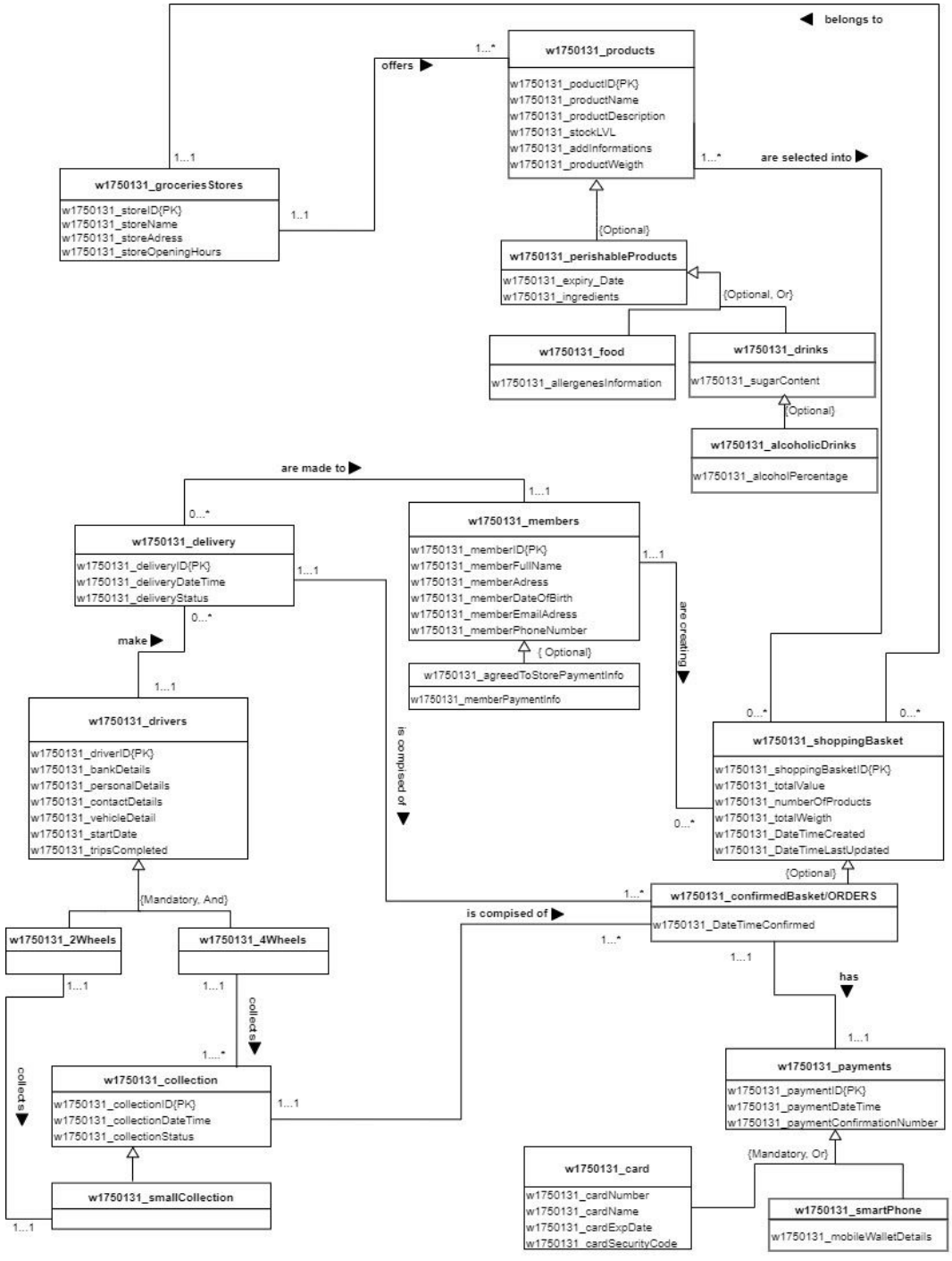
Student name: Cosmin Alexandru Stefanescu
Student ID: w1750131
Tutorial group attended: Every Tuesday 2:00 PM to 4:00 PM 5CS09

5COSC002W DATABASE SYSTEMS COURSEWORK

Module leader: Francois ROUBERT
Tutorial teacher: Tasos Ptohos

Student name: Cosmin Alexandru Stefanescu
Student ID: w1750131
Tutorial group attended: Every Tuesday 2:00 PM to 4:00 PM 5CS09

PART A:



Module leader: Francois ROUBERT
Tutorial teacher: Tasos Ptohos

Student name: Cosmin Alexandru Stefanescu
Student ID: w1750131
Tutorial group attended: Every Tuesday 2:00 PM to 4:00 PM 5CS09

Entity name	***Description***
w1750131_groceriesStores	This entity represents the groceries stores that are working with FOODTOOYOO.
w1750131_collection	This entity represents the collections made by the drivers. It is an entity and not a relationship because it is an EVENT that needs to be carefully recorded.
w1750131_delivery	This entity represents the deliveries made by the drivers at the addresses of the customers. It is an entity and not a relationship because it is an EVENT that needs to be carefully recorded.
w1750131_products	This entity represents the products provided by each store.
w1750131_members	This entity represents the people who registered an account with FOODTOOYOO.
w1750131_shoppingBasket	This entity represents the shopping baskets created by members.
w1750131_payments	This entity represents the payments made by the customer to FOODTOOYOO for each confirmed basket.
w1750131_drivers	This entity represents the drivers contracted by FOODTOOYOO to collect and deliver the orders(confirmed shopping baskets) to the members.

General entity	***Specialized entity***	***Explanation***
w1750131_products	w1750131_perishableProducts	Groceries are divided into 2 main categories: perishable and nonperishable. FOODTOOYOO is interested to record different attributes for the perishable products only.
w1750131_perishableProducts	w1750131_food	There are 2 types of perishable

		products that have unique attributes and FOOTOOYOO wants to be carefully recorded, food and drinks.
	w1750131_drinks	There are 2 types of perishable products that have unique attributes and FOOTOOYOO wants to be carefully recorded, food and drinks.
w1750131_drinks	w1750131_alcoholicDrinks	In the drinks category, we need to carefully record the alcohol per volume for alcoholic drinks.
w1750131_members	w1750131_agreedToStorePaymentInfo	Some members agree for us to encrypt and store their payment info(card information) for easier future checkouts.
w1750131_shoppingBasket	w1750131_confirmedBasket/ORDERS	The baskets that are confirmed, have a valid payment and are ready to be collected and delivered.
w1750131_payments	w1750131_card	The payments that are made by card.
	w1750131_smartPhone	The payments that are made by mobile wallet.
w1750131_drivers	w1750131_4Wheels	The drivers who use a vehicle that has four wheels(car/van) to collect and deliver the orders.
	w1750131_2Wheels	The drivers who use a vehicle that has two wheels(bike/moped/motorcycle).

Student name: Cosmin Alexandru Stefanescu
Student ID: w1750131
Tutorial group attended: Every Tuesday 2:00 PM to 4:00 PM 5CS09

w1750131_collection	w1750131_smallCollection	The collections that have a small weight or number of products, suitable to be collected and delivered by 2 Wheelers.

Entity name	***Multiplicity***	***Relationship***	***Multiplicity***	***Entity name***	***Justification for the multiplicity***
w1750131_groceriesStores	1...1	offers	1...*	w1750131_products	Each product must be offered by a grocery store.
					Each product can be offered only by one grocery store as they are unique to them.
					Each grocery store must offer at least 1 product.
					Each grocery store can offer many products.
w1750131_shoppingBasket	0...*	Belongs to	1...1	w1750131_groceriesStores	A shopping basket must belong to a grocery store.
					A shopping basket can belong to only one grocery store.
					A grocery store can have 0 shopping baskets created by the members with their products.
					A grocery store can have many shopping baskets created by the members with their products.
w1750131_products	1...*	are selected into	0...*	w1750131_shoppingBasket	A product can be selected in 0 shopping baskets.
					A product can be selected in many shopping baskets.
					A shopping basket must have at least 1 product selected.

Module leader: Francois ROUBERT
Tutorial teacher: Tasos Ptohos

Student name: Cosmin Alexandru Stefanescu
Student ID: w1750131
Tutorial group attended: Every Tuesday 2:00 PM to 4:00 PM 5CS09

					A shopping basket can have many products selected.
w1750131_delivery	0...*	are made to	1...1	w1750131_members	Each delivery must have one member to be delivered to.
					A delivery can be made to only one member.
					A member can have 0 deliveries made to them(as they didn't order anything yet).
					A member can have many deliveries made to them.
w1750131_members	1...1	Are creating	0...*	w1750131_shoppingBasket	A member can have 0 baskets created.
					A member can have many baskets created.
					A basket must be created by one member.
					A basket can be created by one member only.
w1750131_confirmedBasket/ORDERS	1...1	has	1...1	w1750131_payments	A confirmed basket/order must have a confirmed payment.
					A confirmed basket/order can have only one confirmed payment.
					A confirmed payment must belong to a confirmed basket/order.
					A confirmed payment can belong to only one confirmed basket/order.
w1750131_2Wheels	1...1	collects	0...*	w1750131_smallCollection	A driver that uses a vehicle with 2 wheels might not have collected any orders so far.
					A driver that uses a vehicle with 2 wheels can have collected many orders.

Module leader: Francois ROUBERT
Tutorial teacher: Tasos Ptohos

					A small collection must be collected by one driver that uses a vehicle with 2 wheels.
					A small collection can be collected by only one driver that uses a vehicle with 2 wheels.
w1750131_4Wheels	1...1	collects	0...*	w1750131_collection	A driver that uses a vehicle with 4 wheels might not have done any collection so far.
					A driver that uses a vehicle with 4 wheels might have done many collections.
					A collection must be done by one driver.
					A collection can be done by only one driver.
w1750131_drivers	1...1	make	0...*	w1750131_delivery	A driver can have 0 orders delivered.
					A driver can have many orders delivered.
					A delivery must be delivered by one driver.
					A delivery can be delivered by a maximum of one driver.
w1750131_delivery	1...1	Is comprised of	1...*	w1750131_confirmedBasket/ORDERS	A delivery must have at least a shopping basket to be delivered.
					A delivery can have many shopping baskets to be delivered.
					An order must belong to a delivery.
					An order can belong to only one delivery.
w1750131_collection	1...1	Is comprised of	1...*	w1750131_confirmedBasket/ORDERS	A collection must have at least a shopping basket to be collected.
					A collection can have many shopping baskets to be

					collected.
					An order must belong to a collection.
					An order can belong to only one collection.

Entity name	***Attributes for this entity(include PK)***	***Justification***
w1750131_groceriesStores	w1750131_storeID{PK}	For each store that works with FOODTOOYOO will be assigned a unique identification number. This is the most suitable candidate key to be selected as a primary key.
	w1750131_storeName	The name of the grocery store is essential so everybody knows where the products are coming from.
	w1750131_storeAdress	The address of the grocery store is used by the drivers to locate the shop and by the customers to know where they are buying from.
	w1750131_storeOpeningHours	The Opening hours of the store are very important for the customers and the drivers.
w1750131_collection	w1750131_collectionID{PK}	For each collection that is made at a store will be assigned a unique identification number. This is the most suitable candidate key to be selected as a primary key.
	w1750131_collectionDateTime	This attribute stores the date and time when the collection was made.
	w1750131_collectionStatus	This attribute represents the status of the collection(getting it ready for collection, ready to be collected, collected).
w1750131_delivery	w1750131_deliveryID{PK}	For each delivery that is made to a member will be assigned a unique identification number. This is the most suitable candidate key to be selected as a primary key.
	w1750131_deliveryDateTime	This attribute stores the date and time when the delivery was made.

	w1750131_deliveryStatus	This attribute represents the status of the delivery(pending, in progress, delivered).
w1750131_products	w1750131_productId{PK}	For each product that the groceries stores who are working with FOODTOOYOO offer will be assigned a unique identification number. This is the most suitable candidate key to be selected as a primary key.
	w1750131_productName	The name of the product used by the customers to identify the products they need.
	w1750131_productDescription	A brief description for each product which is helpful for the customer in deciding what to buy.
	w1750131_stockLVL	The amount of stock for each product is used to avoid over-ordering or ordering a product that is not available.
	w1750131_addInformations	Additional information provided for each product, such as: “freshly baked”, “frozen”, ”refurbished” etc.
	w1750131_productWeigth	The weight of each product that helps us decide if a collection can be categorized as small or no.
w1750131_perishableProducts	w1750131_expiryDate	The expiry date for all perishable products that is used to control the stock level and that provides useful information to the customers.
	w1750131_ingredients	A list with all the ingredients of the product that provides useful information to the customers.
w1750131_food	w1750131_allergensInformation	For all foods, we must store the allergen information for the customer to be well informed in making their decision regarding what to buy.
w1750131_drinks	w1750131_sugarContent	For all non-alcoholic drinks, we must store the sugar content per volume for the customer to be well informed in making their decision regarding what to buy.
w1750131_alcoholicDrinks	w1750131_alcoholPercentage	For all alcoholic drinks, we must store the alcohol percentage per volume.
w1750131_members	w1750131_memberID{PK}	For each member that registers with us, it

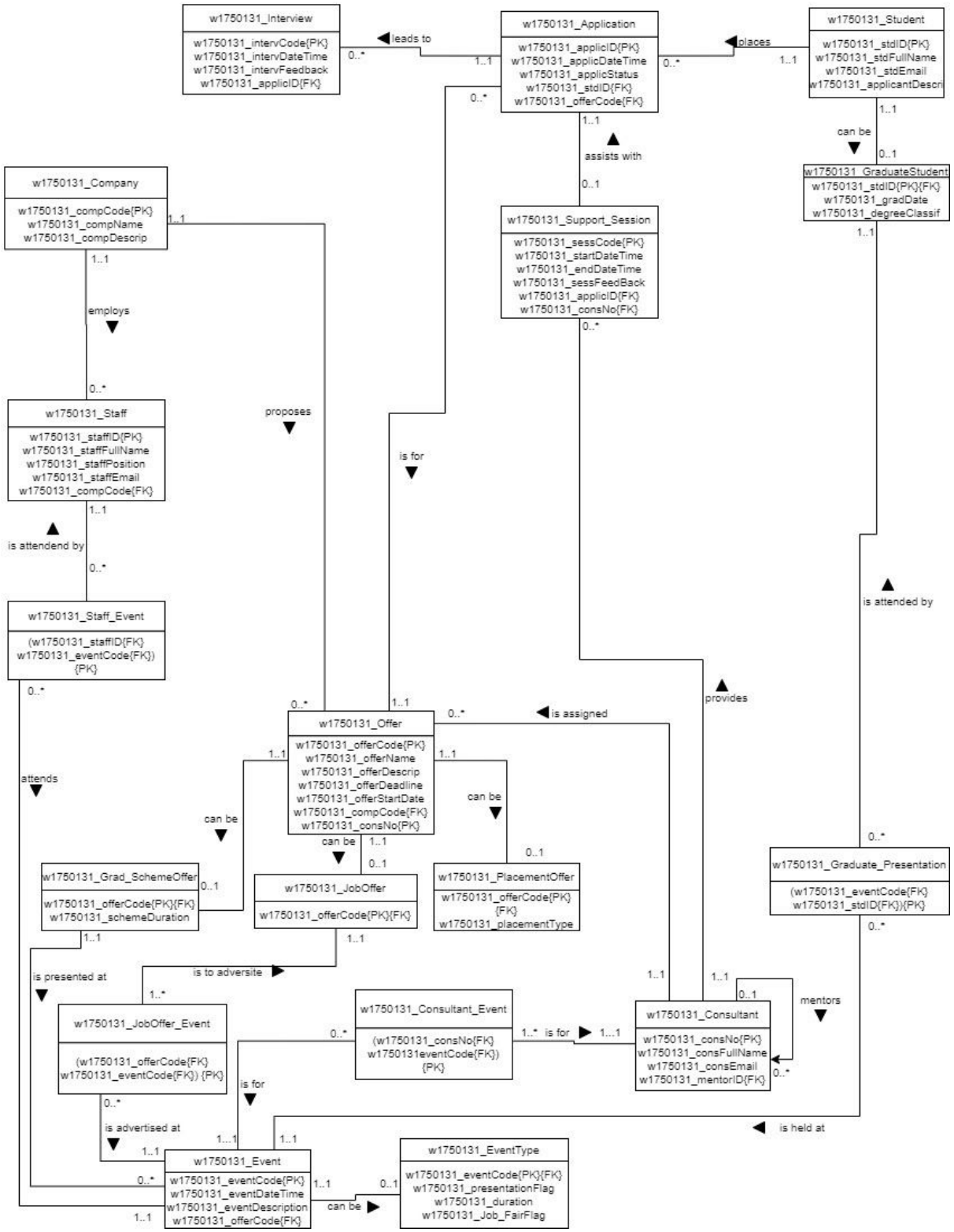
		will be assigned a unique identification number. This is the most suitable candidate key to be selected as a primary key.
	w1750131_memberFullName	The name of the member is useful in case we need to contact him/her or for promotional campaigns.
	w1750131_memberAdress	The address of the customer is essential to deliver the groceries.
	w1750131_memberDateOfBirth	The member's date of birth is essential to know if we can deliver alcohol to this person and to inform the driver to ask for the ID before delivering.
	w1750131_memberEmailAdress	The email address of the member is useful in case we need to contact him/her or for promotional campaigns.
	w1750131_phoneNumber	The phone number of the member is useful in case we need to contact him/her or for promotional campaigns.
w1750131_agreedToStorePaymentInfo	w1750131_memberPaymentInfo	To facilitate the checkout process for future orders, we will store the member's payment info, only if they agree. This data will be encrypted.
w1750131_shoppingBasket	w1750131_shoppingBasketID{PK}	For each shopping basket that is created, it will be assigned a unique identification number. This is the most suitable candidate key to be selected as a primary key.
	w1750131_totalValue	The total value of the shopping basket is essential for the checkout process.
	w1750131_numberOfProducts	This attribute represents the number of products that the basket contains and is very useful in determining if the order is small and can be picked up by a vehicle with two wheels or no.
	w1750131_totalWeight	This attribute represents the total weight of the basked and is very useful in determining if the order is small and can be picked up by a vehicle with two wheels or no.
	w1750131_DateTimeCreated	This attribute represents the date and time when the basket was created.
	w1750131_DateTimeLastUpdated	This attribute represents the date and time

		when the basket was last updated.
w1750131_confirmedBasket/ ORDERS	w1750131_DateTimeConfirmed	This attribute stores the date and time when the order was confirmed and is essential for FOODTOOYOO's delivery process as the delivery must be completed in less than 2 hours.
w1750131_payments	w1750131_paymentID{PK}	For each attempted payment, it will be assigned a unique identification number. This is the most suitable candidate key to be selected as a primary key.
	w1750131_paymentDateTime	This attribute stores the date and time when the payment was attempted and it helps FOODTOOYOO to identify payments.
	w1750131_paymentConfirmationNumber	Each successful payment will have a unique identification number assigned.
w1750131_card	w1750131_Number	This attribute stores the encrypted card number used for payment.
	w1750131_Name	This attribute stores the encrypted name on the card used for payment.
	w1750131_ExpDate	This attribute stores the encrypted card expiry date used for payment.
	w1750131_SecurityCode	This attribute stores the encrypted card security code used for payment.
w1750131_smartPhone	w1750131_mobileWalletDetails	This attribute stores the details of the mobile walled used for payment.
w1750131_drivers	w1750131_driverID{PK}	For each registered driver, it will be assigned a unique identification number. This is the most suitable candidate key to be selected as a primary key.
	w1750131_bankDetails	This attribute stores the driver's bank details to be paid.
	w1750131_personalDetails	This attribute stores the personal details of the driver such as name, address, date of birth, driving license number, NINo, UTR number.
	w1750131_contactDetails	This attribute stores the contact details of the driver.
	w1750131_vehicleDetails	This attribute stores the vehicle/vehicles

		details that the driver uses. Is essential from a legal point of view and very helpful for the customers to identify the delivery driver when he has problems in finding the address.
	w1750131_startDate	This attribute stores the date when the driver has started working with FOODTOOYOO.
	w1750131_tripsCompleted	This attribute stores the trips completed by the driver working for FOODTOOYOO.

PART B:

Student name: Cosmin Alexandru Stefanescu
Student ID: w1750131
Tutorial group attended: Every Tuesday 2:00 PM to 4:00 PM 5CS09



1. I applied rule number 10 for the generalization of the “w1750131_Offer” entity. I will have one parent table for the super-entity “w1750131_Offer” and 3 child tables for the 3 sub-entities that we have. The primary keys of the child tables will be the primary key of the parent table “w1750131_Staff” and the foreign key will reference it. The multiplicities are 1..1 on the parent side and optional (0..1) on the child’s side.

Student name: Cosmin Alexandru Stefanescu
Student ID: w1750131
Tutorial group attended: Every Tuesday 2:00 PM to 4:00 PM 5CS09

2. I applied rule number 8 for the generalisation of the “w1750131_Event” entity. I will have a parent table for the super-entity “w1750131_Event” and a child table being the result of the merging of the 2 sub-entities(“w1750131_Presentation” and “w1750131_Job_Fair”). The child table will have a flag for each sub-entity, to clearly identify what each tuple represents. The primary key for the parent table stays the same and is now the primary key for the child table as well. The foreign key of the child table references the primary key. The multiplicities are 1..1 on the parent side and optional (0..1) on the child’s side
3. I applied rule number 10 for the generalization of the “w1750131_Student” entity. I will have one parent table for the super-entity “w1750131_Staff” and one child table for the sub-entity “Graduate”. The primary key of the child table(“GraduateStudent”) will be the primary key of the parent table(“w1750131_Student”) and the foreign key will reference it. The multiplicities are 1..1 on the parent side and optional (0..1) on the child’s side.
4. I applied rule number 5 for the relationship between the entities “w1750131_GraduateStudent” and “w1750131_Event” as it is a many to many type of relationship. Applying this rule results in a child entity “w1750131_Graduate_Presentation” that has a compound primary key(as the combination of the primary keys that belong to the parents cannot be repeated). The foreign keys of the child table(“w1750131_Graduate_Presentation”) reference the primary keys of the parent tables(“w1750131_GraduateStudent” and “w1750131_Event”). The multiplicities are 1..1 on the parent’s side and 0...* on the child’s side.
5. I applied rule number 5 for the relationship that takes place between “w1750131_Event” and “w1750131_Consultant” as it is a many to many type of relationship. Applying this rule results in a child entity “w1750131_Consultant_Event” that has a compound primary key(as the combination of the primary keys that belong to the parents cannot be repeated). The foreign keys of the child table(“w1750131_Consultant_Event”) reference the primary keys of the parent tables(“w1750131_Event” and “w1750131_Consultant”). The multiplicities are 1..1 on the side of the parents, 0...* and 1...* on the side of the children.
6. I applied rule number 5 for the relationship that takes place between “w1750131_Job_Fair” (note that the relationship is transferred to w1750131_Event after I applied rule number 8 for the generalization of “w1750131_Event”) and “w1750131_Job”(note that the relationship is transferred to w1750131_JobOffer after I applied rule number 10 for the generalization of “w1750131_Offer”) as it is a many to many type of relationship. Applying this rule results in a child entity “w1750131_JobOffer_Event” that has a compound primary key(as the combination of the primary keys that belong to the parents cannot be repeated). The foreign keys of the child table(“w1750131_JobFair_Event”) reference the primary keys of the parent tables(“w1750131_Event” and “w1750131_JobOffer”). The multiplicities are 1..1 on the side of the parents, 0...* and 1...* on the side of the children.
7. I applied rule number 5 for the relationship that takes place between “w1750131_Staff” and “w1750131_Event” as it is a many to many type of relationship. Applying this rule results in a child entity “w1750131_Staff_Event” that has a compound primary key(as the combination of the primary keys that belong to the parents cannot be repeated). The foreign keys of the child table(“w1750131_Staff_Event”) reference the primary keys of the parent tables(“w1750131_Event” and “w1750131_Staff”). The multiplicities are 1..1 on the side of the parents and 0...* on the side of the children.

8. I applied rule number 3 for the relationship that takes place between "w1750131_Application" and "w1750131_Support_Session" as it is a one-to-one, mandatory on one side type of relationship. Applying this rule results into creating a foreign key with the same name as the primary key of the table "w1750131_Application" in the table "w1750131_SupportSession" (as it is the child table because it is on the "optional" participation side) that references the primary key of the parent table.
9. I applied rule number 1 for the relationship that takes place between "w1750131_Application" and "w1750131_Interview" as it is a one-to-many type of relationship. The parent is "w1750131_Application"(as it is on the "one" side (cardinality)) and the child is "w1750131_Interview"(as it is on the "many" side(cardinality)). I created a foreign key in the child table that copied the primary key of the foreign table and references it.
10. I applied rule number 1 for the relationship that takes place between "w1750131_Application" and "w1750131_Student" as it is a one-to-many type of relationship. The parent is "w1750131_Student"(as it is on the "one" side (cardinality)) and the child is "w1750131_Application"(as it is on the "many" side(cardinality)). I created a foreign key in the child table that copied the primary key of the foreign table and references it.
11. I applied rule number 1 for the relationship that takes place between "w1750131_Company" and "w1750131_Staff" as it is a one-to-many type of relationship. The parent is "w1750131_Company"(as it is on the "one" side (cardinality)) and the child is "w1750131_Staff"(as it is on the "many" side(cardinality)). I created a foreign key in the child table that copied the primary key of the foreign table and references it.
12. I applied rule number 1 for the relationship that takes place between "w1750131_Application" and "w1750131_Offer" as it is a one-to-many type of relationship. The parent is "w1750131_Offer"(as it is on the "one" side (cardinality)) and the child is "w1750131_Application"(as it is on the "many" side(cardinality)). I created a foreign key in the child table that copied the primary key of the foreign table and references it.
13. I applied rule number 1 for the relationship that takes place between "w1750131_GradScheme"(note that after applying rule number 10 for the generalization of "w1750131_Offer", "w1750131_GradScheme" changed into "w1750131_GradSchemeOffer") and "w1750131_Presentation"(note that after applying rule number 8 for the generalisation of "w1750131_Event", now the super-entity will have the relationship with "w1750131_GradScheme") as it is a one-to-many type of relationship. The parent is "w1750131_GradSchemeOffer"(as it is on the "one" side (cardinality)) and the child is "w1750131_Event"(as it is on the "many" side(cardinality)). I created a foreign key in the child table that copied the primary key of the foreign table and references it.
14. I applied rule number 1 for the relationship that takes place between "w1750131_Company" and "w1750131_Offer" as it is a one-to-many type of relationship. The parent is "w1750131_Company"(as it is on the "one" side (cardinality)) and the child is "w1750131_Offer"(as it is on the "many" side(cardinality)). I created a foreign key in the child table that copied the primary key of the foreign table and references it.
15. I applied rule number 1 for the relationship that takes place between "w1750131_Company" and "w1750131_Offer" as it is a one-to-many type of relationship. The parent is "w1750131_Company"(as it is on the "one" side (cardinality)) and the child is

"w1750131_Offer"(as it is on the "many" side(cardinality)). I created a foreign key in the child table that copied the primary key of the foreign table and references it.

16. I applied rule number 1 for the relationship that takes place between "w1750131_Consultant" and "w1750131_Support_Session" as it is a one-to-many type of relationship. The parent is "w1750131_Consultant"(as it is on the "one" side (cardinality)) and the child is "w1750131_Support_Session"(as it is on the "many" side(cardinality)). I created a foreign key in the child table that copied the primary key of the foreign table and references it.
17. I applied rule number 1 for the relationship that takes place between "w1750131_Consultant" and "w1750131_Offer" as it is a one-to-many type of relationship. The parent is "w1750131_Consultant"(as it is on the "one" side (cardinality)) and the child is "w1750131_Offer"(as it is on the "many" side(cardinality)). I created a foreign key in the child table that copied the primary key of the foreign table and references it.
18. I applied rule number 1 for the recursive relationship that "w1750131_Consultant" has. I created a foreign key named "w1750131_mentorID" which will reference the primary key "w1750131_consNo".

SQL code for creating the three tables and the screenshots for each of them:

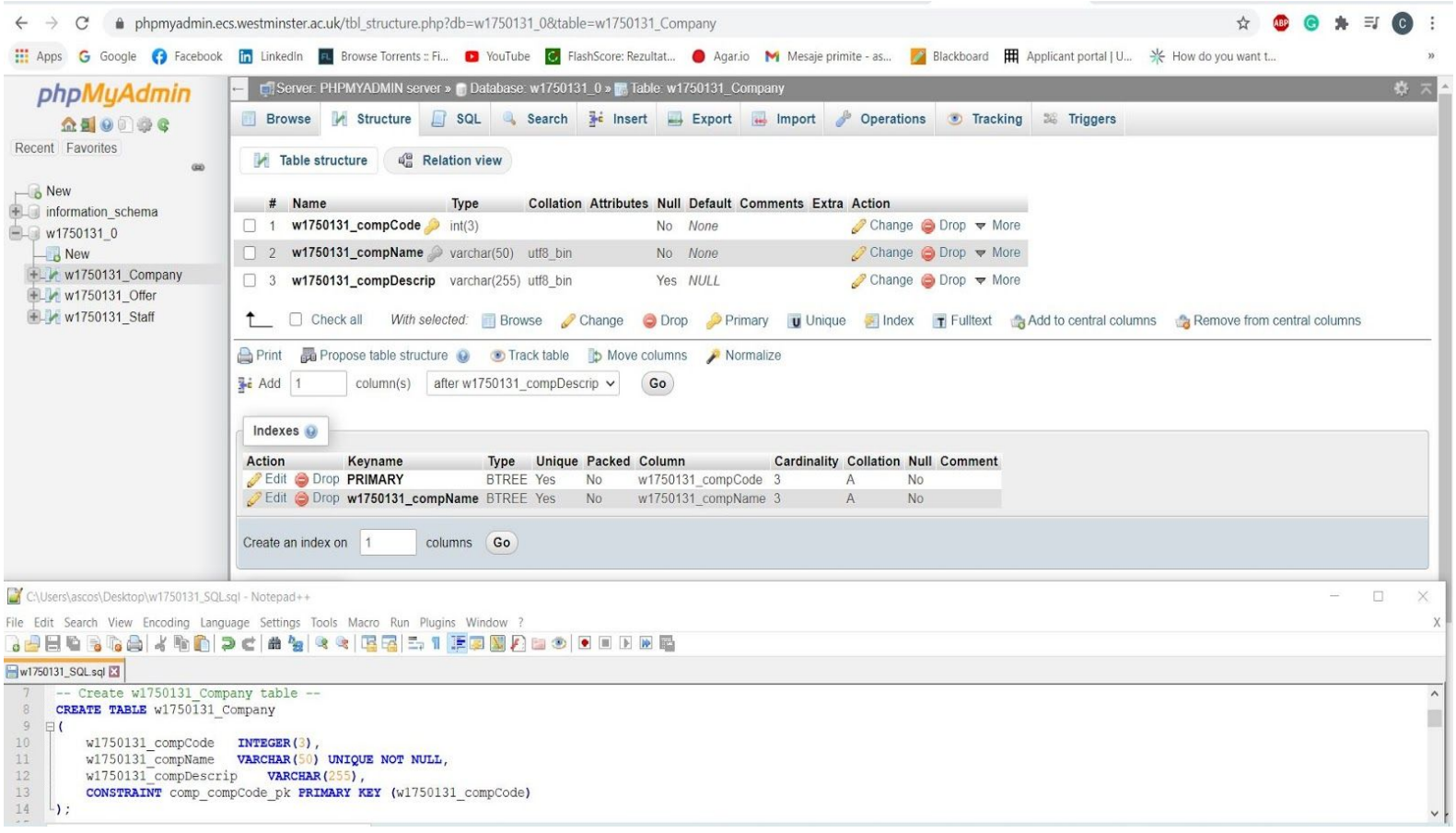
```
-- Create w1750131_Company table --
CREATE TABLE w1750131_Company
(
	w1750131_compCode   INTEGER(3),
	w1750131_compName  VARCHAR(50) UNIQUE NOT NULL,
	w1750131_compDescrip	VARCHAR(255),
	CONSTRAINT comp_compCode_pk PRIMARY KEY (w1750131_compCode)
);

-- Create w1750131_Staff table --
CREATE TABLE w1750131_Staff
(
	w1750131_staffID	INTEGER(3),
	w1750131_staffFullName	VARCHAR(50) NOT NULL,
	w1750131_staffPosition	VARCHAR(50) NOT NULL,
	w1750131_staffEmail	VARCHAR(50) UNIQUE NOT NULL,
	w1750131_compCode   INTEGER(3) NOT NULL,
	CONSTRAINT st_stid_pk PRIMARY KEY (w1750131_staffID),
	CONSTRAINT st_compCode_fk FOREIGN KEY (w1750131_compCode)
	REFERENCES w1750131_Company(w1750131_compCode)
	ON DELETE CASCADE
	ON UPDATE CASCADE
);

-- Create w1750131_Offer table --
CREATE TABLE w1750131_Offer
(
```

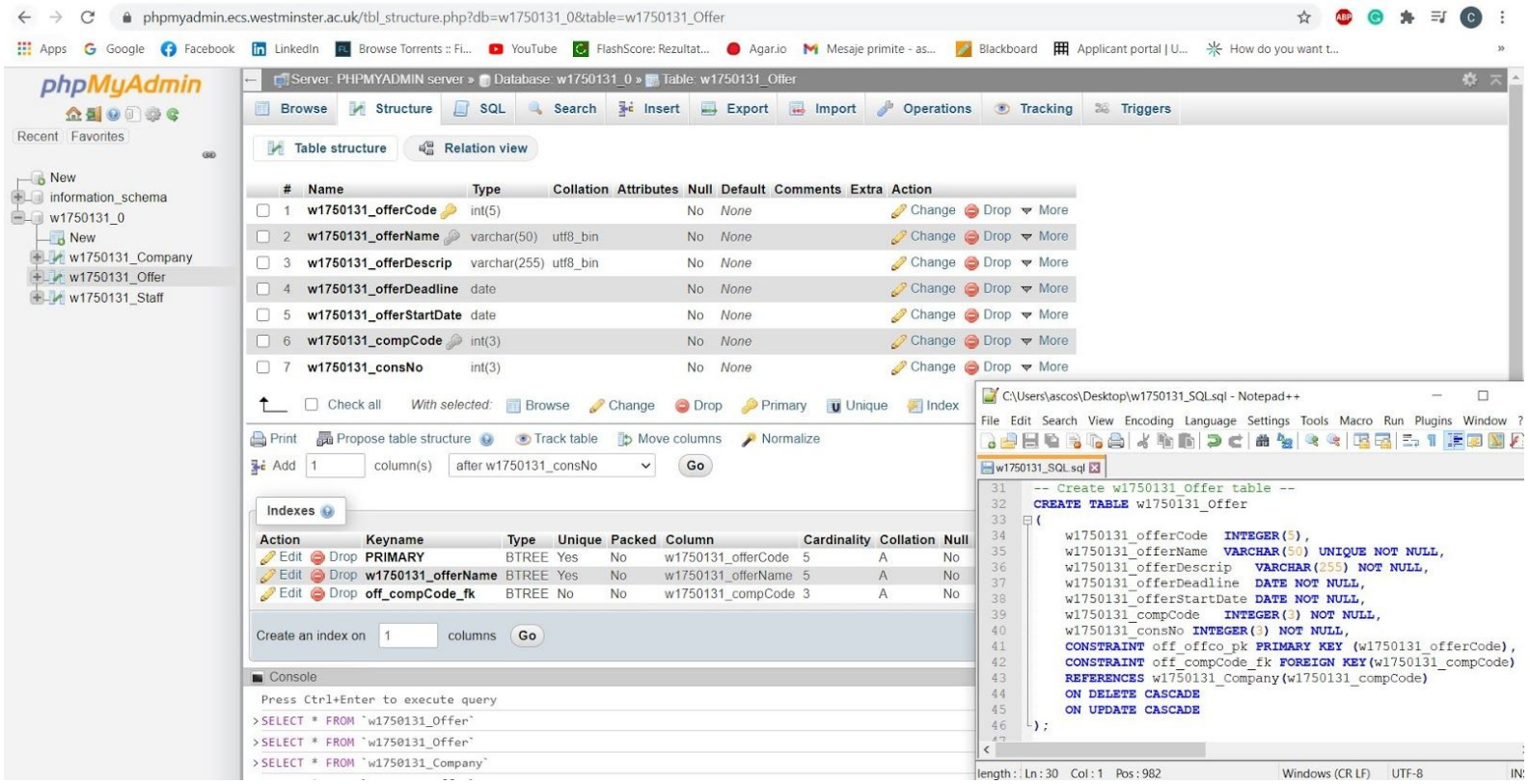
Student name: Cosmin Alexandru Stefanescu
Student ID: w1750131
Tutorial group attended: Every Tuesday 2:00 PM to 4:00 PM 5CS09

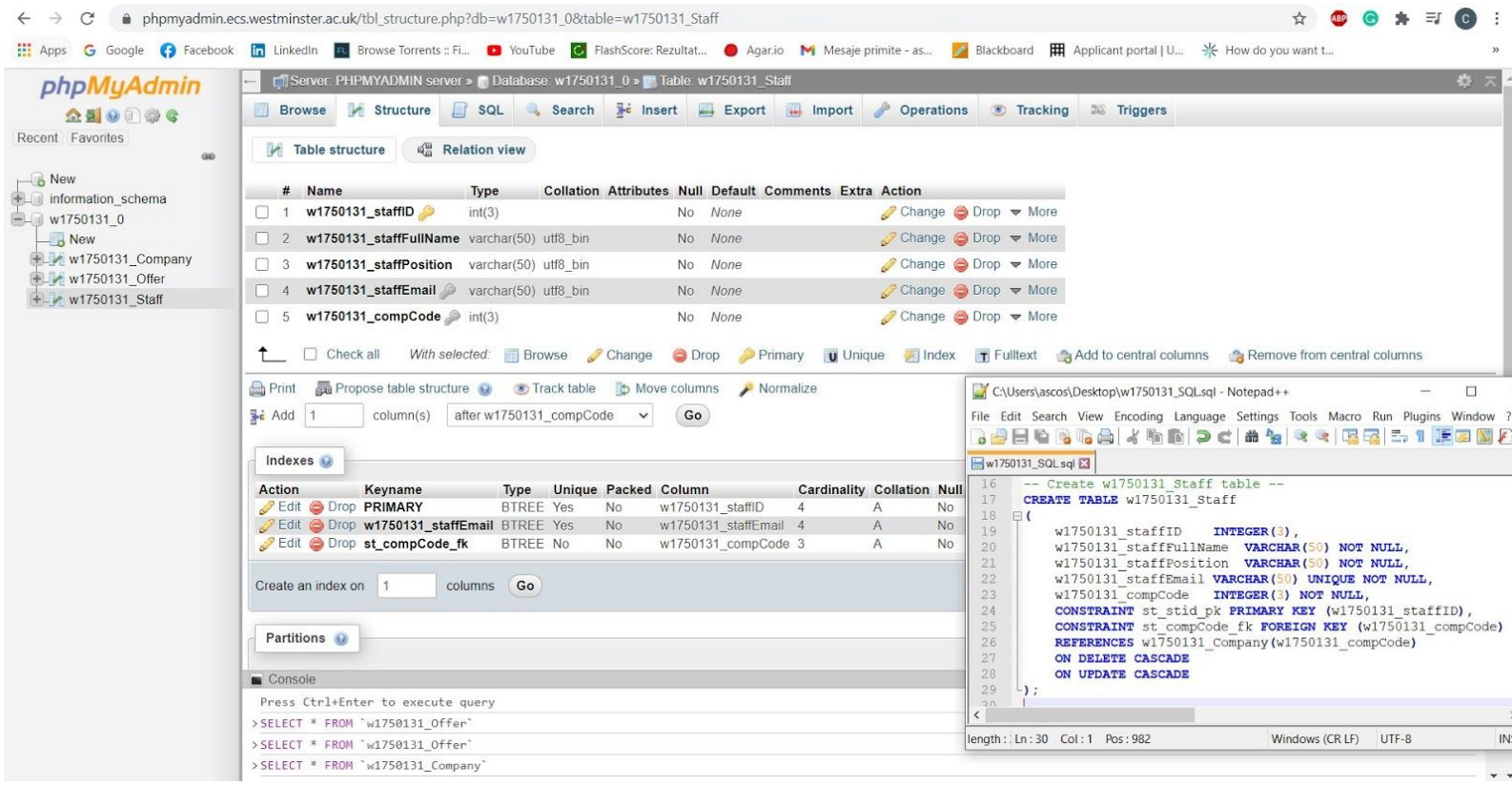
```
w1750131_offerCode    INTEGER(5),
w1750131_offerName    VARCHAR(50) UNIQUE NOT NULL,
w1750131_offerDescripVARCHAR(255) NOT NULL,
w1750131_offerDeadline          DATE NOT NULL,
w1750131_offerStartDate         DATE NOT NULL,
w1750131_compCode    INTEGER(3) NOT NULL,
w1750131_consNo INTEGER(3) NOT NULL,
CONSTRAINT off_offco_pk PRIMARY KEY (w1750131_offerCode),
CONSTRAINT off_compCode_fk FOREIGN KEY(w1750131_compCode)
REFERENCES w1750131_Company(w1750131_compCode)
ON DELETE CASCADE
ON UPDATE CASCADE);
```



Module leader: Francois ROUBERT
Tutorial teacher: Tasos Ptohos

Student name: Cosmin Alexandru Stefanescu
Student ID: w1750131
Tutorial group attended: Every Tuesday 2:00 PM to 4:00 PM 5CS09





SQL code for inserting the table contents and the required screenshots

-- Populating the w1750131_Company table --

Module leader: Francois ROUBERT
Tutorial teacher: Tasos Ptohos

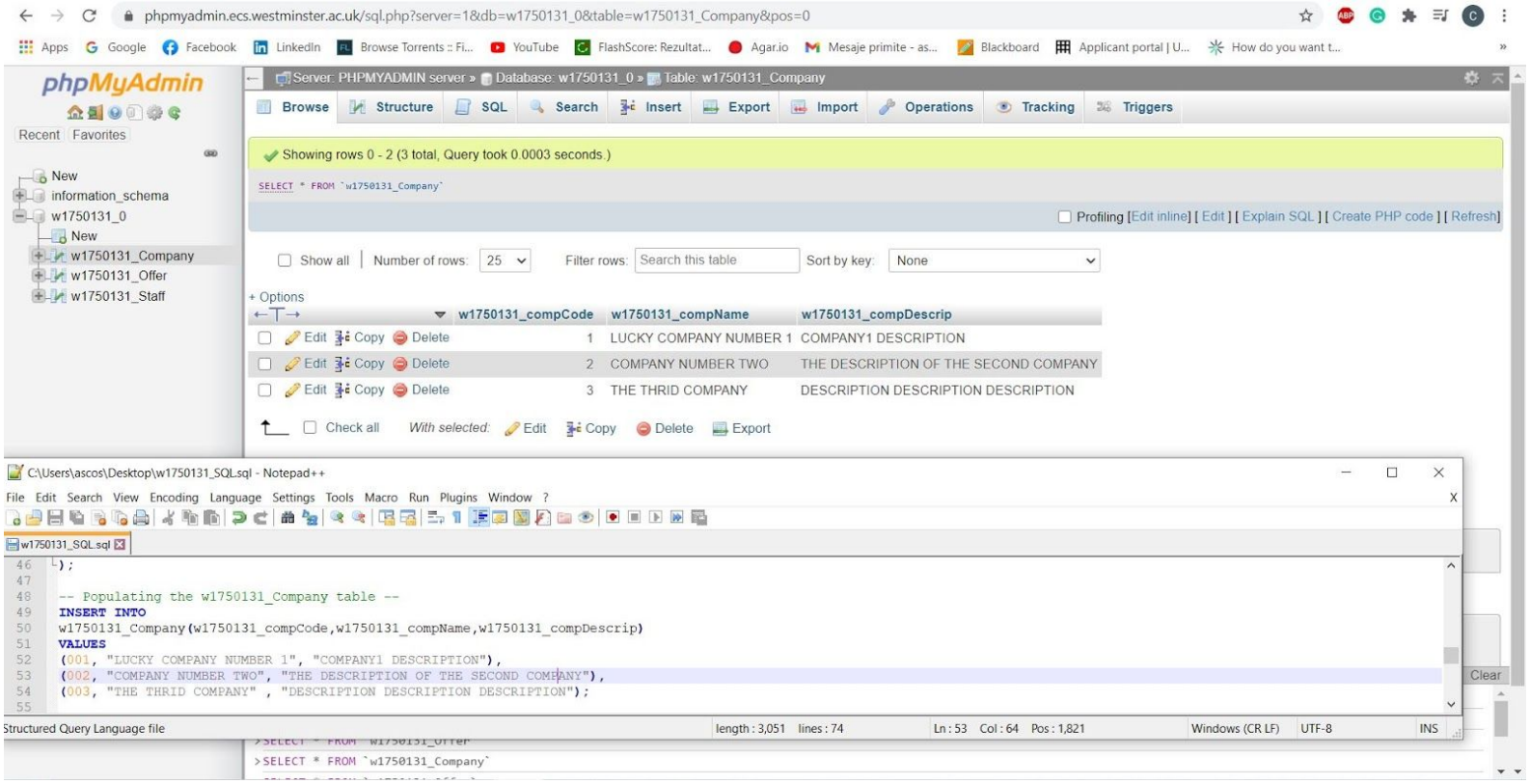
Student name: Cosmin Alexandru Stefanescu
Student ID: w1750131
Tutorial group attended: Every Tuesday 2:00 PM to 4:00 PM 5CS09

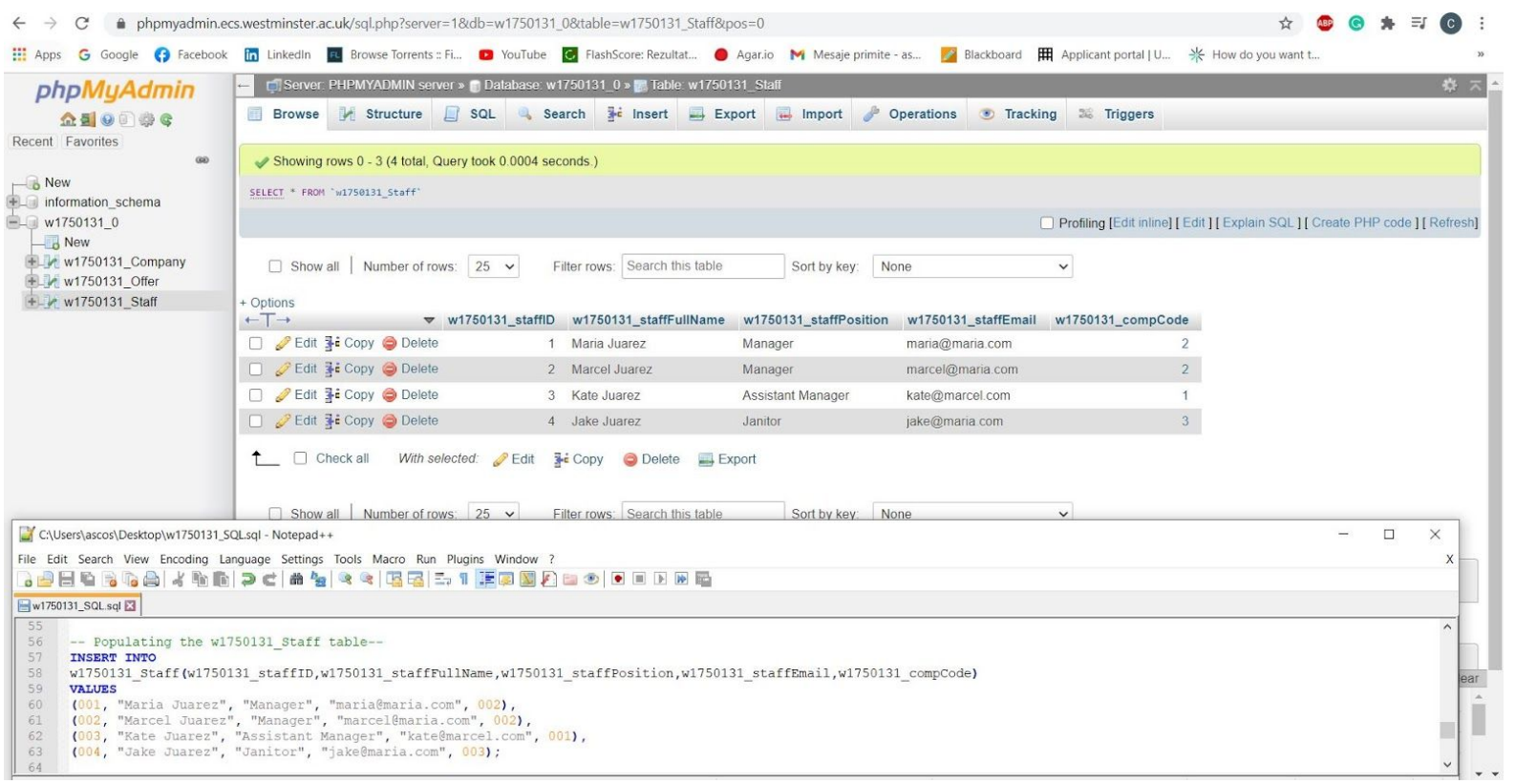
```
INSERT INTO
w1750131_Company(w1750131_compCode,w1750131_compName,w1750131_compDescrip)
VALUES
(001, "LUCKY COMPANY NUMBER 1", "COMPANY1 DESCRIPTION"),
(002, "COMPANY NUMBER TWO", "THE DESCRIPTION OF THE SECOND COMPANY"),
(003, "THE THRID COMPANY" , "DESCRIPTION DESCRIPTION DESCRIPTION");

-- Populating the w1750131_Staff table--
INSERT INTO
w1750131_Staff(w1750131_staffID,w1750131_staffFullName,w1750131_staffPosition,w1750131_st
affEmail,w1750131_compCode)
VALUES
(001, "Maria Juarez", "Manager", "maria@maria.com", 002),
(002, "Marcel Juarez", "Manager", "marcel@maria.com", 002),
(003, "Kate Juarez", "Assistant Manager", "kate@marcel.com", 001),
(004, "Jake Juarez", "Janitor", "jake@maria.com", 003);

-- Populating the w1750131_Offer table --
INSERT INTO
w1750131_Offer(w1750131_offerCode,w1750131_offerName,w1750131_offerDescrip,w1750131_o
fferDeadline,w1750131_offerStartDate,w1750131_compCode,w1750131_consNo)
VALUES
(001, "Janitor required", "We need an experienced janitor", "2011-01-21", "2011-01-22", 003, 001),
(002, "Manager required", "We need an easy going manager", "2014-05-10", "2014-05-11", 002,
002),
(003, "Junior Software Developer", "Graduate stundents required", "2020-12-23", "2020-12-25",
001, 002),
(004, "Senior Software Developer", " We need bright minds", "2020-12-23", "2020-12-24", 001, 003),
(005, "Hired assasin required", "We need silent ninjas", "2001-01-01", "2001-01-03", 001, 003);
```

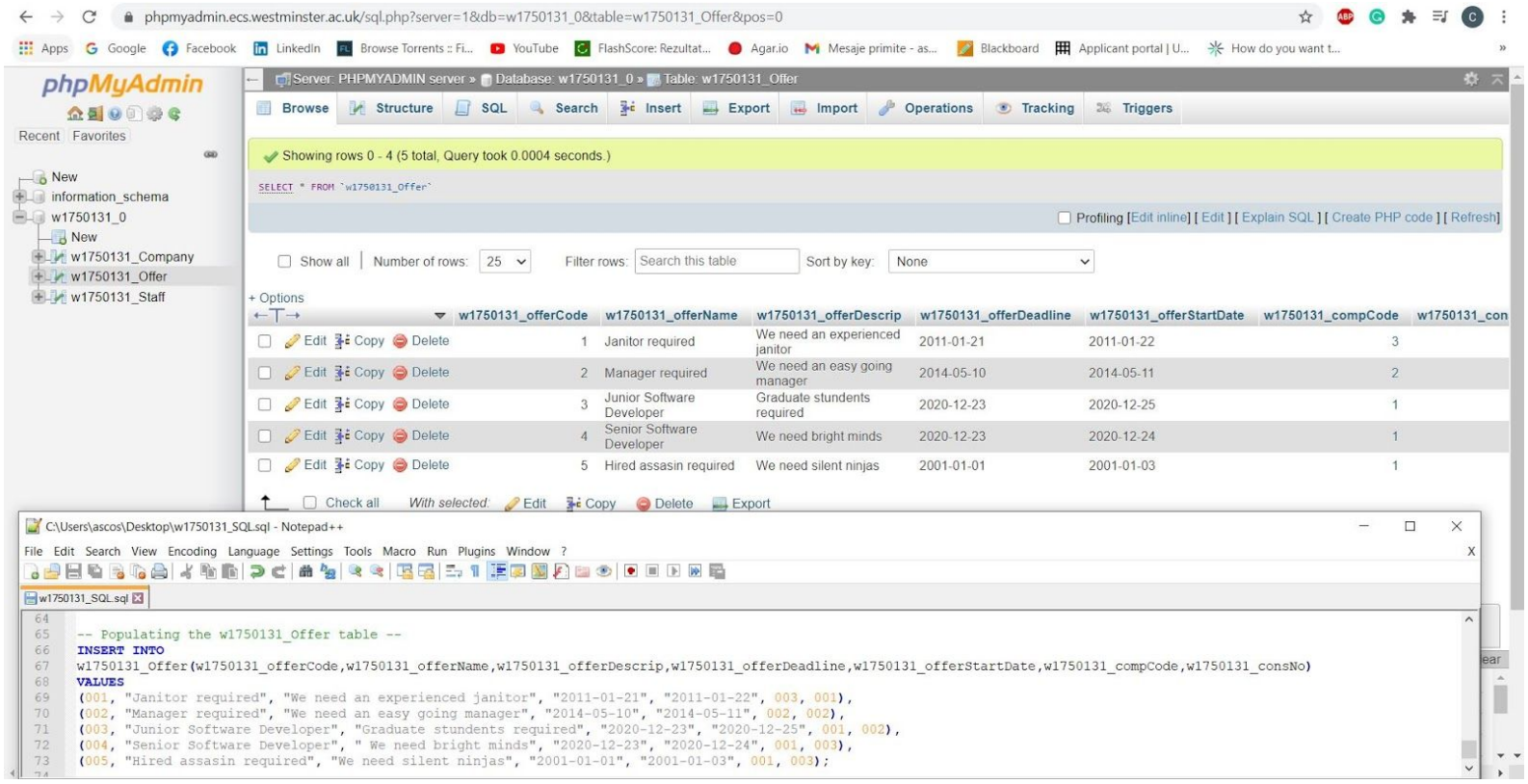
Student name: Cosmin Alexandru Stefanescu
Student ID: w1750131
Tutorial group attended: Every Tuesday 2:00 PM to 4:00 PM 5CS09





Module leader: Francois ROUBERT
Tutorial teacher: Tasos Ptohos

Student name: Cosmin Alexandru Stefanescu
Student ID: w1750131
Tutorial group attended: Every Tuesday 2:00 PM to 4:00 PM 5CS09

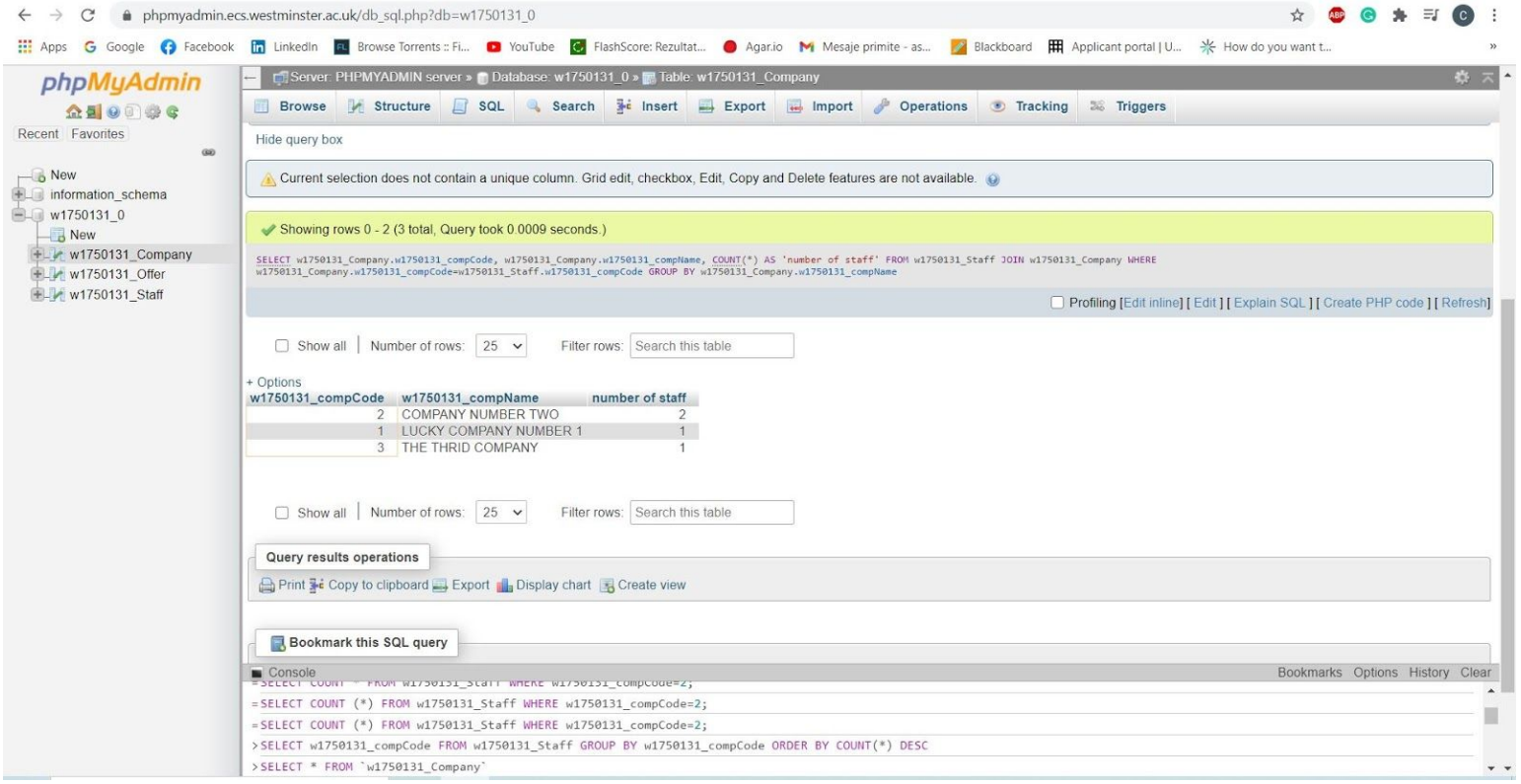


SQL query to display company code, company name and number of employees for each company

SELECT w1750131_Company.w1750131_compCode, w1750131_Company.w1750131_compName, COUNT(*) AS 'number of staff' FROM w1750131_Staff JOIN w1750131_Company WHERE w1750131_Company.w1750131_compCode=w1750131_Staff.w1750131_compCode GROUP BY w1750131_Company.w1750131_compName ;

Module leader: Francois ROUBERT
Tutorial teacher: Tasos Ptohos

Student name: Cosmin Alexandru Stefanescu
Student ID: w1750131
Tutorial group attended: Every Tuesday 2:00 PM to 4:00 PM 5CS09

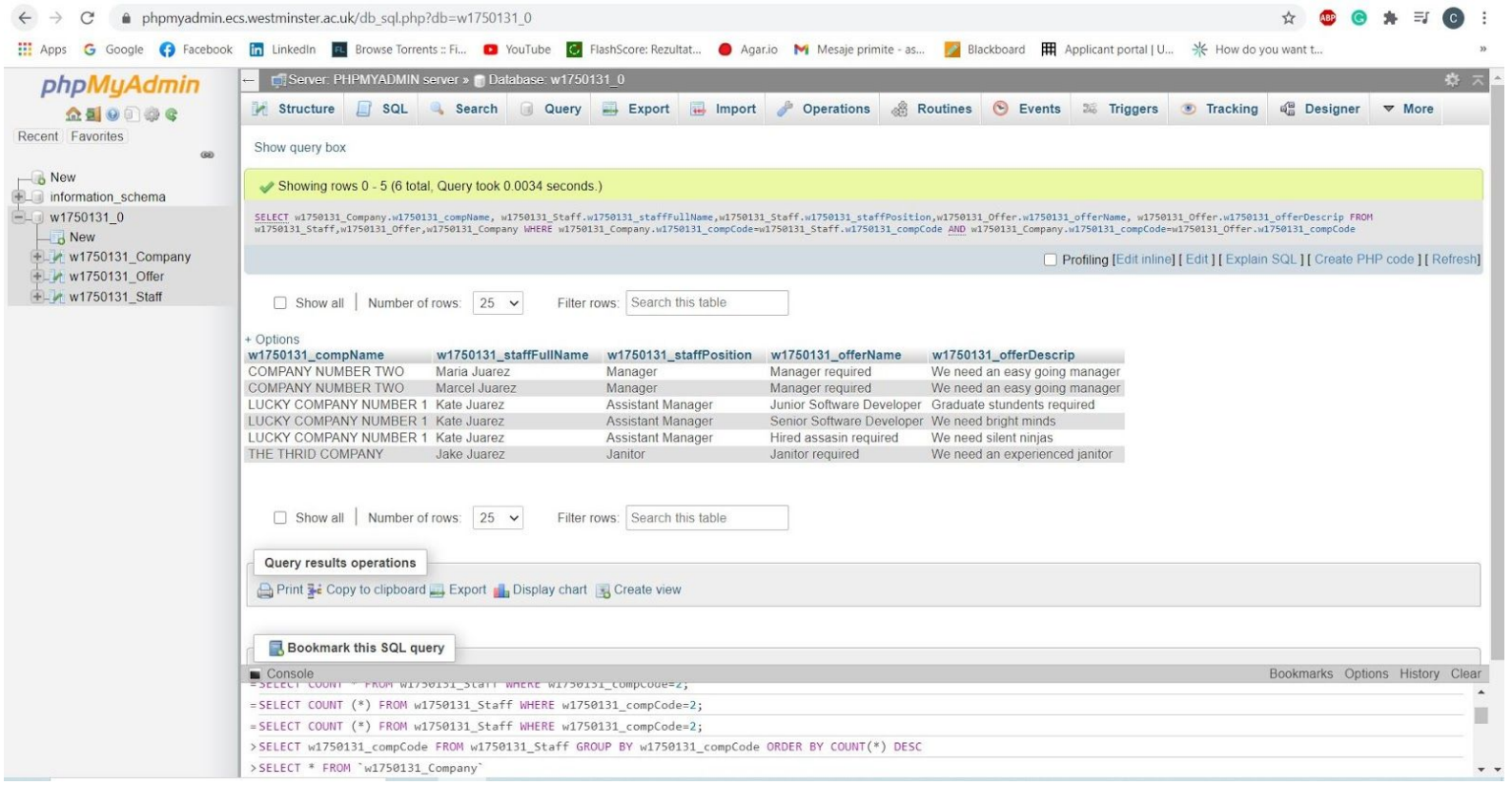


SQL query that displays a list of company names along the names and positions of staff they employ and the names and descriptions of the offers they propose

SELECT w1750131_Company.w1750131_compName,
w1750131_Staff.w1750131_staffFullName,w1750131_Staff.w1750131_staffPosition,w1750131_Offer.w1750131_offerName, w1750131_Offer.w1750131_offerDescrip
FROM w1750131_Staff,w1750131_Offer,w1750131_Company
WHERE w1750131_Company.w1750131_compCode=w1750131_Staff.w1750131_compCode
AND w1750131_Company.w1750131_compCode=w1750131_Offer.w1750131_compCode;

Module leader: Francois ROUBERT
Tutorial teacher: Tasos Ptohos

Student name: Cosmin Alexandru Stefanescu
Student ID: w1750131
Tutorial group attended: Every Tuesday 2:00 PM to 4:00 PM 5CS09



Module leader: Francois ROUBERT
Tutorial teacher: Tasos Ptohos