Graduate Diploma

Level 7

(Information Technology Strand)

STD513 Database

NQF Level 5, 10 credits

Project

(Worth 50% of final Mark)

(50 Marks)

Instructions and guidelines for the project:

1. Submission date and time: ____________________________________.
2. Completed project is to be submitted at the beginning of the class on the due date.
3. Submit a bonded copy of your report along with the electronic copy of all work. The lecturer will inform you how to submit the soft copy.
4. ***Warning: All media must be virus free!*** Media containing virus or media that cannot be run directly will result in a FAIL grade.
5. You must read and understand Aspire2's policy on 'Academic Dishonesty and Plagiarism'. Projects completed using unfair means or plagiarized will receive a FAIL grade.
6. The report must have a title page with your name, class and Id number clearly printed.
7. We advise that you start working on the project as soon as it has been handed out in class. Working on the project from day one will ensure that it is completed on time.
8. Work through each task, making copies of the source codes, diagrams and output produced as you completed them as they will be required as part of your submission.
9. Use the right naming and indentation style. Use comments to document each procedure, table and query.
10. Projects will be judged on the basis of completeness, relevancy and clearness.

Introduction to the project:

The aim of this project is to allow you to demonstrate an understanding of database design and analysis. The following are the objectives you have to meet based on the given tasks:

1. Analyse the description of an information system to create a data model representing the information system. (Learning Outcome 1)
2. Design and develop a working database that demonstrates the understanding of database development issues. (Learning Outcome 2)

Tasks details and Learning outcomes targeted

Task	**Topic**	**Learning outcomes targeted**	**Marks**
1	Project Scope	1	5
2	Project Design	1, 2	10

3	Project Implementation	2	20
4	Project Testing	2	10
5	Documentation and Final Submission	1	5
		Total	**50**

Project Description

Your project will be a small database system developed using SQL. You are required to use all the knowledge acquired throughout this course.

Task 1 – Project Scope (5 Marks)

It is important that you pick your project carefully, as you will be working on it throughout the term. It is much easier and fun to work on a project that is interesting and meaningful to you.

Each student has to select one of the projects provided at the end of this document. Students are also encouraged to work on other projects they would propose but they need first to write a short description like the one given at the end of document and seek lecturer approval.

Write an introduction (1-2 pages) that includes the project idea, its main functionality, the goal and objectives of the project.

Task 2 – Project Design (10 Marks)

Once your project proposal has been approved by your lecturer you can move on to design your project and develop the database system. In your design you should:

1) Describe the Data model using ER Diagram.
2) Apply your model at each stage of normalisation.
3) Develop a database schema for DBMS of your choice
4) List all the tables, relationships and constrains
5) List all the procedures and triggers
6) List at least three different reports

Task 3 – Project Implementation (20 Marks)

In this task you need to write the SQL code to implement your design. Make sure to meet the database design and implementation principles such as:

- Creating tables for the proposed schema
- Creating indices for each table
- Using different relationships for the related tables

- Declaration, assignments, control statement, and exception in SQL language.
- Using different single table and multi-table queries

Task 4 – Project Testing (10 Marks)

Perform database testing on the main functionality of your database and document the testing results. The database testing should include testing of data integrity, data accessing, query retrieving, modifications, updating and deletion.

Task 5: Documentation and Final Submission (5 Marks)

You are required to submit a report that includes the following items:

1. Introduction
2. ERD Diagram and other supported design documents
3. SQL code
4. Output Reports (queries)
5. Testing result

Sample Projects Ideas

Project 1

Design a database system for another university in Auckland with the following information and requirements:

- Professors have an SSN, a name, an age, a rank, and a research specialty.
- Projects have a project number, a sponsor name (e.g., NSF), a starting date, an ending date, and a budget.
- Graduate students have an SSN, a name, an age, and a degree program (e.g., M.S. or Ph.D.).
- Each project is managed by one professor (known as the project's principal investigator).
- Each project is worked on by one or more professors (known as the project's coinvestigators).
- Professors can manage and/or work on multiple projects.
- Each project is worked on by one or more graduate student (known as the project's research assistants).
- When graduate students work on a project, a professor must supervise their work on the project. Graduate students can work on multiple projects, in which case they will have a (potentially different) supervisor for each one.
- Departments have a department number, a department name, and a main office.
- Departments have a professor (known as the chairman) who runs the department.
- Professor works in one or more departments, and for each department that they work in, a time percentage is associated with their job.
- Graduate students have one major department in which they are working on their degree.
- Each graduate student has another, more senior, graduate student (known as a student advisor) who advises him or her on what courses to take.

Project 2:

Musica NZ has decided to store information about musicians who perform on its albums (as well as other company data) in a database. The company has wisely chosen to hire you as a database designer (at your usual consulting fee of $2,500/day).

- Each musician that records at Musica NZ has an SSN, a name, an address, and a phone number. Poorly paid musicians often share the same address, and no address has more than one phone.
- Each instrument that is used in songs recorded at Musica NZ has a name (e.g., guitar, synthesizer, flute) and a musical key (e.g., C, B-flat, E-flat).
- Each album that is recorded on the Musica NZ label has a title, a copyright date, a format (e.g., CD or MC), and an album identifier. Each song recorded at Musica NZ has a title and an author.
- Each musician may play several instruments, and a given instrument may be played by several musicians.
- Each album has a number of songs on it, but no song may appear on more than one album.
- Each song is performed by one or more musicians, and a musician may perform a number of songs.
- Each album has exactly one musician who acts as its producer. A musician may produce several albums, of course.

Project 3:

Oriental Health frequent fliers have been complaining to Auckland Airport officials about the poor organisation at the airport. As a result, the officials have decided that all information related to the airport should be organised using a DBMS, and you have been hired to design the database. Your first task is to organise the information about all the airplanes stationed and maintained at the airport. The relevant information is as follows:

- Every airplane has a registration number, and each airplane is of a specific model.
- The airport accommodates a number of airplane models, and each model is identified by a model number (e.g., DC-10) and has a capacity and a weight.
- A number of technicians work at the airport. You need to store the name, SSN, address, phone number, and salary of each technician.
- Each technician is an expert on one or more plane model(s), and his or her expertise may overlap with that of other technicians. This information about technicians must also be recorded.
- Traffic controllers must have an annual medical examination. For each traffic controller, you must store the date of the most recent exam.
- All airport employees (including technicians) belong to a union. You must store the union membership number of each employee. You can assume that each employee is uniquely identified by the social security number.
- The airport has a number of tests that are used periodically to ensure that airplanes are still airworthy. Each test has a Federal Aviation Administration (FAA) test number, a name, and a maximum possible score.
- The FAA requires the airport to keep track of each time that a given airplane is tested by a given technician using a given test. For each testing event, the information needed is the date, the number of hours the technician spent doing the test, and the score that the airplane received on the test.

Project 4:
A prominent healthcare chain of pharmacies has offered a lifetime supply of Medicare for anyone to design their database. You have agreed to this proposition. Here is the information about the requirements.

- Patients are identified by an SSN, and their names, addresses, and ages must be recorded.
- Doctors are identified by an SSN. For each doctor, the name, specialty, and years of experience must be recorded.
- Each pharmaceutical company is identified by name and has a phone number.
- For each drug, the trade name and formula must be recorded. Each drug is sold by a given pharmaceutical company, and the trade name identifies a drug uniquely from among the products of that company. If a pharmaceutical company is deleted, you need not keep track of its products any longer.
- Each pharmacy has a name, address, and phone number. Every patient has a primary physician.
- Every doctor has at least one patient. Each pharmacy sells several drugs and has a price for each. A drug could be sold at several pharmacies, and the price could vary from one pharmacy to another.
- Doctors prescribe drugs for patients. A doctor could prescribe one or more drugs for several patients, and a patient could obtain prescriptions from several doctors. Each prescription has a date and a quantity associated with it. You can assume that if a doctor prescribes the same drug for the same patient more than once, only the last such prescription needs to be stored.
- Pharmaceutical companies have long-term contracts with pharmacies. A pharmaceutical company can contract with several pharmacies, and a pharmacy can contract with several pharmaceutical companies. For each contract, you have to store a start date, an end date, and the text of the contract.
- Pharmacies appoint a supervisor for each contract. There must always be a super-visor for each contract, but the contract supervisor can change over the lifetime of the contract.

Project Marking Guide

Marking Criteria	Designed Marks	Awarded Mark	Comment(s):
Task 1: Scope			
Introduction (1-2 pages) that includes the project idea, its main functionality, the goal and objectives of project.	5		
Total of Task 1>>	**5**		
Task 2: Design			
ERD Diagram is created	2		
The model is applied for each stage of normalization	3		
List of tables, relationships and constrains	2		
List of procedures and triggers	2		
List at least three different reports	1		
Total of Task 2>>	**10**		
Task 3: Implementation			
Database schema principles used correctly, the right tables are created and used the right relationships, primary keys and foreign keys.	8		
Functionality All reports defined in the design are working correctly	4		
All triggers and procedures defined in the design are developed and working correctly	4		
System running and Output (At least three scenarios are used to test the program)	4		
Total of Task 3>>	**20**		
Task 4: Testing			
Database testing done correctly (data integrity, data accessing, query retrieving, updating and deletion)	10		
Total of Task 4>>	**10**		
Task 5: Final Submission			
Introduction	1		

ERD Diagram and other supported	1		
design documents SQL code Output Reports Testing result	 1 1 1		
Total of Task 5>>	**5**		
Total:	**50**		

Contents

Task 1 - Project Scope

1. Introduction

Tropical Roof is a company that hosts bands' performance, the company after a couple of years in the market has finally decided to upgrade adopting some technology to improve their schedule management. The I.T department after having some research into the market the company hired a developer to create the software that is going to be implemented and a company that is going to work together but also to design a database according to the Tropical Roof's business requirement.

The most important functionality of it, is to keep a well-structured database with integrity and to have disponible the schedule performance considering the band and members information, also the number of tickets sold on a specific day, the amount received in a day or a month is another important fact for future analysis and comparison. Furthermore, the performance schedule will be created for ad purposes and have a record of the shows.

Nevertheless, the focus of this investment is on the reason of avoiding time-wasting, duplicated data, information about the band members, date's performance, and timing. The house is also planning to control the number of people that purchased the tickets to watch the performances according to the number of available tickets, and the amount that will be received to pay the band members.

Considering the database structure, it will be created according to the following items:

- Each staff member has a SSN, a name, surname, an address, city, country, email, salary, department, and mobile phone.

- Department, each department has a code and department name.

- City table, each city has a code, city name and country.

- Every performance must be confirmed with an id, band's name, date, time, audience (number of people), and duration. More than one band can perform in one day.

- Each band can perform only single time in a day.

- More than one staff can work in the hosting event

- The band must be registered, with an id, band's name, genre, city, commission, and times in the house.

- Each band has a booking agent, having name, phone number and email.

- All the tickets will contain the performance's id, date, price, purchased date, and quantity available. More than one ticket can be purchased per person.

- Every band member has a name, instrument, city, country, and years of experience. More than one instrument can be registered.

- A mensal band ranking will be updated having the band's name, times in the house, audience, and position.

At long last, the company is investing in this project in reason to believe in the benefits that the technology may bring. It is clear that the Tropical Roof's data is going to be safer in a database which will be used for many years, some I.T staff members will be required to self-qualification to keep maintaining the software, database, and future upgrades if necessary.

Task 2 - Project Design

2.1 ER Diagram

ER Diagram stands for Entity relationship diagram, which is useful in software engineering describing the data or explain the logical structure of databases. The figure below describes the project scenario in the ER Diagram format having entities, attributes, and relationship between the entities.

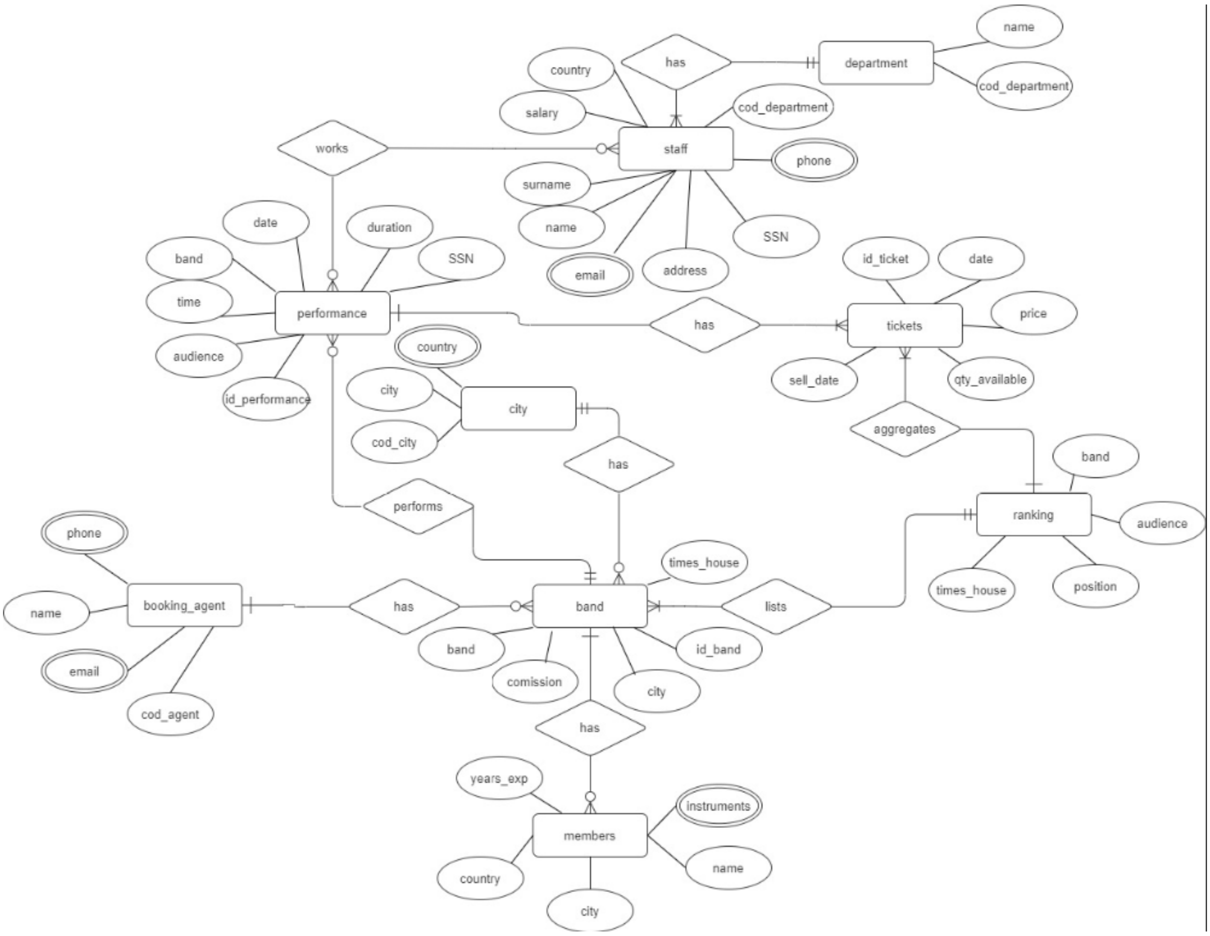


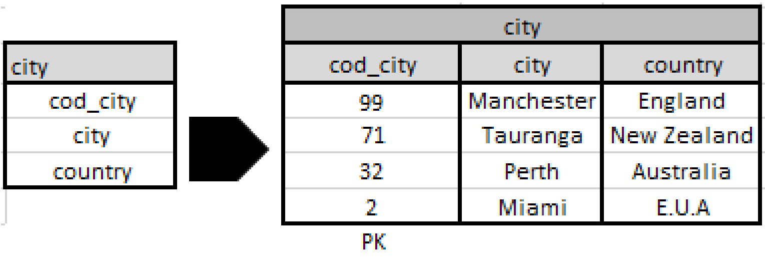
Figure 1: Entity-relationship model (ER Diagram)

2.2 Each stage of normalization

To start the normalization we have the tables non-normalized, and it is going to be changed into the 1NF, 2NF and finally the 3NF if necessary.

- The first normal form is used to identify the multivalued and composed attributes.
- The second normal form is used to identify the partial dependencies.
- The third normal form is used to identify the transitive dependencies.

Attributes Values



city
cod_city
city
country

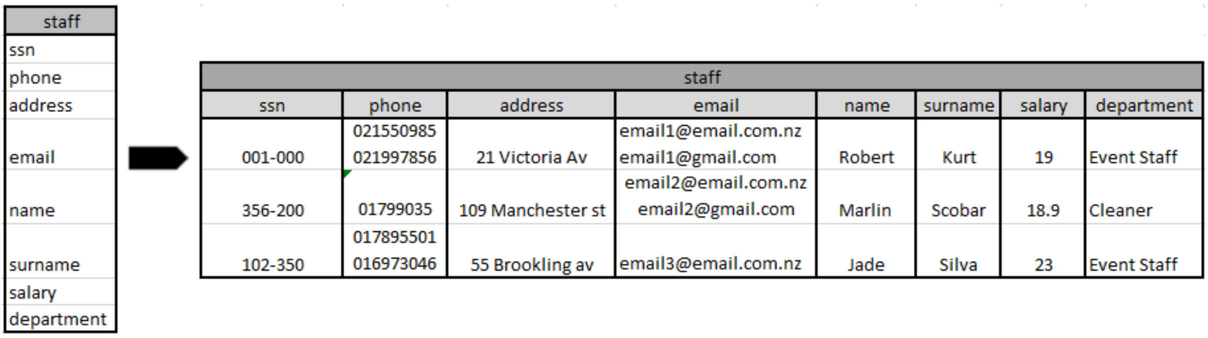
city		
cod_city	city	country
99	Manchester	England
71	Tauranga	New Zealand
32	Perth	Australia
2	Miami	E.U.A
PK		

Figure 2: Table not normalized

Description: the table city has the cod_city attribute as a primary key, being defined as a foreign key into the table band to define the bands' origin, also the others attributes define the city name and country from the band.

Table **staff**

Attributes Values

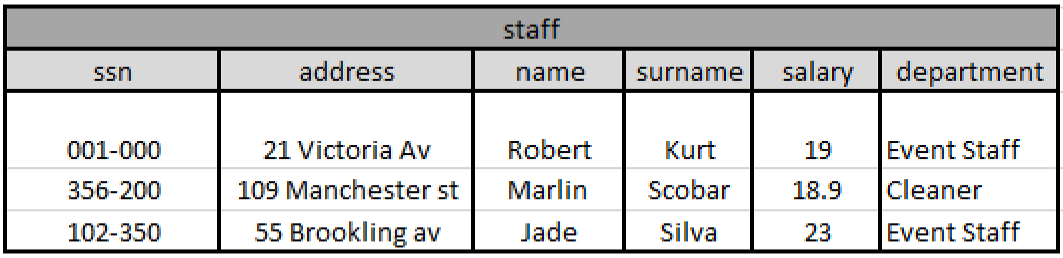


staff
ssn
phone
address
email
name
surname
salary
department

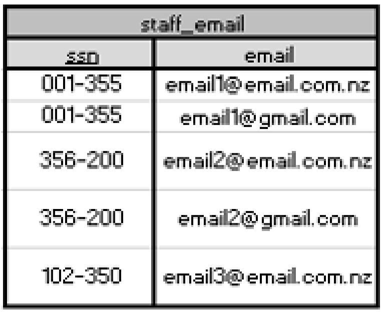
staff							
ssn	phone	address	email	name	surname	salary	department
001-000	021550985 021997856	21 Victoria Av	email1@email.com.nz email1@gmail.com	Robert	Kurt	19	Event Staff
356-200	01799035	109 Manchester st	email2@email.com.nz email2@gmail.com	Marlin	Scobar	18.9	Cleaner
102-350	017895501 016973046	55 Brookling av	email3@email.com.nz	Jade	Silva	23	Event Staff

Figure 3: Table not normalized

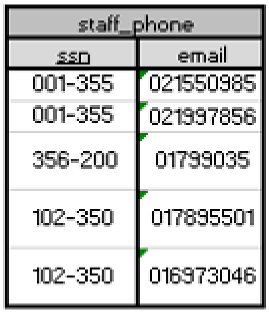
Table after being normalized (1NF)



staff					
ssn	address	name	surname	salary	department
001-000	21 Victoria Av	Robert	Kurt	19	Event Staff
356-200	109 Manchester st	Marlin	Scobar	18.9	Cleaner
102-350	55 Brookling av	Jade	Silva	23	Event Staff

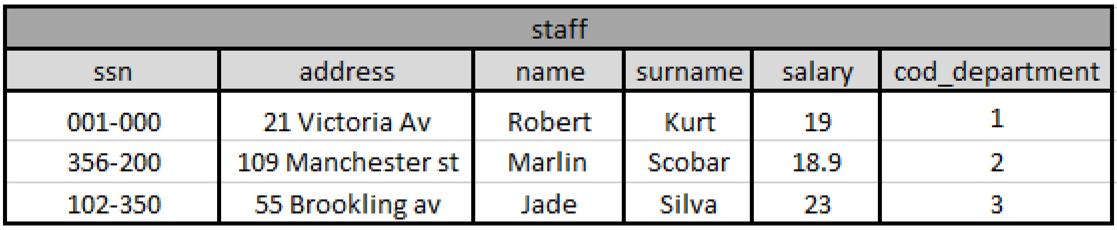


staff_email	
ssn	email
001-355	email1@email.com.nz
001-355	email1@gmail.com
356-200	email2@email.com.nz
356-200	email2@gmail.com
102-350	email3@email.com.nz

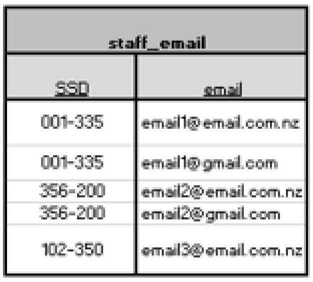


staff_phone	
ssn	email
001-355	021550985
001-355	021997856
356-200	01799035
102-350	017895501
102-350	016973046

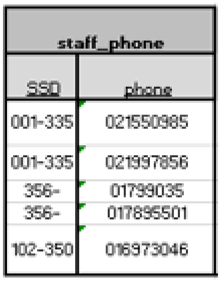
Table after being normalized (3NF)



staff					
ssn	address	name	surname	salary	cod_department
001-000	21 Victoria Av	Robert	Kurt	19	1
356-200	109 Manchester st	Marlin	Scobar	18.9	2
102-350	55 Brookling av	Jade	Silva	23	3



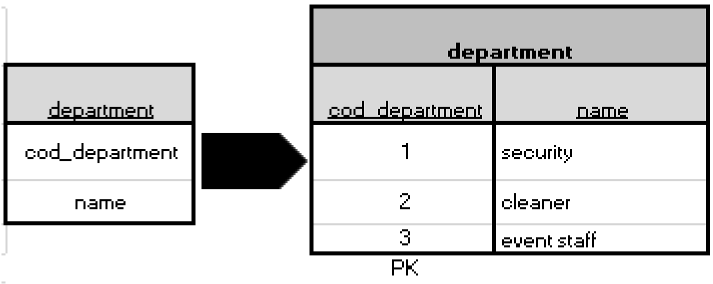
staff_email	
ssn	email
001-335	email1@email.com.nz
001-335	email1@gmail.com
356-200	email2@email.com.nz
356-200	email2@gmail.com
102-350	email3@email.com.nz



staff_phone	
ssn	phone
001-335	021550985
001-335	021997856
356-	01799035
356-	017895501
102-350	016973046

Description: In the table, the staff has the attribute ssn as the primary key which is going to be a foreign key into the tables staff_email to link all the emails that staff have, staff_phone to link all the phones that staff have and into the table performance, to link all the performances that the staff is going to work on. The attribute cod_department is a foreign key which links to the table department having a description of the department that the staff works.

Table **department**



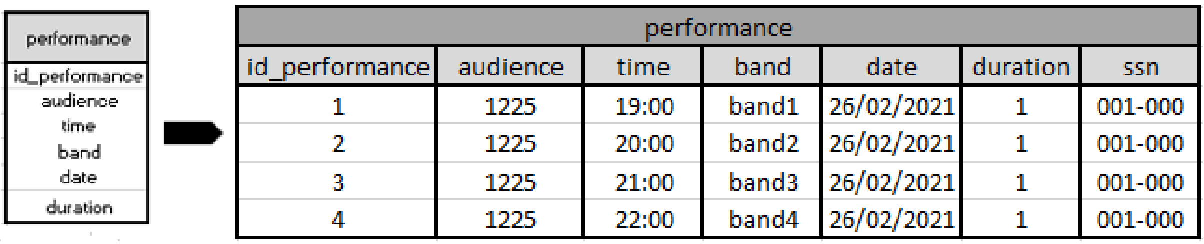
department
cod_department
name
PK

department	
cod_department	name
1	security
2	cleaner
3	event staff

Figure 4: Table not normalized

Description*: The table department has the attribute cod_department which is going to be a foreign key into the table staff to define a department code to staff, and the attribute name describes the department's name.*

Table **performance**

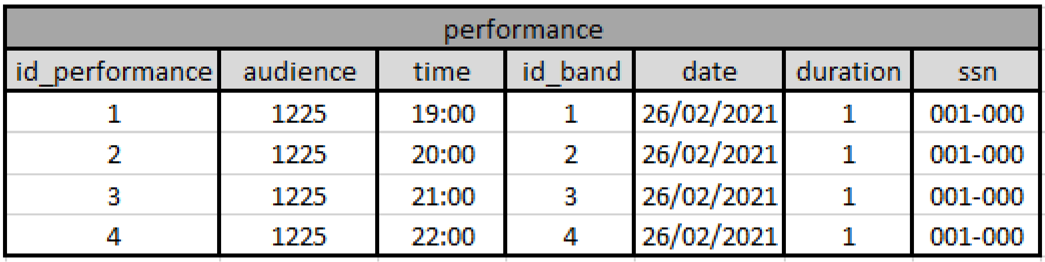


performance
id_performance
audience
time
band
date
duration

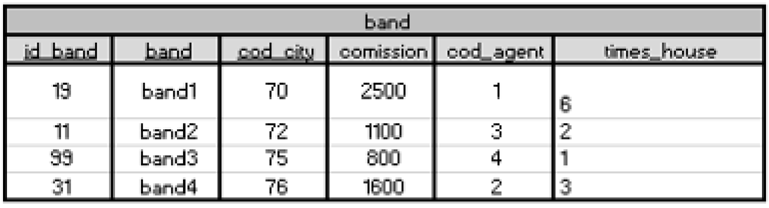
performance						
id_performance	audience	time	band	date	duration	ssn
1	1225	19:00	band1	26/02/2021	1	001-000
2	1225	20:00	band2	26/02/2021	1	001-000
3	1225	21:00	band3	26/02/2021	1	001-000
4	1225	22:00	band4	26/02/2021	1	001-000

Figure 5: Table not normalized

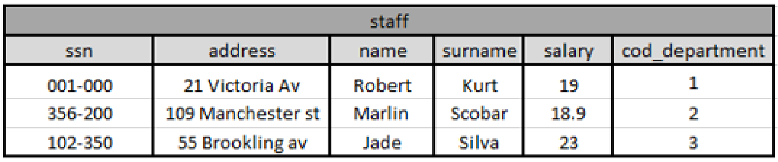
Table after being normalized (2NF)



performance						
id_performance	audience	time	id_band	date	duration	ssn
1	1225	19:00	1	26/02/2021	1	001-000
2	1225	20:00	2	26/02/2021	1	001-000
3	1225	21:00	3	26/02/2021	1	001-000
4	1225	22:00	4	26/02/2021	1	001-000



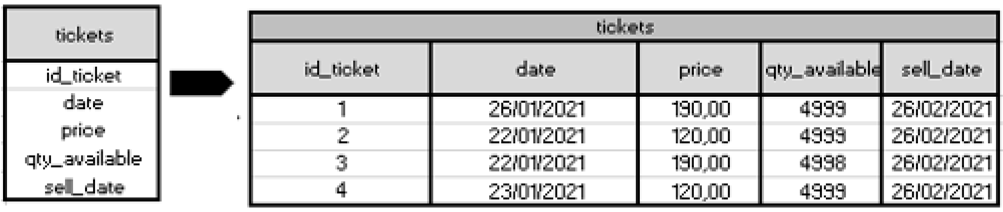
band					
id_band	band	cod_city	comission	cod_agent	times_house
19	band1	70	2500	1	6
11	band2	72	1100	3	2
99	band3	75	800	4	1
31	band4	76	1600	2	3



staff					
ssn	address	name	surname	salary	cod_department
001-000	21 Victoria Av	Robert	Kurt	19	1
356-200	109 Manchester st	Marlin	Scobar	18.9	2
102-350	55 Brookling av	Jade	Silva	23	3

Description: The table performance will list all the performances that the company is going to host, having the attribute id_performance as primary key, the attribute audience is going to be filled by a trigger according to the number of selling tickets, the attribute time informs the exact moment when the performance will start, the attribute id_band is a foreign key that links with the band responsible for that performance, having a date, duration of the presentation, and ssn as attributes; ssn is a foreign key from the table staff that informs who worked to that performance.

Table **tickets**



tickets
id_ticket
date
price
qty_available
sell_date

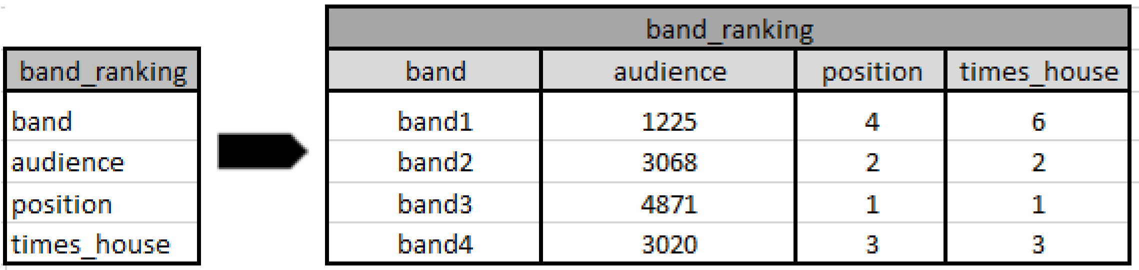
tickets				
id_ticket	date	price	qty_available	sell_date
1	26/01/2021	190,00	4999	26/02/2021
2	22/01/2021	120,00	4999	26/02/2021
3	22/01/2021	190,00	4998	26/02/2021
4	23/01/2021	120,00	4999	26/02/2021

Figure 6: Table not normalized

Description: The table tickets have the id_ticket as a primary key and contains the information of the selling, the attribute qty_available is going to be linked to a trigger with the table performance which will allow measuring the number of audiences.

Table **band_ranking**

Attributes Values

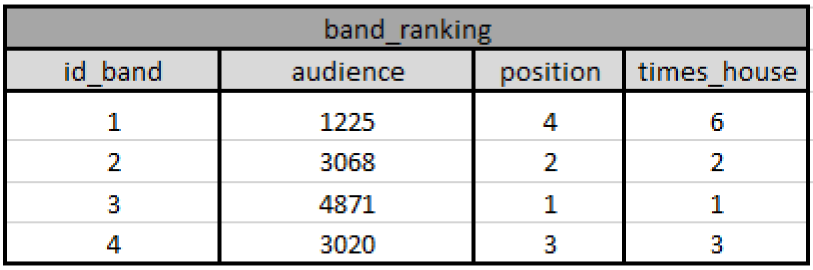


band_ranking
band
audience
position
times_house

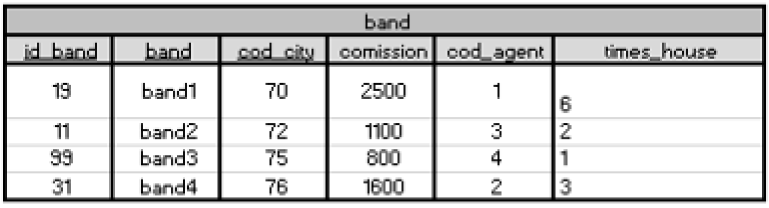
band_ranking			
band	audience	position	times_house
band1	1225	4	6
band2	3068	2	2
band3	4871	1	1
band4	3020	3	3

Figure 7: Table not normalized

Table after being normalized (2NF)



band_ranking			
id_band	audience	position	times_house
1	1225	4	6
2	3068	2	2
3	4871	1	1
4	3020	3	3



band					
id_band	band	cod_city	comission	cod_agent	times_house
19	band1	70	2500	1	6
11	band2	72	1100	3	2
99	band3	75	800	4	1
31	band4	76	1600	2	3

Description: The table band_ranking has the attribute id_band_ranking as primary and the foreign key that goes into the table band, it will contain the ranking information with the bands that have enrolled in the performance, being ordered according to the audience's number, the audience is also going to be filled with a trigger coming from the table performance. In addition, the times that the band have been in the house is going to be recorded.

Table **booking agent:**

Attributes

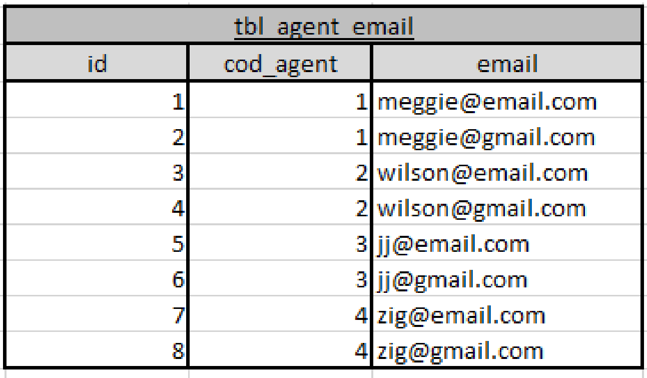
booking_agent
phone
name
email

Values

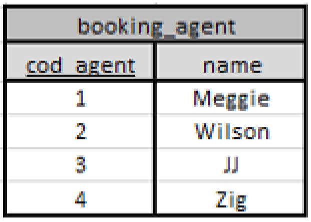
booking_agent		
phone	name	email
021024369 021559857	Meggie	meggie@email.com meggie@gmail.com
019054885 019855494	Wilson	wilson@email.com wilson@gmail.com
022895452 022556265	JJ	jj@email.com jj@gmail.com
20225487	Zig	zig@email.com zig@gmail.com

Figure 8: Table not normalized.

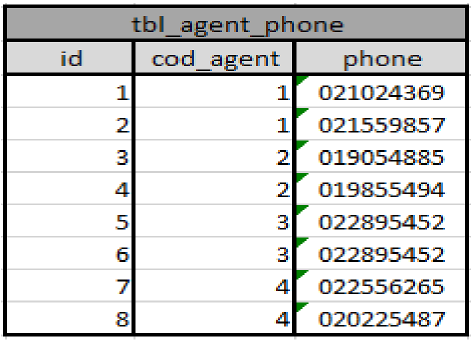
Table after being normalized (1NF)



tbl_agent_email		
id	cod_agent	email
1	1	meggie@email.com
2	1	meggie@gmail.com
3	2	wilson@email.com
4	2	wilson@gmail.com
5	3	jj@email.com
6	3	jj@gmail.com
7	4	zig@email.com
8	4	zig@gmail.com



booking_agent	
cod_agent	name
1	Meggie
2	Wilson
3	JJ
4	Zig

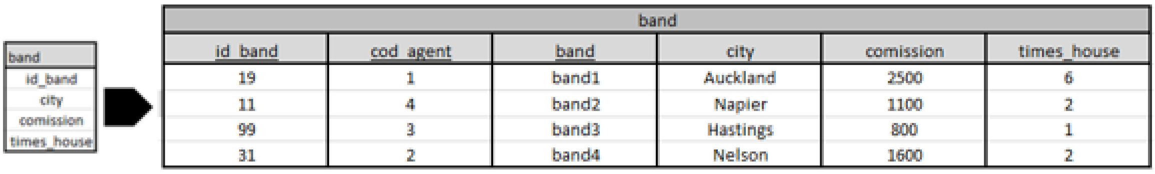


tbl_agent_phone		
id	cod_agent	phone
1	1	021024369
2	1	021559857
3	2	019054885
4	2	019855494
5	3	022895452
6	3	022895452
7	4	022556265
8	4	020225487

Description: The table booking_agent has the attribute cod_agent as a primary key, this table contain the booking agents that are responsible for each band, this attribute is also into the tables tbl_agent_email and tbl_agent_phone as foreign key which will have a link with the agent's email and phone information.

Table **band**

Attributes Values

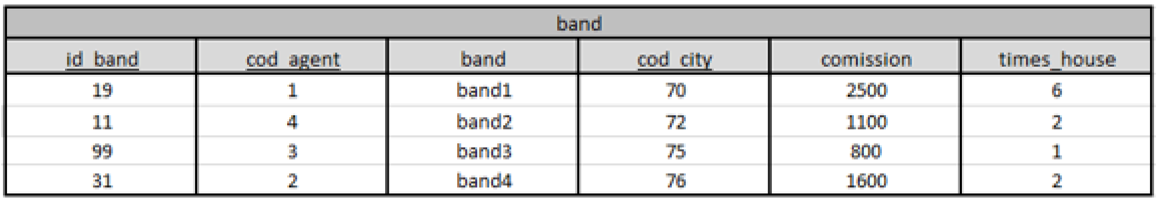


band
id_band
city
comission
times_house

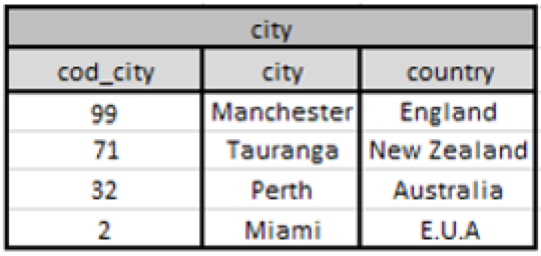
band					
id_band	cod_agent	band	city	comission	times_house
19	1	band1	Auckland	2500	6
11	4	band2	Napier	1100	2
99	3	band3	Hastings	800	1
31	2	band4	Nelson	1600	2

Figure 9: Table not normalized

Table after being normalized (2NF)



band					
id_band	cod_agent	band	cod_city	comission	times_house
19	1	band1	70	2500	6
11	4	band2	72	1100	2
99	3	band3	75	800	1
31	2	band4	76	1600	2

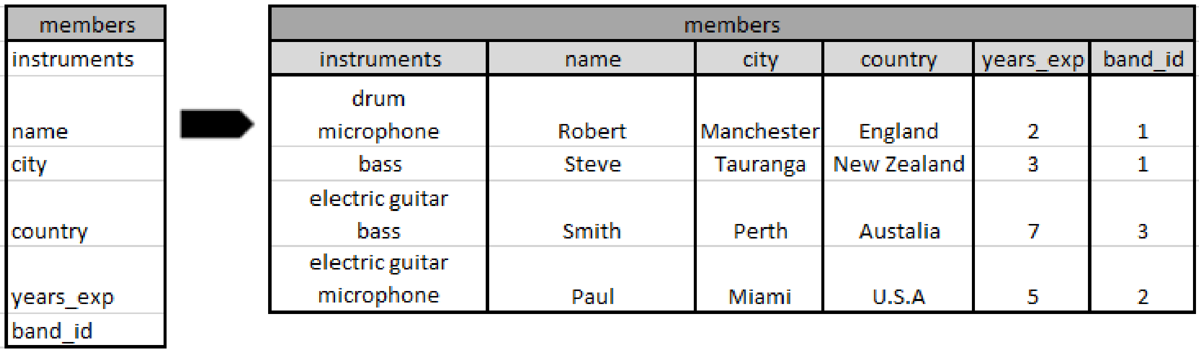


city		
cod_city	city	country
99	Manchester	England
71	Tauranga	New Zealand
32	Perth	Australia
2	Miami	E.U.A

Description: The table band has the attribute id_band as primary key, it also goes into the table performance and band_ranking

Table **members**

Attributes Values

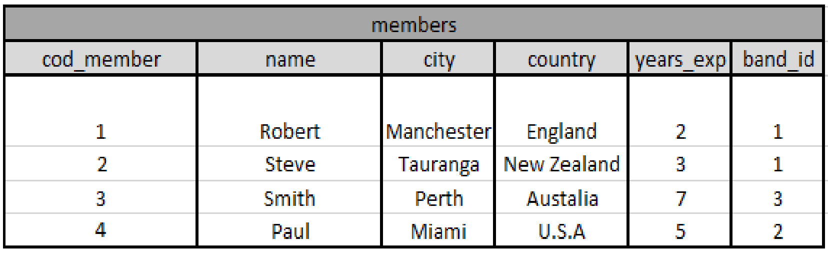


members
instruments
name
city
country
years_exp
band_id

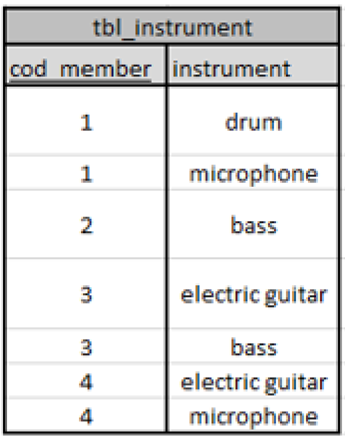
members					
instruments	name	city	country	years_exp	band_id
drum microphone	Robert	Manchester	England	2	1
bass	Steve	Tauranga	New Zealand	3	1
electric guitar bass	Smith	Perth	Austalia	7	3
electric guitar microphone	Paul	Miami	U.S.A	5	2

Figure 10: Table not normalized

Table after being normalized (1NF).

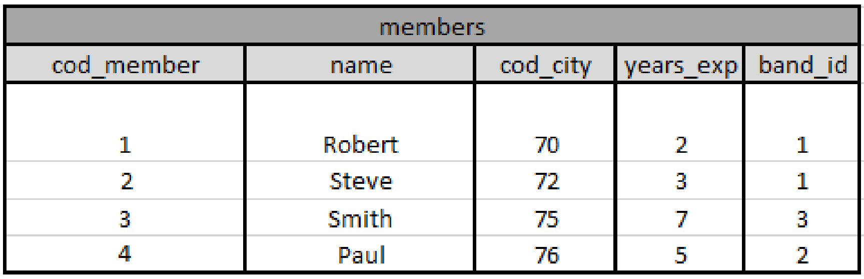


members					
cod_member	name	city	country	years_exp	band_id
1	Robert	Manchester	England	2	1
2	Steve	Tauranga	New Zealand	3	1
3	Smith	Perth	Austalia	7	3
4	Paul	Miami	U.S.A	5	2

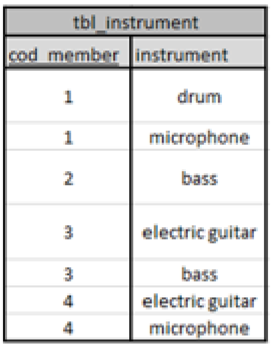


tbl_instrument	
cod_member	instrument
1	drum
1	microphone
2	bass
3	electric guitar
3	bass
4	electric guitar
4	microphone

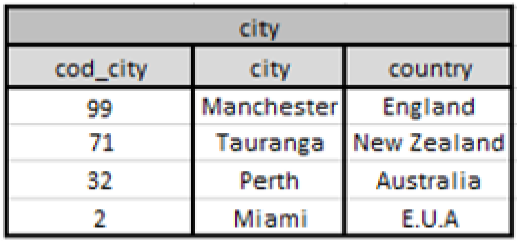
Table after being normalized (2NF)



members				
cod_member	name	cod_city	years_exp	band_id
1	Robert	70	2	1
2	Steve	72	3	1
3	Smith	75	7	3
4	Paul	76	5	2



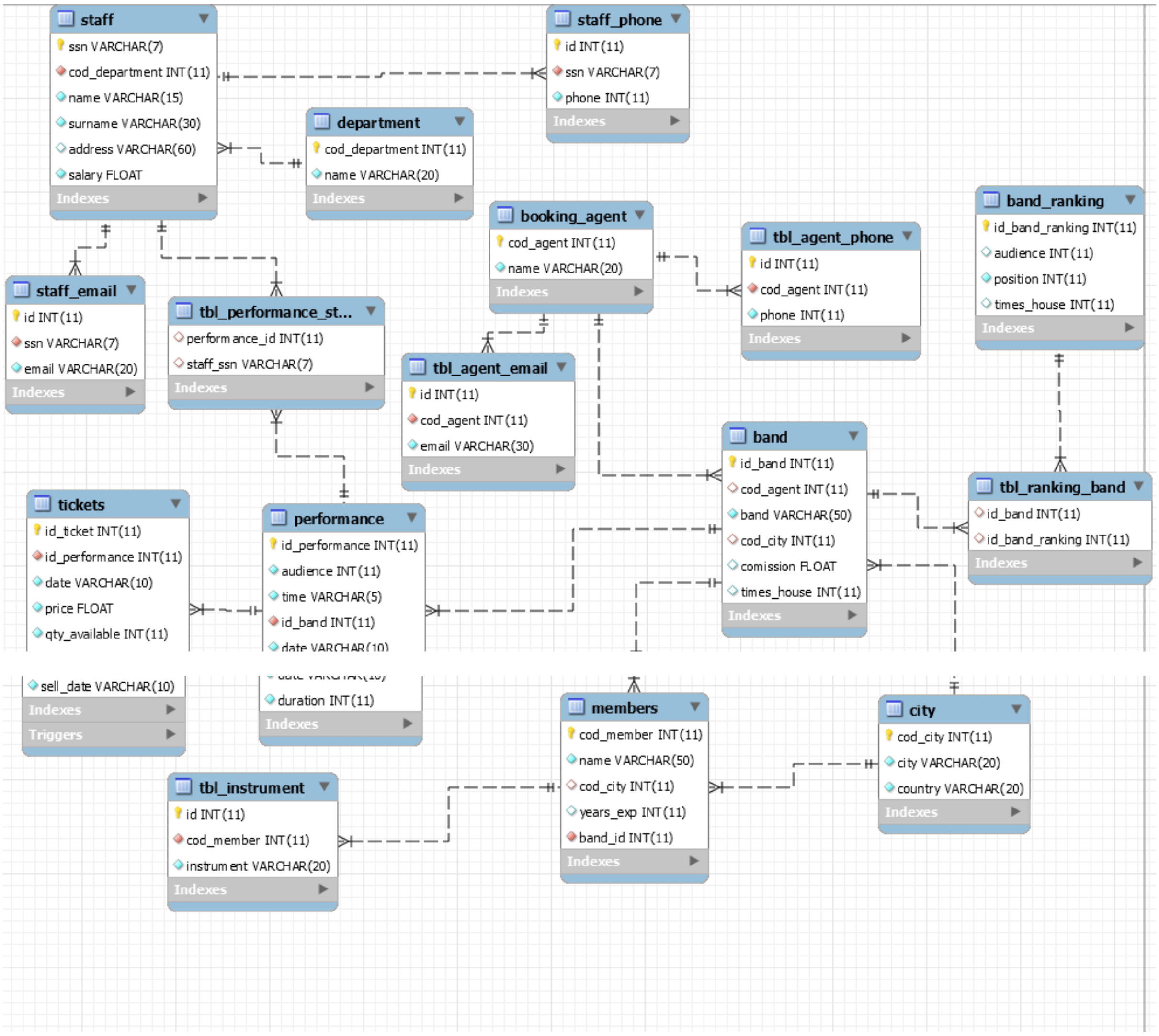
tbl_instrument	
cod_member	instrument
1	drum
1	microphone
2	bass
3	electric guitar
3	bass
4	electric guitar
4	microphone



city		
cod_city	city	country
99	Manchester	England
71	Tauranga	New Zealand
32	Perth	Australia
2	Miami	E.U.A

Description: The table members have the attribute cod_member as primary key, and it goes as a foreign key into the table tbl_instrument which has the role to define each instrument that a member plays, the attribute cod_city comes from the table city as a foreign key, linking to the city that the member comes from and the attribute id_band comes from the table band connecting the members to the band.

2.3 Schema


staff
ssn VARCHAR(7)
cod_department INT(11)
name VARCHAR(15)
surname VARCHAR(30)
address VARCHAR(60)
salary FLOAT
Indexes
staff_phone
id INT(11)
ssn VARCHAR(7)
phone INT(11)
Indexes
department
cod_department INT(11)
name VARCHAR(20)
Indexes
booking_agent
cod_agent INT(11)
name VARCHAR(20)
Indexes
tbl_agent_phone
id INT(11)
cod_agent INT(11)
phone INT(11)
Indexes
band_ranking
id_band_ranking INT(11)
audience INT(11)
position INT(11)
times_house INT(11)
Indexes
staff_email
id INT(11)
ssn VARCHAR(7)
email VARCHAR(20)
Indexes
tbl_performance_st...
performance_id INT(11)
staff_ssn VARCHAR(7)
Indexes
tbl_agent_email
id INT(11)
cod_agent INT(11)
email VARCHAR(30)
Indexes
band
id_band INT(11)
cod_agent INT(11)
band VARCHAR(50)
cod_city INT(11)
comission FLOAT
times_house INT(11)
Indexes
tbl_ranking_band
id_band INT(11)
id_band_ranking INT(11)
Indexes
tickets
id_ticket INT(11)
id_performance INT(11)
date VARCHAR(10)
price FLOAT
qty_available INT(11)
sell_date VARCHAR(10)
Indexes
Triggers
performance
id_performance INT(11)
audience INT(11)
time VARCHAR(5)
id_band INT(11)
date VARCHAR(10)
duration INT(11)
Indexes
members
cod_member INT(11)
name VARCHAR(50)
cod_city INT(11)
years_exp INT(11)
band_id INT(11)
Indexes
city
cod_city INT(11)
city VARCHAR(20)
country VARCHAR(20)
Indexes
tbl_instrument
id INT(11)
cod_member INT(11)
instrument VARCHAR(20)
Indexes

2.4 List all the tables, relationships, and constraints

This is the list of tables created in the database:

Table name	
1	tbl_instrument
2	members
3	city
4	staff
5	staff_email
6	department
7	staff_phone
8	band
9	performance
10	Band_ranking
11	booking_agent
12	tbl_agent_phone
13	tbl_agent_email
14	Tickets
15	relation_rank_band

Relationship

List of relationship between the tables created in the database:

Table name		Relationship	Table name
1	members	one to many	tbl_instrument
2	city	one to many	members
3*	staff	many to many	performance
4	staff_email	many to one	staff
5	staff	many to one	department
6	staff_phone	many to one	staff
7	band	many to one	city
8	performance	many to one	band
9*	band	one to many*	Band_ranking
10	booking_agent	one to many	band
11	Booking_agent	One to many	tbl_agent_phone
12	Booking_agent	One to many	tbl_agent_email
13	band	one to many	members

Constraints

This list is going to describe the constraints in list that contains for each table:

Table tbl_instrument:

tbl_instrument	
Attributes	**Constraints**
Id	Auto_increment and primary key
cod_member	not null, primary key and foreign key
instrument	not null

Table members:

members	
Attributes	**Constraints**
cod_member	not null, auto_increment and primary key
name	not null
cod_city	foreign key
years_exp	Department
band_id	Foreign key, not null

Table city:

city	
Attributes	**Constraints**
cod_city	auto_increment and primary key
city	not null
country	not null

Table staff

staff	
Attributes	**Constraints**
ssn	not null and primary key
cod_department	not null and foreign key
name	not null
surname	not null
address	NONE
salary	not null

Table staff_email:

staff_email	
Attributes	**Constraints**
Id	Auto_increment and foreign key
ssn	not null, primary key and foreign key
email	not null

Table department:

department	
Attributes	**Constraints**
cod_department	auto_increment and primary key
name	not null

Table staff_phone:

staff_phone	
Attributes	**Constraints**
Id	Auto_increment and foreign key
ssn	not null, primary key and foreign key
phone	not null

Table band:

band	
Attributes	**Constraints**
id_band	not null, auto_increment and primary key
cod_agent	foreign key
band	not null
cod_city	foreign key
comission	NONE
times_house	NONE

Table performance:

performance	
Attributes	**Constraints**
id_performance	auto_increment and primary key
audience	not null
times_house	not null
id_band	not null and foreign key
date	not null
Duration	not null

Table band_ranking:

Band_ranking	
Attributes	**Constraints**
id_band_ranking	Not null, auto_increment and foreign key
id_band	not null, primary key and foreign key
audience	NONE
position	not null and unique
times_house	not null

Table booking_agent:

booking_agent	
Attributes	**Constraints**
cod_agent	not null, auto_increment and primary key
name	not null

Table tbl_agent_phone:

tbl_agent_phone	
Attributes	**Constraints**
id	Auto_increment and primary key
cod_agent	not null, primary key and foreign key
phone	not null

Table tbl_agent_email:

tbl_agent_email	
Attributes	**Constraints**
Id	Auto_increment and foreign key
cod_agent	not null, primary key and foreign key
email	not null

Table tickets:

tickets	
Attributes	**Constraints**
id_ticket	auto_increment and primary key
date	not null
price	not null
qty_available	not null
sell_date	not null

2.5 List all the procedures and triggers

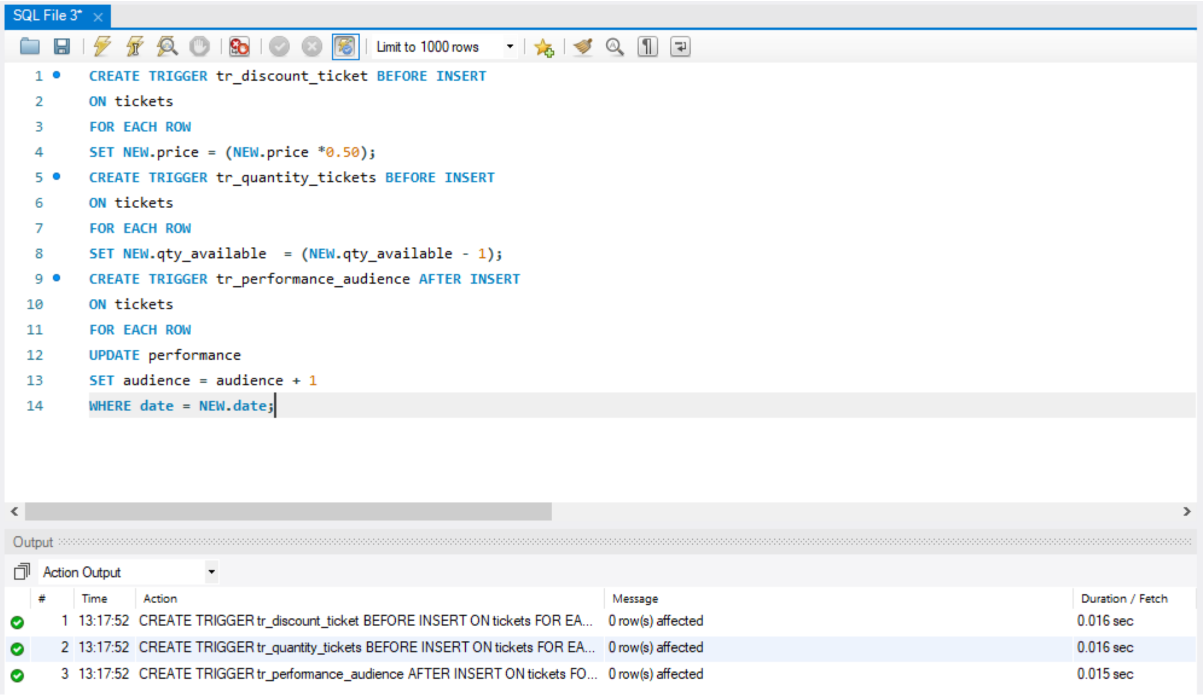
Procedures:

1. local_bands () - Display the number of local bands considering a specific city
2. amount_performance () - Lists the amount received by a specific performance's date having a SUM of all the sold tickets
3. bands_with_agent() - Lists how many bands have a booking agent searching by the agent id code
4. band_ranking () - Lists the ranking according to the number of audience from the concerts performed
5. worked_hours() - Sum the worked hours in the performances from a staff

Triggers:

1. tr_discount_ticket – Gives a discount on the ticket inclusion according to the percentage set
2. tr_quantity_tickets – Decrease a unit of tickets available after each ticket inclusion, which means that a ticket was sold having the control about how many the company have left
3. tr_performance_audience – After every ticket sell, a unit is increased considering as an audience that is going to be attended on the performance day.
4. tr_times_house – Every time that the band is going to perform in the house the database is going to count

2.6 List at least three different reports

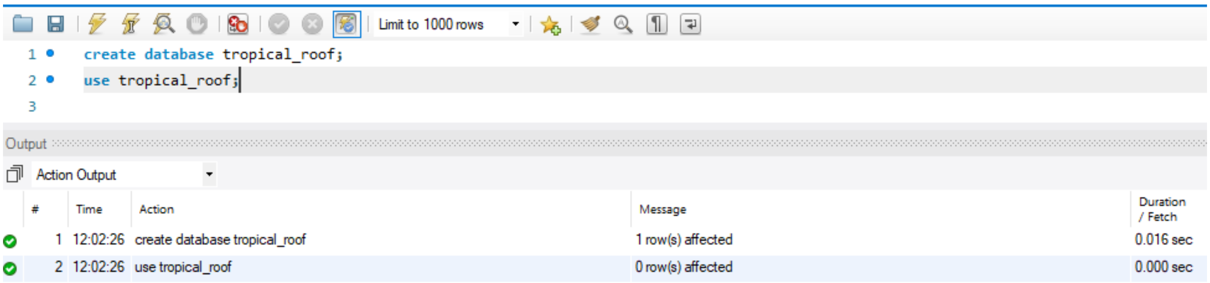


```sql
CREATE TRIGGER tr_discount_ticket BEFORE INSERT
ON tickets
FOR EACH ROW
SET NEW.price = (NEW.price *0.50);
CREATE TRIGGER tr_quantity_tickets BEFORE INSERT
ON tickets
FOR EACH ROW
SET NEW.qty_available  = (NEW.qty_available - 1);
CREATE TRIGGER tr_performance_audience AFTER INSERT
ON tickets
FOR EACH ROW
UPDATE performance
SET audience = audience + 1
WHERE date = NEW.date;
```

#	Time	Action	Message	Duration / Fetch
1	13:17:52	CREATE TRIGGER tr_discount_ticket BEFORE INSERT ON tickets FOR EA...	0 row(s) affected	0.016 sec
2	13:17:52	CREATE TRIGGER tr_quantity_tickets BEFORE INSERT ON tickets FOR EA...	0 row(s) affected	0.016 sec
3	13:17:52	CREATE TRIGGER tr_performance_audience AFTER INSERT ON tickets FO...	0 row(s) affected	0.015 sec

Task 3 – Project Implementation

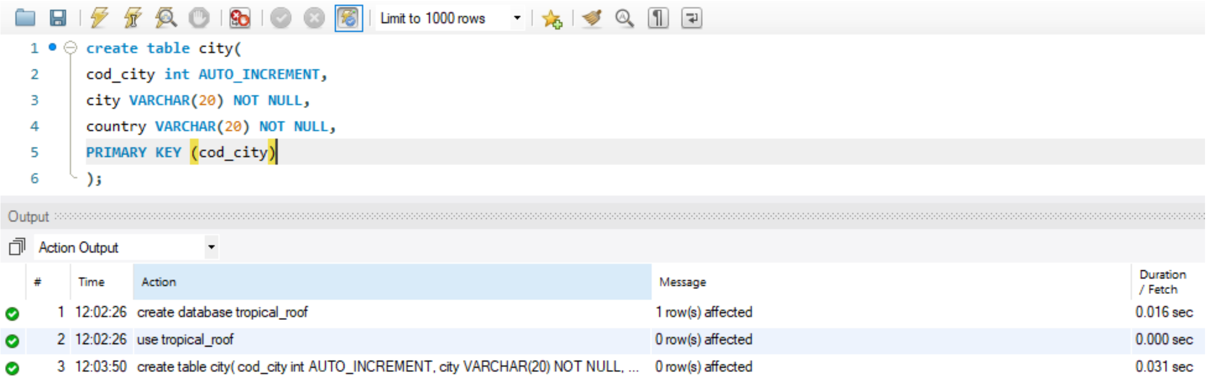
1. Creating the database which is called tropical_roof:



```
create database tropical_roof;
use tropical_roof;
```

```
Output
Action Output
 #  Time      Action                          Message           Duration / Fetch
 1  12:02:26  create database tropical_roof   1 row(s) affected 0.016 sec
 2  12:02:26  use tropical_roof               0 row(s) affected 0.000 sec
```

2. Creating the table city with the indexes



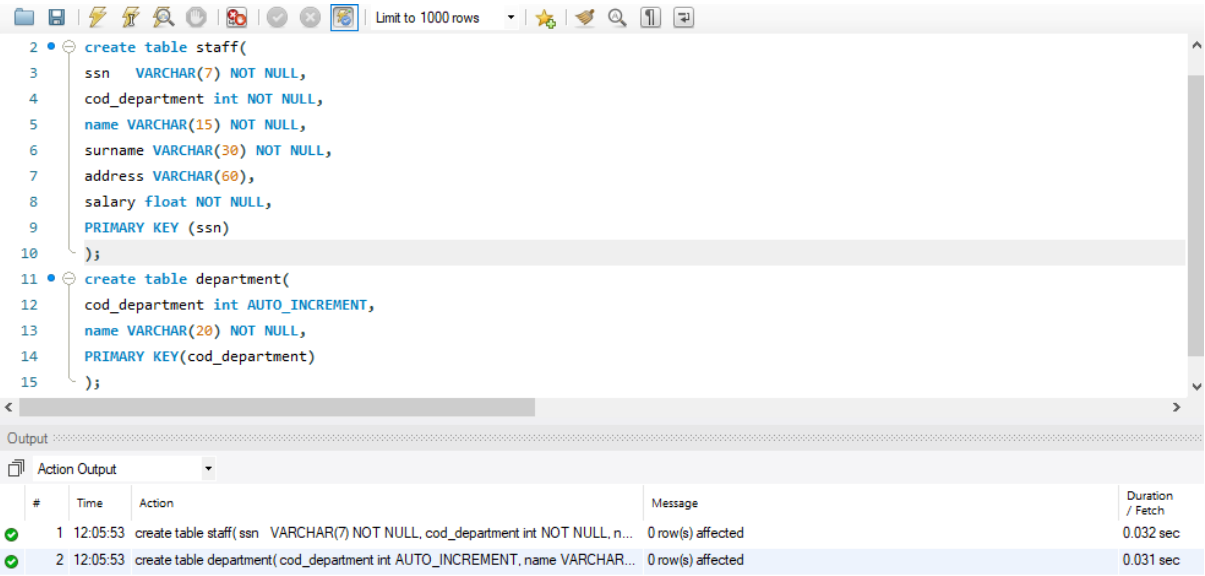
```
create table city(
cod_city int AUTO_INCREMENT,
city VARCHAR(20) NOT NULL,
country VARCHAR(20) NOT NULL,
PRIMARY KEY (cod_city)
);
```

```
Output
Action Output
 #  Time      Action                                                                       Message           Duration / Fetch
 1  12:02:26  create database tropical_roof                                                1 row(s) affected 0.016 sec
 2  12:02:26  use tropical_roof                                                            0 row(s) affected 0.000 sec
 3  12:03:50  create table city( cod_city int AUTO_INCREMENT, city VARCHAR(20) NOT NULL, ... 0 row(s) affected 0.031 sec
```

create database tropical_roof;
use tropical_roof;

create table city(
cod_city int AUTO_INCREMENT,
city VARCHAR(20) NOT NULL,
country VARCHAR(20) NOT NULL,
PRIMARY KEY (cod_city)
);

3. Creating tables staff and department with the indexes:



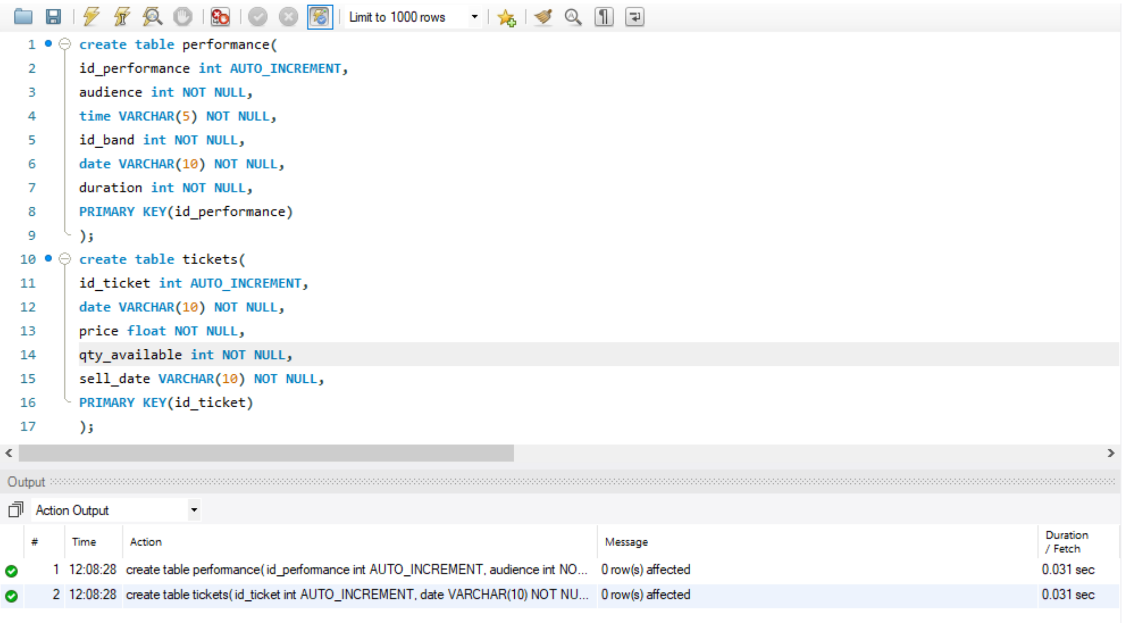
```
create table staff(
ssn   VARCHAR(7) NOT NULL,
cod_department int NOT NULL,
name VARCHAR(15) NOT NULL,
surname VARCHAR(30) NOT NULL,
address VARCHAR(60),
salary float NOT NULL,
PRIMARY KEY (ssn)
);
create table department(
cod_department int AUTO_INCREMENT,
name VARCHAR(20) NOT NULL,
PRIMARY KEY(cod_department)
);
```

	#	Time	Action	Message	Duration / Fetch
✓	1	12:05:53	create table staff(ssn VARCHAR(7) NOT NULL, cod_department int NOT NULL, n...	0 row(s) affected	0.032 sec
✓	2	12:05:53	create table department(cod_department int AUTO_INCREMENT, name VARCHAR...	0 row(s) affected	0.031 sec

```
create table staff(
ssn   VARCHAR(7) NOT NULL,
cod_department int NOT NULL,
name VARCHAR(15) NOT NULL,
surname VARCHAR(30) NOT NULL,
address VARCHAR(60),
salary float NOT NULL,
PRIMARY KEY (ssn)
);

create table department(
cod_department int AUTO_INCREMENT,
name VARCHAR(20) NOT NULL,
PRIMARY KEY(cod_department)
);
```

4. Creating tables performance and tickets with the index:



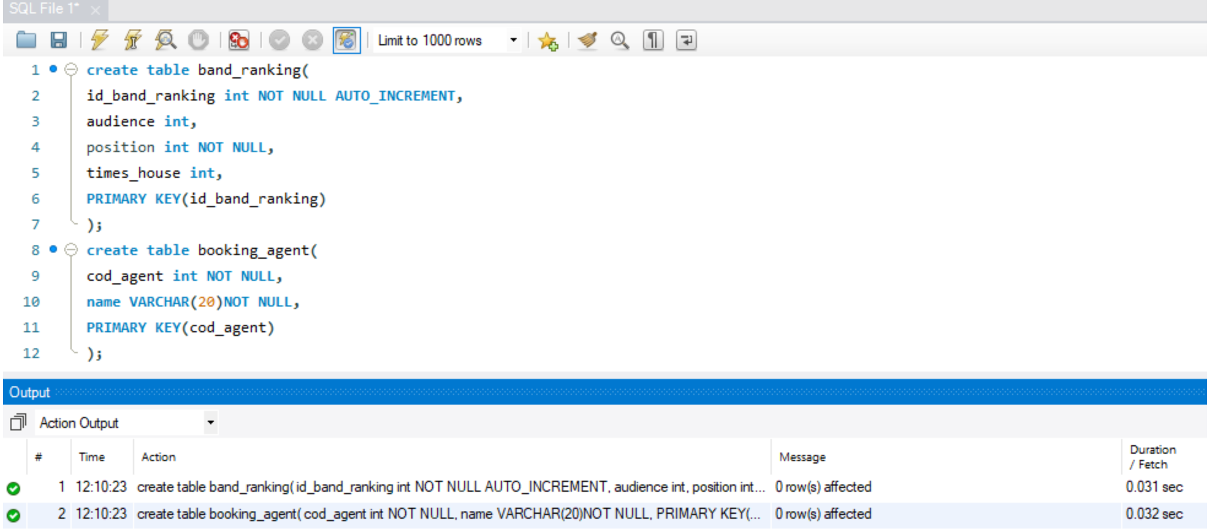
```
create table performance(
id_performance int AUTO_INCREMENT,
audience int NOT NULL,
time VARCHAR(5) NOT NULL,
id_band int NOT NULL,
date VARCHAR(10) NOT NULL,
duration int NOT NULL,
PRIMARY KEY(id_performance)
);
create table tickets(
id_ticket int AUTO_INCREMENT,
date VARCHAR(10) NOT NULL,
price float NOT NULL,
qty_available int NOT NULL,
sell_date VARCHAR(10) NOT NULL,
PRIMARY KEY(id_ticket)
);
```

#	Time	Action	Message	Duration / Fetch
1	12:08:28	create table performance(id_performance int AUTO_INCREMENT, audience int NO...	0 row(s) affected	0.031 sec
2	12:08:28	create table tickets(id_ticket int AUTO_INCREMENT, date VARCHAR(10) NOT NU...	0 row(s) affected	0.031 sec

```
create table performance(
id_performance int AUTO_INCREMENT,
audience int NOT NULL,
time VARCHAR(5) NOT NULL,
id_band int NOT NULL,
date VARCHAR(10) NOT NULL,
duration int NOT NULL,
PRIMARY KEY(id_performance)
);


create table tickets(
id_ticket int AUTO_INCREMENT,
id_performance int,
date VARCHAR(10) NOT NULL,
price float NOT NULL,
qty_available int NOT NULL,
sell_date VARCHAR(10) NOT NULL,
PRIMARY KEY(id_ticket)
);
```

5. Creating tables band_ranking and booking_agent with the indexes:



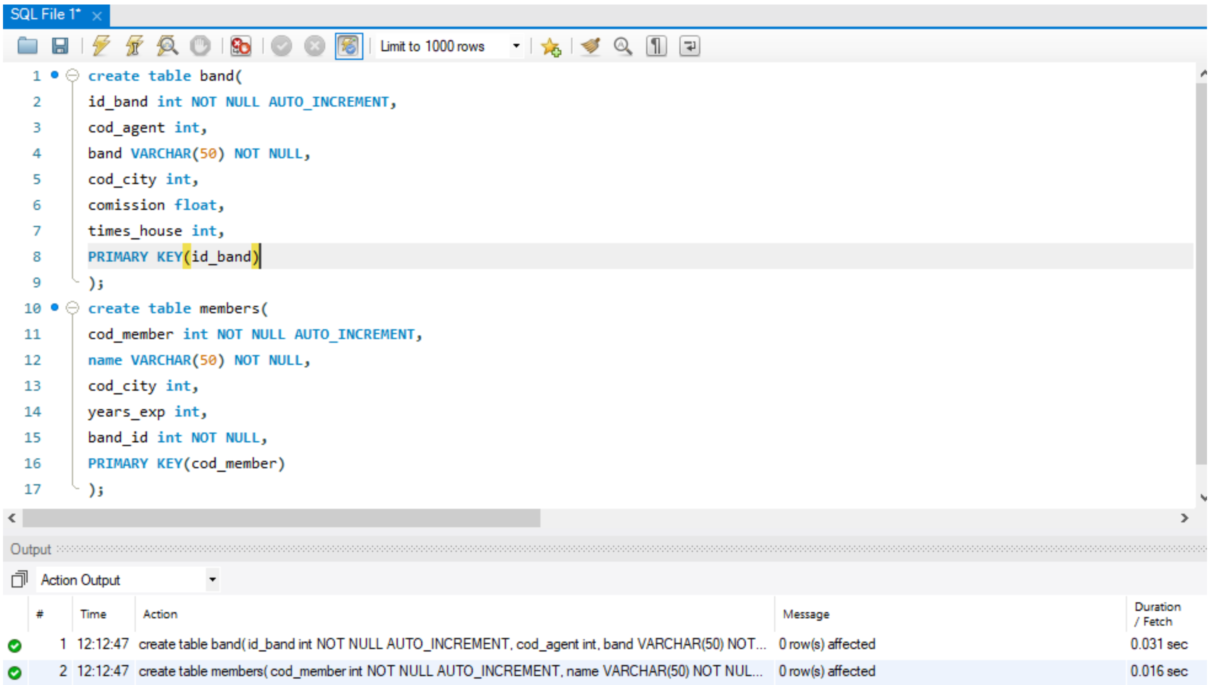
```
create table band_ranking(
id_band_ranking int NOT NULL AUTO_INCREMENT,
audience int,
position int NOT NULL,
times_house int,
PRIMARY KEY(id_band_ranking)
);
create table booking_agent(
cod_agent int NOT NULL,
name VARCHAR(20)NOT NULL,
PRIMARY KEY(cod_agent)
);
```

```
#  Time      Action                                                                                       Message            Duration / Fetch
1  12:10:23  create table band_ranking( id_band_ranking int NOT NULL AUTO_INCREMENT, audience int, position int...  0 row(s) affected  0.031 sec
2  12:10:23  create table booking_agent( cod_agent int NOT NULL, name VARCHAR(20)NOT NULL, PRIMARY KEY(...     0 row(s) affected  0.032 sec
```

```
create table band_ranking(
id_band_ranking int NOT NULL AUTO_INCREMENT,
audience int,
position int NOT NULL,
times_house int,
PRIMARY KEY(id_band_ranking)
);

create table booking_agent(
cod_agent int NOT NULL,
name VARCHAR(20)NOT NULL,
PRIMARY KEY(cod_agent)
);
```

6. Creating tables band and members with the indexes:



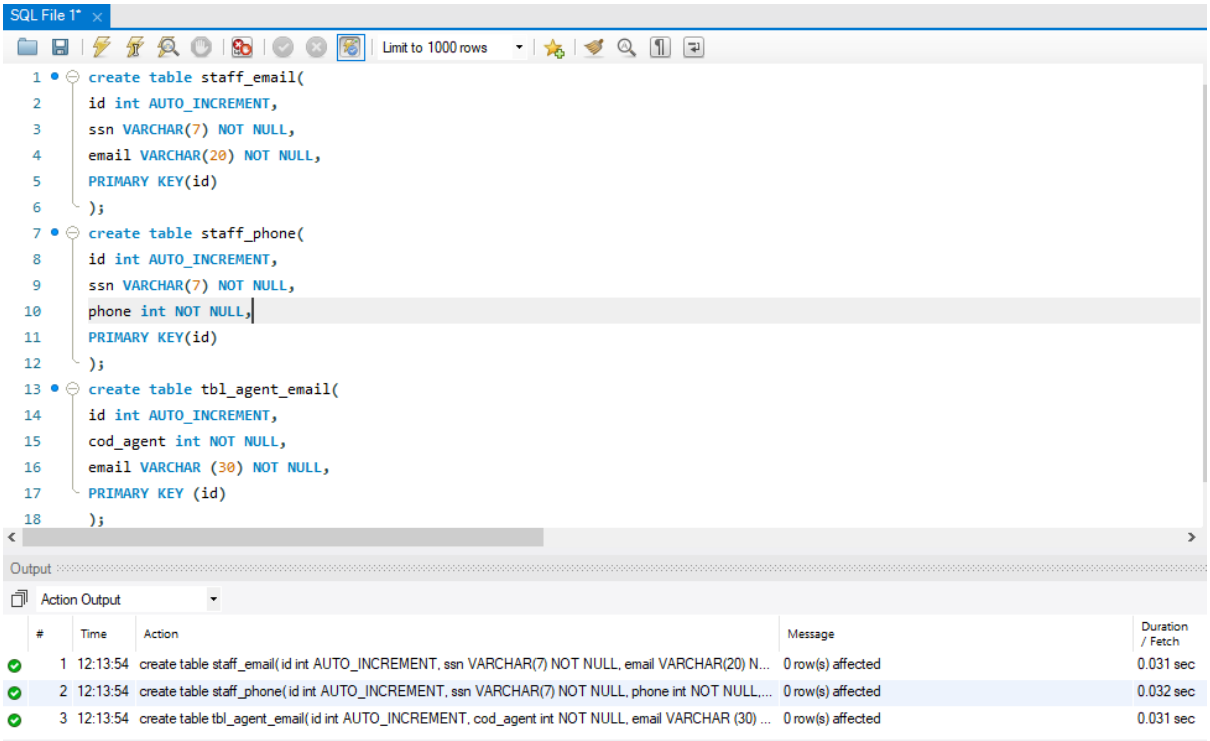
```
create table band(
id_band int NOT NULL AUTO_INCREMENT,
cod_agent int,
band VARCHAR(50) NOT NULL,
cod_city int,
comission float,
times_house int,
PRIMARY KEY(id_band)
);
create table members(
cod_member int NOT NULL AUTO_INCREMENT,
name VARCHAR(50) NOT NULL,
cod_city int,
years_exp int,
band_id int NOT NULL,
PRIMARY KEY(cod_member)
);
```

	#	Time	Action	Message	Duration / Fetch
✓	1	12:12:47	create table band(id_band int NOT NULL AUTO_INCREMENT, cod_agent int, band VARCHAR(50) NOT...	0 row(s) affected	0.031 sec
✓	2	12:12:47	create table members(cod_member int NOT NULL AUTO_INCREMENT, name VARCHAR(50) NOT NUL...	0 row(s) affected	0.016 sec

```
create table band(
id_band int NOT NULL AUTO_INCREMENT,
cod_agent int,
band VARCHAR(50) NOT NULL,
cod_city int,
comission float,
times_house int,
PRIMARY KEY(id_band)
);


create table members(
cod_member int NOT NULL AUTO_INCREMENT,
name VARCHAR(50) NOT NULL,
cod_city int,
years_exp int,
band_id int NOT NULL,
PRIMARY KEY(cod_member)
);
```

7. Creating tables staff_email, staff_phone and tbl_agent_email with indexes:



```
create table staff_email(
id int AUTO_INCREMENT,
ssn VARCHAR(7) NOT NULL,
email VARCHAR(20) NOT NULL,
PRIMARY KEY(id)
);
create table staff_phone(
id int AUTO_INCREMENT,
ssn VARCHAR(7) NOT NULL,
phone int NOT NULL,
PRIMARY KEY(id)
);
create table tbl_agent_email(
id int AUTO_INCREMENT,
cod_agent int NOT NULL,
email VARCHAR (30) NOT NULL,
PRIMARY KEY (id)
);
```

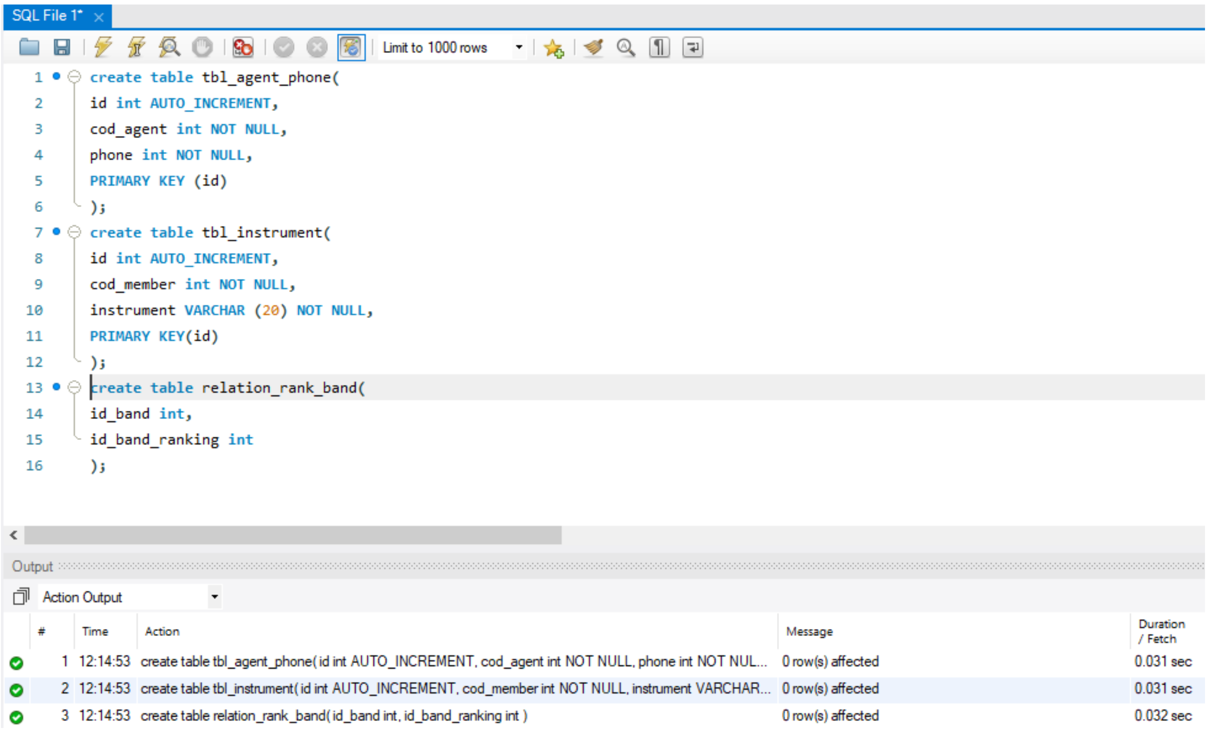
```
   #  Time      Action                                                                                    Message            Duration / Fetch
✓  1  12:13:54  create table staff_email( id int AUTO_INCREMENT, ssn VARCHAR(7) NOT NULL, email VARCHAR(20) N...  0 row(s) affected  0.031 sec
✓  2  12:13:54  create table staff_phone( id int AUTO_INCREMENT, ssn VARCHAR(7) NOT NULL, phone int NOT NULL,...  0 row(s) affected  0.032 sec
✓  3  12:13:54  create table tbl_agent_email( id int AUTO_INCREMENT, cod_agent int NOT NULL, email VARCHAR (30) ...  0 row(s) affected  0.031 sec
```

create table staff_email(
id int AUTO_INCREMENT,
ssn VARCHAR(7) NOT NULL,
email VARCHAR(20) NOT NULL,
PRIMARY KEY(id)
);

create table staff_phone(
id int AUTO_INCREMENT,
ssn VARCHAR(7) NOT NULL,
phone int NOT NULL,
PRIMARY KEY(id)
);

create table tbl_agent_email(
id int AUTO_INCREMENT,
cod_agent int NOT NULL,
email VARCHAR (30) NOT NULL,
PRIMARY KEY (id)
);

8. Creating tables tbl_agent_phone, tbl_instrument and tbl_ranking_band with indexes:



```
create table tbl_agent_phone(
id int AUTO_INCREMENT,
cod_agent int NOT NULL,
phone int NOT NULL,
PRIMARY KEY (id)
);
create table tbl_instrument(
id int AUTO_INCREMENT,
cod_member int NOT NULL,
instrument VARCHAR (20) NOT NULL,
PRIMARY KEY(id)
);
create table relation_rank_band(
id_band int,
id_band_ranking int
);
```

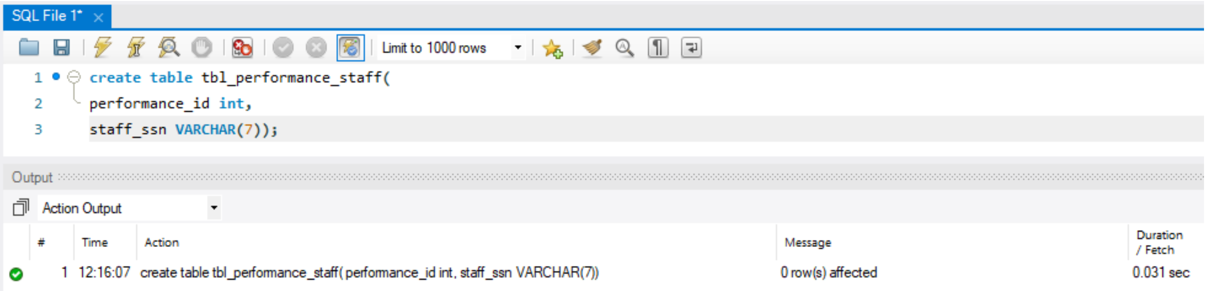
```
#  Time      Action                                                                                   Message              Duration / Fetch
1  12:14:53  create table tbl_agent_phone( id int AUTO_INCREMENT, cod_agent int NOT NULL, phone int NOT NUL...   0 row(s) affected    0.031 sec
2  12:14:53  create table tbl_instrument( id int AUTO_INCREMENT, cod_member int NOT NULL, instrument VARCHAR...  0 row(s) affected    0.031 sec
3  12:14:53  create table relation_rank_band( id_band int, id_band_ranking int )                                0 row(s) affected    0.032 sec
```

```
create table tbl_agent_phone(
id int AUTO_INCREMENT,
cod_agent int NOT NULL,
phone int NOT NULL,
PRIMARY KEY (id)
);

create table tbl_instrument(
id int AUTO_INCREMENT,
cod_member int NOT NULL,
instrument VARCHAR (20) NOT NULL,
PRIMARY KEY(id)
);

create table tbl_ranking_band(
id_band int,
id_band_ranking int
);
```

Creating the table tbl_performance_staff:

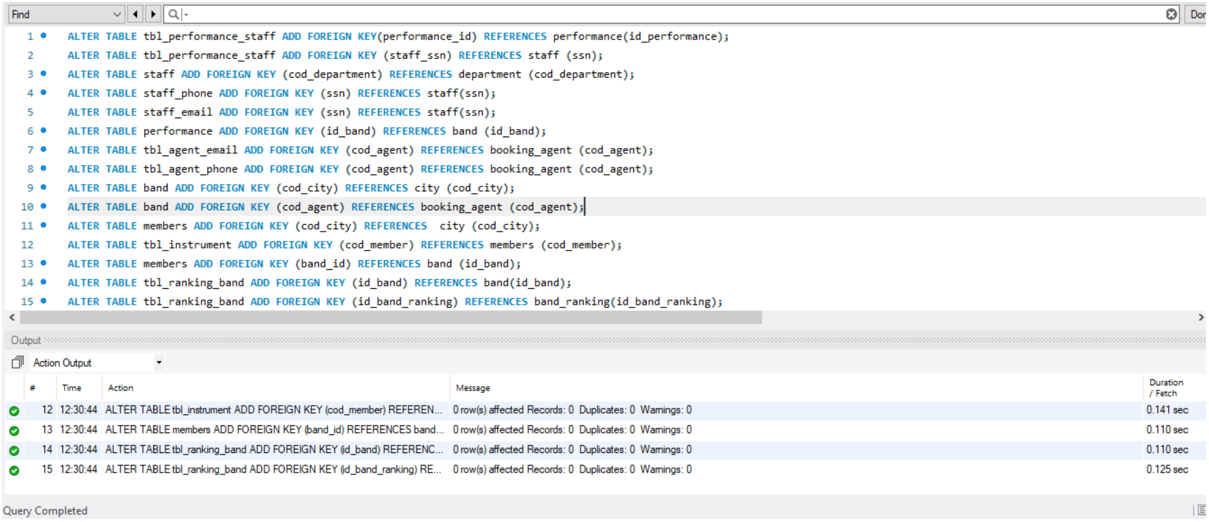


```
create table tbl_performance_staff(
performance_id int,
staff_ssn VARCHAR(7));
```

```
Output
Action Output
#   Time      Action                                                                    Message           Duration / Fetch
1   12:16:07  create table tbl_performance_staff( performance_id int, staff_ssn VARCHAR(7))  0 row(s) affected  0.031 sec
```

create table tbl_performance_staff(
performance_id int,
staff_ssn VARCHAR(7));

Including the foreign keys:



```
ALTER TABLE tbl_performance_staff ADD FOREIGN KEY(performance_id) REFERENCES performance(id_performance);
ALTER TABLE tbl_performance_staff ADD FOREIGN KEY (staff_ssn) REFERENCES staff (ssn);
ALTER TABLE staff ADD FOREIGN KEY (cod_department) REFERENCES department (cod_department);
ALTER TABLE staff_phone ADD FOREIGN KEY (ssn) REFERENCES staff(ssn);
ALTER TABLE staff_email ADD FOREIGN KEY (ssn) REFERENCES staff(ssn);
ALTER TABLE performance ADD FOREIGN KEY (id_band) REFERENCES band (id_band);
ALTER TABLE tbl_agent_email ADD FOREIGN KEY (cod_agent) REFERENCES booking_agent (cod_agent);
ALTER TABLE tbl_agent_phone ADD FOREIGN KEY (cod_agent) REFERENCES booking_agent (cod_agent);
ALTER TABLE band ADD FOREIGN KEY (cod_city) REFERENCES city (cod_city);
ALTER TABLE band ADD FOREIGN KEY (cod_agent) REFERENCES booking_agent (cod_agent);
ALTER TABLE members ADD FOREIGN KEY (cod_city) REFERENCES  city (cod_city);
ALTER TABLE tbl_instrument ADD FOREIGN KEY (cod_member) REFERENCES members (cod_member);
ALTER TABLE members ADD FOREIGN KEY (band_id) REFERENCES band (id_band);
ALTER TABLE tbl_ranking_band ADD FOREIGN KEY (id_band) REFERENCES band(id_band);
ALTER TABLE tbl_ranking_band ADD FOREIGN KEY (id_band_ranking) REFERENCES band_ranking(id_band_ranking);
```

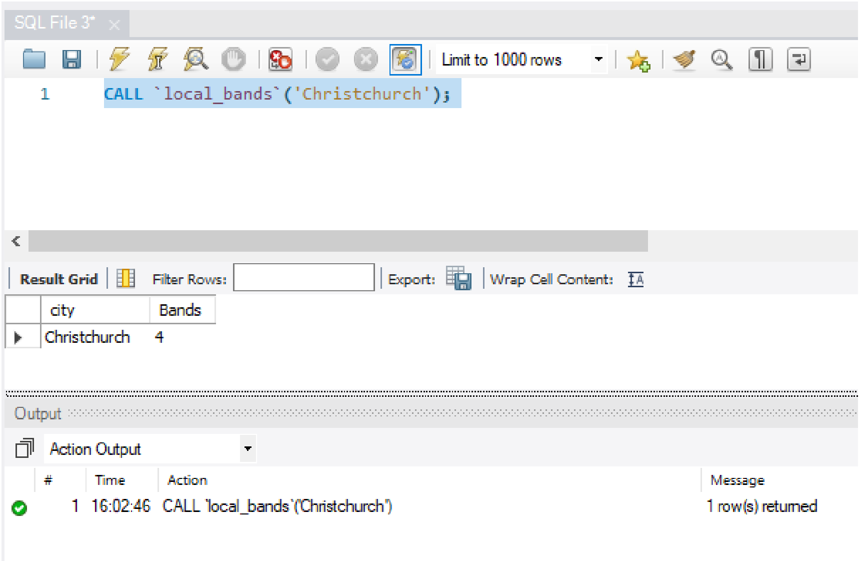
```
Output
Action Output
#   Time      Action                                                             Message                                                  Duration / Fetch
12  12:30:44  ALTER TABLE tbl_instrument ADD FOREIGN KEY (cod_member) REFEREN...  0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0  0.141 sec
13  12:30:44  ALTER TABLE members ADD FOREIGN KEY (band_id) REFERENCES band...    0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0  0.110 sec
14  12:30:44  ALTER TABLE tbl_ranking_band ADD FOREIGN KEY (id_band) REFERENC...  0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0  0.110 sec
15  12:30:44  ALTER TABLE tbl_ranking_band ADD FOREIGN KEY (id_band_ranking) RE...  0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0  0.125 sec
Query Completed
```

ALTER TABLE tickets ADD FOREIGN KEY (id_performance) REFERENCES performance(id_performance);
ALTER TABLE tbl_performance_staff ADD FOREIGN KEY(performance_id) REFERENCES performance(id_performance);
ALTER TABLE tbl_performance_staff ADD FOREIGN KEY (staff_ssn) REFERENCES staff (ssn);
ALTER TABLE staff ADD FOREIGN KEY (cod_department) REFERENCES department (cod_department);
ALTER TABLE staff_phone ADD FOREIGN KEY (ssn) REFERENCES staff(ssn);
ALTER TABLE staff_email ADD FOREIGN KEY (ssn) REFERENCES staff(ssn);
ALTER TABLE performance ADD FOREIGN KEY (id_band) REFERENCES band (id_band);
ALTER TABLE tbl_agent_email ADD FOREIGN KEY (cod_agent) REFERENCES booking_agent (cod_agent);
ALTER TABLE tbl_agent_phone ADD FOREIGN KEY (cod_agent) REFERENCES booking_agent (cod_agent);
ALTER TABLE band ADD FOREIGN KEY (cod_city) REFERENCES city (cod_city);
ALTER TABLE band ADD FOREIGN KEY (cod_agent) REFERENCES booking_agent (cod_agent);
ALTER TABLE members ADD FOREIGN KEY (cod_city) REFERENCES city (cod_city);
ALTER TABLE tbl_instrument ADD FOREIGN KEY (cod_member) REFERENCES members (cod_member);

```
ALTER TABLE members ADD FOREIGN KEY (band_id) REFERENCES band (id_band);
ALTER TABLE tbl_ranking_band ADD FOREIGN KEY (id_band) REFERENCES band(id_band);
ALTER TABLE tbl_ranking_band ADD FOREIGN KEY (id_band_ranking) REFERENCES
band_ranking(id_band_ranking);
```

Procedures:

1. local_bands ()



```
CALL `local_bands`('Christchurch');
```

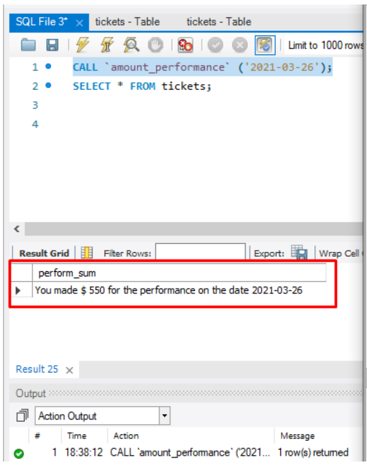
city	Bands
Christchurch	4

```
#   Time      Action                                  Message
1   16:02:46  CALL `local_bands`('Christchurch')       1 row(s) returned
```

```
CREATE PROCEDURE `local_bands` (in cityName VARCHAR(20))
BEGIN
SELECT
            c.city,   COUNT(b.band) as Bands
FROM band b
INNER JOIN city c
ON b.cod_city = c.cod_city
GROUP BY
           city
HAVING
 c.city = cityName;

END
```

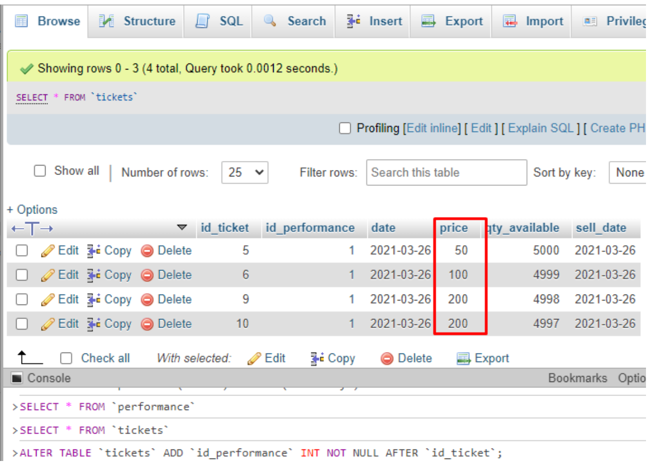
2. amount_performance () Lists the amount received by a specific performance's date having a SUM of all the sold tickets



```
CALL `amount_performance` ('2021-03-26');
SELECT * FROM tickets;
```

perform_sum
You made $ 550 for the performance on the date 2021-03-26

#	Time	Action	Message
1	18:38:12	CALL `amount_perfomance` ('2021...	1 row(s) returned



✓ Showing rows 0 - 3 (4 total, Query took 0.0012 seconds.)

```
SELECT * FROM `tickets`
```

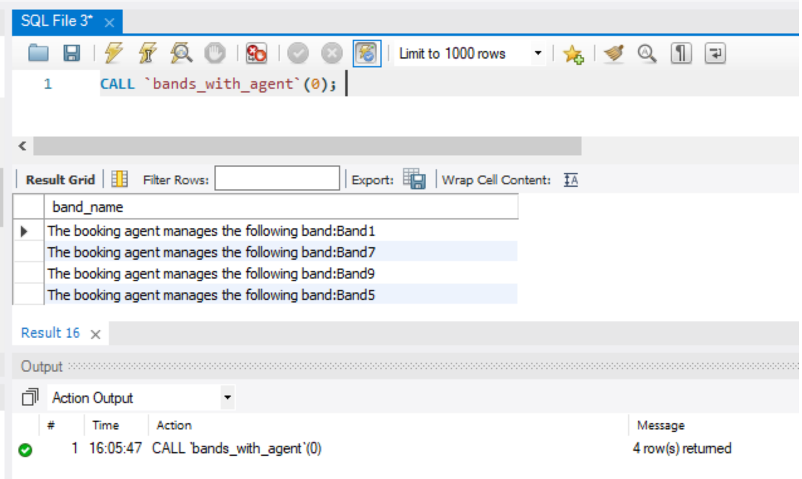
	id_ticket	id_performance	date	price	qty_available	sell_date
□ Edit Copy Delete	5	1	2021-03-26	50	5000	2021-03-26
□ Edit Copy Delete	6	1	2021-03-26	100	4999	2021-03-26
□ Edit Copy Delete	9	1	2021-03-26	200	4998	2021-03-26
□ Edit Copy Delete	10	1	2021-03-26	200	4997	2021-03-26

```
>SELECT * FROM `performance`
>SELECT * FROM `tickets`
>ALTER TABLE `tickets` ADD `id_performance` INT NOT NULL AFTER `id_ticket`;
```

```
 CREATE PROCEDURE `amount_performance` (dateTicket VARCHAR(10))
BEGIN
SELECT CONCAT('You made $ ',SUM(price),' for the performance on the date ', date)
AS perform_sum
FROM tickets
WHERE date = dateTicket;

END
```

3. bands_with_agent() Lists how many bands have a booking agent



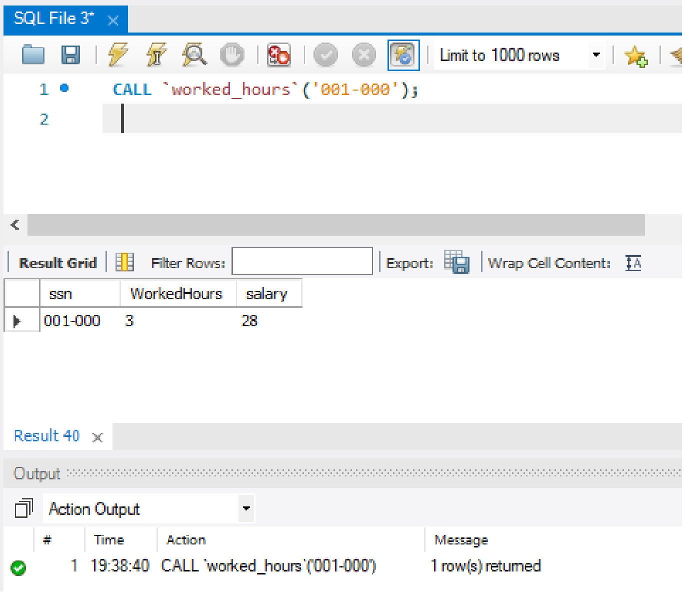
```
CALL `bands_with_agent`(0);
```

band_name
The booking agent manages the following band:Band1
The booking agent manages the following band:Band7
The booking agent manages the following band:Band9
The booking agent manages the following band:Band5

#	Time	Action	Message
1	16:05:47	CALL `bands_with_agent`(0)	4 row(s) returned

```
CREATE PROCEDURE `bands_with_agent` (agentNumber smallint)
BEGIN
SELECT CONCAT('The booking agent manages the following band:', band) AS
band_name
FROM band
WHERE cod_agent = agentNumber;
END
```

4. worked_hours() Sum the worked hours in the performances from a staff



```
CALL `worked_hours`('001-000');
```

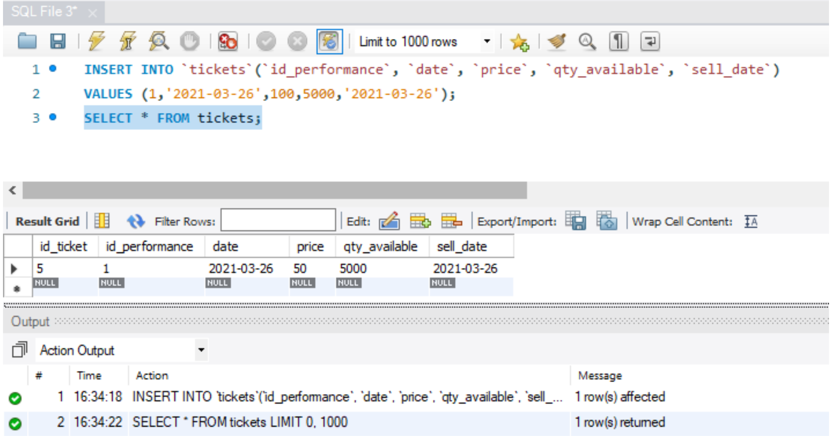
ssn	WorkedHours	salary
001-000	3	28

#	Time	Action	Message
1	19:38:40	CALL `worked_hours`('001-000')	1 row(s) returned

CREATE PROCEDURE `worked_hours`(in staffSSN VARCHAR(7))
BEGIN
SELECT
    s.ssn, COUNT(p.duration) AS WorkedHours, s.salary
FROM
    `tbl_performance_staff` AS ps
        INNER JOIN
    staff s ON ps.staff_ssn = s.ssn
        INNER JOIN
    performance p ON ps.performance_id = p.id_performance
GROUP BY s.ssn
HAVING s.ssn = staffSSN;
END

Triggers:

1. tr_discount_ticket:



```
INSERT INTO `tickets`(`id_performance`, `date`, `price`, `qty_available`, `sell_date`)
VALUES (1,'2021-03-26',100,5000,'2021-03-26');
SELECT * FROM tickets;
```

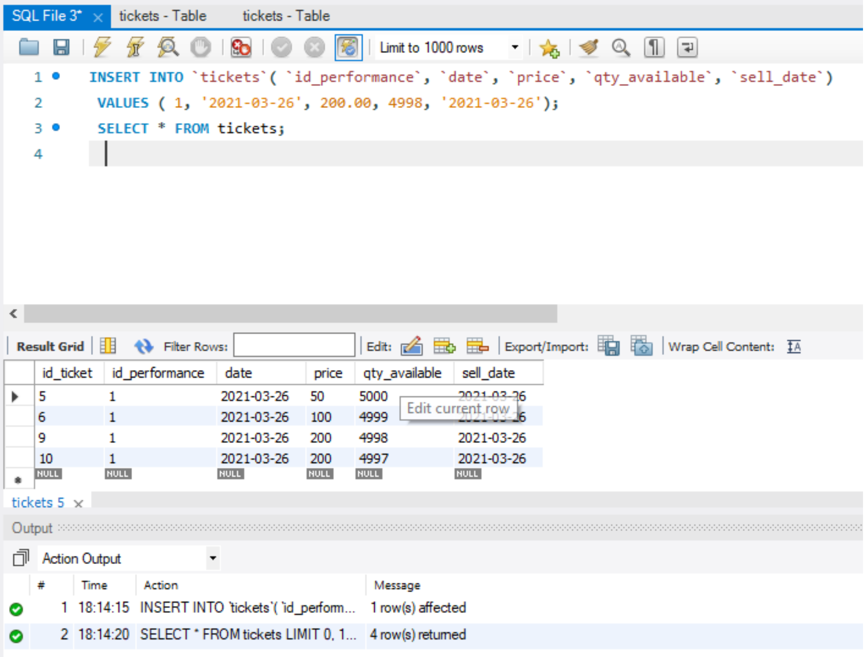
id_ticket	id_performance	date	price	qty_available	sell_date
5	1	2021-03-26	50	5000	2021-03-26
NULL	NULL	NULL	NULL	NULL	NULL

#	Time	Action	Message
1	16:34:18	INSERT INTO `tickets`(`id_performance`, `date`, `price`, `qty_available`, `sell_...	1 row(s) affected
2	16:34:22	SELECT * FROM tickets LIMIT 0, 1000	1 row(s) returned

CREATE TRIGGER tr_discount_ticket BEFORE INSERT
ON tickets

FOR EACH ROW
SET NEW.price = (NEW.price *0.50);

2. tr_quantity_tickets:



```
INSERT INTO `tickets`( `id_performance`, `date`, `price`, `qty_available`, `sell_date`)
 VALUES ( 1, '2021-03-26', 200.00, 4998, '2021-03-26');
 SELECT * FROM tickets;
```

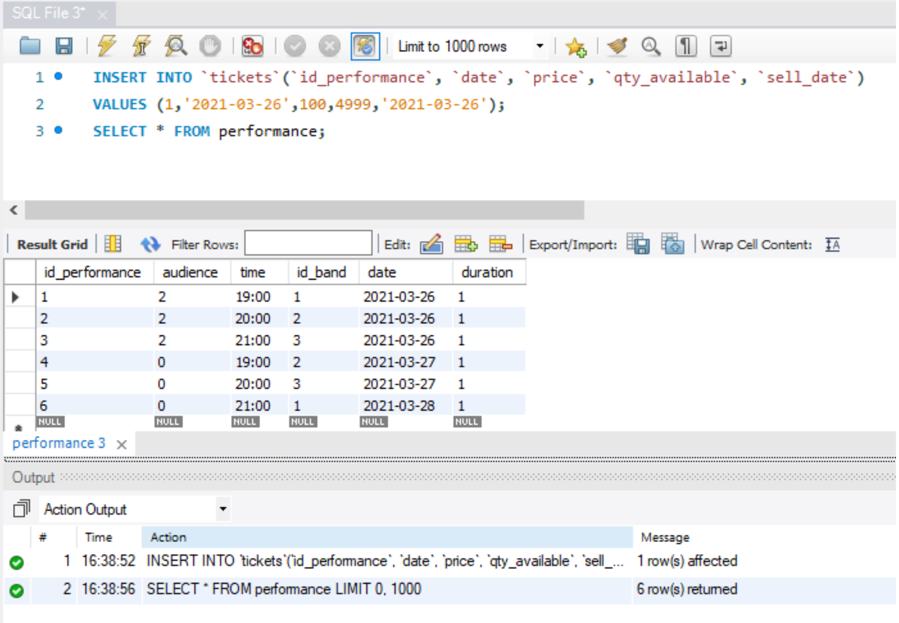
id_ticket	id_performance	date	price	qty_available	sell_date
5	1	2021-03-26	50	5000	2021-03-26
6	1	2021-03-26	100	4999	2021-03-26
9	1	2021-03-26	200	4998	2021-03-26
10	1	2021-03-26	200	4997	2021-03-26
NULL	NULL	NULL	NULL	NULL	NULL

#	Time	Action	Message
1	18:14:15	INSERT INTO `tickets`(`id_perform...	1 row(s) affected
2	18:14:20	SELECT * FROM tickets LIMIT 0, 1...	4 row(s) returned

CREATE TRIGGER tr_quantity_tickets BEFORE INSERT
ON tickets
FOR EACH ROW BEGIN
IF NEW.qty_available > 0 THEN
SET NEW.qty_available= (NEW.qty_available - 1);

END IF; END//

3. tr_performance_audience:



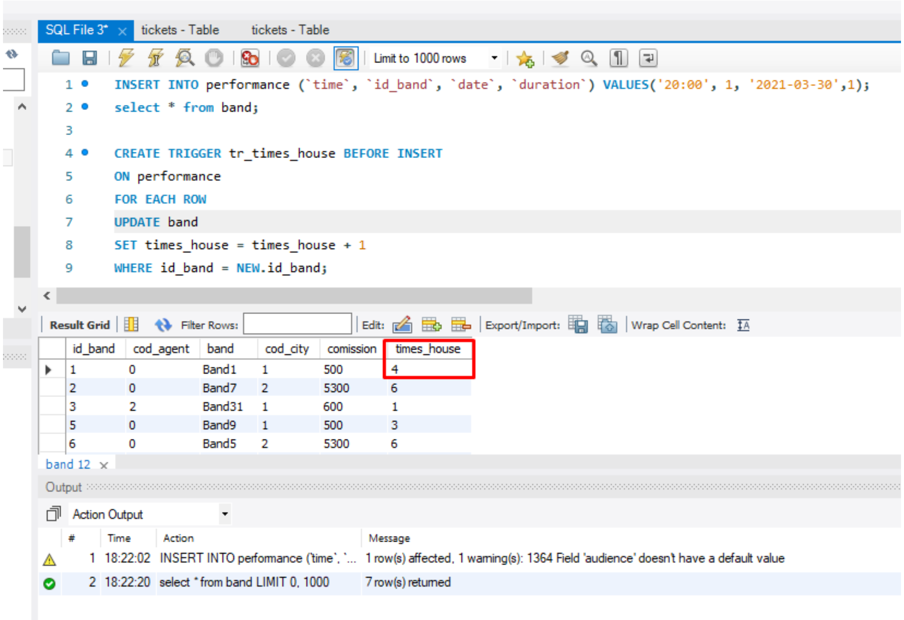
```
INSERT INTO `tickets`(`id_performance`, `date`, `price`, `qty_available`, `sell_date`)
VALUES (1,'2021-03-26',100,4999,'2021-03-26');
SELECT * FROM performance;
```

id_performance	audience	time	id_band	date	duration
1	2	19:00	1	2021-03-26	1
2	2	20:00	2	2021-03-26	1
3	2	21:00	3	2021-03-26	1
4	0	19:00	2	2021-03-27	1
5	0	20:00	3	2021-03-27	1
6	0	21:00	1	2021-03-28	1
NULL	NULL	NULL	NULL	NULL	NULL

#	Time	Action	Message
1	16:38:52	INSERT INTO `tickets`(`id_performance`, `date`, `price`, `qty_available`, `sell_...	1 row(s) affected
2	16:38:56	SELECT * FROM performance LIMIT 0, 1000	6 row(s) returned

```
CREATE TRIGGER tr_performance_audience AFTER INSERT
ON tickets
FOR EACH ROW
UPDATE performance
SET audience = audience + 1
WHERE date = NEW.date;
```

4. tr_times_house:



```
INSERT INTO performance (`time`, `id_band`, `date`, `duration`) VALUES('20:00', 1, '2021-03-30',1);
select * from band;

CREATE TRIGGER tr_times_house BEFORE INSERT
ON performance
FOR EACH ROW
UPDATE band
SET times_house = times_house + 1
WHERE id_band = NEW.id_band;
```

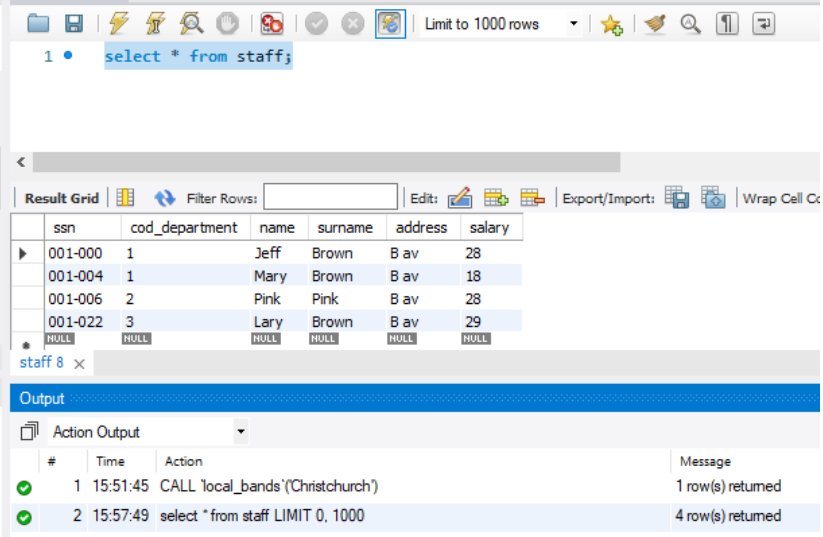
id_band	cod_agent	band	cod_city	comission	times_house
1	0	Band1	1	500	4
2	0	Band7	2	5300	6
3	2	Band31	1	600	1
5	0	Band9	1	500	3
6	0	Band5	2	5300	6

#	Time	Action	Message
1	18:22:02	INSERT INTO performance (`time`, `...	1 row(s) affected, 1 warning(s): 1364 Field 'audience' doesn't have a default value
2	18:22:20	select * from band LIMIT 0, 1000	7 row(s) returned

```
CREATE TRIGGER tr_times_house AFTER INSERT
ON performance
FOR EACH ROW
UPDATE band
SET times_house = times_house + 1
WHERE id_band = NEW.id_band;
```

- System running and output

Staff table report:

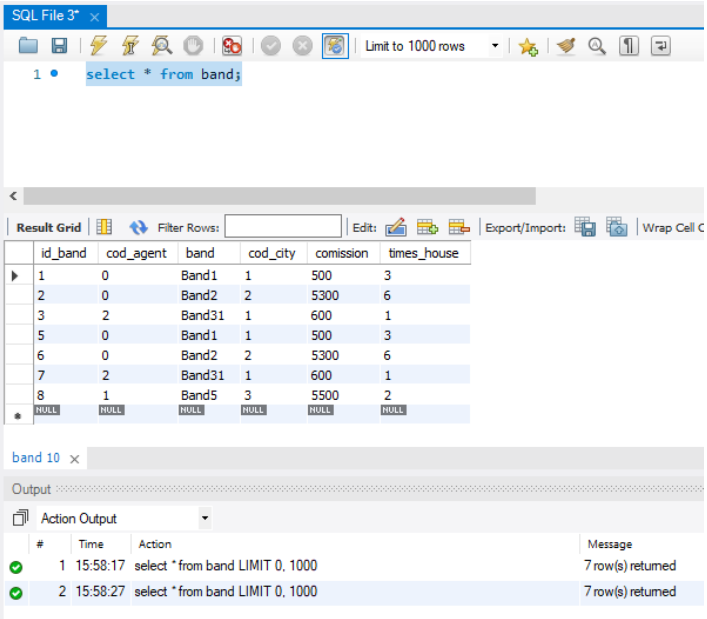


```
1 • select * from staff;
```

ssn	cod_department	name	surname	address	salary
001-000	1	Jeff	Brown	B av	28
001-004	1	Mary	Brown	B av	18
001-006	2	Pink	Pink	B av	28
001-022	3	Lary	Brown	B av	29
NULL	NULL	NULL	NULL	NULL	NULL

#	Time	Action	Message
1	15:51:45	CALL 'local_bands'('Christchurch')	1 row(s) returned
2	15:57:49	select * from staff LIMIT 0, 1000	4 row(s) returned

Band table report:

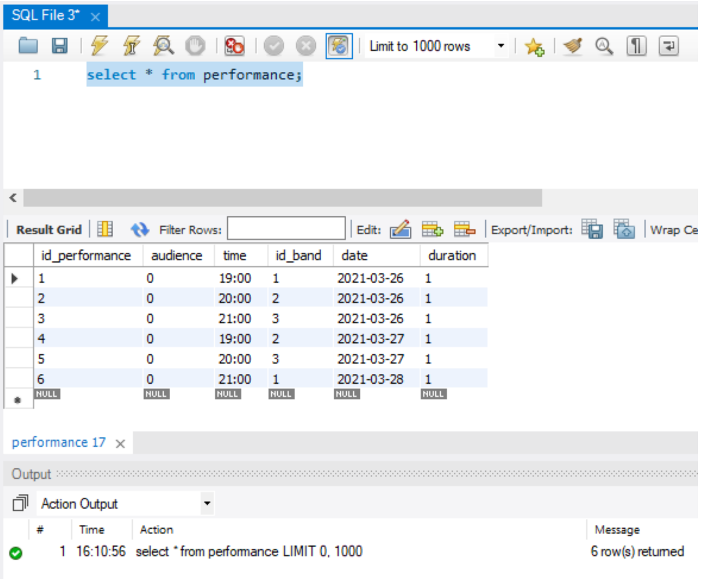


```
1 • select * from band;
```

id_band	cod_agent	band	cod_city	comission	times_house
1	0	Band1	1	500	3
2	0	Band2	2	5300	6
3	2	Band31	1	600	1
5	0	Band1	1	500	3
6	0	Band2	2	5300	6
7	2	Band31	1	600	1
8	1	Band5	3	5500	2
NULL	NULL	NULL	NULL	NULL	NULL

#	Time	Action	Message
1	15:58:17	select * from band LIMIT 0, 1000	7 row(s) returned
2	15:58:27	select * from band LIMIT 0, 1000	7 row(s) returned

Performance report:



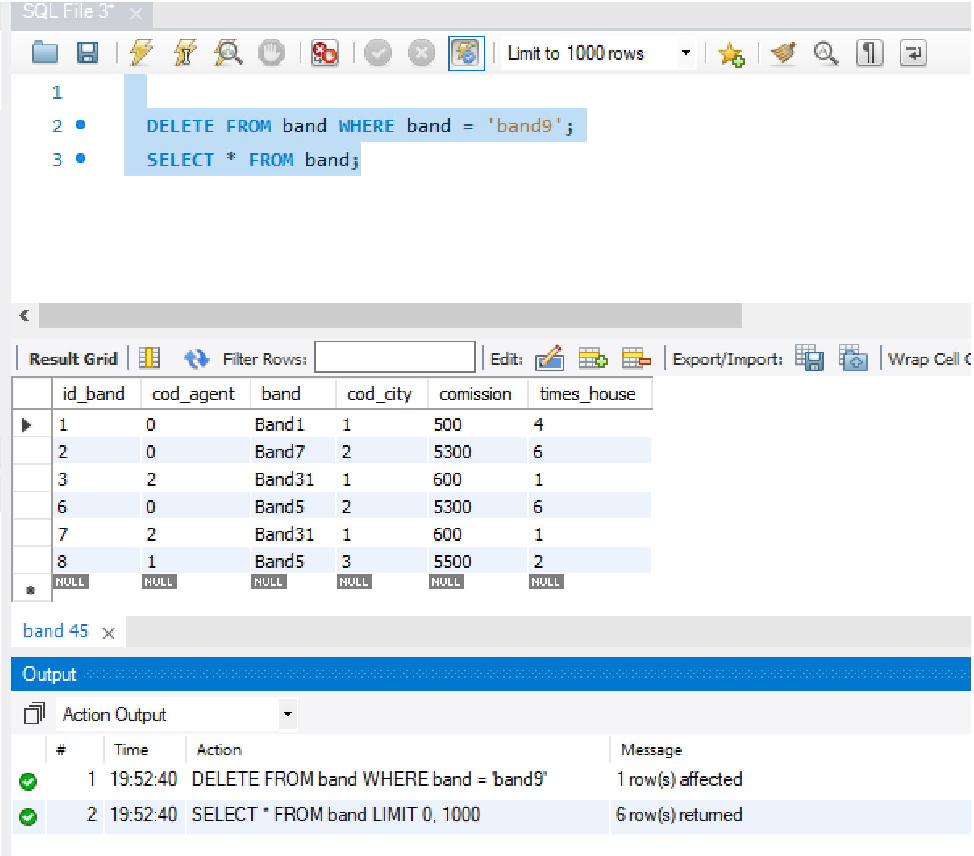
```
1 select * from performance;
```

id_performance	audience	time	id_band	date	duration
1	0	19:00	1	2021-03-26	1
2	0	20:00	2	2021-03-26	1
3	0	21:00	3	2021-03-26	1
4	0	19:00	2	2021-03-27	1
5	0	20:00	3	2021-03-27	1
6	0	21:00	1	2021-03-28	1
NULL	NULL	NULL	NULL	NULL	NULL

```
#  Time      Action                                 Message
1  16:10:56  select * from performance LIMIT 0, 1000  6 row(s) returned
```

Task 4 – Testing

Register deletion having the row with a value ‘band9’:

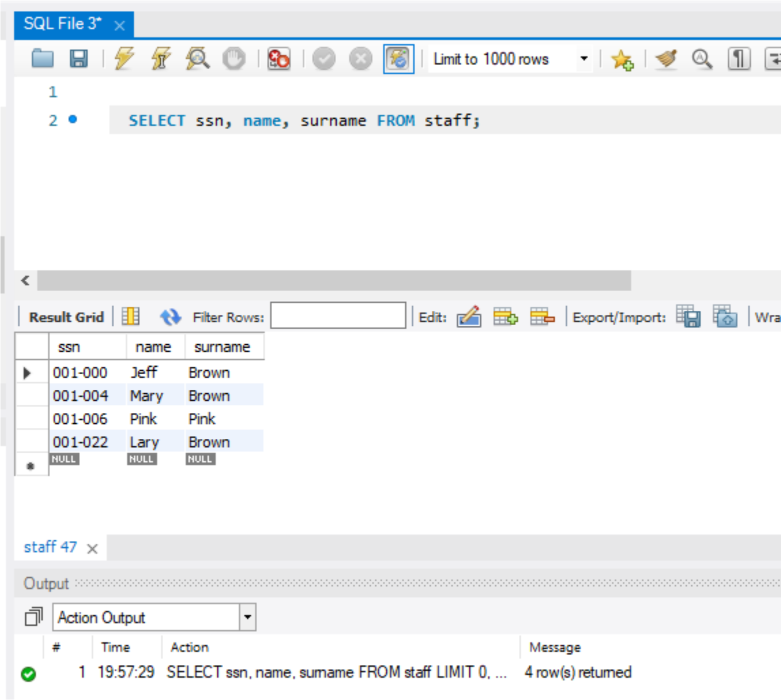


```
1
2 DELETE FROM band WHERE band = 'band9';
3 SELECT * FROM band;
```

id_band	cod_agent	band	cod_city	comission	times_house
1	0	Band1	1	500	4
2	0	Band7	2	5300	6
3	2	Band31	1	600	1
6	0	Band5	2	5300	6
7	2	Band31	1	600	1
8	1	Band5	3	5500	2
NULL	NULL	NULL	NULL	NULL	NULL

```
#  Time      Action                                 Message
1  19:52:40  DELETE FROM band WHERE band = 'band9'  1 row(s) affected
2  19:52:40  SELECT * FROM band LIMIT 0, 1000       6 row(s) returned
```

Retrieving information from three columns:

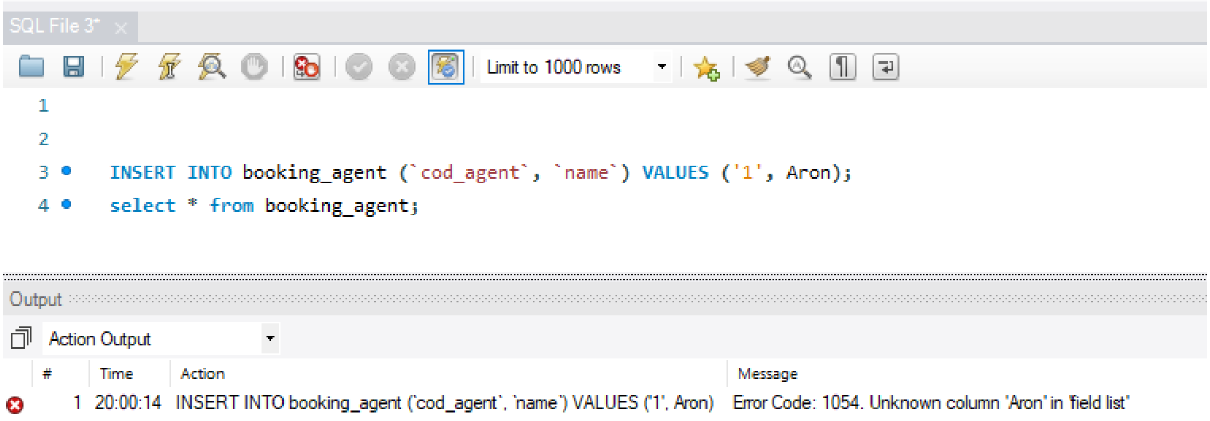


```
SELECT ssn, name, surname FROM staff;
```

ssn	name	surname
001-000	Jeff	Brown
001-004	Mary	Brown
001-006	Pink	Pink
001-022	Lary	Brown
NULL	NULL	NULL

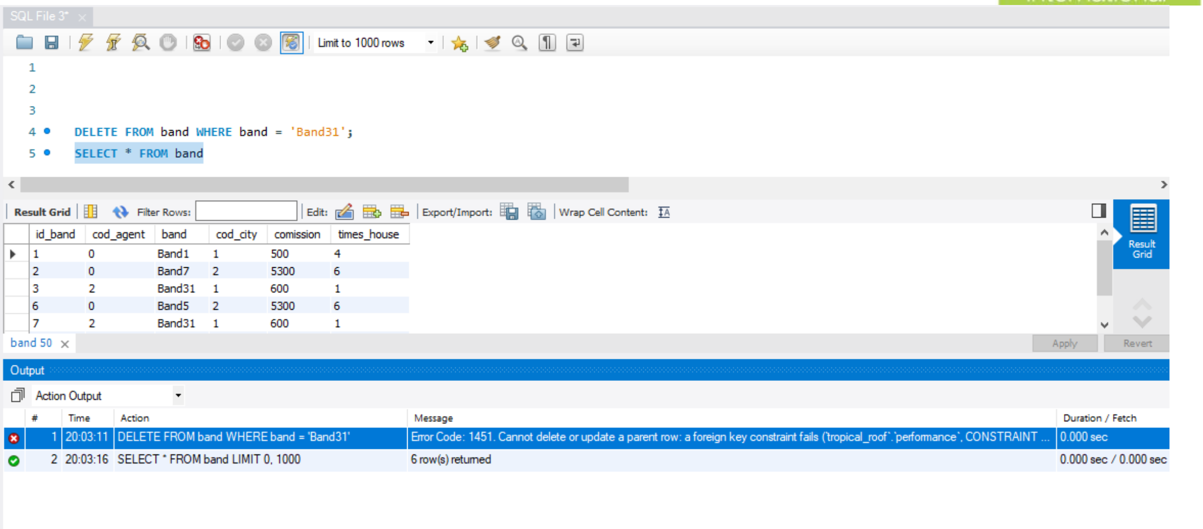
```
#   Time      Action                                          Message
1   19:57:29  SELECT ssn, name, surname FROM staff LIMIT 0, ...  4 row(s) returned
```

Data integrity:



```
INSERT INTO booking_agent (`cod_agent`, `name`) VALUES ('1', Aron);
select * from booking_agent;
```

```
#   Time      Action                                                         Message
1   20:00:14  INSERT INTO booking_agent (`cod_agent`, `name`) VALUES ('1', Aron)  Error Code: 1054. Unknown column 'Aron' in 'field list'
```



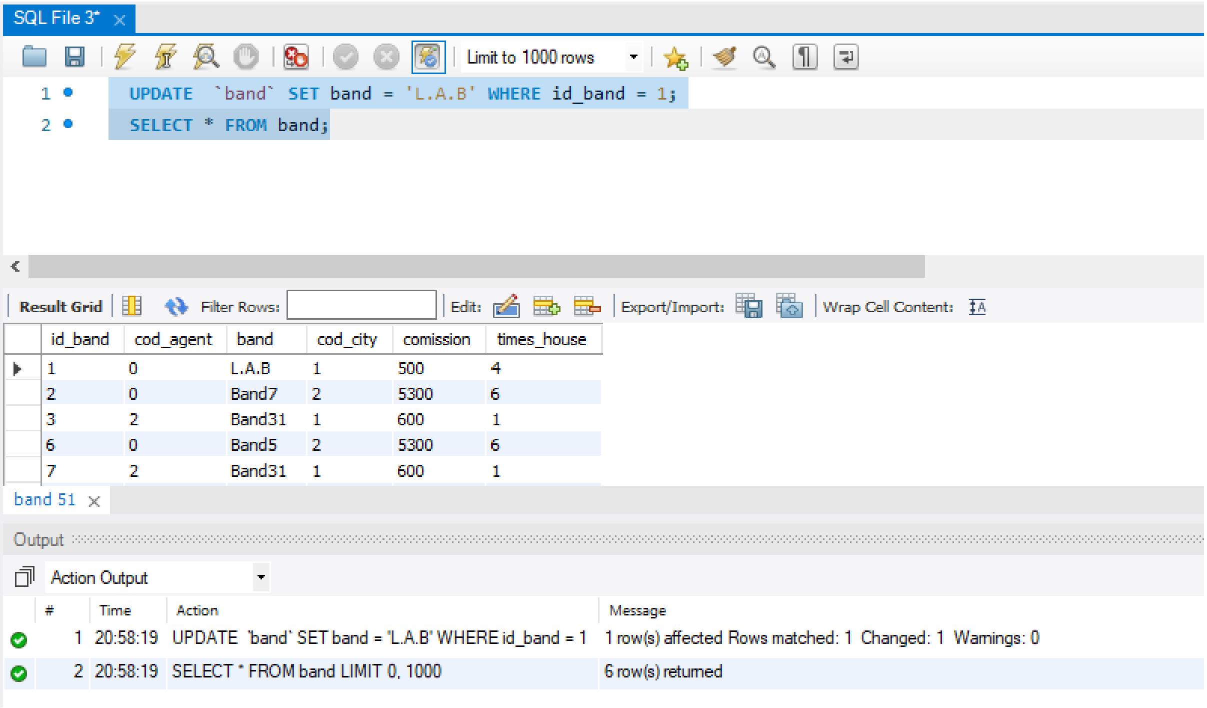
```sql


DELETE FROM band WHERE band = 'Band31';
SELECT * FROM band
```

id_band	cod_agent	band	cod_city	comission	times_house
1	0	Band1	1	500	4
2	0	Band7	2	5300	6
3	2	Band31	1	600	1
6	0	Band5	2	5300	6
7	2	Band31	1	600	1

```
#  Time      Action                                 Message                                                                                                                     Duration / Fetch
1  20:03:11  DELETE FROM band WHERE band = 'Band31'  Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails ('tropical_roof'.'performance', CONSTRAINT ...  0.000 sec
2  20:03:16  SELECT * FROM band LIMIT 0, 1000       6 row(s) returned                                                                                                           0.000 sec / 0.000 sec
```

Updating the table band:



```sql
UPDATE `band` SET band = 'L.A.B' WHERE id_band = 1;
SELECT * FROM band;
```

id_band	cod_agent	band	cod_city	comission	times_house
1	0	L.A.B	1	500	4
2	0	Band7	2	5300	6
3	2	Band31	1	600	1
6	0	Band5	2	5300	6
7	2	Band31	1	600	1

```
#  Time      Action                                             Message
1  20:58:19  UPDATE `band` SET band = 'L.A.B' WHERE id_band = 1  1 row(s) affected Rows matched: 1  Changed: 1  Warnings: 0
2  20:58:19  SELECT * FROM band LIMIT 0, 1000                   6 row(s) returned
```

References

GURU99. (n.d.). *What is Normalization? 1NF, 2NF, 3NF, BCNF Database Example*. Retrieved from Guru99: https://www.guru99.com/database-normalization.html

mysqltutorial. (n.d.). *MySQLTUTORIAL*. Retrieved from My SQL: https://www.mysqltutorial.org/