	Technical Point #	Query/Task	FileName	Line Number	Result to be shown	Comments/Thought
1	The project should be written using Python and MySQL, and should use Flask with Flask-MySQLAlchemy		forms.py routes.py			
2	add css and/or javascript		static/css static/js static/vendor			files used throughout the application
3	Include at least one structural change to the database using DDL. (Create, Alter, Drop)		ExcelSchemav2.0 ExcelSchemav3.0			version 2.0 and version 3.0 of excel schema show changes to database. https://github.com/nikhilagrwl07/ecommerce-flask-python/tree/master/ddl
4	Provide the DDL as well as the INSERT SQL for creating the tables and initially populating the database.		DDL-v2.0.sql Insert.sql			https://github.com/nikhilagrwl07/ecommerce-flask-python/tree/master/ddl
5	Include at least one insertion of a new record that will occur during the execution of the application	orderedproduct=OrderedProduct(orderid=orderid, productid=item.productid, quantity=item.quantity)	forms.py			https://github.com/nikhilagrwl07/ecommerce-flask-python/blob/master/ecommerce/forms.py#L291
		salesTransaction = SaleTransaction (orderid=orderid, transaction_date=datetime.utcnow(), amount=totalsum, cc_number=ccnumber, cc_type=cctype, response="success")	forms.py			https://github.com/nikhilagrwl07/ecommerce-flask-python/blob/master/ecommerce/forms.py#L308
		order = Order (order_date=orderdate, total_price=totalsum, userid=userId)	forms.py			https://github.com/nikhilagrwl07/ecommerce-flask-python/blob/master/ecommerce/forms.py#L265

6	Include at least one update of a record--changing an existing record, not adding a new one. Use SQLAlchemy.	if quantity is not None:
		cart = Cart (userid=userId, productid=productId, quantity=quantity[0] + 1)
		else:
		cart = Cart (userid=userId, productid=productId, quantity=1)
		
		db.session.merge (cart)
			forms.py			https://github.com/nikhilagrwl07/ecommerce-flask-python/blob/master/ecommerce/forms.py#L139

7	Include at least one delete of a record. Use SQLAlchemy.	db.session.query(Cart).filter(Cart.userid == userid).delete()	forms.py			https://github.com/nikhilagrwl07/ecommerce-flask-python/blob/master/ecommerce/forms.py#L301
8	Include at least one simple SELECT SQL statement. Use regular SQL for this	check if user exists	forms.py			https://github.com/nikhilagrwl07/ecommerce-flask-python/blob/master/ecommerce/forms.py#L56

9	Include one query using Flask-SQLAlchemy filter or filter_by.	cart=Cart.query. with_entities(Cart. productid,Cart.quantity). filter(Cart.userid == userId)	forms.py			gets cart data by userid
10	Inlcude at least one SELECT using an aggregate function. Use regualr SQL for this.		routes.py			https://github.com/nikhilagrwl07/ecommerce-flask-python/blob/master/ecommerce/routes.py#L177
11	Include at least one SELECT using a compound condition using regular SQL, and also the equivalent of a compound select using Flask-SQLAlchemy	productDetailsByCategoryId = Product.query.join (ProductCategory, Product.productid == ProductCategory. productid) \ .add_columns (Product.productid, Product.product_name, Product.regular_price, Product. discounted_price, Product. image) \ .join(Category, Category.categoryid == ProductCategory. categoryid) \ .filter(Category. categoryid == int (categoryId)) \ .add_columns (Category. category_name) \ .all()	routes.py	69		display all products belonging to selected category on category display page
12	Include at least one JOIN query using SQL, and also one using Flask-SQLAlchemy.	productsincart = Product.query.join(Cart, Product.productid == Cart.productid) \ .add_columns (Product.productid, Product.product_name, Product. discounted_price, Cart. quantity) \ .add_columns (Product. discounted_price * Cart. quantity).filter(Cart.userid == userId)	forms.py	162		https://github.com/nikhilagrwl07/ecommerce-flask-python/blob/master/ecommerce/forms.py#L172
13	Include at least one subquery. Regular SQL. Excellence points if you also use Flask-SQLAlchemy.	subqury = Cart.query. filter(Cart.userid == userId).filter(Cart. productid == productId). subquery() qry = db.session. query(Cart.quantity). select_entity_from (subqury).all()	https://github.com/nikhilagrwl07/ecommerce-flask-python/blob/master/ecommerce/forms.py#L125	125		extracting pre existing quantity of a productId belonging to Userid of Cart

14	Use a form to collect user data, as shown in our CRUD labs.	extractOrderdetails	forms.py	219		extract order details from checkout page
15	Populate a field on a form or table from the database. This would most likely be for your update, and you can model this directly off of our examples in class.	update_product update_category	forms.py routes.py	https://github.com/nikhilagrwl07/ecommerce-flask-python/blob/master/ecommerce/routes.py#L238	Update completed successfully	forms for updating products and categories
16	Check for empty data fields. You can use the built-in validations for this.		checkoutpage.html			checks added for empty fields.
17	Implement referential intergrity. Demonstrate what happens when it is violated.	delete_product	routes.py	https://github.com/nikhilagrwl07/ecommerce-flask-python/blob/master/ecommerce/routes.py#L273	Product is deleted successfully	Delete the entry in product_category and cart tables where productid is the same with the product that we're deleting (foreign key)
18	Use an appropriate structure for your project package.	https://github.com/nikhilagrwl07/ecommerce-flask-python	https://github.com/nikhilagrwl07/ecommerce-flask-python/tree/master/ecommerce			followed modular folder based project structure
Excellence points						
1	implemented dropdown boxes and custom jquery validations		checkoutpage.html checkoutvalidate.js	https://github.com/nikhilagrwl07/ecommerce-flask-python/blob/master/ecommerce/static/js/checkoutvalidate.js		
2	implemented email functionality using smtp lib		forms.py	https://github.com/nikhilagrwl07/ecommerce-flask-python/blob/master/ecommerce/forms.py#L317		
3	in user registration page using javacript for showing dependent dropdown for country, state, city, zipcode		register.html	https://github.com/nikhilagrwl07/ecommerce-flask-python/blob/master/ecommerce/static/js/validateForm.js#L67		
4	On user registration page all validation is controlled by custom written javacript checks		validateForm.js	https://github.com/nikhilagrwl07/ecommerce-flask-python/blob/master/ecommerce/static/js/validateForm.js		
5	text confirmation functionality using smtplib emailsms gateway		forms.py	https://github.com/nikhilagrwl07/ecommerce-flask-python/blob/master/ecommerce/forms.py#L317		
6	using flask-sqlalchemy subquery		routes.py	https://github.com/nikhilagrwl07/ecommerce-flask-python/blob/master/ecommerce/forms.py#L136		

7	Graph representation using plotly		routes.py	https://github.com/nikhilagrwl07/ecommerce-flask-python/blob/master/ecommerce/routes.py#L330		