Mirna Arivalagan - 220142881

Database & Information Retrieval – SIT772

Assignment 1B

Trimester 3 2021

Question 1 – Home Loan

Home Loan Assumptions & Business Rules

Assumptions:

- Average sold price is calculated from sold properties in the last 10 years that exists in the bank's database.
- Bank account can only be held by 1 customer and for join home loan application, 2 account numbers are captured in the table.
- The brief did not mention any particular suburb that Tom and Anna mentioned, so I've just picked as Ringwood to demonstrate

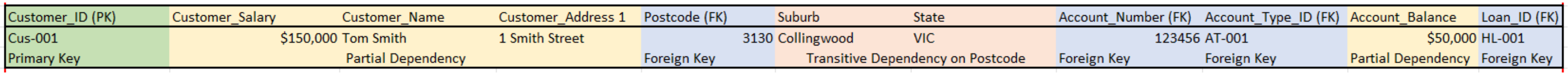
Business Rules:

- Customer needs to have a bank account to be a customer of the bank
- Only 1 customer can be linked to an account
- Multiple customers can apply for a home loan
- A loan application bridging entity was included to avoid many to many relationships and an application id is unique and attached to each home loan
- Only 1 staff can be assigned to manage the home loan and a staff may not be assigned a home loan to manage. The staff entity covers all staff, not just home loan.
- More than 1 customer can own a property, and each customer can own more than 1 property.
- An Owner property entity was designed as a bridging entity to avoid many to many relationships.

Home Loan Normalization

1.1) Customer Entity Schema

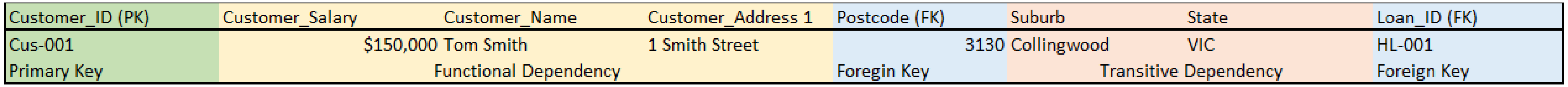
Not Normalized



Customer_ID (PK)	Customer_Salary	Customer_Name	Customer_Address 1	Postcode (FK)	Suburb	State	Account_Number (FK)	Account_Type_ID (FK)	Account_Balance	Loan_ID (FK)
Cus-001	$150,000	Tom Smith	1 Smith Street	3130	Collingwood	VIC	123456	AT-001	$50,000	HL-001
Primary Key		Partial Dependency		Foreign Key	Transitive Dependency on Postcode		Foreign Key	Foreign Key	Partial Dependency	Foreign Key

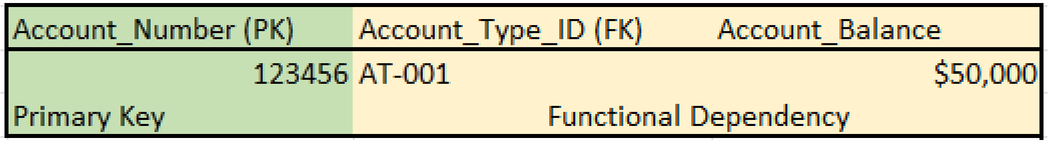
2nd Normal Form :

Table 1 (Customer Info):



Customer_ID (PK)	Customer_Salary	Customer_Name	Customer_Address 1	Postcode (FK)	Suburb	State	Loan_ID (FK)
Cus-001	$150,000	Tom Smith	1 Smith Street	3130	Collingwood	VIC	HL-001
Primary Key		Functional Dependency		Foregin Key	Transitive Dependency		Foreign Key

Table 2 (Account Info):

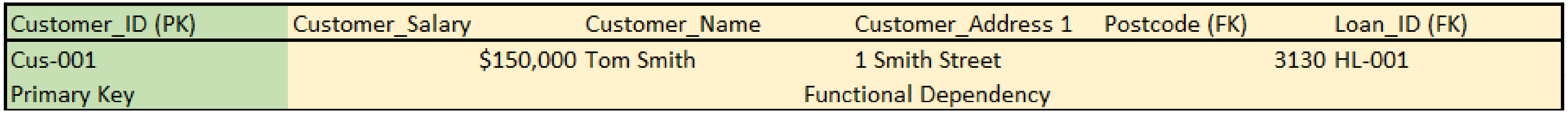


Account_Number (PK)	Account_Type_ID (FK)	Account_Balance
123456	AT-001	$50,000
Primary Key	Functional Dependency	

3rd Normal Form:

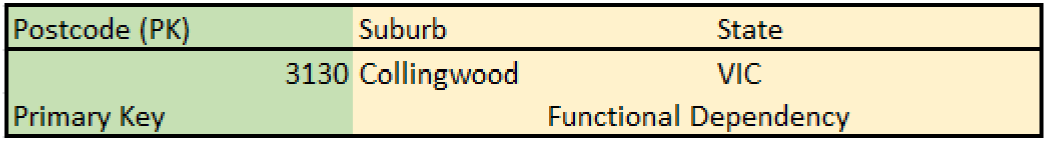
The transitive dependencies identified were with the postcode, suburb and state, where with the postcode, you can identify the suburb and state. So only the customer info table will be broken down further.

Table 1 (Customer Info):



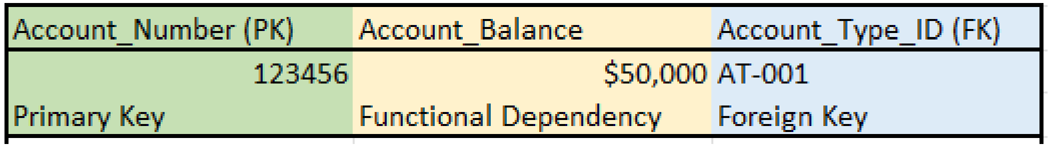
Customer_ID (PK)	Customer_Salary	Customer_Name	Customer_Address 1	Postcode (FK)	Loan_ID (FK)
Cus-001	$150,000	Tom Smith	1 Smith Street	3130	HL-001
Primary Key	Functional Dependency				

Table 2 (Postcode):



Postcode (PK)	Suburb	State
3130	Collingwood	VIC
Primary Key	Functional Dependency	

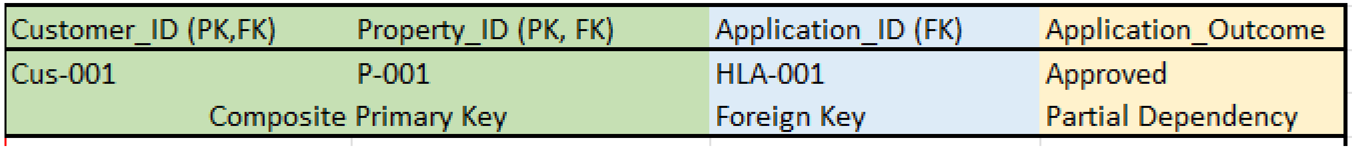
Table 3 (Account Info):



Account_Number (PK)	Account_Balance	Account_Type_ID (FK)
123456	$50,000	AT-001
Primary Key	Functional Dependency	Foreign Key

1.2) Loan Application Entity Schema

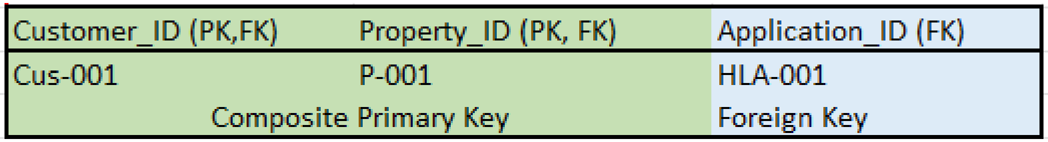
Not Normalized



Customer_ID (PK,FK)	Property_ID (PK, FK)	Application_ID (FK)	Application_Outcome
Cus-001	P-001	HLA-001	Approved
Composite Primary Key		Foreign Key	Partial Dependency

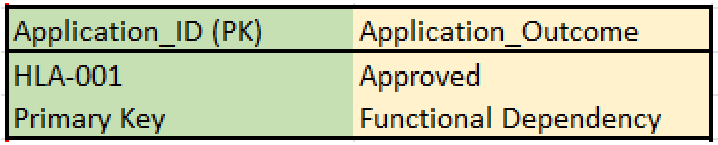
2nd Normal Form:

Table 1 (Loan Application):



Customer_ID (PK,FK)	Property_ID (PK, FK)	Application_ID (FK)
Cus-001	P-001	HLA-001
Composite Primary Key		Foreign Key

Table 2 (Application Outcome):

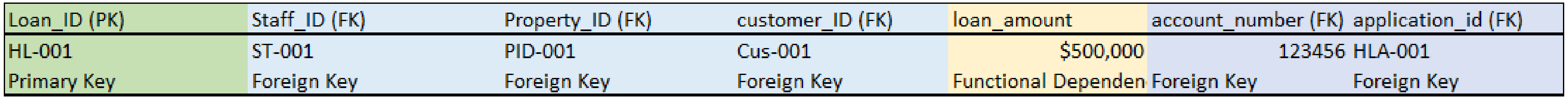


Application_ID (PK)	Application_Outcome
HLA-001	Approved
Primary Key	Functional Dependency

3rd Normal Form: There are no transitive dependencies, so these tables are already normalized to 3NF.

1.3) Home Loan Entity Schema

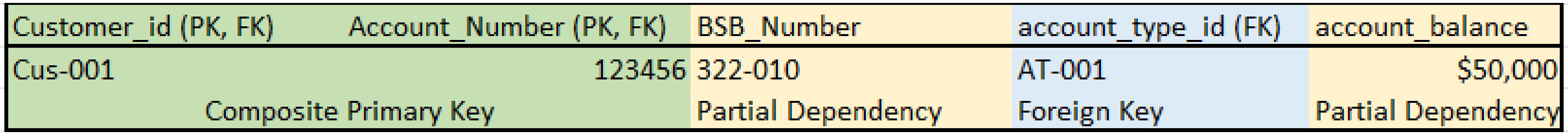
This table is already normalized to 2NF and 3NF as there are no partial dependencies or transitive dependencies



Loan_ID (PK)	Staff_ID (FK)	Property_ID (FK)	customer_ID (FK)	loan_amount	account_number (FK)	application_id (FK)
HL-001	ST-001	PID-001	Cus-001	$500,000	123456	HLA-001
Primary Key	Foreign Key	Foreign Key	Foreign Key	Functional Dependen	Foreign Key	Foreign Key

1.4) Bank Account Entity Schema

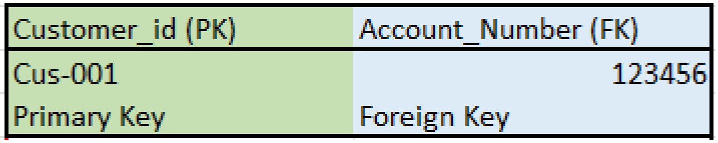
Not Normalized:



Customer_id (PK, FK)	Account_Number (PK, FK)	BSB_Number	account_type_id (FK)	account_balance
Cus-001	123456	322-010	AT-001	$50,000
Composite Primary Key		Partial Dependency	Foreign Key	Partial Dependency

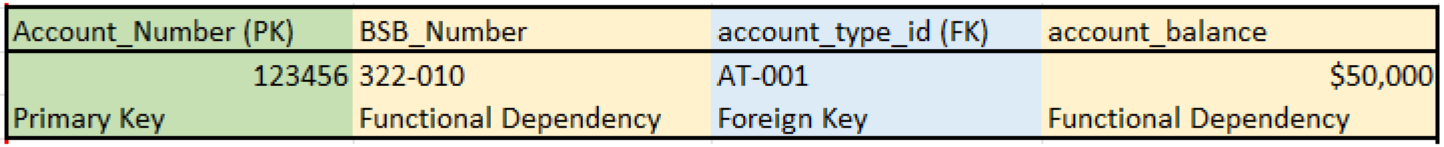
2nd Normal Form:

Table 1 (Account Number):



Customer_id (PK)	Account_Number (FK)
Cus-001	123456
Primary Key	Foreign Key

Table 2 (Account Balance):

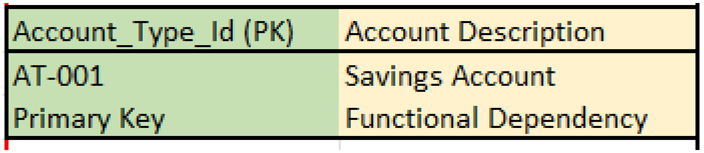


Account_Number (PK)	BSB_Number	account_type_id (FK)	account_balance
123456	322-010	AT-001	$50,000
Primary Key	Functional Dependency	Foreign Key	Functional Dependency

3rd Normal Form: These tables are already in 3NF as there are no transitive dependencies.

1.5) Account Type Entity Schema

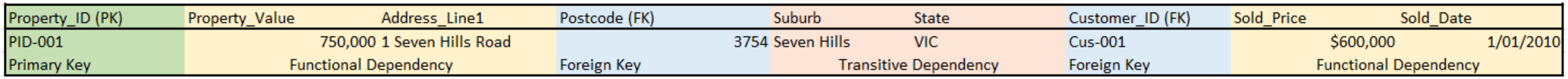
Table is already set up to normalize to 2nd and 3rd normal form. No further normalization required.



Account_Type_Id (PK)	Account Description
AT-001	Savings Account
Primary Key	Functional Dependency

1.6) Property Entity Schema

Not Normalized:

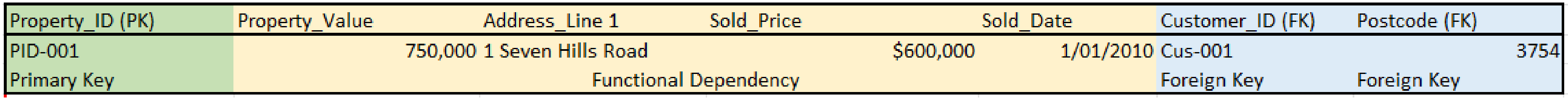


Property_ID (PK)	Property_Value	Address_Line1	Postcode (FK)	Suburb	State	Customer_ID (FK)	Sold_Price	Sold_Date
PID-001	750,000	1 Seven Hills Road	3754	Seven Hills	VIC	Cus-001	$600,000	1/01/2010
Primary Key	Functional Dependency		Foreign Key	Transitive Dependency		Foreign Key	Functional Dependency	

2nd Normal Form: Table is already in 2nd Normal Form as there are no partial dependencies, but will need to normalize to 3rd normal form as there is transitive dependencies present with the postcodes

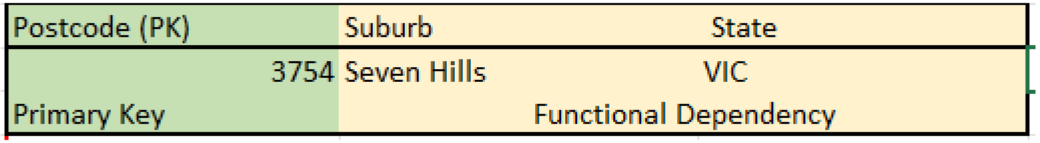
3rd Normal Form:

Table 1 (Property Sold Information):



Property_ID (PK)	Property_Value	Address_Line 1	Sold_Price	Sold_Date	Customer_ID (FK)	Postcode (FK)
PID-001	750,000	1 Seven Hills Road	$600,000	1/01/2010	Cus-001	3754
Primary Key		Functional Dependency			Foreign Key	Foreign Key

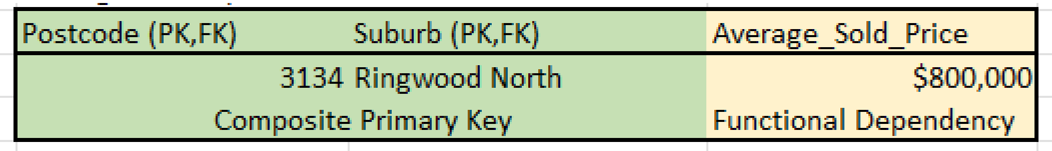
Table 2 (Postcode Table):



Postcode (PK)	Suburb	State
3754	Seven Hills	VIC
Primary Key	Functional Dependency	

1.7) Average Price Suburb Entity Schema

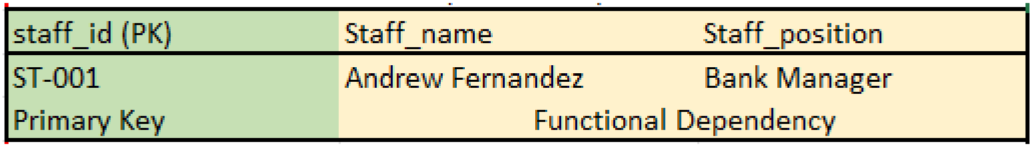
This table is already normalised to 2nd and 3rd normal form as there are no partial dependencies or transitive dependencies.



Postcode (PK,FK)	Suburb (PK,FK)	Average_Sold_Price
3134	Ringwood North	$800,000
Composite Primary Key		Functional Dependency

1.8) Staff Entity Schema

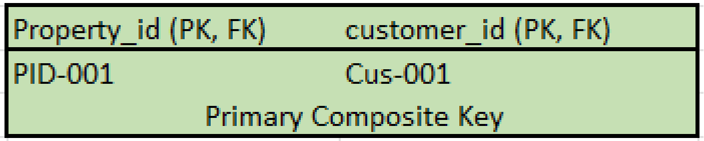
This table is also already normalized to 2nd and 3rd normal form as there are no partial and transitive dependencies.



staff_id (PK)	Staff_name	Staff_position
ST-001	Andrew Fernandez	Bank Manager
Primary Key	Functional Dependency	

1.9) Owner Property Entity Schema

As this is a bridging entity to stop the many to many relations, this table is already normalized to 2nd and 3rd normal forms.



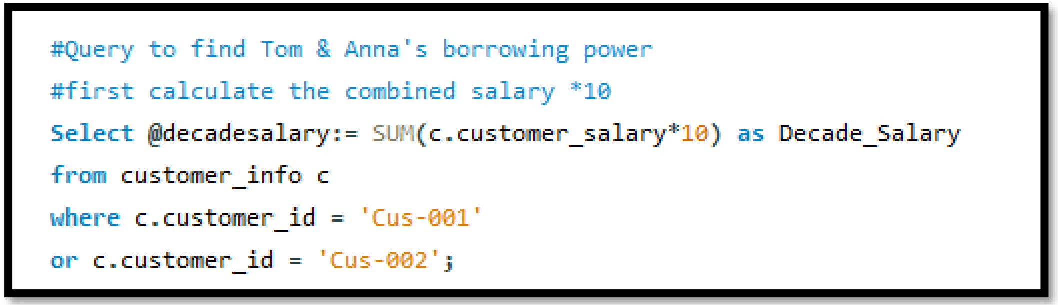
Property_id (PK, FK)	customer_id (PK, FK)
PID-001	Cus-001
Primary Composite Key	

Home Loan Query – Tom & Anna

To compute the total amount that Tom & Anna can borrow, I've decided to use the declared variables approach. There are 4 declared variables that needed to be created to work out how much Tom and Anna can borrow.

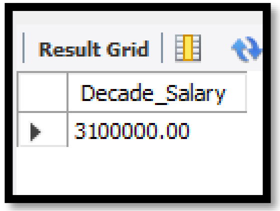
a. Calculate Tom and Anna's salary x 10 years - @decadesalary:

Query Logic



```
#Query to find Tom & Anna's borrowing power
#first calculate the combined salary *10
Select @decadesalary:= SUM(c.customer_salary*10) as Decade_Salary
from customer_info c
where c.customer_id = 'Cus-001'
or c.customer_id = 'Cus-002';
```

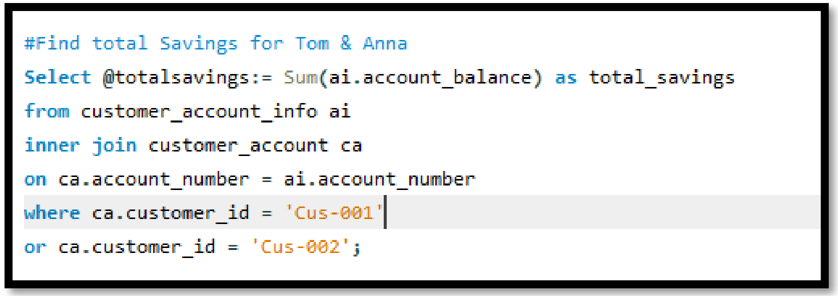
Query Results



```
Decade_Salary
3100000.00
```

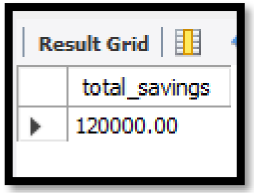
b. Calculate total savings for Tom & Anna - @totalsavings:

Query Logic



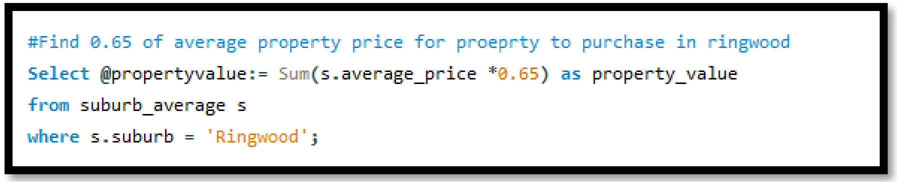
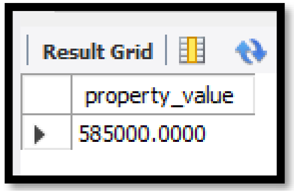
```
#Find total Savings for Tom & Anna
Select @totalsavings:= Sum(ai.account_balance) as total_savings
from customer_account_info ai
inner join customer_account ca
on ca.account_number = ai.account_number
where ca.customer_id = 'Cus-001'
or ca.customer_id = 'Cus-002';
```

Query Results

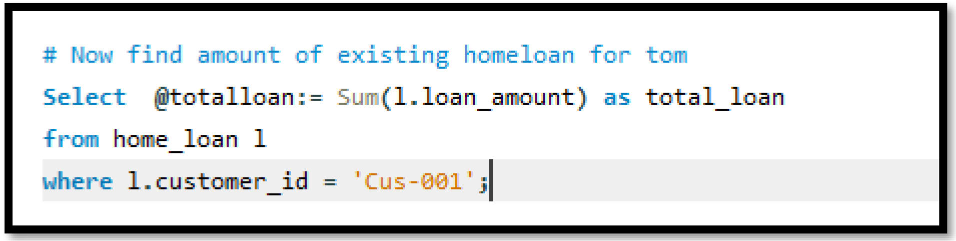
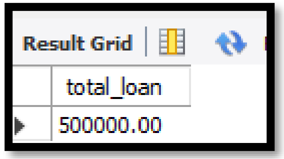


```
total_savings
120000.00
```

c. Calculate 65% of the average property value - @property value:

Query Logic	Query Results
 #Find 0.65 of average property price for proeprty to purchase in ringwood Select @propertyvalue:= Sum(s.average_price *0.65) as property_value from suburb_average s where s.suburb = 'Ringwood';	 property_value 585000.0000

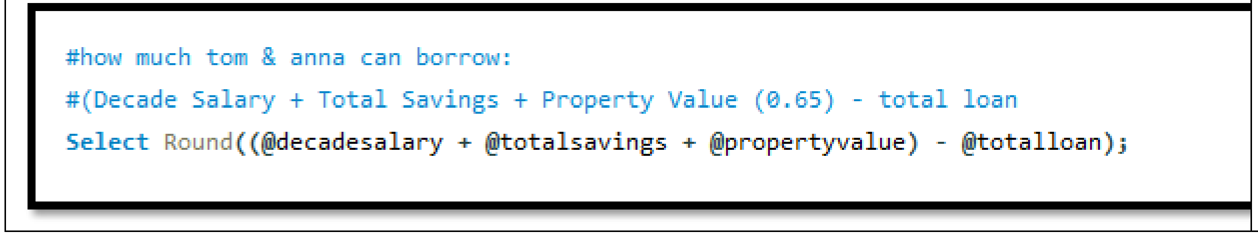
d. Calculate existing home loan amount for Tom - @totalloan:

Query Logic	Query Results
 # Now find amount of existing homeloan for tom Select @totalloan:= Sum(l.loan_amount) as total_loan from home_loan l where l.customer_id = 'Cus-001';	 total_loan 500000.00

Final Calculation:

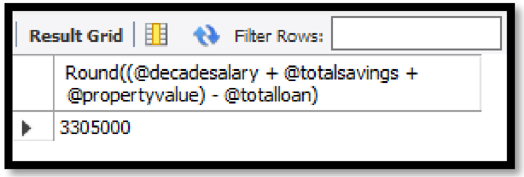
Once the declared variables have been created, we can use them to calculate the final amount they can borrow:

Query Logic



```
#how much tom & anna can borrow:
#(Decade Salary + Total Savings + Property Value (0.65) - total loan
Select Round((@decadesalary + @totalsavings + @propertyvalue) - @totalloan);
```

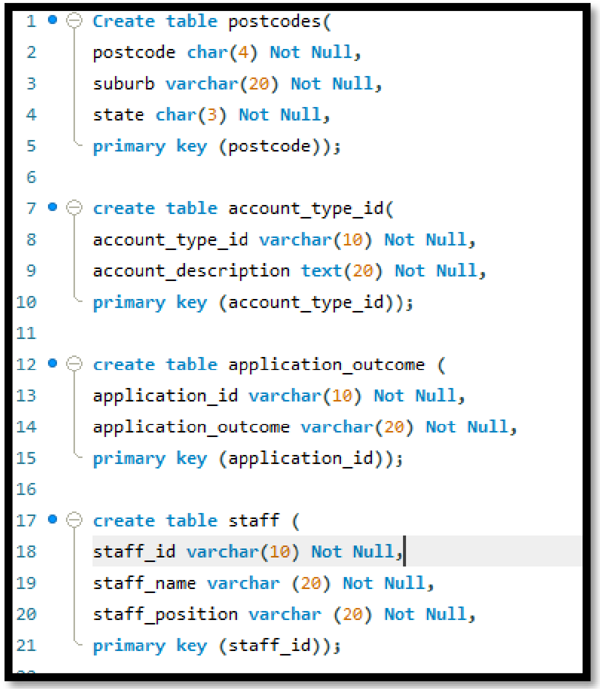
Query Results



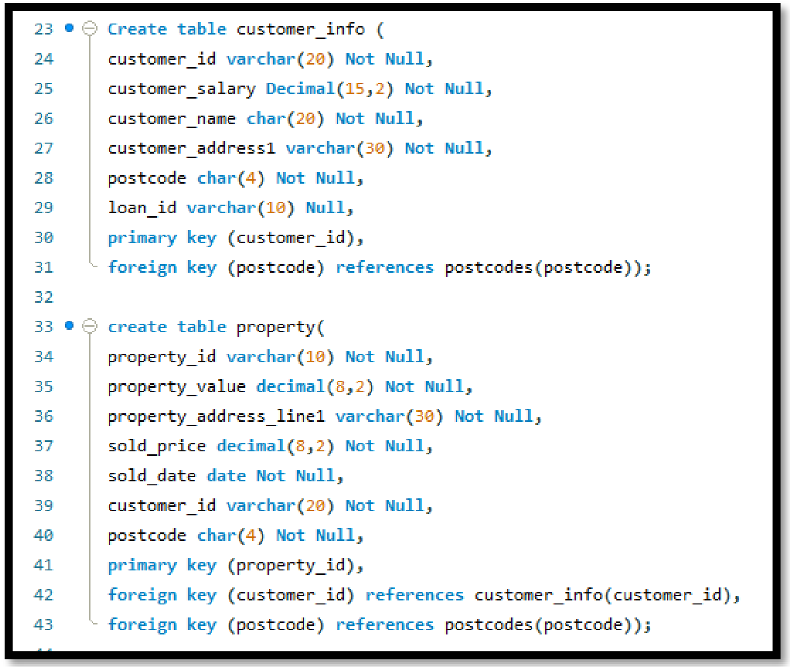
```
Round((@decadesalary + @totalsavings +
@propertyvalue) - @totalloan)
3305000
```

So based on the dummy data and calculation that I have done, the total amount Tom and Anna can borrow is: **$3,305,000.00**

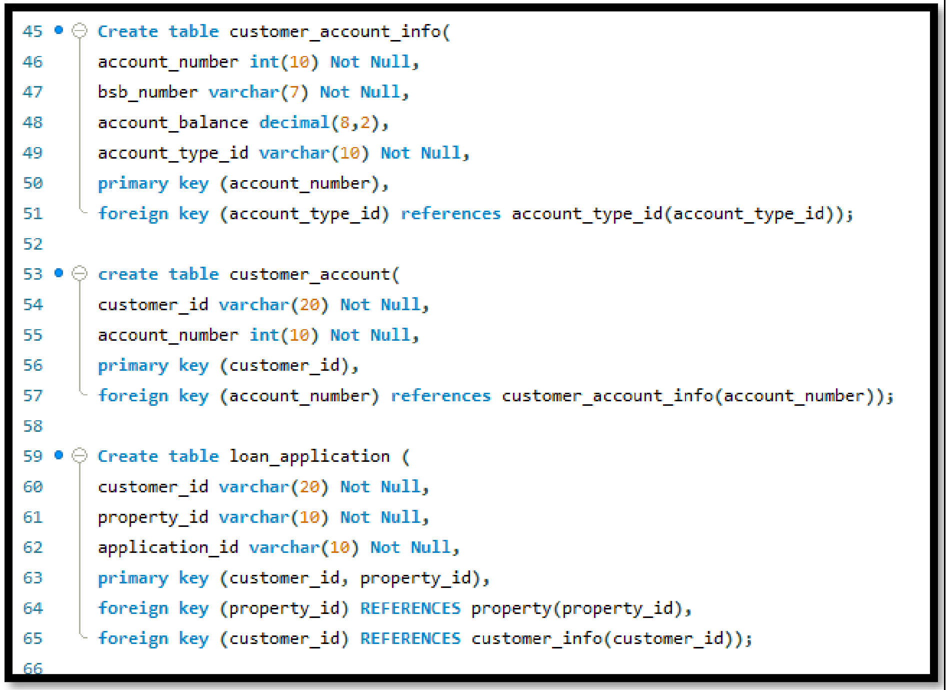
Home Loan Query – Create Tables



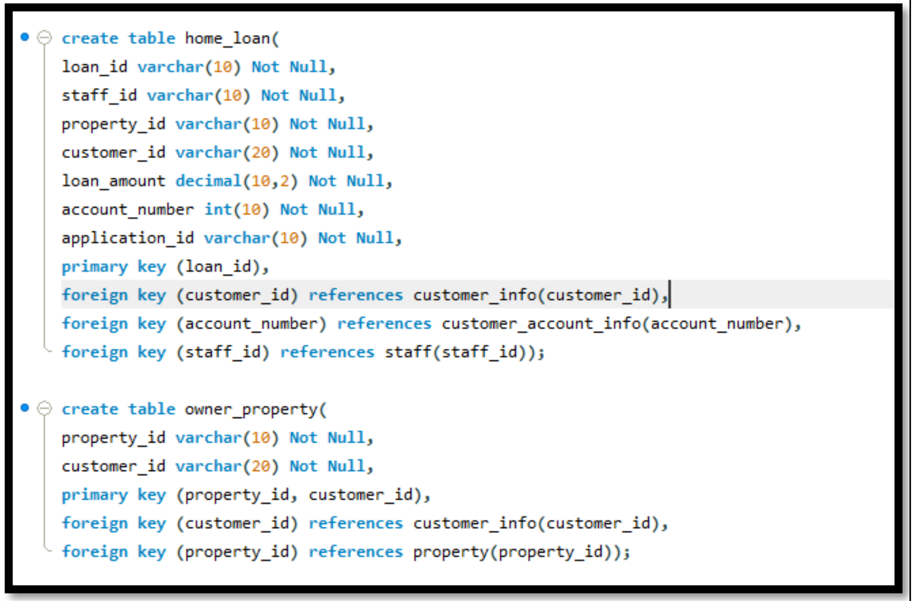
```sql
Create table postcodes(
postcode char(4) Not Null,
suburb varchar(20) Not Null,
state char(3) Not Null,
primary key (postcode));

create table account_type_id(
account_type_id varchar(10) Not Null,
account_description text(20) Not Null,
primary key (account_type_id));

create table application_outcome (
application_id varchar(10) Not Null,
application_outcome varchar(20) Not Null,
primary key (application_id));

create table staff (
staff_id varchar(10) Not Null,
staff_name varchar (20) Not Null,
staff_position varchar (20) Not Null,
primary key (staff_id));
```



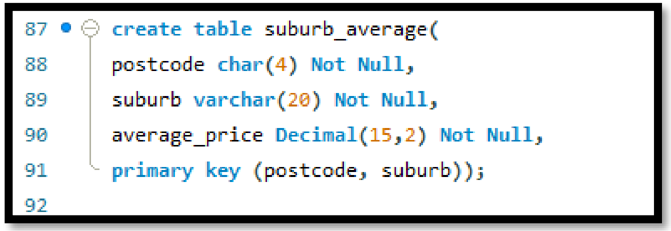
```sql
Create table customer_info (
customer_id varchar(20) Not Null,
customer_salary Decimal(15,2) Not Null,
customer_name char(20) Not Null,
customer_address1 varchar(30) Not Null,
postcode char(4) Not Null,
loan_id varchar(10) Null,
primary key (customer_id),
foreign key (postcode) references postcodes(postcode));

create table property(
property_id varchar(10) Not Null,
property_value decimal(8,2) Not Null,
property_address_line1 varchar(30) Not Null,
sold_price decimal(8,2) Not Null,
sold_date date Not Null,
customer_id varchar(20) Not Null,
postcode char(4) Not Null,
primary key (property_id),
foreign key (customer_id) references customer_info(customer_id),
foreign key (postcode) references postcodes(postcode));
```



```
Create table customer_account_info(
account_number int(10) Not Null,
bsb_number varchar(7) Not Null,
account_balance decimal(8,2),
account_type_id varchar(10) Not Null,
primary key (account_number),
foreign key (account_type_id) references account_type_id(account_type_id));

create table customer_account(
customer_id varchar(20) Not Null,
account_number int(10) Not Null,
primary key (customer_id),
foreign key (account_number) references customer_account_info(account_number));

Create table loan_application (
customer_id varchar(20) Not Null,
property_id varchar(10) Not Null,
application_id varchar(10) Not Null,
primary key (customer_id, property_id),
foreign key (property_id) REFERENCES property(property_id),
foreign key (customer_id) REFERENCES customer_info(customer_id));
```



```
create table home_loan(
loan_id varchar(10) Not Null,
staff_id varchar(10) Not Null,
property_id varchar(10) Not Null,
customer_id varchar(20) Not Null,
loan_amount decimal(10,2) Not Null,
account_number int(10) Not Null,
application_id varchar(10) Not Null,
primary key (loan_id),
foreign key (customer_id) references customer_info(customer_id),
foreign key (account_number) references customer_account_info(account_number),
foreign key (staff_id) references staff(staff_id));

create table owner_property(
property_id varchar(10) Not Null,
customer_id varchar(20) Not Null,
primary key (property_id, customer_id),
foreign key (customer_id) references customer_info(customer_id),
foreign key (property_id) references property(property_id));
```



```
create table suburb_average(
postcode char(4) Not Null,
suburb varchar(20) Not Null,
average_price Decimal(15,2) Not Null,
primary key (postcode, suburb));
```

Home Loan Query – Insert Data

Postcodes:

```
insert into postcodes(postcode, suburb, state) values (3130, 'collingwood', 'VIC');
insert into postcodes(postcode, suburb, state) values (3000, 'Melbourne', 'VIC');
insert into postcodes(postcode, suburb, state) values (3132, 'Fitzroy', 'VIC');
insert into postcodes(postcode, suburb, state) values (3754, 'Seven Hills', 'VIC');
insert into postcodes(postcode, suburb, state) values (3153, 'Bayswater', 'VIC');
```

Customer_info:

```
Insert into customer_info (customer_id, customer_salary, customer_name, customer_address1, postcode, loan_id) values ('Cus-001', 150000.00, 'Tom Smith', '1 Smith Street', 3130, 'HL-001');
Insert into customer_info (customer_id, customer_salary, customer_name, customer_address1, postcode, loan_id) values ('Cus-002', 160000.00, 'Anna Smith', '1 Smith Street', 3130, 'HL-001');
Insert into customer_info (customer_id, customer_salary, customer_name, customer_address1, postcode, loan_id) values ('Cus-003', 75000.00, 'Andrea Smith', '1 Sam Street', 3000, 'HL-002');
Insert into customer_info (customer_id, customer_salary, customer_name, customer_address1, postcode, loan_id) values ('Cus-004', 95000.00, 'Mark Smith', '5 John Street', 3132, 'HL-003');
Insert into customer_info (customer_id, customer_salary, customer_name, customer_address1, postcode, loan_id) values ('Cus-005', 60000.00, 'Andrew Smith', '5 Mark Street', 3000, 'HL-004');
```

Property:

```
insert into property(property_id, property_value, property_address_line1, sold_price, sold_date, customer_id, postcode) values ('PID-001', 700000.00, '1 Seven Hills Rd', 500000, '2019-12-01', 'Cus-001', 3754);
insert into property(property_id, property_value, property_address_line1, sold_price, sold_date, customer_id, postcode) values ('PID-005', 600000.00, '20 Melview Dr', 4, '2019-10-01', 'Cus-001', 3132);
insert into property(property_id, property_value, property_address_line1, sold_price, sold_date, customer_id, postcode) values ('PID-002', 500000.00, '50 Bakes Rd', 300000, '2018-08-01', 'Cus-003', 3000);
insert into property(property_id, property_value, property_address_line1, sold_price, sold_date, customer_id, postcode) values ('PID-003', 450000.00, '70 East parade', 300000, '2017-01-01', 'Cus-004', 3130);
insert into property(property_id, property_value, property_address_line1, sold_price, sold_date, customer_id, postcode) values ('PID-004', 800000.00, '12 John Street', 750000, '2021-12-01', 'Cus-005', 3130);
```

account_type_id:

```
insert into account_type_id(account_type_id, account_description) values ('AT-001', 'Savings Account');
insert into account_type_id(account_type_id, account_description) values ('AT-002', 'Credit Account');
insert into account_type_id(account_type_id, account_description) values ('AT-003', 'Fixed Deposit');
insert into account_type_id(account_type_id, account_description) values ('AT-004', 'Everyday Account');
```

customer_account_info:

```
Insert into customer_account_info(account_number, bsb_number, account_balance, account_type_id) values (123456, '322_010', 50000.00, 'AT-001');
Insert into customer_account_info(account_number, bsb_number, account_balance, account_type_id) values (234567, '322_010', 70000.00, 'AT-001');
Insert into customer_account_info(account_number, bsb_number, account_balance, account_type_id) values (345678, '322_010', 35000.00, 'AT-001');
Insert into customer_account_info(account_number, bsb_number, account_balance, account_type_id) values (456789, '322_010',70000.00, 'AT-001');
Insert into customer_account_info(account_number, bsb_number, account_balance, account_type_id) values (567891, '322_010', 45000.00, 'AT-001');
```

customer_account:

```
insert into customer_account(customer_id, account_number) values ('Cus-001', 123456);
insert into customer_account(customer_id, account_number) values ('Cus-002', 234567);
insert into customer_account(customer_id, account_number) values ('Cus-003', 345678);
insert into customer_account(customer_id, account_number) values ('Cus-004', 456789);
insert into customer_account(customer_id, account_number) values ('Cus-005', 567891);
```

home_loan:

```
insert into home_loan(loan_id, staff_id, property_id, customer_id, loan_amount, account_number, application_id) values ('HL-001', 'ST-001', 'PID-001', 'Cus-001', 300000.00, 123456, 'HLA-005');
insert into home_loan(loan_id, staff_id, property_id, customer_id, loan_amount, account_number, application_id) values ('HL-005', 'ST-001', 'PID-005', 'Cus-001', 200000.00, 123456, 'HLA-006');
insert into home_loan(loan_id, staff_id, property_id, customer_id, loan_amount, account_number, application_id) values ('HL-002', 'ST-001', 'PID-002', 'Cus-003', 500000.00, 345678, 'HLA-002');
insert into home_loan(loan_id, staff_id, property_id, customer_id, loan_amount, account_number, application_id) values ('HL-003', 'ST-001', 'PID-003', 'Cus-004', 600000.00, 456789, 'HLA-003');
insert into home_loan(loan_id, staff_id, property_id, customer_id, loan_amount, account_number, application_id) values ('HL-004', 'ST-001', 'PID-004', 'Cus-005', 450000.00, 567891, 'HLA-004');
```

Suburb_average:

```
insert into suburb_average(postcode, suburb, average_price) values (3134, 'Ringwood', 900000.00);
insert into suburb_average(postcode, suburb, average_price) values (3153, 'Bayswater', 800000.00);
insert into suburb_average(postcode, suburb, average_price) values (3002, 'East Melbourne', 1200000.00);
insert into suburb_average(postcode, suburb, average_price) values (3145, 'Heathmont', 850000.00);
insert into suburb_average(postcode, suburb, average_price) values (3155, 'Wantirna', 900000.00);
```

Question 2 – Aladin Company

Aladin Assumptions & Business Rules

Assumptions:

- For a supplier to be on the database, the supplier must have sold products to Aladin
- Last transaction date is a calculated field that is obtained from a transactions log that is not in the scope of the assignment brief

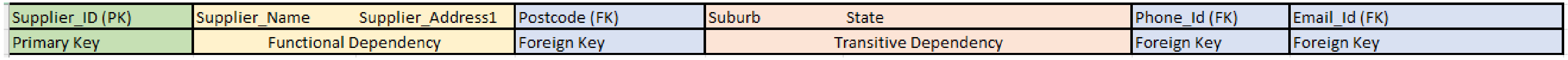
Business Rules:

- Supplier can have more than 1 and maximum 2 phone numbers. A phone_id and email_id key was created so that this can be used to look up the multiple numbers in the supplier email and supplier phone entity.
- Supplier can supply 1 or many products
- A product can be purchased and reflected in a purchase item, or a product can stay as not been purchased.
- A purchase item is unique for each product for each purchase order, thus can only have 1 purchase order linked
- A customer can have a sales list item or can stay as a customer without any sales
- Purchase item is a bridging entity between product and purchase order to avoid many to many relationships
- Staff has to complete a purchase order. A staff can remain in the staff entity without a purchase order as the staff can be staffed in other areas of the business

Aladin Company Normalization

2.1) Supplier Entity Schema

Not Normalized:

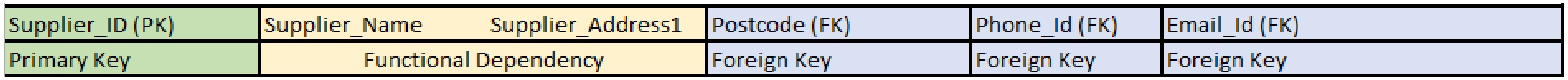


Supplier_ID (PK)	Supplier_Name	Supplier_Address1	Postcode (FK)	Suburb	State	Phone_Id (FK)	Email_Id (FK)
Primary Key	Functional Dependency		Foreign Key	Transitive Dependency		Foreign Key	Foreign Key

2nd Normal Form: Table is in second normal form as there is no use of composite keys, thus no partial dependency, but there is a transitive dependency that needs to be normalized to 3rd normal form with the postcode, suburb and state.

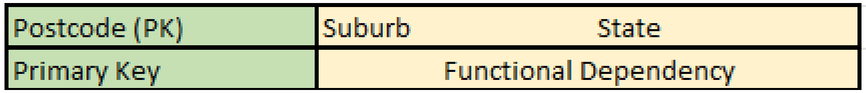
3rd Normal Form:

Table 1 (Supplier Contact Info Table):



Supplier_ID (PK)	Supplier_Name	Supplier_Address1	Postcode (FK)	Phone_Id (FK)	Email_Id (FK)
Primary Key	Functional Dependency		Foreign Key	Foreign Key	Foreign Key

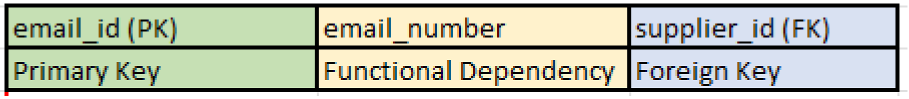
Table 2 (Postcode Table):



Postcode (PK)	Suburb	State
Primary Key	Functional Dependency	

2.2) Supplier Email Entity Schema

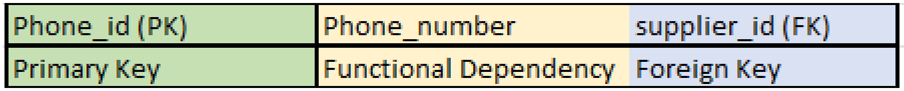
This table is already normalized to 2nd and 3rd normal form as there is not partial or transitive dependencies



email_id (PK)	email_number	supplier_id (FK)
Primary Key	Functional Dependency	Foreign Key

2.3) Supplier Phone Entity Schema

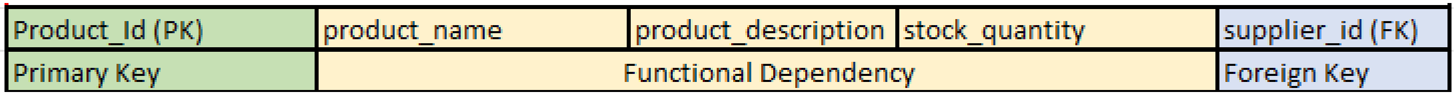
This table is already normalized to 2nd and 3rd normal form as there is not partial or transitive dependencies



Phone_id (PK)	Phone_number	supplier_id (FK)
Primary Key	Functional Dependency	Foreign Key

2.4) Product Entity Schema

Not Normalized:

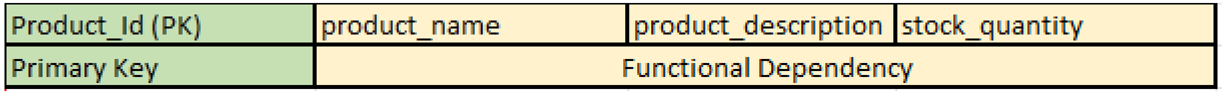


Product_Id (PK)	product_name	product_description	stock_quantity	supplier_id (FK)
Primary Key	Functional Dependency			Foreign Key

2nd Normal Form:

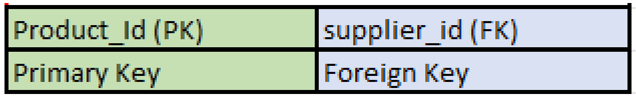
There are no partial dependencies, but I have broken down the table into 2 separate tables to avoid any update errors.

Table 1 (Product Stock Info):



Product_Id (PK)	product_name	product_description	stock_quantity
Primary Key	Functional Dependency		

Table 2 (Product Supplier):

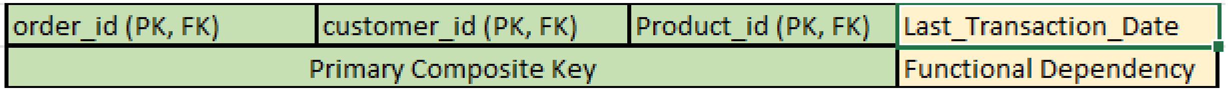


Product_Id (PK)	supplier_id (FK)
Primary Key	Foreign Key

3rd Normal Form: There are no transitive dependencies in the tables above.

2.5) Sales Entity Schema

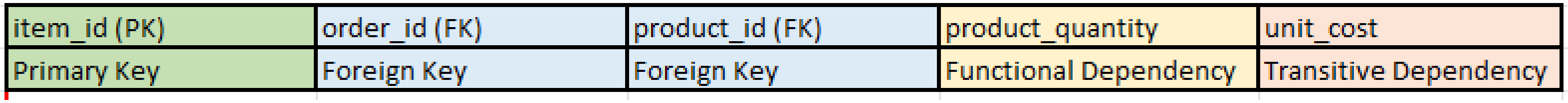
There are no partial or transitive dependencies in this table.



order_id (PK, FK)	customer_id (PK, FK)	Product_id (PK, FK)	Last_Transaction_Date
Primary Composite Key			Functional Dependency

2.6) Purchase Item Entity Schema

Not Normalized:

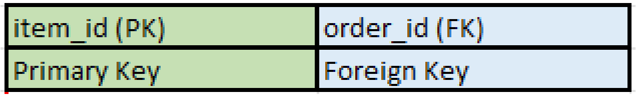


item_id (PK)	order_id (FK)	product_id (FK)	product_quantity	unit_cost
Primary Key	Foreign Key	Foreign Key	Functional Dependency	Transitive Dependency

2nd Normal Form: There are no partial dependencies as there is no composite primary key, so we can proceed to normalize to 3NF to deal with the transitive dependency between product_id and unit_cost.

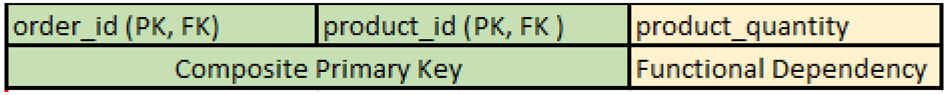
3rd Normal Form:

Table 1 (Purchase Item Table):



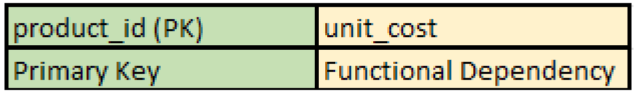
item_id (PK)	order_id (FK)
Primary Key	Foreign Key

Table 2 (Purchase item Order Table):



order_id (PK, FK)	product_id (PK, FK)	product_quantity
Composite Primary Key		Functional Dependency

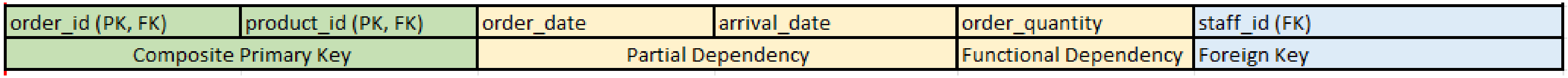
Table 3 (Product Unit Cost Table):



product_id (PK)	unit_cost
Primary Key	Functional Dependency

2.7) Purchase Order Entity Schema

Not Normalized:

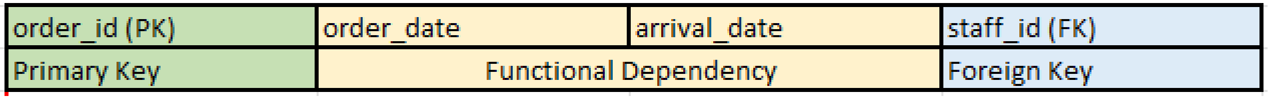


order_id (PK, FK)	product_id (PK, FK)	order_date	arrival_date	order_quantity	staff_id (FK)
Composite Primary Key		Partial Dependency		Functional Dependency	Foreign Key

2nd Normal Form:

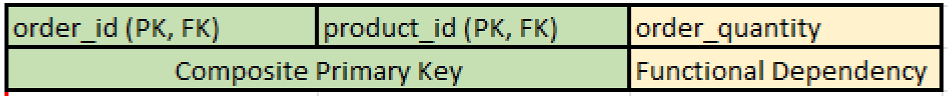
There are partial dependencies between order_date, arrival_date only partially dependent on order_id, and not product_id. So I’ve broken down to 2 further tables:

Table 1 (Order Details):



order_id (PK)	order_date	arrival_date	staff_id (FK)
Primary Key	Functional Dependency		Foreign Key

Table 2 (Order Quantity):

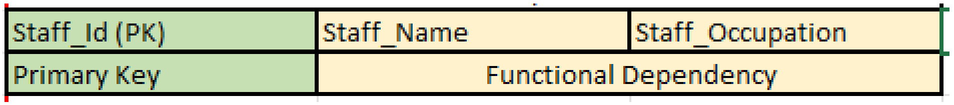


order_id (PK, FK)	product_id (PK, FK)	order_quantity
Composite Primary Key		Functional Dependency

3rd Normal Form: There are no transitive dependencies, so table is already in 3rd normal form.

2.8) Staff Entity Schema

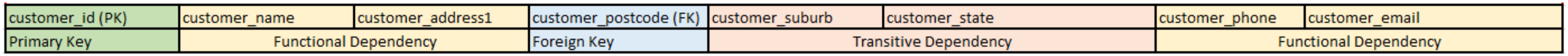
Table is already normalized to 2nd and 3rd normal forms:



Staff_Id (PK)	Staff_Name	Staff_Occupation
Primary Key	Functional Dependency	

2.9) Customer Entity Schema

Not Normalized:

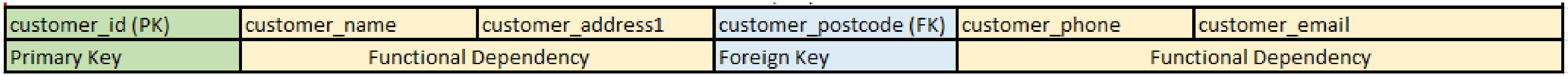


customer_id (PK)	customer_name	customer_address1	customer_postcode (FK)	customer_suburb	customer_state	customer_phone	customer_email
Primary Key	Functional Dependency		Foreign Key	Transitive Dependency		Functional Dependency	

2nd Normal Form: There are no partial dependency to deal with, so we will move to 3rd normal form to remove the transitive dependency.

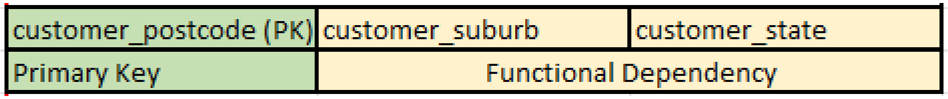
3rd Normal Form:

Table 1 (Customer Company Contact Info):



customer_id (PK)	customer_name	customer_address1	customer_postcode (FK)	customer_phone	customer_email
Primary Key	Functional Dependency		Foreign Key	Functional Dependency	

Table 2 (Customer Company Postcode):



customer_postcode (PK)	customer_suburb	customer_state
Primary Key	Functional Dependency	

Question 3 – Deakin Student Accommodation

Deakin Student Accommodation Assumptions & Business Rules

Assumptions:

- Each bed that is available to be leased has a unique place_id.
- Student can only be assigned 1 mentor
- Program entity is the same as course entity (Master of Business Analytics)
- Room number under program/course relates to the room that the director sits in within the campus
- Student can have 2 leases as there maybe a transition period between the current lease ending and new lease starting.

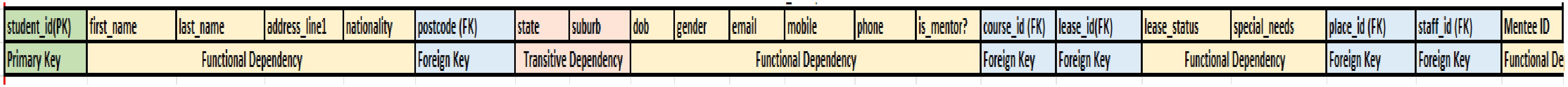
Business Rules:

- Student can only be enrolled in 1 course/program
- Staff entity covers all staff – advisors, staff that inspects, course director and hall manager which can be looked up using the staff id within the inspections and program entities.
- Place entity is only linked to the individual rooms and bed (dorms) and is linked to the student's entity.
- A student can have either 0 place as they may be on a waiting list, or 2 place as they maybe transitioning between years/accommodations type
- A staff can be assigned to either 0 or more than 1 program as the director
- A staff can be assigned to 0 or more than 1 inspection or 0 or more than 1 hall managers.
- A student can have 0 or more than 1 inspection
- Each unit can only have between 3-5 rooms
- Each dorm bed can only be assigned to an undergrad student
- Each dorm can only have 1 building
- Each dorm bed can only have 1 dorm
- Each hall room can only have 1 hall id
- Each unit room can only have 1 unit id

Deakin Student Accommodation Normalization

3.1) Student Entity Schema

Not Normalized:



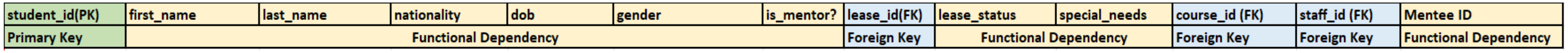
student_id(PK)	first_name	last_name	address_line1	nationality	postcode (FK)	state	suburb	dob	gender	email	mobile	phone	is_mentor?	course_id (FK)	lease_id(FK)	lease_status	special_needs	place_id (FK)	staff_id (FK)	Mentee ID
Primary Key	Functional Dependency				Foreign Key	Transitive Dependency		Functional Dependency						Foreign Key	Foreign Key	Functional Dependency		Foreign Key	Foreign Key	Functional De

2nd Normal Form:

There are no partial dependencies here as I've not used a primary composite key, but there is a transitive dependency with the postcode, state and suburb that needs to be dealt with.

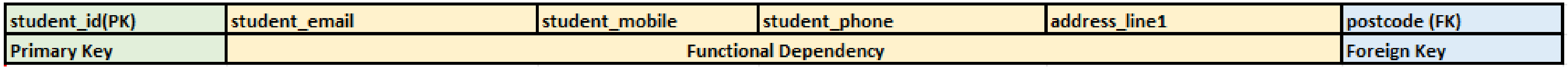
3rd Normal form:

Table 1 (student_details):



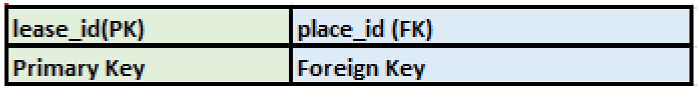
student_id(PK)	first_name	last_name	nationality	dob	gender	is_mentor?	lease_id(FK)	lease_status	special_needs	course_id (FK)	staff_id (FK)	Mentee ID
Primary Key	Functional Dependency						Foreign Key	Functional Dependency		Foreign Key	Foreign Key	Functional Dependency

Table 2 (student_contact):



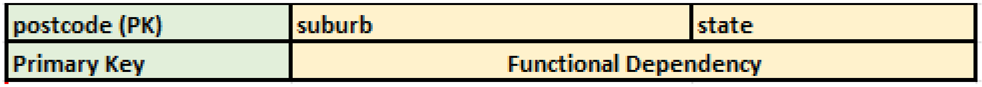
student_id(PK)	student_email	student_mobile	student_phone	address_line1	postcode (FK)
Primary Key	Functional Dependency				Foreign Key

Table 3 (student_lease_place):



lease_id(PK)	place_id (FK)
Primary Key	Foreign Key

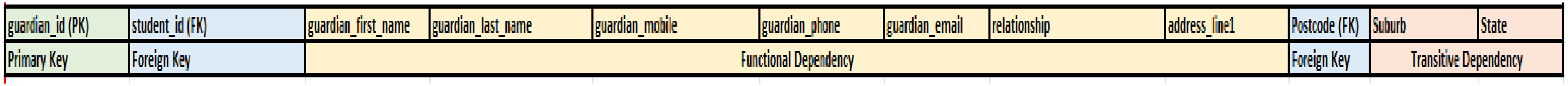
Table 4 (Postcodes):



postcode (PK)	suburb	state
Primary Key	Functional Dependency	

3.2) Guardian Entity Schema

Not Normalized:



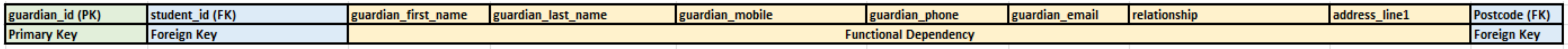
guardian_id (PK)	student_id (FK)	guardian_first_name	guardian_last_name	guardian_mobile	guardian_phone	guardian_email	relationship	address_line1	Postcode (FK)	Suburb	State
Primary Key	Foreign Key	Functional Dependency							Foreign Key	Transitive Dependency	

2nd Normal Form:

There are no partial dependencies here as I've not used a primary composite key, but there is a transitive dependency with the postcode, state and suburb that needs to be dealt with.

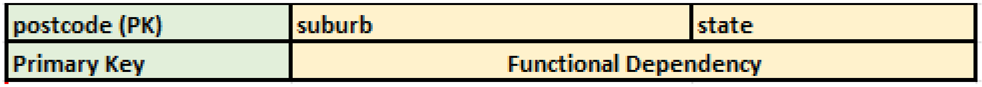
3rd Normal Form:

Table 1 (Guardian Info):



guardian_id (PK)	student_id (FK)	guardian_first_name	guardian_last_name	guardian_mobile	guardian_phone	guardian_email	relationship	address_line1	Postcode (FK)
Primary Key	Foreign Key	Functional Dependency							Foreign Key

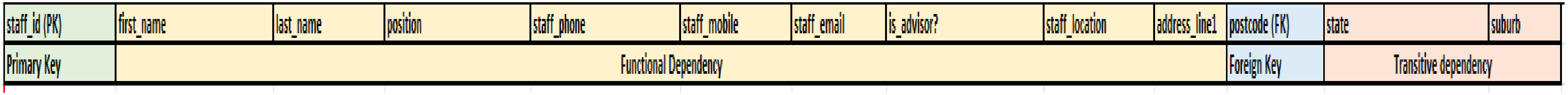
Table 2 (Postcode):



postcode (PK)	suburb	state
Primary Key	Functional Dependency	

3.3) Staff Entity Schema

Not Normalized:

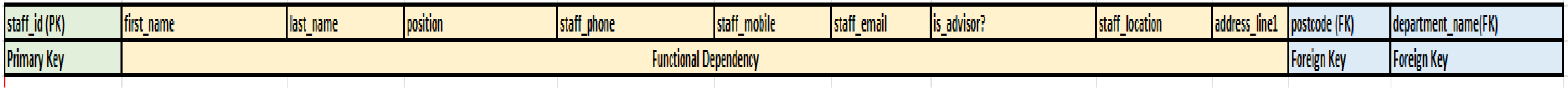


staff_id (PK)	first_name	last_name	position	staff_phone	staff_mobile	staff_email	is_advisor?	staff_location	address_line1	postcode (FK)	state	suburb
Primary Key	Functional Dependency									Foreign Key	Transitive dependency	

There are no partial dependencies here as I've not used a primary composite key, but there is a transitive dependency with the postcode, state and suburb that needs to be dealt with.

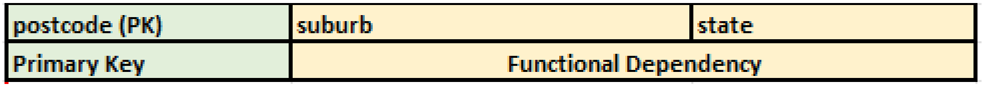
3rd Normal Form:

Table 1 (Staff Info):



staff_id (PK)	first_name	last_name	position	staff_phone	staff_mobile	staff_email	is_advisor?	staff_location	address_line1	postcode (FK)	department_name(FK)
Primary Key	Functional Dependency									Foreign Key	Foreign Key

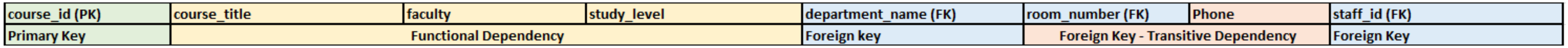
Table 2 (Postcode):



postcode (PK)	suburb	state
Primary Key	Functional Dependency	

3.4) Program Entity Schema

Not Normalized:

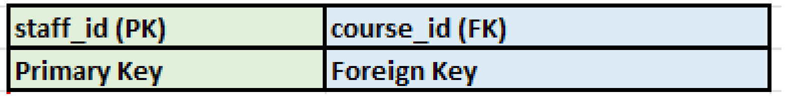


course_id (PK)	course_title	faculty	study_level	department_name (FK)	room_number (FK)	Phone	staff_id (FK)
Primary Key	Functional Dependency			Foreign key	Foreign Key - Transitive Dependency		Foreign Key

2nd Normal Form:

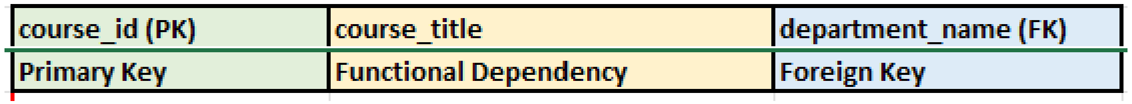
Even though there are no partial dependencies, I have broken down the program table further to have a separate staff course table as a staff can be assigned to more than 1 course and keeping the course details in the main program table will cause duplication of the other attributes.

Table 1 (Staff Course):



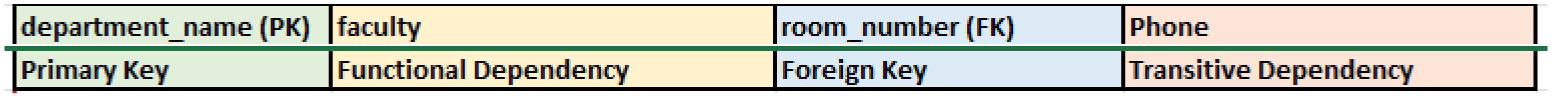
staff_id (PK)	course_id (FK)
Primary Key	Foreign Key

Table 2 (Course Description):



course_id (PK)	course_title	department_name (FK)
Primary Key	Functional Dependency	Foreign Key

Table 3 (Department Room):

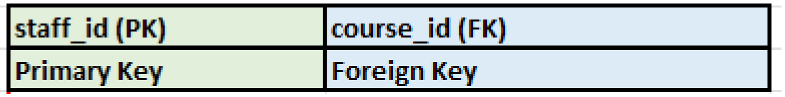


department_name (PK)	faculty	room_number (FK)	Phone
Primary Key	Functional Dependency	Foreign Key	Transitive Dependency

3rd Normal Form:

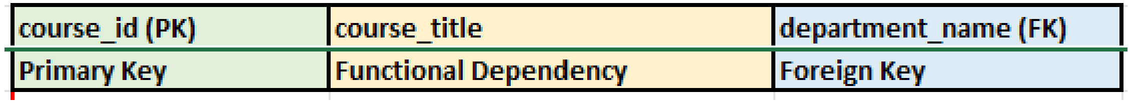
There is a transitive dependency with the department faculty table above where the phone number depends on the room number, so we will have to separate them further.

Table 1 (Staff Course):



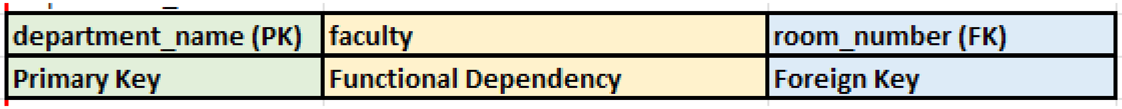
staff_id (PK)	course_id (FK)
Primary Key	Foreign Key

Table 2 (Course Description):



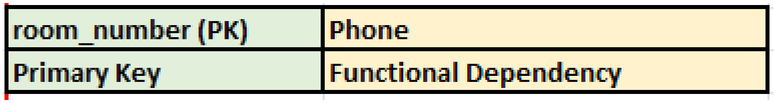
course_id (PK)	course_title	department_name (FK)
Primary Key	Functional Dependency	Foreign Key

Table 3 (Department Room):



department_name (PK)	faculty	room_number (FK)
Primary Key	Functional Dependency	Foreign Key

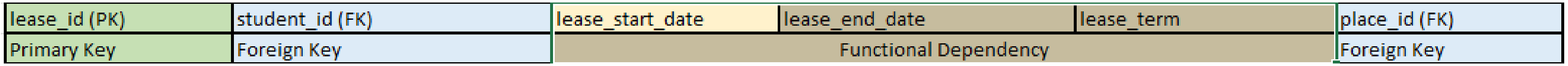
Table 4 (Room Phone):



room_number (PK)	Phone
Primary Key	Functional Dependency

3.5) Lease Entity Schema:

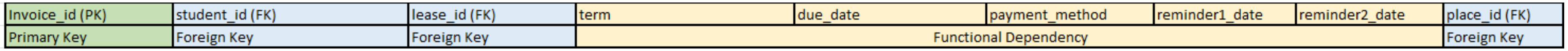
Table is already normalized to 2nd and 3rd normal forms:



lease_id (PK)	student_id (FK)	lease_start_date	lease_end_date	lease_term	place_id (FK)
Primary Key	Foreign Key	Functional Dependency			Foreign Key

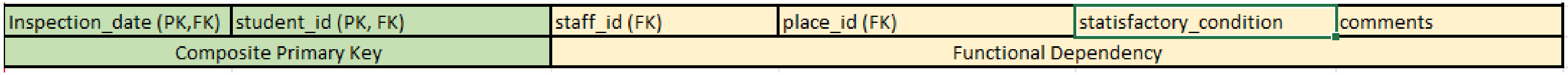
3.6) Payments Entity Schema:

Table is already normalized to 2nd and 3rd normal forms:



Invoice_id (PK)	student_id (FK)	lease_id (FK)	term	due_date	payment_method	reminder1_date	reminder2_date	place_id (FK)
Primary Key	Foreign Key	Foreign Key	Functional Dependency					Foreign Key

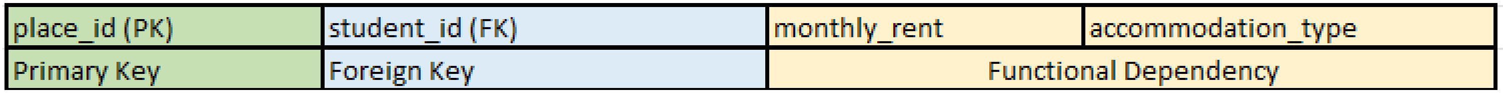
3.7) Inspections Entity Schema:



Inspection_date (PK,FK)	student_id (PK, FK)	staff_id (FK)	place_id (FK)	statisfactory_condition	comments
Composite Primary Key		Functional Dependency			

3.8) Place Entity Schema:

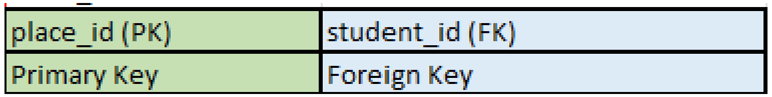
Not Normalized:



place_id (PK)	student_id (FK)	monthly_rent	accommodation_type
Primary Key	Foreign Key	Functional Dependency	

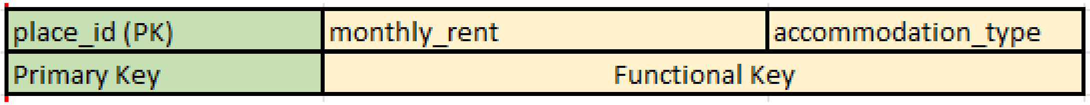
2nd Normal Form: There are no partial dependencies, but the table can be broken down further to avoid update anomalies.

Table 1 (Place Student Table):



place_id (PK)	student_id (FK)
Primary Key	Foreign Key

Table 2 (Place Details Table):

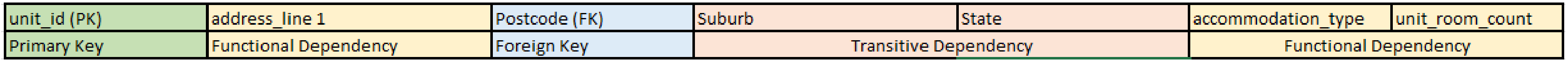


place_id (PK)	monthly_rent	accommodation_type
Primary Key	Functional Key	

3rd Normal Form: Tables are already in 3NF as there are no transitive dependencies.

3.9) Unit Entity Schema:

Not Normalized:

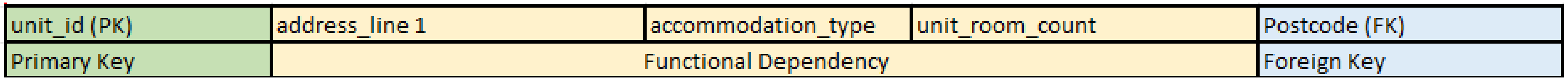


unit_id (PK)	address_line 1	Postcode (FK)	Suburb	State	accommodation_type	unit_room_count
Primary Key	Functional Dependency	Foreign Key	Transitive Dependency		Functional Dependency	

2nd Normal Form: Table has no partial dependencies, so we can proceed to 3rd normal form.

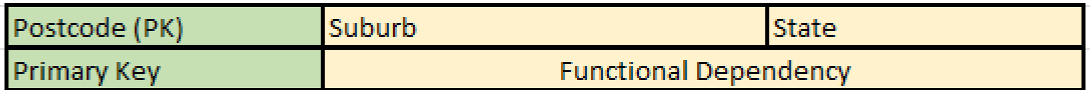
3rd Normal Form:

Table 1 (Unit Details Table):



unit_id (PK)	address_line 1	accommodation_type	unit_room_count	Postcode (FK)
Primary Key	Functional Dependency			Foreign Key

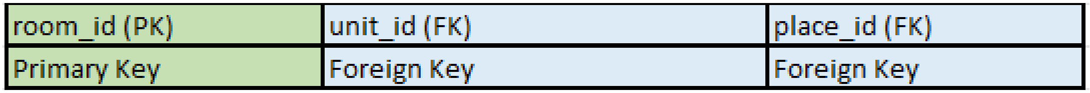
Table 2 (Postcode):



Postcode (PK)	Suburb	State
Primary Key	Functional Dependency	

3.10) Unit-Room Entity Schema:

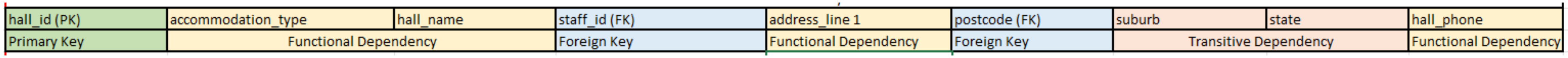
Table is already normalized to 3NF as there are no partial or transitive dependencies.



room_id (PK)	unit_id (FK)	place_id (FK)
Primary Key	Foreign Key	Foreign Key

3.11) Victoria Hall Entity Schema:

Not Normalized:

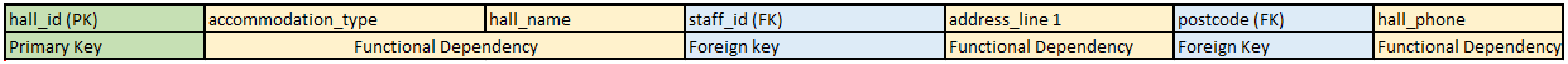


hall_id (PK)	accommodation_type	hall_name	staff_id (FK)	address_line 1	postcode (FK)	suburb	state	hall_phone
Primary Key	Functional Dependency		Foreign Key	Functional Dependency	Foreign Key	Transitive Dependency		Functional Dependency

2nd Normal Form: table is already in 2nd normal form as there are no partial dependency, so we can proceed to 3rd normal form.

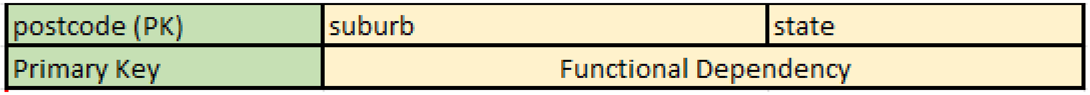
3rd Normal Form:

Table 1 (Hall Details Table):



hall_id (PK)	accommodation_type	hall_name	staff_id (FK)	address_line 1	postcode (FK)	hall_phone
Primary Key	Functional Dependency		Foreign key	Functional Dependency	Foreign Key	Functional Dependency

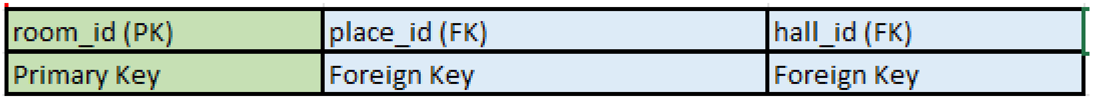
Table 2 (Postcode Table):



postcode (PK)	suburb	state
Primary Key	Functional Dependency	

3.12) Victoria Hall Room Entity Schema:

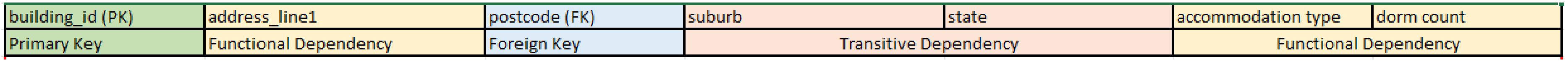
Table is already in 2nd and 3rd normal form.



room_id (PK)	place_id (FK)	hall_id (FK)
Primary Key	Foreign Key	Foreign Key

3.13) Dorm Building Entity Schema:

Not Normalized:

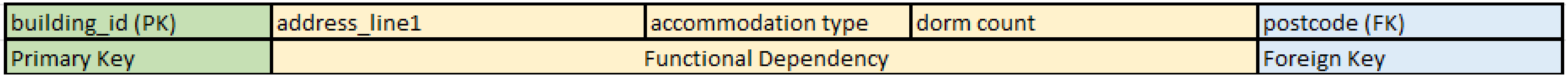


building_id (PK)	address_line1	postcode (FK)	suburb	state	accommodation type	dorm count
Primary Key	Functional Dependency	Foreign Key	Transitive Dependency		Functional Dependency	

2nd Normal Form: Table is already in 2NF, so we can proceed to 3NF.

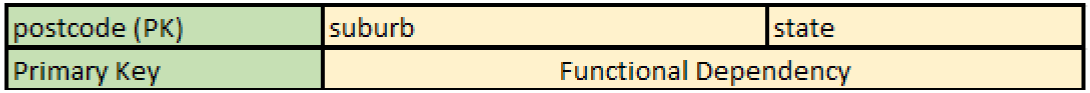
3rd Normal Form:

Table 1 (Dorm Building Table):



building_id (PK)	address_line1	accommodation type	dorm count	postcode (FK)
Primary Key	Functional Dependency			Foreign Key

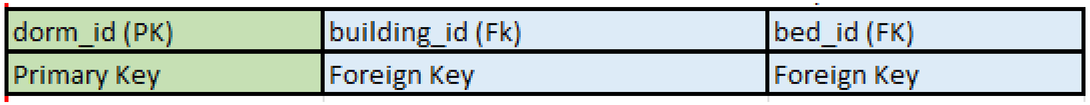
Table 2 (Postcode):



postcode (PK)	suburb	state
Primary Key	Functional Dependency	

3.14) Dorm Entity Schema:

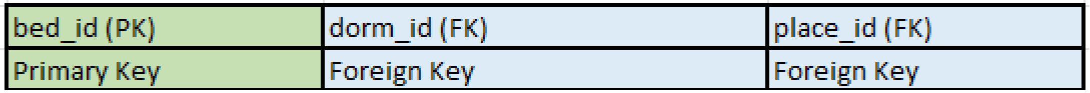
Table is already normalized to 2nd and 3rd normal forms:



dorm_id (PK)	building_id (Fk)	bed_id (FK)
Primary Key	Foreign Key	Foreign Key

3.15) Dorm Bed Entity Schema:

Table is already normalized to 2nd and 3rd normal forms:

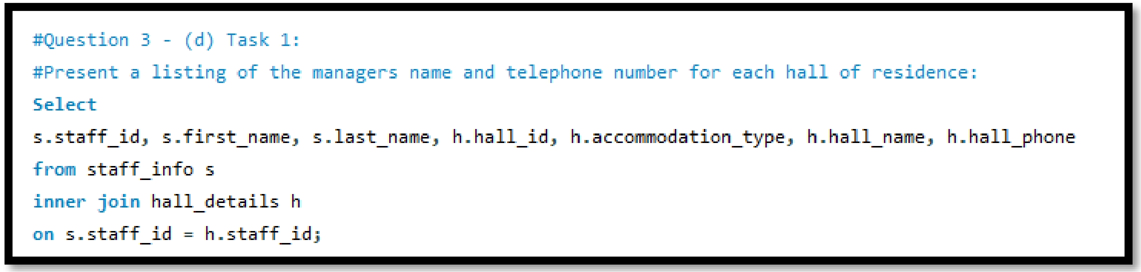


bed_id (PK)	dorm_id (FK)	place_id (FK)
Primary Key	Foreign Key	Foreign Key

Student Accommodation Queries

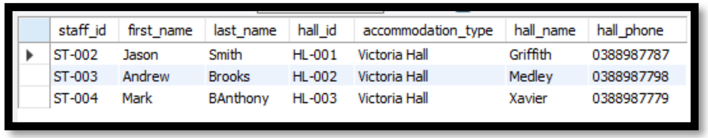
- Present a report listing the Manager's name and telephone number for each hall of residence

Query Logic



```
#Question 3 - (d) Task 1:
#Present a listing of the managers name and telephone number for each hall of residence:
Select
s.staff_id, s.first_name, s.last_name, h.hall_id, h.accommodation_type, h.hall_name, h.hall_phone
from staff_info s
inner join hall_details h
on s.staff_id = h.staff_id;
```

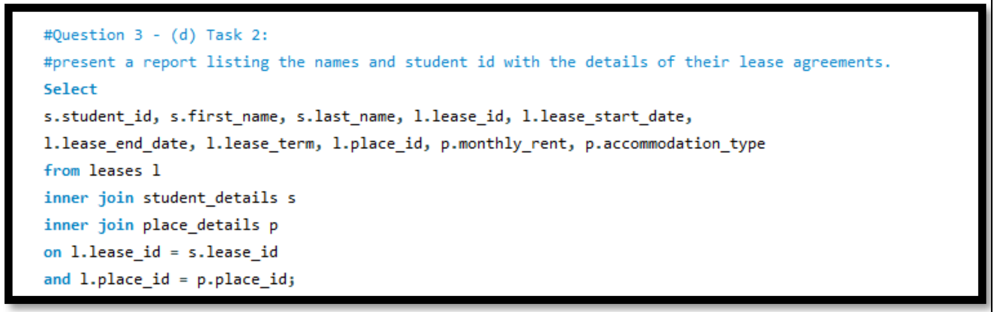
Query Results



staff_id	first_name	last_name	hall_id	accommodation_type	hall_name	hall_phone
ST-002	Jason	Smith	HL-001	Victoria Hall	Griffith	0388987787
ST-003	Andrew	Brooks	HL-002	Victoria Hall	Medley	0388987798
ST-004	Mark	BAnthony	HL-003	Victoria Hall	Xavier	0388987779

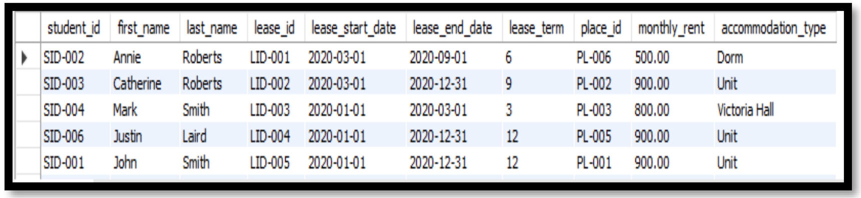
- Present a report listing the names and student id with the details of their lease agreements

Query Logic



```
#Question 3 - (d) Task 2:
#present a report listing the names and student id with the details of their lease agreements.
Select
s.student_id, s.first_name, s.last_name, l.lease_id, l.lease_start_date,
l.lease_end_date, l.lease_term, l.place_id, p.monthly_rent, p.accommodation_type
from leases l
inner join student_details s
inner join place_details p
on l.lease_id = s.lease_id
and l.place_id = p.place_id;
```

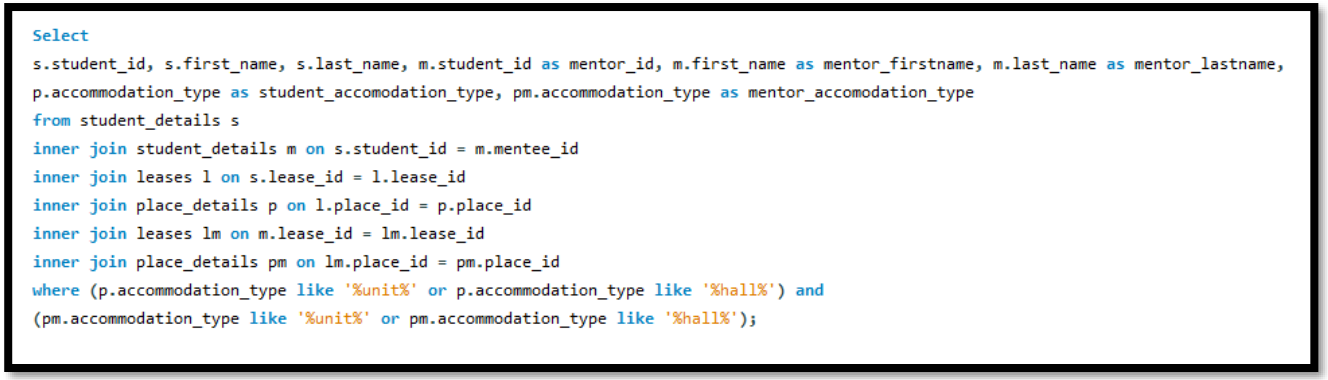
Query Results



	student_id	first_name	last_name	lease_id	lease_start_date	lease_end_date	lease_term	place_id	monthly_rent	accommodation_type
▶	SID-002	Annie	Roberts	LID-001	2020-03-01	2020-09-01	6	PL-006	500.00	Dorm
	SID-003	Catherine	Roberts	LID-002	2020-03-01	2020-12-31	9	PL-002	900.00	Unit
	SID-004	Mark	Smith	LID-003	2020-01-01	2020-03-01	3	PL-003	800.00	Victoria Hall
	SID-006	Justin	Laird	LID-004	2020-01-01	2020-12-31	12	PL-005	900.00	Unit
	SID-001	John	Smith	LID-005	2020-01-01	2020-12-31	12	PL-001	900.00	Unit

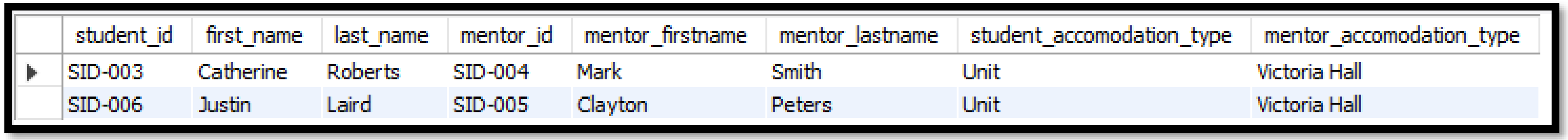
- List each student and his mentor who lives in either Victoria Hall or Deakin Unit

Query Logic



```
Select
s.student_id, s.first_name, s.last_name, m.student_id as mentor_id, m.first_name as mentor_firstname, m.last_name as mentor_lastname,
p.accommodation_type as student_accomodation_type, pm.accommodation_type as mentor_accomodation_type
from student_details s
inner join student_details m on s.student_id = m.mentee_id
inner join leases l on s.lease_id = l.lease_id
inner join place_details p on l.place_id = p.place_id
inner join leases lm on m.lease_id = lm.lease_id
inner join place_details pm on lm.place_id = pm.place_id
where (p.accommodation_type like '%unit%' or p.accommodation_type like '%hall%') and
(pm.accommodation_type like '%unit%' or pm.accommodation_type like '%hall%');
```

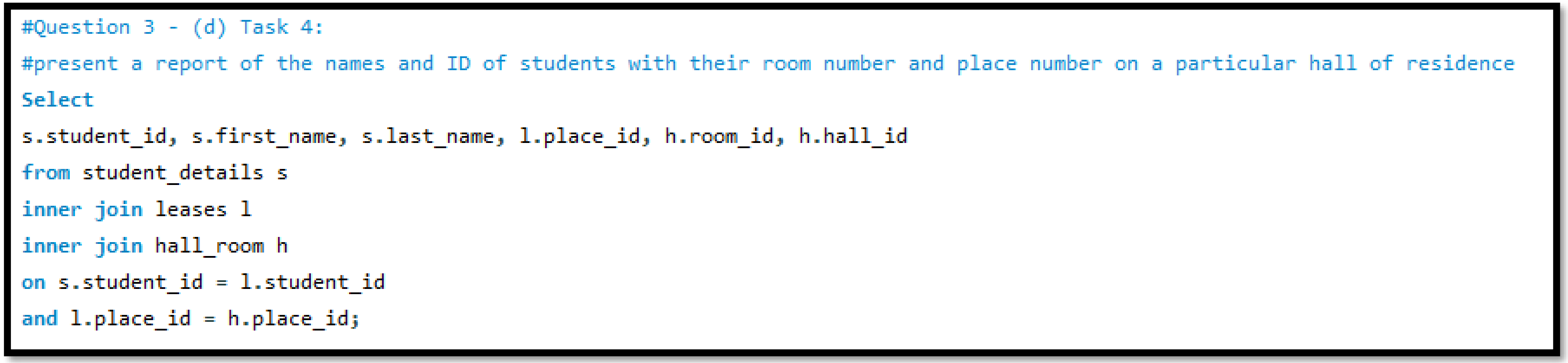
Query Results



	student_id	first_name	last_name	mentor_id	mentor_firstname	mentor_lastname	student_accomodation_type	mentor_accomodation_type
▶	SID-003	Catherine	Roberts	SID-004	Mark	Smith	Unit	Victoria Hall
	SID-006	Justin	Laird	SID-005	Clayton	Peters	Unit	Victoria Hall

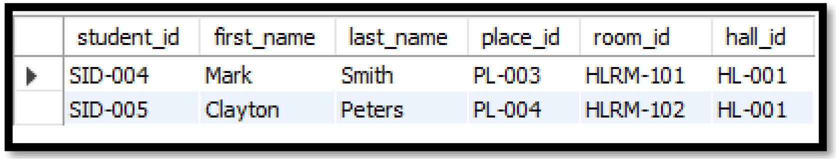
- Present a report of the names and ID of students with their room number and place number in a particular hall of residence

Query Logic



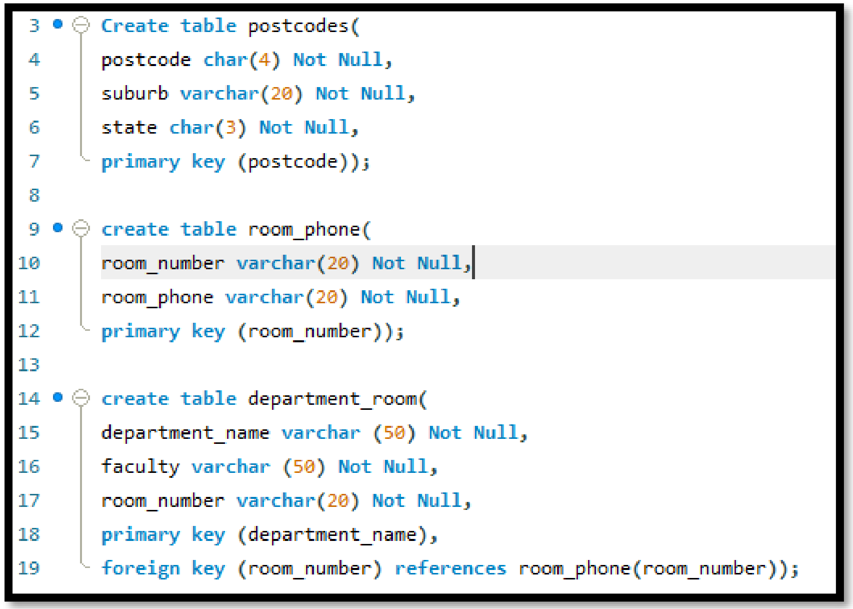
```
#Question 3 - (d) Task 4:
#present a report of the names and ID of students with their room number and place number on a particular hall of residence
Select
s.student_id, s.first_name, s.last_name, l.place_id, h.room_id, h.hall_id
from student_details s
inner join leases l
inner join hall_room h
on s.student_id = l.student_id
and l.place_id = h.place_id;
```

Query Results

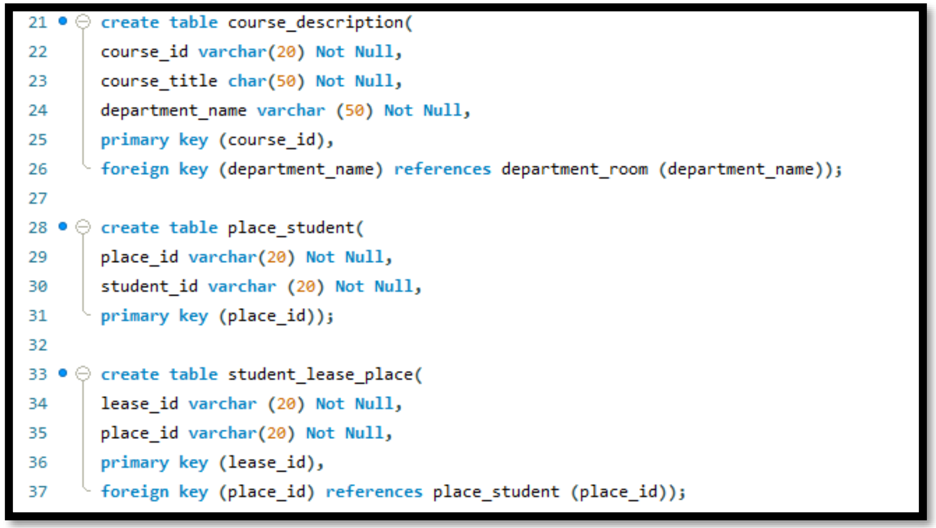


student_id	first_name	last_name	place_id	room_id	hall_id
SID-004	Mark	Smith	PL-003	HLRM-101	HL-001
SID-005	Clayton	Peters	PL-004	HLRM-102	HL-001

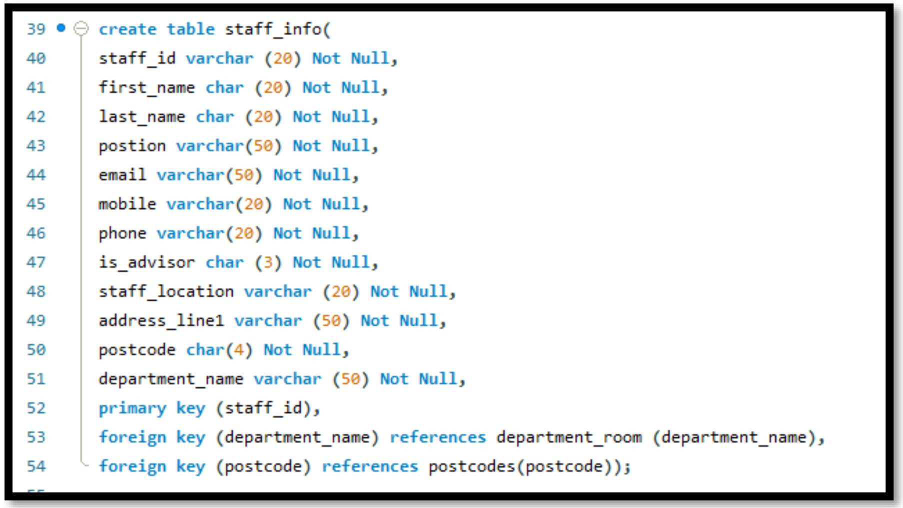
Student Accommodation – Create Tables



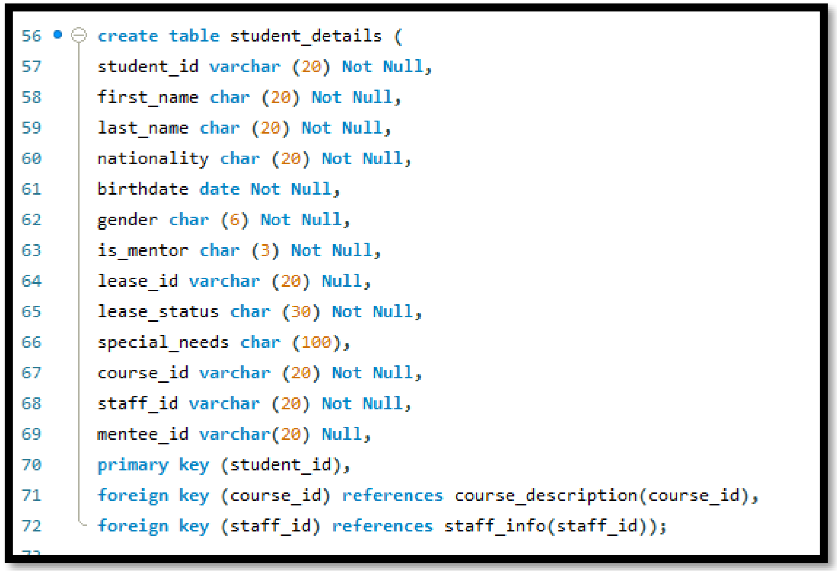
```
Create table postcodes(
postcode char(4) Not Null,
suburb varchar(20) Not Null,
state char(3) Not Null,
primary key (postcode));

create table room_phone(
room_number varchar(20) Not Null,
room_phone varchar(20) Not Null,
primary key (room_number));

create table department_room(
department_name varchar (50) Not Null,
faculty varchar (50) Not Null,
room_number varchar(20) Not Null,
primary key (department_name),
foreign key (room_number) references room_phone(room_number));
```



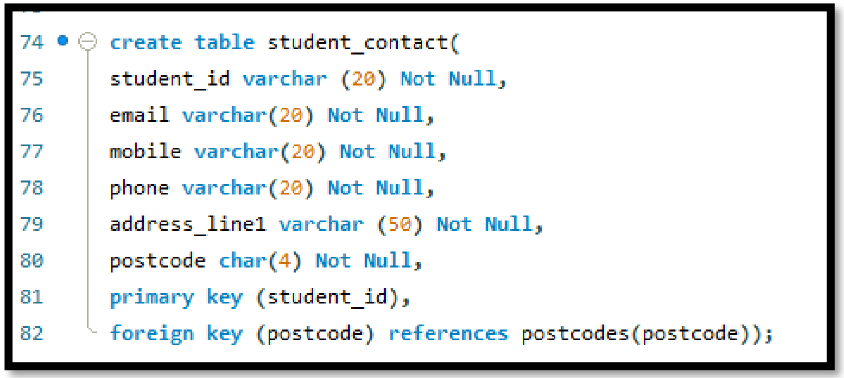
```
create table course_description(
course_id varchar(20) Not Null,
course_title char(50) Not Null,
department_name varchar (50) Not Null,
primary key (course_id),
foreign key (department_name) references department_room (department_name));

create table place_student(
place_id varchar(20) Not Null,
student_id varchar (20) Not Null,
primary key (place_id));

create table student_lease_place(
lease_id varchar (20) Not Null,
place_id varchar(20) Not Null,
primary key (lease_id),
foreign key (place_id) references place_student (place_id));
```



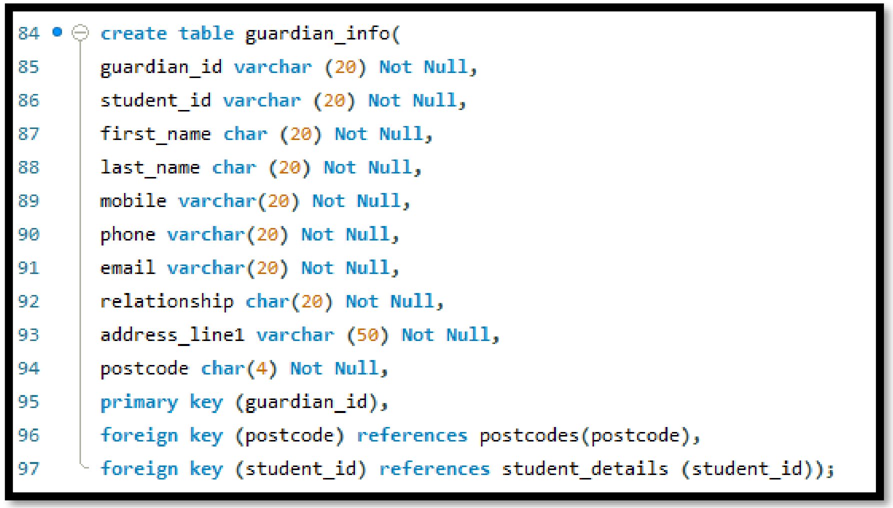
```sql
create table staff_info(
staff_id varchar (20) Not Null,
first_name char (20) Not Null,
last_name char (20) Not Null,
postion varchar(50) Not Null,
email varchar(50) Not Null,
mobile varchar(20) Not Null,
phone varchar(20) Not Null,
is_advisor char (3) Not Null,
staff_location varchar (20) Not Null,
address_line1 varchar (50) Not Null,
postcode char(4) Not Null,
department_name varchar (50) Not Null,
primary key (staff_id),
foreign key (department_name) references department_room (department_name),
foreign key (postcode) references postcodes(postcode));
```



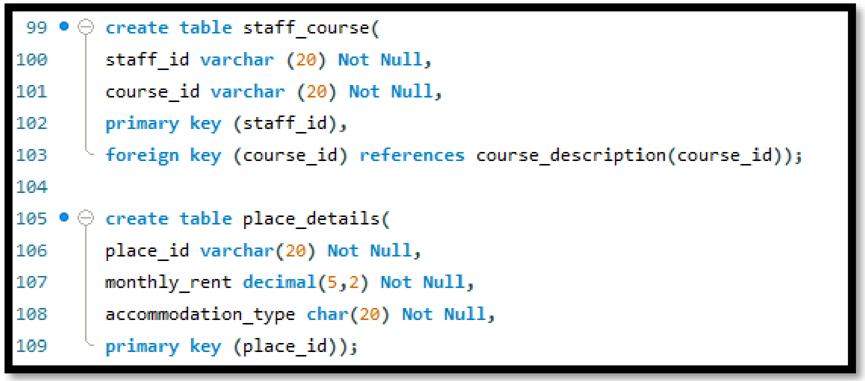
```sql
create table student_details (
student_id varchar (20) Not Null,
first_name char (20) Not Null,
last_name char (20) Not Null,
nationality char (20) Not Null,
birthdate date Not Null,
gender char (6) Not Null,
is_mentor char (3) Not Null,
lease_id varchar (20) Null,
lease_status char (30) Not Null,
special_needs char (100),
course_id varchar (20) Not Null,
staff_id varchar (20) Not Null,
mentee_id varchar(20) Null,
primary key (student_id),
foreign key (course_id) references course_description(course_id),
foreign key (staff_id) references staff_info(staff_id));
```



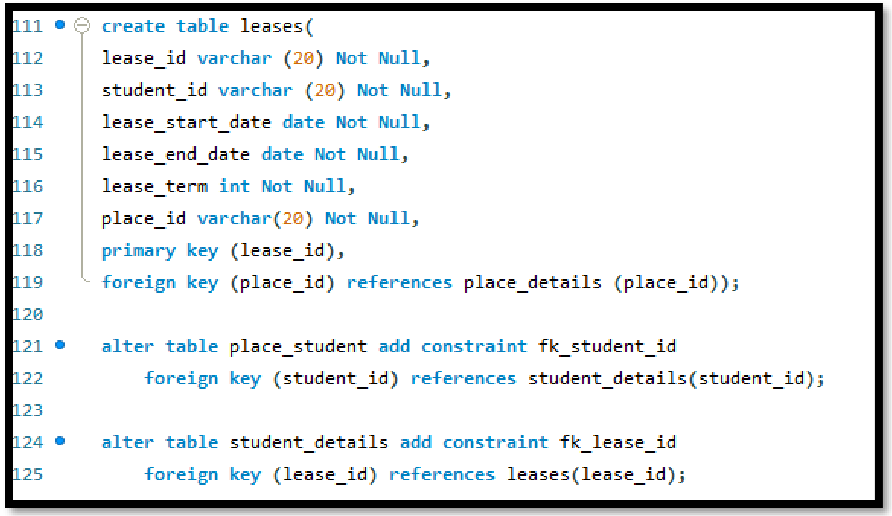
```sql
create table student_contact(
student_id varchar (20) Not Null,
email varchar(20) Not Null,
mobile varchar(20) Not Null,
phone varchar(20) Not Null,
address_line1 varchar (50) Not Null,
postcode char(4) Not Null,
primary key (student_id),
foreign key (postcode) references postcodes(postcode));
```



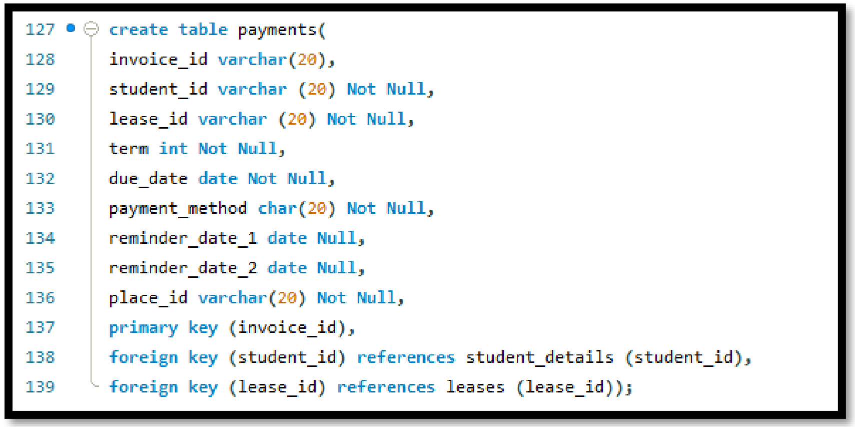
```sql
create table guardian_info(
guardian_id varchar (20) Not Null,
student_id varchar (20) Not Null,
first_name char (20) Not Null,
last_name char (20) Not Null,
mobile varchar(20) Not Null,
phone varchar(20) Not Null,
email varchar(20) Not Null,
relationship char(20) Not Null,
address_line1 varchar (50) Not Null,
postcode char(4) Not Null,
primary key (guardian_id),
foreign key (postcode) references postcodes(postcode),
foreign key (student_id) references student_details (student_id));
```



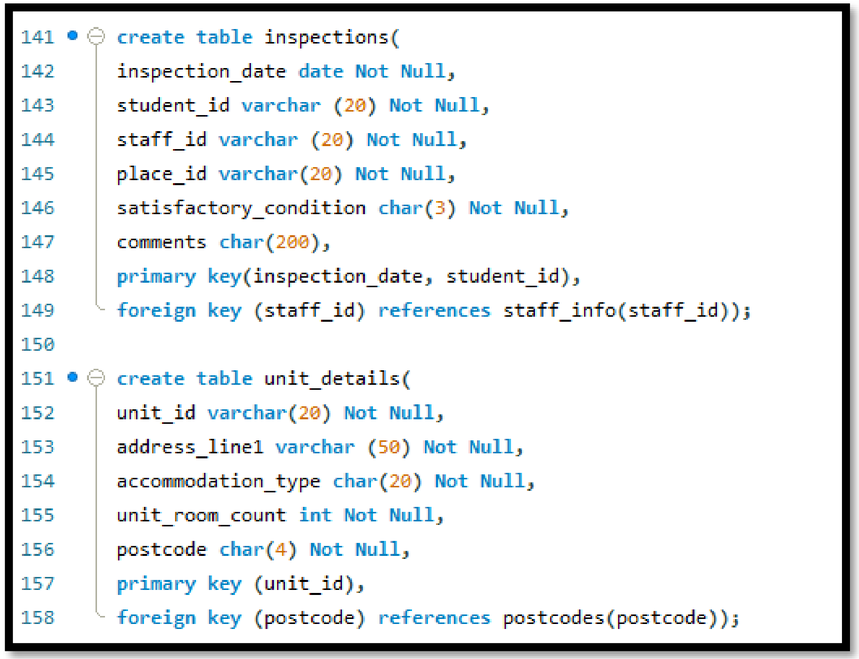
```sql
create table staff_course(
staff_id varchar (20) Not Null,
course_id varchar (20) Not Null,
primary key (staff_id),
foreign key (course_id) references course_description(course_id));

create table place_details(
place_id varchar(20) Not Null,
monthly_rent decimal(5,2) Not Null,
accommodation_type char(20) Not Null,
primary key (place_id));
```



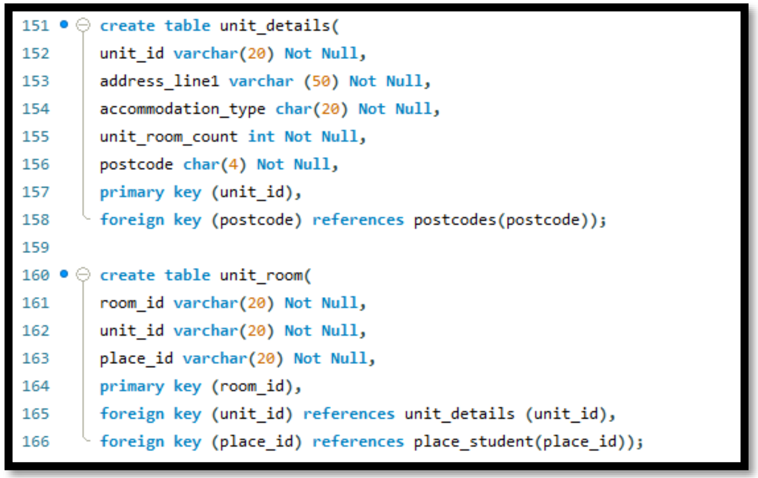
```sql
create table leases(
lease_id varchar (20) Not Null,
student_id varchar (20) Not Null,
lease_start_date date Not Null,
lease_end_date date Not Null,
lease_term int Not Null,
place_id varchar(20) Not Null,
primary key (lease_id),
foreign key (place_id) references place_details (place_id));

alter table place_student add constraint fk_student_id
    foreign key (student_id) references student_details(student_id);

alter table student_details add constraint fk_lease_id
    foreign key (lease_id) references leases(lease_id);
```



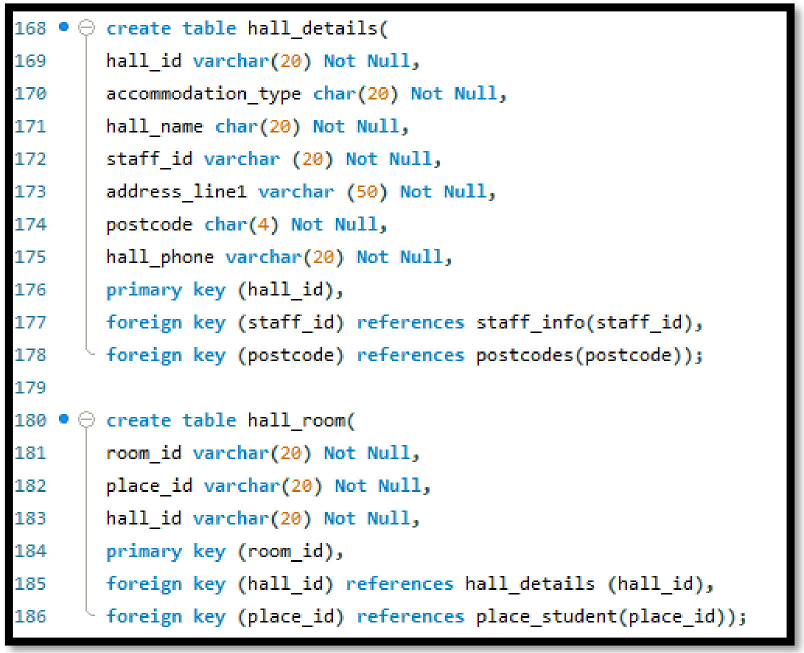
```sql
create table payments(
invoice_id varchar(20),
student_id varchar (20) Not Null,
lease_id varchar (20) Not Null,
term int Not Null,
due_date date Not Null,
payment_method char(20) Not Null,
reminder_date_1 date Null,
reminder_date_2 date Null,
place_id varchar(20) Not Null,
primary key (invoice_id),
foreign key (student_id) references student_details (student_id),
foreign key (lease_id) references leases (lease_id));
```



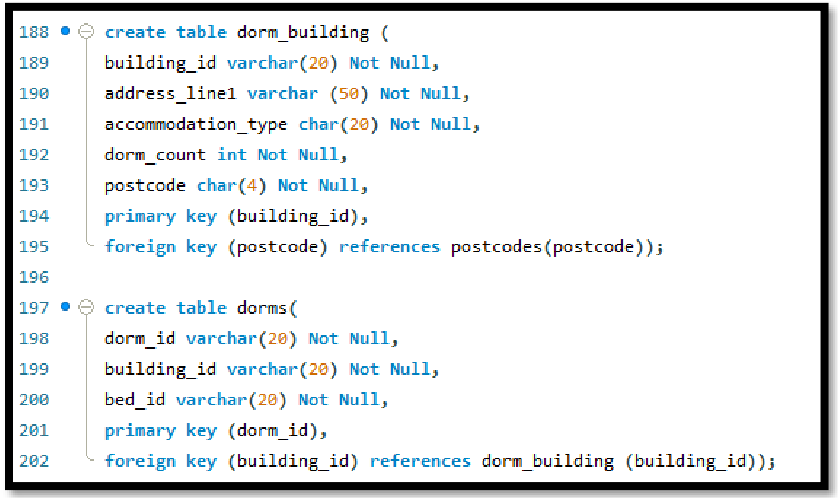
```sql
create table inspections(
inspection_date date Not Null,
student_id varchar (20) Not Null,
staff_id varchar (20) Not Null,
place_id varchar(20) Not Null,
satisfactory_condition char(3) Not Null,
comments char(200),
primary key(inspection_date, student_id),
foreign key (staff_id) references staff_info(staff_id));

create table unit_details(
unit_id varchar(20) Not Null,
address_line1 varchar (50) Not Null,
accommodation_type char(20) Not Null,
unit_room_count int Not Null,
postcode char(4) Not Null,
primary key (unit_id),
foreign key (postcode) references postcodes(postcode));
```



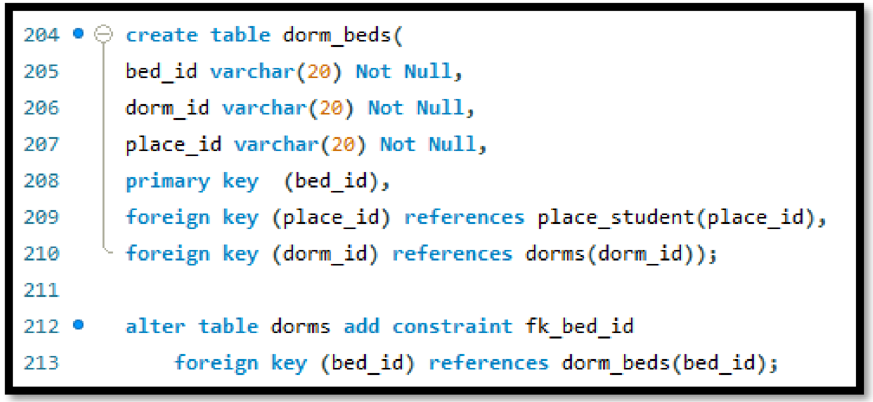
```sql
create table unit_details(
    unit_id varchar(20) Not Null,
    address_line1 varchar (50) Not Null,
    accommodation_type char(20) Not Null,
    unit_room_count int Not Null,
    postcode char(4) Not Null,
    primary key (unit_id),
    foreign key (postcode) references postcodes(postcode));

create table unit_room(
    room_id varchar(20) Not Null,
    unit_id varchar(20) Not Null,
    place_id varchar(20) Not Null,
    primary key (room_id),
    foreign key (unit_id) references unit_details (unit_id),
    foreign key (place_id) references place_student(place_id));
```



```sql
create table hall_details(
    hall_id varchar(20) Not Null,
    accommodation_type char(20) Not Null,
    hall_name char(20) Not Null,
    staff_id varchar (20) Not Null,
    address_line1 varchar (50) Not Null,
    postcode char(4) Not Null,
    hall_phone varchar(20) Not Null,
    primary key (hall_id),
    foreign key (staff_id) references staff_info(staff_id),
    foreign key (postcode) references postcodes(postcode));

create table hall_room(
    room_id varchar(20) Not Null,
    place_id varchar(20) Not Null,
    hall_id varchar(20) Not Null,
    primary key (room_id),
    foreign key (hall_id) references hall_details (hall_id),
    foreign key (place_id) references place_student(place_id));
```



```sql
create table dorm_building (
    building_id varchar(20) Not Null,
    address_line1 varchar (50) Not Null,
    accommodation_type char(20) Not Null,
    dorm_count int Not Null,
    postcode char(4) Not Null,
    primary key (building_id),
    foreign key (postcode) references postcodes(postcode));

create table dorms(
    dorm_id varchar(20) Not Null,
    building_id varchar(20) Not Null,
    bed_id varchar(20) Not Null,
    primary key (dorm_id),
    foreign key (building_id) references dorm_building (building_id));
```



```sql
create table dorm_beds(
    bed_id varchar(20) Not Null,
    dorm_id varchar(20) Not Null,
    place_id varchar(20) Not Null,
    primary key  (bed_id),
    foreign key (place_id) references place_student(place_id),
    foreign key (dorm_id) references dorms(dorm_id));

alter table dorms add constraint fk_bed_id
    foreign key (bed_id) references dorm_beds(bed_id);
```

Student Accommodation – Insert Data

Student_details:

```
insert into student_details(student_id, first_name, last_name, nationality, birthdate, gender, is_mentor, lease_id, lease_status, special_needs, course_id, staff_id,
mentee_id)
values ('SID-001', 'John', 'Smith', 'Australian', '1988-01-01', 'Male', 'No', 'LID-005', 'Leased', ' ', 'MIS716', 'ST-001', '');
insert into student_details(student_id, first_name, last_name, nationality, birthdate, gender, is_mentor, lease_id, lease_status, special_needs, course_id, staff_id,
mentee_id)
values ('SID-002', 'Annie', 'Roberts', 'American', '1989-01-01', 'Female', 'Yes', 'LID-001', 'Leased', ' ', 'MIS718', 'ST-001', 'SID-001');
insert into student_details(student_id, first_name, last_name, nationality, birthdate, gender, is_mentor, lease_id, lease_status, special_needs, course_id, staff_id,
mentee_id)
values ('SID-003', 'Catherine', 'Roberts', 'Australian', '1982-02-01', 'Female', 'No', 'LID-002', 'Leased', ' ', 'MIS716', 'ST-002','');
insert into student_details(student_id, first_name, last_name, nationality, birthdate, gender, is_mentor, lease_id, lease_status, special_needs, course_id, staff_id,
mentee_id)
values ('SID-004', 'Mark', 'Smith', 'Australian', '1982-02-28', 'Male', 'Yes', 'LID-003', 'Leased', ' ', 'MIS717', 'ST-003', 'SID-003');
insert into student_details(student_id, first_name, last_name, nationality, birthdate, gender, is_mentor, lease_id, lease_status, special_needs, course_id, staff_id,
mentee_id)
values ('SID-005', 'Clayton', 'Peters', 'Irish', '1979-07-28', 'Male', 'Yes', 'LID-007', 'Leased', 'Disabled Toilet', 'MIS715', 'ST-003', 'SID-006');
insert into student_details(student_id, first_name, last_name, nationality, birthdate, gender, is_mentor, lease_id, lease_status, special_needs, course_id, staff_id,
mentee_id)
values ('SID-006', 'Justin', 'Laird', 'Scottish', '1988-07-17', 'Male', 'No', 'LID-004', 'Leased', '', 'MIS716', 'ST-004','');
```

Student_lease_place:

```
insert into student_lease_place(lease_id, place_id) values ('LID-005', 'PL-001');
insert into student_lease_place(lease_id, place_id) values ('LID-001', 'PL-006');
insert into student_lease_place(lease_id, place_id) values ('LID-002', 'PL-002');
insert into student_lease_place(lease_id, place_id) values ('LID-003', 'PL-003');
insert into student_lease_place(lease_id, place_id) values ('LID-004', 'PL-005');
insert into student_lease_place(lease_id, place_id) values ('LID-007', 'PL-004');
```

Place_student:

```
insert into place_student (place_id, student_id) values ('PL-001', 'SID-001');
insert into place_student (place_id, student_id) values ('PL-006', 'SID-002');
insert into place_student (place_id, student_id) values ('PL-002', 'SID-003');
insert into place_student (place_id, student_id) values ('PL-003', 'SID-004');
insert into place_student (place_id, student_id) values ('PL-004', 'SID-005');
insert into place_student (place_id, student_id) values ('PL-005', 'SID-006');
```

Staff_info: insert into staff_info (staff_id, first_name, last_name, postion, email, mobile, phone, is_advisor, staff_location, address_line1, postcode, department_name) values ('ST-001', 'Michael', 'Patterson', 'Unit Chair - SIT772', 'michael.patterson@deakin.edu.au', '0412187722', '0399723344', 'No', 'Burwood Campus', '1 Melview Drive', '3134', 'School of IT'); insert into staff_info (staff_id, first_name, last_name, postion, email, mobile, phone, is_advisor, staff_location, address_line1, postcode, department_name) values ('ST-002', 'Jason', 'Smith', 'Accommodation Manager', 'jason.smith@deakin.edu.au', '0412187723', '0388723344', 'No', 'Victoria Hall', '20 Bedford Road', '3023', 'Deakin Student Accommodation'); insert into staff_info (staff_id, first_name, last_name, postion, email, mobile, phone, is_advisor, staff_location, address_line1, postcode, department_name) values ('ST-003', 'Andrew', 'Brooks', 'Accommodation Manager', 'andrew.brooks@deakin.edu.au', '042287723', '0388723355', 'No', 'Victoria Hall', '15 Sample Road', '3023', 'Deakin Student Accommodation'); insert into staff_info (staff_id, first_name, last_name, postion, email, mobile, phone, is_advisor, staff_location, address_line1, postcode, department_name) values ('ST-004', 'Mark', 'BAnthony', 'Accommodation Manager', 'mark.anthony@deakin.edu.au', '0477885643', '0396547899', 'No', 'Victoria Hall', '23 John Street', '3321', 'Deakin Student Accommodation'); insert into staff_info (staff_id, first_name, last_name, postion, email, mobile, phone, is_advisor, staff_location, address_line1, postcode, department_name) values ('ST-005', 'Andrew', 'Smith', 'Unit Chair - SIT720', 'andrea.smith@deakin.edu.au', '0411223344', '0391145678', 'No', 'Burwood Campus', '15 Grey Street', '3331', 'School of IT'); insert into staff_info (staff_id, first_name, last_name, postion, email, mobile, phone, is_advisor, staff_location, address_line1, postcode, department_name) values ('ST-006', 'Katy', 'Shaw', 'Student Advisor', 'katy.shaw@deakin.edu.au', '0455667788', '0398982323', 'Yes', 'Geelong Campus', '20 Grey Street', '3331', 'Deakin Student Accommodation');

Place_details:
insert into place_details (place_id, monthly_rent, accommodation_type) values ('PL-001', '900.00', 'Unit');
insert into place_details (place_id, monthly_rent, accommodation_type) values ('PL-002', '900.00', 'Unit');
insert into place_details (place_id, monthly_rent, accommodation_type) values ('PL-003', '800.00', 'Victoria Hall');
insert into place_details (place_id, monthly_rent, accommodation_type) values ('PL-004', '800.00', 'Victoria Hall');
insert into place_details (place_id, monthly_rent, accommodation_type) values ('PL-005', '900.00', 'Unit');
insert into place_details (place_id, monthly_rent, accommodation_type) values ('PL-006', '500.00', 'Dorm');

Leases: insert into leases (lease_id, student_id, lease_start_date, lease_end_date, lease_term, place_id) values ('LID-005', 'SID-001', '2020-01-01', '2020-12-31', '12', 'PL-001'); insert into leases (lease_id, student_id, lease_start_date, lease_end_date, lease_term, place_id) values ('LID-001', 'SID-002', '2020-03-01', '2020-09-01', '6', 'PL-006'); insert into leases (lease_id, student_id, lease_start_date, lease_end_date, lease_term, place_id) values ('LID-002', 'SID-003', '2020-03-01', '2020-12-31', '9', 'PL-002'); insert into leases (lease_id, student_id, lease_start_date, lease_end_date, lease_term, place_id)

```
values ('LID-003', 'SID-004', '2020-01-01', '2020-03-01', '3', 'PL-003');
insert into leases (lease_id, student_id, lease_start_date, lease_end_date, lease_term, place_id)
values ('LID-007', 'SID-005', '2020-01-01', '2020-06-01', '6', 'PL-004');
insert into leases (lease_id, student_id, lease_start_date, lease_end_date, lease_term, place_id)
values ('LID-004', 'SID-006', '2020-01-01', '2020-12-31', '12', 'PL-005');
```

Unit_Details:

```
insert into unit_details(unit_id, address_line1, accommodation_type, unit_room_count, postcode)
values ('UN-001', '1/233 Burwood Hwy', 'Unit', 3, '3023');
insert into unit_details(unit_id, address_line1, accommodation_type, unit_room_count, postcode)
values ('UN-002', '2/233 Burwood Hwy', 'Unit', 4, '3023');
insert into unit_details(unit_id, address_line1, accommodation_type, unit_room_count, postcode)
values ('UN-003', '3/233 Burwood Hwy', 'Unit', 4, '3023');
insert into unit_details(unit_id, address_line1, accommodation_type, unit_room_count, postcode)
values ('UN-004', '4/233 Burwood Hwy', 'Unit', 5, '3023');
insert into unit_details(unit_id, address_line1, accommodation_type, unit_room_count, postcode)
values ('UN-005', '5/233 Burwood Hwy', 'Unit', 3, '3023');
insert into unit_details(unit_id, address_line1, accommodation_type, unit_room_count, postcode)
values ('UN-006', '6/233 Burwood Hwy', 'Unit', 3, '3023');
```

Unit_Room:

```
insert into unit_room (room_id, unit_id, place_id) values ('UNRM-101', 'UN-001', 'PL-001');
insert into unit_room (room_id, unit_id, place_id) values ('UNRM-102', 'UN-001', 'PL-002');
insert into unit_room (room_id, unit_id, place_id) values ('UNRM-103', 'UN-001', 'PL-005');
insert into unit_room (room_id, unit_id, place_id) values ('UNRM-201', 'UN-002', 'PL-007');
insert into unit_room (room_id, unit_id, place_id) values ('UNRM-202', 'UN-002', 'PL-008');
insert into unit_room (room_id, unit_id, place_id) values ('UNRM-203', 'UN-002', 'PL-009');
```

Hall Details:

```
insert into hall_details (hall_id, accommodation_type, hall_name, staff_id, address_line1, postcode, hall_phone)
values ('HL-001', 'Victoria Hall', 'Griffith', 'ST-002', '240 Burwood Hwy', '3023', '0388987787');
insert into hall_details (hall_id, accommodation_type, hall_name, staff_id, address_line1, postcode, hall_phone)
values ('HL-002', 'Victoria Hall', 'Medley', 'ST-003', '241 Burwood Hwy', '3023', '0388987798');
insert into hall_details (hall_id, accommodation_type, hall_name, staff_id, address_line1, postcode, hall_phone)
values ('HL-003', 'Victoria Hall', 'Xavier', 'ST-004', '242 Burwood Hwy', '3023', '0388987779');
insert into hall_details (hall_id, accommodation_type, hall_name, staff_id, address_line1, postcode, hall_phone)
```

```
values ('HL-004', 'Victoria Hall', 'Lowther', 'ST-121', '243 Burwood Hwy', '3023', '0388988879');
insert into hall_details (hall_id, accommodation_type, hall_name, staff_id, address_line1, postcode, hall_phone)
values ('HL-005', 'Victoria Hall', 'Bowden', 'ST-223', '244 Burwood Hwy', '3023', '0388966679');
insert into hall_details (hall_id, accommodation_type, hall_name, staff_id, address_line1, postcode, hall_phone)
values ('HL-006', 'Victoria Hall', 'Union', 'ST-527', '245 Burwood Hwy', '3023', '0388922279');
```

Hall Room:

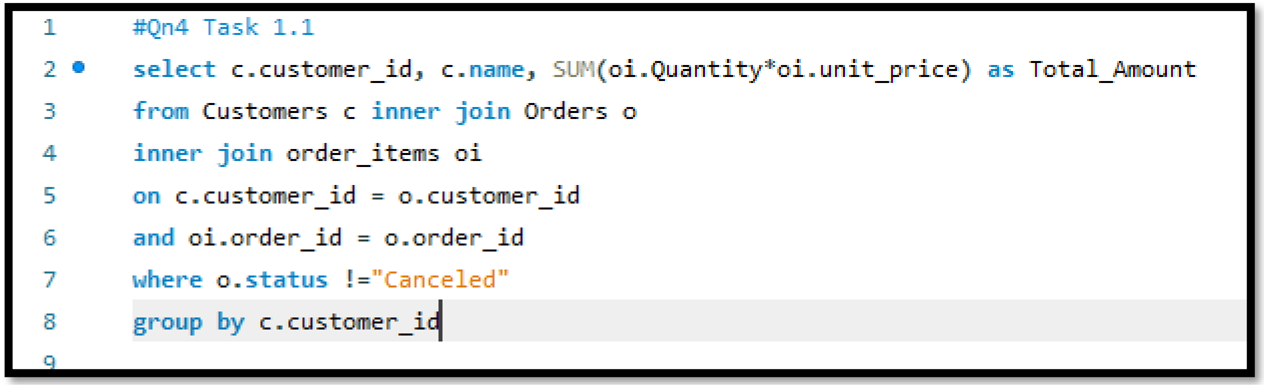
```
insert into hall_room (room_id, place_id, hall_id) values ('HLRM-101', 'PL-003', 'HL-001');
insert into hall_room (room_id, place_id, hall_id) values ('HLRM-102', 'PL-004', 'HL-001');
insert into hall_room (room_id, place_id, hall_id) values ('HLRM-201', 'PL-015', 'HL-002');
insert into hall_room (room_id, place_id, hall_id) values ('HLRM-202', 'PL-016', 'HL-002');
insert into hall_room (room_id, place_id, hall_id) values ('HLRM-203', 'PL-221', 'HL-002');
insert into hall_room (room_id, place_id, hall_id) values ('HLRM-501', 'PL-321', 'HL-005');
insert into hall_room (room_id, place_id, hall_id) values ('HLRM-502', 'PL-447', 'HL-005');
```

Question 4 – Product Database

Task 1.1 – Write the SQL query to list the Customer Name and total purchase (amount) in all orders.

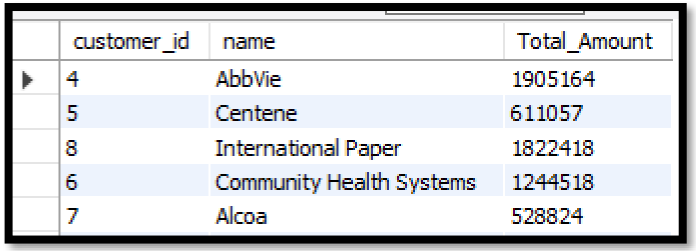
This query returned a total of 40 rows

Query Logic

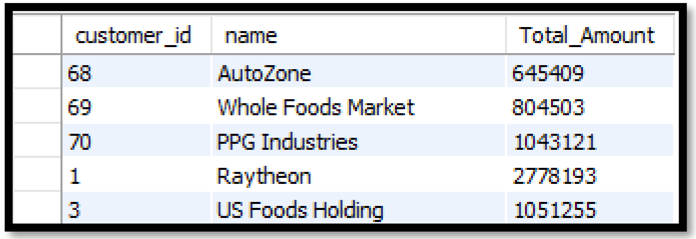


```sql
#Qn4 Task 1.1
select c.customer_id, c.name, SUM(oi.Quantity*oi.unit_price) as Total_Amount
from Customers c inner join Orders o
inner join order_items oi
on c.customer_id = o.customer_id
and oi.order_id = o.order_id
where o.status !="Canceled"
group by c.customer_id
```

Query Results



customer_id	name	Total_Amount
4	AbbVie	1905164
5	Centene	611057
8	International Paper	1822418
6	Community Health Systems	1244518
7	Alcoa	528824

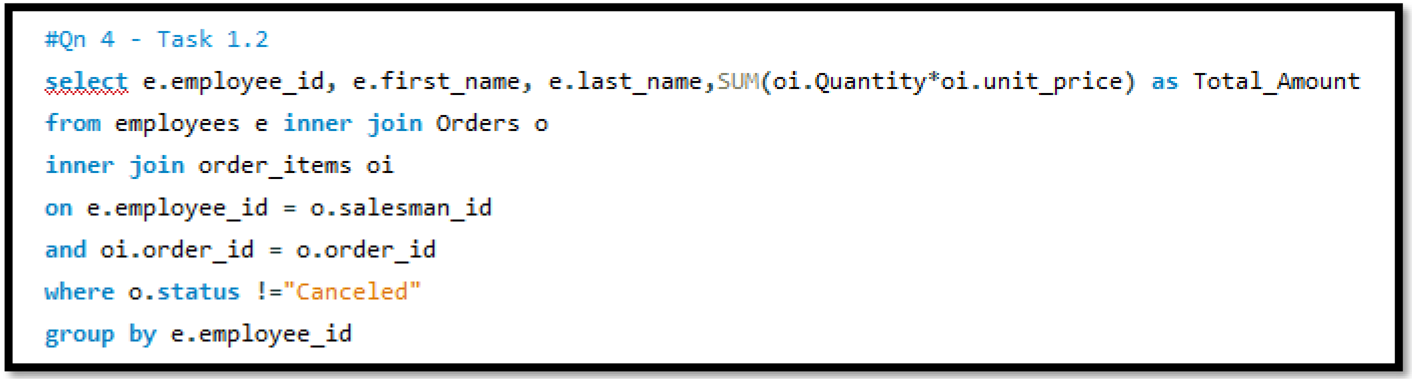


customer_id	name	Total_Amount
68	AutoZone	645409
69	Whole Foods Market	804503
70	PPG Industries	1043121
1	Raytheon	2778193
3	US Foods Holding	1051255

Task 1.2 – Write the SQL query to find total sale by each employee.

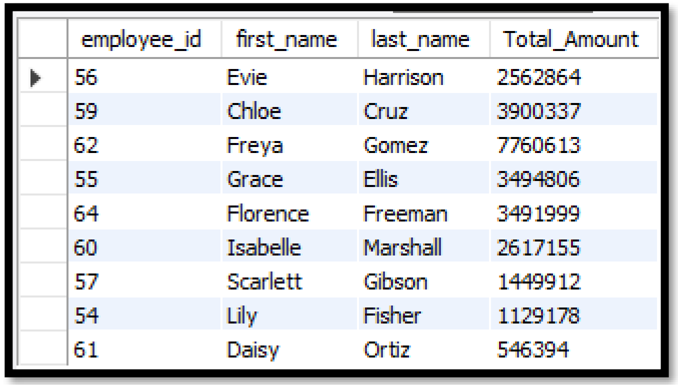
This query returned 9 rows

Query Logic



```
#Qn 4 - Task 1.2
select e.employee_id, e.first_name, e.last_name,SUM(oi.Quantity*oi.unit_price) as Total_Amount
from employees e inner join Orders o
inner join order_items oi
on e.employee_id = o.salesman_id
and oi.order_id = o.order_id
where o.status !="Canceled"
group by e.employee_id
```

Query Results

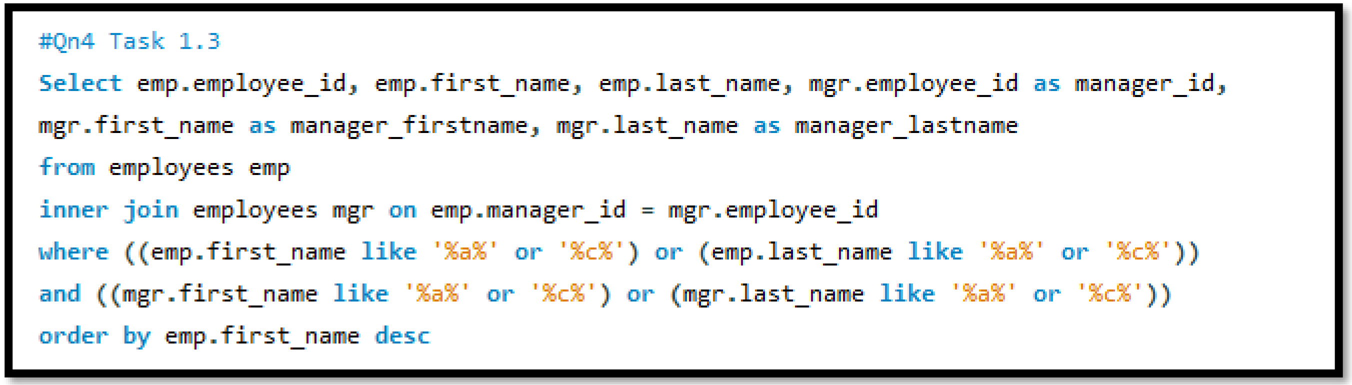


employee_id	first_name	last_name	Total_Amount
56	Evie	Harrison	2562864
59	Chloe	Cruz	3900337
62	Freya	Gomez	7760613
55	Grace	Ellis	3494806
64	Florence	Freeman	3491999
60	Isabelle	Marshall	2617155
57	Scarlett	Gibson	1449912
54	Lily	Fisher	1129178
61	Daisy	Ortiz	546394

Task 1.3 – Write the SQL query to list all employee who have the sequential letters ‘c’ or ‘a’ in their name and their manager name. List must include the employee ID, names and ordered by their names in ascending.

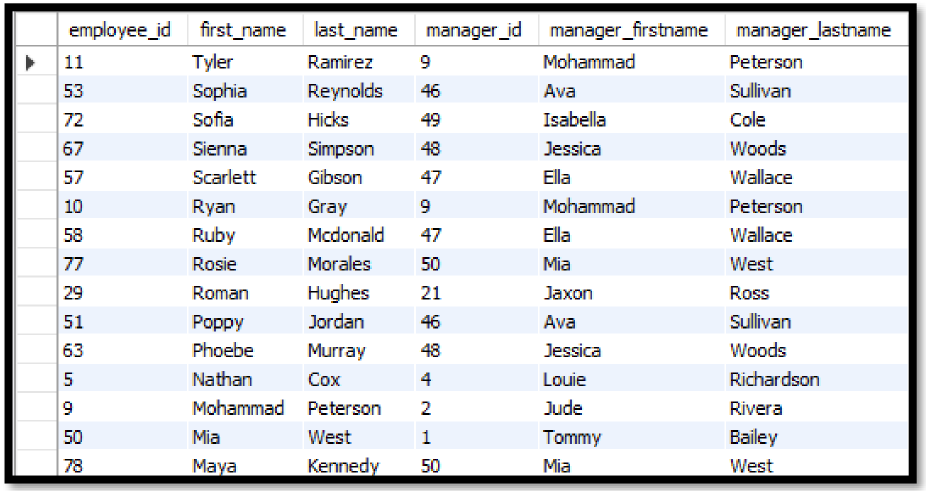
This query returned 71 rows

Query Logic

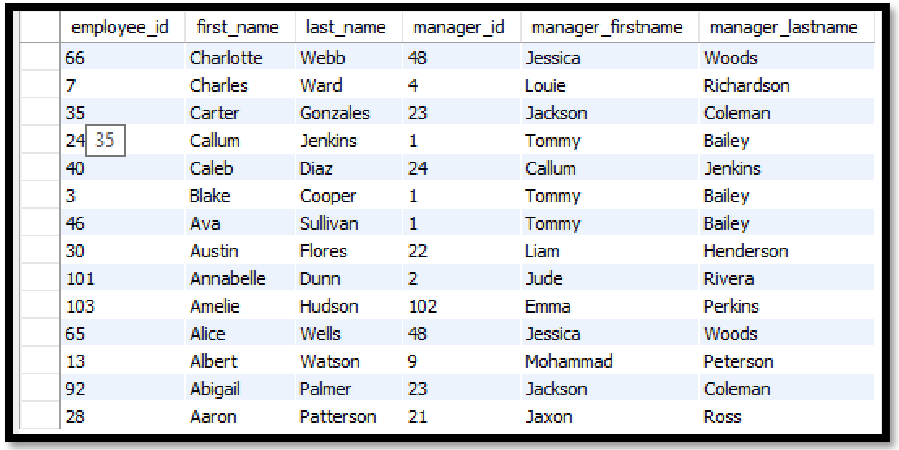


```
#Qn4 Task 1.3
Select emp.employee_id, emp.first_name, emp.last_name, mgr.employee_id as manager_id,
mgr.first_name as manager_firstname, mgr.last_name as manager_lastname
from employees emp
inner join employees mgr on emp.manager_id = mgr.employee_id
where ((emp.first_name like '%a%' or '%c%') or (emp.last_name like '%a%' or '%c%'))
and ((mgr.first_name like '%a%' or '%c%') or (mgr.last_name like '%a%' or '%c%'))
order by emp.first_name desc
```

Query Results



employee_id	first_name	last_name	manager_id	manager_firstname	manager_lastname
11	Tyler	Ramirez	9	Mohammad	Peterson
53	Sophia	Reynolds	46	Ava	Sullivan
72	Sofia	Hicks	49	Isabella	Cole
67	Sienna	Simpson	48	Jessica	Woods
57	Scarlett	Gibson	47	Ella	Wallace
10	Ryan	Gray	9	Mohammad	Peterson
58	Ruby	Mcdonald	47	Ella	Wallace
77	Rosie	Morales	50	Mia	West
29	Roman	Hughes	21	Jaxon	Ross
51	Poppy	Jordan	46	Ava	Sullivan
63	Phoebe	Murray	48	Jessica	Woods
5	Nathan	Cox	4	Louie	Richardson
9	Mohammad	Peterson	2	Jude	Rivera
50	Mia	West	1	Tommy	Bailey
78	Maya	Kennedy	50	Mia	West

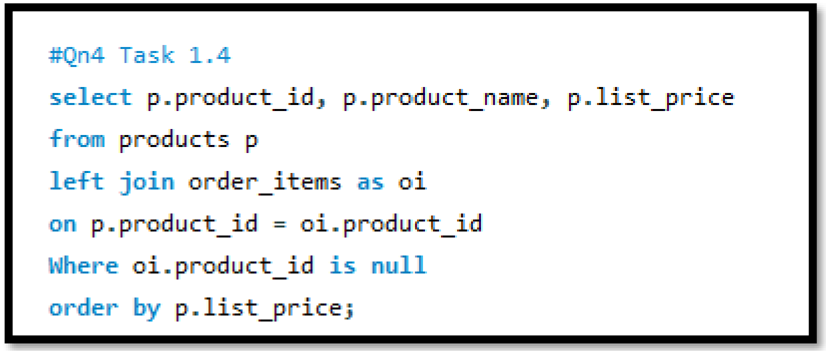


employee_id	first_name	last_name	manager_id	manager_firstname	manager_lastname
66	Charlotte	Webb	48	Jessica	Woods
7	Charles	Ward	4	Louie	Richardson
35	Carter	Gonzales	23	Jackson	Coleman
24 35	Callum	Jenkins	1	Tommy	Bailey
40	Caleb	Diaz	24	Callum	Jenkins
3	Blake	Cooper	1	Tommy	Bailey
46	Ava	Sullivan	1	Tommy	Bailey
30	Austin	Flores	22	Liam	Henderson
101	Annabelle	Dunn	2	Jude	Rivera
103	Amelie	Hudson	102	Emma	Perkins
65	Alice	Wells	48	Jessica	Woods
13	Albert	Watson	9	Mohammad	Peterson
92	Abigail	Palmer	23	Jackson	Coleman
28	Aaron	Patterson	21	Jaxon	Ross

Task 1.4 – Write the SQL query to list all products' ID, Name and price where the products haven't been purchased by any customer in the database. The list must be ordered by the product price.

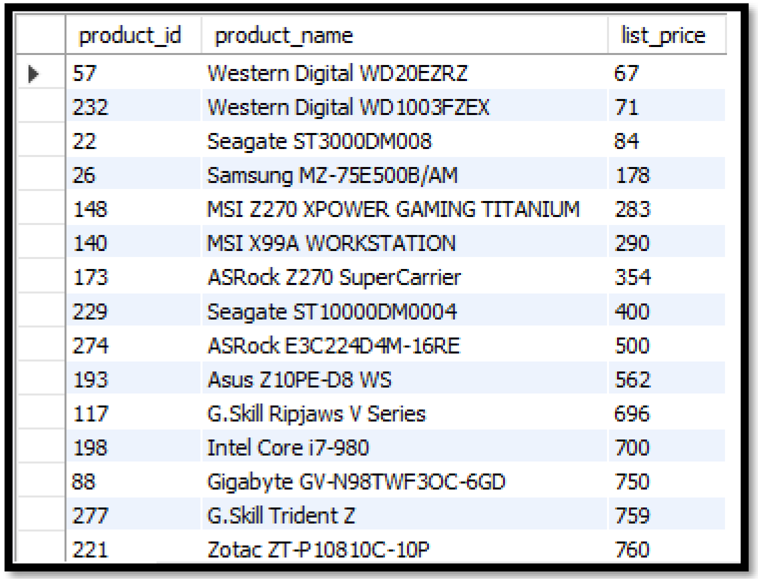
This query returned 32 rows

Query Logic

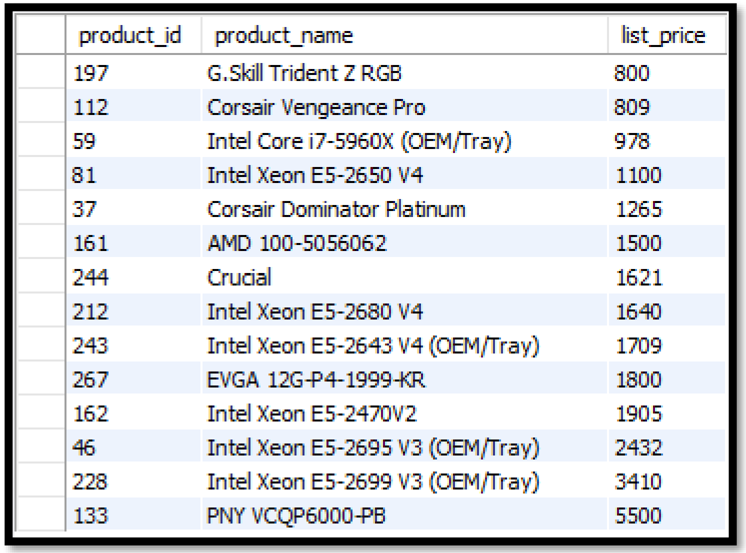


```
#Qn4 Task 1.4
select p.product_id, p.product_name, p.list_price
from products p
left join order_items as oi
on p.product_id = oi.product_id
Where oi.product_id is null
order by p.list_price;
```

Query Results



	product_id	product_name	list_price
▶	57	Western Digital WD20EZRZ	67
	232	Western Digital WD1003FZEX	71
	22	Seagate ST3000DM008	84
	26	Samsung MZ-75E500B/AM	178
	148	MSI Z270 XPOWER GAMING TITANIUM	283
	140	MSI X99A WORKSTATION	290
	173	ASRock Z270 SuperCarrier	354
	229	Seagate ST10000DM0004	400
	274	ASRock E3C224D4M-16RE	500
	193	Asus Z10PE-D8 WS	562
	117	G.Skill Ripjaws V Series	696
	198	Intel Core i7-980	700
	88	Gigabyte GV-N98TWF3OC-6GD	750
	277	G.Skill Trident Z	759
	221	Zotac ZT-P10810C-10P	760

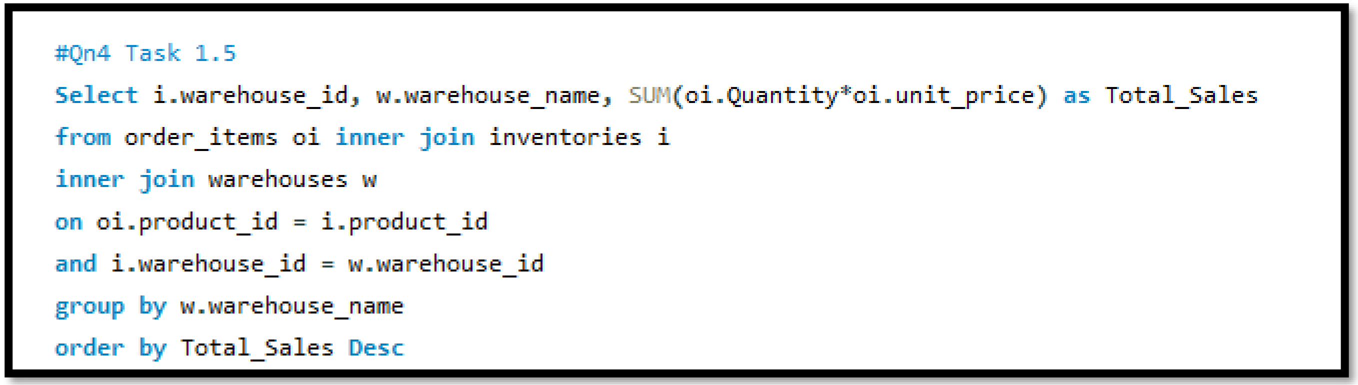


	product_id	product_name	list_price
	197	G.Skill Trident Z RGB	800
	112	Corsair Vengeance Pro	809
	59	Intel Core i7-5960X (OEM/Tray)	978
	81	Intel Xeon E5-2650 V4	1100
	37	Corsair Dominator Platinum	1265
	161	AMD 100-5056062	1500
	244	Crucial	1621
	212	Intel Xeon E5-2680 V4	1640
	243	Intel Xeon E5-2643 V4 (OEM/Tray)	1709
	267	EVGA 12G-P4-1999-KR	1800
	162	Intel Xeon E5-2470V2	1905
	46	Intel Xeon E5-2695 V3 (OEM/Tray)	2432
	228	Intel Xeon E5-2699 V3 (OEM/Tray)	3410
	133	PNY VCQP6000-PB	5500

Task 1.5 – Write the SQL query to list all the warehouses and their total sales. Here, given a product, the total sale of the product is calculated by the sold quantity of the product and its unit price. The list must be ordered by the total sales in the descending.

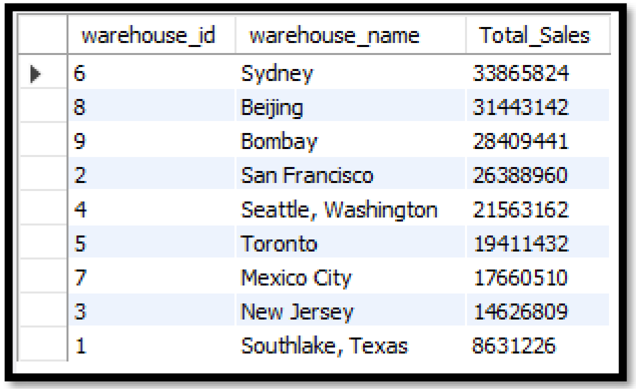
This query returned 9 rows

Query Logic



```
#Qn4 Task 1.5
Select i.warehouse_id, w.warehouse_name, SUM(oi.Quantity*oi.unit_price) as Total_Sales
from order_items oi inner join inventories i
inner join warehouses w
on oi.product_id = i.product_id
and i.warehouse_id = w.warehouse_id
group by w.warehouse_name
order by Total_Sales Desc
```

Query Results

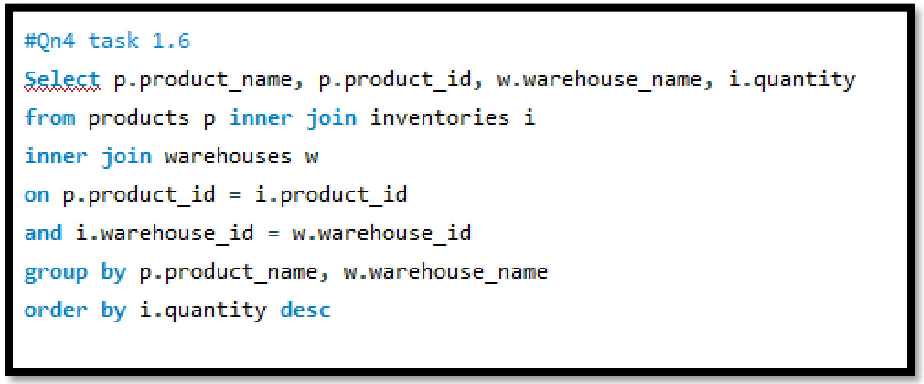


warehouse_id	warehouse_name	Total_Sales
6	Sydney	33865824
8	Beijing	31443142
9	Bombay	28409441
2	San Francisco	26388960
4	Seattle, Washington	21563162
5	Toronto	19411432
7	Mexico City	17660510
3	New Jersey	14626809
1	Southlake, Texas	8631226

Task 1.6 – Write the SQL query to list the product and available stock in all warehouses. The list must be sorted by the quantity of available product in the descending order.

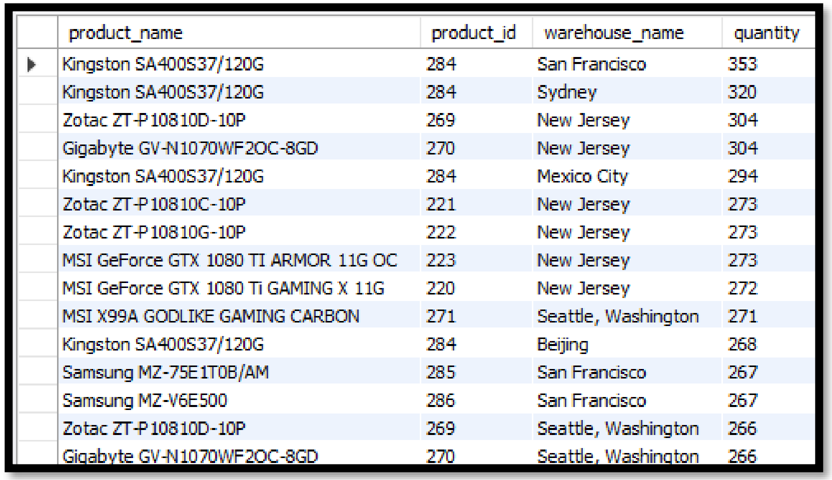
This query returned 949 rows

Query Logic

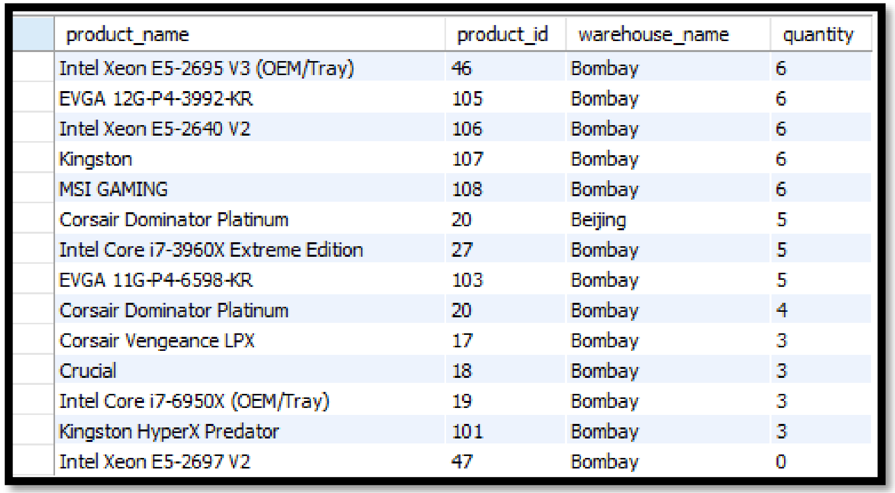


```
#Qn4 task 1.6
Select p.product_name, p.product_id, w.warehouse_name, i.quantity
from products p inner join inventories i
inner join warehouses w
on p.product_id = i.product_id
and i.warehouse_id = w.warehouse_id
group by p.product_name, w.warehouse_name
order by i.quantity desc
```

Query Results



product_name	product_id	warehouse_name	quantity
Kingston SA400S37/120G	284	San Francisco	353
Kingston SA400S37/120G	284	Sydney	320
Zotac ZT-P10810D-10P	269	New Jersey	304
Gigabyte GV-N1070WF2OC-8GD	270	New Jersey	304
Kingston SA400S37/120G	284	Mexico City	294
Zotac ZT-P10810C-10P	221	New Jersey	273
Zotac ZT-P10810G-10P	222	New Jersey	273
MSI GeForce GTX 1080 TI ARMOR 11G OC	223	New Jersey	273
MSI GeForce GTX 1080 Ti GAMING X 11G	220	New Jersey	272
MSI X99A GODLIKE GAMING CARBON	271	Seattle, Washington	271
Kingston SA400S37/120G	284	Beijing	268
Samsung MZ-75E1T0B/AM	285	San Francisco	267
Samsung MZ-V6E500	286	San Francisco	267
Zotac ZT-P10810D-10P	269	Seattle, Washington	266
Gigabyte GV-N1070WF2OC-8GD	270	Seattle, Washington	266



product_name	product_id	warehouse_name	quantity
Intel Xeon E5-2695 V3 (OEM/Tray)	46	Bombay	6
EVGA 12G-P4-3992-KR	105	Bombay	6
Intel Xeon E5-2640 V2	106	Bombay	6
Kingston	107	Bombay	6
MSI GAMING	108	Bombay	6
Corsair Dominator Platinum	20	Beijing	5
Intel Core i7-3960X Extreme Edition	27	Bombay	5
EVGA 11G-P4-6598-KR	103	Bombay	5
Corsair Dominator Platinum	20	Bombay	4
Corsair Vengeance LPX	17	Bombay	3
Crucial	18	Bombay	3
Intel Core i7-6950X (OEM/Tray)	19	Bombay	3
Kingston HyperX Predator	101	Bombay	3
Intel Xeon E5-2697 V2	47	Bombay	0