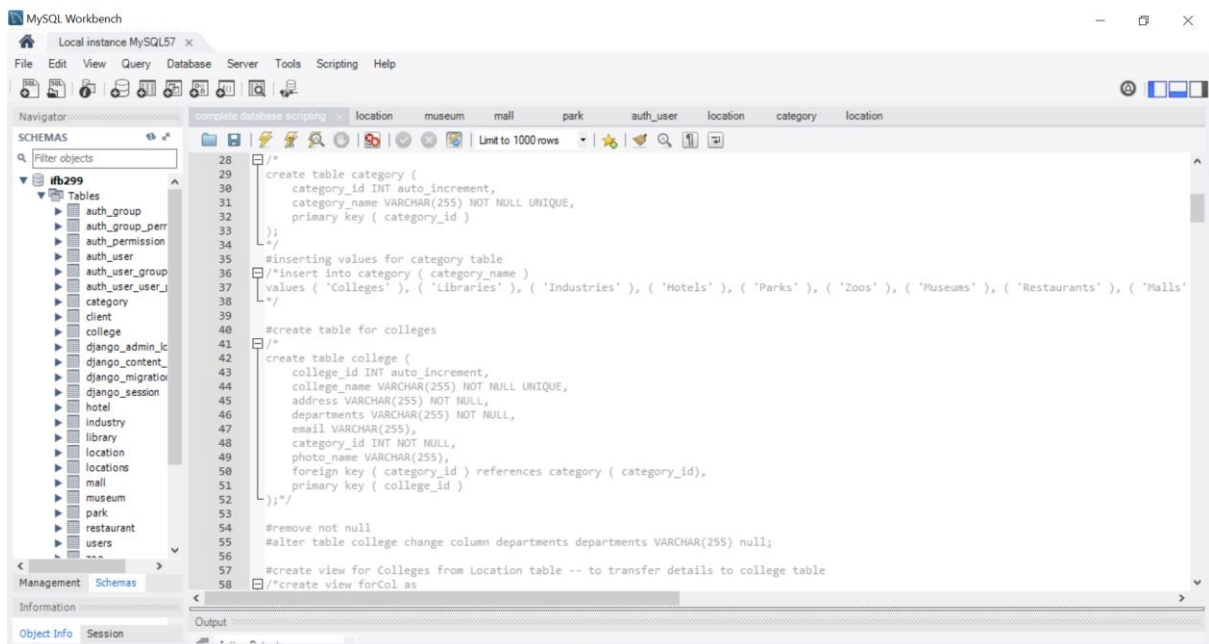
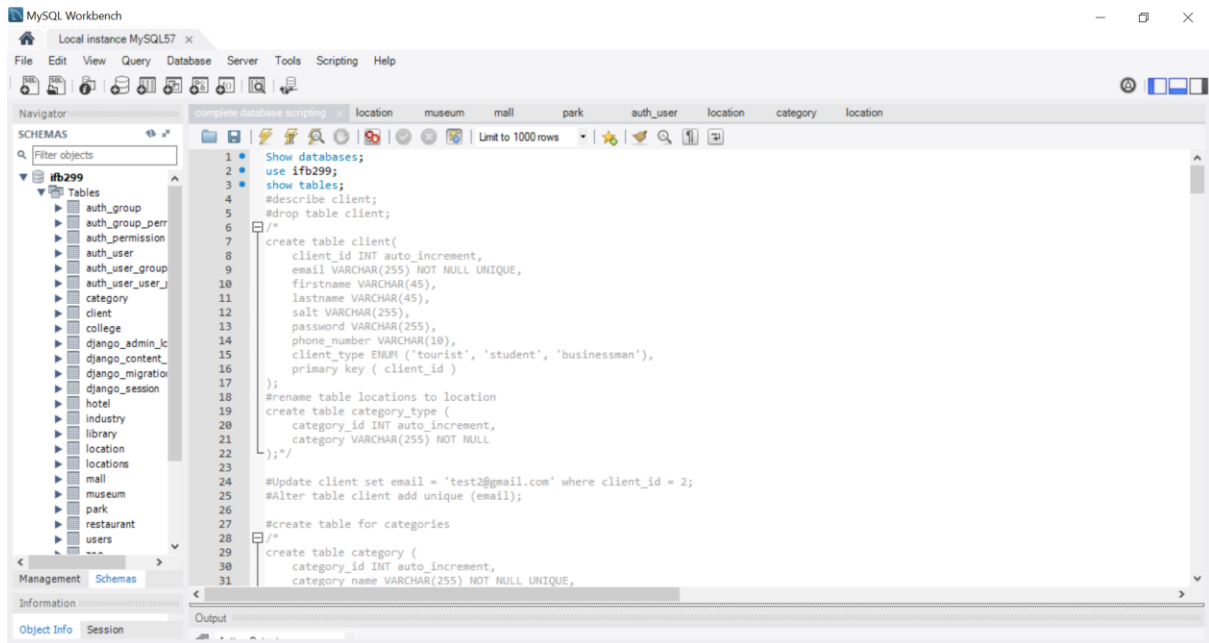
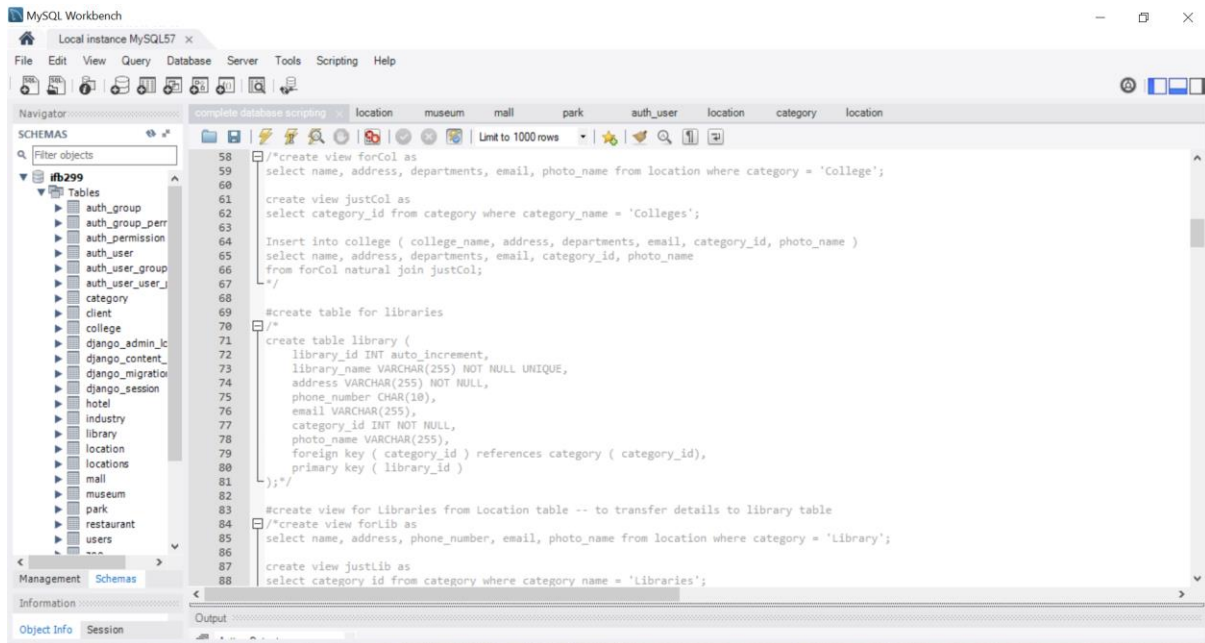


Screen capture of all database scripting trialled and completed

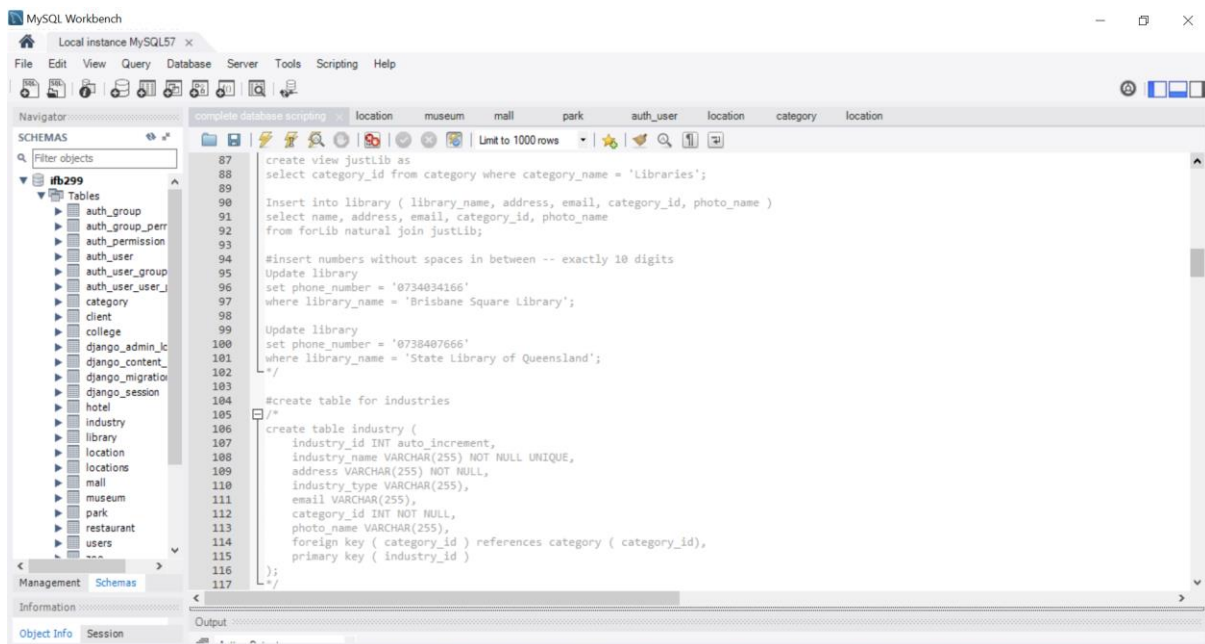


Screen capture of all database scripting trialled and completed



This screenshot shows the MySQL Workbench interface with a SQL script in the editor. The script includes the following SQL commands:

```
58 /*create view forCol as
59 select name, address, departments, email, photo_name from location where category = 'College';
60
61 create view justCol as
62 select category_id from category where category_name = 'Colleges';
63
64 Insert into college ( college_name, address, departments, email, category_id, photo_name )
65 select name, address, departments, email, category_id, photo_name
66 from forCol natural join justCol;
67
68
69 #create table for libraries
70
71 create table library (
72     library_id INT auto_increment,
73     library_name VARCHAR(255) NOT NULL UNIQUE,
74     address VARCHAR(255) NOT NULL,
75     phone_number CHAR(10),
76     email VARCHAR(255),
77     category_id INT NOT NULL,
78     photo_name VARCHAR(255),
79     foreign key ( category_id ) references category ( category_id),
80     primary key ( library_id )
81 );
82
83 #create view for Libraries from Location table -- to transfer details to library table
84
85 /*create view forLib as
86 select name, address, phone_number, email, photo_name from location where category = 'Library';
87
88 create view justLib as
89 select category_id from category where category_name = 'Libraries';
```



This screenshot shows the MySQL Workbench interface with a SQL script in the editor. The script includes the following SQL commands:

```
87 create view justLib as
88 select category_id from category where category_name = 'Libraries';
89
90 Insert into library ( library_name, address, email, category_id, photo_name )
91 select name, address, email, category_id, photo_name
92 from forLib natural join justLib;
93
94 #insert numbers without spaces in between -- exactly 10 digits
95 Update library
96 set phone_number = '0734034166'
97 where library_name = 'Brisbane Square Library';
98
99 Update library
100 set phone_number = '0738407666'
101 where library_name = 'State Library of Queensland';
102
103
104 #create table for Industries
105
106 create table industry (
107     industry_id INT auto_increment,
108     industry_name VARCHAR(255) NOT NULL UNIQUE,
109     address VARCHAR(255) NOT NULL,
110     industry_type VARCHAR(255),
111     email VARCHAR(255),
112     category_id INT NOT NULL,
113     photo_name VARCHAR(255),
114     foreign key ( category_id ) references category ( category_id),
115     primary key ( industry_id )
116 );
117
```

Screen capture of all database scripting trialled and completed

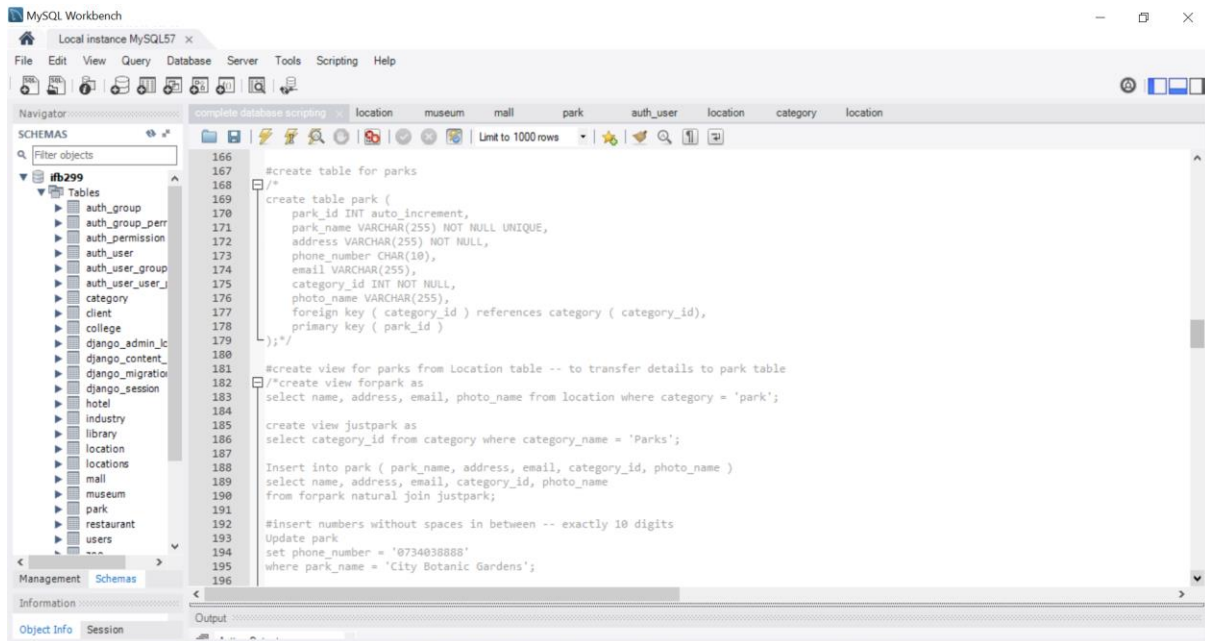
This screenshot shows the MySQL Workbench interface with a script titled 'complete database scripting'. The script contains SQL commands to create views and tables. The left sidebar shows the 'Schemas' pane with a tree view of tables including 'auth_group', 'auth_group_perr', 'auth_permission', 'auth_user', 'auth_user_group', 'auth_user_user_j', 'category', 'client', 'college', 'django_admin_lc', 'django_content_', 'django_migration', 'django_session', 'hotel', 'industry', 'library', 'location', 'locations', 'mail', 'museum', 'park', 'restaurant', and 'users'. The main editor shows the following SQL code:

```
117 L=/  
118 #create view for industries from Location table -- to transfer details to industry table  
119 /*create view forInd as  
120 select name, address, industry_type, email, photo_name from location where category = 'industry';  
121  
122 create view justInd as  
123 select category_id from category where category_name = 'Industries';  
124  
125 Insert into industry ( industry_name, address, industry_type, email, category_id, photo_name )  
126 select name, address, industry_type, email, category_id, photo_name  
127 from forInd natural join justInd;  
128 */  
129  
130 #create table for hotels  
131 /*  
132 create table hotel (  
133     hotel_id INT auto_increment,  
134     hotel_name VARCHAR(255) NOT NULL UNIQUE,  
135     address VARCHAR(255) NOT NULL,  
136     phone_number CHAR(10),  
137     email VARCHAR(255),  
138     photo_name VARCHAR(255),  
139     primary key ( hotel_id )  
140 );*/  
141  
142 #describe hotel;  
143  
144 #create view for hotels from Location table -- to transfer details to hotel table  
145 /*create view forHotel as  
146 select name, address, phone_number, email, photo_name from location where category = 'hotel';  
147
```

This screenshot shows the MySQL Workbench interface with a script titled 'complete database scripting'. The script contains SQL commands to create views, insert data, and create tables. The left sidebar shows the 'Schemas' pane with a tree view of tables including 'auth_group', 'auth_group_perr', 'auth_permission', 'auth_user', 'auth_user_group', 'auth_user_user_j', 'category', 'client', 'college', 'django_admin_lc', 'django_content_', 'django_migration', 'django_session', 'hotel', 'industry', 'library', 'location', 'locations', 'mail', 'museum', 'park', 'restaurant', and 'users'. The main editor shows the following SQL code:

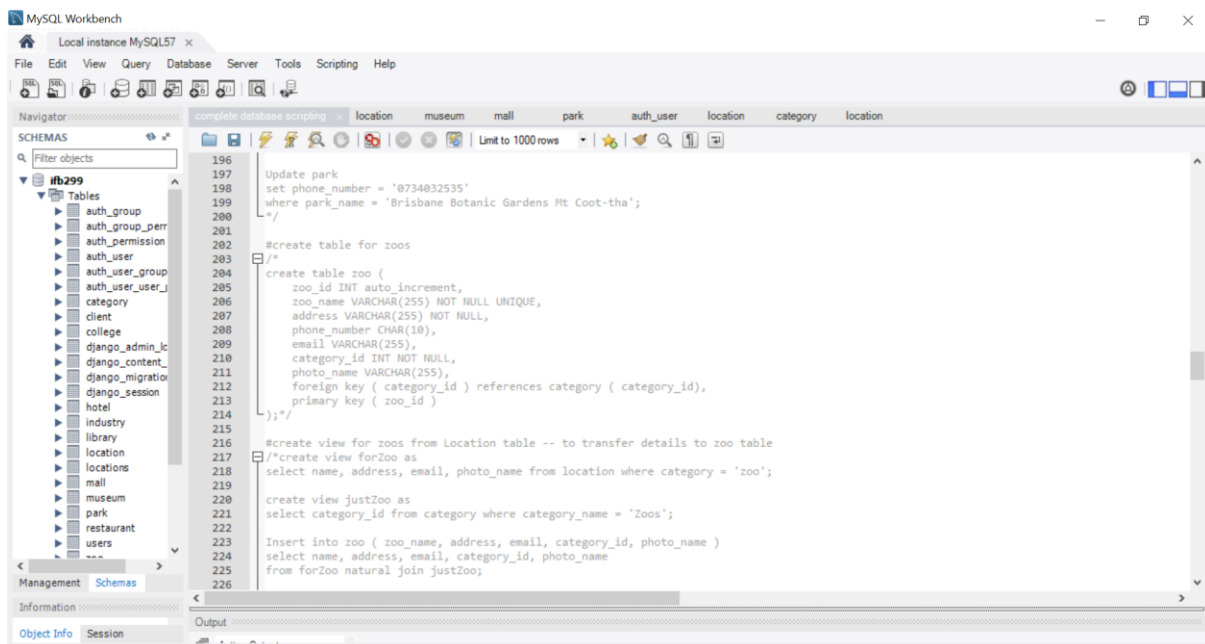
```
147  
148 create view justHotel as  
149 select category_id from category where category_name = 'Hotels';  
150  
151 Insert into hotel ( hotel_name, address, email, photo_name )  
152 select name, address, email, photo_name  
153 from forHotel;  
154  
155 #insert numbers without spaces in between -- exactly 10 digits  
156 Update hotel  
157 set phone_number = '0733068888'  
158 where hotel_name = 'Treasury Hotel';  
159  
160 Update hotel  
161 set phone_number = '0738536000'  
162 where hotel_name = 'The Sebel Quay West Brisbane';  
163  
164 select * from hotel;  
165 */  
166  
167 #create table for parks  
168 /*  
169 create table park (  
170     park_id INT auto_increment,  
171     park_name VARCHAR(255) NOT NULL UNIQUE,  
172     address VARCHAR(255) NOT NULL,  
173     phone_number CHAR(10),  
174     email VARCHAR(255),  
175     category_id INT NOT NULL,  
176     photo_name VARCHAR(255),  
177     foreign key ( category_id ) references category ( category_id ),
```

Screen capture of all database scripting trialled and completed



This screenshot shows the MySQL Workbench interface with a SQL script in the 'complete database scripting' window. The script includes comments and SQL commands for creating a 'park' table and a view for parks. The table 'park' has columns: park_id (INT auto increment), park_name (VARCHAR(255) NOT NULL UNIQUE), address (VARCHAR(255) NOT NULL), phone_number (CHAR(10)), email (VARCHAR(255)), category_id (INT NOT NULL), and photo_name (VARCHAR(255)). It also includes a foreign key constraint for category_id and a primary key for park_id. The view 'forpark' selects name, address, email, and photo_name from the 'location' table where category is 'park'. The script also includes an insert statement for the 'park' table and an update statement for the 'park' table.

```
166
167
168 #create table for parks
169 /*
170 create table park (
171     park_id INT auto increment,
172     park_name VARCHAR(255) NOT NULL UNIQUE,
173     address VARCHAR(255) NOT NULL,
174     phone_number CHAR(10),
175     email VARCHAR(255),
176     category_id INT NOT NULL,
177     photo_name VARCHAR(255),
178     foreign key ( category_id ) references category ( category_id),
179     primary key ( park_id )
180 );*/
181
182 #create view for parks from location table -- to transfer details to park table
183 /*create view forpark as
184 select name, address, email, photo_name from location where category = 'park';
185
186 create view justpark as
187 select category_id from category where category_name = 'Parks';
188
189 Insert into park ( park_name, address, email, category_id, photo_name )
190 select name, address, email, category_id, photo_name
191 from forpark natural join justpark;
192
193 #insert numbers without spaces in between -- exactly 10 digits
194 Update park
195 set phone_number = '0734038888'
196 where park_name = 'City Botanic Gardens';
197
```



This screenshot shows the MySQL Workbench interface with a SQL script in the 'complete database scripting' window. The script includes comments and SQL commands for creating a 'zoo' table and a view for zoos. The table 'zoo' has columns: zoo_id (INT auto increment), zoo_name (VARCHAR(255) NOT NULL UNIQUE), address (VARCHAR(255) NOT NULL), phone_number (CHAR(10)), email (VARCHAR(255)), category_id (INT NOT NULL), and photo_name (VARCHAR(255)). It also includes a foreign key constraint for category_id and a primary key for zoo_id. The view 'forZoo' selects name, address, email, and photo_name from the 'location' table where category is 'zoo'. The script also includes an insert statement for the 'zoo' table and an update statement for the 'zoo' table.

```
196
197
198 Update park
199 set phone_number = '0734032535'
200 where park_name = 'Brisbane Botanic Gardens Mt Coot-tha';
201
202 #create table for zoos
203 /*
204 create table zoo (
205     zoo_id INT auto increment,
206     zoo_name VARCHAR(255) NOT NULL UNIQUE,
207     address VARCHAR(255) NOT NULL,
208     phone_number CHAR(10),
209     email VARCHAR(255),
210     category_id INT NOT NULL,
211     photo_name VARCHAR(255),
212     foreign key ( category_id ) references category ( category_id),
213     primary key ( zoo_id )
214 );*/
215
216 #create view for zoos from location table -- to transfer details to zoo table
217 /*create view forZoo as
218 select name, address, email, photo_name from location where category = 'zoo';
219
220 create view justZoo as
221 select category_id from category where category_name = 'Zoos';
222
223 Insert into zoo ( zoo_name, address, email, category_id, photo_name )
224 select name, address, email, category_id, photo_name
225 from forZoo natural join justZoo;
226
```

Screen capture of all database scripting trialled and completed

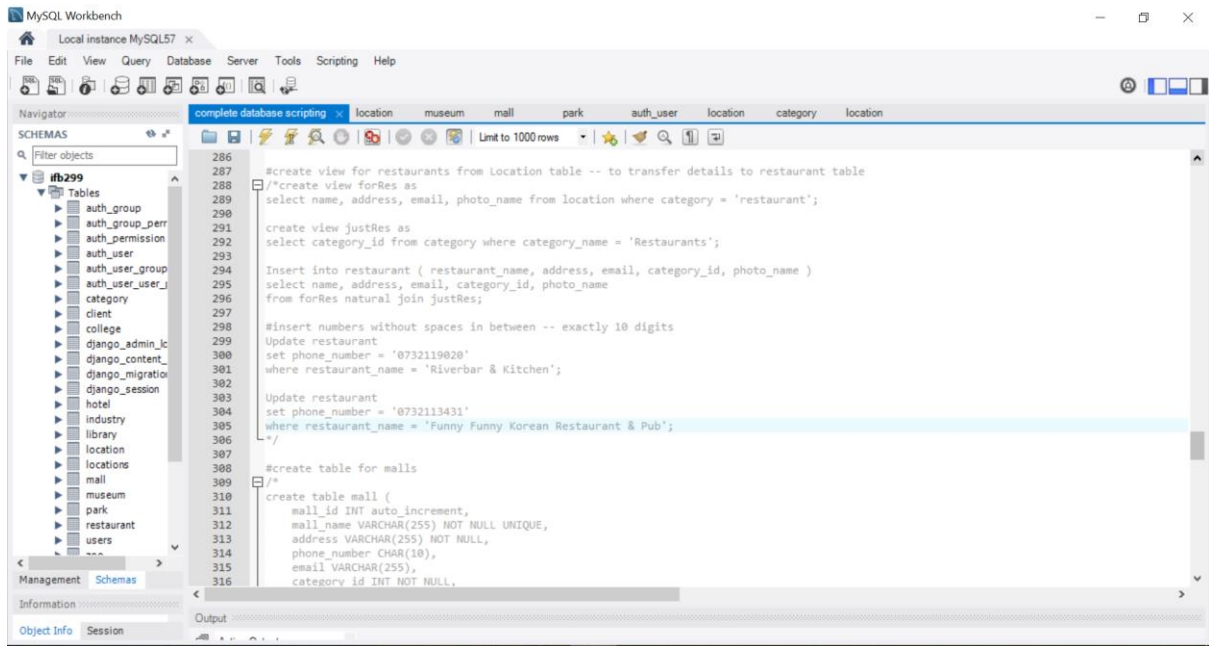
This screenshot shows the MySQL Workbench interface with a script editor containing SQL code for setting up a museum database. The left sidebar displays the 'Schemas' tree with a list of tables including 'auth_group', 'auth_group_permission', 'auth_permission', 'auth_user', 'auth_user_group', 'auth_user_user_j', 'category', 'client', 'college', 'django_admin_log', 'django_content_type', 'django_migration', 'django_session', 'hotel', 'industry', 'library', 'location', 'locations', 'mail', 'museum', 'park', 'restaurant', and 'users'. The main editor window shows the following SQL script:

```
226
227
228 #insert numbers without spaces in between -- exactly 10 digits
229 Update zoo
230 set phone_number = '0754362000'
231 where zoo_name = 'Australia Zoo';
232
233 Update zoo
234 set phone_number = '0755341266'
235 where zoo_name = 'Currumbin Wildlife Sanctuary';
236
237
238 /*
239 #create table for museums
240 */
241 create table museum (
242     museum_id INT auto_increment,
243     museum_name VARCHAR(255) NOT NULL UNIQUE,
244     address VARCHAR(255) NOT NULL,
245     phone_number CHAR(10),
246     email VARCHAR(255),
247     category_id INT NOT NULL,
248     photo_name VARCHAR(255),
249     foreign key ( category_id ) references category ( category_id ),
250     primary key ( museum_id )
251 );
252
253 /*create view forMuseum as
254 select name, address, email, photo_name from location where category = 'museum';
255
256 create view justMuseum as
257 select category_id from category where category_name = 'Museums';
```

This screenshot shows the MySQL Workbench interface with a script editor containing SQL code for setting up a restaurant database. The left sidebar displays the 'Schemas' tree with a list of tables including 'auth_group', 'auth_group_permission', 'auth_permission', 'auth_user', 'auth_user_group', 'auth_user_user_j', 'category', 'client', 'college', 'django_admin_log', 'django_content_type', 'django_migration', 'django_session', 'hotel', 'industry', 'library', 'location', 'locations', 'mail', 'museum', 'park', 'restaurant', and 'users'. The main editor window shows the following SQL script:

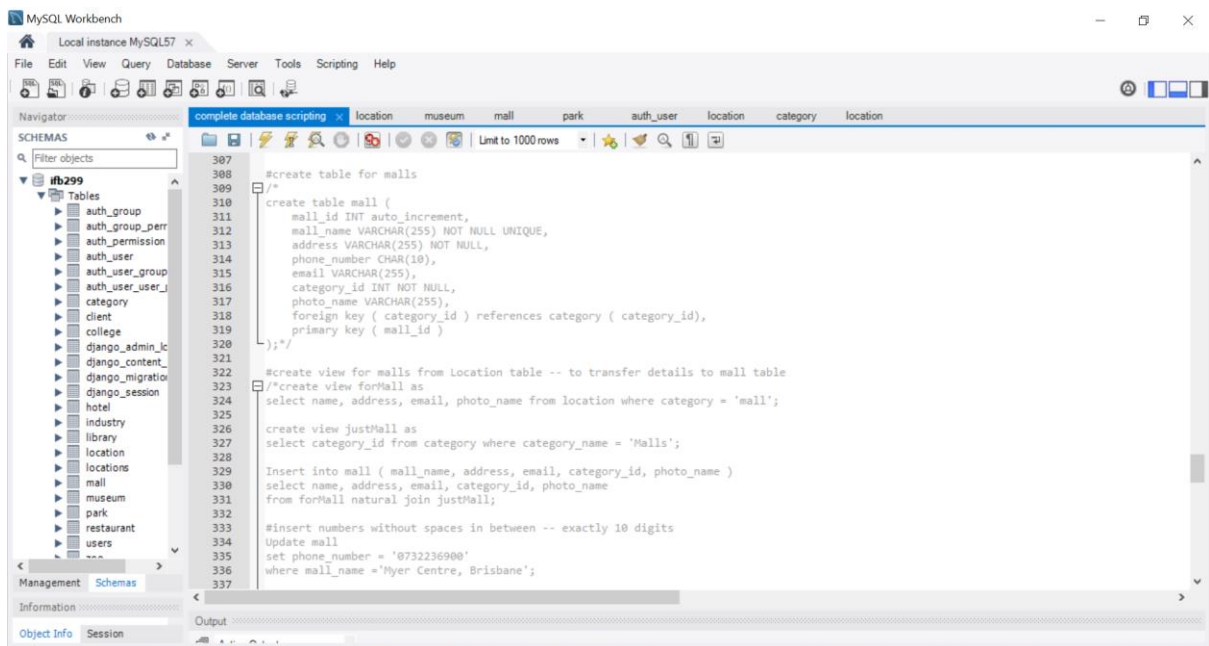
```
255
256
257 create view justMuseum as
258 select category_id from category where category_name = 'Museums';
259
260 Insert into museum ( museum_name, address, email, category_id, photo_name )
261 select name, address, email, category_id, photo_name
262 from forMuseum natural join justMuseum;
263
264 #insert numbers without spaces in between -- exactly 10 digits
265 Update museum
266 set phone_number = '0738407555'
267 where museum_name = 'Queensland Museum & Science Centre';
268
269 Update museum
270 set phone_number = '0732117052'
271 where museum_name = 'MacArthur Museum Brisbane';
272
273
274 /*
275 #create table for restaurant
276 */
277 create table restaurant (
278     restaurant_id INT auto_increment,
279     restaurant_name VARCHAR(255) NOT NULL UNIQUE,
280     address VARCHAR(255) NOT NULL,
281     phone_number CHAR(10),
282     email VARCHAR(255),
283     category_id INT NOT NULL,
284     photo_name VARCHAR(255),
285     foreign key ( category_id ) references category ( category_id ),
286     primary key ( restaurant_id )
287 );
288
```


Screen capture of all database scripting trialled and completed



This screenshot shows the MySQL Workbench interface with a SQL script for creating a database schema. The script is titled 'complete database scripting' and is located in the 'location' tab. The script includes the following SQL statements:

```
286
287
288 #create view for restaurants from Location table -- to transfer details to restaurant table
289 /*create view forRes as
290 select name, address, email, photo_name from location where category = 'restaurant';
291
292 create view justRes as
293 select category_id from category where category_name = 'Restaurants';
294
295 Insert into restaurant ( restaurant_name, address, email, category_id, photo_name )
296 select name, address, email, category_id, photo_name
297 from forRes natural join justRes;
298
299 #insert numbers without spaces in between -- exactly 10 digits
300 Update restaurant
301 set phone_number = '0732119820'
302 where restaurant_name = 'Riverbar & Kitchen';
303
304 Update restaurant
305 set phone_number = '0732113431'
306 where restaurant_name = 'Funny Funny Korean Restaurant & Pub';
307
308 */
309
310 #create table for malls
311 /*
312 create table mall (
313     mall_id INT auto increment,
314     mall_name VARCHAR(255) NOT NULL UNIQUE,
315     address VARCHAR(255) NOT NULL,
316     phone_number CHAR(10),
317     email VARCHAR(255),
318     category_id INT NOT NULL,
319 );
320
321 */
```



This screenshot shows the MySQL Workbench interface with a SQL script for creating a database schema. The script is titled 'complete database scripting' and is located in the 'location' tab. The script includes the following SQL statements:

```
307
308
309 #create table for malls
310 /*
311 create table mall (
312     mall_id INT auto increment,
313     mall_name VARCHAR(255) NOT NULL UNIQUE,
314     address VARCHAR(255) NOT NULL,
315     phone_number CHAR(10),
316     email VARCHAR(255),
317     category_id INT NOT NULL,
318     photo_name VARCHAR(255),
319     foreign key ( category_id ) references category ( category_id),
320     primary key ( mall_id )
321 );
322
323 */
324
325 #create view for malls from Location table -- to transfer details to mall table
326 /*create view forMall as
327 select name, address, email, photo_name from location where category = 'mall';
328
329 create view justMall as
330 select category_id from category where category_name = 'Malls';
331
332 Insert into mall ( mall_name, address, email, category_id, photo_name )
333 select name, address, email, category_id, photo_name
334 from forMall natural join justMall;
335
336 #insert numbers without spaces in between -- exactly 10 digits
337 Update mall
338 set phone_number = '0732236900'
339 where mall_name = 'Myer Centre, Brisbane';
340
341 */
```

Screen capture of all database scripting trialled and completed

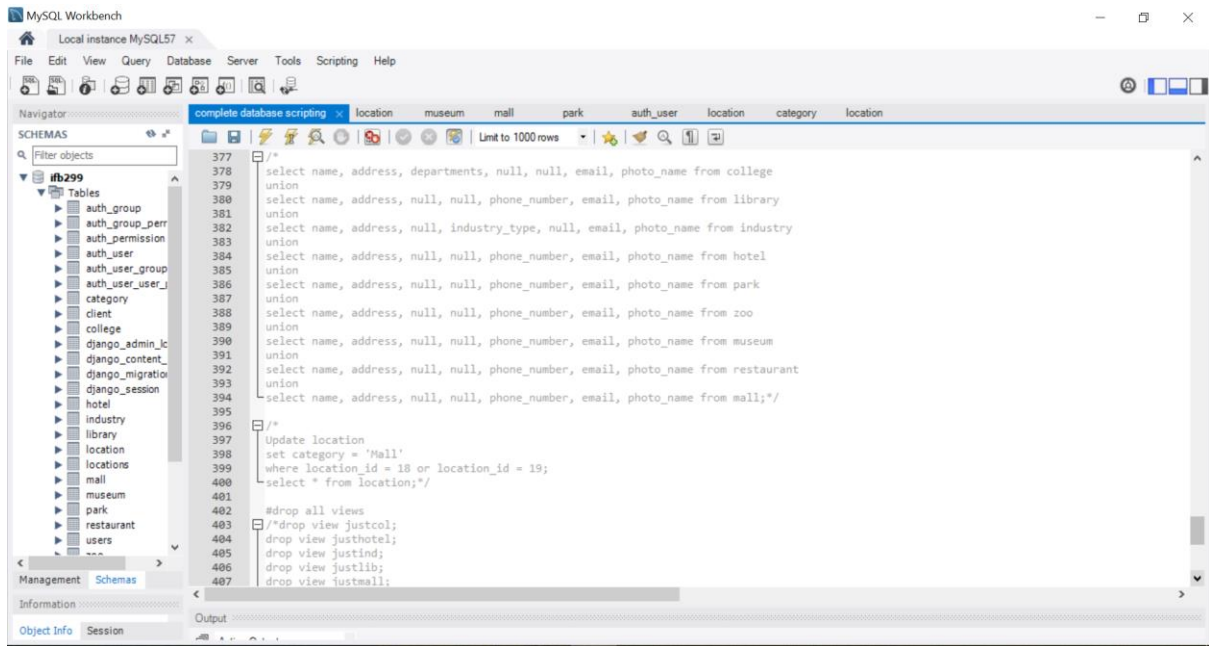
This screenshot shows the MySQL Workbench interface with the 'complete database scripting' window open. The left sidebar displays the 'SCHEMAS' tree with a search filter 'fb299'. The main editor area contains SQL code for creating and managing a database. The code includes comments and SQL statements for creating a summary table, altering existing tables, and deleting a category table.

```
337 Update mall
338 set phone_number = '0730066290'
339 where mall_name = 'Queen Street Mall';*/
340
341
342 #syntax to change all category_name column to just name
343 /*alter table hotel change 'hotel_name' 'name' VARCHAR(255) NOT NULL;
344 describe hotel;*/
345
346 #reuse queries below to remove foreign keys in individual category table
347 /*show create table museum;
348 alter table hotel drop foreign key hotel_ibfk_1;
349 alter table hotel drop column category_id;
350 select * from hotel;*/
351
352 #delete category table
353 #SHOW ENGINE INNODB STATUS;
354 #drop table category;
355
356 /*
357 #create a summary table of all locations
358 create table locationSummary (
359 location_id INT auto increment,
360 name VARCHAR(255) NOT NULL,
361 category VARCHAR (255) NOT NULL,
362 primary key ( location_id ),
363 foreign key college_id_fk ( name ) references college ( name ) ON UPDATE CASCADE ON DELETE NO ACTION,
364 foreign key library_id_fk ( name ) references library ( name ) ON UPDATE CASCADE ON DELETE NO ACTION,
365 foreign key industry_id_fk ( name ) references industry ( name ) ON UPDATE CASCADE ON DELETE NO ACTION,
366 foreign key hotel_id_fk ( name ) references hotel ( name ) ON UPDATE CASCADE ON DELETE NO ACTION,
367 foreign key park_id_fk ( name ) references park ( name ) ON UPDATE CASCADE ON DELETE NO ACTION,
```

This screenshot shows the MySQL Workbench interface with the 'complete database scripting' window open. The left sidebar displays the 'SCHEMAS' tree with a search filter 'fb299'. The main editor area contains SQL code for creating and managing a database. The code includes comments and SQL statements for creating a summary table, inserting data, and selecting data from various tables.

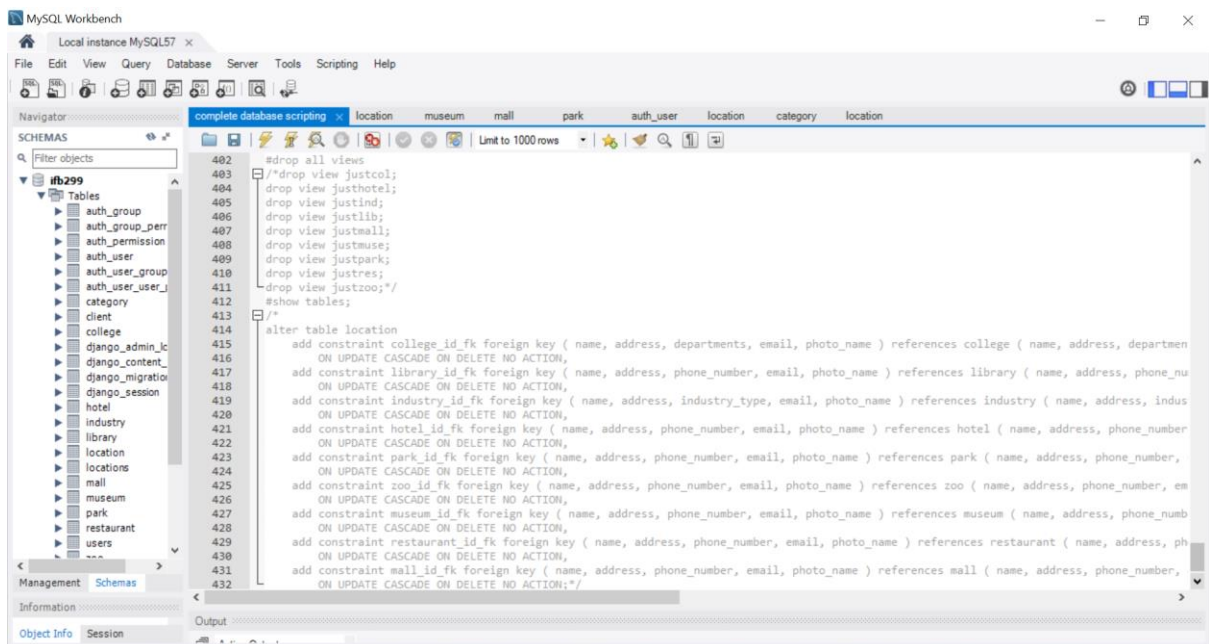
```
357 #create a summary table of all locations
358 create table locationSummary (
359 location_id INT auto increment,
360 name VARCHAR(255) NOT NULL,
361 category VARCHAR (255) NOT NULL,
362 primary key ( location_id ),
363 foreign key college_id_fk ( name ) references college ( name ) ON UPDATE CASCADE ON DELETE NO ACTION,
364 foreign key library_id_fk ( name ) references library ( name ) ON UPDATE CASCADE ON DELETE NO ACTION,
365 foreign key industry_id_fk ( name ) references industry ( name ) ON UPDATE CASCADE ON DELETE NO ACTION,
366 foreign key hotel_id_fk ( name ) references hotel ( name ) ON UPDATE CASCADE ON DELETE NO ACTION,
367 foreign key park_id_fk ( name ) references park ( name ) ON UPDATE CASCADE ON DELETE NO ACTION,
368 foreign key zoo_id_fk ( name ) references zoo ( name ) ON UPDATE CASCADE ON DELETE NO ACTION,
369 foreign key museum_id_fk ( name ) references museum ( name ) ON UPDATE CASCADE ON DELETE NO ACTION,
370 foreign key restaurant_id_fk ( name ) references restaurant ( name ) ON UPDATE CASCADE ON DELETE NO ACTION,
371 foreign key mall_id_fk ( name ) references mall ( name ) ON UPDATE CASCADE ON DELETE NO ACTION
372 );*/
373
374 #INSERT INTO locationSummary (name, category) VALUES
375 # ( (SELECT name from college WHERE college_id=1), 'College' );
376
377 /*
378 select name, address, departments, null, null, email, photo_name from college
379 union
380 select name, address, null, null, phone_number, email, photo_name from library
381 union
382 select name, address, null, industry_type, null, email, photo_name from industry
383 union
384 select name, address, null, null, phone_number, email, photo_name from hotel
385 union
386 select name, address, null, null, phone_number, email, photo_name from park
387 union
```

Screen capture of all database scripting trialled and completed



This screenshot shows the MySQL Workbench interface with a SQL script for database setup. The script includes several SELECT statements for data insertion and an UPDATE statement for a specific location. The left sidebar shows the database schema with tables like auth_group, auth_group_permission, auth_permission, auth_user, auth_user_group, auth_user_user_j, category, client, college, django_admin_log, django_content_type, django_migration, django_session, hotel, industry, library, location, locations, mail, museum, park, restaurant, and users.

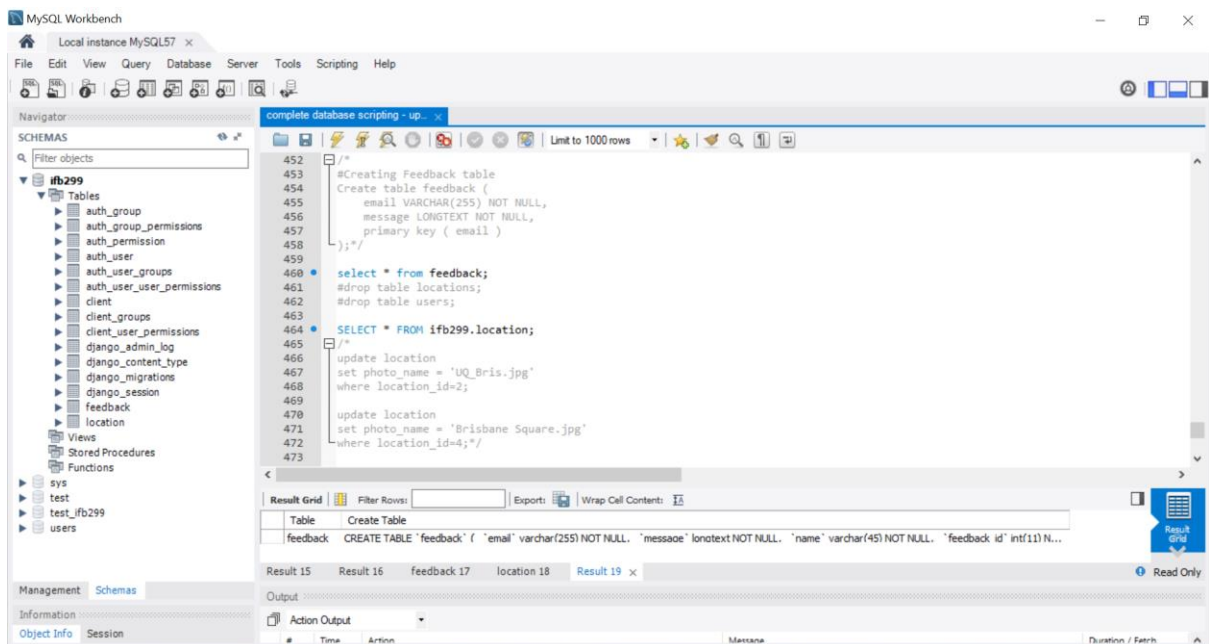
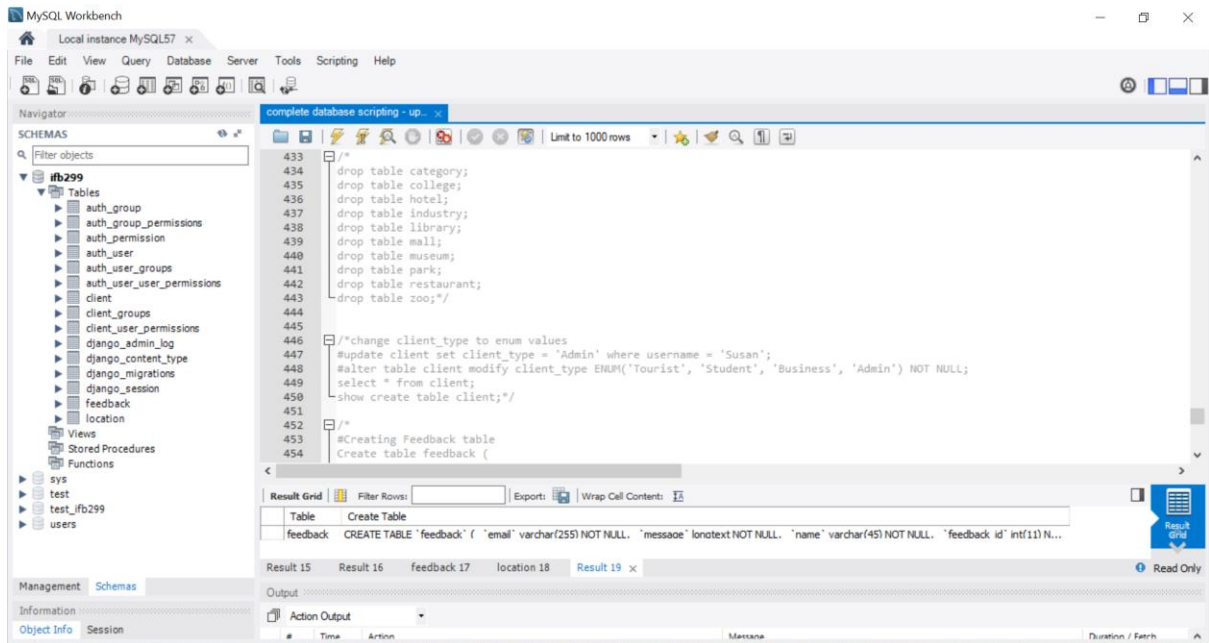
```
377 /*
378 select name, address, departments, null, null, email, photo_name from college
379 union
380 select name, address, null, null, phone_number, email, photo_name from library
381 union
382 select name, address, null, industry_type, null, email, photo_name from industry
383 union
384 select name, address, null, null, phone_number, email, photo_name from hotel
385 union
386 select name, address, null, null, phone_number, email, photo_name from park
387 union
388 select name, address, null, null, phone_number, email, photo_name from zoo
389 union
390 select name, address, null, null, phone_number, email, photo_name from museum
391 union
392 select name, address, null, null, phone_number, email, photo_name from restaurant
393 union
394 select name, address, null, null, phone_number, email, photo_name from mail;*/
395
396 /*
397 update location
398 set category = 'Hall'
399 where location_id = 18 or location_id = 19;
400 select * from location;*/
401
402 #drop all views
403 /*drop view justcol;
404 drop view justhotel;
405 drop view justind;
406 drop view justlib;
407 drop view justmail;
408 drop view justmuse;
409 drop view justpark;
410 drop view justres;
411 drop view justzoo;*/
412
413 #show tables;
414
415 /*
416 alter table location
417 add constraint college_id_fk foreign key ( name, address, departments, email, photo_name ) references college ( name, address, departmen
418 ON UPDATE CASCADE ON DELETE NO ACTION,
419 add constraint library_id_fk foreign key ( name, address, phone_number, email, photo_name ) references library ( name, address, phone_nu
420 ON UPDATE CASCADE ON DELETE NO ACTION,
421 add constraint industry_id_fk foreign key ( name, address, industry_type, email, photo_name ) references industry ( name, address, indus
422 ON UPDATE CASCADE ON DELETE NO ACTION,
423 add constraint hotel_id_fk foreign key ( name, address, phone_number, email, photo_name ) references hotel ( name, address, phone_number
424 ON UPDATE CASCADE ON DELETE NO ACTION,
425 add constraint zoo_id_fk foreign key ( name, address, phone_number, email, photo_name ) references zoo ( name, address, phone_number, em
426 ON UPDATE CASCADE ON DELETE NO ACTION,
427 add constraint museum_id_fk foreign key ( name, address, phone_number, email, photo_name ) references museum ( name, address, phone_numb
428 ON UPDATE CASCADE ON DELETE NO ACTION,
429 add constraint restaurant_id_fk foreign key ( name, address, phone_number, email, photo_name ) references restaurant ( name, address, ph
430 ON UPDATE CASCADE ON DELETE NO ACTION,
431 add constraint mail_id_fk foreign key ( name, address, phone_number, email, photo_name ) references mail ( name, address, phone_number,
432 ON UPDATE CASCADE ON DELETE NO ACTION;*/
```



This screenshot shows the MySQL Workbench interface with a SQL script for database setup. The script includes several DROP statements for views and an ALTER TABLE statement for the location table. The left sidebar shows the database schema with tables like auth_group, auth_group_permission, auth_permission, auth_user, auth_user_group, auth_user_user_j, category, client, college, django_admin_log, django_content_type, django_migration, django_session, hotel, industry, library, location, locations, mail, museum, park, restaurant, and users.

```
402 #drop all views
403 /*drop view justcol;
404 drop view justhotel;
405 drop view justind;
406 drop view justlib;
407 drop view justmail;
408 drop view justmuse;
409 drop view justpark;
410 drop view justres;
411 drop view justzoo;*/
412
413 #show tables;
414
415 /*
416 alter table location
417 add constraint college_id_fk foreign key ( name, address, departments, email, photo_name ) references college ( name, address, departmen
418 ON UPDATE CASCADE ON DELETE NO ACTION,
419 add constraint library_id_fk foreign key ( name, address, phone_number, email, photo_name ) references library ( name, address, phone_nu
420 ON UPDATE CASCADE ON DELETE NO ACTION,
421 add constraint industry_id_fk foreign key ( name, address, industry_type, email, photo_name ) references industry ( name, address, indus
422 ON UPDATE CASCADE ON DELETE NO ACTION,
423 add constraint hotel_id_fk foreign key ( name, address, phone_number, email, photo_name ) references hotel ( name, address, phone_number
424 ON UPDATE CASCADE ON DELETE NO ACTION,
425 add constraint zoo_id_fk foreign key ( name, address, phone_number, email, photo_name ) references zoo ( name, address, phone_number, em
426 ON UPDATE CASCADE ON DELETE NO ACTION,
427 add constraint museum_id_fk foreign key ( name, address, phone_number, email, photo_name ) references museum ( name, address, phone_numb
428 ON UPDATE CASCADE ON DELETE NO ACTION,
429 add constraint restaurant_id_fk foreign key ( name, address, phone_number, email, photo_name ) references restaurant ( name, address, ph
430 ON UPDATE CASCADE ON DELETE NO ACTION,
431 add constraint mail_id_fk foreign key ( name, address, phone_number, email, photo_name ) references mail ( name, address, phone_number,
432 ON UPDATE CASCADE ON DELETE NO ACTION;*/
```


Screen capture of all database scripting trialled and completed



Screen capture of all database scripting trialled and completed

