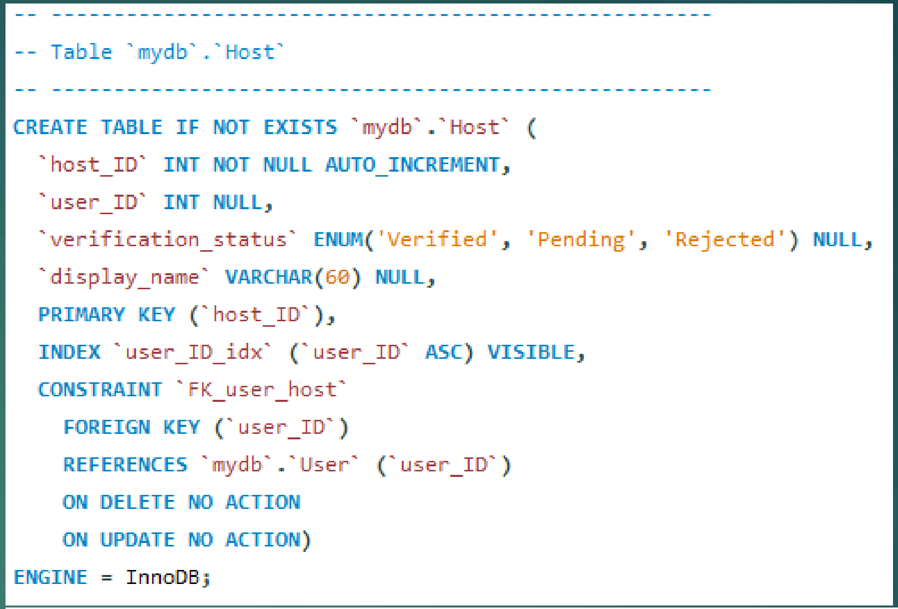
```
SELECT g.guest_ID, u.user_ID, u.name, u.email, u.phone, u.role, g.payment_info
FROM Guest g
JOIN User u ON g.user_ID = u.user_ID;
```



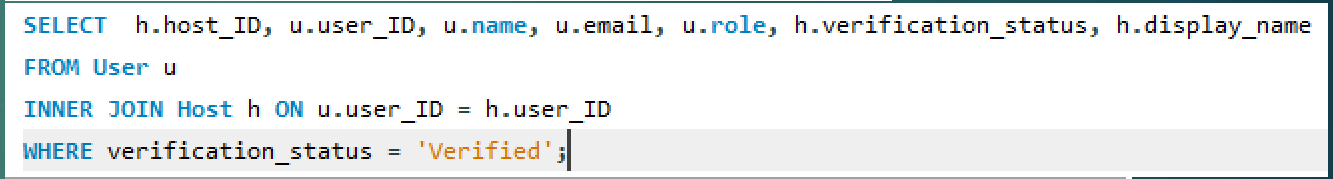
guest_I	user_II	name	email	phone	role	payment_info
1	1	Alice	alice@example.com	1234567890	Guest	Visa 1234
2	3	Charlie	charlie@example.com	3456789012	Guest	MasterCard 5678
3	5	Emma	emma@example.com	5678901234	Guest	PayPal emma@example.com
4	7	Grace	grace@example.com	7890123456	Guest	Visa 9012
5	9	Ivy	ivy@example.com	9012345678	Guest	MasterCard 3456
6	11	Kate	kate@example.com	2233445566	Guest	Visa 7890
7	13	Mona	mona@example.com	4455667788	Guest	PayPal mona@example.com
8	15	Oscar	oscar@example.com	6677889900	Guest	Visa 1122
9	17	Quinn	quinn@example.com	8899001122	Guest	MasterCard 3344
10	19	Steve	steve@example.com	1011121314	Guest	PayPal steve@example.com
11	1	Alice	alice@example.com	1234567890	Guest	Visa 5678
12	3	Charlie	charlie@example.com	3456789012	Guest	MasterCard 9012
13	5	Emma	emma@example.com	5678901234	Guest	Visa 3456
14	7	Grace	grace@example.com	7890123456	Guest	PayPal grace@example.com
15	9	Ivy	ivy@example.com	9012345678	Guest	Visa 7890
16	11	Kate	kate@example.com	2233445566	Guest	MasterCard 1122
17	13	Mona	mona@example.com	4455667788	Guest	Visa 3344
18	15	Oscar	oscar@example.com	6677889900	Guest	PayPal oscar@example.com
19	17	Quinn	quinn@example.com	8899001122	Guest	MasterCard 5566
20	19	Steve	steve@example.com	1011121314	Guest	Visa 7788

Host Entity

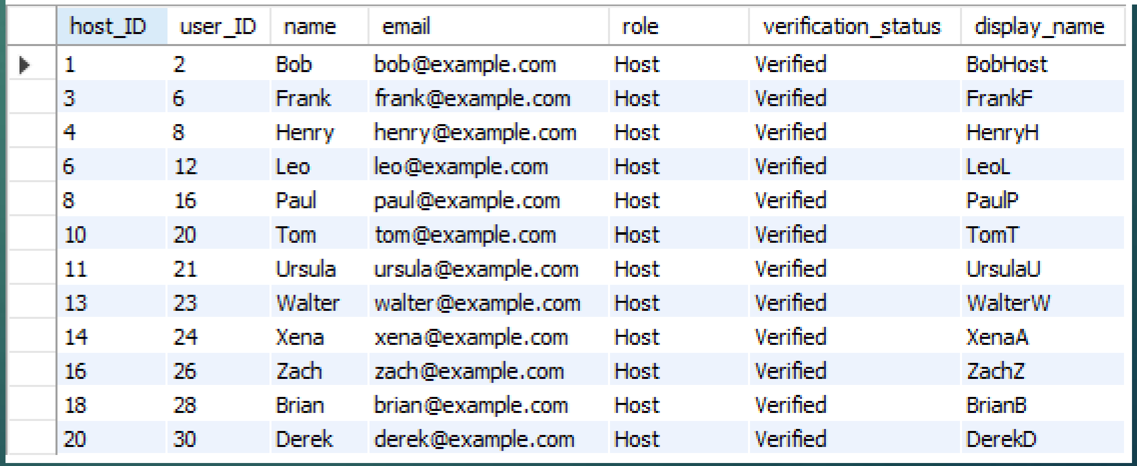
- The first picture shows the statement for creating the Host table
- The host_ID attribute is the primary key of the Host table, while the user_ID attribute is the foreign key which references the User table
- The test case retrieves user details of hosts whose verification status is ‘Verified’



```
-- -----------------------------------------------------
-- Table `mydb`.`Host`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Host` (
  `host_ID` INT NOT NULL AUTO_INCREMENT,
  `user_ID` INT NULL,
  `verification_status` ENUM('Verified', 'Pending', 'Rejected') NULL,
  `display_name` VARCHAR(60) NULL,
  PRIMARY KEY (`host_ID`),
  INDEX `user_ID_idx` (`user_ID` ASC) VISIBLE,
  CONSTRAINT `FK_user_host`
    FOREIGN KEY (`user_ID`)
    REFERENCES `mydb`.`User` (`user_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```



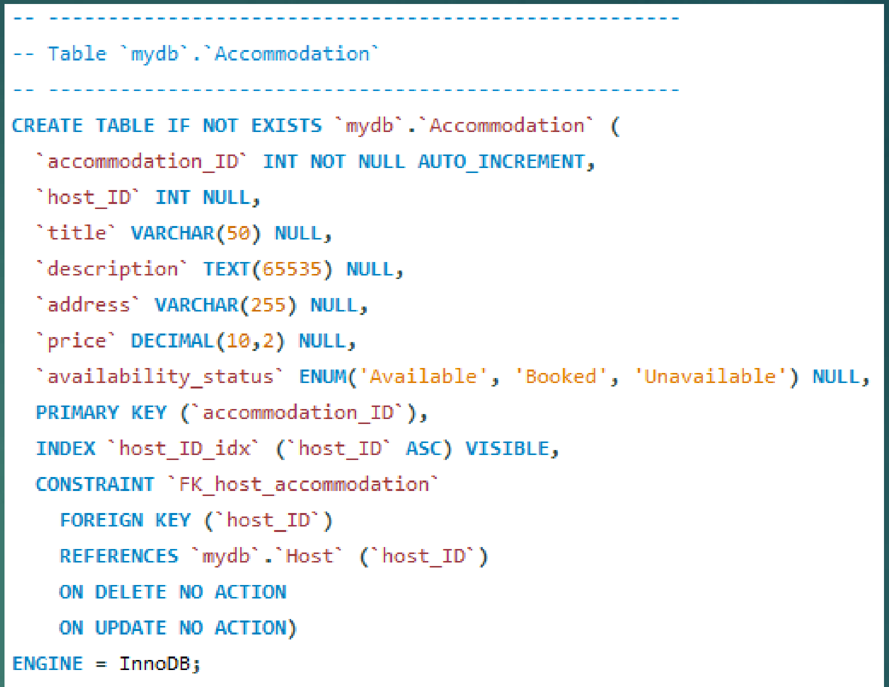
```
SELECT  h.host_ID, u.user_ID, u.name, u.email, u.role, h.verification_status, h.display_name
FROM User u
INNER JOIN Host h ON u.user_ID = h.user_ID
WHERE verification_status = 'Verified';
```



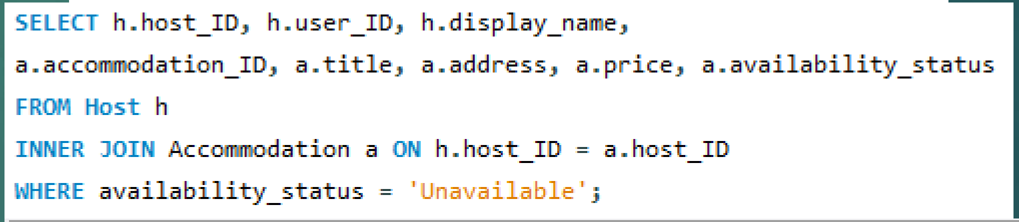
host_ID	user_ID	name	email	role	verification_status	display_name
1	2	Bob	bob@example.com	Host	Verified	BobHost
3	6	Frank	frank@example.com	Host	Verified	FrankF
4	8	Henry	henry@example.com	Host	Verified	HenryH
6	12	Leo	leo@example.com	Host	Verified	LeoL
8	16	Paul	paul@example.com	Host	Verified	PaulP
10	20	Tom	tom@example.com	Host	Verified	TomT
11	21	Ursula	ursula@example.com	Host	Verified	UrsulaU
13	23	Walter	walter@example.com	Host	Verified	WalterW
14	24	Xena	xena@example.com	Host	Verified	XenaA
16	26	Zach	zach@example.com	Host	Verified	ZachZ
18	28	Brian	brian@example.com	Host	Verified	BrianB
20	30	Derek	derek@example.com	Host	Verified	DerekD

Accommodation Entity

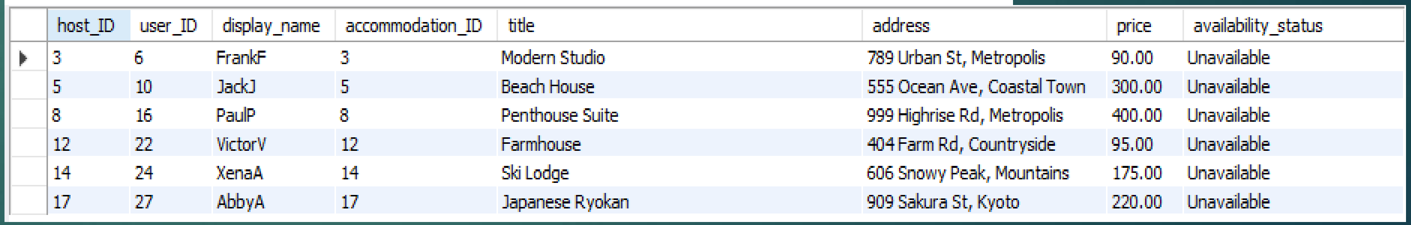
- The first picture shows the statement for creating the Accommodation table
- The accommodation_ID attribute is the primary key of the Accommodation table, while the host_ID attribute is the foreign key which references the Host table
- The test case retrieves information about accommodations that are currently unavailable along with their corresponding hosts



```
-- -----------------------------------------------------
-- Table `mydb`.`Accommodation`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Accommodation` (
  `accommodation_ID` INT NOT NULL AUTO_INCREMENT,
  `host_ID` INT NULL,
  `title` VARCHAR(50) NULL,
  `description` TEXT(65535) NULL,
  `address` VARCHAR(255) NULL,
  `price` DECIMAL(10,2) NULL,
  `availability_status` ENUM('Available', 'Booked', 'Unavailable') NULL,
  PRIMARY KEY (`accommodation_ID`),
  INDEX `host_ID_idx` (`host_ID` ASC) VISIBLE,
  CONSTRAINT `FK_host_accommodation`
    FOREIGN KEY (`host_ID`)
    REFERENCES `mydb`.`Host` (`host_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```



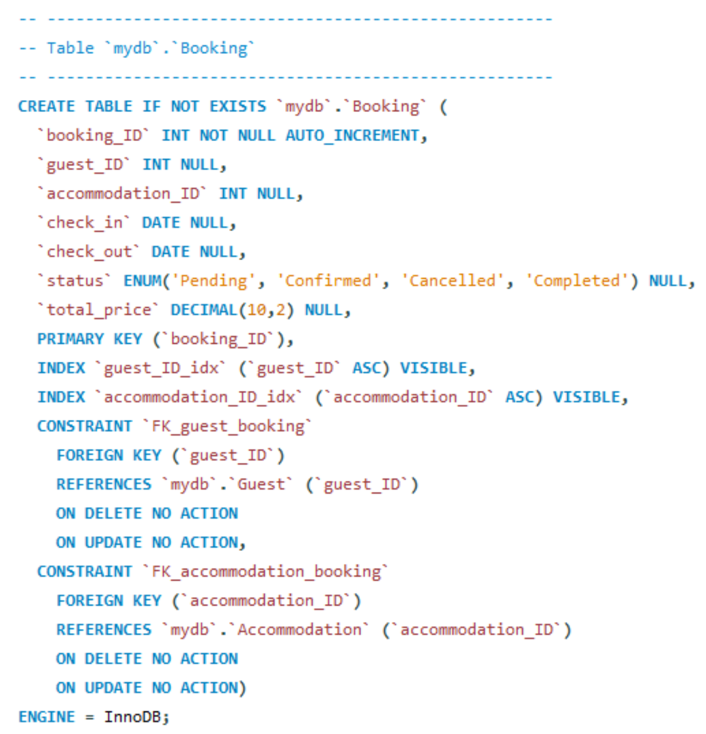
```
SELECT h.host_ID, h.user_ID, h.display_name,
a.accommodation_ID, a.title, a.address, a.price, a.availability_status
FROM Host h
INNER JOIN Accommodation a ON h.host_ID = a.host_ID
WHERE availability_status = 'Unavailable';
```



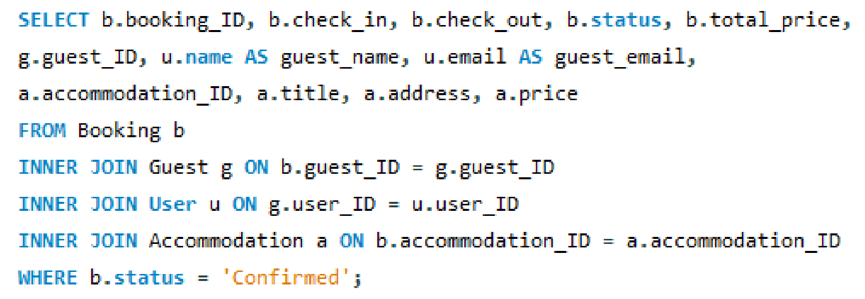
host_ID	user_ID	display_name	accommodation_ID	title	address	price	availability_status
3	6	FrankF	3	Modern Studio	789 Urban St, Metropolis	90.00	Unavailable
5	10	JackJ	5	Beach House	555 Ocean Ave, Coastal Town	300.00	Unavailable
8	16	PaulP	8	Penthouse Suite	999 Highrise Rd, Metropolis	400.00	Unavailable
12	22	VictorV	12	Farmhouse	404 Farm Rd, Countryside	95.00	Unavailable
14	24	XenaA	14	Ski Lodge	606 Snowy Peak, Mountains	175.00	Unavailable
17	27	AbbyA	17	Japanese Ryokan	909 Sakura St, Kyoto	220.00	Unavailable

Booking Entity

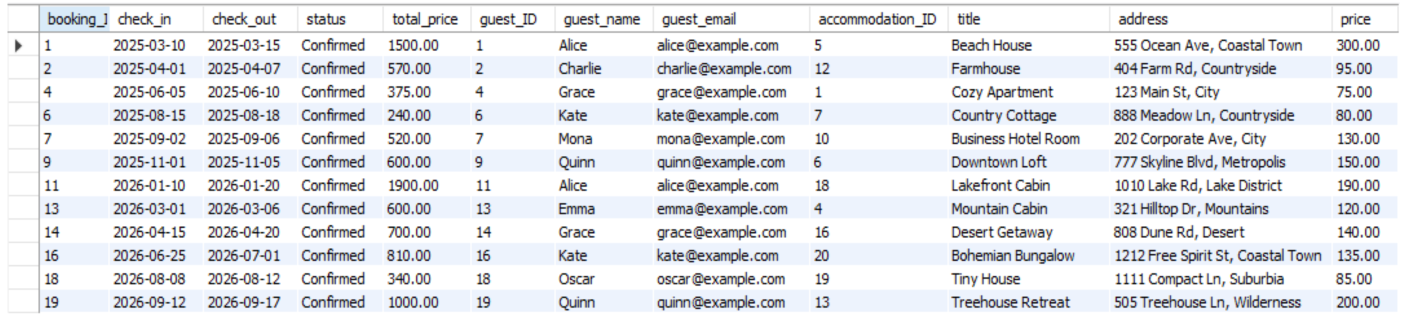
- The first picture shows the statement for creating the Booking table
- The booking_ID attribute is the primary key of the Booking table, while the attributes guest_ID and accommodation_ID are foreign keys which reference the Guest and Accomodation tables, respectively
- The test case retrieves details of confirmed bookings along with the guest and accommodation information



```
-- -----------------------------------------------------
-- Table `mydb`.`Booking`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Booking` (
  `booking_ID` INT NOT NULL AUTO_INCREMENT,
  `guest_ID` INT NULL,
  `accommodation_ID` INT NULL,
  `check_in` DATE NULL,
  `check_out` DATE NULL,
  `status` ENUM('Pending', 'Confirmed', 'Cancelled', 'Completed') NULL,
  `total_price` DECIMAL(10,2) NULL,
  PRIMARY KEY (`booking_ID`),
  INDEX `guest_ID_idx` (`guest_ID` ASC) VISIBLE,
  INDEX `accommodation_ID_idx` (`accommodation_ID` ASC) VISIBLE,
  CONSTRAINT `FK_guest_booking`
    FOREIGN KEY (`guest_ID`)
    REFERENCES `mydb`.`Guest` (`guest_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `FK_accommodation_booking`
    FOREIGN KEY (`accommodation_ID`)
    REFERENCES `mydb`.`Accommodation` (`accommodation_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```



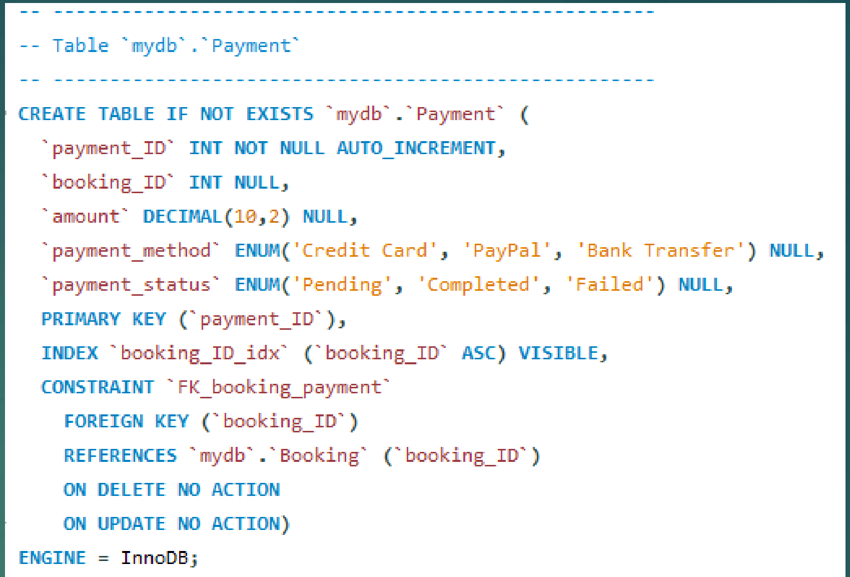
```
SELECT b.booking_ID, b.check_in, b.check_out, b.status, b.total_price,
g.guest_ID, u.name AS guest_name, u.email AS guest_email,
a.accommodation_ID, a.title, a.address, a.price
FROM Booking b
INNER JOIN Guest g ON b.guest_ID = g.guest_ID
INNER JOIN User u ON g.user_ID = u.user_ID
INNER JOIN Accommodation a ON b.accommodation_ID = a.accommodation_ID
WHERE b.status = 'Confirmed';
```



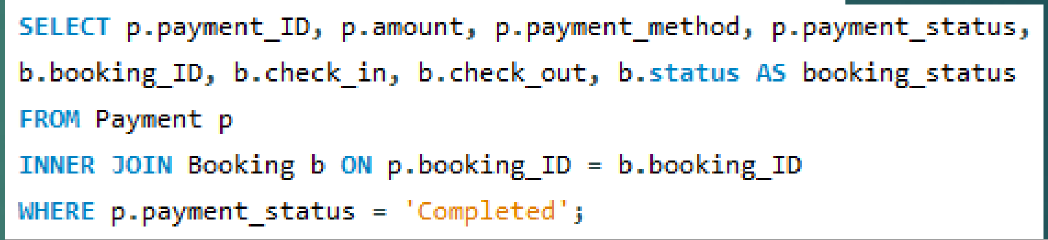
booking_I	check_in	check_out	status	total_price	guest_ID	guest_name	guest_email	accommodation_ID	title	address	price
1	2025-03-10	2025-03-15	Confirmed	1500.00	1	Alice	alice@example.com	5	Beach House	555 Ocean Ave, Coastal Town	300.00
2	2025-04-01	2025-04-07	Confirmed	570.00	2	Charlie	charlie@example.com	12	Farmhouse	404 Farm Rd, Countryside	95.00
4	2025-06-05	2025-06-10	Confirmed	375.00	4	Grace	grace@example.com	1	Cozy Apartment	123 Main St, City	75.00
6	2025-08-15	2025-08-18	Confirmed	240.00	6	Kate	kate@example.com	7	Country Cottage	888 Meadow Ln, Countryside	80.00
7	2025-09-02	2025-09-06	Confirmed	520.00	7	Mona	mona@example.com	10	Business Hotel Room	202 Corporate Ave, City	130.00
9	2025-11-01	2025-11-05	Confirmed	600.00	9	Quinn	quinn@example.com	6	Downtown Loft	777 Skyline Blvd, Metropolis	150.00
11	2026-01-10	2026-01-20	Confirmed	1900.00	11	Alice	alice@example.com	18	Lakefront Cabin	1010 Lake Rd, Lake District	190.00
13	2026-03-01	2026-03-06	Confirmed	600.00	13	Emma	emma@example.com	4	Mountain Cabin	321 Hilltop Dr, Mountains	120.00
14	2026-04-15	2026-04-20	Confirmed	700.00	14	Grace	grace@example.com	16	Desert Getaway	808 Dune Rd, Desert	140.00
16	2026-06-25	2026-07-01	Confirmed	810.00	16	Kate	kate@example.com	20	Bohemian Bungalow	1212 Free Spirit St, Coastal Town	135.00
18	2026-08-08	2026-08-12	Confirmed	340.00	18	Oscar	oscar@example.com	19	Tiny House	1111 Compact Ln, Suburbia	85.00
19	2026-09-12	2026-09-17	Confirmed	1000.00	19	Quinn	quinn@example.com	13	Treehouse Retreat	505 Treehouse Ln, Wilderness	200.00

Payment Entity

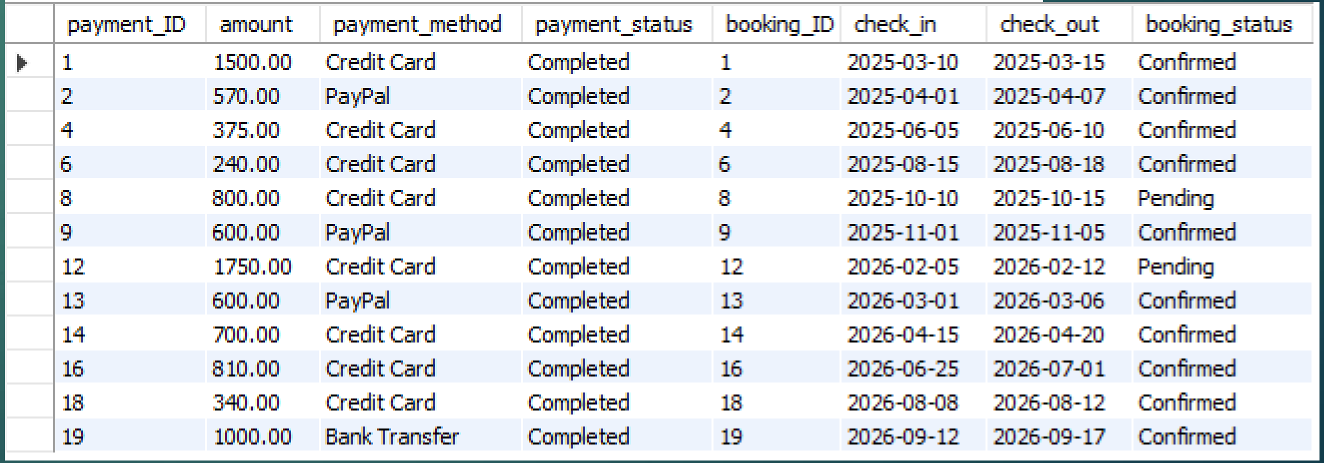
- The first picture shows the statement for creating the Payment table
- The attribute payment_ID is the primary key of the Payment table, while booking_ID is a foreign key which references the Booking table
- The test case retrieves details of completed payments along with the associated bookings



```
-- -----------------------------------------------------
-- Table `mydb`.`Payment`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Payment` (
  `payment_ID` INT NOT NULL AUTO_INCREMENT,
  `booking_ID` INT NULL,
  `amount` DECIMAL(10,2) NULL,
  `payment_method` ENUM('Credit Card', 'PayPal', 'Bank Transfer') NULL,
  `payment_status` ENUM('Pending', 'Completed', 'Failed') NULL,
  PRIMARY KEY (`payment_ID`),
  INDEX `booking_ID_idx` (`booking_ID` ASC) VISIBLE,
  CONSTRAINT `FK_booking_payment`
    FOREIGN KEY (`booking_ID`)
    REFERENCES `mydb`.`Booking` (`booking_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```



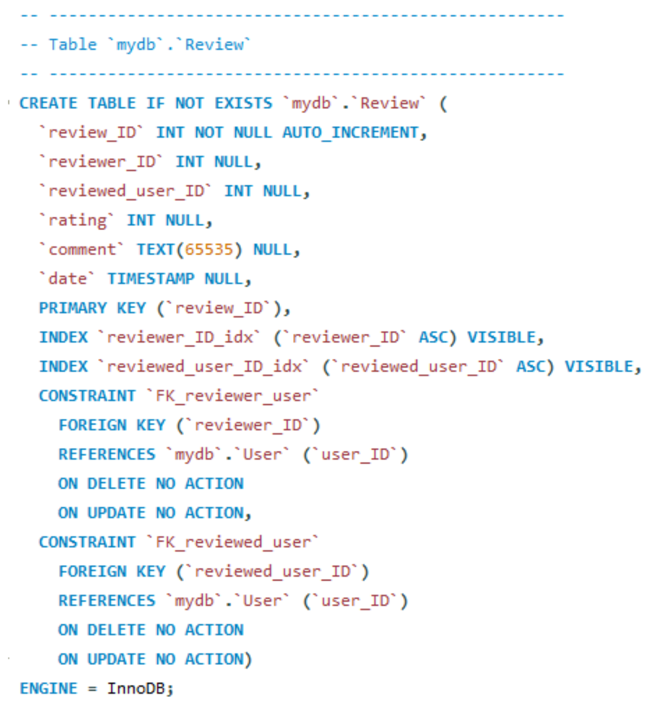
```
SELECT p.payment_ID, p.amount, p.payment_method, p.payment_status,
b.booking_ID, b.check_in, b.check_out, b.status AS booking_status
FROM Payment p
INNER JOIN Booking b ON p.booking_ID = b.booking_ID
WHERE p.payment_status = 'Completed';
```



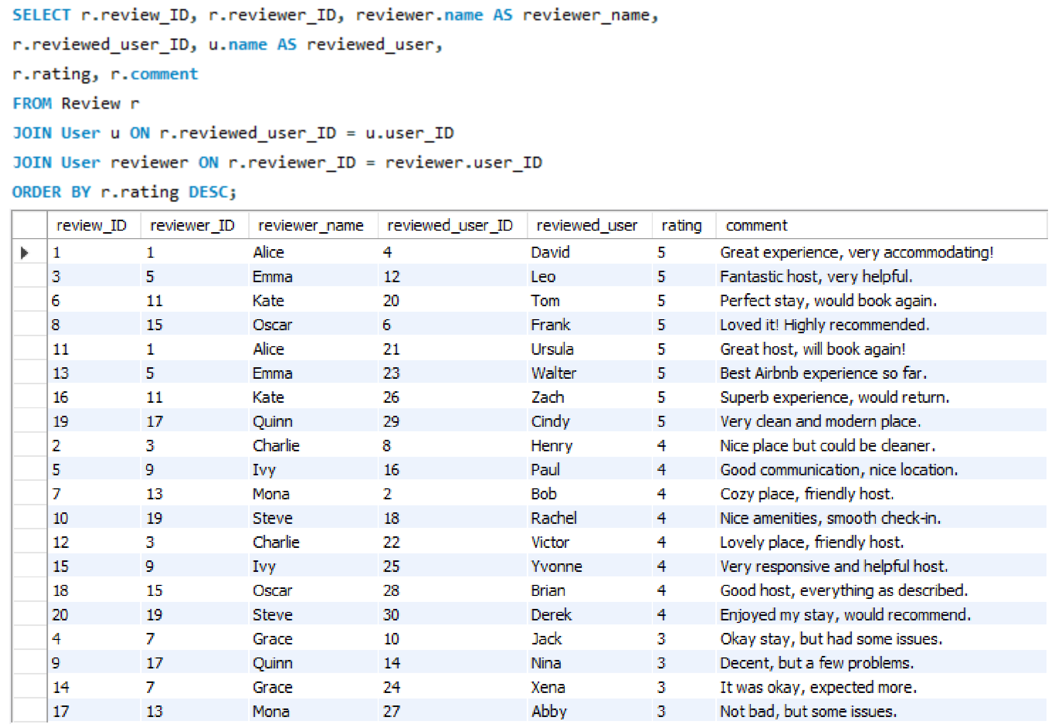
payment_ID	amount	payment_method	payment_status	booking_ID	check_in	check_out	booking_status
1	1500.00	Credit Card	Completed	1	2025-03-10	2025-03-15	Confirmed
2	570.00	PayPal	Completed	2	2025-04-01	2025-04-07	Confirmed
4	375.00	Credit Card	Completed	4	2025-06-05	2025-06-10	Confirmed
6	240.00	Credit Card	Completed	6	2025-08-15	2025-08-18	Confirmed
8	800.00	Credit Card	Completed	8	2025-10-10	2025-10-15	Pending
9	600.00	PayPal	Completed	9	2025-11-01	2025-11-05	Confirmed
12	1750.00	Credit Card	Completed	12	2026-02-05	2026-02-12	Pending
13	600.00	PayPal	Completed	13	2026-03-01	2026-03-06	Confirmed
14	700.00	Credit Card	Completed	14	2026-04-15	2026-04-20	Confirmed
16	810.00	Credit Card	Completed	16	2026-06-25	2026-07-01	Confirmed
18	340.00	Credit Card	Completed	18	2026-08-08	2026-08-12	Confirmed
19	1000.00	Bank Transfer	Completed	19	2026-09-12	2026-09-17	Confirmed

Review Entity

- The first picture shows the statement for creating the Review table
- The attribute review_ID is the primary key, while the attributes reviewer_ID and reviewed_user_ID are foreign keys which both reference the User table
- The test case retrieves all reviews from the Review table along with the reviewer and the reviewed user; the result is sorted by rating in descending order



```
-- -----------------------------------------------------
-- Table `mydb`.`Review`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Review` (
  `review_ID` INT NOT NULL AUTO_INCREMENT,
  `reviewer_ID` INT NULL,
  `reviewed_user_ID` INT NULL,
  `rating` INT NULL,
  `comment` TEXT(65535) NULL,
  `date` TIMESTAMP NULL,
  PRIMARY KEY (`review_ID`),
  INDEX `reviewer_ID_idx` (`reviewer_ID` ASC) VISIBLE,
  INDEX `reviewed_user_ID_idx` (`reviewed_user_ID` ASC) VISIBLE,
  CONSTRAINT `FK_reviewer_user`
    FOREIGN KEY (`reviewer_ID`)
    REFERENCES `mydb`.`User` (`user_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `FK_reviewed_user`
    FOREIGN KEY (`reviewed_user_ID`)
    REFERENCES `mydb`.`User` (`user_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

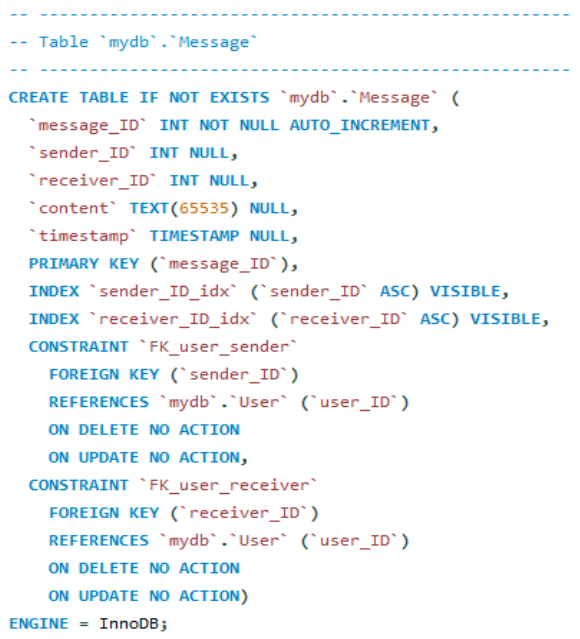


```
SELECT r.review_ID, r.reviewer_ID, reviewer.name AS reviewer_name,
r.reviewed_user_ID, u.name AS reviewed_user,
r.rating, r.comment
FROM Review r
JOIN User u ON r.reviewed_user_ID = u.user_ID
JOIN User reviewer ON r.reviewer_ID = reviewer.user_ID
ORDER BY r.rating DESC;
```

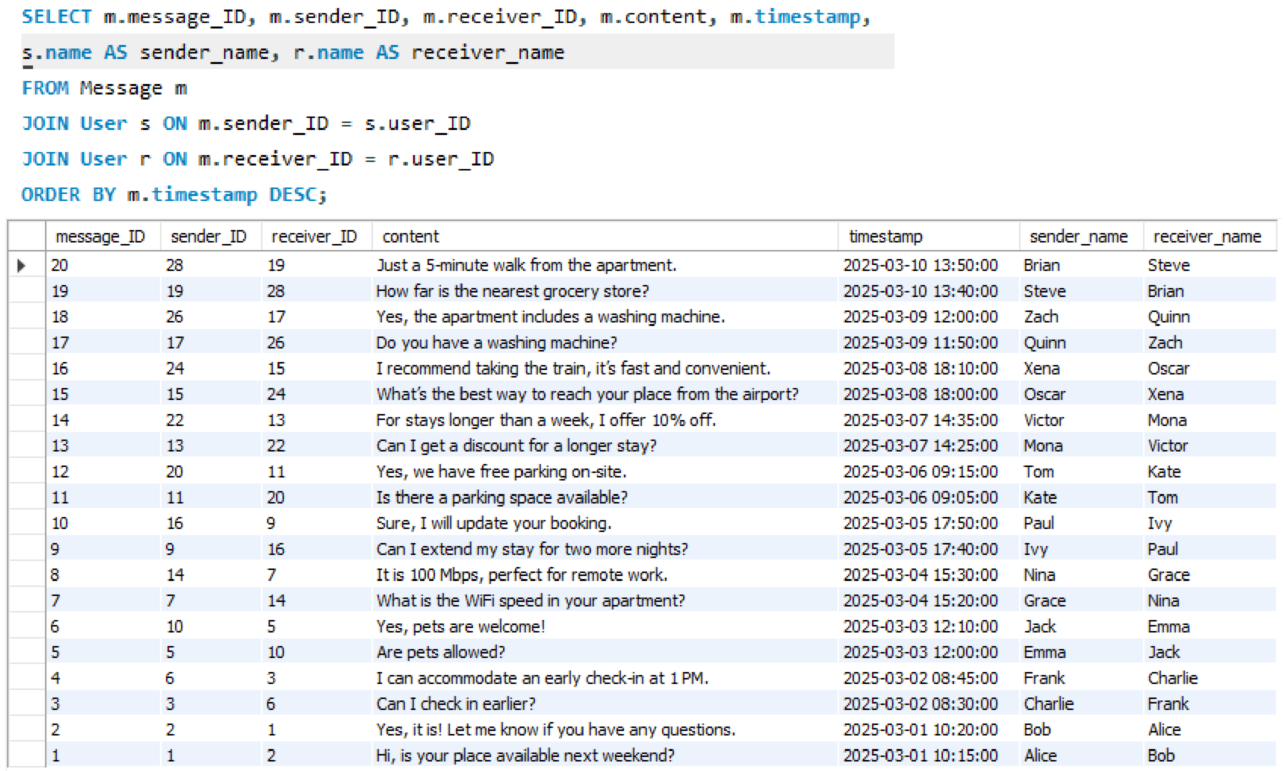
review_ID	reviewer_ID	reviewer_name	reviewed_user_ID	reviewed_user	rating	comment
1	1	Alice	4	David	5	Great experience, very accommodating!
3	5	Emma	12	Leo	5	Fantastic host, very helpful.
6	11	Kate	20	Tom	5	Perfect stay, would book again.
8	15	Oscar	6	Frank	5	Loved it! Highly recommended.
11	1	Alice	21	Ursula	5	Great host, will book again!
13	5	Emma	23	Walter	5	Best Airbnb experience so far.
16	11	Kate	26	Zach	5	Superb experience, would return.
19	17	Quinn	29	Cindy	5	Very clean and modern place.
2	3	Charlie	8	Henry	4	Nice place but could be cleaner.
5	9	Ivy	16	Paul	4	Good communication, nice location.
7	13	Mona	2	Bob	4	Cozy place, friendly host.
10	19	Steve	18	Rachel	4	Nice amenities, smooth check-in.
12	3	Charlie	22	Victor	4	Lovely place, friendly host.
15	9	Ivy	25	Yvonne	4	Very responsive and helpful host.
18	15	Oscar	28	Brian	4	Good host, everything as described.
20	19	Steve	30	Derek	4	Enjoyed my stay, would recommend.
4	7	Grace	10	Jack	3	Okay stay, but had some issues.
9	17	Quinn	14	Nina	3	Decent, but a few problems.
14	7	Grace	24	Xena	3	It was okay, expected more.
17	13	Mona	27	Abby	3	Not bad, but some issues.

Message Entity

- The first picture shows the statement for creating the Message table
- The attribute message_ID is the primary key, while the attributes sender_ID and receiver_ID are foreign keys which both reference the User table
- The test case retrieves all messages from the Message table and joins the User table twice to display the names of the sender and receiver; the result is sorted in descending order, i.e., the most recent message is displayed first



```sql
-- -----------------------------------------------------
-- Table `mydb`.`Message`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Message` (
  `message_ID` INT NOT NULL AUTO_INCREMENT,
  `sender_ID` INT NULL,
  `receiver_ID` INT NULL,
  `content` TEXT(65535) NULL,
  `timestamp` TIMESTAMP NULL,
  PRIMARY KEY (`message_ID`),
  INDEX `sender_ID_idx` (`sender_ID` ASC) VISIBLE,
  INDEX `receiver_ID_idx` (`receiver_ID` ASC) VISIBLE,
  CONSTRAINT `FK_user_sender`
    FOREIGN KEY (`sender_ID`)
    REFERENCES `mydb`.`User` (`user_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `FK_user_receiver`
    FOREIGN KEY (`receiver_ID`)
    REFERENCES `mydb`.`User` (`user_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

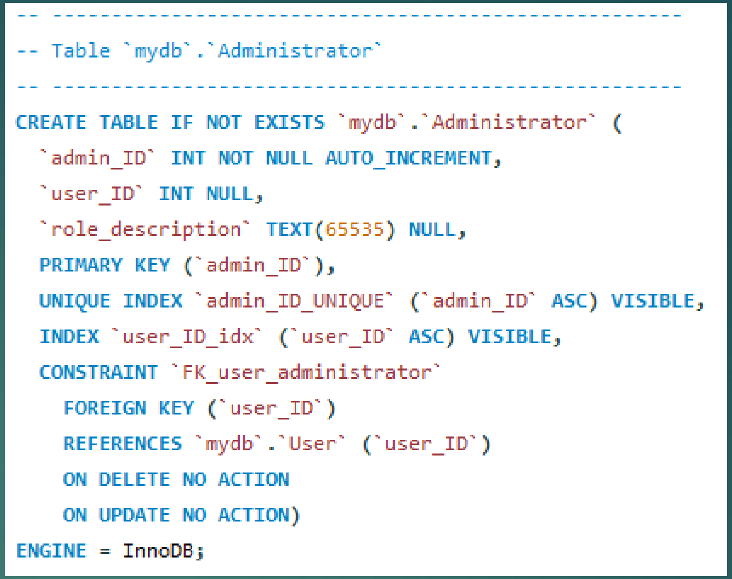


```sql
SELECT m.message_ID, m.sender_ID, m.receiver_ID, m.content, m.timestamp,
s.name AS sender_name, r.name AS receiver_name
FROM Message m
JOIN User s ON m.sender_ID = s.user_ID
JOIN User r ON m.receiver_ID = r.user_ID
ORDER BY m.timestamp DESC;
```

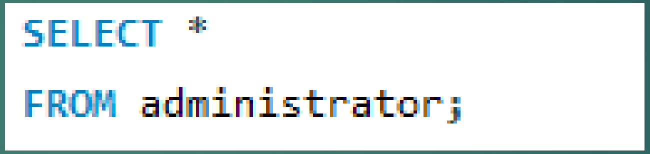
message_ID	sender_ID	receiver_ID	content	timestamp	sender_name	receiver_name
20	28	19	Just a 5-minute walk from the apartment.	2025-03-10 13:50:00	Brian	Steve
19	19	28	How far is the nearest grocery store?	2025-03-10 13:40:00	Steve	Brian
18	26	17	Yes, the apartment includes a washing machine.	2025-03-09 12:00:00	Zach	Quinn
17	17	26	Do you have a washing machine?	2025-03-09 11:50:00	Quinn	Zach
16	24	15	I recommend taking the train, it's fast and convenient.	2025-03-08 18:10:00	Xena	Oscar
15	15	24	What's the best way to reach your place from the airport?	2025-03-08 18:00:00	Oscar	Xena
14	22	13	For stays longer than a week, I offer 10% off.	2025-03-07 14:35:00	Victor	Mona
13	13	22	Can I get a discount for a longer stay?	2025-03-07 14:25:00	Mona	Victor
12	20	11	Yes, we have free parking on-site.	2025-03-06 09:15:00	Tom	Kate
11	11	20	Is there a parking space available?	2025-03-06 09:05:00	Kate	Tom
10	16	9	Sure, I will update your booking.	2025-03-05 17:50:00	Paul	Ivy
9	9	16	Can I extend my stay for two more nights?	2025-03-05 17:40:00	Ivy	Paul
8	14	7	It is 100 Mbps, perfect for remote work.	2025-03-04 15:30:00	Nina	Grace
7	7	14	What is the WiFi speed in your apartment?	2025-03-04 15:20:00	Grace	Nina
6	10	5	Yes, pets are welcome!	2025-03-03 12:10:00	Jack	Emma
5	5	10	Are pets allowed?	2025-03-03 12:00:00	Emma	Jack
4	6	3	I can accommodate an early check-in at 1 PM.	2025-03-02 08:45:00	Frank	Charlie
3	3	6	Can I check in earlier?	2025-03-02 08:30:00	Charlie	Frank
2	2	1	Yes, it is! Let me know if you have any questions.	2025-03-01 10:20:00	Bob	Alice
1	1	2	Hi, is your place available next weekend?	2025-03-01 10:15:00	Alice	Bob

Administrator Entity

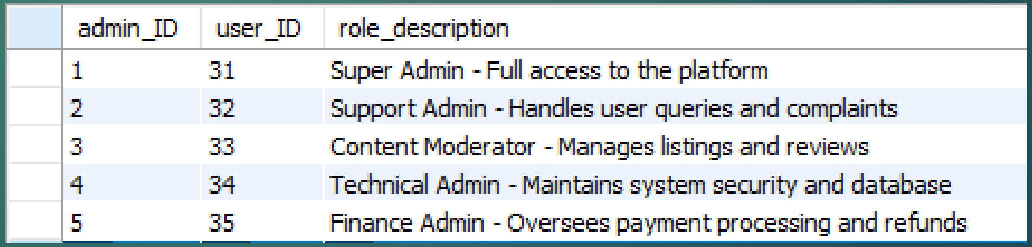
- The first picture shows the statement for creating the Administrator table
- The attribute admin_ID is the primary key of the Administrator table, while user_ID is the foreign key referencing the User table
- The test case retrieves all records from the Administrator table



```
-- -----------------------------------------------------
-- Table `mydb`.`Administrator`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Administrator` (
  `admin_ID` INT NOT NULL AUTO_INCREMENT,
  `user_ID` INT NULL,
  `role_description` TEXT(65535) NULL,
  PRIMARY KEY (`admin_ID`),
  UNIQUE INDEX `admin_ID_UNIQUE` (`admin_ID` ASC) VISIBLE,
  INDEX `user_ID_idx` (`user_ID` ASC) VISIBLE,
  CONSTRAINT `FK_user_administrator`
    FOREIGN KEY (`user_ID`)
    REFERENCES `mydb`.`User` (`user_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```



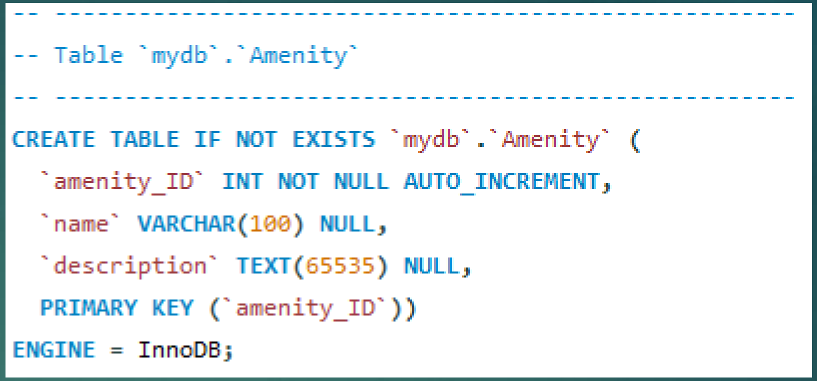
```
SELECT *
FROM administrator;
```



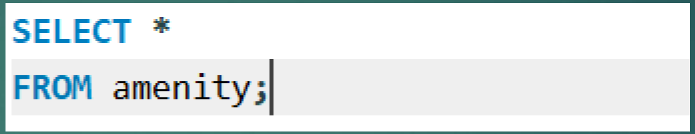
admin_ID	user_ID	role_description
1	31	Super Admin - Full access to the platform
2	32	Support Admin - Handles user queries and complaints
3	33	Content Moderator - Manages listings and reviews
4	34	Technical Admin - Maintains system security and database
5	35	Finance Admin - Oversees payment processing and refunds

Amenity Entity

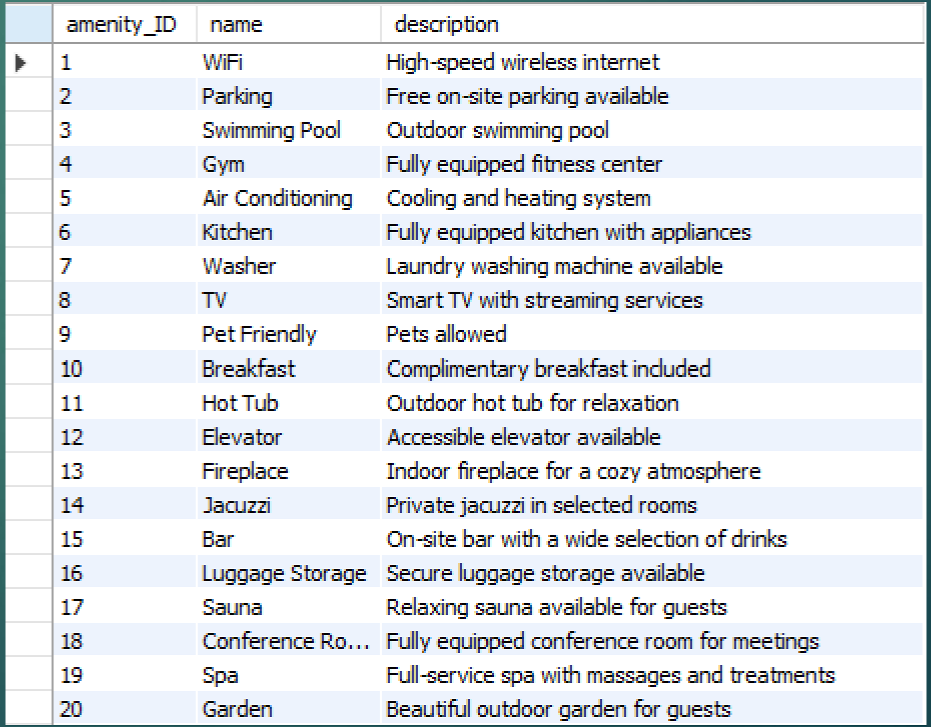
- The first picture shows the statement for creating the Amenity table
- The attribute amenity_ID is the primary key of the Amenity table
- The test case retrieves all records from the Amenity table



```
-- -----------------------------------------------------
-- Table `mydb`.`Amenity`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Amenity` (
  `amenity_ID` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(100) NULL,
  `description` TEXT(65535) NULL,
  PRIMARY KEY (`amenity_ID`))
ENGINE = InnoDB;
```



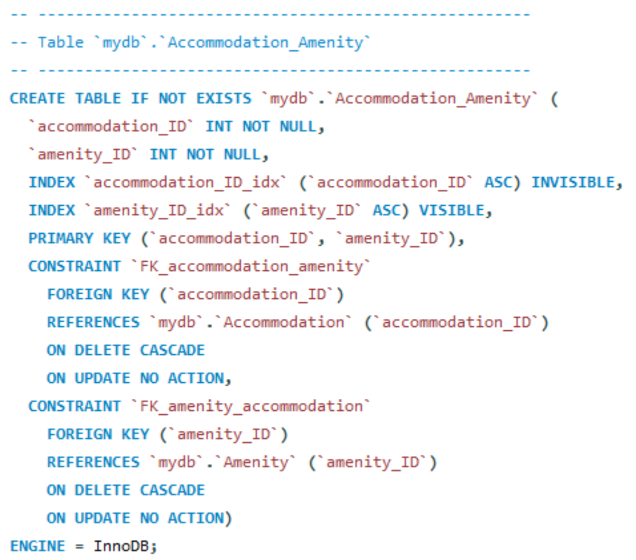
```
SELECT *
FROM amenity;
```



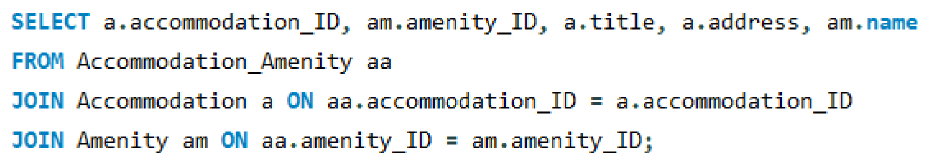
amenity_ID	name	description
1	WiFi	High-speed wireless internet
2	Parking	Free on-site parking available
3	Swimming Pool	Outdoor swimming pool
4	Gym	Fully equipped fitness center
5	Air Conditioning	Cooling and heating system
6	Kitchen	Fully equipped kitchen with appliances
7	Washer	Laundry washing machine available
8	TV	Smart TV with streaming services
9	Pet Friendly	Pets allowed
10	Breakfast	Complimentary breakfast included
11	Hot Tub	Outdoor hot tub for relaxation
12	Elevator	Accessible elevator available
13	Fireplace	Indoor fireplace for a cozy atmosphere
14	Jacuzzi	Private jacuzzi in selected rooms
15	Bar	On-site bar with a wide selection of drinks
16	Luggage Storage	Secure luggage storage available
17	Sauna	Relaxing sauna available for guests
18	Conference Ro...	Fully equipped conference room for meetings
19	Spa	Full-service spa with massages and treatments
20	Garden	Beautiful outdoor garden for guests

Accommodation_Amenity Entity

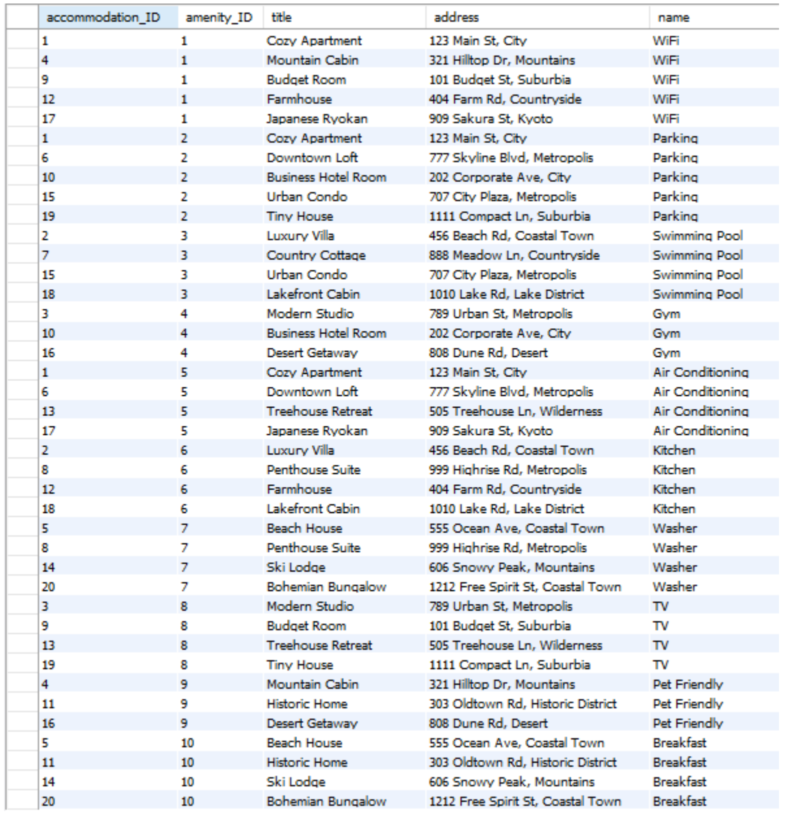
- The first picture shows the statement for creating the Accommodation_Amenity table
- The primary keys in this table (accommodation_ID and amenity_ID) are also foreign keys that reference the Accommodation and Amenity tables, respectively; This establishes a many-to-many relationship between accommodations and their amenities
- The test case retrieves the accommodations and their associated amenities



```
-- -----------------------------------------------------
-- Table `mydb`.`Accommodation_Amenity`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Accommodation_Amenity` (
  `accommodation_ID` INT NOT NULL,
  `amenity_ID` INT NOT NULL,
  INDEX `accommodation_ID_idx` (`accommodation_ID` ASC) INVISIBLE,
  INDEX `amenity_ID_idx` (`amenity_ID` ASC) VISIBLE,
  PRIMARY KEY (`accommodation_ID`, `amenity_ID`),
  CONSTRAINT `FK_accommodation_amenity`
    FOREIGN KEY (`accommodation_ID`)
    REFERENCES `mydb`.`Accommodation` (`accommodation_ID`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION,
  CONSTRAINT `FK_amenity_accommodation`
    FOREIGN KEY (`amenity_ID`)
    REFERENCES `mydb`.`Amenity` (`amenity_ID`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```



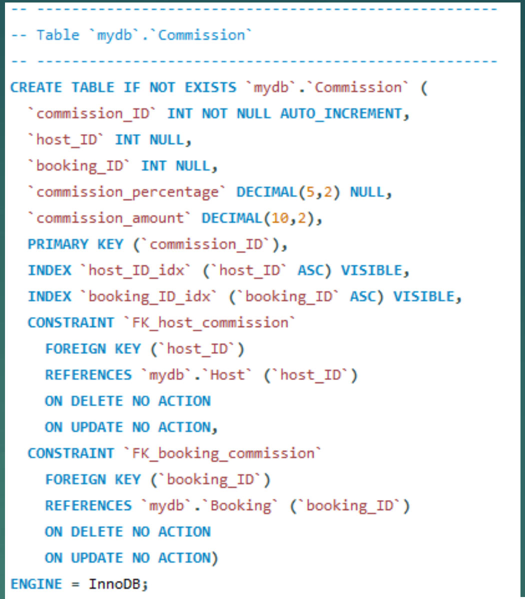
```
SELECT a.accommodation_ID, am.amenity_ID, a.title, a.address, am.name
FROM Accommodation_Amenity aa
JOIN Accommodation a ON aa.accommodation_ID = a.accommodation_ID
JOIN Amenity am ON aa.amenity_ID = am.amenity_ID;
```



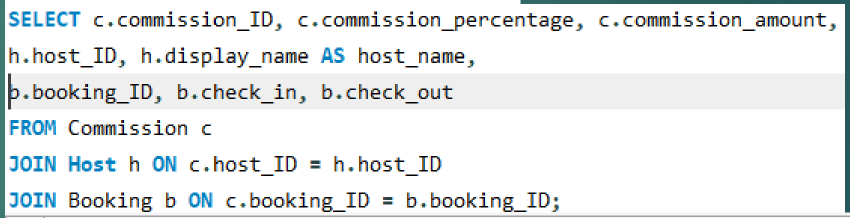
accommodation_ID	amenity_ID	title	address	name
1	1	Cozy Apartment	123 Main St, City	WiFi
4	1	Mountain Cabin	321 Hilltop Dr, Mountains	WiFi
9	1	Budget Room	101 Budget St, Suburbia	WiFi
12	1	Farmhouse	404 Farm Rd, Countryside	WiFi
17	1	Japanese Ryokan	909 Sakura St, Kyoto	WiFi
1	2	Cozy Apartment	123 Main St, City	Parking
6	2	Downtown Loft	777 Skyline Blvd, Metropolis	Parking
10	2	Business Hotel Room	202 Corporate Ave, City	Parking
15	2	Urban Condo	707 City Plaza, Metropolis	Parking
19	2	Tiny House	1111 Compact Ln, Suburbia	Parking
2	3	Luxury Villa	456 Beach Rd, Coastal Town	Swimming Pool
7	3	Country Cottage	888 Meadow Ln, Countryside	Swimming Pool
15	3	Urban Condo	707 City Plaza, Metropolis	Swimming Pool
18	3	Lakefront Cabin	1010 Lake Rd, Lake District	Swimming Pool
3	4	Modern Studio	789 Urban St, Metropolis	Gym
10	4	Business Hotel Room	202 Corporate Ave, City	Gym
16	4	Desert Getaway	808 Dune Rd, Desert	Gym
1	5	Cozy Apartment	123 Main St, City	Air Conditioning
6	5	Downtown Loft	777 Skyline Blvd, Metropolis	Air Conditioning
13	5	Treehouse Retreat	505 Treehouse Ln, Wilderness	Air Conditioning
17	5	Japanese Ryokan	909 Sakura St, Kyoto	Air Conditioning
2	6	Luxury Villa	456 Beach Rd, Coastal Town	Kitchen
8	6	Penthouse Suite	999 Highrise Rd, Metropolis	Kitchen
12	6	Farmhouse	404 Farm Rd, Countryside	Kitchen
18	6	Lakefront Cabin	1010 Lake Rd, Lake District	Kitchen
5	7	Beach House	555 Ocean Ave, Coastal Town	Washer
8	7	Penthouse Suite	999 Highrise Rd, Metropolis	Washer
14	7	Ski Lodge	606 Snowy Peak, Mountains	Washer
20	7	Bohemian Bungalow	1212 Free Spirit St, Coastal Town	Washer
3	8	Modern Studio	789 Urban St, Metropolis	TV
9	8	Budget Room	101 Budget St, Suburbia	TV
13	8	Treehouse Retreat	505 Treehouse Ln, Wilderness	TV
19	8	Tiny House	1111 Compact Ln, Suburbia	TV
4	9	Mountain Cabin	321 Hilltop Dr, Mountains	Pet Friendly
11	9	Historic Home	303 Oldtown Rd, Historic District	Pet Friendly
16	9	Desert Getaway	808 Dune Rd, Desert	Pet Friendly
5	10	Beach House	555 Ocean Ave, Coastal Town	Breakfast
11	10	Historic Home	303 Oldtown Rd, Historic District	Breakfast
14	10	Ski Lodge	606 Snowy Peak, Mountains	Breakfast
20	10	Bohemian Bungalow	1212 Free Spirit St, Coastal Town	Breakfast

Commission Entity

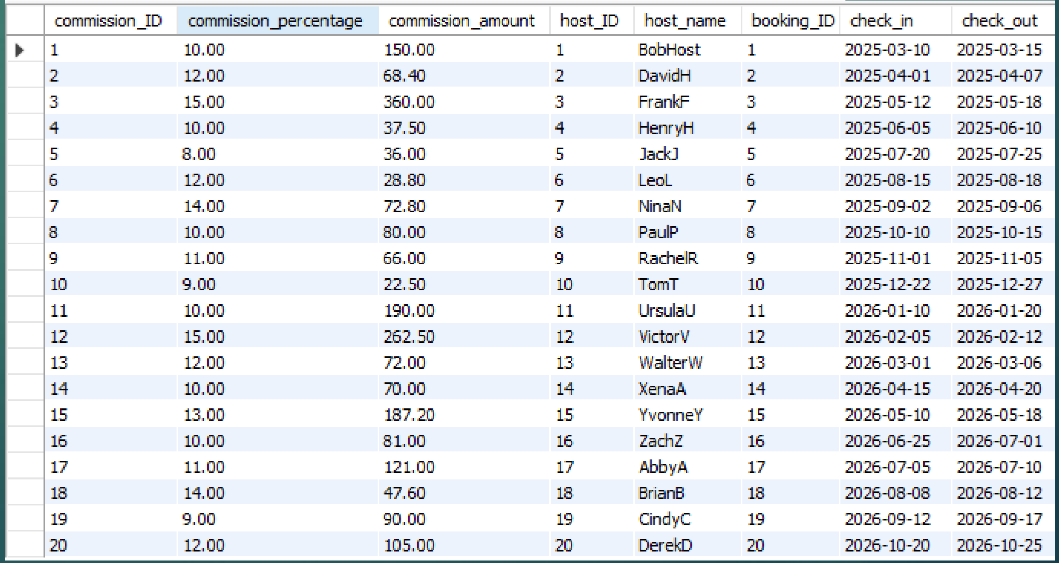
- The first picture shows the statement for creating the Commission table
- The attribute commission_ID is the primary key, while host_ID and booking_ID are foreign key which reference the Host and Booking tables, respectively
- The test case retrieves commission details along with associated hosts and bookings



```
-- -----------------------------------------------------
-- Table `mydb`.`Commission`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Commission` (
  `commission_ID` INT NOT NULL AUTO_INCREMENT,
  `host_ID` INT NULL,
  `booking_ID` INT NULL,
  `commission_percentage` DECIMAL(5,2) NULL,
  `commission_amount` DECIMAL(10,2),
  PRIMARY KEY (`commission_ID`),
  INDEX `host_ID_idx` (`host_ID` ASC) VISIBLE,
  INDEX `booking_ID_idx` (`booking_ID` ASC) VISIBLE,
  CONSTRAINT `FK_host_commission`
    FOREIGN KEY (`host_ID`)
    REFERENCES `mydb`.`Host` (`host_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `FK_booking_commission`
    FOREIGN KEY (`booking_ID`)
    REFERENCES `mydb`.`Booking` (`booking_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```



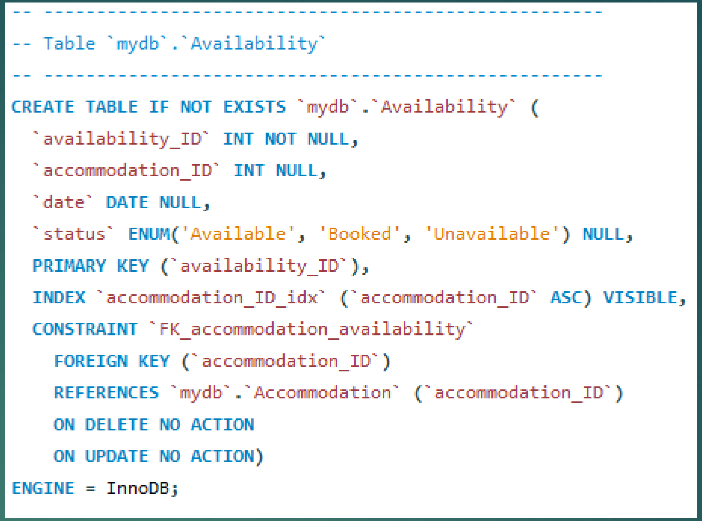
```
SELECT c.commission_ID, c.commission_percentage, c.commission_amount,
h.host_ID, h.display_name AS host_name,
b.booking_ID, b.check_in, b.check_out
FROM Commission c
JOIN Host h ON c.host_ID = h.host_ID
JOIN Booking b ON c.booking_ID = b.booking_ID;
```



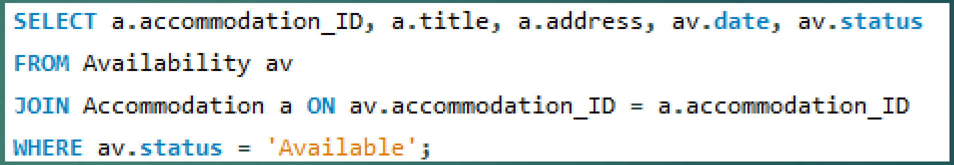
commission_ID	commission_percentage	commission_amount	host_ID	host_name	booking_ID	check_in	check_out
1	10.00	150.00	1	BobHost	1	2025-03-10	2025-03-15
2	12.00	68.40	2	DavidH	2	2025-04-01	2025-04-07
3	15.00	360.00	3	FrankF	3	2025-05-12	2025-05-18
4	10.00	37.50	4	HenryH	4	2025-06-05	2025-06-10
5	8.00	36.00	5	JackJ	5	2025-07-20	2025-07-25
6	12.00	28.80	6	LeoL	6	2025-08-15	2025-08-18
7	14.00	72.80	7	NinaN	7	2025-09-02	2025-09-06
8	10.00	80.00	8	PaulP	8	2025-10-10	2025-10-15
9	11.00	66.00	9	RachelR	9	2025-11-01	2025-11-05
10	9.00	22.50	10	TomT	10	2025-12-22	2025-12-27
11	10.00	190.00	11	UrsulaU	11	2026-01-10	2026-01-20
12	15.00	262.50	12	VictorV	12	2026-02-05	2026-02-12
13	12.00	72.00	13	WalterW	13	2026-03-01	2026-03-06
14	10.00	70.00	14	XenaA	14	2026-04-15	2026-04-20
15	13.00	187.20	15	YvonneY	15	2026-05-10	2026-05-18
16	10.00	81.00	16	ZachZ	16	2026-06-25	2026-07-01
17	11.00	121.00	17	AbbyA	17	2026-07-05	2026-07-10
18	14.00	47.60	18	BrianB	18	2026-08-08	2026-08-12
19	9.00	90.00	19	CindyC	19	2026-09-12	2026-09-17
20	12.00	105.00	20	DerekD	20	2026-10-20	2026-10-25

Availability Entity

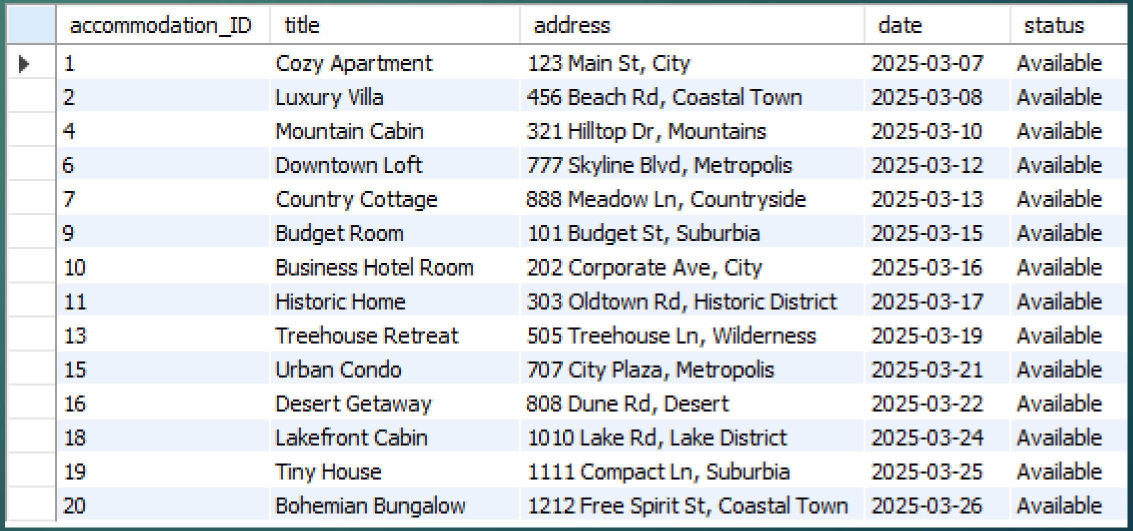
- The first picture shows the statement for creating the Availability table
- The attribute availability_ID is the primary key, while accommodation_ID is the foreign key which references the Accommodation table
- The test case retrieves records from the Availability table, joining it with the Accommodation table; The results are filtered so that only accommodations with the availability status of ‘Available’ are shown



```sql
-- -----------------------------------------------------
-- Table `mydb`.`Availability`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Availability` (
  `availability_ID` INT NOT NULL,
  `accommodation_ID` INT NULL,
  `date` DATE NULL,
  `status` ENUM('Available', 'Booked', 'Unavailable') NULL,
  PRIMARY KEY (`availability_ID`),
  INDEX `accommodation_ID_idx` (`accommodation_ID` ASC) VISIBLE,
  CONSTRAINT `FK_accommodation_availability`
    FOREIGN KEY (`accommodation_ID`)
    REFERENCES `mydb`.`Accommodation` (`accommodation_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```



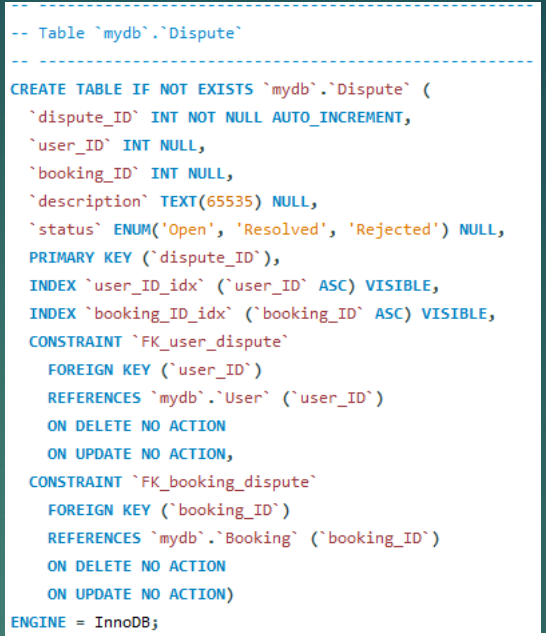
```sql
SELECT a.accommodation_ID, a.title, a.address, av.date, av.status
FROM Availability av
JOIN Accommodation a ON av.accommodation_ID = a.accommodation_ID
WHERE av.status = 'Available';
```



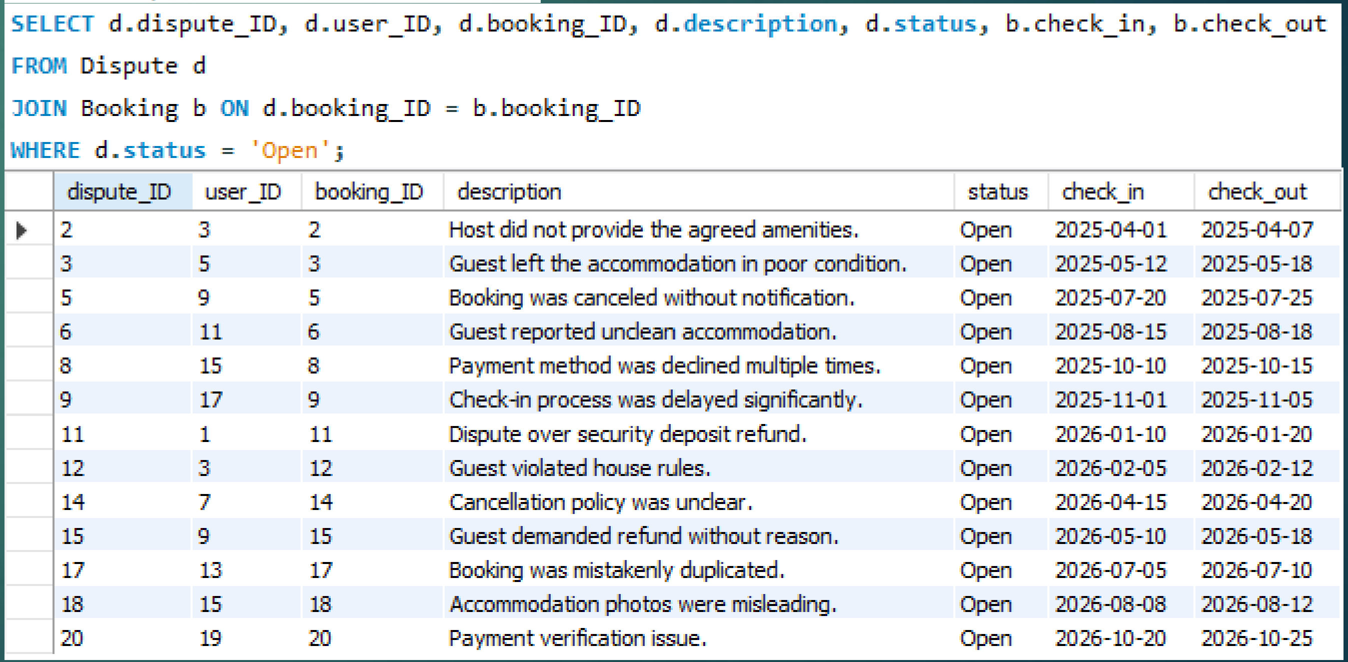
accommodation_ID	title	address	date	status
1	Cozy Apartment	123 Main St, City	2025-03-07	Available
2	Luxury Villa	456 Beach Rd, Coastal Town	2025-03-08	Available
4	Mountain Cabin	321 Hilltop Dr, Mountains	2025-03-10	Available
6	Downtown Loft	777 Skyline Blvd, Metropolis	2025-03-12	Available
7	Country Cottage	888 Meadow Ln, Countryside	2025-03-13	Available
9	Budget Room	101 Budget St, Suburbia	2025-03-15	Available
10	Business Hotel Room	202 Corporate Ave, City	2025-03-16	Available
11	Historic Home	303 Oldtown Rd, Historic District	2025-03-17	Available
13	Treehouse Retreat	505 Treehouse Ln, Wilderness	2025-03-19	Available
15	Urban Condo	707 City Plaza, Metropolis	2025-03-21	Available
16	Desert Getaway	808 Dune Rd, Desert	2025-03-22	Available
18	Lakefront Cabin	1010 Lake Rd, Lake District	2025-03-24	Available
19	Tiny House	1111 Compact Ln, Suburbia	2025-03-25	Available
20	Bohemian Bungalow	1212 Free Spirit St, Coastal Town	2025-03-26	Available

Dispute Entity

- The first picture shows the statement for creating the Dispute table
- The attribute dispute_ID is the primary key, while user_ID and booking_ID are foreign keys which reference the User and Booking tables, respectively
- The test case retrieves the details of disputes from the Dispute table, joining with the Booking table to include relevant booking information; The results are filtered so that only disputes with an ‘Open’ status are shown



```
-- -----------------------------------------------------
-- Table `mydb`.`Dispute`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Dispute` (
  `dispute_ID` INT NOT NULL AUTO_INCREMENT,
  `user_ID` INT NULL,
  `booking_ID` INT NULL,
  `description` TEXT(65535) NULL,
  `status` ENUM('Open', 'Resolved', 'Rejected') NULL,
  PRIMARY KEY (`dispute_ID`),
  INDEX `user_ID_idx` (`user_ID` ASC) VISIBLE,
  INDEX `booking_ID_idx` (`booking_ID` ASC) VISIBLE,
  CONSTRAINT `FK_user_dispute`
    FOREIGN KEY (`user_ID`)
    REFERENCES `mydb`.`User` (`user_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `FK_booking_dispute`
    FOREIGN KEY (`booking_ID`)
    REFERENCES `mydb`.`Booking` (`booking_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

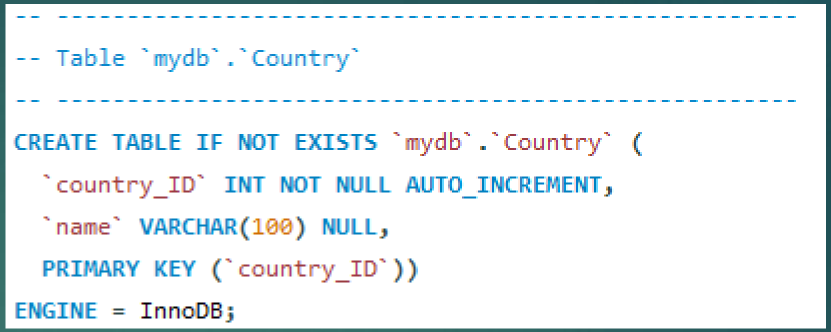


```
SELECT d.dispute_ID, d.user_ID, d.booking_ID, d.description, d.status, b.check_in, b.check_out
FROM Dispute d
JOIN Booking b ON d.booking_ID = b.booking_ID
WHERE d.status = 'Open';
```

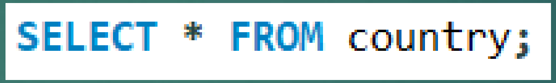
dispute_ID	user_ID	booking_ID	description	status	check_in	check_out
2	3	2	Host did not provide the agreed amenities.	Open	2025-04-01	2025-04-07
3	5	3	Guest left the accommodation in poor condition.	Open	2025-05-12	2025-05-18
5	9	5	Booking was canceled without notification.	Open	2025-07-20	2025-07-25
6	11	6	Guest reported unclean accommodation.	Open	2025-08-15	2025-08-18
8	15	8	Payment method was declined multiple times.	Open	2025-10-10	2025-10-15
9	17	9	Check-in process was delayed significantly.	Open	2025-11-01	2025-11-05
11	1	11	Dispute over security deposit refund.	Open	2026-01-10	2026-01-20
12	3	12	Guest violated house rules.	Open	2026-02-05	2026-02-12
14	7	14	Cancellation policy was unclear.	Open	2026-04-15	2026-04-20
15	9	15	Guest demanded refund without reason.	Open	2026-05-10	2026-05-18
17	13	17	Booking was mistakenly duplicated.	Open	2026-07-05	2026-07-10
18	15	18	Accommodation photos were misleading.	Open	2026-08-08	2026-08-12
20	19	20	Payment verification issue.	Open	2026-10-20	2026-10-25

Country Entity

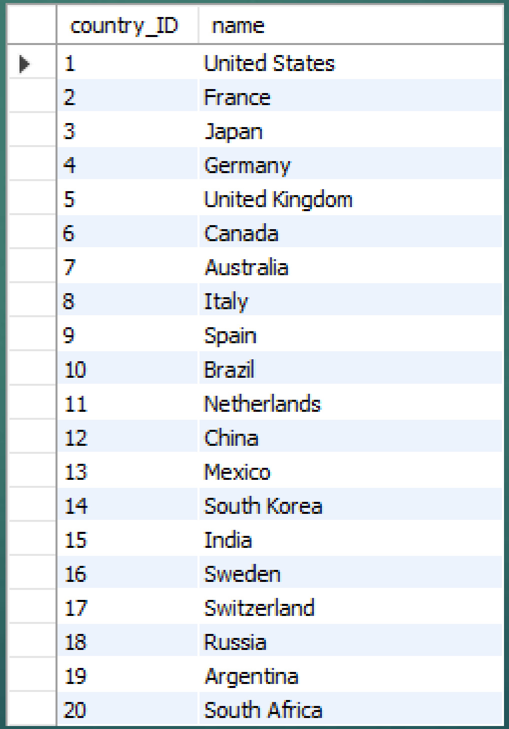
- The first picture shows the statement for creating the Country table
- The attribute country_ID is the primary key of the Country table
- The test case retrieves all records from the Country table



```
-- -----------------------------------------------------
-- Table `mydb`.`Country`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Country` (
  `country_ID` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(100) NULL,
  PRIMARY KEY (`country_ID`))
ENGINE = InnoDB;
```



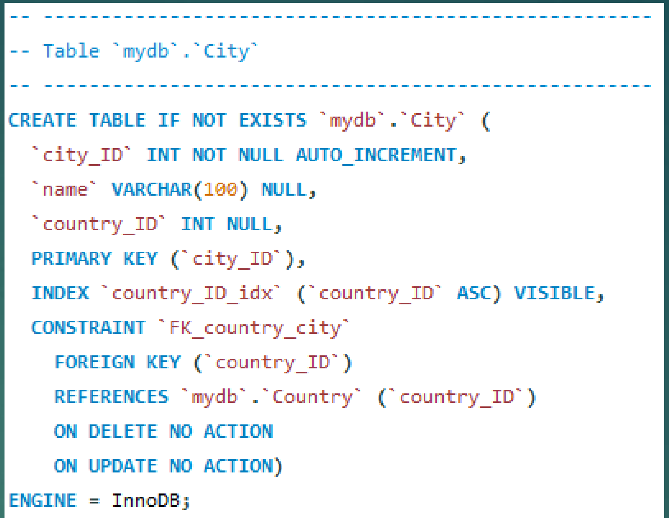
```
SELECT * FROM country;
```



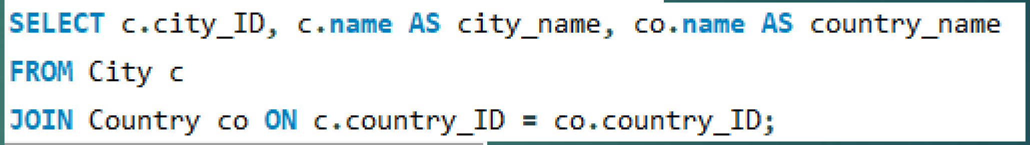
country_ID	name
1	United States
2	France
3	Japan
4	Germany
5	United Kingdom
6	Canada
7	Australia
8	Italy
9	Spain
10	Brazil
11	Netherlands
12	China
13	Mexico
14	South Korea
15	India
16	Sweden
17	Switzerland
18	Russia
19	Argentina
20	South Africa

City Entity

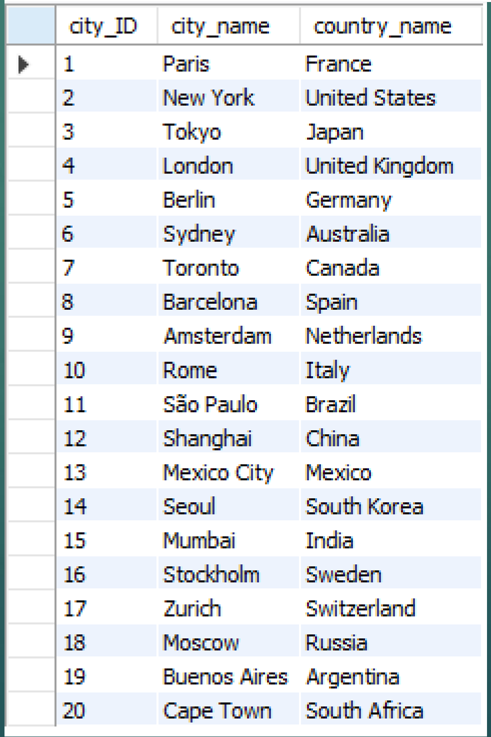
- The first picture shows the statement for creating the City table
- The attribute city_ID is the primary key, while country_ID is the foreign key referencing the Country table
- The test case retrieves city details along with the associated country



```
-- -----------------------------------------------------
-- Table `mydb`.`City`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`City` (
  `city_ID` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(100) NULL,
  `country_ID` INT NULL,
  PRIMARY KEY (`city_ID`),
  INDEX `country_ID_idx` (`country_ID` ASC) VISIBLE,
  CONSTRAINT `FK_country_city`
    FOREIGN KEY (`country_ID`)
    REFERENCES `mydb`.`Country` (`country_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```



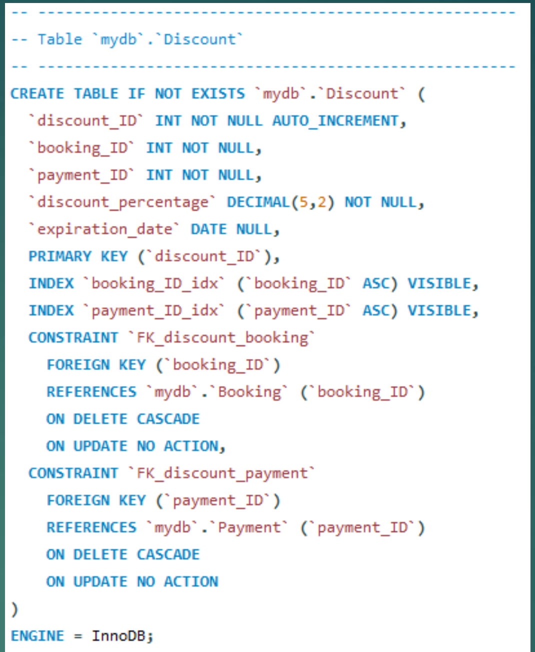
```
SELECT c.city_ID, c.name AS city_name, co.name AS country_name
FROM City c
JOIN Country co ON c.country_ID = co.country_ID;
```



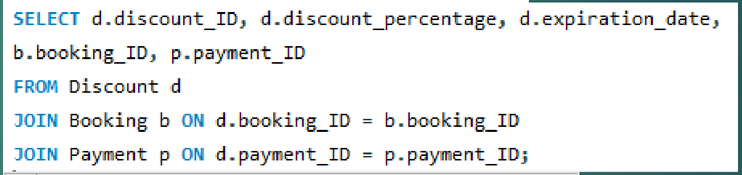
city_ID	city_name	country_name
1	Paris	France
2	New York	United States
3	Tokyo	Japan
4	London	United Kingdom
5	Berlin	Germany
6	Sydney	Australia
7	Toronto	Canada
8	Barcelona	Spain
9	Amsterdam	Netherlands
10	Rome	Italy
11	São Paulo	Brazil
12	Shanghai	China
13	Mexico City	Mexico
14	Seoul	South Korea
15	Mumbai	India
16	Stockholm	Sweden
17	Zurich	Switzerland
18	Moscow	Russia
19	Buenos Aires	Argentina
20	Cape Town	South Africa

Discount Entity

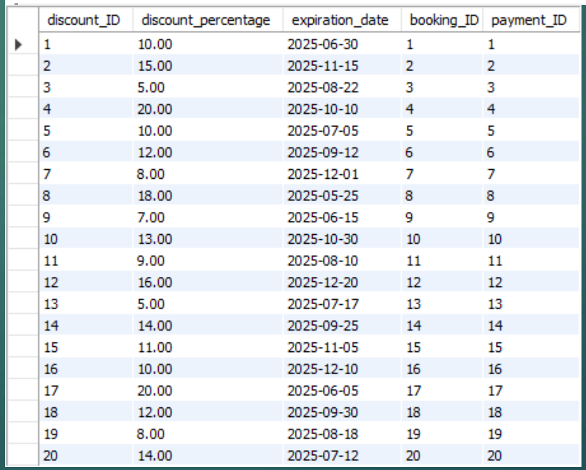
- The first picture shows the statement for creating the Discount table
- The attribute discount_ID is the primary key, while booking_ID and payment_ID are foreign keys which reference the tables Booking and Payment, respectively
- The test case retrieves discount details from the Discount table, joining it with the Booking and Payment tables to associate each discount with its respective booking and payment



```
-- -----------------------------------------------------
-- Table `mydb`.`Discount`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Discount` (
  `discount_ID` INT NOT NULL AUTO_INCREMENT,
  `booking_ID` INT NOT NULL,
  `payment_ID` INT NOT NULL,
  `discount_percentage` DECIMAL(5,2) NOT NULL,
  `expiration_date` DATE NULL,
  PRIMARY KEY (`discount_ID`),
  INDEX `booking_ID_idx` (`booking_ID` ASC) VISIBLE,
  INDEX `payment_ID_idx` (`payment_ID` ASC) VISIBLE,
  CONSTRAINT `FK_discount_booking`
    FOREIGN KEY (`booking_ID`)
    REFERENCES `mydb`.`Booking` (`booking_ID`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION,
  CONSTRAINT `FK_discount_payment`
    FOREIGN KEY (`payment_ID`)
    REFERENCES `mydb`.`Payment` (`payment_ID`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION
)
ENGINE = InnoDB;
```



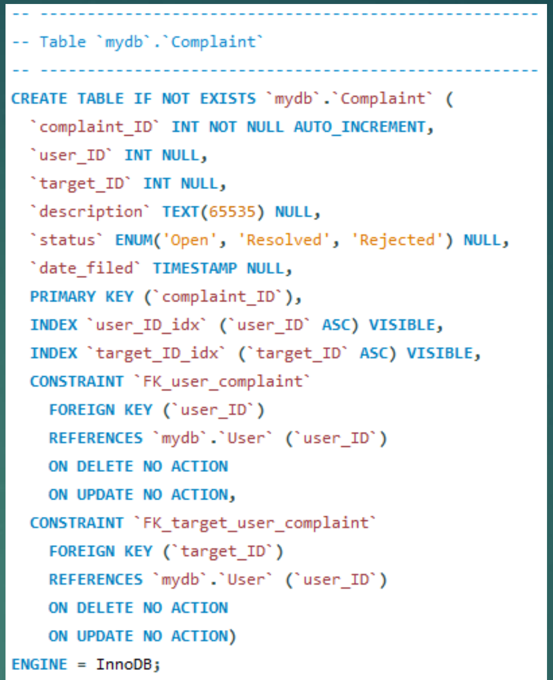
```
SELECT d.discount_ID, d.discount_percentage, d.expiration_date,
b.booking_ID, p.payment_ID
FROM Discount d
JOIN Booking b ON d.booking_ID = b.booking_ID
JOIN Payment p ON d.payment_ID = p.payment_ID;
```



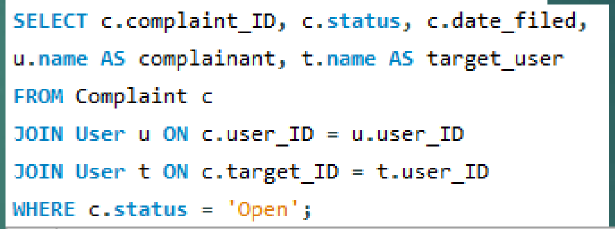
discount_ID	discount_percentage	expiration_date	booking_ID	payment_ID
1	10.00	2025-06-30	1	1
2	15.00	2025-11-15	2	2
3	5.00	2025-08-22	3	3
4	20.00	2025-10-10	4	4
5	10.00	2025-07-05	5	5
6	12.00	2025-09-12	6	6
7	8.00	2025-12-01	7	7
8	18.00	2025-05-25	8	8
9	7.00	2025-06-15	9	9
10	13.00	2025-10-30	10	10
11	9.00	2025-08-10	11	11
12	16.00	2025-12-20	12	12
13	5.00	2025-07-17	13	13
14	14.00	2025-09-25	14	14
15	11.00	2025-11-05	15	15
16	10.00	2025-12-10	16	16
17	20.00	2025-06-05	17	17
18	12.00	2025-09-30	18	18
19	8.00	2025-08-18	19	19
20	14.00	2025-07-12	20	20

Complaint Entity

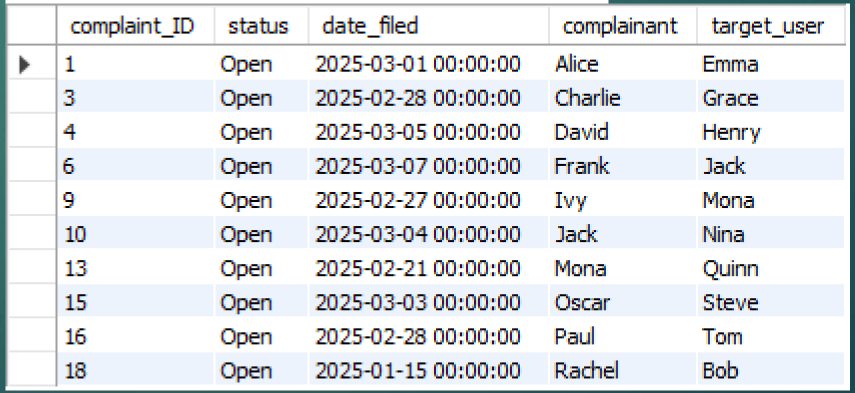
- The first picture shows the statement for creating the Complaint table
- The attribute complaint_ID is the primary key, while user_ID and target_ID are foreign keys which both reference the User table
- The test case retrieves the details of complaints that are currently open, joining the Complaint table with the User table twice to retrieve the names of both the complainant and the target user



```
-- -----------------------------------------------------
-- Table `mydb`.`Complaint`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Complaint` (
  `complaint_ID` INT NOT NULL AUTO_INCREMENT,
  `user_ID` INT NULL,
  `target_ID` INT NULL,
  `description` TEXT(65535) NULL,
  `status` ENUM('Open', 'Resolved', 'Rejected') NULL,
  `date_filed` TIMESTAMP NULL,
  PRIMARY KEY (`complaint_ID`),
  INDEX `user_ID_idx` (`user_ID` ASC) VISIBLE,
  INDEX `target_ID_idx` (`target_ID` ASC) VISIBLE,
  CONSTRAINT `FK_user_complaint`
    FOREIGN KEY (`user_ID`)
    REFERENCES `mydb`.`User` (`user_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `FK_target_user_complaint`
    FOREIGN KEY (`target_ID`)
    REFERENCES `mydb`.`User` (`user_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```



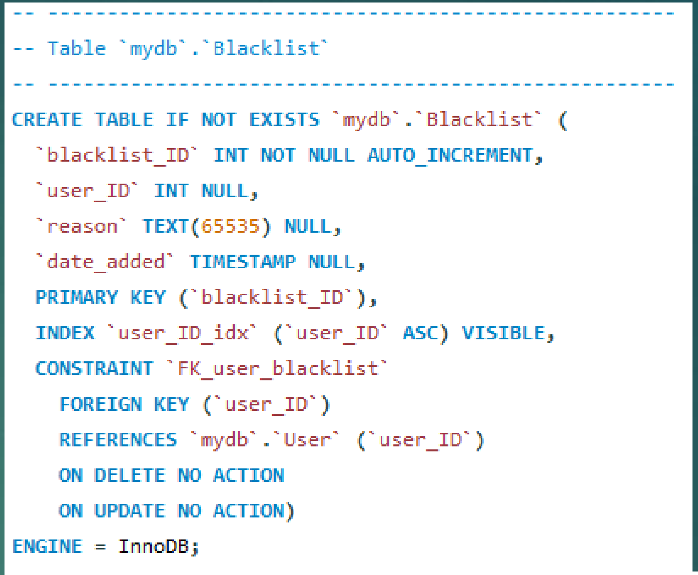
```
SELECT c.complaint_ID, c.status, c.date_filed,
u.name AS complainant, t.name AS target_user
FROM Complaint c
JOIN User u ON c.user_ID = u.user_ID
JOIN User t ON c.target_ID = t.user_ID
WHERE c.status = 'Open';
```



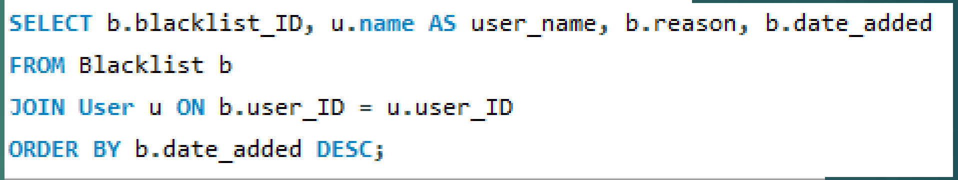
complaint_ID	status	date_filed	complainant	target_user
1	Open	2025-03-01 00:00:00	Alice	Emma
3	Open	2025-02-28 00:00:00	Charlie	Grace
4	Open	2025-03-05 00:00:00	David	Henry
6	Open	2025-03-07 00:00:00	Frank	Jack
9	Open	2025-02-27 00:00:00	Ivy	Mona
10	Open	2025-03-04 00:00:00	Jack	Nina
13	Open	2025-02-21 00:00:00	Mona	Quinn
15	Open	2025-03-03 00:00:00	Oscar	Steve
16	Open	2025-02-28 00:00:00	Paul	Tom
18	Open	2025-01-15 00:00:00	Rachel	Bob

Blacklist Entity

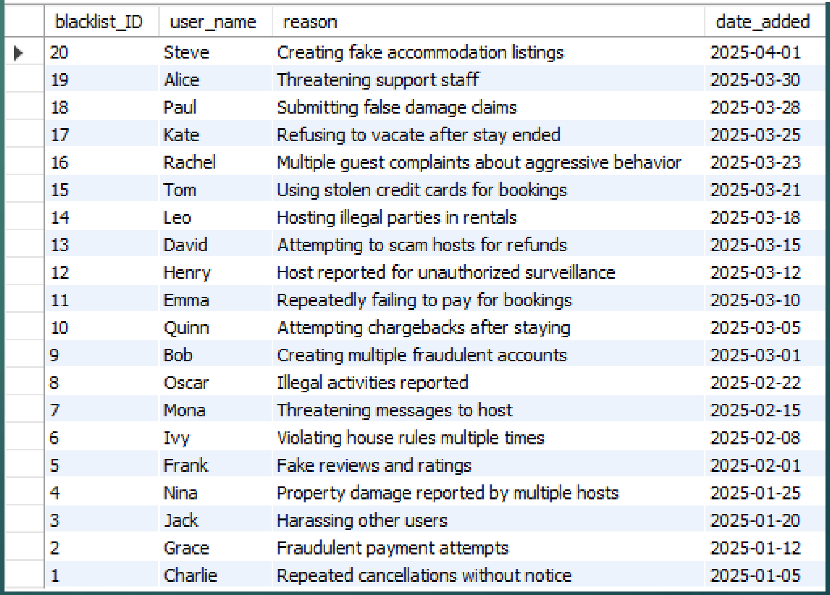
- The first picture shows the statement for creating the Blacklist table
- The attribute blacklist_ID is the primary key, while user_ID is the foreign key which references the User table
- The test case retrieves blacklist details from the Blacklist table, joining it with the User table to retrieve the user's name; The result is sorted by the date they were added, showing the most recent entries first



```
-- -----------------------------------------------------
-- Table `mydb`.`Blacklist`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Blacklist` (
  `blacklist_ID` INT NOT NULL AUTO_INCREMENT,
  `user_ID` INT NULL,
  `reason` TEXT(65535) NULL,
  `date_added` TIMESTAMP NULL,
  PRIMARY KEY (`blacklist_ID`),
  INDEX `user_ID_idx` (`user_ID` ASC) VISIBLE,
  CONSTRAINT `FK_user_blacklist`
    FOREIGN KEY (`user_ID`)
    REFERENCES `mydb`.`User` (`user_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```



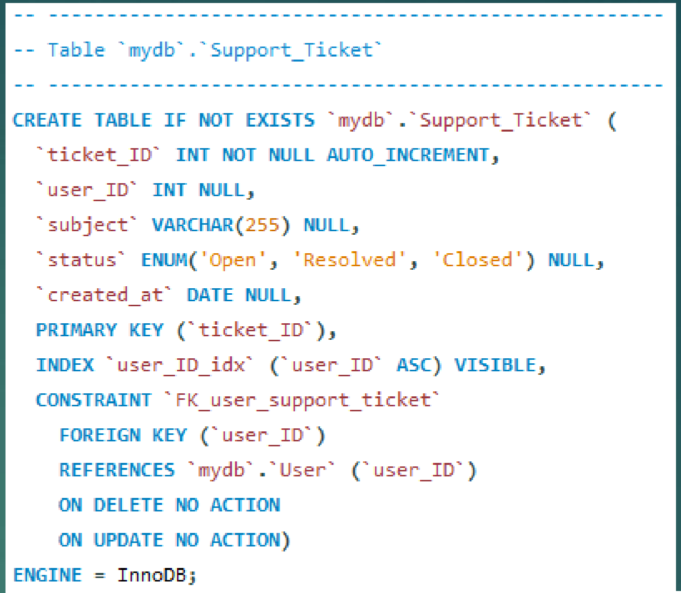
```
SELECT b.blacklist_ID, u.name AS user_name, b.reason, b.date_added
FROM Blacklist b
JOIN User u ON b.user_ID = u.user_ID
ORDER BY b.date_added DESC;
```



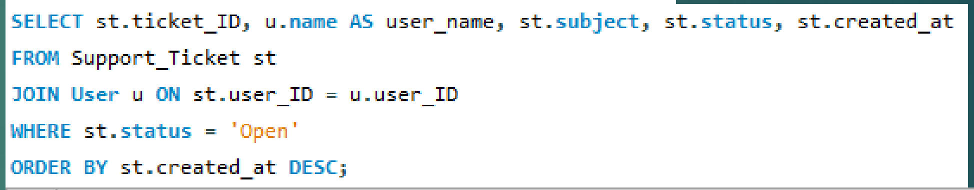
blacklist_ID	user_name	reason	date_added
20	Steve	Creating fake accommodation listings	2025-04-01
19	Alice	Threatening support staff	2025-03-30
18	Paul	Submitting false damage claims	2025-03-28
17	Kate	Refusing to vacate after stay ended	2025-03-25
16	Rachel	Multiple guest complaints about aggressive behavior	2025-03-23
15	Tom	Using stolen credit cards for bookings	2025-03-21
14	Leo	Hosting illegal parties in rentals	2025-03-18
13	David	Attempting to scam hosts for refunds	2025-03-15
12	Henry	Host reported for unauthorized surveillance	2025-03-12
11	Emma	Repeatedly failing to pay for bookings	2025-03-10
10	Quinn	Attempting chargebacks after staying	2025-03-05
9	Bob	Creating multiple fraudulent accounts	2025-03-01
8	Oscar	Illegal activities reported	2025-02-22
7	Mona	Threatening messages to host	2025-02-15
6	Ivy	Violating house rules multiple times	2025-02-08
5	Frank	Fake reviews and ratings	2025-02-01
4	Nina	Property damage reported by multiple hosts	2025-01-25
3	Jack	Harassing other users	2025-01-20
2	Grace	Fraudulent payment attempts	2025-01-12
1	Charlie	Repeated cancellations without notice	2025-01-05

Support Ticket Entity

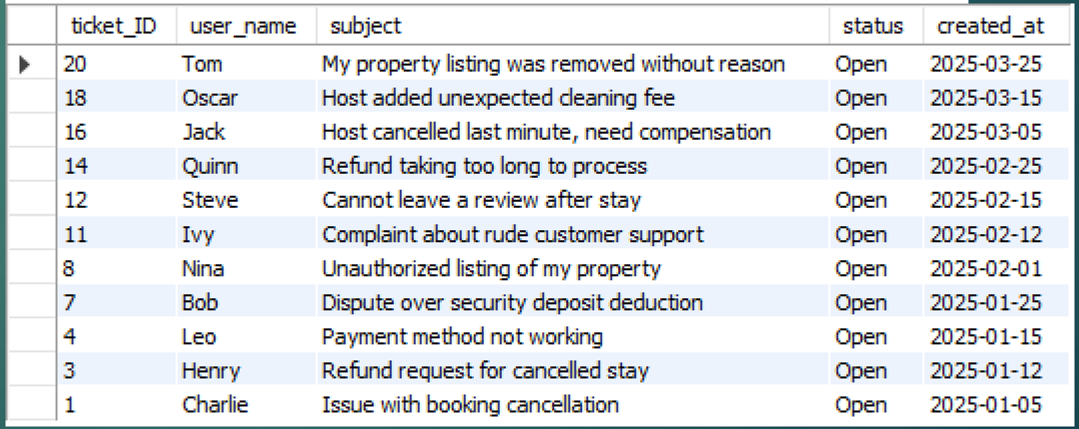
- The first picture shows the statement for creating the Support_Ticket table
- The attribute ticket_ID is the primary key, while user_ID is the foreign key referencing the User table
- The test case retrieves the details of all open support tickets, joining the Support_Ticket table with the User table to retrieve the user's name; The results are sorted in descending order based on the ticket creation date, showing the most recently created tickets first



```
-- -----------------------------------------------------
-- Table `mydb`.`Support_Ticket`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Support_Ticket` (
  `ticket_ID` INT NOT NULL AUTO_INCREMENT,
  `user_ID` INT NULL,
  `subject` VARCHAR(255) NULL,
  `status` ENUM('Open', 'Resolved', 'Closed') NULL,
  `created_at` DATE NULL,
  PRIMARY KEY (`ticket_ID`),
  INDEX `user_ID_idx` (`user_ID` ASC) VISIBLE,
  CONSTRAINT `FK_user_support_ticket`
    FOREIGN KEY (`user_ID`)
    REFERENCES `mydb`.`User` (`user_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```



```
SELECT st.ticket_ID, u.name AS user_name, st.subject, st.status, st.created_at
FROM Support_Ticket st
JOIN User u ON st.user_ID = u.user_ID
WHERE st.status = 'Open'
ORDER BY st.created_at DESC;
```



ticket_ID	user_name	subject	status	created_at
20	Tom	My property listing was removed without reason	Open	2025-03-25
18	Oscar	Host added unexpected cleaning fee	Open	2025-03-15
16	Jack	Host cancelled last minute, need compensation	Open	2025-03-05
14	Quinn	Refund taking too long to process	Open	2025-02-25
12	Steve	Cannot leave a review after stay	Open	2025-02-15
11	Ivy	Complaint about rude customer support	Open	2025-02-12
8	Nina	Unauthorized listing of my property	Open	2025-02-01
7	Bob	Dispute over security deposit deduction	Open	2025-01-25
4	Leo	Payment method not working	Open	2025-01-15
3	Henry	Refund request for cancelled stay	Open	2025-01-12
1	Charlie	Issue with booking cancellation	Open	2025-01-05