

Team 5
IS 680 Group project
Rental-zone rentals
Chakrapani Suresh, Deepti Rathore, Priyanka Katre, Ansuya Patel

Introduction:

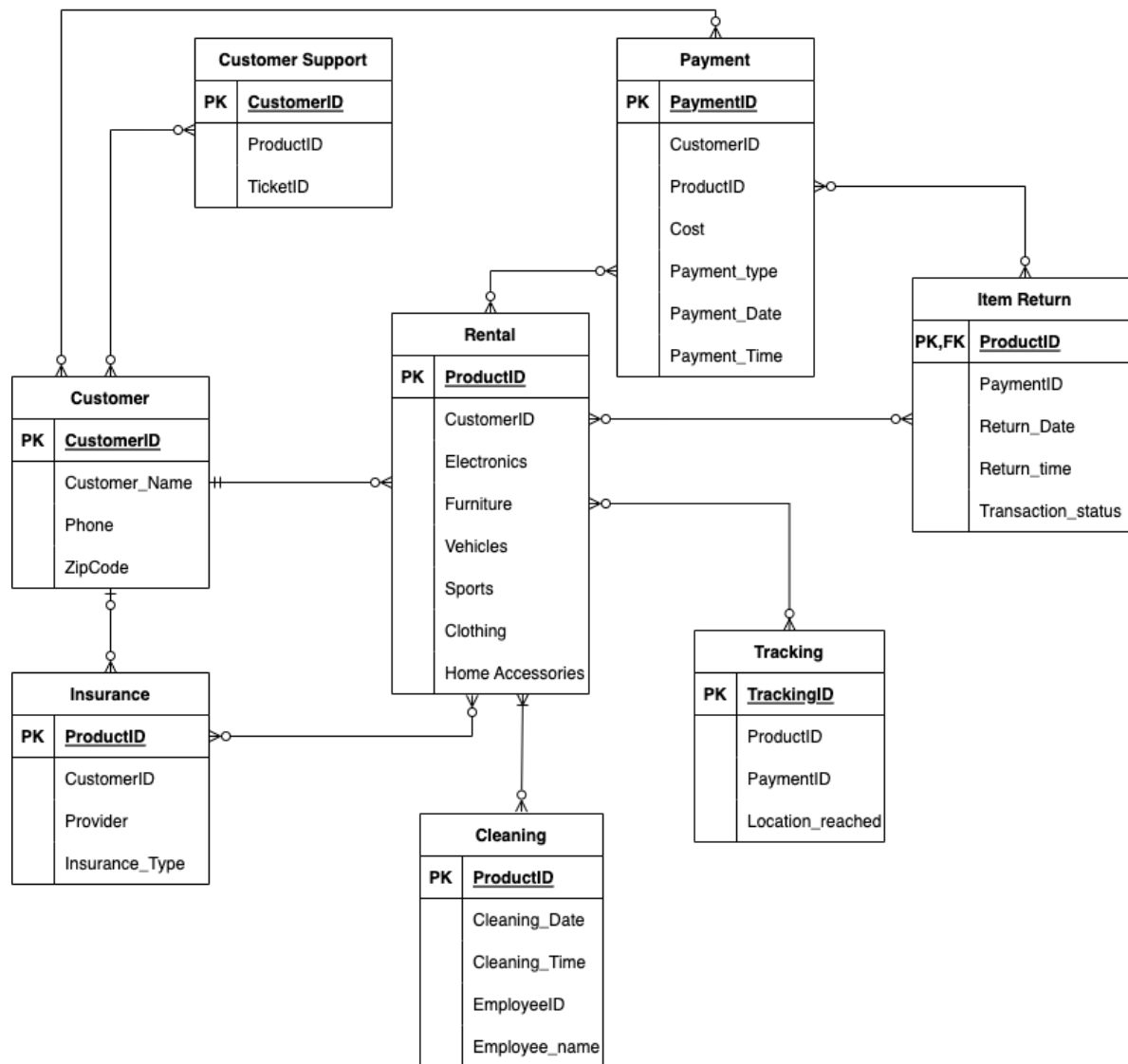
Classified industries have come a long way with rapid digitalization of records, back in the days people wishing to sell or rent things they don't need post advertisements in their local classifieds in hopes subscribers will see and call them to make a deal. Although this was the primary way of customer-to-customer sales in olden days, the process is slow and tedious, most of the time the adverts getting lost in the midst of hundreds of others.

With digital technologies like Search engine optimization (SEO) , indexing and targeted advertisements, classifieds have been revolutionized by companies like craigslist. Advertisements were made more reachable, targeted to customers of interest, hence both parties had greater convenience and success in getting their deals done.

The primary issue with classifieds is that they deal with huge amounts of data, ranging from product info, customer details, payment, shipment tracking and other metadata like customer interests, location etc. Hence it is crucial and tedious to store, optimize and retrieve so much datas. We take the case of our rental-zone which is a online rental platform for everything imaginable, users can rent anything they want, the company will give services like insurance, shipment, payment security , hence as discussed above will need to track and keep tabs on several aspects of the services, the aim of this project is aid efficient storage, retrieval and usage of data involved in rental-zone. A system that will index and provide good reach of posts to the right customers, manage shipments, keep tab on transactions and return of the item, hence providing a seamless platform to do online customer to customer business.

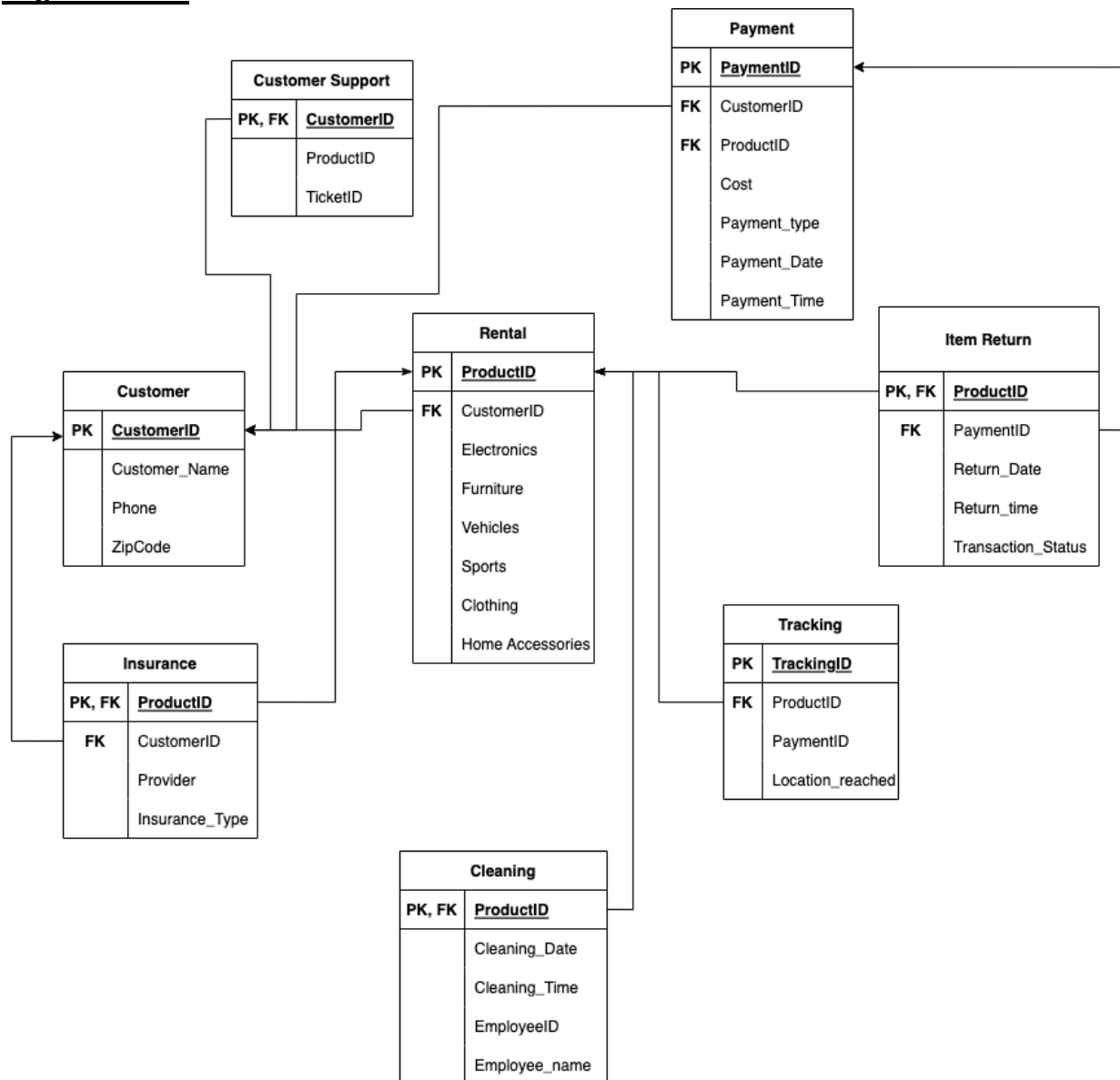
In this design for the database, we have included a conceptual model, logical model, data dictionary and the queries used to design the database.

Conceptual model:



In our conceptual diagram we include eight entities all having the proper defined relationship. From the multiple relation types, we used mostly the many to many , optional and the mandatory ones to create and show the link between the entities. In our conceptual diagram mostly, entities are connected to rental entities. For in our case the many to many we use at multiple places, so the one instance is the connection between rental and payment is many to many and it is optional from both sides. Which means that one rental can have 0 more than 1 payment and vice versa in case of payment.

Logical Model:



The above is the logical diagram for our rental-zone project. We came out with a logical diagram because we can use the foreign and primary key here which is not present in the conceptual diagram. In a logical diagram, we are working on attributes, whereas in a conceptual diagram, we are using entities. In our logical diagram, the rental entity is clustering where all the entities are linked to the rental entities. In order to receive the payment, the rental has to know which rental got on rent. It has the category ID to identify.

Data dictionary:

Rental Table:

This table contains information about all the types of things that can be rented.

Column	Data type	Description	Required	PK or FK
CustomerID	INT	Unique ID for the customer	Y	FK
ProductID	INT	Unique ID for the product	Y	
Electronics	VARCHAR	Electronic item type	Y	
Furniture	VARCHAR	Furniture type/name	Y	
Vehicles	VARCHAR	Vehicle type/name	Y	
Sports	VARCHAR	Sports item name	Y	
Clothing	VARCHAR	Clothing type	Y	

Customer Table:

This table will give the details of all the customers who rented an item.

Column	Data type	Description	Required	PK or FK
CustomerID	INT	Unique ID for the customer	Y	PK
Customer_Name	VARCHAR	Customer name	Y	
Phone	DOUBLE	Customer's phone number	Y	
ZipCode	INT	Zip code of the customer's address	Y	

Customer Support:

This table will keep track of the customer support given to customers and products that they wanted support for.

Column	Data type	Description	Required	PK or FK
CustomerID	INT	Unique ID for the customer	Y	PK, FK
ProductID	INT	Unique ID for the product	Y	
TicketID	INT	Unique TicketID given to a raised request.	Y	

Insurance table:

This table will keep track of insured products and which provider is giving the insurance for the product.

Column	Data type	Description	Required	PK or FK
ProductID	INT	Unique ID for the product	Y	PK,FK
CustomerID	INT	Unique ID for the customer	Y	FK
Provider	VARCHAR	Insurance provider	Y	
Insurance_Type	VARCHAR	Type of insurance being provide	Y	

Tracking Table:

This table will help in tracking items which were shipped to some other place.

Column	Data type	Description	Required	PK or FK
TrackingID	INT	Unique ID given for tracking every item	Y	PK
ProductID	INT	Unique ID for the product	Y	FK
PaymentID	INT	Payment method used for purchasing the item	Y	
Location_reached	VARCHAR	Current location where the item has reached	Y	

Payment Table:

Payment table will keep details of all the transactions that were made to rent an item.

Column	Data type	Description	Required	PK or FK
PaymentID	INT	Payment method used for purchasing the item	Y	PK
CustomerID	INT	Unique ID for the customer	Y	FK
ProductID	INT	Unique ID for the product	Y	FK
Cost	DECIMAL	Cost of each item	Y	
Payment_type	VARCHAR	Mode of payment used	Y	
Payment_Date	DATE	Date when the payment was done	Y	
Payment_Time	TIME	Time when the payment was done	Y	

Item Return Table:

This table will help keep track of items that are returned.

Column	Data type	Description	Required	PK or FK
ProductID	INT	Unique ID for the product	Y	PK, FK
PaymentID	INT	Payment method used for purchasing the item	Y	FK
Return_Date	DATE	Date when the item was returned	Y	
Return_time	TIME	Time when the item was returned	Y	
Transaction_Status	VARCHAR	Transaction status will be recorded	Y	

Cleaning Table:

Cleaning table will help in keeping track of which item is cleaned and who cleaned it.

Column	Data type	Description	Required	PK or FK
ProductID	INT	Unique ID for the product	Y	PK, FK
Cleaning_Date	DATE	Date when the item has been cleaned	Y	
Cleaning_Time	TIME	Time when the item has been cleaned	Y	
EmployeeID	CHAR	EmployeeID who cleaned the item	Y	
Employee_name	VARCHAR	Employee name who cleaned the product	Y	

Why we need the database:

Databases support good data access because: large volumes of data can be stored in one place. Multiple users can read and modify the data at the same time. Databases are searchable and sortable, so the data you need can be found quickly and easily.

In our rental-zone project we have huge data to maintain . We need the database for example we have the 8 entities, and they have a number of attributes which are linked with different entities or interconnect for their functionality to perform.

In our rental-zone project the most connected entity is Rental and the data from rental to any other entity has to be taken care of very carefully. Like in rental there are so many categories which are connected to payment . We have to keep the track through which rental we are getting payment and what type of payment it is like it is from return or renting or else it is from insurance. So in our project the database is very important accept which help us to track all the functionality and data from where it is going and from it is coming.

Why we need the SQL Queries:

In the Rental-zone project till now we have created the actual design of the database. Now to make it functionality work we have implemented the database in sql. In SQL varieties we have chosen the MYSQL for our project. The implementation of the database is started with the first step which is creating the table . In our project we have 13 entities so we created 13 tables in MYSQL with the correct data type occupying each column .When we are creating our tables, it is important to also establish the relationship that an entity might have with another entity, which is portrayed by primary keys and foreign keys.

In the second step we are implementing the actual database into MYSQL. For that we are using the table data of each entity . For example, in the table of payment the data related to payment has to be stored using the relationship with another entity .

Creating Customer Table:

```
CREATE TABLE Customer(  
  `CustomerID` INT NOT NULL,  
  `CustomerName` VARCHAR(45) NOT NULL,  
  `Phone` DOUBLE NULL,  
  `Zipcode` INT NOT NULL,  
  PRIMARY KEY (`CustomerID`));
```

Creating Customer Support Table:

```
CREATE TABLE Customer Support (  
  `CustomerID` INT NOT NULL,  
  `ProductID` INT NOT NULL,  
  `TicketID` INT NOT NULL,  
  PRIMARY KEY (`CustomerID`));
```

Creating Payment Table:

```
CREATE TABLE Payment (  
  `PaymentID` INT NOT NULL,  
  `CustomerID` INT NULL,  
  `ProductID` INT NULL,  
  `Payment_type` VARCHAR(45) NULL,  
  `Payment_Date` DATE NULL,  
  `Payment_Time` TIME NULL,  
  PRIMARY KEY (`PaymentID`));
```

Creating ItemReturn Table:

```
CREATE TABLE ItemReturn (  
  `ProductID` INT NOT NULL,  
  `PaymentID` INT NOT NULL,  
  `Return_Date` DATE NOT NULL,  
  `Return_time` TIME NOT NULL,  
  `Transaction_Status` VARCHAR(45) NOT NULL,  
  PRIMARY KEY (`ProductID`));
```

Creating Insurance Table:

```
CREATE TABLE Insurance(  
  `ProductID` INT NOT NULL,  
  `CustomerID` INT NULL,  
  `Provider` VARCHAR(45) NULL,  
  `Insurance_Type` VARCHAR(45) NULL,  
  PRIMARY KEY (`ProductID`));
```

Creating Tracking Table:

```
CREATE TABLE Tracking Table(  
  `TrackingID` INT NOT NULL,  
  `ProductID` INT NOT NULL,  
  `PaymentID` INT NOT NULL,  
  `Location_reached` VARCHAR(45) NOT NULL,
```

`Tracking Tablecol` VARCHAR(45) NOT NULL,
PRIMARY KEY (`TrackingID`));

Inserting data into Rental Table:

```
INSERT INTO Rental(CategoryID, CustomerID, ProductID, Electronics)
VALUES (01, 111, 1001, 'iphone 13 pro');
INSERT INTO Rental(CategoryID, CustomerID, ProductID, Furniture)
VALUES (02, 112, 1002, 'Folding chair');
INSERT INTO Rental(CategoryID, CustomerID, ProductID, Vehicles)
VALUES (03, 113, 1003, 'Kia K5');
INSERT INTO Rental(CategoryID, CustomerID, ProductID, Sports)
VALUES (04, 114, 1004, 'Basket Ball');
INSERT INTO Rental(CategoryID, CustomerID, ProductID, Clothing)
VALUES (05, 115, 1005, 'Bell bottom Trouser');
INSERT INTO Rental(CategoryID, CustomerID, ProductID, Home)
VALUES (06, 116, 1006, 'Washing Machine');
INSERT INTO Rental(CategoryID, CustomerID, ProductID, Electronics, Vehicle)
VALUES (07, 117, 1007, 'Samsung Galaxy S6', 'Volkswagen polo');
```

CustomerID	ProductID	Electronics	Furniture	Vehicles	Sports	Clothing	Home
111	1001	iphone 13 pro					
112	1002		Folding chair				
113	1003			Kia K5			
114	1004				Basket Ball		
115	1005					Bell bottom Trouser	
116	1006						Washing Machine
117	1007	Samsung Galaxy S6					
117	1008			Volkswagen polo			

Time	Action	Response
225 18:02:16	SELECT * FROM IS680Project..Tracking Table LIMIT 0, 1000	3 row(s) returned

Inserting data into Payment Table:

```
INSERT INTO
Payment(PaymentID, CustomerID, ProductID, Cost, Payment_type, Payment_Date, Payme
nt_Time)
VALUES (10001, 111, 1001, 400, 'credit card', '2022-02-02', '13:09:10');
```

INSERT INTO

Payment(PaymentID, CustomerID, ProductID, Cost, Payment_type, Payment_Date, Payment_Time)

VALUES (10002, 112, 1002, 39, 'credit card', '2022-02-03', '10:01:10');

INSERT INTO

Payment(PaymentID, CustomerID, ProductID, Cost, Payment_type, Payment_Date, Payment_Time)

VALUES (10003, 113, 1003, 20000, 'credit card', '2022-01-04', '12:10:10');

INSERT INTO

Payment(PaymentID, CustomerID, ProductID, Cost, Payment_type, Payment_Date, Payment_Time)

VALUES (10004, 114, 1004, 50, 'COD', '2022-02-01', '11:03:10');

INSERT INTO

Payment(PaymentID, CustomerID, ProductID, Cost, Payment_type, Payment_Date, Payment_Time)

VALUES (10005, 115, 1005, 15, 'credit card', '2022-03-30', '15:02:10');

INSERT INTO

Payment(PaymentID, CustomerID, ProductID, Cost, Payment_type, Payment_Date, Payment_Time)

VALUES (10006, 116, 1006, 300, 'credit card', '2022-04-24', '11:05:10');

INSERT INTO

Payment(PaymentID, CustomerID, ProductID, Cost, Payment_type, Payment_Date, Payment_Time)

VALUES (10007, 111, 1001, 25000, 'Financing', '2022-04-02', '13:01:10');

PaymentID	CustomerID	ProductID	Cost	Payment_type	Payment_Date	Payment_Time
10001	111	1001	400	credit card	2022-02-02	13:09:10
10002	112	1002	39	credit card	2022-02-03	10:01:10
10003	113	1003	20000	credit card	2022-01-04	12:10:10
10004	114	1004	50	COD	2022-02-01	11:03:10
10005	115	1005	15	credit card	2022-03-30	15:02:10
10006	116	1006	300	credit card	2022-04-24	11:05:10
10007	111	1001	25000	Financing	2022-04-02	13:01:10

Payment 1

Action Output

Time	Action	Response
225 18:02:16	SELECT * FROM IS680Project..Tracking Table LIMIT 0, 1000	3 row(s) returned

Inserting data into Insurance Table:

INSERT INTO Insurance(`ProductID`, `CustomerID`, `Provider`, `Insurance_Type`)

VALUES ('1003', '113', 'AAA', 'Car Insurance');

```

INSERT INTO Insurance(`ProductID`, `CustomerID`, `Provider`, `Insurance_Type`)
VALUES ('1001', '111', 'AppleCare', 'Phone insurance');
INSERT INTO Insurance(`ProductID`, `CustomerID`, `Insurance_Type`) VALUES
('1006', '116', 'Home appliance insurance');

```

Result Grid

ProductID	CustomerID	Provider	Insurance_Type
1001	111	Apple Care	Phone Insurance
1003	113	AAA	Car Insurance
1006	116	Choice home warranty	Home appliance insurance
HULL	HULL	HULL	HULL

Insurance 1

Action Output

Time	Action	Response	Duration / Fetch Time
225 18:02:16	SELECT * FROM IS680Project."Insurance Table" LIMIT 0, 1000	3 row(s) returned	0.00057 sec / 0.0000...

Query Completed

Inserting data into Tracking Table:

```

INSERT INTO Tracking Table(`TrackingID`, `ProductID`, `PaymentID`,
`Location_reached`) VALUES ('1400', '1004', '10004', 'Irvine');
INSERT INTO Tracking Table (`TrackingID`, `ProductID`, `PaymentID`,
`Location_reached`) VALUES ('1200', '1002', '10002', 'Los Angeles');
INSERT INTO Tracking Table(`TrackingID`, `ProductID`, `PaymentID`,
`Location_reached`) VALUES ('1500', '1005', '10005', 'Long Beach');

```

Result Grid

TrackingID	ProductID	PaymentID	Location_reach...
1200	1002	10002	Los Angeles
1400	1004	10004	Irvine
1500	1005	10005	Long Beach
HULL	HULL	HULL	HULL

Tracking Table 1

Action Output

Time	Action	Response
225 18:02:16	SELECT * FROM IS680Project."Tracking Table" LIMIT 0, 1000	3 row(s) returned

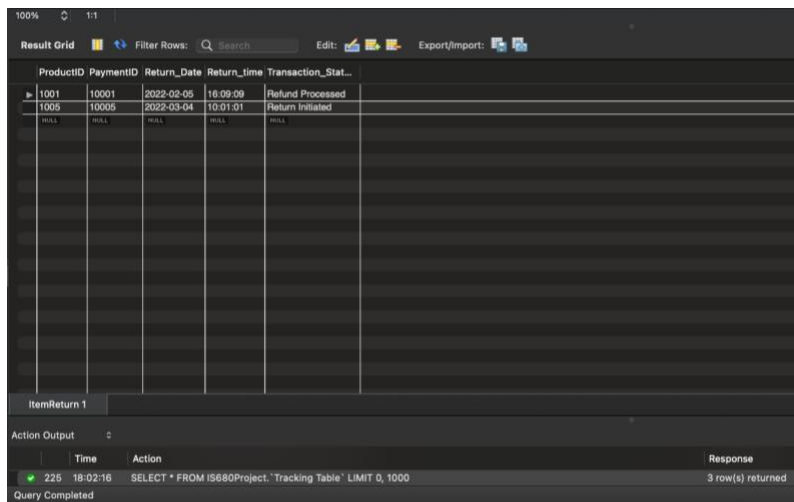
Inserting data into ItemReturn Table:

INSERT INTO

```
ItemReturn(ProductID,PaymentID,Return_Date,Return_time,Transaction_Status)
VALUES(1001,10001,'2022-02-05','16:09:09','Refund Processed');
```

INSERT INTO

```
ItemReturn(ProductID,PaymentID,Return_Date,Return_time,Transaction_Status)
VALUES(1005,10005,'2022-03-04','10:01:01','Return Initiated');
```



The screenshot shows a database management interface. At the top, there's a 'Result Grid' with a search bar and 'Filter Rows' button. Below it, a table with columns: ProductID, PaymentID, Return_Date, Return_time, Transaction_Status. The table contains two rows of data. Below the table, there's a tab labeled 'ItemReturn 1'. At the bottom, there's an 'Action Output' panel showing a query execution log.

ProductID	PaymentID	Return_Date	Return_time	Transaction_Status
1001	10001	2022-02-05	16:09:09	Refund Processed
1005	10005	2022-03-04	10:01:01	Return Initiated

ItemReturn 1

Action Output

Time	Action	Response
225 18:02:16	SELECT * FROM IS680Project.'Tracking Table' LIMIT 0, 1000	3 row(s) returned

Query Completed

Inserting data into Cleaning Table:

INSERT INTO

```
Cleaning(ProductID,Cleaning_Date,Cleaning_Time,EmployeeID,Employee_name)
VALUES (1003,'2022-05-02','11:00:20',E001,'Jon Doe');
```

INSERT INTO

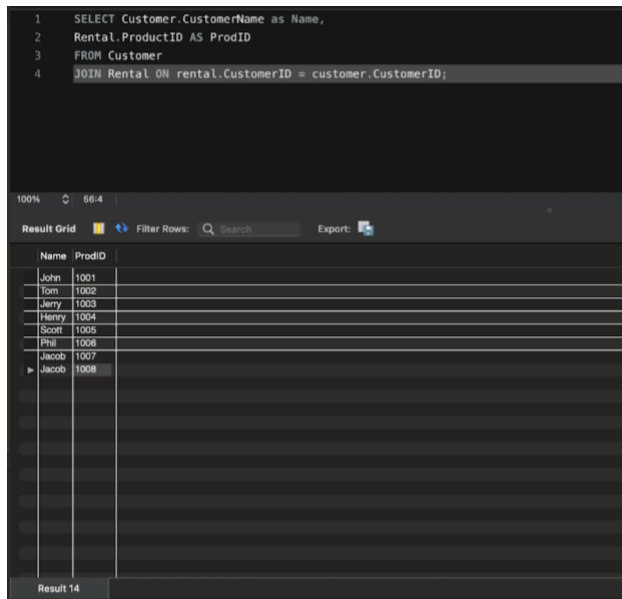
```
Cleaning(ProductID,Cleaning_Date,Cleaning_Time,EmployeeID,Employee_name)
VALUES (1007,'2022-05-04','15:10:15',E002,'Little Dude');
```

INSERT INTO

```
Cleaning(ProductID,Cleaning_Date,Cleaning_Time,EmployeeID,Employee_name)
VALUES (1006,'2022-01-05','10:00:10',E003,'Big Dude');
```



```
Rental.ProductID AS ProdID
FROM Customer
JOIN Rental ON rental.CustomerID = customer.CustomerID;
```



The screenshot shows a SQL query editor with the following query:

```
1 SELECT Customer.CustomerName as Name,
2 Rental.ProductID AS ProdID
3 FROM Customer
4 JOIN Rental ON rental.CustomerID = customer.CustomerID;
```

Below the query editor is a results grid with the following data:

Name	ProdID
John	1001
Tom	1002
Jerry	1003
Henry	1004
Scott	1005
Phil	1006
Jacob	1007
Jacob	1008

The results grid is labeled "Result 14" at the bottom.

Query 3:

Retrieving Payment details of customers where payment date is greater than return date 2022-02-05

```
SELECT *
FROM Payment
WHERE Payment_Date >
(SELECT Return_Date from ItemReturn
WHERE Return_Date = '2022-02-05');
```

